Dynamic Programming Boosting for Discriminative Macro-Action Discovery

Leonidas Lefakis[1,2] LEONIDAS.LEFAKIS@IDIAP.CH
François Fleuret[1,2] FRANCOIS.FLEURET@IDIAP.CH
[1]Idiap Research Institute, Martigny, Switzerland
[2]École Polytechnique Fédérale de Lausanne, Lausanne, Switzerland

Abstract

We consider the problem of automatic macro-action discovery in imitation learning, which we cast as one of change-point detection. Unlike prior work in change-point detection, the present work leverages discriminative learning algorithms.

Our main contribution is a novel supervised learning algorithm which extends the classical Boosting framework by combining it with dynamic programming. The resulting process alternatively improves the performance of individual strong predictors and the estimated change-points in the training sequence.

Empirical evaluation is presented for the proposed method on tasks where change-points arise naturally as part of a classification problem. Finally we show the applicability of the algorithm to macro-action discovery in imitation learning and demonstrate it allows us to solve complex image-based goal-planning problems with thousands of features.

1. Introduction

The supervised learning framework provides a large variety of powerful tools capable of addressing the complexity and variability of a number of growing applications. In this framework, methods are typically developed under an i.i.d. assumption concerning the application data. Here however we seek to leverage supervised machine learning methods in a slightly different setting. We consider the data to be generated sequentially via a mixture model comprising a latent variable, given which the i.i.d. assumption holds. The value of this latent variable however is not known during the training phase and must be inferred.

Proceedings of the 31st International Conference on Machine Learning, Beijing, China, 2014. JMLR: W&CP volume 32.

This problem of latent variable inference in sequentially generated data can be seen as one of change-point detection (Fearnhead & Liu, 2007). Given the sequentially generated data $(X, Y) \in R^d \times \{1...C\}$ change-point detection consists in finding positions n in the sequence which mark changes in the nature of the joint probability $p(X, Y)$. Change-points can then be seen as an abrupt change in the source generating the data. Such a change can concern changes either some of the parameters of the underlying model, or even to the nature of the model itself. To give a concrete example, which we will address in section 4.2, in the case of speech segmentation, a change-point in a conversation can signal either the change of speaker or a change in the nature of the background noise.

Due to the nature of the problem, most previous work on change-point detection has focused on generative approaches. Such approaches attempt to model the joint distribution $p(X, Y)$ and to detect changes in these models over the sequences. In the present work however we are concerned with classification problems which lend themselves more naturally to discriminative approaches. Thus we propose to take a discriminative view of the problem and instead look directly at the conditional probability distributions $p(Y|X)$ and more specifically at the decision function $\operatorname{argmax}_Y p(Y = y|X)$. This allows us to leverage discriminative classifiers, while avoiding the complex, and ultimately unnecessary, problem of joint distribution modeling (Ng & Jordan, 2001).

Our main contribution is a novel method that automatically assigns samples in a training set to a mixture of classifiers using the concept of macro-classes. Given a training set, our proposed algorithm DPBoost (Dynamic Programming Boosting) comprises an alternating procedure that goes back and forth between a)training classifiers and b)estimating for each sample in the training set which macro-class it should be assigned to. We are specifically interested in situations where the macro-labels have a temporal regularity. This regularity could be that they do not change "too often", that the number of times they change is upper-bounded *a priori*, or that we know in advance ex-

actly what the sequence of macro-labels will be (without knowing at what times these changes occur).

Furthermore, beyond the presentation of a novel change-point detection algorithm, the present work seeks to leverage the presented method to solve complex goal-planning tasks with a visual component. Specifically, DPBoost is employed within the framework of imitation learning to perform automatic macro-action discovery. This allows the segmentation of complex goal-planning policies into a series of simpler ones. The segmentation in turn facilitates the solution of these tasks. The tasks addressed are complex not only due to the size of the state-space but also due to the partial observability of the underlying Markov Decision Process and to their visual nature.

2. Related Work

Change-point detection (Fearnhead & Liu, 2007) has been studied extensively in statistics. Approaches typically focus on the statistical properties of the generating model to detect abrupt changes. Some such approaches method, such as CUSUM (Brown et al.) and GLR (Siegmund & Venkatraman, 1995), only compute certain properties of the data, for example the logarithm of the likelihood ratio between segments of data. On the other hand, there are methods that attempt to model the generating process either in a MAP (Braun & Muller, 1998) or Bayesian setting (Adams & MacKay, 2007).

As stated in section 1, in the present work we would like to address change-point detection from a discriminative point of view. In this respect our work present certain similarities with previous work (Harchaoui & Lévy-Leduc, 2007). There the authors use a dynamic programming formulation and LASSO to perform change-point detection. Unlike our work however the authors are interested mainly in change-point detection, whereas the present work is interested in change-point detection only as an implicit goal which will help simplify a complex classification task.

The ultimate goal of DPBoost is to train a mixture of classifiers trained on disjoint subsets of the training data. The well known framework of using a mixture of experts can, as has been noted (Jordan, 1994), be seen as a divide and conquer approach to solving complex learning tasks. Thus the problem at hand is decomposed into a number of smaller problems each addressed by a different predictor whose capacity may not be adequate to solve the over-arching task on its own.

In (Ladicky & Torr, 2011) the authors decompose the training set as to detect regions where the decision function is approximately linear thus building complex decision functions from a mixture of linear support vector machines. In the context of reinforcement learning, previous work (Li et al., 2006) has proposed to use a mixture of linear regressors to model the complex value function of the environment.

The notion of latent variables has also been employed (Felzenszwalb et al., 2010), there authors propose a latent-SVM approach to automatically discover the various different parts of an object which they seek to detect. Similarly we propose to use a latent variable to implicitly uncover the latent variables of the sequential data-generating process.

The proposed method, Dynamic Program Boosting, seeks to leverage the Boosting family of algorithms (Freund & Schapire, 1995) which can itself be seen as a way of combining multiple weak experts corresponding to the weak learners. On a higher level Boosting classifiers can themselves be mixed in more complex frameworks, e.g. Noisy-OR formulations (Kim & Cipolla, 2008) or cascade structures (Saberian & Vasconcelos, 2010). Though these approaches invariably impose some form of structure on the samples, these dependencies tend to be spatial and not temporal. Unlike these aforementioned approaches we present here an approach to combining a number Boosting classifiers in a setting where data is temporally connected.

Such settings, arise naturally in goal-planning tasks typically addressed in a reinforcement learning setting, where an intelligent agent gradually learns to solve the task at hand by interacting with the environment usually modeled as a (Partially Observable) Markov Decision Process. Due to the complexity of the tasks we wish to address we seek to leverage two extensions of this framework. These are imitation learning and hierarchical reinforcement learning.

In imitation learning (Argall et al., 2009), also referred to as learning from demonstration or mimicking, the intelligent agent is presented with a number of trajectories which are typically considered to be generated by an optimal or near-optimal policy. In practice these trajectories are created by having an expert – the teacher – navigate the environment (Ross et al., 2011). A typical approach to imitation learning, adopted here, is to cast the learning problem as a classification one (e.g. (Hadsell et al., 2009)) by using classifiers to perform state-to-action mapping.

Hierarchical reinforcement learning and more specifically the options framework (Sutton et al., 1999), offers a promising route towards scaling up reinforcement learning. In this setting, the agent has at its disposal, beyond the simple primitive actions, a number of options (variably known as skills or macro-actions) which consist each of a policy, a termination condition, and an initiation set. One of the core issues that arise in the options framework is that of automatically discovering these macro-actions (Singh et al., 2004) .

As has been previously shown (Konidaris et al., 2010), in an imitation learning setting the problem of macro-actions discovery becomes one of segmenting the teacher trajectories. Each segment is then assigned to a different predictor, this in turn can be cast as a change-point detection problem. Similar to the work presented here, previous work (Konidaris et al., 2010) forgoes modeling the joint distribution $p(X, Y)$. Instead the authors propose modeling $p(Y|X)$ directly in order to learn the value function of the Markov Decision Process. The modeling however, and by extension the change-point detection itself, is performed with a Hidden Markov Model which is a generative method. On the contrary, the present work uses a discriminative approach (namely boosting) to directly learn a policy. This in turn allows us to address more complex tasks. We not that in this setting, the macro-classes inferred by the Dynamic Programming Boosting, correspond to the discovered macro-actions in teacher trajectories.

3. Dynamic Programming Boosting

In the following we assume access to a sequence of training samples, each composed of an available signal and a label. Each of these samples is associated to a macro-class the value of which is unknown. The only prior knowledge is a) the regularity of these macro-classes, which do not change often, and b) that given these macro-classes the label can be predicted from the available signal. We consider here the general case of training a family of Q multiclass classifiers from a sequence of N training samples that we know is composed of at most Q underlying macro-classes.

Our iterative DPBoost procedure described below alternatively re-estimates the optimal macro-labeling of the samples (see § 3.2), and adds weak learners to the strong classifiers associated with the macro-labels (see § 3.3) using a standard Boosting criterion.

As stated the corresponding macro-labels are unknown during training, the assumption is made however that given their values the training samples and their labels are generated i.i.d.. We are interested in situations where the macro-labels have a temporal regularity. We focus here specifically on the case where there is an upper-bound T on the number of times said macro-labels change in the training sequence.

3.1. Objective

Let

$$(x_n, y_n) \in \mathbb{R}^D \times \{1, \dots, C\}, \quad n = 1, \dots, N$$

be a training set, where the x_n are the observable states and y_n the sample labels. Let Q be a number of classifiers (or macro-classes). Our objective is to train

$$f_q : \mathbb{R}^D \to \{1, \dots, C\}, \quad q = 1, \dots, Q$$

such that there exists a sequence of macro-labels

$$q_n \in \{1, \dots, Q\}, \quad n = 1, \dots, N$$

, with a maximum number of changes T, leading to the minimum of the prediction errors

$$\sum_n \mathbf{1}_{\{f_{q_n}(x_n) \neq y_n\}}.$$

Given a prediction loss

$$L(f_1, \dots, f_Q, q_1, \dots, q_N) = \sum_{n=1}^{N} l(f_{q_n}(x_n), y_n) \qquad (1)$$

where the per-sample loss l accounts for the mistake in prediction on sample n, we want to find a family of predictors $f_1, \dots, f_Q$ and a sequence of macro-labels $q_1, ..., q_N$ which minimize the loss, under the constraint that the number of transitions is lesser than T. That is

$$\underset{\substack{f_1,\dots,f_Q,q_1,\dots,q_N \\ \mathcal{T}(q_1,\dots,q_N) \leq T}}{\operatorname{argmin}} \quad L(f_1, \dots, f_Q, q_1, \dots, q_N).$$

where we define the number of transitions

$$\mathcal{T}_T(q_1, \dots, q_N) = \sum_{n=1}^{N-1} \mathbf{1}_{\{q_n \neq q_{n+1}\}}$$

The following sections describe an alternating procedure which goes back and forth between improving the predictors $f_1, \dots, f_Q$, given the training set and the macro-labels $q_1, \dots, q_N$, and estimating the optimal $q_1, \dots, q_N$, given the predictors $f_1, \dots, f_Q$ and the training set.

3.2. Estimating the macro-labels with Dynamic Programming

Given the training set and predictors $f_1, \dots, f_Q$, let $S_n^{q,t}$ stand for the optimal achievable loss summed over the first n samples, with exactly t transitions, and arriving at $q_n = q$. We have, $\forall n, t, q$

$$S_n^{q,t} = \begin{cases} 0 & \text{if } n = 0 \\ S_n^{q,0} + l(f_q(x_n), y_n) & \text{if } n > 0,\ t = 0 \\ \min\limits_{q'} S_{n-1}^{q',t-\mathbf{1}_{\{q \neq q'\}}} + l(f_{q'}(x_n), y_n) & \text{if } n > 0,\ t > 0 \end{cases}$$

from the $S_n^{q,t}$, we can compute the optimal sequence $q_1, \dots, q_N$ under the constraint that there are less then T transitions with

$$(q_n, t_n) = \begin{cases} \underset{q,t \leq T}{\operatorname{argmin}} \; S_N^{q,t} & \text{if } n = N \\ \underset{q,t=t_{n+1}-\mathbf{1}_{q \neq q_{n+1}}}{\operatorname{argmin}} \; S_n^{q,t} & \text{if } n < N \end{cases}$$

Table 1. Alternating DPBoost procedure.

$\forall q, f_q^1 \leftarrow Boost\left(0, \{(x_n, y_n)\}_{n=1}^N\right)$
for $k = 1, \ldots, K-1$ **do**
  $(q_1^{k+1}, \ldots, q_N^{k+1}) \leftarrow DPSeq\left(f_1^k, \ldots, f_Q^k, \{(x_n, y_n)\}_{n=1}^N\right)$
  $\forall q, f_q^{k+1} \leftarrow Boost\left(f_q^k, \{(x_n, y_n)\}_{n:\, q_n^{k+1}=q}\right)$
end for

Given this Dynamic Programming procedure we define

$$DPSeq(f_1, \ldots, f_Q, (x_1, y_1), \ldots, (x_N, y_N))$$

as the optimal sequence of macro-labels $q_1, \ldots, q_N$ it computes.

Note that this procedure can be modified to impose different constraints on the transitions. In particular we can force an exact number of transitions instead of an upper bound. Alternatively we can impose a deterministic sequence of macro-classes, letting the dynamic programming find the best timing for said transitions.

3.3. Boost

Let

$$Boost(f, \{(x_1, y_1), \ldots, (x_N, y_N)\})$$

be an update of a strong predictor f by adding one weak learner such that the loss on the provided samples is strictly improved.

Then our DPBoost algorithm, as sketched on Table 1, initializes Q strong classifiers by adding one weak learner in each using the full dataset and from there on alternates between the optimization of the macro-labels $q_1, \ldots, q_N$ using $DPSeq$ and the improvement of the strong classifiers using $Boost$ on subsets of the training set. Specifically for each predictor f_q, this subset consists of the training samples (x_n, y_n) for which $q_n = q$. Using this set $Boost$ then adds an optimal weak learner h to the ensemble f_q so that $f_q^{k+1} = f_q^k + a_q^k h_k$, where k is the Boosting round and a_q^k the weak learner coefficient. The criterion of optimality depends on the Boosting algorithm used and the family of weak learners used. In the experiments presented here the weak learners consist of classification stumps.

3.4. Convergence Analysis

Given that both the $DPSeq$ and $Boost$ steps minimize the same positive loss (1), it is easy to see that the process is guaranteed to converge to the limit.

We note that given that the weak learnability assumption holds for the underlying Boosting algorithm [1] then it automatically holds for each classifier f_q and the weights w_q it assigns to the samples (x_n, y_n) such that $q_n = q$. This follows from the fact that the weight vector w_q can be extended to all N samples by simply setting $w_q^n = 0$ for those samples for which $q_n \neq q$.

In the case where $Boost$ comprises Adaboost we can show that DPBoost's loss converges at an exponential rate.

Let $\mathbf{f}^k = (f_1^k, \ldots, f_Q^k)$ and $\mathbf{q}^k = (q_1^k, \ldots, q_N^k)$. We have that

$$L(\mathbf{f}^k, \mathbf{q}^{k+1}) = \sum_{q=1}^{Q} \sum_{n:q_n^{k+1}=q} \exp\left(-y_n f_q^{k+1}(x_n)\right).$$

Thus the loss $L(\mathbf{f}^k, \mathbf{q}^{k+1})$ can be decomposed as the sum of Q separate losses

$$\forall q,\ L_q(f_q^{k+1}, \mathbf{q}^{k+1}) = \sum_{n:q_n^{k+1}=q} \exp\left(-y_n f_q^{k+1}(x_n)\right). \quad (2)$$

For each one of these losses it can be proven (Schapire & Singer, 1999) that due to Adaboost's weak learnability assumption,

$$\exists \gamma > 0, \forall k, q, \exists \gamma_{k+1,q} \geq \gamma,$$
$$L_q(f_q^{k+1}, \mathbf{q}^{k+1}) \leq \sqrt{1 - 4\gamma_{k+1,q}^2}\, L(f_q^k, \mathbf{q}^{k+1}) \quad (3)$$

where $\frac{1}{2} - \gamma_{k+1,q}$ is the weighted error of the weak learner h_q^{k+1} added to f_q at iteration $k+1$.

It follows that

$$L(\mathbf{f}^{k+1}, \mathbf{q}^{k+1}) = \sum_q L_q(f_q^{k+1}, \mathbf{q}^{k+1}) \quad (4)$$
$$\leq \sum_q \sqrt{1 - 4\gamma^2} L_q(f_q^k, \mathbf{q}^{k+1}) \quad (5)$$
$$= \sqrt{1 - 4\gamma^2}\, L(\mathbf{f}^k, \mathbf{q}^{k+1}) \quad (6)$$
$$\leq \sqrt{1 - 4\gamma^2}\, L(\mathbf{f}^k, \mathbf{q}^k) \quad (7)$$

where equalities (4) and (6) are the loss decomposition (2), inequality (5) is due to Adaboost's weak learnability assumption, and inequality (7) comes from the dynamic programming step of DPBoost.

Given that $L(\mathbf{f}^0, \mathbf{q}^0) = N$, for every initialization $\mathbf{q}^0$, and from the fact that the exponential loss bounds the 0-1 loss from above we have that

$$L(f_q^K, \mathbf{q}^K) \leq N \prod_{k=1}^{K} \sqrt{1 - 4\gamma^2},$$

and finally

$$\hat{Pr}(f_{q_n^K}^K(x_n) \neq y_n) \leq \prod_{k=1}^{K} \sqrt{1 - 4\gamma^2} \leq \exp\left(-2 \sum_{k=1}^{K} \gamma^2\right).$$

[1] There is a $\gamma > 0$ such that for every distribution w of weights over the training data (x_n, y_n) there exists a weak learner h such that the weighted error on the data is $\epsilon \leq \frac{1}{2} - \gamma$.

4. Experiments

In order to highlight the strengths of the proposed algorithm we present a series of experiments in three distinct settings. On a synthetic task in 4.1, on a speech-to-text transcription task in 4.2, and finally on highly complex goal-planning tasks. This final set of tasks served as a primary motivation for the development of the method in 4.3.

4.1. Synthetic Experiments

We provide in Figure 1 some results on a synthetic problem. The feature space is the square $[0, 1]^2$ and there are two macro-classes. For the first macro-class, the positive class comprises the points contained in a small disc in the upper-left of the square. For the second macro-class the positive class comprises the points contained in a large centered disc. We construct the training set by changing the macro-class with probability $p = 0.01$ between each sample and by sampling the x's uniformly in $[0, 1]^2$. Thus, as per DPBoost's assumption, given the macro-class the samples are generated i.i.d.

Given such a training set of $N = 1,000$ samples, we trained predictors composed of $1,000$ stumps built from linear classifiers using the following learning procedures:

The first one has access to the macro-labels and is the best baseline one can build using Boosting and the family of predictors we consider. We separate the full training set into two separate training sets corresponding to the two macro-classes and train the two strong classifiers separately using Adaboost.

The second is our DPBoost procedure. It does not have access to the macro-labels and relies on a loose constraint on the number of changes of the macro-labels. We set a bound of $T = 20$ which is twice more than the actual number we are expected to meet. It should be noted that the procedure is extremely robust to that value.

In Fig 1 the top row shows the training samples from each macro-class extracted from the training set. Note that this representation does not show their ordering in the set, hence the temporal consistency. In all images, the circles depict the positive class in the two respective macro-classes. The middle row shows the responses of two predictors trained with AdaBoost, using for each only the samples from the corresponding macro-class. The bottom row shows the responses of two predictors built with DPBoost, trained without the knowledge of what sample belongs to what macro-class but relying on the sequence consistency by imposing that there are less than 20 macro-class transitions in the training set.

Finally we note that we also tried a second baseline which also does not use the macro-class labels. This is similar to DPBoost without regularization, i.e. without the dynamic programming component. At each Boosting step, each sample is included in the training set for the strong classifier with the best response. This procedure degenerates and produces two constant strong classifiers, one negative and one positive, each sample being associated to the one with the correct response. This result is not shown in Figure 1.

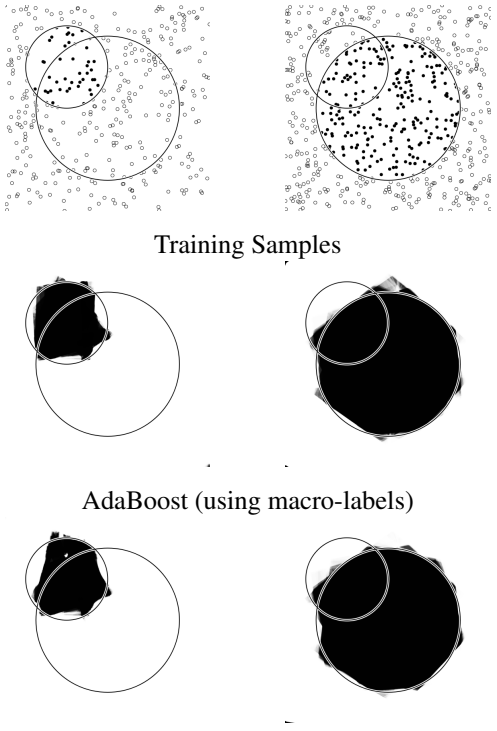


Figure 1. Synthetic task results.

4.2. Experiments in Speech-to-Text Transcription

In order to evaluate DPBoost in a more challenging setting, we consider the task of speech-to-text transcription. More specifically given an audio recording of a conversation, we would like to identify the uttered phonemes. In such a setting it is beneficial to first segment the speech according to speaker identity. This is due to the fact that audio-transcription is sensitive to the particularities of the speech patterns of different speakers. The goal here is to perform concurrent, speech-to-text transcription and speaker diarization which will allow for speaker adaptation of the individual classifiers.

We run experiments using the TIMIT dataset which is a acoustic-phonetic corpus used in the development and evaluation of automatic speech recognition systems. It contains data from 630 different users, representing eight dialects of

American English, each reading ten sentences.

For our experiments we use data corresponding to a single accent, New England, so as to make the task more challenging. We isolate from the training data, and for each user, utterances of each phoneme, and collect all utterances of the ten most common phonemes. We then create "conversations" with the following process: At any given moment, we designate a speaker and select, uniformly at random, a phoneme utterance from the data corresponding to that speaker; this process is repeated sequentially, while switching between speakers with a predefined probability p_s (in our experiments we set $p_s = 0.01$).

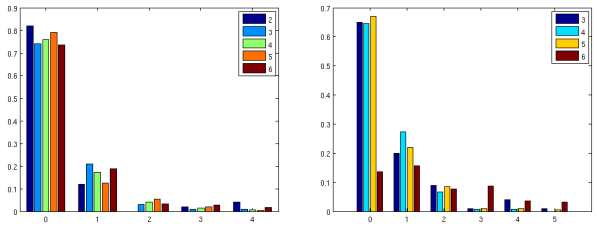
Using the above process we create batches of fifty conversations, with each batch comprising conversations with a specific number of speakers, different for every batch. Each batch is then used separately to train DPBoost with the explicit goal of perform speech to text transcription and the implicit goal of speaker diarization. Prior to training we pre-process the data to extract MFCC features (Davis & Mermelstein, 1980) which results in data of dimensionality $D = 39$.

Figure 2 (A) shows the results obtained from the DPBoost training procedure when we set the number of macro-classes equal to the number of speakers. That is we assume prior knowledge of the number of speakers in a conversation. We calculate the number of discrepancies between the macro-classes assigned by DPBoost and the true macro-classes representing speaker identity. We plot the distribution of the Hamming distance between the two labellings for conversations of each batch (for $\{2, 3, 4, 5, 6\}$ speakers in each conversation).

In Figure 2 (B) we present results when limited prior knowledge on the number of speakers is available. Specifically for each batch of conversations comprising $\{3, 4, 5, 6\}$ speakers, we set the upper limit T on the number of macro-classes to be twice the true number of speakers. For a fair amount of speakers present in the conversation (i.e. up to 5), DPBoost successfully discovers the redundancy in the number of macro-classes discarding the excessive number. In these cases the Hamming distance between macro-classes and speakers remains small. For larger parties (i.e. 6) however the process performs poorly. Thus though DPBoost exhibits robustness to the tightness of the macro-class bound, for larger problems it seems some care is needed, in the form of prior knowledge, for the method to perform well.

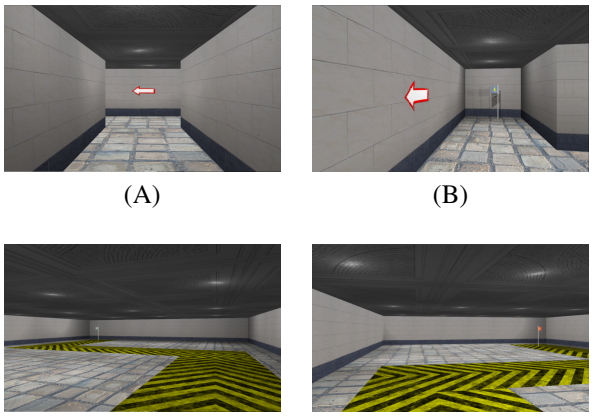
4.3. Experiments in Goal-Planning

In this section we address complex goal-planning tasks entailing an avatar navigating a 3D simulated environment. We generate this 3D environment with the OGRE



(A) (B)

Figure 2. Distribution of the Hamming distance between true and estimated macro-classes A) with and B) without prior knowledge of the number of speakers.



(A) (B) (C) (D)

Figure 3. Rendering of the avatar's view.

3D graphics rendering engine. OGRE produces a realistic rendering of the scene as seen by the avatar. The engine is capable of depicting complex 3D objects, as well as the effects of differing lighting sources. As can be seen in Figure 3 the rendered result is of high quality. The simulator is furthermore equipped with the BULLET physics engine that allows for the realistic simulation of the physical interactions between the avatar and its environment.

This environment is challenging from a goal-planning perspective due to its complexity. The size of the state-space is considerably larger than that of typical maze problems often found in the literature. The signal passed to the goal planner consists of images, thus making the problem one of visual goal-planning; as a consequence the goal-planner must also address the task from a computer vision point of view.

4.3.1. Tasks

The possible actions that the avatar can take in this 3D environment are "move forward", which moves the avatar forward by 3cm, and "turn left" or "turn right" which alter the avatar's orientation by $\pi/300$. In all the following environ-

ments the textures displayed on the walls and ground are randomly selected. The lights are set at fixed locations in order to avoid overly dark areas.

We consider the two following tasks:

Follow The Arrow The avatar is situated in a T-shaped corridor. At each of its branches there is a flag of random color. At the end of T's stem there is an arrow painted on the wall which indicates which flag the avatar must reach, see Figure 3 (D). Reaching the wrong flag results in a failed run. The length of the branches are generated at random to be between 3 and 5 meters whereas the length of the stem is set between 8 and 20 meters.

Stay On Path The avatar is situated in a large room (with dimension between 25 and 40 meters). On the floor there is a painted path painted. The avatar must follows this path to reach the flag situated at its end. Stepping outside this path results in a failed run.

4.3.2. Macro-Action Switching

During the training phase, change point detection, i.e. the macro-action discovery, is handled by the dynamic programming step of the algorithm. During the testing phase however, the algorithm must rely on switching functions

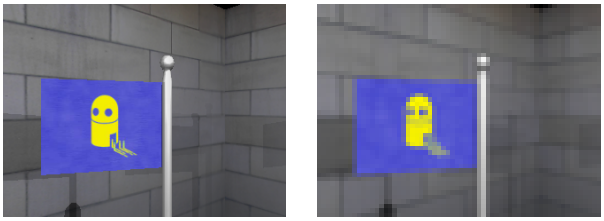
$$H_q^{q'} : \mathbb{R}^D \rightarrow \{-1, 1\}, \quad \forall q \neq q'$$

which will signal the shift from macro-action q to macro-action q'.

Such functions can be trained from the $q_1, \ldots, q_N$ estimated by $DPBoost$. In the context of goal-planning these functions would act as indicators for moving from one macro-action to the next.

In order for these switch functions to be effective in practice it is imperative that they do not signal a transition too soon. Thus we would like that $\forall n \leq m, H_q^{q'}(x_n) < 0$, where m signifies the moment of transition. On the other hand, the switching function need only respond positively at the moment of transition, thus its response is indifferent $\forall n > m$.

Based on this, the training set for each switching function $H_q^{q'}$ is built as follows, for each trajectory where the transition $q \rightarrow q'$ occurs at moment m, we gather the samples x_n for which $n < m$ and couple them with a negative label $y_n^{q \rightarrow q'} = -1$. Here $y_n^{q \rightarrow q'}$ signifies the label of sample x_n for training $H_q^{q'}$ as opposed to its true label y_n. For the positive samples we simply need the samples x_m at the moment the transitions from $q \rightarrow q'$ occur in the various trajectories. In practice however in order to make our switching functions more robust we add J positive samples $x_m, ..., x_{m+J-1}$ from each trajectory.



(A) (B)

Figure 4. (A) shows the original image I_t rendered by the simulator. (B) is the downscaled version I_t^s of the image which is passed to DPBoost

Using the constructed training sets $\{x_n, y_n^{q \rightarrow q'}\}$, the functions $H_q^{q'}$ are learned using AdaBoost with classification stumps. Of course any other discriminative approach can be employed at this step.

4.3.3. Results

For each of the aforementioned tasks we generate twenty trajectories, using a hard-coded teacher, which represent the optimal policy in the given environment. It should be noted that the teachers are given access to knowledge which is not available to the avatar (at least not directly). For example, in the case of "follow the arrow" the teacher is aware of the direction pointed to by the arrow. By contrast, the avatar's only information at each time step is its view, to be more specific, the data x_t available to DPBoost consist of the RGB values of a scaled down version (from 320×240 to 64×48 pixels) of the avatar's current view, see Figure 4. As this hidden state is not readily available to the avatar it must be inferred with the help of the goal-planner.

Follow the Arrow This task presents the avatar with the further challenge that given solely the avatar's view, the environment is only partially observable; without further information the avatar cannot know whether it has already seen the arrow, or the direction the arrow is pointing in. Without this knowledge it is unclear, to the avatar, whether it should be looking for the arrow or a flag (or even which flag).

In order to empirically evaluate our approach we train three DPBoost goal-planners, where we set the upper limit on the number of macro-actions to $\{2, 3, 4\}$ respectively. We compare DPBoost against a baseline which consists of a goal-planner based on a single AdaBoost classifier which performs state-to-action mapping; in this case there is no Dynamic Programming component to the learning phase nor any switching function. This baseline is denoted with a dash in Table 2.

Observing the training process of DPBoost, in the case of three macro-actions, we notice the discovery of three intu-

Table 2. Goal-Planning Results (Percentage of successful runs). In both cases the skill-tree baseline has a 0% success rate.

Stay on Path Results				
Nb. of macro-actions	-	2	3	4
Success Rate	16 %	52%	64%	48%

Follow The Arrow				
Nb. of macro-actions	-	2	3	4
Success Rate	28 %	40%	56%	24%

itive policies, namely "Go to the arrow", "Turn in direction of arrow", and "Go to flag". In Table 2, we present the percentage of successful test runs for the baseline goal-planner and DPBoost with $\{2, 3, 4\}$, macro-actions, that is we count, over 25 test runs, the percentage of times the avatar successfully completes the task.

As can be seen DPBoost performs considerably better than the baseline when we set the number of macro-actions to 2 or 3. For 4 macro-actions however the performance degrades to the level of the baseline. Thus it is clear that some care must be taken in setting the number of macro-actions.

Stay On Path The second goal-planning task, does not present the avatar with the challenge of partial observability, it remains however a very challenging task as the avatar must infer from its current view its position on the path, and by extent whether it should continue to go forward or start turning. Adding to the difficulty of the task is the fact that a single mistake, i.e. stepping outside the path, is enough to result in a failed run.

As can be seen by the results in Table 2, the tasks proves too challenging for the baseline. As before we show results for $\{2, 3, 4\}$ macro-actions. In this case, DPBoost proves to be more robust to the choice of the number of macro-actions. Though there does not seem to be an intuitive interpretation of the discovered macro-actions, the presented results demonstrate that these macro-actions are necessary for completing the task.

4.3.4. Comparison to Skill Trees

As mentioned, the previous work on skill trees (Konidaris et al., 2010) is closely related to the present work in that it first addressed automatic macro-action discovery in imitation learning by casting the problem as one of change-point detection. Skill trees however attempt to learn a function approximation of the value function, instead of a state-to-action mapping To do so they employ a generative framework (HMMs) to perform change-point detection.

Furthermore skill trees, before performing trajectory segmentation, first project the data onto a Fourier Basis the size of which grows exponentially with the size of the input features. As the dimensionality of our data is of the order of magnitude of a few thousand, the skill tree approach cannot be applied as is. Instead in order to apply skill trees to our goal-planning tasks we first employ slow feature analysis (Kompella et al., 2012) to reduce the dimensionality of the data.

Once the projected data has been segmented and the skill trees constructed, we use the data in each node of the tree to build a state-to-action mapping for the specific macro-action corresponding to the node. We experimented both with training these classifiers in the projected space, as well as in the original state space representation (RGB images). In both cases the agent failed at the goal-planning tasks. Over the 25 test runs, the agent never succeeded in reaching the objective. This highlights one of the main strengths of the proposed method. Namely that it is able to perform macro-action discovery in high-dimensional space while addressing the discriminative objective. This allows it to solve complex goal-planning tasks which prove too challenging even for state-of-the-art methods.

5. Conclusions

We presented a novel method which combines a family of discriminative learning algorithms with dynamic programming to perform change-point detection in sequentially generated data. The strength of the proposed method was shown empirically on a synthetic data problem, on speech-to-text transcription, and on complex, and at times partially observable, goal-planning tasks. In the case of speech-to-text transcription, DPBoost was shown to successfully address the implicit goal of speaker diarization. In the goal-planning case DPBoost was shown to successfully discover macro-actions necessary to address the complexity of the environment.

We note that DPBoost's generic nature allows it to be used in a variety of applications. We aim to investigate its applicability to automatic pose discovery in complex computer vision tasks. We also plan to extend the DPBoost framework to combine it with the DAGGER framework (Ross et al., 2011) to address more complex goal-planning tasks.

Acknowledgements This work was supported by the Hasler Foundation (www.haslerstiftung.ch) through the MASH2 project and by the European Community's Seventh Framework Programme - Challenge 2 - Cognitive Systems, Interaction, Robotics - under grant agreement No 247022 - MASH. The authors would like to thank Marc Ferrás for his help with the speech experiments and George Konidaris for providing code and for helpful discussions. The simulator has been developed by Philip Abbet.

References

Adams, Ryan Prescott and MacKay, David J.C. Bayesian online changepoint detection. *Technical Report, University of Cambridge*, 2007.

Argall, Brenna, Chernova, Sonia, Veloso, Manuela M., and Browning, Brett. A survey of robot learning from demonstration. *Robotics and Autonomous Systems*, 57 (5):469–483, 2009.

Braun, J. V. and Muller, H. G. Statistical methods for DNA sequence segmentation. *Statistical Science*, 13:142–162, 1998.

Brown, R. L., Durbin, J., and Evans, J. M. Techniques for testing the constancy of regression relationships over time. *Journal of the Royal Statistical Society. Series B*, (2):149–192.

Davis, S. and Mermelstein, P. Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences. *Acoustics, Speech and Signal Processing, IEEE Transactions on*, 28(4):357–366, 1980.

Fearnhead, P and Liu, Z. Online inference for multiple changepoint problems. *Journal of the Royal Statistical Society: Series B*, 69(4):589–605, 2007.

Felzenszwalb, P.F., Girshick, R.B., McAllester, D., and Ramanan, D. Object detection with discriminatively trained part-based models. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 32(9):1627 – 1645, 2010.

Freund, Yoav and Schapire, Robert E. A decision-theoretic generalization of on-line learning and an application to boosting. In *Proceedings of the Second European Conference on Computational Learning Theory*, EuroCOLT, 1995.

Hadsell, Raia, Sermanet, Pierre, Ben, Jan, Erkan, Ayse, Scoffier, Marco, Kavukcuoglu, Koray, Muller, Urs, and LeCun, Yann. Learning long-range vision for autonomous off-road driving. *Journal of Field Robotics*, 26(2):120–144, February 2009.

Harchaoui, Zaïd and Lévy-Leduc, Céline. Catching change-points with lasso. In *Proceedings of Neural Information Processing Systems (NIPS)*, 2007.

Jordan, Michael I. Hierarchical mixtures of experts and the em algorithm. *Neural Computation*, 6:181–214, 1994.

Kim, Tae-Kyun and Cipolla, Roberto. Mcboost: Multiple classifier boosting for perceptual co-clustering of images and visual features. In *Proceedings of Neural Information Processing Systems (NIPS)*, pp. 841–856, 2008.

Kompella, Varun Raj, Luciw, Matthew D., and Schmidhuber, Jürgen. Incremental slow feature analysis: Adaptive low-complexity slow feature updating from high-dimensional input streams. *Neural Computation*, 24(11): 2994–3024, 2012.

Konidaris, George, Kuindersma, Scott, Barto, Andrew G., and Grupen, Roderic A. Constructing skill trees for reinforcement learning agents from demonstration trajectories. In *Proceedings of Neural Information Processing Systems (NIPS)*, pp. 1162–1170, 2010.

Ladicky, Lubor and Torr, Philip H. S. Locally linear support vector machines. In *Proceedings of the International Conference on Machine learning (ICML)*, pp. 985–992, 2011.

Li, Hui, Liao, Xuejun, and Carin, Lawrence. Region-based value iteration for partially observable markov decision processes. In *Proceedings of the International Conference on Machine learning (ICML)*, pp. 561–568, 2006.

Ng, Andrew Y. and Jordan, Michael I. On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes. In *Proceedings of Neural Information Processing Systems (NIPS)*, pp. 841–848, 2001.

Ross, Stéphane, Gordon, Geoffrey J., and Bagnell, Drew. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS)*, pp. 627–635, 2011.

Saberian, Mohammad J. and Vasconcelos, Nuno. Boosting classifier cascades. In *Proceedings of Neural Information Processing Systems (NIPS)*, pp. 2047–2055, 2010.

Schapire, RobertE. and Singer, Yoram. Improved boosting algorithms using confidence-rated predictions. *Machine Learning*, 37(3):297–336, 1999.

Siegmund, D. and Venkatraman, E. S. Using the generalized likelihood ratio statistic for sequential detection of a Change-Point. *The Annals of Statistics*, 23(1):255–271, 1995.

Singh, Satinder P., Barto, Andrew G., and Chentanez, Nuttapong. Intrinsically motivated reinforcement learning. In *Proceedings of Neural Information Processing Systems (NIPS)*, pp. 1281–1288, 2004.

Sutton, Richard S., Precup, Doina, and Singh, Satinder P. Between MDPs and Semi-MDPs: A framework for temporal abstraction in reinforcement learning. *Artifificial Intelligence*, 112(1-2):181–211, 1999.