IJDAR (2008) 11:53–65
DOI 10.1007/s10032-008-0069-1

ORIGINAL PAPER

Partial discriminative training for classification of overlapping classes in document analysis

Cheng-Lin Liu

Received: 15 April 2008 / Revised: 9 August 2008 / Accepted: 11 August 2008 / Published online: 23 September 2008

Abstract For character recognition in document analysis, some classes are closely overlapped but are not necessarily to be separated before contextual information is exploited. For classification of such overlapping classes, either discriminating between them or merging them into a metaclass does not satisfy. Merging the overlapping classes into a metaclass implies that within-metaclass substitution is considered as correct classification. For such classification problems, this paper proposes a partial discriminative training (PDT) scheme, in which, a training pattern of an overlapping class is used as a positive sample of its labeled class, and neither positive nor negative sample for its allied classes (those overlapping with the labeled class). In experiments of offline handwritten letter and online symbol recognition using various classifiers evaluated at metaclass level, the PDT scheme mostly outperforms ordinary discriminative training and merged metaclass classification.

Keywords Character recognition · Overlapping classes · Discriminative training · Partial discriminative training

1 Introduction

In document analysis applications involving character recognition, there is often the case that some patterns from different classes have very similar characteristics. In the feature space, such patterns of different classes correspond to co-incident or very close points, residing in an overlapping region. Such classes are called overlapping classes. A typical example problem is handwritten alphanumeric recognition, where some classes such as letters ‘O’, ‘o’ and numeral ‘0’ have identical shape, and it is neither possible nor necessary to separate them. Some other classes, such as upper-case letters ‘A’, ‘M’ and ‘N’, have many samples written in lower-case shapes (see Fig. 1). Thus, the upper-case letter and its corresponding lower-case have partially overlapping regions in the feature space. For such overlapping classes and all the pairs of upper-case/lower-case letters, it is not necessary to separate them at character level because it is easy to disambiguate according to the context, say, upper-case letters and lower-case letters rarely co-occur in the same word except the first letter.

There are generally two ways to deal with the overlapping classification problem. One way is to simply merge the overlapping classes into a metaclass[1] and ignore the boundary between these classes. In metaclass classification, the substitution between overlapping classes (within a metaclass) is considered as correct. In English letter recognition, the 52 letters can be merged into 26 case-insensitive classes. The 52-class letter classifier can also be evaluated at metaclass level by ignoring the substitution between the upper and lower cases of the same letter. Another way is to separate the overlapping classes by refining the classification boundary in the feature space, via extracting complementary or more discriminative features, using multi-stage classifier or combining multiple classifiers [1–4]. Particularly, using class verifiers [3] or pairwise classifiers [4] to verify or discriminate the top ranked classes output by a first-stage classifier can improve the classification accuracy. Neverthe-

This work was supported by the Hundred Talents Program of Chinese Academy of Sciences and the National Natural Science Foundation of China (NSFC) under Grants No. 60775004 and No. 60723005.

C.-L. Liu (✉)
National Laboratory of Pattern Recognition (NLPR),
Institute of Automation, Chinese Academy of Sciences,
95 Zongguancun East Road, 100190 Beijing,
People’s Republic of China
e-mail: liucl@nlpr.ia.ac.cn

[1] In this paper, a metaclass refers to a union of overlapping classes or a single class that is not merged.

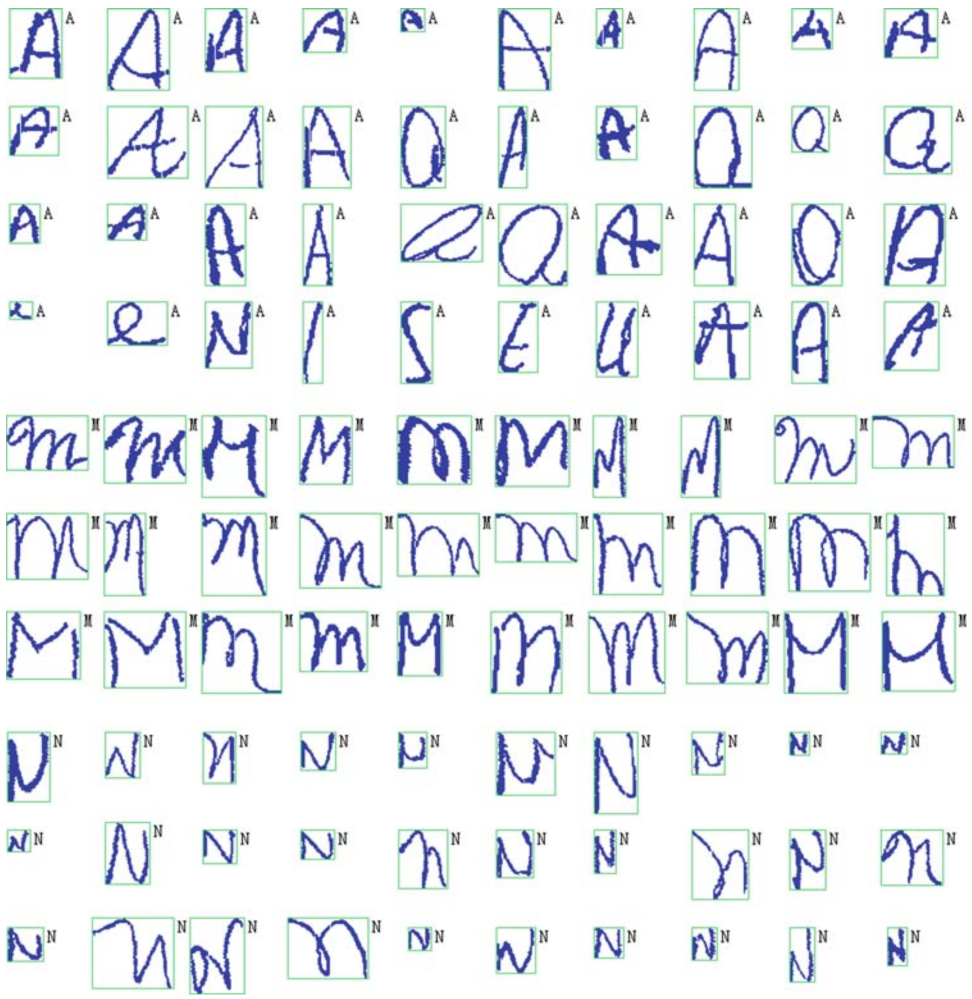
Fig. 1 Many samples of "AMN" are written in lower-case shape

less, the accuracy of overlapping classes separation is limited by the inherent feature space overlap, and in some cases, such attempt of discrimination is not necessary due to the availability of context.

The problem of overlapping classification is similar to multi-labeled classification [5,6], where a pattern may belong to multiple classes. If we enhance the class label of a pattern from an overlapping class such that it belongs to the labeled class as well as the allied classes (those overlapping with the labeled class), the overlapping classification problem becomes a multi-labeled one. In evaluation, the classification of a pattern to any of its allied classes is considered correct.

Unlike that many works attempt to separate overlapping classes, this work attempts to improve the separation between metaclasses while ignoring within-metaclass substitution. To achieve this goal, this paper proposes a new scheme for training neural networks, support vector machines (SVMs), and other discriminative classifiers. Ordinary discriminative training tends to complicate the boundary between overlapping classes, and consequently, may deteriorate the generalized classification performance. In the proposed partial discriminative training (PDT) scheme, the pattern of an overlapping class is used as a positive sample of its labeled class, and neither positive nor negative sample of the allied classes. By contrast, in ordinary discriminative training, the pattern of a class is used as negative sample of all the other classes, and in multi-labeled classification (cross-training [5]), the pattern is used as positive sample of its allied classes.

The effect of PDT to improve the separation between metaclasses is helpful in three scenarios:

- If some classes (such as letters 'O' and 'o') are not separable in isolation, discriminative training aiming to separate them will deteriorate the generalization performance.
- When overlapping or partially overlapping classes can be easily disambiguated by context, they are not necessarily separated at single-class level.
- When pairwise (within-metaclass) classifiers are used to discriminate overlapping classes, a complementary classifier focusing on the separation between metaclasses gives better combined performance.

To justify the effect of partial discriminative training, I evaluated various discriminative classifiers on two databases: the C-Cube handwritten letter database [7,8] and a subset of online handwritten symbols from the HANDS databases of TUAT [9]. The evaluated classifiers include neural networks, SVMs, nearest prototype classifiers [10], and discriminative learning quadratic discriminant function (DLQDF) [11]. The results show that the proposed PDT mostly outperforms cross-training, ordinary discriminative training, and merged metaclass classification when evaluated at metaclass level.

In the rest of this paper, Sect. 2 briefly reviews the related works; Sect. 3 describes the proposed PDT scheme and Sect. 4 describes the specific classifiers; Sect. 5 presents the experimental results, and Sect. 6 offers concluding remarks.

2 Related works

Statistical classifiers [12] and artificial neural networks [13] have been popularly applied to pattern recognition. A parametric statistical classifier, which estimates the probability density function of each class without considering the boundary between classes, is ready for classification of overlapping classes. The overlap between classes will not affect the estimation of parameters of parametric statistical classifiers. In training neural networks, the connecting weights are iteratively adjusted by optimizing an objective of minimum squared error or cross-entropy between class outputs and desired targets [13]. The overlap between classes will affect the complexity of decision boundary.

The support vector machine (SVM) [14] is an emerging classifier for solving difficult classification problems. Though multi-class SVM was proposed, binary SVM is popularly used, even for multi-class classification, by combining multiple binary classifiers for one-versus-all, pairwise, or other binary coded tasks. In any case, the binary SVM is trained (coefficients of kernel functions estimated) by maximizing the margin between two classes. The overlap between two classes also affects the trained SVM.

Both neural networks and SVMs, as discriminative classifiers, attempt to separate different classes in the feature space. If we ignore the substitution between overlapping classes, as for handwritten letter recognition, the overlapping classes can be merged into a metaclass and then the ordinary discriminative classifiers can be applied to this reduced class set problem. Koerich [15] designed several neural network classifiers for recognizing 52 letters, 26 upper-case letters, 26 lower-case letters and 26 metaclasses, respectively, and showed that the metaclass classifier outperforms the 52-class classifier (evaluated at metaclass level) and the combination of upper-case and lower-case classifiers.

In another work, Blumenstein et al. [16] merged 52 letters into 36 metaclasses: all upper-case letters except "ABDEGHNQRT" are merged with their lower-case letters, and use a neural network for 36-class classification. Camastra et al. [8] use one-versus-all SVM classifiers for classifying handwritten letters in 52 classes, 26 classes and adaptively merged classes according to the overlap degree between upper and lower cases. Using classifiers of 52 classes, 38 classes and 26 classes, they obtained test accuracies (evaluated at 26-metaclass level) of 89.20, 90.05 and 89.61%, respectively.

When ignoring the substitution between overlapping classes, the patterns of these overlapping classes can be viewed as multi-labeled. Multi-labeled classification is generally transformed to multiple binary classification tasks, and different methods differ in the way of attaching binary labels to training samples [6]. An effective method, cross-training [5], uses each multi-labeled sample as the positive sample of each class it belongs to and not as negative sample for any of the labeled classes. For example, if a sample belongs to classes 'A' and 'a', it is used as positive sample when training the binary classifiers for 'A' and 'a', and as negative samples for the binary classifiers of the other classes.

3 Partial discriminative training

The partial discriminative training (PDT) scheme is first proposed for neural networks [17], and then extended to other classifiers.

3.1 Training objectives of neural networks

Assume to classify a pattern (represented by a feature vector $\mathbf{x} \in \mathcal{R}^d$) to one of M classes $\{\omega_1, \ldots, \omega_M\}$. There are N training samples $(\mathbf{x}^n, c^n)$ (c^n is the class label of sample $\mathbf{x}^n$), $n = 1, \ldots, N$, for training a multi-class classifier. On an input pattern $\mathbf{x}$, the classifier outputs (sigmoidal) confidence values $y_k(\mathbf{x}, W)$ (W denotes the set of parameters) for classes $k = 1, \ldots, M$. Often, neural network weights are estimated by minimizing the squared error (SE):

$$\min_W \mathrm{SE} = \min_W \sum_{n=1}^{N} \sum_{k=1}^{M} \left[y_k(\mathbf{x}^n, W) - t_k^n \right]^2, \tag{1}$$

where t_k^n denotes the target output:

$$t_k^n = \delta(c^n, k) = \begin{cases} 1, & k = c^n, \\ 0, & \text{otherwise.} \end{cases} \tag{2}$$

The SE in Eq. (1) can be re-written as

$$\mathrm{SE} = \sum_{k=1}^{M} \sum_{n=1}^{N} \left[y_k(\mathbf{x}^n, W) - t_k^n \right]^2 = \sum_{k=1}^{M} E_k, \tag{3}$$

where $E_k = \sum_{n=1}^{N} [y_k(\mathbf{x}^n, W) - t_k^n]^2$ is the squared error of a binary classifier for class ω_k versus the others. Thus, the training of the multi-class neural network is equivalent to the training of multiple binary one-versus-all classifiers. Accordingly, the class output $y_k(\mathbf{x}, W)$ functions as the discriminant for separating class ω_k from the others.

The cross-entropy (CE) objective for neural networks can be similarly decomposed into multiple binary tasks:

$$\begin{aligned} \mathrm{CE} &= -\sum_{n=1}^{N}\sum_{k=1}^{M}\left[t_k^n \log y_k + (1-t_k^n)\log(1-y_k)\right]^2 \\ &= -\sum_{k=1}^{M}\sum_{n=1}^{N}\left[t_k^n \log y_k + (1-t_k^n)\log(1-y_k)\right]^2 \\ &= \sum_{k=1}^{M} \mathrm{CE}_k. \end{aligned} \quad (4)$$

For multi-labeled classification, each sample $\mathbf{x}^n$ is labeled to belong to a subset of classes C^n. For training neural networks in such case, the objective is the same as Eqs. (1), (3) or (4) except that the target output is changed to

$$t_k^n = \begin{cases} 1, & k \in C^n, \\ 0, & \text{otherwise.} \end{cases} \quad (5)$$

Due to the class modularity of objective functions SE and CE, for either single-labeled or multi-labeled classification, we can either train a multi-class classifier or train multiple binary one-versus-all classifiers.

To apply multi-labeled learning algorithms to overlapping classification, we can enhance the label of each sample with the allied classes (those overlapping with the labeled class) to convert the problem to be multi-labeled.

3.2 Partial discriminative training (PDT)

By ordinary discriminative training, which maximizes the separation between classes, the boundary between overlapping classes will be complicated. This may deteriorate the generalized classification performance and affect the boundary between metaclasses. On the other hand, simply merging the overlapping classes into a metaclass will complicate the distribution of the metaclass.

If ignoring the substitution between overlapping classes, the training objective of neural networks, either squared error (SE) or cross-entropy (CE), can be modified to ignore the error of the allied classes of each training sample. Denote the allied classes of ω_k as a set $\Lambda(k)$ (e.g., in alphanumeric recognition, the allied classes of 'O' are "o0"), the squared error of Eq. (3) is modified as

$$\mathrm{SE} = \sum_{k=1}^{M} \sum_{n=1, k \notin \Lambda(c^n)}^{N} \left[y_k(\mathbf{x}^n, W) - t_k^n\right]^2. \quad (6)$$

This implies, the training pattern $\mathbf{x}^n$ is not used as negative sample for the allied classes of c^n. Note that the relation of alliance is symmetric, i.e., $k \in \Lambda(c) \Leftrightarrow c \in \Lambda(k)$.

Excluding a training pattern from the negative samples of the allied classes of the label prevents the classifier from over-fitting the boundary between the labeled class and its allied classes (which are overlapping with the labeled class). Still, the number of classes remains unchanged (the structure of the multi-class classifier does not change), unlike in metaclass merging, the number of classes is reduced. Keeping the number of classes has the benefit that the classifier still outputs confidence scores to each of the overlapping classes. If two allied classes are partially overlapped, a sample from the un-overlapped region can be classified to its class unambiguously. By class merging, however, the boundary between allied classes is totally ignored.

3.2.1 Extension to other classifiers

The PDT scheme can be applied to all types of binary classifiers, with multiple binary classifiers combined to perform multi-class classification. For multi-class classification using one-versus-all SVMs, when training an SVM for a class ω_k, if ω_k is an allied class of a sample from a different class, this sample is excluded from the negative samples of ω_k. For example, in training the binary classifier for 'A' in letter recognition, the samples of 'a' are not used as negative samples.

Many classifiers can be trained by the minimum classification error (MCE) method [18], such as the nearest prototype classifier [10] and the discriminative learning quadratic discriminant function (DLQDF) [11]. The MCE method is to estimate the classifier parameters by minimizing the empirical classification error on a training dataset:

$$L_0 = \frac{1}{N}\sum_{n=1}^{N}\sum_{i=1}^{M} l_i(\mathbf{x}^n) I(\mathbf{x}^n \in \omega_i) = \frac{1}{N}\sum_{n=1}^{N} l_{c^n}(\mathbf{x}^n), \quad (7)$$

where $I(\cdot)$ is an indicator function. The classification error (loss) $l_c(\mathbf{x})$ on a training pattern $\mathbf{x}$ with genuine class ω_c is approximated by the sigmoidal function:

$$l_c(\mathbf{x}) = \sigma(d_c) = \frac{1}{1 + e^{-\xi d_c}}. \quad (8)$$

where ξ is a constant to control the smoothness of loss function. The misclassification measure $d_c(\mathbf{x})$ is usually approximated by the difference of discriminant function between the genuine class (positive class) and the closest rival class (negative class):

$$d_c(\mathbf{x}) = -f(\mathbf{x}, \omega_c) + \max_{i \neq c} f(\mathbf{x}, \omega_i), \quad (9)$$

where the discriminant function $f(\mathbf{x}, \omega_i)$ involves the classifier parameters, which are adjusted in minimizing the empirical loss by stochastic gradient descent.

To apply the PDT scheme to classifiers trained by MCE, the misclassification measure $d_c(\mathbf{x})$ is modified as

$$d_c(\mathbf{x}) = -f(\mathbf{x}, \omega_c) + \max_{i \neq c, i \notin \Lambda(c)} f(\mathbf{x}, \omega_i). \tag{10}$$

This is to say, the training pattern $\mathbf{x}$ is used as a positive sample of the labeled class ω_c and as a negative sample of the other classes except the allied classes of the label.

4 Specific classifiers

The proposed PDT scheme has been applied to five types of neural networks, SVMs with two types of kernel functions, a nearest prototype classifier by learning vector quantization (LVQ) [10], and a DLQDF classifier [11]. The neural classifiers are single-layer neural network (SLNN), multilayer perceptron (MLP), radial basis function (RBF) network [13], polynomial network classifier (PNC) [19,20], and class-specific feature polynomial classifier (CFPC) [21]. Two one-versus-all SVM classifiers use a polynomial kernel and an RBF kernel, respectively.

4.1 Neural networks

The neural classifiers have a common nature that each class output is the sigmoidal (logistic) function of the weighted sum of values of the previous layer. In SLNN, the input feature values are directly linked to the output layer:

$$y_k(\mathbf{x}, W) = \sigma\left(\sum_{i=1}^{d} w_{ki}x_i + w_{k0}\right), \quad k = 1, \ldots, M, \tag{11}$$

where w_{ki} denotes the connecting weights.

The MLP used in our experiments has one hidden layer, with class output computed by

$$y_k(\mathbf{x}, W) = \sigma\left[\sum_{j=1}^{N_h} w_{kj} \cdot \sigma\left(\mathbf{v}_j^T\mathbf{x} + v_{j0}\right) + w_{k0}\right], \tag{12}$$

where N_h denotes the number of hidden units, w_{kj} and v_{ji} denote the connecting weights of the output layer and the hidden layer, respectively.

The RBF network has one hidden layer of Gaussian kernel units:

$$h_j(\mathbf{x}) = \exp\left(-\frac{\|\mathbf{x} - \mu_j\|^2}{2\sigma_j^2}\right), \tag{13}$$

and the kernel values are linked to the sigmoidal outputs:

$$y_k(\mathbf{x}, W) = \sigma\left[\sum_{j=1}^{N_h} w_{kj}h_j(\mathbf{x}) + w_{k0}\right]. \tag{14}$$

The Gaussian centers and variance values are initialized by clustering and are optimized together with the weights by error minimization.

The PNC is a single-layer network with the polynomials of feature values as inputs. To avoid high complexity, I use a PNC with the binomial terms of the principal components [20]. Denoting the principal components in the m-dimensional $(m < d)$ subspace by $\mathbf{z}$, the class output is

$$y_k(\mathbf{x}, W) = \sigma\left[\sum_{i=1}^{m}\sum_{j=i}^{m} w_{kij}^{(2)} z_i(\mathbf{x}) z_j(\mathbf{x}) + \sum_{i=1}^{m} w_{ki}^{(1)} z_i(\mathbf{x}) + w_{k0}\right], \tag{15}$$

where $z_j(\mathbf{x})$ is the projection of $\mathbf{x}$ onto the j-th principal axis of the subspace.

The CFPC uses class-specific subspaces as well as the residuals of subspace projection:

$$y_k(\mathbf{x}, W) = \sigma\left[\sum_{i=1}^{m}\sum_{j=i}^{m} w_{kij}^{(2)} z_{ki}(\mathbf{x}) z_{kj}(\mathbf{x}) + \sum_{i=1}^{m} w_{ki}^{(1)} z_{ki}(\mathbf{x}) + w_k^e \epsilon_k(\mathbf{x}) + w_{k0}\right], \tag{16}$$

where $z_{kj}(\mathbf{x})$ is the projection of $\mathbf{x}$ onto the j-th principal axis of the subspace of ω_k, $\epsilon_k(\mathbf{x})$ is the projection residual.

For saving the computation of projection onto class-specific subspaces, the CFPC is trained class by class [21], i.e., the binary one-versus-all classifiers are trained separately. The other four neural networks are trained for all classes simultaneously. The weights of the neural networks are trained by minimizing the squared error criterion by stochastic gradient descent.

4.2 SVM classifiers

The one-versus-all SVM classifier has multiple binary SVMs each separating one class from the others. The discriminant function of a binary classifier is

$$f(\mathbf{x}) = \sum_{i=1}^{\ell} y_i\alpha_i \cdot k(\mathbf{x}, \mathbf{x}_i) + b, \tag{17}$$

where ℓ is the number of learning patterns, y_i is the target value of learning pattern $\mathbf{x}_i$ (+1/−1 for positive/negative class), b is a bias, and $k(\mathbf{x}, \mathbf{x}_i)$ is a kernel function which implicitly defines an expanded feature space:

$$k(\mathbf{x}, \mathbf{x}_i) = \Phi(\mathbf{x}) \cdot \Phi(\mathbf{x}_i). \tag{18}$$

Two types of kernels, polynomial and Gaussian (RBF), are frequently used. They are computed by

$$k(\mathbf{x}, \mathbf{x}_i, p) = (1 + \mathbf{x} \cdot \mathbf{x}_i)^p \tag{19}$$

and

$$k(\mathbf{x}, \mathbf{x}_i, \sigma^2) = \exp\left(-\frac{\|\mathbf{x} - \mathbf{x}_i\|^2}{2\sigma^2}\right), \tag{20}$$

respectively. The SVM classifiers using polynomial kernel and RBF kernel are called SVM-poly and SVM-rbf, respectively.

In our implementation of SVMs, the pattern vectors are appropriately scaled for the polynomial kernel, with the scaling factor estimated from the lengths of the sample vectors. For the Gaussian kernel, the kernel width σ^2 is estimated from the variance of the sample vectors. In partial discriminative training (PDT) of a binary SVM for class ω_k, the only change is to remove from negative samples the ones of the allied classes of ω_k.

4.3 LVQ and DLQDF

The PDT scheme is also applied to an LVQ classifier with prototypes trained by the MCE method [10] and a discriminative learning quadratic discriminant function (DLQDF) classifier with parameters trained by MCE [11]. During MCE training, the misclassification measure is modified as (10) for ignoring the boundary between overlapping classes.

The LVQ classifier uses equal number of prototypes per class, initialized by k-means clustering on the training data of one class. The prototypes of all classes are then optimized by minimizing the empirical classification error on training data. The discriminant function of a class ω_i is the negative of minimum distance (square Euclidean) to the prototypes of this class:

$$f(\mathbf{x}, \omega_i) = -\min_j \|\mathbf{x} - \mathbf{m}_{ij}\|^2, \tag{21}$$

where $\mathbf{m}_{ij}$ denotes the prototype vectors of class ω_i.

The DLQDF has the same form with the modified quadratic discriminant function (MQDF) of Kimura et al. [22]. The MQDF of a class represents a quadratic distance, whose negative is regarded as the discriminant function:

$$\begin{aligned}-f(\mathbf{x}, \omega_i) &= d_Q(\mathbf{x}, \omega_i)\\ &= \sum_{j=1}^{k} \frac{1}{\lambda_{ij}} \left[\phi_{ij}^T(\mathbf{x} - \mu_i)\right]^2\\ &\quad + \frac{1}{\delta_i}\left\{\|\mathbf{x} - \mu_i\|^2 - \sum_{j=1}^{k}\left[(\mathbf{x} - \mu_i)^T \phi_{ij}\right]^2\right\}\\ &\quad + \sum_{j=1}^{k} \log \lambda_{ij} + (d - k)\log \delta_i,\end{aligned} \tag{22}$$

where μ_i denotes the mean vector of class ω_i, ϕ_{ij} and λ_{ij} ($j = 1, \ldots, d$) are the eigenvectors and eigenvalues of the covariance matrix sorted in descending order of eigenvalues. Retaining k principal eigenvectors, the minor eigenvalues are replaced by a constant δ_i (which is further made class-independent in our implementation). The initial parameters of DLQDF are inherited from the MQDF, and are then optimized by minimizing the empirical classification error on training data.

To improve the generalization performance of MCE training and make the classifier to be resistant to non-character patterns [23], a regularization term is added to the empirical loss:

$$L_1 = \frac{1}{N}\sum_{n=1}^{N}[l_c(\mathbf{x}^n) + \alpha d(\mathbf{x}^n, \omega_c)], \tag{23}$$

where ω_c denotes the genuine class of ($\mathbf{x}^n$), $d(\cdot)$ stands for either the square Euclidean distance (LVQ classifier) or the quadratic distance (DLQDF). Empirically, the regularization coefficient α is set as $\alpha = 0.05/\text{var}$ for LVQ (var is the average within-class cluster variance) and $\alpha = 0.1/D_Q$ for DLQDF (D_Q is the average within-class quadratic distance estimated with the initial parameters) [23].

5 Experimental results

The proposed partial discriminative training (PDT) scheme has been evaluated with various classifiers on a public handwritten letter database C-Cube [7,8][2] and on online handwritten symbols from the HANDS databases of TUAT [9].

5.1 Results on C-Cube letter database

The C-Cube database contains 57,293 samples of 52 English letters, partitioned into 38,160 training samples and 19,133 test samples. The samples, segmented from handwritten words, are very cursive and the number of samples per class is seriously imbalanced. In addition to the confusion between upper-case and lower-case letters, the confusion between different case-insensitive letters is also considerable, such as the letters in Fig. 2. By k-NN classification based on vector quantization, the authors ordered the upper/lower case overlap degree of each letter for merging the cases of selected letters. The database provides binary images as well as extracted feature values (34D) of the samples [24]. I experimented on the given 34D features as well as 200D gradient direction histogram features (extracted by moment normalization, 8-direction decomposition and 5×5 sampling [25,26]).

[2] Downloadable at http://ccc.idiap.ch/.

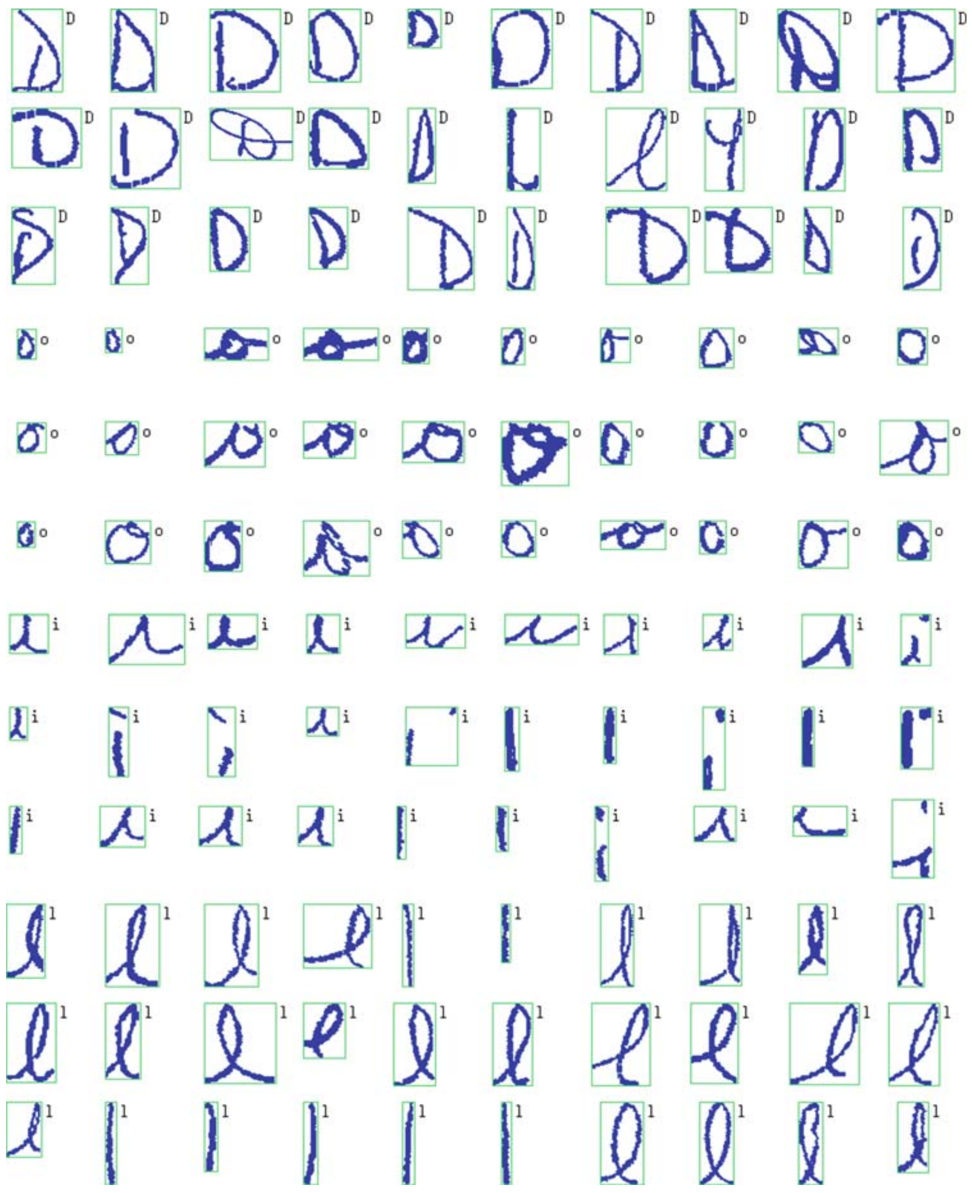

Fig. 2 Some samples of ’D’ confuse with ’o’, some samples of ’l’ confuse with ’i’ and ’e’

Three numbers of classes are considered as in [7,8]: 52 case-sensitive letters, 38 classes by merging the upper/lower cases of 14 letters (“CXOWYZMKJUNFVA”), and 26 case-insensitive letters. In the cases of 38 classes and 26 letters, each merged upper-case letter is allied with its lower-case and vice versa. The configuration parameters of some classifiers (number of hidden units N_h for MLP and RBF, principal subspace dimensionality m for PNC and CFPC, number of prototypes n per class for LVQ, number of principal eigenvectors k for DLQDF) were empirically set as in Table 1. PCA was not performed for PNC on the original 34D features ($m = 34$). These configurations are not guaranteed to be optimal but perform reasonably well. The SVM-poly uses a fourth order polynomial kernel on uniformly re-scaled pattern vectors, and the SVM-rbf uses an RBF kernel with kernel width fixed at 0.5 times the average within-class variance [25].

Table 1 Classifier configurations on original and gradient features of C-Cube database

Feature	MLP	RBF	PNC	CFPC	LVQ	DLQDF
	N_h	N_h	m	m	n	k
Original (34D)	100	150	34	25	5	15
Gradient (200D)	100	150	70	40	5	40

First, four multi-class neural networks (SLNN, MLP, RBF, and PNC) were trained with three training schemes optimizing the squared error: ordinary discriminative training, PDT, and cross-training (enhancing the label of each sample with its allied classes). The error rates on test samples are shown in Table 2, where each row gives the accuracies evaluated at a number of metaclasses (52, 38 or 26, within-metaclass substitution is ignored). By ordinary discriminative training, the number of classes can be reduced by class merging, whereas by PDT and cross-training, the number of classes remains unchanged but the samples are attached allied classes or multi-labeled. Each classifier can be evaluated at a reduced number of classes by ignoring within-metaclass substitution. At each row (evaluated at a number of metaclasses), the lowest error rate is highlighted in bold face, and the error rates of merged metaclass training and PDT are boxed for comparison. The error rates of metaclass classification of all-class discriminative training are underlined for comparison with merged metaclass training and PDT.

The difference of error rates is measured by the confidence of significance z [27]. For two error rates p_1 and p_2 evaluated on n test samples, $z = (p_2 - p_1)/\sigma$, where $\sigma = \sqrt{2p(1-p)/n}$ (p is the average error rate). If $|z| > 1.96$, two error rates are judged to be significantly different with confidence higher than 0.95. The z values for PDT compared to all-class discriminative training (DT) and PDT compared to class merging (CM) are shown in Table 2. When $|z| > 1.96$, a "(+)" or "(−)" is attached to indicate significant superiority or inferiority, otherwise the error rates are comparable.

In Table 2, it is evident that all-class discriminative training (third column) yields the lowest error rate for 52-class classification. This is reasonable because all the classes are aimed to be separated in this case, while PDT ignores the separation between allied classes. When evaluated at reduced number of classes, however, merged metaclass training (fourth and fifth columns) and PDT (sixth and seventh columns) may give lower error rates than all-class training. In seven of eight cases (two class numbers 38 and 26 combined with four classifiers), PDT gives lower error rates than both all-class training and merged metaclass training. The inferior performance of cross-training can be explained that the framework of multi-labeled classification does not match the problem of overlapping classification.

Ordinary discriminative training and PDT were then applied to three one-versus-all classifiers (CFPC, SVM-poly and SVM-rbf) and LVQ and DLQDF classifiers. The error rates are shown in Table 3. Again, all-class discriminative training gives the lowest error rates for 52-class classification. When evaluated at metaclass level, both merged metaclass training and PDT gives lower error rates than all-class training. For the CFPC, LVQ and DLQDF classifiers, PDT also outperforms merged metaclass training. For the SVM classifiers, merged metaclass training gives the lowest error rates of metaclass classification, but the error rates of PDT are closely competitive.

The error rates of discriminative training and PDT using 200D gradient features are shown in Table 4. We can see that the error rates are lower than the ones of same classifiers in Tables 2 and 3. This is because the gradient direction histogram feature is more discriminative than the original 34D features given by [24]. Again, the error rate of PDT is compared with all-class discriminative training and class merging at metaclass level. In all the 18 cases of metaclass classification (two metaclass numbers combined with nine classifiers), PDT gives lower error rates than all-class training ($z > 0$). In 15 of 18 cases, PDT also outperforms the merged metaclass training, and in four cases, PDT outperforms significantly.

Table 5 gives the statistics of comparing PDT with all-class discriminative training and class merging. In Tables 2, 3 and 4, a $z > 1.96$ indicates a win of PDT, $z < 1.96$ indicates a

Table 2 Error rates (%) of neural networks on C-Cube letter database (34D feature)

Classifier	#Class	Discriminative			PDT		Cross-train		z to DT	z to CM
		52	38	26	38	26	38	26		
	52	**33.85**			34.51	34.77	47.34	65.82		
SLNN	38	$\underline{28.07}$	$\boxed{29.60}$		$\boxed{\mathbf{27.25}}$	27.53	29.62	47.34	1.79	5.10(+)
	26	$\underline{27.47}$	29.06	$\boxed{32.06}$	26.64	$\boxed{\mathbf{26.56}}$	29.08	32.02	2.00(+)	11.82(+)
	52	**21.03**			21.79	21.80	39.17	54.54		
MLP	38	$\underline{15.36}$	$\boxed{15.02}$		$\boxed{\mathbf{14.58}}$	14.94	15.63	31.92	2.14(+)	1.21
	26	$\underline{15.00}$	14.66	$\boxed{15.58}$	**14.21**	$\boxed{14.43}$	15.19	16.38	1.57	3.15(+)
	52	**21.94**			22.29	22.25	38.80	55.16		
RBF	38	$\underline{16.24}$	$\boxed{\mathbf{15.63}}$		$\boxed{15.75}$	15.69	16.00	33.32	1.31	−0.32
	26	$\underline{15.84}$	15.28	$\boxed{15.72}$	15.39	$\boxed{\mathbf{15.19}}$	15.64	16.30	1.76	1.43
	52	**18.91**			19.33	19.36	36.66	56.66		
PNC	38	$\underline{13.13}$	$\boxed{12.89}$		$\boxed{\mathbf{12.38}}$	12.39	13.20	34.59	2.20(+)	1.50
	26	$\underline{12.82}$	12.58	$\boxed{13.71}$	12.07	$\boxed{\mathbf{11.97}}$	12.90	14.35	2.52(+)	5.09(+)

Each row gives the error rates evaluated at a number of metaclasses, and each column corresponds to a number of metaclasses in training. Fourth and fifth columns correspond to merged metaclass training. Tenth and eleventh columns give the significance level of PDT compared to all-class discriminative training (DT) and PDT compared to class merging (CM)

Table 3 Error rates (%) of other classifiers on C-Cube letter database (34D feature)

Classifier	#Class	Discriminative			PDT		z to DT	z to CM
		52	38	26	38	26		
	52	**18.93**			19.27	19.29		
CFPC	38	$\underline{13.24}$	$\boxed{13.35}$		$\boxed{\mathbf{12.78}}$	12.79	1.34	1.65
	26	$\underline{12.91}$	13.01	$\boxed{14.14}$	12.44	$\boxed{\mathbf{12.30}}$	1.80	5.31(+)
	52	**17.81**			18.03	18.12		
SVM-poly	38	$\underline{11.87}$	$\boxed{\mathbf{11.34}}$		$\boxed{11.39}$	11.49	1.46	−0.15
	26	$\underline{11.59}$	11.06	$\boxed{\mathbf{10.97}}$	11.12	$\boxed{11.01}$	1.79	−0.13
	52	**17.35**			17.58	17.65		
SVM-rbf	38	$\underline{11.27}$	$\boxed{\mathbf{10.88}}$		$\boxed{10.90}$	10.98	1.15	−0.06
	26	$\underline{11.00}$	10.67	$\boxed{\mathbf{10.57}}$	10.61	$\boxed{10.60}$	1.26	−0.10
	52	**22.44**			23.58	23.99		
LVQ	38	$\underline{16.81}$	$\boxed{16.36}$		$\boxed{\mathbf{15.78}}$	16.10	2.73(+)	1.54
	26	$\underline{16.42}$	15.97	$\boxed{17.22}$	15.37	$\boxed{\mathbf{15.36}}$	2.84(+)	4.93(+)
	52	**20.75**			21.75	22.56		
DLQDF	38	$\underline{14.66}$	$\boxed{14.30}$		$\boxed{\mathbf{14.05}}$	14.73	1.70	0.70
	26	$\underline{14.22}$	13.95	$\boxed{14.52}$	**13.64**	$\boxed{13.93}$	0.82	1.65

lose, and $|z| < 1.96$ indicates a tie (insignificant difference of error rates). Overall, when evaluated at metaclass level, PDT outperforms all-class discriminative training significantly in 12 of 36 cases, and performs comparably in the remaining cases. Comparing PDT with class merging, PDT outperforms significantly in 10 of 36 cases, and performs comparably in the remaining cases.

On both two types of features and all classifier structures, PDT yields lower metaclass-level error rates than all-class discriminative training. This confirms that discriminative training attempting to separate overlapping classes is detrimental to the between-metaclass separation. Comparing PDT to merged metaclass training, the contrast of performance turns out to be depending on feature type and classifier structure, yet the metaclass-level error rate of PDT is lower in most cases. PDT outperforms merged metaclass training significantly on both feature types only with three classifiers: SLNN, MLP and LVQ. This is because the merged metaclass has more complicated distribution than a single class, while some classifiers cannot sufficiently model classes of complex distribution.

Table 6 shows the confusion matrices of a selected classifier (DLQDF on 200D gradient feature) trained by all-class discriminative training and PDT. Since it is space-consuming to list confusion numbers between all 26 letters, here I only give the confusion numbers between 13 letters that are frequently confused. We can see that by PDT, though the correct rates of two letters (‘a’ and ‘v’) are considerably lowered, the correct rates of five letters (“cmnou”) are apparently raised. Particularly, the confusions between ‘c’–‘e’, ‘l’–‘s’, ‘m’–‘n’, ‘o’–‘s’, and ‘u’–‘v’ are apparently reduced. This is because PDT improves the separation between different letters while sacrificing the separation between upper/lower cases of each letter.

5.2 Results on TUAT-HANDS symbols

The HANDS databases of TUAT (Tokyo University of Agriculture and Technology) [9] include two databases (Kuchibue and Nakayosi) of online handwritten Japanese characters and symbols. The number of symbol classes (including alphanumeric, hiragana, Katakana, punctuation marks, mathematic symbols, and others) is 380. The symbols have many confusing classes that have identical or similar shapes, and so, they cause appreciable percentage of misrecognition [21]. Based on the fact that many symbols are used in different contexts and are not necessarily separated in isolation, partial discriminative training (PDT) is suitable for this case to design classifiers focusing on the separation between metaclasses.

Conforming to previous works, the symbol samples of the Nakayosi database (163 writers) are used for training classifiers, and the samples of the Kuchibue database (120 writers) for testing. Because each writer wrote texts of natural language and the characters missing in the texts were written only once, the sample numbers of different characters are remarkably variable. My experiments used at most 300 training/test samples for each symbol. If a symbol has more than 300 samples, 300 of them were selected randomly. The resulting training set contains 163 or 300 samples per symbol, and the test set contains 120, 240 or 300 samples per symbol. The total numbers of training samples and test samples are 80,298 and 79,380, respectively.

Table 4 Error rates (%) on C-Cube letter database (200D feature)

Classifier	#Class	Discriminative training			Partial training		z to DT	z to CM
		52	38	26	38	26		
	52	**21.90**			22.2	22.27		
SLNN	38	$\underline{15.61}$	$\boxed{15.84}$		$\boxed{\mathbf{14.52}}$	14.61	2.98(+)	3.60(+)
	26	$\underline{15.13}$	15.34	$\boxed{17.01}$	14.04	$\boxed{\mathbf{13.91}}$	3.39(+)	8.39(+)
	52	**18.57**			18.84	19.13		
MLP	38	$\underline{12.32}$	$\boxed{11.74}$		$\boxed{\mathbf{11.22}}$	11.44	3.34(+)	1.60
	26	$\underline{11.91}$	11.41	$\boxed{12.54}$	10.86	$\boxed{\mathbf{10.65}}$	3.90(+)	5.77(+)
	52	**16.51**			16.7	16.67		
RBF	38	$\underline{10.30}$	$\boxed{\mathbf{9.58}}$		$\boxed{9.72}$	9.71	1.89	−0.46
	26	$\underline{9.95}$	**9.25**	$\boxed{9.27}$	9.39	$\boxed{9.28}$	2.22(+)	−0.03
	52	**13.67**			14.16	14.30		
PNC	38	$\underline{7.41}$	$\boxed{7.23}$		$\boxed{\mathbf{7.14}}$	7.29	1.02	0.34
	26	$\underline{7.14}$	6.97	$\boxed{7.31}$	**6.88**	$\boxed{6.91}$	0.88	1.52
	52	**13.48**			14.21	14.35		
CFPC	38	$\underline{7.33}$	$\boxed{7.23}$		$\boxed{\mathbf{6.91}}$	7.05	1.60	1.20
	26	$\underline{7.05}$	6.95	$\boxed{7.19}$	**6.64**	$\boxed{6.74}$	1.20	1.73
	52	**14.08**			14.16	14.34		
SVM-poly	38	$\underline{7.43}$	$\boxed{7.12}$		$\boxed{\mathbf{6.91}}$	7.17	1.97(+)	0.80
	26	$\underline{7.15}$	6.85	$\boxed{6.85}$	**6.72**	$\boxed{6.77}$	1.46	0.31
	52	13.78			**13.69**	13.75		
SVM-rbf	38	$\underline{7.16}$	$\boxed{7.00}$		$\boxed{\mathbf{6.91}}$	6.97	0.96	0.35
	26	$\underline{6.89}$	6.73	$\boxed{6.82}$	**6.65**	$\boxed{\mathbf{6.65}}$	0.93	0.66
	52	**15.91**			16.97	16.84		
LVQ	38	$\underline{9.50}$	$\boxed{9.57}$		$\boxed{\mathbf{9.15}}$	9.25	1.18	1.41
	26	$\underline{9.21}$	9.26	$\boxed{9.72}$	**8.83**	$\boxed{\mathbf{8.83}}$	1.30	3.00(+)
	52	**14.44**			15.26	15.38		
DLQDF	38	$\underline{7.89}$	$\boxed{\mathbf{7.39}}$		$\boxed{7.52}$	7.73	1.36	−0.48
	26	$\underline{7.56}$	**7.07**	$\boxed{7.51}$	7.20	$\boxed{7.15}$	1.54	1.35

Table 5 Counts of win/tie/lose of PDT versus all-class discriminative training (DT) and class merging (CM)

Feature	PDT versus DT			PDT versus CM		
	#Win	#Tie	#Lose	#Win	#Tie	#Lose
Original (34D)	6	12	0	6	12	0
Gradient (200D)	6	12	0	4	14	0

According to the similarity between symbol shapes, I merged the 380 symbols into 315 metaclasses. The merged classes are shown in Fig. 3, where the symbols in each group are merged into a metaclass, or allied with each other in partial discriminative training.

For symbol recognition, each symbol is represented by a 288D feature vector extracted by moment normalization, trajectory direction decomposition and 6×6 sampling [28]. Because of the large number of classes (380 or 315), not all the classifiers described in Sect. 4 are suitable. The MLP and RBF networks were not used in this case because to determine the appropriate number of hidden units is nontrial and the training process is time-consuming and prone to local optima. SVM classifiers with polynomial and RBF kernels were not used since for large number of classes and large sample set, the training process is very time-consuming and the resulting classifier has very high operational complexity due to the large number of support vectors. Instead, I tested an SVM classifier with linear kernel (SVM-linear), which has low complexity in both training and operation. The SVM-linear is a linear classifier like the SLNN, but they are trained under different criteria. For the other classifiers, the PNC uses 80 subspace features, the CFPC uses 40D principal subspace per class, the LVQ classifier has 5 prototypes per class, and the DLQDF uses 40 principal eigenvectors per class.

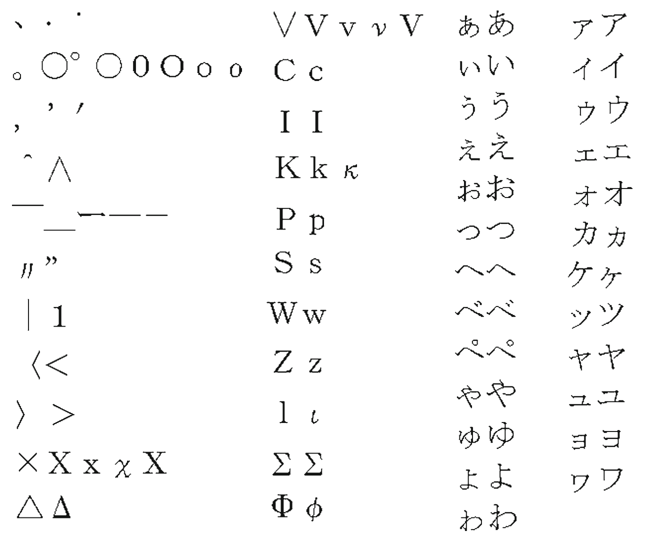
Fig. 3 Merged classes of online handwritten symbols in TUAT-HANDS database

Table 7 Error rates (%) on online symbols of TUAT-HANDS database using generative classifiers

Classifier	#Class	380	315
	380	27.23	
LDF	315	16.12	16.74
	380	19.91	
MQDF	315	8.38	8.95

Each row gives the error rates evaluated at a number of metaclasses, and each column corresponds to a number of metaclasses in training

First, I tested two parametric statistical classifiers (generative classifiers): linear discriminant function (LDF) and MQDF. The LDF assumes Gaussian densities and equal covariance matrices for all classes. Table 7 shows the error rates evaluated at 380 classes and at 315 metaclasses. We can see that when evaluated at metaclass level, merging classes in training does not benefit the classification accuracy because the class distribution becomes complicated. In contrast, the all-class generative classifier gives higher accuracy of metaclass classification.

Table 6 Confusion numbers between 13 letters (DLQDF on 200D feature)

	a	c	e	h	i	l	m	n	o	r	s	u	v
DLQDF, all-class discriminative training													
a	1,489	0	7	5	1	3	0	3	21	5	9	3	1
c	3	737	21	0	5	10	0	0	2	5	2	0	0
e	0	15	1,213	0	10	86	0	1	0	2	7	0	1
h	0	0	1	495	0	9	0	14	0	7	2	0	0
i	0	8	5	2	662	65	0	0	1	37	7	7	0
l	1	4	85	9	54	1,346	0	1	0	5	21	0	0
m	0	0	0	9	1	1	732	53	0	1	0	3	0
n	4	0	0	16	4	2	52	1,728	0	22	0	2	4
o	17	1	6	1	0	0	1	1	1,328	2	14	2	3
r	5	2	6	1	14	2	1	7	0	1,151	9	1	6
s	6	2	1	1	1	1	0	4	5	15	951	0	0
u	2	0	0	0	0	0	7	14	0	2	0	701	29
v	1	0	0	0	0	0	0	4	0	2	1	11	204
DLQDF, PDT													
a	1,479	0	7	7	1	3	0	4	32	9	7	3	0
c	3	**755**	13	0	3	9	0	0	0	1	2	0	0
e	0	18	1,216	0	10	83	0	2	1	2	4	0	1
h	0	0	1	496	0	9	1	15	0	6	2	0	0
i	0	9	4	2	661	67	1	0	1	38	6	6	0
l	1	6	89	8	55	1,345	0	2	1	6	12	0	0
m	0	0	0	9	1	1	**753**	36	0	0	0	2	0
n	2	0	0	14	3	2	47	**1,741**	0	19	0	2	4
o	12	2	5	1	0	0	1	1	**1,353**	2	5	5	2
r	4	1	7	1	12	2	2	8	2	1,154	6	1	5
s	6	1	1	1	0	1	0	4	6	16	951	0	0
u	2	0	0	1	0	0	6	12	1	1	0	**708**	20
v	1	0	0	0	0	0	0	5	0	2	1	15	199

Table 8 Error rates (%) on online symbols of TUAT-HANDS database using discriminative classifiers

Classifier	#Class	Discriminative		PDT	z to DT	z to CM
		380	315	315		
SLNN	380	**22.54**		22.86		
	315	$\underline{11.78}$	$\boxed{10.99}$	$\boxed{\mathbf{10.83}}$	5.98(+)	1.02
SVM-linear	380	22.19		**21.90**		
	315	$\underline{12.15}$	$\boxed{10.75}$	$\boxed{\mathbf{10.67}}$	9.27(+)	0.52
PNC	380	**19.76**		19.92		
	315	$\underline{8.47}$	$\boxed{\mathbf{7.88}}$	$\boxed{\mathbf{7.88}}$	4.29(+)	0
CFPC	380	**19.46**		20.04		
	315	$\underline{8.53}$	$\boxed{\mathbf{7.69}}$	$\boxed{\mathbf{7.69}}$	6.13(+)	0
LVQ	380	**20.67**		21.88		
	315	$\underline{9.53}$	$\boxed{\mathbf{8.91}}$	$\boxed{8.98}$	3.78(+)	−0.49
DLQDF	380	20.27		**20.05**		
	315	$\underline{8.44}$	$\boxed{\mathbf{7.72}}$	$\boxed{7.99}$	3.26(+)	−2.00(−)

Fourth column corresponds to merged metaclass training. Sixth and seventh columns give the significance level of PDT compared to all-class discriminative training (DT) and PDT compared to class merging (CM)

The error rates of six discriminative classifiers on the online symbols of TUAT-HANDS database are shown in Table 8. It is interesting to see that the DLQDF, as the discriminative version of the MQDF, gives lower error rates of metaclass classification than the MQDF when it is trained on merged classes or by PDT. The DLQDF trained by all-class discriminative training (DT), however, does not outperform the generative MQDF. For all the six classifiers, merged metaclass training and PDT give lower error rates of 315-metaclass classification than all-class DT. PDT outperforms all-class DT significantly in all the six cases. Comparing PDT and merged metaclass training, PDT performs comparably in five cases and loses in one case.

Unlike in handwritten letter recognition where the upper/lower cases of most letters are partially overlapped, the overlap between allied classes of online symbols is so heavy that the shapes are almost identical, especially the pairs of hiragana/Katakana characters. This is why ordinary discriminative training with class merging performs very well. Anyway, PDT performs comparably and the trained classifier provides confidence scores to all classes, unlike the merged metaclass classifier that totally ignores the separation between allied classes.

6 Conclusion

This paper proposed a partial discriminative training (PDT) scheme for classification of overlapping classes. It is applicable to all types of binary one-versus-all classifiers, including neural networks, SVM classifiers, and the classifiers trained by the MCE method. The rationale of PDT is to ignore the difference between overlapping classes in training so as to improve the separation between metaclasses. Experiments in offline handwritten letter recognition and online handwritten symbol recognition show that when evaluated at metaclass level, the PDT scheme mostly outperforms ordinary all-class discriminative training. Compared to merged metaclass training, the PDT gives higher or comparable accuracies at metaclass level and provides more informative confidence scores.

We plan to apply PDT to character string recognition and multiple classifier systems. In character string recognition, since linguistic context and geometric context are available for disambiguating confusing characters, to make the classifier focus on discrimination between metaclasses while sacrificing the separation between confusing classes will be meaningful. In multiple classifier systems, combining a classifier focusing on between-metaclass discrimination and some others focusing on within-metaclass discrimination will be beneficial.

References

1. Lu, B.-L., Ito, M.: Task decomposition and modular combination based on class relations: a modular neural network for pattern classification. IEEE Trans. Neural Netw. **10**(5), 1244–1256 (1999)
2. Podolak, I.T.: Hierarchical classifier with overlapping class groups. Expert Syst. Appl. **34**(1), 673–682 (2008)
3. Zhou, J., Krzyzak, A., Suen, C.Y.: Verification—a method of enhancing the recognizers of isolated and touching handwritten numerals. Pattern Recognit. **35**(5), 1179–1189 (2002)
4. Bellili, A., Gilloux, M., Gallinari, P.: An MLP–SVM combination architecture for offline handwritten digit recognition: reduction of recognition errors by support vector machines rejection mechanisms. Int. J. Document Anal. Recognit. **5**(4), 244–252 (2003)
5. Boutell, M.R., Luo, J., Shen, X., Browm, C.M.: Learning multi-label scene classification. Pattern Recognit. **37**(9), 1757–1771 (2004)

6. Tsoumakas, G., Katakis, I.: Multi-label classification: an overview. Int. J. Data Warehousing Min. **3**(3), 1–13 (2007)
7. Camastra, F., Spinetti, M., Vinciarelli, A.: Offline cursive character challenge: a new benchmark for machine learning and pattern recognition algorithms. In: Proceedings of 18th ICPR, Hong Kong, vol. 2, pp. 913–916 (2006)
8. Camastra, F.: A SVM-based cursive character recognizer. Pattern Recognit. **40**(12), 3721–3727 (2007)
9. Nakagawa, M., Matsumoto, K.: Collection of on-line handwritten Japanese character pattern databases and their analysis. Int. J. Document Anal. Recognit. **7**(1), 69–81 (2004)
10. Liu, C.-L., Nakagawa, M.: Evaluation of prototype learning algorithms for nearest neighbor classifier in application to handwritten character recognition. Pattern Recognit. **34**(3), 601–615 (2001)
11. Liu, C.-L., Sako, H., Fujisawa, H.: Discriminative learning quadratic discriminant function for handwriting recognition. IEEE Trans. Neural Netw. **15**(2), 430–444 (2004)
12. Fukunaga, K.: Introduction to Statistical Pattern Recognition, 2nd edn. Academic Press, New York (1990)
13. Bishop, C.M.: Neural Networks for Pattern Recognition. Oxford University Press, New York (1995)
14. Burges, C.J.C.: A tutorial on support vector machines for pattern recognition. Knowl. Discov. Data Min. **2**(2), 1–43 (1998)
15. Koerich, A.L.: Unconstrained handwritten character recognition using different classification strategies. In: Gori, M., Marinai, S. (eds.) Proceedings of 1st IAPR Workshop on Artificial Neural Networks in Pattern Recognition, pp. 52–56 (2003)
16. Blumenstein, M., Liu, X.Y., Verma, B.: An investigation of the modified direction feature for cursive character recognition. Pattern Recognit. **40**(2), 376–388 (2007)
17. Liu, C.-L.: Partial discriminative training of neural networks for classification of overlapping classes. In: Artificial Neural Networks in Pattern Recognition: Third IAPR Workshop, Paris, France, LNAI, vol. 5064, pp. 137–146. Springer, Heidelberg (2008)
18. Juang, B.-H., Katagiri, S.: Discriminative learning for minimum error classification. IEEE Trans. Signal Process. **40**(12), 3043–3054 (1992)
19. Shürmann, J.: Pattern Classification: A Unified View of Statistical and Neural Approaches. Wiley Interscience, New York (1996)
20. Kreßel, U., Schürmann, J.: Pattern classification techniques based on function approximation. In: Bunke, H., Wang, P.S.P. (eds.) Handbook of Character Recognition and Document Image Analysis, pp. 49–78. World Scientific, Singapore (1997)
21. Liu, C.-L., Sako, H.: Class-specific feature polynomial classifier for pattern classification and its application to handwritten numeral recognition. Pattern Recognit. **39**(4), 669–681 (2006)
22. Kimura, F., Takashina, K., Tsuruoka, S., Miyake, Y.: Modified quadratic discriminant functions and the application to Chinese character recognition. IEEE Trans. Pattern Anal. Mach. Intell. **9**(1), 149–153 (1987)
23. Liu, C.-L., Sako, H., Fujisawa, H.: Effects of classifier structures and training regimes on integrated segmentation and recognition of handwritten numeral strings. IEEE Trans. Pattern Anal. Mach. Intell. **26**(11), 1395–1407 (2004)
24. Camastra, F., Vinciarelli, A.: Cursive character recognition by learning vector quantization. Pattern Recognit. Lett. **22**(6-7), 625–629 (2001)
25. Liu, C.-L., Nakashima, K., Sako, H., Fujisawa, H.: Handwritten digit recognition: benchmarking of state-of-the-art techniques. Pattern Recognit. **36**(10), 2271–2285 (2003)
26. Liu, C.-L., Nakashima, K., Sako, H., Fujisawa, H.: Handwritten digit recognition: investigation of normalization and feature extraction techniques. Pattern Recognit. **37**(2), 265–279 (2004)
27. Dietterich, T.G.: Approximate statistical test for comparing supervised classification learning algorithms. Neural Comput. **7**(10), 1895–1924 (1998)
28. Liu, C.-L., Zhou, X.-D.: Online Japanese character recognition using trajectory-based normalization and direction feature extraction. In: Proceedings of 10th IWFHR, La Baule, France, pp. 217–222 (2006)