ESANN'2009 proceedings, European Symposium on Artificial Neural Networks - Advances in Computational Intelligence and Learning. Bruges (Belgium), 22-24 April 2009, d-side publi., ISBN 2-930307-09-9.

Hyperparameter Learning in Robust Soft LVQ

Petra Schneider[1] and Michael Biehl[1] and Barbara Hammer[2]

1- University of Groningen - Institute of Mathematics and Computing Science
P.O. Box 407, 9700 AK Groningen - The Netherlands

2- Clausthal University of Technology - Institute of Computer Science
Julius Albert Strasse 4, 38678 Clausthal-Zellerfeld - Germany

Abstract. We present a technique to extend Robust Soft Learning Vector Quantization (RSLVQ). This algorithm is derived from an explicit cost function and follows the dynamics of a stochastic gradient ascent. The RSLVQ cost function involves a hyperparameter which is kept fixed during training. We propose to adapt the hyperparameter based on the gradient information. Experiments on artificial and real life data show that the hyperparameter crucially influences the performance of RSLVQ. However, it is not possible to estimate the best value from the data prior to learning. We show that the proposed variant of RSLVQ is very robust with respect to the choice of the hyperparameter.

1 Introduction

Learning Vector Quantization (LVQ) constitutes a family of supervised learning algorithms which are widely used for the classification of potentially high-dimensional data. The basic approach is very intuitive: classification is based on a set of so-called prototype vectors representing the classes, and a new feature vector is assigned to the class represented by the closest prototype. Since the basic LVQ scheme was introduced in 1986 [1], a large variety of modifications has been proposed; see e.g. [1, 2, 3]. The methods are very attractive due to their computational simplicity and flexibility. The algorithms are easy to implement, they can naturally deal with multi-class problems and the prototype vectors allow for an immediate interpretation of the resulting classifier.

The learning rules of several LVQ procedures involve a hyperparameter, such as the window size in LVQ2.1 [1] or the softness parameter σ^2 in Soft LVQ [3] and RSLVQ [3]. Usually, the hyperparameter is kept constant in the learning process, and it is chosen by means of a validation procedure. In [4], an annealing schedule is proposed to reduce the respective hyperparameter of an LVQ algorithm in the course of training. However, this schedule is purely heuristically motivated and does not follow any learning objective.

In this paper, we focus on RSLVQ and introduce a well-founded strategy to deal with the algorithms' hyperparameter σ^2. RSLVQ is derived from an explicit cost function which is optimized with respect to the model parameters using the method of steepest ascent. Since the cost function also contains σ^2, we propose to introduce σ^2 as a further degree of freedom and to maximize the objective function with respect the hyperparameter based on the gradient information.

In our experiments, we demonstrate the influence of the hyperparameter on

ESANN'2009 proceedings, European Symposium on Artificial Neural Networks - Advances in Computational Intelligence and Learning. Bruges (Belgium), 22-24 April 2009, d-side publi., ISBN 2-930307-09-9.

the classification accuracy of RSLVQ and study the effect of the proposed optimization method using artificial data and a real-life data set.

2 Review of Robust Soft LVQ

Assume training data $\{\boldsymbol{\xi}_k, y_k\}_{k=1}^{l} \in \mathbb{R}^N \times \{1, \ldots, C\}$ are given, N denoting the data dimensionality and C the number of different classes. An LVQ network $W = \{(\boldsymbol{w}_j, c(\boldsymbol{w}_j)) : \mathbb{R}^N \times \{1, \ldots, C\}\}_{j=1}^{m}$ consists of a number of prototypes $\boldsymbol{w} \in \mathbb{R}^N$ which are characterized by their location in feature space and their class label $c(\boldsymbol{w}) \in \{1 \ldots, C\}$. Given a distance measure $d(\boldsymbol{\xi}, \boldsymbol{w})$ in $\mathbb{R}^N$, classification is based on a winner-takes-all scheme: a data point $\boldsymbol{\xi} \in \mathbb{R}^N$ is assigned to the label $c(\boldsymbol{\xi}) = c(\boldsymbol{w}_i)$ of the prototype i with $d(\boldsymbol{w}_i, \boldsymbol{\xi}) \leq d(\boldsymbol{w}_j, \boldsymbol{\xi}), \forall j \neq i$. Many LVQ variants use the squared Euclidean distance $d(\boldsymbol{\xi}, \boldsymbol{w}) = (\boldsymbol{\xi} - \boldsymbol{w})^T(\boldsymbol{\xi} - \boldsymbol{w})$ to quantify the similarity between feature vectors and prototypes.

The Robust Soft LVQ - algorithm [3] to train the prototype locations is based on a statistical modeling of the given data distribution, i.e. the probability density is described by a mixture model. It is assumed that every component j of the mixture generates data which belongs to only one of the C classes. The probability density of the data is approximated by

$$p(\boldsymbol{\xi}|W) = \sum_{i=1}^{C} \sum_{j:c(\boldsymbol{w}_j)=i} P(j)\, p(\boldsymbol{\xi}|j), \tag{1}$$

where $\sum_j P(j) = 1$, and the conditional density $p(\boldsymbol{\xi}|j)$ is a function of the prototype $\boldsymbol{w}_j$. A possible choice is the normalized exponential form $p(\boldsymbol{\xi}|j) = K(j) \cdot \exp f(\boldsymbol{\xi}, \boldsymbol{w}_j, \sigma_j^2)$. In [3] a Gaussian mixture is assumed with $K(j) = (2\pi\sigma_j^2)^{-N/2}$ and $f(\boldsymbol{\xi}, \boldsymbol{w}_j, \sigma_j^2) = -d(\boldsymbol{\xi}, \boldsymbol{w}_j)/2\sigma_j^2$; where d is the squared Euclidean distance, and every component is assumed to have equal variance $\sigma_j^2 = \sigma^2$ and equal prior probability $P(j) = 1/m$, $\forall j$. RSLVQ maximizes the likelihood ratio

$$L = \prod_{k=1}^{l} \frac{p(\boldsymbol{\xi}_k, y_k|W)}{p(\boldsymbol{\xi}_k|W)} \tag{2}$$

with respect to the prototype locations by means of gradient ascent. $p(\boldsymbol{\xi}, y|W)$ is the probability density that sample $\boldsymbol{\xi}$ is generated by a component of the correct class y. This local density corresponds to the inner sum in Eq. 1. The learning rule is obtained by taking the derivatives of the RSLVQ cost function $E = \log(L)$ with respect to $\boldsymbol{w}_j$ (see [3])

$$\Delta\boldsymbol{w}_j = \frac{\alpha_1}{\sigma^2} \begin{cases} (P_y(j|\boldsymbol{\xi}) - P(j|\boldsymbol{\xi}))(\boldsymbol{\xi} - \boldsymbol{w}_j), & c(\boldsymbol{w}_j) = y, \\ -P(j|\boldsymbol{\xi})(\boldsymbol{\xi} - \boldsymbol{w}_j), & c(\boldsymbol{w}_j) \neq y, \end{cases} \tag{3}$$

where $\alpha_1 > 0$ is the learning rate, and $P_y(j|\boldsymbol{\xi})$ and $P(j|\boldsymbol{\xi})$ are assignment probabilities

$$P_y(j|\boldsymbol{\xi}) = \frac{\exp f(\boldsymbol{\xi}, \boldsymbol{w}_j, \sigma^2)}{\sum_{i:c(\boldsymbol{w}_i)=y} \exp f(\boldsymbol{\xi}, \boldsymbol{w}_i, \sigma^2)}, \qquad P(j|\boldsymbol{\xi}) = \frac{\exp f(\boldsymbol{\xi}, \boldsymbol{w}_j, \sigma^2)}{\sum_i \exp f(\boldsymbol{\xi}, \boldsymbol{w}_i, \sigma^2)} \tag{4}$$

ESANN'2009 proceedings, European Symposium on Artificial Neural Networks - Advances in Computational Intelligence and Learning. Bruges (Belgium), 22-24 April 2009, d-side publi., ISBN 2-930307-09-9.

with respect to one example $(\boldsymbol{\xi}, y)$. The update rules reflect the fact that prototypes with $c(\boldsymbol{w}) = y$ are attracted by the training sample, while prototypes carrying any other class label are repelled.

The performance of the algorithm highly depends on the hyperparameter σ^2: since it determines the value of the assignment probabilities (see Eq. 4), it controls the strength of the attractive and repulsive forces in Eq. 3. In the limit $\sigma^2 \to 0$, RSLVQ reduces to a learning-from-mistakes scheme, i.e. only in case of erroneous classification, the closest correct and incorrect prototype are updated. In the soft version of the algorithm, all training samples lying in an active region around the decision boundary cause an update of the prototype constellation; at the same time, a larger number of prototypes is adapted in each learning step (see [3] for details).

3 Hyperparameter adaptation in RSLVQ

In [4], a heuristic approach is introduced to anneal the value of the hyperparameter in the course of training. The authors propose a schedule which reduces σ^2 continuously in each learning step. This may lead to non-monotonic learning curves, as the performance deteriorates when σ^2 becomes lower than the potential optimum. Hence, the method has to be used in combination with an early stopping procedure.

In this work, we propose a more systematic approach to treat the hyperparameter according to the optimization of the likelihood ratio in Eq. 2. We adapt the hyperparameter according to the gradient of E with respect to σ^2. The derivative in [5] in combination with

$$\frac{\partial f(\boldsymbol{\xi}, \boldsymbol{w}, \sigma^2)}{\partial \sigma^2} = \frac{(\boldsymbol{\xi} - \boldsymbol{w})^T (\boldsymbol{\xi} - \boldsymbol{w})}{2\,\sigma^4}$$

leads us to the update rule

$$\Delta\sigma^2 = \alpha_2 \sum_j \left(\left(\delta_{y,c(\boldsymbol{w}_j)} \left(P_y(j|\boldsymbol{\xi}) - P(j|\boldsymbol{\xi}) \right) - \left(1 - \delta_{y,c(\boldsymbol{w}_j)}\right) P(j|\boldsymbol{\xi}) \right) \frac{d(\boldsymbol{\xi}, \boldsymbol{w}_j)}{\sigma^4} \right).$$

The Kronecker symbol $\delta_{y,c(\boldsymbol{w}_j)}$ tests whether the labels $c(\boldsymbol{w}_j)$ and y coincide, and $\alpha_2 > 0$ is the learning rate. The method becomes even more flexible by training an individual hyperparameter σ_j^2 for every prototype $\boldsymbol{w}_j$. Due to the derivative in [5], in combination with

$$\frac{\partial f(\boldsymbol{\xi}, \boldsymbol{w}, \sigma_j^2)}{\partial \sigma_j^2} = \frac{(\boldsymbol{\xi} - \boldsymbol{w})^T (\boldsymbol{\xi} - \boldsymbol{w})}{2\,\sigma_j^4}, \qquad \frac{\partial K(j)}{\partial \sigma_j^2} = -\frac{N}{2} \frac{1}{(2\pi\sigma_j^2)^{N/2}\,\sigma_j^2}$$

we obtain the learning rule

$$\Delta\sigma_j^2 = \frac{\alpha_2}{\sigma_j^2} \cdot \begin{cases} \left(P_y(j|\boldsymbol{\xi}) - P(j|\boldsymbol{\xi})\right)\left(-N + d(\boldsymbol{\xi}, \boldsymbol{w}_j)/\sigma_j^2\right), & c(\boldsymbol{w}_j) = y, \\ -P(j|\boldsymbol{\xi})\left(-N + d(\boldsymbol{\xi}, \boldsymbol{w}_j)/\sigma_j^2\right), & c(\boldsymbol{w}_j) \neq y. \end{cases}$$

Using this approach, the update rules for the prototypes in Eq. (3) also include the local hyperparameters σ_j^2.

ESANN'2009 proceedings, European Symposium on Artificial Neural Networks - Advances in Computational Intelligence and Learning. Bruges (Belgium), 22-24 April 2009, d-side publi., ISBN 2-930307-09-9.

4 Experiments

In the following experiments, we investigate the training procedure of σ^2 in the context of several learning scenarios. Artificial data sets are used to study the relation between the optimal hyperparameter and the variance of the training data. To analyse the influence of hyperparameter optimization on the classification performance of RSLVQ, the method is applied to the Letter data set from the UCI repository of machine learning [6]. All findings presented in the following are averaged over 10 cross-validation repetitions.

Artificial data sets: The data sets constitute binary classification problems and consist of two Gaussian clusters in a two-dimensional space. The clusters are centered on $\mu_1 = [-2, 0]$, $\mu_2 = [2, 0]$ and consist of 1000 data points, respectively. To analyse the adaptation of a global σ^2, we generate clusters with equal variance φ^2 varying between 0.5 and 2.5. At first, we fix the prototypes to the mean values of the distributions to analyse the adaptation of σ^2 independent of the other parameters of the system. In a second set of experiments, the hyperparameter and the prototypes are optimized simultaneously. The initial learning parameters are set to $\alpha_1 = 0.01$ and $\alpha_2 = 0.001$; we use the same learning rate schedule as in [5] with $c = 0.001$ and set the hyperparameter initially to $\sigma^2(0) = 1$. We choose the mean values of random subsets of training samples from each class to initialize the prototypes and train the system for 1000 epochs.

Fig. 1 (left) visualizes the mean final values of the hyperparameter obtained on the different data sets as a function of the clusters' variance. If the prototypes are constant and are placed in the cluster centers, σ^2 approaches the variance of the Gaussians. However, σ^2 converges towards smaller values in the experiments with adaptive prototypes. This result is also confirmed by further experiments with two spherical clusters of different variance $\varphi^2_{1,2}$ and local adaptive hyperparameter (see Fig. 1, right). Hence, maximizing the likelihood ratio in Eq. 2 corresponds to a density estimation, only if the prototypes correspond to the cluster means. However, the optimal hyperparameter cannot be estimated from the data directly, if the classifier is also optimized with respect to the prototype positions. This holds because of three reasons: for multi-modal data sets which are trained using several prototypes per class, the assignment of data to prototypes is not clear a priori, such that no statistical estimations can be made. Even for data sets which are represented using only one prototype per class, an estimation of σ^2 from the data is not obvious since, besides the bandwidth, σ^2 determines the influence of training points on the adaptation and hence the overall dynamics. Further, prototypes do not necessarily coincide with the class centers, rather, prototype locations and bandwidth are adapted by the learning rule to give an optimum decision boundary.

Letter data set: The data set consists of 20 000 feature vectors which encode 16 numerical attributes of black-and-white rectangular pixel displays of the 26 capital letters of the English alphabet. We split the data randomly into a training and a test set of equal size and compare the test performance of RSLVQ with constant and adaptive hyperparameter for different initial settings $\sigma^2(0)$. We choose various values $\sigma^2(0)$ from the interval [0.5, 3.5]. The initial learning

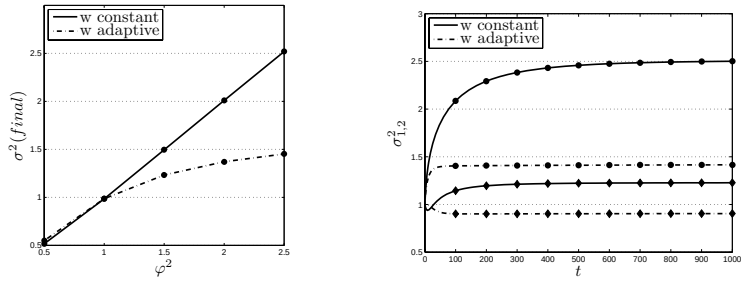

Fig. 1: **Left:** Variance φ^2 of the Gaussians vs. mean final value of the global hyperparameters σ^2 obtained on data sets with clusters of equal variance and constant and adaptive prototypes. **Right:** Evolution of the local hyperparameters $\sigma^2_{1,2}$ as a function of training time observed on a data set of Gaussians with unequal variance and constant and adaptive prototypes. The variances are $\varphi^2_1 = 2.5$ and $\varphi^2_2 = 1.25$. The symbols correspond to σ^2_1 (•), σ^2_2 (⧫).

parameters are set to $\alpha_1 = 0.01$, and $\alpha_2 = 0.001 \cdot \sigma^2(0)$; the learning rates are annealed according to the same schedule as in the previous experiment with $c = 0.1$. We approximate each class with one prototype respectively and train the system for 100 epochs.

Our experiments with constant σ^2 show that the performance of RSLVQ is very sensitive with respect to the value of hyperparameter (see Fig. 2, left). The lowest mean rate of misclassification on the test sets is achieved with $\sigma^2_{opt} = 1.25$; the performance constitutes $\varepsilon_{test} \approx 23.1\%$. However, the curve in Fig. 2 shows a very sharp minimum, indicating a strong dependence of the classification performance on the value of the hyperparameter. For small $\sigma^2 < 1$, we observe instabilities and highly fluctuating learning curves.

Remarkably, by including the proposed optimization scheme into the training, the sensitivity of the algorithm with respect to σ^2 can be eliminated. In all experiments with adaptive hyperparameter, the mean test error saturates at $\varepsilon_{test} \approx 23.4\%$, independent of the initial setting $\sigma^2(0)$ (see Fig. 2, left). Furthermore, the initialization $\sigma^2(0)$ does not influence the final value of the hyperparameter. As depicted in Fig. 2, right, the parameter converges towards $\sigma^2(final) \approx 1.8$ in all experiments. Hence, the proposed variant of RSLVQ is much more robust related to the initial choice of the hyperparameter. Especially for large values $\sigma^2(0)$, the proposed optimization method achieves a clear improvement in classification performance and speed of convergence, compared to RSLVQ training with constant σ^2. However, despite the extended flexibility, learning with constant $\sigma^2 = \sigma^2_{opt}$ still achieves a slightly better performance than our method. This observation can be explained by the fact that the relation between the RSLVQ cost function and the classification performance is not obvious. The optimum of the likelihood ratio does not necessarily coincide with minimal rate of misclassification. Nevertheless, the learning strategy for σ^2 may simplify the identification of σ^2_{opt} to achieve the optimal performance.

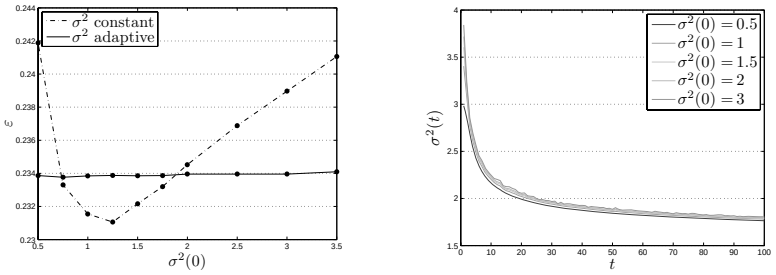

Fig. 2: **Left:** Mean test performance at the end of RSLVQ-training with constant and adaptive hyperparameter as a function of the initial value $\sigma^2(0)$. Standard error bars would be smaller than the symbol size. **Right:** Evolution of the hyperparameter σ^2 as a function of training time for different initial settings $\sigma^2(0)$.

5 Conclusion

In this study, we introduced hyperparameter learning in Robust Soft Learning Vector Quantization. The classification accuracy of RSLVQ is highly sensitive with respect to the correct choice of σ^2. We proposed to adapt σ^2 according to the optimization of the likelihood ratio which takes the influence of the hyperparameter on the RSLVQ cost function into account. As demonstrated in our experiments, this approach makes the algorithm very robust with respect to the hyperparameter and renders any trial and error search for an appropriate value unnecessary.

A restriction of standard RSLVQ consists in the use of the Euclidean distance measure. In [5], the algorithm is extended with respect to an adaptive metric structure. We are currently working on combining metric adaptation and hyperparameter adaptation in RSLVQ, showing first promising results.

References

[1] T. Kohonen. *Self-Organizing Maps*. Springer, Berlin, Heidelberg, second edition, 1997.

[2] A. Sato and K. Yamada. Generalized Learning Vector Quantization. In M. C. Mozer D. S. Touretzky and M. E. Hasselmo, editors, *Advances in Neural Information Processing Systems 8. Proceedings of the 1995 Conference*, pages 423–9, Cambridge, MA, USA, 1996. MIT Press.

[3] Sambu Seo and Klaus Obermayer. Soft Learning Vector Quantization. *Neural Computation*, 15(7):1589–1604, 2003.

[4] Sambu Seo and Klaus Obermayer. Dynamic hyper parameter scaling method for LVQ algorithms. In *International Joint Conference on Neural Networks*, Vancouver, Canada, 2006.

[5] Petra Schneider, Michael Biehl, and Barbara Hammer. Distance learning in discriminative vector quantization. *Neural Computation*, 2009, in press.

[6] D. J. Newman, S. Hettich, C. L. Blake, and C. J. Merz. UCI repository of machine learning databases. `http://archive.ics.uci.edu/ml/`, 1998.