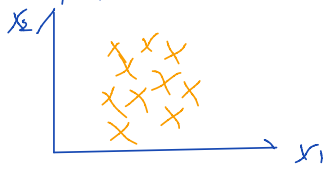
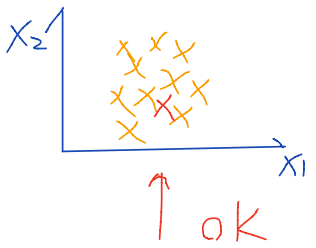
Anomaly Detection — Problem motivation

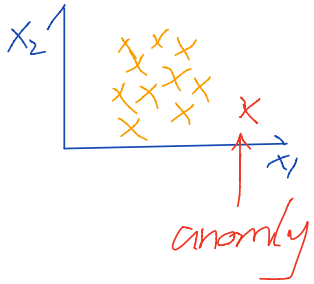
- An Example

Features $\{x_1^{(1)}, x_2^{(1)}\}, \cdots\cdots, \{x_1^{(m)}, x_2^{(m)}\}$ has distribution:



If now having a new data set, it could be of following cases:





- Density Estimation

Dataset: $\{x^{(1)}, x^{(2)}, \cdots\cdots x^{(m)}\}$

Is x_{test} anomalous?

Model: density $p(x)$ $\begin{cases} p(x_{test}) < \varepsilon & \rightarrow \text{anomaly} \\ p(x_{test}) \geq \varepsilon & \rightarrow \text{ok} \end{cases}$

Example:

— Fraud detection

- $x^{(i)}$ features of user i's activities
- Model $p(x)$ from data

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

— Manufacturing

— Monitoring computers in a data center

Anomaly Detection — Gaussian distribution

Anomaly Detection — Algorithm

◦ Density Estimation

Training set $\{x^{(1)}, \cdots x^{(m)}\}$ $\quad x \in \mathbb{R}^n$

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

Assume each feature follows Gaussian distribution, and they are independent to each other. (In computation, even if they are not independent, it's fine ?? — Andrew said so, I'm not sold)

$$p(x) = P(x_1; \mu_1, \sigma_1^2) \cdots P(x_n; \mu_n, \sigma_n^2)$$
$$= \prod_{j=1}^{n} P(x_j; \mu_j, \sigma_j^2)$$

Algorithm:

1. Choose features x_i that you think might be indicative of anomalous examples
2. Fit parameters $\mu_1, \cdots \mu_n, \sigma_1^2, \cdots \sigma_n^2$

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

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

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

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

anomaly if $p(x) < \varepsilon$

Building an anomaly detection system

- Develop anomaly system

The importance of real-number evaluation

When developing a learning algorithm, making decisions is much easier is we have a way of evaluating our algorithm

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

Training set: $x^{(1)}, x^{(2)}, \cdots, x^{(m)}$, assume they are all normal examples (it's ok if a few anomalous examples are included.)

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

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

are labeled data sets (meaning we know exactly if they are anomalous or non-anomalous

Aircraft engines motivating example:

10000 good (normal) engines, 20 flawed engines (anomalous)

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

Algorithm:

1) Fit model $p(x)$ on training set $\{x^{(1)}, \ldots, x^{(m)}\}$
2) On a cv/test set example x, predict

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

Possible evaluation metrics:

- True positive, false positive, false negative
  true negative
- Precision / Recall
- F_1-Score

Can also use CV set to choose ε value

Note CV/test set is very skewed!! So classification accuracy is not a good way to evaluate the algorithm.

- Anomaly detection VS supervised learning

Anomaly detection	Supervised learning
• very small num of positive examples (y=1)	• Large number of positive and negative examples
• large num of negative examples (y=0)	• Enough positive examples for algorithm to get sence of what positive examples are look like
• Many different "types" of anomaly examples hard for any algorithm to learn from it	• Future positive examples likely to be similar to ones in training set
• future anomalies may look nothing like any of the anomalous examples seen before	

Application

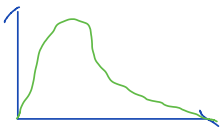
Anomaly detection	Supervised learning
• Fraud dection	• Email spam classification
• manufacturing	• Weather prediction
• Monitoring machines in data center	• cancer classification
⋮	⋮

If have large number of positive examples (y=1), anomaly detection can be transfered to supervised learning

o choosing what features to use

non-gaussian features transform

- Plot the feature histogram to exam if it's Gaussian
- If its not a good gaussian



do some transformation to the feature

to make it more gaussian, e.g. $\log(x+c)$, $\sqrt{x}$, x^t, ...

Error analysis for anomaly detection

- want: $p(x)$ large for normal ($y=0$) examples x

  $p(x)$ small for anomalous ($y=1$) examples x.

- most common problem:

  $p(x)$ is comparable for normal/anomalous examples

- Solution

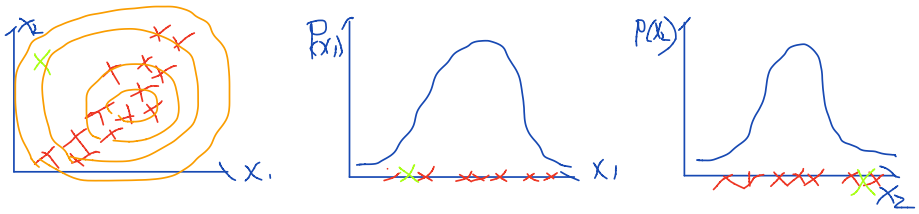
  Inspect mistakes and try to create new features

- experience

  · choose features that might take on unusually large or small values in an event of an anomaly

Multivariate Gaussian Distribution

- When anomaly detection fails:



Fail to detect x as positive

Solution:

$x \in \mathbb{R}^n$, don't model $p(x_1), \cdots, p(x_n)$ seperately

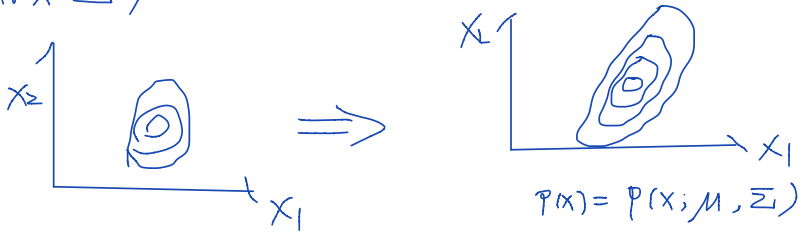
Model $p(x)$ all in one go

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

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

$|\Sigma| = \det(\Sigma)$

By doing this, actually we're allowed to "rotate" the Gaussian distribution (due to off-diagnal value of covariance matrix Σ)



$p(x) = p(x_1)p(x_2)\cdots p(x_n)$

$p(x) = p(x; \mu, \Sigma)$

- Algorithm

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

Recommender Systems

- Problem formulation

Example: predicting movie ratings

MOVIE	USER 1	USER 2	USER 3	USER 4	.....
movie 1	5	?	3	4	
movie 2	1	2	?	4	
⋮	..	..	..	..	

n_u = no. users n_m = no. 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 when $r(i,j)=1$)

Question: how to predict movie ratings that hasn't been rated by a certain user?

- Content based recommendations

$n_u = 4$, $n_m = 5$

x_1 (romance) x_2 (action)

MOVIE	USER 1	USER 2	USER 3	USER 4		
movie 1	5	?	3	4	0.9	0
movie 2	1	2	?	4	1.0	0.01
⋮	⋮	⋮	⋮	⋮	⋮	⋮

$x_0 = 1 \quad x^{(1)} = \begin{bmatrix} 1 \\ 0.9 \\ 0 \end{bmatrix} \quad x^{(2)} = \begin{bmatrix} 1 \\ 1.0 \\ 0.01 \end{bmatrix} \cdots\cdots$

num of features : 3 $x_0 = 1$ x_1, x_2

For each user j, learn a parameter $\theta^{(j)} \in \mathbb{R}^3$. Predict user as rating movie j with $(\theta^{(j)})^T x^{(i)}$ stars

Problem formulation:

- $r(i,j) = 1$ if user j has rated movie i, else 0
- $y^{(i,j)}$ = 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 $(\theta^{(j)})^T x^{(i)}$

To learn $\theta^{(j)}$, $m^{(j)}$ = no. of movies rated by user j

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

just like linear regression, use gradient decent to find θ

Collaborative Filtering (feature learning)

MOVIE	USER 1	USER 2	USER 3	USER 4	x_1	x_2
movie 1	5	?	3	4	?	?
movie 2	1	2	?	4	?	?
⋮	⋮	⋮	⋮	⋮	⋮	⋮
					?	?

If ~~the~~ users can ~~tell~~ us how much ~~they~~ like each kind of movies (e.g. $\theta^{(1)} = \begin{bmatrix} 0 \\ 5 \\ 0 \end{bmatrix}$, $\theta^{(2)} = \begin{bmatrix} 0 \\ 0 \\ 5 \end{bmatrix}$ ...)
~~then~~ we can learn ~~the~~ features for each rated movies

The problem turned out ~~to~~ be; what feature value $x_1^{(i)}$ for movie i should be, so that $(\theta^{(1)})^T x^{(i)} \approx 5$, $(\theta^{(2)})^T x^{(i)} \approx 0$, ...

Optimization algorithm

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

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

Collaborative filtering

- Given $x^{(1)}, \dots, x^{(n_m)}$ (and movie ratings)
  can estimate $\theta^{(1)}, \dots, \theta^{(n_u)}$
- Given $\theta^{(1)}, \dots, \theta^{(n_u)}$

can estimate $x^{(1)}, \dots, x^{(n_m)}$

$\Rightarrow$ Guess $\theta \to x \to \theta \to x \to \theta \to x \to \theta \to \cdots$

This is awesome !!!

- Collaborative filtering algorithm

non-optimized algorithm

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

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

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

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

Guess $x^{(1)}, \dots x^{(n_m)}$ or $\theta^{(1)}, \dots \theta^{(n_u)}$, then go back and forth (i) and (ii) to optimize the learning

— Optimized algorithm

Do (i) and (ii) at the same time

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

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

no $x^0 = 1$ in this case

1) Initialize $x^{(1)}, \dots x^{(n_m)}, \theta^{(1)}, \dots \theta^{(n_u)}$ to small random values

2) Minimize $J(x^{(1)} \dots, \theta^{(1)} \dots)$ using gradient decent, e.g.

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

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

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

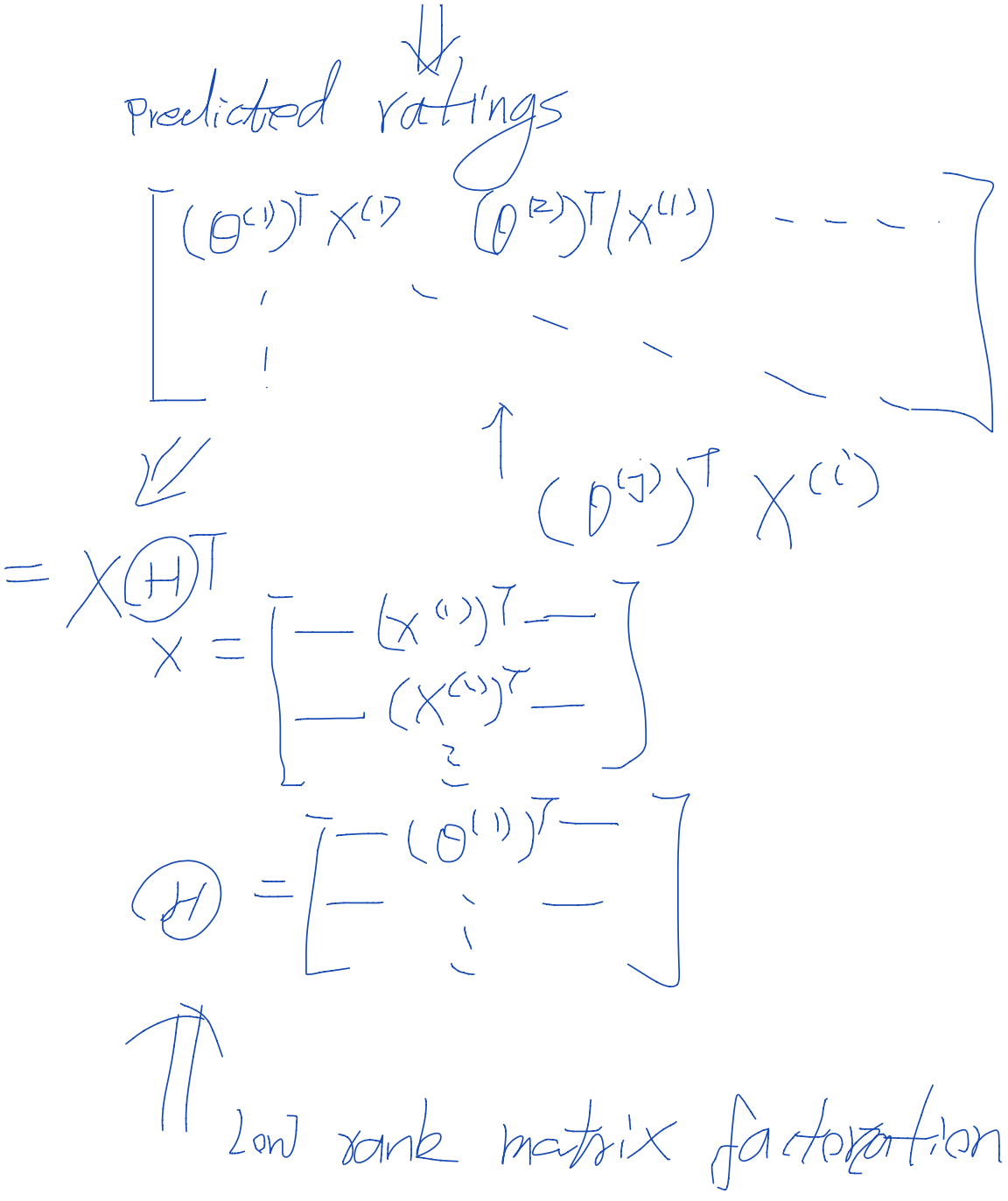
Low rank Matrix Factorization

MOVIE	USER 1	USER 2	USER 3	USER 4	.....
movie 1	5	?	3	4	
movie 2	1	2	?	4	
⋮	⋮	⋮	⋮	⋮	

$$Y = \begin{bmatrix} 5 & ? & 3 & 4 \\ 1 & 2 & ? & 4 \\ \cdots & \cdots & \cdots & \cdots \end{bmatrix}$$

$y(i,j)$

1.



- Finding related movies
  - For each product i, we learn a feature vector $x^{(i)} \in \mathbb{R}^n$
  - ? How to find movies j related to movie i ?
    Small $\|x^{(i)} - x^{(j)}\|$ → movie i, i are "similar"

Implementational detail: Mean Normalization

For users hasn't rated any movie, $\min J(\theta) \longrightarrow$ with give $\theta = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$ so predict this user will always rate movie 0

NOT RIGHT!

Solution: Mean Normalization

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

$$\Rightarrow \mu = \begin{bmatrix} 2.5 \\ 2.5 \\ \vdots \\ \end{bmatrix} \longrightarrow Y' = \begin{bmatrix} 2.5 & 2.5 & -2.5 & 2.5 & ? \\ \cdot & \cdot & \cdot & \cdot & \\ \cdot & \cdot & \cdot & \cdot & \end{bmatrix}$$

For user j on movie i, predict $(\theta^{(j)})^T(x^{(i)}) + \mu_i$

In this case the above problem can be avoided.