SMAI (07/01/2020)

→ Collect some data – D.

p-dim. vector	label space
x_1	y_1
x_2	y_2

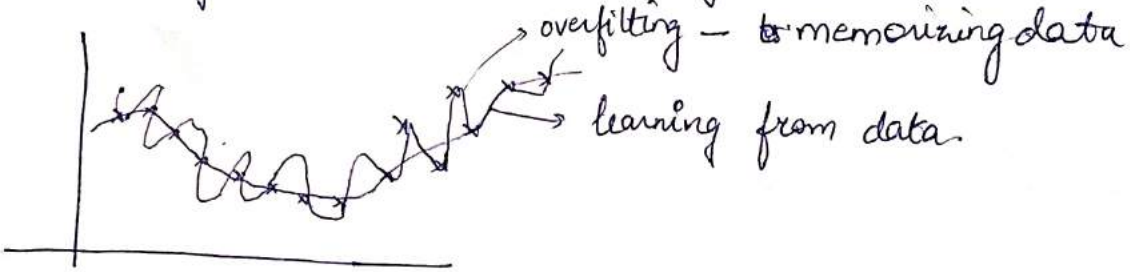
$\{1 \ldots 10\}$.

→ input p-dim vector and.
output is a value – classification problem.

→ ML starts with collecting good data [Imp.].

→ Goal – to do well on collected data ?
OR to do well on future data ?

→ Goal is not to do perfect on collected data but to do good on future data.
overfitting on current data is not good.



⇒ overfitted one – has less loss percentange
[loss percent – diff. ~~from~~ b/w predicted & actual value].

→ Data – | Train | Test. |

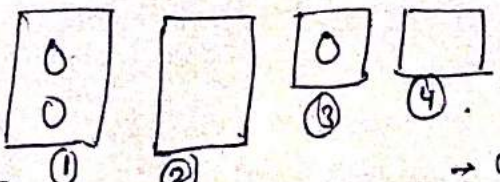
making data splits
[very careful during splitting data].

→ K-nearest Neighbours.

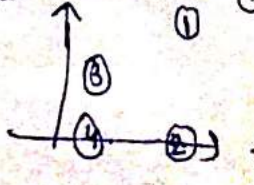
k – odd no. (generally).

major drawback – we have to calculate dist. from all points in dataset & then find k-nearest for classification.
This ~~causes~~ dist. calculation is very time consuming.

eg- Factory →



hole area



→ algo to sort them automatically, based on $\frac{\text{hole area}}{\text{total area}}$

$$x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \qquad x' = \begin{bmatrix} x_1' \\ x_2' \end{bmatrix}$$

euclidian distance $= \sqrt{(x_1 - x_1')^2 + (x_2 - x_2')^2}$

⇒ Cosine distance – Angle b/w two objects

document – collection of words

→ bag of words representation.

[x has a macho figure. In laboratory test .]

doc 1

[figure 1 illus. our study laboratory test forms ... We assume labo.]

doc 2

Suppose in our vocabulary, there are 2 words – laboratory, figure.

we represent doc. by bag of words – $\begin{bmatrix} 1 \\ 1 \end{bmatrix}$ $\begin{bmatrix} 1/2 \\ 1/2 \end{bmatrix}$ → Sparse generally. Many values may be 0.

↑ bag of words vector.

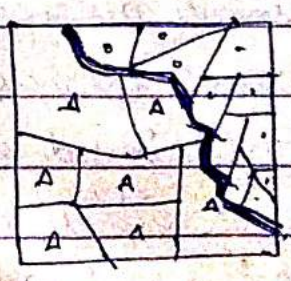
→ According to data, we have to decide which algo to use, which distance to use (euclidian or cosine).

Nearest Neighbour classification

→ Voronoi Tesellation

→ partition space in regions.

→ boundary : points at same dist. from two diff. training examples.



[1-NN classification]

⇒ We look at one point near the data and based on it decide its region.

→ decision boundary : non-linear ; compare with other classification approaches.
not a single line - non-linear, arbitary.

→ Issue : sensitive to outliers.
Single mislabled example can significantly change boundary.

KNN Classification

→ Given pairs of training examples $\{x_i, y_i\}$, we want to classify a testing point x.

→ Algo -

(i) Compute distance $D(x_i, x_j)$ to every training example x_j.

(ii) Select k closest instances $x_{i_1} \dots x_{i_k}$ and their labels $y_{i_1} \dots y_{i_k}$.

(iii) Output the class y^* which is most frequent in $y_{i_1} \dots y_{i_k}$.

→ No parametric approach, no training.

KNN for learning

→ Given pair of traing examples $\{x_i, y_i\}$, we want $y \in R$ for given input x.

→ Algo - (i) Compute distance $D(x_i, x_j)$ to every traing example x_j.
(ii) Select k closest instances $x_{i_1} \dots x_{i_k}$ and their labels $y_{i_1} \dots y_{i_k}$.
(iii) Output the weighted mean of $y_{i_1} \dots y_{i_k}$.
weight $\propto$ 1/distance → closer : more weight & farther : less weight.

[This algo is mainly used for classification but can be used for regression as well.]

How to decide value of k?

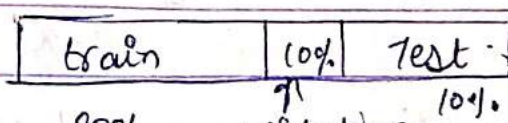
→ if value of k is large — the majority class will always be the result i.e. class having minority won't be selected in result.

→ if k is very small — a lot of noise, due to some noise, the result will change. highly susceptible to outliers.

→ need a decent k.

→ experiment with diff. k.; diving dataset; cross-validation.

→ divide dataset into 3 parts



train with 80% data and the test for k=1 — n for the validation data; decide on what k are you getting max. accuracy (how many of 10% ~~are~~ can you correctly classify).
Test on testing data.

⇒ k should always be less than $\sqrt{n}$ $\{k \leq \sqrt{n}\}$.
{5, 7, 11 commonly used} ↳ usually take much lower than that.

→ checking k-on test data — overfitting

→ Distance Metric.

(a) Euclidien $D(x, x') = \sqrt{\sum_i (x_i - x)^2}$

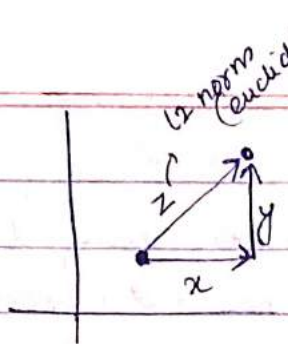
→ (dividing) splitting dataset: split happens across labels also (each label 80%); signature verification; handwriting — all data must not be in training. and some should be left for test untouched

⇒ Mahalarobis distance: $D(x, x') = (x - x')^T \Sigma^{-1} (x - x^{-1})$.

⇒ Minkowski ~~Distance~~ Norm (p-norm): $D(x, x') = \sqrt[p]{\sum_i (x_i - x_i')^p}$

- p=1 → manhattan
- p=2 → euclidian dist.

$\left(\begin{matrix} p=\infty \\ p=0 \end{matrix}\right)$ → interesting read about them.



$x + y = z$.

↓

mahaltan dist.

L2 Norm

→ Cosine Dist.

→ Hamming Dist.

$D(x, x') = \sum_i 1_{x_i \neq x_i'}$

eg - (a) at
(b) at $D = 1 + 0 + 0$.

→ KL Divergence

$$D(x, x') = \sum x_i \log\left(\frac{x_i}{x_i'}\right)$$

→ information, entropy, mutual info, self energy.
→ comes from information theory.
→ not symmetric.

Norm is of a Vector. [Not 2 vectors]

- L2 norm — $x_1^2 + x_2^2 \cdots x_n^2$ (sq. sum of vectors).
- L2 norm — $|x_1| + |x_2| \cdots |x_n|$ (mod sum of vectors)

SMAI (lecture - 2) (11/01)

Minkowski Dist. → if $p = \infty$, it selects $\boxed{\max(x - x')}$.

$$\left((x_0 - x_0')^p + (x_1 - x_1')^p + (x_2 - x_2')^p \cdots\right)^{1/p}.$$

eg - $\left((3)^{1000} + (1.5)^{1000} \cdots (4)^{1000}\right)^{1/1000}$.

$\simeq \left((4)^{1000}\right)^{1/1000} \simeq 4$.

→ is comp. to other values max value will dominate in case of $p = \infty$.

when $p = \infty \Rightarrow \boxed{\max_i (x_i - x_i')}$

When $p=0$ $\Rightarrow$

→ MacAdam Ellipses – is a region on a chromaticity diagram which contains all colors which are indistinguishable to avg. human eye from the color at the centre of ellipse.

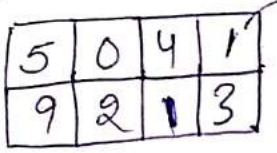
→ How to pick odd K → flip coin, pick class with greater prior

→ Suppose we do 7-NN and there are 4 classes C_1 C_2 C_3 C_4.
3 3 1 0.
Now to choose from C_1 & C_2 – we use 1-NN. i.e. what is closest is C_1 & C_2.

→ what if there are missing values?
take median, mean etc. – replace by some value. performing calc on that class.

Applications of KNN

① MNIST dataset ← manual data extend to be read)



→ these all are images.

← [handwriting recognition]

→ decent performance when lots of data

→ Nearest neighbour is competitive.

② Problem: Where (which country or GPS location) was this image taken?
↑
Research Paper.

Data is a challenge; representation is also a challenge.

→ image – set of pixels – 2D grid.

how problem 2 was solved.

- (i) Get 6M images from Flickr with gps info (dense sampling across world)
- (ii) Represent each image with meaningful features.
- (iii) Do KNN (large k better, they use K=120)

Paper

③ Given an image can you caption it?

→ query expansion approach $q = \frac{1}{NM} \sum_{i=1}^{N} \sum_{j=1}^{M} \text{sim}(I_q, I_i) \cdot \frac{j}{i}$

Read word2vec.

KNN Pros & Cons

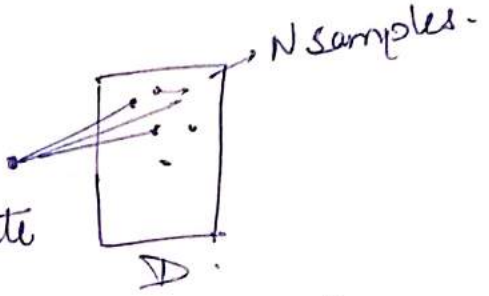
- → Simple, easy to understand.
- → Non parametric (No training req.)
- → Widely Applicable.
- → No assumption about data.
- → Easy to update.
- → Could be used as baseline algo.
- → K & D — only two choices.

Cons

- → computation time is very high. — the major issue.
- → sensitive to outliers.

→ $O(n \cdot d)$ — Time Complexity.
d - dimensionality

for each of the sample, you compute N times.



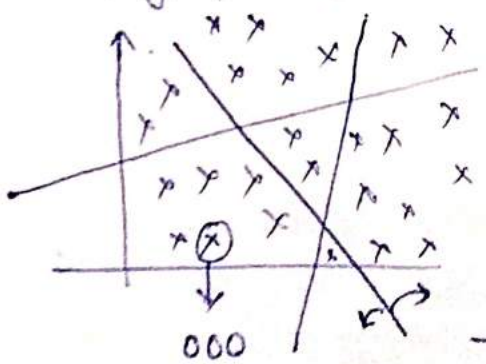
Making KNN Fast

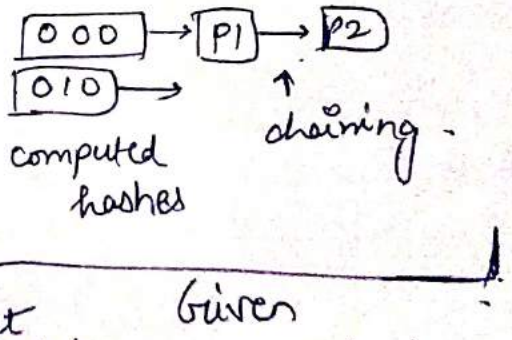
→ Ideas : (i) Reduce d — dimensionality reduction.
(ii) Reduce n — don't compare with all training example.

- Kd Trees $O(d \log_2 n)$
- locally sensitive hashing.
- inverted indices.

Locally Sensitive Hashing (— Really Useful)

→ line / Hyperplane (4-D & above) divide space in 2 parts.





→ define k-hyperplanes, randomly decided

→ point → k-dim. vector.

→ for a test point we compute hash and then apply knn in the chain of that hash only.

→ Complexity → $O(kd + dn/2^k)$

for 1 plane, 2 choices, k-planes — 2^k choices.
distance from $n/2^k$ points.

k-dim, d-planes & then checking +ve/-ve for ~~hyperplanes~~ hashing.

⇒ Parsen Window — anything within a const. dist. will be considered. is nearest neighbours.

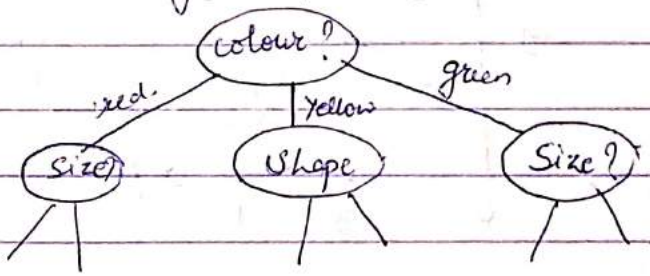
SMAI (18/01/2020).

Decision Trees

→ one of the most intuitive classifiers.
→ easy to understand and construct.
→ Surprisingly, also works very (very) well.

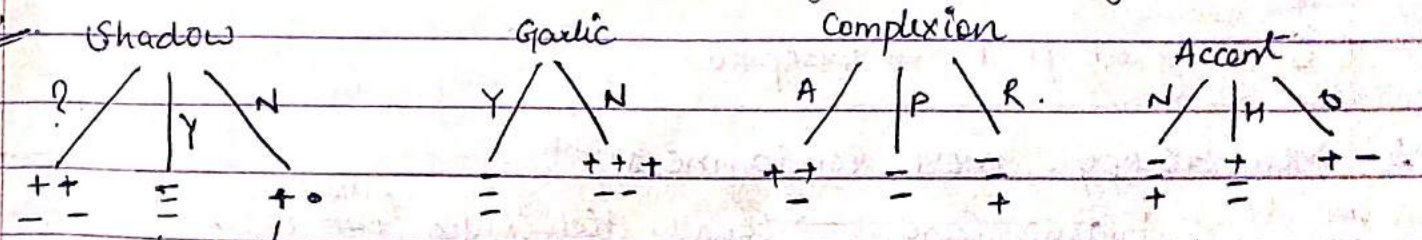
→ Netflix used decision tree for recommendation feature. (had large data but algo worked well).

Eg. fruit described using 4 tuple : {color, shape, taste, size}.



Goal : to build this tree ; data (large amount) is given.

Eg:



↳ Pure sets (only one value).

→ Choose factor having more pure sets.

Hence Shadow → Garlic. (for decision tree)

Impurity functions

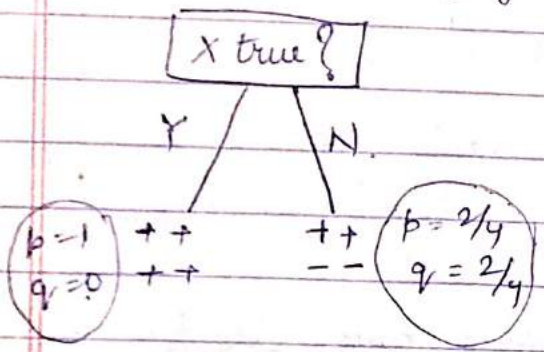
① Entropy : $i(v) = -[q \log q + (1-q)\log(1-q)]$] More commonly used

② Gini Index : $i(v) = 2q(1-q)$

③ Misclassification Rate : $i(v) = \min(q, 1-q)$

Entropy $\Rightarrow$ $\sum -p_i \log p_i$

$= -(p \log p + (1-p)\log(1-p))$ → since 2 classes.



p = +ve class

q = -ve class.

entropy set 1.

$= -(1\log 1 + 0 \log 0)$

compute limit → $= 0$.

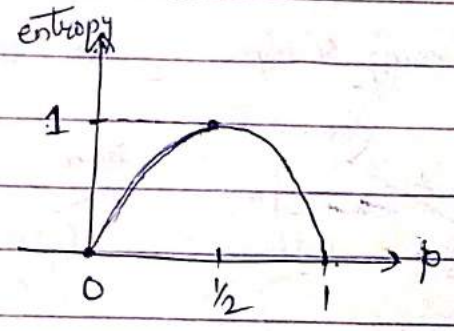
$\left\{ \lim_{p \to 0} p \log p = \lim_{p \to 0} \frac{1/p}{-1/p^2} = 0 \right\}$.

Similarly at $p=1 \Rightarrow$ entrop $= 0$.

set 2 entropy.

$= -\left(\frac{1}{2}\log\frac{1}{2} + \frac{1}{2}\log\frac{1}{2}\right)$

$= 1$



More Entropy, more randomness [~~Pure~~]

Neither sure which way, both equally likely → Worst Case → Equally distributed [Most ~~Most~~ random].

Best Case → pure set.

Most random thing will be equally distributed

Sets which gives ~~higher~~ lower entropy are better.

Shadow

S1	S2	S3
++	=	+
--	=	

$\frac{1}{2}$(entropy S1) + $\frac{3}{8}$(entropy S2) + $\frac{1}{8}$(entropy S3)

$= \frac{1}{2}(1) + \frac{3}{8}(0) + \frac{1}{8}(0) = 0.5$.

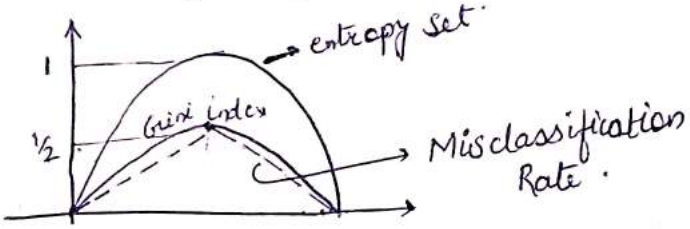
Entropy [Garlic] $= \frac{3}{8} \times 0 + \frac{5}{8} \times -$ = (0.6) Suppose.

Entropy [Complexion] $= \frac{3}{8} \times - + \frac{2}{8} \times 0 + \frac{3}{8} \times -$.

⇒ Entropy Set = weighted Sum of entropies of Subsets.

⇒ In real world, entropy set & gini index are used.

⇒ Check Entropy diagrams. (slides).



⇒ In case of numeric data — we distinguish on the basis of numerical value. [$x > x_4$; $x < x_2$ — etc.].

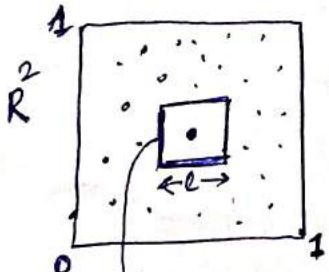
When do we Stop?

We use validation sets [Best Way].

→ more partition will lead to overfitting.

SMAI (21/01/2020)

Curse of Dimensionality



let, N points (≈ 1000) in a sq. region –
find K nearest neighbours (k = 10).
→ uniformly distributed (not dense or sparse at any pt.).

$\frac{K}{N} \approx \ell^2$

$\ell \approx \left(\frac{K}{N}\right)^{1/2}$ → 2 dimensions.

[Total area = 1].

$\boxed{\frac{K}{N} \approx \ell^d}$ → d dimensions.

↳ hypercube

$\boxed{\ell = \left(\frac{K}{N}\right)^{1/d}}$.

what should be the size on avg., ℓ can have K points.

KNN assumption – points which are closer are more likely to be part of same class.

d	l
2	0.1
10	0.63
100	0.995
1000	0.9954

→ $N = \frac{k}{l^d}$ if $l = 1/10$ ⇒ $\boxed{N = K \cdot 10^d}$ data points req. for 10% of space.

→ As d increases, area req. to find k points increases.

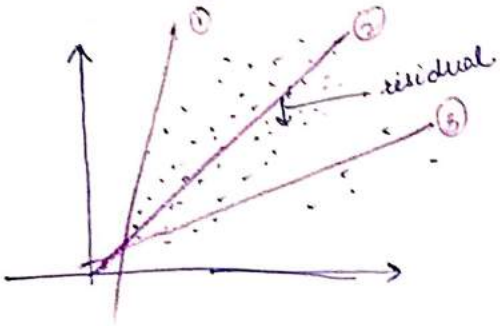
→ More amount of data req. for training.

Hence, called "curse" of dimensionality.

⇒ But, still KNN is used & give good result as data is not uniformly distributed. Data lies on many fold.

⇒ Sometimes, KNN won't work, this may be the reason.

Linear Regression



Which line describes the data best?

⇒ find sum of distance of points from line. The line from which sum is least is best line.

→ find a line, sum of errors is minimum.

$y = w^T x$.

$y = w_0 + w_1 x_1 + w_2 x_2^2 + w_3 x_3^3 \cdots w_d x_d$.

Still, $\boxed{y = w^T x}$.

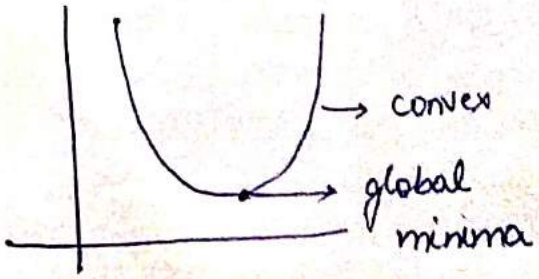
→ Output can be predicted as weighted sum of ~~distances~~ inputs.

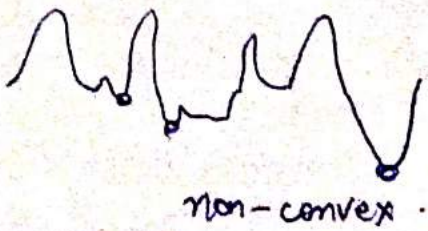
$y = mx + c$ [w → set of parameters vector.]

$$\boxed{J(w) = \sum \epsilon_i^2 = \sum_{\forall i} (w^T x_i - y_i)^2} \rightarrow \text{cost function}$$

→ our task is to minimize this cost fn.

→ Eventually, we have to find set of w.





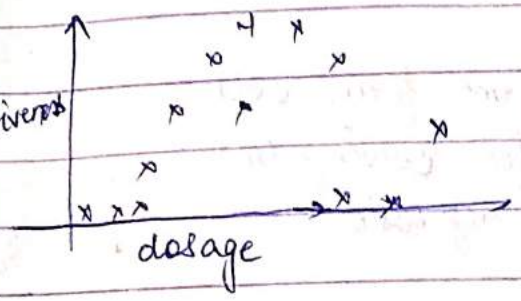
→ polynomial fitting can be done using linear regression.

→ x needs to be linear, y can be anything.

for x_i → output y_i.

$\sum_{\forall i} (w^T x_i - y_i)^2$ → error to be minimized.

⇒ Where Linear Regression can't be used?



Here, no line can fit this.
So, linear Regression can't be used in such cases.

→ Solve for J(w) which minimizes J(w).

$$J(w) = \frac{1}{2} \sum_i (w^T x_i - y_i)^2.$$

$$A = \begin{bmatrix} - x_1^T - \\ - x_2^T - \\ - x_3^T - \\ \vdots \\ - x_n^T - \end{bmatrix}_{N \times d} \qquad W = \begin{bmatrix} w_0 \\ w_1 \\ \vdots \\ w_{d-1} \end{bmatrix}_{d \times 1} \qquad Y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix}_{N \times 1}$$

$$J(w) = \frac{1}{2} (AW - Y)^T (AW - Y)$$

$$AW = \begin{bmatrix} x_1^T w \\ x_2^T w \\ \vdots \\ x_n^T w \end{bmatrix}$$

$$\begin{bmatrix} 1 & x_1^1 & x_1^2 & \cdots & x_1^d \end{bmatrix} \begin{bmatrix} w_0 \\ w_1 \\ \vdots \\ w_d \end{bmatrix} = w_0 + w_1 x_1 + w_2 x_2 \cdots w_d x_d = w^T x_i .$$

[Hence, $AW = w^T x_i$].

$\{(AW - Y)^T (AW - Y) = (AW - Y)^2\}$

$$\frac{\partial J}{\partial W} = \frac{1}{2} \frac{\partial}{\partial W} \left[(AW - Y)^T (AW - Y) \right]$$

$$= \frac{1}{2} \frac{\partial}{\partial W} \left[(AW)^T AW - (AW)^T Y - Y^T AW + Y^T Y \right]$$

$P = AW$; $Q = Y$, $P^T Q = Q^T P$

$$= \frac{1}{2} \frac{\partial}{\partial w} \left[(Aw)^T Aw - 2(Aw)^T Y + Y^T Y \right].$$

$$= \frac{1}{2} \frac{\partial}{\partial w} \left[w^T A^T Aw - 2 w^T A^T Y + Y^T Y \right].$$

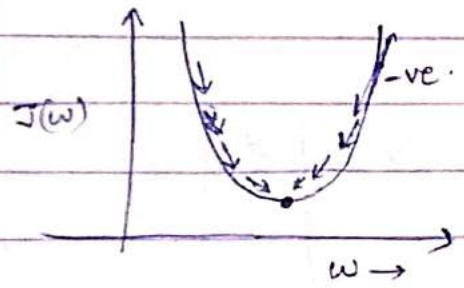
$$= \frac{1}{2} \frac{\partial}{\partial w} \left[2 A^T A w - 2 A^T Y \right]$$

$$\Rightarrow A^T Aw = A^T Y$$

$$\boxed{w = (A^T A)^{-1} A^T Y}$$ → normal form soln. for solving linear regression.

↓ pseudo inverse.

Gradient descent



$J(w)$ is differentiable

→ if w is large, may not get best soln.

→ if w is small, time taken is large.

→ we find gradient and go in negative direction to reach 0.

$$w_{t+1} \rightarrow w_t - \lambda J'(w_t).$$

Taylor Series :

$$J(w+h) \simeq J(w) + h J'(w) + \cdots$$

$$\simeq J(w) + h J'(w)$$

$$w_{t+1} = w_t - \lambda J'(w).$$

$$h = -\lambda J'(w)$$

$$J(w_t - \lambda J'(w)) \simeq J(w) - (+\lambda J'(w)) J'(w).$$

$$J(w_{t+1}) \simeq J(w) - \lambda (J'(w))^2.$$

$$\boxed{J(w_t + 1) \leq J(w)}$$

Gradient descent always work till curve is convex.

SMAI (25/01/2020)

$w_{t+1} \leftarrow w_t - \lambda J(w_t)$

$\Rightarrow$ Amount of step depends on λ and $J(w)$.

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

$$J'(w) = \sum_{i=1}^{N} x_i (w^T x_i - y_i)$$

{ λ bigger – bigger rate
λ smaller – smaller rate }

Algo:

① Choose a learning rate, $\lambda \in R^+$

② Choose a random W, $w \in R^{d+1}$

③ Repeat 4 till conversions.

④ $w = w - \lambda \sum_{i=1}^{N} x_i [w^T x_i - y_i]$

← Batch gradient descent Algo.

↳ Problem – we have to go through all data once to make one step.

Repeat for all samples –

$w := w - \lambda x_i (w^T x_i - y_i)$

← stochastic gradient descent

↳ Problem: w will fluctuate a lot; data is noisy. So cost fluctuates a lot.

Mini Batch Gradient Descent

Batches of samples from the training set (eg - 32 samples)

$$w = w - \lambda \sum_{i=1}^{32} x_i (w^T x_i - y_i)$$

Finish one pass over the entire data] epoch.

* Extrapolation - don't always work.
* Overfitting – avoid it.

Regularization (Imp.)

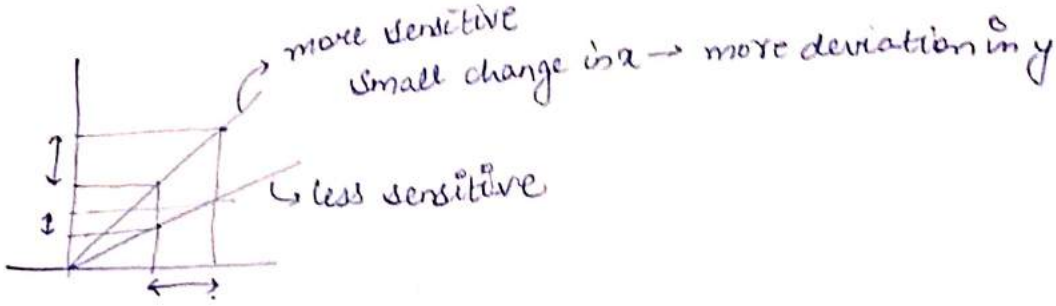
$$J(w) = \sum_{i=1}^{N} (w^T x_i - y_i)^2 + \left(\sum_{i=1}^{d} w_i^2 \right)$$

→ Norm of weights $L_p = (|w_i|^p)^{1/p}$.

[J'w can still be computed efficiently.]

$$J(w) = \sum_{i=1}^{N} (w^T x_i - y_i)^2 + \lambda \sum_{i=1}^{d} \|w_i\|_2$$

→ for L_2 norm.



$$J(w) = \sum_{i=1}^{N} (w^T x_i - y_i)^2 + \lambda \sum_{i=1}^{d} \|w_i\|_2 .$$

$$y_i = w_0 + w_1 x + w_2 x^2 - - w_{25} x^{25}$$

↳ very large than

Without regularization [$\sum (w^T x_i - y_i)^2$ will be small but other term is very large.

⇒ With regularization, term 2 can't be large, so suppose it is modest, soln. is slightly worse but it's still small.

There is tradeoff.

⇒ Regularization helps reduce overfitting.

⇒ Ridge Regression = Normal Regression + l_2 norm regularization.

⇒ Lasso Regression = Normal regression + l_1 norm regression.

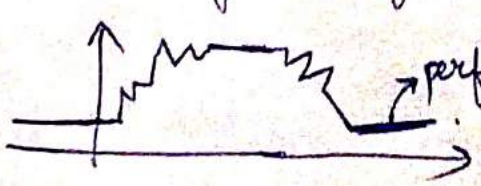
$$J(w) = \sum_{i=1}^{n} (W^T x_i - y_i)^2 + \lambda \sum_{i=1}^{d} |w_i|$$

↓ works as feature selection.

↓ hyperparameters — found using cross-validation

{contour?}.

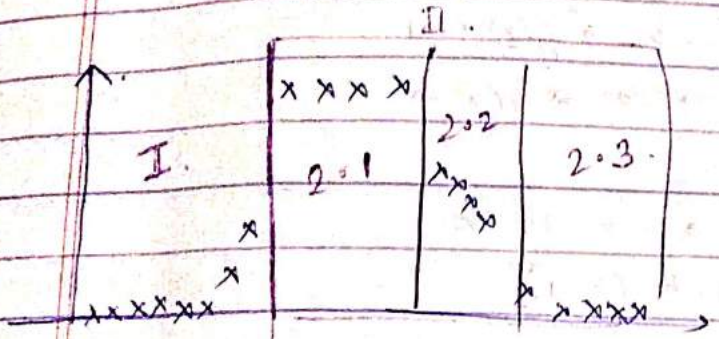
⇒ lasso can give perfect slope but ridge doesn't give.



ridge can't give this.

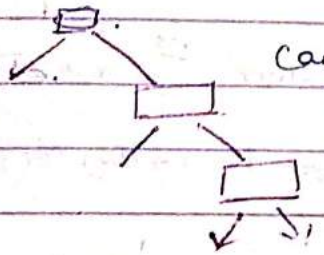
Elastic Net Regression

$$\sum (w^T x_i - y_i)^2 + \lambda_1 \sum \|w_i\|_2 + \lambda_2 \sum |w_i|_1$$



Here regression can't be used. No curve can fit effectively.

We make partitions. → We use decision trees



can be predicted using decision trees.

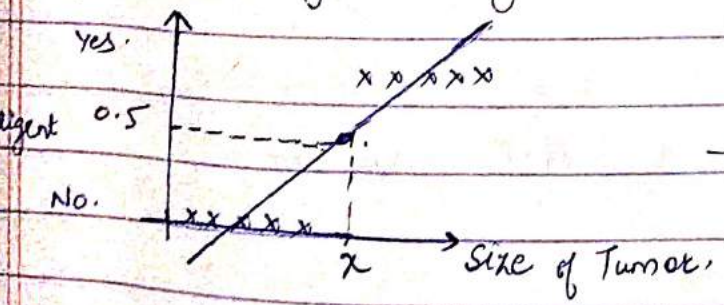
SMAI (28/01)

Logistic Regression

— classification approach

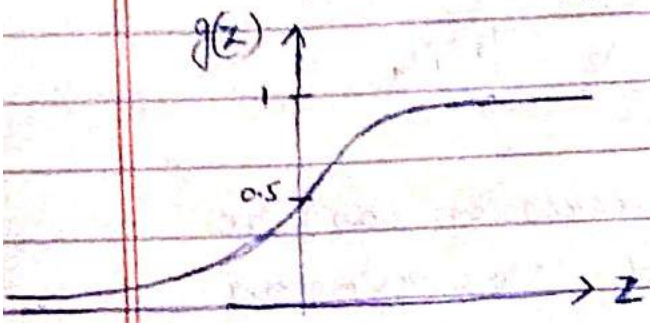
classification problems — (i) spam or not (2 class prob.)
(ii) Spam, primary, promotion (multi class)
(iii) Tumor - malign or benign
(iv) Transaction - genuine or fraud.

logistic — logit — Sigmoid



How to make this problem a classification problem?
→ we decide a threshold x and calculate y at it. on line.
Now for any x, if $y(x)$ & > threshold, we classify it as 'Yes' else classify it to 'No' label.

$$g(z) = \frac{1}{1+e^{-z}}$$ ← Sigmoid fn.



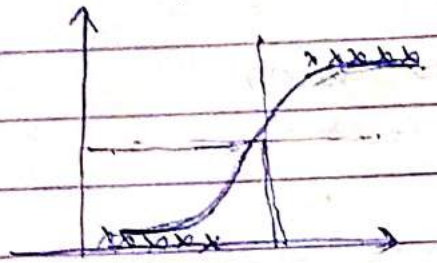
$z = 0 \Rightarrow g(z) = 0.5$

$z \to \infty \Rightarrow g(z) \simeq 1$

$z \to -\infty \Rightarrow g(z) \to 0$.

Fn. nice. [$\mathbb{R} \to -\infty$ to ∞.
output $(0,1)$.

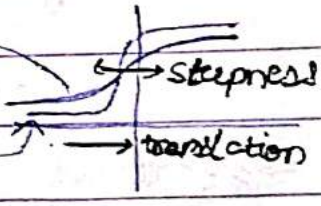
Now for our problem, we can make some shifted sigmoid fn.



$$\frac{1}{1+e^{-wx}} \simeq \frac{1}{1+e^{(-wx+w_0)}}$$

$\Rightarrow$ w can be controlled.

→ if w is small — curve smooth
w is large — curve steep.



let, $\sigma(x) = \frac{1}{1+e^{-x}}$

Then, $\sigma'(x) = \frac{+e^{-x}}{(1+e^{-x})^2} = \left(1 - \frac{1}{(1+e^{-x})}\right)\frac{1}{(1+e^{-x})}$

$= \sigma(x) \cdot [1 - \sigma(x)]$.

$$\boxed{\sigma'(x) = \sigma(x) \cdot (1 - \sigma(x))}$$

$\Rightarrow$ In sigmoid fn., we want to control steepness and translation mainly.

In linear regression → $J(w) = \sum_i (y_i - w^T x_i)^2$

logistic Regression → $J(w) = \sum_i [y_i - \sigma(w^T x)]^2$ → not a convex function.

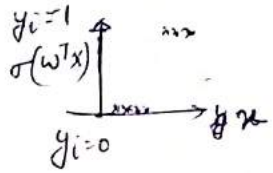
Maximum likelihood Estimation

$$P(y_i = 1 \mid x_i, w) = \sigma(w^T x_i)$$

$$P(y_i = 0 \mid x_i, w) = 1 - \sigma(w^T x_i)$$

2 class classification.
{ $y_i \in 0$ or 1 as it can belong to a class or not, no in blw }

$$\underbrace{P(y_i \mid x_i, w)}_{\text{likelihood}} = \underbrace{(\sigma(w^T x_i))^{y_i}}_{y=0\ X} \times \underbrace{(1 - \sigma(w^T x_i))^{(1-y_i)}}_{y=1\ X}$$



Maximize this — our goal → bcz. when $y_i = 0$, we want $\sigma(w^T x_i)$ to be close to 0.

→ To maximize 1st term → minimize $(1 - \sigma(w^T x_i))$. as at $y_i = 0$ 1st term doesn't come into play

at $y = 1$ ⇒

Maximize P ⇒ Is it a right goal?

$y_i = 0 \rightarrow \sigma(w^T x)$ is close to zero

$y_i = 1 \rightarrow \sigma(w^T x)$ is close to 1

So, by maximizing single fn. P → we achieve our goal. Hence, a good idea.

$$\boxed{L(w) = \prod_{i=1}^{N} P(y_i \mid x_i, w)} \rightarrow \text{maximize this.}$$

→ finding parameters for w → turns to → maximizing above fn.

→ find set of w where above fn. is maximized, we are done.

$$\log(L(w)) = \log \prod_{i=1}^{N} P(y_i \mid x_i, w)$$

$$G(w) = \log(L(w)) = \sum_{i=1}^{N} \log(P(y_i \mid x_i, w)).$$

$$= \sum_{i=1}^{N} \log\left((\sigma(w^T x_i))^{y_i} \times (1 - \sigma(w^T x_i))^{(1-y_i)}\right)$$

$$G(w) = \sum_{i=1}^{N} y_i \log(\sigma(w^T x_i)) + (1 - y_i) \log(1 - \sigma(w^T x_i)).$$

$$G'(w) = \sum_{i=1}^{N} y_i \cdot \frac{1}{\sigma(w^T x_i)} \cdot \sigma'(w^T x_i) \cdot x_i + \frac{(1-y_i)}{(1-\sigma w^T x_i)} \sigma'(w^T x_i) x_i$$

$$= \sum_{i=1}^{N} \left(\frac{y_i}{\sigma(w^T x_i)} - \frac{(1-y_i)}{1-\sigma(w^T x_i)} \right) \times \sigma'(w^T x_i) \cdot x_i$$

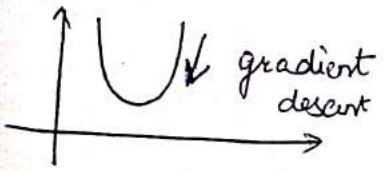
$$= \sum_{i=1}^{N} \left(\frac{y_i}{\sigma(w^T x_i)} - \frac{(1-y_i)}{1-\sigma(w^T x_i)} \right) \sigma(w^T x_i)\left(1-\sigma(w^T x_i)\right) \cdot x_i$$

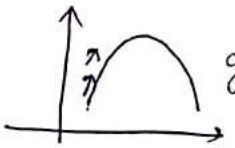
$$= \sum_{i=1}^{N} \left(y_i\left(1-\sigma(w^T x_i)\right) - (1-y_i)\sigma(w^T x_i) \right) x_i$$

$$= \sum_{i=1}^{N} \left(y_i - y_i \cdot \sigma(w^T x_i) - \sigma(w^T x_i) + y_i \sigma(w^T x_i) \right) x_i$$

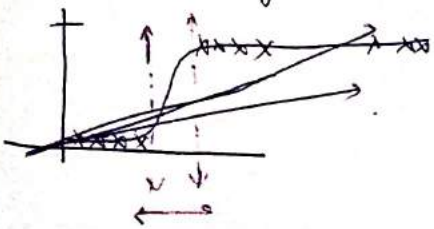
$$G'(w) = \sum_{i=1}^{N} \left(y_i - \sigma(w^T x_i) \right) x_i$$

→ Goal was to ~~minimize~~ maximize $G(w)$.





→ In case of logistic regression, outliers may not affect much as in case of linear regression.



gray area — unsure → logistic regression ~~do~~ gives clear idea of gray area.

{After training, $\sigma(w^T x) > 0.5$ — class 1
$\sigma(w^T x) \leq 0.5$ — class 2.}

2 Class classification

one vs one

4 classes → ${}^4C_2$ classifiers.

A, B, C, D.

A vs B, B vs C, A vs C, B vs D, A vs D, C vs D.

] run these classifiers & take decision.

Multi class classification

one vs all.

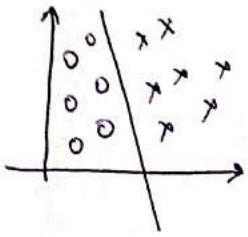
C_1 → A vs B, C, D.
C_2 → B vs A, C, D.
C_3 → C vs A, B, D
C_4 → D vs A, B, C.

} run all & take decision

SMAI (01/02/2020)

SVM (Support Vector Machines)

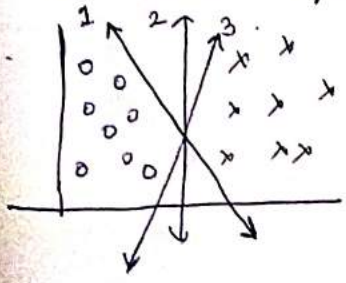
→ linearly separable data (assumption)



Linear classification in 2-D : line (2 parameters) — learning
in 3-D : plane (3 parameters).

(We assume no noise for now)

→ Multiple soln. exist for linearly separable data.
Are all soln. equally good?

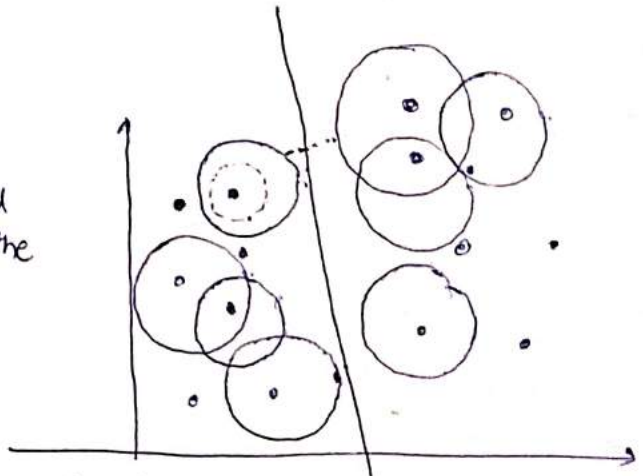


→ line 2 is better as it gives more margin.

Margin : bubbles around samples.

Margin – radius of a region around each training sample, through which the decision boundary can't pass.

As margin increases, the feasible region reduces.

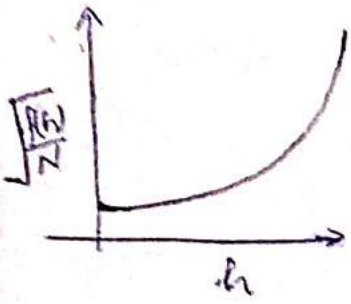


→ Keeping the line away from bubble.
→ increasing size of bubble.

Goal : Line which divides data in two parts (assuming linearly separable data. We have to find a line which divides data and maximizes margin. [Algo called SVM])

Bound on Expected loss $= R(\alpha) \leq R_{train}(\alpha) + \sqrt{f(h)/N}$

h is VC dimension and $f(h) = h + h\log(2N) - h\log(h) - c$.



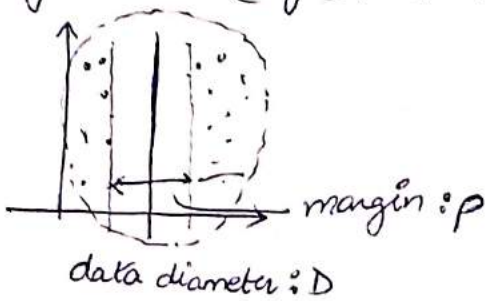
→ monotonically increasing fn.
→ taking lower value of h gives lower value of $\sqrt{f(h)/N}$

Why maximise the margin ?

→ To reduce test error, keep training error low (say 0), and minimize the VC dimension, h.

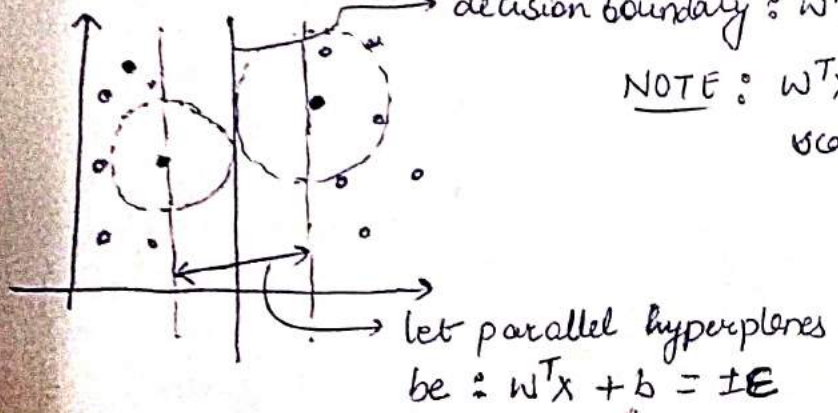
Relative margin = ρ/D

VC-D, $h \leq \min\left\{d, \left[\frac{D^2}{\rho}\right]\right\} + 1$



→ Maximizing margin improves generalization
→ h can be made independent of the dimensionality : d.

Formalizing the margin



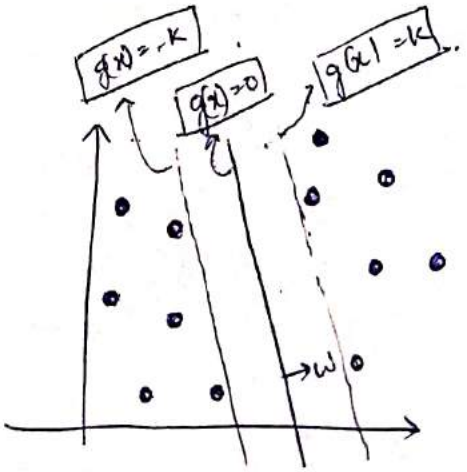
NOTE : $w^T x_i + b$ is dependent on the scale of X and W.

Formulation

→ Let $g(x) = w^T x + b$.
→ We want to maximize such that

$-w^T x_i + b \geq k$ for $d_i = 1$.

$-w^T x_i + b \leq -k$ for $d_i = -1$.

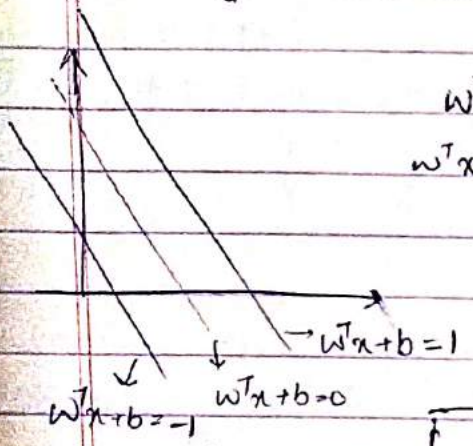


→ Values of $g(x)$ depends on $||w||$:

1. Keep $||w|| = 1$, and maximize $g(x)$, or.
2. Let $g(x) \geq 1$ and maximize $||w||$.

→ We use two approaches and formulate the problem as:

1. Minimize : $\frac{1}{2} w^T w$.
2. Subject to : $d_i(w^T x_i + b) \geq 1$, for $i = 1 \ldots N$.



$w^T x_{pos} + b = 1$ — (1)

$w^T x_{neg} + b = -1$ — (2).

$w^T (x_{pos} - x_{neg}) = 2$.

$\frac{w^T}{||w||} (x_{pos} - x_{neg}) = \frac{2}{||w||}$ ← margin = 2/norm(w)

→ **minimize** $\frac{1}{2} w^T w$.
subject to $d_i (w^T x_i + b) \geq 1 \quad \forall i$.

Quadratic fn. : QP solvers.

↳ idea to optimization problem.

d_i = +1 → +ve → for one class.
d_i = −1 → −ve → for other class.

SMAI (04/02/2020)

→ $J(w, b, \alpha) = \frac{1}{2} w^T w - \sum_{i=1} \alpha_i d_i (w^T x_i + b) + \sum \alpha_i$

s.t. $\alpha_i \geq 0 \quad \forall i$

Lagrangian form.

Minimize wrt w & b and max. wrt α.

→ $w^* = \sum_{i=1}^{N} \alpha_i d_i x_i$ ← $\frac{\partial J}{\partial w} = 0$

$\sum_{i=1}^{N} \alpha_i d_i = 0$ ← $\frac{\partial J}{\partial b} = 0$

$$\alpha_i \left(d_i \left(w^T x_i + b^* \right) - 1 \right) = 0$$

$$Q(\alpha) = \sum \alpha_i - \frac{1}{2} \sum\sum \alpha_i \alpha_j d_i d_j x_i^T x_j$$

α_i

$$w^* = \sum \alpha_i d_i x_i \; ; \; b^* = 1 - w_0^T x_{st}$$

→ Now model is deployed.

$$\left\{ \text{sign} \left(w^T x_t + b^* \right) \longrightarrow \begin{array}{l} \text{+ve : class 1} \\ \text{-ve : class 2.} \end{array} \right\}$$

QP solver

↓

α_i

↓

$$w_0 = \sum_{i=1}^{N} \alpha_i d_i x \longrightarrow \alpha_i \left[d_i \left(w_0^T x_i + b_0 \right) - 1 \right] = 0 \longrightarrow b_0 = 1 - w_0^T x_{st}$$

where α is non zero in SV.

How to modify current program statement to support noise?

$$\frac{1}{2} w^T w .$$

$$\text{s.t.} \quad d_i \left(w^T x_i + b \right) \geq 1 \quad \forall i .$$

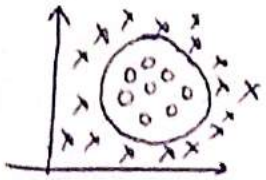
ξ_i → slack variable.

modified contraint :

$$d_i \left(w^T x_i + b \right) \geq 1 - \xi_i \quad \forall i$$

Non - Separable Data

1. Noisy data / Bad features
2. Non - linear boundary.



SVM with noisy data

Minimize : $\phi(w) = \frac{1}{2} w^T w$.

Subject to : $d_i (w^T x_i + b) \geq 1 - v_i$

↓

Introduce slack variables $\xi_i \geq 0$

↓

Minimize : $\phi(w) = \frac{1}{2} w^T w$.

Subject to : $d_i (w^T x_i + b) \geq 1 - \xi_i \quad \forall i$

↓

Minimize : $\phi(w) = \frac{1}{2} w^T w + C \sum_{i=1}^{N} \xi_i$

Subject to : $d_i (w^T x_i + b) \geq 1 - \xi_i \; ; \; \xi_i \geq 0 \; \forall i$

Dual form with Slack Variables.

$$Q(\alpha) = \sum \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j d_i d_j x_i^T x_j$$

Subject to $\alpha_i \geq 0$ and $\sum_{i=1}^{N} \alpha_i d_i = 0$.

↓

$$Q(\alpha) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{i=1}^{N} \alpha_i \alpha_j d_i d_j x_i^T x_j .$$

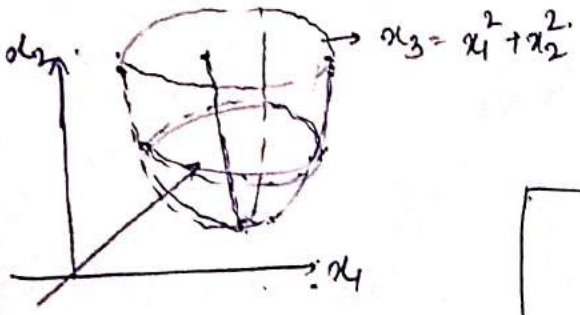
Subject to $0 \leq \alpha_i \leq C \quad \forall i$ and $\sum_{i=1}^{N} \alpha_i d_i = 0$.

→ Note that neither the slack variables, nor their lagrange multipliers appear in dual.

→ The only change is the additional constraint on α_i.

→ The parameter C controls the relative weight between training error and the VC dimension.

Non-linear Boundaries



$$\phi(x) = \begin{pmatrix} x_1 \\ x_2 \\ x_1^2 + x_2^2 \end{pmatrix}$$

SVM post mapping.

Training —

$$Q(\alpha) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j d_i d_j \, \phi(x_i) \cdot \phi(x_j)$$

Subject to $0 \le \alpha_i \le C \ \forall i$ and $\sum_{i=1}^{N} \alpha_i d_i = 0$.

Testing — for testing also we have to transform point to 3-D and then check.

dot product

$$\text{Label} = \text{sign}\left(w_0 \cdot \phi(x_{test}) + b_0\right)$$

$$w_0 = \sum_{i=1}^{N} \alpha_i d_i \phi x_i$$

$$\therefore \text{label} = \text{sign}\left(\left(\sum_{i=1}^{N} \alpha_i d_i \phi(x_i) \cdot \phi(x_{test})\right) + b_0\right)$$

SVM post Mapping

→ Data vectors occur only as dot products in SVM - learning & testing.

→ If we can find a fn. $K(X, Y)$, which is equivalent to $\phi(X) \cdot \phi(Y)$, we can avoid

$$Q(\alpha) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{i=1}^{N} \alpha_i \alpha_j d_i d_j K(x_i, x_j)$$

$$\text{Label} = \text{sign}\left(\sum_{i=1}^{N} \left(\alpha_i d_i K(x_i, x_{test})\right) + b_0\right).$$

What do we gain by using K?

original space : 2-dimensional

let $K(X, Y) = (X \cdot Y)^3 = (x_1 y_1 + x_2 y_2)^3$

let $\phi(x) = \phi\left(\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}\right) = \begin{bmatrix} x_1 x_1 \\ x_1 x_2 \\ x_2 x_1 \\ x_2 x_2 \end{bmatrix}$

Let $K(X,Y) = \phi(X) \cdot \phi(Y) = \begin{bmatrix} x_1^2 \\ x_1x_2 \\ x_2x_1 \\ x_2^2 \end{bmatrix} \cdot \begin{bmatrix} y_1^2 \\ y_1y_2 \\ y_2y_1 \\ y_2^2 \end{bmatrix}$

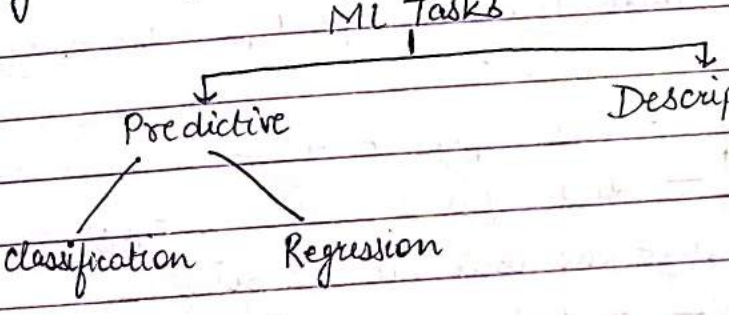
$= x_1^2y_1^2 + 2x_1x_2y_1y_2 + x_2^2y_2^2$

$= (x_1y_1 + x_2y_2)^2$

$= (X \cdot Y)^2$

Unsupervised learning – Clustering (15/02/2020)

→ clustering news of same type (topic) – Google news.



ML Tasks :: Descriptive –

→ Study / Exploit the 'structure' of data.
- clustering
- Dimensionality Reduction
- Density Estimation.

→ Also called as 'unsupervised learning'
- Input data without paired output.

→ cluster images – sea, sunset, flowers etc.

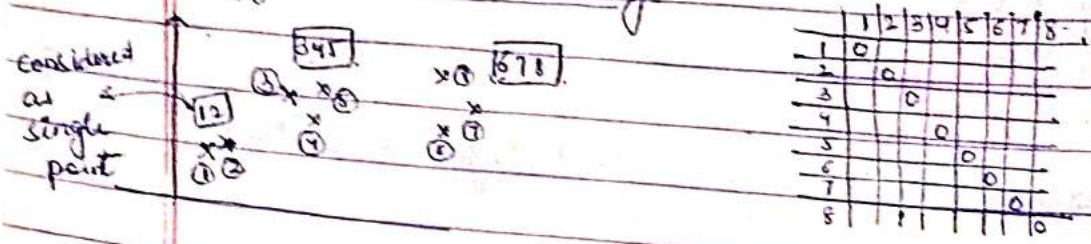
→ Determine groups of people in image
- based on clothing styles & features.
- gender, age etc.

→ Determine moving objects in videos.

What is clustering?

→ high intra-group similarity (within members of a cluster)
→ low inter-group similarity (across clusters).

Agglomerative Clustering



	1	2	3	4	5	6	7	8
1	0							
2		0						
3			0					
4				0				
5					0			
6						0		
7							0	
8								0

Symmetric matrix.

⇒ we calculate distance of every point with every other point.

→ we find closest point; make 12 as single point; 12 can be defined as mean of 1 & 2.

→ Average link clustering — we represent cluster as mean of points.

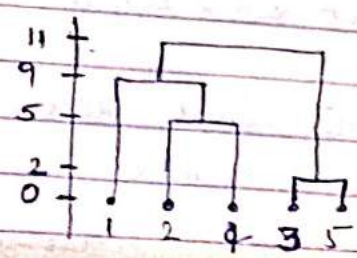
→ Single link clustering — dist. b/w 2 clusters is equal to dist b/w their closest point.

→ Complete link clustering — dist. b/w 2 clusters is the dist b/w their farthest point.

3 ways of agglometric clustering based on distance computation.

⇒ Start with clusters equal to no. of points and then keep merging them.

	1	2	3	4	5
1	0				
2	9	0			
3	3	7	0		
4	6	5	9	0	
5	11	10	2	8	0



→ Dendogram.

	35	1	2	4
35	0			
1	11	0		
2	10	9	0	
4	9	6	5	0

$$J = \sum_{k=1}^{K} \sum_{i=1}^{n_k} \|x_{ki} - \mu_k\|^2$$ ← to minimize in case clustering.

Beehive clustering

→ decide threshold = 0.3 (suppose).

→ c_i when new pt. comes, compute distance from c_i & either give it to same cluster or form a new cluster.
Done for every new point. Compute distance from each cluster & work around.

→ faster computationally.

→ Sir ki khoj.

⇒ In clustering, distance metric to be used is also crucial. (cosine, euclidean, ~~mean ck~~).

→ clustering and then KNN. ⇒ very effective.

K-Means Clustering

Approach 1: Assume that we (somehow) know no of clusters.

→ # of clusters = k (often a 'guess').

→ How to represent a cluster?
- Cluster members
- A suitable statistic of the cluster members.

K-Means clustering: ① Initialization → decide k, and initialize k centers (randomly).
→ Take a point, see which chosen point is it closest to and assign it to that cluster.

② Iteration 1 — assign all objects to the nearest center; move a center to mean of its members

③ Iteration 2 — After moving centers, reassign the objects to nearest centers. Move a center to the mean of its new members.

Reassign and move centers; until no objects changed membership.

Time complexity = $O(k \cdot n)$. ← practically faster; k means ~~do~~ can't always converge but agg. converges theoritically.

{for agglomerative clustering — TC = $O(n^2)$}.

$\{x^{(1)} \text{---} x^{(m)}\}$; $x^{(i)} \in R^n$.

The k-means clustering algo is as follows:

① Initialize cluster centroids $\mu_1, \mu_2 \text{---} \mu_k \in R^n$ randomly.

② Repeat until convergence. {

for every ~~set~~ i, set

$$c^{(i)} = \arg\min_j \|x^{(i)} - \mu_j\|^2$$ → assignment step: assign each data pt to closest cluster.

for each j, set

$$\mu_j = \frac{\sum_{i=1}^{m} 1\{c^{(i)} = j\} x^{(i)}}{\sum_{i=1}^{m} 1\{c^{(i)} = j\}}$$

← Refitting ~~opp~~ step: Move each cluster center to the center of the data assigned to it.

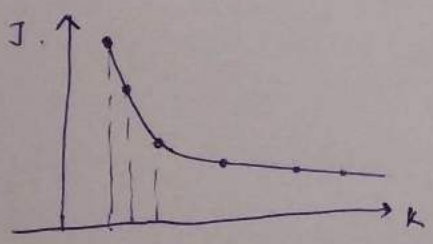
}.

Why k-means Works?

→ high intra cluster similarity.

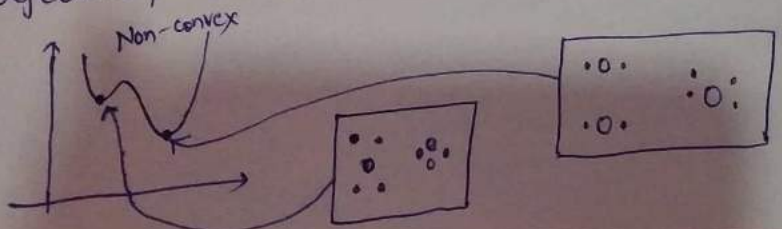
→ k-means optimizes → $J = \sum_{k=1}^{k} \sum_{i=1}^{n_k} \|x_{ki} - \mu_k\|^2$ → closed clustering - this term will be decreased (dist from mean).

How to decide K?



Plot for diff values of k & choose k where elbow appears generally.

⇒ Objective function for k-means is non-convex



⇒ for k, Try with diff. initialisations & pick k which gives lowest J.

SMAI (01/03/2020)

Initialization

① (Sample a small set of points
⇓
Agglomerative clustering).

K-means ++

① Randomly choose one of the observation to be a cluster center.
② For each observation x, determine $d(x)$, where $d(x)$ denotes the minimal distance from x to a cluster center.
③ Choose next cluster center; $(d(x)^2)$
④ Repeat 2 & 3 till you have k clusters.

SVM Kernels.

→ experiment common Kernels & find what works best.

ML - A probabilistic perspective. (Generative Modelling)

→ Mean, variance, co-variance, expectation etc.
→ Gaussian dist
→ Sampling from a dist

$$\text{Cov.} = \sum (x-\mu_x)(y-\mu_y)$$

cov. matrix =

	x	y
x	σ_x^2	σ_{xy}^2
y	σ_{xy}	σ_y^2

$$\text{Expectation} = E(x) = \sum_i x_i P(x = x_i)$$

$$E[(x-\mu)^2] = \sum (x_i - \mu)^2 \cdot P(x_i)$$

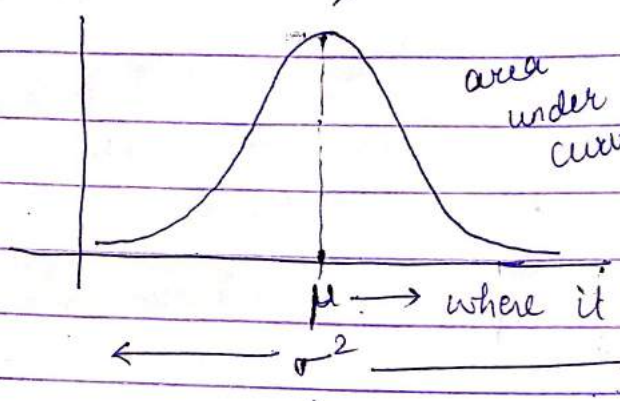
$$V(x) = E[x^2] - (E[x])^2$$

Gaussian distribution (Normal)

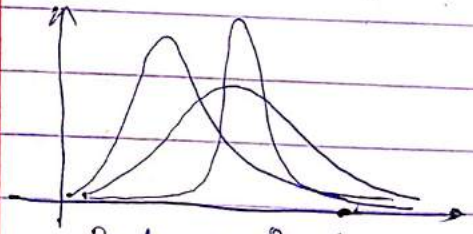
$$f(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{\frac{-(x-\mu)^2}{2\sigma^2}}$$

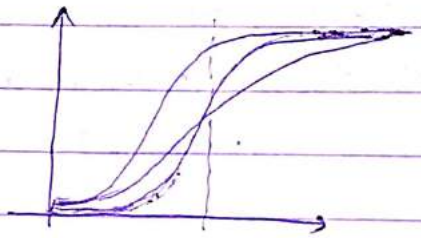
} Normal Gaussian

shows how likely a data can appear.



Sample from a gaussian distribution →





→ Generative vs discriminative classification
→ density estimation – anomaly detection
→ Gaussian distribution (univariate & multivariate)
→ Generative classification (Naive Bayes)

SVM – discriminative classification, we discriminate on the basis of hyperplane. (divide).

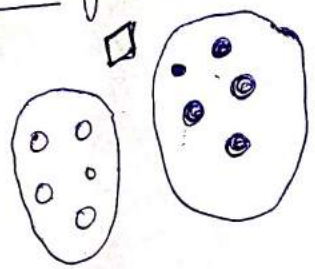
Generative – fitting a distribution to each class.

Given a ~~distribution~~ ~~class~~ sample, we try to find which ~~class~~ discrimination it belongs

Not discriminating, doing generative modelling.

→ fit some density fn. to each class.

Density Estimation



Compute $\begin{cases} p(x/y_1) \\ p(x/y_2) \end{cases}$

→ Given a point, how do I fit a fn. to it.

2-D Gaussian.

→ ~~Mode~~ Most of the processes we see in real world, is actually a cumulation of many random events.

→ As you take many processes and sum their effect, you tend towards a gaussian fn.

Anomaly Detection

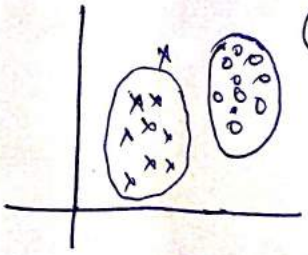
Is everything fine with my aircraft? — 2 class classification prob. less negative data present, calculate prob. based on some values and then set threshold like if pdf values is less than 0.1, than perform manual check.

→ Multivariate Gaussian Distribution.

$$f(x) = \frac{1}{(2\pi)^{n/2} |\Sigma|^{1/2}} e^{\left(-\frac{1}{2}(x-\mu)^T \Sigma^{-1} (x-\mu)\right)}.$$

SMAI (04/03)

likelihood.



$p(x/y_1)$ $p(x/y_2)$ — we can calculate this.

~~discrimati~~ discriminative classification — we are learning to discriminate b/w the two

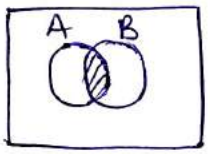
Generative classification — model each of the class & based on model, likelihood. Compute.

$p(y_i/x)$ $\quad p(x/y)$ ⟶ from pdf.

decision rule — $\underset{i}{\text{argmax}}\ p(y_i/x)$

How to compute $p(x/y)$ — use Bayes's Rule.

$$P(y/x) = \frac{P(x/y) \cdot P(y)}{P(x)}$$



$$P(A/B) = \frac{P(A \cap B)}{P(B)} \quad ; \quad P(B/A) = \frac{P(A \cap B)}{P(A)}$$

$$P(A/B) \cdot P(B) = P(B/A) \cdot P(A)$$

$$\Rightarrow \text{Bayes Rule} - P(A/B) = \frac{P(B/A) \cdot P(A)}{P(B)}$$

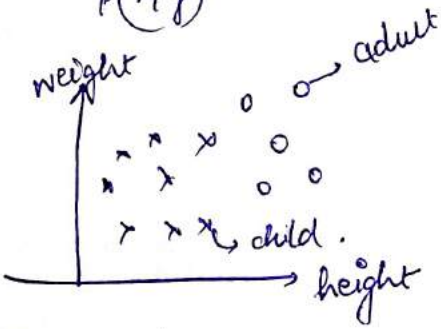
⇒ Decision — comb. of what we observe & what priors we have.

Naive Bayes

$$P(x_1, x_2 - x_d/y) = \prod_{(i)}^{d} p(x_i/y)$$ → computing likelihood of each feature.

→ it fails when features depend on each other.

$p(x/y)$



Given height & weight, predict adult or child?

$$p(a/x) = \frac{p(x/a) \cdot p(a)}{p(x)}$$

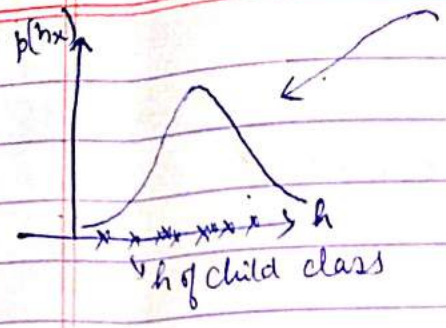
$$p(c/x) = \frac{p(x/c) \cdot p(c)}{p(x)}$$

{adults = 4, child = 12}

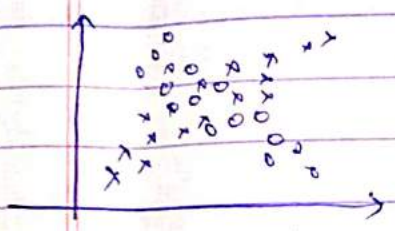
$p(a) = 0.25$; $p(c) = 0.75$

$p(x/c) = p(h_x/c) \cdot p(w_x/c)$; $p(x/a) = p(h_x/a) \cdot p(w_x/a)$

$x = [h_x, w_x]$



$$p(c/x) = \frac{p(x/c) \cdot p(c)}{p(x/a) \cdot p(a) + p(x/c) \cdot p(c)}$$



→ Naive Bayes will fail completely in this case.

⇒ Multinomial Naive Bayes — Text Classification.

email → Normal
email → Spam

[Once you model a class you can sample that class. No ovr, ovo req.].

[dear, friend, lunch, money]

Normal — P(dear/N) = 0.47.
P(friend/N) = 0.29
P(lunch/N) = 0.18
P(money/N) = 0.06
P(N) = 0.67.

Spam — P(dear/S) = 0.29
P(friend/S) = 0.14
P(lunch/S) = 0
P(money/S) = 0.51
P(S) = 0.33

Based on training data.

E-mail ⇒ "Dear Friend" P(N/x) P(S/x).

= p(x/N) · P(N).
= p(dear/N) · P(friend/N) · P(N)
= 0.47 × 0.29 × 0.67. = 0.09 ∝ P(N/ dear friend).

$= p(x/s) \cdot P(s)$

$= p(\text{dear}/s) \cdot p(\text{friend}/s) \cdot P(s)$

$= 0.29 \times 0.14 \times 0.33 = 0.01 \propto P(s/\text{Dear friend})$.

→ So, we classify it to Normal Class.

→ Use words from vocabulary only; don't use is, am, an, for classification.

Q. E-mail : Lunch Money Money Money.

$p(\text{lunch}/N) \cdot (P(\text{Money}/N))^3 \cdot P(N)$.

$= 0.00002$ (almost) $\propto P(N/x)$.

$p(\text{lunch}/s) \cdot (P(\text{Money}/s))^3 \cdot P(s)$.

$= 0. \qquad \propto P(s/x)$

→ should be classified to Spam but due to lunch being 0, didn't happen.

→ So, we assign min value (Suppose 0.01 is added everywhere).

Now, $(0.01) \cdot (0.58)^3 > 0.00002$ → Hence Spam.

Can you apply Naive Bayes to Textual data? (Gaussian)

No, bcz. there isn't continuous variable, we have categorical data.

Not used together:

- Naive Bayes — Works very good on textual data.
- decision trees — textual → — data.

challenge - Missing data - add avg.

→ prob may not add upto 1.

Text classification.

- Naive Bayes ✓
- Bag of Words

→ SVM not used - bcz numerical data; fixed dim. representation.

[Stop word removal - high freq words - is, an, all, the]

love, cinema, genius, happy, movie, Vocab. $\begin{bmatrix} 2 \\ 1 \\ 1 \\ 0 \\ \vdots \\ 2 \end{bmatrix}$ / words. $= \begin{bmatrix} 2/7 \\ 1/7 \\ 1/7 \\ 0 \\ \vdots \\ 2/7 \end{bmatrix}$ ← representation of textual data.

tf - idf = $tf(t_i D) \cdot idf(t_i D)$.

$tf(t_i D) = \log(1 + freq(t_i D))$

$idf(t_i D) = \log\left(\frac{N}{\text{no. of doc. having term } D}\right)$.

$\begin{matrix} w_0 \\ w_1 \\ \vdots \end{matrix} \begin{bmatrix} tf(w_0) \cdot idf(w_0) \\ tf(w_1) \cdot idf(w_1) \\ \vdots \end{bmatrix}$

→ words which occur rarely, gets more tf - idf value in this method.

→ bag of words is just normalized freq.

→ both donot consider order of words.

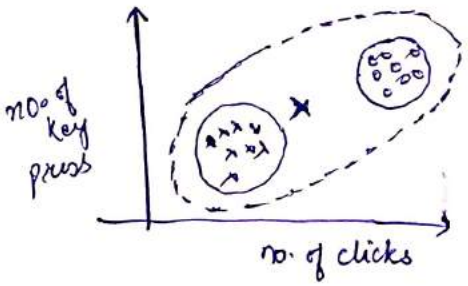
[bigram] [bigram] .

|

SMAI (08/03)

Gaussian Mixture Models

→ Anomaly Detection



Usual behaviour

→ real world data might not be a single gaussian.

That's why need GMM.

weights · → each gaussian

$$f(x) = \sum_c \pi_c \, N(x_i ; \mu_c, \sigma_c)$$

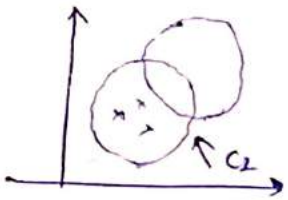
$$\{\Sigma \pi_c = 1\}.$$

weighted sum of gaussian.

K-means

→ hard assignment of clusters.

→ what if clusters are overlapping?



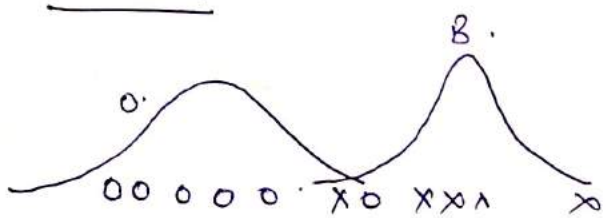
GMM

→ clusters are modeled as gaussians (not just by mean).

↳ trained using EM algo – generative model.

↳ you can sample from it.

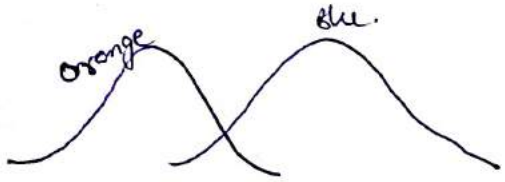
Scenario 1



→ if I know the source, I can estimate the (gaussians) parameters.

Scenario 2 – parameters of gaussian are given

→ I know the gaussians



$$P(b|x_i) = \frac{P(x_i|b) \cdot P(b)}{P(x_i|b) \cdot P(b) + P(x_i|a) \cdot P(a)}$$

$P(x_i|b)$ = substitute in formula

these prob. can be computed in this Scenario.

→ GMM assumes how many clusters. Suppose $K = 2$.

→ EM Algo – [Expectation Maximization Algo]

E → ⊙ Start with two randomly placed Gaussians $(\mu_a, \sigma_a^2), (\mu_b, \sigma_b^2)$

⊙ for each point x_i, compute $P(b/x_i)$ – does it look like it came from b.

$P(a/x_i)$ → " " " from a.

M → ⊙ Adjust (μ_a, σ_a^2) and (μ_b, σ_b^2) to fit points assigned to them.

→ Here Soft clustering, not hard like K-means.

Expectation Step → $P(x_i/b) = \frac{1}{\sqrt{2\pi\sigma_b^2}} e^{\frac{(x-\mu_b)^2}{2\sigma_b^2}}$.

$$b_i = P(b/x_i) = \frac{P(x_i/b)\,P(b)}{P(x_i/b)\,P(b) + P(x_i/a)\,P(a)}.$$

$$\boxed{a_i = 1 - b_i}$$

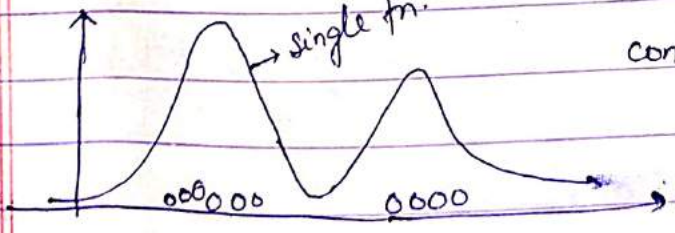
Maximization Step →

$$\mu_b = \frac{b_1 x_1 + b_2 x_2 \cdots b_n x_n}{b_1 + b_2 \cdots b_n}; \quad \sigma_b^2 = \frac{b_1 (x_1 - \mu_b)^2 + b_2 (x_2 - \mu_b)^2 \cdots}{b_1 + b_2 \cdots b_n}.$$

μ_a = ✓ ; σ_a^2 = ✓

$P(a) = \pi_a = \frac{a_1 + a_2 \cdots a_n}{N}$.

$$\boxed{\pi_a = 1 - \pi_b} \qquad \boxed{\sum \pi_c = 1}$$



convex combination of a set of gaussian.

→ C gaussian – $\underbrace{\pi_1, \pi_2 \cdots \pi_C}_{0.1 \;\; 0.1 \;\; 0.4 \cdots 0.1}$

$\pi_4 \rightarrow (\mu_4, \sigma_4)$. pick(find) a gaussian. know its parameters. So, sample point from that gaussian.

2-way sampling.

SMAI (15/03)

Principal Component Analysis (PCA) – for dimensionality reduction

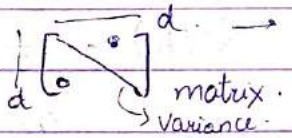
feature dimension – 1000 dimensional vectors as input.
↳ 98%.

→ unwanted data goes away
→ more structured data.

20 → 97%.

→ 98% accuracy.
bcz lot of unwanted columns are req.

Algo 1 – Covariance matrix

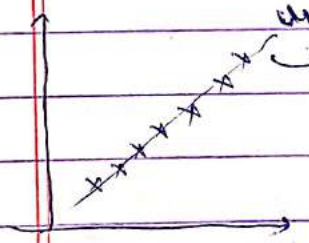


→ look at high values & keep one variable among highly correlated.

Algo 2 – classification – 1d, then add feature and then if acc. increase with best acc. then keep them.

Algo 3 – dropping from 100 d.

PCA – ~~defined~~ defines each dimension as linear weighted sum of other dimension
↳ compression



↳ 1-D data put in 2-D.
↳ PCA prob – How to retrieve this?
prob – to find axis on which to project.

→ if data is projected on u_1 – data reduced to 1-D.
↳ variance is maximized.

Maximizing Variance – leads to maximizing info.

{ entropy = $-(p_i \log p_i)$ }
↳ measure of some sort of randomness ; variation.

→ small direction, less variance, ~~high~~ better prediction.

$$\text{Covariance} \rightarrow \frac{1}{N-1} \sum_{i=0}^{N} (x_i - \mu_x)(y_i - \mu_y)$$

↳ each pair one by one.

$X = [x_1 \; x_2 \; \cdots \; x_n]_{d \times n}$. $\quad x_1 = \begin{bmatrix} \vdots \\ \vdots \end{bmatrix}_d$

(x_1 ↓ vector)

$$cov_{x_1 x_2} = \frac{1}{N-1} \sum_{i=1}^{N} (x_1^i - \mu_{x_1})(x_2^i - \mu_{x_2})$$

$$cov_{01} = \frac{1}{N-1} \sum_{i=0}^{N} (x_i^0 - \mu_0)(x_i^1 - \mu_1)$$

$\left\{ \frac{1}{N}(x_1 + x_2 - x_N) \right\}$ → μ vector.

$(X - M) \rightarrow [x_1 - \mu \quad x_2 - \mu \; \cdots \; x_n - \mu]_{d \times n}$.

$(X-M)_{d \times n} \; (X-M)^T_{n \times d} \rightarrow d \times d \rightarrow (S)$ → covariance matrix.

$$\begin{bmatrix} c_{00} & c_{01} & c_{02} & \cdots \\ c_{10} & v_1 & \cdots & \\ \vdots & & v_2 & \\ \vdots & & & \ddots \end{bmatrix}$$

$Y = \mu_{1\,d \times 1}^T X_{d \times n} \rightarrow 1 \times n$.

$Y = [y_1 \; y_2 \; \cdots \; y_n]$

$y_i \in \mathbb{R}$.

$\left[\boxed{\max . u_1} \; - \; var(u_1^T X) \right]$ → $E\left(u_1^T X - E[u_1^T X]\right) - \left(u_1^T X - E[u_1^T X]^T\right)$.

(optimization Prob)

$\left\{ E[X] = \mu . \right.$

$\left. E[u_1^T X] = u^T \mu . \right\}$

↳ scalar times mean

$\rightarrow u_1^T E\left[(X-\mu)(X-\mu)^T\right] u_1$

$\boxed{Var(u_1^T X) = u_1^T S u_1 .}$ ← maximize this .

⇒ Maximize $u_1^T S u_1$ wrt u_1 → quadratic fn. | U → no maxima.

s.t. $u_1^T u_1 = 1$.

$$\left[\begin{array}{l} \max\limits_{u_1} \; u_1^T S u \\ \text{s.t. } u_1^T u = 1. \end{array} \right] \rightarrow \text{optimization prob. for PCA .}$$

$$J(u) = \underset{1\times d}{U_1^T} \underset{d\times d}{S} \underset{d\times 1}{U_1} - \lambda(U_1^T U_1 - 1)$$

$= 1\times 1$ → scalar.

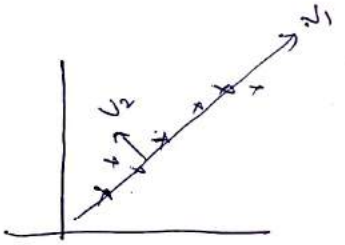
{ diff. scalar wrt. vector we get vector of same dimension. }

$J'(U) = 0 \Rightarrow 2SU_1 - 2\lambda U_1 = 0.$

$\boxed{SU_1 = \lambda U_1}$ → KEY RESULT.

↳ eigen vector.

$$\left\{ \begin{array}{l} \max. J(U) = U_1^T \lambda U_1 \\ \max. U_1 = \lambda. \end{array} \right\}.$$



1st principal component

$\underset{2d}{[\ ]}$ → cov.

$\lambda_1 V_1$, $\lambda_2 V_2$ } 2 eigen vectors.

$\lambda_1 V_1$ → direction vector.

V_2 → ⊥ orthogonal.

pick larger eigen vector i.e with larger λ. ($\lambda_1 > \lambda_2$ suppose)

↳ convert 2D to 1-D.

$$p = [p_1\ p_2\ \cdots\ p_d]$$

$$p_1^2 + p_2^2 + \cdots p_d^2 = 1 \rightarrow \text{unit vector.}$$

$$p^T p = [p_1\ p_2\ \cdots\ p_d] \begin{bmatrix} p_1 \\ p_2 \\ | \\ p_d \end{bmatrix}$$

$p^T p = 1 \rightarrow$ direction.

{ PCA — linear method. Kernel PCA exist. }.

SVD (Similar Value decomposition)

$X = U \Sigma V^T$

- U → eigen vector of XX^T
- V → eigen vectors X^TX.
- Σ → diagonal matrix $\begin{bmatrix} \lambda_1 & & \\ & \ddots & \\ & & \lambda_n \end{bmatrix}$

$\tilde{X} = (X - M)$ $d \times N$ $d \times N$.

$\tilde{X}^T \tilde{X}$ → covariance matrix.

→ when we do SVD on X

$X = [x_1 \text{ —— } x_d]$

$\tilde{X} = [x_1 - \mu \quad x_2 - u \quad x_n - u]_{d \times n}$ ⟶ SVD

Procedure for PCA.

d- dim. data

↳ SVD —— d eigen values.

$\{\lambda_1 \ \lambda_2 \text{ —— } \lambda_d\}$ —— d eigen vector.

$\{u_1 \ u_2 \quad u_d\}$.

if 1-D — multiply by λ_1, u_1.

2-D — " " $\lambda_1 u_1$ & $\lambda_2 u_2$.

for p-D data — multiply by 1st p vectors..

How many dimension should we pick?

$$r_p = \frac{\lambda_1 + \lambda_2 \text{ —— } \lambda_p}{\sum_{i=1}^{d} \lambda_i} = 0.99$$ ⟵ take p in such way.

$d = 1000$

$p = 40$.

SMAI (28/03)

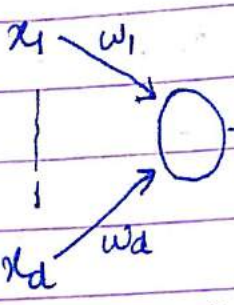
Neural Networks

→ Artificial Neural N/w.

→ parallel distributed info processor. — —.

{Slides}

Single Neuron Model



Net Input : $\eta = \sum_j w_j x_j$ → Output : $f(\eta)$

$y_j = \nearrow$

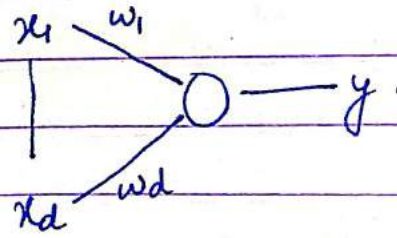
$f(\cdot)$ is called activation fn.

$\left\{ L = \sum (y_j - \hat{y})^2 \right\}$ → loss function.

$w_{t+1} \rightarrow w_t - \lambda \cdot \frac{\partial L}{\partial w}$.

Perceptron

$y = \sum_i w_i x_i = W^T X$.



$$X = \begin{bmatrix} x_1 \\ \vdots \\ x_d \end{bmatrix} \qquad W = \begin{bmatrix} w_1 \\ \vdots \\ w_d \end{bmatrix}$$

2 class classification

$\left\{ \begin{array}{l} W^T X \geq t \rightarrow \text{class 1} \\ W^T X < t \rightarrow \text{class 2.} \end{array} \right\}$

$y = w^T x + b$ (→ bias)

$\text{sign}(w^T x + b.) \rightarrow$ classification

Goal – to learn how to calculate weights.

$$x = \begin{bmatrix} 1 \\ x_1 \\ \vdots \\ x_d \end{bmatrix} \qquad w = \begin{bmatrix} w_0 \\ w_1 \\ \vdots \\ w_k \end{bmatrix}$$

$$\boxed{w^T x = w_0 + \sum_{i=1}^{d} w_i x_i}$$

→ Rosenblatt (1962)

~~$\Delta w(k)$~~ $\Delta w_k = w_{k+1} - w_k$.

$$\Delta w_k = 0 \begin{cases} \text{if } w_k^T x > 0 \ \& \ y(k) = 1, \\ \text{or } w_k^T x < 0 \ \& \ y(k) = 0 \end{cases}$$

$$\Delta w_k = \begin{cases} x(k) \text{ if } w_k^T x \leq 0 \ \& \ y(k) = 1 \\ -x(k) \text{ if } w_k^T x > 0 \ \& \ y(k) = 0. \end{cases}$$

Prediction → $\text{sign}(w^T x)$

made a mistake → $y(k) = 1$ ($\rightarrow y_i = 1$), $\boxed{w_k^T x \leq 0}$.

$$w_{k+1} = w_k + x_k x_i$$

$\boxed{w_k^T x}$ → $\boxed{w_{k+1}^T x}$

old prediction, made an error — new prediction

$$w_{k+1}^T x = (w_k + x_k)^T x$$

$$\boxed{w_{k+1}^T x = w_k^T x + (x_k)^2}.$$

→ prev. prediction + a small positive value.

Perceptron learning Algo — Slide.

→ In finite no. of iterations, perceptron algo finds correct line.

$w^T x_1 + b = 0$

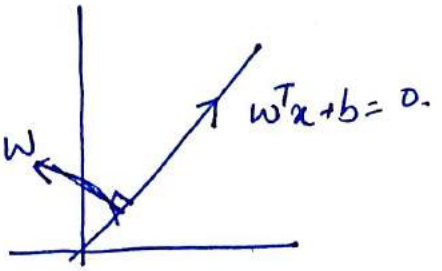
$w^T x_2 + b = 0$.

$b + w^T x_1 = w^T x_2 + b$.

$w^T(x_1 - x_2) = 0$.

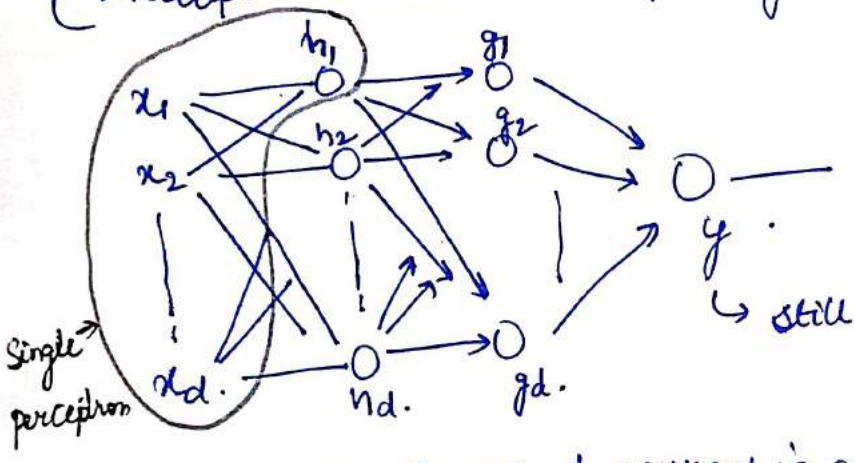
↓ ↓

Vector . Vector.



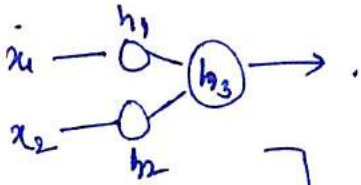
Multilayer Perceptron [— learning a linear fn. only]

(Multiple neurons in multiple layers)



↳ still weighted sum of $x_1 — x_d$.

⇒ Adding multiple layers of neurons is a chain.



$$h_1 = w_{11}x_1 + w_{12}x_2$$

$$h_2 = w_{21}x_1 + w_{22}x_2.$$

$$h_3 = w'_{11}h_1 + w'_{12}h_2.$$

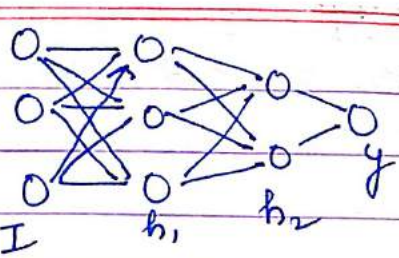
$$h_3 = w'_{11}(w_{11}x_1 + w_{12}x_2) + w'_{12}(w_{21}x_1 + w_{22}x_2)$$

$$h_3 = (w'_{11}w_{11} + w'_{12}w_{21})x_1 + (w'_{11}w_{12} + w'_{12}w_{22})x_2$$

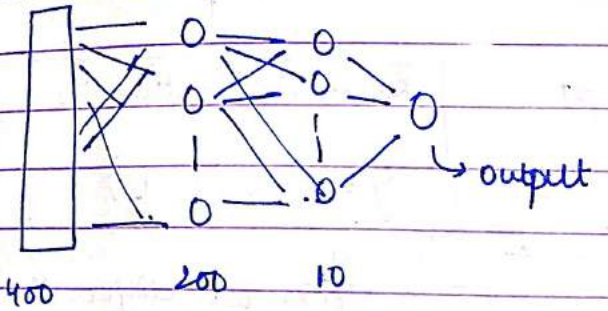
↳ h_3 — still linear sum of x_1, x_2

$h_i' = \phi(w_i x_i)$

y is now a non-linear fn. of x.



Non-linear classification OR Regression.



$(400 \times 200 + 200 \times 10 + 10 \times 1)$ → weights you have to train.

→ backpropegate your laws.

eg- house price prediction.

$$\left\{ \begin{aligned} \vec{h_1} &= \phi(w_1^T \vec{x_i}) \\ \vec{h_2} &= \phi(w_2^T \vec{h_1}) \\ y &= \phi(w_3^T \vec{h_2}) \end{aligned} \right\}$$

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

↑ minimize this loss by changing weights.

→ Use gradient descent.

$$w_{t+1} \rightarrow w_t + \lambda \cdot \frac{\partial L}{\partial w}$$ ← gradient descent

[Use chain Rule].

$$\frac{\partial z}{\partial w} = \frac{\partial z}{\partial y} \cdot \frac{\partial y}{\partial w}$$

$$\frac{\partial L}{\partial w_3} = \frac{\partial L}{\partial y} \cdot \frac{\partial y}{\partial w_3}$$

$$= \sum_{\text{over batch}} \left[2(y - \hat{y}) \cdot \frac{\partial \phi(w_3^T h_2)}{\partial w_3} \right]$$

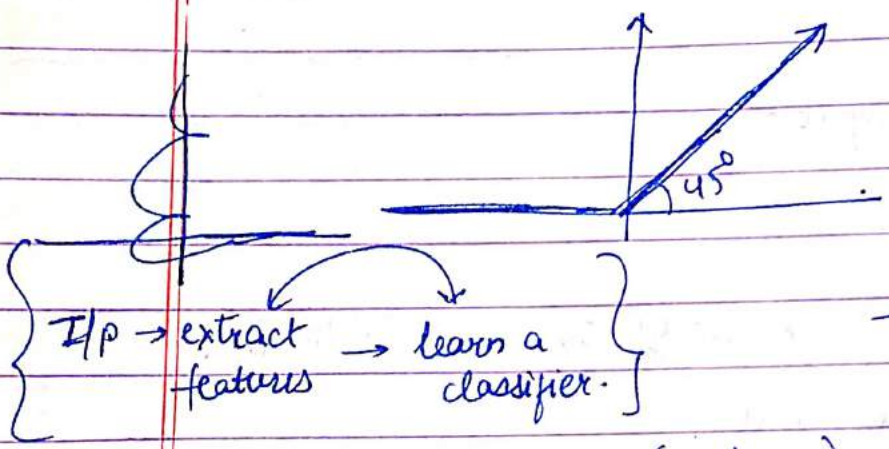
Backpropagation → gradient descent of with chain rule.

↓

Neural N/w

→ Gradient start at loss fn & backpropagates.

⇒ Slides.



Relu Activation fn.

$y = \max(0, x)$

↓

piecewise differential.

→ we can manage with these

SMAI (22/03).

MLP (Multilayered Perceptron)

$x \quad w_1^T x \quad \phi(w_1^T x) \quad w_2^T z \quad \phi(w_2^T z) \quad \frac{1}{2}(y - \hat{y})$.

(z over $\phi(w_1^T x)$, y over $\phi(w_2^T z)$)

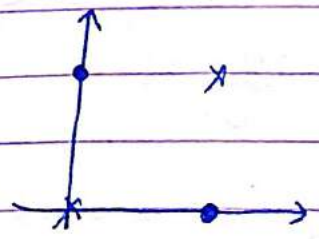
train w_1 & w_2.

$$\frac{\partial L}{\partial w_2} = \frac{\partial L}{\partial y} \cdot \frac{\partial y}{\partial w_2} = (y - \hat{y}) \cdot \phi'(w_2^T z) \cdot z.$$

$$\frac{\partial L}{\partial w_1} = \frac{\partial L}{\partial y} \cdot \frac{\partial y}{\partial z} \cdot \frac{\partial z}{\partial w_1} = (y - \hat{y}) \cdot \phi'(w^T z) \cdot w_2 \cdot \phi'(w_1^T x) \cdot x \cdot w_1$$

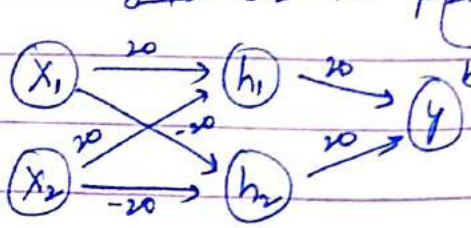
XOR

0 0	0
0 1	1
1 0	1
1 1	0



$\vec{x_i} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \quad \vec{h} = \begin{bmatrix} h_1 \\ h_2 \end{bmatrix}$

~~b=10~~ $b = -10$ $\phi(20x_1 + 20x_2 - 10)$



$b = 30$

$y = \phi(20h_1 + 20h_2 - 30)$

$h_1 = \phi(20x_1 + 20x_2 - 10)$

$h_2 = \phi(-20x_1 - 20x_2 + 30)$

h_1 h_2 y

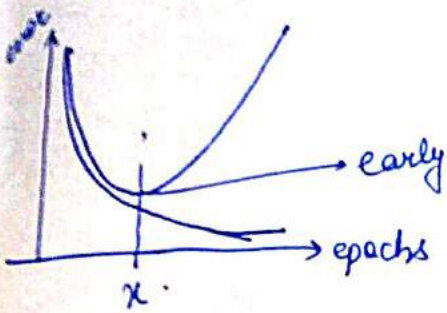
$\phi(20\times0 + 20\times0 - 10) \simeq 0$ $\quad \phi(-20\times0 + -20\times0 + 30) \simeq 1$ $\quad \simeq 0.$

$\phi(0 + 20 - 10) \simeq 1$ $\quad \phi(0 - 20 + 30) \simeq 1$ $\quad \simeq 1$

$y = 20h_1 + 20h_2 - 30.$

$\boxed{y = 100x_1 + 100x_2 + 100}$ → line.

XOR problem can't be solved by a line.



: when error on validation starts increasing.

Loss function.

① Regression → $\frac{1}{N} \times \frac{1}{2}\sum_i (y - \hat{y})^2$ — MSE. (MSE → avg.).

$\Sigma_i |y - \hat{y}|$ — MAE

KL divergence = $\sum -p_i \log \frac{q_i}{p_i}$

regularization terms can be added.

$$\boxed{\frac{1}{2}\sum_i (y - \hat{y})^2 + \lambda \sum_{(2)}^{N} w_i^2}$$

② Classification

Cross entropy — one hot encoding. class1 $\begin{bmatrix}0\\0\\0\\1\end{bmatrix}$ class2 $\begin{bmatrix}0\\1\\0\\0\end{bmatrix}$ - - -

size of output layer = no. of labels

$\boxed{-\log p_i}$

i = ground truth

$p = \begin{bmatrix} p_0 \\ p_1 \\ | \\ p_{n-1} \end{bmatrix}$ $\quad p_0 + p_1 - p_{n-1} = 1.$

$\log 0 = -\infty$; $\log 1 = 0$.

↓

max. loss .

Binary Cross Entropy $= y \log p_i + (1-y) \log(1-p_i)$

SoftMax

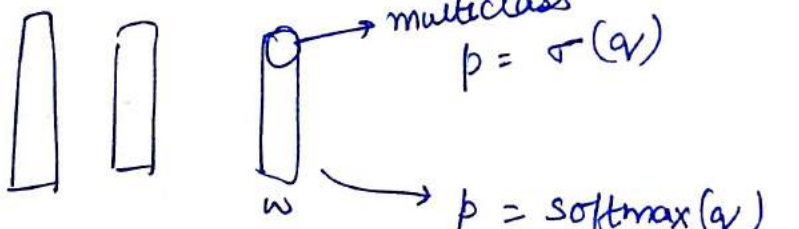
$$\begin{bmatrix} q_{10} \\ q_{11} \\ \vdots \\ q_{1n-1} \end{bmatrix} \xrightarrow{\text{Softmax}} \begin{bmatrix} p_0 \\ p_1 \\ \vdots \\ p_{n-1} \end{bmatrix}$$

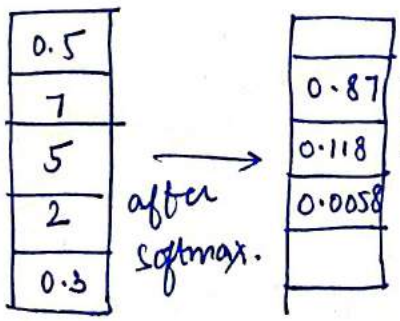
$$p_0 = \frac{e^{q_0}}{e^{q_0} + e^{q_1} - e^{q_{n-1}}}$$ → softmax .

↳ used in multiclass setting.
multiple binary classification
multiple class true.

→ Binary case doesn't need softmax, we can use ~~softmax~~ sigmoid.

→ Sigmoid & softmax used mainly as activation fn.





SMAI (25/03/21).

CNN

Image → 3-D array (H-w-L R-G-B).

Image — Flatten it — use SVM, etc.

↓

large size.

Other features — [Min Red. / Max Red / Mean Red / ⋮] → Same class dist. can be diff ~~Same~~; diff class same dist.

Higher level of Representations: Bag of Words

Images

↓ ↓ ↓

Take patches.

↓

draw histograms (of fixed dim.).

CNN ⟶ ImageNet

Dense Connections

$W^T x$ [ ]

W [ ] { } → densely connected.

x → [ ]

Convolutional Connection

PPT.

1	0	2

filter (1×3)

4	1	7	3	5

I/P (1×5)

↓

18	7	17

O/P (1×3)

→ padding may or may not be done.

eg:

1	2	0	1
0	4	7	2
3	1	4	0
1	2	0	2

4×4

2	1	2
7	0	0
4	1	3

3×3

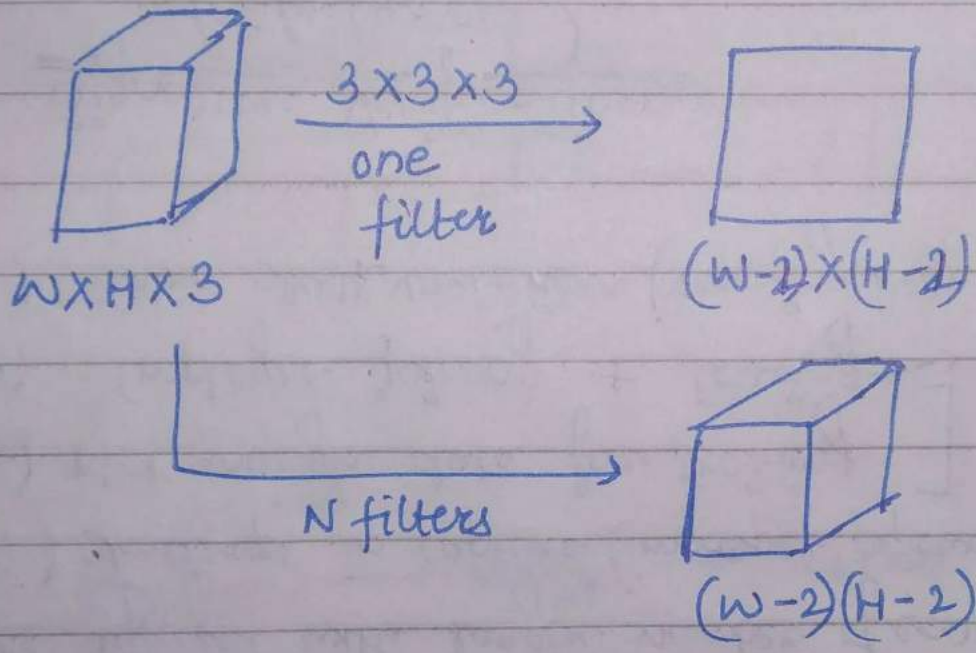
29	42

2×2

→ taking 3×3 filter & traversing it over image.

→ N filter of 3×3 $\xrightarrow{\text{no padding}}$ 2×2×N o/p.
↳ depth

Padding → 4×4×N o/p.



idea of convolution layer.

→ Stride & Padding.

→ Pooling — take max. in small window.

max-pooling; min-pooling; avg-pooling

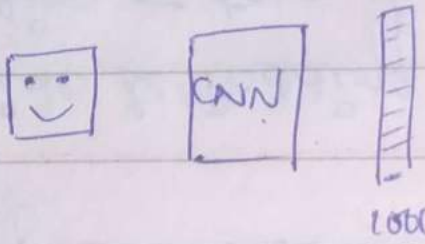
~~Sia~~ SMAI (01/04)

Siamese Networks + Metric learning

→ Face recognition based entry system for company.

[Suppose 1000 employees]

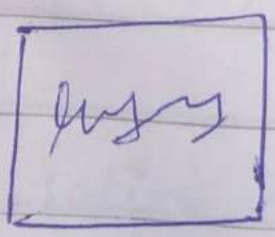
Sol 1 — 1000 class classification



→ what are the problems

(i) difficult with larger number of employees.
(ii) challenges in capture (improper capture) → agnostic.
(iii) Not enough data for training
(iv) Employee joining + leaving → train again every time.
(v) Intra class variation (removing beard, goggle etc.)

→ Signature Verification
(- in france)

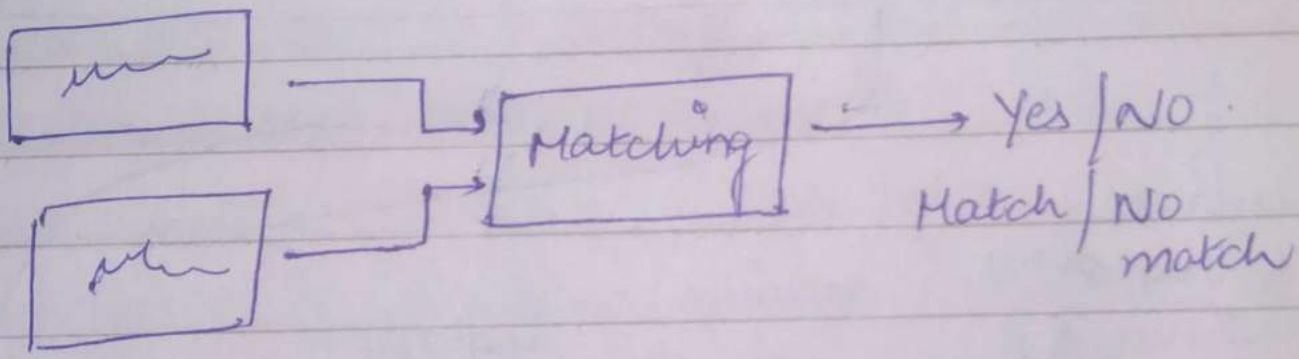


→ employee ID.

15 million classification probs.

↓
not doable

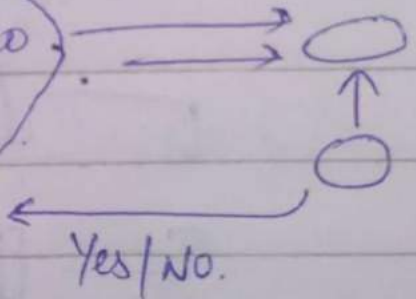
→ problem is ~~not~~ now verification problem.



→ Aadhar

during verification only. Thumb → verify → Yes / NO.

UIDAI → adhar no + image



During registration

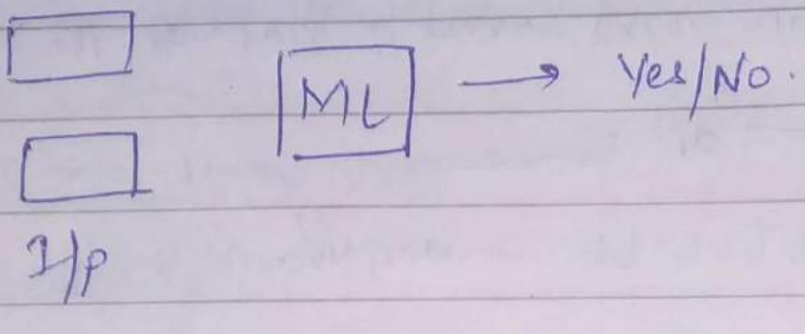
→ 2 thumbs, 8 fingers, IRIS — ? why all this data

Thumb recognition — 99.99999 % accurate

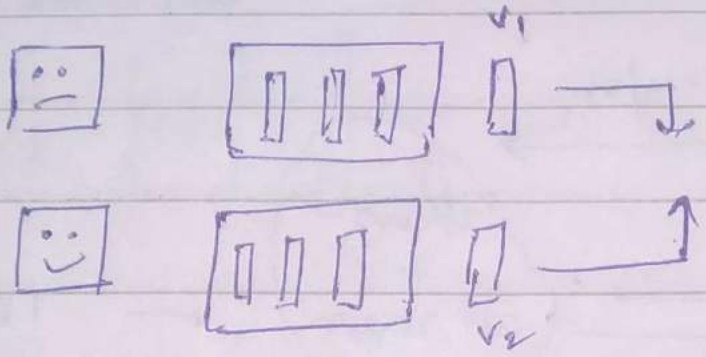
(1 thumb → 10^5 — 1 mistake) ⟹ 10^{10} — 1 mistake.

[take all to check if person exist already in db while registering decreasing prob. of error.]

Verification



Face verification



$y = 1 \rightarrow$ faces are of same identity $\quad d(v_1, v_2) \rightarrow$ small (dist)

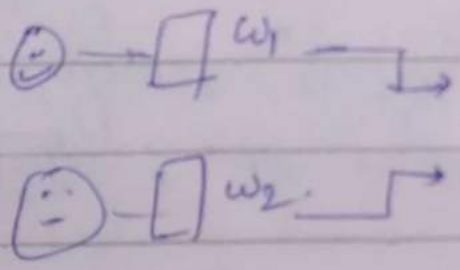
$y = 0 \rightarrow$ faces are of diff. identity $\quad d(v_1, v_2) \rightarrow$ large & more than a margin.

$$\text{loss fn.} = \min \left(y \cdot d^2 + (1-y) \max(0, m-d) \right)$$

→ contrastive loss fn.

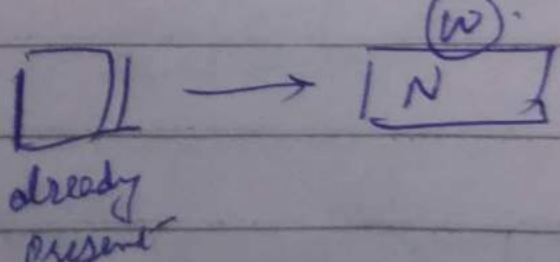
→ idea of constrastive ~~learning~~ loss – metric learning.
→ post processing had to be done earlier.
→ Now deep learning used – metric learning.
→ now we are transforming a face into a vector in that vector space (metric), faces of same identity are closer to each other (irrespective of intra class variation) – that is why metric learning.

→ Shared network.



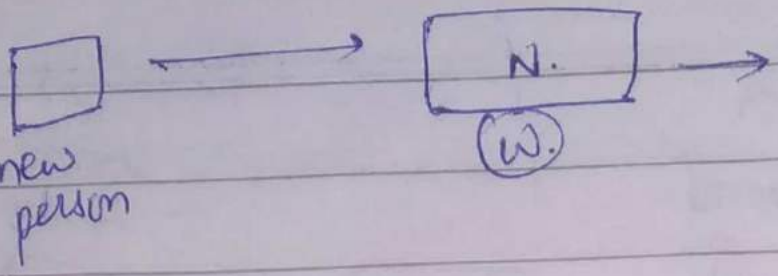
update w_1 & w_2 → $w = w_1 + w_2/2$ i.e mean.

Inference : 1000 employee problem (pre-trained network for face verification)



already present pic

precompute & store them.

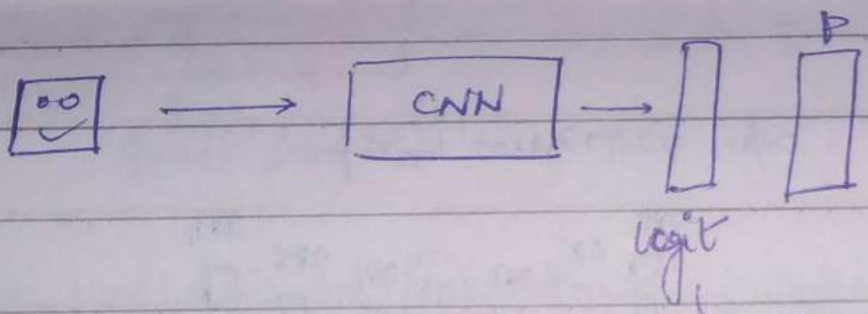


→ One forward pass + dist computation only.

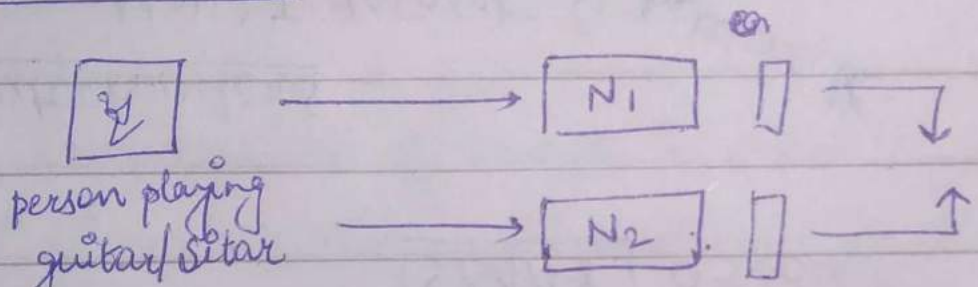
~~req~~ req.

Hard Negative Mining - use negatives which are harder to distinguish, training will be much stronger.

→ Celebrity dataset not work correctly on real world data due to biases
- clarity of photo, smiling, makeup on
These biases can drop accuracy on real world data.



Multi Model



N_1 & N_2 → diff. networks.

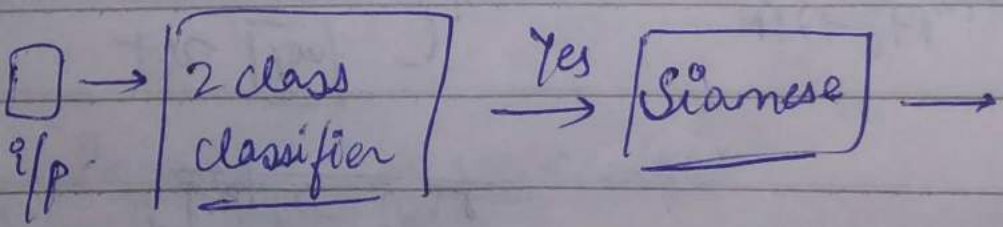
If image & caption match — Yes.

else — NO.

Q. What if instead of human, photo is placed?

~~instead of~~ Ans ⓐ → depth sensor reading (depth measurement)

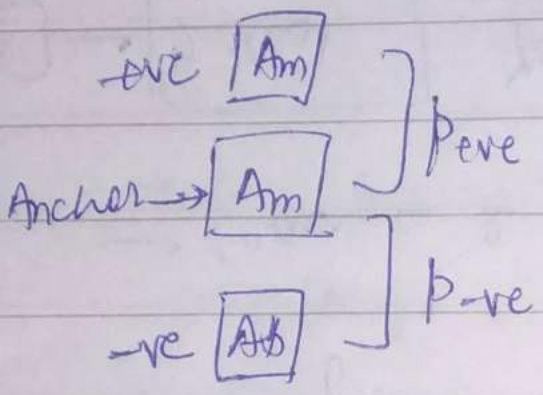
→ learn a 2 class classifier (real person V/S. photograph).



→ Person Re-identification

Matching patches - Shift (best algo)

Triplet loss



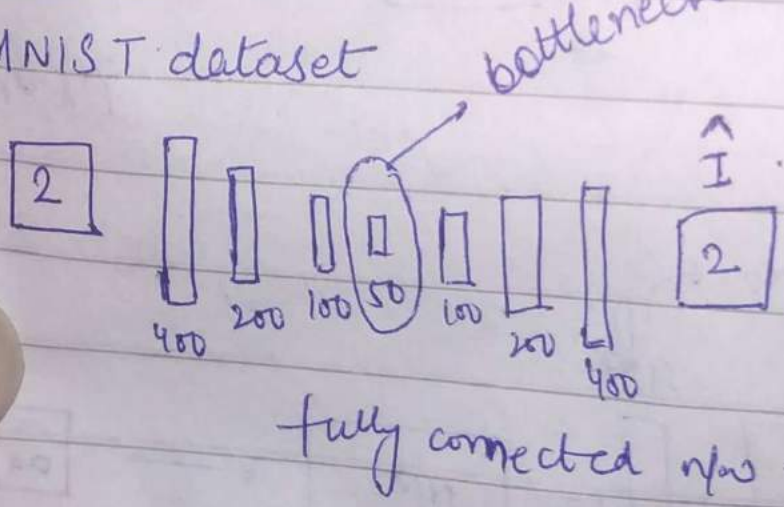
dist b/w non-matching pair should be atleast a margin more than matching pair.

(Contrastive & triplet gives almost similar accuracies)

SMAI (05/04)

Autoencoders

MNIST dataset



fully connected n/w.

$\text{Loss} = \Sigma(\hat{I} - I)$ $\quad \hat{I} \simeq I$

↓

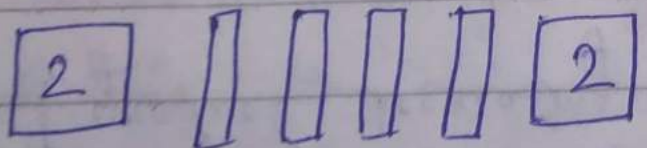
quite small.

⇒ if we are able to reconstruct image back, that means 50 dim. vector has all the info req. to construct the image.

→ Compression of image is done.
→ Smaller the bottleneck — greater the compression.
→ Instead of 400 dim. image — apply ML on 50 dim.
→ PCA — linear

Autoencoder — non-linear (activation fn. can be non-linear).

Network



→ just passing of info.
→ N/w should be able to do it.

← identity mapping →

Benefits of Autoencoders.

→ compression
→ dim. reduction — feature extraction (good feature extractor).

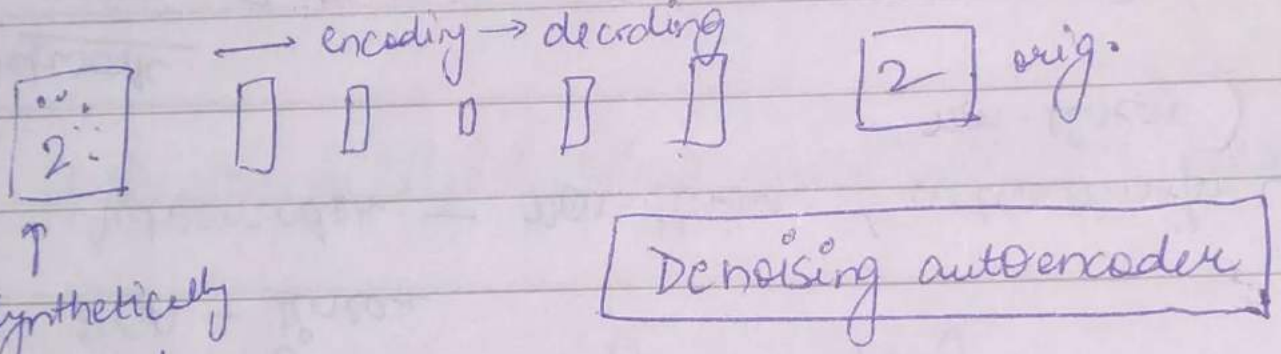
→ PCA not always good feature extractor.

→ Retrieval done using kNN mostly.
(if less dim. retrieval very fast)

→ 784 → 30 → 784

784 → 1000 → 500 → 250 → 30 → 250 → 500 → 1000 → 784

2nd case i.e. deep encoder is better.
[gradual decreasing and increasing better.]



→ Self Supervision : people create dataset
grey → color.
↓
(weighted sum of color gives grey)

[creating dataset easy]

→ Supervised / Self - Supervised. autoencoders.

GANs (Generative Modeling)

→ Given training data, generate new samples from same distribution.

GANs (Application of Game Theory)

→ generative adversial Networks

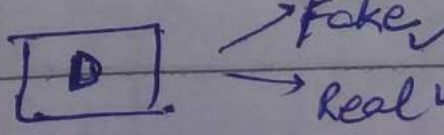
→ A game b/w two players —
discriminator D
Generator G.

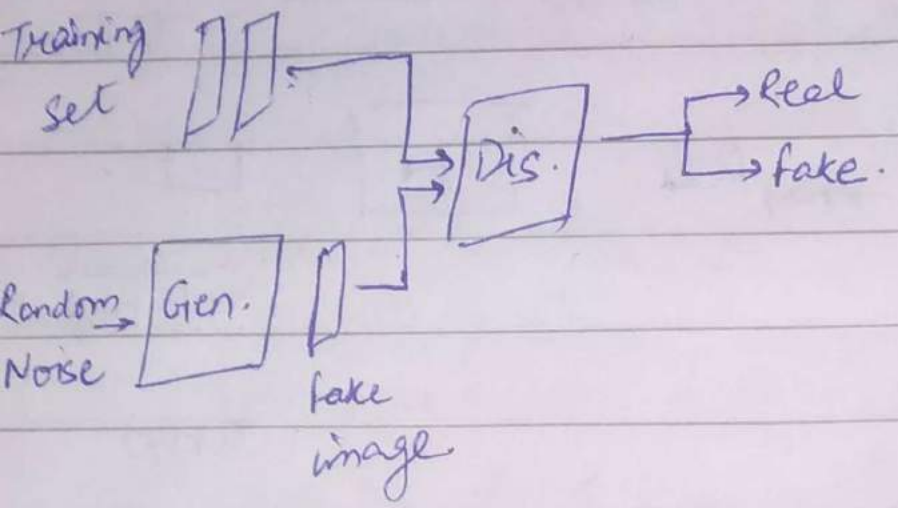
→ D tries to discriminate b/w.
a sample from data dist.
a sample from generator G.

→ G tries to "trick" D by generating samples that are hard for D to distinguish from data.

FNG.

→ Imp.





→ no explicit supervision in GAN's

freeze gen. → train discriminator → freeze dis.
→ train gen.

[continue this way]

Key Idea – Slides.

→ GAN difficult to train.

SMAI (12/04).

Word2Vec → 1) lower dimensional.

Mango = $\begin{bmatrix} 0.04 \\ \vdots \\ \end{bmatrix}_{300}$ 2) Semantics.

Hotel Motel

$\begin{bmatrix} \vdots \end{bmatrix} \cdot \begin{bmatrix} \vdots \end{bmatrix}$ Hotel X Motel ≈ 0 (house)

how to create embedding ?

→ curate dataset — fill in the blanks.

2 layer MLP (paper)

dictionary size = 10k.

Softmax → cross entropy.

(one hot embedding)

$W = 10,000 \times 300$

$U = 300 \times 10,000$

$z = w^T x$

$y = U^T z$.

z → hidden (layer) vector

↓

after training it becomes word2vec.

→ Skip gram model. (slightly beneficial if word occurs less).

→ CBOW Model.

king − man + woman → queen.

[ ] [ ] [ ] [ ]

$S_1 \simeq S_2 \simeq S_3$

India − delhi (S1) france − paris (S2)

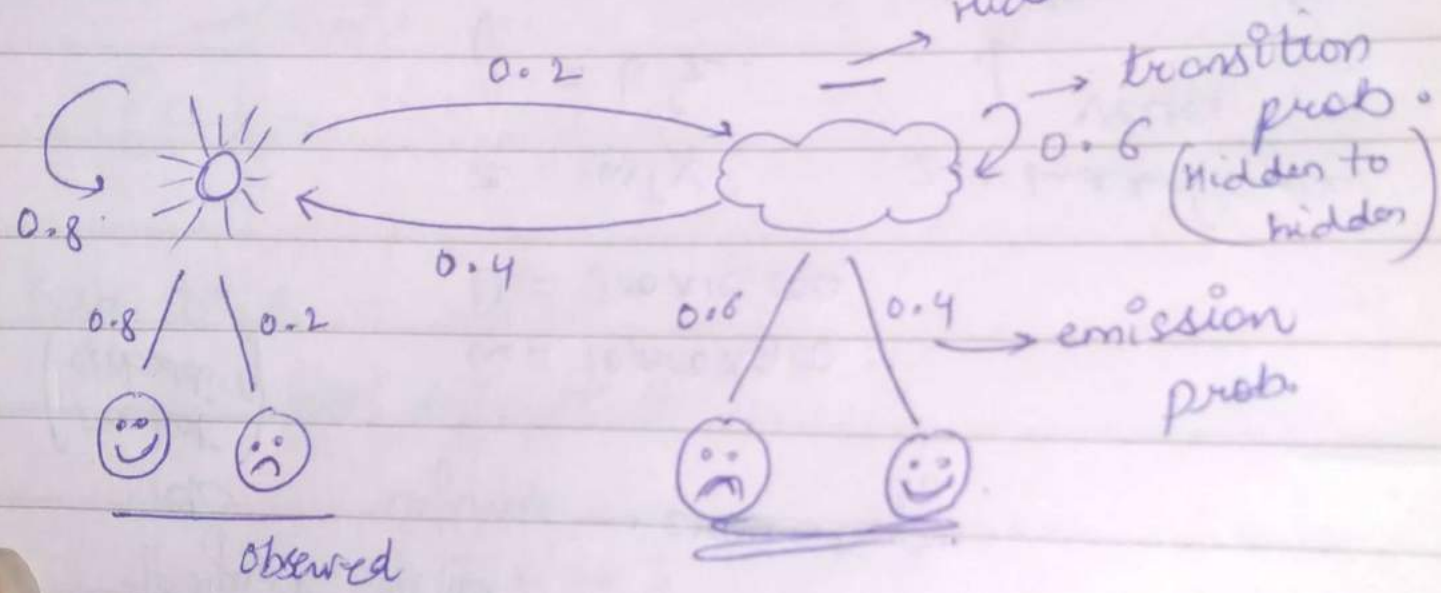
Nepal − kathm (S3) [ ] − [ ] = [ ] [ ] − [ ] = [ ].

[ ] − [ ] = [ ]

Glove, Word2Vec, Bert (↓ google), Elmo, USE } for word embedding.

{ techniques used are — supervised.
data comes easy / labels comes free — self supervised. }

Hidden Markov Model (HMM)



→ Given seq. of observations, I want to predict seq. of hidden variables.

→ HMM — predicting what you are not observing

How do I get these prob? — emission & transition

eg.

S S S S R R R S S S S R R S S

G H H H G G H G H H H G H H H

Trans. prob.

$S \to S = 8 = 0.8$ $\quad R \to R = 3 = 0.6$

$R \to S = 2 = 0.2.$ $\quad S \to R = 2 = 0.4.$

emm. prob.

$S \to H = P(H/S) = 0.8$

$S \to G = P(G/S) = 0.2.$

$P(H/R) = 0.4$

$P(G/R) = 0.6.$

Viterbi Algorithm

	M	T	W	Th	F	Sat.
	H	H	G	G	G	H.
2/3 ☀	$\frac{2}{3} \times 0.8 = 0.533$	$0.533 \times 0.8 \times 0.8$	0.0546	0.0087	0.0014	0.0017
☁ 1/3	$\frac{1}{3} \times 0.4 = 0.133$	$0.533 \times 0.2 \times 0.4$	0.041	0.0147	0.0053	0.0013

↑ prior prob

most likely config. to reach this box.

$= 0.341$

Sunny on Tuesday → max. $\begin{cases} 0.533 \times 0.8 \times 0.8, \\ 0.133 \times 0.4 \times 0.4 \end{cases}$

Rainy on Tuesday → max. $\begin{cases} 0.533 \times 0.2 \times 0.4 = 0.043. \\ 0.133 \times 0.6 \times 0.4. \end{cases}$

Sunny on w = max. $\begin{cases} 0.341 \times 0.8 \times 0.2, \\ 0.043 \times 0.2 \times 0.4 \end{cases}$

R on w = max. $\begin{cases} 0.341 \times 0.2 \times 0.6, \\ 0.043 \times 0.6 \times 0.6 \end{cases}$

O/P prediction — S S S R R S.

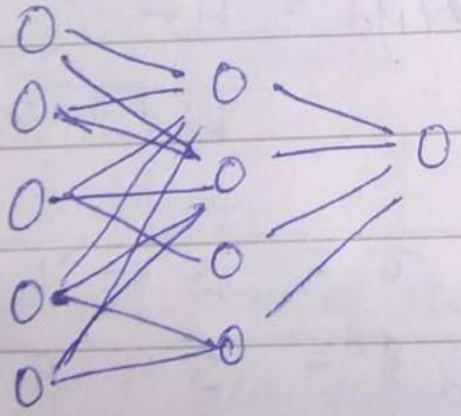
HMM → object tracking

→ Automatic cinematographic (camera selection)

SMAI (15/04).

Recurrent Neural Networks (RNN's)

MLP's



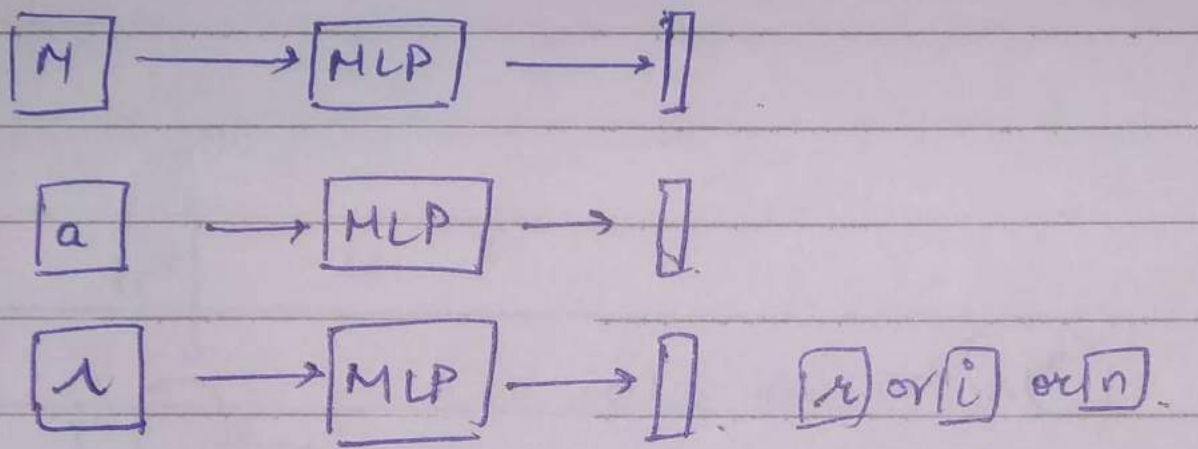
$y_1 \rightarrow y_2 \rightarrow y_3$

$$y_2 = \phi(w_1^T y_1)$$

$$y_3 = \phi(w_2^T y_2)$$

one sample at a time.

Machine



→ one sample at a time
→ fixed dimension
} prob. is MLP

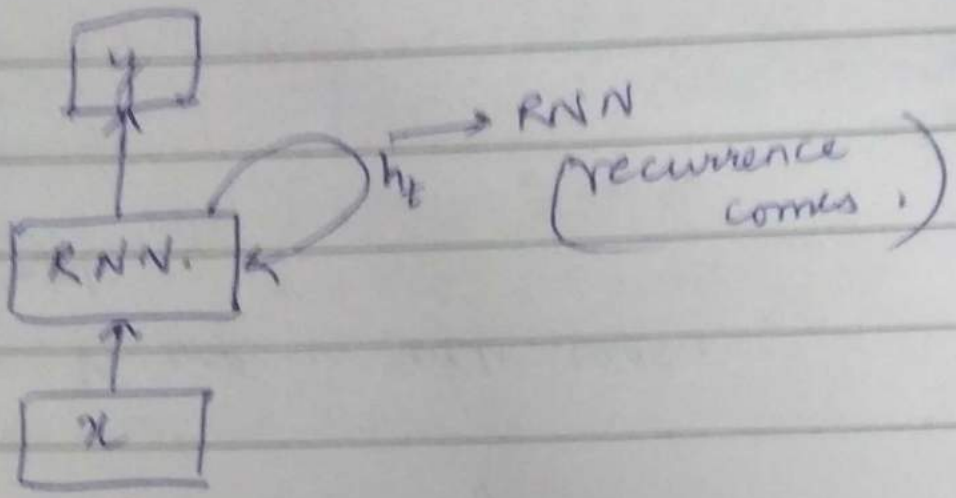
Sentiment Analysis

→ It was a good movie.
→ ——— ——— ——— ———
——— ——— but I did not like the movie....

→ The food was good, not bad at all.
→ The food was bad, not good at all.
} Bag of words Similar.

order is imp. in such case.

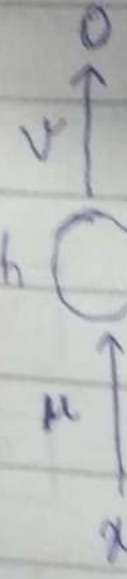
So, Need RNN



ML used RNNs.

→ seq. to seq. predictions.
(Lang. translations; — unimodal or multimodal Us, ASR)

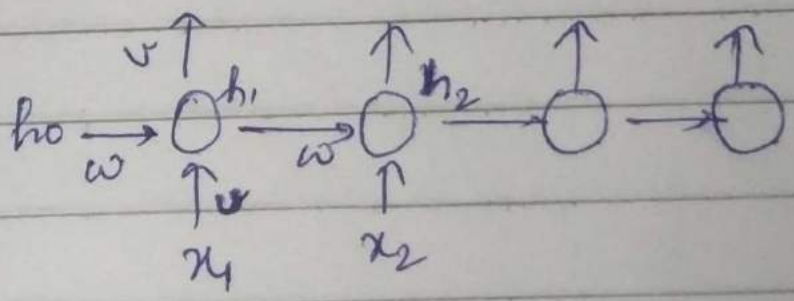
→ time series prediction
(→ stock price prediction etc.
prediction based on prev. data.).



activation fn

$$h_t = f(Ux_t + Wh_{t-1})$$

$$O_t = \text{softmax}(Vh_t)$$



$$h_1 = f(ux_1 + wh_0)$$

$$o_1 = \text{softmax}(vh_1)$$

$$h_2 = f(ux_2 + wh_1)$$

$$o_2 = \text{softmax}(vh_2)$$

{ current o/p will be based on current i/p & past observation. }

$$\frac{\partial L}{\partial w} = \sum_t \frac{\partial L_t}{\partial o_t} \frac{\partial o_t}{\partial w}$$

→ RNN stores short term info.

LSTM's

Stacked RNN.