__/__/____

⑦ Linear Classification (Linear Transformation)

Goal: Is to classify data points into classes using a linear boundary. We predict a class label.

$$\left(f(x) = w^T x + b\right) : \hat{y} = \begin{cases} 1 & w^T x + b > 0 \\ 0, & \text{otherwise} \end{cases}$$

(Decision boundary → $w^T x + b = 0$) → it seperates classes.

Linear classifier gives hard class outputs [0/1] not probability.

Fix

Predicts boundary directly

⑧ Logistic Regression (Discriminative Model) $P(y|x)$

↳ we map linear output → probability range [0, 1] using sigmoid function

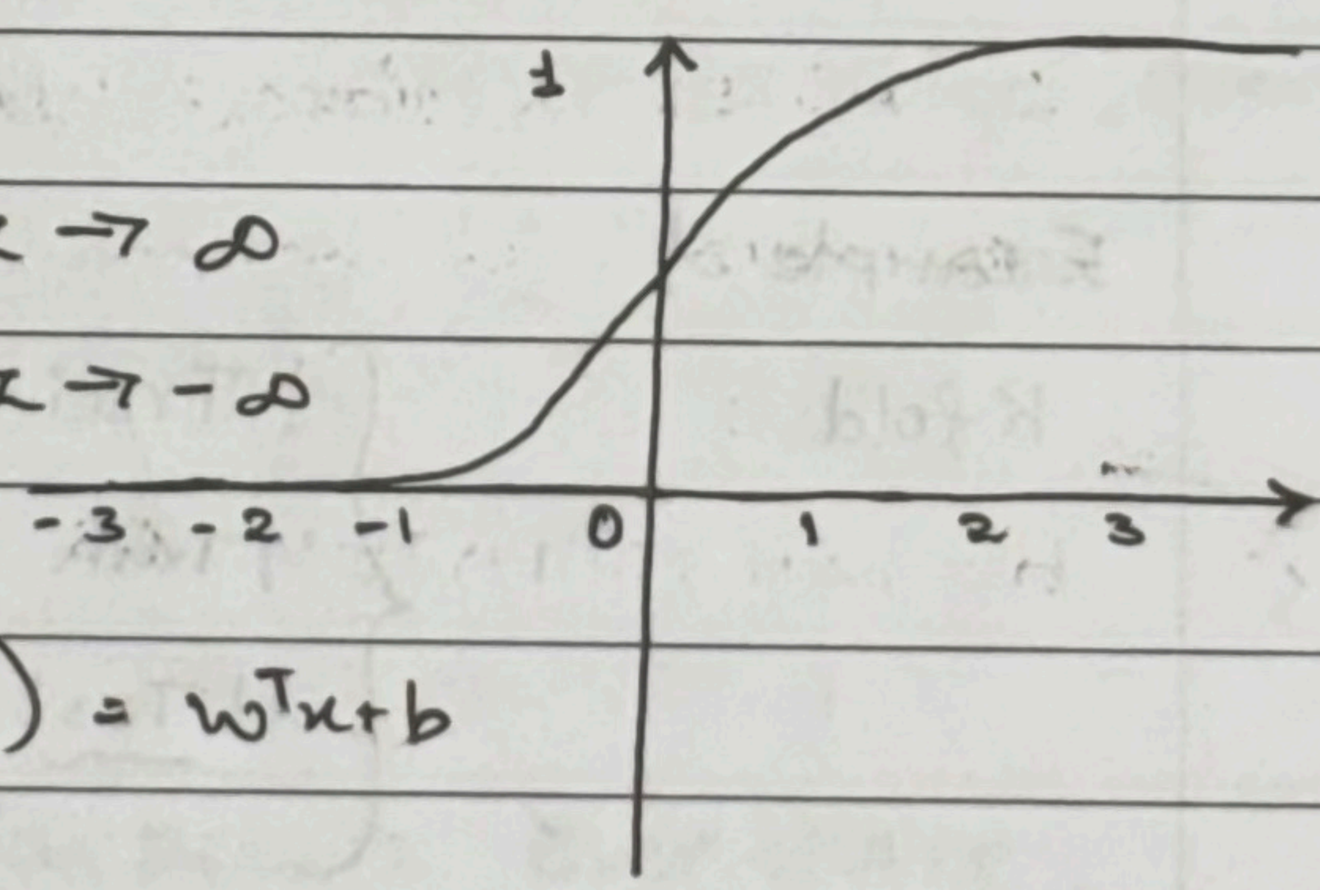
(Large Data)

(Convex)

$$\begin{cases} P(y=1 \mid x) = \sigma(z) = \dfrac{1}{1+e^{-z}}, \quad z = w^T x + b \\ P(y=0 \mid x) = 1 - P(y=1 \mid x) \end{cases}$$

$\sigma(z)$ tends → 1 as $z \to \infty$

$\sigma(z)$ tends → 0 as $z \to -\infty$



→ Log odds (Logit): $\log(P/1-P) = w^T x + b$

1. Each prediction is a Bernoulli random variable with success probability $\bar{p}_i$

2. We want to find 'w' and 'b' that maximize probability of observing data.

Likelihood: $L(w) = \prod_{i=1}^{n} P(y_i \mid x_i ; w)$

Minimizing Loss $\underline{\underline{\propto}}$ Maximizing Likelihood

__/__/____

$\therefore L(w) = \prod_{i=1}^{n} (\hat{p_i})^{y_i} (1-\hat{p_i})^{(1-y_i)}$

3. For numerical stability, we take log:

$l(w) = \log L(w) = \sum_{i=1}^{n} \left[y_i \log(\hat{p_i}) + (1-y_i) \log(1-\hat{p_i}) \right]$

This is the log likelihood function.

4. We want to maximize likelihood → maximize log-likelihood

In optimization, we minize a cost, we take negative log likelihood

$J(w) = -l(w) = -\sum_{i=1}^{n} \left[y_i \log(\hat{p_i}) + (1-y_i) \log(1-\hat{p_i}) \right]$

This is log loss / Binary Cross Entropy Loss

5. Gradient Descent : $w = w - \alpha \, \partial J/\partial w$, $b = b - \alpha \, \partial J/\partial b$

predicts how each class generates data

(9) Linear Discriminant Analysis (Generative Model)

$P(x|y)$ & $P(y)$

(small Data) LDA: Classification, Dimensionality Reduction

It finds a linear combination of features that best seperates two or more classes.

★ Assumptions of LDA: 1. Gaussian / Normal Distribution

2. Same Covariance Matrices

3. Linear Separability.

Goal is to find a projection vector w,

such that seperation $= \dfrac{\text{Between class variance}}{\text{Within class variance}}$ is maximised

Fisher Criterion, $J(w)$: $S_b / S_w \approx \uparrow$

measures seperability of classes along projection w.

μ_0, μ_1 : mean vectors for class 0 and 1

Σ_0, Σ_1 : covariance matrices

S_W : $\Sigma_0 + \Sigma_1$: within clas scatter matrix

S_B : $(\mu_1 - \mu_0)(\mu_1 - \mu_0)^T$: between class scatter matrix

we find $w^* = S_W^{-1}(\mu_1 - \mu_0) \rightarrow \frac{\mu_1 - \mu_0^2}{S_W}$

The discriminant function for class k,

$$\delta_k(x) = x^T \Sigma^{-1} \mu_k - \frac{1}{2} \mu_k^T \Sigma^{-1} \mu_k + \log(\pi_k)$$

π_k: prior probability of class k.

Predict class = $\hat{y} = \underset{k}{\text{argmax}}\ \delta_k(x)$

→ Finds (k-1) new axes for k classes.

↳ Linear combinations of original features.

(10) Seperating Hyperplane ~ Perceptron (Historical- Base of NN)
(Discriminative Model)

Goal: Finding a linear decision boundary (hyperplane) that divides data points belonging to different classes.

1. Initialise weights and bias
2. Iterate through each training sample:
   - Predict $\hat{y}$
   - If $\hat{y} \neq y$: update weights

     $w: w - \alpha\, n y_i x_i$

     $b: b - \alpha\, n y_i$
3. Repeat until convergance.

→ Outputs a hard classification

→ Finds some hyperplane that seperates classes.