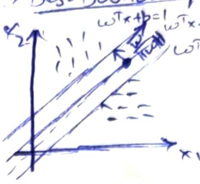
Spectral Clustering (Graph based) – cosine distance, inverse Euclidean dist.
From similarity matrix find Adjacency matrix (W) with k nearest neigh.
Obj – $cut(A_1, A_2, \dots A_k) \triangleq \frac{1}{2}\sum_{k=1}^{K} W(A_k, \bar{A}_k) \Rightarrow W(A,B) \triangleq \sum_{i\in A, j\in B} W_{ij}$
→ This is NP hard problem (No optimal solution)
→ Graph Laplacian → $L \triangleq D - W$ (∴ D – Diagonal matrix [sum of each row in W]
→ D represents degree of corresponding Node.
Properties (L): ① each row sums to zero ② 1 eigen vector with eigen value 0 → constant ③ Symmetric and positive semi-definite, ④ Has N nonnegative eigen value vectors ⑤ K comp then L has k eigen vector with eigen values '0'
K-Means:- Revise cluster centers using centroid or medoid (more time needed) using Euclidean provide circular clusters. Use Trail and error to set K
obj ⇒ $J(C,R) = \sum_{n=1}^{N}\sum_{k=1}^{K} R_{nk}\|x_n - c_k\|^2$ (∴ $R_{nk} = \begin{cases}1 & \text{if } x_n \in C_k \\ 0 & \text{otherwise}\end{cases}$)
→ very sensitive to initialization. {choose random, farthest, second farthest}
Strengths:- Simple, can be extended to other type of data, Easy to parallelise
Weakness:- Circular clusters, Choose k, Not guaranteed to be optimal,
hard clustering:- Assign every data point to exactly one cluster.
→ for HC probability will be 1 for one cluster 0 for other's.
PCA – Dimentionality Reduction (loss of information (minimize this))
Why DR:- Reduce data (storage, compute, transport) | Visualization. | improves data (important features, address correlation, remove noice)
Other algo do DR by preserving ① Distance b/w two points in dataset ② Nearest Neighbor property
PCA tries to preserve the variance of the data.
→ Project data in maximal variance direction. (I want a unit vector u such that when I project along unit vector my variance should be maximised. For PCA we need data to be mean centered. for this you have to substract mean for each row to make it mean centered.
variance $= \frac{1}{N}\sum_{i=1}^{N}(x_i^T\hat{u})^2$ (variance of reduced data along u) $S = \frac{1}{N}\sum_{i=1}^{N}(x_i - u)(x_i - u)^T$ $\boxed{S = \frac{1}{N}\sum_{i=1}^{N} x_i x_i^T}$
S → covariance matrix, use legrangian multiplier (λ) (Find eigen value, vector of S)
$Su = \lambda u$ (u eigen vector of S)
(λ value) → The best embedding will be the eigen vector with largest eigen-value.
obj fun ⇒ $\hat{u}^T\lambda u$ or $\hat{u}^T S\hat{u}$
First PC:- Eigen vector of the (S) with largest eigen-value.
Second PC:- ↳ orthogonal to first one.
variance of each PC is given by λ_i | Variance captured by PC $\frac{\sum_{i=1}^{L}\lambda_i}{\sum_{i=1}^{D}\lambda_i} \times 100$, $1 \le L \le D$
Reinforcement learning:- An agent, An environment, The environment has a state and the environment will give reward on the action taken by the agent to change the state. Agent will always tries to improve the reward.
Ex:- Playing chess, Go, Robot learning to walk, Every day activity, Controller adjustment Parameters of an engineered system (~~~~) (oil refinery) real time.
Exploitation:- learning from prev experiences and using that to the advantage to maximize the reward to work properly in this case.
Exploration:- First tasting the unknown and then improving the performance to that new environments so that you can actually leverage that and exploit that in future. (Trade of b/w Expolitation and Exploration)
→ Agent operating in an env, state of the agent at time t – $S_t \in S$, Action taken by agent at time t – $A_t \in A(S_t)$, Reward at time t – $R_t \in R$, Policy – π (descion making rules)
π(s) : S → A (Action at a given state). | RL wants to learn a policy Optimum policy – we try to take actions that will maximize reward
Goal of RL:- Learn optimal policy π that maximizes the reward
① Policy, ② Reward signal – used by the env to inform the agent of the reward at a given time step (current state and action) ③ Value function:- Expected total reward stating at a given state [hard to determine, long term desirability of a state), (A state might have a small reward but a high value – might be followed by a sequence of states with high reward ④ Model – Allows us to model the env (predict next steps and next rewards)] learning
– Using the value of current state to update the value of prev state:-
$V(S_t) \leftarrow V(S_t) + \alpha[V(S_{t+1}) - V(S_t)]$. α step size parameters
NN are combined with RL to handle high dimension state action spaces, approximate function like Q-Values or policies and generalize across states, enabling RL to tackle complex real world problems efficiently.
Difference from evolutionary:- operates at a policy level, Evaluates a policy over many games and then chooses the next policy, for each game, only the final out come is considered. Value function learning:- Evaluates individual states during the course of play.
MDP:- The critical assumption is the descion that we take at the current step is only influenced by the action we have taken in the prev. MDP:- The Mathematical Idealized formulation of the RL problem. [two types of learning ① Episodic ② continuing.]
p defines the dynamic fun of the MDP:- A four arguments :- $S \times R \times S \times A \to [0,1]$
Episodic reward:- save the reward for a seq of actions that machine took in response to some of states. Expected return with discount factor
$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \dots R_T$ | γ=0 maximizing imediate reward (short sighted) γ=1 more weight to future.

→ if y is linked ... , then it is Linear; it is Robust Regression. Linear Regression is susceptable to o...
→ Logistic Regression:- if $\frac{1}{1+e^{-(w^Tx)}} > 0.5$, then y=1 (or) y=0
$\frac{1}{1+e^{-(w^Tx)}} > \frac{1}{2} \Rightarrow 2 > 1 + e^{-(w^Tx)} \Rightarrow 1 > e^{-(w^Tx)} \Rightarrow 0 > -w^Tx$ $\boxed{w^Tx > 0}$
• Perceptron if $w^Tx > 0$, then y = 1 else y = 0
→ LR, P, SVM have same decision function, different learning strategy. $\boxed{\text{Decision function in SVM } y = w^Tx + b}$
→ Best Boundary:- Boundary that seperates the data the most



→ Size of the Margin $= \frac{2}{\|w\|}$.
→ we want to learn w, b such that:
⇒ for all points x_i for which $y_i = 1$, $w^Tx + b > 0$
⇒ for all points x_i for which $y_i = -1$, $w^Tx + b < 0$
⇒ $\frac{2}{\|w\|}$ is as large as possible.
→ The distance from the origin to the Margin $= \frac{b}{\|w\|}$
→ Missclassification:- To create a threshold not sensitive to outliers we use missclassification.
→ Soft margin:- The threshold which is determined after considering missclassification. Soft Margin is also called SVC.
→ SVC can handle outliers. And can handle overlapping data.
→ Kernel function:- This helps to choose (or) draw SVC at a higher dimension.
→ Kernel Trick:- compare relationships b/w observations as they would appear in higher dimension without taking them to higher dimension.
→ Polynomial kernel function:- This contains a parameter degree which will control the complexity of the non-linear decision boundary.
→ Minimize $f(x,y) = 2 - x^2 - 2y^2$, ST $h(x,y) = x + y - 1 = 0$. Here we apply Lagrangian multiplier (β) ⇒ $\min_{x,y,\beta} L(x,y,\beta) = f(x,y) + \beta h(x,y)$
→ for multiple constraint $\min_{x,y,\beta} L(x,y,\beta) = f(x,y) + \sum_i \beta_i h_i(x,y)$
$\frac{\partial L}{\partial x} = 0 \Rightarrow -2x + \beta = 0$, $\frac{\partial L}{\partial y} = 0 \Rightarrow -4y + \beta = 0$, $\frac{\partial L}{\partial \beta} = 0 \Rightarrow x + y - 1 = 0$
$x = \frac{\beta}{2}$ $y = \beta/4$ $\frac{3\beta}{4} = 1 \Rightarrow \beta = \frac{4}{3}$
$\boxed{x = \frac{4}{6},\ y = \frac{4}{12}}$
→ Inequality Contraints = $\min_{x,y} f(x,y) = x^3 + y^3$ ST $g(x) = x^2 - 1 \le 0$. $\boxed{L(x,y,\alpha) = f(x,y) + \alpha g(x)}$
→ Handling both constraints: minimize $f(w)$, ST $g_i(w) \le 0$, $h_i(w) = 0$
$L(w, \alpha, \beta) = f(w) + \sum_{i=1}^{k}\alpha_i g_i(w) + \sum_{i=1}^{l}\beta_i h_i(w)$ ST $\alpha_i \ge 0$
→ Primal optimization:- $\theta_p(w) = \max_{\alpha,\beta,\alpha_i \ge 0} L(w,\alpha,\beta)$
$p^* = \min_w \theta_p(w) = \min_w \max_{\alpha,\beta,\alpha_i\ge 0} L(w,\alpha,\beta)$. p^* solving is equivalent to solving the constrained optimization problem.
→ Dual Optimization:- $\theta_d(\alpha,\beta) = \min_w L(w,\alpha,\beta) \Rightarrow d^* = \max_{\alpha,\beta,\alpha_i \ge 0} \theta_d(\alpha,\beta)$
$d^* = \max_{\alpha,\beta,\alpha_i\ge 0}\min_w L(w,\alpha,\beta)$. $\boxed{d^* \le p^* \text{ weak duality Theorem}}$
→ Advs of Dual Optimization: 1) Easier to solve than other optim problem 2) Determines some interesting facts about SVM.
→ $d^* == p^*$ ⇒ strong duality, when 1) Data is convex 2) constraints are affined. → if $d^* - p^*$ is very small then dual can be used as a proxy for primal.
→ KKT conditions: for $d^* == p^* = L(w^*, \alpha^*, \beta^*)$:⇒ $\frac{\partial}{\partial w}L(w^*,\alpha^*,\beta^*) = 0$, $\frac{\partial}{\partial \beta}L(w^*,\alpha^*,\beta^*) = 0$, $\alpha_i^* g_i(w^*) = 0$, $g_i(w^*) \le 0$, $\alpha_i^* \ge 0$
→ $\min_w f(w) = \frac{\|w\|^2}{2} = \frac{w_1^2 + w_2^2}{2}$ ST $y_1(w^Tx_1 + b) - 1 \ge 0$ $x_1 = 1,1$ $y_1 = -1$
$y_2(w^Tx_2 + b) - 1 \ge 0$ $x_2 = 2,2$ $y_2 = 1$
⇒ $-(w^Tx_1 + b) - 1 \ge 0$, $(w^Tx_2 + b) - 1 \ge 0$ ⇒ $-w_1 - w_2 - b - 1 \ge 0$
$2w_1 + 2w_2 + b - 1 \ge 0$ ⇒ $L = \frac{w_1^2 + w_2^2}{2} + \alpha_1(-w_1 - w_2 - b - 1) + \alpha_2(2w_1 + 2w_2 + b - 1)$
$\frac{\partial L}{\partial w_1} = w_1 - \alpha_1 + 2\alpha_2 = 0$ $\frac{\partial L}{\partial \alpha_1} = -w_1 - w_2 - b - 1 = 0$ $\boxed{w_1 = w_2 = 1,\ b = -3}$
$\frac{\partial L}{\partial w_2} = w_2 - \alpha_1 + 2\alpha_2 = 0$ $\frac{\partial L}{\partial \alpha_2} = 2w_1 + 2w_2 + b - 1 = 0$

→ Posterior ∝ Likelihood × Prior | Best hypothesis = maximum likelihood method

⇒ $2\times(2^d-1)$ parameters for two classes

→ All features are independent and each variable can be assumed to be a Bernoulli random variable.

→ we reduce to 2D parameters

→ $P(Y=1|X=x^*) = \dfrac{\prod P(x=x^*|Y=1)\,P(Y=1)}{P(x=x^*|Y=1)\,P(Y=1) + P(x=x^*|Y=0)\,P(Y=0)}$

$\theta_{01} = P(X_i=1|Y=0)$

θ_{10}

$\theta_{00} = 1-\theta_{11}$

separable case: 0 training loss, $y_n(w^Tx_n+b)\geq 1$

review:- Non separable: Relax Constraints - $y_n(w^Tx_n+b)\geq 1-\epsilon_n$ (slack variable)

1) role of a prior:- a prior represents the initial belief about the probabilities of classes before considering any evidence from the data. It helps incorporate prior knowledge and serves as a baseline for updating probabilities using Baye's theorem. It make sure that current probabilities will not have full control on final outcome

2) limitation: It assume feature independence, which rarely happen in real world. 2) Perform poorly on small datasets due to unreliable probability estimates 3) sensitive to noise data and irrelevant features

softmax map ent classifier

3) Bernoulli:- because it simplifies classification into binary outcomes, aligning with the algorithmic assumption of independent feature contribution. This make it computationally efficient and well suited for binary classification tasks.

overfitting, add prior

$H' = H - \lambda I$

for multiclasses, we multinoulli

4) conditionally independent → $2(2^d-1)$

totally → $2(d)$

5) why only 2D: we assume features independence, this ~~avoids~~ avoids the need to account for feature interaction. with d features and 2 classes, we only estimate d parameters per class, resulting in 2D

→ More N → More D

Hessian is the second order derivative of object function $H = -\sum_i \theta_i(1-\theta_i)x_ix_i^T$

iid → independent, identical

Linear regression

Probabilistic Interpretation:- y is assumed to be normally distributed

'C' controls the impact of margin and margin error

$y \sim N(w^Tx, \sigma^2)$, random variable, mean, variance

(w, σ^2) → unknown parameters

$P(Y|X) = N(w^TX, \sigma^2)$

loglikelihood or MLE (maximum)

$w = (X^TX)^{-1}X^Ty$

$\hat{\sigma}^2 = \frac{1}{N}(y-Xw)^T(y-Xw)$

Geometric:

$w \leftarrow w + \eta H^{-1}\nabla$ (Hessian)

$y = w^Tx = w_1x_1 + w_2x_2 + \dots w_dx_d$

$y = w_0 + w_1x_1$ (bias)

$x \to [1\ x]$

$X \to N \times (d+1)$

$y \to N \times 1$

minimize squared loss:

Least squares estimate: $J(w) = \frac{1}{2}\sum_{i=1}^{N}(y_i - w^Tx_i)^2$

Gradient descent:

$\frac{\partial J(w)}{\partial w_j} = \sum_{i=1}^{N}(w^Tx_i - y_i)x_{ij}$ newton method

$w_j = w_j - \alpha\frac{\partial J(w)}{\partial w_j}$

setting η is sometimes tricky: too large, wrong results, too small, slow convergence

issues:- 1) susceptible to outliers 2) too simplistic - underfitting 3) Unstable in presence of correlated input attributes

Bayesian linear regression

we assume a prior on 'w'

→ multivariate gaussian is an appropriate prior.

$P(w|D) = \frac{P(D|w)P(w)}{\text{total}}$ → prior, likelihood

$\frac{d}{dw}LL(w) = \sum_{i=1}^{N}(y_i - \theta_i)x_i$

$w_{k+1} = w_k + \eta\frac{d}{dw_k}LL(w_k)$

diff ways to predict y^* given x^*:-

1) MLE: $P(y^*|x^*) = N(w^T_{MLE}x^*, \sigma^2_{MLE})$

2) for MAP: $P(y^*|x^*) = N(w^T_{map}x^*, \sigma^2_{MLE})$

3) full bayesian treatment: $P(y^*|x^*) = \int_{\forall w} N(w^Tx^*, \sigma^2_{MLE})\,P(w|D)\,dw$

log likelihood: $LL(w) = \sum_{i=1}^{N} y_i\log\theta_i + (1-y_i)\log(1-\theta_i)$

→ impact of prior on w:-

1) Penalizes large values of w 2) translate to a zero mean gaussian prior

likelihood:

$y_i = 1, \theta_i = \frac{1}{1+\exp(-w^Tx)}$

$y_i = 0, 1-\theta_i = \frac{1}{1+\exp(w^Tx)}$

$P(y_i) = \theta_i^{y_i}(1-\theta_i)^{1-y_i}$

for non linear curves:- $\phi(x) \to x^2$

basis function → $\phi(x) = [1, x, x^2, \dots x^d]$

prior is Gaussian

Ridge regression: controls overfitting

$\hat{w} = \arg\min J(w) + \alpha\|w\|^2$

$J(w) = \frac{1}{2}\sum_{i=1}^{N}(y_i - w^Tx_i)^2 + \frac{1}{2}\lambda\|w\|_2^2$

$w = (\lambda I_D + X^TX)^{-1}X^Ty$

L2-norm, L2 regularization

Optim for Non separable case: maximize $f(w,b) = \|w\|^2 + C\sum_{n=1}^{N}\epsilon_n$

how to control overfitting:-

1) Use simpler models (might lead to underfitting)

2) Use regularized complex models

$\hat{\theta} = \arg\min J(\theta) + \lambda R(\theta)$ → penalty paid for complexity of model

→ Ridge helps in reducing impact of correlated inputs

Lasso:- $\hat{w} = \arg\min J(w) + \lambda|w|$ — least absolute shrinkage and selection operator; L1 norm

1) Helps in feature selection, inducing sparsity

2) some of the features be zero and some non zero from which we can ignore zero, that is feature selection

VC: analysis of how well a line fits various datasets

→ prior is laplace distribution: $P(w) = \frac{1}{2b}\exp\left(-\frac{|w-\mu|}{b}\right)$

review:-

1) why cannot $y = w^Tx$: because this form assume linear relation b/w inputs & outputs and also forces the line to pass through origin. many models include an intercept to allow flexibility ~~and also involve~~

2) diff of class and regre:-

class:- output types discrete categories or labels. objective is assign labels to data points.

regr:- output is continuous numerical values, objective is predict values by fitting a function

weight L1: $y \sim laplace(w^Tx, b)$

robust regression → handles outliers better

$J(w) = \sum_{i=1}^{N}|y_i - w^Tx|$

why robust better for outliers:- This doesn't have square in the loss function, hence impact of error can be minimized. The errors caused by outliers do not have impact

$y = \text{Bernoulli}(\theta)$, $\theta \neq w^Tx$, $0<\theta<1$

sigmoid $= \frac{1}{1+e^{-z}}$

$y = \text{Bernoulli}\left(\frac{1}{1+e^{-(w^Tx)}}\right)$

Generative model

Discriminative model better as these need lesser training examples for PAC learning