Improving the Classification Accuracy of the Scanning n-tuple Method.

George Tambouratzis
Institute for Language and Speech Processing, Artemidos & Epidavrou Str., Paradissos Amaroussiou, 151 25, Athens, Greece.
giorg_t@ilsp.gr

Abstract.

In this article, the application of the scanning n-tuple technique to classification tasks is studied. The performance of this technique is examined in a hand-written character recognition task where the accuracy is initially low. This task is employed as a case study for designing a general-purpose algorithm that improves the scanning n-tuple performance in hard classification tasks, by focusing on the characteristics of the pattern space. Experimental results indicate that the use of the algorithm results in a substantial improvement of the scanning n-tuple classification performance in comparison to previous results. This improvement is shown to be equivalent to that achieved by employing structural knowledge regarding the specific pattern space.

1. Introduction.

Currently, considerable research effort is aimed at recognising hand-written characters with a high accuracy. Due to the large variability of the data, several of the methods proposed are based on neural networks. Among neural network models, the n-tuple method stands out due to its readily implementable structure [1] and its short recognition time. The standard n-tuple is a statistical pattern recognition method, which decomposes a given pattern into u sets of n points (termed n-tuples). Recently, research activity has focused on the n-tuple method, both regarding theoretical issues ([2], [3]) as well as real-world applications. In [4], an extension to the n-tuple method termed the scanning n-tuple (hereafter also denoted as sn-tuple) is proposed. The sn-tuple relies on a small number of n-tuples, which repetitively scan the input pattern.

This article focuses on the sn-tuple approach, in an effort to devise methods that further improve its classification accuracy. To that end, the pattern space defined by the training set is studied in detail, to investigate how significant overlaps between pattern classes affect the classification performance and in particular whether one or more classes may be automatically divided into sub-classes to improve the classification performance. The results of this approach are found to be of an equivalent quality to these obtained when utilising knowledge concerning the specific pattern space to manually split the classes into sub-classes that optimise the classification performance.

2. The scanning n-tuple method.

The standard n-tuple method [1] decomposes a given input pattern $\underline{p}$ into u sets of n points (termed n-tuples). In the given context of character recognition, these points are binary pixels. For each n-tuple, a physical entity termed a function is provided, which consists of 2^n memory locations, one for every possible combination of pixel values. In that memory location, either (i) a binary number is stored (indicating whether the corresponding combination has occurred during training, this being termed the non-weighted n-tuple) or (ii) an integer is stored (representing the frequency-of-occurrence of the combination, this being termed the weighted n-tuple).

The j^{th} n-tuple scans a fixed set of points $\{e_{1,j}, e_{2,j}, ..., e_{n,j}\}$. In the corresponding function, these determine the memory location with address a_j:

$$a_j = \sum_{k=1}^{n} p(e_{k,j}) * \sigma^k \qquad (1)$$

where $p(x)$ denotes the x^{th} element of pattern $\underline{p}$ and σ is the number of possible values of a point (for binary pixels, $\sigma = 2$). During training, as input patterns are presented, in each n-tuple the contents of the memory locations designated by (1) are updated, to record the patterns' occurrence. During classification, the contents of the memory locations a_j are added to generate the final classification result. Thus, the recognition task is transformed into u elementary recognition tasks, each performed on the basis of the corresponding n points. The response r_l of a given node l is:

$$r_l = \sum_{j=1}^{u} content(a_j) \qquad (2)$$

0-7695-0750-6/00 $10.00 © 2000 IEEE

In the sn-tuple, a mask $m = \{m_1, m_2, m_3, .., m_n\}$ is defined for the n sampled points, which are situated at a regular interval f (termed offset) from each other [4]. This mask is applied repetitively with different initial points, in order to cover the entire pattern. Thus, each sn-tuple scans the whole retina, while in the standard n-tuple method each n-tuple only samples a specific area of the retina. Prior to applying the sn-tuple technique, the input pattern is chain-coded, starting at a given point of the character and traversing its perimeter until the whole character is covered. This results in a one-dimensional vector of k elements $\underline{c} = \{c(1), c(2), .., c(x), .., c(k)\}$, where $1 < x < k$. The resulting chain-code has a variable length k, preventing the application of the standard n-tuple method. The scanning n-tuple is used instead, each sn-tuple instance being defined as:

$$\underline{e} = \{c(e_{1,j}), c(e_{2,j}), .., c(e_{n,j})\}, \text{ where for } \forall i,\ 1 < i \le n : c(e_{i,j}) = c(e_{i-1,j} + f) \quad (3)$$

The address determined by each mask application is:

$$a_{j,l} = \sum_{k=1}^{n} c(e_{k,j}) * \sigma^k \quad (4)$$

The value of σ is equal to the number of possible directions used during chain-coding. Then, the response of the l^{th} sn-tuple node consisting of u sn-tuples is:

$$r_l = \sum_{j=1}^{u} \{ \sum_{k=1}^{f_{max}} content(a_{j,l}) \} \quad (5)$$

where f_{max} is the maximum offset value possible for the current chain-code $\underline{c}$, so as to cover all chain-code elements. For a given position of the mask on the chain-code, σ^n possible combinations of the sn-tuple exist. Consequently, for each sn-tuple σ^n memory locations are provided to store the corresponding frequencies-of-occurrence. During classification, for all combinations occurring in the chain-coded pattern, the stored frequencies are added to generate the node response.

3. The character classification task.

In [4], an sn-tuple network using one node per class is studied for both the ESSEX and CEDAR datasets of hand-written digits, giving competitive recognition rates of 91.4% and 97.3% respectively. Here, the ESSEX dataset, which is difficult to classify accurately, is studied in an effort to improve the sn-tuple classification performance. The ESSEX dataset consists of 3917 training and 1835 test patterns, each being a 42x50 matrix of binary points. To reduce the character variation, the dataset is pre-processed by centring and scaling each character so that it fully occupies the retina and applying a median-filter operator to suppress the existing noise. The normalised pattern is then chain-coded by scanning the character from the upmost-left corner and recording the transitions along the character edges for each point as one of eight possible directions.

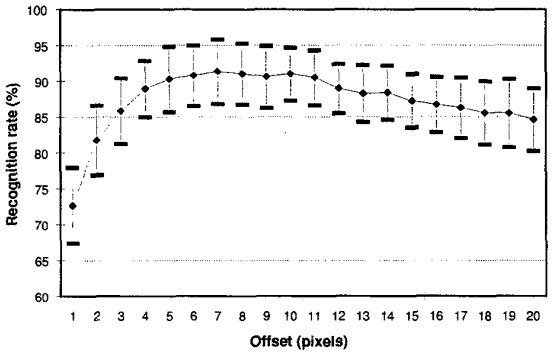

Figure 1 – Average test set accuracy over the 10 digit classes and 95% confidence interval using single sn-tuple nodes sampling 5 points.

When the configuration proposed in [4] is simulated (one node per class, each with 4 sn-tuples with offsets of 2, 3, 4 and 5 points), a recognition rate of 89.3% is obtained. The deviation from the recognition rate quoted in [4] is attributable to different pre-processing steps. To improve this result, the performance of networks whose nodes consist of only a single sn-tuple is studied as a function of the offset. The results displayed in figure 1 indicate that the recognition performance rises for offsets higher than these used in [4] and for offsets between 5 and 11 gives better results than the four sn-tuple system. As summarised in table 1, the optimal recognition rate is equal to 91.4% for single sn-tuple nodes and 91.6% for nodes consisting of four sn-tuples. Notably, the single sn-tuple system gives a recognition rate equivalent to the optimal value of [4] where four sn-tuple nodes were used.

Even for the highest-performing network of four-sn-tuple nodes, digit class "1" has a low recognition accuracy (only 79.8%). This is due to the existence of two fundamentally different sub-classes for digit "1", without and with a horizontal line at the bottom of the character (denoted as sub-classes "1A" and "1B" respectively). Within the training set for class "1", only 6% of the patterns belong to sub-class "1B". Consequently, the node corresponding to class "1" is unable to extract the characteristics of "1B". Instead, most patterns from sub-class "1B" are classified as members of digit class "2",

due to the horizontal line at the bottom of the digit which characterises both classes "1B" and "2".

To improve the performance of a deformable model-based character recogniser, Cheung et al. [5] introduce multiple sub-classes for certain digit classes. This approach is adopted here, with two sub-classes established for class "1". The partition of class "1" is achieved by a rule examining whether the maximum character width in the lower half of the retina is equal to or exceeds 50% of the retina width.

Network configuration	recognition rate	recognition variance
4 x sn-tuples *(f = 6, 7, 8 & 10)*	91.63 %	6.04
1 x sn-tuple *(f = 7)*	91.38 %	6.27
4 x sn-tuples with sub-classes *(f = 6, 7, 8 & 10)*	92.55 %	4.19
1 x sn-tuple with sub-classes *(f = 7)*	92.23 %	4.50

Table 1 - Classification results for different network configurations and offset values.

The experimental results obtained with these modifications are shown in the last two rows of table 1. When each node consists of four sn-tuples, the recognition rate is equal to 92.6%, improving the best results previously reported [4] by 1.2% (giving a reduction in the error rate of approximately 14%). Even the single sn-tuple system gives an optimal recognition rate of 92.2%, reducing the error rate of [4] by 10%.

4. Improving the sn-tuple accuracy.

The improvement obtained by defining two sub-classes for class "1" is based on studying the testing set and determining the pattern classes that are not accurately classified. It is desirable to design a general-purpose method for improving the classification accuracy for different datasets in an automated manner, rather than manually examining each class. Such an algorithm is of particular use when the pattern space is complex, with a large number of pattern archetypes (as is the case in this task). The algorithm should be based exclusively on the training set, without resorting to the testing set. To that aim, four approaches are investigated. The first two involve studying the characteristics of each pattern class in an effort to determine cases where possible problems might occur. The remaining two utilise the response of the trained network to the training patterns so as to determine the relative position of classes in the pattern space.

Method 1.

This method employs a self-organising scheme to separate each class into constituent sub-classes. This scheme is based on the provision of several discriminator nodes for every pattern class, each of these nodes representing one sub-class. As a new pattern is presented to the network, it is compared to the knowledge accumulated already in each discriminator node (which forms a top-down expectation of the pattern), using the distribution constraint [6]. This constraint has been shown to allow a self-organising network consisting of standard n-tuple nodes to cluster pattern classes in an autonomous manner [6]. However, when using a network with only a few (in our experiments 4) sn-tuples per node, this method is found not to provide a significant improvement, without extensive experimentation with the dataset so as to determine an appropriate distribution constraint value.

Method 2:

As in the previous method, each class is examined in isolation, in an effort to determine patterns that differ considerably to the main part of the class. Due to the chain-coding operation, the pattern space consisting of the frequencies of occurrence of each direction in the chain-code is used. Initially, the class average is determined for all directions. As a measure of the pattern difference, the distance from the class average is used. Thus, this method determines the direction of maximal variability and then splits the pattern class along this direction. This results in an improvement in the clustering performance, the best classification rate being equal to 91.98% when focusing in class "1". However, the method is found to be sensitive to the exact displacement from the class centroid where the boundary between the current class and the new sub-class is set.

Method 3:

This method involves a type of reinforcement learning to improve the network response. Reinforcement learning is required for training patterns which are misclassified (i.e. are classified to class j though they belong to class i) or (b) are correctly classified, but with a very low decision margin between the top two class responses. Cases (a) and (b) differ but training for both types of effects can be expected to improve the network response. Evidently, the training for patterns of type (a) needs to be more intensive than for patterns of type (b).

Experiments indicate that this method improves the network performance, giving a classification rate of up to 91.89%. However, the results depend strongly on determining the optimal amount by which the network is trained for each pattern. Also, this method is susceptible to overtraining since by adapting the network excessively to 'problematic' patterns its response to other patterns is adversely affected, the collective recognition rate suffering accordingly.

Method 4:

The fourth method is also based on the confidence with which the network classifies the training patterns, the network size being varied by introducing new sub-classes in order to improve the classification. More specifically, initially all the training patterns are classified by the system, recording the confidence of each classification. Out of these, N patterns are studied (in our experiments, N=100), which correspond to the negative-confidence classifications (i.e. cases where patterns are misclassified) and the lowest-confidence classifications (where patterns are correctly classified, but with a low confidence).

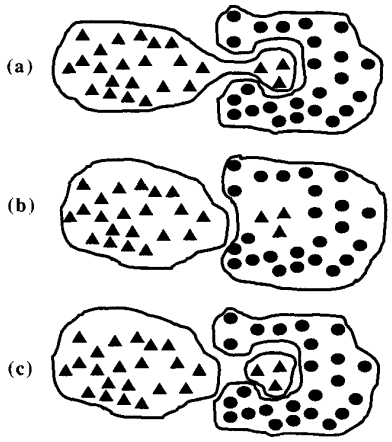

Figure 2 – Representation of two overlapping classes (a), together with the clustering before (b) and after (c) applying method 4.

The pattern class C_k to be divided into two sub-classes is determined to be the one with the highest fraction of negative-confidence and low-confidence decisions. The pattern class C_l, which causes the negative- and low-confidence classifications for C_k, is also determined. Then, a new network node is introduced, defining an additional sub-class for C_k. The nucleus of that sub-class consists of the patterns belonging to C_k, which have a low or negative classification confidence due to C_l. Following the introduction of the new node, the training set is repetitively classified by the new network which consists of $N+1$ nodes. During these iterations, the two constituent sub-classes of C_k are refined, by studying the M lowest-confidence patterns as assigned to the $N+1$ classes. This procedure is repeated until the transfer of patterns between the sub-classes of C_k either ceases or is stabilised, so that an equal number of patterns are exchanged between the two sub-classes in subsequent iterations. This procedure is summarised in figure 2.

For the Essex dataset, the class with most misclassifications is class "1", due to class "2". Therefore this is initially split into two new classes. According to the aforementioned procedure, seven patterns form the nucleus of the new sub-class, resulting in an improved recognition rate of 92.04%. The network settles after nine refinement iterations, when the new sub-class consists of 29 patterns and the network recognition rate reaches 92.52%. This is very close to the recognition rate obtained using the geometry rule to split class "1" (92.55%). These results are further improved by dividing class "1" to remove the confusion with class "7", giving after only three iterations a recognition rate exceeding 92.6%. This represents the best classification accuracy reported so far for the ESSEX dataset, indicating the effectiveness of the automated class-splitting method.

5. Conclusions.

In this paper, a number of methods to improve the scanning n-tuple performance have been investigated. This study has focused on the design of an algorithm that allows the automated splitting into sub-classes of pattern classes with a high variability. It has been found that the best results are obtained when making use of the classification confidence of the training set to create new clusters for low-confidence training patterns. The experimental results illustrate that the best method gives a recognition performance equivalent to that obtained using structural knowledge regarding the different classes.

6. Acknowledgement.

The author wishes to thank Dr. A. Amiri for his assistance in replicating the original experiments of [4].

7. References.

[1]. Aleksander, I. (1983) Emergent Intelligent Properties of Progressively Structured Pattern Recognition Nets, *Pattern Recognition Letters*, **1**, pp. 375-384.

[2]. Rohwer, R. & Morciniec, M. (1998) The Theoretical and Experimental Status of the n-tuple Classifier. *Neural Networks*, **11** (1) pp. 1-14.

[3]. Jorgensen, T. M. & Linneberg, C. (1999) Theoretical Analysis and Improved Decision Criteria for the n-tuple Classifier. *IEEE Trans. On Pattern Analysis & Machine Intelligence*, **21** (4), pp. 336-347.

[4]. Lucas, S. & Amiri, A. (1996) Statistical Syntactic Methods for High-performance OCR, *IEE Proc.-Vis. Image Signal Process.*, **143** (1) pp. 23-31.

[5]. Cheung, K.-W., Yeung, D.-Y. & Chin, R.T. (1998) A Bayesian Framework for Deformable Pattern Recognition with Application to Hand-written Character Recognition, *IEEE Trans. on Pattern Analysis and Machine Intelligence*, **20** (12) pp. 1382-1388.

[6]. Tambouratzis, G. & Tambouratzis, D. (1994) Self-Organisation in Complex Pattern Spaces Using a Logic Neural Network. *Network: Computation in Neural Systems*, **5**, pp. 599-617.