Univariate Margin Tree

Olcay Taner Yıldız

Department of Computer Engineering, Işık University, TR-34980, Şile, Istanbul, Turkey, olcaytaner@isikun.edu.tr

Abstract. In many pattern recognition applications, first decision trees are used due to their simplicity and easily interpretable nature. In this paper, we propose a new decision tree learning algorithm called univariate margin tree, where for each continuous attribute, the best split is found using convex optimization. Our simulation results on 47 datasets show that the novel margin tree classifier performs at least as good as C4.5 and LDT with a similar time complexity. For two class datasets it generates smaller trees than C4.5 and LDT without sacrificing from accuracy, and generates significantly more accurate trees than C4.5 and LDT for multiclass datasets with one-vs-rest methodology.

1 Introduction

Machine learning aims to determine a description of a given concept from a set of examples provided by teacher and from the background knowledge. Learning examples can be defined as positive or negative for a two class problem. Background knowledge contains the information about the language used to describe the examples and concepts. For instance, it can include possible values of variables and their hierarchies or predicates. The learning algorithm then builds on the type of examples, on the size and relevance of the background knowledge, and on the representational issues [1], [2].

Decision trees are one of the well-known learning algorithms in Machine learning. They are tree-based structures which consist of internal nodes having one or more attributes to test and leaves to show the decision made. The type of the split determines the type of the decision tree. In univariate decision trees, the split is based on one attribute. If that attribute is continuous, there will be two children of each internal node (C4.5) [3], if that attribute is discrete, there will be L children of each internal node corresponding to the L different outcomes of the test (ID3) [4].

Linear discriminant tree (LDT) [5] is another univariate decision tree technique which uses a statistical approach to quickly determine the best split. Finding the best split with Fisher's Linear Discriminant Analysis (LDA) [6] is done as a nested optimization problem. In the inner optimization problem, Fisher's linear discriminant is used for finding a good split for the given two distinct groups of classes. In the outer optimization problem, at each node m, one searches for the best separation of K classes into two groups, C_m^L and C_m^R [7].

E. Gelenbe et al. (eds.), *Computer and Information Sciences*, Lecture Notes in Electrical Engineering 62, DOI 10.1007/978-90-481-9794-1_3,

Maximum margin classifiers [8], especially support vector machines (SVM), became popular in the recent years for solving problems in classification, regression, and one class classification. Maximum margin classifiers (i) approach the classification problem through the concept of *margin*, which is defined to be the smallest distance between the decision boundary and the closest data points (called support vectors) (ii) determine the model parameters by setting up a convex optimization problem, (iii) use hinge loss instead of misclassification error, and (iv) usually work for two-class problems.

In this paper we propose a novel decision tree classifier which finds the best split for each attribute at each decision node using convex optimization. Our simulation results on 47 datasets from UCI repository [9] show that our univariate margin tree classifier performs better than C4.5 and LDT in terms of accuracy and tree size.

The paper is organized as follows: In Section 2 we present and discuss our proposed Univariate Margin Tree algorithm. We present the experimental setup and results in Section 3 where we compare in detail our proposed algorithm with C4.5 and LDT. Section 4 gives the conclusions and discusses possible future directions.

2 Univariate Margin Tree

We consider the well-known supervised learning setting where the learning algorithm uses a sample of N labeled points $S = ((\boldsymbol{x}^1, y^1), \ldots, (\boldsymbol{x}^N, y^N)) \in (X \times Y)^N$, where X is the input space and Y the label set, which is $\{-1, +1\}$. The input space X is a continuous vectorial space of dimension d, the number of features. The data pairs $(\boldsymbol{x}^i, y^i)$ are independently and identically distributed according to an unknown but fixed distribution.

In training a univariate test, one tries to find the best feature x_j and the threshold w_0, namely the split $x_j + w_0 \geq 0$ that best separates two (or K) classes. To separate K classes as good as possible, C4.5 tries to minimize the impurity or maximize the information gain, LDT assumes the two class groups are normally distributed and tries to maximize the ratio of between class distance to within class distance. In this paper, we take the wide margin approach and express finding best split as a minimization problem. To find the best threshold for feature j, we require

$$y^t(x_j^t + w_0) \geq C - \epsilon^t \tag{1}$$

for each data point x_j^t with output y^t, where $|C|$ is the length of the margin. If $\epsilon^t = 0$, the instance is on the separating line. If $0 < \epsilon^t < |C|$, the instance is correctly classified but it is in the margin. If $\epsilon^t > |C|$, it is misclassified. Since there is only single variable, and its weight is 1, we only need the penalty term $(\sum \epsilon^t)$ in the objective function. So we get the following formulation:

$$\begin{aligned} Min \quad & \sum_t \epsilon^t \\ s.t.\ & y^t(x_j^t + w_0) \geq C - \epsilon^t \\ & \epsilon^t \geq 0 \end{aligned} \tag{2}$$

This is a linear programming problem in which ϵ^t and w_0 are variables. By adding Langrange multipliers ($\alpha^t \geq 0$ and $\mu^t \geq 0$), the problem can be converted to

$$L(w_0, \epsilon^t, \alpha^t, \mu^t) = \sum_t \epsilon^t - \sum_t \alpha^t [y^t(x_j^t + w_0) - C + \epsilon^t] - \sum_t \mu^t \epsilon^t \tag{3}$$

We can remove the primal variables ϵ^t and w_0 by maximization, i.e. set the following derivatives to zero:

$$\begin{aligned} \frac{\partial L}{\partial w_0} = 0 \Rightarrow \sum_t \alpha^t y^t = 0 \\ \frac{\partial L}{\partial \epsilon^t} = 0 \Rightarrow \alpha^t + \mu^t = 1 \end{aligned} \tag{4}$$

Plugging the equations 4 in the equation 3, we get the dual form:

$$D(\alpha^t, \mu^t) = \sum_t \alpha^t (C - y^t x_j^t) \tag{5}$$

Since we have $\alpha^t \geq 0$, $\mu^t \geq 0$, and $\alpha^t + \mu^t = 1$, it follows $0 \leq \alpha^t \leq 1$. Now the dual optimization problem becomes

$$\begin{aligned} Max & \sum_t \alpha^t (C - y^t x_j^t) \\ s.t. & \sum_t \alpha^t y^t = 0 \\ & 0 \leq \alpha^t \leq 1 \end{aligned} \tag{6}$$

Neither the primal (Equation 2) nor the dual (Equation 6) is easily solvable. To solve the problem, we search all possible solutions in terms of the split point w_0 exhaustively. The inequality $y^t(x_j^t + w_0) \geq C - \epsilon^t$ can be written as

$$\epsilon^t \geq C - x_j^t y^t - w_0 y^t \tag{7}$$

According to the convex optimization theory, the solution to a convex optimization problem (if there is a solution) lies in one of the vertices of the convex polytope and each vertex is specified by a set of $n + 1$ inequalities tight, where $n + 1$ represents the number of distinct variables. Setting the left side of the Equation 7 to zero ($\epsilon^i = 0$) gives us all possible values for $w_0 = \dfrac{C - x_j^t y^t}{y^t}$. Then the only remaining thing is to calculate other ϵ^j's and check for the maximum value of the objective function $\sum_t \epsilon^t$.

The pseudocode for finding the best split at each decision node of the univariate margin tree is given in Figure 1. For each feature i we search for the optimal w_0 exhaustively (Line 2). Given the feature i, there exists at most N different w_0's corresponding to N different inequalities shown in Equation 7, where each time a different ϵ^t is zero (Line 4). To get the minimum value of the penalty

```
Split UnivariateMarginTreeBestSplit(N, d, S, V)
1  for C = -2.0 to 2.0 step 0.1
2      for i = 1 to d
3          for j = 1 to N
4              w_0 = (C − x_i^j y^j) / y^j ; sumepsilon = 0.0
5              for k = 1 to N
6                  ε^k = C − x_i^k y^k − w_0 y^k
7                  if ε^k > 0 then sumepsilon += ε^k
8              if sumepsilon < minepsilon then best w_0 = w_0
9          error = ErrorOfSplit(x_i + best w_0 ≥ 0, V)
10         if error < bestErr then bestErr = error; bestSplit = x_i + best w_0 ≥ 0
11 return bestSplit
```

Fig. 1. The pseudocode of the search algorithm for finding the best split at each decision node of the univariate margin tree: N: Number of examples at the decision node, d: Number of inputs in the dataset, $S = ((\boldsymbol{x}^1, y^1), \ldots, (\boldsymbol{x}^N, y^N))$: Sample of N labeled data points at the decision node, V: Validation data used to optimize C value

term $\sum_t \epsilon^t$ for a specific w_0, we set ϵ^k to zero if $C - x_i^k y^k - w_0 y^k < 0$ and to $C - x_i^k y^k - w_0 y^k$ if $C - x_i^k y^k - w_0 y^k > 0$ (Line 7). The threshold w_0 corresponding to the minimum overall penalty will be selected as a best split candidate (Line 8). Similar to the support vector machines, to optimize the length of the margin C, we need a separate validation set. The instances are normalized in order to use the same C value for each feature. We calculate the error rate of the current best split candidate on the validation set and compare it with the best error so far (Line 10). We return the split with minimum error as the best split.

For a given data size N and dimension d, the computational complexity of the algorithm is $\mathcal{O}(dN^2)$. But like C4.5, one can sort w_0's ($\mathcal{O}(N \log N)$) and calculate the minimum overall penalty in $\mathcal{O}(N)$ time resulting in a lower computational complexity of $\mathcal{O}(dN \log N)$, which is the same as C4.5's complexity.

3 Experiments

In this section, we compare the performance of our proposed univariate margin tree algorithm (UMT) with C4.5 and LDT in terms of generalization error and model complexity as measured by the number of nodes in the decision tree. We use a total of 47 data sets where 36 of them are from UCI repository [9] and 11 are bioinformatics cancer datasets [10].

Our methodology in generating train, validation and test sets is as follows: A data set is first divided into two parts, with 1/3 as the test set, *test*, and 2/3 as the training set. The training set is then resampled using 2×5 cross-validation to generate ten training and validation folds, tra_i, val_i, $i = 1, \ldots, 10$. tra_i are used to train the decision trees and val_i are used to prune the decision trees using

Table 1. The average and standard deviations of error rates (tree size) of decision trees generated using C4.5, LDT, and UMT for $K = 2$ class datasets

Dataset	**C4.5**		**LDT**		**UMT**	
	Error Rate	Tree Size	Error Rate	Tree Size	Error Rate	Tree Size
ads	**3.4±0.4**	38.8±19.8	3.7±0.4	50.2±21.6	3.9±0.3	**31.0±11.8**
breast	6.9±1.2	11.5±7.7	6.5±0.5	10.0±5.7	**6.3±1.5**	**7.6±3.4**
bupa	**38.5±4.4**	18.7±15.1	40.8±4.4	13.3±16.3	42.0±5.2	**8.2±7.0**
dlbcl	23.3±8.9	3.4±1.9	25.6±5.4	2.5±2.1	**21.5±7.2**	**1.9±1.5**
german	29.4±1.2	**4.3±7.0**	29.3±1.4	10.9±20.9	**28.2±1.6**	9.1±8.7
haberman	26.4±0.3	4.6±8.7	27.3±2.8	4.6±8.7	**26.2±0.9**	**2.2±3.8**
heart	30.9±4.0	9.4±5.4	**29.0±4.8**	8.2±5.1	32.4±2.8	**4.9±2.9**
hepatitis	22.3±4.0	10.9±9.7	21.7±1.3	**2.5±3.2**	**19.6±2.5**	4.9±4.5
ironosphere	14.2±4.4	**11.8±6.7**	14.9±4.5	12.4±9.0	**11.3±5.9**	12.1±5.8
magic	**17.1±0.4**	86.8±28.5	17.7±0.3	121.3±44.4	19.2±0.4	**63.4±13.3**
musk2	**4.7±0.6**	114.1±25.2	5.3±0.6	131.5±28.4	12.7±1.2	**24.4±11.1**
parkinsons	15.4±4.5	9.1±5.8	15.5±5.4	8.8±4.1	**14.9±3.6**	**5.8±3.5**
pima	28.9±2.7	15.1±13.3	28.4±4.3	23.8±24.8	**26.3±1.3**	**7.6±5.1**
p.adenylation	30.6±2.1	38.5±20.1	**29.6±1.7**	**54.7±37.4**	29.7±1.3	81.7±28.4
p.tumor	**15.7±2.4**	5.2±2.1	20.3±10.6	4.9±2.0	22.0±11.4	**3.7±1.0**
ringnorm	**12.0±0.7**	157.0±44.7	23.1±1.2	261.1±67.4	15.1±2.0	**70.9±11.0**
satellite47	14.6±1.3	40.6±16.4	15.2±1.2	27.4±20.3	**14.6±0.8**	**26.5±9.7**
spambase	9.3±1.2	79.9±31.8	**9.2±0.8**	106.6±38.9	10.1±0.6	**54.7±17.7**
transfusion	24.0±0.0	**1.0±0.0**	**23.8±0.8**	1.9±2.9	24.0±0.0	**1.0±0.0**
twonorm	**17.5±0.6**	225.1±45.0	17.7±0.7	248.8±51.3	19.1±0.7	**171.7±23.6**

cross-validation based postpruning. *test* is used to estimate the generalization error of the decision trees.

Table 1 shows average and standard deviations of error rates (tree size in terms of number of nodes) of decision trees generated using C4.5, LDT, and UMT for $K = 2$ class datasets. We see from the results that, UMT is as good as C4.5 and LDT in terms of error rate. In 9 (10) datasets out of 20 UMT has smaller error rate than C4.5 (LDT). On the other hand, C4.5 (LDT) has smaller error rate than UMT in 10 (10) datasets out of 20.

Better than the results above, UMT is significantly better than both C4.5 and LDT in terms of tree complexity. In 16 (18) datasets out of 20 UMT generates smaller trees than C4.5 (LDT). Whereas, C4.5 (LDT) generates smaller trees than UMT in only 3 (2) datasets out of 20. We can conclude that for two class datasets, UMT generates smaller trees than C4.5 and LDT without sacrificing from accuracy.

In UMT, for $K > 2$ class problems, we take the most commonly used approach and reduce the single multiclass problem into multiple binary classification problems. UMT uses two well-known methods to build binary classifiers where each classifier distinguishes between (i) one of the labels to the rest (one-versus-rest) or (ii) between every pair of classes (one-versus-one). The results show that, UMT (one-vs-one) is as good as C4.5 and LDT in terms of error rate.

In 13 (12) datasets out of 27 UMT (one-vs-one) has smaller error rate than C4.5 (LDT), whereas C4.5 (LDT) has smaller error rate than UMT (one-vs-one) in 14 (15) datasets out of 27. On the other hand, UMT (one-vs-rest) is significantly better than C4.5 and LDT in terms of error rate. In 19 datasets out of 27 UMT (one-vs-rest) has smaller error rate than C4.5 and LDT, whereas C4.5 and LDT have smaller error rate than UMT (one-vs-rest) only in 8 datasets out of 27.

4 Conclusion

In this paper, we propose a new decision tree learning algorithm called univariate margin tree, where the best split is found using convex optimization at each decision node. The main idea comes from simplifying the formulation of multivariate linear support vector machines. Simplification is done by (i) replacing the multivariate discriminant $\boldsymbol{w}$ with the univariate axis-orthogonal split $x_j + w_0 \geq 0$, (ii) removing the $||\boldsymbol{w}||^2$ factor from the objective function since there is only single variable x_j, and (iii) redefining the margin in the one-dimensional space. None of the resulting primal and dual formulations are easily solvable, therefore we resort searching for the optimal w_0 exhaustively. Although the first-come to mind exhaustive search is expensive, with the same trick applied in C4.5, one can get a computational complexity of $\mathcal{O}(dN \log N)$, which is much better than the usual time complexity of the traditional support vector machines.

Experimental results show that for two class problems our proposed univariate margin tree not only performs as good as C4.5 and LDT in terms of accuracy, but it produces significantly smaller trees to do that. For multiclass problems, we tried two well-known reduction techniques, namely, one-vs-one and one-vs-rest. In terms of generalization error, one-vs-rest is significantly better than one-vs-one technique, which is also as good as C4.5 and LDT.

References

1. Mitchell, T.: Machine Learning. McGraw-Hill (1997)
2. Alpaydın, E.: Introduction to Machine Learning. The MIT Press (2010)
3. Quinlan, J.R.: C4.5: Programs for Machine Learning. Morgan Kaufmann, San Meteo, CA (1993)
4. Quinlan, J.R.: Induction of decision trees. Machine Learning **1** (1986) 81–106
5. Yıldız, O.T., Alpaydın, E.: Linear discriminant trees. International Journal of Pattern Recognition and Artificial Intelligence **19**(3) (2005) 323–353
6. Duda, R.O., Hart, P.E., Stork, D.G.: Pattern Classification. John Wiley and Sons (2001)
7. Guo, H., Gelfand, S.B.: Classification trees with neural network feature extraction. IEEE Transactions on Neural Networks **3** (1992) 923–933
8. Vapnik, V.: The Nature of Statistical Learning Theory. Springer Verlag, New York (1995)
9. Asuncion, A., Newman, D.J.: UCI machine learning repository (2007)
10. Statnikov, A., Aliferis, C., Tsamardinos, I., Hardin, D., Levy, S.: A comprehensive evaluation of multicategory classification methods for microarray gene expression cancer diagnosis. Bioinformatics **21** (2005) 631–643