Top-k Parametrized Boost

Turki Turki[1,4], Muhammad Amimul Ihsan[2], Nouf Turki[3], Jie Zhang[4], Usman Roshan[4]

[1] King Abdulaziz University
P.O. Box 80221, Jeddah 21589, Saudi Arabia
tturki@kau.edu.sa
[2] Department of Electrical Engineering, Stanford University
350 Serra Mall, Stanford, CA 94305, United States
aihsan@stanford.edu
[3] King Abdulaziz University
P.O. Box 80200, Jeddah 21589, Saudi Arabia
nouf.turkey@aol.com
[4] Department of Computer Science, New Jersey Institute of Technology
University Heights, Newark, NJ 07102
{ttt2,jz266}@njit.edu, usman@cs.njit.edu

Abstract. Ensemble methods such as AdaBoost are popular machine learning methods that create highly accurate classifier by combining the predictions from several classifiers. We present a parametrized method of AdaBoost that we call Top-k Parametrized Boost. We evaluate our and other popular ensemble methods from a classification perspective on several real datasets. Our empirical study shows that our method gives the minimum average error with statistical significance on the datasets.

Keywords: Ensemble methods, AdaBoost, statistical significance.

1 Introduction

Ensemble methods are popular machine learning methods that produce a single highly accurate classifier by combining the predictions from several classifiers [1]. Among many such methods the AdaBoost(AB) [2] is a very popular choice. AB outputs a single classifier by combining T weighted classifiers and prediction [3] is given by

$$h^*(x_j) = \arg\max_y \sum_{i=1}^{T} \alpha_i I(h_i(x_j) = y) \tag{1}$$

where $x_j \in R^d$ for $j = 1...m$, $y \in \{-1, +1\}$, h_i the ith classifier that maps the instance x_j to y , α_i the weight of h_i, and $I(.)$ is an indicator function that outputs 1 if its argument is true and 0 otherwise. In this paper we consider a parametric version of Equation 1 that we call the Parametrized AdaBoost(P-AdaBoost) given by

$$h^*(x_j) = \arg\max_y \sum_{i=1}^{T} \beta\alpha_i I(h_i(x_j) = y), \quad (2)$$

where $\beta = \sum_{i=1}^{k} \beta_i \leq 1$ and $0 < \beta_i \leq 1$. In addition to P-AdaBoost, we present a method that we call Top-k Boost which uses P-AdaBoost to search for the top-k parameter values in β that achieve the best classification results. We combine the k parameter values to produce optimal classifier. For given dataset we obtain this optimal classifier and β by cross validation. Both P-AdaBoost with Top-k Boost yield our method which we call Top-k Parametrized Boost. Compared to other popular ensemble methods our empirical study on 25 datasets shows that Top-k Parametrized Boost yields the minimum average error with statistical significance.

The rest of this paper is organized as follows. In section 2 we review related work. In section 3 we present our method Top-k Parametrized Boost. Following that we present empirical study and discussion before concluding.

2 Related Work

AdaBoost [2] combines sequentially classifiers(e.g. decision trees) to produce highly accurate classifier. In detail, the AdaBoost algorithm [4] which is outlined in Algorithm 1 works as follows. Line 1 receives as an input a set of m labeled training examples $S = \{(x_1, y_1), ..., (x_m, y_m)\}$ where the label associated to the instance $x_m \in R^d$ is $y_m \in \{-1, +1\}$ drawn i.i.d to a distribution used for both validation and training. Lines 2-4 initialize F, which will hold T weighted classifiers [3]. Lines 5-7 assign equal weight distribution to all training examples.The for loop of lines 8-16 update weight distribution and combine T weighted classifiers. In Line 9 the learning algorithm will find classifier $h_t \in H$ using weight distribution D_t to map instances in x to y with small error. Line 10 incurs the loss $Pr_{D_t}[y_i \neq h_t(x_i)] = \sum_{i=1}^{m} D_t I[y_i \neq h_t(x_i)]$. Line 11 calculates α_t the weight of the classifier h_t. Line 12 stores the weighted classifier $\alpha_t h_t$. Lines 13-15 update weight distribution [3] by upweighting of examples which are incorrectly classified to focus on and decreasing the weights of correctly classified examples. Lines 17 gives the final weighted classifier. Line 18 yields the prediction by taking the sign of the sum of T weighted classifiers(Equation 1).

Algorithm 1 AdaBoost algorithm

```
1:  AdaBoost($S = \{(x_1, y_1), ..., (x_m, y_m)\}$)
2:    for i = 1 to m do
3:      $F_0(i) \leftarrow 0$
4:    end for
5:    for i = 1 to m do
6:      $D_1(i) \leftarrow \frac{1}{m}$
7:    end for
8:    for t = 1 to T do
9:      $h_t \leftarrow fit\ classifier\ h_t \in H\ with\ D_t$
10:     $\epsilon_t \leftarrow Pr_{D_t}[y_i \neq h_t(x_i)]$
11:     $\alpha_t \leftarrow \frac{1}{2} \log \frac{1-\epsilon_t}{\epsilon_t}$
12:     $F_t \leftarrow F_{t-1} + \alpha_t h_t$
13:     for i = 1 to m do
14:       $D_{t+1}(i) \leftarrow \frac{D_t(i) exp(-\alpha_t h_t(x_i) y_i)}{\sum\limits_{i=1}^{m} D_t(i)}$
15:     end for
16:   end for
17:   $F \leftarrow \sum\limits_{t=1}^{T} \alpha_t h_t$
18:   $h^* = sgn(F)$
```

3 Top-k Parametrized Boost

As shown above the AdaBoost algorithm [3,4] receives the labeled training sample S as an input(line 1) and outputs the final weighted classifier as an output (line 17). Our algorithm outlined in Algorithm 3 uses as a subroutine a parametrized version of the AdaBoost algorithm [3,4] which is outlined in Algorithm 2. The P-AdaBoost algorithm receives a fixed parameter $\beta \in (0, 1)$ and a training sample $S = \{(x_1, y_1), ..., (x_m, y_m)\}$ where the label associated to the instance $x_m \in R^d$ is $y_m \in \{-1, +1\}$(line 1). Lines 2-11 are the same as AdaBoost algorithm outlined in Algorithm 1. Line 12 stores the parameterized weighted classifiers. Lines 17 gives the final parametrized weighted classifier. Line 18 yields the prediction by taking the sign of the sum of T parametrized weighted classifiers(Equation 2) . Line 19 incurs the loss $Pr[y \neq h^*(x)] = \sum\limits_{i=1}^{m} I[y_i \neq h^*(x_i)]$. Line 20 returns (F, E) the final classifier and the corresponding error respectively.

We can now use the P-AdaBoost algorithm as a subroutine in the Top-k Boost algorithm which is outlined in Algorithm 3. The Top-k Boost algorithm outputs the final parametrized weighted classifier(line 33 in Algorithm 3) that outperforms the AdaBoost's final classifier(line 17 in Algorithm 1). To output the final parametrized weighted classifier(line 33 in Algorithm 3), we make the initial call Top-k Boost(β, S), where $\beta =< \beta[1], ..., \beta[n] >$ for fixed number of parameter values n to be precisely specified in section 4.1 such that $\beta_i \in (0, 1)$ and $S = \{(x_1, y_1), ..., (x_m, y_m)\}$ where $x_m \in R^d$ and $y_m \in \{-1, +1\}$. The for loop of Lines 2-5 iterates n times to store the parameter values in β and the errors of the corresponding classifiers returned by P-AdaBoost algorithm in (P, E) respectively. Line 7-12 store the error of classifier h_l with $\beta_l = 1$ in *error* variable(line 9), where $F \leftarrow \sum_{t=1}^{T} \beta\alpha_t h_t$(line 17 in Algorithm 2) $= F \leftarrow \sum_{t=1}^{T} \alpha_t h_t$(line 17 in Algorithm 1) when $\beta = 1$. The *error* variable(line 9) corresponds to the same error incurred by AdaBoost algorithm outlined in Algorithm 1. The for loop of Lines 14-19 searches for parameter values in β already stored in P that achieve the same or better performance than one by $\beta_l = 1$. The Min subroutine in line 22 finds the ith smallest error e_i to return the corresponding ith parameter b_i which is stored in P_i. The for loop of lines 26-32 adds the top-k parameter values in P, where $\sum_{i=1}^{k} P_i \leq 1$. In line 33 we call P-AdaBoost by passing the top-k parameter values in β and training sample S to obtain the final classifier as shown in line 33. Line 34 yields the prediction by taking the sign of the sum of T parametrized weighted classifiers(Equation 2).

Algorithm 2 P-AdaBoost algorithm

```
1:  P-AdaBoost($\beta, S = \{(x_1, y_1), ..., (x_m, y_m)\}$)
2:    for i = 1 to n do
3:      $F_0(i) \leftarrow 0$
4:    end for
5:    for i = 1 to m do
6:      $D_1(i) \leftarrow \frac{1}{m}$
7:    end for
8:    for t = 1 to T do
9:      $h_t \leftarrow fit\ classifier\ h_t \in H\ with\ D_t$
10:     $\epsilon_t \leftarrow Pr_{D_t}[y_i \neq h_t(x_i)]$
11:     $\alpha_t \leftarrow \frac{1}{2}\log\frac{1-\epsilon_t}{\epsilon_t}$
12:     $F_t \leftarrow F_{t-1} + \beta\alpha_t h_t$
13:     for i = 1 to m do
14:       $D_{t+1}(i) \leftarrow \frac{D_t(i)exp(-\alpha_t h_t(x_i)y_i)}{\sum_{i=1}^{m} D_t(i)}$
15:     end for
16:   end for
17:   $F \leftarrow \sum_{t=1}^{T} \beta\alpha_t h_t$
18:   $h^* = sgn(F)$
19:   $E \leftarrow Pr[y \neq h^*(x)]$
20:   return $(F, E)$
```

Algorithm 3 Top-k Boost algorithm

```
1:  **Top-k Boost**$(\beta, S = \{(x_1, y_1), ..., (x_m, y_m)\})$
2:    for $l = 1$ to $length(\beta)$ do
3:      $(h_l, \epsilon_l) \leftarrow P - AdaBoost(\beta_l, S)$
4:      $(P, E) \leftarrow (\beta_l, \epsilon_l)$
5:    end for
6:    $(error, index) \leftarrow (0, 0)$
7:    for $l = 1$ to $length(\beta)$ do
8:      if $P_l = 1$ then
9:        $(error, index) \leftarrow (E_l, l)$
10:       $break$
11:     end if
12:   end for
13:   $k \leftarrow 1$
14:   for $l = 1$ to $length(\beta)$ do
15:     if $l \neq index\ and\ error \geq E_l$ then
16:       $(b_k, e_k) \leftarrow (P_l, E_l)$
17:       $k \leftarrow k + 1$
18:     end if
19:   end for
20:   $P \leftarrow 0$
21:   for $i = 1$ to $k$ do
22:     $(b_i) \leftarrow Min(e, i)$
23:     $P_i \leftarrow b_i$
24:   end for
25:   $\beta \leftarrow P_1$
26:   for $i = 2$ to $k$ do
27:     if $\beta + P_i \leq 1$ then
28:       $\beta \leftarrow \beta + P_i$
29:     else
30:       $break$
31:     end if
32:   end for
33:   $(F, E) \leftarrow P - AdaBoost(\beta, S)$
34:   $h^* = sgn(F)$
```

4 Empirical Study

To evaluate the performance of our method, an empirical study from a classification perspective [5] is performed on 25 real datasets shown in Table 1 from the UCI Machine Learning Repository [6]. This section describes the experimental methodology, then presents the experimental results.

4.1 Experimental Methodology

We compare three ensemble classification algorithms: Top-k Parametrized Boost (T-K PB), AdaBoost(AB), Random forests(RF) [7]. For each dataset we use the

Code	Dataset	Classes	Dimension	Instances
1	Haberman's Survival	2	3	306
2	Skin Segmentation	2	3	245057
3	Blood Transfusion Service Center	2	4	748
4	Liver-disorders	2	6	345
5	Diabetes	2	8	768
6	Breast Cancer	2	10	683
7	MAGIC Gamma Telescope	2	10	19020
8	Planning Relax	2	12	182
9	Heart	2	13	270
10	Australian Credit Approval	2	14	690
11	Climate	2	18	540
12	Two norm	2	20	7400
13	Statlog German credit card	2	24	1000
14	Breast cancer	2	30	569
15	Ionosphere	2	34	351
16	Qsar	2	41	1055
17	SPECTF heart	2	44	267
18	Spambase	2	57	4597
19	Sonar	2	60	208
20	Digits	2	63	762
21	Ozone	2	72	1847
22	Insurance company coil2000	2	85	5822
23	Hill valley	2	100	606
24	BCI	2	117	400
25	Musk	2	166	476

Table 1: Datasets from the UCI Machine Learning repository that we used in our empirical study [6]

10-fold cross-validation with the same splits for each algorithm. For ensemble method we find the best parameter values with further cross-validation on the training set.

In T-K PB we let β range from $\{1,.9,.8,.7,.6,.5,.4,.375,.35,.3,.25,.2,.15,.1,0.05\}$. Recall that T-K PB is given by $h^*(x) = \underset{y}{\operatorname{argmax}} \sum_{t=1}^{T} \beta\alpha_t I(h_t(x) = y)$ [3]. For each parameter value we select the parameter value that gives the minimum error on the training. Thus we obtain the top-k values of parameter β with cross-validation on the training set. We then apply the top-k parameter values of β to the validation set. We wrote our code in R.

4.2 Experimental Results on Twenty Five Datasets

We measure the error as the number of the validation instances incorrectly predicted divided by the number of validation instances. We compute the error of our classification tasks for each training-validation split in the cross-validation and take the the average result of ten folds to be the average cross-validation error. In Table 2 we show the average cross-validation error on each dataset. Over the 25 datasets T-K PB gives the minimum average error of 10.468% and has the minimum error in 22 out of 25 datasets. The second best is RF that gives an average error of 15.174% and has the minimum error in 2 out of 25 datasets. AB gives higher average error of 15.520% and has the minimum error in 1 out of 25 datasets.

We measure the statistical significance of the methods with Wilcoxon rank test [5, 8]. This test is a standard test to measure the statistical significance between two methods in many datasets. It shows that if one method outperforms the other in many datasets then it is considered statistically significant. In Table 3 the p-value shows that our method T-K PB outperforms the other two methods in the 25 datasets with statistical significance.

Code	Dataset	T-K PB	AB	RF
1	Haberman's Survival	0.31154	0.35769	**0.26923**
2	Skin Segmentation	**0.00005**	0.0005	0.00044
3	Blood Transfusion Service Center	0.21470	**0.21176**	0.22647
4	Liver-disorders	**0.236**	0.3	0.244
5	Diabetes	**0.21618**	0.27353	0.24412
6	Breast Cancer	**0.00794**	0.02381	0.02222
7	MAGIC Gamma Telescope	0.16142	0.16395	**0.13205**
8	Planning Relax	**0.3**	0.39167	0.375
9	Heart	**0.14**	0.185	0.16
10	Australian Credit Approval	**0.06167**	0.14167	0.14833
11	Climate	**0.026**	0.044	0.06
12	Two norm	**0.00795**	0.02534	0.02356
13	Statlog German credit card	**0.24222**	0.26333	0.25889
14	Breast cancer	**0.00794**	0.02381	0.02222
15	Ionosphere	**0.00323**	0.03871	0.04194
16	Qsar	**0.10211**	0.13368	0.12737
17	SPECTF heart	**0.11765**	0.22941	0.26471
18	Spambase	**0.03392**	0.04573	0.04508
19	Sonar	**0.03889**	0.11667	0.11111
20	Digits	**0.00278**	0.01528	0.0125
21	Ozone	**0.04463**	0.05593	0.05876
22	Insurance company coil2000	**0.05682**	0.06731	0.06469
23	Hill valley	**0.04245**	0.41509	0.43868
24	BCI	**0.18**	0.28667	0.33333
25	Musk	**0.06087**	0.06957	0.10870
	Average error	0.10468	0.15520	0.15174

Table 2: Average cross-validation error of different ensemble algorithms on each of the 25 real datasets from the UCI machine learning repository. The method with the minimum error is shown in bold.

	AB	RF
T-K PB	0.00002	0.0005
AB		0.58232

Table 3: P-values of Wilcox rank test(two-tailed test) between all pairs of methods.

5 Discussion

T-K PB finds the top-k parameter values to the given dataset. We add these top-k values such that its sum is not greater than one, then we take the sign of the sum of T parametrized weighted classifiers to make prediction. This approach is better than AB and RF(results shown in Table 2 and Table 3).

In this study we varied β for T-K PB in the cross validation to obtain the top-k parameter values. We chose the standard decision trees as classifiers in AB and our method T-K PB. We combined T decision trees to construct the final classifier for T-K PB and AB where $T = 500$. In the current experiments T-K PB is the slowest method but still computationally tractable for large datasets. However, T-K PB achieves better accuracy than the other two methods most of the time.

We chose RF [7] to be compared with our method due to its stability and its popularity as a powerful method. We used the standard package for RF in R [9]

6 Conclusion

We introduce a parametrized method of AdaBoost and optimize it with cross-validation for the classification. Our method outperforms the other popular methods by giving the minimum average error with statistical significance on many real datasets selected from UCI machine learning repository.

References

1. Opitz, D., Maclin, R.: Popular ensemble methods: An empirical study. Journal of Artificial Intelligence Research **11** (1999) 169–198
2. Freund, Y., Schapire, R.E.: A desicion-theoretic generalization of on-line learning and an application to boosting. In: Computational learning theory, Springer (1995) 23–37
3. Pang-Ning, T., Steinbach, M., Kumar, V., et al.: Introduction to data mining. In: Library of Congress. (2006)
4. Mohri, M., Rostamizadeh, A., Talwalkar, A.: Foundations of machine learning. MIT press (2012)
5. Japkowicz, N., Shah, M.: Evaluating Learning Algorithms. Cambridge University Press (2011)
6. A. Asuncion, D.N.: UCI machine learning repository (2007)
7. Breiman, L.: Random forests. Machine learning **45**(1) (2001) 5–32
8. Kanji, G.K.: 100 Statistical Tests. Sage Publications Ltd (1999)
9. Liaw, A., Wiener, M.: Classification and regression by randomforest. R News **2**(3) (2002) 18–22