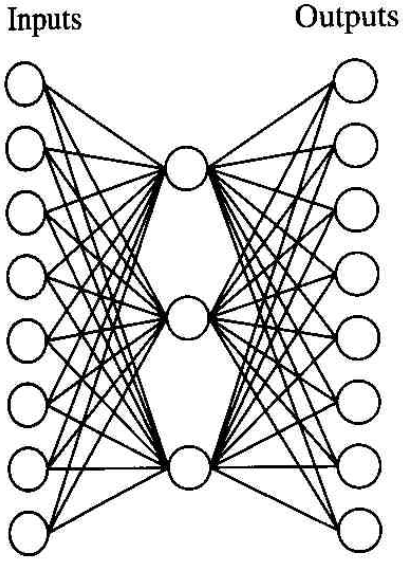
CS6140 Machine Learning

HW2B -Neural Networks

Make sure you check the syllabus for the due date. Please use the notations adopted in class, even if the problem is stated in the book using a different notation.

PROBLEM 1 [40 points] Autoencoder Neural Network

Consider the following neural network (left graph), with 8 input units (for data with 8 features), 3 hidden units and 8 output units, and assume the nonlinear functions are all sigmoid.



Input		Hidden Values				Output
10000000	→	.89	.04	.08	→	10000000
01000000	→	.15	.99	.99	→	01000000
00100000	→	.01	.97	.27	→	00100000
00010000	→	.99	.97	.71	→	00010000
00001000	→	.03	.05	.02	→	00001000
00000100	→	.01	.11	.88	→	00000100
00000010	→	.80	.01	.98	→	00000010
00000001	→	.60	.94	.01	→	00000001

a)The 8 training data inputs are identical with the outputs, as shown in the right side table. Implement this network and the back-propagation algorithm (square loss, sigmoid function) to compute all the network weights; you should initialize the weights with nontrivial values (i.e not values that already minimize the error).

HINT: on the trained network, you should obtain values for the hidden units somehow similar with the ones shown in the table (up to symmetry). Feel free to make changes to the algorithm that suit your implementation, but briefly document them.

b) Since the outputs and inputs are identical for each datapoint, one can view this network as an encoder-decoder mechanism. (this is one of the uses of neural networks). In this context, explain the purpose of the training algorithm. (I expect a rather nontechnical --but documented-- answer).

Hints: Some students worked easier from this algorithm, rather than the one from Mitchell's book
Others found this BackPropagation post from Brilliant easier to code up.

PROBLEM 2 [20 points] Autoencoder Neural Network

Implement the Autoencoder Network in PB1 using TensorFlow or PyTorch, and compare with your own implementation. You can use other NNet libraries, but TAs might not be able to help as much.

PROBLEM 3 [30 points] Classifier Square Loss Neural Network

Wine Dataset : train ; test
Implement a multi-class supervised Neural Network. Train and test on wine data (3 labels, 13 features). Layers should look like 13-input, k-hidden, 3-output. Use a square loss and a sigmoid activation function.

PROBLEM 4 [20 points] Classifier Square Loss Neural Network

Train and test the network in PB3 using TensorFlow or PyTorch.

PROBLEM 5 [30 points GR-ONLY] Classifier Max Likelihood Neural Network

Implement a multi-class supervised Neural Network using likelihod objective . Train and test on wine data.
Use the Maximum Likelihood (cross entropy) for loss function.
Use SoftMax activation function for output layer.
Use either sigmoid or RELU activation function for hidden layer

PROBLEM 6 [20 points GR-ONLY] Classifier Max Likelihood Neural Network

Train and test the network in PB5 using TensorFlow.

PROBLEM 7 [Extra Credit]

Run your Max Likelihood Neural Network on a medium-size dataset
Digits_small

..and on a larger dataset
Digits Large Dataset (Training data, labels. Testing data, labels): about 60,000 images, each 28x28 pixels representing digit scans. Each image is labeled with the digit represented, one of 10 classes: 0,1,2,...,9.