A Novel Classifier Selection Approach for Adaptive Boosting Algorithms

ABM Shawkat Ali and Tony Dobele

School of Computing Sciences
Central Queensland University
Rockhampton, QLD 4702, Australia
E-mail: {s.ali,t.dobele}@cqu.edu.au

Abstract

Boosting is a general approach for improving classifier performances. In this research we investigated these issues with the latest Boosting algorithm AdaBoostM1. A trial and error classifier feeding with the AdaBoostM1 algorithm is a regular practice for classification tasks in the research community. We provide a novel statistical information-based rule method for unique classifier selection with the AdaBoostM1 algorithm. The solution also verified a wide range of benchmark classification problems.

1. Introduction

Boosting is a well established method in the machine learning community for improving the performance of any learning algorithm. The boosting algorithms were first presented by Schapire [1] and Freund [2]. Continuing this research they introduced a new generation of boosting algorithm called Adaptive Boosting (AdaBoost) [3-7]. Schapire and Freund argued that this new boosting algorithm has certain properties which make it more practical and easier to implement than its predecessors [8]. In this paper we investigate this issue within a wide range of classification problems.

During the AdaBoost performance testing we noted that classifier feeding with the AdaBoost algorithm is a trial and error approach. It is a lengthy process to find out a suitable classifier for a specific problem. Our present research provides a statistical information-based classifier selection for the AdaBoost algorithm. The rest of the paper is organized as follows: first we provide a brief description of boosting, AdaBoostM1 and some popular base classifiers. After that we summarize the experimental outcome of our current research. We conclude our research with a discussion of the limitations and future prospects of our research.

2. Algorithm description and experimental setup

In boosting, a relatively weak learning algorithm is repeatedly applied to a set of training data, with classifiers being produced at each iteration. These classifiers are finally combined into one composite classifier. Provided that the algorithm can produce classifiers that are at least slightly better than random guessing, boosting can significantly reduce the classification error rate [9].

With AdaBoost, which can be used to boost a wide range of learning algorithms, data instances are weighted after each iteration. AdaBoost is adaptive in that the weight of an instance is increased when the classification is incorrect. AdaBoost may be susceptible to noisy data, but is less susceptible to overfitting than most other learning algorithms. It can identify outliers which are usually the instances with the greatest weight [10].

AdaBoost.M1 and AdaBoost.M2 were developed by Schapire and Freund from their AdaBoost algorithm. For binary classification problems the two versions are equivalent, differing only in the way they handle problems with more than two classes [9].

AdaBoost.M1 has access to a learning algorithm (which the developers generically title WeakLearn) which it calls repeatedly with distributions over the training set. WeakLearn calculates a hypothesis, or classifier, that attempts to correctly classify all instances of the test data. As described previously, examples that are incorrectly classified are given

6th IEEE/ACIS International Conference on Computer and Information Science (ICIS 2007)
0-7695-2841-4/07 $25.00 © 2007 IEEE

greater weighting for the next pass. Finally, the boost algorithm combines all the hypotheses into one final hypothesis [9].

We chose four classifiers, namely J48, DecisionStump, NaïveBayes and PART as a base classifier for the AdaBoostM1 algorithm. All these classifiers are available in the WEKA [11] implementation. WEKA is a Java based machine learning tools. The DecisionStump classifier is a default setting with AdaBoostM1 in WEKA. The following section provides a brief explanation about the base classifiers.

J48: J4.8 is a supervised learning algorithm which induces a decision tree. It was developed by the developers of the Weka package (please see below) and is based on the widely-used C4.5 algorithm developed by J.R.Quinlan [11]. A decision tree is a tool for carrying out classification of data instances input to it. Decision trees have production rules of the type IF – THEN (IF feathers = 'yes' THEN Animal = 'bird'). Data are input firstly to the root node, which typically has a binary output, leading to two edges ('branches'). Which edge is followed depends on the answer to the condition posed in the node. Each edge may lead to a subsequent node with further outputs. Finally, the choices lead to a 'leaf' node which definitively classifies the instance into one class or another with no further choices [12].

DecisionStump: A DecisionStump is a simplified Decision Tree, having just one level. Usually a weak classifier, decision stumps are commonly used with a boosting algorithm to markedly increase classification accuracy. Interestingly, missing values are simply treated as a third class [11].

NaïveBayes: Naïve Bayes is based on the well-known Bayes Theorem. It is termed 'naïve' because it assumes that attributes of the training set are conditionally independent and that the prediction procedure is not influenced by any hidden or latent attributes. It works by calculating the maximum posterior probability of each class [13, 14].

PART: Part is developed from the C4.5 and RIPPER algorithms and is a partial decision tree algorithm. However, unlike C4.5 and RIPPER, PART does not have to perform global optimization in order to generate rules [14].

We fed all these base algorithms with AdaBoostM1 to classify a wide range of problems. First, we fed each classifier one-by-one with AdaBoostM1 and kept a record of the classification performance for the 113 problems. We selected all data sets from two different data repositories [15, 16]. All classification problems descriptions are available in Appendix I. We chose ten-fold cross validation over the experiment. Then we collected the descriptive statistical information about each of the 113 classification problems. The list of descriptive statistics is follows:

Statistical Name	**Symbolic Name**
Geometric mean	*geomean*
harmonic mean	*harmmean*
statistical mean	*sm*
median	*me*
trim mean,	*trimmean*
inter quartile range	*iqr*
mad	*ma*
range	*r*
standard deviation	*std*
variance	*v*
prctile	*p*
chi-square cumulative distribution	*chiscdf*
normal cumulative distribution	*normcdf*
skewness	*s*
kurtosis	*k*
correlation coefficient	*cc*
Z-score	*z*

The explanations of these descriptive statistical terms are available in any statistical text book. Moreover one can find the implementation in the Statistical Toolbox in Matlab [17].

We constructed a data matrix with these statistical algorithms and the name of the best algorithm performance. Then we employed the C5.0 [18] algorithm to generate the rules. These rules have been considered to select a unique classifier for AdaBoostM1 algorithms to classify any problem with better accuracy and faster computation.

3. Experimental results

We observed from the experiment that the J48 classifier is the best choice for the AdaBoostM1 algorithm and it shows the highest percentage of average accuracy for the 113 problems. However, in terms of computational complexity DecisionStump is the best choice among the four classifiers.

The rules were generated using the C5.0 decision tree algorithm to select a unique classifier for the AdaBoostM1 algorithm. C5.0 has two parameters, pruning confidence (c) and minimum cases (m). Pruning confidence affects error estimation and therefore how severely the tree may be pruned; a smaller value of c enables more pruning and a higher value less pruning. Minimum cases affect how the tree fits the data; a higher value of m allows more pre-pruning. The value of m should be at least two for

6th IEEE/ACIS International Conference on Computer and Information Science (ICIS 2007)
0-7695-2841-4/07 $25.00 © 2007 IEEE

every node in the tree [14]. We tuned both parameters to produce the best rule. The generated rules were verified by ten-fold cross validation and the percentage of accuracy is summarized with the rules. These rules are as follows:

3.0.1 Rules for J48 Classifier

Rule: IF Z > 71.932 OR harmmean <= 36.87 and s > 1.6085 OR harmmean <= 36.87 and p > 18.2 OR trimmean > 57.731 THEN select J48 Classifier for AdaBoostM1 Algorithm.

Rule Accuracy = 85.72%

3.0.2 Rules for DecisionStump Classifier

Rule: IF s <= 1.6085 and z > 40.276 OR geomean <= 0.4649 and trimmean > 0.88472 and s <= 1.6085 OR geomean > 117.06 OR *ma* <= 0.96388 and k <= 5.2902 OR trimmean <= 57.731 and s <= 1.6085 THEN select DecisionStump Classifier for AdaBoostM1 Algorithm.

Rule Accuracy = 81.24%

3.0.3 Rules for NB Classifier

Rule: IF trimmean <= 57.731 and normcdf <= 0.84223 and s > 0.77518 and s <= 1.736 and Z > 18.205 OR geomean <= 8.97 and iqr <= 2.2064 and normcdf > 0.84223 and *cc* > 0.17111 OR trimmean <= 57.731 and s <= 1.736 THEN select NB Classifier for AdaBoostM1 Algorithm.

Rule Accuracy = 80%

3.0.4 Rules for PART Classifier

Rule: IF trimmean > 81.997 and trimmean <= 98.664 OR harmmean > 5.0671 and trimmean <= 98.664 and *iqr* > 14.778 and s <= 13.596 OR geomean <= 117.06 and trimmean > 137.04 OR r <= 0.652 and s > 1.7405 OR iqr <= 14.778 and normcdf > 0.85796 THEN select PART Classifier for AdaBoostM1 Algorithm.

Rule Accuracy = 70.59%

4. Conclusions

This research contributes a new approach to selecting a unique classifier for the AdaBoostM1 algorithm. A rule based approach has been introduced for the unique classifier selection. These rules are generated based on described statistical information of 113 classification problems. All generated rules showed higher accuracy during the ten-fold cross validation except for the PART algorithm. The PART algorithm showed the best classification performance for only a few data sets. This performance could be increased by considering more classification problems. We have planned to extend our research using more problems from different domains with a variety of classifiers.

5. References

[1] R. E. Schapire, "The strength of weak learnability," Machine Learning, vol. 5, no. 2 pp 197–227, 1990.
[2] Y. Freund, "Boosting a weak learning algorithm by majority," Information and Computation, vol.121 no. 2 pp 256–285, 1995.
[3] A. Grove and D. Schuurmans, "Boosting in the limit: Maximizing the margin of learned ensembles," in Proceedings of the Fifteenth National Conference on Artifical Intelligence 1998, pp. 692–699.
[4] G. Ratsch, T. Onoda and K. Muller, "Regularizing AdaBoost," Advances in Neutral Information Processing Systems 11. 1999.
[5] G. Ratsch, T. Onoda and K. Muller, "Soft Margins for AdaBoost," Machine Learning, 42, pp287–320, 2000.
[6] G. Lebanon and J. Lafferty, "Boosting and maximum likelihood for exponential models," Neural Information Processing Systems (NIPS). 2001.
[7] R. Jin, Y. Liu, L. Si, J. Carbonell and A. Hauptmann, "A New Boosting Algorithm Using Input-Dependent Regularizer," in Proceedings of Twentieth International Conference on Machine Learning (ICML'03) 2003.
[8] Y. Freund and R. E. Schapire, "A decision-theoretic generalization of online learning and an application to boosting", Unpublished manuscript available electronically (on our web pages, or by email request). An extended abstract appeared in *Computational Learning Theory: Second European Conference, EuroCOLT '95*, pp 23–37, Springer-Verlag, 1995.
[9] Y. Freund and R.E. Schapire, "Experiments with a new boosting algorithm," In the Proceeding of International Conference on Machine Learning, pp 148–156, 1996.
[10] P. Radeva, Theoretical analysis of AdaBoost, online, accessed 15 February 2007. Available at: http://iua-share.upf.es/wikis/seminaris/images/5/59/Adaboost.pdf
[11] I.H. Witten and E. Frank, *Data Mining: Practical Machine Learning Tools and Techniques*, Morgan Kaufmann, 2005.
[12] P. Tan, M. Steinbach and V. Kumar, *Introduction to Data Mining*, Addison-Wesley, 2006.
[13] G.H. John and P. Langley, "Estimating continuous distributions in Bayesian classifiers," in Proceedings of the Eleventh Conference on Uncertainty in Artificial Intelligence, Morgan Kaufmann, San Mateo, CA, pp. 338–345, 1995.
[14] S. Ali and K.A. Smith, "On Learning Algorithm Selection for Classification," Applied Soft Computing, Elsevier Science, vol.6, no. 2, pp.119-138, January 2006.
[15] C. Blake and C.J. Merz, UCI Repository of machine learning databases, University of California, Irvine, CA, 2002. http://www.ics.uci.edu/mlearn/ mlrepository.html.

[16] T. S. Lim, Knowledge Discovery Central datasets, 2002, http://www.kdcentral.com/.
[17] Statistics Toolbox User's Guide, version 3, The MathWorks Inc., USA, 2001. URL: http://www.mathworks.com.
[18] See 4: http://www.rulequest.com/see5-info.html, Accessed: 02 February 2007.

Appendix I: Datasets description.

# Data set	Data set name	# Instances	# Attributes	# Class
1	abalone	1253	9	3
2	adp	1351	12	3
3	adult+stret	20	5	2
4	adult-stret	20	5	2
5	allbp	840	7	3
6	ann1	1131	7	3
7	ann2	1028	7	3
8	aph	909	19	2
9	art	1051	13	2
10	australian	690	15	2
11	balance-sca	625	5	3
12	bcw	699	10	2
13	bcw_noise	683	19	2
14	bld	345	7	2
15	bld_noise	345	16	2
16	bos	910	14	3
17	bos_noise	506	26	2
18	breast-canc	286	7	2
19	breast-canc	699	10	2
20	bupa	345	7	2
21	c	1500	16	2
22	cleveland-heart	303	14	5
23	cmc	1473	10	3
24	crx	490	16	2
25	dar	1378	10	5
26	dhp	1500	8	2
27	DNA-n	1275	61	3
28	dna	2000	61	3
29	dna_noise	2000	81	3
30	dph	590	11	2
31	echocardiogram	131	8	2
32	flare	1389	11	2
33	german	1000	25	2
34	glass	214	10	6
35	h-d	303	14	2
36	hayes-roth	132	6	3
37	hea	270	14	2
38	hea_noise	270	21	2
39	heart	270	14	2
40	hepatitis	155	20	2
41	horse-23	368	23	2
42	horse-colic	368	28	2
43	house-votes-84	435	17	2
44	hyp	2847	16	2
45	hypothyroid	1265	26	2
46	iris	150	5	3
47	khan	1063	6	2
48	kr-vs-kp	1279	37	2
49	labor-neg	40	17	2
50	led-noise	1047	10	10
51	lenses	24	6	3
52	letter-a	1334	17	2
53	lung-cancer	32	57	2
54	lymphography	148	19	8
55	mha	1269	9	4
56	monk1	556	7	2
57	monk2	601	7	2
58	monk3	554	7	2
59	mushroom	1137	12	2
60	nettalk_str	1141	8	5
61	page-blocks	1149	11	5
62	pendigits-8	1399	17	2
63	pha	1070	10	5
64	phm	1351	12	3
65	phn	1500	10	2
66	pid	532	8	2
67	Pima	768	9	2
68	poh	527	12	2
69	post-operative	90	9	3
70	primary-tum	339	18	2
71	pro	1257	13	2
72	promoter	106	58	2
73	pvro	590	19	2
74	rph	1093	9	2
75	satimage	1351	11	6
76	shuttle-landing control	15	7	2
77	sick-euthyroid	1582	16	2
78	sma	409	8	4
79	smo	1855	9	2
80	smo_noise	1855	16	2
81	sonar	208	61	2
82	splice	1589	61	3
83	t_series	62	3	2
84	tae	151	6	3
85	tae_noise	151	11	2
86	thy	1887	22	3
87	thynoise	1132	11	3
88	tic-tac-toe	958	10	2
89	titanic	2201	4	2
90	tmris	100	4	2
91	tqr	1107	12	2
92	trains-transformed	10	17	2
93	va-heart	200	9	4
94	veh	846	19	4
95	veh_noise	761	31	4
96	vehicle	658	20	0

IEEE COMPUTER SOCIETY

97	votes_noise	391	31	2
98	waveform	5000	22	2
99	waveform_noise	5000	41	2
100	wdbc	569	31	2
101	wine	178	14	3
102	wpbc	199	34	2
103	xaa	94	19	4
104	xab	94	19	4
105	xac	94	19	4
106	xad	94	19	4
107	xae	94	19	4
108	xaf	94	19	4
109	xag	94	19	4
110	xah	94	19	4
111	xai	94	19	4
112	yha	1601	10	2
113	zoo	101	17	7

0-7695-2841-4/07 $25.00

IEEE COMPUTER SOCIETY