Prototype Learning with Margin-Based Conditional Log-likelihood Loss

Xiaobo Jin Cheng-Lin Liu Xinwen Hou
National Laboratory of Pattern Recognition, Institute of Automation, Chinese Academy of Sciences
95 Zhongguancun East Road, Beijing 100190, P. R. China
{xbjin,liucl,xwhou}@nlpr.ia.ac.cn

Abstract

The classification performance of nearest prototype classifiers largely relies on the prototype learning algorithms, such as the learning vector quantization (LVQ) and the minimum classification error (MCE). This paper proposes a new prototype learning algorithm based on the minimization of a conditional log-likelihood loss (CLL), called log-likelihood of margin (LOGM). A regularization term is added to avoid over-fitting in training. The CLL loss in LOGM is a convex function of margin, and so, gives better convergence than the MCE algorithm. Our empirical study on a large suite of benchmark datasets demonstrates that the proposed algorithm yields higher accuracies than the MCE, the generalized LVQ (GLVQ), and the soft nearest prototype classifier (SNPC).

1. Introduction

Nearest neighbor classification is a simple and appealing non-parametric method for pattern classification. It does not need any preprocessing of the training data, but suffers from heavy burden of storage space and computation in classification. Prototype selection and prototype learning methods are efficient to reduce the number of prototypes and meanwhile improve the classification accuracy of nearest neighbor classifier (also called nearest prototype classifier).

The well-known Learning Vector Quantization (LVQ) algorithm of Kohonen [4] offers intuitive and simple, yet powerful prototype learning rules in supervised learning. The LVQ algorithm yields superior classification performance although it does not guarantee convergence in training [7]. Crammer et al. [2] show that LVQ falls in a family of maximal margin learning algorithms and they present a rigorous bound on the generalization error of LVQ .

Many other prototype learning algorithms based on loss minimization have been shown to give higher classification accuracies than LVQ [5]. These algorithms include the minimum classification error (MCE) method [3], the generalized LVQ (GLVQ) [7], the maximum class probability (MAXP) method [5], the soft nearest prototype classifier (SNPC) [8], etc. The MCE and GLVQ methods minimize margin-based loss functions, while the MAXP and SNPC formulate multiple prototypes in a Gaussian mixture framework and aim to minimize the Bayesian classification error on training data. The margin-based algorithms, including LVQ, MCE and GLVQ, only adjust two prototypes during one update. The training complexity of the MAXP and SNPC methods can be similarly decreased by pruning prototype updating according to the proximity of prototypes to the input pattern or the windowing rule.

This paper proposes a new margin-based prototype learning algorithm, called log-likelihood of margin (LOGM). This algorithm is derived from the conditional log-likelihood loss (CLL), which is a convex function of the margin. The convex loss function leads to better convergence performance. A regularization term is added to the loss function for avoiding overfitting in training. The experimental results show that the LOGM algorithm gives higher test accuracies than the MCE, GLVQ and SNPC methods in most cases. Section 2 describes the framework of prototype learning algorithm as well as the MCE algorithm. Then Section 3 presents the LOGM algorithm and its regularization. Section 4 details some experiments and the conclusions are given in Section 5.

2. Prototype Learning Algorithm

Let us consider a labeled data set $D = \{(\mathbf{x}_n, y_n)|n = 1, 2, \cdots, N\}$, where $\mathbf{x}_n \in \mathcal{R}^d$ and $y_n \in \{1, 2, \cdots, L\}$ (L is the number of classes). The prototype learning algorithms design a set of prototypes $\{\mathbf{u}_{ls}|s = 1, 2, \cdots, S\}$ for each class l (S can be different for different classes). The test pattern $\mathbf{x}$

978-1-4244-2175-6/08/$25.00 ©2008 IEEE

is classified to the class of the closest prototype. The choice of distance measure plays a crucial role in the algorithm performance but it is not on our focus. We only consider the Euclidean distance for simplicity. The expected risk based on loss function ϕ is defined as $R_\phi f = \int \phi(f(\mathbf{x})|\mathbf{x})p(\mathbf{x})d\mathbf{x}$ where $\phi(f(\mathbf{x})|\mathbf{x})$ is the loss that $\mathbf{x}$ is classified by the decision $f(\mathbf{x})$, and $p(\mathbf{x})$ is the probability density at the sample point $\mathbf{x}$. In practice, the empirical average loss on finite samples is computed by $\hat{R}_\phi f = \frac{1}{N}\sum_{n=1}^{N}\phi(f(\mathbf{x}_n)|\mathbf{x}_n)$. Generally, the loss function depends on the set of the prototype vectors $\Theta = \{\mathbf{u}_{ls}|l = 1, 2, \cdots, L; s = 1, 2, \cdots, S\}$. At the stationary point, the loss function $\hat{R}_\phi$ satisfies $\nabla_\Theta \hat{R}_\phi = 0$. The prototypes are updated on each sample by gradient descent:

$$\Theta(t+1) = \Theta(t) - \eta(t)\nabla\phi(f(\mathbf{x})|\mathbf{x})|_{\mathbf{x}=\mathbf{x}_n}, \quad (1)$$

where $\eta(t)$ is the learning rate of the algorithm in the t-th iteration.

In MCE algorithm, $\phi(f)$ is represented as the misclassification rate. When $\mathbf{x}$ with the genuine class l is classified, the misclassification rate is $1 - P(C_l|\mathbf{x})$. MCE algorithm aims to find the optimal parameters to minimize the misclassification rate on the training set by approximating the posterior probability $P(C_l|\mathbf{x})$.

The discriminant function for $\mathbf{x}_n$ (k is the genuine class label) is given by:

$$g_k(\mathbf{x}_n) = \max_s\{-\|\mathbf{x}_n - \mathbf{u}_{ks}\|^2\} = -\|\mathbf{x}_n - \mathbf{u}_{ki}\|^2, \quad (2)$$

where $\|\cdot\|$ is the Euclidean metric.

The misclassification measure [3] can be defined as follow:

$$d_k(\mathbf{x}_n) = -g_k(\mathbf{x}_n) + g_r(\mathbf{x}_n), \quad (3)$$

where $g_r(\mathbf{x}_n) = \max_{l\neq k} g_l(\mathbf{x}_n) = -\|\mathbf{x}_n - \mathbf{u}_{rj}\|^2$. We may view $-d_k(\mathbf{x}_n)$ as the margin of $\mathbf{x}_n$ which is similar to the hypothesis margin [2]. The misclassification loss of $\mathbf{x}_n$ is approximated by a the sigmoid function $\sigma(x)$:

$$\phi(\mathbf{x}_n) = 1 - P(C_k|\mathbf{x}_n) = 1 - \frac{1}{1 + e^{\xi d_k(\mathbf{x}_n)}}, \quad (4)$$

where ξ ($\xi > 0$) is a constant.

It is easy to calculate the derivative of the loss as

$$\frac{\partial\phi(\mathbf{x}_n)}{\partial g_{ki}} = -\xi\phi(\mathbf{x}_n)(1-\phi(\mathbf{x}_n)), \frac{\partial\phi(\mathbf{x}_n)}{\partial g_{rj}} = -\frac{\partial\phi(\mathbf{x}_n)}{\partial g_{ki}}, \quad (5)$$

where $g_{ki} = -\|\mathbf{x}_n - \mathbf{u}_{ki}\|^2$ and $g_{rj} = -\|\mathbf{x}_n - \mathbf{u}_{rj}\|^2$.

When the training pattern $\mathbf{x_n}$ is given in the t-th iteration, two prototypes in class k and r are updated by:

$$\begin{aligned}\mathbf{u}_{ki} &= \mathbf{u}_{ki} - 2\eta(t)\frac{\partial\phi(\mathbf{x}_n)}{\partial g_{ki}}(\mathbf{x}_n - \mathbf{u}_{ki}) \\ \mathbf{u}_{rj} &= \mathbf{u}_{rj} - 2\eta(t)\frac{\partial\phi(\mathbf{x}_n)}{\partial g_{rj}}(\mathbf{x}_n - \mathbf{u}_{rj}). \quad (6)\end{aligned}$$

In GLVQ [7], $d_k(\mathbf{x}_n)$ is measured by the relative difference instead:

$$d_k(\mathbf{x}_n) = \frac{-g_k(\mathbf{x}_n) + g_r(\mathbf{x}_n)}{g_k(\mathbf{x}_n) + g_r(\mathbf{x}_n)}, \quad (7)$$

which results in a perfect converge performance.

3. Conditional Log-Likelihood Loss for Prototype Learning

3.1. LOGM Algorithm

Conditional Log-Likelihood Loss (CLL) is widely used in statistical pattern classification such as logistic regression. In general, the conditional likelihood function for multi-class is given by

$$P(T|\Theta) = \prod_{n=1}^{N}\prod_{l=1}^{L} P(C_l|\mathbf{x}_n)^{t_{nl}}, \quad (8)$$

where $t_{nl} = I[\mathbf{x}_n \in C_l]$ and $I[x]$ is the indicator. Taking the negative logarithm then gives

$$E = -\sum_{n=1}^{N}\sum_{l=1}^{L} t_{nl}\ln P(C_l|\mathbf{x}_n), \quad (9)$$

which is known as Conditional Log-Likelihood Loss. The loss associated with $\mathbf{x}_n$ is computed as:

$$\phi(\mathbf{x}_n) = -\sum_{l=1}^{L} t_{nl}\ln P(C_l|\mathbf{x}_n). \quad (10)$$

For a pattern $\mathbf{x}_n$, we may view the genuine class of $\mathbf{x}_n$ as the positive class and the remain classes as the negative class. Given that $\mathbf{u}_{ki}$ and $\mathbf{u}_{rj}$ are the closest prototype to $\mathbf{x}_n$ in the positive class and in the negative class, respectively, we can define the discriminant function for a binary classification problem:

$$f(\mathbf{x}_n) = g_k(\mathbf{x}_n) - g_r(\mathbf{x}_n) = -d_k(\mathbf{x}_n). \quad (11)$$

Like MCE, the posterior probability can be approximated by the sigmoid function:

$$P(C_k|\mathbf{x}_n) = \sigma(\xi y f(\mathbf{x}_n)) = \frac{1}{1 + e^{\xi d_k(\mathbf{x}_n)}}, \quad (12)$$

where $y = +1$ since the posterior probability estimation is about the positive class (the genuine class k). Obviously, $yf(\mathbf{x})$ is the margin of the pattern $\mathbf{x}$. If $yf(\mathbf{x}) < 0$, $\mathbf{x}$ is misclassified.

We can obtain the following derivative:

$$\frac{\partial\phi(\mathbf{x}_n)}{\partial g_{ki}} = -(1 - P(C_k|\mathbf{x}_n))\xi, \frac{\partial\phi(\mathbf{x}_n)}{\partial g_{rj}} = -\frac{\partial\phi(\mathbf{x}_n)}{\partial g_{ki}}. \quad (13)$$

Two prototypes are updated by (6) when the training pattern $\mathbf{x}_n$ with the genuine class k is given.

3.2. Regularization of Prototype Learning Algorithms

The minimization of misclassification loss may lead to unstable solutions. Unlike that the maximization of sample margin (for support vector machines, e.g.) is beneficial to the generalization performance, the enlarging of hypothesis margin may deteriorate the generalization performance.

To constrain the hypothesis margin of MCE and LOGM algorithms, we add a regularization term to the loss function in a similar way of [6]:

$$\tilde{\phi}(\mathbf{x}_n) = \phi(\mathbf{x}_n) + \alpha\|\mathbf{x}_n - \mathbf{u}_{ki}\|^2, \qquad (14)$$

where α is the regularization coefficient. Institutively, the regularizer makes the winning prototype move toward $\mathbf{x}_n$. The prototypes now are updated by

$$\begin{aligned} \mathbf{u}_{ki} &= \mathbf{u}_{ki} - 2\eta(t)(\frac{\partial\phi(\mathbf{x}_n)}{\partial g_{ki}} - \alpha)(\mathbf{x}_n - \mathbf{u}_{ki}) \\ \mathbf{u}_{rj} &= \mathbf{u}_{rj} - 2\eta(t)\frac{\partial\phi(\mathbf{x}_n)}{\partial g_{rj}}(\mathbf{x}_n - \mathbf{u}_{rj}). \end{aligned} \qquad (15)$$

For fair comparison, we also add a regularization term in the SNPC algorithm [8]:

$$\tilde{\phi}(\mathbf{x}_n) = \phi(\mathbf{x}_n) - \alpha\log(\sum_j \exp(\xi g_{kj})), \qquad (16)$$

where $\phi(\mathbf{x}_n) = 1 - P_k(\mathbf{x}_n)$ and $P_l(\mathbf{x}_n) = \sum_s P_{ls}(\mathbf{x}_n)$ given $P_{ls}(\mathbf{x}_n) = \frac{\exp(\xi g_{ls}(\mathbf{x}_n))}{\sum_{i,j}\exp(\xi g_{ij}(\mathbf{x}_n))}$. The second term is the negative log-likelihood of the class-conditional density about the positive class, which makes all prototypes in the positive class move toward $\mathbf{x}_n$ with the ratio $\frac{P_{ls}(\mathbf{x}_n)}{P_k(\mathbf{x}_n)}$. Then the prototypes are updated by

$$\begin{aligned} \mathbf{u}_{ls} &= \mathbf{u}_{ls} - 2\eta(t)\frac{\phi(\mathbf{x}_n)}{g_{ls}}(\mathbf{x}_n - \mathbf{u}_{ls}) \\ &\quad +2\eta(t)\alpha\xi\frac{P_{ls}(\mathbf{x}_n)}{P_k(\mathbf{x}_n)}(\mathbf{x}_n - \mathbf{u}_{ls}), l = k \\ \mathbf{u}_{ls} &= \mathbf{u}_{ls} - 2\eta(t)\frac{\phi(\mathbf{x}_n)}{g_{ls}}(\mathbf{x}_n - \mathbf{u}_{ls}), l \neq k, \end{aligned} \qquad (17)$$

where $s = 1, 2, \cdots, S$ and the derivative of the loss function is as follow:

$$\frac{\partial\phi(\mathbf{x}_n)}{\partial g_{ls}} = \begin{cases} -\xi P_{ls}(\mathbf{x}_n)(1 - P_k(\mathbf{x}_n)), & l = k; \\ \xi P_{ls}(\mathbf{x}_n)P_k(\mathbf{x}_n), & l \neq k. \end{cases} \qquad (18)$$

3.3. Discussions

The margin-based prototype learning algorithms aim to minimize different loss functions based on certain margin. MCE and LOGM optimize the loss

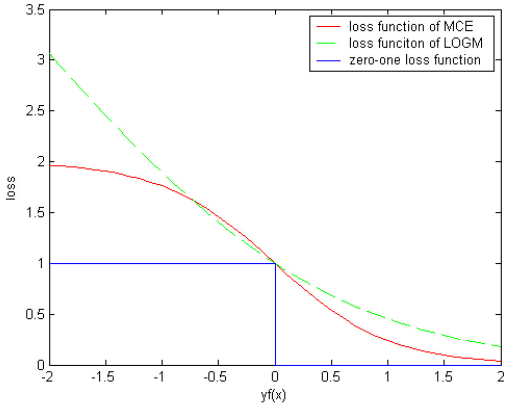

Figure 1. The loss of MCE and LOGM algorithm based on the margin $yf(\mathbf{x})$

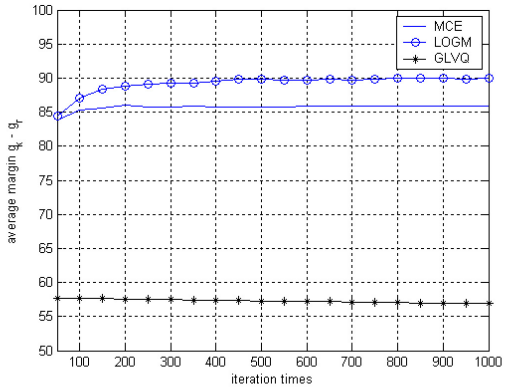

Figure 2. Average margins of different algorithms during training on the USPS dataset with $S = 20$ and $\alpha = 0.001$

$\frac{2}{1+\exp(2yf(\mathbf{x}))}$ and $\frac{\ln(1+\exp(-yf(\mathbf{x})))}{\ln 2}$ [1], respectively. The loss functions are plotted in Figure 1 where $yf(\mathbf{x}) = g_k(\mathbf{x}) - g_r(\mathbf{x})$. The property of classification-calibration [1] about two losses guarantees that the minimization of $R_\phi(f)$ may provide a reasonable surrogate for the minimization of 0-1 loss. The strictness of convexity about the loss of LOGM avoids running into the local optimal point and obtains more optimal $\hat{R}_\phi(f^*)$ risk. Figure 2 demonstrates average margins of different algorithms on the USPS dataset during learning. We see that after 100 iterations LOGM can obtain larger average margin than MCE and GLVQ on the regularization condition.

4. Experiments

We compare the performance of the proposed LOGM with other three representative prototype learning algorithms (MCE, SNPC and GLVQ) on 10 small

[1] Two loss functions are scaled to pass through the point $(0, 1)$

UCI benchmark data-sets [2] and 2 large datasets which include USPS and 20NG (20newgroups) listed in Table 1. Each example of USPS is a 256-dimensional vector with entries between 0 and 255. The 20NG dataset is a collection of approximately 20,000 documents that were collected from 20 different newsgroups [3].The **by-date** version of this data set along with its train/test split was used in our experiments for the convenience of comparisons. The information gain method was used for selecting 1,000 highest score features.

In implementing the algorithms, the prototypes were initialized by K-means clustering of classwise data. The initial learning rate $\eta(0)$ was set to $0.1\tau * cov$, where τ was drawn from $\{0.1, 0.5, 1, 1.5, 2\}$ and cov is the average distance of all training patterns to the nearest cluster center. In the t-th iteration, the learning rate of the n-th pattern was $\eta(0)(1 - \frac{tN+n}{TN})$, where T is the maximum iteration time. For 10 small UCI datasets, T was set to 100 and the prototype number S was set to $\{1, 2, 3, 4, 5\}$. For USPS and 20NG, T was set to 40 and S was set to $\{5, 10, 15, 20, 25\}$ following [4]. The regularization coefficient α was set to $\{0, 0.001, 0.005, 0.01, 0.05\}$. The training parameters and model parameters could be optimized in the space of the cubic grid (α, S, τ) by cross-validation on the training data. All experiments ran on the framework of Torch [4], a C++ library.

In 10 small data-sets, the accuracy of an algorithm on a data set was obtained via 10 runs of ten-fold cross validation. Runs of the various algorithms were carried out on the same training sets and evaluated on the same test sets. About 1/3 of the training data-set were used for the validation of hyper-parameters. But for USPS and 20NG, we only chose 1/5 of the training examples for validation. Then all of the training data-set were retrained under the optimal hyper-parameters.

Table 1 gives the accuracies and the standard deviations of different prototype learning algorithms. For USPS and 20NG, the validation parameters (α, S, τ) are included in the brace. The positivity of α shows that the regularization technique benefits the performance of prototype learning. We see LOGM gives the highest accuracy on 7 of 12 data-sets. In the difficult datasets glass2 and vehicle, the advantage of LOGM is remarkable. We use sign-rank test to compare LOGM with other algorithms in the statistical significance. It is found that LOGM is more superior than MCE ($p = 0.1099$), SNPC ($p = 0.1099$) and GLVQ ($p = 0.0640$). The p-value shows the significance probability of no difference between two algorithms.

[2] http://www.ics.uci.edu/mlearn/MLRepository.html

[3] http://people.csail.mit.edu/jrennie/20Newsgroups/

[4] http://www.torch.ch/

Table 1. Classification results: Empirical accuracy of the algorithms in $[\%]$ with standard deviation.

dataset	#fea.	#obj.	MCE	SNPC	LOGM	GLVQ
diabetes	8	768	76.13(0.80)	75.60(1.68)	**76.80(0.64)**	75.37(1.15)
glass2	9	163	73.77(2.51)	72.63(1.96)	**74.41(2.95)**	72.26(3.69)
heart	13	270	82.37(1.56)	**83.22(0.84)**	81.18(1.55)	83.04(1.99)
iris	4	150	95.60(0.95)	95.73(0.95)	**96.20(0.89)**	95.53(0.89)
ionosphere	34	351	87.75(0.99)	87.47(0.68)	88.38(1.13)	**89.12(1.60)**
pima	8	768	75.94(0.59)	75.75(1.09)	**76.39(0.85)**	75.75(0.66)
sonar	60	208	86.74(1.37)	85.77(1.36)	**86.81(2.55)**	84.69(2.39)
vehicle	18	846	80.38(0.85)	77.02(1.07)	**81.95(0.66)**	78.25(1.27)
wdbc	30	569	97.35(0.37)	**97.44(0.37)**	97.22(0.29)	97.21(0.32)
wine	13	178	97.22(0.62)	**97.83(0.44)**	97.27(0.61)	96.44(0.64)
USPS	256	7,291	**94.62**	94.37	94.32	93.77
	#test	2,007	(0.005;20;1.5)	(0.001;25;2)	(0.005;10;1.5)	(0.001;5;1.5)
20NG	1,000	11,314	74.24	73.80	**74.84**	74.19
	#test	7,532	(0.01;5;0.5)	(0.01;20;0.5)	(0.001;5;0.5)	(0.005;5;0.1)

5. Conclusion and Further Research

We have proposed a prototype learning algorithm (LOGM) based on the minimization of conditional log-likelihood loss and discuss its relations with MCE algorithm. A regularization term is added to avoid overfitting in training. In experiments on a large suite of benchmark datasets, LOGM is demonstrated to outperform the previous methods MCE,GLVQ and SNPC.

6. Acknowledgements

This work is supported by the National Natural Science Foundation of China (NSFC) under grant no.60723005.

References

[1] P. L. Bartlett, M. I. Jordan, and J. D. McAuliffe. Convexity, classification, and risk bounds. *Journal of the American Statistical Association*, 101(473):138–156, 2006.

[2] K. Crammer, R. Gilad-Bachrach, A. Navot, and N. Tishby. Margin analysis of the lvq algorithm. In *NIPS*, pages 462–469, 2002.

[3] B.-H. Juang and S. Katagiri. Discriminative learning for minimum error classification. *IEEE Transactions on Signal Processing*, pages 3042–3054, 1992.

[4] T. Kohonen, J. Hynninen, J. Kangas, J. Laaksonen, and K. Torkkola. Lvq pak: The learning vector quantization program package. Technical report, Helsinki University of Technology, 1995.

[5] C.-L. Liu and M. Nakagawa. Evaluation of prototype learning algorithms for nearest-neighbor classifier in application to handwritten character recognition. *Pattern Recognition*, 34(3):601–615, 2001.

[6] C.-L. Liu, H. Sako, and H. Fujisawa. Discriminative learning quadratic discriminant function for handwriting recognition. *IEEE Transactions on Neural Networks*, 15(2):430–444, 2004.

[7] A. Sato and K. Yamada. Generalized learning vector quantization. In *NIPS*, pages 423–429, 1995.

[8] S. Seo and K. Obermayer. Soft learning vector quantization. *Neural Computation*, 15(7):1589–1604, 2003.