10 Neural Networks + Deep Learning

$Y = f(\underline{x}) + \varepsilon$ $\qquad$ $f(\underline{x})$ = multiple regression, tree, bagged trees, random forest, boosted trees ...

Neural networks specify $f(\underline{x})$ as a composition of functions.

10.1 Single layer neural networks

A feed-forward single layer neural network uses p variables in $\underline{x}$ as input layer + feeds these into K hidden units (we pick K).

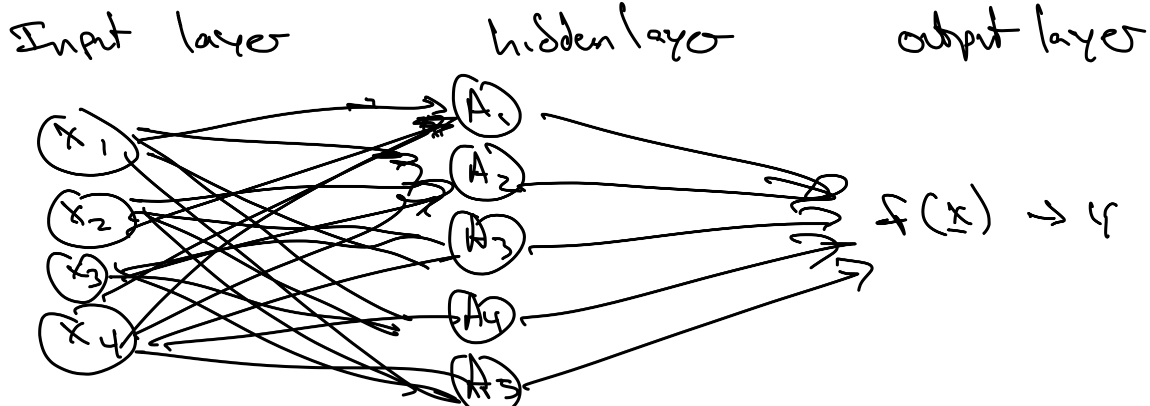
$$f(\underline{x}) = \beta_0 + \sum_{k=1}^{K} \beta_k \, g\left(w_{k0} + \sum_{j=1}^{p} w_{kj} x_j \right)$$

g is called an activation function to produce K activations:

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

note these activations are a hidden layer in the regular regression

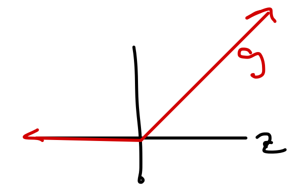
$$f(\underline{x}) = \beta_0 + \sum_{k=1}^{K} \beta_k A_k$$



Activation function $g(\cdot)$ is usually (these days)

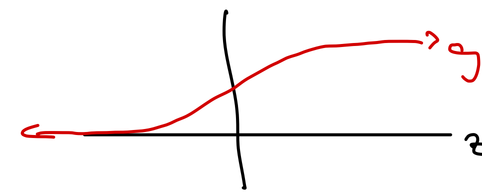
rectified linear unit (ReLU)

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



Historically the sigmoid was used

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



Important for g to be nonlinear, otherwise $g(z) = z$

$$f(x) = \beta_0 + \sum_{k=1}^{K} \beta_k \, g\left(w_{k0} + \sum_{j=1}^{p} w_{kj} x_j\right)$$

$$= \beta_0 + \sum_{k=1}^{K} \beta_k \left(w_{k0} + \sum_{j=1}^{p} w_{kj} x_j\right)$$

$$= \left(\beta_0 + \sum_{k=1}^{K} \beta_k w_{k0}\right) + \sum_{j=1}^{p} \left(\sum_{k=1}^{K} \beta_k w_{kj}\right) x_j$$

$$= \gamma_0 + \sum_{j=1}^{p} \gamma_j x_j$$

= regular multiple regression.

Depending on g we can get interactive effects from $\underline{X}$:

$p = 2 \quad K = 2 \quad g(z) = z^2$

$\beta_0 = 0 \quad \beta_1 = \frac{1}{4} \quad \beta_2 = -\frac{1}{4}$

$w_{10} = 0 \quad w_{11} = 1 \quad w_{12} = 1$

$w_{20} = 0 \quad w_{21} = 1 \quad w_{22} = -1$

$$f(\underline{x}) = \beta_0 + \sum_{k=1}^{2} \beta_k g\left(w_{k0} + \sum_{j=1}^{2} w_{kj} x_j \right)$$

$$= \beta_0 + \beta_1 \left(w_{10} + w_{11} x_1 + w_{12} x_2 \right)^2$$

$$+ \beta_2 \left(w_{20} + w_{21} x_1 + w_{22} x_2 \right)^2$$

$$= \frac{1}{4} (x_1 + x_2)^2 - \frac{1}{4} (x_1 - x_2)^2$$

$$= x_1 x_2 .$$

To estimate $\beta_0, \dots, w_{Kp}$ we could use

$$\sum_{i=1}^{n} \left(y_i - f(\underline{x}_i) \right)^2$$

For classification problems, suppose $Y \in \{1, \dots, M\}$,

model

$$P(Y=m \mid \underline{x}) = \frac{e^{z_m}}{\sum_{\ell=1}^{M} e^{z_\ell}}$$

using the <u>soft max function</u> where

$$z_m = f_m(\underline{x}) \qquad m = 1, \ldots, M$$

a neural network.

Estimation minimizes cross-entropy:

$$- \sum_{i=1}^{n} \sum_{m=1}^{M} Y_{im} \log(f_m(x_i)).$$

10.2 Multilayer + Deep Networks

A multiple layer network gives rise to <u>deep learning</u>.

single layer

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

$$A_k = g\left(w_{k0} + \sum_{j=1}^{p} w_{kj} x_j\right)$$

two layer

K_1 nodes in 1st layer, K_2 in 2nd layer

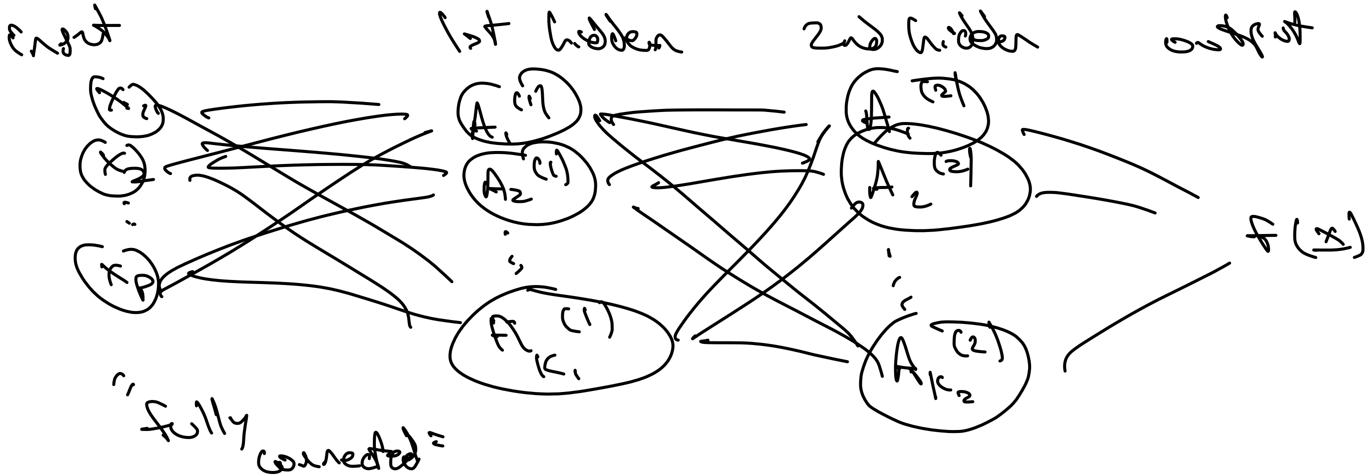
$$f(\underline{x}) = \beta_0 + \sum_{k=1}^{K_2} \beta_k A_k^{(2)}$$

$k=1,\dots,K_2 \rightarrow$ $$A_k^{(2)} = g\left(w_{k0}^{(2)} + \sum_{j=1}^{K_1} w_{kj}^{(2)} A_j^{(1)}\right)$$

not x

$$A_k^{(1)} = g\left(w_{k0}^{(1)} + \sum_{j=1}^{p} w_{kj}^{(1)} x_j\right)$$

$\leftarrow$ $k=1,\dots,K_1$



<u>Deep</u> network with L layers & ℓth layer having K_ℓ nodes

$$f(\underline{x}) = \beta_0 + \sum_{k=1}^{K_L} \beta_k A_k^{(L)}$$

$$A_k^{(L)} = g\left(w_{k0}^{(L)} + \sum_{j=1}^{K_{L-1}} w_{kj}^{(L)} A_j^{(L-1)}\right) \quad k=1,\dots,K_L$$

$$\vdots$$

$$A_k^{(2)} = g\left(w_{k0}^{(2)} + \sum_{j=1}^{K_1} w_{kj}^{(2)} A_j^{(1)}\right) \qquad k = 1, \ldots, K_2$$

$$A_k^{(1)} = g\left(w_{k0}^{(1)} + \sum_{j=1}^{P} w_{kj}^{(1)} x_j\right) \qquad k = 1, \ldots, K_1$$