Machine Learning

Mindmap

Unsupervised Learning

= "unlabeled data"

Def: If you only have input x, you want to learn the data structure of it

Clustering

k means clustering: Flat clustering

Input = n data points $\{x_1, \cdots x_n\}$ and # of cluster = k

Output = k disjoint sets $C_1, \cdots C_k$ s.t $\bigcup_{i=1}^{k} C_i = \{x_1, \cdots x_n\}$

Define $d(x, x') = \|x - x'\| = [\sum_{i=1}^{d} (x_i - x_i')^2]^{\frac{1}{2}}$ Euclidean norm

Cost function:

① J_{avg} Average distance to center

$$J_{avg} = \sum_{j=1}^{k} \sum_{x \in C_j} d(x, m_j)$$

$$m_j = \frac{1}{|C_j|} \sum_{x \in C_j} x$$

② J_{avg^2} Average square distance to cluster center

$$J_{avg^2} = \sum_{j=1}^{k} \sum_{x \in C_j} d(x, m_j)^2$$

↑ Cluster center of C_j

③ $J_{IC} = \sum_{j=1}^{k} \frac{1}{|C_j|} \sum_{x \in C_j} \sum_{x' \in C_j} d(x, x')^2$

Sum of squared intra-cluster distance

④ $J_{max} = \max_{i=1}^{k} \max_{x \in C_i} d(x, m_i)$

Max distance to cluster center

HW1: $2 J_{avg^2} = J_{IC}$

Hints: $d(x, x')^2 = (x - x')^T (x - x')$

$d(x, m_j)^2 = (x - \frac{1}{|C_j|} \sum_{x' \in C_j} x')^T (x - \frac{1}{|C_j|} \sum_{x' \in C_j} x')$

k means Algorithm is to minimize one of the 4 cost function

Agglomerative clustering

Hierachical clustering

Mixture of Gaussians

Model based clustering

Dimension Reduction

Input = $x_1, x_2, \cdots x_n \in \mathbb{R}^d$

Output = $y_1, y_2, \cdots y_n \in \mathbb{R}^p$, $p \ll d$

linear = PCA

Nonlinear = MDS

Specific procedure = r_i denote as the cluster x_i is assigned to

$$\min J_{avg^2} = \min \sum_{j=1}^{k} \sum_{x \in C_j} d(x, m_j)^2 = \min \sum_{i=1}^{n} d(x_i, m_{r_i})^2$$

① Assign k cluster center randomly $m_1, \cdots m_k$ (First step for the whole procedure)

② Update the cluster assignment = Find the closest cluster center for each x_i and get new clusters

$$r_i \leftarrow \underset{j \in \{1, 2, \cdots k\}}{\operatorname{argmin}} d(x_i, m_j)$$

③ Update the centroids:

$$m_j = \frac{1}{|C_j|} \sum_{i: r_i = j} x_i$$

By HW1 we know the procedure can min J_{avg^2}

# of iteration $\leq n^{kd}$ ↑ degree of freedom

k means ++

Initialization: m_1 uniformly at random

Choose m_j $(j = 2, \cdots k)$ with

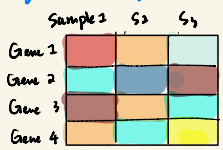
$$P(m_j = x_j) = \frac{(D_{j-1}(x_j))^2}{\sum_{i} (D_{j-1}(x_i))^2}$$

$D_{j-1}(x) = \min_{p \in \{1, \cdots j-1\}} \|x - m_p\|$

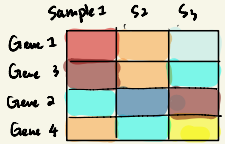
The points more far away from previous cluster centroids have higher prob to be new centroid

Hierachical Clustering

Eg. Heatmap - dendrograms



Hierachy



Dendrogram

Cluster 1 most similar
shortest branch

Cluster 3
longest branch

Cluster 2

Similarity ① "Euclidean distance"

	S2	S3
Gene 1	1.6	0.5
Gene 2	-0.5	-1.9

$\sqrt{(1.6+0.5)^2+(0.5+1.9)^2}$

↑ Difference between Gene 1 and Gene 2 in Sample 1

② "Manhanten Distance"

$\sum_{i=1}^{k}$ | Difference between Gene m and Gene n in sample i|

Input: $x_1, \cdots x_n \in \mathbb{R}^d$

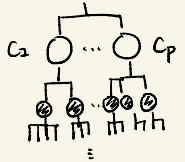
Output: A clustering tree (Dendrogram)

Advantage: No need to specify # of clusters

You can cut the tree with k = ?
or the height in your code

Divisive Algorithm

$\{x_1, \cdots x_n\}$ Start with only one cluster

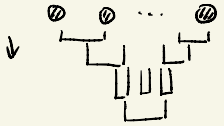


↓ then split it into smaller clusters (Recursive clustering)

Agglomerative Algorithm

① Start with n data points as n clusters

② merge clusters pairwise until they get $\{x_1, \cdots x_n\}$



Merge method =

i. Complete linkage: $d(C_i, C_j) = \max_{\substack{x \in C_i \\ x' \in C_j}} d(x, x')$

ii. Single linkage: $d(C_i, C_j) = \min_{\substack{x \in C_i \\ x' \in C_j}} d(x, x')$

iii. Average linkage: $d(C_i, C_j) = \frac{1}{|C_i|} \cdot \frac{1}{|C_j|} \cdot \sum_{x \in C_i} \sum_{x' \in C_j} d(x, x')$

Mixture of Gaussian Model

"Answering How likely is it for a datapoint to belong to a certain cluster"

Input = n data points $\{x_1, \cdots x_n\}$ and # of cluster = k

Output = k disjoint sets $C_1, \cdots C_k$ s.t $\bigcup_{i=1}^{k} C_i = \{x_1 \cdots x_n\}$

(x_i, z_i)

- Observed ith datapoint $x_i \in \mathbb{R}^d$
- $z_i \in \{1, 2, \cdots k\}$ the cluster assignment of the ith data point Hidden

★ Assumption : "Generative modeling"

$(x_i, z_i) \sim p_\theta$

(x_i, z_i) is drawn independently from some probability distribution p_θ

we model the joint distribution as $p(x, z) = p(x|z) \cdot p(z)$

$$p(x) = \sum_{z=1}^{k} p(x, z) = \sum_{z=1}^{k} p(x|z) \cdot p(z)$$

Gaussian mixture model

Given a data point $x \in \mathbb{R}^d$, $z = 1, 2, \cdots k$

$p(x, z) = \pi_z \cdot N(x|\mu_z, \Sigma_z)$,

$p(x) = \sum_{z=1}^{k} \pi_z \, N(x|\mu_z, \Sigma_z)$, where $\sum_{z=1}^{k} \pi_z = 1$

(π_z = p_z; $N(x|\mu_z, \Sigma_z)$ = $p(x|z)$)

Maximum Likelihood

Likelihood for $\{(x_1, z_1), \cdots (x_n, z_n)\}$

$$L(\theta) = \log\left(P(x_1, z_1) \cdot P(x_2, z_2) \cdots P(x_n, z_n)\right)$$

$$= \sum_{i=1}^{n} \log P(x_i, z_i)$$

$$= \sum_{i=1}^{n} \log\left[\pi_{z_i} \cdot N(x_i|\mu_{z_i}, \Sigma_{z_i})\right]$$

$$= \text{constants} + \sum_{i=1}^{n} \log \pi_{z_i} - \frac{1}{2}\sum_{i=1}^{n} \log|\Sigma_{z_i}| - \sum_{i=1}^{n} (x_i - \mu_{z_i})^T \Sigma_{z_i}^{-1} (x_i - \mu_{z_i})$$

Estimation Step =

$$\hat{\gamma}_{ik} = \frac{\hat{\pi}_k \cdot N(x_i|\hat{\mu}_k, \hat{\sigma}_k)}{\sum_{j=1}^{k} \hat{\pi}_j \, N(x_i|\hat{\mu}_j, \hat{\sigma}_j)}$$

↓

Pr(data ith belongs to kth Gaussian / cluster)

Multivariate Normal Distribution $N(x; \mu, \Sigma)$

$$\frac{1}{(2\pi)^{\frac{d}{2}} |\Sigma|^{\frac{1}{2}}} \cdot e^{-\frac{1}{2}(x-\mu)^T \cdot \Sigma^{-1} \cdot (x-\mu)}$$

Variance = Σ Mean = μ

1D: $\frac{1}{(\sigma^2)^{\frac{1}{2}}(2\pi)^{\frac{1}{2}}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} = \frac{1}{\sqrt{2\pi}\,\sigma} \cdot e^{-\frac{(x-\mu)^2}{2\sigma^2}}$

the expected log-likelihood

$$E(L(\theta)) = \sum_{j_1=1}^{k} \sum_{j_2=1}^{k} \cdots \sum_{j_n=1}^{k} P(\{(x_1, z_1 = j_1), \cdots (x_n, z_n = j_n)\}) \cdot L(\theta)$$

x_i independent and known ↓

$$= \sum_{j_1=1}^{k} \cdots \sum_{j_n=1}^{k} P(z_1 = j_1 | x_1) \cdot P(z_2 = j_2 | x_2) \cdots P(z_n = j_n | x_n) \cdot L(\theta)$$

$$= \sum_{i=1}^{n} \sum_{j=1}^{k} \gamma_{ij} \log \pi_j - \frac{1}{2}\sum_{i=1}^{n} \gamma_{ij} \log|\Sigma_j| - \frac{1}{2}\sum_{i=1}^{n} \sum_{j=1}^{k} \gamma_{ij} (x_i - \mu_j)^T \Sigma_j^{-1} (x_i - \mu_j)$$

Maximize $E(\theta)$ so we can update parameters $\hat{\pi}_k$, $\hat{\mu}_k$, $\hat{\sigma}_k^2$

By using Lagrange method =

$$\hat{\pi}_k = \sum_{i=1}^{n} \frac{\gamma_{ik}}{n}$$

= the new kth weight = the average of probabilities that a point belongs to kth Gaussian

$$\hat{\mu}_k = \frac{\sum_{i=1}^{n} \hat{\gamma}_{ik} \cdot x_i}{\sum_{i=1}^{n} \hat{\gamma}_{ik}}$$

= the new kth mean = the weighted average of all points

$$\hat{\sigma}_k^2 = \frac{\sum_{i=1}^{n} \hat{\gamma}_{ik} (x_i - \hat{\mu}_k)^2}{\sum_{i=1}^{n} \hat{\gamma}_{ik}}$$

proved in HW 1

Dimension Reduction

PCA (Linear)

= Find the most relevant subspace for the data" — dimension reduction

After normalizing the data. We want to keep their variance as much as possible in its subspace

= Variance " - Variance matrix $\hat{\Sigma} = \frac{1}{n}\sum_{i=1}^{n} x_i x_i^T$

Advantage=

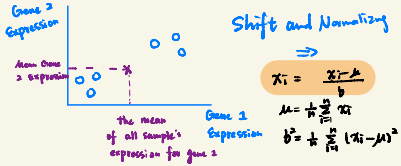
① Find the most relevant subspace for the Data

Disadvantage=

① expensive to compute

② Components are not sparse

Example in 2D



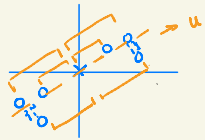
Shift and Normalizing

$\Rightarrow$

$x_i = \frac{x_i - \mu}{b}$

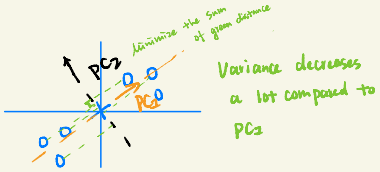
$\mu = \frac{1}{n}\sum_{i=1}^{n} x_i$

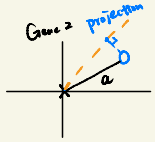
$b^2 = \frac{1}{n}\sum_{i=1}^{n} (x_i - \mu)^2$



PC1 = "Major axis of variation"

the direction on which data's variation is maximized. In other words, find a unit vector u s.t the data is projected onto this direction with largest variance.





$\min \left(\frac{1}{n}\sum_{i=1}^{n} \| x_i - (x_i)_v \|^2 \right)$

$\because \| x_i \|^2 = \| x_i - (x_i)_v \|^2 + \| (x_i)_v \|^2$

$\max \left(\frac{1}{n}\sum_{i=1}^{n} \| (x_i)_v \|^2 \right)$

How to find principal components

Maximize variance $= \frac{1}{n}\sum_{i=1}^{n} (x_i^T u)^2 = \frac{1}{n}\sum_{i=1}^{n} \langle x_i^T u, x_i^T u \rangle$

$= \frac{1}{n}\sum_{i=1}^{n} u^T x_i x_i^T u = u^T \left(\frac{1}{n}\sum_{i=1}^{n} x_i x_i^T \right) u$

$\Rightarrow$ As $\frac{1}{n}\sum_{i=1}^{n} x_i x_i^T$ is symmetric

we want $u = \text{argmax} \langle v, \hat{\Sigma} v \rangle$

$\Rightarrow u_1 = p_d$

With u_1, we can project all data points onto subspace $\text{Span}\{p_1, \cdots p_{d-1}\}$

Continuous this process, we get $u_2, \cdots u_d$

And $\hat{\Sigma} = \sum_{i=1}^{d} \lambda_i u_i u_i^T$

(Nonlinear) Multidimensional scaling (MDS)

• Classic MDS:

Input: n data points $\{x_1, x_2, x_3, \cdots, x_n\} \in \mathbb{R}^d$

Output: n lower dimensional points $y_1, y_2, \cdots y_n \in \mathbb{R}^d$

$p \ll d$ s.t

$\min \varepsilon_{CMDS} = \| D - D^* \|_{Frob}^2 = \sum_{i,j} (D_{ij} - D_{ij}^*)^2$

$D_{ij} = \| x_i - x_j \|^2$ and $D_{ij}^* = \| y_i - y_j \|^2$

Gram matrix of $\{x_1, x_2, \cdots, x_n\}$ $x_i \in \mathbb{R}^d$

$G_{ij} = \langle x_i, x_j \rangle = x_i^T x_j$

$$G = \begin{pmatrix} x_1^T \\ \vdots \\ x_n^T \end{pmatrix} (x_1, x_2 \cdots x_n) = \begin{pmatrix} x_1^T x_1 & \cdots & x_1^T x_n \\ x_2^T x_1 & \cdots & \vdots \\ \vdots & & \\ x_n^T x_1 & \cdots & x_n^T x_n \end{pmatrix}$$

i. $\text{Rank}(G) \le d$ $\quad \text{Rank}(AB) \le \min\{\text{Rank}(A), \text{Rank}(B)\}$

ii. For $k \in \mathbb{R}^{n \times n}$ symmetric positive semi-definite matrix of rank r, $r \le d$, we can find n points $x_1, x_2, \cdots x_n \in \mathbb{R}^d$ s.t $G = k$.

(HW2) Define $X =$ the first d row of $\Lambda^{\frac{1}{2}} Q^T$

$k = X^T X$

Propositions:

① $\varepsilon_{CMDS} = \| \tilde{G} - G^* \|_{Frob} = \sum (G_{ij} - G_{ij}^*)^2$

$\tilde{G}$ - Centered Gram matrix, $G_{ij} = (x_i - \mu)^T (x_j - \mu)$

G^* - Gram matrix of $\{y_1, \cdots y_n\}$

② $\tilde{G} = -\frac{1}{2} P D P$, $P = I - \frac{1}{n} 1 1^T$, $D_{ij} = \| x_i - x_j \|^2$

③ $\underset{G^*, \text{rank}(G^*) \le p}{\text{argmin}} \| \tilde{G} - G^* \|_{Frob} = Q \Lambda^* Q^T$, $\Lambda^* = \text{diag}(\lambda_1, \cdots \lambda_p, 0, \cdots 0)$

$\lambda_1 \ge \lambda_2 \ge \cdots \ge \lambda_p$

Approach:

1. Compute $\tilde{G}$ (Centered Gram matrix)
2. Compute eigendecomposition $Q \Lambda Q^T = \tilde{G}$
3. Assuming $\Lambda = \text{Diag}(\lambda_1, \lambda_2, \cdots \lambda_d)$, $\lambda_1 \ge \cdots \ge \lambda_d$

$\Lambda^* = \text{Diag}(\lambda_1, \cdots \lambda_p, 0, \cdots 0)$. $G^* = Q \Lambda^* Q^T$

4. $Y_i = [Q \Lambda^{*\frac{1}{2}}]_{i,*}^T$

Locally Linear Embedding (LLE)

Idea: Each point should be approximately reconstructed as a linear combination of its neighbors $x_i = \sum_{j \in knn(i)} w_{i,j} x_j$

$\sum_{j \in knn(i)} w_{i,j} = 1$.

Now we want to find $y_1, \dots y_n \in \mathbb{R}^p$ s.t

$y_i \approx \sum_{j \in knn(i)} w_{i,j} y_j$

Steps for this method:

1. **Find the weights for each i**

For each i, we want to minimize

$$\Phi = \| x_i - \sum_{j \in knn} w_{i,j} x_j \|^2 \quad s.t \quad \sum_{j \in knn(i)} w_{i,j} = 1$$

$$\Rightarrow \Phi = \| \sum_{j \in knn(i)} w_{i,j} (x_i - x_j) \|^2 = w^T k^{(i)} w$$

$$w = (w_j)_{j \in knn(i)} \qquad k^{(i)}_{j,j'} = (x_i - x_j)^T (x_i - x_{j'})$$

$$L(\lambda) = \Phi - \lambda \left(\sum_{j \in knn(i)} w_{i,j} - 1 \right) = w^T k^{(i)} w - \lambda (w^T 1 - 1)$$

$$\frac{\partial L}{\partial w_j} = [2 k^{(i)} w - \lambda 1]_j = 0 \quad \Rightarrow \quad w = \lambda (k^{(i)})^{-1} 1$$

$$\Rightarrow \quad w = \frac{(k^{(i)})^{-1} 1}{\| (k^{(i)})^{-1} 1 \|_1}$$

2. Find y_j's minimize $\sum_{i=1}^n \| y_i - \sum_{j \in knn(i)} w_{i,j} y_j \|^2$

s.t $\sum_{i=1}^n y_i = 0$ and $\frac{1}{n} \sum_{i=1}^n y_i y_i^T = I$ (Make Problem well-posed)

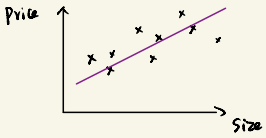
$$\Psi = \sum_{i=1}^n \| y_i - \sum_{j \in knn(i)} w_{i,j} y_j \|^2 = \sum_{i,j} M_{i,j} G_{i,j}, \quad M_{i,j} = b_{i,j} - w_{i,j} - w_{j,i} + \sum_k w_{k,i} w_{k,j}$$

Indicator function $i=j$, $b=1$

Supervised Learning

Def: Given input X with labels Y from training sets, we'd like to learn a mapping $h: X \to Y$ s.t $h(x)$ is a good predictor for new input X and the corresponding y

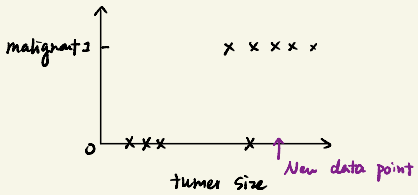
"Regression problem"



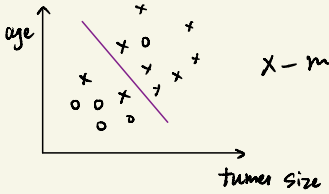
Y is continuous

Y is discrete

"Classification Problem"



Logistic Regression Algorithm



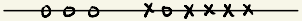
X – malignant

SVM

∞ – dimensional features

"Support vector Machine"

⇓ Projecting to 1D



"Dimension Reduction"

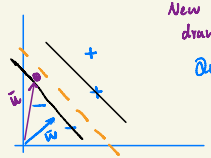
Unsupervised

As input X could be high-dimension in real life

SVM Support Vector Machine =

Idea: "separating data with a large gap"

Binary Classification:



New unknown object ❷
draw a vector from origin to ❷ as $\vec{u}$
Question: is ❷ in gap or not ?
check if $\vec{w} \cdot \vec{u} \geq c$, $c = -b$

Remark: $\vec{w} \perp$ Margin

$\Rightarrow$ Decision Rule ①
$\vec{w} \cdot \vec{u} + b \geq 0$ Then +
↑ unknown"

For all the known samples:

$\vec{w} \cdot \vec{x}_+ + b \geq 1$

$\vec{w} \cdot \vec{x}_- + b \leq -1$

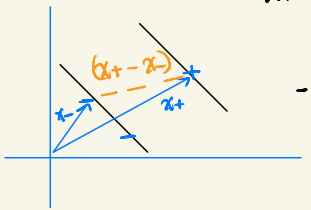
Introduce y_i s.t $y_i = +1$ For positive samples

$y_i = -1$ For negative samples

$$\left.\begin{array}{l} y_i(\vec{w}\cdot\vec{x}_+ + b) \geq 1 \\ y_i(\vec{w}\cdot\vec{x}_- + b) \geq 1 \end{array}\right\} \Rightarrow y_i(\vec{w}\cdot\vec{x}_i + b) \geq 1$$

Constraint

$y_i(\vec{w}\cdot\vec{x}_i + b) - 1 = 0$, for $\vec{x}_i$ in the cutter ②



$$\text{width} = (\vec{x}_+ - \vec{x}_-) \cdot \frac{\vec{w}}{\|\vec{w}\|}$$

↑ unit vector

$x_+ \cdot \vec{w} = 1 - b$
$-x_- \cdot \vec{w} = 1 + b$

$\Downarrow$ ∵ $y_i(\vec{x}_i \cdot \vec{w} + b) - 1 = 0$ for $\vec{x}_i$ at the margin

$$\text{width} = (1 - b + 1 + b) \cdot \frac{1}{\|\vec{w}\|} = \frac{2}{\|\vec{w}\|}$$

Maximize width $\Rightarrow$ Maximize $\frac{1}{\|\vec{w}\|} \Rightarrow \min \|\vec{w}\|$

$\Rightarrow$ $\min \frac{1}{2}\|\vec{w}\|^2$ ③

↑ mathematically convenient

$$L = \frac{1}{2}\|\vec{w}\|^2 - \sum \alpha_i \left[y_i(\vec{w}\cdot\vec{x}_i + b) - 1 \right]$$

Find the extremum:

$$\frac{\partial L}{\partial \alpha_i} = 0$$

$$\frac{\partial L}{\partial \vec{w}} = \vec{w} - \sum \alpha_i y_i \vec{x}_i = 0 \Rightarrow \boxed{\vec{w} = \sum \alpha_i y_i \vec{x}_i}$$

$\vec{w}$ is a linear combination of $\vec{x}_i$ samples

$$\frac{\partial L}{\partial b} = -\sum \alpha_i y_i = 0 \Rightarrow \boxed{\sum \alpha_i y_i = 0}$$

plug $\vec{w}$ back to L =

$$L = \frac{1}{2}\left(\sum \alpha_i y_i \vec{x}_i\right)\cdot\left(\sum \alpha_j y_j \vec{x}_j\right) - \sum \alpha_i \left[y_i\left(\sum \alpha_j y_j \vec{x}_j \cdot \vec{x}_i + b\right) - 1 \right]$$

$$L = -\frac{1}{2}\left(\sum \alpha_i y_i \vec{x}_i\right)\cdot\left(\sum \alpha_j y_j \vec{x}_j\right) - \underbrace{\sum \alpha_i y_i \cdot b}_{0} + \sum \alpha_i$$

④ $$\boxed{L = \sum \alpha_i - \frac{1}{2}\sum_i \sum_j \alpha_i \alpha_j y_i y_j \vec{x}_i \cdot \vec{x}_j}$$

Optimization only depends on dot product

Decision Rule:

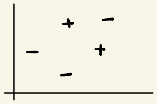
$\sum \alpha_i y_i \vec{x_i} \cdot \vec{u} + b \geq 0$ (dot product again)

i: x_i is at the margin

Remark: this question is in Convex space

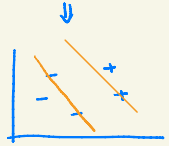
so we won't get stuck in local minimum

we get global minimum

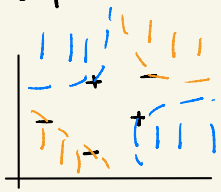


we might get stuck in this case if we stay in this space

But if we change our space we might be able to find that gap



Output



kernel + SVM

$\Leftarrow$ Inspiration = Dot product

$K(x_i, x_j) = \Phi(\vec{x_i}) \cdot \Phi(\vec{x_j})$

Inner product of x_i and x_j in another space

Kernel options:

① $(\vec{u} \cdot \vec{v} + 1)^n$

② $e^{\frac{-\|x_i - x_j\|}{b}}$

Rmk: If b is small, shrink the distance between x_i and x_j. We get overfitting

Kernel

positive semidefinite:

A kernel k is positive semidefinite on $\mathcal{X}$ =

Given a finite sequence $\{x_1, \dots x_n\}$, $x_i \in \mathcal{X}$ for $i = 1, 2, \dots n$

Kernel Matrix

define $k_{ij} = k(x_i, x_j)$, for any $f \in \mathbb{R}^n$, $f^T K f \geq 0$

$$f^T K f = \sum_j^n \sum_i^n f_j k_{ji} f_i = \sum_i^n \sum_j^n f_i k(x_i, x_j) \cdot f_j \geq 0$$

f: vector in $\mathbb{R}^n$

symmetric: $k(x, x') = k(x', x)$, for all $x, x' \in \mathcal{X}$

positive: $k(x, x') \geq 0$ for all $x, x' \in \mathcal{X}$

Example: Positive semidefinite + symmetric but not positive

$k(x, x') = x \cdot x'$, $\mathcal{X} = \mathbb{R}$, $k: \mathbb{R} \times \mathbb{R} \to \mathbb{R}$

positive semidefinite: $y^T K y = y^T x^T \cdot x y = \|xy\|^2 \geq 0$

Positive + symmetric but not positive semidefinite:

$k = \mathcal{X} \times \mathcal{X} \to \mathbb{R}$, $k(x, x') = 1 - \cos(x \cdot x')$, $\mathcal{X} = \{0, \pi\}$

$$K = \begin{bmatrix} 1-\cos(0), & 1-\cos(-\pi) \\ 1-\cos(\pi), & 1-\cos(0) \end{bmatrix} = \begin{bmatrix} 0 & 2 \\ 2 & 0 \end{bmatrix}, \quad \lambda = 2, -2$$

Proposition for psd kernel

i. If k' is a psd kernel on $\mathcal{X}'$, given any function $\varphi: \mathcal{X} \to \mathcal{X}'$

$k(x, x') = k'(\varphi(x), \varphi(x'))$, then k is psd kernel. (HW3)

ii. If k_1 and k_2 are psd kernels on $\mathcal{X}$, $K(x, x') = k_1(x, x') + k_2(x, x')$ is also psd kernel

iii. $K_\oplus((x_1, x_2), (x_1', x_2')) = k_1(x_1, x_2) + k_2(x_1', x_2')$ is psd

iv. $\alpha(x, x')$ = angle between $x, x' \in \mathbb{R}^n$, $k_\angle(x, x') = \cos\alpha(x, x')$ is psd

$k_\angle(x, x') = \frac{x \cdot x'}{\|x\| \cdot \|x'\|}$, define $\varphi(x) = \frac{x}{\|x\|}$

$k_0(\varphi(x), \varphi(x')) = k_\angle(x, x')$, by (i). We get psd.

Gaussian Process

Def: A finite collection of random variables have a joint Gaussian distribution.

$f(x) \sim GP(m(x), k(x, x'))$ iff for all $N \in \mathbb{N}$, $(f(x_1), \dots, f(x_N)) \sim N(f_\mu, K)$ with

($m(x)$: mean function; $k(x, x')$: covariance function)

$f_{\mu_i} = \mu(x_i)$ and $k_{ij} = \mathrm{cov}(f(x_i), f(x_j)) = k(x_i, x_j)$

Rmk: μ can be any function $: \mathbb{R} \to \mathbb{R}$

$k: \mathbb{R} \times \mathbb{R} \to \mathbb{R}$ is any positive semi-definite kernel on $\mathbb{R}$

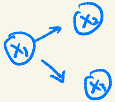
↑ Have a lot of choices for k here

$y = f(x) + \varepsilon$ "Noise function" $\varepsilon \sim N(0, b^2)$

Directed Graphical Models (Bayes nets)

Pros: Using Bayes network can significantly reduce the # of probabilities needed when describing joint distribution of $(X_1, \cdots X_N)$

Example: X_1, X_2, X_3 are all binary R.V



Without Bayesian = $2^3 - 1 = 7$

Bayes Net = $1 + 2 + 2 = 5$ ← $P(X_2=0|X_1=0)$, $P(X_2=0|X_1=1)$

↑ $P(X_1=1)$ ↑ $P(X_3=0|X_1=1)$

$P(X_3=0|X_1=0)$

X_1 \ X_2	0	1	
0			1
1			1

Markov Chain:

Def: $P(X_t | X_1, \cdots X_{t-1}, X_{t+1}, \cdots) = P(X_t | X_1, \cdots X_{t-1})$

X_t should only depend on what happened in the past

Def: "K'th order Markov Chain"

$P(X_t | X_1, X_2, \cdots X_{t-1}, X_{t+1} \cdots) = P(X_t | X_{t-1}, \cdots X_{t-k})$

A first order Markov Chain is said to be "stationery" if $P(X_t | X_{t-1})$ is independent from t, $P(X_t | X_{t-1}) = M_{X_t, X_{t-1}}$

= D-separation"

Determine which variables are independent in a Bayes net?

Bayes Net assumption: "Each variable (X) is conditionally independent of its non-descents, given its parents

(non-descents: there's no directed path starting from X to =)

Step: ① Draw ancestral graph

(all nodes in X, all ancestor of any node in X and all edges between those nodes)

② "Moralize" the parents

(Each pair of variables with a common children draw line between them)

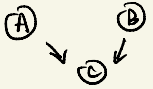
③ "Disorient" the graph (replacing "→" with "—")

④ Delete given and their edges

Multiplication Rule:

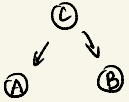
$P(A_1 \cap \cdots \cap A_N) = P(A_1) \cdot P(A_2|A_1) \cdot P(A_3 | A_1 \cap A_2)$

"V structure"



$$P(X_A, X_B, X_C) = P(X_A, X_C) \cdot P(X_B, X_C) = P(X_A) \cdot P(X_C \mid X_A) \cdot P(X_B) \cdot P(X_C \mid X_B)$$

"Inverse V structure"



$$P(X_A, X_B, X_C) = P(X_C) \cdot P(X_A \mid X_C) \cdot P(X_B \mid X_C)$$

General form of discrete Bayes net:

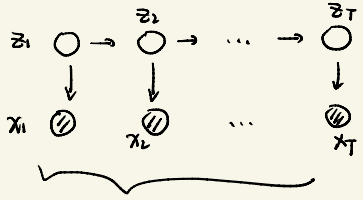
$$P(x) = \prod_{v \in V} P(x_v \mid x_{pa(v)})$$

Simpson's Paradox:

1. A graphical model can't capture all variables that might possibly relevant.
2. We can't know the causal relationship based on observational study. We need randomalized controlled trials.

HMM Hidden Markov Model:

A First-order discrete HMM:



"z_t" Hidden variable at time t

Transition Model $p(z_t \mid z_{t-1})$

Observation Model $p(x_t \mid z_t)$

Discrete HMM (General)

$HMM = \{N, M, A, B, \pi\}$

A: transition matrix = Hidden

Store transition probability from state i to state j

every time step, a robert must move from one state to another (it's okay to stay in place)

B: A vector of seeing the different observations given you're in a particular state

π: probability of starting in a particular state and then moving through the whole states and making observations from time 1 to time T

Questions HMM try to answer:

① What's the probability that a model generated a sequence of observations $O = \{O_1, \dots O_T\}$ given $\lambda = (A, B, \pi)$

② What sequence of states best explain $O = \{O_2, \dots, O_T\}$?

Find $Q = \{q_2, q_2, \dots q_T\}$?

★ ③ Given $O = \{O_2, \dots O_T\}$ Baum-Welch algorithm

How do we learn parameters that would generate them ?

$\lambda = (A, B, \pi)$?

Forward Parameter: What came in the past

$\alpha_t(i) = P(O_1, \dots, O_t, q_t = S_i \mid \lambda)$

All observations capture from time 1 to t and we're at S_i state, given λ

Backward Parameter: What'll come in the future

$\beta_t(i) = P(O_{t+1}, O_{t+2}, \dots O_T \mid q_t = S_i, \lambda)$

All observations after time t given current state is S_i and λ

$\gamma_t(i) = P(q_t = S_i \mid O, \lambda)$

given we know what comes before and after t

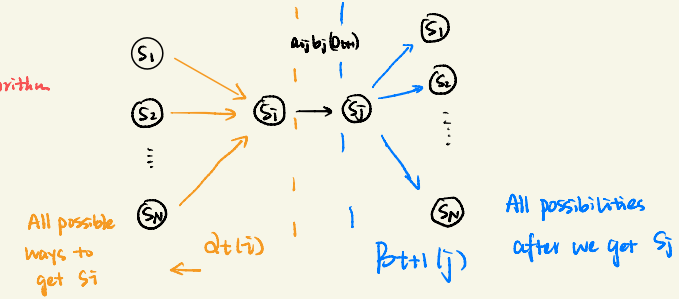
Baum-Welch Algorithm (find local optimal)

= "Gradient descent algorithm"

Mathematics tool:

$\xi_t(i,j) = P(q_t = S_i, q_{t+1} = S_j \mid O, \lambda)$

Prob of transition from S_i to S_j at time t given all observations and λ



$$\xi_t(i,j) = \frac{\alpha_t(i) \cdot a_{ij} b_j O(t+1) \beta_{t+1}(j)}{P(O \mid \lambda)}$$

$$= \frac{\alpha_t(i) \cdot a_{ij} b_j O(t+1) \beta_{t+1}(j)}{\sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_t(i) \cdot a_{ij} b_j O(t+1) \beta_{t+1}(j)}$$

$$\gamma_t(i) = \sum_{j=1}^{N} \xi_t(i,j)$$

$$E(\#\text{ of times } S_i \text{ is visited}) = \sum_{t=1}^{T-1} \gamma_t(i) \cdot 1$$

$$E(\#\text{ of transition from } S_i \text{ to } S_j) = \sum_{t=1}^{T-1} \xi_t(i,j)$$

$\lambda = (A, B, \pi)$

Data Set = O

tools: $\alpha_t(i)$, $\beta_t(i)$, $\gamma_t(i)$, $\xi_t(i,j)$

$\bar{\lambda}$ = ?

$$\bar{a}_{ij} = \frac{\text{Expected number of transition from } S_i \text{ to } S_j}{\text{Expected number of transition from } S_i}$$

$$= \frac{\sum_{t=1}^{T-1} \xi_t(i,j)}{\sum_{t=1}^{T-1} \gamma_t(i)}$$

$$\bar{b}_j(k) = \frac{\text{Expected \# of times in state } j \text{ observing } v_k}{\text{Expected \# of times in state } j}$$

$$= \frac{\sum_{t=1, \text{ s.t. } O_t = v_k}^{T} \gamma_t(i)}{\sum_{t=1}^{T} \gamma_t(i)}$$

Given $\alpha_t(i)$, $\beta_t(i)$, $\gamma_t(i)$, $\xi_t(i,j)$

we can produce $\bar{\lambda} = (\bar{A}, \bar{B}, \bar{\pi})$

⇒ "iterative steps"

Undirected Graphical Models (Markov Random Fields)

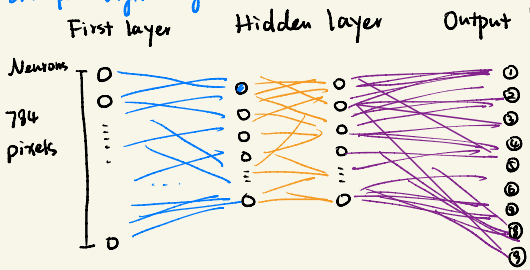
Def: the general form of joint distribution

$$p(x) = \frac{1}{Z} \cdot \prod_{C \in \text{Clique}(G)} \phi_C(x_C)$$

- ϕ_C = clique potential (A positive function)
- Z = normalizing factor s.t $\sum_x \frac{1}{Z} \prod_{C \in \text{Clique}(G)} \phi_C(x_C) = 1$

Neuro Network

Example: Digit Recognition



Neuron - Function that takes in input to a number (= "Activation")

Activation functions:

① Sigmoid function: $\sigma(x) = \frac{1}{1+e^{-x}} \in [0,1]$

$a = \sigma\left(\sum_i w_i a_i + b\right)$

- a: A point on the next layer
- w_i: points from current layers
- b: bias

b: only activate meaningfully when weighted sum > b

From one layer to the next layer

$$\sigma\left(\begin{bmatrix} w_{0,0} & w_{0,1} & \cdots & w_{0,n} \\ \vdots & & & \\ w_{k,0} & & \cdots & w_{k,n} \end{bmatrix}\begin{bmatrix} a_0^0 \\ a_1^0 \\ \vdots \\ a_n^0 \end{bmatrix} + \begin{bmatrix} b_0 \\ b_1 \\ \vdots \\ b_n \end{bmatrix}\right)$$

$\sigma(Wa^0 + b)$

We have $784 \times 16 + 16 \times 16 + 16 \times 10$ (weights) $+ 16 \times 2 + 10$ (bias) $= 13002$ numbers for weights and bias.

Learning means to find right weights and bias

Cost function / Loss function - Machine learning Minimizes Cost function

i. Feed-forward NN for regression

$f: \mathbb{R}^n \to \mathbb{R}^m$, Output Y is continuous

for one example (x, y),

$\varepsilon(\hat{y}, y) = \frac{1}{2}\sum_{i=1}^{m} (a_{z(i)} - y_i)^2$,

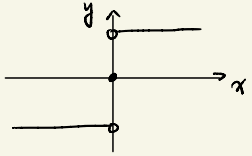
$a_{z(i)}$ - the output of ith neuron in the output layers

ii. Feed-forward NN for classification (Y is discrete)

$f: \mathbb{R}^n \to \{1, 2, \ldots, k\}$, $\varepsilon(\hat{y}, y) = -\log\left(\frac{e^{a_{z(i)}}}{\sum_{i=1}^{k} e^{a_{z(i)}}}\right)$

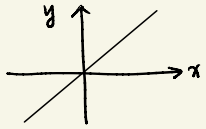
② $\mathrm{ReLu}(a) = \max(0, a)$

③ Hard Shreshold $b(t) = \mathrm{Sgn}(t)$



Problem: Not differentiable

④ Linear: $b(t) = t$ "identity function"



Problem:

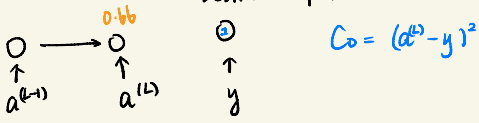
1. It's not possible to use back propagation as derivation is 1.
2. No matter the number of layers in the neural network, a linear activation function turns it into 1 layer.

Backpropagation

(Gradient Descent Method)

- Stochastic gradient descent
  Subdivide data into "mini batches"
  and compute each step by the gradient
  of the mini batch

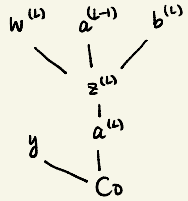
Imagine only one neuron for each layer



$C_0 = (a^{(L)} - y)^2$

$a^{(L)} = b(w^{(L)} a^{(L-1)} + b^{(L)})$

let $z^{(L)} = w^{(L)} a^{(L-1)} + b^{(L)}$, then $a^{(L)} = b(z^{(L)})$



$\frac{\partial C_0}{\partial w^{(L)}}$ = the sensitivity of cost function regarding small change in weights

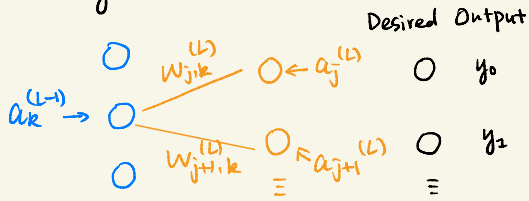
$$\frac{\partial C_0}{\partial w^{(L)}} = \frac{\partial z^{(L)}}{\partial w^{(L)}} \cdot \frac{\partial a^{(L)}}{\partial z^{(L)}} \cdot \frac{\partial C_0}{\partial a^{(L)}}$$

↑ Chain Rule

$$\frac{\partial C_0}{\partial w^{(L)}} = a^{(L-1)} \cdot b'(z^{(L)}) \cdot 2(a^{(L)} - y)$$

$$\frac{\partial C_0}{\partial a^{(L-1)}} = \frac{\partial z^{(L)}}{\partial a^{(L-1)}} \cdot \frac{\partial a^{(L)}}{\partial z^{(L)}} \cdot \frac{\partial C_0}{\partial a^{(L)}} = w^{(L)} b'(z^{(L)}) 2(a^{(L)} - y)$$

Imagine multiple neuron from one layer to next:



$$C_0 = \sum_{j=0}^{n_L - 1} (a_j^{(L)} - y_j)^2$$

$$z_j^{(L)} = \sum_{k}^{n_{L-1}-1} w_{j,k}^{(L)} a_k^{(L-1)}$$

$$a_j^{(L)} = b(z_j^{(L)})$$

$$\frac{\partial C_0}{\partial w_{j,k}^{(L)}} = \frac{\partial z_j^{(L)}}{\partial w_{j,k}^{(L)}} \cdot \frac{\partial a_j^{(L)}}{\partial z_j^{(L)}} \cdot \frac{\partial C_0}{\partial a_j^{(L)}}$$

$$\frac{\partial C_0}{\partial a_k^{(L-1)}} = \sum_{j=0}^{n_L - 1} \frac{\partial z_j^{(L)}}{\partial a_k^{(L-1)}} \cdot \frac{\partial a_j^{(L)}}{\partial z_j^{(L)}} \cdot \frac{\partial C_0}{\partial a_j^{(L)}}$$

Numerical Differenciation

(Finite difference — Taylor Expansion)

$$\frac{\partial f}{\partial x_i} \approx \frac{f(x+he_i)-f(x)}{h}$$

e_i: i-th unit vector; h: step size

Requires $O(n)$ evaluations : $x_1 \cdots x_n$

* Symbolic differentiation: SymPy

$f: \mathbb{R}^n \to \mathbb{R}^m$

$$Jf = \left[\frac{\partial f}{\partial x_1} \cdots \frac{\partial f}{\partial x_n}\right] = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \vdots & & & \\ \frac{\partial f_m}{\partial x_1} & \frac{\partial f_m}{\partial x_2} & \cdots & \frac{\partial f_m}{\partial x_n} \end{bmatrix}$$

= Jacobian Matrix

First-Order Matrix $= \begin{bmatrix} \nabla^T f_1 \\ \vdots \\ \nabla^T f_m \end{bmatrix}$

When $n \ll m$ Forward mode is idea

Remark: Second-Order Matrix — = Hessian Matrix

Automatic differentiation:

Chain Rule: $y = f(g(h(x)))$, $w_0 = x$, $w_1 = h(w_0)$, $w_2 = g(w_1)$, $w_3 = f(w_2)$

$$\frac{\partial y}{\partial x} = \frac{\partial y}{\partial w_2}\cdot\frac{\partial w_2}{\partial w_1}\cdot\frac{\partial w_1}{\partial w_0} = \frac{\partial f(w_2)}{\partial w_2}\cdot\frac{\partial g(w_1)}{\partial w_1}\cdot\frac{\partial h(x)}{\partial x}$$

Forward Accumulation computes $\frac{\partial w_i}{\partial x} = \frac{\partial w_i}{\partial w_{i-1}}\cdot\frac{\partial w_{i-1}}{\partial x}$ with $w_0 = x$ — "$i-1$" → "i"

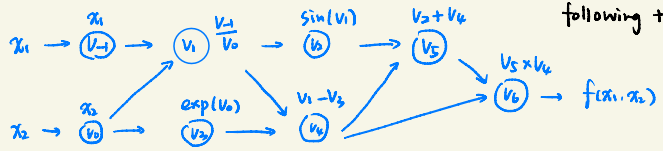
Seed: $\frac{\partial x}{\partial x} = 1$

Reverse Accumulation computes $\frac{\partial y}{\partial w_i} = \frac{\partial y}{\partial w_{i+1}}\cdot\frac{\partial w_{i+1}}{\partial w_i}$ with $w_0 = x$ — "$i+1$" → "i"

Seed: $\frac{\partial y}{\partial y} = 1$

2-D case (Forward Accumulation)

eg. $f(x_1, x_2) = \left[\sin\left(\frac{x_1}{x_2}\right) + \frac{x_1}{x_2} - e^{x_2}\right] \times \left[\frac{x_1}{x_2} - e^{x_2}\right]$



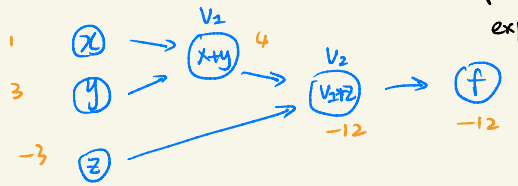
We can compute $\frac{\partial f}{\partial x_1}$ by following the map

$$\frac{\partial v_{-1}}{\partial x_1} = 1 \qquad \frac{\partial v_1}{\partial x_1} = \frac{\partial v_1}{\partial v_{-1}}\cdot\frac{\partial v_{-1}}{\partial x_1} = \frac{1}{v_0}\cdot 1$$

$$\frac{\partial v_2}{\partial x_1} = \frac{\partial v_2}{\partial v_1}\cdot\frac{\partial v_1}{\partial x_1} = \cos(v_1)\cdot\frac{1}{v_0} \quad \cdots$$

Simple example: $(x+y)*z = f$

Forward Pass (Orange Path): find the value for mathematical expression.



Backward Pass (To compute the gradient)

$$\frac{\partial v_1}{\partial x} = 1 \qquad \frac{\partial v_2}{\partial x_1} = \frac{\partial v_2}{\partial v_1}\cdot\frac{\partial v_1}{\partial x_1} = z\cdot 1 = z$$

$$\frac{\partial v_2}{\partial y} = 1 \qquad \frac{\partial v_2}{\partial z} = z \qquad \frac{\partial v_2}{\partial z} = v_1 = x+y$$