Homework # 4

15-486/782: Artificial Neural Networks
Dave Touretzky, Fall 2006

- Due November 1, 2006
- Recommended reading: *Forward models: Supervised learning with a distal teacher* by Michael I. Jordan and David E. Rumelhart.
- Software you need is in `/afs/cs/academic/class/15782-f06/matlab/distal`
- Answers must be typed. Handwritten answers will not be accepted.

In this howework we will explore the use of neural networks as controllers. Following the material presented in the lecture on this subject, you are asked to build two controllers for the three link kinematic arm. One controller is a direct inverse model controller, while the other is a controller trained using the distal teacher discussed in the paper by Jordan and Rumelhart.

Preliminaries

1. The dataset used in this homework is in the file *distal_data.mat*, type: **load distal_data** to load the data. The training dataset is contained in the arrays *train_x train_q* and *train_qnet*. *train_x* contains a set of end effector (free end of the arm) positions in (x, y). *train_q* contains the corresponding joint angles in radians, (q_1, q_2, q_3), and *train_qnet* contains a transformation of the joint angles in *train_q* to values between -1 and 1. *train_qnet* is used in place of *train_q* for input and target output of the neural networks. The transformation is $(\cos(q_1 + \pi/2), \cos(q_2), \cos(q_3))$.
2. The matlab script *run* initializes the graphical display and sets up some important variables. Run this script before all other scripts and functions in the directory. You can play with the graphical arm by moving the sliders corresponding to the joint angles. The script *run* and a function called *test_controller* use a function *forward* that computes the forward kinematics of the three link arm. That is, it takes the joint angles as inputs and computes the end effector position in (x, y) as well as the positions of all the joints.
3. The script *train_fwd_model* was used to train the forward model weights, *fwd_Weight1* and *fwd_Weight2*. It uses the delta-bar-delta method (see Bishop, chapter 7) to adapt the learning rate and momentum. Feel free to use this code as the basis for the code you will be generating to train the controllers.

Problems

1. Write a program called *train_inv_model* to train a neural network, using backpropagation, as a direct inverse model to control the three link robotic arm. The network will take as input the desired position of the end effector and give as output the joint angles required to achieve the desired end effector position. For training, use *train_x* as input and *train_qnet* as output.

Note: you must use a much lower learning rate to train the inverse model than was used to train the forward model. Stop training when the error has changed by less than 1% of its value over 50 epochs. Hand in this code.

2. Test the performance of the inverse model controller. The function *test_controller(Weights1, Weights2)* produces a quiver plot of the deviation between the desired and actual position of the end effector. The arrows point from the desired position to the actual position. The larger the arrow, the larger the error at that point. Hand in this plot. Your results should be comparable to the corresponding results shown in the Jordan and Rumelhart paper.

3. The arrays *fwd_Weight1* and *fwd_Weight2* provided in *distal_data* are the weight values of a previously trained forward model. These weights were trained with the joint angles *train_qnet* as input and end effector positions *train_x* as output. Using these weights in a model of the forward kinematics, write a program that trains a neural network using the distal teacher approach. Initialize the weights as the final weight values found while training the direct inverse model. As before, stop training when the error has changed by less than 1% of its value over 50 epochs (this will take a while). Hand in this code.

4. Use *test_controller* to test the performance of the distal teacher controller. Hand in the quiver plot of the performance of the controller trained using a distal teacher. Again, your results should resemble the corresponding results shown in the Jordan and Rumelhart paper.

5. Compare the two approaches. (a) How does the performance of the distal teacher trained controller compare to the controller trained as a direct inverse model, based on the plots? (b) How many epochs were required to train each controller? (c) What was the final sum-squared error for each controllre?