Learning for anytime classification

Geoffrey I. Webb and **Janice R. Boughton** and **Ying Yang**
Faculty of Information Technology
Monash University, Vic. 3800, Australia

Abstract

Many on-line applications of machine learning require that the learned classifiers complete classification within strict real-time constraints. In consequence, efficient classifiers such as naive Bayes (NB) are often employed that can complete the required classification tasks even under peak computational loads. While NB provides acceptable accuracy, more computationally intensive approaches can improve thereon. The current paper explores techniques that utilize any additional computational resources available at classification time to improve upon the prediction accuracy of NB. This is achieved by augmenting NB with a sequence of super-parent one-dependence estimators. As many of these are evaluated as possible within the available computational resources and the resulting set of probability estimates aggregated to produce a final prediction. The algorithm is demonstrated to provide consistent improvements in accuracy as computational resources are increased.

Introduction

In many machine learning applications, the computations of the deployed classifier have to be completed within strict elapsed time limits in order to be useful. For example, to be acceptable by users, interactive systems typically need to complete their task within a few seconds. Examples include information retrieval (Baeza-Yates, Baeza-Yates, & Ribeiro-Neto 1999), recommender systems (Resnick & Varian 1997), user modeling (Webb, Pazzani, & Billsus 2001) and online fraud detection (Chan *et al.* 1999).

While the total available resources on acceptable elapsed processing time per job remain constant, typically the number of simultaneous jobs varies from time to time. Hence the resources available for each job will vary. Further, the complexity of individual jobs may vary from job to job, for example due to the amount of available information varying.

The variance in computational resources available to complete an online classification task poses a complex optimization problem for the system designer. From anecdotal evidence, we suggest that the current standard response is as follows. First, an efficient classifier is developed that could provide an acceptable accuracy (or whatever other performance metrics are sought to be optimized). Second, the classifier's resource requirements during peak time are estimated. Third, the classifier is provided necessary resources so as to perform within the required elapsed time constraints during peak time. One real-world example is naive Bayes (NB). NB provides relatively high accuracy with low computational requirements (Lewis 1998). Hence it is very popular with online applications. However one disadvantage in following the above strategy is that the available computational resources will be under-utilized outside peak periods. As a result, a learned classifier can be less accurate than it might be within the available resources if a less efficient but more accurate classifier were employed.

Accordingly this paper studies how to improve learning for classification under varying and uncertain resources and how to optimize accuracy to the extent made possible by the resources available, topics that have received surprisingly little attention to date. A specific goal here is to develop a classification algorithm that utilizes varying classification time. Accomplishing this goal is of clear potential for immediate practical application with substantial benefit.

Problem definition

This paper addresses the problem of classification learning using varying and uncertain computational resources at classification time. We assume k classes $c_1 \dots c_k$; m attributes; and a training set of t objects, $T = \langle\langle y_1, \mathbf{x}_1\rangle \dots \langle y_t, \mathbf{x}_t\rangle\rangle$ where each $y_i \in \{c_1 \dots c_k\}$ and each $\mathbf{x}_i$ is a vector of attribute values $\langle x_1 \dots x_m\rangle$. We use v to denote the average number of values per attribute.

A learner λ is applied to T at *training time* to generate a *model* $\lambda(T)$. A new vector of attribute values $\mathbf{x}$ is presented at *classification time*, together with the model, to a *classifier* ϕ, which produces a classification $\phi(\lambda(T), \mathbf{x}) \in \{c_1 \dots c_k\}$. Unlike most characterizations of the learning task (Mitchell 1997, for example), this characterization allows that a single model might produce different classifications for a single object, when utilized by different classifiers. The standard analysis does not distinguish models and classifiers.

Computation takes place at two different times, training time and classification time. Four distinct computational resources can be distinguished: *training time* (tt), *training space* (ts), *classification time* (ct) and *classification space* (cs). For each of these four resources, we assume a budget that is composed of a *contract* and an *anytime* component (Bernstein *et al.* 2002). The former is specified in advance of computation. The latter is unknown to the learner and classifier until it has been exhausted. It follows that if there is no anytime budget, the system is working within known resource constraints. If there is no contract budget, the system can exhaust its resources at any time.

Our problem is a form of reasoning under resource constraints (Horvitz 1988). Much previous research in this area has tackled deductive reasoning. In contrast, we tackle in-

ductive reasoning. These two types of reasoning might require different optimal strategies to cope with resource constraints. To our knowledge, in the context of classification learning, only a few approaches exist to study how a learning system might adjust to different computational constraints.

Clearly 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 context that we believe 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 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, the contract classification time budget is sufficient to allow standard NB classification to be performed.

A number of researchers have investigated learning under varying training time computational resources (Grefenstette & Ramsey 1992; Opitz 1995; Wah 1992). However, there has been very little work on utilizing varying computational resources at classification time. *Attribute measurement policies* (Arnt & Zilberstein 2004) consider computational resources when gathering information to be used for classification. This differs from the issue tackled in this paper of how to make best use of the available information. *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. The current paper shows how the same principal can be applied to Bayesian classification with the desirable property that the resulting classification are returned as conditional probabilities rather than simple selection of a single class.

Anytime AODE

Our approach is based on the AODE algorithm (Webb, Boughton, & Wang 2005). The reasons for using a variant of AODE for this purpose include that AODE has demonstrated high accuracy with relatively modest computational overheads and supports incremental learning, properties that have already led to considerable interest in its application and refinement (Cerquides & de Mantaras 2005; De Ferrari 2005; Flikka *et al.* 2005; Nikora 2005; Yang *et al.* 2005). It provides conditional probability estimates for each class rather than simply selecting a single class label per instance to be classified. Finally, its use of an ensemble methodology leads to an elegant solution to the anytime classification problem.

AODE

AODE modifies NB to reduce the loss in accuracy that results from inappropriate use of the attribute independence assumption. NB classifies by first estimating $P(y \mid \mathbf{x})$, the probability of each class y given the object to be classified, $\mathbf{x}$. It utilizes Bayes formula

$$P(y \mid \mathbf{x}) = P(y)P(\mathbf{x} \mid y)/P(\mathbf{x}). \quad (1)$$

$P(\mathbf{x} \mid y)$ is estimated using the *attribute independence assumption*, under which,

$$P(\mathbf{x} \mid y) = \prod_{i=1}^{n} P(x_i \mid y) \quad (2)$$

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

AODE is inspired by the notion of *n*-dependence estimators (Sahami 1996). An n-dependence estimator is similar to NB except that each attribute depends upon at most n other attributes in addition to the class. NB is a zero-dependence estimator. The well-known TAN (Friedman, Geiger, & Goldszmidt 1997) is a one-dependence estimator.

Higher-dependence estimators typically have lower bias but higher variance than NB. The more effective ones also typically have very high computational complexity for training time (Webb, Boughton, & Wang 2005). AODE avoids training time computation and reduces variance by averaging over all of a limited class of one-dependence estimators. It is justified by the observation that for any x_i,

$$P(y \mid \mathbf{x}) = P(x_i, y)P(\mathbf{x} \mid x_i, y)/P(\mathbf{x}). \quad (3)$$

It follows that for any $I \subseteq \{1 \ldots m\}$,

$$P(y \mid \mathbf{x}) = \sum_{i \in I} \frac{P(x_i, y)P(\mathbf{x} \mid x_i, y)}{P(\mathbf{x})} / |I| \quad (4)$$

where $|I|$ denotes the number of elements in I.

By assuming all attributes are independent of each other given x_i and y we can derive

$$P(\mathbf{x} \mid x_i, y) = \prod_{j=1}^{m} P(x_j \mid x_i, y). \quad (5)$$

By using (5) within (3) we obtain a super-parent one-dependence estimator (SPODE), a one-dependence estimator for which there is a single attribute such that all other attributes depend upon only it and the class. AODE uses (5) within (4) resulting in a new one-dependence estimator that averages over some set of SPODEs.

Probability estimates are likely to be inaccurate if there are too few relevant examples in the training data. In consequence, when classifying object $\mathbf{x}$, I is restricted to the set of attributes for which the value x_i occurs at least $minfreq$ times in the training data. In the current research we use $minfreq = 30$ as 30 is widely accepted as a minimum sample size from which to expect acceptable estimates.

AODE has low training time complexity as all it does is collect a table of joint frequencies from which the requisite probabilities are estimated at classification time. Table 1

Table 1: Computational Complexity

	Training		Classification	
	Time	**Space**	**Time**	**Space**
NB	$O(tm)$	$O(kmv)$	$O(km)$	$O(kmv)$
AODE	$O(tm^2)$	$O(k(mv)^2)$	$O(km^2)$	$O(k(mv)^2)$

shows the comparative computational complexity of NB and AODE for each of the four computational resources required at training and classification time. In each case the time complexity of AODE is m times that of NB. For data that contain large numbers of attributes this can represent a substantial difference in the amount of computation required. As discussed above, we believe that there are many applications for which the contract tt and ts budgets will be large and the model component will not count toward the cs budget. AODE has been shown consistently to produce more accurate classifiers than NB (Webb, Boughton, & Wang 2005), but it is likely frequently to require more compute time than the combined contract and anytime ct budgets allow. With Anytime AODE we seek to develop an algorithm with a base ct complexity equivalent to NB and which can perform additional computation to refine its initial class probability estimates up to a computational ct budget equivalent to AODE.

The Anytime AODE algorithm

Our initial algorithm modified AODE to compute the probability estimates for each SPODE in sequence, terminating and returning the average probabilities for all SPODEs completed when the ct budget was exhausted. Investigation of this strategy revealed a weakness, however. As mentioned above, SPODEs typically have higher variance than NB. 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. In consequence, we modified our approach to compute the probability estimates for NB before those of the SPODEs and to average over all of the NB and SPODE estimates. This is justified by the following equality that is true of any $I \subseteq \{1 \dots m\}$ and that follows directly from (1) and (4).

$$P(y \mid \mathbf{x}) = \frac{P(y)P(\mathbf{x} \mid y) + \sum_{i \in I} P(x_i, y)P(\mathbf{x} \mid x_i, y)}{P(\mathbf{x})(|I| + 1)}$$

Table 2 presents the resulting AAODE algorithm. 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 7. Lines 11 to 16 compute the probability estimates TP for a single SPODE. Lines 18 to 21 update the average conditional 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 inconsistent state. In consequence, if the time budget expires during this process there will be a slight delay before computation terminates.

Line 9 starts a loop over the set of qualified attributes I that selects the super-parent for each SPODE to be computed. The algorithm does not specify how to select the set of qualified attributes, and as discussed above, this is a promising direction for further research. For the current research $I = \{i \mid C(x_i) \geq 30\}$, where $C(x_i)$ is the count of

Table 2: The Anytime AODE Algorithm

Algorithm: AAODE

Inputs:

- $\mathbf{x}$: the object to be classified.
- I: the set of attributes that are to be the parents for the SPODEs over which AAODE will average.
- $PP[c_1 \dots c_k]$: 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 estimate for each class and attribute-value pair.
- $CP[c_1 \dots c_k, attvals]$: conditional probability estimate for each attribute value given each class.
- $CP[c_1 \dots c_k, attvals, attvals]$: conditional probability estimate for each attribute-value (the final index) given each class and another attribute-value.

Outputs:

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

```
1: for y := c_1 ... c_k do
2:   P[y] := PP[y]
3:   for i := 1 ... m do
4:     P[y] := P[y] × CP[y, x_i]
5:   end for
6: end for
7: on interrupt goto 23
8: count := 1
9: for all p ∈ I do
10:   count := count + 1
11:   for y := c_1 ... c_k do
12:     TP[y] := PP[y, x_p]
13:     for i := 1 ... m do
14:       TP[y] := TP[y] × CP[y, x_p, x_i]
15:     end for
16:   end for
17:   suspend interrupts
18:   for y := c_1 ... c_k do
19:     P[y] := P[y] + (TP[y] − P[y])/count
20:   end for
21:   restore interrupts
22: end for
23: normalize(TP)
24: return P
```

the number of times x_i occurs in T. Nor does the algorithm specify the order in which the elements of I should be processed. It is clearly desirable to process first the SPODES that have the highest expected accuracy. Establishing such an order is another promising avenue for investigation. In the current work we use an order that can be established with minimal computation—descending order on $C(x_i)$. Note that this is a lazy technique, the order of the attributes is determined at classification time on the basis of the specific values of the attributes for the instance to be classified.

Evaluation

The prediction performance of both NB and AODE in comparison to other algorithms has already been studied extensively for a wide variety of applications (Webb, Boughton, & Wang 2005). With minimal time, AAODE will deliver

identical predictions to NB. If computation is not terminated early, we expect AAODE to deliver similar predictions to AODE, as the calculations performed will be very similar. With these considerations, we seek to assess the following aspects of AAODE's performance. 1) How does the accuracy of AAODE compare to that of AODE when AAODE is allowed to run uninterrupted? 2) How does the accuracy of AAODE change as classification time increases?

We do not seek to assess the compute time taken by the algorithm for the following reasons. First, the compute time will vary greatly depending upon the hardware architecture on which it is executed, being affected greatly by such issues as whether the entire table of frequency data can be retained in the processor's cache. Hence, our results are not likely to generalize well. Second, the computational characteristics of NB are well understood. Each SPODE over which AAODE averages has similar computational requirements to NB. Hence, the computational requirements will be approximately $n + 1$ times those of NB, where n is the number of SPODEs that are completed in addition to NB. Further, while for ease of exposition we present our analysis in terms of sequential computation, the compute-time budget might be allocated in numbers of processors. AAODE can be readily parallelized, with one sub-model processed on each available processor. In such a configuration, classification would respond to available computational resources by adjusting the number of sub-models to be processed in parallel. Finally, for the sake of developing code that can be readily shared with other researchers, we have implemented a simulation of AAODE in the Weka (Witten & Frank 2005) machine learning environment. However, this environment does not readily lend itself to implementing time-based interrupts. Hence, rather than a true implementation of AAODE, we have developed a simulation for which the amount of computation to be performed is specified in terms of the number of sub-models to be evaluated.

We conducted experiments using all 37 datasets used in previous research into AODE. These datasets are described in Table 3, which shows the number of instances, attributes, and classes for each. For each dataset we performed ten-fold cross validation. This process divides the data set into ten equal-size subsets. Each subset is used in turn as a test set with the remaining nine datasets use for training, creating ten training-set – test-set pairs. For each of these we ran AAODE repeatedly, once for each number of sub-models (NB plus up to m SPODEs) as well as AODE. In keeping with Weka's NB and AODE, we estimated the base probabilities $P(y)$, $P(y, x_i)$ and $P(y, x_i, x_j)$ using the Laplace estimate as detailed in Webb, Boughton, & Wang 2005. As our techniques require discrete-valued data, each training set was discretized using the Weka MDL discretizer.

Table 3: Datasets

Dataset	Instances	Attributes	Classes
lung-cancer	32	56	3
labor-neg	57	16	2
post-operative	90	8	3
promoters	106	57	2
echocardiogram	131	6	2
iris	150	4	3
hepatitis	155	19	2
wine	178	13	3
sonar	208	60	2
glass	214	9	3
new-thyroid	215	5	3
heart	270	13	2
hungarian	294	13	2
cleveland	303	13	2
primary tumor	339	17	22
bupa	345	6	2
ionosphere	351	34	2
horse-colic	368	21	2
house-votes-84	435	16	2
chess	551	39	2
syncon	600	60	6
balance-scale	625	4	3
crx	690	15	2
bc wisconsin	699	9	2
vehicle	846	18	4
anneal	898	38	6
tic-tac-toe	958	9	2
german	1000	20	2
led	1000	7	10
mfeat-mor	2000	6	10
segment	2310	19	7
hypothyroid	3163	25	2
satellite	6435	36	6
pendigits	10992	16	10
sign	12546	8	3
letter-recognition	20000	16	26
adult	48842	14	2

Results

The first question we address is how varying the amount of computation affects classification accuracy. As might be expected, this varies considerably from dataset to dataset. At one extreme are datasets like pendigits, where increasing the number of sub-models progressively decreases error, starting from the error of NB with a single sub-model, and ending with error slightly below that of AODE. Figure 1 plots this result. In this and the next error plot, the error of AODE is shown by a dotted line extending across the graph.

At the other extreme lies a dataset like heart, presented in Figure 2, for which error initially drops, but for which increased numbers of sub-models at one stage leads to worse error than either NB or AODE. Such results are inevitable if sufficiently broad evaluation is performed, as the no-free-lunch theorem (Wolpert 1992) shows that for any pair of algorithms there must exist learning tasks for which each dominates the other.

Fortunately, well-behaved performance dominates ill-behaved performance. This is demonstrated by the win-draw-loss results presented in Table 4. They show the relative numbers of datasets for which the error of one comparator dominates another, where the comparators are NB (AAODE with one-submodel), MID (AAODE with half the maximum possible number of sub-models), MAX (AAODE with all possible sub-models) and AODE. Each entry contains three numbers. The first is the number of datasets for which the algorithm with which the row is labelled obtains lower average error than the algorithm with which the col-

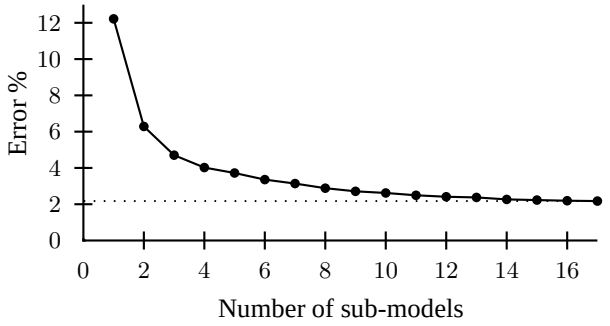

Figure 1: Learning curve for pendigits

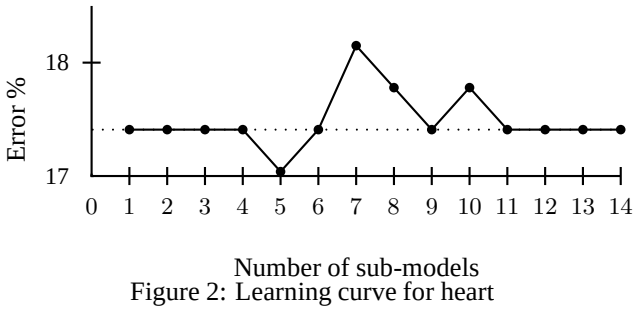

Figure 2: Learning curve for heart

umn is labelled. The second value is the number of datasets for which both algorithms obtain identical average error (computed to four decimal places). The third value is the number of datasets for which the algorithm with which the column is labelled obtains lower average error than that with which the row is labelled. Entries set in bold face represent results where a binomial test assesses as less than 0.05 the probability of obtaining the observed record of wins relative to losses by chance if each were equally likely. One-tailed tests were applied, using in every case the expectation that NB would obtain the highest error followed in order of decreasing error by MID, MAX then AODE. As can be seen, the win/draw/loss records in all cases result in the most wins for the algorithm for which we predict the lowest error, although the 17 wins of MID vs. 9 for NB are not significant at the 0.05 level ($p = 0.084$), nor are the 12 for AODE vs. 5 for MAX ($p = 0.072$).

Table 4: Win/Draw/Loss Table

	MID	**MAX**	**AODE**
NB	9/11/17	**2/11/24**	**3/10/24**
MID		**4/10/23**	**4/7/26**
MAX			5/20/12

To provide an overview of the step-by-step performance across all 37 datasets we standardized the mean classification error for each experiment by dividing it by the mean classification error for NB for the relevant dataset. Figure 3 presents the average across all datasets of these standardized errors. The x-axis presents the number of sub-models and the y-axis presents the average standardized error for that number of sub-models. Note that the number of datasets over which the result is averaged decreases gradually as the number of sub-models increases, because the number of sub-models for a dataset cannot exceed $m + 1$. As can be seen, this curve shows a clear trend for error relative to NB to decrease as the number of sub-models increases.

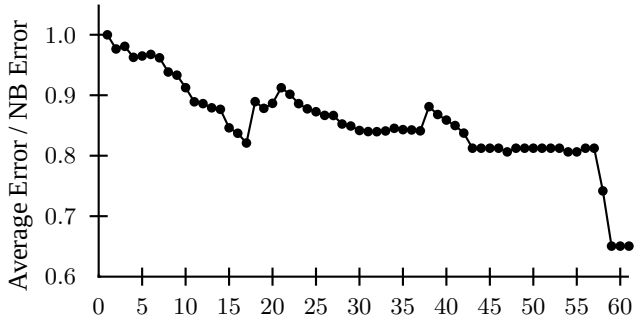

Figure 3: Learning curve for AAODE

Discussion and further research

In principle, one could turn any ensemble classifier into an anytime classifier by including as many component classifiers as classification time allows. However, AAODE has a number of very desirable features that would not be shared by most such anytime classification systems. First, unlike most ensemble classifiers, one does not need to know before classification time which attribute will be the class variable[1]. Second, when new training data are available, AAODE's component classifiers (SPODEs) can be easily updated, simply by incrementing the relevant entries in the table of joint frequencies. With most ensemble learners the only way to utilize new data is to relearn from scratch. Third, the classifiers return the estimated probability for each class label, offering greater information than simple classifications. Fourth, the approach does not have the high learning time complexity of most ensemble learners. At learning time all that is required is the collection of the joint frequencies, an operation of time complexity $O(tm^2)$ and space complexity $O(k(mv)^2)$. In contrast, most ensemble learners must repeat the entire learning process of the base learning algorithm once for each ensemble classifier to be learned.

Of the multifaceted problem that is defined in Section , the current work has investigated only one dimension: varying classification time budgets. Further research could extend the AAODE framework to also address other types of varying resource budgets.

One possibility is to enable AAODE to cope with training time variance, using extra training time to further improve classification accuracy. For example, one may select an order for processing the SPODES at line 10 of AAODE. Preliminary investigations have shown clear potential for ordering the SPODES by expected accuracy. However, this implies that the final attributes processed will have low expected accuracy, and this frequently results in learning curves for which error increases for the final attributes processed. It appears that any such ordering strategy will need to be coupled with an appropriate attribute-selection strategy (Yang *et al.* 2005).

[1]However, if the class variable is known beforehand, more compact frequency tables can be formed.

Another dimension is to extend AAODE to accommodate varying space resources. For example, one may alter the level of dependence of the classifiers (one-dependence estimators, two-dependence estimators, etc.) according to the space available to store the conditional frequency tables.

Further research could also be explored to optimize performance metrics other than classification accuracy that this paper focuses on. Those metrics may include accuracy of the probability estimates, true positive rate, true negative rate, sensitivity, specificity, precision or recall since they can often be primary objectives in practice. Techniques that manipulate computational resources to optimize each of those objectives are useful as well as desirable.

Last but not least, learning under resource constraints could also be studied for learning contexts other than the Bayesian probabilistic learning in which AAODE is engaged, such as SVMs and decision trees.

Conclusion

Inductive learning often has to abide by strict resource constraints in real-world applications. In this paper we highlight four of those computational resources, training space, training time, classification space and classification time.

In particular, we have focused on applications where naive-Bayes (NB) is currently employed. We have argued that periods may well exist during which classification time is available beyond what NB needs. Accordingly we have proposed a new anytime learning strategy, AAODE. AAODE first estimates class probabilities by NB and then takes advantage of idle classification-time resources to refine its estimates by superparent one-dependence estimators (SPODEs). The order in which SPODEs are processed is determined at classification time. Techniques for ordering SPODEs remain a promising direction for further research.

AAODE has several desirable properties. It lends itself to parallel computing by utilizing a variable number of processors, one for each SPODE. It is flexible with regard to which attribute is used as the class variable. It has low training time complexity. It supports incremental learning. It provides class probability estimates rather than simple classifications. Finally, our empirical evaluation suggests that AAODE is able to improve classification accuracy by effectively making use of increasing classification time.

References

Arnt, A., and Zilberstein, S. 2004. Attribute measurement policies for time and cost sensitive classification. In *4th ICDM*, 323–326.

Baeza-Yates, R. A.; Baeza-Yates, R.; and Ribeiro-Neto, B. 1999. *Modern Information Retrieval*. Addison-Wesley Longman.

Bernstein, D. S.; Perkins, T. J.; Zilberstein, S.; and Finkelstein, L. 2002. Scheduling contract algorithms on multiple processors. In *18th AAAI/IAAI*, 702–706.

Cerquides, J., and Mantaras, R. L. 2005. Robust bayesian linear classifier ensembles. In *16th ECML*, 72–83.

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

De Ferrari, L. 2005. Mining housekeeping genes with a naive Bayes classifier. MSc Thesis, University of Edinburgh, School of Informatics.

DeCoste, D. 2002. Anytime interval-valued outputs for kernel machines: Fast support vector machine classification via distance geometry. In *19th ICML*, 99–106.

Flikka, K.; Martens, L.; Vandekerckhove, J.; Gevaert, K.; and Eidhammeri, I. 2005. Improving throughput and reliability of peptide identifications through spectrum quality evaluation. In *9th Annual Int. Conf. Research in Computational Molecular Biology*.

Friedman, N.; Geiger, D.; and Goldszmidt, M. 1997. Bayesian network classifiers. *Machine Learning* 29(2):131–163.

Grefenstette, J., and Ramsey, C. 1992. An approach to anytime learning. In *9th Int. Machine Learning Workshop*, 189–195. Morgan Kaufmann.

Horvitz, E. J. 1988. Reasoning under varying and uncertain resource constraints. In *7th AAAI*, 111–116.

Lewis, D. D. 1998. Naive Bayes at forty: The independence assumption in information retrieval. In *10th ECML*, 4–15.

Mitchell, T. 1997. *Machine Learning*. McGraw Hill.

Nikora, A. P. 2005. Classifying requirements: Towards a more rigorous analysis of natural-language specifications. In *16th IEEE Int. Symposium on Software Reliability Engineering*, 291–300. IEEE Press.

Opitz, D. 1995. *An Anytime Approach to Connectionist Theory Refinement: Refining the Topologies of Knowledge-Based Neural Networks*. Ph.D. Dissertation, Dept. of Computer Sciences, Univ. of Wisconsin-Madison.

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

Sahami, M. 1996. Learning limited dependence Bayesian classifiers. In *8th SIGKDD*, 334–338.

Wah, B. W. 1992. Population-based learning: A method for learning from examples under resource constraints. *IEEE TKDE* 4(5):454–474.

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

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

Witten, I. H., and Frank, E. 2005. *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations, 2nd Edition*. Morgan Kaufmann.

Wolpert, D. H. 1992. On the connection between in-sample testing and generalization error. *Complex Systems* 6:47–94.

Yang, Y.; Korb, K.; Ting, K. M.; and Webb, G. I. 2005. Ensemble selection for superparent-one-dependence estimators. In *18th Australian AI*, 102–111.