ML Techniq

WEEK 1

→ Broad Paradigms

↳ Supervised learning: data, features, and labelling. Classification, regression, ranking

Unsupervised learning: only datapoints. Clustering, Representational learning

Sequential learning: make a decision → get feedback → make decision … Reinforcement learning

→ Representation Learning

↳ Sub-category of Unsupervised learning

↳ Goal: given a set of datapoints, understand something useful about them

↳ Problem:

Input: $\{x_1, x_2, \dots, x_n\}$ $\quad x_i \in \mathbb{R}^{d \leftarrow \text{features}}$

Output: some "compressed" representation of the dataset

↳ Example:

Data: $\left\{\begin{bmatrix}-7\\-14\end{bmatrix}, \begin{bmatrix}2.5\\5\end{bmatrix}, \begin{bmatrix}0.5\\1\end{bmatrix}, \begin{bmatrix}0\\0\end{bmatrix}\right\}$

Representative: $\begin{bmatrix}1\\2\end{bmatrix}$ $\qquad$ Coefficients: $\{-7, 2.5, 0.5, 0\}$

↳ Example.

Data: $\left\{\begin{bmatrix}-5\\-5\end{bmatrix}, \begin{bmatrix}1\\1\end{bmatrix}, \begin{bmatrix}7\\7\end{bmatrix}, \begin{bmatrix}10\\10\end{bmatrix}, \begin{bmatrix}5\\6\end{bmatrix}\right\}$

Rep: $\left\{\begin{bmatrix}1\\0\end{bmatrix}, \begin{bmatrix}0\\1\end{bmatrix}\right\}$ $\quad$ Co-eff: $\{(-5,-5), (1,1), (7,7), (10,10), (5,6)\}$

The only way to compress: project $\begin{bmatrix}5\\6\end{bmatrix}$ onto the line $y = x$.

↳ Goal: Find the line that has the least "reconstruction" error.

Dataset: $\{x_1, x_2, \dots, x_n\}$ $\quad x_i \in \mathbb{R}^d$

$$\text{ERROR}(\text{line}, \text{dataset}) = \sum_{i=1}^{n} \text{error}(\text{line}, x_i) \quad \text{(represented using } w \text{ as unit vector)}$$

$$= \sum_{i=1}^{n} \text{length}^2\left(x - (x^T w)w\right)$$

$$f(w) = \frac{1}{n}\sum_{i=1}^{n} \|x - (x_i^T w)\cdot w\|^2$$

$$= \frac{1}{n}\sum_{i=1}^{n} (x - (x_i^T w)\cdot w)^T (x - (x_i^T w)\cdot w)$$

$$= \frac{1}{n}\sum_{i=1}^{n} x_i^T x_i - (x_i^T w)^2 - (x_i^T w)^2 + (x_i^T w)^2$$

$$= \frac{1}{n}\sum_{i=1}^{n} x_i^T x_i - (x_i^T w)^2$$

→ $x_i^T x_i$ is a constant. So instead of minimizing $f(w)$, minimise $g(w)$

$$g(w) = \frac{1}{n}\sum_{i=1}^{n} -(x_i^T w)^2 \longrightarrow \text{maximise } g(w)$$

$$\max_w \; g(w) = \frac{1}{n}\sum (w^T x_i)(x_i^T w) = \frac{1}{n}\sum w^T (x_i x_i^T) w$$

$$= w^T \underbrace{\left(\frac{1}{n}\sum_{i=1}^{n} x_i x_i^T\right)}_{C \,\to\, \text{cov matrix}} w$$

equivalently: $\max_{w:\, \|w\|^2 = 1} w^T C w$; where $\underbrace{C = \frac{1}{n}\sum x_i x_i^T}_{\text{Covariance matrix}}$

↳ The residue data (error from reconstruction) might also have information instead of just being error

Possible algorithm to deal with this:

Input: $\{x_1, x_2, \dots, x_n\}$ $x_i \in \mathbb{R}^d$

→ De-mean the data: $x_i = x_i - \mu \ \forall i$

→ Find the best line $w_1 \in \mathbb{R}^d$

→ Replace x_i with $x_i - (x_i^T w_1) w_1$

→ Repeat to obtain w_2

↳ By continuing this procedure, we get

$\{\underbrace{w_1, w_2, \dots, w_d}_{\text{orthonormal vectors}}\}$ s.t. $\|w_k\|^2 = 1 \ \forall k$ and $w_i^T w_j = 0 \ \forall i \neq j$

↳ Residues after 2 rounds: $\{x_i - (x_i^T w_1) w_1 - (x_i^T w_2) w_2, \dots\}$ ← looks exactly like the gram-schmidt process

" " d rounds: $\forall i \quad x_i - [(x_i^T w_1) w_1 + \dots + (x_i^T w_d) w_d]$

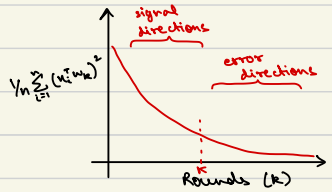
↳ Q. What if data "approximately" in low-dimensional space?

For any $w \in \mathbb{R}^d$ s.t. $\|w\|^2 = 1$

$\forall i \quad \|x_i\|^2 = \|x_i - (x_i^T w) w\|^2 + \|(x_i^T w) w\|^2$

$\frac{1}{n} \sum_{i=1}^n \|x_i\|^2 = \frac{1}{n} \sum_{i=1}^n \|x_i - (x_i^T w) w\|^2 + \underbrace{\frac{1}{n} \sum_{i=1}^n \|(x_i^T w) w\|^2}$

The larger the value of this term, the better the fit.



WEEK 2

→ Issues with PCA

① Time complexity of finding the eigenvalues and eigenvectors
↳ typically $O(d^3)$

② Only looks at linear relationships between features.

→ Time-complexity issue with PCA

↳ large d $[d \gg n]$, where $d \to$ # features $n \to$ # datapoints

↳ $X = \begin{bmatrix} | & | & & | \\ x_1 & x_2 & \cdots & x_n \\ | & | & & | \end{bmatrix}$ $X \in \mathbb{R}^{d \times n}$ $C = \frac{1}{n} \sum_{i=1}^{n} x_i x_i^T$

Covariance matrix $= \frac{1}{n} XX^T$

↳ Let w_k be the k^{th} largest eigenvalue of C, corresponding to λ_k

$C w_k = \lambda_k w_k$

$\left(\frac{1}{n} \Sigma x_i x^T\right) w_k = \lambda_k w_k$

$w_k = \frac{1}{n \lambda_k} \sum_{i=1}^{n} (x_i^T w_k) \cdot x_i$

↳ w_k is a linear combination of data points.

$w_k = X \alpha_k$ for some $\alpha_k \in \mathbb{R}^n$ $\quad \begin{bmatrix} | & | & & | \\ x_1 & x_2 & \cdots & x_n \\ | & | & & | \end{bmatrix} \begin{bmatrix} \alpha_{k_1} \\ \alpha_{k_2} \\ \vdots \\ \alpha_{k_n} \end{bmatrix}$

$d \times n$ $\quad n \times 1$ $= d \times 1$

$(X^T X) \alpha_k = (n \lambda_k) \alpha_k$ — ①

$1 = \alpha_k^T (X^T X) \alpha_k$ — ② Any vector that satisfies ① & ② is the α_k

↳ **Algorithm**

Input: $D = \{x_1, \ldots, x_n\}$ $x_i \in \mathbb{R}^d$ $[d \gg n]$

Steps:

① Compute $K = X^T X$ $K \in \mathbb{R}^{n \times n}$

② Compute eigendecomposition of K
↳ eigenvectors $\{\beta_1, \ldots, \beta_n\}$ corresponding to eigenvalues $\{n\lambda_1, \ldots, n\lambda_n\}$ $\longrightarrow O(n^3)$

③ Set $\alpha_k = \frac{\beta_k}{\sqrt{n \lambda_k}}$ $\forall k$

④ $w_k = X \alpha_k$ $\forall k$

→ Non-linear relationships

↳ Transform features from low dimension $\mathbb{R}^d$ to high dimension $\mathbb{R}^D$

$x \in \mathbb{R}^2 \xrightarrow{\phi} \phi(x) \in \mathbb{R}^D$ → $\phi(x)^T \phi(x)$ for eigen decomposition

↳ Issue : there might be too many transformations if the power of features is high.
In general : d features, p^{th} power → $\sum_{i=0}^{p} {}^{d}C_i \simeq O(d^p)$

→ Kernel Functions

↳ $\phi(x_i)$ $\quad$ $\phi(x_j) \rightarrow \square \rightarrow \phi(x_i)^T \phi(x_j) = K_{ij}$

x_i $\quad$ x_j $\quad$??

↳ Example :

$x = [f_1, f_2]$ $\quad$ $x' = [g_1, g_2]$

Consider the function : $(x^T x' + 1)^2 = (f_1 g_1 + f_2 g_2 + 1)^2$

$$= \begin{bmatrix} f_1^2 & f_2^2 & 1 & \sqrt{2} f_1 f_2 & \sqrt{2} f_1 & \sqrt{2} f_2 \end{bmatrix} \begin{bmatrix} g_1^2 \\ g_2^2 \\ 1 \\ \sqrt{2} g_1 g_2 \\ \sqrt{2} g_1 \\ \sqrt{2} g_2 \end{bmatrix}$$

$$= \phi(x)^T \phi(x')$$

where, $\phi\left(\begin{bmatrix} a \\ b \end{bmatrix}\right) = \begin{bmatrix} a^2 \\ b^2 \\ 1 \\ \sqrt{2} ab \\ \sqrt{2} a \\ \sqrt{2} b \end{bmatrix}$

Note : $(x^T x' + 1)^2$ computes the dot-product in a "transformed space".

↳ $k(x, x') = (x^T x' + 1)^p$ for some $p \geq 1$ $\quad x \in \mathbb{R}^d$

→ can be shown to be a valid Function

i.e., $\exists\ \phi : \mathbb{R}^d \rightarrow \mathbb{R}^D$ s.t $k(x, x') = \phi(x)^T \phi(x')$

↳ $k(x, x') = \exp\left(\frac{-\|x - x'\|^2}{2\sigma^2}\right)$ For some $\sigma > 0$

↑ Radial Basis Function

→ can be shown to be valid

→ ϕ in this case maps x to an infinite dimensional space.

→ think of mapping a datapoint to a "function"
and dot-products b/w functions become integrals.

↳ Kernel Functions : Any function $k : \mathbb{R}^d \times \mathbb{R}^d \rightarrow \mathbb{R}$ which is a "valid" map

↳ Given a function, how to determine if it's a valid kernel ?

Method ① : Exhibit a map ϕ explicitly. Hard sometimes.

Method ② : Mercer's Theorem

↳ $k : \mathbb{R}^d \times \mathbb{R}^d \rightarrow \mathbb{R}$ is a valid kernel iff :

① k is symmetric i.e., $k(x, x') = k(x', x)$

② For any dataset $\{x_1, \ldots, x_n\}$, the matrix $K \in \mathbb{R}^{n \times n}$ where

$K_{ij} = k(x_i, x_j)$ is Positive Semi-Definite

↳ All eigenvalues ≥ 0

→ Kernel PCA

↳ Algorithm:

Input: $\{x_1, \dots, x_n\}$ $x_i \in \mathbb{R}^d$, kernel $k: \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$

Steps:

① Compute $K \in \mathbb{R}^{n \times n}$, where $k_{ij} = k(x_i, x_j)$ $\forall i, j$

② Compute $\beta_1, \dots, \beta_k$ eigenvectors and $n\lambda_1 \geq \dots \geq n\lambda_k$ eigenvalues of K and normalise to get $\alpha_u = \frac{\beta_u}{\sqrt{n\lambda_u}}$

③ $w_k = \phi(X)\alpha_k$ ⟶ defeats the purpose of kernel because it needs $\phi(x)$

↳ shouldn't compute w_k ⇒ we cannot reconstruct eigenvectors of co-variance mat.

↳ but we can compute the "compressed" representation

$$\phi(x_i)^T w_k = \phi(x_i)^T \left(\sum_{j=1}^n \phi(x_j)\alpha_{kj}\right)$$

$$= \sum_{j=1}^n \alpha_{kj}\, \phi(x_i)^T \phi(x_j) = \sum_{j=1}^n \alpha_{kj} K_{ij}$$

modified ③ Compute $\sum_{j=1}^n \alpha_{kj} K_{ij}$ $\forall k$

$$x_i \in \mathbb{R}^k \longrightarrow \left[\sum_{j=1}^n \alpha_{1j} K_{ij}, \sum_{j=1}^n \alpha_{2j} K_{ij}, \dots, \sum_{j=1}^n \alpha_{kj} K_{ij}\right]$$

WEEK 3

→ Clustering

↳ Goal: $\{x_1, \dots, x_n\}$ $x_i \in \mathbb{R}^d$

→ Partition data into k different clusters

↳ datapoints: $x_1, x_2, \dots, x_n$

cluster indicator: $z_1, z_2, \dots, z_n$ where $z_i \in \{1, \dots, k\}$

↳ Performance measurement of clustering:

$$F(z_1, \dots, z_n) = \sum_{i=1}^n \|x_i - \mu_{z_i}\|_2^2$$, where μ_{z_i} = mean/average of z_i cluster

$$\mu_k = \frac{\sum_{i=1}^n x_i \mathbb{1}(z_i = k)}{\sum_{i=1}^n \mathbb{1}(z_i = k)}$$ where $\mathbb{1}(u) = \begin{cases} 1 \text{ if true} \\ 0 \text{ otherwise} \end{cases}$

↳ Goal: $\min_{\{z_1, \dots, z_k\}} \sum_{i=1}^n \|x_i - \mu_{z_i}\|^2$

→ Lloyd's Algorithm / K-means clustering

↳ Initialization

$z_1^0, z_2^0, \dots, z_n^0 \in \{1, \dots, k\}$ (iteration $*$) each datapoint has been assigned to a cluster

→ Algorithm runs until convergence

① Compute means

↳ $\forall k \quad \mu_k^t = \frac{\sum_{i=1}^n x_i \mathbb{1}(z_i^t = k)}{\sum_{i=1}^n \mathbb{1}(z_i^t = k)}$

② Reassignment

↳ $\forall i \quad z_i^{t+1} = \arg\min_k \|x_i - \mu_k^t\|_2^2$

↳ Convergence

→ Let $x_1, x_2, \dots, x_\ell \in \mathbb{R}^d$

$v^* = \arg\min_{v \in \mathbb{R}^d} \sum_{i=1}^\ell \|x_i - v\|^2 \longrightarrow v^* = \frac{1}{\ell} \sum_{i=1}^\ell x_i$

→ Because every reassignment reduces the aggregate distance from the mean of each cluster and its datapoints, every iteration reduces the objective function.
Because there are a finite number of clusters, the algorithm has to converge.

↳ Nature of clusters

→ Let's say Lloyd's algorithm clusters data points into 2 clusters with mean μ_1 and μ_2

↳ For points in cluster 1 : $\|x - \mu_1\|^2 \leq \|x - \mu_2\|^2$

$= \|x\|^2 + \|\mu_1\|^2 - 2x^T\mu_1 \leq \|x\|^2 + \|\mu_2\|^2 - 2x^T\mu_2$

$= x^T(\mu_2 - \mu_1) \leq \frac{\|\mu_1\|^2 - \|\mu_2\|^2}{2}$

→ Cluster regions are intersection of half spaces.
↳ Voronoi Regions

↳ Initialization

→ One method is to pick k-means uniformly at random and then run the algorithm several times.

→ K-means++ algorithm

① choose first mean μ_1^0 uniformly at random

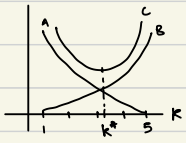
② For $l = 2, \dots, k$: choose μ_l^0 probabilistically proportional to score. $S(x) = \min_{j=1,\dots,l} \|x - \mu_j^0\|^2 \quad \forall x$

↳ Scores are given to each datapoint by looking at how far is the respective datapoint to the closest mean (from all the means that have already been picked).

↳ Choice of K

→ Find K that has the smallest " objective function + penalty(K) "

objective function = $\sum_{i=1}^{n} \|x_i - \mu_{z_i}\|^2$



A → objective function
B → penalty function
C → A + B

→ Criterion for picking the penalty function

↳ A.I.C: Akaike Information Criterion $[2K - 2\log(L(\theta^*))]$

↳ B.I.C: Bayesian Information Criterion $[K\log(n) - 2\log(L(\theta^*))]$

WEEK 4

Unsupervised learning

→ Rep. learning - PCA / Kernel

→ Clustering - Lloyd's / K-means

→ Estimation

↳ Max Likelihood

↳ Bayesian Modeling

→ Maximum Likelihood Estimation

↳ Likelihood function:

$$L(p, \{x_1, x_2, \dots, x_n\}) = P(x_1, x_2, \dots, x_n ; p) \quad (p: \text{underlying parameter})$$
$$= \prod_{i=1}^{n} P(x_i; p)$$

$$\hat{p}_{ML} = \arg\max_p \prod_{i=1}^{n} P(x_i; p) = \arg\max_p \log\left(\prod_{i=1}^{n} P(x_i; p)\right)$$

→ derivative of above function then set it to 0.

↳ assuming Gaussian distribution with μ and σ^2:

$$L(\mu, \sigma^2, \{x_1, x_2, \dots, x_n\}) = \prod_{i=1}^{n} \left[\frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(\frac{-(x_i-\mu)^2}{2\sigma^2}\right)\right]$$

$$\hat{\mu}_{ML} = \arg\max_{\mu} \sum_{i=1}^{n} -(x_i - \mu)^2$$

→ Bayesian Estimation

↳ Think of the parameter to estimate as a random variable.

↳ Punch → codified using a probability distribution over θ → $P(\theta)$ ← Prior Dist.

Update → " " " " " → $P(\theta|Data)$ ← Posterior Dist.

↳ Bayes' Theorem

$$P(A|B) = \frac{P(B|A) \cdot P(A)}{P(B)}$$

$A \rightarrow$ Parameter θ

$B \rightarrow$ Data $\{x_1, x_2, \dots, x_n\}$

$$\underbrace{P(\theta|\{x_1, x_2, \dots, x_n\})}_{\text{Posterior}} = \frac{\overbrace{P(\{x_1, \dots, x_n\}|\theta)}^{\text{Likelihood}}}{\underbrace{P(\{x_1, \dots, x_n\})}_{\text{Doesn't depend on } \theta}} \cdot \underbrace{P(\theta)}_{\text{Prior}}$$

↳ Beta prior

$$f(p; \alpha, \beta) = (p)^{\alpha-1} (1-p)^{\beta-1}$$

↳ $P(\theta|Data) \propto P(Data|\theta) \cdot P(\theta)$

→ Gaussian Mixture Models

↳ Steps to model a generative distribution with multiple gaussians:

① Pick which mixture a data point comes from

② Generate a data point from that mixture

→ Step 1 : Generate a mixture component among $\{1, \dots, k\}$

$z_i \in \{1, \dots, k\}$ → z_i indicates which mixture i-th datapoint comes from

$P(z_i = l) = \pi_l$, where $\sum_{i=1}^{k} \pi_i = 1$ and $0 \leq \pi_i \leq 1 \ \forall i$

→ Step 2 : Generate $x_i \sim N(\mu_{z_i}, \sigma_{z_i}^2)$

↳ Parameters to be estimated:

→ $\pi = [\pi_1, \pi_2, \dots, \pi_k]$

→ $\forall k \ (\mu_k, \sigma_k^2)$

Total = $2k + k - 1 = 3k - 1$

→ Maximum Likelihood of GMM

↳ $L\left(\begin{matrix}\mu_1,\dots,\mu_K,\\ \sigma_1^2,\dots,\sigma_K^2,\\ \pi_1,\dots,\pi_K\end{matrix}\ x_1,\dots,x_n\right) = \prod_{i=1}^{n} f_{mix}\left(x_i;\begin{matrix}\mu_1,\dots,\mu_K,\\ \sigma_1^2,\dots,\sigma_K^2,\\ \pi_1,\dots,\pi_K\end{matrix}\right)$

$= \prod_{i=1}^{n}\left[\sum_{k=1}^{K}\pi_k \cdot f(x_i;\mu_k,\sigma_k^2)\right]$

↳ Not possible to solve $\log L(\theta)$ analytically.

↳ For convex functions: $f(\lambda a + (1-\lambda)b) \le \lambda f(a) + (1-\lambda)f(b)$, where $\lambda \in [0,1]$

For concave functions: $f(\lambda a + (1-\lambda)b) \ge \lambda f(a) + (1-\lambda)f(b)$

Jensen's Inequality: $f\left(\sum_{k=1}^{K}\lambda_k a_k\right) \ge \sum_{k=1}^{K}\lambda_k f(a_k)$

↳ Log of GMM

→ $\log L(\theta) = \sum_{i=1}^{n}\log\left(\sum_{k=1}^{K}\pi_k \exp\left(\frac{-(x_i-\mu_k)^2}{2\sigma_k^2}\right)\frac{1}{\sqrt{2\pi}\sigma_k}\right)$

Log is a concave function

↳ Introduce for every datapoint i, the parameters

$\{\lambda_1^i, \lambda_2^i, \dots, \lambda_K^i\}$ s.t. $\sum_{k=1}^{K}\lambda_k^i = 1$, $0 \le \lambda_k^i \le 1 \ \forall i,k$

$\log L(\theta) = \sum_{i=1}^{n}\log\left(\sum_{k=1}^{K}\lambda_k^i\left(\frac{\pi_k \exp\left(\frac{-(x_i-\mu_k)^2}{2\sigma_k^2}\right)\frac{1}{\sqrt{2\pi}\sigma_k}}{\lambda_k^i}\right)\right)$

By Jensen's Inequality: $\log L(\theta) \ge$ modified_log $L(\theta,\lambda)$

modified_log $L(\theta,\lambda) = \sum_{i=1}^{n}\sum_{k=1}^{K}\lambda_k^i \log\left(\frac{\pi_k \exp\left(\frac{-(x_i-\mu_k)^2}{2\sigma_k^2}\right)\frac{1}{\sqrt{2\pi}\sigma_k}}{\lambda_k^i}\right)$

Log of sum is greater than sum of log because log is a concave function.

gives a lower bound for the true log likelihood at θ for any choice of λ , where $\theta \to \begin{Bmatrix}\mu_1,\dots,\mu_K\\ \sigma_1^2,\dots,\sigma_K^2\\ \pi_1,\dots,\pi_K\end{Bmatrix}$; $\lambda \to \begin{Bmatrix}\lambda_1^1,\dots,\lambda_K^1\\ \lambda_1^2,\dots,\lambda_K^2\\ \lambda_1^n,\dots,\lambda_K^n\end{Bmatrix}$

→ fixing λ will allow us to maximise w.r.t θ and vice versa.

→ Fix λ and maximise over θ

$\max_\theta$ modified_log $L(\theta,\lambda)$

$= \max_\theta \sum_{i=1}^{n}\sum_{k=1}^{K}\left[\lambda_k^i \log \pi_k - \lambda_k^i \frac{(x_i-\mu_k)^2}{2\sigma_k^2} - \lambda_k^i \log\sqrt{2\pi}\sigma_k\right]$

↳ Take derivative w.r.t. μ, σ to get:

$\hat{\mu}_k^{MML} = \frac{\sum_{i=1}^{n}\lambda_k^i x_i}{\sum_{i=1}^{n}\lambda_k^i}$; $\hat{\sigma}_k^{2\,MML} = \frac{\sum_{i=1}^{n}\lambda_k^i (x_i-\hat{\mu}_k^{MML})^2}{\sum_{i=1}^{n}\lambda_k^i}$; $\hat{\pi}_k^{MML} = \frac{\sum_{i=1}^{n}\lambda_k^i}{n}$

→ can think of λ_k^i as the probability i-th datapoint belongs to the k-th "cluster". Then, $\hat{\mu}_k^{MML}$ and $\hat{\sigma}_k^{2\,MML}$ are just sample mean and variance for k-th "cluster"

→ can find $\pi_k \ \forall k$ using lagrangian multiplier method.

→ Fix θ and maximise over λ

$\hat{\lambda}_k^{i\,MML} = \frac{\left(\frac{1}{\sqrt{2\pi}\sigma_k}\exp\left(\frac{-(x_i-\mu_k)^2}{2\sigma_k^2}\right)\right)\cdot \pi_k}{\sum_{l=1}^{K}\left(\frac{1}{\sqrt{2\pi}\sigma_l}\exp\left(\frac{-(x_i-\mu_k)^2}{2\sigma_k^2}\right)\cdot \pi_l\right)}$

→ EM Algorithm

↳ ① Initialise $\theta^0 = \{\mu_1^0, \dots, \mu_n^0, \sigma_1^0, \dots, \sigma_n^0, \pi_1^0, \dots, \pi_n^0\}$

② Until convergence $(\|\theta^{t+1} - \theta^t\| \leq \epsilon)$ (ϵ: tolerance parameter)

$\lambda^{t+1} = \arg\max_{\lambda}$ modified $\log L(\theta^t, \lambda)$ ⟶ Expectation step

$\theta^{t+1} = \arg\max_{\theta}$ modified $\log L(\theta, \lambda^{t+1})$ ⟶ Maximisation step

↳ EM produces "soft clustering", whereas Lloyd's algorithm produces hard clustering.

↳ EM takes variances into account, which becomes covariances in higher dimensions.

WEEK 5

→ Supervised learning

↳ Input: $\{x_1, \dots, x_n\}$ $x_i \in \mathbb{R}^d$ ← Features / Attributes

$\{y_1, \dots, y_n\}$ ← Labels

↳ $\{+1, -1\}$, $\{0, 1\}$ → Binary classification

↳ $\{0, 1, \dots, k\}$ → Multiclass classification

↳ $\in \mathbb{R}$ → Regression

→ Regression

↳ Input / Training data: $\{x_1, \dots, x_n\}$ $x_i \in \mathbb{R}^d$; $\{y_1, \dots, y_n\}$ $y_i \in \mathbb{R}$

Goal: Learn $h: \mathbb{R}^d \to \mathbb{R}$

→ Measure 'goodness' of a function h: $\text{error}(h) = \sum_{i=1}^{n} (h(x_i) - y_i)^2$

↳ Error (h) can be as small as 0. The h that achieves error $(h) = 0$ is s.t. $h(x_i) = y_i \ \forall i$

↳ Assuming h is linear: $h_w: \mathbb{R}^d \to \mathbb{R}$ s.t. $h_w(x) = w^T x$

Goal:

$$\min_{h_w} \sum_{i=1}^{n} (h_w(x_i) - y_i)^2 \longrightarrow \min_{w \in \mathbb{R}^d} \sum_{i=1}^{n} (w^T x_i - y_i)^2$$

$$\begin{bmatrix} — & x_1^T & — \\ — & x_2^T & — \\ & \vdots & \\ — & x_n^T & — \end{bmatrix}_{n \times d} \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_d \end{bmatrix}_{d \times 1} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}_{n \times 1}$$

$$= \min_{w \in \mathbb{R}^d} \|X^T w - y\|_2^2$$

↳ $\min_{w \in \mathbb{R}^d} (X^T w - y)^T (X^T w - y)$ ← unconstrained optimisation

↳ Solution: set gradient to zero

$$f(w) = (X^T w - y)^T (X^T w - y)$$

$$\nabla f(w) = 2(XX^T)w - 2(Xy)$$

$$(XX^T) w^* = Xy$$

$$w^* = (XX^T)^\dagger (Xy)$$

↳ $\dagger$ = pseudo matrix. Like a covariance matrix.

Depends on a covariance-like matrix, like PCA.

$$(AB)^T = (B^T A^T)$$

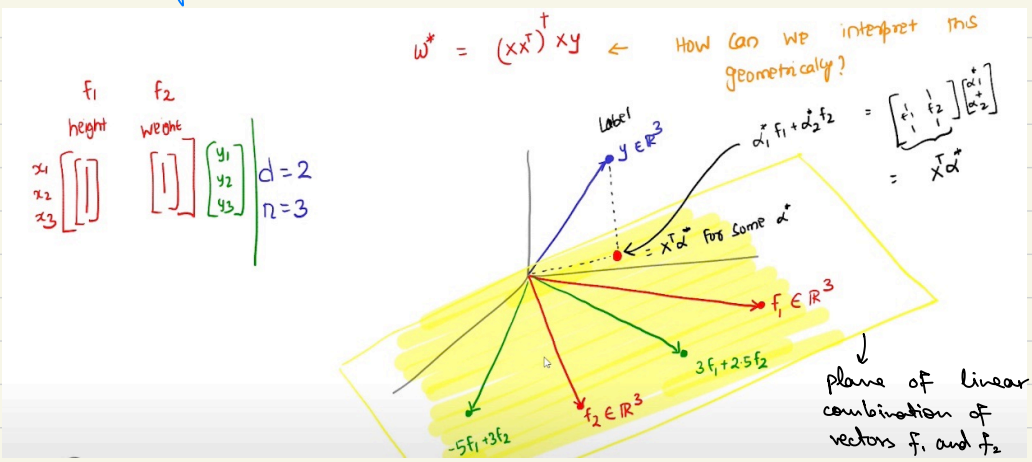
$$f(w) = (w^T X - y^T)(X^T w - y)$$

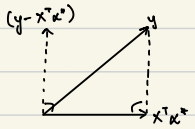
$$= w^T X X^T w - w^T X y - y^T X^T w + y^T y$$

$$= 2\left(\tfrac{1}{2} w^T X X^T w - w^T X y\right)$$

→ Geometric interpretation of Linear Regression

↳





$$\Rightarrow (y - X^T\alpha^*)(X^T\alpha^*) = 0$$

$$y^T X^T \alpha^* - \alpha^{*T}(XX^T)\alpha^* = 0$$

Recall, $w^* = (XX^T)^\dagger Xy$

↳ substitute $w^* = \alpha^*$: $y^T X^T ((XX^T)^\dagger Xy) - ((XX^T)^\dagger Xy)^T - (XX^T)((XX^T)^\dagger Xy)$

↳ Conclusion: $X^T w^*$ is the projection of the labels onto the subspace spanned by the features.

→ Gradient Descent

↳ $W^* = (XX^T)^+ XY$ ← inverse computation expensive $O(d^3)$

↳ we can apply gradient descent algorithm to obtain W^*.

↳ $W^{t+1} = W^t - \eta^t \nabla f(W^t)$

↳ Gradient descent for linear regression:

$$W^{t+1} = W^t - \eta^t \left[(XX^T)W - XY \right]$$

↳ If d (dimensions) is large, computing inverse of $(XX^T)^+$ was computationally expensive that's why use gradient descent algorithm.

↳ If n (# of datapoints) is large, computing XX^T is expensive. Use stochastic gradient descent.

↳ Stochastic Gradient Descent

↳ For $t = 1, \dots, T$

→ At each step, sample k datapoints uniformly at random from set of all the points.

→ Pretend the sampled datapoints is the entire dataset and take a gradient step w.r.t. the sampled points.

→ After T rounds, use $w^T_{sgd} = \frac{1}{T} \sum_{t=1}^{T} w^t$ → average of all w

↳ guaranteed to converge to optima with high probability.

→ Kernel Regression

↳ $W^* = (XX^T)^+ XY$

→ W^* must lie in the span of datapoints.

↳ if it didn't lie on the plane, then it cannot be the optimal solution because the projection of w^* would further minimize the error

→ W^* is some linear combination of the datapoints

$W^* = X\alpha^*$ for some $\alpha^* \in \mathbb{R}^n$

$\Rightarrow X\alpha^* = (XX^T)^+ XY$

$(XX^T) X \alpha^* = XY$

$X^T (XX^T) X \alpha^* = X^T XY$

$\underbrace{(X^T X)^2}_{K} \alpha^* = (X^T X) Y \quad \Rightarrow K^2 \alpha^* = KY \Rightarrow \alpha^* = K^{-1} Y$

↳ $\in \mathbb{R}^{n \times n}$

↳ Prediction:

For some $x_{test} \in \mathbb{R}^d$, $w^{*T} \phi(x_{test})$

where, $w^* = \arg\min \sum_i (\phi(x_i) - y_i)^2$

$$w^{*T} \phi(x_{test}) = \left(\sum_{i=1}^{n} \alpha_i^* \phi(x_i) \right)^T \phi(x_{test})$$

$$= \sum_{i=1}^{n} \alpha_i^* K(x_i, x_{test})$$

α_i^*: how important is i-th datapoint towards w^*

$K(x_i, x_{test})$: kernel function is as if it's asking "how similar are these two datapoints?"

→ Probabilistic view of Linear Regression

↳ $x \in \mathbb{R}^d$, $y \in \mathbb{R}$ Dataset: $\{(x_1, y_1), \dots, (x_n, y_n)\}$

→ Assume the label is generated as: $y|x \sim w^T x + \epsilon$ (unknown but fixed $\in \mathbb{R}^d$; Noise $\sim N(0, \sigma^2)$)

Probability of y label given x datapoint

↳ can view this as an estimation problem

↳ Maximum Likelihood

$$L(w; \underset{y_1,\dots,y_n}{x_1,\dots,x_n}) = \prod_{i=1}^{n} \exp\left(\frac{-(w^T x_i - y_i)^2}{2\sigma^2}\right) \cdot \frac{1}{\sqrt{2\pi}\sigma}$$

$$\log L(w; \underset{y_1,\dots,y_n}{x_1,\dots,x_n}) = \sum_{i=1}^{n} \frac{-(w^T x_i - y_i)^2}{2\sigma^2} \cdot \frac{1}{\sqrt{2\pi}\sigma}$$

$$\Updownarrow$$

$$\max_w \sum_{i=1}^{n} -(w^T x_i - y_i)^2 = \min_w \sum_{i=1}^{n} (w^T x_i - y_i)^2$$

$$\hat{w}_{ML} = w^* = (XX^T)^{-1} Xy$$

→ Maximum Likelihood estimator assuming zero-mean gaussian noise is same as Linear Regression with squared error!

WEEK 6

→ Goodness of ML estimator for linear regression

↳ $w^* = \hat{w}_{ML}$

$y|x = w^T x + \epsilon$, where $\epsilon \sim N(0, \sigma^2)$

↳ $y|x \sim N(w^T x, \sigma^2)$

↳ $E[\|\hat{w}_{ML} - w\|^2]$

↳ over randomness in y

↳ will converge to the expected value of this random variable

$$E[\|\hat{w}_{ML} - w\|] = \sigma^2 \cdot \text{trace}((XX^T)^{-1})$$

(trace of inverse of covariance matrix of the features)

average deviation b/w estimator $\hat{w}_{ML}$ and true w.

variance of the noise that is added to $w^T x$.

→ cannot do much about σ but can reduce $\text{trace}((XX^T)^{-1})$.

↳ $\text{trace}((XX^T)^{-1})$ $\quad tr(A) = \sum_i \lambda_i$

Let eigenvalues of (XX^T) be $\{\lambda_1, \dots, \lambda_d\}$

eigenvalues of $(XX^T)^{-1}$ are $\{1/\lambda_1, \dots, 1/\lambda_d\}$

→ Mean squared error of $\hat{w}_{ML}$:

$$E[\|\hat{w}_{ML} - w\|^2] = \sigma^2 \left(\sum_{i=1}^{d} 1/\lambda_i\right)$$, where λ_i are the eigenvalues of (XX^T)

→ Cross-validation

↳ Consider the estimator: $\hat{w}_{new} = \underbrace{(XX^T + \lambda I)^{-1} XY}_{\in \mathbb{R}^{d \times d}}$

For some matrix A, let eigenvalues: $\{\lambda_1, \dots, \lambda_d\}$

What are eigenvalues of $(A + \lambda I)$

↳ $\{\lambda_1 + \lambda, \dots, \lambda_d + \lambda\}$

$Av_i = \lambda_i v_i$

$(A+\lambda I)v_i = Av_i + \lambda v_i$

$= \lambda_i v_i + \lambda v_i$

$= (\lambda_i + \lambda) v_i$

$\text{trace}((XX^T + \lambda I)^{-1}) = \sum_{i=1}^{d} \frac{1}{\lambda_i + \lambda}$

→ **Existence Theorem**: $\exists \lambda \in \mathbb{R}$ s.t. $\hat{w}_{new} = (XX^T + \lambda I)^{-1} XY$ has lesser mean squared error than $\hat{w}_{ML}$

↳ Find λ by cross validation

Train set	Validation set
80%	20%

↳ Find $\hat{w}_{new}$ on training set.

↳ Train on the training set and check for error on validation set.

↳ Pick λ that gives the least error.

↳ K-Fold cross validation

F_1	F_2		...		...		F_K

↳ Train on folds $\{F_1, \dots, F_{i-1}, F_{i+1}, \dots, F_K\}$ Then validate on F_i

↳ Pick λ that gives the least average error.

→ Bayesian Modeling for Linear Regression

↳ Need a prior on w.

Likelihood: $y|x \sim N(w^T x, 1)$

A choice for prior: $w \sim N(0, \gamma^2 I)$ ($0 \in \mathbb{R}^d$; covariance matrix $\in \mathbb{R}^{d \times d}$: $\begin{bmatrix} \gamma^2 & & 0 \\ & \gamma^2 & \\ 0 & & \ddots \end{bmatrix}$)

Posterior:

$$P(w | \{(x_1,y_1),\dots,(x_n,y_n)\}) \propto \underbrace{P(\{(x_1,y_1),\dots,(x_n,y_n)\} | w)}_{\text{Likelihood}} \cdot \underbrace{P(w)}_{\text{Prior}}$$

$$\propto \left[\prod_{i=1}^{n} \exp\left(\frac{-(y_i - w^T x_i)^2}{2}\right)\right] \cdot \underbrace{\left[\prod_{i=1}^{n} \exp\left(\frac{-(w_i - 0)^2}{2\gamma^2}\right)\right]}_{\exp\left(\frac{-\|w\|^2}{2\gamma^2}\right)}$$

Max Apriori estimate:

$$\hat{w}_{MAP} = \arg\max_w \log(\square) = \arg\max_w \sum_{i=1}^{n} \frac{-(y_i - w^T x_i)^2}{2} - \frac{\|w\|^2}{2\gamma^2}$$

$$\hat{w}_{MAP} = \arg\min_w \frac{1}{2} \sum_{i=1}^{n} (y_i - w^T x_i)^2 + \frac{\|w\|^2}{\gamma^2}$$

↳ Take gradient set to 0.

$$\nabla f(w) = (XX^T)w - XY + \frac{w}{\gamma^2}$$

→ set $\nabla f(w) = 0$

$$\hat{w}_{MAP} = (XX^T + \frac{1}{\gamma^2} I)^{-1} XY$$

↳ MAP estimation for linear regression with a gaussian prior for w is equivalent to the w_{new} we used in cross-validation.

→ Ridge Regression

↳ $\hat{W}_{ML} = \underset{W}{\operatorname{argmin}} \sum_{i=1}^{n} (W^T x_i - y_i)^2$

$\hat{W}_R = \underset{W}{\operatorname{argmin}} \sum_{i=1}^{n} (W^T x_i - y_i)^2 + \lambda \|W\|^2$ → **Ridge Regression**

(Loss) (Regulariser)

↳ The goal of the regulariser is to push W towards zero. This helps in picking the W which has the least exposures to redundant features.

↳ "λ" in the regulariser is $\left(\frac{1}{\gamma^2}\right)$ where γ^2 is the variance. In the scenario where most features are redundant, a lot of W's will be zero, meaning low variance. If γ^2 is small, then λ is going to be large, which means it's going to pay a large penalty for increasing the length of W.

↳ $\min_{W \in \mathbb{R}^d} \sum_{i=1}^{n} (W^T x_i - y_i)^2 + \lambda \|W\|^2$ — (A)

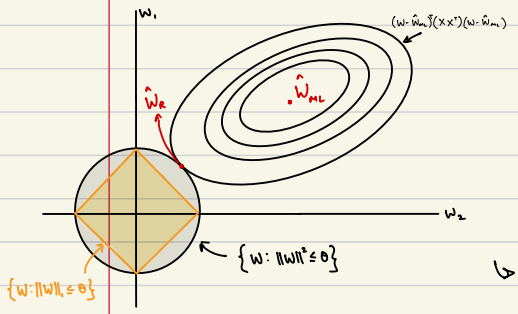
≡ is equivalent to $\min_{W \in \mathbb{R}^d} \sum_{i=1}^{n} (W^T x_i - y_i)^2$ s.t. $\|W\|^2 \le \theta$ — (B)

(θ depends on λ)

→ For every choice of $\lambda > 0$, $\exists \theta$ s.t. the optimal solutions of problems (A) and (B) coincide

Assuming $W = \begin{bmatrix} w_1 \\ w_2 \end{bmatrix}$

$\|W\|^2 \le \theta \Rightarrow w_1^2 + w_2^2 \le \theta$



→ Loss of linear regression at $\hat{W}_{ML}$: $f(W_{ML}) = \sum_{i=1}^{n} (\hat{W}_{ML}^T x_i - y_i)^2$

Consider the set of all W s.t. $f(W) = f(\hat{W}_{ML}) + c$; where $c > 0$

↳ $S_c = \{W : f(W) = f(\hat{W}_{ML}) + c\}$

every $W \in S_c$ satisfies $\|X^T W - Y\|^2 = \|X^T \hat{W}_{ML} - Y\|^2 + c$

If $XX^T = I$, $\|W - \hat{W}_{ML}\|^2 = c$ ⟸ $(W - \hat{W}_{ML})^T (XX^T)(W - \hat{W}_{ML}) = c'$

(c': some constant that depends on c, (XX^T), $\hat{W}_{ML}$ and not on W.)

↳ Conclusion:

→ Ridge regression pushes weight values towards zero. But does not necessarily make it zero.

→ Lasso Regression

↳ Alternate way of regularisation: use $\|\cdot\|_1$ norm instead of $\|\cdot\|_2^2$ norm.

$\|W\|_1 = \sum_{i=1}^{n} |w_i|$

↳ $\min_{W \in \mathbb{R}^d} \sum_{i=1}^{n} (W^T x_i - y_i)^2 + \lambda \|W\|_1 \equiv \min_{W \in \mathbb{R}^d} \sum_{i=1}^{n} (W^T x_i - y_i)^2$ s.t. $\|W\|_1 \le \theta$

↳ increases the chances of picking a W which has weights as zeros instead of really small values.

↳ Points to note:

→ Lasso does not have a closed-form solution.

→ Sub-gradient methods usually used to solve Lasso.

↳ **Subgradient**

→ A vector $g \in \mathbb{R}^d$ is a sub-gradient of $f: \mathbb{R}^d \to \mathbb{R}$ at a point $x \in \mathbb{R}^d$ if:

$$\forall z \quad f(z) \geq f(x) + g^T(z-x)$$

→ If the function f to minimize is a convex function, then sub-gradient descent algorithm converges

---

WEEK 7

→ Introduction to Binary Classification

↳ Input: $\{x_1, \dots, x_n\}$ $x_i \in \mathbb{R}^d$ and labels $y_1, \dots, y_n$ $y_i \in \{0,1\}$ / $\{-1,+1\}$

Goal: $h: \mathbb{R}^d \to \{0,1\}$

Loss: $\text{Loss}(h) = \frac{1}{n} \sum_{i=1}^{n} \mathbb{1}(h(x) \neq y_i)$ $\quad \mathbb{1}(z) = \begin{cases} 1 & \text{if true} \\ 0 & \text{otherwise} \end{cases}$

$\min_{h \in H_{\text{linear}}} \frac{1}{n} \sum_{i=1}^{n} \mathbb{1}(h(x) \neq y_i)$ $\quad$ where $H_{\text{linear}} = \{ h_w : h_w(x) = \text{sign}(w^T x) \}$

↳ NP-hard problem in general

↳ Can we use linear regression to solve classification problem?

$\{(x_1,y_1),\dots,(x_n,y_n)\} \to$ [Lin. reg.] $\to w \in \mathbb{R}^d \to h_w: \mathbb{R}^d \to \{0,1\}$

→ Regression is sensitive to the datapoints and not just the "side" on which data lies w.r.t. separator.

→ K-nearest Neighbour

↳ Given a test point $x_{\text{test}} \in \mathbb{R}^d$. Find the closest point x^* to x_{test} in the training test. Predict $y_{\text{test}} = y^*$

↳ Issue: can get affected by outliers

Fix: look at the distance from k- different neighbours.

↳ K-NN algorithm

→ Given x_{test}, find the k-closest points in training set - $(x_1^*, x_2^*, \dots x_k^*)$

→ Predict y_{test} = majority $(y_1^*, y_2^*, \dots, y_k^*)$

→ Too few k will result in outliers impacting the algorithm and too high k will make the algorithm very general (high loss)

→ Choosing k: cross-validate → try diff. k and see performance on validation set

→ Issues with K-nn:

① choosing a distance function

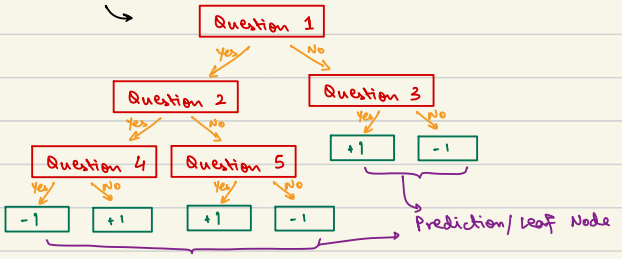
② Prediction is computationally expensive because of sorting

③ No model is learned - cannot throw data after "learning"

→ Introduction to Decision Trees

↳ Input: $\{(x_1, y_1), \ldots, (x_n, y_n)\}$ $\forall i$ $x_i \in \mathbb{R}^d$, $y_i \in \{+1, -1\}$

Output: Decision Tree



Prediction: Traverse through the tree to reach a leaf node.

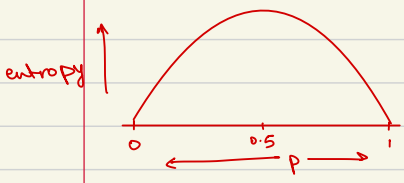
Question: (feature, value) pair. Example: height ≤ 180 cm?

↳ How to measure "goodness" of a question?

$D = \{(x_1, y_1), \ldots, (x_n, y_n)\}$

$f_k \leq \theta$ (Yes / No)

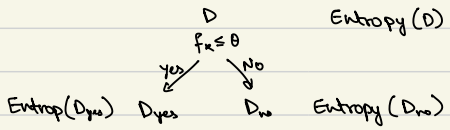
$D_{yes} = \{(x_1, y_1), (x_{10}, y_{10}), \ldots\}$ $D_{no} = \{(x_2, y_2), (x_3, y_3) \ldots\}$



→ measure of "impurity" for a set of labels $\{y_1, \ldots, y_n\}$

↳ Entropy $(\{y_1, \ldots, y_n\})$ = Entropy (p) $= -(p \log p + (1-p)\log(1-p))$

↳



Goodness of the question $f_k \leq \theta$.

Information Gain (Feature, value) = Entropy (D) − [γ Entropy (D_{yes}) + (1 − γ) Entropy (D_{no})]

where $\gamma = \frac{|D_{yes}|}{|D|} = \frac{\text{cardinality of } D_{yes}}{\text{cardinality of } D}$

↳ measures how much of the impurity is being removed by asking this question.

→ Decision Tree Algorithm

↳ Discretize each feature in [min, max] range

↳ Pick the question that has the highest Information Gain.

↳ Repeat the procedure for D_{yes}, D_{no}

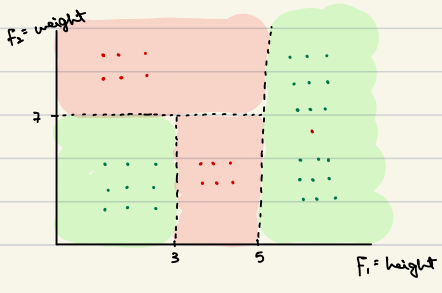
↳ Points:

→ can stop growing a tree if a node becomes "sufficiently" pure.

→ Depth of the tree (# of questions) is a hyperparameter.

→ Alternate measures for "goodness" of a question. Ex. Gini Index

Decision Boundary



→ Generative and Discriminative Models

↳ Types of Modelling

→ Assumption : there is some distribution D over $X \times Y$ (X: features, Y: labels)

$\{x_1, \dots, x_n\}$
$\{y_1, \dots, y_n\}$ $\quad (x_i, y_i) \sim D$

→ implies there is some probabilistic mechanism that generates the data.
↳ The link b/w Train and Test

↳ Classification

↳ ① Generative Model
② Discriminative Model

Generative	Discriminative
→ $P(x, y)$. There is some probability / density for every feature-value pair	→ $P(y \mid x)$. Probability of a label given the feature.

example: There is a sentence and the task is to tell whether the sentence is in Bengali or not

→ Generative model is to model how Bengali sentences are generated. Modelling x themselves. Joint Modelling.	→ Your goal is only to differentiate if the sentence is of the Bengali class or not. Don't necessarily need to know how Bengali sentences are generated.

→ If you have reasons to believe how the exact features are generated, then it is a good idea to do generative modelling. If, on the other hand, you have no idea how the features are generated, you're better off with Discriminative.

WEEK 8

→ Generative model-based algorithm

↳ Data = $\{(x_1, y_1), \ldots, (x_n, y_n)\}$ $\quad x_i \in \{0,1\}^d \quad y_i \in \{0,1\}$

example: spam classification of e-mails

$x_i \in \{0,1\}^d$, where d = # of words in a dictionary

e.g email = "Hello, how are you?"

$P(x,y) = P(\text{"Hello, how are you"}, \text{spam}) = ?$

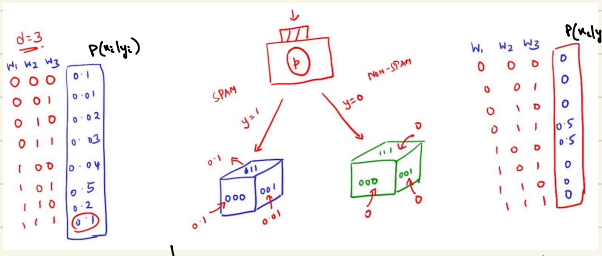
$P(x,y) = P(x) \cdot P(y/x) = \underbrace{P(y) \cdot P(x|y)}$ better way of modelling

↳ Generative Story

Steps:

① Decide the label by tossing a coin. $P(y_i = 1) = p$

② Decide the features using the label in step 1 by $P(x_i | y_i)$



↳ # of parameters in the model

$= 1 + [(2)^d - 1] + [(2)^d - 1] \rightarrow P(x_i|y_i)$

label, whether an email is spam or not.

for the blue die (spam world)

↳ for the green die (spam world)

$= 15$ for the example above

→ If we know all the 15 parameters, then we have the ability to generate a new (x,y) pair.

$= 1 + (2^d - 1) + (2^d - 1)$

$= 1 + 2(2^d - 1)$

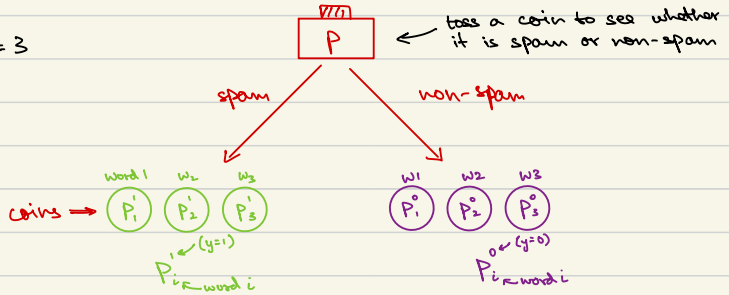
$= 2^{d+1} - 1$ parameters

↳ Issue: Too many parameters!

→ Alternate generative model-based algorithm

↳

$d = 3$



parameters = $1 + d + d$

$= 2d + 1$ ← much better than $(2^{d+1} - 1)$

→ The difference is that the features in this algorithm are indepent to each other. The features are orthogonal.

→ Class Conditional Inpendence Assumption!

↳ Steps of the algorithm:

① $P(y=1) = p$

② $P(x = [f_1, f_2, \dots, f_d] \mid y) = \prod_{i=1}^{d} (p_i^y)^{f_i} (1 - p_i^y)^{1-f_i}$

→ Naive Bayes Algorithm

↳ Parameters to estimate: $p, \{p_1^1, \dots, p_d^1\}, \{p_1^0, \dots, p_d^0\}$

Maximum likelihood estimates:

$\hat{p} = \frac{1}{n} \sum_{i=1}^{n} y_i$ → Fraction of spam e-mails in the dataset

$\forall j \in \{1, \dots, d\}$ $\forall y \in \{0, 1\}$ $\hat{p}_j^y = \frac{\sum_{i=1}^{n} \mathbb{1}(f_j^i = 1, y_i = y)}{\sum_{i=1}^{n} \mathbb{1}(y_i = y)}$ ← # of emails with label y

→ Fraction of y-labelled emails that contain the jth word

↳ Prediction

Given $x^{test} \in \{0,1\}^d$, what is $\hat{y}^{test}$?

$P(y^{test} = 1 \mid x^{test}) > P(y^{test} = 0 \mid x^{test}) \Rightarrow \hat{y}^{test} = 1$, 0 otherwise

↳ How to obtain $P(y|x)$ from $P(y)$ and $P(x|y)$

→ $P(y^{test} = 1 \mid x^{test}) = \frac{P(x^{test} \mid y^{test} = 1) \cdot P(y^{test} = 1)}{P(x^{test})}$

$P(A|B) \cdot \frac{P(B|A) \cdot P(A)}{P(B)}$

$P(x^{test} \mid y^{test} = 1) \cdot P(y^{test} = 1) = \left[\prod_{i=1}^{d} (\hat{p}_i^1)^{f_i} (1 - \hat{p}_i^1)^{(1-f_i)} \right] \cdot \hat{p}$

compare ⇕ with

$P(x^{test} \mid y^{test} = 0) \cdot P(y^{test} = 0) = \left[\prod_{i=1}^{d} (\hat{p}_i^0)^{f_i} (1 - \hat{p}_i^0)^{(1-f_i)} \right] \cdot (1 - \hat{p})$

↳ Pitfalls of Naive Bayes

→ If a word does not appear in the train set but appears in a test datapoint, $\hat{p}_j^1 = 0$ and $\hat{p}_j^0 = 0$

↳ this will result in $\hat{y}^{test}$ not being classified. Occurence of one word is throwing away all the information from other words.

↳ Possible Fix: Laplace Smoothing

↳ Add two "pseudo" emails with all words present — one email has label 0 and another 1.

$X_1^{pseudo} = [1, 1, 1, \ldots, 1]$ $\quad y_1^{pseudo} = 1$

$X_0^{pseudo} = [1, 1, 1, \ldots, 1]$ $\quad y_0^{pseudo} = 0$

→ this way every word will have occured at least once in the test dataset.

↳ Decision Function:

Given x_{test}, $y_{test} = 1$ if $\dfrac{P(y_{test}=1 \mid x_{test})}{P(y_{test}=0 \mid x_{test})} \geq 1$

$$\Rightarrow \log\left[\frac{P(x_{test} \mid y_{test}=1) \cdot P(y_{test}=1) \,/\, P(x_{test})}{P(x_{test} \mid y_{test}=) \cdot P(y_{test}=0) \,/\, P(x_{test})}\right] \geq 0$$

$x_{test} = [f_1, f_2 \cdots f_d]$

$$\Rightarrow \log\left[\prod_{i=1}^{d} \frac{(\hat{p}_i^1)^{f_i} (1-\hat{p}_i^1)^{(1-f_i)} \cdot \hat{p}}{(\hat{p}_i^0)^{f_i} (1-\hat{p}_i^0)^{(1-f_i)} \cdot (1-\hat{p})}\right] \geq 0$$

$$\Rightarrow \left[\sum_{i=1}^{d} f_i \log\left(\frac{\hat{p}_i^1}{\hat{p}_i^0}\right) + (1-f_i)\log\left(\frac{(1-\hat{p}_i^1)}{(1-\hat{p}_i^0)}\right) + \log\left(\frac{\hat{p}}{(1-\hat{p})}\right)\right] \geq 0$$

$$\Rightarrow \sum_{i=1}^{d} f_i \left(\log\left(\frac{\hat{p}_i^1 (1-\hat{p}_i^0)}{\hat{p}_i^0 (1-\hat{p}_i^1)}\right)\right) + \log\left(\frac{(1-\hat{p}_i^1)}{(1-\hat{p}_i^0)}\right) + \log\left(\frac{\hat{p}}{(1-\hat{p})}\right) \geq 0$$

f_i: features; the first log term: constants; the remaining log terms: constants

Decision function is of the form:

Predict $y_{test} = 1$ if $w^T x_{test} + b \geq 0$, $\quad w \in \mathbb{R}^d$

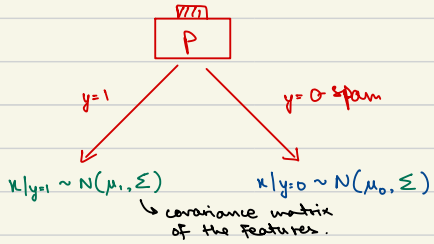
where $w_i = \log\left(\dfrac{\hat{p}_i^1 (1-\hat{p}_i^0)}{\hat{p}_i^0 (1-\hat{p}_i^1)}\right)$ and $b = \log\left(\dfrac{(1-\hat{p}_i^1)}{(1-\hat{p}_i^0)}\right) + \log\left(\dfrac{\hat{p}}{(1-\hat{p})}\right)$

Conclusion: decision function of naive bayes is linear!

→ Gaussian Naive Bayes

↳ Data: $\{(x_1, y_1), \ldots, (x_n, y_n)\}$, $x_i \in \mathbb{R}^d$ $y_i \in \{0, 1\}$

A generative story:



Note: covariances of x are the same in both cases of y=0 or y=1.

parameters: p, μ_1, μ_0, Σ

Max Likelihood estimate for parameters:

$$\hat{p} = \frac{1}{n} \sum_{i=1}^{n} y_i$$

$$\hat{\mu}_1 = \frac{\sum_{i=1}^{n} \mathbb{1}(y_i = 1) \cdot x_i}{\sum_{i=1}^{n} \mathbb{1}(y_i = 1)}$$ ← sample mean of data points labelled 1

$$\hat{\mu}_0 = \frac{\sum_{i=1}^{n} \mathbb{1}(y_i = 0) \cdot x_i}{\sum_{i=1}^{n} \mathbb{1}(y_i = 0)}$$

$$\hat{\Sigma} = \frac{1}{n} \sum_{i=1}^{n} (x_i - \hat{\mu}_{y_i})(x_i - \hat{\mu}_{y_i})^T$$

↳ Prediction

$$P(y_{test} | x_{test}) \propto \underbrace{P(x_{test} | y_{test})}_{f(x_{test}; \hat{\mu}_{y_{test}}, \hat{\Sigma})} \cdot \underbrace{P(y_{test})}_{\hat{p}}$$

$$y_{test} = 1 \text{ if } f(x_{test}; \hat{\mu}_1, \hat{\Sigma}) \cdot \hat{p} \geq f(x_{test}; \hat{\mu}_0, \hat{\Sigma}) \cdot (1 - \hat{p})$$

$$\exp\left(-(x_{test} - \hat{\mu}_1)^T \cdot \hat{\Sigma}^{-1} \cdot (x_{test} - \hat{\mu}_1)\right) \cdot \hat{p} \geq \exp\left(-(x_{test} - \hat{\mu}_0)^T \cdot \hat{\Sigma}^{-1} \cdot (x_{test} - \hat{\mu}_0)\right) \cdot (1 - \hat{p})$$

⇓ simplification by taking a log ⇓

Predict $y_{test} = 1$ if:

$$\underbrace{(\hat{\mu}_1 - \hat{\mu}_0)^T \hat{\Sigma}^{-1}}_{w^T} \underbrace{x_{test}}_{x_{test}} + \underbrace{\hat{\mu}_0^T \hat{\Sigma}^{-1} \hat{\mu}_0 - \hat{\mu}_1^T \hat{\Sigma}^{-1} \hat{\mu}_1 + \log\left(\frac{(1 - \hat{p})}{\hat{p}}\right)}_{b} \geq 0$$

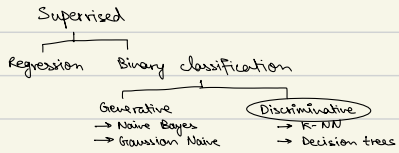
$$w^T x_{test} + b \geq 0$$

⇒ Decision function is again linear!

↳ assumption of Σ is same in both classes is imp.

→ If the covariance matrix is identity matrix, the contours are circular, not eliptical.

WEEK 9



→ Discriminative Models for classification

↳ How to model $P(y=1|x)$

Simplest assumption:

$$P(y=1|x) = \begin{cases} 1 & \text{if } W^T x \geq 0 \\ 0 & \text{otherwise} \end{cases}$$

↳ **Linear Separability Assumption**:

$\exists\, W \in \mathbb{R}^d$ s.t. $\text{sign}(W^T x_i) = y_i \;\; \forall i$

→ Perceptron Learning Algorithm

↳ Input: $\{(x_1, y_1), \ldots, (x_n, y_n)\}$ $\quad x_i \in \mathbb{R}^d$, $y_i \in \{+1, -1\}$

→ W^{0} ← iteration $= 0 \in \mathbb{R}^d$ $\quad [0 \ldots 0]$

Until convergence:

→ Pick (x_i, y_i) pair from dataset

→ If $\text{sign}(W^{t^T} x_i) = y_i$, do nothing

Else $W^{t+1} = W^t + x_i y_i$ ← Update rule

↳ Update Rule:

Mistake type 1	Mistake type 2
→ Predicted = 1 ← $W^T x_i \geq 0$	→ Predicted = -1 ← $W^T x_i < 0$
Actual = -1 ← $y_i = -1$	Actual = 1 ← $y_i = +1$
$W^{t+1} = W^t + x_i y_i$	$(W^{t+1})^T x_i = \underbrace{W^{t^T} x_i}_{<0} + \underbrace{y_i \|x_i\|^2}_{>0}$
$(W^{t+1})^T x_i = (W^t + x_i y_i)^T x_i$	
$= \underbrace{W^{t^T} x_i}_{\geq 0} + \underbrace{y_i}_{-1} \underbrace{\|x_i\|^2}_{\geq 0}$	
Negative	

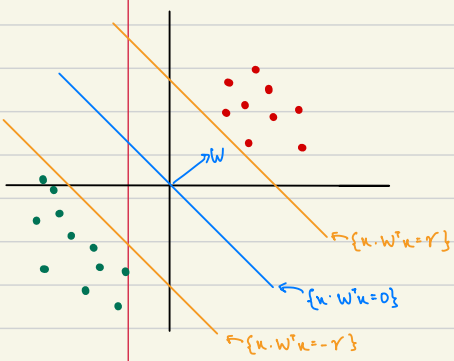
→ Update rule pushes W in the right direction for x_i

→ Fixing it locally might break something else for some other datapoint.

↳ Assumption: **Linear Seperability with γ - margin**

→ Dataset $\{(x_1, y_1), \ldots, (x_n, y_n)\}$ is linearly separable with γ margin if

$\exists\, W^* \in \mathbb{R}^d$ s.t. $((W^*)^T x_i) y_i \geq \gamma \;\; \forall i \;\; ; \gamma > 0$



↳ Assumption : Radius Assumption
→ $\|x_i\|_2 \le R \quad \forall i$ for some $R > 0$
↳ all the points fall within a ball of radius R

↳ Assumption : Without loss of generality, assume $\|W^*\| = 1$
→ if you rescale W^* by $\|W^*\|$, γ also gets rescaled.

↳ Proof of Perceptron
→ Analysis of "mistakes" made by perceptron
Update happens only when there is a mistake
Say W^l is the current guess and a mistake happens w.r.t (x, y)

$$W^{l+1} = W^l + x \cdot y$$

$$\|W^{l+1}\|^2 = \|W^l + x \cdot y\|^2$$

$$= (W^l + x \cdot y)^T (W^l + x \cdot y)$$

$$= \|W^l\|^2 + \underbrace{2(W^{lT} \cdot x) y}_{\le 0 \text{ because mistake}} + \underbrace{\|x\|^2 \cdot y^2}_{\le R^2 \text{ radius assumption}}$$

$$\|W^{l+1}\|^2 \le \|W^l\|^2 + R^2$$

$$\|W^{l+1}\|^2 \le (\|W^{l-1}\|^2 + R^2) + R^2$$

$$\vdots$$

$$\|W^{l+1}\|^2 \le \|W^0\|^2 + lR^2 \quad \text{(1)}$$

(l ← # of mistakes)

→ $(W^{l+1})^T W^* = (W^l + x \cdot y)^T W^*$

$$= W^{lT} W^* + \underbrace{(W^{*T} \cdot x) y}_{\ge \gamma}$$

$$(W^{l+1})^T W^* \ge W^{lT} W^* + \gamma$$

$$\vdots$$

$$(W^{l+1})^T W^* \ge W^{0T} W^* + l\gamma \quad \text{(2)}$$

→ $l\gamma \le (W^{l+1})^T W^*$

$\Rightarrow (l\gamma)^2 \le ((W^{l+1})^T W^*)^2 \le \|W^{l+1}\|^2 \cdot \underbrace{\|W^*\|^2}_{1}$ → From Cauchy-Schwartz

$\Rightarrow \|W^{l+1}\|^2 \ge l^2\gamma^2 \quad \text{(3)}$

$\Rightarrow l^2\gamma^2 \le \|W^{l+1}\|^2 \le lR^2$

$\Rightarrow l^2\gamma^2 \le lR^2$

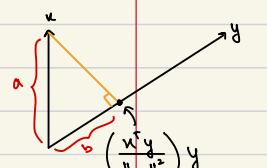
$\Rightarrow l \le \frac{R^2}{\gamma^2}$ ← Radius, ← Margin

← Radius Margin Bound

→ # of mistakes is bounded because $\gamma > 0$

Cauchy-Schwartz

For any x, y



$b < a$ — Pythagorean Theorem

$$\left\| \left(\frac{x^T y}{\|y\|^2}\right) y \right\|^2 \le \|x\|^2$$

$$(x^T y)^2 \frac{\|y\|^2}{\|y\|^4} \le \|x\|^2$$

$$(x^T y)^2 \le \|x\|^2 \cdot \|y\|^2$$

→ Modelling Class Probabilities

↳ For perceptron: $P(y=1|x) = 1$ if $W^T x \geq 0$, 0 otherwise

↳ Simple linear model

$z = W^T x$ → score

→ We want such that larger the value of z, score, higher the probability of being classified as +1.

→ $g: \mathbb{R} \to [0,1]$ $\quad g(z) = 0.5$ if $z = 0$

$g(z) \to 1$ as $z \to \infty$

$g(z) \to 0$ as $z \to -\infty$

Sigmoid / Logistic function · $g(z) = \frac{1}{1+e^{-z}}$

→ Logistic Regression

↳ Model: $P(y=1|x) = \frac{1}{1+e^{-W^T x}} = g(W^T x)$ Data: $\{(x_1, y_1), \ldots, (x_n, y_n)\}$ $x_i \in \mathbb{R}^d$, $y_i \in \{0,1\}$

Find W using Max. Likelihood

$$L(W; \text{Data}) = \prod_{i=1}^{n} (g(W^T x_i))^{y_i} \cdot (1 - g(W^T x_i))^{(1-y_i)}$$

$$\log L(W; \text{Data}) = \sum_{i=1}^{n} y_i \log(g(W^T x_i)) + (1-y_i) \log(1 - g(W^T x_i))$$

$$= \sum_{i=1}^{n} y_i \log\left(\frac{1}{1+e^{-W^T x_i}}\right) + (1-y_i) \log\left(\frac{e^{-W^T x_i}}{1+e^{-W^T x_i}}\right)$$

$$= y_i \log(1) - y_i \log(1+e^{-W^T x_i}) + \log(e^{-W^T x_i}) - \log(1+e^{-W^T x_i}) - y_i \log(e^{-W^T x_i}) + y_i \log(1+e^{-W^T x_i})$$

$$= -W^T x_i + y_i(W^T x_i) - \log(1+e^{-W^T x_i})$$

$$= \sum_{i=1}^{n} (1-y_i)(-W^T x_i) - \log(1+e^{-W^T x_i})$$

Goal: $\max_W \log L(W; \text{Data})$ → no closed-form solution

→ Use gradient ascent

$$\nabla \log L(W) = \sum_{i=1}^{n} (1-y_i)(-x_i) - \left(\frac{e^{-W^T x_i}}{1+e^{-W^T x_i}}\right)(-x_i)$$

$$= \sum_{i=1}^{n} y_i x_i - x_i \left(\frac{1}{1+e^{-W^T x_i}}\right)$$

$$= \sum_{i=1}^{n} x_i \left(y_i - \frac{1}{1+e^{-W^T x_i}}\right)$$

$$W_{t+1} = W_t + \eta_t \nabla \log(W)$$

$$= W_t + \eta_t \left[\sum_{i=1}^{n} x_i \left(y_i - \frac{1}{1+e^{-W^T x_i}}\right)\right]$$

$\nabla \log L$ is some linear combination of the dataset.

$y_i \in \{0,1\}$, $\frac{1}{1+e^{-W^T x_i}} \to g(W_t^T x_i)$

↳ Prediction: $\hat{y}_{test} = \text{sign}(\hat{W}^T x_{test})$

WEEK 10

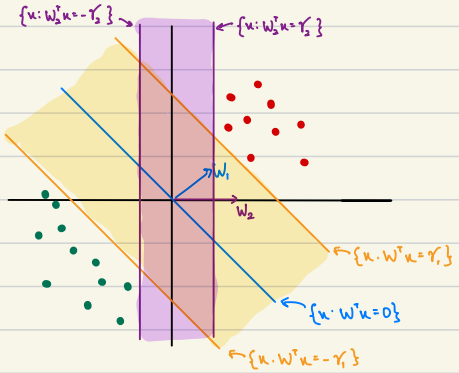
→ Perceptrons and Margin

↳ Quality of Final Solution

→ W and W_2 are both acceptable seperators.

→ # of mistakes depend on the best possible W^* margin

→ The W that perceptron outputs need not be the one with the largest margin γ



↳ Maximise Margin

Goal: formulate margin maximisation

→ $\max_{W,\gamma} \gamma$ s.t. $(W^T x_i) y_i \geq \gamma \ \forall i$

↳ This is not useful because W can grow arbitrarily

→ $\max_{W,\gamma} \gamma$ s.t $(W^T x_i) y_i \geq \gamma \ \forall i$ and $\|W\|^2 = 1$

→ If you fix $\|W\|^2$, γ varies and if you fix γ, $\|W\|^2$ can be scaled.

→ $\max_W$ width(W) s.t. $(W^T x_i) y_i \geq 1 \ \forall i$

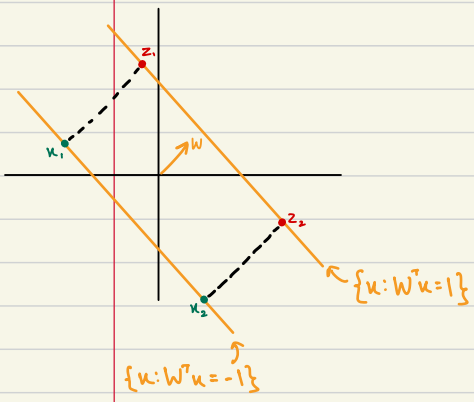
→ Define width(W).

↳ $\min_z \frac{1}{2}\|x - z\|^2$ s.t $W^T z = 1$, where $W^T x = -1$

↳ solution to the optimisation: width$(W) = \frac{2}{\|W\|^2}$

→ Final formulation: $\max_W \frac{2}{\|W\|^2}$ s.t. $(W^T x_i) y_i \geq 1 \ \forall i$

$\equiv \min_W \frac{\|W\|^2}{2}$ s.t. $(W^T x_i) y_i \geq 1 \ \forall i$



→ Constrained Optimisation

↳ $\min_w f(w) \quad \text{s.t. } g(w) \le 0$

$L(w, \alpha) = f(w) + \alpha g(w)$

Fix any w. Consider $\max_{\alpha \ge 0} L(w, \alpha) = \max_{\alpha \ge 0} f(w) + \alpha g(w)$

$$= \begin{cases} \infty & \text{if } g(w) > 0 \\ f(w) & \text{if } g(w) \le 0 \end{cases}$$

→ $\min_w \left[\max_{\alpha \ge 0} f(w) + \alpha g(w) \right]$

If $f(w)$ and $g(w)$ are convex functions,

then $\min_w \left[\max_{\alpha \ge 0} f(w) + \alpha g(w) \right] \equiv \max_{\alpha \ge 0} \left[\min_w f(w) + \alpha g(w) \right]$

↳ for multiple constraints

$$\min_w f(w) \quad \text{s.t. } g_i(w) \le 0 \quad \forall i = 1, \dots, K$$

$$\equiv \min_w \left[\max_{\alpha_i \ge 0 \, \forall i} f(w) + \sum_{i=1}^{K} \alpha_i g_i(w) \right]$$

→ Perceptrons and Margins

↳ $\min_w \frac{\|w\|^2}{2} \quad \text{s.t. } (w^T x_i) y_i \ge 1 \quad \forall i = 1, \dots, n$

$\frac{\|w\|^2}{2}$: quadratic in w ; $(w^T x_i) y_i \ge 1$: linear in w

$f(w) = \frac{\|w\|^2}{2} \qquad g_i(w) = 1 - (w^T x_i) y_i$

$L(w, \alpha) = \frac{\|w\|^2}{2} + \sum_{i=1}^{n} \alpha_i (1 - (w^T x_i) y_i)$

$$\min_w \max_{\alpha \ge 0} L(w, \alpha) \equiv \max_{\alpha \ge 0} \min_w L(w, \alpha)$$

① ②

① will fix the w and maximise over $\alpha \ge 0$.

② will fix the $\alpha \ge 0$ and minimize over w; makes the minimisation of w an unconstrained minimisation (solved by setting the gradient to 0).

Example using ②:

Fix $\alpha = \begin{bmatrix} 2 \\ 1 \\ 3 \end{bmatrix}$; $\min_w \frac{1}{2} \|w\|^2 + \sum_{i=1}^{n} \alpha_i (1 - (w^T x_i) y_i)$

$0 = w + \sum_{i=1}^{n} \alpha_i (-x_i y_i)$

$w^* = \sum_{i=1}^{n} \alpha_i (x_i y_i)$ ($\alpha_i \ge 0$, $x_i \in \mathbb{R}^d$, $y_i \in \{\pm 1\}$)

$X = \begin{bmatrix} | & & | \\ x_1 & \cdots & x_n \\ | & & | \end{bmatrix}_{d \times n} \quad Y = \begin{bmatrix} y_1 & & 0 \\ & \ddots & \\ 0 & & y_n \end{bmatrix}_{n \times n} \quad \alpha = \begin{bmatrix} \alpha_1 \\ \vdots \\ \alpha_n \end{bmatrix}_{n \times 1}$

$w^* = XY\alpha$

Substitute w into objective:

$\alpha^T 1 - \frac{1}{2} (XY\alpha)^T (XY\alpha) \qquad 1 = \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix}$

↳ Dual Problem: $\max_{\alpha \geq 0} \; \alpha^T 1 - \frac{1}{2} \alpha^T y^T X^T X y \alpha$

↳ Advantages:

→ Dual variable dimension in $\mathbb{R}^n_+$ while primal problem dimension in $\mathbb{R}^d$.

→ Dual constraints are "easier"

→ The objective of dual depends on $X^T X$ and so can be "kernelized"

→ Support Vector Machine

↳ Revisiting Lagrangian:

$$\min_w \left[\max_{\alpha \geq 0} f(w) + \alpha g(w) \right] \equiv \max_{\alpha \geq 0} \left[\min_w f(w) + \alpha g(w) \right]$$

Primal — Dual

→ say w^* is the primal solution and α^* is the dual solution

$$\max_{\alpha \geq 0} f(w^*) + \alpha g(w^*) = \min_w f(w) + \alpha^* g(w)$$

$$f(w^*) = \min_w f(w) + \alpha^* g(w)$$

$$\Rightarrow f(w^*) \leq f(w^*) + \alpha^* g(w^*)$$

$$\Rightarrow \alpha^* g(w^*) \geq 0 \quad — ①$$

→ but we know $\alpha^* \geq 0$ and $g(w^*) \leq 0$

$$\Rightarrow \alpha^* g(w^*) \leq 0 \quad — ②$$

→ Only way ① & ② are true: $\alpha^* g(w^*) = 0$

Complimentary Slackness

↳ In our problem:

$$\alpha_i^* (1 - (w^{*T} x_i) y_i) = 0 \quad \forall i$$

$$\Rightarrow \text{If } \alpha_i^* > 0 \Rightarrow 1 - (w^{*T} x_i) y_i = 0 \Rightarrow (w^{*T} x_i) y_i = 1$$

Only the points on the "supporting" hyperplane can contribute to w^*

w^* is a sparse linear combination of the data points

↳ Prediction:

Given x_{test}, $w^{*T} x_{test} = \left(\sum_{i=1}^n \alpha_i^* x_i y_i \right)^T x_{test}$

$$= \sum_{i=1}^n \alpha_i^* \cdot y_i (x_i^T \cdot x_{test})$$

In the kernel,

$$w^{*T} \phi(x_{test}) = \sum_{i=1}^n \alpha_i^* y_i k(x_i, x_{test})$$

$\alpha_i^* > 0$ only for support vector

→ Soft-Margin SVM

↳ How to deal with outliers?

Insight: Make every W feasible. Earlier, only the W that that correctly classified the datapoints were feasible.

↳ Fix any W. W classifies some points correctly and misclassifies some other points

For the incorrectly classified points "pay penalty" to go to the "correct" side.

↳ Modified Formulation:

≥ 0 → Hyperparameter

$$\min_{W,\epsilon} \frac{1}{2}\|W\|^2 + C\sum_{i=1}^{n}\epsilon_i$$

$$\text{s.t. } (W^T x_i)y_i + \underbrace{\epsilon_i}_{\geq 0} \geq 1 \quad \forall i$$

→ If $C=0$, $W=0 \in \mathbb{R}^d$ is the solution

If $C=\infty$, W is the same as strict linear seperator

WEEK 11

→ Dual Formulation For Soft-Margin SVM

↳ Primal Problem

$$\min_{W,\epsilon} \frac{1}{2}\|W\|^2 + C\sum_{i=1}^{n}\epsilon_i \quad \text{s.t. } (W^T x_i)y_i + \epsilon_i \geq 1 \ \forall i \quad \epsilon_i \geq 0 \ \forall i$$

Lagrangian:

$$L(W,\epsilon,\alpha,\beta) = \frac{1}{2}\|W\|^2 + C\sum_{i=1}^{n}\epsilon_i + \sum_{i=1}^{n}\alpha_i[1-(W^T x_i)y_i - \epsilon_i] + \sum_{i=1}^{n}\beta_i(-\epsilon_i)$$

$$\min_{W,\epsilon}\left[\max_{\substack{\alpha\geq 0\\ \beta\geq 0}} L(W,\epsilon,\alpha,\beta)\right] \equiv \max_{\substack{\alpha\geq 0\\ \beta\geq 0}}\left[\min_{W,\epsilon} L(W,\epsilon,\alpha,\beta)\right]$$

$$\frac{\partial L}{\partial W} = 0 \Rightarrow W^*_{\alpha,\beta} = \sum_{i=1}^{n}\alpha_i x_i y_i \quad \text{——} \ (1)$$

$$\frac{\partial L}{\partial \epsilon_i} = 0 \Rightarrow C + \alpha_i(-1) + \beta_i(-1) = 0 \Rightarrow \alpha_i + \beta_i = C \quad \text{——} \ (2)$$

Substitute $W^*_{\alpha,\beta}$ into the Lagrangian:

Dual Problem: $\max_{\substack{\alpha\geq 0\\ \beta\geq 0\\ \alpha+\beta=C1}} \alpha^T 1 - \frac{1}{2}\alpha^T Y^T X^T X Y \alpha$

↖ no β. β is effectively restricting α to C.

↳ Dual Problem: $\max_{0 \le \alpha \le C} \alpha^T 1 - \frac{1}{2} \alpha^T y^T x^T x y \alpha$

→ If $C=0$, $\alpha^* = 0 \in \mathbb{R}^n \Rightarrow W^* = \sum \alpha_i x_i y_i = 0$
→ If $C=\infty$, ⇒ Hard-margin SVM
→ Kernelisable because of $X^T X$

↳ Compliwentary Slackness
Let (W^*, ξ^*) be the primal optimal solutions
Let (α^*, β^*) " " dual " "

① $\forall i \ \alpha_i^* (1 - W^* x_i y_i - \xi_i^*) = 0$
② $\forall i \ \beta_i^* (\xi_i^*) = 0$

Cases possible: (dual problem)
① $\alpha_i^* = 0 \Rightarrow \beta_i^* = C \Rightarrow \xi_i^* = 0 \Rightarrow W^{*T} x_i y_i \ge 1$
↳ W^* classifies (x_i, y_i) correctly
② $\alpha_i^* \in (0, C) \Rightarrow 0 < \beta_i^* < C \Rightarrow \xi_i^* = 0 \Rightarrow W^{*T} x_i y_i = 1$
↳ x_i lies on the supporting hyperplane
③ $\alpha_i^* = C \Rightarrow \beta_i^* = 0 \Rightarrow \xi_i^* \ge 0 \Rightarrow (1 - W^{*T} x_i y_i - \xi_i^*) = 0$
↳ $W^{*T} x_i y_i \le 1 \Rightarrow x_i$ is being misclassified by W^* or isn't classified with enough margin

Cases possible: (primal problem)
① $W^{*T} x_i y_i < 1 \Rightarrow \xi_i^* > 0 \Rightarrow \beta_i^* = 0 \Rightarrow \alpha_i^* = C$
↳ x_i is misclassified
② $W^{*T} x_i y_i = 1 \Rightarrow \xi_i^* \ge 0 \Rightarrow \alpha_i^* \in [0, C]$
↳ x_i is on the supporting hyper-plane
③ $W^{*T} x_i y_i > 1 \Rightarrow \alpha_i^* = 0$
↳ x_i which is classified with margin >1 does not contribute to W^*

↳ Summary

Dual:
① $\alpha_i^* = 0 \Rightarrow W^{*T} x_i y_i \ge 1$
② $\alpha_i^* \in (0, C) \Rightarrow W^{*T} x_i y_i = 1$
③ $\alpha_i^* = C \Rightarrow W^{*T} x_i y_i \le 1$

Primal:
① $W^{*T} x_i y_i < 1 \Rightarrow \alpha_i^* = C$
② $W^{*T} x_i y_i = 1 \Rightarrow \alpha_i^* \in [0, C]$
③ $W^{*T} x_i y_i > 1 \Rightarrow \alpha_i^* = 0$

→ Algorithms so far

↳ Generative: Naive Bayes, Gaussian Naive Bayes
Discriminative: K-NN, Decision trees, Logistic Regression, Perceptron, SVM

→ Meta classifiers | Ensemble classifiers

↳ Weak learners ⟶ Strong learners
(better than random)

↳ Underfitting → high bias: even if the data changes a little, the model will vary within a small range.
Overfitting → high variance: if the data changes a little bit, the prediction is going to change by a lot.

↳ Bias: tells you how good is your assumed structure actually representing the truth.

↳ error = bias + variance

→ Bagging: Bootstrap Aggregation

↳ $X_1, X_2, \dots, X_n \sim N(\mu, 1)$

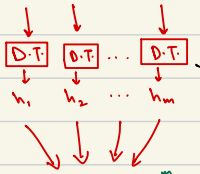
$\hat{\mu}_1 = X_1, \hat{\mu}_2 = X_2, \dots, \hat{\mu}_n = X_n$ } these are all unbiased estimators
$\hat{\mu}_{ML} = \frac{1}{n}\sum x_i$

Unbiased estimators: $E[\hat{\theta}] = \theta$

$E[\hat{\mu}_1] = E[X_1] = \mu$; similarly, $E[\hat{\mu}_{ML}] = \mu$

→ Using $\hat{\mu}_{ML}$ instead of $\hat{\mu}_1, \dots, \hat{\mu}_n$ reduces "variance"

↳ $D_1, D_2, \dots, D_m$ where each D_i has n datapoints;



$D_i = \{(x_1^i, y_1^i), (x_2^i, y_2^i), \dots, (x_n^i, y_n^i)\}$ where $x_j^i \in \mathbb{R}^d$, $y_j^i \in \{\pm 1\}$ $\forall i,j$

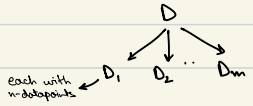
→ Decision Tree algorithms

→ $h_i : \mathbb{R}^d \to \{\pm 1\}$

Problem: we don't have m diff. datasets

↳ Input: $D = \{(x_1, y_1), \dots, (x_n, y_n)\}$ $x_i \in \mathbb{R}^d$ $y_i \in \{\pm 1\}$ $\forall i$

Bootstrapping: sampling with replacement



create m different datasets using bootstrapping and then train different models on each dataset

Probability that a point doesn't get picked: $1 - 1/n$

" " " " " " " " in any bag: $(1 - 1/n)^n$

Probability that a point appears in any bag: $1 - (1 - 1/n)^n \approx 67\%$ for $\gg n$

→ Bagging reduces variance

↳ Random Forest

→ Bagg Decision Trees (typically overfit trees)

→ Feature Bagging

↳ when building a decision tree, pick a subset of features using the concept of impurity.

→ Boosting

↳ also weak learner ⟶ strong learner

Decision stumps: $\boxed{f_k < \theta}$: 1-level or 2-level

$+1$ ↙ ↘ -1

↳ **Ada-Boost Algorithm**

Input: $S = \{(x_1, y_1), \dots, (x_n, y_n)\}$ $\quad x_i \in \mathbb{R}^d$ $\quad y_i \in \{\pm 1\}$

→ every datapoint gets assigned a weight

Initialize: $D_0(i) = \frac{1}{n}$

(iteration → 0; i ← i-th datapoint)

For $t = 1, \dots, T$:

→ h_t = Input (S, D_t) to a weak learner

→ $\tilde{D}_{t+1}(i) = \begin{cases} D_t(i) \cdot e^{\alpha_t} & \text{if } h_t(x_i) \neq y_i \\ D_t(i) \cdot e^{\alpha_t} & \text{if } h_t(x_i) = y_i \end{cases}$

→ $D_{t+1}(i) = \dfrac{\tilde{D}_{t+1}(i)}{\sum_j \tilde{D}_{t+1}(j)}$

Output: $h_1, h_2, \dots, h_T$. $h^*(x) = \text{sign}\left(\sum_t \alpha_t h_t(x)\right)$

$\alpha_t = \ln\sqrt{\dfrac{1 - \text{err}(h_t)}{\text{err}(h_t)}}$

WEEK 12

→ Classification Loss Functions

↳ Data: $\{(x_1, y_1), \dots (x_n, y_n)\}$ $x_i \in \mathbb{R}^d$, $y_i \in \{\pm 1\}$

Goal: $h : \mathbb{R}^d \to \{\pm 1\}$

Performance measure: $\sum_{i=1}^{n} \mathbb{1}(h(x_i) \neq y_i)$

if $h \in H_{linear}$,

$$\min \sum_{i=1}^{n} \mathbb{1}(h(x_i) = y_i) \equiv \min_{\substack{w \in \mathbb{R}^d \\ b \in \mathbb{R}}} \sum_{i=1}^{n} \mathbb{1}(\text{sign}(w^T x_i + b) \neq y_i)$$

↳ NP-HARD

$x \in \mathbb{R}^d$, $x' = [x\ 1] \in \mathbb{R}^{d+1}$

$w \in \mathbb{R}^d$, $w' = [\underbrace{w_1, \dots, w_d}_{w}, \underbrace{w_{d+1}}_{b}]$ $\quad w'^T x' = w^T x + b$

↳ Loss-function view

→ every point suffers a loss of 1 or 0

(x, y) with $x \in \mathbb{R}^d$, $y \in \{\pm 1\}$ $\quad h : \mathbb{R}^d \to \{\pm 1\}$

$\mathbb{1}(h(x) \neq y) = \mathbb{1}(h(x) \cdot y < 0)$ $\quad h_w(x) = \text{sign}(w^T x)$

↳ Algorithm

→ Using Regression for Classification

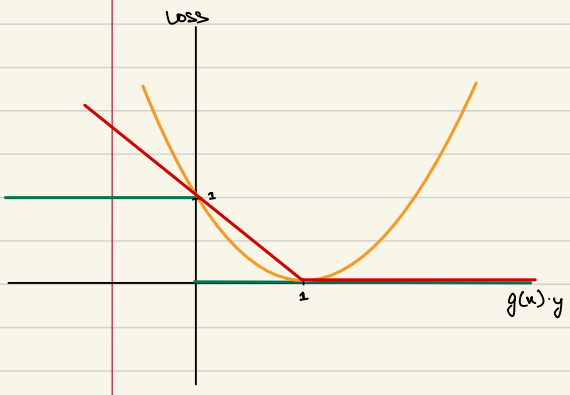
$$\text{Loss}(g, (x,y)) = (g(x) - y)^2 \quad \text{— ①} \quad \text{where } g(x) = w^T x + b$$

$$= (g(x)y - 1)^2 \quad \text{— ②}$$

$$= g(x)^2 + 1 - 2g(x) \cdot y$$

→ ① and ② are not graphically same although they're algebraically equivalent.

→ Therefore, regression is not a great algorithm for binary classification



→ 0-1 Loss

→ Squared Loss $(g(x)y - 1)^2$

→ SVM Loss $\max(0, 1 - g(x) \cdot y)$

→ Support Vector Machine

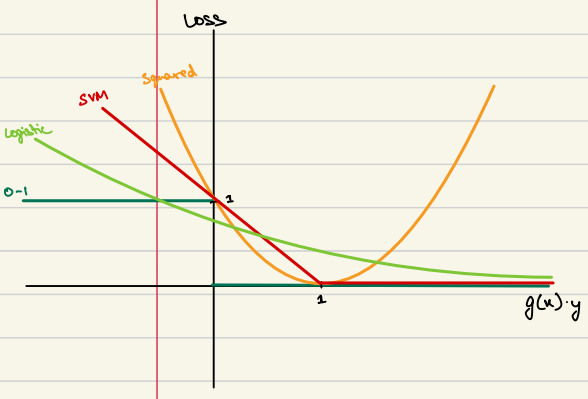
$$\min_{w, \xi} \frac{1}{2}\|w\|^2 + C \sum_{i=1}^{n} \xi_i \quad \text{s.t.} \quad (w^T x_i) \cdot y_i + \xi_i \geq 1 \text{ and } \xi_i \geq 0$$

$$\equiv \min_{w} \underbrace{\frac{1}{2}\|w\|^2}_{\substack{\text{Regulariser} \\ \text{maximise margin} \\ \text{Model Dependent}}} + C \underbrace{\sum_{i=1}^{n} \max(0, 1 - (w^T x_i) y_i)}_{\substack{g(x_i) \\ \text{Data-dependent; Loss}}}$$

→ Convex function which is also a good approximation for the 0-1 loss function.

→ A.K.A. Hinge Loss

→ SVM: Regularization + Hinge Loss



→ 0-1 loss
→ Squared loss $(g(x)\cdot y - 1)^2$
→ SVM loss $\max(0, 1 - g(x)\cdot y)$
→ Logictis loss $\log(1 + e^{-g(x)\cdot y})$

→ Logistic Regression

$$\max_w \prod_{i=1}^{n} \sigma(w^T x_i)^{z_i} \left(1 - \sigma(w^T x_i)\right)^{(1-z_i)}, \quad \text{where } \sigma(a) = \frac{1}{1+e^{-a}}$$

$$z_i = \begin{cases} 1 & \text{if } y_i = +1 \\ 0 & \text{if } y_i = -1 \end{cases}$$

Then take log ↓

$$\equiv \min_w \sum_{i=1}^{n} -z_i \log\left(\sigma(w^T x_i)\right) + (z_i - 1)\log\left(1 - \sigma(w^T x_i)\right)$$

Loss where $z_i = 1$: $\log\left(1 + e^{-w^T x_i}\right) = \log\left(1 + e^{-w^T x_i y_i}\right)$ because $y_i = 1$

$z_i = 0$: $\log\left(1 + e^{w^T x_i}\right) = \log\left(1 + e^{-w^T x_i y_i}\right)$ because $y_i = -1$

$$\equiv \min_w \sum_{i=1}^{n} \log\left(1 + e^{\overbrace{-w^T x_i y_i}^{g(x)}}\right)$$

→ Conclusion

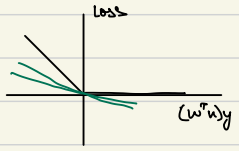
→ 0-1 loss is NP-hard.
→ Diff. algorithms use different "surrogate" loss.
→ Surrogates are convex.

→ Perceptron

$$w_{t+1} = w_t + x_t y_t$$

modified $\ell_{hinge}(w, (x,y)) = \max(0, -(w^T x)y)$

$$\nabla_w \ell_{hinge} = \begin{cases} -xy & (w^T x)y < 0 \\ 0 & (w^T x)y > 0 \\ [-1, 0]\, xy & (w^T x)y = 0 \end{cases}$$



Gradient descent : $w_{t+1} = w_t - \eta \nabla_w \text{loss}(w_t)$

Perceptron : $w_{t+1} = w_t - \underbrace{x_t y_t}_{\eta = 1}$

→ Perceptron can be interpreted as Stoghastic Gradient Descent with modified hinge loss and step size = 1

→ Boosting

$$\text{Loss}(h, (x,y)) = e^{-y h(x)}$$

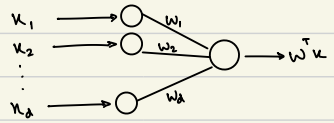
→ exponential loss

→ Neural Networks

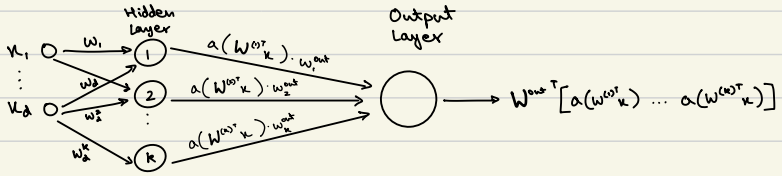
↳ Inspired by Perceptrons
Doesn't use convex loss functions

↳ $x \in \mathbb{R}^d \longrightarrow \text{sign}(w^T x)$
$[x_1, x_2, \dots, x_d]$

Perceptron:



Neural Network:



Parameters: $\{w^{(1)}, \dots, w^{(k)}\}$ $w^{(i)} \in \mathbb{R}^d$, $w^{out} \in \mathbb{R}^k$

Prediction:

$$\hat{y} = \sum_{i=1}^{k} w_i^{out}\, a(w^{(i)T}x)$$

(activation function)

↳ examples of activation function

① $a(z) = \dfrac{1}{1+e^{-z}}$ Sigmoid

② $a(z) = \max(0, z)$ Rectified linear unit

↳ Parameter estimation:

Regression : $L(NN(x_i, \theta), y_i)$ ($\theta = \{w^{(1)}, \dots, w^{(k)}, w^{out}\}$)
$= \sum_n (NN(x_i, \theta) - y_i)^2$

→ Learn θ using Gradient descent
→ Gradient computed using Back-Propagation algorithm, which uses the chain rule.
↳ converges to local minima

Classification: Cross entropy loss
↳ compares $\hat{y}_i = P(y|x_i)$ with y_i

↳ Conclusion:

→ NN learns local minima of non-concave functions
→ Typically works very well in practice, especially for unstructured data.

Week 4 GA

⑦ $$\frac{(0.054)(0.7)}{(0.242)(0.3)+(0.054)(0.7)} = 0.342$$

⑧ $$\begin{bmatrix} \lambda_1^1 & \lambda_2^1 \\ \vdots & \vdots \\ \lambda_1^4 & \lambda_2^4 \end{bmatrix} = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \end{array} \begin{array}{c} \begin{array}{cc} 1 & 2 \end{array} \\ \begin{bmatrix} 0.66 & 0.34 \\ 0.41 & 0.59 \\ 0.21 & 0.79 \\ 0.09 & 0.91 \end{bmatrix} \end{array} \qquad z = \begin{bmatrix} 1 \\ 2 \\ 2 \\ 2 \end{bmatrix}$$

⑨ $$\hat{\mu}_1 = \frac{\sum \lambda_1^i \cdot x_i}{\sum \lambda_1^i} = 1.8029$$

Week 7 GA

③ $d(x_1) = \sqrt{1+4} = \sqrt{5}$ — 1

$d(x_2) = \sqrt{16+16+36+81} = \sqrt{149}$ — 0

$d(x_3) = \sqrt{4+1+9} = \sqrt{14}$ — 1

$d(x_4) = \sqrt{1+1} = \sqrt{2}$ — 0

$d(x_5) = \sqrt{81+36+16+1} = \sqrt{134}$ — 0

Week 8 GA

⑩ $P(y=1|x) \propto P(x|y) \cdot P(y)$

$P(y=1|x)$

$= P(F_1=0|y=1) \cdot P(F_2=1|y=1) \cdot P(y=1)$

$= (0.9)(0.02)(0.3)$

$= 0.0054$

$P(y=0|x)$

$= P(F_1=0|y=0) \cdot P(F_2=1|y=0) \cdot P(y=0)$

$= (0.8)(0.3)(0.7)$

$= 0.168$