CS 434 - Spring 2016 - Final Project Report

Garrett Smith, Cody Malick

smithgar@oregonstate.edu — malickc@oregonstate.edu

Abstract

In this report, we examine four different algorithms for machine learning: linear regression, logistic regression, support vector machine, and decision tree. We apply these four algorithms to the same problem, spam classification, and examine how they perform, how they perform in different conditions, and how they compare to each other.

CONTENTS

I. Introduction

Classification is a common problem in machine learning. There are many approaches, some simple, some more complex, that all involve sorting a given set of items into two or more classifications. The goal, of course, being able to quickly and efficiently classify large sets of items into their correct classes. In our final project, we have examined four seperate classification algorithms, and applied them all to the same problem of classifying spam.

II. The Data

Our data set is a pre-processed set of spam from University of California Irvine, the `Spambase Data Set`.[1] The data set is comprised of the following:

Number of Instances: 4601

Number of Attributes: 57

- 48 continous real [0,100] attributes of type word_freq_WORD percentage of words in the e-mail that match WORD, i.e. 100 * (number of times the WORD appears in the e-mail) / total number of words in e-mail. A "word" in this case is any string of alphanumeric characters bounded by non-alphanumeric characters or end-of-string.
- 6 continuous real [0,100] attributes of type char_freq_CHAR] = percentage of characters in the e-mail that match CHAR, i.e. 100 * (number of CHAR occurences) / total characters in e-mail
- 1 continuous real [1,...] attribute of type capital_run_length_average = average length of uninterrupted sequences of capital letters
- 1 continuous integer [1,...] attribute of type capital_run_length_longest = length of longest uninterrupted sequence of capital letters
- 1 continuous integer [1,...] attribute of type capital_run_length_total = sum of length of uninterrupted sequences of capital letters = total number of capital letters in the e-mail
- 1 nominal 0,1 class attribute of type spam = denotes whether the e-mail was considered spam (1) or not (0), i.e. unsolicited commercial e-mail.

Class Distribution

- Spam: 1813 (39.4%)
- Non-Spam: 2788 (60.6%)

This data was great for a comparison set because we could look at the exact distribution of data, what kind of data it was, what the different attributes meant, etc. Because we did not have a separate training set, we built our training set out of about 750 entries from the original training set of 4000. The training set contained both spam and non-spam classified entries.

Now that we have the layout of the data, next we will examine our first algorithm, linear regression.

III. Linear Regression

Linear regression is a very simple algorithm, designed to handle data in very simple dimensions. We ran into a lot of problems trying to implement this algorithm over a data set with over fifty features. After some trial and error, we decided that applying this algorithm to a feature set this complex didn't make a lot of sense. We were hitting a few percentage

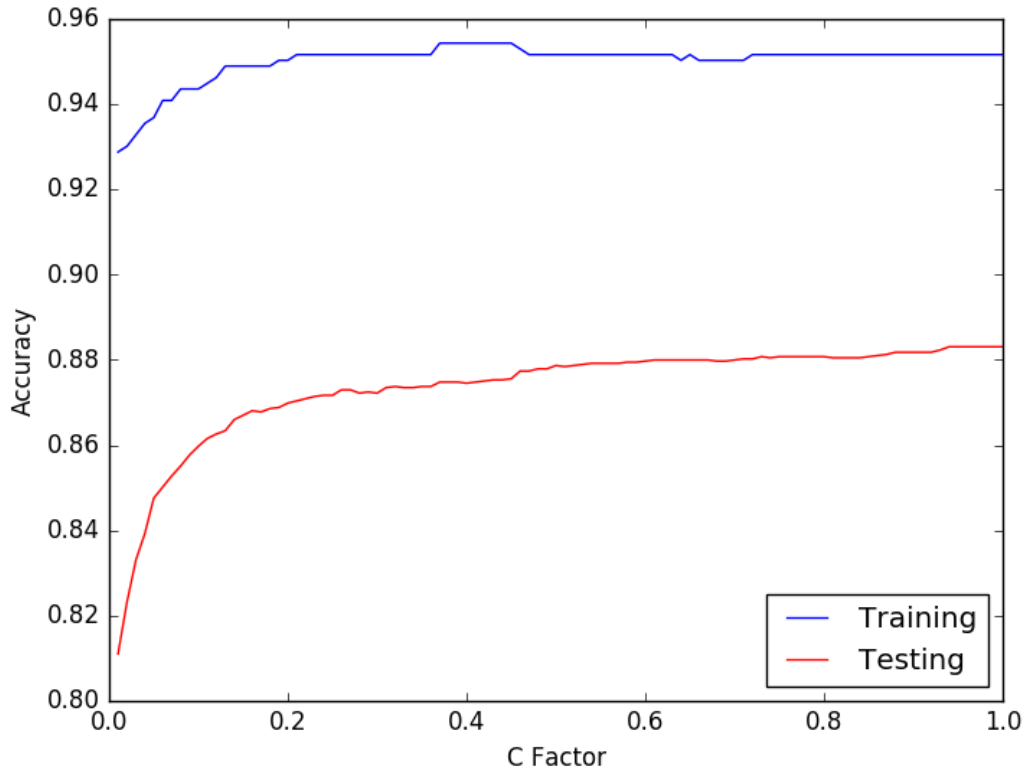
points in accuracy, and we believe that it was purely luck that we got that many. Instead of thoroughly investigating linear regression, we added support vector machine to our lineup of algorithms to examine. Before jumping into a support vector machine, we will now examine the next closest thing to linear regression, logistic regression.

IV. Logistic Regression

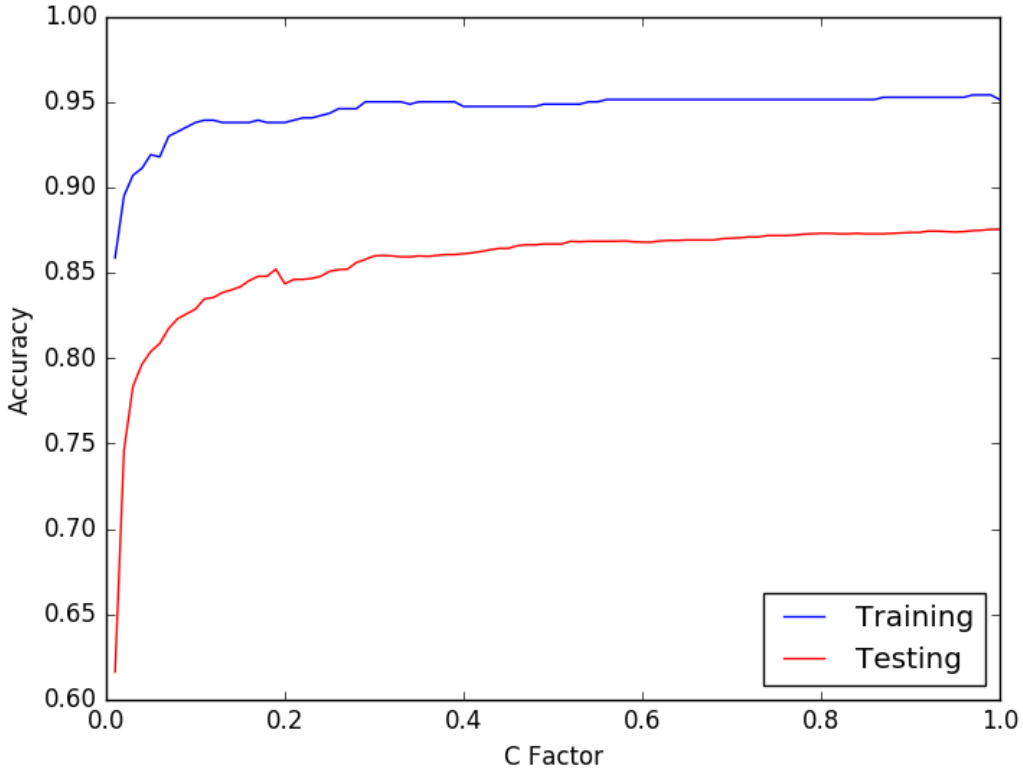
Logistic regression is a form of classification that uses probability. We decided to use logistic regression because it is a form of linear regression that is able to handle large amounts of features with relative accuracy, and often yields good results in a binary classification problem.

A. Implementation

We implemented logistic regression by using the `scikit-learn` library in python. `scikit-learn` is a great library that has several implementations available and an extremely simple setup process. We used the `linear_model.LogisticRegressio` module. As far as execution, we first set it up with the appropriate features, and got surprisingly good results right out of the box. With default settings (no regularization, default penalty of l2), we achieved 95% accuracy on the training set, and 88% accuracy on the test set. We experimented with different regularization values from .1, high regularization, to 1.0, no regularization.



Surprisingly, we only saw an increase in accuracy as regularization decreased. From this behavior we can extrapolate that our data has few to no outliers skewing the data in an unhelpful way. Next we tried applying l1 penalty instead of l2:



With l1 penalty we see it takes quite a bit longer for the accuracy to peak, and overall there is a lower overall accuracy of about 3%. This is expected as l1 penalty is designed for data sets with sparse data, while l2 is better for larger, more complete data sets.

V. Support Vector Machine

VI. Decision Tree

VII. Conclusion

VIII. Appendix A - Linux Structs

for future code samples

IX. Bibliography

References

[1] E. R. Mark Hopkins. (1999, jul) Spambase data set. [Online]. Available: https://archive.ics.uci.edu/ml/datasets/Spambase