Assignment 4: Convolutional Neural Networks

CS489/698 – Spring 2019

Out: July 2, 2019
Due: July 12 (11:59pm), 2019

Submit an electronic copy of your assignment via LEARN. Late submissions incur a 2% penalty for every rounded up hour past the deadline. For example, an assignment submitted 5 hours and 15 min late will receive a penalty of ceiling(5.25) * 2% = 12%.

Be sure to include your name and student number with your assignment.

1. **[50 pts]** For this question, you will experiment with fully connected neural networks and convolutional neural networks, using the Keras open source package. Keras is one of the simplest deep learning package that serves as a wrapper on top of TensorFlow, CNTK and Theano. Preliminary steps:

   - Familiarize yourself with Keras: `https://keras.io/`. Click on "Getting Started" and read the "Guide to the Sequential Model".
   - Download and install Keras on a machine with a GPU or use Google's Colaboratory environment (`https://colab.research.google.com`), which allows you to run Keras code on a GPU in the cloud. Colab already has Keras pre-installed. You simply need to create a Python notebook. Click on "edit", then "notebook settings" and select "GPU" for hardware acceleration. It is also possible to select "TPU", but the Keras code provided with this assignment will need to be modified in a non-trivial way to take advantage of TPU acceleration.
   - Download the file `cifar10_cnn_cs480.py` from the course website.

   Answer the following questions by modifying the code in `cifar10_cnn_cs480.py`.

   (a) Compare the accuracy of the convolutional neural network in the file `cifar10_cnn_cs480.py` on the cifar10 dataset to the accuracy of simple dense neural networks with 0, 1, 2, 3 and 4 hidden layers of 512 rectified linear units each. Run the code in the file `cifar10_cnn_cs480.py` without changing the parameters to train a convolutional neural networks. Then, modify the code in `cifar10_cnn_cs480.py` to obtain simple dense neural networks with 0, 1, 2, 3 and 4 hidden layers of 512 rectified linear units (with a dropout rate of 0.5). Produce two graphs that contain 6 curves (one for the convolutional neural net and one for each dense neural net of 0-4 hidden layers). The y-axis is the accuracy and the x-axis is the number of epochs (# of passes through the training set). Since neural networks take a while to train, cross-validation is not practical. Instead, produce one graph where all the curves correspond to the training accuracy and a second graph where all the curves correspond to the test accuracy. Train the neural networks for 20 epochs. Although 20 epochs is not sufficient to reach convergence, it is sufficient to see the trend. Submit your code and explain the results (i.e., why some models perform better or worse than other models).

   (b) Compare the accuracy achieved by rectified linear units and sigmoid units in the convolutional neural network in `cifar10_cnn_cs480.py`. Modify the code in `cifar10_cnn_cs480.py` to use sigmoid units. Produce two graphs (one for training accurqcy and one for test accuracy) that each contain 2 curves (one for rectified linear units and another one for sigmoid units). The y-axis is the accuracy and the x-axis is the number of epochs. Train the neural networks for 20 epochs. Although 20 epochs is not sufficient to reach convergence, it is sufficient to see the trend. Explain the results (i.e., why one model performs better or worse than the other model). No need to submit your code since the modifications are simple.

(c) Compare the accuracy achieved with and without drop out as well as with and without data augmentation in the convolutional neural network in `cifar10_cnn_cs480.py`. Modify the code in `cifar10_cnn_cs480.py` to turn on and off dropout as well as data augmentation. Produce two graphs (one for training accuracy and the other one for test accuracy) that each contain 4 curves (with and without dropout as well as with and without data augmentation). The y-axis is the accuracy and the x-axis is the number of epochs. Produce curves for as many epochs as you can up to 100 epochs. Explain the results (i.e., why did some models perform better or worse than other models and are the results consistent with the theory). No marks will be deducted for doing less than 100 epochs, however make sure to explain what you expect to see in the curves as the number of epochs reaches 100. No need to submit your code since the modifications are simple.

(d) Compare the accuracy achieved when training the convolutional neural network in `cifar10_cnn_cs480.py` with three different optimizers: RMSprop, Adagrad and Adam. Modify the code in `cifar10_cnn_cs480.py` to use the Adagrad and Adam optimizers (with default parameters). Produce two graphs (one for training accuracy and the other one for test accuracy) that each contain 3 curves (for RMSprop, Adagrad and Adam). The y-axis is the accuracy and the x-axis is the number of epochs. Produce curves for as many epochs as you can up to 100 epochs. Explain the results (i.e., why did some optimizers perform better or worse than other optimizers). No marks will be deducted for doing less than 100 epochs. No need to submit your code since the modifications are simple.

(e) Compare the accuracy of the convolutional neural network in the file `cifar10_cnn_cs480.py` with a modified version that replaces each stack of (CONV2D, Activation, CONV2D, Activation) layers with 3x3 filters by a smaller stack of (CONV2D, Activation) layers with 5x5 filters. Produce two graphs (one for training accuracy and the other one for test accuracy) that each contain 2 curves (for 3x3 filters and 5x5 filters). The y-axis is the accuracy and the x-axis is the number of epochs. Produce curves for as many epochs as you can up to 100 epochs. Submit your code and explain the results (i.e., why did one architecture perform better or worse than the other architecture). No marks will be deducted for doing less than 100 epochs.

2. **[30 pts]** In object recognition, translating an image by a few pixels in some direction should not affect the category recognized. Suppose that we consider images with an object in the foreground on top of a uniform background. Suppose also that the objects of interest are always at least 10 pixels away from the borders of the image. Are the following neural networks invariant to translations of at most 10 pixels in some direction? Here the translation is applied only to the foreground object while keeping the background fixed. If your answer is yes, show that the neural network will necessarily produce the same output for two images where the foreground object is translated by at most 10 pixels. If your answer is no, provide a counter example by describing a situation where the output of the neural network is different for two images where the foreground object is translated by at most 10 pixels.

(a) **[15 pts]** Neural network with one hidden layer consisting of convolutions (5x5 patches with a stride of 1 in each direction) and a softmax output layer.

(b) **[15 pts]** Neural network with two hidden layers consisting of convolutions (5x5 patches with a stride of 1 in each direction) followed by max pooling (4x4 patches with a stride of 4 in each direction) and a softmax output layer.