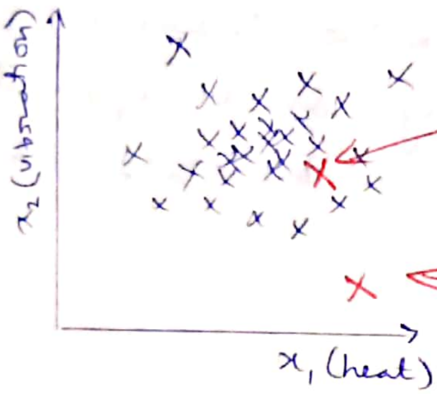
Week 9

Anomaly detection

Eg. of anomaly detection

Aircraft engine features:
x_1 = heat generated
x_2 = Vibration intensity

Dataset: $\{x^{(1)}, x^{(2)}, \dots, x^{(m)}\}$
New engine: x_{test}



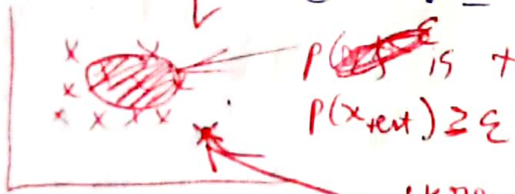
Engine looks ok

Anomaly

X - is the new engine that needs to be tested

Density estimation:

We build a model that gives prediction $p(x)$ and if $p(x_{test}) < \varepsilon \rightarrow$ flag as anomaly
$p(x_{test}) \geq \varepsilon \rightarrow$ OK



$p(x)$ is this area
$p(x_{test}) \geq \varepsilon$

Here $p(x_{test}) < \varepsilon$ so anomaly

Applications:

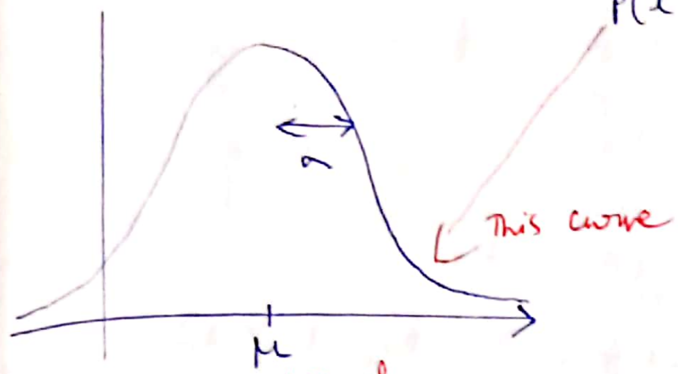
- Fraud detection - $x^{(i)}$ is features of user i's activity (logins, typing speed, etc). A fraud will show abnormal behavior. ($p(x) < \varepsilon$)
- Manufacturing
- Monitoring computers in a data center

Gaussian (Normal) distribution

Say $x \in \mathbb{R}$, If x is a distributed Gaussian with mean μ, variance σ^2

probability of x parameterised by μ & σ^2

$$p(x;\mu,\sigma^2) = \frac{1}{\sqrt{2\pi}\,\sigma} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$$



$x \sim \mathcal{N}(\mu, \sigma^2)$

Normal

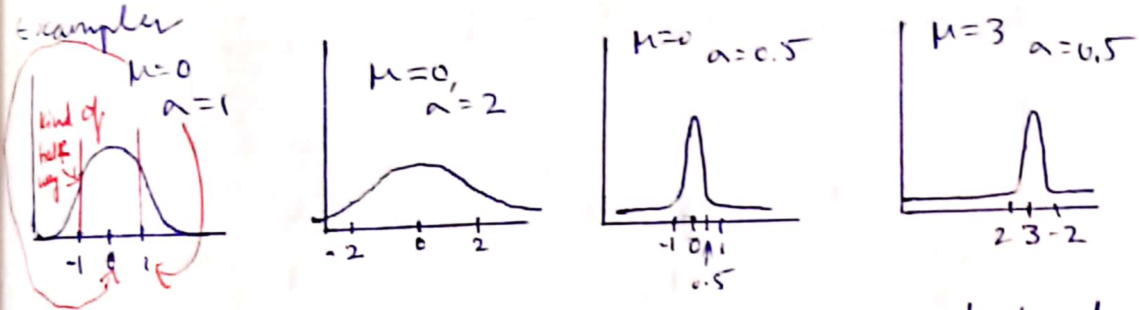
distributed as

mean

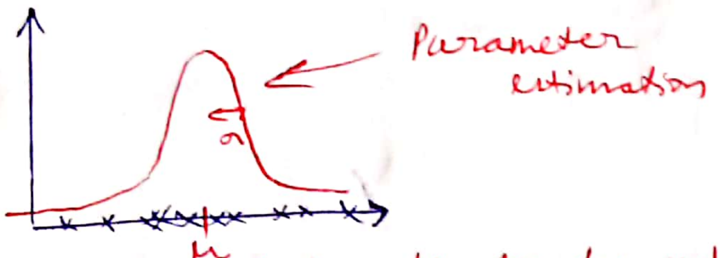
σ^2 is variance

σ is standard deviation

Examples



Parameter estimation : Lets say we suspect that $x^{(i)} \sim \mathcal{N}(\mu,\sigma^2)$ - the data was gaussian



Given my data set I want to estimate the value μ & σ^2.

$$\mu = \frac{1}{m}\sum_{i=1}^{m} x^{(i)} \qquad \sigma^2 = \frac{1}{m}\sum_{i=1}^{m}\left(x^{(i)}-\mu\right)^2$$

Algorithm

Given training set: $\{x^{(1)}, \dots, x^{(m)}\}$, each example is $x \in \mathbb{R}^n$

Where $x_1 \sim \mathcal{N}(\mu_1, \sigma_1^2)$, $x_2 \sim \mathcal{N}(\mu_2, \sigma_2^2)$, $x_3 \sim \mathcal{N}(\mu_3, \sigma_3^2)$

$$\therefore\ p(x) = p(x_1;\mu_1,\sigma_1^2)\cdot p(x_2;\mu_2,\sigma_2^2)\cdot p(x_3;\mu_3,\sigma_3^2)\cdots p(x_n;\mu_n,\sigma_n^2)$$

Π- product $\rightarrow$ $= \prod_{j=1}^{n} p(x_j;\mu_j,\sigma_j^2)$

This is like Σ (summation) but used for product

Anamoly detection algoritm

1. Choose features x_i that you think might be indicative of anomalous examples.

2. Fit parameters $\mu_1, \dots, \mu_n, \sigma_1^2, \dots, \sigma_n^2$

$$\mu_j = \frac{1}{m}\sum_{i=1}^{m} x_j^{(i)}$$

$$\sigma_j^2 = \frac{1}{m}\sum_{i=1}^{m} (x_j^{(i)} - \mu_j)^2$$

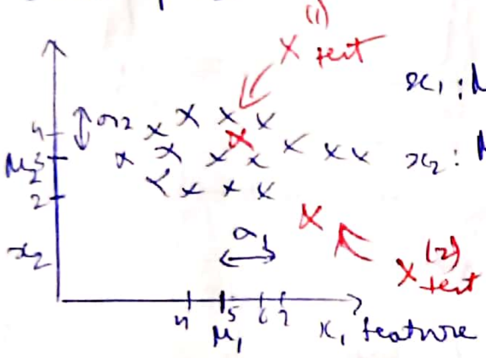
3. Given new example x, compute $p(x)$:

$$p(x) = \prod_{j=1}^{n} p(x_j; \mu_j, \sigma_j^2) = \prod_{j=1}^{n} \frac{1}{\sqrt{2\pi}\sigma_j} \exp\left(-\frac{(x_j - \mu_j)^2}{2\sigma_j^2}\right)$$

$\sqrt{2\pi}\,\sigma_j$

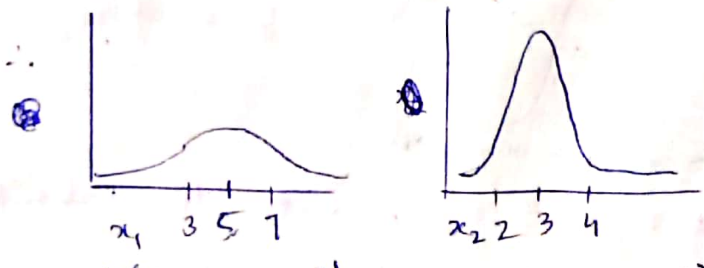
Anamaly if $p(x) < \varepsilon$

Example:



$x_1: \mu_1 = 5, \sigma_1 = 2, \sigma_1^2 = 4$

$x_2: \mu_2 = 3, \sigma_2 = 1, \sigma_2^2 = 1$



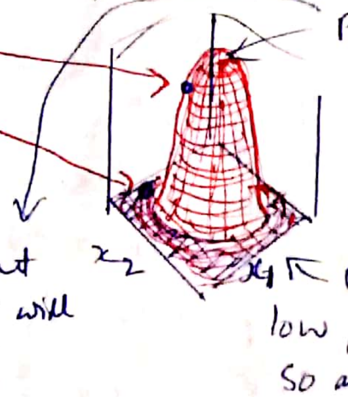
$p(x_1; \mu_1, \sigma_1^2)$ $\qquad$ $p(x_2; \mu_2, \sigma_2^2)$

$\therefore p(x) = p(x_1; \mu_1, \sigma_1^2) \times p(x_2; \mu_2, \sigma_2^2)$

$\varepsilon = 0.02$

$P(x^{(1)}_{test}) = 0.0426 \geq \varepsilon$

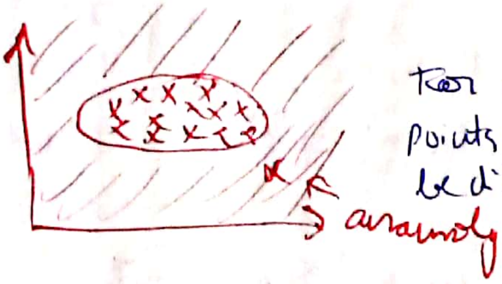
$P(x^{(2)}_{test}) = 0.0021 < \varepsilon$



$p(x)$ is the height

Here low probability so anamoly

For different points height will be different



Building an anamoly detection system

We need a number (like accuracy) to evaluate the performance of our learning algorithm. We have discussed this before. If we make a change (exa: increase features) we can see its result on the number to decide whether the change helped or not.

Evaluating anamoly detection system:

→ Assume we have some labelled data, of anomalous & non-anomalous examples ($y = 0$ if normal, $y = 1$ if anomalous)

→ Training set: $x^{(1)}, x^{(2)}, \ldots, x^{(m)}$ (assume normal examples / not anomalous)

→ Cross validation set: $(x_{cv}^{(1)}, y_{cv}^{(1)}), \ldots, (x_{cv}^{(m_{cv})}, y_{cv}^{(m_{cv})})$

→ Test set: $(x_{test}^{(1)}, y_{test}^{(1)}), \ldots, (x_{test}^{(m_{test})}, y_{test}^{(m_{test})})$

Exa: Aircraft engines

→ Lets say there are

→ 10000 good (normal) engines

→ 20 flawed ~~engines~~ engines (anomalous)

Training set: 6000 good engines ($y=0$)

$\mu_1, \sigma_1^2, \ldots \mu_n, \sigma_n^2$ $\quad p(x) = p(x_1; \mu_1, \sigma_1^2) \cdots p(x_n; \mu_n, \sigma_n^2)$

CV: 2000 good engines ($y=0$), 10 anomalous ($y=1$)

Test: 2000 good engines ($y=0$), 10 anomalous ($y=1$)

Algorithm evaluation:

→ Fit model $p(x)$ on training set $\{x^{(1)}, \ldots, x^{(m)}\}$

→ On ~~a~~ cross validation / test example x, predict

$$y = \begin{cases} 1 & \text{if } p(x) < \varepsilon \text{ (anomaly)} \\ 0 & \text{if } p(x) \geq \varepsilon \text{ (normal)} \end{cases}$$

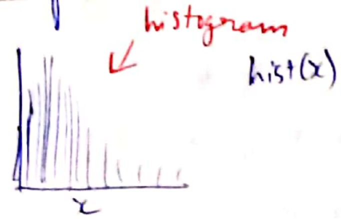
→ Since the data is skewed (only 20 $y=1$ examples) we use F_1-score

→ We can use CV set to choose parameter ε

Scanned with CamScanner

Anamoly Detection	Supervised learning
→ Very small no. of +ve examples ($y=1$). Exa: 0-20	→ Large no. of +ve & -ve examples
→ Large no. of ~~a~~ -ve ($y=0$) examples (we fit $p(x)$ on this)	→ Use if there are enough ~~the~~ +ve examples for the algorithm to get a sense of what +ve examples are like
→ use if There are many "types" of anamalies and the model can't learn all of them (Exa: In aircraft engine many things can go wrong) - It is better to group the OK engines and flag everything else as bad rather than trying to learn what bad engines are	
→ Applications • Fraud detection • Manufacturing • Monitoring machines in a data centre	→ Applications • Email

If your data is non-gaussian (non-gaussian features)



Then transform the data by trying the following:

$x_1 \leftarrow \log(x_1)$, $x_2 \leftarrow \log(x_2+1)$,

$x_3 \leftarrow \sqrt{x_3}$ or $x_3^{1/2}$, $x_4 \leftarrow x_4^{1/3}$

and see if ~~the~~ it becomes gaussian

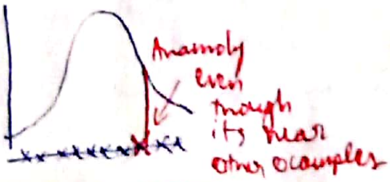
Error analysis for anomaly detection

we want $p(x)$ large for normal examples x

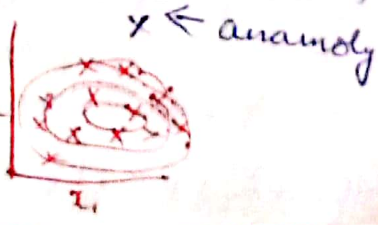
$p(x)$ small for anomalous examples x

Most common problem:

$p(x)$ is comparable ~~or~~ (say, both large) for normal & anomalous examples



So we find a new feature x_2 →



Choosing ~~features~~ what features to use:

Choose features that might take unusually large or small values in the events of an anomaly.

exa: Monitoring computers in a data centre

Given x_1 - memory use of computer

x_2 - no. of disk accesses/sec

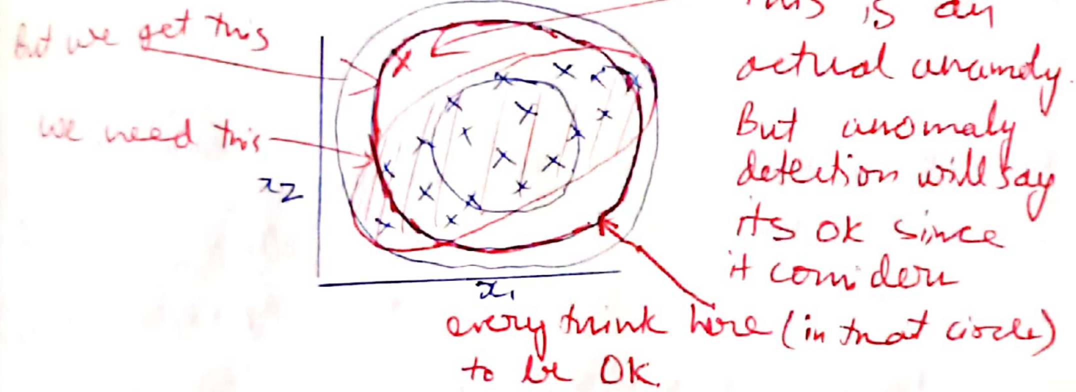
x_3 - cpu load x_4 - network traffic

Then we can use $x_5 = \frac{\text{cpu load}}{\text{network traffic}}$ - if this is high its abnormal since cpu load $\propto$ network traffic

To choose new features automatically

This can automatically capture correlation between different features

Multivariate gaussian distribution



$\therefore x \in \mathbb{R}^n$. Don't model $p(x_1), p(x_2), \ldots$, etc separately.

Model $p(x)$ all in one go.

Parameters: $\mu \in \mathbb{R}^n$, $\Sigma \in \mathbb{R}^{n \times n}$ (covariance matrix)

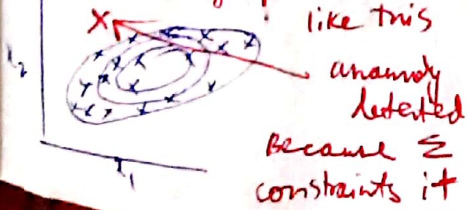
$$p(x; \mu, \Sigma) = \frac{1}{(2\pi)^{n/2} |\Sigma|^{1/2}} \exp\left(-\tfrac{1}{2}(x-\mu)^T \Sigma^{-1} (x-\mu)\right)$$

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

Anamoly detection with the multivariate gaussian.

1. Fit model $p(x)$ by seting
2. Given a new example x, compute
   Flag an anamoly if $p(x) < \varepsilon$

Then, the graph will come like this



Because Σ constraints it

But this is computationally expensive

And $\underset{\text{training set size}}{m} > \underset{\text{no. of features}}{n}$ (In original model even if m is small it works)

Recommender System - Predicting movie ratings

User rates movies using 0-5 stars. The system predicts the star rating for the movies a user hasn't watched

	Movie	Alice(1)	Bob(2)	Carol(3)	Dave(4)
Romance movies	Love at last	5	5	0	0
	Romance Forever	5	? 4.5	? 0	0
	Cute puppies of love	? 5	4	0	? 0
Action movies	Nonstop car chases	0	0	5	4
	Swords vs karate	0	0	5	? 4

These are prediction

n_u - no. of users n_m - no. of movies

$r(i,j)$ - 1 if user j has rated movie i, else 0

$y(i,j)$ - rating given by user j to movie i (defined only if $r(i,j)$).

Content based recommender systems

$n_u = 4$ $n_m = 5$

Movie	Alice(1) $\theta^{(1)}$	Bob(2) $\theta^{(2)}$	Carol(3) $\theta^{(3)}$	Dave(4) $\theta^{(4)}$	x_1 (romance)	x_2 (action)
$x^{(1)}$ Love at last	5	5	0	0	0.9	0
$x^{(2)}$ Romance Forever	5	?	?	0	1.0	0.01
$x^{(3)}$ Cute puppies of love	? 4.95	4	0	?	0.99	0
$x^{(4)}$ Nonstop car chases	0	0	5	4	0.1	1.0
$x^{(5)}$ Swords vs karate	0	0	5	?	0	0.9

Exa

we set parameters that define how much romance & action a movie has

$$x^{(1)} = \begin{bmatrix} 1 \\ 0.9 \\ 0 \end{bmatrix} \leftarrow \text{we set } x_0 = 1$$

For each user j, learn a parameter $\theta^{(j)} \in \mathbb{R}^3$.

Predict user j as rating movie i with $(\theta^{(j)})^T x^{(i)}$ stars

$\therefore$ For example: $x^{(3)} = \begin{bmatrix} 1 \\ 0.99 \\ 0 \end{bmatrix}$ $\theta^{(1)} = \begin{bmatrix} 0 \\ 5 \\ 0 \end{bmatrix}$ ← After training we get this

$$\therefore (\theta^{(1)})^T x^{(3)} = 5 \times 0.99 = 4.95$$

Problem Formulation

$r(i,j) = 1$ if user j has rated movie i (0 otherwise)

$y^{(i,j)}$ = rating by user j on movie i (if defined)

$\theta^{(j)}$ = parameter vector for user j

$x^{(i)}$ = feature vector for movie i

For user j, movie i, predicted rating: $(\theta^{(j)})^T(x^{(i)})$

$m^{(j)}$ = no. of movies rated by user j

To learn $\theta^{(j)}$,

$$\min_{\theta^{(j)}} \frac{1}{2\cancel{m^{(j)}}} \sum_{i:r(i,j)=1} \left((\theta^{(j)})^T(x^{(i)}) - y^{(i,j)}\right)^2 + \frac{\lambda}{2\cancel{m^{(j)}}} \sum_{k=1}^{n} (\theta_k^{(j)})^2$$

we remove $m^{(j)}$ since we multiply the equation by a constant $m^{(j)}$

To learn $\theta^{(j)}$ (parameter for user j):

$$\min_{\theta^{(j)}} \frac{1}{2} \sum_{i:r(i,j)=1} \left((\theta^{(j)})^T x^{(i)} - y^{(i,j)}\right)^2 + \frac{\lambda}{2} \sum_{k=1}^{n} (\theta_k^{(j)})^2$$

To learn $\theta^{(1)}, \theta^{(2)} \ldots \theta^{(n_u)}$,

$$\min_{\theta^{(1)},\ldots,\theta^{(n_u)}} \frac{1}{2} \sum_{j=1}^{n_u} \sum_{i:r(i,j)=1} \left((\theta^{(j)})^T x^{(i)} - y^{(i,j)}\right)^2 + \frac{\lambda}{2} \sum_{j=1}^{n_u} \sum_{k=1}^{n} (\theta_k^{(j)})^2$$

$J(\theta^{(1)}, \ldots \theta^{(n_u)})$ - optimization algorithm

Gradient descent update:

$$\theta_k^{(j)} := \theta_k^{(j)} - \alpha \sum_{i:r(i,j)=1} \left((\theta^{(j)})^T x^{(i)} - y^{(i,j)}\right) x_k^{(i)} \quad (\text{for } k = 0)$$

$$\theta_k^{(j)} := \theta_k^{(j)} - \alpha \underbrace{\left(\sum_{i:r(i,j)=1} \left((\theta^{(j)})^T x^{(i)} - y^{(i,j)}\right) x_k^{(i)} + \lambda \theta_k^{(j)} \right)}_{\text{This is nothing but } \frac{\partial}{\partial \theta_k^{(j)}} J(\theta^{(1)}, \ldots, \theta^{(n_u)})} \quad (\text{for } k \neq 0)$$

Collaborative Filtering

Before we were given $x^{(1)}, \dots, x^{(n_m)}$ (and movie ratings)
& we estimated $\theta^{(1)}, \dots, \theta^{(n_u)}$

Now we are given $\theta^{(1)}, \dots, \theta^{(n_u)}$
& we estimate $x^{(1)}, \dots, x^{(n_m)}$

$\therefore$ we can first given θ, find x, then find θ, then find x.
& refine our parameter

Given $\theta^{(1)}, \dots, \theta^{(n_u)}$ to learn $x^{(1)}, \dots, x^{(n_m)}$

$$\min_{x^{(1)},\dots,x^{(n_m)}} \frac{1}{2}\sum_{i=1}^{n_m} \sum_{j:r(i,j)=1} \left((\theta^{(j)})^T x^{(i)} - y^{(i,j)}\right)^2 + \frac{\lambda}{2}\sum_{i=1}^{n_m}\sum_{k=1}^{n}(x_k^{(i)})^2$$

Its called collaborative since users collaborate to rate the movies & help the algorithm predict better.

Algorithm

Instead of going back & forth, we have an algorithm that minimizes both $x^{(1)} \dots x^{(n_m)}$ & $\theta^{(1)} \dots \theta^{(n_u)}$ ~~together~~ Simultaneously

$$\min_{\substack{x^{(1)},\dots,x^{(n_m)} \\ \theta^{(1)},\dots,\theta^{(n_u)}}} J(x^{(1)},\dots,x^{(n_m)}, \theta^{(1)} \dots \theta^{(n_u)}) = \frac{1}{2}\sum_{(i,j):r(i,j)=1} \left((\theta^{(j)})^T x^{(i)} - y^{(i,j)}\right)^2 + \frac{1}{2}\sum_{i=1}^{n_m}\sum_{k=1}^{n}(x_k^{(i)})^2 + \frac{1}{2}\sum_{j=1}^{n_u}\sum_{k=1}^{n}(\theta_k^{(j)})^2$$

1. Initialize $x^{(1)},\dots,x^{(n_m)}, \theta^{(1)},\dots,\theta^{(n_u)}$ to small random values.
2. Minimize $J(x^{(1)},\dots,x^{(n_m)}, \theta^{(1)},\dots,\theta^{(n_u)})$ using gradient descent (or an advanced optimization algorithm). E.g. for every $j=1, \dots n_u, i=1 \dots$

$$x_k^{(i)} := x_k^{(i)} - \alpha\left(\sum_{j:r(i,j)=1}\left((\theta^{(j)})^T x^{(i)} - y^{(i,j)}\right)\theta_k^{(j)} + \lambda x_k^{(i)}\right)$$

$$\theta_k^{(j)} := \theta_k^{(j)} - \alpha\left(\sum_{i:r(i,j)=1}\left((\theta^{(j)})^T x^{(i)} - y^{(i,j)}\right)x_k^{(i)} + \lambda \theta_k^{(j)}\right)$$

3. For a user with parameters θ and a movie with (learned) features x, predict a star rating of $\theta^T x$.

Scanned with CamScanner

Vectorization - Low Rank Matrix Factorization

$$Y = \begin{bmatrix} 5 & 5 & 0 & 0 \\ 5 & ? & ? & 0 \\ ? & 4 & 0 & ? \\ 0 & 0 & 5 & 4 \\ 0 & 0 & 5 & 0 \end{bmatrix}$$

Predicted ratings:

$$\begin{bmatrix} (\theta^{(1)})^T(x^{(1)}) & (\theta^{(2)})^T(x^{(1)}) & \cdots & (\theta^{(n_u)})^T(x^{(1)}) \\ (\theta^{(1)})^T(x^{(2)}) & (\theta^{(2)})^T(x^{(2)}) & \cdots & (\theta^{(n_u)})^T(x^{(2)}) \\ \vdots & \vdots & & \vdots \\ (\theta^{(1)})^T(x^{(n_m)}) & (\theta^{(2)})^T(x^{(n_m)}) & \cdots & (\theta^{(n_u)})^T(x^{(n_m)}) \end{bmatrix}$$

$$X = \begin{bmatrix} - & (x^{(1)})^T & - \\ - & (x^{(2)})^T & - \\ & \vdots & \\ - & (x^{(n_m)})^T & - \end{bmatrix} \qquad \Theta = \begin{bmatrix} - & (\theta^{(1)})^T & - \\ - & (\theta^{(2)})^T & - \\ & \vdots & \\ - & (\theta^{(n_u)})^T & - \end{bmatrix}$$

This can be found by: $X\Theta^T$

For each product (movie in this case), we learn a feature vector $x^{(i)} \in \mathbb{R}^n$,

x_1 = romance, x_2 = action, x_3 = comedy ....

The model learns these features. In actual, we may not be able to categorise it like romance, etc by looking at it - we won't know what it'll be but it'll be something that the algorithm would have found in a pattern.

How to find movies j related to movie i?

Choose the movies that minimise $\|x^{(i)} - x^{(j)}\|$

Mean Normalization

If a user has not ~~predicted~~ rated any movies, we will end up predicting 0 for all movies. Instead we do mean normalisation

$$Y = \begin{bmatrix} 5 & 5 & 0 & 0 & ? \\ 5 & ? & ? & 0 & ? \\ ? & 4 & 0 & ? & ? \\ 0 & 0 & 5 & 4 & ? \\ 0 & 0 & 5 & 0 & ? \end{bmatrix}$$

← user who hasn't ~~pred~~ rated any movie

← Then all this will become 0

$\rightarrow 5+5+0+0/4$

$$\rightarrow \mu = \begin{bmatrix} 2.5 \\ 2.5 \\ 2 \\ 2.25 \\ 1.25 \end{bmatrix} \rightarrow Y = \begin{bmatrix} 2.5 & 2.5 & -2.5 & -2.5 & ? \\ 2.5 & ? & ? & -2.5 & ? \\ ? & 2 & -2 & ? & ? \\ -2.25 & -2.25 & 2.75 & 1.75 & ? \\ -1.25 & -1.25 & 3.75 & -1.25 & ? \end{bmatrix}$$

For user j, on movie i predict:

$\rightarrow (\theta^{(j)})^T(x^{(i)}) + \mu_i$

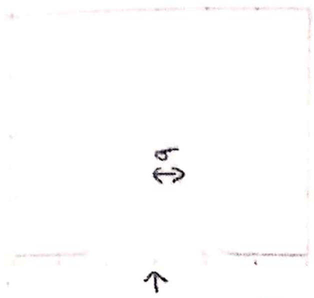
Since for user 5,

$\theta^{(5)} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \quad \therefore \quad \underbrace{(\theta^{(j)})^T(x^{(i)})}_{\text{This will be } 0} + \mu_i$ ←

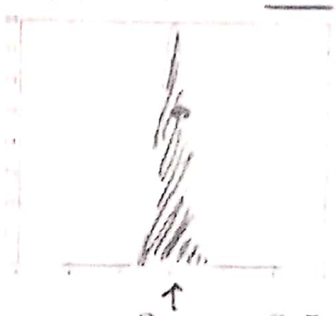
So instead of predicting 0, it'll predict μ_i (1.25 in this case)

Gaussian distribution example

$\rightarrow \mu = 0, \sigma = 1$

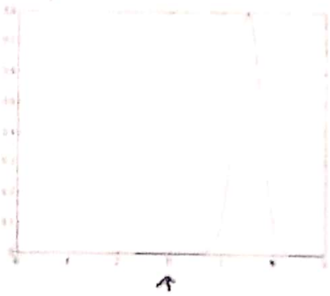


$\rightarrow \mu = 0, \sigma = \underline{0.5}$ $\quad \sigma^2 = 0.25$



$\rightarrow \mu = 0, \sigma = 2$

$\rightarrow \mu = 3, \sigma = 0.5$

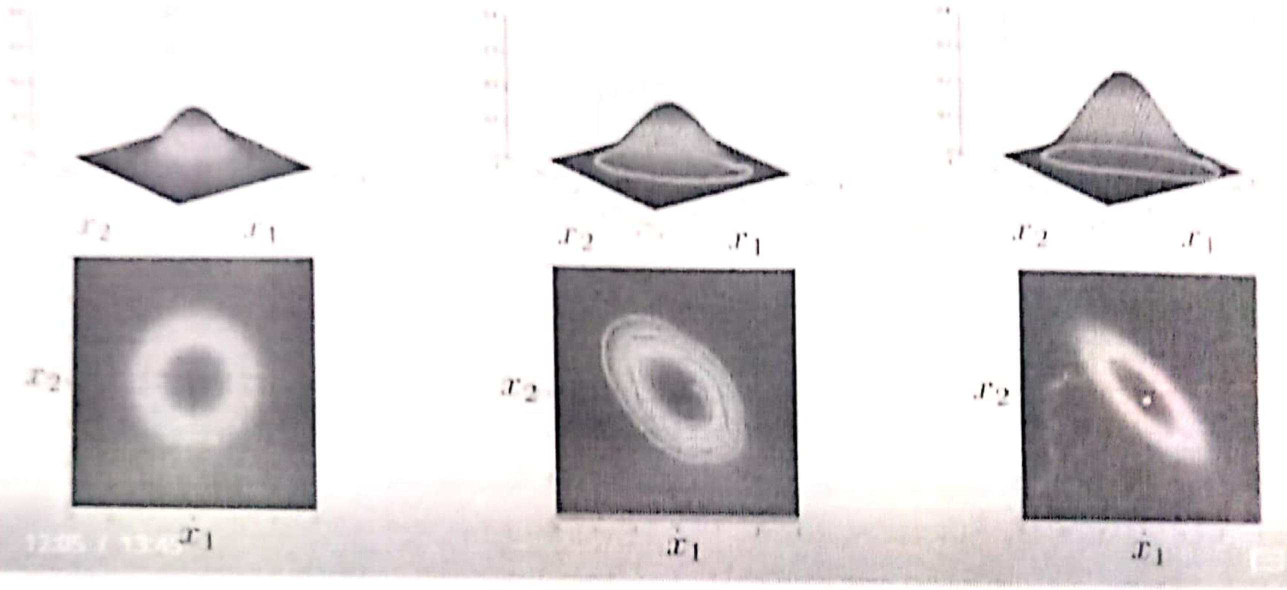


Multivariate Gaussian (Normal) examples

$\mu = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \quad \Sigma = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$

$\mu = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \quad \Sigma = \begin{bmatrix} 1 & 0.5 \\ 0.5 & 1 \end{bmatrix}$

$\mu = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \quad \Sigma = \begin{bmatrix} 1 & 0.8 \\ 0.8 & 1 \end{bmatrix}$

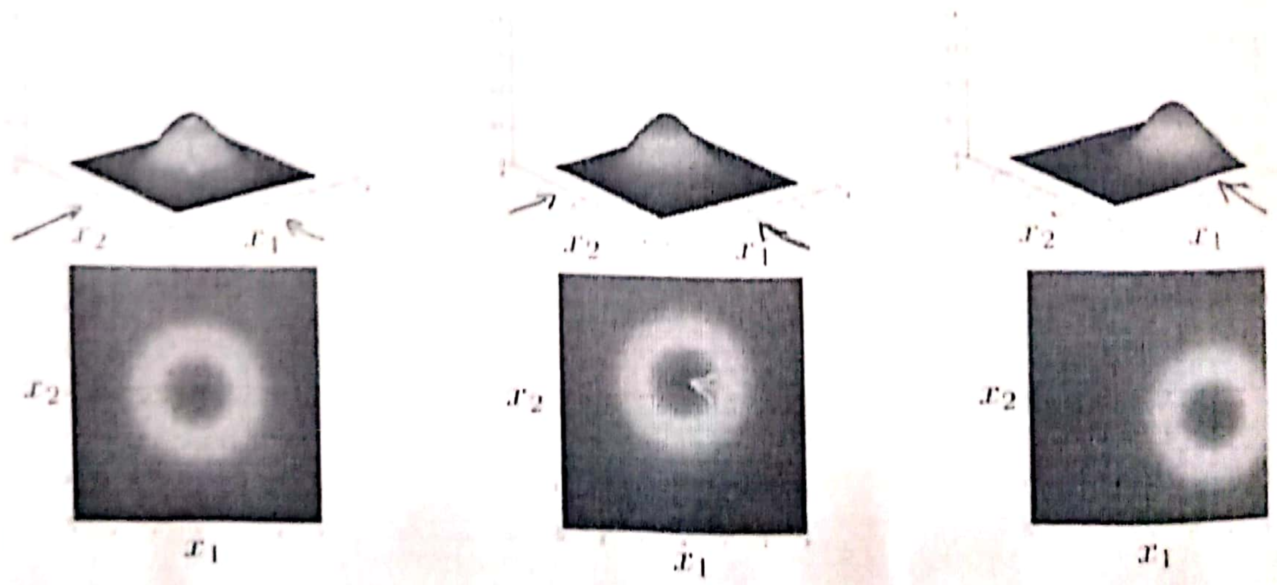


Multivariate Gaussian (Normal) examples

$\mu = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \quad \Sigma = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$

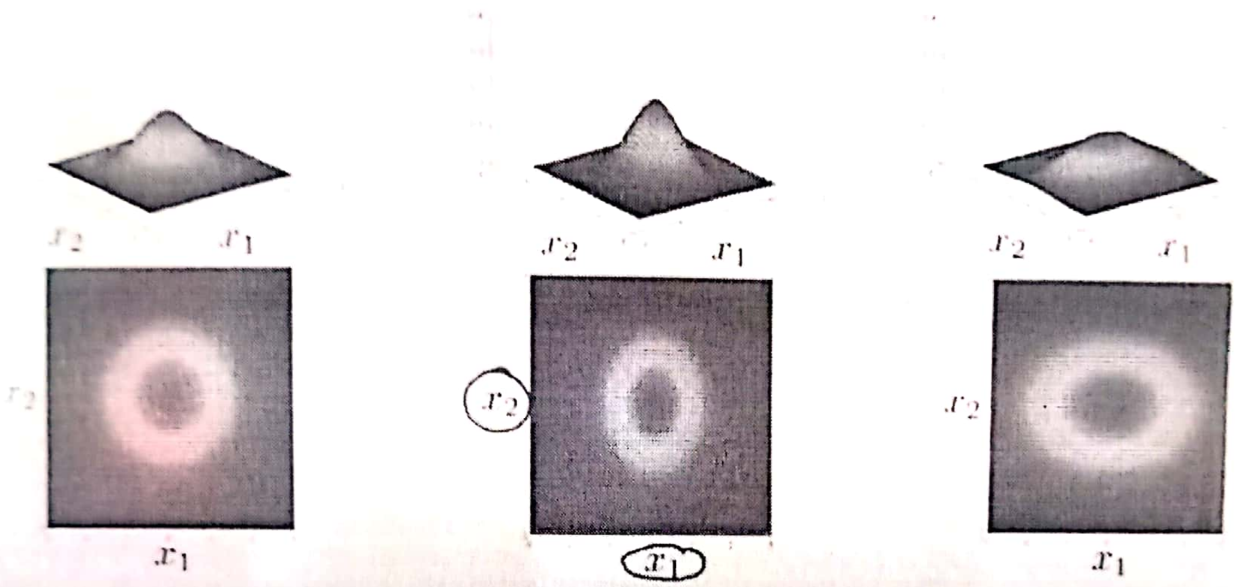
$\mu = \begin{bmatrix} 0 \\ 0.5 \end{bmatrix} \quad \Sigma = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$

$\mu = \begin{bmatrix} 1.5 \\ -0.5 \end{bmatrix} \quad \Sigma = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$



Multivariate Gaussian (Normal) examples

$\mu = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \Sigma = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$ $\mu = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \Sigma = \begin{bmatrix} 0.6 & 0 \\ 0 & 1 \end{bmatrix}$ $\mu = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \Sigma = \begin{bmatrix} 2 & 0 \\ 0 & 1 \end{bmatrix}$



Andrew Ng

Multivariate Gaussian (Normal) examples

$\mu = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \Sigma = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$ $\mu = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \Sigma = \begin{bmatrix} 1 & 0.5 \\ 0.5 & 1 \end{bmatrix}$ $\mu = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \Sigma = \begin{bmatrix} 1 & 0.8 \\ 0.8 & 1 \end{bmatrix}$