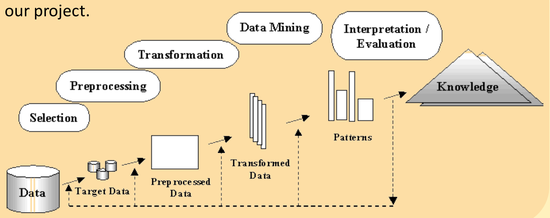
The Beginnings of a Search Engine

Viren Khandal

Stanford Department of Computer Science and Statistics

Abstract

The objective of this project is to determine the relevance of various URL queries based on an array of 12 different predictors. The classifiers tried include Random Forests, Support Vector Machines, Decision Trees, and Naive Bayes. Based on our experimentation, Support Vector Machines had the lowest error and highest accuracy, which I further tuned to maximize the accuracy and minimize the misclassification error of predictions. Larry Page developed a similar but much more complex model that is used in Google and several other search engines. The diagram below shows the process that was followed for the duration of our project.



Introduction

The main goal of a search engine is to optimize URL queries requested by users and return a webpage with links, often in order of relevance. The motivation behind this project was to understand the functions of *R Programming* and innovate a simple search engine based off 12 variables. The four models used in this research project are summarized below.

Decision Trees
- Recursive procedure involving splitting of attributes
- Pruning to avoid overfitting data and maintaining flexibility

Naïve Bayes
- Cancels out irrelevant features to maintain focused model
- Easy accommodation of missing values during probability calculations

Random Forests
- Multiple decision trees combined in one model to reduce overfitting
- Outputs class based on individual trees for higher accuracy

Support Vector Machines
- Maximize margin between positive and negative outcomes
- Use different kernels based on spread/pattern of data

Data Preparation

The *Data Preparation* stage involved Selection, Preprocessing, and Transformation. Before employing any methods onto the training data, we decided to determine which of the 12 variables were significant. Performing a logistic regression, there were 8 variables that were found to be significant due to their low p-values. From this, we were also able to understand that the variables query id and URL id were both highly correlated to each other. Our future model making was done to keep these variables as independent from each other as possible. The main transforming of the data was done using k-fold cross validation. We were able to set up an environment, in which we ran 10-fold cross validation. This cross validation helped us verify that all of our classifiers were working properly and were trained enough to perform well on the test set.

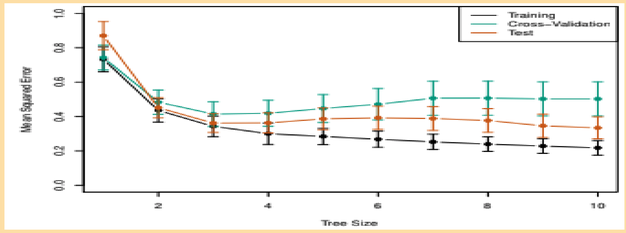
```
> summary(fit_glm)

Call:
glm(formula = relevance ~ ., family = "binomial", data = trainData)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-7.4201  -0.9895  -0.7061   1.1359   2.0723

Coefficients:
              Estimate Std. Error z value Pr(>|z|)
(Intercept) -2.024e+00  5.756e-02 -35.157  < 2e-16 ***
query_id    -1.150e-05  7.610e-06  -1.511   0.1308
url_id       2.055e-06  1.228e-06   1.673   0.0944 .
query_length 1.063e-01  7.375e-03  14.409  < 2e-16 ***
is_homepage -1.797e-01  2.990e-02  -6.009 1.86e-09 ***
sig1         9.578e-01  7.745e-02  12.366  < 2e-16 ***
sig2         3.312e+00  7.369e-02  44.950  < 2e-16 ***
sig3        -8.320e-07  8.022e-07  -1.037   0.2996
sig4        -1.732e-06  2.597e-06  -0.667   0.5047
sig5        -2.771e-06  1.102e-05  -0.252   0.8014
sig6         8.827e-03  5.073e-04  17.400  < 2e-16 ***
sig7         1.254e+00  1.005e-01  12.481  < 2e-16 ***
sig8        -5.863e-01  5.644e-02 -10.389  < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 54852  on 40015  degrees of freedom
Residual deviance: 49782  on 40003  degrees of freedom
AIC: 49808
```

Data Mining

The majority of the project took place in this stage. We tested the performance of 4 different methods and then decided which method to employ on the test data. The 4 methods we chose included Support Vector Machines, Random Forests, Decision Trees, and Naive Bayes. The results of the 4 classifiers are shown in the sub sections below. As Support Vector Machines had the lowest error, we decided to move on with this method and further tune it to decrease the error and maximize the accuracy of our model.

Decision Trees

The decision tree model that I built had an accuracy of approximately 61.5%. This was a significantly low accuracy, therefore I had low hopes for the future tuning I would perform for decision trees. However, I decided to tune the model by omitting certain variables and with pruning. These tuning methods removed the unnecessary parts of the model, allowing it to have more accurate results. The highest yielded accuracy was **64.05%**.



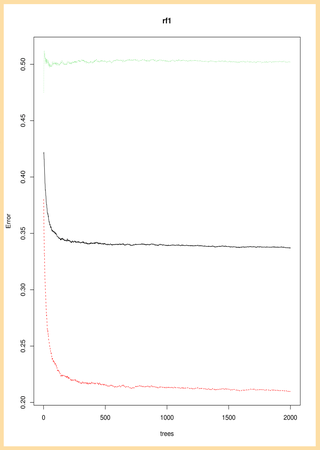
Naïve Bayes

The overall accuracy with the Naïve Bayes model came out to be around 60%. This was a very low value compared to our various other models. We tried to implement an L-place smoothing metric on our Naive Bayes model, however, it did not result in a significant increase in accuracy. Due to this and the limited time, we decided not to go further with Naive Bayes and not further tune the model. The highest obtained accuracy with Naïve Bayes was **60.14%**.

```
> table(predictions_mlr[,1],train$relevance)

        0     1
  0 43117 30030
  1  1942  4957
>
```

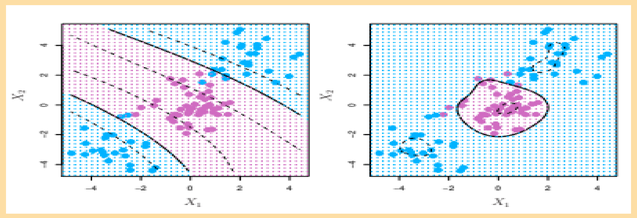
Random Forests

The main results from running Random Forests are shown in the figure below. The error, using a validation set for predictions, came to be around 34%, equating to an accuracy of 66%. To further analyze the causes of the error and in hopes to increase accuracy, I tuned the random forest by modifying the number of trees in the model and by removing irrelevant predictors (found from logistic regression). This tuning capped off the accuracy for Random Forests at **66.4%**.

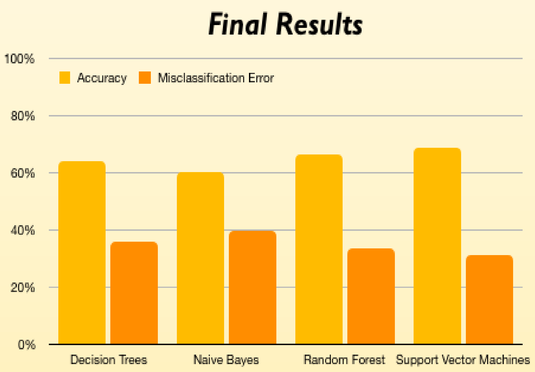


Support Vector Machines

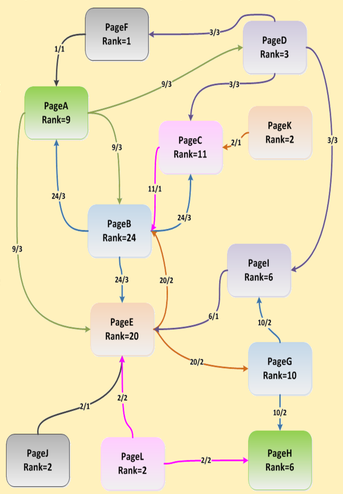
SVM turned out to have the highest accuracy, 67%, of the four models. Although it had an accuracy very close to that of Random Forest, I decided to move on with tuning our SVM to reduce the error. The two main ways I tuned our model were using different types of kernels (linear, radial, polynomial) and changing the cost/budget for each model. With the implemented tuning, the SVM model yielded an accuracy of **68.8%**.



Results and Conclusions



The above graph illustrates that **Support Vector Machines** had the highest accuracy (**68.8%**) and the lowest error (**31.2%**). This accuracy projects that the constructed model will predict the relevancy of a page correctly 68.8% of the time. My hypothesis, stated earlier in the *Introduction and Objectives* section, was proven to be correct through this experiment, as SVM did indeed have the highest accuracy. The diverse combinations of tuning methods, including kernel type and budgeting, for SVMs allow users to optimize their results, thus decreasing error and maximizing accuracy. This model, however, is unrealistic and would likely have a higher error if used in the real world, due to the millions of predictors needed to predict relevancy. Google, Bing, and Yahoo have constructed similar page rank algorithms that follow the graphic shown here. I look forward to learning more and continuing my research behind the beauty of search engines, over time implementing more complex formulas and intricate details.



Acknowledgements

- Special thanks to Bill Snow, Dr. Guru Parulkar, and all members of the Open Networking Foundation.