Neural Networks Fall 2020

Gal Chechik, Shauli Ravfogel

1. *A skeleton for the code in this exercise is available here.*

2. (**20 pts**) *Back propagation*

   We use the same train and test data tests from the previous exercise, with the digits '5' and '8'. labels values are now 0 and 1 (for '5' and '8', respectively).

   Implement a back-propagation neural network learning algorithm

   a. The network should have the following architecture:
      i. 784 input neurons (the 28*28 pixels) in the input layer (X).
      ii. 150 RELU neurons in one hidden layer (Z). The output value of each of these neurons is $\mathrm{RELU}(\sum_{i=1}^{785} \alpha_{ji} x_i)$ where the index *i* runs over 785 α weights (one for each neuron, and an additional free weight for the bias).
      iii. One sigmoidal output neuron – the classifier (y). Its output value is $\sigma(\sum_{j=1}^{150} \beta_j z_j)$ where the index *j* runs over 150 β weights (one for each hidden neuron z_j).

   b. The error function for the sample *i* is $E_i = (y_i - f(x_i))^2$, and the total error is $E = \sum_{i}^{numexamples} E_i$.

   c. Coding instruction: The Python fucntion `train_model()`, provided on the notebook, contains a skeleton that takes care of initialization, computing the total error (using the function `evaluate()`) and the main loop. You have to implement the following functions that are called from that function:
      1. `CalcForwardZ` – compute the activity of the hidden neurons given an input sample.
      2. `CalcForwardY` – compute the activity of the output neuron given the activity of the hidden level.
      3. `UpdateWeights` – update the alpha and beta weights using the back-propagation algorithm.
      4. `Evaluate` – calculate the error function and the accuracy of the predictor.

3. (**20 pts**) *A basic back-prop experiment*

   Train your network to classify input images into the two classes: the digit '5' and the digit '8', using the back-propagation algorithm. Use a learning rate of eta = 0.001. Implement the back-propagation training algorithm for this network and run it for 50 cycles over the training set. Each cycle involves presenting all images to the network and modifying the weights accordingly.
   There are two error functions to calculate here – the first is the mean squared error, which is the error we are trying to minimize. The second is the number of classification mistakes. Since the output unit *y* is continuous, we classify the inputs by thresholding y: if $y > \frac{1}{2}$ we classify it as 1, if $y \leq \frac{1}{2}$ we classify it as 0. Calculate the mean training errors and the mean test (generalization) errors (both error types) after each cycle, and plot the errors as a function of the number of training cycles. Repeat the experiment with 5 different seeds. Discuss the results.

4. (**40 pts**) *The effect of hidden layer size (model complexity)*

The experiment above has three important hyperparameters: (1) the step size eta, (2) the size of the hidden layer, and (3) the number of gradient steps. Please read all the questions below before you start.

a. Run the same experiment as above, with 100 gradient steps, and with various values of eta (both larger and smaller than the one used above), changing on an exponential scale. Report the effects of this change and discuss.
b. Same as above, only now vary the number of hidden units. Try as little as 2, and as many as you dare. Report the effects of this change and discuss.
c. To avoid over-fitting it could be useful to stop the algorithm early, after a limited number of cycles. Choose the number of cycles that produce the lowest test error, and stop the algorithm at that stage. If the algorithm takes too long to converge you can just stop it arbitrarily and report the results, while indicating that your stopping criteria didn't take effect.