MACHINE LEARNING COMS 4771, HOMEWORK 2
Assigned February 14, 2013. Due February 28, 2013 before 1:00pm.

Here are the instructions for submitting your homework. Archive/package all of the files you are submitting as a single tarball or zip archive: "UNI-HW2.tar.gz" or "UNI-HW2.zip". For example, a compressed tarball would be "ir2322-HW2.tar.gz". Your homework should contain:

- a writeup (PDF, TXT, or PostScript)
- code (as Matlab M files, shorter code is generally better but include comments)
- any figures/pictures not included in the writeup (PDF or PostScript)
- if you have special instructions, include them as a plain text file called README.txt.

Submit your homework through CourseWorks by doing the following:

1 Log into https://courseworks.columbia.edu/
2 Click "Assignments" on the left side.
3 Choose the appropriate HW Folder to submit to.
4 Use the filename "yourUNI-HW2.tar.gz" or "yourUNI-HW2.zip".
5 Make sure that the "title" is yourUNI-HW1 (example: zz9999-HW1).
6 Add any special instructions in both the description and the README.txt.
7 Click "Submit" at the bottom to upload your file.
8 If you submit multiple times, only the last submission prior to the deadline will count.
9 If something goes wrong, ask the TAs for help.
10 In a dire emergency, if nothing else works, send your homework to the TAs.

Handwritten writeups are not allowed without prior approval.

All your code should be written in Matlab (other languages may be used only with prior permission from an instructor). Please submit all your souce files, each function in a separate file. Clearly denote what each function does, its inputs and outputs, and to which problem it belongs. Do not resubmit code or data provided to you. Do not submit code written by others. Identical submissions will be detected and both parties will get zero credit. Sample code is available on the Tutorials web page. Datasets are available from the Handouts web page. You may include figures directly in your write-up, or separately and refer to them by filename.

Each homework counts equally towards your grade (other than your worst which will be dropped). Points shown here for each problem indicate relative weights for this specific homework. As always, up to 10% bonus points are available for exceptional, relevant work going beyond what is asked.

1 Problem 1 (25 points)

Perceptron: Implement the linear perceptron in Matlab (using stochastic or gradient descent). Train it on **dataset2.mat**. Type "load dataset2" and you will have the variables X (inputs) and Y (labels) in your Matlab environment which contains the dataset. Use the whole data set as training.

1. Show with figures the resulting linear decision boundary on the 3D X data.

2. Show the binary classication error and the perceptron error you obtain throughout the run from random initialization until convergence on a successful run (some random inits may not converge or may require many iterations). Note the number of iterations needed. If using the non-stochastic algorithm, discuss the convergence behavior as you vary the step size (h).

2 Problem 2 (20 points)

Linear Separability:

2.1 (10 points)

Given two sets of vectors $S_1 = \{x_1, \ldots, x_n\}$, and $S_2 = \{z_1, \ldots, z_n\}$, consider the corresponding sets $X = \sum_{i=1}^{n} \alpha_i \mathbf{x}_i$, and $Z = \sum_{i=1}^{n} \beta_i \mathbf{z}_i$, where the coefficients α_i satisfy $\alpha_i \geq 0$, $\sum_{i=1}^{n} \alpha_i = 1$, and similarly $\beta_i \geq 0$, $\sum_{i=1}^{n} \beta_i = 1$. Show that either the sets S_1, S_2 are linearly separable or the sets X, Z intersect.

2.2 (10 points)

Let $\mathbf{x}$ be a binary d-dimensional vector (each attribute is either 0 or 1). Let $S = \sum_{i=1}^{d} x(i) \mod 2$. If S = 0, x is assigned to class 1. If S = 1, x is assigned to class 2. Show that the set of vectors generated according to the rule above is not linearly separable if $d > 1$.

3 Problem 3 (15 points)

3.1 Equivalence (10 points)

Consider a two layer NN expressed as:

$$f_k(x) = g\left(\sum_{j=1}^{H} w_{kj}^{(2)} h\left(\sum_{i=1}^{D} w_{ji} x_i + w_{j0}\right) + w_{k0}^{(2)}\right) \tag{1}$$

Assume that the hidden-unit activation functions $h(a)$ are given by the logistic sigmoid functions of the form:

$$h(a) = \frac{1}{1 + exp(-a)} \tag{2}$$

Show that there exists an equivalent network, which computes exactly the same function, but with hidden-unit activation functions given by hyperbolic tangents ($tanh(a)$). Hint: find the relation between logistic sigmoid and tanh, then show that the parameters of the two networks differ by linear transformation.

3.2 Nice Property (5 points)

Show that the derivative of the hyperbolic tangent (tanh) function can be expressed in terms of the function value itself.

4 Problem 4 (30 points)

Neural Networks:

4.1 K outputs (15 points)

Recall the network learning solution we derived in class (lecture 7, slide 16). Now lets generalize the derivation to K outputs (instead of one output):

1. Write the problem for K outputs. Be accurate with your notation.
2. Define the corresponding Lagrangian function L.
3. Find the stationary point of L. You should show the derived formulae for each subcondition you consider.

4.2 N inputs (15 points)

Again consider the network learning solution we derived in class for one output (lecture 7, slide 16). Now lets generalize the derivation to N inputs (instead of one input):

1. Write the problem for N inputs. Be accurate with your notation.
2. Define the corresponding Lagrangian function L.
3. Find the stationary point of L. You should show the derived formulae for each subcondition you consider.

5 Problem 5 (20 points)

Back-Propagation Implementation: Consider a two-layer network having 3 hidden units with tanh activation functions, and one linear output unit. We will use this network to approximate two functions: (a) $f(x) = x^2$, (b) $f(x) = |x|$

1. For each function, generate a random set of 50 data points, sampled uniformly in x over the interval $(-1, 1)$, and plot the pairs $\{x, f(x)\}$.
2. Implement Back-Prop and include your code. Your implementation can hardcode the specific assumptions in this problem.
3. For each function, use the data you generated to train your network (no cross validation is needed here, use your entire data for training).

4. Plot the resulting network functions on top of your generated data.
5. What can you say about the approximation error for each case? (How do they compare?)