EEE 443/543 - Spring 2025 - Project #4
Due: 03/07/2025, 11:00pm.

Note: It may be a good idea to attempt this project before our first quiz as a way to study for the quiz.
Note: Upload codes and reports as usual.

- **Q1 (100 pts)** In this computer project, we will use a neural network for curve fitting.

  1. Draw $n = 300$ real numbers uniformly at random on $[0, 1]$, call them $x_1, \ldots, x_n$.
  2. Draw n real numbers uniformly at random on $[-\frac{1}{10}, \frac{1}{10}]$, call them $\nu_1, \ldots, \nu_n$.
  3. 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.
  4. 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: Since this is a very simple network, you can manually derive the derivatives without using the BP 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.
  5. 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. If you are not getting good fit, try different hyperparameters until you do.
  6. Your report should include a pseudocode of your training algorithm including all gradient descent update equations written out explicitly. 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 usual, upload a copy of your code to box with the filename 04-IDNumber-LastName.py.