Multi-class Classification with Machine Learning and Fusion

Cristina GARCIA CIFUENTES
Double Degree student at TELECOM ParisTech

Year 2008-2009

Intership supervisor: M. Marc STURZEL, EADS Innovation Works SI-IS
Academic supervisor: M. Michel ROUX, Telecom ParisTech TSI

Intership supervisor visa	Security service visa
Date and signature	Date and signature

This document is classified as « Réservé EADS ».
It contains 74 pages numbered from n° 1/74 to 74/74.
3 copies were printed.

Acknowledgements

I would like to thank TELECOM ParisTech and EADS IW for the valuable opportunity of studying and working in France.

Thanks also to Marc and Livier for their helpful guidelines, comments and suggestions, and for the trust they have put in me.

Many thanks to all the members of the laboratory for the nice atmosphere, especially to my friends Manu and Andre. It has been a pleasure to share this experience with you all.

I am especially grateful to Salva, my family and friends, for their unconditional love and support.

Abstract

Multi-class classification is the core issue of many pattern recognition tasks. Several applications require high-end machine learning solutions to provide satisfying results in operational contexts. However, most efficient ones, like SVM or Boosting, are generally mono-class, which introduces the problem of translating a global multi-class problem is several binary problems, while still being able to provide at the end an answer to the original multi-class issue.

Present work aims at providing a solution to this multi-class problematic, by introducing a complete framework with a strong probabilistic and structured basis. It includes the study of error correcting output codes correlated with the definition of an optimal subdivision of the multi-class issue in several binary problems, in a complete automatic way. Machine learning algorithms are studied and benchmarked to facilitate and justify the final selection. Coupling of automatically calibrated classifiers output is obtained by applying iterative constrained regularisations, and a logical temporal fusion is applied on temporal-redundant data (like tracked vehicles) to enhance performances. Finally, ranking scores are computed to optimize precision and recall is ranking-based systems.

Each step of the previously described system has been analysed from a theoretical an empirical point of view and new contributions are introduced, so as to obtain a complete mathematically coherent framework which is both generic and easy-to-use, as the learning procedure is almost completely automatic. On top of that, quantitative evaluations on two completely different datasets have assessed both the exactitude of previous assertions and the improvements that were achieved compared to previous methods.

Keywords: *machine learning, classification, fusion, pattern recognition, multiclass, calibration, error correcting output codes (ECOC), coupling, logical fusion, SVM, HTM Numenta, probabilistic framework.*

Contents

Table of figures

Presentation of EADS

EADS

EADS (European Aeronautic Defence and Space Company) emerged in 2000 from the link-up of the German *DaimlerChrysler Aerospace AG*, the French *Aérospatiale Matra* and the Spanish *Construcciones Aeronauticas SA*. EADS is a global leader in aerospace, defence and related services. The group includes the aircraft manufacturer Airbus, the world's largest helicopter supplier Eurocopter and EADS Astrium, the European leader in space programmes from Ariane to Galileo. EADS is also the major partner in the Eurofighter consortium and a stakeholder in the missile systems provider MBDA, as well as the developer of the military transport aircraft A400M.

It employs about 116,000 people at more than 70 production sites, above all in France, Germany, Great Britain and Spain as well as in the U.S. and Australia. A global network of more than 30 Representative Offices maintains contact with the customers. In 2007, the company generated revenues of € 39.1 billion.

The following diagram shows the Divisions into which the company is subdivided corresponding to the product range and the transversal service *Innovation Works* in charge of the research and development activities for the whole group.

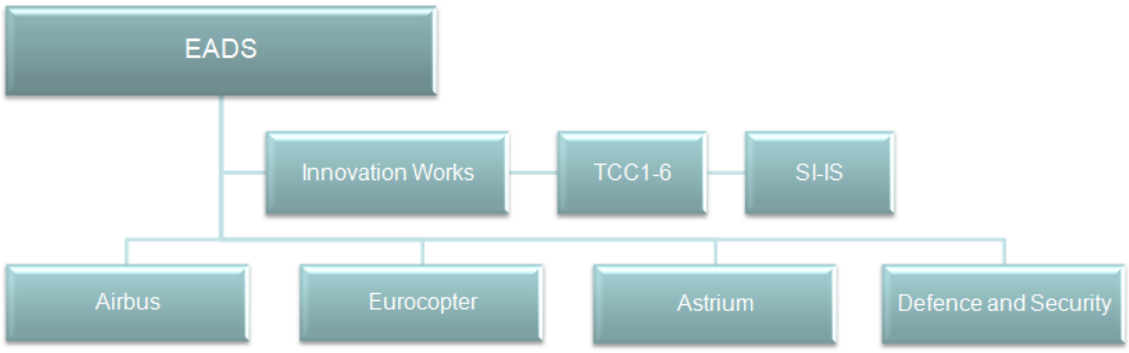


Figure 1. Organisation of EADS

EADS Innovation Works

EADS is convinced that continuous innovation has been the basis of its success in the past, so this is one of the key areas EADS is focusing on as growth drivers for the future.

	2007	2006	2005	2004
New inventions filed (some of which covered by several patents)	967	792	586	521
EADS patents portfolio (year end)	20,653	18,366	15,036	13,515

Table 1. Annual inventions and patents by EADS

EADS Innovations Works is in charge of the EADS corporate Research and Technology production facilities that guarantee the group's technical innovation potential with a focus on the long-term horizon. Driven by the EADS R&T strategy, it identifies new technologies that will create value and competitive advantages. It is located in Germany, France, Spain, UK, Singapore and Russia and it employs approximately 600 people including doctorates and university interns. These production facilities are organized according to a segmentation of 6 transnational *Technical Capability Centres* (TCC), consistent with the R&T strategy and covering the skills and technology fields that are critical for the Group:

- TCC1 - Composites Technologies
- TCC2 - Metallic Technologies and Surface Engineering
- TCC3 - Structures Engineering, Production and Mechatronics
- TCC4 - Sensors, Electronics and Systems Integration
- TCC5 - Simulation, Information Technologies and Systems Engineering,
- TCC6 - Advanced Concepts

TCC4 and SI-IS

With a workforce of approximately 75 people, the **TCC4** supplies a high level research in the fields of microwaves, biological and chemical sensors, electronics and communication, autonomous systems, image and signal processing, optronics, chemical process engineering, natural radiation environment, and technologies and reliability in electronic systems.

The **SI-IS** research team has been integrated in EADS IW since 2007. Its mission consists of proposing and developing innovative image processing solutions that will improve the performance or reliability of future EADS products. Some application areas are:

- navigation, guidance and vision for piloted or autonomous systems,
- advanced video-surveillance,
- intelligence systems,
- ground stations for satellites, UAV (Unmanned Air Vehicles), and reconnaissance aircrafts.

The developed technologies encompass the following technical domains:

- image enhancement,
- sensor modeling,
- video and image analysis: motion detection, tracking, automatic annotation, indexation, data mining, pattern recognition…
- image-based techniques for navigation,
- compression.

1 Context

The internship took place within the SI-IS research team, in the context of the Infom@gic URBAN VIEW project, which focuses on following problems: consider a net of surveillance cameras and the number of hours of video that they generate. In order to facilitate the retrieval of an interesting piece of information from this huge amount of data, it is necessary to extract metadata that will allow to quickly access the relevant parts of the videos.

The Infom@gic processing chain handles the videos obtained from a net of urban surveillance cameras and applies motion detection techniques in order to track the different moving objects, such as pedestrians and different kinds of vehicles. The different objects tracked are stored in a database together with information about motion, viewing conditions, colour, place, time... An important step in the processing chain is object categorisation, as it allows the access to the database via queries indicating the type of vehicle. The results of the request have to be a selection of tracks ranked according to their pertinence: those tracks with high probability of belonging to the desired category will be shown first and the ambiguous ones later, so that desired vehicle can be efficiently found among the results.

This is an example of multiclass classification: we have a number of categories or classes (pedestrian, car, van, bus...) and a set of objects (the images of the tracked vehicles with the associated motion information) and we want to assign one of those categories to each object. Multiclass classification is typically carried out through machine learning techniques, which will be presented in the following section. Besides making a decision about the category of each object, it is interesting to assess the reliability of this decision, so to allow a correct ranking of the retrieved objects.

The objective of the internship has been the design and implementation of a classification system compatible with this context, therefore in compliance with several desired characteristics such as the multiclass nature, the consideration of temporal aspects to infer the category of the object, an output in the form of a score adapted to an efficient ranking, and the user's control of the prediction time according to its needs.

The aim has been, however, the development of a general purpose system, regardless of the type of data to be classified or the number of categories, so that no user tuning is needed in order to adapt the system to different applications.

2 Machine learning: an overview

2.1 Introduction

The learning mechanism is a fascinating characteristic and a distinctive attribute of intelligent behaviour, and one of the central challenges addressed by *artificial intelligence* research from its beginning. This is why *machine learning* has been developed, in an attempt to making computers imitate the innate ability of humans to acquire facts, skills and more abstract concepts.

Machine learning (ML) aims at enabling computers to automatically learn from data, that is to say to take data as input and give as output algorithms capable of performing, over the same kind of data, a desired task.

Typically, the task consists in making a prediction or assigning a category to an object, and taking an action based on this. The data can be of any nature: image, sound or video, as well as any other kind of textual or numerical data describing a real world situation. Independently of the type of the data, there are two major steps to be done. First, analyse the raw data to convert it into a format suitable for ML. This step is often called feature extraction, and may consist of groups of measurements or observations, defining points in an appropriate multidimensional space. Then, apply ML techniques in order to perform the task of interest. Therefore, ML is suitable for wide range of applications, including search engines, medical diagnosis, stock market analysis, speech and handwriting recognition, robot locomotion, etc.

The power of ML underlies in the fact that data is too variable and ambiguous to be treated "by hand" or to be interpreted with explicit rules covering all possible cases. ML techniques address such problems by using statistics to model large amounts of elementary information and their relations. In summary, ML makes it possible to perform complex tasks over challenging data, where the information to be extracted is never explicit, but rather hidden behind the data statistical properties. On the top of that, in recent years it has been proved to be more efficient than model-based approaches in most cases, even when such models are feasible.

2.2 Machine learning paradigms

In the case of humans, learning is the innate ability to acquire facts, skills and more abstract concepts. [1] describes the many facets of this phenomenon:

Learning processes include the acquisition of new declarative knowledge, the development of motor and cognitive skills through instruction or practice, the organization of new knowledge into general, effective representations, and the discovery of new facts and theories through observation and experimentation.

The study and computer modeling of learning processes constitutes the core of machine learning.

It has been developed around three research lines, which borrow issues and inspiration from one another [1]: the simulation of human cognitive processes, the theoretical exploration of possible learning

methods and algorithms independent of the application domain, and task-oriented studies. Machine learning methods are mainly the results of the two latest.

We can distinguish different machine learning strategies according to the amount of inference performed by the learner on the information provided by the teacher [1]: rote learning, learning from instruction, learning by analogy and learning from examples.

Learning from examples has recently become so popular that it is simply called *learning*. Given a limited set of examples of a concept, the *learning problem* can be described as finding a general description of the concept, in the form of a concise rule that explains the examples [2].

Learning techniques can be grouped in three big families: *supervised learning*, *reinforcement learning* and *unsupervised learning*. In supervised learning, labeled examples are provided and the aim is to find a function that can yield the output given the input. In unsupervised learning the aim is to extract some structure from the data. In this case, the concise description of the data can be a set of clusters or a probability density, and the algorithm is given only unlabeled examples. In reinforcement learning, the algorithm learns a policy of how to act given an observation of the world. Every action has some impact in the environment, and the environment provides feedback that guides the learning algorithm, but the correct answers are not given explicitly.

In the following section, we focus on supervised learning, as this is the kind of learning algorithm that best characterises the classification problem.

2.3 Supervised learning

In supervised learning, the examples are already associated with their target values. Given a sample of input-output pairs $D = \{(\mathbf{x}_1, y_1), ..., (\mathbf{x}_N, y_N)\} \in \mathbf{R}^d \times Y$ –called the *training set*– the task is to find a deterministic function $f : \mathbf{R}^d \rightarrow Y$ that maps any input $\mathbf{x}$ to an output, so that it can predict the output of future observations minimizing the errors as much as possible. This obtained function is the concise description of the data, and is also called *learning machine* or *model*.

According to the type of output, there are two kinds of supervised learning: *regression* and *classification*.

In a *regression* problem (or *function learning*) the output space is formed by continuous variables, i.e. $Y \subseteq \mathbf{R}^K$. A typical example is the prediction of stock market indexes.

In *classification* problem, the output space has no structure except whether two elements are equal or not. The output space consists of a number of discrete *classes* or *categories*, i.e. $Y = \{C_1, ..., C_K\}$. The learning machine that solves the problem is called *classifier*. This problem characterizes most pattern recognition tasks, for example the recognition of handwritten letters and digits.

2.4 Model selection

The performance of a supervised learning method is assessed by measuring its *generalisation* capability: the resulting model must not only explain the training set, but more importantly make accurate predictions for cases that are not in the training set. Learning methods can suffer from either underfitting or overfitting. A model that is not sufficiently complex can fail to describe all pertinent properties in a complicated data set, leading to underfitting. A model that is too complex may fit the noise in the training set, not just the information, leading to overfitting. Underfitting produces excessive bias in the outputs, whereas overfitting produces excessive variance, so there is a trade-off between the two.

Learning methods are characterised by the presence of parameters that have to be tuned to obtain optimal performances. The same learning algorithm can be trained using different configurations of parameters, generating different learning machines. The problem of selecting the best one among different learning machines is called *model selection*.

A usual approach for model selection and for generalization error estimation consists in dividing the available data in three subsets: a *training* set, a *validation* set and a *test* set. The training set is used to train different models that use different configurations of parameters. The validation set is used to estimate the performance of the different models with the aim of picking the best one. The test set is used to estimate the generalization error after having selected the final model.

The theoretical issues of learning, such as model complexity and bounds on the generalization error, are discussed by the *statistical learning theory*, which provides a mathematical framework for the learning problem. See [2] for more details.

3 Problem statement

The internship aims at designing and implementing a pattern recognition system suitable for multiclass classification and rankings. In the target application, we want to access to a database whose elements belong to a finite set of categories and a query consists of one of these categories. Results to a query must be shown in descending order of pertinence: the most confident elements must appear first, the ambiguous elements later.

In order to achieve this, the system must assign to each element a set of scores which serve as basis to rank the elements with regards to each class.

Furthermore, the system has to be able to handle several observations of the same object, corresponding typically to several video captures of this object over the time. The number of observations of each object is variable. Thus, the system has to be able to fuse a variable number of sources of information when they are provided.

The system must work for any number of categories and regardless of the nature of the data. Moreover, it is expected to be used not only for the database ranking but also in real time classification, so prediction time constraints must be introduced somehow.

4 Presentation of the solution

4.1 An approach to the multiclass classification problem

In the general case, a classification problem consists in assigning elements to a finite set of classes or categories. Some machine learning models, such as decision trees and neural networks, are able to naturally handle the multiclass case. For others, such as boosting and support-vector machines, which were conceived for distinguishing between only two classes, a direct extension to the multiclass case may be problematic [14].

In such cases the multiclass problem is typically reduced to many binary classification problems that can be solved separately. The resulting set of binary classifiers must then be combined in some way [14]. From this point of view, each binary classifier is a source of information. Both the sources and the mode of fusion must be chosen. Moreover, we want to ascribe a confidence to the decision.

The straightforward solution for the reduction is to create k binary problems, where k is the number of classes. Each binary problem involves the discrimination between one class and the set of all other classes. This reduction is called *one-against-all* [14]. One might expect that only one of the classifiers responds positively to an input sample – only the classifier corresponding to the real class. It would then be easy to make the decision. Unfortunately, this is not always the case. In practice, many classifiers may respond positively or, on the contrary, none of them. The approach becomes therefore problematic.

Calibration

Often, the classifier's output is a *score* whose magnitude can be taken as a measure of confidence in the prediction. This means that these scores can be used to rank the examples in the test set from the most probable member to the least probable member of the positive class with regards to *this* classifier. This could lead to decide according to the class whose corresponding classifier has the highest score. However, this would not be appropriate, as the scores of different classifiers cannot be compared directly: they are problem dependent and – even after re-scaling them – they do not represent the probability that the input is a member of the class of interest.

Probability estimates are needed when the classification output has to be combined with or compared to other sources of information for decision-making. The transformation of a classifier's score to a probability estimate is called *calibration*. Calibration is essential if we want the outputs to be directly interpretable as the chance of membership of the class.

Besides, it must be noticed that this kind of classifiers which are specialized in the detection of a single class can be trained towards different objectives depending on the application. Sometimes we are interested in detecting all targets. This is only achievable at the expense of a higher number of false alarms. On the contrary, in other cases it is preferable to miss targets rather than to raise false alarms. The objective may also be to minimize the number of mistakes in terms of false alarms and non-detections. Clearly, in the case of contradiction between detectors of the same class that have been trained differently, depending on their answers it is logical to trust more one of them rather than another, and the decision can be richer than if we had only one classifier. To take advantage of this aspect, we should find a way of quantifying this preference. Calibration is also an answer to this question.

Exploiting redundancy

At this point, we have been able to overcome some of the limitations of the one-against-all approach the way it has been presented here, notably with regards to the decision-making. Nevertheless, there are clues that lead to think that other ways of reducing the multiclass problem could be possible and useful.

To begin with, it has been noticed that apparently redundant classifiers (classifiers trained differently to detect the same class) may indeed add extra information and improve the decision. This does not match with the "one classifier per class" scheme.

Moreover, if we want to know about one particular class, in this scheme it seems that only the specialized classifier gives information whereas nothing can be inferred from all the rest of them. Are we making the most of our classifiers?

Furthermore, we may wonder which one of these situations is preferable: a classifier that is capable to tell one class A from the rest but is *sometimes* mistaken, or a classifier that completely confuses A with B but tells *for sure* A or B from the rest. The answer clearly depends on the complementary information provided by the rest of classifiers.

All this leads to envisage another approach: the combination of classifiers that discriminate between *groups of classes*.

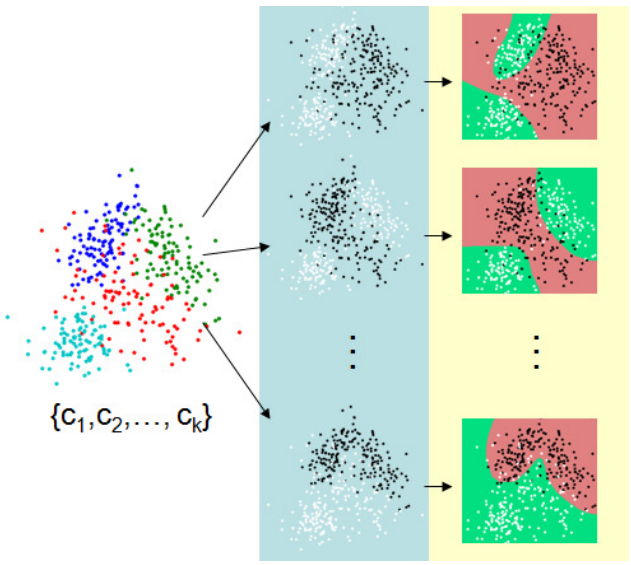


Figure 2. Groups of classes and their corresponding binary classifiers

Instead of designing individual classifiers to detect individual classes, the idea is to design the *set of classifiers* to respond differently to each class. Therefore, the set of answers is taken into account to predict the class. Many classifiers may respond equally to two classes, but other classifiers are designed to respond differently, so the two classes are distinguishable.

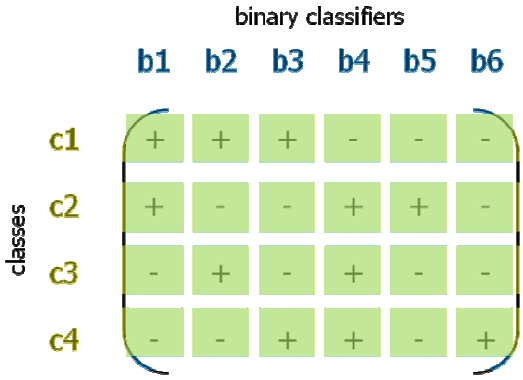


Figure 3. Example of Coding Matrix.

Dietterich and Bakiri (1995) proposed to reduce the multiclass problem into multiple binary problems by means of assigning each class to a word of an **error-correcting code**. The one-against-all reduction is a particular case of this general framework. A possible way of classifying an example is to compare the set of obtained answers to each of the **code words**: the predicted class is the one which fits best.

The variety of possible classifiers is much wider that in the one-against-all case and the associated discrimination problems may be easier, so the performances of the individual classifiers are potentially better. Moreover, the number of classifiers is not limited to *k*. Compared to the one-against-all strategy, this approach may add redundancy and robustness to individual classifiers mistakes.

4.2 Overview of the designed classification strategy

In the design of the new classification system, we have tried as much as possible to take advantage of the work that has already been done in the laboratory with regards to this subject. This is why we propose an extension of the existing classification strategy – which consists mainly of single-class detection and biclass disambiguation – in order to achieve the objective in an effective way.

On the one hand, the common points are the modelling of the classification problem by means of several binary classifiers and the use of the same machine learning technologies. On the other hand, the new classification strategy extends the previous one in two fundamental aspects. First, it provides a probabilistic framework for the classification system and thus a basis for the computation of ranking scores with strong theoretical foundation. Second, it completely generalises the type and number of binary problems.

Having many observations of an object and the application of several classifiers to each observation make it necessary to use a fusion method. We have chosen distributed fusion, in contrast to centralised fusion. This means that fusion is done in two steps: we first combine all the information concerning one observation and then the results of these combinations on all observations of the same object. Therefore, we treat less data at the same time, with the resulting computational advantages. Moreover, this makes the system flexible to the number of observations. In particular, in the case of temporal tracking of an object, this makes possible to give preliminary answers before having all the observations. Usually, the risk of distributed fusion is the weak cooperation among sources and possible contradictory decisions. This is not critical in our case because the result obtained for each observation is a probability distribution over the classes rather than a local decision. As we will see in sections §5.4 and §5.5, the specific rule for fusing is conditioned by the probabilistic nature of the results. The first step reduces itself to solving an over-constrained equation system, and the second is a probabilistic combination of distributions. See section §5.5.1 for more details on information fusion.

Inference procedure

In this section, the different functional components of the system and their relationships are presented. In order to infer its class, the treatment of a single observation consists of the following steps:

① **Set of *b* binary classifiers**. Each classifier distinguishes between a group of classes and the rest of classes. They have been trained and optimised[1] individually with the learning and the validation sets, labelled according to a coding matrix[2]. This set of classifiers represents the reduction of the multiclass problem into several binary problems.

② **Calibration of scores**. Each classifier has been calibrated with the validation base. Each score is transformed into an estimate of the probability of belonging to the corresponding disjunction of classes:

$$r_j = \tilde{\mathrm{P}}\left(y \in A_j \middle| \mathbf{x}\right) \qquad j = 1,\ldots,B \quad A_j \subset \{c_1,\ldots,c_k\}$$

[1] Model selection is done through a genetic algorithm.
[2] The procedure for obtaining this coding matrix is detailed in section §5.3.2

with: $\mathbf{x}$ = the observation to be classified

y = the real class of this object

j = each of the classifiers

A_j = set of classes considered positive by the j-th classifier

B = number of classifiers

k = number of classes

③ **Coupling of these estimates**. The estimates r_j are combined according to the coding matrix to obtain an estimate of the probability of belonging to each class:

$$\tilde{\mathrm{P}}(c_i|\,\mathbf{x}) \quad i=1,\ldots,k$$

Steps 1 to 3 are summarised in Figure 4.

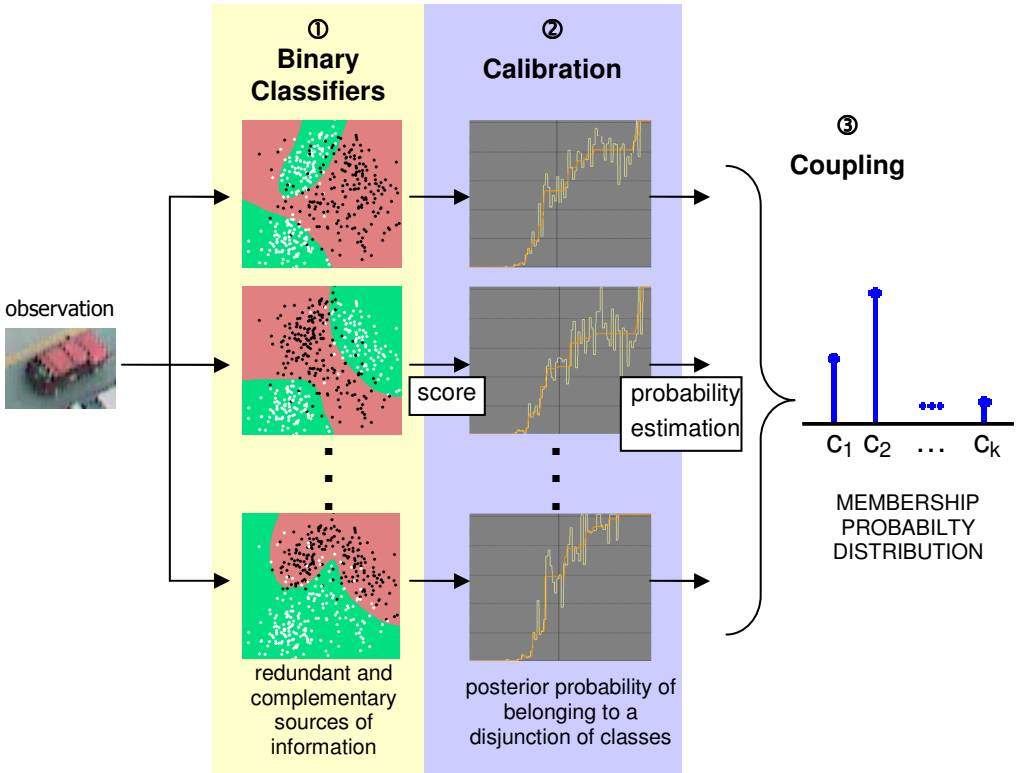


Figure 4. Treatment of a single observation.

Then, if there are may observations of the same object:

④ **Temporal fusion**. Steps ①,② and ③ are done for all the observations. We then take the probability distributions obtained for each observation and we consider them as different beliefs on the object's category. These beliefs are fused into a single global belief, which is a new distribution.

A decision on the category can be made using the obtained probability distribution. If we want to be able to rank all the objects in a database, a further step has to be done:

⑤ **Computation of the ranking scores**. The probability distribution is used to compute one ranking score per class. Roughly, the score associated to a class is related to the distance between the obtained distribution and a Dirac delta distribution.

All the steps are summarized in Figure 5.

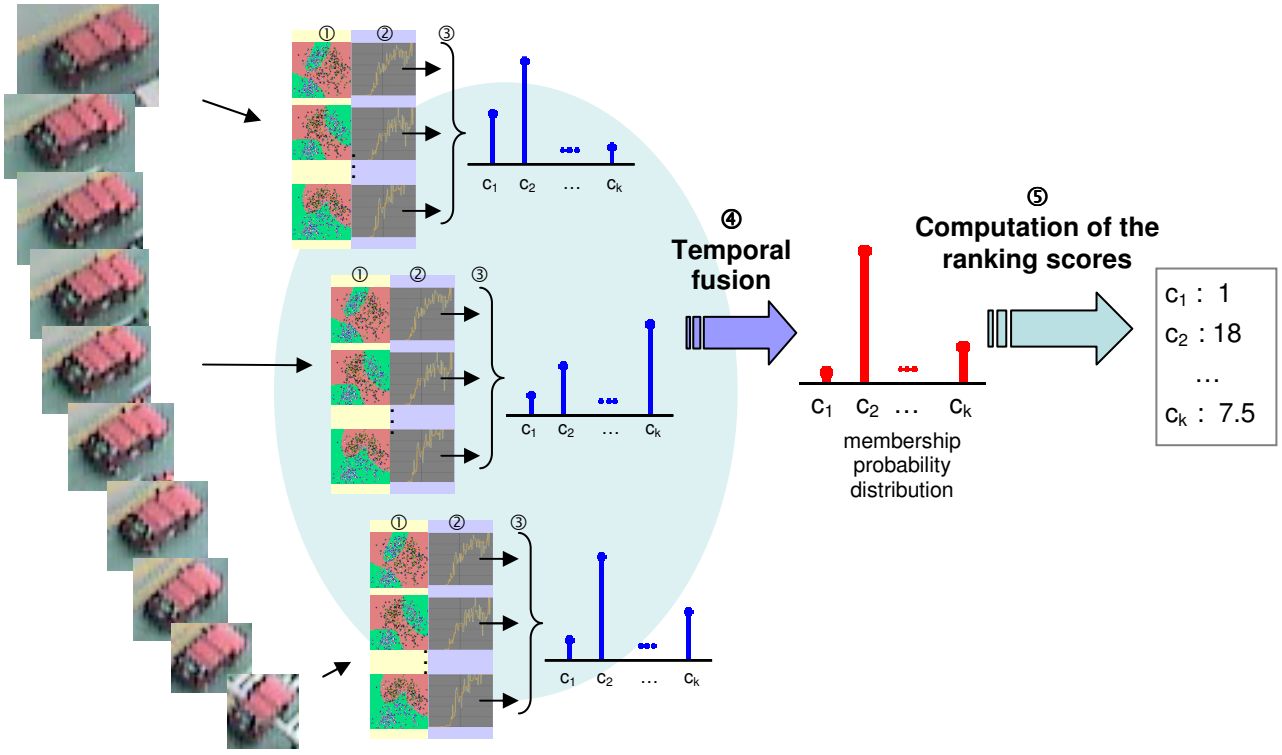


Figure 5. Processing of a track[3].

In the following section, the details on design and algorithms of each block are presented.

[3] The track consists of one or several versions of the same object.

5 System description

5.1 Binary Classifiers

5.1.1 Introduction

There is a great deal of choice for solving supervised learning problems. Neural networks have been the center of most machine learning research for long years, and have proven to be suitable for a great number of applications. In the last decade, new learning algorithms have emerged that have excellent performance, mainly support-vector machines and boosting.

In the last three years, a new technology has become popular among online articles of scientific vulgarisation, in spite of the criticism it has received from the AI community: hierarchical temporal memory networks.

These four learning methods applied to supervised binary classification problems were studied and compared. In this section, the principles and strong points of each one are presented, as well as the results of the comparative study.

5.1.2 Neural Networks

Artificial neural networks are non-linear statistical tools that can be used to model complex relationships between inputs and outputs or to find patterns in data.

They are based on the paradigm of **connectionism**[4] originally inspired by the examination of the central nervous system. According to this paradigm, the interconnection of simple processing elements (called *neurons* or *nodes*) can exhibit a complex global behaviour, determined by the connections between the processing elements. The connections between neurons have associated *weights* that can be modified through a learning process in order to associate a desired output to a given input.

A generic **artificial neuron** is shown in Figure 6. The inputs multiplied by the connections weights are summed to a bias and the result is given as input to an *activation function*[5] that gives the neuron output.

Here we focus on the **multilayer perceptron** (MLP), probably the most important kind of neural network. The MLP is a feed-forward network consisting of a set of input nodes, one or many sets of hidden nodes[6] and a set of output nodes. Thus, they have at least two layers of connections between neurons, in contrast with the *single layer perceptron*, which has only one layer of connections as there are no hidden neurons.

[4] Also known neural computation, parallel distributed processing, neurocomputing, [2].
[5] Typically a smoothed version of a step function.
[6] Only one hidden layer is needed for approximating arbitrarily well any decision boundary, provided that the number of hidden neurons is sufficiently large, [2].

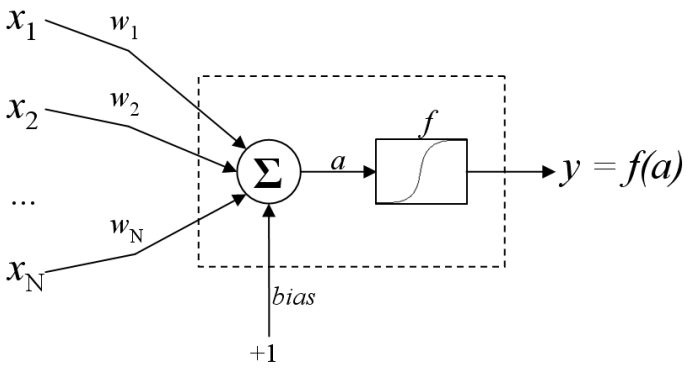


Figure 6. A generic artificial neuron.

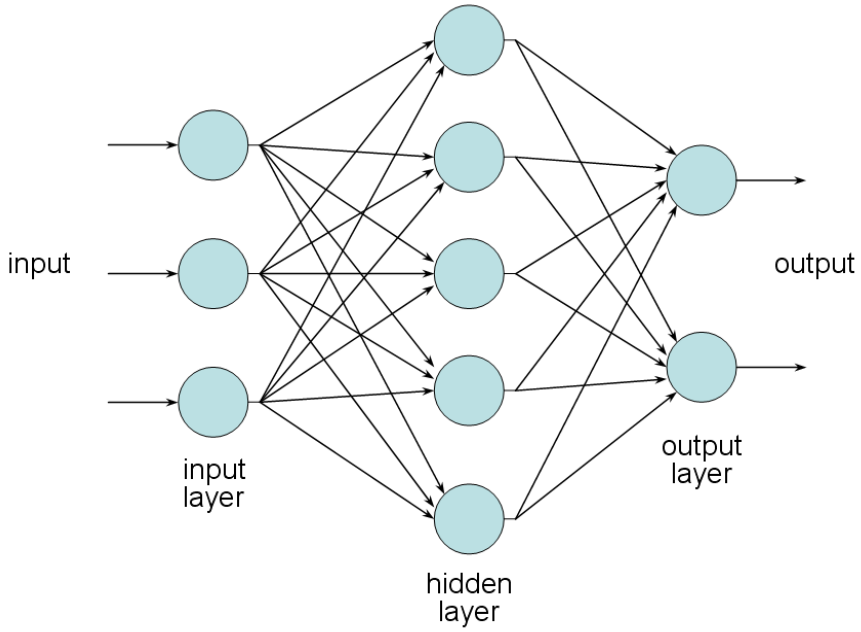


Figure 7. A multilayer perceptron

In supervised learning, we wish to infer the mapping between inputs and their associated outputs. The training modifies the biases and the weights of the connections as data flows through the network in order to minimise a cost function related to the mismatch between our mapping and the data. The application of the *gradient descent* technique to minimize the *mean-square error* in a MLP leads to the well-known **backpropagation** algorithm.

See [2], [3] or [4] for more details on neural networks.

5.1.3 Support Vector Machines

Support Vector Machine (SVM) is a supervised learning technique permitting to treat non-linear discrimination problems and to reformulate the classification problem as a quadratic optimization problem.

This technique is based in the notion of maximal margin. In the case of two linearly separable classes, the objective of the learning phase is to find a hyperplane that separates the examples of the two

classes and that maximises the distance to the nearest examples. These nearest examples are called *support vectors*. Generally, there are ambiguities between the classes and no hyperplane exists that perfectly separates the two classes. Thus, this condition is relaxed to tolerate that some examples are on the other side of the frontier. The violator examples introduce a cost in the objective function.

The Lagrangian formulation of the problem depends only on scalar products between vectors in the input space. Therefore, the **kernel trick** can be used in order to apply this algorithm to non-linear discrimination problems. The kernel trick implicitly transforms the input space into a high (possibly infinite) dimensional space, where a separating hyperplane is likely to be found.

For more details on SVMs, refer to [5].

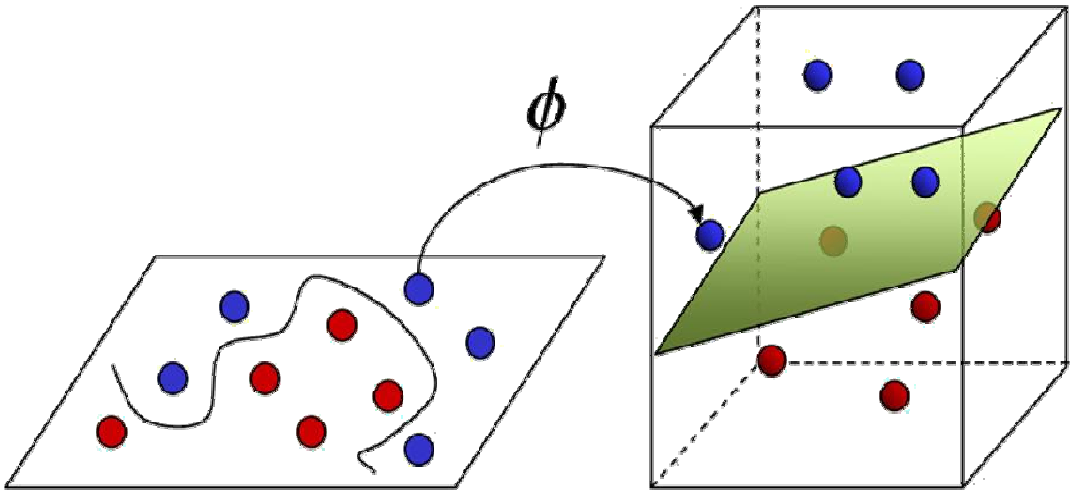


Figure 8. A representation of a SVM.[7]

5.1.4 Boosting

Boosting belongs to the family of *ensemble methods*, techniques that combine the predictions of a set of single learners trained individually in order to obtain an overall learner which performs better than any of the single learners.

Boosting is a machine learning method that builds a 'strong' classifier by linear combination of 'weak' classifiers. It can be viewed as a general method for improving the accuracy of any given base learning algorithm. AdaBoost is the main boosting algorithm. The weak classifiers are generated iteratively, by calling a base learning algorithm, and feeding it each time with different weightings over the training examples. At each iteration, the error rate of the weak classifiers is assessed, and the weight of the misclassified examples is increased. This has the effect of forcing the base learner to focus its attention on the "hardest" examples. The final classifier is obtained by a weighted majority vote of the weak classifiers, were the weight is higher if the error rate is lower. Provided that the base algorithm performs better than random, the final classifier can be (theoretically) arbitrarily accurate.

[7] Illustration from www.imtech.res.in/raghava/rbpred/svm.jpg

Figure 9 represents consecutive iterations of the AdaBoost algorithm. The separating lines represent the weak classifiers, and the size of a circle represents the weight of the example.

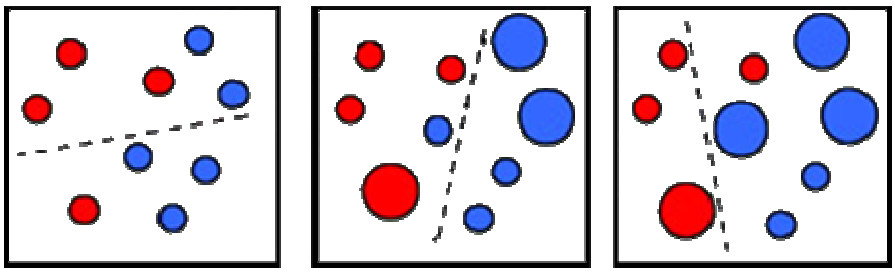


Figure 9. A representation of AdaBoost[8].

See [6] for more details on the AdaBoost algorithm. Moreover, [7] gives an interesting comparison between AdaBoost and SVM.

5.1.5 Hierarchical Temporal Memory

Hierarchical Temporal Memory (HTM) is a machine learning model developed by Jeff Hawkins and Dileep George of Numenta, Inc. inspired in some of the structural and algorithmic properties of the neocortex. HTM is based on the memory-prediction theory of brain function described by Jeff Hawkins in his book *On Intelligence*. [8]

The central concept of the memory-prediction framework is that bottom-up inputs are matched in a hierarchy of recognition, and evoke a series of top-down expectations. These expectations interact with the bottom-up signals to both analyse those inputs and generate predictions of subsequent expected inputs. [9]

The HTM paradigm has been implemented in a software API called The Numenta Platform for Intelligent Computing (NuPIC). A research release became available on March 2007. [11]

From [10] and [11], we can figure out how HTM works.

HTM networks are a multi-level hierarchy of nodes in a tree structure.

[8] Illustration from http://www.cc.gatech.edu/~kihwan23/imageCV/Final2005/images/bestrong.JPG

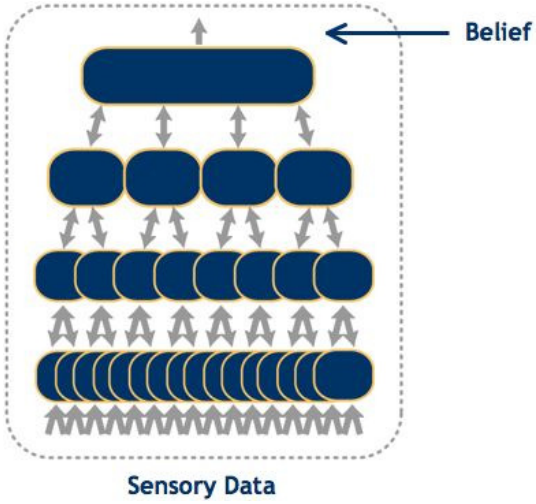


Figure 10. Representation of a HTM. [9]

The HTM documentation makes an extensive use of the concept of *cause*. It defines it as a persistent and repeating structure in the world: an object, a situation, an idea. The data depends on the underlying causes that generate it. These causes are what we want to identify. This is roughly equivalent to the notion of *category* in other machine learning contexts.

An HTM network receives sensory input and builds an internal representation of the causes in an **unsupervised** way, looking for correlations between data points that are adjacent in space and time. In other words, it assigns causes to recurring **spatial and temporal patterns** in its input.

The HTM Network as a whole performs the functions of identifying these recurring spatial and temporal patterns, and classifying its input data relative to known patterns that it has identified. Moreover, it can predict future events based on the input and the learned temporal sequences.

Each **node** in a HTM network implements a common learning and memory function: they all encapsulate the **same algorithm**. They store a number of recurring **spatial patterns** and **temporal sequences**[10]. At any time, a node looks at its input and assigns a probability that this input matches each element in a set of stored spatial patterns. Then the node takes this probability distribution and assigns a probability that the current input is part of the stored temporal sequences. The distribution over the set of sequences is the output of the node, and it is passed up the hierarchy.

As outputs are distributions over the stored patterns, the number of output variables is the number of stored patterns. This number is fixed. If the node is learning, then it might modify the set of stored spatial and temporal patterns to reflect the new input. This means that over the course of learning the meaning of the outputs gradually changes. In fact, the nodes do not have a separate **training** and **inference** mode. If learning is enabled, the set of stored patterns might change with the new input learning is off; else, the set

[9] From http://www.numenta.com/for-developers/education/htm_symb_rep.jpg
[10] This was true in the first implementation of HTM. Nowadays, the spatial and temporal functionalities have been separated into two different types of nodes. These two kinds of nodes must work together as a unit in order to achieve the original behaviour. Nevertheless, it remains true that roughly all nodes encapsulate the same algorithm: a node stores a quantization of its input space according to the recurring patterns seen, either from the spatial or temporal point of view, in an unsupervised way.

of stored patterns remains fixed. It is possible to disable learning when the training is finished, or to keep on learning while inferring.

A HTM network resembles a Bayesian network in the sense that the nodes constantly share information. HTMs exploit a variation of **belief propagation** to do inference, so that all nodes quickly reach a set of mutually consistent beliefs. The sensory data imposes a set of beliefs at the lowest level in an HTM hierarchy, and by the time the beliefs propagate to the highest level, each node in the system represents a belief that is consistent with the lower levels. Moreover, it has to be noted that the information does not only circulate **upwards**. As it has been said, part of the learning algorithm performed by each node is to store likely sequences of patterns. By combining memory of likely sequences with current input, each node has the ability to make predictions of what is likely to happen next. This prediction goes **downwards** in the hierarchy and acts as prior probability, meaning it biases the inferences in the lower node. This prediction helps the system understand noisy or missing data.

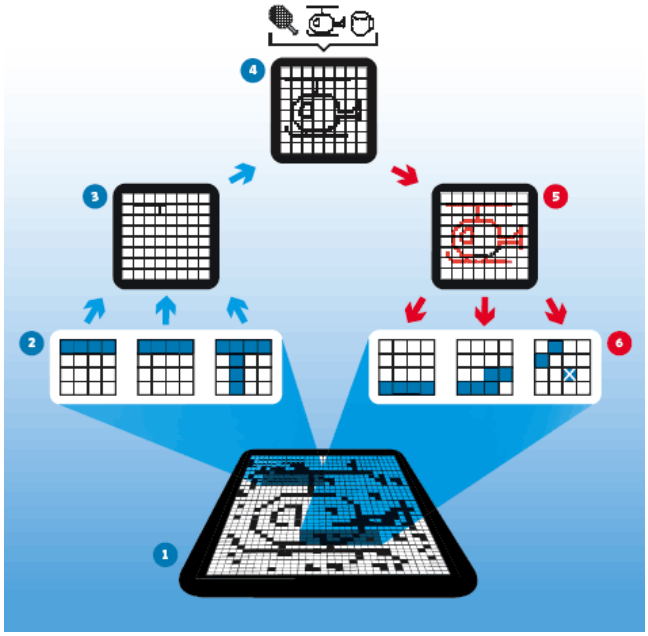


Figure 11. Illustration of the information flow in a HTM network.[11]
The learned sequences act as an a priori for the following inferences.

Each node in the lowest level of the hierarchy receives raw local sensory data. Nodes at higher levels do not receive sensory data, but rather the beliefs of their child nodes. For nodes at higher levels, spatial patterns consist of commonly occurring combinations of beliefs that their child nodes simultaneously report. The temporal sequences consist of recurring changes in those beliefs: they are sequences of sequences. Therefore, HTMs do not just exploit the **spatial hierarchy**. They take advantage of the **temporal hierarchy** of the input as well. As the information rises up the hierarchy, beliefs are formed at successively higher nodes, each representing causes over larger and larger spatial areas and longer and longer temporal periods.

[11] Image by Arno Ghelfi, from http://www.wired.com/wired/archive/15.03/images/MF_104_hawkins2_f.gif

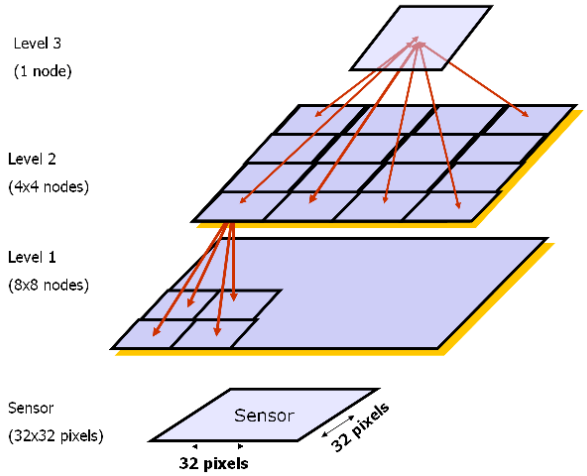


Figure 12. Example of the hierarchical arrangement of nodes.
The input can be a vector (1D sensor) or an image (2D sensor)

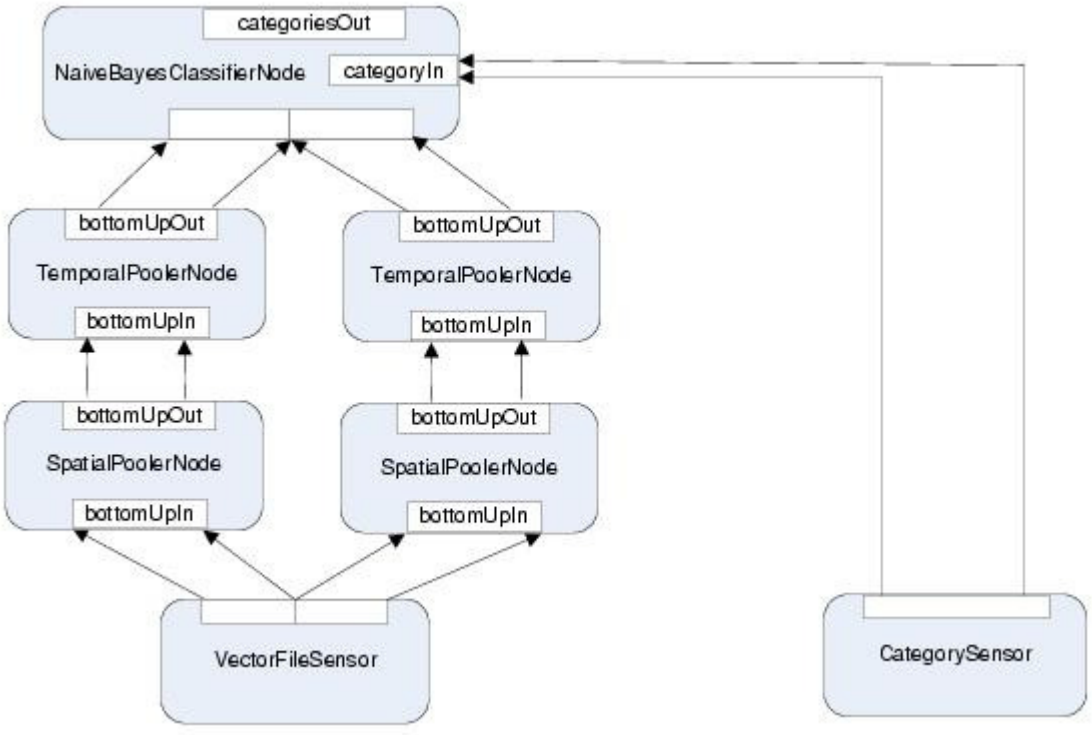


Figure 13. A very simple supervised network.

It has only a sensor and one level of spatial-temporal nodes. The top node can be any traditional classifier.

As it has been said, a HTM works in an unsupervised way. However, **supervision** can be introduced by imposing beliefs at the top level of the hierarchy during the training. The top node in the hierarchy works in a different way from the rest. It is a **classifier node**: it makes groups and maps those groups to categories. If category information is submitted during the training (supervised learning), the groups are

created according to the categories (see Figure 13). If not (unsupervised learning), the groups are created based on the characteristics of the input data. Even if Numenta has developed its own algorithms for the top node, the user can substitute it by any other kind of classifier. SVM, for example, could be a suitable candidate in the case of a supervised binary problem.

An important constraint in the use of HTMs is that even if we want to use the network for static inference, the system needs to be trained on time-varying inputs. This is because the learning of sequences is a key point in the HTM inference[12], notably in the downward flow of information[13].

All things considered, it can be drawn that a suitable application for HTMs would satisfy these conditions: the domain to model has an inherent hierarchy, the data have some spatial and temporal correlation, and the problem domain provides a large set of training examples organized in temporal sequences.

5.1.6 Preliminary comparison

From the theoretical foundations of the different algorithm, we can anticipate some of their properties.

An important practical aspect is the number and impact of the **parameters** of the model. The more parameters to tune, the more difficult will it be to find the optimal configuration, and this can lead to poor performances in general cases. In a multilayer perceptron, the parameters are the number of hidden units (assuming that there is only one hidden layer) and eventually the parameters of the backpropagation algorithm. The number of iterations is given by the training set size. In a SVM, the parameters are trade-off between regularisation, low training error and the kernel function. In AdaBoost, the parameters are the type of weak learner and the number of weak learners. In a HTM, each node has a considerable amount of spatial and temporal parameters to tune, besides the number of outputs. Moreover, the hierarchy structure must be chosen.

The best algorithm with regards to the parameter tuning is therefore AdaBoost, because the impact of its parameters is not critical. We can fix a weak learning algorithm (we use decision trees in this work) and give a sufficiently high number of iterations (AdaBoost hardly ever overfits) in order to achieve reasonable performances. On the contrary, HTM have too many parameters to tune, and they are critical. Thus, the optimization time therefore explodes, and there is a risk of poor performances if incompatible parameters are chosen. Moreover, the complexity of the representation of the problem must be tuned by the user, so there is a risk of assigning unnecessarily too many resources. SVM can perform poorly if the type of kernel function and the associated parameters do not suit the data. In a multilayer perceptron the optimal network structure is also problem dependent, and the number of possibilities explodes if we do not use heuristics.

However, an important characteristic of neural networks and HTMs is that they are **inherently multiclass**. This means that they could be used directly to solve our multiclass classification problem. At the same time, the multiclass nature is related to the difficulty to handle the notion of reject class.

[12] Recall that there is no separation between the training and the inference mode.
[13] This has changed lately, see footnote 10. It remains true as long as the network incorporates the temporal functionalities.

An advantage of SVM and AdaBoost is their ability to efficiently handle **high dimensionality** [7], as they both find linear classifiers for extremely high dimensional spaces. They address the problem of overfitting by maximising the margin (explicitly in the case of SVMs and implicitly in the case of AdaBoost, see [7]). The computational problem is addressed by kernels in the case of SVMs, as low dimensional calculations are equivalent to inner products in a high dimensional, and in the case of AdaBoost by the greedy search of those weak classifiers that have a non-negligible correlation with the labels of the examples.

On the contrary, high dimensional input vectors force the multilayer perceptron and the HTM to have a high number of inputs, and the resources used by the network must be high in order to handle such complexity.

An extra advantage of AdaBoost is its ability to handle **noisy components** of the input vectors and to identify outliers.

Besides, HTM is the only that automatically exploits complex **sequential redundancy**, which can be very useful in many applications.

5.1.7 Comparative study

5.1.7.1 Description of the comparison protocol

The classical and mostly used biclass problem is taken into consideration. We compare directly on the same databases the accuracy and efficiency of the four machine learning approaches. Each machine learning algorithm benefits from exactly the same inputs and is tested on exactly the same data as the other ones.

Several synthetic datasets have been generated. They simulate different discrimination problems with different kinds of difficulty: non-linearly separable classes, non-convexity, non-connectivity, ambiguity, high dimensionality, noisy components, under-represented learning sets. Our aim is to examine the behaviour of the different approaches under different conditions, to explore and confirm their theoretical advantages and disadvantages.

Moreover, there is an interest in the visualisation of these behaviours, so we have generated some datasets in a 2-dimensinonal input space corresponding to the coordinates in a plane, allowing the graphical representation of the model that each approach generates of each kind of problem. Real data has also been used.

All the algorithms have parameters (see previous section) which are optimised through a genetic algorithm. The multilayer perceptron always uses the same structure (in terms of nodes and layers) which is based on classic heuristics. It could have been optimised also, which would have been very tedious; therefore, the basic structure was always chosen, which justifies some very limited results (in other terms: an optimised neural network could obtain better results than those illustrated here, but it implies much more efforts of optimisation). In the SVM, the possible kernels were sigmoid, Gaussian and polynomial. In

AdaBoost the weak learners are decision trees. During optimisation, the maximal depth of the trees is selected.

HTM is a special case, because the required optimisation process is not compatible with the genetic algorithm. To begin with, we limit the HTM's structure to spatial nodes (as no temporal functionalities are supported by the rest of algorithms). Besides, when working with synthetic data (2 and 50-dimensional) a fixed hierarchy of nodes is chosen. All nodes have the same parameters, which are optimised through the genetic algorithm. In the case of real data, if the dimensionality is low enough, the architecture of the net has been chosen manually according to the hierarchy of the components in the feature vector, and the parameters of each node have been tuned by hand.

5.1.7.2 Results and analysis

A selection of the most relevant tests is presented here, and the conclusions of the study.

2-D data

The black-and-white background corresponds to the real data distribution. Green and red points are evaluated points, green being points where the predict function answered 1 (should be white in the background), red where it answered 0 (should be black in the background). Training, optimisation and evaluation were each done with 1000 points per class.

On the first and second databases (Figure 14 and Figure 15) which are very easy, the four algorithms are coarsely equivalent. HTM is however significantly slower in prediction time.

On the third one (Figure 16) boosting is clearly the best algorithm. It profits from its very limited number of parameters, which helps it to achieve more easily the optimal parameterisation. HTM and SVM have honourable results, but SVM performs better with half the computational cost.

The following test (Figure 17) illustrates how the four machine learning algorithms deal with complex 2D-data where some confusions exist (grey values indicate the probability of belonging in each class). In this case, boosting provides decent results. SVM and HTM have comparable predictions and comparable results, but SVM are five times faster so that they are the best algorithm on this database (since boosting is twice faster, if computational cost is an issue then boosting should be chosen).

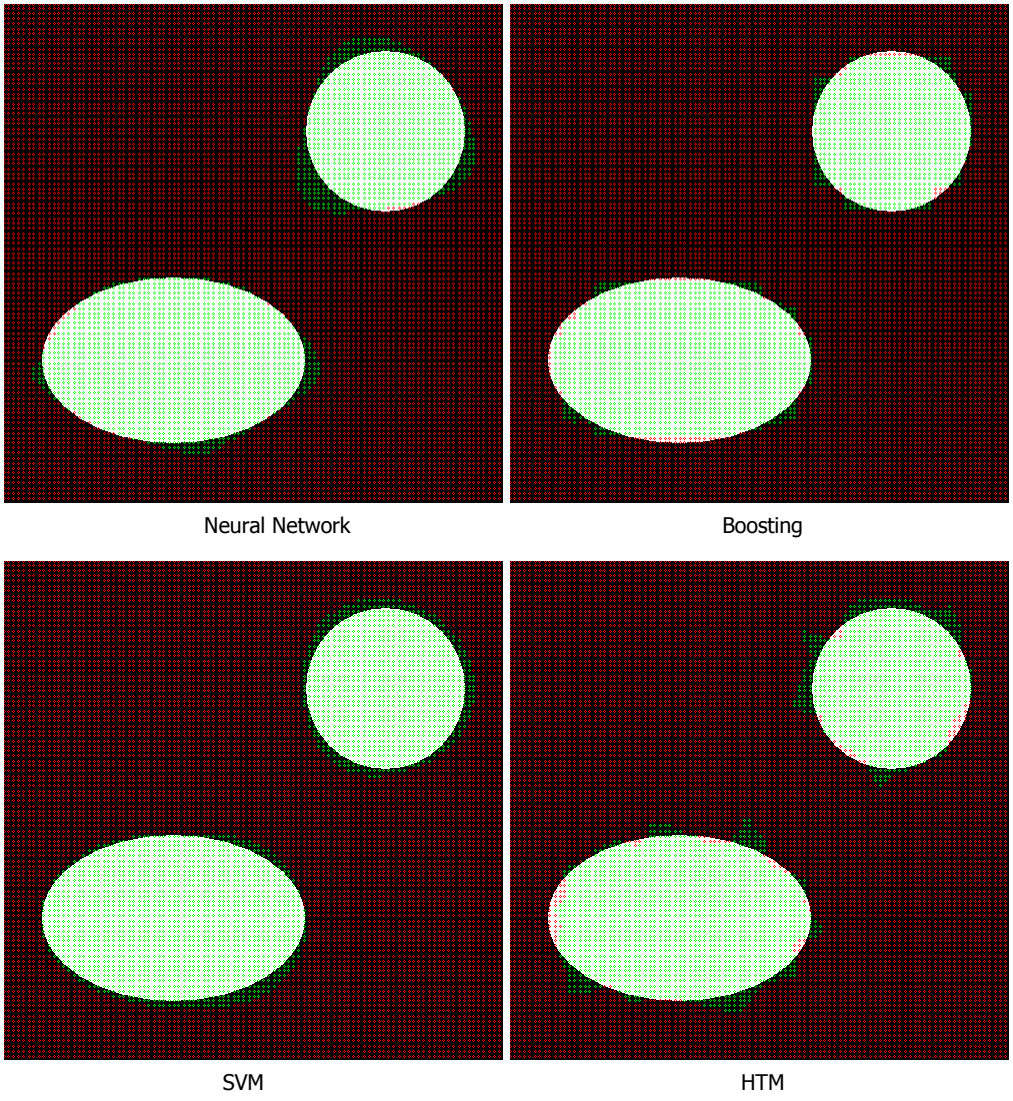


Figure 14. Results on non-connected data.

	Neural Network	Boosting	SVM	HTM
Probability of false alarm	3.8%	1.9%	4.5%	3.7%
Probability of detection	99.2%	97.2%	100%	96.4%
Precision	87.9%	93.2%	86.1%	87.7%
Prediction time (in seconds)	4.84e-5	6.24e-5	8.75e-5	43.1e-5

Table 2. Results on non-connected data.

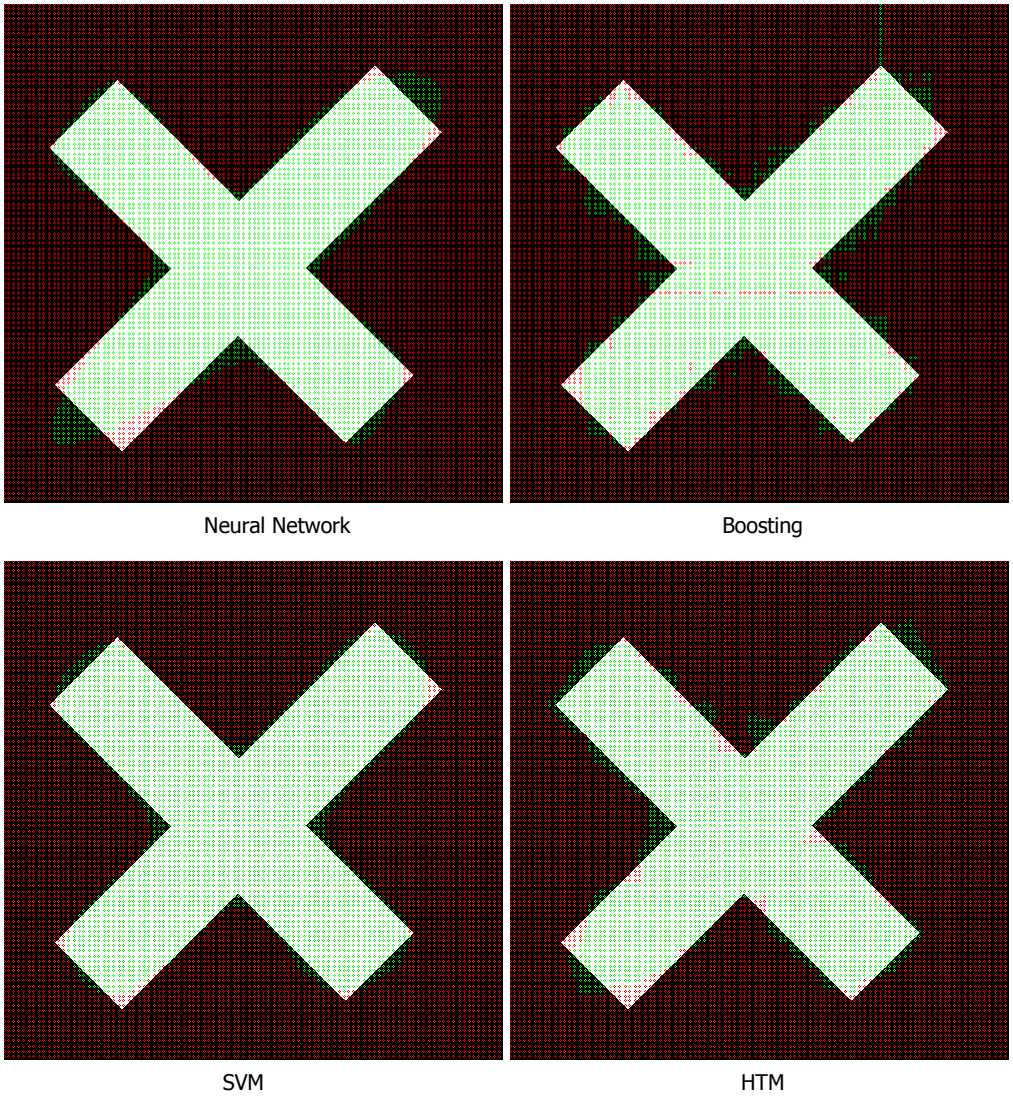
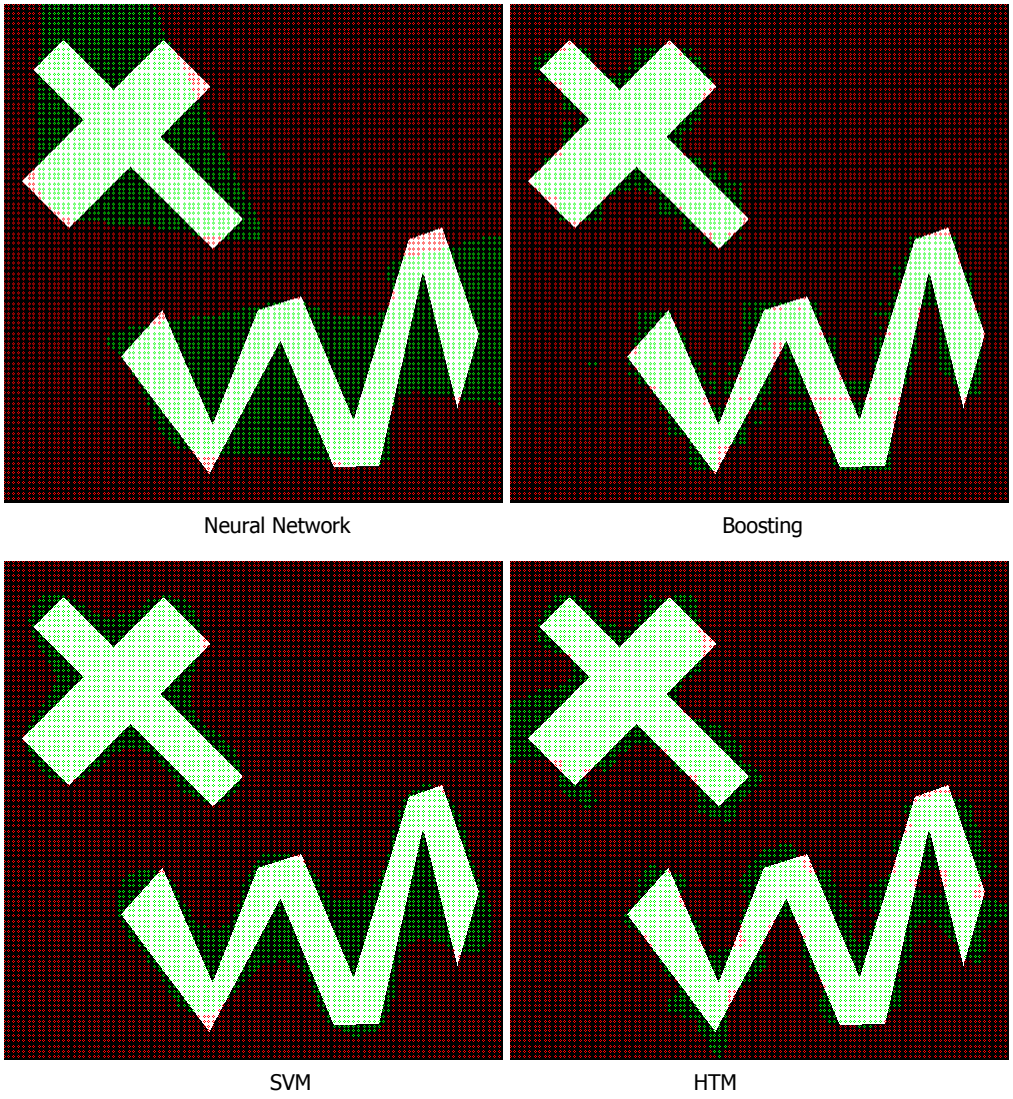


Figure 15. Results on non-convex data.

	Neural Network	Boosting	SVM	HTM
Probability of false alarm	4.6%	5.5%	3.7%	5.3%
Probability of detection	96.7%	95.1%	98.3%	96.8%
Precision	89.9%	87.9%	91.9%	88.3%
Prediction time (in seconds)	4.68e-5	8.13e-5	8.12e-5	27.5e-5

Table 3. Results on non-convex data.



Neural Network | Boosting

SVM | HTM

Figure 16. Results on non-connected and non-convex data.

	Neural Network	Boosting	SVM	HTM
Probability of false alarm	22.4%	6.9%	12.0%	11.2%
Probability of detection	95.3%	95.7%	99.0%	96.4%
Precision	50.6%	76.9%	66.4%	67.4%
Prediction time (in seconds)	3.75e-5	8.45e-5	10.8e-5	28.0e-5

Table 4. Results on non-connected and non-convex data.

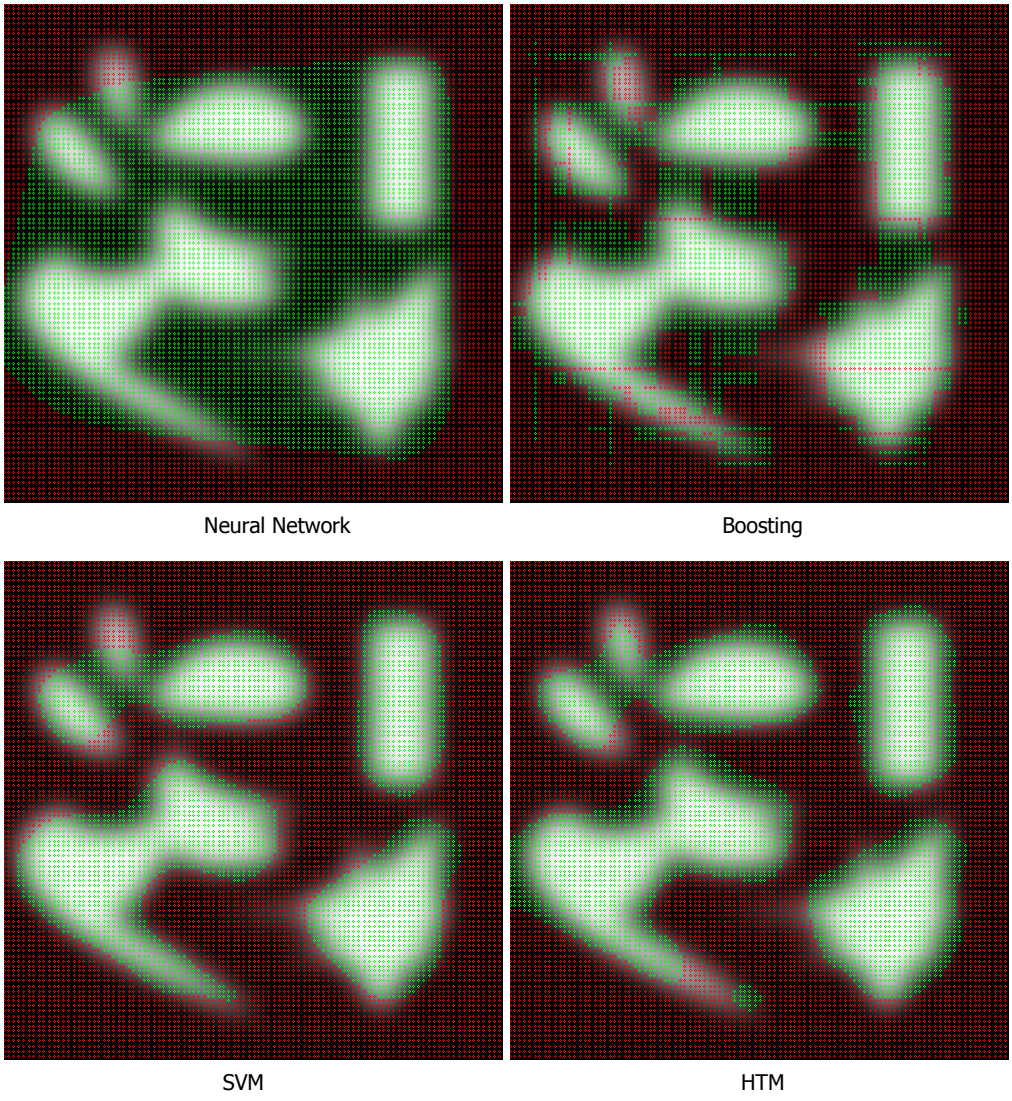


Figure 17. Results on complex 2D-data with some confusions.

	Neural Network	Boosting	SVM	HTM
Probability of false alarm	51.3%	27.1%	15.3%	20.8%
Probability of detection	95.2%	82.5%	66.2%	60.1%
Precision	40.3%	52.5%	66.2%	60.1%
Prediction time (in seconds)	3.76e-5	5.64e-5	12.3e-5	58.8e-5

Table 5. Results on complex 2D-data with some confusions.

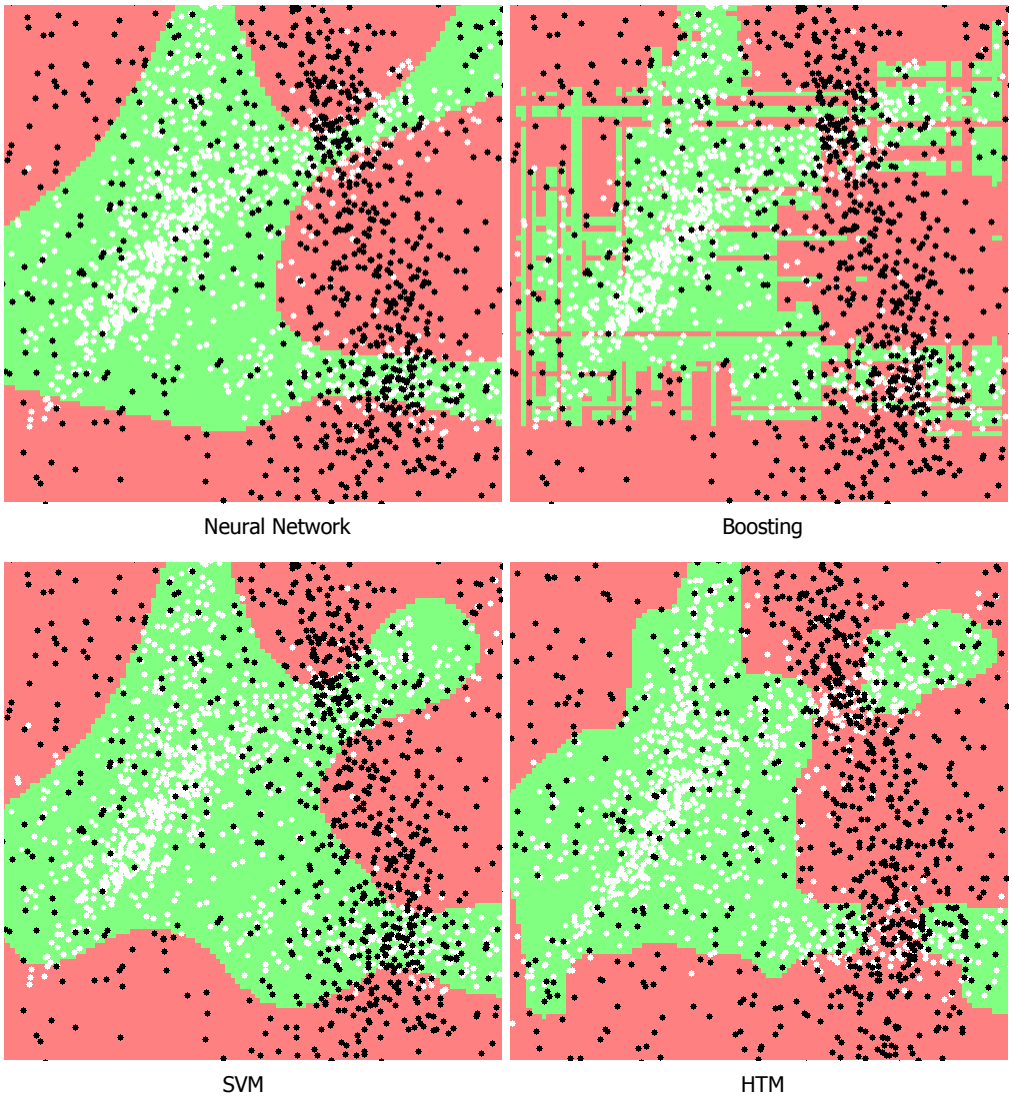


Figure 18. Results on complex 2D-data with strong confusions.

	Neural Network	Boosting	SVM	HTM
Probability of false alarm	27.1%	24.4%	37.5%	21.1%
Probability of detection	84.4%	76.9%	92.0%	82.0%
Precision	75.7%	75.9%	72.0%	79.5%
Prediction time (in seconds)	3.90e-5	7.85e-5	11.7e-5	16.4e-5

Table 6. Results on complex 2D-data with strong confusions.

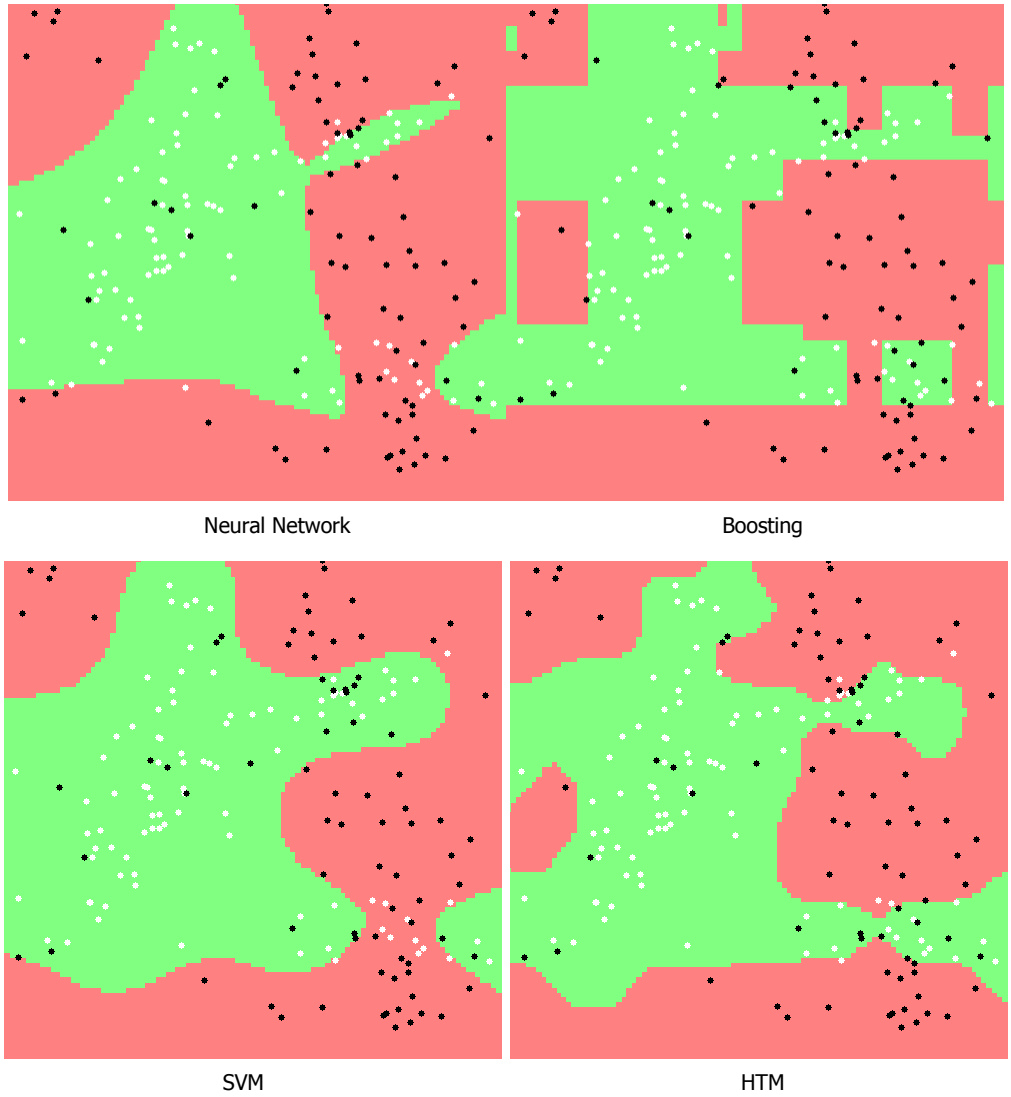


Figure 19. Results on complex 2D-data with strong confusions, with few learning data

	Neural Network	Boosting	SVM	HTM
Probability of false alarm	21.0%	28.0%	26.0%	22.0%
Probability of detection	70.0%	70.0%	77.0%	79.0%
Precision	76.9%	71.4%	74.7%	78.2%
Prediction time (in seconds)	0 (below 1e-5)	0 (below 1e-5)	7.5e-5	N/A

Table 7. Results on complex 2D-data with strong confusions, with few learning data.

The databases shown in Figure 18 and Figure 19 were generated with Gaussian mixtures, resulting in non-connected, non-convex and mixed data (strong confusion). These pictures must be read as follows: white dots are learning base positive samples (class "1"), black dots are learning base negative samples (class "0"), green pixels are prediction values at 1 and red pixels are prediction values at 0. Ideally, green pixels should be below white dots and red pixels below black dots.

In the first case, all algorithms have decent results. Neural networks and boosting have interesting prediction times (boosting is twice faster than HTM). SVM and HTM have very good prediction results.

In the second case, the training and evaluation data has been generated as in the previous one, but with only 50 elements per class for training. This database illustrates the same degree of complexity as many real databases, for which training data is insufficient and where the intrinsic complexity introduces strong confusions. The four machine learning algorithms deal pretty well with this concrete example of reduced training data.

N-D data

Real data mostly have lots of dimensions, resulting in far more complex predicting models as those illustrated before in two dimensions.

Therefore, training and evaluation data were generated in a 50 dimensions space, with distributions consisting of mixtures of Gaussians.

	Neural Network	Boosting	SVM	HTM
Probability of false alarm	27.0%	10.7%	9.4%	63.4%
Probability of detection	85.0%	91.5%	93.1%	97.5%
Precision	75.9%	89.5%	90.8%	60.5%
Prediction time (in seconds)	0 (below 1e-5)	0 (below 1e-5)	7.5e-5	N/A

Table 8. Results on synthetic 50-dimensional data.

Best results are achieved with the SVM. However, they lack in computational efficiency (boosting is 6 times faster and even HTM is faster in this test). Boosting is an appropriate trade-off between computational cost and classification power. We observe that the fixed architecture chosen for HTM is insufficient for modelling the problem, as can be deduced from the high false alarm rate.

Real data

The data used in the following tests corresponds to real video-surveillance sequences (see section §6.1.1). MPEG-7 (texture, colours, shape) and geometric descriptors were extracted resulting in input vectors of several hundreds and 7 values respectively. Data is complex since the resolution of the vehicles objects is very poor and compression artefacts are important. Here we show the results obtained for the classifier of cars and the classifier of trucks.

	Boosting	SVM	HTM
Probability of	1.7%	1.9%	4.2%

false alarm			
Probability of detection	94.4%	72.7%	91.1%
Precision	98.9%	98.4%	97.2%
Prediction time (in seconds)	1.8e-3	0.3e-3	3.6e-3

Table 9. Results on real data (MPEG and geometric descriptors). Car detector.

	Boosting	SVM	HTM
Probability of false alarm	3.1%	1.7%	0%
Probability of detection	71.1%	59.0%	0%
Precision	78.8%	85.0%	0%
Prediction time (in seconds)	1.8e-3	0.29e-3	N/A

Table 10 Results on real data (MPEG and geometric descriptors). Truck detector.

	Boosting	SVM	HTM
Probability of false alarm	1.7%	1.5%	3.7%
Probability of detection	88.4%	74.8%	96.1%
Precision	98.8%	98.8%	97.6%
Prediction time (in seconds)	4.9e-5	12.6e-5	30e-5

Table 11. Results on real data (only geometric descriptors). Car detector.

Globally, boosting achieves the best results. It is beaten by HTM on the last test (with only geometric descriptors), but is still 6 times faster to provide predictions. SVM have very good prediction times (less support vectors were chosen, resulting in lower quantitative results for classification).

HTM have interesting results on cars (it can deal properly with noisy data, which is the reason why SVM has limited performances), even the best ones on the geometric descriptors only. However, one must know that the HTM has been precisely defined (in terms of structure) according to a priori knowledge concerning the descriptors contained in the input vectors, which is an information that wasn't provided to the other algorithms.

The interpretation is that real data is noisy data. It looks as if geometric descriptors have some very discriminative values and some noisy values, whereas MPEG-7 descriptors are globally noisier. Boosting achieves the best performances with all the available data, since it can remove by its own useless and noisy values within the input vectors. The HTM and the SVM suffer from noisy data, but HTM is more robust to it.

It is remarkable that HTM could not achieve a single classifier on the truck data. This shows to which extent they are difficult to parameterise.

Interpretation

All these benchmarks tend to explain why Boosting and SVM are currently used in almost any classification problem. Generally, SVM obtains better results when data is not noisy, when boosting is usually characterized by lower computational costs and performs better in presence of noisy descriptors.

These tests have allowed observing some other advantages and drawbacks of HTM, compared to these state-of-the-art machine learning algorithms. The advantages are that it can be defined so as to profit from the structure of the input data; it performs good results, most of the time with scores equivalent or close to those of Boosting or SVM. Besides, it is still under development, so it is still promising. The drawbacks are that it needs a very complex and tedious preliminary step of definition, so as to select the structure and to fix all the available parameters to their adequate values (or poor results – sometimes even no result at all – are achieved); for classification scores similar to those of Boosting and SVM, it needs more computational cost (from two to ten times more); it is a proprietary technology which is more complex than Boosting and SVM. The high prediction time can also be a consequence of the fact that the complexity of the representation of the problem must be tuned by the user, and we run the risk of assigning too many resources.

In our work, we have chosen SVM as the binary classifier. We also use boosting in certain steps of the system optimisation, when binary classifiers must be generated quickly and there is no time for optimising its parameters.

5.2 Calibration

5.2.1 Introduction

As explained in previous sections, many binary learning algorithms map the input to a score whose sign indicates if the input has been classified as positive or negative and whose magnitude *can be taken as* a measure of confidence in the prediction, [14] [13]. This means that these scores can be used to rank the examples in the test set from the most probable member to the least probable member of the positive class.

However, these scores depend on each classifier and they only can rank the examples with regards to this classifier. Even after re-scaling them, they do not represent the probability that the input is a member of the class of interest, and therefore the direct comparison of scores from different classifiers does not make sense. This is especially true with SVM, where the score is the distance to the learned hyperplane and where the measure of distance varies with the choice of the kernel function. Probability estimates are needed when the classification output has to be combined with other sources of information for decision-making.

Consider the example in Figure 20. It shows two classes (black and white) in a 2-dimensional space, and three possible classifiers depending on the adopted optimization criterion: a penalty for false alarms, a penalty for non-detections or an equal penalty for both types of errors. Each classifier models the problem differently, and we have the output score as the only insight to the model. We can see that examples far from the separating hyperplane are more likely to be classified correctly – this is what we mean when we

say that the score ranks the examples. But the score itself is not directly interpretable as the chance of membership of the class. For instance, if we consider two symmetric scores coming from one classifier, we can see that they do not necessarily correspond to the same probability of error. If score magnitudes cannot be thought of as probabilities within the context of one single classifier, even less are they comparable among several classifiers.

We can think of a score as the projection of the example into a 1-dimensional space, which *presumably* corresponds to the direction that best discriminates between the two classes. Calibration attempts to regain some of the lost information in this projection.

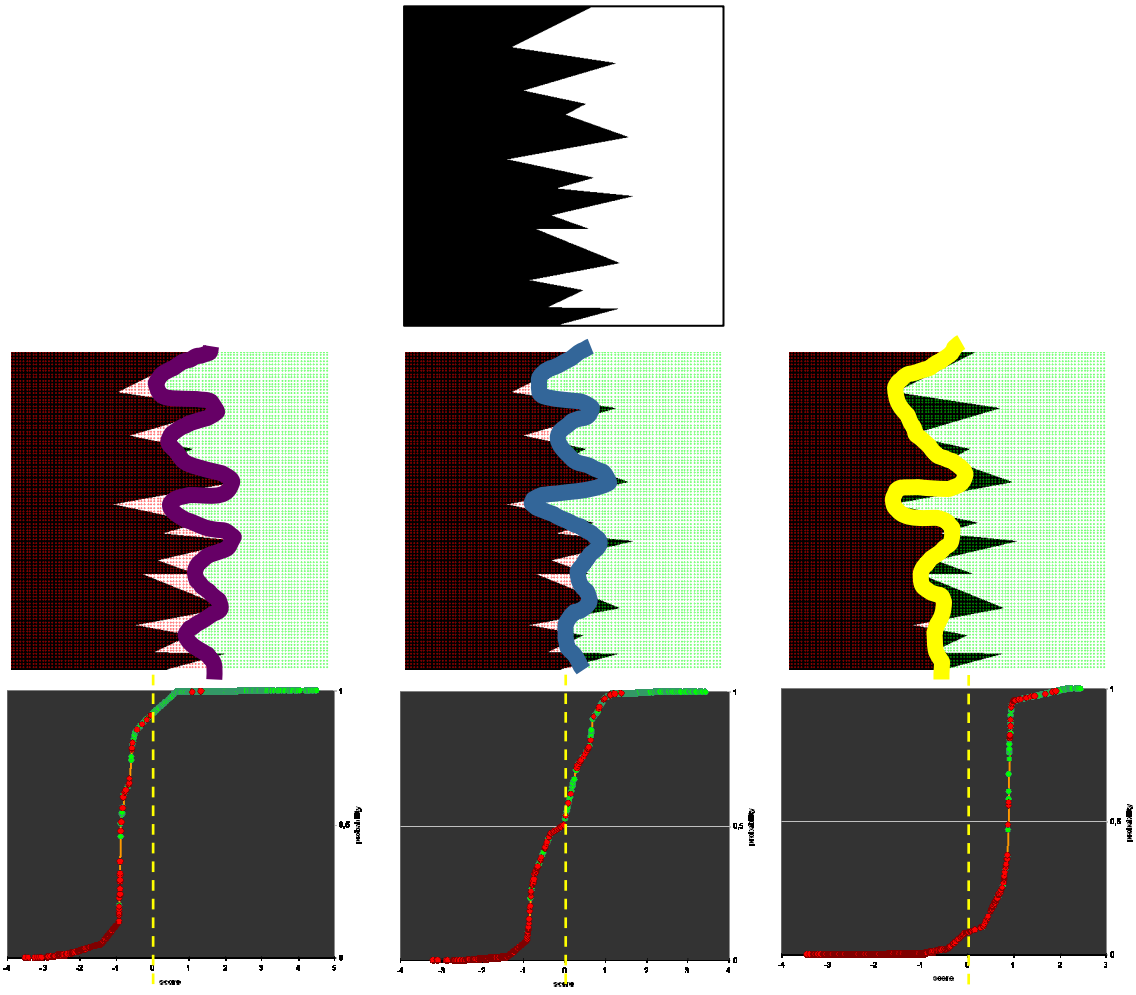


Figure 20. Calibration

The reliability of the answer depends on the classifier and on the answer.

During the training of the whole system, calibration consists in finding the mapping from raw scores to accurate probability estimations. When inferring the class of a new example, calibration is the application of this mapping to the obtained raw score.

5.2.2 Related work

[13] recalls the definition of a calibrated classifier:

> *Assume that we have a classifier that for each example* x *outputs a score* s(x) *between 0 and 1. This classifier is said to be well-calibrated if the empirical class membership probability* P(c|s(x)=s) *converges to the score value* s(x)=s*, as the number of examples classified goes to infinity.*

The straightforward way of calibrating consists in dividing the possible scores into intervals and calculating the empirical probability in each of them. In the simplest version, the size of the interval would be fixed. However, this does not guarantee a sufficient number of examples in each interval for an accurate estimate. The choice of the intervals size is a trade-off between a sufficiently fine representation of the mapping function and a sufficiently accurate estimate in each interval.

[13] makes a review of other existing methods for calibrating two-class classifiers: Platt's method and binning.

The calibration proposed by Platt (1999) is a parametric approach. It consists in fitting a sigmoid function, so as to minimize the negative log-likelihood of the data. This method is motivated by the fact that imposing a parametric shape is a possible way of regularizing and avoiding overfitting, and the relation between SVM scores and empirical probabilities seems to be sigmoidal for many datasets. The problem is that this does not necessarily hold for all datasets and all learning algorithms: the shape of the function is unknown.

Binning is a non-parametric method. The examples are sorted according to their score and divided into groups with the same number of examples (bins). This induces a division of the score range into intervals and an associated probability estimate in each interval. On the contrary to the straightforward solution, this method imposes a number of examples in each interval. However, this approach does not maintain the idea of local averaging to obtain the estimates. Examples with very different scores may be grouped. If examples that clearly should have different probability estimates are averaged together, the method will fail to give accurate estimates. Again, the optimal size of the bins is unknown.

[13] proposes a new method based on Isotonic Regression (Robertson et al., 1988). This calibration is non-parametric, and imposes that the mapping has to be non-decreasing. This is an interesting way of regularizing, because the scores are supposed to be a measure of confidence in the prediction.

As for which learning methods need calibration, [13] remarks that Naive Bayes and SVM scores are not well-calibrated. [12] uses Platt's method and isotonic regression to calibrate several learning methods. Empirically, it is shown that the best calibration method depends on the algorithm and the dataset. In general, boosted trees and SVM perform better in their experiments when calibrated with Platt's method. They also claim that neural nets are so well calibrated from the beginning that they are even slightly hurt by calibration.

In this work, Platt's calibration is rejected because it imposes a shape of the function, which contradicts the objective of designing a completely general system. Among the non-parametric methods,

isotonic regression is preferred, because no arbitrary choice of the size of the bins or the intervals has to be done.

We have noticed that binning and isotonic regression are in fact particular cases of the straightforward solution, in which the size of the intervals is variable and induced by the distribution of the data. Thus, they can be analyzed in a common framework of probability distribution estimation. Moreover, we will also address the issue of computing the reliability of these estimates.

5.2.3 Formal framework

5.2.3.1 Probability distribution estimation

The starting point is an already trained binary classifier. The classifier represents one possible model of the binary problem. It assigns a score to each input example. We want to translate this score into the probability of belonging to the positive class, given this particular model.

Consider the random variable formed by the pair class-score $X = (C, S)$, which takes values in the space $T = \{0,1\} \times D$ with $D \subseteq \mathbf{R}$. This random variable follows a probability distribution $P(X) = P(C, S)$. To infer the distribution of the random variable X, we have a **random sample** of independent observations $(X_1, X_2, ..., X_n)$. Each observation $X_i = (C_i, S_i)$ corresponds to an input example from the validation base.

In particular, we are interested in the conditional probability $P(C = 1 \mid S = s) \equiv r(s)$, which verifies the following relations (Bayes):

$$P(C = 1 \mid S = s) = \frac{f_{CS}(1, s)}{f_S(s)} = \frac{f_S(s \mid C = 1)\, P(C = 1)}{f_S(s)}$$

where f_{CS} and f_S denote density functions.

In the following paragraphs, we are going to infer an estimator for this probability, based on the course material about empirical distributions in [15]. Refer to this source for more details or general insights into the basic theory and applications of probability, statistics, and certain special models and random processes.

Empirical distribution

The **distribution** of X is the probability measure in T given by $P(A) = P(X \in A)$ for $A \subseteq T$. The **relative frequency** of $A \subseteq T$ corresponding to a random sample of size n is

$$P_n(A) = \frac{\#\{i \in \{1,2,...,n\} : X_i = (C_i, S_i) \in A\}}{n} = \frac{N_n(X \in A)}{n}$$

were $N_n(B)$ designates the number of times that the event B has happened (the frequency of B) in n observations. The relative frequency P_n satisfies the axioms of a probability measure. In fact, it gives the **empirical distribution** of X based on the random sample. It is a discrete distribution that places probability mass $1/n$ at each X_i. The empirical distribution has the following properties:

$$E[P_n(A)] = P(A)$$
$$\text{var}[P_n(A)] = P(A)[1 - P(A)]$$
$$P_n(A) \to P(A) \text{ as } n \to \infty \quad (\text{with probabilty} 1)$$

Empirical density function

In our case, we are working simultaneously with C – a random variable with a discrete distribution on the countable set $\{0,1\}$ –, and S – a random variable with a continuous distribution on a subset $D \subseteq \mathbf{R}$. Their density functions are estimated differently.

In the case of the **discrete variable** C, let f_C denote its **density function** such that

$$f_C(c) = P(C = c) \quad \text{for } c \in \{0,1\}.$$

The **empirical density function** corresponds to the relative frequency function defined before:

$$f_{C_n}(c) = P_n(C = c) = \frac{\#\{i \in \{1,2,...,n\} : C_i = c\}}{n} = \frac{N_n(C = c)}{n} \quad \text{for } c \in \{0,1\}$$

Thus, it also verifies:

$$E[f_{C_n}(c)] = f_C(c)$$
$$\text{var}[f_{C_n}(c)] = f_C(c)[1 - f_C(c)]$$
$$f_{C_n}(c) \to f_C(c) \text{ as } n \to \infty$$

The case of the **continuous variable** S is somehow more complicated. Let f_S denote the **density function** of S. Technically, f_S is the density with respect to m_1, the length measure on $\mathbf{R}$. By definition

$$P(S \in A) = \int_A f_S(s)ds \quad \text{for } A \subseteq D$$

In order to define its empirical density function, a partition of D into a countable number of intervals $\{D_l\}$ must first be defined. As before, the empirical probability of D_l can be defined by:

$$P_n(S \in D_l) = \frac{\#\{i \in \{1,2,...,n\} : S_i \in D_l\}}{n} = \frac{N_n(S \in D_l)}{n}$$

The empirical density function is then defined as:

$$f_{S_n}(s) = \frac{P_n(S \in D_l)}{m_1(D_l)} \quad \text{for s} \in D_l$$

It corresponds to the distribution for which $P_n(S \in D_l)$ is uniformly distributed over D_l. This empirical density function **depends on the partition**, so we cannot establish the same relationships between $f_{S_n}(s)$ and $f_S(s)$ as in the previous case. But by the very definition of density, if the partition is sufficiently fine (so that D_l is small for each l) and if n is sufficiently large, then by the law of large numbers, $f_{S_n}(s) \sim f_S(s)$ for $s \in D_l$.

The obtained estimator

Putting all these results together, we obtain an estimator for our probability of interest for each $s \in D_l$ that fits perfectly with our intuition:

$$\tilde{r}(s) \equiv \tilde{P}(C=1 \mid S=s) = \frac{f_{S_n}(s \mid C=1)\,P_n(C=1)}{f_{S_n}(s)} = \frac{N_n(C=1,\ S \in D_l)}{N_n(S \in D_l)}$$

or equivalently:

$$\tilde{r}(s) = \tilde{P}(C=1 \mid S \in D_l) = \frac{P_n(C=1,\ S \in D_l)}{P_n(S \in D_l)}$$

Obviously, the properties of $\tilde{r}(s)$ as an estimator of $r(s)$ depend on the partition of the scores into intervals. To obtain accurate estimates, the partition has to be sufficiently fine and the number of observations sufficiently high.

Remark that both binning and isotonic calibration are particular cases of this framework. What differentiates one method from the other is the way in which the partition of scores is made.

5.2.3.2 Variability of the estimation

The probability estimates $\tilde{r}(s)$ issued from calibration are not used isolated. As we will see in section §5.4.2, these estimates are combined in a way such that we are not only interested in the obtained estimate but also in its reliability, so that we can weight differently the contribution of each source of information.

As a first approach, we have **empirically** studied the variability of the obtained mapping with regards to the validation set. In particular, we have split the validation set in two subsets, and we have combined the two obtained calibrations in the following way:

- the resulting mapping is the average of both calibrations;
- at each score, the two calibrations and the average are used to output a variance.

In early phases of the design, we have empirically verified that the use of this variance as a measure of confidence in the estimate improves the results of the posterior combination.

In Figure 21 we show a result on real data. Pink and blue curves have been generated with the two halves of the validation data. The yellow curve has been generated with all the available data.

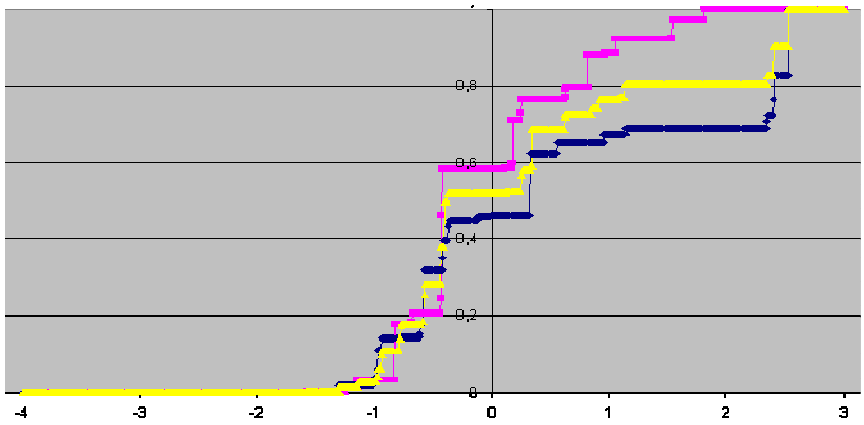


Figure 21. Empirical test on calibration variability

Other variants of this approach would be inspired by cross-validation, in which the original validation set is split into *t* subsets, and *t-1* are used each time to obtain *t* different calibrations. On the one hand, each of the calibrations would be presumably more accurate, as a higher number of examples are used. On the other hand, the subsets are not independent, so the variability of the calibration with regards to the set would be less evident.

As a second approach, we have tried to analyse this problem from the **theoretical** point of view, in order to assess the reliability of the estimates from the statistical properties of the data, without having to calibrate several times with random subsets of the available data.

We have seen that the obtained estimate is the same for all scores within an interval D_l. With our estimate, we try to calculate $P(C=1 \mid S=s)$ averaged in $s \in D_l$. To avoid confusion, we highlight the two different sources of error: the error that we make in the estimation of $P(C=1 \mid S \in D_l)$, and the difference between $P(C=1 \mid S \in D_l)$ and each $P(C=1 \mid S=s)$. The first one decreases with the number of examples in the interval, but the second one decreases when the partition of scores is fine. Thus, there is a trade-off between the two, and they are so coupled that it is very difficult to analyse them separately. For example, intuitively we could think that the variance of the estimate in an interval is inversely proportional to the number of examples in the interval. This is not appropriate because the choice of the intervals may follow a regularisation criterion[14] that in practice makes the estimator much less variable. This kind of global aspects should also be taken into account.

As it has been said, the dependence with the partition prevents from deducing the properties of the estimator. However, we will try to give insights on this question.

The following approach lies on the concept of the "sufficiently fine" partition of scores into subsets and on the computation of a **confidence interval** for the estimate $\tilde{r}(s)$ in each subset.

[14] This is the case of the calibration method that we have adopted : isotonic regression.

We define the indicator variable I so that it takes the value 1 when $C = 1$ and 0 otherwise. If we restrict it to a subset of examples whose scores fall in A, this indicator variable has the Bernoulli distribution with parameter $P(C = 1 \mid S \in A)$. For brevity, we note this probability $p(A)$:

$$E[I] = P(C = 1 \mid S \in A) \equiv p(A)$$
$$\text{var}[\text{I}] = p(A)[1 - p(A)]$$

Recall that at each D_l, $r(s)$ is estimated through the estimation of $p(D_j)$. The obtained $\tilde{r}(s)$ is the mean value of I in D_l, for $s \in D_l$.

We define D_l to be sufficiently fine if $P(C = 1 \mid S \in D_l)$ is representative of $P(C = 1 \mid S = s)$ for all $s \in D_l$. That is to say that $p(A) = P(D_l)$ for all $A \subseteq D_l$.

First, we establish the hypothesis that D_l is sufficiently fine. We then analyse some relevant subsets of D_l and try to find if there is sufficient statistical evidence to reject the hypothesis. In that case, we could assign to the violator subsets a low reliability in the estimate $\tilde{r}(s)$.

We do not know the exact values of $p(A)$, but we can compute estimates $\tilde{p}(A)$ as the mean of I in A:

$$\tilde{p}(A) = M = \frac{\sum_{j=1}^{m} I_j}{m}$$

where $(I_1,...,I_j,...,I_m)$ corresponds to the subset of examples whose scores fall in A. Moreover, supposing that $(I_1,...,I_j,...,I_m)$ is a random sample[15] from the Bernoulli distribution with unknown parameter $p(A)$, we can compute approximate confidence intervals for these estimates $p(A)$ using the properties of the Bernoulli distribution (see [15]). If we use M as an estimator for $p(A)$, we can approximate the variance of this estimator as $\frac{M(1-M)}{m}$ and, by the central limit theorem[16], build a random variable with approximately a standard normal distribution[17]:

$$Z = \frac{M - p(A)}{\sqrt{\frac{M(1-M)}{m}}}$$

and hence an approximate pivot variable for $p(A)$. An approximate $1 - r$ level confidence interval (a), confidence upper bound (b), and confidence lower bound (c) for $p(A)$ are thus given as:

---

[15] $I_1,...,I_j,...,I_m$ independent and identically distributed.

[16] Valid if m is large. As an alternative we could use a variable with binomial distribution as pivot variable. The inconvenient is that the quantiles depend on m, so they should have to be extracted from a table containing all possible values.

[17] The distribution of Z is closest to normal when $p(A)$ is near to ½, and farthest when $p(A)$ is near to 0 or 1.

$$a) \quad \left[M - z_{1-r/2}\sqrt{\frac{M(M-1)}{m}} \;,\quad M + z_{1-r/2}\sqrt{\frac{M(M-1)}{m}} \right]$$

$$b) \quad M + z_{1-r}\sqrt{\frac{M(M-1)}{m}}$$

$$c) \quad M - z_{1-r}\sqrt{\frac{M(M-1)}{m}}$$

where z_r is the quantile of order r for the standard normal distribution.

We can then use these confidence intervals to test the hypothesis and assign different reliability values to different subsets of D_l, according to the statistical properties of the data. See section §5.2.4 for more details on a possible implementation of these tests.

5.2.3.3 Weights trick

In practice, to calibrate our classifiers we have at our disposal a database whose distribution among classes may be very different from the distribution in the system's operational environment. This happens for example when one of the classes has considerable variability and the others much less. For this variability to be well represented in the database, there must be a great number of examples of the first class, and a low number of examples of the second class is enough, independently of the real proportion of classes in the operational context.

In this case, the observation do not follow the probability law that we want to estimate, and therefore the estimation with the previous approach will not be pertinent. In an attempt to correct this database fault, we add some extra knowledge about the operational distribution of classes and use it to weight the examples.

In the following paragraphs, we give a possible interpretation of this correcting mechanism, and the resulting expression for the probability estimation.

We can consider that, instead of a sample of the real distribution P, we have a sample of another distribution Q. We suppose that we know the real distribution between classes: $P(C=1)$ and $P(C=0) = 1 - P(C=1)$. Besides, we make the hypothesis[18] that the law Q has the same behaviour as P with regards to the scores: if we compare Q and P, we find the frequency of each class increased or decreased uniformly along the scores, i.e. multiplied by a factor independent of the score. Therefore, we have the following relations:

$$\begin{cases} P(C=1 \mid s) = k_1 Q(C=1 \mid s) \\ P(C=0 \mid s) = k_0 Q(C=0 \mid s) \end{cases} \quad \forall s \quad \Rightarrow \quad \frac{P(C=1 \mid s)}{P(C=0 \mid s)} = k \frac{Q(C=1 \mid s)}{Q(C=0 \mid s)} \quad \forall s \qquad (1)$$

Besides, a consequence of this hypothesis is:

[18] This hypothesis may not be true in general, especially is one class is formed by a disjunction of classes.

$$Q(s \mid C) = P(s \mid C) \quad \forall s$$

In particular, we have:

$$\frac{P(C=1)}{P(C=0)} = k \frac{Q(C=1)}{Q(C=0)} \qquad (2)$$

Therefore, we can characterise completely P through Q is we find the factor k.

According to this trick, we have a sample of observations that follows the law Q:

$$(X_1, X_2, ..., X_n)$$

We want to estimate $P(C \mid s)$ from this sample. The following relations hold:

$$\begin{aligned} P(C=1 \mid S \in D_j) &= \frac{P(S \in D_j \mid C=1)P(C=1)}{P(S \in D_j)} \\ &= \frac{P(S \in D_j \mid C=1)P(C=1)}{P(S \in D_j \mid C=1)P(C=1) + P(S \in D_j \mid C=0)P(C=0)} \\ &= \frac{1}{1 + \frac{P(S \in D_j \mid C=0)P(C=0)}{P(S \in D_j \mid C=1)P(C=1)}} = \frac{1}{1 + \frac{Q(S \in D_j \mid C=0)}{Q(S \in D_j \mid C=1)} \frac{P(C=0)}{P(C=1)}} \\ &= \frac{1}{1 + \frac{Q(C=0 \mid S \in D_j)}{Q(C=1 \mid S \in D_j)} \frac{Q(C=1)}{Q(C=0)} \frac{P(C=0)}{P(C=1)}} = \frac{1}{1 + \frac{1}{k} \frac{Q(C=0 \mid S \in D_j)}{Q(C=1 \mid S \in D_j)}} \end{aligned}$$

Thus, we obtain as an estimator for $s \in D_j$:

$$\begin{aligned} \tilde{r}(s) \equiv \tilde{P}(C=1 \mid S=s) = \tilde{P}(C=1 \mid S \in D_j) &= \frac{1}{1 + \frac{1}{\tilde{k}} \frac{N_n(C=0, S \in D_j)}{N_n(C=1, S \in D_j)}} \\ &= \frac{w_1 N_n(C=1, S \in D_j)}{w_1 N_n(C=1, S \in D_j) + w_0 N_n(C=0, S \in D_j)} \end{aligned}$$

We see that it is equivalent to calculate the relative frequency of the class 1 in the interval D_j with the counting of each class being affected by some weights w_0 and w_1 such that:

$$\frac{w_1}{w_0} = \tilde{k} = \frac{P(C=1)}{P(C=0)} \frac{N_n(C=0)}{N_n(C=1)}$$

We remark that these weights are random variables, as there are statistics obtained from the sample. Moreover, they break the independence of the observations, as each observation is affected by a weight that depends on the *whole* sample.

If we wanted to apply the computation confidence intervals to **assess the reliability** of these estimates, we should have to calculate an **approximate variance** of this weighted average. To simplify the analysis, we consider that w_0 and w_1 are deterministic.

Again, consider $(I_1,...,I_j,...,I_m)$ the indicator variables corresponding to the subset of examples whose scores fall in A. Let $N_1 = \sum_{j=1}^{m} I_j$. It is a binomial variable.

All the possible outputs are given by $r_i = \frac{w_1 i}{w_1 i + w_0(m-i)}$ for $i = 1,...,m$

Each of these possible outputs has probability $p_i = \binom{m}{i} q^i (1-q)^{m-i}$

where $q = Q(C = 1 \mid S \in A)$ that we estimate as $\frac{N_1}{m}$.

Therefore, the estimate for interval A is

$$\tilde{P}(C = 1 \mid S \in A) = \frac{w_1 N_1}{w_1 N_1 + w_0(m - N_1)}$$

and an approximate variance

$$\mathrm{var}\left[\tilde{P}(C = 1 \mid S \in A)\right] \approx \sum_{i=1}^{m} p_i r_i^2 - \left(\sum_{i=1}^{m} p_i r_i\right)^2$$

This approximate calculation of the variance depends on the weights and on the estimate of $q = Q(C = 1 \mid S \in A)$. We observe that there are extreme cases which are very unfavourable. Very different weights may lead to considerable errors, notably when one class has much more weight than the other and the opposite proportion is observed in the studied interval A. Figure 22 shows the variance obtained with the previous formula for different weight ratios, different number m of examples in the interval and with q chosen so that the obtained value is maximal. The vertical axis corresponds to $\frac{\mathrm{maxvar} - 0.25}{0.25\left(1 - \frac{1}{m}\right) + 1}$.

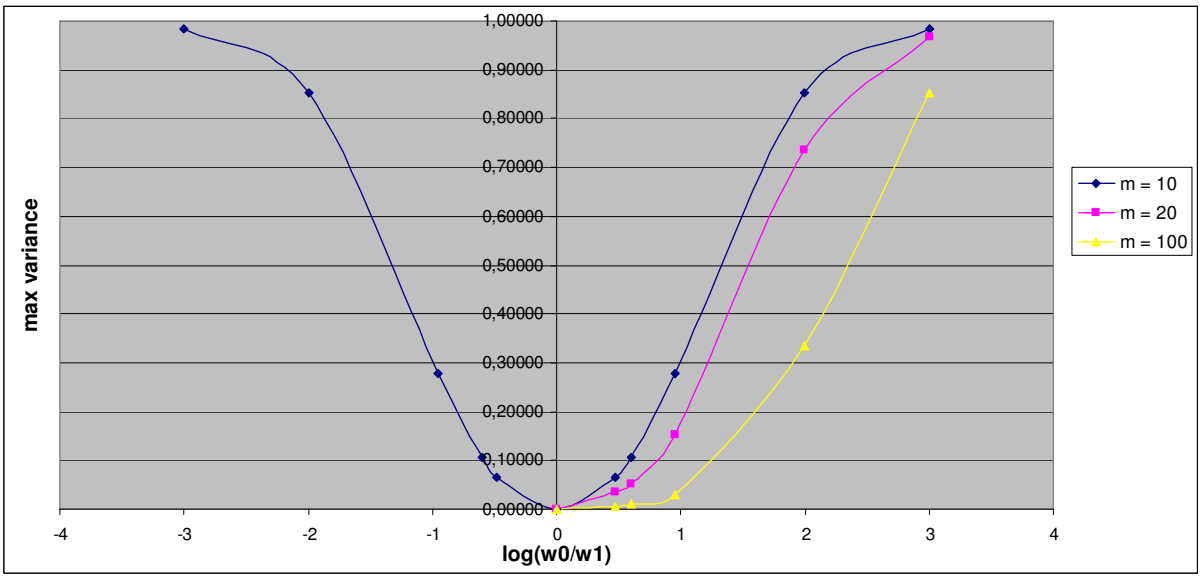


Figure 22. Worst case of variance for different weight rations.

Therefore, we think that the theoretical study in this case may not be appropriate, and a better solution could be the empirical assessment through multiple calibrations using subsets of the data.

5.2.4 Chosen approach: Isotonic Regression

Consider $X \in \mathbf{R}^d$ the explanatory variable (input) and $Y \in \mathbf{R}$ the corresponding response variable. The problem of **isotonic regression** (or monotonic smoothing) on a set $\{(X_i, Y_i)\}_{i=1}^n$ of two-dimensional data can be formalized as follows [16]:

- Sort the data $\{(X_i, Y_i)\}_{i=1}^n$ by X into $\{(X_{(i)}, Y_{(i)})\}_{i=1}^n$

- Find $\{\hat{m}(X_{(i)})\}_{i=1}^n$ to minimize $\frac{1}{n}\sum_{i=1}^{n}\left(Y_{(i)} - \hat{m}(X_{(i)})\right)^2$ subject to the monotonicity restriction:

$$\hat{m}(X_{(1)}) \leq \hat{m}(X_{(2)}) \leq \ldots \leq \hat{m}(X_{(n)})$$

The **pool adjacent violators** algorithm (PAV) (Ayer et al., 1955) gives a stepwise-constant solution. Roughly, it consists in sorting the data by X, scanning the data in order and averaging those outputs Y that violate the monotonicity restriction as one goes along. See [16] for more details.

In our context of application, $X \in \mathbf{R}$ is the score and $Y \in \{0, 1\}$ is the class[19]. The intervals of score are defined by the groups of examples that we average together following the PAV algorithm. The output values correspond to the empirical probability in each interval. If the classifier ranks the examples correctly, the mapping between scores and probabilities is indeed non-decreasing. Thus this calibration method is very appropriate, as this is a presumable characteristic of the classifiers that we use.

[19] Moreover, the algorithm has been adapted to include the different weights of each piece of data.

The interesting advantage of this calibration method is that the user does not choose any parameter. The intervals are defined according to the statistical properties of the data and the regularisation constraint (monotonicity). We could say that the intervals are narrow or wide where it is needed.

The problem arises when the classifier does not rank the examples correctly, i.e. the ideal mapping is not non-decreasing. From the point of view of the PAV algorithm, this provokes outliers or aberrant observations among the data. If there are outliers, the PAV algorithm produces long flat levels (Figure 23).

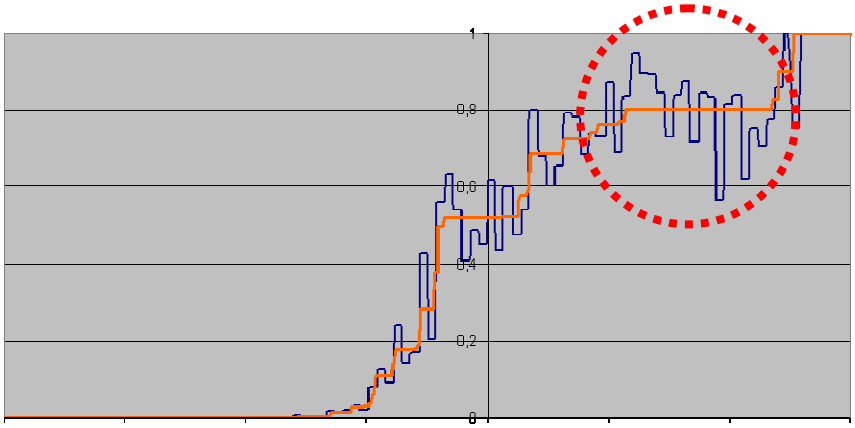


Figure 23. Calibration through isotonic regression (orange), and approximation of the underlying distribution (blue).

In these cases, the estimate for the given interval may not be pertinent for all the scores: it is impossible to know if the outliers are just due to noisy data or if the ideal mapping is decreasing in that interval. We can then use the approach described in following paragraphs to assign different reliability to different sub-intervals.

In Figure 24, we can see the underlying function in an interval of scores and the estimated average. We can assign an upper bound to this estimate and keep it as a reference[20]. Then, we can consider shorter sub-intervals towards the left, calculate the average and the corresponding lower bound. If the underlying function is indeed decreasing, we could find sufficient statistical evidence to affirm that the estimated average in the sub-interval does not correspond to the average in the whole interval, i.e. the lower bound is greater than the reference upper bound. We can repeat the same process towards the left, considering the bounds in the other sense.

In the case of finding this statistical evidence, we could assign a new estimate to the sub-interval. This would obviously violate the non-decreasing constraint. Instead, we can choose to keep the estimate as it is (i.e. the one obtained through isotonic regression) but give less reliability to the sub-interval, according to the difference between the local average and the average in the whole interval.

[20] For clarity purposes, the figure shows the confidence interval. In practice, it may be more convenient to work only with upper or lower bounds according to the need, as the obtained bounds are tighter.

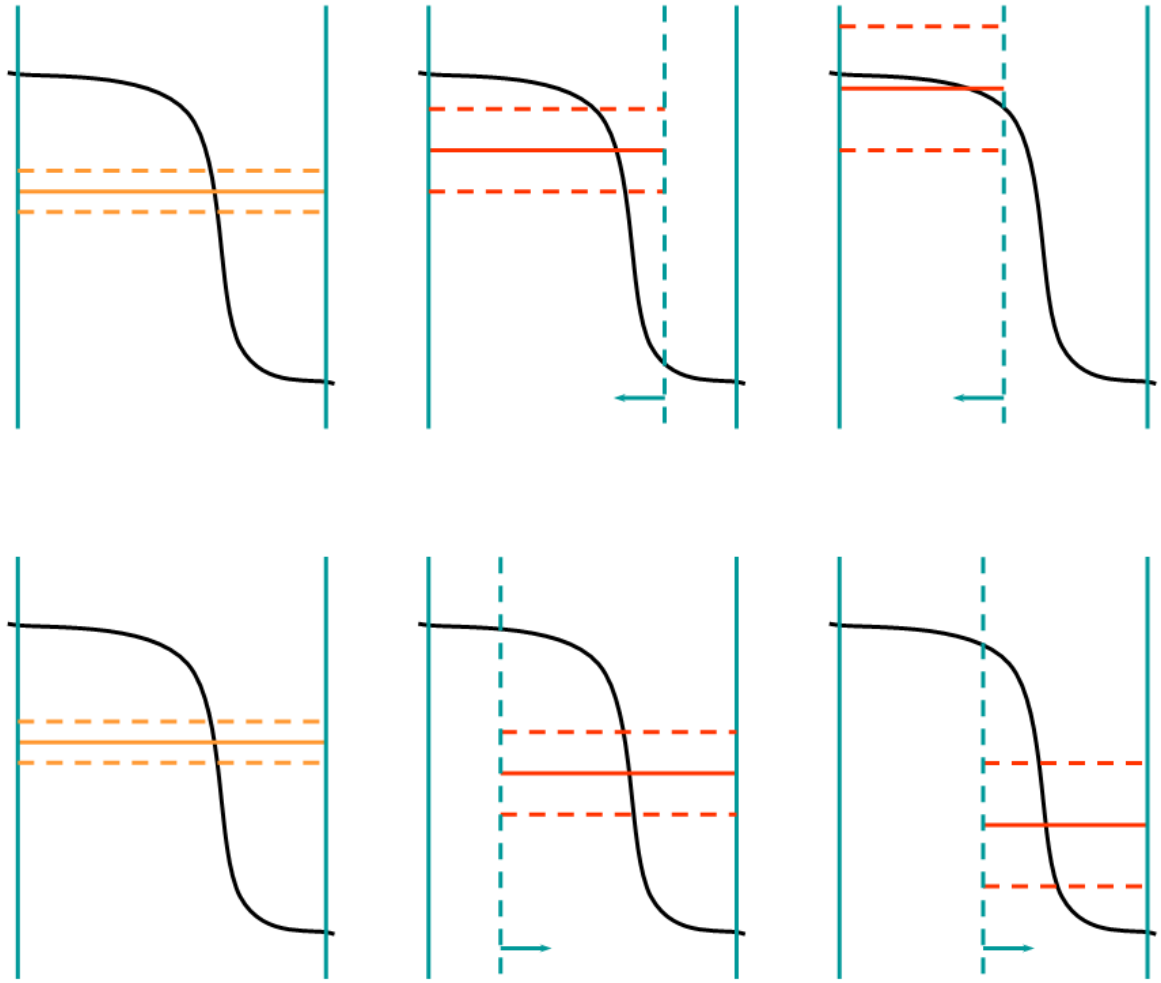


Figure 24. Exploration of an interval towards the left and towards the right.
We can see the underlying function (black), the reference average (orange) and the reference confidence interval (dotted orange), the average on a subinterval (red) and the corresponding confidence interval (dotted red).

5.3 Error Correcting Output Codes

5.3.1 Related work

For some classification problems, both two-class and multiclass, it is known that the lowest error rate is not always reliably achieved by trying to design a single best classifier [17]. An alternative approach is to combine a set of simpler classifiers. These classifiers may be simpler for different reasons. For example, in boosting, the weak classifiers are allowed to have a performance only better than chance [6]. In other cases, the classifiers handle simplified versions of the original problem. This is the case of the reduction of the multiclass problem into several binary problems, which has already been introduced in section §4.1.

A necessary condition for improvement by combining is that the results of the base classifiers are not too well correlated, as discussed in [18]. Strategies to promote independence are perturbing feature sets, injecting randomness or perturbing training sets [17]. The last is the case of bagging and boosting [2]. The reduction of the multiclass problem to several binary problems can be seen as a perturbation of class labels [17].

There are several motivations for decomposing the original problem into separate and complementary binary problems. The main one is that there are some accurate and efficient binary classifiers that cannot be naturally extended to handle the multiclass case [14]. Moreover, solving different binary sub-problems may help to reduce error in the original problem, [21].

There are many ways of making this decomposition. In the *one-against-all* approach mentioned in section §4.1, each class is compared to all others.

Hastie and Tibshirani suggest a different approach in which all pairs of classes are compared to each another. This approach is called *all-pairs* and generates $\binom{k}{2}$ binary classifiers.

A more general suggestion was given by Dietterich and Bakiri. Their idea is to associate each class to a row of a coding matrix $M \in \{-1,1\}^{kxl}$ for some l. Each column induces a binary problem, as explained in section §4.1. This is the method of *error-correcting output codes* (ECOC) [20]. The ECOC method was originally motivated by error-correcting principles, under the assumption that the learning task can be modelled as a communication problem, in which class information is transmitted over a channel.

Allwein et al. [14] propose a unifying generalization of all three. In fact, they take the matrix from the larger set $\{-1,0,1\}^{kxl}$. If $M(c,b)=1$, then the examples of the class c are considered to be positive examples for the binary classification problem b. If $M(c,b)=-1$, the examples belonging to c are considered to be negative examples for b. Finally, if $M(c,b)=0$, the examples belonging to c are not used to train b. This extension allows the inclusion of the *all-pairs* approach as a particular case.

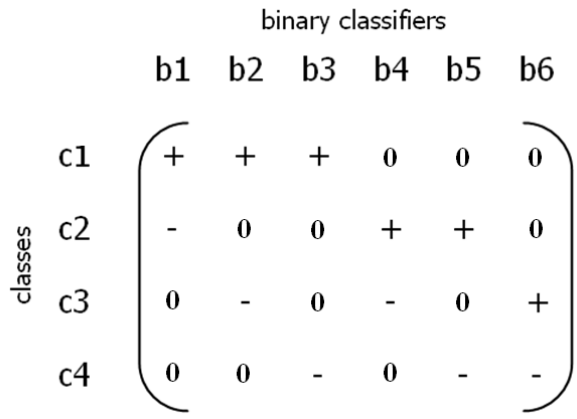


Figure 25. The all-pairs code matrix in the 4-class case.

Since its appearance, ECOC has been successfully used in many application areas but there is discussion about why it works well, and some have brought into question the significance of the coding strategy [17]. Several papers have appeared that address the problem of choosing an appropriate coding matrix[21].

From the perspective of error-correcting theory, it is desirable that codewords are far apart. [14] deduce a bound on the generalisation error that confirms this, as well as a low number of zero entries. However, they note that this may lead to difficult binary problems.

[17] discusses different aspects of the ECOC algorithm, notably the desired properties of the coding matrix and a method for producing equidistant codes. They note that sub-problems are more independent and likely to benefit from combining if distance between columns is high. They remark that according to previous reported results, a long random code appears to perform as well or better than a code designed for its error-correcting properties.

In [19] the authors attempt to prove a theoretical statement about the performance of the ECOC approach. They argue that one reason why the powerful theorems from coding theory cannot be directly applied to prove stronger bounds on the performance of the ECOC approach is that, unlike in coding theory where one usually assumes that the errors in the different bit positions occur independently, in the classification context the errors made by the various hypothesis can be considerably correlated. They prove that, given strong error-correction properties of the underlying code and a small error-correlation between the hypotheses, the ECOC approach is guaranteed to produce a classifier with small error. They use a particular family of codes called Hadamard-matrix codes.

[26] list some of the different families of codes that have been suggested besides the one-against-all and the all-pairs: random codes, exhaustive codes and linear correcting codes. They argue that the use of predefined output codes ignores the complexity of the induced binary problems. They are the first to discuss the problem of designing a good output code for a given multiclass problem, instead of using predefined

[21] For clarity, the the possible combining strategies are discussed separately in section §5.4. However, it has to be remarked that both aspects may be tightly related. Thus, the optimal matrix ultimately depends on which combining strategy is used.

codes. Motivated by the intractability results, they introduce the notion of *continuous codes* in opposition to discrete codes, and they cast the design problem as a constrained optimization problem.

With the same concern about the learnability of the binary problems, [27] introduce the notion of *probabilistic output codes*, and propose an approach that ties the learning problem with the class representation problem: they describe an algorithmic framework in which the set of classifiers and the code are found concurrently. We have observed that, as far as our work is concerned, this framework has the limitation that the classifiers must be logistic regressors. Margin classifiers like SVM are not suitable because the do not provide an *analytical* expression for the conditional probabilities.

In parallel to the work about the decomposition of the multiclass problem into binary problems, it is still desirable to have a multiclass algorithm that treats all classes simultaneously. This is addresses by [28]. They focus on margin-based classifiers, among which SVM and boosting are the most popular techniques. They remind that both techniques have excellent performances in the binary classification problem, but it is nontrivial to extend them to multiclass cases. About the widely used *one-against-all* strategy, they say that while working fine with AdaBoost, this approach can perform poorly with SVM if there is no dominant class, and give some references to these results. Then they recall some recent and successful proposals of treating the multiclass problem directly, without decomposing it into binary problems. Moreover, they propose a new framework to formulate the margin classifier, in which the binary and the multiclass problems are solved by the same principle: regularized empirical risk minimization. We think that this approach may be very interesting, and it is worthy to follow the improvements in this field. However, in spite of the good performances in the classification task, we must reject it as a possible approach in this work because it does not focus on the discovering of the conditional probabilities of belonging to each class, which is our objective. Instead, the output is a vector of scores, which we cannot calibrate with the techniques proposed so far due to the curse of dimensionality.

In the next section we focus on the construction of a matrix with good properties. We propose a heuristic method compatible with practical constraints of computational cost. The decoding or combination strategies will be treated in section §5.4.

5.3.2 Construction of the coding matrix

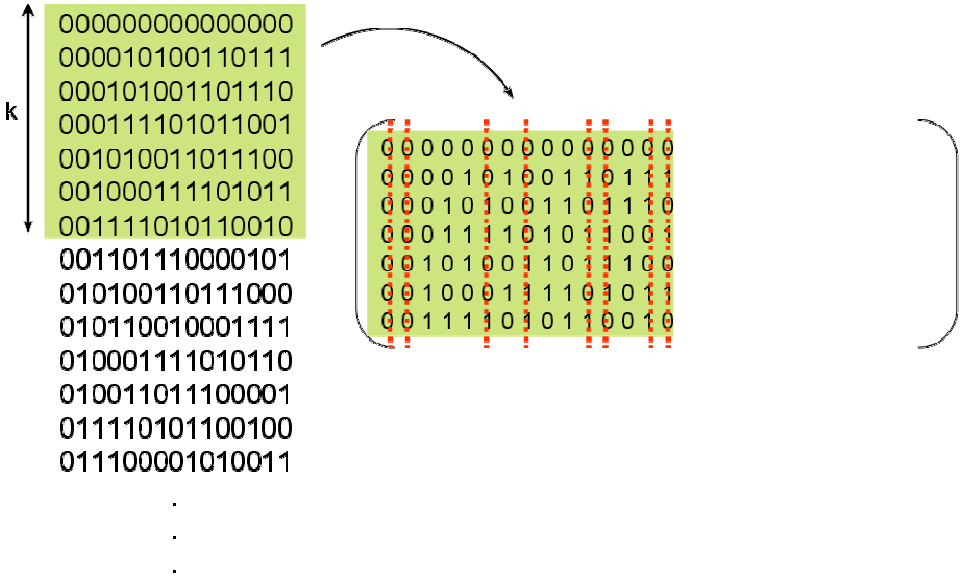
The number of possible matrices explodes with the number of classes. Considering all of them is neither possible nor interesting. As an alternative we limit our research to a maximal number of columns. Besides, we are going to focus on two-valued matrices (-1/+1) instead of the more general case of three-valued matrices (-1/0/+1), as we consider interesting the fact that all classifiers give some information about *all* classes, and that all classifiers are trained with the same number of samples[22].

If we wanted to create a coding matrix with determined correcting properties from the point of view of coding theory, we would have to generate an error-correcting code and pick a number of words equal to the number of classes in our problem. The number of classifiers would be the length of the words.

[22] In the context of the matrix construction, we may use -1/+1 or 0/1 interchangeably.

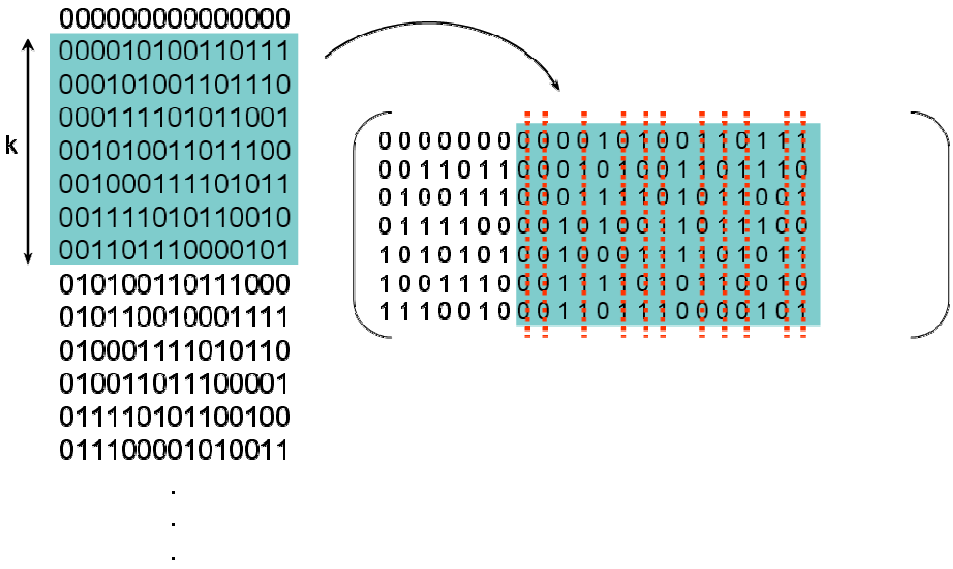
In practice, we want to build a coding matrix in which the number of rows (classes) and columns (classifiers) is adapted to the problem. The ideal number of columns is unknown, but intuitively it increases with the number of classes. As codes are not easy to generate, it is not practical to generate a different code for each case. And even if this could be done, repeated classifiers must be deleted from the coding matrix, so the integrity and properties of the code would not be guaranteed. In any case, it has already been discussed that the correcting properties are useful only if the classifiers are sufficiently uncorrelated. The correlation among classifiers cannot be studied theoretically as we do not know anything about the nature of the classes. Therefore, we can only assess the quality of the matrix empirically.

As alternative to the matrix being exactly a code, we rather use a code only as a base for building the matrix. Concretely, we have chosen a fixed **BCH code**[23]. The code has 32 words 15 bits long each and the Hamming distance between words is at least 7. With this code[24] and a given number of classes, we can build a matrix of a sufficiently large number of columns, and presumably with better correcting properties than a random matrix. The procedure is illustrated in Figure 26.



(a)

[23] See [29] and [30] for more details on polynomial codes and the correcting properties of BCH codes, as well as their construction.
[24] This particular code was chosen because it is sufficient for a large number of classes and the distance was high. If it turned out to be insufficient for a particular number of classes, we can always generate a different one.



(b)

Figure 26. Procedure for generating the classifier candidates for the coding matrix: first (a) and second (b) iteration.

Consider the sub-matrix formed by the first k words (from 1 to k) of the BCH code. We take its columns one by one and add them to our coding matrix, provided that the column or its complementary does not already exist in the coding matrix. We repeat the same actions for words from 2 to k+1, until the 32 words have been considered. When the number of classes is very low, this procedure generates all possible classifiers. When the number of classes is high, a **sufficiently high number of classifiers** are generated and we expect the matrix to have **better correcting properties than a random matrix**, as it has been constructed using pieces of codewords as "bricks".

Once this maximal number of columns has been generated, we would like to find the **optimal combination**, because keeping all the columns may not be the most appropriate solution. Besides, we have constraints in the prediction time.

The number of possible combinations explodes. We have designed a heuristic procedure for **iteratively** exploring these possible matrices and choosing one. Initially, the matrix is evaluated. At each iteration **the worst column is deleted** and the resulting matrix is re-evaluated. By "worst column" we mean that the induced binary problem is the most difficult. **The evaluation of the matrix and the difficulty of the binary problems are based on empirical performance**, as it is described next.

The step following the generation of all columns is the generation of the induced classifiers. To do so, a learning set labelled according to each column is used to train a binary classifier, and a validation base to calibrate it. For the procedure to be quick, no optimization of the classifier's parameters is done, so it is desirable that the parameterisation of the classifier algorithm does not critically influence its performance, in order to reliably estimate the difficulty of the binary problem. Boosting is a good candidate provided it is implemented to output scores and not binary decisions. The difficulty of the problem is assessed according to the accuracy of its predictions on the validation base. In particular, the minimal probability of detection is

taken as evaluation of the classifier. For each example in the validation base, the predictions of the set of classifiers are combined to obtain a probability distribution, and the class with maximal probability is compared to the real class of the example. We take the percentage of correct answers as the evaluation of the whole matrix. We include also a penalty for the number of classifiers (or equivalently the prediction time).

The predictions of the classifiers on the validation base and the evaluation of the difficulty of the binary problems are always the same independently of the combination of classifiers. So, in order to save time and memory resources, they are pre-calculated only once and stored. The only thing that has to be done at each iteration is the fusion of these predictions in order to obtain the evaluation of the considered sub-matrix.

When a number of columns equal to the number of classes has been reached, the procedure stops and we take as **definitive matrix** the one with the **best evaluation** among the set of sub-matrices that we have evaluated. Optimised classifiers can then be trained and calibrated.

5.4 Combination of classifiers outputs

5.4.1 Related work

Once we have produced diverse classifiers, a suitable combining strategy must be designed. Different approaches have been suggested according to two possible aims: inferring the correct class given the set of outputs, or obtaining probability estimates for each class.

5.4.1.1 Making decisions in the context of ECOC

According to the original motivation of the ECOC method, it seems sensible to predict the class whose row of the matrix is closest to the set of outputs, for some distance.

The first method of combining was *Hamming decoding*. Here, the distance is the number of positions in which the decisions of the base classifiers differ from the row. This is a hard-level combining strategy, meaning that it only takes into account the individual decisions, in contrast with soft-level which considers a measure of confidence associated to each individual prediction.

[14] propose a combining strategy suitable for margin-based classifiers, a general class of binary algorithms that attempt to minimize a loss function of the margins[25]. SVM and AdaBoost are particular cases, among others[26]. They remark that the disadvantage of Hamming decoding is that it ignores entirely the magnitude of the predictions, which can be taken as a measure of confidence in the case of margin-based classifiers. They propose *loss-based decoding* as an alternative. This approach takes into account not only the obtained scores but also the loss function that was minimized during the training to generate the

[25] The margin of an example is defined as the magnitude of the output and a positive sign or negative sign depending on if the example is correctly classified or not

[26] They show that SVM and AdaBoost can be viewed as a binary margin-based learning algorithm in which the loss function is $L(z)=(1-z)_+$ and $L(z)=e^{-z}$ respectively.

classifiers. The distance measure is the total loss on an example. They also note that the loss-based decoding method for log-loss is the well-known and widely used maximum-likelihood decoding, which was studied briefly in the context of ECOC by [19].

[17] describes also Dempster-Shafer decoding, based on maximising probability mass function, and Centroid decoding, based on minimising Euclidean distance to Centroid of classes.

5.4.1.2 Probability estimation in the context of ECOC

Other decoding strategies consist in recovering individual class probabilities.

Let's consider an arbitrary matrix $M \in \{-1,0,1\}^{kxl}$ and the associated set of binary classifiers, whose outputs are probability estimates. For each column b of M and each example $\mathbf{x}$ with class c, we have an estimate $r_b(\mathbf{x})$ such that:

$$r_b(\mathbf{x}) = P(c \in I \mid c \in I \cup J, \mathbf{x}) = \frac{\sum_{c_i \in I} P(c = c_i \mid \mathbf{x})}{\sum_{c_i \in I \cup J} P(c = c_i \mid \mathbf{x})}$$

where I and J are the sets of classes for which $M(c,b) = 1$ and $M(c,b) = -1$ respectively. For each example $\mathbf{x}$, we would like to obtain a set of probabilities $P(c = c_i \mid \mathbf{x}) = p_i(\mathbf{x})$ compatible with the set of $r_b(\mathbf{x})$. Note that if the matrix has no zero entries, the expression reduces to $r_b(\mathbf{x}) = P(c \in I \mid \mathbf{x}) = \sum_{c_i \in I} p_i(\mathbf{x})$.

This is an over-constrained problem. [13] recalls two approaches that have been proposed to solve it. The first one is a least-squares method with non-negativity constraints [22]. The second one is a method called *coupling* [23], an iterative algorithms that finds the best approximate solution minimizing the Kullback-Leibler divergence instead of squared error. It is an extension of the pairwise coupling method proposed by Hastie et Tibshirani [24] that only applies to all-pairs matrices. Which approach is the most appropriate is still an open question.

5.4.2 Chosen approach: coupling

In the case of the one-against-all matrix, the simplest way of obtaining the membership probability estimates consist in normalizing the results, as each result already corresponds to the estimate of belonging to one class. For an arbitrary matrix, the over-constrained system presented above has to be solved. We have chosen the extension of the Hastie-Tibshirani method for coupling the probability estimates produced by the calibrated binary classifiers.

The method starts with a guess of probabilities (we choose $p_i = 1/k$ for all classes). The set of $\hat{r}_b$ corresponding to these probabilities are calculated according to the expression in section §5.4.1.2., and they

are compared to the observed r_b[27]. The probabilities p_i are then updated in a way that progressively minimizes the Kullback-Leibler divergence between the vectors formed by the set of $\hat{r}_b$ and the set of r_b. At each iteration, the probabilities p_i are normalized to sum 1. See [23] for more details.

In [24] a demonstration of the convergence of the algorithm for the all-pairs case is provided. In [23], the extended method is claimed to converge as well. In our implementation we have included a criterion that stops the algorithm when the divergence has varied little with regards to the previous iteration. We have observed that after few iterations there is normally one class whose probability stands out from the rest. This means that if we only want to make a classification – i.e. to decide a class without caring about probabilities – we can adjust this parameter to save a lot of computation time without an impact on performances. On the contrary, if we are interested in accurate probability estimates – notably in order to make a ranking later on – we have to push the algorithm further.

It is important to remark that the calculation of the divergence – and thus the value of the updating factors – may be weighted to give more or less importance to the match between certain components $\hat{r}_b$ and r_b. The idea is to focus on matching the reliable observations r_b, so as to converge to the good solution even if the r_b are not compatible.

In the original paper [24] for the all-pairs matrix, Hastie and Tibshirani include the number of examples used for training each classifier[28] as weights as "a crude way for accounting for the different precisions in the pairwise probability estimates". We argue that this can be done in a finer way. In our case, the probability estimates come from the calibration. We have observed that depending on the score, the obtained estimates can be more or less reliable. Instead of assigning a confidence to each classifier, we have paid attention to the assignation of a confidence to each output score of each classifier. Thus, each classifier is weighted differently according to the input example and its ambiguity.

Finally, we have remarked that the Kullback-Leiber divergence is widely used to compare probability distributions [25], but the set of r_b is *not* a probability distribution. They are individual probability estimates, but they not correspond to mutually exclusive events, and thus the sum is not supposed to be 1. In our opinion, this leaves an open door to other possible measures and other resulting algorithms.

5.5 Temporal fusion

5.5.1 Fusion of information sources

A number of different paradigms for performing data and information fusion have been developed. They differ in the way they represent information and more concretely in the way they represent uncertainty. This means that each of them models different aspects of knowledge. Therefore, they are

[27] The Kullback-Leibler divergence between the two vectors is calculated.
[28] In the all-pairs case, and in general when the matrix has zero entries, the classifiers are not trained with all the examples in the data set, as the classes corresponding to the zero entries are ignored.

difficult to compare in order to choose the most appropriate for a given problem. In this section, we describe briefly the three big families, based on the explanations in [31] and the introductions in [34] and [35].

Bayesian probability theory articulates belief through the assignment of probability mass to mutually exclusive hypothesis. Typically, the combination method in this framework consists in applying Bayes rule. Probabilistic approaches have many advantages, notably their axiomatic justification, a coherent and powerful mathematic framework and the great number of existing statistics tools, due to their generalised utilisation for many years. They have also many disadvantages. The probabilistic approach constitutes a very constrained model of knowledge, which represents uncertainty but is unable to model imprecision or ignorance, and which finds it hard to introduce information that is not easily represented as probabilities. Moreover, it needs a lot of prior information and imposes hypothesis that are not always verified in practice.

The paradigm based on fuzzy logic and possibilities represents belief through the definition of a mapping between quantities of interest and belief functions. It handles imprecision and offers a great number of fusion operators with different behaviours, which gives flexibility in the modes of combination.

Dempster-Shafer theory generalises Bayesian theory to consider upper and lower bounds on probability. This theory – so called theory of evidence or theory of belief functions – has been further reinterpreted and generalised by means of the transferable belief model and Dezer-Smarandache theory. It represents uncertainty through the assignment of probability mass – or degrees of belief – to all possible subsets of a set of mutually exclusive hypotheses. The strong point of this theory is that it manipulates uncertain, imprecise and conflicting information, as well as ignorance. Furthermore, it handles the case of different judgement spaces among the sources of information. Dempster's rule is the basic tool for combining the degrees of belief when they are based on independent items of evidence, which does not necessarily hold in practice. Many alternative combination rules have been proposed in this framework (see [34]). See [32] or [33] for an introduction to this theory and applications. An important drawback is that the computational cost of the associated methods is considerably high, as it increases exponentially with the number of classes.

Recently, some work has appeared that tries to conciliate the probability theory and the theory of evidence approaches. In particular, [35] considers how to devise Bayesian models that have the same properties of handling uncertain, imprecise and conflicting information, exploiting recent advances in the Bayesian analysis of complex data.

In [36] they propose a conceptual framework for fusion of beliefs in hypothesis represented by propositions of the form "the element *e* belongs to the set *V*". They use the notion of belief in such a hypothesis in the framework presented by R.T. Cox in 1946, which combines propositional logic and probability theory. Probability is reinterpreted as a formal system of logic, an extension of classical logic in which the valuation of propositions is not known with certainty [37]. In their paper, the approach by logical fusion is deduced and different modes of fusion are obtained.

As a probabilistic approach, logical fusion has the advantage that all the obtained modes of fusion are formal theorems – and not heuristics – directly deduced from the axiomatic of probabilities. Moreover, it allows the handling of contradictory beliefs in a naturally way, which is not possible with purely Bayesian fusion. However, the authors remark that its main drawback is precisely due to its logical foundations, as the

fusion is limited to a simple logical combination, which might not be enough in some applications. They leave an open door to approaches based in the theory of evidence and the handling of multimodal hypotheses.

We have chosen logical fusion for this part of our processing chain because it is coherent with the probabilistic framework so far, it is less complex to implement, it has a very low computational cost and the logical combination of beliefs is well-adapted to the ranking purpose.

5.5.2 Probabilistic Logical Fusion

In the framework of logical fusion, the *system* ignores if a hypothesis *H* is true, and asks the opinion of *n experts*. The particular case of interest to our work is when the *H* has the form "element *e* belongs to set *V*". In our case, the element is the object to which we want to assign a category, and the set is a singleton containing one of the possible categories. Each expert *i* provides a *belief*, which is conditioned to a set of hypothesis[29]. The system assumes that this set of hypothesis is considered actually true by this expert – this assumed proposition is called E_i – and obtains a belief of the form $P(H|E_i)$, $i = 1,\ldots,n$.[30]

Roughly, the output of the fusion is the probability of *H* conditioned to the knowledge of the system and a logical combination of experts E_i. However, to be consistent with the axiomatic of probability, not all combinations are possible. The conjunction can only be applied to sets of globally non-contradictory experts. A group of experts is **globally non-contradictory** if the intersection of their law's supports is not empty, that is that they all agree at least in one of the categories.

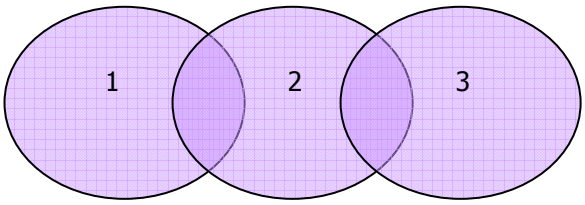


Figure 27. Group of globally contradictory experts.
The figure represents the support of their beliefs in the space of the possible categories.

In the example of Figure 27, experts 1 and 2 are non-contradictory, experts 2 and 3 too, but experts 1 and 3 are contradictory. Thus, the system considers that E_1 and E_3 are necessarily exclusive. As E_1E_3 is false, the system cannot possibly condition his belief to this proposition. Instead, it could condition its belief to $E_1E_2 + E_2E_3$.[31]

The framework provides different **modes of fusion** depending on the logical combination of experts. The typical modes are:

- **Conjunctive**. In this mode of fusion, the system assumes that all combined experts are reliable. Of course, it can only be applied to a group of globally non-contradictory experts. Therefore, there are cases where it is impossible to apply it to the total of experts. The resulting belief is limited to the categories in

[29] For example, the circumstances surrounding the expert influence his belief. These circumstances are unknown by the system.
[30] This belief is a probability distribution, as there is one different H for each possible category.
[31] We use the product for conjunctions and the sum for disjunctions.

which all the combined experts believe. From the computational point of view, it corresponds to the product of the P($H|E_i$) followed by a normalisation.

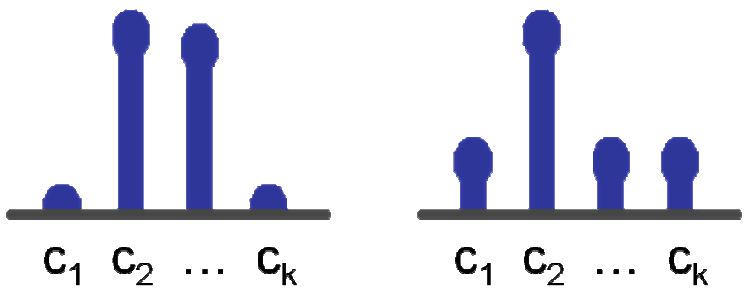
- **Disjunctive**. In this mode of fusion, the system assumes that at least one expert is reliable. The resulting belief is limited to the categories in which at least one expert believes. From the computational point of view, it corresponds to the normalised average of the P($H|E_i$) pondered by the degree of confidence in each expert.

- **Hybrid**. It consists of a conjunctive fusion of those groups of experts that are globally non-contradictory, and a disjunctive fusion of the beliefs of these groups. Thus, it reduces to the conjunctive fusion when all experts are globally non-contradictory and to the disjunctive fusion when all experts are in contradiction. We can think of it as the most conjunctive fusion possible that takes into account all experts.

We have implemented all possible modes of fusion, verifying which logical combinations of experts are compatible. In the context of a query with regards to a particular class, a conjunctive fusion would rank an object for which all experts agree in this class before another object for which not all of them agree. On the other hand, a disjunctive fusion would rank an object for which one expert believes in this class before another object for which any expert does. Both behaviours are appropriate in our context, so we have finally chosen a weighted sum of hybrid and disjunctive fusion, and we have empirically observed that it works better that simply hybrid fusion in some databases. The weight can be set to obtain a fusion that varies from disjunctive fusion to the most conjunctive fusion possible. We have chosen a balance between the two of them, but we could think of more sophisticated ways of adjusting this parameter according to the nature of the data. In particular, we could use a training set to learn it statistically.

5.6 Computation of the ranking scores

Consider the two distributions obtained for two objects in Figure 28. If we want to rank them with regards to the second class, we would intuitively prefer the first one rather than the second, as it seems to be less doubt that it belong to the desired class. So the estimated probability alone seems not enough to do a sensible ranking. We have empirically verified this in the case of a lot of classes, when the probabilities are spread out among the classes. Sometimes a class stands out from the rest but its absolute value is not high because the other classes have important residual probabilities. So not only the probability estimate for one class is a relevant aspect, but also the relationship with the estimates for the rest of classes. A score that represents all this information would be useful.



(a) (b)

Figure 28. Two distributions obtained for two objects.
The class probability estimate is not enough to rank the objects.

To solve this question, we have thought of the Dirac delta distribution as the representation of the most pertinent distribution with regards to a particular class. For each possible query, we can compute a distance between the obtained distribution and a Dirac delta distribution and use it to build a score that leads to sensible rankings. We have chosen a Euclidean distance because it introduces a penalty in the case of Figure 28.a in comparison to Figure 28.b. In the case of a query consisting of multiple classes, we could first sum the probability estimates and then apply the same procedure for computing the score.

6 Evaluation

6.1 Infom@gic database

6.1.1 Description

In the context of the Infom@gic URBAN VIEW chain, we have worked with the NGSIM database. It consists of very low resolution images obtained from several surveillance cameras. Each small image belongs to one of these categories: pedestrian, group of people, bicycle, motorbike, car, van, bus, truck and group of cars. Moreover, each image is associated to metadata issued from the motion detection and the viewing conditions, notably a binary mask that separates the object from the background and information about the speed vector.

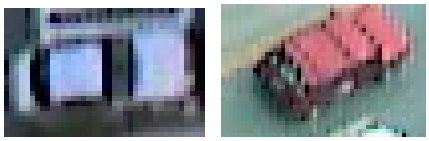


Figure 29. Examples from the NGSIM database

This information has been used to extract 6 geometric descriptors that represent each example. Some categories have been merged to avoid insufficient representation of the classes in the database. The learning, validation and test sets come from different cameras, so that they are not too correlated. The tables below show the distribution of the elements in the different sets, before and after merging classes.

	Learning		Validation		Test	
	#	%	#	%	#	%
0 pedestrian	725	10,5%	282	6,1%	164	13,2%
1 car	4848	70,5%	2335	50,2%	648	52,0%
2 van	207	3,0%	307	6,6%	117	9,4%
3 bicycle	9	0,1%	1	0,0%	0	0,0%
4 motorbike	17	0,2%	20	0,4%	0	0,0%
5 group of people	16	0,2%	19	0,4%	69	5,5%
6 truck	348	5,1%	576	12,4%	133	10,7%
7 bus	80	1,2%	215	4,6%	40	3,2%
8 group of cars	385	5,6%	769	16,5%	51	4,1%
99 false alarm	241	3,5%	130	2,8%	25	2,0%
total	6876		4654		1247	

Table 12. Distribution of classes in the NGSIM database before merging

	Learning		Validation		Test	
	#	%	#	%	#	%
0and5	741	10,8%	301	6,5%	233	18,7%
1and2	5055	73,5%	2642	56,8%	765	61,3%
3and4and99	267	3,9%	151	3,2%	25	2,0%
6and7	428	6,2%	791	17,0%	173	13,9%
8	385	5,6%	769	16,5%	51	4,1%
Total	6876		4654		1247	

Table 13. Distribution of classes in the NGSIM database after merging

6.1.2 Results

In order to assess the quality of the obtained ranking, we have represented the curve precision-recall. Each point of the curve corresponds to the precision and recall for the first N answers to a given query, with N from 1 to the size of the test base. A possible measure of the ranking quality is the area under the curve (the average precision). Other interesting measures are the precision for a given number of answers, or the precision at which we reach a recall equal to one.

In Figure 30 we can see the curves for different queries. Three different coding matrices have been tested: *matrix1* is the one-against-all matrix (5 classifiers), *matrix2* is a matrix formed by 7 classifiers[32] and *matrix3* is the union of the classifiers from *matrix1* and *matrix2*. There is no temporal fusion of the images corresponding to a same track. We have also represented the curve for the previous ranking strategy in the laboratory (based in a one-against-all classification with all-pairs disambiguation).

It is interesting to note that the previous strategy does not rank all the images in the set, so in general it does not reach a recall equal to one. The remaining images would be ranked randomly. This does not happen with the new approach.

For pedestrians (Figure 30.a) and cars (Figure 30.b) the ranking was already excellent, and the new approach is roughly equivalent when not better. For trucks (Figure 30.c) the new approach outperforms the previous one. In particular, if we compare the previous with the new one-against-all matrix, we can see that with the same set of classifiers we can obtain much better results, thanks to the probabilistic approach and without the need of the disambiguating classifiers. It is also remarkable than matrix2 does not work better than matrix1 even if it has more classifiers, and even if it globally outperforms the previous strategy, the first errors occur in higher positions. Interestingly, matrix3 performance is roughly between that of matrices 1 and 2.

We have explored the influence of some aspects of the Hastie-Tibshirani method for coupling probability estimates. In particular, we have tested *matrix2* leaving more time to reach convergence. We have also tested *matrix2* introducing some confidence information. There is no remarkable difference in the results for pedestrians and cars. The results for trucks are shown in Figure 31.

[32] An optimized matrix was not generated because a lack of time.

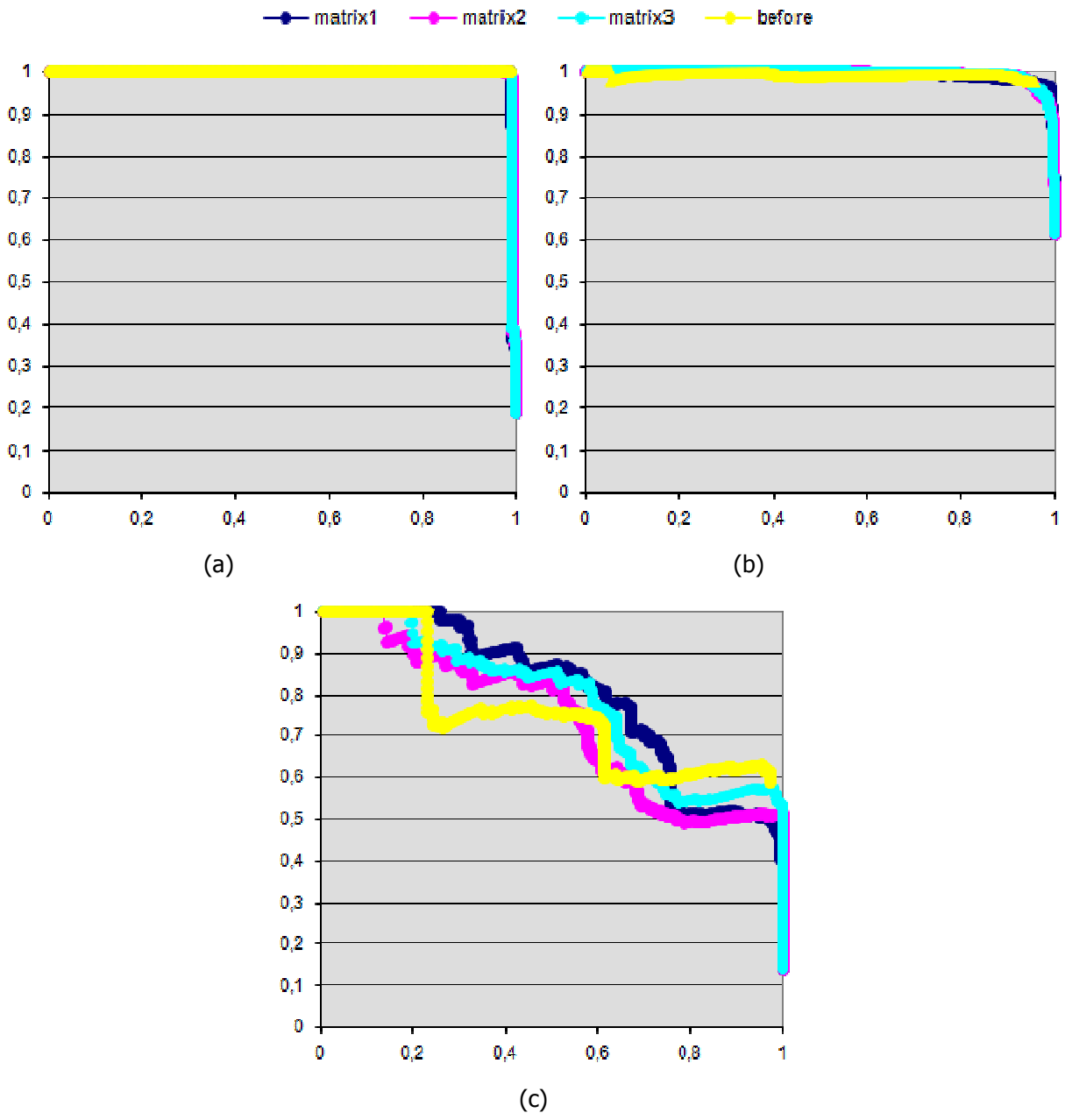
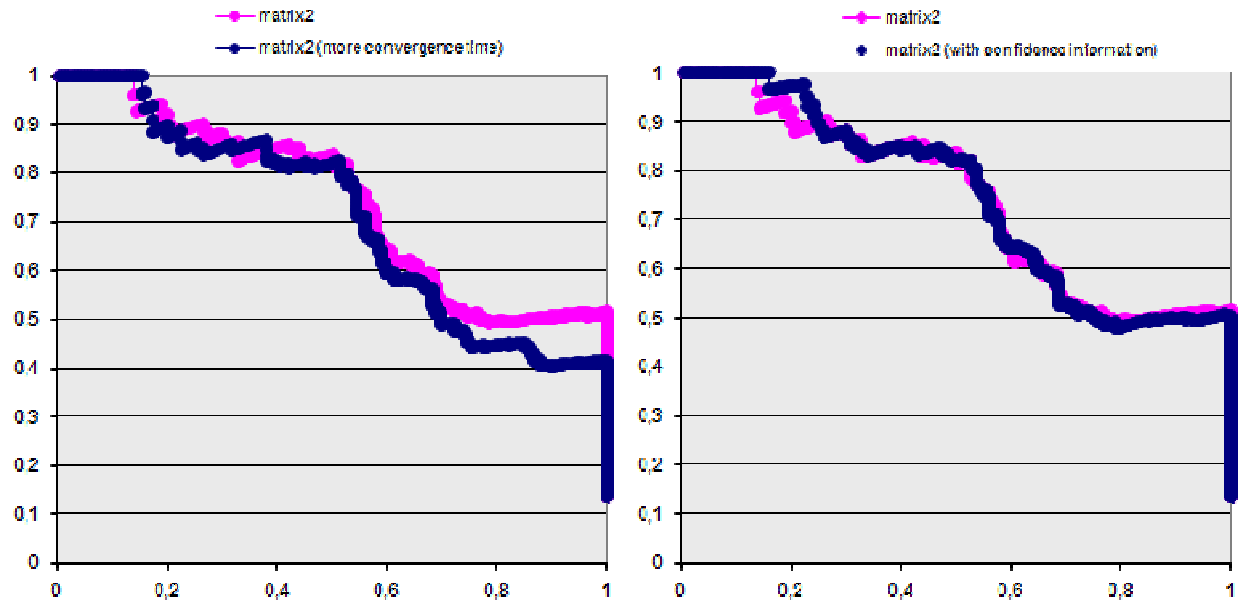


Figure 30. Ranking results on the NGSIM base for pedestrians (a), cars (b) and trucks (c) without temporal fusion.

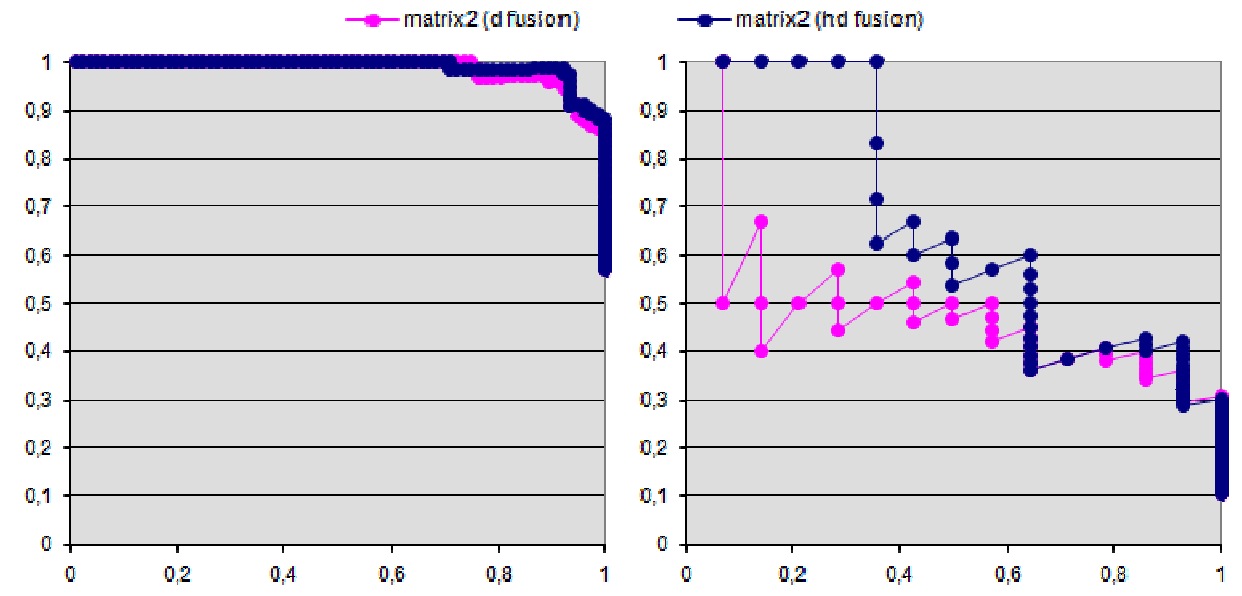
In the design phase we had already observed that modifying the stopping criterion was not critical for classification purposes. Here we can see that increasing the convergence time does not necessary improve performance. In fact, in this case it works worse for the lower ranking positions. On the other hand, the introduction of confidence information slightly improves the ranking for the higher positions in this case.



(a) (b)

Figure 31. Ranking results for trucks with perturbations in the coupling algorithm (without temporal fusion)

We now introduce the results with temporal fusion of the observations belonging to the same track. In Figure 32 we can see the difference between disjunctive fusion and weighted sum of hybrid and disjunctive, for cars and trucks (for pedestrians both kinds of fusion lead to perfect rankings).



(a) (b)

Figure 32. Ranking results for cars (a) and trucks (b) with two different kinds of temporal fusion.

In Figure 33 we compare once again the results for matrices 1, 2 and 3, this time with temporal fusion (weighted hybrid and disjunctive fusion). We show also the results for the previous strategy (the fused score is the mean of scores). The results for pedestrians are not shown because they were perfect with the four strategies.

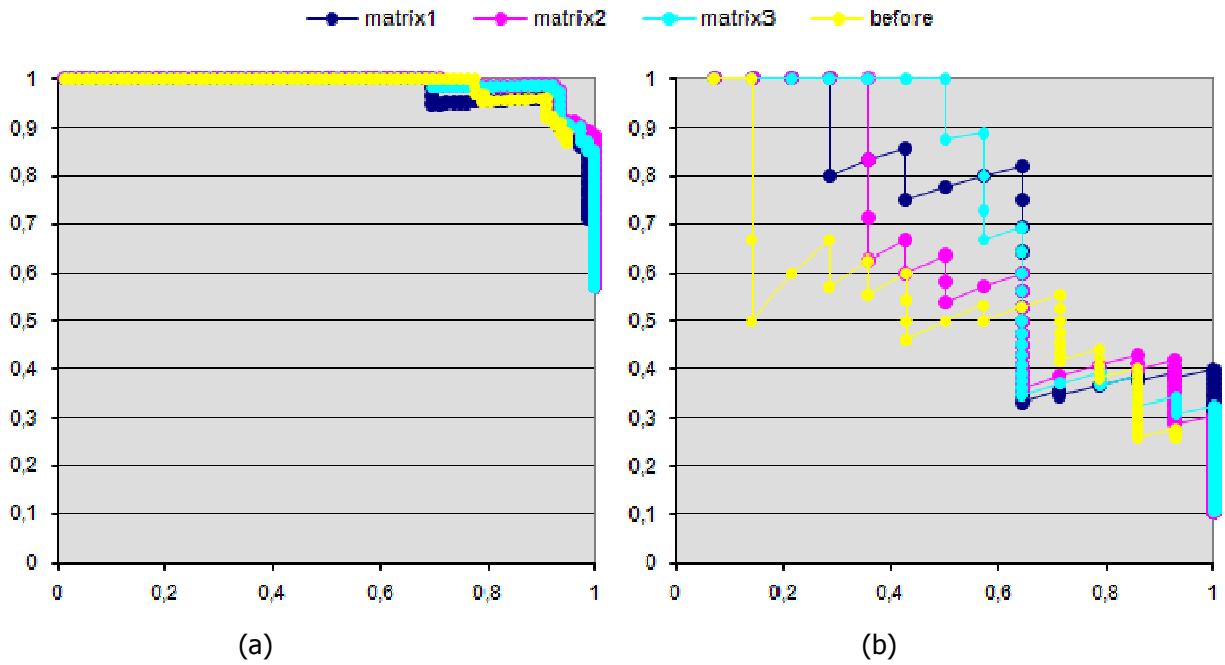


Figure 33. Ranking results on the NGSIM base for cars (a) and trucks (b) with temporal fusion.

This time, matrix2 outperforms matrix1 for cars. The results for trucks are less representative statistically, as there are few tracks in the base.

6.2 ISOLET database

6.2.1 Description

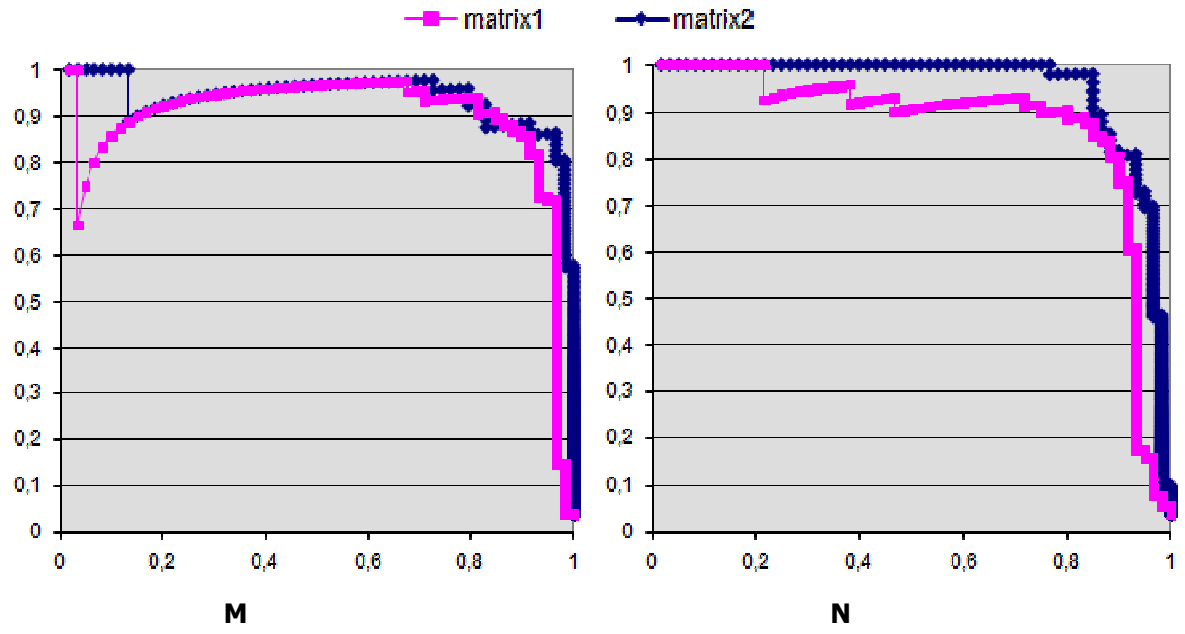
ISOLET is a database of letters of the English alphabet spoken in isolation. The database consists of 7797 elements, which are the 26 letters spoken by 150 subjects (each subject spoke the name of each letter twice, and three elements are missing). The speakers are grouped into sets of 30 speakers each, and are referred to as isolet1, isolet2, isolet3, isolet4, and isolet5. We have used isolet4 as the validation set, isolet5 as the test set and the rest as the training set. Each letter is represented by 617 numerical features. The features include spectral coefficients, contour features, sonorant features, pre-sonorant features, and post-sonorant features. Exact order of appearance of the features is not known.

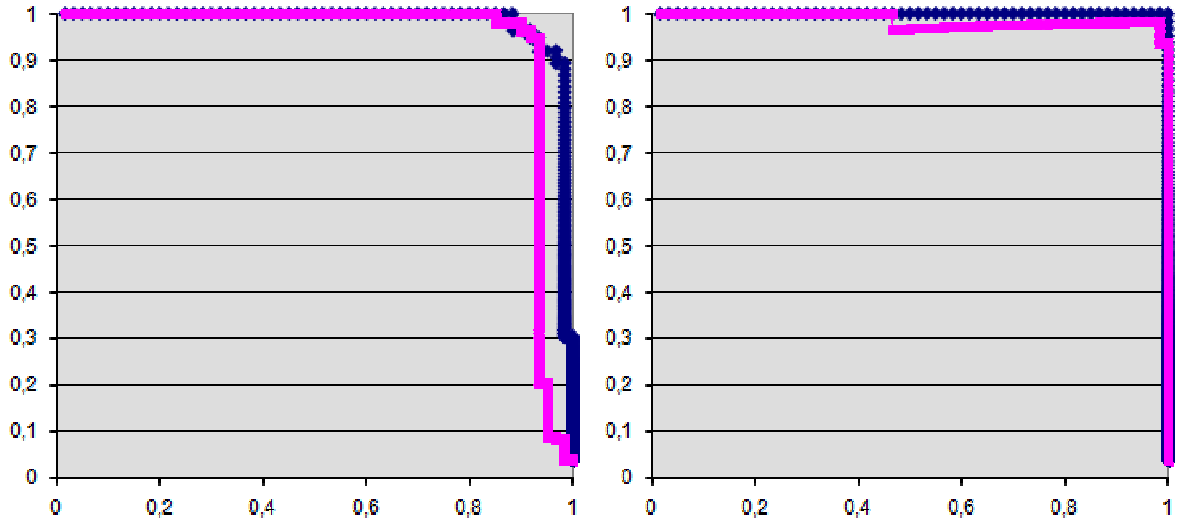
This database can be downloaded at [38].

It has been useful for verifying and improving the scalability of our system.

6.2.2 Results

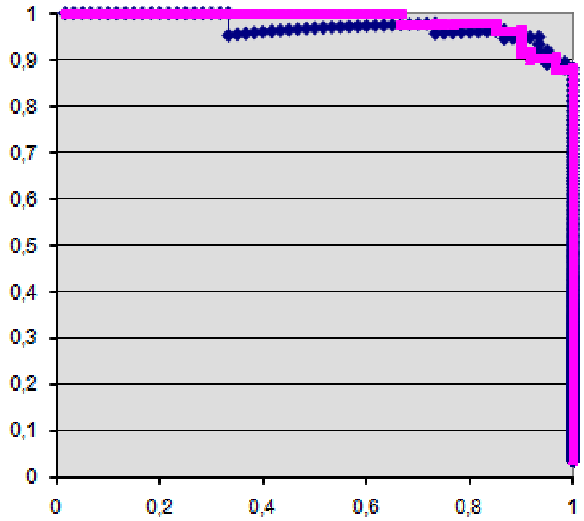
We have tested the one-against-all matrix (*matrix1*) with another matrix (*matrix2*) consisting of 39 classifiers including the 26 one-against-all classifiers. In this dataset there is no possible temporal fusion. We do not have any previous ranking results to compare with. The rankings are perfect or excellent for most of the letters. Here we show the worse cases (letters M and N) and also other groups of letters that tend to be occasionally confused (T-G and E-B-V).



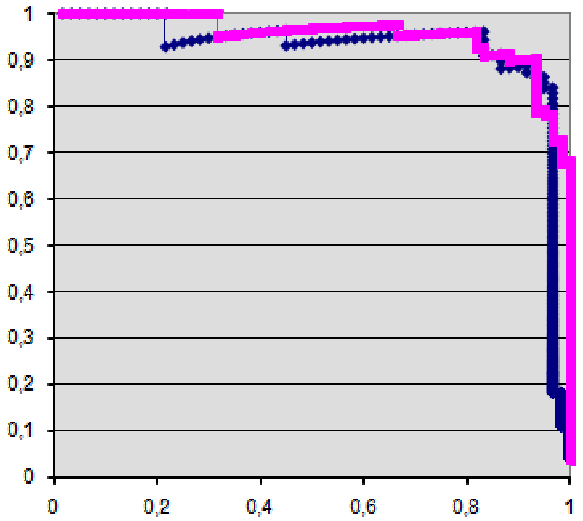


T

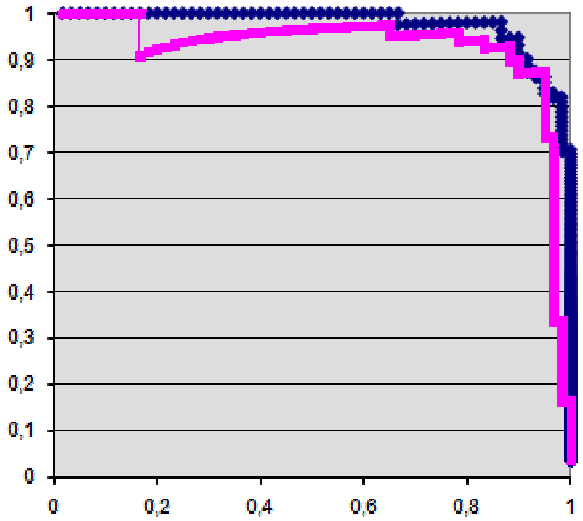
G



E



B



V

Figure 34. Ranking results on the ISOLET database for some letters.

6.3 Results analysis

In the NGSIM dataset, the comparison between matrix1 and the previous strategy (which is also based in the one-against-all set of classifiers) leads to think that the probabilistic approach introduced in this work (both calibration and coupling of probabilities) works better than the previous comparison of individual classifications.

With regards to the ECOC approach, is cannot be worse than the previous approach, as the one-against-all is a particular case. However, we have noticed the importance of an appropriate optimization method for the coding matrix, as the performance can be poor if the errors of the retained classifiers are too correlated. In particular, it is important to ensure that the selected matrix is at least as good as the one-against-all matrix (as it gives a reference of the reachable performance with a low number of classifiers). Sometimes, we have observed that one matrix is better than another one for a given class but it is the other way round for other classes (see Figure 34). It is difficult to examine the dependencies among all classes. Furthermore, it is not trivial to assess the quality of the ranking for all possible queries in order to obtain a single score. Besides, we have observed that the obtained ranking can be perfect even is the classification statistics for a given class are not. All these aspects could be taken into consideration to elaborate an appropriate criterion for the matrix evaluation during the optimization process.

The results shown in Figure 31.b makes us think that a more sophisticated way of computing the confidence of the single classifiers and a more appropriate way of using them in the coupling process could lead to better results, at least in the case of not too ambiguous examples, in which we can take advantage of the redundancy provided by the codewords. This redundancy allows to safely ignore some classifiers, if the correcting capability of the code is high enough. Thus, the obtained probability distribution could be more accurate.

More extensive databases should be tested in order to assess the capabilities of the new system.

7 Future work

There are many open questions concerning different aspects.

To begin with, it would be interesting to study the most appropriate way of splitting the available data into the training, validation and test sets. Concerning the binary classifiers, the kind of learning algorithm and the optimisation criteria should be validated.

In this work, a hypothesis about the operational environment (the expected distribution on the classes) is automatically taken into account in some phases of the system construction. It should be verified if this introduction of extra knowledge is appropriate. And if it is so, the question is how to estimate this distribution.

With regards to the matrix optimisation, it is worthy to focus on the heuristic method in order to improve it. In particular, it is interesting to study the choice of an appropriate evaluation criterion that fits the application needs. For instance, it might be relevant to substitute the current correct classification rate by some ranking-oriented measure. Another promising possibility is to investigate if the mutual information provided by the classifiers is an appropriate criterion to take into account when deciding which classifier should be deleted at each iteration.

As for calibration, further work can be done in order to find the best way of assessing the confidence in the probability estimate. Some axes of investigation have been already suggested in the corresponding section.

With respect to coupling, the stopping criterion does not seem to be critical. It seems more interesting to investigate how to use the confidence information provided by the previous step in order to improve the obtained probability distribution.

Regarding the temporal fusion, we have said that disjunctive and hybrid fusion can be combined to obtain different behaviours, but the optimal weight of each one depends on the data. For the moment, this weight is fixed. It could be possible to introduce a phase to statistically learn from examples a suitable weight.

We also leave an opened door to more sophisticated ways of transforming the probability distributions into ranking scores.

Other directions of work that differ more radically from the proposed one could be an approach not handling probabilities but other ways of representing and fusing information, or an approach not based on the reduction of the multiclass problem into binary problems, as briefly discussed in the end of section §5.3.1, but intrinsically multiclass.

8 Conclusions

We have developed a system that satisfies the specifications and is adapted to the laboratory's resources and needs.

The design is creative. It differs from the previous strategy in several fundamental aspects. We have introduced the probabilistic approach, by means of the calibration and the subsequent probabilistic fusion, inexistent in the previous strategy. The system is also a general solution, in the sense that we have expanded the possible binary classifiers to any suitable set, the one-against-all approach remaining a particular case. Moreover, it is scalable to a very high number of classes, the system definition is fully automatic, independent of the nature and properties of the data.

We have had a global vision of the problem. The design is ranking-oriented, and the obtained system is both elegant and coherent as a whole from a theoretical point of view, thanks to a strong formal framework.

This work did not only consist in putting suitable parts together. Each part has been worked in depth, from the theoretical and empirical point of view. Benchmarks have proven this whole framework to be efficient, easy to use and to adapt, and more accurate than previous solutions.

The result is a system that provides to the laboratory a complete and automatic multiclass system with a significant gain of performances, in the field of multiclass classification, compared to former implementations.

References

1] R.S. MICHALSKI, J.G. CARBONELL and T.M. MITCHELL. **An overview of Machine Learning**. *Machine Learning: an Artificial Intelligence approach*. Tioga Publishing Company. 1983.

2] F. CAMASTRA and A. VINCIARELLI. **Machine Learning for audio, image and video analysis**. Springer. 2008.

3] Wikipedia. **Artificial Neural Network**.

4] W.S. SARLE. **Neural Network FAQ**. 1997. (ftp://ftp.sas.com/pub/neural/FAQ.html)

5] C. J. C. BURGES. **A tutorial on Support Vector Machines for pattern recognition**. 1998.

6] R.E. SCHAPIRE. **The boosting approach to machine learning: an overview**. 2002.

7] Y. FREUND and R.E. SCHAPIRE. **A short introduction to boosting**. 1999.

8] Wikipedia. **Hierarchical Temporal Memory**.

9] Wikipedia. **Memory-prediction framework**.

10] J. HAWKINS and D. GEORGE. **Hierarchical Temporal Memory: concepts, theory and terminology**. 2006.

11] Vulgarisation and technical documentation in the Numenta web site:
http://www.numenta.com/about-numenta/numenta-technology.php
http://www.numenta.com/for-developers/education/htm-summary.php
http://www.numenta.com/for-developers/software/prog_guide/Output/TOC.html

12] R. CARUANA and A. NICULESCU-MIZIL. **An empirical comparison of supervised learning algorithms**. 2006.

13] B. ZADROZNY and C. ELKAN. **Transforming classifier scores into accurate probability estimates**. 2002.

14] E.L. ALLWEIN, R.E. SCHAPIRE and Y. SINGER**. Reducing Multiclass to binary: A unifying approach for margin classifiers**. 2000.

15] K. SIEGRIST. *Virtual Laboratories in Probability and Statistics*.
(http://www.math.uah.edu/stat/ and http://www.ds.unifi.it/VL/VL_EN/index.html)
Random Samples: Relative Frequency and Empirical Distributions
(http://www.ds.unifi.it/VL/VL_EN/sample/sample3.html)
Interval estimation: Estimation in the Bernoulli Model
(http://www.ds.unifi.it/VL/VL_EN/interval/interval4.html)

16] W. HAERDLE. *Applied Nonparametric Regression*
(http://fedc.wiwi.hu-berlin.de/xplore/ebooks/html/anr)
Monotonic and unimodal smoothing
(http://fedc.wiwi.hu-berlin.de/xplore/ebooks/html/anr/anrhtmlnode43.html)

17] T. WINDEATT and R. GHADERI. **Coding and Decoding strategies for multi-class learning problems**. 2002.

18] K. TUMER and J. GOSH. **Error correlation and error reduction in ensemble classifiers**. 1996.

19] V. GURUSWAMI and A. SAHAI. **Multiclass learning, boosting and error-correcting codes**, 1999.

20] T.G. DIETTERICH and G. BAKIRI. **Solving multiclass learning problems via error-correcting output codes**, 1995.

21] E.G. KONG and T.G. DIETTERICH. **Error-correcting output coding corrects bias and variance**, 1995.
22] E.G. KONG and T.G. DIETTERICH. **Probability estimation using error-correcting output coding**, 1997.
23] B. ZADROZNY. **Reducing multiclass to binary by coupling probability estimates**. 2002.
24] T. HASTIE and R. TIBSHIRANI. **Classification by pairwise coupling**, 1998.
25] Wikipedia. **Kullback-Leiber divergence**.
26] K. CRAMMER and Y. SINGER. **On the Learnability and Design of Output Codes for Multiclass Problems**, 2000.
27] O. DEKEL and Y. SINGER. **Multiclass Learning by Probabilistic Embeddings**, 2002.
28] H. ZOU, J. ZHU and T. HASTIE. **The Margin Vector, Admissible Loss and Multi-class Margin-based Classifiers**, 2005.
29] Wikipedia. **Polynomial code**.
30] Wikipedia. **BCH code**.
31] I. BLOCH**. Fusion d'informations en image et vision**. 2006
32] D. MERCIER. **Le Modèle des Croyances Transférables : Une interprétation de la théorie des fonctions de croyanc**e. 2007.
33] T. DENŒUX. **Théorie des Fonctions de Croyance et Classification**. 2004.
34] M.C. FLOREA, A. JOUSSELME, E. BOSSE and D. GRENIER. **Robust combination rules for evidence theory**, 2008.
35] S. MASKELL. **A Bayesian approach to fusing uncertain, imprecise and conflicting information**, 2007
36] E. PIAT and D. MEIZEL. **Proposition d'un cadre probabiliste de fusion de croyances**, 1997.
37] Wikipedia**. Cox's Theorem** (http://en.wikipedia.org/wiki/Cox%27s_theorem)
38] **ISOLET database** http://archive.ics.uci.edu/ml/datasets/ISOLET