EECS/EEAP 484 Computational Intelligence, Fall 2015
Problem Set 7: Feedforward Neural Networks as Radial Basis Function Networks

We previously demonstrated that nonlinear mappings can be approximated by smooth surfaces generated by multilayer perceptron networks (feedforward neural nets). Good function approximation can be performed with a relatively small number of neurons in a single hidden layer. The challenge to such networks is discovering a good set of synaptic weights. The back-propagation technique is commonly used for engineering solutions, but it is slow, prone to local-minima traps, and is not biologically plausible. In contrast, biological systems can learn from single-exposure experiences, and the neurons involved presumably do not require global coordination of weight changes.

An alternative network is the "radial basis function" network, in which a single hidden layer is comprised of neurons with activation functions that are more complex than integrate-and-fire neuron models. In the integrate-and-fire model, we are treating "firing" in terms of the instantaneous spiking rate, which is abstracted as a monotonically increasing (e.g. sigmoidal) function of the weighted sum of inputs. Radial basis functions can be more complex, such as $g(\mathbf{x}) = e^{\|\mathbf{x}_c - \mathbf{x}\|^2/\sigma^2}$. For such functions, the output is strong only within some "radius" of approximately σ with respect to a point $\mathbf{x}_c$ in input space. Equivalently, a neuron with such an activation function has a "receptive field" in input space centered on $\mathbf{x}_c$ with a radius of approximately σ. Neurons in this hidden layer do not utilize a weighted sum of inputs. Rather, each of the terms of the inputs are treated separately, as a complete vector. Thus, input "weights" for such neurons are not meaningful. The neurons are "tuned" by selecting the centers ($\mathbf{x}_c$) and radii (σ) of their respective receptive fields. This is typically done without regard for the training target values (but may be guided by clustering of training points in input space).

While the hidden neurons of a radial-basis-function network are more complex than a multi-layer perceptron network, the output neuron (one neuron, for a single-output system) may be simpler—a linear perceptron. Only the (vector of) weights to this single output neuron need to be learned, which greatly simplifies the training process.

A conceptual problem with radial-basis-function networks is that the form of the activation functions is not biologically plausible, and the procedure for computing the output weights (e.g., via the pseudo-inverse) is also not biologically credible.

In this problem set, you will experiment with responses to these two issues. You will create the equivalent of radial-basis functions using two hidden layers. The first hidden layer is bipolar with tanh() activation functions, and the second hidden layer uses sigmoidal activation functions. There is a single output neuron, which is linear. (See the on-line class notes for greater detail). Your training data consists of mappings for an anthropomorphic robot arm from joint coordinates onto Cartesian hand coordinates. You only need to train for the x-coordinate of the hand. (The y-coordinate could be fit by a similar process with a separate network).

You will need to implement a strategy for setting the input synapses to the beta neurons. These values should be selected intelligently, since they get set once then are *not* updated during learning.

The provided starter code includes a solution for the weights from beta nodes to gamma node, using the pseudo-inverse. You should implement a more biologically-plausible strategy for setting these weights (i.e., random perturbations).

In addition to creating an intelligent strategy for setting beta-node input synapses and gamma-node synapses, you should evaluate the influence of:

- Range of random weights to set for inputs to alpha nodes
- Number of first-layer (alpha) hidden nodes
- Number of second-layer (beta) hidden nodes
- Means to initialize synaptic weights
- Search parameters for learning synaptic weight vector for inputs to gamma (linear output) neuron

Report on your observations in terms of fit error and convergence rates.