ECE/CS 559 Neural Networks, Fall 2018 - Homeworks #4 ~~and #5~~ Due: 10/23/2018, the end of class.

Erdem Koyuncu

This homework is worth two homeworks. Half of the total grade will be your Homework #4 grade. The other half will be your Homework #5 grade. All the notes in the beginning of Homework #1 apply. There are two computer experiments, both of which will use backpropagation. Therefore, I would suggest you to first have a working and well-tested backpropagation implementation. The problems will take some time so do not leave them to the last day! Make sure to include the computer codes in your report.

1. In this computer project, we will use a neural network for curve fitting.

   (a) Draw $n = 300$ real numbers uniformly at random on $[0, 1]$, call them $x_1, \ldots, x_n$.

   (b) Draw n real numbers uniformly at random on $[-\frac{1}{10}, \frac{1}{10}]$, call them $\nu_1, \ldots, \nu_n$.

   (c) Let $d_i = \sin(20x_i) + 3x_i + \nu_i$, $i = 1, \ldots, n$. Plot the points (x_i, d_i), $i = 1, \ldots, n$.
   We will consider a $1 \times N \times 1$ neural network with one input, $N = 24$ hidden neurons, and 1 output neuron. The network will thus have $3N + 1$ weights including biases. Let $\mathbf{w}$ denote the vector of all these $3N + 1$ weights. The output neuron will use the activation function $\phi(v) = v$; all other neurons will use the activation function $\phi(v) = \tanh v$. Given input x, we use the notation $f(x, \mathbf{w})$ to represent the network output.

   (d) Use the backpropagation algorithm with online learning to find the optimal weights/network that minimize the mean-squared error (MSE) $\frac{1}{n}\sum_{i=1}^{n}(d_i - f(x_i, \mathbf{w}))^2$. Use some η of your choice. Plot the number of epochs vs the MSE in the backpropagation algorithm.
   Hint: As discussed in class, for a given fixed η, the algorithm may not always result in a monotonically decreasing MSE (the descent may overshoot the locally optimal point). You may have to modify the gradient descent algorithm in such a way that you decrease η (e.g. via $\eta \leftarrow 0.9\eta$) whenever you detect that the MSE has increased. Also, beginning with a very large η may result in an immediate divergence of the weights.

   (e) Let us call the weights resulting from the backpropagation algorithm (when it converges) as $\mathbf{w}_0$. The curve $(x, f(x, \mathbf{w}_0))$, $x \in [0, 1]$ will then be a fit to the points (x_i, d_i), $i = 1, \ldots, n$. Plot the curve $f(x, \mathbf{w}_0)$ as x ranges from 0 to 1 on top of the plot of points in (c). The fit should be a "good" fit.

   (f) Your report should include a pseudocode of your training algorithm including all update equations written out explicitly (similar to what I had done in the second question of your second homework). The pseudocode should be written in such a way that anyone would be able to implement your algorithm without knowing anything about neural networks.

As a preparation for the next problem, we review the formulation of a general classification problem. Suppose that we have a set $\mathcal{C}$ of classes. For example, $\mathcal{C} = \{0, 1, \ldots, 9\}$ for the digit classification problem, or we could have $\mathcal{C} = \{\text{cat}, \text{dog}, \text{dragon}\}$ for a problem where we wish to determine whether a given picture contains a cat, dog, or a dragon. In the supervised learning setup, we have a training set $\mathbf{x}_1, \ldots, \mathbf{x}_n$ with the corresponding predetermined classes/labels $c_1, \ldots, c_n \in \mathcal{C}$. We wish to design a neural network that provides the hopefully-correct class of any given input $\mathbf{x}$.

Suppose that you consider a network with m output neurons to solve the classification problem described above. The first thing to do is to assign, for each class $i \in \mathcal{C}$, a representative output vector $\mathbf{d}_i \in \mathbb{R}^m$. Let $\mathbf{D} = \{\mathbf{d}_i : i \in \mathcal{C}\}$ denote the set of all representative output vectors of classes. For example, for the digit classification problem in Homework 2, we considered $m = 10$ output neurons with representative output vectors $\mathbf{d}_0 = [1\,0\,0\,\cdots\,0]^T$, $\mathbf{d}_1 = [0\,1\,0\cdots\,0]^T$, and so on. In particular, for the training sample $\mathbf{x}_i$, the class label is c_i, so that the desired output vector would be $\mathbf{d}_{c_i}$.

The next thing to do is to consider an energy function of the form $\frac{1}{n}\sum_{i=1}^{n} D(\mathbf{d}_{c_i}, f(\mathbf{x}_i, \mathbf{w}))$. Here,

- i is the training sample index,
- n is the number of training samples,
- $\mathbf{d}_{c_i} \in \mathbf{C}$ is the desired output for training sample i,
- $f(\mathbf{x}_i, \mathbf{w})$ is the network output given input (training sample) $\mathbf{x}_i$ and weights $\mathbf{w}$,
- $D(\cdot, \cdot)$ is some arbitrary distance function (metric). In particular, we use the function $D(\mathbf{d}_{c_i}, f(\mathbf{x}_i, \mathbf{w}))$ to measure how far off the network output $f(\mathbf{x}_i, \mathbf{w})$ is from the desired output $\mathbf{d}_{c_i}$. Typically, we choose $D(\mathbf{z}_0, \mathbf{z}_1) = \|\mathbf{z}_0 - \mathbf{z}_1\|^2$.

The next thing to do (which is the harder part) is to use the backpropagation algorithm to find some optimal weights, say $\mathbf{w}_0$, that minimize $\frac{1}{n}\sum_{i=1}^{n} D(\mathbf{d}_{c_i}, f(\mathbf{x}_i, \mathbf{w}))$. Having done this, the question is now how to determine the class of some arbitrary input pattern $\mathbf{x}$? Intuitively, we should choose the class whose representative output vector is closest to the output provided by $\mathbf{x}$ according to distance function $D(\cdot, \cdot)$. In other words, given some arbitrary input pattern $\mathbf{x}$ and weights $\mathbf{w}_0$ (or any weights in general), we first calculate the network output $f(\mathbf{x}, \mathbf{w}_0)$. We can then estimate the class of $\mathbf{x}$ as

$$\arg\min_{i \in \mathcal{C}} D(\mathbf{d}_i, f(\mathbf{x}, \mathbf{w}_0)).$$

In other words, the estimated class i for input pattern $\mathbf{x}$ should minimize $D(\mathbf{d}_i, f(\mathbf{x}, \mathbf{w}_0))$.

---

2. In this computer project, we will design a neural network for digit classification using the backpropagation algorithm (see the notes above). You should use the MNIST data set (see Homework 2 for details) that consists of 60000 training images and 10000 test images. **The training set should only be used for training, and the test set should only be used for testing.** Your final report should include the following:

   - The details of your architecture/design, including
     - Your network topology, i.e. how many layers, how many neurons in each layer, etc. (Obviously you will have 784 neurons in the input layer).
     - The way you represented digits $0, \ldots, 9$ in the output layer. For example, one may use the same setup as in Homework 2, 10 output neurons, with $[1\ 0\ \cdots 0]$ representing a 0, $[0\ 1\ 0 \cdots 0]$ representing a 1 and so on. Another person may have just one output neuron with an output of 0 representing a 0, an output of 1 representing a 1, and so on.
     - Neuron activation functions, learning rates for each neuron, and any dynamic update of learning rates (as explained in the question above) if it exists.
     - The energy/distance functions of your choice.
     - Any other tricks such as regularization, dropout, momentum method, etc.
   - The reasons as to why you chose a particular hyperparameter the way it is (e.g. why did you choose 100 hidden neurons, but not 50, why do you have 1 hidden layer but not 2?)
   - Your design process. It is unlikely that the first network you train will actually work. Write about your design process, your failures, together with your comments on why do you think a particular approach failed (e.g. I began with $\eta = 100$ and the algorithm just diverged, η was just too large).
   - A pseudocode of your final algorithm as described in (f) of Question 1 above.
   - A plot that shows epoch number vs the number of classification errors on both training and test images. Another plot that shows the epoch number vs the energy on both training and test images. Your test set should include all 10000 test images.
   - As usual, you should use your own code without any help from any external neural network/machine learning algorithms.

Your work will be graded mainly based on the depth and the quality of your report. Your final network should also achieve a decent success rate. You should be able to achieve around 95% success rate on the test set (all 10000 images). Very poor network performance will also result in a low grade.