Assignment 3 Project Report

CMPT310
Trevor Grabham
Spring 2021

Neural Net Learning

In the process of learning about learning in neural nets, I went about using scikitlearn in python. I used the MLPClassifier class that would set up a neural net with a single output node. Before I was able to pass that news articles into the neural net though, I had to get the data into a more computer readable form.

I parsed out the title and the body of the articles, and used some of the natural language processing ideas, specifically a count vectorizer, and a stemmer from the nltk module in python. Using the stemmer, I was able to take the words that were present in the articles and reduce them down to their root words, so that we did not have multiple representations of words that are very similar, such as friend and friends. I then was able to get out some of the extraneous words by comparing the words in the article against known english stop words that were loaded in from the nltk module. After I got this condensed set of words, I applied the count vectorizer on the words to create a bag of words.

Some of the attributes that I was able to play around with in regards to the count vectorizer was the number of features to take, which I found 5000 to be optimal, and the maximum length of the word chains, which I found to be optimal at 3. If we had any less than 5000 features than I found that the neural net would default to .5, meaning that it was unable to learn any insights from the data. If we went into any more then 5000 features than I found that we would get about the same accuracy, my guess being that the cause of that has something to do with the fact that we are giving the neural net too many variables to optimize for and it was unable to find a reasonable solution in a reasonable amount of time. I also found that reducing the size of the maximum length of the word chains to anything less than 3 also leads to a major drop off in accuracy, again regressing to about .5.

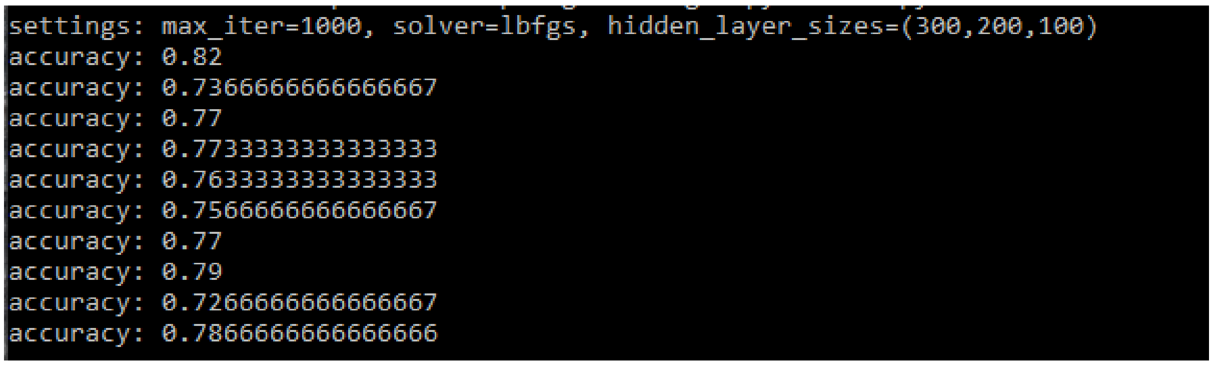
The last change that I tried to make with regards to the aggregation of the data, was breaking the data up into a total set, just the titles of the articles, and just the body of the articles. I actually found not too significant of an accuracy drop between any of the data sets, with the total data set being slightly more accurate, but running into a lot more memory issues because of the size of the data set. Because of the memory issues, I decided to use the title data set.

Now on to the neural net attributes. I started off running the neural net with attributes that are similar to the ones covered in class. I started off using stochastic gradient descent and set the maximum number of iterations to 500, leaving the rest of the settings as default. This worked out decently well, but was still shy of the 0.85 mark that we were aiming for. I then tried to add an extra hidden layer to the neural net, to see if we would be able to make more meaningful insights with an extra hidden layer. This didn't seem to have much of an effect. To properly rule out the added hidden layers, I then added a third hidden layer, having a slightly better result, but not a statistically better result. I then tried to change the solver function from stochastic gradient descent to lbfgs, a family of quasi-newton methods, and reverted all of the changes back to default. This seems to be a better option for a solver.

The final iteration of changes that I made to the settings of the neural net was to increase the maximum number of iterations to 1000, and to add more and larger hidden layers to the neural net. This gave us our best accuracy score yet, but it still left us shy of the 0.85 mark that we were aiming for.

Conclusion

My final takeaways from this assignment were how important getting the data into the right format was for the neural net. Exploring the different options as to how we were formatting the data with the count vectorizer showed how having the data in a non optimal form quickly led to accuracy that would have been no better off than a guess. I also found that the solver was the most important setting to change for the neural net to get a more accurate result. The accuracy increase that came with increasing the complexity of the hidden layers was modest, but the time and memory costs of this was quite expensive. I was unable to run any larger neural nets without getting memory errors, and I found that this was a huge roadblock in terms of getting to our goal of 0.85 accuracy.



```
settings: max_iter=1000, solver=lbfgs, hidden_layer_sizes=(300,200,100)
accuracy: 0.82
accuracy: 0.7366666666666667
accuracy: 0.77
accuracy: 0.7733333333333333
accuracy: 0.7633333333333333
accuracy: 0.7566666666666667
accuracy: 0.77
accuracy: 0.79
accuracy: 0.7266666666666667
accuracy: 0.7866666666666666
```