Improving Cascading Classifiers with Particle Swarm Optimization

Luiz S. Oliveira, Alceu S. Britto Jr., and Robert Sabourin
Pontifícia Universidade Católica do Paraná, Curitiba, Brazil
Ecole de Technologie Supérieure - Montreal, Canada
soares@ppgia.pucpr.br

Abstract

This paper addresses the issue of class-related reject thresholds for cascading classifier systems. It has been demonstrated in the literature that class-related reject thresholds provide an error-reject trade-off better than a single global threshold. In this work we argue that the error-reject trade-off yielded by class-related reject thresholds can be further improved if a proper algorithm is used to find the thresholds. In light of this, we propose using a recently developed optimization algorithm called Particle Swarm Optimization. It has been proved to be very effective in solving real valued global optimization problems. In order to show the benefits of such an algorithm, we have applied it to optimize the thresholds of a cascading classifier system devoted to recognize handwritten digits.

Keywords: Decision Thresholds, Cascading Classifiers, Particle Swarm Optimization, Handwriting Recognition.

1 Introduction

Cascading classifiers have been quite used to solve pattern recognition problems in the last years. The main motivations behind such a strategy are the improvement of classification accuracy and the reduction of the complexity. The latter is motivated by the observation that in many cases, the majority of patterns can be explained by a simple rule, i.e., they are well-behaved. Therefore, they can be classified using a relatively small portion of the available features while just for a few hard cases more sophisticated classifiers (using more features, therefore more expensive) are needed.

In a cascading classifier the inputs rejected by the first stage are handled by the next ones using costlier features or classifiers. As stated by Pudil et al in [9], the whole idea is to use some more informative (but at the same time more costly) measurements by adding them to the set of less informative (but less costly) measurements used in the previous stage. Figure 1 depicts this architecture.

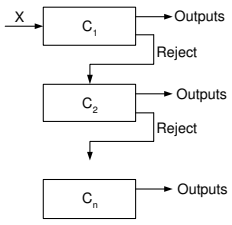

Figure 1. The cascading classifier architecture.

It is clear from Figure 1 that to implement such a strategy we must use the so-called reject option. In this case, the patterns that are the most likely to be misclassified are rejected (i.e., they are not classified) and then handled by more sophisticated procedures. In cascading classifiers a trade-off between error and reject is mandatory, because this kind of system can bring a reduction of complexity only when most of the samples are correctly classified by the earlier levels.

One of the most used error-reject trade-off was given by Chow [1]. In this rule, a pattern x is rejected if

$$\max_{k=1,\ldots,N} P(\omega_k|x) = P(\omega_i|x) < T, \qquad (1)$$

where $T \in [0,1]$. On the other hand, the pattern x is accept and assigned to the class ω_i, if

$$\max_{k=1,\ldots,N} P(\omega_k|x) = P(\omega_i|x) \geq T. \qquad (2)$$

Fumera et al [3] point out that Chow's rule provides the optimal error-reject trade-off, only if the posteriori probabilities are exactly known, which does not happen in real applications since they are affected by significant

estimate errors. In order to overcome such a problem, Fumera et al have proposed the use of multiple reject thresholds for the different data classes to obtain the optimal decision and reject regions, even if the posteriori probabilities are affected by errors. Thus, the rule for a classification task with N data classes that are characterized by estimated a posteriori probabilities $\widehat{P}(\omega_i|x), i = 1, \ldots, N$. A pattern x is rejected if

$$\max_{k=1,\ldots,N} \widehat{P}(\omega_k|x) = \widehat{P}(\omega_i|x) < T_i, \tag{3}$$

while it is accepted and assigned to the class ω_i if

$$\max_{k=1,\ldots,N} \widehat{P}(\omega_k|x) = \widehat{P}(\omega_i|x) \geq T_i, \tag{4}$$

Under the hypothesis that the a posteriori probabilities are affected by significant errors, it has been demonstrated that, for any reject rate R, such values of the thresholds $T_1, \ldots, T_N$ exist that the corresponding classifiers accuracy $A(T_1, \ldots, T_N)$ is equal or higher than the accuracy $A(T)$ provided by Chow's rule. For real application, the problem consists in finding the N thresholds that maximize the accuracy for a fixed error rate ϵ. This can be formulated in terms of a constrained maximization problem as follows:

$$\begin{cases} \text{maximize} & A(T_1, \ldots, T_N) \\ \text{subject to} & \text{Error Rate} \leq \epsilon \end{cases} \tag{5}$$

The algorithm proposed by Fumera et al to solve this problem takes into account the hypothesis that the number of rejected patterns cannot decrease for increasing values of the thresholds. They argue that this hypothesis holds most of the time, but we have realized through experimentation that in several cases it is not true. Hence, a more appropriate algorithm should be used to solve Equation 5.

In this paper we propose Particle Swarm Optimization (PSO) to determine class-related reject thresholds. PSO is a population based stochastic optimization technique developed by Eberhart and Kennedy in 1995 [4], inspired by social behavior of bird flocking or fish schooling. It shares many similarities with evolutionary computation techniques such as genetic algorithms (GAs), but unlike GAs, PSO has no evolution operators such as crossover and mutation. Differently from GAs, which were conceived to deal with binary coding, PSO was designed and proved to be very effective in solving real valued global optimization problems, which makes it suitable for our study. The detailed information will be given in Section 2.

In order to show the improvements that this kind of algorithm can yield, we have applied it to determine the reject thresholds for a cascading classifier system so that we can reduce as much as possible the complexity of the classification task. Comprehensive experiments demonstrate the efficiency of this algorithm.

The remaining of this paper is organized as follows. Section 2 introduces the basics of PSO. Section 3 describes the baseline cascading system we have adopted in our experiments. Section 4 reports our experimental results while Section 5 concludes this work.

2 Particle Swarm Optimization

As stated before PSO simulates the behaviors of bird flocking or fish schooling. To better understand how PSO works, consider the following scenario: a group of birds are randomly searching food in an area. There is only one piece of food in the area being searched. All the birds do not know where the food is, but they do know how far the food is in each iteration. An effective strategy to find the food is to follow the bird which is nearest to the food.

PSO learned from the scenario and used it to solve optimization problems. In PSO, each single solution is a bird in the search space. Such solutions are called *particles*. All particles have fitness values which are evaluated by the fitness function to be optimized, and have velocities which direct the flying of the particles. They fly through the problem space by following the current optimum particles.

Suppose that the search space is D-dimensional, then the i-th particle of the population, which is called *swarm*, can be represented by a D-dimensional vector $S = (s_1, s_2, \ldots, s_D)$. The velocity of this particle, can be represented by another D-dimensional vector $V = (v_1, v_2, \ldots, v_D)$. The best previously visited position of the i-th particle is denoted as $P = (p_1, p_2, \ldots, p_D)$. Let g be the index of the best particle in the swarm and let the superscripts denote the iteration number, then the swarm is manipulated according to the following two equations:

$$v_i^{k+1} = w v_i^k + c_1 r_1^k (p_i^k - s_i^k) + c_2 r_2^k (p_g^k - s_i^k) \tag{6}$$

$$s_i^{k+1} = s_i^k + v_i^{k+1} \tag{7}$$

where w is the inertia weight; c_1 and c_2 are two positive constants called cognitive and social parameters respectively, also known as learning factors; r_1 and r_2 are random numbers uniformly distributed in [0,1]; $i = 1, 2, \ldots, N$ and N is the size of the swarm, and $k = 1, 2, \ldots$ is the current iteration. Figure 2 depicts the concept of modification of a searching point by PSO in a 2-dimensional space.

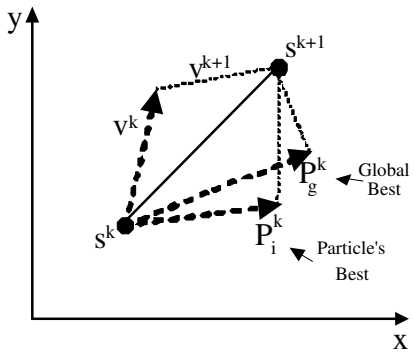

Figure 2. Concept of modification of a searching point by PSO in a 2-dimensional space.

The role of the inertia weight in Equation 6 is to regulate the trade-off between exploration and exploitation. A large weight facilitates global search (exploration) while a small one tends to facilitate fine-tuning the current search area (exploitation). Proper fine-tuning of the parameters c_1 and c_2 may result in faster convergence of the algorithm and alleviation of the local minima. As default values, $c_1 = c_2 = 2$ were proposed. The parameters r_1 and r_2 are used to maintain the diversity of the population, and they are uniformly distributed in the range [0,1]. The pseudo-code of the procedure is as follows

```
FOR each particle
  Initialize particle
END

DO
  FOR each particle
    Compute fitness value
    If fitness value is better than the
    best fitness (pBest) in history
      Set current value as the new pBest
  END

  Choose the particle with the best fitness
  as the gBest

  FOR each particle
    Compute its velocity
    Update its position
  END
WHILE maximum iterations or stop criteria are
not attained.
```

The aforementioned procedure is based on the sociometric principle called *gbest*, which conceptually connects all the members of the swarm to one another. In such a case each particle is influenced by the very best performance of any member of the entire population.

Although PSO is an evolutionary technique, it differs form other evolutionary algorithm (EA) techniques. Three main operators are usually involved in EA techniques: recombination, mutation, and selection. PSO does not have a direct recombination operator. Nevertheless, the stochastic acceleration of a particle towards its previous best position, as well as towards the best particle of the swarm, resembles the recombination procedure in EA [2]. In PSO the information exchange takes place only among the particle's own experience and the experience of the best particle in the swarm, instead of being carried from fitness dependent selected parents to descendants as in genetic algorithms. Besides, the directional position updating used in PSO is similar to mutation of GA, with a kind of memory built in.

Finally, PSO belongs to the class of EAs that does not use the "survival of the fittest" concept. It does not utilize a direct selection function, therefore, particles with lower fitness can survive during the optimization and potentially visit any point of the search space [2].

3 Cascading Classifier System

In order to demonstrate the importance of the decision thresholds we have used as baseline system the cascading classifier presented in [7]. It uses an optimized Hill-Climbing algorithm to select subsets of features for handwritten character recognition. The search is conducted taking into account a random mutation strategy and the initial relevance of each feature in the recognition process. This method takes as input a base classifier trained with a given feature set and then searches for the best subset of features that yields similar or better performance.

The base classifier is a neural network (MLP) trained with the gradient descent applied to a sum-of-squares error function, where the activation function is a softmax. The feature set contains 132 measures of concavities and contour [8]. The database in this case is the NIST SD19 and it was composed of 195,000, 28,000, and 30,089 images of handwritten digits for training, validation, and testing, respectively. The performance at zero-rejection level of the base classifier on the testing set is 99.13%.

In order to generate a set of classifiers using different parts of the feature set, the authors introduced the concept of error tolerance. The idea is to allow some loss of performance, hence, producing less expensive

classifiers. In their experiments the maximum error allowed was 2% and the number of classifiers generated was 5. Table 1 describes the classifiers that compose the cascading system.

Table 1. The performance of the classifiers in the testing set

Classifier	Features	Accuracy (%)
C_1	25	97.43
C_2	33	97.85
C_3	52	98.55
C_4	59	98.72
C_5	77	98.94

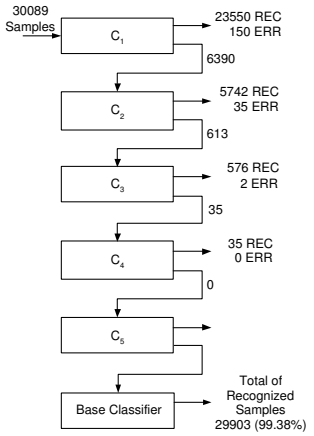

Figure 3. The performance of the baseline cascading classifier system.

Figure 3 illustrates the cascading system proposed in [7]. From this figure, we can observe for each level of the cascade the number of samples recognized, misclassified, and rejected for the next level. The reject thresholds were determined on the validation set using Fumera's algorithm for $\epsilon = 0.5\%$. The base classifier has no reject option. As we can observe, the complexity has dropped considerably since the cascading system do not invoke the last two classifiers (the most expensive ones) of the cascade.

In order to access the required computational effort of the cascading system we have used the total number of feature-values [6] which is given by

$$TVF = \sum_{i=1}^{n} m_i x_i \tag{8}$$

where n is the number of classifiers in the cascade, m_i is the number of features used by classifier i, and x_i is the number of instances classified by classifier i. The number of feature-values accessed by the base classifier is $M \times X$, where M is the total number of features (132 in this case) and X is the number of samples correctly classified by the base classifier.

By computing TVF for the data described in Figure 3 we can observe that the cascading system reduces in about 75% the number of feature-values compared to the base classifier, which is quite impressive. In addition, the cascading system surpassed the final recognition rate of the base classifier in about 0.2%. It is worth of remark that this improvement is not significant, statistically speaking, but the point is that the cascading system is much less complex than the base classifier while maintaining the same performance. In the next section we demonstrate that such a system can be further improved by using PSO to define the reject thresholds.

4 Optimizing the Cascading System with PSO

The conventional PSO algorithm described previously was used in our experiments. The only modification we have done was to use the constraint proposed in [5], which imposes a maximum velocity to the particle in order to prevent the swarm from exploding. Thus, if $v_i^{k+1} > V_{max}$ in Equation 6, then $v_i^{k+1} = V_{max}$.

The particle (individual) has 10-dimensional space to encode the N thresholds defined in Equation 3. Therefore, each axis has values ranging from 0 to 1. As mentioned above, each particle has a position and a velocity, which have their values randomized initially. In order to speed up the optimization process, one particle of the swarm was initialized with the threshold produced by Chow's rule. Thereafter, the entire dynamics is governed by Equation 6.

The size of the swarm used in our experiments is 20. The typical range for the number of particles ranges from 20 to 40. The learning factors c_1 and c_2 were set to 2. The maximum number of iteration ($iter_{max}$) was set to 2,000 and the inertia weight w as initially set to 0.9 to allow a global search. In order reduce this weight over the iterations, allowing the algorithm to exploit some specific areas, we have used the following equation

$$w = w_{max} - \frac{w_{max} - w_{min}}{iter_{max}} \times iter \tag{9}$$

where $w_{max} = 0.9$, $w_{min} = 0.01$, and $iter$ is the current iteration number.

Once all parameters have been defined, the PSO was used to solve the problem stated in Equation 5. In other words, maximize the accuracy of the classifier constrained to a given ϵ. In order to be able to compare our results to the cascading classifier introduced in the previous section, we have used $\epsilon = 0.5\%$. Since PSO is a stochastic optimization technique, we have run the algorithm several times. However, the best solution was found usually in the first run of the algorithm and before the iteration 500.

After optimizing the reject thresholds for each classifier reported in Table 1 independently, using for that the validation set described in Section 3, we have recalculated the results presented in Figure 3. Figure 4 depicts the results achieved by the cascading system optimized with PSO. After optimization, the system was able to classify correctly 82.4% of the samples at the first level of the cascade, against 78.2% of the baseline system. This means more reduction in terms of complexity. Similarly to the baseline system, the optimized one also brought a slight improvement to the final performance.

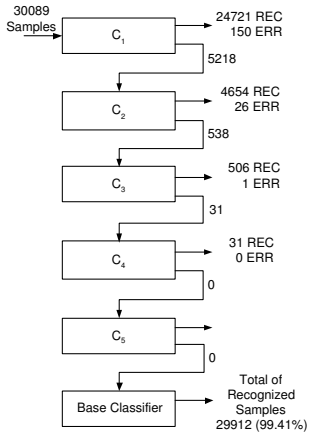

Figure 4. The performance of the cascading classifier system optimized with PSO.

As we can see, the cascading classifier system is a very interesting approach to reduce complexity while keeping performance at the same levels. However, it can perform even better when the reject option is properly fine-tuned.

5 Conclusion and Future Works

In this work we have proposed Particle Swarm Optimization to search for optimum class-related reject thresholds. To demonstrate the feasibility of the proposed approach, we have tested it on a cascading classifier system, which uses 75% less feature-values than the base classifier. Our experiments confirms that a proper optimization algorithm still can bring some benefits to well-known strategies like cascading classifiers.

In spite of the fact we have used a cascade classifier to demonstrated the feasibility of PSO to determine class-reject thresholds, this algorithm can also be applied when just one classifier is available as well.

For future works we plan a deeper optimization, addressing parameters such as the number of classifiers in the cascade and a reject option in the base classifier.

Acknowledgements

This research has been supported by The National Council for Scientific and Technological Development (CNPq) grant 150542/2003-8.

References

[1] C. K. Chow. On optimum error and reject tradeoff. *IEEE Trans. on Information Theory*, 16:41–46, 1970.

[2] R. C. Eberhart and Y. Shi. Comparison between genetic algorihms and particle swarm intelligence. In V. W. P. et al, editor, *Evolutionary Programming VII*, pages 611–616. Springer, 1998.

[3] G. Fumera, F. Roli, and G. Giacinto. Reject option with multiple thresholds. *Pattern Recognition*, 33(12):2099–2101, 2000.

[4] J. Kennedy and R. C. Eberhart. Particle swarm intelligence. In *Procs of the International Conference on Neural Network*, pages 1942–1948, 1995.

[5] J. Kennedy and R. C. Eberhart. *Swarm Intelligence*. Morgan Kaufmann, 2001.

[6] M. Last, H. Bunke, and A. Kandel. A feature-based serial approach to classifier combination. *Pattern Analysis and Aplications*, 5:385–398, 2002.

[7] C. M. Nunes, A. S. Britto, C. Kaestner, and R. Sabourin. An optimized hill climbing algorithm for feature subset selection: Evaluation on handwritten character recognition. In *Procs of the 9^{th} International Workshop on Frontiers of Handwriting Recognition*, pages 365–370, 2004.

[8] L. S. Oliveira, R. Sabourin, F. Bortolozzi, and C. Y. Suen. Automatic recognition of handwritten numerical strings: A recognition and verification strategy. *IEEE Trans. on PAMI*, 24(11):1438–1454, 2002.

[9] P. Pudil, J. Novovicova, S. Blaha, and J. Kittler. Multistage pattern recognition with reject option. In *Procs of the 11^{th} International Conference on Pattern Recognition*, pages 92–95, 1992.