Proceedings of the
46th IEEE Conference on Decision and Control
New Orleans, LA, USA, Dec. 12-14, 2007

High Performance Quadratic Classifier and the Application On PenDigits Recognition

ZhengYi John ZHAO
Dept. of Electrical & Computer Engineering,
National University of Singapore, Singapore
zzytgx@yahoo.com.cn

Jie Sun
Business School
National University of Singapore, Singapore
jsun@nus.edu.sg

Shuzhi Sam Ge
Dept. of Electrical & Computer Engineering
National University of Singapore, Singapore
elegesz@nus.edu.sg

***Abstract*— A nonconvex quadratic classifier is proposed for pattern recognition. The classifier is obtained by solving a second-order cone optimization problem on the training data set. Numerical results are presented to compare this classifier with the Gaussian classifier and k-NN classifiers. Regarding to the application of hand written digits recognition, the computational result shows that the proposed quadratic classifier always achieves highest correctness in the testing stage although it takes the longest computational time in the training stage.**

***Index Terms*— second-order cone optimization, pattern recognition, pen-digit recognition, quadratic programming.**

I. Introduction

Pattern recognition is the study of how machines can observe the environment, learn to distinguish patterns of interest from their background, and make sound decision about the categories of the patterns [1]. It has found applications in various engineering and scientific disciplines such as biology, psychology, medicine, marketing, computer vision, artificial intelligence, and remote sensing.

A complete procedure of statistical pattern recognition usually consists of three steps. (1) preprocessing, which includes sampling, removing noise, scaling or normalization and then each sample is represented by a list of data, called as $d-$ dimensional feature vector; (2) training, and (3) classification (testing). This paper will focus on the training and testing. Namely, we assume the sample vectors have already been normalized and their noise level is acceptable.

We will also focus on Supervised Learning, which means that the class of each training sample is known.

Suppose that the total number of the training samples is N and each sample data is a d dimensional vector. Sometime it may happen that $d >> N$, such as in image processing and classification. Then the *peaking phenomenon* (or *curse of dimensionality*) will occur [3]. Jain proposed in [4] that taking ($\frac{N}{d} > 10$) is a good practice to follow in classifier design. However, if the peaking phenomenon does occur, then methods of proper feature extraction or methods of principal component analysis could be applied to reduce the dimension of the data, see for example [5], [6].

Inspired by the Bayesian decision model based on Gaussian distribution assumption, this paper develops a quadratic classifier through a minimization problem with quadratic conic constraints. The problem is formulated to make it a convex problem. By removing the positive semi-definite requirement inherited from the Bayesian model, the proposed quadratic classifier achieved the highest degree of correctness in our numerical experiment, compared with the k-NN classifier and the Gaussian classifier. In the experiment on pen-digits recognition this quadratic classifier could achieve 86% correct rate for over 7400 testing samples, and the ratio of $\frac{N}{d}$ is only 5. The time of computing the quadratic classifier is within 4 seconds in an ordinary Pentium IV computer.

Regarding the application of hand written digits recognition, it is just taken as one of many available test-data to bench-mark our proposed classifier. We donot bother the preprocessing (sampling, noise removing, normalization) at all, while only focus on the training and testing. Thanks to the on-line Machine Learning database (available at *ftp://ftp.ics.uci.edu/pub/machine-learning-databases/pendigits/*), we can start out work directly and necessary tools only include matlab and some open source solvers. It is encouraging to find our proposed classifier can achieve consistantly better correct rate than other well-know classifiers, such as Linear, Gaussian and K-NN [8] [9]. Although our approach takes longest training time, several strategies (parallel computing, Step Up Size) are proposed to speed it up.

Table I gives the notations which will be used in our problem formulation and solution.

II. Formulation of the Quadatratic Classifiers

Assume that the training samples follow the Gaussian distribution, then the mean vector $\mu_k \in \mathbf{R}^d$ and the

TABLE I
NOTATION FOR SAMPLE, FEATURE AND CLASSIFIERS

Symbol	Description
N	Total number of training samples
d	Feature dimension
$\mathbf{s}_i \in \mathbf{R}^d$	$i \in 1, ..., N$, Training data
$\mathbf{x} \in \mathbf{R}^d$	Testing data
K	Total number of classes
c_i	actual class of sample i
ω_k	$k \in 1, 2, ..., K$, Symbols for all classes
$n(\omega_k)$	Priori of class ω_k, it $= P(\omega_k)N = \sum_{c_i=\omega_k} 1$ Total number of samples whose class is ω_k
$L_k(\mathbf{s}) : \mathbf{R}^d \rightarrow \mathbf{R}$	A likelihood function associated with each class $k \in 1, 2, ..., K$
$\mathcal{M}_k \in \mathbf{R}^{d\times d}$	A symmetric matrix in likelihood function $k \in 1, 2, ..., K$, associated with quadratic term
$\mathbf{p}_k \in \mathbf{R}^d$	A vector in likelihood function $L_k(\mathbf{s})$ $k \in 1, 2, ..., K$, associated with linear term
$q_k \in \mathbf{R}$	A constant parameter in likelihood function $L_k(\mathbf{s})$ $k \in 1, 2, ..., K$, associated with constant term

covariance matrix $\Sigma_k \in \mathbf{R}^{d\times d}$ of class k are calculated as in Equation (1) and in Equation (2), respectively. Then the probability density function of the feature vector in class k is as in Equation (3).

$$\mu_k = \frac{\sum_{i:c_i=\omega_k} \mathbf{s}_i}{n(\omega_k)} \tag{1}$$

$$\Sigma_k = \frac{1}{n(\omega_k)-1} \sum_{i:c_i=\omega_k} (\mathbf{s}_i - \mu_k)(\mathbf{s}_i - \mu_k)^T \tag{2}$$

$$p(\mathbf{x}|\omega_k) = \frac{e^{\left(-\frac{1}{2}(\mathbf{x}-\mu_k)^T {\Sigma_k}^{-1}(\mathbf{x}-\mu_k)\right)}}{(2\pi)^{d/2}|\Sigma_k|^{1/2}} \tag{3}$$

According to the Baysian decision theory, the posterior probabilty $P(\omega_k|\mathbf{x})$, is used in the final decision on whether an unknown sample $\mathbf{x}$ belongs to class ω_k or not. This posterior probability is calculated as follows.

$$P(\omega_k|\mathbf{x}) = \frac{p(\mathbf{x}|\omega_k)P(\omega_k)}{p(\mathbf{x})} \tag{4}$$

The classification is given by

$$k^* = \text{argmax}\{P(\omega_k|\mathbf{x}) : k = 1, ..., K\} \tag{5}$$

Three points should be noted.

B1: The prior probability $P(\omega_k)$ is computed by $\frac{n(\omega_k)}{N}$.
B2: $p(\mathbf{x})$ is common in all posterior functions, $\{P(\omega_k|\mathbf{x}) : k = 1, ..., K\}$.
B3: The relative values of the posteriori are more important for decision making, for the final decision prefers the relatively largest one in (5).

Rearranging Equation (4), and define the likihood function for class ω_k, there will be

$$\begin{aligned} L_k(\mathbf{x}) &= P(\omega_k|\mathbf{x})p(\mathbf{x}) = p(\mathbf{x}|\omega_k)P(\omega_k) \\ &= p(\mathbf{x}|\omega_k)\frac{n(\omega_k)}{N} \end{aligned}$$

Then, the likelihood function based on Gaussian distribution assumption is following.

$$L_k^G(\mathbf{x}) = \frac{e^{\left(-\frac{1}{2}(\mathbf{x}-\mu_k)^T {\Sigma_k}^{-1}(\mathbf{x}-\mu_k)\right)}}{(2\pi)^{d/2}|\Sigma_k|^{1/2}} \cdot \frac{n(\omega_k)}{N} \tag{6}$$

Since the log-function is a monotone one, the logarithm of function (6) can be used in (5) for classification, where $\mathbf{x}$ is the input, while $n(\omega_k)$, μ_k and Σ_k are parameters already known through the training mode.

Let

$$T_k = \frac{n(\omega_k)}{N(2\pi)^{d/2}|\Sigma_k|^{1/2}}.$$

Then

$$\begin{aligned} \ln(L_k^G(\mathbf{x})) &= -\frac{1}{2}(\mathbf{x}-\mu_k)^T {\Sigma_k}^{-1}(\mathbf{x}-\mu_k) + \ln(T_k) \\ &= -\frac{1}{2}\mathbf{x}^T\left({\Sigma_k}^{-1}\right)\mathbf{x} + \left(\mu_k^T {\Sigma_k}^{-1}\right)\mathbf{x} \\ &\quad -\frac{1}{2}\mu_k^T\left({\Sigma_k}^{-1}\right)\mu_k + \ln(T_k) \end{aligned}$$

Since function $\ln()$ is monotonic, i.e.

$$\ln(L_{k1}^G(\mathbf{x})) > \ln(L_{k2}^G(\mathbf{x})) \equiv L_{k1}^G(\mathbf{x}) > L_{k2}^G(\mathbf{x})$$

The log-likelihood function $L_k^G(\mathbf{x})$ based on Gaussian distribution assumption, is actually a special quadratic function, where $-\frac{1}{2}\mathbf{x}^T\left({\Sigma_k}^{-1}\right)\mathbf{x}$ is the 2^{nd} order term, $\left(\mu_k^T {\Sigma_k}^{-1}\right)\mathbf{x}$ is the 1^{st} order term and $-\frac{1}{2}\mu_k^T {\Sigma_k}^{-1}\mu_k + \ln(T_k)$ is a constant term. Note that its second order term has a symmetric positive semi-definite part $\mathbf{x}^T\left({\Sigma_k}^{-1}\right)\mathbf{x}$.

We relax the positive semi-definite property, and keep its quadratic form in the proposed quadratic classifier (7), where $\mathcal{M}_k \in \mathbf{R}^{d\times d}$ is symmetric matrix, $\mathbf{p}_k \in \mathbf{R}^d$ is a vector and $q_k \in \mathbf{R}$ is a scalar. And $\{\mathcal{M}_k, \mathbf{p}_k, q_k\}$ is associated with class ω_k.

$$L_k^Q(\mathbf{x}) = \mathbf{x}^T\mathcal{M}_k\mathbf{x} + \mathbf{p}_k^T\mathbf{x} + q_k \tag{7}$$

In the spirit of the Bayes Decision Rules, we hope $L_k^Q(\mathbf{x})$ to have the following four properties:

Q1 : $L_k^Q(\mathbf{x}) \geq 1$, if $\mathbf{x}$ belongs to class k.
Q2 : $L_k^Q(\mathbf{x}) \leq -1$, if $\mathbf{x}$ doesnot belong to class k.
Q3 : $-1 < L_k^Q(\mathbf{x}) < 1$, if not clear whether $\mathbf{x}$ belongs to class k or not.

Q4 : $L_{k1}^Q(\mathbf{x}) > L_{k2}^Q(\mathbf{x})$, if it is more likely that $\mathbf{x}$ belongs to class $k1$ than that $\mathbf{x}$ belongs to class $k2$.

If there exist $\{\mathcal{M}_k, \mathbf{p}_k, q_k\}$, such that [Quad-1] and [Quad-2] are satisfied for all the training samples, it means class ω_k is completely separable by $L_k^Q(\mathbf{x})$ with other classes. If [Quad-1] and [Quad-2] cannot be satisfied for all the training samples, [Quad-4] comes up for a decision in the worst case, and [Quad-3] comes up to avoid error decision.

Aiming at the above four properties a minimization problem with quadratic conic constraints is proposed to find the parameters of $L_k^Q(\mathbf{x})$.

The problem $MinQuad_k$ is associated with $L_k^Q(\mathbf{x})$ for class ω_k. If the system has total of K classes, there will be K parallel problems, $\{MinQuad_k : k = 1, 2, ..., K\}$. In all the K problems, $\mathcal{M}_k, \mathbf{p}_k, q_k$ are decision variables to be solved, e_i, ϵ are slack decision variables, and training sample $\{\mathbf{s}_i : i = 1, ..., N\}$ are parameters, whose classes are known. C is a positive scalar, indicating the weight of the penalties of misclassification.
Equation (9) states that the matrix $\mathcal{M}_k$ is symmetric.
Inequality (10) states that the 1^{st} parameter and 2^{nd} parameter are bounded by a sphere, whose radius, ϵ, should be as small as possible. This constraints is actually trying to achieve numerical stability, as to avoid ill-condition and bad convergence.
Inequality (11) states that if sample $\mathbf{s}_i$ belongs to class ω_k, there should be $L_k^Q(\mathbf{s}_i) \geq 1$, or there will be $C \cdot e_i$ incurred to the total cost, where $e_i = \max\left(0, 1 - L_k^Q(\mathbf{s}_i)\right)$.
Inequality (12) states that if sample $\mathbf{s}_j$ does not belong to class ω_k, there should be $L_k^Q(\mathbf{s}_j) \leq -1$, or there will be $C \cdot e_j$ incurred to the total cost, where $e_j = \max\left(0, 1 + L_k^Q(\mathbf{s}_i)\right)$.
So, inequalities (11) and (12) try to meet the requirements [Quad-1] to [Quad-4]. Inequality (13) states that all the slack error variables are non-negative.

After the training process, K quadratic functions will be available, and the decision rule in recognition process is following in (14), which is similar to the Bayes decision rule (5).

$$k^* = \text{argmax}\{L_k^Q(\mathbf{x}) : k = 1, ..., K\} \tag{14}$$

The problem $MinQuad_k$ is a so-called second-order cone program. It is a special convex optimization problem and is easy to solve. Various scientific and commercial packages are available.

TABLE II
PROBLEM SIZE OF PROBLEM $MinQuad_k$

Total Variables	$\frac{(d^2+3d)}{2} + N + 2$
Total Constraints	$N + d^2 + 1$
Size Quadratic Cone	$\frac{(d+1)J}{2} + d + 1$

TABLE III
COMPUTATIONAL COMPARISION IN TRAING MODE

Classifer	Quadratic	Gaussian	k-NN
Multiply	$N * d^2$	$n(\omega_k)\frac{d^2+d}{2} + d^2 + n(\omega_k)$	0
Addition	0	$(n(\omega_k) - 1)\frac{d^2+d}{2} + n(\omega_k)d$	0
Others	Solver	$d \times d$ Matrix Inversion	0

III. COMPUTATIONAL COMPARISON AMONG QUADRATIC CLASSIFIER, GAUSSIAN AND KNN CLASSIFIERS

Above $MinQuad_k$ is a formulation to solve the quadratic model of $\mathcal{M}_k, \mathbf{p}_k, q_k$ to form $L_k^Q(\mathbf{x})$, and totally K such models to be solved. This is done in the training mode. The same for the Gaussian classifier $L_k^G(\mathbf{x})$, whos parameter is $\{\mu_k, \Sigma_k\}$. For KNN (K Nearest Neighbourhood) classifier [7], however, there is no training mode, but only testing mode, which is done directly from the training samples.
Table (II) shows the problem size of the $MinQuad_k$ corresponding to sample size N and feature dimension d. The variable number grows linearly with N while quadratically with d. When problem size grows large, sparse technique and solvers with interior point methods (i.e. CPLEX, MOSEK, SEDUMI) have to be used.

Table (III) shows the computational comparison of three classifiers during training mode. The multiplication and addition in Quadratic classifier only count those in formulation of $MinQuad_k$ but not consider those in the solver process. Compuation in Gaussian classifier doesnot consider the matrix inversion, which can be done by SVD algorithms because sometimes singularity might happen. Computation in KNN's training mode is zero. So

TABLE IV
COMPUTATIONAL COMPARISION IN TESTING MODE

Classifer	Quadratic	Gaussian	k-NN
Multiply	$\left(d^2 + 2d\right) K$	$\left(d^2 + d + 2\right) K$	Nd
Addition	$\left(\frac{d^2+d}{2} + d + 1\right) K$	$d^2 K$	N(d-1)
others		1 exp	1 sqrt
Sorting	1 over K	1 over K	$\sqrt{N}$ over N then 1 over K

$$MinQuad_k\text{: minimize}_{\mathcal{M}_k,\mathbf{p}_k,q_k,e_i,\epsilon} \quad \epsilon + C\sum_{i=1}^{N} e_i \tag{8}$$

$$\text{subject to} \quad \mathcal{M}_k(m,n) = \mathcal{M}_k(n,m), \forall m < n, \text{and } m,n \in \{1,2,...,d\} \tag{9}$$

$$\epsilon \geq \sqrt{\sum_{1\leq i\leq j\leq d} \mathcal{M}_k(i,j)^2 + \sum_{1\leq i\leq d} \mathbf{p}_k(i)^2} \tag{10}$$

$$\mathbf{s}_i^T\mathcal{M}_k\mathbf{s}_i + \mathbf{p}_k^T\mathbf{s}_i + q_k \geq 1 - e_i, \text{ if } c_i = \omega_k, \forall i \in \{1,2,...,N\} \tag{11}$$

$$\mathbf{s}_j^T\mathcal{M}_k\mathbf{s}_j + \mathbf{p}_k^T\mathbf{s}_j + q_k \leq -1 + e_j, \text{ if } c_j \neq \omega_k, \forall j \in \{1,2,...,N\} \tag{12}$$

$$e_i \geq 0, \forall i \in \{1,2,...,N\} \tag{13}$$

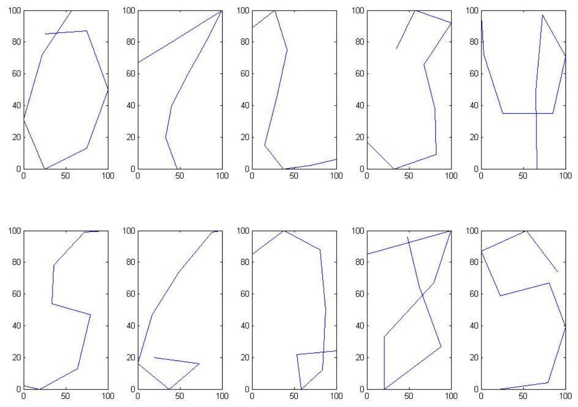
the Quadratic classifier has highest computational cost in training mode. And expriments shows that formulation of $MinQuad_k$ even takes much longer time than solving it.

Table (IV) shows the computational comparison of three classifiers during testing mode. Here the k-NN is the most expensive one and the Quadratic and the Gaussian are much fast.

IV. Application in Pen-digit Recognition and Performance Comparison

The pen-based handwritten digit recognition, generally, has three main steps: preprocessing, training and recognition. The preprocessing is elaborated in [7]. This paper only focus on the training and recognition process and the sample data is from the On-Line Machine Learning database (available at *ftp://ftp.ics.uci.edu/pub/machine-learning-databases/pendigits/*). There are totally 7494 samples and each sample is a 17-dimension array. In the 17-dimension array, the first 16 numbers are actually 8 pairs of (x, y) coordinates of 8 points and followed by the actually written digit as the 17th number. Hence the sequenced data in sample i is $\{x_1^i, y_1^i, x_2^i, y_2^i, ..., x_8^i, y_8^i, c_i\}$. So, $N <= 7494$, and the feature dimension $d = 16$, as we only take the first 16 number as sample feature, i.e. $\mathbf{s}_i = \{x_1^i, y_1^i, x_2^i, y_2^i, ..., x_8^i, y_8^i\}$, whose actual class is c_i. Figure 1 shows some sample graphs in dynamic representation from digit 0 to digit 9, as well as randomly chosen 16 samples of digit 9.

Each digit will be a class, so $K = 10$. For each class $\omega_k \in \{0,1,2,3,4,5,6,7,8,9\}$, there is a symmetric matrix $\mathcal{M}_k \in R^{d\times d}$, a vector $\mathbf{p}_k \in R^d$ and a scalar $q_k \in R$. Feature dimension $d = 16$ will be the size of the symmetric matrix M and of the vector p. The traing mode of Quadratic classifier is simply formulation and solving K problems $\{MinQuad_k : k = 1,2,...,K\}$, each problem has the objective function (8) and constrained by (9), (10), (11), (12), (13).


(a) From 0 to 9

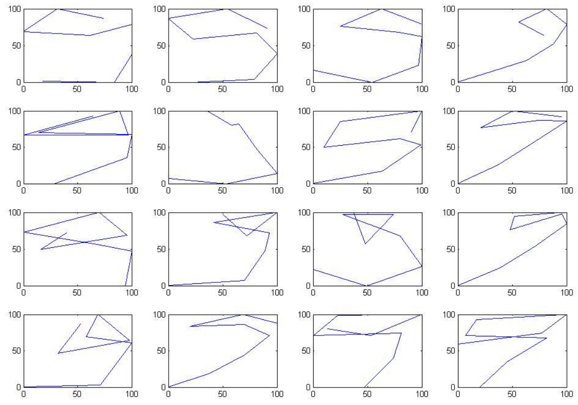
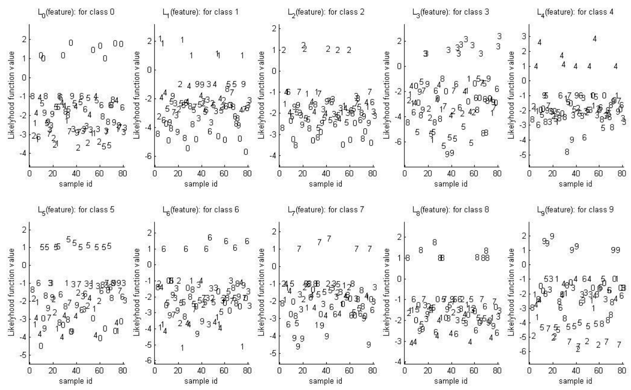
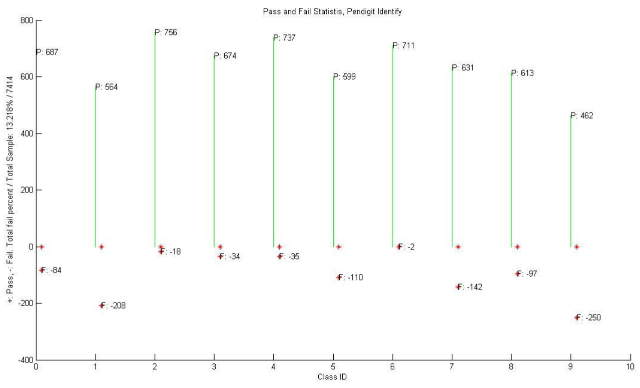

(b) Randomly pick 16 samples of digit 9

Fig. 1. Sample Handwritten Digits

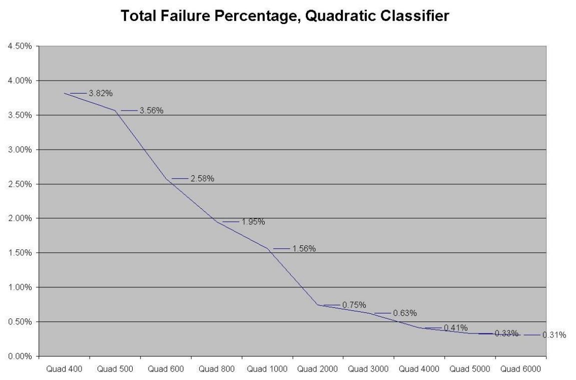


(a) Behavior of Likelihood Function



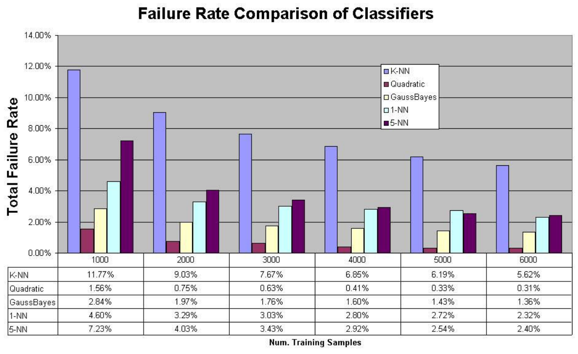
(b) Recognition Result with testing data

Fig. 2. Solution with 80 Training Samples



(a) FailureRate v.s. Num. Training Samples

Fig. 3. Quadratic Classifier Failure Rate



	1000	2000	3000	4000	5000	6000
K-NN	11.77%	9.03%	7.67%	6.85%	6.19%	5.62%
Quadratic	1.56%	0.75%	0.63%	0.41%	0.33%	0.31%
GaussBayes	2.84%	1.97%	1.76%	1.60%	1.43%	1.36%
1-NN	4.60%	3.29%	3.03%	2.80%	2.72%	2.32%
5-NN	7.23%	4.03%	3.43%	2.92%	2.54%	2.40%

(a) FailureRate Comparison

Fig. 4. Comparison among Classifiers

The 1st graph in Fig. 2 shows the behavior of likelihood functions from 0 to 9, which are the result taught by 80 samples. The x-axis is sample id, the text in the figure is the actual class of that sample, and the y-axis is the output value by likelihood function. It can be seen that all the training samples are completely separated by classifiers, such that class ω_k's sample will have the largest value output by likelihood function $L_k^Q()$. This value is nearly 1. While for other samples whose class is not ω_k, the values output by $L_k^Q()$ is usually less than or near -1. This is just as designed to cater for property [Quad-1] to [Quad-4].

After the training process, the remaining data can serve for testing, (14) is used as decision rule. And the result of testing is shown in the 2nd graph of Fig. 2. X-axis is class-id, digits from 0 to 9. Y-axis shows the correct rate and failure rate for each class. Height of the green line above zero is the correctly recognition samples and the star below zero is the failure recognition samples. The total correct rate for all testing samples is greater than 86%. Note that the total number of training samples is 80 out of 7494, and this correct rate is already comparable with the highest correct rate achieveable by linear classifiers.

V. Simulation Result and Comparison

Fig. 3 shows the total failure rate of the quadratic classifier v.s. number of training samples. It is shown that when the number of training sample increased, the total failure rate will decrease and could achieve 0.31% (above 99.6% correct rate) when total number of training sample is 6000.

Fig. 4 compares different classifiers including k-NN, Bayesian Classifier based on Gaussian distribution, Quadratic classifier, and 5-NN, 1-NN. The first KNN, the boundary circle just enclose $\sqrt{N}$ of nearest point, while for 5-NN it enclose 5 nearest point and 1-NN it enclose single nearest point. It is shown that the proposed quadratic classifier always achieves better correct rate than any other classifiers through all the case of training samples. Noting that the experiment is done without considering the writer dependency issue. In [9], when the writer dependency issue is considered, the best misclassified rate could achieve

0.88%. It is our belief that adding consideration of writer dependency to the quadratic classifier will achieve even better result.

VI. Conclusion and Suggestion for Further Research

This paper would rather propose a new perspective to an existing problem, than declare some best result. Although the quadratic classifier achieve the best correct rate, it cost the most time in computation. Experiments shows that the formulation time is much longer while the solution time usually less than 10 seconds even for problem with $N = 6000$. Two things can be done to improve the speed. One is parallel computing, the other improvement is by step-up-size techneque. The parallel computing is illustrated in Fig. 5. Because the problems $\{MinQuad_k : k = 1, 2, ..., K\}$ are independent to each other, the K problems can be solved in parallel at the same time. Because constraints for $\{MinQuad_k : k = 1, 2, ..., K\}$ are almost the same, except a negative sign in (11), (12). A common template could be formulated and passed to all K problems, while each problem can do a slight adjustment so as to save time. By observing constraints (9), (10), (11), (12), (13), it is found that only (11), (12) are dependent on the number of training samples N while other 3 constraints are the same always. The stretagy of step-up-size (SUS) is:

S1: To solve a smaller problem (i.e. $N = 400$ correct rate better than 96% and solution time is around 1 minute) first, and give the decision for all those testing samples which exactly satisfy [Quad-1] and [Quad-2], while donot give decision (classification) for those point satisfying [Quad-3].

S2: Increase the problem size, during the formulation, the matrix element for (9), (10) can be reused from previous smaller size problem.

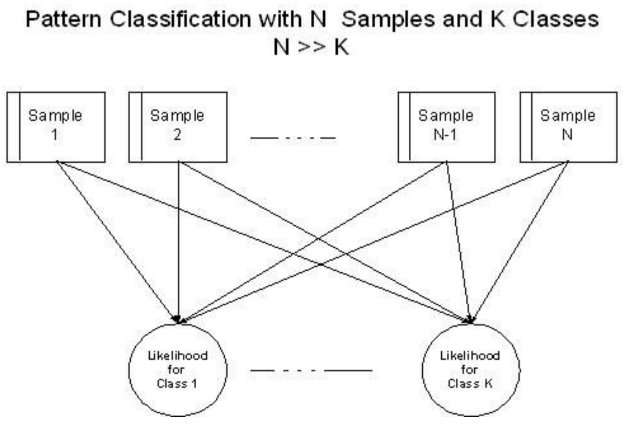


Fig. 5. Parallel Learning Process for N Samples and K Classes, N is much greater than K

References

[1] A. K. Jain, Robert P.W. Duin and J. Mao, "Statistical Pattern Recognition: A Review," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol.22, No.1, pp.4-37, Jan. 2000.

[2] R. O. Duda, Peter E. Hart, D. G. Stork, "Pattern Classification", *Wiley Interscience*, 2nd Edition, 2000.

[3] S. J. Raudys and A. K. Jain, "Small Sample Size Effects in Statistical Pattern Recognition: Recommandations for Practitioners,", *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 13, no. 3, pp.252-264, 1991.

[4] A. K. Jain and B. Chandrasekaran, "Dimensionality and Sample Size Considerations in Pattern Recognition Practice," *Handbook of Statistics*. P.R. Drishnaiah and L.N.Kanal, eds., vol.2, pp.835-855, Amsterdam: North-Holland, 1982.

[5] H. Chernoff, "The Use of Faces to Represent Points in k-Dimensional Space Graphically," J.Am. Statistical Assoc., vol.68, pp.361-368, June 1973.

[6] V. Belhumeur, J. Hespanha, and D. Driegman, "Eigenfaces vs. Fisherfaces: Recognition Using Class Specific Linear Projection," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol.19, no.7, pp. 711-720, Jul 1997.

[7] R. H. Kassel, "A Comparison of approaches to On-line handwritten character recognition," *PhD Thesis, Massachussets Institute of Technology,* 1995.

[8] M. Schenkel, H. Weissman, I. Guyon, C. Nohl, and D. Henderson, "Recognition-based segmentation of on-line hand-printed words," in *Advances in Neural Information Processing Systems(S. J. Hanson, J.D. Cowan, and C.L. Giles. eds.)*, vol. 5, (Denver, Colorado), pp. 723-730, Morgan Kaufman, 1993.

[9] C. C. Tappert, C. Y. Suen, and T. Wakahara, "The state of the art in on-line handwriting recognition," in *IEEE Transaction on Pattern Analysis and Machine Intelligence*, vol. 12, pp. 787-808, Aug. 1990.