CSL603 - Machine Learning : Lab 1

Shivam Mittal
2015CSB1032
2015csb1032@iitrpr.ac.in

September 4, 2017

1 Introduction

This task of this assignment was to classify the sentiment of movie reviews. Formally, the task was to assign a positive label for a movie review with positive sentiment ($rating >= 7$) and a negative label for a movie review with negative sentiment ($rating <= 4$).
We approached the problem using a supervised learning algorithm by using decision trees and decision forests. ID3 algorithm has been implemented to form the decision tree from the training dataset. The accuracy is measured on a test dataset.
Then, we improve the test accuracy by various techniques such as, using some heuristic within the information gain, early stopping criteria to avoid overfitting, post-pruning strategy, and making a decision forest using feature bagging.

2 Experiment 1 - Preprocessing

2.1 Sampling data

The Large Movie Review Dataset from Stanford was used for running our experiments. The dataset is very large, so we select randomly 1000 examples from the large training dataset as our training in which 500 examples are positive labelled and other 500 are negative labelled. We select another 1000 examples (distinct from those in the small training data) from the large training dataset as validation set, and a 1000 more examples from the large test daataset as our test dataset. We make a data file of these 3000 examples as our data for all the different experiments.

2.2 Sampling attributes

The Movie Review Dataset also contains a imdb vocabulary, which is used to create a representation for each review, using a bag of words model. These words are basically the features/attributes that characterize a review, and these

features will be used to create the decision tree. Since the number of features are very large (89k+), we will sample 5000 features from the large vocabulary. We used **Stochastic Hill Climbing** to generate the selected sample of features.
The file imdbEr.txt contains the average polarity of all the words given in the imdb.vocab file. While experimenting, we choose the top 2500 words with maximum positive average polarity, and also the top 2500 words with maximum (magnitude) negative average polarity. This is a good choice, because if the word which has high average polarity, when it is used as a split node, there are high chances that one of a child will get split in such a way that it contains examples with one label class (minimum entropy). But while running the experiment, if these attributes are used as split nodes, the proportion of split is very uneven because most of the reviews do not even contain a single occurrence of these words.
So, a observation was made that some of the words in the first 5000 words are common words (although not discriminative), but they split the nodes such that the information gain is high, and there is fairly uniform splitting so that the size of the subproblem in the child becomes fairly small.
To combine the advantages of both these attributes, we use stochastic hill climbing and will make a heuristic in the information gain to yield good results. We select 5000 words with extreme average polarities. Then depending on a walk probability (0.8 taken after tuning the parameter empirically), we will choose a attribute from the first 5000 with walk_probability=0.8 and choose the next extreme average polarity attribute with probability=0.2.
We generate a selected-features-indices.txt file, which contain the attributes selected using the strategy explained above.

3 Experiment 2 - ID3 algorithm

The general ID3 algorithm was implemented. The two basic things we had to decide were, choosing the best attribute as a split node, discretizing or choosing a threshold for splitting from the split node.

3.1 Splitting strategy

For any attribute, calculate the average frequency (number of times that word occurs) among the positive examples and store in positive-average, and similarly calculate the average frequency among the negative examples and store in negative-average.
Then, threshold1=min(positive-average, negative-average) and threshold2=max(positive-average, negative-average). Then the data will be splitted into 3 node, depending of the frequency of that attribute in the example.
If $frequency <= threshold1$, example goes to child1,
Else if $frequency > threshold2$, example goes to child3,
Else, example goes to child2.

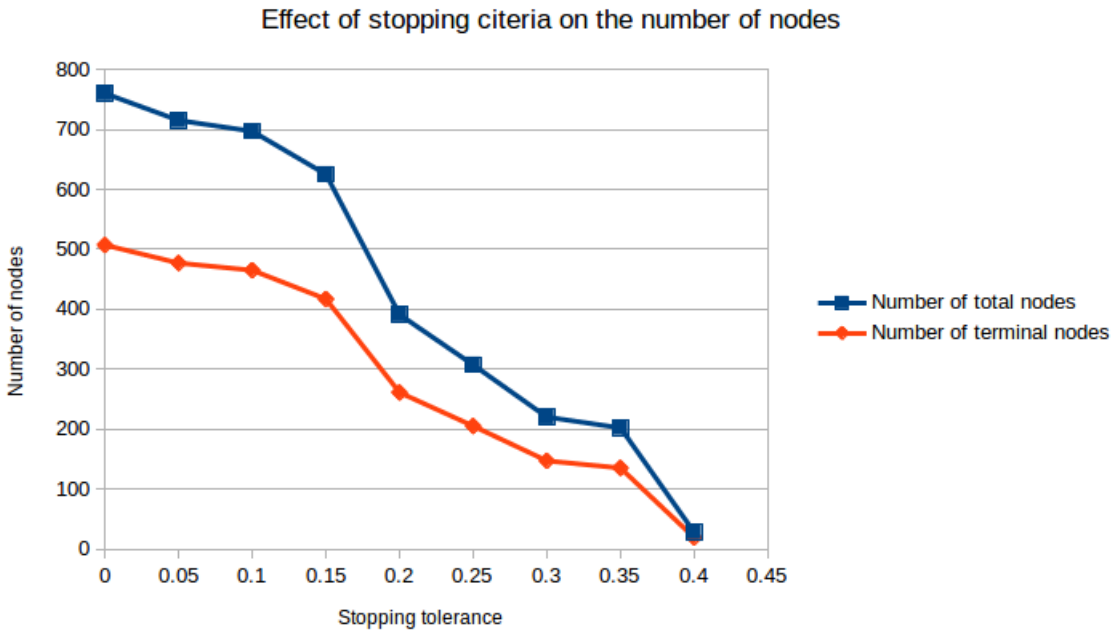
3.2 Choosing the best attribute

Since, we have a mix of attributes which contains the first 5000 attributes and also the extreme average polarity attributes.
When using just the information gain heuristic for choosing the best attribute, the extreme polarity attributes never get chosen, which might be able to better classify the data but they are not chosen because according to our splitting strategy, only one of the child gets the majority examples and other children get a very few examples. So, we add multiply the information gain for the extreme average polarity attributes by a constant factor chosen empirically, so that they are favoured a little in comparison to the first 5000 attributes.
But, this factor is multiplied only for nodes when the height of the tree exceeds 8 (chosen empirically by parameter tuning), because we need that the size of the sub-problem gets reduced (which is done efficiently by the first 5000 attributes), because calculating information gain for 1000 attributes again and again is very expensive.
The overall algorithm is, to calculate the attribute with highest information gain which are from the first 5000, and store in a and its information gain in ig_a. Then select the attribute with highest information gain which are from the extreme average polarity, and store in b and its information gain in ig_b. Multiply ig_b by a bias factor. If depth of the node < 8, then choose a and return, else if ig_b $>$ ig_a, choose b.

3.3 Effect of early stopping criteria

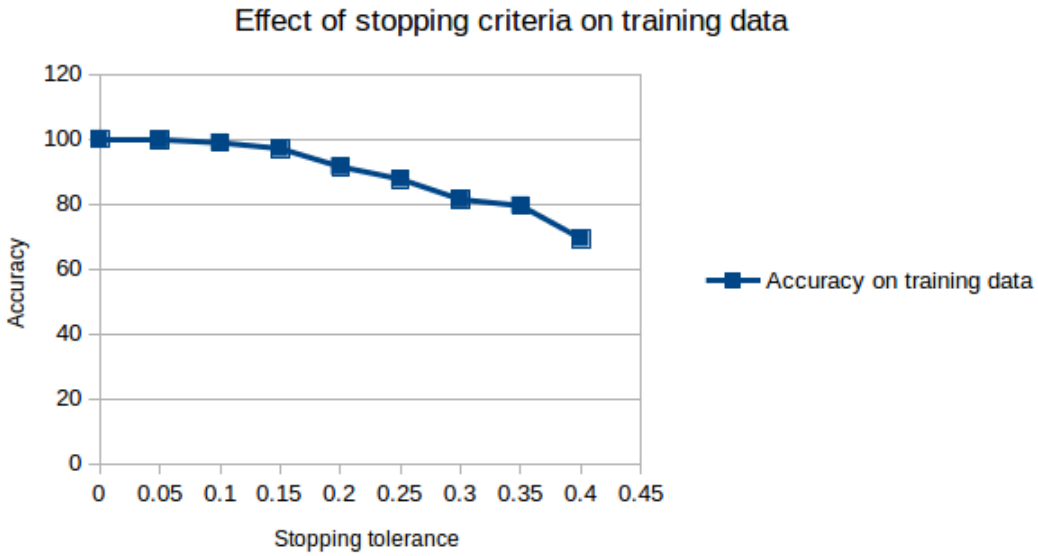
We have a stopping_tolerance, the stopping criteria with respect to the stopping tolerance is :
if (proportion of positive_examples $>$ (1-stopping_tolerance)), label positive and stop.
if (proportion of positive_examples $<$ (stopping_tolerance)), label negative and stop.

3.3.1 Effect on number of nodes

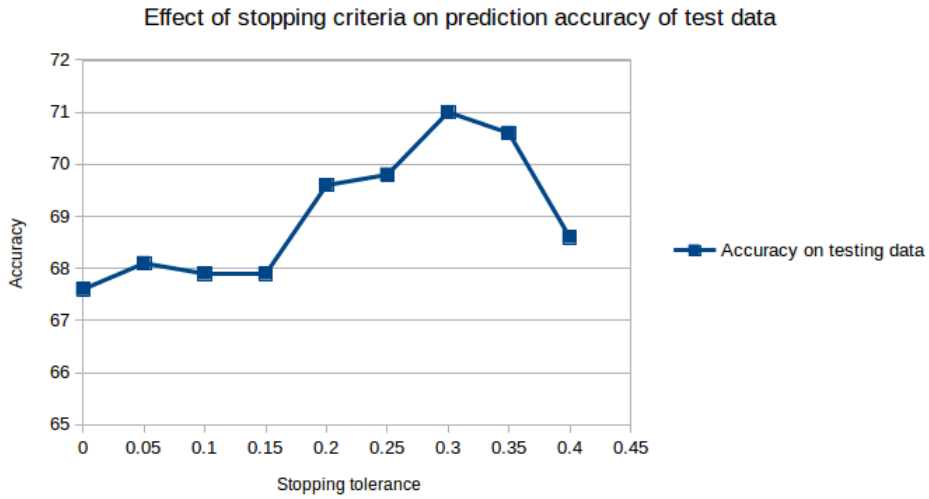


As we increase the stopping tolerance, the number of nodes (total and terminal) will decrease, because the nodes which would have been further expanded by choosing an attribute and so on, are made leaf nodes because they fulfill the stopping criteria earlier that either the proportion of positive examples, or the proportion of negative examples is less than the stopping tolerance. More the stopping tolerance, earlier the nodes are made terminal nodes, hence the trend shown.

3.3.2 Effect on prediction accuracy



As we increase the stopping tolerance, the prediction accuracy on the training set will decrease because at a particular node, where we have mixed positive and negative labelled examples, instead of further classifying them, we are making it a leaf node, and assigning a common label to them. So, its obvious that we are mislabelling some of the training data, hence the accuracy on the training data will reduce.



As we increase the stopping tolerance, the prediction accuracy on the test set will generally increase (upto a particular threshold of stopping tolerance) because early stopping is preventing to overfit the data. Instead of trying to fit the training data completely, we're making little generalization and improving the accuracy on the test dataset. But, if we keep on increase the stopping tolerance, after a point the prediction accuracy will decrease, because we making a lot of generalization and no real splitting or classification is actually being done.

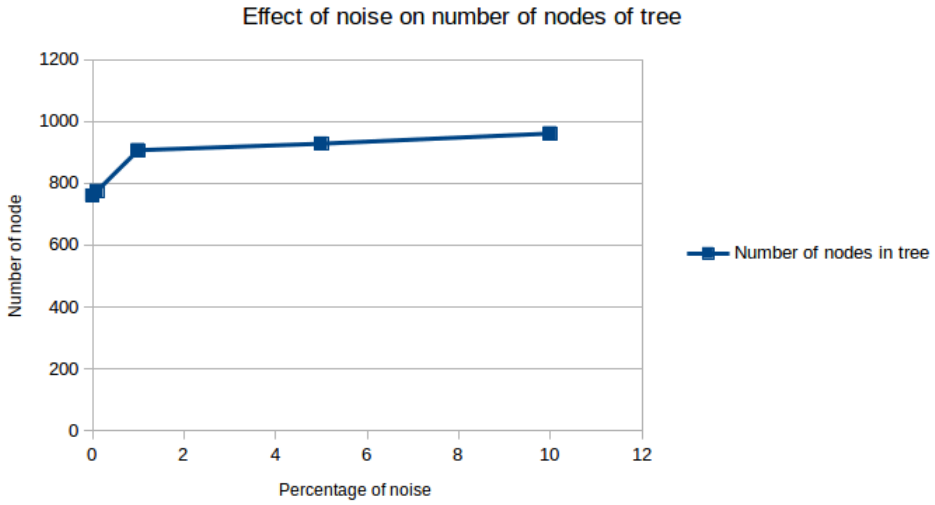
3.4 Attributes most frequently used as attributes as split function

The attributes that are most frequently used as attributes are calculated by making 4 different trees by varying the stopping tolerance. Then the frequency of the attributes that are used as split function is calculated in each tree, and the average is taken. Then the list is sorted in decreasing order, and the attributes used most frequently are displayed.

4 Experiment 3 - Adding noise

We add noise to the training data by randomly switching the labels.

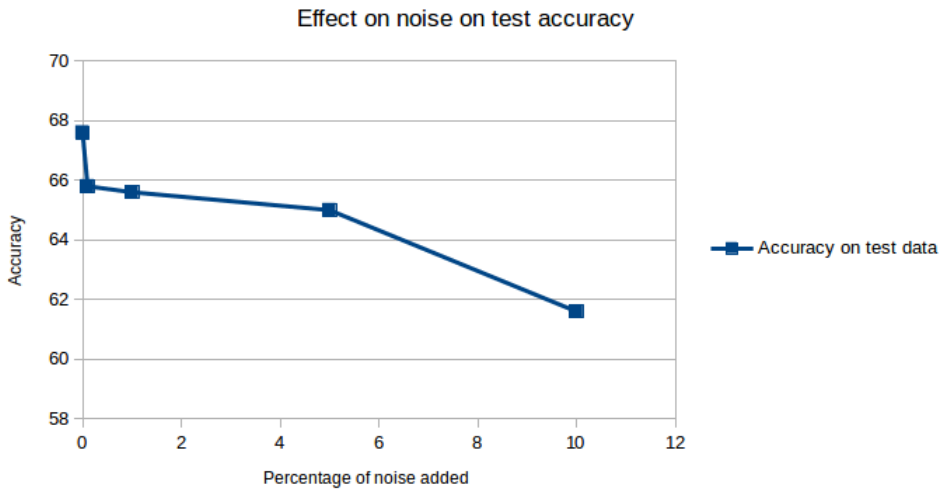
4.1 Effect on number of nodes



As we know that noisy data is a very common cause of over-fitting, because mislabeled instances may contradict the class labels of other similar records. Decision tree tries to achieve maximum accuracy on the training dataset. So to

be able to classify such noisy training data, it will increase the complexity of the tree to even classify the mislabelled instance.

4.2 Effect on prediction accuracy

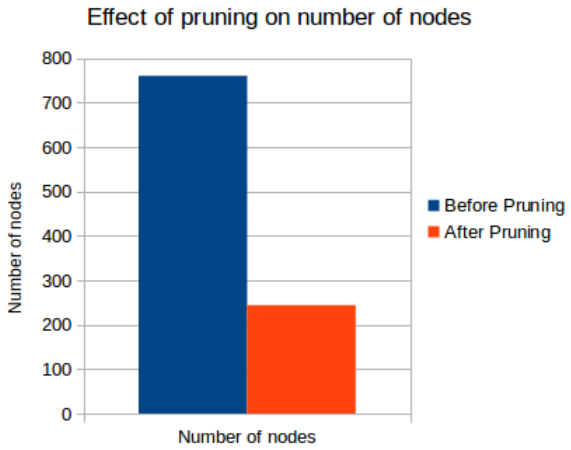


The prediction accuracy on the test dataset generally decreases if there is noisy training data. This is because the decision tree learned from the training data is overfitted on the training data (including the noise in it). The tree learns to classify mislabelled data correctly, so it will misclassify similar data in test set. It can be seen from another perspective, that generally a shorter learned tree gives better accuracy than a larger learned tree.

5 Experiment 4 - Pruning tree

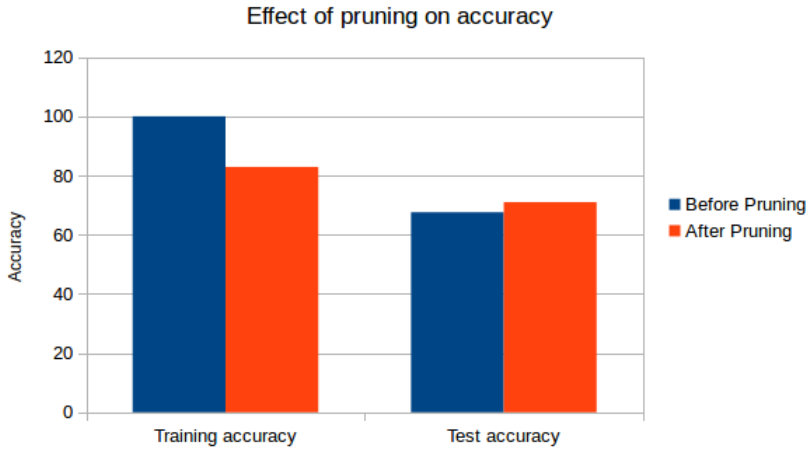
We use a post pruning strategy, we first generate a tree with no stopping criteria. Then to do pruning, we use the validation data. We perform a post order traversal of the tree, and check if that particular split node is replaced by a leaf node that will be labelled according the majority labelled data, whether the accuracy on the validation data will increase or not. If the accuracy on validation data is increased, we replace that node by a terminal node (subtree rooted at that node is pruned).

5.1 Effect on number of nodes



Due to pruning, the complexity of the tree will decrease (number of nodes decreases). This is obvious because we are removing (pruning) those subtrees which are really not needed, the subtrees which when removed cause the validation accuracy to increase.

5.2 Effect on prediction accuracy

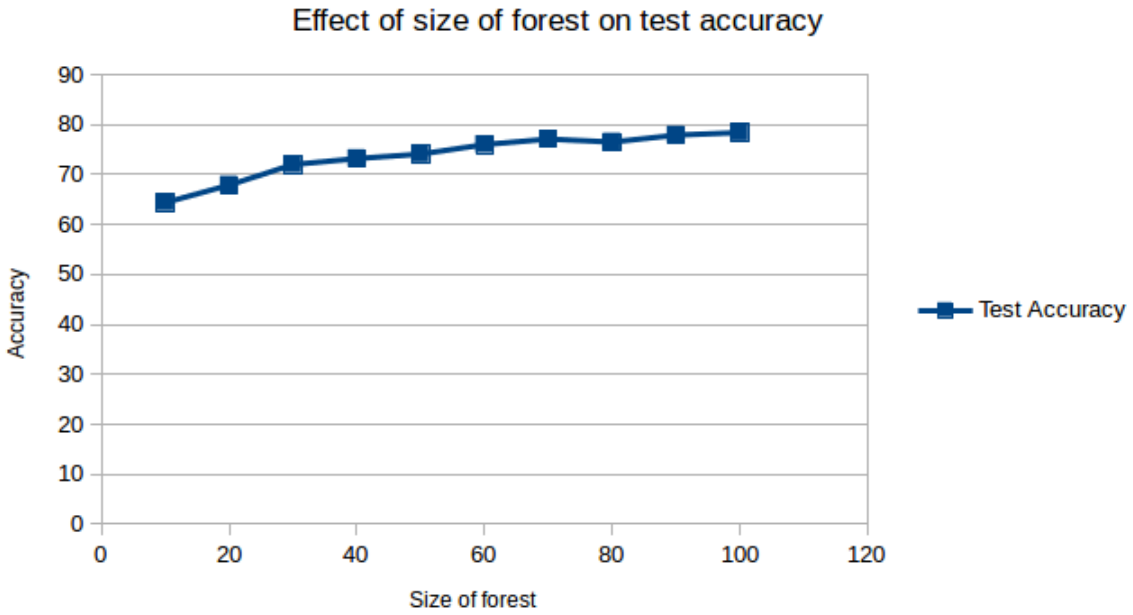


As a result of pruning, the prediction accuracy on the training set will decrease because at a particular node, where we have mixed positive and negative labelled examples, instead of further classifying them, we are making it a leaf node, and assigning a common label to them. So, its obvious that we are mislabelling some of the training data, hence the accuracy on the training data will reduce.
But, the prediction accuracy on the test data set will generally increase, because pruning is reducing the overfitting done in the decision tree and removing the

sub-trees trying provide perfect coverage to training data which is really not needed.
By making the decision tree a little more generalized and also testing on separate data whether the generalization is a improvement or not. Our accuracy on test data generally increases.

6 Experiment 5 - Decision Forests

We make a decision forest (ensemble learning method) by using feature bagging. Decision forests reduces overfitting, because the result is the average of a multiple of trees.
From the set of 5000 features, we randomly select 500 features and learn a decision tree using those features and all examples. Then again, we randomly select 500 features and learn a decision tree using those features. We make multiple such tree, and basically we have a forest. The size of the forests specify the number of trees in it.
Then, for classifying any example, we classify that example by all of the trees, and maintain a count of positive and negative. The count of whichever label is greater, that example is classifies accordingly by majority voting.



Increasing the number of trees in the forest increase the prediction accuracy of the test data, because as we increase the size, the predicted label is the average (majority voting) of an increasing number of trees. As the number of trees increase, overfitting is further reduced, and hence the accuracy on the test set increases.

References

[1] Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and ChristopherPotts. (2011). *Learning Word Vectors for Sentiment Analysis.* The 49th Annual Meeting of the Association for Computational Linguistics (ACL 2011).

[2] Large Movie Review Dataset
`http://ai.stanford.edu/ amaas/data/sentiment/index.html`