

Final Project:
Course Value: 250 points

Deadline: May 17, 2011

Duplicate the results of the Jordan / Rumelhart paper, Figure 15. Use a robot arm with one meter long links, with the first joint set at (0,0).

a) Train a network to mimic the forward kinematics of the arm. You can do this by generating a 100 sets of random numbers in joint space. Constrain the first joint angle to $-\pi < \theta < \pi$ and the second and third joints to $0 < \theta < \pi$. Your training should look like the following (block diagram for illustration purposes only, please do not use Simulink).

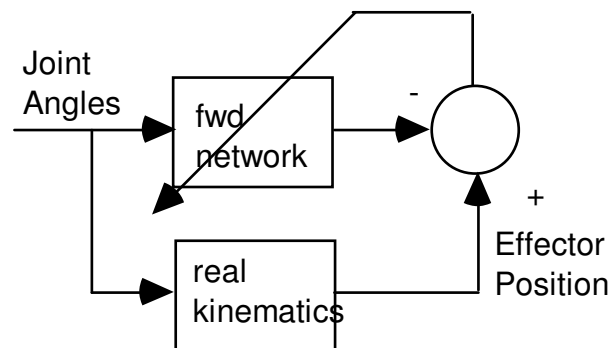


Figure 1: Training the Forward network

Plot the targets and network predictions together after training is complete.

b) Now add the inverse network, and train the inverse and forward networks simultaneously using the *distal teacher method*. The learning set-up is illustrated below.

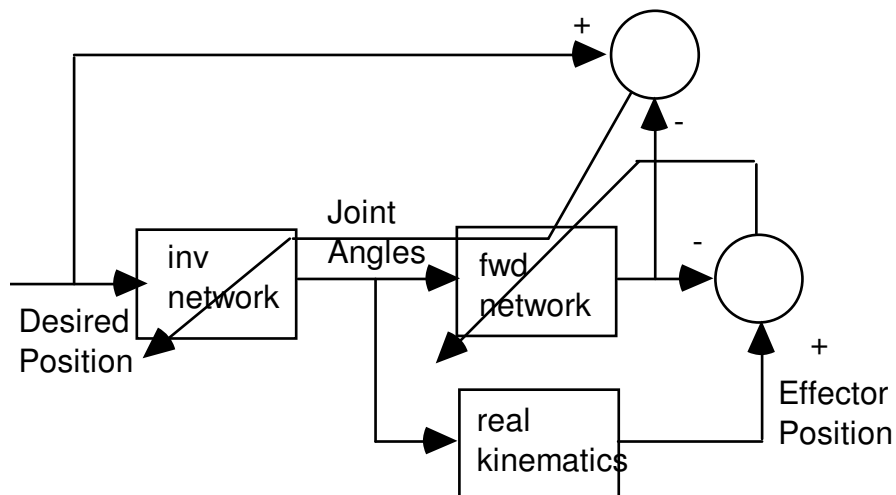


Figure 2: Training the Forward and Inverse Networks

First, generate a grid of targets in Cartesian space. You should use $x=-1:0.2:1$ and $y=0:0.2:2$. The targets generated by the real kinematics in part a) will not completely span this space, so the forward network needs further training. Note that you will use error back propagation to train each network, however, you will use a different set of deltas. Update the forward network based on the difference between the real kinematics and the forward network prediction. Update the inverse network based on the difference between the desired position in Cartesian space and the actual position. In order to update the inverse network, you need to back-propagate this difference through the forward network. You are welcome to try Levenberg-Marquardt training to speed up the training, but this is not required.

Once your inverse network is adequately trained, demonstrate the performance of your inverse network by testing all 121 sets of training points using the following configuration. Plot the desired and actual positions. A sample plot is displayed outside of my office.

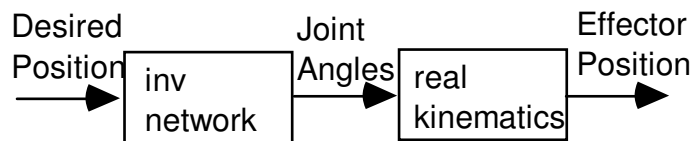


Figure 3: Testing the inverse network

What we have is a very exciting result. I've just developed a system where I can tell the controller network where I want the effector (end of the arm) to be positioned. The network computes a correct set of joint angles, and moves the arm accordingly.

Programming Hint: My results are based on inverse and forward networks that each have one hidden layer of 24 units. Limiting the number of hidden layers should make programming a bit easier. Also, keep in mind the dimensions of inputs and outputs. The inverse network is two input-three output. The forward network is three input-two output. When linked together, the networks look like this:

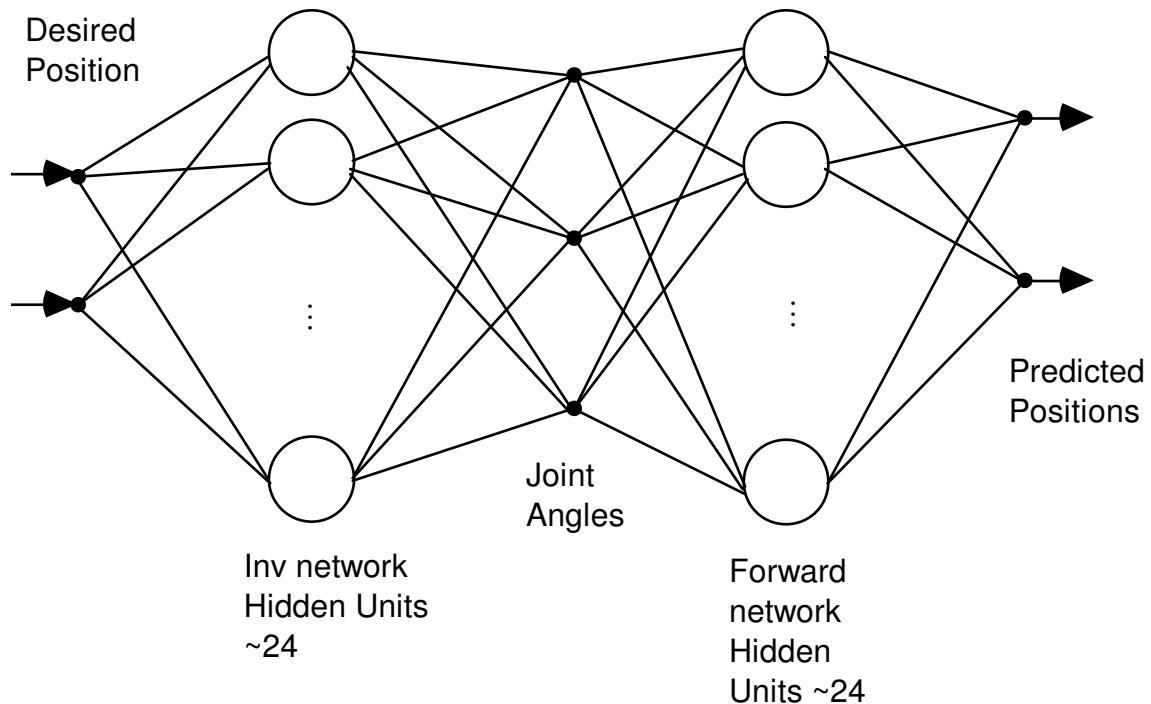


Figure 4: Inverse and Forward Networks together

I intend to share lots more hints in class, so don't miss it!