IN4085
Pattern Recognition
Final Assignment

Contents

1 Introduction

In the final assignment, we are required to implement a pattern recognition system which are able to recognize handwritten digits under two different scenarios. The first one requires us to train a classifier whose overall error rate is lower than 5% by using a training data set of 200 to 1000 digits per class.And the second one requires us to use a training set with only 10 digits per class for training to achieve the error rate of less than 25% .

The training set we used is a subset of NIST(National Institute of Standards and Technology) digits dataset, and the software library is PRtools(version 5.3) which provides most popular classifiers' functions for Matlab.

Because the underlying algorithms have been implemented in PRtools so we can train the classifiers by simply calling the provided functions, the main work of this assignment is finding parameters for certain classifiers, comparing the performance of different classifiers under different data representation(pixel, feature and dissimilarity) and using necessary optimizing approaches to improve those basic classifiers so that they can some of them can satisfy the minimum requirements for error rate.

The report will give a introduction to the design flow about how to train and select suitable classifiers for the two different scenarios in chapter 2.Then, the candidate basic classifiers will be introduced in chapter 4, and the comparison of these classifiers under different data representation will also be illustrated in this chapter.In chapter 5, some approaches for improving basic classifiers selected from last chapter will be discussed including boosting, combining , feature reduction and feature selection. By applying these methods, we could improve the basic classifier and get better performance.Then followed by chapter 6 where we will do the benchmark for the final classifiers for the two scenarios using the provided function $nist\_eval$.

The last two chapters are for the additional points.Chapter 7 will show how we use the classifiers we got from previous chapter to classify our own handwritten digits.In the last chapter 8, we will give the recommendations for our clients about the possible steps to improve the performance of overall system they are constructing.

2 System Design

We will construct our system for both scenarios following the steps below.

- Data processing

In this step, the NIST subset will be loaded to the program and training datasets will be created with different data representations by pixel, feature or dissimilarity.Some image processing functions will be applied to the original data like resizing the images to 16 x 16.

- Testing and selecting basic classifiers

when training data is ready, we will then start to test the the candidate classifiers mentioned before to select those with acceptable performance for further improving. However in scenario 1, the dataset is relatively large which means that training advanced classifier like Support Vector Machine and Neural Network is time-consuming in our laptop. So we decided to exclude the classifiers whose training time is more than one hour. This decision is reasonable because we cannot avoid making a trade-off between performance and resource in a real project.

- Improving the selected classifiers

After selecting those best classifiers under different data representations, we have to apply some method for getting better performance.We chose several common and popular optimization methods including feature selection, feature reduction, combining and boosting.However, those methods are not able to guarantee better performance in any cases, so we still need to do some necessary analysis and experiments.

- Benchmark

When we determine our final classifiers through above steps, we can do benchmark by using *nist_eval* to test the classifiers.This function loads NIST digits which are not subset of training set and then tests the error rate of the classifier. This result is more convincing because the test set and training set are irrelevant. If the minimal requirements are satisfied, we can use them to do live test, otherwise we have to rethink our previous steps.

- Live Test

In this section, we use our own handwritten digits, ten per class, as the test dataset to test the performance of our final classifiers for both scenarios. Because the input images of our digit are not standard, we have to convert them to the data form which can be recognized by our classifier.

3 Data Processing

Before optimization or improvement phase, we process data roughly without considerations for feature extraction and reduction.The methods for three data representations are shown below.

- Pixel

As we simply plot the raw data, we can see that empty borders exist among some of the data.

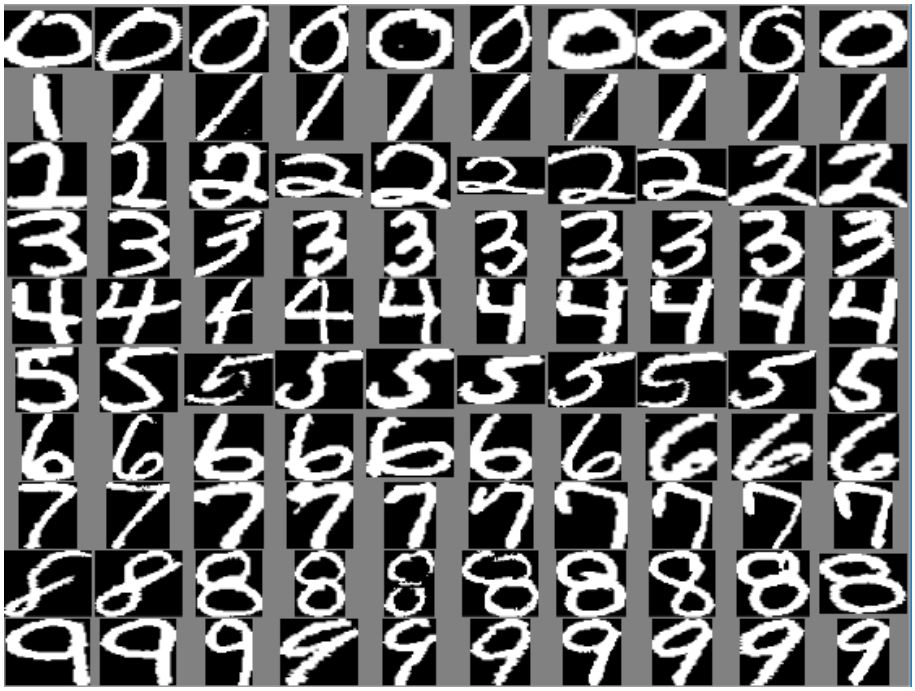

Figure 1: raw data set

In order to make all the data in a consistent scale, we first box the image and then we resize them into 16*16. In this way, we obtain the pixel representation of the dataset.

Figure 2: pixel representation

- Feature

Feature is an important conception in pattern recognition, because we always hope to find the best mapping from sample to feature. As for this assignment, with the help of the *PRtools*, we can easily get the features of image through the function *im_features()*. In the initial step, we just get all the features of the raw data. And we will try to do the feature selection or feature extraction after testing. An image has 24 features, therefore the feature representation of a single sample is a 24 dimension vector.

- Dissimilarity

Dissimilarity is another way to represent the data which focus on the difference between two samples. It is a mapping from high dimension to a real number. In previous step, we have obtained the pixel representation dataset in size of 16*16. Here we choose the Euclidean distance as the dissimilarity type to obtain dissimilarity representation. Here is the dissimilarity matrix of our test set in the following test, we put it here as an example to show our dissimilarity representation.

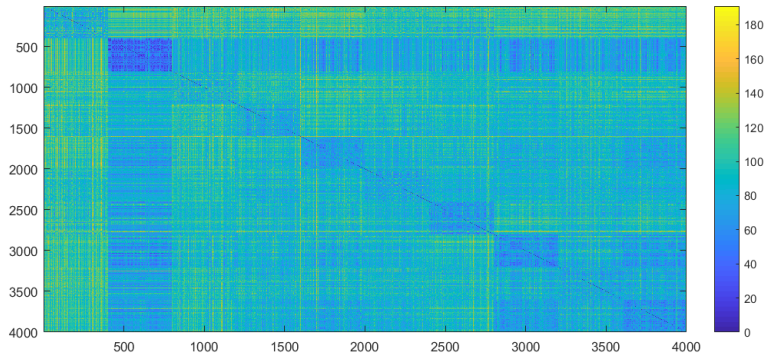

Figure 3: Dissimilarity representation

4 Selecting Basic Classifiers

4.1 Introduction to the candidate classifiers

The classifiers we used for this assignment are what we have learned during the lectures.We also have the experience of using them after completing weekly lab assignments.

- Nearest Mean Classifier (nmc)

Nearest Mean classifier is a simple linear classifier which classifies each object based on the euclidean distance between the input object and mean for each class.

- Linear Bayes Normal Classifier (ldc)

Linear Bayes Normal Classifier or Linear Discriminant Classifier assumes same covariance matrix for all the classes.It is scale insensitive and a simple classifier, but it needs a large amount of data in order to estimate the unknown parameters.

- Quadratic Bayes Normal Classifier (qdc)

Quadratic Bayes Normal Classifier or Quadratic Discriminant Classifier assumes different means and covariance matrix for each of the classes. Like LDC mentioned above, this classifier also needs a lot of training data because it has many unknown parameters which should be estimated.

- Fisher's Least Square Linear Discriminant (fisherc)

Fisher is a linear classifier, which aims to minimize least square error. It is simple linear classifier which works well with small datasets and it is scale insensitive. Fisher classifier is similar to LDC, although their underlying principles are different, their performance are almost similar.

- K-Nearest Neighbor Classifier (knnc)

KNN is a kind of nonparametric estimation method. The advantages of this kind of classifier is obvious. It is simple and flexible, and often is able to achieve relatively good performance, especially when the training set is large compared with the feature dimensionality.There are also some notable disadvantages.First it requires to store the complete training set and the classifying is consuming because it should compare the input objective to training set.Also, we have to optimize the value for k

- Parzen classifier (parzenc)

The underlying concept of parzen classifier is similar to KNN. The former has fixed volume, which is also called as parzen window, and the later has fixed number of training samples to be found. Therefor, the advantages and the disadvantages of parzen classifier are same to KNN, we also have to optimize the value of h.

- Automatic neural network classifier with one hidden layer (neurc)

This kind of neural network is a plug-in function of PRTOOLS, so we are not able to change the parameters but only providing the training dataset.The underlying program will choose most 'suitable' parameters such as the number of layers.Typically, Neural network performs well when the training data is large, but its training period is relatively long.

- Support Vector Machine Classifier (svc)

SVM is one of the most classifiers in machine learning and pattern recognition,the original version is a linear classifier. By means of kernel method, we can use svm to classify dataset which are not linearly separable. SVM is suitable for dataset with large dimensionality and small dataset.And it is time consuming when the dataset is large.

4.2 scenario 1

As for scenario 1, we choose 400 digits per class(4000 digits in total) to build up our training set. And we choose 300 other different digits per class(3000 digits in total) as the test set.Then we test the candidate classifiers under pixel,feature and dissimilarity representation according to the preprocessing step mentioned in chapter 3. We choose cross-validation to test the performance of the pixel representation and use true error of the test set to judge the performance of feature and dissimilarity representation. Firstly we need to find the optimal h value of parzen classifier.

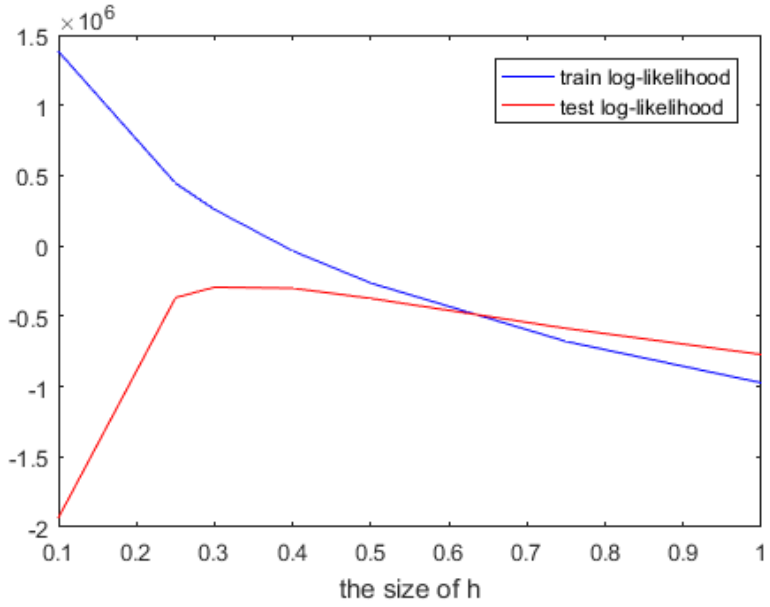

Figure 4: Parzen density estimator

From 5 we can obtain the optimal value of parameter h in this case is 0.3. In the following test step, we will use 0.3 to test further.

We test them for 10 times and record the average error rate in table 1.

Table 1: Average classification error rate

Classifier	Pixel	Feature	Dissimilarity
fisherc	0.1591	0.1615	0.4303
knn1	0.0792	0.3865	0.0150
knn3	0.0970	0.4640	0.0816
ldc	0.1339	0.1443	0.3560
loglc	0.2201	0.1298	0.4313
neurc	0.5860	0.1415	0.8847
nmc	0.6411	0.7099	0.4079
qdc	0.2513	0.2062	0.9000
parzenc	0.0733	0.3923	0.9000

Where **red** means that these classifiers perform good among the candidates, and we can use them for further analysis and improvement. We can see the the error rates in pixel and dissimilarity representations are relatively lower, therefore we would like to test further into these two representation. This is interesting because the performance among different representations are quite different.For example, let us focus on ldc which has the best performance in the feature representation but performs badly in pixel and dissimilarity representation. And 1-nn has great performance in pixel and dissimilarity representation but has a high error rate when classifying feature representation test set. For this scenario,we can say the pixel representation has the best performance among candidate classifiers, more than half of them return a low error rate. And there are 2 classifiers that perform well in dissimilarity representation.

As we learn from the table above, knn classifier is the only classifier that has a good performance.Here we would find out the best value of k through testing. We test different k from 1 to 8 and show the relationship between test error and k.

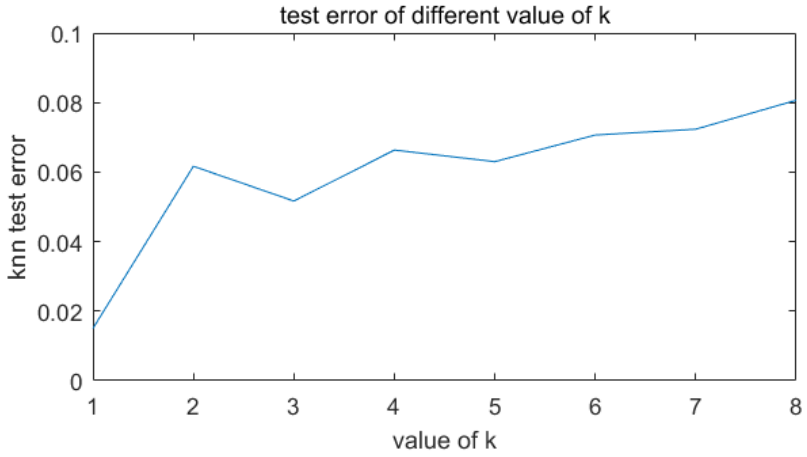

Figure 5: test error of different value of k

Therefore we can obtain that 1-nn classifier has the best performance among all the candidate classifier.

In the next section, we will focus on how to improve the performance of pixel representation because with highest probability we can get a low error rate in this representation.

4.3 scenario 2

In this section, we test the candidate classifiers under three different data representations and list their performance.Unlike the testing method of scenario 1, we do not use cross-validation to test the performance in this scenario, because the number of training set is really small, only ten digits per class and the cross-validation can not get reliable result with such small dataset. In this case, we just randomly select 10 from 1000 as our training set and the remaining 990 digits per class as our test set.Clearly, in this scenario, the number of training samples(10 per class) relatively small compared with the feature dimensionality, so we exclude Parzenc for the reason that it usually performs bad in this case and it is similar to knn. This decision helps reduce our unnecessary workload for finding the optimized h for this classifier.

Table 2: Average classification error rate

Classifier	Pixel	Feature	Dissimilarity
fisherc	0.4229	0.2868	0.4303
knn1	0.3140	0.6768	0.3135
knn2	0.3929	0.6914	0.3631
knn3	0.3754	0.6983	0.3529
ldc	0.9000	0.2464	0.3560
loglc	0.4142	0.3958	0.4313
neurc	0.7435	0.6160	0.6907
nmc	0.2834	0.7099	0.4079
qdc	0.9000	0.5773	0.9000
svm_e	0.2795	0.8744	0.8947
svm_h	0.3619	0.3507	0.3241
svm_s	0.5911	0.8272	0.7344

Where **red** means that these classifiers perform good among the candidates, and we can use them for further analysis and improvement.It is interesting to note that, classifier's performance really depends on the data representation.For instance, ldc achieves best performance(0.2464) when using feature as data representation which it even does not work(0.9) under Pixel. The reason why there are classifiers whose error rate is 0.9 is that the parameters can not be estimated with such small training set compared with the corresponding feature dimensionality.
Obviously, ldc has already satisfied the performance requirement under feature as the data representation, even without any further improvement.In fact, by using the provided benchmark function *nist_eval*, we may get different result because it only provides 100 digits per class as test set which is much smaller than the set we use here.In the next chapter 5, we will use some improvement methods like combining or feature election/extraction. Please note that the improvement methods cannot guarantee a better performance.

5 Improvement

In this section, according to the result in chapter 4, we hope to improve our performance by further testing and evaluation.

5.1 Scenario 1

5.1.1 Pixel

We would like to use combined classifier to test the performance. 1-nn and 3-nn are in the same type therefore here we would choose 1-nn, the better one to test. We would combine the best two classifiers(1-nn and parzenc(0.3)) and test them in different rules.

Table 3: Average classification error rate for combined classifier

combining strategy	minimum	maximum	product	mean	vote
error rate	0.0734	0.0693	0.0761	0.0730	0.0711

Compared with the uncombined situation, we can see the result of combined classifier is slightly better. Next we would try to improve the single classifier. we would draw the learning curve first to see which one performs best.

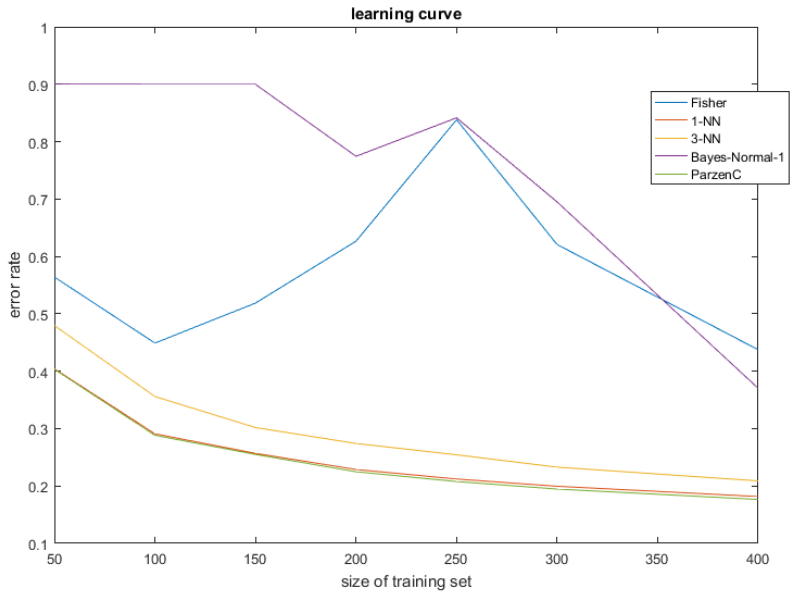

Figure 6: learning curve of classfiers

From the learning curve we can learn that the parzen classifier has the best performance. Next we would like to use PCA to see if we can obtain a better result. In this case, we train the classifier with our training set and use the test set mentioned in section 4.2 to compute the test error as the criteria.

Table 4: PCA fraction value

Pca fraction value	0.9	0.85	0.75	0.6	0.5
error rate	0.031	0.022	0.024	0.026	0.029

From the table we can see the that when the PCA fraction value is 0.85, the classifier performs best among all the values. Therefore, we would choose parzen classifier with Pca fraction value 0.85.

5.2 Scenario 2

As shown in table 2, nmc and svm with exponential kernel perform best under Pixel representation. Fisherc and ldc get the top performance under feature representation. All classifiers perform bad under dissimilarity representation.We consider different representations respectively.

5.2.1 Pixel

For Pixel representation, it is reasonable to use combing to get better performance, because these two classifiers nmc and svm are good but different.So combining these two classifiers is more likely to get a new classifier with performance better than these two. By using similar test method, we

Table 5: Average classification error rate for combined classifier

combining strategy	minimum	maximum	product	mean	vote
error rate	0.2793	0.2584	0.2833	0.2586	0.2818

trained two classifier with different training sets to get more generalized combined classifier.From above table we can see that maximum combining classifier outperform other strategies as well as two original classifiers. Please note that the error rates are not the final result from standard benchmark function *nist_eval*, they are just reference and can only be used to study the relative performance.

5.2.2 Feature

Unlike previous situation, fisherc and ldc are mathematically equivalent, so we are less likely to get better classifier by simply combining them.By contrast, we are going to try feature selection and extraction to see if we can get better performance.Because ldc shows better performance, so we choose ldc for further analysis. First we plot the feature curve by using 'NN' as selection strategy.

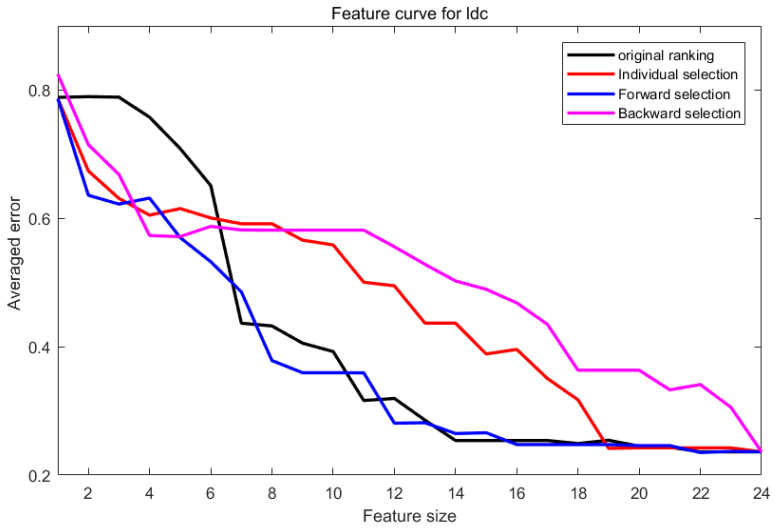

Figure 7: feature curve of ldc

From this plot we can see that when using forward selection, we can get a good error rate at feature size of 18, although the error rate still keep decreasing slightly after this size.Then we employ this method to do feature selection and train ldc by new dataset. Unfortunately, we get higher error rate (0.2654), this means that we cannot get better performance by feature selection in this case.

6 Benchmark

As discussed in chapter 4 and 5, the final classifier of scenario 1 is parzen classifier with pca fraction value 0.85 and scenario 2 is a combing classifier mentioned in 5.2.1. Now we are going to use provided function *nist_eval* to simulate the clients and get the authoritied error rates.The test set for both scenario 1 and 2 is 100 digits per class(maximum setting).

Table 6: benchmark for final classifier

	scenario 1	**scenario 2**
error rate	0.048	0.1911

7 Live test

In this section, we will test our two 'optimal' classifiers by our own handwritten digits.

7.1 data preprocessing

The data preprocessing contains the following steps.

- Read the PNG files with function *imread* provided in matlab.
- Change the image into gray and invert it into black(background) and white(digit) format.
- Convert the image datafile to data object.
- convert the obtained data object to prdataset format.

7.2 Digital image

Instead of writing on a sheet of paper and scan it into computer, we write directly on computer.Because compared with scanned images, this kind of image has less noise. Actually, this decision is reasonable, because touch screen becomes more and more popular and has replaced real paper in many application areas.Figure 8 is the an original image, and after processing procedure, it becomes figure 9.

Figure 8: original image

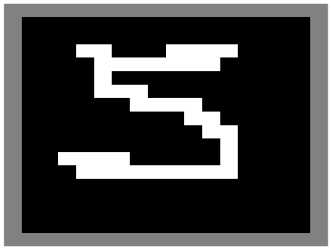

Figure 9: image after processing

7.3 Result

Using the trained classifiers to classify our samples, we get the error rates in table 7

Table 7: Live test result

	scenario 1	**scenario 2**
error rate	0.11	0.31

7.4 Discussion

From table 7 we can see that there is a gap between the performance we get here and the result of benchmark in chapter 6.There are some possible reasons.First, the test set of live test is relatively some, only 10 digits per class, so the error rates are not very general. Second, we can see that the images here and NIST images are not very similar, for example, the size of digits. Also, the data processing may also cause some difference.Overall, the performance of live test is still acceptable.

8 Recommendations

In this section, we give some recommendations for our client about this overall system.

- More training data will help, If you(client) want to get a classifier which is even more reliable than humans' eyes, you should choose a more complex classifier model such as neural network with lots of layers.In this case, you have to 'fed' this kind of classifier with extremely large dataset so that it can achieve its optimal performance.

- Rejection should be taken into consideration if the cost of misclassification is really high.The the threshold of rejection should be optimized. High threshold may bring higher risk because the classifier may not reject even it should reject, while low threshold may have negative effect on the performance.

- If you are planning to construct your system for a bank, you should consider using redundant classifiers and design a vote rule for them. This design does increase the cost, but it is more reliable.

- Sometimes, time is an important criteria. If you are planning to build a real-time system, some classifiers may not be able to finish classification in limiting time.For example, knn and parzenc, they are 'lazy' classifiers, which means that there is no explicit training phase.It consumes a lot of time to calculate the distance between sample and training set,and search the nearest neighbors. Also, this kind of classifier has to store the complete training set.

- Do not try to find a 'perfect' classifier. Instead, you should focus on your assignment, and try to find a suitable classifier for this certain application. There is no classifier which performs good in any cases, but there exists popular classifiers like svm. You should consider those most popular classifiers first because there are more available materials related to them and you save a lot of time if you know how to use them.