Homework # 1

15-496/782: Introduction to Artificial Neural Networks
Dave Touretzky, Spring 2004

- Due January 21, 2002.
- Read HK&P chapter 5 first.
- Software you need is in `/afs/cs/academic/class/15782-s04/matlab/perceptron`
- Answers must be typed. Handwritten answers will not be accepted.

Problems

1. Suppose you want to train a perceptron on the following classification problem:

$$\mathit{Patterns} = \begin{bmatrix} 2 & 6 \\ 1 & 3 \\ 3 & 9 \end{bmatrix} \qquad \mathit{Desired} = \begin{bmatrix} 0 \\ 1 \\ 1 \end{bmatrix}$$

   Using inequalities expressed in terms of the weights w_0, w_1, and w_2, prove that the perceptron cannot learn this task.

2. The proof of the perceptron convergence theorem states that if $\hat{w}$ is a weight vector that correctly classifies all patterns, and $\overline{w}^{(\tau)}$ is the weight vector at step τ of the algorithm, then $\hat{w} \cdot \overline{w}^{(\tau)}$ increases at each step. Modify the `perceptron` program to demonstrate this by displaying the value of this dot product at each step. Turn in your source code and a sample run.

   Note: in order to do this you will need to know the correct weight vector at the start of the run. You can calculate this vector directly from the slope and y-intercept.

3. Run the `bowl` demo with learning rates of 0.01, 0.1, 0.142, and 0.15. Hand in a printout of each run. What can you say about the model's behavior at each learning rate?

4. Consider the function $f(x, y) = \exp\left(-(0.6y - 0.7)^2 - (x - 0.4)^2\right)$. The following code will graph f for you:

```
pts = 0 :  0.1 :  1;
[x,y] = meshgrid(pts);
z = exp(-((0.6*y-0.7).^2+(x-0.4).^2));
surf(x,y,z)
box on, rotate3d on
```

   Problem: (a) Train an LMS network to approximate $f(x, y)$ over the unit square ($0 \le x \le 1$, $0 \le y \le 1$). You may use the code in `lms3d.m` to get started, if you wish. (b) What is the shape of your approximation function? (c) By looking at the weights of your trained network, you can see the first degree polynomial that the neural network has devised to approximate f. Write down this polynomial, and hand it in along with the code you wrote to solve this problem.

5. How much information does it take to describe a two-input perceptron? The classical description uses a vector of three real-valued parameters: $\vec{w} = \langle w_0, w_1, w_2 \rangle$. But the perceptron's decision boundary is a line, which can be uniquely specified with just two parameters, e.g., slope and intercept.

   Jack says: "I claim a perceptron can be described with less information than three real numbers. Here's how I would do it with just two real values: set $s_0 = w_0/w_2$, and $s_1 = w_1/w_2$. From the description $\langle s_0, s_1 \rangle$, I can construct a weight vector $\langle s_0, s_1, 1 \rangle$ that behaves exactly the same as $\vec{w}$ for all inputs."

   Jill replies: "I claim a perceptron requires more than just two real numbers to describe. Consider the case where w_2 is negative. What will your approach do?"

   (a) Whose claim is correct, and why?

   (b) How much information does it really take to correctly describe a two-input perceptron? (Don't worry about the case where $w_2 = 0$.)