Pattern Recognition

DEFAULT OF CREDIT CARD CLIENTS

Master's in Informatics engineering | May 26, 2016
António Simões- 2012145253 | Joao Campos- 2012144140

Introduction

For this assignment, we were asked to create several classifiers that try to predict if the clients of a bank will be able to pay their next credit card bill. Our work is based on real data that comes from a dataset collected in Taiwan. Several features were collected, although, only a few are used in the automated process. We allow the user to go through the GUI manually or to perform all the operations automatically to assure the best result possible. We guide the user through our binary classifier by displaying all the phases necessary to create and design a classifier. The GUI starts with an option to load a custom dataset, followed by a pre-analysis of the entire data, feature selection, an optional feature scaling and reduction, the percentage of the dataset to be randomly selected for the train and test step, and what distance metric to use.

To compare between all the possible combinations of classifiers, we use a confusion matrix, which provides all the necessary information combined in a single plot.

Feature Selection

In this part of the project we do a pre-processing of the data, called feature selection. In this phase, sort the irrelevant features from the features that might help the classifier discern between the 2 types of clients (said in the previous topic), the ones that will default their next payment and the ones that will be able to pay in time.

With this in mind the data is processed in 3 stages:

- Kruskal-Wallis Analysis

In this stage we used the Kruskal-Wallis test to see if the different features and the target (the expected output of each instance) follow the same distribution.

With this test we try to see what are the most discriminant features that are available in the data to use in training of the classifier.

To discard a feature the p-value of the feature has to be less than 0.05.

- Correlation coefficient

In the next stage we use a correlation coefficient based on the features that weren't discarded in the Kruskal-Wallis test.

This coefficient gives us a measure of the correlation and dependence (statistical relationship) between two features. This was done between the feature who was most discriminate in the Kruskal-Wallis test and all the rest that didn't get discarded. With this we see which features are statistically different from the most discriminate feature. The ones who have a positive coefficient get discarded, the other are used in the next stage.

- Minimum Redundancy Maximum Relevance

With the remaining features we used the algorithm mrmr (Minimum Redundancy Maximum Relevance) to see which features were better to classify the target. This method sorts the features to see which ones have the minimum redundancy between the target, and the maximum relevance to the target.

After some test with these methods we reduce the number of features from 23 to 5 (LIMIT_BAL, PAY_0, PAY_2, PAY_3, PAY-AMT3).

Feature Scaling

Feature scaling is a step in machine learning that can be considered optional, although it is usually applied before reducing the problem dimensionality and training phase. It is used to standardize the range of the data independently of its domain.

The data can be scaled before or after the feature reduction and, as said before, because it can be an optional step, we allow the user to create a classifier without scaling any data.

As for the methods used, we provide:

- **Rescaling**: change the range of the features to be in the range [0, 1].

$$x' = \frac{x - \min(x)}{\max(x) - \min(x)}$$

- **Standardization**: change the range of each feature to have zero-mean.

$$x' = \frac{x - \bar{x}}{\sigma}$$

Feature Reduction

After analyzing and selecting which features offer the best class separability to classify our original dataset, we proceeded with a further reduction of the data, this time reducing its original dimensions using unsupervised Principal Component Analysis (PCA) or supervised Linear Discriminant Analysis (LDA). The option to use none is also available. The use of feature reduction grant us the capability to lower the dimensionality of the previously selected data while preserving important characteristics because we assume that at least a few features provide redundant information.

In the experiments of the PCA reduction method, the dimensions used were based on the Kaiser Factor analysis criteria method for PCA (displayed in the interface). The final dimensions are 6 and 1. In case of the LDA reduction method, the dimensionality bares in mind the number of classes, in this case there are only two. So the tests done use only one dimension (number of classes - 1).

Distance Measurement

For the Minimum Distance Classifier in this project we used Euclidean Linear Discriminants and Mahalanobis Linear Discriminants to create decision rules to classify the data.

- Euclidean Linear Discriminants

  Linear discriminant function:

$$g_k(\mathbf{x}) = (\mathbf{w}_k \cdot \mathbf{x}) + w_{k,0}$$

$$\text{with } \mathbf{w}_k = \mathbf{m}'_k \text{ and } w_{k,0} = ||\mathbf{m}_k||^2$$

  In this case, **mk** is the average point of class k and **x** is the point/client we want to classify.

- Mahalanobis Linear Discriminants

  Linear discriminant function:

$$g_k(\mathbf{x}) = (\mathbf{w}_k \cdot \mathbf{x}) + w_{k,0}$$

$$\text{with } \mathbf{w}_k = \mathbf{C}^{-1}\mathbf{m}_k \text{ and } w_{k,0} = -0.5\mathbf{m}'_k\mathbf{C}^{-1}\mathbf{m}_k$$

  In this case, **mk** is the average point of class k and **x** is the point/client we want to classify and **C** is the covariance of the features of the selected instances to train the classifier.

The difference between the two metrics is that Euclidian is a straight forward metric that calculates the distance between two points using the square root of difference between the two and doesn't have anything more in account.

The Mahalanobis uses the same type of metrics as the Euclidian metric but has the variance between the two points in to account.

Classifier

Minimum distance classifier

This is a simple classifier; first we compute the average point for each class of our training data set. Then we test it with never seen data using cross validation. For each test case we test the distance between the test point and the average point of each class. The class corresponding to the shortest distance is considered the class of the new point. For the distance between two points, several metrics can be used. In this project, only two are considered, the Mahalanobis and the Euclidean distance.

Naive Bayes classifier

This classifier is based on the Bayes theorem; it applies a density estimation of the data and assumes that the predictors are conditionally independent, given the class. The algorithm that Matlab uses is the following:

1. It estimates the densities of the predictors within each class.
2. Models posterior probabilities according to the Bayes rule. Using :

$$\hat{P}(Y = k|X_1,..,X_p) = \frac{\pi(Y=k)\prod_{j=1}^{P} P(X_j|Y=k)}{\sum_{k=1}^{K}\pi(Y=k)\prod_{j=1}^{P} P(X_j|Y=k)},$$

Where,

- k = 1,....,K.
- *Y* is the random variable corresponding to the class index of an observation.
- $X_1,...,X_P$ are the random predictors of an observation.
- $\Pi(Y=k)$ is the prior probability that a class index is *k*.

3. Classifies an observation by estimating the posterior probability for each class, and then assigns the observation to the class yielding the maximum posterior probability.

This is the training stage of the model, then we use cross validation with different data to Test the accuracy of the classifier.

K NEAREST NEIGHBOOR

New data points are classified according to the nearest k neighbors as defined by some metric distance. We found that a rule of thumb is to use K = $\sqrt{n}$, n = number of data points in the training set. To avoid the curse of dimensionality, feature selection and/or feature reduction was used.

SUPPORT VECTOR MACHINE

A SVM is built using the example data points. The algorithm tries to maximize the margin between the classification vector and the vector that defines each class. We tried several kernel functions to maximize the data but there was virtually no improvement to using a polynomial or Gaussian or other mapping function. We settled with a linear SVM.

Results

The results below have the dataset split into two groups, the training and the test set and are randomly selected with the same amount of instances for each class and then assigned 70% of it to training set and 30% to the test set. The results concern only the test set and were obtained after 30 runs. The accuracy is measure as the sum of true positive plus true negative divided by the total amount of test cases. For the propose of not having a large table we use the best Scaling method (Rescaling) before the classification this was chosen because of better classification done in tests.

Feature Reduction	Features	Classifier	Distance	Accuracy (%)	False Positives (%)	False Negatives (%)	True Positives (%)	True Negatives (%)
Auto	5	Minimum Distance	Euclidean	57,9	15,7	26,3	23,7	34,3
PCA(18)	23	Minimum Distance	Euclidian	58	16,3	25,6	24,4	33,7
LDA(1)	**23**	**Minimum Distance**	**Euclidean**	**96,2**	**2,1**	**1,8**	**48,2**	**47,9**
Auto	5	Minimum Distance	Mahalanobis	55,8	14,9	29,3	20,7	35,1
PCA(18)	23	Minimum Distance	Mahalanobis	59,6	23,0	17,4	32,6	27
LDA(1)	**23**	**Minimum Distance**	**Mahalanobis**	**96,6**	**1,6**	**1,8**	**48,2**	**48,4**
Auto	5	Naive Bayes	None	68,3	7,7	24	42,3	26
PCA(18)	23	Naive Bayes	None	57,4	29,4	13,2	20,6	36,8
LDA(1)	**23**	**Naive Bayes**	**None**	**96,9**	**1,4**	**1,6**	**48,6**	**48,4**
Auto	5	KNN	None	65,9	8,5	25,6	41,9	24,4
PCA(18)	23	KNN	None	67,9	14	18,1	36	31,9
LDA(1)	**23**	**KNN**	**None**	**94,7**	**2,6**	**2,7**	**47,4**	**47,3**
Auto	5	SVM	None	66,1	6,5	27,4	43,5	22,6
PCA(18)	23	SVM	None	67,7	5,4	27	44,6	23
LDA(1)	**23**	**SVM**	**None**	**96,7**	**1,3**	**2,1**	**48,7**	**47,9**

Table 1. Results obtained from running the statistics on our classifier GUI for all possible combinations and using Rescaling before the classification.

For the tests there are two things in mind, one the overall accuracy of the classifier and two the overall percentage of False Positives versus the True Negatives. This is because of the problem at hand, the Bank losses money to a client who is unable to pay his credit card debt, so the classifier has to be prepared to classify the clients as better as possible and try to give less False Positives as possible.

Overall the results give a low False Positive rate (except for the 5 cases highlighted in red), the best results were archived using the LDA method for reducing the features.

Conclusions

The best classifier was the SVM; this is because of the overall accuracy coupled with the low rate of False Positives, which in this problem domain is what we are prioritizing. False positive in this context means that a client that is going to pay the bank is predicted as not being able to fulfill the next payment. LDA proved to be the most effective reduction method, immediately followed by our automated feature selection. Combining the automated feature selection and LDA or PCA, our preliminary results show that the combination of both are worse than LDA or PCA individually.

Acknowledgments

Credits to:

- Laurens van der Maaten, Delft University of Technology
  (http://homepage.tudelft.nl/19j49)
  For his Matlab Toolbox for Dimensionality Reduction

- Quan Wang. Kernel Principal Component Analysis and its *Applications in Face Recognition and Active Shape Models*.
  (arXiv:1207.3538 [cs.CV], 2012)
  For his kPCA and PCA code.

- Hanchuan Peng, Fuhui Long, and Chris Ding and its *Feature selection based on mutual information: criteria of max-dependency, max-relevance, and min-redundancy.*
  (http://penglab.janelia.org/proj/mRMR/)
  For MRMR code.