Multiclass cascades for ensemble-based boosting algorithms

Susnjak, Teo; Barczak, Andre; Reyes, Napoleon; Hawick, Ken

Published in:
STAIRS 2012: Proceedings of the Sixth Starting AI Researchers' Symposium

DOI:
10.3233/978-1-61499-096-3-330

STAIRS 2012
K. Kersting and M. Toussaint (Eds.)

doi:10.3233/978-1-61499-096-3-330

Multiclass Cascades for Ensemble-based Boosting Algorithms

Teo SUSNJAK [a,1], Andre BARCZAK [a] Napoleon REYES [a] and Ken HAWICK [a]
[a] *Institute of Information and Mathematical Sciences, Massey University Albany, New Zealand*

Abstract. We propose a general method applicable to existing multiclass boosting-algorithms for creating cascaded classifiers. The motivation is to introduce more tractability to machine learning tasks which require large datasets and involve complex decision boundaries, by way of separate-and-conquer strategies that reduce both the training and detection-phase overheads. The preliminary study explored the application of our method to AdaBoost.ECC on six UCI datasets and found that a decrease in the computational training and evaluation overheads occurred without significant effects on the generalization of the classifiers.

Keywords. ensemble-based learning, classifier cascades, boosting, multiclass classification,

1. Introduction

The combination of ensemble-based machine learning methods with boosting and weak underlying models, have experienced widespread use recently due to their effectiveness at addressing many challenging classification problems. The ability to further combine these approaches with coarse-to-fine learning strategies that partition the induction and the classifier execution phases into cascades, has been a particularly important development for complex tasks that involve massive datasets and expensive features. This is commonly experienced in computer vision, where classifier cascading techniques have been successfully applied for binary-class problems and have enabled significantly faster training and real-time detection runtimes.

However,the application of cascaded classifiers to the challenges of multiclass domains is still an open problem [1]. Most common methods have involved either constructing separate parallel cascades for each class [2], or building cascaded detector trees [3]. Recently, notable contributions by Verschae et al. [4] have seen the development of multiclass cascades for multi-view face detection.

We present a preliminary study into the feasibility of converting existing multiclass boosting algorithms into integrated cascaded algorithms. We use the concepts of classifier margins to construct cascades with embedded rejection points

[1]Corresponding Author E-mail: T.Susnjak@massey.ac.nz.

in an ensemble that are configurable based on the required confidence levels. The investigation of the proposed generic approach was trialled on the AdaBoost.ECC [5] algorithm and tested on six UCI datasets.

2. Proposed Algorithm

The AdaBoost.ECC merges error correcting output coding with boosting. The columns of the coding matrix are iteratively generated after each round using the *colouring* function μ. An additional distribution $\tilde{D}$ is maintained to maximize the error correcting ability of each column in the coding matrix. The final hypothesis H on sample x is computed as being the class label l, which receives the highest weighted vote from all class labels returned by $h_t(x)$.

The proposed method is highlighted in Algorithm 1, which is an intuitive extension of AdaBoost.ECC. Each cascade layer consists of wc weak classifiers. At the completion of a given layer, the sums of weighted votes for each class label from all samples are calculated. For each class y, the maximum sum of correct weighted-votes p_y for a true positive sample is calculated, as well as the highest sum q_y assigned to a false positive sample. The difference of $p_y - q_y$ represents the maximum achieved confidence margin for a given class y. We combine this difference with a tunable coefficient Φ, ranging between 0 and 1 in order to set the required confidence threshold a_y for class y. Coarse-to-fine learning and detection is realized when a sample is removed from further training or evaluation at detection time when its confidence vote satisfies the threshold margin for y, thus increasingly focusing on more difficult samples. In order to mitigate against overfitting, γ represents the minimum proportion of samples per class that must remain during training in respect to the original class total.

Algorithm 1 Cascaded AdaBoost.ECC

Given: Dataset $D(x_1, y_1), ..., (x_m, y_m)$ where $x_i \in X, y_i \in Y$, L = total layers, T = total boosting rounds, wc = total weak classifiers per layer, Φ = confidence threshold margin, γ = lowest limit for removing samples of a given class, $\vec{p}_{y,...,Y}, \vec{q}_{y,...,Y}$ vector of the highest correct and incorrect vote respectively for class label y

Output: Hypothesis $H_{final}(x) = \arg\max_{\ell \in Y} \sum_{t=1}^{T} g_t(x)\mu_t(\ell)$ and cascade hash table $M_{t,\vec{a}}$ where t^{th} classifier accesses vector $a_{y,...,Y}$ of confidence threshold values for each class label y at a given layer

$wc = T/L$
Initialize $\tilde{D}_1(i, \ell) = [\![\ell \neq y_i]\!]/(m(k-1))$
for $t = 1$ to T **do**
  Compute colouring $\mu : Y \rightarrow \{-1, 1\}$
  Let $U_t = \sum_{i=1}^{m} \sum_{\ell \in Y} \tilde{D}_t(i, \ell)[\![\mu_t(y_i) \neq \mu_t(\ell)]\!]$
  Let $D_i = \frac{1}{U_t} \cdot \sum_{\ell \in Y} \tilde{D}_t(i, \ell)[\![\mu_t(y_i) \neq \mu_t(\ell)]\!]$
  Train weak learner on examples $(x_1, \mu_t(y_1)), ..., (x_m, \mu_t(y_m))$ weighted according to D_t
  Get weak hypothesis $h_t : X \rightarrow \{-1, 1\}$
  Compute the weight of positive and negative votes α_t and β_t respectively
  Define: $g_t(x) = \begin{cases} \alpha_t & \text{if } h_t(x) = 1 \\ \beta_t & \text{if } h_t(x) = -1 \end{cases}$
  if $t \ \% \ wc = 0$ **then**
    Get highest correct vote $\forall y, p_y = \arg\max_{\ell \in Y} D_i \sum_{s=1}^{t} g_s(x_i)\mu_s(\ell)$ where $\ell = y_i$
    Get highest incorrect vote $\forall y, q_y = \arg\max_{\ell \in Y} D_i \sum_{s=1}^{t} g_s(x_i)\mu_s(\ell)$ where $\ell \neq y_i$
    Set layer thresholds for each class $\forall y, M_{t,a_y} = (p_y - q_y) \times \Phi$
    Remove samples from training set if $\arg\max_{\ell \in Y} \sum_{s=1}^{t} g_s(x_i)\mu_s(\ell) > a_\ell$ and $\sum D(i, \ell)/Tot_\ell > \gamma$
  end if
  Update $\tilde{D}_{t+1}(i, \ell) = \frac{1}{Z_t} \cdot \tilde{D}_t(i, \ell) \exp\{(g_t(x_i)\mu_t(\ell) - g_t(x_i)\mu_t(y_i)) \cdot \frac{1}{2}\}$
  where $\tilde{Z}_t$ is the normalization factor so that $\tilde{D}_{t+1}$ will sum to 1.
end for

3. Method

We evaluated five cascaded-algorithm classifiers with various parameters against the conventional AdaBoost.ECC, using decision stumps as weak learners. Each cascaded classifier consisted of 10 layers of equal size. In formulating the tunable parameters, at $\Phi = 0$, rapid learning would be expected resulting in unreliable thresholds and a poor generalization; while the reverse could be anticipated at the opposite spectrum for $\Phi = 1$. Therefore, the five classifiers were trained as follows; three with $\Phi = \{0.5, 0.75, 0.9\}$ and $\gamma = 0.1$; one with $\Phi = 0.9$ and $\gamma = 0.3$; one with $\Phi = 0.5$ and $\gamma = 1.0$ denoting no removal of samples during training.

The classifiers were trained on six UCI datasets containing predefined training and test datasets. Dataset details are shown with the results in Table 2. Each classifier was trained 10 times; the results were aggregated and standard deviations calculated. Both the test errors and the g-mean measures were examined due to the presence of skewed class distributions. Mean ranks were used to summarize the accuracy results and the non-parametric Friedman and the Iman-Davenport statistical tests were applied in order to assess their significance.

4. Results

The effects of the coarse-to-fine learning strategy on the training runtime phase of the classifiers is first examined. Figures 1 depict the cumulative proportion of samples that successfully satisfy the confidence margin per layer and are removed from the training of subsequent layers. The figures indicate that for smaller confidence margin values of Φ, the faster the learning and the removal rate of samples occurs. Given the larger complexity of the Letter dataset in terms of the number of class labels, the learning rate is markedly slower than that of the Optdigits dataset with fewer class labels.

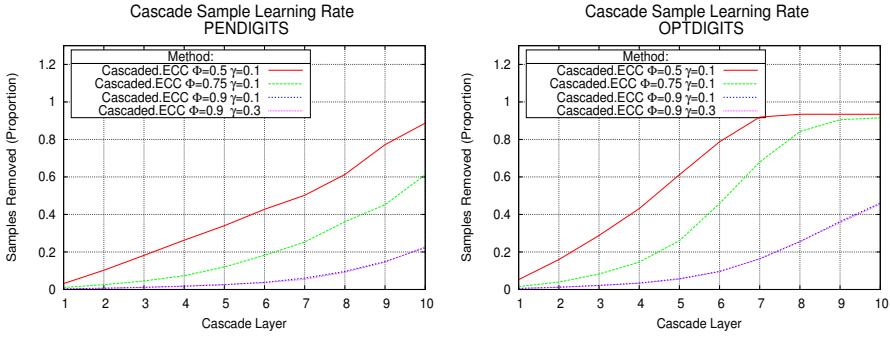

Figure 1. Cumulative proportion of training samples learned and removed per layer.

Table 1 demonstrates the factor by which the cascading method has achieved an acceleration in training runtimes over the ECC algorithm. The improvement in the training runtime over ECC classifiers occurred in most cases except for clas-

sifiers with the $\Phi = 0.5$ and $\gamma = 1.0$ parameters where the cascading thresholds were calculated but the samples satisfying it were not removed. The marginally slower runtimes of these classifiers reflects the additional overheads in calculating the rejection thresholds. The most significant speed-up over the ECC classifiers was seen on the Shuttle dataset. The table is summarized with the mean and geometric mean values for more precision for ratio-scale values. The fastest runtimes were attained by the cascaded classifiers with the lowest values of Φ.

Table 1. Factor of training runtime speed-up of cascaded classifiers over ECC classifiers.

		Cascaded.ECC classifier parameters				
	Φ	0.5	0.75	0.9	0.5	0.9
Dataset	γ	0.1	0.1	0.1	1.0	0.3
Letter		1.04	1.02	1.01	1	1
Pendigit		1.42	1.14	1.03	0.98	1.02
Optdigit		2.03	1.51	1.11	0.99	1.09
Satimage		1.32	1.16	1.06	0.98	1.06
Vowel		2.42	1.94	1.4	0.98	1.38
Shuttle		4.82	2.91	1.3	1	1.27
Average		**2.18**	**1.61**	**1.15**	**0.99**	**1.14**
Geometric mean		**1.89**	**1.5**	**1.14**	**0.99**	**1.13**

The total error rates as a function of the training runtimes are shown in Figure 2 for two datasets. The graphs indicate the tendency of the cascaded algorithm to produce classifiers which generate notably lower test error rates for training runtimes in the initial phases of training. The graphs show that both the process of rapidly learning and removing trivial samples in the initial layers of a cascade, resulted in faster training runtimes. The classification of samples by the earlier layers of the cascade has also accurately encoded the decision boundaries between the classes, preserving a strong generalizability.

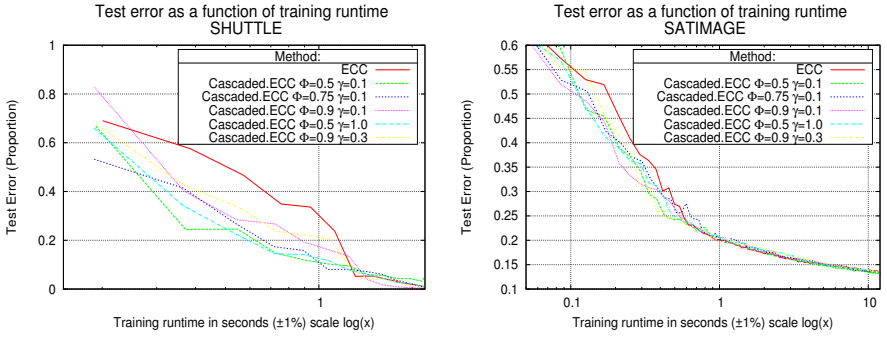

Figure 2. Total test error rates as a function of the training runtimes.

Comprehensive accuracy of the classifiers on the test sets are shown in Table 2. The results indicate that comparable accuracies have been attained by all classifiers. Notably poorer generalization was seen by cascaded classifiers particularly on the Vowel dataset. The test error convergence graphs for the cascaded classifiers trained with $\Phi = 0.5, 0.75, 0.9$ and $\gamma = 0.1$ parameters, in particular

showed that overfitting was taking place in the final layers of the cascade as the total number of training samples were reduced. This was mitigated on the cascaded classifier trained with $\Phi = 0.9$ and $\gamma = 0.3$. On this dataset however, the classifier with $\Phi = 0.5$ and $\gamma = 1.0$ settings attained the highest accuracy on both metrics. This indicates that the use of cascading margin-thresholds is effective for classification but may lead to overfitting on datasets with high levels of noise as the number of training samples decrease.

Table 2. Test error rates as proportions and the g-means for all classifiers on the six UCI datasets. Total boosting rounds, total samples and class labels per dataset are also listed respectively.

		Cascaded.ECC					ECC
	Φ	0.5	0.75	0.9	0.5	0.9	
Dataset	γ	0.1	0.1	0.1	1.0	0.3	
LETTER	Test error	**0.185**	0.187	0.188	0.189	0.188	0.186
(2000)		**±0.003**	±0.005	±0.004	±0.004	±0.006	±0.004
(20000)	G-mean	**0.813**	0.811	0.810	0.810	0.810	0.812
(26)		**±0.005**	±0.006	±0.005	±0.005	±0.009	±0.006
PENDIGITS	Test error	0.08	0.061	0.058	0.058	**0.056**	0.057
(500)		±0.02	±0.002	±0.002	±0.002	**±0.003**	±0.001
(10992)	G-mean	0.91	0.938	0.942	0.942	**0.944**	0.942
(10)		±0.02	±0.004	±0.003	±0.003	**±0.004**	±0.001
OPTDIGITS	Test error	0.11	0.074	0.059	**0.058**	0.062	0.059
(500)		±0.02	±0.005	±0.003	**±0.003**	±0.004	±0.001
(5620)	G-mean	0.89	0.925	0.940	**0.942**	0.937	0.941
(10)		±0.03	±0.008	±0.003	**±0.003**	±0.005	±0.002
SATIMAGE	Test error	**0.128**	0.13	0.130	0.131	0.131	0.129
(500)		**±0.004**	±0.01	±0.004	±0.002	±0.004	±0.003
(6197)	G-mean	**0.846**	0.845	0.840	0.845	0.842	0.845
(6)		**±0.009**	±0.008	±0.008	±0.006	±0.004	±0.005
VOWEL	Test error	0.80	0.76	0.64	**0.57**	0.61	0.59
(500)		±0.03	±0.03	±0.03	**±0.02**	±0.02	±0.01
(990)	G-mean	0.29	0.27	0.32	**0.39**	0.33	0.37
(11)		±0.07	±0.05	±0.06	**±0.02**	±0.04	±0.02
SHUTTLE	Test error	0.03	0.02	0.01	**0.0001**	0.01	**0.0001**
(1000)		±0.02	±0.03	±0.01	**±0.0000**	±0.02	**±0.0000**
(58000)	G-mean	0.77	0.78	0.84	**0.98**	0.91	**0.98**
(9)		±0.16	±0.14	±0.13	**±0.01**	±0.05	**±0.02**
Mean Ranks (Total error)		4.3	4.5	3.3	3.2	3.7	2.0
Mean Ranks (G-mean)		4.2	4.3	4.2	2.8	3.3	2.2

The ECC attained the best mean ranks on both metrics. Statistical analysis of the ranks was conducted to determine significant differences from the 3.0 mean rank, expected under the null-hypothesis. For the total error mean-ranks, the Friedman statistic $\chi^2_F(5) = 7.048, p < 0.25$ was calculated, while the Iman-Davenport statistic yielded $F_F(5, 25) = 1.535, p < 0.5$, from which the null-hypothesis could not be rejected. Likewise, for the g-mean ranks, the Friedman test produced $\chi^2_F(5) = 6.571, p < 0.5$ and the Iman-Davenport test $F_F(5, 25) = 1.402, p < 0.5$. In both cases the significance was insufficient to reject the null-hypothesis; thus, no further *post hoc* tests were warranted, since there was inadequate evidence from this data to indicate that the accuracy of the algorithms differed.

Finally, the effects of the cascading approach on the detection runtime of the classifiers is examined. Table 3 shows the factor by which the cascaded classifiers exceed the detection runtimes of the ECC classifiers. Once again, the most rapid runtimes were recorded by cascaded classifiers lowest values of Φ and ranged up to a 800% improvement. Though overall decreases in the execution runtimes of the cascaded classifiers with $\Phi = 0.9$ over ECC were realized, they were comparatively modest. The fact that this artificial problem domain involved no feature extraction at detection time needs to be taken into consideration. Given a real-world problem set involving the extraction of computationally intensive features, it can be expected that the reduction in the detection runtimes would substantially increase.

Table 3. Factor of detection time speed-up of cascaded classifiers over ECC classifiers.

		Cascaded.ECC classifier parameters				
	Φ	0.5	0.75	0.9	0.5	0.9
Dataset	γ	0.1	0.1	0.1	1.0	0.3
Letter		1.06	1.02	1.03	1.03	1.01
Pendigit		1.41	1.12	0.99	1.19	0.99
Optdigit		2.01	1.44	1.05	1.63	1.04
Satimage		1.23	1.07	0.99	1.07	1.02
Vowel		2.89	1.79	1.22	2.06	1.24
Shuttle		7.98	3.5	1.26	4.62	1.23
Arithmetic mean		**2.76**	**1.66**	**1.09**	**1.93**	**1.09**
Geometric mean		**2.1**	**1.49**	**1.08**	**1.65**	**1.08**

5. Conclusion

This research has proposed a generic method for training cascaded classifiers using existing multiclass boosting algorithms. The motivation was to address complex machine learning tasks which entail massive datasets and complex decision boundaries by applying coarse-to-fine learning strategies for reducing runtime overheads at training and detection by directing the focus on difficult samples. The preliminary experiments into this method examined the results on six UCI datasets. The results indicated that a reduction in both the training and execution runtimes of the cascaded classifiers over conventional classifiers was realized, without incurring significant penalties in their generalizability.

References

[1] Zhang, C., Ma, Y.: Ensemble Machine Learning: Methods and Applications. Springer-Verlag New York Inc (2012)
[2] Schneiderman, H., Kanade, T.: A statistical method for 3d object detection applied to faces and cars. In: IEEE Conf. on Comp. Vis. and Patt. Recog. (CVPR 2000). (2000) 1746–1759
[3] Lienhart, R., Liang, L., Kuranov, A.: A detector tree of boosted classifiers for real-time object detection and tracking. In: ICME2003, IEEE (2003) 277–280
[4] Verschae, R., del Solar, J.R.: Coarse-to-fine multiclass nested cascades for object detection. Pattern Recognition, International Conference on (2010) 344–347
[5] Guruswami, V., Sahai, A.: Multiclass learning, boosting, and error-correcting codes. In: Proc. of the 12th an. conf. on Comp. learning theory. COLT '99, NY, ACM (1999) 145–155