

Lab 2 Report

Yuliang Xiao
Student ID: 1929288

Abstract

The aim of our experiment was to evaluate the performance of two statistical pattern recognition methods based on UCI iris dataset using 5-fold cross-validation, which are Modified Quadratic Discriminant Function (MQDF) and Parzen Window. The UCI iris dataset is a well-known dataset that contains measurements of the sepal length, sepal width, petal length, and petal width of three different species of iris flowers. In order to analyse and compare the performances for different statistical algorithms, not only the accuracy be recorded, but also their time complexity, since when their accuracies are approximate, the algorithm with less time complexity is better. The code is made available at <https://github.com/Regen2001/MQDF-IRIS-recognition> for MQDF and <https://github.com/Regen2001/Parzen-window-iris-recognition> for Parzen window.

1 Introduction

Modified Quadratic Discriminant Function (MQDF) and Parzen window method are two commonly used classification algorithms in pattern recognition. MQDF is a linear classifier that assumes a multivariate normal distribution for each class, which works by computing the Mahalanobis distance between the input feature vector and the mean vector of each class. While Parzen window is a non-parametric classifier that estimates the class conditional densities using kernel density estimation, which works by assigning a kernel function to each data point in the training set and using them to estimate the density of the class conditional distribution.

Furthermore, both two methods are based on Bayesian theory, which estimate the maximum likelihood function of each classification based on the available training dataset. And then, the predicted value is inserted into each likelihood function to estimate the probability of belonging to each classification, while the final result is determined based on the classification with the highest probability.

Different from the popular machine learning methods such as multilayer perceptron and convolutional neural network, algorithms use the training set to obtain the weights for each hidden layer. MQDF and Parzen window does not involve a training process, but rather relies solely on the establishment of probability models for the given samples.

For large datasets, as the high computational requirements, deep learning methods are suitable. However, statistical learning can be an efficient and effective choice when dealing with a small dataset. Overall, for data sets with a few samples features, the statistical learning method also shows a high accuracy. And as the small amount of computation required, they can be a better choice to learning a small dataset.

For large datasets, as the high computational requirements, deep learning methods are suitable. However, statistical learning can be an efficient and effective choice when dealing with a small dataset. Overall, for data sets with a small number of samples and a small number of features, the statistical learning method also shows a high accuracy. And as the small amount of computation required, they can be a better choice to learning a small dataset.

2 Methodology

Since to avoid overfitting, cross-validation is applied, which testing model on a separate dataset than the one used for training. One common type of cross-validation is k-fold cross-validation, where the dataset is split into k subsets (or folds) of approximately equal size, while the model is trained and tested k times, with each fold being used as the testing data once. In this experiment, our dataset was split into 5 folds, while each fold includes 120 training sample and 30 test sample.

Furthermore, Parzen Window method and Quadratic Discriminant Function (QDF) are related to Bayesian rule, which is

$$P(\omega_x|x) \propto P(\omega_x)P(x|\omega_x) \quad (1)$$

where $P(\omega_x)$ prior class probability and expressed as

$$P(\omega_x) = \frac{n_k}{\sum_{i=1}^K n_i} \quad (2)$$

where n_k is the number of examples from the class ω_x .

2.1 Parzen Window Method

The basic idea For the Parzen window method is to estimate the probability density function of a random variable by using a kernel function to smooth out the empirical distribution of a set of observations Kwak & Choi (2002). Let x_1, x_2, \dots, x_n be a set of n observations of a random variable x with an unknown probability density function $P(\omega_x|x)$ Parzen (1962). The conditional probability $P(\omega_x|x)$ is calculated as

$$P(\omega_x|x) = \frac{1}{n_k} \sum_{x_i \in \omega_k} \frac{1}{h^d} \phi\left(\frac{x - x_i}{h}\right) \quad (3)$$

where h is the hyperparameter known as the bandwidth for the model, and $\phi(u)$ is the kernel function, written as

$$\phi(u) = \frac{1}{\sqrt{2\pi}} e^{-\frac{u^2}{2}} \quad (4)$$

2.2 Modified Quadratic Discriminant Functions

The Quadratic Discriminant Function (QDF) is a parametric method used for classification and density estimation, which assumes that the data come from a Gaussian distribution and estimates the mean vector and covariance matrix of each class Kimura et al. (1987). Furthermore, the mdoel of QDF is derived from the Bayesian rule, which is expressed as

$$g_0^{(l)}(x) = (x - \mu^{(l)})^T \{\Sigma^{(l)}\} (x - \mu^{(l)}) + \log |\Sigma^{(l)}| - 2 \log P(\omega^{(l)}) \quad (5)$$

for a class $\omega^{(l)}$ where $\mu^{(l)}$ and $\Sigma^{(l)}$ denote the mean vector and the convariance matrix x in the class $\omega^{(l)}$, respectively. As μ_M and Σ_M are the maximum likelihood estimates of the mean and the covariance, respectively, and Σ_M is espressed as

$$\Sigma_M = \sum_{i=1}^n \lambda_i \varphi_i \varphi_i^T \quad (6)$$

The term of $P(\omega^{(l)})$ is omitted from the QDF and it is rewritten as

$$g_0^{(l)}(x) = \sum_{i=1}^n \frac{1}{\lambda_i} \{\varphi_i^T (x - \mu_M)\}^2 + \log \prod_{i=1}^n \lambda_i \quad (7)$$

This orthogonal expansion of QDF plays a very important role in observing the influence of estimation error in QDF covariance matrix and improving it. Since the covariance matrix corresponds to a set of eigenvalues and eigenvectors one by one, the influence of the

estimation errors in the covariance matrix is evaluated by estimating the estimation errors in the eigenvalues and eigenvectors. What is important, the estimation error of non-dominant eigenvectors is much larger than that of dominant eigenvectors. It means the non-dominant eigenvector term in 7 (the term with larger i) is more sensitive to the three estimation errors in the covariance matrix for the term in equation 7, and may seriously reduce the performance of QDF [Wacker & El-Sheikh \(1984\)](#).

In order to increase the performance of the discriminant function, a kind of pseudo-Bayesian estimate of the covariance matrix is applied to instead of Σ_M , which is

$$\Sigma_P = \Sigma_M + h^2 I \quad (8)$$

where h is a appropriate constant. Therefore, there is

$$\varphi_i \Sigma_P \varphi_i^T = \varphi_i \Sigma_M \varphi_i^T + h^2 = \lambda_i + h^2 \quad (9)$$

Thus, one improved discriminant function MQDF 1 is shown as [Kimura et al. \(1987\)](#)

$$g_1^{(l)}(x) = \sum_{i=1}^n \frac{1}{\lambda_i + h^2} \{\varphi_i^T(x - \mu_M)\}^2 + \sum_{i=1}^n \log(\lambda_i + h^2) \quad (10)$$

From equation 10, it can be observed that if h is zero, it is same with QDF, while if h is too large, the term of $(\lambda_i + h^2)$ is approximate to h^2 , which mean eigenvalues λ_i will not effect the accuracy of MQDF1.

Another idea MQDF2 is substituting h^2 for all the eigenvalues λ_i , $i \geq k+1$ of Σ_M in QDF, which is expressed as [Kimura et al. \(1987\)](#)

$$g_2^{(l)}(x) = \sum_{i=1}^k \frac{1}{\lambda_i} \{\varphi_i^T(x - \mu_M)\}^2 + \sum_{i=k+1}^n \frac{1}{h^2} \lambda_i \{\varphi_i^T(x - \mu_M)\}^2 + \log(h^{2(n-k)} \prod_{i=1}^k \lambda_i) \quad (11)$$

Since there is

$$\sum_{i=1}^n \{\varphi_i^T(x - \mu_M)\}^2 = \|x - \mu_M\|^2 \quad (12)$$

The equation of MQDF2 11 is rewritten as

$$g_2^{(l)}(x) = \frac{1}{h^2} \left[\|x - \mu_M\|^2 - \sum_{i=1}^k \left(1 - \frac{h^2}{\lambda_i}\right) \{\varphi_i^T(x - \mu_M)\}^2 \right] + (n-k) \log h^2 + \sum_{i=1}^k \log \lambda_i \quad (13)$$

Though observing equation 13, the required time used to calculate is $\frac{k}{n}$ times for QDF and MQDF1 [Kimura et al. \(1987\)](#). From equation 13, the performance should be evaluated varying both h and k independently. However, to reduce the cost of the experiment, we only varied the informance of the value of k , and the value of h^2 was set as the average λ_{k+1} of λ_{k+1} over all classes, and MQFD2 in experiment is

$$g_2^{(l)}(x) = \frac{1}{\lambda_{k+1}^-} \left[\|x - \mu_M\|^2 - \sum_{i=1}^k \left(1 - \frac{\lambda_{k+1}^-}{\lambda_i}\right) \{\varphi_i^T(x - \mu_M)\}^2 \right] + (n-k) \log \lambda_{k+1}^- + \sum_{i=1}^k \log \lambda_i \quad (14)$$

3 Experiment Result and Analysis

3.1 Parzen Window Method

In the sub-experiment, we tested the performance of this method though changing the value of h from 0 to 10 with step of 0.01. Figure 1 is the screen shot for the running result which is printed into terminal, since the size limitation of window, only five results are shown.

```

code -- zsh -- 100x30
Last login: Sat Apr 22 15:43:33 on ttys001
(base) xiaoy@MacBook-Pro-xiao ~ % cd /Users/xiaoy/Library/CloudStorage/OneDrive-个人/Year_4_semester_2/INT304/assignment_2/code
(base) xiaoy@MacBook-Pro-xiao code % python Parzen_Window_for_lab.py
starting...
/Users/xiaoy/Library/CloudStorage/OneDrive-个人/Year_4_semester_2/INT304/assignment_2/code/Parzen_Window_for_lab.py:190: RuntimeWarning: divide by zero encountered in log10
  opt_size += np.log10(p_xw) # Maximum log-likelihood estimation to find the optimal kernel size
Average accuracy when h = 0.1 is 0.96
Average accuracy when h = 0.3 is 0.96
Average accuracy when h = 0.5 is 0.96
Average accuracy when h = 0.7 is 0.95
Average accuracy when h = 0.9 is 0.94
Running time of Parzen Window with 5-fold cross validation for each h: 0.706
Optimal Kernel Size: 0.9
The max average accuracy when h = 0.9 is 0.94
(base) xiaoy@MacBook-Pro-xiao code %

```

Figure 1: The running result which is printed into the terminal.

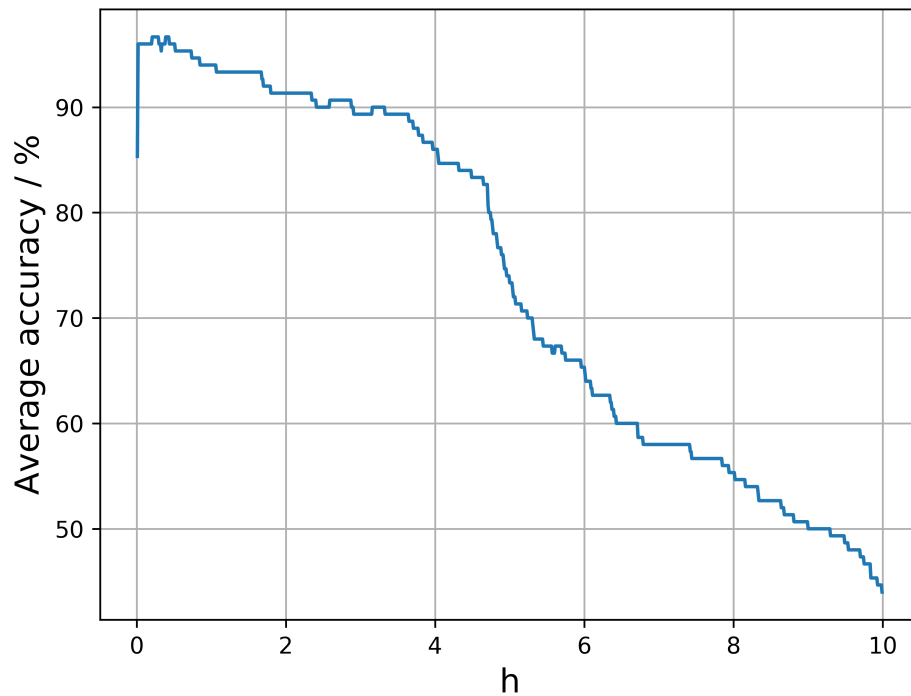


Figure 2: The accuracy of Parzen window method while the value of h is changing.

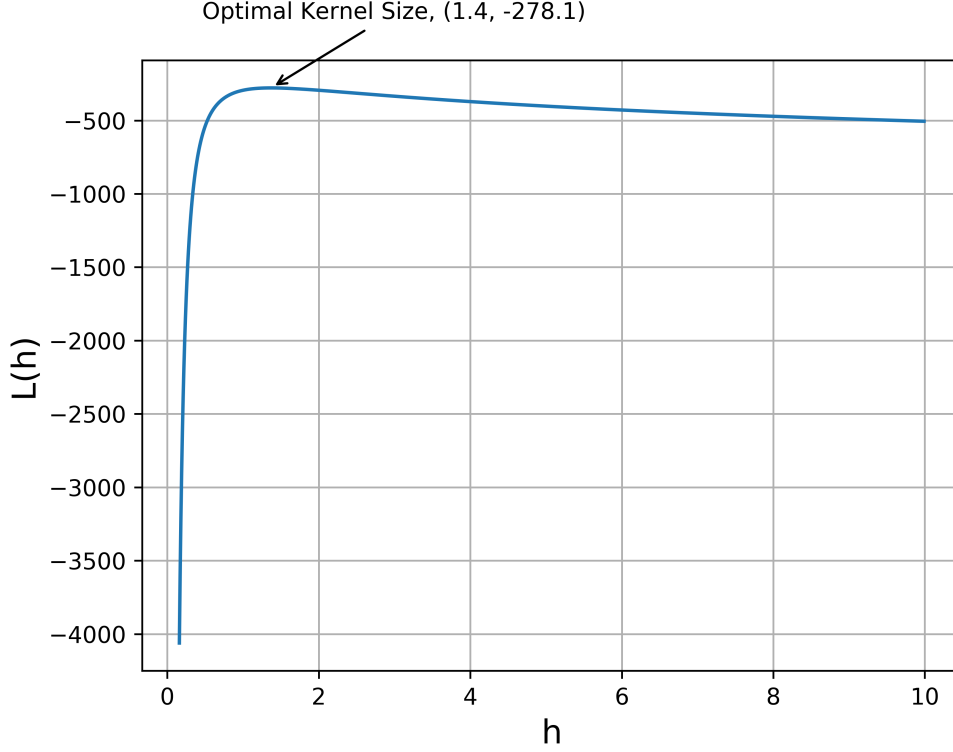


Figure 3: The log-likelihood value while the value of h is changing.

Then, we recorded the accuracy and log-likelihood value for each value of h , while Figure 2 is the accuracy and Figure 3 is the value of log-likelihood. In the experiment, the maximum accuracy was present to $h = 0.21$, which is 0.967. However, the maximum log-likelihood value was -278.1 , appeared at $h = 1.4$.

In most time, when we set $h = \frac{1}{\ln n}$, the performance of Parzen window is best, where n is the sample size of a particular dataset Meilhac & Natar (1999). Since there are 150 samples, the theoretical value of h is 0.2, which is same with our experimental result.

Furthermore, the log-likelihood value is a measure of how well a statistical model fits a set of observations or data points. A higher log-likelihood value indicates a better fit, while a lower value indicates a poor fit. Therefore, from Figure 3, we consider that the best model was appeared when $h = 1.4$ and if the value of h increasing, the performance will not change. However, from Figure 2, as the increasing of the value of h , the accuracy was decrease dramatically, which is a different result. Thus, we consider our model was overfitting if the value of h is too large.

3.2 Modified Quadratic Discriminant Functions

In MQDF1, we tested the performance of this method though changing the value of h from 0 to 10 with step of 0.001. Figure 4 is the screen shot for the running result which is printed into terminal, since the size limitation of window, only ten results are shown.

Then, we recorded the accuracy for MQDF1 when the value of h is changing, which is shown in Figure 5.

When $h = 0$, MQDF1 is same with QDF, which means the accuracy for QDF is 0.98. What it more, before the value of h is more than 0.185, the accuracy for MQDF1 all are 0.98. Then, as the increasment of h , the accuracy was decreasing. If $h > 1.059$, the accuracy

```

code -- -zsh -- 100x30
Last login: Sun Apr 23 16:34:55 on ttys001
[(base) xiaoy@MacBook-Pro-xiao ~ % cd /Users/xiaoy/Library/CloudStorage/OneDrive-个人/Year_4_semester
_2/INT304/assignment_2/code
[(base) xiaoy@MacBook-Pro-xiao code % python MQDF_for_lab.py
starting...
loaded training set
Averay accuracy of 5-fold cross-validation when h = 0.00 is 98.00, running time is 0.072
Averay accuracy of 5-fold cross-validation when h = 0.10 is 98.00, running time is 0.060
Averay accuracy of 5-fold cross-validation when h = 0.20 is 97.33, running time is 0.060
Averay accuracy of 5-fold cross-validation when h = 0.30 is 96.67, running time is 0.058
Averay accuracy of 5-fold cross-validation when h = 0.40 is 96.00, running time is 0.059
Averay accuracy of 5-fold cross-validation when h = 0.50 is 95.33, running time is 0.058
Averay accuracy of 5-fold cross-validation when h = 0.60 is 94.67, running time is 0.059
Averay accuracy of 5-fold cross-validation when h = 0.70 is 93.33, running time is 0.058
Averay accuracy of 5-fold cross-validation when h = 0.80 is 93.33, running time is 0.061
Averay accuracy of 5-fold cross-validation when h = 0.90 is 92.67, running time is 0.059
Running time of MQDF1 with 5-fold cross validation for each h: 0.060
(base) xiaoy@MacBook-Pro-xiao code %

```

Figure 4: The running result which is printed into the terminal for MQDF1

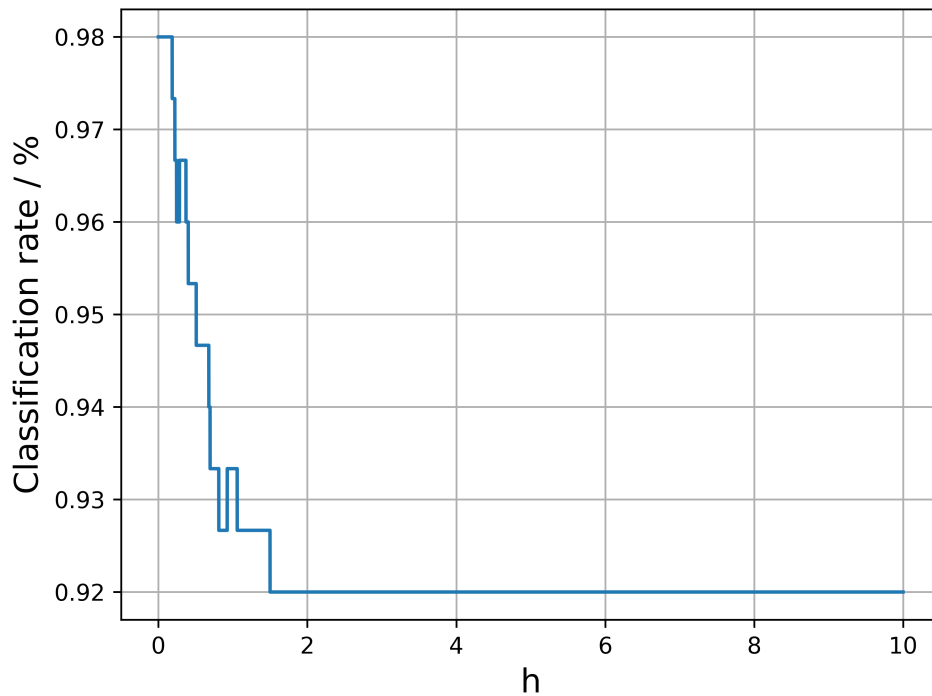


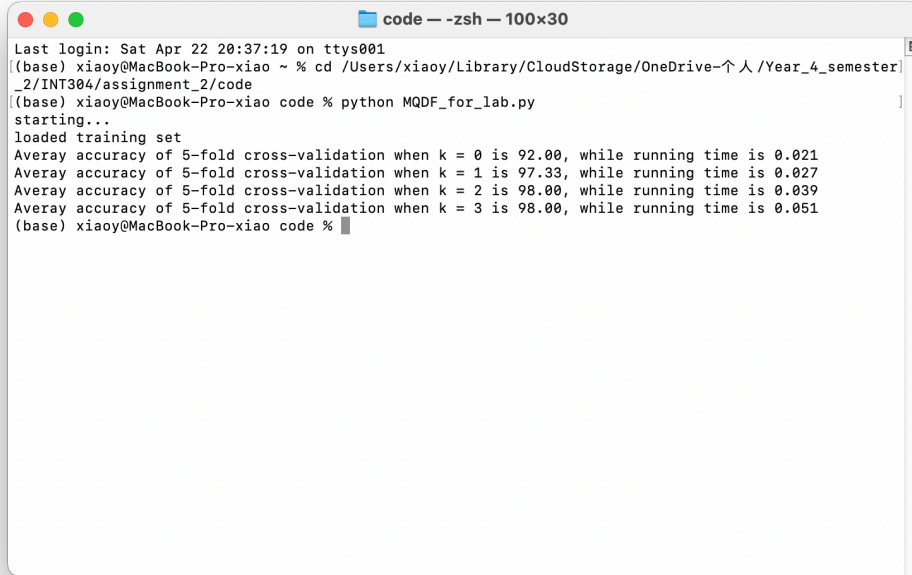
Figure 5: The accuracy for MQDF 1 while the value of h is changing.

was maintained at 0.927. This result is same with what we talked about in above, if h is too large, the term of $(\lambda_i + h^2)$ is approximate to h^2 , which means h will be the only influence factor in MQDF1. Furthermore, for the convenience of comparison with MQDF2, we recorded the result when $h^2 = \bar{\lambda}_i = 0.22$. At that time, the accuracy is 0.927, which is recorded in Table 1.

However, different with Parzen window method, in MQDF1, as the increasing of h , the accuracy will not continue to decline, but stay at a constant, while the performance of accuracy is much better than Parzen Window Method. Since MQDF1 can be considered as a function derived by substituting Σ_M in QDF by $\Sigma_{P'}$, which is the covariance matrix of the density $p_m(x)$ estimated by the Parzen window method though applying circular normal density as the kernel, and it is shown as

$$\Sigma_{P'} = \int (x - \mu_M)(x - \mu_M)^T p_m(x) dx \quad (15)$$

In MQDF2, we set $h^2 = \bar{\lambda}_i$, so in this sub-experiment, only the value of k is changed. Figure 4 is the screen shot for the running result which is printed into terminal.



```

Last login: Sat Apr 22 20:37:19 on ttys001
(base) xiaoy@MacBook-Pro-xiao ~ % cd /Users/xiaoy/Library/CloudStorage/OneDrive-个人/Year_4_semester/2/INT304/assignment_2/code
(base) xiaoy@MacBook-Pro-xiao code % python MQDF_for_lab.py
starting...
loaded training set
Averay accuracy of 5-fold cross-validation when k = 0 is 92.00, while running time is 0.021
Averay accuracy of 5-fold cross-validation when k = 1 is 97.33, while running time is 0.027
Averay accuracy of 5-fold cross-validation when k = 2 is 98.00, while running time is 0.039
Averay accuracy of 5-fold cross-validation when k = 3 is 98.00, while running time is 0.051
(base) xiaoy@MacBook-Pro-xiao code %

```

Figure 6: The running result which is printed into the terminal for MQDF2

For MQDF2, the accuracy will decrease when the value of k decrease, while its time complexity will decrease at the same time. After calculating the relationship between MQDF1 and MQDF2, we found the required time for MQDF2 approach $\frac{k}{n}$ times of MQDF1, while MQDF2 has a smaller time complexity.

For both MQDF1 and MQDF2, when $h^2 = \bar{\lambda}_i$, MQDF2 has a better performance as when k more than 1, the accuracy is more than 0.97. If $k \geq 3$, MQDF2 has the same accuracy with MQDF1. From this experiment, we cannot test the influence for the number of feature chosen since only four features are included in UCI iris dataset.

4 Conclusion

For Parzen Window method, when the value of h is small, the Parzen window is narrow, and the classifier has a high bias. This can lead to underfitting, where the classifier is too

Table 1: The accuracy and running time for MQDF1 when $h = 0.18$, $h^2 = \bar{\lambda}_i$ and MQDF2.

Model	MQDF1, $h = 0.18$	MQDF1, $h^2 = \lambda_i$	MQDF2			
The Value of K	None	None	0	1	2	3
Accuracy	0.980	0.927	0.920	0.973	0.980	0.980
Running time	0.066	0.066	0.016	0.026	0.038	0.048

simplistic and fails to capture the complexity of the data distribution. On the other hand, when the value of h is large, the Parzen window becomes too wide, and the classifier has a high variance. This can lead to overfitting, where the classifier is too complex and fits the noise in the data instead of the underlying pattern.

MQDF1 and MQDF2 are two variations of the MQDF method that were proposed to address some limitations of the original QDF method. QDF assumes that the covariance matrices of all classes are identical and estimates a common covariance matrix from the training data. However, in practice, the covariance matrices may vary across classes, and estimating a common covariance matrix may not be optimal.

MQDF1 assumes that the covariance matrices of all classes are diagonal, meaning that they only have non-zero entries on the main diagonal. This greatly reduces the number of parameters to estimate and makes the method more robust to small training datasets. The diagonal assumption also simplifies the computation of the likelihood function and makes it computationally efficient. However, the diagonal assumption may not hold in practice, especially for datasets with high correlations between the features. In such cases, the model may not be able to capture the full complexity of the data and may lead to reduced recognition performance.

MQDF2, on the other hand, allows each class to have a separate full covariance matrix. This provides more flexibility in modeling the data and allows the method to capture the correlations between the features more accurately. The full covariance matrix also allows the model to better capture the shape of the data distribution, especially for datasets with high intra-class variations. However, the increased number of parameters may lead to overfitting and reduced robustness to small training datasets.

References

- Fumitaka Kimura, Kenji Takashina, Shinji Tsuruoka, and Yasuji Miyake. Modified quadratic discriminant functions and the application to chinese character recognition. *IEEE transactions on pattern analysis and machine intelligence*, (1):149–153, 1987.
- Nojun Kwak and Chong-Ho Choi. Input feature selection by mutual information based on parzen window. *IEEE transactions on pattern analysis and machine intelligence*, 24(12):1667–1671, 2002.
- Christophe Meilhac and Chahab Nastar. Relevance feedback and category search in image databases. In *proceedings IEEE International Conference on Multimedia Computing and Systems*, volume 1, pp. 512–517. IEEE, 1999.
- Emanuel Parzen. On estimation of a probability density function and mode. *The annals of mathematical statistics*, 33(3):1065–1076, 1962.
- A Garry Wacker and Talaat S El-Sheikh. Average classification accuracy over collections of gaussian problems—common covariance matrix case. *Pattern recognition*, 17(2):259–273, 1984.