Benelearn 2005

Annual Machine Learning Conference of Belgium and the Netherlands

CTIT Proceedings of the fourteenth
Annual Machine Learning Conference
of Belgium and the Netherlands

Martijn van Otterlo, Mannes Poel and Anton Nijholt (eds.)

CIP GEGEVENS KONINKLIJKE BIBLIOTHEEK, DEN HAAG

Otterlo, van. M., Poel, M., Nijholt, A.

Benelearn 2005
Proceedings of the Fourteenth Annual Machine Learning Conference of Belgium and the Netherlands
M. van Otterlo, M. Poel, A. Nijholt (eds.)
Enschede, Universiteit Twente, Faculteit Elektrotechniek, Wiskunde en Informatica

ISSN 0929–0672
CTIT Workshop Proceedings Series WP05-03

trefwoorden: machine learning, neural networks, adaptive systems,
computational learning theory.

Book orders:
Ms. C. Bijron
University of Twente
Faculty of Electrical Engineering, Mathematics and Computer Science
Human Media Interaction
P.O. Box 217
NL 7500 AE Enschede
tel: +31 53 4893680
fax: +31 53 4893503
Email: bijron@cs.utwente.nl

Druk- en bindwerk: Reprografie U.T. Service Centrum, Enschede

Preface

It is a pleasure for the Human Media Interaction (HMI) Group of the University of Twente to welcome you in Enschede for the Benelearn 2005 conference. We are very happy to be able to present to you a wide variety of work, from a wide variety of researchers in machine learning. In personal communication, Professor Nicholas Findler suggested the name "Beneluxlearn", and although original and clever, given the wide variety of originating countries this year, this might not even be general enough.

Benelearn 2005 is the fourteenth conference in a long tradition of events in which researchers from the Netherlands, Belgium and other countries gather to present and discuss their recent work in machine learning. Given the prominent role Benelux researchers play in the European and international area of machine learning, as well as the growing interest to learn and mine in the context of increasing numbers of datasets and applications, we hope that Benelearn will continue to function as a useful machine learning forum.

This year's major themes are computational learning theory, reinforcement learning, evolution and datamining. Multiple authors take a principled approach, grounded in theory and statistics. This is in line with recent trends in the machine learning community. Applications of machine learning that are addressed during the conference range from facial recognition and microarrays analysis to speaker head orientations in meetings and forest fire control.

We are very pleased to have the opportunity to welcome two invited speakers this year. Kristian Kersting, from the University of Freiburg in Germany, will talk about "Probabilistic Logic Learning and Reasoning" in which a growing direction in current machine learning – that of combining learning, probability and relational (or logical) representation languages – will be addressed. Samy Bengio, from the IDIAP Research Institute in Martigny, Switzerland, will discuss "Machine Learning Challenges for Multi-Modal Processing" identifying lots of machine learning challenges in areas such as meeting modelling and other multi-modal interaction domains.

Benelearn 2005 has been made possible with financial support from NWO (Netherlands Organization for Scientific Research), SIKS (Dutch Resarch School for Information and Knowledge Systems), IOP-MMI (Senter, Ministry of Economic Affairs) and CTIT's (Centre of Telematics and Information Technology) Special Research Objective NICE (Natural Interaction in Computer-mediated Environments). We are grateful to all these supporting organizations.

At the University of Twente several people have been involved in the organization of this conference. Using their experience gathered from the organization of many other workshops and events, Charlotte Bijron, Alice Vissers and Lynn Packwood have done a terrific job in managing all organizational and financial matters. Hendri Hondorp has proved once again capable of professionally managing and LaTeX-editing the proceedings, in addition to his efforts on creating a high-quality web page. The research institute CTIT has given us permission to publish the proceedings in her CTIT Proceedings series. The organizers of this conference would like to thank the programme committee members and reviewers, the authors of the submitted papers and all those who have helped bringing about Benelearn 2005.

Martijn van Otterlo, Mannes Poel and Anton Nijholt — February 2005

Previous Benelearn conferences

Previous Benelearn conferences were:

Benelearn 2004	Brussels, Belgium
Benelearn 2002	Utrecht, The Netherlands
Benelearn 2001	Antwerp, Belgium
Benelearn 2000	Tilburg, The Netherlands
Benelearn 1999	Leuven, Belgium
Benelearn 1998	Wageningen, The Netherlands
Benelearn 1997	Tilburg, The Netherlands
Benelearn 1996	Maastricht, The Netherlands
Benelearn 1995	Brussels, Belgium
Benelearn 1994	Rotterdam, The Netherlands
Benelearn 1993	Brussels, Belgium
Benelearn 1991	Amsterdam, The Netherlands
Benelearn 1990	Leuven, Belgium

Programme Committee Benelearn 2005

H. Bersini, (Université Libre de Bruxelles)
W. Daelemans, (Universiteit Antwerpen (UIA))
B. De Moor, (K.U. Leuven)
A. Feelders, (Universiteit Utrecht)
M. Hutter, (Istituto Dalle Molle di Studi sull'Intelligenza Artificiale)
B. Naudts, (Universiteit Antwerpen (RUCA))
M. Pantic, (Universiteit Delft)
S. ten Hagen, (Universiteit Amsterdam)
M. Van Someren, (Universiteit Amsterdam)
H. Blockeel, (K.U. Leuven)
B. De Baets, (Universiteit Gent)
P. Dupont, (Université catholique de Louvain)
P. Flach, (University of Bristol)
T. Lenaerts, (Universite Libre de Bruxelles)
A. Nowe, (Vrije Universiteit Brussel)
E. Postma, (Universiteit Maastricht)
D. Thierens, (Universiteit Utrecht)
N. Vlassis, (Universiteit Amsterdam)
M. Wiering, (Universiteit Utrecht)

Benelearn 2005 Programme Chairs

Martijn van Otterlo, Mannes Poel and Anton Nijholt

Benelearn 2005 Local Organization

Charlotte Bijron and Alice Vissers
(Programme Committee Secretary)

Hendri Hondorp (Proceedings LaTeX Editor)
Lynn Packwood (Financial)

Sponsors

CTIT Centre for Telematics and Information Technology [1]
NWO Netherlands Organisation for Scientific Research [2]
SIKS [3]
IOP [4]

[1] http://www.ctit.utwente.nl
[2] http://www.nwo.nl
[3] http://www.siks.nl
[4] http://www.iop.nl

Contents

Invited Speakers

Regular Speakers

Multi-Channel Sequence Processing

Samy Bengio BENGIO@IDIAP.CH
IDIAP Research Institute, Martigny, Switzerland

Hervé Bourlard BOURLARD@IDIAP.CH
IDIAP Research Institute and Swiss Federal Institute of Technology at Lausanne (EPFL), Switzerland

Abstract

This paper summarizes some of the current research challenges arising from multi-channel sequence processing. Indeed, multiple real life applications involve simultaneous recording and analysis of multiple information sources, which may be asynchronous, have different frame rates, exhibit different stationarity properties, and carry complementary (or correlated) information. Some of these problems can already be tackled by one of the many statistical approaches towards sequence modeling. However, several challenging research issues are still open, such as taking into account asynchrony and correlation between several feature streams, or handling the underlying growing complexity. In this framework, we discuss here two novel approaches, which recently started to be investigated with success in the context of large multimodal problems. These include the asynchronous HMM, providing a principled approach towards the processing of multiple feature streams, and the layered HMM approach, providing a good formalism for decomposing large and complex (multi-stream) problems into layered architectures. As briefly reported here, combination of these two approaches yielded successful results on several multi-channel tasks, ranging from audio-visual speech recognition to automatic meeting analysis.

1. Introduction

Given the proliferation of electronic recording devices (cameras, microphones, EEGs, etc) with ever cheaper, and ever increasing processing speed, storage, and bandwidth, together with the advances in automatically extracting and managing information recorded from these devices (such as speech recognition, face tracking, etc), it becomes more and more feasible to simultaneously capture a same event (or multiple events) with several devices, generating richer and more robust sets of feature-streams.

Modeling such data coming from multiple channels (thus resulting in multiple observation streams) is the goal of *multi-channel sequence processing*. Examples of practical applications of this field are numerous, such as audio-visual speech recognition, which can be more robust to ambient noise than only using an audio stream. While several statistical models were presented recently in the literature to cope with this growing amount of data accessible in parallel, several open research problems are still to be solved. The purpose of this paper is thus to discuss some of these solutions, and specifically addressing two important issues, i.e., *asynchrony* (when the feature streams are supposed to be piecewise stationary, but with different stationary properties) and *complexity* (when it is furthermore necessary to split the problem into several multi-stream sub-problems).

The outline of the paper is as follows. Section 2 justifies the need for multi-channel sequence processing by discussing some of the numerous applications that require such a framework. Section 3 reviews some of the current models used in the literature. Section 4 shows that despite all these models, there is still room for several improvements. Section 5 proposes a model to handle temporal asynchrony between channels, while Section 6 proposes a principled approach to control the complexity of multi-channel sequence processing through "optimal" hierarchical processing.

2. Some Applications

Several tasks that are currently handled with only one stream of information could in fact benefit from the addition of other parallel streams. Furthermore, like

in speech recognition (as well as video processing), it becomes more and more usual to apply different feature extraction techniques to the same signal, resulting in multiple feature streams

For instance, in *audio-visual speech recognition*, the audio signal is typically complemented by the video recording of the face (and thus the lips) of the person. It has already been shown (Dupont & Luettin, 2000; Bengio, 2004) that if the resulting audio and visual feature streams are properly modeled, such a multi-channel approach will significantly help in recognizing the speech utterances under noise conditions. Similar settings have also been used successfully for *audio-visual person authentication* (Bengio, 2004). In fact, even using only one raw source of information can yield better results in a multi-channel setting, e.g., using multiple sampling rates (*multi-rate*) or feature extraction (*multi-stream*) techniques, as already demonstrated for the task of speech recognition (Morris et al., 2001).

The field of *multimedia analysis*, which includes analysis of news, sports, home videos, meetings, etc, is very rich and these events are often recorded with at least two streams of information (audio and video) and sometimes more (as for the meeting scenario described later in this paper), and may contain complex human human interactions (McCowan et al., 2005). These multimedia documents also give rise to other applications such as *multimodal tracking of objects/humans* (Gatica-Perez et al., 2003). Furthermore, as the quantity of such archived documents grows, it becomes important to develop *multimedia document retrieval* systems (Renals et al., 2000; Westerveld et al., 2003) to find relevant documents based not only on their textual content but also on their joint visual and audio content.

Finally, numerous multi-channel sequence processing processing also appear in the context *wearable computers* (Mann, 1997), aiming at assisting people in various everyday activities (e.g., life saving, security, health monitoring, mobile web services) by using small devices such as cameras, microphones (e.g., recording all what you see and all what you hear), and multiple extra sensors (e.g., recording diverse physiological signals), etc.

In all the above applications, multi-channel processing presents several challenges. As already mentioned earlier, we first have to develop new sequence recognition strategies accommodating multiple frame rates, asynchrony, correlation between stream, etc. One solution to this problem, referred to as "Asynchronous HMM" (AHMM) will be discussed in the paper (Section 5). Furthermore, multi-channel processing may also impact differently the different levels of information that we aim at extracting from the observation streams. While AHMM can be well suited to classify sequential patterns into "low level" classes, they may not be appropriate, or easily tractable (because of training data and complexity issues), when one aims at extracting higher level information, such as semantic classes. In this case, it may be necessary to use a "hierarchical HMM" approach, where each "HMM layer" will use different types of multiple observation streams (possibly resulting of the previous HMM layer). This layered approach will be discussed in Section 6.

3. Notation and Models

Several models have already been proposed in the literature to handle multi-channel applications. We briefly discuss here some of the most successful approaches, using a unified notation. Let us denote an observation sequence $\mathbf{O}$ of T feature vectors as

$$\mathbf{O} = (\mathbf{o}_1, \mathbf{o}_2, \ldots, \mathbf{o}_T), \tag{1}$$

where $\mathbf{o}_t$ is the vector of all multimodal features available at time t. In general, such a set of features can be broken down into multiple streams (associated with channels, modalities, or different pre-processing) m. We thus further define the feature vector

$$\mathbf{o}_t^m \in \mathbb{R}^{N_m}, \tag{2}$$

where N_m is the number of features for stream m, with $1 \leq m \leq M$ (the total number of observation streams). Each observation sequence is typically associated with a corresponding sequence of high level classes or "events". For instance, in speech or handwriting recognition, this would correspond to a sequence of words. The most successful types of model used to handle observation sequences are all based on a statistical framework. In this context, the general idea is to estimate, for each type of high level event $\mathbf{v}_j \in V$, the parameters θ_j of a distribution over corresponding observation sequences $p(\mathbf{O}|\theta_j)$, where $\mathbf{O}$ would correspond to the event $\mathbf{v}_j$. The most well-known solution to efficiently model such distributions is to use Hidden Markov Models (HMMs).

HMMs have been used with success for numerous sequence recognition tasks, including speech recognition (Rabiner & Juang, 1993), video segmentation (Boreczky & Wilcox, 1998), sports event recognition (Xie et al., 2002), and broadcast news segmentation (Eickeler & Müller, 1999). HMMs introduce a state variable q_t and factor the joint distribution

of the observation sequence and the underlying (unobserved) HMM state sequence into two simpler distributions, namely emission distributions $p(\mathbf{o}_t|q_t)$ and transition distributions $p(q_t|q_{t-1})$. Such factorization assumes an underlying piece-wise stationary process (each stationary segment being associated with a specific HMM state), and yields efficient training algorithms such as the Expectation-Maximization (EM) algorithm (Dempster et al., 1977) which can be used to select the set of parameters θ_j^* of the model corresponding to event $\mathbf{v}_j$ in order to maximize the likelihood of L observation sequences:

$$\theta_j^* = \arg\max_{\theta_j} \prod_{l=1}^{L} p(\mathbf{O}_l|\theta_j). \tag{3}$$

The success of HMMs applied to sequences of events is based on a careful design of sub-models (topologies and distributions) corresponding to lexical units (phonemes, words, letters, events), and possibly semantic units (like the meeting group actions discussed in Section 6.1). Given a training set of observation sequences for which we know the corresponding labeling in terms of high level events (but not necessarily the precise alignment), we create a new HMM for each sequence as the concatenation of sub-model HMMs corresponding to the sequence of high level events. This HMM can then be trained using EM, thus adapting each sub-model HMM accordingly.

During testing, when observing a new observation sequence, the objective is simply to find the optimal sequence of sub-model HMMs (representing high level events) that could have generated the given observation sequence. Multiple algorithms have been developed to efficiently solve this problem, even in large search spaces, including stack decoders (Jelinek, 1969), or different approximations based on the well-known Viterbi algorithm (Viterbi, 1967).

While HMMs can be used to model various kinds of observation sequences, several extensions have been proposed to handle simultaneously multiple streams of observations, all corresponding to the same sequence of events (Morris et al., 2001; Dupont & Luettin, 2000; Oliver et al., 2002). The first and simplest solution is to *merge* all observations related to all streams into a single stream (frame by frame), and to model it using a single HMM as explained above. This solution is often called *early integration*. Note that in some cases, when the streams represent information collected at different frame rates (such as audio and video streams for instance), up-sampling or down-sampling of the streams is first necessary in order to align the streams to a common frame rate.

A better solution may be to use the *multi-stream* approach (Bourlard & Dupont, 1997). In this case, each stream is modeled separately using its own HMM. For instance, if we consider the modalities as separate streams, we would create one model $\theta_{m,j}^*$ for each event $\mathbf{v}_j$ and stream m such that

$$\theta_{m,j}^* = \arg\max_{\theta_{m,j}} \prod_{l=1}^{L} p(\mathbf{O}_l^m|\theta_{m,j}), \tag{4}$$

where $\mathbf{O}_l^m$ is the l^{th} observation sequence of stream m. When a new sequence of events needs to be analyzed, a special HMM is then created, recombining all the single stream HMM likelihoods at various specific temporal ("anchor") points automatically determined during training and decoding. Depending on these recombination points, various solutions appear. When the models are recombined after each state, the underlying system is equivalent to making the hypothesis that all streams are state-synchronous and independent of each other given a specific HMM state. This solution can be implemented efficiently and has shown robustness to various stream-dependent noises. The emission probability of the combined observations of M streams in a given state of the model corresponding to event $\mathbf{v}_j$ at time t is estimated as:

$$p(\mathbf{o}_t|q_t) = \prod_{m=1}^{M} p(\mathbf{o}_t^m|q_t, \theta_{m,j}). \tag{5}$$

One can see this solution as searching the best path into an HMM where each state i would be a combination of all states i of the single stream HMMs[1]. A more powerful recombination strategy enables some form of asynchrony between the states of each stream: one could consider an HMM in which states would include all possible combinations of the single stream HMM states. Unfortunately, the total number of states of this model would be exponential in the number of streams, hence quickly intractable. An intermediate solution, which we call *composite HMM*, considers all combinations of states in the same event only (Potamianos et al., 2004). Hence, in this model, each event HMM j now contains all possible combinations of states of the corresponding event $\mathbf{v}_{m,j}$ of each stream HMM m. The total number of states remains exponential but is more tractable, when the number of states of each stream remains low as well as the number of streams. The underlying hypothesis of this intermediate solution is that all streams are now event-synchronous instead of state-synchronous.

[1] Note that this solution forces the topology of each single stream to be the same.

Several other approaches to combine multiple streams of information have been proposed in the literature, but generally suffer from an underlying training or decoding algorithm complexity which is exponential in the number of streams. For instance, *Coupled Hidden Markov Models* (CHMMs) (Brand, 1996) can model two concurrent streams (such as one audio and one video stream) with two concurrent HMMs where the transition probability distribution of the state variable of each stream depends also on the value of the state variable of the other stream at the previous time step. More formally, let q and r be respectively the state variables of both streams, then CHMMs model transitions according to $p(q_t{=}i|q_{t-1}{=}j, r_{t-1}{=}k)$ and $p(r_t{=}i|r_{t-1}{=}j, q_{t-1}{=}k)$. While the exact training algorithm for such a model quickly becomes intractable when extended to more than 2 streams, an approximate algorithm which relaxes the requirement to visit every transition (termed the N-heads algorithm) was proposed in (Brand, 1996), and can be tractable for a small number of streams.

Two additional approaches have been proposed recently, and will be the focus of Sections 5 and 6. These are the *Asynchronous HMM* (Bengio, 2003), that can handle asynchrony between streams, and the *Layered HMM* (Zhang et al., 2004; Bourlard et al., 2004) than can help in constraining the model according to levels of prior knowledge.

4. Challenges

While there are already several models proposed in the literature to cope with multi channel sequence processing, we believe that there are still several research challenges that have not been adequately addressed yet, including:

1. **How to handle more than two streams?** Most solutions that model the joint probability of the streams need in general exponential resources with respect to the number of streams, the number of states of each underlying Markov chain, or the size of each stream. This practically means that handling more than two streams is already a challenge. One possible alternative is to limit the search space through the use of reasonable heuristics, which should depend on *a priori* knowledge on the interdependencies of the streams.

2. **How to handle learning in high dimensional spaces?** The observation space (the total number of observed features per time step) grows naturally with the number of streams. Furthermore, it is often the case that the total number of parameters of the model grows linearly or more with the number of observations (for instance if the conditional observation distributions are modeled with Gaussian Mixture Models). Hence, one has to fight the well-known *curse of dimensionality* (Bishop, 1995).

3. **How to handle long term temporal dependencies?** This problem deals with sequential data where one needs to relate information observed at time t with information observed at time $t + k$ where k is rather large. It has been shown (Bengio et al., 1994) that this becomes exponentially difficult with k when no structural knowledge is built *a priori* in the model. Hence, in order for multi channel processing to be successful, an appropriate structure is necessary.

4. **Joint feature extraction and heterogeneity of sources.** In current systems involving multiple streams of information, features used to represent each stream are extracted independently. On the other hand, if one agrees that there may be some correlation between the streams, one should therefore devise joint feature extraction techniques, which should then yield more robust performance. However, what should we then do with streams of different nature (such as the slides of a presentation, together with the video of the person performing the same presentation)?

5. **How to handle different levels of** *a priori* **knowledge constraints?** It has been known for decades that in order to obtain good speech recognition performance, one has to constrain the recognition model with a good language model, that only permits valid and probable sequences of words to be recognized. The same idea should thus be applied to other domains, such as videos, which contain rich high level information that should be constrained somehow. Several levels of description should thus be used in such language model; for instance, a visual scene could be described by the pixels of the image, the persons present in the image, the action taking place, the body language, etc. For each of these levels, a probabilistic model of what is possible and what is not should therefore be trained. Furthermore, one should devise **multi channel language models** in order to take into account information coming from several streams at the same time.

6. **Asynchrony between streams.** Let us consider the simplest multi-channel case, with 2 streams, and let us assume that these 2 streams describe the same sequence of 3 "events" (classes)

A, B and C. Furthermore, let us assume, as illustrated in Figure 1, that the best piecewise stationary alignment of each stream to the sequence A-B-C would not coincide temporally with each other (which we refer to "stream asynchrony"). In such a case (which is discussed in more details in Section 5), a naive solution to try to model the joint probability of the two streams (e.g., applying early integration) would need an exponential number of states (with respect to the number of streams), as depicted in the third line of Figure 1. A better solution, depicted in the fourth line of Figure 1, would *stretch* or *compress* the streams along a single HMM model with the goal to *realign* them during training and decoding. Such a model is described in Section 5.

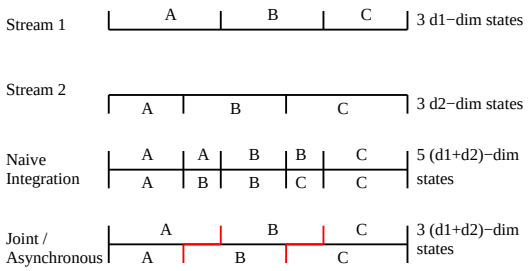


Figure 1. Complexity issue with asynchronous streams.

7. **Available benchmark datasets for evaluation**. One of the reasons of the steady progress of speech recognition has been the ever increasing availability of larger and larger realistic labeled datasets, and the yearly organization of international competitions. It is well known that this is a key point for progress in any scientific research field. However, to date, very little material has been recorded and properly annotated for multi channel sequence processing. Audio-visual speech recognition and person authentication are probably the fields where most available databases can be found. What about other scenarios, such as multimedia analysis, multimodal surveillance, etc? In Section 6, we describe a first initiative of such a benchmark database available for the meeting scenario.

5. Handling Asynchrony

Properly modeling asynchrony and correlation between multiple observation streams is thus a challenging problem. However, as a matter of fact, there are multiple evidences of real life applications involving several asynchronous streams. For instance, audio-visual speech recognition usually exhibits asynchrony. Indeed, the lips of a person often start moving earlier than any sound is uttered, mainly because the person is preparing to utter the sound. Another example is the *speaking and pointing* scenario, where a person complement the speech signal with a pointing gesture (to a point of interest). In this case, of course, although the two streams are related to the same high-level event, the pointing event will usually never occur exactly at the same time as the vocal event. One last example of asynchrony: in a news video, there is almost always a variable delay between the moment when the newscaster says the name of a public personality and the moment when the personality's picture actually appears on the screen.

One can think of several other instances involving asynchrony between streams, and there is thus a need to model this phenomenon in a principled way. As described below, such a solution, referred to as *Asynchronous HMM* was recently proposed.

5.1. The Asynchronous HMM

Let us consider the case where one is interested in modeling the joint probability of two asynchronous streams, denoted here $\mathbf{O}^1$ of length T_1 and $\mathbf{O}^2$ of length T_2 with $T_2 \leq T_1$ without loss of generality[2]. We are thus interested in modeling $p(\mathbf{O}^1, \mathbf{O}^2)$. Following the ideas introduced for HMMs, we represent this distribution using a hidden variable Q which represents the (discrete) *state* of the generating system, which in our case is synchronized with the longest sequence $\mathbf{O}^1$.

Moreover, since we know that $\mathbf{O}^2$ is smaller than $\mathbf{O}^1$, let the system always emit $\mathbf{o}_t^1$ at time t but only sometimes emit $\mathbf{o}_s^2$ at time t, with $s \leq t$. Let us define $\tau_t{=}s$ as the fact that o_t^1 is emitted at the same time as o_s^2; τ can thus be seen as the alignment between $\mathbf{O}^1$ and $\mathbf{O}^2$. Hence, an Asynchronous HMM (AHMM) (Bengio, 2003) models $p(\mathbf{O}^1, \mathbf{O}^2, Q, \tau)$.

Using these hidden variables, and using several reasonable independence assumptions, we can factor the joint likelihood of the data and the hidden variables into several simple conditional distributions:

- $P(q_t{=}i|q_{t-1}{=}j)$, the probability to go from state j to state i at time t,
- $p(\mathbf{o}_t^1, \mathbf{o}_s^2|q_t{=}i)$, the joint emission distribution of $\mathbf{o}_t^1$ and $\mathbf{o}_s^2$, while in state i at time t,

[2]Since all the reasoning below can easily be generalized to sequences (even of the same length) where the warping (stretching and compressing) can occur at different instances in the different streams.

- $p(\mathbf{o}_t^1|q_t{=}i)$, the emission distribution of $\mathbf{o}_t^1$ only, while in state i at time t,
- $P(\tau_t{=}s|\tau_{t-1}{=}s-1, q_t{=}i, \mathbf{o}_{1:t}^1, \mathbf{o}_{1:s}^2)$, the probability to emit on both sequences while in state i at time t.

We showed in (Bengio, 2003) that using these simple distributions, new algorithms could be developed to (1) estimate the *joint likelihood* of the two streams, (2) *train a model* to maximize the joint likelihood of pairs of streams, and (3) jointly estimate the *best sequence of states* Q and the best alignment between pairs of streams.

Furthermore, one can still constrain the model to consider only reasonable alignments, e.g., integrating some minimum and maximum asynchrony between the streams. Using this constraint and denoting N_q the number of states of the model, the training and decoding complexity become $\mathcal{O}(N_q^2 \cdot T_1 \cdot k)$, which is only k times the usual HMM complexity.

5.2. Audio-Visual Speech Recognition

The proposed AHMM model was applied to several tasks, including audio-visual speech recognition and speaker verification (Bengio, 2004), as well multi-channel meeting analysis (Zhang et al., 2004). We report here results on the M2VTS database (Pigeon & Vandendorpe, 1997) for the task of audio-visual speech recognition, where the speech features where standard Mel-Frequency Cepstral Coefficients (MFCCs), while the visual features where shapes and intensities around the mouth region, obtained by lip tracking. In order to evaluate the robustness of audio-visual speech recognition, various levels of noise were injected into the audio stream during decoding, while training was always done using clean audio only. The noise was taken from the Noisex[3] database (Varga et al., 1992), and added to the speech signal injected to reach segmental signal-to-noise ratios (SNR) of 10dB, 5dB and 0dB.

Asynchronous HMMs were compared to classical HMMs using only the audio stream, only the video stream, or both streams combined using the *early integration* scheme. Figure 2 presents the results in terms of *Word Error Rate* (WER), a commonly used measure in the field of speech recognition, which takes into account the number of insertions, deletions and substitutions[4]. As observed from Figure 2, the AHMM consistently yielded lower WER as soon as the noise

[3]We took the *stationary speech noise*.

[4]Basically, the edit (Levenshtein) distance between the recognized and reference word sequences.

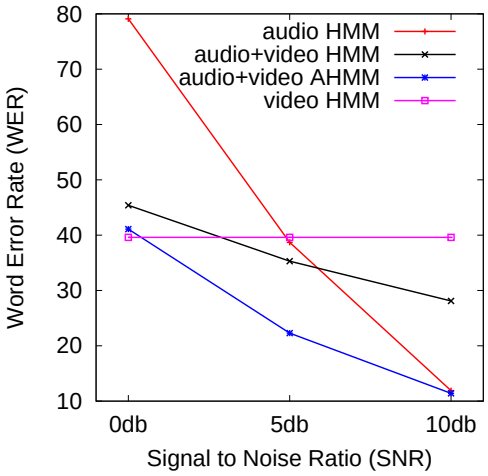


Figure 2. Word Error Rates (in percent, the lower the better), of various systems under various noise conditions.

level was significant. Actually, it did not yield significantly lower performance (using a 95% confidence interval) than the video stream alone in case of very low (0dB) SNR, while performing as well as the audio stream alone in case of "clean" speech (10dB).

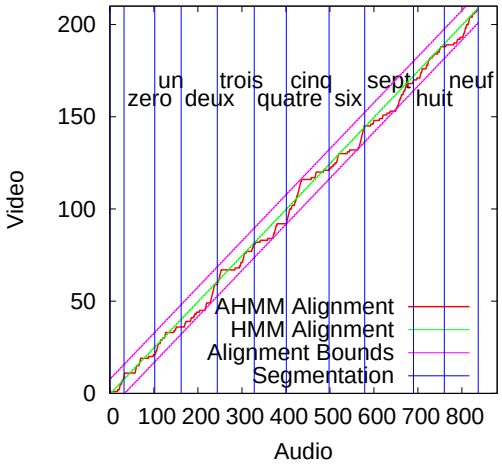


Figure 3. Alignment obtained by the model between video and audio streams on a typical sequence corrupted with a 10dB Noisex noise. The vertical lines show the obtained segmentation between the words. The alignment bounds represent the maximum allowed stretch between the audio and the video streams.

An interesting side effect of the model is to provide the "optimal" alignment between the audio and the video streams, as a by-product of the decoding process. This is illustrated in Figure 3 showing the audio-visual stream alignment resulting from the AHMM decoding

of a specific digit sequence corrupted with 10dB Noisex noise. As it can be seen, the alignment is far from being linear. This shows that computing and maximizing the joint stream probability using AHMM appears more informative than using a naive alignment and a normal HMM.

6. A Layered Approach

6.1. The Meeting Scenario

Automatic analysis of meetings (including, e.g., automatic modeling of human interaction in meetings by modeling the joint behavior of participants through multiple audio and visual features) is a particularly challenging application of multi-channel sequence processing. It is multimodal by nature (meetings can be recorded with several cameras and microphones, as well as with other devices capturing information coming from the white-board, the slide projector, etc) and is also a rich case study of human interaction.

In (McCowan et al., 2005), a principled approach to the automatic analysis of meetings was proposed, defining meetings as continuous sequences of *group actions* chosen from a predefined dictionary of actions (including, for instance, monologue, discussion, white-board presentation, with or without note-taking, agreement/disagreement, etc). This made the problem well suited for supervised learning approaches. The group actions should be mutually exclusive, exhaustive, and as much as possible unambiguous to human observers. To this end, we have collected a corpus of 60 short meetings of about 5 minutes each (30 for training, and 30 for test purposes) in a room equipped with synchronized multi-channel audio and video recorders. The resulting corpus, including annotation, is now publicly available at `http://mmm.idiap.ch`[5]. Each meeting consisted of four participants seated at a table in a typical workplace setting. Three cameras captured the participants, the projector screen and white-board. Audio was recorded using one lapel microphone per participant and an eight-microphone array located in the center of the table. The overall goal was to minimize the *Action Error Rate* (AER), similarly to what is done in speech recognition with Word Error Rate (WER), but over sequences of high level group actions. To this end, several extensions of HMMs, including AHMMs, were tested and results are reported in (McCowan et al., 2005).

More recently, we proposed a multi-layered solution (Zhang et al., 2004; Bourlard et al., 2004) intended at simplifying the complexity of the task, based on an approach presented in (Oliver et al., 2002).

[5]In the framework of the AMI European Integrated Project (`http://www.amiproject.org`) this corpus is now extended to about 100 hours of multimodal meeting data.

6.2. A Two-Layer Approach

Let us define two sets of actions, whether they are specific to individual participants or to the group. While the overall goal is at the level of group actions, we believe that individual actions could act as a bridge between high level complex group actions and low level features, thus decomposing the problem into stages, or layers.

To this end, we defined the group action vocabulary set with the following 8 actions: *discussion*, *monologue*, *monologue+note-taking*, *note-taking*, *presentation*, *presentation+note-taking*, *white-board*, *white-board+note-taking*. Furthermore, we defined the individual action vocabulary with the following 3 actions: *speaking*, *writing*, *idle*.

Obviously, individual actions should be easier to annotate in the corpus (as being less ambiguous) and should also be easier to learn with some training data, as they are obviously more related to low level features that can be extracted from the raw multiple channels. Furthermore, knowing the sequence of individual actions of each participant, one should easily be able to infer the underlying sequence of group actions. Thus considering every meeting participant as a "multi-stream generator", each of the participant's streams should be processed by a first layer of HMMs, and the resulting HMM's outputs (likelihoods/posteriors) will then be combined by a second HMM layer yielding, higher level, group actions.

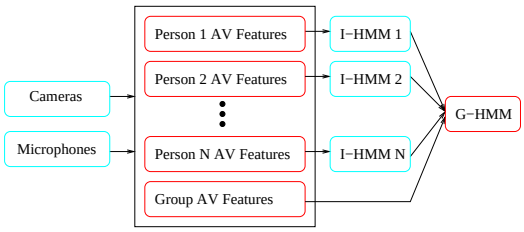


Figure 4. A two-layer approach

Figure 4 illustrates the overall strategy. Audio-visual features are first extracted for each of the meeting participants (Zhang et al., 2004), complemented by more general *group-level* features. An *individual HMM* (I-HMM) is then trained for each participant, using the individual action vocabulary. To have these I-HMMs as much "participant independent" as possible, all parameters are shared among all models, yielding up to

Table 1. Action error rates (AER) for various systems applied to the meeting scenario.

Method		AER (%)
Single-layer	Visual only	48.20
	Audio only	36.70
	Early Integration	23.74
	Multi-Stream	23.13
	Asynchronous	22.20
Two-layer	Visual only	42.45
	Audio only	32.37
	Early Integration	16.55
	Multi-Stream	15.83
	Asynchronous	15.11

4 times more data to train the I-HMMs. Several models were compared, including early integration, multi-stream, and asynchronous HMMs (AHMM).

We then estimate for each participant i the posterior probability of each individual task $\mathbf{v}_{i,j}$ at each time step t given the individual observation sequence up to time t, $p(\mathbf{v}_{i,j}|\mathbf{o}^i_{1:t})$. These posterior probabilities, together with *group-level features*, are then used as observations for the second layer, the *group* HMM, (G-HMM), which are trained on the group action vocabulary. Again, this G-HMM was implemented in various flavors, including early integration, multi-stream and asynchronous HMMs. Section 6.3 below further discusses this aspect and shows how these (lower level) posterior probabilities can be estimated to guarantee some form of "optimality", while preserving maximum information (i.e., avoiding local decisions) across the different layers.

Table 1 reports the AER performance achieved by the different systems. It can be seen that (1) the two-layer approach always outperforms the single-layer one, and (2) the best I-HMM model is the Asynchronous HMM, which probably means that some asynchrony exists in this task, and is actually well captured by the model.

6.3. General Multi-Layered (Hierarchical) HMM Approach

As illustrated from the above meeting scenario, the complexity resulting from the processing of multiple channels of information, in order to extract low-level as well as high-level information (such as the analysis of multimodal meetings in terms of high level meeting actions), is often such that it will often be necessary to *break down* the problem in terms of multiple layers of sub-problems, probably using different constraints and prior knowledge information sources. The layered approach is one possible and principled solution to achieve this. Given a complex task, the goal is then to break it down into several hierarchically embedded sub-tasks, for which one can devise proper models (from enough training data), and use adequate (level specific) constraints.

We recently proposed such an approach for the task of speech recognition (Bourlard et al., 2004), where a general theoretical framework was proposed to compute low-level (e.g., phoneme) class posteriors, based on all the acoustic context, and to hierarchically combine those posteriors to yield higher-level (e.g., sentence) posteriors. In this approach, each layer is integrating its own prior constraints.

More precisely, a first layer, which could be an HMM or an AHMM, as in the meeting scenario, or any other model such as an Artificial Neural Network (ANN), is used to estimate posterior probabilities $p(q_t = i|\mathbf{O})$ of sub-classes i (such as phonemes, for the case of speech recognition) at each time step t given all the available information (for instance, all the acoustic sequence $\mathbf{O}$). In HMM, as well as in hybrid HMM/ANN systems, this posterior probability estimate is given by the so-called $\gamma(i,t) = p(q_t = i|\mathbf{O})$, which can be obtained by running and combining the so-called α and β recurrences through the appropriate HMM. Ideally, this HMM should embed all known lexical constraints about legal and probable sequences of phonemes. One should then use the resulting posterior probabilities (of every sub-class at every time step) as input to the next layer model, which would then estimate the posterior probabilities (again through new γ's) of higher level classes, such as words, constraining the underlying HMM model with all known language constraints that pertains to legal and probable sequences of words. In theory, this operation could be repeated up to the the level of sentences, and even to the level of summarization, always using posterior probabilities resulting from the previous layer as intermediate features.

Initial results on several speech tasks, as well as on the meeting task discussed previously, resulted in significant improvements. In (Bourlard et al., 2004), speech recognition results where presented on Numbers'95 (speaker independent recognition of free format numbers spoken over the telephone) and on a reduced vocabulary version (1,000 words) of the DARPA Conversational Telephone Speech-to-text (CTS) task, and both resulted in significant improvements.

7. Conclusion

This paper discussed several issues arising from the processing of complex multi-channel data, including large multimodal problems (meeting data). More specifically, this paper focused on two important issues, namely stream asynchrony and complexity of high-level decision processes. The proposed Asynchronous HMMs (AHMM) actually maximize the likelihood of the joint observation sequences through a single HMM, while also automatically allowing for stretching and/or compressing of the different streams. However, in the case of very complex problems, using AHMMs is often not enough, and the problem needs to be broken down into simpler processing blocks. A solution to this problem, referred to as "multi-layered/hierarchical HMMs" (and where each layer can integrate different levels of constraints and prior information) was also proposed and shown to be effective in modeling the joint behavior of participants in multimodal meetings. A full theoretical motivation of this approach is described in (Bourlard et al., 2004).

Acknowledgements

This work was partly funded (through OFES) by the European PASCAL Network of Excellence and the AMI Integrated Project. The results reported here also benefited from financial support from the Swiss National Science Foundation, through the National Center of Competence in Research IM2 (Interactive Multimodal Information Management).

References

Bengio, S. (2003). An asynchronous hidden markov model for audio-visual speech recognition. *Advances in Neural Information Processing Systems 15*.

Bengio, S. (2004). Multimodal speech processing using asynchronous hidden markov models. *Information Fusion*, *5*, 81–89.

Bengio, Y., Simard, P., & Frasconi, P. (1994). Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*, *5*, 157–166.

Bishop, C. (1995). *Neural networks for pattern recognition*. London, UK: Oxford University Press.

Boreczky, J. S., & Wilcox, L. D. (1998). A Hidden Markov Model framework for video segmentation using audio and image features. *Proc. of ICASSP*.

Bourlard, H., Bengio, S., Doss, M. M., Zhu, Q., Mesot, B., & Morgan, N. (2004). Towards using hierarchical posteriors for flexible automatic speech recognition systems. *Proc. of DARPA EARS Rich Transcription Workshop*.

Bourlard, H., & Dupont, S. (1997). Subband-based speech recognition. *Proc. IEEE ICASSP*.

Brand, M. (1996). *Coupled hidden markov models for modeling interacting processes* (Technical Report 405). MIT Media Lab Vision and Modeling.

Dempster, A. P., Laird, N. M., & Rubin, D. B. (1977). Maximum-likelihood from incomplete data via the EM algorithm. *Journal of Royal Statistical Society B*, *39*, 1–38.

Dupont, S., & Luettin, J. (2000). Audio-visual speech modeling for continuous speech recognition. *IEEE Transactions on Multimedia*, *2*, 141–151.

Eickeler, S., & Müller, S. (1999). Content-based video indexing of TV broadcast news using Hidden Markov Models. *Proc. of ICASSP*.

Gatica-Perez, D., Lathoud, G., McCowan, I., & Odobez, J.-M. (2003). A mixed-state i-particle filter for multi-camera speaker tracking. *Proc. of WOMTEC*.

Jelinek, F. (1969). A fast sequential decoding algorithm using a stack. *IBM Journal of Research and Development*, *13*, 675–685.

Mann, S. (1997). Smart clothing: The wearable computer and wearcam. *Personal Technologies*. Volume 1, Issue 1.

McCowan, I., Gatica-Perez, D., Bengio, S., Lathoud, G., Barnard, M., & Zhang, D. (2005). Automatic analysis of multimodal group actions in meetings. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *27*, 305–317.

Morris, A., Hagen, A., Glotin, H., & Bourlard, H. (2001). Multi-stream adaptive evidence combination for noise robust ASR. *Speech Communication*.

Oliver, N., Horvitz, E., & Garg, A. (2002). Layered representations for learning and inferring office activity from multiple sensory channels. *Proc. of the Int. Conf. on Multimodal Interfaces*.

Pigeon, S., & Vandendorpe, L. (1997). The M2VTS multimodal face database (release 1.00). *Proc. of the Conf. on AVBPA*.

Potamianos, G., Neti, C., Luettin, J., & Matthews, I. (2004). Audio-visual automatic speech recognition: An overview. In G. Bailly, E. Vatikiotis-Bateson and

P. Perrier (Eds.), *Issues in visual and audio-visual speech processing*. MIT Press.

Rabiner, L. R., & Juang, B.-H. (1993). *Fundamentals of speech recognition*. Prentice-Hall.

Renals, S., Abberley, D., Kirby, D., & Robinson, T. (2000). Indexing and retrieval of broadcast news. *Speech Communication*, *32*, 5–20.

Varga, A., Steeneken, H., Tomlinson, M., & Jones, D. (1992). *The noisex-92 study on the effect of additive noise on automatic speech recognition* (Technical Report). DRA Speech Research Unit.

Viterbi, A. (1967). Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE Transactions on Information Theory*, 260–269.

Westerveld, T., de Vries, A. P., van Ballegooij, A., de Jong, F., & Hiemstra, D. (2003). A probabilistic multimedia retrieval model and its evaluation. *EURASIP Journal on Applied Signal Processing*, *2*.

Xie, L., Chang, S.-F., Divakaran, A., & Sun, H. (2002). Structure analysis of soccer video with Hidden Markov Models. *ICASSP*.

Zhang, D., Gatica-Perez, D., Bengio, S., McCowan, I., & Lathoud, G. (2004). Modeling individual and group actions in meetings: a two-layer hmm framework. *IEEE Workshop on Event Mining at CVPR*.

Probabilistic Logic Learning and Reasoning

Kristian Kersting KERSTING@INFORMATIK.UNI-FREIBURG.DE
University of Freiburg, Institute for Computer Science, Machine Learning Lab, Georges-Koehler-Alle 079, 79110 Freiburg, Germany

Probabilistic logic learning which is sometimes also called statistical relational learning addresses one of the central questions of artificial intelligence: the integration of probabilistic reasoning with first order logic representations and machine Learning. In the past few years, this question has received a lot of attention. Various different approaches have been developed in several related, but different areas (including machine learning, statistics, inductive logic programming, databases, and reasoning under uncertainty). Most researchers only have exposure to one or two of the constituents underlying Probabilistic Logic Learning, cf. Figure 1.

In this talk, I shall start from a Bayesian network perspective and sketch how to incorporate logical concepts of objects and relations among these objects. As time and actions are not just other relations, I shall afterwards outline probabilistic logic reasoning over time and making complex decision in relational domains.

More specifically, I shall overview how to upgrade *Bayesian networks* to *Bayesian Logic Programs* (Kersting & De Raedt, 2001), *hidden Markov models* to *logical hidden Markov models* (Kersting et al., 2003); and *Markov decision processes* to *Markov decision programs* (Kersting et al., 2004). I shall also show that probabilistic logic learning approaches naturally yield kernels for structured data (Kersting & Gärtner, 2004). The resulting approaches will be illustrated using examples from genetics, bio-informatics and classical planning domains.

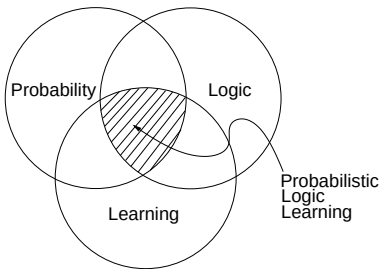


Figure 1. Probabilistic Logic Learning as the intersection of *Probability*, *Logic*, and *Learning*.

This is joint work with Luc De Raedt, Thomas Gaertner, Tapani Raiko, Martijn Van Otterlo and is part of the EU projects APRIL I+II (Application of Probabilistic Inductive Logic Programming I+II), `http://www.aprill.org`.

The talk will also partly be based on (De Raedt & Kersting, 2003; De Raedt & Kersting, 2004).

References

De Raedt, L., & Kersting, K. (2003). Probabilistic Logic Learning. *SIGKDD Explorations: Special issue on Multi-Relational Data Mining*, *5*, 31–48.

De Raedt, L., & Kersting, K. (2004). Probabilistic Inductive Logic Programming. *Proceedings of the 15th International Conference on Algorithmic Learning Theory (ALT-04)* (pp. 19–36). Padova, Italy.

Kersting, K., & De Raedt, L. (2001). Towards Combining Inductive Logic Programming and Bayesian Networks. *Proceedings of the Eleventh Conference on Inductive Logic Programming (ILP-01)* (pp. 118 – 131). Strasbourg, France.

Kersting, K., & Gärtner, T. (2004). Fisher kernels for logical sequences. *Proceedings of the 15th European Conference on Machine Learning (ECML-2004)* (pp. 205 – 216). Pisa, Italy.

Kersting, K., Raiko, T., Kramer, S., & De Raedt, L. (2003). Towards discovering structural signatures of protein folds based on logical hidden markov models. *Proceedings of the Pacific Symposium on Biocomputing (PSB-03)* (pp. 192 – 203). Hawaii, USA.

Kersting, K., Van Otterlo, M., & De Raedt, L. (2004). Bellman goes Relational. *Proceedings of the Twenty-First International Conference on Machine Learning (ICML-04)* (pp. 465 – 472). Banff, Canada.

Monotone Constraints in Frequent Tree Mining

Jeroen De Knijf[†] JKNIJF@CS.UU.NL
Ad Feelders AD@CS.UU.NL
Utrecht University, Institute of Information and Computing Sciences, PO Box 80.089, 3508 TB Utrecht.

Abstract

Recent studies show that using constraints that can be pushed into the mining process, substantially improves the performance of frequent pattern mining algorithms. However the constraints and algorithms have not yet been explored for frequent structure mining. In this paper we present monotone constraints for trees and develop an opportunistic pruning algorithm that mines frequent trees with monotone constraints. We illustrate the use of these constraints within the application of web log mining. Finally, the effect of applying these constraints on synthetic data sets is evaluated. The opportunistic pruning methods leads to a considerable speedup and a reduction in the number of candidates generated compared to the basic algorithm.

1. Introduction

In frequent itemset mining the use of constraints is well explored(Ng et al., 1998; Bayardo, 1998; Bayardo et al., 1999; Pei & Han, 2000). Constraints that can be pushed into the mining process — as opposed to post pruning — allow the mining algorithm to reduce the search space and hence result in a substantial efficiency gain. Furthermore, constraints provide the end user with a tool to focus his interest such that many uninteresting rules don't have to be examined. Since the number of rules can be exponential in the size of the input data, these tools are necessary in practical data mining applications.

When structured data is taken into account several algorithms (Asai et al., 2002; Zaki, 2002; Nijssen & Kok, 2003) are available — based upon the same principle as the Apriori (Agrawal & Srikant, 1994) — that mine for frequent structures in the database. A structure may be a sequence, tree or graph, but in this paper we limit the discussion to trees. In general, structured data can be seen as an extension of unstructured data. Structured data arise extensively in domains such as computational biology, XML databases, and molecule databases. Since the structure of an item encodes an important part of its semantics, mining for structures is an important extension of frequent itemset mining.

Because of the advantages of constraints in frequent itemset mining, we extend and explore the scope of constraints to trees. For monotone constraints an extension of (a slightly modified version of) FREQT (Asai et al., 2002) is described that computes exactly the frequent trees that satisfy these constraints. Besides the optimisation based upon the Apriori property, our algorithm also performs opportunistic pruning on the monotone constraints. The pruning strategy does not prune all trees that falsify the constraints, but all pruned trees do falsify the constraints. In this sense the pruning is opportunistic.

The web mining problem serves as an example for mining frequent trees. In the rest of this paper we will use this example to clarify constraints we introduce.

This paper is organised as follows. In section 2 we give the basic concepts of tree mining and of FREQT in particular. In section 3 constraints for trees are explored and a pruning strategy for monotone constraints is described. We performed a number of experiments on synthetic data sets, to evaluate the monotone tree mining algorithm. The results of these experiments are reported in section 4. In the last section we give conclusions and further research directions.

2. Background

In this section we provide the basic concepts and notation that is used in the remainder of this paper.

2.1. Basic Concepts

A labeled ordered tree $T = \{V, E, \Sigma, L, v_0, \leq\}$ consists of a vertex set V, an edge set E, an alphabet Σ for vertex labels, a labeling function $L : V \rightarrow \Sigma$ that assigns labels to vertices and a binary relation $\leq \subseteq V^2$

[†]supported by the Netherlands Organisation for Scientific Research (NWO) under grant no. 612.066.304

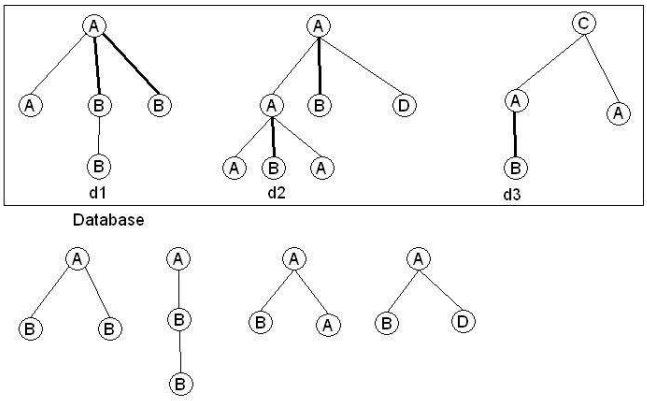


Figure 1. Example database with frequent pattern $A - B$ marked with bold lines (upper half). Below the candidate patterns of size three extended from the frequent $A - B$ pattern.

that represents a sibling relation among the children of a node. The special node v_0 is called the root. If $(u, v) \in E$ then u is the parent of v and v is the child of u. For a node v, any node u on the path from the root node to v is called an ancestor of v. If u is an ancestor of v then v is called a descendent of u.

Given two labeled rooted ordered trees T_1 and T_2 we call T_2 an induced subtree of T_1, denoted as $T_2 \preceq T_1$ if there exists a injective matching function Φ of T_2 into T_1, that satisfies the following conditions for any $v, v_1, v_2 \in V_{T_2}$:

1. Φ preserves the parent relation : $(v_1, v_2) \in E_{T_2}$ if $(\Phi(v_1), \Phi(v_2)) \in E_{T_1}$.
2. Φ preserves the sibling relation :if $v_1 \leq_{T_2} v_2$ then $\Phi(v_1) \leq_{T_1} \Phi(v_2)$.
3. Φ preserves the labels : $L_{T_2}(v) = L_{T_1}(\Phi(v))$.

Let D denote a database where each transaction $d \in D$ is a labeled rooted ordered tree. For a given pattern tree t, which is also a labeled rooted ordered tree, we say t occurs in a transaction d if t is a subtree of d. Let $\phi_d(t)$ denote the number of distinct occurrences of t in d. Let $\psi_d(t) = 1$ if $\phi_d(t) > 0$ and 0 otherwise. The support of a subtree t in the database D is then defined as $\sum_{d \in D} \psi_d(t)$. A pattern tree t is called frequent if the support of t is greater or equal then a user defined minimum support (minsup) value. The goal of frequent tree mining is to find all frequent trees in a given database. Frequent tree mining algorithms make use of the apriori property: any subtree of a frequent tree is also frequent and any supertree of an infrequent tree is also infrequent.

2.2. The FREQT Algorithm

With the background given in section 2.1 we are now able to describe the FREQT (Asai et al., 2002) algorithm. Our work is based upon this algorithm, but can be extended to any other frequent tree mining algorithm that uses the induced subtree relation. The algorithm described below is a slight modification of FREQT: where FREQT uses as a database one data tree, we use a set of trees. The support is defined in FREQT as $\phi_d(t)$ with d the data tree and t the pattern tree. This notion is also known as weighted support. The modification we made does not lead to conceptual differences in the algorithm.

The key notion of FREQT is that frequent subtrees of size $k-1$, where the size of a tree is defined as the number of nodes, are expanded by attaching a new node only to a node on the rightmost branch of the tree, to yield a larger tree of size k. The rightmost branch of a tree is the unique path from the root to the rightmost leaf. This procedure of extending pattern trees ensures that each pattern tree is counted exactly once. The algorithm in pseudo-code is given below:

$k \leftarrow 1$
$F_1 \leftarrow$ set of frequent trees of size 1
$RMO_1 \leftarrow$ rightmost occurrences of trees in F_1
While $F_k \neq \emptyset$

$k \leftarrow k+1$
$C_k, RMO_k \leftarrow$ compute the candidate trees
and their occurrences from (F_{k-1}, RMO_{k-1})
$F_k \leftarrow$ count C_k, RMO_k

Consider figure 1 which shows an example database with three data trees. The pattern tree t with nodes $A-B$ occurs twice in d_1 and d_2 and once in d_3, hence t has a support of 3. The candidate trees generated from t according to the rightmost extension technique are shown in the lower half of figure 1. From d_3 no candidate trees could be generated.

3. Mining Trees with Monotone Constraints

In this section we define monotone constraints for trees and describe an opportunistic pruning method to compute frequent trees.

3.1. Constraints for Trees

The different types of constraints defined for sets (Ng et al., 1998) can also be applied in structured data mining. The structure of the data can be ignored and the constraints can be applied as a post-processing step of the mining algorithm. This approach has not all benefits of constraint based mining, because all frequent trees have to be computed and afterward all trees that falsify the constraints are thrown away. Efficient algorithms that directly mine the trees that satisfy the constraints (beside the frequency constraint) are however not available yet. We now give a brief outline how the different classes of item set constraints can be pushed deeply into the mining process of structured data. For this we assume that each vertex has an attribute which holds a measure of interest. This corresponds with e.g. the price of items, or any other defined measure in frequent itemset mining. When we define an SQL based aggregate function over a tree t, such as $sum(t)$, $avg(t)$, $min(t)$, the function is applied on the attributes of all the vertexes of t.

Anti-monotone constraints: A constraint C defined over a tree t has the anti-monotonicity property if

$$C(t) = true \Rightarrow \forall t' \preceq t \ : C(t') = true$$

Examples of anti-monotone constraints are: minimum frequency, $sum(t) \leq constant$ (for positive values of the node attributes of t), $size(t) \leq constant$. The minimum frequency constraint is already incorporated in structured data mining algorithms, hence other constraints which have the anti-monotonicity property can be incorporated in the same way.

Succinct constraints: A constraint C is succinct if C is pre-counting prunable. Examples are: $min(t) \leq c$ and $max(t) \leq c$. Succinct constraints are computed in frequent itemset mining by first selecting the items that satisfy the constraints and then generating the candidates using these items. In frequent tree mining the same approach could be used, but by applying the rightmost expansion, or any other known method to enumerate all trees that satisfy the constraints, a lot of trees may be missed or enumerated twice. In order to incorporate these constraints, another enumeration technique would be needed.

Convertible constraints: A constraint C is convertible (anti)monotone if there is an order among the items, such that whenever the items are pushed in order the constraint becomes (anti)monotone. Consider as an example the constraint $avg(t) \geq 10$. This constraints becomes anti-monotone if the items are added in descending order. To incorporate this class of constraint in structured data mining is probably hard, because the order in which the attributes must be processed to become (anti)monotone will in general not coincide with the order imposed by the structure of the data.

Monotone constraints: A constraint C defined over a tree t is monotone if:

$$C(t) = true \Rightarrow \forall t' \succeq t \ : C(t') = true$$

Examples of monotone constraints are $sum(t) \geq c$ (for positive values of the node attributes of t) and $size(t) \geq c$. Monotone constraints can be mined with a top down algorithm, which starts with the complete set of items as in (Bayardo, 1998). The straightforward adoption of the top down approach does not work for frequent tree mining, because we don't have in general one tree which is a supertree of all frequent trees. Instead there is a set of supertrees, which is exactly the database.

To illustrate the use of monotone constraints for trees consider a weblog application. Suppose we have a database of web access logs at a popular site, where each record (transaction in frequent itemset terms) is the entire forward accesses of a visitor. Suppose we are interested in the most frequently accessed subtrees at that site. Vertices in the tree correspond to webpages, edges to visitor's action going from one webpage to another. To each webpage a value is assigned according to some interest measure of the end user. For example, we could associate with each vertex (webpage), the number of images contained by the page. Another constraint could be that the end user is only interested in frequent accessed subtrees of size bigger than some constant.

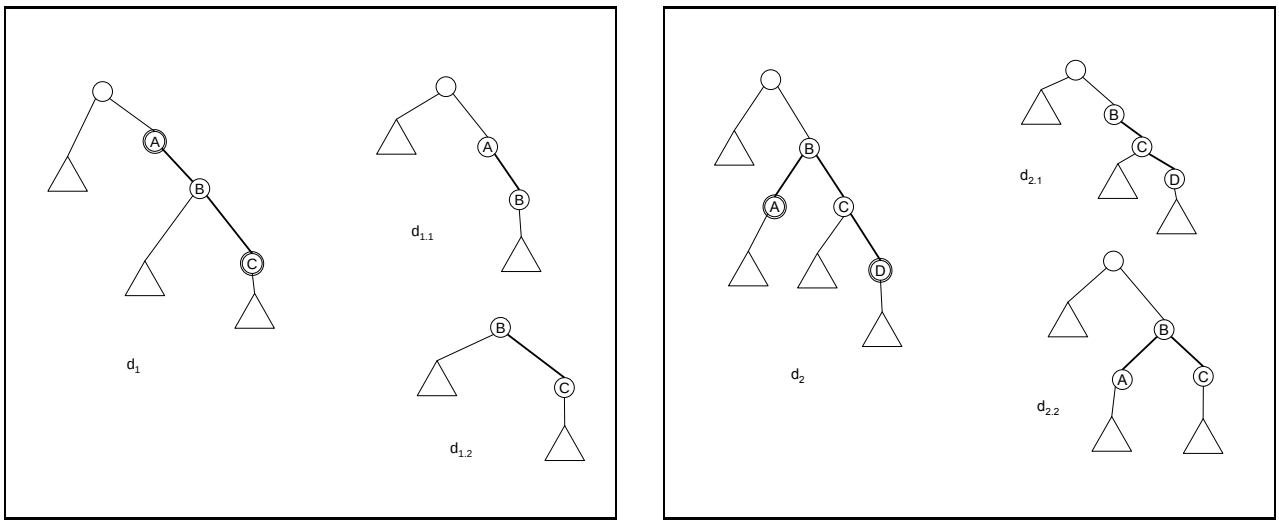


Figure 2. A data tree is split up into two subtrees whenever a candidate pattern tree is infrequent (case one left and case two right).

3.2. Algorithm

To mine trees with monotone constraints we need the following basic property of trees:

> *each pair of vertices is connected by exactly one path.*

From this property it follows that: candidate pattern trees which are infrequent split the data tree into two or more (not necessarily disjoint) subtrees.

We distinguish two cases. In the first case the pattern tree forms a chain of length $n - i$ with $i < n$, $t_1 = (v_i, v_{i+1}, \ldots, v_n)$. In the second case there is exactly one node of this chain that has two children, say node v_j with $i \leq j \leq n$, hence we have two chains $(v_i, v_{i+1}, \ldots, v_n)$ and $(v_j, v_{j,1}, \ldots, v_{j,m})$. For more general patterns the data tree is split into more than two subtrees, but we will not further discuss these cases because it is of no practical use for the pruning algorithm. For the first case, suppose that the extension of t_1 with a node v_{n+1} is infrequent. Then for all data trees $d \in D$ in which t_1 occurs, any frequent pattern in which v_i or any ancestor of v_i occurs can not be extended with v_{n+1}, because the path from an ancestor of v_i to v_{n+1} leads through v_i and the pattern t_1 extended with v_{n+1}, is infrequent so any of its supertrees is also infrequent. The node labeled v_{n+1}, and any of its descendants, can not be extended with the node labeled v_i, because any path from a descendant of v_{n+1} through node v_i leads through v_{n+1}, and hence the infrequent pattern $(v_i, v_{i+1}, \ldots, v_n, v_{n+1})$ must be part of any such path. As an immediate result it follows that for all data trees $d \in D$ in which t_1 occurs, d can be split into two trees d_1 and d_2. Here $d_1 = d \setminus t(v_{n+1})$ and $d_2 = t(v_{i+1})$, where $t(v_k)$ is the subtree of d starting at node v_k and $d \setminus t(v_k)$ is the tree d minus it's subtree starting at node v_k. Note that both d_1 and d_2 share the tree $t(v_{i+1}) \setminus t(v_{n+1})$. For the second case the argumentation is similar, given that the pattern tree consisting of the two chains is extended with a node v_{n+1} on the rightmost path $(v_i, v_{i+1}, \ldots, v_n)$ is infrequent. Then for all data trees $d \in D$ in which the pattern tree occurs, d is split into two trees d_1 and d_2, with $d_1 = d \setminus t(v_{n+1})$ and $d_2 = d \setminus t(v_{j,m})$.

In figure 2 (left) an example is given of the first case, where a frequent pattern A, B is extended with a node C. In case that the extended pattern A, B, C is infrequent, d_1 is split into $d_{1.1}$ and $d_{1.2}$. The second case is displayed in figure 2(right), where the frequent pattern A, B, C is extended with D. The trees $d_{2.1}$ and $d_{2.2}$ are the result of the split of d_2.

Recall that monotone constraints have the following property: if a tree t satisfies a monotone constraint, then all its supertrees also satisfy that constraint. Likewise, if a tree t falsifies a monotone constraint then all subtrees of t falsify that constraint as well. The procedure for pruning with a monotone constraint C is as follows.

For all trees $d \in D$ that satisfy $C(d)$, whenever a candidate tree is generated which is infrequent and of case one or two, check the constraint on the two data trees (obtained after splitting) for all occurrences of this infrequent pattern. Whenever a data tree d does not

satisfy the monotone constraint, the occurrences of a pattern tree t in d are not counted. Furthermore the pattern t is not extended in d; in fact d can be deleted from the database. Suppose that in figure 2 the data tree d_1 satisfies a monotone constraint. Because the pattern (A, B, C) is infrequent, d_1 is split into $d_{1.1}, d_{1.2}$. If one of the data trees $d_{1.1}, d_{1.2}$ does not satisfy the constraint, these data trees can be deleted. Suppose $d_{1.1}$ falsifies the constraint, then $\phi_{d_1}(A, B)$ is decreased (with one, in this case) which results in a decreased support of the pattern (A, B).

4. Experiments

For the experiments we used a slightly modified synthetic data generator provided by Mohamed Zaki. This data generator (described in (Zaki, 2002)) mimics website browsing behavior. Our experiments have been performed on the same dataset, but with different minimum support parameters. The dataset was created by sampling 10000 trees of maximal depth 5 out of a master tree with 10000 nodes and 50 different node labels. Figure 3 shows the distribution of the frequent trees by length for the two minimum support parameters used. Note that in general there are much more possible labeled subtrees of size $n+1$ then of size n. The goal of the experiments is to examine the use of monotone constraints, so we compared the running time and the number of candidates generated for our implementation of FREQT, with the extended version that also performs monotone constraints pruning. We compared the two versions with several values for the constraints, expressed as selectivity. A selectivity of $x\%$ means that $x\%$ of the frequent trees satisfies the constraint. The running time and the number of candidates generated is compared with the basic case for different levels of selectivity, where the basic case (indexed at 100 in figures 4 and 5) is the FREQT algorithm without constraints pruning.

The effects of the opportunistic pruning are more visible for the experiments with a higher support. The maximal speedup obtained with minimum support of 1% is 2.40 while the maximal speedup with minimum support 0.5% is 1.80. The same difference holds for the number of candidate trees generated, for the runs with the lowest percentage of selectivity. With a minimum support of 1% the algorithm generated about 41% of the candidate trees generated without constraint pruning, while for a minimum support of 0.5% the algorithm generated about 60% of the candidate trees. The most likely explanation is that the pruning method we use depends on the occurrence of infrequent patterns. When the number of infrequent patterns generated is decreased (by using a lower minsup value), the effect of pruning with monotone constraints flattens.

5. Conclusion

In this paper we described types of constraints used in frequent itemset mining, and ways of incorporating these constraints in frequent tree mining. We have described an opportunistic pruning method for tree mining that uses monotone constraints. The opportunistic pruning methods leads to a considerable speedup and a reduction in the number of candidates generated compared to the basic algorithm. In future work, we will explore the development of algorithms that directly compute frequent trees that satisfy succinct and convertible constraints. Besides the constraints inherited from frequent item set mining, it makes sense to define constraints on the structure of the data, and develop algorithms that compute the frequent structures that satisfy these constraints. Extending constraints to embedded subtrees, free trees and graphs is another challenge for future work.

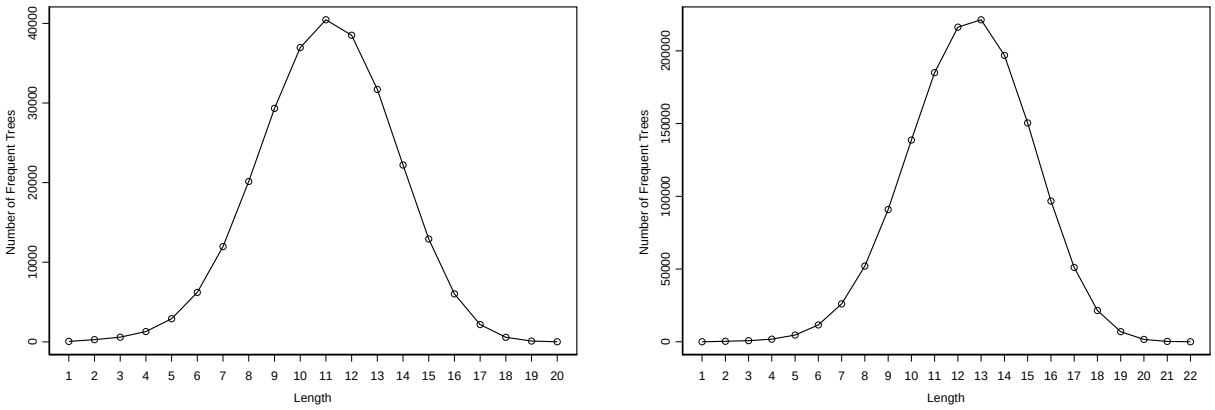


Figure 3. Distributions of frequent trees by their size. The distribution on the left is with a minimum support parameter of 1% on the right minsup = 0.5%.

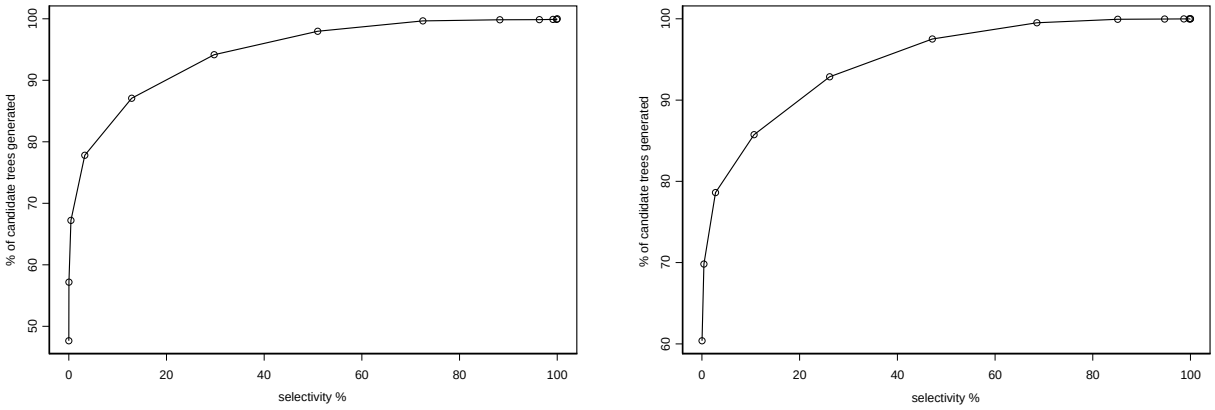


Figure 4. The number of candidate trees generated (as a percentage of the number of candidates generated without constraint pruning) for different levels of selectivity. Left of minsup 1% and right with misup 0.5 %

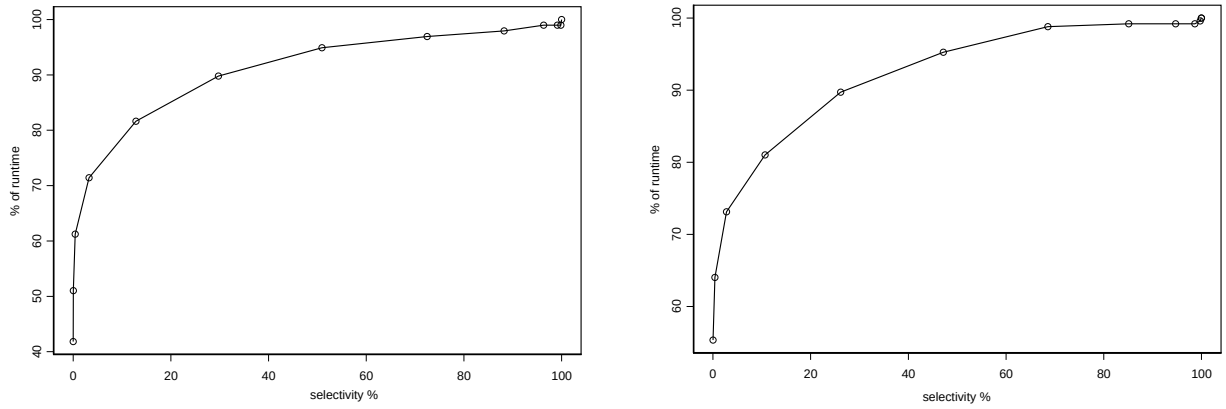


Figure 5. The run time for different levels of selectivity for minsup 1 % left and minsup 0.5 % right.The runtime is here also displayed as a percentage of the runtime without constraint pruning.

References

Agrawal, R., & Srikant, R. (1994). Fast algorithms for mining association rules. *Proc. 20th Int. Conf. Very Large Data Bases, VLDB* (pp. 487–499).

Asai, T., Abe, K., Kawasoe, S., Arimura, H., Sakamoto, H., & Arikawa, S. (2002). Efficient substructure discovery from large semi-structured data. *Proceedings of the Second SIAM International Conference on Data Mining.*

Bayardo, R. (1998). Efficiently mining long patterns from databases. *SIGMOD 1998, Proceedings ACM SIGMOD International Conference on Management of Data, June 2-4, 1998, Seattle, Washington, USA* (pp. 85–93).

Bayardo, R., Agrawal, R., & Gunopulos, D. (1999). Constraint-based rule mining in large, dense databases. *Proceedings of the 15th International Conference on Data Engineering* (p. 188).

Inokuchi, A., Washio, T., & Motoda, H. (2000). An apriori-based algorithm for mining frequent substructures from graph data. *Principles of Data Mining and Knowledge Discovery* (pp. 13–23).

Ng, R. T., Lakshmanan, L. V. S., Han, J., & Pang, A. (1998). Exploratory mining and pruning optimizations of constrained association rules. *SIGMOD 1998, Proceedings ACM SIGMOD International Conference on Management of Data, June 2-4, 1998, Seattle, Washington, USA* (pp. 13–24).

Nijssen, S., & Kok, J. (2003). Efficient discovery of frequent unordered trees. *In Proceedings of the first International Workshop on Mining Graphs, Trees and Sequences (MGTS2003)* (pp. 55–64).

Pei, J., & Han, J. (2000). Can we push more constraints into frequent pattern mining? *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining.*

Wang, K., & Liu, H. (2000). Discovering structural association of semistructured data. *Knowledge and Data Engineering*, *12*, 353–371.

Zaki, M. J. (2002). Efficiently mining frequent trees in a forest. *Proc. of the Int'l Conf. on Knowledge Discovery and Data Mining* (pp. 71–80).

Amplifying the Block Matrix Structure for Spectral Clustering

Igor Fischer FISCHER@NT.UNI-SAARLAND.DE
Telecommunications Lab, Saarland University 22.10, P.O. Box 151150, 66041 Saarbrücken, Germany

Jan Poland JAN@IDSIA.CH
IDSIA, Galleria 2, CH-6928 Manno-Lugano, Switzerland

Abstract

Spectral clustering methods perform well in cases where classical methods (K-means, single linkage, etc.) fail. However, for very non-compact clusters, they also tend to have problems. In this paper, we propose three improvements which we show that perform better in such cases. We suggest that spectral decomposition is merely a method for determining the *block structure* of the affinity matrix. Consequently, it is advantageous for clustering techniques if the affinity matrix has a clear block structure. We propose two independent steps to achieve this goal. In the first, which we term *context-dependent affinity*, we compute point affinities by taking their neighborhoods into account. In the second, the *conductivity method*, we aim at amplifying the block structure of the affinity matrix. Combining these two enables us to achieve a clear block-diagonal structure, despite starting with very weak affinities. For the last step, clustering spectral images, K-means is commonly used. Instead, as a third improvement, we suggest using our *K-lines* algorithm. When compared to other clustering algorithms, our methods display promising performance on both artificial and real-world data sets.

1. Introduction

Consider the standard clustering problem. Let $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N$ be N data points in a metric space. Our goal is to group them into K clusters, such that the distance within the clusters is low whereas the distance between clusters is high. Determining the number of clusters K is itself a non-trivial problem. However, we shall assume here that K is known. Recently, spectral methods have become increasingly popular, together with other kernel methods for machine learning. In spectral clustering, one constructs an affinity or kernel matrix $\boldsymbol{A}$ from the data points and performs a spectral decomposition of $\boldsymbol{A}$, possibly after normalization. Then the dominant eigenvalues and the corresponding eigenvectors are used for clustering the original data. Spectral clustering may be applied in particular in cases where simple algorithms such as K-means fail, such as in Figure 1.

The affinity matrix $\boldsymbol{A}$ is usually interpreted as the adjacency matrix of a graph. Thus spectral clustering algorithms are decomposed into two distinct stages: (a) build a good affinity graph and (b) find a good clustering of the graph. Significant theoretical progress has been made addressing the latter, such as Alpert et al., 1994; Spielman & Teng, 1996; Meilă & Shi, 2001. An optimal graph clustering may be achieved by fulfilling some partitioning criterion (Kannan et al., 2000). The optimization of this criterion is usually NP-hard. A spectral algorithm may thus be regarded as a polynomial time approximation. A recent approach (von Luxburg et al., 2004) considers both sub-problems together and analyzes the properties of the spectral decomposition directly. Our work focusses on the first task of constructing a good affinity matrix. Most commonly the affinities are given by a Gaussian kernel $\boldsymbol{A}(i,j) = \exp\left(-\|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2/(2\sigma^2)\right)$, where $\|\boldsymbol{x}_i - \boldsymbol{x}_j\|$ denotes the Euclidean distance between $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ (Perona & Freeman, 1998; Weiss, 1999; Shi & Malik, 2000; Ng et al., 2002; Verma & Meilă, 2003). Determining the kernel parameter σ is a pivotal issue and greatly influences on the final clustering result. In some cases the "right" σ is "obvious". However, generally it is non-trivial to find a good σ value. A possible method is trying different values for σ and choosing the one which optimizes some quality measure (Ng et al., 2002).

In contrast to many authors who focus their analysis on spectral properties, we believe that the crucial issue is the construction of a good affinity matrix which

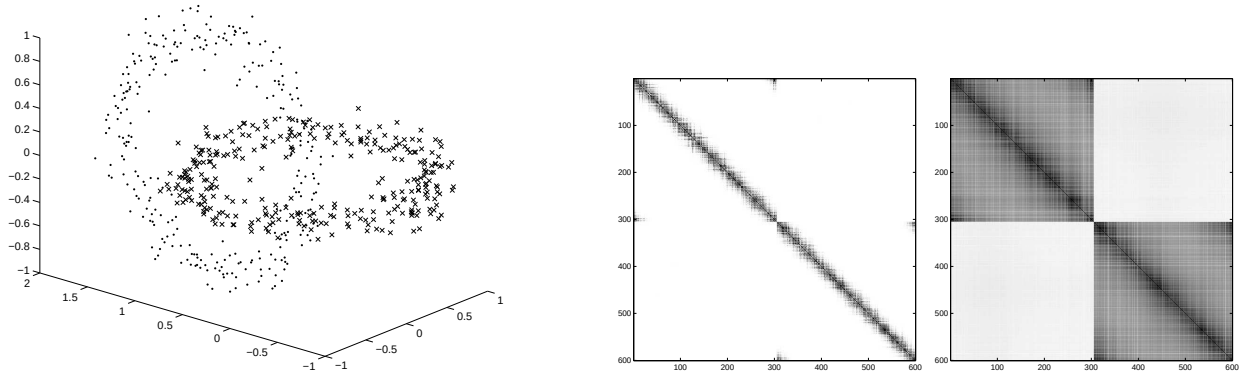


Figure 1. Two interlocked ring clusters: scatter plot of the data (left), affinity matrix $\boldsymbol{A}$ computed with the Gaussian kernel (middle) and the conductivity matrix $\boldsymbol{C}$.

is as block diagonal as possible. In that case spectral decomposition is merely the most convenient way to discover this block structure. In order to amplify the block structure of an affinity matrix, we introduce the *conductivity method*, which is described in Section 2. This allows us to start with a *weak* affinity matrix where the affinities of many points might be close to zero. A context dependent way of constructing such weak affinity matrices is suggested in section 3. For the final clustering of the spectral images, we propose a novel algorithm, termed K-lines (Section 4). Section 5 compares our clustering method to other algorithms. We conclude with a discussion on the scope and limitations of the proposed methods (Section 6).

2. Block Structure Amplification

Consider a block-diagonal affinity matrix $\boldsymbol{A} = \left(\begin{smallmatrix}\mathbf{1} & \mathbf{0}\\ \mathbf{0} & \mathbf{1}\end{smallmatrix}\right)$, where each $\mathbf{1}$ denotes an arbitrary-sized block of ones. The remaining zeros in the matrix are denoted by $\mathbf{0}$s. This block structure is easily discovered, e.g. by looking at the two dominant eigenvectors, i.e. the eigenvectors corresponding to the two dominant eigenvalues, of $\boldsymbol{A}$. If the blocks are of different sizes, then the space spanned by eigenvectors is not rotation invariant. In that case, even the first eigenvector is sufficient for identifying the blocks. Furthermore, permutation of points does not make an essential difference, this only leads to a permutation of rows and columns of the matrix. The eigenvectors will be permuted respectively and the eigenvalues will remain unchanged. Therefore we also consider matrices to be block-diagonal if they are block-diagonal after a permutation of their rows and columns.

An affinity matrix generated from real-world data is virtually never block-diagonal. If the clusters are nicely shaped (e.g. Gaussian) and well-separated, then the affinity matrix may be approximately block diagonal. For brevity, we will also refer to such matrices as block-diagonal. In contrast, consider the data set in Figure 1. Although the two rings are interlocked (like two links in a chain) and therefore not separated in a Euclidean sense, we tend to regard them as two distinct clusters. In this setting, the clustering intuition relies on continuous concentration of data points, a notion which was applied by (Ben-Hur et al., 2001). Two points belong to a cluster if they are close to each other, or if they are well connected by paths of short "hops" over the other points. The more such paths exist, the higher the chances are that the points belong to the same cluster. However, it is not immediately clear in this case how to obtain a good affinity matrix for clustering. Suppose a Gaussian kernel is used, then a large σ merges the clusters, resulting in an undesirable clustering. Choosing a small σ leads to a matrix which is approximately composed of two band-diagonal matrices. We will denote such matrices *block-band*.

According to theoretical results using matrix perturbation theory (Ng et al., 2002) and empirical evidence (Section 5), spectral clustering can produce satisfactory results when the matrix is block-diagonal. For block-band matrices, the case is less clear from the theoretical point of view. The δ of the eigengap condition presented in (Ng et al., 2002) tends to zero when increasing the size of the sample, yet maintaining the same fixed band structure. As a consequence, the theory cannot guarantee a good solution. Simulations suggest that in practice the spectral clustering of a block-band matrix is good. However, the choice of σ in this case is significantly more difficult than for block-diagonal matrices. We must set σ small enough

to obtain a *weak* affinity matrix, i.e. where only the affinities of directly neighboring points are high. On the other hand, σ must not be too small. Therefore, in some cases, the range of admissible σ's can be very narrow.

These considerations motivate a change of the affinity measure. Specifically, we want to *amplify* weak affinities in matrix $\boldsymbol{A}$. Instead of considering two points similar if they are connected by a high-weight edge in the graph, we assign them a high affinity if the overall graph *conductivity* between them is high. We define conductivity as for electrical networks, i.e. the conductivity of two points depends on *all* paths between them. This measure should not be confused with the graph "conductance" by (Sinclair & Jerrum, 1989), which is – up to a constant factor – equivalent to the normalized cuts (Shi & Malik, 2000). In the normalized cuts approach, the overall flow between two disjunct graph parts is considered. Our approach is, in a sense, complementary: we consider the flow between any two points, and construct a new graph based on it.

The conductivity for any two points $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ is computed in the following way. We first solve the system of linear equations:

$$\boldsymbol{G} \cdot \boldsymbol{\varphi} = \boldsymbol{\eta}_{ij} \tag{1}$$

where $\boldsymbol{G}$ is a $N \times N$ matrix constructed from the original affinity matrix $\boldsymbol{A}$:

$$\boldsymbol{G}(p,q) = \begin{cases} \text{if } p = 1: & \begin{cases} 1 & \text{if } q = 1 \\ 0 & \text{else} \end{cases} \\ \text{else}: & \begin{cases} \sum_{k \neq p} \boldsymbol{A}(p,k) & \text{if } p = q \\ -\boldsymbol{A}(p,q) & \text{else} \end{cases} \end{cases} \tag{2}$$

and $\boldsymbol{\eta}_{ij}$ is an indicator vector of length N. It represents points $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ for which we want to compute the conductivity, and is composed as follows:

$$\boldsymbol{\eta}_{ij}(k) = \begin{cases} -1 & \text{for } k = i \text{ and } i > 1 \\ 1 & \text{for } k = j \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

Solving Equation 1 gives us vector $\boldsymbol{\varphi}$. The conductivity between $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$, $i < j$ is then given by

$$\boldsymbol{C}(i,j) = 1/\left[\boldsymbol{\varphi}(j) - \boldsymbol{\varphi}(i)\right] \tag{4}$$

which, due to the fact that $\boldsymbol{\eta}_{ij}$ is extremely sparse, can be simplified to

$$\boldsymbol{C}(i,j) = 1/\left[\boldsymbol{G}^{-1}(i,i) + \boldsymbol{G}^{-1}(j,j) - \boldsymbol{G}^{-1}(i,j) - \boldsymbol{G}^{-1}(j,i)\right] \tag{5}$$

so that $\boldsymbol{\eta}_{ij}$ never explicitly appears. Due to the symmetry, it follows that $\boldsymbol{C}(i,j) = \boldsymbol{C}(j,i)$. The diagonal elements $\boldsymbol{C}(i,i)$ can be set to $\max_{i,j} \boldsymbol{C}(i,j)$. It therefore suffices to compute $\boldsymbol{G}^{-1}$ only once, in $O(N^3)$ time, and to compute the conductivity matrix $\boldsymbol{C}$ in $O(N^2)$ time.

An intuitive motivation for the above method comes from electrical engineering, where the method is known as node analysis. We consider a resistor network, where *direct* conductivities (i.e. inverse resistances) between two nodes i and j are denoted by G_{ij}. To compute the *overall* conductivity between the nodes, we measure the voltage U_{ij} between them when we let a known current I enter the network at node i and leave it at the node j. The overall conductivity is then given by Ohm's law: $G_{ij} = I/U_{ij}$. The voltage is defined as the potential difference between the nodes: $U_{ij} = \varphi_j - \varphi_i$, and the potentials can be computed from Kirchhoff's law, stating that all currents entering a node i must also leave it: $\sum_{j \neq i} I_{ij} = 0$. Applying Ohm's law again, the currents can be expressed over voltages and conductivities, so that this Equation becomes: $\sum_{j \neq i} G_{ij} U_{ij} = \sum_{j \neq i} G_{ij}(\varphi_j - \varphi_i) = 0$. Grouping the direct conductivities by the corresponding potentials and formulating the equation for all nodes, we obtain the Equation (1). The vector $\boldsymbol{\eta}$ represents the known current I, which we have transferred to the right side of the equation.

As we know from graph theory, the adjacency matrix of a connected graph with N nodes is of rank $N-1$. In our case that means that, if we would compose $\boldsymbol{G}$ relying only on Kirchhoff's and Ohm's law, its rows would sum to zero. In other words the system would be undetermined. In a physical sense, currents entering and leaving $N-1$ nodes determine also the currents in the N-th node, since they have nowhere else to go. In order to obtain a determined system, we have to choose a node and fix it to a known potential, so it becomes the reference node. In our method we set the potential of the first node to zero ($\boldsymbol{\varphi}(1) = 0$), which is reflected by the way the first rows of $\boldsymbol{G}$ and $\boldsymbol{i}$ are defined in Equations (2) and (3).

The method here seems to require using a different $\boldsymbol{\eta}_{ij}$ and solving the equations anew for all $\binom{N}{2}$ pairs of nodes. That would be the computational analogy of connecting the current source between every pair of nodes and measuring the voltage. This, fortunately, is not the case: First, since direct conductivities between nodes do not change, it suffices to invert the matrix $\boldsymbol{G}$ only once. And second, for computing the overall conductivity between two nodes, we do not need all voltages in the network; the voltage between these nodes

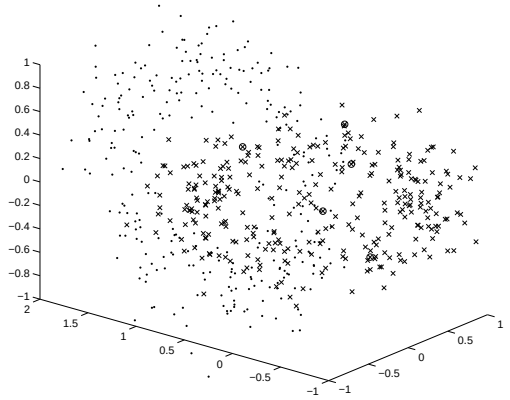


Figure 2. Two interlocked ring clusters with $\sigma = 0.2$

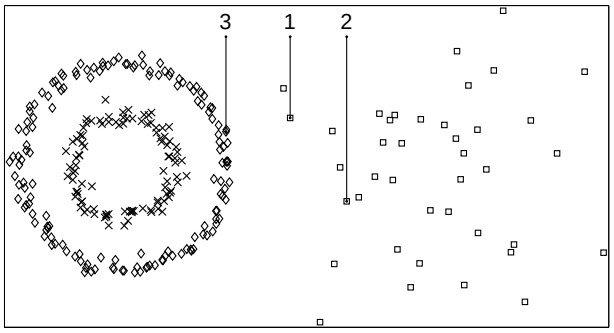


Figure 3. Two rings with low dispersion and a Gaussian cluster with large dispersion

suffices. This allows us to observe only two rows in the $\boldsymbol{G}^{-1}$ matrix. Furthermore, the fact that for each vector $\boldsymbol{\eta}_{ij}$, all except two components are zeros (i.e. the external current source is attached only to two nodes), entails that we only need to consider two columns in $\boldsymbol{G}^{-1}$. Consequently, the conductivity between any two nodes can be computed from only four elements of the matrix $\boldsymbol{G}^{-1}$, as Equation (5) shows.

The resulting conductivity matrix $\boldsymbol{C}$ is obtained from the matrix $\boldsymbol{A}$, but displays a reinforced block structure. We may use $\boldsymbol{C}$ for the remainder of the spectral clustering task. But, before we proceed with it, we need to know how to construct $\boldsymbol{A}$.

Observe that (2) almost is the Laplacian of A, except for the first row. The first row may be regarded as a "trick" to make the matrix invertible. Thus, we gave a different motivation for using the Laplacian. In this light, observe the high similarity of our algorithm to that of Saerens et al. (2004). They consider a random walk criterion on the graph. Thus they arrive at a formula which is almost identical to (5), except for the inverse which is replaced by the pseudo-inverse of the Laplacian.

3. Constructing Affinity Matrices

In this section we consider the question of how to construct affinity matrices $\boldsymbol{A}$, in particular weak ones. We restrict our discussion to the case that the affinities are Gaussian: $\boldsymbol{A}(i,j) = \exp\left(-\|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2/2\sigma_{ij}^2\right)$, where $\|\boldsymbol{x}_i - \boldsymbol{x}_j\|$ denotes the Euclidean distance between $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$. Often, σ is set to a global value ($\sigma_{ij} \equiv \sigma$). In this case the performance of spectral clustering depends heavily on the selection of σ. A common selection method is to try different values for σ and use the best one. This process can be automated in an unsupervised way as suggested by (Ng et al., 2002), since concentration of the spectral images may be an indicator for the quality of the choice of σ. Another possibility is to make use of the *distance histogram*. If the data form clusters, then we expect the histogram of their distances to be multi-modal. The first mode would correspond to the average intra-cluster distance and others to between-cluster distances. By choosing σ around the first mode, the affinity values of points forming a cluster can be expected to be significantly larger than others. Consequently, the affinity matrix is likely to resemble a block diagonal matrix. For a weaker affinity matrix, σ may and should be chosen smaller, below the first histogram mode. This is the case when using the conductivity method, which itself amplifies the weak affinities. We have found out empirically that, for this method, a good choice is at about half the position of the first peak in the distance histogram, or somewhat below.

This distance histogram heuristic has the drawback that it cannot be automated in a robust way. Moreover, for some data sets, finding a single value of σ that performs equally well over the whole data set might not be even possible. Consider the two concentric rings and a Gaussian cluster in Figure 3. The points in the third, Gaussian cluster have a significantly larger dispersion than the points in the two rings. So each of the points in the third cluster will be connected to almost no other point than itself if σ is chosen correctly for the rings. However, a human would probably apply a *context-dependent* affinity in this case. For example, the affinity of the points labelled 1 and 2 should be higher than the affinity of points 1 and 3, although the latter distance is smaller.

We therefore propose to use a context-dependent method for constructing the affinity matrix, rather than using a single σ for all points. For each point $\boldsymbol{x}_i$ we suggest to choose a different σ_i. This results

in an asymmetric similarity matrix, which is given by $\tilde{\boldsymbol{A}}(i,j) = \exp\left(-\|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2/2\sigma_i^2\right)$. In order to determine each σ_i, we enforce the following condition:

$$\sum_{j=1}^{n} \exp\left(-\frac{\|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2}{2\sigma_i^2}\right) = \tau \qquad (6)$$

for each $1 \leq i \leq n$, where $\tau > 0$ is a constant. That is, we choose σ_i such that the row sum of $\boldsymbol{A}$ assumes the fixed value τ. We may interpret τ as the fixed *neighborhood size*.

Choosing τ is easier than choosing σ, since τ is scale invariant. We will see in Section 5 that, in contrast to σ, the clustering result is not very sensitive to the value of τ. In order to obtain weak affinities, τ is set to a small value, such that only immediately neighboring points obtain a significant affinity. We suggest the choice of $\tau = 1 + 2D$, where D is the dimension of the data — there should be two neighbors for each dimension, plus one because the point is neighbor of itself. This value has been used for all simulations below. Now we can determine σ_i which satisfies Equation (6) by iterative bisection. Few iterations are sufficient, since there is no great precision required.

In order to proceed with the conductivity matrix, we need to obtain a *symmetric* matrix $\boldsymbol{A}$ from the asymmetric matrix $\tilde{\boldsymbol{A}}$. Again for a weak affinity matrix, high values should only exist between immediately neighboring points. Thus the affinity between $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ should be the *smaller* of the respective values, $\boldsymbol{A}(i,j) = \min\{\tilde{\boldsymbol{A}}(i,j), \tilde{\boldsymbol{A}}(j,i)\}$. This ensures that, for example, the points 1 and 3 in Figure 3 obtain a lower affinity value than the points 1 and 2. This is because point 1, although being nearer to point 3, actually lies in the relative neighborhood of point 2.

The idea to consider a context-dependent neighborhood is not new. A very interesting parallel approach is given by Rosales and Frey (2003). While they suggest and motivate an EM-like repeated update of the neighborhood size, our method is heuristic and computationally cheaper. Compare also the row normalization by Meilă and Shi (2001). Very recently, Zelnik-Manor and Perona (2004) have suggested another local neighborhood approach which is very similar to ours.

4. Clustering the Spectral Images

Given the conductivity matrix $\boldsymbol{C}$ (or the affinity matrix $\boldsymbol{A}$, if conductivity is not applied), we compute the matrix of the dominant eigenvectors $\boldsymbol{V} = \left[\boldsymbol{v}_1 \dots \boldsymbol{v}_K\right] \in \mathbb{R}^{N\times K}$, that is the eigenvectors which correspond to the K largest eigenvalues. Then we call $\left[\boldsymbol{y}_1 \dots \boldsymbol{y}_N\right] = \boldsymbol{Y} = \boldsymbol{V}' \in \mathbb{R}^{K\times N}$ the *spectral images* of the original data $\boldsymbol{x}_1, \dots, \boldsymbol{x}_N$.

There are different ways to compute the final clustering from the spectral images $\boldsymbol{y}_1, \dots, \boldsymbol{y}_N$. The images of points belonging to the same cluster were experimentaly observed to be distributed angularly around straight lines — the more compact the cluster, the better alignment of the spectral images. A simple, off-the-shelf clustering algorithms like K-means cannot be used, because it assumes approximately round clusters. To circumvent the problem, (Ng et al.) suggest projecting the points on the unit sphere and then using K-means. However, this leads to a certain loss of information (namely the radius), and the resulting clusters are still not round. We propose a similar algorithm here, which exploits the observed structure of the clusters and which we have found to be slightly superior in the simulations (compare a parallel approach, the K-planes algorithm in (Bradley & Mangasarian, 2000)).

Algorithm K-lines
Each cluster is given by a line through the origin, represented by a vector $\boldsymbol{m}_i \in \mathbb{R}^k$ of unit length, $1 \leq i \leq k$.

1 Initialize $\boldsymbol{m}_1 ... \boldsymbol{m}_K$ (e.g. by the canonical unit vectors)
2 For each $i \in \{1, \dots, k\}$, let S_i be the set of points containing all points $\boldsymbol{y}_j$ that are closest to the line defined by $\boldsymbol{m}_i$
3 For each $i \in \{1, \dots, k\}$, let $\boldsymbol{m}_i$ define the line through the origin which has smallest sum square distance to all points in S_i
4 Repeat from 2 until convergence

Our final spectral clustering algorithm then reads as follows.

Algorithm Context dependent clustering with block amplification

1 Calculate the context dependent affinity matrices $\tilde{\boldsymbol{A}}$ and $\boldsymbol{A}$.
2 Construct the conductivity matrix $\boldsymbol{C}$ according to (5).
3 Determine K principal eigenvectors, e.g. by Lanczo's method, and compute the spectral images $\boldsymbol{y}_1, \dots, \boldsymbol{y}_n$.
4 Cluster $\boldsymbol{y}_1, \dots, \boldsymbol{y}_n$ by K-lines.

5. Simulations

We present simulation results and compare our proposed method with three other clustering algorithms: standard K-means, a more sophisticated, kernel-based expectation-maximization method (Girolami, 2002),

Name	N	K	D	K-means	Girolami	Ng et al.	Our-σ	Our-τ
2R3D.1	600	2	3	197 (197.9 ± 1.0)	**0** (162.2 ± 86.4)	**0** (17.7 ± 70.4)	**0** (0.0 ± 0.0)	**0**
2R3D.2	600	2	3	195 (195.1 ± 0.2)	68 (202.9 ± 54.6)	**4** (7.5 ± 28.9)	6 (8.2 ± 1.9)	93
2RG	290	3	2	110 (125.3 ± 2.4)	66 (121.7 ± 26.5)	101 (102.2 ± 3.8)	2 (16.8 ± 38.0)	**0**
2S	220	2	2	26 (34.5 ± 6.4)	25 (49.9 ± 26.6)	**0** (9.3 ± 10.8)	97 (101.8 ± 4.8)	**0**
4G	200	4	3	24 (40.0 ± 24.2)	2 (2.6 ± 5.8)	18 (35.1 ± 22.2)	2 (3.0 ± 7.0)	**1**
5G	250	5	4	41 (62.0 ± 27.8)	13 (21.2 ± 14.1)	33 (40.1 ± 14.6)	13 (34.5 ± 14.6)	**11**
2Spi	386	2	2	191 (192.3 ± 0.5)	151 (178.5 ± 10.7)	**0** (79.1 ± 95.4)	**0** (39.4 ± 55.5)	193
Iris	150	3	4	16 (27.8 ± 22.2)	8 (10.3 ± 2.2)	14 (14.8 ± 3.6)	10 (12.3 ± 5.3)	**7**
Wine	178	3	13	5 (8.7 ± 10.5)	**3** (3.9 ± 0.9)	**3** (3.3 ± 0.5)	12 (18.9 ± 6.6)	65
BC	683	2	9	26 (26.5 ± 0.5)	**20** (21.5 ± 1.0)	22 (22.8 ± 0.6)	21 (25.1 ± 2.0)	**20**

Table 1. Empirical comparison of the algorithms for different data sets. N denotes the number of points in the set, K the number of clusters, and D the data dimensionality. For each algorithm we document the minimum number of incorrectly clustered points and the mean and standard deviation (due to using a different σ in each run) in brackets. The respective best value is bold. We present two variants of our algorithm: "Our-σ" denotes the algorithm with a global kernel width determined from the distance histogram. "Our-τ" denotes the algorithm with a point-specific σ_i, according to Eq. 6.

and the algorithm suggested by (Ng et al.). Whereas K-means simply assumes K spherical clusters and looks for their centers, the (Girolami) method projects the data into a feature space and clusters them there using a standard EM method. The (Ng et al.) algorithm differs from ours in the way it computes and scales the affinity matrix, and by using K-means instead of K-lines in the final step.

All algorithms are evaluated on a number of benchmark and real-world data sets. Assessing the quality of a clustering algorithm is not easy. First, clustering is an ill-posed problem. We overcome this difficulty by defining an "optimal" reference clustering: For the artificial data sets, we know the probability distribution which generated the data, whereas for the real world data sets labels are available which we use for reference (but which are unknown to the clustering algorithm). Another problem is that the outcome is partly very sensitive to the choice of the kernel parameter σ. In order to give a somewhat realistic evaluation of the (Girolami) and the (Ng et al.) algorithms which depend on this parameter, the experiments were performed in two phases: First, for each data set we systematically scanned a wide range of σ's (the range was set manually to cover most of the distance histogram) and ran the clustering algorithms. We denote by σ^* the σ which produced the best results. In the second phase, we performed 100 runs, with σ randomly sampled form $[\sigma^* \pm 20\%]$. The rationale for this procedure is that the optimal σ cannot be known in real-world application, but the experimenter will probably be able to guess it with some precision (which we suppose to be somewhere around $\pm 20\%$) based on the distance histogram. The automatic computation of σ, as suggested by (Ng et al., 2002), sometimes led to suboptimal σ and, consequently, to poor results. In most cases, the histogram method resulted to a σ very close to the best value and led to very similar results. We present the best result found as well as the mean and standard deviation. In contrast, our context-dependent algorithm produced deterministic results, because K-lines was initialized deterministically, and the neighborhood size was set to $\tau = 2D+1$, where D is the dimension of the data. Therefore, for this algorithm we refrain from quoting the mean and standard deviation. The experimental results are summarized in Table 1.

The data set *2R3D.1* was presented in Figure 1 as a motivation for conductivity. It is an artificial data set, obtained by dispersing points around two interlocked rings in 3D. The dispersion is Gaussian, with standard deviation equal to 0.1. This set is impossible to cluster by K-means, but other algorithms have no problem with it. Conductivity-based algorithm (*Our-σ*) is, however, more stable than competing algorithms. *2R3D.2* is similar, but with the double standard deviation of the points around the rings. Here, all algorithms start having problems. Although *Ng et al* algorithm is the best considering the number of false assignments, it is less stable than the conductivity method. The context-dependent approach is of no advantage here and is clearly outperformed, since the dispersion of points in clusters is constant. The ring segments passing through the center of the other ring are incorrectly assigned to that ring. Our interpretation is that the context-sensitive kernel width acts as a drawback in this situation, since it actually abolishes the already weak gap between the clusters.

The next four data sets have varying data dispersions and thus the context-dependent approach is clearly superior. *2RG* is the data set from Figure 3, *2S* consists of two S-shaped clusters with inhomogeneous dispersion in 2D, and *4G* and *5G* have four and five Gaussian

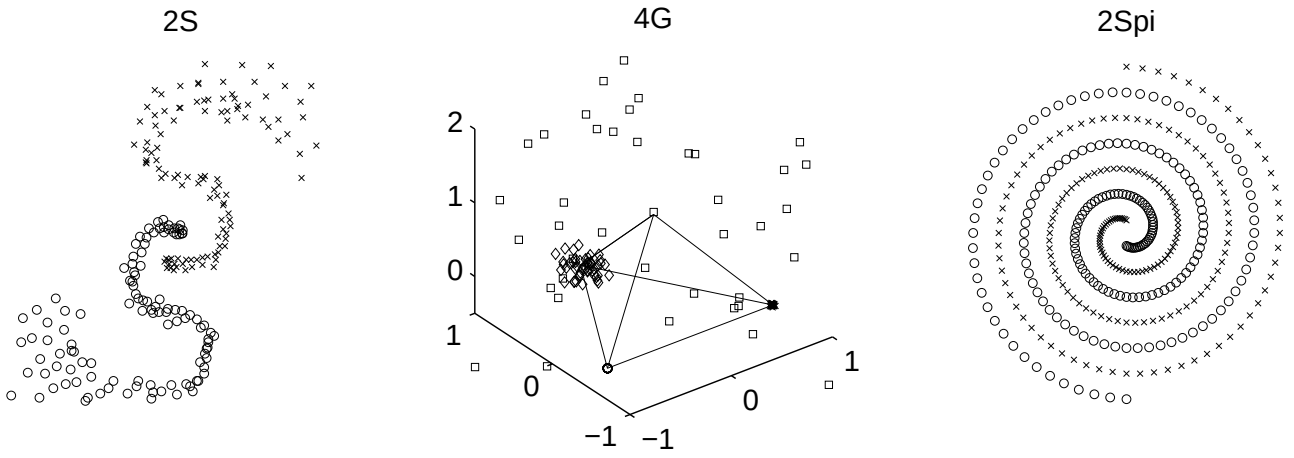


Figure 4. Data sets, left-to-right: *2S*, *4G*, and *2Spi*.

clusters with different dispersion in 3D and 4D, respectively. Due to the different dispersions of clusters and, in the case of *2S* within clusters, no obvious global σ can be chosen. *2Spi* is a variant of Wieland's two spirals (see Fahlman, 1988) with double point density (193 points per spiral). Contrary to our expectations this is one of the examples where the context-dependent approach fails; it is only successful on the fourfold density problem.

The last three data sets are common benchmark sets with real-world data (Murphy & Aha, 1994): the iris, the wine and the breast cancer data set. Both our methods perform very well on iris and breast cancer. However, the wine data set is too sparse for context-dependent method: only 178 points in 13 dimensions, giving the conductivity too many degrees of freedom to connect the points. If the context-dependent method is used without the conductivity, then the algorithm also achieves a good clustering, with only four false assignments.

Our algorithm yields good results on most of the data sets. In addition, it is very insensitive to the choice of the neighborhood size τ. For example, all results remain the same if the constant value $\tau = 10$ is used instead of $\tau = 2D + 1$. The *Ng et al.* algorithm also produces very good experimental results, if σ is well chosen. However, the range of good values for σ may be very small. It this case, conductivity can significantly improve robustness. For the data sets *2R3D.2* and *2Spi* for example, a symmetric algorithm which uses conductivity displays a variance much lower than for the *Ng et al.* algorithm (Table 1). The increased robustness of the conductivity method can also be observed in the computation time required for the spectral decomposition (with the implementation of Lanczo's method coming with MATLAB). Computing the spectrum of the conductivity matrix needs only about 70% of the time which is needed for computing a block-band matrix. Finally, we note that our context-dependent way of constructing affinities can handle data which is inhomogeneously distributed. It works well if data is sufficiently dense and tends to fail if the data is too sparse.

6. Conclusions

Amplifying the block structure of the affinity matrix can improve spectral clustering. This method allows us to start with a weak affinity matrix, where only immediately neighboring points have significant affinities. The conductivity method yields a matrix which is relatively close to block diagonal. Such matrix may be used for clustering with spectral methods. For constructing a weak affinity matrix, we have proposed a context-dependent method which is easily automated and not sensitive to its parameter, the neighborhood size τ. This affinity measure compares favorably with the Gaussian kernel. For the clustering of the spectral images, we have proposed the K-lines algorithm. The resulting spectral clustering algorithm is fully automatic and very robust, and it displays good performance in many cases. It tends to fail on sparse data. All three methods we proposed – conductivity, context-dependent affinity, and K-lines – can be used independently from each other as components in (spectral) clustering algorithms.

We have motivated our methods by purely intuitive and empirical means; a theoretical justification remains open. Future issues to investigate would be: (a) Study the properties of the conductivity matrix with

respect to graph partitioning criteria. (b) Give quantitative assertions on the spectrum of the conductivity matrix. (c) Find out more about the distribution of the spectral images, thus giving a sound foundation of K-lines or another method.

Authors generally agree that it is still incompletely understood how and why spectral clustering works. However, this might be the wrong question to pose. Spectral decomposition might be only one convenient way to discover the block structure of the affinity matrix, there may be other ways to achieve this. In our opinion, the central issue is constructing a "good" affinity matrix from the data with a block structure as expressed as possible.

7. Acknowledgements

We would like to thank Efrat Egozi, Michael Fink, Yair Weiss, and Andreas Zell, for helpful suggestions and discussions. Igor Fischer is supported by a Minerva fellowship, and Jan Poland by BMBF grant 01 1B 805 A/1 and by SNF grant 2100-6712.02.

References

Alpert, C., Kahng, A., & Yao, S. (1994). *Spectral partitioning: The more eigenvectors, the better* (Technical Report). UCLA CS Dept. Technical Report #940036.

Ben-Hur, A., D. Horn, H. S., & Vapnik, V. (2001). Support vector clustering. *Journal of Machine Learning Research*, *2*, 125–137.

Bradley, P. S., & Mangasarian, O. L. (2000). k-plane clustering. *Journal of Global Optimization*, *16*, 23–32.

Fahlman, S. (1988). Faster-learning variations on back-propagation: An empirical study. *Proceedings of the 1988 Connectionist Models Summer School* (pp. 38–51). San Mateo, CA, USA: Morgan Kaufmann.

Girolami, M. (2002). Mercer kernel-based clustering in feature space. *IEEE Transactions on Neural Networks*, *13*, 780–784.

Kannan, R., Vempala, S., & Vetta, A. (2000). On clusterings: Good, bad and spectral. *Proceedings of the 41st Symposium on Foundations of Computer Science*.

Meilă, M., & Shi, J. (2001). Learning segmentation by random walks. *Advances in Neural Information Processing Systems 13* (pp. 873–879). MIT Press.

Murphy, P., & Aha, D. (1994). UCI repository of machine learning databases.

Ng, A., Jordan, M., & Weiss, Y. (2002). On spectral clustering: Analysis and an algorithm. *Advances in Neural Information Processing Systems 14*. Cambridge, MA: MIT Press.

Perona, P., & Freeman, W. (1998). A factorization approach to grouping. *Lecture Notes in Computer Science*, *1406*, 655–670.

Rosales, R., & Frey, B. (2003). Learning generative models of affinity matrices. *19th Conference on Uncertainty in Artificial Intelligence (UAI)*.

Saerens, M., Fouss, F., Yen, L., & Dupont, P. (2004). The principal component analysis of a graph, and its relationship to spectral clustering. *Proceedings of the 15th European Conference on Machine Learning (ECML 2004)* (pp. 371–383). Springer.

Shi, J., & Malik, J. (2000). Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *22*, 888–905.

Sinclair, A., & Jerrum, M. (1989). Approximate counting, uniform generation and rapidly mixing markov chains. *Information and Computation*, *82*, 93–133.

Spielman, D. A., & Teng, S. (1996). Spectral partitioning works: Planar graphs and finite element meshes. *IEEE Symposium on Foundations of Computer Science* (pp. 96–105).

Verma, D., & Meilă, M. (2003). *A comparison of spectral clustering algorithms* (Technical Report). UW CSE. Technical report 03-05-01.

von Luxburg, U., Boudquet, O., & Belkin, M. (2004). Towards convergence of spectral clustering on random samples. *COLT* (pp. 457–471).

Weiss, Y. (1999). Segmentation using eigenvectors: A unifying view. *ICCV (2)* (pp. 975–982).

Zelnik-Manor, L., & Perona, P. (2004). Self-tuning spectral clustering. NIPS 2004, to appear.

Maximizing Expected Utility in Coevolutionary Search

Edwin D. de Jong DEJONG@CS.UU.NL
Decision Support Systems Group, Institute of Information and Computing Sciences
Utrecht University, PO Box 80.089, 3508 TB Utrecht, The Netherlands.

Abstract

Coevolution can be used to adaptively choose the tests used for evaluating candidate solutions. A long-standing question is how this dynamic setup may be organized to yield reliable search methods. Recently, monotonic coevolution algorithms have been proposed for several solution concepts. Here, we introduce a new algorithm that guarantees monotonicity for the solution concept of maximizing the expected utility of a candidate solution. The method, called MaxSolve, is compared to the IPCA algorithm and found to perform more efficiently for a range of parameter values.

Introduction

Coevolution (Barricelli, 1962; Axelrod, 1987; Miller, 1989; Miller, 1996; Hillis, 1990; Koza, 1992; Lindgren, 1992; Kauffman & Johnsen, 1992) offers methods for problems where the quality of individuals is determined by tests. For example, in two-player games, the quality of a first-player strategy is somehow determined by its outcome against all possible opponents. Other test-based problem domains include concept learning and function approximation.

Since the number of tests in a test-based problem can be very large, evaluating individuals on all possible tests is typically infeasible. Another option is to define a heuristic evaluation function, or to select a fixed set of tests for evaluation purposes, but in both of these approaches it is unclear to what extent the resulting biased evaluation function suits its purpose.

Coevolution selects the tests used in evaluation *adaptively*; next to the population of candidate solutions, a population of tests is maintained that are used to evaluate the candidate solutions. It has been shown that such a setup can in principle yield *ideal evaluation* (De Jong & Pollack, 2004), equivalent with evaluating against all tests, using only a small set of tests. This is possible because at every point in time, only the relations between the current candidate solutions need to be evaluated. Thus, a set of tests that reveal all relations between the existing candidate solutions is sufficient.

While the theoretical possibility of ideal evaluation is interesting in itself, an important practical question is to what extent this ideal can be achieved. The DELPHI algorithm we presented in earlier work (De Jong & Pollack, 2004) is designed to approximate ideal evaluation, and is based on the idea of searching for an ideal evaluation set as an inner loop inside the search algorithm. Since it cannot always be known whether an ideal evaluation set has already been reached, the DELPHI algorithm can only approximate ideal evaluation. Furthermore, the algorithm has a limited potential for exploration.

A second approach to achieving reliable progress in coevolution is the use of an archive that guarantees monotonic progress, i.e. progress such that a distance function exists for which the distance to the solution concept decreases with every change to the archive. The IPCA algorithm (De Jong, 2004a; De Jong, 2004b) guarantees monotonicity, meaning any changes to the archive result in progress.

The IPCA algorithm is based on the solution concept of the Pareto-Optimal Equivalence Set, and converges to this set for any finite search space if the generator of new individuals is sufficiently explorative. Specifically, it must be capable of generating all combinations of individuals with some non-zero probability. The Pareto-optimal set is the set of solutions that provide a maximal trade-off between the different objectives, so that a solution's performance in an objective cannot be improved without reducing its performance in other objectives. Since every test is viewed as an objective in this setup, the Pareto-optimal set may be very large. An important question for the practice of coevolution therefore is whether monotonic archives for different solution concepts can be designed.

In this paper, we first define several main solution concepts that can be used in coevolution. Next, for one of these solution concepts, we design an algorithm that guarantees monotonicity. To test the degree to which this new method addresses the above limitation of IPCA, we compare the two methods in experiments. The results show that the new method offers a trade-off between archive size and performance, and for intermediate choices of the parameter that governs this trade-off, the algorithm outperforms IPCA.

The paper is structured as follows Section 1 defines several main solution concepts for coevolution. Section 2 discusses related work. The MaxSolve algorithm is presented in Section 3. Section 4 defines the discretized COMPARE-ON-ONE problem. Results are presented in Section 5, followed by conclusions.

1. Solution Concepts for Coevolution

In this section, we define several main solution concepts that can be used in coevolution.

We use the following notation. The set of all possible candidate solutions is denoted as $\mathbb{C}$, and the set of all possible tests as $\mathbb{T}$. The outcome of a test T for a candidate C may in principle come from any ordered set. Without loss of generality, it will be assumed to be a real number here. The *interaction function* used to determine this outcome is written as G (for Game): $G(C,T) \rightarrow \mathbb{R}$.

1.1. S0: Simultaneous Maximization of All Outcomes

The first solution concept requires an optimal solution C to maximize the outcome over all possible tests simultaneously, as formalized in the following requirement:

$$S0 \ = \ \{C \in \mathbb{C} | \forall C' \in \mathbb{C} : \forall T \in \mathbb{T} : G(C,T) \geq G(C',T)\}$$

This solution concept has a limited application scope, as for many problems there does not exist a single solution that simultaneously maximizes the outcome of all possible tests.

1.2. S1: Maximization of Expected Utility

The second solution concept specifies as a solution the individuals that maximize the expected score against a randomly selected opponent:

$$S1 \ = \ \{C \in \mathbb{C} | \forall C' \in \mathbb{C} : E(G(C,T)) \geq E(G(C',T))$$

where E is the expectation operator and T is randomly drawn from $\mathbb{T}$. This solution concept is appropriate for many problems, for example identifying the best chess player. It is equivalent to maximizing the sum of an individual's outcomes over all tests, or to a uniform linear weighting of the objectives. It thus implicitly assumes that all tests are of equal importance; while this may not be the case, it is a reasonable assumption in the absence of knowledge about the relative importance of the different tests.

For binary problems, this solution concept may equivalently be defined as the set of candidate solutions that solve the largest possible number of tests; the candidate solutions with the highest expected score are those that solve the largest number of tests. Thus, an alternative definition for $S1$ is:

$$\begin{aligned} S1 \quad &= \quad \{C \in \mathbb{C} | \forall C' \in \mathbb{C} : \\ |\{T \in \mathbb{T} | solves(C,T)\}| \quad &\geq \quad |\{T \in \mathbb{T} | solves(C',T)\}| \end{aligned}$$

Here, a candidate solution C is said to *solve* a test T if it has a positive outcome on the test:

$$solves(C,T) \ \equiv \ G(C,T) > 0$$

1.3. S2: Nash Equilibrium

Game theory provides the solution concept of the Nash equilibrium. A Nash equilibrium specifies a strategy for each player such that no player can profitably deviate given the strategies of the other players.

Formally, let the n classes of individuals in a problem be written as $I_1, I_2, \ldots I_n$, where for a test-based problem we could have for example $I_1 = \mathbb{C}$ and $I_2 = \mathbb{T}$. Let $I = \times_{j \in N} I_j$ where N is the set of indices: $N = \{1, 2, \ldots, n\}$. Given a set of individuals I_i, let $\Delta(I_i)$ denote the set of probability distributions over I_i, and let $\Omega = \times_{j \in N} \Delta(I_j)$. A mixed strategy profile $\alpha \in \Omega$ specifies a probability distribution for each class of individuals. The expected outcome for the i^{th} class of individuals in a mixed strategy profile is written as: $E(G_i(\alpha)) = \sum_{a \in I} \prod_{j \in N} \alpha_j(a_j) G_i(a)$, where $G_i(a)$ returns the outcome for the i^{th} individual. Then a mixed-strategy Nash-equilibrium is a mixed-strategy $\alpha*$, such that:

$$\begin{aligned} S2 = \{\alpha* \in \Omega | \forall i : \forall \alpha_i \in \Delta(I_i) : \\ E(G_i(\alpha*)) \geq \ E(G_i(\alpha*_1, .., \alpha*_{i-1}, \alpha_i, \alpha*_{i+1}, .., \alpha*_N)))) \end{aligned}$$

An attractive feature of the Nash equilibrium as a solution concept is that, while being general, the set of

individuals it represents can be relatively small; this is a valuable property for coevolutionary search. A disadvantage is that there can be (infinitely) many Nash equilibria, part of which may be dominated; thus, finding a Nash equilibrium does not guarantee that the highest possible outcomes are achieved.

1.4. S3: Pareto-Optimal Set

Evolutionary Multi-Objective Optimization extends common evolutionary methods by facilitating the use of multiple *objectives*. This may be viewed as the use of a fitness function that is vector-valued rather than scalar. In Pareto-Coevolution, every possible test is viewed as an objective. A central concept in Evolutionary Multi-Objective Optimization is that of Pareto-dominance. Applied to the current context, a candidate solution C_1 is said to *dominate* another candidate solution C_2 if the following holds:

$$\begin{aligned} dom(C1, C2) \quad &\equiv \quad \forall T \in \mathbb{T} : G(C_1, T) \geq G(C_2, T) \\ &\wedge \quad \exists T \in \mathbb{T} : G(C_1, T) > G(C_2, T) \end{aligned}$$

The solution concept of the Pareto-Optimal Set consists of all candidate solutions that are not dominated by any other solutions:

$$S3 \; = \; \{C \in \mathbb{C} | \nexists C' \in \mathbb{C} : dom(C', C)\}$$

1.5. S4: Pareto-Optimal Equivalence Set

For each maximal combination of tests that can be solved, the Pareto-Optimal set contains all candidate solutions that solve it. Thus, the set may contain many equivalent candidates that each solve the same combination of tests. We define a variant of the Pareto-Optimal set that does not contain such duplicate candidate solutions. This set is defined by the requirement that for each maximal combination of tests that can be solved, it contains at least one candidate solution that solves it. Since multiple such sets may exist, we define $S4$ as the collection of all such sets:

$$\begin{aligned} S4 = \{S \subseteq \mathbb{C} | \forall TS \subseteq \mathbb{T} : \quad &\exists C \in \mathbb{C} : solves(C, TS) \\ \implies \quad &\exists C' \in S : solves(C', TS) \end{aligned}$$

2. Related Work

A number of researchers have investigated the use of coevolution as a problem solving technique (Reynolds, 1994; Miller & Cliff, 1994; Angeline & Pollack, 1994; Schmidhuber, 1999; Pagie & Hogeweg, 1998; Paredis, 1996; Pollack & Blair, 1998; Funes, 2001; Rosin, 1997; Lubberts & Miikkulainen, 2001; Werfel et al., 2000; Moriarty & Miikkulainen, 1998; Potter & De Jong, 2000; Stanley & Miikkulainen, 2004).

While we focus on the application of coevolution to test-based problems here, coevolution can also be used to address problems where a fitness function is given; this form of coevolution is called *Cooperative* or *Compositional Coevolution* (Potter & De Jong, 2000; Watson & Pollack, 2003; Wiegand, 2003; Jansen & Wiegand, 2003)

Recent research in coevolution has benefited from an increased understanding of the adaptive search process, as expressed in the development of several recent theoretical approaches. These include the use of Evolutionary Multi-Objective Optimization concepts to describe Pareto-coevolution (Ficici & Pollack, 2000; Watson & Pollack, 2000; De Jong & Pollack, 2004), order theory (Bucci & Pollack, 2002; Bucci & Pollack, 2003), and game theory (Ficici, 2004; Ficici & Pollack, 2000; Wiegand, 2003; Wiegand et al., 2002).

The focus of this work is on monotonicity. In the following, we therefore discuss existing coevolutionary algorithms with a monotonicity guarantee. In his thesis, Rosin (Rosin, 1997) presents the *covering competitive algorithm*, which guarantees monotonicity for the $S0$ solution concept. Schmitt (Schmitt, 2003) presents a convergence proof for a variant of $S0$ involving more than two types of individuals ('species'), but still requiring the existence of individuals that simultaneously maximize the outcome over all individuals of other types. Ficici has described an algorithm that guarantees monotonicity for $S2$, the Nash equilibrium. Finally, the IPCA algorithm guarantees monotonicity for the solution concepts of the Pareto-Optimal Equivalence Set.

3. The MaxSolve Algorithm

We now consider how an algorithm may be devised that guarantees monotonicity for the solution concept $S1$. The algorithm is called MaxSolve, since it searches for candidate solutions that solve the maximum number of tests. Pseudocode for the algorithm is shown in Figure 1.

In order to guarantee monotonic progress towards the set of candidate solutions solving the largest number of tests, the algorithm compares candidate solutions based on how many of the tests seen by the candidate solution so far they are able to solve. By ensuring that this number can only increase, monotonicity for

```
submit(CSnew, TSnew){
  CA := CA ∪ CSnew;
  TA := TA ∪ TSnew;
  ∀C ∈ CA
    no_solved[C] = number_solved(C, TA);
  ∀C ∈ CA
    ∀C' ∈ CA, C' ≠ C
      if ∀T ∈ TA : G(C, T) = G(C', T)
        no_solved[C'] = 0;
      end
  sort(CA, no_solved[]);
  for i = 1 : archive_size
    if (no_solved[CA[i]] > 0)
      select(CA[i])
    end
  end
  ∀T ∈ TA
    if ∃C ∈ CA : solves(C, T)
      select(T);
    end
  ∀T ∈ TA
    ∀T' ∈ TA, T' ≠ T
      if ∀C ∈ CA : G(C, T) = G(C, T')
        deselect(T);
      end
}
```

Figure 1. Pseudocode for the MaxSolve algorithm.

the solution concept of $S1$ is assured.

The algorithm receives a set of new candidate solutions $CSnew$ and a set of new tests $TSnew$ that are to be considered for placement in the archive. First, it measures for each candidate how many of the tests available so far it solves. Candidates with identical outcomes for all tests are considered superfluous and assigned a zero score, so that they will not be selected.

We note that a candidate discarded in this manner could potentially solve more unseen tests than the candidates that are maintained. This does not violate the monotonicity of the archive however, since progress is measured with respect to the set of tests seen so far. If the rejected candidate solution does indeed solve more tests, this will eventually be discovered given sufficient further search, at which point it will be included.

The next step in the algorithm is to sort the candidate solutions. The *sort* function sorts candidates based on the number of tests they solve. $number\_solved(C, TS)$ returns the number of tests in TS solved by a candidate solution C. Using the sorted ordering, the highest scoring individuals are selected to remain in the archive, with the restriction that selected individuals must solve at least one test, and up to a maximum of n individuals, where n is a parameter specifying the maximum size of the archive.

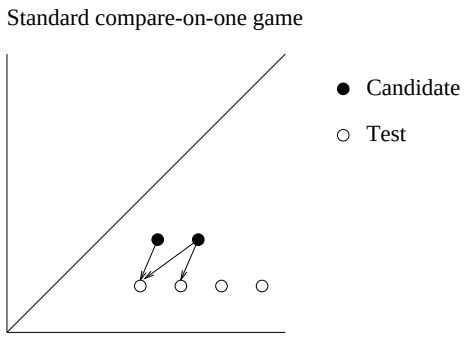


Figure 2. The standard COMPARE-ON-ONE game. Candidates are tested on the highest dimension of the test. Thus, tests below the diagonal test the horizontal dimension of candidates. Comparison of the tests solved by candidates (see arrows) provides a gradient indicating the direction of improvement.

To determine which tests to maintain, all tests solved by one or more of the selected candidates are selected. The algorithm described so far is monotonic. For efficiency, the following procedure is added: of multiple tests with identical outcome vectors only one is maintained.

The following section describes the test problem that will be used in the experiments.

4. The Discretized COMPARE-ON-ONE Problem

One of the issues that complicates the design of reliable coevolution algorithms is the problem of *overspecialization* (Watson & Pollack, 2001; De Jong & Pollack, 2004). For any problem, there can be multiple underlying factors that determine the quality of individuals, and it may well be the case that individuals only progress on one or more of these factors, but fail to do so on others.

The COMPARE-ON-ONE problem (De Jong & Pollack, 2004) is a Numbers Game problem (Watson & Pollack, 2001) that is designed to make over-specialization likely. It does so by letting tests test on a single dimension only. Both candidate solutions and tests are points in an n-dimensional space. A test tests whether a candidate is at least as high as the test in the dimension in which the test is highest:

$$m = \arg\max_i T_i$$

$$G(C,T) = \begin{cases} 1 & \text{if} \quad C_m \geq T_m \\ -1 & \text{otherwise} \end{cases}$$

where C is a candidate, T is a test, and x_i denotes the value of individual x in dimension i.

Due to the definition of the game, a test only evaluates a candidate on a single dimension. Unless special attention is paid to maintaining a diverse set of tests, it is unlikely that the test population will maintain tests for each dimension. This effect is enhanced by the tendency of tests to increase only in the dimension on which they test, thus moving away from the diagonal. This makes it unlikely that a lost dimension will later be regained. If a dimension is lost, individuals can only progress on some of the underlying dimensions, but will be likely to drift in others.

To further increase the difficulty of the problem, a negative *mutation bias* is used to model the property of actual problems that a variation is more likely to cause regress than progress. Thus, unless regress is avoided for all underlying objectives of the problem, the values of individuals are expected to actually *decrease* in the lost dimensions rather than just drift. The need to detect both progress and regress is further increased by applying mutation to multiple dimensions at the same time, as will be discussed.

The COMPARE-ON-ONE problem was developed to induce over-specialization. In order to determine whether overspecialization occurs, performance is measured as the *lowest* value of an individual's dimensions, and not for example the average. Due to this, the decreasing values in a dimension will at some point result in a decreasing performance for the individual, so that over-specialization can be clearly detected. The experiments will employ the 3-dimensional version of the COMPARE-ON-ONE problem.

Since we expect exploration to be a necessary ingredient for real-world coevolutionary problems, a question is how reliability can be combined with exploration. In order to test the ability of coevolution algorithms to perform exploration, we employ a discretized version of the COMPARE-ON-ONE problem.

In the discretized COMPARE-ON-ONE problem (see Figures 2, 3), the value in each dimension of an individual is rounded to the nearest multiple of a parameter δ below it. This discretization is applied to both candidates and tests before evaluating the outcome of the problem, without affecting the genotype.

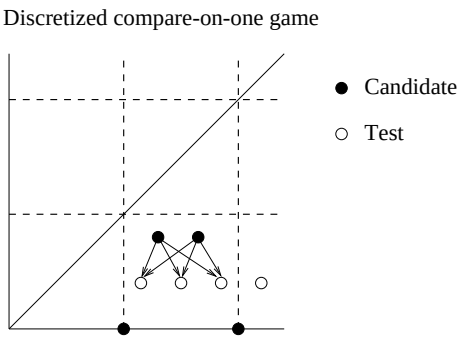


Figure 3. The discretized COMPARE-ON-ONE game. All individuals are mapped to the lower-left corner of their square in the discretization grid. Thus, the candidates solve all tests in their square. As a result, much of the gradient information is lost.

$$m = \arg\max_i d(T_i) \tag{1}$$

$$G(C,T) = \begin{cases} 1 & \text{if} \quad d(C_m) \geq d(T_m) \\ -1 & \text{otherwise} \end{cases} \tag{2}$$

where $d(x) = \delta \lfloor \frac{x}{\delta} \rfloor$

As an example, using $\delta = 0.25$ as in the experiments, the individual $[0.23, 0.30, 0.47]$ would be mapped to $[0, 0.25, 0.25]$ before calculating the outcome of the standard COMPARE-ON-ONE problem. The discretization procedure greatly reduces the amount of gradient present in the problem; individuals have no means to determine whether a value of .45 is better than .25. The mutation range is set such that improvements can only be reached by making *multiple* subsequent steps in the right direction. Addressing the discretized COMPARE-ON-ONE problem therefore requires a substantial amount of random exploration in addition to the difficulties posed by the standard COMPARE-ON-ONE problem.

5. Results

We now investigate the behavior of IPCA and MaxSolve on the discretized COMPARE-ON-ONE problem. Figure 4 shows the performance of the two methods measured as a function of the number of evaluations, where each evaluation computes the outcome of a test for a given candidate. All outcomes are cached, and therefore computed and measured only once. For MaxSolve, archive sizes of 1, 2, 5, 10, and 20 are used.

As the graph shows, the performance of MaxSolve varies with the maximum archive size parameter. For several parameters, it performs better than IPCA. The

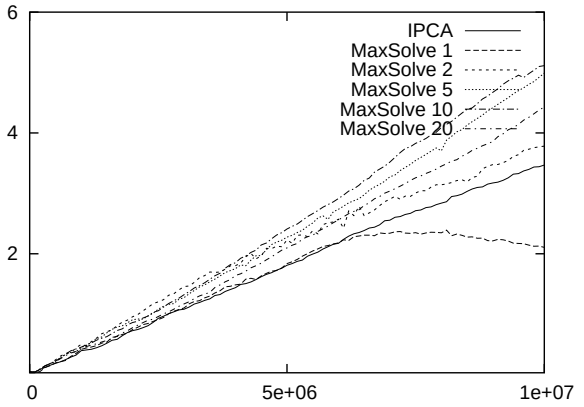


Figure 4. Experimental comparison of IPCA and Maxsolve with archive sizes 1, 2, 5, 10, and 20 on the discretized COMPARE-ON-ONE problem with mutation bias. For several parameter values, MaxSolve performs better than IPCA.

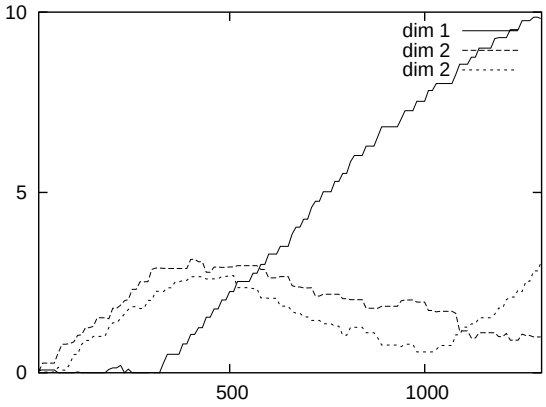


Figure 5. Performance of the best archive member in the three underlying objectives. While each transition is guaranteed to yield improvement in at least one objective, the value of the lowest objective may nonetheless decrease over time.

best performance is obtained using an archive size of 10; the performance in this case is substantially higher than that of IPCA. Thus the MaxSolve algorithm is seen to provide a useful extension to the small but growing collection of reliable algorithms that are available for coevolution.

A first striking observation is that for an archive size of 1, the performance measure at some point begins to decline. This may at first sight seem at odds with the claim that the algorithm guarantees monotonicity. The reason this surprising phenomenon can occur is that the test problem features multiple *underlying objectives* (De Jong & Pollack, 2004), as will now be discussed.

The underlying objectives for this problem correspond precisely to the dimensions of the space in which the individuals reside. Specifically, the performance of each candidate solution is determined by the three coordinates that make up its genotype. The performance measure shown in the graph reflects the *lowest* value for each individual, and shows the highest score for this value over the individuals in the archive, averaged over 30 runs.

While the individuals in the archive are selected so as to solve an incremental number of the tests that have been collected, and thus to improve in *some* underlying objective, this does not imply that the performance level obtained in each underlying objective will be maintained. This is seen when the three genotypic values of the best archive member are tracked over time; see Fig. 5. While each change in the archive improves the performance of some objective, the value of the lowest objective may decrease. As a result of this, the performance measure shown earlier can decrease, as it reflects the performance of individuals in their lowest dimension. If the problem space is finite however, the potential for improvement in each underlying objective will at some point be exhausted, after which progress in other underlying objectives is inevitable given sufficient exploration.

While it is straightforward to achieve the theoretical guarantee that progress will at some point occur, it may take an impractical amount of time before such progress occurs, and accordingly the occurrence of over-specialization does pose a problem to search algorithms in practice. Therefore, it is particularly interesting to see that for larger archive sizes, the MaxSolve algorithm *is* able to avoid over-specialization. Our explanation for this observation is that given a larger number of candidates, the tests solved by the candidate archive form a more diverse set, so that individuals progressing in multiple dimensions can be successfully distinguished from those focusing on a subset of the objectives. While there is no explicit pressure to avoid over-specialization, the maintenance of a more diverse set of tests can apparently be sufficient to facilitate progress in all underlying objectives.

A further observation is that there appears to be a trade-off between the archive size and performance; the best results are obtained for an intermediate archive size. To visualize this trade-off more clearly, we have plotted the performance of MaxSolve after 10^7 function evaluations as a function of the maximum archive size. Figure 6 shows the results, and clearly

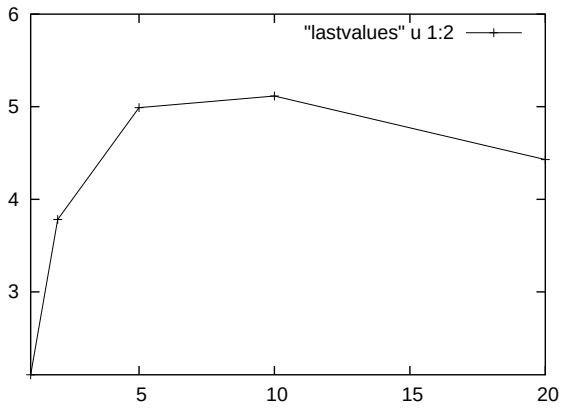


Figure 6. The trade-off between archive size and performance; the best performance is obtained for an intermediate archive size. See text.

shows the trade-off.

6. Conclusions

We have investigated the solution concept of maximizing the expected outcome against a random opponent. This solution concept may equivalently be seen as maximizing the number of defeated opponents or solved tests.

An algorithm that guarantees monotonicity for this solution concept has been presented, named MaxSolve. MaxSolve has been compared to IPCA, which guarantees monotonicity for the solution concept of the Pareto-Optimal Equivalence set. In experiments with a test problem likely to induce over-specialization, MaxSolve was seen to outperform IPCA for intermediate archive sizes. MaxSolve furthermore avoids the limitation of IPCA that the solution concept may involve a large number of individuals, since the algorithm maximizes the number of tests solved and avoids the maintenance of equivalent individuals. Regarding the choice of the archive size, a trade-off was observed, and intermediate values were seen to yield the best results.

A last factor that may limit MaxSolve's application scope is that the test archive grows incrementally. For problems where the number of tests is large but feasible however, the algorithm may in principle provide efficiency improvements compared to evaluating against all tests. Thus, it may be worthwhile to begin applying this theoretically justified coevolution algorithm to problems of practical interest.

References

Angeline, P. J., & Pollack, J. B. (1994). Coevolving high-level representations. *Artificial Life III* (pp. 55–71). Redwood City, CA: Addison-Wesley.

Axelrod, R. (1987). The evolution of strategies in the iterated prisoner's dilemma. *Genetic Algorithms and Simulated Annealing* (pp. 32–41). London: Pitman Publishing.

Barricelli, N. A. (1962). Numerical testing of evolution theories. Part I: Theoretical introduction and basic tests. *Acta Biotheoretica*, *16*, 69–98.

Bucci, A., & Pollack, J. B. (2002). Order-theoretic analysis of coevolution problems: Coevolutionary statics. *Proceedings of the GECCO-02 Workshop on Coevolution: Understanding Coevolution.*

Bucci, A., & Pollack, J. B. (2003). A mathematical framework for the study of coevolution. *Foundations of Genetic Algorithms (FOGA-2002).* San Francisco, CA: Morgan Kaufmann.

De Jong, E. D. (2004a). Guaranteeing progress in pareto-coevolution. *Proceedings of the Annual Machine Learning Conference of Belgium and The Netherlands, BeNeLearn-04* (pp. 22–29).

De Jong, E. D. (2004b). The Incremental Pareto-Coevolution Archive. *Proceedings of the Genetic and Evolutionary Computation Conference, GECCO-04.*

De Jong, E. D., & Pollack, J. B. (2004). Ideal evaluation from coevolution. *Evolutionary Computation*, *12*, 159–192.

Ficici, S. G. (2004). *Solution concepts in coevolutionary algorithms.* Doctoral dissertation, Brandeis University.

Ficici, S. G., & Pollack, J. B. (2000). A game-theoretic approach to the simple coevolutionary algorithm. *Parallel Problem Solving from Nature, PPSN-VI.* Berlin: Springer.

Funes, P. (2001). *Evolution of complexity in real-world domains.* Doctoral dissertation, Brandeis University, Waltham, MA.

Hillis, D. W. (1990). Co-evolving parasites improve simulated evolution in an optimization procedure. *Physica D*, *42*, 228–234.

Jansen, T., & Wiegand, R. P. (2003). Exploring the explorative advantage of the cooperative coevolutionary (1+1) EA. *Genetic and Evolutionary Computation – GECCO-2003* (pp. 310–321). Chicago: Springer-Verlag.

Kauffman, S. A., & Johnsen, S. (1992). Co-evolution to the edge of chaos: Coupled fitness landscapes, poised states, and co-evolutionary avalanches. In C. Langton, C. Taylor, J. Farmer and S. Rasmussen (Eds.), *Artificial life II*, vol. X of *SFI Studies in the Sciences of Complexity*, 325–369. Redwood City, CA: Addison-Wesley.

Koza, J. R. (1992). Genetic evolution and co-evolution of computer programs. In C. Langton, C. Taylor, J. Farmer and S. Rasmussen (Eds.), *Artificial life II*, vol. X of *SFI Studies in the Sciences of Complexity*, 603–629. Redwood City, CA: Addison-Wesley.

Lindgren, K. (1992). Evolutionary phenomena in simple dynamics. In C. Langton, C. Taylor, J. Farmer and S. Rasmussen (Eds.), *Artificial life II*, vol. X of *SFI Studies in the Sciences of Complexity*, 295–312. Redwood City, CA: Addison-Wesley.

Lubberts, A., & Miikkulainen, R. (2001). Co-evolving a go-playing neural network. *Proceedings of the GECCO-01 Workshop on Coevolution: Turning Adaptive Algorithms upon Themselves* (pp. 14–19).

Miller, G., & Cliff, D. (1994). Protean behavior in dynamic games: Arguments for the co-evolution of pursuit-evasion tactics. *From Animals to Animats 3: Proceedings of the Third International Conference on Simulation of Adaptive Behavior, SAB-94* (pp. 411–420). Cambridge, MA: The MIT Press.

Miller, J. (1989). The coevolution of automata in the repeated prisoner's dilemma. Santa Fe Institute working paper 89-003.

Miller, J. (1996). The coevolution of automata in the repeated prisoner's dilemma. *Journal of Economic Behavior and Organization*, *29*, 87–112.

Moriarty, D. E., & Miikkulainen, R. (1998). Forming neural networks through efficient and adaptive coevolution. *Evolutionary Computation*, *5*, 373–399.

Pagie, L., & Hogeweg, P. (1998). Evolutionary consequences of coevolving targets. *Evolutionary Computation*, *5*, 401–418.

Paredis, J. (1996). Coevolutionary computation. *Artificial Life*, *2*.

Pollack, J. B., & Blair, A. D. (1998). Co-evolution in the successful learning of backgammon strategy. *Machine Learning*, *32*, 225–240.

Potter, M. A., & De Jong, K. A. (2000). Cooperative coevolution: An architecture for evolving coadapted subcomponents. *Evolutionary Computation*, *8*, 1–29.

Reynolds, C. W. (1994). Competition, coevolution and the game of tag. *Proceedings of the Fourth International Workshop on the Synthesis and Simulation of Living Systems* (pp. 59–69). Cambridge, MA: The MIT Press.

Rosin, C. D. (1997). *Coevolutionary Search among Adversaries*. Doctoral dissertation, University of California, San Diego, CA.

Schmidhuber, J. (1999). Artificial curiosity based on discovering novel algorithmic predictability through coevolution. *Proceedings of the Congress on Evolutionary Computation, CEC-99* (pp. 1612–1618). Piscataway, NJ: IEEE Press.

Schmitt, L. M. (2003). Theory of coevolutionary genetic algorithms. *Parallel and Distributed Processing and Applications, International Symposium, ISPA 2003* (pp. 285–293). Berlin: Springer.

Stanley, K. O., & Miikkulainen, R. (2004). Competitive coevolution through evolutionary complexification. *Journal of Artificial Intelligence Research*, *21*, 63–100.

Watson, R. A., & Pollack, J. B. (2000). Symbiotic combination as an alternative to sexual recombination in genetic algorithms. *Parallel Problem Solving from Nature, PPSN-VI*. Berlin: Springer.

Watson, R. A., & Pollack, J. B. (2001). Coevolutionary dynamics in a minimal substrate. *Proceedings of the Genetic and Evolutionary Computation Conference, GECCO-01* (pp. 702–709). San Francisco, CA: Morgan Kaufmann.

Watson, R. A., & Pollack, J. B. (2003). A computational model of symbiotic composition in evolutionary transitions. *Biosystems*, *69*, 187–209. Special Issue on Evolvability, ed. Nehaniv.

Werfel, J., Mitchell, M., & Crutchfield, J. P. (2000). Resource sharing and coevolution in evolving cellular automata. *IEEE Transactions on Evolutionary Computation*, *4*, 388.

Wiegand, R. P. (2003). *An analysis of cooperative coevolutionary algorithms*. Doctoral dissertation, George Mason University, Fairfax, Virginia.

Wiegand, R. P., Liles, W., & De Jong, K. (2002). Analyzing cooperative coevolution with evolutionary game theory. *Proceedings of the 2002 Congress on Evolutionary Computation CEC2002* (pp. 1600–1605). IEEE Press.

Assessment of SVM Reliability for Microarrays Data Analysis

Andrea Malossini MALOSSIN@DIT.UNITN.IT
Enrico Blanzieri BLANZIER@DIT.UNITN.IT
Department of Information and Communication Technology, University of Trento, via Sommarive 14, 38050 Povo, Italy

Raymond T. Ng RNG@CS.UBC.CA
Department of Computer Science, University of British Columbia, Vancouver, B. C., V6T 1Z4, Canada

Abstract

The goal of our research is to provide techniques that can assess and validate the results of SVM-based analysis of microarray data. We present preliminary results of the effect of mislabeled training samples. We conducted several systematic experiments on artificial and real medical data using SVMs. We systematically flipped the labels of a fraction of the training data. We show that a relatively small number of mislabeled examples can dramatically decreases the accuracy of the classifier. This phenomenon persists even if the dimensionality of the input space is drastically decreased, by using for example feature selection. Moreover we show that for SVM recursive feature elimination, even a small fraction of mislabeled samples can completely change the resulting set of genes.

1. Introduction

Gene-expression microarrays make it possible to simultaneously measure the rate at which a cell or tissue is *expressing* (translating into a protein) each of its thousands of genes. One can use these comprehensive snapshot of biological activity to infer regulatory pathways in cells, identify novel targets for drug design, and improve the diagnosis, prognosis, and treatment planning for those suffering from disease. The amount of data this new technology produces is more than one can manually analyze. Thus, applying data mining techniques is necessary. However, while data mining techniques are proved successful for business applications (where the number of samples is usually high), gene expression data sets have usually characteristics rather different from those of business data sets. We observe three key issues: high dimensionality, small sample size, and noise.

- The dimensionality of the data, p, can be very high. In the human genome, there are at least 20,000 genes. And in the human body, there are more than a million proteins. Thus, for one patient, there can quite easily be over 50,000 pieces of data.

- The number of samples, n, can be small (relative to typical business applications). For many biomedical and pathology studies, 40-80 patients are considered decent-sized. Sample sizes in the order of hundreds are less common. There are a number of reasons why this is the case. First, data acquisition itself may be very expensive. While microarray costs are decreasing, other costs (e.g., wet laboratory cost for micro-dissection of tissues) remain high. For instance, the cost associated with one patient can very easily exceed 10,000 Euros. Money aside, the second reason is that for many diseases, there are simply not enough patients available. One prime example is early stage lung cancer (e.g., carcinoma-in-situ). Because early stage lung cancer is very hard to detect by normal pathological means (e.g., x-rays), we do not know of any medical research center in the world which has a database of such patients exceeding 100. Finally, the third reason is that even if the patients are there, many of them or their families may not want to participate in research studies.

- Biomedical data can be very noisy. One reason is that data may be acquired in laboratory environment, which sometimes can be hard to keep unchanged. Another reason is that making diagnostic decisions (e.g., grading a biopsy) is not completely objective or black-and-white. For the

same medical condition, there may be different gold-standards, which could lead to different decisions. Thus, robust techniques are very important.

Recently a state-of-the-art classification method, Support Vector Machine (Cortes & Vapnik, 1995) has been used successfully in microarray data analysis (Golub & et al., 1999; Furey & et al., 2000; Valentini, 2002; Lee & Lee, 2003; Simek & et al., 2004). Unfortunately microarray data sets are characterized by the huge dimensionality of the input space p (which comprises thousands of genes) versus the extremely low number n of training samples (usually of the order of tens) as shown in Table 1. In such cases, a small error in the training set could result in a really poor-performance classifier.

The goal of our work is to assess the reliability of the results obtained by SVM techniques on microarray data. Here we present preliminary work that considers mislabeled training samples as a possible source of unreliability.

2. Is the SVM reliable for microarray data sets?

Initially, we started to investigate the problem of mislabeled samples on an artificial dataset and assess the performance of the classifier.

We generated a two-class classification problem with an input space of $p = 2000$ features. The first class, labeled with "-1", is sampled from a multivariate normal distribution with $\boldsymbol{\mu} = 0$ and $\Sigma = 3 \cdot I$, where I id the identity matrix. The second class, labeled with "+1", is distributed as the first class except for 20 features where the component of the mean is $\mu_i = 3$ and $\Sigma = I$. This procedure has been adopted in order to simulate the differential expression of a limited number of genes. Sampling from the distributions described above, we generated a series of training sets with $n = 10, 20, 30, 40, 50, 100, 200$ elements and a test set of 100 elements. Each training set and test set has half of the elements labeled as "+1" and half labeled as "-1", i.e. is balanced. For each training set we trained 5 different SVM classifiers. One on the unmodified training set, and the other 4 on the training set with different percentage of label flipping. We trained the SVM on the training set and then we randomly flipped the labels of a fraction of the training set and trained other SVMs. We used the standard value for the regularization parameter $C = 1$, which measures the trade-off between error and complexity. We performed the flipping on the original training set for percentages of $\{5, 10, 15, 20\}\%$ (the number of flipping has been truncated to an integer). For each classifier we calculated the accuracy on the test set. For each experiment identified by a value of n and a flipping percentage, the entire procedure has been repeated 20 times and mean and variance of each classifier's accuracy has been calculated.

To visualize the performance of each classifier obtained we used a boxplot. In Fig. 1 we show the results of the simulation for different percentage of label flipping for a training set of 20,30,40,50,100,200 elements respectively. We can note that that with only 10% of flipping the difference between the unflipped and flipped accuracy's classifiers is about 0.1 (i.e. 10% difference in accuracy). By incrementing the number of training samples, the variance decreases but there is still a difference in accuracy. In Fig.2 we show two examples of non-linear SVM classifiers, using a polynomial kernel of degree 2 and a radial basis kernel. The effect of the flipping is still there and in it is more accentuate.

The problem of overfitting which arises when the number of features is much greater than the number of training samples can be lowered by reducing the number of features. Some wrapper techniques for feature selection involving SVM have been developed (e.g. the Recursive Feature Elimination (RFE) (Vapnik & et al., 2002), E-RFE (entropy-based feature elimination) (Furlanello & et al., 2003)), which have been used to reduce the number of features as shown in Tab. 1.

In Fig. 3 we show the effect of reducing the number of features from 2000 to 200, but including all the important features which permits to separate the two classes. Notice that even if the number of features is low, $p = 200$, given a low number of samples, the effect on the the accuracy of the mislabeled classifiers is still present despite the classifiers generated have good absolute accuracy.

Since a real dataset is far more complex than the synthetic data we generated, we tested the procedure on a real biological dataset, a human breast cancer dataset from (West & et al., 2001), which included 49 samples, 24 marked as ER+ and 25 marked as ER-. We built randomly two training sets of 20 and 30 elements and test the SVM classifier on a disjoint random test set of 19 samples. In Fig. 4 we show the results on the Breast Cancer dataset using the SVM and randomly flipping a percentage of the original labels. Again with 10 % of flipping the resulting classifier has an lower accuracy (about 0.1) than the unflipped classifier. This means that only 2 or 3 wrong labels suffice to have a sensible degradation of the accuracy. Class prediction is not the only purpose of SVM application in microar-

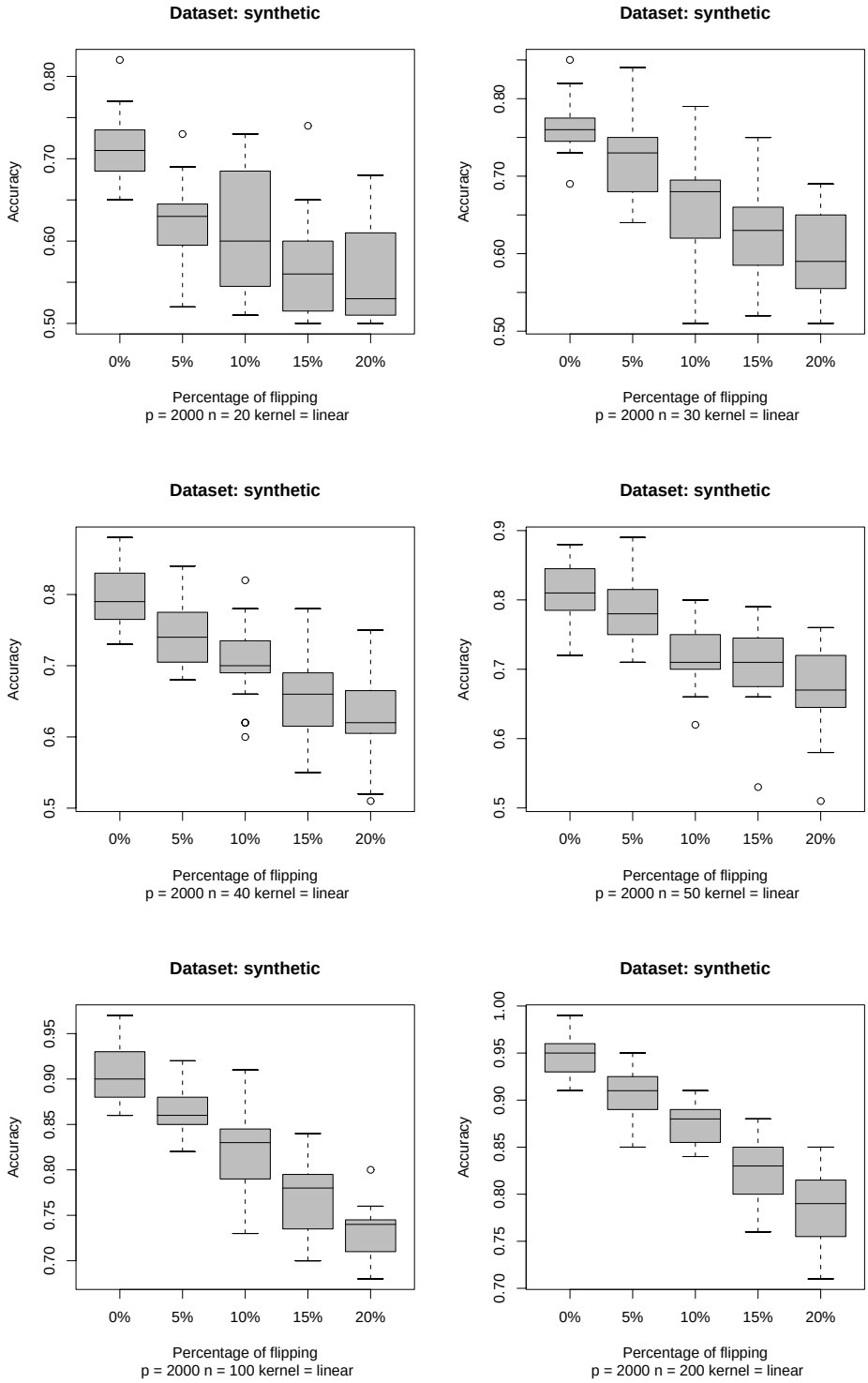


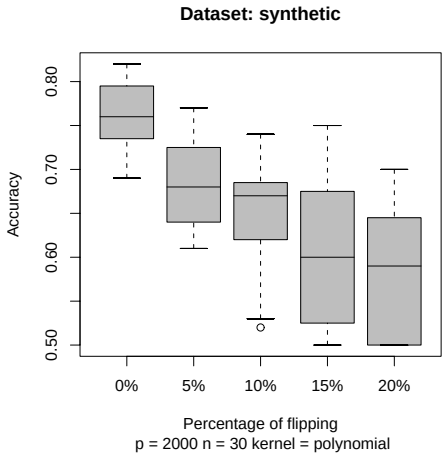
Figure 1. Boxplots of the accuracy of the SVM classifiers. An incremental percentage of random flipping of the labels is performed and the SVM tested on a 100-samples unflipped test set.

Table 1. Number of features p and number of available samples n in microarray data analysis literature ("-" means no feature selection is performed in the paper).

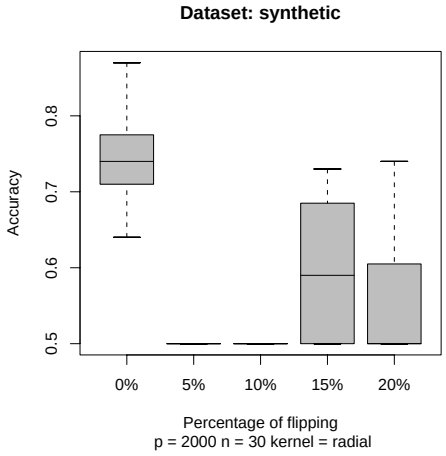
Source	n	p	p after feature selection
West et al. (West & et al., 2001)	49	7129	-
Golub et al. (Golub & et al., 1999)	38	6817	-
Vapnik et al. (Vapnik & et al., 2002)	38	6817	16
Alon et al. (Alon & et al., 1999)	62	2000	-
Alizadeh et al. (Alizadeh & et al., 2000)	96	4026	-
Ramaswamy et al. (Ramaswamy & et al., 2003)	76	16063	-
Furlanello et al. (Furlanello & et al., 2003)	76	16063	315

Table 2. Lists of genes selected by SVM-RFE from the Breast Cancer data set with growing percentage of flipping, respectively 0 %, 5 % and 10 %. The genes indicated with a • are present also in the list of genes selected from the unflipped data set.

Rank	0 % of flipping	5 % of flipping	10 % of flipping
1	X03635_at	U81984_at	L38608_at
2	X55037_s_at	• M23263_at	U39840_at
3	U57650_at	U77665_at	X59131_at
4	M23263_at	X65977_at	M33493_s_at
5	M26311_s_at	M61853_at	U09196_at
6	L43366_at	U67963_at	M13699_at
7	X91220_at	• M62403_s_at	U68019_at
8	U39817_at	X03656_rna1_at	• M23263_at
9	U96113_at	HG742-HT742_at	• D38550_at
10	M62403_s_at	U41060_at	X65977_at
11	U05340_at	• D63485_at	• Z29083_at
12	X58072_at	Z49878_at	X86681_at
13	L20860_at	D26599_at	L77864_at
14	X57351_at	X69636_at	M60614_at
15	D63485_at	• X03635_at	• X58072_at
16	D45906_at	HG3105-HT3281_s_at	L23333_s_at
17	U32907_at	U78180_at	S38953_s_at
18	U46746_s_at	J04056_at	X98260_at
19	D38550_at	X95677_at	M35851_s_at
20	U61232_at	• U62325_at	D38500_at
21	U63455_at	• U57650_at	D79988_at
22	U21931_at	• U05340_at	U57093_at
23	M83652_s_at	D16105_at	X95826_at
24	U27193_at	M29877_at	M64347_at
25	U62325_at	HG3543-HT3739_at	U81984_at
26	U34044_at	U60319_at	D28124_at
27	J03910_rna1_at	D78586_at	X74262_at
28	Z29083_at	L20591_at	• L43366_at
29	X16866_at	X90840_at	X06323_at
30	L38932_at	J03827_at	U03886_at
31	U68385_at	J03242_s_at	D79994_at
32	X63578_rna1_at	• U61232_at	HG3123-HT3299_at

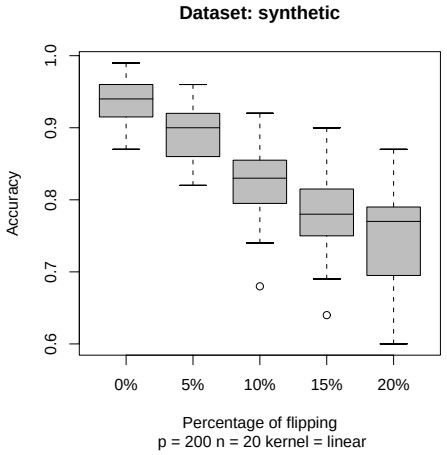


(a) A polynomial kernel of degree 2 is used in the SVM classification.



(b) . A radial basis kernel is used in the SVM classification

Figure 2. Boxplots of SVM classifiers for some non-linear kernels. Artificial dataset of 2000 features and 20 training samples.



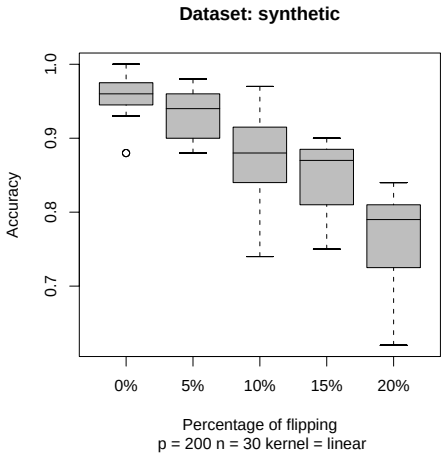


Figure 3. Boxplot of SVM classifiers. An incremental percentage of random flipping of the labels is performed and the SVM tested on a 100-samples unflipped test set. Each experiment is repeated 20 times.

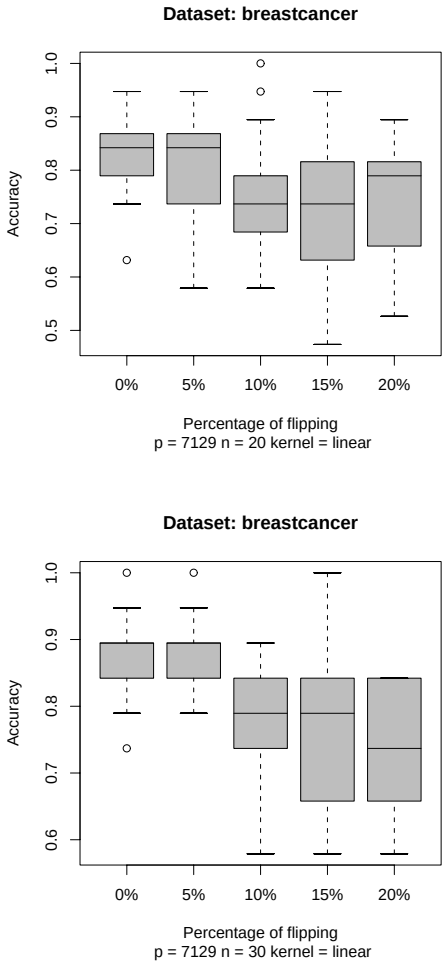


Figure 4. Boxplot of SVM classifiers for the breast cancer dataset. An incremental percentage of random flipping of the labels is performed and the SVM obtained tested on a 19-samples unflipped test set. Each experiment is repeated 20 times.

ray data analysis. In fact SVM is used for discovering of important genes by using feature selection (Golub & et al., 1999; Vapnik & et al., 2002; Furlanello & et al., 2003). In Fig. 3 we reduced the number of features from 2000 to 200 but including all the relevant features. In real data sets we do not know which are the important features, hence mislabeling could also affect the outcomes of a feature selection procedure. If in the training set there are some mislabeled patterns, these misleading information will propagate through the feature selection procedure so we expect, finally, to get a different set of important features (genes). For investigating this effect we tested the recursive feature elimination (Vapnik & et al., 2002) on the Breast Cancer dataset with percentage of flipping of 5 % and 10 %. As shown in Table 2, only 5 genes over 32 are common to the final pool of genes in the unflipped and 10 % flipped cases. Moreover the ranking is completely different. For a 5 % of flipping (in our case about 3 mislabeled samples over 49) there are only 8 common genes. We repeated 20 times the recursive feature elimination on Breast Cancer data by flipping randomly a 10 % of the labels. In Table 3 is shown the genes with higher frequency of presence in the final lists of 32 genes each one. Only 4 genes are present in at least 10 of the selected pools. The others are not in the selected list in the majority of the repetitions. Hence the effect of a 10 % of label flipping on the RFE procedure is to produce a very variable set of selected genes.

In real data set however, the situation is much more complicated than the relative simple situation described in the synthetic data set. Correlation between genes and the much higher variability of the expression values, surely play a crucial role for the sensitivity of SVM to mislabeled samples.

3. Conclusion and future work

We presented the results of experiments on artificial and medical data aimed to assess the sensitivity of SVM classification and feature selection with respect to mislabeled training samples. This is a preliminary step toward the definition of new techniques devoted to evaluate the reliability of the use of SVM for analysis of microarray data. Obviously, scientists should guarantee the quality of the data they use for their research, however, our results show that the robustness of this approaches can be a critical issue. It seems crucial now to take care of this source of error because neither by increasing the number of training samples or decreasing the number of features is a good recipe to increase the performance of a SVM classifier. A statistical/computational method for detecting and solv-

Table 3. Genes that appear with higher frequency in the list of selected genes in 20 repetitions of SVM-RFE from Breast Cancer data set with 10 % of flipping.

Gene	Frequency
X03635_at	18
M23263_at	16
X55037_s_at	16
X58072_at	10
U41060_at	9
U32907_at	9
L43366_at	8
U79293_at	8
U96113_at	8
U62325_at	8
Z29083_at	8
D38550_at	7
U57650_at	7
X16866_at	7
M65062_at	6
U05340_at	6
M26311_s_at	6
M62403_s_at	6
U21931_at	6
D82343_at	5
HG3400-HT3579_at	5
D38437_f_at	5
M26061_at	5
U46746_s_at	5
X65977_at	5
AF000234_at	4
D26135_at	4
D50370_at	4
D63485_at	4
X91220_at	4
X98834_rna1_at	4
(Other)	417

ing such problem should be developed because the microarray analysis based on SVMs is wide-spreading in the scientific community.

References

Alizadeh, A., & et al. (2000). Distinct types of diffuse large b-cell lymphoma identified by gene expression profiling. *Nature*, *403*, 503–511.

Alon, U., & et al. (1999). Broad patterns of gene expression revealed by clustering analysis of tumor and normal colon tissues probed by oligonucleotides array. *PNAS*, *96*, 6745–6750.

Cortes, C., & Vapnik, V. (1995). Support vector networks. *Machine Learning*, *20*, 273˜ –˜ 297.

Furey, T. S., & et al. (2000). Support vector machine classification and validationof cancer tissue samples using microarray expression data. *Bioinformatics*, *16*, 906–914.

Furlanello, C., & et al. (2003). Entropy-based gene ranking without selection bias for the predictive classification of microarray data. *BMC Bioinformatics*, 54–64.

Golub, T. R., & et al. (1999). Molecular classification of cancer: class discovery and class prediction bye gene expression monitoring. *Science*, 531–537.

Lee, Y., & Lee, C.-K. (2003). Classification of multiple cancer types by nulticategory support vector machines using gene expression data. *Bioinformatics*, *19*, 1132–1139.

Ramaswamy, S., & et al. (2003). A molecular signature of metastasis in primary solid tumors. *Nature Genetics*, *33*, 1–6.

Simek, K., & et al. (2004). Using SVD and SVM methods for selection, classification, clustering and modeling of DNA miscoarray data. *Engineering application of Artificial Intelligence*, *17*, 417–427.

Valentini, G. (2002). Gene expression data analysis of human lymphoma using support vector machine and output coding ensembles. *Artifical intelligence in medicine*, *26*, 281–304.

Vapnik, V., & et al. (2002). Gene selection for cancer classification using support vector machine. *Machine Learning*, *46*, 389–422.

West, M., & et al. (2001). Predicting the clinical status of human breast cancer by using gene expression profiles. *PNAS*, *98*, 11462–11467.

Best-response Play in Partially Observable Card Games

Frans Oliehoek FAOLIEHO@SCIENCE.UVA.NL
Matthijs T. J. Spaan MTJSPAAN@SCIENCE.UVA.NL
Nikos Vlassis VLASSIS@SCIENCE.UVA.NL
Informatics Institute, Faculty of Science, University of Amsterdam,
Kruislaan 403, 1098 SJ Amsterdam, The Netherlands

Abstract

We address the problem of how to play optimally against a fixed opponent in a two-player card game with partial information like poker. A game theoretic approach to this problem would specify a pair of stochastic policies that are best-responses to each other, i.e., a Nash equilibrium. Although such a Nash-optimal policy guarantees a lower bound to the attainable payoff against any opponent, it may not necessarily be optimal against a fixed opponent. We show here that if the opponent's policy is fixed (either known or estimated by repeated play), then we can model the problem as a partially observable Markov decision process (POMDP) from the perspective of one agent, and solve it by dynamic programming. In particular, for a large class of card games including poker, the derived POMDP consists of a finite number of belief states and it can be solved exactly. The resulting policy is guaranteed to be optimal even against a Nash-optimal policy. We provide experimental results to support our claims, using a simplified 8-card poker game in which Nash-policies can be computed efficiently.

1. Introduction

A partially observable stochastic game (POSG) is general model that captures the sequential interaction of two or more agents under conditions of uncertainty. This model can be regarded as an extension of a stochastic game (Shapley, 1953), with parts of the state being hidden to at least one agent. It can also be viewed as an extension of a partially observable Markov decision process (POMDP) (Sondik, 1971), with state transitions being influenced by the combined actions of two or more agents. A POSG is also very closely related to the model of an extensive game with imperfect information (Kuhn, 1953).

The literature on POSGs is still relatively sparse. Hespanha and Prandini (2001) showed that a two-player finite-horizon POSG always has a Nash equilibrium in stochastic policies. Koller et al. (1994) demonstrated how to efficiently compute such a Nash equilibrium in the special case of a two-player zero-sum POSG with a tree-like structure, like the card games (e.g., poker) we consider here. Becker et al. (2003), Nair et al. (2003) and Emery-Montemerlo et al. (2004) have developed similar algorithms for computing solutions in the special case of common-interest POSGs. Only recently an algorithm for solving general (albeit still small) POSGs has been proposed (Hansen et al., 2004).

In this paper we consider the class of two-player zero-sum finite-horizon POSGs with a tree-like state structure, that includes many card games like poker. We depart from the game theoretic approach of computing Nash equilibria for these games, and instead deal with the problem of how to compute optimal policies (best-responses) against a fixed opponent. Our interest is motivated by recent suggestions to adopt an 'agent-centric' agenda in multiagent decision making, by which best-response learning algorithms (like Q-learning) replace classical game theoretic approaches to finding an optimal policy (Powers & Shoham, 2005). There are two strong arguments in favor of this approach: first, computing a Nash equilibrium is a difficult problem in general, and efficient algorithms exist only in special cases. Second, a Nash-policy is too conservative in the sense that it will not exploit possible weaknesses of opponents.

As we show below, in order to compute a best-response policy against a fixed opponent in a game like poker, we can model the game as a partially observable Markov decision process (POMDP) by defining a belief state for our protagonist agent. This reduction only requires knowing (e.g., estimating by repeated play) the

stochastic policy of the opponent. The POMDP can be subsequently solved, for instance by dynamic programming, deriving a best-response (deterministic) policy for the agent. In general, an approximate POMDP algorithm may be needed to deal with the continuous belief space (Spaan & Vlassis, 2004). For many card games however, including poker, the particular structure of the problem allows the POMDP to be solved exactly: there are a finite set of reachable beliefs in the POMDP, which allows mapping the POMDP model into a discrete-state MDP and then solve it with an exact dynamic programming method (e.g., value iteration). The resulting policy is optimal (gives the highest possible payoff) against the particular opponent. Moreover we can easily prove that it is no worse than the optimal Nash-policy when playing against a Nash-agent (an opponent that uses a Nash-optimal policy).

To illustrate the method, we have implemented a simplified 8-card poker game for which Nash equilibria can be computed (Koller & Pfeffer, 1997). Then, using the POMDP approach, we have computed best-response policies against Nash-agents. We have experimentally verified that a POMDP best-response policy can indeed reach the optimal Nash payoff when playing against a Nash-optimal agent.

In the following, we first describe our simplified poker game that we use as a running example throughout the paper (Section 2). Then we briefly outline the game-theoretic approach for solving such partially observable games (Section 3). In Section 4 we describe our POMDP-based approach to playing poker against a fixed opponent. We provide experimental results in Section 5, and Section 6 concludes.

2. A simplified poker game

Poker is an example of a stochastic game with partial (imperfect or incomplete) information. In poker a player cannot tell the exact state of the game (e.g., the card deal), and he does not know what policy his opponent will follow. Still, his (sequential) decision making must include comparing potential reward to the risk involved, trying to deceive the opponent by bluffing, dealing with unreliable information from the opponents' actions, and modeling the opponent. All these aspects make poker a very complex game.

There are many poker variants, which all share these properties (Billings et al., 2003). Most of these variants, however, are too large to analyze in an exact fashion because of the number of card combinations and possible betting sequences. Our goal in this paper is not to compute the ultimate strategy for playing general poker, but to address the problem how to play optimally against a fixed opponent. As the opponent strategy we consider can be an optimal Nash policy, we need to be able to compute the Nash policies. To this end, and as running example, we will use a small 8-card poker variant, described by Koller and Pfeffer (1997), which we Nash-solved by using their Gala system.

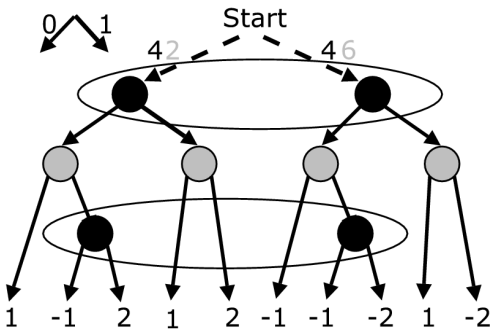


Figure 1. The partial game-tree of 8-card poker for the deals $(4, 2)$ and $(4, 6)$. Gambler's decision nodes are black, dealer's are grey. The payoffs are given for the gambler.

This 8-card poker is played by two players: a dealer and a gambler, who both own two coins. Before the game starts, each player puts one coin to the pot, the ante. Then the dealer deals both players one card out of a deck of eight cards (1 suit, ranks 1–8). After the players have observed their card, they are allowed to bet their remaining coin, starting with the gambler. If the gambler bets his coin, the dealer has the option to fold or call. If the dealer folds he loses the ante, and if he calls showdown follows. If the gambler does not bet, the dealer can choose to bet his coin. If the dealer does so, the gambler will have to decide whether to fold or call. If the game reaches the showdown (neither player bets or the bet of the dealer is called), the player with the highest card wins the pot.

3. Game-theoretic approach

In this section we will briefly outline the game theoretic approach for representing and solving such poker games.

3.1. Poker as an extensive form game

We can model 8-card poker as an *extensive form* game with partial (imperfect) information (Kuhn, 1953). The extensive form of a game is given by a tree, in which nodes represent game states and whose root is the starting state. There are two types of nodes: de-

cision nodes that represent points at which agents can make a move, and chance nodes which represent random transitions by 'nature'. In 8-card poker, the only chance node is the starting state, in which two cards are chosen uniformly at random from the 8-card deck and are dealt to the agents.

In a partial information game, an agent may be uncertain about the true state of the game. In particular, an 8-card poker agent may not be able to discriminate between two nodes in the tree. The nodes that an agent cannot tell apart are grouped in *information sets*. In Fig. 1 a part of the game-tree of 8-card poker is drawn. At the root of tree ('Start' node) a card is dealt to each agent. At each decision node the agents can choose between action 1 (*bet*), and action 0 (*fold*). Fig. 1 shows the partial game tree for two deals: in the first the dealer receives card 2, in the second he receives card 6. The gambler receives card 4 in both cases. Therefore the gambler cannot discriminate between the two deals. This is illustrated by the information sets indicated by ovals. The leaves of the tree represent the outcomes of the game and the corresponding payoffs. In the figure we only show the payoff of the gambler; the payoff of the dealer is exactly the opposite. Games in which payoffs add up to zero, as is the case here, are called *zero-sum* games.

3.2. Solving poker

Solving poker means computing the optimal *policies* the agents should follow. In 8-card poker, a policy for an agent is a mapping from his information sets to actions. A *pure* policy specifies, for each information set, an action that should be taken with probability one. A *stochastic* policy is a probability distribution over pure policies: it specifies, for each information set, an action that should be taken with some specific probability. For example, in the 8-card poker game shown in Fig. 1, a stochastic policy for the gambler could specify that he should bet with probability 0.4 after having received card 4.

The solution of a game specifies how each agent should play *given that the opponent also follows this advise*. As such, it provides an optimal policy for each agent. The solution of a game is given by one or more of its Nash equilibria. Let π_i denote a policy for agent i. A pair of policies $\pi = (\pi_1, \pi_2)$ induce expected payoff $H_i(\pi)$ to agent i, where the expectation is over over the chance nodes in the game; in a zero-sum game $H_1(\pi) = -H_2(\pi)$. When π is a Nash equilibrium, $H_1(\pi)$ is the *value of the game*.

Definition 1. A tuple of policies $\pi = (\pi_1, \pi_2)$ is a Nash equilibrium if and only if it holds that:

$$\begin{aligned} &\forall_{\pi_1'} \left(H_1(\pi_1, \pi_2) \geq H_1(\pi_1', \pi_2)\right) \wedge \\ &\forall_{\pi_2'} \left(H_2(\pi_1, \pi_2) \geq H_2(\pi_1, \pi_2')\right) \end{aligned} \tag{1}$$

That is, for each agent i, playing π_i gives an equal or higher expected payoff than playing π_i'. So both policies are best responses to each other.

In two-player zero-sum games, a Nash policy is a security policy and the value of the game is the security value for an agent. The latter means that the expected payoff of an agent who plays a Nash policy cannot be lower than the value of the game. In other words, a Nash policy gives the payoff that an agent can maximally guarantee for himself, given that the opponent will act in a best-response manner in order to minimize this payoff.

Nash (1951) and Kuhn (1953) have shown that any extensive-form game with perfect recall[1] has at least one Nash equilibrium in stochastic policies. Moreover, in two-player zero-sum games, all Nash equilibria specify the same value for the game (Osborne & Rubinstein, 1994). Therefore, *any* Nash policy is optimal against a best-response opponent, in the sense that it guarantees the security value of the game.

3.3. Computing Nash equilibria

To find the Nash equilibria of an extensive form game, the game can be first transformed into its *normal form*. This is a matrix representation of all pure policies available to the agents. Entry (i, j) of the matrix gives the expected payoff of agent 1's policy i versus agent 2's policy j. Consequently, when converting from the extensive to the normal form, the tree-like structure of the game is discarded. Moreover, the normal form representation of an extensive game can be very large. Note that every pure policy is a mapping from information sets to actions, therefore the number of pure policies is exponential in the size of the game tree.

To compute the Nash equilibria from the normal form we can use linear programming. The normal form of a two-player zero-sum game defines a linear program whose solutions are the Nash-equilibria of the game. However, transforming an extensive form game to its normal form results in very large games, making such an approach for computing Nash equilibria impractical.

Koller and Pfeffer (1997) proposed the 'Gala' system, which solves games in an alternative representation

[1] Perfect recall implies that an agent remembers all actions that he has taken in the past.

called the *sequence* form, whose size is polynomial with the number of nodes in the game tree. This allows one to solve larger games, but real-life games are typically still too large to solve. For example consider Texas Hold-em poker[2], whose game-tree contains $O(10^{18})$ nodes (Billings et al., 2003) and therefore computing Nash equilibria is computationally infeasible.

4. Playing against a fixed opponent

In this section we depart from the game-theoretic approach, and address the problem of how to play optimally against a fixed opponent. It turns out that, if the (stochastic) policy of the opponent agent j can be summarized by a model in the form $p(a_j|s, a_i)$, where a_j denotes his action at some state s and a_i our action, then we can model the poker game as a partially observable Markov decision process (POMDP) *from the perspective* of our protagonist agent i, and compute an optimal (deterministic) policy for it.

According to our POMDP model, at any time step the game is in a state $s \in S$, where the state space S contains all chance nodes in the game-tree, as well as all nodes in which our protagonist agent i takes a decision (the black nodes in Fig. 1 if agent i is the gambler). The game starts at the 'Start' state (chance node). At any other state (decision-node) s, our protagonist agent i takes an action $a_i \in A_i = \{bet, fold\}$, and the opponent takes an action a_j according to a stochastic policy $\pi_j = p(a_j|s, a_i)$. Then the game switches to a new state s' as a result of the two actions (a_1, a_2) and according to a stochastic joint transition model $p(s'|s, a_i, a_j)$. Using the policy of the opponent, we can compute a single transition model for our agent i as follows:

$$p(s'|s, a_i) = \sum_{a_j} p(s'|s, a_i, a_j)p(a_j|s, a_i). \quad (2)$$

This allows us to treat the game as a POMDP from the perspective of our protagonist agent i. In our 8-card poker, the joint transition model $p(s'|s, a_i, a_j)$ is stochastic only in the 'Start' state and deterministic elsewhere (given the actions of the two agents), since there are no other chance nodes in the game-tree. However, as we assume a stochastic policy for agent j, from the viewpoint of agent i the game is stochastic in every state.

In partial information games the agents cannot directly observe the true state of the game, but they receive observations (clues) about the state. In our POMDP model of the game, in each state s our protagonist agent i perceives an observation $o_i \in O_i$ that is related to the current state s and his last action a_i through a stochastic observation model $p(o_i|s, a_i)$. The first observation o_i agent i receives indicates the hand that is dealt to it, consecutive observations signal the last action a_j of the opponent. Moreover, the observation model is deterministic: after the cards are dealt an agent observes $o_i \in \{1, \ldots, 8\}$ with probability one, and in consecutive states he perceives $o_i = a_j$ with full certainty.

At every time step an agent k receives an individual scalar reward signal $r_k(s, a_i, a_j, s')$ based on the previous game state s, current state s' and the joint action (a_i, a_j). Using the policy of the opponent agent j we can compute the reward our agent i receives as follows:

$$r_i(s, a_i, s') = \sum_{a_j} r_i(s, a_i, a_j, s')p(a_j|s, a_i). \quad (3)$$

In poker the reward is 0 except for transitions into one of the end-states, i.e., when one of the agents has folded or the game reaches showdown (Section 2).

As all sets S, O_i, and A_i are discrete and finite in a poker game, we can convert the discrete POMDP model in a continuous belief-state Markov decision process (MDP) (Sondik, 1971), in which the agent summarizes all information about its past using a *belief* vector $b(s)$. The belief b is a probability distribution over S, and grants the agent perfect recall. Our agent i starts with an initial belief b_0, which in our poker setting is set to a Dirac distribution on the 'Start' state. Every time agent i takes an action a_i and observes o_i, it's belief is updated by Bayes' rule:

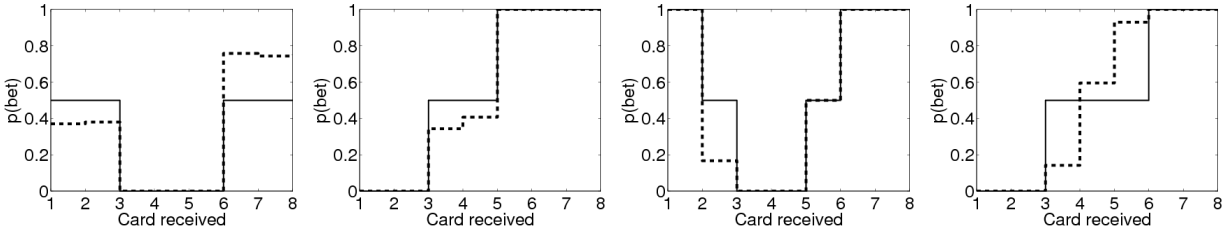
$$b^{a_i}_{o_i}(s') = \frac{p(o_i|s', a_i)}{p(o_i|a_i, b)} \sum_{s \in S} p(s'|s, a_i)b(s), \text{ where} \quad (4)$$

$$p(o_i|a_i, b) = \sum_{s' \in S} p(o_i|s', a_i) \sum_{s \in S} p(s'|s, a_i)b(s) \quad (5)$$

is a normalizing constant. To solve the belief-state MDP in general a large range of POMDP solution techniques can be applied, including exact (Sondik, 1971) or approximate ones (Spaan & Vlassis, 2004).

It turns out, however, that in the class of two-player card games that we are considering only a relatively small *finite* set of beliefs B can ever be experienced by the agent. The set is finite as the problem has a finite horizon, and small because the horizon is low and the sets O_i and A_i are small. Furthermore, after the card dealing, in each state only one of two observations is possible (the action of the opponent agent), reducing the branching factor of the tree of beliefs. In partic-

[2]Texas Hold-em is a popular poker variant in which each player gets two private cards and share 5 public cards.



(a) Gambler policy at start of game. (b) Gambler policy after seeing a bet. (c) Dealer policy after seeing a pass. (d) Dealer policy after seeing a bet.

Figure 2. POMDP and Nash policies for 8-card poker. Dashed lines indicate a Nash policy computed by Gala and solid lines the best-response POMDP policy against that Nash policy, computed by solving the belief-state MDP. The x-axis denotes the card dealt to the gambler ((a),(b)) or the dealer ((c),(d)), and the y-axis indicates the probability of betting.

ular, each information set of agent i in the game-tree will induce a single belief.

Given a finite set B we can compute a finite belief-state MDP from the continuous belief-state MDP, by taking each $b \in B$ as a possible state. Computing a transition model $p(b'|b, a_i)$ specifying how our agent i switches from a particular belief b to another belief b' when taking action a_i is straightforward:

$$p(b'|b, a_i) = p(o_i|a_i, b), \tag{6}$$

where $p(o_i|a_i, b)$ is giving by (5). The reward $r_i(b, a_i, b')$ is defined as:

$$r_i(b, a_i, b') = \sum_{s,s'} r_i(s, a_i, s')b(s)b'(s'). \tag{7}$$

This MDP can now be solved in an exact fashion using standard dynamic programming techniques, for instance using value iteration (Sutton & Barto, 1998). The result is a deterministic best-response policy π_i for our protagonist agent, that maps beliefs it encounters to optimal actions. Value iteration allows us to compute the value of the initial belief state, which equals the expected reward of the game when agent i follows π_i and agent j behaves according to π_j.

Note that the derived best-response policy for the protagonist agent is a deterministic one: we know that the optimal policy of a POMDP is always deterministic (Puterman, 1994). Although this may seem a limiting factor, we can use the following result from game theory: a stochastic policy is a best-response policy against some opponent if and only if all the deterministic policies to which it assigns nonzero probability are also best-response policies to this opponent. Our POMDP derived policies are best-response policies because our agent maximizes his payoff *exactly* over the space of his deterministic policies, and therefore they *must* be in the support of a best-response stochastic policy. In other words, a deterministic POMDP policy is equally good to any best-response stochastic policy in terms of achieved payoff.

5. Experiments

We will now present an experiment performed in the 8-card poker setting, in which we solved 8-card poker as described in Section 2 using the Gala system. The results from this were a pair of optimal Nash policies and the value of the game, which is $+0.0625$ coins per game in favor of the dealer. The next step was to create a POMDP model for the gambler, using the found optimal Nash policy for the dealer by incorporating the dealer's policy in the POMDP transition model, as described in Section 4. This was also done with the roles reversed. From this POMDP model a finite belief-state MDP was extracted which we solved using value iteration. To construct the policy, the action with the highest expected payoff for a belief was selected. When for a certain belief the expected values for both actions are equal, these actions are taken with equal probability.

The resulting policies are shown in Fig. 2. The expected payoff for the POMDP policies is equal to the value attained by the optimal policies ($+0.0625$ for the dealer and -0.0625 for the gambler) and as these are the best payoffs obtainable, the policies are clearly best-response policies. Furthermore, we see that the computed POMDP policies are quite similar to the Nash policies. In particular, betting with probability 1 or 0 happens in exactly the same situations as in the Nash policies. However there are situations in which the two policies differ: the cases in which the POMDP policy is indifferent between both actions and which are assigned probability 0.5.

6. Conclusions

In this paper we addressed the problem of computing a best-response policy for an agent playing against a fixed opponent in a partially observable card game like poker. In such a card game an agent only receives partial information regarding the true state of the game, i.e., the cards dealt to each agent. An agent can only observe its own hand and the action the other agent has executed. A second source of uncertainty is the unknown policy of the other agent. A game-theoretic approach to solving such games would be to compute a pair of stochastic policies that are best-responses to each other, i.e., a Nash equilibrium. Unfortunately, computing Nash equilibria is a difficult problem in general and such a Nash policy is secure but conservative: it will not exploit possible weaknesses of an opponent.

However, when we assume the opponent agent has a fixed policy (known or estimated by repeated play), we can model the game as partially observable Markov decision process (POMDP) from the perspective of our protagonist agent. We have shown that by solving the resulting POMDP model we can compute a deterministic best-response policy for our agent. We focused on a simplified 8-card poker game in which Nash equilibria can be computed. We have argued and experimentally verified that the computed POMDP best-response policy can indeed reach the optimal Nash payoff when playing against a Nash-optimal agent. Avenues of future research include investigating more compact state representations, tackling larger poker variations and considering more general partially observable stochastic games.

References

Becker, R., Zilberstein, S., Lesser, V., & Goldman, C. V. (2003). Transition-independent decentralized Markov decision processes. *Proc. of Int. Joint Conference on Autonomous Agents and Multi Agent Systems*.

Billings, D., Burch, N., Davidson, A., Holte, R., Schaeffer, J., Schauenberg, T., & Szafron, D. (2003). Approximating game-theoretic optimal strategies for full-scale poker. *Proc. Int. Joint Conf. on Artificial Intelligence*. Acapulco, Mexico.

Emery-Montemerlo, R., Gordon, G., Schneider, J., & Thrun, S. (2004). Approximate solutions for partially observable stochastic games with common payoffs. *Proc. of Int. Joint Conference on Autonomous Agents and Multi Agent Systems*.

Hansen, E., Bernstein, D., & Zilberstein, S. (2004). Dynamic programming for partially observable stochastic games. *Proc. 19th National Conf. on Artificial Intelligence (AAAI-04)*. San Jose.

Hespanha, J., & Prandini, M. (2001). Nash equilibria in partial-information games on Markov chains. *Proc. of the 40th Conf. on Decision and Control*.

Koller, D., Megiddo, N., & von Stengel, B. (1994). Fast algorithms for finding randomized strategies in game trees. *Proc. of the 26th ACM Symposium on Theory of Computing (STOC)* (pp. 750–759).

Koller, D., & Pfeffer, A. (1997). Representations and solutions for game-theoretic problems. *Artificial Intelligence*, *94*, 167–215.

Kuhn, H. (1953). Extensive games and the problem of information. *Annals of Mathematics Studies*, *28*, 193–216.

Nair, R., Tambe, M., Yokoo, M., Pynadath, D., & Marsella, S. (2003). Taming decentralized POMDPs: Towards efficient policy computation for multiagent settings. *Proc. Int. Joint Conf. on Artificial Intelligence*. Acapulco, Mexico.

Nash, J. F. (1951). Non-cooperative games. *Annals of Mathematics*, *54*, 286–295.

Osborne, M. J., & Rubinstein, A. (1994). *A course in game theory*. MIT Press.

Powers, R., & Shoham, Y. (2005). New criteria and a new algorithm for learning in multi-agent systems. In L. K. Saul, Y. Weiss and L. Bottou (Eds.), *Advances in neural information processing systems 17*. Cambridge, MA: MIT Press.

Puterman, M. L. (1994). *Markov decision processes—discrete stochastic dynamic programming*. New York, NY: John Wiley & Sons, Inc.

Shapley, L. (1953). Stochastic games. *Proceedings of the National Academy of Sciences*, *39*, 1095–1100.

Sondik, E. J. (1971). *The optimal control of partially observable Markov decision processes*. Doctoral dissertation, Stanford University.

Spaan, M. T. J., & Vlassis, N. (2004). *Perseus: randomized point-based value iteration for POMDPs* (Technical Report IAS-UVA-04-02). Informatics Institute, University of Amsterdam.

Sutton, R. S., & Barto, A. G. (1998). *Reinforcement learning: An introduction*. Cambridge, MA: MIT Press.

Detecting Deviation in Multinomially Distributed Data

Jan Peter Patist JPP@FEW.VU.NL
Department of Artificial Intelligence, Mathematics and Computer Science, Vrije Universiteit, Computational Intelligence, 1081 HV, Amsterdam, the Netherlands

Abstract

Multinomial models are used in describing the distribution of categorial or discrete variables. In practice we are often interested whether a given sample deviates significantly from a certain multinomial distribution. To determine this, often Pearson's χ^2 or likelihood ratio tests are used. Besides these common tests other tests are possible.
Cressie and Read (Cressie & Read, 1984) proposed a family of Power Divergence test statistics of which these most popular tests are members. These tests have different power given different alternative hypotheses. In this paper different characteristics of the tests are shown. These are: the unimodal shape of the power of the tests as a function of λ, the accuracy of the χ^2 approximation, and the dependency of the power of the test on sample size and different null- and alternative hypotheses. There is a large variety of tests which differ in power under different circumstances. Selecting the right test can increase the performance in detecting deviation. Because for small samples the χ^2 approximation of the test statistic is not applicable, and because the power of the tests depends on many parameters, simulation can be used to determine a suitable test with higher power.

1. Introduction

The multinomial distribution is commonly used in building models describing distributions of categorical data. For example: modelling buying behavior in retail, where the parameters of the multinomial model represent the probability of buying a certain product. Multinomial distributions are also used in discretizing continuous variables into bins or categories. Here, the parameters of multinomial distribution represent the expected amount of values relative to the total amount of values. The distribution of the amount of cases falling in bins can be compared with the expected number of cases falling in bins given some hypothesized distribution of a continuous variable. It can then be tested whether the continuous variable follows some hypothesized distribution. In the last two examples the problem is determining whether some observed counts of the categories are generated from the hypothesized distribution or not. When this hypothesis is correct the expected counts of the categories and the observed categories are expected to be more similar than when it is not correct. In (Ye et al., 2000) the χ^2 test is used to detect intrusions in computer networks. It is very important to be equipped with powerful methods to detect deviation in case of the above examples. Suppose, for example, that the buying behavior of clients is modelled over some time interval. Being able to detect change in this behaviour faster (or at all) can help in acting upon these changes. More powerful methods need less data to detect deviation and thus enable us to react faster upon these changes. Detecting deviation is done by scoring samples or observations and comparing them with what is expected. Comparing what is expected with what is observed is done by hypothesis testing. There are many different scoring functions. Popular ones are the Pearson's χ^2 test and the log likelihood ratio. In (Chapman, 1976) these two tests are compared for low dimensionality and a small amount of counts. Cressie and Read (Cressie & Read, 1984) proposed a family of these scoring functions, in which these popular ones are incorporated. The power of these tests are different under different circumstances. In this paper we give an overview of these tests in respect to their characteristics. In practice it is difficult to decide which test to choose - test's performance depends on many parameters such as: sample size, data dimensionality, distribution of parameters that specify the underlying multinomial, etc. Therefore, our paper may be viewed as a contribution towards an experimental framework that supports the process of selecting the most suitable tests by simulations. In other words, to choose a suitable test one may try several setups and select the

best one. All the tests presented in this paper have been implemented in Matlab.

1.1. The multinomial distribution

Assume a stochastic vector $X_{(1)}, .., X_{(m)}$ i.i.d. and $X_{(i)}$ is multinomially distributed. The multinomial distribution is defined by:

$$p_\pi(x_1, .., x_d \mid N) = \binom{N}{x_1, \cdots, x_d} \prod_{i=1}^{d} \pi_i^{x_i}$$

Where π_i denotes the probability that category i will have a success in a Bernoulli experiment. p_π is the probability distribution of the vector of the stochastic variables $X = (x_1, ..x_d)$ of dimension d, representing counts per category. The sum over the counts is equal to N. $X_{(m)}$ resembles a sample of m stochastic vectors. The subscript π tells us that the distribution is fixed by $\pi = (\pi_1 \dots \pi_d)$, where $\sum_{i=1}^{d} \pi_i = 1$, $0 \leq \pi_i \leq 1$, and $\sum_{i=1}^{d} x_i = N$.

1.2. Deviation detection and statistical tests

To determine whether a sample is likely to deviate from the null hypothesis the sample is scored according to a scoring function. The variable representing the score is called a test statistic whenever its value is only dependent on the data. The test statistic has a fixed distribution under the null distribution. Based on some criteria on the score of sample the sample is flagged as deviated. The criteria are completely specified in statistical hypothesis testing. In section 2 we elaborate more on this. The reason to use a test statistic this is either that the test statistic possesses some nice properties, such as following some known distribution, or that it is unfeasible to do hypothesis testing based on the distribution parameters themselves. Testing samples against some distribution is called goodness-of-fit testing.
Most popular test statistics for the multinomial model are the Pearson χ^2 and the likelihood ratio test statistic. These test statistics have an approximate χ^2 distribution under the null hypothesis. Besides these tests, other tests are possible. Cressie and Read (Cressie & Read, 1984) proposed a family of test statistics called the family of Power Divergence test statistics, where the above mentioned tests are members. Outside of this family, other tests are also possible. For an overview of statistical testing see (Lehman, 1997) and for a more extensive overview of goodness-of-fit-tests see (Agostino & Stephens, 1986).

1.3. Power Divergence test statistic

Cressie and Read proposed a unified family of test statistics. The Likelihood Ratio and the Pearson χ^2 tests are members of this family. Besides these well known tests, Kullback-Leibler divergence, the Hellinger distance, and the modified Neyman test statistic are also part of this family. In Cressie and Read it is proven that these tests are all χ^2 distributed when the total number of counts $N \to \infty$.

1.4. Practice

When the number of samples or the number of counts are small, the χ^2 approximation of the distribution of the test statistics may not be applicable. Then the possible consequences of using the approximate χ^2 are either to be unnecessarily conservative or unnecessarily progressive in rejecting samples. An other possibility is to simulate the distribution under the null hypothesis. This empirical distribution will converge to the real distribution in case the number of samples is sufficiently large. Being equipped with these tests and a mechanism to choose between them would be important when it is clear that under different circumstances different tests are needed. A test is "best" whenever the test statistic has the highest expected detection rate against all alternative hypotheses.

1.5. Overview

In the following section we elaborate on the aforementioned topics. First we explain statistical hypothesis testing and its criteria, followed by the definition of the family of Power Divergence test statistics and insight in the differences between its members. Next, several dependencies and characteristics of the Power Divergence statistics are illuminated by several experiments. These characteristics are the dependencies on the sample size, the null- and alternative hypothesis and the χ^2 approximation. After this we discuss the influence of these characteristics on the power of the tests and how we should proceed in practice. Of course the practical problem is how to choose between the tests.

2. Statistical Hypothesis Testing

Let us assume a sample $X = (x_1, \dots, x_d)$ i.i.d. according to a multinomial distribution as defined earlier. The goal is to define criteria which help us in determining whether the sample is likely generated from a hypothesized distribution.

2.1. The null and alternative hypothesis

In hypothesis testing two hypotheses are specified called the null hypothesis and the alternative hypothesis. The null hypothesis describes the case of the data being distributed according the distribution. The alternative hypothesis is either its complement or a subset of the complement. The alternative hypothesis can be seen as the hypothesis we wish to prove. So in order to determine whether a sample X is from $P := (p_{i0})$ a test is set up.

In the case where P as well as the alternative distributions are multinomial distributions, the null-hypothesis and alternative hypotheses are:

$$\begin{aligned} H_0 &: \vec{\pi} = \vec{\pi_0} \equiv \forall \pi_i : (\pi_i = \pi_{0,i}) \\ H_1 &: \vec{\pi} \neq \vec{\pi_0} \equiv \exists \pi_i : (\pi_i \neq \pi_{0,i}) \end{aligned}$$

The null hypothesis states that all π_i, $\pi_{0,i}$ are equal and the alternative hypothesis the opposite, namely that at least one $(\pi_i,\ \pi_{0,i})$-tuple is not equal. The null-hypothesis states that X is from $P := (p_{i0})$, and the alternative hypotheses defines all other hypotheses or subset of them.

2.2. Test criteria

The possible test outcomes are: 1 sample X is rejected or 2 sample X is not rejected. By "sample X is rejected" we mean that sample X is likely not generated by distribution P. Because samples generated from the null hypothesis are possible under the alternative hypothesis and vice versa, mistakes are possible while making a decision in favor of (1) or (2). Although the probability of X given P can be very small it is still possible that X was generated by P. Choosing in favour of (2) in the case X was generated by P is called error of type 1, or false alarm. Choosing (1) in the case were X was not generated by P is called an error of type 2. The best test should be the one which minimizes both errors.
However, there is a conflict if we want to minimize both the errors. The error of type 1 can be minimized by never rejecting samples. But this would clearly increase the error of type 2. On the other hand, rejecting every sample would minimize error of type 2, but would increase the error of type 1 to its maximum. Making an error of type 1 is less severe than making an error of type 2. This is because not rejecting X can be interpreted as a lack of evidence against the null hypothesis. The best test is then defined as the test, which gives at most an expected error of type 1 and gives the lowest error of type 2. Usually there is no best test for all alternative hypotheses because some tests are more powerful than others for different alternative hypotheses. a given sufficient statistic (that depends only on the data) is exceeded. This threshold is set such that the probability of exceeding it is smaller than a predefined α, and the probability of exceeding this threshold given a sample generated from an alternative hypothesis is maximized. In the tests discussed in this article the thresholds are set to the $(1-\alpha)$-quantile of the distribution of the test statistic. Clearly this threshold corresponds with the demand that the error of type 1 is under the α-level and that simultaneously the power under the alternative hypothesis is maximized.

3. Power Divergence test Statistic

Popular choices of the goodness-of-fit tests for multinomial models are the Pearson's χ^2 and the log likelihood ratio test. Besides these two tests other tests are possible. In (Cressie & Read, 1984), Cressie and Read proposed a unified approach to goodness-of-fit-tests for multinomial models, namely through the family of Power Divergence statistics. Using this approach the above mentioned tests e.g. the Pearson's χ^2 and the likelihood ratio test are members of the family. The family of Power Divergence statistics is defined by:

$$\begin{aligned} &2NI^{\lambda}(p,\pi_0), \qquad \text{where} \\ &I^{\lambda}(p,\pi_0) \qquad = \qquad \sum_{i=1}^{d} \frac{p_i}{\lambda(\lambda+1)}\{\left(\frac{p_i}{\pi_{0,i}}\right)^{\lambda} - 1\} \\ &\qquad\qquad\qquad\qquad + \frac{\pi_{0,i} - p_i}{\lambda+1}, \lambda \epsilon \mathcal{R} \end{aligned}$$

where N is the sum of the counts of d categories, $p = (p_1, \ldots, p_d)$ is the vector of the relative observed frequencies of category i and $\pi = (\pi_1, \ldots, \pi_d)$ is the probability vector of the multinomial distribution. Members of the family are distinguished by λ. Known statistics that are member of the family are: Pearson's χ^2, the likelihood ratio ,the Hellinger distance, Kullback-Leibler divergence and the Neyman modified χ^2 test statistic correspond with $\lambda = 1, 0, -1/2, -1, -2$, respectively. The family can also be interpreted as a weighted function G of disparities δ:

$$2NI^{\lambda} \equiv 2N \sum_{i=1}^{d} G(\delta_i)\pi_i, \quad \text{where} \tag{1}$$

$$\delta_i = (\pi_i^{-1} p_i - 1), \quad \text{and} \tag{2}$$

$$G(\delta) = \frac{(\delta+1)^{(\lambda+1)} - (\delta+1)}{\lambda(\lambda+1)} - \frac{\delta}{\lambda+1} \tag{3}$$

G is a function of disparities δ, where δ_i is the difference between the observed and expected frequency divided by the expected frequency.

In Figure 1 we show the function $G(\delta)$ for various λ. In the figure we can see that for most λ the function $G(\cdot)$ is not symmetric round $\delta = 0$. For larger positive values of δ, positive λ give higher G scores and vice versa. Statistics defined by positive λ are more sensitive to outliers (when observed frequency is larger than the expected frequency) and negative λ more to inliers. However, outliers in some category (π_j) will lead to inliers in other π_i, this is because $\sum \pi_i = 1$. The most powerful test, with respect to the change, is the test with the highest probability of rejecting a sample from the alternative hypothesis or changed distribution. It is proven (Cressie & Read, 1984) that for all λ, $2NI^{\lambda}$ is asymptotically χ^2 distributed. Later we will show that for small N the χ^2 distribution is not applicable. In some situations it can easily be seen which λ represents the best test. For example, when under the null hypothesis the categories of the distribution are equilly probable and the alternative hypothesis is equal to the change of a single category upwards and the remaining categories equilly downwards or vice versa. A test equally sensitive in all directions can be obtained by taking the likelihood.
Some tests are extremely sensitive towards zero-cells. A zero-cell is a category for which the observed count in the sample is equal to zero. This follows from the G function. To deal with this it is possible to add a penalty term for the zero cells and not summing over the original partial score of the zero cells. As proposed in (Basu et al., 2002) the penalty term is defined by:

$$\text{penalty} = h \sum_{p_i=0} \pi_i$$

In the case $h = \frac{1}{\lambda+1}$ the penalized G function equals the original G function. Of course Power Divergence statistics extended with the penalty term are asymptotically χ^2 distributed when $N \to \infty$ because the expectation of the zero-cells goes to zero.

4. Methodology

The main goal of the experiments is to describe the characteristics of the above described tests. The characteristics of interest are: the influence of the total number of counts or items on the detection rate, and of the χ^2 approximation. The detection rate of the tests given different null- and alternative hypotheses. Also the Power Divergence statistic extended with the penalty term is tested, because some tests are oversensitive towards empty cells. Adding this penalty term can increase the power. This results in the following experiments:

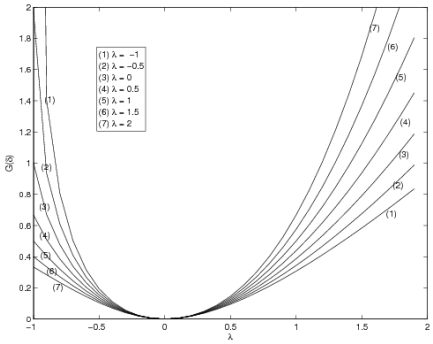


Figure 1. The values of the G function is displayed for different λ. The different values of λ are popular tests. Pearson χ^2, Likelihood ratio, the Hellinger distance, Kullback-Leibler divergence, and the Neyman modified χ^2 correspond with $\lambda = 1, 0, -1/2, -1, -2$, respectively.

- Approximation of the test statistic distribution with the
- Detection rate with varying λ and number of total counts.
- Detection rate with varying λ and different null- and alternative hypotheses.
- Detection rate with varying λ and varying zero-cell penalty.

The test procedure is as follows. We generated three data sets, call them R, D, F. The R set is data generated from the 'real' distribution, D the deviated data, thus generated by another multinomial distribution. Another set F, similar to R which was used to determine the real false alarm rate. The set R is used to find the parameters of the real multinomial using the maximum likelihood estimation. Then the empirical distribution of the test statistic under the real distribution is determined using R. This is done by calculating the scores according the Power Divergence function. The threshold is set on the value of the top 1 percent case. This is how the threshold is set by simulation. In the case of the approximate χ^2 distribution the threshold is set to the 0.99-quantile of the χ^2 distribution. Using these thresholds we counted the number of cases from D exceeding the thresholds using the estimated parameters. The total of the counts divided by the size of the set D is the detection rate. These are the ways to set the thresholds either by simulation or by the χ^2 distribution. The threshold found under the distribution describes the 1 percent threshold and reflects a probability of 0,01 on exceeding the threshold.

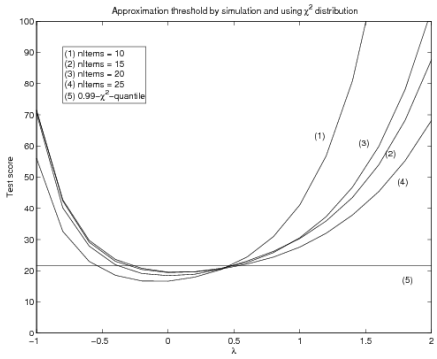


Figure 2. Experiment1: The straight line is the 0.99-quantile of the χ^2 distribution, whereas other lines are the 0.99-quantiles found by simulation for different amount of items.

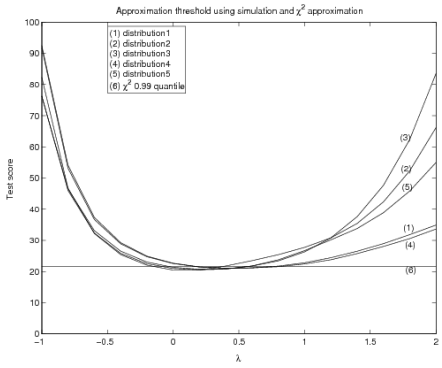


Figure 3. Experiment1: The straight line is the 0.99-quantile of the χ^2 distribution, whereas other lines are the 0.99-quantiles found by simulation for different null hypotheses.

5. Experiment

In all the experiments the following parameters are used:

- dimensionality = 10
- λ = -2:0.2:2 + e^{-10}

Where dimensionality is the number of parameters of the multionamial distribution and λ represents the different tests.

Experiment 1 The approximation by simulation and χ^2 of the 0.99-quantile of the distribution of the test statistics in the case of different number of items is shown in Figure 2 and 3.

In both figures the straight line is the 0.99-quantile of the χ^2 distribution. In figure 2 is displayed different

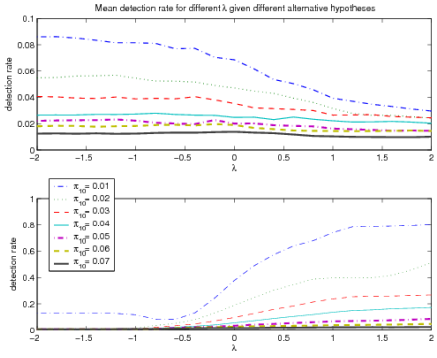


Figure 4. Experiment2: The detection rate of different tests in the case of H_0 is uniform and H_1 is skewed and vice versa.

combinations of λ and the number of counts the 0.99-quantile found by simulation. In Figure 3 it is done for different distributions. Tests for which λ are closer to zero are usually closer to the quantile of the χ^2 distribution. The bigger the number of items the closer the χ^2 approximation.

Experiment 2 The detection rate of two cases are compared. First a uniform null hypothesis and a family of skewed alternative hypotheses. The alternative hypotheses are distributions of which its 10th category is one of the list of H2 displayed hereunder and the 1 till 9-th are equal. Second the skewed alternatives are null-hypotheses and the uniform distribution is the alternative.

- number of counts per sample = 10 when h_0 = H2, 40 when h_1 = H1
- H1 = [0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1]
- H2 = $\pi_1 0$ = [0.01, 0.02, 0.03, 0.04, 0.05, 0.06, 0.07]

The results of experiment2 are shown in Figure 4. The uppermost graph in Figure 4 represents the case where H0 is the null hypothesis h_0 and H1 the alternative hypothesis h_1. The lowermost graph represents the case where H1 is the null hypothesis and H0 the alternative. It can be seen that in the uppermost negative λ are better whenever the last category is smaller and vice versa.

Experiment3 The detection rate is determined in the situation he penalty is added. The penalty is a weight h on the sum of empty cells. Different penalties are investigated.

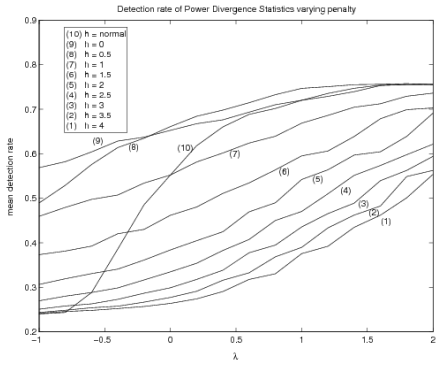


Figure 5. Experiment3: The detection rate of different tests extended with a penalty term for zero-cells. The h=normal references to $h = \frac{1}{\lambda+1}$, and is equal to the power family without penalty.

The result is shown in Figure 5 In the figure $h = normal$ references to the case $h = \frac{1}{\lambda+1}$ which is equal to the family without penalty term. As can be seen the detection rate of negative λ can increase by adding penalty.

Experiment4 Two experiments were conducted. In the first the average detection rate is determined in the case of several null hypotheses over 100 randomly sampled from the uniform distribution over the probability simplex. The distributions were sampled by the following procedure: Generate a random vector of dimensionality 9. Sort this vector. Add zero and 1 to the front and back of the vector. Then the differences between π_{i+1} and π_i are uniformly distributed. And in the second case the average detection rate is determined in the case of several null hypotheses over 100 randomly sampled from the skewed distribution over the probability simplex. The skewed distributions were sampled in a simular way. We sampled uniformly distributed distributions of dimensionality 14. Then we replaced the last five categories by one category equal to the sum over the last five.

The results are displayed in figure 6 and 7 Although in the case of skewed alternatives all the tests gave simular results.

Experiment5 The average detection rate was measured in the case were the null hypothesis is a skewed distribution having uniformly and skewed distributed alternative. Two kind of skewed distributions were chosen. One that is rightly skewed. By normalizing the vector $\exp 1:10$, and multiplying it by a factor $1-\eta$ and adding $\eta/10$, for $\eta = 0.05, 0.1, 0.25, 0.4$ we obtain the skewed distributions of the first kind. The

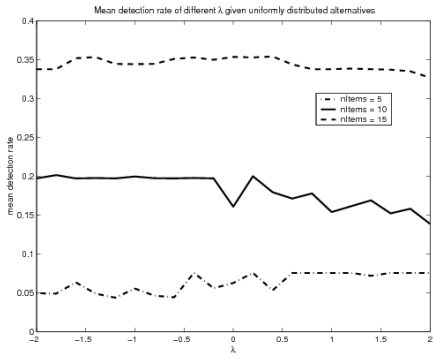


Figure 6. Experiment4: The mean detection rate given different uniformly ditributed alternatives. The null hypothesis is uniform. The detection rate is averaged over 100 uniform and skewed alternative distributions.

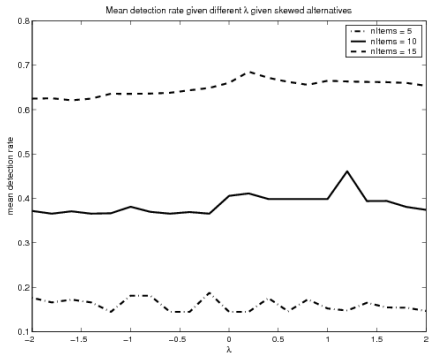


Figure 7. Experiment 4: The mean detection rate given different skewly distributed alternative hypotheses. The null hypothesis is uniform. The detection rate is averaged over 100 uniform and skewed alternative distributions.

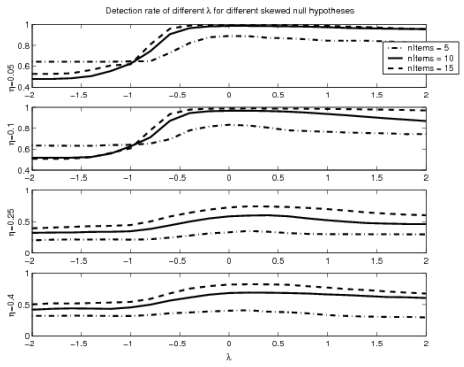


Figure 8. Experiment5a: The mean detection rate given different alternative hypotheses. The null hypothesis is equal to several skewed distributions defined by a. Bigger η are more skewed. The detection rate is averaged over 100 multinomial distributions uniformly distributed.

- $H0_1 = \text{skewed1} : \eta = 0.05,\ 0.1,\ 0.25,\ 0.4$
- $H0_2 = \text{skewed2} : \eta = 0.1,\ 0.25,\ 0.4$

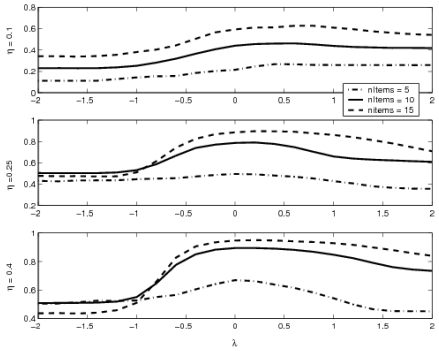


Figure 9. Experiment5b: The mean detection rate given different alternative hypotheses. The null hypothesis is equal to a uniform distribution. The detection rate is averaged over 100 skewed alternative distributions.

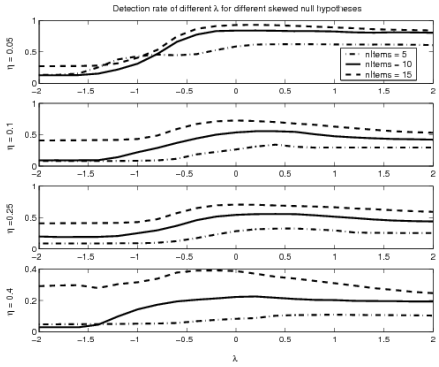


Figure 10. Experiment5c: The mean detection rate given different alternative hypotheses. The null hypothesis is equal to several skewed distributions defined by a. Bigger a are more skewed. The detection rate is averaged over 100 multinomial distributions uniformly distributed.

second kind of distributions were generated by 0.1 - skewed1*0.2 + 0.02, for $\eta = 0.1, 0.25, 0.4$. In this way the most right categories of the distribution are less bigger than the rest.

The results are shown in the Figures 8, 9, 10, 11. In the figures the different subplots correspond with the different skewed distribution. Whenever the plot shows 4 subplots it means they are rightly skewed. Else the other skewed distribution. All the test experiments were performed with number of items of 5, 10 and 15. The subplots are defined by η which resembles a distribution mentioned before. In all the graphs λ around zero have highest detection rate.

6. Discussion

The power of the power statistics differs and is dependent on many factors, the null hypothesis, the alternative hypothesis and the number of items. In practice we don't know this amount. It is unfeasible to tabulate for which λ, π of different dimension which corresponding number of counts would be sufficient. However whether we can use the χ^2-approximation for some λ can easily be tested. The number of items for which the χ^2 approximation is reasonable is dependent on the dimensions and distribution itself and can the number can fastly be found by simulation. To decide which λ is best is dependent on the alternative distribution. In some special cases heuristics can be used. As shown is that in most of the cases positive λ are to be preferred. Another important fact is that positive cases are sensitive to outliers and negative to inliers. So when one category would jump upwards positive λ are to be preferred.

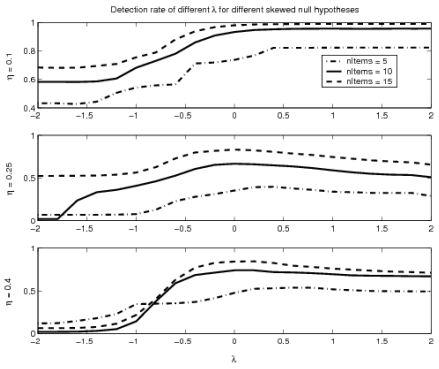


Figure 11. Experiment5d: The mean detection rate given different alternative hypotheses. The null hypothesis is equal to a uniform distribution. The detection rate is averaged over 100 skewed alternative distributions.

The family of Power Divergence tests encompasses a large variety of tests as possible goodness-of-fit-tests to determine deviation in the multinomial distribution. Making use of these tests has some advantages. By selecting different λ we can control the direction of sensitivity of the tests. This can be advantageous in the cases where it is approximately known in which direction the deviation will be. This is an advantage over for example using the Likelihood as the test statistic. The likelihood test is evenly sensitive because it maps the observations to their probabilities given a certain distribution. Setting a threshold will make the test equally sensitive in all directions. Another advantage of using the Power Divergence test statistics is that it is proven to be asymptotically χ^2 distributed. This approximation is dependent on the distribution and λ. The approximation is useable for a certain minimal number of items. The approximation of χ^2 given small amount of samples is not applicable and finding a suitable test can be done by simulation.

7. Conclusion

In this article we described approaches to detect deviation in multinomial models. These approaches are made from the context of hypothesis testing. In the case of a single multinomial model we gave an overview of using the family of Power Divergence statistics where its members are determined by λ. Different λ make the test sensitive to different observables. The power of these tests follow a nice unimodal curve making the choice of λ more stable. Increasing the amount of observations increases the power of the test. Using negative λ can lead to oversensitivity of the tests and this can be corrected by replacing the zero cells in the original family of Power Divergence statistics by a penalty term. Using these tests has several advantages, namely the approximate χ^2 distribution and the control of the sensitivity towards a certain change. A lot of different tests are available for use under different alternative hypotheses. When the alternatives are uniformly distributed, a λ close to zero is favoured. In the case of skewed alternative distributions positive λ are to be preferred above negative ones and when the null hypothesis is uniform the differences between tests are small. However when more information about the alternative distribution is known better tests can be chosen. Tests defined by positive λ are more sensitive to the situation in which the observed is bigger than expected and for negative λ more sensitive when the observed is smaller than expected. Selecting the best test can increase the power. Selecting a good test can be done by simulating.

References

Agostino, R., & Stephens, M. (1986). *Goodness-of-fit-tests*, vol. 86. Madison Avenue, New York, New York: Dekker.

Basu, A., Ray, S., Park, C., & Basu, S. (2002). Improved power in multinomial goodness-of-fit tests. *Journal royal statistical society, series B.*, *51*, 381–393.

Chapman, J. (1976). A comparison of the χ^2, -2 log r, and the multinomial probability criterium for significance testing when the expected frequencies are small. *Journal of the American Statistical Association*, *71*, 854–863.

Cressie, N., & Read, T. (1984). Multinomial goodness-of-fit tests. *Journal royal statistical society, series, seriec B.*, *46*, 440–464.

Lehman, E. (1997). *Testing statistical hypothesis*. Springer.

Ye, N., Chen, Q., Emran, S., & Noh, K. (2000). Chi-square statistical profiling for anomaly detection. *Proceedings IEEE Systems*.

Master Algorithms for Active Experts Problems based on Increasing Loss Values

Jan Poland JAN@IDSIA.CH
Marcus Hutter MARCUS@IDSIA.CH
IDSIA, Galleria 2, CH-6928 Manno-Lugano, Switzerland, http://www.idsia.ch

Abstract

We specify an experts algorithm with the following characteristics: (a) it uses only feedback from the actions actually chosen (bandit setup), (b) it can be applied with countably infinite expert classes, and (c) it copes with losses that *may grow in time* appropriately slowly. We prove loss bounds against an adaptive adversary. From this, we obtain master algorithms for "active experts problems", which means that the master's actions may influence the behavior of the adversary. Our algorithm can significantly outperform standard experts algorithms on such problems. Finally, we combine it with a universal expert class. This results in a (computationally infeasible) *universal master algorithm* which performs – in a certain sense – almost as well as any computable strategy, for any online problem.

1. Introduction

Expert algorithms have been popular since about fifteen years ago (Littlestone & Warmuth, 1989). They are appropriate for online prediction or repeated decision making or repeated game playing (we call these setups *online problems* for brevity), based on a class of "experts". In each round, each expert gives a recommendation. From this, we derive a master decision. After that, losses (or rewards) are assigned to each expert by the environment, also called adversary. Our goal is to perform almost as well as the best expert in hindsight in the long run. In other words, we try to minimize the regret.

The early papers deal with the full information game, where we get to know the losses of each expert after each round. The analysis holds for the *worst case*, where the environment is fully adversarial and tries to maximize our regret in the long run. Later, Auer et al. (1995) gave a worst-case analysis for the *bandit* setup, where the master algorithm knows only the loss of its own decision after each round. This has been further generalized to *label-efficient prediction* (Helmbold & Panizza, 1997) and *partial monitoring* (Cesa-Bianchi et al., 2004).

Recently, de Farias and Megiddo (2004) introduced a *strategic experts algorithm* which performs well for a broader class of environments. The algorithm has still asymptotically optimal properties against a worst-case adversary. Additionally, it may perform much better than a standard experts algorithm in more favorable situations, when the actions influence the behavior of the environment. We refer to these as *active experts problems*. One example is the repeated prisoner's dilemma when the opponent is willing to cooperate under certain conditions (see Section 5 for some details). However, de Farias and Megiddo give only asymptotic guarantees, but no convergence rate.

In this paper, we introduce a different algorithm for active experts problems with the same asymptotic guarantees, but in addition a convergence rate (of $t^{-\frac{1}{10}}$) is shown. Both algorithm and analysis are assembled from a standard "toolkit", basing on Kalai and Vempala (2003); McMahan and Blum (2004). The basic idea is the following: We use the bandit experts algorithm by McMahan and Blum, but allow the losses to increase with time t. This allows us to give control to one expert for an *increasing* period of time steps.

Secondly, we generalize our analysis to the case of *infinitely many* experts, basing on Hutter and Poland (2004b). The master algorithm stays computable (if the experts are), since only a finite (with time increasing) number of experts is involved. Allowing infinitely many experts also permits to define a *universal expert class* by means of all programs on some universal Turing machine. (This construction is quite common in Algorithmic Information Theory, see e.g. Hutter, 2004.) Thus, we obtain a *universal master algorithm*,

which we show to perform in a certain sense almost as well as *any computable* strategy on *any online problem*. Thus, we introduce a new approach to universal artificial intelligence, which is in a sense dual to the AIXI model based on Bayesian learning (Hutter, 2004). Although the master algorithm is computable, the resulting universal agent is not (like the AIXI model), since the experts may be non-responsive.

The paper is structured as follows. Section 2 introduces the problem setup, the notation, and the algorithm. In Sections 3 and 4, we give the (worst-case) analysis for finite and infinite expert classes. The implications to active experts problems and a universal master algorithms are given in Section 5. Section 6 contains discussion and conclusions.

2. The Algorithm

Our task is an online decision problem. That is, we have to make a sequence of decisions, each of which results in a certain loss we incur. "We" is an abbreviation for the master algorithm which is to be designed. For concreteness, you may imagine the task of playing a game repeatedly. In each round, i.e. at each time step t, we have access to the recommendations of $n\in\mathbb{N}\cup\{\infty\}$ "experts" or strategies. We do not specify what exactly a "recommendation" is – we just follow the advice of one expert. *Before* we reveal our move, the adversary has to assign losses $\ell_t^i\geq 0$ to *all* experts i. There is an upper bound B_t on the maximum loss the adversary may use, i.e. $\ell_t\in[0,B_t]^n$. This quantity may depend on t and is known to us. After the move, only the loss of the selected expert i is revealed. This is the *bandit setup*, as opposed to the full information game where we get to know the losses *all* experts. Our goal is to perform nearly as well as the best available strategy in terms of cumulative loss, after any number T of time steps which is not known in advance. The difference between our loss and the loss of some expert is also termed *regret*. We consider the general case of an *adaptive* adversary, which may assign losses depending on our past decisions.

If there is a finite number n of experts or strategies, then it is common to give no prior preferences to any of them. Formally, we define *prior weights* $w^i=\frac{1}{n}$. Moreover, we define the complexity of expert i as $k^i=-\ln w^i$. This arises in the full observation game, where the regret can be bounded by some function of the best expert's complexity. On the other hand, if there are reasons not to trust all strategies equally in the beginning, we may use a non-uniform prior w. This is mandatory for infinitely many experts. We then require $w^i>0$ for all experts i and $\sum_i w^i\leq 1$.

For $t=1,2,3,...$
Sample $r_t\in\{0,1\}$ independently s.t. $P[r_t=1]=\gamma_t$
If $r_t=0$ Then
  Play *FPL*(t)'s decision ($I_t^{FoE}:=I_t^{FPL}$)
  Set $\hat\ell_t^i=0$ for all $1\leq i\leq n$
Else
  Sample $I_t^{FoE}\in\{1...n\}$ uniformly & play $I:=I_t^{FoE}$
  Let $\hat\ell_t^I=\ell_t^I n/\gamma_t$ and $\hat\ell_t^i=0$ for all $i\neq I$

Figure 1. The algorithm *FoE*

Sample $q_t^i\overset{d.}{\sim}Exp$ independently for $1\leq i\leq n$
select and play $I_t^{FPL}=\arg\min_{1\leq i\leq n}\{\eta_t\hat\ell^i_{<t}+k^i-q_t^i\}$

Figure 2. The algorithm *FPL*(t)

Our algorithm "Follow or Explore" (*FoE*) builds on McMahan and Blum's online geometric optimization algorithm. (For finite n and uniform prior, it even is their algorithm, save for the adaptive parameters.) It is a bandit version of a "Follow the Perturbed Leader" experts algorithm. This approach to online prediction and playing repeated games has been pioneered by Hannan (1957). For the full observation game, Kalai and Vempala (2003) gave a very elegant analysis which is distinct from the standard analysis of exponential weighting schemes. It is particularly handy if the learning rate is dynamic rather than fixed in advance. A dynamic learning rate is necessary if there is no target time T known in advance.

The algorithm is composed of two standard ingredients: *exploration* and *follow the (perturbed) leader*. Since we are playing the bandit game (as opposed to the full information game), we need to explore sufficiently. Otherwise, there could be a strategy which we think is poor (and thus never play), but in reality it is good. At each time step t, we decide randomly according to some exploration rate $\gamma_t\in(0,1)$ whether to explore or not. If so, we choose an expert according to the uniform distribution (or the prior distribution, compare (5), in case of non-uniform priors). After observing the loss of the selected expert, we want to give an *unbiased estimate* of the true loss vector. We achieve that by dividing the observed loss by the probability of exploring this expert, and estimate the unobserved losses of all other experts by zero. We call the resulting loss vector $\hat\ell_t$.

When not exploring, we follow some strategy which performed well in the past. It may be not advisable to pick always the *best* strategy so far - the adver-

sary could fool us in this case. Instead we introduce a *perturbation* for each expert and follow the advice of the strategy with the best perturbed score. In order to assign a score to each expert, note that we have only access to the *estimated* losses $\hat{\ell}_t$. Let $\hat{\ell}^i_{<T} = \sum_{t=1}^{T-1} \hat{\ell}^i_t$ be the estimated cumulative past loss of expert i. Then his complexity-penalized score is defined as $\eta_T \hat{\ell}^i_{<T} + k^i$, i.e. high scores are bad. Here, $\eta_T > 0$ is the *learning rate*. The perturbed score is then given by $\eta_T \hat{\ell}^i_{<T} + k^i - q^i$, where the perturbations q^i are chosen independently exponentially distributed. This ensures a convenient analysis.

The algorithms "Follow or Explore" *FoE* and "Follow the perturbed Leader" *FPL* are fully specified in Figures 1 and 2. Note that each time randomness is used, it is assumed to be *independent* of the past randomness. Note also that all algorithms occurring in this paper work with the *estimated losses* $\hat{\ell}$. We may evaluate their performance in terms of true or estimated losses, this is specified in the notation. E.g. for the true loss of *FPL* up to and including time T we write $L^{FPL} = \ell^{FPL}_{1:T}$, while the estimated loss is $\hat{L}^{FPL} = \hat{\ell}^{FPL}_{1:T}$.

3. Analysis for Uniform Prior

In this section we assume a uniform prior $w \equiv \frac{1}{n}$ over finitely many experts. (The general case is treated in the next section.) We assume that $B_t \geq 0$ is some sequence of upper bounds on the true losses, $\gamma_t \in (0,1)$ is a sequence of exploration rates, and $\eta_t > 0$ is a *decreasing* sequence of learning rates.

The analysis is according to the following diagram:

$$L^{FoE} \lesssim \mathbf{E}L^{FoE} \lesssim \mathbf{E}L^{FPL} \lesssim \mathbf{E}\hat{L}^{FPL} \lesssim \mathbf{E}\hat{L}^{IFPL} \lesssim \hat{L}^{best} \lesssim L^{best} \quad (1)$$

The symbol L is used informally for the cumulative loss $\ell_{1:T}$. Each "$\lesssim$" means that we bound the quantity on the left by the quantity on the right plus some additive terms. The first and the last expressions are the losses of the *FoE* algorithm and the best expert, respectively. The intermediate quantities belong to different algorithms, namely *FoE*, *FPL*, and a third one called *IFPL* for "infeasible" FPL (Kalai & Vempala, 2003). *IFPL* is the same as *FPL* except that it has access to an oracle providing the current estimated loss vector $\hat{\ell}_t$ (hence infeasible). Then it assigns scores of $\eta_t \hat{\ell}^i_{1:t} + k^i - q^i_t$ instead of $\eta_t \hat{\ell}^i_{<t} + k^i - q^i_t$. We assume that *IFPL* uses the same randomization as *FPL* (i.e. the respective q_t are the same).

The randomization of *FoE* and *FPL* gives rise to two filters of σ-algebras. By $\mathcal{A}_t$ for $t \geq 0$ we denote the σ-algebra generated by the *FoE*'s randomness $\{u_{1:t}, r_{1:t}\}$ up to time t. We may also write $\mathcal{A} = \bigcup_{t \geq 0} \mathcal{A}_t$. Similarly, $\mathcal{B}_t$ is the σ-algebra generated by the *FoE*'s *and FPL*'s randomness up to time t (i.e. $\mathcal{B}_t \widehat{=} \{u_{1:t}, r_{1:t}, q_{1:t}\}$). Then clearly $\mathcal{A}_t \subset \mathcal{B}_t$ for each t.

The arguments below rely on *conditional expectations* – the expectations in (1) should also be understood conditional. In particular we will often need the conditional expectations with respect to *FoE*'s past randomness $\mathcal{A}_{t-1}$, abbreviated as

$$\mathbf{E}_t[X] := \mathbf{E}[X|\mathcal{A}_{t-1}],$$

where X is some random variable. Then $\mathbf{E}_t[X]$ is an $\mathcal{A}_{t-1}$-measurable random variable, meaning that its value is determined for fixed past randomness $\mathcal{A}_{t-1}$. Note in particular that the estimated loss vectors $\hat{\ell}^i_t$ are random vectors which depend on *FoE*'s randomness $\mathcal{A}_t$ up to time t (only). In this way, *FoE*'s (and *FPL*'s and *IFPL*'s) actions depend on *FoE*'s past randomness. Note, however, that they do not depend on *FPL*'s randomness $q_{1:t}$. Finally, I^{FoE}_t and ℓ^{FoE}_t are $\mathcal{A}'_t$ measurable, i.e. depend on $u_{<t}, r_{<t}, q_t$, but are independent of $q_{<t}$.

We now start proving the diagram (1). It is helpful to consider each intermediate algorithm as a standalone procedure which is actually executed (with an oracle if necessary) and has the asserted performance guarantees (e.g. in terms of expected losses).

Lemma 1 $\left[L^{FoE} \lesssim \mathbf{E}L^{FoE}\right]$ *For each* $T \geq 1$ *and* $\delta_T \in (0,1)$*, with probability at least* $1 - \frac{\delta_T}{2}$*, we have*

$$\ell^{FoE}_{1:T} \leq \sum_{t=1}^{t} \mathbf{E}_t \ell^{FoE}_t + \sqrt{(2 \ln \tfrac{4}{\delta_T}) \sum_{t=1}^{T} B_t^2}.$$

Proof. The sequence of random variables $X_T = \sum_{t=1}^{T} \left[\ell^{FoE}_t - \mathbf{E}_t \ell^{FoE}_t\right]$ is a martingale with respect to the filter $\mathcal{B}_t$ (not $\mathcal{A}_t$!). In order to see this, observe $\mathbf{E}[\ell^{FoE}_T | \mathcal{B}_{T-1}] = \mathbf{E}\big(\mathbf{E}[\ell^{FoE}_T | \mathcal{A}_{T-1}] \big| \mathcal{B}_{T-1}\big)$ and $\mathbf{E}[\ell^{FoE}_t | \mathcal{B}_{T-1}] = \ell^{FoE}_t$ for $t < T$, which implies

$$\begin{aligned}
&\mathbf{E}(X_T | \mathcal{B}_{T-1}) = \\
&\quad = \sum\nolimits_{t=1}^{T} \big(\mathbf{E}[\ell^{FoE}_t | \mathcal{B}_{T-1}] - \mathbf{E}\big[\mathbf{E}[\ell^{FoE}_t | \mathcal{A}_{t-1}] \big| \mathcal{B}_{T-1}\big]\big) \\
&\quad = \sum\nolimits_{t=1}^{T-1} \big(\ell^{FoE}_t - \mathbf{E}[\ell^{FoE}_t | \mathcal{A}_{t-1}]\big) = X_{T-1}.
\end{aligned}$$

Its differences are bounded: $|X_t - X_{t-1}| \leq B_t$. Hence, it follows from Azuma's inequality that the probability that X_T exceeds some $\lambda > 0$ is bounded by $p = 2\exp\big(-\frac{\lambda^2}{2\sum_t B_t^2}\big)$. Requesting $\frac{\delta_T}{2} = p$ and solving for λ gives the assertion. □

The relation $\mathbf{E}L^{FoE} \lesssim \mathbf{E}L^{FPL}$ follows immediately from the specification of the algorithm *FoE*.

Lemma 2 $\left[\mathbf{E}L^{FoE}\lesssim\mathbf{E}L^{FPL}\right]$ *For each* $t\geq 1$*, we have* $\mathbf{E}_t\ell_t^{FoE}\leq(1-\gamma_t)\mathbf{E}_t\ell_t^{FPL}+\gamma_t B_t$.

The next lemma relating $\mathbf{E}L^{FPL}$ and $\mathbf{E}\hat{L}^{FPL}$ is technical but intuitively clear. It states that in (conditional) expectation, the real loss suffered by *FPL* is the same as the estimated loss. This is simply because the loss estimate is unbiased. A combination with the previous lemma was shown in McMahan and Blum (2004).

Lemma 3 $\left[\mathbf{E}L^{FPL}\lesssim\mathbf{E}\hat{L}^{FPL}\right]$ *For each* $t\geq 1$*, we have* $\mathbf{E}_t\ell_t^{FPL}=\mathbf{E}_t\hat{\ell}_t^{FPL}$.

Note that $\hat{\ell}_t^{FPL}$ is the loss $\hat{\ell}_t^I$ estimated by *FoE*, but for the expert $I=I_t^{FPL}$ chosen by *FPL*.

Proof. Let $f_t^i=f_t^i(\mathcal{A}_{t-1})=\mathbf{P}[I_t^{FPL}=i|\mathcal{A}_{t-1}]$ be the probability distribution over actions i which *FPL* uses at time t, depending on the past randomness $\mathcal{A}_{t-1}$. Let $u_t=[1...1]/n$ be the uniform distribution at time t (for non-uniform weights this will be replaced appropriately later). Then

$$\begin{aligned}\mathbf{E}_t[\hat{\ell}_t^{FPL}] =&\gamma_t\textstyle\sum_{i=1}^n f_t^i[(1-u_t^i)\cdot 0+u_t^i\hat{\ell}_t^i|_{r_t=1\wedge I_t^{FoE}=i}]\\ =&\textstyle\sum_{i=1}^n f_t^i\ell_t^i=\mathbf{E}_t[\ell_t^{FPL}],\end{aligned}$$

where $\hat{\ell}_t^i|_{r_t=1\wedge I_t^{FoE}=i}=\ell_t^i/(u_t^i\gamma_t)$ is the estimated loss under the condition that *FoE* decided to explore ($r_t=1$) and chose action $I_t^{FoE}=i$. □

The following lemma from Kalai and Vempala (2003) relates the losses of *FPL* and *IFPL*. We repeat the proof, since it is the crucial and only step in the analysis where we have to be careful with the upper loss bound B_t. Let $\hat{B}_t=B_t(n/\gamma_t)$ denote the upper bound on the instantaneous estimated losses.

Lemma 4 $\left[\mathbf{E}\hat{L}^{FPL}\lesssim\mathbf{E}\hat{L}^{IFPL}\right]$ $\mathbf{E}_t\hat{\ell}_t^{FPL}\leq\mathbf{E}_t\hat{\ell}_t^{IFPL}+\gamma_t\eta_t\hat{B}_t^2$ *holds for all* $t\geq 1$.

Proof. If $r_t=0$, $\hat{\ell}_t=0$ and thus $\hat{\ell}_t^{FPL}=\hat{\ell}_t^{IFPL}$ holds. This happens with probability $1-\gamma_t$. Otherwise we have

$$\mathbf{E}_t\hat{\ell}_t^{FPL}=\sum_{i=1}^n\int\mathbb{1}_{I_t^{FPL}=i}\hat{\ell}_t^i d\mu(x), \tag{2}$$

where μ denotes the (exponential) distribution of the perturbations, i.e. $x_i:=q_t^i$ and density $\mu(x):=\mathrm{e}^{-\|x\|_\infty}$. The idea is now that if action i was selected by *FPL*, it is – because of the exponentially distributed perturbation – with high probability also selected by *IFPL*. Formally, we write $u^+=\max(u,0)$ for $u\in\mathbb{R}$, abbreviate $\lambda=\hat{\ell}_{<t}+k/\eta_t$, and denote by $\int...d\mu(x_{\neq i})$ the integration leaving out the ith action. Then, using $\eta_t\lambda_i-x_i\leq\eta_t\lambda_j-x_j$ for all j if $I_t^{FPL}=i$ in the first line, and $\hat{B}_t\geq\hat{\ell}_t^i-\hat{\ell}_t^j$ in the fourth line, we get

$$\begin{aligned}\int\mathbb{1}_{I_t^{FPL}=i}\hat{\ell}_t^i d\mu(x) &= \int\int_{x_i\geq\max_{j\neq i}\{\eta_t(\lambda_i-\lambda_j)+x_j\}}\hat{\ell}_t^i d\mu(x_i)d\mu(x_{\neq i})\\ &= \int\hat{\ell}_t^i\,\mathrm{e}^{-(\max_{j\neq i}\{\eta_t(\lambda_i-\lambda_j)+x_j\})^+}d\mu(x_{\neq i})\\ &\leq \int\hat{\ell}_t^i\,\mathrm{e}^{\eta_t\hat{B}_t}\mathrm{e}^{-(\max_{j\neq i}\{\eta_t(\lambda_i-\lambda_j)+x_j\}+\eta_t\hat{B}_t)^+}d\mu(x_{\neq i})\\ &\leq \mathrm{e}^{\eta_t\hat{B}_t}\int\hat{\ell}_t^i\,\mathrm{e}^{-(\max_{j\neq i}\{\eta_t(\lambda_i+\hat{\ell}_t^i-\lambda_j-\hat{\ell}_t^j)+x_j\})^+}d\mu(x_{\neq i})\\ &= \mathrm{e}^{\eta_t\hat{B}_t}\int\mathbb{1}_{I_t^{IFPL}=i}\hat{\ell}_t^i d\mu(x).\end{aligned}$$

Summing over i and using the analogue of (2) for *IFPL*, we see that if $r_t=1$, then $\mathbf{E}_t\hat{\ell}_t^{FPL}\leq\mathrm{e}^{\eta_t\hat{B}_t}\mathbf{E}_t\hat{\ell}_t^{IFPL}$ holds. Thus $\mathbf{E}_t\hat{\ell}_t^{IFPL}\geq\mathrm{e}^{-\eta_t\hat{B}_t}\mathbf{E}_t\hat{\ell}_t^{FPL}\geq(1-\eta_t\hat{B}_t)\mathbf{E}_t\hat{\ell}_t^{FPL}\geq\mathbf{E}_t\hat{\ell}_t^{FPL}-\eta_t\hat{B}_t^2$. The assertion now follows by taking expectations w.r.t r_t. □

The next lemma relates the losses of *IFPL* and the best action in hindsight. For an oblivious adversary (which means that the adversary's decisions do not depend on our past actions), the proof was given in Kalai and Vempala (2003). An additional step is necessary for an adaptive adversary. We omit the proof here, the reader may reconstruct it from the proof of Lemma 9.

Lemma 5 $\left[\mathbf{E}\hat{L}^{IFPL}\lesssim\hat{L}^{best}\right]$ *Assume decreasing learning rate* η_t *and* $\sum_i\mathrm{e}^{-k^i}\leq 1$*. For all* $T\geq 1$ *and* $1\leq i\leq n$*, we have* $\sum_{t=1}^T\mathbf{E}_t\hat{\ell}_t^{IFPL}\leq\hat{\ell}_{1:T}^i+\frac{k^i}{\eta_T}$ *(recall that* $\hat{\ell}_{1:T}^i$ *is a random variable depending on* $\mathcal{A}_t$*).*

Finally, we give a relation between the estimated and true losses, adapted from McMahan and Blum (2004).

Lemma 6 $\left[\hat{L}^{best}\lesssim L^{best}\right]$ *For each* $T\geq 1$*,* $\delta_T\in(0,1)$*, and* $1\leq i\leq n$*, w.p. at least* $1-\frac{\delta_T}{2}$ *we have*

$$\hat{\ell}_{1:T}^i\leq\ell_{1:T}^i+\sqrt{(2\ln\tfrac{4}{\delta_T})\textstyle\sum_{t=1}^T\hat{B}_t^2}. \tag{3}$$

Proof. $X_t=\hat{\ell}_{1:t}^i-\ell_{1:t}^i$ is a martingale, since

$$\begin{aligned}\mathbf{E}[X_t|\mathcal{A}_{t-1}] &= \mathbf{E}[\hat{\ell}_{1:t}^i|\mathcal{A}_{t-1}]-\ell_{1:t}^i\\ &= X_{t-1}+\mathbf{E}[\hat{\ell}_t^i|\mathcal{A}_{t-1}]-\ell_t^i=X_{t-1}.\end{aligned}$$

Its differences are bounded: $|X_t-X_{t-1}|\leq\hat{B}_t$. By Azuma's inequality, its actual value at time T does not exceed $\sqrt{(2\ln\frac{4}{\delta_T})\sum_{t=1}^T\hat{B}_t^2}$ w.p. $1-\frac{\delta_T}{2}$. □

We now combine the above results and derive an upper bound on the expected regret of *FoE* against an adaptive adversary.

Theorem 7 [*FoE* against an adaptive adversary] *Let n be finite and $k^i=\ln n$ for all $1\le i\le n$. Let η_t be decreasing, and $\ell_t\in[0,B_t]^n$ some possibly adaptive assignment of loss vectors. Then for all experts i,*

$$\ell_{1:T}^{FoE} \le \ell_{1:T}^i + \sqrt{(2\ln\tfrac{4}{\delta_T})}\left(\sqrt{\sum_{t=1}^T \tfrac{B_t^2 n^2}{\gamma_t^2}} + \sqrt{\sum_{t=1}^T B_t^2}\right) + \tfrac{\ln n}{\eta_T} + \sum_{t=1}^T \tfrac{\eta_t B_t^2 n^2}{\gamma_t} + \sum_{t=1}^T \gamma_t B_t \quad w.p.\ 1-\delta_T \text{ and}$$

$$\mathbf{E}\ell_{1:T}^{FoE} \le \ell_{1:T}^i + \tfrac{\ln n}{\eta_T} + \sum_{t=1}^T \tfrac{\eta_t B_t^2 n^2}{\gamma_t} + \sum_{t=1}^T \gamma_t B_t + \sqrt{(2\ln\tfrac{4}{\delta_T})\sum_{t=1}^T \tfrac{B_t^2 n^2}{\gamma_t^2}} + \tfrac{\delta_T}{2}\sum_{t=1}^T \tfrac{B_t n}{\gamma_t}.$$

Proof. The first high probability bound follows by summing up all excess terms in the above lemmas, observing that $\hat B_t = B_t(n/\gamma_t)$. For the second bound on the expectation, we take expectations in Lemmas 2-5, while Lemma 1 is not used. For Lemma 6, a statement in expectation is obtained as follows: (3) fails w.p. at most $\frac{\delta_T}{2}$, in which case $\hat\ell_{1:T}^i - \ell_{1:T}^i \le \sum_{t=1}^T \hat B_t$. □

Corollary 8 *Under the conditions of Theorem 7,*

$$\begin{array}{lllll}
(i) & B_t\equiv 1 & \Rightarrow & \mathbf{E}\ell_{1:T}^{FoE} \le \ell_{1:T}^i + O(n^2T^{\frac34}\sqrt{\ln T}), \\
(ii) & B_t\equiv 1 & \Rightarrow & \ell_{1:T}^{FoE} \le \ell_{1:T}^i + O(n^2T^{\frac34}\sqrt{\ln T}), \\
(iii) & B_t = t^{\frac18} & \Rightarrow & \mathbf{E}\ell_{1:T}^{FoE} \le \ell_{1:T}^i + O(n^2T^{\frac78}\sqrt{\ln T}), \\
(iv) & B_t = t^{\frac18} & \Rightarrow & \ell_{1:T}^{FoE} \le \ell_{1:T}^i + O(n^2T^{\frac78}\sqrt{\ln T}),
\end{array}$$

for all i and T. Here, (ii) and (iv) hold with probability $1-T^{-2}$. Moreover, in both cases (bounded and growing B_t) FoE is asymptotically optimal, i.e.

$$\limsup_{T\to\infty} \tfrac{1}{T}\left(\ell_{1:T}^{FoE} - \min_i \ell_{1:T}^i\right) \le 0 \quad \textit{almost surely.}$$

$B_t = t^{\frac18}$ in (iii) and (iv) is just one choice to achieve asymptotic optimality while the losses may grow unboundedly. Asymptotic optimality is sometimes termed *Hannan-consistency*, in particular if the limit equals zero. We only show the upper bound.

Proof. (i) and (ii) follow by applying the previous theorem to $\eta_t = t^{-\frac12}$, $\gamma_t = t^{-\frac14}$, $\delta_T = T^{-2}$, and observing $\sum_{t=1}^T t^\alpha \le \int_0^{T+1} t^\alpha \le 2(T+1)^{1+\alpha}$ for $\alpha \ge -\frac12$. In order to obtain (iii) and (iv), set $\eta_t = t^{-\frac34}$, $\gamma_t = t^{-\frac14}$, and $\delta_T = T^{-2}$. The asymptotic optimality finally follows from the Borel-Cantelli Lemma, since

$$\mathbf{P}\left[\tfrac{1}{T}(\ell_{1:T}^{FoE} - \min_i \ell_{1:T}^i) > CT^{-\frac18}\sqrt{\ln T}\right] \le \tfrac{1}{T^2}$$

for an appropriate $C>0$ according to (ii) and (iv). □

For $t=1,2,3,...$
Sample $r_t\in\{0,1\}$ independently s.t. $P[r_t=1]=\gamma_t$
If $r_t=0$ Then
  Invoke $FPL^\tau(t)$ and play its decision
  Set $\hat\ell_t^i=0$ for $i\in\{t\ge\tau\}$
Else
  Sample I_t w.r.t. u_t in (5) and play $I:=I_t^{FoE^\tau}$
  Set $\hat\ell_t^I = \ell_t^I/(u_t^I\gamma_t)$ and $\hat\ell_t^i=0$ for $i\in\{t\ge\tau\}\setminus\{I\}$
Set $\hat\ell_t^i = \hat B_t$ for $i\notin\{t\ge\tau\}$

Figure 3. The algorithm FoE^τ

Sample $q_t^i \overset{d.}{\sim} Exp$ independently for $i\in\{t\ge\tau\}$
select and play $I_t^{FPL} = \arg\min_{i:t\ge\tau}\{\eta_t\hat\ell_{<t}^i + k^i - q_t^i\}$

Figure 4. The algorithm $FPL^\tau(t)$

4. Infinitely Many Experts and Arbitrary Priors

The following considerations are valid for both finitely and infinitely many experts with arbitrary prior weights w^i. For notational convenience, we write $n=\infty$ in the latter case. When admitting infinitely many experts, two difficulties arise: Since the prior weights of the experts sum up to one and thus become arbitrarily small, the estimated losses – obtained by dividing by these weights – would possibly get arbitrarily large. We therefore introduce, for each expert i, a time $\tau^i\ge 1$ at which the expert enters the game. All algorithms *FoE*, *FPL*, *IFPL* are substituted by counterparts FoE^τ, FPL^τ, $IFPL^\tau$ which use expert i only for $t\ge\tau^i$. Thus, the maximum estimated loss possibly assigned to these *active* experts is

$$\hat B_t = B_t/[\gamma_t \min\{w^i : t\ge\tau^i\}]. \tag{4}$$

We denote the set of active experts at time t by $\{t\ge\tau\} = \{i : t\ge\tau^i\}$. Experts which have not yet entered the game are given an estimated loss of $\hat B_t$. This also solves the computability problem: Since at every time t only a finite number of experts is involved, FoE^τ is computable (if each expert is). The algorithms FoE^τ and FPL^τ are specified in Figures 3 and 4.

Again, the analysis follows the outline (1). Lemmas 1–4 have equivalent counterparts, the proofs of which remain almost unchanged. In Lemma 3, the "uniform" distribution over experts u_t now becomes

$$u_t^i = w^i \mathbb{1}_{t\ge\tau^i}/[\textstyle\sum_j w^j \mathbb{1}_{t\ge\tau^j}]. \tag{5}$$

The upper bound on the estimated loss $\hat B_t$ in Lemma

4 is given by (4). We only need to prove assertions corresponding to Lemmas 5 and 6.

Lemma 9 $\big[\mathbf{E}\hat{L}^{IFPL^\tau} \lesssim \hat{L}^{best^\tau}\big]$ *Assume that* $\sum_i \mathrm{e}^{-k^i} \le 1$ *and* τ^i *depends monotonically on* k^i*, i.e.* $\tau^i \ge \tau^j$ *if and only if* $k^i \ge k^j$*. Assume decreasing learning rate* η_t*. For all* $T \ge 1$ *and all* $1 \le i \le n$*, we have*

$$\sum_{t=1}^{T} \mathbf{E}_t \hat{\ell}_t^{IFPL^\tau} \le \hat{\ell}^i_{1:T} + \tfrac{k^i+1}{\eta_T}.$$

Proof. This is a modification of the corresponding proofs in Kalai and Vempala (2003) and Hutter and Poland (2004b). We may fix the randomization $\mathcal{A}$ and suppress it in the notation. Then we only need to show

$$\mathbf{E}\hat{\ell}_{1:T}^{IFPL^\tau} \le \min_{1\le i\le n}\{\hat{\ell}^i_{1:T} + \tfrac{k^i+1}{\eta_T}\}, \tag{6}$$

where the expectation is with respect to *IFPL*'s randomness $q_{1:T}$.

Assume first that the adversary is oblivious. We define an algorithm A as a variant of $IFPL^\tau$ which samples only one perturbation vector q in the beginning and uses this in each time step, i.e. $q_t \equiv q$. Since the adversary is oblivious, A is equivalent to $IFPL^\tau$ in terms of expected performance. This is all we need to show (6). Let $\eta_0 = \infty$ and $\lambda_t = \hat{\ell}_t + (k-q)\big(\frac{1}{\eta_t} - \frac{1}{\eta_{t-1}}\big)$, then $\lambda_{1:t} = \hat{\ell}_{1:t} + \frac{k-q}{\eta_t}$. Recall $\{t \ge \tau\} = \{i : t \ge \tau^i\}$. We argue by induction that for all $T \ge 1$,

$$\sum_{t=1}^{T} \lambda_t^A \le \min_{T\ge\tau} \lambda^i_{1:T} + \max_{T\ge\tau}\big\{\tfrac{q^i-k^i}{\eta_T}\big\}. \tag{7}$$

This clearly holds for $T = 0$. For the induction step, we have to show

$$\min_{T\ge\tau} \lambda^i_{1:T} + \max_{T\ge\tau} \tfrac{q^i-k^i}{\eta_T} + \lambda^A_{T+1} \le \lambda^{I^A_{T+1}}_{1:T} \tag{8}$$
$$+ \max_{T+1\ge\tau} \tfrac{q^i-k^i}{\eta_{T+1}} + \lambda^{I^A_{T+1}}_{T+1} = \min_{T+1\ge\tau} \lambda^i_{1:T+1} + \max_{T+1\ge\tau} \tfrac{q^i-k^i}{\eta_{T+1}}.$$

The inequality is obvious if $I^A_{T+1} \in \{T \ge \tau\}$. Otherwise, let $J = \operatorname{argmax}\big\{q^i - k^i : i \in \{T \ge \tau\}\big\}$. Then

$$\min_{T\ge\tau}\lambda^i_{1:T} + \max_{T\ge\tau}\big\{\tfrac{q^i-k^i}{\eta_T}\big\} \le \lambda^J_{1:T} + \tfrac{q^J-k^J}{\eta_T} = \sum_{t=1}^{T} \hat{\ell}^J_t$$
$$\le \sum_{t=1}^{T} \hat{B}_t = \sum_{t=1}^{T} \hat{\ell}_t^{I^A_{T+1}} \le \lambda^{I^A_{T+1}}_{1:T} + \max_{T+1\ge\tau}\big\{\tfrac{q^i-k^i}{\eta_{T+1}}\big\}$$

shows (8). Rearranging terms in (7), we see

$$\sum_{t=1}^{T} \hat{\ell}^A_t \le \min_{T\ge\tau}\lambda^i_{1:T} + \max_{T\ge\tau^i}\big\{\tfrac{q^i-k^i}{\eta_T}\big\} + \sum_{t=1}^{T}(q-k)^{I^A_t}\big(\tfrac{1}{\eta_t} - \tfrac{1}{\eta_{t-1}}\big).$$

The assertion (6) – still for oblivious adversary and $q_t \equiv q$ – then follows by taking expectations and using

$$\mathbf{E}\min_{T\ge\tau}\lambda^i_{1:T} \le \min_{T\ge\tau}\{\hat{\ell}^i_{1:T} + \tfrac{k^i}{\eta_T} - \mathbf{E}\tfrac{q^i}{\eta_T}\} \overset{(*)}{\le} \min_{1\le i\le n}\{\hat{\ell}^i_{1:T} + \tfrac{k^i-1}{\eta_T}\}$$
$$\text{and } \mathbf{E}\sum_{t=1}^{T}(q-k)^{I^A_t}\big(\tfrac{1}{\eta_t} - \tfrac{1}{\eta_{t-1}}\big) \le \mathbf{E}\max_{T\ge\tau}\big\{\tfrac{q^i-k^i}{\eta_T}\big\} \le \tfrac{1}{\eta_T}.$$

Here, $(*)$ holds because τ^i depends monotonically on k^i, and $\mathbf{E}q^i = 1$, and maximality of $\hat{\ell}^i_{1:T}$ for $T < \tau_i$. The last inequality can be proven by an application of the union bound (Hutter & Poland, 2004b, Lem.1).

Sampling the perturbations q_t independently is equivalent under expectation to sampling q only once. So assume that q_t are sampled independently, i.e. that $IFPL^\tau$ is played against an oblivious adversary: (6) remains valid. In the last step, we argue that then (6) also holds for an *adaptive* adversary. This is true because the future actions of $IFPL^\tau$ do not depend on its past actions, and therefore the adversary cannot gain from deciding after having seen $IFPL^\tau$'s decisions. (For details see Hutter & Poland, 2004a. Note the subtlety that the future actions of FoE^τ would depend on its past actions.) □

Lemma 10 $\big[\hat{L}^{best^\tau} \lesssim L^{best}\big]$ *For each* $T \ge 1$*,* $\delta_T \in (0,1)$*, and* $1 \le i \le n$*, we have* $\hat{\ell}^i_{1:T} \le \ell^i_{1:T} + \sqrt{(2\ln\frac{4}{\delta_T})\sum_{t=1}^{T}\hat{B}_t^2} + \sum_{t=1}^{\tau^i-1}\hat{B}_t$ *w.p.* $1 - \frac{\delta_T}{2}$.

This corresponds to Lemma 6. The proof proceeds in a similar way: we have to note that $\hat{\ell}^i_{1:t} - \ell^i_{1:t}$ is a martingale only for $t \ge \tau^i$, and $\hat{\ell}^i_{<\tau^i}$ exceeds $\ell^i_{<\tau^i}$ by at most $\sum_{t=1}^{\tau^i-1}\hat{B}_t$. Then the following theorem corresponds to Theorem 7 and is proven likewise.

Theorem 11 [FoE^τ against an adaptive adversary] *Let* n *be finite or infinite,* $\sum_i \mathrm{e}^{-k^i} \le 1$*,* τ^i *depend monotonically on* k^i*, and the learning rate* η_t *be decreasing. Let* ℓ_t *some possibly adaptive assignment of (true) loss vectors satisfying* $\|\ell_t\|_\infty \le B_t$*. Then for all experts* i*, we have*

$$\ell^{FoE^\tau}_{1:T} \le \ell^i_{1:T} + \sqrt{(2\ln\tfrac{4}{\delta_T})}\left(\sqrt{\sum_{t=1}^{T}\tfrac{B_t^2}{\gamma_t^2(w_t^*)^2}} + \sqrt{\sum_{t=1}^{T}B_t^2}\right)$$
$$+ \tfrac{k^i+1}{\eta_T} + \sum_{t=1}^{\tau^i-1}\tfrac{B_t}{\gamma_t w_t^*} + \sum_{t=1}^{T}\tfrac{\eta_t B_t^2}{\gamma_t(w_t^*)^2} + \sum_{t=1}^{T}\gamma_t B_t$$

with probability $1 - \delta_T$*, where* $w_t^* = \min\{w^i : t \ge \tau^i\}$*. A corresponding statement holds for the expectation (compare Theorem 7).*

Corollary 12 *Assume the conditions of Theorem 11. Then for all i and T, the following holds w.p. $1-\delta_T$.*

$$(i)\ B_t\equiv 1, \tau^i=\lceil (w^i)^{-8}\rceil \\ \Rightarrow \ell^{FoE}_{1:T}\le \ell^i_{1:T}+O\big((\tfrac{1}{w^i})^{11}+T^{\frac{7}{8}}\sqrt{\ln T}\big),\ \textit{and}$$

$$(ii)\ B_t=t^{\frac{1}{16}}, \tau^i=\lceil (w^i)^{-16}\rceil \\ \Rightarrow \ell^{FoE}_{1:T}\le \ell^i_{1:T}+O\big((\tfrac{1}{w^i})^{22}+T^{\frac{7}{8}}\sqrt{\ln T}\big).$$

Corresponding assertions are true for the expectation (compare Corollary 8). In both cases (bounded and growing B_t) FoE is asymptotically optimal w.r.t. each expert: $\limsup_{T\to\infty}\frac{1}{T}\big(\ell^{FoE}_{1:T}-\ell^i_{1:T}\big)\le 0$ *a.s. for all i.*

Proof. Let $\eta_t=t^{-\frac{3}{4}}$, $\gamma_t=t^{-\frac{1}{4}}$, and $\delta_T=T^{-2}$. For $\tau^i=\lceil (w^i)^{-\alpha}\rceil$ and $B_t=t^\beta$, we have $w^*_T=\min\{w^i: T\ge \lceil (w^i)^{-\alpha}\rceil\}\ge\min\{w^i: T^{-\frac{1}{\alpha}}\le w^i\rceil\}\ge T^{-\frac{1}{\alpha}}$ and

$$\sum_{t=1}^{\tau^i-1}\hat B_t\le(\tau^i-1)\hat B_{\tau^i-1}\le\frac{(w^i)^{-\alpha}B_{\tau^i-1}}{\gamma_{\tau^i-1}w^*_{\tau^i-1}}\le\frac{(w^i)^{-\alpha}(w^i)^{-\alpha\beta}}{(w^i)^{\frac{\alpha}{4}}w^i}$$

(observe $w^*_{\tau^i-1}\ge(\tau^i-1)^{-\frac{1}{\alpha}}\ge(w^i)^{(-\alpha)(-\frac{1}{\alpha})}$). Then set $\alpha=8$, $\beta=0$, for (i) and $\alpha=16$, $\beta=\frac{1}{16}$ for (ii). Asymptotic optimality is shown as in Corollary 8. □

5. Active Expert Problems and a Universal Master Algorithm

If the adversary's goal is just to maximize our (expected) regret, then it is well known what he can achieve (at least for uniform prior, see e.g. the lower bound in Cesa-Bianchi et al., 1997; Auer et al., 2002). We are interested in different situations. An example is the repeated playing of the "Prisoner's dilemma" against the Tit-for-Tat[1] strategy (de Farias & Megiddo, 2004). If we use two strategies as experts, namely "always cooperate" and "always defect", then it is clear that always cooperating will have the better long-term reward. It is also clear that a standard expert advice or bandit master algorithm will not discover this, since it compares only the losses in one step, which are always lower for the defecting expert.

We therefore propose to give the control to a selected expert for *periods of increasing length*. Precisely, we introduce a new time scale $\tilde t$ at which we have single games with losses $\tilde\ell_{\tilde t}$. The master's time scale t does not coincide with $\tilde t$. Instead, at each t, the master gives control to the selected expert i for $\tilde T_t$ single games and receives loss $\ell^i_t=\sum_{\tilde t=\tilde t(t)}^{\tilde t(t)+\tilde T_t-1}\tilde\ell^i_{\tilde t}$. Assume that the game has bounded instantaneous losses $\tilde\ell^i_{\tilde t}\in[0,1]$. Then the master algorithm's instantaneous losses are bounded by $\tilde T_t$. We denote this algorithm by $FoE_{\tilde T}$ or $FoE^\tau_{\tilde T}$.

Corollary 13 *Assume $FoE_{\tilde T}$ (or $FoE^\tau_{\tilde T}$, respectively) plays a repeated game with bounded instantaneous losses $\tilde\ell^i_{\tilde t}\in[0,1]$. Let the exploration and learning rates be $\gamma_t=t^{-\frac{1}{4}}$ and $\eta_t=t^{-\frac{3}{4}}$. In case of uniform prior, choose $\tilde T_t=\lfloor t^{\frac{1}{8}}\rfloor$ ($\tau^i\equiv 0$). In case of arbitrary prior let $\tilde T_t=\lfloor t^{\frac{1}{16}}\rfloor$ and $\tau^i=\lceil (w^i)^{-16}\rceil$. Then for all experts i and all $\tilde T$, suppressing the dependence on the prior of expert i, we have*

$$\ell^{FoE_{\tilde T}}_{1:\tilde T}\ \le\ \ell^i_{1:\tilde T}+O(\tilde T^{\frac{9}{10}})\ \textit{w.p. } 1-\tilde T^2\ \textit{and}$$

$$\mathbf{E}\ell^{FoE_{\tilde T}}_{1:\tilde T}\ \le\ \ell^i_{1:\tilde T}+O(\tilde T^{\frac{9}{10}}).$$

Consequently, $\limsup_{T\to\infty}(\ell^{FoE_{\tilde T}}_{1:\tilde T}-\ell^i_{1:\tilde T})/\tilde T\le 0$ *almost surely. The rate of convergence is at least $\tilde T^{-\frac{1}{10}}$. The same assertions hold for $FoE^\tau_{\tilde T}$.*

Proof. This follows from changing the time scale from t to $\tilde t$ in Corollaries 8 and 12: $\tilde t$ is of order $t^{1+\frac{1}{8}}$ in the uniform case and $t^{1+\frac{1}{16}}$ in the general case. Then the bounds are $\tilde T^{\frac{8}{9}}\sqrt{\ln\tilde T}$ in the former and $\tilde T^{\frac{15}{17}}\sqrt{\ln\tilde T}$ in the latter case. Both are upper bounded by $\tilde T^{\frac{9}{10}}$. □

Broadly spoken, this means that $FoE_{\tilde T}$ performs asymptotically as well as the best expert. Asymptotic guarantees for the Strategic Experts Algorithm have been derived by de Farias and Megiddo. Our results approve upon this by providing a rate of convergence. One can give further corollaries, e.g. in terms of flexibility as defined by de Farias and Megiddo.

It is also possible to specify a *universal* experts algorithm. To this aim, let expert i be derived from the ith program p^i of some fixed universal Turing machine. The ith program can be well-defined, e.g. by representing programs as binary strings and lexicographically ordering them (Hutter, 2004). Before the expert is consulted, the relevant input is written to the input tape of the corresponding program. If the program halts, the appropriate number of first bits is interpreted as the expert's recommendation. E.g. if the decision is binary, then the first bit suffices. (If the program does not halt, we may for well-definedness just fill its output tape with zeros.) Each expert is assigned a prior weight by $w^i=2^{-\mathrm{length}(p^i)}$, where $\mathrm{length}(p^i)$ is the length of the corresponding program and we assume the program tape to be binary. This construction parallels the definition of Solomonoff's *universal*

[1] In the prisoner's dilemma, two players both decide independently if thy are *cooperating (C)* or *defecting (D)*. If both play C, they get both a small loss, if both play D, they get a large loss. However, if one plays C and one D, the cooperating player gets a very large loss and the defecting player no loss at all. Thus defecting is a *dominant* strategy. A Tit-for-Tat player play C in the first move and afterwards the opponent's respective preceding move.

prior (1978). This has been used to define a universal agent AIXI in a quite different way by Hutter (2004). Note that like the universal prior and AIXI, our universal agent is not computable, since we cannot check if a program halts. It is however straightforward to impose a bound on the computation time which for instance increases rapidly in t. If used with computable experts, the algorithm is computationally feasible. The universal master algorithm performs well with respect to *any* computable strategy.

Corollary 14 *Assume the universal set of experts specified in the last paragraph. If FoE$^{\tau}_{T}$ is applied with* $\gamma_t = t^{-\frac{1}{4}}$, $\eta_t = t^{-\frac{3}{4}}$, $\tilde{T}_t = \lfloor t^{\frac{1}{16}} \rfloor$, *and* $\tau^i = \lceil (w^i)^{-16} \rceil$, *then it performs asymptotically at least as good as* any computable expert i. *The rate of convergence is exponential in the complexity* k^i *and proportional to* $\tilde{T}^{-\frac{1}{16}}$.

6. Discussion

For large or infinite expert classes, the bounds we have proven are irrelevant in practice, although asserting almost sure optimality and even a convergence rate: the exponential of the complexity is far too huge. Imagine for instance a moderately complex task and some good strategy, which can be coded with mere 500 bits. Then its weight is 2^{-500}, a constant which is not distinguishable from zero in all practical situations. Thus, it seems that the bounds can be relevant at most for small expert classes with uniform prior. This is a general shortcoming of bandit experts algorithms: For uniform prior a lower bound on the expected loss which is linear in $\sqrt{n}$ has been proven (Auer et al., 2002).

If the bounds are not practically relevant, maybe the algorithms are so? We leave this interesting question unanswered. Intuitively, it might seem that the algorithms proposed here are too much tailored towards worst-case bounds and fully adversarial setups. For example, the exploration rate of $t^{-\frac{1}{4}}$ is quite high. Master algorithms which are less "cautious" might perform better for many practical problems. Finally, it would be nice to investigate the differences between the proposed expert style approach and other definitions of universal agents, such as by Hutter (2004).

Acknowledgement: This work was supported by SNF grant 2100-67712.02.

References

Auer, P., Cesa-Bianchi, N., Freund, Y., & Schapire, R. E. (1995). Gambling in a rigged casino: The adversarial multi-armed bandit problem. *Proc. 36th Annual Symposium on Foundations of Computer Science (FOCS 1995)* (pp. 322–331).

Auer, P., Cesa-Bianchi, N., Freund, Y., & Schapire, R. E. (2002). The nonstochastic multiarmed bandit problem. *SIAM Journal on Computing*, *32*, 48–77.

Cesa-Bianchi et al., N. (1997). How to use expert advice. *Journal of the ACM*, *44*, 427–485.

Cesa-Bianchi, N., Lugosi, G., & Stoltz, G. (2004). *Regret minimization under partial monitoring* (Technical Report).

de Farias, D. P., & Megiddo, N. (2004). How to combine expert (and novice) advice when actions impact the environment? In S. Thrun, L. Saul and B. Schölkopf (Eds.), *Advances in neural information processing systems 16*. Cambridge, MA: MIT Press.

Hannan, J. (1957). Approximation to Bayes risk in repeated plays. In M. Dresher, A. W. Tucker and P. Wolfe (Eds.), *Contributions to the theory of games 3*, 97–139. Princeton University Press.

Helmbold, D., & Panizza, S. (1997). Some label efficient learning results. *Proceedings of the tenth annual conference on Computational learning theory* (pp. 218–230). Nashville, Tennessee, United States: ACM Press.

Hutter, M., & Poland, J. (2004a). *Adaptive online prediction by following the perturbed leader* (Technical Report IDSIA-30-04).

Hutter, M., & Poland, J. (2004b). Prediction with expert advice by following the perturbed leader for general weights. *International Conference on Algorithmic Learning Theory (ALT)* (pp. 279–293).

Hutter, M. (2004). *Universal artificial intelligence: Sequential decisions based on algorithmic probability.* Berlin: Springer.

Kalai, A., & Vempala, S. (2003). Efficient algorithms for online decision. *Proc. 16th Annual Conference on Learning Theory (COLT-2003)* (pp. 506–521). Berlin: Springer.

Littlestone, N., & Warmuth, M. K. (1989). The weighted majority algorithm. *30th Annual Symposium on Foundations of Computer Science* (pp. 256–261). Research Triangle Park, North Carolina.

McMahan, H. B., & Blum, A. (2004). Online geometric optimization in the bandit setting against an adaptive adversary. *17th Annual Conference on Learning Theory (COLT)* (pp. 109–123). Springer.

Solomonoff, R. J. (1978). Complexity-based induction systems: comparisons and convergence theorems. *IEEE Trans. Information Theory*, *IT-24*, 422–432.

Strong Asymptotic Assertions for Discrete MDL in Regression and Classification

Jan Poland JAN@IDSIA.CH
Marcus Hutter MARCUS@IDSIA.CH
IDSIA, Galleria 2, CH-6928 Manno-Lugano, Switzerland

Abstract

We study the properties of the MDL (or maximum penalized complexity) estimator for Regression and Classification, where the underlying model class is countable. We show in particular a finite bound on the Hellinger losses under the only assumption that there is a "true" model contained in the class. This implies almost sure convergence of the predictive distribution to the true one at a fast rate. It corresponds to Solomonoff's central theorem of universal induction, however with a bound that is exponentially larger.

1. Introduction

Bayesian methods are popular in Machine Learning. So it is natural to study their predictive properties: How do they behave asymptotically for increasing sample size? Are loss bounds obtainable, either for certain interesting loss functions or even for more general classes of loss functions?

In this paper, we consider the two maybe most important Bayesian methods for prediction in the context of regression and classification. The first one is *marginalization*: Given some data and a model class, obtain a predictive model by integrating over the model class. This *Bayes mixture* is "ideal" Bayesian prediction in many respects, however in many cases it is computationally untractable. Therefore, a commonly employed method is to compute a *maximum penalized complexity* or *maximum a posteriori (MAP)* or *minimum description length (MDL)* estimator. This predicts according to the "best" model instead of a mixture. The MDL principle is important for its own sake, not only as approximation of the Bayes mixture.

Most work on Bayesian prediction has been carried out for *continuous* model classes, e.g. classes with one free parameter $\vartheta \in \mathbb{R}^d$. While the predictive properties of the Bayes mixture are excellent under mild conditions (Clarke & Barron, 1990; Hutter, 2003b; Ghosal et al., 2000), corresponding MAP or MDL results are more difficult to establish. For MDL in the strong sense of *description length*, the parameter space has to be discretized appropriately (and dynamically with increasing sample size) (Rissanen, 1996; Barron et al., 1998; Barron & Cover, 1991). A MAP estimator on the other hand can be very bad in general. In statistical literature, some important work has been performed on the asymptotical discovery of the true parameter, e.g. (Cam & Yang, 2000). This can only hold if each model occurs no more than once in the class. Thus it is violated e.g. in the case of an artificial neural network, where exchanging two hidden units in the same layer does not alter the network behavior.

In the case of *discrete* model classes, both loss bounds and asymptotic assertions for the Bayes mixture are relatively easy to prove, compare Theorem 2. In (Poland & Hutter, 2004a), corresponding results for MDL were shown. The setting is sequence prediction but otherwise very general. The only assumption necessary is that the true distribution is contained in the model class. Assertions are given directly for the predictions, thus there is no problem of possibly undistinguishable models. In order to prove that the MDL estimator (precisely, the *static* MDL estimator in terms of (Poland & Hutter, 2004a)) has good predictive properties, we introduce an intermediate step and show first the predictive properties of *dynamic MDL*, where a new MDL estimator is computed for each possible next observation.

In this paper, we will derive analogous results for regression and classification. While results for classification can be generalized from sequence prediction by conditionalizing everything to the input, regression is technically more difficult. Therefore the next section, which deals with the regression setup, covers the major part of the paper. Instead of the popular Euclidian and Kullback-Leibler distances for measur-

ing prediction quality we need to exploit the Hellinger distance. We show that online MDL converges to the true distribution in mean Hellinger sum, which implies "rapid" convergence with probability one. Classification is briefly discussed in Section 3, followed by a discussion and conclusions in Section 4.

2. Regression

We neglect computational aspects and study the properties of the *optimal* Bayes mixture and MDL predictors. When a new sample is observed, the estimator is updated. Thus, regression is considered in an *online framework*: The first input x_1 is presented, we predict the output y_1 and then observe its true value, the second input x_2 is presented and so on.

Setup. Consider a regression problem with arbitrary domain $\mathcal{X}$ (we need no structural assumptions at all on $\mathcal{X}$) and co-domain $\mathcal{Y} = \mathbb{R}$. The task is to learn/fit/infer a function $f : \mathcal{X} \to \mathcal{Y}$, or more generally a conditional probability density $\nu(y|x)$, from data $\{(x_1, y_1), ..., (x_n, y_n)\}$. Formally, we are given a countable class $\mathcal{C}$ of models that are functions ν from $\mathcal{X}$ to *uniformly bounded probability densities* on $\mathbb{R}$. That is, $\mathcal{C} = \{\nu_i : i \geq 1\}$, and there is some $C > 0$ such that

$$0 \leq \nu_i(y|x) \leq C \text{ and } \int_{-\infty}^{\infty} \nu_i(y|x)dy = 1 \quad (1)$$
$$\text{for all } i \geq 1,\ x \in \mathcal{X},\ \text{and } y \in \mathcal{Y}.$$

Each ν induces a probability density on $\mathbb{R}^n$ for n-tuples $x_{1:n} \in \mathcal{X}^n$ by $\nu(y_{1:n}|x_{1:n}) = \prod_{t=1}^n \nu(y_t|x_t)$. The notation $x_{1:n}$ for n-tuples is common in sequence prediction. Each model $\nu \in \mathcal{C}$ is associated with a *prior weight* $w_\nu > 0$. The logarithm $\log_2 w_\nu$ has often an interpretation as model *complexity*. We require $\sum_\nu w_\nu = 1$. Then by the Kraft inequality, one can assign to each model $\nu \in \mathcal{C}$ a prefix-code of length $\lceil \log_2 w_\nu \rceil$.

We assume that an infinite stream of data $(x_{1:\infty}, y_{1:\infty})$ is generated as follows: Each x_t may be produced by an arbitrary mechanism, while y_t is sampled from a *true distribution* μ conditioned on x_t. In order to obtain strong convergence results, we will require that $\mu \in \mathcal{C}$.

Example 1 Take $\mathcal{X} = \mathbb{R}$ and $\mathcal{C}_\sigma^{\text{lin1}} \cong \{ax + b + N(0, \sigma^2) : a, b \in \mathbb{Q}\}$ to be the class of linear regression models with rational coefficients a, b, and independent Gaussian noise of fixed variance $\sigma^2 > 0$. That is, $\mathcal{C}_\sigma^{\text{lin1}} = \{\nu^{a,b,\sigma} : a, b \in \mathbb{Q}\}$, where

$$\nu^{a,b,\sigma}(x, y) = \phi_{\sigma^2}(y - ax - b) = \tfrac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{1}{2\sigma^2}(y-ax-b)^2}.$$

Alternatively, you may consider the class $\mathcal{C}_{\geq\sigma_0}^{\text{lin1}} = \{\nu^{a,b,\sigma} : a, b, \sigma \in \mathbb{Q}, \sigma \geq \sigma_0\}$ for some $\sigma_0 > 0$, where also the noise amplitude is part of the models. In the following, we also discuss how to admit degenerate Gaussians that are point measures such as $\mathcal{C}_{\geq 0}^{\text{lin1}}$.

The setup (1) guarantees that all subsequent MDL estimators [(9) and (10)] exist. However, our results and proofs generalize in several directions. First, for the co-domain $\mathcal{Y}$ we may choose any σ-finite measure space instead of $\mathbb{R}$, since we need only Radon-Nikodym densities below. Second, the uniformly boundedness condition can be relaxed, if the MDL estimators still exist. This holds for example for the class $\mathcal{C}_{\geq 0}^{\text{lin1}}$ (see the preceding example), if the definition of the MDL estimators is adapted appropriately (see footnote 2 on page 4). Third, the results remain valid for semimeasures with $\int \nu \leq 1$ instead of measures and $\sum w_\nu \leq 1$, which is however not very relevant for regression (but for universal sequence prediction). In order to keep things simple, we develop all results on the basis of (1). Note finally that the models in $\mathcal{C}$ may be time-dependent, and we need not even make this explicit, since the time can be incorporated into $\mathcal{X}$ ($x_t = (x'_t, t) \in \mathcal{X}' \times \mathbb{N} = \mathcal{X}$). In this way we may also make the models depend on the actual past outcome, if this is desired ($x_t = (x'_{1:t} y_{1:t-1}) \in \mathcal{X}'^* \times \mathcal{Y}^* = \mathcal{X}$).

The case of independent Gaussian noise as in Example 1 is a particularly important one. We therefore introduce the family

$$\mathcal{F}_{\geq}^{\text{Gauss}} = \Big\{\mathcal{C} = \{\nu_i, \sigma_i\}_{i=1}^{\infty} : \nu_i(x, y) = \phi_{\sigma_i^2}\big(y - f_i(x)\big), \sigma_i \geq \sigma_0 > 0, f_i : \mathcal{X} \to \mathbb{R}\Big\}. \quad (2)$$

of all countable regression model classes with lower bounded Gaussian noise. Clearly, $\mathcal{C}_\sigma^{\text{lin1}}, \mathcal{C}_{\geq\sigma_0}^{\text{lin1}} \in \mathcal{F}_{\geq}^{\text{Gauss}}$ is satisfied. Similarly $\mathcal{F}^{\text{Gauss}} \supset \mathcal{F}_{\geq}^{\text{Gauss}}$ denotes the corresponding family without lower bound on σ_i. Then $\mathcal{C}_{\geq 0}^{\text{lin1}} \in \mathcal{F}^{\text{Gauss}} \setminus \mathcal{F}_{\geq}^{\text{Gauss}}$.

We define the *Bayes mixture*, which for each $n \geq 1$ maps an n-tuple of inputs $x_{1:n} \in \mathcal{X}^n$ to a probability density on $\mathbb{R}^n$:

$$\xi(y_{1:n}|x_{1:n}) = \sum_{\nu\in\mathcal{C}} w_\nu \nu(y_{1:n}|x_{1:n}) = \sum_{\nu\in\mathcal{C}} w_\nu \prod_{t=1}^{n} \nu(y_t|x_t) \quad (3)$$

(recall $\sum_\nu w_\nu = 1$). Hence, the Bayes mixture *dominates* each ν by means of $\xi(\cdot|x_{1:n}) \geq w_\nu \nu(\cdot|x_{1:n})$ for all $x_{1:n}$. For $\nu \in \mathcal{C}$ and $x_n \in \mathcal{X}$, the ν-prediction of $y_n \in \mathbb{R}$, that is the $n u$-probability density of observing y_n, is

$$\nu(y_n|x_{1:n}, y_{<n}) = \nu(y_n|x_n).$$

This is independent of the history $(x_{<n}, y_{<n}) = (x_{1:n-1}, y_{1:n-1})$. In contrast, the *Bayes mixture pre-*

diction or regression, which is also a measure on $\mathbb{R}$, depends on the history:

$$\xi(y_n|x_{1:n},y_{<n}) = \frac{\xi(y_{1:n}|x_{1:n})}{\xi(y_{<n}|x_{<n})} = \frac{\sum_\nu w_\nu \prod_{t=1}^n \nu(y_t|x_t)}{\sum_\nu w_\nu \prod_{t=1}^{n-1} \nu(y_t|x_t)}. \tag{4}$$

This is also known as *marginalization*. Observe that the denominator in (4) vanishes only on a set of μ-measure zero, if the true distribution μ is contained in $\mathcal{C}$. Under condition (1), the Bayes mixture prediction is uniformly bounded. It can be argued intuitively that in case of unknown $\mu \in \mathcal{C}$ the Bayes mixture is the best possible model for μ. Formally, its predictive properties are excellent:

Theorem 2 *Let $\mu \in \mathcal{C}$, $n \geq 1$, and $x_{1:n} \in \mathcal{X}^n$, then*

$$\sum_{t=1}^n \mathbf{E} \int \Big(\sqrt{\mu(y_t|x_{1:t},y_{<t})} - \sqrt{\xi(y_t|x_{1:t},y_{<t})}\Big)^2 dy_t \leq \ln w_\mu^{-1}. \tag{5}$$

$\mathbf{E}$ denotes the expectation with respect to the true distribution μ. Hence in this case we have $\mathbf{E}\ldots = \int \ldots \mu(dy_{<t})$. The integral expression is also known as *square Hellinger distance*. It will emerge as a main tool in the subsequent proofs. So the theorem states that on any input sequence $x_{<\infty}$ the expected cumulated Hellinger divergence of μ and the Bayes mixture prediction is bounded by $\ln w_\mu^{-1}$. A closely related result was discovered by Solomonoff ((Solomonoff, 1978)) for universal sequence prediction, a "modern" proof can be found in (Hutter, 2003a). This proof can be adapted in our regression framework. Alternatively, it is not difficult to give a proof in a few lines analogous to (14) and (15) by using (12).

We introduce the term *convergence in mean Hellinger sum (i.m.H.s.)* for bounds like (5): For some predictive density ψ, the ψ-predictions converge to the μ-predictions i.m.H.s. on a sequence of inputs $x_{<\infty} \in \mathcal{X}^\infty$, if there is $R > 0$ such that $H^2_{x_{<\infty}}(\mu,\psi) \leq R$, where

$$H^2_{x_{<\infty}}(\mu,\psi) = \sum_{t=1}^\infty \mathbf{E}[h_t^2] \text{ with} \tag{6}$$
$$h_t^2 = \int \Big(\sqrt{\mu(y_t|x_{1:t},y_{<t})} - \sqrt{\psi(y_t|x_{1:t},y_{<t})}\Big)^2 dy_t.$$

Convergence i.m.H.s. is a very strong convergence criterion. It asserts a finite expected cumulative Hellinger loss in the first place. If the co-domain $\mathcal{Y}$ is finite as for classification (see Section 3), then convergence i.m.H.s. implies almost sure (a.s.) convergence of the (finitely many) posterior probabilities. For regression, the situation is more complex, since the posterior probabilities are densities, i.e. Banach space valued. Here, convergence i.m.H.s. implies that with μ-probability one the square roots of the predictive densities converge to the square roots of the μ-densities in $L^2(\mathbb{R})$ (endowed with the Lebesgue measure). In other words, h_t^2 converges to zero a.s.:

$$\begin{aligned} \mathbf{P}\Big(\exists t \geq n : h_t^2 \geq \varepsilon\Big) &= \mathbf{P}\Big(\bigcup_{t\geq n} \{h_t^2 \geq \varepsilon\}\Big) \\ &\leq \sum_{t\geq n} \mathbf{P}\big(h_t^2 \geq \varepsilon\big) \\ &\leq \frac{1}{\varepsilon}\sum_{t=n}^\infty \mathbf{E} h_t^2 \overset{n\to\infty}{\longrightarrow} 0 \end{aligned} \tag{7}$$

holds by the union bound, the Markov inequality for all $\varepsilon > 0$, and $H^2_{x_{<\infty}} < \infty$, respectively, where $\mathbf{P}$ is the μ-probability. If the densities are uniformly bounded, then also the differences of the densities (as opposed to the difference of the square roots) converge to zero:

$$\psi(y_t|x_{1:t},y_{<t}) - \mu(y_t|x_{1:t},y_{<t}) \overset{t\to\infty}{\longrightarrow} 0 \quad \text{in } L^2(\mathbb{R}) \text{ a.s.}$$

Moreover, the finite bound on the cumulative Hellinger distances can be interpreted as a convergence rate. Compare the parallel concept "convergence in mean sum" (Hutter, 2003b; Poland & Hutter, 2004a).

MDL Predictions. In many cases, the Bayes mixture is not only intractable, but even hard to approximate. So a very common substitute is the (ideal) MDL[1] estimator, also known as maximum a posteriori (MAP) or maximum complexity penalized likelihood estimator. Given a model class $\mathcal{C}$ with weights (w_ν) and a data set $(x_{1:n},y_{1:n})$, we define the two-part MDL estimator as

$$\begin{aligned} \nu^* = \nu^*_{(x_{1:n},y_{1:n})} &= \arg\max_{\nu\in\mathcal{C}}\{w_\nu \nu(y_{1:n}|x_{1:n})\} \text{ and} \\ \varrho(y_{1:n}|x_{1:n}) &= \max_{\nu\in\mathcal{C}}\{w_\nu \nu(y_{1:n}|x_{1:n})\} \\ &= w_{\nu^*}\nu^*(y_{1:n}|x_{1:n}). \end{aligned} \tag{8}$$

Note that we define both the model ν^* which is the MDL estimator and its weighted density ρ. In our setup (1), the MDL estimator is well defined, since

[1] There is some disagreement about the exact meaning of the term MDL. Sometimes a specific prior is associated with MDL, while we admit arbitrary priors. More importantly, when coding some data x, one can exploit the fact that once the model ν^* is specified, only data which lead to the maximizing element ν^* need to be considered. This allows for a shorter description than $\log_2\nu^*(x)$. Nevertheless, the *construction principle* is commonly termed MDL, compare for instance the "ideal MDL" in (Vitányi & Li, 2000).

all maxima exist[2]. Moreover, $\varrho(\cdot|x_{1:n})$ is a density but its integral is less than 1 in general. We have $\varrho(\cdot|x_{1:n}) \geq w_\nu \nu(\cdot|x_{1:n})$, so like ξ, ϱ dominates each $\nu \in \mathcal{C}$. Also, $\varrho(\cdot|x_{1:n}) \leq \xi(\cdot|x_{1:n})$ is clear by definition. If we use ν^* for (sequential online) prediction, this is the *static MDL prediction*:

$$\varrho^{\text{static}}(y_n|x_{1:n}, y_{<n}) = \nu^*_{(x_{<n}, y_{<n})}(y_n|x_n). \qquad (9)$$

This is the common way of using MDL for prediction. Clearly, the static MDL predictor is a probability density on $\mathbb{R}$. Alternatively, we may compute the MDL estimator for each possible y_n separately, arriving at the *dynamic* MDL predictor:

$$\varrho(y_n|x_{1:n}, y_{<n}) = \frac{\varrho(y_{1:n}|x_{1:n})}{\varrho(y_{<n}|x_{<n})}. \qquad (10)$$

We have $\varrho(y_n|x_{1:n}, y_{<n}) \leq \nu^*_{(x_{1:n}, y_{1:n})}(y_n|x_n)$ for each y_n, which shows that under condition (1) the dynamic MDL predictor is uniformly bounded. On the other hand, $\varrho(y_n|x_{1:n}, y_{<n}) \geq \nu^*_{(x_{<n}, y_{<n})}(y_n|x_n)$ holds, so the dynamic MDL predictor may be a density with mass more than 1. Hence we must usually *normalize* it for predicting:

$$\bar{\varrho}(y_n|x_{1:n}, y_{<n}) = \frac{\varrho(y_{1:n}|x_{1:n})}{\int \varrho(y_{1:n}|x_{1:n}) dy_n}. \qquad (11)$$

Both fractions in (10) and (11) are well-defined except for a set of measure zero. Dynamic MDL predictions are in a sense computationally (almost) as expensive as the full Bayes mixture.

Convergence Results. Our principal aim is to prove predictive properties of *static* MDL, since this is the practically most relevant variant. To this end, we first need to establish corresponding results for the dynamic MDL. Precisely, the following holds.

Theorem 3 *Assume the setup (1). If $\mu \in \mathcal{C}$, where μ is the true distribution, and $H^2_{x_{<\infty}}(\cdot,\cdot)$ is defined as in (6), then for all input sequences $x_{<\infty} \in \mathcal{X}^\infty$ we have*

$$(i) \qquad H^2_{x_{<\infty}}(\mu, \bar{\varrho}) \leq w_\mu^{-1} + \ln w_\mu^{-1},$$
$$(ii) \qquad H^2_{x_{<\infty}}(\bar{\varrho}, \varrho) \leq 2w_\mu^{-1}, \text{ and}$$
$$(iii) \qquad H^2_{x_{<\infty}}(\varrho, \varrho^{\text{static}}) \leq 3w_\mu^{-1}.$$

[2]For a model class with Gaussian noise $\mathcal{C} \in \mathcal{F}^{\text{Gauss}}$ (2), we may dispose of the uniform boundedness condition and admit e.g. also $\mathcal{C}^{\text{lin}1}_{>0}$. In order to compute the MDL estimator, we must then first check if there is nonzero mass concentrated on $(x_{1:n}, y_{1:n})$, in which case the mass is even one and the corresponding model with the largest weight is chosen. Otherwise, the MDL estimator is chosen according to the maximum penalized density. All results and proofs below generalize to this case.

Since the triangle inequality holds for $\sqrt{H^2_{x_{<\infty}}(\cdot,\cdot)}$, we immediately conclude:

Corollary 4 *Given the setup (1) and $\mu \in \mathcal{C}$, then all three predictors $\bar{\varrho}$, ϱ, and ϱ^{static} converge to the true density μ in mean Hellinger sum, for any input sequence $x_{<\infty}$. In particular, we have $H^2(\mu, \varrho^{\text{static}}) \leq 21 w_\mu^{-1}$.*

We will only prove (i) of Theorem 3 here. The proofs of (ii) and (iii) can be similarly adapted from (Poland & Hutter, 2004a, Theorems 10 and 11), since the Hellinger distance is bounded by the absolute distance: $\int \left(\sqrt{\mu(y)} - \sqrt{\nu(y)}\right)^2 dy \leq \int \left|\mu(y) - \nu(y)\right| dy$ follows from $(\sqrt{a} - \sqrt{b})^2 \leq |a-b|$ for any $a, b \in \mathbb{R}$ (this shows also that the integral h_t^2 in (6) exists). In order to show (i), we make use of the fact that the squared Hellinger distance is bounded by the *Kullback-Leibler divergence*:

$$\int \left(\sqrt{\mu(y)} - \sqrt{\nu(y)}\right)^2 dy \leq \int \mu(y) \ln \frac{\mu(y)}{\nu(y)} dy \qquad (12)$$

for any two probability densities μ and ν on $\mathbb{R}$ (see e.g. (Borovkov & Moullagaliev, 1998, p. 178)). So we only need to establish the corresponding bound for the Kullback-Leibler divergence and show

$$\begin{aligned} D_x(\mu\|\bar{\varrho}) &:= \sum_{t=1}^n \mathbf{E} \int \mu(y_t|x_{1:t}, y_{<t}) \ln \frac{\mu(y_t|x_{1:t}, y_{<t})}{\bar{\varrho}(y_t|x_{1:t}, y_{<t})} dy_t \\ &\leq w_\mu^{-1} + \ln w_\mu^{-1} \end{aligned} \qquad (13)$$

for all $n \geq 1$. In the following computation, we take $x_{<\infty}$ to be fixed and suppress it in the notation, writing e.g. $\mu(y_t|y_{<t})$ instead of $\mu(y_t|x_{1:t}, y_{<t})$. Then

$$\begin{aligned} D_x(\mu\|\bar{\varrho}) &= \sum_t \mathbf{E} \ln \frac{\mu(y_t|y_{<t})}{\bar{\varrho}(y_t|y_{<t})} \qquad (14) \\ &= \sum_t \mathbf{E} \left[\ln \frac{\mu(y_t|y_{<t})}{\varrho(y_t|y_{<t})} + \ln \frac{\int \varrho(y_{1:t}) dy_t}{\varrho(y_{<t})} \right]. \end{aligned}$$

The first part of the last term is bounded by

$$\begin{aligned} \sum_t \mathbf{E} \ln \frac{\mu(y_t|y_{<t})}{\varrho(y_t|y_{<t})} &= \mathbf{E} \ln \prod_{t=1}^n \frac{\mu(y_t|y_{<t})}{\varrho(y_t|y_{<t})} \qquad (15) \\ &= \mathbf{E} \ln \frac{\mu(y_{1:n}|x_{1:n})}{\varrho(y_{1:n}|x_{1:n})} \\ &\leq \ln w_\mu^{-1}, \end{aligned}$$

since always $\frac{\mu}{\varrho} \leq w_\mu^{-1}$. For the second part, use $\ln u \leq u - 1$ to obtain

$$\mathbf{E} \ln \frac{\int \varrho(y_{1:t}) dy_t}{\varrho(y_{<t})}$$

$$\leq \sum_t \mathbf{E}\left[\frac{\int \varrho(y_{1:t})dy_t}{\varrho(y_{<t})} - 1\right]$$
$$= \int \frac{\mu(y_{<t})(\int \varrho(y_{1:t})dy_t - \varrho(y_{<t})}{\varrho(y_{<t})} dy_{<t}$$
$$\leq w_\mu^{-1}\left[\int \varrho(y_{1:t})dy_{1:t} - \int \varrho(y_{<t})dy_{<t}\right].$$

If this is summed over $t = 1 \ldots n$, the last term is telescoping. So using $\varrho(\emptyset) = \max_\nu w_\nu \geq 0$ and $\varrho \leq \xi$, we conclude

$$\begin{aligned}\sum_t \mathbf{E} \ln \frac{\int \varrho(y_{1:t})dy_t}{\varrho(y_{<t})} &\leq w_\mu^{-1}\left[\int \varrho(y_{1:n})dy_{1:n} - \varrho(\emptyset)\right] \\ &\leq w_\mu^{-1}\int \xi(y_{1:n})dy_{1:n} \qquad (16) \\ &= w_\mu^{-1}.\end{aligned}$$

Hence, (14), (15), and (16) show together (13). □

We may for example apply the result for the static predictions in a Gaussian noise class $\mathcal{C} \in \mathcal{F}^{\mathrm{Gauss}}$.

Corollary 5 *Let $\mathcal{C} \in \mathcal{F}^{\mathrm{Gauss}}_{\geq}$ [see (2)] then the mean and the variance of the static MDL predictions converge to their true values almost surely. The same holds for $\mathcal{C} \in \mathcal{F}^{\mathrm{Gauss}}$. In particular, if the variance of all models in $\mathcal{C}$ is the same value σ^2, then $\sum_t 2\big[1 - \exp(-\frac{(g^*(x_t|\ldots)-f(x_t))^2}{8\sigma^2})\big] \leq 21 w_\mu^{-1}$, where $f(x_t)$ is the mean value of the true distribution and $g^* = \arg\min_{f_i}\{\frac{1}{n-1}\sum_{t=1}^{n-1}(y_t - f_i(x_t))^2 + 2\sigma^2 \ln w_i^{-1}\}$ is the mean of the MDL predictor.*

For $\mathcal{C} \in \mathcal{F}^{\mathrm{Gauss}}_{\geq}$, almost sure convergence holds since otherwise the cumulative Hellinger distances would be infinite, see (7). This generalizes to $\mathcal{C} \in \mathcal{F}^{\mathrm{Gauss}}$; compare the footnote 2 on page 4. In the case of constant variance, the cumulative Hellinger distances can be explicitly stated as above. Note that since $1 - \exp\big(-\frac{(g^*(x_t|\ldots)-f(x_t))^2}{8\sigma^2}\big) \approx \frac{(g^*(x_t|\ldots)-f(x_t))^2}{8\sigma^2}$ for small $(g^*(x_t|\ldots) - f(x_t))^2$, this implies convergence of g^* to f faster than $O(\frac{1}{\sqrt{t}})$ if the convergence is monotone. Moreover, deviations of a fixed magnitude can only occur finitely often.

Compared with the bound for the Bayes mixture in Theorem 2, MDL bounds are exponentially larger. The bounds are sharp, as shown in (Poland & Hutter, 2004a, Example 9), this example may be also adapted to the regression framework.

3. Classification

The classification setup is technically easier, since only a finite co-domain $\mathcal{Y}$ has to be considered. Results corresponding to Theorem 3 and Corollary 4 follow analogously. Alternatively, one may conditionalize the results for sequence prediction in (Poland & Hutter, 2004a) with respect to the input sequence $x_{<\infty}$, arriving equally at the assertions for classification. The results in (Poland & Hutter, 2004a) are formulated in terms of mean (square) sum convergence instead of Hellinger sum convergence. On finite co-domain, these two convergence notions induce the same topology.

Theorem 6 *Let $\mathcal{X}$ be arbitrary and $\mathcal{Y}$ be a finite set of class labels. $\mathcal{C} = \{\nu_i : i \geq 1\}$ consists of classification models, i.e. for each $\nu \in \mathcal{C}$, $x \in \mathcal{X}$ and $y \in \mathcal{Y}$ we have $\nu(y|x) \geq 0$ and $\sum_y \nu(y|x) = 1$. Each model ν is associated with a prior weight $w_\nu > 0$, and $\sum_\nu w_\nu = 1$ holds. Let the MDL predictions be defined analogously to (8), (9) and (10) (the difference being that here probabilities are maximized instead of densities). Assume that $\mu \in \mathcal{C}$, where μ is the true distribution. Then for each $x_{<\infty} \in \mathcal{X}^\infty$,*

$$\sum_{t=1}^{\infty} \mathbf{E} \sum_{y\in\mathcal{Y}} \Big(\sqrt{\mu(y|x_t)} - \sqrt{\varrho^{\mathrm{static}}(y|x_{1:t}, y_{<t})}\Big)^2 \leq 21 w_\mu^{-1},$$
$$\sum_{t=1}^{\infty} \mathbf{E} \sum_{y\in\mathcal{Y}} \Big(\mu(y|x_t) - \varrho^{\mathrm{static}}(y|x_{1:t}, y_{<t})\Big)^2 \leq 21 w_\mu^{-1}$$

holds. Similar assertions are satisfied for the normalized and the un-normalized dynamic MDL predictor. In particular, the predictive probabilities of all three MDL predictors converge to the true probabilities almost surely.

The second bound on the quadratic differences is shown in (Poland & Hutter, 2004a). The assertions about almost sure convergence follows as in (7).

4. Discussion and Conclusions

We have seen that discrete MDL has good asymptotic predictive properties. On the other hand, the loss bounds for MDL are exponential compared to the Bayes mixture loss bound. This is no proof artifact, as examples are easily constructed where the bound is sharp (Poland & Hutter, 2004a).

This has an important implication for the practical use of MDL: One need to choose the underlying model class and the prior carefully. Then it can be expected that the predictions are good and converge fast: this is supported by theoretical arguments in (Rissanen, 1996; Poland & Hutter, 2004b). The Bayes mixture in contrast, which can be viewed as a very large (infinite) weighted committee, also converges rapidly with unfavorable model classes, but at higher computational expenses.

One might be interested in other loss functions than the Hellinger loss. For the classification case, a bound on the expected error loss (number of classification errors) of MDL may be derived with the techniques from (Hutter, 2003a), using the bound on the quadratic distance. (Hutter, 2003a) gives also bounds for *arbitrary* loss functions, however this requires a bound on the Kullback-Leibler divergence rather than the quadratic distance. Unfortunately, this does not hold for static MDL (Poland & Hutter, 2004a). For the regression setup, analysis of other, more general or even arbitrary loss functions is even more demanding and, as far as we know, open.

Considering only discrete model classes is certainly a restriction, since many models arising in science (e.g. physics or biology) are continuous. On the other hand there are arguments in favor of discrete classes. From a computational point of view they are definitely sufficient. Real computers may even treat only finite model classes. The class of all programs on a fixed universal Turing machine is countable. It may be related to discrete classes of stochastic models by the means of semimeasures, this is one of the central issues in Algorithmic Information Theory (Li & Vitányi, 1997).

References

Barron, A. R., & Cover, T. M. (1991). Minimum complexity density estimation. *IEEE Trans. on Information Theory*, *37*, 1034–1054.

Barron, A. R., Rissanen, J. J., & Yu, B. (1998). The minimum description length principle in coding and modeling. *IEEE Trans. on Information Theory*, *44*, 2743–2760.

Borovkov, A. A., & Moullagaliev, A. (1998). *Mathematical statistics.* Gordon & Breach.

Cam, L. L., & Yang, G. (2000). *Asymptotics in statistics.* Springer. 2nd edition.

Clarke, B. S., & Barron, A. R. (1990). Information-theoretic asymptotics of Bayes methods. *IEEE Trans. on Information Theory*, *36*, 453–471.

Ghosal, S., Gosh, J., & van der Vaart, A. (2000). Convergence rates of posterior distributions. *Ann. Statist.*, *28*, 500–531.

Hutter, M. (2003a). Convergence and loss bounds for Bayesian sequence prediction. *IEEE Trans. on Information Theory*, *49*, 2061–2067.

Hutter, M. (2003b). Optimality of universal Bayesian prediction for general loss and alphabet. *Journal of Machine Learning Research*, *4*, 971–1000.

Li, M., & Vitányi, P. M. B. (1997). *An introduction to Kolmogorov complexity and its applications.* Springer. 2nd edition.

Poland, J., & Hutter, M. (2004a). Convergence of discrete MDL for sequential prediction. *17th Annual Conference on Learning Theory (COLT)* (pp. 300–314).

Poland, J., & Hutter, M. (2004b). On the convergence speed of MDL predictions for Bernoulli sequences. *International Conference on Algorithmic Learning Theory (ALT)* (pp. 294–308).

Rissanen, J. J. (1996). Fisher Information and Stochastic Complexity. *IEEE Trans. on Information Theory*, *42*, 40–47.

Solomonoff, R. J. (1978). Complexity-based induction systems: comparisons and convergence theorems. *IEEE Trans. Information Theory*, *IT-24*, 422–432.

Vitányi, P. M., & Li, M. (2000). Minimum description length induction, Bayesianism, and Kolmogorov complexity. *IEEE Trans. on Information Theory*, *46*, 446–464.

Speaker Prediction based on Head Orientations

An Evaluation of Machine Learning and Human Performance

Rutger Rienks RIENKS@EWI.UTWENTE.NL
Ronald Poppe POPPE@EWI.UTWENTE.NL
Mannes Poel MPOEL@EWI.UTWENTE.NL
Human Media Interaction Group, Department of Electrical Engineering, Mathematics and Computer Science
University of Twente, PO Box 217, 7500 AE, Enschede, The Netherlands

Abstract

To gain insight into gaze behavior in meetings, this paper compares the results from a Naive Bayes classifier, Neural Networks and humans on speaker prediction in four-person meetings given solely the azimuth head angles. The Naive Bayes classifier scored 69.4% correctly, Neural Networks 62.3% and humans only 37.7%. None of the classifiers was able to generalize over meetings. We show that there are strong indications that human specific gaze behavior influences the fact that the models do not generalize. Additionally, we show that for all classifiers the performance of the prediction in the beginning and at the end of a speaker turn is worse than halfway through the speaker turn.

1. Introduction

Nowadays a lot of research is being done on machine processing and interpretation of signals from people or other elements present in an observed environment. Smart rooms are widely used as test environments for research in this area. Sensors in the room can capture information which can be fused, manipulated and augmented in order to achieve interpretation at desired levels (Reidsma et al., 2004).

Currently smart meeting rooms are being used for all kinds of research purposes in e.g. the AMI project, the CMU meeting Room Project and the NIST Meeting Room project. One can think of a smart meeting room with a system that creates notes by understanding speech, creates summaries (Mani & Maybury, 1999), or automatically switches microphones on and off. EasyMeeting (Chen & Perich, 2004) is an example of a system that can provide relevant services and information to meeting participants.

State of the art in computer graphics and animation of embodied agents allows us to build quite realistic 3D virtual environments in which real humans can meet virtual human-like avatars (Vilhjalmsson & Cassell, 1998; Nijholt, 2004). These virtual environments can be used as a test bed for data visualization and for studying human perception and interpretation of meeting situations.

In this paper we examine the performance of humans and machine learning techniques on the task of speaker prediction from horizontal head orientation angles (azimuth). Gaze is not solely determined by the head orientation but also by the direction of eye gaze. It has, however, been shown that there is a high correlation between gaze and head orientation (more than 85%) (Stiefelhagen, 2002).

2. Research aim

Within a meeting context, our aim is to gain insight into the nature of human gaze behavior. We can use this knowledge for both generation of gaze behavior in virtual meetings and the extraction of useful information from gaze behavior such as the speaker, the addressee and the focus of attention.

Imagine a virtual chairman who understands the meeting sufficiently to be able to structure the meeting according to an agenda by giving appropriate turns, interrupting if someone speaks for too long and keeping participants focused when they seem distracted. This might sound far-fetched, but when a machine understands where someone is looking during a meeting, it might conclude based on its models that the person is focused on the object in line with the head orientation. If a non-speaker is always looking at the ceiling instead of at the speaker, this might reveal something about his attention level or even about the personality of that person.

Autonomous agents can use the derived gaze behavior models in combination with their own beliefs, desires, intentions and emotions (Wright, 1997) to become 'aware' of the perceived situation and even to act according to the models. Current applications of such models lead to increased appreciation of such agents (van Es et al., 2002), since they become more lively. This directly improves remote communication (Garau et al., 2001).

2.1. Gaze behavior

In general it is believed that gaze can bear a conversational function. When someone is looked at, the person who is looking might expect a reaction from the other, either visual or vocal. According to Kendon (Kendon, 1967) gaze serves four functions: visual feedback, regulation of conversation flow, communication of emotions and relationships, and improvement of concentration by restricting visual input.

Argyle et al. (Argyle et al., 1973) define six almost similar categories: information seeking, signaling, controlling the synchronization of speech, mutual gaze and intimacy, avoiding undue intimacy, and avoiding excess input of information. We are especially interested in examining the information seeking and the conversational flow regulation. While speaking a person emits information so we expect the listeners to look at the speaker, seeking for information.

Research showed that people gaze nearly twice as much at others while listening (75%) than while speaking (41%) (Argyle & Cook, 1976). In the case of a single speaker, all listeners would be focussing more in the direction of the speaker than the speaker is focussing at all of them. In accordance with this, Vertegaal found that, in a setting with three persons, people gaze much more at the speaker (62.4%) than at others (8.5%) (Vertegaal, 1998).

Assuming this, we expect head orientations of persons in a meeting to be good indicators for speaker identification. The remainder of this paper addresses this issue. First we describe the data collection. Then we show how well machine learning techniques are able to predict the speaker amongst four meeting participants, followed by a discussion how humans performed on this task. Finally, we compare the results for both approaches and elaborate on these results revealing more insight in gaze behavior.

3. Data collection

We used three four-person meetings with a total duration of 21 minutes that were recorded in the IDIAP

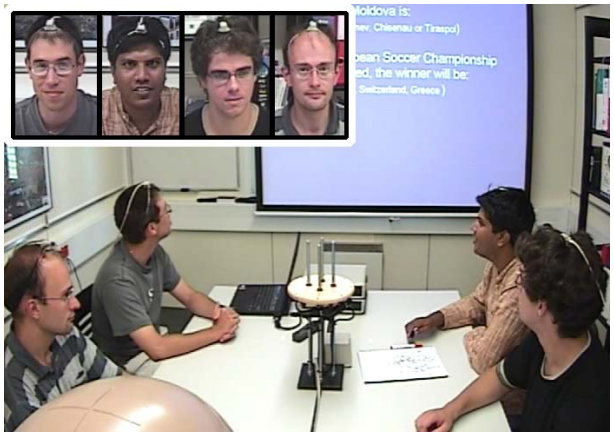


Figure 1. The setting of meeting 6, with close-ups of participants

smart room. Apart from the video and audio recordings, we recorded head position and orientation for all meeting participants. Flock of Bird sensors were used to accurately measure position and orientation at a rate of 50 Hz. The sensor is a small box and when mounted on top of a participant's head is not obtrusive and does not cause any distraction, as can be seen in close-ups in Fig. 1.

The non-scripted meetings contain lively discussions about pre-formulated statements. To make the experiment more realistic we also incorporated a whiteboard, where statements were shown.

After recording, we analyzed both video and orientation data to discover possible biases due to incorrect mounting of the Flock sensor on the head. We corrected the orientation data for these biases. For simplicity reasons, we only used azimuth data, see Fig. 2. Since all participants reside in the same plane, parallel to the table surface, we expect that azimuth orientation contains the most relevant information.

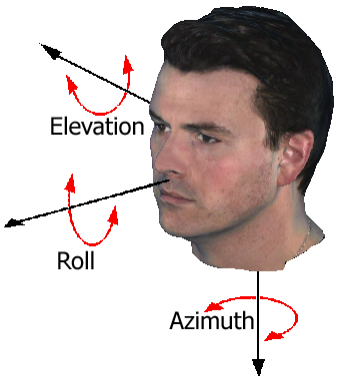


Figure 2. Azimuth, elevation and roll angles for heads

Furthermore, all occurrences with non-speech (laughter, silence, etc.) or with speech overlap were removed from the data set. The number of frames for each meeting, including *a priori* probabilities for each meeting and for all meetings, can be found in Table 1.

	M1	M2	M3	Total
Samples	11333	13078	28148	52559
A priori SP1	40.4%	26.9%	24.8%	28.7%
A priori SP2	27.3%	23.4%	9.8%	16.9%
A priori SP3	7.7%	8.6%	29.4%	19.5%
A priori SP4	24.7%	41.2%	36.0%	34.9%

Table 1. Number of samples and a priori probabilities of speakers in each meeting and all meetings. *M1* is meeting 1, *SP1* corresponds to speaker 1.

Trained	Test	ND	D
M1	M1	63.0%	82.8 %
M2	M2	51.3%	90.0 %
M3	M3	54.5%	76.6 %
M1, M2 & M3	M1, M2 & M3	50.0%	69.4 %
M1 & M2	M3	39.5%	35.4%
M1 & M3	M2	38.8%	33.5%
M2 & M3	M1	45.6%	40.4%
M1	M2 & M3	34.5%	36.5%
M2	M1 & M3	37.5%	32.7%
M3	M1 & M2	42.5%	32.3%

Table 2. Classification results for the Naive Bayes classifier without (ND) and with (D) discretization

4. Machine learning performance

In this section we compare the prediction results of Naive Bayes classifiers and Neural Networks for the task of speaker identification.

4.1. Data representation

For our machine learning experiment, we used input vectors $\mathbf{v}_t = (\alpha(1)_t, \alpha(2)_t, \alpha(3)_t, \alpha(4)_t)$, sampled at time t. Each $\alpha(i)_t$ corresponds to the azimuth angle of person i at time t. We used batch training, during which we also presented the speaker SP_t at time t, where $SP_t \in \{1, 2, 3, 4\}$.

In both experiments four series of tests were performed, each test having a different composition of training and test sets. In a first series both training and test set were obtained from a single meeting. In series two, both training and testing were performed on samples from all three meetings. In the third series, we trained on two meetings and tested on the third. Finally, we trained on a single meeting and tested on the other two. We discuss the results for the two machine learning techniques below.

4.2. Naive Bayes classifier

We trained a Naive Bayes classifier with and without supervised discretisation (Yang & Webb, 2003; Dougherty et al., 1995) using gaze vectors. We conducted the test for three different meetings using ten-fold cross validation. The results for the four series are shown in Table 2. The discretized data is shown in the column *D*, the original not discretized data is shown in the column *ND*.

From the table it can be seen that within a single meeting the classifier performs quite well. However when our training and test sets are taken from different meetings the performance drops significantly. Discretization improves our results when we test on samples from the meeting on which we trained and it decreases our results when we test on samples from other meetings than those trained on. Since the discretization algorithm is supervised, the bins created by the algorithm when trained on a particular meeting do not apply for samples from a different meeting. This results in a worse instead of an increased performance.

4.3. Neural Networks

We also used Neural Networks to estimate the speaker from the azimuth angle data. The Levenberg-Marquardt algorithm (Moré, 1978) was used for training. We performed the same four series of tests used for the Naive Bayes classifier.

In each series, we experimented with different numbers of neurons in the hidden layer. In the first two series, 25 neurons were found to yield best results, in the third series we used 15 neurons and in the last series only 5 neurons were used. In the first two series, the data was divided into a training set (60%), a test set (20%) and a validation set (20%). In the last two series, the training set contained all samples from the training meeting. The validation set contained 20% of the test meeting samples. The test set contained the other 80%. In each test, 5 runs were performed and the Neural Network with the best performance on the validation set was used to obtain the test performance. In Table 3, these results are summarized.

Again, we see that when training and test sets are sampled from the same meetings, the performance is high. The results however, do not generalize over meetings.

Training	Test	Result
M1	M1	82.6%
M2	M2	81.3%
M3	M3	72.3%
M1, M2 and M3	M1, M2 and M3	62.9%
M1 and M2	M3	44.2%
M1 and M3	M2	43.7%
M2 and M3	M1	48.1%
M1	M2 and M3	38.2%
M2	M1 and M3	40.2%
M3	M1 and M2	42.4%

Table 3. Classification results for Neural Networks

5. Human performance

In this section we describe how we tested the human performance on predicting speaker turns based only on head orientations. The main problem is that interpreting the numerical $\mathbf{v}_t$ vector is hard for humans. However, presenting the video data gives more information than just $\mathbf{v}_t$, such as possible facial expressions and gestures. But there are other problems such as the fact that humans have background knowledge. This information enables them to reason about gaze behavior and use their prior knowledge about meetings.

5.1. Experiment setup

To overcome the fact that we cannot present a number of numerical vectors to humans we exploited the fact that humans have background knowledge about meetings. We created a virtual meeting room (VMR, Fig. 3), allowing precise control over the delivered stimuli. In this VMR the setting is visualized, including the locations of the participants and the white board.

One problem with this kind of controlled virtual environment is the trade-off between the ecological validity and the experimental control, resulting in sterile artificial environments (Loomis et al., 1999). However, since we are only interested in head orientations we actually want to neglect other influences. By replacing the real setting (Fig. 1) with a virtual setting we are not only able to display the necessary information but also to remove all other possibly distorting information. This makes it possible, to a minimal extent, for the humans to interpret the gaze vectors and for other possible variables to be controlled.

Participants in the experiment were shown the meeting room with the participants as well as an option panel where they were able to choose among the four speakers, being either confident or very confident. Also there

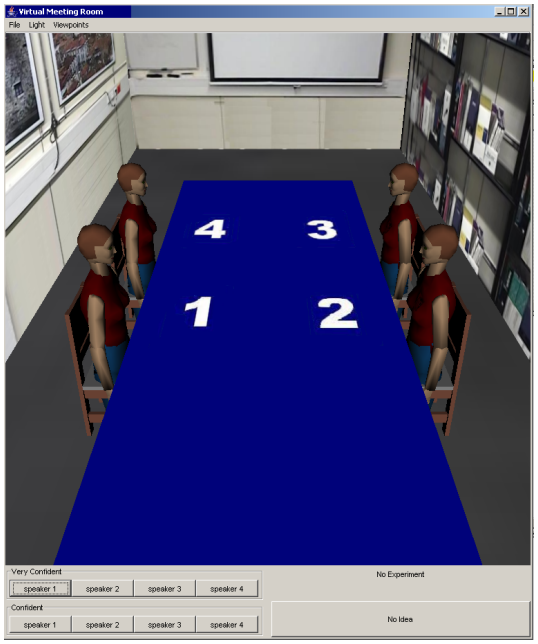


Figure 3. The virtual meeting room setting

was a 'no idea' button to prevent biased unfounded choices.

Each experiment consisted of a session with four parts, each containing 20 samples. There were two types of sessions. Type 1 contained feedback only on the first part whereas in type 2 the feedback was omitted completely. For the first two parts of both session types two times 20 samples were randomly chosen from meeting 3, the third part contained 20 randomly chosen samples from meeting 2 and the last part contained 20 randomly chosen samples from meeting 1.

The idea behind this was twofold. In the first place it enabled us to see if the feedback was helpful to the participants. Secondly, we were able to see whether feedback on samples from one meeting influenced the results on samples from different meetings. The feedback was given by showing a red arrow above the head of the correct speaker directly after the participants had judged the sample. We asked students and employees of our department to do the test.

5.2. Results

Both session types were completed 20 times, resulting in a total of 3200 answered samples. The results are shown in Table 4.

The table shows that the human performance is approximately 38%, which is lower than we expected. An interesting thing to note here is that there are sig-

Type	Part 1	Part 2	Part 3	Part 4	Total
1	47.8%	49.3%	29.8%	24.8%	37.9%
2	39.3%	42.0%	33.3%	35.3%	37.4%

Table 4. Classification results for humans per session type

nificant ($p < 0.05$ using a paired T-test) differences between the two session types. In the first place the results on the first two session parts are better with feedback than without feedback and for the last two session parts we see a significantly worse performance. Furthermore it appeared that when no feedback was given the performance remained much more stable over the different session parts.

We expect that if feedback is given humans create rules or models that work better for the meeting on which they received feedback. These could be *a priori* models that are applied when there is doubt about a possible outcome. When these models were tested on samples from different meetings they did not generalize. This seems to be in accordance with our machine learning findings.

6. Evaluation

From the results of both the machine learning techniques and the experiment with humans, it appears that the models do not generalize over all meetings. Apparently in the meetings different gaze behavior is displayed. This could be caused by different meeting topics, more or less frequent use of the whiteboard and differences in individual gaze behavior.

To gain more insight into differences between and within meetings, we examine two factors more closely in this section. First we investigate if there is a relation between the predicted speaker and the actual speaker in terms of location in the meeting. The second topic we investigate is the performance of the speaker prediction on different moments in a speaker turn.

6.1. Location effect on performance

We expect that different persons display different gaze behavior. Because we do not have sufficient meeting data in which the same persons participate, we try to find indications that there are differences between participants. We examine this by looking at differences in prediction performance for all speakers given the confusion matrices for all classifiers. Then we look at the person specific performance for neural networks on all meetings. Finally we examine whether the position with respect to other participants is of any influence.

In Table 5, 6, 7 and 8 the confusion matrices for the prediction results are shown for all classifiers. For the machine learning algorithms, training and testing is performed on samples from all three meetings (series 2 of Table 2 and 3) to obtain the most reliable model.

Actual	Estimated speaker			
speaker	SP1	SP2	SP3	SP4
SP1	26.2%	12.9%	9.9%	51.1%
SP2	9.6%	60.1%	6.0%	24.3%
SP3	8.7%	12.4%	42.5%	36.4%
SP4	11.8%	11.6%	7.6%	69.0%

Table 5. Confusion matrix for actual speakers (row) and predicted speakers (column) for Bayes classifier without discretization. Performance is 50.0%

Actual	Estimated speaker			
speaker	SP1	SP2	SP3	SP4
SP1	65.5%	8.0%	7.7%	18.9%
SP2	9.5%	71.6%	6.2%	12.7%
SP3	11.5%	7.0%	67.0%	14.5%
SP4	15.9%	4.7%	6.3%	73.2%

Table 6. Confusion matrix for actual speakers (row) and predicted speakers (column) for Bayes classifier with discretization. Performance is 69.4%

Actual	Estimated speaker			
speaker	SP1	SP2	SP3	SP4
SP1	50.6%	9.7%	11.3%	28.4%
SP2	10.1%	61.4%	8.0%	20.5%
SP3	9.8%	8.3%	60.1%	21.2%
SP4	12.3%	6.1%	6.6%	75.0%

Table 7. Confusion matrix for actual speakers (row) and predicted speakers (column) for the experiment with Neural Networks. Performance is 62.9%

Actual	Estimated speaker			
speaker	SP1	SP2	SP3	SP4
SP1	31.8%	15.4%	24.0%	28.8%
SP2	15.3%	42.8%	24.0%	17.9%
SP3	14.3%	14.7%	51.1%	20.0%
SP4	21.7%	12.7%	22.0%	43.7%

Table 8. Confusion matrix for actual speakers (row) and predicted speakers (column) for the experiment with humans. Performance is 42.3%

It appears that there are differences in prediction results for a certain location. For example, for the Naive Bayes classifier without discretization (Table 5) correct

performance for speaker 1 is 26.2% whereas the performance for speaker 4 is 69.0%. The results from the above tables are summarized over the three meetings. To find out whether different persons display different gaze behavior we examine the differences between meetings. In Table 9, the Neural Network prediction results for each meeting are summarized.

Location	M1	M2	M3	Total
1	51.4%	47.8%	51.6%	59.0%
2	67.8%	57.4%	58.7%	59.0%
3	88.7%	45.1%	59.8%	62.2%
4	73.8%	70.2%	77.8%	73.0%

Table 9. Correct Neural Network predictions for each location per meeting

Given the results of the Neural Network it appears that the differences in speaker prediction performance are also present between meetings. This shows that there are differences in performance for each person in each meeting, which is a strong indication that there are differences in gaze behavior for different persons. This might be an explanation for the fact that our models do not generalize.

If we look at the origin of the prediction errors, we can tell what mistakes are made with respect to the relative position. In Fig. 3 we see that participants 1 and 4 are sitting next to each other, whereas participants 1 and 2 are sitting opposite to each other. Finally, participants 1 and 3 are sitting diagonally to each other. Table 10 summarizes the errors in these directions.

	Next to	Opposite	Diagonal
Naive Bayes classifier (ND)	36.0%	32.4%	31.7%
Naive Bayes classifier (D)	39.2%	31.1%	29.8%
Neural Network	37.1%	31.6%	31.4%
Humans	38.2%	32.3%	29.5%

Table 10. Estimation errors in different directions for machine learning techniques and humans for all meetings

We see that there is more confusion between two persons who are sitting next to each other than between two people who are in opposite corners of the table. The results are similar for all four classifiers. We expect a relation between physical participant distance and the prediction error. The distance between participants at one side of the table is smaller than the distance between two participants on opposite sides of the table. Changing the meeting setting from a square table with participants sitting opposite to each other to a round table as is used in (Stiefelhagen, 2002) could eliminate those biases but cannot explain the differences from Table 9.

6.2. Performance within a speaker turn

We can take a closer look at the data and determine how our classifiers perform during a speaker turn. Can the speaker be determined better in the beginning, in the middle or at the end of a speaker turn? If we look at the speaker prediction scores within speaker turns longer than 1 second (92.4% of all samples), we obtain the results from Fig. 4. We ignored speaker turns shorter than 1 second, containing short utterances. Ten equally sized bins were used to assure that for each interval sufficient samples (over 300 per bins) remained.

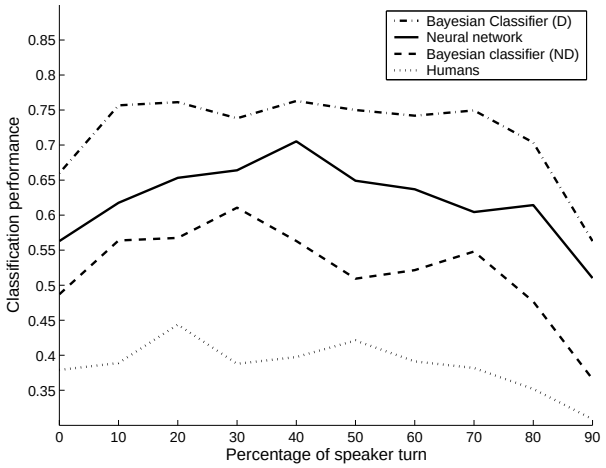


Figure 4. Speaker prediction performance during a speaker turn using 10 bins

We see a trend in the graph for all four classifiers. In the beginning and at the end of a speaker turn, the speaker is harder to determine than halfway the turn. This could be explained by assuming that participants switch from speaking to listening, from listening to speaking or start gazing at the new speaker. A similar explanation could be found for the lower identification performance at the end of a speaker turn. Participants might start gazing at the person who they expect will reply to the current speaker.

7. Conclusions

To gain insight into gaze behavior in meetings, a comparison of classification results for a Naive Bayes classifier, Neural Networks and humans was made on the task of speaker prediction from azimuth head angles. In four-person meetings, a Naive Bayes classi-

fier was able to predict 69.4% correctly, Neural Networks scored 63.2%, and humans only 37.7%. The machine learning classification results do not generalize over meetings. In the experiment with humans we see similar results. The model that was learned using the feedback increased the outcome for the meeting where the feedback was given, but decreased the result for the other meetings. We showed that there are strong indications that human specific gaze behavior influences the fact that the models do not generalize. Additionally, we showed that for all classifiers the performance in the beginning and at the end of a speaker turn is worse than halfway through the speaker turn.

8. Future work

To improve insight into gaze behavior we plan to investigate whether adding more information, such as body orientation will increase the classification performance. Experiments have started where we analyze head orientations of complete speaker turns. Also, we plan to verify simple protocols possibly applied by humans when predicting the speaker.

With respect to the fact that our models do not generalize over meetings, we intend to do more research on person specific gaze behavior. Information such as the typical duration of personal speaker turns, the average head movement during a turn might reveal more cues along our path to addressee detection and focus of attention estimation. Also, more research needs to be done on the effect of meeting topics and their context on the prediction of gaze behavior.

9. Acknowledgements

This work was partly supported by the European Union 6th FWP IST Integrated Project AMI (Augmented Multi-party Interaction, FP6-506811, publication AMI-33).

References

Argyle, M., & Cook, M. (1976). *Gaze and mutual gaze.* Cambridge University Press.

Argyle, M., Ingham, R., Alkema, F., & McCallin, M. (1973). The different functions of gaze. *Semiotica, 7*, 19–32.

Chen, H., & Perich, F. e. a. (2004). Intelligent agents meet semantic web in a smart meeting room. *Proceedings of the Third International Joint Conference on Autonomous Agents & Multi Agent Systems (AAMAS 2004).*

Dougherty, J., Kohavi, R., & Sahami, M. (1995). Supervised and unsupervised discretization of continuous features. *International Conference on Machine Learning* (pp. 192–202).

Garau, M., Slater, M., Bee, S., & Sasse., M. (2001). The impact of eye gaze on communication using humanoid avatars. *Proceedings of the SIG-CHI conference on Human factors in computing systems* (pp. 309–316). Seattle, WA USA.

Kendon, A. (1967). Some functions of gaze direction in social interaction. *Acta Psychologica, 32*, 1–25.

Loomis, J. M., Blascovich, J. J., & Beall, A. C. (1999). Immersive virtual environment technology as a basic research tool in psyychology. *Behavior Research Methods, Instruments and Computers, 31(4)*, 557–564.

Mani, I., & Maybury, M. T. (1999). *Advances in automatic text summarization.* MIT Press.

Moré, J. (1978). The Levenberg-Marquardt algorithm: Implementation and theory. *Lecture Notes in Mathematics 630* (pp. 104–116).

Nijholt, A. (2004). Where computers disappear, virtual humans appear. *Computers and Graphics, 28.* to appear.

Reidsma, D., Rienks, R., & Jovanovich, N. (2004). Meeting modeling in the context of multimodal communication. *Proceedings of the MLMI'04, Martigny.*

Stiefelhagen, R. (2002). Tracking focus of attention in meetings. *Proc. of the ICMI2002.* Pittsburgh.

van Es, I., Heylen, D., van Dijk, E., & Nijholt, A. (2002). Gaze behavior of talking faces makes a difference. *Proceedings of the ACM-CHI 2002* (pp. 734–735). Minneapolis, USA.

Vertegaal, R. (1998). *Who is looking at whom.* Doctoral dissertation, University of Twente.

Vilhjalmsson, H., & Cassell, J. (1998). Bodychat: Autonomous communicative behaviors in avatars. *Proc. of the 2nd Annual ACM Int. Conf. on Autonomous Agents.* Minneapolis, USA.

Wright, I. (1997). *Emotional agents.* Doctoral dissertation, University of Birmingham.

Yang, Y., & Webb, G. I. (2003). On why discretization works for naive-bayes classifiers. *Proceedings of the 16th Australian Joint Conference on Artificial Intelligence (AI).*

A Modular Approach to Facial Expression Recognition

Michal Sindlar SINDLAR@PHIL.UU.NL
Cognitive Artificial Intelligence, Utrecht University, Heidelberglaan 6, 3584 CD, Utrecht

Marco Wiering MARCO@CS.UU.NL
Intelligent Systems Group, Utrecht University, Padualaan 14, 3508 TB, Utrecht

Abstract

We study the use of multi-layer perceptrons in applying machine learning to the recognition of emotional expressions from frontal images of human faces. The perceptrons are trained using per-pixel luma data from the images' mouth and eye areas, and map the inputs to one of 6 emotions. We compare 3 different methods for processing input information: 1) one network module for all inputs; 2) one network module for both eyes, and one for the mouth; 3) one network module for the mouth, one for the left eye, and one for the right eye. Our results show that involving multiple modules leads to better results, resulting in a performance high of 84% images classified correctly.

1. Introduction

Automated facial expression recognition from static images can be useful in a number of different applications, such as human-machine interaction, or detection of audience response. The goal of this study is to find out whether it is possible to successfully perform facial expression recognition using multi-layer perceptrons in a modular setup on practically unprocessed input data. For this specfic purpose a software tool named Narcissus[1] was developed.

The sections 2–7 deal with the following: Section 2 discusses the setup of the research and the image data used, section 3 is about the application used to process the data, section 4 briefly describes the training and testing procedure, in section 5 the tests and the results are discussed, and section 6 discusses our study in a context of related work. Section 7 concludes this paper.

[1] After the mythical figure who fell in love with himself after seeing the reflection of his own face in a pond.

2. Research setup

Since the main object is to construct a classifier to use with static frontal images of human faces expressing emotions, a proper classifier as well as suitable images will be needed. This section discusses the selection of both.

2.1. Choice of classifier

We want a classifier that is robust, and relatively easy to implement, like a multi-layer perceptron (MLP) or radial basis function network (RBF). Since a study similar to this one obtained good results with MLPs, and results with RBFs were somewhat worse (Gargesha & Kuchi, 2002), MLPs with a logistic sigmoid activation function are selected as classifiers. Learning takes place through back-propagation. The classifier's outputs are Ekman's 6 basic emotions with clear facial signals: anger, disgust, fear, happiness, sadness, and surprise (Ekman, 1994). Other output categories are of course possible —such as the 2-dimensional model of emotion (Russell, 1980)— but Ekman's emotions are widely-used in research, and also the image data described in section 2.2 is based on them, so this model was adopted.

2.2. Image data

To investigate the expression of emotions in facial images, a sufficient amount of images expressing those emotions is of course required. Three sets were used, described in more detail in Sections 2.2.1, 2.2.2 and 2.2.3. Sample pictures are shown in Figure 1.

2.2.1. Cohn-Kanade image set

The Cohn-Kanade Facial Expression Database (Kanade et al., 2000) is a collection of approximately 2,000 grayscale image sequences from over 200 subjects. The images used were expertly analyzed with FACS for the occurrence of so-called action

Figure 1: *Sample images expressing surprise, one from the Cohn-Kanade database (left), one from the JAFFE database (middle), and one from the POFA set (right).*

units, as described in the Facial Action Coding System (Ekman & Friesen, 1978). The AU-codes were manually translated into Ekman's 6 basic emotions using the rules from the FACS Investigator's Guide (Ekman et al., 2002), and only the images that were expressing emotion according to this system were used in this research.

2.2.2. JAFFE image set

The Japanese Female Facial Expression (JAFFE) database (Lyons et al., 1998) consists of 213 grayscale images of Japanese women posing the 6 basic expressions used in this research, plus a neutral one. The images have been rated on a 5-point scale (from 1 to 5) for each of the 6 emotion categories by 60 female Japanese students. Each of the images was assigned to the category for which it achieved the highest overall rating.

2.2.3. POFA image set

The Pictures Of Facial Affect image set (Ekman & Friesen, 1976) contains grayscale photographs of 14 actors portraying expressions that are reliably classified by naive observers as the 6 basic expressions used in this study (the overall agreement is 91.6%).

3. Specifics of the application

The inner workings of the software tool Narcissus[2] are briefly described in this section. Section 3.1 describes the loading of images into the application for processing, in section 3.2 training one or more networks using data from the opened images is described, and section 3.3 describes testing one or more networks on the opened images.

3.1. Opening images

Images of the formats GIF, JPEG and PNG are supported. These images can be opened into the application, and can be used either for creating and training a network, or testing an existing network. When images are opened for the first time, the areas containing the left eye, the right eye, and the mouth have to be selected by hand. These areas are the facial features that are expected to contain most emotion-related information. The selection procedure is manual: the user draws a selection rectangle to select the region where the features are. The aspects of these selection rectangles are constant, with a *horizontal* : *vertical* ratio of $1.3 : 1$ for the eyes, and $2.2 : 1$ for the mouth. The aspects are kept constant, so that no distortion occurs when the feature images are scaled. The numbers of 1.3 and 2.2 were chosen more or less arbitrarily; the criterion being that the whole feature region (e.g. the left eye) could be fitted, without incorporating a lot of the surrounding 'noise'.

3.2. Training one or more networks

In training mode the images that have been opened can be used to train one or more networks. Table 1 shows the possible situations.

# of networks	input
1	both eyes and the mouth
2	both eyes into one network, the mouth into the other
3	each eye goes into a separate network, as does the mouth

Table 1: *Different networks and their input.*

When one network is selected, this network processes all image features. When two networks are selected, one network will be set up to process the left as well as the right eye, the other will process the mouth. In the case of three networks, each feature is processed by a separate network. From now on, when talking about the networks used by Narcissus, these will be referred to as 'modules'. The mode in which 2 modules are selected for processing — wherein one processes both eyes and the other the mouth — will be referred to as 'a system of 2 modules' or a '2-module system', and likewise for 3 modules.

After selecting the number of modules, the desired feature dimensions have to be set for the eyes (same dimensions for both) and the mouth. All features are scaled to these dimensions using the standard Java AffineTransform.getScaleInstance(x,y) scaling operation, to ensure that every feature image yields the same amount of inputs. The default setting is a dimension of $20 * 15$ pixels for the eyes and $40 * 18$ pixels for the mouth.

[2] Available at `http://narcissus.no-ip.org/`

For a 1-module system, this yields (including the bias):

$$(20 * 15) * 2 + (40 * 18) + bias = 1,321 \text{ inputs}$$

The data that is the actual input for the networks, is the *luma* (Y') value of each pixel, which is calculated with the following standard formula for obtaining luma values from non-linear RGB.

$$Y' = 0.299 * R + 0.587 * G + 0.114 * B$$

This comes down to converting the image from RGB color to grayscale. Using the luma values reduces the dimensionality of the data: instead of using three values (R, G, and B) from each pixel as input, we now only have to use one (Y'). This is also the case for images that already are in grayscale, which in RGB color space are represented as (Y', Y', Y'). The network input value then is $0.299\ Y' + 0.587\ Y' + 0.114\ Y' = Y'$.

3.3. Testing one or more networks

In testing mode, the networks that were created in training mode can be evaluated. The necessary parameters, such as the feature dimensions, are taken from the network settings. Images can be evaluated with respect to a 1-, 2-, or 3-module system one by one or all at once. This mode provides a lot of visual feedback. The outputs of all networks are recorded and displayed, as well whether or not an image was classified correctly. Also, the outputs can be visualized in a graph, for easy viewing.

The face in Figure 2 was analyzed manually as described in 2.2.1 and reported to be expressing sadness. The output of the 3-module system as visualized in Figure 3 tells that two modules got it right, and one did not. There are six groups of bars; one for each emotion category.

Figure 2: *Sample image from the Cohn-Kanade database, expressing sadness.*

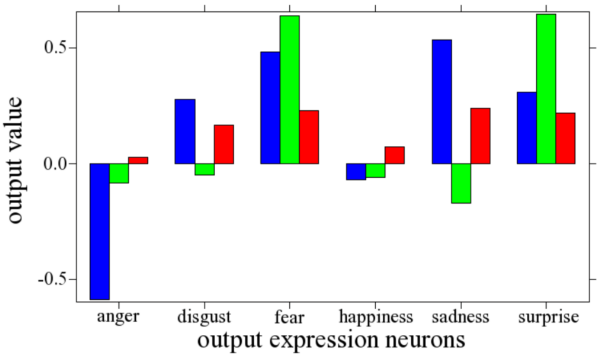


Figure 3: *Graph showing the output from three networks.*

From left to right: anger, disgust, fear, happiness, sadness, surprise. There are three bars: one for each module. The module analyzing the left eye is represented by the leftmost bar in each group (blue), the middle (green) bar stands for the right eye, and the module analyzing the mouth is the rightmost bar in each group (red). In this case, the left eye (left, blue) and mouth (right, red) correctly reported sadness (the 5^{th}group of bars), while the right eye module (middle, green) got it all wrong: it reported surprise, with fear coming in second by the smallest of margins. Sadness actually was the least likely option for this network.

4. Training and testing procedure

The image set available for training and testing consists of 458 images in total. The images are spread unevenly over the 6 emotion categories , as follows (with the Cohn-Kanade / JAFFE / POFA ratio shown in parentheses): 60 in *anger* (23/24/13), 82 in *disgust* (51/18/13), 30 in *fear* (10/7/13), 116 in *happiness* (73/30/13), 55 in *sadness* (22/22/11) and 115 in *surprise* (77/25/13).

Since the number of samples is quite low for some categories, cross-validation has been adopted as training procedure. Cross-validation allows us to obtain valid results using only a small number of samples. This is done by dividing the data into S segments, using data from $S - 1$ of these segments for training, and testing performance with the remaining segment. This process is repeated S times, and the results of S runs are then averaged to obtain the final result. In our case the data has been split up in 10 sets, each containing approximately 90% of the samples as training data and the remaining 10% or so for testing. A certain network's performance has been defined as the average of its results on the 10 subsets.

5. Tests & results

The tests described in this section were performed following the procedure explained in section 4.

5.1. Determining optimal parameters

The following sections are about determining a function that returns a reasonable number of hidden units based on the number of inputs (5.1.1), and finding a feature size that has enough detail, without being unnecessarily large and slow to process (5.1.2).

5.1.1. Hidden units

The focus here is on calculating the number of hidden units as a function of the number of inputs. This number should not be too small because this allows for fewer possible mappings, and thus less expressive power, but there should not be too many hidden units either, because this increases processing time and can lead to overfitting the data.

No optimal feature size has been determined yet, so three sets of feature sizes have been considered, listed below in order of increasing number of inputs.

Low-detail: The dimension of the eyes is $5 * 4 = 20$ pixels and the dimension of the mouth is $10 * 5 = 50$ pixels. Including the bias, this amounts to $(2 * 20) + 50 + 1 = 91$ inputs.

Medium-detail: Eyes are $10 * 8$ and mouth is $20 * 9$. Total of $160 + 180 + 1 = 341$ inputs.

High-detail: Eyes are $40 * 31$ and mouth is $80 * 36$. Total of $2,480 + 2,880 + 1 = 5,361$ inputs.

All features are processed by one single network. For this test there was no need to do otherwise, because the number of hidden units applies to a network in general, and does not depend on the particular features it is processing. The learning rate was fixed at 0.02 and all networks ran 500 passes. If a 100% score was achieved on the training set before the 500^{th}pass, back-propagation (and thus further learning) stopped.

First the number of hidden units n_{hidden} was calculated from the number of inputs n_{input} using the function:

$$n_{hidden} = \sqrt[x]{n_{input}}$$

Then, 4 tests were performed using the aforementioned formula and the values 2, 3, 4 and 5 for x. This test yielded the results shown in Table 2. This table shows the average score over 10 cross-validation runs, and in parentheses the *standard deviation (σ)* of the 10 scores that make up the final score.

x	low	medium	high
2	70% (5.6)	79% (4.3)	81% (3.6)
3	64% (3.9)	78% (6.5)	81% (4.2)
4	57% (7.9)	73% (5.8)	81% (4.3)
5	48% (8.9)	57% (7.0)	77% (5.0)

Table 2: *Scores for the $\sqrt[x]{n_{input}}$ function.*

The most desirable results were obtained with $x = 2$ for the low-detail system, $x = 2$ for medium-detail, and $x = 4$ for high-detail. A function that yields a suitable number of hidden units, in a range comparable to the one given by $n_{hidden} = \sqrt[x]{n_{input}}$ with the aforementioned values for x, is

$$n_{hidden} = 3 * \ln n_{input}$$

This function returns a relatively large amount of hidden units for small networks, and (compared to $\sqrt[2]{n_{input}}$, for instance) a small n_{hidden} for large networks. Considering the results as seen in Table 2, this is what we want.

5.1.2. Feature size

As Table 2 shows, more inputs and a lot of hidden neurons seem to give the best results. However, there is a downside to having a large network. Table 3 shows the approximated processing times[3] for one pass on 414 examples *without* back-propagation.

x	low	medium	high
2	0.05	0.24	12.70
3	0.04	0.10	3.20
4	0.03	0.06	1.50
5	0.02	0.05	1.10

Table 3: *Approximated processing times in seconds for each of the networks.*

This means the network processing the low-detail features (91 inputs in total) with $\sqrt[2]{n_{input}}$ function for the hidden neurons took 50 milliseconds (0.05 seconds) for classifying 414 images, while the network processing high-detail features with $\sqrt[2]{n_{input}}$ hidden neurons took 12.7 seconds. Training both these network for 500 passes would take well over (because of back-propagation, which was not considered in Table 3) 25 seconds for the low-detail network, compared to 1 hour and 45 minutes for the high-detail network. This is not really a problem if learning takes place off-line, but there's no use in having large processing times if it doesn't improve results.

The medium-detail network did not have as good results as the high-detail one, but the latter took a lot longer to process, without spectacular improvements

[3] On an Athlon XP 3200+ with 1 gigabyte of memory.

in performance. Therefore an intermediate sized network was trained. This network has a $20 * 15$ size for the eyes, and a $40 * 18$ size for the mouth. Including the bias, this amounts to 1,321 inputs. The $n_{hidden} = 3 * \ln n_{input}$ function returns 22 hidden neurons. This network has quite a good processing time: about 0.8 seconds per pass. The averaged score of 10 cross-validation runs is 82% with a standard deviation of 4.7, which is the best result so far. In all following experiments these settings (Table 4) are used.

eyes	mouth	n_{hidden}
$20 * 15$	$40 * 18$	$3 * \ln 1,321 = 22$

Table 4: *Optimal settings for feature size and n_{hidden}.*

5.2. Performance of networks

In this section, the performances of 1-, 2,- and 3-module systems are evaluated using the determined settings, and also a 'voting' system that combines the other systems' outputs is reviewed.

5.2.1. Cross-module comparison

Table 5 shows the performance of each network from a 1-, 2-, and 3-module system as the overall percentage of correctly classified images, with the standard deviation σ on 10 cross-validation runs in brackets.

system	module	score (σ)
1 module	eyes & mouth	82% (4.7)
2 modules	eyes	68% (5.3)
	mouth	70% (7.9)
3 modules	left eye	62% (7.8)
	right eye	60% (5.0)
	mouth	69% (6.0)

Table 5: *Comparison of 1-, 2-, and 3-module systems.*

Clearly, the single-module system does best. It has the highest overall score of 82% correctly classified images, and the lowest σ (4.7), which means that it had the least amount of variation on the 10 cross-validation runs, and therefore is the most consistent of the 3 systems. Table 6 shows the highest and lowest scores for each of the networks in Table 5.

network	module	highest	lowest
1 module	eyes & mouth	89%	74%
2 modules	eyes	74%	59%
	mouth	85%	60%
3 modules	left eye	76%	54%
	right eye	65%	51%
	mouth	77%	61%

Table 6: *Performance high/low for 1-, 2-, and 3-module systems.*

Also note how the networks processing a single eye do not perform much worse than the network processing both eyes. The highest score for the 3-module system working on the left eye (76%) was even higher than the one for the 2-module network working on both eyes (74%). This is interesting, because it shows that even from a partially occluded face (where perhaps only half of the face is visible) expression recognition is possible when using a modular approach.

5.2.2. Module addition

Section 5.2.1 showed how a 1-module system outperformed the networks from the 2- and 3-module systems, and how the networks from the 2-module system also outperformed those from the 3-module system. Now let's see what happens when the individual modules are combined, so that they all 'cast a vote' in a single system. An easy way to achieve this is by simply adding up the 6 outputs of each system and pretending the resulting values are the outputs of a single system. Table 7 shows the results of this procedure for the 2- and 3-module systems, along with the result of the 1-module system for comparison.

system	module	score (σ)
1 module	eyes & mouth	82% (4.7)
2 modules	eyes + mouth	82% (3.5)
3 modules	left eye + right eye + mouth	84% (5.3)

Table 7: *Performance of the 2- and 3-module systems after addition of output activations.*

After addition of output activations, the 3-module system suddenly performs best! Same as with the discussion of the separate modules, let's have a look at the highest and lowest scores of each of the systems. The 1-module system's scores are the same, of course. The results are shown in Table 8.

system	module	highest	lowest
1 module	eyes & mouth	89%	74%
2 modules	eyes + mouth	87%	77%
3 modules	left eye + right eye + mouth	93%	79%

Table 8: *Performance high/low for 1-, 2-, and 3-module systems after addition.*

Adding these systems up again produces a combined super-system in which all three systems are casting their vote. This system performs as shown in Table 9.

system	score	σ	highest	lowest
combined	85%	5.5	93%	75%

Table 9: *Score, σ and performance high/low for 1-, 2-, 3-module systems added together.*

This is the best system so far, but not by a great margin and at a price. It takes approximately three times

as much time to run compared to the other systems, because it actually consists of those systems. Its processing time is the processing time of the 1-module system, added up to that of the 2-module system, and again to that of the 3-module system. Therefore it's quite inefficient without major improvement over the other systems.

5.3. In-depth analysis

In this section the systems discussed in section 5.2.2 will be analyzed in-depth. The analysis is presented in form of a *confusion matrix*, as defined in (Kohavi & Provost, 1998). This matrix visualizes the *combined results* of the 10 cross-validation test runs in a diagram, whose row as well as column headings show a category label. *A* stands for anger, *D* for disgust, *F* for fear, *H* for happiness, *Sa* for sadness and *Su* for surprise. The column headings (first row) stand for the desired (correct) classification, the row headings (first column) for the actual classification. The cell values show how often a certain error (confusion) occurred. The values in the Σ-column's and Σ-row's cells show the summation over the preceding cells in their respective row and column. For the Σ-column, this can be interpreted as the *bias* towards a category, and for the Σ-row it represents the total number of *misclassifications* for images from this category. On the diagonal the percentage of *correctly* classified samples of a certain category is shown, and the final row (**T**) shows the total number of samples in each category, together with the total number of pictures. (Σ, Σ) shows the performance of this system as a percentage of correctly classified images, with the total number of *misclassified* images in brackets.

So, for example, by going to (A, Sa) in Table 10, we find that after 10 cross-validation tests for the 1-module system, 3 images in total that should have been classified as *Sadness* were confused for the category *Anger*.

(A, A) tells us that 63% of the test samples from the *Anger* category were correctly classified as such. This can be verified by checking the total number of misclassified *Anger* images in (Σ, A), which is 22, and indeed $\frac{60-22}{60} * 100\% = 63\%$.

The following sections 5.3.1 to 5.3.4 discuss the four addition systems from section 5.2.2.

5.3.1. 1-module system, Table 10

As Table 10 shows us, the best-recognized category (happiness) was recognized much better than the worst-recognized (fear): 94% compared to 47%. Anger and disgust were often confused for each other: 15 times in total (add (A, D) and (D, A) together). Surprise is mistaken for disgust only once, and disgust never for surprise. In fact, only fear is often mistaken for surprise, and this quite often too (7 times). Another thing to note is that sadness is mistaken for happiness — which could be considered the opposite emotion — 5 times in total, which is a lot in this context.

	A	*D*	*F*	*H*	*Sa*	*Su*	Σ
A	*63%*	6	1	2	3	3	**15**
D	9	*85%*	2	0	3	1	**15**
F	2	0	*47%*	2	2	3	**9**
H	3	3	2	*94%*	5	2	**15**
Sa	6	3	4	3	*73%*	0	**16**
Su	2	0	7	0	2	*92%*	**11**
Σ	**22**	**12**	**16**	**7**	**15**	**9**	***82%*** (81)
T	*60*	*82*	*30*	*116*	*55*	*115*	***458***

Table 10: *Confusion matrix for the 1-module system.*

5.3.2. 2-module system, Table 11[4]

	A	*D*	*F*	*H*	*Sa*	*Su*	Σ
A	*73%*	7	2	4	5	1	**19**
D	5	*84%*	1	2	1	0	**9**
F	3	0	*47%*	1	3	5	**12**
H	3	4	3	*93%*	6	1	**17**
Sa	5	2	4	1	*65%*	1	**13**
Su	0	0	6	0	4	*93%*	**10**
Σ	**16**	**13**	**16**	**8**	**19**	**8**	***82%*** (80)
T	*60*	*82*	*30*	*116*	*55*	*115*	***458***

Table 11: *Confusion matrix for the 2-module system, after addition of individual modules.*

Anger and disgust are mistaken for each other 12 times in total, again the highest score, while disgust and surprise are never mistaken for each other. Again, fear is often mistaken for surprise: 6 times. This is a lot, especially considering the fact that fear only has 30 samples. Sadness is mistaken for happiness 6 times.

5.3.3. 3-module system, Table 12

The anger-disgust confusion is lower for the 3-module system, only 8 mistakes. Sadness is mistaken for happiness 7 times. Disgust and surprise are confused only once. Fear is mistaken for surprise 5 times.

5.3.4. Combined super-system, Table 13

Since this system reflects the previously discussed three systems, there are no real surprises.

[4]For (Σ, Σ) in Table 11, $(378/458) * 100\% = 83\%$ and not 82%. This is not an error, but a reflection of the fact that the performance has been calculated as the average of the performance of individual cross-validation tests, which in this case leads to a discrepancy of 1%.

	A	*D*	*F*	*H*	*Sa*	*Su*	Σ
A	*70%*	3	1	2	5	0	**11**
D	5	*85%*	3	3	1	1	**13**
F	2	2	*53%*	1	3	1	**9**
H	5	6	3	*93%*	7	1	**22**
Sa	5	1	2	1	*69%*	1	**10**
Su	1	0	5	1	1	*97%*	**8**
Σ	**18**	**12**	**14**	**8**	**17**	**4**	***84%*** (73)
T	*60*	*82*	*30*	*116*	*55*	*115*	***458***

Table 12: *Confusion matrix for the 3-module system, after addition of individual modules.*

	A	*D*	*F*	*H*	*Sa*	*Su*	Σ
A	*72%*	5	1	1	3	0	**10**
D	6	*87%*	3	1	2	1	**13**
F	1	0	*50%*	1	3	3	**8**
H	3	4	3	*96%*	6	1	**17**
Sa	5	2	4	2	*73%*	0	**13**
Su	2	0	4	0	1	*96%*	**7**
Σ	**17**	**11**	**15**	**5**	**15**	**5**	***85%*** (68)
T	*60*	*82*	*30*	*116*	*55*	*115*	***458***

Table 13: *Confusion matrix for the combined super-system.*

The overall pattern for the four systems considered is that fear is recognized worst by far, and that performance on happiness and surprise is best. There is a clear correlation between performance and the total number of samples for a certain category: the more samples, the better the performance. For all cases considered, bias towards happiness was significantly higher than bias towards surprise. Perhaps this is because the expression of surprise has a more unique signal (wide-open mouth), only to be confused with fear, while the expression of happiness comes in many different forms. It could therefore be that the networks assign indeterminable samples other than fear to happiness, because this has the highest chance of being correct.

6. Related work

In the first part of this section, related studies are briefly reviewed. In the second part, a comparison is drawn between this study and similar ones. Finally, suggestions for further research are given, as well as possible improvements for the current approach.

6.1. Review of related work

The past decade has seen a lot of activity in the field of (semi-) automatic facial expression recognition, using widely differing approaches. This brief review will focus on a neural network-based method operating on input data gathered from static facial images. Other possibilities are analysis of image sequences (with output to FACS AU's, for instance), and analysis of static images using template- or rule-based methods. For a concise overview of the myriad of possible approaches, see (Pantic & Rothkrantz, 2000).

Zhang et al. (Zhang et al., 1998) compare geometry-based and Gabor wavelets-based approaches to facial expression recognition using multi-layer perceptrons. Their findings are that Gabor wavelets are much more powerful than geometric position. They achieve an overall score of 90.1% using combined information from Gabor wavelets and geometric position, with 7 hidden units and output to 7 categories (anger, disgust, fear, happiness, sadness, surprise, and neutral), using 213 images from the JAFFE database. Fear is problematic: when excluding it, the results are 92.3% correctly classified images, and human agreement with the expressors' intention rises with 6% to 85.6%.

A very interesting approach is taken by Dailey et al. (Dailey et al., 2002) in a system called EMPATH. They present an artificial approach to facial expression recognition, modelled on the human perceptual system. Their system has three major layers. The *perceptual layer* uses Gabor filters and represents the human complex cells in the visual cortex. The *Gestalt layer* performs principal component analysis using linear hidden units, and is comparable to the 'face cells' in the human inferior temporal cortex. The *category layer* is the output layer, and has the 6 categories of anger, disgust, fear, happiness, sadness and surprise. This system's performance on the POFA image database as input was as depicted in Table 14. Again, performance on fear is particularly bad.

Expression	*System performance*	*Human agreement*
Happiness	100.0%	98.7%
Surprise	100.0%	92.4%
Disgust	100.0%	92.3%
Anger	89.1%	88.9%
Sadness	83.3%	89.2%
Fear	67.2%	87.7%
Average	**90.0%**	**91.6%**

Table 14: *Performance of EMPATH and human agreement on image data from the POFA database.*

6.2. Comparison to related work

Compared to other approaches, ours is simple and straightforward because it uses hardly any preprocessing (apart from converting all data to grayscale, which doesn't affect much since all image sets used have grayscale images). Still, it achieves pretty good results, with a high around 85%. However, feature selection is manual, and one could wonder how much bias is introduced by hand-selecting the features.

An interesting phenomenon is the (relatively) poor detection of fear by artificial neural network-based systems, as well as human observers (as illustrated by Table 14 with the EMPATH results). In our system, detection of fear was quite terrible too, which we first attributed to the low number of training samples. Now it seems that a higher amount of fear-examples maybe could improve results, but that fear also has inherent characteristics which make it harder to recognize.

It should be noted that we used three different image collections, and that most other studies use only one. The sample images from a single set for a single category can be quite low; e.g. 7 samples from the JAFFE database for fear. Most likely our results could improve by using more samples per image set, since it is much harder for machine learning methods to generalize when learning from a low number of samples.

7. Discussion

Our goal of creating a working system to classify emotions from static frontal images, using hardly preprocessed images and multi-layer perceptrons, has been met quite well. Most notable of the method used in this study is perhaps the use of a separate network module per facial feature (left eye, right eye, and mouth). Separate feature modules yield reasonable results (between 60% and 70%), and simple addition of output values improves results to well over 80%). This could be useful in processing facial images for recognition of emotional expression, where features have been (partially) obscured. It is probable that a modular approach could be successful in *identity recognition* from partially obscured facial images as well.

Using preprocessing, such as application of Gabor filters, seems to improve results. This study could be extended by applying a Gabor wavelet-based approach and a modular one to build a more robust Gabor-based system. It would also be interesting to see how a modular approach would perform when automatic face and/or feature detection is applied. Moreover, to obtain a generally applicable emotion-recognition tool, training should be performed using many samples showing expressions in many different circumstances. This means using samples involving multicultural subjects in different lighting conditions, and perhaps also lateral facial images, partially occluded images, or subjects wearing glasses.

Independent of particular image sets or preprocessing techniques used, though, a modular approach to facial expression recognition appears to be favorable over a non-modular approach.

References

Dailey, M., Cottrell, G., Padgett, C., & Adolphs, R. (2002). EMPATH: A neural network that categorizes facial expressions. *Journal of Cognitive Neuroscience*, *14*, 1158–1173.

Ekman, P. (1994). All emotions are basic. *The Nature of Emotions; Ekman, P. and Davidson, R.J., eds.*, pages 15–19.

Ekman, P., & Friesen, W. (1976). *Pictures of facial affect*. Consulting Psychologists Press, Palo Alto, CA.

Ekman, P., Friesen, W., & Hager, J. (2002). *Facial Action Coding System Investigator's Guide*. A Human Face, Salt Lake City, UT.

Ekman, P., & Friesen, W. V. (1978). *Facial Action Coding System*. Consulting Psychologist Press, Palo Alto, CA.

Gargesha, M., & Kuchi, P. (2002). Facial expression recognition using artificial neural networks. `http://www.public.asu.edu/~pkuchi/ExpressionRecognition.pdf`.

Kanade, T., Cohn, J. F., & Tian, Y. (2000). Comprehensive database for facial expression analysis. *Proceedings of the Fourth IEEE International Conference on Automatic Face and Gesture Recognition (FG'00), March 2000, Grenoble, France*.

Kohavi, R., & Provost, F. (1998). Glossary of terms. *Machine Learning*, *30*, 271–274.

Lyons, M. J., Akamatsu, S., Kamachi, M., & Gyoba, J. (1998). Coding facial expressions with Gabor wavelets. *Proceedings of the Third IEEE International Conference on Automatic Face and Gesture Recognition, April 14-16 1998, Nara Japan*, pages 200–205.

Pantic, M., & Rothkrantz, L. (2000). Automatic analysis of facial expressions: the state of the art. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *22*, 1424–1445.

Russell, J. A. (1980). A circumplex model of affect. *Journal of Personality and Social Psychology*, *39*, 1161–1178.

Zhang, Z., Lyons, M., Schuster, M., & Akamatsu, S. (1998). Comparison between geometry-based and Gabor wavelets-based facial expression recognition using multi-layer perceptrons. *Proc. Int'l Conf. Automatic Face and Gesture Recognition*, 454–459.

Reliability yields Information Gain

I.G. Sprinkhuizen-Kuyper KUYPER@CS.UNIMAAS.NL
E.N. Smirnov SMIRNOV@CS.UNIMAAS.NL
IKAT, Universiteit Maastricht
P.O.BOX 616, 6200 MD Maastricht, The Netherlands

G.I. Nalbantov NALBANTOV@FEW.EUR.NL
ERIM, Erasmus University Rotterdam

Abstract

In this paper we prove that the reliability of the classifications of individual instances, provided by a classifier, results in information gain with respect to the accuracy of the classifier. We illustrate this result using our new approach to classification reliability called version space support vector machines.

1. Introduction

In the last ten years machine-learning classifiers have been applied to various classification problems (Kukar & Kononenko, 2002). Nevertheless, almost no classifiers have been employed in real applications, especially in critical domains. The main reason is that it is difficult to determine whether a classification assigned to a particular instance is reliable or not.

There are two groups of approaches to classification reliability. The approaches in the first group estimate some parameter(s) that are related to classification reliability. Then, they learn a threshold on that parameter(s) to decide whether an instance classification is reliable.

The approaches in the first group differ in the parameters estimating the reliability. The oldest approach used the posterior probability of the predicted class as a reliability parameter (Duda et al., 2000). Newer approaches use reliability parameters based on the theory of randomness (cf. (Kukar & Kononenko, 2002; Papadopoulos et al., 2002; Proedru et al., 2002)). The underlying scheme to compute these parameters is as follows: (1) measure the level of randomness of the training data; (2) classify an instance; (3) label the instance with the new class and add the instance to the training data; (4) measure the level of randomness of the updated training data. The reliability parameter is inversely proportional to the difference between the two levels of randomness of the training data.

The second group of the approaches to classification reliability is based on version spaces (Mitchell, 1997; Smirnov, 2001). The key idea assumes that we can maintain version spaces containing (close approximations of) the target classifiers. If the assumption is correct for the training data under consideration, the classification rule of unanimous voting applied on these version spaces guarantees that if an instance is classified, then the instance is classified correctly; i.e., the classification assigned to each instance has reliability equal to one.

In (Smirnov et al., 2004) we proposed the first approach from the second group. The approach is applicable for binary classification tasks. It is a combination of version spaces and support vector machines (SVM) (Vapnik, 1998), and it is called version space support vector machines (VSSVM). VSSVM employ the training-data representation of version spaces (Hirsh, 1992; Smirnov, 2001). The unanimous-voting classification rule for this representation is based on testing version spaces for collapse. Testing is realised with SVM.

We conducted experiments with VSSVM on datasets from the UCI ML repository (Blake & Merz, 1998). We found an acceptable coverage and the accuracy on the coverage was 1 (100%), so all instances where classified with a reliability of 1. We computed the information gain with respect to SVM and found for all datasets a considerable information gain.

In this paper we prove that the results of our experiments are not accidental, they show an important principle of classification reliability. The principle states that extra information on the reliability of the classifications of individual instances results in information

gain with respect to the information given by the accuracy of a classification algorithm.

The remainder of the paper is organized as follows. Section 2 formalizes the classification task. In section 3 the notions of entropy and information gain are summarized. Our main theorem on information gain is proved in section 4. Section 5 explains the information gain obtained by combining classifiers. In section 6 we introduce VSSVM. Our experiments with VSSVM are given in section 7. Section 8 concludes the paper.

2. Classification Task

In this paper we restrict ourselves to two-class classification tasks.

Assume that we have l training instances $\mathbf{x}_i$ in an n-dimensional Euclidian space R^n. Each training instance has a class label $y_i \in Y$, where $Y = \{-1, +1\}$. The class labels separate the training instances into two sets I^+ and I^- ($\mathbf{x}_i \in I^+$ iff $y_i = +1$; $\mathbf{x}_i \in I^-$ iff $y_i = -1$). Given a hypothesis space H of all possible functions h ($h : R^n \rightarrow Y$), the classification task is to find a function (classifier) h that accurately classifies future, unseen data.

Usually the performance of a classifier is measured by its accuracy. The accuracy is the proportion of correctly classified instances of a test set that has the independently identically distributed (i.i.d.) property, i.e. it is large enough, independent of the training set and has a distribution over the instances corresponding to the underlying distribution of all instances. So accuracy is a global property over a large number of instances.

In this paper we consider the classification reliability of individual instances. The reliability r_i of the classification of an individual instance $\mathbf{x}_i$ is an estimate (in the range [0,1]) that the classification of that instance is correct. Two extreme cases are $r_i = 1$, i.e. we know for sure that the classification is correct, and $r_i = 0$, i.e. we know for sure that this instance is incorrectly classified, and this instance belongs to the other class.

3. Entropy and Information Gain

In information theory, the notion of entropy is introduced using the notion of ensembles[1] (MacKay, 2003). An ensemble X is a triple $\langle x, A_X, P_X \rangle$, where the outcome x is the value of a random variable, which takes on one of a set of possible values, $A_X = \{a_1, a_2, \cdots, a_i, \cdots, a_M\}$, having probabilities $P_X = \{p_1, p_2, \cdots, p_M\}$, with $p(x = a_i) = p_i$, $p_i \geq 0$ and $\sum_{a_i \in A_X} p(x = a_i) = 1$. The entropy of an ensemble X is defined to be the average Shannon information content of an outcome:

$$H(X) = \sum_{x \in A_X} -p(x) log_2 p(x) \tag{1}$$

In machine learning the entropy of a classifier is an important measure of the quality of the classifier. In our paper we consider two types of entropy: class entropy and meta-class entropy.

Given a set K of classes and an i.i.d. test set S_0, the class entropy H is defined by formula 1 if the set A_X equals the set K and the probabilities $p_i \in P_X$ are the proportions of the instances in S_0 belonging to the classes in K.

Given an i.i.d. test set S_0 and a classifier A, the meta-class entropy $E(S_0|A)$ is defined by formula 1 if the set A_X equals $\{correct, incorrect\}$ with probabilities $p_1, p_2 \in P_X$ so that $p_1 = a$ and $p_2 = 1 - a$, where *(in)correct* is an event: "the classifier A is (in)correct", and a is the accuracy of the classifier A on the test set S_0.

For both types of entropy we can define the information gain of a classifier A with respect to an other classifier A' as the difference between the entropy for A' and the entropy for A. The lower the entropy, the higher the information gain.

If the set S_0 is divided into disjoint sets S_i, the meta-class entropy $E(S_0|A)$ can be computed as follows:

$$E(S_0|A) = \sum_{S_i \subseteq S_0} \frac{|S_i|}{|S_0|} E(S_i|A) \tag{2}$$

4. Information Gain of Reliability

In this section we consider a fixed i.d.d. test set S_0 of size N and some classifier(s) having accuracy a and/or reliability $r = (r_1, \cdots, r_N)$. To stress on accuracy and reliability, we will use in this section the notations E_a, for entropy based on accuracy a, and E_r, for entropy based on reliability r, and we will skip S_0 and the name of the classifier from our notation.

Consider a classifier without reliability information. If we know that its accuracy equals a, the entropy E_a of an i.i.d. test set equals

[1] Please note the difference of the notion of ensemble in information theory and machine learning.

$$E_a = -a\,log_2 a - (1-a)\,log_2(1-a) \quad (3)$$

Based on this accuracy we expect each instance of the test set to have a reliability r_i equal to a. If the test set consists of N instances, the entropy E_a equals

$$E_a = \frac{1}{N}\sum_{i=1}^{N}(-a\,log_2 a - (1-a)\,log_2(1-a)) \quad (4)$$

where $-a\,log_2 a-(1-a)\,log_2(1-a)$ is the contribution of each individual instance.

Suppose that the algorithm also outputs information on the reliabity r_i of each individual instance, such that the average reliability equals the accuracy

$$a = \frac{1}{N}\sum_{i=1}^{N} r_i \quad (5)$$

Each individual instance will contribute $-r_i\,log_2 r_i - (1-r_i)\,log_2(1-r_i)$ to the entropy E_r of the test set. So, the entropy of the test set will be equal to

$$E_r = \frac{1}{N}\sum_{i=1}^{N}(-r_i\,log_2 r_i - (1-r_i)\,log_2(1-r_i)) \quad (6)$$

The main theorem of this paper yields that the information gain, i.e. the difference between E_a and E_r is positive for all values of the reliabilities, such that equation 5 holds.

Theorem 1 *For all values of r_i, $0 \leq r_i \leq 1$, $i = 1, 2, \cdots, N$ such that equation 5 holds, the information gain IG is positive:*

$$IG = E_a - E_r \geq 0 \quad (7)$$

where the entropy E_r is given in equation 6 and the entropy E_a is given in equation 3. Moreover, $IG = 0$ only holds when $r_i = a$, for all $i = 1, 2, \cdots, N$.

Proof Let us introduce the notation $f(x) = -x\,log_2 x - (1-x)\,log_2(1-x)$ for the entropy function. This function has its range equal to $[0, 1]$. See figure 1.

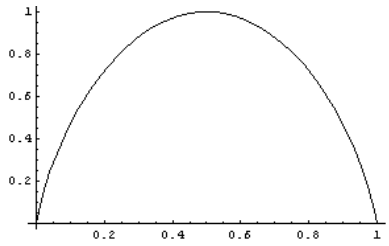


Figure 1. The entropy function $f(x) = -x\,log_2 x - (1-x)\,log_2(1-x)$.

We have to prove that $IG \geq 0$ for values $r_i \in [0, 1]$, $i = 1, 2, \cdots, N$ that satisfy equation 5. Let us consider the function IGG, obtained from equation 7 by replacing a in equation 3 by using equation 5:

$$IGG = f(\frac{1}{N}\sum_{i=1}^{N} r_i) - \frac{1}{N}\sum_{i=1}^{N} f(r_i)$$

Since the function $f(x)$ is strictly concave the inequality $IGG \geq 0$ is an immediate consequence of Jensen's inequality (see (MacKay, 2003)). Jensen's inequality also implies that $IGG = 0$ only if all r_i's are equal.

As a consequence the information gain $IG \geq 0$ and $IG = 0$ only holds when $r_i = a$, for all $i = 1, 2, \cdots, N$. □

5. Information gain by combining classifiers

An extreme case of classifiers that use classification-reliability information is the case of classifiers *Cov* that classify a part of the instances with reliability 1, while the remaining instances are not classified. An example is our algorithm VSSVM presented in section 6. The meta-class entropy of such a classifier *Cov* is computed according to formula 2. The classifier *Cov* divides the set S_0 into two sets S_1 and S_2: the set S_1 is the covered set (all the instances in S_1 are classified correctly by *Cov*); and the set S_2 is the non-covered set (all the instances in S_2 are not classified by *Cov*). Since the accuracy a of *Cov* on S_1 is 1, the meta-class entropy of the classifier on the set S_1 is 0 (see formula 1). Since the instances in S_2 are not classified, the meta-class entropy of the classifier on the set S_2 is 1. Thus, according to formula 2 the meta-class entropy $E(S_0|Cov)$ of a classifier *Cov* is $(1-c)$.

In order to get more information on the part of instances not covered by the classifier *Cov* it is natural to combine *Cov* with another classifier A classifying

all instances of S_0 with an accuracy a.

Consider the algorithm *CovA* that classifies the instances of S_0 that are covered by *Cov* according to *Cov* and the remaining instances according to A. The next theorem proves that the information gain of *CovA* is positive with respect to both classifiers *Cov* and A.

In the theorem we need that the accuracy of the classifier A is at least 0.5 in order increasing accuracy to result in decreasing entropy. This is no restriction at all for two-class problems, since an algorithm with accuracy below 0.5 immediately results in an algorithm with accuracy above 0.5 by inverting all classifications to the opposite class.

Theorem 2 *Given a classifier A with accuracy $a \geq 0.5$ and a classifier Cov with coverage $c > 0$. Suppose that each instance in the coverage of Cov is classified correctly with a reliability equal to 1. Then these algorithms can be combined to an algorithm CovA with information gain greater than those of A and Cov.*

Proof Let the classifier A have accuracy $a \geq 0.5$ and no reliability information. So, the entropy of A is given by $E(S_0|A) = f(a)$. Let the classifier *Cov* have coverage c, such that the reliability of the classification of each covered instance equals 1. So, the entropy of *Cov* is given by $E(S_0|C) = 1 - c$, since on the covered part we have complete information, while on the remaining part we have no information at all.

Consider the classifier *CovA* that uses A and *Cov* as described above. Let b be the proportion of S_0 that is differently classified by A and *Cov*. Since, the algorithm *CovA* follows for this proportion b the classification of *Cov*, the algorithm *CovA* has an accuracy $a' = a + b$.

The contribution to the entropy of *CovA* of the proportion classified by *Cov* equals 0.

Let us consider the remaining proportion $1 - c$ of S_0. In this proportion, a proportion $1 - a - b$ with respect to the the original set, thus $(1 - a - b)/(1 - c)$ with respect to the remaining part, is classified incorrectly by A, while a proportion $(a+b-c)/(1-c)$ is classified correctly by A. So, the contribution to the entropy of *CovA* of the proportion $1 - c$ equals $(1 - c)f((a + b - c)/(1 - c))$.

So, the entropy of *CovA* equals:

$$E(S_0|CovA) = (1 - c)f(\frac{(a + b - c)}{(1 - c)}) \quad (8)$$

The best classifier will classify the instances in the proportion $1-c$ according to the algorithm A if $a+b-c \geq 1 - a - b$, while it will invert these classifications if $a + b - c < 1 - a - b$.

Considering the entropy $E(S_0|CovA)$, it is smaller than $E(S_0|Cov)$ iff $a + b - c \neq 1 - a - b$. The entropy $E(S_0|CovA)$ is obtained from an algorithm with accuracy $a' \geq a$, by adding reliability information. By theorem 1 we find that $E(S_0|CovA) < f(a')$, and since $0.5 \leq a \leq a'$, we conclude that $E(S_0|CovA) < E(S_0|A)$.

So, we conclude that $E(S_0|CovA) < E(S_0|A)$ and $E(S_0|CovA) < E(S_0|Cov)$, with exception of the case that $a + b - c = 1 - a - b$. In this case $E(S_0|CovA) = E(S_0|Cov)$, but almost certainly another classifier A' will give information gain. □

6. Version Space Support Vector Machines

In this section we illustrate our findings from sections 4 and 5 using the example of version space support vector machines (VSSVM) (Smirnov et al., 2004). VSSVM are a new approach to classification reliability based on version spaces (Mitchell, 1997) and support vector machines (Vapnik, 1998). They do implicitly maintain the version space w.r.t. training data assuming that the version space does contain (close approximations of) the target function. Under this assumption applying the unanimous-voting classification rule means that if an instance is classified, then it is classified correctly. Thus, VSSVM assign a reliability equal to 1 to each instance they are able to classify.

To avoid problems with explicit version-space representations, VSSVM employs the training data representation (Hirsh, 1992; Smirnov, 2001). The unanimous-voting classification rule for this representation is based on testing version spaces for collapse. Testing is realised with SVM.

For our illustrative purposes we consider VSSVM in the next two subsections. In subsection 6.1 we define version spaces employed in VSSVM, formalise their classification rule, and prove that it can be implemented if it is possible to test version spaces for collapse. In subsections 6.2 and 6.3 we provide the classification algorithm of VSSVM and an example.

6.1. Version Spaces

Version spaces are sets of functions $h \in H$ consistent with training sets I^+ and I^- (Mitchell, 1997):

$$VS(I^+, I^-) = \{h \in H | cons(h, \langle I^+, I^- \rangle)\},$$

where *cons* is the consistency predicate defined as:

$$cons(h, \langle I^+, I^- \rangle) \leftrightarrow (\forall \mathbf{x}_i \in I^+ \cup I^-)(y_i = h(\mathbf{x}_i)).$$

The version-space classification rule is the unanimous voting. Given a nonempty version space $VS(I^+, I^-)$, an instance $\mathbf{x} \in R^n$ receives a classification $y \in Y \cup \{0\}$ as follows:

$$y = \begin{cases} +1 & \text{if}(\forall h \in VS(I^+, I^-))(h(\mathbf{x}) = +1) \\ -1 & \text{if}(\forall h \in VS(I^+, I^-))(h(\mathbf{x}) = -1) \\ 0 & \text{otherwise.} \end{cases}$$

The unanimous-voting rule assigns class $+1$ (-1) to the instance $\mathbf{x}$ if $VS(I^+, I^-)$ is nonempty and all functions h in $VS(I^+, I^-)$ assign the same class $+1$ (-1) to the instance. In all other cases, the class is unknown which is denoted by 0.

VSSVM apply the unanimous-voting rule for the training-data representation of version spaces (Hirsh et al., 1997; Smirnov, 2001). The implementation of the rule in this case requires testing whether the version space is empty. The whole process can be explained by theorem 3 (Hirsh et al., 1997; Smirnov, 2001). Assume that we have a nonempty version space $VS(I^+, I^-)$ and an instance $\mathbf{x}$ to be classified. Then, theorem 3 states that all the functions in $VS(I^+, I^-)$ assign class $+1$ (-1) to $\mathbf{x}$ if and only if the subset $VS(I^+, I^- \cup \{\mathbf{x}\})$ $(VS(I^+ \cup \{\mathbf{x}\}, I^-))$ of $VS(I^+, I^-)$ of which the functions assign class -1 $(+1)$ is empty. In all other cases, the class of $\mathbf{x}$ is unknown.

Theorem 3 *If $VS(I^+, I^-)$ is nonempty, then for an arbitrary instance $\mathbf{x} \in R^n$:*

$$(\forall h \in VS(I^+, I^-))(h(\mathbf{x}) = +1) \leftrightarrow VS(I^+, I^- \cup \{\mathbf{x}\}) = \emptyset,$$

$$(\forall h \in VS(I^+, I^-))(h(\mathbf{x}) = -1) \leftrightarrow VS(I^+ \cup \{\mathbf{x}\}, I^-) = \emptyset.$$

6.2. Classification Algorithm

The classification algorithm of VSSVM implements the unanimous-voting rule on version spaces in the hypothesis space H. It is based on theorem 3. To test whether version spaces are empty SVM are employed.

Given a set of training instances $I^+ \cup I^-$, SVM with cost parameter $C > 0$ find the hyperplane that minimizes the sum of the inverse of the margin between (most instances of) the sets I^+ and I^- and C times the errors introduced by those points of I^+ and I^- that are at the wrong side of the margin. Note that points at the wrong side of the margin are classified correctly by SVM as long as they are between the hyperplane found by SVM and the hyperplane defining the margin of their class (see e.g. (Burges, 1998).

The classification algorithm is given in figure 2. The algorithm input is: training data sets I^+ and I^-; an instance $\mathbf{x}$ to be classified; and the cost parameter C of SVM. The algorithm outputs the classification of $\mathbf{x}$: $+1$, -1, or 0.

The classification algorithm starts by building a hyperplane $h(C, \langle I^+, I^- \rangle)$. If $h(C, \langle I^+, I^- \rangle)$ is inconsistent with $\langle I^+, I^- \rangle$, then the version space $VS(I^+, I^-)$ is empty in the hypothesis space H for the value of the parameter C and according to the unanimous-voting rule the algorithm returns 0; i.e., the classification of $\mathbf{x}$ is unknown. If the hyperplane $h(C, \langle I^+, I^- \rangle)$ is consistent with $\langle I^+, I^- \rangle$, then the version space $VS(I^+, I^-)$ is nonempty in H for the value of the parameter C. In this case the algorithm builds hyperplanes $h(C, \langle I^+, I^- \cup \{\mathbf{x}\} \rangle)$ and $h(C, \langle I^+ \cup \{\mathbf{x}\}, I^- \rangle)$. If $h(C, \langle I^+, I^- \cup \{\mathbf{x}\} \rangle)$ is inconsistent with $\langle I^+, I^- \cup \{\mathbf{x}\} \rangle$ and $h(C, \langle I^+ \cup \{\mathbf{x}\}, I^- \rangle)$ is consistent with $\langle I^+ \cup \{\mathbf{x}\}, I^- \rangle$, then $VS(I^+, I^- \cup \{\mathbf{x}\})$ is empty and $VS(I^+ \cup \{\mathbf{x}\}, I^-)$ is nonempty in H for the value of the parameter C. This means that all the hyperplanes in $VS(I^+, I^-)$ assign class $+1$ to $\mathbf{x}$. Thus, by theorem 3 the algorithm assigns class $+1$ to $\mathbf{x}$. If the class $+1$ cannot be assigned, the algorithm checks analogously if it can assign class -1. If both classes cannot be assigned, the algorithms returns 0; i.e., the classification of $\mathbf{x}$ is unknown.

If there is an assumption that the version space contains (close approximations of) the target hyperplane (classifier), the classification algorithm of VSSVM returns implicitly the reliability of processed instances. More precisely,

- if an instance is classified by the classification algorithm, then all the hyperplanes in the version space assign a classification C to the instance. Since (a close approximation of) the target hyperplane is in the version space, the instance receives the same classification C by (close approximations of) the target hyperplane. This means that the classification C of the instance is indeed correct and the reliability of the classification C is 1.
- if an instance is not classified by the classification algorithm, then the hyperplanes in the ver-

Input: An instance $\mathbf{x}$ to be classified;
Training data sets I^+ and I^-;
The parameter C of SVM;
Output: classification of $\mathbf{x}$;

Build a hyperplane $h(C, \langle I^+, I^- \rangle)$;
if $\neg cons(h(C, \langle I^+, I^- \rangle), \langle I^+, I^- \rangle)$ **then return** 0;
Build a hyperplane $h(C, \langle I^+, I^- \cup \{\mathbf{x}\} \rangle)$;
Build a hyperplane $h(C, \langle I^+ \cup \{\mathbf{x}\}, I^- \rangle)$;
if $\neg cons(h(C, \langle I^+, I^- \cup \{\mathbf{x}\} \rangle), \langle I^+, I^- \cup \{\mathbf{x}\} \rangle)$ and
$cons(h(C, \langle I^+ \cup \{\mathbf{x}\}, I^- \rangle), \langle I^+ \cup \{\mathbf{x}\}, I^- \rangle)$ **then return** $+1$;
if $cons(h(C, \langle I^+, I^- \cup \{\mathbf{x}\} \rangle), \langle I^+, I^- \cup \{\mathbf{x}\} \rangle)$ and
$\neg cons(h(C, \langle I^+ \cup \{\mathbf{x}\}, I^- \rangle), \langle I^+ \cup \{\mathbf{x}\}, I^- \rangle)$ **then return** -1;
return 0.

Figure 2. The Classification Algorithm of VSSVM.

sion space assign different classifications to the instance and we cannot determine the classification assigned by (a close approximation of) the target hyperplane (remember we can reason only about the version space as a whole). Thus, in this case we can think that the instance receives both classifications but their reliability is 0.5.

6.3. Example

We illustrate our classification algorithm on a classification task: the space H is the set of all oriented lines in R^2, and training data consist of the sets $I^+ = \{(1,0),(2,0),(1,1),(2,1)\}$ and $I^- = \{(-1,0),(-2,0),(-1,1),(-2,1)\}$ (see figure 3). For large C ($C = +\infty$) only the points to the right of the three line segments through the training points $(1,0)$ and $(1,1)$ will be classified as positive and the corresponding region to the left of the three line segments through the training points $(-1,0)$ and $(-1,1)$ will be classified as negative. The version space in this case consists of all the lines not intersecting the line segments shown. Running our algorithm with $C = 30$ results in the classifications in figure 3: positively classified: $+$, negatively classified: $*$, and not classified: $\square$. It is clear from the figure that for $C = 30$ the version space is smaller and thus the coverage is larger, than for $C = +\infty$. In the figure some unclassified points (e.g., the point $(-1, -0.1)$) are expected to be classified. Such points can correspond to solutions of the SVM that are not uniquely determined. In such a case the hyperplane defining the solution can be translated a bit while resulting in the same minimum of the inverse of the margin and the cost term, resulting in separation for some solutions while other solutions will not separate.

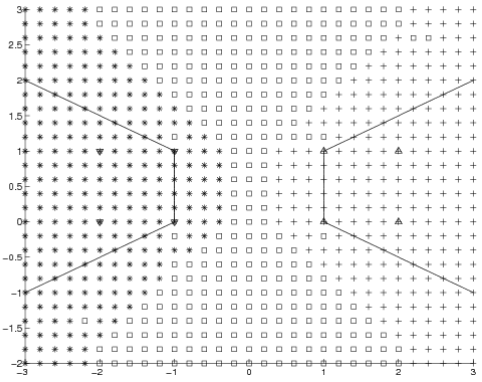


Figure 3. Illustration of the version spaces for $C = +\infty$ (bounded by the lines) and $C = 30$ (I^+ marked by $\triangle$, I^- marked by ∇, positively classified: $+$, negatively classified: $*$, and not classified: $\square$).

7. Experiments

We implemented VSSVM in WEKA (Witten & Frank, 2000) using the SMO implementation of SVM (Platt, 1998). We experimented with VSSVM using polynomial function kernels (P) and radial-basis function kernels (RBF). The method for evaluation was the leave-one-out method. We searched for values of the parameters (E (exponent) and C for P and G (gamma) and C for RBF) resulting in the highest coverage for the leave-one-out method. In table 1 we provide the best results we obtained for 7 datasets (from the UCI ML repository (Blake & Merz, 1998)) with VSSVM together with the results of SVM.

To measure the performance of VSSVM we computed the information gain of VSSVM w.r.t. to SVM (we used the same parameter settings for SVM and

Data Set	Parameters	c_{vssvm}	a_{vssvm}	a_{svm}	e_{vssvm}	e_{svm}	IG
Heart-Statlog	P, E=2.0, C=1730	0.563	1.0	0.730	0.42	0.84	0.42
Heart-Statlog	RBF, G=0.2 , C=2182	0.407	1.0	0.737	0.59	0.83	0.24
Hepatitis	P, E=1.4, C=11.7	0.800	1.0	0.800	0.00	0.72	0.72
Hepatitis	RBF, G=0.02 , C=2140	0.697	1.0	0.819	0.29	0.68	0.39
Horse Colic	P, E=1.3, C=154.5	0.508	1.0	0.783	0.49	0.75	0.26
Horse Colic	RBF, G=0.015 , C=12030	0.549	1.0	0.791	0.45	0.74	0.29
Ionosphere	P, E=1.1, C=5200	0.778	1.0	0.892	0.22	0.49	0.27
Ionosphere	RBF, G=0.05, C=4030	0.772	1.0	0.903	0.22	0.46	0.24
Labor	P, E=1.25, C=1.17	0.842	1.0	0.877	0.12	0.54	0.42
Labor	RBF, G=0.02, C=61	0.842	1.0	0.930	0.16	0.37	0.21
Sonar	P, E=1.0, C=3340	0.707	1.0	0.740	0.15	0.83	0.68
Sonar	RBF, G=0.65, C=0.664	0.625	1.0	0.856	0.36	0.59	0.23
W. Breast Cancer	P, E=3, C=58.6	0.850	1.0	0.936	0.15	0.34	0.19
W. Breast Cancer	RBF, G=0.8 , C=70.9	0.825	1.0	0.943	0.16	0.32	0.16

Table 1. Coverage c_{vssvm} of VSSVM and accuracy a_{svm} of SVM. The accuracy a_{vssvm} of VSSVM on the coverage is 1. The coverage of SVM is 1. P is a polynomial kernel with exponent E. RBF is a radial-basis function kernel with parameter G. The cost parameter is C. The other variables are: e_{vssvm}, the entropy of VSSVM when SVM classifies the instances not covered by VSSVM; e_{svm}, the entropy of SVM, and IG, the information gain of VSSVM w.r.t. SVM.

VSSVM; see table 1). The entropy of SVM was computed using the proportions of correctly and incorrectly classified instances estimated by the accuracy a_{svm}. The coverage of VSSVM is denoted by c_{vssvm} The entropy of VSSVM was computed as the weighted sum of the entropy of the set of reliably classified instances and the entropy of the set of unclassified instances (according to formula 2). Note that the entropy of the sets of reliably classified instances is 0 since the accuracy of VSSVM on the sets of reliably classified instances is 1. In order to classify the instances not covered by VSSVM using SVM, we remark that each instance in the coverage of VSSVM will be correctly classified by the SVM algorithm with the same parameters, since that algorithm is used in VSSVM to decide on the separability. Thus, for this combination of algorithms we have $c_{vssvm} \leq a_{svm}$. Thus the proportion $a_{svm} - c_{vssvm}$ of the original set is correctly classified in that set, while the proportion $1 - a_{svm}$ is incorrectly classified. So the accuracy a_{nc} on the set not covered by VSSVM equals $(a_{svm} - c_{vssvm})/(1 - c_{vssvm})$, resulting in a reliability $r_i = a_{nc}$, corresponding to equation 5. The contribution of the set not covered by VSSVM to the entropy is equal to:

$$(1 - c_{vssvm})(-a_{nc}\, log_2 a_{nc} - (1 - a_{nc})\, log_2(1 - a_{nc})).$$

All cases in table 1 resulted in a considerable information gain. We especially mention the hepatitis dataset (the case of polynomial kernel) of which the information gain is 0.72 (we even obtained perfect information!) and the labor dataset (the case of polynomial kernel) of which the information gain is 0.42.

8. Conclusion

For practical application of machine learning algorithms knowledge about the reliability of individual instances is absolutely necessary, since decisions have to be taken over individual cases.

This paper proves that reliability information results in information gain. This result is illustrated by a new approach to classification reliability called version space support vector machines (VSSVM). The experiments show that VSSVM achieve an accuracy of 1 on the instances they are able to classify for data sets from the UCI repository. The information gain of VSSVM w.r.t. SVM was computed and was considerable for the data sets investigated.

A research question for future research is how the information gain provided by the reliability information can be used to improve the accuracy of the existing machine learning algorithms. We plan to investigate this question by exploiting combinations of VSSVM based on ensemble techniques.

In addition, we foresee two future directions of research that are specific for VSSVM. The first one is to extend VSSVM for classification tasks with more than two classes. The second direction is to apply VSSVM when it is not possible to find consistent hyperplanes w.r.t. the training data. In this context we consider to apply the generalized version spaces (Mitchell, 1997).

9. Acknowledgements

We thank the anonymous referees for their valuable comments.

References

Blake, C., & Merz, C. (1998). UCI repository of machine learning databases.

Burges, C. (1998). A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, *2*, 121–167.

Duda, R., Hart, P., & Stork, D. (2000). *Pattern classification*. Willey. second edition.

Hirsh, H. (1992). Polynomial-time learning with version spaces. *Proceedings of the Tenth National Conference on Artificial Intelligence (AAAI-92)* (pp. 117–122). Menlo Park, CA: AAAI Press.

Hirsh, H., Mishra, N., & Pitt, L. (1997). Version spaces without boundary sets. *Proceedings of the Fourteenth National Conference on Artificial Intelligence (AAAI-97)* (pp. 491–496). Menlo Park, CA: AAAI Press.

Kukar, M., & Kononenko, I. (2002). Reliable classifications with machine learning. *Proceedings of the 13th European Conference on Machine Learning (ECML-2002)* (pp. 219–231). Springer.

MacKay, D. J. C. (2003). *Information theory, inference and learning algorithms*. Cambridge University Press.

Mitchell, T. (1997). *Machine learning*. New York, NY: McGraw-Hill.

Papadopoulos, H., Proedru, K., Vovk, V., & Gammerman, A. (2002). Comparing the bayes and typicalness frameworks. *Proceedings of the 13th European Conference on Machine Learning (ECML-2002)* (pp. 345–356). Springer.

Platt, J. (1998). *A fast algorithm for training support vector machines* (Technical Report). Microsoft Research.

Proedru, K., Nouretdinov, I., Vovk, V., & Gammerman, A. (2002). Transductive confidence machines for pattern recognition. *Proceedings of the 13th European Conference on Machine Learning (ECML-2002)* (pp. 381–390). Springer.

Smirnov, E. (2001). *Conjunctive and disjunctive version spaces with instance-based boundary sets*. Doctoral dissertation, Department of Computer Science, Maastricht University, Maastricht, The Netherlands.

Smirnov, E., Sprinkhuizen-Kuyper, I., & Nalbantov, G. (2004). Unanimous voting using support vector machines. *BNAIC-2004: Proceedings of the Sixteenth Belgium-Netherlands Conference on Artificial Intelligence* (pp. 43–50).

Vapnik, V. (1998). *Statistical learning theory*. NY: John Wiley.

Witten, I., & Frank, E. (2000). *Data mining: Practical machine learning tools and techniques with java implementations*. Morgan Kaufmann.

Reinforcement Learning using Optimistic Process Filtered Models

Funlade T. Sunmola FTS@CS.BHAM.AC.UK
Jeremy L. Wyatt JLW@CS.BHAM.AC.UK
School of Computer Science, University of Birmingham, Edgbaston, Birmingham B15 2TT,UK.

Abstract

An important problem in reinforcement learning is determining how to act while learning sometimes referred to as the exploration-exploitation dilemma or the problem of optimal learning. The problem is intractable, usually solved through approximation such as by being optimistic in the face of uncertainty. In environments with inherent determinism, arising for example from known process templates, acting conforms to certain acceptable conventions that limit exploration. We present an algorithm for the learning problem in which action selection is through optimistic models filtered to conform to conventions specified by a process language. We show results to illustrate the approach and its benefits for task transfers.

1. Introduction

A reinforcement learning (RL) agent learns by iteratively performing actions in the world and using resulting experiences to decide future actions. The experiences encapsulate among other things a reward or reinforcement on the actions taken. The environments in which agents act are usually stochastic and typically modelled as Markov Decision Processes (MDPs). Model-based RL agents build MDP models of their environments then plan and act using the models. Model-based approaches are advantageous in domains where real-world actions are expensive and computation time is relatively cheap.

Central to reinforcement learning is the challenges of exploration-exploitation trade off (Wyatt, 1997) and seeking to profit from structure inherent in the task environment (Dearden, 2000). The former is crucial for controlling action selection and is, typically for a reinforcement learning agent, an intractable problem of knowing how to act while learning so as to maximise lifetime performance. This inevitably involves balancing exploitation of current knowledge and exploration to discover new knowledge that might lead to better performance in the future. The latter contributes to alleviating the problems of scaling up to large domains by exploiting the natural organisation of the task environment.

In Markov Decision Processes (MDPs) the optimal Bayesian solution to the trade off problem is well known, but intractable (Martin, 1967; Bellman, 1961). Many approximate ways of dealing with the trade off in RL have been proposed. A new heuristic method was recently introduced (Wyatt, 2001) that brings together ideas from several recent approaches (Kaelbling, 1990; Wiering & Schmidhuber, 1998; Kearns & Singh, 1991). The heuristics provides a framework for exploration control in reinforcement learning based on optimistic model selection (OMS) from a density over possible models. The heuristics was shown to outperform existing methods when optimised. A structured version of the heuristics for factored MDPs using dynamic Bayesian networks is presented in (Sunmola & Wyatt, 2003). Essentially, through OMS, it is possible to perform a structured optimistic search on a density of possible process models while allowing a prior distribution over the space of models to be incorporated in the learning steps.

In this paper we study the leverage offered to the agents while learning to carry out a new task in an environment with inherent determinism arising, for example, from known process templates, value and policy constraints acquired typically from experiences on related tasks. We focus here on process (transition) model determinism and the constraints imposed on possible process models due to availability of conventions to which the learnt process models must conform. The templates are specified by a process (action) language to which there is an underlying context free grammar. The process templates provide added information and it would seem likely that acting based on optimistic models filtered through the templates should improve learning.

The paper is organised as follows: in section 2 we start by introducing reinforcement learning and the exploration control problem. Also contained in the section is an overview of the standard OMS algorithm. We concentrate on explicitly represented state-based MDPs. In section 3 we discuss the idea of constraining exploration and present a new OMS algorithm called *most optimistic first* that allows for process filtering. Section 4 describes empirical results on a circular flags world task and the paper is concluded in section 5 with directions for future research.

2. Reinforcement Learning

Our agent is learning to control a stochastic environment modelled as an MDP. An explicitly represented, state based, MDP is simply a 4-tuple (S, A, P, R). In the tuple, S is a set of distinct states, A is a set of actions, $p^a_{ij} \in P$ is a transition function that captures the probability of reaching state j after executing an action a at state i such that $i, j \in S$, and $r \in R$ is a reward function mapping S into real-valued rewards. The state transition coefficients p^a_{ij} obey standard stochastic constraints hence they have the following properties $0 \leq p^a_{ij} \leq 1$ and $\sum_j p^a_{ij} = 1$. The decision problem for an agent in the MDP is to find a policy π that optimises the expected discounted total reward $V = E(\sum_{t=1}^{\infty} \gamma^{t-1} r_t)$, where r_t is the reward received t steps into the future and $\gamma \in [0, 1]$ is a discount factor.

2.1. Exploration Control Problem

In Bayesian formulation of the problem, we assume that there is a space $\mathcal{P}$ of possible transition functions (parametric models) for the MDP and that there exists a prior probability density over this space. Given that a state $i \in S$ in the process has N possible succeeding states when an action a is taken, then the transition function from that state action pair is a multinomial distribution over the outcomes:

$$\vec{p}_i{}^a = \{p^a_{i1}, p^a_{i2}, \ldots, p^a_{iN}\} \tag{1}$$

The possible transition functions from i, a are the possible $\vec{p}_i{}^a$.

A convenient, natural conjugate, choice of prior distribution over the $\vec{p}_i{}^a$ is Dirichlet density:

$$f(\vec{p}_i{}^a | \vec{m}^a_i) = \frac{\Gamma(\sum_{j=1}^N m^a_{ij})}{\prod_{j=1}^N \Gamma(m^a_{ij})} \prod_{j=1}^N (p^a_{ij})^{m^a_{ij}-1} \tag{2}$$

and the density is parameterised by the $m^a_{ij} > 0$ for all j. The transition functions $\vec{p}_i{}^a$ may be chosen based on prior information available for the process or initialised to non-informative uniform values.

We observe process transitions. From our choice of likelihood function and prior distribution, it follows directly that the ensuing posterior distribution will be Dirichlet too with parameters $m^{a''}_{ij}$. For a single transition $i \overset{a,r}{\rightsquigarrow} j$ the update rule is $m^{a''}_{ij} = m^{a'}_{ij} + 1$. The density for the multi-state case follows directly from this since the densities over the one step transition functions for all state action pairs are independent. The density $f(P|M)$ for a possible transition function $P \in \mathcal{P}$ for the MDP is therefore simply the product of the $f(\vec{p}_i{}^a | \vec{m}^a_i)$ over all i. The density is parameterised by the matrix $M = [m^a_{ij}]$ where $M \in \mathcal{M}$. In a Bayesian framework, we choose a prior matrix M' which specifies our prior density over the space of possible models. The additional information from a sequence of observations is captured in a count matrix F. The posterior density given these observations is therefore simply parameterised by $M'' = M' + F$.

Given the usual squared error loss function, the Bayesian estimator of expected return under the optimal policy is the expectation of the value function $\tilde{V}_i$ in our MDP with unknown transition probabilities:

$$V_i(M) = E[\tilde{V}_i | M] = \int_{\mathcal{P}} V_i(P) f(P|M) dP \tag{3}$$

where $V_i(P)$ is the value of i given the transition function P. We know from the central result of both Bellman and Martin that when this integral is evaluated we transform our problem into one of solving an MDP with unknown transition probabilities, defined on the information space $\mathcal{M} \times \mathcal{S}$:

$$V_i(M) = max_a \ \{\sum_i \bar{p}^a_{ij}(M)(r^a_{ij} + \gamma V_j(T^a_{ij}(M)))\} \tag{4}$$

in which, for convenience, the transformation on M due to a single observed transition $i \overset{a_t,r_t}{\rightsquigarrow} j$ is denoted $(T^a_{ij}(M))$, $\bar{p}^a_{ij}(M)$ is the marginal expectation of the Dirichlet, and r^a_{ij} is the reward associated with the transition $i \overset{a,r}{\rightsquigarrow} j$. This shows how the Bayesian estimate of value elegantly incorporates the value of future information. The optimal solution to our exploration-exploitation trade off problem is thus to act greedily with respect to the Bayes Q-values. The solution to the problem is however intractable because it involves dynamic programming over a tree of information states. Approximate solutions may be found by either simply using the certainty equivalent (CE) estimate constructed by replacing $T^a_{ij}(M)$ with M, approximate the integral by random sampling, or select models optimistically in the face of uncertainty.

2.2. Optimistic Model Selection

The goal of model selection is to identify the one model, from a set of competing models, that best captures the regularities underlying the cognitive process of interest. The OMS method (Wyatt, 2001) integrates ideas from a popular family of approximate approaches to the exploration-exploitation trade off which typically use some instantiation of the heuristic 'be optimistic in the face of uncertainty' (Kaelbling, 1990; Wiering & Schmidhuber, 1998; Meuleau & Bourgine, 1999). OMS integrates the instantiation idea with the Bayesian view of exploration by selecting an optimistic model P_{opt} from $\mathcal{P}$ using probability intervals calculated based on $f(P|M)$. Wyatt suggests two ways of selecting P_{opt}, which were termed simple and full OMS. In simple OMS we are optimistic only about hypothesised transition to an imaginary terminal state. In full OMS we can be optimistic about transitions to other states too.

2.3. Full OMS

Following an initialisation step (step 1), the main loop (steps 2-5) of an OMS-oriented solution approach to the exploration-exploitation trade off problem is organised as follows:

Step 1: initialise process values and model parameters.

Step 2: select action based on the current value estimate, interact with environment and observe transition.

Step 3: update parameter matrix in standard way, based on the observed transition.

Step 4: select an optimistic model P_{opt}.

Step 5: do value backup using P_{opt}.

In the full OMS, P_{opt} is selected as follows. Given state action pair i, a we order its successors by the current estimate of the value function, in descending order. We then calculate the lower and upper bounds of the $(1-\alpha)$ probability interval for each transition. We construct an optimistic transition function by allocating the maximum probability mass to states early in the ordering while keeping all probabilities within their lower and upper bounds. For example, consider a process shown in Figure 1 at state $i=1$ under action a and with four observed successor states $j=[2,4,3,1]$. The successor states are ordered according to their value function estimates in descending order and we select a full optimistic model $\vec{p}^{\,a}_{opt,1}=[0.38,0.32,0.20,0.10]$.

Once $\vec{p}^{\,a}_{opt,i}$ is selected, the state action pair i, a is then backed up using the optimistic one step transition function that results. The relevant Bellman equation is:

$$\xi^a_i(M)=\sum_j p^a_{opt,ij}(M)(r^a_{\mathbf{ij}}+\gamma max_{a'}\{\xi^{a'}_j(M)\}) \quad (5)$$

where $p^a_{opt,ij}(M)$ are the transition probabilities according to P_{opt}. The agent selects the action with the highest optimistic value ξ^a_i.

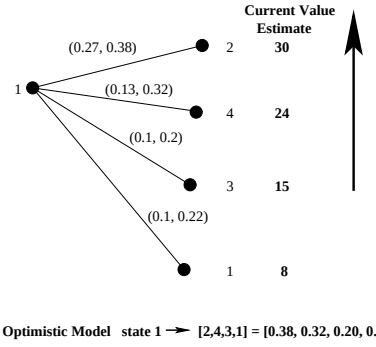


Figure 1. An example process at state $i=1$ under action a with four successor states. Bounds on probabilities for each transition are shown as (lowerbound, upperbound).

3. Constraining Exploration

We consider feasible regions $\mathcal{P}' \in \mathcal{P}$ defined as the set of points in the model space for which the MDP satisfies all specifications on its behavior. By selecting models that belong only to the feasible region we constrain exploration to relevant parts in the space of possible process models. The integral of equation 3 is constrained to the following.

$$V_i(M)=E[\tilde{V}_i|M]=\int_{\mathcal{P}'} V_i(P)f(P|M)dP \quad (6)$$

$\mathcal{P}' \in \mathcal{P}$ represents the feasible region where the model parameters are such that the MDP and, consequently, the learning agent satisfy behavioral requirements.

In this work, we do not attempt to evaluate the above integral. Instead, the feasible region is approximated using a process template and we attempt to select optimistic models that comply with the approximated feasible region. The optimal policy is the point P at which the expected return $V_i(M)$ is maximised given the feasible region defined by the process template. In the sections that follow, we introduce a process description language that specifies the template and describe an algorithm for selecting conforming optimistic models.

3.1. Process Description Language

A process description language is a specialised language (or grammar) that helps express an MDP precisely and compactly in a system of words that can be understood by an agent and its learning algorithm. Central to the process description of an MDP is an action language component. The action description language (Gelfond & Lifschitz, 1993) which was introduced in 1992 became very popular particularly for model checking (Bultan, 2000) and has been designed to come up with a specification language which describes the effect of actions in a simple, elegant and natural way. In general, action languages are formal models specifically designed to specify actions and their effects.

Formal languages (or automata) constitute a cornerstone of computer science (Salomaa, 1973). A formal language is normally defined by an alphabet and formation rules. The alphabet of a formal language is a set of symbols on which the language is built. Some of the symbols in an alphabet may have a special meaning. The formation rules specify which strings of symbols should be considered as well-formed, often called words, expressions, formulas, or terms. The formation rules are usually recursive, they postulate that such and such expressions belong to the language in question or establish how to build well-formed expressions from other expressions belonging to the language. Formation rules are sufficient for defining simple languages. More syntactically complex languages can be defined by means of finite automata, grammars, regular expressions or certain operations.

In a grammar system we have an initial symbol and a set of rewriting rules, which state how a word is derived from another. A grammar defining formal language $\mathcal{L}$ is a quadruple (L_N, L_T, L_R, L_S), where L_N is a finite set of nonterminals, L_T is a finite set of terminal symbols, L_R is a finite set of productions, and L_S is an element of L_N. The set L_T of terminal symbols is L's alphabet. Nonterminals are symbols representing language constructs. The sets L_N and L_T should not intersect. L_S is called the start symbol. Productions are rules of the form: $prod_a \rightarrow prod_b$, where both $prod_a$ and $prod_b$ are strings of terminals and nonterminals, $prod_a$ contains at least one nonterminal.

3.2. Optimistic Process Filtered Models

Filtering is the problem of estimating the state of a system as a set of observations becomes available online. A process filter is predicated on the ideas of the general filtering problem. Given a process template based upon a process description language $\mathcal{L}$, a process filter estimates the importance of a process model P_i for an MDP, attaching weight w_i to the model. A weight is conceptually the probability of the process model given the language.

Table 1. Main loop of the OMS algorithm with Process Filtering

Optimistic Process Filtered Model Selection

begin
Initialise t, prior parameter matrix m, process description language $\mathcal{L}$, and exploratory value function ξ
repeat
observe current state x.
choose an action a based on value function ξ_x, breaking ties randomly.
do a and observe transition $x \overset{a,r}{\rightsquigarrow} y$.
update parameter matrix m.
until your ARTDP algorithm stops
choose state i
for each action b
find $\vec{p}_{opt}^{\,b}$ using most-optimistic-first
update ${\xi_i}^b$ using equation 5
update $t \rightarrow t+1$.
end

$$w_i = Pr(P_i|\mathcal{L}) \tag{7}$$

The model with maximum weight is selected as the most conforming process model. Feasible process models are approximated as those whose weights are greater than a specified threshold β i.e. $Pr(P_i|\mathcal{L}) > \beta$, with β typically small ≈ 0.

We add a filter component to the standard OMS algorithm. The filter component discourages process models which have a zero conformance probability. It takes as input an optimistic model, passes the model through the process language and return state transitions in the optimistic model that are not in conformance with the specified language. The main loop of the OMS algorithm incorporating process filtering in an asynchronous real time dynamic programming (ARTDP)framework is shown in table 1. Any other asynchronous methods may be used with the OMS-algorithms in a convenient way.

Search of the feasible model space is conducted by using a full OMS procedure to first obtain the most optimistic model for the MDP at the current time step. The most optimistic model is then passed through the process filter component. If all transitions of the optimistic model conform to the language the most optimistic model is selected and the state action pair is

Table 2. A description of process shown in Figure 1.

Language fragment for a sample process
** greatestsuccstate is the state that has the largest transition probability in the set of successors to the current state under action (act) ** Prob($i \overset{act}{\rightsquigarrow} j$) is the transition probability for state i to state j under action act
var := {greatestsuccstate} operators := {mult,>}
state_1transitions(act): greatestsuccstate := 2; Prob($1 \overset{act}{\rightsquigarrow} 1$) > mult(0.5, Prob(greatestsuccstate));

then backed up using the selected optimistic model. Otherwise, if there are non-conforming transitions, we reapply the full OMS algorithm on the non-conforming transitions.

To illustrate, assume the process language of table 2 applies to the example of Figure 1. The language specifies transition from state 1 to state 2 (i.e. $1 \overset{act}{\rightsquigarrow} 2$) as having the largest probability in the set of successors of state 1. Using the current optimistic probability estimates for transitions from state 1, the transition probability of $1 \overset{act}{\rightsquigarrow} 1$ is expected to be greater than $0.5 * 0.38 = 0.16$ according to the process language. The lower bound for the transition $1 \overset{act}{\rightsquigarrow} 1$ is reset to 0.16, defining a feasible region for the model. The current optimistic process model is updated to $\vec{p}^{\,act}_{opt,1} = [0.38, 0.32, 0.14, 0.16]$ as shown in Figure 2.

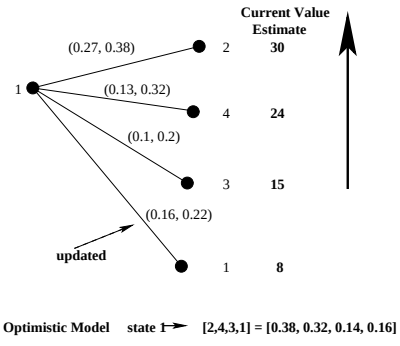


Figure 2. An extension of example shown in Figure 1, allowing for the language fragment described in table 2

4. Empirical Results

To test the optimistic methods presented in this paper, we used the optimistic model selection algorithms with and without process filtering in learning how to behave in a reinforcement driven environment. We employed a flags world domain similar in principle to the one described in (Dearden, 2000) where an agent attempts to collect flags and get them to a goal.

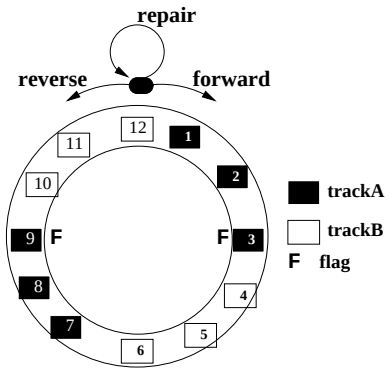


Figure 3. The reinforcement learning task in a three action circular flags world composed of twelve cells and two binary flags positioned at cells 3 and 9. Cell 12 is the start and goal cell.

We devised a circular flags world shown in Figure 3 that contains twelve cells and two binary flags. The agent navigates the flags world using an automatic guided vehicle (agv) that may occasionally breakdown. At any given time, flags in the world may be either set or unset and the agv may have a status of either faulty or not faulty. In all, the flags world have $12 \times 2^2 \times 2 = 96$ possible states. In the flags world instance of Figure 3, the start and goal positions are located in cell 12 while the flags (marked 'F' in the Figure) are positioned in cells 3 and 9. The cells in the flags world are divided into sections which in our experiments we labeled as *track A* and *track B*. The agent receives reward on reaching the goal cell based on the number of flags collected.

Initially, the agent starts at the start cell with an agv that is not faulty and all flags unset. The agent navigates the world with three possible actions *forward, reverse*, and *repair* moving the agent forward to the next cell, backwards to the previous cell and carrying out repairs when the agv is faulty, respectively. The i'th flag is set only if the agv is not faulty and when the agent executes an action at the cell containing the flag. Once set, the flag may only be unset at the goal cell. All the flags are unset once the agent executes an action at the goal cell. Transitions from one cell to an-

Table 3. A description of the circular flags world instance used in the experiments

Circular flags world instance with 96 states and 2 flags.

** greatestsuccstate is the state that has the largest transition probability in the set of successors to the current state under action (act)
** Prob(i,act,j) is the transition probability for state i to state j under action *act*

action := { forward | reverse | repair }
operators := {move, not, override, tol, add, mult, toggle, trackfactor, ingoalcell, inflagcell, flagisset, faulty}
var := {greatestsuccstate,curstate,altsuccstate}

doingForwardReverse(action,curstate):
greatestsuccstate := move(action, curstate);
altsuccstate := move(toggle(action, 'action'), curstate);
if (inflagcell, not(flagisset), not(faulty))
greatestsuccstate:=override(greatestsuccstate,flagisset);
altsuccstate := override (altsuccstate, flagisset);
if (ingoalcell)
greatestsuccstate:=override(greatestsuccstate, not(flagisset));

if (faulty)
Prob(curstate,action,altsuccstate) := tol(0.14, 0.02);
if (action == 'forward')
Prob(curstate,action,curstate) := tol(0.2, 0.01);
else Prob(curstate,action,curstate) := tol(0.3, 0.01);
Prob(curstate,action,greatestsuccstate):=
add(mult(trackfactor(curstate,action),Prob(curstate, action, curstate)), Prob(curstate,action,altsuccstate));

doingRepair(action,curstate):
if (faulty)
if (ingoalcell) greatestsuccstate := toggle (override(curstate,not(flagisset)), 'fault');
else greatestsuccstate := toggle(curstate,'fault');
Prob(curstate,action,greatestsuccstate):=tol(0.945,0.013);
Prob(curstate,action,curstate) := tol(0.045, 0.013);

other is stochastic for all three actions. Inherent in the flags world are structural dependencies. For example, due to the flags, the probability of transitions from one cell to another does not depend on flags setting.

We carried out our experiments on a specific instance of the circular flags world of Figure 3. The task to learn is specified with a reward of 10 for setting a single flag and 100 for setting both flags. We choose a non-informative prior matrix with zero transitions for all entries except for transitions leading to an imaginary state which we set to 1. The template provides information about the process particularly at states in which the agv is faulty.

In addition, available to the agent, is a template (shown in table 3) that specifies a fragment of the process. The process fragment described in table 3 consists of two action modules. The doFowardReverse module applies to the forward and reverse actions, and the doingRepair module applies to the repair action. In each of the modules, given the current state (curstate) and the current action, the agent can determine the state (greatestsuccstate) which has the largest transition probability amongst the successor states to the current state. In deriving the greatestsuccstate, operators *move* and *override* are used. Move is a function that takes as input an action and the current state then returns the next state in the flags world when the action is performed successfully, ignoring requirements for flag settings. Override takes as input a state and a flag setting then returns a corresponding state in which the flag variable has the same setting as the flag setting parameter passed to it.

Also specified in the action modules are guides for the transition probability from the current state to the greatestsuccstate under a given action. The specification uses function *tol* which takes two floats (fl_1, fl_2) and returns a tolerance range $fl_1 \pm fl_2$. Also used is the function *trackfactor* that returns a number for a given action and state. Trackfactor is set as follows. For forward action, trackfactor returns 3.0 if the current state is in track A and 2.4 for track B. For reverse action, trackfactor returns 1.33 for both tracks A and B.

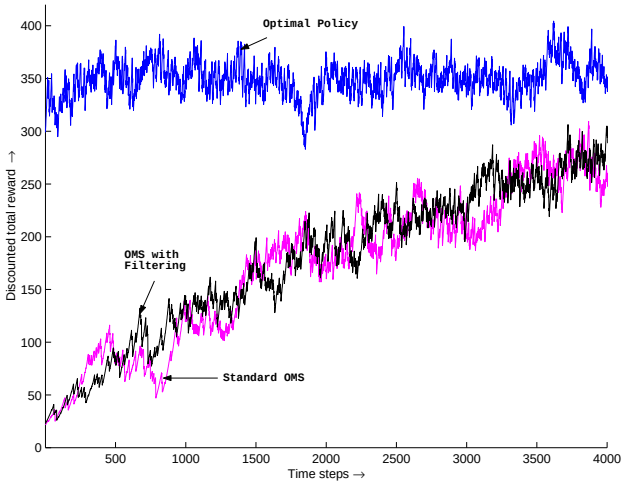


Figure 4. Plot of discounted total rewards over time for the OMS algorithms in comparison with Optimal Policy on a circular flags world instance averaged over 10 trials.

Performance of the learning agent can be measured in several ways. To account for exploration and ex-

ploitation tradeoff we measured the discounted total reward to-go at each point at each time step. More precisely, suppose the agent receives the following rewards $r_1, r_2, \ldots, r_t$ in a run of time length t. The reward-to-go at time t' is defined to be $\sum_{t' \geq t} r'_t \gamma^{t'-t}$. This estimate is reliable only for points that are far enough from the end of the run. In Figure 4, we plot the discounted total reward-to-go for each of the OMS algorithms in comparison to the optimal policy, as a function of time averaged over 10 runs. As expected, the optimum policy gave the largest discounted total reward, averaging 350 from start. The most optimistic first algorithm based on optimistic model selection with filtering resulted in an improved performance when compared to standard OMS without filtering. The slight gain is attributed to the additional information made available to the most optimistic first algorithm. Whilst these are preliminary results, it thus offer prospects for accomplishing task transfers as the template provides a constraining influence on exploration early in learning for related tasks covered by the template.

5. Conclusions and Future Work

We have extended the standard optimistic model selection algorithm to incorporate a process filter. The process filter discourages the selection of models that do not conform to a specified process description language. We studied a version of the OMS algorithms called most optimistic first. Preliminary results indicate that by filtering the process models the most optimistic first algorithm is able to constrain exploration with prospects of improving learning performance. Areas of further work include establishing the efficiency of the most optimistic first algorithm, improving its performance, and studying other filtering methods. In addition, we are studying additional leverage obtained using factored representations. Finally, other important areas of future work are the sensitivity of learning to variations in process templates and the linkage between the process templates and the prior information matrix.

References

Bellman, R. E. (1961). *Adaptive control processes: A guided tour*. Princeton University Press.

Bultan, T. (2000). Action language: A specification language for model checking reactive systems. *Proceedings of the 22nd International Conference on Software Engineering (ICSE 2000)* (pp. 335–344).

Dearden, R. (2000). *Learning and planning in structured world*. Department of Computer Science, University of British Columbia, Canada: Ph.D.Thesis.

Gelfond, M., & Lifschitz, V. (1993). Representing action and change by logic programs. *Journal of Logic Programming*, *17*, 301–321.

Kaelbling, L. P. (1990). *Learning in embedded systems*. Dept. of Computer Science, Stanford: Ph.D.Thesis.

Kearns, M., & Singh, S. (1991). Near-optimal reinforcement learning in polynomial time. *Proceedings of the Fifteenth International Conference on Machine Learning*, *12*, 993–1001.

Martin, J. (1967). *Bayesian decision problems and Markov chains*. New York: Wiley.

Meuleau, N., & Bourgine, P. (1999). Exploration of muti-state environments: Local measures and back-propagation of uncertainty. *Machine Learning*, *35*, 117–154.

Salomaa, A. (1973). *Formal languages*. Academic Press.

Sunmola, F., & Wyatt, J. (2003). Optimistic model selection in structure based reinforcement learning. *Proceedings of the Sixth European Workshop on Reinforcement Learning* (pp. 31–32).

Wiering, W., & Schmidhuber, J. (1998). Efficient model-based exploration. *From Animals to Animats 5: Proceedings of the Fifth International Conference on Simulation of Adaptive Behaviour* (pp. 223–228).

Wyatt, J. L. (1997). *Exploration and inference in learning from reinforcement*. University of Edingurgh, Dept. of Artificial Intelligence, Edingurgh University: Ph.D.Thesis.

Wyatt, J. L. (2001). Exploration control in reinforcement learning using optimistic model selection. *International Conference on Machine Learning (ICML 2001)*, 593–600.

Experiments with Relational Neural Networks

Werner Uwents WERNER.UWENTS@CS.KULEUVEN.AC.BE
Hendrik Blockeel HENDRIK.BLOCKEEL@CS.KULEUVEN.AC.BE
Department of Computer Science, Katholieke Universiteit Leuven, Celestijnenlaan 200A, 3001 Leuven, Belgium

Abstract

The fundamental difference between propositional and relational learners is the ability to handle sets. Most current relational learners handle sets either by aggregating over them or by testing the occurrence of elements with specific properties, but a non-trivial combination of both remains a challenge. In this paper, we present a neural network approach to solve relational learning tasks. These relational neural networks are in principle able to make such a combination. We will discuss some experiments that we conducted to test the capacity of our approach.

1. Introduction

Neural networks have been applied to solve many different learning tasks, but their use is still limited to relatively simple data types. Feedforward neural networks, for example, only deal with propositional data, where each tuple consists of a fixed-size vector of real values. Recurrent networks are able to process sequences. However, few attempts have been made to extend the data domain of neural networks beyond this point. Allowing different types of relations in the dataset and relationships between tuples would be a powerful extension. We will present an approach, based on standard neural networks, to learn concepts over such relational data.

The most fundamental difference between propositional and relational learning is the ability to handle sets. These sets are the result of following one-to-many and many-to-many relationships in the dataset. Some approaches to deal with these sets already exist, but they are often biased as will be explained in the next section. There are also some approaches, based on neural networks, that deal with problems very similar to the relational learning task.

The existing work that is probably closest to our approach, is a line of work in the neural networks community on learning from structured data using recursive neural networks or folding architecture networks (Goller & Küchler, 1996; Sperduti & Starita, 1997; Frasconi et al., 1998). These authors describe how to learn from structured data (e.g. logical terms, trees, graphs) and discuss tasks like the identification of substructures. Those tasks relate to the tasks we consider, more or less as inductive logic programming (ILP) relates to our approach and some existing results on learnability of recursive neural networks may carry over to our setting. However, they do not specifically consider the problem of learning aggregate functions over sets and the problem of different types of data. They also focus on learning in graph structures instead of learning in relational databases.

There has also been some research in using neural networks for multi-instance problems (Ramon & De Raedt, 2000). These problems can be seen as a special case of relational learning because they deal with learning over a single set. If one instance in the set is positive, the sample as a whole will be classified as positive. The basic idea behind multi-instance networks is to use a feedforward network to feed all the instances into and to combine all the results with an aggregation function, namely the maximum function.

Neural logic programs (Ramon et al., 2002) are also somewhat similar to our relational neural networks, with as main differences that they are described in a first order logic framework and that, just like for multi-instance neural networks, specific aggregate functions are encoded in advance by the user, instead of learned. Typically, they represent logical conjunctions and disjunctions.

Our approach is also based on neural networks, but it is oriented specifically towards relational data domains and it does not rely on predefined aggregate functions or concepts. We believe that from the point of view of relational learning, the ability to learn aggregate functions is a crucial advantage of this approach.

In the next section, we will discuss the difference be-

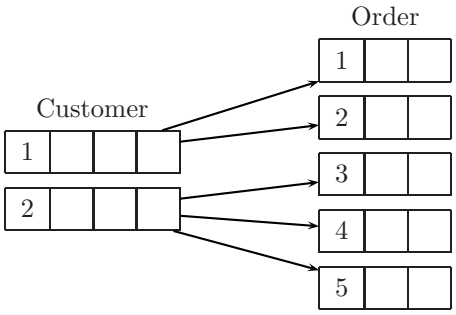


Figure 1. Example of a relational dataset.

tween selection and aggregation over sets. Section 3 gives a definition of the structure and training of relational neural networks. The results of some experiments with these networks will be presented in section 4. We end with some conclusions about the presented approach in section 5.

2. Combining Selection and Aggregation

The learning task that we are considering, is a relational learning task. This means that we have a dataset with a number of different relations. For each relation, a set of tuples is given. Each tuple has a number of attribute values and can also have relationships with other tuples. We want to classify all tuples belonging to some target relation R_T, based on their own attribute values and the attribute values of related tuples.

The specific problem that arises in relational datasets, is how to deal with sets. These sets are the result of one-to-many or many-to-many relationships. An example dataset containing the relations customer and order is given in figure 2. Customer tuples can be linked to a number of order tuples. These linked tuples form a set. The first customer is linked to two orders, the second to three, so the number of tuples in these sets can vary. This is the reason why we can not reduce this dataset to a propositional dataset.

Relational learners can be divided into two categories, depending on how they handle one-to-many and many-to-many relationships, or, equivalently, how they handle sets of tuples (Blockeel & Bruynooghe, 2003). Most current relational learners are restricted to one of these categories. This imposes a significant, possibly undesirable bias on these learners.

Methods in the first category, selective methods, handle sets by looking at properties of their elements. A set S is examined by testing a condition of the form $\exists\, x \in S : P(x)$. This $P(x)$ can be a complicated criterium, but it only considers the attributes of a single tuple. Using this method, we can learn a concept like 'people with at least one son'.

Aggregating methods, the second category, compute a function $F(S)$ over a set S of tuples. This reduces the set to a single value. One example of such a function is the cardinality function that simply counts the number of elements in the set. With such a function, we could express a concept like 'people with two children'.

Many approaches to relational learning rely on some kind of propositionalization of the relational data. On the resulting propositional dataset, a propositional learner can be used. An example of this is the RELAGGS system (Krogel & Wrobel, 2001). The propositional data is extended with some extra attributes, which are the result of evaluating predefined aggregate functions over the related data. This method, however, cannot learn undefined aggregate functions. How combinations of aggregation and selection could be learned, is not explained by the authors.

This is a problem when we want to express a concept like 'people with two sons'. This concept clearly combines aggregation and selection: we have to select all males from the set of children and then count them to check this criterium. Other concepts may require different kinds of combinations of selection and aggregation. As aggregation and selection are both very natural operations, a relational learning system should be able to combine both in the models it builds. However, these combinations can be quite complicated and diverse, and they may depend on the structure of the dataset and the relations in it.

For instance, probabilistic relational models (PRMs), as defined by (Getoor et al., 2001), cannot learn the concept of 'people having two sons' without having separate relations for sons and daughters. Manually introducing these separate relations of course presupposes that the user is aware of the possible importance of these concepts. Alternatively, one could predefine a large number of aggregate functions that have appropriate selection conditions built in. In that case, a search through a space of aggregate functions is needed. The power of this approach largely depends on which aggregate functions are defined.

In inductive logic programming (ILP), one could for instance define aggregate functions as background knowledge. Then, e.g., the rule `p(X) :- count(Y, (child(X,Y), male(Y)), 2)` expresses the concept of people having two sons. The main difficulty here is that the second argument of the count metapred-

icate is itself a query that is the result of a search through some hypothesis space. It is not obvious how such a search should be conducted. The many results in ILP on how to search a first-order hypothesis space efficiently (Nienhuys-Cheng & De Wolf, 1997), do not consider the case where the resulting hypothesis will be used as the argument of a metapredicate.

ILP-like approaches that do not include aggregate functions, can still express the concept as, e.g., 'the person has a male child x and a male child y and $x \neq y$ and there does not exist a child z such that z is male and $z \neq x$ and $z \neq y$'. But in practice, the length of this rule, as well as the occurrence of a negation, make it difficult to learn. The comprehensibility of the result is also negatively influenced.

Knobbe et al. (2002) are, to our knowledge, the first to present a method that performs a systematic search in a hypothesis space (in this case, that of 'selection graphs') where hypotheses combine aggregation and selection. Their approach is however limited to monotone aggregate functions, which limits its applicability somewhat (for instance, sum and average are not monotone), and to selecting aggregate functions from a limited set given by the user.

Our relational neural networks would have as advantage over the other approaches that they can learn an aggregate function, without that function being predefined and with selection possibly integrated in it. Training the relational neural network automatically constitutes a search through aggregations and selections simultaneously.

3. Relational Neural Networks

Assume that we have a dataset with a target relation R_T and some other relations $R_1, \ldots, R_N$. We denote the attribute sets of R_i by U_i. For any relation R, we define

- $S_1(R)$: $R_i \in S_1(R)$ iff each tuple $t \in R$ is connected to exactly one tuple in R_i. This means R has a one-to-one or many-to-one relationship with R_i, in which R participates completely.
- $S_{01}(R)$: $R_i \in S_{01}(R)$ iff each tuple $t \in R$ is connected to at most one tuple in R_i. This is, again, a one-to-one or many-to-one relationship between R and R_i, but now with partial participation.
- $S_N(R)$: $R_i \in S_N(R)$ iff each tuple $t \in R$ is connected to zero, one or more tuples in R_i. This is a one-to-many or many-to-many relationship between R and R_i, with complete or partial participation.
- $S_U(R)$: $R_i \in S_U(R)$ iff R_i is a relation of the relational dataset, but not in $S_1(R)$, $S_{01}(R)$ or $S_N(R)$. This means R is not directly connected to R_i.

Given a tuple $t \in R_T$, we want to classify it based on the information contained in the tuple and in any tuples linked to this tuple. For a relation R_i, we use U_i to denote the original attribute set of that relation. All attributes in U_i must be real values, as this is the only type of input a neural network can process. Other types of attributes need to be converted to real values first. We use I_i to denote the attribute set actually used as input to our neural network. One might expect that $I_i = U_i$, but there will be some small differences:

- For R_T, the target relation, $I_T = U_T - \{C\}$, where C is the class attribute.
- For any $R_i \in S_{01}(R)$, there can be a tuple $t \in R$ for which there exists no tuple $s \in R_i$ that t is directly connected to. As neural networks do not have a distinguished encoding for null values, we will use an extra attribute E_i that indicates whether the link to R_i yielded a tuple or not. $I_i = U_i \cup \{E_i\}$.
- The same problem arises for $R_i \in S_N(R)$, so here also $I_i = U_i \cup \{E_i\}$.

Based on the above, we can construct a relational neural network that classifies $t \in R_T$ based on its own attribute values as well as those of related tuples. For each tuple $t \in R_T$, we construct a tuple t' with attributes

$$I_T \cup \Big(\bigcup_{i:R_i \in S_1(R_T) \cup S_{01}(R_T)} O_{1i} \Big) \cup \Big(\bigcup_{i:R_i \in S_N(R_T)} O_{Ni} \Big)$$

with I_i as defined above, O_{1i} a set of attributes that are the output values of a feedforward neural network taking I_i as input values and O_{Ni} a set of attributes that are the output values of a recurrent neural network taking I_i as input values. This tuple t' has then a fixed set of attributes which can be used to feed into a feedforward neural network. The output of this network gives us the final result of our classifier.

In the described approach, sets resulting from relations in $S_N(R_T)$ are processed using recurrent neural networks. These networks are able to process tuple sequences of indefinite length. However, we are presenting the tuples to the network in some imposed order

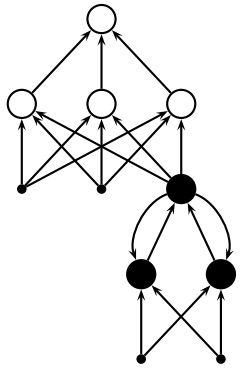


Figure 2. Example of the structure of a relational neural network.

while the sets are actually unordered. As we will see, this fact can be exploited in training the network.

The precise structure of the different neural networks in our classifier must be defined now. We take both the feedforward and recurrent networks to have two layers. The ideal number of neurons in each layer needs to be tuned by conducting experiments, there is no straightforward rule to determine this.

For the recurrent networks, we also have to define which recurrent connections are allowed. The most expressive recurrent network is a fully connected network in which each neuron has connections with all other neurons. But as this makes the number of connections increase quadratically when the number of neurons increases, we prefer the Jordan recurrent network (Jordan, 1986).

In this kind of recurrent network, each neuron in the second layer is connected with all neurons in the first layer. The number of recurrent connections is then $n_1 \times n_2$, with n_1 and n_2 the number of neurons in the first and second layer respectively. This gives us a good trade-off between expressiveness and the number of neurons and connections in the network.

A small example of a relational network is given in figure 3. The two attributes of the target relation are fed into the feedforward part of the network (white neurons). The two attributes of the tuples of another relation, linked to the target relation, are fed into the recurrent part of the network (black neurons). The output of this recurrent part is used as extra input to the feedforward part.

The technique of adding to t the O_{1i} and O_{Ni} attributes that summarize related tuples, can be repeated for those tuples, thus also incorporating information in indirectly linked tuples. In the end, this yields a hierarchical structure where each node is a neural network that takes the attributes of a relation and the outputs of its children as input and propagates the result to its parent node.

Training this relational neural network can be done with an adapted form of the standard backpropagation algorithm. The feedforward neural networks in the relational network are trained with standard backpropagation. The recurrent networks are trained with backpropagation through time (BPTT) (Werbos, 1990). The key idea to BPTT is the unfolding of the recurrent network into a feedforward network.

As many folds (copies of the original network) are created as there are instances in the input sequence and recurrent connections are converted into feedforward connections between successive folds. The resulting feedforward network is trained using the standard backpropagation algorithm, but with one important restriction: since all folds have been created by replicating the original network, weights in all folds should be the same.

The fact that sets are fed into the recurrent network in some imposed order, can be used to improve our training algorithm. This can be done by reshuffling the sequence and presenting the set to the recurrent network in a different order. Two possibilities can be considered: reshuffling after every training iteration and expanding the training set by adding reshuffled copies of the initial instances.

4. Experiments

To evaluate this approach, we have performed experiments on the musk and trains datasets. The musk dataset, available from UCI (Merz & Murphy, 1996), is an example of a multi-instance learning task. As mentioned above, this can be seen as a special case of relational learning. In this dataset, each example describes a molecule. For each example, several poses (instances) are given, each with 166 attributes. If at least one of these poses has some property, the molecule is said to be musk.

There are two versions of this dataset, musk 1, containing 92 molecules, and musk 2, containing 102 molecules, which differ in size. Musk 2 has more conformations per molecule than musk 1. Several learning approaches have been compared on this dataset (Dietterich et al., 1997). To be able to compare our results with these results, we conduct our experiments in the same setting, namely ten-fold cross-validation.

Overall results are summarized in table 1 (results for multi-instance neural networks come from Ramon and

Table 1. Classification accuracies on the musk dataset.

method	musk 1	musk 2
iterated-discrim APR	92.4%	89.2%
GFS elim-kde APR	91.3%	80.4%
GFS elim-count APR	90.2%	75.5%
GFS all-positive APR	83.7%	66.7%
all-positive APR	80.4%	72.6%
simple backpropagation	75.0%	67.7%
multi-instance neural networks	88.0%	82.0%
C4.5	68.5%	58.8%
1-nearest neighbor (euclidean distance)	/	75%
neural network (standard poses)	/	75%
1-nearest neighbor (tangent distance)	/	79%
neural network (dynamic reposing)	/	91%
relational neural networks	89.1%	85.3%

Table 2. Training configurations for the musk dataset (n_1 = number of neurons in first layer of the recurrent component, n_2 = number of neurons in second layer of the recurrent component, η = learning rate, μ = momentum term).

	dataset	n_1	n_2	η	μ	reshuffle	iterations	accuracy
1	musk 1	50	10	0.5	0.2	every it.	190	84.8%
2	musk 1	50	10	0.5	0.2	none	110	88.0%
3	musk 1	50	10	0.5	0.2	30 copies	10	89.1%
4	musk 2	80	20	0.5	0.2	every it.	40	76.5%
5	musk 2	80	20	0.5	0.2	none	240	85.3%
6	musk 2	40	15	0.5	0.2	none	50	77.5%
7	musk 2	40	15	0.5	0.2	5 copies	20	80.4%

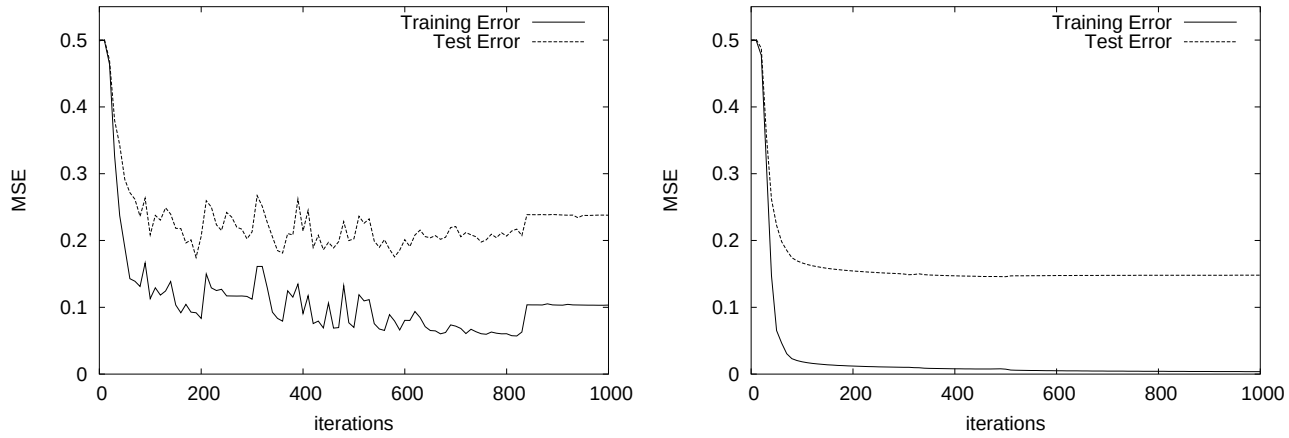
De Raedt (2000), other results from Dietterich et al. (1997)). The tangent distance and dynamic reposing technique require computation of the molecular surface, which cannot be done using the feature vectors included in the dataset. A comparison of different configurations for the relational neural networks, is shown in table 2.

These results show that relational neural networks are performing quite well. They give results that are a lot better than simple backpropagation and even better than multi-instance neural networks. The latter can only be the result of better parameters because the hypothesis space searched by relational neural networks, H_{RNN}, is a superset of the hypothesis space H_{MINN} searched by multi-instance neural networks. This means that the hypothesis in H_{RNN} that best approximates the target hypothesis, must also be in H_{MINN}. The accuracy of iterated-discrim axis-parallell rectangles (iterated-discrim APR), the best method, is still significantly higher, but this method is specifically designed for this kind of problems.

When we look at the differences between different configurations for training the relational neural networks, we see that reshuffling the training data after every iteration gives poor results. However, expanding the training set with reshuffled copies does improve training. Although the training set becomes larger, the accuracy is better and the necessary number of iterations is reduced. It also helps to avoid overfitting.

Figure 3 shows us why reshuffling after every iteration is not working well. Reshuffling after every iteration actually changes the training set continuously. This means that the error function also changes continuously, and what was the gradient in the previous iteration may be unrelated to the gradient in this iteration. This makes it difficult for the backpropagation algorithm to converge. The mean square error on training and test set is very jagged in this case (see figure 4(a)). This is not the case for adding reshuffled copies to the training set (see figure 4(b)). Adding reshuffled copies has a similar effect as enlarging the dataset with new samples.

Figure 3. Mean square error (MSE) for training and test set in function of the number of training iterations.



(a) With reshuffling after every iteration.

(b) With adding reshuffled copies.

Table 3. Accuracies on the trains dataset, comparing relational neural networks with first order decision trees using random forrests (RNN 1 is without reshuffling, RNN 2 is with reshuffled copies).

dataset	concept	samples	noise	RNN 1	RNN 2	FORF
trains 1	simple	100	none	80%	95%	100.0%
trains 2	simple	100	5%	78%	89%	92.8%
trains 3	simple	1000	none	100%	100%	100.0%
trains 4	complex	800	none	89.4%	97.8%	96.1%
trains 5	complex	800	5%	84.5%	89.8%	90.3%

The second tested dataset is the train dataset based on the Michalski train problem. This problem was invented by Ryszard Michalski around 25 years ago (Michalski, 1980). The aim is to find a concept which explains why trains are travelling eastbound or westbound. Every train consists of a number of cars, carrying some load. The concept is based on the properties of these cars and their loads.

We used a generator for this train problem to create a dataset (Michie et al., 1994). Two different concepts are used for the generation, a simple and a more complicated. The simple concept defines trains that are eastbound as trains with at least two circle loads, the other trains are westbound. The more complicated concept defines westbound trains as trains that have more than seven wheels in total but not more than one open car with a rectangle load, or trains that have more than one circle load; the other trains are eastbound.

First-order decision trees using random forests (Vens et al., 2004) are another approach to tackle the problems combining selection and aggregation. We compare our results with results obtained with the latter method. A complete overview of the results is given in tables 3 and 4. The used datasets are generated to test different settings: a simple versus a more complicated concept and noise versus no noise. The tests were done with five-fold cross-validation.

What we see for the simple concept (datasets 1, 2 and 3), is that 100 samples is not enough to learn the concept completely. Taking a dataset with 1000 samples, however, we can reach 100% accuracy. Adding 5% noise results in an accuracy that is a little more than 5% lower, so our classifier seems to be rather noise resistant.

For the more complicated concept (datasets 4 and 5), the results are quite similar to those for first-order decision trees using random forests. The effect of noise is the same as for the simple concept. For both the simple and complicated concept, there is a remarkable difference between training without reshuffling and with added reshuffled copies.

Table 4. Training configurations for the trains dataset (n_1 = number of neurons in first layer of the recurrent component, n_2 = number of neurons in second layer of the recurrent component, η = learning rate, μ = momentum term).

dataset	n_1	n_2	η	μ	reshuffle	iterations	accuracy
trains 1	20	10	0.1	0.0	none	250	80%
trains 1	20	10	0.1	0.0	10 copies	100	95%
trains 2	20	10	0.1	0.0	none	300	78%
trains 2	20	10	0.1	0.0	20 copies	150	89%
trains 3	20	10	0.1	0.0	none	40	100%
trains 3	20	10	0.1	0.0	10 copies	10	100%
trains 4	60	40	0.2	0.1	none	300	89.4%
trains 4	60	40	0.2	0.1	20 copies	200	97.8%
trains 5	60	40	0.2	0.1	none	200	84.5%
trains 5	60	40	0.2	0.1	20 copies	130	89.8%

5. Conclusions

We have presented a novel, neural network based approach to relational learning. The approach consists of constructing a neural network that reflects the structure of the relational dataset. This relational neural network may contain both feedforward and recurrent parts. The training algorithm for this relational network is based on the standard backpropagation (through time) algorithm.

Our approach was tested on two datasets. It turned out that relational neural networks are performing quite well compared with other methods. They can deal with noisy data and the expressive power of Jordan recurrent networks seems to be sufficient to learn the desired concepts.

While many open questions remain regarding the optimal architecture of these relational neural networks, their learning behaviour, the optimal learning methodology, etcetera, we did obtain a first important result from these experiments: the 'copy reshuffling' method to temove the effect of set ordering is clearly superior to the other reshuffling method. Its effects include a higher final accuracy, less iterations needed to train the network and reducing the risk of overfitting. These advantages outweigh the disadvantage of having a larger training set.

Acknowledgements

Hendrik Blockeel is a postdoctoral fellow of the Fund for Scientific Research of Flanders (FWO-Vlaanderen). Werner Uwents is supported by IDO/03/006 'Development of meaningful predictive models for critical disease'. We thank Celine Vens for generating the train dataset and obtaining results with first-order random forests.

References

Blockeel, H., & Bruynooghe, M. (2003). Aggregation versus selection bias, and relational neural networks. *IJCAI-2003 Workshop on Learning Statistical Models from Relational Data, SRL-2003, Acapulco, Mexico, August 11, 2003*.

Dietterich, T. G., Lathrop, R. H., & Lozano-Pérez, T. (1997). Solving the multiple-instance problem with axis-parallel rectangles. *Artificial Intelligence*, *89*, 31–71.

Frasconi, P., Gori, M., & Sperduti, A. (1998). A general framework for adaptive processing of data structures. *IEEE-NN*, *9*, 768–786.

Getoor, L., Friedman, N., Koller, D., & Pfeffer, A. (2001). Learning Probabilistic Relational Models. In S. Dzeroski and N. Lavrac (Eds.), *Relational data mining*, 307–334. Springer-Verlag.

Goller, C., & Küchler, A. (1996). Learning task-dependent distributed representations by backpropagation through structure. *Proceedings of the IEEE International Conference on Neural Networks (ICNN-96)* (pp. 347–352).

Jordan, M. I. (1986). Attractor dynamics and parallelism in a connectionist sequential machine. *Proceedings of the Eighth Annual Conference on Cognitive Science* (pp. 531–546).

Knobbe, A., Siebes, A., & Marseille, B. (2002). Involving aggregate functions in multi-relational search. *Principles of Data Mining and Knowledge Discovery, Proceedings of the 6th European Conference* (pp. 287–298). Springer-Verlag.

Krogel, M.-A., & Wrobel, S. (2001). Transformation-based learning using multi-relational aggregation. *Proceedings of the Eleventh International Conference on Inductive Logic Programming* (pp. 142–155).

Merz, C., & Murphy, P. (1996). UCI repository of machine learning databases [`http://www.ics.uci.edu/~mlearn/mlrepository.html`]. Irvine, CA: University of California, Department of Information and Computer Science.

Michalski, R. (1980). Pattern Recognition as Rule-Guided Inductive Inference. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *2*, 349–361.

Michie, D., Muggleton, S., Page, D., & Srinivasan, A. (1994). *To the international computing community: A new east-west challenge* (Technical Report). Oxford University Computing Laboratory, Oxford, UK. Available at ftp.comlab.ox.ac.uk.

Nienhuys-Cheng, S.-H., & De Wolf, R. (1997). *Foundations of Inductive Logic Programming*, vol. 1228 of *Lecture Notes in Computer Science and Lecture Notes in Artificial Intelligence*. New York, NY, USA: Springer-Verlag.

Ramon, J., & De Raedt, L. (2000). Multi instance neural networks. *Proceedings of the ICML-Workshop on Attribute-Value and Relational Learning*.

Ramon, J., Driessens, K., & Demoen, B. (2002). Neural logic programs. `http://www.cs.kuleuven.ac.be/~janr/nlptechrep.ps`, to be published as a technical report.

Sperduti, A., & Starita, A. (1997). Supervised neural networks for the classification of structures. *IEEE Transactions on Neural Networks*, *8*, 714–735.

Vens, C., Van Assche, A., Blockeel, H., & Džeroski, S. (2004). First order random forests with complex aggregates. *Proceedings of the 14th International Conference on Inductive Logic Programming* (pp. 323–340). Springer.

Werbos, P. J. (1990). Back propagation through time: What it does and how to do it. *Proceedings of the IEEE* (pp. 1550–1560).

Evolving Neural Networks for Forest Fire Control

Marco Wiering MARCO@CS.UU.NL
Intelligent Systems Group, Utrecht University, Padualaan 14, 3508TB, Utrecht

Filippo Mignogna FMIGNOGN@CS.UU.NL
Presidio Siemens, I.C.P., via Daverio 6. 20100 Milano

Bernard Maassen BMAASSEN@CS.UU.NL
Computer Science Department, Utrecht University, Padualaan 14, 3508TB, Utrecht

Abstract

Forest fire control is a challenging research problem involving a non-stationary environment and multiple cooperating agents. In this paper we describe the application of enforced sub-populations (ESP) to evolve neural network controllers that solve different instances of the forest fire control problem. Our system works by initially generating subgoals and assigning subgoals to the different agents. The subgoal generator and task assignment module are modelled as multi-layer perceptrons which are evolved to minimize the damage done by the spreading fire. The experiments show that agents learn to stop forest fires and that incremental learning can be used to solve more complex problems.

1. Introduction

Forests play a crucial role for sustaining the human environment and because forest fires are among the largest dangers for forest preservation, it is not a surprise to see increasing state expenditures for forest fire control. Despite of this, annually millions of hectares of forests are still destroyed by fires. If we look at the way current forest fires are attacked, then usually the fire boss makes an initial attack plan to stop the spread of the fire. This plan consists of a number of fire-lines that break the fire-propagation. Then he allocates resources from neighboring resource bases to fulfil all subplans. After this the field-commander is in control and reevaluates the plans constantly based on a stream of online information (Wiering & Dorigo, 1998). In case of very large fires, a first attack plan usually does not work. Therefore in this case, no-one knows how to deal with the problem and the forest fire control team usually waits until the weather changes which can last as long as three weeks. This of course results in large forest areas being destroyed. Therefore we want to study the application of intelligent algorithms for dealing with large forest fires.

Forest fires as expanding processes resemble disease epidemics and volcanic eruptions, and for all these catastrophes propagation lines should be removed. E.g., to control disease epidemics roads leading away from diseased areas should be severely controlled so that the disease cannot spread itself further. In forest fire control there are three ways to deal with the problem; removing fuel, removing oxygen, and decreasing temperature. Removing fuel can be done by bulldozers or other ground agents that cut away trees or grass and this is the most effective way for dealing with large forest fires. For removing oxygen or decreasing temperature, airborne agents can be used that throw waterbombs containing chemicals, but without cutting fire-lines this method is not sufficiently effective to stop a large forest fire (although airborne agents are helpful to decrease the propagation speed of the fire). Therefore, there are basically three types of attacks: airborne attack, a ground attack, and a mixed attack employing both air- and ground-forces. In this paper we concentrate on ground-attacks, although our simulator also allows for mixed attacks.

In previous work, forest fire control is done using planning algorithms. One of the first attempts was Phoenix (Cohen et al., 1989), a simulated environment modelling forest fires in Yellowstone National Park. Agents, which included watchtowers, fuel trucks, helicopters, bulldozers, and a coordinating fire boss, were used to fight the fire using planning techniques. The CHARADE project (Ricci et al., 1994; Avesani et al., 1997; Avesani et al., 2000) is a working environmental decision support system for managing first inter-

vention in forest fires. The planning system integrates case-based reasoning and constraint satisfaction (Avesani et al. 1997) and is integrated with a geographic information system (GIS). The case-based reasoning system uses a database of previous plans to deal with forest fires in the south of France to select appropriate plans for the current fire. The problem of this approach is that there are no plans available for very large forest fires, which have never been controlled successfully. Instead, our approach relies on simulation and learning, where a large number of plans is simulated and after each simulation the plan-generating policy is adapted based on results of previous plans.

For learning to control forest fires, we use enforced sub-populations (Gomez & Miikkulainen, 1998) which is a promising evolutionary algorithm for evolving neural network controllers. Basically, enforced sub-populations (ESP) is an approach to solving reinforcement learning (RL) problems using direct policy search instead of learning a value function. The disadvantages and advantages of direct policy search versus value function based RL are still being debated, but we think that direct policy search based on evolutionary algorithms makes it faster to find initially good controllers, although the fine tuning of the controllers is more difficult (due to the small amount of policies that work better instead of worse compared to the best previous policy). Nevertheless, they might also be combined in some fruitful way, and we intended to research first direct policy search since the environment is composed of multiple agents and due to the highly non-stationary forest fire dynamics, learning accurate value functions is very difficult.

In the next section we will describe the forest fire simulator called "Bushfire". In section 3, we explain how we use the ESP algorithm for evolving neural network controllers implementing forest fire control policies. In sections 4 and 5 we present experimental results. Finally in section 6 we discuss the obtained results and describe possibilities for future research.

2. Forest Fire Simulator: Bushfire

We study forest fire control by a learning multi-agent system. For this we developed a forest fire simulator, named Bushfire, based on a stochastic cellular automaton where single cells may contain different kinds of trees, grass, water, digged paths, and cells may be on fire or not. The fire starts at some place and then propagates itself according to wind strength and direction, and humidity.

The basic variable entity of a cell is its fire activity. If a cell is ignited the cell starts to release fire activity to its neighboring cells according to its type, wind direction and speed, and humidity. After a cell has received more fire activity than a specific threshold for this cell-type (e.g. the threshold for trees is larger than for grass) the cell starts to burn as well and releases fire activity to its neighbours. After some time the fuel of a cell becomes depleted and the cell enters a burned state that is not able to release fire activity any longer. By setting the different parameters, we can construct forest fires which are moving slowly or very fast. This enables us to deal with a large number of different difficulty levels for controlling them. One of the most important parameters here is the trade-off between the number of steps a bulldozer can make compared to the number of steps the fire is able to propagate itself. Setting this parameter to a high value means that it is easy to perform many steps to control the fire, thereby making the success ratio much larger and the size of the burned area much smaller.

The goal of the multi-agent system is to control the propagation of the forest fire. This they can do by cutting fire-lines around the fire. Therefore the question becomes: where should the agents cut fire-lines to minimize the damage done by the forest fire?

The problem can in principle be described by a Markov decision process consisting of states, actions for the agents, transition rules to go from one state to the next when the agents have executed their actions, and a reward function. The number of states is huge; first of all there are at least 10,000 discrete cells in our simulations which have their own properties such as consisting of grass, trees, digged paths, etc. Furthermore the properties of a cell also consist of the fire activity and the remaining unburned fuel which are continuous numbers. Thus, it is clear that even if we could measure all these properties, we cannot take the whole state information into account when selecting an action. The actions are much simpler, basically there are 8 of them; drive north, south, east, west, and dig north, south, east, and west. The transition rules are complicated and depend on the cellular automaton and its parameters. Finally, the reward function should handle the case that an agent is burned in the fire, that the fire is contained by the fire-lines, that special cells are put on fire (e.g. houses), and the size of the area being burned.

To solve the problem, we propose an algorithm that generates subgoals after which the bulldozers cut fire-lines between subgoals. The planning between subgoals is currently done using a simple linear path-planner, but can also be done using the A* algo-

rithm that takes into account that cutting fire-lines over grass can be done much faster than cutting away trees. The main problem is to generate optimal subgoals. This is a difficult control problem, since each forest fire looks different and the state space is huge.

The method we devised for controlling forest fires consists of four steps:

- Generate initial subgoals around the fire
- Enhance subgoals using local information
- Assign each bulldozer to a specific path from one subgoal to another subgoal
- Dig lines between subgoals using a path-planner

Some replanning of subgoals is done automatically if a straight line from an agent to a subgoal goes through the fire. We explain these steps in more detail below.

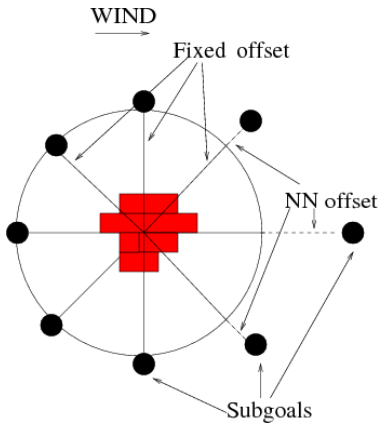


Figure 1. Around the fire we first construct 8 subgoals using a fixed offset. After this, the neural networks receive information from each subgoal and learn to displace it from the fire. The 8 subgoals will then be used for the bulldozers, although replanning may change them afterwards.

2.1. Generating Initial Subgoals

For controlling forest fires, we developed a representation using 8 subgoals in all wind-directions around the fire. If the agents are able to cut fire-lines around the fire by going through and connecting all subgoals, the fire has been contained. For generating the subgoals we use 8 lines in all 8 directions to place each subgoal. First we use a fixed offset from the centre of the fire (this can be seen as a non-learning approach) and then we learn with a neural network how much the offsets should be displaced in all 8 directions. This neural network can receive as inputs the distance and angle to the fire front, the wind and speed direction, and some measures of the average humidity, fuel, and the threshold of the fire front neighbouring cells. Figure 1 shows how initial subgoals are generated.

2.2. Subgoal Enhancer

The previously described subgoal generator only used global information such as wind speed and direction and distance to the fire front as inputs in the neural network. Sometimes it is necessary to look at local characteristics as well for generating subgoals. E.g. if there is a house or river nearby, this may change the optimal plan. The task of the subgoal enhancer module is to learn to map local information to a small change of the position of the subgoal. The enhancer neural network looks at all initially generated subgoals and receives local information given the place of the subgoal to compute an evaluation. Then the subgoal may be changed some squares in the cellular automaton and using the new input generated by the new relative local surroundings, the neural network computes a new evaluation. If some neighboring square has a higher evaluation than the current square, we continue this local hillclimbing strategy with the new square and otherwise we stop.

2.3. Task Assignment

Although we assume that the number of resources is fixed before a simulation (the number of resources can be chosen by the user), we still have to assign (allocate) the subgoals to each individual agent. This is also done using neural networks. The neural networks obtain information such as the distance from a subgoal to the agent, the wind speed and direction, the distance from an agent to the fire front, and which subgoals have already been chosen before, to choose for each agent a single subgoal to go to. If there are multiple agents, all agents examine all subgoals from which a list of evaluations is obtained by propagating subgoal and agent information through the neural networks, after which first the highest evaluation is taken and that agent is allocated to that subgoal. Then, the fact that this agent is allocated to a subgoal is used as input in the neural network for computing the subgoals of the next agents, after which the second agent is allocated to its highest subgoal etc. In this way the task assignment module is responsible for learning to coordinate the multi agent team. There is no need for communication etc., since we alternatively assign each agent to a subgoal and this information is used by the task assignment module to assign the other agents one by one.

2.4. Path Planning

Once subgoals are generated and allocated to each agent, the agent uses a path-planner to dig a fire-line from its current position to the subgoal. In case the simulation just started, the agents first go to a subgoal before digging a fire-line thereby driving with higher speed to reach a subgoal. We implemented a simple path-planner that constructs more or less a direct line to the subgoal. If it is detected that this straight line goes through the fire, the subgoal's location is changed by the subgoal generation module now taking into account different input information. The other path-planner is A* and can take into account that going close to the fire is dangerous, and that digging through grass goes faster than digging through trees, but since A* is much more time consuming, we only used the simple path-planner in our current experiments.

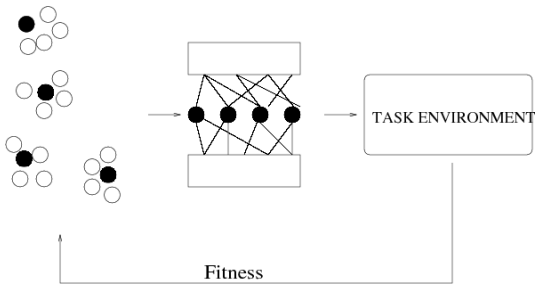


Figure 2. Enforced Sub-Populations (ESP) constructs a neural network by taking one neuron from each sub-population. The resulting neural network is tested and the evaluation is used to evolve novel sub-populations of neurons.

3. Enforced Sub-Populations for Evolving Neural Networks

For learning to generate subgoals and assign agents to them, we use Enforced Sub-Populations (ESP). ESP (Gomez & Miikkulainen, 1998) is an evolutionary method for evolving neural networks. It works by keeping different subpopulations containing neurons that have weighted connections to inputs and outputs (see Figure 2). To generate a neural network, one neuron is selected from each subpopulation and these neurons then form a feedforward neural network. In order to train the system, each neuron from each subpopulation is combined a number of times (we use 10 times on average) with neurons from different subpopulations, and the neuron is assigned the average (or maximal) fitness of the networks in which it took part. Then crossover and mutation are used within the subpopulations to generate new neurons. In this way, neurons that can collaborate well with other neurons will receive higher fitness values, and will be used to evolve novel neurons. ESP has already been used for particular difficult reinforcement learning problems such as double pole balancing with hidden state (Gomez & Miikkulainen, 1998) and active guidance of a rocket (Gomez & Miikkulainen, 2003) and obtained good results.

The reason we used ESP for evolving neural networks is that they do not suffer from the permutation problem when using crossover on complete neural networks. This permutation problem is caused by different orderings of neurons in a network, so that offspring can easily have the same functional units multiple times and loose an important different unit. Since we want to use crossover for finding solutions, we use a symbiotic algorithm with specialized neuron sub-populations. A difference with SANE (Moriarty & Miikkulainen, 1996) which keeps a single population and no sub-populations is that SANE requires a meta-stable population in which neurons with different functions stay in the population at the same time, whereas in ESP each sub-population may converge to a unique neuron. Furthermore, recombination of neurons implementing different functions is usually not very effective, and therefore ESP only recombines neurons in the same sub-population. In some aspects, ESP resembles the cooperative co-evolution method from Potter and de Jong (2003) and SEAM from Watson and Pollack (2000) which are also evolutionary algorithms based on symbiotic combination instead of sexual reproduction, but ESP is specifically tailored for evolving neural networks.

We use ESP for evolving the different modules; the subgoal generator, the subgoal enhancer, and the task assignment (although in our current experiments we did not use the subgoal enhancer module). These different modules can be evolved at the same time. An advantage of ESP is that it is easy to use for multi-agent learning. In multi-agent learning, issues arise about credit assignment to individual agents given a team reward. In ESP these issues are solved using the same mechanism as with single agent learning; each agent uses its own neurons and each neuron is again evaluated by how well the resulting combinations of neurons (and multiple neural networks) work. The fitness of such a combination of neural networks is computed by testing the system in a forest fire simulation. The fitness function takes into account the burned area, whether the fire-propagation was stopped, how much subgoals were successfully digged, and how often it was necessary to recompute subgoals. By evolving neurons which have higher and higher fitness values,

the resulting neural networks also become better and after a while we can save the best found complete neural network modules.

4. Experiments with Learning the Subgoal Generation Module

In this section we perform two different experiments to study the ability of the system to learn to control forest fires using only the subgoal generation module. The enhancer was not necessary, since the local regions look alike (there are no houses, rivers etc.). Furthermore, we use the fixed nearby assign module for allocating agents to subgoals. Therefore in the experiments we describe in this section we only used a learning subgoal generation neural network.

In the **first experiment**, we study a virtual pine forest environment in which three agents are located in different parts of the world that have to cooperate to stop the fire propagation. In this environment we gradually lower the cells' thresholds to increase the fire propagation speed. In our **second experiment**, we show the *Incremental Learning* skill of ESP. If we train a population to solve a determined task, we can use the same population to solve a similar, but more difficult task. In previous work (Gomez & Miikkulainen, 1998), it was shown that starting the evolution from a trained population in an easier task resulted much faster in good performing individuals for a more complex task than starting evolution from a randomly initialised population.

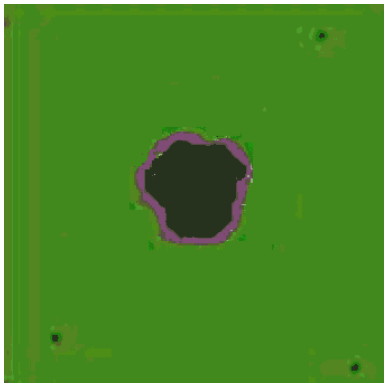


Figure 3. Pine forest environment: in the centre the fire has spread as far as the fire-line has been digged. In the angles north-east, south-west, and south-east of the map we can see the bulldozers' depots.

4.1. Three Bulldozers in a Pine Forest

The first set of experiments are done on a virtual pine forest by an agency composed of three bulldozers. Trees are more difficult to ignite than grass, however once ignited they release much more heat thus propagation is still fast. For this reason pine-tree cells present bigger fuel and threshold than grass. Controllers operate in much more difficult circumstances in a pine trees forest than in a grassy terrain. We model this by decreasing the digging and moving speed in a pine forest environment. We experimented in a world of 120×120 cells. Experiments with a single agent were not satisfactory. One single bulldozer was not able to dig a complete line around the fire in a good time. Therefore we used a system composed of three bulldozers initially located in three different depots. In Figure 3 we show a virtual pine forest in which fire has been contained in the line digged by the agents.

Setup. We tried different kinds of neural networks with different architectures and activation functions. We decided to use a single hidden layer network with 8 hidden units and 1 linear output unit, because we did not find meaningful improvements using more layers and neurons. The sigmoid function has been proved to work well in the hidden layer but not in the output unit. The setting of the ESP parameters applied to our architecture are as follows: Population size is 30. We assign the maximal (and not the average) fitness obtained in one of the (on average ten) tests of a neuron (individual). The crossover rate is 50%. We used linear mutation with 25% probability and Gaussian mutation with 33% probability. We designed a quite complex fitness function including information such as the time of spreading, the overall cost of the cells burnt, the number of generated subgoals and the number of goals reached.

Threshold	Burnspeed	Gen.	Fitness	Efficiency
330	1.11	1	11.438	52.4%
305	2.35	2	11.375	53.5 %
280	2,66	2-3	11.279	55.9 %
255	2,75	3-4	11.044	58.8 %
230	3,16	5-6	10.201	64.8 %
205	3,33	-	3.787	0

Table 1. ESP performances using a team of three agents in a virtual pine forest environment.

Results. In Table 1 we can see the results. The task is solved if the system finds a solution with fitness bigger than 10 (in this case the fire propagation has been stopped effectively). We always stopped the simulation after the twentieth generation. The simulations were repeated 10 times for each problem and we store the average of the maximum fitness received in each

simulation in the table. We also store the generation in which the system finds the solution in the majority of the simulations. Efficiency is the ratio between the time of the controllers to dig a complete line surrounding the fire and the time of the fire to burn the cell inside that line. When the system finds a solution with a low efficiency ($\leq 35\%$) it means that the task was too easy or that the solution found was not so useful, because the agents could have digged a smaller line to reduce the area destroyed by the fire. However, if this value is too big ($\geq 90\%$) the solution computed is not robust, because if there is a slight change of the environmental conditions the system cannot anymore deal with the problem.

For all the tasks except for the last one we found a solution in each simulation; in the last task we tried with a threshold of 205 we never found a solution. The fastest fire propagation the system could successfully deal with is characterised by a threshold of 230 and an average propagation speed of about 3.16 cells/step.

We can also see from Table 1 that the fitness obtained decreases when lowering the threshold. However, the efficiency of the system increases. The explanation lies in the fact that for faster propagation the system has to generate a larger digged line to surround the fire. Thus, more cells burn and the fitness is lower. However, in the same situation the system has proved to be more efficient in terms of time to dig the line versus the time for the cells to burn. Figure 4 shows a typical learning evolution of ESP with a threshold of 230.

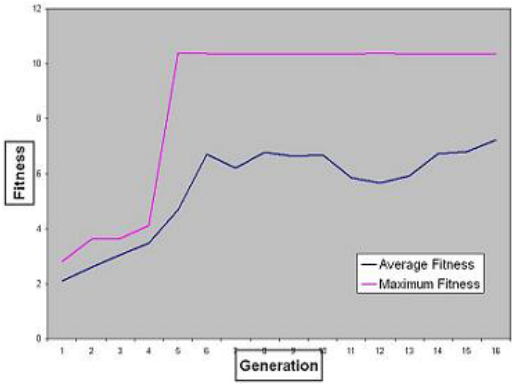


Figure 4. ESP applied to contain flames in a pine forest environment with a threshold of 230.

ESP works well even if at the first generations the maximum and the average fitness are very low. In these cases the fire propagation was not stopped. After the fifth generation we can see that the maximum fitness increases a lot. In that case a solution has been found

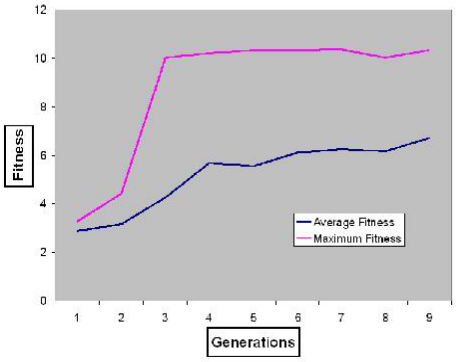


Figure 5. Incremental ESP applied to contain flames in a pine forest environment with a threshold of 230.

with an efficiency of 65%. After that individuals start to converge to the best performing ones. We can see ESP's principal strength: the fast search for a satisfactory solution. ESP is not so good in improving the solution found: when a satisfactory result is obtained ESP rarely improves it. In this experiment we found that ranking the neurons according to the maximal fitness received in the tests worked better than taking the average fitness.

4.2. Incremental Learning

We repeat the previous experiments on the virtual pine forest with the same neural network and ESP settings. However, we do not start from a randomly generated population, but we use a population already trained in an easier task. In our experiments we start to solve the task with a threshold of 280. Once we have found a solution we use the population trained to search a solution for the task with a threshold of 255. Then we repeat the same operation until we reach the task with a threshold of 205. In Figure 5 we can see the fitness evolution in solving the task with a threshold of 230, starting with a population already trained to solve the easier tasks with a threshold of 280 and 255. With respect to the fitness evolution of the same task starting with a random population we can observe that the system finds a solution in the 3rd generation instead of after 5 or 6 generations. Then, we used this trained population to solve the task with a threshold of 205. We repeated this last part of the experiment 10 times and we found a solution in 80% of the trials. The generations in which we found the solutions range from the eighth generation to the twelfth generation. We note that without incremental learning, we never found a solution for this difficult problem. Thus, starting from a population already evolved to solve an

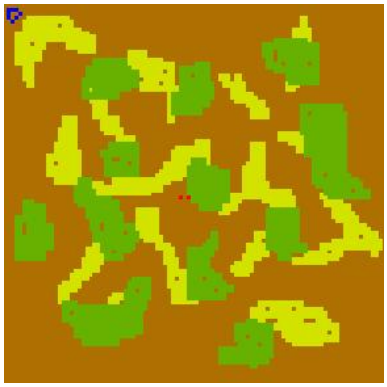


Figure 6. Heterogeneous environment consisting of hay, small brushes, and oak trees. In the north-west we can see the bulldozer's depot.

easier task has been proved effective in this case.

5. Experiments with Learning Subgoals and Task Assignment

In the previous experiments, we only trained the subgoal generator module (SGG), since the nearby assign module works well for homogeneous terrain types. However, for heterogeneous terrain types, learning subgoals and the assignment may prove worthwhile.

The map used for these experiments consists of 3 terrain types, one is fast burning but does not generate much heat (hay), one is slow burning but generates lots of heat (oak trees) and we use a terrain type that is between the two (small brushes). The terrains are setup in such a way that the hay will ignite the small brushes in time, but has big problems igniting the oak trees. The small brushes can ignite the hay with ease and will ignite the oak trees in time. The oak trees can ignite both with ease, and they all can ignite themselves. With these terrain types we have built a map that has corridors of fast burning hay and slow burning plains. In this way the fire does not spread in a circle form but has a far more grim form, making it much more difficult to extinguish. In Figure 6 we show the map.

All experiments are done on the same map under the same conditions. We used only 1 agent to extinguish the fire. The map had a dimension of 100 by 100 cells. Each simulation went on until the fire was extinguished or the program had done 2000 steps. These 2000 steps are more than enough to burn almost the whole map giving a very low evaluation value. The agent started in the upper left corner and the fire started in the middle of the map.

We used the map and let the fire spread with different speeds. Instead of adjusting the threshold values, we used the Fire Propagation (FP) step divider that regulates how many steps FP has to wait before it can do one step. By making this each time 1 less we can find a value for which the system cannot find a solution.

5.1. Experiment 1: Without Learning

As a baseline for the results, we first examined the performance when learning was not used at all in this map. In this experiment the subgoals are placed at a specified distance from the fire front, and the agent will drive/dig towards the closest free subgoal. The results of this experiment are shown in Table 2.

FP step divider	Number of Successes
10	10/10
9	9/10
8	3/10
7	0/10

Table 2. Results without learning. A lower FP step divider increases the propagation of the fire relative to the number of steps a bulldozer can make, thereby making the problem more difficult.

5.2. Experiment 2: Learning only SGG

In this experiment we only learn to generate subgoals and use the fixed nearby assignment module. The neural network for generating subgoals only used two inputs; the wind vector, and the distance from the fire centre to the fire-front, where wind vector means the difference between the wind direction and the line angle used to find the direction. Since we used static weather for the experiments, it was not necessary to include the wind speed. The results of this experiment are shown in Table 3.

FP step divider	Nr. Generations
10	5
9	6
8	8
7	20
6	-

Table 3. Results with learning only subgoals.

5.3. Experiment 3: Learning Assign

In this experiment we used a non learning SGG module that placed the subgoals at a fixed distance from the fire front. But now we used a learning assign module. The neural network for learning the assignment task has the following inputs: the distance from the fire to the subgoal, the distance from the fire to the agent, the distance from agent to the subgoal, the distance from the subgoal to the middle of the fire, the distance

from the agent to the north, east, south and west of the map, the wind direction and force, and for each subgoal if it is free or already assigned.

The results of learning assign are given in Table 4.

FP step divider	Nr. of Generations
10	8
9	15
8	19
7	-

Table 4. Results with learning only the assign module.

5.4. Experiment 4: Learning SGG and Assign

In this experiment we used the learning SGG module and the learning assign module. We compare two approaches: learning both modules at the same time, and learning the modules alternatively. We started with training both modules at the same time, but this did not work at all. Even when the FP step divider was set upon 10, the program was not able to learn a solution within 50 generations, hence we aborted this.

The second experiment went a lot better. Here we first trained the SGG module while using the nearby assign module. Then when this reached an acceptable level we stopped training the SGG module and evolved the Assign module. Then when Assign showed intelligent behavior we stopped training the Assign module and started training the SGG module again. This process was repeated until no significant improvements were shown. In this way, we obtained successful controllers even when the FP step divider was set to 5. A very good solution to this difficult problem was obtained after 40 generations, but after 25 generations we already had some good results.

6. Discussion

In the experiments we have seen that cooperative agent policies are evolved to solve quite large forest fires. By using incremental learning the system was able to find solutions to even harder tasks with a larger fire propagation speed. This means that incremental learning can be effective to solve even more complicated problems. We have also studied the evolution of different modules (the subgoal generator and the task assignment). It turned out that evolving them synchronously did not work well. The reason of this is that if both modules are constantly changing and initially random it takes a lot of time to progress to a solution. In other experiments, when we evolve only one module at a time, evolution is much more steady and although in the initial generations no solutions were found, the system finally learned good solutions. One way to cope with multiple adaptive modules is to train them alternatively as was done in the experiments in the last section. By optimizing both modules, but not at the same time, solutions could be found even for environments in which the fire propagation was very fast. In the future, we want to research even larger and more complex forest fires. The Bushfire simulator allows us to generate a learning environment in an easy way, and therefore it is possible to perform much more experiments. For this we want to focus on incremental learning and training all modules in an asynchronous way and test the enhancer module by generating environments containing houses or rivers.

References

Avesani, P., Perini, A., & Ricci, F. (1997). CBET: A case base exploration tool. *Proceedings of the 5th congress on the Italian Asscociation for Artificial Intelligence on Advances in Artificial Intelligence* (pp. 405–416). Spinger-Verlag, London.

Avesani, P., Perini, A., & Ricci, F. (2000). Interactive case-based planning for forest fire management. *Applied Intelligence*, *13(1)*, 41–57.

Cohen, P., Greenberg, M., Hart, D., & Howe, A. (1989). Trial by Fire: Understanding the design requirements for agents in complex environments. *AI Magazine*, *10(3)*, 32–48.

Gomez, F., & Miikkulainen, R. (1998). 2-d pole balancing with recurrent evolutionary networks. *International Conference on Artificial Neural Networks* (pp. 425–430). Berlin, New York: Springer.

Gomez, F., & Miikkulainen, R. (2003). Active guidance for a finless rocket through neuroevolution. *Genetic and Evolutionary Computation Conference (Gecco-03)* (pp. 2085–2095).

Moriarty, D. E., & Miikkulainen, R. (1996). Efficient reinforcement learning through symbiotic evolution. *Machine Learning*, *22*, 11–32.

Potter, M., & Jong, K. D. (2003). Cooperative coevolution: An architecture for evolving coadapted subcomponents. *Evolutionary Computation*, *8(1)*, 1–29.

Ricci, F., Mam, S., Marti, P., Normand, P., & Olmo, P. (1994). *CHARADE: a platform for emergencies management systems* (Technical Report 94094-07). IRST, Trento, Italy.

Watson, R., & Pollack, J. (2000). Symbiotic combination as an alternative to sexual recombination in genetic algorithms. *Proceedings of Parallel Problem Solving from Nature (PPSNVI)* (pp. 425–434).

Wiering, M. A., & Dorigo, M. (1998). Learning to control forest fires. *Proceedings of the 12th international Symposium on "Computer Science for Environmental Protection"* (pp. 378–388). Marburg: Metropolis Verlag.

List of authors