Classification Project

Recognize mines and rocks using sonar

1. Introduction

In this project, the task is to train a classifier to discriminate a metal cylinder a roughly cylindrical rock, based on the sonar echo from these objects. During measurement, both cylinders and stones were lying on a sand surface, and the sonar chirp projected at them from different angles (aspect-angles) produced the variation in the data. The data was filtered and a spectral envelope of 60 samples was extracted.

There are 208 samples in total, the training set contains 104 of these and the test set contains the other 104 samples. The samples are sorted in increasing order of aspect-angle. The target is to use the training set of 104 samples to build a model and use the model to classify the test set into two classes, which is whether it is mine or rock. In this project, we use two methods to pretreat the data, which are the downsampling (sum the power in six bands) and Principle Components Analysis. We use three kinds of linear models, which are linear Gaussian classifier, logistic regression classifier and simple perceptron. We use two kinds of nonlinear models which are KNN classifier and Multi-layer Perceptron.

2. Methodology

2.1 Data pretreatment

2.1.1 PCA

Principal component analysis (PCA) is a mathematical procedure that uses an orthogonal transformation to convert a set of observations of possibly correlated variables into a set of values of linearly uncorrelated variables called principal components [[1]](#endnote-1). With the PCA transform, we can make the data concentrate on some few dimensions so that we can use fewer dimensions of data but still get large amount of information.

2.1.2 Fisher Index

One way to estimate if a variable is useful for classification or not is to use the “Fisher index”. The Fisher index (FI) is defined as

where the indices 1 and 2 refer to the two categories, respectively, and refers to the mean value, refers to the variance. A variable with a high Fisher index is a good variable for classification, because the classes are well separated along [[2]](#endnote-2).

2.2 Linear Model

2.2.1 Linear Gaussian Classifier

If we have two classes with Gaussian conditional densities:

where is the mean for category , is the covariance matrix for category and is the determinant for the covariance matrix. The decision boundary is then defined by all such that the a posteriori probabilities are equal [[3]](#endnote-3). In the linear case, we assume in each class, the variable has the different means and common covariance matrix . A new observation is classified by computing the probability based on the Bayes’ Rule: . So the decision boundary between and is given by

which defines a hyperplane [[4]](#endnote-4).

2.2.2 Logistic Regression Classifier

The logistic regression is another linear parametric classifier. The logistic regression classifier builds on the insight that the a posteriori probabilities for Gaussian distributed categories is described by the logistic sigmoid function. The common logistic function is:

The parameters and are set by maximizing the likelihood for the data. Actually, this algorithm looks like that we use the regression way to do the classification task.

2.2.3 Simple Perceptron

Simple perceptron is the algorithm which uses the neural network without hidden layer. It is trained by gradient descent.

2.3 Nonlinear Model

2.3.1 KNN classifier

KNN algorithm is proposed to calculate the output based on the N nearest neighbors. The weights of the neighbor is proportional with the reciprocal of the distance between the input sample and the corresponding neighbor. So the nearer the distance is, the output value will closer to the value of that neighbor. The parameter N is used to control the over-fitting. The greater the N is, the smaller possibility of over-fitting will be.

2.3.2 Multi-layer Perceptron

Multi-layer perceptron is the algorithm which uses the neural network with hidden layer. In this project, only one hidden layer is used. Similar with the simple perceptron, MLP it is trained by gradient descent.

2.4 Cross-validation

In general, the error on the training data will be a biased estimate of the generalization error. It will tend to be smaller than the generalization error if we select our model such that it minimizes the training error. We can therefore not use the training error as our selection criteria. Instead, we must try to estimate the generalization error. One way to estimate the generalization error is to do cross-validation. This means using a test data set, which is a subset of the available data that is removed before any training is done to test the error. In this project, we use the 10-fold cross-validation to estimate the generalization error. Each loop we use 94 samples as the training set to train the model and use 10 samples as the test set to get the generalization error.

3. Data

In this project, there are 208 samples in total, the training set contains 104 of these and the test set contains the other 104 samples. Each sample has 60 variables. Before classify the data, we should do some pretreating work. We plot the histogram to see the distribution of the data. We also do the scatter plot to see the relation between two data. Finally, Fisher Index algorithm is used to test the importance of each variable.

3.1 Data distribution

First we plot the histogram to see the distribution situation of the data, and we found that most of the variables are normally distributed. Here is one example which is the distribution of the 1st variable:

Fig 3.1 Histogram for variable#1 of both classes

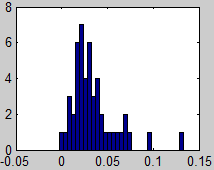
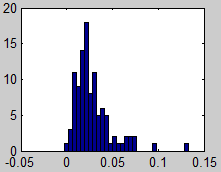


Fig 3.2 Histogram for variable#1 of Class 1

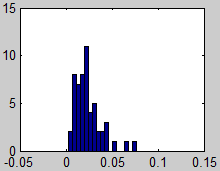


Fig 3.3 Histogram for variable#1 of Class 0

We can see that the data is roughly normally distributed. As there are only 104 samples which is a bit too few, so the histogram is not totally with the same shape of Gaussian distribution, but we still can see it’s similar.

3.2 Relation between the variables

To see the relation between the variables, correlated or uncorrelated, we do the scatter plot to analyze that. Here are two examples:

Fig 3.4 Scatter Plot between the variable#2 and variable#3

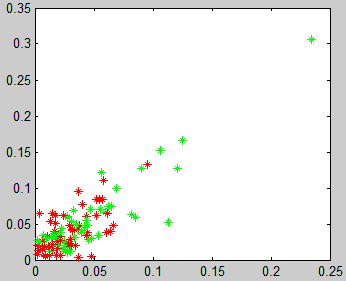
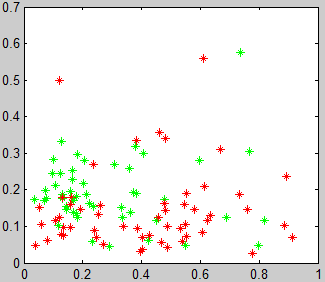


Fig 3.5 Scatter Plot between the variable#9 and variable#37



In both figures above, we use two different colors refer to the two different classes. The left figure shows the scatter plot between the variable#2 and variable#3, and the right figure shows the scatter plot between the variable#9 and variable#37. From the left figure, we can see a strong relation between variable#2 and variable#3, while the right figure shows that variable#9 and variable#37 is almost un-correlated.

3.3 Importance of variable

Using the algorithm of Fisher Index, we can get the following result:

|  |  |
| --- | --- |
| Index | FI Value |
| 49 | 0.007747 |
| 48 | 0.005719 |
| 47 | 0.004502 |
| 36 | 0.004354 |
| 45 | 0.003784 |
| 11 | 0.003754 |
| 13 | 0.003386 |
| 46 | 0.003362 |
| 12 | 0.003341 |
| 35 | 0.00333 |
| … | … |

Table 3.1 Result of the Fisher Index

Table 3.1 lists the 10 greatest Fisher Index value of each variable sorting in a descending order. Select two un-correlated variables, for example, variable#49 and variable#12, do the scatter plot:

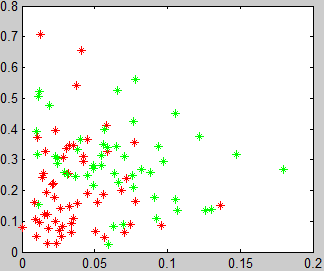


Fig 3.6 Scatter Plot between the variable#49 and variable#12

From Fig 3.6, we can see that the two classes is a bit easier to split, not like Fig 3.4 and Fig 3.5 whose data of the two classes overlap with each other.

4. Results

The results of different kinds of classifiers are shown below. All the error estimations are based on the 10 fold cross-validation (10 samples to validate, 94 samples to train). About the variable selection, we use two methods, which are the downsampling (sum the power in six bands) and Principle Components Analysis.

4.1 Linear model

4.1.1 Linear Gaussian Classifier

First we use the downsampling method to reduce the dimensions from 60 to 10. If we use all the 10 dimensions, the generalization error we get is 0.2800. Then we try to optimize the set of inputs. We use the backward elimination. Each time we remove one variable, and use the rest 9 dimensions of data. The table below shows the result:

|  |  |  |
| --- | --- | --- |
| Variable Num to Remove | Error | Error Change |
| 1 | 0.3300 | 0.5↑ |
| 2 | 0.3100 | 0.3↑ |
| 3 | 0.2400 | 0.4↓ |
| 4 | 0.2800 | 0 |
| 5 | 0.3000 | 0.2↑ |
| 6 | 0.2700 | 0.1↓ |
| 7 | 0.2600 | 0.2↓ |
| 8 | 0.3000 | 0.2↑ |
| 9 | 0.2800 | 0 |
| 10 | 0.2900 | 0.1↑ |

Table 4.1 Result of the backward elimination

From the table below, we can see that the elimination of variable#3 can decrease the error most. Then we eliminate the variable#3, and continue to eliminate the next variables until we find a least error. Finally, we find that removing variable#3 and variable#6 can help us get the best result. The error is 0.2400.

Then we use the PCA transformation to pretreat the data. For the 60 dimensions of data, we found that it cannot all be used to do the principle component analysis, because there are some noise in the data, we will try to eliminate some of them. We use the Fisher Index result in Table 3.1 to sort the variable. Then we use the method similar with the forward selection to add the variable one by one and build the linear Gaussian classifier to test the generalization error. Then we get the following result shown in Fig 4.1. The horizontal axis refers to the quantity of the variables be selected, and the vertical axis refers to the error. We found the satisfied quantity should be 6 or around 20. Since 6 is too small, which means if we choose only 6 dimensions of variable, we will lose too much information. So we prefer the quantity around 20. After the deeper test of the quantity around 20. We found that the quantity 22 is the best one. Next, we try to figure out how many principle components to use. Again, we build the linear Gaussian classifier use 1-22 principle components and test its error. The result is shown in Fig 4.2. As we can see in Fig 4.2, the error get minimum. So we choose 2 principle components to use.

The 2-demension PCA result is plotted in the Fig 4.3. We can see that the area of the two different colors of dots overlap quite a little, which means it’s easy to split them.

Fig 4.2 Error of different quantity of principle components to use when use 22 variables to do PCA

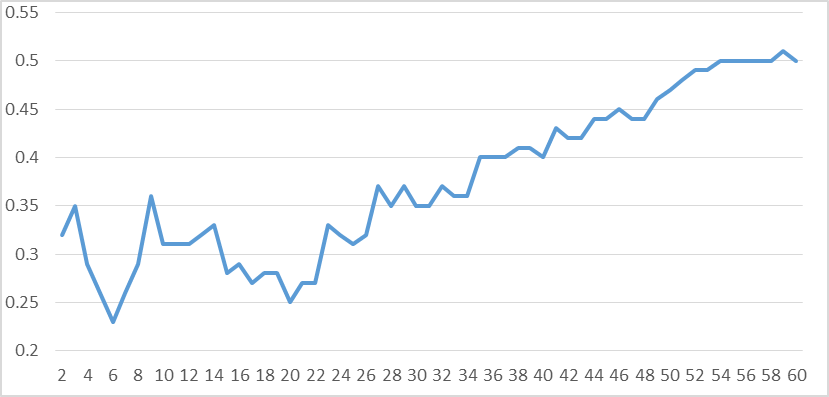
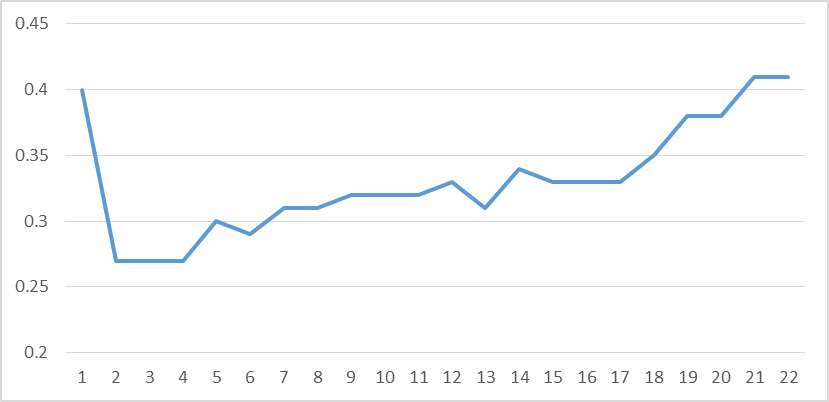


Fig 4.1 Error of different quantity of variables selected to do PCA

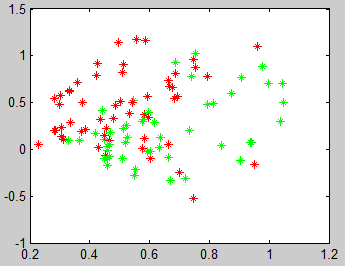
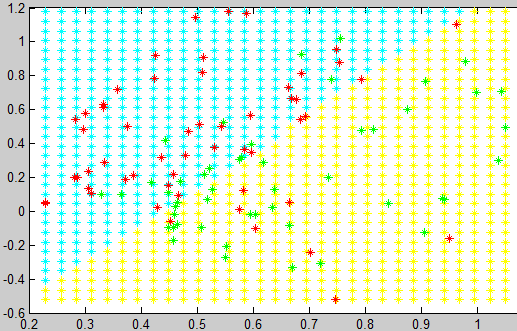


Fig 4.3 Scatter plot of the 2 dimension of data got from PCA

Based on these 2-dimension PCA data, the generalization error is 0.27. Here is the figure shows the classifier.

Fig 4.4 Linear Gaussian Classifier (PCA)

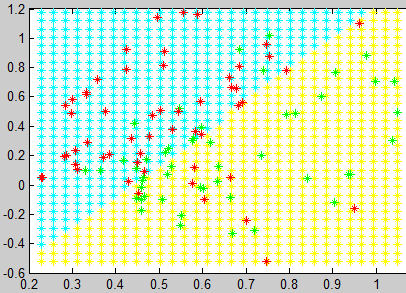


4.1.2 Logistic Regression Classifier

With the whole downsampling data, the generalization error is 0.2700. If we eliminate the variable#3 and variable#6, the error is still 0.2700, but the variance of the generalization is decreased from 0.0490 to 0.0290, which means the confidence of the generalization error is increased.

With the PCA result, the generalization error is still 0.2700, but the variance of the generalization even smaller, which is decreased to 0.0179. Here is the figure shows the classifier.

Fig 4.5 Logistic Regression Classifier (PCA)



4.1.3 Simple Perceptron

In the simple perceptron, we use 80% of the training data (94 samples of 104 samples) to build the model, 20% of the training data as the validation data to determine when the training process ends. Since the BP neural network is easy to get a local minimum point, not a general minimum point, for each target neural network model, we will train 10 times consecutively, and take the one with the best performance. With the whole downsampling data, the generalization error is about 0.2000. If we eliminate the variable#3 and variable#6, the error is about 0.1900.

With the PCA result, the generalization error is still 0.2100. Here is the figure shows the one of the classifiers and its training process:

Fig 4.6 Simple Perceptron Classifier (PCA)

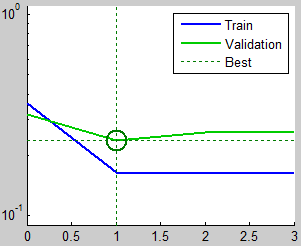
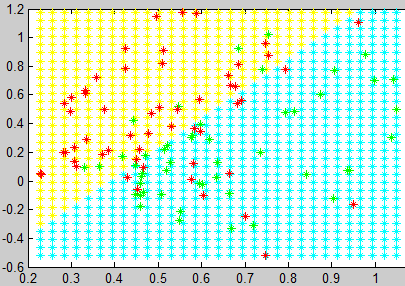


Fig 4.7 Training process

4.2 Non-linear model

4.2.1 KNN Classifier

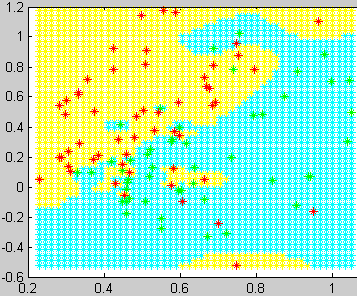


Fig 4.8 KNN Classifier (PCA)

With the whole downsampling data, the generalization error is minimum when N=1, which is 0.4200. If we eliminate the variable#3 and variable#6, the generalization error is minimum when N=5, which is 0.3500.

With the PCA result, the generalization error is minimum when N=5, which is 0.3300. Here is the figure shows the classifier:

4.2.2 Multi-Layer Perceptron

In the Multi-layer perceptron, we use 80% of the training data (94 samples of 104 samples) to build the model, 20% of the training data as the validation data to determine when the training process ends. Since the BP neural network is easy to get a local minimum point, not a general minimum point, for each target neural network model, we will train 10 times consecutively, and take the one with the best performance. One of the most important work in training MLP is to decide how many hidden nodes to use. With the whole downsampling data, after our trying, we found that the network with 10 hidden nodes usually has the relatively better performance, the generalization error is about 0.2000. If we eliminate the variable#3 and variable#6, the error can be decreased to about 0.1600.

With the PCA result, after our trying, we found that the network with 7 hidden nodes usually has the relatively better performance, the generalization error is 0.1700. Here is the figure shows the one of the classifiers and its training process:

Fig 4.9 One of the MLP Classifier (PCA)

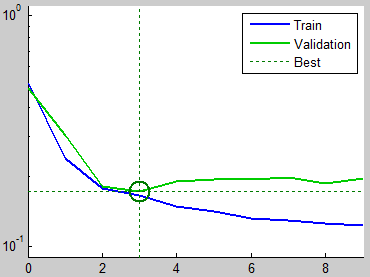
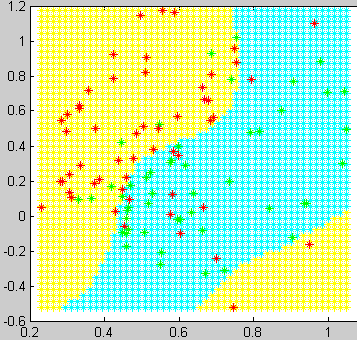


Fig 4.10 Training Process

1. Principal component analysis. <http://en.wikipedia.org/wiki/Principal_component_analysis> [↑](#endnote-ref-1)
2. Some issues for the projects. Page9-10. Thorsteinn S. Rögnvaldsson, September 14, 2005 [↑](#endnote-ref-2)
3. Introduction to classification. Page 5-6. Thorsteinn S. Rögnvaldsson, January 21, 2001 [↑](#endnote-ref-3)
4. Learning in linear systems. Page 17. Antanas Verikas, 2013 [↑](#endnote-ref-4)