7) ANN

Perceptron

- An independent building block (node) of ANN

Ex) (8) $w_1=0.5$, (5) $w_2=0.3$, (-4) $w_3=0.2$ → Σ (7.7) → sign → 1; Σ ↑ $w_0=3$; $x_0=1$ — bias term

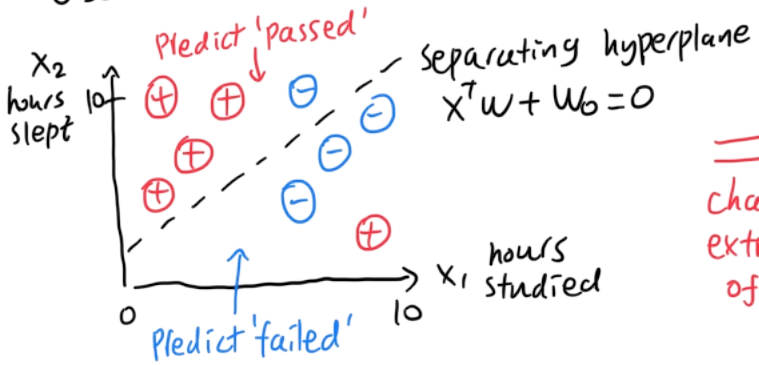
$\Rightarrow \hat{y} = \sigma(x^T w + w_0)$

output $\hat{y}$ | input x | weights w | bias (intercept) w_0

σ: nonlinearity (activation fn) | $x^T w$: Dot product (linear weighted sum)

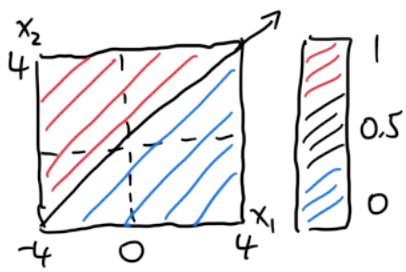
Forward Propagation

- It iteratively transforms feature space (Geometric representation of data inputs)
  - Use axes as features, coordinates as observations



σ changes extremities of values

slope & intercept stays unchanged



Backward Propagation

- 2 common Loss functions

① MSE (Mean-Squared Error) Loss: $L = \frac{1}{2}(\hat{y} - y)^2$ for numerical predictions

($\hat{y}$: predicted, y: reality)

② Cross-Entropy / Logistic Loss: $L = -\sum_{i=1}^{n} y_i \log(\hat{y}_i)$ for categorical predictions

- Train on individual loss function is not ideal bc:

i) Time consuming ii) Can cause overfitting

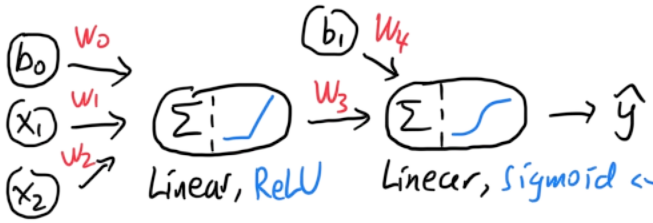
- So instead, optimize Cost function: $J = \frac{1}{m}\sum_{i=1}^{m} L(\hat{y}_i, y_i)$, m = # batch of samples
  - Goal: $w = \underset{w}{\operatorname{argmin}}\, J(w)$
  - Solution: backpropagation (gradient descent on w) → $w := w - \alpha \frac{\partial J}{\partial w}$ (lr: α)

ANN Single Layer

- Simple Regression: $y_i = \beta_0 + \sum \beta_i x_i + \varepsilon_i$, the linear part of perceptron ($x^T w$)

(y_i: ith data value; $\beta_0 + \sum \beta_i x_i$: predicted value; ε_i: ith error (residual))

- model:



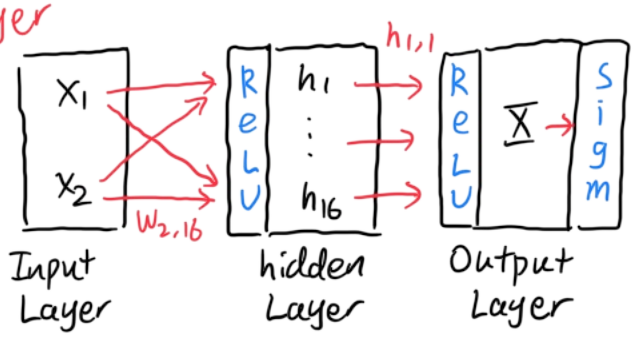
← include this if ur doing classification

- Training workflow: for i in epoch { forward prop → Compute & store loss → backprop }
  Make final forward prop → Calculate accuracy:

- Testing workflow: Report final accuracy → Show Losses vs. Epochs graph

ANN Multilayer

- architecture:

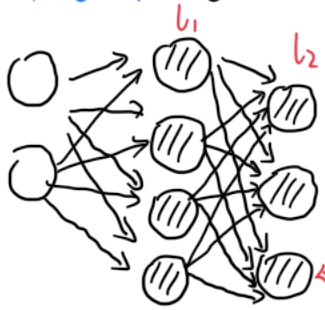


- b_i is bias in each layer that produces output to next layer (they don't take inputs!)
- W_{ij} (nn.Linear) comes from the transformation matrix into next layer

- U need nonlinearities in-b/t layers (else all layers will collapse into 1 layer)
- Use softmax over sigmoid for multi-classification: output is $\{P(y_1), \dots P(y_n)\}$

Dimension of DL model

- Breadth (# nodes in hidden layer) vs. Depth (# hidden layers) are the dimension

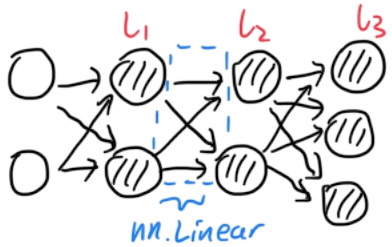


7 total units

27 weights

$(8 + 12 + 7)$ ← bias

L_1 L_2



7 total units

21 weights to learn

$(4 + 4 + 6 + 7)$ ← bias

L_1 L_2 L_3

Slower train time to learn more complex mappings