CX 4803 CML – Spring 2022

Assignment 8 (Final)

Be sure to show your code for the questions below. Include your program listings in your submitted pdf file.

1. Let E be the neural network error function, which is a function of the network weights and biases. Let $b_i^{(l)}$ denote the i-th component of the bias at layer l. Derive an expression for

$$\frac{\partial E}{\partial b_i^{(l)}}$$

and explain how this quantity is calculated during neural network training.

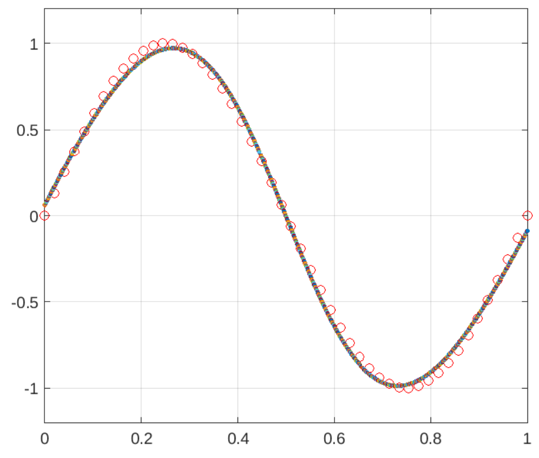
2. Reproduce Figure 5.3(b) in Bishop, which uses a neural network for regression to predict points on a sine curve. A two-layer (one hidden layer and one output layer) neural network is used. The input to the neural network is a scalar and the output is a scalar. The hidden layer has 3 hidden units. Use the tanh activation function, but note that *the output layer does not use an activation function* (i.e., it uses the identity or linear activation function) since the network is being used for regression.

   Your figure should plot the training data ($n = 50$ was used by Bishop) and a large number of predictions for test data (e.g., 100 or even 1000 test points). The curves formed by the training data and the test data may not line up as nicely as in Bishop's figure. But you may want to see how close you can get them to line up. You do not need to plot the outputs of the hidden units (dashed lines in Figure 5.3(b)).

   Write your own code for the neural network (including forward and backward propagation, and gradient descent). You may use the code `demo_nn.m` provided on Canvas (in the Apr-13 lecture notes folder) as a guide.

   If you have problems with the convergence of gradient descent, you may wish to try using a small learning rate (e.g., 0.001) and check that the error function is being reduced. Stop the gradient descent iterations in any manner you wish.

   As an example, I obtained the following figure, where the circles are the training points.



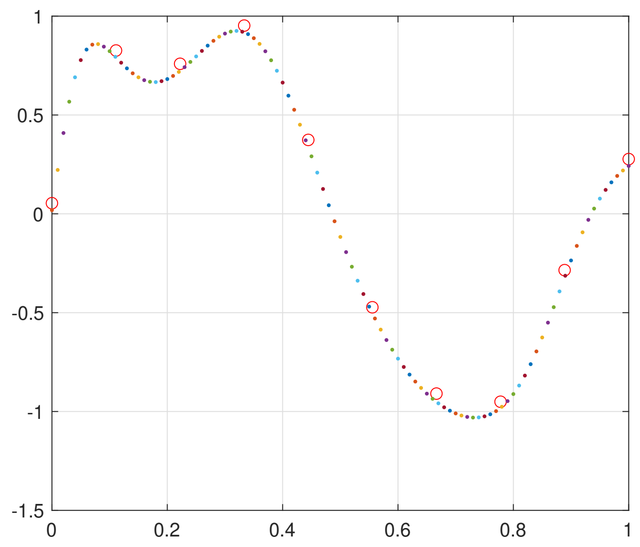
3. This question is the same as above, but now add some noise to the data, and also use fewer training points. For example, I used

```
fun = @(x) sin(2*pi*x);            % true function
x = (0:1/9:1);                     % 10 equally spaced points
y = fun(x) + 0.1*randn(size(x)); % noise added
```

Experiment with the number of layers and hidden units. For example, I used 3 layers (which has 2 hidden layers) and 10 hidden units per hidden layer. (I also used a learning rate of 0.01 and 100000 gradient descent steps.) I obtained the predictions (dots) for my training data (circles) in the figure below.

It seems clear that the large number of hidden units has allowed overfitting to occur. See Figure 5.9 in Bishop for more information.

Using your own code from the previous question, plot some results from neural networks with different numbers of layers and hidden units and with gradient descent trained with a different number of steps (and thus obtaining a different final value of the error function). Stopping the training early can also help avoid overfitting. Explain what you observe in the different plots.



4. **(Bonus)** For the above question (noise case), add regularization to your neural network. In particular, add

$$\frac{\lambda}{2} w^T w,$$

which is called *weight decay*, to the usual error function, where λ is a regularization parameter that you can experiment with. You would need to modify the computation of the gradients to take into account the regularization term (this sounds harder than it is). Design a test and show with a plot that regularization reduces overfitting. Be sure to explain how you set up the test data, and how you obtain your results with and without regularization. Reference: Bishop, Section 5.5.

Please fill out the CIOS survey for the course – that would be very helpful to me.
Thank you for being a great audience in this course!