Problem Set 1 Due: Tuesday, October 11, 3:30pm.

1. Suppose we have an input vector with six elements: $x = (x_1, x_2, x_3, x_4, x_5, x_6)$. We compute the function:

$$f(x) = sigmoid\left(\log\left(\left(\left(max(x_1, x_2) * \frac{x_3}{x_4}\right) - (x_5 + x_6)\right) * 5\right) + \frac{1}{2}\right)$$

   What is the value of $f(x)$ when $x = (5, -1, 6, 12, 7, -5)$? Using the chain rule, compute the gradient of $f(x)$ for this value of x.

2. The next problems involve classification. We will suppose that there are K classes. Each class consists of elements drawn from a Gaussian distribution in a d-dimensional space. Both training and testing data should be drawn from these distributions with equal likelihood (eg., just use the same number of samples from each class). Unless we say otherwise, we assume that each distribution has the identity matrix as its covariance matrix.

   We will ask you to implement neural networks to perform classification. For this assignment, the networks should be fully connected. You should be able to do this assignment using Matconvnet (http://www.vlfeat.org/matconvnet/), which is easy to install, or LightNet (discussed in class). You should be able to keep the size of your networks and training data small enough so that you can run experiments on a normal laptop. If you wish to use a different environment for neural networks (eg, Caffe, Torch....) that is fine, but the TAs will only support Matconvnet or LightNet.

   These questions are pretty open-ended. It is part of the assignment for you to determine a good way to make them specific. As an example, you may need to determine whether a network of a particular size can solve a particular problem. You have to decide how you can tell whether a network is solving a problem. To do this, you will need to make some reasonable choices about how much training data to use, how long to train the network, and how to measure success. You might, for example, try training the network many times, and see how often the error on a test set is reduced below some threshold. As another example, you will have to figure out how big the input space should be (ie, what is d) to get an interesting result. Try to explain the choices that you have made.

   (a) Suppose we have two classes ($K = 2$), and that the mean values of the classes are separated by a distance of 10. Train a three-layer network that has one hidden layer with H units to perform classification. Describe fully the architecture that you used, including the loss function. Describe how the performance of the network varies with H, for some suitable measure of performance.

   (b) Given the conditions of Problem 2a, find some value of H and some amount of training data for which the you can overfit the data. That is, create a situation in which the training error becomes very small, but the error on a test set is still very high. Describe the choices you needed to make to create this situation.

(c) This is not a programming problem, but a paper and pencil exercise. What is the expected error of a Bayes optimal classifier on the problem 2a? How does this error vary if we vary the distance between the means of the two classes? Be explicit; give an equation showing the error as a function of the distance.

(d) Next, consider Problem 2a, but allow the distance between the means of the two classes to vary. As in that problem, we want to train a three-layer network to perform classification. Of course, we cannot expect to acheive classification results that are better than the Bayes optimal classifier. But if we want to train a network to get near this point, does this become more difficult as the distance between the classes varies? If yes, provide a quantitative estimate of this. That is, provide a graph showing how the difficulty varies with distance, for some suitable measure of difficulty.

(e) Next, consider this problem when the two classes do not have the same covariance matrix. Suppose, for example, one class has a covariance matrix that is the identity matrix, and the other has a covariance matrix that is two times the identity matrix. How would this change the results of the previous problem.

(f) Now, suppose that we vary K, the number of classes. Do this by choosing the mean of each class from a suitable uniform distribution. How many hidden units, H, do you need to achieve good classification results? How does this vary with K or with d, the dimension of the space? Display at least one graph showing how the number of hidden units needed varies with the number of classes.

(g) Suppose that for problem 2f you are allowed to use a deeper network. Does using a deeper network allow you to solve the problem with fewer overall weights in your network? It may be difficult to answer this question definitively, so try to formulate some reasonable set of experiments to consider it.

(h) Consider Problem 2g, but this time suppose that the elements of the classes are restricted to lie in a m-dimensional linear subspace, with $m << d$. So the means of the classes will lie in this subspace, and the covariance matrices will have rank m, and be chosen so that the covariance is zero outside this subspace. Does the benefit of using a deep network change compared to your answer to Problem 2g?