Assessment of offline Digital Signature Recognition Classification Techniques

Dina Darwish

Assistant Professor, International Academy for Engineering and Media Science – Egypt
6th October city, Egypt

Abstract

The digital signature verification has become an interesting domain, which is widely needed. The usage of online and offline digital signatures has been spreaded worldwide due to the increase of use of bank transactions and user authentication and other similar activities. This requires the creation and the diversification of new online and offline signature verification methods. The signature verification methods contain both online (or dynamic) and offline (or static) signature verification methods. In this paper, an offline digital signature verification technique is proposed, that depends on extracting several features from the signatures to be used during simulation. Some signatures were used for training and others were used for testing only. Different methods such as, vectors manipulation, ensemble classification using boosted trees, and bagged trees, were used in this paper during simulation to obtain results.

Keywords: *Signature Verification, Offline Digital Signature, Features Extraction, Vectors Manipulation, Ensemble Classification, Bagged Trees.*

1. Introduction

The growth in today's online and offline transactions that includes banking transactions has posed the question of how to make secure online and offline signature verification techniques, to eliminate the possibility of personal information theft. There are a different number of personnel characteristics that can be used to identify each person, such as, voice, lip movements, hand geometry, face, iris, retina, fingerprint, and others. These characteristics are called biometrics, and these biometrics can be used to distinguish between one person and another. But the most commonly used biometric nowadays in e-commerce and banking activities is the signature recognition.

The signature can be defined as follows; "the name of a person written with his or her own hand; or the act of signing one's name", according to the American Heritage Dictionary. There is a second definition of signature, which is related to the whole process of signing, it means, that the way the signature is made and the characteristics of the signature, including velocity, pen pressure, stroke, and others are unique to every person.

The first definition, is close to the definition of offline signature, which treats the signature as a two-dimensional image with static characteristics, that does not contain any time-related information. The second definition, is close to the definition of online signature, and is based on dynamic characteristics of the process of signing, such as velocity, pen pressure and others.

The signature verification is a typical pattern recognition task. But both types of signatures; online or offline; use different techniques to verify signatures based on either static or dynamic characteristics.

The task of signature verification includes extracting some characteristics from the recorded information of the signature, and further, comparing them with the characteristics of the reference signature. Let us make a brief survey on different signature recognition and verification techniques used.

Various methods have been implemented for creating features from the signature image, which can be grouped into two main categories: direct methods and transform methods. Direct methods allow generating features directly from image pixels such as grid-based information, pixel density, gray-level intensity, texture… etc. In contrast, transform methods need a transformation of the image into another domain in which features could be created. Fourier, Wavelet, Radon transforms are the most popular methods for creating features [1][5]. Hence, another transform has been proposed namely the contourlet transform (CT) [6].

The main advantage of the CT is the ability to capture significant information about an object.

Furthermore, it offers a flexible multiresolution, local and directional image expansion. These properties are interesting to exploit more specifically for the handwritten signature verification since the signature contains often special characters and flourishes [7].

In [2], this paper describes a method for verification of signatures after extraction of features based on clustering techniques. Clustering involves dividing a set of data points into non-overlapping groups, or clusters, of

points, where points in a cluster are "more similar" to one another than to points in other clusters. In [3], this paper, two methods are proposed to track the variations in signatures. Given the set of training signature samples, the 1rst method measures the positional variations of the one-dimensional projection profiles of the signature patterns; and the second method determines the variations in relative stroke positions in the two-dimension signature patterns. In [4], this paper evaluates the performance of an Error Back Propagation (EBP) Artificial Neural Network (ANN) for authenticating the signatures. The work done has provided encouraging results and has re-confirmed the ability of Artificial Neural Networks to recognize patterns and in this case their skill to generalize. An efficient Static Signature Verification (SSV) system that consists of rigorous preprocessing and feature extraction followed by a classifier is used.
In [8], a paper presents a method for verifying handwritten signatures by using NN architecture. Various static (e.g., area covered, number of elements, height, slant, etc.) and dynamic (e.g., velocity, pen tip pressure, etc.) signature features are extracted and used to train the NN. In [9], a paper is primarily focused on skilled forgery detection. It emphasizes on the extraction of the critical regions which are more prone to mistakes and matches them following a modular graph matching approach.
In section 2, we described the features extracted to be used for signature recognition. In section 3, the signature recognition classification techniques were described. In sections 4, the simulation results were discussed and analyzed. In section 5, the conclusion is presented. And finally, the references are cited.

2. Features Extraction for signature recognition

The features needed to be extracted to identify signature are [10]:

1 - The new curve of the signature after rotating the original curve of the signature points around the center x and y coordinates of the original signature curve based on making the original signature curve rotate around its center x and y coordinates, to make the new signature curve that will be used in pattern recognition.

2 - The number of pixels in the signature based on calculating the total number of pixels of the signature.

3 - The occupancy Ratio of the signature to the whole image which is described as :

Occupancy ratio = total number of pixels of the signature/total number of pixels of the signature image * 100

4 - The minimum Eigen value of the signature curve

Where the eigen values of a matrix A are obtained from the solution of the characteristic equation:

$$\det(A - \lambda I) = 0\,. \qquad (1)$$

where det is the determinant of the matrix (A - λI) and I is the $n \times n$ identity matrix, λ is the eigen value

5 - The maximum height of the signature is based on the following:

Maximum height of the signature = maximum x coordinate of signature – minimum x coordinate of signature

6 - The maximum width of the signature is based on the following

Maximum width of the signature = maximum y coordinate of signature – minimum y coordinate of the signature

7 - The Euclidean distance between every two consecutive points in the signature curve

8 - The angle between every two consecutive points in the signature curve

9 - The height to width ratio of the signature

3. Assessment of Signature Recognition classification techniques

Three different signature recognition classification techniques were used to recognize signatures. These techniques were:

1) Vectors manipulation
2) Ensemble classification using boosted trees
3) Tree Classification using bagged trees.

Assessment of these techniques is based on simulation in which we used 500 signatures for 100 persons, each person has 5 signatures. We used 60% of the signatures for training, and the other 40% were used for testing. We used MATLAB 2011 during simulation.

3.1 Vectors Manipulation Technique

Vectors manipulation is based on finding the differences between each vector to be tested and each pattern vector used to identify one signature. Each tested vector is compared with the 300 reference patterns representing the 100 persons, the smallest difference between any tested vector and reference pattern vector, means that the tested vector belongs to the N person having this reference pattern vector.

For each vector the following rules applies,

For i=1:vector length

For j=1:300

$D_{ixj} = abs\ (\ VP_{ixj} - VT_i\ )$

End

End

Where, *VP* is the vector reference pattern
VT is the tested pattern

We calculate the sum of each vector in the difference matrix

$S_j = sum(D_{ixj})$

Where, *i* represent rows from 1 to vector length and *j* represent columns from 1 to 300

S_j is a vector containing the sum of each column

Then,

Find $min(S_j)$

Where, *j* represents the column of the person *y* for example

By this way, we classify all the 500 patterns, either being only test patterns or patterns used as references by finding the least difference between any test pattern and any reference pattern.

Figure (1) shows the proposed simulation for assessment of vector manipulation technique.

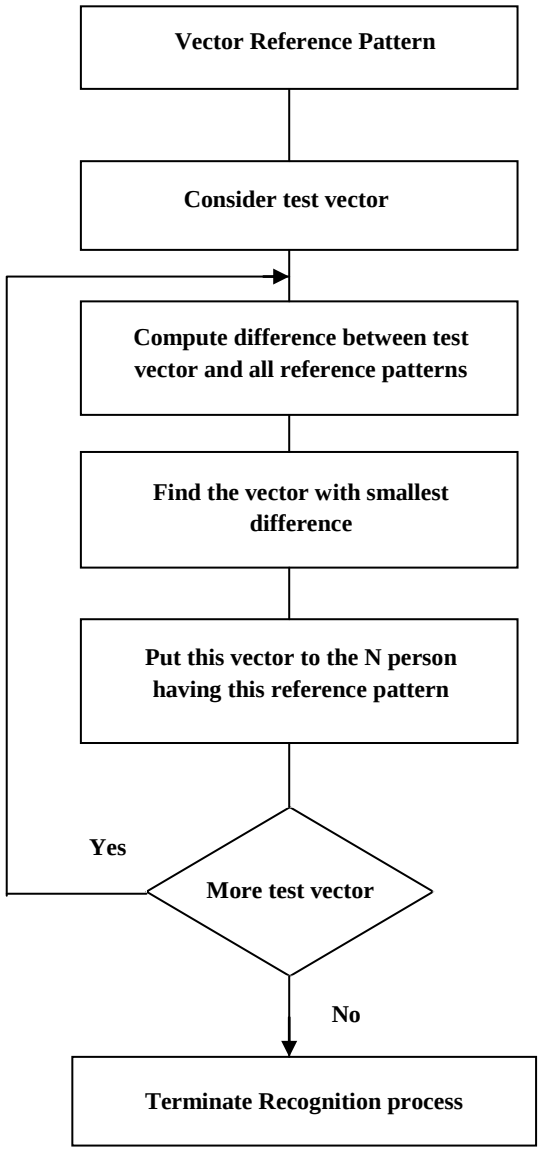

Fig.1 Simulation chart of vector manipulation technique

3.2 Ensemble classification using boosted trees

(A) Common types of ensembles

1. Bayes optimal classifier

The Bayes Optimal Classifier is an optimal classification technique. It is an ensemble of all the hypotheses in the hypothesis space. On average, no other ensemble can outperform it, so it is the ideal ensemble. Each hypothesis is given a vote proportional to the likelihood that the training dataset would be sampled from a system

if that hypothesis were true. To facilitate training data of finite size, the vote of each hypothesis is also multiplied by the prior probability of that hypothesis. The Bayes Optimal Classifier can be expressed with following equation [11]:

$$y = \operatorname{argmax}_{c_j \in C} \sum_{h_i \in H} P(c_j|h_i)P(T|h_i)P(h_i) \quad (2)$$

where y is the predicted class, C is the set of all possible classes, H is the hypothesis space, P refers to a *probability*, and T is the training data. As an ensemble, the Bayes Optimal Classifier represents a hypothesis that is not necessarily in H. The hypothesis represented by the Bayes Optimal Classifier, however, is the optimal hypothesis in *ensemble space* (the space of all possible ensembles consisting only of hypotheses in H).

2. Bootstrap aggregating (bagging)

Bootstrap aggregating, often abbreviated as *bagging*, involves having each model in the ensemble vote with equal weight. In order to promote model variance, bagging trains each model in the ensemble using a randomly drawn subset of the training set. As an example, the random forest algorithm combines random decision trees with bagging to achieve very high classification accuracy.

3. Boosting

Boosting involves incrementally building an ensemble by training each new model instance to emphasize the training instances that previous models mis-classified. In some cases, boosting has been shown to yield better accuracy than bagging, but it also tends to be more likely to over-fit the training data. By far, the most common implementation of Boosting is Adaboost, although some newer algorithms are reported to achieve better results.

4. Bucket of models

A "bucket of models" is an ensemble in which a model selection algorithm is used to choose the best model for each problem. When tested with only one problem, a bucket of models can produce no better results than the best model in the set, but when evaluated across many problems, it will typically produce much better results, on average, than any model in the set.

The most common approach used for model-selection is cross-validation selection. It is described with the following pseudo-code [11]:

For each model m in the bucket:

Do c times: (where 'c' is some constant)

Randomly divide the training dataset into two datasets: A, and B.

Train m with A

Test m with B

Select the model that obtains the highest average score

Cross-Validation Selection can be summed up as: "try them all with the training set, and pick the one that works best".

5. Stacking

The crucial prior belief underlying the scientific method is that one can judge among a set of models by comparing them on data that was not used to create any of them. This same prior belief underlies the use in machine learning of *bake-off contests* to judge which of a set of competitor learning algorithms is actually the best fit in selected domains.

This prior belief can also be used by a single practitioner, to choose among a set of models based on a single data set. This is done by partitioning the data set into a *held-in* data set and a *held-out* data set; training the models on the held-in data; and then choosing whichever of those trained models performs best on the held-out data. This is the cross-validation technique, mentioned above.

Stacking (sometimes called *stacked generalization*) exploits this prior belief further. It does this by using performance on the held-out data to combine the models rather than choose among them, thereby typically getting performance better than any single one of the trained models. It has been successfully used on both supervised learning tasks (regression) and unsupervised learning (density estimation). It has also been used to estimate Bagging's error rate.

Because the prior belief concerning held-out data is so powerful, stacking often out-performs Bayesian model-averaging. Indeed, renamed *blending*, stacking was extensively used in the two top performers in the recent Netflix competition.

(B) Simulation of Ensemble Classification Technique

MATLAB 2011 is used to simulate the ensemble technique for which figure (2) shows the information necessary to create an ensemble [12].

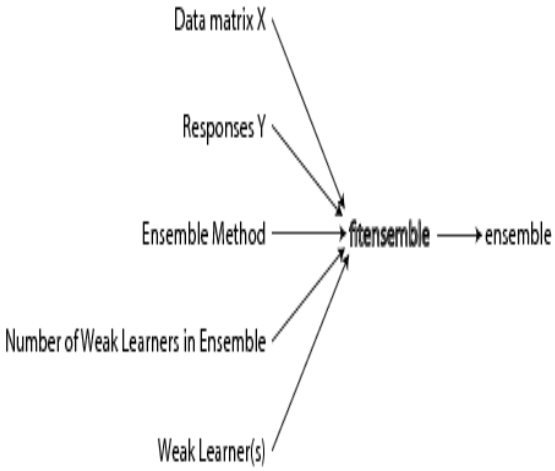

Fig. 2 Elements of the simulation technique

For all classification or nonlinear regression problems, follow these steps to create an ensemble [12]:

1. Put Predictor Data in a Matrix
2. Prepare the Response Data
3. Choose an Applicable Ensemble Method
4. Set the Number of Ensemble Members
5. Prepare the Weak Learners
6. Call fitensemble

Ensemble Algorithms

- AdaBoostM1
- AdaBoostM2
- Bag
- GentleBoost
- LogitBoost
- LPBoost
- LSBoost
- RobustBoost
- RUSBoost
- Subspace
- TotalBoost
- AdaBoostM1

Where, LSBoost is the used algorithm, and is described as follows:

LSBoost (least squares boosting) fits regression ensembles. At every step, the ensemble fits a new learner to the difference between the observed response and the aggregated prediction of all learners grown previously. The ensemble fits to minimize mean-squared error.

You can use LSBoost with shrinkage by passing in the LearnRate parameter. By default this parameter is set to 1, and the ensemble learns at the maximal speed. If you set LearnRate to a value from 0 to 1, the ensemble fits every new learner to $yn - \eta f(xn)$, where

yn is the observed response.

$f(xn)$ is the aggregated prediction from all weak learners grown so far for observation xn.

η is the learning rate. Figure (3) shows the proposed simulation for Ensemble classification technique.

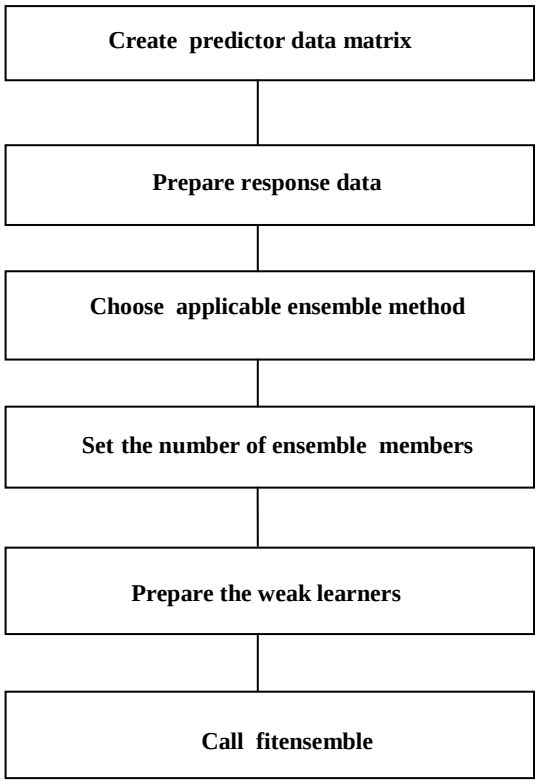

Fig. 3 Simulation chart for ensemble classification technique

3.3 Tree classification using bagged trees

Classification trees and regression trees [14] predict responses to data. To predict a response, follow the decisions in the tree from the root (beginning) node down to a leaf node. The leaf node contains the response. Classification trees give responses that are nominal, such as 'true' or 'false'. Regression trees give numeric responses.

Each step in a prediction involves checking the value of one predictor (variable). Figure (4) is a simple classification tree:

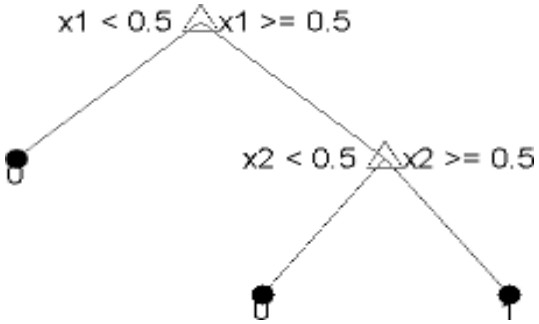

Fig. 4 Classification tree

This tree predicts classifications based on two predictors, x1 and x2. To predict, start at the top node, represented by a triangle (Δ).The first decision is whether x1 is smaller than 0.5. If so, follow the left branch, and see that the tree classifies the data as type 0.

If, however, x1 exceeds 0.5, then follow the right branch to the lower-right triangle node. Here the tree asks if x2 is smaller than 0.5. If so, then follow the left branch to see that the tree classifies the data as type 0. If not, then follow the right branch to see that the tree classifies the data as type 1.

The classification tree and the regression tree methods perform the following steps to create decision trees:

1. Start with all input data, and examine all possible binary splits on every predictor.
2. Select a split with best optimization criterion.
3. If the split leads to a child node having too few observations (less than the minimum leaf parameter), select a split with the best optimization criterion subject to the minimum leaf constraint.
   - ✓ Impose the split.
   - ✓ Repeat recursively for the two child nodes.

The explanation requires two more items: description of the optimization criterion, and stopping rule.

Stopping rule: Stop splitting when any of the following hold:

- The node is *pure*.
  - For classification, a node is pure if it contains only observations of one class.
  - For regression, a node is pure if the mean squared error (MSE) for the observed response in this node drops below the MSE for the observed response in the entire data multiplied by the tolerance on quadratic error per node (qetoler parameter).
- There are fewer than minimum parent observations in this node.
- Any split imposed on this node would produce children with fewer than minimum leaf observations.

Optimization criterion:

- Regression: mean-squared error (MSE). Choose a split to minimize the MSE of predictions compared to the training data.
- Classification: One of three measures, depending on the setting of the split criterion name-value pair provided in MATLAB 2011:
  - 'gdi' (Gini's diversity index, the default)
  - 'twoing'
  - 'deviance'

For a continuous predictor, a tree can split halfway between any two adjacent unique values found for this predictor. For a categorical predictor with L levels, a classification tree needs to consider $2L^{-1}–1$ splits. To obtain this formula, observe that you can assign L distinct values to the left and right nodes in $2L$ ways. Two out of these $2L$ configurations would leave either left or right node empty, and therefore should be discarded. Now divide by 2 because left and right can be swapped. A classification tree can thus process only categorical predictors with a moderate number of levels. A regression tree employs a computational shortcut: it sorts the levels by the observed mean response, and considers only the $L–1$ splits between the sorted levels.

The classification tree splits nodes based on either *impurity* or *node error*. Impurity means one of several things, depending on your choice of the split criterion name-value pair in MATLAB 2011:

- Gini's Diversity Index (gdi) —The Gini index of a node is [14]

$$1 - \sum_i p^2(i), \tag{3}$$

where the sum is over the classes *i* at the node, and *p*(*i*) is the observed fraction of classes with class *i* that reach the node. A node with just one class (a *pure* node) has Gini index 0; otherwise the Gini index is positive. So the Gini index is a measure of node impurity.

- Deviance ('deviance') —With *p*(*i*) defined as for the Gini index, the deviance of a node is [14]

$$-\sum_i p(i) \log p(i). \tag{4}$$

A pure node has deviance 0; otherwise, the deviance is positive.

- Twoing rule ('twoing') —Twoing is not a purity measure of a node, but is a different measure for deciding how to split a node. Let *L*(*i*) denote the fraction of members of class *i* in the left child node after a split, and *R*(*i*) denote the fraction of members of class *i* in the right child node after a split. Choose the split criterion to maximize

$$P(L)P(R)\left(\sum_i |L(i) - R(i)|\right)^2, \tag{5}$$

where *P*(*L*) and *P*(*R*) are the fractions of observations that split to the left and right respectively. If the expression is large, the split made each child node purer. Similarly, if the expression is small, the split made each child node similar to each other, and hence similar to the parent node, and so the split did not increase node purity.

- Node error — The node error is the fraction of misclassified classes at a node. If *j* is the class with largest number of training samples at a node, the node error is

$$\mathbf{1 - p(j).} \tag{6}$$

Figure (5) shows the proposed simulation for tree classification technique.

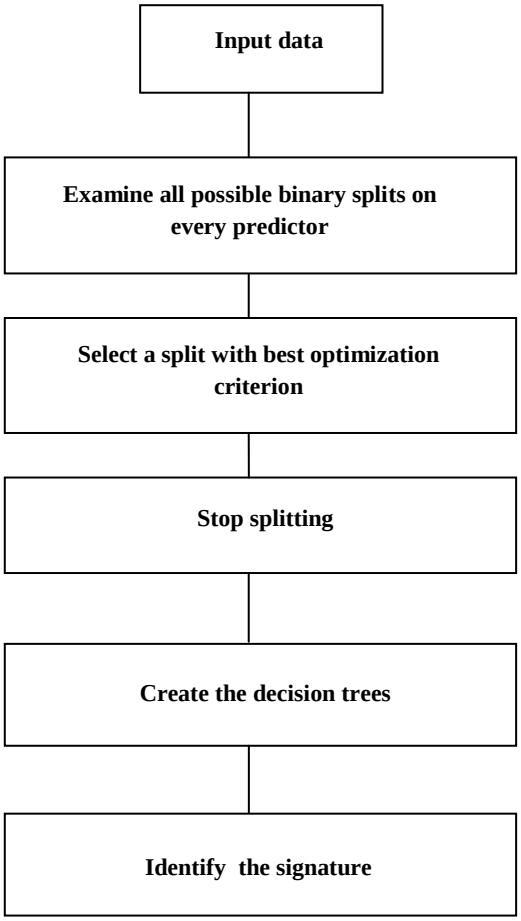

Fig. 5 Simulation chart for tree classification technique

Bagged Decision Trees

Bagging [13], which stands for "bootstrap aggregation," is a type of ensemble learning. To bag a weak learner such as a decision tree on a dataset, generate many bootstrap replicas of this dataset and grow decision trees on these replicas. Obtain each bootstrap replica by randomly selecting N observations

out of N with replacement, where N is the dataset size. To find the predicted response of a trained ensemble, take an average over predictions from individual trees.

Bagging works by training learners on resampled versions of the data. This resampling is usually done by bootstrapping observations, that is, selecting N out of N observations with replacement for every new learner. In addition, every tree in the ensemble can randomly select predictors for decision splits—a technique known to improve the accuracy of bagged trees.

By default, the minimal leaf sizes for bagged trees are set to 1 for classification and 5 for regression. Trees grown with the default leaf size are usually very deep. These settings are close to optimal for the predictive power of an ensemble. Often you can grow trees with larger leaves without losing predictive power. Doing so reduces training and prediction time, as well as memory usage for the trained ensemble.

Another important parameter is the number of predictors selected at random for every decision split. This random selection is made for every split, and every deep tree involves many splits. By default, this parameter is set to a square root of the number of predictors for classification, and one third of predictors for regression.

Several features of bagged decision trees make them a unique algorithm. Drawing N out of N observations with replacement omits on average 37% of observations for each decision tree. These are "out-of-bag" observations. You can use them to estimate the predictive power and feature importance. For each observation, you can estimate the out-of-bag prediction by averaging over predictions from all trees in the ensemble for which this observation is out of bag. You can then compare the computed prediction against the observed response for this observation. By comparing the out-of-bag predicted responses against the observed responses for all observations used for training, you can estimate the average out-of-bag error. This out-of-bag average is an unbiased estimator of the true ensemble error. You can also obtain out-of-bag estimates of feature importance by randomly permuting out-of-bag data across one variable or column at a time and estimating the increase in the out-of-bag error due to this permutation. The larger the increase, the more important the feature. Thus, you need not supply test data for bagged ensembles because you obtain reliable estimates of the predictive power and feature importance in the process of training, which is an attractive feature of bagging.

Another attractive feature of bagged decision trees is the proximity matrix. Every time two observations land on the same leaf of a tree, their proximity increases by 1. For normalization, sum these proximities over all trees in the ensemble and divide by the number of trees. The resulting matrix is symmetric with diagonal elements equal to 1 and off-diagonal elements ranging from 0 to 1. You can use this matrix for finding outlier observations and discovering clusters in the data through multidimensional scaling.

4. Simulation Results

500 patterns were used for simulation, representing 500 signatures for 100 persons, each person having 5 signatures. We used 60% of the patterns for training and the rest 40% for testing. We tested all the 500 patterns, which are all vectors of the same size. Each vector representing a signature. The results obtained are summarized in table (1).

Table 1: Simulation results

Simulation environment	Percentage of correctly classified signatures
Vectors Manipulation	77.4%
Ensemble classification using boosted trees	61.2%
Tree classification using Bagged trees	79.8%

Figure (6) shows the percentage of correctly classified signature for the three techniques. From table (1) and figure (6), we noticed that the percentage of correctly classified signatures using the ensemble classification using boosted trees is 61.2%, which represents the lowest percentage of correctly classified patterns among the three simulation environments. Using the bagged trees classification, 79.8% of the signatures were correctly classified, which represents the highest percentage of correctly classified patterns among the three simulation environments. Using the vectors manipulation, the percentage of the correctly classified patterns was 77.4%, which is closest to the percentage of the bagged trees classification.

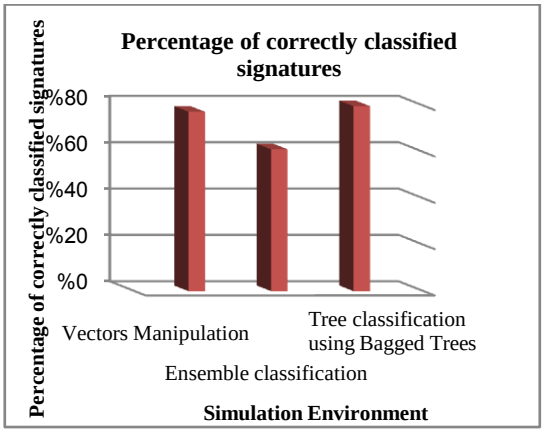

Fig. 6 Percentage of correctly classified signature

5. Conclusions

From the implemented simulation, the tree classification using bagged trees showed the best performance with signature recognition ratio of 79.8%, then, the vectors manipulation technique follows it with a signature recognition ratio of 77.4%, then, comes the ensemble classification using boosted trees with a signature recognition ratio of 61.2%, which is the least recognition ratio among the three simulation environments. b

References

[1] D. Impedovo and G. Pirlo, "Automatic Signature Verification: The State of the Art", IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews, Vol. 38, 2008, pp. 609–635.

[2] S. Biswas, T.Kim and D. Bhattacharyya, "Features Extraction and Verification of Signature Image using Clustering Technique", International Journal of Smart Home, Vol. 4, 2010.

[3] B. Fangaet al.,"Off-line signature verification by the tracking of feature and stroke positions", Pattern Recognition, Vol. 36, 2003, pp. 91 – 101.

[4] K.Dial and S. Mahesh, "Off-line Handwritten Signature Verification using Artificial Neural Network Classifier", International Journal of Recent Trends in Engineering, Vol. 2, 2009.

[5] H. Nemmour and Y. Chibani, "Handwritten Arabic word recognition based on Ridgelet transform and support vector machines", in International Conference on High Performance Computing and Simulation (HPCS), Istanbul, July 2011, pp. 357–361.

[6] M. N. Do and M. Vetterli, "The Contourlet Transform: An Efficient Directional Multi resolution Image Representation", Image Processing, Vol. 14, 2005, pp. 2091-2106.

[7] M. R. Pourshahabi, M. H. Sigari and H. R. Pourreza, "Offline Handwritten Signature Identification and Verification Using Contourlet Transform", in International Conference of Soft Computing and Pattern Recognition, Malacca, December 2009, pp. 670–673.

[8] S. Dewangan, (2011) "Neural Network-based Offline Handwritten Signature Verification System using Hu's Moment Invariant Analysis", International Journal of Engineering and Advanced Technology (IJEAT), Vol. 1, 2011.

[9] A.Gupta, G.Khandelwal and S.Chakraverty, "Offline Signature Verification Using Critical Region Matching", Journal of Signal Processing, Image Processing and Pattern, Vol. 2, 2009.

[10] Dina Darwish, "Simulation and evaluation of Signature Recognition Techniques", Egyptian Computer Journal, Vol.37, 2013.

[11] http://en.wikipedia.org/wiki/Ensemble_learning

[12] http://www.mathworks.com/help/stats/ensemble-methods.html

[13] C.Sutton, Classification and Regression Trees, Bagging, and Boosting, Handbook of Statistics, Elsevier B.V., 2005.

[14] http://www.mathworks.com/help/stats/classifica-tion-trees-and-regression-trees.html

Dina Darwish received the B.Sc. in 2004 and the M.Sc. in 2006 with honors degree from Arab Academy for Science and Technology, Egypt. She received the Ph.D. degree from Cairo University, Egypt, 2009. Her main interests include communications systems, computer networks, internet technology, and multimedia systems. She is assistant professor of communications and computer networks, International Academy for Engineering and Media Science (IAEMS), Egypt, since September 2009.