A Model-Based Approach for Building Optimum Classification Cascades

Ezzat El-Sherif and Sherif Abdelazeem,
Electronics Engineering Department, American University in Cairo
ezzatali@aucegypt.edu, shazeem@aucegypt.edu

For a long time, the main concern of the pattern recognition research community has been to achieve high accuracy. For example, in the area of Multi-Classifier Systems (MCS), researchers have developed very powerful techniques that combine large number of classifiers through complex combination schemes to achieve satisfactory accuracies. However, this was at the expense of complexity. Even powerful single classifiers (e.g. SVM) have very high complexity. This might give the impression that high accuracies could not be achieved without sacrificing recognition time.
Classification cascades are relatively neglected because speed is usually considered a secondary issue by researchers in pattern recognition field. This fact is going to change in the near future because as the world relies more and more on the Internet, web applications are going to include very complex pattern recognition and data mining tasks that are required to be done online.
Classification cascades are usually created manually using domain knowledge and are composed in most cases of two or three stages. In this chapter, a model-based algorithm of automatic generation of optimum classification cascades is devised. Given a large pool of classifiers (of size *N*), it builds a cascade that achieves the lowest possible recognition time while preserving the accuracy of the most powerful classifier in the pool. The proposed algorithm has a low complexity of $O(N^2)$ where N is the number of classifiers in the pool. This gives us the freedom of using a large pool of classifiers which leads to more efficient cascades. Other cascade design techniques devised in the literature have very high complexity which hinders using large pool of classifiers.
In this chapter we also analyze the performance of the devised algorithm showing its powerfulness and limitations. Also we present an algorithm for building a classification cascade of a given fixed length. This helps building cascades with space complexity constraints and helps in analyzing the performance of the devised algorithm for building optimum classification cascades.

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 accuracies could not be achieved without sacrificing recognition time. In fact, this is not true. The high recognition time of a classifier in many cases is due to improper resource allocation. To achieve a high recognition rate, the classifier is built to recognize the hardest of patterns; though most of the patterns are 'regular' patterns and could be classified using a simple classification technique. This observation led to the development of cascade systems which is the main concern of this paper. 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 [VIOLA & JONES 2001, LUO 2005, CHEN, X. & YUILLE 2005, YUANN ET AL. 2005, Brubaker et al. 2006, SUN ET AL. 2004, WU ET AL. 2008].
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.

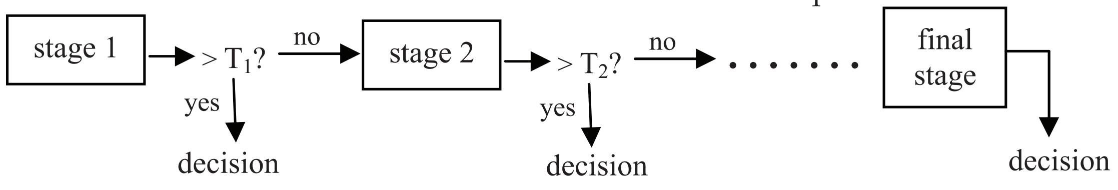


Fig 1 Typical classification cascade system.

The remaining of the chapter is organized as follows. Section 2 presents an algorithm for automatically generating optimum classification cascades. In section 3, we present an algorithm for generating optimum classification cascades of specific number of stages. Section 4 presents an experimental validation of our proposed algorithms. In section 5 we review previous works on classification cascades and in section 6 we conclude.

2. Model-Based Algorithm for Automatically Generating Optimum Classification Cascades

In this section a novel algorithm for automatically generating optimum classification cascades is proposed that achieves as high accuracy as we can get with the lowest complexity possible. The proposed algorithm is built according to a model of how classification cascade works; thus we named it 'model-based' approach. Assume that we have a pool of N classifiers: S_1, S_2, …, S_N, and a powerful classifier S_F that achieves an accuracy higher than any classifier in the pool. However, S_F has a very high complexity. The proposed algorithm automatically builds a cascade that achieves accuracy not less than that of S_F with the lowest complexity possible. Classifiers of the pool are assumed to be trained independently before applying the proposed algorithm, and no further training is done for any of them after applying the algorithm.

2.1 Problem statement

We first present our notation. 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. $A_1, A_2, A_3, \ldots \in \{\mathbf{A}\}$). Note that the subscripts of the unordered set are arbitrary and hold no ordering significance. An ordered set (or an array) is denoted by just a boldface character and its elements by the same character but in italics and subscripted by numbers according to their order in the array (e.g. $A_1, A_2, A_3, \ldots \in \mathbf{A}$, where A_1 is the first element in **A**, and A_2 is the second element, etc.). $\mathbf{B} \subset \mathbf{A}$ means that all the elements of the ordered set **B** exists in the ordered set **A** with the same order. $\mathbf{B} \subset \{\mathbf{A}\}$ means that all the elements of the ordered set **B** exist in the unordered set {A}. $\mathbf{B} \equiv \{\mathbf{A}\}$ means that the elements of **B** are the same as that of **A** but with order, i.e. **B** is an ordered version of **A**. We enumerate the elements of an unordered set {**A**} as follows $\{\mathbf{A}\} = \{A_1, A_2, \ldots\}$ and the elements of an ordered set **A** as follows $\mathbf{A} = [A_1\ A_2\ \ldots]$. **C**=[**A B**] means that the ordered set **C** is a concatenation of the two ordered sets **A** and **B**. 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.

Here we state our problem. Given a set of N classifiers $\{\mathbf{S}\} = \{S_1, S_2, S_3, \ldots, S_N\}$ and a powerful classifier $S_F \notin \{\mathbf{S}\}$ that achieves a satisfactory accuracy. The problem is to select an ordered set $\mathbf{S}^{opt} \subset \{\mathbf{S}\}$ and a corresponding ordered set of thresholds $\mathbf{T}^{opt}$ that makes $[\mathbf{S}^{opt}\ S_F]$ if put in a cascade structure give the optimal cascade. We mean by 'optimal cascade' the one that gives the least possible complexity with accuracy not less than that of S_F.

2.2 The Algorithm

In this section, an algorithm that automatically generates optimal classification cascades will be presented. The algorithm is composed of three major steps:

i. Find the set of thresholds {**T**}. Each element of {**T**} is a threshold for the corresponding classifier in {**S**} such that $\mathbf{T}^{opt} \subset \{\mathbf{T}\}$.

ii. Sort the set {**S**} to form $\mathbf{S}^{ord} \equiv \{\mathbf{S}\}$ such that $\mathbf{S}^{opt} \subset \mathbf{S}^{ord}$. With the same ordering pattern of $\mathbf{S}^{ord}$, sort {**T**} to form $\mathbf{T}^{ord}$.

iii. From $\mathbf{S}^{ord}$, select $\mathbf{S}^{opt} \subset \mathbf{S}^{ord}{}_{t}$ and the corresponding $\mathbf{T}^{opt} \subset \mathbf{T}^{ord}$.

These three steps is illustrated in Figure 2 and described in the following subsections. But we want here to give a note about how {**S**} and S_F were generated. The dataset available for training and testing the system was first partitioned into 3 parts: training set, validation set, and test set. Then, many classifiers were generated with different accuracies and complexities and were trained using the training set. The most powerful of these classifiers was found and denoted by S_F , and the rest were grouped in the set {**S**}. The validation set is used to find the best cascade. The test set is used for testing the overall system. The algorithm proposed in this chapter for generating classification cascade trains the classifiers only once and uses different performance measurements (specifically, their rejection rates and complexities, as will be discussed soon) of these classifiers using the validation set to build the cascade. No retraining of classifiers is held.

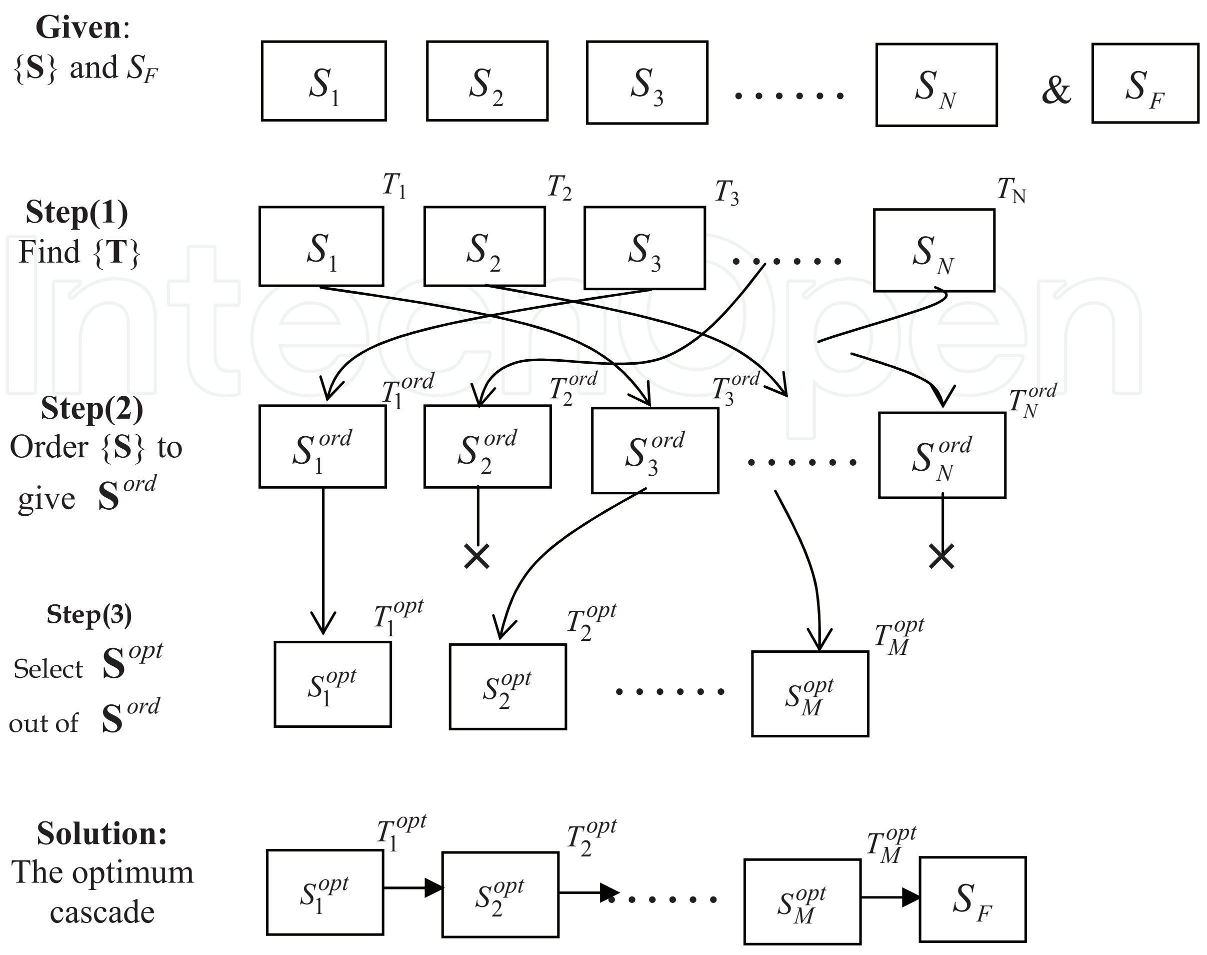
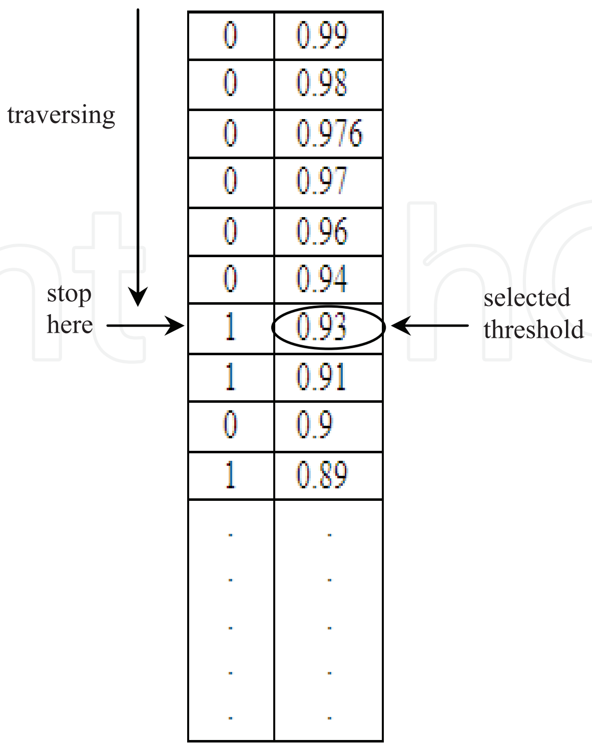

Given:
{**S**} and S_F
S_1 S_2 S_3 …… S_N & S_F
Step(1)
Find {**T**}
T_1 T_2 T_3 T_N
Step(2)
Order {**S**} to give $\mathbf{S}^{ord}$
T_1^{ord} T_2^{ord} T_3^{ord} T_N^{ord}
S_1^{ord} S_2^{ord} S_3^{ord} …… S_N^{ord}
Step(3)
Select $\mathbf{S}^{opt}$ out of $\mathbf{S}^{ord}$
T_1^{opt} T_2^{opt} T_M^{opt}
S_1^{opt} S_2^{opt} …… S_M^{opt}
Solution:
The optimum cascade
S_1^{opt} S_2^{opt} …… S_M^{opt} S_F

Fig. 2. An example to show how the proposed algorithm works.

2.2.1 Step1: Find {T} for {S}

Our procedure for finding {**T**} of {**S**} will be as follows. Using every classifier $S_i \in \{\mathbf{S}\}$, classify the patterns of the validations set and order them according to the confidence scores they are given by S_i. Traverse these ordered patterns from 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 the errors of the overall cascade and can be safely ignored, and we continue traversing the patterns. We stop when we hit a pattern that is falsely classified by the classifier under consideration S_i but correctly classified by $S_{F.}$ Then we set the threshold T_i of the classifier S_i to be the confidence score of the pattern we stopped at. We do the same for all the classifiers in the set {**S**} to form the corresponding set of thresholds {**T**}. This procedure is illustrated in Figure 3.



0	0.99
0	0.98
0	0.976
0	0.97
0	0.96
0	0.94
1	0.93
1	0.91
0	0.9
1	0.89
.	.
.	.
.	.
.	.
.	.

Fig. 3. A hypothetical example illustrating threshold selection process for some classifier S_i belongs to the pool {**S**}. Each row represents a different pattern of the validation set classified by S_i. The right entry of each record is labeled '1' if a classification error is committed by S_i while the pattern is correctly classified by the most powerful classifier S_F. The second entry of each record is the top decision score of the corresponding pattern given to it by S_i. The patterns are ordered according to the decision score. This process is done for every classifier in {**S**} to get the corresponding set of thresholds {**T**}.

2.2.2 Step 2: Sort {S} to form $\mathbf{S}^{ord} = \{\mathbf{S}\}$

The criterion by which we sort {**S**} is based on the following assumption:

Assumption 1 Using sufficiently tough thresholds T_i and T_j for the two classifiers S_i and S_j, respectively, if the classifier S_j has a lower rejection rate than S_i, then it is said that S_j is more powerful than S_i; and S_i would reject all the patterns that S_j would reject.

The "rejection rate" of a certain classifier S_i using threshold T_i is the number of rejected patterns divided by the number of validation set patterns if the threshold T_i is applied on its output.

This assumption, while not perfectly realistic, is reasonable. Since S_j is more powerful than S_i, then it will be of little possibility for S_i to *confidently* classify some pattern that was hard for S_j to classify. To what extent this assumption is valid will be discussed in section 4.2.
Assumption 1 leads to a great simplification of the system design. Suppose that while designing a cascade system we put a classifier S_2 that has high rejection rate after a classifier S_1 that has lower rejection rate. According to Assumption 1, S_1 is more powerful than S_2, and S_2 would be then useless. This is because S_2 would reject all the patterns that are rejected from S_1. In this case, S_2 would do nothing except increasing the complexity of the cascade.

This means that the only reasonable criterion of sorting the classifiers in the cascade is to sort them by decreasing rejection rates. By this principle, {**S**} is sorted to give $\mathbf{S}^{ord}$. The corresponding set of thresholds for $\mathbf{S}^{ord}$ will be $\mathbf{T}^{ord}$. Now we are guaranteed that $\mathbf{S}^{opt} \subset \mathbf{S}^{ord}$, because any other order would give more complex cascade.

2.2.3 Step 3: Select $\mathbf{S}^{opt}$ out of $\mathbf{S}^{ord}$

Now we want to select $\mathbf{S}^{opt} \subset \mathbf{S}^{ord}$. This selection process, if done exhaustively, is of complexity $O(2^N)$. Hence, exhaustive search would not be feasible for large values of *N*. In this section an algorithm is suggested for finding $\mathbf{S}^{opt} \subset \mathbf{S}^{ord}$ of complexity $O(N^2)$.

Now, a formula will be introduced to have an estimate for the average complexity of a cascade given the complexities and rejection rates of the constituent classifiers. This formula will be used to find the best cascade in a sequential manner without doing an exhaustive search. How this formula is deduced can be shown through an example. Let $\mathbf{S}' = [S'_1\, S'_2\, S'_3 \ldots . S'_M\,] \subset \mathbf{S}^{ord}$ have thresholds $\mathbf{T}' = [T'_1 T'_2 T'_3 \ldots T'_M\,]$, complexities $\mathbf{C}' = [C'_1\, C'_2\, C'_3 \ldots C'_M]$, and rejection rates $\mathbf{R}' = [R'_1\, R'_2\, R'_3 \ldots . R'_M\,]$. Let C_F be the complexity of S_F. Figure 4 represents each stage in the cascade by a rectangular representing the validation set. The hashed portion of the rectangular is the portion that is not rejected from the validation set. Assumption 1 is evident in Figure 4 as each stage rejections include the rejections of preceding stage. It is obvious from Figure4 that the cascade $[\mathbf{S}'\, C_F\,]$ has an overall complexity C_{tot}, where,

$$C_{tot} = C'_1 + R'_1 C'_2 + R'_2 C'_3 + \ldots\ldots + R'_{M-1} C'_M + R'_M C_F \tag{1}$$

We note here that Equation (1) validity is dependent on a hypothetical model of how the classification cascades works which is represented by Figure 4. This model in its turn is dependent on the validity Assumption 1.

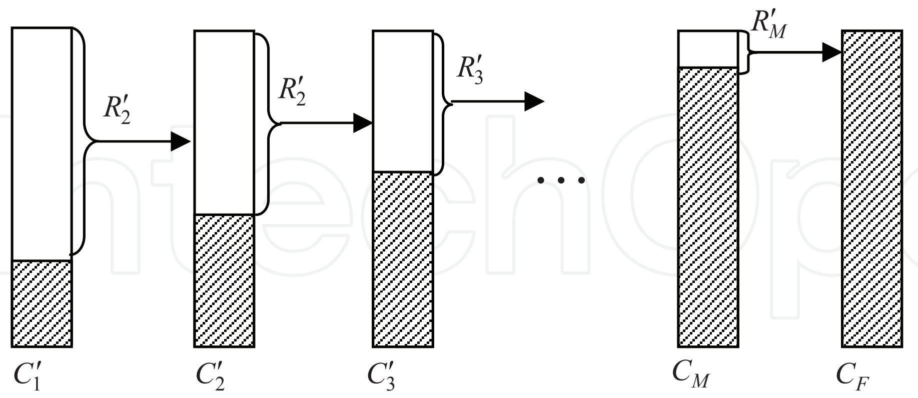


Fig. 4. A cascade represented by accepted and rejected patterns. The shaded region represents the accepted patterns.

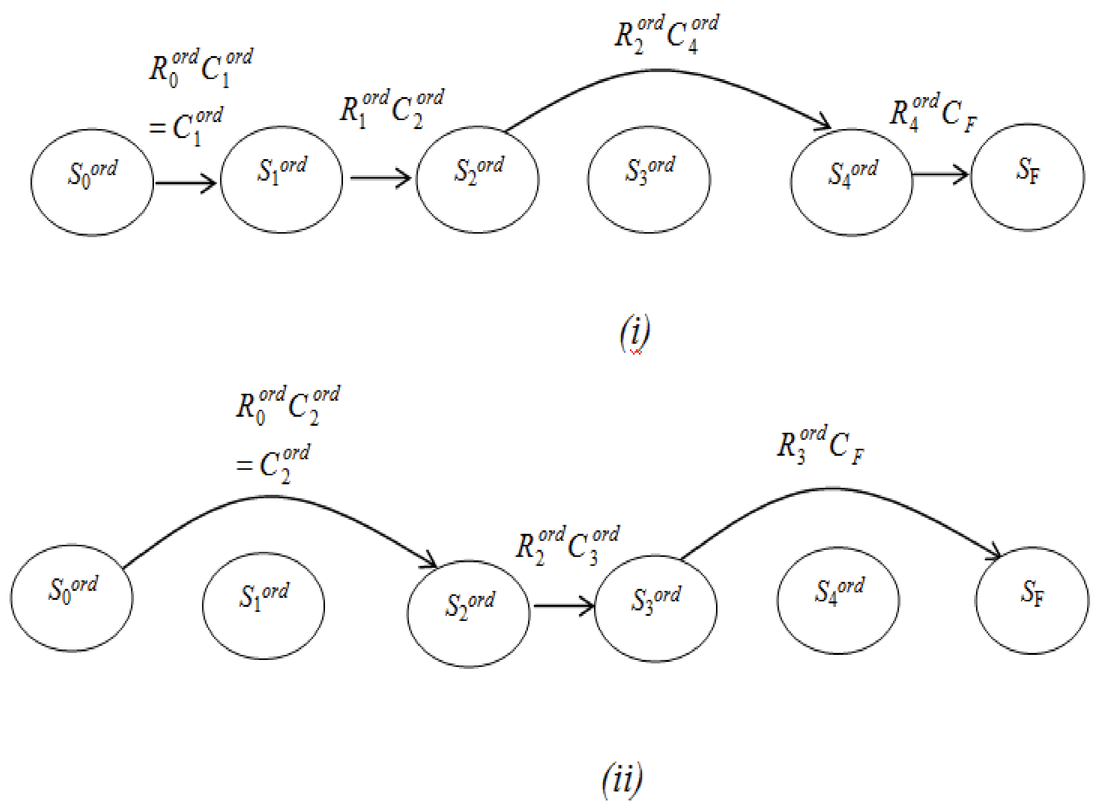


Fig. 5. Two examples to illustrate how $\mathbf{S}^{opt}$ is selected from $\mathbf{S}^{ord}$

Equation (1) suggests a very simple algorithm to find $\mathbf{S}^{opt} \subset \mathbf{S}^{ord}$. The algorithm is going to be described through an example. Assume that $\mathbf{S}^{ord}$ is composed of four stages: S_1^{ord}, S_2^{ord}, S_3^{ord}, S_4^{ord}, and the last stage S_F. We can represent such scheme by successive nodes in a digraph as shown in Figure 5. For convenience, we added a dummy node S_0^{ord} before S_1^{ord}. Node S_0^{ord} is the source of all the patterns to be classified and has zero complexity and a rejection rate of 1 (i.e. $C_0^{ord} = 0, \text{and } R_0^{ord} = 1$). The problem now is to get the path from S_0^{ord} to S_F that leads to the least complex cascade. Now, define the distance from node S_i^{ord} to node S_j^{ord} for $j>i$ to be equal to $R_i^{ord}C_j^{ord}$. Hence, each cascade can be represented by some path in $\mathbf{S}^{ord}$. For example, the path indicated in Figure 3(*i*) has a distance of $C_1^{ord} + R_1^{ord}C_2^{ord} + R_2^{ord}C_4^{ord} + R_4^{ord}C_F$ which is equal to the cascade complexity. Another possible path of complexity $C_2^{ord} + R_2^{ord}C_3^{ord} + R_3^{ord}C_F$ is shown in Figure 3(*ii*). The problem of finding the least complex cascade can be seen then as finding the shortest path in a directed acyclic graph (DAG) which is a well-known problem. However, for completeness, we will present its solution in the context of our problem.

Assume that each node tries to find the shortest path from itself to S_F as well as the distance of this path. All the nodes can easily collect this information if we started by last stage and proceeded backwardly to the first. For example, in Figure 5 we start by S_4^{ord}. The shortest path from S_4^{ord} to S_F is obviously [S_4^{ord} S_F] as this is the only possible path. The distance of this path is $R_4^{ord}C_F$. For node S_3^{ord}, we have two possible paths: [S_3^{ord} S_4^{ord} S_F] and [S_3^{ord} S_F]. Hence, we compare the two paths distances: $R_3^{ord}C_4^{ord} + R_4^{ord}C_F$ and $R_3^{ord}C_F$, respectively. The shortest path as well as its distance are found and saved at node S_3^{ord}. Then, we proceed to S_2^{ord}. Node S_2^{ord} has just 3 options: to jump to S_3^{ord}, to jump to S_4^{ord}, or to jump to S_F. If we jumped form S_2^{ord} to S_3^{ord}, and if we are interested in just shortest paths, the path from S_3^{ord} to S_F would be previously decided by S_3^{ord} and need not to be recalculated; and the complexity of the path in this case is $R_2^{ord}C_3^{ord}$ + the distance of shortest path from S_3^{ord} to S_F. The remaining two options for S_2^{ord} (to jump to S_4^{ord} and to jump to S_F.) are also examined and the option of the least distance is saved at node S_2^{ord}. The same procedure is done for S_1^{ord} and S_0^{ord}. Finally, the shortest path from S_0^{ord} to S_F is the cascade of least complexity.

To present the algorithm more formally, it would be of much help to introduce some simple definitions. Define 'single-hop distance' $d_h(S_i^{ord}, S_j^{ord})$ between stages S_i^{ord} and S_j^{ord}, $j \geq i$, $S_i^{ord}, S_j^{ord} \in \mathbf{S}^{ord} \forall i, j$,

$$d_h(S_i^{ord}, S_j^{ord}) = \begin{cases} R_i^{ord} C_j^{ord} & , j > i \\ 0 & , i = j \\ \text{undefined} & , j < i \end{cases} \tag{2}$$

Define also the shortest distance $D(S_i^{ord}, S_j^{ord})$ between stages S_i^{ord} and S_j^{ord}, $j \geq i$ as the minimum distance between stages S_i^{ord} and S_j^{ord}. Note that as in the case of single-hop distance, for *i=j*, the distance will be 0 and undefined for *i>j*. Clearly the shortest path is a succession of single hops, whose distances are calculated as given in Equation (2).

Finally, define the shortest path $\mathbf{P}(S_i^{ord}, S_j^{ord})$ between stages S_i^{ord} and S_j^{ord}, $j \geq i$ as the path of shortest distance (that is of $D(S_i^{ord}, S_j^{ord})$) between stages S_i^{ord} and S_j^{ord} represented as an ordered set starting with S_i^{ord} and ending with S_j^{ord}. If *i=j*, then $\mathbf{P}(S_i^{ord}, S_j^{ord})$ = [S_i^{ord}]; and $\mathbf{P}(S_i^{ord}, S_j^{ord})$ is undefined for *i>j*.

Algorithm 1 shows the three steps of the proposed algorithm for automatic generation of optimum classification cascades. Step 3 uses the definitions given above to show how to Select $\mathbf{S}^{opt} \subset \mathbf{S}^{ord}$.

Algorithm 1 The three steps of the proposed algorithm to find optimum classification cascade

```
Step 1. Use the validation set to get a set of thresholds
$\{\mathbf{T}\}$ for $\{\mathbf{S}\}$ that ensures that none of the classifiers in
$\{\mathbf{S}\}$ commits any error that is not committed by $S_F$.

Step 2. Sort $\{\mathbf{S}\}$ in a descending order of rejection rates
to form $\mathbf{S}^{ord}$.

Step 3. Select $\mathbf{S}^{opt} \subset \mathbf{S}^{ord}$ as follows,

$D(S_N^{ord}, S_F) = d_h(S_N^{ord}, S_F) = R_N^{ord} C_F$
$P(S_N^{ord}, S_F) = [S_F]$

for $i = N - 1$ down to 0
    $k = \underset{j=i+1,i+2,\dots,N}{\arg\min} \, (d_h(S_i^{ord}, S_j^{ord}) + D(S_j^{ord}, S_F))$
    $D(S_i^{ord}, S_F) = d_h(S_i^{ord}, S_k^{ord}) + D(S_k^{ord}, S_F)$
    $\mathbf{P}(S_i^{ord}, S_F) = [S_i^{ord} \;\; P(S_k^{ord}, S_F)]$
end

// $\mathbf{P}(S_0^{ord}, S_F)$ is the optimum cascade, where
// $\mathbf{P}(S_0^{ord}, S_F) = [\mathbf{S}^{opt} \;\; S_F]$
// and $D(S_0^{ord}, S_F)$ is its complexity
```

3. An Algorithm for Generating the Optimum Classification Cascade of a Given Length

In the previous section, we introduced an algorithm that finds the least complex cascade that achieves an accuracy not less than that of S_F. The found cascade in this case would be of any length ≤ N (for convenience, the cascade length would be used to mean the number of stages of the cascade excluding S_F). In some cases, due to memory limitation, we would like to use a cascade of smaller length. In this section, an algorithm will be introduced that builds the best cascade of any preferred length. This will also help in studying the effect of increasing the number of stages in a cascade on its performance (refer to section 4.3).

To introduce the algorithm, we will first add two more definitions. Define the shortest path of length *n*, $\mathbf{P}_n(S_i^{ord}, S_j^{ord})$, between stages S_i^{ord} and S_j^{ord} as the shortest path of *n* hops between nodes S_i^{ord} and S_j^{ord}. Define also shortest distance of length *n*, $D_n(S_i^{ord}, S_j^{ord})$, between nodes S_i^{ord} and S_j^{ord} as the distance (complexity) of $\mathbf{P}_n(S_i^{ord}, S_j^{ord})$.

The algorithm for finding the best cascades of different lengths is presented in Algorithm 2. Step 1 and 2 of the algorithm is the same as that of Algorithm 1. Step 3 is a modified version of Step 3 of Algorithm 1. Instead of saving only the information of best path, each node saves the best path of every possible length.

4. Experiments and Discussion

In this section we present some experiments that validate the proposed approach and shed light on its strengths and limitations.

4.1 Model-Based Approach versus DFS

The problem of finding optimal classification cascades has other possible solutions than that proposed (e.g. using stochastic search techniques [Chellapilla et al. 2006a]). But we will compare the proposed technique with the most elegant solution, that is the one using depth-first search (DFS) devised by Chellapella et al. [Chellapilla et al. 2006b] (refer to section 5.6 for more details on this technique). Given a set of *N* ordered classifiers, the DFS algorithm searches systematically all possible cascade structures with *Q* permissible threshold values to find the optimum cascade. Using some heuristics, all the search space need not be visited and extensive pruning of search space is possible. The goal of the DFS algorithm is more general than ours. Given a certain permissible margin of error, it finds the least complex cascade. The proposed model-based system design procedure on the other hand finds least complex cascade that commits *no more error* than the last stage in the cascade. Actually, The DFS algorithm could be used to solve the problem if we adjusted the margin of error to be that of the final stage.

Algorithm 2. Generating optimum cascades of different lengths

Step 1. Use the validation set to get a set of thresholds {**T**} for {**S**} that ensures that none of the classifiers in {**S**} commits any error that is not committed by S_F.

Step 2. Sort {**S**} in a descending order of rejection rates to form $\mathbf{S}^{ord}$.

Step 3. Select cascades of different lengths,

```
$D_1(S_N^{ord}, S_F) = d_h(S_N^{ord}, S_F) = R_N^{ord} C_F$
$\mathbf{P}_1(S_N^{ord}, S_F) = [S_F]$

for $i = N - 1$ down to 0

    $D_1(S_i^{ord}, S_F) = d_h(S_i^{ord}, S_F) = R_i^{ord} C_F$
    $\mathbf{P}_1(S_i^{ord}, S_F) = [S_i^{ord} \ S_F]$

    for $n = 2$ to $N - i + 1$
        $k = \underset{j=i+1,i+2,\ldots,N-n+2}{\arg\min} (d_h(S_i^{ord}, S_j^{ord}) + D_{n-1}(S_j^{ord}, S_F))$
        $D_n(S_i^{ord}, S_F) = d_h(S_i^{ord}, S_k^{ord}) + D_{n-1}(S_k^{ord}, S_F)$
        $\mathbf{P}_n(S_i^{ord}, S_F) = [S_i^{ord} \ \ \mathbf{P}_{n-1}(S_k^{ord}, S_F)]$
    end $n$

end $i$
// $\mathbf{P}_n(S_0^{ord}, S_F)$ is the least complex cascade
//of length $n, n = 1,2,....,N$
// and its complexity is $D_n(S_0^{ord}, S_F)$
```

The proposed algorithm has a number of advantages over the DFS algorithm:

i. DFS algorithm has a training complexity of $O(Q^N)$, where Q is the number of thresholds quantization levels which equals 32 as suggested by [Chellapilla et al. 2006b]. While extensive pruning of search space is possible, the algorithm takes very long training times especially for large N. On the other hand, the proposed algorithm is of only a complexity of $O(N^2)$. This means that the proposed procedure is faster in training.

ii. Besides being faster, the proposed algorithm is scalable to large numbers of N. While the actual number of stages selected for the optimum cascade is only around 6 or 7 stages, making a large number of classifiers N available for the algorithm gives it more flexibility to *select* the most suitable stages for the cascade. In case of DFS, and to make the algorithm terminates with reasonable time, we should select by hand around 8 stages for the algorithm as done by [Chellapilla et al. 2006b]. This makes the proposed algorithm a means for system design and for classifier selection at the same time. This means also that the proposed system is more automatic than DFS.

In this section, we will compare the performance of the proposed system design procedure with the DFS algorithm on the digit recognition problem. The dataset used in the experiments is the MNIST [LECUN ET AL. 1998]. The MNIST has a total of 70,000 digits which partitioned into 3 parts: *i*) a training set, which includes 50,000 digits and used for training the classifiers, *ii*) a validation set, which contains 10,000 digits used for optimizing the cascade system, and *iii*) a test set, which is used for final testing of the cascade. Each digit image of all sets was transformed into 200-element feature vector using the gradient feature extraction technique [LIU ET AL. 2003].

Forty-eight different classifiers were trained with different complexities and accuracies on the training set. Three different types of classifiers are used: one-layer neural network (1-NN or linear classifier), two-layer neural network (2-NN), and RBF SVM [WEBB 2002]. For each classifier type, a number of classifiers of different structures were generated. First all 200 gradient feature elements were ranked according to their importance using ReliefF feature ranking technique [KONONENKO 1994]. Then, for 1-NN, a classifier was generated that has as the most important 25 feature elements as input, and then another one with the most important 50 feature elements, then 75, and so on, till finally a classifier with all the 200 feature elements was generated. 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, a classifier was generated with different number of hidden units: 50, 100 150, and 200 (i.e. 2-layer neural network of structures: 25-50-10, 25-100-10, …, 25-200-10, 50-50-10, 50-100-10, ……, 200-200-10). Hence, we have 8 1-NN classifiers, 8 SVMs, and 8×4 2-NN classifier; hence we have a total of 48 classifiers of different structure and accuracies. Table 1 shows the performance of the most powerful classifier (that is, the SVM classifier with 200 feature elements): its number of errors and its complexity on the test set. The complexity of a certain classifier is measured as the number of floating point operations (flops) [RIDDER ET AL. 2002] it needs to classify one pattern divided by the number of flops the least complex classifier generated (that is, the 1-NN with 25 inputs) needs to classify one pattern.

Test set errors	**Complexity**
66	5638.2

Table 1 The performance of the most powerful classifier (SVM with 200 features as input)

	Complexity	**Errors**	**Number of Stages**
Model-Based on 5 randomly selected stages from {S}	376.9	67.3	3.5
DFS on 5 randomly selected stages from {S}	358.7	66.5	4.9

Table 2 Average complexity, average errors, and average number of stages when applying both model-based approach and DFS on 5 randomly selected stages from the pool of classifiers

	Complexity	**Errors**	**Number of Stages**
Model-Based applied to the whole {S}	187.8	70	6

Table 3 Complexity and errors of model-based approach on the entire pool of classifiers

Now, all the generated 48 classifiers can be handed to the proposed algorithm to generate the best cascade out of them. However, this would not be possible for the DFS algorithm due to its large complexity. In order to compare the proposed algorithm with DFS, the most powerful classifier (that is, the SVM with 200 features) was used as the last stage and then randomly selected 5 classifiers from the remaining 47 classifiers. The same was done 15 times; each time with different randomly selected 5 classifiers. For each of these selections, both model-based algorithm and DFS was used to select the optimum cascade. The averages of the 15 trials of both cases are calculated and summarized in Table 2. The average cascade length (number of stages excluding S_F) is also indicated for both cases. Table 2 shows that DFS got slightly better results.

However, the main advantage of the proposed approach is that we can use classifiers pools of large sizes *N* which is impossible for DFS. The entire set of 48 classifiers was given to the proposed algorithm to build the optimum cascade out of them. The results are shown in Table 3. We see that the cascade built when the proposed algorithm is given all the 48 has a complexity of 187.8 which is lower than that of both model-approach and DFS when applied to 5 randomly selected stages. This clarifies the importance of using large number of classifiers *N*, and hence, the importance of the low complexity of our proposed algorithm.

4.2 The Validity of Assumption 1

The optimality of Algorithm 1 is guaranteed if Equation (1) is valid. Equation (1) on its turn is based on the validity of Assumption 1. In this section we are going to explore the condition under which Assumption 1 is valid.

Assumption 1 states that the weaker classifier rejects all the patterns that the stronger classifier rejects. In fact, this is true only when the stronger classifier is *much* stronger than the weaker. Classifiers of near degree of strength reject overlapping sets of patterns; but never totally conform to Assumption 1; this phenomenon is called 'diversity' and is exploited in building classifiers ensembles [Kuncheva 2000]. Figure 6 is a scatter plot showing the dependency of the invalidity of Assumption 1 for different classifier pairs (taken from the pool of 48 classifiers mentioned above) on the difference in strength between them. The invalidity of Assumption 1 is measured by the number of cases that violated it and put on the vertical axis. The difference in strength between two classifiers is represented by the difference in rejection rates between them. As clear from Figure 6, the more the difference in rejection rates between two classifiers is, the less the number of violations of Assumption 1 is found.

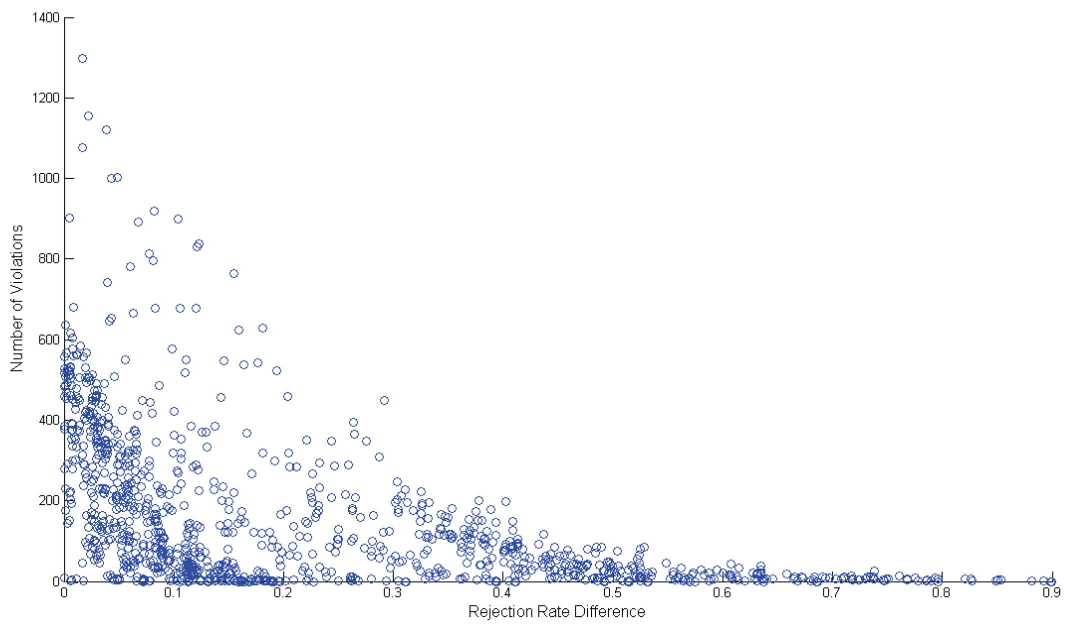


Fig. 6. A scatter plot that showing the dependency of the validity of Assumption 1 on the difference in strength between the two classifiers.

But how could the inaccuracy of Assumption 1 affect the structure of cascades built by the model-based approach? In fact, Assumption 1 leads us to believe that putting two classifiers of near degree of strength one after the other is of no use. This is because the second classifiers will not confidently recognize much of the patterns rejected from the first classifier, and hence would not do but just increasing the complexity of the cascade. This leads the model-based approach selects classifiers of very distant rejection rates from each other to build the cascade. Because the rejection rate domain is finite (from 0 to 1), this makes the model-based approach tends to build cascades of small lengths. This is clear if we refer to Table 2 and compared the average cascades lengths built by model-based approach and DFS.

Now, would this lead us to miss an opportunity for building less complex cascades? The answer is yes. Putting a classifier of near strength to some classifiers increases its complexity indeed *but* might result in a drop in rejection rate. Consider the following hypothetical example. Suppose that there are two classifiers; both have rejection rate of 0.5. According to Assumption 1, there is no point of putting one after the other in a cascade as this will lead to more complex system of the same rejection rate of 0.5. But according to the concept of diversity, there is. To take the extreme case of Assumption 1 violation, assume that the sets of patterns rejected from the two classifiers are exclusive. This will make the overall rejection rate of both classifiers when put one after the other in a cascade drop to 0. Of course, this extreme case does not occur in reality, but this shows how two classifiers of near strength could lead to rejection rates enhancements, and hence to more efficient classification cascades. This is clear from Table 2, as the DFS (which is an exhaustive procedure) finds cascades of better performance than model-based approach. However, DFS is of very high complexity and does not scale to large pool sizes and this leads us to select classifiers for it by hand which leads to inferior results to model-based approach as clear from Table 3.

4.3 The Effect of Number of Stages on the Cascade Performance

In section 3 we presented an algorithm that builds the best cascade of specific length. Besides being useful for memory-limited applications, it is very helpful in studying the effect of increasing the number of cascade stages on its performance. Figure 7 shows the complexities of best cascades with i stages, where i = 2, 3, 4, …, 48. It is clear from the figure that the complexity decreases as we add more stages until certain limit, after which the complexity starts to increase. Note that the complexities are calculated using the test set.

It would be interesting if we compared the theoretical cascade complexities anticipated by Equation (1) with the actual cascade complexities. Because the theoretical complexities were calculated using the validation set while building the cascade; hence, we will compare them to the actual complexities calculated using the validation set, not the test set. This is to clarify the difference between the theoretical and actual complexity setting aside the differences between the validation and test sets. Figure 8 compares the theoretical and actual complexities of best cascades with i stages, where i = 2, 3, 4, …, 48. From Figure 8, we note that the theoretical complexity decreases with adding more stages until we reach the cascade with 5 stages, then the complexity starts to increase again. On the other hand, the actual complexities continue to decrease with adding more stages till reaching the cascade with 22 stages. The actual complexity difference between best 5 stages and best 22 stages cascade is not substantial; however, it sheds the light on the concept of diversity discussed in section 4.2. In the point of view of Assumption 1, adding more stages after 5 stages limit does nothing but increasing the complexity; however, in reality, the diversity continues to enhance the performance and the cascade complexity continues to drop. We also note that with adding more and more stages, the difference between the theoretical and actual complexities increases since the effect of diversity increases between more classifiers.

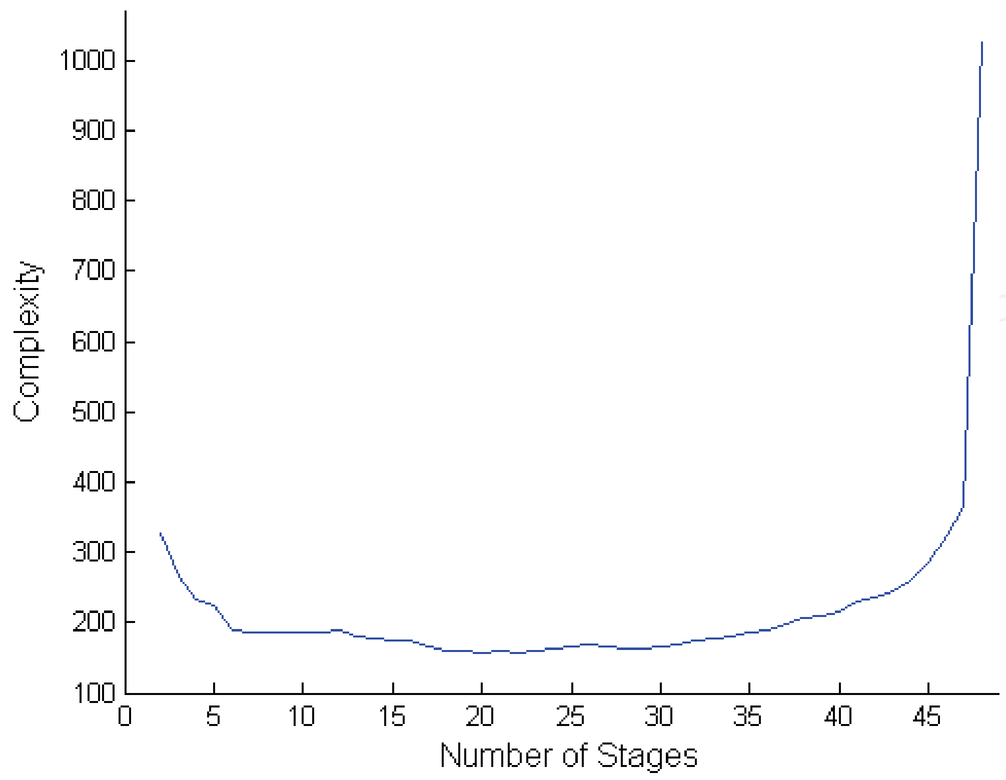


Fig. 7. The cascade actual complexity as the number of stages increases (calculated using the test set)

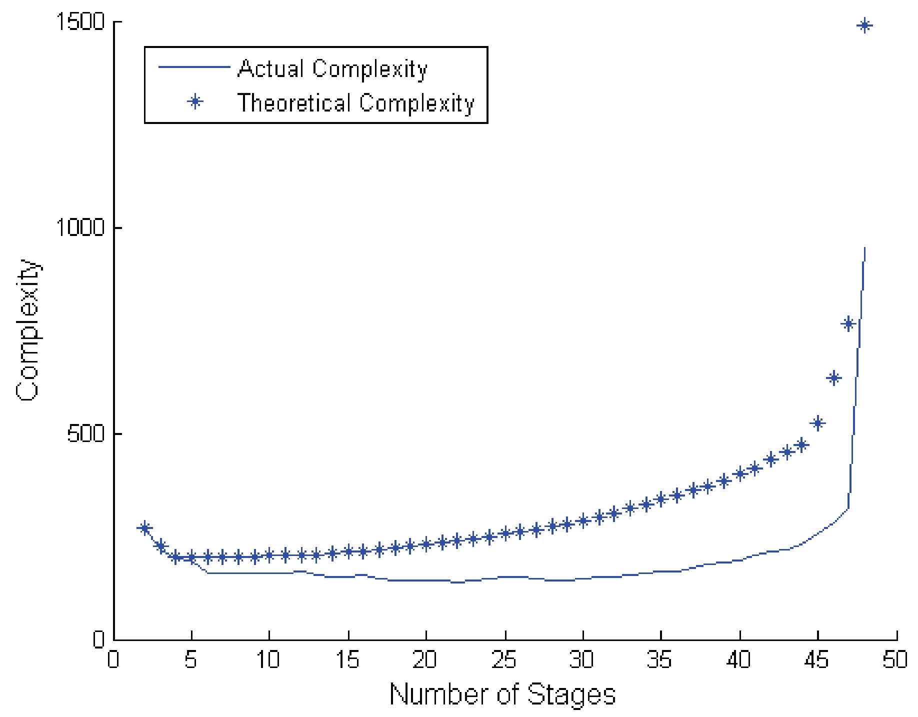


Fig. 8. The cascade actual as well as theoretical complexities as the number of stages increases (calculated using the validation set)

5. Related Works

In this section, we first present a taxonomy for the classification cascade research and then review some related works.

Classification cascades could be categorized according to four different aspects:

1- Accuracy versus speed oriented cascades.

One could build a cascade to increase the accuracy [RAHMAN & FAIRHURST 1999], or to increase the speed of the classification system [KAYNAK & ALPAYDIN 1997, PUDIL et al. 1992, GIUSTI ET AL. 2002, GORGEVIK & CAKMAKOV 2004, ,FERRI et al. 2004, Chellapilla et al. 2006a, Chellapilla et al. 2006b].

2- Reevaluation-based versus information-passing cascades.

In reevaluation-based cascades, the pattern to be classified is presented to the first classifier to give a decision with a confidence score. If the confidence score is higher than some threshold, the classification process terminates and the decision taken by the first classifier is declared to be the final decision. If the confidence score is lower than the threshold, the pattern is passed to the next classifier to re-classify it, and the process continues in the same manner. There is no information passed from one stage to the next. Each stage, if evoked, starts the classification process from scratch. In information passing cascades, each stage passes some information to the next stage. The most important of this category is the class reduction cascade, in which each stage passes a list of the most probable classes the pattern could belong to. Each stage focuses only on this list neglecting other classes [TSAY ET AL. 2004].

3- Dependent versus independent training of classifiers.

Each stage in the cascade could be trained independently using all the training set patterns [Chellapilla et al. 2006a, Chellapilla et al. 2006b This is called 'independent training of stages'. On the other hand, each stage could be trained using only the patterns rejected from the previous stage [FERRI ET AL. 2004]. This is called 'dependent training of stages'.

4- Manual versus automatic building of cascades.

A cascade could be manually built [GORGEVIK & CAKMAKOV 2004,], or automatically built [Chellapilla et al. 2006a, Chellapilla et al. 2006b]. The degree of cascade building automation differs. For example, in some cascade design technique, the structure of the cascade is automated but some other parameters (e.g. thresholds) are not.

According to this categorization scheme, the proposed model-based approach for building classification cascades is: speed-oriented, re-evaluation based, with independent training of classifiers, and entirely automatic.

There are many cascade design techniques in the literature. However, they have common themes. In the following some works on classification cascades will be presented, each representing some theme, mentioning similar works.

5.1 Risk analysis of multistage pattern recognition with reject option by Pudil et al.

One way to build a multistage system is to generate different systems with different structures and then to asses each of them using some criterion. Then the best cascade in terms of this criterion is selected. One problem with this method is that there are two conflicting requirements of a cascade system: high accuracy and low complexity. Any reasonable criterion should consider both requirements. Pudil et al. [PUDIL ET AL. 1992] suggested a criterion to assess the performance of multistage systems using a modified version of average risk analysis [DUDA et al. 2000].

Pudil's et al. technique is then considered according to the proposed categorization of cascades: speed-oriented, reevaluation-based, independent learning of stages, and automatic if we are ready to generate very large set of cascades and select the best; and partly-manual if we used our experience of the problem to select some reasonable set of cascades.

5.2 Kaynak-Alpaydin cascade

Kanyak and Alpaydin [KAYNAK & ALPAYDIN 1997] suggested a technique for building classification cascades that achieves high accuracy with low complexity. In this technique, a sequence of learners S_j's is used, where S_{j+1} learner is more complex than S_j. Associated with each learner is a confidence score $conf_j$ such that we say S_j is confident of its output and can be used if $conf_j > t_j$ where $0 < t_j \le t_{j+1} < 1$ is the confidence threshold. Learner S_j is used if all the preceding learners are not confident. Starting with $j = 1$, given a training set, S_j is trained. All the patterns on which S_j's performance is not acceptable are found and used to train S_{j+1}. This means that Kaynak-Alpaydin cascade falls in the category of dependent-training cascades.

This technique is to some degree similar to AdaBoost learning [DUDA et al. 2000, CHEVA (2004)]. Both techniques build a sequence of classifiers, each specializes in recognizing the pattern not recognized (or not confidently recognized) by the previous stage. However, there are some important differences between the two techniques. In AdaBoost classification, all the stages should be evoked in order to get the final classification decision. In Kaynak-Alpaydin cascade, the decision could be made at any stage according to the decision confidence of that stage. This property is behind the low complexity of the cascade classifier. AdaBoost uses weak learners of the same type. Kaynak-Alpaydin cascade uses different learner of increasing powerfulness and complexity.

Experiments show good performance of Kaynak-Alpaydin cascade. However, Kaynak-Alpaydin technique is not fully automatic, and relies of the users' experience to select the classifiers constituting the cascade as well as their rejection thresholds by hand.

Kaynak-Alpaydin cascade is considered according to the proposed categorization of classification cascades: speed-oriented, reevaluation-based, dependent training of stages, and manual.

5.3 Delegating classifier

Delegating classifier is another name coined by Ferri et al. [FERRI ET AL. 2004] for cascade classifier. Ferri et al. first suggest a two-stage system in which the first stage has a threshold at its output to reject the uncertain classifications to the second stage. This threshold is found such that the first stage would reject a certain percentage of the examples to the second stage. Here the first stage is trained using all the available examples and the second stage is trained using only the samples rejected by the first stage. This idea is also generalized to the case of more than two stages.

Ferri et al. suggest an interesting modification to the two-stage system. They put another threshold on the second stage output of the two-stage system. If the confidence score of the second stage falls below this threshold, the decision of the second stage is ignored and the final decision would be of the first stage. This approach is verified by the fact that the second stage inclines to overfit as it is trained using the noisy patterns rejected by the first stage. This technique was called 'Round Rebound' and was shown to slightly improve the results of the two-stage system.

Delegating classifier is considered according to the proposed categorization of classification cascades: speed-oriented, reevaluation-based, dependent training of stages. Ferri et al. suggested an automatic way of building cascade, though it is not theoretically verified and needs some manual calibration.

5.4 Two-stage system of Giusti et al.

Kaynak et al. [KAYNAK & ALPAYDIN 1997] studied one implementation of Kaynak-Alpaydin Cascade in which there is only two stages: the first stage is a global classifier like ANN, and the second stage is a local classifier like KNN. Giusti et al. [GIUSTI ET AL. 2002] studied a similar system theoretically with the addition to one time-saving technique. That is, if the first stage rejects some patterns, it indicates the h top most probable classes that the pattern belongs to. The KNN does not need then to search in its whole database, only within patterns belonging to the h top classes.

Giusti's two-stage system is considered according to the proposed categorization of classification cascades: speed-oriented, information-passing-based, dependent training of stages, and manual. Similar works to Giusti's system are [TSAY ET AL. 2004, GORGEVIK & CAKMAKOV 2004,].

5.5 Sequential combination of classifiers by Rahman and Fairhurst

'Sequential classifier' is another name for cascade classifier. Rahman and Fairhurst [RAHMAN & FAIRHURST 1999] presented two versions of the cascade classifiers: one is reevaluation-based and the other is information-passing-based. The information-passing-version passes a subset of most probable classes from one stage to the next narrowing down the scope of classes we search in. The first version resembles Kaynak-Alpaydin cascade but the stages are trained independently. The second version resembles the work of Giusti et al. but the role of successive stages is only to narrow down the list of possible classes more and more; an intermediate stage cannot classify a pattern; just the last stage can.

The major difference between Rahman and Fairhurst's cascade and other cascades is that it is accuracy-oriented. However, it is remarked that it has much less complexity than other accuracy-oriented classifiers combination scheme. Also, while they optimized the cascade accuracy, they could optimize its speed as well or they could optimize a cost function that considers both accuracy and speed.

It is understood how could a cascade enhance the speed; but how could it enhance the accuracy? The answer is different for each of the two versions of Rahman and Fairhurst's cascade. For the reevaluation version, the cause is as follows. If all stage before the last rejects or misclassify the patterns of the last stage, there will be no gain in accuracy. But actually what happens is that some stages correctly and confidently classify some patterns that are not correctly classified by the last stage (the concept of diversity discussed earlier). This is why the accuracy increases. For the information-passing version of the cascade, the cause behind the increase in accuracy is as follows. It happens that the last stage confuses between the true class of some pattern and other class. If this other class has been omitted from the list of considered classes passed through the cascade, this will lead the last stage make the correct classification as the rival class is omitted beforehand. This could increase the overall accuracy of the system.

Rahman and Fairhurst's cascade is considered according to the proposed categorization of classification cascades: accuracy-oriented but can easily modified to speed-oriented, reevaluation-based for the first version and information-passing-based for the second version, independent training of stages, and manual.

5.6 Searching in the space of thresholds by Chellapilla et al.

The most elegant work on classification cascade design is that of Chellapilla et al. [Chellapilla et al. 2006a, Chellapilla et al. 2006b]. They first presented a framework for the cascade design problem as an optimization problem that can be solved using any combinatorial optimization technique. Their cascade is speed-oriented, reevaluation-based, with independent training of stages, and is automatic to a large extent.

They start with a cascade of N classifier S_1, S_2, . . ., S_N; each has a complexity C_i and a threshold t_i, i=1, 2, . . ., N. The stages are ordered in the cascade in an ascending order of complexities (i.e. $C_1 < C_2 < \ldots < C_N$). The pattern to be classified goes initially through the first stage. If it is classified with confidence score higher than t_1, then it is absorbed (i.e. the classification process terminates taking the decision of S_1 to be the final decision). If the confidence score is below t_1, the pattern is rejected to the next stage S_2, and the process continues. The last stage has a threshold t_N=0 (i.e. it absorbs all the patterns it receives and rejects nothing).

The problem of cascade design now reduces to the setting of the set of N thresholds t_1, t_2, . . , t_N. Note that a stage could be excluded from the cascade by setting its threshold to 1 (i.e. it rejects everything). The problem is then formulated into an optimization problem. There are actually two optimization problems reflecting the goal from building the cascade. The first goal is to minimize the overall system complexity given some error constraint. The second goal is to minimize the error given some complexity constraint. The search space of solutions is then $V=\{t_1\}\times\{t_2\}\times \ldots \times\{t_N\}$, where $\{t_i\}$ is the set of all thresholds of stage *i*. The goal is then to find the optimal threshold vector $T^* = [t_1^*, t_2^*, \ldots, t_N^*]$ that solves one of the following two optimization problems,

i) minimizing the complexity,

$$T^* = \arg\min\{C(T) \mid T \in V, e(T) \le e_{\max}\} \qquad (3)$$

or *ii*) minimizing the error,

$$T^* = \arg\min\{e(T) \mid T \in V, C(T) \le C_{\max}\} \qquad (4)$$

where $C(T)$ is the complexity of the cascade with threshold vector $T=[t_1, t_2, \ldots, t_N]$, $e(T)$ is the error rate of the cascade with threshold vector T, e_{max} is the error constraint, and C_{max} is the complexity constraint.

Left is the procedure by which the set of possible threshold $\{t_i\}$ for the stage S_i for each *i*, i=1, 2, …, N is prepared. First, each stage S_i is used to classify all the examples of a validation set. The examples are then sorted in a descending order according to the confidence scores they are given by S_i. The examples are partitioned into Q-2 subset. The thresholds $\{t_i\}$ are then the confidence score of the first example of each subset, plus the two thresholds: 0 (means S_i absorbs all the examples) and 1 (means S_i rejects all the examples). Here then we have Q thresholds in the set $\{t_i\}$. This is equivalent to quantizing t_i to Q quantization levels. Then the size of the space of thresholds V is Q^N. The optimization problem is then to search through the space V of threshold to satisfy either Equation (3) or Equation (4).

This problem can be solved using any combinatorial optimization technique. Cellapilla et al. tried solving the problem using steepest descent, dynamic programming, simulated annealing, depth first search (DFS). All these algorithms are suboptimal except DFS. The DFS [Chellapilla et al. 2006b] is a simple search algorithm that searches through the space of solution intelligently. It prunes large sections of the search space that are guaranteed not to give the best solution.

This framework is elegant and fully automatic except that the procedure of ordering the stages by increasing complexity is not verified theoretically. The DFS solution is elegant and optimal but it has an exponential complexity in N (that is, $O(Q^N)$) which means that using large value number of stages is computationally prohibitive. This made Chellapilla et al. do manual selection of the N classifiers to be used with algorithm. Hence, though DFS could be fully automatic, its high computational complexity hinders it to be.

6. Conclusion

In this chapter, we presented a model-based approach for automatically building classification cascades. The experiments showed that the algorithm is efficient and scalable. The algorithm was also analyzed and its strengths and limitations were clarified. In addition, we presented an algorithm that builds cascades with given lengths which is useful in memory-limited systems helped in studying the effect of increasing the number of stages in a cascade on its performance.

7. References

Brubaker, S., Mullin, M., and Rehg J., (2006), “Towards optimal training of cascaded detectors,” ECCV06, vol. 1, pp. 325-337, Graz, Austria, May.

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).

Chen, X. & Yuille, A., (2005), “A time-efficient cascade for real-time object detection: with application for the visually impaired,” IEEE CVPR-05, vol. 3, pp. 28, San Diego, CA, USA, June 20-25.

Duda, R., Hart, P., Stork, D., (2000), Pattern Classification, 2nd Edition, Wiley, New York.

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.

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

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

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.

Luo, H. , (2005), “Optimization design of cascades classifiers,” IEEE CVPR-05, vol. 1, pp. 480- 485, San Diego, CA, USA, June 20-25.

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.

Sun, J., Regh, J., Bobick, A. ,(2004), “Automatic cascade training with perturbation bias,” IEEE CVPR-04, vol. 2, pp. 276-283, Washington, DC, June 27 – July 2.

Tsay, J., Lin, C. ,Hung, C. ,and Lin, C. , 2004 ,“Cascaded class reduction for time-efficient multi-class classification,” 18th Annual ACM International Conference on Supercomputing (ICS’04), pp. 189-194, Saint-Malo, France, June 26-July 1.

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

Webb, (2002), Statistical pattern recognition, 2nd Edition, Wiley.

Wu, J., Brubaker, S., Mullin, M., and Regh, J. ,(2008), "Fast asymmetric learning for cascade face detection," IEEE PAMI, vol. 30, no. 3, pp. 369-382.

Yuann, Q. Thangali, A. . Sclaroff, S,(2005) ,"Face identification by a cascade of rejection classifiers," IEEE CVPR-05, vol. 3, p. 152, San Diego, CA, USA, June 20-25.