A Simple Additive Re-weighting Strategy for Improving Margins

Fabio Aiolli and **Alessandro Sperduti**
Department of Computer Science, Corso Italia 40, Pisa, Italy
e-mail: {aiolli, perso}@di.unipi.it

Abstract

We present a sample re-weighting scheme inspired by recent results in margin theory. The basic idea is to add to the training set replicas of samples which are not classified with a sufficient margin. We prove the convergence of the input distribution obtained in this way. As study case, we consider an instance of the scheme involving a 1-NN classifier implementing a Vector Quantization algorithm that accommodates tangent distance models. The tangent distance models created in this way have shown a significant improvement in generalization power with respect to the standard tangent models. Moreover, the obtained models were able to outperform state of the art algorithms, such as SVM.

1 Introduction

In this paper we introduce a simple additive re-weighting method that is able to improve the margin distribution on the training set. Recent results in computational learning theory [Vapnik, 1998; Schapire *et al.*, 1998; Bartlett, 1998] have tightly linked the expected risk of a classifier (i.e. the probability of misclassification of a pattern drawn from an independent random distribution), with the distribution of the margins in the training set. In general, it results that we can expect best performances on generalization (minimal error on test data) when most of the patterns have high margins.

The aforementioned results are at the basis of the theory of two of the most impressive algorithms: Support Vector Machines and Boosting. Either SVM's and Boosting effectiveness is largely due to the fact that they, directly or not, effectively improve the margins on the training set. In particular, SVM explicitly finds the hyper-plane with the largest minimum margin in a dimensional-augmented space where training points are mapped by a kernel function. In this case, margin theory permits to explain impressive performances even in very high dimensional spaces where data are supposed to be more separated. Most of the recent efforts in SVMs are in the choice of the right kernels for particular applications. For example, in OCR problems, the polynomial kernel was proven to be very effective.

On the other side, boosting algorithms, and in particular the most famous version AdaBoost, produce weighted ensemble of hypotheses, each one trained in such a way to minimize the empirical error in a given "difficult" distribution of the training set. Again, it has been shown [Schapire, 1999] that boosting essentially is a procedure for finding a linear combination of weak hypotheses which minimizes a particular loss function dependent on the margins on the training set, literally $Loss = \sum_i \exp(-\mu_i)$. Recently, research efforts related to boosting algorithms faced the direct optimization of the margins on the training set. For example, this has been done by defining different margin-based cost functions and searching for combinations of weak hypotheses so to minimize these functions [Mason *et al.*, 1998].

We will follow a related approach that aims to find a single (eventually non linear) optimal hypothesis where the optimality is defined in terms of a loss-function dependent on the distribution of the margins on the training set. In order to minimize this loss we propose a re-weighting algorithm that maintains a set of weights associated with the patterns in the training set. The weight associated to a pattern is iteratively updated when the margin of the current hypothesis does not reach a predefined threshold on it. In this way a new distribution on the training data will be induced. Furthermore, a new hypothesis is then computed that improves the expectation of the margin on the new distribution. In the following we prove that the distribution converges to a uniform distribution on a subset of the training set.

We apply the above scheme to an OCR pattern recognition problem, where the classification is based on a 1-NN tangent distance classifier [Simard *et al.*, 1993], obtaining a significant improvement in generalization. Basically, the algorithm builds a set of models for each class by an extended version of the Learning Vector Quantization procedure (LVQ [Kohonen *et al.*, 1996]) adapted to tangent distance. In the following we will refer to this new algorithm as Tangent Vector Quantization (TVQ).

The paper is organized as follows. In Section 2, we introduce the concept of margin regularization via the input distribution on the training set. Specifically, we present the Θ-Margin Re-weighting Strategy, which holds the property to guarantee the convergence of the input distribution. In Section 3, we introduce a definition for the margins in a 1-NN scheme that considers the discriminative ratio observed for a particular pattern, and in Section 4 we define the TVQ algorithm. Finally, in Section 5 we present empirical results comparing TVQ with other 1-NN based algorithms, including SVM.

2 Regularization of the margins

When learning takes place, the examples tend to influence in a different way the discriminant function of a classifier. A discriminant function can be viewed as a resource that has to be shared among different clients (the examples). Often, when pure Empirical Risk Minimization (ERM) principle is applied, that resource is used in a wrong way since, with high probability, it is almost entirely used by a fraction of the training set. Margin theory formally tells us that it is preferable to regularize the discriminant function in such a way to make the examples sharing more equally its support.

Inspired on the basic ideas of margin optimization, here, we propose a simple general procedure applicable, eventually, to any ERM-based algorithm. It permits to regularize the parameters of a discriminant function so to obtain hypotheses with large margins for many examples in the training set.

Without generality loss we consider the margin for a training example as a real number, taking values in $[-1, +1]$, representing a measure of the confidence shown by a classifier in the prediction of the correct label. In a binary classifier, e.g. the perceptron, the margin is usually defined as $yf(x)$ where y is the target and $f(x)$ is the output computed by the classifier. Anyway, it can be easily re-conduced to the $[-1, +1]$ range by a monotonic (linear or sigmoidal) transformation of the output. In any case, a positive value of the margin must correspond to a correct classification of the example.

Given the function $\mu_h(x_i)$ that, provided an hypothesis h, associates to each pattern its margin, we want to define a loss-function that, when minimized, permits to obtain hypotheses with large margins (greater than a fixed threshold θ) for many examples in the training set. For this, we propose to minimize a function that, basically, is a re-formulation of SVM's slack variables:

$$L = \sum_{x_i \in S} (\theta - \mu_h(x_i)) 1_{[\theta - \mu_h(x_i)]}, \qquad (1)$$

where S is a training set with N examples, and $1_{[\theta-\mu_h(x_i)]} = 1$ if $\mu_h(x_i) < \theta$, and 0 otherwise. The L function is null for margins higher than the threshold θ and is linear with respect to the values of the margins when they are below this threshold.

We suggest to minimize L indirectly via a two-step iterative method that "simultaneously" (1) searches for an a priori distribution $\{\gamma_i\}$ for the examples that, given the current hypothesis h, better approximates the function $1_{[\theta-\mu_h(x)]}$ and (2) searches for a hypothesis h (e.g. by a gradient based procedure) that, provided the distribution $\{\gamma_i\}$, improves the weighted function

$$H = \sum_{p=1}^{N} \gamma_p \mu_h(x_p). \qquad (2)$$

This new formulation is equivalent to that given in eq. (1) provided that $\{\gamma_i\}$ converges to the uniform distribution on the θ-mistakes (patterns that have margin less than the threshold).

Θ-Margin Re-weighting Strategy

Input:
T: number of iterations;
$\mathcal{H}$: hypotheses space;
θ: margin threshold;
$k(\cdot) \in (0, K]$: bounded function;
$S = \{(x_i, y_i)\}_{i=1,..,N}$: training set;

Initialize
$h_0 \in \mathcal{H}$ (initial hypothesis);
for $i = 1, .., N$
$w_i(0) \leftarrow 1$, $\gamma_i(0) \leftarrow \frac{1}{N}$;

for $t = 1, .., T$
begin
find h_t such that
$\sum_{i=1}^{N} \gamma_i(t-1)\mu_{h_t}(x_i) > \sum_{i=1}^{N} \gamma_i(t-1)\mu_{h_{t-1}}(x_i)$;
$w_i(t) \leftarrow w_i(t-1) + k(t)1_{[\theta-\mu_{h_t}(x_i)]}$;
$\gamma_i(t) \leftarrow \frac{w_i(t)}{\sum_{j=1}^{N} w_j(t)}$;
end
return h_T;

Figure 1: The Θ-Margin Re-weighting Strategy.

The algorithm, shown in Figure 1, consists of a series of trials. An optimization process, that explicitly maximizes the function $\mu_h(x)$ according to the current distribution for the examples, works on an artificial training set S', initialized to be equal to the original training set S. For each t, $k(t)$ replicas of those patterns in S that have margin below the fixed threshold θ are added to S' augmenting their density in S' and consequently their contribution in the optimization process. Note that $w_i(t)$ denotes the number of occurrences in the extended training set S' of the pattern x_i.

In the following, we will prove that the simple iterative procedure just described makes the distribution approaching a uniform distribution on the θ-mistakes, provided that $k(t)$ is bounded.

2.1 Convergence of the distribution

For each trial t, given the margin of each example in the training set S, we can partition the training sample as $S = S_M(t) \cup S_C(t)$ where $S_M(t)$ is the set of θ-mistakes and $S_C(t) = S - S_M(t)$ is the complementary set of θ-correct patterns.

Let denote $W(t) = |S'(t)|$ and let $w_i(t)$ be the number of occurrences of pattern x_i in S' at time t, with density $\gamma_i(t) = w_i(t)/W(t)$. Moreover, let $\Lambda(t)$ be a suitable function of t such that $W(t+1) = W(t)\Lambda(t)$.
Let $w_i(t+1) = w_i(t)+k(t)$ be the update rule for the number of occurences in S', where $k(t)$ is bounded and takes values in $(0, K]$ (note that $k(t)$ may change at different iterations but it is independent from i). It's easy to verify that $\Lambda(t) \geq 1$ for each t because of the monotonicity of $W(t)$, and that $\Lambda(t) \rightarrow$

1 with the number of iterations. In fact

$$1 \leq \frac{W(t+1)}{W(t)} = \frac{W(t) + \Delta W(t)}{W(t)} \leq 1 + \frac{K|S|}{W(t)} \to 1.$$

At time $t+1$ we have

$$\gamma_i(t+1) = \frac{w_i(t) + k(t)1_{[\theta-\mu(t)]}}{\Lambda(t)W(t)} = \frac{\gamma_i(t) + \frac{k(t)}{W(t)}1_{[\theta-\mu(t)]}}{\Lambda(t)}.$$

First of all we show that the distribution converges. This can be shown by demonstrating that the changes tend to zero with the number of iterations, i.e., $\forall x_i \in S, |\Delta\gamma_i(t)| \to 0$.

We have

$$\Delta\gamma_i(t) = \frac{\frac{k(t)}{W(t)}1_{[\theta-\mu(t)]} - (\Lambda(t)-1)\gamma_i(t)}{\Lambda(t)},$$

which can be easily bounded in module by a quantity that tends to 0:

$$|\Delta\gamma_i(t)| \leq \frac{\frac{K}{W(t)} + (\Lambda(t)-1)\gamma_i(t)}{\Lambda(t)} \to 0.$$

We now show to which values they converge. Let m_i and ϵ_i be, respectively, the cumulative number and the mean ratio of θ-mistakes for x_i on the first t epochs, $\epsilon_i = \frac{m_i(t)}{t}$, then

$$\begin{aligned}\gamma_i(t) &= \frac{w_i(0) + E[k(t)]m_i(t)}{W(0) + E[k(t)]\sum_{j=1}^N m_j(t)} \\ &= \frac{1 + tE[k(t)]\epsilon_i}{N + tE[k(t)]\sum_{j=1}^N \epsilon_j} \to \frac{\epsilon_i}{\sum_{j=1}^N \epsilon_j}.\end{aligned}$$

Given the convergence of the optimization process that maximizes H in eq. (2), the two sets S_M and S_C are going to become stable and the distribution on S will tend to a uniform distribution in S_M (where $\epsilon_i \to 1$) and will be null elsewhere (where $\epsilon_i \to 0$). This can be understood in the following way as well.

Given the definition of the changes made on the gamma values on each iteration of the algorithm, we calculate the function that we indeed minimize. Since $\Delta w(t) = k(t) \cdot 1_{[\theta-\mu(t)]}$, after some algebra, we can rewrite $\Delta\gamma_i(t)$ as:

$$\Delta\gamma_i(t) = \frac{\Delta W(t)}{W(t+1)}\left(\frac{1_{[\theta-\mu_i(t)]}}{|S_M(t)|} - \gamma_i(t)\right).$$

Thus, $\Delta\gamma_i(t) = 0$ when $\gamma_i = \frac{1}{|S_M(t)|}1_{[\theta-\mu_i(t)]}$, for which the minimum of function

$$E_\gamma = \frac{\Delta W(t)}{W(t+1)}\left(\frac{1}{2}\sum_i \gamma_i^2(t) - \frac{1}{|S_M(t)|}\sum_i 1_{[\theta-\mu_i(t)]}\gamma_i(t)\right)$$

is reached. Note that, the minimum of E_γ is consistent with the constraint $\sum_i \gamma_i = 1$.

In general, the energy function is modulated by a term decreasing with the number of iterations, dependent on the $k(t)$ used but independent from gamma, that can be viewed as a sort of annealing introduced in the process.

In the following, we study a specific instance of the Θ-Margin Re-weighting Strategy.

3 Margins in a 1-NN framework and tangent distance

Given a training example $(x_i, y_i) \in S$, $y_i \in Y$ and a fixed number of models for each class, below, we give a definition of the margin for the example when classified by a distance based 1-NN classifier.

Given the example (x_i, y_i), let z_i^p and z_i^n be the squared distances between the nearest of the positive set of models and the nearest of the negative sets of models, respectively. We can define the margin of a pattern in the training set as:

$$\mu_i = \frac{z_i^n - z_i^p}{z_i^n + z_i^p}. \tag{3}$$

This formula takes values in the interval $[-1, +1]$ representing the confidence in the prediction of the 1-NN classifier. Higher values of the μ_i's can also be viewed as an indication of a higher discriminative power of the set of models with respect to the pattern. Moreover, a pattern will result correctly classified in the 1-NN scheme if and only if its margin is greater than zero.

In this paper, we are particularly interested in distances that are invariant to given transformations. Specifically, we refer to the *one-sided tangent distance* [Simard, 1994; Hastie *et al.*, 1995; Schwenk and Milgram, 1995b; 1995a], which computes the distance between a pattern x_1 and a pattern x_2 as the minimum distance between x_1 and the linear subspace $\tilde{x}_2(\alpha)$ approximating the manifold induced by transforming the pattern x_2 according to a given set of transformations:

$$d_T(x_1, x_2) = \min_\alpha \|x_1 - \tilde{x}_2(\alpha)\|. \tag{4}$$

If the transformations are not known a priori, we can learn them by defining, for each class y, a (one-sided) tangent distance model M_y, compounded by a *centroid* C_y (i.e., a prototype vector for class y) and a set of *tangent vectors* $\mathcal{T}_y = \{T_k\}$ (i.e., an orthonormal base of the linear subspace $\tilde{C}_y(\alpha)$), that can be written as $M_y(\alpha) = C_y + \sum_{k=1}^{|\mathcal{T}_y|}\alpha_k T_k$.

This model can be determined by just using the N_y positive examples of class y, i.e. $(x_i, y) \in S$, as

$$M_y = \arg\min_M \sum_{p=1}^{N_y} \min_{\theta_p} \|M(\theta_p) - x_p\|^2, \tag{5}$$

which can easily be solved by resorting to principal component analysis (PCA) theory, also called *Karhunen-Loéve Expansion*. In fact, equation (5) can be minimized by choosing C_y as the average over all available positive samples x_p, and $\mathcal{T}_y$ as the set of the most representative eigenvectors (principal components) of the covariance matrix $\Sigma_y = \frac{1}{N_y}\sum_{p=1}^{N_y}(x_p - C_y)(x_p - C_y)^t$.

The corresponding problem for the two-sided tangent distance can be solved by an iterative algorithm, called (two-sided) HSS, based on Singular Value Decomposition, proposed by Hastie et al. [Hastie *et al.*, 1995]. When the one-sided version of tangent distance is used, HSS and PCA coincide. So, in the following, the one sided version of this algorithm will be simply referred as to HSS.

Given a one-sided tangent distance model M_y, it is quite easy to verify that the squared tangent distance between a pattern x and the model M_y can be written as:

$$d_T^2(x, M_y) = \delta^t \delta - \sum_{k=1}^{|\mathcal{T}|} \alpha_k^2 \quad (6)$$

where $\delta = x - C_y$, $\alpha_k = \delta^t T_k$ and δ^t denotes the transpose of δ. Consequently, in our definition of margin, we have $z_i^p = \min_{M_y \in y_i} d_T^2(x_i, M_y)$, and $z_i^n = \min_{M_y \notin y_i} d_T^2(x_i, M_y)$.

Given this definition of margin, we can implement the choice of the new hypothesis in the Θ-Margin Re-weighting Strategy by maximizing the margin using gradient ascent on the current input distribution.

3.1 Improving margins as a driven gradient ascent

Considering the tangent distance formulation as given in equation (6) we can verify that it is defined by scalar products. Thus, we can derivate it with respect to the centroid C and the tangent vectors $\mathcal{T} = \{T_k\}$ of the nearest positive model obtaining:

$$\frac{\delta z_i^p}{\delta C} = -2(\delta - \sum_{k=1}^{|\mathcal{T}|} \alpha_k T_k), \qquad \frac{\delta z_i^p}{\delta T_k} = -2\alpha_k \delta.$$

Considering that $\frac{\delta \mu_i}{\delta C} = \frac{\delta \mu_i}{\delta z_i^p} \frac{\delta z_i^p}{\delta C}$ and $\frac{\delta \mu_i}{\delta T_k} = \frac{\delta \mu_i}{\delta z_i^p} \frac{\delta z_i^p}{\delta T_k}$ we can compute the derivative of the margin with respect to changes in the nearest positive model:

$$\begin{aligned} \frac{\delta \mu_i}{\delta C} &= 4 \frac{z_i^n}{(z_i^n + z_i^p)^2} (\delta - \sum_{k=1}^{|\mathcal{T}|} \alpha_k T_k), \\ \frac{\delta \mu_i}{\delta T_k} &= 4 \frac{z_i^n}{(z_i^n + z_i^p)^2} \alpha_k \delta. \end{aligned}$$

A similar solution is obtained for the nearest negative model since it only differs in changing the sign and in exchanging indexes n and p. Moreover, the derivatives are null for all the other models.

Thus, we can easily maximize the average margin in the training set if for each pattern presented to the classifier we move the nearest models in the direction suggested by the gradient. Note that, like in the LVQ algorithm, for each training example, only the nearest models are changed.

When maximizing the expected margin on the current distribution $\{\gamma_i\}$, i.e., H, for each model $M = (C, \{T_k\})$ we have:

$$\Delta C = \eta \sum_{i=1}^{|S|} \gamma_i \frac{\delta \mu_i}{\delta C}, \qquad \Delta T_k = \eta \sum_{i=1}^{|S|} \gamma_i \frac{\delta \mu_i}{\delta T_k},$$

where η is the usual learning rate parameter. In the algorithm (see Figure 2), for brevity, we will group the above variations by referring to the whole model, i.e., $\Delta M = \eta \sum_{i=1}^{|S|} \gamma_i \frac{\delta \mu_i}{\delta M}$.

TVQ Algorithm

Input:
- T: no. of iterations;
- Q: no. of models per class;
- θ: margin threshold;

Initialize
- $W \leftarrow N, \gamma_i \leftarrow \frac{1}{N}$;
- $\forall y, q$, initialize $M_y^{(q)} \leftarrow (C_y^{(q)}, \mathcal{T}_y^{(q)})$ with random models;

for $t = 1, .., T$
- $\forall y, \forall q,\ \Delta M_y^{(q)} = 0$;
- $\forall (x_i, y_i) \in S$, select $(q_p, \bar{y}_i, q_n)$ s.t. $M_{y_i}^{(q_p)}$ and $M_{\bar{y}_i}^{(q_n)}$ are the nearest, positive and negative, models. Compute μ_i as in eq. (3) and accumulate the changes on the nearest models

$$\Delta M_{y_i}^{(q_p)} \leftarrow \Delta M_{y_i}^{(q_p)} + \gamma_i (\frac{\delta \mu_i}{\delta M_{y_i}^{(q_p)}};$$

$$\Delta M_{\bar{y}_i}^{(q_n)} \leftarrow \Delta M_{\bar{y}_i}^{(q_n)} + \gamma_i \frac{\delta \mu_i}{\delta M_{\bar{y}_i}^{(q_n)}};$$

- $\forall y, \forall q,\ M_y^{(q)} = M_y^{(q)} + \eta \Delta M_y^{(q)}$ and orthonormalize its tangents;
- $\forall (x_i, y_i) \in S$, update the distribution γ_i by the rule

$$\gamma_i \leftarrow \gamma_i + \frac{1_{[\theta - \mu_i]}}{W}$$

- Normalize γ_i's such that $\sum_{i=1}^{N} \gamma_i = 1$;
- $W \leftarrow W + |\{(x_i, y_i) | \mu_i < \theta\}|$;

End

Figure 2: The TVQ Algorithm.

4 The TVQ algorithm

The algorithm (see Figure 2) starts with random models and a uniform distribution on the training set. For each pattern, the variation on the closest positive and the closest negative models are computed accordingly to the density of that pattern on the training set S'. When all the patterns in S are processed, the models are updated performing a weighted gradient ascent on the values of the margin. Moreover, for each pattern in the training set such that the value of the margin is smaller than a fixed value, the distribution is augmented. The effect is to force the gradient ascent to concentrate on hardest examples in the training set. As we saw in Section 2 the increment to the distribution is simply the effect of adding a replica ($k(t) = 1$) of incorrectly classified patterns to the augmented training set.

The initialization of the algorithm may be done in different ways. The default choice is to use random generated models however, when the training set size is not prohibitive, we can drastically speed up the algorithm by taking as initial models the ones generated by any algorithm (e.g., HSS). How-

Method	Parameters	Err%
HSS 1-sided	15 tangents	3.58
LVQ 2.1	16 codebooks	3.52
TD-Neuron	15 tangents	3.51
HSS 2-sided	9 tangents	3.40
Euclidean 1-NN	prototypes	3.16
SVM	Linear	10.64
SVM	Poly d=2	2.82
SVM	Poly d=3	3.23
SVM	Poly d=4	4.02

Table 1: Test results for different 1-NN methods.

ever, in the case of multiple models per class the initialization through the HSS method would generate identical models for each class and that would invalidate the procedure. A possible choice in this case, is to generate HSS models by using different random conditional distributions for different models associated to the same class. Another solution, which is useful when the size of the training set is relatively large, is to initialize the centroids as the average of the positive instances and then generating random tangents. Experimental results have shown that the differences on the performance obtained by using different initialization criteria are negligible. As we could expect the speed of convergence with different initialization methods may be drastically different. This is due to the fact that when TVQ is initialized with HSS models it starts with a good approximation of the optimal hypothesis (see Figure 3-(a)), while random initializations implicitly introduce an initial poor estimate of the final distribution due to the mistakes that most of the examples do on the first few iterations.

5 Results

We compared the TVQ algorithm versus SVMs and other 1-NN based algorithms: 1-sided HSS, 2-sided HSS, TD-Neuron [Sona *et al.*, 2000], and LVQ. The comparison was performed using exactly the same split of a dataset consisting of 10705 digits randomly taken from the NIST-3 dataset. The binary 128x128 digits were transformed into 64-grey level 16x16 images by a simple local counting procedure[1]. The only preprocessing performed was the elimination of empty borders. The training set consisted of 5000 randomly chosen digits, while the remaining digits were used in the test set.

The obtained results for the test data are summarized in Table 1. For each algorithm, we reported the best result, without rejection, obtained for the dataset. Specifically, for the SVM training we used the SVM^{Light} package available on the internet[2]. Different kernels were considered for the SVMs: linear and polynomial with degrees 2,3 and 4 (we used the default for the other parameters). Since SVMs are binary classifiers, we built 10 SVMs, one for each class against all the others, and we considered the overall prediction as the label with higher margin. The best performance has been obtained

[1]The number of pixel with value equal to 1 is used as the grey value for the corresponding pixel in the new image.

[2]http://www-ai.cs.uni-dortmund.de/SOFTWARE/SVM_LIGHT/

	$\theta = 0.3$		$\theta = 0.4$		$\theta = 0.1$
Q	$\lvert\mathcal{T}\rvert$ =10	$\lvert\mathcal{T}\rvert$ = 15	$\lvert\mathcal{T}\rvert$ =10	$\lvert\mathcal{T}\rvert$ =15	$\lvert\mathcal{T}\rvert$ = 0
1	2.40	2.20	3.00	2.22	6.40
3	2.10	2.26	2.43	2.10	–

Table 2: Test results for TVQ.

with a polynomial kernel of degree 2. We ran the TVQ algorithm with two different values for θ and four different architectures. Moreover, we ran also an experiment just using a single centroid for class (i.e., $|\mathcal{T}_y| = 0$) with $\theta = 0.1$. The smaller value for θ has been chosen just to account for the far smaller complexity of the model.

In almost all the experiments the TVQ algorithm obtained the best performance. Results on the test data are reported in Table 2. Specifically, the best result for SVM is worst than almost all the results obtained with TVQ. Particularly interesting is the result obtained by just using a single centroid for each class. This corresponds to perform an LVQ with just 10 codebooks, one for each class.

In addition, TVQ returns far more compact models allowing a reduced response time in classification. In fact, the 1-NN using polynomial SVMs with $d = 2$, needs 2523 support vectors, while in the worst case the models returned by the TVQ involve a total of 480 vectors (one centroid plus 15 tangents for each model).

In Figure 3, typical error curves for the training and test errors (3-(b)), as well as the margin distributions on the training set (3-(c)) and the induced margin distribution on the test set (3-(d)) are reported. From these plots it is easy to see that the TVQ doesn't show overfitting. This was also confirmed by the experiments involving the models with higher complexity and smaller values of θ. Moreover, the impact of the θ-margin on the final margin distribution on the training set is clearly shown in 3-(c), where a steep increase of the distribution is observed in correspondence of θ at the expenses of higher values of margin. Even if at a minor extent, a similar impact on the margin distribution is observed for the test data.

In Figure 4 we have reported the rejection curves for the different algorithms. As expected, the TVQ algorithm was competitive with the best SVM, resulting to be the best algorithm for almost the whole error range.

6 Conclusions

We proposed a provably convergent re-weighting scheme for improving margins, which focuses on “difficult” examples. On the basis of this general approach, we defined a Vector Quantization algorithm based on tangent distance, which experimentally outperformed state of the art classifiers both in generalization and model compactness. These results confirm that the control of the shape of the margin distribution has a great effect on the generalization performance.

When comparing the proposed approach with SVM, we may observe that, while our approach shares with SVM the Statistical Learning Theory concept of uniform convergence of the empirical risk to the ideal risk, it exploits the input distribution to directly work on non-linear models instead of resorting to predefined kernels. This way to proceed is

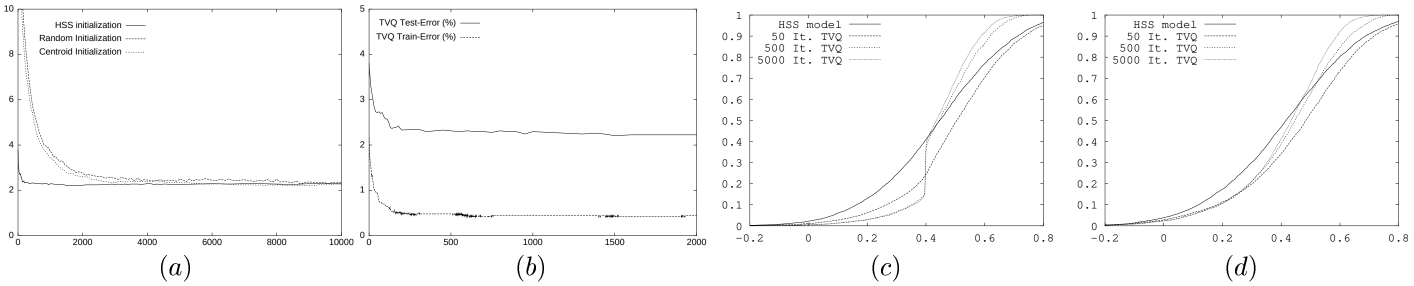

Figure 3: TVQ with 15×1 tangents, and $\theta = 0.4$: (a) comparison with different initialization methods; (b) test and training error; (c) cumulative margins on the training set at different iterations; (d) cumulative margins on the test set at different iterations.

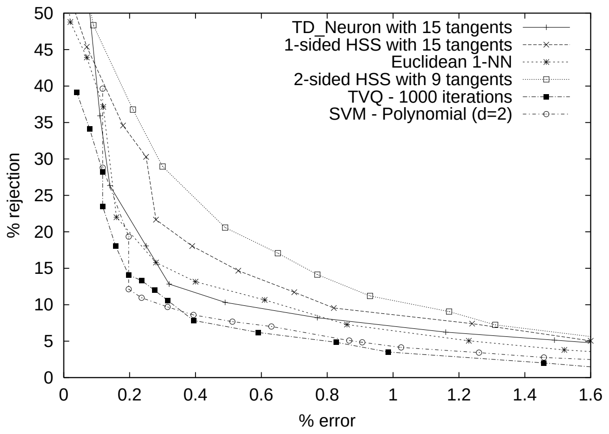

Figure 4: Detail of rejection curves for the different 1-NN methods. The rejection criterion is the difference between the distances of the input pattern with respect to the first and the second nearest models.

very similar to the approach adopted by Boosting algorithms. However, in Boosting algorithms, several hypotheses are generated and combined, while in our approach the focus is on a single hypothesis. This justifies the adoption of an additive re-weighting scheme, instead of a multiplicative scheme which is more appropriate for committee machines.

Acknowledgments

Fabio Aiolli wishes to thank Centro META - Consorzio Pisa Ricerche for supporting his PhD fellowship.

References

[Bartlett, 1998] P.L. Bartlett. The sample complexity of pattern classification with neural networks: the size of the weights is more important than the size of the network. *IEEE Trans. on Infor. Theory*, 44(2):525–536, 1998.

[Hastie *et al.*, 1995] T. Hastie, P. Y. Simard, and E. Säckinger. Learning prototype models for tangent distance. In G. Teasauro, D. S. Touretzky, and T. K. Leen, editors, *Advances in Neur. Inform. Proc. Systems*, volume 7, pages 999–1006. MIT Press, 1995.

[Kohonen *et al.*, 1996] T. Kohonen, J. Hynninen, J. Kangas, J. Laaksonen, and K. Torkkola. Lvq_pak: The learning vector quantization program package. Technical Report A30, Helsinki Univ. of Tech., Lab. of Computer and Inform. Sci., January 1996.

[Mason *et al.*, 1998] L. Mason, P. Bartlett, and J. Baxter. Improved generalization through explicit optimization of margins. Technical report, Dept. of Sys. Eng., Australian National University, 1998.

[Schapire *et al.*, 1998] R.E Schapire, Y. Freund, P. Bartlett, and W.S. Lee. Boosting the margin: A new explanation for the effectiveness of voting methods. *An. of Stat.*, 26(5), 1998.

[Schapire, 1999] R. Schapire. Theoretical views of boosting. In *Computational Learning Theory: Proc. of the 4th European Conference, EuroCOLT'99*, 1999.

[Schwenk and Milgram, 1995a] H. Schwenk and M. Milgram. Learning discriminant tangent models for handwritten character recognition. In *Intern. Conf. on Artif. Neur. Netw.*, pages 985–988. Springer-Verlag, 1995.

[Schwenk and Milgram, 1995b] H. Schwenk and M. Milgram. Transformation invariant autoassociation with application to handwritten character recognition. In G. Teasauro, D. S. Touretzky, and T. K. Leen, editors, *Advances in Neur. Inform. Proc. Systems*, volume 7, pages 991–998. MIT Press, 1995.

[Simard *et al.*, 1993] P. Y. Simard, Y. LeCun, and J. Denker. Efficient pattern recognition using a new transformation distance. In S. J. Hanson, J. D. Cowan, and C. L. Giles, editors, *Advances in Neural Information Processing Systems*, volume 5, pages 50–58. Morgan Kaufmann, 1993.

[Simard, 1994] P. Y. Simard. Efficient computation of complex distance metrics using hierarchical filtering. In J. D. Cowan, G. Teasauro, and J. Alspector, editors, *Advances in Neur. Inform. Proc. Systems*, volume 6, pages 168–175. Morgan Kaufmann, 1994.

[Sona *et al.*, 2000] D. Sona, A. Sperduti, and A. Starita. Discriminant pattern recognition using transformation invariant neurons. *Neur. Comput.*, 12(6):1355–1370, 2000.

[Vapnik, 1998] V. Vapnik. *Statistical Learning Theory*. Wiley, 1998.