Morgan Rogers

12-1-21

Program 3 Report - Neural Networks / Machine Learning

Developing the Network

My network was developed using Keras. I used the Sequential model where each layer in the model has exactly one input tensor and one output tensor. My main resources were notes taken in the python deep learning class I am currently enrolled in at UMKC. I've created a few neural networks using Tensorflow and Keras so this project wasn't too difficult having that experience. My first task in developing the network was to read the data from the StudentsPerformance.csv file. I read the data using pandas and then created separate data frames for the input values and the 'target' values which I wanted the network to predict (math, reading, and writing scores). I used the 'get_dummies' method in pandas to one-hot-code the inputs and then I finally split the data up into training and testing sets using the 'train_test_split' method from scikit-learn. The testing and validation sets make up 30% of the data together and the training sets make up 70% of the data. When training the model, I used the 'model.fit' method and I have the validation data set to X_valid and y_valid which is used to tune the model with the result of the metrics (accuracy and lost) received from the validation data. I then test the data using the 'model.evaluate' method which uses the testing data to evaluate the accuracy of the model.

Final Network Configuration

My model has an input layer that takes in the shape of the input dataset (x), one hidden layer with 30 units and uses the rectified linear unit (ReLU) activation function, and an output layer with 3 output units that uses the softmax activation function. The 3 output units are for the predicted math, reading, and writing score. The ReLU activation is a piecewise linear function that will output the input directly if it is positive otherwise it will output zero. The softmax activation function converts a vector of numbers into a vector of probabilities. I found that leaving out the softmax activation function on the output layer only gave the model a 30% accuracy but adding the softmax activation gave the model an accuracy of around 60%. Adding more hidden layers did not improve the accuracy and neither did adding more input units to the hidden layer. I think using a larger dataset could have improved the accuracy of the model.

Validation Strategy

The validation strategy I used was to split the data up into the training, validation, and testing datasets. The training set contained 70% of the original data and the validation and testing sets contained 15% of the original data each. I trained the model with the training set and used the validation set to test the model against during training. I then used the testing set for the final evaluation of the model. I used the early stopping callback from Keras which monitored the loss during training and stopped training after 3 epochs of no improvement. I passed in 50 epochs for fitting the model and training only made it to 22 epochs before stopping.

Results

The network ended up evaluating to an accuracy of 62.67% and a loss of 4813.07.

Further Exploration and Follow Up

I believe the model could have a better accuracy given more data. The given dataset was small, and I was questioning whether the input data features were related at all to predicting test scores. For example, gender and race. Further exploration could involve switching the input and target data so that given the test scores, could the network predict the parental level of education or if the test preparation course was completed? I think this could be an interesting exploration. This project gave me good practice on using Keras and I have a better understanding of how to apply the concepts we learned in class.

Resources

Softmax Activation Function with Python

A Gentle Introduction to the Rectified Linear Unit

Pandas Get Dummies

Keras EarlyStopping

Keras Model Training APIs