Masters' Degree in Informatics Engineering
Report

Pattern Recognition Techniques Off-line Signature Recognition

Fábio Miguel Ferreira Pedrosa
fmfp@student.dei.uc.pt

Marco António Machado Simões
msimoes@student.dei.uc.pt

Nuno António Marques Lourenço
naml@student.dei.uc.pt

December 30, 2010

FCTUC DEPARTAMENTO
DE ENGENHARIA INFORMÁTICA
FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE DE COIMBRA

Contents

List of Figures

Chapter 1

Introduction

The signature verification problem is very important, since every time we need to authorize important operations, like an opening of a bank account, we need to perform signatures. As the techniques to perform forgeries have improved in the last years, the pattern recognition techniques have improved, either in the feature extraction or in the classifiers fields.

In this report we will present the results that we obtained using some of the best known features to off-line signature verification. This features are briefly introduced, as well as some other features we implemented. Besides this feature we did a study on which was the best classifier. The remainder of this report is the following: in the next section we will explain the Feature Extraction process, in the section 3 we expose the feature selection process. In the section 4 we present the classifiers used, as well as the results obtained. In the section 5 we will show a *graphical user interface* that we made for a simple usage of the system. We end this report with section 6, making some conclusions about the work done.

Chapter 2

Feature Extraction

The feature extraction process intends to reduce the dimensionality of the problem, but at the same time keep the essential information presented in each image.

In the beginning of the semester, a set of commonly used features were given to us. In the first phase of the project, our objective was to implemented that set of features and try to reproduce some of the results present in [1]. Then we were encouraged to explore new features that could improve the results.

In this section we will briefly explain the features that we implemented, based on [1], and we will give more details on the ones that were made by us.

2.1 Modified Direction Feature

This technique is a combination of two other techniques called "Direction Feature" and "Transition Feature" [2]. The first extracts the direction transitions based on the replacement foreground pixels by the direction values of the segment were they are present. The second obtains the locations of transitions between foreground and the background. The image is scanned in all possible directions: left to right, right to left, top to bottom, bottom to top. Every time a changes happens, the ration between the location and length of the image is determined and saved as a feature[3]. Later, and to reduce the number of values we have, we use an averaging technique, so that we have a number of features that are, at the same time, sufficiently descriptive, and treatable by our machines. After all this process, we ended up with 120 values that compose this feature.

To finalize the description of this feature, just mention a implementation

detail, which is related with the pre-processing of the image, just before we extract the feature. In this pre-processing we had to implement an thinning algorithm to obtain the skeleton of the image, so that the values of this feature were more accurate. After implementing it, we realized that our programming environment, which is Matlab, had already a function that was able to do the same. Since the Matlab version were faster by far, we decide to replace our version of the thinning algorithm.

2.2 Centroid Feature

Another feature, that was already documented, is the centroid feature. This one relates the dominant angle of the signatures pixels distribution. To calculate it, we split the image in two, and find the centroids of the both parts. Then, and using simple euclidean geometry, and some trigonometry we calculate the angle between the horizontal and the line that connected the two centroids. The total values of this feature is resumed to 1.

2.3 Tri - Surface Feature

This feature gives the area occupied by a signature. However, and in order to increase accuracy, the image is divided in 3 similar parts and the area of each part in calculated. In the end of the extraction of this feature we will have 3 values, corresponding to the area of the 3 parts.

2.4 Length Feature

This feature represents the length of the signature after scaling all the signatures in the database. As one can expect the amount of information given by this feature is reduced to one value.

2.5 Sixfold - Surface Feature

This feature differs form the one presented in section 2.3 in mainly in one way: the number of values given are doubled, from 3 to 6. To calculate it, first, and just like in section 2.3, we split the image in three equal parts, and then find the centroid of each. Then we calculate the area above and under each centroid, and store that values.

2.6 Best-Fit Feature

By finding the upper points and lower points of the signature border, a line is found that best approximates these points 2.1. The feature considered is the angle between both lines and the X-axis and the surface area enclosed between the two lines. These three values are normalized accordingly: angles are divided by 90º, and the area is divided by the image total area, therefore making it a percentage of the image size.

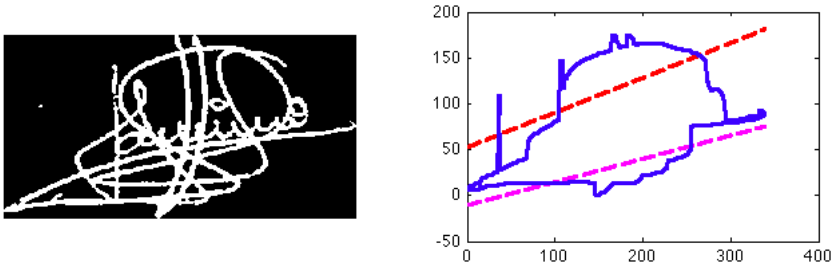

Figure 2.1: Approximation lines for the signature upper and lower border

2.7 Stroke Feature

The stroke feature is the first one made by us. In this feature, we analyze the relation between the signature and its skeleton. We came up with this feature, because when someone is signing a document, they do not worry about the stroke of the lines. Otherwise, when a forger is copying a signature, he has mimic the exactly stokes of the genuine signature, so he has to be more careful, leading to a more precise stroke.

2.8 Histogram Feature

Based on a hunch that the amount of pixels in each row, and each column of the image were a good characterization of a specific signature, we developed this feature. We get the histogram of pixels on each row and build a histogram row-wise 2.2, and also build a column-wise 2.3. To reduce the dimensions of this signal (dimension of the image), we get a polynomial approximation using the least-squares method and save only this coefficients. We currently save 10 coefficients for each signal (9th order poly).

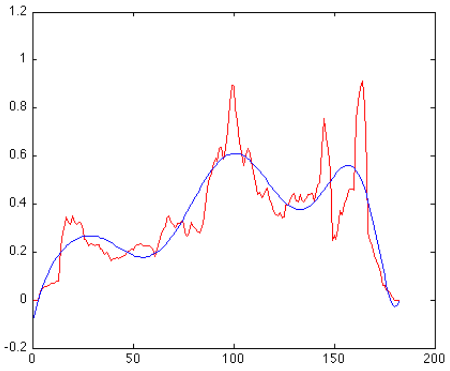

Figure 2.2: 9th order polynomial approximation for the signature row-wise histogram

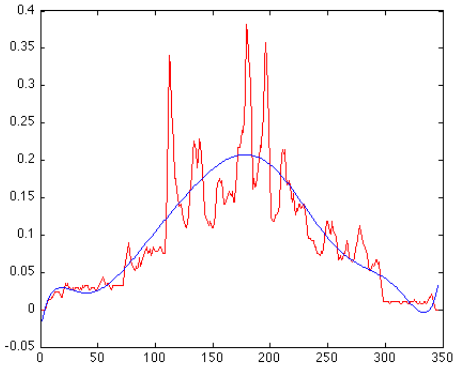

Figure 2.3: 9th order polynomial approximation for the signature column-wise histogram

2.9 Splitting Feature

This feature is based on the vertical splitting procedure presented in [4]. The idea is to use the centroid point to split the image in two parts, and then calculate the new centroids for each part. Then, split each part again and calculate the new centroid points. The figure 2.4 presents the final points of an example signature. The feature values consist of the centroids' coordinates for the seven final points.

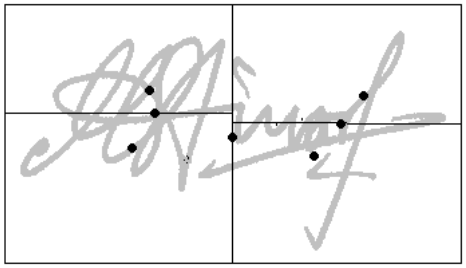

Figure 2.4: Vertical splitting example in a signature image

Chapter 3

Feature Selection

In this section we presented the study made in order to choose the best configuration of features the given problem. The experiences are detailed and a statistical analysis is used to substantiate our results.

3.1 Experimental Analysis

In order to determine what were the best combination of features to our problem we decide to make an experiment with all possible combinations of features. Since we have 9 features, we have $\binom{9}{1} + \binom{9}{2} + \binom{9}{3} + \binom{9}{4} + \binom{9}{5} + \binom{9}{6} + \binom{9}{7} + \binom{9}{8} + \binom{9}{9} = 511$ possible combinations.

To run this tests we choose the support vector machine classifier, since after a quick preliminary analysis we conclude it was the classifier with the best results and the one that consumed less time to train and classify.

To add significance to the results we ran every configuration 30 times (5 x 6-fold cross validation).

3.1.1 Results

As the reader can see in 3.1, the number of possible combinations is very large. So, to enable the reader to see the results, we split them in 5 partitions, with about 100 combinations each. This results are presented in the figures 3.1, 3.2, 3.3, 3.4 and 3.5.

Then, and to check what was the best feature combination, and assuming that our data did not followed a normal distribution, we applied the Kruskal-Wallis test, to see what was the best features configuration. Note that we worked with a confidence interval of 95%.

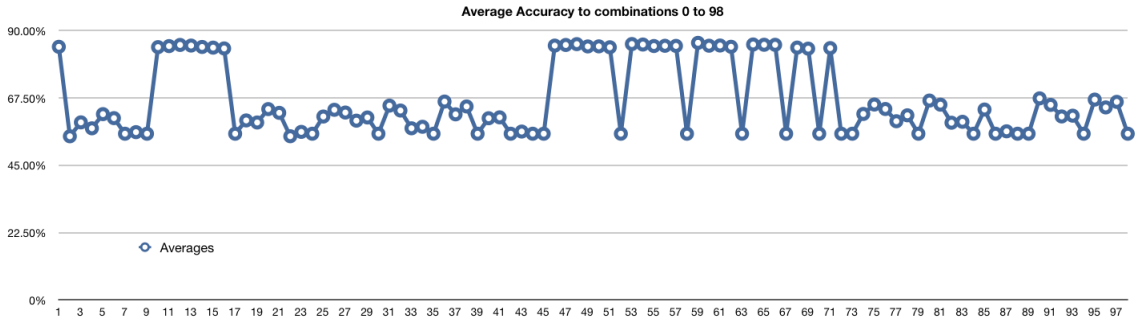

Figure 3.1: Average Accuracy to combination 0 to 98

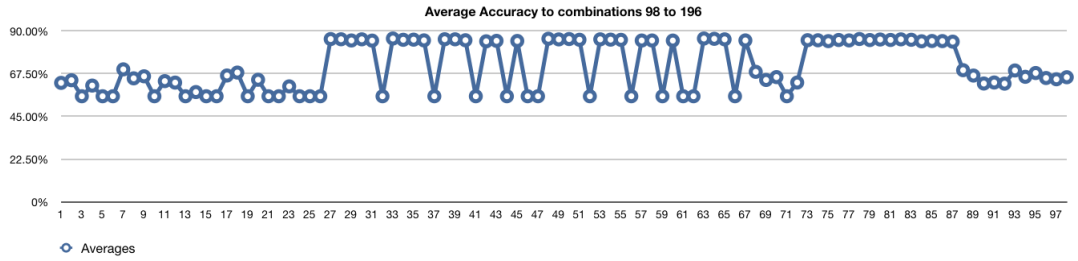

Figure 3.2: Average Accuracy to combination 98 to 196

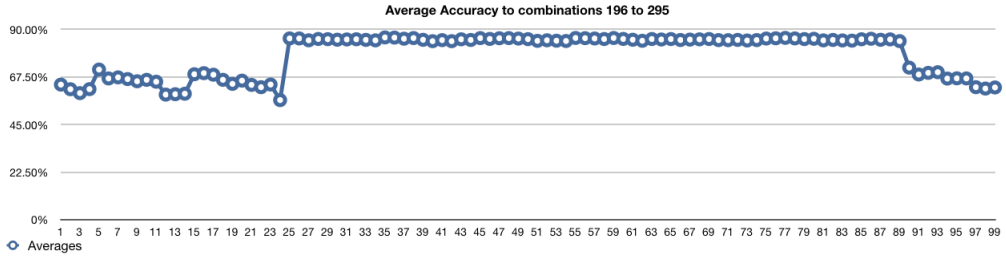

Figure 3.3: Average Accuracy to combination 196 to 295

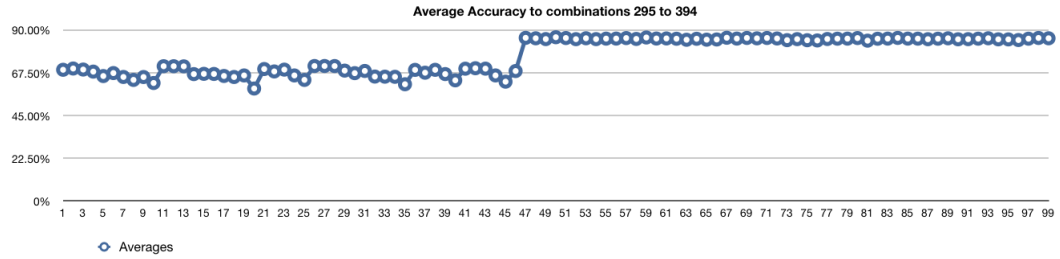

Figure 3.4: Average Accuracy to combination 295 to 394

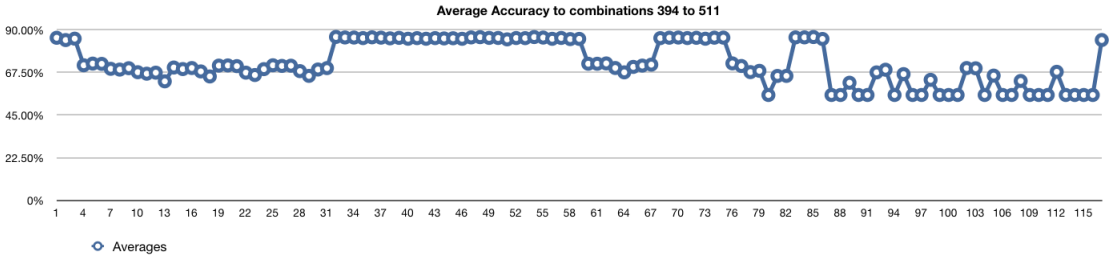

Figure 3.5: Average Accuracy to combination 394 to 511

After applying these tests, we checked the results table and concluded that, at level of significance of 95%, the combinations were different between each other3.6, and the best one, which is the one with the best mean rank, is the combination: *'mdf' 'centroid' 'length' 'sixfold' 'bestfit'* with an accuracy average of 86.20%. It is interesting to verify that, without the statistical test, the best feature combination (in terms of accuracy) is *'mdf' 'centroid' 'trisurface' 'length' 'sixfold' 'splitting'* with an average of 86.28%, bigger than the 86.20% obtained after the Kruskal-Willis analysis. This reflects the importance of the statistical significance analysis, which removes outliers and considerates the variance in the rank classification.

Kruskal-Wallis ANOVA Table

Source	SS	df	MS	Chi-sq	Prob>Chi-sq
Columns	2.66572e+11	510	5.2269e+08	13613.65	0
Error	3.35886e+10	14819	2.26659e+06		
Total	3.0016e+11	15329			

Figure 3.6: Statistical significance analysis using Kruskal-Wallis test

3.2 Automatic Methods

In order to check the best feature combination, we have tried some automatic combinations. This means that, instead of considering the features 'mdf', 'bestfit', etc, each one consisting of several values, we now consider each value as a feature. So, we have a dimension of 169 features.

We base this tests of the concept of the more different the features are, the more value they give to the classifications. So, we want to consider the most different features to the classification.

In order to verify which ones are those best features we ordered them in ascending order of total covariance. This means that, the first feature has the lowest covariance with the other features and the last feature has the biggest covariance values.

We then run a SVM classifier starting with the first feature and incrementing, five by five, the features by the order previously defined. We have run each one 30 times (5 x 6-fold cross-validation) to give statistical significance to the results.

The same procedure was made considering the mutual information instead of the covariance.

3.2.1 Results

For each method (covariance and mutual information) we have got 34 different combinations of features (1 to 166, 5 by 5), with 30 results of each one. We needed first to check if the results are different from each other, or if they have the same median (which would not bring significance to the results). So, because we do not know whether the distributions are normal or not, and there are two much combinations to check normality for each one, we used the Kruskal-Wallis statistical test.

As we can see in the Kruskal-Wallis table for the covariance feature analysis 3.7, the distributions are statistically different. Analyzing the mean ranks result, the best configuration is with the least correlated 151 features, with an accuracy of 85.93%. A detailed verification can be checked in the figure 3.8, which contains a boxplot of the different configurations.

The mutual information verification got worst results. They are statistical significant (we followed the same statistical analysis procedure as for the covariance) as the reader can check in the Kruskal-Wallis table 3.9. The best result has an average accuracy value of 61,38%, which is much lower than the obtained by the covariance method or by the experimental analysis of the extracted features. A detailed verification can be done by checking the figure 3.10.

Kruskal-Wallis ANOVA Table

Source	SS	df	MS	Chi-sq	Prob>Chi-sq
Columns	8.47317e+07	33	2567627.05	976.43	0
Error	3.69425e+06	986	3746.7		
Total	8.84259e+07	1019			

Figure 3.7: Kruskal-Wallis statistical test result for features results with covariance analysis

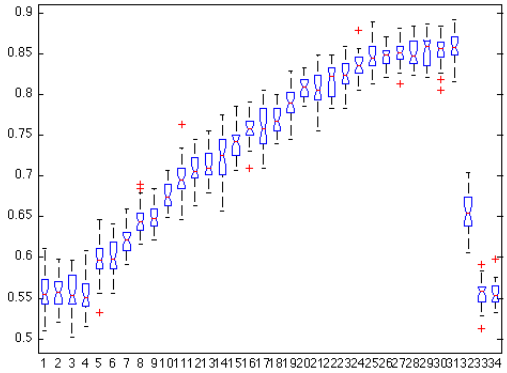

Figure 3.8: Boxplot for features results with covariance analysis

Kruskal-Wallis ANOVA Table

Source	SS	df	MS	Chi-sq	Prob>Chi-sq
Columns	7.44281e+06	33	225539.6	85.86	1.36553e-06
Error	8.08927e+07	986	82041.2		
Total	8.83355e+07	1019			

Figure 3.9: Kruskal-Wallis statistical test result for features results with mutual information analysis

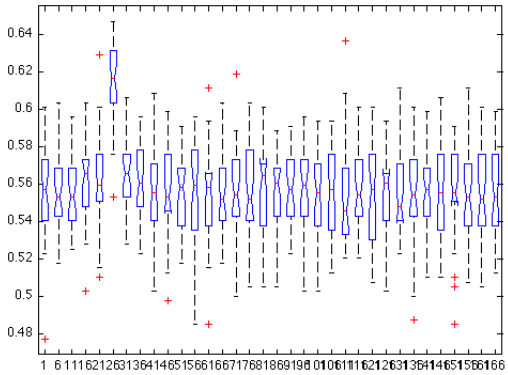

Figure 3.10: Boxplot for features results with mutual information analysis

3.3 Dataset Size Analysis

The given dataset is very large, including 16200 different signatures. Applying our different classifiers to this whole data results in long training periods and generalization penalties.

To reduce the amount of sets (folders of 54 signatures) we used 44 sets of signatures allowing us to compare previous results [1].

A comparison was made 3.11 using the four essential metrics of classifier performance (precision, recall, accuracy and f-score) varying the number of sets used. The sets used are picked in the original sequence (using 44 sets means the first 44 folders of the original dataset).

There can be seen a degradation of performance using more sets, reaching a global minimum when using the full set. Finding a way to generalize results to the full set while keeping not going down is still a challenge for future work.

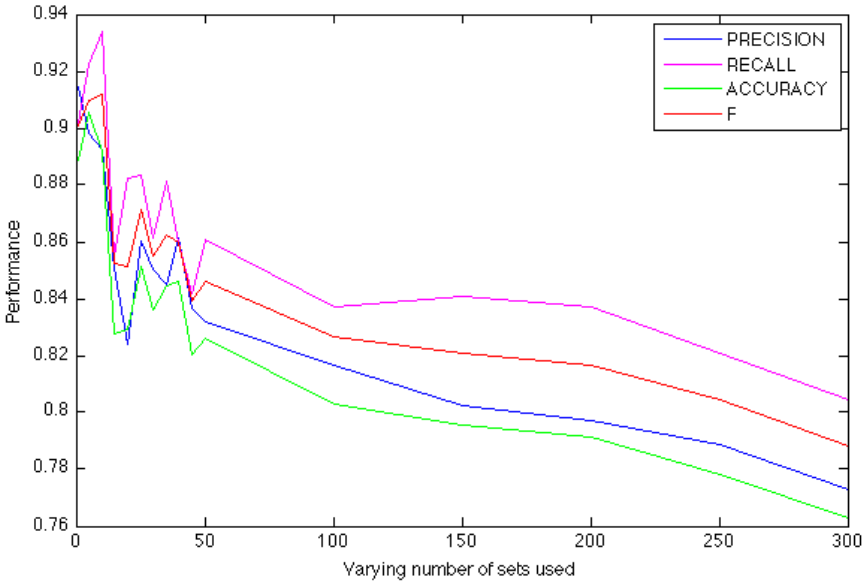

Figure 3.11: Comparison of performance varying the number of sets used (6-fold cross-validation)

Chapter 4

Classification

4.1 Classifiers

In this section we will briefly present the classifiers used in our project.

4.1.1 Support Vector Machines

This are the state of the art classifiers for pattern recognition systems and almost every framework that works with pattern recognition has an implementation of support vector machines (SVM).

There basis is linear algebra, more specifically, finding the hyperplane that maximizes the distance between the positive class and the negative one. Such type of SVMs are called the *Hard-Margin*. As consequence, because they only try to find the hyperplane that makes the condition true, sometimes we could not find a solution to the problem. Nevertheless, and to solve this problem, there are other type of SVMs called *Soft-Margin* that gives a penalty to the points of the classes that are closest to the hyperplane of the separation, so relaxing the initial premisses of the SVMs.
The last method to find the hyperplane consists in projecting the data of the classes in space of greater dimensionality, find that hyperplane in there, and then project it in the original space of the classes.

In our project we used the implementation of SVMs presented in the stprtools toolbox.

4.1.2 K-Nearest Neighbors

This pattern recognition algorithm classifies the data based on the training samples that, and in the feature space, are close to it. k-NN is a lazy learning

method, meaning that generalization beyond the training data is postponed until the system receive a query.

The k-NN algorithm is among the simplest ones in machine learning: it classifies the objects based on the majority of the votes of the closest neighbors. This votes are made taking into account the k nearest neighbors.

4.1.3 Radial Basis Function Networks

This type of networks are similar to ones presented in section 4.1.6. However, the activation function is a radial basis function. Normally this networks have 3 layers4.1: the input, where the data to be analyze enters, the RBF layer, and the output where, in case we are dealing with a classification problem, the result goes out.

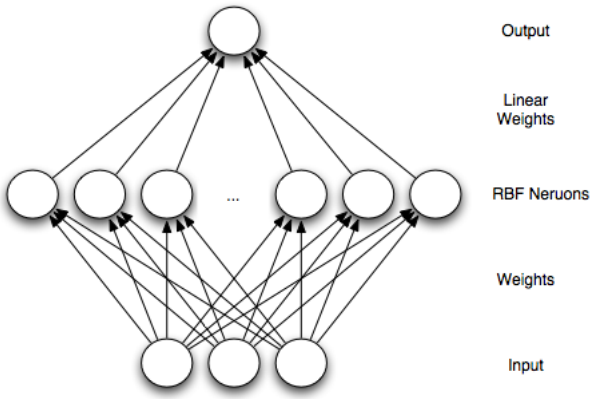

Figure 4.1: Radial Basis Function Network

In the RBF layer, the function that activates the neurons is a RBF, similar to the one depicted in 4.2.

4.1.4 Fisher Linear Discriminant

Fisher Linear Discriminant is a method used in pattern recognition as well in machine learning. This method tries to find a linear combination of features, in order to separate the classes of the problem. Despite we used the FLD as a classifier, it could be used to reduce the dimensions of the problem.

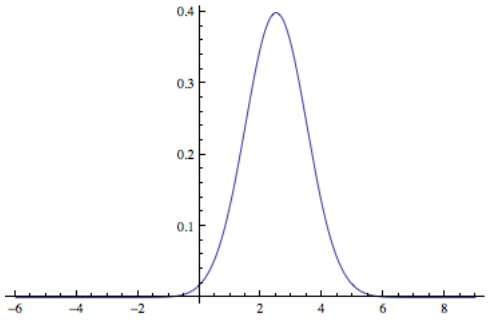

Figure 4.2: Example of RFB function with center in 2.5 and spread 1.

4.1.5 K-Means

The aim of this algorithm is to joint a set of observations into a number k of clusters. In our case, we want to group the observations, meaning signatures, into 2 clusters, one for the genuine and other for the forgeries.

4.1.6 Multi-Layer Neural Network

This classifier tries to mimic what happens in human brain. It is made of simple entities called perceptrons, that have weighted inputs, inside of it verifies if the input is larger enough to activate it. If this is true, then the perceptron is activated, and will produce an output, which in turn can be connected to another perceptron. Like this we can have a large mesh of perceptron which resembles, like was said before, what happens in human brain. To use this classifier we used the neural network toolbox of the Matlab framework.

4.1.7 Naive Bayesian Classifier

This classifier is a probabilistic one, and it consists in applying the *Bayes' Theorem*, with strong, and naive assumptions about the independence of the probabilities. This means that the classifier assumes that the presence of a particular feature of a class is not related with other feature of any other class. This is a not very good assumption, since it more likely to find the word computer in a book that teaches in pattern recognition techniques, than, for instance in a book that talks about poems.

It is a method of supervised learning, and it uses less training data than some other classifiers (e.g Neural Networks).

4.1.8 Template

We have tried a probabilistic method as classifier. Our idea was to create a map of the most common areas of the signatures' images, even for the value ones and for the forged ones. With the train images, we create this maps. Then, to classify a new signature we check the overlap of the signature with this mapped areas. If the overlap is bigger in the valid map than in the forged map, the signature is classified as valid, otherwise is classified as forged. A template map example of forged images can be viewed in figure 4.3.

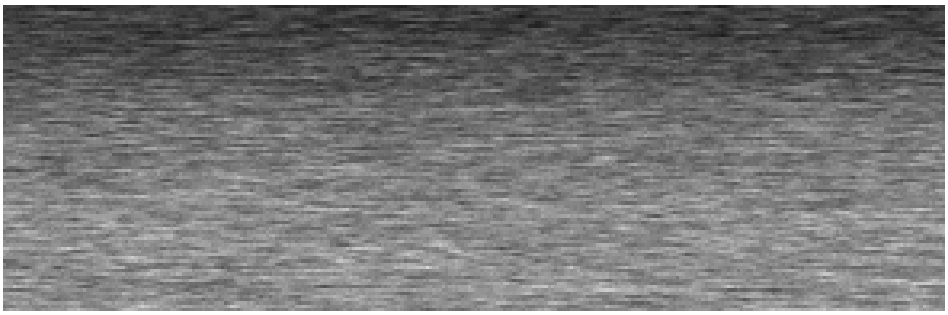

Figure 4.3: Template map for forged images (example)

4.2 Experimental Analysis

In this section we will present the results obtained by all the classifiers that we implemented. This experimental analysis was made using the feature study made in section 3. We ran every classifier for 30 times, and then we made a statistical analysis of the results.

4.2.1 Results

Since we have multiple performance indicators, we decide to present only the most important ones, like accuracy, f-score, precision and recall of each classifier.

As we can see in figure 4.4, the classifier with best accuracy is the Support Vector Machine. This result is supported in figure 4.5, which shows that the F-score for this classifier is one of the bests, only overran by the RBF neural network, but this one has a much bigger standard deviation, which makes us prefer the SVM classifier.

We can also see that the linear classifiers (FLD, FLDQP) have a low performance. The precision and recall (figures 4.6 and 4.7, respectively) of

these classifiers are both bellow than 70%. The bayes classifier has all results arround 80%, which makes it a stable solution.

The clustering techniques (c-means and knn) have not achieved good results. The c-means is the worst classifier in all comparisons. The KNN classifier performs better, but even so it is not in the group of the best classifiers.

The neural network shows some instability in the results. With a big recall but low precision, it has a F-score smaller that 70% and a even smaller accuracy.

The template has also low accuracy and a F-score around the 70%. The small precision and high recall makes it unreliable for a automatic offline validation system.

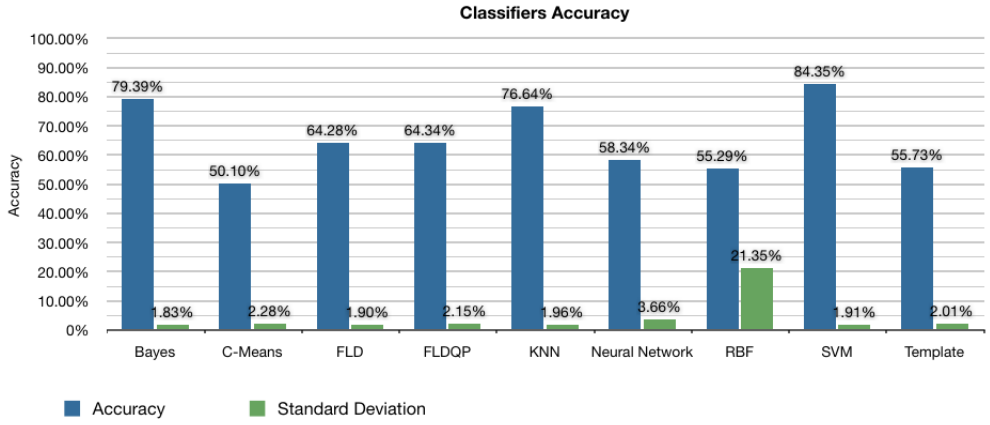

Figure 4.4: Classifiers accuracy

We have also implemented ROC curves but, because we only have them to SVM, FLD and Bayes classifiers we decided to not use them to compare all classifiers. Even so, figure 4.8 presents an example for the FLD classifier.

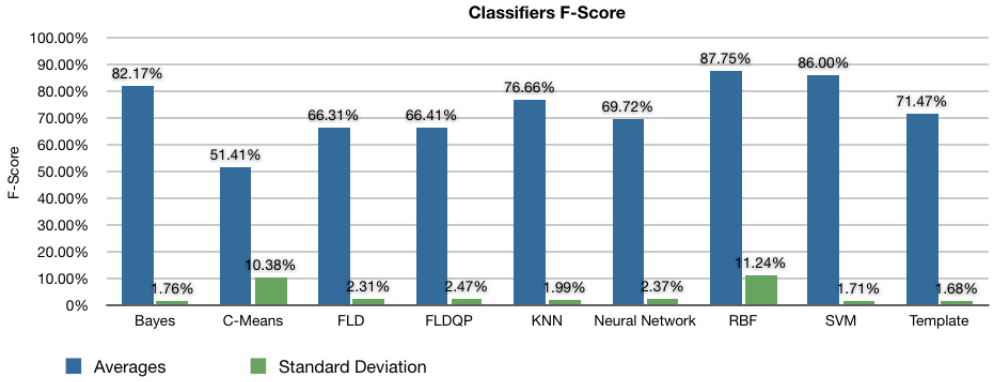

Figure 4.5: Classifiers F-Score

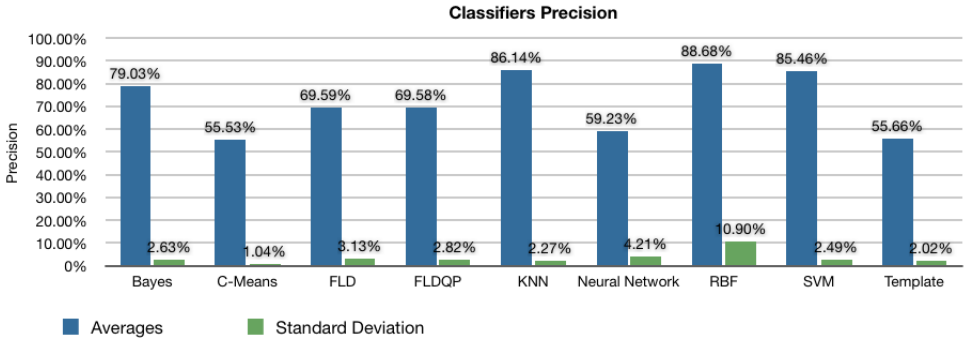

Figure 4.6: Classifiers precision

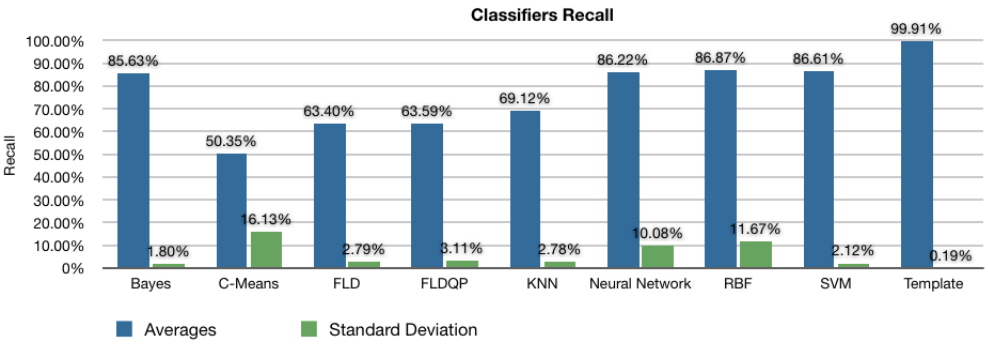

Figure 4.7: Classifiers recall

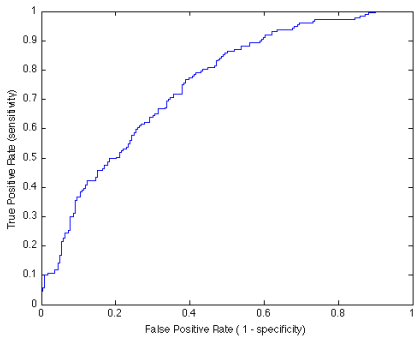

Figure 4.8: ROC curve example for the FLD classifier

Chapter 5

Graphical User Interface

To simplify our application use we built a graphical interface using matlab GUIDE that should allow any user to test our classifiers and general work done.

By running the "mainwindow" command in matlab the GUI window should popup like seen in 5.1 and allow the user to make several customizations before running a classifier.

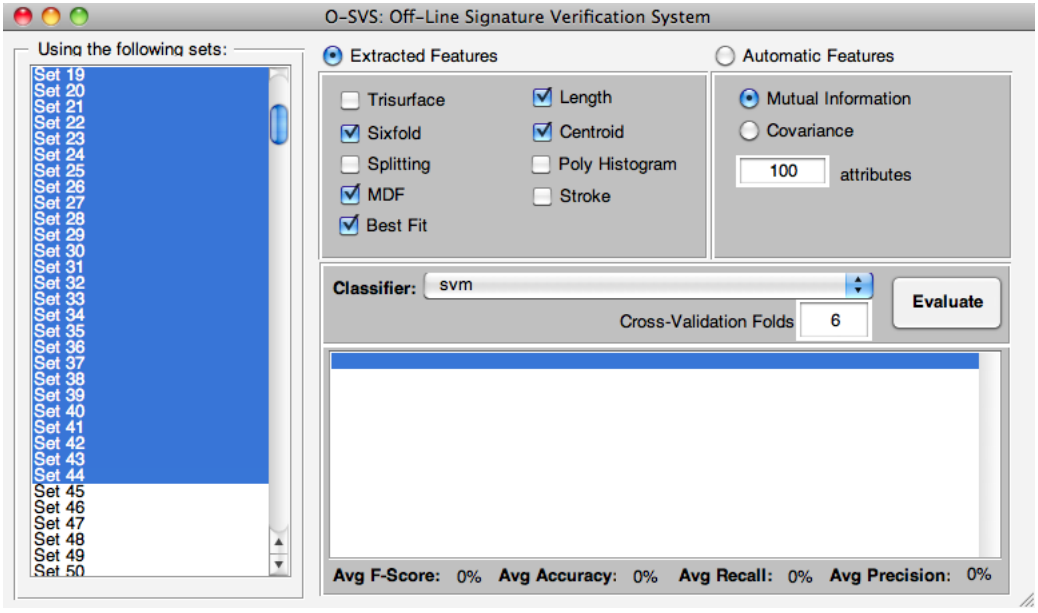

Figure 5.1: Graphical Main Window for O-SVS Application

In this window 5.1 the user can select the sets of signatures to use in training and testing, select the classifier to use and select features to use.

There are two different types of feature to use: the extracted features are those produced off-line for the original images, and can be selected independently by the user (the best combination is already selected by default) and automatic selected features using the automated methods described before (mutual information and covariance). In the automated case the user can pick how many attributes (a feature can have several attributes) the automated method should select.

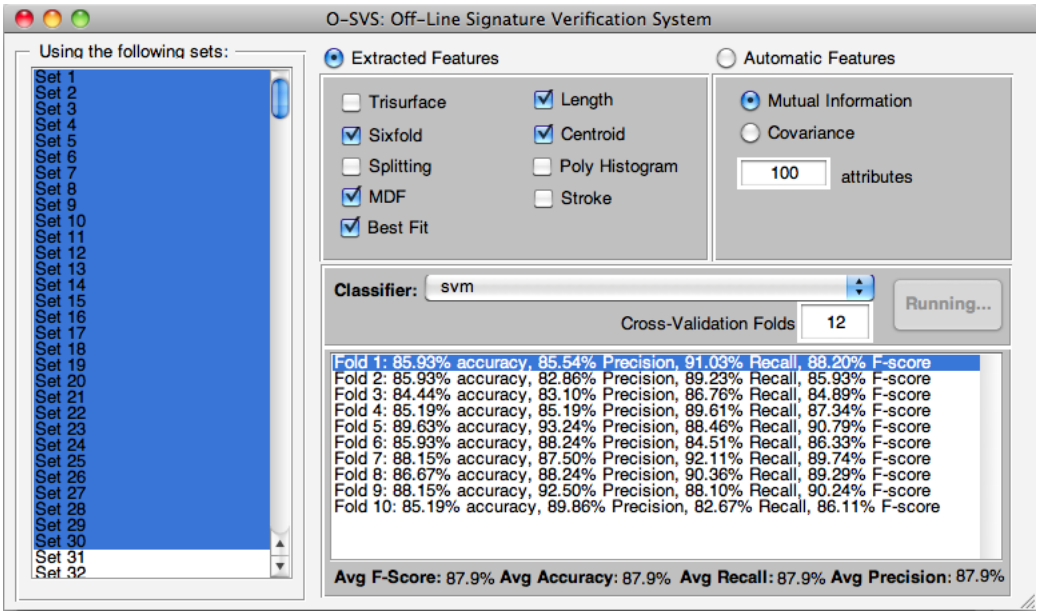

Figure 5.2: Window while running a 12 fold SVM classifier

Chapter 6

Conclusions

Our best classification results are provided by a Support Vector Machine with the features of *'mdf' 'centroid' 'length' 'sixfold' 'bestfit'*, with an average accuracy of 86,2%.

We have implemented some features different of the provided in the bibliography, which have shown some good results (the splitting feature, in example). In the end we have verified that the MDF is the most discriminant feature, because even when we only consider that feature we have got results around 80%.

We tried also to create different classification methods, with the template classifier. This proved to not work very well because it considered the images by itself, not extracting nor selecting any features of the images. This proved to have a dimensionality problem (curse of dimensionality). It was very heavy in terms of memory and computation time and, in the end, did not bring good results. This re-enforced the need of the feature extraction and selection in this problem.

We think that trying to generalize the signature recognition system to identify if any signature is valid is not the best way. It would be better to have a system for each person's signature validation. In this way, the system could specify itself for each person instead of generalize for any person. Even though we understand the need of a single system and the need of this generalization.

We achieved lower results than the presented in bibliography. However, the results are close to those ones. Although, we are not sure about the sets used in the bibliography. They used 44 sets from a version of this database dated from 2006. Our current database has 300 sets and may have suffered changes in the sets from 2006, and even though we do not know what 44 sets were used. So, this is not a fair comparison, but is the one possible. We supported every result with a statistical analysis to ensure the quality of the

conclusions extracted.

Bibliography

[1] Armand, Stephane, Michael Myer Blumenstein, and Vallipuram Muthukkumarasamy: *Off-line signature verification using an enhanced modified direction feature with single and multi-classifier approaches.* 2007.

[2] Blumenstein, M., B. Verma, and H. Basli: *A novel feature extraction technique for the recognition of segmented handwritten characters.* Document Analysis and Recognition, International Conference on, 1:137, 2003.

[3] Gader, P D, M Mohamed, and J H Chiang: *Handwritten word recognition with character and inter-character neural networks.* IEEE transactions on systems man and cybernetics Part B Cybernetics a publication of the IEEE Systems Man and Cybernetics Society, 27(1):158–164, 1997.

[4] Majhi, Banshider, Y Santhosh Reddy, and D Prasanna Babu: *Novel Features for Off-line Signature Verification.* International Journal, I(1):17–24, 2006.