SYDE552/BIOL487 Assignment: Regression, Classification, and Neural Networks

15 marks total
Due Feb 3

This assignment should be done in pairs. This assignment can be done in Matlab or Python. You will need .mat or .pkl files *perceptron-data*, *regression-data*, and *backprop-data*, from Learn, in the file *assignment1-data.zip*.

Submit (via Learn):

- A PDF containing the requested figures and answers.
- Your code.

1. [5 marks] Load the *perceptron-data* file. Implement the perceptron learning rule. Train a perceptron to discriminate the *vectors* with *labels* 0 from those with *labels* 1. Present the samples to the perceptron in random order. Plot the fraction correct vs. the step number for 10 separate runs (in a single plot). Stop each run after 300 steps.

2. [5 marks] Load the *regression-data* file. Plot 20 Gaussian functions with peak values of 1, widths of σ=0.2, and centres evenly spaced from -1 to 1. Calculate least-squares optimal regression weights for these basis functions, for approximation of the training data *trainx* and *trainy* (where the latter is a function of the former). Use the regularizer that we discussed in class. Plot the training data along with regularized regression models for the following values of λ (10^{-8}, 10^{-5}, 10^{-2}, and 10). (Plot the models' output in steps of 0.05.) Calculate the sum-squared error for both training and test data, for each case, and plot as a function of ln(λ).

3. [5 marks] Load the *backprop-data* file. Implement a 2-layer feedforward network with 10 hidden-layer units and a single output unit. Implement this from scratch, i.e. don't use a neural network library. The hidden units should have nonlinearity $z=\tanh(a)$ and the output unit should have nonlinearity $z=1/(1+e^{-a})$. Train the network using gradient descent and backpropagation to discriminate the *vectors* with *labels* 0 from those with *labels* 1. Plot contours of the network output for inputs over the range [-8,8], along with the training data. Use different colours for data points that the network assigns to different classes. Include plots for the network before training and after 1000 iterations. [Bonus: 1 mark] Describe a procedure for estimating $\partial E_n/\partial w_{ji}$ for each weight in a network using small perturbations to the **activation** of each node (i.e. an alternative to backpropagation). Roughly how much computation time would this take relative to backpropagation?