INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Binary Feature Selection with Conditional Mutual Information

François Fleuret

N° 4941

Octobre 2003

THÈME 3

ISSN 0249-6399 ISRN INRIA/RR--4941--FR+ENG

Binary Feature Selection with Conditional Mutual Information

François Fleuret

Thème 3 — Interaction homme-machine,
images, données, connaissances
Projet Imedia

Rapport de recherche n° 4941 — Octobre 2003 — 16 pages

Abstract: In a context of classification, we propose to use conditional mutual information to select a family of binary features which are individually discriminating and weakly dependent. We show that on a task of image classification, despite its simplicity, a naive Bayesian classifier based on features selected with this Conditional Mutual Information Maximization (CMIM) criterion performs as well as a classifier built with AdaBoost. We also show that this classification method is more robust than boosting when trained on a noisy data set.

Key-words: classification, feature selection, Bayesian classifier, mutual information

Unité de recherche INRIA Rocquencourt
Domaine de Voluceau, Rocquencourt, BP 105, 78153 Le Chesnay Cedex (France)
Téléphone : +33 1 39 63 55 11 — Télécopie : +33 1 39 63 53 30

Sélection de descripteurs par maximisation de l'information mutuelle conditionnelle

Résumé : Dans un contexte de classification, nous proposons d'utiliser l'information mutuelle conditionnelle pour sélectionner une famille de descripteurs binaires qui sont individuellement informatifs tout en étant faiblement dépendants entre eux. Nous montrons sur un problème de classification d'images que malgré sa simplicité un classifieur de type Bayésien naïf utilisant des descripteurs sélectionnés de cette manière obtient des taux d'erreur similaires à ceux d'un classifieur construit à l'aide d'AdaBoost. Nous montrons également que cette technique est beaucoup plus robuste que le boosting dans un cadre bruité.

Mots-clés : classification, sélection de features, classifieur bayésien, information mutuelle

1 Introduction

By reducing the number of considered features, one can both reduce over-fitting of learning methods, and increase the computation speed of prediction [GE03]. We focus in this paper on the selection of a few tens of binary features among a set of several tens of thousands of them in a context of classification.

The most standard ways to select features consist in ranking them according to their individual predictive power, which is estimated by various methods such as Fisher score [FCD+00], Kolmogorov-Smirnov test, Pearson correlation [MP00] or mutual information [Bat94, BW96, Tor03]. Selection based on such a ranking does not ensure a good orthogonality between features, and can lead to redundant and thus less informative selected families.

Our approach consists in picking features which maximize their mutual information with the class to predict, conditionally to the response of any feature already picked. This Conditional Mutual Information Maximization criterion (CMIM) does not select features similar to already picked ones, even if they are individually powerful, as they do not carry additional information about the class to predict. Thus, it ensures a good tradeoff between redundancy and discrimination.

Experiments on a face vs. non-face classification task demonstrate that features chosen according to this criterion can be efficiently combined with a naive Bayesian approach [DH73, LIT92] and lead to error rates similar to those obtained with AdaBoost [FS96a]. Also, experiments show the robustness of this method with respect to noisy training sets, as it achieves better results than regularized AdaBoost, even though it does not require the tuning of a regularization parameter.

In §2 we summarize the context of this work, we introduce the notations for the rest of the paper and we give a short summary of standard classification techniques. We quickly present tools from information theory and describe our feature-selection scheme in §3. We give results of experiments – mainly comparisons with AdaBoost – in §4 and §5, and we finally discuss those results and the forthcoming developments of this work in §6.

People in a hurry who are familiar with the topic can jump directly to section §2.1 (page 3) for the notations, §3.2 (page 7) for the description of the CMIM algorithm, and to tables 1 and 2 (pages 11 and 14) for error rates and comparison with AdaBoost.

2 Classification

2.1 Notations

Because this work was originally motivated by the design of a face detector, our experiments are based on a classification of face vs. non-face images. For that reason, we introduce notations related to image classification for the sake of understanding, but our approach is generic and can be applied to any set of binary features. We give here a rough description of the experimental settings (images, features, etc.) but we go into details in §4.

Let $\mathcal{I}$ denote the set of 28×28 pixel grayscale images, and X a random variable on $\mathcal{I}$ standing for the distribution of images. We denote Y a boolean random variable for the class to predict, the value 0 standing for *non-face* and 1 for *face* (cf. §4.1).

We consider a set of boolean features $f_1, \ldots, f_N$, which are mappings from $\mathcal{I} \to \{0, 1\}$. The number of features N is pretty large, of the order of tens of thousands. In the experiments those features are boolean functions whose values depend with the presence or absence of edges at certain position in the picture. Each of them is thus defined by its location in the 28×28 picture frame, its orientation and its tolerance (cf. §4.2).

We denote $F_1 = f_1(X), \ldots, F_N = f_N(X)$ the boolean random variables standing for the responses of the features. We will often make a confusion between the features as mappings and as random variables (i.e. between the f_i and the F_i).

Finally, we denote $f_{n(1)}, \ldots, f_{n(M)}$ the selected features, where M is the number of features we select, and is very small compared to N (of the order of fifty).

All statistical estimations during training and testing are based on samples of a few hundreds of pictures, labeled by hand with their real classes (x_1, y_1), ..., (x_T, y_T) (cf. §4.1). Those sets contain as many face and non-face pictures, which correspond to an equilibrated prior $P(Y = 1) \simeq 1/2$. We will use them implicitly for all the empirical estimations during training or testing.

2.2 Perceptron and naive Bayesian classifier

Given a subset of features $f_{n(1)}, \ldots, f_{n(M)}$ already selected, we consider linear decision rules, which depend on the sign of an expression of the form:

$$f(x) \quad = \quad \sum_{i=1}^{M} \omega_i \, f_{n(i)}(x) \; + \; b$$

We have used two algorithms to estimate the $(\omega_1, \ldots, \omega_M)$ and b from the training set. The first one is the classical perceptron [Ros58, Nov62] and the second one is the naive Bayesian classifier [DH73, LIT92].

Perceptron

In a nutshell, the perceptron learning scheme consists in estimating iteratively the vector $(\omega_1, \ldots, \omega_M)$ by correcting it as long as training examples are misclassified. More precisely, as long as there exists a misclassified example (y_i, x_i) its feature vector is added to the normal vector if it is of class positive, and is otherwise subtracted:

$$\forall k \quad \omega_k \quad \leftarrow \quad \omega_k \; + \; (2\, y_i \; - \; 1)\, f_{n(k)}(x_i)$$

The bias term b is computed by considering a constant feature always equal to 1. If the training set is linearly separable in the feature space, the process is proved to converge to

a separating hyperplane and the number of iterations can be easily bounded (see [CST00] pages 12-14). If the data are not separable, the process never ends, and has to be terminated after a number of iterations fixed a priori.

Naive Bayesian

The naive Bayesian classifier is a simple likelihood ratio test with an assumption of conditional independence between the features. The predicted class depends on the sign of:

$$f(x) \quad = \quad \ln \frac{\hat{P}(Y=1 \,|\, F_{n(1)} = f_{n(1)}(x), \ldots, F_{n(K)} = f_{n(K)}(x))}{\hat{P}(Y=0 \,|\, F_{n(1)} = f_{n(1)}(x), \ldots, F_{n(K)} = f_{n(K)}(x))}$$

Under the assumption that the $F_{n(.)}$ are conditionally independent, given Y, we have :

$$\begin{aligned}
f(x) &= \ln \frac{\hat{P}(F_{n(1)} = f_{n(1)}(x), \ldots, F_{n(K)} = f_{n(K)}(x) \,|\, Y=1)}{\hat{P}(F_{n(1)} = f_{n(1)}(x), \ldots, F_{n(K)} = f_{n(K)}(x) \,|\, Y=0)} \; + \; a \\
&= \ln \frac{\prod_{i=1}^{K} \hat{P}(F_{n(i)} = f_{n(i)}(x) \,|\, Y=1)}{\prod_{i=1}^{K} \hat{P}(F_{n(i)} = f_{n(i)}(x) \,|\, Y=0)} \; + \; a \\
&= \sum_i \ln \frac{\hat{P}(F_{n(i)} = f_{n(i)}(x) \,|\, Y=1)}{\hat{P}(F_{n(i)} = f_{n(i)}(x) \,|\, Y=0)} \; + \; a \\
&= \sum_i \left\{ \ln \frac{\hat{P}(F_{n(i)} = 1 \,|\, Y=1)}{\hat{P}(F_{n(i)} = 1 \,|\, Y=0)} f_{n(i)}(x) + \ln \frac{\hat{P}(F_{n(i)} = 0 \,|\, Y=1)}{\hat{P}(F_{n(i)} = 0 \,|\, Y=0)} (1 - f_{n(i)}(x)) \right\} \; + \; a \\
&= \sum_i \left\{ \ln \frac{\hat{P}(F_{n(i)} = 1 \,|\, Y=1)}{\hat{P}(F_{n(i)} = 1 \,|\, Y=0)} \frac{\hat{P}(F_{n(i)} = 0 \,|\, Y=0)}{\hat{P}(F_{n(i)} = 0 \,|\, Y=1)} \right\} f_{n(i)}(x) \; + \; b
\end{aligned}$$

Thus we finally obtain a simple expression for the coefficients:

$$\omega_i \quad = \quad \ln \frac{\hat{P}(F_{n(i)} = 1 \,|\, Y=1)}{\hat{P}(F_{n(i)} = 1 \,|\, Y=0)} \frac{\hat{P}(F_{n(i)} = 0 \,|\, Y=0)}{\hat{P}(F_{n(i)} = 0 \,|\, Y=1)}$$

The bias b is related to the prior $P(Y=1)$ and can be estimated given the ω_i to minimize the error rate on the training set.

Remark

Those two classifiers have different strengths and weaknesses. The perceptron takes into account the joint statistics of the features, and deals with outliers by increasing their influence in the final orthogonal vector. On the contrary, the naive Bayesian classifier is based

on estimates of the conditional marginal probabilities and is more robust to the presence of outliers. This means that while the perceptron is able to deal with dependent features, it suffers more from overfitting than the naive Bayesian.

2.3 AdaBoost

The idea of boosting is to select and combine several classifiers (often referred to as *weak-learners*, as they achieve individually high error rate) into a more global one with a voting procedure. In our case, the features are considered as weak-learners. Thus, the training of the weak-learner consists simply in picking the features with the minimum error rate. We also consider for each feature its anti-feature (i.e. the one which responds the opposite).

The selection is done by maintaining a distribution on the training examples which accumulates on the misclassified ones during the training. At iteration k, the weak learner $f_{n(k)}$ which minimizes the weighted error rate is selected, and the distribution is refreshed to increase the weight of the misclassified samples and reduce the importance of the others. Note that boosting can be seen as a functional gradient descent [Bre00, MBBF00, FHT00] in which each added weak learner is a step in the space of classifiers.

In our comparisons, we have used the original AdaBoost procedure [FS96a, FS96b], which is known to suffer from overfitting. For noisy tasks, we have chosen a soft-margin version called AdaBoost$_{reg}$ [ROM98]. It regularizes the classical AdaBoost by penalizing samples which influence too heavily the training, as they are usually outliers.

To use boosting as a feature selector, we run the algorithm to accumulate the expected number of features, and we then recompute the weightings with another procedure (perceptron or naive Bayesian).

3 Feature selection based on conditional mutual information

3.1 Information theory tools

Information theory provides intuitive tools to quantify how much information is required to describe random quantities, or how much information is shared by a few of them [CT91]. We consider here only finite random variables and we denote U, V and W three of them.

The most fundamental concept in information theory is the entropy $H(U)$ of a random variable. Roughly, it quantifies the average number of bits required to encode or describe the value of U. A deterministic variable has a null entropy (as nothing is required to describe its value, which is known by advance), while a uniform distribution on $1, \ldots, 2^l$ has an entropy of l. If a distribution is unbalanced, it is more efficient to use less bits to code frequent values, even if that force to use more bits for rare values. It can be shown that

$$H(U) \quad = \quad -\sum_u \log_2 \left(P(U=u)\right) P(U=u)$$

The conditional entropy $H(U \mid V) = H(U, V) - H(V)$ quantifies the average number of bits required to describe U, when the value of V is already known. For instance, if U is a deterministic function of V, then this conditional entropy is null, as no more information is required to describe U when V is known. On the contrary, if they are independent, knowing V does not tell you anything about U and the conditional entropy is almost equal to the entropy itself.

Our feature selection is based on the conditional mutual information:

$$I(U, V \mid W) \quad = \quad H(U \mid W) - H(U \mid W, V)$$

This value quantifies how much information is shared between U and V, given the value of W. Another way to see it, as it it decomposed above, is as the difference between the information required to describe U given W, and the information to describe U given both W and V. If V and W carry the same information about U, the two terms on the right are equal, and the conditional mutual information is zero. On the contrary if both V and W bring information, and if those information are complementary, the difference is large.

3.2 Conditional Mutual Information Maximization

The main goal of feature selection is to select a small subset of features that carries as much information as possible. The ultimate goal would be to minimize $\hat{H}\left(Y \mid F_{n(1)}, \ldots, F_{n(M)}\right)$. But this expression can not be estimated with a training set of realistic size as it requires the estimation of 2^{M+1} probabilities. Furthermore, even if there were ways to have an estimation, its minimization would be computationally untractable.

At the other extreme, one could do a trivial random sampling which would ensure to some extent independence between feature (if different types of features are equally represented) but would not at all take care of predictive power. To deal with that, one could select the best features according to some estimate of their individual predictive power. The main weakness of this approach is that although it takes care of individual predictive power, it does not avoid at all redundancy among the selected features. Basically, one would pick many similar features, as the ones carrying a lot of information are likely to be of a certain type. For a face detection task for instance, edges on the eyebrows and the mouth would be the only ones competitive.

We propose an intermediate solution. Our approach deals with the tradeoff between individual power and independence by comparing each new feature with the ones already picked. We say that a feature $F^\star$ is good only if $\hat{I}(Y, F^\star \mid F)$ is high for every F already picked. This means that $F^\star$ is good only if it carries information about Y, and if this information has not been caught by any of the F already picked. More formally, we propose the following iterative scheme:

$$n(1) \quad = \quad \arg\max_i \hat{I}(Y, F_i)$$

$$n(k+1) \quad = \quad \arg\max_i \left\{ \min_{l \leq k} \hat{I}(Y, F_i \,|\, F_{n(l)}) \right\}$$

By taking the minimum of the conditional mutual information on all features already picked, we ensure that a new feature is both informative and different than the preceding ones. The computation of those scores can be done accurately as they require only the estimation of distributions of triplets of boolean variables.

3.3 Mutual Information Maximization

To compare our approach to selection based on individual prediction power, we have also implemented a method which picks the K features maximizing individually their mutual information with the class to predict $\hat{I}\left(Y, F_{n(l)}\right)$. In the result sections, we call this method MIM for Mutual Information Maximization.

3.4 Complexity

The computational costs of CMIM and AdaBoost are equivalent when the set of all available features is finite and remain identical at all iterations. We still denote here N the total number of features, M the number of features we want to select, and T the size of the training set.

Both boosting and CMIM have a main loop repeated M times, which picks feature one after another. For each iteration of this loop, AdaBoost requires to compute a weighted error rate for each one of the N available features, which costs $O(N \times T)$ operations. Learning with CMIM requires to compute the $\hat{H}\left(Y \,|\, F_{n(k)}, F_i\right)$, for every $1 \leq i \leq N$ where k is the index of the last selected feature. This is true as long as we can keep a table $s[i] = \min_{l \leq k} \hat{I}(Y, F_i \,|\, F_{n(l)})$ with the current minimum value of the conditional mutual information for every weak feature of the complete set. Such a computation require also $O(N \times T)$ operations. Finally, both methods cost is $O(M \times N \times T)$.

4 Experiments

4.1 Data sets

We have used training and test sets extracted from two large sets of pictures. Those original big sets were assembled by collecting a few thousands scenes from the web and marking by hand the locations of eyes and mouth on every visible face. From every face we generate ten small grayscale face images by randomizing its pose. This lead to tens of thousands face pictures. We have also collected complex scenes (forests, buildings, furnitures, etc.) from which we have automatically extracted tens of thousands of background (non-face) pictures. All those pictures are of size 28×28 pixels, and quantified in 256 grayscale levels. Faces have been registered roughly, so that the center of the eyes is in a 2×2 central square, the

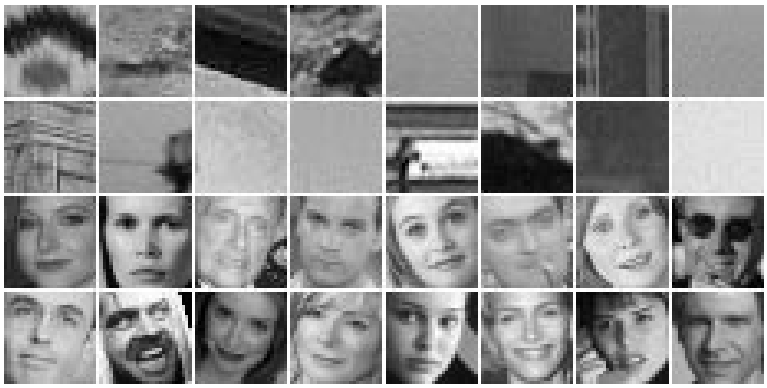

Figure 1: The two upper rows show examples of background pictures, and the two lower rows show examples of face pictures. All those images are grayscale of size 28×28 pixels, extracted from complete pictures taken on the WWW. Faces are roughly centered and standardized in size.

distance between the eyes is 10 to 12 pixels and the tilt is between -10 and $+10$ degrees (cf. figure 1).

For each experiment both the training and the test sets contain 500 images, roughly as many of which are faces and non-faces.

4.2 Edge features

We use features similar to the edge detectors in [FG01, FG02]. They are easy to compute and robust to illumination variations. Each feature is a boolean function indexed by a location (x, y) in the 28×28 reference frame of the image, a direction d which can take 8 different values (cf. figures 2 and 3) and a tolerance t which is an integer value between 1 and 7. The tolerance corresponds to the size of the neighborhood where the edge has to be present for the feature to be equal to 1 (cf. figure 2). Hence, we have a set of $28 \times 28 \times 8 \times 7 = 43,904$ features.

4.3 Training and testing

During the training of the perceptron, coefficients are not bound and the number of training loops is limited to 100 iterations in the non-separable case. The coefficients of the naive Bayesian are computed directly as described in 2.2.

Error rates are averaged on 25 rounds for each experiment. For each round, the training set and the test set are extracted randomly from the original large data sets (cf. §4.1).

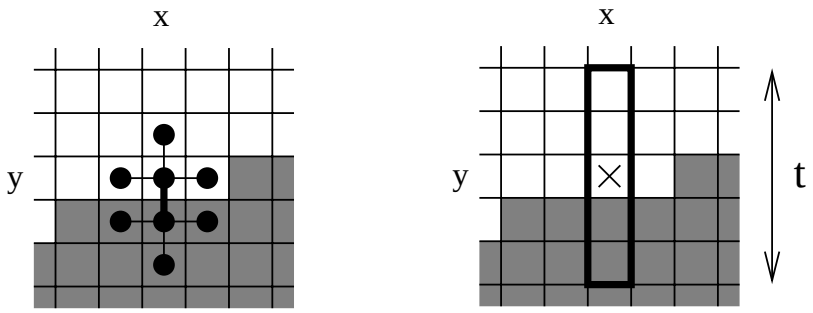

Figure 2: The edge detectors we are using are crude but invariant to changes in illumination. The picture on the left shows the criterion for a horizontal edge located in (x, y). The detector responds if the six differences between pixels connected by a thin segment are lesser in absolute value than the difference between the pixels connected by the thick segment. The relative values of the two pixels connected by the thick line define the polarity of the edge (dark to light or light to dark). The picture on the right shows the neighborhood where the edge has to be detected for a feature of tolerance $t = 5$ to respond.

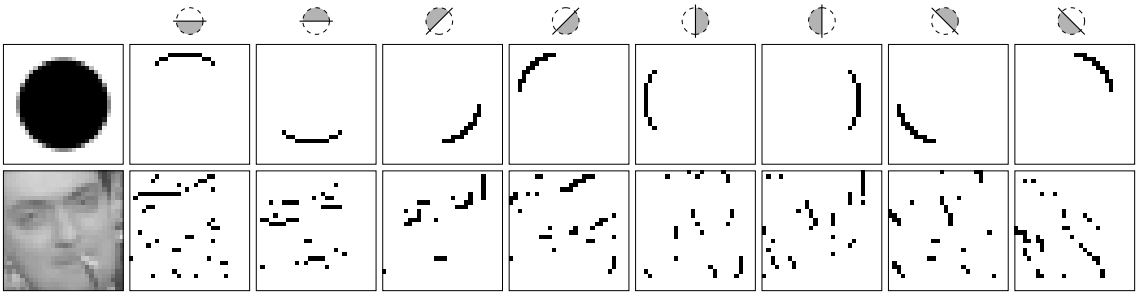

Figure 3: The original grayscale pictures are shown on the left. The eight binary maps on the right show the responses of the edge detectors at every locations in the 28×28 frame, for every one of the 8 possible directions and polarities.

Classifier	Training error	Test error
AdaBoost	0%	1.45%
CMIM feature selection + Bayesian	0.52%	1.52%
CMIM feature selection + perceptron	0%	2.28%
AdaBoost feature selection + perceptron	0%	2.46%
AdaBoost feature selection + Bayesian	0.4%	3.51%
MIM feature selection + perceptron	3.56%	8.28%
MIM feature selection + Bayesian	5.58%	8.54%

Table 1: Error rates with 50 features on the accurate training set.

5 Results

5.1 Experiments with accurate training sets

The first round of experiments was designed to compare the performances of our methods with two other feature selection methods (AdaBoost used as a feature selector, and MIM which is a simple selection based on individual mutual information cf. §3.3) when the features are combined with a perceptron. The results (cf. figure 4 and table 1) show that as a feature selector, AdaBoost does not perform better than our method. But if we use the weights computed by AdaBoost itself, the global classifier outperforms ours.

In the second round we combine the features with a naive Bayesian classifier, instead of a perceptron. This experiment shows that features selected by the CMIM lead to a naive Bayesian as powerful as the AdaBoost classifier (cf. figure 5 and table 1).

Those results are consistent. The perceptron suffers from its simplicity: as soon as the null error rate is obtained on the training set, it stops, leading to worst error rate than what maximum margin approaches could do. AdaBoost goes on maximizing the responses on training sample even when the null error rate is reached on the training set. This maximizes the margin between the classification boundary and the training samples and generalizes better. The result with the AdaBoost + naive Bayesian are consistent too, considering that AdaBoost does not take care of the independence between features.

The very bad performance of the MIM criterion can be explained by looking at the chosen features. As expected they are highly similar both in locations and direction, they are basically all detecting the upper part of the skull.

5.2 Experiment with noisy training sets

To test the robustness of the combination of CMIM and the naive Bayesian, we did a third round of experiments with noisy training data, known to be difficult for boosting schemes. We generated this training set by flipping at random 5% of the training labels. It creates a difficult situation for learning methods which take care of outliers, as there is 5% of them, distributed uniformly among the training population. Results are summarized on figures 6 and table 2. Both CMIM + perceptron and AdaBoost perform very badly, as

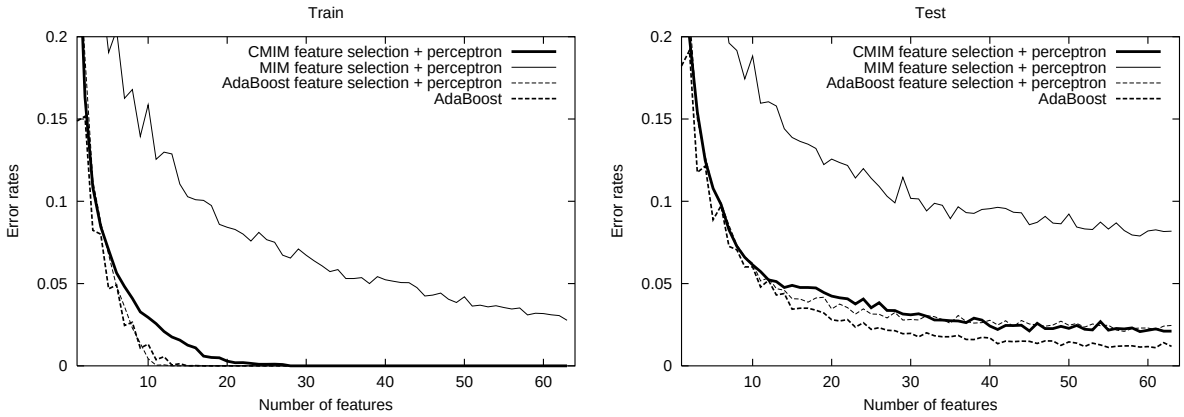

Figure 4: The performances of the two types of feature selections combined with a perceptron training. The curves show the error rates vs. the number of considered features. The error on the training set is shown on the left and the errors on the test set on the right. A perceptron built with the features selected by CMIM (thick continuous line) performs as well as a perceptron based on the features chosen by AdaBoost (thin dash line). Still, the pure AdaBoost reaches a better error rate (thick dash line).

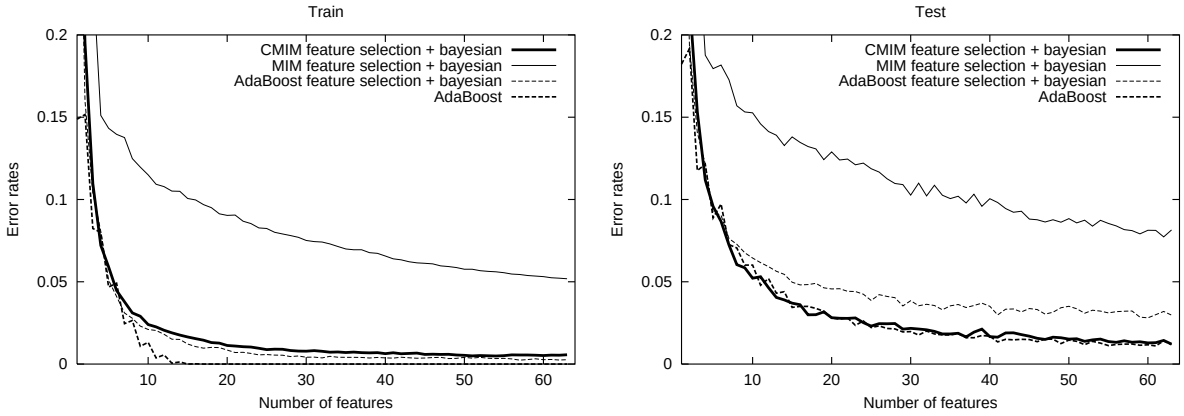

Figure 5: The performances of the two types of feature selections combined with a naive Bayesian classifier. The curves show the error rates vs. the number of considered features. The error on the training set is shown on the left and the errors on the test set on the right. The naive Bayesian based on the features selected by CMIM (thick continuous line) performs as well as pure AdaBoost (thick dash line), and better than a naive Bayesian using the features chosen by AdaBoost (thin dash line).

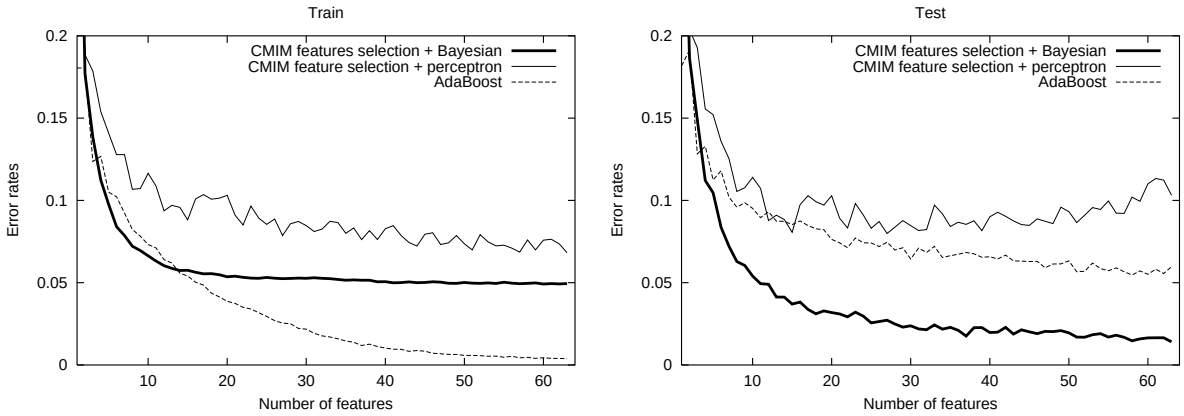

Figure 6: Comparison of performances with a noisy training set. The curves show the error rates vs. the number of considered features for a training set whose labels have been flipped with a probability 5%. The error on the training set is shown on the left and the errors on the test set on the right. As expected, AdaBoost (dash line) overfits and reaches a null error rate on the training set and a high error rate on the test set. The perceptron based on the CMIM features (continuous thin line) does bad both on training and testing, but the same features combined with a naive Bayesian (continuous thick line) demonstrate a robust behavior.

they are heavily influenced by pathological training points. On the contrary CMIM + naive Bayesian remains almost as efficient as with the original training set. The bad performance of AdaBoost as a feature selection scheme combined with the naive Bayesian demonstrates that naive Bayesian alone can not handle the flipped training labels.

To be fair with boosting, we made a fourth round of experiments with AdaBoost$_{reg}$. The experiment is the same as before, but is repeated for various values of the regularization parameter C. The error rates given in table 2 correspond to the C leading to the best test error rate.

6 Conclusion

We have presented in this article a simple scheme for binary feature selection in a context of classification. Combined with a naive Bayesian classifier, those features lead to performances as good as those obtained with the classical AdaBoost on clean data, and to better performances on noisy data, even compared to a regularized AdaBoost which requires the tuning of a regularization parameter.

The strong points of CMIM are:

Classifier	Training error	Test error
CMIM feature selection + Bayesian	5.06%	1.95%
AdaBoost$_{reg}$ (optimal)	3.8%	3.06%
AdaBoost	0.58%	6.33%
MIM feature selection + Bayesian	9.47%	8.59%
CMIM feature selection + perceptron	7.36%	9.32%
AdaBoost feature selection + Bayesian	10.28%	9.46%
MIM feature selection + perceptron	11.53%	13.12%

Table 2: Error rates with 50 features on the noisy training set whose labels have been flipped with probability 5%.

- It is robust, as it requires only estimations of distributions of triplets of boolean variables ;
- it provides a good tradeoff between redundancy and individual power of selected features;
- it selects features that can be combined with the naive Bayesian classifier, which is an explicit and robust scheme.

Nevertheless, CMIM suffers from three main drawbacks:

- It is usable only with a finite set of binary features, and restricted to classification ;
- it works under the assumption that dependencies between features can be caught by looking at couples of variables, thus ignoring dependencies between triplets or larger families of features ;
- its complexity can be an issue for very large families of features and training sets.

Note that AdaBoost suffers from similar weaknesses: it works only if dependency between the errors of the classifier built so far and at least one of the weak learner can be spotted, and its complexity is similar.

The forthcoming works will address some of those issues. For instance, the approach could probably be extended from binary features and classification to continuous features and regression by considering parametric density models.

References

[Bat94] R. Battiti. Using mutual information for selecting features in supervised neural net learning. In *IEEE Transactions on Neural Networks*, volume 5, 1994.

[Bre00] L. Breiman. Some infinity theory for predictors ensembles. Technical Report 579, Department of Statistics, University of California, Berkeley, 2000.

[BW96] B.V. Bonnlander and A.S. Weigend. Selecting input variables using mutual information and nonparametric density estimation. In *Proceedings of ISANN*, 1996.

[CST00] Nello Christiani and John Shawe-Taylor. *An Introduction to Support Vector Machines and other kernel-based learning methods.* Cambridge University Press, 2000.

[CT91] Thomas M. Cover and Joy A. Thomas. *Elements of Information Theory.* Wiley-Interscience, 1991.

[DH73] R. Duda and P. Hart. *Pattern classification and scene analysis.* John Wiley & Sons, 1973.

[FCD+00] T. S. Furey, N. Cristianini, N. Duffy, D. W. Bednarski, M. Schummer, and D. Haussler. Support vector machine classification and validation of cancer tissue samples using microarray expression data. *Bioinformatics*, 16(10), 2000.

[FG01] F. Fleuret and D. Geman. Coarse-to-fine visual selection. *International Journal of Computer Vision*, 41(1/2):85–107, 2001.

[FG02] F. Fleuret and D. Geman. Fast face detection with precise pose estimation. In *Proceedings of ICPR2002*, volume 1, pages 235–238, 2002.

[FHT00] J. H. Friedman, T. Hastie, and R. Tibshirani. Additive logistic regression: a statistical view of boosting. *Annal of Statistics*, 28:337–407, 2000.

[FS96a] Y. Freund and R.E. Schapire. Experiments with a new boosting algorithm. In *Proc. 13th International Conference on Machine Learning*, pages 148–156. Morgan Kaufmann, 1996.

[FS96b] Y. Freund and R.E. Schapire. Game theory, on-line prediction and boosting. In *Proc. 9th Annu. Conf. on Comput. Learning Theory*, pages 325–332. ACM Press, New York, NY, 1996.

[GE03] Isabelle Guyon and André Elisseeff. An introduction to variable and feature selection. *Journal of Machine Learning Research*, 3:1157–1182, 2003.

[LIT92] P. Langley, W. Iba, and K. Thompson. An analysis of bayesian classifiers. In *Proceedings of AAAI-92*, pages 223–228, 1992.

[MBBF00] L. Mason, J. Baxter, P. Bartlett, and M. Frean. Boosting algorithms as gradient descent. In *Neural Information Processing Systems*, pages 512–518. MIT Press, 2000.

[MP00] Koji Miyahara and Michael J. Pazzani. Collaborative filtering with the simple bayesian classifier. In *Pacific Rim International Conference on Artificial Intelligence*, pages 679–689, 2000.

[Nov62] A. Novikoff. On convergence proofs for perceptrons. In *In Symposium on Mathematical Theory of Automata*, pages 615–622, 1962.

[ROM98] Gunnar Ratsch, Takashi Onoda, and Klaus-Robert Muller. Regularizing adaboost. In *Proceedings of NIPS*, pages 564–570, 1998.

[Ros58] Frank Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65:386–408, 1958.

[Tor03] Kari Torkkola. Feature extraction by non-parametric mutual information maximization. *Journal of Machine Learning Research*, 3:1415–1438, 2003.

Contents

Unité de recherche INRIA Rocquencourt
Domaine de Voluceau - Rocquencourt - BP 105 - 78153 Le Chesnay Cedex (France)

Unité de recherche INRIA Futurs : Parc Club Orsay Université - ZAC des Vignes
4, rue Jacques Monod - 91893 ORSAY Cedex (France)
Unité de recherche INRIA Lorraine : LORIA, Technopôle de Nancy-Brabois - Campus scientifique
615, rue du Jardin Botanique - BP 101 - 54602 Villers-lès-Nancy Cedex (France)
Unité de recherche INRIA Rennes : IRISA, Campus universitaire de Beaulieu - 35042 Rennes Cedex (France)
Unité de recherche INRIA Rhône-Alpes : 655, avenue de l'Europe - 38334 Montbonnot Saint-Ismier (France)
Unité de recherche INRIA Sophia Antipolis : 2004, route des Lucioles - BP 93 - 06902 Sophia Antipolis Cedex (France)

Éditeur
INRIA - Domaine de Voluceau - Rocquencourt, BP 105 - 78153 Le Chesnay Cedex (France)
http://www.inria.fr
ISSN 0249-6399