5243 Introduction to Data Mining

Assignment 3

Johns Gresham

Introduction

In an effort to build automatic article categorization, search, and graphs, the issue of categorizing and comparing the contents of articles arises. To classify the class of a new article, we can classify articles into groups or topics using classification. This can be used to more easily predict and analyze a document. A KNearestNeighbors and DecisionTree algorithm were fit to the feature vector data created in Assignment 1. The algorithms' scalability and quality was measured to give some insight on the algorithms performance.

External Sources

- Sci-kit learn 0.16.0
    - Provided the *KNeighborsClassifier* and *DecisionTreeClassifier* algorithm and the accuracy score algorithm.
- NLTK 3.0.5
    - *Nltk.tokenize* and *nltk.corpus.stopwords* helped in preprocessing the data.
- NumPy 1.9.2
    - Required *numpy.array* format for Sci-kit learn and has a *numpy.var* function used to find the variance (skew) for a cluster.
- BeautifulSoup4.4.0
    - Used when parsing the article information from the html files.

Data Formatting

Picking up from Assignment 1, we had a set samples and each sample had a feature vector and label. However, the feature vectors were not ready to be input into a clustering algorithm. First, the features had to be a matrix format [#samples, #features] and labels in an array [#samples]. In my case, the features were just the tf-idf value for that sample of every filtered word(non-stop word,

occurs more than once, etc.) that appears in the corpus in an article. If there were 1000 filtered words in the corpus and the article had a few words with non-zero tf-idf values, then only a few of the 1000 features are non-zero. So there were no discrete or string features. Then each feature is normalized over the sample set so certain features with high values aren't orders of magnitude larger than others and every feature is in the range 0 – 1.

Scalability

Here are the scalability and quality measurements for each classifier configuration. Every test was done using 6151 features over 3179 samples.

	KNearestNeighbors	DecisionTree
Offline Cost		
2543 samples	0.74s	1.74s
1907 samples	0.56	1.13s
Online Cost		
2543 samples	26.1s	0.02s
1907 samples	19.2s	0.02s

The most significant difference in scalability between the two algorithms is the offline cost. KNearestNeighbors requires more distance calculations when predicting new samples. The DecisionTree preformed faster by an order of magnitude than all the other times.

Quality

A simple accuracy was used to score how well the algorithms predicted classes of new samples. Both the algorithms preformed better by training on a larger data set by splitting 80% of the data into the training set and the remaining 20% in the test set. This is expected given that there were only 3179 articles with topic labels to use for classification, so the more the training data, the better. However, the decision tree outperforms the k nearest neighbors regardless of the data split.

	KNearestNeighbors	DecisionTree
Accuracy		
80/20 training/test split	59%	73%
60/40 training/test split	55%	70%

Parameter Tuning

KNearestNeighbors k-value

	80/20 training/test split accuracy	60/40 split accuracy
K		
1	59%	55%
2	57%	55%
5	54%	52%
10	53%	50%
25	45%	43%

Discussion

The decision tree algorithm outperformed k nearest neighbors in both scalability and quality measurements. The algorithms produced promising results considering there was a limited data set of only 3179 articles with top labels. There was another decision that had to be made with the topic labels as well, mentioned later. An accuracy of 59% and 73% for neighbors and the tree algorithms respectively could be improved with more data and considered useful for some applications. The online cost of k nearest neighbors' algorithm is significant compared to the offline cost and could be too time consuming.

Secondly, I had to chose how to label the samples from the topic information. For the articles, there were often a number of topics, but for many of the measurements and algorithms I needed a single labeling. I chose the first topic listed. I could've chosen a random one, but that seemed arbitrary as

well. A better solution could be some vector representation of all the topics and find some tolerable distance between vectors under a label.

Parameter tuning was limited to changing the *k* neighbors in k nearest neighbors. With a limited data set, one could guess that a lower *k* value would produce greater accuracy. That is what the results show looking at some values of k between 1 and 25.

File Locations

directory : /home/7/gresham/5243/lab3/

makefile

readme: contains run instructions

report.docx

knear.py

dtree.py