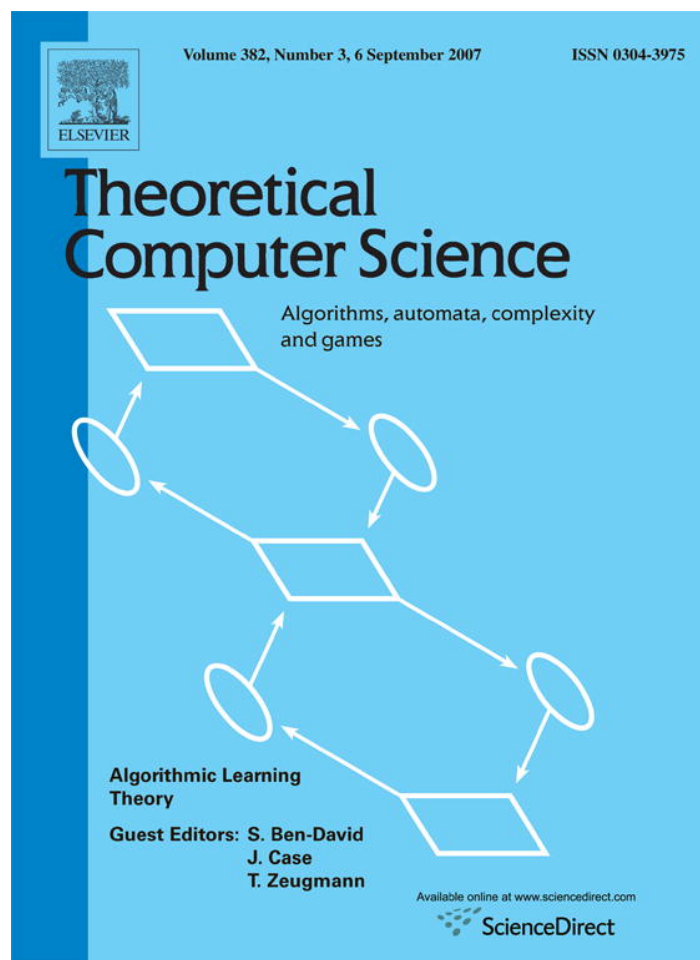


Provided for non-commercial research and education use.  
Not for reproduction, distribution or commercial use.



This article was published in an Elsevier journal. The attached copy is furnished to the author for non-commercial research and education use, including for instruction at the author's institution, sharing with colleagues and providing to institution administration.

Other uses, including reproduction and distribution, or selling or licensing copies, or posting to personal, institutional or third party websites are prohibited.

In most cases authors are permitted to post their version of the article (e.g. in Word or Tex form) to their personal website or institutional repository. Authors requiring further information regarding Elsevier's archiving and manuscript policies are encouraged to visit:

<http://www.elsevier.com/copyright>



# Applications of regularized least squares to pattern classification<sup>☆</sup>

Nicolò Cesa-Bianchi

*DSI, Università degli Studi di Milano, via Comelico 39, 20135 Milano, Italy*

---

## Abstract

We survey a number of recent results concerning the behaviour of algorithms for learning classifiers based on the solution of a regularized least-squares problem.

© 2007 Elsevier B.V. All rights reserved.

*Keywords:* On-line learning; Selective sampling; Ridge regression; Perceptron

---

## 1. Introduction

In pattern classification some unknown source is supposed to generate a sequence  $\mathbf{x}_1, \mathbf{x}_2, \dots$  of *instances* (data elements)  $\mathbf{x}_t \in \mathcal{X}$ . Each instance  $\mathbf{x}_t$  is associated with a *class label*  $y_t \in \mathcal{Y}$  (where  $\mathcal{Y}$  is a finite set) indicating a certain semantic property of the instance. For example, in a handwritten digit recognition task,  $\mathbf{x}_t$  is the digitalized image of a digit and its label  $y_t \in \{0, 1, \dots, 9\}$  is the corresponding numeral.

A learning algorithm for pattern classification operates by observing a sequence of *training examples*, that is pairs  $(\mathbf{x}_t, y_t)$  where  $\mathbf{x}_t$  is emitted by the source and  $y_t$  is the associated label (usually obtained via human supervision). The goal of the learner is to build a classifier  $f : \mathcal{X} \rightarrow \mathcal{Y}$  that predicts as accurately as possible the label of any further instance generated by the source. The performance of a learning algorithm is measured in terms of its ability to trade-off the accuracy of the generated classifier with the amount of training data used.

In the case  $\mathcal{X} = \mathbb{R}^d$ , which we study here, an important parametric family of classification functions are the linear classifiers  $f(\mathbf{x}) = \text{SGN}(\mathbf{w}^\top \mathbf{x})$ , where  $\mathbf{w} \in \mathbb{R}^d$  is the parameter vector and  $\text{SGN}(\cdot) \in \{-1, +1\}$  is the signum function. Although these classifiers can be only applied to binary classification problems, where the label set is  $\mathcal{Y} = \{-1, +1\}$ , there are several effective techniques to reduce a nonbinary classification task to a set of binary classification problems (see, e.g., [1]).<sup>1</sup> Hence in this survey we will restrict our attention to the simple binary classification task.

Another issue that we discuss is the fact that linear classifiers are usually too simple to perform accurate predictions on real-world data. To fix this problem we may use kernel functions. Kernels define embeddings of the instance space  $\mathbb{R}^d$  in complex high-dimensional feature spaces in such a way that functions that are nonlinear in terms of the

---

<sup>☆</sup> A preliminary version appeared in the Proceedings of the 15th International Conference on Algorithmic Learning Theory. LNAI 3244, Springer, 2004.

*E-mail address:* [cesa-bianchi@dsi.unimi.it](mailto:cesa-bianchi@dsi.unimi.it).

<sup>1</sup> Other more sophisticated approaches use specific loss measures for multiclass problems [2–4]. The application of these specific losses to the algorithms considered here is still an open problem.

instances  $\mathbf{x} \in \mathbb{R}^d$  become linear when expressed as functions of the embedded instances. If the predictions of a linear learning algorithm do not change when training instances are rotated in  $\mathbb{R}^d$  (a property that holds for all the algorithms studied here), then linear classifiers can be learned in feature space with a reasonable computational overhead. See the monographs [5,6] for a thorough analysis of kernel-based learning.

In this survey we focus on a specific family of algorithms for learning linear classifiers based on the solution of a regularized least squares (RLS) problem. We advocate the use of these algorithms in learning for several reasons.

- *Incremental learning.* Like the Perceptron, RLS algorithms can be run on large amounts of data since they are trained incrementally.
- *Empirical performance.* RLS algorithms are generally as accurate as the best incremental linear learners, and often comparable to the best batch classifiers.
- *Learning with kernels.* The computations performed by RLS algorithms can be expressed using just inner products, thus allowing efficient implementation of kernel-based learning.
- *Versatility.* Applications of RLS have been derived for tasks such as data filtering [7], active learning [8], and hierarchical classification [9].
- *Performance guarantees.* RLS algorithms are analytically very tractable. For the basic binary classification task we can prove theoretical performance bounds in both the adversarial and statistical data models; more specialized bounds have been proven for filtering, active learning, and hierarchical classification.

The paper is organized as follows. In Section 2 we introduce the framework of regularized least squares and describe some of the basic algorithms. In Section 3 we state and discuss performance bounds for three types of data sources. In Section 4 we study the properties of RLS algorithms in an active learning scenario, where the goal is to attain a certain learning performance using as few training labels as possible. Section 5 is devoted to the conclusions.

## 2. Regularized least squares for classification

Regularized least squares methods have been introduced to solve linear regression problems where the labels  $y_t$  are real numbers. The most basic algorithm of this family is the ridge regression (RR) procedure of Hoerl and Kennard [13]. Given a training set  $(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_n, y_n)$ , where  $(\mathbf{x}_t, y_t) \in \mathbb{R}^d \times \mathbb{R}$ , the RR algorithm outputs a linear function  $f(\mathbf{x}) = \mathbf{w}^\top \mathbf{x}$  where  $\mathbf{w}$  is the solution of

$$\min_{\mathbf{v} \in \mathbb{R}^d} \left( \sum_{t=1}^n (\mathbf{v}^\top \mathbf{x}_t - y_t)^2 + a \|\mathbf{v}\|^2 \right) \tag{1}$$

and  $a > 0$  is the regularization parameter. A closed form expression for  $\mathbf{w}$  is  $(aI + S S^\top)^{-1} S \mathbf{y}$ , where  $I$  is the  $n \times n$  identity matrix,  $S$  is the matrix whose columns are the instances  $\mathbf{x}_1, \dots, \mathbf{x}_n$  and  $\mathbf{y} = (y_1, \dots, y_n)$  is the vector of instance labels. The presence of the regularization term  $a \|\mathbf{v}\|^2$  guarantees unicity of the solution in the case  $\mathbf{x}_1, \dots, \mathbf{x}_n$  do not span  $\mathbb{R}^d$ .

We may apply RR directly to a binary classification problem. In this case, since labels  $y_t$  belong to  $\{-1, +1\}$ , problem (1) can be equivalently rewritten as

$$\min_{\mathbf{v} \in \mathbb{R}^d} \left( \sum_{t=1}^n (1 - y_t \mathbf{v}^\top \mathbf{x}_t)^2 + a \|\mathbf{v}\|^2 \right). \tag{2}$$

The minimization problem (2) is quite similar to the 2-norm support vector machine (see, e.g, [14]), whose classifier is based on the solution of

$$\min_{\mathbf{v} \in \mathbb{R}^d} \left( \sum_{t=1}^n ([1 - y_t \mathbf{v}^\top \mathbf{x}_t]_+)^2 + a \|\mathbf{v}\|^2 \right). \tag{3}$$

Here we use  $[x]_+$  to denote  $\max\{0, x\}$ . An important difference between the two expressions is that the solution of (2) always depends on all training examples  $(\mathbf{x}_1, \dots, \mathbf{x}_n)$ , whereas the solution of (3) is sparse, as it only depends on a (often small) subset of the training instances (the support vectors).

Sparsity is important in controlling the time required to compute predictions when using kernel functions (see the discussion at the end of this section). Moreover, sparsity can be an indicator of good predictive power. For instance, it can be directly used to control the generalization error in statistical learning [15].

In the binary classification case, sparsity can be obtained in the RR solution through a simple technique which we now describe. The RR algorithm generating sparse solutions is called *second-order Perceptron algorithm* [16]. This algorithm works incrementally, starting with the constant linear classifier  $\mathbf{w}_0 = (0, \dots, 0)$ . At each time step  $t = 1, 2, \dots$  a new instance  $\mathbf{x}_t$  is obtained from the source and the algorithm outputs the prediction  $\hat{y}_t = \text{SGN}(\mathbf{w}_{t-1}^\top \mathbf{x}_t)$  for the label of  $\mathbf{x}_t$ , where  $\mathbf{w}_{t-1}$  is the parameter of the current linear classifier. Then, the label  $y_t$  is obtained. If  $\hat{y}_t \neq y_t$ , the example  $(\mathbf{x}_t, y_t)$  is stored by updating  $\mathbf{w}_{t-1}$  to  $\mathbf{w}_t = (aI + S_t S_t^\top)^{-1} S_t y_t$ . Here  $S_t$  is defined as the matrix whose columns are the stored instances and  $\mathbf{y}_t$  is the vector of labels of the stored instances. If  $\hat{y}_t = y_t$  then  $\mathbf{w}_t = \mathbf{w}_{t-1}$  and no update occurs.

Expressing  $\mathbf{w}_t$  in a form similar to (2) we obtain

$$\mathbf{w}_t = \underset{\mathbf{v} \in \mathbb{R}^d}{\text{argmin}} \left( \sum_{s \in \mathcal{M}_t} (1 - y_s \mathbf{v}^\top \mathbf{x}_s)^2 + a \|\mathbf{v}\|^2 \right)$$

where  $\mathcal{M}_t = \{1 \leq s \leq t : \text{SGN}(\mathbf{w}_{s-1}^\top \mathbf{x}_s) \neq y_s\}$  is the set of indices of the examples  $(\mathbf{x}_s, y_s)$  stored in the first  $t$  steps.

Note that by keeping the representation of  $\mathbf{w}_{t-1}$  split in the inverse matrix  $(aI + S_{t-1} S_{t-1}^\top)^{-1}$  and the vector  $S_{t-1} \mathbf{y}_{t-1}$ , we can compute the weight  $\mathbf{w}_t$  in constant time  $\Theta(d^2)$  using standard linear algebra techniques. If kernel functions are used, we can compute  $\mathbf{w}_t$  from  $\mathbf{w}_{t-1}$  in time order of  $(dm)^2$ , where  $m$  is the number of examples stored so far. The same time is also needed to compute each prediction  $\hat{y}_t = \text{SGN}(\mathbf{w}_{t-1}^\top \mathbf{x}_t)$ . By contrast, the kernel (first-order) Perceptron takes constant time to update the weight vector and time of the order of  $dm$  to compute a prediction. See [16] for details on these computations.

### 3. Analysis

We now state bounds on the predictive error of regularized least squares for classification under different assumptions on the source generating the data. In particular, we consider three families of sources.

- *Arbitrary sources*: no assumption is made on the mechanism generating the instances  $\mathbf{x}_t$  and their associated labels  $y_t$ . Performance bounds proven for this source therefore hold on any individual sequence of examples.
- *Independent and identically distributed sources*: here the examples  $(\mathbf{x}_t, y_t)$  are drawn independently from a fixed and unknown distribution on  $\mathbb{R}^d \times \{-1, +1\}$ .
- *Probabilistic linear sources*: instances are generated arbitrarily (as in the first source) while the following linear noise model is assumed for the labels. There exists a fixed and unknown vector  $\mathbf{v} \in \mathbb{R}^d$  with  $\|\mathbf{v}\| = 1$  such that  $\mathbb{P}(y_t = 1 | \mathbf{x}_t) = (1 + \mathbf{v}^\top \mathbf{x}_t)/2$  for all  $\mathbf{x}_t$ . To guarantee that  $\mathbf{v}^\top \mathbf{x}_t \in [-1, 1]$  we also assume  $\|\mathbf{x}_t\| = 1$  for all  $t$  (in other words, each instance is normalized before being fed into the learning algorithm).

The analysis of learning algorithms on arbitrary data sequences has been introduced in learning theory by Angluin [17] and Littlestone [18]. However, the Perceptron convergence theorem [19,20] can be viewed as an early example of this approach. The i.i.d. sources are the standard data model used in statistical learning theory [21,22]. Here we illustrate a simple technique showing how results for arbitrary sources can be specialized to obtain good bounds in the case of i.i.d. sources. Probabilistic linear sources have been considered in [9] to analyze the properties of regularized least squares as a statistical estimator in binary classifications tasks.

To measure the performance of our learning algorithms we count prediction mistakes. We say that an example  $(\mathbf{x}_t, y_t) \in \mathbb{R}^d \times \{-1, +1\}$  is mistaken by a linear classifier  $\mathbf{w}_{t-1}$  when  $\hat{y}_t \neq y_t$ , where  $\hat{y}_t = \text{SGN}(\mathbf{w}_{t-1}^\top \mathbf{x}_t)$ . The mistake indicator function  $\mathbb{I}_{\{\hat{y}_t \neq y_t\}}$  is also called *zero-one loss*.

Our results are all derived from analysis of on-line learning processes. They cannot be directly used to relate the training error of a classifier to its generalization error, as in the traditional statistical learning approach. On the other hand, our results for i.i.d. sources provide bounds on the generalization error of the linear classifiers generated by the run of the on-line algorithm. Rather than being expressed in terms of the training error, these results are expressed in terms of the number of mistakes made by the on-line algorithm on the training set.

### 3.1. Arbitrary sources

We start by studying the behaviour of the second-order Perceptron run over an arbitrary sequence of examples. In particular, we bound the number of time steps  $t$  in which the next example  $(\mathbf{x}_t, y_t)$  is mistaken by the current classifier  $\mathbf{w}_{t-1}$ . The bound is expressed in terms of the structure of the data sequence and in terms of the performance of an arbitrary fixed linear classifier  $\mathbf{u} \in \mathbb{R}^d$  (which we call the *reference classifier*) on the same sequence.

We measure the performance of a reference classifier  $\mathbf{u}$  using the *hinge loss*  $[1 - y \mathbf{u}^\top \mathbf{x}]_+$ , which is the loss function occurring in the expression (3) related to the support vector machine algorithm. Note that  $[1 - yp]_+ \geq \mathbb{I}_{\{\hat{y} \neq y\}}$  for  $\hat{y} = \text{SGN}(p)$ . Hence the hinge loss upper bounds the zero–one loss.

The definition and analysis of the second-order Perceptron are based on three different losses (the squared loss, the zero–one loss and the hinge loss). The square loss, which is the loss defining the predictions of RLS algorithms, allows us to express each linear classifier  $\mathbf{w}_t$  in a simple closed form. On the other hand the zero–one loss, which counts the prediction mistakes, is a more natural performance measure in binary classification tasks. However, devising a simple and efficient on-line linear classifier that does well against the best fixed linear classifier is a hard problem when the zero–one loss is used to score *both* algorithms. Indeed, it is well known that the problem of approximating the mistakes of the best linear classifier, even when the sequence of examples is known in advance, cannot be solved in polynomial time unless  $P = NP$  (see [10,11]). This justifies the use of the hinge loss in the competitive analysis of linear learning algorithms.

We also remark that other approaches succeeded in comparing Perceptron-like algorithms and reference classifiers using the *squared hinge loss* as the common performance measure [12]. However, these alternative analyses typically cause the appearance in the bound of a factor larger than 1 multiplying the loss of the reference classifier.

For any fixed  $\mathbf{u}$  and for any fixed sequence  $(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2) \dots \in \mathbb{R}^d \times \{-1, +1\}$  define the cumulative hinge loss

$$L_n(\mathbf{u}) = \sum_{t=1}^n [1 - y_t \mathbf{u}^\top \mathbf{x}_t]_+.$$

We are now ready to state the main result for the second-order Perceptron. For a proof of this bound we refer to [16].

**Theorem 1.** *If the second-order Perceptron is run with regularization parameter  $a > 0$  on a sequence  $(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2) \dots \in \mathbb{R}^d \times \{-1, +1\}$ , then, for all  $n \geq 1$  and for all  $\mathbf{u} \in \mathbb{R}^d$ ,*

$$\sum_{t=1}^n \mathbb{I}_{\{\hat{y}_t \neq y_t\}} \leq L_n(\mathbf{u}) + \sqrt{(a \|\mathbf{u}\|^2 + \mathbf{u}^\top A_n \mathbf{u}) \sum_{i=1}^d \ln(1 + \lambda_i/a)}$$

where  $\lambda_1, \dots, \lambda_d$  are the eigenvalues of the matrix

$$A_n = \sum_{t=1}^n \mathbf{x}_t \mathbf{x}_t^\top \mathbb{I}_{\{\hat{y}_t \neq y_t\}}.$$

For  $a \rightarrow \infty$ , the bound of Theorem 1 reduces to the bound for the classical Perceptron algorithm (see, e.g., [23]). To see this, note that

$$\begin{aligned} \lim_{a \rightarrow \infty} (a \|\mathbf{u}\|^2 + \mathbf{u}^\top A_n \mathbf{u}) \sum_{i=1}^d \ln(1 + \lambda_i/a) &= \lim_{a \rightarrow \infty} (a \|\mathbf{u}\|^2 + \mathbf{u}^\top A_n \mathbf{u}) \sum_{i=1}^d \lambda_i/a \\ &= \|\mathbf{u}\|^2 \text{tr}(A_n A_n^\top) \end{aligned}$$

where  $\text{tr}(\cdot)$  is the trace (sum of the eigenvalues) of a matrix.

Let  $m = \sum_{t=1}^n \mathbb{I}_{\{\hat{y}_t \neq y_t\}}$  be the total number of mistakes made by the second-order Perceptron. We now use a basic algebraic fact stating that  $A_n$  has the same nonzero eigenvalues as the  $m \times m$  matrix  $G$  with entries  $G_{i,j} = \mathbf{x}_i^\top \mathbf{x}_j$  ( $G$  is called the *Gram matrix* of the instances  $\mathbf{x}_1, \dots, \mathbf{x}_m$ ). This implies that

$$\text{tr}(A_n A_n^\top) = \text{tr}(A_n^\top A_n) = \mathbf{x}_1^\top \mathbf{x}_1 + \dots + \mathbf{x}_m^\top \mathbf{x}_m \leq m \max_t \|\mathbf{x}_t\|^2 \tag{4}$$

where  $\max_t$  ranges over the mistaken examples  $(\mathbf{x}_t, y_t)$ . Hence, for  $a \rightarrow \infty$ , the bound of Theorem 1 can be written as

$$m \leq L_n(\mathbf{u}) + \|\mathbf{u}\| \left( \max_t \|\mathbf{x}_t\| \right) \sqrt{m}.$$

Solving for  $m$  gives the Perceptron mistake bound.

The bound of Theorem 1 is not in closed form as the matrix  $A_n$  is defined in terms of the mistakes  $\mathbb{I}_{\{\hat{y}_t \neq y_t\}}$ . In order to help the reader compare this result with other mistake bounds, we now derive a (somewhat weaker) closed form version of the second-order mistake bound. To keep things simple, we assume that the sequence  $(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2) \dots \in \mathbb{R}^d \times \{-1, +1\}$  is linearly separable with margin  $\gamma > 0$ . That is, there exist  $\mathbf{u} \in \mathbb{R}^d$  such that  $\|\mathbf{u}\| = 1$  and  $y_t \mathbf{u}^\top \mathbf{x}_t \geq \gamma$  for all  $t \geq 1$ . Moreover, we assume that instances  $\mathbf{x}_t$  are normalized,  $\|\mathbf{x}_t\| = 1$  for all  $t \geq 1$ . Under these assumptions, and setting the parameter  $a$  of the algorithm to 1, the bound of Theorem 1 becomes

$$m \leq \frac{1}{\gamma} \sqrt{(1 + \lambda(\mathbf{u})) \sum_{i=1}^d \ln(1 + \lambda_i)}$$

where  $m$  is the number of mistakes and we write

$$\lambda(\mathbf{u}) = \mathbf{u}^\top A_n \mathbf{u} = \sum_t (\mathbf{u}^\top \mathbf{x}_t)^2 \mathbb{I}_{\{\hat{y}_t \neq y_t\}}.$$

Since  $\mathbf{u}$  is a unit-norm linear separator with margin  $\gamma$ , and since  $\|\mathbf{x}_t\| = 1$ , we know that  $\gamma^2 m \leq \lambda(\mathbf{u}) \leq m$ . Thus we may write  $\lambda(\mathbf{u}) = \lambda_0 m$  for some  $\gamma^2 \leq \lambda_0 \leq 1$ . Moreover, from (4) we know that  $\lambda_1 + \dots + \lambda_d = m$ .

The quantity  $(1 + \lambda_1) \times \dots \times (1 + \lambda_d)$ , under the constraint  $\lambda_1 + \dots + \lambda_d = m$ , is maximized when  $\lambda_i = m/d$  for all  $i$ . Thus the bound of Theorem 1 can be weakened to

$$m \leq \frac{1}{\gamma} \sqrt{(1 + \lambda_0 m) d \ln(1 + m/d)}. \tag{5}$$

To get a reasonable bound, we now need to assume that  $\lambda_0 \leq c\gamma$  for some  $c > 0$ . This amounts to saying that the second-order Perceptron tends to make mistakes on instances  $\mathbf{x}_t$  such that  $|\mathbf{u}^\top \mathbf{x}_t| \leq \sqrt{\gamma}$ . Although this condition is not particularly nice, as it is defined in terms of the behaviour of the algorithm on the sequence, it is certainly plausible that many mistakes are made on instances on which the separating hyperplane  $\mathbf{u}$  achieves a small margin. If  $\lambda_0 \leq c\gamma$  holds, then one can easily show that

$$m = \Theta\left(\frac{d}{\gamma} \ln \frac{1}{\gamma}\right)$$

satisfies (5). This bound is related to the essentially optimal mistake bound  $d \ln(1/\gamma)$  achieved by the Bayes Point Machine [24]. However, whereas for the second-order Perceptron updating the weight vector takes only time  $\Theta(d^2)$  (see Section 2), the Bayes Point Machine must solve a computationally intractable (#P-hard) problem at each step.

In [16] we show that whenever the reference classifier  $\mathbf{u}$  is correlated with some eigenvector of the matrix  $A_n$  associated with an eigenvalue significantly smaller than the largest eigenvalue (see Fig. 1), then the bound of Theorem 1 reaches its minimum at  $a < \infty$ . This means that, in such situations, the bound for the second-order Perceptron with parameter  $a$  properly tuned is better than the corresponding bound for the classical Perceptron algorithm. The cases where the second-order bound loses its advantage are those when the matrix  $A_n$  has a nearly uniform spectrum (the eigenvalues are close to each other) and  $d \gg m$  (which typically happens when the algorithm is run with kernels). In practice, however, if the instances  $\mathbf{x}_t$  are normalized so that  $\|\mathbf{x}_t\| = 1$ , then the second-order Perceptron, run with  $a = 1$  and without using kernels, performs typically better than the classical Perceptron even on high-dimensional real-world datasets such as Reuters Corpus Volume 1.

### 3.2. Independent and identically distributed sources

We now move on to analyzing the performance of the second-order Perceptron in the classical statistical learning model, where data sequences  $(\mathbf{x}_t, y_t) \in \mathbb{R}^d \times \{-1, +1\}$  for  $t = 1, 2, \dots$  are generated via independent draws from a fixed and unknown distribution. To stress that instances and labels are now random variables we write  $\mathbf{X}_t$  and  $Y_t$ .

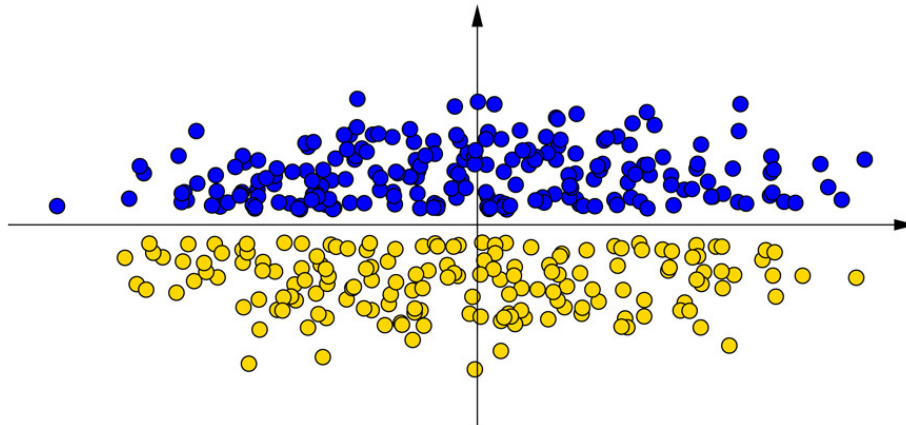


Fig. 1. A linearly separable set of instances  $x_t$  on which the second-order Perceptron has an advantage over the classical Perceptron algorithm. The data correlation matrix has two eigenvectors aligned with the axis shown in the picture. The vector perpendicular to the separating hyperplane turns out to be well aligned with the eigenvector associated with the smallest eigenvalue.

In this framework, the standard measure of performance for a classifier  $f : \mathbb{R}^d \rightarrow \{-1, +1\}$  is the *risk*  $R(f) = \mathbb{P}(f(\mathbf{X}) \neq Y)$ . This is the probability that  $f$  misclassifies the next example  $(\mathbf{X}_t, Y_t)$  generated by the source.

Introduce the notation  $R(\mathbf{w}) = \mathbb{P}(\text{SGN}(\mathbf{w}^\top \mathbf{X}) \neq Y)$  to indicate the risk of the linear classifier using weight  $\mathbf{w}$ . Consider the second-order Perceptron run on a data sequence of length  $n$  and let  $\mathbf{w}_0, \mathbf{w}_1, \dots, \mathbf{w}_{n-1}$  be the weights that have been used by the algorithm to compute the sequence  $\hat{y}_1, \dots, \hat{y}_n$  of predictions. Define

$$M_n = \sum_{t=1}^n \mathbb{I}_{\{\hat{y}_t \neq Y_t\}}$$

where  $\hat{y}_t = \text{SGN}(\mathbf{w}_{t-1}^\top \mathbf{X}_t)$ . We now show that the average risk

$$\frac{1}{n} \sum_{t=1}^n R(\mathbf{w}_{t-1})$$

is close to  $M_n/n$  with high probability over the random draw of

$$(\mathbf{X}_1, Y_1), \dots, (\mathbf{X}_n, Y_n).$$

In other words, the average risk of the linear classifiers generated by the second-order Perceptron during its run on a data sequence is close to the fraction of mistakes made.

To see this note that, since each example  $(\mathbf{X}_t, Y_t)$  is drawn i.i.d.,

$$\mathbb{E}[\mathbb{I}_{\{\hat{y}_t \neq Y_t\}} \mid (\mathbf{X}_1, Y_1), \dots, (\mathbf{X}_{t-1}, Y_{t-1})] = R(\mathbf{w}_{t-1}).$$

Hence the stochastic process  $R(\mathbf{w}_{t-1}) - \mathbb{I}_{\{\hat{y}_t \neq Y_t\}}$  is a bounded martingale difference sequence. We may now use standard large deviation results (such as the Hoeffding–Azuma inequality [25,26]) to show that, for any fixed  $\delta \in (0, 1)$

$$\frac{1}{n} \sum_{t=1}^n R(\mathbf{w}_{t-1}) \leq \frac{M_n}{n} + \sqrt{\frac{2}{n} \ln \frac{1}{\delta}}$$

holds with probability at least  $1 - \delta$ .

We are now left with the problem of exhibiting a specific linear classifier whose risk is close to the average. In [27] a technique is described to select a weight  $\mathbf{w}^*$  among the ensemble  $\mathbf{w}_0, \mathbf{w}_1, \dots, \mathbf{w}_{n-1}$  such that the linear classifier using  $\mathbf{w}^*$  achieves a risk just slightly higher than the average risk of the ensemble. In particular, it is shown that

$$R(\mathbf{w}^*) \leq \frac{M_n}{n} + 6\sqrt{\frac{2}{n} \ln \frac{2(n+1)}{\delta}} \tag{6}$$

holds with probability at least  $1 - \delta$ .

We have thus probabilistically bounded the risk of a linear classifier generated by the second-order Perceptron algorithm in terms of the number  $M_n$  of mistakes made by the algorithm run on a random training sequence. Note that (6) has been derived without exploiting any specific property of the learning algorithm. Hence the bound actually holds for *any* incremental learning algorithm making  $M_n$  mistakes on the data sequence.

When considering a specific incremental learning algorithm, we can specialize (6) by replacing  $M_n$  with a bound proven in the arbitrary source model. In particular, for the second-order Perceptron we may bound  $M_n$  using Theorem 1 and thus obtain a data-dependent bound on the risk which depends on the spectrum of the observed data sequence. Note that, in the case of i.i.d. sources, the hinge loss term  $L_n(\mathbf{u})$  in Theorem 1 appears as an expected value  $\mathbb{E} L_n(\mathbf{u})/n$ . Since the source is i.i.d., we have that

$$\frac{1}{n} \mathbb{E} L_n(\mathbf{u}) = \mathbb{E}[1 - Y \mathbf{u}^\top \mathbf{X}]_+.$$

Hence, this analysis enables us to compare the risk of  $\mathbf{w}^*$  with the *expected hinge loss* of an arbitrary linear classifier.

### 3.3. Probabilistic linear sources

In this section we temporarily abandon the analysis of the second-order Perceptron and go back to the RR procedure (1). Recall that, unlike the second-order Perceptron, the classifiers generated by RR depend on *all* previously observed examples.

In regression tasks, if the real labels are expressed by a linear function of the instances plus a Gaussian noise term, then RR provides a (biased) estimator of this hidden linear function (see, e.g., [28]). We now show that, in an appropriate classification noise model, RR is a biased estimator of a hidden linear classifier.

Given an arbitrary sequence  $\mathbf{x}_1, \mathbf{x}_2, \dots$  of instances  $\mathbf{x}_t \in \mathbb{R}^d$  with  $\|\mathbf{x}_t\| = 1$ , we assume that the labels  $Y_1, Y_2, \dots$  are random variables with conditional distribution  $\mathbb{P}(Y_t = 1 \mid \mathbf{x}_t) = (1 + \mathbf{v}^\top \mathbf{x}_t)/2$  for some fixed and hidden  $\mathbf{v} \in \mathbb{R}^d$  with  $\|\mathbf{v}\| = 1$ . Note that the linear classifier  $\text{SGN}(\mathbf{v}^\top \mathbf{x}_t)$  is Bayes optimal for this data model.

We consider a slight variant of the RR algorithm (2). In this variant, which has been introduced in [29] for solving regression problems, the weight used to predict the label of the next instance  $\mathbf{x}_t$  is defined as

$$(aI + S_t S_t^\top)^{-1} S_{t-1} \mathbf{y}_{t-1} \tag{7}$$

where, for each  $t = 1, 2, \dots$ ,  $S_t$  is the matrix  $[\mathbf{x}_1, \dots, \mathbf{x}_t]$ . Note that the only difference with the weight  $(aI + S_{t-1} S_{t-1}^\top)^{-1} S_{t-1} \mathbf{y}_{t-1}$  of the RR algorithm is that we have added the current instance  $\mathbf{x}_t$  in the inverse matrix. This amounts to a slightly stronger regularization, and has the side effect of making the final classifier nonlinear. For this reason, we denote with  $\widehat{\mathbf{w}}_t$  (with index  $t$  rather than  $t - 1$ ) the weight (7) used at time  $t$  to compute the prediction  $\widehat{y}_t = \text{SGN}(\widehat{\mathbf{w}}_t^\top \mathbf{x}_t)$  for the random label  $Y_t$ .

In the case of probabilistic linear sources we evaluate the performance of the modified RR algorithm by directly comparing the number of mistakes made by the algorithm with the number of mistakes made by the Bayes optimal classifier based on the hidden vector  $\mathbf{v}$ . That is, we look at the expected value of the difference

$$\sum_{t=1}^n \mathbb{I}_{\{\widehat{y}_t \neq Y_t\}} - \sum_{t=1}^n \mathbb{I}_{\{\text{SGN}(\mathbf{v}^\top \mathbf{x}_t) \neq Y_t\}}.$$

In [9] the following result is proved.

**Theorem 2.** *Suppose that the modified RR algorithm (7) is run with regularization parameter  $a = 1$  on a sequence  $(\mathbf{x}_1, Y_1), (\mathbf{x}_2, Y_2) \dots \in \mathbb{R}^d \times \{-1, +1\}$  such that  $\|\mathbf{x}_t\| = 1$  and  $\mathbb{P}(Y_t = 1 \mid \mathbf{x}_t) = (1 + \mathbf{v}^\top \mathbf{x}_t)/2$  for all  $t$  and for some  $\mathbf{v} \in \mathbb{R}^d$  with  $\|\mathbf{v}\| = 1$ . Then, for all  $n \geq 1$ ,*

$$\sum_{t=1}^n \mathbb{P}(\widehat{y}_t \neq Y_t) - \sum_{t=1}^n \mathbb{P}(\text{SGN}(\mathbf{v}^\top \mathbf{x}_t) \neq Y_t) \leq \frac{16(1 + 1/e)}{\gamma^2} \sum_{i=1}^d \log(1 + \lambda_i)$$

where  $\gamma = \min_{t=1, \dots, n} |\mathbf{v}^\top \mathbf{x}_t|$  and  $\lambda_1, \dots, \lambda_d$  are the eigenvalues of the matrix  $\Lambda_n = \mathbf{x}_1 \mathbf{x}_1^\top + \dots + \mathbf{x}_n \mathbf{x}_n^\top$ .

Note that the bound of [Theorem 2](#) implies that the expected number of mistakes made by the modified RR algorithm in excess with respect to those made by the Bayes optimal classifier is bounded by a logarithmic function of  $n$ .

To prove this logarithmic bound, we use (4) to show that  $\lambda_1 + \dots + \lambda_d = \mathbf{x}_1^\top \mathbf{x}_1 + \dots + \mathbf{x}_n^\top \mathbf{x}_n \leq n$ . Similarly to the reasoning used to derive (5), we observe that  $(1 + \lambda_1) \times \dots \times (1 + \lambda_d)$  is maximized when  $\lambda_i = n/d$  for all  $i = 1, \dots, d$ . Thus we conclude

$$\sum_{i=1}^d \ln(1 + \lambda_i) \leq d \ln\left(1 + \frac{n}{d}\right).$$

Hence, in this probabilistic linear model, regularized least squares ensures fast convergence to the mistake rate of the Bayes optimal classifier.

#### 4. Selective sampling

[Theorem 2](#) states that modified ridge regression has a good performance bound when data are generated by probabilistic linear sources. However, to achieve this bound all examples have to be stored. This becomes a problem if we use kernels, as the space required to represent a classifier built on  $t$  observed instances is  $\Theta(t^2)$ . To control the space requirements without a significant reduction in predictive performance, we now introduce a *selective sampling* version of this algorithm.

At each time step  $t$ , selective sampling uses weights having the same form as the weights (7) used by the modified RR algorithm. However, selective sampling stores the current example  $(\mathbf{x}_t, y_t)$  only if  $|\widehat{\mathbf{w}}_t^\top \mathbf{x}_t| \leq c\sqrt{(\ln t)/N_t}$ , where  $N_t$  is the number of examples stored up to time  $t$  and  $c$  is some constant (the choice  $c = 5$  works well in practice when instances are normalized). Note the following important fact: to decide whether to store an example  $(\mathbf{x}_t, Y_t)$  selective sampling does not need to observe the label  $Y_t$ . Thus, without loss of generality, we may assume that the labels of non-stored examples remain forever unknown (for this reason, selective sampling algorithms are also called *label efficient*).

We performed some experiments using the 50 most frequent categories in the first 40,000 news stories (in chronological order) of the Reuters Corpus Volume 1 [30] dataset. During the preprocessing phase we first encoded the documents using a standard TF-IDF bag-of-words representation, and then we normalized all the encoded vectors with respect to the Euclidean norm. We compared the average  $F$ -measure of Perceptron, second-order Perceptron and selective sampling on 50 binary classification tasks (one task for each one of the 50 categories). In all experiments the parameter  $a$ , used by the RLS algorithms, was set to 1 and we did not use kernels.

These experiments (see [Fig. 2](#)) show that selective sampling works very effectively, achieving the same performance as the second-order Perceptron while storing a rapidly decreasing fraction of the observed instances (we did not compare the performance of selective sampling with that of the modified RR algorithm as the latter cannot be run in a reasonable amount of time on the moderately large datasets used in our experiments).

The use of a threshold  $\tau_t = \sqrt{(\ln t)/N_t}$  in the selective sampling procedure is motivated by the following observation: when enough examples have been stored then, with high probability,  $\widehat{\mathbf{w}}_t$  and the Bayes optimal  $\mathbf{v}$  classify in the same way any new instance that has a large enough “margin”. That is, any instance  $\mathbf{x}_t$  such that  $|\widehat{\mathbf{w}}_t^\top \mathbf{x}_t| > \tau_t$ . The form of the dependence of  $\tau_t$  on  $t$  and  $N_t$  is derived from simple large deviation analysis. Although the theoretical investigation of the selective sampling procedure is still in progress, some preliminary results have been published in [8].

We close this section by describing a selective sampling technique guaranteeing good predictive performance even when data are generated by arbitrary sources. More precisely, we consider the second-order Perceptron using the following randomized selective sampling rule: with probability  $C/(C + |\widehat{\mathbf{w}}_t^\top \mathbf{x}_t|)$ , where  $C > 0$  is a parameter, get the true label  $y_t$ . Then store  $(\mathbf{x}_t, y_t)$  if  $\widehat{y}_t \neq y_t$ . Thus, an example whose label is correctly predicted is never stored, and an example  $(\mathbf{x}_t, y_t)$  on which a mistake is made is stored with a probability inversely proportional to the margin  $|\widehat{\mathbf{w}}_t^\top \mathbf{x}_t|$ .

In [31] the following is proven.

**Theorem 3.** *If the selective sampling second-order Perceptron is run on a sequence  $(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2) \dots \in \mathbb{R}^d \times \{-1, +1\}$ , then, for all  $n \geq 1$  and for all  $\mathbf{u} \in \mathbb{R}^d$ ,*

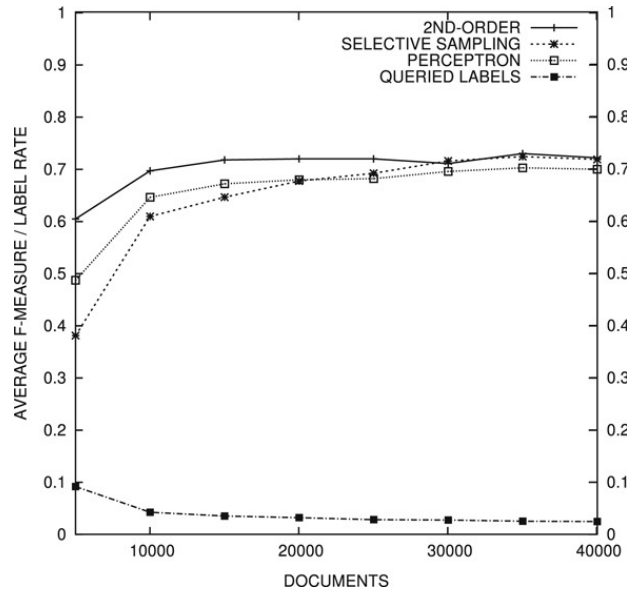


Fig. 2. Behaviour of the selective sampling variant of the modified RR algorithm on the first 40,000 documents in the Reuters Corpus Volume 1 dataset. The increasing curves are the  $F$ -measures for the second-order Perceptron and the Perceptron (both observing all labels), and for selective sampling. The decreasing curve is the rate of labels used by the selective sampling procedure. All quantities are averaged over 50 binary classification tasks. Note that the label rate of selective sampling decreases very fast and the performance eventually reaches the performance of the second-order Perceptron, which is allowed to observe all labels.

$$\mathbb{E} \left[ \sum_{t=1}^n \mathbb{I}_{\{\hat{y}_t \neq y_t\}} \right] \leq L_n(\mathbf{u}) + \frac{C}{2} \left( a \|\mathbf{u}\|^2 + \sum_{t=1}^n (\mathbf{u}^\top \mathbf{x}_t)^2 \mathbb{E} K_t \right) + \frac{1}{2C} \sum_{i=1}^d \mathbb{E} \ln \left( 1 + \frac{\Lambda_i}{a} \right)$$

where  $K_t$  is the indicator of the event “ $(\mathbf{x}_t, y_t)$  is stored”, and  $\Lambda_1, \dots, \Lambda_d$  are the eigenvalues of the random matrix  $(\mathbf{x}_1 \mathbf{x}_1^\top) K_1 + \dots + (\mathbf{x}_n \mathbf{x}_n^\top) K_n$ .

It is important to remark that Theorem 3 provides a bound on the expected number of mistakes made on the whole sequence of examples, irrespective of which of these examples have been stored. Note also that, similarly to Theorem 1, the bound proven in Theorem 3 is not in closed form as the random variables  $K_t$  depend on the algorithm’s mistakes.

This result also reveals an interesting phenomenon. A proper choice of the parameter  $C$  of the randomized rule yields, in expectation, the same bound as that stated in Theorem 1 for the second-order Perceptron without the selective sampling mechanism. Hence, in some sense, this technique uses the margin information to select those labels that can be ignored without increasing (in expectation) the overall number of mistakes. One may suspect that this gain is not real, as the tuning of  $C$  preserving the original mistake bound might force the algorithm to sample all but an insignificant number of labels. Fig. 3 brings experimental evidence that goes against this claim. By running the algorithms on real-world textual data, we show that no significant decrease in the predictive performance is suffered even when the parameter  $C$  is set to values that cause a significant fraction of the labels to be not observed. We refer the interested reader to [31] for more extensive experiments.

## 5. Conclusions

In this survey we have reported on an ongoing research project aimed at studying the theoretical properties of regularized least squares algorithms for classification tasks. Due to their analytic tractability, versatility and good empirical behaviour, RLS methods provide a nice case in support to the systematic use of principled methods for solving learning problems.

The results mentioned here leave many problems open. We lack a tight closed form expression for the second-order Perceptron bound. Our risk bound (6) for arbitrary incremental algorithms is not tight when  $M_n$  is very small. Finally, our analysis of selective sampling for probabilistic linear sources is only done for a simpler and less effective variant of the algorithm described here.

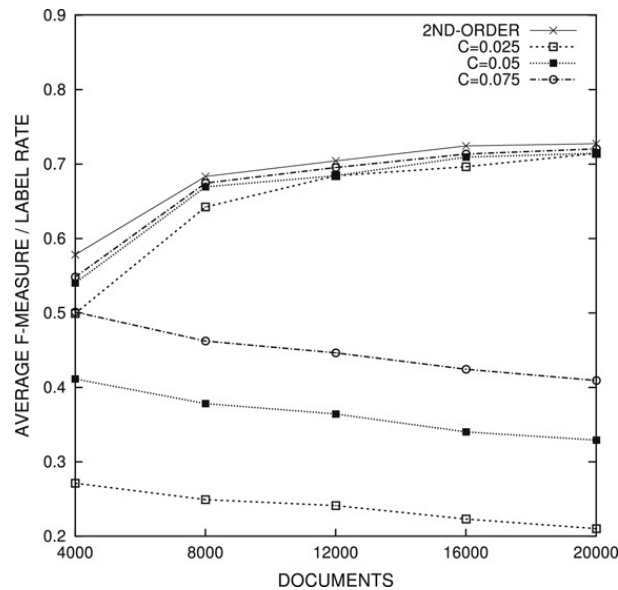


Fig. 3. Behaviour of the label efficient second-order Perceptron on the same news story categorization task of Fig. 2 (in this experiment we only used the first 20,000 documents). For each value of the parameter  $C$  two curves are plotted as a function of the number of instances observed. As in Fig. 2, the increasing curve is the average  $F$ -measure and the decreasing curve is the rate of sampled labels. Note that, for the range of parameter values displayed, the performance does not get significantly worse as the parameter value decreases causing the label rate to drop faster. In all cases, the performance remains close to that of the algorithm that queries all labels (corresponding to the choice  $C \rightarrow \infty$ ).

## Acknowledgement

This work was supported in part by the PASCAL Network of Excellence under EC grant no. 506778. This publication only reflects the authors' views.

## References

- [1] R. Rifkin, A. Klautau, In defense of one-vs-all classification, *Journal of Machine Learning Research* 5 (2004) 101–141.
- [2] J. Weston, C. Watkins, Support vector machines for multi-class pattern recognition, in: *Proceedings of the Seventh European Symposium on Artificial Neural Networks*, April 1999.
- [3] A. Elisseeff, J. Weston, A kernel method for multi-labeled classification, in: *Advances in Neural Information Processing Systems 14*, 2002.
- [4] K. Crammer, Y. Singer, Ultraconservative online algorithms for multiclass problems, *Journal of Machine Learning Research* 3 (2003) 951–991.
- [5] N. Cristianini, J. Shawe-Taylor, *Kernel Methods for Pattern Analysis*, Cambridge University Press, 2004.
- [6] B. Schölkopf, A. Smola, *Learning with Kernels*, MIT Press, 2002.
- [7] N. Cesa-Bianchi, A. Conconi, C. Gentile, Margin-based algorithms for information filtering, in: *Advances in Neural Information Processing Systems*, vol. 15, MIT Press, 2003, pp. 470–477.
- [8] N. Cesa-Bianchi, A. Conconi, C. Gentile, Learning probabilistic linear-threshold classifiers via selective sampling, in: *Proceedings of the 16th Annual Conference on Learning Theory*, in: *LNAI*, vol. 2777, Springer, 2003, pp. 373–386.
- [9] N. Cesa-Bianchi, C. Gentile, L. Zaniboni, Incremental algorithms for hierarchical classification, *Journal of Machine Learning Research* 7 (2006) 31–54.
- [10] E. Amaldi, V. Kann, The complexity and approximability of finding maximum feasible subsystems of linear relations, *Theoretical Computer Science* 147 (1995) 181–210.
- [11] S. Ben-David, N. Eiron, P.M. Long, On the difficulty of approximately maximizing agreements, *Journal of Computer and System Sciences* 66 (3) (2003) 496–514.
- [12] K. Crammer, O. Dekel, J. Keshet, S. Shalev-Shwartz, Y. Singer, Online passive-aggressive algorithms, *Journal of Machine Learning Research* 7 (2006) 551–585.
- [13] A. Hoerl, R. Kennard, Ridge regression: Biased estimation for nonorthogonal problems, *Technometrics* 12 (1970) 55–67.
- [14] N. Cristianini, J. Shawe-Taylor, *An Introduction to Support Vector Machines*, Cambridge University Press, 2001.
- [15] T. Graepel, R. Herbrich, J. Shawe-Taylor, Generalisation error bounds for sparse linear classifiers, in: *Proceedings of the 13th Annual Conference on Computational Learning Theory*, ACM Press, 2000, pp. 298–303.
- [16] N. Cesa-Bianchi, A. Conconi, C. Gentile, A second-order Perceptron algorithm, *SIAM Journal on Computing* 34 (3) (2005) 640–668.
- [17] D. Angluin, Queries and concept learning, *Machine Learning* 2 (4) (1988) 319–342.
- [18] N. Littlestone, Learning quickly when irrelevant attributes abound: a new linear-threshold algorithm, *Machine Learning* 2 (4) (1988) 285–318.

- [19] H. Block, The Perceptron: A model for brain functioning, *Review of Modern Physics* 34 (1962) 123–135.
- [20] A. Novikoff, On convergence proofs of Perceptrons, in: *Proceedings of the Symposium on the Mathematical Theory of Automata*, vol. XII, 1962, pp. 615–622.
- [21] L. Devroye, L. Györfi, G. Lugosi, *A Probabilistic Theory of Pattern Recognition*, Springer Verlag, 1996.
- [22] V. Vapnik, *Statistical Learning Theory*, Wiley, 1998.
- [23] C. Gentile, The robustness of the  $p$ -norm algorithms, *Machine Learning* 53 (3) (2003) 265–299.
- [24] R. Gilad-Bachrach, A. Navot, N. Tishby, Bayes and Tukey meet at the center point, in: *Proceedings of the 17th Annual Conference on Learning Theory*, in: *LNAI*, vol. 3120, Springer, 2004, pp. 549–563.
- [25] W. Hoeffding, Probability inequalities for sums of bounded random variables, *Journal of the American Statistical Association* 58 (1963) 13–30.
- [26] K. Azuma, Weighted sums of certain dependent random variables, *Tohoku Mathematical Journal* 68 (1967) 357–367.
- [27] N. Cesa-Bianchi, A. Conconi, C. Gentile, On the generalization ability of on-line learning algorithms, *IEEE Transactions on Information Theory* 50 (9) (2004) 2050–2057.
- [28] T. Hastie, R. Tibshirani, J. Friedman, *The Elements of Statistical Learning*, Springer, 2001.
- [29] V. Vovk, Competitive on-line statistics, *International Statistical Review* 69 (2001) 213–248.
- [30] [about.reuters.com/researchandstandards/corpus/](http://about.reuters.com/researchandstandards/corpus/).
- [31] N. Cesa-Bianchi, C. Gentile, L. Zaniboni, Worst-case analysis of selective sampling for linear classification, *Journal of Machine Learning Research* 7 (2006) 1205–1230.