	True label	
Model Pred	TP	FP
	FN	TN

$$\text{Precision} = \frac{TP}{TP+FP}$$

$$\text{Recall} = TP/(TP+FN)$$

$$\text{Accuracy} = \frac{TP+TN}{TP+FP+TN+FN}$$

$$F_1 = \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

TP rate $= \frac{TP}{P} =$ Recall

FP rate $= FP/N$

Learning curve → identify how much data needed
- increase rapidly → get more data
- completely flat → more data can help
- Gradual increase → need a lot more data

Train Acc → decrease with more data

Validation Acc → increase with more data

if gap between train and val
- → don't close overfitting
- → close too fast underfitting

ROC Graph is TP rate (Y) vs FP rate (X), used to depict trade offs between benefits (TP) and costs (FP), Note goes through (0,0) (1,1)

Bias → inherent error

Variance → how "over specialized"

Regularized Loss: $J_\lambda(\theta) = J(\theta) + \lambda R(\theta)$
- $J(\theta)$ → training loss / cost function
- $R(\theta)$ → regularizer
- λ Regularization parameter

Linear Regression:

$$\min_\theta \frac{1}{2}\sum_{i=1}^{m}(y^i - \theta^T \cdot x^i)^2 + \frac{\lambda}{2}|\theta|^2$$

- $\lambda = 0$ → no regularization → maybe overfitting
- if λ too big → forcing all parameters to be close to 0

Sign unit
- 1 when >0
- -1 otherwise

$$\hat{y} = g(W_0 + W^T X)$$

$$X = \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix} \quad W = \begin{pmatrix} w_1 \\ \vdots \\ w_n \end{pmatrix}$$

$$\hat{y} = g\left(w_0 + \sum_{i=1}^{n} w_i \cdot x_i\right)$$

linear combo inputs
non activation linear function
w_0 is bias

Sigmoid function

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

$$g'(z) = \sigma(z)(1-\sigma(z))$$

Hyperbolic Tangent

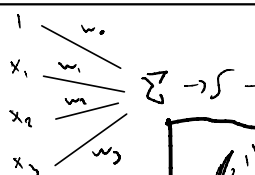
$$g(z) = \sigma(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$$

$$g'(z) = 1 - g(z)^2$$

Rectified Linear Unit (ReLU)

$$g(z) = \max\{0, z\}$$

$$g'(z) = \begin{cases} 1, & n > 0 \\ 0, & \text{otherwise} \end{cases}$$



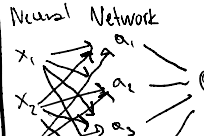
Gradient Rule

$$w^1 \leftarrow w^1 - \alpha dW^1$$

Forward Prop → z values

Backward Prop → derivative

Neural Network



Next layer index subscript

$$z_1 = \begin{pmatrix} z_1^1 \\ z_2^1 \\ z_3^1 \\ z_4^1 \end{pmatrix} = \begin{pmatrix} w_1^{1T} \\ w_2^{1T} \\ w_3^{1T} \\ w_4^{1T} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} + \begin{pmatrix} b_1^1 \\ b_2^1 \\ b_3^1 \\ b_4^1 \end{pmatrix}$$

W^1 $\quad$ b^1

Logistic Regression

$$L(\hat{y}, y) = -(y \log \hat{y} + (1-y)\log(1-\hat{y}))$$

- $y=1$ → want $-\log \hat{y}$ → want $\hat{y}$ large
- $y=0$ → want $\log(1-\hat{y})$ → want $\hat{y}$ small

$$z^1 = w_3^{1T} x + b_3^1, \quad a_3^1 = \sigma(z_3^1)$$

$$z_4^1 = w_4^{1T} x + b_4^1, \quad a_4^1 = \sigma(z_4^1)$$

$$z_1^1 = w_1^{1T} x + b_1^1, \quad a_1^1 = \sigma(z_1^1)$$

$$z_2^1 = w_2^{1T} x + b_2^1, \quad a_2^1 = \sigma(z_2^1)$$

Hyperparameters
- learning rate α
- # hidden layer L
- # hidden units n^1, n^2
- choice of activation functions

Reinforcement learning
Req input → Reward Func
State S → Action #
Return $= R_1 + \gamma R_2 + \gamma^2 R_3 + \ldots$

Discount Factor
γ: $0 < \gamma < 1$

$\sigma(z)$ is input for the NN second layer

Build ML System:
Start with idea, implement to code, experiment to test idea worked

Bellman equation

$$Q(S,a) = R(S) + \gamma \max_{a'}(s', a')$$

If NN underfit add layer
Overfit add data

Bellman max is max of → or ← input on state

x, w, b → $z = w^T x + b \rightarrow a = \sigma(z) \rightarrow L(a,y)$

$dw = dz \cdot \frac{dz}{dw} = dz \cdot x$

$db = dz \cdot \frac{dz}{db} = dz$

$da = \frac{dL(a,y)}{da} = -\frac{y}{a} + \frac{(1-y)}{1-a} = -\frac{y}{a} - \frac{(1-y)}{1-a}$

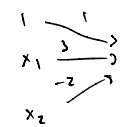
$dz = da \cdot \frac{da}{dz} = da \cdot \sigma'(z) = -\frac{y}{a} - \frac{(1-y)}{(1-a)} \times a(1-a) = a - y$

$\frac{dH}{dV} = 3$

$\frac{dH}{da} = \frac{dH}{dV} \cdot \frac{dV}{da} = 3$

$\frac{dH}{du} = \frac{dH}{dV} \cdot \frac{du}{du} = 3$

$\frac{dH}{db} = \frac{dH}{dV} \cdot \frac{dv}{db} = 3 \times 2 = 6$



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

$= g\left(1 + \begin{pmatrix}3\\-2\end{pmatrix}^T \begin{pmatrix}x_1\\x_2\end{pmatrix}\right)$

$= g(1 + 3x_1 - 2x_2)$

given input $\begin{pmatrix}-1\\2\end{pmatrix}$

$\hat{y} = g(1 + 3(-1) - 2(2))$

$= g(-6) = \frac{1}{1+e^{-(-6)}} = 0.002$

$a = 5$, $b = 3$, $c = 2$ → $u = bc$ → $v = a + u$ → $H = 3v$

$\frac{dH}{dc} = \frac{dH}{du} \cdot \frac{dv}{dc} = 3 \times 3 = 9$

m = # inputs pair or sight

Gradient Decent

Perceptron tree use values to scale branch then match then multiply running to node and add bias → the perceptron formula

$AUC = \frac{1}{2} \sum \frac{1}{2}(FPR_2 - FPR_1)(TPR_1 + TPR_2)$

$L = \frac{1}{m} \sum_{i=1}^{m} (o^i - y^i)^2$

$\min J(\theta_1, \theta_2)$ $\theta_1, \theta_2 \in \mathbb{R}$

$\theta_1 = \theta_1 - \alpha \frac{d}{d\theta_1}(J(\theta_1, \theta_2))$

$\theta_1 = \theta_2 - \alpha \frac{d}{d\theta_2}(J(\theta_1, \theta_2))$

$\frac{dL}{dw_{21}} = \frac{dL}{do_1} \times \frac{do_1}{dw_{21}} + \frac{dL}{do_2} \times \frac{do_2}{dw_{21}}$

$\leftarrow w_{21} \leftarrow w_{21} - \alpha \frac{dL}{dw_{21}}$

$\frac{dL}{do_1} = 2(o_1 - t_1)$

$\frac{do_1}{dw_{21}} = \frac{do_1}{dh_3} \times \frac{dh_3}{dh_1} \times \frac{dh_1}{dw_{21}}$

$= o_1(1-o_1) w_{31} \times 1 \times h_1(1-h_1) i_2$

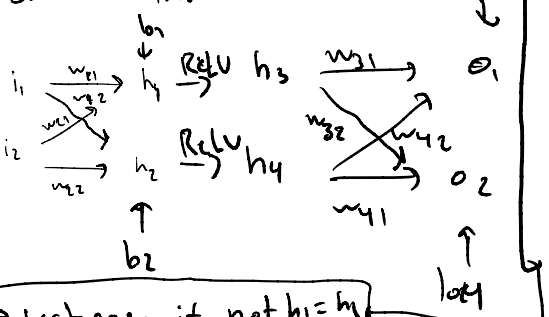
$\frac{dL}{do_2} = 2(o_2 - t_2)$

$\frac{do_2}{dw_{21}} = \frac{do_2}{dh_3} \times \frac{dh_3}{dh_1} \times \frac{dh_1}{dw_{21}}$

$= o_2(1-o_2) w_{32} \times 1 \times h_1(1-h_1) i_2$

t_i = target = y_i

sum of these is dL/dw_{21}



if diff $h_3 \neq h_1$ → test case if not $h_1 = h_3$

$\frac{dh_3}{dh_1} = \frac{dh_3}{dz_3} \cdot \frac{dz_3}{dh_1}$ → $h_3 \cdot (1-h_3) \cdot w_{31}$

Plug into Activation derivative

$\frac{dh_3}{dz_3} = h_3 \cdot (1-h_3)$

$z_3 = w_{31} h_1 + w_{32} h_2 + b_3$

$\frac{dz_3}{dh_1} = w_{31}$

in gradient decent if weights set zero only need to compute one set of hidden units there semetric.