Neural Networks

Project Schedule

Introduction Paper due 9/25/2014

Demonstration of the Equivalence of Various Perceptrons Networks due 2014-09-23

Statement and Proof of Perceptron Training algorithm due 2014-09-30

Turing Equivalence 2014-09-30

Universality Theorem 2014-10-07

Statement and Proof of Backpropogation algo? 2014-10-14

Overview of Neural Networks

A neural network is a system in which many similar functions, called *neurons*, are composed together in order to classify inputs, perform some computation, or approximate some function. Neural networks are used today in various machine-learning applications such as handwriting- or speech-recognition, and have several interesting mathematical properties.

Definition of a Neuron

A neuron is a function $f(x_1, x_2, \ldots, x_n) = \phi(\Sigma_{i=1}^{n} w_i x_i)$, where $\{w_1, w_2, \ldots w_n\}$ is a set of weights, with each weight w_i corresponding to an input x_i, and where ϕ is the activation function that determines the output of the neuron based on the sum of the products of the weights and inputs. For the sake of brevity we will also notate the weight and input vectors as $\vec{w}$ and $\vec{x}$, respectively.

A neuron could be also thought of as a partial application of ϕ and $\vec{w}$ over the factory function $F(\phi, \vec{w}, \vec{x})$.

There are two popular activation functions, the perceptron and the sigmoid function. The perceptron can be defined as the function

$$\phi_P(x) = \begin{cases} 0 & : x+b \leq 0 \\ 1 & : x+b > 0 \end{cases}$$

where b is a bias that is specific to the neuron.

The sigmoid function can be defined as $\sigma(x) = \frac{1}{1+e^{-x}}$. Though bearing some resemblance to the perceptron, the sigmoid function has the advantage of being both smooth and differentiable. These properties make training a neural network much easier.

Demonstration of the Equivalence of Various Perceptron Networks to Certain Boolean Logic Functions

Consider the following network consisting of one perceptron-type neuron, with inputs $\binom{x_1}{x_2}$.
Let $\vec{w} = \binom{1}{1}$ and $b = -1$. Compare the behavior of the network to that of the AND function:

x_1	x_2	$\vec{x} \odot \vec{w}$	$\vec{x} \odot \vec{w} + b$	Output	x_1 AND x_2
0	0	0	-1	0	0
0	1	1	0	0	0
1	0	1	0	0	0
1	1	2	1	1	1

Note how the network is equivalent to the AND function over the inputs for which AND is defined.

Now let $\vec{w} = \binom{1}{1}$ and $b = 0$. Compare the behavior of the network to that of the AND function:

x_1	x_2	$\vec{x} \odot \vec{w}$	$\vec{x} \odot \vec{w} + b$	Output	x_1 OR x_2
0	0	0	0	0	0
0	1	1	1	1	1
1	0	1	1	1	1
1	1	2	2	1	1

Now, simply by changing the weights, the network output becomes equivalent to the OR function over the domain of OR.

TODO: expand this section to include NAND (since all logic functions can be expressed with NAND?) and XOR (I read that there were difficulties in creating this network back in the 70's before backpropogation).

Statement and Proof of Correctness for a Training Algorithm for a Perceptron

Statement and Proof of the Turing-completeness of Neural Networks

Statement and Proof of the Universality Theorem