ML

Andrew Ng

Coursera

Scanned by CamScanner

Week 9

Date:

Big idea in ML - in some problems there are algorithms that can automatically learn features for you

Recommender Systems

Problem Formulation

Big idea in ML - in some problems there exist algos that can automatically learn features for you.

↳ recommender systems one such situation/problems

Example: predicting movie ratings

Users rate movies from 0 to 5 stars

Movie	Alice (1)	Bob (2)	Carol (3)	Dave (4)
Love at last	5	5	0	0
Romance forever	5	? (4.5)	? (0)	0
Cute ...	? (5)	4	0	? (0)
Nonstop ...	0	0	5	4
Swords vs...	0	0	5	? (4)

Scanned by CamScanner

$n_u = 4$, $n_m = 5$

Date:

Notation:

n_u = # of users

n_m = # of movies

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

$y(i,j)$ = rating given by user j to movie i
(defined only if $r(i,j) = 1$)

The first 3 movies are of romantic type while the last 2 are action. We can fill in the question marks of the unknown ratings

Given all $r(i,j)$ & $y(i,j)$, look through the data & predict missing ratings

↳ what value should they have

In realistic settings this table / matrix is more sparse as users would have rated only a small fraction of the movies available

Scanned by CamScanner

called content based because method assumes we have info about the content of movies / features about the movies

Date:

Content based Recommendation Systems

↳ first approach

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

$n_u = 4$, $n_m = 5$

Suppose your movie: x_1, x_2 have a set of features for each movie ($n = 2$)

Also $x_0 = 1$

So $x_1 = \begin{bmatrix} 1 \\ 0.9 \\ 0 \end{bmatrix}$ ← features of movie "love at last"

Victory

Page No.

Scanned by CamScanner

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

$\theta^{(j)} \rightarrow$ preference parameters of user j

So if we wish to find out Alice's preference for "Cute puppies of love" then:

$$x^{(3)} = \begin{bmatrix} 1 \\ 0.99 \\ 0 \end{bmatrix}, \quad \theta^{(1)} \begin{bmatrix} 0 \\ 5 \\ 0 \end{bmatrix}$$

↑ suppose value already learnt

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

Different copy of linear regression for eeech user

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:

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

Scanned by CamScanner

Date:

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

↑ temporarily introduced

To ~~introduce~~ learn $\theta^{(j)}$:

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

- $m^{(j)}$ is removed from this eqn for simplicity

Optimization Objective

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

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

↳ $J(\theta^{(j)})$

Scanned by CamScanner

Date:

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

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

Parameters for all users → $J(\theta^{(1)}, \ldots, \theta^{(n_u)})$

Gradient descent update :

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

for $k = 0$

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

(for $k \neq 0$)

$$\frac{\partial}{\partial \theta_k^{(j)}} J(\theta^{(j)})$$

Page No.

Victory

Scanned by CamScanner

→ The algo that will be discussed here does feature learning by itself — will learn by itself what features to use

Date:

Collaborative Filtering

Problem motivation

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

Suppose we don't know x_1, x_2 of each user

Suppose the users tell us how much they liked action + packed romance or following romance-packed movies. Hence we have the parameters:

$\theta^{(1)} = \begin{bmatrix} 0 \\ 5 \\ 0 \end{bmatrix}$, $\theta^{(2)} = \begin{bmatrix} 0 \\ 5 \\ 0 \end{bmatrix}$, $\theta^{(3)} = \begin{bmatrix} 0 \\ 5 \\ 5 \end{bmatrix}$, $\theta^{(4)} = \begin{bmatrix} 0 \\ 0 \\ 5 \end{bmatrix}$

Victory

Page No.

Scanned by CamScanner

Date:

Alice & Bob tell us they like romantic movies. Carol & Dave tell us they like action movies. Thus we set the values of $\theta^{(1)}, \theta^{(2)} \dots \theta^{(4)}$ as such.

Now it becomes possible to infer values x_1 & x_2

What should $x^{(1)}$ be that:

$(\theta^{(1)})^T x^{(1)} \approx 5$

$(\theta^{(2)})^T x^{(1)} \approx 5$

$(\theta^{(3)})^T x^{(1)} \approx 0$

$(\theta^{(4)})^T x^{(1)} \approx 0$

note $x_0 = 1$ always

Then we can reasonably assume that:

$$x^{(1)} = \begin{bmatrix} 1 \\ 1.0 \\ 0 \end{bmatrix}$$

Optimisation Algorithm

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

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

Victory

Scanned by CamScanner

Date:

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

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

$$+ \frac{\lambda}{2} \sum_{i=1}^{n_m} \sum_{k=1}^{n} \left(x_k^{(i)} \right)^2$$

content based recommendations

Collaborative filtering

(A)

Given $x^{(1)}, \ldots, x^{(n_m)}$ & movie ratings
can estimate $\theta^{(1)}, \ldots, \theta^{(n_u)}$

Given $\theta^{(1)}, \ldots, \theta^{(n_u)}$ & movie ratings, (B)
can estimate $x^{(1)}, \ldots, x^{(n_m)}$

This creates a chicken & egg problem — which comes first?

Do this:

~~Randomly guess $\theta \rightarrow x \rightarrow \theta \rightarrow x \rightarrow \theta \rightarrow x \rightarrow \ldots$~~

Randomly guess θ then:

$\theta \rightarrow x \rightarrow \theta \rightarrow x \rightarrow \theta \rightarrow x \rightarrow \ldots$

do (B) and then (A) & then (B) iteratively

Scanned by CamScanner

Date:

This will cause algo to converge to reasonable values of θ & x

As $x \to \theta$ & then $\theta \to x$ the values of x & θ will iteratively get better

Here you are actually simultaneously learning the features & the parameters

This iterative process is called collaborative filtering

The reason why this iterative process works is because each user rates multiple movies & each movie is rated by multiple users

The reason why this is called collaborative filtering is because when you ran this with a ~~big~~ large set of users, these users are collaborating to get better movie predictions for everyone because each user rates some subset of movies

↳ ~~every user is helping the algo a little bit~~

Collaboration - every user is helping the system learn better features for common good

Scanned by CamScanner

→ going back & forth method between θ & x

Date:

Collaborative Filtering Algorithm

Given $x^{(1)}, \dots, x^{(n_m)}$ estimate $\theta^{(1)}, \dots, \theta^{(n_u)}$:

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

(F) (A)

Given $\theta^{(1)}, \dots, \theta^{(n_u)}$ estimate $x^{(1)}, \dots, x^{(n_m)}$:

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

(E) (B)

computationally cheaper

Solve for x & θ simultaneously

Minimizing $x^{(1)}, \dots, x^{(n_m)}$ & $\theta^{(1)}, \dots, \theta^{(n_u)}$ simultaneously:

$$J(x^{(1)}, \dots x^{(n_m)}, \theta^{(1)}, \dots, \theta^{(n_u)}) =$$

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

(D) (C)

$$\min_{x^{(1)},\dots,x^{(n_m)}, \theta^{(1)},\dots,\theta^{(n_u)}} J(x^{(1)}, \dots, x^{(n_m)}, \theta^{(1)}, \dots, \theta^{(n_u)})$$

(C)

Scanned by CamScanner

Date:

(C) created when both objectives (A) & (B) are put together

(A) - sum over all users j & sum over all movies i rated by user
↳ summing over all pairs i, j that correspond to a that was rated by a user

(B) - for every movie i, sum over all the users j that have rated it

$(i,j) : r(i,j) = 1$

↳ iterate over all possible pairs of (i,j) such that $r(i,j) = 1$

(D) is the same as (F) & (E) because it computes the same thing but in no specific order

(C) has an interesting property:
if you hold all the x's constant & then minimize w.r.t theta then you will be solving (A) ~~& also for~~
This is also true for (B)

Scanned by CamScanner

Date:

Convention for collaborative filtering algorithm:

$x_0 = 1$ is not needed & hence:
$x^{(i)} \in \mathbb{R}^n$ (not $x^{(i)} \in \mathbb{R}^{n+1}$)
similarly θ_0 is not needed either, hence:
$\theta^{(j)} \in \mathbb{R}^n$

Reason for convention:
we're learning all the features - no need to hardcode x_0, the algorithm may choose to learn the value 1 for a feature by itself if needed

Summary:

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

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

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

special case where $k=0$ does not exist as (where there is no regularization) $x_0 = 1$ does not exist

$$\frac{\partial}{\partial x_k^{(i)}} J(\ldots)$$

Scanned by CamScanner

Date:

~~In the~~

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

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

Low Rank Matrix Factorization

↳ will talk about vectorised form of collaborative algorithm

Movie	Alice (1)	Bob (2)	Carol (3)	Dave (4)
Love at last	5	5	0	0
:	5	?	?	0
:	?	4	0	?
	0	0	5	4
	0	0	5	?

Create a matrix from the table: Y

Page No.

Victory

Scanned by CamScanner

$n_m = 3$

$n_u = 4$

Date:

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

$(n_m \times n_u)$

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

↳ rating user j will give to movie i

Suppose :

$$X = \begin{bmatrix} -(x^{(1)})^T- \\ -(x^{(2)})^T- \\ \vdots \\ -(x^{(n_m)})^T- \end{bmatrix}$$

low rank matrix hence the entire collaborative algo is called Low rank matrix factorization

$$\theta = \begin{bmatrix} -(\theta^{(1)})^T- \\ -(\theta^{(2)})^T- \\ \vdots \\ -\theta^{(n_u)T}- \end{bmatrix}$$

All predicted ratings

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

Scanned by CamScanner

Date:

You can run the collaborative learning algo with $X\Theta^T$

~~How~~

Finding related movies

For each product i, we learn a feature vector $x^{(i)} \in \mathbb{R}^n$ such as:

x_1 = romance, x_2 = action, x_3 = comedy, x_4 = ...

Once n features are learned, it is difficult to tell what these features are → what do they actually represent

But it will learn features that are meaningful for capturing whatever are the most important or most salient ~~feature~~ features properties of a movie.

How to find movies j related to movie i?

small $\|x^{(i)} - x^{(j)}\|$ → movie j & i are "similar"

can only be found when collaborative learning algo has run

Scanned by CamScanner

Date:

5 most similar movies to movie i:

Find the 5 movies j with the smallest $\|x^{(i)} - x^{(j)}\|$

Scanned by CamScanner

Week 9 continued... Date:

Implementation Detail:

Mean Normalization

↳ can make the algorithm work a little better

Movie	Alice (1)	Bob (2)	Carol (3)	Dave (4)	Eve (5)
1	5	5	0	0	? (0)
2	5	?	?	0	? (0)
3	?	4	?	?	? (0)
4	0	0	5	4	? (0)
5	0	?	5	?	? (0)

Suppose Eve has not rated any movies (0's)

Suppose $n = 2$

$\theta^{(5)} \in \mathbb{R}^2$

Page No.

Victory

Scanned by CamScanner

this part plays no role as there are no movies for Eve where ~~$r(i,j)=1$~~ $r(i,j)=1$
↳ This becomes 0

Date:

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

$$\frac{\lambda}{2}\sum_{i=1}^{n_m}\sum_{k=1}^{n}(x_k^{(i)})^2 + \frac{\lambda}{2}\sum_{j=1}^{n_u}\sum_{k=1}^{n}(\theta_k^{(j)})^2 \quad (B)$$

~~Last~~ becomes:

$$\frac{\lambda}{2}\left[(\theta_1^{(5)})^2 + (\theta_2^{(5)})^2\right]$$

We want to minimize θ so that the regularization component (B) is as small as possible → the smallest possible value is 0
Usually it is the data in (A) that pulls away the equation from becoming 0 but (A) is 0

Because the goal is to minimize everything then you end up with: $\theta^{(5)} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$

Hence: $(\theta^{(5)})^T x^{(i)} = 0$ for all i

inner product

Scanned by CamScanner

Date:

Eve will rate every movie as 0 then

It's not useful to predict all movies as 0 because some people do like some movies & if all ratings are 0 then we can't recommend a movie as well

Mean normalization fixes this problem with the matrix Y

$$Y = \begin{bmatrix} 5 & 5 & 0 & 0 & ? \\ 5 & ? & ? & 0 & ? \\ ? & 4 & 0 & ? & ? \\ 0 & 0 & 5 & 4 & ? \\ 0 & 0 & 5 & 0 & ? \end{bmatrix} \rightarrow \mu = \begin{bmatrix} 2.5 \\ 2.5 \\ 2 \\ 2.25 \\ 1.25 \end{bmatrix}$$

obtain average rating of each movie in μ

Then subtract each column of Y with μ:

$$Y_{norm} = \begin{bmatrix} 2.5 & 2.5 & -2.5 & -2.5 & ? \\ 2.5 & ? & ? & -2.5 & ? \\ ? & 2 & -2 & ? & ? \\ -2.25 & -2.25 & 2.75 & 1.75 & ? \\ -1.25 & -1.25 & 3.75 & -1.25 & ? \end{bmatrix}$$

→ normalised each movie to have an average rating of 0

→ learn $\theta^{(j)}, x^{(i)}$

Scanned by CamScanner

Date:

"question marks" will remain "question marks"

For row 3 of Y:

$$\frac{4+0}{2} = 2$$

Use Y_{norm} in collaborative filtering algo & learn $\theta^{(j)}$, $x^{(i)}$

Now for Y_{norm} the rating is predicted as such:

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

In the case of Y_{norm} $\theta^{(5)}$ will still be $\begin{bmatrix} 0 \\ 0 \end{bmatrix}$

Hence:

$$\underbrace{(\theta^{(5)})^T (x^{(i)})}_{0} + \mu_i = \mu_i$$

What this does is that if we have a new user (such as Eve) then we predict the average rating for that movie

↳ predict on the basis of popularity

Scanned by CamScanner

Date:

In this case the rows of Y were normalized to have mean 0. In case where some movie exists with no ratings then it is analgous to a user who hasn't rated rating

↳ You can use a different version of this algorithm where columns are norm to have mean 0

Anomaly Detection

Problem Motivation

↳ mainly for unsupervised problems

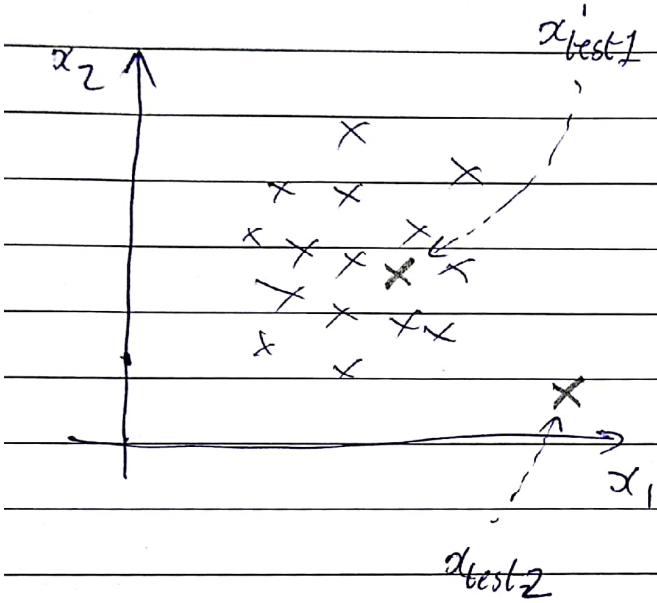
Assume that you're an aircraft engine manufacturer & you do quality assurance testing & you measure various for that :

x_1 = heat generated
x_2 = vibration intensity
⋮
x_m = ...

Scanned by CamScanner

Date:

We want to find out that is the engine rolling off the assembly line anomalous in any way i.e. should it undergo further testing? If it is an anomaly then it should be sent to further testing before shipping it to a customer



x_{test1} is not anomalous

Suppose you have another engine rolling off the assembly belt: x_{test1} & x_{test2}

x_{test2} looks very different from all of the aircraft engines seen before & hence is an anomaly

Density estimation

Suppose data set: $\{x^{(1)}, x^{(2)}, \ldots, x^{(m)}\}$

assume that this unlabeled dataset is normal (no data points are anomalous)

Scanned by CamScanner

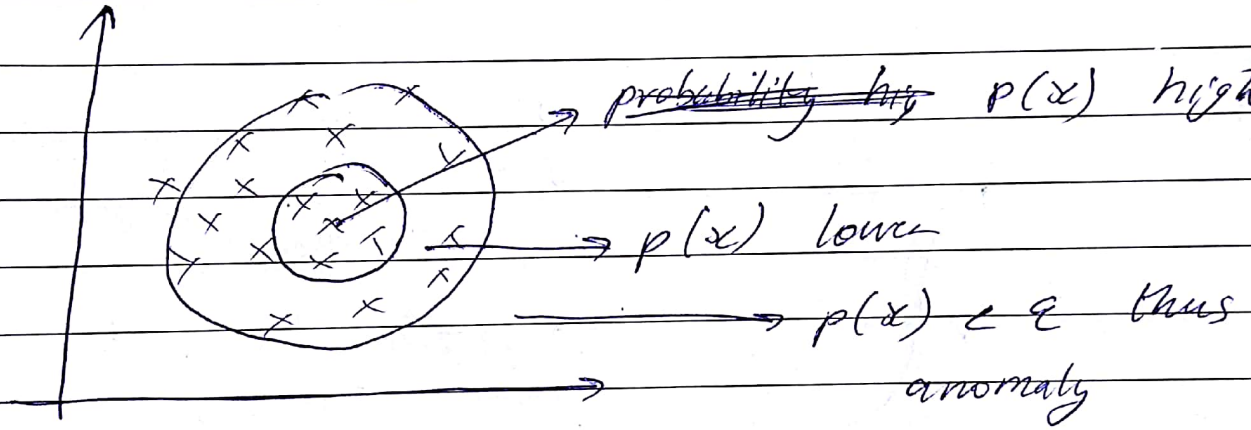
Date:

Build a model $p(x)$ from dataset $\{x^{(1)}, x^{(2)}, \ldots, x^{(m)}\}$ as training data that outputs the probability that x is normal (not anomalous)

You can flag anomalies as such:

$p(x_{test}) < \varepsilon \rightarrow$ flag anomaly

$p(x_{test}) \geq \varepsilon \rightarrow$ okay (not anomaly)



Anomaly detection example

Fraud detection:

$x^{(i)}$ = features of user i's behaviour

model $p(x)$ from data

Identify unusual users by checking which have $p(x) < \varepsilon$

Scanned by CamScanner

Date:

Gaussian Distribution

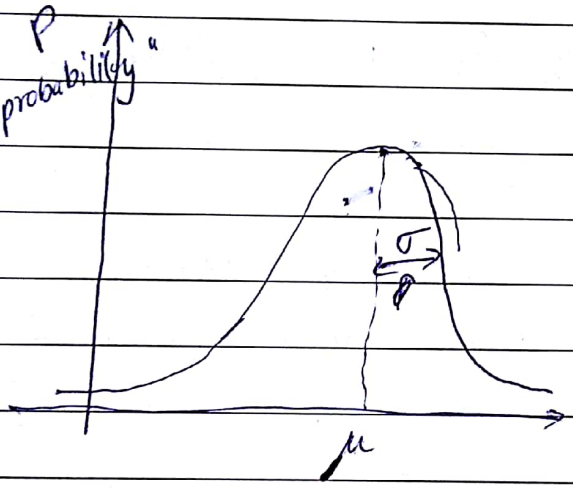
↳ also called the normal distribution

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

$$X \sim N(\mu, \sigma^2)$$

"N" stands for normal

read "distributed as"



σ^2 – variance

σ – standard deviation

↳ specifies the width of the curve

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

assume these are constant

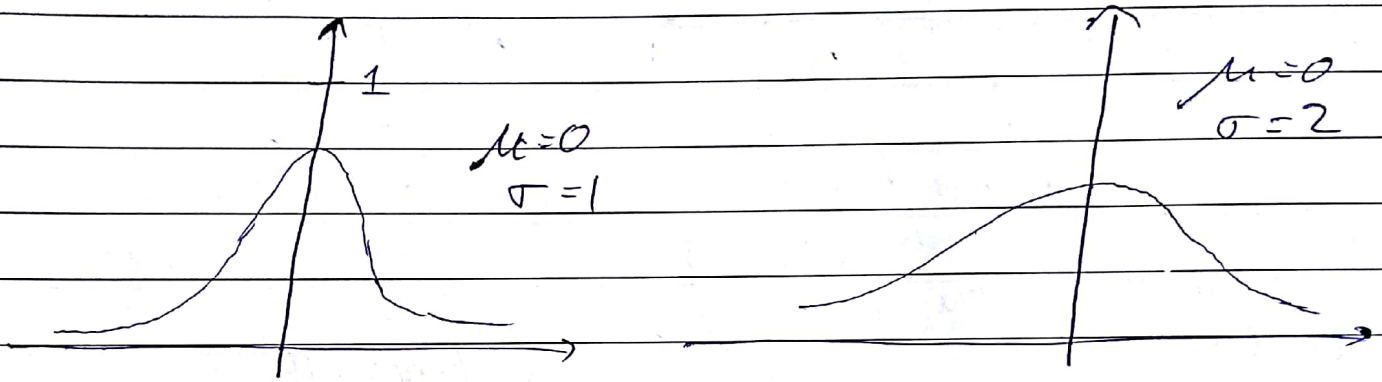
μ – determines where the curve is centered at

Scanned by CamScanner

Date:

σ - determines the rate of drop of curve : bigger drop with smaller σ

The area of the curve must always equal 1. If σ is small then it tends to be taller to compensate for the area required



$\sigma \downarrow$: thinner & longer curve

$\sigma \uparrow$: thicker & shorter curve

Parameter estimation

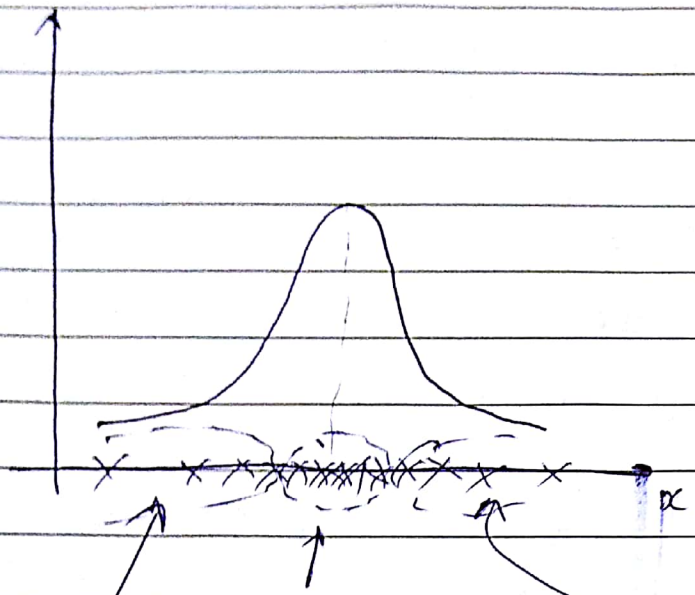
Dataset : $\{x^{(1)}, x^{(2)}, \ldots, x^{(m)}\}$ $x^{(i)} \in \mathbb{R}$

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

Assume that each of examples x are distributed according Gaussian distribution but you don't know what the values of μ & σ^2 are

Scanned by CamScanner

Date:



Given the dataset I want to estimate the values of μ & σ

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

average of all of the values

probability drops off to the sides

x has highest probability in the middle

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

at times it is $\frac{1}{m-1}$ but makes very little difference when m is large

Algorithm

Density estimation

Training set: $\{x^{(1)}, \ldots, x^{(m)}\}$

Each example is $x \in \mathbb{R}^n$

Scanned by CamScanner

Date:

Assume all features are distributed with Gaussian distribution:

$$x_1 \sim N(\mu_1, \sigma_1^2)$$
$$x_2 \sim N(\mu_2, \sigma_2^2)$$
$$\vdots$$
$$x_n \sim N(\mu_n, \sigma_n^2)$$

$$p(x) = p(x_1; \mu_1, \sigma_1^2)\, p(x_2; \mu_2, \sigma_2^2) \dots p(x_n; \mu_n, \sigma_n^2)$$

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

↓

product of a set of values

independence of features is assumed
↳ eqn works out fine if features are not actually independent

$$\sum_{i=1}^{n} i = 1+2+3+\dots+n$$

$$\prod_{i=1}^{n} i = 1 \times 2 \times 3 * \dots * n$$

Finding out $p(x)$ is called the problem of probability distribution (hence title)

Scanned by CamScanner

Date:

Anomaly Detection Algorithm

1) Choose features x_j that you think might be indicative of anomalous examples

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

$$\mu_j = \frac{1}{m} \sum_{i=1}^{m} x_j^{(i)} \qquad \mu = \begin{bmatrix} \mu_1 \\ \mu_2 \\ \vdots \\ \mu_n \end{bmatrix}$$

Vectorized:

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

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

Vectorized:

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

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

$$p(x) = \prod_{j=1}^{n} p(x_j; \mu_j, \sigma_j^2) =$$

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

Anomaly if $p(x) < \varepsilon$

Scanned by CamScanner

Date:

Developing & Evaluating an Anomaly Detection System

The importance of real number evaluation

When developing a learning algo (choosing features, etc), making decisions is much easier if we have a way of evaluating our learning algo that gives you back a single number

↳ example: Does algo improve or worsen if another feature is added

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

↳ assume unlabeled

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: $\{(x_{test}^{(1)}, y_{test}^{(1)}), \ldots, (x_{test}^{(m)}, y_{test}^{(m)})\}$

Both CV & test set has anomalous data
↳ $y = 1$

Scanned by CamScanner

Date:

Aircraft engines motivating example:
- 10000 good (normal) engines
- 20 flawed engines (anomalous)

Training set: 6000 good engines ($y=0$)
CV: 2000 good engines ($y=0$), 10 anomalous ($y=1$)
Test: 2000 good engines ($y=0$), 10 anomalous ($y=1$)

60% of $y=0$ into training
20% of $y=0$ into CV
20% of $y=0$ into test

Algorithm evaluation
Fit model $p(x)$ on training set $\{x^{(1)}, \ldots, x^{(m)}\}$
On a CV/test examples x, predict:

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

Similar to supervised learning setting i.e same evaluation metrics can be used as in supervised learning

Possible evaluation metrics:
- true positives, false positives, false negatives, true negative
- precision/recall
- F_1-score

because data highly skewed, accuracy will be a bad measure of evaluation metric

Scanned by CamScanner

Date:

Once all the features, value of ε are decided then check evaluation metrics of test set

Can also use CV set to choose parameter ε

↳ Try many ~~different values of epsilon &~~ then pick epsilon ~~that maximises~~ f-1

Once all the features, value of ε are decided then check evaluation metrics of test set.

Anomaly Detection vs Supervised Learning

↳ If all of the anomalies are labeled then why isn't a supervised learning algo used instead of anomaly detection algo

Scanned by CamScanner

Such small amount of +ve examples that its best to save it for CV & test set & train with negative examples

Date:

Anomaly detection	vs	Supervised Learning
→ very small # of +ve examples ($y=1$)		→ large # of +ve & -ve examples
→ large # of negative examples		
→ Many different "types" of anomalies. Hard for any algorithm to learn from +ve examples what anomalies look like; future anomalies may look nothing like any of the anomalies seen so far		→ Enough +ve examples for algo to get sense of what +ve examples are like, future +ve examples likely to be similar to ones in training set
→ difficult to tell what anomalies may look like from the small set of +ve examples. Not all possible situations of what sort of anomalies may be present are there hence these anomalies can't be learned via supervised learning		→ example: spam where different classes have many examples
→ Examples: fraud detection, Manufacturing, monitoring machines in data centers		→ Examples: email spam, weather prediction, cancer classification

Victory

Page No.

Scanned by CamScanner

Date:

Choosing what features to use

Guideline: plot data or histogram of data to make sure that data looks vaguely Gaussian before feeding it to anomaly detection algo

↳ will still work fine if data looks non-Gaussian

"hist" command in Octave is used to plot histograms

Guideline: if the data looks very different then play around different transformations of the data to make it look more Gaussian

↳ algo may still work without transformations but it will work slightly better with the transformations

Example transformation: $\log(x_1)$ ← feature to be transformed

$x_1 \leftarrow \log(x_1)$
$x_2 \leftarrow \log(x_2 + 1)$
$x_3 \leftarrow \sqrt{x_4}$
$x_4 \leftarrow x_5^a$

} convert data & reassign back

Scanned by CamScanner

Error analysis (as it was in supervised learning) -
look at the examples that the algo got wrong
& see if some extra features can mitigate examples being wrongly classified

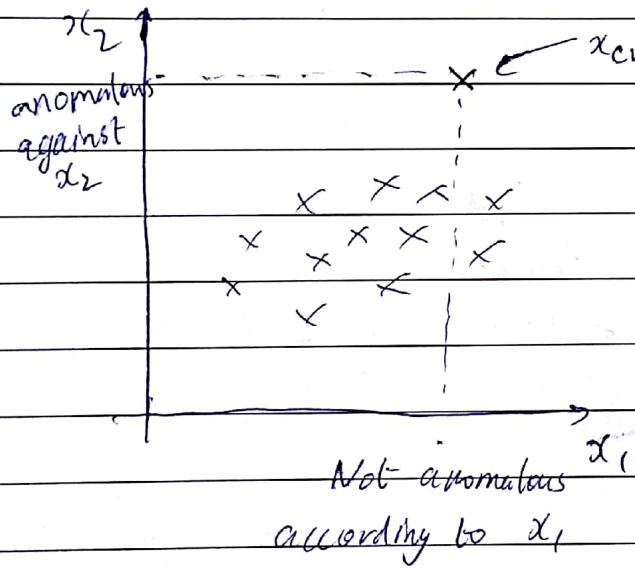
Date:

Error analysis for anomaly detection

Want $p(x)$ large for normal examples x
$p(x)$ small for anomalous examples x

Most common problem:
$p(x)$ is comparable (say, both large) for normal & anomalous examples

↳ come up with some new feature that can flag the ~~examples as anomalous~~



If the feature x_2 is present then only ~~anomalies~~ can the algo catch ~~is~~ anomaly

Choose such features in anomaly detection that might take on unusually large or small values in the event of an anomaly

You can add features as well:

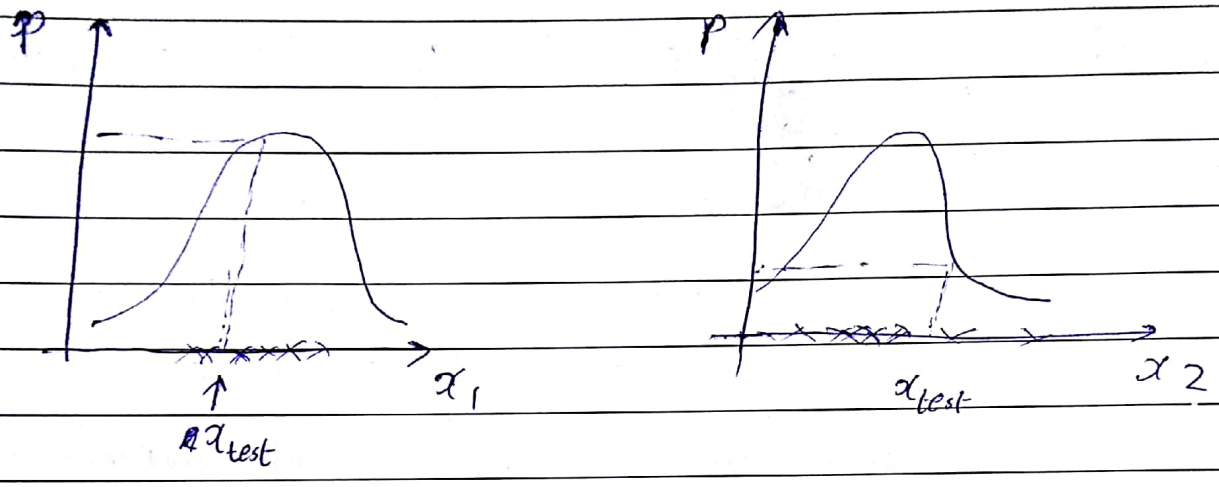
$$x_5 = \frac{(x_1)^2}{x_2}$$

Scanned by CamScanner

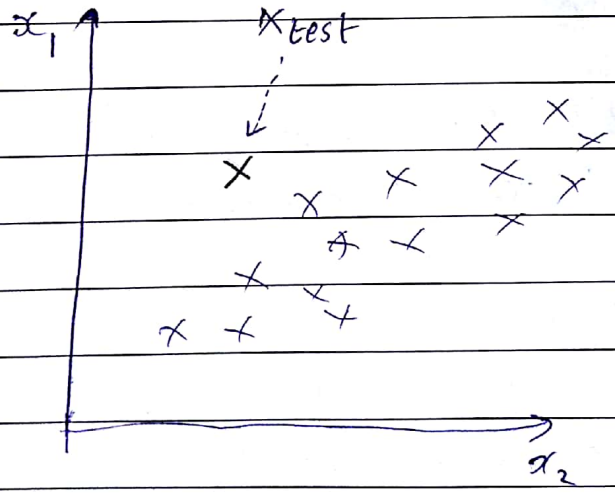
can sometimes catch some anomalies the earlier algorithm didn't

Date:

Multivariate Gaussian Distribution



Suppose that x_{test} was supposed to be an anomaly but looking at both x_1 & x_2 it isn't



To fix this use multivariate gaussian distribution

Scanned by CamScanner

Date:

$x \in \mathbb{R}^n$, Don't model $p(x_1), p(x_2), \ldots$ etc seperately. Model $p(x)$ all in one go

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

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

↳ determinant of Σ calculated in octave. as such: "det(sigma)"

Highest probability will be at μ (peak of distribution)

Probability distribution of the multivariate gaussian distribution integrates to 1

if $\Sigma = 0.6\,I$ then graph is thinner & taller

↑ identity matrix

if $\Sigma = 2I$ then graph is fatter & shorter

$$0.6\,I = \begin{bmatrix} 0.6 & 0 \\ 0 & 0.6 \end{bmatrix} \qquad 2I = \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix}$$

for $n = 2$

Scanned by CamScanner

off-diagonal non-zero

If ~~both non zero~~ values are ~~different~~ then shape becomes skewed eliptical as opposed to being ~~circular~~ ∧ non-skewed eliptical when ~~both are~~ off diagonals are 0 ~~equal~~

MGD (multivariate Gaussian Distribution) can be used to model correlations in data by changing the non-diagonal values

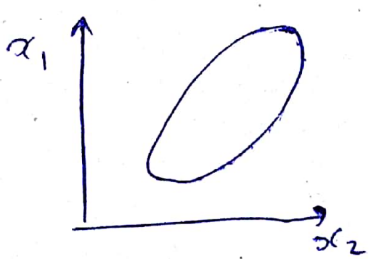
Anomaly Detection using MGD

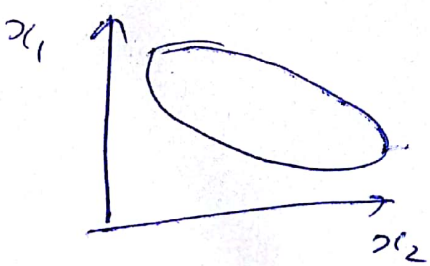
Parameter fitting:

Given training set $\{x^{(1)}, x^{(2)}, \ldots, x^{(m)}\}$

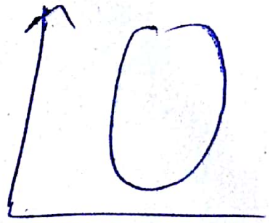
$$\mu = \frac{1}{m}\sum_{i=1}^{m} x^{(i)}, \quad \Sigma = \frac{1}{m}\sum_{i=1}^{m} (x^{(i)}-\mu)(x^{(i)}-\mu)^T$$

~~no~~ skewed eliptical shapes:





~~no~~ non-skewed:



Scanned by CamScanner

Sequence of steps:

1) Fit model $p(x)$ by setting

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

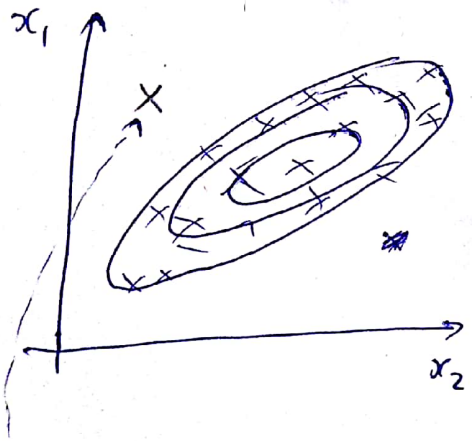
$$\Sigma = \frac{1}{m}\sum_{i=1}^{m} (x^{(i)} - \mu)(x^{(i)} - \mu)^T$$

2) Given a new example x, compute:

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

3) Flag an anomaly if $p(x) < \epsilon$

With this new distribution we can now correctly model the probability of an example being normal:



will correctly classify this as an anomaly

This was not possible with single-variate gussian ~~distri~~ distribution (SGD)

↳ The assumption that x_1 & x_2 are independent breaks down here.

Scanned by CamScanner

Relationship between SGD & MGD

Date:

Original model (SGD):

$$p(x) = p(x_1; \mu_1, \sigma_1^2)\, p(x_2; \mu_2, \sigma_2^2) \dots p(x_n; \mu_n, \sigma_n^2)$$

Corresponds to multivariate Gaussian

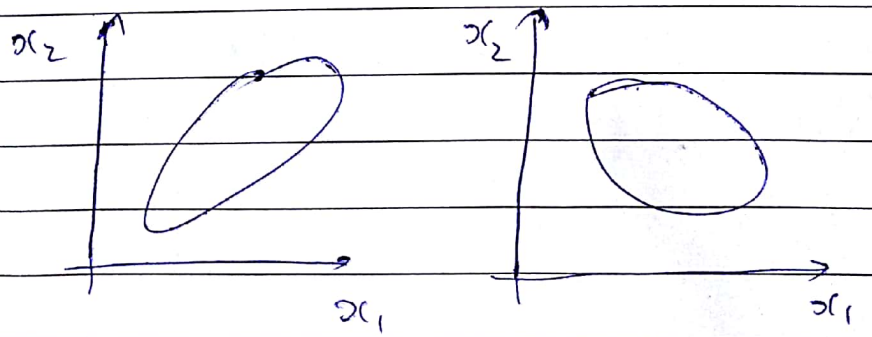
$$p(x; \mu, \Sigma) = \frac{1}{(2\pi)^{n/2} |\Sigma|^{\frac{1}{2}}} \exp\left(-\frac{1}{2}(x-\mu)^T \Sigma^{-1} (x-\mu)\right)$$

where $\Sigma = \begin{bmatrix} \sigma_1^2 & & 0 \\ & \sigma_2^2 & \\ & \ddots & \\ 0 & & \sigma_n^2 \end{bmatrix}$

SGD is a special version of MGD

where off diagonal values are kept at 0, contour axis are aligned

Hence skewed elipses are only possible in MGD, not in SGD:



x_1 & x_2 are dependent here: goes against the assumption of independence in SGD

Scanned by CamScanner

Date:

SGD

Manually create features to capture anomalies where x_1, x_2 take unusual combinations of values

↳ such as $x_3 = \frac{x_1}{x_2}$

this captures correlations between x_1 & x_2

Computationally cheaper (Alt. scales better to large n)

Ok even if m (training set size) is small

MGD

Automatically captures correlations between features

Computationally more expensive because of Σ^{-1} $(n \times n)$

The bigger Σ i.e greater value of n, more expensive to invert

→ No need to do this for MGD

Must have $m > n$ or else Σ is non-invertible

Σ^{-1} can be singular for 2 reasons:

1) $m > n$
2) redundant features present as such:

$x_1 = x_2$ or

$x_3 = x_4 + x_5$ & etc.

Scanned by CamScanner