Neural Networks

Christopher Hittner

March 29, 2017

Introduction

(Kinda) Needed Concepts

We will briefly utilize the following mathematical concepts:

- Addition
- Multiplication
- Matrices
- Derivatives

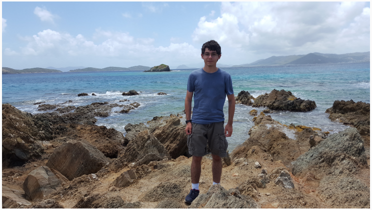
History

- Donald Hebb - The Organization of Behavior (1949)
  - Defined a form of neural network where connections that are frequently used are strengthened.
- Bernard Widrow and Marcian Hoff (Stanford Univ., 1959)
  - Developed the theory behind ADALINE (ADAptive LINEardi Neuron) Networks.
  - Neural networks that sum a set of inputs and returns a binary response based on its firing threshold.
- Perceptrons: an introduction to computational geometry (1969)
  - Published by Marvin Minsky and Seymour Papert.
  - Suggested that Neural Networks could not handle the XOR problem.
  - Contributed to the first AI winter.

Recent History

- AlphaGo
  - Google AI capable of beating world champions in Go.
  - Uses Monte Carlo tree search with probabilities generated by a neural net.
- Google DeepDream
  - Uses Convolutional Nets to enhance parts of images.
  - Similar methodology to the backpropagation algorithm
  - `https://deepdreamgenerator.com/`
- Deep Photo Style Transfer
  - Convolutional Neural Net that modifies the style of images.
  - `https://github.com/luanfujun/deep-photo-styletransfer`

Sample Dreams



(a) My unaltered self

(b) Me as a Van Gogh

(c) Me as a DaVinci

(d) Such dream. Wow.

Neural Networks

Definition

- A Neural Network is a computational model inspired by the human brain.
- Made of layers of neurons and synapses between them.
- Neuron - Unit that takes input and send output through synapses.
- Synapse - Connection between neurons.

Mathematical Model

Neural Networks consist of neurons with weights between them. (ex: n_i and n_j with weight w_{ji} between them).

Every neuron has input and output activations based on the weights and neuron outputs:

$s_j = \sum_{i=0}^{N-1} w_{ji} x_i$

$y_j = f_j(s_j)$, where f_j is the activation function of n_j, which is shared across its entire layer.

For neurons in the input layer, the x_i values will instead be the inputs to the network. Elsewhere, they will be the outputs from the previous layer.

Mathematical Model

Linear Algebra can be used to represent the model in the form of matrices and vectors:

$\vec{s_i} = W_i \vec{x_i}$

$\vec{y_i} = f_i(\vec{s_i})$

In this case, i represents the layer for which the values are being computed, and $\vec{x_i}$ can be either the network input $\vec{x}$ or the previous layer's output $\vec{y}_{i-1}$.

Neural Network Models

Types of Neural Networks

There are numerous variations of the neural network model:

- Feedforward Neural Net
    - Perceptron
    - ADALINE
    - Convolutional Neural Net
- Recurrent Neural Net
    - Hopfield Networks
    - Jordan/Elman Networks

Feedforward Networks

Activations feed forward sequentially through the network.

- ADALINE (ADAptive LINear Element)
  - Developed by Widrow and Hoff in 1959.
  - Single layer network that uses the unit step function as its activation function.
- Perceptron
  - Developed by Frank Rosenblatt in 1957.
  - Generally uses unit step as its activation function.
  - Can be expanded to multiple layers.
- Both use gradient descent to derive error in training.

Recurrent Neural Networks

Network that uses repeating layers to compute results.

- Hopfield Network
  - Output activations update continuously until stable value is reached.
  - At time i, $\vec{y}_{i+1} = W\vec{y}_i$
  - Network execution will converge on a value.
- Elman Network
  - Takes a set of time-indexed vectors $\vec{x}_i$.
  - At time i, $\vec{s}_{i+1} = f(W\vec{x}_i + R\vec{s}_i)$
  - $\vec{y}_i = f(W\vec{s}_{i+1})$
  - Jordan Network is similar, but uses $\vec{y}_i$ instead of $\vec{s}_i$ to compute $\vec{s}_{i+1}$.

Convolutional Neural Networks

- Networks based on the function of the visual cortex.
- Use filters spanning input to interpret and reduce data to an answer.
- Applies ReLU function to filter outputs.

Training Algorithms

Training Algorithms

Training algorithms are procedures used to teach neural networks to correctly respond to inputs. A few examples of training algorithms include:

- Perceptron Rule
  - Learning rule for single layer perceptrons.
- Hebbian Rule
  - Rule based on conditioning that does not require supervision.
- Backpropagation
  - General algorithm for multilayer networks.

Perceptron Rule

Training rule used to train perceptrons to correctly classify inputs.
Given an input vector $\vec{x}$ and a target vector $\vec{t}$:

1. Compute the output $\vec{y}$ of the network.
2. Determine the error $\vec{e} = \vec{t} - \vec{y}$.
3. Update the weight matrix $W' = W + \vec{e} * \vec{x}^T$.

This process can be repeated for as many training pairs as necessary, and for as many rounds as needed.
Limitation: Can only be used on linearly separable data.

Backpropagation

- Training algorithm for multilayer networks that takes advantage of the chain rule to propagate error through the network.
- We say that the error $E = 0.5(t - y)^2$ so that $\vec{e} = \frac{\partial \vec{E}}{\partial \vec{y}} = \vec{y} - \vec{t}$.
- We seek to determine $\frac{\partial \vec{E}}{\partial W}$ for each weight matrix W.
- On each layer, we compute a sum vector $\vec{s}$ and an output vector $\vec{y} = f(\vec{s})$ in order to feed forward.
- Error is propagated backwards by determining $\frac{\partial \vec{y}_i}{\partial \vec{s}_i}$ for each layer and $\frac{\partial \vec{s}_i}{\partial \vec{y}_{i-1}}$ between each layer.
- To determine change to each weight matrix, we must compute $\frac{\partial \vec{s}_i}{\partial W_i}$ for each matrix.

Backpropagation

The needed derivatives are as follows:

$$\frac{\partial \vec{y}_i}{\partial \vec{s}_i} = \frac{\partial f_i}{\partial \vec{s}_i} = \begin{bmatrix} f'(s_{i,0}) & 0 & \dots & 0 \\ 0 & f'(s_{i,1}) & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & f'(s_{i,n-1}) \end{bmatrix}$$

$\frac{\partial \vec{s}_i}{\partial \vec{y}_{i-1}} = \frac{\partial}{\partial \vec{y}_{i-1}} W \vec{y}_{i-1} = W_i^T$
$\frac{\partial \vec{s}_i}{\partial W_i} = x_i^T$
Using these equations, we can use the chain rule to propagate error through a network.

Backpropagation

- Vanishing gradient problem
  - Propagated error approaches zero as it passes through multiple layers.
- Multi-level hierarchies
  - Train layers separately to solve parts of a problem.
- Long Short Term Memory
  - Networks that can remember values for short or long periods of time.
- Residual Networks
  - Passing the network inputs into deeper layers.

Hebbian Rule

- Associative learning rule that can be supervised or unsupervised.
- In the supervised version, the weight matrix is continuously updated like so: $W' = W + \vec{t} * \vec{x}^T$
- In the unsupervised version, $W' = W + \vec{y} * \vec{x}^T$.
- Limitation: Has difficulty with non-orthogonal inputs.

Kohonen Rule (Self-organizing Map)

- Unsupervised learning model based on Hebbian rule.
- Generates a low-dimensional representation of data.
- First layer builds a linear sum from the input vector.
- Second layer is a recurrent layer that applies ReLU function.
    - Steady-state result of second layer is the end result.
    - We call this a competitive layer.

Kohonen Rule (cont.)

In competitive layer, we have the following properties:

$$W_{ij} = 1 \text{ if } i == j \text{ else } -\epsilon$$

$$\text{where } 0 < \epsilon < \frac{1}{S-1}$$

$$S = \text{layer size}$$

$$f(\vec{x}) = relu(\vec{x}) = x \text{ if } x > 0 \text{ else } 0$$

The result of this is that any input will decrease until only one dimension of input has a non-zero value, resulting in a steady-state solution.

Kohonen Rule (cont.)

In training the network, we seek to find vectors that categorize the data.

We compute output $\vec{y}$ such that only one dimension of $\vec{y}$ is nonzero.

Then, $\vec{W}'_{rowi} = \vec{W}_{rowi} + \alpha(\vec{y} - \vec{w}_{rowi})$, where α is the learning rate.

This causes row i to become closer to the input vector $\vec{x}$.

Additional Material

Intelligent Machinery, A Heretical Theory

Alan Turing, 1948

The Organization of Behavior

Donald Hebb, 1949

An Adaptive ADALINE Neuron Using Chemical Memsistors

Bernard Widrow, 1960

Mastering the Game of Go with Deep Neural Networks and Tree Search

Google DeepMind, 2016

Neural Net Design

Martin Hagan, Howard Demuth, Mark Hudson Beale, and Orlando De Jesus