A stochastic algorithm for feature selection in pattern recognition

Sébastien Gadat GADAT@CICT.FR
CMLA, ENS Cachan, 61 avenue du président Wilson,
CACHAN 94235 Cachan Cedex, FRANCE

Laurent Younes LAURENT.YOUNES@CIS.JHU.EDU
Center for Imaging Science,
The Johns Hopkins University, 3400 N-Charles Street,
Baltimore MD 21218-2686, USA

Abstract

We introduce a new model adressing feature selection from a large dictionary of variables that can be computed from a signal or an image. Features are extracted according to an efficiency criterion, on the basis of specified classification or recognition tasks. This is done by estimating a probability distribution $\mathbb{P}$ on the complete dictionary, which distributes its mass over the more efficient, or informative, components. We implement a stochastic gradient descent algorithm, using the probability as a state variable and optimizing a multi-task goodness of fit criterion for classifiers based on variable randomly chosen according to $\mathbb{P}$. We then generate classifiers from the optimal distribution of weights learned on the training set. The method is first tested on several pattern recognition problems including face detection, handwritten digit recognition, spam classification and micro-array analysis. We then compare our approach with other step-wise algorithms like random forests or recursive feature elimination.

Keywords: stochastic learning algorithms, Robbins-Monro application, pattern recognition, classification algorithm, feature selection

1. Introduction

Most of the recent instances of pattern recognition problems (whether in computer vision, image understanding, biology, text interpretation, or spam detection) involve highly complex datasets with a huge number of possible explanatory variables. For many reasons, this abundance of variables significantly harms classification or recognition tasks. Weakly informative features act as artificial noise in data and limit the accuracy of classification algorithms. Also, the variance of a statistical model is typically an increasing function of the number of variables, whereas the bias is a decreasing function of this same quantity (Bias-Variance dilemma discussed by Geman et al. (1992)); reducing the dimension of the feature space is necessary to infer reliable conclusions. There are efficiency issues, too, since the speed of many classification algorithms is largely improved when the complexity of the data is reduced. For instance, the complexity of the q-nearest neighbor algorithm varies proportionally with the number of variables. In some cases, the application of classification algorithms like Support Vector Machines (see Vapnik, 1998, 2000) or q-nearest neighbors on the full feature space is not possible or realistic due to the time needed to apply the decision rule. Also, there are many applications for which detecting the pertinent explanatory variables is critical, and as important as correctly performing classification tasks. This

is the case, for example, in biology, where describing the source of a pathological state is equally important to just detecting it (Guyon et al., 2002; Golub et al., 1999).

Feature selection methods are classically separated into two classes. The first approach (*filter methods*) uses statistical properties of the variables to filter out poorly informative variables. This is done before applying any classification algorithm. For instance, singular value decomposition or independent component analysis (Jutten and Hérault, 1991) remain popular methods to limit the dimension of signals, but these two methods do not always yield relevant selection of variables. Superpositions of several efficient filters (Bins and Draper, 2001) has been proposed to remove irrelevant and redundant features, and the use of a combinatorial feature selection algorithm has provided results achieving high reduction of dimensions (more than 80 % of features are removed) preserving good accuracy of classification algorithms on real life problems of image processing. (Xing et al., 2001) have proposed a mixture model and afterwards an information gain criterion and a Markov Blanket Filtering method to reach very low dimensions. They next apply classification algorithms based on Gaussian classifier and Logistic Regression to get very accurate models with few variables on the standard database studied in (Golub et al., 1999). The heuristic of Markov Blanket Filtering has been likewise competitive in feature selection for video application in (Liu and Render, 2003).

The second approach (*wrapper methods*) is computationally demanding, but often is more accurate. A wrapper algorithm explores the space of features subsets to optimize the induction algorithm that uses the subset for classification. These methods based on penalization face a combinatorial challenge when the set of variables has no specific order and when the search must be done over its subsets since many problems related to feature extraction have been shown to be NP-hard (Blum and Rivest, 1992). Therefore, automatic feature space construction and variable selection from a large set has become an active research area. For instance, in (Fleuret and Geman, 2001; Amit and Geman, 1999; Breiman, 2001) the authors successively build tree-structured classifiers considering statistical properties like correlations or empirical probabilities in order to achieve good discriminant properties . In a recent work of Fleuret (2004), the author suggests to use mutual information to recursively select features and obtain performance as good as that obtained with a boosting algorithm (Friedman et al., 2000) with fewer variables. Weston et al. (2000) and Chapelle et al. (2002) construct another recursive selection method to optimize generalization ability with a gradient descent algorithm on the margin of Support Vector classifiers. Another effective approach is the Automatic Relevance Determination (ARD) used in MacKay (1992) which introduce a learning hierarchical prior over weights in a Bayesan Network, the weights connected to irrelevant features are automatically penalysed which reduces their influence near zero. At last, an interesting work of Cohen et al. (2005) use an hybrid wrapper and filter approach to reach highly accurate and selective results. They consider an empirical loss function as a Shapley value and perform an iterative ranking method combined with backward elimination and forward selection. This last work is not so far from the ideas we develop in this paper.

In this paper, we provide an algorithm which attributes a weight to each feature, in relation with their importance for the classification task. The whole set of weights is optimized by a learning algorithm based on a training set. This weighting technique is not so new in the context of feature selection since it has been used in Sun and Li (2006). In

our work, these weights result in an estimated probability distribution over features. This estimated probability will also be used to generate randomized classifiers, where the randomization is made on the variables rather than on the training set, an idea introduced by Amit and Geman (1997), and formalized by Breiman (2001). The selection algorithm and the randomized classifier will be tested on a series of examples.

The article is organized as follows. In section 2, we describe our feature extraction model and the related optimization problem. Next, in sections 3 and 4, we define stochastic algorithms which solve the optimization problem and discuss its convergence properties. In section 5, we provide several applications of our method, first with synthetic data, then on image classification, spam detection and on microarray analysis. Section 6 is the conclusion and addresses future developments.

2. Feature extraction model

We start with some notations.

2.1 Primary notation

We follow the general framework of supervised statistical pattern recognition: the input signal, belonging to some set $\mathcal{I}$, is modeled as a realization of a random variable, from which several computable quantities are accessible, forming a complete set of variables (or tests, or features) denoted $\mathcal{F} = \{\delta_1, \dots \delta_{|\mathcal{F}|}\}$. This set is assumed to be finite, although $|\mathcal{F}|$ can be a very large number, and our goal is to select the most useful variables. We denote $\delta(I)$ the complete set of features extracted from an input signal I.

A classification task is then specified by a finite partition $\mathcal{C} = \{C_1, \dots C_N\}$ of $\mathcal{I}$; the goal is to predict the class from the observation $\delta(I)$. For such a partition $\mathcal{C}$ and $I \in \mathcal{I}$, we write $\mathcal{C}(I)$ for the class C_i to which I belongs, which is assumed to be deterministic.

2.2 Classification algorithm

We assume that a classification algorithm, denoted $\mathbb{A}$, is available, for both training and testing. We assume that $\mathbb{A}$ can be adapted to a subset $\omega \subset \mathcal{F}$ of *active variables* and to a specific classification task $\mathcal{C}$. (There may be some restriction on $\mathcal{C}$ associated to $\mathbb{A}$, for example, support vector machines are essentially designed for binary (two-class) problems.) In training mode, $\mathbb{A}$ uses a database to build an optimal classifier $\mathbb{A}_{\omega,\mathcal{C}} : \mathcal{I} \to \mathcal{C}$, such that $\mathcal{A}_{\omega,\mathcal{C}}(I)$ only depends on $\omega(I)$. We shall drop the subscript $\mathcal{C}$ when there is no ambiguity concerning the classification problem $\mathcal{C}$. The test mode simply consists in the application of $\mathcal{A}_\omega$ on a given signal.

We will work with a randomized version of $\mathbb{A}$, for which the randomization is with respect to the set of variables ω (see Amit and Geman, 1997; Breiman, 2001, 1998). In the training phase, this works as follows: first extract a collection $\{\omega^{(1)}, \dots, \omega^{(n)}\}$ of subsets of $\mathcal{F}$, and build the classifiers $\mathbb{A}_{\omega^{(1)}}, \dots, \mathbb{A}_{\omega^{(n)}}$. Then, perform classification in test phase using a majority rule for these n classifiers. This final algorithm will be denoted $\bar{\mathbb{A}}_{\mathcal{C}} = \bar{\mathbb{A}}(\omega^{(1)}, \dots, \omega^{(n)}, \mathcal{C})$. It is run with fixed $\omega^{(i)}$'s, which are obtained during learning.

Note that in this paper, we are not designing the classification algorithm, $\mathbb{A}$, for which we use existing procedures; rather, we focus on the extraction mechanism creating the random

subsets of $\mathcal{F}$. This randomization process depends on the way variables are sampled from $\mathcal{F}$, and we will see that the design of a suitable probability on $\mathcal{F}$ for this purpose can significantly improve the classification rates. This probability therefore comes as a new parameter and will be learned from the training set.

Figure 1 summarizes the algorithm in test phase.

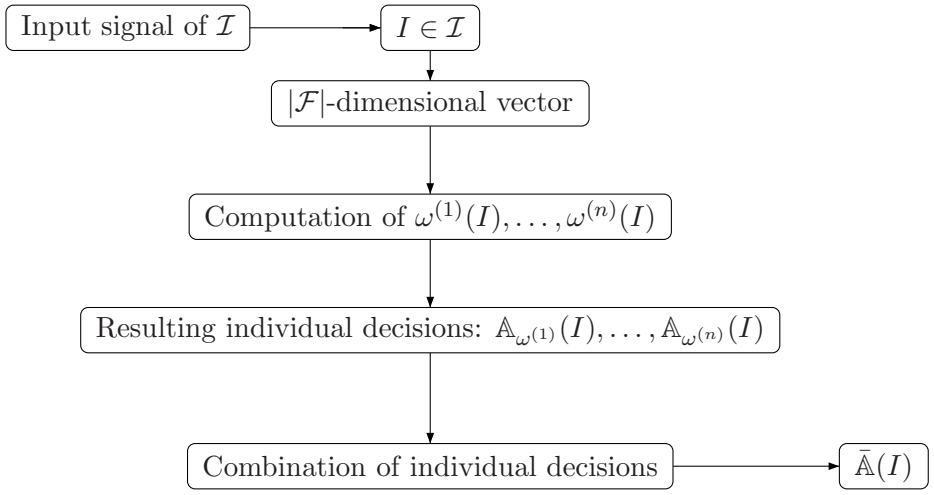

Figure 1: Global algorithm in test phase.

2.3 Measuring the performance of the algorithm $\mathbb{A}$

The algorithm $\mathbb{A}$ provides a different classifier $\mathbb{A}_\omega$ for each choice of a subset $\omega \subset \mathcal{F}$. We let q be the classification error: $q(\omega, \mathcal{C}) = \mathbf{P}(\mathbb{A}_\omega(I) \neq \mathcal{C}(I))$ which will be estimated by

$$q^\star(\omega, \mathcal{C}) = \hat{\mathbf{P}}(\mathbb{A}_\omega(I) \neq \mathcal{C}(I)) \tag{1}$$

where $\hat{\mathbf{P}}$ is the empirical probability on the training set.

We shall consider two particular cases for a given $\mathbb{A}$.

- Multi-class algorithm: assume that $\mathbb{A}$ is naturally adapted to multi-class problems (like a q-nearest neighbour, or random forest classifier). We then let $g(\omega) = q^\star(\omega, \mathcal{C})$ as defined above.

- Two-class algorithms: this applies to algorithms like support vector machines, which are designed for binary classification problems. We use the idea of the one-against-all method: denote by $\mathcal{C}_i$ the binary partition $\{C_i,\ \mathcal{I} \setminus C_i\}$. We then denote:

$$g(\omega) = \frac{1}{N} \sum_{i=1}^{N} q^\star(\omega, \mathcal{C}_i)$$

  which is the average classification rate of the one vs. all classifiers. This "one against all strategy" can easily be replaced by others, like methods using error correcting output codes (see Dietterich and Bakiri, 1995).

2.4 A computational amendment

The computation $q^\star(\omega, \mathcal{C})$, as defined in equation (1), requires training a new classification algorithm with variables restricted to ω, and estimating the empirical error; this can be rather costly with large datasets (this has to be repeated at each of the steps of the learning algorithm).

Because of this, we use a slightly different evaluation of the error. In the algorithm, each time an evaluation of $q^\star(\omega, \mathcal{C})$ is needed, we use the following procedure (T being a fixed integer and $\mathcal{T}_{train}$ will be the training set):

1. Sample a subset $\mathcal{T}_1$ of size T (with replacements) from the training set.
2. Learn the classification algorithm on the basis of ω and $\mathcal{T}_1$.
3. Sample, with the same procedure, a subset $\mathcal{T}_2$ from the training set, and define $\hat{q}_{(\mathcal{T}_1,\mathcal{T}_2)}(\omega, \mathcal{C})$ to be the empirical error of the classifier learned *via* $\mathcal{T}_1$ on $\mathcal{T}_2$.

Since $\mathcal{T}_1$ and $\mathcal{T}_2$ are independent, we will use $\hat{q}(\omega, \mathcal{C})$ defined by

$$\boxed{\hat{q}(\omega, \mathcal{C}) = \mathbb{E}_{(\mathcal{T}_1,\mathcal{T}_2)}\left[\hat{q}_{(\mathcal{T}_1,\mathcal{T}_2)}(\omega, \mathcal{C})\right]} \tag{2}$$

to quantify the efficiency of the subset ω, where the expectation is computed over all the samples $\mathcal{T}_1, \mathcal{T}_2$ of signals taken from the training set of size T. It is also clear that defining such a cost function contributes in avoiding overfitting in the selection of variables. For the multiclass problem, we define

$$\hat{g}_{\mathcal{T}_1,\mathcal{T}_2}(\omega) = \frac{1}{N}\sum_{i=1}^{N} \hat{q}_{(\mathcal{T}_1,\mathcal{T}_2)}(\omega, \mathcal{C}_i)$$

and we replace the previous expression of g by the one below:

$$\boxed{g(\omega) = \mathbb{E}_{\mathcal{T}_1,\mathcal{T}_2}\left[\hat{g}_{\mathcal{T}_1,\mathcal{T}_2}(\omega)\right]}$$

This modified function g will be used later in combination with a stochastic algorithm which will replace the expectation over the training and validation subsets by empirical averages. The selection of smaller training and validation sets for the evaluation of $\hat{g}_{\mathcal{T}_1,\mathcal{T}_2}$ then represents a huge reduction of computer time. The selection of the size of $\mathcal{T}_1$ and $\mathcal{T}_2$ depends on the size of the original training set and of the chosen learning machine. It has so far been selected by hand.

In the rest of our paper, the notation $\mathbb{E}_\xi[.]$ will refer to the expectation using ξ as the integration variable.

2.5 Weighting the feature space

To select a group a variables which are most relevant for the classification task one can think first of a *hard selection method*, *i.e.* search $\underline{\omega}$ such that

$$\hat{q}(\underline{\omega}, \mathcal{C}) = \arg\min_{\omega \in \mathcal{F}^{|\omega|}} \hat{q}(\omega, \mathcal{C})$$

But sampling all possible subsets (ω covers $\mathcal{F}^{|\omega|}$) may be untractable since $|\mathcal{F}|$ can be thousands and $|\omega|$ ten or hundreds.

We address this with a soft selection procedure that attributes weights to the features $\mathcal{F}$.

2.6 Feature extraction procedure

Consider a probability distribution $\mathbb{P}$ on the set of features $\mathcal{F}$. For an integer k, the distribution $\mathbb{P}^{\otimes k}$ corresponds to k independent trials with distribution $\mathbb{P}$. We define the cost function $\mathcal{E}$ by

$$\mathcal{E}(\mathbb{P}) = \mathbb{E}_{\mathbb{P}^{\otimes k}} g(\omega) = \sum_{\omega \in \mathcal{F}^k} g(\omega) \mathbb{P}^{\otimes k}(\omega). \tag{3}$$

Our goal is to minimize this averaged error rate with respect to the selection parameter, which is the probability distribution $\mathbb{P}$. The relevant features will then be the set of variables $\delta \in \mathcal{F}$ for which $\mathbb{P}(\delta)$ is large. The global (iterative) procedure that we propose for estimating $\mathbb{P}$ is summarized in Figure 2.

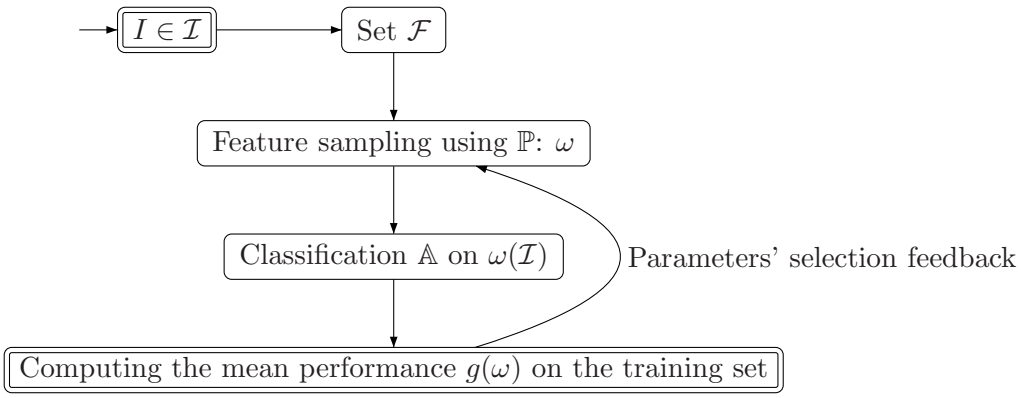

Figure 2: Scheme of the procedure for learning the probability $\mathbb{P}$.

Remark: We use $\mathbb{P}$ here as a control parameter: we first make a random selection of features before setting the learning algorithm. It is thus natural to optimize our way to select features from $\mathcal{F}$ and formalize it as a probability distribution on $\mathcal{F}$. The number of sampled features (k) is a hyperparameter of the method. Although we have set it by hand in our experiments, it can be estimated by cross-validation during learning.

3. Search algorithms

We describe in this section three algorithmic options to minimize the energy $\mathcal{E}$ with respect to the probability $\mathbb{P}$. This requires computing the gradient of $\mathcal{E}$, and dealing with the constraints implied by the fact that $\mathbb{P}$ must be a probability distribution. These constraints are summarized in the following notation.

We denote by $\mathcal{S}_\mathcal{F}$ the set of all probability measures on $\mathcal{F}$: a vector $\mathbb{P}$ of $\mathbb{R}^{|\mathcal{F}|}$ belongs to $\mathcal{S}_\mathcal{F}$ if

$$\sum_{\delta \in \mathcal{F}} \mathbb{P}(\delta) = 1 \tag{4}$$

and

$$\forall \delta \in \mathcal{F} \quad \mathbb{P}(\delta) \geq 0 \tag{5}$$

We also denote $\mathcal{H}_\mathcal{F}$ the hyperplane in $\mathbb{R}^{|\mathcal{F}|}$ which contains $\mathcal{S}_\mathcal{F}$, defined by (4).

We define the projections of an element of $\mathbb{R}^{|\mathcal{F}|}$ onto the closed convex sets $\mathcal{S}_\mathcal{F}$ and $\mathcal{H}_\mathcal{F}$. Let $\pi_{\mathcal{S}_\mathcal{F}}(x)$ be the closest point of $x \in \mathbb{R}^{|\mathcal{F}|}$ in $\mathcal{S}_\mathcal{F}$

$$\pi_{\mathcal{S}_\mathcal{F}}(x) = \arg\min_{y \in \mathcal{S}_\mathcal{F}} \left\{ \|y - x\|_2{}^2 \right\} \tag{6}$$

and similarly

$$\pi_{\mathcal{H}_\mathcal{F}}(x) = \arg\min_{y \in \mathcal{H}_\mathcal{F}} \left\{ \|y - x\|_2{}^2 \right\} = x - \frac{1}{|\mathcal{F}|} \sum_{\delta \in \mathcal{F}} x(\delta). \tag{7}$$

The latter expression comes from the fact that the orthogonal projection of a vector x onto a hyperplane is $x - \langle x, N \rangle N$ where N is the unit normal to the hyperplane. For $\mathcal{H}_\mathcal{F}$, N is the vector with all coordinates equal to $1/\sqrt{|\mathcal{F}|}$.

Our first option will be to use projected gradient descent to minimize $\mathcal{E}$, taking only constraint (4) into account. This implies solving the gradient descent equation

$$\frac{\mathrm{d}\mathbb{P}_t}{dt} = -\pi_{\mathcal{H}_\mathcal{F}} \left(\nabla \mathcal{E}(\mathbb{P}_t) \right) \tag{8}$$

which is well-defined as long as $\mathbb{P}_t \in \mathcal{S}_\mathcal{F}$. We will also refer to the discretized form of (8),

$$\mathbb{P}_{n+1} = \mathbb{P}_n - \epsilon_n \pi_{\mathcal{H}_\mathcal{F}} \left(\nabla \mathcal{E}(\mathbb{P}_n) \right) \tag{9}$$

with positive $(\epsilon_n)_{n \in \mathbb{N}}$. Again, this equation can be implemented as long as $\mathbb{P}_n \in \mathcal{S}_\mathcal{F}$. We will later propose two new strategies to deal with the positivity constraint (5), the first one using the change of variables $\mathbb{P} \mapsto \log \mathbb{P}$, and the second being a constrained optimization algorithm, that we will implement as a constrained stochastic diffusion on $\mathcal{S}_\mathcal{F}$.

3.1 Computation of the gradient

However, returning to (8), our first task is to compute the gradient of the energy. We first do it in the standard case of the Euclidean metric on $\mathcal{S}_\mathcal{F}$, that is we compute $\nabla_\mathbb{P} \mathcal{E}(\delta) = \partial \mathcal{E} / \partial \mathbb{P}(\delta)$. For $\omega \in \mathcal{F}^k$ and $\delta \in \mathcal{F}$, denote by $C(\omega, \delta)$ the number of occurences of δ in ω:

$$\boxed{C(\omega, \delta) = |\{i \in \{1, \dots k\} \quad | \quad \omega_i = \delta\}|} \tag{10}$$

$C(\omega, .)$ is then the $|\mathcal{F}|$-dimensional vector composed by the set of values $C(\omega, \delta), \delta \in \mathcal{F}$. Then, a straightforward computation gives:

Proposition 1 *If $\mathbb{P}$ is any point of $\mathcal{S}_{\mathcal{F}}$, then*

$$\forall \delta \in \mathcal{F} \qquad \nabla_{\mathbb{P}} \mathcal{E}(\delta) = \sum_{\omega \in \mathcal{F}^k} \frac{C(\omega, \delta) \mathbb{P}^{\otimes k}(\omega)}{\mathbb{P}(\delta)} g(\omega) \tag{11}$$

Consequently, the expanded version of (9) is

$$\mathbb{P}_{n+1}(\delta) \;\; = \;\; \mathbb{P}_n(\delta) - \epsilon_n \sum_{\omega \in \mathcal{F}^k} \mathbb{P}^{\otimes k}(\omega) g(\omega) \left(\frac{C(\omega, \delta)}{\mathbb{P}(\delta)} - \frac{1}{|\mathcal{F}|} \sum_{\mu \in \omega} \frac{C(\omega, \mu)}{\mathbb{P}(\mu)} \right) \tag{12}$$

In the case when $\mathbb{P}(\delta) = 0$, then, necessarily $C(\omega, \delta) = 0$ and the term $C(\omega, \delta)/\mathbb{P}(\delta)$ is by convention equal to 0.

The positivity constraint is not taken in account here, but this can be dealt with, as described in the next section, by switching to an exponential parameterization. It is also be possible to design a constrained optimization algorithm, exploring the faces of the simplex when needed, but this is a rather complex procedure, which is harder to conciliate with the stochastic approximations we will describe in section 4. This approach will in fact be computationally easier to handle with a constrained stochastic diffusion algorithm, as described in section 3.3.

3.2 Exponential parameterization and Riemannian gradient

Define $y(\delta) = \log \mathbb{P}(\delta)$ and

$$\mathcal{Y} = \left\{ y = (y(\delta), \delta \in \mathcal{F}) \;\mid\; \sum_{\delta \in \mathcal{F}} e^{y(\delta)} = 1 \right\} \tag{13}$$

which is in one-to-one correspondence with $\mathcal{S}_{\mathcal{F}}$ (allowing for the choice $y(\delta) = -\infty$). Define

$$\tilde{\mathcal{E}}(y) = \mathcal{E}(\mathbb{P}) = \sum_{\omega \in \mathcal{F}^k} e^{y(\omega_1) + \cdots + y(\omega_k)} g(\omega) \tag{14}$$

Then, we have:

Proposition 2 *The gradient of $\mathcal{E}$ with respect to these new variables is given by:*

$$\nabla_y \tilde{\mathcal{E}}(\delta) = \sum_{\omega \in \mathcal{F}^k} e^{y(\omega_1) + \cdots + y(\omega_k)} C(\omega, \delta) g(\omega). \tag{15}$$

We can interpret this gradient on the variables y as a gradient on the variables $\mathbb{P}$ with a Riemannian metric

$$\langle u, v \rangle_{\mathbb{P}} = \sum_{\delta \in \mathcal{F}} u(\delta) v(\delta) / \mathbb{P}(\delta).$$

The geometry of the space $\mathcal{S}_{\mathcal{F}}$ with this metric D has the property that the boundary points $\partial \mathcal{S}_{\mathcal{F}}$ are at infinite distance from any point into the interior of $\mathcal{S}_{\mathcal{F}}$.

Denoting $\tilde{\nabla}$ for the gradient with respect to this metric, we have in fact, with $y = \log \mathbb{P}$:

$$\tilde{\nabla}_{\mathbb{P}} \mathcal{E}(\delta) = \nabla_y \tilde{E} = \sum_{\omega \in \mathcal{F}^k} \mathbb{P}^{\otimes k}(\omega) C(\omega, \delta) g(\omega) \tag{16}$$

To handle the unit sum constraint, we need to project this gradient on the tangent space to $\mathcal{Y}$ at point y. Denoting this projection by π_y, we have

$$\pi_y(w) = w - \langle w | e^y \rangle / \|e^y\|^2$$

where e^y is the vector with coordinates $e^{y(\delta)}$. This yields the evolution equation in the y variables

$$\frac{dy_t(\delta)}{\mathrm{d}t} = -\nabla_{y_t} \tilde{\mathcal{E}}(\delta) + \kappa_t e^{y_t(\delta)}, \tag{17}$$

where

$$\kappa_t = \left(\sum_{\delta' \in \mathcal{F}} \nabla_{y_t} \tilde{\mathcal{E}}(\delta') e^{y_t(\delta')} \right) / \left(\sum_{\delta' \in \mathcal{F}} e^{2y_t(\delta')} \right)$$

does not depend on δ.

The associated evolution equation for $\mathbb{P}$ becomes

$$\frac{d\mathbb{P}_t(\delta)}{\mathrm{d}t} = -\mathbb{P}_t(\delta) \left(\tilde{\nabla}_{\mathbb{P}_t} \mathcal{E}(\delta) - \kappa_t \mathbb{P}_t(\delta) \right). \tag{18}$$

Consider now a discrete time approximation of (18), under the form

$$\mathbb{P}_{n+1}(\delta) = \frac{\mathbb{P}_n(\delta)}{K_n} e^{-\epsilon_n \left(\tilde{\nabla}_{\mathbb{P}_n} \mathcal{E}(\delta) - \kappa_n \mathbb{P}_n(\delta) \right)} \tag{19}$$

where the newly introduced constant K_n ensures that the probabilities sum to 1. This provides an alternative scheme of gradient descent on $\mathcal{E}$ which has the advantage of satisfying the positivity constraints (5) by construction.

- Start with $\mathbb{P}_0 = \mathcal{U}_{\mathcal{F}} \longmapsto y_0 = \log \mathbb{P}_0$
- Until convergence: Compute $\mathbb{P}_{n+1}$ from equation (19).

Remark: In terms of y_n, (19) yields

$$y_{n+1}(\delta) = y_n(\delta) - \epsilon_n \left(\tilde{\nabla}_{\mathbb{P}_n} \mathcal{E}(\delta) - \kappa_n \mathbb{P}_n(\delta) \right) - \log K_n \,. \tag{20}$$

The definition of the constant K_n implies that

$$K_n = \sum_{\delta \in \mathcal{F}} \mathbb{P}_n e^{-\epsilon_n \left(\tilde{\nabla}_{\mathbb{P}_n} \mathcal{E}(\delta) - \kappa_n \mathbb{P}_n(\delta) \right)}.$$

We can write a second order expansion of the above expression to deduce that

$$K_n = \sum_{\delta \in \mathcal{F}} \mathbb{P}_n(\delta) - \epsilon_n \mathbb{P}_n(\delta) \left(\tilde{\nabla}_{\mathbb{P}_n} \mathcal{E}(\delta) - \kappa_n \mathbb{P}_n(\delta) \right) + A_n \epsilon_n^2 = 1 + A_n \epsilon_n^2$$

since, by definition of κ_n:

$$\sum_{\delta \in \mathcal{F}} \mathbb{P}_n(\delta) (\tilde{\nabla}_{\mathbb{P}_n} E(\delta) - \kappa_n \mathbb{P}_n(\delta)) = 0.$$

Consequently, there exists a constant B which depends on k and $\max(\epsilon_n)$ such that, for all n, $|\log K_n| \leq B \epsilon_n^2$.

3.3 Constrained diffusion

The algorithm (12) can be combined with reprojection steps to provide a consistent procedure. We implement this using a stochastic diffusion process constrained to the simplex $\mathcal{S}_{\mathcal{F}}$. The associated stochastic differential equation is

$$\mathrm{d}\mathbb{P}_t = -\nabla_{\mathbb{P}_t}\mathcal{E}\mathrm{d}t + \sqrt{\sigma}\mathrm{d}W_t + \mathrm{d}Z_t \tag{21}$$

where $\mathcal{E}$ is the cost function introduced in (3), σ is a positive nondegenerate matrix on $\mathcal{H}_{\mathcal{F}}$ and $\mathrm{d}Z_t$ is a stochastic process which accounts for the jumps which appear when a reprojection is needed. In other words, $\mathrm{d}|Z_t|$ is positive if and only if $\mathbb{P}_t$ hits the boundary $\partial\mathcal{S}_{\mathcal{F}}$ of our simplex.

The rigorous construction of such a process is linked to the theory of Skorokhod maps, and can be found in works of Dupuis and Ishii (1991) and Dupuis and Ramanan (1999). Existence and uniqueness are true under general geometric conditions which are satisfied here.

4. Stochastic approximations

The evaluation of $\nabla\mathcal{E}$ in the previous algorithms requires summing the efficiency measures $g(\omega)$ over all ω in $\mathcal{F}^k$. This is, as already discussed, an untractable sum. This however can be handled using a stochastic approximation, as described in the next section.

4.1 Stochastic gradient descent

We first recall general facts on stochastic approximation algorithms.

4.1.1 Applying the ODE method

Stochastic approximations can be seen as noisy discretizations of deterministic ODE's (see Benveniste et al., 1990; Benaïm, 2000; Duflo, 1996). They are generally expressed under the form

$$X_{n+1} = X_n + \epsilon_n F(X_n, \xi_{n+1}) + \epsilon_n^2\eta_n \tag{22}$$

where X_n is the current state of the process, ξ_{n+1} a random perturbation, and η_n a secondary error term. If the distribution of ξ_{n+1} only depends on the current value of X_n (Robbins-Monro case), then one defines an average drift $X \mapsto G(X)$ by

$$G(X) = \mathbb{E}_\xi[F(X,\xi)] \tag{23}$$

and the equation (22) can be shown to evolve similarly to the ODE $\dot{X} = G(X)$, in the sense that the trajectories coincide when $(\epsilon_n)_{n\in\mathbb{N}}$ goes to 0 (a more precise statement is given in section 4.1.4).

4.1.2 Approximation terms

To implement our gradient descent equations in this framework, we therefore need to identify two random variables d_n or $\tilde{d}_n$ such that

$$\mathbb{E}\left[d_n\right] = \pi_{\mathcal{H}_{\mathcal{F}}}\left[\nabla_{\mathbb{P}_n}\mathcal{E}\right] \qquad \text{and} \qquad \mathbb{E}\left[\tilde{d}_n\right] = \pi_{y_n}\left[\nabla_{y_n}\tilde{\mathcal{E}}\right] \tag{24}$$

This would yield the stochastic algorithms:

$$\mathbb{P}_{n+1} = \mathbb{P}_n - \epsilon_n d_n \text{ or } \mathbb{P}_{n+1} = \mathbb{P}_n \frac{e^{-\epsilon_n \tilde{d}_n}}{K_n}.$$

¿From (11), we have:

$$\nabla_{\mathbb{P}} \mathcal{E}(\delta) = \mathbb{E}_{\omega} \left[\frac{C(\omega, \delta) g(\omega)}{\mathbb{P}(\delta)} \right].$$

Using the linearity of the projection $\pi_{\mathcal{H}_{\mathcal{F}}}$, we get

$$\pi_{\mathcal{H}_{\mathcal{F}}} \left(\nabla \mathcal{E}(\mathbb{P}) \right) (\delta) = \mathbb{E}_{\omega} \left[\pi_{\mathcal{H}_{\mathcal{F}}} \left(\frac{C(\omega, .) g(\omega)}{\mathbb{P}(.)} \right) (\delta) \right].$$

Consequently, following (24), it is natural to define the approximation term of the gradient descent (8) by:

$$\boxed{d_n = \pi_{\mathcal{H}_{\mathcal{F}}} \left(\frac{C(\omega_n, .) \hat{q}_{\mathcal{T}_1^n, \mathcal{T}_2^n}(\omega_n, \mathcal{C})}{\mathbb{P}_n(.)} \right)} \tag{25}$$

where the set of k features ω_n is a random variable extracted from $\mathcal{F}$ with law $\mathbb{P}_n^{\otimes k}$ and $\mathcal{T}_1^n, \mathcal{T}_2^n$ are independently sampled into the training set $\mathcal{T}$.

In a similar way, we can compute the approximation term of the gradient descent based on (15) since

$$\nabla_y \tilde{\mathcal{E}}(\delta) = \mathbb{E}_{\omega} \left[g(\omega) C(\omega, \delta) \right]$$

yielding

$$\tilde{d}_n = \pi_{y_n} \left(C(\omega_n, .) \hat{q}_{\mathcal{T}_1^n, \mathcal{T}_2^n}(\omega_n, \mathcal{C})) \right) \tag{26}$$

where π_y is the projection on the tangent space $\mathcal{T}\mathcal{Y}$ to the sub-manifold $\mathcal{Y}$ at point y, and ω_n is a random variable extracted from $\mathcal{F}$ with the law $\mathbb{P}_n^{\otimes k}$.

By construction, we therefore have the proposition

Proposition 3 *The mean effect of random variables d_n and $\tilde{d}_n$ is the global gradient descent, in other words:*

$$\mathbb{E}\left[d_n \right] = \pi_{\mathcal{H}_{\mathcal{F}}} \left(\nabla \mathcal{E}(\mathbb{P}_n) \right) \tag{27}$$

and

$$\mathbb{E}\left[\tilde{d}_n \right] = \pi_{y_n} \left(\nabla \tilde{\mathcal{E}}(y_n) \right). \tag{28}$$

4.1.3 Learning the probability map $(\mathbb{P}_n)_{n \in \mathbb{N}}$

We now make explicit the learning algorithms for equations (8) and (17). We recall the definition of

$$C(\omega, \delta) = |\{ i \in \{1, \dots k\} \quad | \quad \omega_i = \delta \}|$$

where δ is a given feature and ω a given feature subset of length k which is an hyperparameter (see bottom of page 6). $\hat{q}_{\mathcal{T}_1, \mathcal{T}_2}(\omega, \mathcal{C})$, which is the empirical classification error on $\mathcal{T}_2$ for a classifier trained on $\mathcal{T}_1$ using features in ω.

• Euclidean gradient (figure 3):

Let $\mathcal{F} = (\delta_1, \dots \delta_{|\mathcal{F}|})$, integers μ, T and a real number α (stoping criterion)
$n = 0$: define $\mathbb{P}_0$ to be the uniform distribution $\mathcal{U}_{\mathcal{F}}$ on $\mathcal{F}$
while $\left|\mathbb{P}_{n-\lfloor n/\mu \rfloor} - \mathbb{P}_n\right|_\infty > \alpha$ and $\mathbb{P}_n \geq 0$
Extract ω_n with replacement from $\mathcal{F}^k$ with respect to $\mathbb{P}_n^{\otimes k}$
Extract $\mathcal{T}_1^n$ and $\mathcal{T}_2^n$ of size T with uniform independent samples over $\mathcal{T}_{train}$
Compute $\hat{q}_{\mathcal{T}_1^n, \mathcal{T}_2^n}(\omega_n, \mathcal{C})$and the drift vector d_n where

$$d_n(\delta) = \hat{q}_{\mathcal{T}_1^n, \mathcal{T}_2^n}(\omega_n, \mathcal{C}) \left(\frac{C(\omega_n, \delta)}{\mathbb{P}_n(\delta)} - \sum_{\mu \in \omega_n} \frac{C(\omega_n, \mu)}{|\mathcal{F}| \mathbb{P}_n(\mu)} \right)$$

Update $\mathbb{P}_{n+1}$ with $\mathbb{P}_{n+1} = \mathbb{P}_n - \epsilon_n . d_n$.
$n \mapsto n + 1$

Figure 3: Euclidean gradient Algorithm.

• Riemannian Gradient: For the Riemannian case, we have to give few modifications for the update step (figure 4).

The mechanism of the two former algorithms summarized by figures 3,4 can be intuitively explained looking carefully at the update step. For instance, in the first case, at step n, one can see that for all features of $\delta \in \omega_n$, we substract from $\mathbb{P}_n(\delta)$ amount proportional to the error performed with ω and inversely proportional to $\mathbb{P}_n(\delta)$ although for other features out of ω_n, weights are a little bit increased. Consequently, worst features with poor error of classification will be severely decreased, particularly when they are suspected to be bad (small weight $\mathbb{P}_n$).

Remark We provide the Euclidean gradient algorithm, which is subject to failure (one weight might become nonpositive) because it may converge for some applications, and in these cases, is much faster than the exponential formulation.

4.1.4 Convergence of the approximation scheme

This more technical section can be skipped without harming the understanding of the rest of the paper. We here rapidly describe in which sense the stochastic approximations we

Let $\mathcal{F} = (\delta_1, \dots \delta_{|\mathcal{F}|})$, integers μ, T and a real number α (stoping criterion)
$n = 0$: define $\mathbb{P}_0$ to be the uniform distribution $\mathcal{U}_{\mathcal{F}}$ on $\mathcal{F}$
while $\left|\mathbb{P}_{n-\lfloor n/\mu \rfloor} - \mathbb{P}_n\right|_\infty > \alpha$
Extract ω_n from $\mathcal{F}^k$ with respect to $\mathbb{P}_n^{\otimes k}$
Extract $\mathcal{T}_1^n$ and $\mathcal{T}_2^n$ of size T with uniform independent samples over $\mathcal{T}_{train}$
Compute $\hat{q}_{\mathcal{T}_1^n, \mathcal{T}_2^n}(\omega_n, \mathcal{C})$.
Update $\mathbb{P}_{n+1}$ with:

$$\mathbb{P}_{n+1}(\delta) = \frac{\mathbb{P}_n(\delta) e^{-\epsilon_n (C(\omega_n, \delta) \hat{q}_{\mathcal{T}_1^n, \mathcal{T}_2^n}(\omega_n, \mathcal{C}) + \kappa_n \mathbb{P}_n(\delta))}}{K_n} \text{ with}$$

$$\kappa_n = \left(\sum_{\delta' \in \omega_n} C(\omega_n, \delta') \mathbb{P}_n(\delta') \right) / \left(\left(\sum_{\delta' \in \mathcal{F}} \mathbb{P}_n(\delta')^2 \right) \right)$$

and K_n is a normalization constant.
$n \mapsto n + 1$

Figure 4: Riemannian gradient Algorithm.

have designed converge to their homologous ODE's. This is a well-known fact, especially in the Robbins-Monro case that we consider here, and the reader may refer to works of Benveniste et al. (1990); Duflo (1996); Kushner and Yin (2003), for more details. We follow the terminology employed in the approach of Benaim (1996).

Fix a finite dimensional open set E. A differential flow $(t, x) \mapsto \phi_t(x)$ is a time-indexed family of diffeomorphisms satisfying the semi-group condition $\phi_{t+h} = \phi_h \circ \phi_t$ and $\phi_0 = Id$; $\phi_t(x)$ is typically given as the solution of a differential equation $\frac{dy}{dt} = G(y)$, at time t, with initial condition $y(0) = x$. Asymptotic pseudotrajectories of a differential flow are defined as follows:

Definition 4 *A map $X : \mathbb{R}^+ \longmapsto E$ is an asymptotic pseudotrajectory of the flow ϕ if, for all positive numbers T*

$$\lim_{t \mapsto \infty} \sup_{0 \leq h \leq T} \|X(t+h) - \phi_h(X(t))\| = 0. \tag{29}$$

In other words, the tails of the trajectory X asymptotically coincides, within any finite horizon T, with the flow trajectories.

Consider algorithms of the form

$$X_{n+1} = X_n + \epsilon_n F(X_n, \xi_{n+1}) + \epsilon_n^2 \eta_{n+1} \tag{30}$$

with $X_n \in E$, ξ_{n+1} a first order perturbation (such that the conditional distribution knowing all present and past variables only depends on X_n), and η_n a secondary noise process. The variable X_n can be linearly interpolated into a time-continuous process as follows: define $\tau_n = \sum_{k=1}^{n} \epsilon_k$ and $X_{\tau_n} = X_n$; then let X_t be linear and continuous between τ_n and τ_{n+1}, for $n \geq 0$.

Consider the mean ODE

$$\frac{dx}{dt} = G(x) = \mathbb{E}_\xi[F(X, \xi)|X = x] \tag{31}$$

and its associated flow ϕ. Then, under mild conditions on F and η_n, and under the assumption that $\sum_{n>0} \epsilon_n^{1+\alpha} < \infty$ for some $\alpha > 0$, the linearly interpolated process X_t is an asymptotic pseudotrajectory of ϕ. We will consequently choose $\epsilon_n = \epsilon/(n + C)$ where ϵ and C are positive constants fixed at start of our algorithms. We can here apply this result with $X_n = y_n$, $\xi_{n+1} = (\omega_n, \mathcal{T}_1^n, \mathcal{T}_2^n)$ and $\eta_{n+1} = \log K_n/\epsilon_n^2$ for which all the required conditions are satisfied since for the euclidean case, when $\omega_n \sim \mathbb{P}_n^{\otimes k}$ and $(\mathcal{T}_1, \mathcal{T}_2) \sim \mathcal{U}_T^{\otimes 2T}$:

$$\begin{aligned}
\mathbb{E}_{\omega_n}\left[F(\mathbb{P}_n,\omega_n)\right] &= \mathbb{E}_{\omega_n,\mathcal{T}_1,\mathcal{T}_2}\left[d_n(\omega_n,\mathcal{T}_1,\mathcal{T}_2)\right] \\
&= \mathbb{E}_{\omega_n,\mathcal{T}_1,\mathcal{T}_2}\left[\pi_{\mathcal{H}_\mathcal{F}}\left(\frac{C(\omega_n,.)\hat{q}_{\mathcal{T}_1,\mathcal{T}_2}(\omega_n,\mathcal{C})}{\mathbb{P}_n(.)}\right)\right] \\
&= \mathbb{E}_{\omega_n}\left[\pi_{\mathcal{H}_\mathcal{F}}\left(\frac{C(\omega_n,.)\mathbb{E}_{\mathcal{T}_1,\mathcal{T}_2}\left[\hat{q}_{\mathcal{T}_1,\mathcal{T}_2}(\omega_n,\mathcal{C})\right]}{\mathbb{P}_n(.)}\right)\right] \\
&= \mathbb{E}_{\omega_n}\left[\pi_{\mathcal{H}_\mathcal{F}}\left(\frac{C(\omega_n,.)\hat{q}(\omega_n,\mathcal{C})}{\mathbb{P}_n(.)}\right)\right] \\
\mathbb{E}_{\omega_n}\left[F(\mathbb{P}_n,\omega_n)\right] &= \pi_{\mathcal{H}_\mathcal{F}}\left(\nabla\mathcal{E}(\mathbb{P}_n)\right)
\end{aligned}$$

4.2 Numerical simulation of the diffusion model

We use again (25) for the approximation of the gradient of $\mathcal{E}$. The theoretical basis for the convergence of this type of approximation can be found in (Buche and Kushner (2001); Kushner and Yin (2003)), for example. A detailed convergence proof is provided in (Gadat (2004)).

This results in the following numerical scheme. We recall the definition of

$$C(\omega,\delta) = |\{i \in \{1,\dots k\} \quad | \quad \omega_i = \delta\}|$$

and of $\hat{q}_{\mathcal{T}_1,\mathcal{T}_2}(\omega,\mathcal{C})$,which is the empirical classification error on $\mathcal{T}_2$ for a classifier trained on $\mathcal{T}_1$ using features in ω.

Let $\mathcal{F} = (\delta_1,\dots\delta_{|\mathcal{F}|})$, an integer
$n = 0$: define $\mathbb{P}_0$ to be the uniform distribution $\mathcal{U}_\mathcal{F}$ on $\mathcal{F}$
Iterate the loop
Extract ω_n from $\mathcal{F}^k$ with respect to $\mathbb{P}_n^{\otimes k}$
Extract $\mathcal{T}_1^n$ and $\mathcal{T}_2^n$ of size T with uniform independent samples over $\mathcal{T}_{train}$
Compute $\hat{q}_{\mathcal{T}_1^n,\mathcal{T}_2^n}(\omega_n,\mathcal{C})$
Compute the intermediate state $\mathbb{Q}_n$ (may be out of $\mathcal{S}_\mathcal{F}$):

$$\mathbb{Q}_n = \mathbb{P}_n - \epsilon_n\frac{C(\omega_n,.)\hat{q}_{\mathcal{T}_1^n,\mathcal{T}_2^n}(\omega_n)}{\mathbb{P}_n} + \sqrt{\epsilon_n}\sqrt{\sigma}\mathrm{d}\xi_n$$

where $\mathrm{d}\xi_n$ is a centered normal $|\mathcal{F}|$ dimensional vector.
Project $\mathbb{Q}_n$ on $\mathcal{S}_\mathcal{F}$ to obtain $\mathbb{P}_{n+1}$:

$$\mathbb{P}_{n+1} = \pi_{\mathcal{S}_\mathcal{F}}\left(\mathbb{Q}_n\right) = \mathbb{Q}_n + \mathrm{d}z_n$$

$n \mapsto n+1$

Figure 5: Constrained diffusion.

4.3 Projection on $\mathcal{S}_\mathcal{F}$

The natural projection on $\mathcal{S}_\mathcal{F}$ can be computed in a finite number of steps as follows.

1. Define $X^0 = X$, if X^0 does not belong to the hyperplane $\mathcal{H}_\mathcal{F}$, project first X^0 to $\mathcal{H}_\mathcal{F}$:

$$X^1 = \pi_{\mathcal{H}_\mathcal{F}}(X^0)$$

2. – If X^k belongs to $\mathcal{S}_{\mathcal{F}}$, stop the recursion.
   – Else, call J^k the set of integers i such that $X_i^k \leq 0$ and define X^{k+1} by

$$\forall i \in J^k \qquad X_i^{k+1} = 0$$

$$\forall i \notin J^k \qquad X_i^{k+1} = X_i^k + \frac{1}{|\mathcal{F}| - |J^k|} \left(1 - \sum_{j \notin J^k} X_j^k\right)$$

One can show that the former recursion stops in at most $|\mathcal{F}|$ steps (see Gadat, 2004, chap. 4).

5. Experiments

This section provides a series of experimental results using the previous algorithms. Table 1 briefly summarizes the parameters of the several experiments performed.

Data Set	Dim.	$\mathbb{A}$	Classes	Training Set	Test Set
Synthetic	100	N.N.	3	500	100
IRIS	4	CART	3	100	50
Faces	1926	SVM	2	7000	23000
SPAM	54	N.N.	2	3450	1151
USPS	2418	SVM	10	7291	2007
Leukemia	3859	SVM	2	72	$\emptyset$
ARCENE	10000	SVM	2	100	100
GISETTE	5000	SVM	2	6000	1000
DEXTER	20000	SVM	2	300	300
DOROTHEA	100000	SVM	2	800	350
MADELON	500	N.N.	2	2000	600

Table 1: Characteristics of the data sets used in experiments.

5.1 Synthetic example

We start with a simple, but illustrative, small dimensional example.

5.1.1 Synthetic example

Data We consider $|\mathcal{F}| = 100$ ternary variables and 3 classes (similar results can be obtained with more classes and variables). We let $\mathcal{I} = \{-1; 0; 1\}^f$ and the feature $\delta_i(I)$ simply be the ith coordinate of $I \in \mathcal{I}$. Let $\mathcal{G}$ be a subset of $\mathcal{F}$. We define the probability distribution $\mu(\,;\mathcal{G})$ in $\mathcal{I}$ to be the one for which all δ in $\mathcal{F}$ are independent, $\delta(I)$ follows a uniform distribution on $\{-1; 0; 1\}$ if $\delta \notin \mathcal{G}$ and $\delta(I) = 1$ if $\delta \in \mathcal{G}$. We model each class by a mixture of such distributions, including a small proportion of noise. More precisely, for a class C_i, $i = 1, 2, 3$, we define

$$\mu_i(I) = \frac{q}{3}\left(\mu(I; \mathcal{F}_i^1) + \mu(I; \mathcal{F}_i^2) + \mu(I; \mathcal{F}_i^3)\right) + (1 - q)\mu(I; \emptyset)$$

with $q = 0.9$ and

$$\mathcal{F}_1^1 = \{\delta_1; \delta_3; \delta_5; \delta_7\} \qquad \mathcal{F}_1^2 = \{\delta_1; \delta_5\} \qquad \mathcal{F}_1^3 = \{\delta_3; \delta_7\}$$

$$\mathcal{F}_2^1 = \{\delta_2; \delta_4; \delta_6; \delta_8\} \qquad \mathcal{F}_2^2 = \{\delta_2; \delta_4\} \qquad \mathcal{F}_2^3 = \{\delta_6; \delta_8\}$$

$$\mathcal{F}_3^1 = \{\delta_1; \delta_4; \delta_8; \delta_9\} \qquad \mathcal{F}_3^2 = \{\delta_1; \delta_8\} \qquad \mathcal{F}_3^3 = \{\delta_4; \delta_9\}$$

In other words, these synthetic data are generated with almost deterministic values on some variables (which depends on the class the sample belongs to) and with a uniform noise elsewhere. We expect our learning algorithm to put large weights on features in $\mathcal{F}_i^j$ and ignore the other ones. The algorithm $\mathbb{A}$ we use in this case is an M nearest neighbour classification algorithm, with distance given by

$$d(I_1, I_2) = \sum_{\delta \in \omega} \chi_{\delta_1(I) \neq \delta_2(I)}$$

This toy example is interesting because it is possible to compute the exact gradient of $\mathcal{E}$ for small values of M and $k = |\omega|$. Thus, we can compare the stochastic gradient algorithms with the exact gradient algorithm and evaluate the speed of decay of $\mathcal{E}$. Moreover, one can see in the construction of our signals that some features are relevant with several classes (reusables features), some features are important only for one class and others are simply noise on the input. This will permit to evaluate the model of "frequency of goodness" used by OFW.

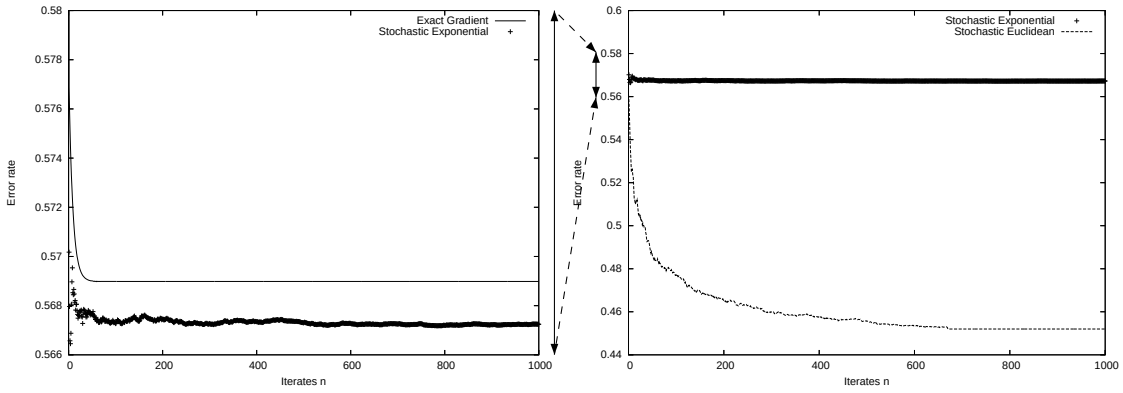

Figure 6: Note that the left and right figures are drawn on different scales. Left: Exact gradient descent (full line) vs. stochastic exponential gradient descent (dashed line) classification rates on the training set. Right: Stochastic Euclidean gradient descent (full line) vs. stochastic exponential gradient descent (dashed line) classification rates on the training set.

Results We provide in figure 6 the evolution of the mean error $\mathcal{E}$ on our training set set against the computation time for exact and stochastic exponential gradient descent algorithms. The exact algorithm is faster but is quickly captured in a local minimum although exponential stochastic descent avoids more traps. Also shown in figure 6, is the fact that the stochastic Euclidean method achieved better results faster than the exponential stochastic approach and avoided more traps than the exponential algorithm to reach lower error rates.

Note that figure 6 (and similar plots in subsequent experiments) is drawn for the comparison of the numerical procedures that have been designed to minimize the training set errors. This does not relate to the generalization error of the final classifier, which is evaluated on test sets.

Finally, figure 7 shows that the efficiency of the stochastic gradient descent and of the reflected diffusion are almost similar in our synthetic example. This has in fact always been so in our experiments: the diffusion is slightly better than the gradient when the latter converges. For this reason, we will only compare the exponential gradient and the diffusion in the experiments which follow.

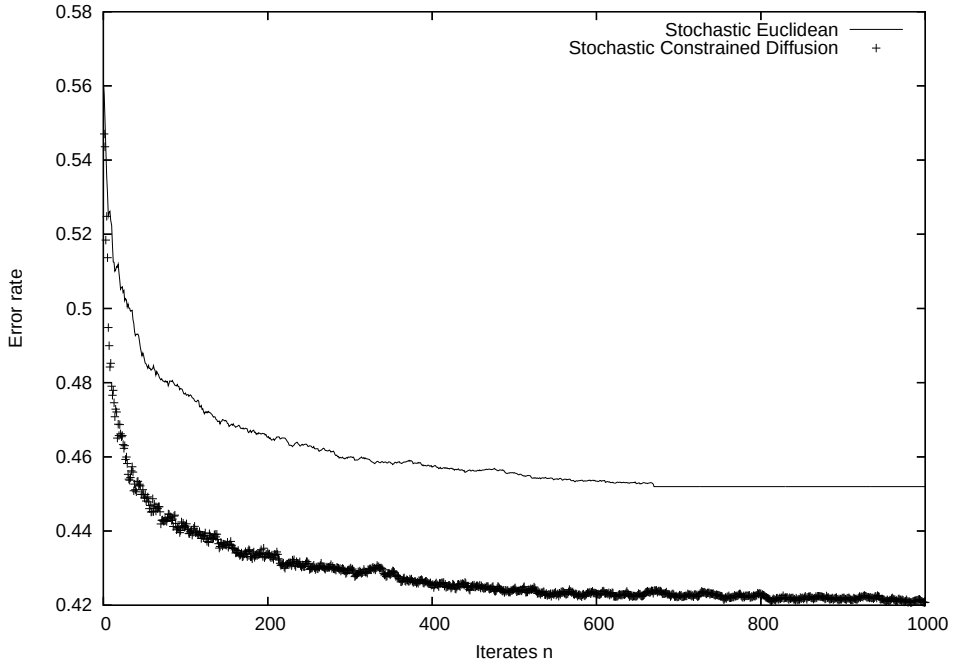

Figure 7: Stochastic Euclidean gradient descent (dashed line) vs. reflected diffusion (full line) classification rate on the training set.

Finally, we summarize this instructive synthetic experiments in figure 8. Remark that in this toy example; the exact gradient descent and the euclidean stochastic gradient (first algorithm of section 4.1.3) are almost equivalent.

In figure 9, we provide the probabilities of the first 15 features in function of $k = |\omega|$. (The graylevel is proportional to the probability obtained at the limit).

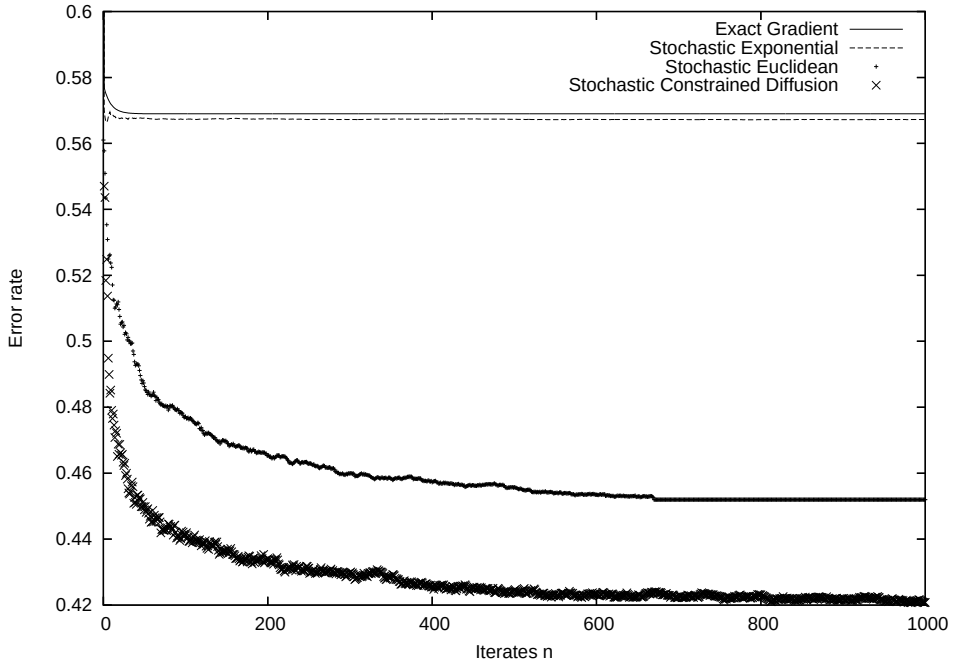

Figure 8: Comparison of the mean error rate computed on the test set with the 4 exact or stochastics gradient descents.

$\mathcal{F}$	δ_1	δ_2	δ_3	δ_4	δ_5	δ_6	δ_7	δ_8	δ_9	δ_{10}	δ_{11}	δ_{12}	δ_{13}	δ_{14}	δ_{15}	...
$\lvert\omega\rvert = 2$																
$\lvert\omega\rvert = 3$																
$\lvert\omega\rvert = 4$																

Figure 9: Probability histogram for several values of $|\omega|$.

Interpretation We observe that the features which are preferably selected are those which lie in several subspaces $\mathcal{F}_i^j$, and which bring information for at least two classes. These are *reusable features*, the knowledge of which being very precious information for the understanding of pattern recognition problems. This result can be compared to selection methods based on information theory. One simple method is to select the variables which provide the most information to the class, and therefore minimize the conditional entropy (see Cover and Thomas, 1991) of the class given each variable. In this example, this conditional entropy is 1.009 for features contained in none of the sets $\mathcal{F}_i^j$, 0.852 for those contained in only one set and approximatively 0.856 for those contained in two of these sets. This implies that this information-based criterion would correctly discard the non-informative variables, but fail to discriminate between the last two groups.

Remark finally that the features selected by OFW after the reusables ones are still relevant for the classification task.

5.1.2 IRIS Database

We use in this section the famous Fisher's IRIS database where data are described by the 4 variables: "Sepal Length", "Sepal Width", "Petal Length" and "Petal Width". Even though our framework is to select features in a large dictionary of variables, it will be interesting to look at the behavior of our algorithm on IRIS since results about feature selection are already known on this classical example. We use here a Classification and Regression Tree (CART) using the Gini index. We extract 2 variables at each step of the algorithm, 100 samples out of 150 are used to train our feature weighting procedure. The figures 10 and 11 describe the behavior of our algorithms (with and without the noise term).

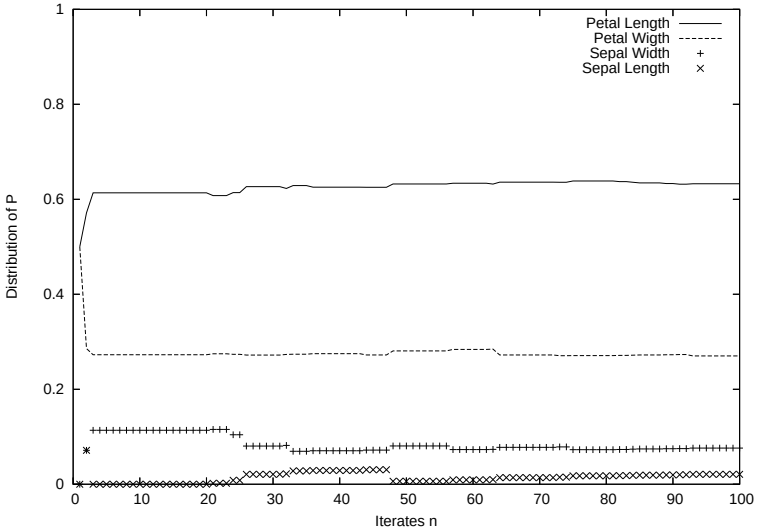

Figure 10: Evolution with n of the distribution on the 4 variables using a stochastic euclidean algorithm.

We remark here that for each one of our two approaches, we approximately get the entire weight on the last two variables " Petal Length" (70%) and "Petal Width" (30%). This result is consistent with the selection performed by CART on this database since we obtain similar results as seen in figure 12. Moreover, a selection based on the Fisher score reaches the same results for this very simple and low dimensional example.

The classification on Test Set is improved selecting two features (with OFW as Fisher Scoring) since we obtain an error rate of 2.6% although without any selection, CART provides an error rate of 4%. In this small low dimensional example, OFW quickly converges to the optimal weight and we obtain a ranking coherent with the selection performed by Fisher Score or CART.

5.2 Real classification problems

We now address real pattern recognition problems. We also compare our results with other algorithms: no selection method, Fisher scoring method, Recursive Feature Elimination method (RFE) (Guyon et al., 2002), L0-Norm for linear support vector machines (Weston

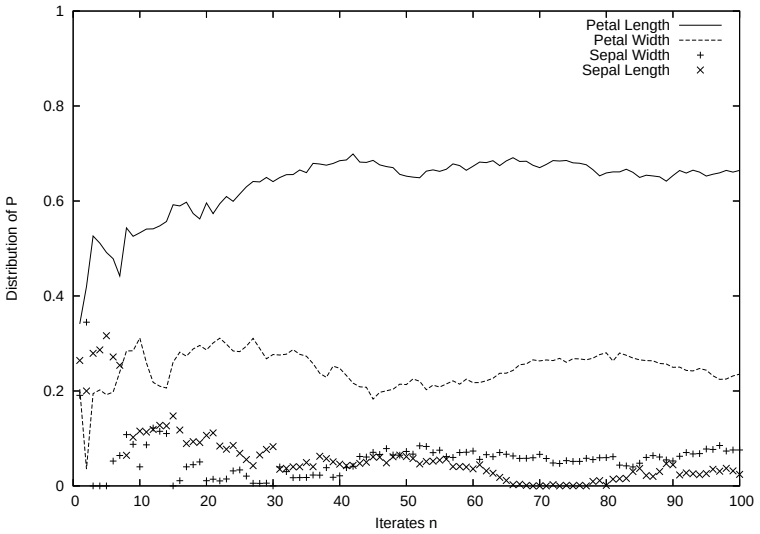

Figure 11: Evolution with n of the distribution on the 4 variables using a stochastic euclidean diffusion algorithm.

et al., 2003) and Random Forests (RF) (Breiman, 2001). We used for these algorithms Matlab implementations provided by the Spider package [1] for RFE and L0-SVM and the random forest package of Leo Breiman [2]. In our experiments, we arbitrarily fixed the number of features per classifier (to 100 for the Faces, Handwritten Digits and Leukemia data and to 15 for the email database). It would be possible to also optimize it, through cross-validation, for example, once the optimal $\mathbb{P}$ has been computed (running this optimization online, while also estimating $\mathbb{P}$ would be too computationally intensive). We have remarked in our experiments that the estimation of $\mathbb{P}$ was fairly robust to to variations of the number of features extracted at each step (k in our notation). In particular, taking k too large does not help much.

5.2.1 Face detection

Experimental framework We use in this section the face database from MIT, which contains 19×19 gray level images; samples extracted from the database are represented in figure 13. The database contains almost 7000 images to train and more than 23000 images to test.

The features in $\mathcal{F}$ are binary edge detectors, as developed in works of Amit and Geman (1999); Fleuret and Geman (2001). This feature space has been shown to be efficient for classification in visual processing. We therefore have as many variables and dimensions as we have possible edge detectors on images. We perform among the whole set of these edge detectors a preprocessing step described in Fleuret and Geman (2001). We then obtain 1926 binary features, each one defined by its orientation, vagueness and localisation.

1. available on http://www.kyb.tuebingen.mpg.de/bs/people/spider/main.html
2. available on http://www.stat.berkeley.edu/users/breiman/RandomForests

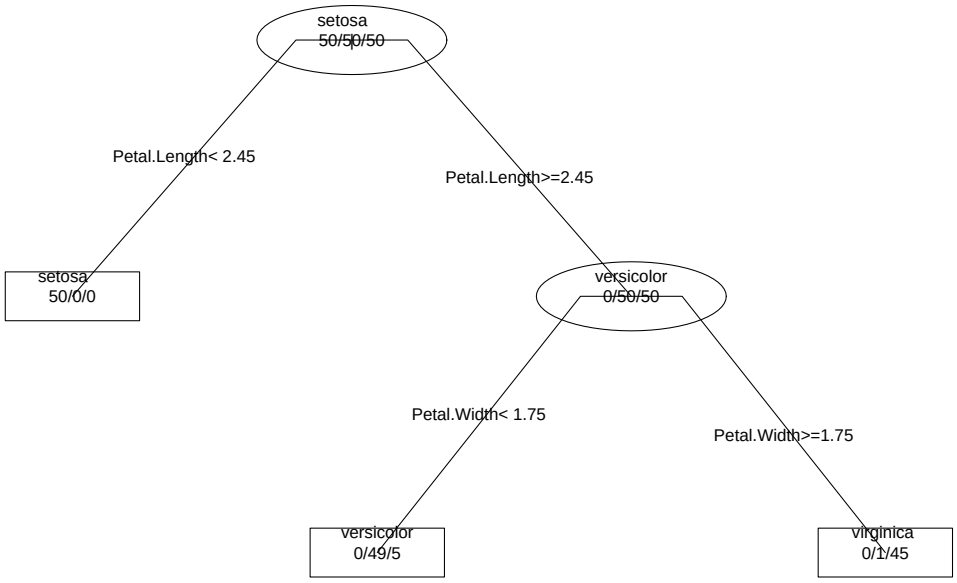

Figure 12: Complete classification tree of IRIS generated from recursive partitioning (CART implementation is using the rpart library of R).

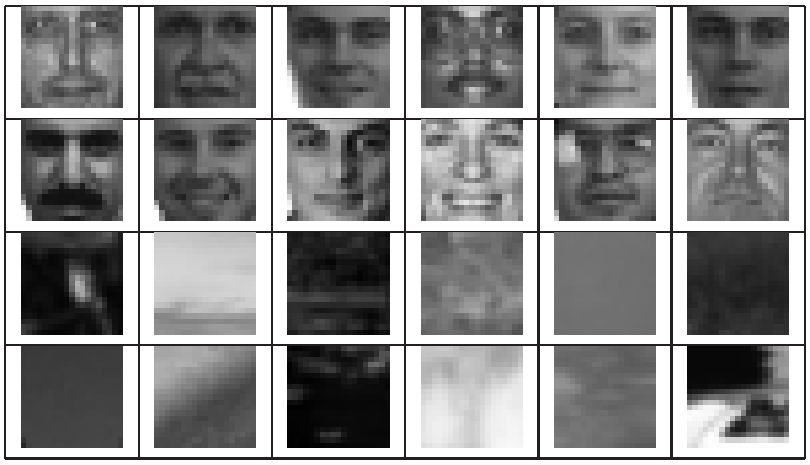

Figure 13: Sample of images taken from MIT database.

The classification algorithm $\mathbb{A}$ which is used here is an optimized version of Linear Support Vector Machines developed by Joachims and Klinkenberg (2000); Joachims (2002) (with linear kernel).

Results We first show the improvement of the mean performance of our extraction method, learned on the training set, and computed on the test set, from a random uniform sampling of features (figure 14).

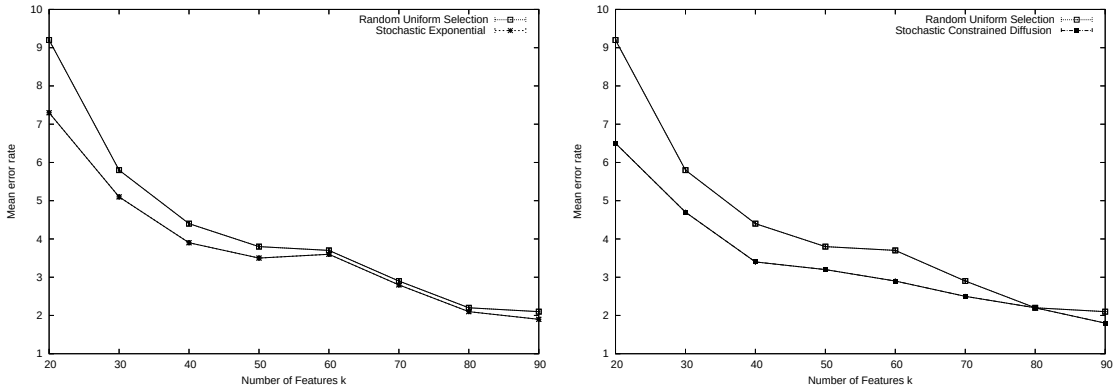

Figure 14: Left: Evolution with k of the average classification error of faces recognition on the test set using a uniform law (dashed line) and $\mathbb{P}_\infty$ (full line), learned with a stochastic gradient method with exponential parameterization. Right: same comparison, for the constrained diffusion algorithm.

Our feature extraction method based on learning the distribution $\mathbb{P}$ improves significantly the classification rate, particularly for weak classifiers ($k = 20$ or 30 for example) as shown in figure 14. We remark again that the constrained diffusion performs better than the stochastic exponential gradient. We achieve a 1.6% error rate after learning with a reflected diffusion, or 1.7% with a stochastic exponential gradient (2% before learning). The analysis of the most likely features (which are the most weighted variables) is also interesting, and occurs in meaningful positions, as shown in figure 15.

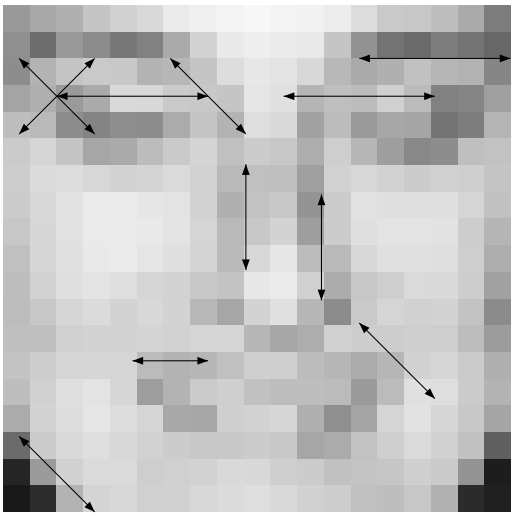

Figure 15: Representation of the main edge detectors after learning.

Figure 16 shows a comparison of the efficiency (computed on the test set) of Fisher, RFE, L0-SVM and our weighting procedure to select features; besides we have shown the performance of $\mathbb{A}$ without any selection and the best performance of Random Forests (as an asymptote).

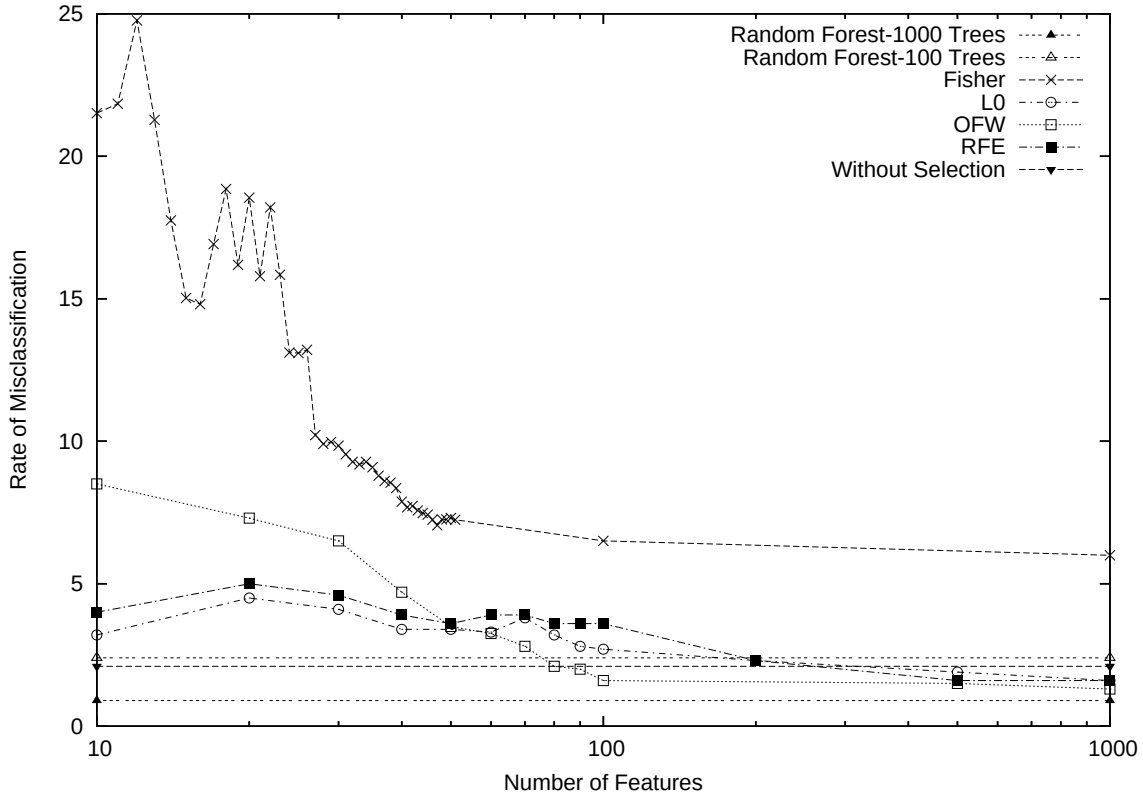

Figure 16: Efficiency of several feature extractions methods for the faces database test set.

We observe that our method is here less efficient with a small number of features (for 20 features selected, we obtain 7.3% while RFE and L0 selections get 4% and 3.2% of misclassification rate). However, for a larger set of features, our weighting method is more effective than other methods since we obtained 1.6% of misclassification for 100 features selected (2.7% for L0 selection and 3.6% for RFE).

The comparison with the Random Forest algorithm is more difficult to estimate: one tree achieves 2.4% error but the length of this tree is more than 1000 and this error rate is obtained by the 3 former algorithms using only 200 features. The final best performance on this database is obtained using Random Forests with 1000 trees. We obtain then a misclassification rate of 0.9%.

5.2.2 SPAM classification

Experimental framework This example uses a text database available at (UCI), which contains emails received by a research engineer from the HP labs, and divided into SPAM

and non SPAM categories. The features here are the rates of appearance of some keywords (from a list of 57) in each text. As the problem is quite simple using the last 3 features of the previous list, we choose to remove these 3 variables (which depends on the number of capital letters in an email), we start consequently with a list of 54 features. We use here a 4-nearest neighbor algorithm and we extract 15 features at each step. The database is composed by 4601 messages and we use 75% of the email database to learn our probability $\mathbb{P}_\infty$, representing our extraction method while the 25% samples of data is left to form the test set.

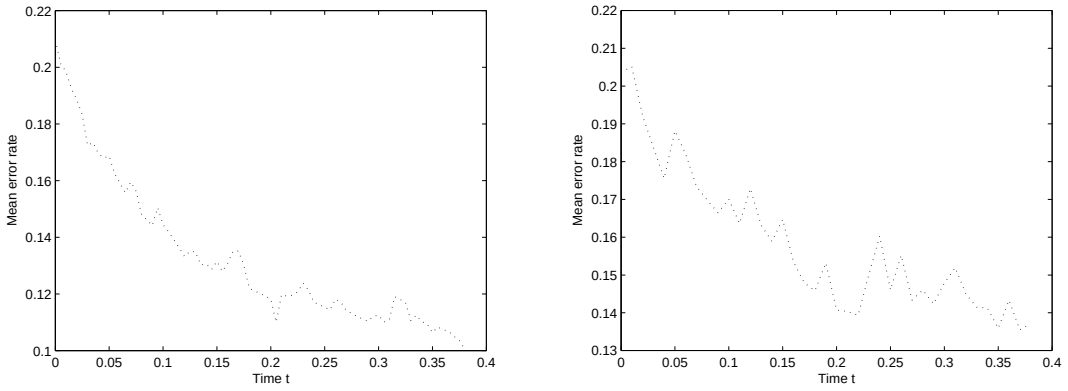

Figure 17: Time evolution of the energy $\mathcal{E}_1$ for the spam/email classification UCI computed on the test set, using a stochastic gradient descent with an exponential parameterization (left) and with a constrained diffusion (right).

Results We plot the average error on the test set in figure 17. On our test set, the method based on the exponential parameterization achieves better results than those obtained by reflected diffusion which is slower because of our Brownian noise. The weighting method is here again efficient in improving the performances of the Nearest Neighbour algorithm.

Moreover, we can analyze the words selected by our probability $\mathbb{P}_\infty$. In the next table, two columns provide the features that are mainly selected. We achieve in a different way similar results to those noticed in Hastie et al. (2001) regarding the ranking importance of the words used for spam detection.

The words which are useful for spam recognition (left columns) are not surprising ("business", "remove", "receive" or "free" are highly weighted). More interesting are the words selected in the right column; these words are here useful to enable a personal email detection. Personal informations like phone numbers ("650", "857") or first name ("george") are here favored to detect real email messages. The database did not provide access to the original messages, but the importance of the phone numbers or first name is certainly due to the fact that many non-spam messages are replies to previous messages outgoing from the mailbox, and would generally repeat the original sender's signature, including its first name, address and phone number.

Words favored for SPAM	Frequency	Words favored for NON SPAM	Frequency
remove	8.8%	cs	5.4%
business	8.7%	857	4.6%
[	6%	415	4.4%
report	5.9%	project	4.3%
receive	5.6%	table	4.2%
internet	4.4%	conference	4.2%
free	4.1%	lab	3.9%
people	3.7%	labs	3.2%
000	3.6%	edu	2.8%
direct	2.3%	650	2.7%
!	1.2%	85	2.5%
$	1%	george	1.6%

Figure 18: Words mostly selected by $\mathbb{P}_\infty$ (exponential gradient learning procedure) for the spam/email classification.

We compare next the performances obtained by our method with RFE, RF and L0-SVM. Figure 19 show relative efficiency of these algorithms on the spam database.

Without any selection, the linear SVM has more than 15% error rate while each one of the former feature selection algorithms achieve better results using barely 5 words. The best algorithm is here the L0-SVM method, while the performance of our weighting method (7.47% with 20 words) is located between RFE (11.1% with 20 words) and L0-SVM (4.47% with 20 words). In addition, RF high performance is obtained using a small forest of 5 trees (not as deep as in the example of faces recognition) and we obtain with this algorithm 7.24% of misclassification rate using trees of size varying from 50 to 60 binary tests.

5.2.3 Handwritten number recognition

Experimental framework A classical benchmark for pattern recognition algorithms is the classification of handwritten numbers. We have tested our algorithm on the USPS database (Hastie et al. (2001); **?**): each image is a segment from a ZIP code isolating a single digit. The 7291 images of the training set and 2007 of the test set are 16×16 eight-bit grayscale maps, with intensity between 0 and 255. We use the same feature set, $\mathcal{F}$, as in the faces example. We obtain a feature space $\mathcal{F}$ of 2418 edge detectors with one precise orientation, location and blurring parameter. The classification algorithm $\mathbb{A}$ we used is here again a linear support vector machine.

Results Since our reference wrapper algorithms (RFE and L0-SVM) are restricted to 2 class problems, we present only results obtained on this database with the algorithm $\mathbb{A}$ which is a SVM based on the "one versus all" idea.

The improvement of the detection rate is also similar to the previous example, as shown in figure 21. We first plot the mean classification error rate before and after learning the probability map $\mathbb{P}$. These rates are obtained by averaging $g(\omega)$ over samples of features uniformly distributed on $\mathcal{F}$ in the first case, and distributed according to $\mathbb{P}$ in the second

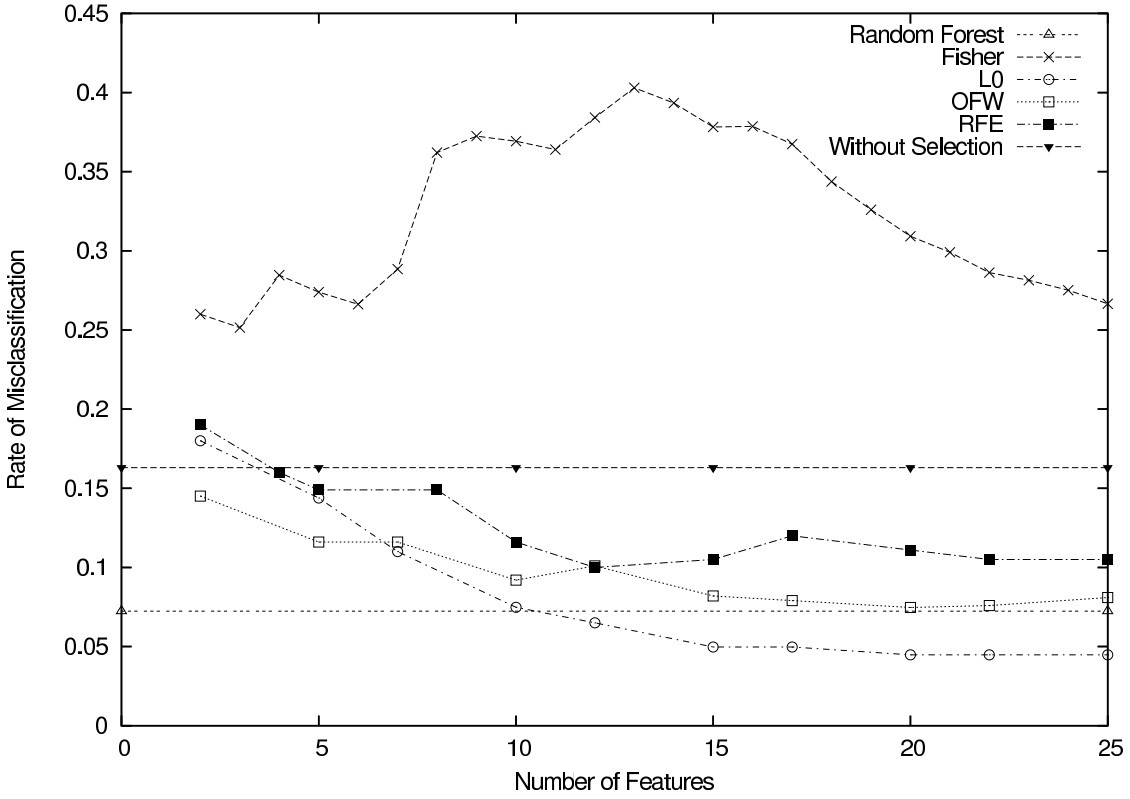

Figure 19: Efficiency of several feature extractions methods on the test set for the SPAM database.

Class	$\mathcal{C}_0$	$\mathcal{C}_1$	$\mathcal{C}_2$	$\mathcal{C}_3$	$\mathcal{C}_4$	$\mathcal{C}_5$	$\mathcal{C}_6$	$\mathcal{C}_7$	$\mathcal{C}_8$	$\mathcal{C}_9$
Image I										
										

Figure 20: Sample of images taken from the USPS database.

case. These numbers are computed on *training data* and therefore serve for evaluation of the efficiency of the algorithm in improving the energy function from $\mathcal{E}_1(\mathcal{U}_\mathcal{F})$ to $\mathcal{E}_1(\mathbb{P}_\infty)$. Figure 21 provides the variation of the mean error rate in function of the number of features k used in each ω. The ratio between the two errors (before and after learning) rates, is around 90% independently on the value of k.

Figure 22 provides the result of the classification algorithm (using the voting procedure) on the test set. The majority vote is based on 10 binary SVM-classifiers on each binary classification problem C_i vs. $I \setminus C_i$. The features are extracted first with the uniform distribution $\mathcal{U}_\mathcal{F}$ on $\mathcal{F}$, then using the final $\mathbb{P}_\infty$.

The learning procedure significantly improves the performance of the classification algorithms. The final average error rate on the USPS database is about 3.2% for 10 elementary classifiers per class $\mathcal{C}_i$, with 100 binary features per elementary classifier. The performance

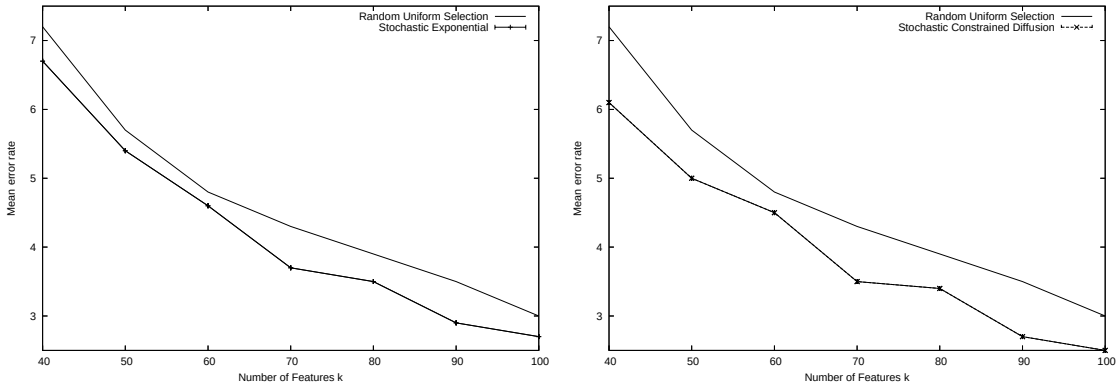

Figure 21: Mean error rate over the training set USPS for k varying from 40 to 100, before (dashed line) and after (full line) a stochastic gradient learning based on exponential parameterization (left) and constrained diffusion (right).

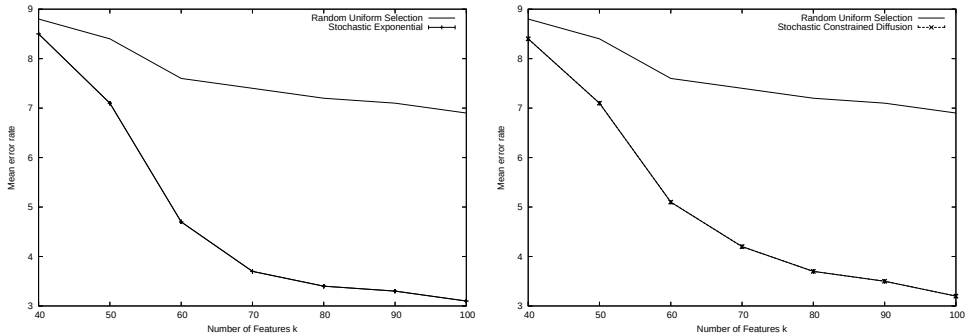

Figure 22: Evolution with k of the mean error of classification on the test set, extraction based on random uniform selection (dashed line) and $\mathbb{P}_\infty$ selection (full line) for USPS data, learning computed with stochastic gradient using exponential parameterization (left) and constrained diffusion (right).

is not as good as the one obtained by the tangent distance method of Simard and LeCun (1998) (2.7% error rate of classification), but we here use very simple (edge) features. And the result is better, for example, than linear or polynomial Support Vector Machines (8.9% and 4% error rate) computed without any selection and than sigmoid kernels (4.1%) (see **?**) with a reduced complexity (measured, for example by the needed amount of memory).

Since the features we consider can be computed at every location in the image, it is interesting to visualize where the selection has occurred. This is plotted in figure 23, for the four types of edges we consider (horizontal, vertical and two diagonal), with grey levels proportional to the value of $\mathbb{P}_\infty$ for each feature.

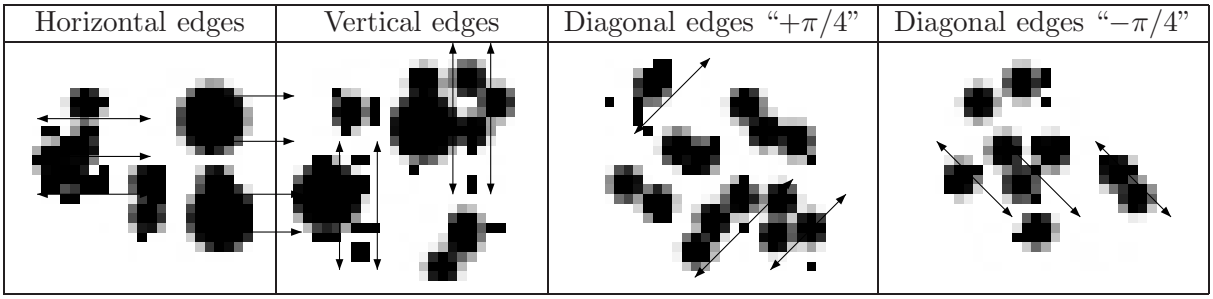

Figure 23: Representation of the selected features after a stochastic exponential gradient learning for USPS digits. Greyscales are proportional to weights of features

5.2.4 Gene selection for Leukemia *AML-ALL* recognition

Experimental framework We carry on our experiments with feature selection and classification for microarray data. We have used the Leukemia Database AML-ALL of Golub et al. (1999). We have a very small number of samples (72 signals) described by a very large number of genes. We run a preselection method to obtain the database used by Deb and Reddy (2003)[3] that contains 3859 genes. Our algorithm $\mathbb{A}$ is here a linear support vector machines. As we face a numerical problem with few samples on each class (AML and ALL), we decide to benchmark each of the algorithms we have tested using a 10-fold cross validation method.

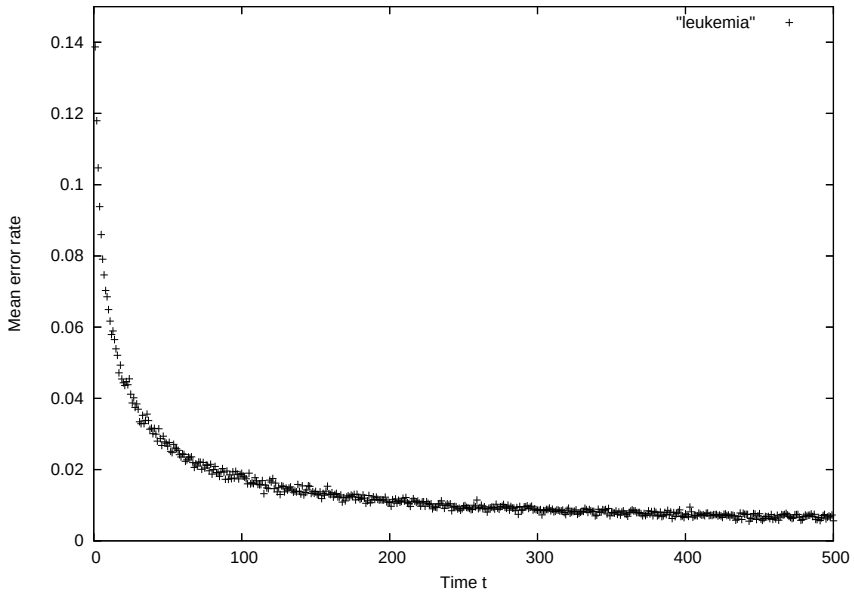

Figure 24: Evolution of the mean energy $\mathcal{E}$ computed by the constrained diffusion method on the training set with time for $k = 100$.

3. available on http://www.iitk.ac.in/kangal/

Results Figure 24 shows the efficiency of our method of weighting features in reducing the mean error $\mathcal{E}$ on the training set. We remark that with random uniform selection of 100 features, linear support vector machines obtain a poor rate larger than 15% while learning $\mathbb{P}_\infty$, we achieve a mean error rate less than 1%.

We now compare our result to RFE, RF and L0-SVM using the 10-fold cross validation method. Figure 25 illustrates this comparison between these former algorithms. In this

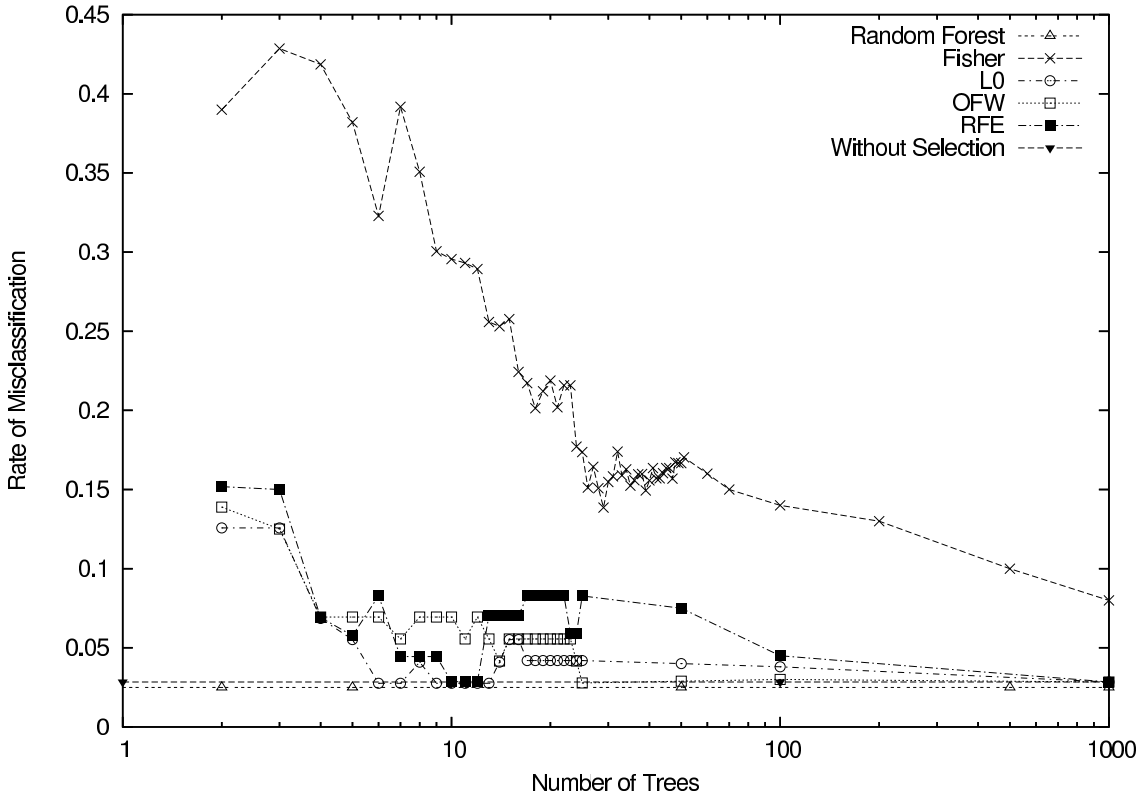

Figure 25: Efficiency of several feature extractions methods for the Leukemia database. Performances are computed using 10 CV.

example, we obtain better results without any selection, but in fact the classification of one linear SVM does not permit to rank features by importance effect on the classification task. We note here again that our weighting method is less effective for short size subsets (5 genes) while our method is competitive with larger subsets (20-25 genes). Here again, we note that L0-SVM outperforms RFE (like in the SPAM study section 5.2.2). Finally, the Random Forest algorithm obtains results which are very irregular in connection with the number of trees as one can see in figure 26.

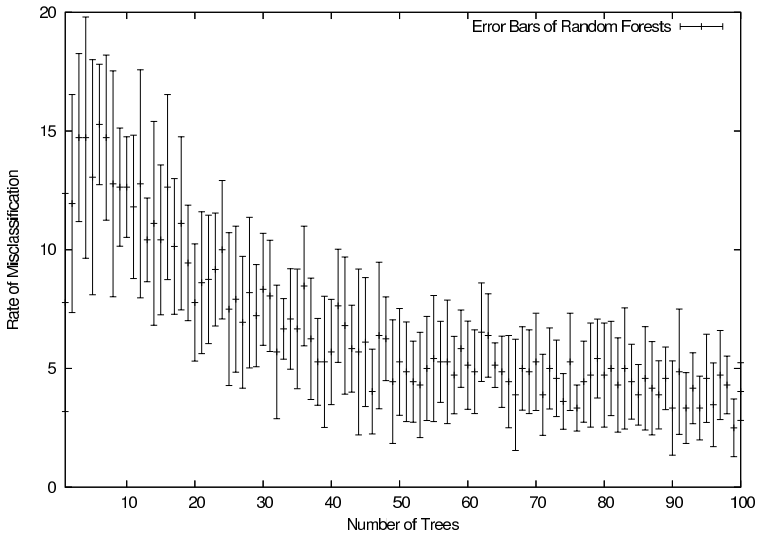

Figure 26: Error bars of Random Forests in connection with the number of trees used computed by cross-validation.

5.2.5 Feature Selection Challenge

We conclude our experiments with results on the Feature Selection Challenge (Guyon et al., 2004) [4]. Datasets cover a large field of the feature selection problem since examples are taken in several areas (Microarray data, Digit classification, Synthetic examples, Text recognition and Drug discovery). Results are provided using the Balanced Error Rate (BER) obtained on the validation set rather than the classical error rate.

We first performed a direct Optimal Feature Weighting algorithm on theses datasets without any feature preselection using a linear SVM for our base classifier $\mathbb{A}$. For four of the five datasets (DEXTER, DOROTHEA, GISETTE and MADELON) the numerical performances of the algorithm are significantly improved if a variable preselection is performed before running it. This preselection was based on the Fisher Score:

$$F_i = \frac{(\overline{x}_i^1 - \overline{x}_i)^2 + (\overline{x}_i^2 - \overline{x}_i)^2}{\frac{1}{n_1 - 1}\sum_{k=1}^{n_1}(x_{k,i}^1 - \overline{x}_i^1)^2 + \frac{1}{n_2 - 1}\sum_{k=1}^{n_2}(x_{k,i}^2 - \overline{x}_i^2)^2}.$$

Here n_1 and n_2 are the numbers of samples of the training set of classes 1 and 2, $\overline{x}_i^1, \overline{x}_i^2$ and $\overline{x}_i$ assign the mean of feature i on class 1, 2 and over the whole training set. We preselect the features with Fisher Score higher than 0.01.

We then perform our Optimal Feature Weighting algorithm with the new set of features obtained by the Fisher preselection using for $\mathbb{A}$ a support vector machine with linear kernel. Figure 27 show the decreasing evolution of the mean BER on the training set for each datasets of the feature selection challenge. One instantaneously can see that OFW is much

4. datasets are available on http://www.nipsfsc.ecs.soton.ac.uk/datasets/

more efficient on GISETTE or ARCENE than on other datasets since the evolution of mean BER is faster and have a larger amplitude.

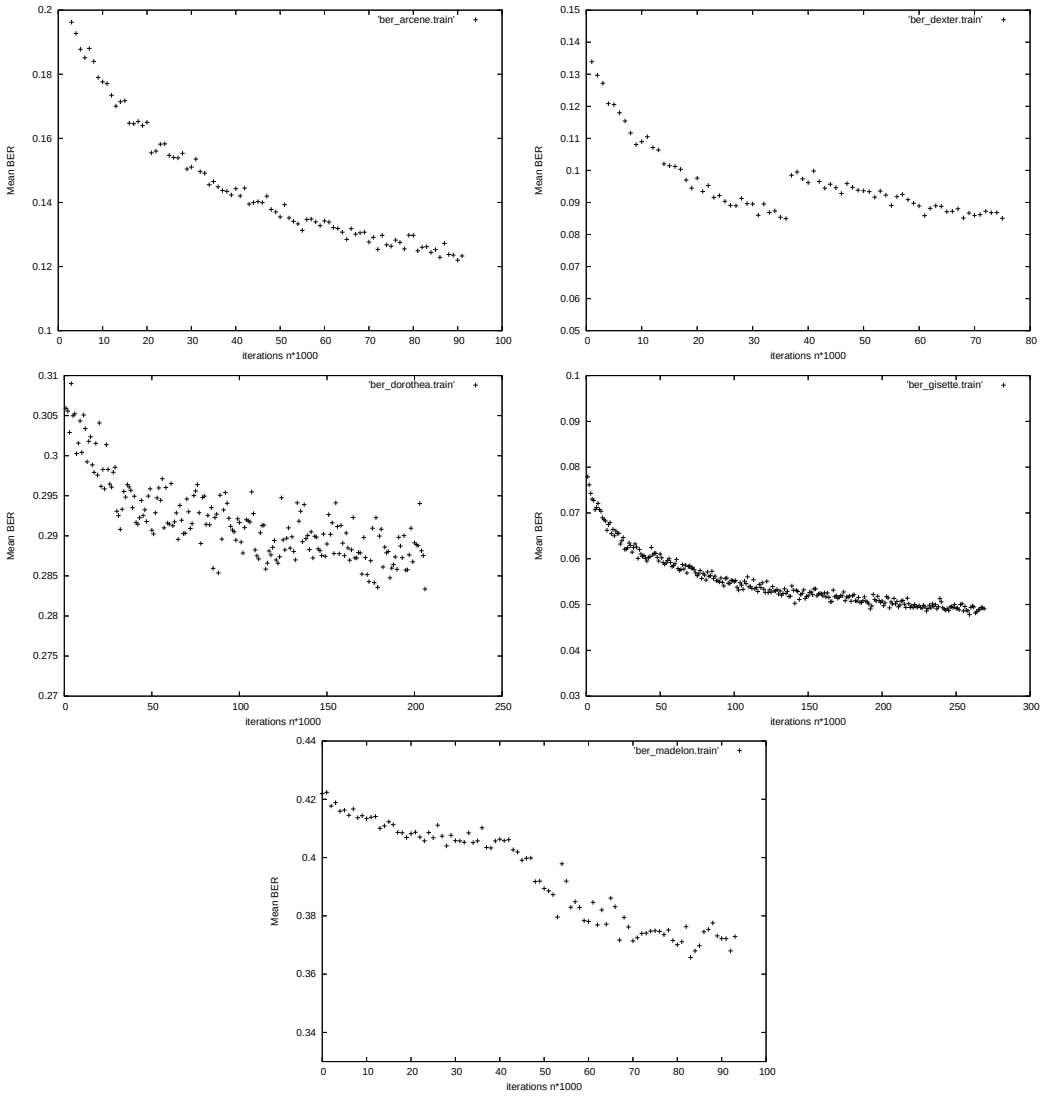

Figure 27: Evolution with iterations n of the Balanced Error Rate on the training set.

For computational efficiency, our weight distribution $\mathbb{P}$ is learned using a linear SVM for the basic algorithm, $\mathbb{A}$. Once this is done, an optimal nonlinear SVM is used for the final classification (the parameters of the kernel of this final SVM being estimated via cross-validation). We select the number of features used for the classification on the validation set on the basis of a 10-fold cross validation procedure on the training set. Table 2 summarizes results obtained by our OFW algorithm and, Linear SVM [5] and others algorithms of feature selection (Transductive SVM of Wu and Li (2003), combined filter methods with svm as

5. Reader can refer at http://www.nipsfsc.ecs.soton.ac.uk/results for other results obtained by feature selection procedure or different classification algorithms

F+SVM of Chen and Lin (2006) and FS+SVM of Lal et al. (2006), G-flip of Gilad-Bachrach et al. (2004), Information-Based Feature Selection of Lee et al. (2006), and analysis of redundancy and relevance (FCBF) of Yu and Liu (2004)). We select these methods since they are meta algorithms (as OFW method) whose aim is to optimize the feature subset entry of standard algorithms. These results are those obtained on the Validation Set since most of the papers previously cited do not report results on the Test Set. One can show that most of these methods outperform the performance of SVM without any selection.

Fisher criterion is numerically effective and can exhibit very reduced sets of features, but using it alone provides performances that are below those reported in table 2. FS+SVM and F+SVM, like all filter approaches, are equally effective and perform quite well, but requires the estimation of several thresholds, and suffer from lack of theoretical optimization background. The G-flip algorithm is to find a growing sequence of features that successively maximize the margin of the classifier. The main idea is consequently not so far from the OFW approach, even though we pick up a new feature subset at each iteration. Results are comparable with OFW and authors obtained generalization error bounds. The Transductive SVM incorporates a local optimization on the training set for a cost function related to the performances of SVM, and updates a parameter defined on each feature using coefficients of the hyperplanes constructed at each step. This approach has the drawback of high computation cost, can fail in the local optimization step and requires to tune many parameters, but obtains interesting results and suggests further developments on model selection. FCBF, which does not intend directly to increase the accuracy of any classifier as a wrapper algorithm, selects the features by identifying the redundancy between features and relevance analysis of variables. The resulting algorithms (FCBF-NBC and FCBF-C4.5) obtains very good results and is numerically simple to handle. However, this approach does not provide any theoretical measure of efficiency selection with respect to the accuracy of classification.

Our method is competitive on all datasets but DOROTHEA. The OFW algorithm is particularly good on the GISETTE dataset. Moreover, we outperform most of methods based on a filter + svm approach.

Table 3 provides our results on the Test Set as well as the results of the best challenge participants. Looking at BER, best results of the challenge outperforms our OFW approach, but this comparison seems unfair since best entries are classification algorithm rather than features selection algorithm (most of features are kept to treat the data) and the difference of BER is not statistically significantly different except for the DOROTHEA database. We add moreover recent results on these 5 datasets obtained by quite simple filter methods Guyon et al. (2006) that reach remarkable BER results.

6. Discussion and conclusion

6.1 Discussion

From the previous empirical study, we can conclude that OFW can dramatically reduce the dimension of the feature space while preserving the accuracy of classification and even enhance it in many cases. We observe likewise that we obtain results comparable to those of reference algorithms like RFE or RF. In most cases, the learning process of $\mathbb{P}_\infty$ is numerically easy to handle with our method and the results on test set are convincing. Besides the

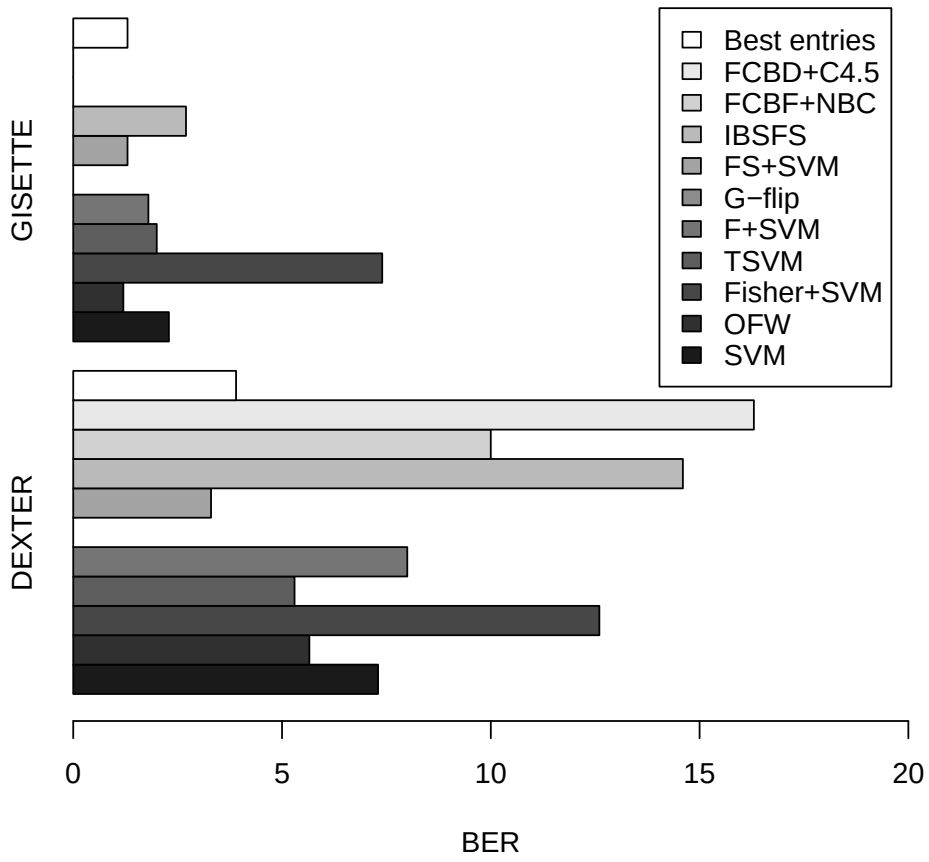

Figure 28: Performances of several algorithms on the Validation Set of the FSC. Zero bar correspond to missing values.

accuracy of classifier, another interesting advantage of OFW is the stability of the subsets which are selected when we run several bootstrap version of our algorithm. Further works could include numerical comparisons on the stability of several algorithms using for instance a bootstrap average of Hamming distances as it is performed in Dune et al. (2002).

Nevertheless, in some rare cases (DEXTER or DOROTHEA), learning the optimal weights is more complicated: in the case of DEXTER database, we can guess from figure 27 that our stochastic algorithm has been temporarily trapped in a neighborhood of a local minimum of our energy $\mathcal{E}$. Even if the OFW has succeeded in escaping the local minimum after a while, this still reduces drastically the convergence speed and the final performance of classification on the validation set. In the case of DOROTHEA, the results of SVM are quite irregular according to subsets selected along time (see figure 27) and the final performance,

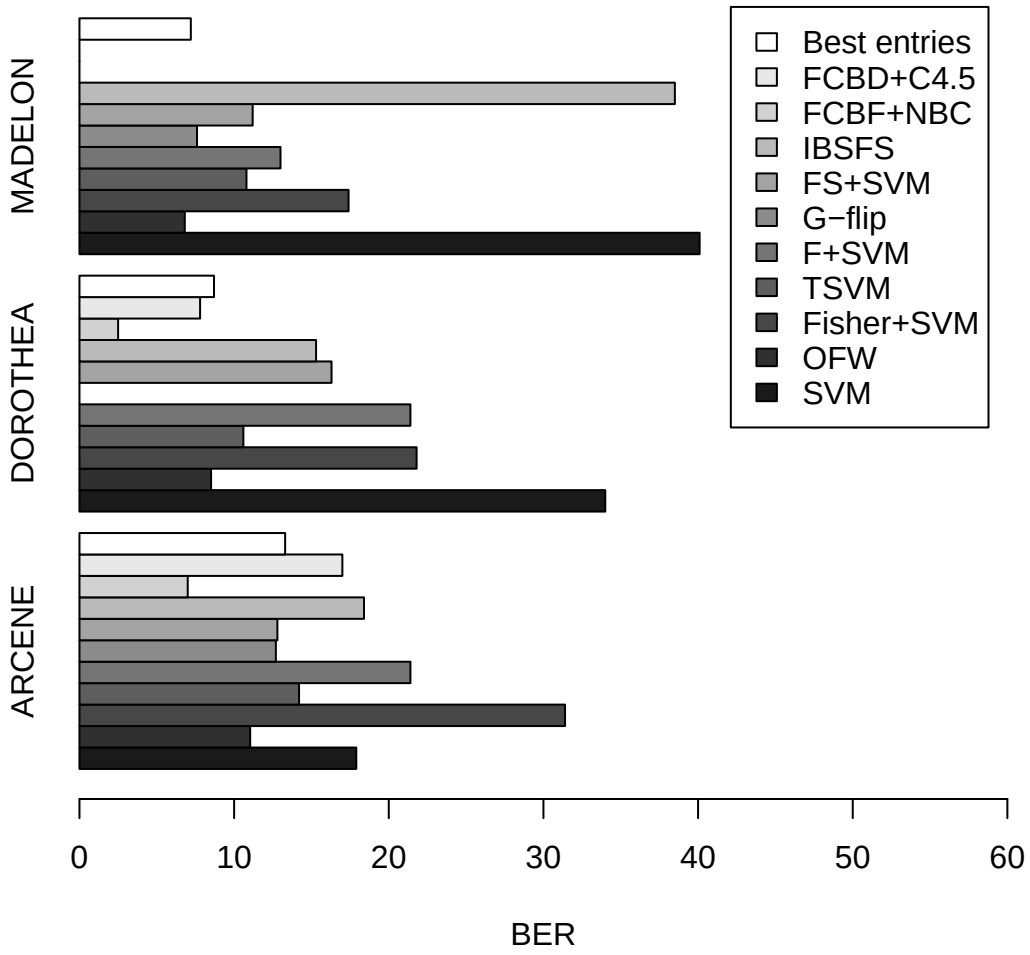

Figure 29: Performances of several algorithms on the Validation Set of the FSC. Zero bar correspond to missing values.

as all methods based on SVM classifiers noticed in table 2, is not as good as other reported for OFW (see best BER of the challenge in table 3 obtained without using any SVM as final classifier). At last, results obtained by OFW are a little bit worse than those obtained by filtering techniques Guyon et al. (2006) (see table 3) that perform efficient feature selection. Note also that all the results obtained require larger feature subsets than OFW and use a larger amount of probes in the set of selected features. To make a comparison of their efficiency, we compute the subsets obtained by these filtering methods with the number

Dataset	ARCENE	DEXTER	DOROTHEA	GISETTE	MADELON
BER of SVM	17.86	7.33	33.98	2.10	40.17
BER of OFW	11.04	5.67	8.53	1.1	6.83
BER of Fisher + SVM	31.42	12.63	21.84	7.38	17.4
% features selected	(3.80)	(1.43)	(0.01)	(6.54)	(2.80)
BER of TSVM	14.2	5.33	10.63	2	10.83
% features selected	(100)	(29.47)	(0.5)	(15)	(2.60)
BER of F+SVM	21.43	8	21.38	1.8	13
% features selected	(6.66)	(1.04)	(0.445)	(18.2)	(2.80)
BER of G-flip	12.66				7.61
% features selected	(0.76)				(3.60)
BER of FS+SVM	12.76	3.3	16.34	1.3	11.22
% features selected	(47)	(18.6)	(1)	(34)	(4)
BER of IBFS	18.41	14.60	15.26	2.74	38.5
% features selected	(1.85)	(5.09)	(0.77)	(9.30)	(2.40)
BER of FCBF+NBC	7	10	2.5		
% features selected	(0.24)	(0.17)	(0.5)		
BER of FCBF+C4.5	17	16.3	7.8		
% features selected	(0.24)	(0.17)	(0.5)		

Table 2: Performances of OFW and other meta algorithms on the Validation Set of the FSC, BER are given in percentage. The best results are in bold characters. The second line compares OFW with the simple Fisher scoring method with the same amount of features and show the error bars obtained by OFW.

Dataset	ARCENE	DEXTER	DOROTHEA	GISETTE	MADELON
BER of OFW/(% features)	11.54/(3.80)	4.8/(1.31)	14.43/(0.04)	1.35/(8.18)	6.78/(3.2)
Best BER of the challenge	13.3/(100)	3.9/(1.52)	8.7/(100)	1.3/(100)	7.2/(100)
BER of Guyon et al. (2006)	10.48/(14)	3.25/(22.5)	9.3/(0.7)	0.98/(15.68)	6.22/(4)
BER of Filters	14.21/(3.80)	4.8/(1.31)	41.33/(0.04)	4.54/(8.18)	7.33/(3.2)

Table 3: Performances of OFW on the Test Set of the FSC, BER are given in percentage.

Dataset	ARCENE	DEXTER	DOROTHEA	GISETTE	MADELON
Probes of OFW	0.79	19.47	2.56	0	0
Probes of best challenge entry	30	12.87	50	50	96
Probes of Guyon et al. (2006)	0.36	55.38	22.14	49.23	0
Probes of Filters methods	7.63	54.58	33.33	45.97	0

Table 4: Fractions of probes selected by OFW and other algorithms on the FSC, fractions are given in percentage.

of features used for OFW. These results are reported in the last line of table 3. One can see that filter methods obtained poorer performances with a reduced number of features, one can note that on the DOROTHEA dataset, the SVM completely miss one of the two unbalanced class and obtained bad results.

Another point of interest is the fraction of probes finally selected by each methods. Probes are artificial features added at random in each datasets with statistical distributions similar to the one of some real features, but these probes do not carry any information on the class labels of signals. Thus, a good feature selection algorithm should obtained a small fraction of probes on the final selection. We show in table 4 the fraction of probes obtained by the methods cited in table 3. One can remark that OFW is particularly effective to reduce the amount of probes of any datasets (see GISETTE for instance).

Better results of OFW have been obtained for two special cases of databases which are microarray data (ARCENE and Leukemia) and image recognition data (USPS, GISETTE and Faces). SVM initially performs well on this datasets, but OFW significantly improve the performance without any selection.

More generally, OFW seems to behave well and to boost accuracy of algorithms $\mathbb{A}$ that have initial performance that vary smoothly with respect to small changes in the feature set. Even if our learning procedure is computed in a very large dimensional training set, recent work have shown that in the context of classification, bootstrap approaches (as it is done in OFW) do not introduce a supplementary important bias (Singhi and Liu (2006)) and it is equally what we can conclude in our case.

In the first synthetic example, one can make the important remark that reusable features are mainly favored by OFW. The algorithm classes those which are relevant, but not reusable in a second group. This point looks favorable to our model of "frequency of use".

Our approach does not address the issue of redundancy (two similar features would most likely receive the same weight). Some ideas in how to take this into account are sketched in the concluding section.

6.2 Computational considerations

We have performed our experiments using a C++ compiler with a 2.2 GHz 1 Go RAM processor pentium IV PC on a Debian system. The learning time of OFW mostly depends on the initial number of variables in the feature space and the step of our stochastic scheme; for the Leukemia database which contains 3859 genes, learning took about one hour.

However, OFW can be easily implemented with parallel techniques since at each step of the stochastic procedure, one can test several subsets of the feature set still using the same update formula of $\mathbb{P}_n$. Moreover, we remark that it can be effective for the calculation time to first filter out very irrelevant features (selected for instance by a Fisher Score) and run the OFW procedure.

Another option would be to use algorithm $\mathbb{A}$ simpler than SVM, CART or NN, based on basic statistical tools as likelihood or mutual information whilst performing a final decision with SVM for instance.

6.3 Conclusion, Future work

Our approach introduces a mathematical model to formalize the search for optimal features. Our selection of features is done by learning a probability distribution on the original feature set, based on a gradient descent of an energy $\mathcal{E}$ within the simplex $\mathcal{S}_{\mathcal{F}}$.

The numerical results show that the performance is significantly improved over an initial rule in which features are simply uniformly distributed. Our Optimal Feature Weighting method is moreover competitive in comparison with other feature selection algorithms and leads to an algorithm which does not depend on the nature of the classifier $\mathbb{A}$ which is used, whereas, for instance, RFE or L0-SVM are only based on SVM.

Our future work will enable the feature space $\mathcal{F}$ to be also modified during the algorithm, by allowing for combination rules, creation and deletion of tests, involving a hybrid evolution in the set of probability measures and in the feature space. This will be implemented as a generalization of our constrained diffusion algorithm to include jumps in the underlying feature space.

Another development would be to speed up the learning procedure using stochastic algorithm techniques to handle even larger databases without using a combination of filter and wrapper method.

Acknowledgements

We thank the anonymous referees for their helpful comments that permit to significantly complete and improve both the experimental section and the general clarity of this manuscript.

References

Yali Amit and Donald Geman. A computational model for visual selection. *Neural computation*, 11: 1691 – 1715, 1999.

Yali Amit and Donald Geman. Shape quantization and recognition with randomized trees. *Neural Computation*, 9(7):1545–1588, 1997.

Michel Benaïm. Convergence with probability one of stochastic approximation algorithms whose average is cooperative. *Nonlinearity*, 13(3):601–616, 2000.

Michel Benaim. A dynamical system approach to stochastic approximations. *SIAM J. Control Optim.*, 34(2):437–472, 1996.

Albert Benveniste, Michel Métivier, and Pierre Priouret. *Adaptive algorithms and stochastic approximations*, volume 22 of *Applications of Mathematics (New York)*. Springer-Verlag, Berlin, 1990.

José Bins and Bruce A. Draper. Feature selection from huge feature sets. In *Proceedings of the Eighth International Conference On Computer Vision*, volume 2, pages 159–165, 2001.

Avrim Blum and Ronald L. Rivest. Training a 3-node neural network is NP-complete. *Neural Networks*, 5:117– 127, 1992.

Leo Breiman. Arcing classifiers. *The Annals of Statistics*, 26(3):801–849, 1998.

Leo Breiman. Random forests. *Machine Learning*, 1:5–32, 2001.

Robert Buche and Harold J. Kushner. Rate of convergence for constrained stochastic approximationalgorithms. *SIAM Journal on Control Optimization*, 40(4):1011–1041, 2001.

Olivier Chapelle, Vladimir Vapnik, Olivier Bousquet, and Sayan Mukherjee. Choosing multiple parameters for support vector machines. *Machine Learning*, 46(1):131–159, 2002.

Yi-Wei Chen and Chih-Jen Lin. Combining SVMs with various feature selection strategies. In *Feature extraction, foundations and applications*. Springer, Berlin, 2006.

Shai Cohen, Eytan Ruppin, and Gideon Dror. Feature selection based on the shapley value. In *International Joint Conference on Artificial Intelligence*, pages 665–670, 2005.

Thomas Cover and Joy Thomas. *Information Theory*. Wiley, New York, 1991.

Kalyanmoy Deb and Raji Reddy. Reliable classification of two-class cancer data using evolutionary algorithms. *BioSystems*, 1:111 – 129, 2003.

Thomas G. Dietterich and Ghulum Bakiri. Solving multiclass learning problems via error-correcting output codes. *Journal of Artificial Intelligence Research*, 2:263–286, 1995.

Marie Duflo. *Algorithmes stochastiques*, volume 23 of *Mathématiques & Applications (Berlin) [Mathematics & Applications]*. Springer-Verlag, Berlin, 1996.

Kevin Dune, Padraig Cunningham, and Francisco Azuaje. Solutions to instability problems with sequential wrapper-based approaches to feature selection. In *Technical Report-2002-28*, Department of Computer Science Courses, Trinity College, Dublin, 2002.

Paul Dupuis and Hitoshi Ishii. On Lipschitz continuity of the solution mapping to the Skorokhod problem, with applications. *Stochastics and Stochastics Reports*, 35(1):31–62, 1991.

Paul Dupuis and Kavita Ramanan. Convex duality and the Skorokhod problem. I, II. *Probability Theory and Related Fields*, 115(2):153–195, 197–236, 1999.

Francois Fleuret. Fast binary feature selection with conditional mutual information. *Journal of Machine Learning Research*, (5):1531–1555, 2004.

Francois Fleuret and Donald Geman. Coarse-to-fine face detection. *International Journal of Computer Vision*, 41(1-2):85–107, 2001.

Jerome Friedman, Trevor Hastie, and Robert Tibshirani. Additive logistic regression: a statistical view of boosting. *The Annals of Statistics*, 28(2):337–407, 2000.

Sébastien Gadat. Apprentissage d'un vocabulaire symbolique pour la détéction d'objets dans une image. *PhD thesis, École Normale Supérieure de Cachan, 94235 Cedex, France*, 2004.

Stuart Geman, Elie Bienenstock, and René Doursat. Neural networks and the bias/variance dilemma. *Neural Computation*, 4:1–58, 1992.

Ran Gilad-Bachrach, Amir Navot, and Naftali Tishby. Margin based feature selection - theory and algorithms. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2004.

T.R. Golub, D.K. Slonim, P. Tamazyo, C. Huard, M. Gaasenbeek, J.P. Mesirov, H. Coller, M.L. Loh, J.R. Downing, M.A. Caligiuri, C.D. Bloomfield, and E.S. Lander. Molecular classification of cancer: Class discovery and class prediction by gene expression monitoring. *Science*, 286:531–537, 1999.

Isabelle Guyon, Jason Weston, Stephen Barnhill, and Vladimir Vapnik. Gene selection for cancer classification using support vector machines. *Machine Learning*, 46:389–422, 2002.

Isabelle Guyon, Steve Gunn, Asa Ben Hur, and Gideon Dror. Result analysis of the nips 2003 feature selection challenge. In *Proceedings of the Neural Information Processing Systems (NIPS)*, 2004.

Isabelle Guyon, Jiwen Li, Theodor Mader, Patrick Pletscher, Georg Schneider, and Markus Uhr. Teaching machine learning from examples. *Unpublished*, 2006.

Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning : Data Mining, Inference, and Prediction*. Springer-Verlag, Berlin, 2001.

T. Joachims. Optimizing search engines using clickthrough data. In *Proceedings of the ACM Conference on Knowledge Discovery and Data Mining*, 2002.

T. Joachims and R Klinkenberg. Detecting concept drift with support vector machines. In *Proceedings of the Seventeenth International Conference on Machine Learning*, 2000.

Christian Jutten and Jeanny Hérault. Blind separation of sources, part 1: An adaptative algorithm bases on neuromimetic architecture. *Signal Processing*, 24:1–10, 1991.

Harold J. Kushner and G. George Yin. *Stochastic approximation and recursive algorithms and applications*, volume 35 of *Applications of Mathematics*. Springer-Verlag, New York, second edition, 2003. Stochastic Modelling and Applied Probability.

Thomas Navin Lal, Olivier Chapelle, and Bernhard Schölkopf. Combining a filter method with svms. In *Feature extraction, foundations and Applications*, pages 139–167, Berlin, 2006. Springer.

Sang-Kyun Lee, Seung-Joon Yi, and Byoung-Tak Zhang. Combining information-based supervised and unsupervised feature selection. In *Feature extraction, foundations and Applications*, Berlin, 2006. Springer.

Yan Liu and John R. Render. Video retrieval under sparse training data. In *Proceedings of the Second International Conference on Image and Video Retrieval*, pages 406–413, 2003.

D.J.C. MacKay. A practical bayesian framework for back propagation networks. *Neural Computation*, 3:448–472, 1992.

Patrice Simard and Yann LeCun. Memory based character recognition using a transformation invariant metric. In *Proceedings of the 12th International Conference on Pattern Recognition*, pages 262–267, 1998.

Surendra Singhi and Huan Liu. Feature subset selection bias for classification learning. In *Proceedings of the 23rd International Conference on Machine Learning*, pages 849 – 856, 2006.

Yijun Sun and Jian Li. Iterative relief for feature weighting. In *International Conference on Machine Learning*, 2006.

UCI. Uci machine learning repository: http://www.ics.uci.edu/ mlearn/mlrepository.html.

Vladimir N. Vapnik. *The nature of statistical learning theory*. Statistics for Engineering and Information Science. Springer-Verlag, New York, second edition, 2000.

Vladimir N. Vapnik. *Statistical learning theory*. Adaptive and Learning Systems for Signal Processing, Communications, and Control. John Wiley & Sons Inc., New York, 1998.

Jason Weston, Sayan Mukherjee, Olivier Chapelle, Massimiliano Pontil, Tomaso Poggio, and Vladimir Vapnik. Feature selection for SVMs. In *Neural Information Processing Systems*, pages 668–674, 2000.

Jason Weston, Andr Elisseeff, Bernhard Schlkopf, and Michael E. Tipping. Use of the zero-norm with linear models and kernel methods. *Journal of Machine Learning Research*, 3:1439–1461, 2003.

Zhili Wu and Chunhung Li. Feature selection for classification using transductive support vector machines. In *Proceedings of the Neural Information Processing Systems (NIPS)*, 2003.

Eric Xing, Michael I. Jordan, and Stuart Russel. Feature selection for high-dimensional genomic microarray data. In *Proceedings of the Eighteenth Internation Conference on Machine Learning*, pages 601–608, 2001.

Lei Yu and Huan Liu. Efficient feature selection via analysis of relevance and redundancy. *Journal of Machine Learning Research*, (5):1205–1224, 2004.