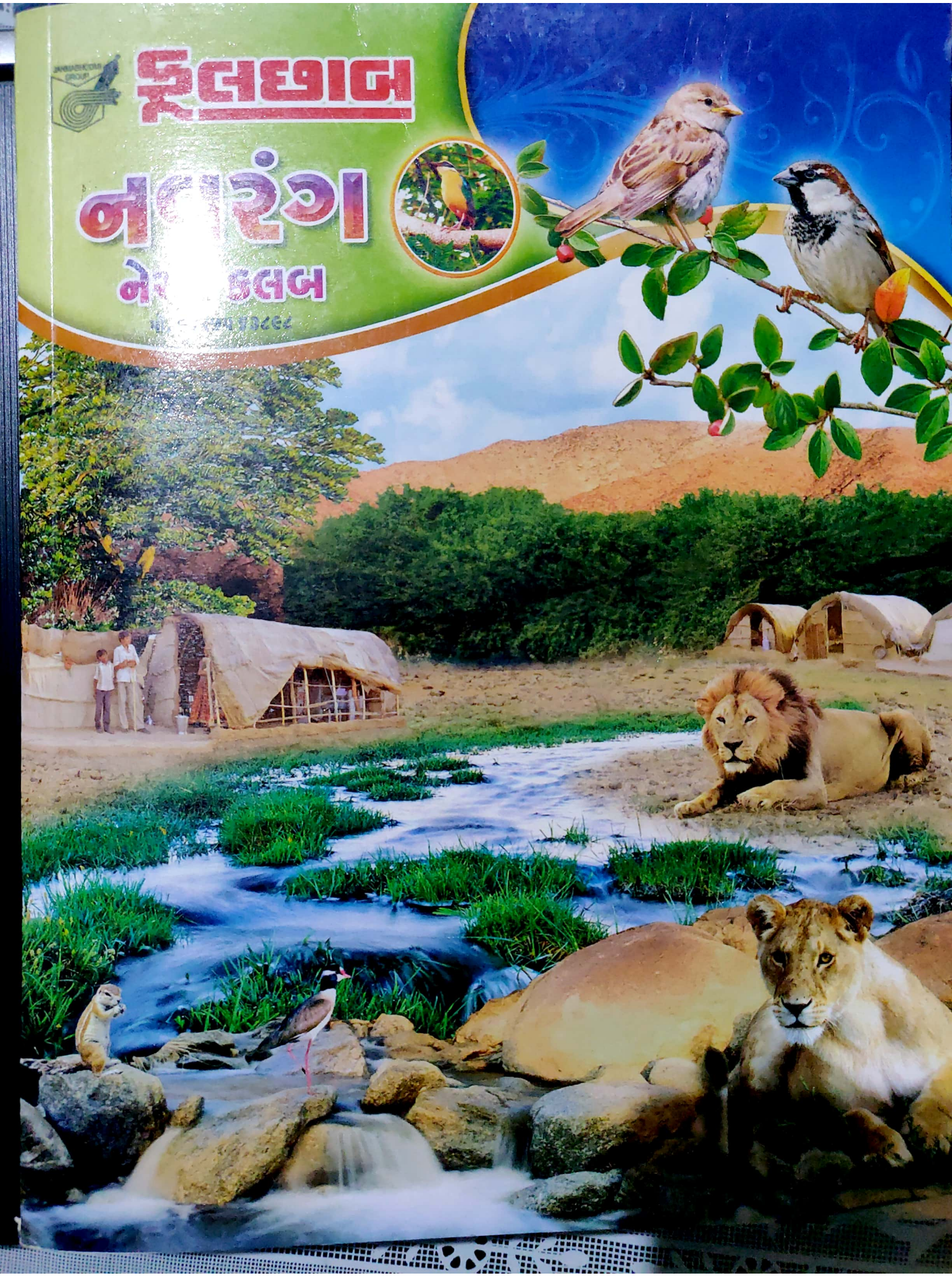


PAGE NO.
DATE :

-Kaushal P. Shah.
-Machine Learning
(2)

PAGE NO.
DATE 25/6/18

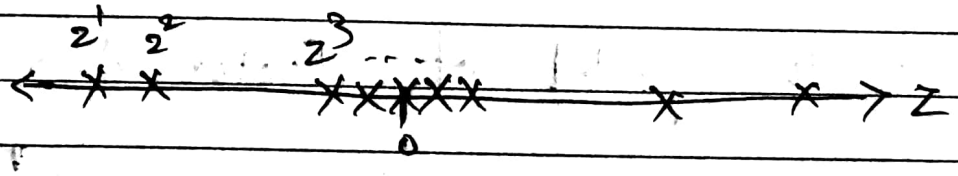
* Example of factor Analysis:-

→ let $z \in R^1$, $x \in R^2$, also

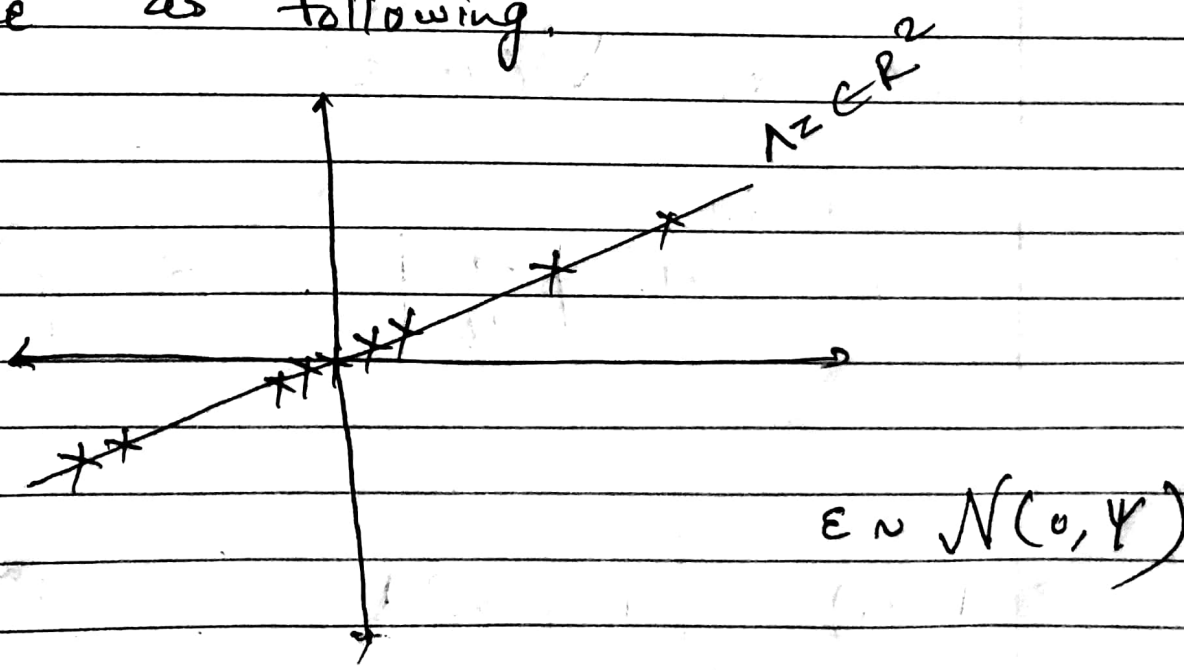
$$z \sim N(0, I), \quad \Lambda = \begin{bmatrix} 2 \\ 1 \end{bmatrix}, \quad \mu = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

$$\Psi = \begin{bmatrix} 1 & 0 \\ 0 & 2 \end{bmatrix}$$

→ Then z can be represented by

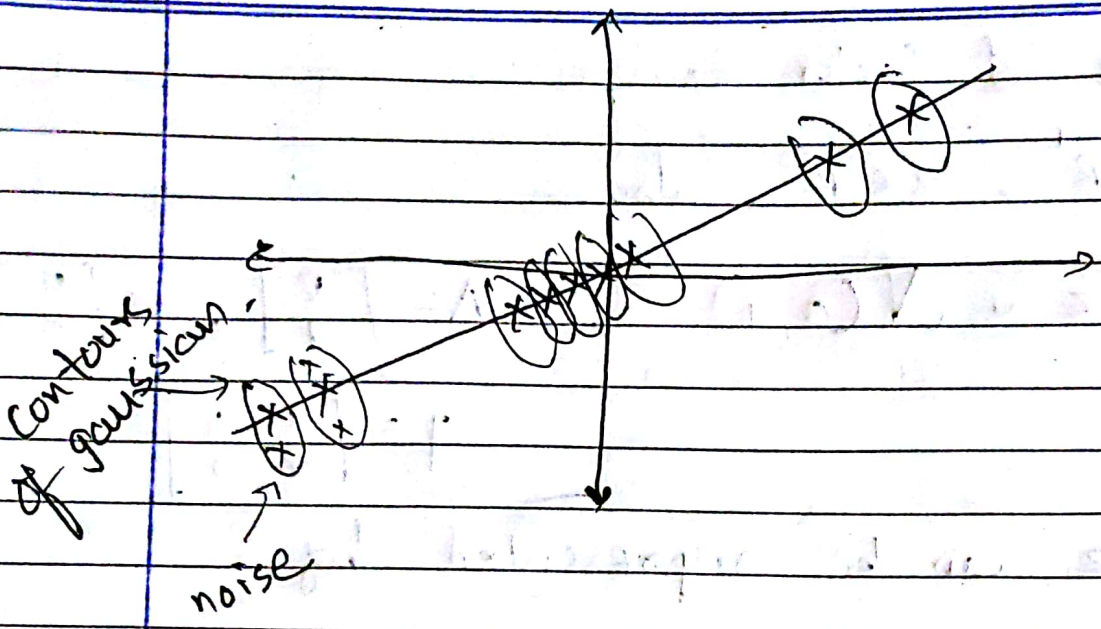


→ Now, Λz will be transformed into 2D space as following.



→ ε is noise term, which is also gaussian. so $x = \mu + \Lambda z + \varepsilon$ can be represented by :-

Note:- If $X \sim N(\mu, \sigma^2)$ and $Y \sim N(0, \sigma^2)$ then $X = \mu + Y$.



→ This is the model of factor Analysis.

→ Let $x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$, where $x_1 \in R^r$, $x_2 \in R^s$

$x \in R^{r+s}$

also, $x \sim N(\mu, \Sigma)$

where,

$$\mu = \begin{bmatrix} \mu_1 \\ \mu_2 \end{bmatrix}, \quad \Sigma = \begin{bmatrix} \Sigma_{11} & \Sigma_{12} \\ \Sigma_{21} & \Sigma_{22} \end{bmatrix} \begin{matrix} I_r \\ I_s \end{matrix}$$

(r and s mark the column blocks)

ex. $\Sigma_{12} \in R^{r \times s}$

→ $P(x_1) = \int_{x_2} P(x_1, x_2)\, dx_2$ ← Marginal distribution of Gaussian.

$x_1 \sim N(\mu_1, \Sigma_{11})$

conditional distribution of Gaussian

→ also,

$$P(X_1/X_2) = \frac{P(X_1, X_2)}{P(X_2)} \leftarrow \mathcal{N}(\mu_1, \Sigma) \leftarrow \mathcal{N}(\mu_2, \Sigma_{22})$$

$$X_1/X_2 \sim \mathcal{N}(\mu_{1/2}, \Sigma_{1/2})$$

→ where

$$\mu_{1/2} = \mu_1 + \Sigma_{12}\Sigma_{22}^{-1}(X_2 - \mu_2)$$

$$\Sigma_{1/2} = \Sigma_{11} - \Sigma_{12}\Sigma_{22}^{-1}\Sigma_{21}$$

→ Under factor analysis, some random variable x and z,

$$\begin{pmatrix} z \\ x \end{pmatrix} \sim \mathcal{N}(\mu_{zx}, \Sigma)$$

$$z \sim \mathcal{N}(0, I)$$

$$x = \mu + \Lambda z + \varepsilon \qquad x/z \sim \mathcal{N}(\mu + \Lambda z, \Psi)$$

$$\varepsilon \sim \mathcal{N}(0, \Psi)$$

→ $E[z] = 0$ $\quad \therefore E[x] = E[\mu + \underbrace{\Lambda z}_{0} + \underbrace{\varepsilon}_{0}]$

$$E[x] = \mu$$

$$\therefore \mu_{zx} = E\begin{bmatrix} z \\ x \end{bmatrix} = \begin{bmatrix} \vec{0} \\ \mu \end{bmatrix} \begin{matrix} I_d \\ I_n \end{matrix} \in R^{n+d}$$

PAGE NO.
DATE :

$$\Sigma = \begin{bmatrix} E(z-Ez)(z-Ez)^T & E(z-Ez)(x-Ex)^T \\ E(x-Ex)(z-Ez)^T & E(x-Ex)(x-Ex)^T \end{bmatrix}$$

$$\Sigma_{11} = \text{Cov}(z) = I$$

$$\Sigma_{21} = E(x-Ex)(z-Ez)^T$$

$$= E[(\mu + \Lambda z + \varepsilon - \mu)(z)^T]$$

$$(\because Ez = 0)$$

$$= E[\Lambda z z^T] - \underbrace{E[\varepsilon z]}_{0}$$

$$= \Lambda E[zz^T]$$

$$= \Lambda$$

$$\Sigma_{22} = E[(\mu + \Lambda z + \varepsilon - \mu)(\mu + \Lambda z + \varepsilon - \mu)^T]$$

$$= \Lambda\Lambda^T + \Psi$$

$$\therefore \begin{bmatrix} z \\ x \end{bmatrix} \sim N\left(\begin{bmatrix} \vec{0} \\ \mu \end{bmatrix}, \begin{bmatrix} I & \Lambda^T \\ \Lambda & \Lambda\Lambda^T + \Psi \end{bmatrix} \right)$$

$\mu_{2\times}$

PAGE NO.
DATE 7/7/18

→ So, from this what is distribution of x (P(x) = ?)

$$x \sim \mathcal{N}(\mu, \Lambda\Lambda^T + \Psi)$$

→ Likelihood of x is given by

$$\prod_{i=1}^{m} p(x^{(i)}; \Lambda, \mu, \Psi)$$

$$= \prod_{i=1}^{m} \frac{1}{(2\pi)^{n/2} |\Lambda\Lambda^T + \Psi|} \exp\left(\frac{-1}{2}(x-\mu)^T (\Lambda\Lambda^T + \Psi)^{-1} (x-\mu)\right)$$

→ Now, if we do max. likelihood estimation, we won't be find the solution [Take log and then take derivatives and set each derivative to zero].

→ So, we need to use EM algo. to estimate the parameters.

* E-step:- (EM for continuous $z^{(i)}$)

→ Find $Q_i(z^{(i)}) = P(z^{(i)} / x^{(i)}; \Theta)$

→ M-step:-

$$\underset{\Theta}{\text{argmax}} \sum_i \int_{z^{(i)}} Q_i(z^{(i)}) \log \frac{P(x^{(i)}, z^{(i)}; \Theta)}{Q_i(z^{(i)})} dz^{(i)}$$

Expectation = Mean.

PAGE NO.
DATE :

→ Here $z^{(i)}$ is Gaussian Random variable, so, there is Integration instead of Summation.

EM for Factor Analysis

→ E-step:

$$z^{(i)} | x^{(i)} \sim \mathcal{N}(\mu_{z^{(i)}|x^{(i)}}, \Sigma_{z^{(i)}|x^{(i)}})$$

where,

$$\mu_{z^{(i)}|x^{(i)}} = \vec{0} - \Lambda^T (\Lambda\Lambda^T + \Psi)^{-1} (x^{(i)} - \mu)$$

$$\Sigma_{z^{(i)}|x^{(i)}} = I - \Lambda^T (\Lambda\Lambda^T + \Psi)^{-1} \Lambda$$

→ M-step:

$$\int_{z^{(i)}} Q_i(z^{(i)}) \, z^{(i)} \, dz^{(i)}$$

→ Now, Don't write $Q_i(z^{(i)})$ in terms of Gaussian, and try to integrate, just observe that above eqn is expectation of $z^{(i)}$

$$E_{z^{(i)} \sim Q_i}[z^{(i)}] = \mu_{z^{(i)}|x^{(i)}}$$

PAGE NO.
DATE

→ So, we've μ, Λ, Ψ

$$E_{z^{(i)} \sim Q_i}\left[\log \frac{P(x^{(i)}, z^{(i)}; \theta)}{Q_i(z^{(i)})}\right]$$

$$= E_{z^{(i)} \sim Q_i}\left[\log P(x^{(i)} | z^{(i)}; \theta)\right] + E_{z^{(i)} \sim Q_i}\left[\log \frac{P(z^{(i)})}{Q_i(z^{(i)})}\right]$$

only this term depends on parameters, so only maximize it.

→ So, In M-step, maximize

$$\max \sum_{i=1}^{m} E_{z^{(i)} \sim Q_i}\left[\log P(x^{(i)} | z^{(i)}; \Lambda, \Psi, \mu)\right]$$

$$x^{(i)} | z^{(i)} \sim \mathcal{N}(\mu + \Lambda z, \Psi)$$

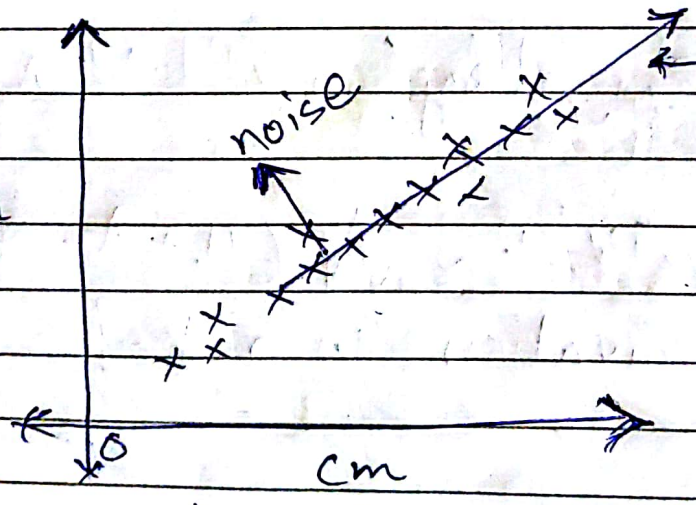
[See Derivation in notes]

PAGE NO.
DATE: 8/7/18

* Principle component Analysis- (PCA)

→ Given $\{x^{(1)}, x^{(2)} \ldots, x^{(m)}\}$ $x^{(i)} \in R^n$, in this algo. we reduce the dimension of each dataset $x^{(i)}$ from n to k, where $(k < n)$.

→ Why this is required?
For ex- if we've collected the data of height of persons in both 'cm' and 'inch'. So this is redundent dimension.



→ The graph will be a straight line (not exact straight line due to rounding of cm to inch or vice-versa).

→ preprocessing the data before running the algo:—

1.) set $\mu = \frac{1}{m}\sum_{i=1}^{m} x^{(i)}$ } zero out the mean

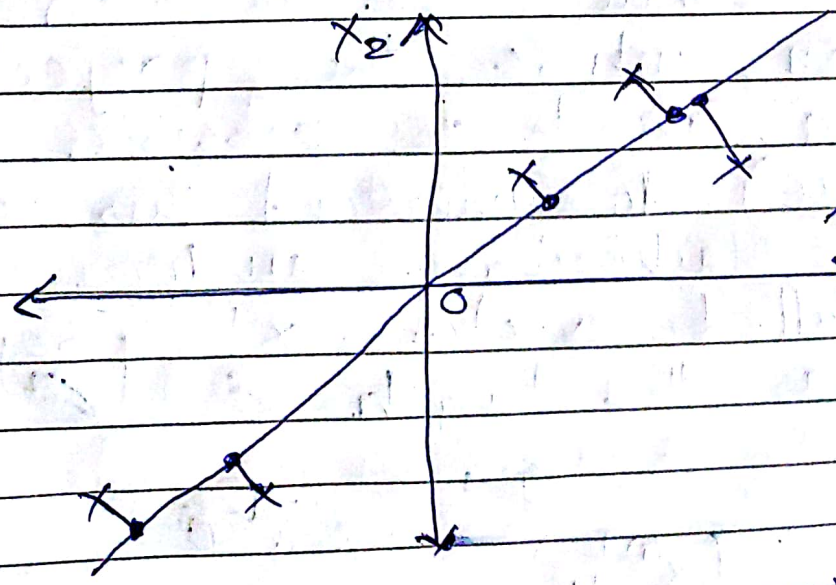
2) Replace $x^{(i)}$ with $x^{(i)} - \mu$ }

3) set $\sigma_j^2 = \frac{1}{m}\sum_{i=1}^{m} (x_j^{(i)})^2$ } Normalize to unit variance.

4) Replace $x_j^{(i)}$ with $\frac{x_j^{(i)}}{\sigma_j}$ }

here j represent feature. So, σ_j^2 is varience of feature j.

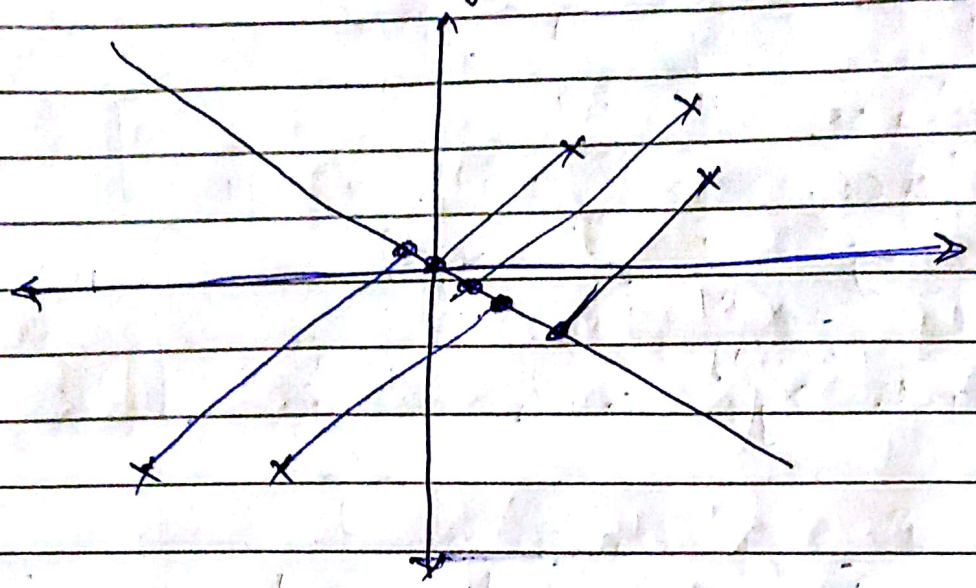
→ Now, how to find the mean axis or principle axis, along which the data varies?



→ direction of axis along which my data varies.

→ Note that the projection (dots) have very large variance

→ If we choose some other direction, like following,



→ Then the projections (dots) have very low variance.

→ So to formalize this idea of how to find the axis, we can say that the axis should be in such a direction, where the projection (dots) vary as much as possible (high variance). To understand this, read the saved stackexchange answer of PCA.

→ If $\|u\| = 1$, vector $x^{(i)}$ projected on u has the length = $(x^{(i)T} u)$

$(x^{(i)T}u)$ is dot product and hence scaler. see the stack-exchange question for confusion

→ So, our problem will be

choose u such that:

$$\max_{u:\|u\|=1} \frac{1}{m} \sum_{i=1}^{m} (x^{(i)T}u)^2$$

$$= \max_{\|u\|=1} \frac{1}{m} \sum_{i=1}^{m} (u^T x^{(i)})(x^{(i)T}u)$$

$$= \max_{\|u\|=1} u^T \left[\frac{1}{m} \sum_{i=1}^{m} x^{(i)} x^{(i)T} \right] u.$$

Now $\frac{1}{m} \sum_{i=1}^{m} x^{(i)} x^{(i)T}$ is co-variance matrix. Σ.

so, $\Sigma = \frac{1}{m} \sum_{i=1}^{m} x^{(i)} x^{(i)T}$

now, $\max_{\|u\|=1} u^T \Sigma u.$

→ We can apply lagrage's multipliers to maximize constrained optimization problem.

so, constraint is $\|u\| = 1 = u^T u$

PAGE NO.
DATE :

→ Lagrangian will be (λ → lagrange multiplier)

$$L(u,\lambda) = u^T \Sigma u - \lambda(u^T u - 1)$$

→ differentiating L,

$$\Sigma u - \lambda u = 0$$

→ which is eigen-vector equation.
Substitute this in objective function,

$$u^T \Sigma u - \lambda(u^T u - 1)$$
$$= u^T \Sigma u - \lambda u^T u + \lambda$$
$$= u^T \Sigma u \quad (\because u^T u = 1)$$
$$= \lambda u^T u$$
$$= \lambda$$

→ So, to maximize f, λ must be the largest eigen value.

→ For any matrix A, and vector u, if

$$Au = \lambda u \text{, then}$$

λ is eigen value and
u is eigen vector of A.

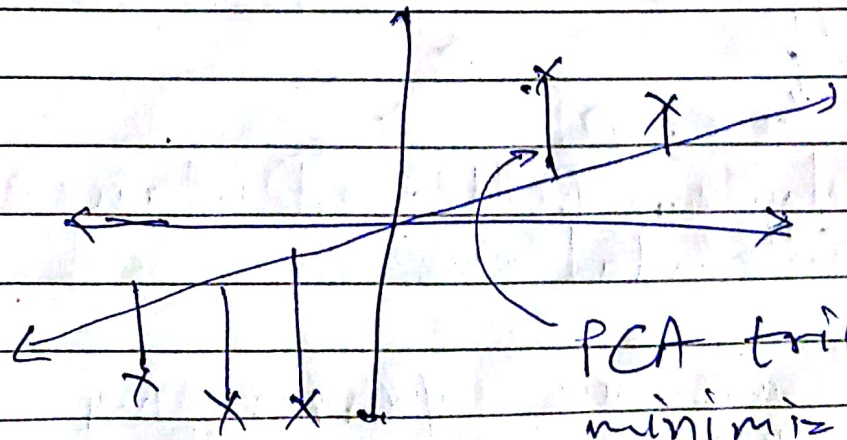
→ More generally, if we want k-dimensional subspace, then choose $u_1, u_2 \dots u_k$ to be k top eigenvectors of Σ. That means eigen vectors corresponding to top k eigen values.

→ So, If we've $x^{(i)} \in R^n$, now new representation of data in $\{u_1, u_2 \dots u_k\}$ basis will be

$$y^{(i)} = (u_1^T x^{(i)}, u_2^T x^{(i)} \dots u_k^T x^{(i)}),$$

$$y^{(i)} \in R^k, \quad k < n.$$

→ Another way to think of PCA is that it tries to minimize the distance between data points and the axis



→ Which is also called reconstruction error

PAGE NO.
DATE :

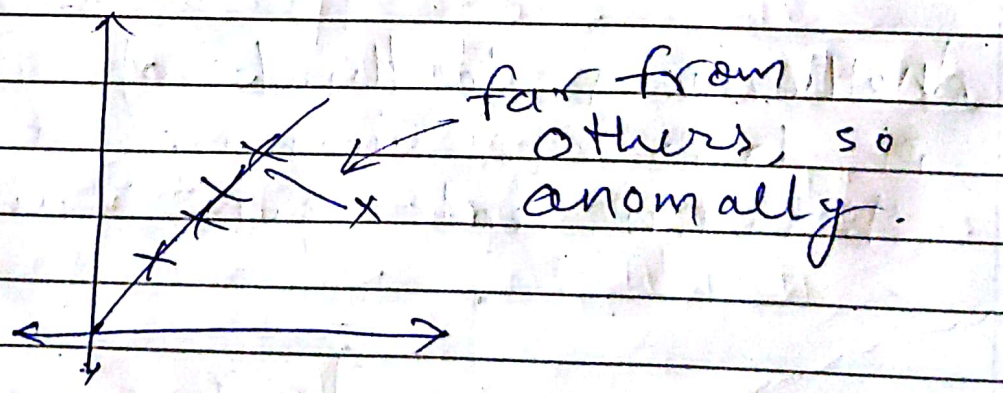
* Applications of PCA :-

1) Visulization :- Very often we've very high dimensional data sets. It's not possible to plot it. So PCA is used to reduce the dimension & plot & visulize the datasets.

2) Compression.

3) Learning algo. runs much faster. Also, with PCA, we've less features, so less prone to over-fitting.

4) Anomaly Detection.



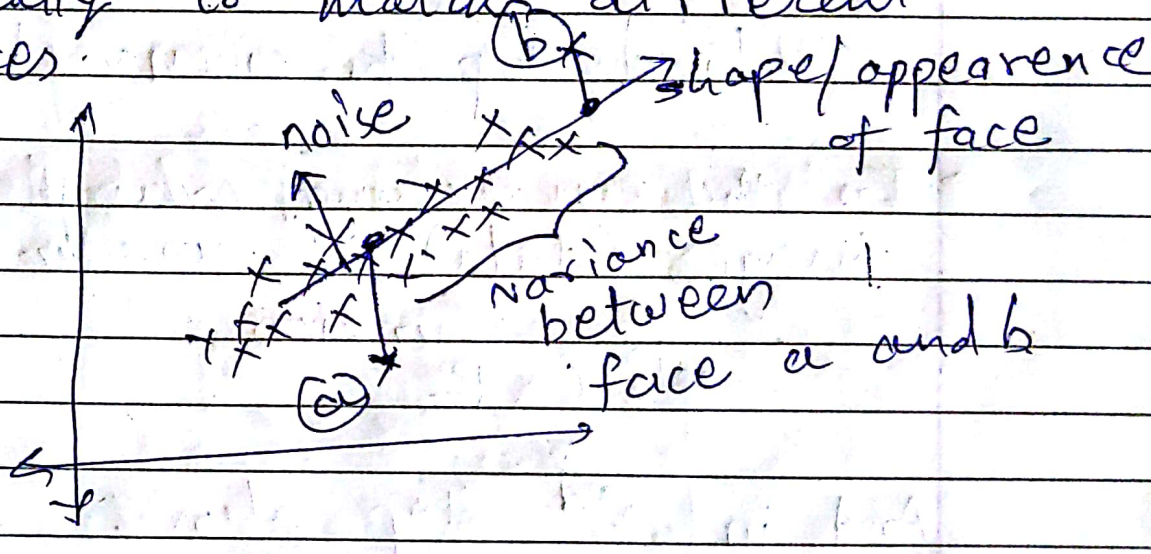
5) Matching / Better Distance Calculations.

→ for ex. we are performing face recognition and we've 100x100 images of faces, then each data point $x^{(i)}$ will be of

$100 \times 100 = 10,000$ dimensional vector

$x^{(i)} \in R^{10000}$

→ If we reduce it to ~~100~~ approx. 50 dimensions, then we can use it efficiently to match different faces.

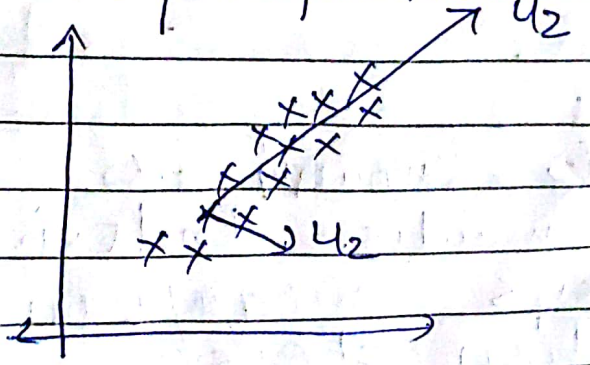


$x^{(i)} \in R^{10,000}$

$u^{(i)} \in R^{10000}$ eigen faces.

* Lecture-15 15/7/18.

Summary of PCA



PAGE NO.
DATE :

Algorithm:

1) Normalize the data to zero mean, unit variance.

2) find covariance Matrix as

$$\Sigma = \frac{1}{m} \sum_{i=1}^{m} x^{(i)} x^{(i)T}$$

3) Find top k eigenvectors of Σ.

→ In the matching example, if our images (data) is 100×100, then

$$x^{(i)} \in R^{100 \times 100}$$

And in 2nd step above, our Co-variance matrix will be

$$\Sigma \in R^{10000 \times 10000}$$, which is of very high dimension.

→ Another use of PCA is to measure similarity between two text documents.

→ This algo. is known as LSI (Latent Semantic Indexing), which usually skips mean/variance normalization steps.

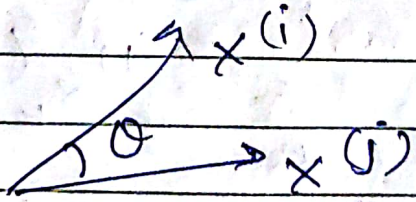
PAGE NO.
DATE

→ To measure the similarity between two documents $x^{(i)}$ and $x^{(j)}$, which are represented as a very high dimensional feature vector like following:

1	a
0	aardwark
0	aardwold
1	learn
⋮	⋮

where $x \in R^{50,000}$ (size of dictionary).

→ then to measure the similarity, $sim(x^{(i)}, x^{(j)})$, we can view these two vectors as following



→ One measure of similarity b/w two documents is the angle b/w the above vectors.

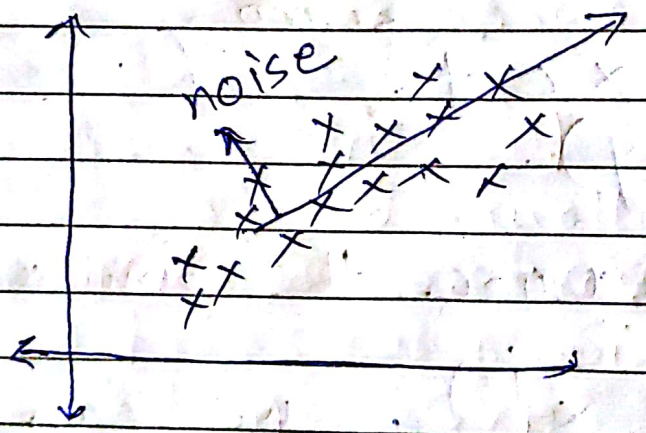
→ If θ is small → Documents are similar. & vice versa.

→ So, similarity is $\cos\theta$. [Angle increases → Similarity decreases and vice-versa]

$$\cos\theta = \frac{x^{(i)T} x^{(j)}}{\|x^{(i)}\| \|x^{(j)}\|}$$

[from Definition of Dot Product]

→ Intuition behind LSI.



→ Numarator of $sim(x^{(i)}, x^{(j)})$, is

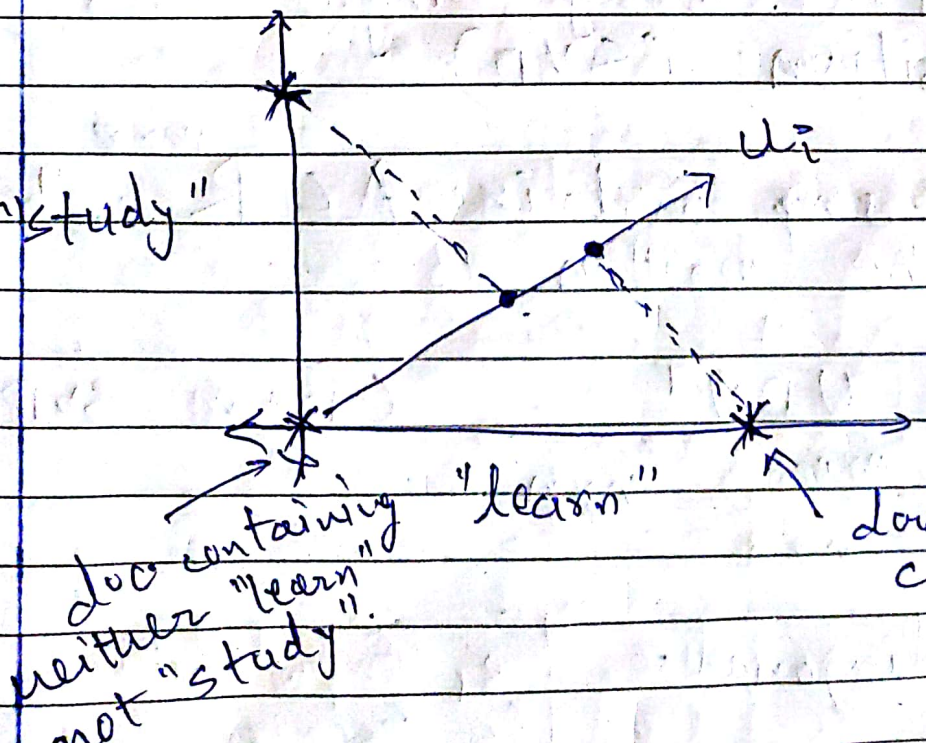
$$x^{(i)T} x^{(j)} = \sum_k x_k^{(i)} x_k^{(j)}$$

which will be zero if documents don't have any word in common. [Dot product], so it can be written as

$x^{(i)T} x^{(j)} = \sum_k 1\{$ Document i and j both contain word $k\}$.

→ and in above formula, $x_k^{(i)}$ means whether document i contains word k and same for $x_k^{(j)}$.

→ E.g. if $x^{(i)}$ = "study" and $x^{(j)}$ = "learn", the both documents are considered completly dissimilar documents.



→ So, from LSI we can say that there is +ve similarity between these two documents.

⇒ But in such cases, Σ will be very very high dimensional matrix. for ex.

$x^{(i)} \in R^{50,000}$, then co-variance matrix will be of dimension

$\Sigma \in R^{50,000 \times 50,000}$

→ So, applying PCA will be difficult for such cases. So another way to apply PCA is to decompose the data matrix using Single value decomposition (SVD)

→ ex. let any matrix $A \in R^{m \times n}$, then it can be written as

$$A = U D V^T \quad \text{(from SVD)}$$

$m \times n \quad m \times n \quad n \times n \quad n \times n$

→ D is diagonal = $\begin{bmatrix} \sigma_1 & 0 & \cdots & 0 \\ \vdots & \sigma_2 & & \vdots \\ 0 & \cdots & \ddots & \sigma_n \end{bmatrix}$

σ_i in D matrix is called singular values of matrix A.

Eg $\underset{m\times n}{[A]} = \underset{m\times n}{[U]}\,\underset{n\times n}{[D]}\,\underset{n\times n}{[V^T]}$

U's columns :- Eigenvectors of AA^T
V's columns :- Eigenvectors of A^TA

→ SVD can be computed by "svd" in matlab or octave.

→ We written Σ as

$$\Sigma = \sum_{i=1}^{n} x^{(i)} x^{(i)T}$$ (Ignore $\frac{1}{n-1}$ for now)

→ Now, let's writer our data points $x^{(i)}$ as rows in design matrix X.

$$X = \begin{bmatrix} — x^{(1)} — \\ — x^{(2)} — \\ \vdots \\ — x^{(m)} — \end{bmatrix}$$

PAGE NO.
DATE :

then co-variance matrix can be written as

$$\Sigma = X^T X \qquad \left(\frac{1}{n-1} \text{ ignored for now}\right)$$

$$= \begin{bmatrix} | & | & | & \\ x^{(1)} & x^{(2)} & \cdots & \\ | & | & | & \end{bmatrix} \begin{bmatrix} — x^{(1)} — \\ — x^{(2)} — \\ \vdots \\ \vdots \end{bmatrix}$$

→ Now, to get top k eigen vectors,

$$X = UDV^T$$

→ Top k columns of U are the top k eigen vectors of $X^T X = \Sigma$.
So this process is same as PCA.

→ So, if we've m training example, and 50,000 features (dictionary size), then dimension of X would be

$$X \in \mathbb{R}^{m \times 50,000}.$$

→ So, this require less computation power compared to $50000 \times 50,000$ matrix Σ.

PAGE NO.
DATE

→ So, if we've very high dimensional data then we should apply SVD to perform PCA.

* PCA, factor Analysis, and differences and when to use each one?

	Model P(x)	Not Probabilistic.
"subspace"	Factor Analysis	PCA
"Group clusters"	Mixture of Gaussian	K-means

similarity

differences.

→ when we want to reduce dimensions, use PCA.

→ when we want annomally defection, use factor analysis.

→ FA is for modeling the data / model P(x).

→ Both assumes, that the data lies in subspace

→ when Data lies in different Groups/classes, and we want density estimation, use Mixture of Gaussian and same for K-means

PAGE NO.
DATE : 21/7/18

* Independent Component Analysis (ICA):

→ In PCA, we're finding the main (principle) axes of variation of data, where as in ICA, we are finding independent components of variation of data.

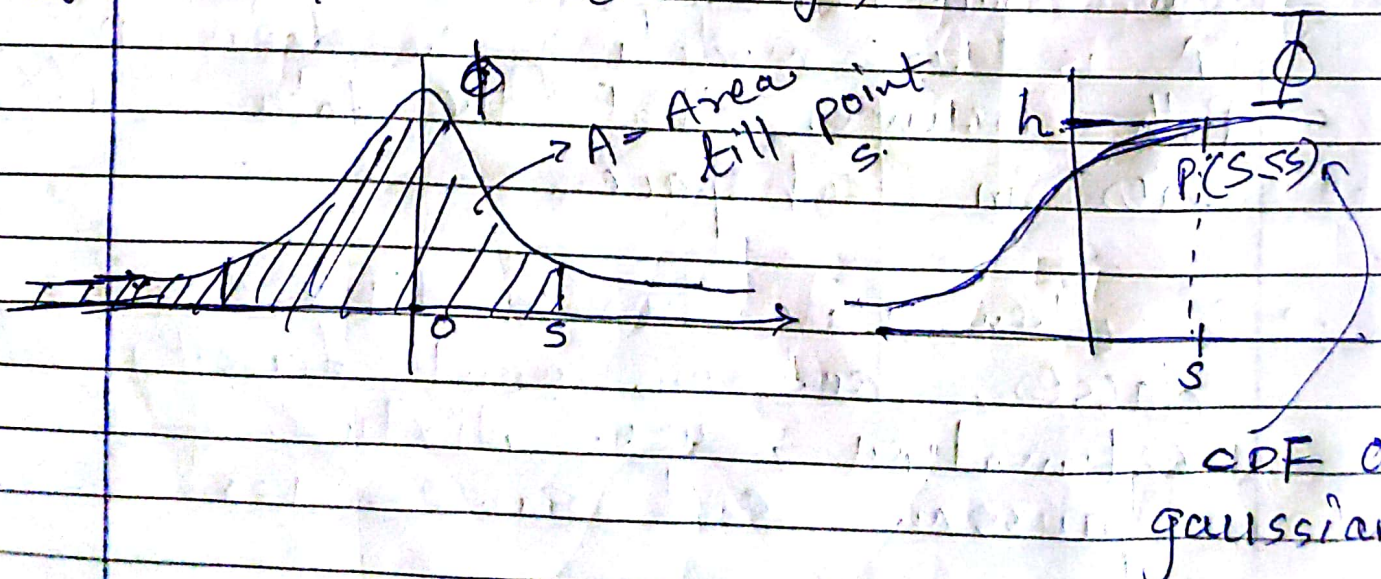
→ One example where we can use ICA is cocktail party problem.

* CDF (Cummulative distribution function):

→ Let's say we've one-dimensional random variable s and probability density function $P_s(s)$, then

CDF $F(s) = P(S \leq s)$

Eg. For Gaussian density;



CDF of gaussian

PAGE NO.
DATE

→ Height of P(S≤s) in Φ(h) is equal to the area covered by Φ (small Φ means gaussian bell-shaped curve) till the point s.

∴ A = h.

→ In CDF, as height approaches to 1 (x → ∞), the area also approaches to 1.

→ From above intuition, Definition of CDF can be given by :-

→ For any real-valued random variable S, or distribution function of S, evaluated at s, CDF is the probability that S will take a value less than or equal to s.

→ So, for continuous distribution functions, CDF is given by

$$\text{CDF } F(s) = \int_{-\infty}^{s} P_S(t)\,dt.$$

→ Suppose we're a random variable S and we want to model the distribution of Random variable S, then,

PAGE NO.
DATE :

s → specify $P_s(s)$ or
→ specify $F(s)$

→ Because both are related.

$$P_s(s) = F'(s)$$

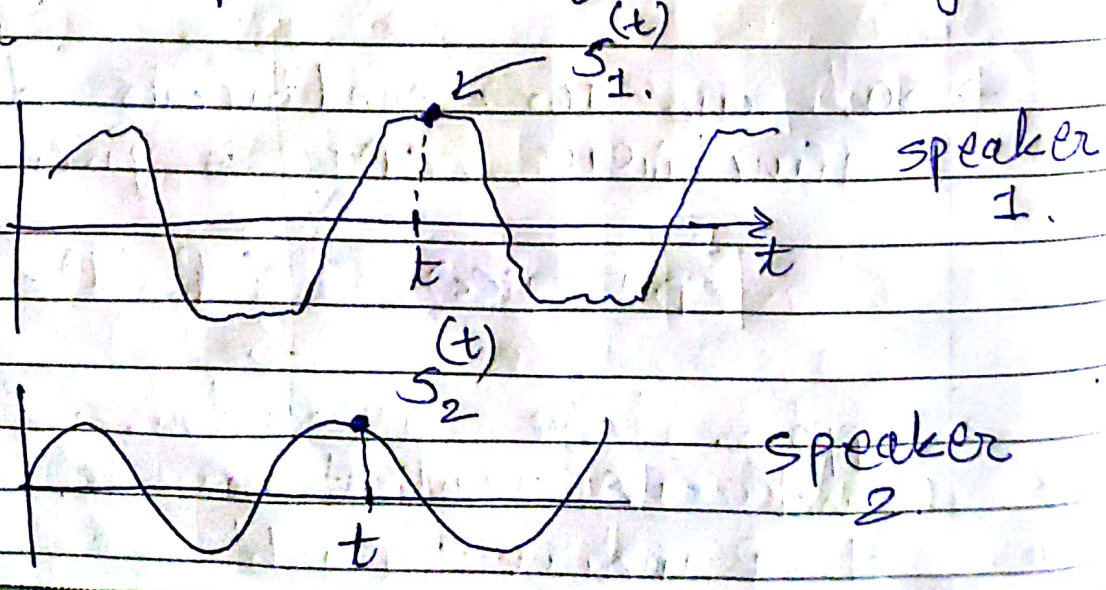
* ICA :-

→ Suppose there are n original data sources. (e.g. n different speakers in cocktail party).

$$s \in R^n$$

$s_j^{(i)}$ means signal from speaker j at time i.

→ Sound is represented by following in graph



→ We observe,

$$x^{(i)} = As^{(i)}, \text{ where } x^{(i)} \in R^n$$

assume we've n microphones. ($x^{(i)}$)

→ so, below equation means that each microphones records some overlapping linear combination of speakers.

→ For ex.

$x_j^{(i)}$ (microphone j records at time i).

$$x_j^{(i)} = \sum_k A_{jk} s_k^{(i)}$$

→ Goal :- Find a matrix $W = A^{-1}$, so that,

$$s^{(i)} = W x^{(i)}$$

→ $Wx^{(i)}$ is original recordings of speaker $s^{(i)}$

→ notation of W,

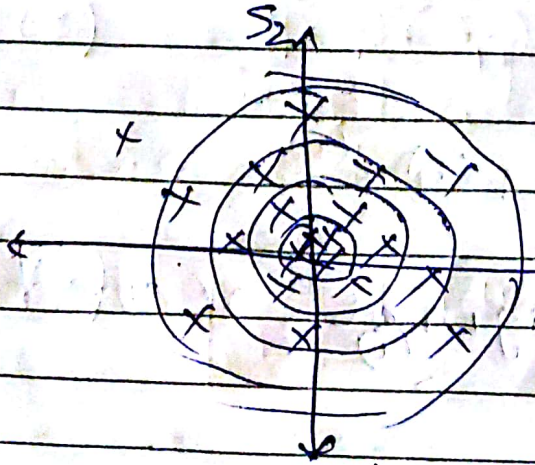
$$W = \begin{bmatrix} — \omega_1^T — \\ — \omega_2^T — \\ \vdots \\ — \omega_n^T — \end{bmatrix}$$

* Ambiguities of ICA:-

1) Order or permutation of speakers recovered not guareenteed.
2) sign of the speakers.

→ The reason for only above 2 ambiguities is that $s_j^{(i)}$ is non-gaussian.

→ But suppose that $s^{(i)} \sim \mathcal{N}(0, I)$, then the data contours would be :-



→ Now, as gaussian distribution is rotationaly symmetric, we can't

recover the original speaker sounds. Because, there is no way we can tell the two axes s_1 and s_2 located.

→ Let $s \in R^n$

Density of s: $P_s(s)$

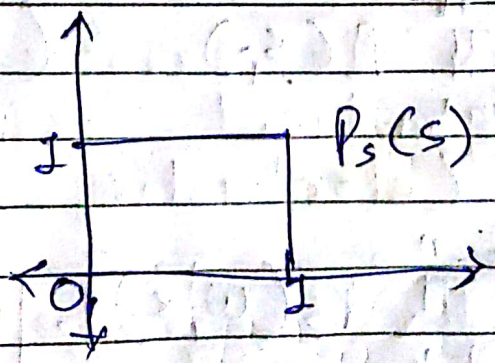
$x = As = W^{-1}s$, $s = Wx$

→ A is mixing matrix and W is unmixing matrix.

$$P_x(x) = P_s(Wx)\,|W|$$

Eg. $P_s(s) = 1\{0 \le s \le 1\}$

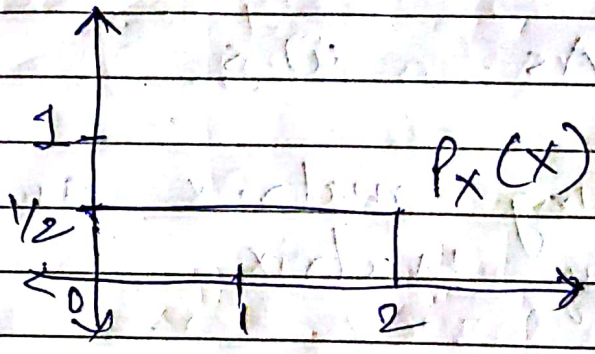
($s \sim$ Uniform $[0, 1]$)



Let $x = 2s$, then $A = 2$, $\omega = 1/2$

→ now, s is uniform distribution over $[0, 1]$, so $x = 2s$ will be uniform distribution over $[0, 2]$.

→ So, density for x would be



$$P_x(x) = 1\{0 \le x \le 2\} \, 1/2$$

* ICA model:- → n speakers.

$$P(s) = \prod_{i=1}^{n} P_s(s_i) \quad (s_i \text{ are independent})$$

$$P(x) = \left[\prod_{i=1}^{n} P_s(w_i^T x)\right] \cdot |W|$$

PAGE NO.
DATE

$\because W = A^{-1} = \begin{bmatrix} — w_1^T — \\ \vdots \end{bmatrix}$, so, $s_i = w_i^T x$.

→ Now, choose density (probability) of what each speakers is speaking.

$$P_s(s_i) = ?$$

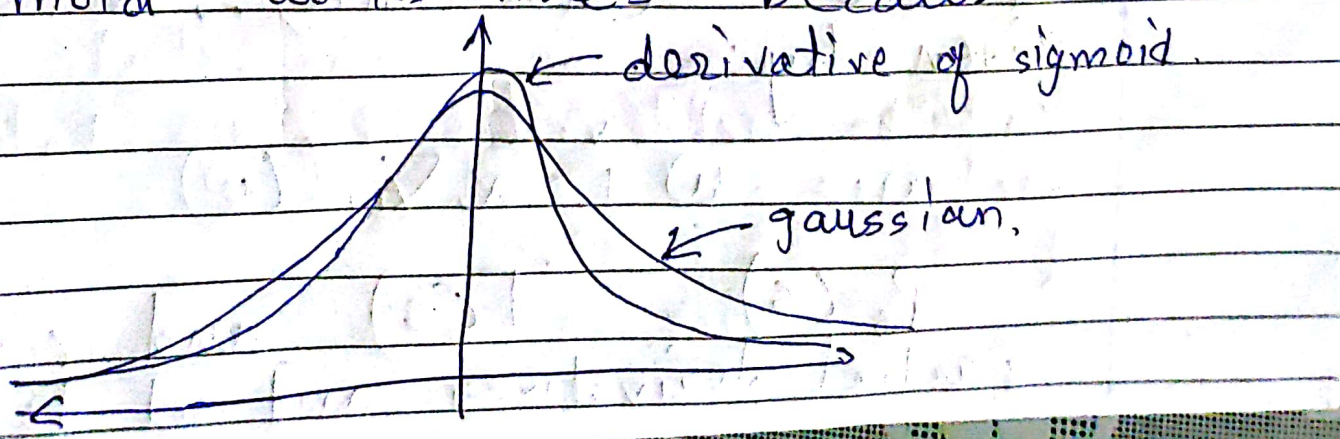
→ As discussed in CDF, we can either choose CDF or density.

→ So, let's choose CDF to be sigmoid function. We can't choose gaussian as mentioned before.

$$\text{So, } F(s) = \frac{1}{1+e^{-s}}.$$

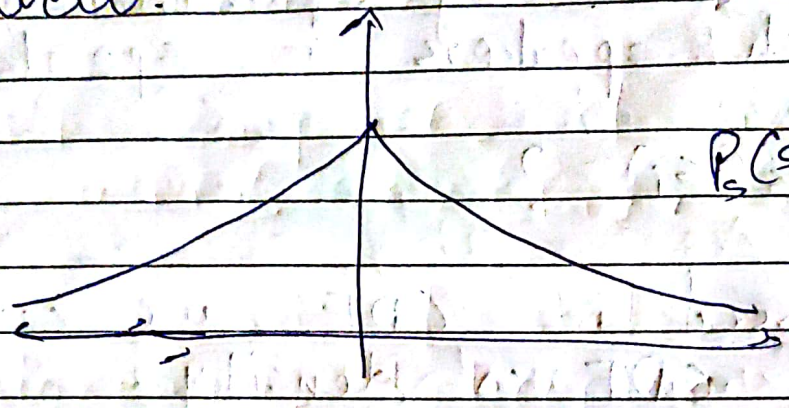
$$P_s(s) = F'(s_i)$$

→ sigmoid works fines because



→ Gaussian decreases by e^{-s}, while sigmoid's derivation by e^{-s^2}

→ Also, Possion distribution works well.



$$P_s(s) = \frac{1}{2} e^{-|s|}$$

→ Given $\{x^{(1)}, \ldots x^{(m)}\}$ training set, log likelihood of examples is given by

$$l(\omega) = \sum_i \log \left(\prod_j P_s(\omega_j^T x^{(i)}) \right)_{|\omega|}$$

→ And to learn the parameters (ω), use the gradient ~~descent~~ ascent. (stochastic), which is given by:-

$$\omega := \omega + \alpha \nabla_\omega l(\omega)$$

→ $P_s(s) = F'(s)$ and compute partial derivatives and set to zero,

PAGE NO.
DATE

we find,

$$\nabla_W L(W) = \begin{bmatrix} 1 - 2g(w_1^T x) \\ \vdots \\ 1 - 2g(w_n^T x) \end{bmatrix} + (W^T)^{-1}$$

→ when we find W, we can recover each sources by

$$s^{(i)} = W x^{(i)}$$

PAGE NO.
DATE : 29/7/16

Lecture-16

* Reinforcement Learning:-

→ RL problems model the world using MDP. (Marcov decision process)

→ MDP comprises of five elements

MDP $(S, A, \{P_{sa}\}, \gamma, R)$

S → set of states [In case of autonomous helicoptor set of positions and set of orientations of helicoptor]

A → set of Action [set of positions in which we can control the helicoptor]

P_{sa} → State transition distributions

$$\sum_{s'} P_{sa}(s') = 1 \text{ and } P_{sa}(s') \geq 0$$

P_{sa} means in that state (s) will be achieved if we take action 'a'.

→ P_{sa} gives the probability distribution of what state we would transition to, if we take action 'a' in state 's'.

→ So, $P_{sa}(s')$ means the probability distribution of state s' achieved by us, when we take action 'a' from state 's'.

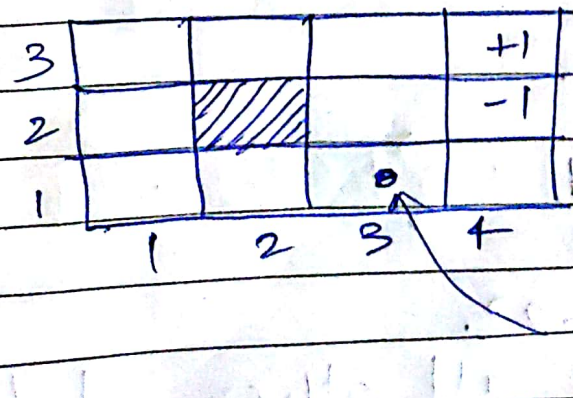
γ → Discount factor, $0 \leq \gamma < 1$

R → Reward function.

$$R : S \rightarrow \mathbb{R}$$

↖ set of real numbers.

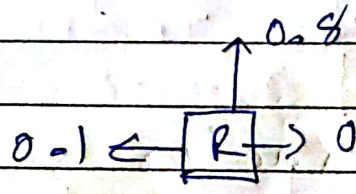
→ Example problem:-



→ MDP for this problem.

S → 11 states.
A = {North, South, East, West}

→ Robot dynamics for moving in this grid.
If we command the robot to go north,



→ It's 80% chance in which robot goes north, 10% in east & west.

→ P((3,1), N, (3,2)) = 0.8
P((3,1), N, (4,1)) = 0.1
P((3,1), N, (2,1)) = 0.1
P((3,1), N, (3,3)) = 0
∴ etc.

→ R((4,3)) = +1
R((4,2)) = -1
R(s) = -0.02
for all other states

PAGE NO.
DATE

→ Now let's see how MDP works.

→ At time zero(0), and initial state S_0, Robot starts. It's task is to choose action a_0

→ Depending on choice, we get some state $S_1 \sim P(S_0, a_0)$

→ Now choose action a_1 from new state S_1.

→ So, it get to new state $S_2 \sim P(S_1, a_1)$

→ After doing this for a while, Robot has visited a sequence of states

$$S_0 \rightarrow S_1 \rightarrow S_2 \cdots$$

→ And to see how well the robot did, sum up the rewards it got.

so,

$$R(S_0) + R(S_1) + R(S_2) + \cdots$$

→ Let's change this total reward as following.

Total Payoff = $\gamma^0 R(s_0) + \gamma R(s_1) + \gamma^2 R(s_2) + \cdots$

$\gamma \in [0, 1)$

→ So, what this does is control the reward with time.

→ So, Reward we got at time zero (initially on step s_0) is more then step s_1 and so on.

→ So the goal of RL algo. is to choose the actions over time $(a_0, a_1, a_2 \cdots)$ to maximize the expectation of total payoff.

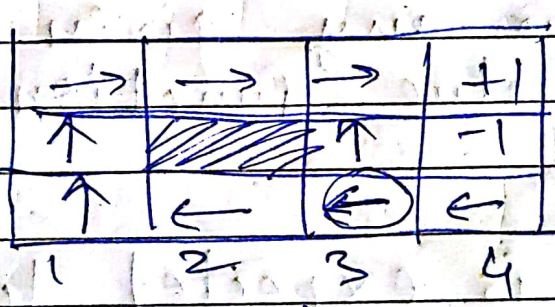
$$E[R(s_0) + \gamma R(s_1) + \gamma^2 R(s_2) + \cdots]$$

→ RL algo. computes the policy

$$\Pi : S \rightarrow A$$

→ Policy is just the mapping from state to action so it tells the algo. that when you are at this state, perform that action

→ Optimal policy for this example is given by



→ So, when we execute the optimal policy, it will give max. total pay off.

→ MDP is good at making certain trade offs. consider the position (3,1), here we can go North instead of following the longer path.

→ But if we go north, there are certain chances (0.1) that Robot end up going Right side (Bcz of Robot dynamics we defined previously)

→ So, discount factor discourage this and rather takes longer route.

* Definitions:-

→ V^{π} :- for any given policy π define value function $V^{\pi} : S \to \mathbb{R}$ such that

$V^{\pi}(s)$ is expected total payoff, starting in state s, and execute policy π.

$$V^{\pi}(s) = E[\text{Total payoff} \mid \pi, s_0 = s]$$

π is not a random variable but the executed policy.