IWSSIP 2010 - 17th International Conference on Systems, Signals and Image Processing

Character Recognition based on Trellis Diagrams

Martin Grafmüller
Vision and Fusion Laboratory
Karlsruhe Institute of Technology (KIT)
76131 Karlsruhe, Germany
grafmueller@kit.edu

Jürgen Beyerer, Kristian Kroschel
Fraunhofer Institute of Optronics, System Technologies and Image Exploitation (IOSB)
76131 Karlsruhe, Germany
{juergen.beyerer, kristian.kroschel}@iosb.fraunhofer.de

Abstract—Many papers on pattern recognition have been published in the last decades but character recognition is still part of current research. However, one main topic are classifiers that can be easily augmented by new training data or even new classes. Furthermore, the classifiers have to have a certain robustness with respect to noise, i.e., the recognition rate must not be significantly affected by the presence of noise in the character images. For this reason a new classifier approach is introduced, which is based on trellis diagrams and thus similar to a Viterbi decoder known from communication systems. The training as well as the classification procedure are discussed in detail. Additionally, to show the competitiveness the performance is compared with already existing classifiers on a character dataset with and without noise.

***Keywords*—character recognition, trellis diagram, Viterbi algorithm**

I. Introduction

In recent decades many classifiers have been introduced in pattern and character recognition. Basically, there are four common methods used for character recognition. Those are statistical methods, artificial neural networks, support vector machines, and ensemble methods where multiple weak learners are combined to one powerful classifier. All of them have been investigated in detail and the pros and cons are well known. A detailed overview of the advantages of the classifiers mentioned and the still remaining problems can be found in [1].

Even if many different classification methods are already known we are interested in a classifier that has the following properties. The classifier must be easily trainable. That means augmentation of the classifier by a new class or new training data should be easily possible without retraining with the old and the new data. Additionally, it is required that the classifier is robust to noise and little distortions contained in the character images. Furthermore, a fast running implementation of the classifier is very important as well. In this paper the main focus is on isolated printed characters since the classification task is to classify numbers printed on different materials in different sizes and fonts. Hence, a classifier with a good generalization ability with respect to different fonts and font sizes is needed. Furthermore, the different materials the characters are printed on cause errors, which can be interpreted as noise in the character images. These are just a few reasons why we want to introduce a new classifier that can cope all these requirements.

The idea of the classifier is based on signal detection used in many communication systems, where the received signal is determined by evaluating a trellis diagram. In this approach the models are built of trellis diagrams, which is the main purpose why the Viterbi algorithm is used for the evaluation of the character models. The evaluation consists of determining the "shortest" path through the trellis diagram of every model. The classification decision is finally taken for the class with the lowest cost. Moreover, as the Viterbi algorithm is well known from communication systems many developments have already been published, which can be simply applied to the classifier introduced here. For example, there are many parallel implementations of the Viterbi algorithm that make classification faster. Mostly, the Viterbi algorithm is currently used for word recognition. The words are modeled by hidden Markov models, which are evaluated with the Viterbi algorithm to obtain the most likely character sequence [2], [3], [4]. In some approaches the shape of the characters are modeled by one or two-dimensional hidden Markov Models [5], [6] in contrast to our classifier where the image itself is modeled. Hence, the classifier is not constraint to character recognition but can be used for any classification task. The classifier is based on one model per class, where the models are similar to Markov models but the states change with the pixel position of an image. So the number of states changes for every pixel at a certain position. Furthermore, weights are introduced that weigh the transitions of the states from one pixel to the next pixel. In the sense of a Markov model the weights can be interpreted as the transition probabilities. Both, the states and the transition weights are determined in training and mainly depend on the training data.

This paper is organized as follows. In Section II the training procedure of the trellis based classifier is discussed in detail. Section III introduces the functionality of the classifier whereas the results of different experiments are presented in Section IV. Finally, a conclusion is drawn and some remarks are given in Section V.

II. Training of the Classifier

The classifier we introduce consists of N_c models, where N_c denotes the number of classes. Hence, in the training procedure for every class one model has to be built and for classification all of them have to be evaluated. For convenience the superscript indicating the class is neglected for the derivation of the classifier. It is introduced in Section III-B where we discuss the entire classifier. Before the training procedure of the trellis based classifier is described some assumptions are made.

Let character image $\mathbf{G} \in \mathbb{B}^{M \times N}$ with M columns, N rows, and gray values in $\mathbb{B} = \{0, 1, \ldots, 255\} \subset \mathbb{N}_0$ be also denoted as column vector $\mathbf{g} \in \mathbb{B}^K$ of dimension $K = M \cdot N$. Now the image can be interpreted as signal sequence similar to a received discrete signal transmitted over a communication channel.

The model consists of vertexes $s_j(k)$, which are called states in the following and edges assigned with weights $w_{ij}(k)$, where both are determined during the training procedure. Furthermore, the weights represent the cost of the transition from state $s_i(k-1)$ to state $s_j(k)$, where k denotes the pixel position of the image in vector $\mathbf{g}$. The model of a certain class can be written as matrix

$$\mathbf{S} := \begin{bmatrix} \mathbf{g}_1 & \mathbf{g}_2 & \ldots & \mathbf{g}_P \end{bmatrix}^{\mathrm{T}} ,$$

which contains the P present training samples of one class line-by-line. Moreover, the rows of matrix $\mathbf{S}$ can be interpreted as the states $s_j(k)$, where the columns denote the position k. It is possible that matrix S contains the same gray values multiple times in a column. Since those states have to be considered only once they are merged to one state with the corresponding gray value, i.e., every possible state is contained only once in a column. Thus, the obtained valid states are put at the beginning of each column, where the remaining elements of the columns are filled with -1. This is only done to keep the matrix structure, but these parts of $\mathbf{S}$ are not evaluated in classification.

A part of a model for one class is illustrated in Figure 1, where the states are vertically and the pixel positions are horizontally displayed. Furthermore, the

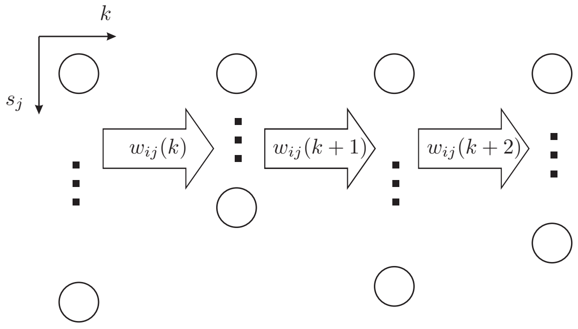


Figure 1. Basic idea of a part of a model, which represents one class.

weights of the transitions are indicated by arrows. The weights $w_{ij}(k)$ are determined according to

$$w_{ij}(k) = \begin{cases} 1 \, , & \text{for} \quad |e_{ij}(k)| > 0 \\ \infty \, , & \text{else} \end{cases} ,$$

where $|e_{ij}(k)|$ denotes the number of transitions from state $s_i(k-1)$ to $s_j(k)$ for all training samples of one class. Hence, transitions that have not occurred in the training data are weighted by infinity, i.e., such transitions are not evaluated in classification.

As the procedure previously described is just for a model of one class it has to be performed N_c times for the discrimination of N_c classes.

One big advantage of this training method is that the trained models can be easily augmented by new training data. Only the nonexisting states and the transition weights have to be added. In the case that a state already exists only the corresponding transitions weights have to be adapted. This means a retraining with the entire training set is not necessary. Moreover, a new class can be easily added as well. Just a new model for this class has to be added to the already existing classifier model.

III. Classification

As for each class one model is trained it is necessary to evaluate each model for classification, i.e., the path of minimal cost—shortest path—has to be determined for each model. However, the cost of the paths can be recursively determined, which allows due to the *principle of optimality* [7] to apply dynamic programming. This is an efficient way of computing the cost of the shortest path. As the evaluation of the trellis diagrams is similar to signal detection in communication systems we use the Viterbi algorithm. This meets a maximum-likelihood detection with respect to the image vector $\mathbf{g}$.

A. Viterbi Algorithm

The Viterbi algorithm is a forward dynamic programming algorithm introduced by A. Viterbi [8] in 1967. It allows the sequential determination of the shortest path through every trellis diagram to be evaluated.

Furthermore, it reduces the computational effort of the evaluation of one trellis diagram from a product

$$O\left(\prod_{k=1}^{K-1} B(k-1)B(k)\right)$$

to a sum

$$O\left(\sum_{k=1}^{K-1} B(k-1)B(k)\right) ,$$

where K is the length of the image vector $\mathbf{g}$ and $B(k)$ indicates the number of states in the trellis diagram at position k. The algorithm starts at image position $g_k,\ k=0$ to determine the shortest path through the trellis diagram. This is done by computing the squared difference between g_k and all given states $s_j(k)$ of the trellis diagram at position k. This can formally be expressed as follows

$$\lambda_j(k) \ = \ (g_k - s_j(k))^2 \ , \qquad k=0,\dots,K-1 \ ,$$

where g_k is the kth element of test vector $\mathbf{g}$ at position k. The next step is the computation of the shortest distance leading to state $s_j(k)$ at position k. For $k=0$ it is just the squared difference

$$\Lambda_j(0) = \lambda_j(0) \ .$$

But for $k>0$ it is the minimum of the sum of the shortest distance in state s_i at position $k-1$ and the weighted squared difference at position k as given by

$$\Lambda_j(k) = \min_i \{\Lambda_i(k-1) + w_{ij}(k)\lambda_j(k)\} \ , \quad k=1,\dots,K-1 \ .$$

The recursion ends when the last pixel $g_k,\ k=K-1$ of the image is reached. Now, the shortest path can be determined by computing the minimum over all possible states at position $K-1$ according to

$$\Lambda_{min}(K-1) = \min_j \Lambda_j(K-1) \ . \qquad (1)$$

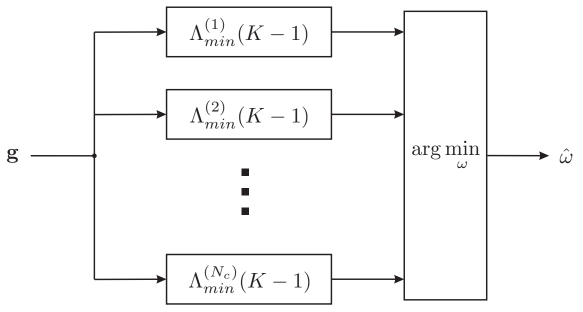
At this point the evaluation of one trellis diagram is finished, i.e., this has to be done N_c times if N_c classes have to be discriminated. The entire classification task of all N_c classes is discussed in the next section.

Further information concerning dynamic programming can be found in [7]. For more detailed information about the Viterbi algorithm and maximum-likelihood detection see [8], [9], [10], [11].

B. The Classifier

In the previous section we described the Viterbi algorithm which is used to determine the shortest path through the trellis diagram. Since the classification task consists of N_c classes this algorithm has to be performed N_c times. Hence, the computational effort of the entire classifier is

$$O\left(N_c \sum_{k=1}^{K-1} B(k-1)B(k)\right) \ .$$



g
$\Lambda_{min}^{(1)}(K-1)$
$\Lambda_{min}^{(2)}(K-1)$
$\Lambda_{min}^{(N_c)}(K-1)$
$\arg\min_\omega$
$\hat{\omega}$

Figure 2. Block diagram of the entire classifier that discriminates between N_c classes.

After the shortest paths of all class models have been determined according to Equation (1) the winning class $\hat{\omega}$ is given by the minimum over all N_c shortest paths

$$\hat{\omega} = \arg\min_\omega \Lambda_{min}^{(\omega)}(K-1) \ ,$$

where superscript ω indicates the class. The superscript has been neglected for convenience of the derivation of the training and classification procedure. The block diagram of the entire classifier for N_c classes is given in Figure 2.

IV. Experiments

This section shows how the character dataset was created and the experiments with the trellis based classifier are performed on this dataset. Furthermore, the character dataset was affected by additive noise of different levels to show the performance of the classifier, which is compared with k-nearest neighbors (NN) classifiers, linear discriminant analysis (LDA), linear discriminant analysis using a naive Bayes approach (LDA-NB).

A. Character Image Dataset

The character database was created by printing numbers from 0 to 9 in twelve different fonts, three different sizes, and different styles to make the classifier more robust to changes in the character images. The printed pages were afterwards captured with an industrial camera and the single characters were separated and scaled to $[24 \times 24]$ pixel images to get a consistent size of all characters. The character dataset was randomly divided into a training set with 4556 samples, and one test set with 2000 samples, respectively. Thus, the test set is not a subset of the training data. In both datasets all classes are almost equally distributed.

B. Classification Results

In this section the classification results are shown and compared to the results of k-NN classifiers, and both LDA approaches. The trellis based classifier is trained as explained in Section II with the training dataset of 4556 samples. Due to the 10 classes in the training dataset the training results in 10 models, where every model consists of approximately 85 states per pixel. The states

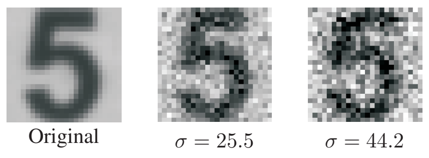


Figure 3. Image of number 5—original and affected by additive zero mean white Gaussian noise with different standard deviations.

Table I
ERROR RATES (%) OF THE TRELLIS BASED CLASSIFIER (TBC) COMPARED TO LDA, LDA-NB, AND k-NN CLASSIFIERS.

Classifier	$\sigma = 0$	$\sigma = 25.5$	$\sigma = 44.2$
TBC	0.05	0.15	0.25
LDA	0.05	34.6	60.1
LDA-NB	1.4	1.5	1.55
1-NN	0.1	0.1	0.1
5-NN	0.3	0.3	0.25
10-NN	0.25	0.2	0.3

of one model are connected by approximately $2 \cdot 10^5$ weights on average.

The classification performance is determined on the test set with 2000 samples on which the trellis based classifier only one character classifies wrongly. Moreover, the same task is performed with a 1-NN classifier, 5-NN classifier, and 10-NN classifier. The error rates are 0.1%, 0.3%, and 0.25%, respectively. Furthermore, the test set is classified by the LDA with an error rate of 0.05%, and the LDA-NB with an error rate of 1.4%. Since we want to show how powerful the trellis based classifier on noisy data is, white Gaussian noise is added to the test dataset. First, zero mean white Gaussian noise with a standard deviation $\sigma = 25.5$ is added to the character images of the test set. The trellis based classifier shows an error rate of 0.15%, which is close to the best result of the 1-NN with an error rate of 0.1%. The 5-NN, and 10-NN are slightly worse with the error rates 0.3%, and 0.2%, respectively. The LDA result with an error rate over 30% is useless but the result of the LDA-NB remains with 1.5% almost constant compared to the result with the original test dataset. For the second experiment the standard deviation is increased to $\sigma = 44.2$. The experiment on this dataset shows that the error rate of the trellis based classifier increases just a little to 0.25% though the error rates of the NN classifiers, and the LDA-NB are remaining almost constant. The LDA again shows the worst result with an error rate of 60.1%. All results of the classifiers previously described can be found summarized in Table I. Furthermore, one example image of one number with and without noise is illustrated in Figure 3.

C. Discussion

It can be concluded that the training of the trellis based classifier is very simple. This makes a classifier augmentation with new training data or further classes quite easy, i.e., no retraining with the entire training data is necessary. One main drawback is that a lot of states and transition weights have to be stored and processed in classification. However, classification speed is comparable to NN classifiers or even slightly faster. Furthermore, it has been shown that the results of the trellis based classifier are close or even better than NN classifiers. Only on very noisy images the classification rate slightly decreases as expected.

V. CONCLUSION

A new kind of classifier for character recognition that is based on trellis diagrams has been introduced. We have discussed the training and classification procedure in detail. The classifier basically works like the Viterbi detector known from communication systems, where it is used for maximum-likelihood detection of received signals. The performance of the classifier has been demonstrated on noiseless and noisy character images; it is competitive to standard classifiers. Part of future work will be the adaptive determination of the transition weights in the trellis diagrams, since we assume that the classification gets more robust. To speed up classification, a further investigation will be the reduction of states in the trellis diagrams without losing significant classification performance.

REFERENCES

[1] C.-L. Liu and H. Fujisawa, “Classification and learning methods for character recognition: advances and remaining problems,” in *Machine Learning in Document Analysis and Recognition*, ser. Studies in Computational Intelligence, vol. 90/2008, 2008, pp. 131–161.

[2] Y. Kessentini, T. Paquet, and A. B. Hamadou, “Off-line handwritten word recognition using multi-stream hidden Markov models,” *Pattern Recognition Letters*, vol. 31, no. 1, pp. 60–70, 2010.

[3] F. Einsele, R. Ingold, and J. Hennebert, “A language-independent, open-vocabulary system based on HMMs for recognition of ultra low resolution words,” in *SAC ’08: Proceedings of the 2008 ACM symposium on Applied computing*. New York, NY, USA: ACM, 2008, pp. 429–433.

[4] Y.-H. Tseng and H.-J. Lee, “Recognition-based handwritten Chinese character segmentation using a probabilistic Viterbi algorithm,” *Pattern Recognition Letters*, vol. 20, no. 8, pp. 791–806, 1999.

[5] C.-C. Hsieh and H.-J. Lee, “A probabilistic stroke-based Viterbi algorithm for handwritten Chinese characters recognition,” in *Proc. th IAPR International Conference on Pattern Recognition Vol.II. Conference B: Pattern Recognition Methodology and Systems*, 1992, pp. 191–194.

[6] O. E. Agazzi and S. shiaw Kuo, “Hidden markov model based optical character recognition in the presence of deterministic transformations,” *Pattern Recognition*, vol. 26, no. 12, pp. 1813–1826, 1993.

[7] D. P. Bertsekas, *Dynamic Programming and Optimal Control*, 3rd ed. Belmont, Mass.: Athena Scientific, 2005, vol. 1.

[8] A. J. Viterbi, “Error bounds for convolutional codes and an asymptotically optimum decoding algorithm,” *IEEE Transactions on Information Theory*, vol. 13, no. 2, pp. 260–269, 1967.

[9] G. D. Forney, “Maximum-likelihood sequence estimation of digital sequences in the presence of intersymbol interference,” *IEEE Trans. Inf. Theory*, vol. 18, no. 3, pp. 363–378, 1972.

[10] ——, “The Viterbi algorithm,” *Proceedings of the IEEE*, vol. 61, no. 3, pp. 268–278, 1973.

[11] T. K. Moon and W. C. Stirling, *Mathematical Methods and Algorithms for Signal Processing*. Upper Saddle River, NJ: Prentice Hall, 2000.