Usability of Neural Networks in Off-line Signature Verification

Adam Horvath, Bence Kovari

Budapest University of Technology and Economics,
Department for Automation and Applied Informatics
Goldmann Gyorgy square 3, Budapest, H-1111
Hungary

horviadam@gmail.com, beny@aut.bme.hu

Abstract: Signature recognition is probably the oldest biometrical identification method with high acceptance. Although automated signature verification has been studied for more than 30 years, present-day offline signature verification systems still can not achieve better error rates than 10%. Our research aims constructing an efficient off-line signature analyzer, which can reconstruct the signing method and several hidden features like velocity or strokes, and use these features by a classifier based on neural network. In contrast with typical applications, our solution is able to take both global and local features of the signatures to consideration. Our tests have shown that at some signatures we can achieve an error rate of 3%. Other ones resulted in error rates between 20 and 40%. Hence the way to improve the efficiency of our system is to improve the efficiency of all the phases it is consist of. This paper focuses on classification, the last phase of signature verification.

Keywords: signature verification, off-line, classification, neural network

1 Introduction

The aim of off-line signature verification is to decide, whether a signature originates from a given signer based on the scanned image of signature and few images of the original signatures of the signer. Unlike on-line signature verification, which requires special acquisition hardware and setup, off-line signature verification can be performed after the normal signing process and is thereby less intrusive and more user friendly. On the other hand, important information like velocity, pressure, or up- and down strokes is partially lost.

Typical signature verification approaches consist of 3 main phases. First they extract some features from the images of signatures. Then compare them and finally, they use some kind of classifier to decide whether a given signature is an

original or a forgery. We have examined several classification methods and chosen neural networks for several reasons. The purpose of the classifier is not only to classify but to determine which features are useful in classification and which are not as well. Therefore classifiers which make one-way transformation on the input data are not applicable in our case.

This paper concentrates on the final phase of signature verification. In the following section several existing signature verifiers are introduced, with a special emphasis on neural network based classification. In Section 3 the features used by classification and our classifiers are introduced. Finally experimental results are presented in Section 4.

2 Related work

Typically signature verifiers take advantage of different general properties (global features) of the signature and use them as an input for different simple classifiers [1],[2],[3]. In [4] a more complex approach can be seen, by creating a two-stage neural network classifier. Different groups of features are defined and separate MLP (multilayer perceptron) classifiers are applied to them. These MLPs are relatively simple, containing only one hidden layer. Learning is not done through backpropagation, but through the ALOPEX algorithm, which allows the network not to get "stuck" in local minima or maxima of the response function. The MLPs have a relative wide range of input parameters, in order: 16, 96, and 48 variables. The inputs of the first network are the global features of the signature. The second takes a simplified representation of the signature as an input, by creating a 12*8 grid and measuring the intensity values in each grid cell. The third network processes texture information. The output layer contains a single neutron, delivering a response value between 0 and 1 representing the similarity between the actually measured signature, and the training set. These output values are then processed by an RBF to make the final decision.

A similar approach is taken in [5]. They use global features (height-width proportion, middle point, corner points, etc.), and grid features as inputs. Tests are performed both by using simple MLP classifiers and by using SVMs. SVMs were tested with kernels with linear, polynomial, and radial basis function. The latter seemed to deliver the best results with an average error rate of 7-8% compared to the 16-22% error rates measured when using MLPs.

Another interesting approach can be found in [6]. It utilizes CGS vectors (originally developed for character recognition) to extract global features. The main idea here is, to assign a 1024 bit long binary vector to each image and compare these vectors in the later phases. Images are divided into 4x8=32 segments, and information (like concavity, gradient, structural properties) is

encoded into the vector for each segment. These vectors are then compared by several algorithms operating with vector distances. In this scenario, the SVM based solution performs poorly, with an average error rate of 46% while a Naive Bayes classifier achieved error rates between 20% and 25%

3 Proposed method

In this section features used by classification and our classifiers are introduced.

3.1 Used features

In this section three different features (baseline, skew and loops) are introduced. Although some of them may seem quiet intuitive, their exact definition and extraction is essential for later processing phases.

3.1.1 Baseline

Based on the algorithm described in [7], the upper and lower bounding envelopes (baselines) and vertical and horizontal projections are compared.

Upper (/lower) Baselines are defined as a curve consisting of the first black pixels from the top (/bottom) in each column of the image. In some papers they are also referred to as parts of the "enclosing envelope". Horizontal (/vertical) projections are defined by the number of black pixels in each column (/row) of the image. Baselines and horizontal projections can be thought of as functions of x while vertical projection is a function of y. Thereby we defined 4 different functions, which can be later compared to analyze their similarity.

To improve the extraction speed and precision, instead of the original image a thinned, vectorized representation of the signature is considered, as described in [8]. An example of such a baseline can be seen in Figure1.

Figure 1
Baseline of a signature

3.1.2 Skew

The very first step of acquiring the skew of a signature is to define what skew stands for. Using the knowledge gained in a consultation with handwriting experts a definition was created which presumably would be helpful at the comparison: the skew information consist of a set of straight lines, where each line represents an imaginary foundation of a component, which can be regarded as an autonomous element of the signature. This definition allows us to assign skew information to the gaps between signature elements, and according to [9] those spaces are just as peculiar as any other feature of a signature.

Our first approach was a naive algorithm, where the goal was to obtain the lower contour of the signature, which was done by starting vertical scan lines from the bottom left corner of the image and store the lowest pixel which was part of the signature. To determine whether a pixel is representing paper or ink a function was created, which not only used the pixel itself but its environment as well to give the best result. This was necessary because of the different kind of images with different amount of noise on them.

After obtaining the lower contour a line to each separable segment was fitted using linear regression. Separable segments are parts divided by a horizontal gap. The resulting lines were often convincing enough, but of course this algorithm has its drawbacks: the most important problem was that it frequently had trouble recognizing the separate parts of a signature, in fact it could only distinguish two segments when a significant horizontal gap existed between them, but unfortunately this was not true for most of the signatures in our database. Another remarkable disadvantage was that it used information only from the image itself, while by the time there was additional information available [10] that could definitely increase the reliability of the algorithm, like stroke positions.

Using the experience gained so far, we came up with a new approach, whose fundamental element became components. Components are parts of the signature that could and should be treated as an independent part, having their own skew information. Typically a component is a part of the name (first name, last name) or an accent. Experiments showed that accents should have their own skew information, as they are a distinctive feature of signatures. The resulting lines seemed to almost perfectly represent our definition of skew (Fig.2.); hence it was time to examine whether they could be used to make a distinction between forged and genuine signatures.

Figure 2

Three skew lines obtained by our algorithm

It is important to note, that in some cases more separable segments were found than expected, but since those extra segments were found on almost all genuine signatures of the given signer, they should be considered as a feature of the signature and not as an error. Representing the skew information with numeric values (angle, length, position) and examining these values for both forged and genuine signatures has shown that our skew lines were adequate features to verify signatures, however alone they are not sufficient to unquestionably separate valid and forged ones.

3.1.3 Loops

Loops in our definition are connected regions in the image which are fully enclosed by "signature" pixels.

Shape descriptors are used to describe the different aspects of loops, thereby allowing an easy comparison. There are several promising formulas described in the literature for calculating shape descriptor values. Instead of choosing one of them, we used as many of them as possible. This will allow us to identify the most significant shape factors in later phases. Our hypothesis was that there will be at most 2-3 significant shape descriptors, and all the others will be redundant and can be ignored in the future.

The following shape descriptors were used during feature extraction: Perimeter, area, formfactor, maximum diameter, maximum diameter angle, roundness, centroid, bounding box, inscribed diameter, extent, modification ratio, compactness, bounding circle, moment axis angle, convexity, solidity, aspect ratio

The nontrivial descriptors are defined as follows:

$$\text{Modification Ratio} = \frac{\text{Inscribed Diameter}}{\text{Maximum Diameter}} \tag{1}$$

$$\text{Formfactor} = \frac{4\pi \cdot \text{Area}}{\text{Perimeter}^2} \tag{2}$$

$$\text{Extent} = \frac{\text{Area}}{\text{Bounding Rectangle Area}} \tag{3}$$

$$\text{Compactness} = \frac{\sqrt{\left(\frac{4}{\pi}\right)\text{Area}}}{\text{Maximum Diameter}} \tag{4}$$

$$\text{Convexity} = \frac{\text{Convex Perimeter}}{\text{Perimeter}} \tag{5}$$

$$\text{Solidity} = \frac{\text{Area}}{\text{Convex Area}} \tag{6}$$

$$\text{Aspect Ratio} = \frac{\text{Maximum Diameter}}{\text{Minimum Diameter}} \tag{7}$$

A detailed introduction of shape descriptors can be found in [11].

3.2 Classification

3.2.1 Transformation before classification

Input values are not always simple values. The baseline of a signature for example is a set of points, which could easily form a large, hard to interpret input for the classifier. To overcome this, the feature values are not directly used as an input. Instead, each signature is compared with every other signature, delivering a wide range of comparison results which are then stored and used as input for classification. Each feature can define its own distance function. For example when calculating the distance of loop centroids, this distance function is a Euclidean distance function, but when calculating the distance of two baselines this function is a DTW (Dynamic Time Warping) function. Input variables can, and should be interpreted on different scale. To allow a general processing, all input values are rescaled to fit in the [0…1] interval.

A neural network has a well defined number of input variables. Some signatures of a signer may have 3 loops, while all the others contain 4 loops. Without further preprocessing, this could result in input vectors of different lengths for the classifier. Hence before the classification we should fill these missing values with infinite, or other appropriate value.

3.2.2 The structure of the classifier

When our data are ready for the classification, the structure of the classifier should be chosen. First the count of the neural networks has to be specified.

There are distance data available comparing each feature of a signature with all the corresponding features in all the other signatures. In this point this data are used in two different approaches: averaging the distance values origin from the other signatures and use one neural network, or use as many neural networks as the count is the training signatures and let each network concentrate on the distances of one signature. In the second case all network present an own result between 0 and 1. Values between 0.4 and 0.6 are ignored because these are ambiguous results. To generate the final result is the average of the remaining values..

Choosing the internal structure of the neural network is also an important decision. Several kinds of networks were tried and there is not an unequivocal proposition. We can decide to use or not a preprocessing layer. That means a group of neurons

each associated with few input data. For example if there are 50 inputs, the first neuron associated with inputs form 1st to 5th the second with inputs from 2nd to 6th, etc. Our tests showed that preprocessing layer is useful to enhance the efficiency of the classifier. After the preprocessing layer, middle layers have to be defined. They are groups of neurons each of them associated with all neurons from the previous layer. It seems preprocessing layers give stability to the network, thereby reducing the number of required training iterations and moderating the changes in the answer of the network during the training. Finally in the last layer there is one single neuron providing the answer.

3.2.3 Classifier training

This is the hardest point of the classification. Signature verification has a special restriction: there are no forgeries at the training phase. This can result in a network learning the "constant true" function. To overcome this, artificial forgeries should be created by adding some small values to the inputs coming from the original signatures. Defining this "small" value is very important. If this value is too big, the artificial forgeries are too far from the original signatures and the network will learn the "constant true" function. On the other hand if this value is too small then the network will learn that a signature is original if and only if it was given on training. To filter out this scenario the network can not get all the original signatures at the training phase.

We use the backpropagation method on training which gives one other parameter called teaching speed. Our neurons activation function is a transformed sigmoid function.

Best results were achieved by a network with a preprocessing layer where each neuron has 13 inputs and two hidden layers (with 2 and 8 neurons). Further results are presented in the next section.

4 Experimental results

In our experiments the database of the Signature Verification Competition 2004 [12] was used. This is an on-line signature database therefore it contains the stroke information, but no images are provided. The stroke information was used to synthesize signatures similar to the original ones. Stroke points were connected with straight lines, fading out on the line borders. Bicubic interpolation and anti-aliasing were used to make the final image smoother. An example of reproduced signature can be seen on Fig. 1. 1600 signatures from 40 signers (20 originals and 20 forgeries from each) ensure a sample large enough for testing our feature extraction and classification algorithms.

The experimental setup uses 10 original signatures from each signer and a set of generated forgeries for training. Afterwards the network is tested with 10 other original and 10 forged signatures. The resulting average error rates are summarized in table 1. It can be seen, that the values vary largely between different signers. It is however really promising, that this error rate is under 10% at two of the signers.

It can also be seen, that the use of multiple networks does not necessarily bring better results, however in some cases (like in the case of 008.csv) these results can be significantly better.

SVC signature identifier	**EER (single network)**	**EER (multiple networks)**
002.csv	56%	53%
006.csv	53%	60%
008.csv	50%	**17%**
010.csv	40%	50%
013.csv	33%	56%
015.csv	33%	30%
018.csv	20%	30%
020.csv	36%	40%
025.csv	46%	36%
028.csv	46%	53%
032.csv	33%	43%
033.csv	36%	36%
035.csv	**3%**	**10%**
037.csv	36%	30%
038.csv	36%	60%
040.csv	**10%**	**7%**
Average	**35%**	**38%**

Table 1

Conclusions

In this paper we discussed problems occurring during feature based off-line signature verification and delivered solutions for the special questions of this problem class. Although our achieved error rates are not yet ready for real world scenarios, we have demonstrated that local features can successfully be used with neural network classification systems, to distinguish original signatures from forgeries.

References

[1] V. E. Ramesh and M. N. Murty, "Off-line signature verification using genetically optimized weighted features," Pattern Recognition, no. 32, pp. 217-233,

[2] J. Coetzer, B. M. Herbst, and J. A. d. Preez, "Off-line Signature Verification Using the Discrete Radon Transform and Hidden Markov Model," EURASIP Journal on Applied Signal Processing, vol. 4, pp. 559-571, 2004.

[3] M. A. Ismail and S. Gad, "Off-line Arabic Signature Recognition and Verification," Pattern Recognition, vol. 33, pp. 1727-1740, 2000.

[4] H. Baltzakisa and N. Papamarkos, "A new signature verification technique based on a two-stage neural network classifier," Engineering Applications of Artificial Intelligence, vol. 14, pp. 95-103, 2001.

[5] E. Ozgunduz, T. Senturk, and M. Karsligil, "Off-line Signature Verification and Recognition by Support Vector Machine," Thirteenth European Signal Processing Conference, 2005.

[6] M. K. Kalera, S. Srihari, and A. Xu, "Offline Signature Verification and Identification Using Distance Statistics," International Journal of Pattern Recognition and Artifcial Intelligence, vol. 18, no. 7

[7] A. A. Kholmatov, "Biometric Identity Verification Using On-Line & Off-Line Signature Verification," 2003.

[8] B. Kovari, "Time-Efficient Stroke Extraction Method for Handwritten Signatures," in ACS07, The 7th WSEAS International Conference on Applied Computer Science, 2007, pp. 157-161.

[9] R. A. Huber and A. M. Headrick, "Handwriting Identification: Facts and Fundamentals," CRC Press, LCC, 1999.

[10] B. Kovari, G. Kiss, and H. Charaf, "Stroke Extraction and Stroke Sequence Estimation for Off-line Signature Verification," The Eighth IASTED International Conference on Visualization, Imaging, and Image Processing.

[11] J. C. Rush, The Image Processing Handbook, Fifth edition. North Carolina State University, 2006.

[12] First International Signature Verification Competition, 2004 [Online]. Available: http://www.cs.ust.hk/svc2004/