10.1 Single Layer Neural Networks

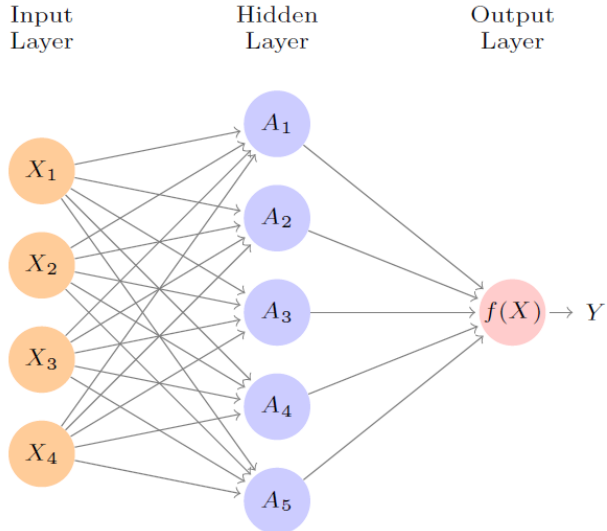
Deep Learning

- Deep learning is primarily something called *neural networks*.
- The content in this chapter is slightly more challenging than that in other chapters, so we will take our time.

Neural Networks

- A neural network takes an $n \times p$ input matrix X and builds a nonlinear function $f(X)$ to predict the response y.
  - What distinguishes neural networks from other methods is the *structure* of the model.

Example Structure



Example Details

- The response is quantitative, with $p = 4$ predictors.
- The features make up the *input layer*.
- Each input layer feeds into each of $K = 5$ *hidden units*.
- Finally, the hidden units feed into the output.

Single Layer Neural Networks

The neural network model takes the form

$$
\begin{aligned}
f(X) &= \beta_0 + \sum_{k=1}^{K} \beta_k h_k(X) \\
&= \beta_0 + \sum_{k=1}^{K} \beta_k g\left(w_{k0} + \sum_{j=1}^{p} w_{jk} X_j \right)
\end{aligned}
$$

Single Layer Neural Networks

These are built up in steps.

- First the K *activations* A_k in the hidden layer are computed as functions of the input features

$$A_k = h_k(X) = g\left(w_{k0} + \sum_{j=1}^{p} w_{jk} X_j\right)$$

  where $g(z)$ is some prespecified nonlinear *activation function*
  - We can think of each A_k as a different transformations of the original features.

Single Layer Neural Networks

- Then the K activations are fed into the output layer:

$$f(X) = \beta_0 + \sum_{k=1}^{K} \beta_k A_k$$

  a linear regression model in the $K = 5$ activations.
- All the parameters $\beta_0, \ldots, \beta_K$ and $w_{10}, \ldots, w_{Kp}$ need to be estimated from the data.
- For quantitative response, the model is fit by minimizing $\sum_{i=1}^{n}(y_i - f(x_i))^2$

Activation Functions

- In early instances of neural networks, the sigmoid function was favored

$$g(z) = \frac{e^z}{1 + e^x}$$

  - This is the same function used in logistic regression to convert a linear function to probabilities.

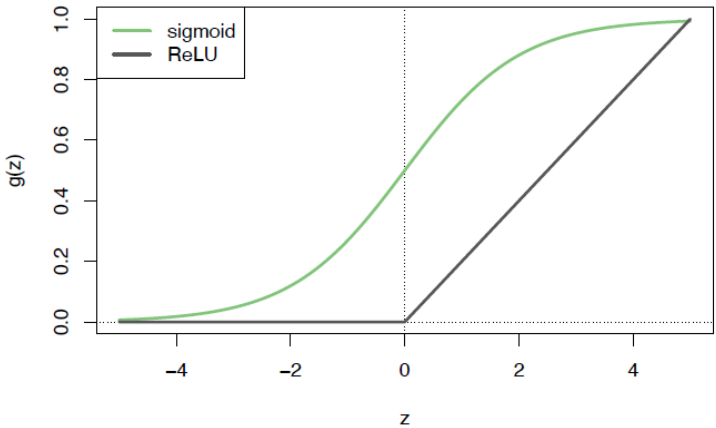
Activation Functions

- Today, the typical choice is *ReLU (rectified linear unit)* functions:

$$g(z) = (z)_+ = \begin{cases} 0 & \text{if } z < 0 \\ z & \text{otherwise} \end{cases}$$

  - These can be computed and stored more efficiently than a sigmoid activation.
  - The constant w_{k0} can shift the inflection point away from 0.

Activation Functions



Single Layer Neural Networks

So basically, the example model

- derives 5 new features by computing 5 different linear combinations of X
- then squashes each through a nonlinear activation function
- and the final model is linear in these derived variables

Single Layer Neural Networks

- Note: the nonlinearity property of the activation functions is necessary.
  - Otherwise, the model would collapse into a simple linear model in X.
  - This also allows us to capture complex nonlinearities and interactions.

Example

Suppose we have $p = 2$ input variables and $K = 2$ hidden units with $g(z) = z^2$ and suppose $\beta_0 = 0$. $\beta_1 = 1/4$, $\beta_2 = -1/4$, $w_{10} = 0$, $w_{11} = 1$, $w_{12} = 1$, $w_{20} = 0$, $w_{21} = 1$, and $w_{22} = -1$.

- Because $A_k = h_k(X) = g\left(w_{k0} + \sum_{j=1}^{p} w_{jk} X_j\right)$,

$$h_1(X) = (0 + X_1 + X_2)^2$$
$$h_2(X) = (0 + X_1 - X_2)^2$$

- Finally, $f(X) = \beta_0 + \sum_{k=1}^{K} \beta_k h_k(X)$ so we get

$$\begin{aligned} f(X) &= 0 + \frac{1}{4}(0 + X_1 + X_2)^2 - \frac{1}{4}(0 + X_1 - X_2)^2 \\ &= \frac{1}{4}[(X_1 + X_2)^2 - (X_1 - X_2)^2] \\ &= X_1 X_2 \end{aligned}$$

  - (The sum of two nonlinear transformations of linear functions can give us an interaction!)