EECS 349 Fall 2015
Jason Brown - Homework 7

Simple Neural Networks

1) Hopfield Net

   a) train_hopfield(X) function implemented as described. The keys steps were:
      i) note:
      ii) build a weighted connection matrix, W, of size n·n
      iii) Sum across each training example such that $W(a, b)$ is the connection between nodes a and b
      iv) Return W

   b) use_hopfield(X) function implemented as described. The keys steps were:
      i) iterate over the n attributes that comprise an instance
      ii) Take the dot product of column i of the weighted connection matrix, W, and the instance under consideration, x
      iii) If the dot product ≥ 0, set $s[i] = 1$; otherwise 0
      iv) Repeat until convergence, which hopefully is a training example class

   c) Please see the attached progression until convergence of a noisy "0" into a correct 0. While this instance worked, if I modify the bits of the noisy "0" just slightly, I can get it to converge to "2." Therefore, I would want to build a more reliable classifier, though this Hopfield network works in current form.

   d) Please see the attached charts. Noisy data, at 20% and at 60%, did converge to the the correct number in 2 of 2 instances. Yes, this pattern looks like the pictures in the *1987 issue of IEE ASSP Magazine*, except that the author used a 128 node neural net, while we used 256 nodes; also, the author used a 25% flip bit percentage, while we used 20%; also, the author's pattern converged after 7+ iterations.

2) Perceptrons

   a) A one-layer percepton is capable of representing linear decision surfaces, as we developed in homework #3, of the equation $w_0 + w_1 \cdot x_1 + w_2 \cdot x_2 + w_n \cdot x_n = 0$. There are no x^y terms for $y \geq 2$, therefore, it can only represent linear decision surfaces.

   b) Yes, it is possible to use a multi-layer perceptron with linear activation functions to represent a non-linear decision surface. Multi-layer perceptron can learn XOR binary algebra, which is non-linear; they are also used for the hopfield network to learn digits, which are non-linear.

   c) The benefit of the sigmoid activation function is that it is differentiable, and therefore at deeper layers it is possible to decipher the contribution of earlier-level perceptrons. This can be useful in using gradient descent to find the best weights.

   d) Adding this sign function would make the function non-differentiable, and there'd be no way to determine where the error was coming from in predecessor nodes, breaking back-propagation.

3) Restricted Boltzman Machines (RBM)

   a) RBMs, also called "harmoniums", are a type of neural network that have the following properties:
      i) Multi-layer generative model is learned iteratively & greedily, layer-by-layer, one layer at a time. This allows for faster learning, rather than back propagation over the entire depth of the network.
      ii) After the weights of the first hidden layer is learned, consider that the input, and learn the next hidden layer; continue getting to smaller and smaller set of nodes per layer
      iii) Layer fully connected to next (hidden) layer; hidden layer has no connections within itself (Bipartite)
   b) A deep belief net is a multi-layer network with hidden layers & nodes, that uses an RBM at each layer to learn from the feature/attribute vector. The key feature is that the input data is represented & preserved in the hidden layer, and higher layers represent abstract representations, while lower levels remove some of the noise.

4) Scaling Learning Algorithms towards AI

   a) Kernel machines are non-parametric learning models which assume the form of the function to be learned, and then learns the best weights for that form from the training data. Kernel machines is a superset that includes K-nearest neighbors, modern kernel machines, gaussian mixture models, and multi-layer networks.
   b) Limitations of kernel machines include:
      i) Trade-off between breadth (space) and depth (time) - A perceptron is an example of a shallow architecture. A circuit to do multiplication is an example of a deeper architecture. Deeper architecture takes longer time to train and execute, but it is more expressive in what it can represent.
      ii) Curse of Dimensionality - Kernel machines are really template-matchers; simple & shallow, that takes an input and return a transformed output.
      iii) Local kernel - May get stuck in a non-globally optimal solution
   c) Deep architectures are multilayer networks where the inner layers are hidden, and comprised of parameterized non-linear modules. Each hidden layer can be thought of as representing an intermediate output getting towards the final output, a higher abstraction of the input.
   d) Deep architectures are a way around the limitations of kernel machines. There are major benefits:
      i) Training each layer as an auto-associator saves time versus RBM
      ii) Trading simple transformation into deeper, multi-layer transformation
      iii) Deep architecture trades breadth (kernel machine) for depth; to represent a complicated learning pattern; however, deep architecture is tough to determine what it's actually *learning*
      iv) Training each layer greedily, as if there were no additional layers, saves time, and allows for multiple layers of abstraction