Introduction to Neural Networks

Prof. Dennis Sun

Data 401
Where We Left Off

- The goal of machine learning is to estimate (a.k.a. “learn”) a function \( f : \mathbf{x} \rightarrow y \) based on training examples \((\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)\).
- Linear regression assumes
  \[
  f(\mathbf{x}) = \beta_0 + \beta_1 x_1 + \ldots + \beta_d x_d. 
  \]
  The model may contain non-linear terms like \(x_j^2\) or \(x_j x_k\), but these terms must be specified in advance.
- To find the optimal \( f \), we choose \( \beta \) to minimize some loss function, e.g.,
  \[
  L(\beta) = (y - f(\mathbf{x}))^2. 
  \]
  In general, this optimization can be carried out using gradient descent, by repeatedly calculating
  \[
  \beta \leftarrow \beta - \eta \frac{\partial L}{\partial \beta}. 
  \]
Limitations of Linear Regression

Linear regression requires us to specify the functional form in advance. If we get it wrong, then the performance will not improve, no matter how much data we have.
Advantage of Neural Networks

- Deep learning = neural networks
- "Most learning algorithms" includes linear regression

Data and machine learning

Performance vs. Amount of data

- New AI methods (deep learning)
- Most learning algorithms

Andrew Ng
A neuron is a node, and a synapse is an edge.

- Each synapse multiplies the value of the upstream neuron by a constant.
- Each neuron sums the values from the incoming synapses and then applies an activation function $g$.

Suppose neuron $A$ uses the ReLU activation function. Then:

$$A = g((-3)(1) + (2)(0) + (.5)(-2)) = g(-4) = 0.$$
Suppose each node in the network below uses the ReLU activation function. What is the value of neuron $B$?

The neuron labeled $?$ has a value of

$$g((3)(-1) + (-2)(0)) = g(-3) = 0,$$

so $B$ has value of

$$g((-1)(0) + (1)(5)) = g(5) = 5.$$
Structure of a Neural Network

Input Layer \[ \rightarrow \] Hidden Layers \[ \rightarrow \] Output Layer

\[ h_1^{(1)} = g\left( \sum_{j=1}^{3} w_{j1} x_j \right) \]

\[ h_2^{(1)} \]

\[ h_3^{(1)} \]

\[ h_4^{(1)} \]

\[ h_1^{(2)} \]

\[ h_2^{(2)} = g\left( \sum_{j=1}^{4} w_{j2} h_j^{(1)} \right) \]

\[ y = \sum_{j=1}^{2} w_{j3} h_j^{(2)} \]
Linear Regression as a Neural Network

Exercise: How would you draw the linear regression model

\[ \hat{y} = \beta_0 + \beta_1 x_1 + \beta_2 x_2 \]

as a neural network? What is the activation function?
Why Activation Functions?

What would happen if we did not have an activation function $g$? Let's consider a simple network:

![Diagram of a neural network with two input nodes, two hidden nodes, and one output node. The weights $w_{ij}$ are shown connecting the nodes.]

Remember that we do not observe the values of hidden neurons $h$, nor the weights $w$. Express $\hat{y}$ as a function of $x_1$ and $x_2$.

$$\hat{y} = w_1^{(2)} h_1 + w_2^{(2)} h_2$$

$$= w_1^{(2)} (w_{11}^{(1)} x_1 + w_{21}^{(1)} x_2) + w_2^{(2)} (w_{12}^{(1)} x_1 + w_{22}^{(1)} x_2)$$

$$= (w_1^{(2)} w_{11}^{(1)} + w_2^{(2)} w_{12}^{(1)}) x_1 + (w_1^{(2)} w_{21}^{(1)} + w_2^{(2)} w_{22}^{(1)}) x_2$$

Since the weights $w$ are unknown parameters, the neural network “collapses” to linear regression if there is no activation function.
The activation function prevents a neural network with a hidden layer from “collapsing” to linear regression.

In fact, it can be shown that a neural network with 1 hidden layer and the sigmoid activation function can approximate any function \( f : \mathbb{R}^d \rightarrow \mathbb{R} \) to any desired accuracy, provided that there are “enough” neurons in the hidden layer.

This result is known as the universal approximation theorem.

This theorem is not as useful as it sounds. Just because the network can approximate the function does not mean that we can estimate it from our data.