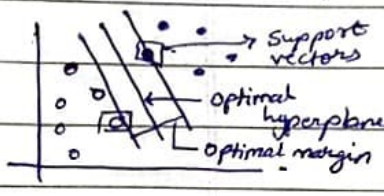
# SVM

→ Determining location of hyperplane that produce optimal separation of classes among infinite no. of boundaries

→ Optimal hyperplane – constructed by searching for maximal marginal hyperplane

⇒ Sum of distances to the hyperplane from closest points of two classes

⇒ sides of margin are parallel

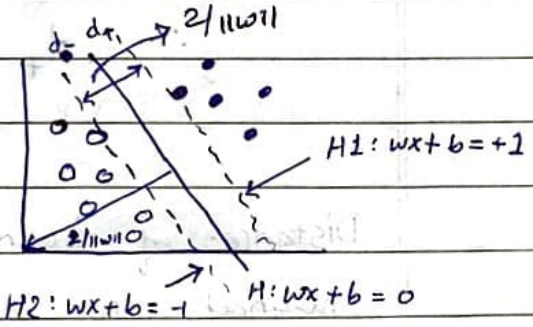



## # Linear SVM

Dataset D $= \{x_i, y_i\}$

$x_i$: $1, 2, \cdots, l$ (no. of samples), $x_i \in R^d$

$y_i$: $\{-1, 1\}$




Eqⁿ of separating hyperplane :

$$\vec{w} \cdot \vec{x} + b = 0$$

1er distance of $(x_0, y_0)$ from hyperplane is :

$$\frac{w_1 x_0 + w_2 y_0 + b}{\sqrt{w_1^2 + w_2^2}} \quad \text{i.e.} \quad \frac{|b|}{\|w\|}$$

∴ 1er distance from origin : $\frac{|b|}{\|w\|}$

Date: / /

For points lying on or above hyperplane,

$H_1 : \vec{x_i} \cdot \vec{w} + b >= +1$ for $y_i = +1$
$H_2 : \vec{x_i} \cdot \vec{w} + b <= -1$ for $y_i = -1$

Both equations can be combined into one set of inequalities as

$$\therefore \boxed{y_i(\vec{x_i} \cdot \vec{w} + b) - 1 >= 0} \text{ for every } i$$

Distance of bounding hyperplane $H_1$ from origin with normal $w$ is,

$$dist(H_1) = \frac{|w_1 x_1 + w_2 x_2 + b - 1|}{\sqrt{w_1^2 + w_2^2}} = \frac{|b-1|}{||w||}$$

$$\therefore d_+ = \frac{|1-b|}{||w||} - \frac{|b|}{||w||} = \frac{|1|}{||w||}$$

Distance of bounding hyperplane $H_2$ from origin with normal $w$ is,

$$dist(H_2) = \frac{|b+1|}{||w||}$$

$$\therefore d_- = \frac{|b|}{||w||} - \frac{|1+b|}{||w||} = \frac{1}{||w||}$$

Hence, margin bet$^n$ two bounding hyperplanes is $\frac{2}{||w||}$.

Objective fn → Maximize $\frac{2}{\|w\|}$

~ Minimize $\frac{\|\vec{w}\|}{2}$

~ Minimize $\frac{\|\vec{w}\|^2}{2}$

Thus, SVM solves objective fn $\Phi(w) = \frac{1}{2}\|\vec{w}\|^2$

subject to constraint $y_i(\vec{x_i} \cdot \vec{w} + b) \geq 1$

→ Convex quadratic programming problem which has single global optimum sol$^n$. Solved in dual space, the space of Lagrange multipliers

Lagrangian of this problem is given by

$$L(\vec{w}, b, \alpha) = \frac{1}{2}\|w\|^2 - \sum_{i=1}^{t} \alpha_i (y_i(\vec{w} \cdot \vec{x_i} + b) - 1) \quad \ldots\ldots (i)$$

Lagrange dual function of optimization problem is given by

$$\max_{\alpha} w(\alpha) = \max_{\alpha} \left( \min_{w, b} L(\vec{w}, b, \alpha) \right)$$

Diff (i) w.r.t w & b and setting derivatives equal to 0,

$$\frac{\partial L(\vec{w}, b, \alpha)}{\partial w} = \vec{w} - \sum_{i=1}^{t} \alpha_i y_i x_i = 0$$

$$\therefore \vec{w} = \sum_{i=1}^{t} \alpha_i y_i x_i \quad \text{—— (ii)}$$

$$\frac{\partial L(\vec{w}, b, \alpha)}{\partial b} = \sum_{i=1}^{l} \alpha_i y_i = 0 \quad \text{—— (iii)}$$

Substituting (ii) and (iii) in (i),

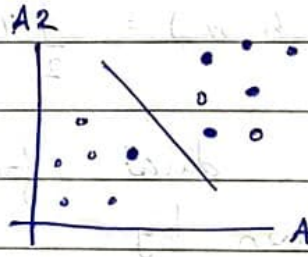
$$W(\alpha) = \frac{1}{2} \sum_{i=1}^{l} \sum_{j=1}^{l} \alpha_i \alpha_j y_i y_j \vec{x_i} \vec{x_j} - \sum_{i=1}^{l} \sum_{j=1}^{l} \alpha_i \alpha_j y_i y_j \vec{x_i} \vec{x_j} - b \cdot (0) + \sum_{i=1}^{l} \alpha_i$$

∴ Dual problem can be now stated as :-

Maximize : $$\boxed{W(\alpha) = \sum_{i=1}^{l} \alpha_i - \frac{1}{2} \sum_{i=1}^{l} \sum_{j=1}^{l} \alpha_i \alpha_j y_i y_j \vec{x_i} \vec{x_j}}$$

Subject to $\sum_{i=1}^{l} \alpha_i y_i = 0$ and $\alpha_i \geq 0$

# Soft Margin SVM




→ Non separable data - contains noisy data
→ Allows mislabelled data points while still maximizing margin
→ Slack variables - measure degree of misclassification

Constraints are now : $\vec{x_i} \cdot \vec{w} + b \geq +1 - \xi_i$ for $y_i = +1$

$\vec{x_i} \cdot \vec{w} + b \leq -1 + \xi_i$ for $y_i = -1$

$\xi_i \geq 0, \forall i$

Date:

Combining constraints, it become

$$y_i(\vec{x_i} \cdot \vec{w} + b) \geq 1 - \xi_i \quad \text{and} \quad \xi_i \geq 0$$

Objective fn now becomes,

Minimize : $\Phi(w, b, \xi_i) = \frac{1}{2}\|\vec{w}\|^2 + C\sum_{i=1}^{l} \xi_i^k$

subject to : $y_i(\vec{w} \cdot \vec{x_i} + b) \geq 1 - \xi_i$ & $\xi_i \geq 0$

C = Regularization parameter
Controls trade off between maximizing margin & minimizing error term.

→ Lagrangian of primal form of objective fn is given :

$$L(w, b, \alpha, \xi, \beta) = \frac{1}{2}\|\vec{w}\|^2 + C\sum_{i=1}^{l} \xi_i - \sum_{i=1}^{l} \alpha_i (y_i(\vec{w} \cdot \vec{x_i} + b) - 1 + \xi_i) - \sum_{i=1}^{l} \beta_i \xi_i$$

→ Solution of above eqn is given as :

$$\max_{\alpha} W(\alpha, \beta) = \max_{\alpha, \beta} \left( \min_{w, b, \xi} L(\vec{w}, b, \alpha, \beta, \xi) \right)$$

Diff. we get, $\frac{\partial(L)}{\partial w} = w = \sum_{i=1}^{l} \alpha_i y_i x_i$

$$\frac{\partial(L)}{\partial b} = \sum_{i=1}^{l} \alpha_i y_i = 0$$

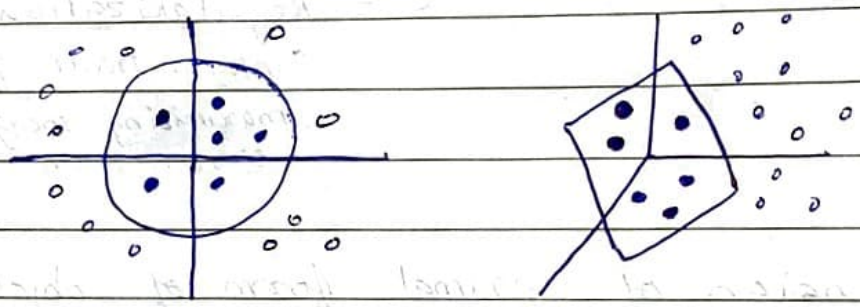
$$\frac{\partial(L)}{\partial \xi} = \alpha_i + \beta_i = C$$

Substituting this into primal form, we get

Maximize : $$W(\alpha) = \sum_{i=1}^{l} \alpha_i - \frac{1}{2} \sum_{i=1}^{l} \sum_{i=1}^{l} \alpha_i \alpha_j y_i y_j \vec{x_i} \vec{x_j}$$

Subject to : $$\sum_{i=1}^{l} \alpha_i y_i = 0 \quad \text{and} \quad C \geq \alpha_i \geq 0$$

## NON LINEAR SVM



Step I Transform input vectors into high dimensional feature space

Step II Use SVM to find hyperplane of maximal margin in new feature space

"Cover's Theorem" - A complex pattern classification problem cast in high dimensional space non linearly is more likely to be linearly separable than in a low-dimensional space.

Let $\Phi$ be nonlinear mapping function,

$$\Phi : R^d \rightarrow \mathcal{H} \quad \text{where } x_i \in R^d \text{ and } \Phi(x_i) \in \mathcal{H}$$

With this transformation, eq$^n$ of hyperplane becomes

$$\vec{w} \cdot \Phi(x) - b = 0$$

Using $\Phi(\cdot)$ function, weight becomes

$$\vec{w} = \sum \alpha_i y_i \Phi(\vec{x_i})$$

Decision function becomes

$$f(x) = sgn\left(\Phi(\vec{x}) \cdot \vec{w} + b\right)$$

$$f(x) = sgn\left(\sum_{i=1}^{N_S} \alpha_i y_i \Phi(\vec{x_i}) \cdot \Phi(\vec{x_j}) + b\right)$$

Dual problem can be rewritten as

$$\boxed{w(\alpha) = \sum_{i=1}^{l} \alpha_i - \frac{1}{2} \sum_{i=1}^{l} \sum_{j=1}^{l} \alpha_i \alpha_j y_i y_j \Phi(\vec{x_i}) \Phi(\vec{x_j})}$$

→ Feature mapping function always appears as dot products $\Phi(\vec{x_i}) \cdot \Phi(\vec{x_j})$

Computing it in high dimensional can be complex, costly or suffer from curse of dimensionality.

Page No.
Date: / /

eg. $\Phi(\vec{x_i}) = (1, x_{i1}^2, \sqrt{2}x_{i1}x_{i2}, x_{i2}^2, \sqrt{2}x_{i1}, \sqrt{2}x_{i2})$

$\Phi : R^2 \to R^6$

$\therefore \Phi(\vec{x_1}) \cdot \Phi(\vec{x_2}) = 1 + 2x_{11}x_{21} + 2x_{12}x_{22} + (x_{11}x_{21} + x_{11}x_{21})^2$

Instead, $(\vec{x_1} \cdot \vec{x_2} + 1)^2 = (x_{11}x_{21} + x_{11}x_{21})^2 + 2x_{11}x_{21} + 2x_{12}x_{22} + 1$

$= \Phi(\vec{x_1}) \cdot \Phi(\vec{x_2})$

## Kernel Functions

→ SVM learning req only on dot product $\Phi(x_i) \cdot \Phi(x_j)$ bet$^n$ training examples.

→ Operations in high dimensional space do not need to be performed explicitly if we find a function $K(\vec{x_i}, \vec{x_j})$ such that

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

Kernel function

Allows to calculate dot product of $(\Phi(x_i), \Phi(x_j))$ without explicitly applying function to input vectors.

Dual Problem is now defined using Kernel function as:

$$w(\alpha) = \sum_{i=1}^{l} \alpha_i - \frac{1}{2} \sum_{i=1}^{l} \sum_{j=1}^{l} \alpha_i \alpha_j y_i y_j K(\vec{x_i}, \vec{x_j})$$

Subject to $\sum_{i=1}^{l} \alpha_i y_i = 0$ and $C \geq \alpha_i \geq 0$

Classification function becomes

$$f(x) = sgn\left(\sum_{i=1}^{N_s} \alpha_i y_i K(\vec{x_i}, \vec{x}) + b\right)$$

→ Kernel function K is valid iff

$$K(\vec{x}, \vec{y}) = \Phi(\vec{x}) \cdot \Phi(\vec{y})$$

→ If K is valid Kernel, then it is:

* Symmetric [ $K(\vec{x}, \vec{y}) = K(\vec{y}, \vec{x})$ for any two objects $\vec{x}, \vec{y} \in R$ ]

* Positive (semi-) definite

$$\left[ z^T K z = \sum_{i=1}^{l} \sum_{i=1}^{l} z_i z_j k(\vec{x_i}, \vec{x_j}) \geq 0 \right]$$

## Some Kernels

| | |
|---|---|
| Linear | $K(\vec{x_i}, \vec{x_j}) = (\vec{x_i} \cdot \vec{x_j})$ |
| Polynomical | $K(\vec{x_i}, \vec{x_j}) = (\vec{x_i} \cdot \vec{x_j} + 1)^d$ |
| Radial Basis Function (RBF) | $K(\vec{x_i}, \vec{x_j}) = e^{-\|\vec{x_i} - \vec{x_j}\|^2 / 2\sigma^2}$ |
| Sigmoid | $K(\vec{x_i}, \vec{x_j}) = \tanh(\kappa(\vec{x_i} \cdot \vec{x_j}) + \delta)$ |

## Advantages of SVM

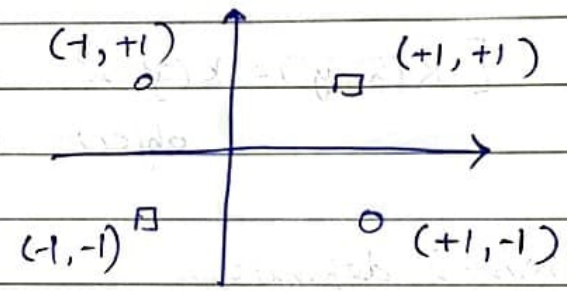
→ Complexity is charachterized by no. of support vectors rather than dimension of data. Hence less prone to overfitting
→ Good generalization capability by maximizing margin in input space
→ No. of SVs found can be used to compute upper bound on expected rate of classifier.

Date: / /

# Non linear SVM example (XOR problem)

| Training set :- | Input vector $X$ | Output $Y$ |
|---|---|---|
| $x_1$ | $(-1, -1)$ | $-1$ |
| $x_2$ | $(-1, +1)$ | $+1$ |
| $x_3$ | $(+1, -1)$ | $+1$ |
| $x_4$ | $(+1, +1)$ | $-1$ |

Plotting training points in 2D space




We can see they are not linearly separable. Let's take

$$K(\vec{x_i}, \vec{x_j}) = (1 + \vec{x_i} \cdot \vec{x_j})^2$$

Image of input vector $\vec{x_i}$ induced in feature space is therefore deduced to be

$$\Phi(\vec{x_i}) = (1, x_{i1}^2, \sqrt{2}\, x_{i1} x_{i2}, x_{i2}^2, \sqrt{2} x_{i1}, \sqrt{2} x_{i2})$$

Now objective function of dual form can be written as

$$w(\alpha) = \sum_{i=1}^{4} \alpha_i - \frac{1}{2} \sum_{i=1}^{4} \sum_{j=1}^{4} \alpha_i \alpha_j y_i y_j K(\vec{x_i}, \vec{x_j})$$

$K(\vec{x_1}, \vec{x_2}) = (1+0)^2 = 1$

$K(\vec{x_1}, \vec{x_1}) = (1+2)^2 = 9$

$K(\vec{x_1}, \vec{x_3}) = (1+0)^2 = 1$

$K(\vec{x_1}, \vec{x_4}) = (1-2)^2 = 1$

$K(\vec{x_2}, \vec{x_2}) = (1+2)^2 = 9$

$K(\vec{x_2}, \vec{x_3}) = (1+(-2))^2 = 1$

$K(\vec{x_2}, \vec{x_4}) = (1+0)^2 = 1$

$K(\vec{x_3}, \vec{x_3}) = (1+2)^2 = 9$

$K(\vec{x_3}, \vec{x_4}) = (1+0)^2 = 1$

$K(\vec{x_4}, \vec{x_4}) = (1+2)^2 = 9$

$$\therefore W(\alpha) = \alpha_1 + \alpha_2 + \alpha_3 + \alpha_4 - \frac{1}{2}\Big( 9\alpha_1^2 - \alpha_1\alpha_2 - \alpha_1\alpha_3 + \alpha_1\alpha_4 - \alpha_1\alpha_2 + 9\alpha_2^2 + \alpha_2\alpha_3 - \alpha_2\alpha_4 - \alpha_1\alpha_3 + \alpha_2\alpha_3 + 9\alpha_3^2 - \alpha_3\alpha_4 + 9\alpha_4^2 + \alpha_1\alpha_4 - \alpha_2\alpha_4 - \alpha_3\alpha_4 \Big)$$

$$= \alpha_1 + \alpha_2 + \alpha_3 + \alpha_4 - \frac{1}{2}\Big( 9\alpha_1^2 + 9\alpha_2^2 + 9\alpha_3^2 + 9\alpha_4^2 - 2\alpha_1\alpha_3 + 2\alpha_1\alpha_4 - 2\alpha_1\alpha_2 + 2\alpha_2\alpha_3 - 2\alpha_2\alpha_4 - 2\alpha_3\alpha_4 \Big) \quad \text{---- (i)}$$

Optimising $W(\alpha)$ w.r.t Lagrange multipliers gives following eq^n's :

$$9\alpha_1 - \alpha_2 - \alpha_3 + \alpha_4 = 1$$
$$-\alpha_1 + 9\alpha_2 + \alpha_3 - \alpha_4 = 1$$
$$-\alpha_1 + \alpha_2 + 9\alpha_3 - \alpha_4 = 1$$
$$\alpha_1 - \alpha_2 - \alpha_3 + 9\alpha_4 = 1$$

Hence optimal values of Lagrange multipliers are

$$\alpha_1 = \alpha_2 = \alpha_3 = \alpha_4 = \frac{1}{8}$$

Substituting values of $\alpha$ in (i), we get

$$W(\alpha) = \frac{1}{4}$$

$$\therefore \quad \frac{1}{2}\|\vec{w}\|^2 = \frac{1}{4}$$

$$\therefore \|\vec{w}\|^2 = \frac{1}{2} \quad \Rightarrow \quad \|w\| = \frac{1}{\sqrt{2}}$$

Now $$w = \sum_{i=1}^{4} \alpha_i y_i \Phi(x_i)$$

$$= \frac{1}{8}\left[y_1 \Phi(\vec{x_1}) + y_2 \Phi(\vec{x_2}) + y_3 \Phi(\vec{x_3}) + y_4 \Phi(\vec{x_4})\right]$$

$$= \frac{1}{8}\left[-\Phi(\vec{x_1}) + \Phi(\vec{x_2}) + \Phi(\vec{x_3}) - \Phi(\vec{x_4})\right]$$

$$= \frac{1}{8}\left[-\begin{bmatrix}1\\1\\\sqrt{2}\\-\sqrt{2}\\-\sqrt{2}\end{bmatrix} + \begin{bmatrix}1\\1\\-\sqrt{2}\\-\sqrt{2}\\-\sqrt{2}\end{bmatrix} + \begin{bmatrix}1\\1\\-\sqrt{2}\\\sqrt{2}\\-\sqrt{2}\end{bmatrix} - \begin{bmatrix}1\\1\\\sqrt{2}\\\sqrt{2}\\\sqrt{2}\end{bmatrix}\right]$$

$$= \begin{bmatrix}0\\0\\-\frac{1}{\sqrt{2}}\\0\\0\\0\end{bmatrix}$$

bias b is 0 because first element of w is 0.

∴ Optimal hyperplane becomes :

$$\vec{w} \cdot \Phi(\vec{x}) = \begin{bmatrix} 0 & 0 & \frac{-1}{\sqrt{2}} & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 1 \\ x_1^2 \\ \sqrt{2}x_1 x_2 \\ x_2^2 \\ \sqrt{2}x_1 \\ \sqrt{2}x_2 \end{bmatrix} = 0$$

$$\therefore \boxed{-x_1 x_2 = 0} \quad \text{—— Ans}$$

# KNN

→ Essential classification algo in ML
→ Supervised learning domain
→ Non parametric meaning doesn't make any underlying assumptions about distribution of data
→ Instance based or lazy learning where $f^n$ is only approx. locally & all computation is deferred until classification.

Process :

→ Let m be no. of training data samples. Let p be unknown point.

1. Store training samples in an array of datapoints arr[]. This means each element of this array represents tuple $(x_1, x_2, \ldots, x_n)$.
2. for i=0 to m : Calculate cosine similarity $\cos\phi = \frac{(a*b)}{(\|a\| \|b\|)}$ where a is input tuple & b is targeted tuple
3. Make set S of K smallest distances obtained. Each of these distances corresponds to an already classified datapoint
4. Return majority label / distance weighted label among S.

Page No.
Date: / /

## Cosine similarity.

$$\cos\theta = \frac{\vec{a}\cdot\vec{b}}{\|\vec{a}\|\,\|\vec{b}\|} = \frac{\sum_1^n a_i b_i}{\sqrt{\sum_1^n a_i^2}\,\sqrt{\sum_1^n b_i^2}}$$

where $\vec{a}\cdot\vec{b} = \sum_{i=1}^n a_i b_i = a_1 b_1 + a_2 b_2 + \cdots + a_n b_n$ is dot product of two vectors

## # Solved example

| | History | Scienca | Research | offers | Students | Hall | Class |
|---|---|---|---|---|---|---|---|
| An | | 0.537 | 0.477 | | 0.673 | 0.177 | A |
| Ar | | | | 0.961 | 0.195 | 0.196 | B |
| Bi | | 0.347 | 0.924 | | 0.111 | 0.112 | A |
| Ch | | 0.175 | | | 0.155 | 0.158 | A |
| Comm | | | | 0.78 | 0.626 | 0 | B |
| Comp | | 0.989 | | | 0.130 | 0.067 | A |
| Cr | | | | | 1 | 0 | B |
| Ec | | | 1 | | 0 | 0 | A |
| En | | | | 0.98 | 0 | 0.199 | B |
| Ge | | 0.849 | | | 0.529 | 0 | A |
| Hi | 0.991 | | | 0.135 | 0 | 0 | B |
| Ma | | 0.616 | 0.549 | 0.49 | 0.198 | 0.201 | A |
| Mo | 0 | | | 0.928 | 0.17 | 0.373 | B |
| Mu | 0.97 | | | | 0 | 0.172 | B |
| Phi | 0.741 | | | 0.458 | 0.315 | 0.136 | B |
| Phy | | | 0.894 | | 0.062 | 0.318 | A |
| Po | | 0.933 | 0.348 | 0.387 | 0.313 | 0.853 | A |
| Ps | | | 0.852 | 0.57 | 0.0459 | 0.162 | A |
| So | | | 0.639 | | 0.967 | 0.237 | A |
| Th | | | | | 0.967 | 0.254 | A |
| An | | 0.537 | 0.477 | | 0.673 | 0.177 | A |

Find class ($\pi_h$) = ?

Calculate cosine similarity with each training sample & return results in descending order.

| Document | Class | Similarity to Theatre ($\pi_h$) |
|---|---|---|
| Cr | B | 0.967075 |
| An | A | 0.695979 |
| Co | B | 0.605667 |
| Gre | A | 0.570589 |
| So | A | 0.5 |
| Ph | A | 0.38 |
| Ps | A | 0.34 |
| Ma | A | 0.21 |
| Ar | B | 0.23 |
| Mu | B | 0.2 |
| Ch | A | 0.18 |
| Comp | A | 0.14 |
| Bi | A | 0.13 |
| Mo | B | 0.09 |
| Po | A | 0.07 |
| En | B | 0.05 |
| Ph | B | 0.03 |
| Hi | B | 0 |
| Ec | A | 0 |

Hence for 1-NN : B
3-NN : B
5-NN : A

Distance weighted 3-NN : B
19-NN : A

Page No.
Date: / /

**Q** How to choose K ?

→ No straightforward method to calculate K.

Thumb rule is $K = \sqrt{n}$ → no. of training examples

Imp ⇒ K should be odd
⇒ K should not be multiple of no. of classes
⇒ shouldn't be too small (greater influence of noise) or too large (biased towards highly probable class)

# CLUSTERING

→ Organization of unlabeled data into similarity groups called clusters.

→ Cluster is collection data items which are similar between them & dissimilar to data items in other clusters.

Evaluation

→ Intra cluster cohesion
- → How near data points in cluster are to cluster centroid
- → Sum of squared error is common measure

→ Inter cluster separation
- → Diff. cluster centroids should be far away from each other

# K Means Clustering

→ Partitional clustering algorithm

→ Let set of datapoints D be $\{x_1, x_2, \cdots, x_n\}$

Here $x_i = (x_{i1}, x_{i2}, \cdots x_{ir})$

$r \rightarrow$ no. of dimensions

→ K-means algo partitions given data into K clusters

↓

Each cluster has centre called centroid.

**Process** Given K, algo works as follows:

1. Choose K (random) datapoints as ini centroids
2. Assign each datapoint to closest centroid
3. Recomputes centroids using current cluster membership
4. If convergence criterion is not met, repeats steps 2 and 3.

**Stopping criteria**

- No reassignments of datapoints to diff cluster
- No change of centroids
- Min. decrease in sum of squared error (SSE)

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

↓

Euclidian distance bet$^n$ x & centroid $m_j$

Date: / /

- **Solved example**

Use K means algo & euclidean distance to cluster the following 8 examples into 3 clusters:

A1 = (2,10) A2 = (2,5) A3 = (8,4) A4 = (5,8)
A5 = (7,5) A6 = (6,4) A7 = (1,2) A8 = (4,9)

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 5 | 6 | $\sqrt{13}$ | $\sqrt{50}$ | $\sqrt{52}$ | $\sqrt{65}$ | $\sqrt{5}$ |
| 2 | | 0 | $\sqrt{37}$ | $\sqrt{18}$ | 5 | $\sqrt{17}$ | $\sqrt{10}$ | $\sqrt{20}$ |
| 3 | | | 0 | 5 | $\sqrt{2}$ | $\sqrt{2}$ | $\sqrt{53}$ | $\sqrt{41}$ |
| 4 | | | | 0 | $\sqrt{13}$ | $\sqrt{17}$ | $\sqrt{52}$ | $\sqrt{2}$ |
| 5 | | | | | 0 | $\sqrt{2}$ | $\sqrt{45}$ | $\sqrt{25}$ |
| 6 | | | | | | 0 | $\sqrt{29}$ | $\sqrt{29}$ |
| 7 | | | | | | | 0 | $\sqrt{58}$ |
| 8 | | | | | | | | 0 |

Suppose initial seeds are A1, A4 and A7. Run K-means algo for 1 epoch. At end of this epoch show:

a) New clusters
b) Centre of new clusters
c) Draw 10 x 10 space with all 8 points & show the clusters after first epoch & new centroids.
d) How many more iterations are needed to converge?

Seed 1 = A1 = (2, 10)

Seed 2 = A4 = (5, 8)

Seed 3 = A7 = (1, 2)

Epoch 1

A1: $d(A1, seed1) = 0$
$d(A1, seed2) = \sqrt{13}$
$d(A1, seed3) = \sqrt{65}$
→ A1 ∈ cluster 1

A2: $d(A2, seed1) = 5$
$d(A2, seed2) = \sqrt{18}$
$d(A2, seed3) = \sqrt{50}$
→ A2 ∈ cluster 3

A3: $d(A3, seed1) = 6$
$d(A3, seed2) = 5$
$d(A3, seed3) = \sqrt{53}$
→ A3 ∈ cluster 2

A4: $d(A4, seed1) = \sqrt{13}$
$d(A4, seed2) = 0$
$d(A4, seed3) = \sqrt{52}$
→ A4 ∈ cluster 2

A5: $d(A5, seed1) = \sqrt{50}$
$d(A5, seed2) = \sqrt{13}$
$d(A5, seed3) = \sqrt{45}$
→ A5 ∈ cluster 2

A6: $d(A6, seed1) = \sqrt{52}$
$d(A6, seed2) = \sqrt{17}$
$d(A6, seed3) = \sqrt{29}$
→ A6 ∈ cluster 2

A7: $d(A7, seed1) = \sqrt{65}$
$d(A7, seed2) = \sqrt{52}$
$d(A7, seed3) = 0$
→ A7 ∈ cluster 3

A8: $d(A8, seed1) = \sqrt{5}$
$d(A8, seed2) = \sqrt{2}$
$d(A8, seed3) = \sqrt{58}$
→ A8 ∈ cluster 2

End of epoch 1.

new clusters :

1 : {A1}

2 : {A3, A4, A5, A6, A8}

3 : {A2, A7}

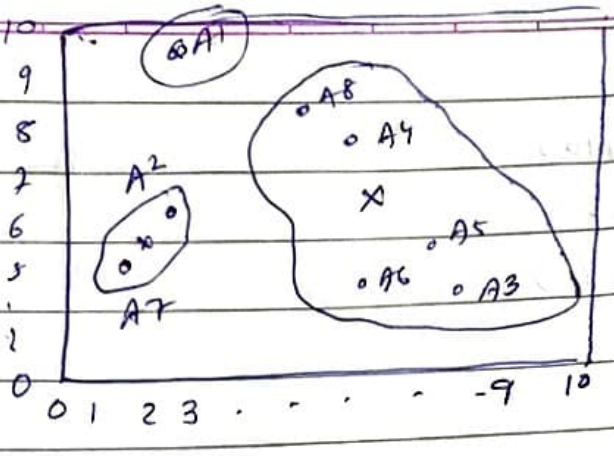
Centres of new clusters :

$C1 = (2, 10)$

$C2 = \left(\frac{8+5+7+6+4}{5}, \frac{4+8+5+4+9}{5}\right)$

$= (6, 6)$

$C3 = \left(\frac{2+1}{2}, \frac{5+2}{2}\right) = (1.5, 3.5)$




We would need 2 more epochs. After 2nd ~~epoch~~ epoch results would be :

1 : {A1, A6}, 2 : {A3, A4, A5, A6}, 3 : {A2, A7}
with centres C1 = (3, 9.5), C2 = (6.5, 5.25) & C3 = (1.5, 3.5)

After 3rd epoch, results :

1 : {A1, A4, A8}, 2 : {A3, A5, A6}, 3 : {A2, A7}
with centres C1 = (3.66, 9), C2 = (7, 4.33) and C3 = (1.5, 3.5)

Make two plots one for each 2nd & 3rd epoch as above.

Limitations :

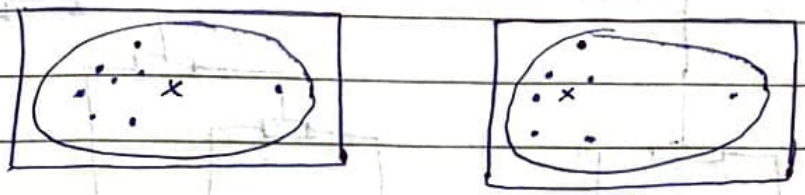
→ Sensitive to outliers as mean is easily influenced by extreme values.

→ Sensitive to initial seeds

# K Medoids (Partitioning around medoids (PAM)) algo

→ K-medoids uses an actual point in the cluster to represent it.

→ Medoid is most centrally located object of cluster, with minimum sum of distances to other points.




→ Mean is greatly influenced by outlier & thus cannot represent correct cluster centre. While medoid is robust to outlier & correctly represents the cluster centre.

# Hierarchical Agglomerative Clustering

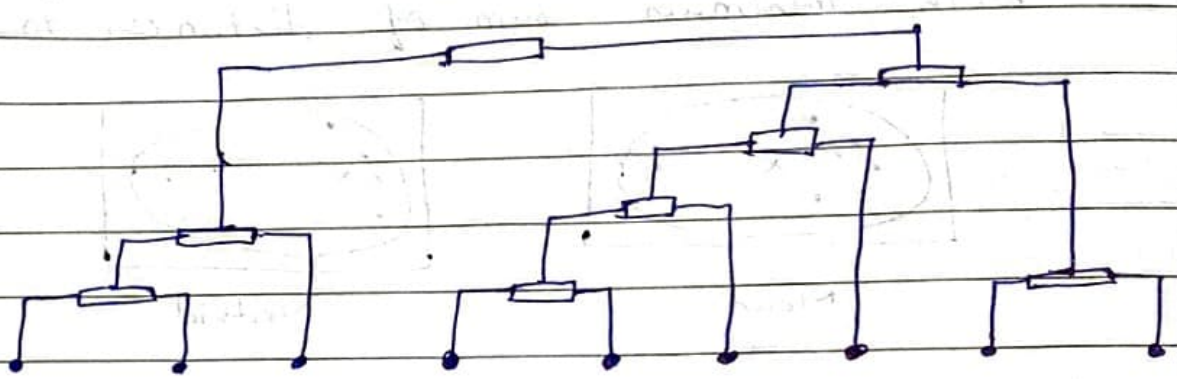
→ Works by grouping data objects into hierarchy or "tree" of clusters.

→ "Hierarchical Agglomerative approach" uses bottom up strategy.

→ Starts by letting each object form its own cluster & iteratively merges cluster into larger and larger clusters, until all the objects are in single cluster or certain termination conditions are satisfied.

## Dendogram

→ Tree that shows how clusters are merged / split hierarchically.
→ Each node on tree is cluster; each leaf node is singleton cluster.



## Agglomerative Clustering

Process:

1. Compute distance matrix
2. Let each data point be cluster
3. Repeat
   1. Merge two closest clusters
   2. Update distance matrix
4. Until only a single cluster remains

## Distance measures

Min distance: $dist_{min}(C_i, C_j) = \min_{p \in C_i, p' \in C_j} \{|p - p'|\}$

Max distance: $dist_{max}(C_i, C_j) = \max_{p \in C_i, p' \in C_j} \{|p - p'|\}$

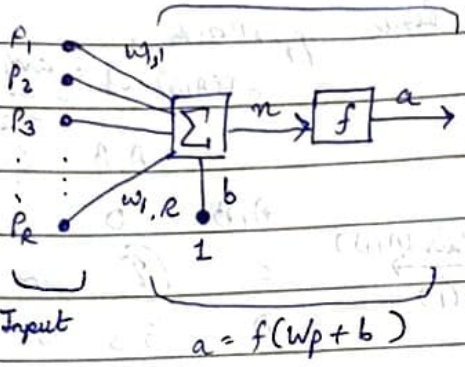
Mean distance: $dist_{mean}(C_i, C_j) = |m_i - m_j|$

Avg distance: $dist_{avg}(C_i, C_j) = \frac{1}{n_i n_j} \sum_{p \in C_i, p' \in C_j} |p - p'|$

# Neural Networks

## Neuron with vector input

Neuron w Vector Input

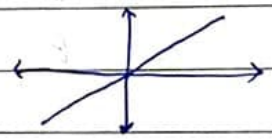



where
R = no. of elements in input vector.

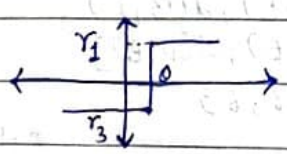
$$n = w_{1,1} p_1 + w_{1,2} p_2 + \cdots + w_{1,R} p_R + b$$

## Activation Functions

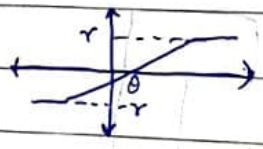
Linear



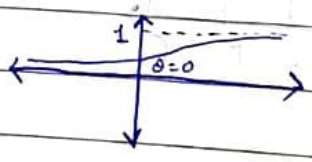
Step



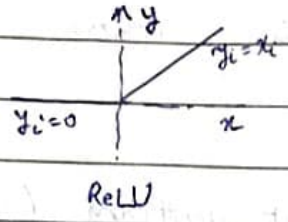

Ramp




Sigmoid



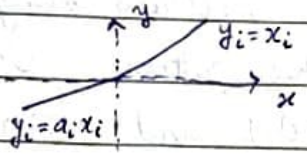
ReLU




ReLU

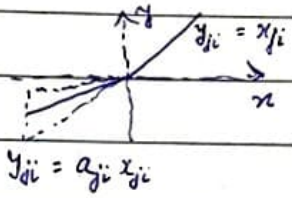
$f(net) = \max(0, net)$

Leaky ReLU




$0.01 \times x_i / a_i \times x_i$

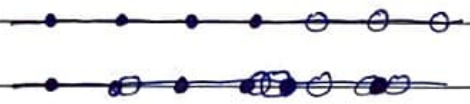
Randomized Leaky ReLU




$$y_{ji} = \begin{cases} x_{ji} & \text{if } x_{ji} \geq 0 \\ a_{ji}\, x_{ji} & \text{if } x_{ji} < 0 \end{cases}$$

## Linear Separability

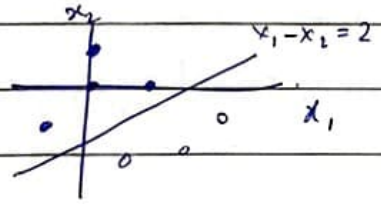
* 1D case :



Separable by perceptron

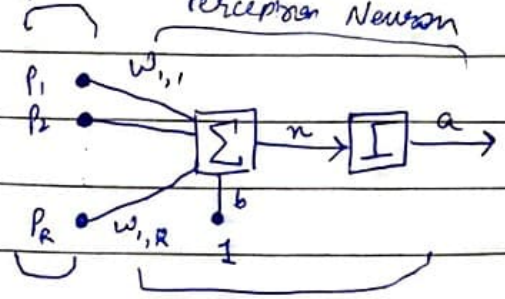
Linearly inseparable

* 2D case :




**Perceptron model** → Machine which can learn to assign input vectors to different classes.

→ Can do 2 class linear classification problem




$a = hardlim(Wp + b)$ $\quad hardlim(n) = 1$, if $n >= 0$ ; 0 otherwise

$W_{new} = W_{old} + \eta e p$

$b_{new} = b_{old} + \eta e$, where $e$ = target - actual

Q. 7 one dimensional input patterns (0.0, 0.17, 0.33, 0.5, 0.67, 0.83, 1.03. Assume that first four patterns belong to class 0 & remaining patterns belong to class 1. Design a perceptron to classify these patterns. Use perceptron learning rule. Assume $\eta = 0.1$ and initial weight & bias to be (-0.36) & (-0.1) respectively. Show computation for two epochs.

# Prev yr problem

| Y | X | $\hat{y}$ | $y - \hat{y}$ |
|---|---|---|---|
| 15.5 | 630 | 10.75 | -4.75 |
| 7.5 | 370 | 4.25 | -3.25 |
| 13.7 | 616 | 10.4 | -3.5 |
| 18.7 | 700 | 12.5 | -6.2 |
| 8.2 | 430 | 5.75 | -2.45 |
| 13.2 | 568 | 9.2 | -4 |
| 23 | 1200 | 25 | 2 |
| 87.3 | 2976 | 69.84 | -17.89 |
| | | | $\Sigma$ = ~~321.25~~ -40.05 |

Initially, $\beta_0 = -5$, $\beta_1 = 0.025$

$\alpha = 0.1$

$$\therefore \beta_0 = \beta_0 - \alpha \frac{1}{m} \Sigma (\hat{y} - y)$$

$$= -5 - \frac{(0.1)}{8} (-40.05)$$

$$= \boxed{-5.500}$$

| | X | Y | class |
|---|---|---|---|
| i | -1 | 1 | - |
| ii | 0 | 1 | + |
| iii | 0 | 2 | - |
| iv | 1 | -1 | - |
| v | 1 | 0 | + |
| vi | 1 | 2 | + |
| vii | 2 | 2 | - |
| viii | 2 | 3 | + |

To be classified → (1, 1)

| | | | |
|---|---|---|---|
| dist (i, (1,1)) | → | $\sqrt{(1-(-1))^2 + (1-1)^2}$ | 2 |
| dist (ii, " ) | → | $\sqrt{(1-0)^2 + (1-1)^2}$ | 1 |
| " | → | $\sqrt{(1-0)^2 + (1-2)^2}$ | $\sqrt{2}$ |
| " | → | $\sqrt{(1-1)^2 + (1-(-1))^2}$ | 2 |
| " | → | $\sqrt{(1-1)^2 + (1-0)^2}$ | 1 |
| " | → | $\sqrt{(1-1)^2 + (1-2)^2}$ | 1 |
| " | → | $\sqrt{(1-2)^2 + (1-2)^2}$ | $\sqrt{2}$ |
| " | → | $\sqrt{(1-2)^2 + (1-3)^2}$ | $\sqrt{5}$ |

Arranging in ascending order

| Dist | class | |
|---|---|---|
| $\sqrt{5}$ | + | |
| 2 | - | → By 3-NN, ans is (-) |
| 2 | - | |
| $\sqrt{2}$ | - | → By 5-NN, ans is (-) |
| $\sqrt{2}$ | - | |
| 1 | + | |
| 1 | + | |
| 1 | + | |