COMPSCI 762 2022 S1 Week 8 Questions – Artificial Neural Networks

Luke Chang

May 7, 2022

Question 1

If you have a fully connected neural network with 4 features, 1 hidden layer with 4 nodes, and an output layer with 3 nodes. Recall a weight is associated with an edge between two nodes.

1. How many weights will you learn?

   $$(4+1) \times 4 + (4+1) \times 3 = 35$$

   Don't forget the bias nodes. If you are not sure about whether include bias or not, you should state you answer includes the bias in each layer (including the input layer and all hidden layers, but excluding the output layer).

2. What will be the form of the hypothesis returned by this neural network model?
   All weights are tunable parameters. The NN model learns the weights. The labels are given. There's no hypothesis for the output labels in a supervised learning task.

3. If you add a second hidden layer with 4 nodes, how many more weights will you learn?

   $$(4+1) \times 4 + (4+1) \times 4 + (4+1) \times 3 = 55$$

4. What is the size of the set of all possible hypotheses? (Let's consider the number of weights and what the possible value are for each weight.)
   35-dimensional space of real numbers. The hypothesis space is infinite.

5. What activation function will you use on each layer? What loss function will you use? Why do you choose them?
   There are many activation functions we can use for hidden layers, such as Sigmoid, tanh, ReLU, LeakyReLU, etc. For simplicity and training efficiency reason, ReLU and LeakyReLU are both valid choices. For the output layer, if probability is desired, we need Softmax function. For the loss function, Negative Log Likelihood (NLL) and Cross Entropy are both valid choices, depends on the activation function we have selected for the output layer. For example, the CrossEntropyLoss in PyTorch combines LogSoftmax and NLLLoss functions, so the output layer should not use the Softmax function.

Question 2

The LeNet-5 model from Yann LeCun's paper "Gradient-based learning applied to document recognition" (1998) `https://axon.cs.byu.edu/~martinez/classes/678/Papers/Convolution_nets.pdf` is one of the pioneer in convolutional neural networks for image classification.

Your task is to implement LeNet-5 using PyTorch or Tensorflow, and train it on the MNIST dataset `http://yann.lecun.com/exdb/mnist/`. (MNIST is a widely used benchmark dataset. Both packages provide built-in methods to download it, so you don't need to download it from the link in this file.)

Note that newer activation function such as ReLU wasn't exist at that time. Feel free to update the model, and use more efficient activation functions.

Both PyTorch and Tensorflow can implement LeNet-5 in few lines of code. There are many online resources on implementing LeNet-5. Feel free to lookup, if you get stuck.

Explain the neural network architecture and how you implemented the model.

The solution is in the coding demo.