Pattern Recognition 43 (2010) 2428–2438

Contents lists available at ScienceDirect

Pattern Recognition

journal homepage: www.elsevier.com/locate/pr

Regularized margin-based conditional log-likelihood loss for prototype learning

Xiao-Bo 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

ARTICLE INFO

Article history:
Received 21 April 2008
Received in revised form
29 December 2009
Accepted 20 January 2010

Keywords:
Prototype learning
Conditional log-likelihood loss
Log-likelihood of margin (LOGM)
Regularization
Distance metric learning

ABSTRACT

The classification performance of nearest prototype classifiers largely relies on the prototype learning algorithm. The minimum classification error (MCE) method and the soft nearest prototype classifier (SNPC) method are two important algorithms using misclassification loss. This paper proposes a new prototype learning algorithm based on the conditional log-likelihood loss (CLL), which is based on the discriminative model called log-likelihood of margin (LOGM). A regularization term is added to avoid over-fitting in training as well as to maximize the hypothesis margin. The CLL in the LOGM algorithm is a convex function of margin, and so, shows better convergence than the MCE. In addition, we show the effects of distance metric learning with both prototype-dependent weighting and prototype-independent weighting. Our empirical study on the benchmark datasets demonstrates that the LOGM algorithm yields higher classification accuracies than the MCE, generalized learning vector quantization (GLVQ), soft nearest prototype classifier (SNPC) and the robust soft learning vector quantization (RSLVQ), and moreover, the LOGM with prototype-dependent weighting achieves comparable accuracies to the support vector machine (SVM) classifier.

1. Introduction

Nearest neighbor classification [1] is a simple and appealing non-parametric method for pattern classification. It does not need any processing of the training data, but involves heavy burden of storage space and computation in classification. Prototype learning is to represent the training data with a set of points in feature space, called prototypes. A test point **x** is then classified to the class of the closest prototype. To reduce the number of prototypes, many prototype selection and prototype learning methods have been proposed [2–7]. By selecting or synthesizing prototypes that better represent the class distributions or decision boundaries, these methods are also effective to improve the classification accuracy. The nearest neighbor classifiers with reduced prototypes are also called nearest prototype classifiers. They have been widely used in applications such as character recognition [7], text categorization [8], classification of mass spectrometry data [9], and so on.

Learning vector quantization (LVQ) [10] is a well known prototype learning algorithm which offers intuitive and simple, yet powerful learning capacity in supervised learning. Kohonen proposed a number of improved versions of LVQ such as LVQ2, LVQ2.1, and LVQ3 [3]. Crammer et al. [11] show that LVQ falls in a family of maximal margin learning algorithms providing a rigorous upper bound of generalization error. Although the LVQ algorithm yields superior classification performance, it does not guarantee convergence in training [4]. The performance of LVQ depends on several factors: the initialization of prototypes, the distance metric, and the selection of informative patterns, etc.

To improve the training and generalization performance of LVQ, many extensions or variants have been proposed. In [12], an initialization insensitive LVQ algorithm uses a harmonic average distance instead of the usual nearest-neighbor distance. Sato and Yamada proposed a generalized LVQ (GLVQ) algorithm [4], where prototypes are updated based on a continuous and differentiable loss function. The generalized relevance LVQ (GRLVQ) algorithm [13] introduces the adaptive weighted metric to extend the GLVQ, by adding a prototype-independent weight to each input dimension indicating its relevance. Similarly, the algorithm of Paredes and Vidal [14] learns prototypes and distance metric simultaneously using a misclassification loss function. In [15], the combination of pattern selection and weighted distance norm is explored. It selects an update set composed of points that are considered to be at the risk of being captured by a prototype of a different class.

Many prototype learning algorithms based on loss minimization have been shown to give higher classification accuracies than LVQ [7]. These algorithms include the minimum classification error (MCE) method [16], the generalized LVQ (GLVQ) [4], the

* Corresponding author. Tel.: +86 10 62632251.
E-mail addresses: xbjin@nlpr.ia.ac.cn (X.-B. Jin), liucl@nlpr.ia.ac.cn (C.-L. Liu), xwhou@nlpr.ia.ac.cn (X. Hou).

0031-3203/$ - see front matter Crown Copyright © 2010 Published by Elsevier Ltd. All rights reserved.
doi:10.1016/j.patcog.2010.01.013

maximum class probability (MAXP) method [7], the soft nearest prototype classifier (SNPC) [17] and the robust soft learning vector quantization (RSLVQ) [18], 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. Similarly, the RSLVQ optimizes the conditional log-likelihood loss (CLL) within a Gaussian mixture framework. The misclassification loss of [14] is similar to those of MCE and GLVQ. The margin-based algorithms, including LVQ, MCE and GLVQ, adjust only two prototypes on a training sample. The training complexity of the MAXP, SNPC and the RSLVQ method can be similarly decreased by pruning prototype updating according to the proximity of prototypes to the input pattern or using the windowing rule like the LVQ2.1.

In this paper, we investigate into the loss functions of prototype learning algorithms and aim to improve the classification performance using a new form of loss function. The misclassification loss functions of MCE, GLVQ and the one in [14], based on the nearest prototype from the genuine class (correct class) of input pattern and the one from incorrect classes, are smoothed functions of the traditional 0–1 loss, while the direct minimization of the 0–1 loss is computationally intractable [19]. The MAXP algorithm [7], the SNPC method [17] and the RSLVQ [18] approximate the misclassification rate in the framework of Gaussian mixture. We will discuss the relationships between these algorithms under certain conditions.

We propose a new prototype learning algorithm by minimizing a conditional log-likelihood loss (CLL), called log-likelihood of margin (LOGM). The convexity of margin-based loss in LOGM ensures that there exists a unique maximum margin. We also add a regularization term to the loss function for avoiding over-fitting in training as well as maximizing the hypothesis margin. The proposed method can be easily extended to a weighted distance metric instead of the Euclidian distance to incorporate the relevance of features. We have extended the LOGM algorithm for prototype learning with both prototype-dependent weighting and prototype-independent weighting. Our empirical study on a large suite of benchmark datasets demonstrates that the LOGM is superior to the MCE, GLVQ, SNPC and RSLVQ methods, and moreover, the LOGM with prototype-dependent weighting achieves comparable accuracies with the support vector machine (SVM) classifier.

The rest of this paper is organized as follows: Section 2 gives the formulation of prototype learning algorithms; Section 3 briefly reviews the MCE and SNPC methods; Section 4 presents a prototype learning algorithm (LOGM) based on the margin-based conditional log-likelihood loss and extends the LOGM to prototype learning with weighted distance metric; Section 5 discusses the relationships between our algorithm with previous ones and their convergence; Section 6 presents our experimental results and the last section concludes the paper. This paper is an extension to our previous conference paper [20] by adding the discussions of relationships with other methods, the analysis of learning convergence and generalization, and the extension to prototype learning with weighted distance metric. Throughout the paper, we use many acronyms, which are listed in Table 1 for readers' convenience.

2. Formulation of prototype learning

For L-class classification, prototype learning is to design a set of prototype vectors $\{\mathbf{m}_{ls}, s=1,2,\ldots,S\}$[1] for each class l. An input pattern $\mathbf{x}\in\mathcal{R}^d$ is classified to the class of the nearest prototype.

[1] S can vary with different classes. For simplicity, we consider the case of equal number of prototypes per class.

Table 1
List of acronyms.

Acronym	Full name
CLL	Conditional Log-likelihood Loss
GLVQ	Generalized Learning Vector Quantization
GRLVQ	Generalized Relevance LVQ
LOGM	LOG-likelihood of Margin
LVQ	Learning Vector Quantization
MAXP	MAXimum class Probability
MAXP1	A variant of MAXP
MCE	Minimum Classification Error
RBF	Radial Basis Function
RSLVQ	Robust Soft Learning Vector Quantization
RSLVQ1	A variant of RSLVQ
SNPC	Soft Nearest Prototype Classifier
SVM	Support Vector Machine

The choice of distance metric is influential to the classification performance, but we first focus on the loss function of the prototype learning under the Euclidean distance and will then explore the effect of the weighted distance metric.

For learning the prototypes, consider a labeled training dataset $D=\{(\mathbf{x}_n,y_n), n=1,2,\ldots,N\}$, where $y_n\in\{1,2,\ldots,L\}$ is the class label of the training pattern $\mathbf{x}_n$. The objective of learning is to minimize the expected risk based on loss function ϕ:

$$R_\phi f=\int \phi(f(\mathbf{x}))p(\mathbf{x})\,d\mathbf{x}, \tag{1}$$

where $\phi(f(\mathbf{x}))$ 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 expected loss is replaced by the empirical loss on a training set:

$$\hat{R}_\phi f=\frac{1}{N}\sum_{n=1}^{N}\phi(f(\mathbf{x}_n)). \tag{2}$$

Generally, the loss function depends on the parameters of prototypes $\Theta=\{\mathbf{m}_{ls}\}$, $l=1,2,\ldots,L$, $s=1,2,\ldots,S$. At the stationary point, the loss function $\hat{R}_\phi$ satisfies $\nabla_\Theta\hat{R}_\phi=0$.

For minimizing the empirical loss, the prototypes are updated on each training pattern by stochastic gradient descent [21]:

$$\Theta(t+1)=\Theta(t)-\eta(t)\nabla\phi(f(\mathbf{x}))|_{\mathbf{x}=\mathbf{x}_n}, \tag{3}$$

where $\eta(t)$ is the learning rate at the t-th iteration.

3. Prototype learning with misclassification loss

In the MCE and SNPC methods, $\phi(f)$ approximates the misclassification rate. When a pattern $\mathbf{x}$ from class l is classified, the misclassification rate is $1-P(C_l|\mathbf{x})$. The MCE method approximates the posterior probability $P(C_l|\mathbf{x})$ by the sigmoid function while the SNPC estimates the posterior probability in the framework of Gaussian mixture.

3.1. Minimum Classification Error (MCE) method

For a pattern $\mathbf{x}_n$ from genuine class k (class label), its discriminant function to a class l is given by

$$g_l(\mathbf{x}_n)=\max_s\{-\|\mathbf{x}_n-\mathbf{m}_{ls}\|^2\}, \tag{4}$$

where $\|\cdot\|$ is the Euclidean metric. We denote the nearest prototype from the genuine class as $\mathbf{m}_{ki}$, i.e., $g_k(\mathbf{x}_n)=-\|\mathbf{x}_n-\mathbf{m}_{ki}\|^2$.

A reasonable definition of misclassification measure is given by [16]

$$d_k(\mathbf{x}_n)=-g_k(\mathbf{x}_n)+g_r(\mathbf{x}_n), \tag{5}$$

where $g_r(\mathbf{x}_n)=\max_{l\neq k} g_l(\mathbf{x}_n)=-\|\mathbf{x}_n-\mathbf{m}_{rj}\|^2$ is the discriminant function of the closest rival class r. $-d_k(\mathbf{x}_n)$ can be viewed as the margin of $\mathbf{x}_n$, which is connected to the hypothesis margin [11].[2]

The misclassification loss of $\mathbf{x}_n$ is approximated by the sigmoid function $\sigma(\cdot)$:

$$\phi(\mathbf{x}_n)=1-P(C_k|\mathbf{x}_n)=1-\frac{1}{1+e^{\xi d_k(\mathbf{x}_n)}}=\frac{1}{1+e^{-\xi d_k(\mathbf{x}_n)}}=\sigma(\xi d_k(\mathbf{x}_n)), \tag{6}$$

where ξ ($\xi>0$) is a constant for tuning the smoothness of sigmoid. From (5) and (6), the loss on a training pattern $\mathbf{x}_n$ only depends on two nearest prototypes: $\mathbf{m}_{ki}$ from the genuine class and $\mathbf{m}_{rj}$ from one of incorrect classes.

It is easy to calculate the derivative of the loss with respective to prototypes as

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

$$\frac{\partial\phi(\mathbf{x}_n)}{\partial g_{rj}}=-\frac{\partial\phi(\mathbf{x}_n)}{\partial g_{ki}}. \tag{7}$$

In training by stochastic gradient descent, the prototypes are updated on each training pattern $\mathbf{x}_n$ at the t-th iteration by

$$\mathbf{m}_{ki}=\mathbf{m}_{ki}-2\eta(t)\frac{\partial\phi(\mathbf{x}_n)}{\partial g_{ki}}(\mathbf{x}_n-\mathbf{m}_{ki}),$$

$$\mathbf{m}_{rj}=\mathbf{m}_{rj}-2\eta(t)\frac{\partial\phi(\mathbf{x}_n)}{\partial g_{rj}}(\mathbf{x}_n-\mathbf{m}_{rj}). \tag{8}$$

The GLVQ (Generalized Learning Vector Quantization) algorithm [4] takes the same loss function as the MCE method but a different misclassification measure $d_k(\mathbf{x}_n)$ from the same two nearest prototypes:

$$d_k(\mathbf{x}_n)=\frac{-g_k(\mathbf{x}_n)+g_r(\mathbf{x}_n)}{g_k(\mathbf{x}_n)+g_r(\mathbf{x}_n)}. \tag{9}$$

It forces the quantity of update $|\Delta\mathbf{m}_{ki}|$ greater than $|\Delta\mathbf{m}_{rj}|$ during training. This was shown to lead to a perfect convergence of training [4].

3.2. Soft Nearest Prototype Classifier (SNPC)

By the SNPC method, the input pattern $\mathbf{x}_n$ is assigned posterior probabilities to prototypes $\mathbf{m}_{ls}$ with the Gaussian mixture assumption:

$$P_{ls}(\mathbf{x}_n)=\frac{\exp(\xi g_{ls}(\mathbf{x}_n))}{\sum_{i=1}^{L}\sum_{j=1}^{S}\exp(\xi g_{ij}(\mathbf{x}_n))}, \tag{10}$$

where $g_{ls}(\mathbf{x})=-\|\mathbf{x}-\mathbf{m}_{ls}\|^2$. The posterior probability of a class l is

$$P(C_l|\mathbf{x}_n)=\sum_{s=1}^{S}P_{ls}(\mathbf{x}_n)=P_l(\mathbf{x}_n). \tag{11}$$

Then, the misclassification loss for a pattern $\mathbf{x}_n$ from genuine class k is

$$\phi(\mathbf{x}_n)=1-P_k(\mathbf{x}_n). \tag{12}$$

Since the loss function of SNPC involves all the prototypes, in training by stochastic gradient descent, all the prototypes are updated on a training pattern:

$$\mathbf{m}_{ls}=\mathbf{m}_{ls}-2\eta(t)\frac{\partial\phi(\mathbf{x}_n)}{\partial g_{ls}}(\mathbf{x}_n-\mathbf{m}_{ls}),$$

$$l=1,2,\ldots,L,\quad s=1,2,\ldots,S, \tag{13}$$

where

$$\frac{\partial\phi(\mathbf{x}_n)}{\partial g_{ls}}=\begin{cases}-\xi P_{ls}(\mathbf{x}_n)(1-P_k(\mathbf{x}_n)), & l=k,\\ \xi P_{ls}(\mathbf{x}_n)P_k(\mathbf{x}_n), & l\neq k.\end{cases} \tag{14}$$

In training, the parameter ξ[3] is initially set to a small value and then increases after every iteration in spirit of deterministic annealing.

The idea of SNPC for prototype learning was proposed earlier as MAXP (MAXimum class Probability) from a different view [7], where the criterion of maximizing class probability was inspired by neural network training [22].

4. Prototype learning with conditional log-likelihood loss

The Conditional Log-likelihood Loss (CLL) is widely used in pattern classification methods such as logistic regression [21] and Bayesian networks [23]. In general, the conditional likelihood function for multi-class is given by

$$P(H|\Theta)=\prod_{n=1}^{N}\prod_{l=1}^{L}P(C_l|\mathbf{x}_n)^{t_{nl}}, \tag{15}$$

where H is an $N\times L$ matrix of target variables with elements $t_{nl}=I[\mathbf{x}_n\in C_l]$ and $I[\cdot]$ is the indicator. Taking the negative logarithm gives

$$E=-\sum_{n=1}^{N}\sum_{l=1}^{L}t_{nl}\ln P(C_l|\mathbf{x}_n), \tag{16}$$

which is known as conditional log-likelihood loss. The loss associated with a pattern $\mathbf{x}_n$ is

$$\phi(\mathbf{x}_n)=-\sum_{l=1}^{L}t_{nl}\ln P(C_l|\mathbf{x}_n). \tag{17}$$

The Robust Soft Learning Vector Quantization (RSLVQ) [18], similar to the MAXP (MAXimum class Probability) and the SNPC (Soft Nearest Prototype Classifier) methods, models the class density in the framework of Gaussian mixture and minimizes the conditional log-likelihood loss. In RSLVQ, the derivative of the loss (17) on a training pattern $\mathbf{x}_n$ from genuine class k is given by

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

Compared with the derivative of MAXP and SNPC in (14), we can see that the derivative of loss for RSLVQ can be obtained by dividing that of MAXP and SNPC by the posterior probability of the genuine class. In the following, we will propose a discriminative model-based prototype learning method, which minimizes a margin-based loss, called LOG-likelihood of Margin (LOGM).

4.1. LOG-likelihood of Margin (LOGM)

For multi-class classification, we may view the genuine class of $\mathbf{x}_n$ as the positive class and the union of the remaining classes as a negative class. Given that $\mathbf{m}_{ki}$ and $\mathbf{m}_{rj}$ are the nearest prototype to $\mathbf{x}_n$ from the positive class (class k) and the one from the negative class (class r), respectively, we can define the discriminant

[2] On the nearest genuine prototype $\mathbf{m}_{ki}$ and the nearest rival prototype $\mathbf{m}_{rj}$ to point $\mathbf{x}$, the hypothesis margin is $\frac{1}{2}(\|\mathbf{x}-\mathbf{m}_{rj}\|-\|\mathbf{x}-\mathbf{m}_{ki}\|)$.

[3] In the Gaussian distribution, ξ is proportional to the precision parameter which is the inverse of the covariance.

function for a binary classification problem (positive class and negative class):

$$f(\mathbf{x}_n) = g_k(\mathbf{x}_n) - g_r(\mathbf{x}_n) = -d_k(\mathbf{x}_n), \tag{19}$$

where $g_l(\mathbf{x}_n)$ (l=k,r) and $d_k(\mathbf{x}_n)$ are as in (4) and (5). If $f(\mathbf{x}_n) < 0$, $\mathbf{x}_n$ is misclassified. $f(\mathbf{x}_n)$ is similar to the hypothesis margin of pattern $\mathbf{x}_n$, which is defined as $\frac{1}{2}(\|\mathbf{x}-\mathbf{m}_{rj}\| - \|\mathbf{x}-\mathbf{m}_{ki}\|)$.

Like the MCE (Minimum Classification Error) method, the posterior probability of genuine class can be approximated by the sigmoid function:

$$P(C_k|\mathbf{x}_n) = \sigma(\xi f(\mathbf{x}_n)) = \frac{1}{1+e^{\xi d_k(\mathbf{x}_n)}}, \tag{20}$$

and the conditional log-likelihood loss is $\phi(\mathbf{x}_n) = -\log P(C_k|\mathbf{x}_n)$. The derivative of the loss (17) is then given by

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

$$\frac{\partial \phi(\mathbf{x}_n)}{\partial g_{rj}} = -\frac{\partial \phi(\mathbf{x}_n)}{\partial g_{ki}}. \tag{21}$$

On the training pattern $\mathbf{x}_n$ from genuine class k, only two prototypes, $\mathbf{m}_{ki}$ and $\mathbf{m}_{rj}$, are updated by gradient descent (8). Comparing the derivatives of (21) for LOGM with those of (7) for MCE, it is seen that the derivatives of LOGM can be obtained via dividing those of MCE by $P(C_k|\mathbf{x}_n)$.

4.2. Regularization of prototype learning algorithms

The minimization of misclassification loss may lead to unstable solutions. For example, when the samples of two classes are linearly separable, a classifier with one prototype per class can classify them perfectly. To minimize the loss function, however, the prototypes will be drawn away from the decision surface as far as possible for enlarging the margin. Unlike that the maximization of sample margin (for support vector machines, e.g.) is beneficial to generalization performance, the enlarging of the margin of MCE as in (19) may deteriorate the generalization performance.

To constrain the margin of the MCE (Minimum Classification Error) and the LOGM (LOG-likelihood of Margin) algorithms, we add a regularization term to the loss function in a similar way to [24]

$$\tilde{\phi}(\mathbf{x}_n) = \phi(\mathbf{x}_n) + \alpha\|\mathbf{x}_n - \mathbf{m}_{ki}\|^2, \tag{22}$$

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

$$\mathbf{m}_{ki} = \mathbf{m}_{ki} - 2\eta(t)\left(\frac{\partial \phi(\mathbf{x}_n)}{\partial g_{ki}} - \alpha\right)(\mathbf{x}_n - \mathbf{m}_{ki}),$$

$$\mathbf{m}_{rj} = \mathbf{m}_{rj} - 2\eta(t)\frac{\partial \phi(\mathbf{x}_n)}{\partial g_{rj}}(\mathbf{x}_n - \mathbf{m}_{rj}). \tag{23}$$

Obviously, the regularization term in (22) effects in reducing the distance to prototype $\|\mathbf{x}_n - \mathbf{m}_{ki}\|$. While the margin of MCE as in (19) equals

$$\|\mathbf{x}-\mathbf{m}_{rj}\|^2 - \|\mathbf{x}-\mathbf{m}_{ki}\|^2 = (\|\mathbf{x}-\mathbf{m}_{rj}\| + \|\mathbf{x}-\mathbf{m}_{ki}\|)(\|\mathbf{x}-\mathbf{m}_{rj}\| - \|\mathbf{x}-\mathbf{m}_{ki}\|), \tag{24}$$

when the margin of MCE is constrained, the reducing of distance to prototype helps enlarge the hypothesis margin (corresponding to the second term above).

For fair comparison, we also add a regularization term in the SNPC (Soft Nearest Prototype Classifier) and the RSLVQ (Robust Soft Learning Vector Quantization) algorithms:

$$\tilde{\phi}(\mathbf{x}_n) = \phi(\mathbf{x}_n) - \alpha\log\left(\sum_j \exp(\xi g_{kj})\right), \tag{25}$$

where the second term is the negative log-likelihood of the class-conditional density about the positive class and it makes all prototypes in the positive class move toward $\mathbf{x}_n$. The prototypes are then updated by

$$\mathbf{m}_{ls} = \mathbf{m}_{ls} - 2\eta(t)\frac{\phi(\mathbf{x}_n)}{g_{ls}}(\mathbf{x}_n - \mathbf{m}_{ls}) + 2\eta(t)\alpha\xi\frac{P_{ls}(\mathbf{x}_n)}{P_k(\mathbf{x}_n)}(\mathbf{x}_n - \mathbf{m}_{ls}), \quad l = k,$$

$$\mathbf{m}_{ls} = \mathbf{m}_{ls} - 2\eta(t)\frac{\phi(\mathbf{x}_n)}{g_{ls}}(\mathbf{x}_n - \mathbf{m}_{ls}), \quad l \neq k, \tag{26}$$

where $s = 1, 2, \ldots, S$. Obviously, the increment toward the prototype $\mathbf{m}_{ls}$ is proportional to the posterior probability $P_{ls}(\mathbf{x}_n)$.

4.3. LOGM with weighted distance metric

The performance of prototype classifier depends on the distance metric. The commonly used Euclidean metric, without considering the relevance of features, may not give sufficient classification performance. To justify the potential of improved performance of prototype learning with weighted distance metric, we extend the LOGM algorithm to prototype learning with both prototype-dependent weighting and prototype-independent weighting. Distance metric learning with prototype-independent feature weights has been applied with the GRLVQ [13]. We will show that using prototype-dependent weights, i.e., a weight vector for each prototype, can achieve a substantial improvement of classification performance.

In the case of prototype-dependent weighting, the square distance from a pattern $\mathbf{x}$ to a prototype $\mathbf{m}_{ls}$ is defined as[4]

$$r(\mathbf{x}, \mathbf{m}_{ls}; \boldsymbol{\lambda}_{ls}) = \sum_{p=1}^{d} \lambda_{ls}^p (x^p - m_{ls}^p)^2, \tag{27}$$

where $\boldsymbol{\lambda}_{ls}$, $l = 1, 2, \ldots, L$, $s = 1, 2, \ldots, S$, is the weight vector attached to the prototype $\mathbf{m}_{ls}$. Replacing $\|\mathbf{x}-\mathbf{m}_{ls}\|^2$ in LOGM with $r(\mathbf{x}, \mathbf{m}_{ls}; \boldsymbol{\lambda}_{ls})$, the updating of the prototypes and weights in stochastic gradient descent can be formulated as ($p = 1, \ldots, d$)

$$m_{ki}^p = m_{ki}^p - 2\eta(t)\left(\frac{\partial \phi(\mathbf{x}_n)}{\partial g_{ki}} - \alpha\right)\lambda_{ki}^p (x_n^p - m_{ki}^p),$$

$$m_{rj}^p = m_{rj}^p - 2\eta(t)\frac{\partial \phi(\mathbf{x}_n)}{\partial g_{rj}}\lambda_{rj}^p (x_n^p - m_{rj}^p), \tag{28}$$

$$\lambda_{ki}^p = \lambda_{ki}^p + \rho(t)\left(\frac{\partial \phi(\mathbf{x}_n)}{\partial g_{ki}} - \alpha\right)(x_n^p - m_{ki}^p)^2,$$

$$\lambda_{rj}^p = \lambda_{rj}^p - \rho(t)\frac{\partial \phi(\mathbf{x}_n)}{\partial g_{ki}}(x_n^p - m_{rj}^p)^2. \tag{29}$$

On another hand, the square distance metric with prototype-independent weights is defined as

$$r(\mathbf{x}, \mathbf{m}_{ls}; \boldsymbol{\lambda}) = \sum_{p=1}^{d} \lambda^p (x^p - m_{ls}^p)^2. \tag{30}$$

[4] The superscript p in the formula represent the p-th dimension of the vector.

Similarly, the updating of parameters in stochastic gradient descent can be induced as

$$m_{ki}^p = m_{ki}^p - 2\eta(t)\left(\frac{\partial\phi(\mathbf{x}_n)}{\partial g_{ki}} - \alpha\right)\lambda^p(x_n^p - m_{ki}^p),$$

$$m_{rj}^p = m_{rj}^p - 2\eta(t)\frac{\partial\phi(\mathbf{x}_n)}{\partial g_{rj}}\lambda^p(x_n^p - m_{rj}^p), \tag{31}$$

$$\lambda^p = \lambda^p + \rho(t)\left[\left(\frac{\partial\phi(\mathbf{x}_n)}{\partial g_{ki}} - \alpha\right)(x_n^p - m_{ki}^p)^2 - \frac{\partial\phi(\mathbf{x}_n)}{\partial g_{ki}}(x_n^p - m_{rj}^p)^2\right]. \tag{32}$$

In the weight vector $\boldsymbol{\lambda}_{ls}$ or $\boldsymbol{\lambda}$, all the components are initially set to 1/d. After each update in stochastic gradient descent, if a component of the weight vector is negative, it is re-set to zero. The weight vector is renormalized to $\|\boldsymbol{\lambda}\|_1 = 1$ after each update. The learning rate $\rho(t)$ for the weights is empirically set to $0.1\eta(t)$ [13].

5. Discussions

In this section, we discuss the relationships between the proposed LOGM algorithm and their predecessors, and discuss their convergence, generalization ability and computational complexity.

5.1. Generative versus discriminative

The proposed prototype learning algorithm LOGM is closely related to the previous ones SNPC (equivalent to MAXP), MCE and RSLVQ. The LOGM and RSLVQ algorithms are based on minimization of the Conditional Log-likelihood Loss (CLL), while the MCE and the SNPC (MAXP) are based on direct minimization of the smoothed misclassification rate. From (18) and (21), the CLL results in a multiplier $1/P(C_k|\mathbf{x}_n)$ (C_k is the genuine class of $\mathbf{x}_n$) in gradient learning. This indicates that on training patterns that are more likely to be misclassified, the prototypes are updated with a larger move. On the other hand, the SNPC (MAXP) and RSLVQ algorithms compute the posterior probabilities of prototypes and classes in the framework of Gaussian mixture, while the MCE and LOGM algorithms compute the probability of correct classification based on the margin via reducing the multi-class problem into a binary one on each training pattern. The formulation of prototypes in a Gaussian mixture can be viewed as a generative model, and the margin-based formulation can be viewed as a discriminative model.

The generative model can be converted to a discriminative one under certain conditions. In the probability formula of the generative model (10), if we separate the terms about the positive class (genuine class k) from the ones about the negative class, the posterior probability can be re-written as

$$P(C_k|\mathbf{x}_n) = \frac{\sum_{s=1}^{S}\exp(\xi g_{ks})}{\sum_{s=1}^{S}\exp(\xi g_{ks}) + \sum_{l\neq k}\sum_{s=1}^{S}\exp(\xi g_{ls})} = \frac{1}{1+\exp(-a)}, \tag{33}$$

where $a = \log\sum_{s=1}^{S}\exp(\xi g_{ks})/\sum_{l\neq k}\sum_{s=1}^{S}\exp(\xi g_{ls})$. If $g_{ki} \gg g_{ki'}$ and $g_{rj} \gg g_{r'j'}$ where $g_{ki'}$ and $g_{r'j'}$ are the second largest in $\{g_{ks},\ s=1,2,\ldots,S\}$ and in $\{g_{ls},\ l\neq k,\ s=1,2,\ldots,S\}$, respectively, then $a \approx \xi(g_{ki} - g_{rj})$, which converts the generative model (10) to the discriminative one (20). This case happens for many training patterns when the prototypes distribute sparsely in the input space. One may note that the discriminative model focuses on the local and confusing regions because the misclassification loss only depends on g_{ki} and g_{rj}, which come from the genuine class and the most confusing rival class, respectively, while the generative model considers the entire space by involving all prototypes in the loss function.

The relationships between the prototype learning algorithms can be summarized in Table 2.

Table 2
Relationships between prototype learning algorithms.

	Generative	Discriminative
Misclassification Rate	SNPC (MAXP)	MCE
Negative Log-Likelihood	RSLVQ	LOGM

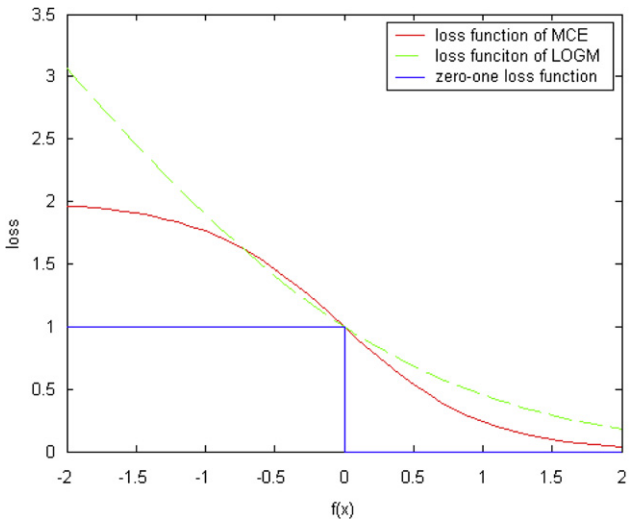

Fig. 1. Loss functions of MCE and LOGM based on hypothesis margin $f(\mathbf{x})$.

5.2. Convergence of prototype learning algorithms

Previous efforts [25,26] have been made to prove the convergence of LVQ, which can be viewed as an online (stochastic) gradient learning, or stochastic approximation problem [27,28]. The stochastic approximation algorithm is to solve the root of a conditional expectation about a pair of random variables. The derivative of the objective of expected risk in Eq. (1) is written as

$$\frac{\partial R_\phi f}{\partial\Theta} = \int\frac{\phi(f(\mathbf{x}))}{\partial\Theta}p(\mathbf{x})\,d\mathbf{x} = \mathbb{E}\left[\frac{\phi(f(\mathbf{x}))}{\partial\Theta}\right]. \tag{34}$$

The goal of prototype learning algorithms is to find Θ at which $\partial R_\phi f/\partial\Theta = 0$, where the stochastic approximation converges to a local minimum of $R_\phi f$ (more details in [28]). If the objective function is continuous with respect to the parameters to learn, a sequence of learning rate starting with a small value and vanishing gradually leads to convergence generally.

The loss functions of SNPC (MAXP) and RSLVQ involve all the prototype vectors and are apparently continuous with them. The loss functions of MCE, GLVQ and LOGM involve only two selected prototypes, which are dynamic depending on the training pattern. However, if we view the $g_i(\mathbf{x}_n)$ in Eq. (4) as the extreme case of a soft-max of multiple prototype measures (as derived in [16]), the loss function involving two soft-max distances is actually continuous with respect to all the prototype vectors. Hence, the convergence of these prototype learning algorithms are guaranteed by stochastic approximation.

5.3. Generalization of prototype learning algorithms

Section 5.1 has shown that the generative model can be converted to a discriminative one under certain conditions. Hence, our discussion of generalization performance focuses on

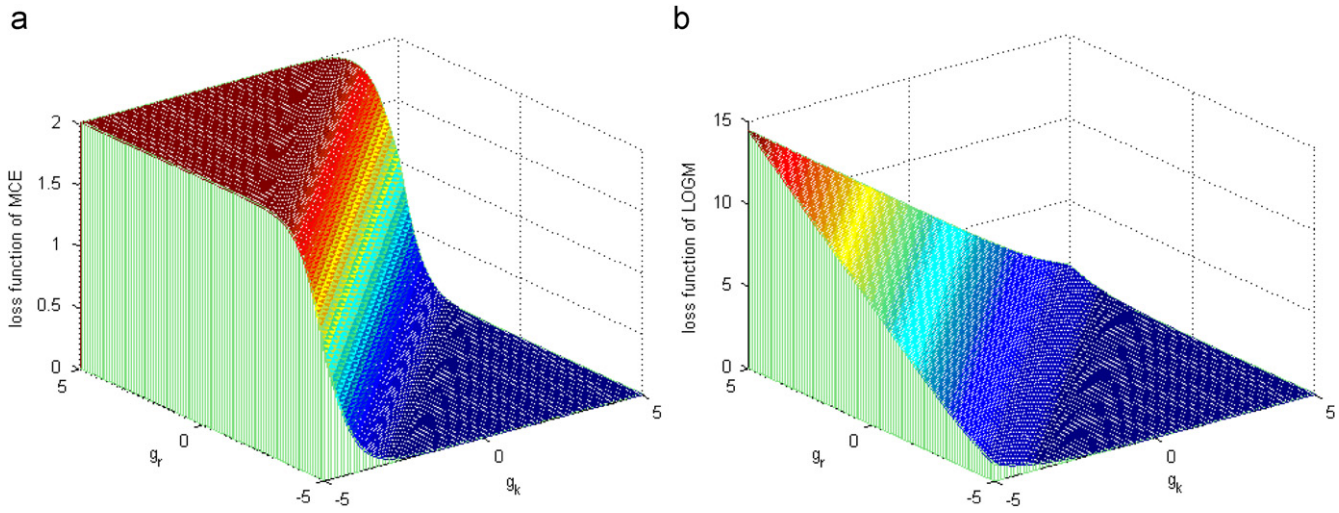

Fig. 2. Left: the loss of MCE as the function of $(g_k(x),g_r(x))$; right: the loss of LOGM as the function of $(g_k(x),g_r(x))$.

the prototype learning algorithms based on the discriminative model (margin-based algorithms).

Margin-based learning algorithms aim to minimize different loss functions based on certain margin. The prototype learning algorithms MCE (Minimum Classification Error) and LOGM (LOG-likelihood of Margin) minimize the loss $2/(1+\exp(2f(\mathbf{x})))$ and $\ln(1+\exp(-f(\mathbf{x})))/\ln 2$,[5] respectively. The loss functions are plotted in Fig. 1, where $f(\mathbf{x}) = g_k(\mathbf{x}) - g_r(\mathbf{x})$. Fig. 2 shows the loss of MCE and LOGM as the functions of $g_k(x)$ and $g_r(x)$. We can see that the loss of LOGM is convex with respect to the margin $f(\mathbf{x})$ and the variables $(g_k(x),g_r(x))$.[6] The property of classification-calibration [19] about two losses guarantees that the minimization of $R_\phi(f)$ may provide a reasonable surrogate for the minimization of 0–1 loss. A convex loss function is preferred because of its unique optima and guaranteed global convergence.

Crammer et al. [11] presented a generalization bound of LVQ based the hypothesis margin, which indicates that large hypothesis margins lead to small generalization error bounds. Our proposed prototype learning algorithms are hypothesis margin maximization ones. The margin-based learning algorithm LOGM minimizes the empirical error based on the margin $\|\mathbf{x}-\mathbf{m}_{rj}\|^2-\|\mathbf{x}-\mathbf{m}_{ki}\|^2$. By constraining the distance to prototypes in regularization (Eq. (22)), the hypothesis margin $\frac{1}{2}(\|\mathbf{x}-\mathbf{m}_{rj}\|-\|\mathbf{x}-\mathbf{m}_{ki}\|)$ is effectively maximized.

For an example, Fig. 3 shows the average hypothesis margins of five algorithms, LOGM, MCE, GLVQ (Generalized Learning Vector Quantization), RSLVQ (Robust Soft Learning Vector Quantization) and MAXP (MAXimum class Probability) on the USPS dataset during learning. We can see that before 600 rounds (each pattern is used once for updating in a round), LOGM can obtain larger average hypothesis margin than MCE and GLVQ, and the discriminative model-based algorithms generate larger hypothesis margins than the ones based on generative model.

5.4. Comparison of time complexity

Obviously, the MCE, GLVQ and LOGM algorithms have the same training time complexity $O(NTLSd + NTd)$ (T is the number of iterations and d is the dimension of input space), where the first term is the time complexity of computing g_r (g_k) and the second term is of two update operations. The SNPC (MAXP) and RSLVQ

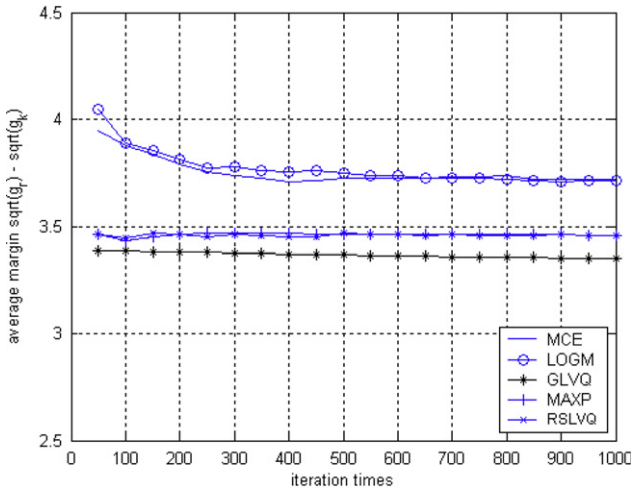

Fig. 3. Average margins of different algorithms during training on the USPS dataset (S=20 and $\alpha = 0.001$).

algorithms have the same time complexity $O(NTLSd + NTLSd)$ overall, where the first term is the time of calculating g_{ls} for all prototypes and the second term (for updating all prototypes) shows that they need more update time than MCE, GLVQ and LOGM. We have a close look at SNPC (soft nearest prototype classifier) and RSLVQ (Robust Soft Learning Vector Quantization) for one iteration. In updating the prototypes of the positive class, the RSLVQ needs more S division operations than the SNPC, but in updating the prototypes of the negative class, the RSLVQ needs less $(L-1)S$ multiplication operations than the SNPC. In sum, The SNPC needs more $TS(L-2)$ $(L \geq 2)$ basic operations than the RSLVQ during training. In classification stage, all the prototype learning algorithms have the same time complexity $O(LSd)$ for a new pattern.

The training and test times of different algorithms on the USPS dataset in Table 3 supports the above discussions about time complexity.

In Table 3, MAXP1 is a modified version of MAXP (equivalent to SNPC) [7], which updates the $S+1$ nearest prototypes (at least one of them comes from the negative class) instead of all prototypes on a training pattern. We similarly modify the RSLVQ algorithm to update $S+1$ nearest prototypes on a training pattern, and obtain the algorithm RSLVQ1. Given that $\mathbf{m}_{l_i s_i}, i = 1, 2, \ldots, S+1$, are the

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

[6] For LOGM it can be proven that $\partial^2\phi(f)/\partial z\,\partial z^T \geq 0$, where $z = (g_k(x), g_r(x))^T$.

Table 3
Training and test times (seconds) on USPS dataset.

Time	*MCE**	MCE	GLVQ	SNPC	MAXP1	LOGM	RSLVQ	RSLVQ1
Train	3.22	39.55	39.92	235.17	52.95	39.88	60.53	53.73
Test	0.26	0.44	0.45	0.47	0.44	0.47	0.44	0.44

$\alpha=0$, S = 20 were set for all prototype learning algorithms except for *MCE** (S = 1).

selected nearest prototypes to training pattern $\mathbf{x}_n$, the posterior probability for the genuine class k is computed by

$$\tilde{P}_k(\mathbf{x}_n)=\sum_{l_i=k}\tilde{P}_{l_i s_i}(\mathbf{x}_n), \tag{35}$$

where

$$\tilde{P}_{l_i s_i}(\mathbf{x}_n)=\frac{\exp(\xi g_{l_i s_i}(\mathbf{x}_n))}{\sum_{j=1}^{S+1}\exp(\xi g_{l_j s_j}(\mathbf{x}_n))}. \tag{36}$$

The MAXP1 and RSLVQ1 algorithms are obtained by replacing $P_k(\mathbf{x}_n)$ with $\tilde{P}_k(\mathbf{x}_n)$ and $P_{ls}(\mathbf{x}_n)$ with $\tilde{P}_{l_i s_i}(\mathbf{x}_n)$ in (14), (18) and (26), on the S+1 selected prototypes. Table 3 shows that the MAXP1 and RSLVQ1 algorithms need less training time than MAXP (SNPC) and RSLVQ.

6. Experiments

In our experiments, we have compared the performance of the proposed LOGM algorithm with previous representative algorithms (MCE, GLVQ, SNPC, MAXP1, RSLVQ and its variant RSLVQ1) as well as the state-of-the-art classifier support vector machine (SVM) [29]. We use the one-versus-all SVM classifier with Gaussian (RBF) kernel, with parameters trained using the SVM-light package [30]. We also evaluate the effects of the LOGM with prototype-dependent weighting and prototype-independent weighting.

6.1. Description of datasets

We evaluated the classification performance on 30 datasets, 28 of which are from the UCI Machine Learning Repository [31]. For 22 small-scale datasets, we evaluate the performance by 10-fold cross validation (cv-10), while eight large datasets are partitioned into unique training and test subsets. Some datasets contain discrete values of attributes, which are converted into continuous ones by mapping k discrete values to $\{0,1,\ldots,k-1\}$ one-by-one. The patterns with missing values were removed from the datasets since our algorithms do not handle missing values. The datasets are summarized in Table 4.

The datasets USPS and 20NG (20Newgroups) are from the public sources other than the UCI. The USPS dataset contains normalized handwritten digit images of 16×16 pixels, with pixel level between 0 and 255. It was commonly partitioned into 7291 training samples and 2007 test samples. The 20NG dataset is a collection of approximately 20,000 documents that were collected from 20 different newsgroups [32]. For convenience of comparison, the *bydate* version of this dataset along with its train/test split was used in our experiments. The text documents were preprocessed by removing tags and stopwords[7] and by word stemming[8] [33]. Then, the documents were transformed into a representation suitable for classification by TFIDF weighting [34]. The information gain method [35] was used to select 1000 features of highest scores.

[7] Stopwords are the frequent words that carry no information (i.e. pronouns, prepositions, conjunctions, etc.).

[8] Word stemming is a process of suffix removal to generate word stems.

Table 4
Description of the datasets used in experiments.

No.	Dataset	#feature	#class	#training	#test
1	Breast	10	2	683	cv-10
2	Corral	6	2	128	cv-10
3	Dermatology	34	6	358	cv-10
4	Diabetes	8	2	768	cv-10
5	Flare	10	2	1066	cv-10
6	Glass	9	7	214	cv-10
7	Glass2	9	2	163	cv-10
8	Heart	13	2	270	cv-10
9	Hepatitis	19	2	80	cv-10
10	Ionosphere	34	2	351	cv-10
11	Iris	4	3	150	cv-10
12	Liver	6	2	345	cv-10
13	Mass	5	2	830	cv-10
14	Pima	8	2	768	cv-10
14	Segment	19	7	2310	cv-10
16	Sonar	60	2	208	cv-10
17	Soybean-large	35	19	562	cv-10
18	Vehicle	18	4	846	cv-10
19	Vote	16	2	435	cv-10
20	Waveform-21	21	3	5000	cv-10
21	Wine	13	3	178	cv-10
22	Wdbc	30	2	569	cv-10
23	20NG	1000	20	11,314	7532
24	Chess	36	2	2130	1066
25	Letter	16	26	15,000	5000
26	Optdigit	64	10	3823	1797
27	Pendigit	16	10	7494	3498
28	Satimage	36	6	4435	2000
29	Thyroid	21	3	3772	3428
30	USPS	256	10	7291	2007

The datasets Chess, Satimage and Letter were splitted into training and test subsets following [36]. The Optdigit, Pendigit and Thyroid were given the training and test subsets designated by the UCI.

At last, all the datasets except 20NG were normalized by linearly scaling each attribute to $[-1,1]$.

6.2. Setup of experiments

In implementing the prototype learning algorithms, the prototypes were initialized by k-means clustering of classwise data. For all the algorithms, 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 square Euclidean distance of all training patterns to the nearest cluster center. In the t-th round of iteration, the learning rate of the n-th pattern was $\eta(0)(1-(tN+n)/TN)$, where T is the maximum number of rounds. For the 22 small datasets, T was set to 100 and the prototype number S was selected from {1, 2, 3, 4, 5}. For the eight large datasets, T was set to 40 and S was selected from {5, 10, 15, 20, 25} following [10]. The regularization coefficient α was selected from {0,0.001,0.005,0.01,0.05}. The training parameters and model parameters were optimized in the space of the cubic grid (α,S,τ) by cross-validation on the training data.

The smoothing parameter ξ was initialized as $2/cov$, and was fixed during training for all the prototype learning algorithms except the SNPC. For SNPC, ξ was increased by ratio 1.00001 after each training pattern in spirit of deterministic annealing. It was observed in our experiments that if ξ is initialized properly, whether to evolve ξ in training or not does not influence the performance.

For the SVM classifier with RBF kernel, we only considered the tradeoff parameter C and the kernel width γ as done in [37], where $\log_2 C$ and $\log_2\gamma$ were selected from {10, 8, 6, 4, 2} and

Table 5
Classification accuracies (%) of prototype classifiers and SVM.

No.	Dataset	MCE	GLVQ	SNPC	MAXP1	LOGM	RSLVQ	RSLVQ1	SVM
1	Breast	94.63	94.24	94.64	**94.76**	94.31	94.13	94.14	94.27
2	Corral	99.77	99.69	99.69	99.85	**100.0**	99.85	99.77	100.0*
3	Dermatology	95.16	95.55	**95.57**	95.33	95.22	94.40	93.86	96.33*
4	Diabetes	76.13	75.37	75.60	75.93	**76.81**	76.50	75.77	76.68
5	Flare	86.97	86.72	87.05	86.90	87.08	86.97	**87.22**	87.04
6	Glass	68.06	66.45	68.29	68.33	67.56	**68.51**	66.56	68.43
7	Glass2	73.77	72.26	72.64	72.84	74.41	**76.07**	75.19	75.84
8	Heart	82.37	83.04	83.22	**83.41**	81.18	81.70	81.44	83.04
9	Hepatitis	87.25	86.88	86.75	86.38	87.50	**88.00**	**88.00**	84.50
10	Ionosphere	87.75	89.12	87.47	87.52	88.38	89.03	**89.17**	94.53*
11	Iris	95.60	95.53	95.73	95.73	**96.20**	96.00	96.00	95.53
12	Liver	69.92	66.76	69.43	69.50	69.46	**70.07**	68.68	71.69*
13	Mass	**81.86**	81.08	80.23	80.05	81.23	81.78	81.22	82.09*
14	Pima	75.94	75.75	75.75	75.94	**76.39**	75.63	75.38	76.56*
15	Segment	95.67	95.48	95.50	95.29	**95.94**	**95.94**	94.91	96.90*
16	Sonar	86.74	84.69	85.77	85.91	**86.81**	86.19	85.51	87.76*
17	Soybean-large	89.27	89.41	89.86	**90.09**	89.90	88.84	87.24	89.75
18	Vehicle	80.38	78.25	77.02	76.95	**81.95**	81.26	75.88	85.61*
19	Vote	94.89	94.53	94.85	95.04	**95.26**	95.17	95.16	94.88
20	Waveform-21	86.84	86.20	83.64	84.64	86.84	**86.96**	86.87	86.97*
21	Wine	97.22	96.44	**97.83**	97.56	97.27	97.45	97.34	98.01*
22	Wdbc	97.35	97.21	**97.44**	97.38	97.22	97.31	97.26	97.34
23	20NG	74.24	74.19	73.80	73.81	**74.84**	73.70	74.22	76.55*
24	Chess	97.75	96.44	98.31	98.22	98.41	98.69	**99.44**	99.16
25	Letter	95.66	95.06	95.74	96.02	95.92	**96.16**	95.76	97.04*
26	Optdigit	98.27	98.00	98.05	98.27	**98.33**	97.77	97.89	98.72*
27	Pendigit	97.86	97.57	98.03	**98.06**	97.00	97.51	97.43	98.31*
28	Satimage	72.15	70.75	72.80	**73.15**	72.80	71.20	71.60	74.85*
29	Thyroid	93.82	93.79	93.90	93.82	93.82	93.90	**94.11**	96.56*
30	USPS	94.62	93.77	94.37	94.77	94.32	94.17	**94.82**	95.42*
Average rank		3.82	5.63	4.15	3.63	**3.05**	3.47	4.25	

The highest accuracy of each dataset given by prototype classifiers is highlighted in boldface. The accuracy of SVM is marked asterisk if it is higher than the prototype classifiers. The last row gives the average rank of prototype classifiers.

$\{-1, -3, -5, -7, -9\}$, respectively. All the experiments ran on the platform of Torch [38], a C++ library.

For the 22 small datasets, the accuracy of an algorithm on a dataset was obtained via 10 runs of 10-fold cross validation. The detailed procedure is below:

(1) For each run, the dataset was randomly divided into 10 disjoint subsets of approximately same size by stratified sampling. We kept the same divisions for all learning algorithms.
(2) Each of 10 subsets was used as test set and the remaining data was used for training. The 10 subsets were used for testing rotationally for evaluating the classification accuracy.
(3) During each training process, the training parameters were determined as follows: first, we held out $\frac{1}{3}$ of the training data by stratified sampling for validation while the classifier parameters were estimated on the remaining $\frac{2}{3}$ of data (the split of training data is the same for all learning algorithms). After selecting training parameters that gave the highest validation accuracy, all the training data were used to re-train the classifier for evaluation on test data.

For the eight large datasets, we only held out $\frac{1}{5}$ of the training data for validation to select training parameters. For evaluation on test data, the classifier was then re-trained on the whole training set with the selected training parameters.

6.3. Results and discussions

The classification accuracies of seven prototype learning algorithms (MCE, GLVQ, SNPC, MAXP1, LOGM, RSLVQ and RSLVQ1) and the SVM classifier with RBF kernel on the 30 datasets are listed in Table 5. On each dataset, the highest accuracy of prototype classifiers is highlighted in boldface. The average ranks[9] of prototype learning algorithms on 30 datasets are given in the bottom row. The SVM is used as a reference to demonstrate the relative performance of the prototype learning algorithms.

Since the advantage of a learning algorithm is variable depending on datasets, we can only compare the algorithms in statistical sense. In Table 5, we can see that among seven prototype learning algorithms, the LOGM (LOG-likelihood of Margin) gives the highest accuracy on 10 of 30 datasets, followed by the RSLVQ (robust soft learning vector quantization) top ranked on 7 datasets, RSLVQ1 on 6 datasets, and then MAXP1 (the variant of MAXP or SNPC) on 5 datasets. Comparing the average ranks, the margin-based algorithm LOGM has the highest average rank (3.05), followed by the RSLVQ (3.47), MAXP1 (3.63) and MCE (3.82). The performance of RSLVQ1 turns out to be less stable than MAXP1 and MCE (Minimum Classification Error).

Comparing prototype classifiers with SVM, the SVM gives the highest accuracy on 18 of 30 datasets. Though the SVM outperforms prototype classifiers on majority of datasets, it has far higher complexity of training and classification on all the datasets. For example, on the USPS dataset, the SVM training by SVM-light costs 124 s, and the classification of test samples costs 22 s due to the large number of support vectors (597 support vectors per class on average). In comparison, learning 20

[9] The top rank (highest accuracy) on a dataset is scored 1, second rank scored 2, and so on.

prototypes per class (200 prototypes in total) by the LOGM algorithm costs only 39.88 s, and the classification time is 0.468 s.

The selected values of training parameters (α, S, τ) in validation give some cues to the characteristics of learning algorithms. The selected parameters on two datasets are shown in Table 6. The positivity of α (regularization coefficient) justifies that the regularization in prototype learning does benefit the generalization performance.

The statistical tests for comparisons of pairs of learning algorithms on multiple datasets are given. The signed ranks test [39] is claimed to be more sensible and safer than the paired t-test. It ranks the differences of performance between two classifiers on each dataset, ignores the signs, and compares the ranks for the positive and the negative differences.

Let d_i be the difference of performance scores of two classifiers on the i-th outcome of K datasets. The differences except H items with $d_i = 0$ are ranked according to their absolute values, and in case of ties, average ranks are assigned. Let R^+ be the sum of ranks for the datasets on which the second algorithm outperforms the first and R^- the sum of ranks for the opposite. The statistics z_0 is constructed below:

$$z_0 = \frac{|\min(R^+, R^-) - \frac{1}{4}N(N+1)|}{\sqrt{\frac{1}{24}N(N+1)(2N+1)}}, \tag{37}$$

where $N = K - H$. Then, the p-value is computed by $p = P(|z| > z_0)$, where the random variable z observes standard normal distribution. The p-value shows the significance probability of no difference between two algorithms. If p-value is nearly zero, the hypothesis that two algorithms perform equally is denied. A value $p < 0.05$ indicates that two algorithms differ significantly with over probability 0.95. We judge whether the second algorithm outperforms the first according to the sign of $R^+ - R^-$.

The results of the signed ranks tests are shown in Table 7. It is seen that the LOGM algorithm is significantly superior to the SNPC (p = 0.0410) and GLVQ (p = 0.0004), and also outperforms the MCE, MAXP1, and RSLVQ1. The RSLVQ is slightly inferior to the LOGM (p=0.6420), but is significantly superior to the GLVQ and RSLVQ1, and slightly superior to the other prototype learning algorithms. The SVM is significantly superior to all the prototype learning algorithms at the cost of higher complexity. In contrast to the average ranks of prototype learning algorithms in Table 5, the MAXP1 algorithm is shown to be statistically inferior to the MCE (p=0.5971) though it has a higher average rank than the MCE, but the difference is not significant.

Finally, we evaluate the effects of prototype learning and distance metric learning under the LOGM criterion. We combine prototype learning and weight learning, then obtain five instances of implementation: (1) prototype initialization by k-means clustering of classwise data, without prototype learning and weight learning; (2) weight learning only by LOGM (LOGM-W, updating weights only in gradient descent); (3) prototype learning only by LOGM (LOGM-P, i.e., LOGM in the former text); (4) prototype learning with prototype-independent weighting (LOGM-PIW); (5) prototype learning with prototype-dependent weighting (LOGM-PDW). Table 8 gives the results of these algorithms on 22 small datasets.

The results in Table 8 show that when learning prototypes or weights only, prototype learning is more effective to improve the classification performance than distance metric learning (LOGM-P

Table 6
Selected parameters (α, S, τ) of prototype learning algorithms on USPS and 20NG.

	MCE	GLVQ	SNPC	MAXP1	LOGM	RSLVQ	RSLVQ1
USPS							
α	0.005	0.001	0.001	0.01	0.005	0.01	0.001
S	20	5	25	15	10	15	20
τ	1.5	1.5	2	2	1.5	2	1.5
20NG							
α	0.01	0.005	0.01	0.001	0.001	0.001	0.001
S	5	5	20	5	5	15	15
τ	0.5	0.1	0.5	1	0.5	2	1.5

Table 8
Comparison of accuracies of prototype learning with weighted distance metric on 22 small datasets.

Dataset	k-means	LOGM-W	LOGM-P*	LOGM-PIW	LOGM-PDW	SVM*
Breast	94.12	94.03	94.31	**94.40**	94.36	94.27
Corral	98.37	99.86	**100.00**	**100.00**	**100.00**	**100.00**
Dermatology	92.95	93.46	95.22	95.72	95.83	**96.33**
Diabetes	70.25	70.61	**76.81**	76.73	76.60	76.68
Flare	78.81	81.67	87.08	**87.45**	87.23	87.04
Glass	62.30	62.39	67.56	67.56	**69.36**	68.43
Glass2	72.12	72.19	74.41	74.61	74.54	**75.84**
Heart	79.92	80.00	81.18	80.85	80.48	**83.04**
Hepatitis	86.75	87.00	87.50	**87.75**	87.63	84.50
Ionosphere	87.35	88.52	88.38	88.78	89.31	**94.53**
Iris	96.13	96.13	96.20	96.20	**96.67**	95.53
Liver	59.51	59.46	69.46	69.46	71.55	**71.69**
Mass	78.51	78.50	81.23	81.41	81.99	**82.09**
Pima	71.33	71.91	76.39	76.46	76.17	**76.56**
Segment	93.92	94.26	95.94	96.17	96.86	**96.90**
Sonar	82.58	82.72	86.81	86.81	**88.16**	87.76
Soybean-large	88.64	89.12	89.90	90.15	**90.26**	89.75
Vehicle	70.44	71.35	81.95	82.19	83.98	**85.61**
Vote	94.24	95.14	95.26	**95.69**	95.67	94.88
Waveform-21	81.07	81.30	86.84	87.10	**87.34**	86.97
Wine	96.48	96.53	97.27	97.33	97.49	**98.01**
Wdbc	95.64	95.70	97.22	97.24	97.21	**97.34**
Average rank	5.75	4.98	3.30	2.39	**2.20**	2.38

(1) k-means (no prototype and weight learning); (2) LOGM for weight learning (LOGM-W); (3) LOGM for prototype learning (LOGM-P); (4) LOGM for prototype learning and prototype-independent weighting (LOGM-PIW); (5) LOGM for prototype learning and prototype-dependent weighting (LOGM-PDW). The results in the columns with asterisk (LOGM-P and SVM) are consistent with those in Table 5.

Table 7
Signed ranks two-tailed tests (p-values) for comparing pairs of algorithms on 30 datasets.

Second\first	MCE	GLVQ	SNPC	MAXP1	LOGM	RSLVQ	RSLVQ1
GLVQ	−**0.0005**						
SNPC	−0.3135	+**0.0305**					
MAXP1	−0.5971	+**0.0417**	+0.1195				
LOGM	+0.1084	+**0.0004**	+**0.0410**	+0.1048			
RSLVQ	+0.7703	+**0.0055**	+0.2098	+0.3524	−0.6420		
RSLVQ1	−0.1274	+0.0878	−0.5999	−0.4106	−0.0703	−**0.0372**	
SVM	+**0.0001**	+**0.0000**	+**0.0003**	+**0.0005**	+**0.0006**	+**0.0004**	+**0.0002**

The entries with $p < 0.05$ are highlighted in boldface. The sign of $R^+ - R^-$ between two algorithms is denoted by + or −. A + indicates that the second algorithm outperforms the first one.

significantly outperforms LOGM-W). The performance of LOGM-P is further improved by combining prototype learning and distance metric learning, while prototype-dependent weighting (LOGM-PDW) yields even higher performance than prototype-independent weighting (LOGM-PIW). According to the results on 22 datasets, the LOGM-PDW even has higher average rank (2.20) than the SVM classifier (2.38). This is because the SVM classifier has very low ranks on a few datasets (e.g., rank 4 on Breast and rank 6 on Hepatitis).When comparing LOGM-PDW and SVM directly, the SVM outperforms the LOGM-PDW on 12 datasets, the LOG-PDW outperforms on nine datasets, and there is a tie on one dataset.

7. Concluding remarks

In this paper, we have proposed a discriminative prototype learning algorithm based on the Conditional Log-likelihood Loss (CLL), called LOG-likelihood of Margin (LOGM). A regularization term is added to avoid over-fitting in training. The joint effect of convex margin-based loss minimization and regularization produces large hypothesis margins, which lead to low generalization error bounds. In experiments on 30 datasets, the LOGM algorithm is demonstrated to outperform the previous representative algorithms MCE, GLVQ, SNPC (MAXP), and RSLVQ. We observed in experiments that the LOGM produces larger average hypothesis margin than the other prototype learning algorithms.

Though this paper focuses on the effects of loss functions on prototype learning algorithms, we have extended the LOGM algorithm to prototype learning with weighted distance metric. Experimental results show that the LOGM with prototype-dependent weighting achieves comparable performance to the state-of-the-art SVM classifier, yet the training time complexity and the test time complexity of SVM is much higher than the prototype classifier. This offers applicability to large scale classification problems such as character recognition [7] and text classification [8]. The LOGM as a learning criterion can also be applied to other classifier structures based on gradient descent, such as neural networks [16] and quadratic discriminant functions [24].

Acknowledgments

This work is supported in part by the Hundred Talents Program of Chinese Academy of Sciences (CAS) and the National Natural Science Foundation of China (NSFC) under Grant nos. 60775004 and 60825301.

References

[1] R.O. Duda, P.E. Hart, D.G. Stork, Pattern Classification, second ed., Wiley Interscience, New York, 2001.
[2] C.-L. Chang, Finding prototypes for nearest neighbor classifiers, IEEE Trans. Comput. 23 (11) (1974) 1179–1184.
[3] T. Kohonen, Improved versions of learning vector quantization, Neural Networks 1 (17–21) (1990) 545–550.
[4] A. Sato, K. Yamada, Generalized learning vector quantization, in: Advances in Neural Information Processing Systems, 1995, pp. 423–429.
[5] J. Bezdek, T. Reichherzer, G. Lim, Y. Attikiouzel, Multiple-prototype classifier design, IEEE Trans. System Man Cybernet. Part C 28 (1) (1998) 67–79.
[6] L. Kuncheva, J. Bezdek, Nearest prototype classification: clustering, genetic algorithms, or random search?, IEEE Trans System Man Cybernet. Part C 28 (1) (1998) 160–164.
[7] C.-L. Liu, M. Nakagawa, Evaluation of prototype learning algorithms for nearest-neighbor classifier in application to handwritten character recognition, Pattern Recognition 34 (3) (2001) 601–615.
[8] M.T. Martín-Valdivia, M.G. Vega, L.A.U. López, LVQ for text categorization using a multilingual linguistic resource, Neurocomputing 55 (3–4) (2003) 665–679.
[9] P. Schneider, M. Biehl, B. Hammer, Advanced metric adaptation in generalized LVQ for classification of mass spectrometry data, in: Proceeding of the Workshop on Self Organizing Maps, 2007.
[10] T. Kohonen, J. Hynninen, J. Kangas, J. Laaksonen, K. Torkkola, LVQ PAK: the learning vector quantization program package, Technical Report, Helsinki University of Technology, 1995.
[11] K. Crammer, R. Gilad-Bachrach, A. Navot, N. Tishby, Margin analysis of the LVQ algorithm, in: Advances in Neural Information Processing Systems, 2002, pp. 462–469.
[12] A. Qin, P. Suganthan, Initialization insensitive LVQ algorithm based on cost-function adaptation, Pattern Recognition 38 (5) (2005) 773–776.
[13] B. Hammer, T. Villmann, Generalized relevance learning vector quantization, Neural Network 15 (8–9) (2002) 1059–1068.
[14] R. Paredes, E. Vidal, Learning prototypes and distances: a prototype reduction technique based on nearest neighbor error minimization, Pattern Recogn. 39 (2) (2006) 180–188.
[15] C.E. Pedreira, Learning vector quantization with training data selection, IEEE Trans. Pattern Anal. Mach. Intell. 28 (1) (2006) 157–162.
[16] B.-H. Juang, S. Katagiri, Discriminative learning for minimum error classification, IEEE Trans. Signal Processing 40 (12) (1992) 3043–3054.
[17] S. Seo, M. Bode, K. Obermayer, Soft nearest prototype classification, IEEE Trans. Neural Networks 14 (2) (2003) 390–398.
[18] S. Seo, K. Obermayer, Soft learning vector quantization, Neural Comput. 15 (7) (2003) 1589–1604.
[19] P.L. Bartlett, M.I. Jordan, J.D. McAuliffe, Convexity, classification, and risk bounds, J. Am. Statist. Assoc. 101 (473) (2006) 138–156.
[20] X. Jin, C.-L. Liu, X. Hou, Prototype learning by margin-based conditional log-likelihood loss, In: Proceedings of the 19th ICPR, Tampa, USA, 2008.
[21] C.M. Bishop, Pattern Recognition and Machine Learning, Springer, New York, 2006.
[22] H. Ney, On the probabilistic interpretation of neural network classifiers and discriminative training criteria, IEEE Trans. Pattern Anal. Mach. Intell. 17 (2) (1995) 107–119.
[23] D. Grossman, P. Domingos, Learning bayesian network classifiers by maximizing conditional likelihood, in: Proceedings of the 21th ICML, 2004, pp. 361–378.
[24] C.-L. Liu, H. Sako, H. Fujisawa, Discriminative learning quadratic discriminant function for handwriting recognition, IEEE Trans. Neural Networks 15 (2) (2004) 430–444.
[25] J.S. Baras, A. LaVigna, Convergence of Kohonen's learning vector quantization, Neural Networks 3 (17–21) (1990) 17–20.
[26] B. Kosmatopoulos, M.A. Christodoulou, Convergence properties of a class of learning vector quantization algorithms, IEEE Trans. Image Processing 5 (2) (1996) 361–368.
[27] H. Robbins, S. Monro, A stochastic approximation method, Ann. Math. Statist. 22 (1951) 400–407.
[28] L. Bottou, On-line learning and stochastic approximations, in: On-line Learning in Neural Networks, Cambridge University Press, Cambridge, 1999, pp. 9–42.
[29] V.N. Vapnik, The Nature of Statistical Learning Theory, Springer, New York, 1995.
[30] T. Joachims, Making large-scale support vector machine learning practical, in: Advances in Kernel Methods: Support Vector Learning, The MIT Press, 1999, pp. 169–184.
[31] C. Blake, C. Merz, UCI machine learning repository, University of California Irvine 〈http://www.ics.uci.edu/mlearn/MLRepository.html〉, 1998.
[32] K. Lang, Newsweeder: learning to filter netnews, in: Proceedings of the 12th ICML, 1995, pp. 331–339.
[33] M. Porter, An algorithm for suffix stripping, Program 14 (3) (1980) 130–137.
[34] F. Sebastiani, Machine learning in automated text categorization, ACM Comput. Surv. 34 (1) (2002) 1–47.
[35] Y. Yang, J.O. Pedersen, A comparative study on feature selection in text categorization, in: Proceedings of the 14th ICML, 1997, pp. 412–420.
[36] N. Friedman, D. Geiger, M. Goldszmidt, Bayesian network classifiers, Mach. Learning 29 (2–3) (1997) 131–163.
[37] W. Wang, Z. Xu, W. Lu, X. Zhang, Determination of the spread parameter in the Gaussian kernel for classification and regression, Neurocomputing 55 (3–4) (2003) 643–663.
[38] R. Collobert, S. Bengio, J. Mariéthoz, Torch: a modular machine learning software library, Technical Report IDIAP-RR 02–46, IDIAP, 2002.
[39] J. Demšar, Statistical comparisons of classifiers over multiple data sets, J. Mach. Learn. Res. 7 (2006) 1–30.

About the Author—XIAO-BO JIN received the B.S. degree in Management Science and Engineering from Xi'an University of Architecture and Technology, Xi'an, China, in 2002 and the M.S. degree in Computer Software and Theory from Northwestern Polytechnic University, Xi'an, China, in 2005. Currently, he is working toward the Ph.D. degree in Pattern Recognition and Intelligent System at the National Laboratory of Pattern Recognition, Institute of Automation, Chinese Academy of Sciences, Beijing, China. His research interests include Text Mining and Machine Learning.

About the Author—CHENG-LIN LIU received the B.S. degree in Electronic Engineering from Wuhan University, Wuhan, China, the M.E. degree in Electronic Engineering from Beijing Polytechnic University, Beijing, China, the Ph.D. degree in Pattern Recognition and Artificial Intelligence from the Institute of Automation, Chinese Academy of Sciences, Beijing, China, in 1989, 1992 and 1995, respectively. He was a postdoctoral fellow at Korea Advanced Institute of Science and Technology (KAIST) and later at Tokyo University of Agriculture and Technology from March 1996 to March 1999. From 1999 to 2004, he was a research staff member and later a senior researcher at the Central Research Laboratory, Hitachi, Ltd., Tokyo, Japan. From 2005, he has been a Professor at the National Laboratory of Pattern Recognition (NLPR), Institute of Automation, Chinese Academy of Sciences, Beijing, China, and is now the deputy director of the laboratory. His research interests include pattern recognition, image processing, neural networks, machine learning, and especially the applications to character recognition and document analysis. He has published over 90 technical papers at international journals and conferences. He won the IAPR/ICDAR Young Investigator Award of 2005.

About the Author—XINWEN HOU received his Ph. D degree from the Department of Mathematics of Peking University in 2001. He got his B.S. and M.S. degrees from Zhengzhou University in 1995 and the University of Science and Technology of China in 1998, respectively. From 2001 to 2003, he was a postdoctoral fellow with the department of Mathematics of Nankai University. He joined the National Laboratory of Pattern Recognition in December 2003, and is now an associate professor. His current research interests include machine learning, image recognition, video understanding and ensemble learning.