Margin Analysis of the LVQ Algorithm

Koby Crammer *kobics@cs.huji.ac.il*
Ran Gilad-Bachrach *ranb@cs.huji.ac.il*
Amir Navot *anavot@cs.huji.ac.il*

Naftali Tishby
tishby@cs.huji.ac.il
School of Computer Science and Engineering and
Interdisciplinary Center for Neural Computation
The Hebrew University, Jerusalem, Israel

Abstract

One of the earliest and most powerful machine learning methods is the *Learning Vector Quantization* (LVQ) algorithm, introduced by Kohonen about 20 years ago. Still, despite its popularity, the theoretical justification of this model is quite limited. In this paper we fill the gap and provide margin based analysis for this model. We present a rigorous bound on the generalization error which is independent of the dimension of the data. Furthermore we show that LVQ is a family of maximal margin algorithms while the different variants emerge from different choices of loss functions. These novel insights are a starting point for developing a variety of new learning algorithms that are based on learning multiple data prototypes.

1 Introduction

Learning Vector Quantization (LVQ) is a well known algorithm in pattern recognition and supervised learning introduced by Kohonen [1]. LVQ is easy to implement, runs efficiently and in many cases provides state of the art performance. The algorithm was originally introduced as a labeled version of vector quantization, or clustering, and as such was closely related to unsupervised learning algorithms. Since the original introduction of LVQ the theory of supervised learning algorithms has been further developed and it provides more effective rigorous bounds on the generalization performance. Such analysis has not been done so far for the LVQ algorithm despite its very wide usage. One of the key theoretical (as well as practical) elements in the analysis of successful supervised learning algorithms is the concept of *margin*.

In this paper we apply margin analysis to provide theoretical reasoning for the LVQ algorithm. Roughly speaking margins measures the level of confidence a classifiers has with respect to it's decisions. Margin analysis has become a primary tool in machine learning during the last decade. Two of the most powerful algorithms in the field, *Support Vector Machines* (SVM) [2] and *AdaBoost* [3] are motivated and analyzed by margins. Since the introduction of these algorithms dozens of paper were published on different aspect of margins in supervised learning [4, 5, 6]. Buckingham and Geva [7] were the first to suggest

that LVQ is indeed a maximum margin algorithm. They presented a variant named LMVQ and analyzed it. As in most of the literature about LVQ they look at the algorithm as trying to estimate a density function (or a function of the density) at each point. After estimating the density the Bayesian decision rule is used. We take a different point of view on the problem. We look at the geometry of the decision boundary induced by the decision rule. Note that in order to generate good classification the only significant factor is where the decision boundary lies (It is a well known fact that classification is easier then density estimation [8]). Hence we define a geometrically based margin, similar to SVM [2] and use it for our analysis.

Summery of the Results In section 2 we present the model and outline the LVQ family of algorithms. A discussion and definition of margin is provided in section 3. The two fundamental results are a bound on the generalization error and a theoretical reasoning for the LVQ family of algorithms. In section B we present a bound on the gap between the empirical and the generalization accuracy. This provides a guaranty on the performance over unseen instances based on the empirical evidence. Although LVQ was designed as an approximation to *Nearest Neighbor* the theorem suggests that the former is more accurate in many cases. Indeed a simple experiment shows this prediction to be true.

In section 5 we show how LVQ family of algorithms emerge from the previous bound. These algorithms minimize the bound using gradient descent. The different variants correspond to different tradeoff between opposing quantities. In practice the tradeoff is controlled by *loss functions*.

2 Problem Setting and the LVQ algorithm

The framework we are intersted in is supervised learning for classification problems. In this framework the task is to find a map from $\mathbb{R}^n$ into a finite set of labels $\mathcal{Y}$. The model we focus on is a classification functions of the following form: The classifiers are parameterized by a set of points $\mu_1, \ldots, \mu_k \in \mathbb{R}^n$ which we refer to as *prototypes*. Each prototype is associated with a label $y \in \mathcal{Y}$. Given a new instance $x \in \mathbb{R}^n$ we predict that it has the same label as the closest prototype, similar to the 1-nearest-neighbor rule (1-NN). We denote the label predicted using a set of prototypes $\{\mu_j\}_{j=1}^k$ by $\mu(x)$. The goal of the learning process in this model is to find a set of prototypes which will predict accurately the labels of unseen instances.

The Learning Vector Quantization (LVQ) family of algorithms works in this model. The algorithms get as an input a labeled sample $S = \{(x_l, y_l)\}_{l=1}^m$, where $x_l \in \mathbb{R}^n$ and $y_l \in \mathcal{Y}$ and use it to find a good set of prototypes. All the variants of LVQ share the following common scheme. The algorithm maintains a set of prototypes each is assigned with a predefined label, which is kept constant during the learning process. It cycles through the training data S and in each iteration modifies the set of prototypes in accordance to one instance (x_t, y_t). If the prototype μ_j has the same label as y_l it is attracted to x_t but if the label of μ_j is different it is repelled from it. Hence LVQ updates the closest prototypes to x_t according to the rule:

$$\mu_j \leftarrow \mu_j \pm \alpha_t (x_t - \mu_j) \;, \tag{1}$$

where the sign is positive if the label of x_t and μ_j agree, and negative otherwise. The parameter α_t is updated using a predefined scheme and controls the rate of convergence of the algorithm. The variants of LVQ differ in which prototypes they choose to update in each iteration and in the specific scheme used to modify α_t.

For instance, LVQ1 and OLVQ1 updates only the closest prototype to x_t in each iteration. Another example is the LVQ2.1 which modifies the two closest prototypes μ_i and μ_j to x_t. It uses the same update rule (1) but apply it only if the following two conditions hold :

1. Exactly one of the prototypes has the same label as x_t, i.e. y_t.
2. The ratios of their distances from x_t falls in a window: $1/s \leq \|x_t - \mu_i\| / \|x_t - \mu_j\| \leq s$, where s is the window size.

More variants of LVQ can be found in [1]. We now turn to discuss the notion of margins and describe their applications in our setting.

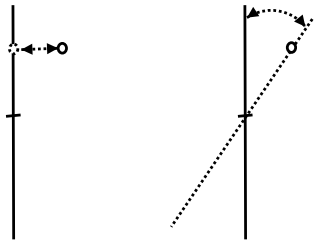
3 Margins

Margins play an important role in current research in machine learning. It measures the confidence a classifier has when making its predictions. One approach is to define margin as the distance between an instance and the decision boundary induced by the classification rule as illustrated in figure 1(a). Support Vector Machines [2] are based on this definition of margin, which we reffer to as *Sample-Margin*. However, an alternative definition of margin can be defined, *Hypothesis Margin*. In this definition the margin is the distance that the classifier can travel without changing the way it labels any of the sample points. Note that this definition requires a distance measure between clasifiers. This type of margin is used in AdaBoost [3] and is illustreated in figure 1(b).

It is possible to apply these two types of margin in the context of LVQ. Recall that in our model a classifer is defined by a set of labeled prototypes. Such a classifier generates a decision boundary by Voronoy tessellation. Although using sample margin is more natural as a first choice, it turns out that this type of margin is both hard to compute and numerically unstable in our context, since small reloactions of the protopyes might lead to a dramatic change in the sample margin. Hence we focus on the hypothesis margin and thus have to define a distance measure between two classifers. We choose to define it as the maximal distance between prototypes pairs as illustrated in figure 2. Formally, let $\mu = \{\mu_j\}_{j=1}^k$ and $\hat{\mu} = \{\hat{\mu}_j\}_{j=1}^k$ define two classifers, then

$$\rho(\mu, \hat{\mu}) = \max_{i=1}^{k} \|\mu_i - \hat{\mu}_i\|_2 \ .$$

Note that this definition is not invariant to permutations of the prototypes but it upper bounds the invariant definition. Furthermore, the induced margin is easy to compute (lemma 1) and lower bounds the sample-margin.



(a) (b)

Figure 1: *Sample Margin* (figure 1(a)) measures how much can an **instance** travel before it hits the decision boundary. On the other hand *Hypothesis Margin* (figure 1(b)) measures how much can the **hypothesis** travel before it hits an instance.

Lemma 1 *Let $\mu = \{\mu_j\}_{j=1}^k$ be a set of prototypes. Let x be sample point then the hypothesis margin of μ with respect to x is $\theta = \frac{1}{2}(\|\mu_j - x\| - \|\mu_i - x\|)$ where μ_i is the closest prototype to x with the same label as x and μ_j is the closest prototype with alternative label.*

Lemma 2 *Let $S = \{x_l\}_{l=1}^m$ be a sample and $\mu = (\mu_1, \ldots, \mu_k)$ be a set of prototypes.*

$$\textit{sample-margin}_S(\mu) \ \geq \textit{hypothesis-margin}_S(\mu)$$

Lemma 2 shows that if we find a set of prototypes with large hypothesis margin then it has large sample margin as well.

4 Margin Based Generalization Bound

In this section we prove bound on the generalization error of LVQ type of classifiers and draw some conclusions.

When a classifier is applied to a training data it is natural to use the training error as a prediction to the generalization error (the probability of misclassification of an unseen instance). In prototype based hypothesis the classifier assigns a confidence level, i.e. margin, to it's predictions. Taking into account the margin by counting instances with small margin as mistakes gives a better prediction and provide a bound on the generalization error. This bound is given in terms of the number of prototypes, the sample size, the margin and the margin based empirical error. The following theorem states this result formally.

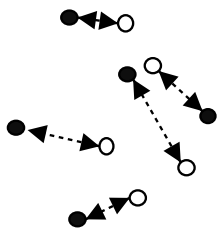


Figure 2: The distance measure on the LVQ hypothesis class. The distance between the white and black prototypes set is the maximal distance between prototypes pairs.

Theorem 1 *In the following setting:*

- *Let* $S = \{x_i, y_i\}_{i=1}^m \in \{\mathbb{R}^n \times \mathcal{Y}\}^m$ *be a training sample drawn by some underlying distribution* $\mathcal{D}$.
- *Assume that* $\forall i \quad \|x_i\| \leq R$.
- *Let* μ *be a set of prototypes with* k *prototypes from each class.*
- *Let* $0 < \theta < 1/2$.
- *Let* $\alpha_S^\theta(\mu) = \frac{1}{m}\left|\{i : \mathit{margin}_\mu(x_i) < \theta\}\right|$.
- *Let* $e_\mathcal{D}(\mu)$ *be the generalization error:* $e_\mathcal{D}(\mu) = \mathit{Pr}_{(x,y)\sim\mathcal{D}}\left[\mu(x) \neq y\right]$.
- *Let* $\delta > 0$.

Then with probability $1 - \delta$ *over the choices of the training data:*

$$\forall \mu \qquad e_\mathcal{D} \leq \alpha_S^\theta(\mu) + \sqrt{\frac{8}{m}\left(d \log^2 \frac{32m}{\theta^2} + \log\frac{4}{\delta}\right)} \tag{2}$$

where d *is the VC dimension:*

$$d = \min\left(n+1, \frac{64R^2}{\theta^2}\right) 2k^{|\mathcal{Y}|} \log ek^2 \tag{3}$$

This theorem leads to a few interesting observations. First, note that the bound is dimension free, in the sense that the generalization error is bounded independently of the input dimension (n) much like in SVM. Hence it makes sense to apply these algorithms with kernels.

Second, note that the VC dimension grows as the number of prototypes grows (3). This suggest that using too many prototypes might result in poor performance, therefore there is a non trivial optimal number of prototypes. One should not be surprised by this result as it is a realization of the *Structural Risk Minimization* (SRM) [2] principle. Indeed simple experiment shows this prediction to be true. Hence not only that prototype based methods are faster than *Nearest Neighbor*, they are more accurate as well. Due to space limitations proofs are provided in the full version of this paper only.

5 Maximizing Hypothesis Margin Through Loss Function

Once margins are properly defined it is natural to ask for algorithms that maximize margin. We will see that this is exactly what LVQ does. Before going any further we have to understand why maximizing the margin is a good idea.

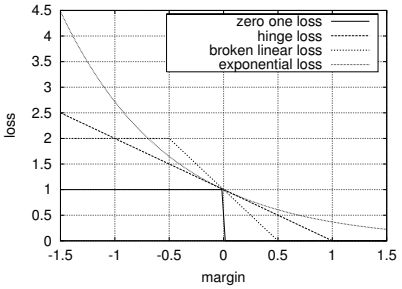


Figure 3: Different loss functions. SVM, LVQ1 and OLVQ1 use the "hinge" loss: $(1-\theta)_+$. LVQ2.1 uses the broken linear: $\min(2,(1-2\theta)_+)$. AdaBoost use the exponential loss ($e^{-\theta}$).

In theorem 1 we saw that the generalization error can be bounded by a function of the margin θ and the empirical θ-error (α). Therefore it is natural to seek prototypes that obtain small θ-error for a large θ. We are faced with two contradicting goals: small θ-error verses large θ. A natural way to solve this problem is through the use of loss function.

Loss function are a common technique in machine learning for finding the right balance between opposed quantities [9]. The idea is to associate a margin based loss (a "cost") for each hypothesis with respect to a sample. More formally, let L be a function such that:

1. For every θ: $L(\theta) \geq 0$.
2. For every $\theta < 0$: $L(\theta) \geq 1$.

We use L to compute the loss of an hypothesis with respect to one instance. When a training set is available we compute the loss of the hypothesis with respect to the training data by summing over the instances: $\mathcal{L}(\mu) = \sum_l L(\theta_l)$ where θ_l is the margin of the l'th instance in the training data. The two axioms of loss functions guarantee that $\mathcal{L}(\mu)$ bounds the empirical error.

It is common to add more restrictions on the loss function, such as requiring that L is a non-increasing function. However, the only assumption we make here is that the loss function L is differentiable.

Different algorithms use different loss functions [9]. AdaBoost uses the exponential loss function $L(\theta) = e^{-\beta\theta}$ while SVM uses the "hinge" loss $L(\theta) = (1-\beta\theta)_+$, where $\beta > 0$ is a scaling factor. See figure 3 for a demonstration of these loss functions.

Once a loss function is chosen, the goal of the learning algorithm is finding an hypothesis that minimizes it. Gradient descent is a natural simple choice for the task. Recall that in our case $\theta_l = (\|x_l - \mu_i\| - \|x_l - \mu_j\|)/2$ where μ_j and μ_i are the closest prototypes to x_l with the correct and incorrect labels respectively. Hence we have that [1]

$$\frac{d\theta_l}{d\mu_r} = S_l(r)\frac{x_l - \mu_r}{\|x_l - \mu_r\|}$$

where $S_l(r)$ is a sign function such that

$$S_l(r) = \begin{cases} 1 & \text{if } \mu_r \text{ is the closest prototype with correct label.} \\ -1 & \text{if } \mu_r \text{ is the closest prototype with incorrect label.} \\ 0 & \text{otherwise.} \end{cases}$$

Taking the derivative of $\mathcal{L}$ with respect to μ_r using the chain rule we obtain

$$\frac{d\mathcal{L}}{d\mu_r} = \sum_l \frac{dL(\theta_l)}{d\theta_l} S_l(r)\frac{x_l - \mu_r}{\|x_l - \mu_r\|} \tag{4}$$

[1] Note that if $x_l = \mu_j$ the derivative is not defined. This extreme case does not affect our conclusions, hence or the sake of clarity we avoid the treatment of such extreme cases in this paper.

Algorithm 1 Online Loss Minimization.
Recall that L is a loss function, and γ_t varies to zero as the algorithm proceeds.

1. Choose an initial positions for the prototypes $\{\mu_j\}_{j=1}^k$.
2. For $t = 1 : T($ or $\infty)$
   (a) Receive a labeled instance x_t, y_t
   (b) Compute the closest correct and incorrect prototypes to x_t: μ_j, μ_i, and the margin of x_t, i.e. $\theta_t = 1/2(\|\mu_j - x_t\| - \|\mu_i - x_t\|)$
   (c) Apply the update rule for $r = i, j$:

$$\mu_r \leftarrow \mu_r + \gamma_t \frac{dL(\theta_t)}{d\theta} S_l(r) \frac{x_t - \mu_r}{\|x_t - \mu_r\|}$$

By comparing the derivative to zero we get that the optimal solution is achieved when

$$\mu_r = \sum_l w_l^r x_l$$

where $\alpha_l^r = \frac{dL(\theta_l)}{d\theta_l} \frac{S_l(r)}{\|x_l - \mu_r\|}$ and $w_l^r = \frac{\alpha_l^r}{\sum_l \alpha_l^r}$. This leads to two conclusions. First, the optimal solution is in the span of the training instances. Furthermore, from its definition it is clear that $w_l^r \neq 0$ only for the closest prototypes to x_l. In other words, $w_l^r \neq 0$ if and only if μ_r is either the closest prototype to x_l which have the same label as x_l, or the closest prototype to x_l with alternative label. Therefore the notion of support vectors [2] applies here as well.

5.1 Minimizing The Loss

Using (4) we can find a local minima of the loss function by a gradient descent algorithm. The iteration in time t computes:

$$\mu_r(t+1) \leftarrow \mu_r(t) + \gamma_t \sum_l \frac{dL(\theta_l)}{d\theta} S_l(r) \frac{x_l - \mu_r(t)}{\|x_l - \mu_r(t)\|}$$

where γ_t approaches zero as t increases. This computation can be done iteratively where in each step we update μ_r only with respect to one sample point x_l. This leads to the following basic update step

$$\mu_r \leftarrow \mu_r + \gamma_t \frac{dL(\theta_l)}{d\theta} S_l(r) \frac{x_l - \mu_r}{\|x_l - \mu_r\|}$$

Note that $S_l(r)$ differs from zero only for the closest correct and incorrect prototypes to x_l, therefore a simple online algorithm is obtained and presented as algorithm 1.

5.2 LVQ1 and OLVQ1

The online loss minimization (algorithm 1) is a general algorithm applicable with different choices of loss functions. We will now apply it with a couple of loss functions and see how LVQ emerges. First let us consider the "hinge" loss function. Recall that the hinge loss is defined to be $L(\theta) = (1 - \beta\theta)_+$. The derivative [2] of this loss function is

[2] The "hinge" loss has no derivative at the point $\theta = 1/\beta$. Again as in other cases in this paper, this fact is neglected.

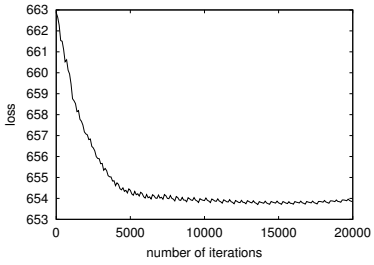


Figure 4: The "hinge" loss function ($\sum(1-\theta_l)_+$) vs. number of iteretions of OLVQ1. One can clearly see that it decreases.

$$\frac{dL(\theta)}{d\theta} = \begin{cases} 0 & \text{if } \theta > 1/\beta \\ -\beta & \text{otherwise} \end{cases}$$

If β is chosen to be large enough, the update rule in the online loss minimization is

$$\mu_r = \mu_r \pm \gamma_t \beta \frac{x_t - \mu_r}{\|x_t - \mu_r\|}$$

This is the same update rule as in LVQ1 and OLVQ1 algorithm [1] beside the extra factor of $\frac{\beta}{\|x_t-\mu_r\|}$ However, this is a minor difference since $\beta / \|x_t - \mu_r\|$ is just a normalizing factor. A demonstration of the affect of OLVQ1 on the "hinge" loss function is provided in figure 4. We applied the algorithm to a simple toy problem consisting of three classes and a training set of 1000 points. We allowed the algorithm 10 prototypes for each class. As expected the loss decreases as the algorithm proceeds. For this perpose we used the lvq_pak package[10].

5.3 LVQ2.1

The idea behind the definition of margin, and especially hypothesis margin was that a minor change in the hypothesis can not change the way it labels an instance which had a large margin. Hence when making small updates (i.e. small γ_t) one should focus only on the instances which have margins close to zero. The same idea appeared also in Freund's boost by majority algorithm [11].

Kohonen adapted the same idea to his LVQ2.1 algorithm [1]. The major difference between LVQ1 and LVQ2.1 algorithm is that LVQ2.1 updates μ_r only if the margin of x_t falls inside a certain window. The suitable loss function for LVQ2.1 is the broken linear loss function (see figure 3). The broken linear loss is defined to be $L(\theta) = \min(2, (1-\beta\theta)_+)$. Note that for $|\theta| > 1/\beta$ the loss is constant (i.e. the derivative is zero), this causes the learning algorithm to overlook instances with too high or too low margin. There exist several differences between LVQ2.1 and the online loss minimization presented here, however these differences are minor.

6 Conclusions and Further Research

In this paper a theoretical reasoning for the LVQ type of algorithms has been presented. We have shown that it generates a large margin classifier. We have also shown how different choices of margin-based loss functions yield different flavors of the algorithm.

This formulation allows easy generating of other similar algorithms in several different ways. The first and most obvious one is by using other loss functions such as the exponential loss. The second and more interesting way is by using other classification rule. Assume that one would like to replace the 1-NN rule with another classification rule during the generalization stage. k-NN, parzan window or any other prototypes based classification function are examples of possible choices. The proper way to adapt the algorithm to the chosen classification rule is by defining a suitable margin, and then adapting the minimization process that is done during the training stage accordingly. We have constructed some basic experiments using the k-NN rule. However the performance we have got did not exceed those of the 1-NN rule. We suggest the following explanation of these results. Usually the k-NN rule perform better than the 1-NN rule as it filters noise better, and in the LVQ setting the noise filtering is already achieved by using a small number of prototype.

A third extension can be devised by replacing the gradient descent that is used for the minimization of the loss function with other minimization techniques such as annealing. As different minimization methods give different trade off between computational complexity and accuracy, a wise choice can be made for a specific task. Another extension is using a different distance measure instead of the l_2 norm. This may result in more complicated formula of the derivative of the loss function, but may improve the results significantly in some cases. One specific interesting distance measure is the Tangent Distance [12].

We also presented a generalization guarantee for LVQ. The bound of the generalization error is based on the margin training error and is dimension free. This means that the algorithm can work well for high dimensional problems, and suggests that kernel version of LVQ may yield good performance. A kernel version is possible as the algorithm can be expressed as function of the inner product of couples of sample points only. We have implemented such kernel version and performed some initial experiments. It seems that using kernel does improve the results when is used with small number of prototypes. However, using the standard version with more prototypes achieve the same improvement.

A possible explanation of this phenomenon is the following. Recall that a classifier is a set of labeled prototypes that define a Voronoy tessellation. The decision boundary of such a classifier is built of some of the lines of the Voronoy tessellation. In standard version these lines are straight lines. In the kernel version these lines are smooth non-linear curves. As the number of prototypes grows, the decision boundary consists of more, and shorter lines. Now, if we remember the fact that any smooth curve can be approximated by a broken linear line, we come to the conclusion that any classifier that can be generated by the kernel version, can be approximated by one that is genereted by the standard version, when is applied with more prototypes. However, more intensive experiments should be conducted before drawing solid conclusions.

Acknowledgment We would like to thank Yoram Singer and Gal Chechik for their contribution to this paper.

References

[1] T. Kohonen. *Self-Organizing Maps*. Springer-Verlag, 1995.

[2] V. Vapnik. *The Nature Of Statistical Learning Theory*. Springer-Verlag, 1995.

[3] Y. Freund and R. E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1):119–139, 1997.

[4] R. E. Schapire, Y. Freund, P. Bartlett, and W. S. Lee. Boosting the margin : A new explanation for the effectiveness of voting methods. *Annals of Statistics*, 1998.

[5] Llew Mason, P. Bartlett, and J. Baxter. Direct optimization of margins improves generalization in combined classifier. *Advances in Neural Information Processing Systems*, 11:288–294, 1999.

[6] C. Campbell, N. Cristianini, and A. Smola. Query learning with large margin classifiers. In *International Conference on Machine Learning*, 2000.

[7] L. Buckingham and S. Geva. Lvq is a maximum margin algorithm. In *Pacific Knowledge Acquisition Workshop PKAW'2000*, 2000.

[8] L. Devroye, L. Gyorfi, and G. Lugosi. *A Probabilistic Theory of Pattern Recognition*. Springer, New York, 1996.

[9] Y. Singer and D. D. Lewis. Machine learning for information retrieval: Advanced techniques. presented at ACM SIGIR 2000, 2000.

[10] T. Kohonen, J. Hynninen, J. Kangas, and K. Laaksonen, J. Torkkola. Lvq_pak, the learning vector quantization program package. http://www.cis.hut.fi/research/lvq_pak, 1995.

[11] Y. Freund. Boosting a weak learning algorithm by majority. *Information and Computation*, 121(2):256–285, 1995.

[12] P. Y. Simard, Y. A. Le Cun, and Denker. Efficient pattern recognition using a new transformation distance. In *Advances in Neural Information Processing Systems*, volume 5, pages 50–58. 1993.

[13] N. Alon, S. Ben-David, N. Cesa-Bianchi, and D. Haussler. Scale-sensativie dimensions, uniform convergence, and learnability. *Journal of the ACM*, 44(4):615–631, 1997.

[14] N. Sauer. On densities of families of sets. *Journal of Combinatorics Theory*, 13:145–147, 1972.

[15] V. Vapnik and A. Y. Chervonenkis. On the uniform covergence of relative frequencies of events to their probabilities. *Theory of Probability and its Applications*, 16(2):264–280, 1971.

[16] N. Cristianini and J. Shawe-Taylor. *Support Vector MAchines*. Cambridge University Press, 2000.

A Lemmas on LVQ margin

In this appendix we provide the proofs for two lemmas on LVQ margin.

Lemma 3 *Let $S = \{x_i\}_{i=1}^m$ be a sample and $\mu = (\mu_1, \ldots, \mu_k)$ be a set of prototypes.*

$$\textit{sample-margin}_S(h_\mu) \geq \textit{hypothesis-margin}_S(h_\mu)$$

Proof: Let θ be greater than the Sample Margin of μ. I.e. there exists $x_i \in S$ and $\hat{x}$ such that $\|x_i - \hat{x}\| \leq \theta$ but x_i and $\hat{x}$ are labeled differently by μ. Since they are labeled differently, the closest prototype to x_i is different then the closest prototype to $\hat{x}$. W.L.G let μ_1 be the closest prototype to x_i and μ_2 be the closest prototype to $\hat{x}$.

Let $w = x_i - \hat{x}$ then $\|w\| \leq \theta$. Define $\forall j \;\; \hat{\mu}_j = \mu_j + w$. We claim that $\hat{\mu}_2$ is the closest prototype to x_i in $\hat{\mu}$. This follows since

$$\|x_i - \hat{\mu}_j\| = \|x_i - (\mu_j + w)\| = \|\hat{x} - \mu_j\|$$

Hence $\hat{\mu}$ assigns x_i a different label and also $\rho(\mu, \hat{\mu}) = \|w\| \leq \theta$ and therefore the hypothesis margin is less then θ.

Since this result holds for any θ greater than the sample margin of μ we conclude that the hypothesis margin of μ is less then or equal to it's sample margin. □

Lemma 4 *Let $\mu = \{\mu_j\}_{j=1}^k$ be a set of prototypes. Let x be an instance then the hypothesis margin of μ with respect to x is $\theta = \frac{1}{2}\left(\|\mu_j - x\| - \|\mu_i - x\|\right)$ where μ_i is the closest prototype to x and μ_j is the closest prototype to x with different label then μ_i .*

Proof: Let θ, x, μ_i and μ_j be as defined in the statement of the lemma. Let $\hat{\mu} = (\hat{\mu}_1, \ldots, \hat{\mu}_k)$ be such that $\rho(\mu, \hat{\mu}) < \theta$. Therefore from the triangle inequality $\|\hat{\mu}_i - x\| \leq \|\mu_i - x\| + \|\hat{\mu}_i - \mu_i\| < \|\mu_i - x\| + \theta$.

On the other hand for any index r we have that $\|\hat{\mu}_r - x\| \geq \|\mu_r - x\| - \|\hat{\mu}_r - \mu_r\| > \|\mu_r - x\| - \theta$. From the definition of $\hat{\mu}_r$ we have that if the label of μ_r is different then the label of μ_i then $\|\mu_r - x\| \geq \|\mu_j - x\|$ and so $\|\hat{\mu}_r - x\| > \|\mu_j - x\| - \theta$.

Finally, we have $\|\hat{\mu}_i - x\| < \|\mu_i - x\| + \theta = \|\mu_j - x\| - \theta < \|\hat{\mu}_r - x\|$ for any r such that μ_r has different label then μ_i and therefore the hypothesis margin of μ is at least θ.

For any $\epsilon > 0$ define $\hat{\mu}$ as follows: For any index l define $w_l = \frac{x - \mu_l}{\|x - \mu_l\|}$ (if $\|x - \mu_l\| = 0$ chose w_l to be any unit vector) hence $\|w_l\| = 1$. Now for any r such that the label of μ_r is different then that of μ_i define $\hat{\mu}_r = \mu_r + (\theta + \epsilon) w_r$. If the label of μ_s is the same as that of μ_i define $\hat{\mu}_s = \mu_s - (\theta + \epsilon) w_s$. It is easy to verify that $\rho(\mu, \hat{\mu}) = \theta + \epsilon$.

On the other hand, for any prototype $\hat{\mu}_s$ which has the same label as μ_i we have that

$$\begin{aligned} \|x - \hat{\mu}_s\| &= \left\| (x - \mu_s)\left(1 + \frac{\theta + \epsilon}{\|x - \mu_s\|}\right) \right\| \\ &= \|x - \mu_s\| + \theta + \epsilon \end{aligned}$$

If $\hat{\mu}_r$ has different label then μ_i then using the same calculation we get that $\|x - \hat{\mu}_r\| = \|x - \mu_r\| - \theta - \epsilon$ and therefore $\hat{\mu}_j$ is the closest prototype to x and it labels it differently. Therefore the hypothesis margin is less then $\theta + \epsilon$ for any $\epsilon > 0$ and so we conclude that the margin is exactly θ. □

B Analysis of LVQ

In this section we present some theoretical results concerning the LVQ type algorithms. We show a bound on the fat-shattering dimension [13] of the algorithm. The bound is independent of the dimension of the problem. We use this bound to show a bound on the generalization error of the algorithm. This bound holds for all margin based classifiers. We show the theorems for binary problems, although they hold for the general case as well.

The following definition will assist us in the discussion:

Definition 1 (θ-error) *Let h be a classifier and $S = \{(x_i, y_i)\}_{i=1}^m$ be a labeled sample. The θ-error of h with respect to S is the portion of the sample points on which h errs or makes a prediction with margin less then θ. This can be defined using the following formula*

$$\varepsilon_S^\theta(h) = \frac{\left|\left\{i : \left(\textit{signed-margin}_{\{x_i\}}(h) < \theta\right)\right\}\right|}{m}$$

Note that this definition works with respect to both types of margins and according to the choice of margin one interprets the term $\textit{margin}_{\{x_i\}}(h)$.

Theorem 2 presents a bound on the VC dimension (or fat-shattering dimension) of LVQ classifiers.

Theorem 2 *Assume we have k prototypes for each class. Assume that the sample points come from a ball of radius R. Then the largest sample that can be shattered with margin θ is smaller then $\min\left(n+1, \frac{R^2}{\theta^2}\right) 2k^2 \log_2 ek^2$ where n is the input dimension.*

Before proving the theorem we show that each LVQ rule can be decomposed into a set of linear classifiers. Each LVQ rule can be defined as a two layer network where the input level has k^2 linear separators (perceptrons) and the output is a predefined binary rule. For each couple of prototypes μ^+ and μ^- which have different labels we define a perceptron that tags the instances closer to μ^+ by "+1" and instances closer to μ^- by "-1". Given a point x, we apply the k^2 perceptrons. Note, if μ^+ is the closest prototype to x then all the perceptrons which are associated with μ^+ will tag it $+1$. In this case the output layer outputs $+1$. Similarly if μ^- is the closest prototype then all the perceptrons associated to it outputs -1 and so the output of the network is -1. Therefore the choice of k^2 prototypes fully determines the output of the network.

If a set of prototypes generates an hypothesis with hypothesis margin of θ then due to lemma 2, we know that the distance between every sample point to the decision boundary is at least θ. This means that if μ^+ is the closest prototype to x then the distance between x to any of the perceptrons associated with μ^+ is at least θ. Thus we now define *threshold perceptron* as a ternary function. Let (w, b) define a perceptron such that $\|w\| = 1$. The distance between x to the decision boundary of (w, b) is $|w \cdot x + b|$. We define:

$$h_{(w,b)}(x) = \begin{cases} 1 & \text{if } w \cdot x + b > \theta \\ -1 & \text{if } w \cdot x + b < -\theta \\ \star & \text{otherwise} \end{cases}$$

We now turn to prove theorem 2:

Proof: We may assume that all the perceptrons in LVQ are threshold perceptrons since we are interested in the θ margin shattering.

Next we ask, how many different outputs can different threshold perceptrons give on a sample of size m. Let $x_1, \ldots, x_m$ be a sample of size m. We say that $h_{(w,b)}$ and $h_{(\hat{w},\hat{b})}$ are the same on this sample if for any x_i such that $h_{(w,b)}, h_{(\hat{w},\hat{b})} \neq \star$ we have that $h_{(w,b)} = h_{(\hat{w},\hat{b})}$ i.e. on all the points which both classifiers have margin greater then θ, they agree on the label. We say that $h_{(w,b)}$ is maximal if it gives the minimal number of $\star$ from all the threshold-perceptrons which outputs the same labels.

We are interested in the number of different label sequences such perceptrons can assign to $x_1, \ldots, x_m$. Therefore we may assume that all the perceptrons are maximal perceptrons since the output level of LVQ ignores perceptrons with margin less them θ.

Counting the number of maximal labellings of $x_1, \ldots, x_m$ is done by repeating Sauer's lemma [14]. Define d to be the maximal size of sample that can be shattered by threshold classifiers, i.e. for which the number of different labellings is 2^d. Note that in this case such labels can not include any $\star$. From [2] we know that for threshold perceptrons $d \leq \min\left(\frac{R^2}{\theta^2}, n+1\right)$ and by repeating Sauer's lemma we get that the number of maximal labellings is bounded by $\left(\frac{em}{d}\right)^d$. Taking into account all the k^2 perceptrons, they can not generate more than then $\left(\frac{em}{d}\right)^{dk^2}$ outputs. Choosing $m \geq 2dk^2 \log_2 ek^2$ we get $\left(\frac{em}{d}\right)^{dk^2} < 2^m$ and hence a sample of such size can not be shattered. □

The next theorem shows that if LVQ has small θ-error then it has small generalization error with high probability. The proof follows the double sample technique introduced in [15]. This method has already been used with margins in [16]. We repeat the argument for the sake of completeness. Note that this argument works for any margin classifier. Before introducing the theorem we will define covering number of hypothesis class.

Definition 2 *Let $S = \{x_1, \ldots, x_m\}$ be a sample. Let H be an hypothesis class and $h \in H$. Assume that there is a margin defined, either hypothesis margin or sample margin. For a choice of $y_i \in \mathcal{Y}$ let $\theta_i(h)$ be the signed margin of h with respect to (x_i, y_i).*

We define the distance τ between $h_1, h_2 \in H$ with respect the sample to be

$$\tau_S(h_1, h_2) = \max_{i=1}^{m} \max_{y_i \in \mathcal{Y}} |\theta_i(h_1) - \theta_i(h_2)|$$

$N(H, \epsilon, (x_1, \ldots, x_m))$ is the minimal number of hypothesis $h_1, \ldots, h_N \in H$ such that for any $h \in H$ there exists h_i such that $\tau(h, h_i) < \epsilon$

$$N(H, \epsilon, m) = \max_{x_1, \ldots, x_m} N(H, \epsilon, (x_1, \ldots, x_m))$$

Note that in the case that we are working with hypothesis margin, we already have a distance measure ρ on the hypothesis class. This new distance measure τ is different then ρ, however it is easy to verify that $\tau_S(h_1, h_2) \leq \rho(h_1, h_2)$ for all choices of h_1, h_2 and S.

Theorem 3 *Let H be an hypothesis class. Let D be a distribution on the sample space X. Let $\varepsilon_D(h)$ be the generalization error of h. Given a sample S_1 of size $m \geq 16/\epsilon^2$. We have that:*

$$Pr_{S_1 \sim D^m}\left[\exists h \in H : \varepsilon_D(h) > \epsilon + \varepsilon^{\theta}_{S_1}(h)\right] \leq$$

$$2N\left(H, \frac{\theta}{2}, 2m\right) e^{-m\frac{\epsilon^2}{8}}$$

Proof: We define the event A as follows:

$$A = \left\{S_1 \in X^m \ : \ \exists h \, \varepsilon_D(h) > \epsilon + \varepsilon^{\theta}_{S_1}(h)\right\}$$

Then our goal is to bound $Pr_{S_1 \sim D^m}[A]$. We introduce the event B

$$B = \{S_1, S_2 : \exists h \, \varepsilon_D(h) > \epsilon + \varepsilon^{\theta}_{S_1}(h)$$

$$\text{and } \varepsilon_D(h) \leq \frac{\epsilon}{2} + \varepsilon_{S_2}(h)\}$$

From Chernoff's bound $Pr_{S_1, S_2 \sim D^{2m}}[B|A] \geq 1 - e^{-m\epsilon^2/8}$ and for $m \geq \frac{16}{\epsilon^2}$ we have that $Pr[B|A] \geq 0.5$.

Since $B \subseteq A$ we have that $Pr[B] = Pr[B, A] = Pr[B|A]Pr[A] \geq 0.5Pr[A]$ and therefore $Pr[A] \leq 2Pr[B]$. From now on we focus on bounding the probability of the event B.

Let S be a sample of size $2m$. Let S_1 and S_2 be a split of S into two parts of size m each. Assume that $(S_1, S_2) \in B$ so there exists $h \in H$ which demonstrates the event B. Let $\hat{h}$ be $\theta/2$ close to h. It follows that $\varepsilon_{S_1}^{\theta/2}(\hat{h}) < \varepsilon_D(h) - \epsilon$ and $\varepsilon_{S_2}^{\theta/2}(\hat{h}) \geq \varepsilon_D(h) - \epsilon/2$.

We define a new probability space. We split $S = \{(x_i, y_i)\}_{i=1}^{2m}$ into S_1 and S_2 according to the following rule: for each $1 \leq i \leq m$ with probability $1/2$ we have that $x_i \in S_1$ and $x_{m+i} \in S_2$, also with probability $1/2$ $x_{m+i} \in S_1$ and $x_i \in S_2$.

We define a random variable z_i for $1 \leq i \leq m$ such that $z_i = +1$ if $\hat{h}$ makes $\theta/2$-error on only one of x_i and x_{i+m} and this mistake is in S_2. If there is only one error among x_i and x_{i+m} and this mistake is in S_1 we assign the value -1 to z_i, in any other case $z_i = 0$.

A few facts about the random variables z_i:

1. For any i it is clear that $E[z_i] = 0$ and therefore $E\left[\sum z_i\right] = 0$.
2. $\sum z_i$ counts the difference between the number of $\theta/2$-errors $\hat{h}$ makes on S_2 and the number of such mistakes on S_1.
3. $\varepsilon_{S_1}^{\theta/2}(\hat{h}) \leq \varepsilon_D(h) - \epsilon$ and $\varepsilon_{S_2}^{\theta/2}(\hat{h}) \geq \varepsilon_D(h) - \epsilon/2$ only if $\sum z_i \geq m\epsilon/2$.

Since the z_i's are independent bounded random variables applying Chernoff's bound again reveals that the probability that $\hat{h}$ will have a big gap between it's error on S_2 and S_1 is bounded by $e^{-m\epsilon^2/8}$

Using the union bound on the elements of a $\theta/2$ covering set of H with respect to $x_1, \ldots, x_{2m}$ we have that the probability that any element in the covering set will have a gap between the number of $\theta/2$ errors it makes on S_2 and on S_1 which is greater then $m\epsilon/4$ is bounded by $N(H, \theta/2, 2m)e^{-m\epsilon^2/8}$. But this also bounds the probability that the event B happens over the splits of S into S_1 and S_2.

Since this is true for all the choices of S it is also true if we randomly select S according to the distribution D^{2m}. And so we conclude that $Pr_{S_1,S_2}[B] \leq N(H, \theta/2, 2m)e^{-m\epsilon^2/8}$ and since $Pr_{S_1}[A] \leq 2Pr_{S_2}[B]$ the statement follows. □

Combining theorems 2 and 3 we get a bound on the generalization of the LVQ algorithm.

Theorem 4 *Assume that the instances $\{x_i, y_i\}_{i=1}^m$ for training were drawn by some underlying distribution $\mathcal{D}$, furthermore assume that $x_i \leq R$ for every i. Let α be the proportion of the training data for which the decision rule induced by the prototypes yields a decision with margin smaller then θ or a mistake for some $0 < \theta < 1/2$. Let $e_{\mathcal{D}}$ be the generalization error of the chosen prototypes with respect to the density $\mathcal{D}$.*

For every $\delta > 0$, with probability $1 - \delta$ over the choices of the training data:

$$e_{\mathcal{D}} \leq \alpha + \sqrt{\frac{8}{m}\left(d \log^2 \frac{32m}{\theta^2} + \log \frac{4}{\delta}\right)} \tag{5}$$

where d is the VC dimension:

$$d = \min\left(n + 1, \frac{64R^2}{\theta^2}\right) 2k^{|\mathcal{Y}|} \log ek^2 \tag{6}$$

Proof: In [13] it is proved that $N(F, \gamma, m) \leq 2\left(\frac{4m}{\gamma^2}\right)^{d \log(2em/(d\gamma))}$ where d is the fat $\gamma/4$ dimension of F for a class of function that it's output is in the range $[a, a+1]$. Since $\theta \leq 0.5$ we can cut all signed-margins we use in the range $[0, 1]$. The statement now follows from theorems 2 and 3. □