Inverse of Lorentzian Mixture for Simultaneous Training of Prototypes and Weights

Atsushi Sato and Masato Ishii
Information and Media Processing Laboratories, NEC Corporation 1753, Shimonumabe, Nakahara-ku, Kawasaki, Japan

Keywords: Lorentzian Mixture, Nearest Neighbor, Loss Minimization, Bayes Decision Theory, Machine Learning.

Abstract: This paper presents a novel distance-based classifier based on the multiplicative inverse of Lorentzian mixture, which can be regarded as a natural extension of the conventional nearest neighbor rule. We show that prototypes and weights can be trained simultaneously by General Loss Minimization, which is a generalized version of supervised learning framework used in Generalized Learning Vector Quantization. Experimental results for UCI machine learning repository reveal that the proposed method achieves almost the same as or higher classification accuracy than Support Vector Machine with a much fewer prototypes than support vectors.

1 INTRODUCTION

Distance-based classifiers have been widely used in real applications, because they achieve good performance in spite of their simple structures with a few prototypes. One popular learning algorithm of prototypes is Learning Vector Quantization (LVQ) (Kohonen, 1995), and the margin-based theoretical analysis has proved that LVQ is a family of large margin classifiers (Crammer et al., 2003). In the early stage of LVQ research, learning algorithms were improved by heuristic approaches. However, improvements have been conducted based on loss minimization, since Generalized Learning Vector Quantization (GLVQ) (Sato and Yamada, 1996; Sato, 1998) was proposed. One of the major improvements concerns the choice of an appropriate distance measure (Hammer and Villmann, 2002; Schneider et al., 2009; Villmann and Haase, 2011).

Meanwhile, kernel classifiers have been investigated, and Support Vector Machine (SVM) (Cortes and Vapnik, 1995) has become one of the powerful classification method. Many works have shown that SVM outperforms conventional classifiers. Extensions of GLVQ by means of kernel functions have been also investigated (Qin and Suganthan, 2004; Sato, 2010), and the experiments reveal that the extended models achieve almost the same classification accuracy as SVM. However, only weights for the kernels are trained, and positions of prototypes are never changed in these models, like in SVM. This may lead to the limit of classification capability, and to the increase of the number of prototypes.

It seems to be natural to train prototypes and their weights simultaneously to achieve good classification performance with a fewer prototypes. A learning method of prototypes as well as weights has been proposed (Karayiannis, 1996), but it cannot be suitable for classification problems, because it was formulated through unsupervised learning. One possible alternative is to repeat adding and removing prototypes during learning to decrease errors (Grbovic and Vucetic, 2009), but the optimality cannot be ensured, because the weights for the prototypes are estimated by a heuristic approach.

In order to train prototypes and their weights through supervised learning framework, this paper presents a novel distance-based classifier by utilizing the multiplicative inverse of Lorentzian mixture. Lorentzian has been evaluated as a response function instead of sigmoid in multi-layered neural networks (Giraud et al., 1995), but in this paper, we utilize it for nearest neighbor classifiers. We show that the proposed classifier can be regarded as a natural extension of the conventional nearest neighbor rule. We also show that prototypes and their weights can be trained simultaneously by General Loss Minimization (Sato, 2010), which is a generalized version of learning criterion used in GLVQ. Experimental results for UCI machine learning repository (Blake and Merz, 1998) reveal that the proposed method achieves almost the same as or higher classification accuracy than SVM, while the number of prototypes is much less than that of support vectors.

2 PROPOSED METHOD

2.1 Inverse of Lorentzian Mixture

Lorentzian, which is also known as Cauchy-distribution, has the probability density function

$$p(x) = \frac{1}{\pi}\left[\frac{\gamma}{(x-y)^2+\gamma^2}\right], \qquad (1)$$

where y is the peak location and γ is the half-width at half-maximum. Then, the discriminant function for class ω_k using the mixture of Lorentzian in d-dimensional space can be written as

$$g_k(\mathbf{x}) = \sum_{i=1}^{m_k} \alpha_{ki}\left[\frac{\gamma_{ki}}{\|\mathbf{x}-\mathbf{y}_{ki}\|^2+\gamma_{ki}^2}\right], \qquad (2)$$

where $\alpha_{ki} \geq 0$ is a weight for the i-th distribution, and $\mathbf{y}_{ki}$'s $(i = 1,\cdots,m_k)$ are regarded as prototypes. Assuming that $\alpha_{ki}\gamma_{ki} = 1$, we can obtain a more simple form

$$g_k(\mathbf{x}) = \sum_{i=1}^{m_k} \frac{1}{\|\mathbf{x}-\mathbf{y}_{ki}\|^2+{b_{ki}}^2}, \qquad (3)$$

where $1/\alpha_{ki}$ is replaced by b_{ki} for ease of description. As shown in Fig. 1, the height (or weight) of each distribution in Eq. (3) is defined only by the bias b_{ki}, so the above assumption is found to be useful for simplifying the mixture of Lorentzian.

Let us consider the dissimilarity form by taking the multiplicative inverse of Eq. (3) as follows:

$$d_k(\mathbf{x}) = \frac{1}{g_k(\mathbf{x})} = \left[\sum_{i=1}^{m_k}\left(\|\mathbf{x}-\mathbf{y}_{ki}\|^2+b_{ki}^2\right)^p\right]^{1/p}, \qquad (4)$$

where $p = -1$. Since this equation is formulated by using L_p norm, we can extend it by taking appropriate value for $p < 0$. For example, when $p \to -\infty$, we can obtain the nearest neighbor rule using biased Euclidean distance as follow:

$$\lim_{p\to-\infty} d_k(\mathbf{x}) = \min_{i=1}^{m_k}\left(\|\mathbf{x}-\mathbf{y}_{ki}\|^2+b_{ki}^2\right). \qquad (5)$$

Therefore, Eq. (4) can be regarded as a natural extension of the conventional nearest neighbor rule. If we use $(d+1)$-dimensional vectors defined by $\mathbf{X} \leftarrow (\mathbf{x}, 0)$ and $\mathbf{Y}_{ki} \leftarrow (\mathbf{y}_{ki}, b_{ki})$, Eq. (4) can be rewritten as

$$d_k(\mathbf{X};\theta) = \left[\sum_{i=1}^{m_k}\left(\|\mathbf{X}-\mathbf{Y}_{ki}\|^2\right)^p\right]^{1/p}, \qquad (6)$$

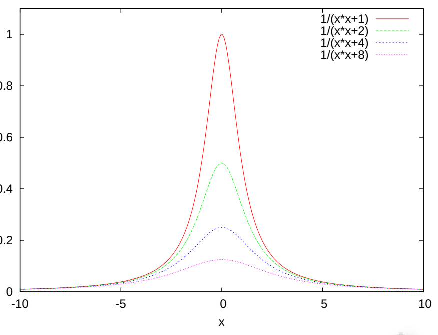

Figure 1: Examples of Lorentzian using different values for the bias. As the bias becomes larger, the height of the Lorentzian becomes lower.

where $\theta = \{\mathbf{Y}_{ki} | k = 1,\cdots,K; i = 1,\cdots,m_k\}$ denotes a set of classifier parameters. Therefore, to train the $(d+1)$-dimensional vectors $\{\mathbf{Y}_{ki}\}$ is to train both the prototypes and ther biases, simultaneously.

2.2 Learning based on GLM

To estimate the values of classifier parameters, General Loss Minimization (GLM) is employed, because it is a general framework for classifier design (Sato, 2010). GLM can deal with prior probabilities and various losses as well as zero-one loss for multi-class classification problems, but more simplified form is used in this paper. Employing zero-one loss and assuming that the prior probabilities are proportional to the number of samples in each class, the total loss can be written as follows:

$$L(\theta) = \frac{1}{N}\sum_{n=1}^{N}\sum_{k=1}^{K}\sum_{j\neq k}^{K} f(\rho_{kj}(\mathbf{x}_n;\theta))1(t_n = \omega_k). \qquad (7)$$

where θ is a set of classifier parameters, N is the number of training samples, K is the number of classes, $\mathbf{x}_n$ is the n-th training sample, t_n is the genuine class of $\mathbf{x}_n$, ω_k is the k-th class, and $1(\cdot)$ denotes an indicator function such that $1(true) = 1$ and $1(false) = 0$. $f(\rho_{kj}(\cdot))$ is a sort of a risk associated with assigning $\mathbf{x}_n$ to class ω_j, and $\rho_{kj}(\cdot)$ is the misclassification measure of $\mathbf{x}_n$ defined by

$$\rho_{kj}(\mathbf{x}_n;\theta) = \frac{d_k(\mathbf{x}_n;\theta)-d_j(\mathbf{x}_n;\theta)}{d_k(\mathbf{x}_n;\theta)+d_j(\mathbf{x}_n;\theta)}, \qquad (8)$$

where $d_k(\cdot)$ and $d_j(\cdot)$ are dissimilarity-based discriminant functions of class ω_k and class ω_j, respectively, having positive values. The value of $\rho_{kj}(\cdot)$ ranges between -1 and 1, and leads to a correct (wrong) decision when it is negative (positive.) In other words, if $\rho_{kj}(\cdot)$ is positive, we can know that $\mathbf{x}_n$ falls into a wrong decision region. The function $f(\cdot)$ is a loss function with respect to the misclassification measure.

In this paper, the following semi-sigmoid function is used, because it resembles the hinge loss used in SVM:

$$f(\rho) = \begin{cases} \dfrac{1}{1+\exp(-\xi\rho)} & \text{for } \rho < 0, \\ (\xi\rho+2)/4 & \text{for } \rho \geq 0. \end{cases} \tag{9}$$

Since the slant of the loss function $\xi(>0)$ defines the margin between classes, we have to tune it with care (Sato, 2010).

In GLM, the classifier parameters are estimated by minimizing $L(\theta)$ based on gradient search. If we employ the steepest descent method, θ is updated iteratively ($t \leftarrow t+1$) as

$$\theta^{(t+1)} = \theta^{(t)} + \Delta\theta^{(t)},\ \ \Delta\theta^{(t)} = -\varepsilon(t) \left.\frac{\partial L(\theta)}{\partial\theta}\right|_{\theta=\theta^{(t)}} \tag{10}$$

until $\theta^{(t+1)} \simeq \theta^{(t)}$, where $\varepsilon(t) > 0$ and t denotes time. The differential of $L(\theta)$ can be derived as

$$\frac{\partial L(\theta)}{\partial\theta} = \frac{1}{N}\sum_{n=1}^{N}\sum_{k=1}^{K}\sum_{j\neq k}^{K} f'(\rho_{kj}(\mathbf{x}_n;\theta)) \times \frac{\partial\rho_{kj}(\mathbf{x}_n;\theta)}{\partial\theta} 1(t_n=\omega_k), \tag{11}$$

where $f'(\cdot)$ denotes the differential of $f(\cdot)$ as follows:

$$f'(\rho) = \begin{cases} \xi f(\rho)(1-f(\rho)) & \text{for } \rho < 0, \\ \xi/4 & \text{for } \rho \geq 0. \end{cases} \tag{12}$$

Here, let us use $(d+1)$-dimensional vectors as shown in Eq. (6). The differentiation of $\rho_{kj}(\cdot)$ in Eq. (11) depends on the class to which the prototype belongs. For $\mathbf{Y}_{ki}$ which belongs to the correct class ω_k,

$$\frac{\partial\rho_{kj}(\mathbf{X}_n;\theta)}{\partial\mathbf{Y}_{ki}} = \frac{-4d_j(\mathbf{X}_n;\theta)}{[d_k(\mathbf{X}_n;\theta)+d_j(\mathbf{X}_n;\theta)]^2} \times \left[\frac{d_k(\mathbf{X}_n;\theta)}{\|\mathbf{X}_n-\mathbf{Y}_{ki}\|^2}\right]^{1-p}(\mathbf{X}_n-\mathbf{Y}_{ki}), \tag{13}$$

and for $\mathbf{Y}_{ji}$ which belongs to the wrong class ω_j,

$$\frac{\partial\rho_{kj}(\mathbf{X}_n;\theta)}{\partial\mathbf{Y}_{ji}} = \frac{4d_k(\mathbf{X}_n;\theta)}{[d_k(\mathbf{X}_n;\theta)+d_j(\mathbf{X}_n;\theta)]^2} \times \left[\frac{d_j(\mathbf{X}_n;\theta)}{\|\mathbf{X}_n-\mathbf{Y}_{ji}\|^2}\right]^{1-p}(\mathbf{X}_n-\mathbf{Y}_{ji}). \tag{14}$$

Therefore, the update rules are obtained by substituting $\theta \leftarrow \mathbf{Y}_{ki}$ or $\theta \leftarrow \mathbf{Y}_{ji}$ in Eq. (10). In practice, the conjugate gradient method is used for optimizing the parameters. Note that we can obtain the prototypes and the biases which minimize the total loss, simultaneously, because $\mathbf{Y}_{ki}$ consists of $\mathbf{y}_{ki}$ and b_{ki}. In the experiments, the initial values of the prototypes were determined by using K-means algorithm for each class.

For the biases, the same value was assigned to each before learning, and they were normalized to satisfy the following constraint during learning, so that the biases range within a limit:

$$\sum_{k=1}^{K}\sum_{i=1}^{m_k} b_{ki}^2 = C^2. \tag{15}$$

The outline of the learning algorithm is summarized in Algorithm I.

Algorithm 1: Learning of the proposed method.

```
INPUT: $p$, $\xi$, $C$, $m_k$ $(k=1,\cdots,K)$
INITIALIZE: $\mathbf{y}_{ki} \leftarrow K$-means, $b_{ki} = C\big/\sqrt{\sum_k m_k}$
    $(i=1,\cdots,m_k; k=1,\cdots,K)$
while not converged do
    · $\theta^{(t+1)} = \theta^{(t)} + \Delta\theta^{(t)}$, where $L(\theta^{(t+1)}) \leq L(\theta^{(t)})$
    · $b_{ki} \leftarrow b_{ki} \times C\big/\sqrt{\sum_k\sum_i b_{ki}^2}$
      $(i=1,\cdots,m_k; k=1,\cdots,K)$
end while
OUTPUT: $\mathbf{y}_{ki}$, $b_{ki}$ $(i=1,\cdots,m_k; k=1,\cdots,K)$
```

3 EXPERIMENTS

To demonstrate the effectiveness of the proposed method, hereinafter referred to as ILM, two kinds of experiments were conducted: one was for two-dimensional artificial data, and the other was for the UCI machine learning repository.

3.1 Preliminary Experiments

Preliminary experiments for two-class two-dimensional artificial data were conducted to evaluate the effects of hyperparameters in ILM. Figure 2 shows examples of decision boundaries using different values for p in Eq. (6) and C in Eq. (15). Prototypes are shown by white circles, and the same value is assigned to every bias in each figure according to the constraint of Eq. (15). As the value of p becomes smaller, the decision boundaries tend to approximate Voronoi diagrams, because the decision rule becomes similar to the nearest neighbor rule. While, as the value of C becomes larger, they tend to form more simple shapes, because the Euclidean distance between a sample and a prototype becomes to have a larger bias.

Figure 3 shows the decision boundaries after learning by ILM. Cross and plus marks denote training samples for each class, and white circles denote prototypes. Each figure corresponds to that shown in the bottom row in Fig. 2, which was used as the

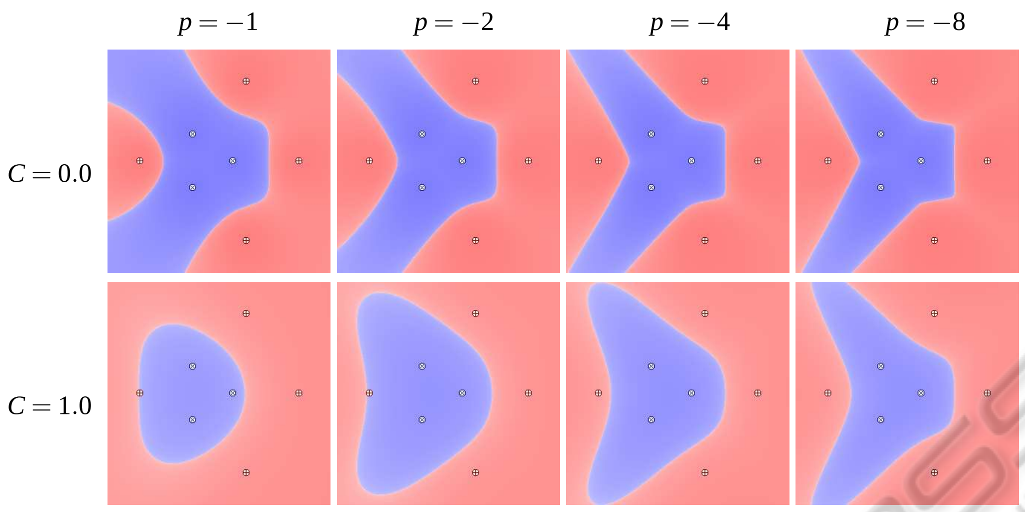

Figure 2: Examples of decision boundaries for two classes using different values for p and C in the proposed method. Prototypes are shown by white circles, and the same value is assigned to every bias in each figure according to the constraint of Eq. (15).

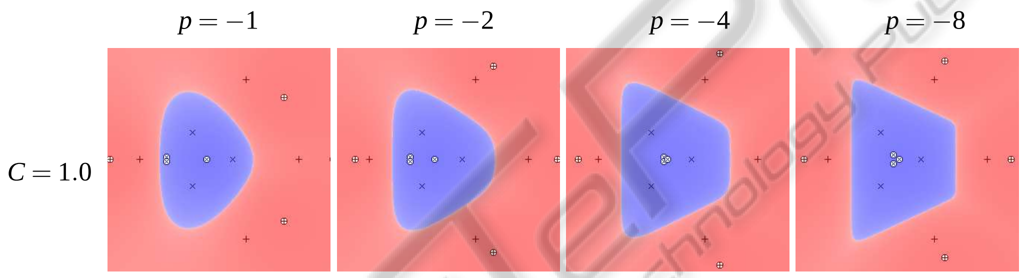

Figure 3: Decision boundaries after learning by the proposed method. Cross and plus marks denote training samples for each class, and white circles denote prototypes. Each figure corresponds to that shown in the bottom row in Fig. 2, which is the initial state before learning.

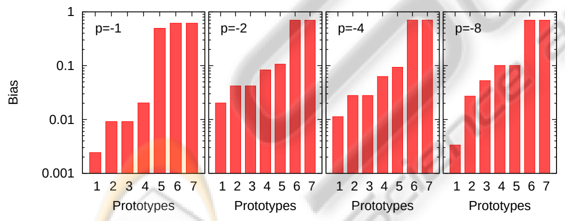

Figure 4: The values of biases after learning, sorted in ascending order. Each graph corresponds to the result shown in Fig. 3.

initial state before learning. Note that all training samples were used as initial values of prototypes in these experiments. As shown in the figures, the prototypes which belongs to the blue class become closer each other as p becomes smaller, and the decision boundary tends to maximize the margin between the two classes after learning. Therefore, ILM trained by GLM can be regarded as a maximal margin classifier as discussed in the literature (Sato, 2010). Figure 4 shows the values of biases after learning. Each graph corresponds to the result shown in Fig. 3. Since the bias is defined as the reciprocal of the weight for Lorentzian, the smaller biases are more important than larger ones, and the prototypes having larger biases can be removed. For example, the top three can be removed for $p=-1$, and the top two can be removed for $p<-1$. The decision boundaries after removal were almost the same as shown in Fig. 3. Since no difference was found to the eye, the figures after removal are not shown in this paper.

3.2 UCI Machine Learning Repository

3.2.1 Experimental Setup

Experiments for the UCI machine learning repository were conducted to evaluate the performance of ILM. Some data preprocessing was employed beforehand as used in the literatures (Meyer et al., 2003): 1) all records containing any missing values were removed from the dataset, 2) a binary coding scheme was used for handling categorical variables, and 3) all metric

Table 1: Experimental results for UCI machine learning repository. "Ins." denotes the number of instances after removing missing data, "Att." denotes the number of attributes excluding class attribute, and "#" denotes the number of support vectors or prototypes. The proposed method, referred to as ILM, was evaluated with different parameter sets: some parameters were fixed as (a) $C = 0$, $p = -2^8$, (b) $C = 0$, (c) $p = -1$, and the other parameters were varied.

Dataset	Ins.	Att.	SVM		ILM					
					(a)		(b)		(c)	
			Error (%)	#	Error (%)	#	Error (%)	#	Error (%)	#
BreastCancer	683	9	2.8	61	2.3	4	2.2	4	2.3	64
Cards	653	15	12.9	218	12.6	4	12.4	4	12.3	8
Heart1	297	13	14.1	139	13.5	8	13.5	8	13.1	2
HouseVotes84	435	16	3.7	102	3.7	2	3.4	8	3.2	4
Ionosphere	351	34	4.3	149	6.8	32	3.4	16	3.1	16
Liver	345	6	27.0	211	28.1	4	26.4	4	24.9	8
P.I. Diabetes	768	8	22.3	423	21.9	2	21.6	4	21.6	4
Sonar	208	60	11.5	154	8.7	32	7.7	32	11.1	8
Tictactoe	958	9	1.5	813	0.2	16	0.2	16	1.5	16

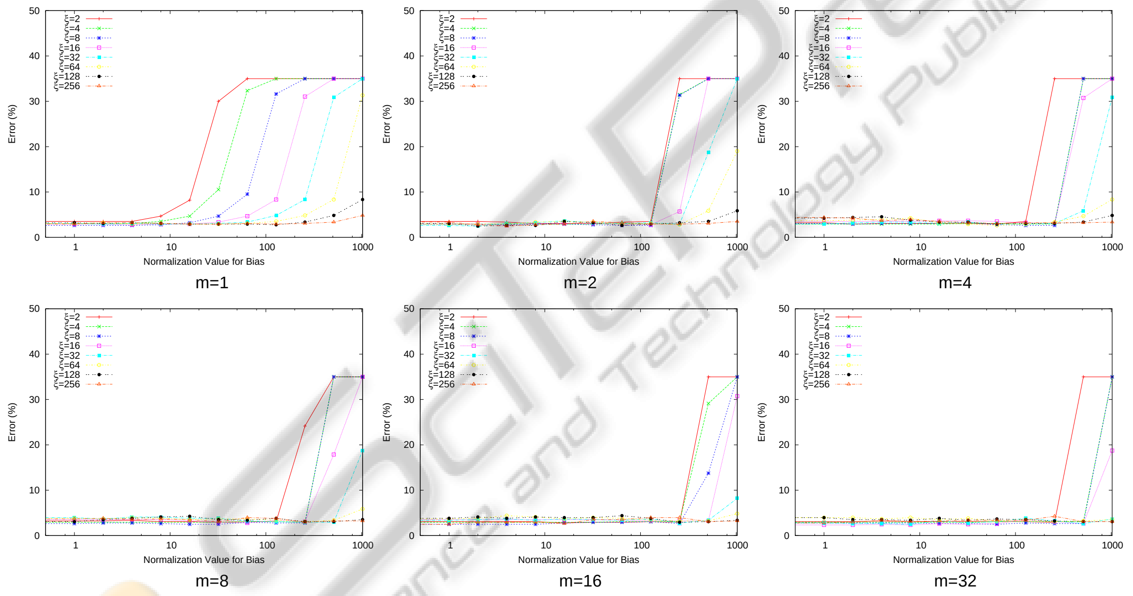

Figure 5: Error rates for BreastCancer in UCI machine learning repository by the proposed method with parameter set (c).

variables were scaled to zero mean and unit variance. Nine datasets[1] as shown in Table 1 were evaluated based on 10-fold cross validation. That is to say, the data were divided into 10 partitions, and 9 partitions were used for training and the rest was used for testing. This process was then repeated 10 times, and the obtained results were averaged to produce a single estimation. All of the datasets are two-class classification problems, so that we can compare the performance of ILM with SVM.

The proposed method has several hyperparameters: p and m_k in Eq. (6), C in Eq. (15), and ξ in Eq. (9). In the experiments, the same number given by m was used for m_k ($k = 1, \cdots, K$), so the performance was evaluated using different values for p, C, m and ξ. Specifically, $p = \{-2^i | i = 0, \cdots, 8\}$, $C = \{0, 2^i | i = 0, \cdots, 10\}$, $m = \{2^i | i = 0, \cdots, 5\}$, and $\xi = \{2^i | i = 1, \cdots, 8\}$ were used.

[1]The actual names in UCI machine learning repository are Breast Cancer Wisconsin (Original), Credit Approval, Heart Disease, Congressional Voting Records, Ionosphere, Liver Disorders, Pima Indians Diabetes, Connectionist Bench (Sonar, Mines vs. Rocks), Tic-Tac-Toe Endgame, respectively.

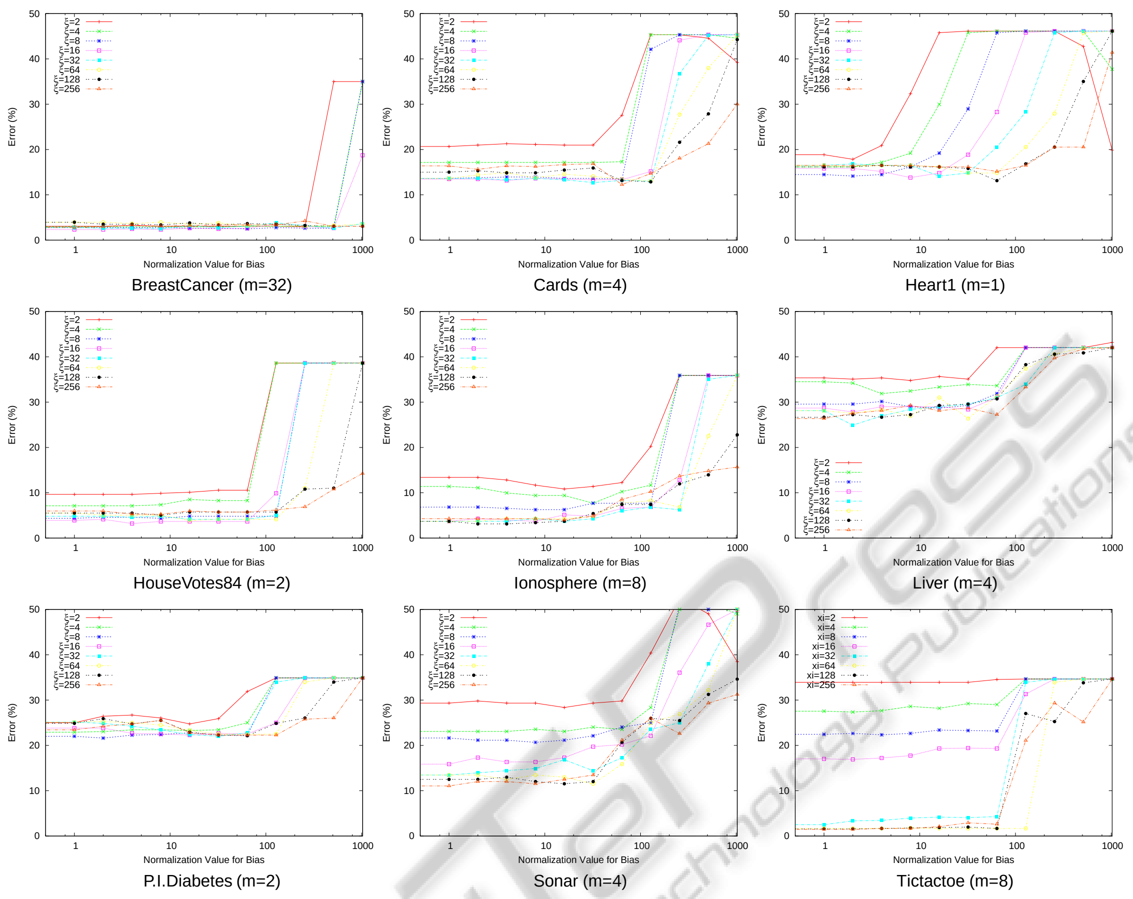

Figure 6: Error rates for UCI machine learning repository by the proposed method with parameter set (c).

3.2.2 Experiment I

Results by ILM are shown in Table 1. The lowest error rates are listed for different parameter set: some parameters were fixed as (a) $C = 0$, $p = -2^8$, (b) $C = 0$, (c) $p = -1$, and the other parameters were varied. Set (a) corresponds to GLVQ ($p \to -\infty$), set (b) corresponds to an extension of GLVQ using L_p norm-based mixture, and set (c) corresponds to the multiplicative inverse of the *linear* mixture of Lorentzian. Overall, the classification accuracy for set (c) is better than set (a) and set (b), except Sonar and Tictactoe. This means that the learning of biases is very effective to improve classification accuracy, while the L_p norm-based mixture is also important for some datasets. Figure 5 shows the error rates for BreastCancer by ILM with set (c). The horizontal axis denotes the value of C, and the error rates using different values for ξ are shown. It is noteworthy that the lowest error in each figure is not so different. This means that ILM does not seem to suffer from the overfitting even though many prototypes are used. Figure 6 shows similar graphs to Fig. 5, but the best result in various m was selected for each dataset. Note that the number of prototypes by ILM as shown in Table 1 is just twice as many as the selected m.

Similar experiments by SVM were conducted for comparison. Actually, SVM^{light} was used for evaluation. The hyperparameters in SVM are the trade-off C and RBF parameter γ in $K(\mathbf{x},\mathbf{y}) = \exp(-\gamma\|\mathbf{x}-\mathbf{y}\|^2)$. The performance was evaluated using different values for C and γ. Specifically, $C = \{2^i | i = -5, \cdots 12\}$ and $\gamma = \{2^i | i = -10, \cdots, 5\}$ were used. The lowest error rates and the number of the obtained support vectors are listed in Table 1. Clearly, the error rates by ILM are lower than those by SVM, while the number of prototypes is much less than that of support vectors. For BreastCancer, the number of prototypes by ILM (c) listed in the table is larger than that of support vectors, but the error rates stayed lower even though a fewer prototypes were used. For example, it was 2.6 (%) for two prototypes (i.e. $m = 1$). Figure 7 shows the results by SVM. These graphs are similar to Fig. 6, but the horizontal axis denotes $\log_2 C$, and the error rates using different values for RBF parameter γ are shown.

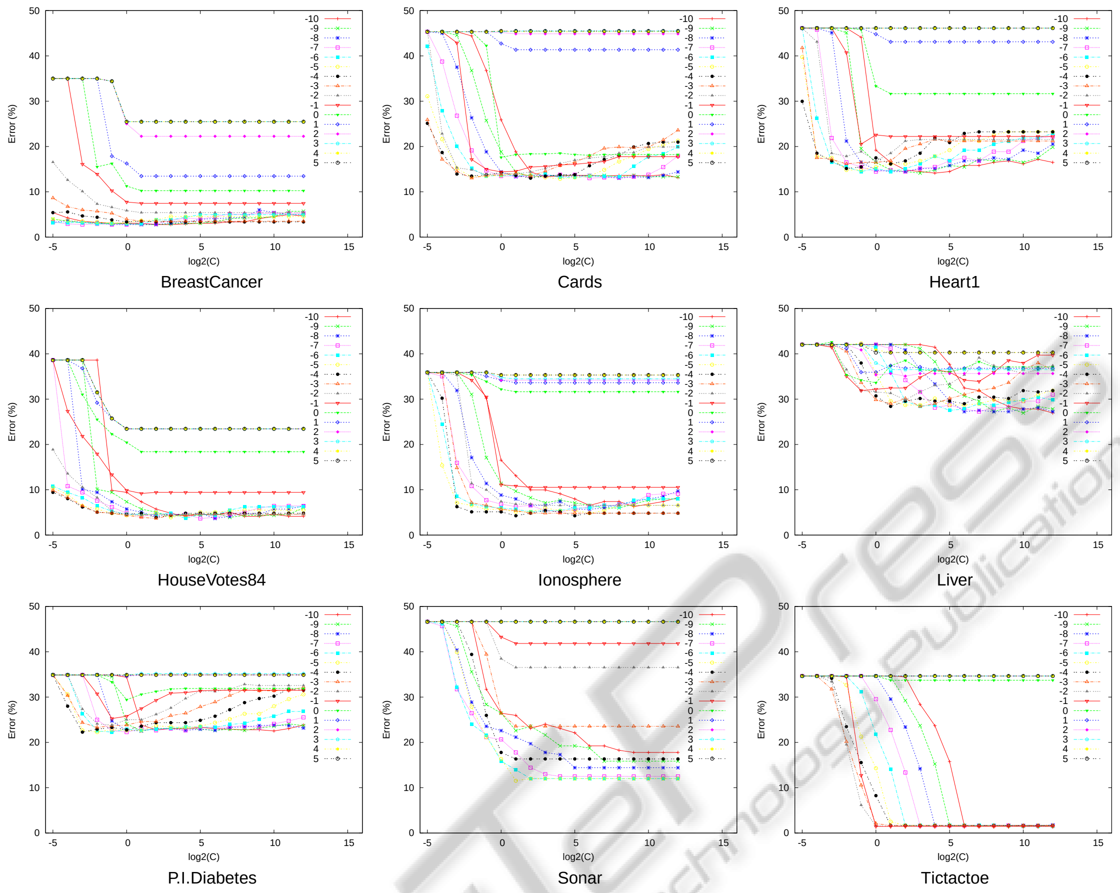

Figure 7: Error rates for UCI machine learning repository by SVM with RBF kernel.

3.2.3 Experiment II

The results in Table 1 show that the proposed method has capability to improve classification accuracy for any datasets used in the experiments. However, in practice, the hyperparameters should be estimated without testing data. Table 2 shows similar experimental results for UCI machine learning repository, but the hyperparameters in ILM and SVM were tuned with only training data through two-fold cross validation. Of course, the error rates are slightly worse than those in Table 1, but ILM retains almost the same as or better performance than SVM on the whole.

4 DISCUSSION

Experimental results for UCI machine learning repository revealed that ILM achieves almost the same as or higher classification accuracy than SVM with a much fewer prototypes than support vectors. The main advantage of ILM seems to train prototypes as well as biases (or weights), while only weights are trained in SVM. This may lead to reducing the number of prototypes for achieving good performance. Since the training of prototypes needs distance calculation at every iteration in gradient search, it results in time consumption as increasing the number of samples or prototypes, or the dimensionality. In addition, the cost function defined by Eq. (7) does not seem to ensure the convexity. However, as shown in the experiments, ILM gives a good performance for real datasets, so it can become one of the powerful methods for classification. It was also shown that prototypes having larger biases can be removed without degrading performance in the preliminary experiments, but this redundancy removal should be investigated further.

5 CONCLUSIONS

A novel distance-based classifier based on the multiplicative inverse of Lorentzian mixture was proposed, which can be regarded as a natural extension of the

Table 2: Experimental results for UCI machine learning repository. Hyperparameters were tuned with *only training data* through cross validation. "Ins." denotes the number of instances after removing missing data. "Att." denotes the number of attributes excluding class attribute, and "#" denotes the number of support vectors or prototypes. The proposed method, referred to as ILM, was evaluated with different parameter sets: some parameters were fixed as (a) $C = 0$, $p = -2^8$, (b) $C = 0$, (c) $p = -1$, and the other parameters were varied like in Table 1.

Dataset	Ins.	Att.	SVM		ILM					
					(a)		(b)		(c)	
			Error (%)	#	Error (%)	#	Error (%)	#	Error (%)	#
BreastCancer	683	9	2.8	66	2.5	32	2.6	64	2.6	64
Cards	653	15	13.6	189	13.2	8	13.9	8	14.6	4
Heart1	297	13	14.5	159	16.5	2	18.2	64	14.8	2
HouseVotes84	435	16	3.7	101	3.7	2	3.9	16	3.7	32
Ionosphere	351	34	4.8	148	8.3	8	3.7	32	4.3	64
Liver	345	6	28.1	203	28.1	4	27.5	4	26.3	4
P.I. Diabetes	768	8	22.3	418	22.5	2	22.3	32	21.9	64
Sonar	208	60	12.1	125	11.1	16	9.6	8	11.5	16
Tictactoe	958	9	1.7	349	1.7	2	0.9	16	1.7	2

conventional nearest neighbor rule. It was shown that prototypes and biases can be trained simultaneously by General Loss Minimization, which is a general framework for classifier design. The preliminary experiments raised the possibility that prototypes having larger biases can be removed without degrading performance, but this redundancy removal should be investigated further. Experimental results for UCI machine learning repository revealed that the proposed method achieves almost the same as or higher classification accuracy than SVM for all of nine datasets with a much fewer prototypes than support vectors. In future, the proposed method will be evaluated for various classification problems in real world.

REFERENCES

Blake, C. and Merz, C. (1998). *UCI repository of machine learning databases*. University of California, Irvine, Dept. of Information and Computer Sciences.

Cortes, C. and Vapnik, V. (1995). Support vector networks. *Machine Learning*, 20:273–297.

Crammer, K., Gilad-Bachrach, R., Navot, A., and Tishby, N. (2003). Margin analysis of the lvq algorithm. In *Advances in Neural Information Processing Systems*, volume 15, pages 462–469. MIT Press.

Giraud, B. G., Lapedes, A. S., Liu, L. C., and Lemm, J. C. (1995). Lorentzian neural nets. *Neural Networks*, 8(5):757–767.

Grbovic, M. and Vucetic, S. (2009). Learning vector quantization with adaptive prototype addition and removal. In *International Conference on Neural Networks*, pages 994–1001.

Hammer, B. and Villmann, T. (2002). Generalized relevance learning vector quantization. *Neural Networks*, 15(8–9):1059–1068.

Karayiannis, N. (1996). Weighted fuzzy learning vector quantization and weighted generalized fuzzy c-means algorithm. In *IEEE International Conference on Fuzzy Systems*, pages 773–779.

Kohonen, T. (1995). *Self-Organizing Maps*. Springer-Verlag.

Meyer, D., Leisch, F., and Hornik, K. (2003). The support vector machine under test. *Neurocomputing*, 55(1–2):169–186.

Qin, A. K. and Suganthan, P. N. (2004). A novel kernel prototype-based learning algorithm. In *International Conference on Pattern Recognition (ICPR)*, pages 621–624.

Sato, A. (1998). A formulation of learning vector quantization using a new misclassification measure. In *the 14th International Conference on Pattern Recognition*, volume 1, pages 322–325.

Sato, A. (2010). A new learning formulation for kernel classifier design. In *International Conference on Pattern Recognition (ICPR)*, pages 2897–2900.

Sato, A. and Yamada, K. (1996). Generalized learning vector quantization. In *Advances in Neural Information Processing Systems*, volume 8, pages 423–429. MIT Press.

Schneider, P., Biehl, M., and Hammer, B. (2009). Adaptive relevance matrices in learning vector quantization. *Neural Computation*, 21(12):3532–3561.

Villmann, T. and Haase, S. (2011). Divergence-based vector quantization. *Neural Computation*, 23(5):1343–1392.