HW2
CS273A Machine Learning
Fall 2015
Due: Thursday October 15 11:59pm on EEE

As in HW1:
Allowed: Reading (but not copying) pseudocode and code in the recommended/optional books listed in the class Syllabus. Also, reading external pseudocode after you have tried to write your own code.
Not allowed: Other outside/external code reading or code use (eg. copying or execution).

For each problem turn in: 1-page written description of your approach and results, together with source code and I/O files proving your results.
Programming language: Your choice. (If this gets hard to grade we may limit it in later HWs.) If unsure, use Python.

In like manner to HW 1:

1. Implement and test a feed-forward artificial neural network (ANN or MLP = multilayer perceptron) with two layers of weights (i.e one layer of hidden units) and logistic transfer functions, using back-propagation of error (online or batch) as the learning algorithm. Parameters to vary include learning rate, and any parameters of the stopping criterion (eg. number of iterations and/or target accuracy); optionally also, the strength of a weight decay term (section 5.5 in Bishop). For each value of the parameters, report average error of the algorithm on both a training and a test set, along with statistical error bars on those quantities. For the problems that the ANN should solve, choose or invent two easy problems, e.g. (a) autoencoder and (b) some logic function on binary vectors such as exactly-one-on (coincides with xor/parity for 2 inputs, but is easier for more inputs For details on one way to approach these particular options, see forthcoming hints sheet from the TA). Show a visual representation (similar to a heat map) of the final trained weights for your best network. Probability distributions to use in reporting a single average error measurement includes a distribution (eg a narrow Gaussian around zero) over starting weight arrays, and a distribution on input patterns from which you draw your training and test sets; just say what distributions you chose.

2. Apply your network of #1 to a simple image classification problem, e.g. that of HW1 or more realistic, e.g. with a larger image such as 5 x 7. Report performance as above and again visualize the final trained weights, but now in an image-centric format for the first layer.

Extra credit:
- Add weight decay (section 5.5 in Bishop) to the exploration of problem 1 or 2 above.
- Or, generalize the Backpropagation of Error algorithm to include weight-sharing, and use that variant to obtain (perhaps!) better performance on #2 due to translation invariant features.
- Or, invent your own variation of the foregoing problems.