CS 412 – Introduction to Data Mining

Assignment 5

Problem 1:

We need to modify the decision tree method to predict the age of the user, which is a continuous predictor and hence we would use a regression tree to achieve the same.

SPLIT:

To partition the data while building a decision tree top down from the root, we wish to maintain homogeneity in the sample partitioned together. Now, to compute the homogeneity of a sample, we can use standard deviation as a metric since standard deviation for a pure (homogenous) sample is 0.

Hence, to decide the attribute to split on in regression trees, rather than information gain (used in classification trees when the target variable is discrete), we use **Standard Deviation Reduction**.

Like information gain, Standard deviation reduction for an attribute is calculated as the decrease in standard deviation once we split the dataset on the selected attribute.

The steps to compute the Reduction in standard deviation are listed below:

Step 1: Compute the Standard Deviation of the target variable (age in our case)

Step 2: Compute the Standard Deviation after splitting the dataset based on an attribute

SD (Target, Attribute) = $\sum_{class \in Attribute} Prob(class)SD(class)$

where *Prob(class)* and *SD(class)* denote the probability and Standard deviation of the label (age in our case) of a class of the attribute in consideration and *SD(X)* = $\sqrt{\sum_{i=1,..,n} \frac{(x_i - \mu)^2}{n}}$

Step 3: Compute the standard deviation reduction for the attribute by subtracting the quantities obtained in Step 1 and 2.

SD Reduction (Target, Attribute) = SD (Target) - SD (Target, Attribute)

Our objective is to select an attribute to split which has the highest standard deviation reduction.

Step 4: Select an attribute which reduces the standard deviation the most when split on.

STOPPING CRITERIA:

The splitting is continued recursively until the one of the following stopping conditions holds true:

1. There are no more training samples that can be assigned to a branch.
2. There are no remaining attributes present to partition the branches further.
3. There are too few data points left in a branch to split on.
4. Standard deviation of a branch becomes smaller than some threshold of the standard deviation of the target variable. In other words, we continue to split until the standard deviation reduction in one of the branches is below a threshold.

PREDICTION:

Next, for predicting the target value (age) of a branch, we calculate **the mean of target values (ages) of all the training set samples that fall in that branch.**

Let each path on the decision tree from the root to leaf define a space say S_k. Then for all the training set sample points X_k that fall in the space S_k, we take the average of the target (age) value Y_k as the prediction of the branch.

$$Y_k{}^{Predicted} = \sum_{i \in S_k} \frac{Y_i \ (Age\ at\ the\ sample\ i)}{No.of\ training\ samples\ in\ S_k}$$

COST FUNCTION:

The aim of the regression trees is to minimize the total least square error by making predictions for each branch i.e. $Minimize_{S_{k|\ k=1\ to\ N}} \sum_{i \in S_k} \frac{(Y_i - Y_k{}^{Predicted})^2}{No.of\ training\ samples\ in\ S_k}$

Hence, to use the binary decision tree for continuous output variable say age, we need to do the following changes:

1. Use **reduction in Standard Deviation** as a metric to decide on the attribute to split on rather than information gain.
2. Use the **mean of the target values** for the training samples falling in a branch of the decision tree to predict the age for a new sample in the test set.
3. The aim is to **minimize the total least square error** by choosing appropriate splits that divide the training samples across the tree branches.

Problem 2:

I agree with the fact that ensemble of multiple classifiers increases the classification accuracy when each of the classifier's accuracy is better than random guess, i.e. better than 0.5.

This is because since each classifier's accuracy is better than 0.5, hence, each of their error rate, say e_i is less than 0.5, and hence error proportion of the ensemble of all these classifiers, say n in number would be $\prod_{i=1}^{n} e_i$.
Since each e_i is less than 0.5, hence the **error rate for the ensemble model $\prod_{i=1}^{n} e_i$ would be smaller than any individual e_i.** Hence, we can conclude that if each classifier is better than a random guess, then the accuracy for the ensemble model would surpass the individual model.

Also, while using an ensemble model, the classifiers are trained on **distinct subsamples** of the same data set sampled with replacement, hence it efficiently reduces the probability of getting stuck in the same noise as a single classifier may have. This makes the ensemble models much more robust to the noisy data and prevents overfitting as well. The ensemble of models also reduces the variance of the individual classifiers, which in turn helps in increasing the model's accuracy.
The majority vote on the classifications on separate subsamples of data are less likely to be stuck in the same inaccuracy and be wrong.
Also, if we use boosting where the mistakes of the previous classifier are given more importance to be improved upon by the next classifier, the accuracy can further be enhanced. In boosting, if a tuple is incorrectly classified, we increase its weight. **Also, the weights of the classifiers that perform better than others are increased which even enhances the performance of the boosted ensemble model.** Hence, if we further control the weights of the classifiers depending on their performance, the accuracy of the ensemble model is even better.

Hence an ensemble model is better than a single classifier with accuracy higher than 0.5 since the cumulative error ratio $\prod_{i=1}^{n} e_i$ would be smaller than an individual error value e_i of a classifier and training on distinct subsamples of the same data set sampled with replacement will significantly reduce the probability of the classifier getting stuck in the same inaccuracy again, hence taking a majority vote would nullify the effect of one classifier that may have classified incorrectly.

Problem 3:

The given data set includes 100 samples with 10,000 binary attributes with binary class labels.

Since the class labels and attributes for the given problem are binary, and all the three algorithms namely Decision trees, Naïve Bayes' and SVM can handle binary classification problems with binary attributes, we can in theory use all the three algorithms for the mentioned problem.

Two main considerations while selecting a classification algorithm is the high dimensionality of the data set since it consists of 10,000 features and the small size of the data set since it consists of only 100 samples.

1.

First considering the Naïve Bayes algorithm, it assumes that the features are independent but clearly, it may not be the case for the given problem where different genes of the tissues may be correlated. **Because of the independence assumption for the Naïve Bayes' classifier, it is not a very good choice of algorithm when the feature space, in our case various genes, are correlated.**

Also, since the training set is small, consisting of only 100 samples, it is possible that sample in the test data set may belong to a class which might not have been observed in the training set, in which case the Naïve Bayes' classifier (without using any smoothing) would assign a probability of zero to such a class, which may also be one of the shortcomings in this case.

2.

Next considering decision trees, since the given **data set is small consisting of only 100 training samples, it is likely that it may overfit since the greedy divide and conquer algorithm is used to construct decision trees.**

Also, since the **feature space of the sample is huge, it is very expensive to create branches for these many features in the decision tree**, and hence the training time would be high and memory efficiency would be very low, making the whole algorithm very inefficient due to the complexity of the model.

Also, the complexity of the model can further be enhanced if the genes present are very correlated, since decision trees are efficient in handling dependencies in features but **not very efficient in handling very complex correlations** between the same. So, it will be a bad idea to use decision trees if the genes are highly intermingled and correlated with complex inter dependencies existing between the same.

It is although a **promising idea to ensemble the decision tree** and use say random forest for the given high dimensional problem to handle the large feature space.

3.

Next considering SVM, since its' complexity is determined by the no. of support vectors present and not the dimensionality of the feature space, which is huge in the given data set, it seems like an appropriate choice for the given problem. The no. of support vectors is, in practice, very low compared to the no. of the features present in the data set since they correspond to only the training instances that lie closest to the maximum margin hyperplane. The evaluation of the trained model is fast due to the same reason.

Since the no. of training samples are low, even the training time for the SVM classifier would be low (which is high when the training set is large). Since, only the support vectors are used for constructing the maximum margin hyperplane, SVM works well with the small training sets, as ours with 100 samples only.

Hence, considering the small data set size and large feature space, SVM seems like a perfect option with most probability of resulting in a robust and higher accuracy model with less overfitting. Hence, the time complexity and the memory required for the same would also be very reasonable.

Hence, for the given problem with small data set and high dimensionality, I would prefer to use SVM since it will be robust, accurate, time and memory efficient, because of its' nature to work with only the support vectors and not the entire feature space.

Submitted by:

Rachneet Kaur

Net ID: rk4