Week 9

- ## Density Estimation:

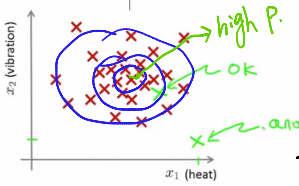
a. Problem Motivation:

Anomaly detection example

Aircraft engine features: quality assurance.
- x_1 = heat generated
- x_2 = vibration intensity
- ...

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

New engine: x_{test}



we want to know if this new aircraft engine is anomalous in any way. in other way, we want to know if this engine should undergo further testing.

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

Is x_{test} anomalous?

model $p(x)$.

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

$\geq \varepsilon$ → ok.

Anomaly detection example

- Fraud detection:
  - $x^{(i)}$ = features of user i's activities
  - Model $p(x)$ from data.
  - Identify unusual users by checking which have $p(x) < \varepsilon$

x_1, x_2, x_3, x_4 → $p(x)$

to try identify users behaving strange

- Monitoring computers in a data center.
  - $x^{(i)}$ = features of machine i
    x_1 = memory use, x_2 = number of disk accesses/sec,
    x_3 = CPU load, x_4 = CPU load/network traffic.
    ...

$p(x) < \varepsilon$

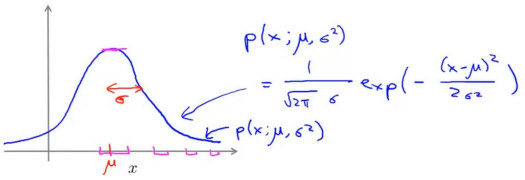
b. Gaussian Distribution:

Gaussian (Normal) distribution

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

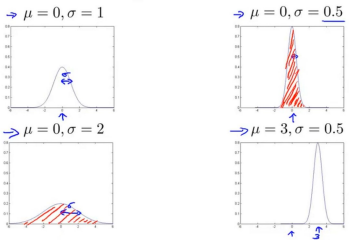
$x \sim \mathcal{N}(\mu, \sigma^2)$ ("distributed as")

σ standard deviation



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

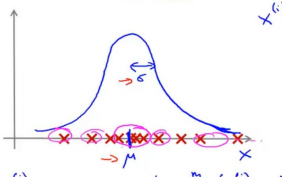
$p(x; \mu, \sigma^2)$



Parameter estimation

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

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



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

c. Algorithm:

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

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

$x_1 \sim \mathcal{N}(\mu_1, \sigma_1^2)$

$x_2 \sim \mathcal{N}(\mu_2, \sigma_2^2)$

$x_3 \sim \mathcal{N}(\mu_3, \sigma_3^2)$

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

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

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

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

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

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

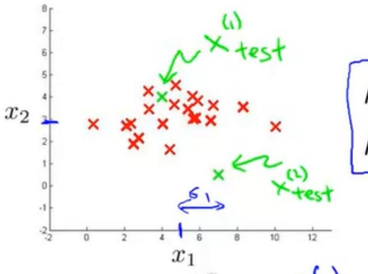
$$\mu_j = \frac{1}{m}\sum_{i=1}^{m} x_j^{(i)} \qquad \sigma_j^2 = \frac{1}{m}\sum_{i=1}^{m} (x_j^{(i)} - \mu_j)^2.$$

③ Give new example x, compute $p(x)$:

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

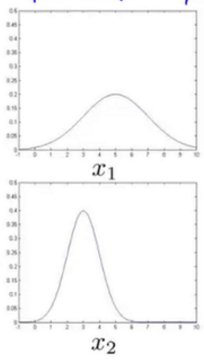
Anomaly $p(x) < \varepsilon$.

Anomaly detection example



$\mu_1 = 5, \sigma_1 = 2$

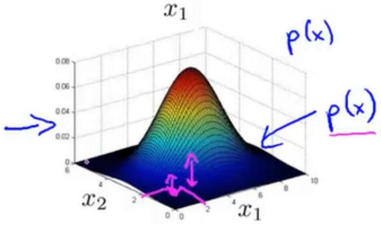
$\mu_2 = 3, \sigma_2 = 1$



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

$p(x_1; \mu_1, \sigma_1^2)$

$p(x_2; \mu_2, \sigma_2^2)$



$\varepsilon = 0.02$

$p(x_{test}^{(1)}) = 0.0426 \geq \varepsilon$

$p(x_{test}^{(2)}) = 0.0021 < \varepsilon$

- # Building Anomaly Detection System:

a. Developing and Evaluating an Anomaly Detection System:

When developing a learning algorithm (choosing features, etc.), making decisions is much easier if we have a way of evaluating our learning algorithm.

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

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

still unsupervised learning.

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

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

$y = 1$

Do I need to add one feature?
I want to run the algorithm including this feature or not. It will return two real numbers to evaluate.

Aircraft engines motivating example

→ 10000 good (normal) engines

→ 20 flawed engines (anomalous)

to fit the parameters.

2-50

$y = 1$

$\mu_1, \sigma_1^2, \ldots, \mu_n, \sigma_n^2$.

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

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

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

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

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

→ On a cross validation/test example x, predict

$(x_{test}^{(i)}, y_{test}^{(i)})$

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

$y = 0$

Possible evaluation metrics:

→ - True positive, false positive, false negative, true negative

→ - Precision/Recall

→ - F_1-score ←

Can also use cross validation set to choose parameter ε

choose ε that max the F_1 score.

b. Anomaly Detection vs. Supervised Learning:

when use an anomaly detecetion algorithm, and whether it might be more fruitful instead of using a supervisor in the algorithm.

Anomaly detection	vs.	Supervised learning
→ Very small number of positive examples ($y = 1$). (0-20 is common).		Large number of positive and negative examples.
→ Large number of negative ($y = 0$) examples. $p(x)$		
→ Many different "types" of anomalies. Hard for any algorithm to learn from positive examples what the anomalies look like;		Enough positive examples for algorithm to get a sense of what positive examples are like, future positive examples likely to be similar to ones in training set.
→ future anomalies may look nothing like any of the anomalous examples we've seen so far.		

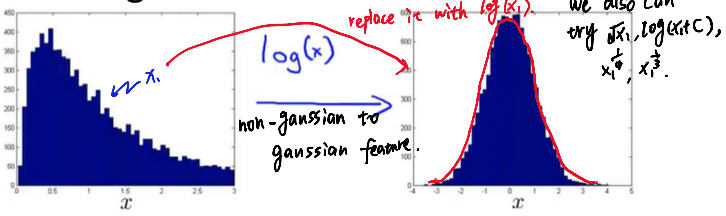
try to hard to model the positive examples because tomorrow's anomaly may be nothing like the ones you've seen so far.

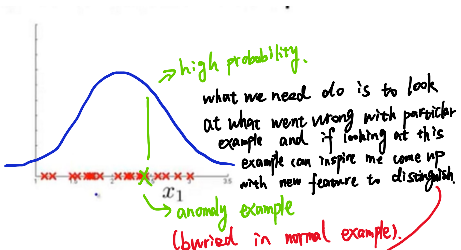
application: Frad detection (a lot of examples of $y=1$) → Email spam classification.

Manufacturing (e.g. aircraft engines) Weather prediction

Monitoring machines in a data center Cancer classification.

c. Choosing What Features to Use:



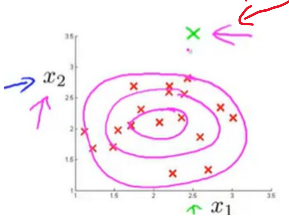


→ **Error analysis for anomaly detection**

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 and anomalous examples



→ **Monitoring computers in a data center**

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

- → x_1 = memory use of computer
- → x_2 = number of disk accesses/sec
- → x_3 = CPU load ←
- → x_4 = network traffic ←

more feature → $x_5 = \frac{\text{CPU load}}{\text{network traffic}}$ $x_6 = \frac{(\text{CPU load})^2}{\text{network traffic}}$

monitor some special case: computer gets stuck in infinite loop

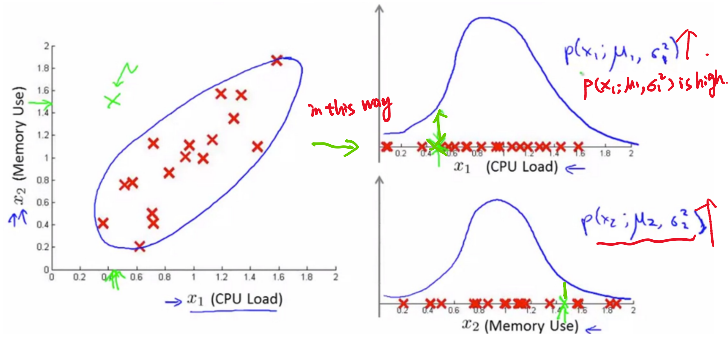
high cpu load ← but low network traffic

so x_5 will be large. ←

- # Multivatirate Gaussian Distribution:

a. Multivariate Gaussian Distribution:

$\Rightarrow x \in \mathbb{R}^n$. 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)



these two points doesn't look like anomaly in the two distributions respectively.

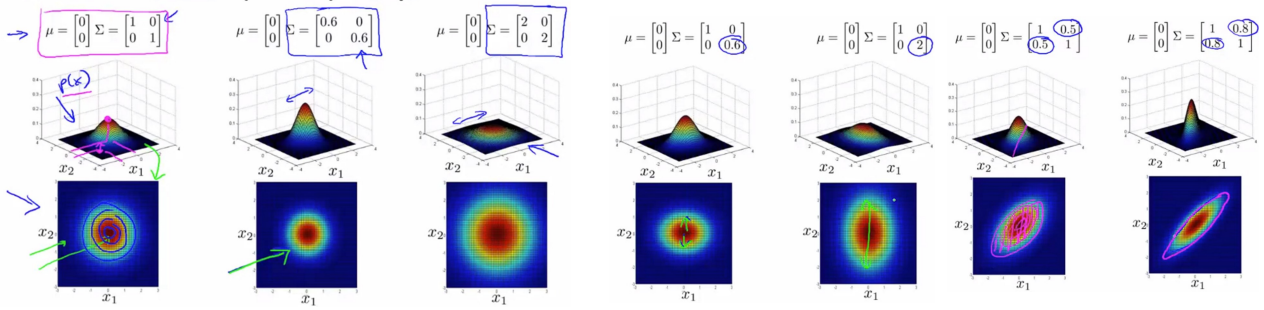
$p(x; \mu, \Sigma) = \frac{1}{(2\pi)^{n/2}|\Sigma|^{\frac{1}{2}}} \exp\left(-\frac{1}{2}(x-\mu)^T \Sigma^{-1}(x-\mu)\right)$

$|\Sigma|$ = determinant of Σ | det (Sigma)

$\mu \in \mathbb{R}^n \quad \Sigma \in \mathbb{R}^{n \times n}$

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

Multivariate Gaussian (Normal) examples

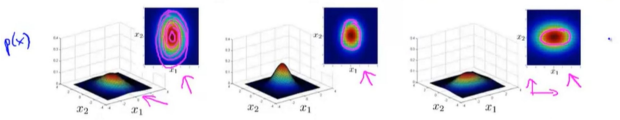


b. Anomaly Detection using the Multivariate Gaussian Distribution:

Parameter fitting:
Given training set $\{x^{(1)}, x^{(2)}, \ldots, x^{(m)}\}$ $x \in \mathbb{R}^n$

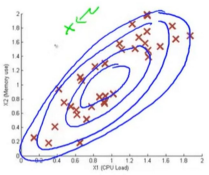
$$\mu = \frac{1}{m}\sum_{i=1}^{m} x^{(i)} \qquad \Sigma = \frac{1}{m}\sum_{i=1}^{m}(x^{(i)} - \mu)(x^{(i)} - \mu)^T$$

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



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

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

Flag an anomaly if $p(x) < \varepsilon$

For the original model, its contours of the Probability density function are axis aligned.

Corresponds to multivariate Gaussian

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

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

$\Rightarrow$ Original model	vs.	$\Rightarrow$ Multivariate Gaussian
$p(x_1; \mu_1, \sigma_1^2) \times \cdots \times p(x_n; \mu_n, \sigma_n^2)$		$p(x; \mu, \Sigma) = \frac{1}{(2\pi)^{\frac{n}{2}}\lvert\Sigma\rvert^{\frac{1}{2}}} \exp\left(-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)\right)$
Manually create features to capture anomalies where x_1, x_2 take unusual combinations of values. $\Rightarrow x_3 = \frac{x_1}{x_2} = \frac{\text{CPU load}}{\text{memory}}$		$\Rightarrow$ Automatically captures correlations between features
$\Rightarrow$ Computationally cheaper (alternatively, scales better to large n) $n = 10{,}000$, $n = 100{,}000$		$\Sigma \in \mathbb{R}^{n \times n}$ Σ^{-1} Computationally more expensive $\Rightarrow \Sigma \sim \frac{n^2}{2}$
OK even if m (training set size) is small		Must have $m > n$, or else Σ is non-invertible. $m \geq 10n$ ① $m \geq n$ ② redundant features

- # Predicting Movie Ratings:

a. Problem Formulation:

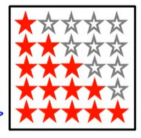
Pick a set of features automatically.

Example: Predicting movie ratings

$\Rightarrow$ User rates movies using one to five stars (zero)

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

$n_u = 4$ $n_m = 5$



$\Rightarrow n_u$ = no. users
$\Rightarrow n_m$ = no. movies
$\Rightarrow r(i,j) = 1$ if user j has rated movie i
$\Rightarrow y^{(i,j)}$ = rating given by user j to movie i (defined only if $r(i,j) = 1$) $0, \ldots, 5$

What recommender system do is to predict "?" area.

b. Content Based Recommendations:

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

$x_0 = 1$; the degree to which a movie is a romantic movie; so as; $n = 2$.

$x^{(1)} = \begin{bmatrix} 1 \\ 0.9 \\ 0 \end{bmatrix}$

We could treat predicting the ratings of each user as a separate linear regression problem.

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

$x^{(3)} = \begin{bmatrix} 1 \\ 0.99 \\ 0 \end{bmatrix}$ suppose $\theta^{(1)} = \begin{bmatrix} 0 \\ 5 \\ 0 \end{bmatrix}$ $\rightarrow (\theta^{(1)})^T x^{(3)} = 4.95$.

- $r(i,j) = 1$ 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: $(\theta^{(j)})^T(x^{(i)})$ $\quad \theta^{(j)} \in \mathbb{R}^{n+1}$
- $m^{(j)}$ = no. of movies rated by user j

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

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

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

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

Gradient descent update:

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

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

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

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

- Collaborative Filtering:

a. Collaborative Filtering:

It can start to learn for itself what features to use.

Problem motivation

Movie	Alice (1) $\theta^{(1)}$	Bob (2) $\theta^{(2)}$	Carol (3) $\theta^{(3)}$	Dave (4) $\theta^{(4)}$	x_1 (romance)	x_2 (action)
Love at last	5	5	0	0	1.0	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	?	?	?

$x^{(1)}$ denote

know these features is time consuming.

we can conclude from Alice and Bob like romantic movie and they give a high rate to "Love at last".

$(\theta^{(1)})^T x^{(1)} \approx 5 \quad (\theta^{(3)})^T x^{(1)} \approx 0$

$(\theta^{(2)})^T x^{(1)} \approx 5 \quad (\theta^{(4)})^T x^{(1)} \approx 0$

$\Rightarrow x^{(1)} = \begin{bmatrix} 1 \\ 1.0 \\ 0.0 \end{bmatrix}$.

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

how much they like romantic movies and how much they like action movies.

Given $\theta^{(1)}, \ldots, \theta^{(n_u)}$, to learn $x^{(i)}$ (ith movie).

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

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

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

Collaborative filtering

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

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

Every users are helping algorithm to learn a better features, and these features can be used by the system to make better movies predictions for everyone else.

Guess $\Theta \to x \to \Theta \to x \to \Theta \to x \to \cdots$

b. Collaborative Filtering Algorithm:

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

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

sum over all users j and then sum over all movies rated by that user.

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

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

for every movie i, sum over all the users j that have rated that movie.

drop!! $x_0 = 1$

when we learn features in this way. convention $x_0 = 1$, but in this way $x \in \mathbb{R}^n$.

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

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

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

Collaborative filtering algorithm

$x_0 = 1$ $\quad x \in \mathbb{R}^n$, $\theta \in \mathbb{R}^n$

1. Initialize $x^{(1)}, \ldots, x^{(n_m)}, \theta^{(1)}, \ldots, \theta^{(n_u)}$ to small random values. ($\theta_0, \theta_1, \ldots, \theta_n$)
2. Minimize $J(x^{(1)}, \ldots, x^{(n_m)}, \theta^{(1)}, \ldots, \theta^{(n_u)})$ using gradient descent (or an advanced optimization algorithm). E.g. for every $j = 1, \ldots, n_u, i = 1, \ldots, n_m$:

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

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

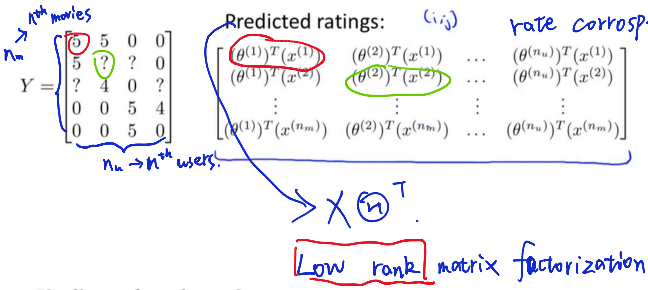
$\frac{\partial}{\partial x_k^{(i)}} J(\ldots)$

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

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

- **Low Rank Matrix Factorization:**

a. Vectorization: Low Rank Matrix Factorization:



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

Finding related movies

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

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

How to find movies j related to movie i?

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

⇒ 5 most similar to movie i: Find the 5 movies j with the smallest $\|x^{(i)} - x^{(j)}\|$.

Finding related movies

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

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

How to find movies j related to movie i?

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

$\Rightarrow$ 5 most similar to movie i: Find the 5 movies j with the smallest $\|x^{(i)} - x^{(j)}\|$.

b. Implementational Detail: Mean Normalization:

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

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

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

$n=2$ $\quad \theta^{(5)} \in \mathbb{R}^2$ $\quad$ equals 0 for Eve. $\quad \frac{\lambda}{2}[(\theta_1^{(5)})^2 + (\theta_2^{(5)})^2] \Rightarrow \theta^{(5)} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$

so for all $x^{(i)}$, $(\theta^{(5)})^T x^{(i)} = 0$.

Mean Normalization:

$$Y = \begin{bmatrix} 5 & 5 & 0 & 0 & ? \\ 5 & ? & ? & 0 & ? \\ ? & 4 & 0 & ? & ? \\ 0 & 0 & 5 & 4 & ? \\ 0 & 0 & 5 & 0 & ? \end{bmatrix} \quad \mu = \begin{bmatrix} 2.5 \\ 2.5 \\ 2 \\ 2.25 \\ 1.25 \end{bmatrix} \rightarrow Y = \begin{bmatrix} 2.5 & 2.5 & -2.5 & -2.5 & ? \\ 2.5 & ? & ? & -2.5 & ? \\ ? & 2 & -2 & ? & ? \\ -2.25 & -2.25 & 2.75 & 1.75 & ? \\ -1.25 & -1.25 & 3.75 & -1.25 & ? \end{bmatrix}$$

learn $\theta^{(j)}, x^{(i)}$

For user j, on movie i predict:

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

User 5 (Eve):

$\theta^{(5)} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$ $\quad (\theta^{(5)})^T (x^{(i)}) + \mu_i$ ($= 0$)