PROTOTYPE-BASED MINIMUM ERROR CLASSIFIER FOR HANDWRITTEN DIGITS RECOGNITION

Roongroj Nopsuwanchai

Computer Laboratory,
University of Cambridge,
Cambridge, CB3 0FD, UK
Email: *rn225@cam.ac.uk*

Alain Biem

IBM T. J. Watson Research Center,
P.O. Box 218, Yorktown Heights,
New York, 10598, USA
Email: *biem@us.ibm.com*

ABSTRACT

This paper describes an application of the Prototype-based Minimum Error Classification (PBMEC) to the offline recognition of handwritten digits. The PBMEC uses a set of prototypes to represent each digit along with an L_ν-norm of distances as the decoding scheme. Optimization of the system is based on the Minimum Classification Error (MCE) criterion. In this paper, we introduce a new clustering criterion adapted to the PBMEC structure that minimizes an L_ν-norm-based distortion measure. The new clustering algorithm can generate a smaller number of prototypes than the standard k-means with no loss in accuracy. It is also shown that the PBMEC trained with the MCE can achieve over 42% improvement from the baseline k-means process and requires only 28Kb storage to match the performance of a 1.46MB-sized k-NN classifier.

1. INTRODUCTION

This paper describes an application of the Prototype-based Minimum Error Classifier (PBMEC) [1] to the problem of handwritten digits recognition. This paper is motivated by the desire to achieve a lightweight but highly performing recognition system.

The primary goal of a pattern classification system is to achieve the minimum error rate possible. The minimization of the probability of error is guaranteed by the Bayes theoretic decision, which advocates choosing the class that yields the maximum posterior probability of the class, given the incoming pattern. However, incomplete knowledge of these probabilities precludes the direct application of the Bayes decision scheme. Instead, the system designer has to rely on indirect methods, such as assuming the form of the distribution and estimating the parameters of the distribution from available training data. However, this parametric approach is a risky enterprise as the correct form of the distribution is rarely known. An alternative solution is to bypass the parametric approach and use the entire training set as prototypic examples to directly infer the Bayes decision boundaries: an incoming pattern inherits the label of its closest neighbors in the training set. This is the nearest-neighbor decision rule widely known under the k-nearest neighbor (k-NN) instantiation [2]. In the k-NN classifier, an incoming pattern inherits the label of the most representative category among the k closest patterns in the training set.

The k-NN classifier is one the oldest classification techniques widely utilized in various tasks, ranging from data-mining, probability estimation, and pattern classification. Its popularity is due to solid theoretical results, i.e. the error rate of the k-NN classifier converges to the optimal error rate generated by the Bayes classifier when both the number of training data N and the value of k increase, and the ratio k/N approaches zero [2]. This confirms that the k-NN classifier can asymptotically approximate the optimal classification decision. Even with a finite number of data, the k-NN classifier has demonstrated very good performance on a wide range of pattern classification tasks. However, because the k-NN uses an entire training set as prototypes, it requires a large storage space, thereby unlikely to be deployed in very small-footprint systems.

Prototype-based systems attempt to improve the storage requirement of the k-NN classifier while keeping comparable performance. These approaches can be categorized into two groups: prototype-selecting techniques and prototype-generating techniques. Prototype selection chooses a subset of the training set by pruning the data based on some heuristic to maintain the performance of the system. The drop in accuracy is usually balanced by an optimized distance metric [3]. In contrast, the prototype-generating scheme relies on a data clustering algorithm to compress the data into fewer representatives, based on some distortion measures. As distortion measures are not related to classification accuracy, these prototypes can be further reshaped by discriminative training methodologies such as Learning Vector Quantization (LVQ) and other discriminative training schemes [4].

The PBMEC, originally proposed for speech recognition tasks, appears to be a valid alternative to the standard prototype-based k-NN classifier in terms of performance, speed, and memory requirement. The PBMEC is a prototype-generating classifier. However, unlike most nearest-neighbor classifiers, all prototypes in the PBMEC belonging to the same classification category are participants in the class-membership decision, through the use of discrimination function. The discrimination function uses an L_ν-norm of distances, which weights the participation of each prototype in the decision process. Tuning can be done to emulate the nearest-neighbor classifier or to implement a more general decision scheme. The PBMEC operates two steps. First, a data clustering technique, typically the k-means algorithm, is run to reduce the data into a set of representatives. Second, the Minimum Classi-

fication Error criterion (MCE) [5] is applied to re-adjust the prototypes with the aim of achieving minimum error status. The MCE criterion has been applied with great success to various speech-related tasks [1] and to offline and online handwriting recognition [6].

In this paper, we show the potential of the PBMEC to handwriting recognition and introduce a new clustering algorithm that is more adapted to the functional form of the PBMEC decision rule. The results show that 1) the nearest-neighbor decision rule is not the optimal one in regard to performance; 2) the new clustering algorithm outperforms the standard k-means; and 3) the MCE criterion realizes a much smaller and more efficient recognizer than the k-NN classifier.

2. PROTOTYPE-BASED MINIMUM ERROR CLASSIFIER FOR CHARACTER RECOGNITION

The PBMEC is a distance-based classifier that uses the notion of discrimination function to quantify the membership of an incoming pattern to a specific category, where each category is represented by a set of prototypes. The original version of the PBMEC was used to classify variable-length patterns [1] and embeded a Dynamic Programming procedure. A version of PBMEC that deals with a fixed-length pattern as proposed in [7] is used in this paper.

2.1. Recognizer structure

We are given a finite set of P character-classes, C_j, where $1 \leq j \leq P$. The character-class C_j is represented by a the parameter set $\lambda_j = \{\mathbf{r}_{j,m}, \Sigma_{j,m}\}$, where $1 \leq m \leq M$. $\mathbf{r}_{j,m}$ represents the m-th prototype of character-class C_j and $\Sigma_{j,m}$ is an adjustable positive-definite matrix, typically initialized to be the covariance matrix relative to the corresponding prototype. The number of prototypes per character is M. We represent the overall system's parameters as Λ.

From a bitmap image, a feature extraction process generates a feature vector $\mathbf{x}$. Each character-class C_j uses a distance-based discrimination function $g_j(\mathbf{x}, \Lambda)$ which indicates the degree that $\mathbf{x}$ belongs to the class and is defined as

$$g_j(\mathbf{x}, \Lambda) = \left\{ \sum_{m=1}^{M} ||\mathbf{x} - \mathbf{r}_{j,m}||^{-\nu} \right\}^{-\frac{1}{\nu}} \quad (1)$$

where ν is a positive constant. The discrimination function is an L_ν-norm of distances between the pattern $\mathbf{x}$ and the prototypes $\mathbf{r}_{j,m}$. Any suitable distance can be used. In this paper, we use the Mahalonobis (or weighted) distance defined as

$$||\mathbf{x} - \mathbf{r}_{j,m}|| = (\mathbf{x} - \mathbf{r}_{j,m})^T (\Sigma_{j,m})^{-1} (\mathbf{x} - \mathbf{r}_{j,m}) \quad (2)$$

which reduces to the Euclidean distance when $\Sigma_{j,m}$ is the unity matrix.

The final decoding is done by the simple classification rule:

$$\mathbf{x} \in C_i \quad \text{if} \quad i = \underset{j}{\operatorname{argmin}}\, g_j(\mathbf{x}, \Lambda) \quad (3)$$

which chooses the character that yields the smallest value of the discrimination function.

Eq. (1) shows that the contribution of each prototype to the discrimination function is modulated by the parameter ν with the property that

$$\lim_{\nu \to \infty} g_j(\mathbf{x}, \Lambda) = \min_m ||\mathbf{x} - \mathbf{r}_{j,m}||. \quad (4)$$

For ν large, the discrimination function is the smallest of the distances, meaning that the pattern $\mathbf{x}$ inherits the label of the closest prototype. This is the nearest-neighbor rule. Thus, the tuning of ν enables the emulation of various decision rules, making the PBMEC a generalization of various classifier structures.

3. PBMEC TRAINING

PBMEC training is done in two stages: an initialization stage which performs a clustering algorithm on a class-by-class basis to generate a set of prototypes representing each class, followed by the MCE training process that re-adjusts the prototypes for better performance.

3.1. L_ν-norm-based k-means initialization

The initialization process is typically done by the k-means clustering algorithm which minimizes the total distortion over the training data representing the class.

Let $\{\mathbf{x}_1^j, \mathbf{x}_2^j, ..., \mathbf{x}_N^j\}$ be the set of training data of class C_j. For a pre-chosen number of clusters M, the k-means algorithm iteratively adjusts the centroid of each cluster, by a within-cluster averaging process which minimizes the distortion D defined as

$$D = \sum_{n=1}^{N} \min_m ||\mathbf{x}_n^j - \mathbf{r}_{j,m}||. \quad (5)$$

For each pattern $\mathbf{x}$ there is a unique prototype yielding the minimum distance, meaning that the k-means clustering generates a disjoint set of clusters with well defined boundaries.

The k-means algorithm has been the standard clustering algorithm in the PBMEC until now. There is, however, an obvious inconsistency: the metric used in Eq. (5) is not matched to the L_ν-norm metric used in the PBMEC classification process. This creates a discrepancy between the k-means-based training phase and the testing phase, especially when training is done by k-means clustering.

In light of this, we introduce an L_ν-norm-based k-means algorithm, referred to as L_ν-k-means, which is adapted to the PBMEC situation. The L_ν-k-means minimizes the following distortion measure:

$$D_\nu = \sum_{n=1}^{N} \left\{ \sum_{m=1}^{M} ||\mathbf{x}_n^j - \mathbf{r}_{j,m}||^{-\nu} \right\}^{-\frac{1}{\nu}}. \quad (6)$$

D_ν uses a similar metric to one used in the PBMEC's discrimination function and is a generalization of the nearest-neighbor-based metric used in the classical k-means algorithm. It operates in a manner similar to fuzzy k-means algorithms as each input pattern is not assigned to a single cluster defined by a single centroid, but distances to all centroids are taken into account in the computation of the distortion. It is clear from Eq. 4 that

$$\lim_{\nu \to \infty} D_\nu = D \quad (7)$$

meaning that for large ν, the L_ν-k-means algorithm is equivalent to the classical k-means algorithm.

3.2. MCE Training

From the prototypes generated by the L_ν-k-means clustering process, the MCE training is run to generate better performing prototypes. The MCE algorithm [5] adjusts the parameters of the system so as to minimize an expected loss $\mathcal{L}(\Lambda)$ which is a reflection of the classification error and is defined as $\mathcal{L}(\Lambda) = E_{\mathbf{x}}(\ell_i(\mathbf{x}, \Lambda))$. The loss $\ell_i(\mathbf{x}, \Lambda)$ is a function of a misclassification measure $d_i(\mathbf{x}, \Lambda)$ for training sample $\mathbf{x} \in C_i$ given the set of parameters Λ:

$$\ell_i(\mathbf{x}, \Lambda) = \sigma(d_i(\mathbf{x}, \Lambda)) \tag{8}$$

where $\sigma(\cdot)$ is a smooth approximation of the step-wise 0-1 function, which is equal to one for positive values and zero otherwise. The misclassification measure $d_i(\mathbf{x}, \Lambda)$ is defined as:

$$d_i(\mathbf{x}, \Lambda) = g_i(\mathbf{x}, \Lambda) - \left[\frac{1}{P-1} \sum_{j=1, j \neq i}^{P} g_j(\mathbf{x}, \Lambda)^{-\eta} \right]^{-\frac{1}{\eta}} \tag{9}$$

where η is a positive number which controls the degree of contribution of competing categories. The misclassification measure emulates the classification decision in scalar value: a positive sign implies an incorrect decision and a negative sign implies a correct decision. For a very large η, the misclassification error is approximated to a difference in score between the best incorrect category and the true category.

$\mathcal{L}(\Lambda)$ can be minimized by a gradient-based optimization techniques such as Generalized Probabilistic Descent (GPD) [5] or the approximated Newton-based optimization known as Quickprop [8].

4. EXPERIMENTAL RESULTS

We performed several experiments to evaluate the performance of the PBMEC training algorithms. The baseline systems are the standard k-means clustering technique and the popular k-NN classifier. The task chosen for the evaluation is the classification of offline unconstrained-style handwritten digits. The database used is the UNIPEN online handwriting database [9]. We generated character images out of the given dynamic information of handwritten strokes. The training set is composed of 15,953 handwritten digits from Train-R01/V07 category $1a$ subset. Testing is done on the corresponding DevTest-R01/V02 $1a$ subset which contains 8,598 samples. The 24-dimensional Discrete Cosine Transform (DCT) feature vector was extracted from the 22×30-pixel character image.

4.1. L_ν-norm contribution

The first experiment was carried out to determine the contribution of the L_ν-norm-based metric to the performance of the system or equivalently which value of ν yields the optimum performance. This study was done using the standard k-means training process based on the Euclidean distance measure at the fine-grained level. Various classifier structures were generated by varying the number of prototypes per class.

Fig. 1 shows the performance for various classifier structures. It is clear that the optimal L_ν-norm distance (obtained for ν between 2 and 5) is independent of the number of prototypes per class. Keeping in mind that a large ν implements the nearest-neighbor rule, the results suggest that the classical use of the nearest-neighbor rule, which represents each category by its closest representative is not optimal. The degree of fuzziness introduced by the use of the relatively small value of ν in the L_ν-norm of distances yields better performance.

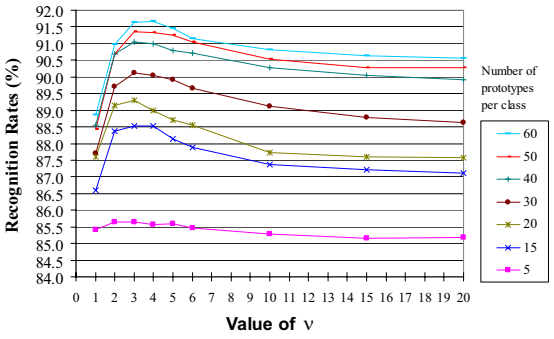

Fig. 1. Recognition rate on the test set versus the value of the value of ν for various classifier structures

4.2. k-means and L_ν-norm-based k-means

Further experiments were performed to evaluate the performance of the L_ν-k-means clustering algorithm. The algorithm was run from the k-means-generated prototypes using a gradient-based minimization scheme. Figure 2 shows the best performance on the test set, using the Euclidean distance at the fine-grained level, versus the number of prototypes per class.

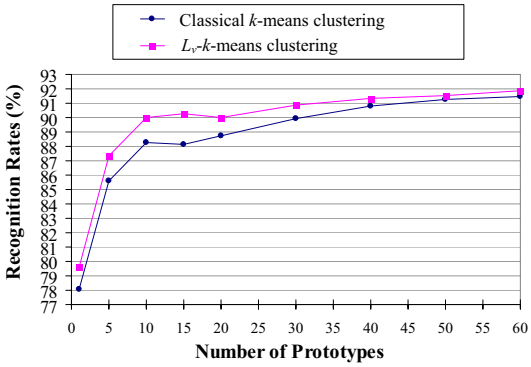

Fig. 2. Comparison of the k-means algorithm and the proposed L_ν-k-means in terms of performance.

From the results, it is obvious that the L_ν-k-means algorithm performs better than classical k-means algorithm, especially when using a fewer number of prototypes per class: 15% relative improvement from the k-means configuration are achieved in the context of 15 prototypes per class but only 5% relative improve-

ment is realized when using 60 prototypes per class. Also the L_ν-k-means can generate a more compact system: a system with 10 prototypes per class generated by the k-means-L_ν algorithm gives similar performance to a system with 30 prototypes per class generated with the classical k-means algorithm.

4.3. Performance of MCE training

A series of experiments was carried out to evaluate the performance of MCE training. Subsequent goals were to study the effect of different distances on MCE and to determine which parameter training yields the best performance. When using the Mahalonobis distance, the parameters that can be trained by MCE are the prototypes $\mathbf{r}_{j,m}$ and covariance matrices $\Sigma_{j,m}$ (or their inverse which are referred to as weights) whereas only the prototypes are trained when using the Euclidean distance. These parameters were initialized by the L_ν-k-means clustering process. Parameter updates in MCE training were carried out for 80 iterations using the Quickprop algorithm. The value of η was set to 20 and empirically determined.

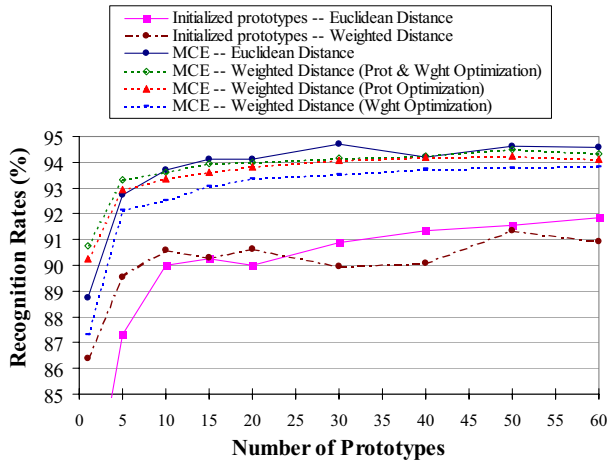

Fig. 3. Performance of MCE on the test set using various distance measures and various parameters training.

The results in Fig. 3 show the best performance of MCE training on the test set versus the number of prototypes for various distance measures and various configurations. When using Mahalonobis distance, even though prototype optimization is more effective than weight optimization, the best result is achieved when both prototypes and weights are updated at the same time. In MCE training, the Euclidean distance is still more efficient than weighted distance. The best result achieved by MCE training gives the recognition rate of 94.69%, which is equivalent to a 42% relative error reduction from the L_ν-k-means's performance.

A k-NN classifier on the same training and testing data achieves 94.39%. This result was achieved at the cost of a 1.46MB required storage for the prototypes whereas the PBMEC only requires 28Kb. Clearly, the PBMEC training approach together with our proposed techniques are obviously more powerful and compact and more suitable to be deployed in the system with limited resources.

5. CONCLUSIONS

This paper has described an application of the Prototype-based Minimum Classification Error classifier to the task of recognizing unconstrained-style handwritten digits. Experimental investigation shows that the nearest-neighbor rule is not optimal. A new clustering criterion, introduced to generate prototypes based on an L_ν-norm of distances, results in a smaller and more efficient system than the classical k-means algorithm. Training with the MCE criterion generates a much smaller system than k-NN classifier without any drop in accuracy.

6. ACKNOWLEDGEMENTS

Mr. Nopsuwanchai is funded for his PhD study by Cambridge Thai Foundation and Cambridge Overseas Trust. The authors would like to thank the members of the Embedded Application Group at the IBM T. J. Watson Center for their help in this research, especially Jane Snowdon for her critical reviewing of the manuscript.

7. REFERENCES

[1] E. McDermott and S. Katagiri, "Prototype-based Minimum Classification Error/Generalized Probabilistic Descent for Various Speech Units," *Computer Speech and Language*, vol. 8, no. 4, pp. 351–368, Oct. 1994.

[2] R. O. Duda and P. E. Hart, *Pattern Classification and Scene Analysis*, Wiley Interscience Publications, 1973.

[3] F. Ricci and P. Avesani, "Data compression and local metrics for nearest neighbor classification," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 21, no. 4, pp. 380–384, 1999.

[4] C. L. Liu and M. Nakagawa, "Evaluation of prototype learning algorithm for nearest neighbor classifier in application to handwritten character recognition," *Pattern Recognition*, vol. 3, no. 34, pp. 601–615, 2001.

[5] S. Katagiri, B.-H. Juang, and C.-H Lee, "Pattern recognition using a family of design algorithms based upon the generalized probabilistic descent method," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2345–2373, 1998.

[6] A. Biem, "Minimum Classification Error Training of Hidden Markov Model for Handwriting Recognition," *Proceedings of ICASSP*, vol. 3, pp. 1529–1532, 2001.

[7] A. Biem and S. Katagiri, "Cepstrum-Based Filter-Bank Design Using Discriminative Feature Extraction Training at Various Levels," in *Proc. IEEE ICASSP*, Apr. 1997, vol. 2, pp. 1503–1506.

[8] S. E. Fahlman, "An Empirical Study of Learning Speed in Back-Propagation Networks," Technical Report CMU-CS-88-162, Carnergie Mellon University, 1988.

[9] E. Ratzlaff, "Methods, report and survey for the comparison of diverse isolated character recognition results on the unipen database," *Proc. of ICDAR*, vol. 1, pp. 629–633, 2003.