Knn:-

L_p

$$L_2 = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$$

$$L_1 = |x_1 - x_2| + |y_1 - y_2|$$

$$L_\infty = \max(|x_1 - x_2|, |y_1 - y_2|)$$

Scaling

Standard $= \frac{x_i - \mu}{\sigma}$

maximum $= \frac{x_i}{max}$

max-min $= \frac{x_i - min}{max - min}$

is it metric?

$d_{p,p} = 0$ $\qquad$ $d_{p,q} \geq 0$

$d_{p,q} = d_{q,p}$ $\qquad$ $d_{p,q} + d_{q,r} \geq d_{p,r}$

Linear Regression:-

predict:

$$\hat{y} = w_1 x_1 + w_2 x_2 + b$$

RSS:

$$Loss_i = (\hat{y}_i - y_i)^2$$

$$J = \frac{1}{2m} \sum (\hat{y} - y)^2$$

Closed-form:

$$J = (Xw - y)^T (Xw - y)$$

$$w = (X^T X)^{-1} X^T y$$

Gradiant desent:-

step 1: initialize w, b

step 2: calculate $\hat{y}$ and cost → RSS

step 3: calculate dw, db

step 4: update w, b

Repeat 2 → 4 until $|J^k - J^{k+1}| < \epsilon$

$$dw_i = \frac{1}{m} (\hat{y} - y) x_i$$

$$db = \frac{1}{m} (\hat{y} - y)$$

$$\text{new } w_i = w_i - \lambda \, dw_i$$

$$\text{new } b = b - \lambda \, db$$

underfitting: high Bais (simple)

- more complex — add more features / polynomial

overfitting :- high variance (complex)

- add more data
- Perform Regularization

$$\hat{y} = 60 + 40x_1 - 20x_2$$
$$\|w\|_2 = \sqrt{40^2 + 20^2}$$
$$\|w\|_2^2 = 40^2 + 20^2$$

high $\lambda \Rightarrow$ underfitting
low $\lambda \Rightarrow$ overfitting

Regularization:-

① Ridge :- add $\lambda \|w\|_2^2$

closed-form :-

$$J = (Xw - y)^T (Xw - y) + \lambda w^T w$$
$$w = (X^T X + \lambda I)^{-1} X^T y$$

Gradiant :-

$$J = \frac{1}{2m} \Sigma (\hat{y} - y)^2 + \lambda \|w\|_2^2$$

Steps:

Same as above

but: $dw_i = \frac{1}{m} \Sigma (\hat{y} - y) . X_i + \lambda \|w\|_2^2$

Feature Selection

② Lasso :- add $\lambda \|W\|_1$

$$Loss = \Sigma (\hat{y} - y)^2 + \lambda \|w\|_1$$

③ Elastic net:- $Loss = \Sigma (\hat{y} - y)^2 + \lambda (\alpha \|w\|_2^2 + (1 - \alpha) \|w\|_1)$

Linear classifier (Perceptron):-

$y = -1$ or 1
$\hat{y} = -1$ or 1

decision boundry:-

$$w_1 x_1 + w_2 x_2 + b = 0$$

goal:

$$y^i (\Sigma w x) > 0$$

update:

when: $y^i (\Sigma w X) \leq 0$

new w: $w \leftarrow w + y^i x^i$

$b \leftarrow b + y^i$

Loss:

* $y(wX+b) > 0$ no loss
* $y(wX+b) \leq 0$ loss $= -y(wX+b)$

# of iterations:

$$\left(\frac{R}{\gamma}\right)^2$$

classification Rule:-

$$y^i = \text{sign}(\Sigma w x_i)$$

- Perform training and check the error:
- If error is high: (underfitting)
  - Add more features
  - More complex model by polynomial
- If overfitting:
  - Add more data
  - Perform regularization

Logistics Regression

$y = 0$ or 1

$\hat{y} = \sigma(\Sigma xw + b)$

testing:-

$$\sigma(\Sigma wx + b) = \frac{1}{1 + e^{-(\Sigma wx + b)}}$$

$= 0.5 \Rightarrow$ undetermind

$> 0.5 \Rightarrow$ +ve

$< 0.5 \Rightarrow$ -ve

Gradiant decent:-

$$\text{Cost} = \frac{1}{m} \Sigma - y \log \hat{y} - (1-y) \log(1-\hat{y})$$

$y = 0$ or 1

$\hat{y} = \sigma(\Sigma wx + b)$

Steps:

Same as above

but: $dw_i = \frac{1}{m} (\hat{y} - y) \cdot x$

$db = \frac{1}{m} (\hat{y} - y)$

General form:-

$$P(y|x) = \frac{1}{1 + e^{-y(wx+b)}}$$

Softmax:- $\frac{e^{z_i}}{\Sigma e^{z}}$

R and P

$$\text{accurecy} = \frac{TP}{all}$$

Actual

Predict	+ve	-ve
+ve	TP	FP
-ve	FN	TN

percision

recall

$$\text{Recall} = \frac{TP}{TP + FN}$$

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

F_β-Score

$$F_\beta = (1+\beta^2)\frac{P \times R}{\beta^2 P + R}$$

$$F_1 = 2\frac{P \times R}{P + R}$$

hard SVM

decision boundry:-

$w_1 x_1 + w_2 x_2 + b = 0$

$w_1 x_1 + w_2 x_2 + b = \pm 1 \longrightarrow$ SV boundry

goal:

Primal $\quad \min \frac{1}{2}\|w\|^2 \quad$ S.t. $\quad y^i(\sum w x) \geq 1$

Dual $\quad \max \sum \alpha_i - \frac{1}{2} \sum \alpha_i \alpha_j y^i y^j (x^i . x^j) \quad$ S.t. $\quad \sum \alpha_i y_i = 0$

$y^i(wx + b) = 1$

$w = \sum \alpha_i y^i x^i$

$b = \sum \alpha_i y^i$

high c $\Rightarrow$ overfitting

low c $\Rightarrow$ underfitting

Soft SVM

decision boundry:-

$w_1 x_1 + w_2 x_2 + b = 0$

$w_1 x_1 + w_2 x_2 + b = \pm 1 \longrightarrow$ SV boundry

goal:

Primal $\quad \min \frac{1}{2}\|w\|^2 + C \sum \xi_i \quad$ S.t. $\quad y^i(\sum w x) \geq 1 - \xi_i$

Dual $\quad \max \sum \alpha_i - \frac{1}{2} \sum \alpha_i \alpha_j y^i y^j (x^i . x^j) \quad$ S.t. $\quad \sum \alpha_i y_i = 0$

that vaulate $\leftarrow$ $C > \alpha > 0 \leftarrow y^i(wx + b) = 1$

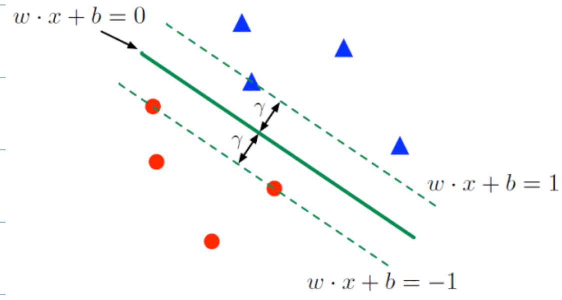
on the boundry $\leftarrow$ $C = \alpha \leftarrow y^i(wx + b) = 1 - \xi_i$

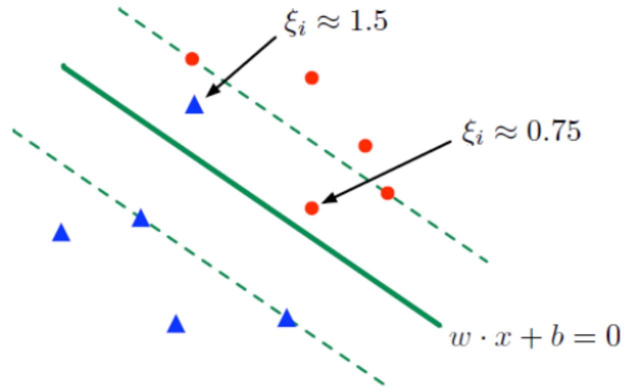
$w = \sum \alpha_i y^i x^i$

$b = \sum \alpha_i y^i$

test:-

$\hat{y} = \text{sign}\left(\sum \alpha y^i x^i \cdot z + b\right)$





measure the similarity

Kernal:-

$$(1 + X \cdot z)^p$$

Jaccard Index:-

$$1 - \frac{X \cap z}{X \cup z}$$

Radial Basis Function:-

$$e^{-\frac{|X-z|^2}{\sigma^2}}$$

high σ ⇒ underfitting

small σ ⇒ overfitting

more data ⇒ σ decreasing

NN

each neuron $= \sigma(\Sigma wx + b)$

cross-entropy $= -y \log \hat{y} - (1-y) \log(1-\hat{y})$

$w \leftarrow w - \alpha \frac{\partial J}{\partial w}$ ① $\frac{\partial J}{\partial w_j} = \frac{1}{m}\Sigma (\hat{y}-y)\, x_j$

$b \leftarrow b - \alpha \frac{\partial J}{\partial b}$ ② $\frac{\partial J}{\partial b} = \frac{1}{m} \Sigma (\hat{y}-y)$

Loss function is not convex!

Three Problems with activation functions:-

1. non zero centered (slower in converge)
2. saturated (no learning)
3. computational cost

	ReLU $\max(0,x)$	Tanh	Sigmoid $\frac{1}{1+e^{-z}}$
zero-centered	Not zero-centered	Zero-centred	Not zero-centered
Saturation	Dose not saturated	saturated	saturated
Computational Cost	efficient	slow	slow
Drivative	$\begin{cases} 0, & \text{if } x \le 0 \\ 1, & \text{other} \end{cases}$	$1 - \tanh^2$	$\sigma(x)(1-\sigma(x))$

for multiple classes :-

What we change

- # of output layer
- activation of output layer SoftMax
- loss

$$\left(\sum_{i=1}^{K} -y \log(\hat{y})\right)$$

# of classes

one-hot-encoding

for Regression:-

What we change

- activation in the output layer (no activation)
- loss

$$MSE = \frac{1}{2m} \sum (\hat{y} - y)^2$$

Decision Tree

$$\text{error} = 1 - \max(P_j)$$

$$\text{Gini} = 1 - \sum (P_j)^2$$

$$\text{Entropy} = -\sum P_i \log_2(P_i)$$

$$IG(D_p) = I(D_p) - I(D_L)\frac{N_L}{N_p} - I(D_R)\frac{N_R}{N_p}$$

Cost of tree = error rate + λ (# of leafs)

Ensemle

AdaBoost

Weighted error rate = $\dfrac{\text{total weight of mistakes}}{\text{total weight of all data}}$

x	S_1	0.1
	S_2	0.6
x	S_3	0.9
	S_4	0.5

$\epsilon = \dfrac{1.0}{2.1}$

weight of classifier = $\dfrac{1}{2} \ln\left(\dfrac{1-\epsilon}{\epsilon}\right)$

④ re w.t the samples $\propto e^{-wt}$, $\propto e^{+wt}$

origin w.t

if it's correct

$$\alpha_i^{j+1} = \begin{cases} \alpha_i^{j} \cdot e^{-w.t}, & \text{if correct} \\ \alpha_i^{j} \cdot e^{+w.t}, & \text{if incorrect} \end{cases}$$

Bagging

RF

Clustering

Loss func = Σ of squared distance from center

① K-means

assign to nearest cluster
update the center

K-means ++ = Smart initialization

LBG = Start with 2 then split until K
MacQueen = update as you assign

② EM

③ Agglomerative hierarchical clustering

Dimensionally Reduction

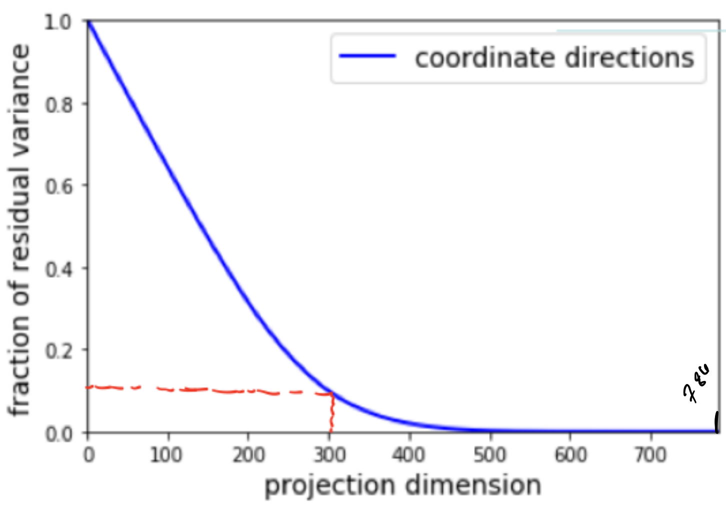
① Remove lowest variance

② PCA

if you don't have exact # of axis you fix % of σ then get $\lambda_1 \dots \lambda_n$ that give you % you want

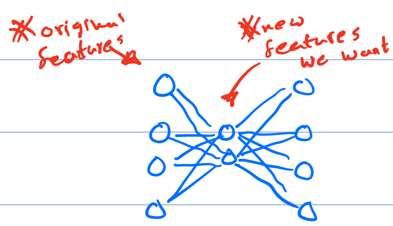
How σ I will get if I select 3

$$\sigma = \frac{\lambda_1 + \lambda_2 + \lambda_3}{\Sigma \lambda}$$



if we keep 300 features we will lose just 10% of the variance

③ T-sne (neural network)



loss = squared loss