CMPS142 Final Project Report

Isaiah Solomon

March 2017

Problem

For my machine learning project, I am attempting to preemptively grade people's essays before they are actually graded. My program will display where the writer is at the current moment and give them an appropriate estimate so that they can tailor it again to get a better possible score. Each essay (in the set I used) is out of 12 and are each roughly about 300 - 400 words long each. I am using 1500 essay data from Kaggle.com in an online machine learning competition they were hosting. I will be using 1000 essays for training and 500 for testing

Approach

My official approach is by using an epsilon support vector regression, using sklearn for the model fitting and Matlab for plotting the data (For some reason my machine couldn't get matlibplot to work).

Rationale

I went with a classification problem with each possible integer from 0 to 12 as its own class. I did this because although a linear regression problem would work as well, I felt that through classification, there were more options available to me in terms of choosing a classifier.

Hypothesis

Some of the questions that I thought about were what aspects make an essay good. Although I cannot make an algorithm based on the meaning of each sentence and judge the clarity of the writer's understanding (at least within the timeline of the project), I thought I can at least judge the essay based on its complexity in terms of vocabulary and sentence structure.

Experimental Design

I also took the time to specifically choose my features as it is hard to make a feature based on the actual message of the essay and each specific sentence without making the program more complex within the timeline of the project. One of the features I chose was average word length. My reasoning is that although word length doesn't necessarily mean better word, but more often than not, more complex words of higher vocabulary are generally longer. If the writer included more of these in their essay, it concludes to usually a more well-rounded essay. Another feature I used was sentence length. Sure there are shorter sentences that achieve a stronger message than their longer counterparts, but I believe if each sentence is wordy enough to get a message across, there is more information being shown to the reader. Another feature is length of essay. Typically if there is a word-count minimum, most rushed essays qualify for the bare minimum. Although like mentioned earlier, a longer essay does not equate to a more well-written essay, a longer essay does show the time spent on the essay to refine it and make it better. I also saw observing where I got the data sets from that the essays were cluttered with words that start with '@'. Kaggle.com provided essays that were replaced with 'variables' that started with '@' that took place of specific proper nouns or key words. My last feature is how often the writer included these types of words as an essay with more proper nouns than pronouns often show that the essay is consistent grammar-wise. With these given features, I try to grade the essay more on grammar than content, which isn't a bad thing either.

When it came to choosing the classifier, I was doing a lot of research; however, was not sure exactly what method was best for my problem. That is why I tried many different classifiers in order to decide on one.

Framework

The primary programming language I used was Python using the sklearn library, as well as Matlab in order to plot data. At first, I was learning Docker in order to not have to install the dependencies on my local machine. This worked out for the most part; however, after a while I felt that using Docker was overkill and part of the way through I ended up tossing it. I initially was also planning to use Tensorflow due to having experience with it on assignments; however, I was still intimidated by it due to the amount of low level programming needed, so I chose sklearn because of its ease of use with most classifiers provided.

Previous Work

I found out about the problem through an essay I found online [1] where a couple individuals were trying to accomplish the same task with the same data set given. I have looked at their classifiers and features chosen and did not want to do the exact same thing so I tested different classifiers and chose slightly

different features in order to see if I can produce a better result doing it my own way.

Challenges

One of the main challenges that I faced is my lack of knowledge in the subject of Machine Learning. I think I have been lacking in the skill to fully grasp the concept of Machine Learning, which hindered my ability to quickly assess the best decisions to make when tackling this project. I ended up spending a lot more time testing things out and looking back on slides. I think if I had more time and more help from peers versus tackling this project alone, I might have had better results.

I also think lack of documentation, examples, and resources on both sklearn and Tensorflow hindered my progress. Not saying that both libraries are bad in any way, both libraries are wonderful to work with. However, do to a lack of community when I need a certain function or method of doing things hard to find, I think a more available resource for those specific questions would have reduced my time in looking for answers.

Lastly, just using sklearn and matlibplot have been difficult on my personal machine. Sklearn is actually a bit too high-level for my taste, I think something like a mixture of Tensorflow and sklearn would have been more useful to me. Matlibplot ended up not working at all and no online resources were available to help me.

Most of how I progressed through my challenges were to just look up more answers, review examples of programs, and look back at my previous assignments as a review. To fix the errors in Matlibplot, I used Matlab for graphing. Most challenges weren't impossible as I was able to find a fix for most, but definitely would have been more efficient without them.

My Implementation

I had set up the parser for the tsv files that held the data set. I also set up the environment for use of sklearn and numpy, as well as writing most of the code for the algorithms themselves. I tested Linear Regression (non-classification), Logistic Regression, SVR, and KNN.

Outcome

As I was a solo member, I have worked on all of the working parts of the implementation. Some limitations are that it is really specific to my program, where if it were to be reusable, I'd have to edit the code a lot.

Knowledge Gained

I think my knowledge gained was just the basic implementation of each of the algorithms I used. I was at first attempting to use linear regression because the targets, although can be seen as classes, are integers as well and can be treated as such. Just by at first trying to get that to work and researching a lot on linear regression, I learned more about the importance of features on training the data and about how useful regularization is to create sparsity.

In terms of logistic regression, I didn't really learn too much from what I already knew about it. I learned a lot about KNN, and how it is easy to implement, but doesn't always produce the best results, which I found through using it on my data.

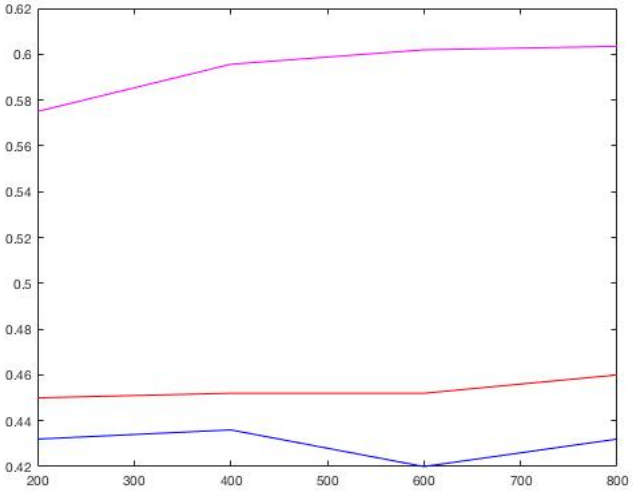
I also learned a lot about SVM and SVR, which still are the classifiers that I have the most trouble with since like I said earlier I have trouble picking up on concepts.

Results

Using my test file test.py, I evaluated four different algorithms to train the essay data which are KNN, SVR, Linear Regression, and Logistic Regression, with success rates shown below:

```
KNN: 0.46
Linear Regression: 0.60333589894
Logistic Regression: 0.432
SVM: 0.603434310495
```

I also graphed the line for error rate spanning through the training set in intervals of 200:



where SVR is magenta, linear regression is green, logistic regression is red, and KNN is blue.

Critical Evaluation

From the results, I have achieved best results using basic linear regression as well as with SVR. I was not, however, able to get any great results, due to my lack of understanding of adjusting the SVR parameters for sklearn, but I was able to find that SVR did perform the best. I also think that although sklearn had the easiest implementation for the problem I am trying to solve, I think I should have stuck with Tensorflow for the sake of learning what I needed to learn at a lower level which I believe would probably have given me a better understanding of what I needed to accomplish. I think my results could have been better in that maybe the classifiers I was using weren't the most optimal for my problem. Another implementation found that using word vectors for classifying produced better results.

I thought that my choice of features also had a big part in which I think some of my features had little correlation with how well the essay really did. I think going for more what a grader would actually look for would have yielded better results, like tone, meaning, etc.

Lesson Learned

I definitely learned more about general machine learning that I previously didn't have a great understanding of. I think knowing my libraries more as well as just

general algorithms and which is most optimal for which problem is what I can learn more in depth next.

Future Work

The next best steps to take would be to more carefully choose features that would greatly affect the fitting more. Also to know more about classifiers before using them.

References

[1] Jason Zhao Shihui Song. http://cs229.stanford.edu/proj2013/songzhao-automatedessayscoring.pdf.