An Optimized Hill Climbing Algorithm for Feature Subset Selection: Evaluation on Handwritten Character Recognition

Carlos M. Nunes[1], Alceu de S. Britto Jr.[1,2], Celso A. A. Kaestner[1] and Robert Sabourin[3]

[1] *Pontifícia Universidade Católica do Paraná (PUC-PR)*
R. Imaculada Conceição, 1155 - Curitiba (PR) 80215-901 – Brazil
{carlosmn, alceu, kaestner}@ppgia.pucpr.br

[2] *Universidade Estadual de Ponta Grossa (UEPG)*
Pr. Santos Andrade S/N, Centro - Ponta Grossa (PR) 84100-000 – Brazil

[3] *École de Technologie Supérieure (ETS)*
1100 Rue Notre Dame Ouest - Montreal (QC) H3C 1K3 - Canada
robert.sabourin@etsmtl.ca

Abstract

This paper presents an optimized Hill-Climbing algorithm to select subset 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. A first set of experiments have shown a reduction in the original number of features used in an MLP-based character recognizer from 132 to 77 features (reduction of 42%) without a significant loss in terms of recognition rates, which are 99.1% for 30,089 digits and 93.0% for 11,941 uppercase characters, both handwritten samples from the NIST SD19 database. Additional experiments have been done by considering some loss in terms of recognition rate during the feature subset selection. A byproduct of these experiments is a cascade classifier based on feature subsets of different sizes, which is used to reduce the complexity of the classification task by 86.54% on the digit recognition experiment. The proposed feature selection method has shown to be an interesting strategy to implement a wrapper approach without the need of complex and expensive hardware architectures.

1. Introduction

Many feature subset selection algorithms [1,2,3] have been developed in the last years, since this procedure can reduce not only the cost of recognition by reducing the number of features that need to be used, but in some cases it can also provide better classification accuracy. Usually, the methods found in the literature [4,5] can be categorized as: Filter or Wrapper-based approach.

In both categories, given a set of candidate features, the objective is to select a subset that performs the best under some classification system. The main difference between them is that in the Filter-based approach the relevance of each feature is defined taking into account statistical information calculated from the training dataset, while in the Wrapper approach [6,7] the classifier is used to evaluate the relevance of each feature during the selection process.

An interesting wrapper-based method was proposed in [8,9] using a genetic algorithm for the subset selection. The authors have achieved a significant feature reduction (from 132 to 100 features), which means about 25% keeping the initial classification rate almost the same (99.16%) for handwritten digits available in the NIST SD19 database. However, their process must to evaluate 128,000 feature subset candidates. For this purpose, the authors use a cluster of 17 personal computers (with 1.1GHz and 512 Mb RAM each).

In order to provide a low-cost solution in terms of architecture needed, this paper presents an optimized Hill Climbing algorithm to select a subset of features for handwritten character recognition. The search is done taking into account a random mutation strategy and a priority associated to each feature by considering its relevance in the recognition accuracy. Different from the method described in [8], the proposed method has shown to be an interesting strategy to implement a wrapper approach, which can be executed in more simple hardware architecture.

Proceedings of the 9th Int'l Workshop on Frontiers in Handwriting Recognition (IWFHR-9 2004)
0-7695-2187-8/04 $20.00 © 2004 IEEE

2. Proposed Method

The proposed method has 5 stages, as shows in Figure 1. The first stage consists of a feature extraction process, which generates 5 files (*train*, val_1, val_2, val_3 and *test*) containing feature vectors extracted from character images. In the second stage, a MLP neural network is trained and evaluated considering the original configuration of the feature vectors and using *train* and val_1. In the third stage, an optimized Hill Climbing (OpHC) algorithm uses val_2 in order to select multiple candidates of feature subsets. A feature randomly selected is removed, or not, taking into account its relevance on the classification accuracy. In addition, a priority is associated to each feature to guide the search. The selected subset candidates (or solutions found in the search space) are finally evaluated using val_3 in the fourth stage. The objective is to select the candidate that provides the best recognition accuracy. Finally, the subset of features selected is used to retrain the neural network, which is adapted to the feature subset configuration.

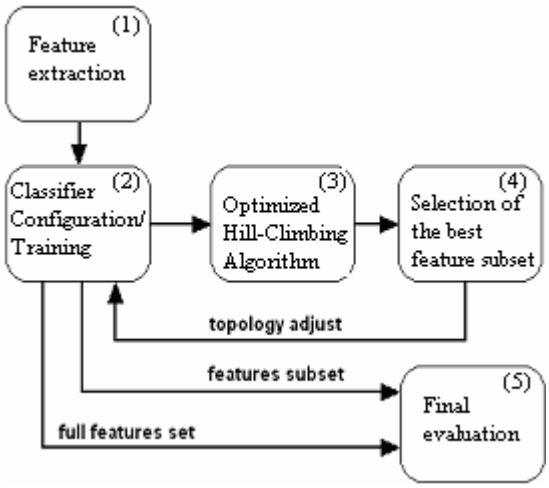

Figure 1. Overview of the proposed method

2.1. Feature Extraction

The same feature extraction method described in [8] is used. It divides a character image into 6 regions and calculates 132 features based on contour, concavity and surface information.

2.2. Classifier configuration/training

The classifier is an MLP neural network trained using the backpropagation algorithm. The initial topology consists of 132 nodes in the input layer, 100 nodes in the hidden layer, and the output layer contains 10 or 26 nodes for digit or uppercase character recognition, respectively.

2.3. Optimized Hill Climbing Algorithm (OpHC)

This module consists of a modified Hill Climbing algorithm. Figure 2 shows, in bold style, the main differences of the proposed algorithm from the Original Hill Climbing (OrHC).

1. **Establish priority for each feature;**
2. Load neural network previously trained;
3. If (Number of Iteration = MAXITER) then exit;
4. Select a feature, randomly;
5. **If (the priority of the selected feature = zero)**
   then remove it (e.g. replace it by it's average value) and update the current feature set mask;
   else increase the feature priority and goes to step 4;
6. Evaluate the classification accuracy with the new feature set configuration;
7. If (current error rate <= previous one) **or**
   (current error rate <= ERROR_TOL)
   then keep the current feature set configuration;
   else backtrack to the previous state;
8. If (number of removed features = TFEAT)
   then save current configuration as a local maximum; go to step 3;
   else go to step 4.

ERROR_TOL – error tolerance;
TFEAT – Total of features;
MAXITER – Maximum of iterations.

Figure 2. Optimized Hill Climbing Algorithm (OpHC)

The algorithm starts by defining a priority for each feature (step 1) available on the initial feature vector configuration (132 for this problem). Each feature, or seed, has its priority level calculated as shown in Figure 3. In the second step of the algorithm, the neural network trained using the entire feature set is loaded. In the kernel of the algorithm (step 4), a random process is used to select a feature to be removed. In case the priory related to this feature is zero (step 5), it will be removed, otherwise its priority will be increased and a different feature will be randomly selected. The step 6 provides the current error rate after removing the selected feature. A decision about to keep the current feature set configuration is taken on step 7. For this purpose, the current error rate is compared to the previous one. In addition, in the optimized version of the algorithm an error tolerance is taking into account. This parameter is experimentally evaluated in this paper (see Section 3.5).

As we can see, a local maximum is found after evaluating all the features in the current state (see step 8). After that, the OpHC algorithm returns to the initial state instead of backtracking to the previous one. Thus, the complete configuration of the feature set is considered again and a new feature or seed (not processed yet) is

selected. The objective is to investigate other areas of the search space. In addition, the priority computed for each feature provides a way of guiding the search taking into account the feature relevance, while the concept of randomized feature removal is maintained.

1. Calculate the error rate when individual features are removed (FERROR), see Figure 4;
2. Select the maximum (MAX) and minimum (MIN) FERROR over all features;
3. Select a number of levels (NLEVELS), which is the number of priority levels to be considered;
4. Calculate the range of each level using R = (MAX-MIN) / NLEVELS;
5. The relevance (priority) of each feature is calculated as P = (- NLEVELS + (MAX–FERROR) / R).

Figure 3. Scheme to calculate the feature priority

As described before, the feature priority takes values between 0 and -NLEVELS. According to Figure 3, the highest error values receive –NLEVEL and the lowest error values receive zero. Each time that a feature is removed, the algorithm evaluates its priority and in case of the priority is 0 (zero) the feature will be removed, otherwise the feature is not removed and the priority is updated (P = P+1). When priority reaches the value zero, the corresponding feature is removed.

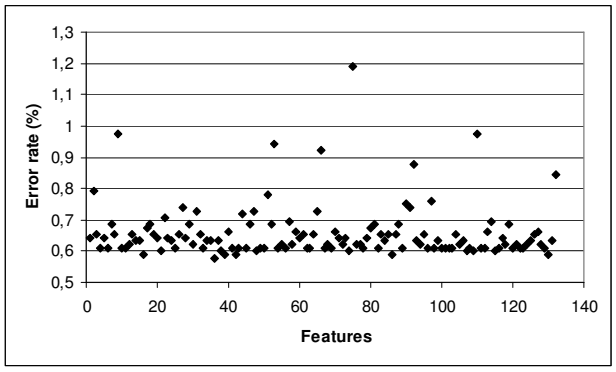

Figure 4. Error rate after individual removal

The number of levels (NLEVELS) was experimentally defined. After evaluating the values 0, 5, 10, 20 and 50, the best results were obtained by using 10 levels. Figure 5 shows that, after 120 iterations the algorithm had already removed 40 features from the original set.

Another important characteristic of the proposed algorithm is the use of sensitivity analysis [8], since to retrain the neural network at each new feature subset is not feasible due to the limits imposed by the learning time of the huge training set considered in this work. The sensitivity of the network to the removal of individual features is used to estimate the relationship between the input features and the network performance. So, in order to evaluate a given feature subset we replace the unselected features by their averages values evaluated on the training database. In this way, we avoid training the neural network and hence turn the wrapper approach feasible for our problem.

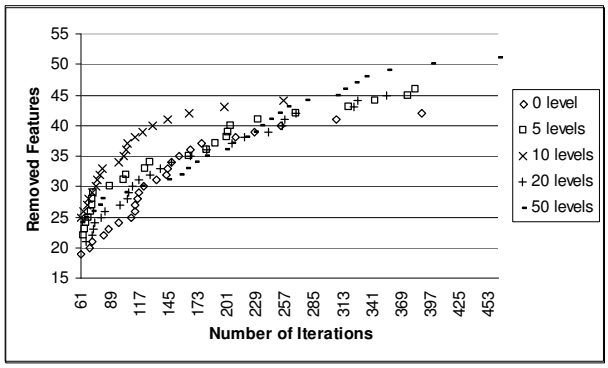

Figure 5. Selection of the number of priority levels

Each local maximum found by this module represents a feature subset candidate. The search for feature subsets is done on val_2.

2.4. Selection of the best feature subset candidate

The selection is done based on the maximum recognition accuracy and the minimum number of features. When two solutions show the same accuracy, this one with the minimum number of features is selected. A different validation dataset (val_3) is used to select one feature subset from those candidates provided by the OpHC algorithm.

2.5. Final Evaluation

In this module, the final feature subset selected in the last stage is used to retrain the neural network, whose topology is adapted to this new configuration of the feature set. A final evaluation is done using the *test* set.

3. Experimental Results

The experiments are based on handwritten digits and uppercase characters available in the NIST SD19 database. In the experiment based on digits the following protocol was used: 195000 samples for training (*train*) and 28008 for validation (9336 in val_1, 9336 in val_2 and 9336 samples in val_3) - all samples from hsf_0, 1, 2 and 3 series. Other 30089 samples were used for testing (*test*), which are available on hsf_7 series.

The data sets used in the experiments using uppercase characters are composed of 37440 samples for training (*train*) and 12092 for validation (4031 in val_1, 4031 in val_2 and 4030 in val_3) – all samples were taken from hsf_0,1,2 and 3 series. Other 11941 samples from hsf_4 were used for testing (*test*).

In the experimental protocol, val_1 is used during training of the neural network to avoid an over training. Val_2 is used during the search for feature subset candidates performed by the HC algorithm. Finally, val_3 is used to select the subset of features, which provides the best recognition accuracy among the candidates provided by the HC algorithm. The testing set is used as a black box just to compare the final recognition results of the classifier after his topology has been adapted to the new configuration of the feature set.

3.1. Experiments on Handwritten Digits

Both algorithms, the original and optimized HC were evaluated. The MAXITER variable was set to 16,000 iterations and the ERROR-TOL was set to zero. At the end of the iterations the original HC has used only 3 seeds, since it does not returns to the initial configuration of the feature set (initial state) after finding a local maximum or after removing all features in the current state, but it returns to the previous state. From this three seeds or starting points, the original algorithm found 172 local maximum (subset candidates). By contrast, the modified algorithm based on the priority scheme has investigated all possible seeds (132) using the same 16,000 iterations and it has generated 47 feature subset candidates. In fact, the proposed algorithm has stopped after all features have been used as seeds. This means that all features were removed during the search and a bigger diversity on evaluated solutions was reached.

In Figure 6, each point represents a feature subset candidate of the 172 found by using the Original Hill Climbing (OrHC). As we can observe, all points are very concentrated in a small area of the search space. By contrast, in Figure 7, the 47 feature subset candidates found by using the modified HC are not concentrated as those provided by the original algorithm.

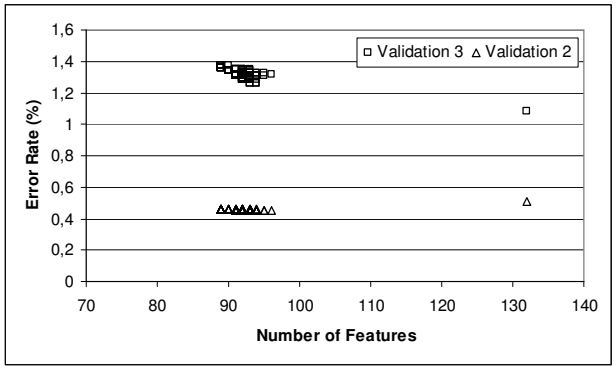

Figure 6. Feature subset candidates provided by the traditional HC algorithm

The reason is that in the modified algorithm each time a local maximum is found, the initial configuration of the feature set is returned in order to select a new seed. Moreover, the strategy of using the priority scheme has provided to the algorithm a faster convergence to a local maximum.

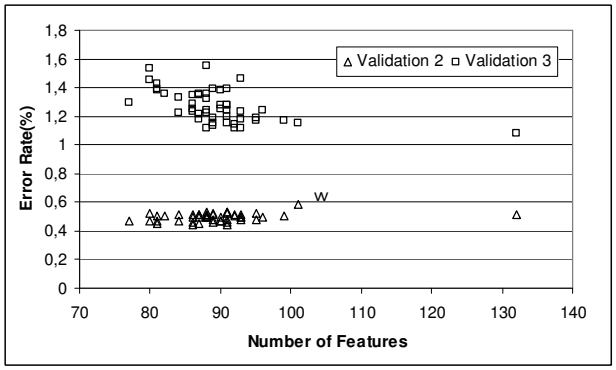

Figure 7. Feature subset candidates provided by the optimized HC algorithm

In a first set of experiments, the val_3 dataset is used to select the best solution (feature subset) from the 172 provided by the original HC algorithm (OrHC). The same strategy is used to select the best solution from the 47 provided by the optimized algorithm (OpHC). In addition, we also experiment to consider just the number of features removed, since the classifier has shown a small loss in terms of accuracy. This means that, in this experiment, the best configuration corresponds to the smallest subset of features.

Table 1. Experimental results on digits

Experiment	Features	Training	Testing
Entire feature set	132	99.77%	99.10%
OrHC (using val_3)	92	99.50%	99.04%
OpHC (using val_3)	87	99.95%	98.95%
OrHC (smallest feature set)	81	99.50%	98.92%
OpHC (smallest feature set)	77	99.86%	98.94%

As we can observe in Table 1, there is no significant loss in terms of classification accuracy. However, it is possible to observe a significant reduction of features when the optimized HC algorithm was used.

In order to compare our results with those obtained by Soares [8], we have used the same experimental protocol. The proposed wrapper approach has shown a more significant reduction (42%) than that observed by using GA (25%). Moreover, while the wrapper-based approach proposed by Soares [9] needs to evaluate around 128,000 solutions, the proposed method evaluates just 16,000 solutions.

3.4. Experiments on Handwritten Characters

The method was also applied to handwritten uppercase characters available in the NIST SD19 database. During the experiments, just the optimized HC was used. However, the experiments considers to select the final feature set configuration taking into account the recognition rate on val_3 dataset, and also it considers the more significant reduction in the number of features.

Table 2. Experimental results on uppercase characters

Experiment	Features	Training	Testing
Feature set	132	97.23%	93.05%
OpHC (validation3)	101	97.57%	92.51%
OpHC (smallest feature set)	79	97.68%	92.50%

The reduction of feature was about 24% in the first experiment and about 40% in the last one.

3.5. Experiments considering the error tolerance

This set of experiments considering the digit recognition task considers some loss in terms of recognition rate through the use of the parameter ERROR-TOL (Error tolerance) of the OpHC algorithm. Initially, the idea was to investigate the search space around a local maximum in order to verify the possibility of starting again to observe an improvement in terms of recognition performance by removing additional features. This expectative were not confirmed. However, it was possible to observe a significant reduction on the number of features even by using a small tolerance of error. Table 3 shows for each error tolerance evaluated the respective number of features selected, and the recognition performance of the corresponding classifier.

Table 3. Experimental results by considering error tolerance

	Error Tolerance (ERROR_TOL)	# Feat	Training	Testing	Loss of recognition performance
C1	0%	77	99.86%	98,94%	
C2	1%	59	99,30%	98,72%	-0,32%
C3	2%	52	99,40%	98,55%	-0,49%
C4	4%	33	98,92%	97,85%	-1,19%
C5	8%	25	98,65%	97,43%	-1,61%
C6	12%	26	98,06%	96,93%	-2.11%

A byproduct of these experiments is the possibility to design a cascade classifier as shown in Figure 8.

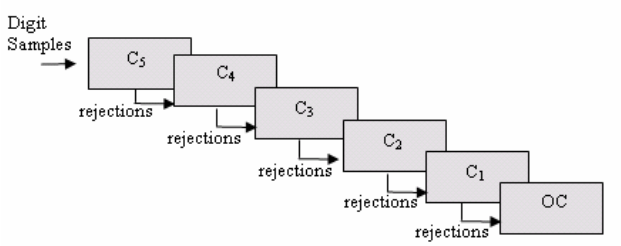

Figure 8. Cascade classifier using C5, C4, C3, C2, C1 and the Original Classifier

In this scheme, the classifiers C5, C4, C3, C2, C1 and OC (in this sequence) deal with the rejections of their respective predecessors. C6 is not used since it uses more features than C5. The rejection thresholds T_i were fixed in order to produce error rates of 0.5% for each classifier C_i, except for OC which is zero. Table 4 resumes all the results. As one can see, 87.72% of the digit samples (26,565 from 30,089) were well-recognized by classifier C1, and only 3,395 samples were rejected. Finally, only 677 digit samples from 30,089 arrive to the original classifier OC.

Table 4. Experimental results using the cascade classifier

Classifier	C_5	C_4	C_3	C_2	C_1	OC	Cascade
# features	26	33	52	59	77	132	
# samples correct-recognized	26565	722	1381	430	179	528	29805
# samples mis-recognized	129	1	4	1	0	149	284
# samples rejected	3395	2672	1287	856	677	0	0
% recognition	99.5	99.5	99.5	99.5	99.5	78.0	99.1
% rejection	11.2	78.7	48.1	66.5	79.0	0	0
% errors	0.5	0.5	0.5	0.5	0.5	22	0.9
# MLP connections	936	1419	2704	4071	6699	14200	

The final recognition performance obtained by using the cascade classifier was the same than using the original classifier with the entire feature set. However, it was possible observe a significant reduction in terms of computational complexity for the classification task. To explain, let us consider the complexity of the original classifier as:

$$Compl_{OC} = ns \times nc$$

where

ns: number of samples;
nc: number of connections of the MLP classifier;
and, the complexity of the cascade classifier as:

$$Compl_{CC} = \sum_{i=1}^{NC} (ns_i \times nc_i)_I$$

where

NC: number of classifiers;
ns_i: number of samples classified by the i[th] classifier;
nc_i: number of connections of the i[th] MLP classifier.

Thus, $Compl_{oc}$ and $Compl_{cc}$ are 427,263,800 and 57,532,586, respectively. It means an impressive reduction of 86,54% in terms of computational complexity for the classification task of 30,089 digit samples.

4. Conclusion

The proposed method for feature subset selection has shown to be an interesting strategy to implement a wrapper-based method, which can be executed in low-cost hardware architecture. In the experiments, a reduction in the original number of features 132 to 77 features (around 42%) without a significant loss in terms of digit recognition rate was observed. Similar result was observed for uppercase characters, considering a reduction in the number of features around 40%.

A cascade classifier obtained as a byproduct of the proposed feature subset selection method when an error tolerance threshold is used during the search. This classifier has shown a significant reduction in terms of computational complexity by 86.54% for the digit recognition task, while it keeps the same recognition performance than using the original classifier configuration based on the entire feature set.

References

[1] Jain, A., Duin, R. P. W. and Mao, J., "Statistical pattern recognition: a review", IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 22(1), pp. 4-37, January 2000.

[2] Boz, Olcay, "Feature subset selection by using sorted feature relevance", ICMLA International Conference on Machine Learning and Application, Las Vegas City, USA, June, pp. 147-153, 2002.

[3] Molina, Luis Carlos, Belanche, Lluis and Nebot, Àngela, "Feature selection algorithms: survey experimental and evaluation", IEEE International Conference on Data Mining, vol. 1, pp. 306-313, Maebashi City, Japan, December 2002.

[4] Blum, Avrim L. and Langlev Pat, "Selection of relevant features and examples in machine learning", special issue Artificial Intelligence, vol. 97, n.1-2, pp. 245-271, December 1997.

[5] Aha, David W. and Richard L. Bankert, "Feature selection for case-based classification of cloud types: an empirical comparison", *Working Notes of the AAAI, Workshop on Case-Based Reasoning*, vol. 1, pp. 106-112, 1994.

[6] Raman, Baranidharan. "Enhancing inductive learning using feature and example selection", Master Thesis submitted to Texas A&M University, 67p. 2003.

[7] Kohavi, Ron and John, George H., "Wrappers for feature subset selection", in *Artificial Intelligence Journal*, Special Issue On Relevance, vol. 97, no. 1, pp. 273-324, 1997.

[8] Oliveira, L. S., Sabourin R., Bortolozzi F., and C. Y. Suen, "Feature selection using multi-objective genetic algorithms for handwritten digit recognition", *Proc. International Conference on Pattern* Recognition, vol. 1, pp. 568-571, Quebec City, August 2002.

[9] Oliveira, L.S., Sabourin, R., Bortolozzi, F. and Suen, C.Y., "A methodology for feature selection using multi-objective genetic algorithms for handwritten digit string recognition", in the International Journal of Pattern Recognition and Artificial Intelligence (IJPRAI), vol. 17, n. 6, pp. 903-929, 2003.