Multivariate class labeling in Robust Soft LVQ

Petra Schneider[1], Tina Geweniger[2], Frank-Michael Schleif[3],
Michael Biehl[4] and Thomas Villmann[2]

1- School of Clinical and Experimental Medicine - University of Birmingham
Birmingham B15 2TT - United Kingdom

2- Department of MPI - University of Applied Sciences Mittweida
Technikumplatz 17 - 09648 Mittweida - Germany

3- CITEC - University of Bielefeld
Universitätsstrasse 21-23 - 33615 Bielefeld - Germany

4- Institute for Mathematics and Computer Science - University of Groningen
P.O. Box 407 - 4700 AK Groningen - The Netherlands

Abstract. We introduce a generalization of Robust Soft Learning Vector Quantization (RSLVQ). This algorithm for nearest prototype classification is derived from an explicit cost function and follows the dynamics of a stochastic gradient ascent. We generalize the RSLVQ cost function with respect to vectorial class labelings. This approach allows to realize multivariate class memberships for prototypes and training samples, and the prototype labels can be learned from the data during training. We present experiments to demonstrate the new algorithm in practice.

1 Introduction

Learning Vector Quantization (LVQ) is a popular family of machine learning algorithms for nearest prototype classification. LVQ was introduced by Kohonen in 1986 [1] and numerous modifications of the original algorithm have been proposed, see e.g. [2, 3, 4]. The popularity of the approach is due to the simplicity and interpretability of the resulting model. Furthermore, LVQ can easily cope with missing values in the data and multi-class problems can be treated in a straight-forward way. Numerous successful applications of LVQ are summarized in [5].

The basic LVQ approach relies on the major assumption that training data and prototypes are uniquely associated with a specific class. This does not allow to represent uncertainty or possibility in the data which is a common phenomenon, e.g. in biological or medical applications.

Extending Robust Soft Learning Vector Quantization (RSLVQ, [3]), we present a novel LVQ variant which allows to realize multivariate class memberships for training data and prototypes. RSLVQ is based on a cost function which is defined in terms of a likelihood ratio. We derive the generalization of the RSLVQ cost function with respect to vectorial class labels. In particular, we suppose that each element of the label vectors is in the range $[0, 1]$ describing the probabilistic assignment of the data and the prototypes to a certain class. Training consists in the optimization of the extended cost function with respect to the prototype locations, the prototype labels and the algorithm's hyperparameter (see [3]). In the working phase, classification is based on the value of the likelihood ratio, since this approach directly follows the objective of learning. As will be demonstrated in the experiments, adaptive prototype labeling additionally simplifies the model selection for LVQ systems.

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 $\mathbf{W} = \{(\boldsymbol{w}_j, c(\boldsymbol{w}_j)) : \mathbb{R}^n \times \{1, \ldots, C\}\}_{j=1}^{m}$ consists of a number of n-dimensional prototypes $\boldsymbol{w}_j$ which are characterized by their location in feature space and their class label $c(\boldsymbol{w}_j) \in \{1 \ldots, C\}$. 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 prototype $\boldsymbol{w}_i$ with $d(\boldsymbol{\xi}, \boldsymbol{w}_i) \leq d(\boldsymbol{\xi}, \boldsymbol{w}_j), \forall j \neq i$. 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 the prototypes.

The Robust Soft LVQ - algorithm [3] to train the prototype locations optimizes a cost function to adapt the prototypes to the data. The cost function 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

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

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 $f(\boldsymbol{\xi}, \boldsymbol{w}_j, \sigma_j^2) = -d(\boldsymbol{\xi}, \boldsymbol{w}_j)/2\sigma_j^2$; d is the squared Euclidean distance, and every component is assumed to have equal variance $\sigma_j^2 = \sigma^2$ and equal prior probability $P(j) = 1/m$, $\forall j$. Accordingly, class-specific densities correspond to

$$p(\boldsymbol{\xi}, y|\mathbf{W}) = \sum_{j:c(\boldsymbol{w}_j)=y} P(j)\, p(\boldsymbol{\xi}|j). \tag{2}$$

RSLVQ maximizes the likelihood ratio

$$L = \prod_{k=1}^{l} L(\boldsymbol{\xi}_k, y_k), \text{with } L(\boldsymbol{\xi}_k, y_k) = \frac{p(\boldsymbol{\xi}_k, y_k|\mathbf{W})}{p(\boldsymbol{\xi}_k|\mathbf{W})}. \tag{3}$$

The RSLVQ cost function is defined as $E_{\text{RSLVQ}} = \log(L)$ and it is maximized with respect to the prototype locations by means of stochastic gradient ascent. The learning rule is derived in [3]:

$$\Delta\boldsymbol{w}_j = \frac{\alpha_1}{\sigma^2} \begin{cases} (P_y(j|\boldsymbol{\xi}) - P(j|\boldsymbol{\xi}))(\boldsymbol{\xi} - \boldsymbol{w}_j), & c(\boldsymbol{w}_j) = y, \\ -P(j|\boldsymbol{\xi})(\boldsymbol{\xi} - \boldsymbol{w}_j), & c(\boldsymbol{w}_j) \neq y, \end{cases} \tag{4}$$

where α_1 is the learning rate, and $P_y(j|\boldsymbol{\xi})$, $P(j|\boldsymbol{\xi})$ are assignment probabilities

$$P_y(j|\boldsymbol{\xi}) = \frac{\exp f(\boldsymbol{\xi}, \boldsymbol{w}_j, \sigma^2)}{\sum_{i:c(\boldsymbol{w}_i)=y} \exp f(\boldsymbol{\xi}, \boldsymbol{w}_i, \sigma^2)}, \quad P(j|\boldsymbol{\xi}) \quad = \frac{\exp f(\boldsymbol{\xi}, \boldsymbol{w}_j, \sigma^2)}{\sum_i \exp f(\boldsymbol{\xi}, \boldsymbol{w}_i, \sigma^2)} \tag{5}$$

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

The training dynamics of RSLVQ highly depend on the hyperparameter σ^2. Since it controls the assignment probabilities in Eq. (5), it controls the strength

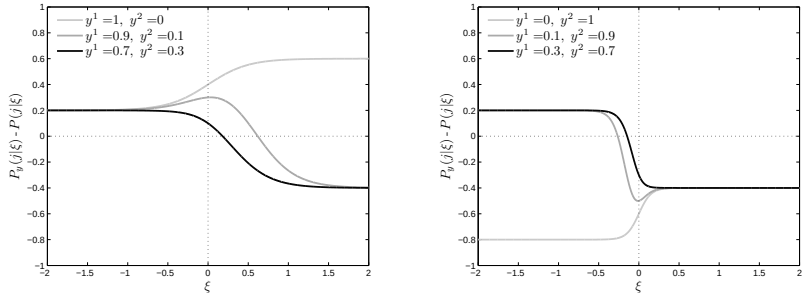

Fig. 1: Illustration of the update strength affecting prototype $\boldsymbol{w}_1$ in a one-dimensional setting with prototypes $\boldsymbol{w}_1 = -1, \boldsymbol{c}_1 = [0.8,\ 0.2]$ and $\boldsymbol{w}_2 = 1, \boldsymbol{c}_2 = [0.4,\ 0.6]$. The curves correspond to training samples with different label $\boldsymbol{y}$. Left: $y^1 > 0.5$, $\sigma^2 = 0.5$. Right: $y^1 < 0.5$, $\sigma^2 = 0.15$.

of the attractive and repulsive forces in Eq. (4). In the limit $\sigma^2 \to 0$, RSLVQ reduces to a learning-from-mistakes scheme (see [3] for details).

3 Generalization of RSLVQ

In the following, we introduce the generalization of the RSLVQ cost function with respect to vectorial class labels for the prototypes and the input data. We provide the learning rules for the prototype vectors, the prototype labels and the hyperparameter σ^2 derived as a stochastic gradient ascent of the generalized cost function. In the limit of crisp class memberships, the cost function and the updates are equivalent to the original RSLVQ algorithm.

Generalized cost function: In the generalized version of RSLVQ, input patterns and prototypes carry probabilistic label vectors $\boldsymbol{y}, \boldsymbol{c} \in [0,1]^C$, with $\sum_i y^i = 1$ and $\sum_i c^i = 1$. Component y^i constitutes the assignment probability of sample $\boldsymbol{\xi}$ to class i; the same holds for the prototypes respectively. Under this assumption, the definition of $\mathbf{W}$ (see Sec. 2) changes to

$$\mathbf{W} = \left\{ (\boldsymbol{w}_j, \boldsymbol{c}_j) \,|\, \mathbb{R}^n \times [0,1]^C, \text{with} \sum_i c_j^i = 1 \right\}_{j=1}^{m} . \tag{6}$$

Accordingly, the global data density is defined by

$$p(\boldsymbol{\xi}|\mathbf{W}) = \sum_{k=1}^{C} \sum_{j=1}^{m} c_j^k \, P(j) \, p(\boldsymbol{\xi}|j). \tag{7}$$

The generalization of the class-specific density in Eq. (2) constitutes the weighted sum of the class-specific densities; the weight values are given by the assignment probabilities of the input sample to the different classes

$$p(\boldsymbol{\xi}, \boldsymbol{y}|\mathbf{W}) = \sum_{k=1}^{C} y^k \sum_{j=1}^{m} c_j^k \, P(j) \, p(\boldsymbol{\xi}|j). \tag{8}$$

The cost function of the generalized version of RSLVQ arises out of E_{RSLVQ} by defining the likelihood ratio (Eq. (3)) in terms of $p(\boldsymbol{\xi}|\mathbf{W})$ and $p(\boldsymbol{\xi}, \boldsymbol{y}|\mathbf{W})$ as defined in Eq.s (7), (8). We term the new algorithm Probabilistic LVQ (PLVQ).

Learning rules: In this contribution, we restrict our analysis to the adaptation of a global hyperparameter $\sigma^2 = \sigma_j^2, \forall j$. To be in accordance with [3], we choose $f(\boldsymbol{\xi}, \boldsymbol{w}_j, \sigma^2)$ and $K(j)$ corresponding to a Gaussian mixture model and use the squared Euclidean distance. Details of the derivatives $\partial E_{\mathrm{PLVQ}}/\partial\{\boldsymbol{w}, \boldsymbol{\xi}, \sigma^2\}$ will be presented elsewhere. The derivatives yield the following update rules for the prototypes and the hyperparameter

$$\Delta \boldsymbol{w}_j = \frac{\alpha_1}{\sigma^2} P_j(\boldsymbol{\xi}, \boldsymbol{y}) (\boldsymbol{\xi} - \boldsymbol{w}_j), \qquad \Delta\sigma^2 = \frac{\alpha_2}{\sigma^2} P_j(\boldsymbol{\xi}, \boldsymbol{y}) \frac{d(\boldsymbol{\xi}, \boldsymbol{w}_j)}{\sigma^2}. \tag{9}$$

with $P_j(\boldsymbol{\xi}, \boldsymbol{y}) = \big(P_{\boldsymbol{y}}(j|\boldsymbol{\xi}) - P(j|\boldsymbol{\xi})\big)$, and $P_{\boldsymbol{y}}(j|\boldsymbol{\xi})$ is the generalized class-specific assignment probability

$$P_{\boldsymbol{y}}(j|\boldsymbol{\xi}) = \frac{\sum_k y^k c_j^k P(j) K(j) \exp f(\boldsymbol{\xi}, \boldsymbol{w}_j, \sigma_j^2)}{p(\boldsymbol{\xi}, \boldsymbol{y}|\mathbf{W})}. \tag{10}$$

The learning rule for the prototype labels results in

$$\Delta \boldsymbol{c}_j = \alpha_3 \left(\frac{\boldsymbol{y}}{p(\boldsymbol{\xi}, \boldsymbol{y}|\mathbf{W})} - \frac{1}{p(\boldsymbol{\xi}|\mathbf{W})} \right) P(j) p(\boldsymbol{\xi}|j). \tag{11}$$

The Probabilistic LVQ algorithm is defined in terms of Eqs. (9) and (11), and $\alpha_{1,2,3}$ are the learning rates.

Note that $P_j(\boldsymbol{\xi}, \boldsymbol{y})$ turns into the weight factors in Eq. (4), if $\boldsymbol{y}$ and $\boldsymbol{c}_j$ specify crisp class memberships. Fig. 1 demonstrates the influence of $\boldsymbol{y}, \boldsymbol{c}_j$ and the hyperparameter σ^2 on $P_j(\xi, \boldsymbol{y})$ in a one-dimensional example setting. In this scenario, all training samples with non-crisp label cause attractive forces on $\boldsymbol{w}_1$ for $\xi \to -\infty (P_j(\xi, \boldsymbol{y}) > 0)$ and have repulsive effects for $\xi \to +\infty$, $(P_j(\xi, \boldsymbol{y}) < 0)$. Contrary, training samples with crisp labeling have either attractive or repulsive effects all over feature space. The extreme values of the weight factors are determined by the prototype label $\boldsymbol{c}_1$ and $\boldsymbol{c}_2$; the hyperparameter σ^2 controls the transition between attractive and repulsive effects.

Classification: The original LVQ approach of distance-based classification is not applicable in PLVQ, since the prototypes represent multiple classes. We propose to use the value of the likelihood ratio defining the PLVQ cost function as classification criterion for the new algorithm: An unknown feature vector $\boldsymbol{\xi}$ is assigned to class i with $L(\boldsymbol{\xi}, i) > L(\boldsymbol{\xi}, j), \forall j \neq i$, where $L(\boldsymbol{\xi}, i) = \frac{p(\boldsymbol{\xi}, \boldsymbol{y}_i|\mathbf{W})}{p(\boldsymbol{\xi}|\mathbf{W})}$ and the label vector $\boldsymbol{y}_i$ specifies the unique assignment to class i. Highest likelihood-ratio classification directly follows the objective of learning and has already shown promising results in RSLVQ [6].

4 Experiments

We apply PLVQ to the artificial data visualized in Fig. 2. The data set consists of three classes in a two dimensional space. Each class consists of 60 samples which are separated in two clusters. Note that we consider the special case of training data with crisp class memberships in this application. We compare the

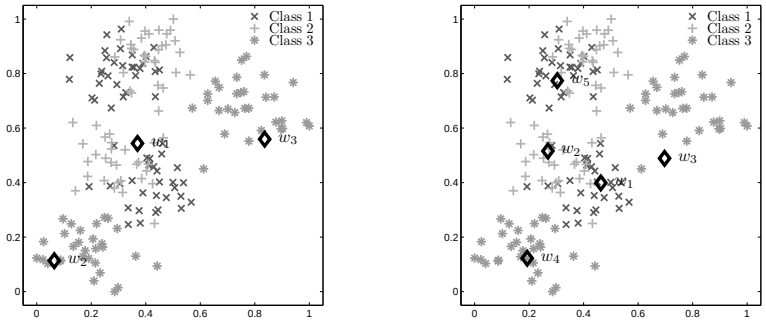

Fig. 2: Artificial data set. The figures show the prototype locations after PLVQ training with three and five prototypes obtained in one training run. The corresponding prototype label are stated in Tab. 1.

performance of RSLVQ and PLVQ using different numbers of prototypes in combination with an adaptive hyperparameter σ^2. We use the learning parameter settings $\alpha_{1,3} = 5 \cdot 10^{-4}$ and $\alpha_2 = \sigma^2(0) \cdot 10^{-5}$ with $\sigma^2(0) = 0.02$. We randomly select 80% of the samples of each class for training and use the remaining data for testing. We perform the experiment on 25 random constellations of training and test set and train the system for 200 epochs in each run. Prototypes are initialized close the global mean value of the training data. Initial prototype labels in PLVQ are generated randomly, followed by a normalization step.

The mean error rates on the test sets are stated in Tab. 1, right. Due to the nature of the data set, RSLVQ fails, if the data is approximated by only one prototype per class. Remarkably, PLVQ already performs significantly better with the same number of prototypes. Fig. 2 (left) exemplary shows the final prototype constellations obtained in one experiment; the corresponding prototype labels are stated in Tab. 1, left. Obviously, prototype $\boldsymbol{w}_1$ represents classes one and two equally. Note that increasing the number of prototypes in PLVQ does not require the specification of the prototypes' class memberships before training is initialized. RSLVQ with two prototypes per class outperforms PLVQ

Table 1: Artificial data set. Left: Prototype label after PLVQ-training with three and five prototypes obtained in one training run. The corresponding prototype locations are displayed in Fig. 2. Right: Mean test errors of RSLVQ and PLVQ using different numbers of prototypes.

	3 Prototypes	5 Prototypes
$\boldsymbol{c}_1$	[0.5, 0.5, 0]	[0.13, 0.87, 0]
$\boldsymbol{c}_2$	[0.0, 0.0, 1.0]	[0.0, 0.0, 1.0]
$\boldsymbol{c}_3$	[0.0, 0.0, 1.0]	[0.84, 0.16, 0.0]
$\boldsymbol{c}_4$		[0.0, 0.0, 1.0]
$\boldsymbol{c}_5$		[0.5, 0.5, 0.0]

#Prototypes	RSLVQ	PLVQ
3	0.5	0.35
4		0.31
5		0.28
6	0.18	0.26

with six prototypes. Note however that for this result prior information about the structure of the data was taken into account to select the model settings. Another showcase of a PLVQ system consisting of five prototypes is given in Fig. 2, right, and Tab. 1, left. It verifies that PLVQ detects prototypes with shared class memberships, if not all clusters can be approximated by individual prototypes.

First experiments on high-dimensional real life data including probabilistic labeling of training data showed promising results with local adaptive hyperparameters σ_j^2. The adaptation of a global σ^2 in PLVQ turned out to be insufficient in more complex classification scenarios. Detailed experiments and investigations of this issue will be presented in forthcoming publications.

5 Conclusion

In this contribution, we introduced a novel LVQ variant which allows to incorporate probabilistic label information into the training process of the model parameters. Probabilistic Learning Vector Quantization generalizes the successful Robust Soft LVQ such that vectorial labeling for prototypes and training samples can be processed. In particular, the approach allows to realize adaptive class labels for the prototypes. It was demonstrated in experiments that this ability is especially beneficial, if no prior information about the structure of the data is available.

Two extensions of PLVQ will be subject to future work: As outlined in Sec. 4, we will investigate PLVQ with local, adaptive hyperparameters in real life applications. Furthermore, a serious restriction of the algorithm consists in the use of the Euclidean distance measure. Metric adaptation techniques have shown great potential to improve the performance of RSLVQ [4], and comparable influence can be expected for PLVQ.

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] P. Schneider, M. Biehl, and B. Hammer. Distance learning in discriminative vector quantization. *Neural Computation*, 21(10):2942–2969, 2009.

[5] *Bibliography on the Self-Organizing Map (SOM) and Learning Vector Quantization (LVQ)*. Neural Networks Research Centre, Helskinki University of Technology, 2002.

[6] P. Schneider, M. Biehl, and B. Hammer. Hyperparameter learning in probabilisitc prototype-based models. *Neurocomputing*, 73(7-9):1117–1124, 2010.