We are IntechOpen, the world's leading publisher of Open Access books Built by scientists, for scientists

4,800
Open access books available

122,000
International authors and editors

135M
Downloads

Our authors are among the

154
Countries delivered to

TOP 1%
most cited scientists

12.2%
Contributors from top 500 universities

CLARIVATE ANALYTICS BOOK CITATION INDEX INDEXED

WEB OF SCIENCE™

Selection of our books indexed in the Book Citation Index in Web of Science™ Core Collection (BKCI)

Interested in publishing with us?
Contact book.department@intechopen.com

Numbers displayed above are based on latest data collected.
For more information visit www.intechopen.com

3

A Greedy Approach for Building Classification Cascades

Sherif Abdelazeem†
shazeem@aucegypt.edu
Electronics Engineering Dept., The American University in Cairo

Classification cascade is a well-known technique to reduce classification complexity (recognition time) while attaining high classification accuracy. Cascades are usually built using ad-hoc procedures. The mission of this chapter is to introduce a principled way of building cascades using a greedy approach. Given a large pool of classifiers, the proposed approach sequentially builds a near-optimal cascade. Given a set of N classifiers and one powerful classifier with satisfactory classification accuracy, the proposed algorithm automatically generates classification cascades with complexity $O(N^2)$ which means it is fast and scalable to large values of N. Experiments show that the proposed algorithm is efficient and builds classification cascades that substantially reduce the overall system complexity while preserving the accuracy.

1. Introduction

Suppose we have a classification task on which we have already found a complex classification technique that achieves a satisfactory accuracy. Suppose also while such classification technique is very powerful, its time complexity is unacceptably high. This scenario happens frequently in real life as many powerful but very time-consuming techniques have been devised in recent years (e.g. SVM and multi-classifier systems). Our goal would be to build a system that preserves the accuracy of that complex classifier while having much better timing performance.

The high complexity of powerful classifiers might give the impression that high accuracy could not be achieved without sacrificing recognition time. In fact, this is not true. The high average recognition time of a classifier in most cases is due to improper resource allocation. Weak classification techniques while not achieving satisfactory accuracy, they do a good job. They are capable of correctly classifying considerable number of cases. Actually, most of patterns in many problems are 'regular' patterns; that is, they could be classified using a simple classifications technique. So why should we use a very complicated time-consuming classification technique for just few extreme cases? This observation led to the development of cascade systems [Kanyank & Alpaydin, 1997] which is the main concern of this chapter. In such a system, all the patterns to be classified first go through a first stage; those patterns that are classified with confidence score higher than a certain threshold leave the system with the labels given to them by the first stage. The patterns that are classified with

confidence scores lower than the threshold are rejected to the second stage. In the same manner, the patterns pass through different stages until they reach the powerful last stage that does not reject any patterns. Figure 1 illustrates this idea.

The idea of classification cascades has been well-known for long time but has not attracted much attention in spite of its practical importance [kuncheva , 2004]. Recently, and since the prominent work of Viola and Jones [Viola & Jones, 2001], the idea of cascade has been attracting considerable attention in the context of object detection which is a rare-event classification problem. To avoid any confusion, we will call the cascades used in the context of object detection "detection cascades" while we will call the cascades used in regular classification problems "classification cascades" in which we are interested in this chapter.

There are many works in the literature that try to build a cascade using some heuristics and ad-hoc procedures [Kanyank & Alpaydin, 1997, Pudil et al. 1992, Giusti et al., 1999, Gorgevik & Cakmakov, 2004, Ferri et al., 2004]. Some automatic approaches of building classification cascades consider the problem as an optimization problem and might be attacked using various optimization techniques (e.g. particle swarm [Oliveira et al., 2005], and simulated annealing [Chellapilla et al., 2006a]). The most elegant automatic approach was that of Chellapilla et al. [Chellapilla et al., 2006b] using Depth-First search. However, the algorithm is of complexity $O(Q^N)$, where Q is a variable suggested to be 32 by [Chellapilla et al., 2006b]. This means that the algorithm is very slow and not scalable to large values of N.

In this chapter, we present an algorithm to automatically generate classification cascades given a set of N classifiers and a powerful classifier with satisfactory accuracy. The algorithm is of complexity $O(N^2)$ which means it is fast and scalable to large values of N.

The remaining of this chapter is organized as follows. Section 2 formulates and states our problem. In section 3, we describe our proposed algorithm. Section 4 presents an experimental validation of our proposed algorithm. And in section 5 we conclude.

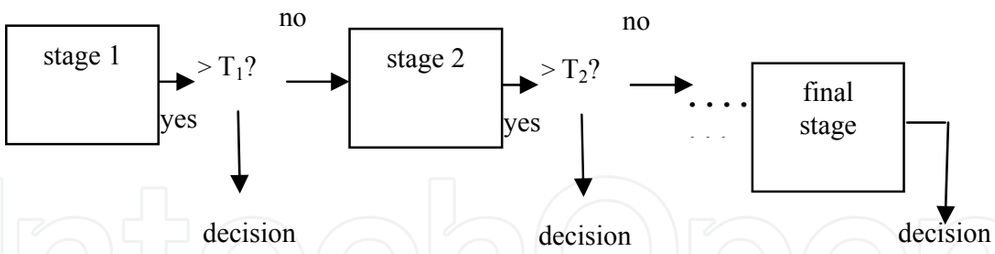


Fig. 1. Typical classification cascade system.

2. Problem statement

We first assume that we have a pool of classifiers of different complexities and accuracies to be candidate members of the cascade. Note that we cannot use all the classifiers of the pool in the cascade. This is because not all cascades are efficient. You may find a cascade that is more complex than the powerful classifier; using a cascade like this is worthless. Also, one subset of classifiers could be better than other subsets. Furthermore, certain ordering of the classifiers in a cascade could be more efficient than other orderings. Hence, we are in need of an algorithm that selects the most efficient *ordered* subset of classifiers.

Our problem could then be formulated as follows. Given a classifier that is powerful and complex and given a pool of classifiers of different complexities and accuracies, we need to select the optimal (or near optimal) ordered subset that if put in a cascade structure gives an accuracy not less than that of the optimal case with the lowest complexity possible.
Now we are going to present some notations that will help presenting our algorithms formally. We denote an unordered set by boldface character surrounded by curly braces, and its elements by the same character but in italics and subscripted by numbers (e.g. $A_2, A_1, A_5, \ldots \in \{\mathbf{A}\}$). An ordered set (or an array) is denoted by just a boldface character, and is elements by the same character but in italics and subscripted by numbers (e.g. $\mathbf{A} = [A_3\ A_1 \ldots]$). Note that the subscripts of an ordered or unordered set are arbitrary and hold no ordering significance. We enumerate the elements of an unordered set {**A**} as follows $\{\mathbf{A}\} = \{A_3, A_1, \ldots\}$ and the elements of the ordered set **A** as follows $\mathbf{A} = [A_4\ A_2 \ldots]$. **C**=[**A B**] means that the ordered set **C** is a concatenation of the two ordered sets **A** and **B**. {**B**}={**A**}-A_i that the unordered set **B** contains all the elements of the unordered set **B** except the element A_i; and if $A_i \notin \mathbf{A}$, then {**B**}={**A**}. In this chapter we will represent a cascade by an ordered set whose elements are the classification stages ordered in the set from left to right. The function *complexity*(**A**) returns the complexity of the cascade represented by array **A** which is estimated using a validation set.

3. A greedy algorithm

We start with partitioning the dataset available for training and testing the system into 3 parts: training set, validation set, and test set. The training set is used for training the available classifier in the pool. The validation set is used by the algorithm that is going to be described in this section to build the cascade. The test set is used to test the performance of the overall system.
We first assume that the powerful classifier we use has accuracy more than any classifier in the pool of classifiers we generated. Then to build a cascade that has accuracy not less than that of the powerful classifier, we should put the powerful classifier as the final stage; hence, we will denote it S_F. The algorithm we propose then has now two jobs: To select an ordered subset of classifiers from the pool to serve as stages of the cascade before S_F and to set the thresholds of the stages. Our algorithm, hence, has two steps presented below.

3.1 Step 1: Set the Thresholds

In our problem formulation, we stated that we do not want to get accuracy less than that of S_F. In a classification cascade, the only source of additional errors to that of S_F is that of the classifiers in the cascade other than S_F. To avoid such source of error, we might adjust the thresholds of all the candidate classifiers in the pool to have almost zero errors. This makes any cascade built using the pool have also zero additional errors to that of S_F. While this procedure will lead to no additional errors to be committed by the cascade, it would make us use too tough thresholds. Tough thresholds make the classifiers reject most of the patterns to the next more complex stages, hence, to lose good opportunities of complexity reduction. Adjusting the errors to give zero errors for different cascade stages is actually unnecessarily too tough a procedure. We actually can let different classifiers commit some

errors without losing any overall accuracy. This is because there are some very elusive patterns that get falsely classified by weak classifiers with high confidence scores. If we put in mind not to commit any errors by any classifier in the pool, such illusive patterns will lead us to choose very high thresholds. Fortunately, most of these patterns could be ignored (not accounted as being errors) as they are most likely falsely classified by S_F too; and, hence, will not lead to any additional errors.

Our procedure for finding thresholds of pool classifiers will then be as follows. Using every classifier in the pool, classify the patterns of the validations set and order them according to the confidence scores they are given. Traverse these ordered patterns form the one that has been classified by the highest confidence score to the lowest. While traversing the patterns, monitor whether the patterns are correctly or falsely classified. Once you hit a pattern that is falsely classified, check whether this same pattern is falsely classified by S_F or not. If yes, then this pattern would not contribute to errors of the overall system and can be safely ignored, and we can continue traversing the patterns. We stop when we hit a pattern that is falsely classified by the classifier under consideration but correctly classified by S_F. Then set the threshold of the classifier under consideration to be the confidence score of the pattern we stopped at. We do the same for all the classifiers in the pool to form the corresponding set of thresholds. This procedure is illustrated in Figure 2.

3.2 Step 2: Select from the Pool

Now we have a pool of classifiers (whose thresholds are set) and the final stage S_F. The most direct and optimal way to build the cascade is to try all the possible combinations of the classifiers in the pool (different number of stages with different order of stages), then selecting the one of the lowest complexity. Note that the complexity of the cascade is calculated using the validation set. While this procedure guarantees selecting the optimal cascade, it has a complexity of $O(\sum_{i=1}^{N} \frac{N!}{(N-i)!})$, where N is the number of classifiers in the pool. This makes the procedure not scalable for large values of N. Here we suggest a sequential greedy approach that selects a near-to-optimal cascade that has a complexity of $O(N^2)$.

We start with the powerful classifier S_F. This is the best *single-stage* system that achieves our constraint of having accuracy not less than S_F. Now we search for the best two-stage classifier. This is done simply by trying all the nodes in the pool as a first stage (the last stage will be S_F), and then selecting the one that results in least complex cascade. Now we pick this classifier out of the pool and put it as the first stage in the cascade.

After finding the best two-stage system, we search for the best three-stage system. We assume that the best three-stage system will have the same first stage as in the best two-stage system calculated in the previous iteration. Now we try every classifier in the pool (except the one picked out as a first stage) as a second stage in a three stage system. We pick the one resulting in the least complex three-stage cascade. If this least complex three-stage cascade is more complex than the previously selected two-stage cascade, then the selected two-stage cascade is declared to be the best cascade and the algorithm stops. Suppose that we found a three-stage system that is better than the two-stage system we previously selected, we then proceed to find the best four-stage cascade. We assume that the best four-stage cascade will have the first two stages of the previously selected three-stage cascade as

first and second stages. Now we want to select the third stage from the pool (the fourth stage will be S_F). We select the classifier that results in the least complex cascade as described above. And the process continues until the pool is empty or until we select a cascade that is more complex than a previously selected one. Figure 3 shows this procedure through an example. Algorithm 1 presents it formally.

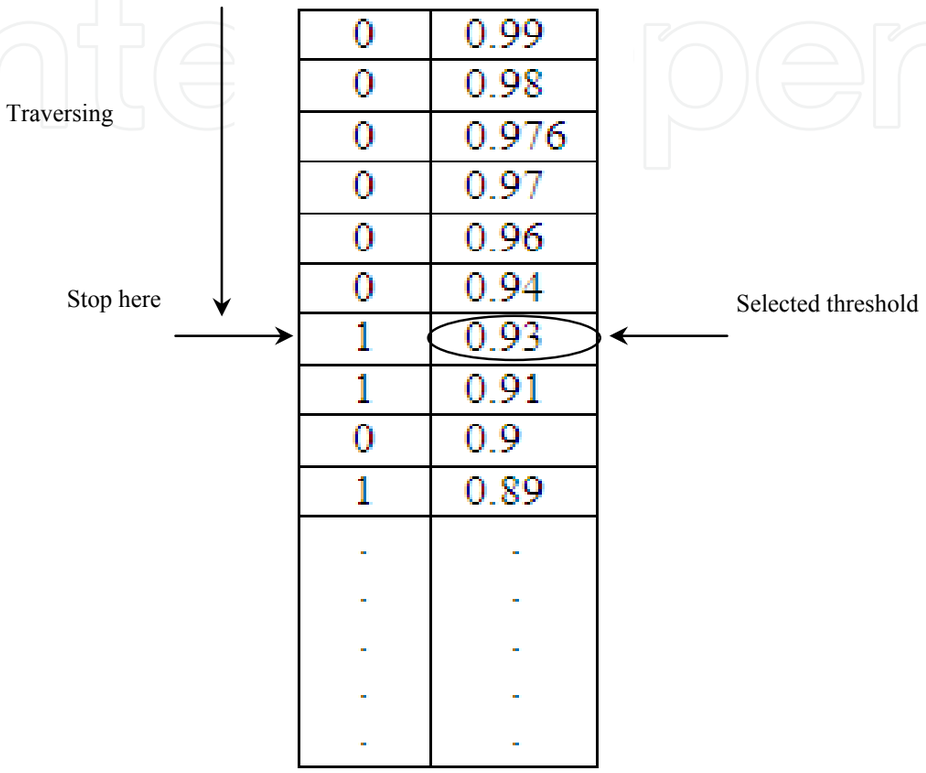


Fig. 2. A hypothetical example illustrating threshold selection process. Each row represents a different pattern of the validation set. The left entry of each record is labeled '1' if a classification error is committed while the pattern is correctly classified by the most powerful classifier S_F. The right entry of each record is the top decision score of the corresponding pattern. The patterns are ordered according to the decision score. This process is done for every classifier in the pool to get the corresponding set of thresholds.

Algorithm 1

// we will denote the pool of classifiers by the
//unordered set $\{\mathbf{S}_p\}$,
// and the final stage by S_F.
// the classifiers of $\{\mathbf{S}_p\}$ have arbitrary numbering
// $\mathbf{S}_c$ denotes the cascade selected so far

Inputs
A pool of classifiers $\{\mathbf{S}_p\}$ and a powerful classifier S_F
Outputs
A cascade S_c.
Initialize
best_complexity = *complexity*($[S_F]$).
Sc=[]. //an empty array
Begin
while (there are classifiers left in $\{\mathbf{S}_p\}$)
{

$$k = \arg\min_{i \forall S_i \in \{\mathbf{S}_P\}} \{complexity([\mathbf{S}_c \quad S_i \quad S_F])\}$$

$$new\_complexity = complexity([\mathbf{S}_c \quad S_k \quad S_F]).$$

if(*new_complexity* ≥ *best_complexity*)
{
$\mathbf{S}_c = [\mathbf{S}_c\ S_F]$.
Terminate.
}
else
{
$\mathbf{S}_c = [\mathbf{S}_c\ S_k\ S_F]$.
$\{\mathbf{S}_p\} = \{\mathbf{S}_p\} - S_k$.
best_complexity = *new_complexity*.
}
}
$\mathbf{S}_c = [\mathbf{S}_c\ S_F]$.
Terminate.

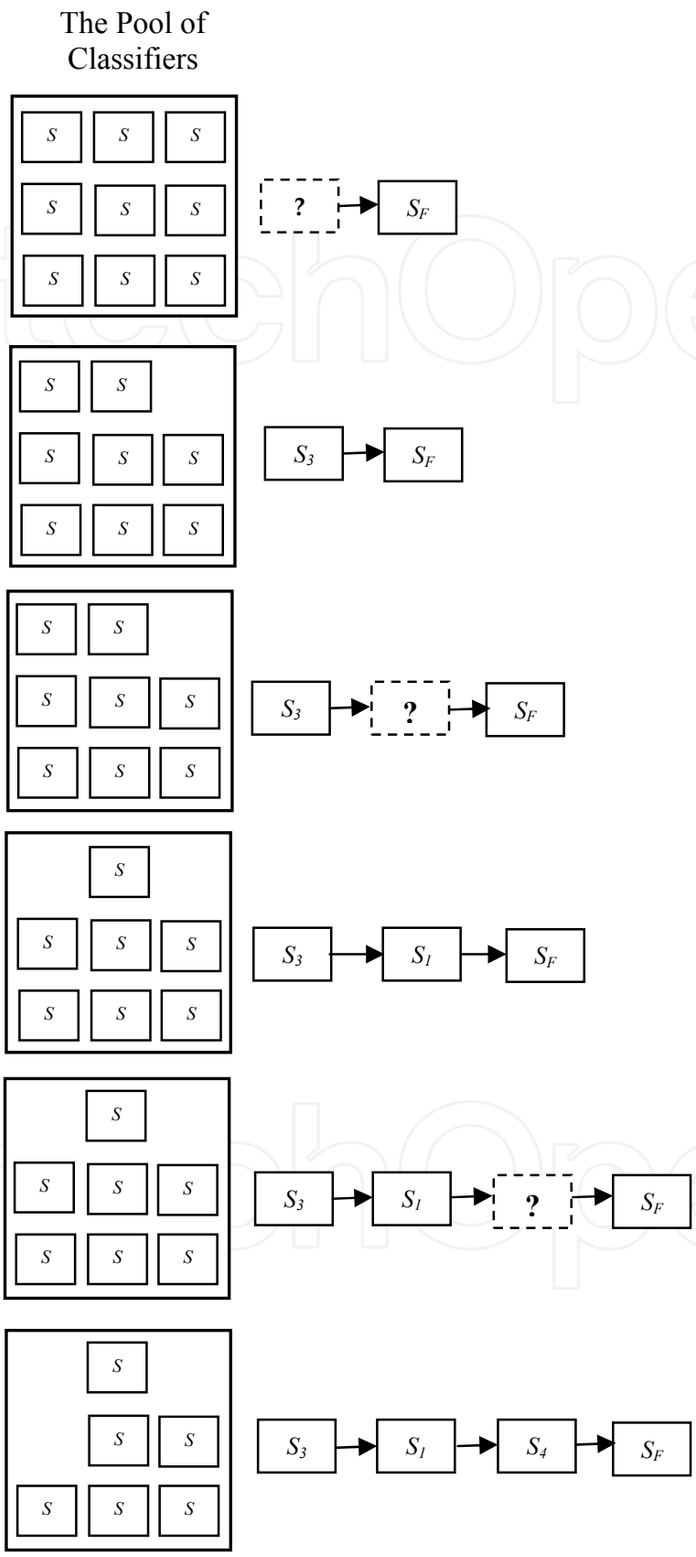


Fig. 3. An example of the process of selecting classifiers from the pool to form the cascade.

4. Experiments

The dataset we used in our experiments is the MNIST [LeCun et al, 1998]. The MNIST has a total of 70,000 digits which we partitioned into 3 parts: i) a training set, which is composed of 50,000 digits and used for training the classifiers, ii) a validation set, which is composed of 10,000 digits used for optimizing the cascade system, and iii) a test set, which is used for final testing of the cascade. We then transformed each digit image of all sets into 200-element feature vector using gradient feature extraction technique [Liu et al. 2003].

We trained 48 different classifiers with different complexities and accuracies on the training set. The strongest of those 48 classifiers has been chosen to be S_F and the remaining 47 classifiers now represent the pool of classifiers (N = 47). Three different types of classifiers are used: one-layer neural network (1-NN or linear classifier), two-layer neural network (2-NN), and SVM. For each classifier type, we generated a number of classifiers of different structures. First we ranked all 200 gradient feature elements according to their importance using ReliefF feature ranking technique [Kononenko, 1994]. Then, for 1-NN network, we generated a classifier that has as the most important 25 feature elements as input, and then another one with the most important 50 feature elements, then 75, an so on, till we finally generated a classifier with all the 200 feature elements. Hence, we have 8 different 1-NN classifiers. The same was done for SVM; hence, we have additional 8 different classifiers. This also was done for 2-NN, but for each number of inputs, we generated a classifier with different number of hidden units: 50, 100 150, and 200. Hence, we have 8 1-NN classifiers, 8 SVMs, and 8×4 2-NN classifier; so, we have a total of 48 classifiers of different structure and accuracies. We chose the most powerful classifier S_F to be the SVM classifier with 200 features as it gained the highest accuracy on the validation set. We measured the complexity of a certain classifier as the number of floating point operations (flops) [Ridder, 2002] it needs to classify one pattern divided by the number of flops the least complex classifier we have generated (that is, the 1-NN with 25 inputs) needs to classify one pattern. That is the complexity of 1-NN of 25 inputs is 1; and it was found that S_F has a complexity 5638.

We applied our greedy algorithm on the generated classifiers. Table 1 shows complexities and errors of S_F and complexities and errors of the cascade built using our greedy algorithm. It is obvious that the complexity is reduced considerably while not sacrificing the accuracy. We would like to note here that not all the 47 classifiers of the pool are selected to be members of the cascade. As shown in Figure 4, the cascade built using our algorithm has only 9 stages (including S_F). The advantage of using a large pool of classifiers is then to give a large room of possible cascades for the algorithm to choose from.

Now to show that our algorithm is near optimal, we compare it with the optimal case which is the exhaustive search we discussed in section 3.2. The exhaustive search has an exponential complexity in N (number of classifier in the pool, which is 47 in our case). Hence, we cannot apply the exhaustive search algorithm on all the 47 classifiers of the pool. We then selected 5 classifiers randomly and applied both the exhaustive and the greedy algorithms on them. We repeated this 15 times and calculated the average of the resulting errors and complexities (see Table 2). Table 2 shows that our algorithm achieves a reduction in complexity that is near to the optimal case. Furthermore, comparing Table 1 and Table 2 shows that applying the algorithm on whole the 47 classifiers has superior results than the case of using 5 classifiers. This shows that using more classifiers in the pool gives more opportunity for reducing the complexity. This fact favors algorithms that are scalable for large values of N like the one represented in this chapter.

5. Conclusion

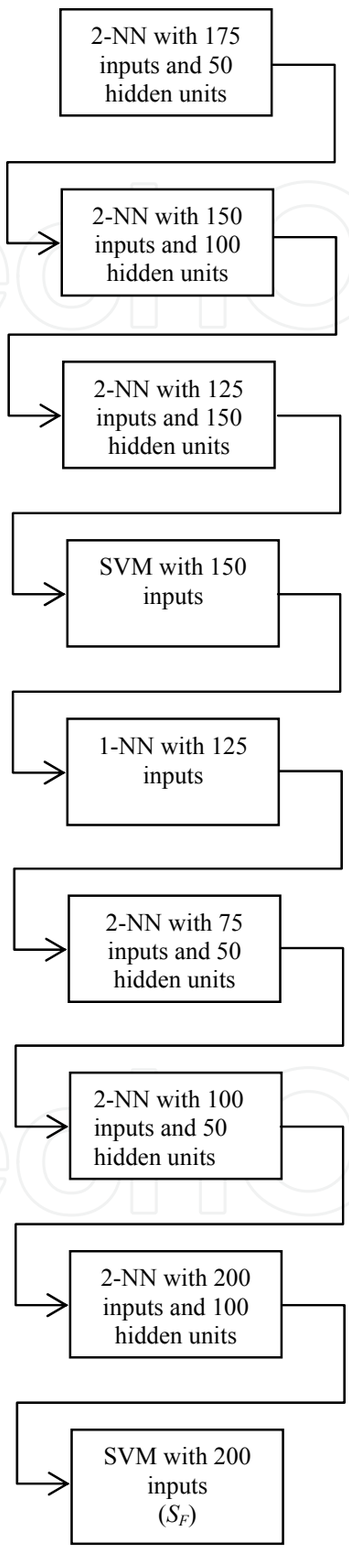


Fig. 4. The build cascade using the proposed algorithm.

In this chapter, we have proposed an algorithm to automatically generate classification cascades. The algorithm is fast and scalable. Experiments showed that our algorithm is efficient and builds classification cascades that substantially reduce the overall system complexity while preserving the accuracy.

	Complexity	**Error**	**# of stages**
S_F only	5638	66	1
Cascade built using our greedy algorithm	239.9	67	9

Table 1. The performance of our algorithm when given all the generated classifiers as compared to using S_F only in terms of errors and complexity (both are calculated using the test set)

	Avgerage Complexity	**Avgerage Error**	**Average # of stages**
SF only	5638	66	1
Cascade built using our greedy algorithm	480.9	66.9	5.2
Cascade built using exhaustive search	440.7	67.3	5.4

Table 2. The performance of our algorithm when given pools of randomly selected 5 classifiers (plus S_F) as compared to using S_F only and using the optimal procedure (exhaustive search) in terms of average errors and average complexity (calculated using the test set).

6. References

Chellapilla, K., M. Shilman, P. Simard, (2006a) "Combining Multiple Classifiers for Faster Optical Character Recognition", DAS, pp. 358-367.

Chellapilla, K.; Shilman, M. , Simard, P., (2006b), "Optimally Combining a Cascade of Classifiers", SPIE Document Recognition and Retrieval (DRR).

Ferri, C.; Flach, P. ,and Hernandez-Orallo, J., (2004) "Delegating classifiers," Proceedings of 21st International Conference on Machine Learning, pp. 37.

Giusti, N.; Masulli, F. and Sperduti, A. (2002), "Theoretical and experimental analysis of a two-stage system for classification," IEEE TPAMI, vol. 24, no. 7, pp. 893-904.

Gorgevik, D.& Cakmakov, D. (2004), "An efficient three-stage classifier for handwritten digit recognition", ICPR'04, pp. 1051-4651.

Kononenko, I. (1994) "Estimating attributes: analysis and extensions of Relief," ECML-94, pp. 171–182.

Kaynak, C. & Alpaydin, E. (1997), "Multistage classification by cascaded classifiers," Proceedings of 1997 IEEE international symposium on Intelligent Control, pp. 95-100.

Kuncheva, L., (2004), *Combining Pattern Classifiers*, Wiley-Interscience.

LeCun, Y.; Bottou, L. Bengio, Y. and Haffner, P. (1998), "Gradient-Based Learning Applied to Document Recognition", Proceedings of the IEEE, vol. 86 no. 11, pp. 2278-2324.

Liu, C.; Nakashima, K. Sako, Fujisawa, H. H., (2003), "Handwritten digit recognition: benchmarking of state-of-the-art techniques," Pattern Recognition, vol. 36, pp. 2271 – 2285.

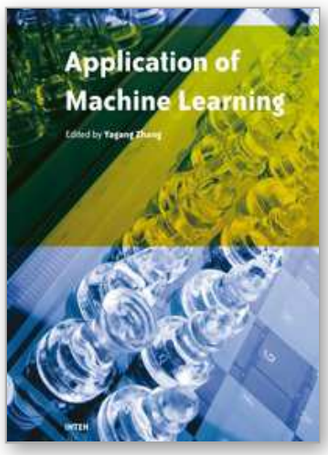
Oliveira, L.; Britto, A., Sabourin, R. (2005), "Optimizing class-related thresholds with particle swarm optimization", IJCNN, vol. 3, pp. 1511- 1516.

Pudil, P.; Novovicova, J. , Blaha, S., Kittler, J., (1992), "Multistage pattern recognition with reject option," 11th IAPR, pp. 92-95.

Rahman, A. & Fairhurst, M,. (1999), "Serial combination of multiple experts: a unified evaluation," Pattern Analysis and Applications, vol. 2, no. 4, pp. 292-311.

Ridder, D.; Pekalska, E., Duin, R. (2002), "The economics of classification: error vs. complexity", The 16th International Conference on Pattern Recognition, pp. 244-247.

Viola, P. & M. Jones, (2001), "Rapid object detection using a boosted cascade of simple features", ICPR, vol 1., pp. 511-518.



Application of Machine Learning
Edited by Yagang Zhang

ISBN 978-953-307-035-3
Hard cover, 280 pages
Publisher InTech
Published online 01, February, 2010
Published in print edition February, 2010

The goal of this book is to present the latest applications of machine learning, which mainly include: speech recognition, traffic and fault classification, surface quality prediction in laser machining, network security and bioinformatics, enterprise credit risk evaluation, and so on. This book will be of interest to industrial engineers and scientists as well as academics who wish to pursue machine learning. The book is intended for both graduate and postgraduate students in fields such as computer science, cybernetics, system sciences, engineering, statistics, and social sciences, and as a reference for software professionals and practitioners. The wide scope of the book provides them with a good introduction to many application researches of machine learning, and it is also the source of useful bibliographical information.

InTech Europe
University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

InTech China
Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821

IntechOpen