CPSC 340 Assignment 6 (due December 2)

PageRank and Neural Networks

1 PageRank

In class we discuss the PageRank algorithm for ranking nodes in a graph, and how it can be viewed in terms of a random walk on the graph: we start at some random node, and at each iteration we either (i) with probability α choose a completely random node in the graph or (ii) with probability $(1-\alpha)$ we choose a random neighbour of the current node. The ranking of each node is proportional to the number of times this random walk visits the node.

The file *arrowhead.mat* contains a dataset with the names of 104 participants in a 1993 meeting on numerical linear algebra, as well as a matrix X containing list of pairs of co-authors among these participants. Write a function that constructs the adjacency matrix for an **undirected** 104-node graph based on these links, and then runs a random walk to approximate the PageRank of the nodes.

1. Hand in your code implementing the random walk algorithm.
2. Who is the person with the largest PageRank?
3. If you normalize the PageRanks so that they sum up to 1, what is the median PageRank value?
4. What is the (normalized) PageRank of the inventor of Matlab?

Use $\alpha = 0.1$ for the above questions.

2 Neural Networks

The file `example_nnet.m` runs a stochastic gradient method to train a neural network on the *basisData.mat* dataset from Assignment 3. However, in its current form it gives a horrible fit to the data. Modify the training procedure to improve the performance of the neural network. Hand in your plot after changing the code to have better performance, and list the changes you made.

Hint: there are many possible strategies you could take to improve performance. Below are some suggestions, but note that the some will be more effective than others:

- Changing the network structure (*nHidden* is a vector giving the number of hidden units in each layer).
- Changing the training procedure (you can change the stochastic gradient step-size, use mini-batches, run it for more iterations, add momentum, switch to *findMin*, and so on).
- Transform the data (you could add a bias variables, standardize the features, standardize the targets, and so on).
- Add regularization (L2-regularization, L1-regularization, dropout, and so on).
- Change the loss function or the non-linearities (right now it uses squared error and tanh to introduce non-linearity).