Mach Learn (2007) 69: 35–53
DOI 10.1007/s10994-007-5020-z

Classifying under computational resource constraints: anytime classification using probabilistic estimators

Ying Yang · Geoff Webb · Kevin Korb · Kai Ming Ting

Received: 20 September 2006 / Revised: 20 September 2006 / Accepted: 9 July 2007 /
Published online: 19 September 2007

Abstract In many online applications of machine learning, the computational resources available for classification will vary from time to time. Most techniques are designed to operate within the constraints of the minimum expected resources and fail to utilize further resources when they are available. We propose a novel anytime classification algorithm, anytime averaged probabilistic estimators (AAPE), which is capable of delivering strong prediction accuracy with little CPU time and utilizing additional CPU time to increase classification accuracy. The idea is to run an ordered sequence of very efficient Bayesian probabilistic estimators (single improvement steps) until classification time runs out. Theoretical studies and empirical validations reveal that by properly identifying, ordering, invoking and ensembling single improvement steps, AAPE is able to accomplish accurate classification whenever it is interrupted. It is also able to output class probability estimates beyond simple 0/1-loss classifications, as well as adeptly handle incremental learning.

Keywords Anytime learning · Anytime classification · Probabilistic prediction · Bayesian classifiers · Ensemble methods

1 Introduction

In many applications, learning must be performed under strong computational resource constraints. The available resources are typically fixed and computations often must be com-

Editor: David Page.

Y. Yang (✉) · G. Webb · K. Korb · K.M. Ting
Clayton School of Information Technology, Monash University, Wellington Road, Clayton, VIC 3800, Australia
e-mail: ying.yang@infotech.monash.edu.au

G. Webb
e-mail: geoff.webb@infotech.monash.edu.au

K. Korb
e-mail: kevin.korb@infotech.monash.edu.au

K.M. Ting
e-mail: kaiming.ting@infotech.monash.edu.au

pleted within strict elapsed time limits in order to be useful. In particular, interactive systems must complete their task within a few seconds in order to gain and retain user acceptance. Some of the many examples of such applications include information retrieval (Baeza-Yates and Ribeiro-Neto 1999), recommender systems (Resnick and Varian 1997), user modeling (Webb et al. 2001) and online fraud detection (Chan et al. 1999).

While the total available resources remain constant, typically the number of simultaneous jobs that must be processed by a system varies from time to time. Hence, the potential computational resources available per job will fluctuate. Further, the complexity of individual jobs may vary considerably, for example, due to differences in the amount of available information. However, the limits on acceptable elapsed processing time per job are likely to remain invariant.

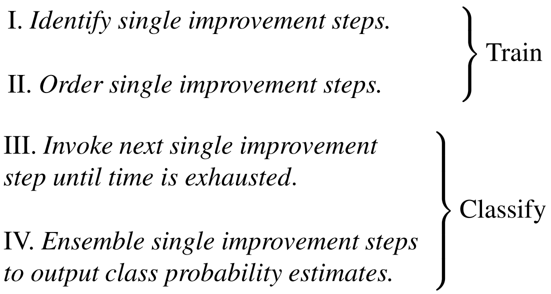
These constraints imply a complex optimization problem for the system designer. From anecdotal evidence, we believe that the standard response is to develop as efficient a learner as will provide acceptable accuracy (or whatever other performance metrics are sought to be optimized); to estimate or observe the resource requirements of this learner under peak loads; and then to provide the necessary resources for it to perform within the required elapsed time constraints under peak loads. We believe that this is one of the reasons for the popularity of naive Bayes (NB) (Langley et al. 1992; Mitchell 1997) in many such online applications. NB provides relatively high accuracy with low computational requirements (Lewis 1998).

Under such a scenario, it is clear that the available computational resources will be under-utilized outside peak periods. As a result, less accurate classification may occur than might be achievable within the available resources if a less efficient but more accurate learner were employed. Accordingly we propose the anytime averaged probabilistic estimator (AAPE), an algorithm that addresses the issue of utilizing such varying and uncertain resources to improve classification. AAPE invokes an ordered sequence of very efficient Bayesian probabilistic estimators, particularly naive Bayes (NB) (Langley et al. 1992; Mitchell 1997) and superparent-one-dependence estimators (SPODEs) (Keogh and Pazzani 2002), until time runs out. The classification can stop anywhere in the ordered sequence of SPODEs. The probabilistic estimators used up to that point compose an ensemble which will carry out the classification.

2 Problem definition and road map

We address the problem of classification learning using varying and uncertain computational resources. We assume m attributes $(X_1 \dots X_m)$ and k class labels $(c_1 \dots c_k)$. The training data are a set of labeled instances, $D = \langle\langle y_1, \mathbf{x}_1\rangle, \dots, \langle y_t, \mathbf{x}_t\rangle\rangle$, where each $y_i \in \{c_1, \dots, c_k\}$ and each $\mathbf{x}_i$ is a vector of attribute values $\langle x_1, \dots, x_m\rangle$. The test data are a set of unlabeled instances, $D' = \langle\langle \mathbf{x}_1\rangle, \dots, \langle \mathbf{x}_t\rangle\rangle$. Classification needs to predict the class label of each test instance given the evidence collected from the training data.

The computation of classification learning occurs at two distinct times, training time and classification time. We consider four distinct computational resources, *training time* (tt), *training space* (ts), *classification time* (ct) and *classification space* (cs). We assume a budget set for each of these resources consisting of a *contract* and an *anytime* component (Bernstein et al. 2002). The former is specified in advance of computation and is guaranteed. The latter is unknown beforehand, and the classifier is only notified when it has been exhausted. At one extreme there may be no anytime budget, in which case the system is working within known resource constraints. At the other extreme there may be no contract budget, in which case the system might exhaust its resources at any time.



I. *Identify single improvement steps.*

II. *Order single improvement steps.*

Train

III. *Invoke next single improvement step until time is exhausted.*

IV. *Ensemble single improvement steps to output class probability estimates.*

Classify

Fig. 1 Anytime classification road map

The types of algorithms that are appropriate for tackling classification learning under resource constraints will differ radically as the contract budgets and expectations about the anytime budgets vary for each of the four computational resources. Rather than seeking to develop algorithms that are appropriate to any such configuration, we address here one specific dimension, the *classification time*. We believe this approach has wide practical application and is typical of many of the online applications in which NB is currently employed. In this context, constraints on training space and time are not a major issue, because models are only developed off-line infrequently, and then employed many times. Further, constraints on classification space do not apply to the model, because a single model can be shared between all jobs, and hence the constraints relating to the classification space for a particular job relate only to its additional space requirements. Finally, we assume that the contract portion of the classification time budget is sufficient to allow standard NB classification to be performed.

Our *road map* to achieving anytime classification is illustrated in Fig. 1. Identifying and ordering single steps take place in training time while invoking and ensembling single steps take place at classification time.

3 Identifying single improvement steps

The first task in creating an anytime classification system is to identify single improvement steps, each of which is a small computational task with respect to the work needed to solve the problem (Grass and Zilberstein 1996). The system invokes improvement steps until the available time is exhausted. In principle, to minimize the risk that a single step is interrupted, each step should be the smallest task possible.

In the context of this paper, the goal is to improve classification accuracy. Candidate single improvement steps are k-dependence estimators (Sahami 1996). A k-dependence estimator is a Bayesian probabilistic classifier where each attribute depends upon at most k other attributes in addition to the class. In order to abide by the 'smallest task possible' principle and considering that the computational expense increases while k grows bigger, we here only employ naive Bayes (0-dependence estimators) and superparent-one-dependence estimators (a restricted class of 1-dependence estimators). Besides, averaged one-dependence estimators (AODE) (Webb et al. 2005) have demonstrated that an ensemble of SPODEs can deliver very high classification accuracy. For convenience, we use the term *probabilistic estimators* (PE) to refer to both NB and individual SPODEs.

A naive Bayes classifier (NB) assumes that attributes are independent of each other given the class. It utilizes Bayes formula to estimate $P(y|\mathbf{x})$, the probability of each class y given the instance to be classified, **x**. It assigns the class with the highest estimated probability to **x**. Note that (2) follows from (1) under NB's attribute independence assumption.

$$\hat{P}(y|\mathbf{x}) = \frac{\hat{P}(y)\hat{P}(\mathbf{x}|y)}{\hat{P}(\mathbf{x})} \tag{1}$$

$$= \frac{\hat{P}(y)\prod_{i=1}^{m}\hat{P}(x_i|y)}{\hat{P}(\mathbf{x})} \tag{2}$$

where $\hat{P}(\cdot)$ denotes an estimate of $P(\cdot)$. $\hat{P}(y)$ and $\hat{P}(x_i|y)$ are estimated from the frequency of the relevant terms in the training data, with possible corrections for sampling error such as the Laplace correction. $\hat{P}(\mathbf{x})$ is invariant across classes and hence need not be estimated if one only seeks to identify $\mathrm{argmax}_y(P(y|\mathbf{x}))$.

A superparent-one-dependence estimator (SPODE) (Keogh and Pazzani 2002) also estimates $P(y|\mathbf{x})$, and then chooses the class $\mathrm{argmax}_y(P(y|\mathbf{x}))$ as the class of **x**. However, it assumes that attributes are independent of each other given a *common* attribute (the superparent) and the class. Expressing X_p's value in **x** as x_p, a SPODE with superparent X_p estimates $P(y|\mathbf{x})$ as:

$$\hat{P}(y|\mathbf{x}) = \frac{\hat{P}(x_p, y)P(\mathbf{x}|x_p, y)}{\hat{P}(\mathbf{x})} \tag{3}$$

$$= \frac{\hat{P}(x_p, y)\prod_{i=1}^{m}\hat{P}(x_i|x_p, y)}{\hat{P}(\mathbf{x})}. \tag{4}$$

As in NB's case, required probabilities can be estimated from corresponding frequencies from the training data, with possible corrections for sampling error such as the Laplace correction. Note that (4) follows from (3) under SPODE's weaker attribute independence assumption. $\hat{P}(\mathbf{x})$ need not be estimated if one only seeks to identify $\mathrm{argmax}_y(P(y|\mathbf{x}))$. A set of training data with m attributes can produce m SPODEs, each of which takes a different attribute as superparent.

4 Ensembling single improvement steps to classify an instance

Suppose when the *ct* budget was exhausted, the anytime classification system has included $k \in [1, m+1]$ single steps, among which the first one is always NB and the remaining are SPODEs. The classification terminates by returning the average probabilities of $P(y|\mathbf{x})$ for all PEs completed. This follows the practice of the ensemble algorithm of AODE (Webb et al. 2005) and is computed as follows.

Assume that I denotes a subset of SPODEs and $|I|$ denotes the number of SPODEs in I. Since (4) holds for any SPODE, it also holds for I, giving:

$$\hat{P}(y|\mathbf{x}) = \frac{\sum_{p\in I}\hat{P}(x_p, y)\hat{P}(\mathbf{x}|x_p, y)}{\hat{P}(\mathbf{x}) \times |I|}$$

$$= \frac{\sum_{p\in I}\hat{P}(x_p, y)\prod_{i=1}^{m}\hat{P}(x_i|x_p, y)}{\hat{P}(\mathbf{x}) \times |I|}. \tag{5}$$

Averaging NB as in (2) and SPODEs as in (5), we get:

$$\hat{P}(y|\mathbf{x}) = \frac{\hat{P}(y)\hat{P}(\mathbf{x}|y) + \sum_{p\in I}\hat{P}(x_p, y)\hat{P}(\mathbf{x}|x_p, y)}{\hat{P}(\mathbf{x}) \times (|I| + 1)}$$
$$= \frac{\hat{P}(y)\prod_{i=1}^{m}\hat{P}(x_i|y) + \sum_{p\in I}\hat{P}(x_p, y)\prod_{i=1}^{m}\hat{P}(x_i|x_p, y)}{\hat{P}(\mathbf{x}) \times (|I| + 1)}. \tag{6}$$

Equation (6) averages probability estimates of an ensemble of PEs to classify a test instance. *That is how the anytime system ensembles single improvement steps' contribution whenever it is required to finish.* We name this procedure averaging probabilistic estimators (APE). Another way to ensemble PEs' individual verdicts is to follow a bagging approach that votes among PEs' predicted classes. We prefer the former to the latter because the former can output class probability estimates beyond simple 0/1 loss classifications.

5 Ordering single improvement steps

Given a data set involving m attributes, there are $m + 1$ single improvement steps: one NB, and m SPODEs, each of which takes a different attribute as superparent. It is desirable that an anytime classification system has *run-time monotonicity*, that is, the classification accuracy is a nondecreasing function of time. It seems reasonable to assume that the available SPODEs may be of varying quality. In an anytime system a varying number of improvement steps might be taken. It will be desirable to take the best steps first. These requirements point to the use of an ordering mechanism over the SPODEs such that the most beneficial may be applied first.

Seven ordering metrics are studied here, including information-theoretic metrics and accuracy-based empirical metrics. They are random (RAN), frequency (FEQ), minimum description length (MDL), minimum message length (MML), cross validation (CV), backward sequential elimination (BSE) and forward sequential addition (FSA).

5.1 Random (RAN)

RAN randomly chooses a SPODE and adds it into the ensemble. RAN has low computational overhead and offers a useful comparator against which to judge the impact on classification error of ordering PEs.

5.2 Frequency (FEQ)

FEQ orders SPODEs upon receiving each instance to be classified. Based on the specific values x_i of the attributes in that instance, FEQ counts $\mathit{frequency}(x_i)$, the number of times x_i occurs in training data. The SPODE whose superparent has the highest frequency is ordered to be first, and so on so forth.

5.3 Information-theoretic metrics (MDL and MML)

Information-theoretic metrics provide a combined score for a proposed explanatory model and for the data given the model: $I(D|h) + I(h)$. Particularly here, h is a SPODE and D is the training data. They aim to find a balance between goodness of fit (minimizing $I(D|h)$)

and model simplicity (minimizing $I(h)$), and thereby to achieve good modeling performance without overfitting the data. Two representative metrics are studied here: minimum description length[1] (MDL) (Suzuki 1996) and minimum message length (MML) (Korb and Nicholson 2004). The best score is the smallest. Hence the lower the score a SPODE gets, the higher its priority to appear in the ensemble.

5.4 Cross validation (CV)

CV scores each individual SPODE with superparent X_p by its error on leave-one-out cross validation in the training data. Given a SPODE, CV loops through the training data n times, each time training the SPODE from $(n-1)$ instances and using it to classify the remaining 1 instance. The misclassifications are summed and averaged over n iterations. The resulting classification error rate is taken as the metric value of the SPODE. The lower the error a SPODE gets, the higher its priority to appear in the ensemble. This cross validation process is very efficient as the model need only be updated for each instance that is left out, rather than recalculated from scratch.

5.5 Backward sequential elimination (BSE)

Inspired by the backward sequential elimination strategy for attribute selection in NB (Langley and Sage 1994), backward sequential elimination starts out with a full ensemble including NB and every SPODE. It then uses hill-climbing search to iteratively eliminate SPODEs whose individual exclusion results in the lowest classification error. In each iteration, suppose the current ensemble is E_{current} involving k SPODEs. BSE eliminates each member SPODE in turn from E_{current} and obtains an ensemble E_{test} with $(k-1)$ SPODEs. It then calculates the leave-one-out error of E_{test}. The ensemble E_{test} that yields the lowest error is selected and the corresponding eliminated SPODE is permanently deleted from the ensemble. The same process is applied to the new PE ensemble and so on, until only NB is left in the ensemble. The reverse of the order of elimination produces an order for SPODEs to be employed.

5.6 Forward sequential addition (FSA)

Inspired by the forward sequential selection strategy for attribute selection in NB (Langley and Sage 1994), forward sequential addition begins with an ensemble containing only NB. It then uses hill-climbing search to iteratively add SPODEs whose individual inclusion results in the lowest classification error. In each iteration, suppose the current ensemble is E_{current} with k SPODEs. FSA in turn adds each candidate SPODE, one that has not been included into E_{current}, and obtains an ensemble E_{test} with $(k+1)$ SPODEs. It then calculates the leave-one-out error of E_{test}. The E_{test} who obtains the lowest error is retained and the corresponding added SPODE is permanently included into the ensemble. The same process is applied to the new PE ensemble and so on, until every SPODE has been included. The order of addition produces an order for SPODEs to be employed.

[1] Suzuki's MDL metric differs only by a constant factor from Akaike's information criterion (AIC) (Akaike 1974) and the Bayesian information criterion (BIC) (Schwarz 1978). In consequence, all three metrics will provide identical orders over the model. Hence, the analysis and evaluation of MDL here also suffice for AIC and BIC.

6 The anytime average probabilistic estimator (AAPE) algorithm

AAPE starts with computing the probability estimates for NB. One reason to do so is that SPODEs typically have higher variance than NB (Webb et al. 2005). Hence, especially for small training sets, there is a risk that the error will be higher than NB if computation is terminated after only very few SPODEs have been processed. Then AAPE computes the probability estimates of each SPODE in sequence, ordered as in Sect. 5, until time runs out. When the *ct* budget has been exhausted, it terminates and returns $P(y|\mathbf{x})$ calculated by (6). This AAPE algorithm is presented in Table 1. Lines 1 to 7 compute NB. It is assumed that this computation can be completed within the contract time budget and so interrupts are only enabled on Line 8. Line 10 starts the loop over the ordered set of qualified attributes I that are each the superparent for a SPODE to be invoked. Lines 12 to 17 compute the probability estimates TP for one of the SPODEs. Lines 19 to 22 update the average condi-

Table 1 The AAPE Algorithm

Algorithm: AAPE

Inputs:

- $\mathbf{x}$: the instance to be classified.
- I: a set of SPODEs ordered by alternative schemes studied in Section 5.
- $PP[c_1 \dots c_k]$: prior probability estimate for each class. This and the following estimates are based on the observed frequency in the training data with possible correction for sampling error such as a Laplace correction.
- $PP[c_1 \dots c_k, attvals]$: prior probability estimates for each class and attribute-value pair.
- $CP[c_1 \dots c_k, attvals]$: conditional probability estimates for each attribute value given each class.
- $CP[c_1 \dots c_k, superparentvals, attvals]$: conditional probability estimates for each attribute-value (the final index) given each class and each superparent value.

Outputs:

- $P[c_1 \dots c_k]$: the estimated probability of each class given $\mathbf{x}$.

```
1:  for $y := c_1 \dots c_k$ do
2:     $P[y] := PP[y]$
3:     for $i := 1 \dots m$ do
4:        $P[y] := P[y] \times CP[y, x_i]$
5:     end for
6:  end for
7:  $normalize(P)$
8:  on interrupt goto 25
9:  $count := 1$
10: for each $p \in I$ in turn do
11:    $count := count + 1$
12:    for $y := c_1 \dots c_k$ do
13:       $TP[y] := PP[y, x_p]$
14:       for $i := 1 \dots m$ do
15:          $TP[y] := TP[y] \times CP[y, x_p, x_i]$
16:       end for
17:    end for
18:    suspend interrupts
19:    for $y := c_1 \dots c_k$ do
20:       $P[y] := \frac{P[y] \times (count-1) + TP[y]}{count}$
21:    end for
22:    $normalize(P)$
23:    restore interrupts
24: end for
25: return $P$
```

tional probability estimate for each class to account for a new set of TP values. Interrupts are suspended during this process as the P values are in an unstable state. In consequence, if the time budget expires during this process there will be a slight delay before computation terminates.

7 Time complexity analysis

Assume that the number of training instances and attributes are n and m, and number of classes is k. Let the average number of values for an attribute be v.

7.1 Classification overhead

No matter what ordering scheme is applied to AAPE, the result is a linear combination of SPODEs. Hence, each scheme's complexity is of the same order $O(m^2k)$, resulting from an $O(mk)$ algorithm applied over an $O(m)$ sized ensemble.

One exception is FEQ, which is a lazy technique that determines the order of SPODEs at classification time. It involves sorting m attribute values on *frequency*(x_i), which incurs an additional time complexity $O(m \log m)$.

7.2 Training overhead

Note that as far as anytime classification is concerned, the training time complexity is not the key concern for AAPE. It is included here for completeness of analysis.

NB constructs a two-dimensional table indexed in one dimension by the class value and in the other by the attribute values. Each entry contains the frequency of the given class and attribute-value pair. Construction of this table requires iteration through each instance ($O(n)$), for each of which it iterates through each attribute ($O(m)$) to increment the frequency of that attribute's value and the instance's class value. This results in time complexity of $O(nm)$.

SPODE requires a three-dimensional table indexed in one dimension by the class value, in the second by the super-parent attribute's values, and in the other dimension by the remaining attribute values. Each entry contains the frequency of the given class, super-parent value and attribute-value tuple. Compilation of this table requires iteration through each instance ($O(n)$), for each of which it iterates through each attribute other than the super-parent ($O(m)$) to increment the frequency of that attribute's value and the instance's class and super-parent values. This also results in time complexity of $O(nm)$.

APE requires a three-dimensional table indexed in one dimension by the class value, in the second by all possible super-parent attribute's values, and in the final dimension by the remaining attribute values. Compilation of this table requires iteration through each instance ($O(n)$), for each of which it iterates through each potential super-parent ($O(m)$). For each potential super-parent it iterates through each remaining attribute to increment the frequency of that attribute's value and the instance's class and super-parent values. This results in time complexity of $O(nm^2)$.

RAN and **FEQ** do not require any additional information to be gathered in training time and hence have no impact on training-time.

MDL's complexity of calculating $I(D|h)$ is $O(mv^2k)$. The dominating part is from $H(X_i, \Phi(i))$ which iterates through each value ($O(v)$), and then each joint value of the superparent and the class ($O(vk)$). The complexity of calculating $I(h)$ is $O(m)$.[2] Since the selection repeats for each attribute ($O(m)$), the overall complexity is $O(m \times (mv^2k + m)) = O(m^2v^2k)$.

MML's dominating complexity is from calculating $\prod_{i=1}^{m+1} \prod_{j=1}^{|\phi_i|} \frac{(v_i-1)!}{(S_{ij}+v_i-1)!} \prod_{l=1}^{v_i} \alpha_{ijl}!$. MML iterates through each attribute ($O(m)$); and then each joint value of the superparent and the class ($O(vk)$) for which two factorials are calculated ($O(v) + O(\frac{n}{vk})$). On top of that it loops through each attribute value ($O(v)$) for which a third factorial is calculated ($O(\frac{n}{v^2k})$). Hence the complexity is $O(m * vk * (v + \frac{n}{vk}) * v * \frac{n}{v^2k}) = O(mn(v + \frac{n}{vk}))$. This repeats for each attribute ($O(m)$) and the overall complexity is hence $O(m^2n(v + \frac{n}{vk}))$.

CV classifies n training instances in turn to score a SPODE. To classify one instance, a SPODE will multiply the conditional probability of each attribute value given each class label and one (constant) superparent value. This results in $O(mk)$. To do leave-one-out cross validation, the classification will repeat n times. Hence the complexity is $O(mkn)$. This repeats for each attribute ($O(m)$) and the overall complexity is hence $O(m^2kn)$.

BSE's hill climbing procedure of reducing a PE ensemble of size $(m + 1)$ to 1 will render a complexity of $O(m^3)$. In the first round, it alternatively eliminates each of m SPODEs. In the second round, it alternatively eliminates each of $(m - 1)$ SPODEs. Following this line of reasoning, the total number of probing a SPODE is $m + (m - 1) + \cdots + 2 + 1 = O(m^2)$. As explained for CV, to test each SPODE by leave-one-out cross validation will incur complexity of $O(mkn)$. As a result, the overall complexity is $O(m^3kn)$.

FSA's hill climbing procedure of increasing a PE ensemble from size 1 to size $(m + 1)$ will render a complexity of $O(m^2)$. In the first round, it alternatively adds each of m SPODEs. In the second round, it alternatively adds each of $(m - 1)$ SPODEs. Following this line of reasoning, the total number of probing a SPODE is $m + (m - 1) + \cdots + 2 + 1 = O(m^2)$. As explained for CV, to test each SPODE by leave-one-out cross validation will incur complexity of $O(mkn)$. As a result, the overall complexity is $O(m^3kn)$.

8 Empirical observations, evaluations and analysis

Extensive experiments are conducted to test how effectively AAPE can utilize increasing classification time to improve classification accuracy, and how effectively AAPE can handle anytime interruption during its classification process.

8.1 Experimental design

The AAPE system is implemented in the WEKA machine learning environment (Witten and Frank 2005). It is tested using a large suite of 60 benchmark data sets from the UCI Machine Learning Repository (Blake and Merz 2004), as described in Table 2.

[2] Although MDL has an extra loop $\prod_{j\in\Phi(i)} v_j$, in case of a SPODE, $|\Phi(i)|$ is of maximum value 2 (the superparent and the class). Hence it can be treated as a constant and does not increase the order of the complexity.

Table 2 Each data set's name, number of instances (Ins.) and number of attributes (Att.)

Data Set	Ins.	Att.	Data Set	Ins.	Att.	Data Set	Ins.	Att.
Abalone	4177	8	HouseVotes84	435	16	Postoperative	90	8
Adult	97686	14	Hungarian	294	13	Promoter	106	57
AE	9961	12	Hypothyroid	3772	29	PrimaryTumor	339	17
Annealing	898	38	Ionosphere	351	34	Satellite	6435	36
Audiology	226	69	IrisClassification	150	4	Soybean	683	35
AutosImports85	205	25	KRvsKP	3196	36	Segment	2310	19
BalanceScale	625	4	LaborNegotiations	57	16	SickEuthyroid	3772	29
Bands	539	36	LED	1000	7	Sign	12546	8
BreastCancer	699	9	LetterRecognition	20000	16	Sonar	208	60
LiverDisorders	345	6	LungCancer	32	56	Splice	3177	60
Chess	551	39	Lymphography	148	18	Syncon	600	60
HeartDisease	303	13	Mfeat-mor	2000	6	Thyroid	9169	29
CMC	1473	9	Mushroom	8124	22	TicTacToe	958	9
CreditApproval	690	15	Musk	476	166	Vehicle	846	18
Echocardiogram	131	6	NewThyroid	215	5	Volcanoes	1520	3
German	1000	20	OpticalDigits	5620	48	Vowel	990	11
GlassIdentification	214	9	PageBlocks	5473	10	Waveform	5000	40
HeartCleveland	270	13	PenDigits	10992	16	Wine	178	13
Hepatitis	155	19	NetTalkPhoneme	5438	7	Yeast	1484	8
HorseColic	368	21	PimaDiabetes	768	8	Zoo	101	16

On each data set, a 60-trial[3] 2-fold cross validation[4] is conducted. Various aspects of AAPE's performance are recorded, including its *training efficiency* under different ordering schemes, its *classification error* in an anytime fashion, and its *classification bias* and *variance* decomposition.

We are interested in studying bias and variance because they each give different insights into AAPE's error (Breiman 1996; Friedman 1997; Kohavi and Wolpert 1996; Kong and Dietterich 1995; Webb 2000). Bias describes the component of error that results from systematic error of the learning algorithm. Variance describes the component of error that results from random variation in the training data and from stochastic behavior in the learning algorithm, and thus measures how sensitive an algorithm is to changes in the training data. We use Kohavi and Wolpert's (1996) definitions of bias and variance, and estimate them using Webb's (2000) cross-validation method.

Various statistics are employed to analyze the experimental results. To compare a pair of algorithms, we measure their *win/loss/tie* record and apply a binomial test on the record. The win, loss, tie each represent the number of data sets in which one algorithm beats, loses to or ties with the other respectively. A one-tailed binomial sign test can then be applied to

[3] The reason that we use 60 trials is because we perform bias-variance analysis, which is more accurate when more trails are conducted. Meanwhile, because a large number of data sets are involved, too many trials will make the whole evaluation process very expensive. 60 is selected as a compromise.

[4] A k-fold cross validation divides a data set into k equal-size subsets. Each subset is used in turn as a test set with the remaining $k-1$ data sets used for training. One may conduct k-fold cross validation for m trials, each trial shuffling the instances and forming k different subsets.

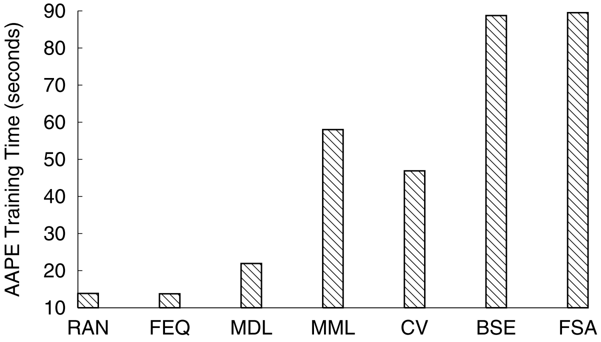


Fig. 2 Comparing AAPE's training efficiency under different ordering schemes

the wins and losses of the record. If its result is less than 0.05, the wins against losses are statistically significant, supporting the claim that the winner has a systematic (instead of by chance) advantage over the loser. To compare multiple algorithms, we use the Friedman test and Nemenyi test as recommended by Demsar (2006). These tests rank rival algorithms and indicate which ones have statistically significant differences.

Finally, as AAPE currently requires discrete-valued data, each training set is discretized using the WEKA MDL discretizer (Witten and Frank 2005). As the software cannot handle missing values when calculating information-theoretic metrics, we employ WEKA's imputation policy. All missing values for nominal and numeric attributes in a data set are replaced with the modes and means from the training data.

8.2 Training efficiency

Recall that AAPE can employ alternative schemes to order SPODEs, as studied in Sect. 5. Different schemes will demand different time. Figure 2 illustrates the training time of AAPE caused by each ordering algorithm averaged across 60 data sets. From the fastest to the slowest are FEQ, RAN, MDL, CV, MML, BSE and FSA. FEQ and RAN are efficient because they do not require any ordering process in training time. Empirical metrics (CV, FSA and BSE) are slower than theoretic MDL because they need to loop through training data for leave-one-out cross validation. One exception is MML whose complexity is higher than CV. Metrics that measure PE ensembles (FSA and BSE) are slower than those that measure individual SPODEs (MDL, MML and CV) because they need to probe different combinations of individual SPODEs.

8.3 Anytime classification accuracy

It is desirable that AAPE improve its classification accuracy as it is given more time (allowed to include more SPODEs). To test this issue, we conducted run-time monitoring on classification error offered by each ordering scheme. Their performance curves over time are drawn in Fig. 3. The X axis corresponds to the time resource in terms of the number of SPODEs allowed to be included in the ensemble with 0 meaning NB alone. The Y axis corresponds to AAPE's classification error averaged on all data sets. Note that the error on each data set is normalized by NB's error in order to permit meaningful averaging over multiple data sets. If X becomes bigger than the number of attributes (SPODEs) in a data set, for the purpose of drawing anytime classification curves, the Y value remains that of the full ensemble. In practice, this means that if the ct budget is more than what AAPE needs to finish, AAPE always returns its complete calculation.

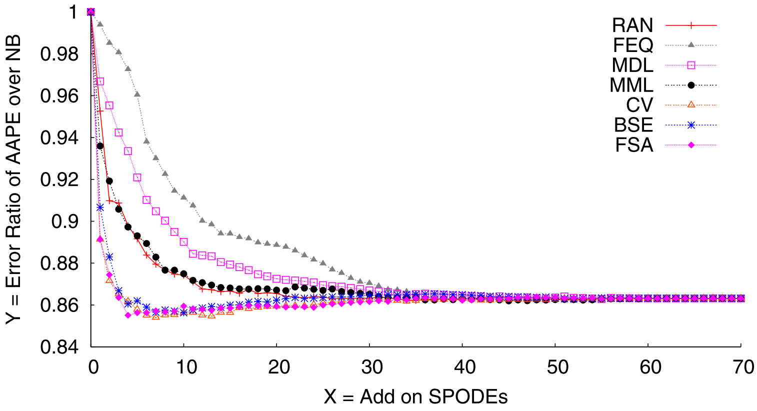


Fig. 3 AAPE's anytime classification accuracy under alternative ordering schemes

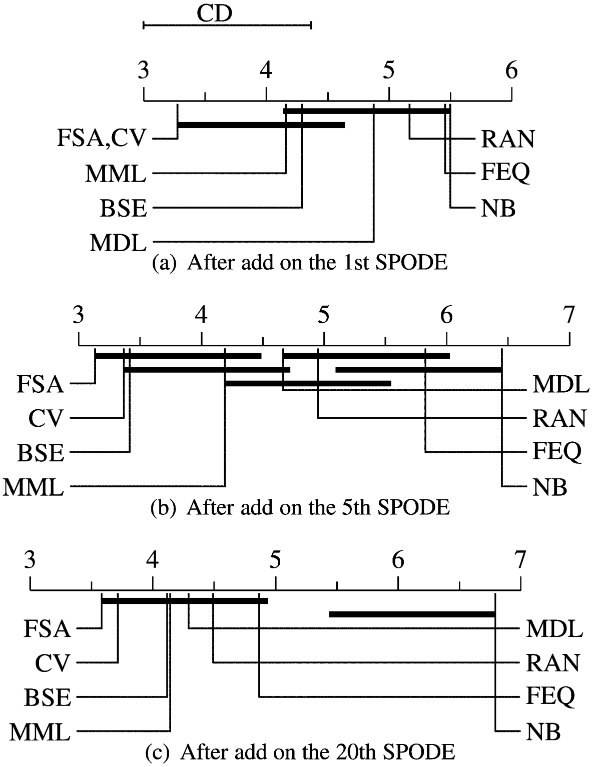
Every ordering scheme starts with NB, resulting in the same classification error at $X = 0$. A rapid decrease in error is witnessed at the early stages of adding SPODEs, where FSA is the most effective and FEQ is the least effective. While more and more SPODEs are included, the differences among alternative schemes shrinks. When all SPODEs are included, all schemes converge to the same classification error.

In addition to the mean error, at every X value in Fig. 3, as recommended by Demsar (2006), we rank alternative ordering schemes according to AAPE's classification error, and apply Friedman test and Nemenyi test to the ranks. Some representative results are visually presented in Fig. 4. The critical difference 'CD' is shown above the graph. In each sub-figure, the top line in the diagram is the axis on which we plot the average ranks of schemes. The more effective a scheme is, the lower rank it is assigned. When comparing all schemes against each other, we connect the groups of algorithms that are not significantly different at the critical level of 0.05. For example, Fig. 4(a) illustrates that when adding the first SPODE ($X = 1$), FSA and CV are equally best at reducing error (ranked lowest, 3.275), while NB is the worst (ranked highest, 5.5). FSA and CV are significantly better than MDL, RAN, FEQ and NB (tics not connected), while they do not have significant difference from MML and BSE (tics connected).

According to Figs. 3 and 4, in general, FSA and CV are the best schemes. Schemes that order SPODEs by their empirical strength in classification accuracy (CV, BSE and FSA) are more effective than schemes that order SPODEs by information theory (MDL and MML). It is interesting to spot that FEQ is slow at decreasing the error, and less effective than RAN. One reason might be that very high frequency attribute values contain relatively little information and hence are not the most useful values to utilize. It is also good to obtain experimental evidence that by employing appropriate ordering schemes such as CV and FSA, AAPE's error is bounded by NB which is an efficient, robust, low variance classifier.

Figure 3 also reveals an interesting issue that the full PE ensemble invoked for every data set ($X = 69$) actually does not produce the minimum average error. For example, FSA reaches its lowest mean error at $X = 7$ and climbs up a little while more SPODEs are included. This arises a question that whether AAPE should terminate earlier (even if the classification time resource is still available to invoke more single steps) in order to minimize the classification accuracy. The following section will study this issue.

Fig. 4 Using Friedman test and Nemenyi test to compare rival ordering algorithms' performance on reducing AAPE's classification error. The lower an algorithm is ranked, the better it is at error reduction. Algorithms that are not significantly different are connected



(a) After add on the 1st SPODE

(b) After add on the 5th SPODE

(c) After add on the 20th SPODE

8.4 When to stop

Currently, AAPE keeps invoking the next SPODE as long as classification time allows. The stopping point where AAPE terminates calculation and outputs classification, if time allows, is when AAPE has eventually included all SPODEs. It is natural to ask whether there exists any optimal stopping point E_{optima}, a subset of PEs, before AAPE reaches the full PE ensemble E_{full}. If so, once AAPE reaches E_{optima}, it can deliver classification, even when some SPODEs remain uninvoked and the *ct* budget yet to be exhausted. The potential advantages are in two-fold. First, E_{optima} can obtain lower error than E_{full} by excluding poorly predictive SPODEs. Second, it takes shorter time for AAPE to produce optimal accuracy. Figure 3 reveals that $X = 69$ (full PE ensemble invoked for every data set) actually does not produce the minimum error. For example, FSA reaches its lowest mean error at $X = 7$ and climbs up a little while more SPODEs are included. This indicates that better stopping points exist earlier than E_{full}.

To explore this issue, we implemented AAPE with two stopping criteria other than reaching the full PE ensemble. Recall that in Sect. 5, both FSA and BSE have natural stopping points: the ensemble $E_{\min}$ that achieves the lowest leave-one-out cross validation error in training during the addition or elimination process. For FSA, one can follow the addition order to include SPODEs until the ensemble reaches the set of SPODEs that delivered $E_{\min}$.

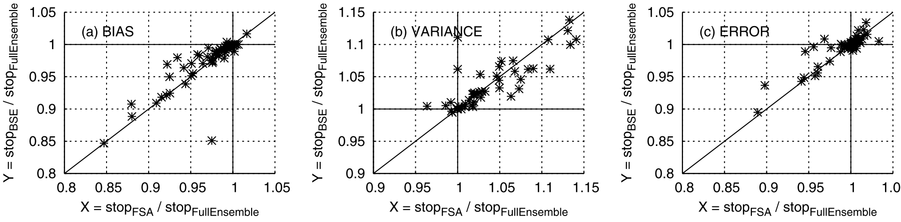


Fig. 5 Alternative stopping strategies' relative performance. $stop_{\mathrm{FSA}}$ and $stop_{\mathrm{BSE}}$ are better at reducing bias while $stop_{\mathrm{FullEnsemble}}$ is better at reducing variance. The end effect is that $stop_{\mathrm{FSA}}$ and $stop_{\mathrm{BSE}}$ cannot beat $stop_{\mathrm{FullEnsemble}}$ on reducing error

For BSE, one can follow the reverse of the elimination order to include SPODEs until the ensemble reaches the set of SPODEs that delivered $E_{\min}$.[5]

We tested AAPE's classification error (decomposed into bias and variance) under the two alternative stopping strategies, and compared them against that under the full PE ensemble as in Fig. 5. For simplicity, we name the three strategies $stop_{\mathrm{FSA}}$, $stop_{\mathrm{BSE}}$ and $stop_{\mathrm{FullEnsemble}}$ respectively. The values on the Y axis are the outcome for $stop_{\mathrm{BSE}}$ divided by that for $stop_{\mathrm{FullEnsemble}}$. The values of the X axis are the outcome for $stop_{\mathrm{FSA}}$ divided by that for $stop_{\mathrm{FullEnsemble}}$. Each point on the graph represents one of the 60 data sets. Points on the left of the vertical line at $X = 1$ in each subgraph are those of which $stop_{\mathrm{FSA}}$ outperforms $stop_{\mathrm{FullEnsemble}}$. Points below the horizontal line at $Y = 1$ indicate that $stop_{\mathrm{BSE}}$ outperforms $stop_{\mathrm{FullEnsemble}}$. Points below the diagonal line $Y = X$ represent that $stop_{\mathrm{FSA}}$ outperforms $stop_{\mathrm{BSE}}$. It is observed that on one hand, both $stop_{\mathrm{FSA}}$ and $stop_{\mathrm{BSE}}$ are more effective in reducing bias compared with $stop_{\mathrm{FullEnsemble}}$ as the majority of points fall within the boundaries $X = 1$ and $Y = 1$ in Fig. 5(a). On the other hand, $stop_{\mathrm{FullEnsemble}}$ is more effective in reducing variance than $stop_{\mathrm{FSA}}$ and $stop_{\mathrm{BSE}}$ as the majority of points fall beyond the boundaries $X = 1$ and $Y = 1$ in Fig. 5(b). The end effect is that neither of them can beat $stop_{\mathrm{FullEnsemble}}$ as $stop_{\mathrm{FSA}}$'s win/lose/tie record against $stop_{\mathrm{FullEnsemble}}$ is 23/34/3, and $stop_{\mathrm{BSE}}$'s is 27/30/3 in Fig. 5(c).

These observations suggest that although it is desirable to find alternative stopping points (if there is any) that exclude counter-productive SPODEs and enable AAPE to reach optimal classification faster, this task is non-trivial. Nonetheless, it remains an interesting topic for further research in our anytime learning systems.

8.5 CV and FSA

It is intuitive to expect that FSA should outperform CV. The former orders SPODEs according to the collective merit of a PE ensemble while the later measures SPODEs individually. However, Fig. 3 interestingly reveals that although FSA is the most effective at the early stages of adding SPODEs, CV actually outperforms FSA later on (although not statistically significantly).

To explore this issue, the relative performance between CV and FSA is illustrated in Fig. 6 along the time line when SPODEs are added on. The Y value is the number of data sets where FSA wins CV *minus* the number of data sets where CV wins FSA. Accordingly,

[5] In either case, if multiple ensembles attain the lowest error, the one that includes most SPODEs is chosen, as a means to reduce variance caused by model selection.

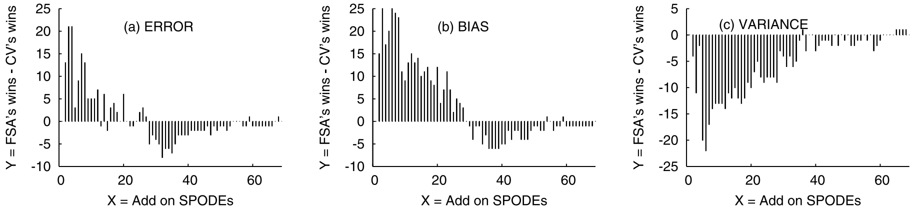


Fig. 6 Comparing FSA and CV. FSA orders SPODEs by their ensemble's collective merit and is most effective at early stages. CV orders SPODEs in isolation, which outperforms FSA later on

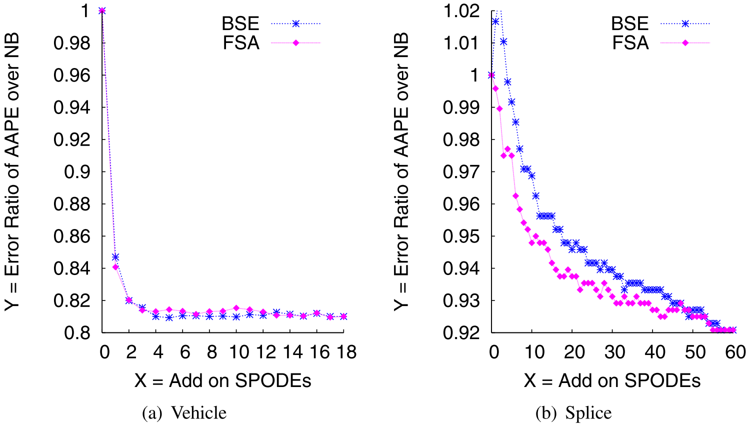
a positive Y value means that FSA outperforms CV more often than not, and a negative Y value means otherwise. The longer a bar, the bigger the win over the loss. It is observed that CV frequently outperforms FSA in term of reducing classification variance as the Y values are mostly negative in Fig. 6(c). This is because SPODEs are ordered independent of each other by CV, while FSA chooses the next SPODE on basis of already chosen ones. On the other hand, at early stages of adding on SPODEs, FSA usually achieves lower bias than CV,[6] but the advantage reduces as more SPODEs are added as in Fig. 6(b). This suggests that it is when a small number of SPODEs are involved that measuring collective merit is more influential. As a result, the curves of FSA and CV cross each other in Fig. 3, FSA being the constant winner initially while CV catches up and takes over at a later stage.

This observation is interesting because it suggests that a SPODE that achieves high accuracy in isolation is often also a valuable one to be included in an ensemble (at least not counter-productive). Furthermore, recall that in Fig. 2, CV makes AAPE much more efficient in training than FSA does. As a result, CV can be a very attractive ordering scheme for AAPE in practice.

8.6 BSE and FSA

Although previous work suggested that backward elimination tends to be more effective than forward addition for feature selection (Koller and Sahami 1996; Kohavi and John 1996; Wu and Urpani 1999), there is no clear-cut advantage to either algorithm in our case. Both of them use greedy search and have their own niches depending on the nature of the data. Figure 7(a) illustrates a representative data set 'Vehicle' where BSE outperforms FSA, while Fig. 7(b) illustrates another representative data set 'Splice' where FSA beats BSE. Nonetheless, as in Fig. 3, more often than not, AAPE using FSA ordering decreases classification error faster than using BSE ordering. We suggest one reason is that very often the biggest improvements of classification performance take place at the early stages, such as when adding the first few SPODEs. FSA optimizes those important starting points while BSE optimizes finishing points which are less critical. Besides, during BSE's proceeding from a full-ensemble to an empty ensemble, whenever it encounters ties among candidate SPODEs, it randomly picks one to eliminate. This strategy can make AAPE deviate from its optimal path and lead to a sub-optimal start. That's why BSE can perform worse than NB when adding first SPODEs as in Fig. 7(b). More sophisticated tie-breaker strategies might improve BSE's accuracy, however at the cost of increasing its training efficiency.

[6] Except when adding the first SPODE where FSA and CV are identical.



(a) Vehicle (b) Splice

Fig. 7 FSA and BSE each have its own niche. However, FSA is often more effective for AAPE because starting SPODEs often drop errors more effectively

9 Related research

Anytime neural networks (Grefenstette and Ramsey 1992; Opitz 1995) tackle the problem of constrained training time by forming an initial model which is then refined. Once the refinement phase has begun, it may be terminated at any time. While this approach might be expected to deliver learning under arbitrary training time constraints, in practice these are computationally intensive systems that are designed to run over a long time. They cannot produce models within time spans measured in seconds or less. They do not address the issue of varying resources at classification time.

Anytime interval-valued outputs for kernel machines (DeCoste 2002) utilize successive support vectors from an SVM to provide ever improving approximations to the final SVM classification. To classify an instance **X**, this approach seeks to optimize the quantity $d^2_{XN} - d^2_{XP}$, where d_{XN} and d_{XP} are the Euclidean distances from **X** to its negative neighbor **N** and positive neighbor **P**, respectively. This paper shows how the same principle can be applied to Bayesian classification with the desirable property that conditional probabilities are returned rather than just a simple selection of a single class.

Another anytime classification scheme was proposed during the search of the best superparent and its favorite child for a single SPODE (Keogh and Pazzani 2002). The search begins with NB. It iterates through each superparent and then every child until it finds the best parent-child pair that increases the classification accuracy most. Because the search is time-consuming and is sub-optimal when fast classification is required, this anytime scheme returns a best-so-far SPODE whenever the search is interrupted. In contrast, this paper searches an ordered sequence of SPODEs. It is an anytime algorithm in the sense that the classification can stop anywhere in the sequence. In addition this paper studies alternative stopping criteria to find the best PE ensemble instead of the best single SPODE.

10 Discussion and further research

Any ensemble learner could be converted into an anytime classifier by simply evaluating at classification time only as many of the available ensemble members as time allows. However, AAPE has a number of desirable properties that favor its use. First, each single step (a probabilistic estimator) itself is adept at incremental learning. They can be updated with new training data at negligible cost, simply by updating the table of joint frequencies. Second, the classifiers can return class probability estimates beyond simple 0/1-loss classifications. Third, AAPE's lowest classification accuracy appears to be bounded by NB which is an efficient, robust, low variance classifier.

AAPE tackles only one dimension of the complex inter-related set of potentially variable computational resources that we identified in Sect. 2. There is clear potential to extend AAPE in the future research.

- The AAPE framework could be extended to address variable training time. For example, if training time allows, AAPE can do feature selection that reduce the number of super-parents or children, which could potentially enhance AAPE's classification accuracy as well as efficiency.
- The AAPE framework could also be extended to accommodate varying space resources, for example, by altering the degree of dependence in single improvement steps (k-dependence estimators) in response to the space available to store the conditional frequency tables.
- In the current work we have assumed that the primary objective is to increase classification accuracy. In real-world applications of inductive learning, there are many different types of cost involved, such as cost of unwanted achievements or cost of intervention (Turney 2000). It would be valuable to consider anytime techniques that utilize varying computational resources to optimize different metrics that users care about.
- AAPE works in an NB-like framework. Lessons learned here could be extended to explore other learning paradigms such as TAN, decision trees and SVM.

11 Conclusion

We have argued that in many real-world applications, inductive inference is conducted under strict computational resource constraints. We have analyzed such situations, identifying four key resources: training space, training time, classification space and classification time.

We have also argued that many applications in which NB is currently deployed may have periods when the classification-time resources are under-utilized. We propose a novel classification algorithm, AAPE, which makes good use of idle classification time. In training time, AAPE identifies and orders NB and SPODEs (the single improvement steps). At classification time, AAPE first computes NB and then utilizes any additional time to refine the probability estimates by invoking SPODEs in order. Whenever it is interrupted, AAPE averages the class probability estimates calculated by invoked single steps and delivers them to the user. The technique of AAPE also lends itself to parallelization, with the possibility of utilizing a variable number of processors. Furthermore, AAPE can elegantly support incremental learning.

To verify AAPE's efficacy and efficiency, we have conducted extensive experiments, using 60 benchmark data sets from the UCI Machine Learning Repository (Blake and Merz 2004) as well as using multiple established statistical tests.

Both theoretical analysis and empirical evidence suggest that by properly identifying, ordering, invoking and ensembling single improvement steps, AAPE is able to effectively utilize increasing classification time to improve classification accuracy. In particular, we suggest that if the training time is not a concern, given a choice amongst the ordering metrics studied here, FSA is the most attractive. If the training time is limited, CV is the method of choice.

Acknowledgements We gratefully acknowledge Professor Janez Demsar for his kind help on our statistical tests. We also thank Professor Pat Langley for his thoughtful and constructive comments on this paper. This research was supported by Australian Research Council grant DP0556279.

References

Akaike, H. (1974). A new look at the statistical model identification. *IEEE Transactions on Automatic Control*, *19*, 716–723.

Baeza-Yates, R., & Ribeiro-Neto, B. (1999). *Modern information retrieval*. Reading: Addison–Wesley.

Bernstein, D. S., Perkins, T. J., Zilberstein, S., & Finkelstein, L. (2002). Scheduling contract algorithms on multiple processors. In *Proceedings of the 18th national conference on artificial intelligence and the 14th conference on innovative applications of artificial intelligence* (pp. 702–706).

Blake, C., & Merz, C. J. (2004). *UCI repository of machine learning databases*. [Machine-readable data repository]. Department of Information and Computer Science, University of California, Irvine, CA, USA.

Breiman, L. (1996). *Bias, variance and arcing classifiers* (Technical report 460). Berkeley: Statistics Department, University of California.

Chan, P., Fan, W., Prodromidis, A., & Stolfo, S. (1999). Distributed data mining in credit card fraud detection. *IEEE Intelligent Systems*, *14*(6), 67–74.

DeCoste, D. (2002). Anytime interval-valued outputs for kernel machines: fast support vector machine classification via distance geometry. In *Proceedings of the 19th international conference on machine learning* (pp. 99–106).

Demsar, J. (2006). Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research*, *7*, 1–30.

Friedman, J. H. (1997). On bias, variance, 0/1-loss, and the curse-of-dimensionality. *Data Mining and Knowledge Discovery*, *1*(1), 55–77.

Grass, J., & Zilberstein, S. (1996). Anytime algorithm development tools. In M. Pittarelli (Ed.), *SIGART Bulletin Special Issue on Anytime Algorithms and Deliberation Scheduling*, *7*(2), 20–27.

Grefenstette, J., & Ramsey, C. (1992). An approach to anytime learning. In *Proceedings of the 9th international machine learning workshop*.

Keogh, E. J., & Pazzani, M. J. (2002). Learning the structure of augmented Bayesian classifiers. *International Journal on Artificial Intelligence Tools*, *11*(40), 587–601.

Kohavi, R., & John, G. H. (1996). Wrappers for feature subset selection. *Artificial Intelligence, Special Issue on Relevance*, *97*(1–2), 273–324.

Kohavi, R., & Wolpert, D. (1996). Bias plus variance decomposition for zero-one loss functions. In *Proceedings of the 13th international conference on machine learning* (pp. 275–283).

Koller, D., & Sahami, M. (1996). Toward optimal feature selection. In *Proceedings of the 13th international conference on machine learning* (pp. 284–292).

Kong, E. B., & Dietterich, T. G. (1995). Error-correcting output coding corrects bias and variance. In *Proceedings of the 12th international conference on machine learning* (pp. 313–321).

Korb, K., & Nicholson, A. (2004). *Bayesian artificial intelligence*. London: Chapman & Hall/CRC.

Langley, P., & Sage, S. (1994). Induction of selective Bayesian classifiers. In *Proceedings of the 10th annual conference on uncertainty in artificial intelligence*.

Langley, P., Iba, W., & Thompson, K. (1992). An analysis of Bayesian classifiers. In *Proceedings of the 10th national conference on artificial intelligence* (pp. 223–228).

Lewis, D. D. (1998). Naive Bayes at forty: the independence assumption in information retrieval. In *Proceedings of the 10th European conference on machine learning* (pp. 4–15).

Mitchell, T. M. (1997). *Machine learning*. New York: McGraw–Hill.

Opitz, D. (1995). *An anytime approach to confectionist theory refinement: refining the topologies of knowledge-based neural networks*. Unpublished doctoral dissertation, Department of Computer Sciences, University of Wisconsin-Madison, USA.

Resnick, P., & Varian, H. R. (1997). Recommender systems. *Communications of the ACM*, *40*(3), 56–58.

Sahami, M. (1996). Learning limited dependence Bayesian classifiers. In *Proceedings of the 2nd international conference on knowledge discovery and data mining*.

Schwarz, G. (1978). Estimating the dimension of a model. *Annals of Statistics*, *6*, 461–465.

Suzuki, J. (1996). Learning Bayesian belief networks based on the MDL principle: an efficient algorithm using the branch and bound technique. In *Proceedings of the 13th international conference on machine learning* (pp. 463–470).

Turney, P. (2000). Types of cost in inductive concept learning. In *Workshop on cost-sensitive learning at ICML 2000* (pp. 15–21).

Webb, G. I. (2000). Multiboosting: a technique for combining boosting and wagging. *Machine Learning*, *40*(2), 159–196.

Webb, G. I., Pazzani, M. J., & Billsus, D. (2001). Machine learning for user modeling. *User Modeling and User-Adapted Interaction*, *11*(1–2), 19–29.

Webb, G. I., Boughton, J., & Wang, Z. (2005). Not so naive Bayes: averaged one-dependence estimators. *Machine Learning*, *58*(1), 5–24.

Witten, I. H., & Frank, E. (2005). *Data mining: practical machine learning tools and techniques with Java implementations* (2nd ed.). Los Altos: Kaufmann.

Wu, X., & Urpani, D. (1999). Induction by attribute elimination. *IEEE Transactions on Knowledge and Data Engineering*, *11*(5), 805–812.