Decision Tree Classifier for Character Recognition Combining Support Vector Machines and Artificial Neural Networks

Martin Grafmüller\textsuperscript{a}, Jürgen Beyerer\textsuperscript{b}, and Kristian Kroschel\textsuperscript{b}

\textsuperscript{a}Vision and Fusion Laboratory, Karlsruhe Institute of Technology (KIT), 76131 Karlsruhe, Germany
\textsuperscript{b}Fraunhofer Institute of Optronics, System Technologies and Image Exploitation (IOSB), 76131 Karlsruhe, Germany

ABSTRACT

Since the performance of a character recognition system is mainly determined by the classifier, we introduce one that is especially tailored to our application. Working with 100 different classes, the most important properties of a reliable classifier are a high generalization capability, robustness to noise and classification speed. For this reason, we designed a classifier that is a combination of two types of classifiers, in which the advantages of both are united. The fundamental structure is given by a decision tree that has in its nodes either a support vector machine or an artificial neural network. The performance of this classifier is experimentally proven and the results are compared with both individual classifier types.

Keywords: character recognition, decision tree, neural network, support vector machine

1. INTRODUCTION

In recent decades many approaches on optical character recognition have been proposed but one question still remains: Which is the “right” classifier? A classifier should have a very good generalization—high recognition performance on data that is not in the training dataset—capability, it should be easily trainable, and if possible robust to noise or other distortions in the character images. An overview of the advances and the still remaining problems of common classifiers are concluded in [1]. A detailed discussion of document analysis systems where the images are captured with digital cameras can be found in [2]. Character recognition found its way not only into industrial applications but also into document analysis. A detailed summary of the last forty years of character and document recognition is given in [3]. Basically, one can say that there is no classifier that can handle all demands of different classification tasks. This can also be concluded from the no free lunch theorem,\textsuperscript{4} which states that there is no classifier that is superior over all classifiers, i.e., there is no classifier that outperforms all other classifiers with respect to all classification tasks. Hence, we are currently working on a classifier for optical character recognition (OCR) that is running on an industrial camera. The camera is employed to several industrial applications with changing environmental conditions. The main focus is on reading serial numbers, product numbers, or expiration dates on different types of packaging. Depending on the application it is necessary to recognize numerals, capital letters, lower case letters or other additional characters, where all of them can vary in size and font. Moreover, constraints are also given by the industrial camera where processing speed and memory is limited.

In document analysis of images captured with handheld cameras the recognition system has to cope with motion blurred and poorly printed or damaged characters. To handle this kind of degradations the images can be restored. Another possibility is to train the classifier with blurred images. This is done in [5,6], where a point spread function and information about the motion blurring are estimated. With this knowledge a synthetic training dataset is created, which is used to train the classifier and make it robust to this kind of degradations. Effectiveness was proven in several experiments. The main advantage of this procedure is the relatively small number of training data. However, for our application we have got a lot of training samples containing several kinds of typefaces in different sizes.

For degraded character recognition many classifiers have been introduced that are a combination of several classifiers. One of them is introduced in [7]. The authors combine a Hopfield neural network with autoassociator networks for the recognition of isolated degraded characters. Since the system works with binarized character
images they have to cope with even more degradations, which occur depending on the binarization method. Compared to our system they consider only 13 classes, which keeps the overall system quite simple. However, they show that the system outperforms the individual classifiers. Since cascading several classifiers shows considerable promise—not only in classification performance but also in classification speed—for character recognition we propose a classifier that is based on Support Vector Machines (SVMs) in combination with Artificial Neural Networks (ANNs). The fundamental structure is given by a decision tree in which the first three nodes contain a SVM for a preselection of the different classes. For classification in the four leaf nodes we apply ANNs. They eventually make multiple decisions in the particular node. The structure of the entire classifier has the advantage that, depending on the application some nodes can be skipped. This speeds up classification and makes the classifier more reliable, given a priori is known what can be skipped.

The paper is structured as follows. First, we deal with some fundamentals of the applied classifiers in Section 2. In Section 3 we introduce the procedure of the proposed classifier. Classification results and a comparison with other classifiers are given in Section 4. In Section 5 we draw a conclusion and give some remarks for future work.

2. FUNDAMENTALS OF THE CLASSIFIERS

This section provides a brief introduction of the classifiers which are combined in the decision tree. These are support vector machines and artificial neural networks.

2.1 Support Vector Machines

A support vector machine is a powerful classification and regression tool, introduced by V. Vapnik in 1995. In recent years SVMs have been deployed in several practical applications like robotics, image processing, data fitting, and prediction of the stock market. Since we are interested in character recognition we only introduce SVM classification briefly.

The SVM is a binary classifier that separates data given in input space by the vectors \( x_i \in \mathbb{R}^N \), \( i = 1, \ldots, M \) and the corresponding labels \( y_i \in \{-1, +1\} \), where \( M \) denotes the number of data samples and \( N \) the dimension of input space. The separation is done by a hyperplane that is induced by the so-called support vectors.

The optimization task for the determination of the support vectors is maximizing the margin between the hyperplane and the two classes. Formally the problem can be stated as

\[
\min_{w, b, \xi} \frac{1}{2} \|w\|^2 + C \sum_{i=1}^{M} \xi_i ,
\]

subject to:

\[
y_i (\langle w, x_i \rangle + b) + \xi_i \geq 1 , \quad i = 1, \ldots, M ,
\]

\[
\xi_i \geq 0 ,
\]

where \( w \in \mathbb{R}^N \) is a weight vector and \( b \in \mathbb{R} \) denotes a bias of the separating hyperplane with respect to the origin. The \( \xi_i \) indicates slack variables, which allow a relaxation of the constraints to accept noise and outliers. The sum of the slack variables is multiplied by the parameter \( C \), which is the so-called regularization parameter. It regularizes the complexity of the decision rule dependent on the error rate.

Equation (1) is a quadratic programming problem which can be rewritten in the dual Lagrangian

\[
\max_{\alpha} \sum_{i=1}^{M} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{M} y_i y_j \alpha_i \alpha_j \langle x_i, x_j \rangle ,
\]

subject to:

\[
\sum_{i=1}^{M} y_i \alpha_i = 0 ,
\]

\[
C \geq \alpha_i \geq 0 , \quad i = 1, \ldots, M ,
\]

where \( \alpha_i \) denotes Lagrangian multipliers. With the solution of this Lagrangian the data is separated by a linear hyperplane in the input space. To achieve a non-linear hyperplane in input space a mapping function \( \phi \) is applied.
to the vector \( x \). The function \( \phi(x) \) maps the vector \( x \) from input space to a high dimensional feature space, where the data is separated by a linear hyperplane. This yields to the following Lagrangian

\[
\max_\alpha \sum_{i=1}^{M} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{M} y_i y_j \alpha_i \alpha_j \langle \phi(x_i), \phi(x_j) \rangle ,
\]

subject to:

\[
\sum_{i=1}^{M} y_i \alpha_i = 0 ,
\]

\[
C \geq \alpha_i \geq 0 , \quad i = 1, \ldots, M .
\]

Since the mapping function \( \phi(x) \) only appears in a scalar product the so-called kernel trick can be applied. Therefore, the scalar product can be substituted by a kernel function \( K(x, z) \), which is a valid kernel function if it fulfills Mercer’s theorem.\(^9\)

The decision function for classification is finally given by

\[
\hat{f}(x) = \text{sign} \left\{ \sum_{i \in SV} \alpha_i y_i K(x, x_i) + b \right\} ,
\]

where sign indicates the signum-function and \( SV \) denotes the index set of the support vectors. For the kernel function we use the radial basis function that is given by

\[
K(x, z) = \exp \left\{ -\gamma \|x - z\|^2 \right\} ,
\]

where the exponent contains the Euclidean distance of both vectors. Additionally, there is another tuning parameter \( \gamma \) that affects the width of the radial basis function.

Since the quadratic programming problem is convex, the located local minimum is always a global minimum. This means, the solution is always an optimal one. Furthermore, the model complexity is regulated by the maximization of the margin, and since the margin size is not dependent on the dimensionality of the training data, the performance of the classifier is not dependent on the dimension of the input space. In other words, even with a very high dimensional input space SVMs are very good in generalization.

### 2.2 Neural Network

An artificial neural network basically tries to simulate the brain by connecting a large number of neurons together. The connections between the neurons represent the synapses. According to [4] a NN with at least one hidden layer (HL) is able to approximate any decision boundary, not depending on dimension or complexity, with any precision. This is possible if a sufficient number of training samples is available. The birth of the ANN dates back to 1943 when McCulloch and Pitts\(^10\) mathematically formulated the behavior of nets of neurons. However, since in the 1980s the backpropagation algorithm became public and computation speed was getting appropriate to the learning methods, ANNs have been widely used in different classification tasks.

Similar to the SVMs, the data is given by the \( N \)-dimensional vectors \( x_i \in \mathbb{R}^N , i = 1, \ldots, M \), where \( M \) is the number of data samples. A feed forward neural network consisting of one hidden layer with \( \nu \) neurons and one output layer with \( \mu \) neurons is given by

\[
g_j(x) = \sigma \left( \tilde{w}_j^T \sigma \left( \begin{array}{c} w_1^T x + b_1 \\ \vdots \\ w_\nu^T x + b_\nu \end{array} \right) + \tilde{b}_j \right) , \quad j = 1, \ldots, \mu ,
\]

where the vectors \( \tilde{w} \) and \( w \) denote weights and the scalars \( \tilde{b} \) and \( b \) the corresponding biases. These parameters are determined in the training procedure which is stated below. The function \( \sigma(x) \) is called the activation function. It often has a sigmoidal shape and can be expressed as

\[
\sigma(x) = \frac{2}{1 + e^{-2x}} - 1 .
\]
The decision is finally performed with respect to the output values according to
\[
\hat{f}(x) = \arg\max_j g_j(x), \quad j = 1, \ldots, \mu.
\]

More information about ANNs can be found in [4,11].

The weight vectors \(\tilde{\mathbf{w}}\) and \(\mathbf{w}\) and the corresponding biases \(\tilde{b}\) and \(b\) are determined by the resilient propagation algorithm. According to [12] this algorithm converges faster than other common learning algorithms and it is more robust to the choice of the initial parameters. However, there is no guarantee that the training algorithm finds the global optimum.

The advantage of ANNs is that, once the parameters are determined in training, evaluation speed is very fast. However, there are several drawbacks, e.g., training can take a long time and it may tend to overfitting. This means that classification accuracy on the training samples is very high, but the generalization capability is very low. To avoid overfitting and to guarantee a good generalization capability as well the use of a validation dataset is recommended. During the training procedure the accuracy of the ANN is checked with the validation data and if necessary the procedure is stopped before overfitting occurs.

### 3. PROPOSED PROCEDURE

The decision tree classifier for character recognition consists of the classifiers introduced in the previous section. The structure of the decision tree is illustrated in Figure 1, where SVM\(_i\) indicates a support vector machine and ANN\(_i\) an artificial neural network.

SVM\(_1\) is the classifier in the root node, which divides the input data into two groups. The first group consists of lower and upper case characters, whereas the second group consists of numbers, mathematical symbols, and other symbols. On the second level of the tree there are two SVMs. SVM\(_2\) classifies the input data into lower case letters and capital letters. Classifier SVM\(_3\) splits the input data into two further groups. One of them consists of numbers and mathematical symbols, the other one of all other symbols. The parameters of the three SVMs are determined by grid search with a training and a validation dataset considering the corresponding classes. Finally, the parameters are chosen with respect to the lowest validation error. A more detailed description of the datasets can be found in the next section.

The classifiers of the leaf nodes are ANNs with one hidden layer as described in Section 2.2. There is still the open question of how to chose the number of hidden layer neurons, since there is no general rule. However, there are some constraints that limit the number of hidden layer neurons. First, the number is dependent on the required classification performance and classification speed. Second, it must be considered that the number of training samples is sufficient with respect to the number of free parameters—weights and biases—of the ANN. Hence, we performed several experiments where the number of hidden layer neurons was varied between 10 and 70. The number of neurons in the output layer corresponds to the number of classes the ANN has to classify.
ANN\(_1\) classifies the capital letters thus the output layer has 26 neurons. Basically, ANN\(_2\) is the same network as ANN\(_1\) except that it classifies the lower case letters. ANN\(_3\) is a net with 20 neurons in the output layer. It classifies numbers and mathematical symbols. The last ANN on the most right in Figure 1 is ANN\(_4\), which distinguishes between 28 other symbols. Since ANNs tend to overfitting the training of the neural networks was stopped after the validation error became worse ten times in sequence.

Though the division of the hundred classes this way may not result in the best classification performance it has one main advantage for our applications. The structure allows that certain levels of the tree can be skipped, if it is known \textit{a priori} what kind of characters are to be classified in the application. For example, if the classification task is to classify nothing else but numbers, then only ANN\(_3\) has to be evaluated.

4. CLASSIFICATION RESULTS

To test the proposed classifier we created one dataset with 100 classes containing almost all characters placed on the German keyboard layout. The characters were printed in different fonts, sizes, and styles. Afterwards the characters were captured with an industrial camera and the single characters were separated. To get a unified size of all characters they were scaled to \([24 \times 24]\) gray scale images. The entire dataset was randomly divided into a training set containing approximately 45000 characters, a validation, and a test dataset with 8775 samples each. Note, all sets are disjoint.

To reduce the computational complexity we extracted feature vectors from the single character images. We have made good experiences with the two-dimensional Discrete Cosine Transform (DCT) regarding the reduction of the dimension of the feature space with a still very good classification performance. Hence, we also used this transform for feature extraction to evaluate the proposed classifier and to compare it with other classifiers. The DCT was introduced by Ahmed\textsuperscript{13} et al. in 1974. They showed that it performs closely to the \textit{Karhunen-Loève} Transform, which is known to be the optimal linear transform in terms of least square error. In contrast to the \textit{Karhunen-Loève} Transform the DCT achieves a similar result without any knowledge about the covariance matrix of the data. Similar to the number of hidden layer neurons, there is no rule of how many coefficients are necessary to represent the data appropriately. Mainly, the number is constraint by the required classification performance and speed, respectively. For this reason, the dimension of the features is varied according to \(d = m^2 - 1\), \(m = 4, \ldots, 11\), where \(m\) indicates a square—\([m \times m]\)—in the upper left corner of the DCT domain. Furthermore, the first coefficient is discarded, since this represents the mean of the image and thus strongly depends on illumination changes.

The seven classifiers arranged in the tree are trained independently with respect to partial tasks and are connected to the tree in order to solve the classification task. First, we discuss the results of the three SVMs, which can be found summarized in the plots in Figure 2. As we can see in Figure 2(a), SVM\(_1\) reaches almost the best performance with only 35 DCT coefficients, i.e., the test error only slightly decreases considering more DCT coefficients. If we look to Figure 2(b) with the result of SVM\(_2\), we can see that the test error is about five times higher, compared to SVM\(_1\). This can be explained, since some of the upper and lower case characters look similar. Moreover, the plot shows that using 24 or more DCT coefficients leads only to a slight further decrease of the test error. SVM\(_3\) shows the best performance on its particular classification task. Using 48 DCT coefficients the test error is below 0.05\%, which corresponds to only two misclassifications. According to Figure 2 it can be concluded, that with at least 24 DCT coefficients the performance of the three SVMs is already very good, whereas the test error significantly increases if only 15 coefficients are used.

The discussion of the results of the ANNs is done based on the plots given in Figure 3. The four plots show the error rates of the ANNs—with 50 hidden layer (HL) neurons—depending on the number of DCT coefficients. Note, since the initial training weights and biases of the ANNs are chosen randomly the training result varies from training to training. Hence, the error rates given in these plots are average values over five error rates trained with different initial weights and biases. As shown in Figure 3(a) the test error almost remains constant if at least 24 DCT coefficients are used. ANN\(_2\) shows a similar result, which is given in Figure 3(b). However, the best performance is given if between 24 and 63 DCT coefficients are used. In the case of ANN\(_3\) the test error slightly decreases if more than 63 DCT coefficients are considered. The best result of this net is obtained with 35 DCT coefficients. ANN\(_4\) also shows the best performance with 35 DCT coefficients. If more than 35 DCT
Figure 2. Training result of the three SVMs depending on the number of DCT coefficients.

Figure 3. Training result of the four ANNs depending on the number of DCT coefficients.

coefficients are considered the test error is increasing until 99 or more DCT coefficients are used. If we compare all four plots, we can conclude that with at least 24 DCT coefficients a very good classification performance is already obtained.

Since we not only performed experiments with ANNs with 50 neurons in the hidden layer, the test errors of the decision tree classifier are summarized in Table 1. As in the plots in Figure 3 the results in this table are average values over five error rates as well. The error rates in bold show the best result with respect to the number of DCT coefficients. The best classification performance is achieved if a feature vector with 99 DCT coefficients and ANNs with 50 neurons in the hidden layer are used.
Table 1. Test error rates (%) of the proposed decision tree classifier (DTC) with respect to the feature dimension and the number of hidden layer neurons.

<table>
<thead>
<tr>
<th>DCT coefficients</th>
<th>hidden layer neurons</th>
<th>70</th>
<th>60</th>
<th>50</th>
<th>40</th>
<th>30</th>
<th>20</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>120</td>
<td>1.26</td>
<td>1.29</td>
<td>1.30</td>
<td>1.31</td>
<td>1.32</td>
<td>1.40</td>
<td>2.03</td>
<td></td>
</tr>
<tr>
<td>99</td>
<td>1.29</td>
<td>1.24</td>
<td>1.23</td>
<td>1.30</td>
<td>1.29</td>
<td>1.38</td>
<td>1.93</td>
<td></td>
</tr>
<tr>
<td>80</td>
<td>1.33</td>
<td>1.36</td>
<td>1.39</td>
<td>1.39</td>
<td>1.39</td>
<td>1.44</td>
<td>1.91</td>
<td></td>
</tr>
<tr>
<td>63</td>
<td>1.37</td>
<td>1.41</td>
<td>1.39</td>
<td>1.39</td>
<td>1.45</td>
<td>1.45</td>
<td>1.96</td>
<td></td>
</tr>
<tr>
<td>48</td>
<td>1.28</td>
<td>1.33</td>
<td>1.35</td>
<td>1.33</td>
<td>1.36</td>
<td>1.41</td>
<td>1.85</td>
<td></td>
</tr>
<tr>
<td>35</td>
<td>1.43</td>
<td>1.42</td>
<td>1.42</td>
<td>1.46</td>
<td>1.50</td>
<td>1.53</td>
<td>2.26</td>
<td></td>
</tr>
<tr>
<td>24</td>
<td>1.61</td>
<td>1.59</td>
<td>1.62</td>
<td>1.64</td>
<td>1.64</td>
<td>1.81</td>
<td>2.78</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>2.65</td>
<td>2.72</td>
<td>2.83</td>
<td>2.77</td>
<td>2.93</td>
<td>3.39</td>
<td>6.01</td>
<td></td>
</tr>
</tbody>
</table>

Table 2. Best error rates (%) of the proposed decision tree classifier (DTC), the one-against-one SVM, and the ANN with 200 HL neurons for different feature dimensions.

<table>
<thead>
<tr>
<th>DCT coefficients</th>
<th>DTC Trn.</th>
<th>DTC Val.</th>
<th>DTC Test</th>
<th>SVM Trn.</th>
<th>SVM Val.</th>
<th>SVM Test</th>
<th>ANN Trn.</th>
<th>ANN Val.</th>
<th>ANN Test</th>
</tr>
</thead>
<tbody>
<tr>
<td>120</td>
<td>0.17</td>
<td>0.98</td>
<td>1.21</td>
<td>0.016</td>
<td>0.90</td>
<td>1.12</td>
<td>0.25</td>
<td>1.55</td>
<td>1.69</td>
</tr>
<tr>
<td>99</td>
<td>0.13</td>
<td>0.95</td>
<td>1.17</td>
<td>0.064</td>
<td>0.87</td>
<td>1.12</td>
<td>0.30</td>
<td>1.59</td>
<td>1.72</td>
</tr>
<tr>
<td>80</td>
<td>0.18</td>
<td>0.98</td>
<td>1.32</td>
<td>0.055</td>
<td>0.89</td>
<td>1.27</td>
<td>0.46</td>
<td>1.69</td>
<td>1.79</td>
</tr>
<tr>
<td>63</td>
<td>0.20</td>
<td>1.07</td>
<td>1.32</td>
<td>0.11</td>
<td>0.99</td>
<td>1.21</td>
<td>0.55</td>
<td>1.81</td>
<td>1.95</td>
</tr>
<tr>
<td>48</td>
<td>0.18</td>
<td>1.06</td>
<td>1.28</td>
<td>0.11</td>
<td>1.0</td>
<td>1.27</td>
<td>0.60</td>
<td>1.86</td>
<td>1.95</td>
</tr>
<tr>
<td>35</td>
<td>0.22</td>
<td>1.21</td>
<td>1.40</td>
<td>0.10</td>
<td>1.15</td>
<td>1.39</td>
<td>0.96</td>
<td>2.15</td>
<td>2.09</td>
</tr>
<tr>
<td>24</td>
<td>0.37</td>
<td>1.42</td>
<td>1.56</td>
<td>0.18</td>
<td>1.33</td>
<td>1.45</td>
<td>1.34</td>
<td>2.53</td>
<td>2.54</td>
</tr>
<tr>
<td>15</td>
<td>0.88</td>
<td>2.74</td>
<td>2.66</td>
<td>0.18</td>
<td>2.03</td>
<td>2.20</td>
<td>2.96</td>
<td>4.12</td>
<td>3.90</td>
</tr>
</tbody>
</table>

In Table 2, the best decision tree classifier we obtained is compared to an one-against-one SVM and an ANN with 200 neurons in its hidden layer. The error rates in bold indicate the best test error achieved with each classifier. If we look to the test error of the decision tree classifier, we can see that the results are competitive to the results of the SVM. Especially, the results are similar for 24 and 35 DCT coefficients. However, the error rates of the ANN are for all features significantly higher. This shows that the performance of the decision tree classifier is almost the same as the performance of the SVM, but the classification is about four times faster. This is due to the combination of the SVMs and the ANNs in a decision tree structure, which exploits the advantage of both classifiers.

4.1 Discussion

At first glance one might think that the training of the different classifiers in the nodes of the decision tree is more complex and time consuming. But it turned out that the decomposition of the complete classification task especially speeds up the training time of the ANNs. The results show that the classification performance is close to the performance of the one-against-one SVM. On the other hand side, classification speed of the decision tree classifier is about four times faster compared to the one-against-one SVM. Additionally, the tree structure allows an activation of only those classifiers required for the actual classification task.

Up to now the number of hidden layer neurons of the leaf node ANNs has been chosen the same way, i.e., the number was varied from 10 to 70 but it has been always the same for all ANNs. One assumption is that with an individual selection of the number of hidden layer neurons depending on the ANN the classification performance of the decision tree classifier can be further improved. Basically, the same thing applies for the number of DCT coefficients that can be varied from classifier—SVMs and ANNs—to classifier as well. Both are topics for further investigations, since these have not been considered yet. Another thing that has not been considered so far are rejections of samples that do not belong to one of the trained classes. This would make the classifier more reliable in cases when the samples to be classified are corrupted due to wrong segmentation or poor print.
5. CONCLUSION

In this article a decision tree classifier in combination with support vector machines and artificial neural networks has been introduced. The fundamentals of the single classifiers and the combination of them have been discussed. Several advantages have been pointed out, e.g., due to prior knowledge of the classification task the irrelevant levels of the tree can be skipped, which finally makes classification faster. Splitting the entire classification task into several smaller tasks leads to a faster training of the ANNs. In fact, the classification performance of the decision tree classifier is slightly worse compared to the results of the one-against-one SVM, but classification speed is four times faster.

REFERENCES