Josh Engelsma

Adam Terwilliger

April 5, 2016

CIS 678 – Machine Learning

Project 4

**Abstract**

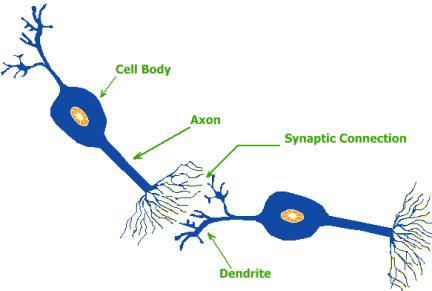
We dive into more advanced supervised learning in CIS 678 – Machine Learning, by implementing a neural network from scratch. Using a sigmoid function, we have implemented forward propagation, and backwards propagation using gradient descent to train a neural network on three distinct datasets, each foundational in machine learning. We looked into classification of Iris (4 features, 3 targets), Cancer (30 features, 1 target), and Wine (10 features, 3 targets). Our NN implementation differentiates itself in three main ways: vectorization, bi-variate parameter design of experiments, and animation of training. We devolved into these distinct sub-areas, while still maintain 95-99% classification rates on both in-class datasets and the three ML datasets.

**Implementation**

Our program is written in Python 2.7, R 3.1.2, and bash scripting in Unix. These programs were executed locally on each member’s respective Macbook Pro (2012), testing on eos23 and okami.

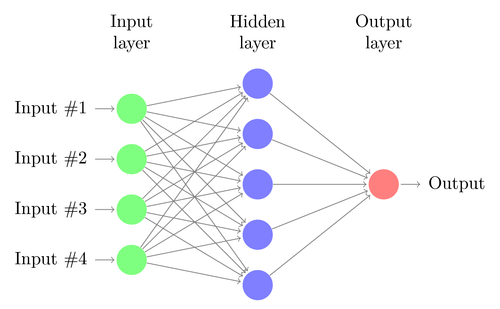
**Datasets**

Two in-class datasets involved fishing, but testing was too variable with too few observations and game data had four features and looked to predict the action of the game character. Additionally, we used the foundational Iris dataset containing four features, 150 observations, and 3 target flowers. Using the UC-Irvine repository, we uncovered two datasets: Wine (classify 3 types of wine using 10 features and 178 obs.) and Cancer (classify benign or malignant using 30 features and 569 obs.). Each dataset had character features that were discretized and continuous features that were normalized between 0 and 1.

**Background**

*Where do neural networks come from?* They are inspired by the neuron from the human brain, as seen in Figure 2.

*How do neural networks work?*

****A neural network, example in Figure 3, maps its inputs to its outputs via multiple non-linear transformations propagated through multiple hidden layers.[1] This process is accomplished by iteratively calculating a weighted sum of the edges to and passed through an activation function. The neural network is able to “learn” through gradient descent and back-propagation of error by updating the edge weights after each iteration.

**Figure 1.** Neuron from Human Brain

**Figure 2.** Simple Neural Network from Human Brain

**Results**

The best relative classification rate vs. training/test split as seen in Figure 3 that our neural network performed was 97.3% for Iris using a 75/25 training/test split, 98.0% for Cancer using a 40/60 training/test split, and 98.9% for Wine using a 50/50 training/test split. We have demonstrated the power of our neural network with high classification rates on reasonable training/test splits. We can infer from the number of observations for Cancer relative to Iris where we only need 40% of the dataset to train, while similarly, the 50/50 with Wine shows the value of adding additional features. Additionally, in Figures 4 and 5, we observe a correlation between classification rate and number of epochs/learning rate.

**gure 1. Sample validation output using original holdout split**

Using the k-fold validation approach, we found 12-fold (92/8) with 86% classification rate offered the most promising results. We find Figure 3, maximizes over the k iterations with the maximum over 2 through 15-fold validation landing around 80%.

The randomized validation approach proved to be the most effective, as we observe in Figure 3. We found nearly 89% classification rate using a 78/22 training/test split for the forum-stemmed data. We explored a sentiment analysis case-study training our Naïve Bayes classifier using over 1.5 million tweets pre-labeled with positive or negative sentiment. Using the randomized validation approach for this twitter data, we observed a classification rate of nearly 76% using an 87/13 training/test split.

**Discussion**

In training our Naïve Bayes classifier using over 1.5 million tweets, we looked to use this classifier to make predictions about tweets outside the sample set. As such, we used the Twitter API to pull the last 3000 tweets (if they have that many) for around 30 users. The types of users were grouped into political, organization, media, figure, and comedy, as seen in Figure 4 and Table 1. Due to the unorthodox syntax of most tweets, we utilized libraries that addressed issues like retweets, emoticons, handles, hashtags, and links. Instead of just removing these types of language, we processed them in a way to better train our classifier. For instance, a smiley emoticon ☺, vs. a sad emoticon, ☹ can convey a positive or a negative sentiment.

**Table 1. Positivity results for Twitter validation set**

As we note from Table 1, all of the current presidential candidates as of February 16, 2016 are included, as well as, scientific organizations and figures. A handful of media, celebrities, and comedy were pulled to round out the validation set. Some correlations we find from Figure 4 and Table 1 include more conservative politicians are more positive, scientific organizations are leading positivity on Twitter, secular media/scientists are less positive, and comedy/parody accounts are often more negative. We can compare the differences between positive and negative tweets in Figures 5 and 6.

**Figure 5. Example of Positive Tweet.**

**Figure 6. Example of Negative Tweet.**

**Future Work**

We have four main directions we would pursue if time allowed: topic clustering, precision/recall, n-grams, and maximum entropy. We can gain some preliminary intuition from the word clouds in Figures 7 and 8 that topics like “atheism” and “religion” may be quite similar as we note words like “god”, “people”, “belief” and “faith” appear frequently in both classes of documents.

**Figure 7. Atheism Word Cloud**

**Figure 8. Religion Word Cloud**

As a complementary analysis to traditional training vs. testing validation, precision/recall offers additional insights into the types of error that the classifier is making, as seen in Figure 9.

**Figure 9. Precision vs. Recall**

As mentioned in the summary of the problem, Naïve Bayes makes the underlying assumption that features/observations are independent. Maximum Entropy classification and n-grams look to an alternative, as we may find instances where independence may not be inferred (i.e. “President”, “Obama” / “President”, “Bush” vs. “President Obama” / “President Bush”). We can understand more about maximum entropy and n-grams in Figures 10 and 11.

**Figure 10. Further exploration of Maximum Entropy Classification**

