SSTEM Predictive Data Science Project

Ashley Joyner

April 2016

1 Introduction

I used a submission originally created by Kirill Kliavin called 'TensorFlow deep NN' [1]. This submission took a provided photo and corresponding number and used it to train a model to predict a number based off of a photo. The language I used was python because that was what the original was written in.

2 Data Manipulation

First, the programmer read in the file containing the training data for this project. The file has 4200 rows and 785 columns. 784 of the columns contained a stretched array of values describing the image. The first column contained the value of the number depicted in the image. This value was only allowed to be between 0 and 9.

In order to print an image the columns that contained the image were rearranged into a two-dimensional array of width and height equal to 28. Next the programmer created a procedure that took an array of the original 784 columns, reshaped them, and displayed them as a photo of a number. We also printed the corresponding values of the number in the image. By looking at this image and comparing it to the value we saw that they were equal.

Because this was a classification problem the best way to work with the values was to use one-hot- vectors. A one-hot vector is a "is a vector that contains a single element equal to 1 and the rest of the elements equal to 0. In this case, the nth digit is represented as a zero vector with 1 in the nth position." [1] I would like to provide an example for clarification, this matrix:

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

is equal to 4. The programmer created a procedure that took a value and turned it into a one-hot vector. Before moving on the data set was split into a part that was used to train the model and a part that was used to test the model for accuracy.

3 Neural Network

The next thing the programmer did was to create a neural network model. He created several procedures. One procedure created weights with slight noise to break symmetry. Another procedure created a slightly positive bias. The next procedure was one that created a convolution layer, which used learn able kernels/filters that correspond to a shape of each digit. The last procedure was a pooling procedure. This procedure took small blocks of the image array and keeps the maximum value for each of the blocks.

After writing the procedures, the programmer created a deep neural network by layering the procedures on top of each other and using the output of one as the input for another. The first layer consisted of a convolution and a max pooling. The product was a reduction of a 28x28 array to a 14x14 array. After the second layer (which was similar in logical format) the array was reduced again to 7x7.

Next, the programmer applied a simple regression to the neural network. Then he used an optimization algorithm on the network. Lastly, the highest probability for each one-hot vector is calculated.

4 Training

To begin training the model the programmer wrote a procedure that split the data into random batches because it is faster and cheaper than using all of the data for every step. After that the programmer began the TensorFlow session. The programmer used TensorFlow because it lets users build lots of interacting operations and then runs them all at once outside of python.

In the next step the programmer took the data designated as the training set and got batches and used it on the graph to replace the placeholders. After the training was finished the program checked the accuracy on the part of the data set that wasn't used during training.

Lastly, once the programmer was satisfied with the predictive accuracy of the second part of the data, he read in a new data set. This data set contained only images with no associated values. The program converted the image array into a 2-dimensional array, just like in the beginning. And then he predicted the values of the numbers in the image based off of the results of the model training earlier

5 Conclusion

I did not make any improvements to the original code. It is highly unlikely that I would have succeeded if I had, because the accuracy of this program was 0.99. It couldn't really get any better.

I am very interested in predictive data science and this project has given me a foot hold into the subject. I learned a tiny bit about how to do predictive modeling, but I can't wait to try my own hand at it. I am considering looking

into the Facial Keypoints Detection competition and reading the code that people have submitted for that one.

References

[1] Kirill Kliavin. Tensorflow deep nn, 2016.