Model Trees for Classification of Hybrid Data Types*

Hsing-Kuo Pao, Shou-Chih Chang, and Yuh-Jye Lee

Dept. of Computer Science & Information Engineering,
National Taiwan University of Science & Technology, Taipei, Taiwan
{pao, M9115009, yuh-jye}@mail.ntust.edu.tw

Abstract. In the task of classification, most learning methods are suitable only for certain data types. For the hybrid dataset consists of nominal and numeric attributes, to apply the learning algorithms, some attributes must be transformed into the appropriate types. This procedure could damage the nature of dataset. We propose a model tree approach to integrate several characteristically different learning methods to solve the classification problem. We employ the decision tree as the classification framework and incorporate support vector machines into the tree construction process. This design removes the discretization procedure usually necessary for tree construction while decision tree induction itself can deal with nominal attributes which may not be handled well by e.g., SVM methods. Experiments show that our purposed method has better performance than that of other competing learning methods.

1 Introduction

In the real world, the datasets usually include both of the (unordered) *nominal* (or discrete) attributes and the *numeric* (or continuous) attributes. We name this kind of datasets as *hybrid* datasets. Most learning algorithms for classification are only suitable for certain specified data types. When the undesired data types are encountered in the dataset, conventionally we transform them into appropriate types so that the learning algorithm can be proceeded [1–4]. E.g., numeric data need a discretization process before the typical decision tree induction can be applied and SVM works on the space of numeric data only. Sometimes the type transformation is artificial and results in changing of the dataset nature.

To overcome this problem we employ a novel model tree approach which a decision tree (DT) framework, combining with SVMs [5, 6] will be used for the classification of hybrid sets. During the tree construction, the SVMs play a role of replacing the discretization procedure and providing a possible way of extending a univariate decision to a multivariate decision. In an internal node, before the tree splitting, SVM will help to generate a synthetic Boolean attribute based on the numeric attributes of current training examples in this node, rather

* Research partially supported by Taiwan National Science Council Grant # 93-2213-E-011-036.

than discretize the numeric attributes regardless of their interdependencies [1, 3]. When we choose the "best" splitting attribute we consider the original nominal attributes as well as the synthesized Boolean attribute. If the synthetic Boolean attribute is chosen as the splitting attribute, it means that the decision node has a multivariate decision implicitly. Therefore, this strategy extends the ability of DTs to include multivariate decisions. On the other hand, SVM itself can not naturally deal with nominal data without creating any artificial encodings. Thus, our proposed model tree, combining the power of DT and SVM, will be suitable to solve the classification problem for hybrid datasets.

2 Decision Tree Induction with Hybrid Data Types

DT methods [7, 8] are used widely in the fields of machine learning and data mining. A DT consists of internal and external nodes where an internal node with several branches represents alternative choices to make based on the (discrete) values of selected attribute and an external node (a leaf) is usually associated with a single class label. A prediction is done following the path from the tree root to a leaf, by several branch choices according to given attribute values. The typical DT construction adopts the top-down, divide-and-conquer strategy to recursively build the classification tree [8]. DTs have some advantages such as easy to interpret, efficient to generate and capable of coping with noisy data [9, 10]. However, DTs are notorious to be unstable (i.e., high variance). Often a small change in the training set results in different trees and produces inconsistent classification results for the same test set. The instability is inherent because the effect of an error on a top split will be propagated down to all of the splits below [11]. Some approaches have been proposed by combining multiple models to improve the accuracy and stability of DT prediction, such as *bagging* or *boosting* [12–14]. Some examples of DT induction are ID3, C4.5[1], C5.0[2] [8, 12] and CART [7]. We shall discuss two issues related to DT induction.

2.1 Incorporating Continuous-Valued Attributes

Many real world classification tasks involve nominal and numeric attributes. For numeric attributes, DT can not be adopted directly unless they are discretized in advance, i.e., partitioning each of the continuous attributes into disjoint intervals [1]. E.g., an attribute X can be separated as $X \leq c$ and $X > c$ for the binary DT. The strategies of discretization are usually categorized by (1) being supervised or unsupervised, (2) being global or local, and (3) being static or dynamic, three options [1, 2, 4, 8, 3]. Most choices are heuristically or empirically decided. Also, for many of the discretization approaches, the number of intervals is decided arbitrarily. These can lead to low prediction accuracies or inefficient tree structures, for datasets with hybrid data types or only the numeric data

[1] Some MDL-based discretization for continuous attributes is adopted in certain versions.

[2] A variant of AdaBoost is implemented.

type [15, 1, 4, 8]. While many DT inductions are more satisfied with discrete attributes than continuous ones, we adopt SVM for classification in the subspace spanned by those continuous attributes. In Sec. 4, a combined classifier from DT and SVM will be introduced to deal with datasets with hybrid types.

2.2 Univariate and Multivariate Decision Trees

The classical approach for building a DT, such as C4.5, uses an orthogonal (or axis-parallel) partition at each decision node, so called *univariate* method [8]. Opposite to that, CART [7] allows for the option of *multivariate* decisions. For instance, one check simultaneously involving two attributes X_1 and X_2, such as $X_1+X_2 \leq 6.5$, may be operated in a decision node. Clearly, there are cases where multivariate approach can work efficiently (producing trees with few nodes), but not for the univariate approach[3] [17, 7, 9, 18–21, 10]. We introduce SVM for being capable of multivariate consideration at a node. Other than using a SVM in each decision node in [17], we adopt the machine *only* for continuous attributes. For discrete attributes, the regular ID3 algorithm is applied. By that, we take advantage of powerful SVM for classification, while not losing the readability of DT induction. Further discussion is in Sec. 4.

3 Support Vector Machines

We are given a training dataset $S = \{(\mathbf{x}^1, y_1), \ldots, (\mathbf{x}^m, y_m)\} \subseteq R^n \times R$, where $\mathbf{x}^i \in R^n$ is the input data and $y_i \in \{-1, 1\}$ is the corresponding class label. The aim of SVM is to find the optimal separating hyperplane with the largest margin from the training data. Here, "optimal" is used in the sense that the separating hyperplane has the best generalization for the unseen data based on statistical learning theory [6]. This can be achieved by solving a convex optimization problem given as follows:

$$\begin{aligned} \min_{(w,b,\xi)\in R^{n+1+m}} \; & C\sum_{i=1}^{m}\xi_i + \tfrac{1}{2}\|w\|_2^2 \\ \text{s.t. } & y_i(w'x^i + b) + \xi_i \geq 1 \\ & \xi_i \geq 0, \quad for\ i = 1, 2, \ldots, m, \end{aligned} \tag{1}$$

where C is a positive control parameter and weights the tradeoff between the training error and the part of maximizing the margin. We have to point out here, due to the nature of SVM it is more suitable for numeric data type.

In smooth support vector machine (SSVM) [22], the SVM model (1) is changed slightly and converted into a unconstrained minimization problem by utilizing the optimality conditions. These give the SVM reformulation defined as follows:

$$\min_{(w,b)\in R^{n+1}} \frac{C}{2}\sum_{i=1}^{m}(1 - y_i(w'x^i + b))_+^2 + \frac{1}{2}(\|w\|_2^2 + b^2), \tag{2}$$

[3] For the multivariate case, the separating hyperplane do not need to be linear [16].

where the *plus* function x_+ is defined as $x_+ = \max\{0, x\}$. In SSVM, the plus function x_+ is approximated by a smooth p-function, $p(x, \alpha) = x + \frac{1}{\alpha}\log(1 + e^{-\alpha x}), \alpha > 0$. By replacing the plus function with a very accurate smooth approximation p-function gives the smooth support vector machine formulation:

$$\min_{(w,b)\in R^{n+1}} \frac{C}{2}\sum_{i=1}^{m}(p(1 - y_i(w'x^i + b), \alpha))^2 + \frac{1}{2}(\|w\|_2^2 + b^2), \tag{3}$$

where $\alpha > 0$ is the smooth parameter. The objective function in problem (3) is strongly convex and infinitely differentiable. Hence, it has a unique solution and can be solved by using a fast Newton-Armijo algorithm [22]. This formulation can be extended to the nonlinear SVM by using the kernel trick. We will not use the nonlinear SSVM in our proposed method because the nonlinear SSVM tends to overfit the small portion of training dataset in the training process.

In next section, we employ the linear SSVM to deal with the numeric attributes and to generate the corresponding synthetic Boolean attribute for the training examples at each node.

4 Model Trees

With the description in the previous sections, we know DTs and SVMs have their own characteristics to deal with different classification problems:

1 Most DTs require a discrete feature space.When a DT encounters numeric attributes, a discretization procedure is applied beforehand to divide each single numeric attribute into many distinct intervals.
2 On the other hand, SVMs are suitable for the classification of numeric data. If datasets contain the nominal attributes, some strategies such as encoding (usually artificial) are applied to transform the nominal attributes into a series of binary attributes and SVMs treat the values of binary attributes as the integers, 0 and 1.

To flexibly choose the most appropriate method for different types of attributes and to overcome the limitation of univariate decision for numeric attributes in DT induction, we propose a new approach which adopts SVM training in the process of DT construction. At each node, we use a SVM classification in the subspace spanned by the (whole) numeric attributes to replace the used-to-be-necessary discretization procedure. Simultaneously, the SVM represents the possible multivariate decision to improve the efficiency of univariate method. After the SVM is built, this "multivariate" decision can be considered and competed with the other nominal attributes, based on information gain, gain ratio or other goodness criteria. Below, we give the modeling process in detail.

4.1 Building Model Trees

Suppose an example in the hybrid dataset is expressed as the form ($\mathbf{x}_{NOM}$, $\mathbf{x}_{NUM}$, y), where $\mathbf{x}_{NOM}$, $\mathbf{x}_{NUM}$ and y represent all of the nominal attributes, all

of the numeric attributes and the associated class label, respectively. Moreover, we use the notation, x_{SVMB}, to represent the synthesized Boolean attribute whose value is assigned at each node by the SSVM classifier, built from the part of numeric attributes and training labels. Afterwards, the gain ratio criterion is employed to decide the best attribute among all of the nominal ones and the synthesized SVM attribute. That is, *in each node*, we do the following steps:

Step 1 Using $(\mathbf{x}_{NUM}, y)$ to build x_{SVMB}. The process consists of three parts. The first work is to search the appropriate weight parameter for the linear SSVM classifier. That is to say, we split $\mathbf{x}_{NUM}$ of training examples into training set and validation set following the stratification and then decide the appropriate weight parameter by them. The second work is to retrain the SSVM classifier by means of the chosen parameter and $(\mathbf{x}_{NUM}, y)$ of training examples. Finally, we use the retrained SSVM classifier, denoted by $f(\mathbf{x}_{NUM})$, to generate the corresponding x_{SVMB} according to $\mathbf{x}_{NUM}$ of each training example. If $f(\mathbf{x}_{NUM}) > 0$, the value of x_{SVMB} is True; otherwise is False. After the process is finished, training examples are transformed to the new form, $(\mathbf{x}_{NOM}, x_{SVMB}, y)$.

Step 2 Using the gain ratio to select the most appropriate splitting attribute from $\mathbf{x}_{NOM}$ or x_{SVMB}. The split with the highest value of gain ratio will be selected as the attribute. After the splitting attribute is decided, the dataset is partitioned into two or more subsets accordingly. Note that in order to avoid the case that our method always chooses the synthetic Boolean attribute generated via the SSVM, we confine ourselves in the *linear* SSVM. Besides, the weight parameter used in SSVM is determined by a tuning procedure to avoid the overfitting risk.

If one attribute of $\mathbf{x}_{NOM}$ is selected, it means that not only the nominal attribute is more distinguishing than x_{SVMB} but also the decision is univariate. Oppositely, if x_{SVMB} is selected, it shows that the linear combination of all numeric attributes has better chance to separate the examples and the decision node is multivariate implicitly. The process is repeated recursively until any stopping criterion is met.

5 Experiments

In this section, we test our method on three benchmark datasets from the UCI repository[4] to evaluate its performance. In order to get a fair result, we repeat four rounds tenfold cross-validation procedure for each experiment. Furthermore, two popular classification methods, Naive Bayes (NB) and k-nearest-neighbor (k-NN), are employed to provide the baseline accuracies. Three series of experiments are performed. First, the classification error rates from different *views* (different parts of attributes) are presented. Then we present the final comparison results from NB, k-NN, C4.5, SSVM and our model tree method. In our experiments, we

[4] http://www.ics.uci.edu/∼mlearn/MLRepository.html

choose three hybrid datasets, Cleveland heart disease, Australian and German that include both of the nominal and numeric attributes from the UCI repository. They are summarized in Table 1.

Dataset	Instances	# of nominal attr.	# of numeric attr.	Majority error
Heart	270	7	6	44.44%
Australian	690	8	6	44.49%
German	1000	13	7	30%

Table 1. Summary of Datasets

In NB approach, for nominal attributes, NB counts the frequencies as the probabilities $P(y)$ and $P(x_i|y)$, for attribute value x_i and class label y; and for numeric attributes, it assumes that the data follows a Gaussian distribution, hence; the probability of the attribute value can be estimated by the probability density function. Finally, the class of the test example is assigned by the posterior probability. In k-NN, for nominal attributes, the distance is zero if the attribute value is identical, otherwise the distance is one; for numeric attributes, the Euclidean distance is applied directly. We discuss three series of experiments.

Different views: nominal attributes In the first series, only nominal attributes are extracted from the dataset. Three learning methods, NB, k-NN and C4.5 are performed. Appropriate parameter tuning is done for each learning algorithm if there is a need. In this series, k-NN is the most questionable method. Because it can not reflect the actual distance among different nominal values. The result is shown in Table 2(a).

Different views: numeric attributes In the second series, only numeric attributes are extracted. There are five learning methods, NB, k-NN, C4.5, linear SSVM and Nonlinear SSVM performed. Appropriate parameter tuning is done if there is a need. In C4.5, the values of the numeric attributes are divided into two intervals in the local discretization procedure. The result is shown in Table 2(b). From the first two series, we discover that the results of nominal attributes are significantly better than the numeric counterparts. Also, it shows that the linear SSVM performs better than all other methods.

Different methods: all attributes In the third experiment, we compare the error rates of different methods for hybrid datasets. Because SSVM can only deal with the numeric attributes, we encode the nominal attributes into a series of Boolean attributes for SSVM. For example, if the nominal attribute has three possible values, we encode them as 001, 010 and 100. In model trees, we use the minimum instances as the early stopping criterion. The number of minimum instances is determined by a tuning procedure. The final results are shown in Table 3. The model tree and linear SSVM have the similar accuracies. Moreover, comparing model trees with C4.5, we find that model trees outperform C4.5 in the Heart and German, and have the similar accuracy in the Australian.

Dataset	Classification Method		
	Naive Bayes	k-NN	C4.5
Heart	21.02	19.81	24.54
Australian	13.73	13.33	14.42
German	25.68	28.20	27.25

(a) only nominal attributes

Dataset	Classification Method				
	Naive Bayes	k-NN	C4.5	Linear SSVM	Nonlinear SSVM
Heart	23.33	22.87	25.83	21.76	29.63
Australian	28.55	25.83	23.80	23.04	24.35
German	29.08	33.70	30.13	28.77	28.90

(b) only numeric attributes

Table 2. Classification based *only* on nominal or numeric attributes (error rates %)

Dataset	Classification Method					
	Naive Bayes	k-NN	C4.5	Linear SSVM	Nonlinear SSVM	Model trees
Heart	16.02	17.78	21.67	13.98	29.81	*15.65*
Australian	22.90	13.33	13.26	13.38	23.88	*12.61*
German	25.25	25.95	26.55	24.38	28.98	*24.67*

Table 3. Classification for hybrid datasets (error rates %)

6 Conclusion

We employed DT as the classification framework and incorporated the SVM into the construction process of DT to replace the discretization procedure and to provide the multivariate decision. The main idea of our proposed method was to generate a synthetic Boolean attribute according to the original numeric attributes and the synthetic Boolean attribute represented the discriminability of the numeric attributes. Hence, the multivariate decision could be taken into account during the selection of next splitting attribute. Finally, the experiment results showed that model tree has better accuracy than the conventional DT C4.5. We noted that our method can not avoid the inherent instability of DTs.

Our model tree was not just designed for the SVM method only. Any learning methods appropriate to apply to numeric attributes such as Fisher's linear discriminant function or neural networks could be adopted to form a synthetic Boolean attribute and the rest induction procedure is the same. We could also accept more than one such synthesized attribute; thus, more than one learning algorithm at a time under the framework of DTs. We have to point out that designing a good tuning process to avoid the overfitting risk is extremely important. Otherwise, DTs tend to choose the synthetic Boolean attribute induced by the learning algorithm which has the overfitting drawback as the splitting attribute. One design is to apply MDL principle to balance between nominal attributes and the synthetic attribute(s), or between the synthetic attributes generated from different learning methods. With the help from characteristically different learning methods, we could build a classifier which can deal with data of hybrid types successfully.

References

1. Dougherty, J., Kohavi, R., Sahami, M.: Supervised and unsupervised discretizations of continuous features. In: Proceedings of the 12th International Conference on Machine Learning, New York, Morgan Kaufmann (1995) 194–202

2. Fayyad, U.M., Irani, K.B.: Multi-interval discretization of continuous valued attributes for classification learning. In: Proceedings of the 13th International Joint Conference on Artificial Intelligence. (1993) 1022–1029
3. Gama, J., Torgo, L., Soares, C.: Dynamic discretization of continuous attributes. In: Proceedings of the Iberoamericam Conference on AI (IBERAMIA-98), Springer-Verlag (1998) 160–169
4. Kohavi, R., Sahami, M.: Error-based and entropy-based discretization of continuous features. In: Proceedings of the Second International Conference on Knowledge Discovery and Data Mining (KDD-96), AAAI Press (1996) 114–119
5. Burges, C.J.C.: A tutorial on support vector machines for pattern recognition. Data Mining and Knowledge Discovery **2** (1998) 121–167
6. Vapnik, V.N.: The Nature of Statistical Learning Theory. Springer-Verlag, New York (1995)
7. Breiman, L., Friedman, J.H., Olshen, R.A., Stone, C.J.: Classification and Regression Trees. Wadsworth, Belmont, CA (1984)
8. Quinlan, J.R.: C4.5 Programs for Machine Learning. Morgan Kaufmann (1993)
9. Brodley, C.E., Utgoff, P.E.: Multivariate decision trees. Machine Learning **19** (1995) 45–77
10. X.-B. Li, Sweigart, J.R., Teng, J.T.C., Donohue, J.M., Thombs, L.A., Wang, S.M.: Multivariate decision trees using linear discriminants and tabu search. Systems, Man and Cybernetics, Part A, IEEE Transactions on **33** (2003) 194–205
11. Hastie, T., Tibshirani, R., Friedman, J.: The Elements of Statistical Learning. Springer-Verlag, New York (2001)
12. Freund, Y., Schapire, R.E.: A decision-theoretic generalization of online learning and an application to boosting. J. of Comp. and Sys. Sciences **55** (1997) 119–139
13. Quinlan, J.R.: Bagging, boosting, and c4.5. In: Proceedings of the Thirteenth National Conference on Artificial Intelligence and Eighth Innovative Applications of Artificial Intelligence Conference, AAAI 96, AAAI Press (1996) 725–730
14. Breiman, L.: Bagging predictors. Machine Learning **24** (1996) 123–140
15. Quinlan, J.R.: Improved use of continuous attributes in C4.5. Journal of Artificial Intelligence Research **4** (1996) 77–90
16. Ittner, A., Schlosser, M.: Non-linear decision trees - NDT. In: Machine Learning, Proc. of the 13th Inter. Conf. (ICML '96), Morgan Kaufmann (1996) 252–257
17. Bennett, K., Blue, J.: A support vector machine approach to decision trees (1997)
18. Heath, D., Kasif, S., Salzberg, S.: Induction of oblique decision trees. In: Proceedings of the 13th Inter. Joint Conf. on AI, San Mateo, CA, Morgan Kaufmann (1993) 1002–1007
19. Murthy, S.K., Kasif, S., Salzberg, S.: A system for induction of oblique decision trees. Journal of Artificial Intelligence Research **2** (1994) 1–33
20. Murthy, S.K., Kasif, S., Salzberg, S., Beigel, R.: OC1: Randomized induction of oblique decision trees. In: Proceedings of the Eleventh Nat. Conf. on AI, Washington, DC, MIT Press (1993) 322–327
21. Utgoff, P.E., Brodley, C.E.: Linear machine decision trees. Technical report, University of Massachusetts (1991) COINS Technical Report 91-10.
22. Y.-J. Lee, Mangasarian, O.L.: SSVM: A smooth support vector machine. Computational Optimization and Applications **20** (2001) 5–22 Data Mining Institute, University of Wisconsin, Technical Report 99-03.