Deep Learning Notes (WIP)

Ryan Gehring

Contents

Theory

Historical Context and Biological Motivation

Neural networks are learning algorithms which take inspiration from biological processes in the human brain. While the algorithms have been around for decades, it's only recently that their mass popularity has been rekindled due to some new training paradigms which enable the training of several layer, 'deep' neural networks against large datasets. Due to complicated parametrizations, many of the statistical properties haven't been worked out, but neural networks have empirically performed well on classical machine learning tasks and benchmarks and have even achieved best in class results on a number of problem domains.

Model Structure: Classical Neural Networks

A classical neural network is a weighted directed graph consisting of nodes called *neurons* connected by weighted directed edges which I guess I will term *axons* after the biological nomenclature (I'm not aware of formal terminology for edges in a neural network).

Neurons in neural networks are arranged in layers. Each layer forms a bipartite graph with it's neighboring layers. In other words, all axons are directed from nodes in layer i to layer $i \pm 1$.

The first layer of a neural network is termed the input layer, and each neuron in the input layer is the raw value for a predictive variable x_i from a training dataset. The inputs are propagated along each axon of the graph to the next layer.

Neurons in every layer except the input layer are characterized by an S-curved *activation function* which operates on the inputs sent along axons. The biological inspiration is that apparently neurons are electrical noise gates—they fire an electrical response only when they receive an electrical signal above some amount.

A popular activation function is the sigmoid activation function:

$$f(z) = \frac{1}{1 + e^{-z}} \tag{1}$$

In a *feed forward* neural network, there are a finite number of *hidden layers* and a final *output layer*. The hidden layers are termed hidden because some form of supervised task is conducted by the output layer, where output is measured, but hidden model layers lack a direct interpretation on their own, useful only in that they contribute to the final output score. However, it is relevant to note that there are occasionally empirical interpretations that arise in specific problem domains. For example, in an image recognition domain, one layer of the network might empirically seem to behave as an edge detector, while another layer might recognize certain types of objects from the relationships of the edges.

For a connected feedforward network where every node in layer i is connected to every node in layer $i+1$), the activation for a node j in layer $i+1$ is given as the activation function evaluated at the dot product of the activations of the neurons in layer i with their axon weights $\mathbf{w}_i^j$. (As noted above, the input layer activations are just the raw training data: $a_j^0 = x_j$).

This algorithm is called forward propagation.

$$a_j^{i+1} = f(\mathbf{w^i} \cdot a^i) \tag{2}$$

From the above you can see that a neural network with only an input layer and an activation layer is mathematically equivalent to a logistic regression.

In matrix form, for a whole layer, and adding a constant neuron to each layer called a *bias unit* which receives no input, and simply adds a constant amount:

$$\mathbf{a}_{i+1} = f(\mathbf{W}\mathbf{a}_i + \mathbf{b}_i) \tag{3}$$

Theory continued on next page...

Training and Backpropagation

A cost function is defined as the sum of squared prediction error plus a regularization term proportional to the magnitude of the weights. For one example our cost function is:

$$J(\mathbf{W}, \mathbf{b}) = \frac{1}{2} (y_i - h_i)^T (y_i - h_i) + \sum_{axons} w^2 \tag{4}$$

These models are commonly fit via batch gradient descent or other numerical methods.

Autoencoders and Self Taught Learning

Neural networks with many inputs but a restrictive hidden layer can be used to learn encodings for input data. These models are used for feature detection, compression, and also self-taught learning, in which autoencoded representations of input vectors are fed as input to a supervised algorithm.

Image Processing - Convolution and Pooling

Pixel data from images is immediately high dimensional, however in natural images nearby pixels tend to be similar, and aggregate statistics do an excellent job at summarizing large patches of images. We can use an autoencoder to learn features over small patches of an image and then apply the feature detector over each such patch of the image via a process called convolution.

Pooling is the action of computing summary statistics over large, exclusive patches of convolved features to obtain a much lower dimensional representation of the data. Generally convolved and pooled images are then used as input to a classifier.

Deep Learning

The technique of stacking autoencoders is a way of pursuing greedy, layerwise training on large neural networks. Train an autoencoder on the raw input, then train one using the output of that, do this layer by layer, then finally train on the supervised task.