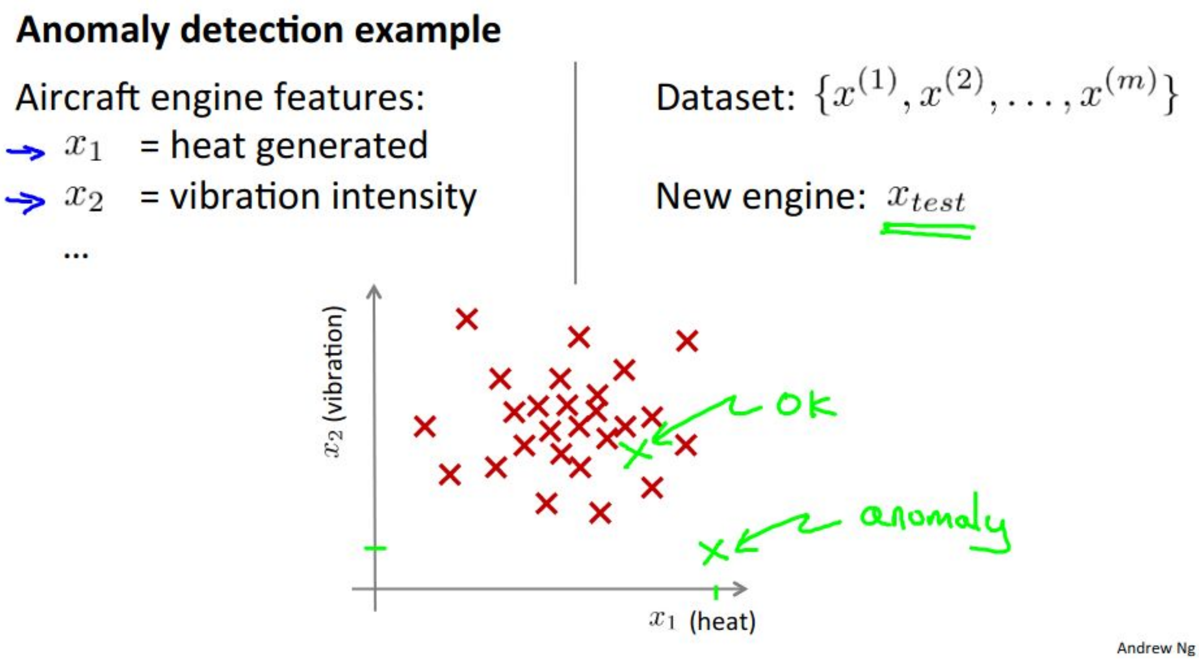
Week : 9

Part one

Anomaly detection :

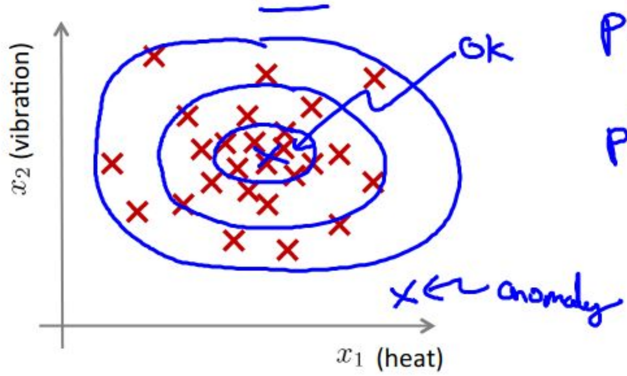
Suppose we have some training sets we want to select which of them are faulty and which are not. In that case we need to make such graph as below using their features and predict new test sets.



Density estimation

- Dataset: $\{x^{(1)}, x^{(2)}, \ldots, x^{(m)}\}$
- Is x_{test} anomalous?

Model $\underline{p(x)}$.



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

$p(x_{test}) \geq \varepsilon \rightarrow$ Ok

Andrew Ng

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 $\underline{p(x) < \varepsilon}$
- Manufacturing
- 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 = <u>CPU load</u>/<u>network traffic</u>.

    ...

x_1
x_2
x_3
x_4

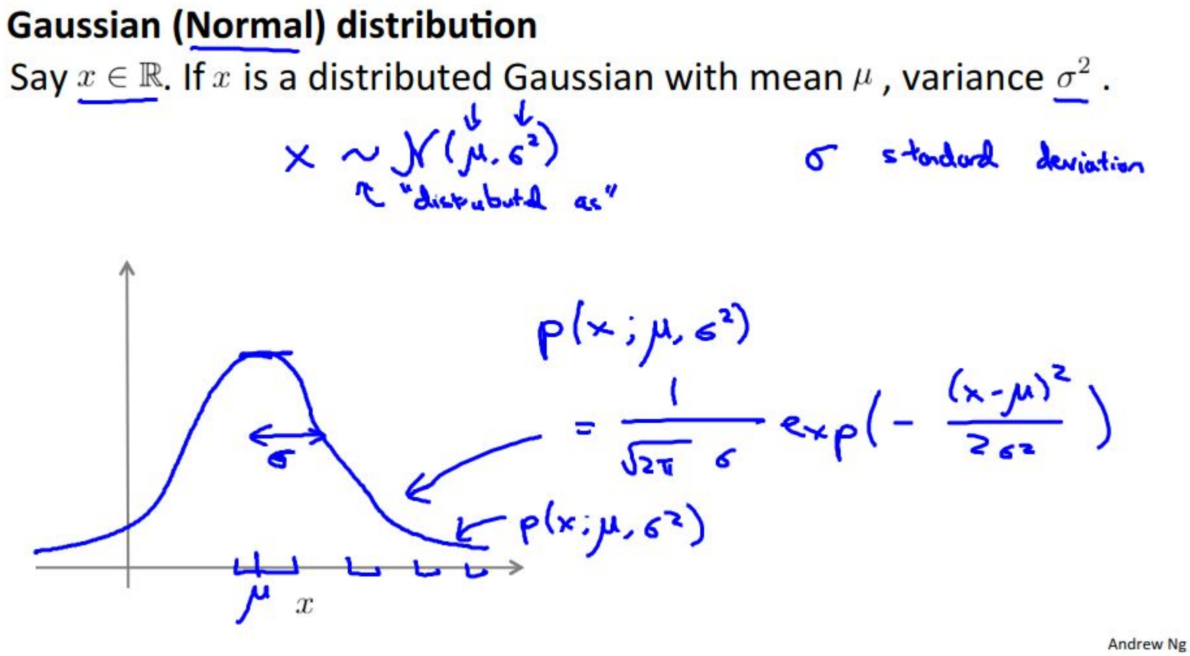
$p(x)$

$p(x) < \varepsilon$

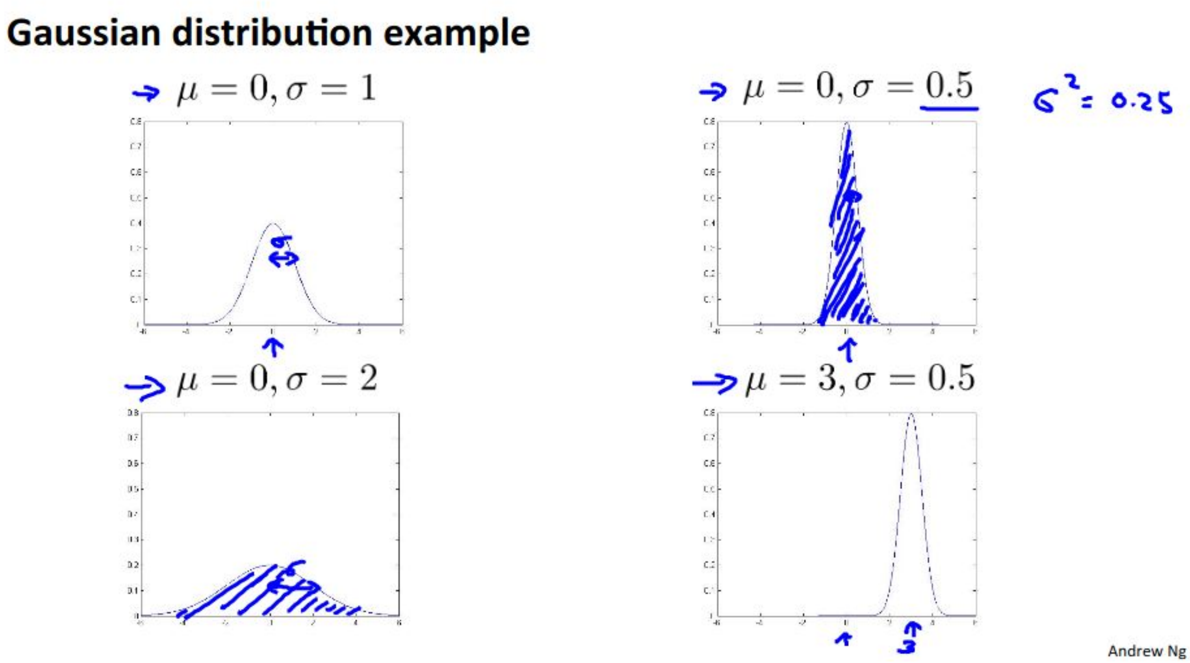
Andrew Ng

Gaussian Distribution :

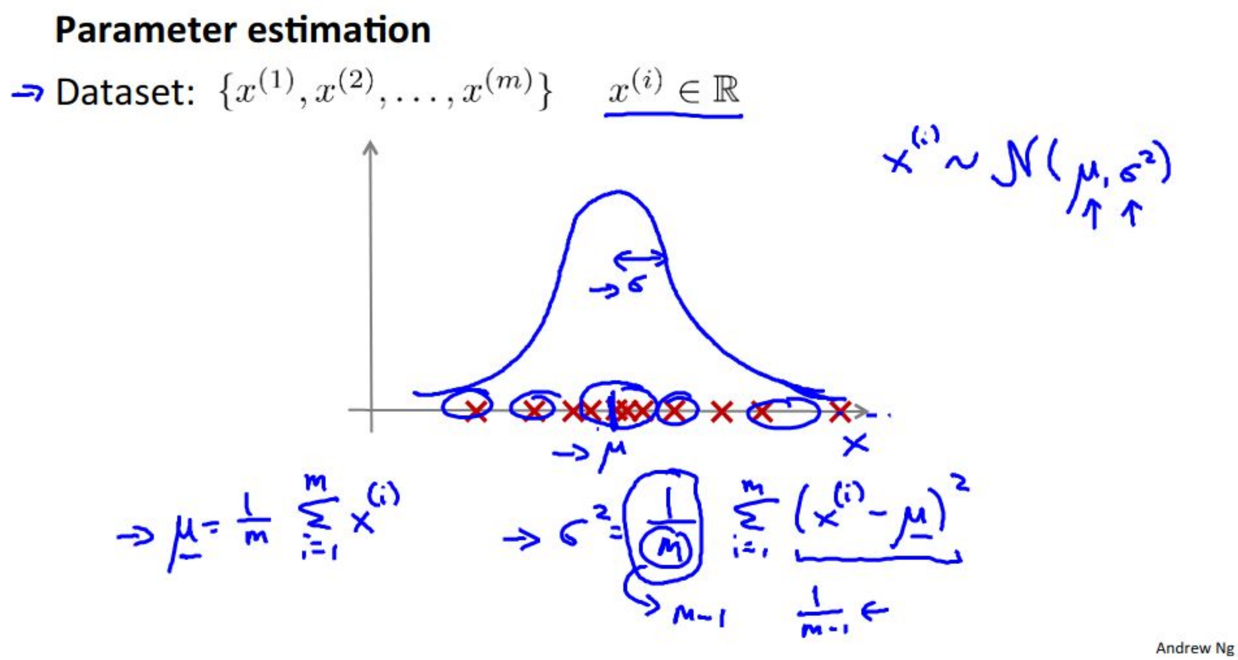
Gaussian distribution is a probability distribution by which we can simulate the density of our training sets. There we used normal gaussian distribution where two perimeter are used : Mewe and sigma. There mewe is the center of the density and sigma is the width of the curve.



From the image below we can see that as sigma grows bigger, the width of the curve becomes thicker.

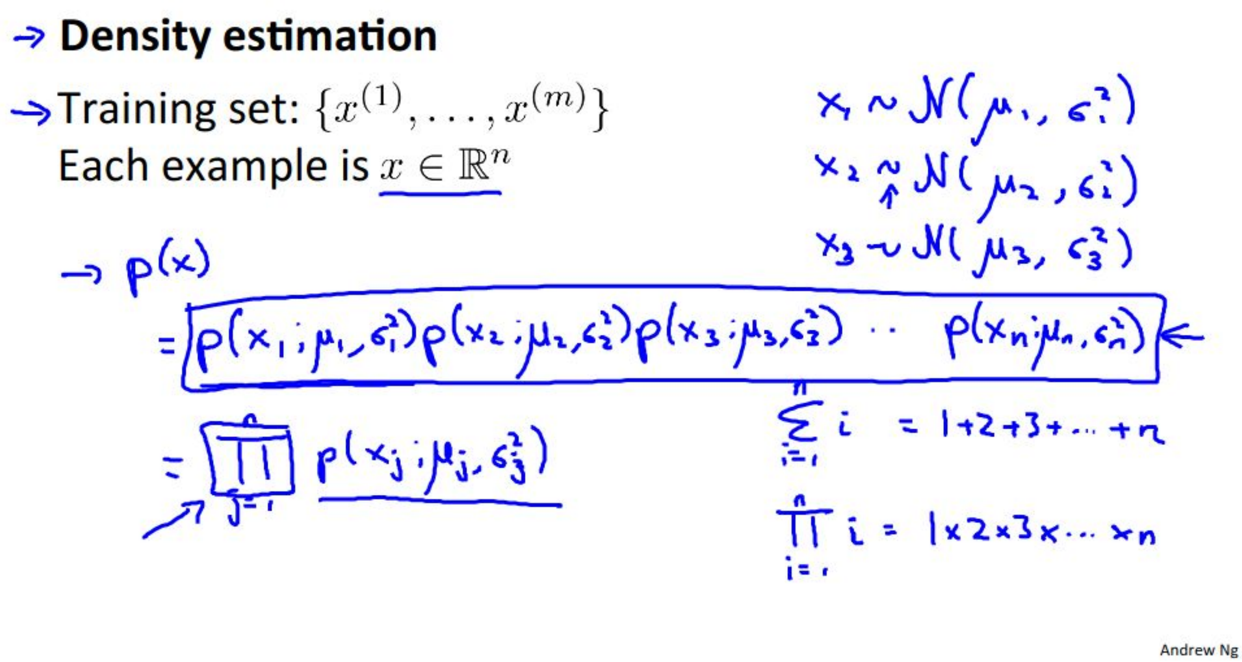


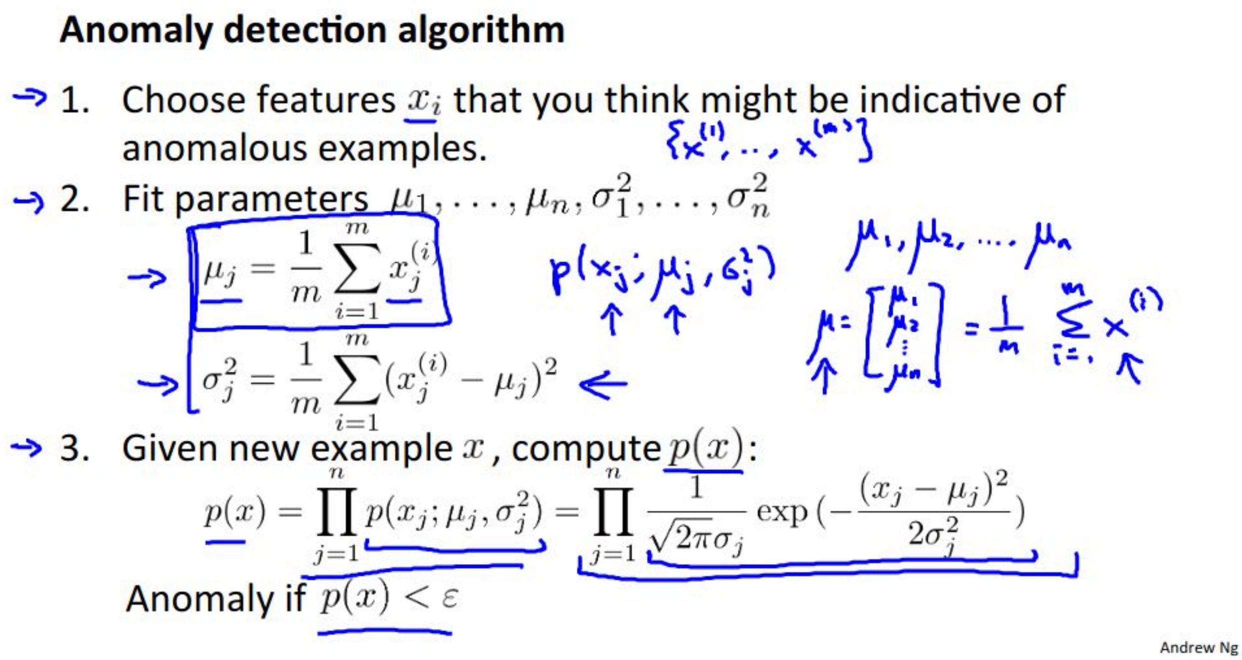
If we have a graph with some random training sets and want to find mewe and sigma, can apply the formula of mewe and sigma.

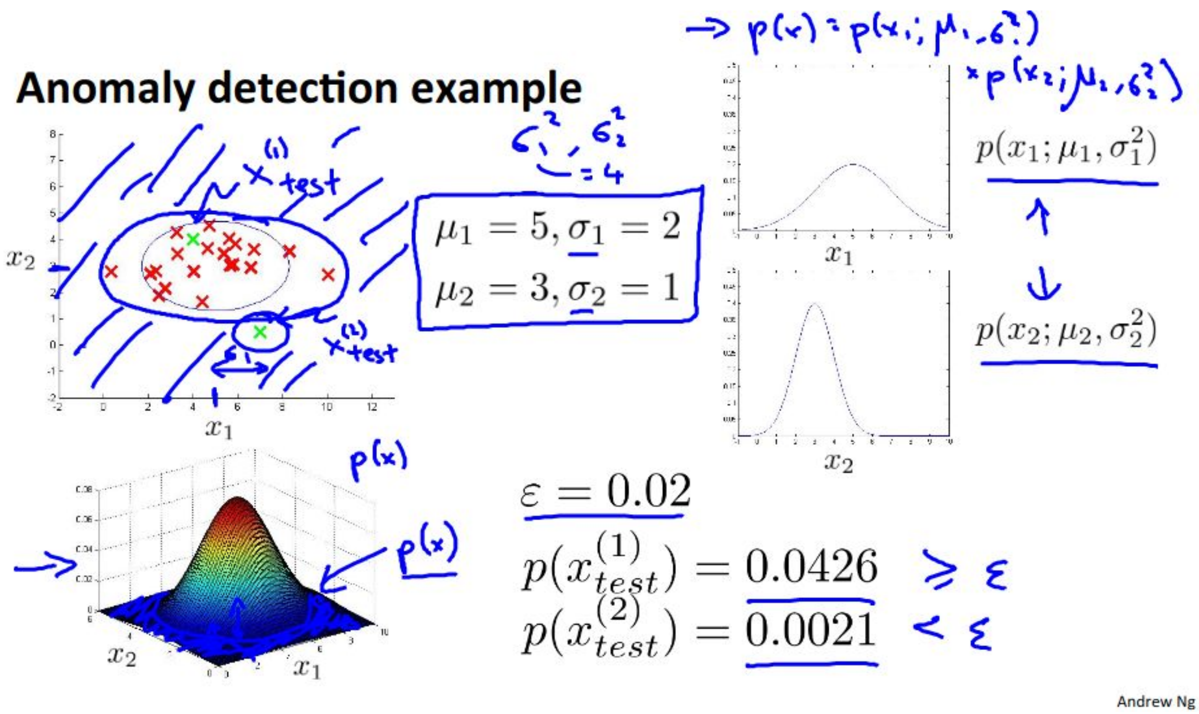


Anomaly detection algorithm :

In this part we will learn an algorithm which will help us to separate test set anomaly and non-anomaly using gaussian distribution.







Developing and evaluating an anomaly detection system :

In order to develop an anomaly evaluation system we need to follow the same process of supervised learning algorithms. First we need some training sets with labels to train our algorithm and then we will have a graph from which we can plot a normal gaussian distribution graph. And this gaussian distribution graph will help us to say if a test set is anomaly or not!

The importance of real-number evaluation

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: $x^{(1)}, x^{(2)}, \dots, x^{(m)}$ (assume normal examples/not anomalous)
- Cross validation set: $(x_{cv}^{(1)}, y_{cv}^{(1)}), \dots, (x_{cv}^{(m_{cv})}, y_{cv}^{(m_{cv})})$
- Test set: $(x_{test}^{(1)}, y_{test}^{(1)}), \dots, (x_{test}^{(m_{test})}, y_{test}^{(m_{test})})$

$y=1$

Andrew Ng

If we have 10k training sets with labels then we can divide them into 60-20-20 for training sets, cross-validation sets and test sets consecutively. Thus we can develop our algorithm.

Aircraft engines motivating example

- 10000 good (normal) engines
- 20 flawed engines (anomalous) 2-50

$y=1$

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

- Training set: 6000 good engines ($y=0$) $p(x) = p(x_1; \mu_1, \sigma_1^2) \dots p(x_n; \mu_n, \sigma_n^2)$

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

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

Alternative:

Training set: 6000 good engines

- CV: 4000 good engines ($y = 0$), 10 anomalous ($y = 1$)
- Test: 4000 good engines ($y = 0$), 10 anomalous ($y = 1$)

Andrew Ng

For evaluating, we need to fit the unlabeled training sets into gaussian distribution and then evaluate our cross validation or test sets.

Algorithm evaluation

- Fit model $p(x)$ on training set $\{x^{(1)}, \ldots, x^{(m)}\}$ $(x^{(i)}_{test}, y^{(i)}_{test})$
- On a cross validation/test example x, predict

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

$y=0$

Possible evaluation metrics:

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

CV

Test set

Can also use cross validation set to choose parameter ε

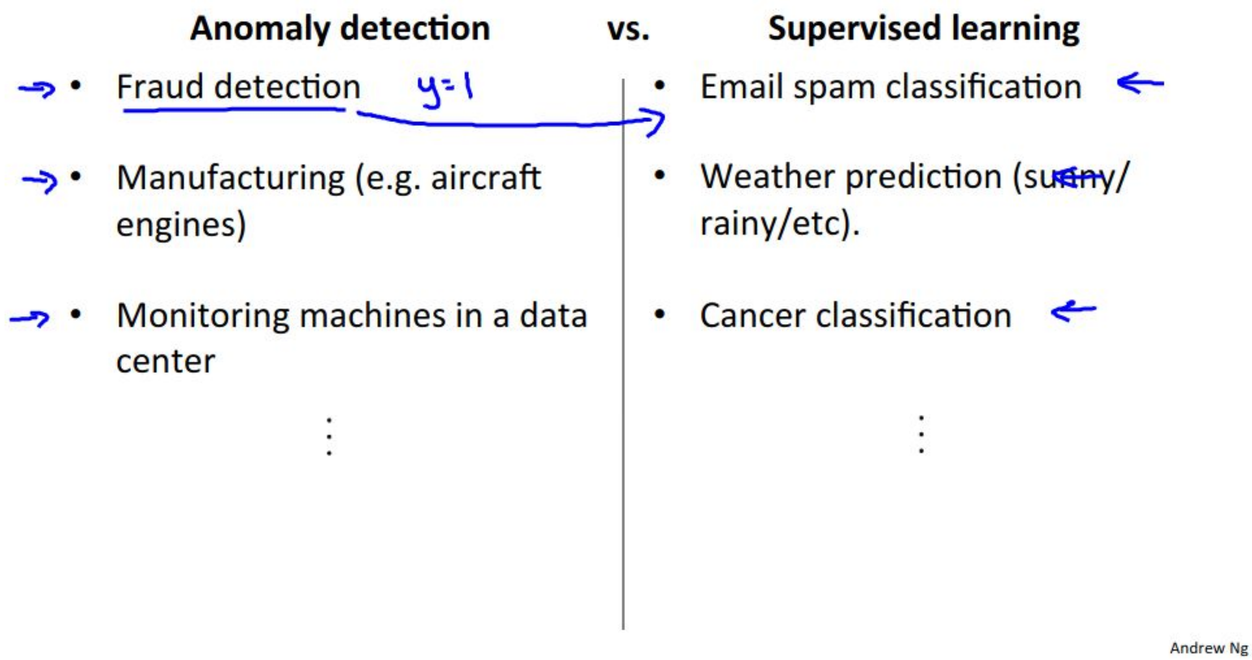
Andrew Ng

Anomaly vs supervised learning :

First few steps of anomaly and supervised learning are the same. So it may be confusing that when should we use anomaly detection and when supervised learning?
It that case we need to evaluate some fact:

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. $\boxed{p(x)}$	
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.	Spam

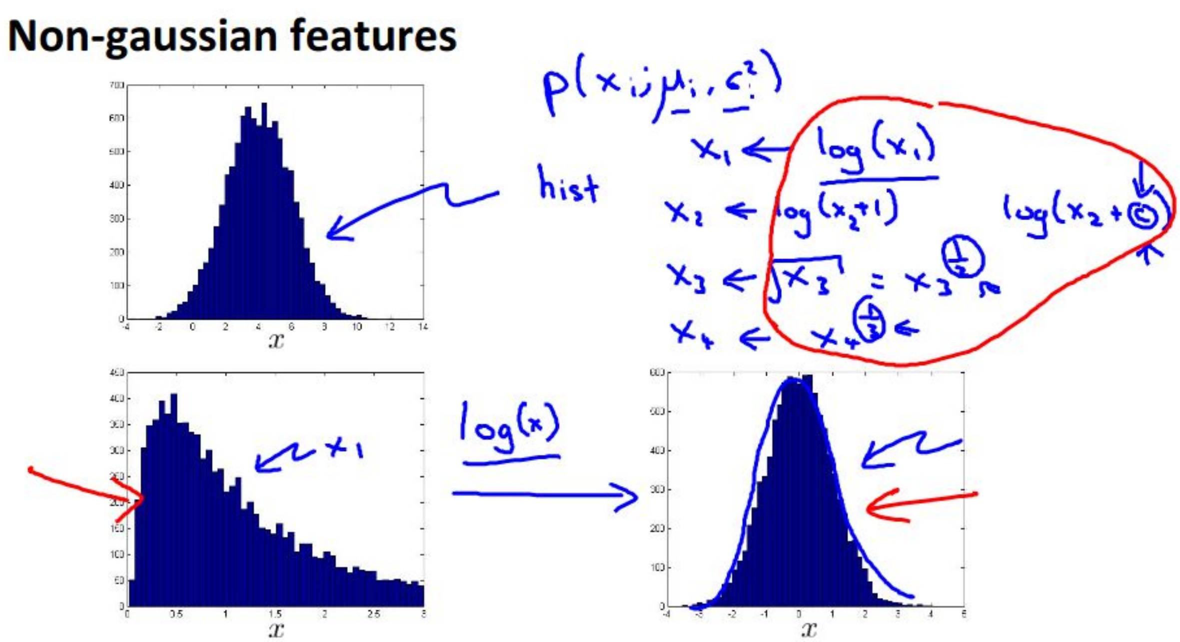
Andrew Ng



Choosing what feature to use:

Sometimes it may be necessary that we need to choose the right features or debug some feature. In that case we need to follow some technique:

1. If our datasets don't give a bowl shape graph then we need to customize our features.

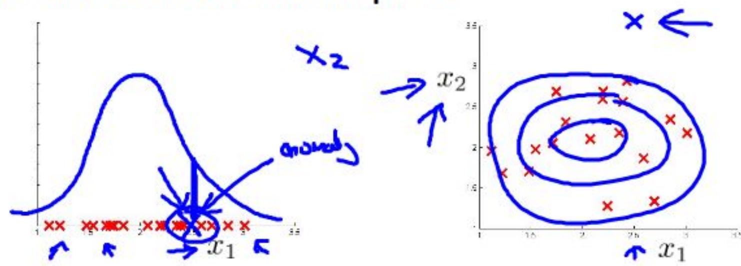


2. Sometimes we need to choose a new feature to evaluate new test case:

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



3. If you think that one or more features become faulty then you need to customize them with new features. In the picture below, we get x3,x4 linearly high. It that case, to detect which feature is faulty we customize them with x5 or x6 feature.

Monitoring computers in a data center

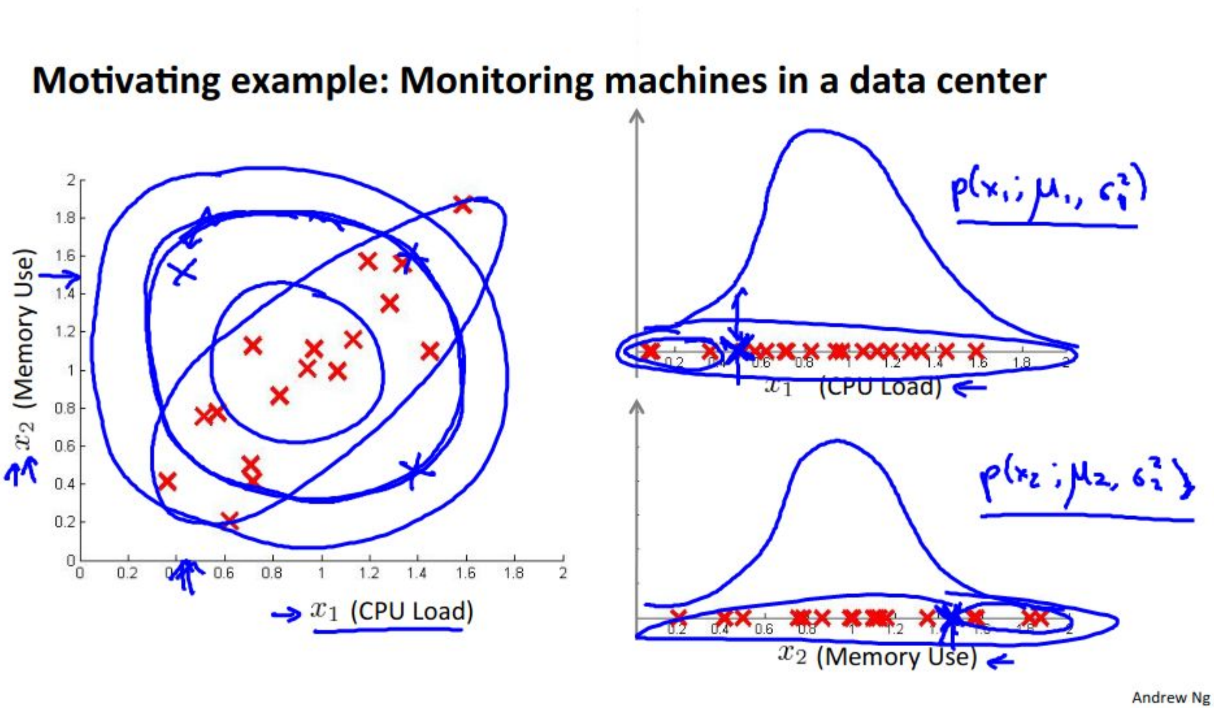
Choose features that might take on unusually large or small values in the event of an anomaly.

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

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

Multivariate Gaussian Distribution :
Sometimes normal gaussian distribution may mark some test cases as non anomaly cases though they are not. It happens because of doing multiplication. In that case, we

can use multivariate gaussian distribution to get more perfect result.



Multivariate Gaussian (Normal) distribution

$x \in \mathbb{R}^n$. Don't model $p(x_1), p(x_2), \ldots$, etc. separately.
Model $p(x)$ all in one go.
Parameters: $\mu \in \mathbb{R}^n$, $\boxed{\Sigma \in \mathbb{R}^{n \times n}}$ (covariance matrix)

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

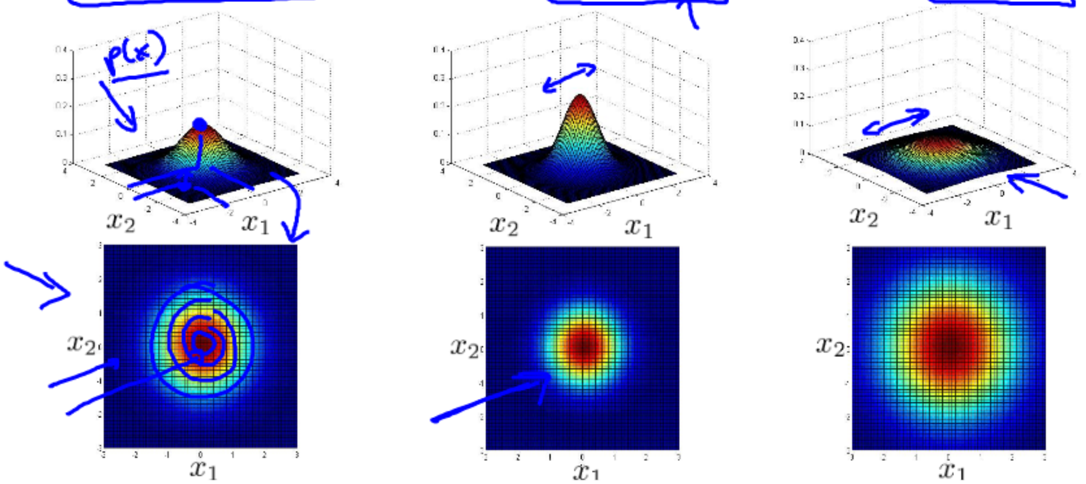
$|\Sigma|$ = determinant of Σ | det(Sigma)

Andrew Ng

Here are some graphs of multivariate gaussian distribution. As we notice, the graphs change with the values of mewe and sigma.

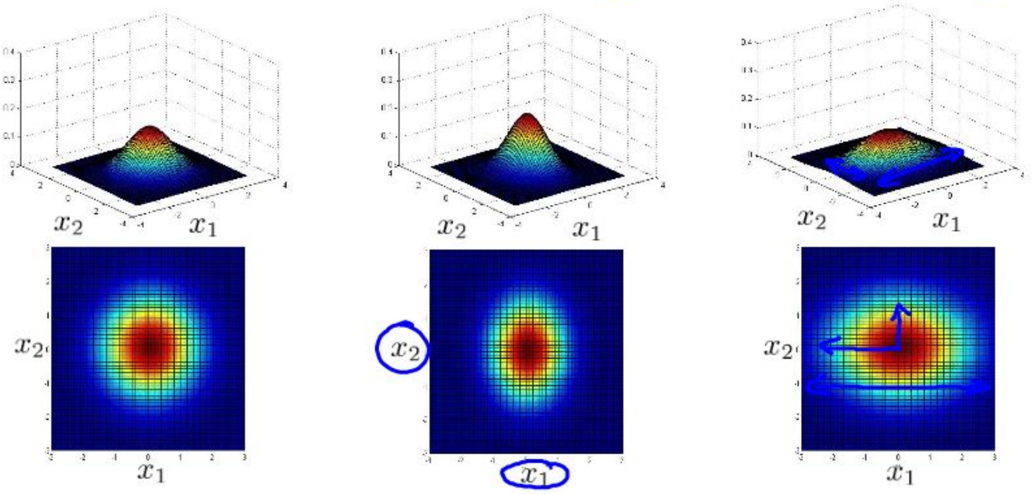
Multivariate Gaussian (Normal) examples

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



Multivariate Gaussian (Normal) examples

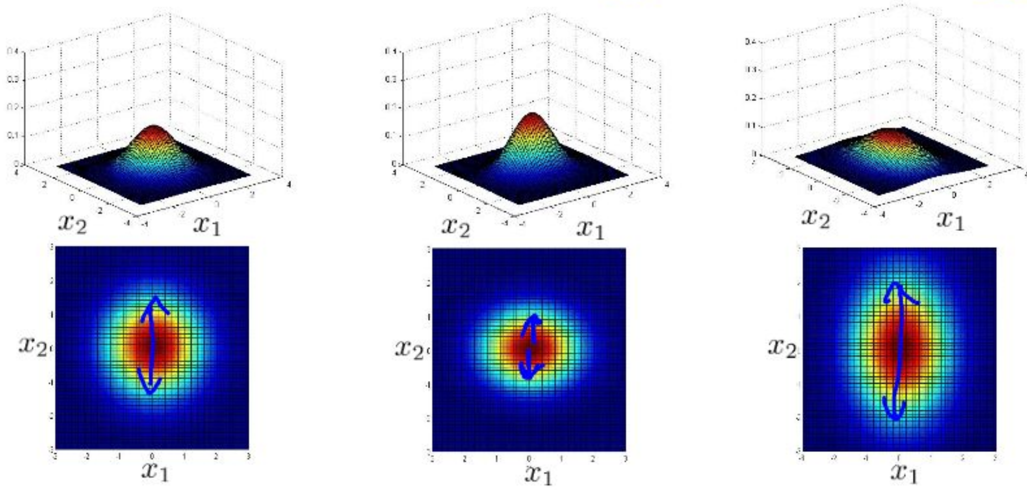
$\mu = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \; \Sigma = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$ $\quad \mu = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \; \Sigma = \begin{bmatrix} 0.6 & 0 \\ 0 & 1 \end{bmatrix}$ $\quad \mu = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \; \Sigma = \begin{bmatrix} 2 & 0 \\ 0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



Andrew Ng

Multivariate Gaussian (Normal) examples

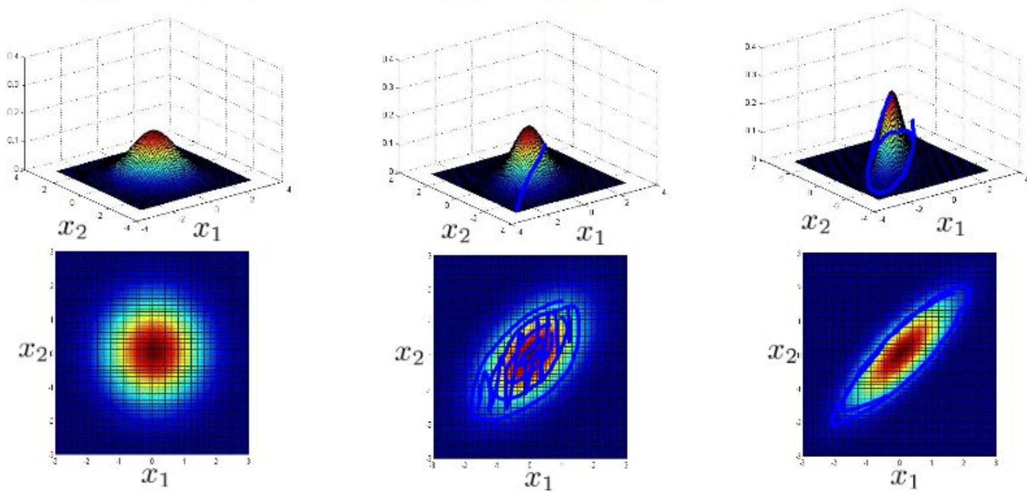
$\mu = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \Sigma = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$ $\mu = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \Sigma = \begin{bmatrix} 1 & 0 \\ 0 & 0.6 \end{bmatrix}$ $\mu = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \Sigma = \begin{bmatrix} 1 & 0 \\ 0 & 2 \end{bmatrix}$



Andrew Ng

Multivariate Gaussian (Normal) examples

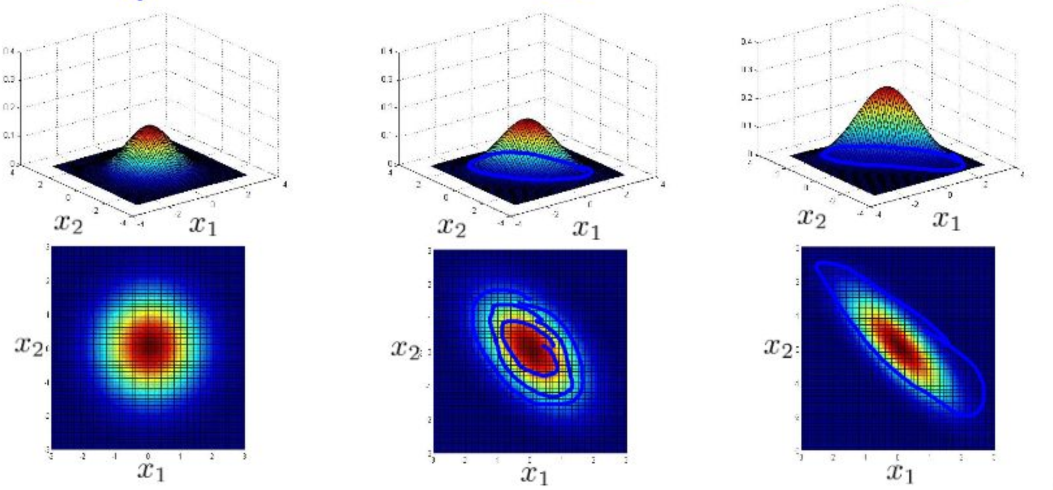
$\mu = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \Sigma = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$ $\mu = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \Sigma = \begin{bmatrix} 1 & 0.5 \\ 0.5 & 1 \end{bmatrix}$ $\mu = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \Sigma = \begin{bmatrix} 1 & 0.8 \\ 0.8 & 1 \end{bmatrix}$



Andrew Ng

Multivariate Gaussian (Normal) examples

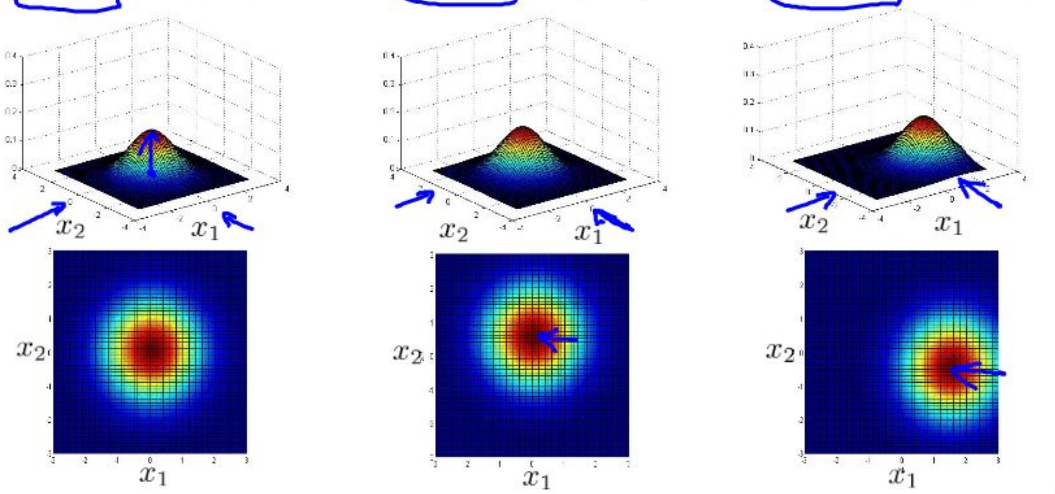
$\mu = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \Sigma = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$ $\mu = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \Sigma = \begin{bmatrix} 1 & -0.5 \\ -0.5 & 1 \end{bmatrix}$ $\mu = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \Sigma = \begin{bmatrix} 1 & -0.8 \\ -0.8 & 1 \end{bmatrix}$



Andrew Ng

Multivariate Gaussian (Normal) examples

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



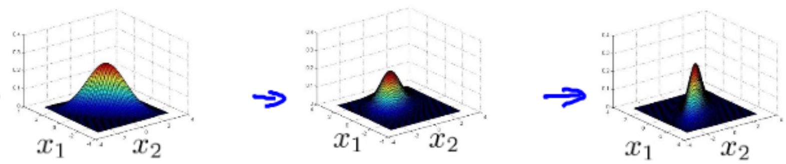
Andrew Ng

Anomaly direction using multivariate gaussian distribution :

Multivariate Gaussian (Normal) distribution

Parameters μ, Σ $\quad \mu \in \mathbb{R}^n \quad \Sigma \in \mathbb{R}^{n \times n}$

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



Parameter fitting:

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

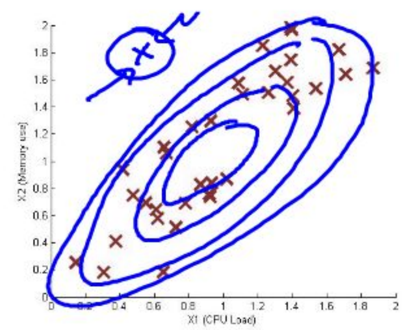
$$\boxed{\mu} = \frac{1}{m}\sum_{i=1}^{m} x^{(i)} \qquad \boxed{\Sigma} = \frac{1}{m}\sum_{i=1}^{m}(x^{(i)}-\mu)(x^{(i)}-\mu)^T$$

Anomaly detection with the multivariate Gaussian

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

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

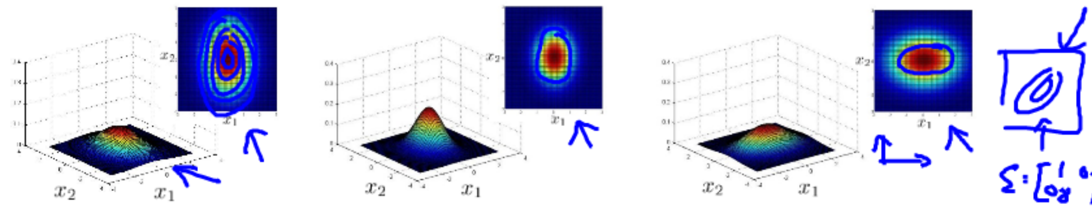
Flag an anomaly if $p(x) < \varepsilon$

Andrew Ng

Relationship to original model

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

$p(x)$



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

Corresponds to multivariate Gaussian

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

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

Andrew Ng

Original model vs. Multivariate Gaussian

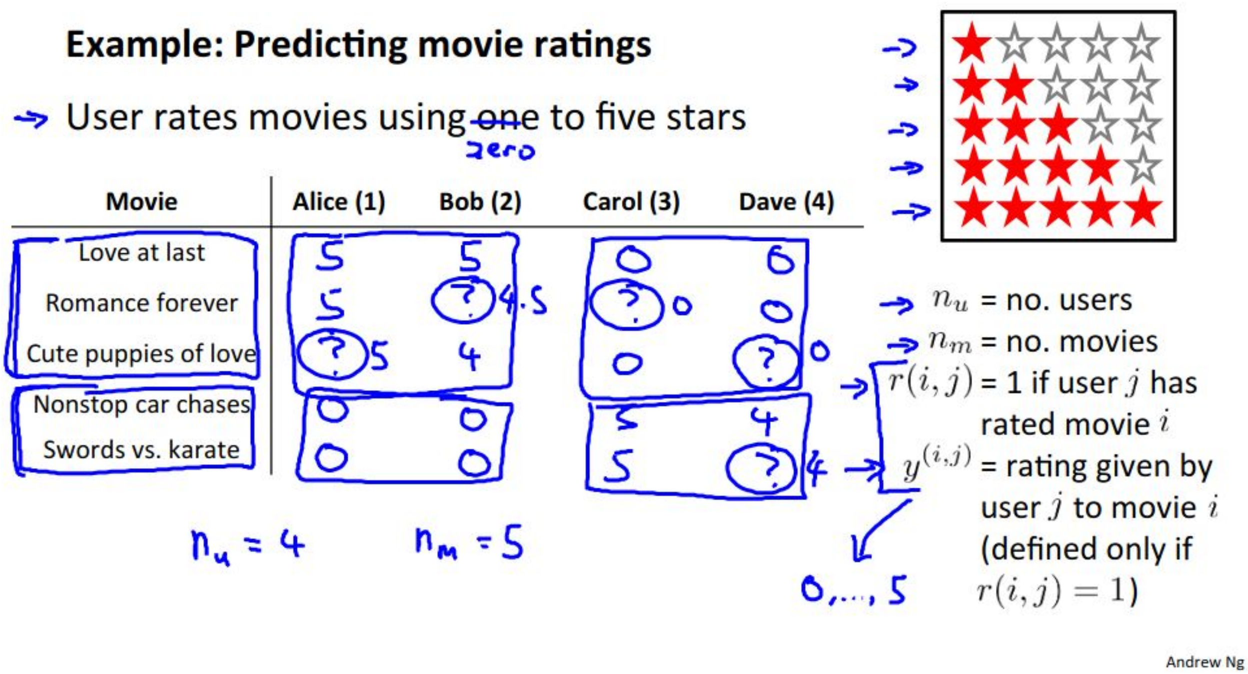
Original model	Multivariate Gaussian
$p(x_1;\mu_1,\sigma_1^2) \times \cdots \times p(x_n;\mu_n,\sigma_n^2)$	$p(x;\mu,\Sigma) = \frac{1}{(2\pi)^{\frac{n}{2}}\|\Sigma\|^{\frac{1}{2}}}\exp\left(-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)\right)$
Manually create features to capture anomalies where x_1, x_2 take unusual combinations of values. $\boxed{x_3 = \frac{x_1}{x_2} = \frac{\text{CPU load}}{\text{memory}}}$	Automatically captures correlations between features. $\Sigma \in \mathbb{R}^{n\times n}$ $\quad \Sigma^{-1}$
Computationally cheaper (alternatively, scales better to large n) $n=10{,}000$, $n=100{,}000$	Computationally more expensive. $\Sigma \sim \frac{n^2}{2}$ $\quad x_1 = \cancel{x_2}$, $\cancel{x_3} = x_4 + x_5$
OK even if m (training set size) is small	Must have $\boxed{m > n}$ or else Σ is non-invertible. $\boxed{m \geq 10n}$

Andrew Ng

Part two

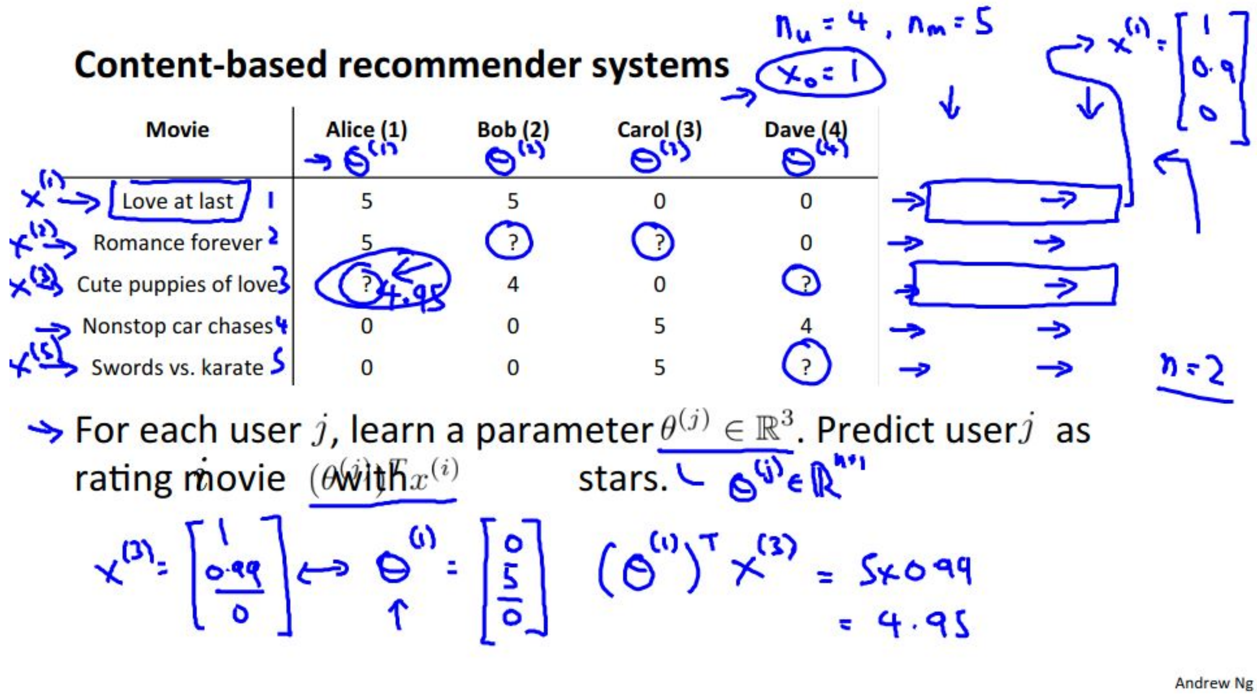
Recommender system :

Online tvs (e.g : netflix, amazon, HVO) use this system to suggest movies or series to their users. In this system they let their users rate movies or series from zero to five stars and then predict their want and choice.

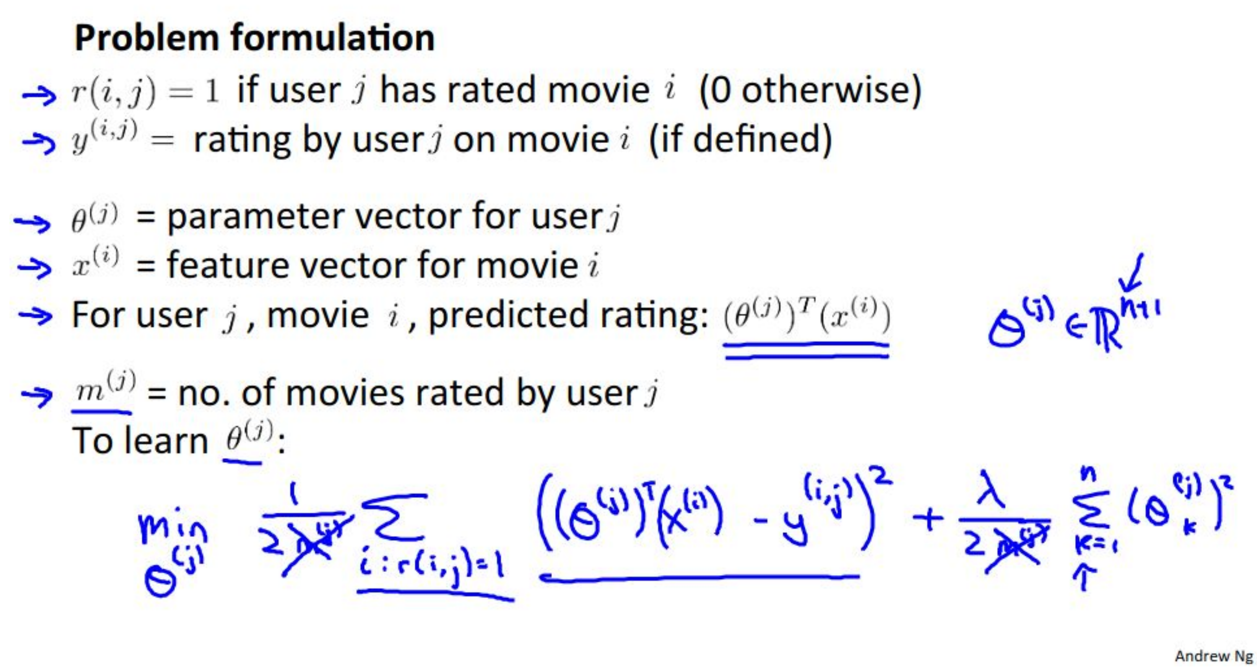


Content based recommendation :

Previously we saw that users rate movies from zero to five stars. But they don't rate all of the movies. In this section we want to make a prediction by which we can say how much a user would rate an unrated movie based on their other movie rating.



The prediction is likely to linear regression.



We want to predict for all users as we calculate for theta 1,2,3,...

Optimization objective:

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

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

Andrew Ng

Optimization algorithm:

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

Gradient descent update:

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

$$\theta_k^{(j)} := \theta_k^{(j)} - \alpha \underbrace{\left(\sum_{i:r(i,j)=1}((\theta^{(j)})^T x^{(i)} - y^{(i,j)})x_k^{(i)} + \lambda\theta_k^{(j)}\right)}_{\frac{\partial}{\partial\Theta_k^{(j)}} J(\Theta^{(1)},\ldots,\Theta^{(n_u)})} \quad \text{(for } k \neq 0)$$

$\cancel{\frac{1}{m^{(j)}}}$

Andrew Ng

Collaborative filtering :

Sometimes it may be difficult to set specific feature values for all movies. Instead of doing that we can let our users rate their taste on different features. Using those inputs we can calculate the specific values for different movies.

Problem motivation

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

Andrew Ng

Problem motivation

$x_0 = 1$

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	~~?~~ 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	?	?	?

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

$x^{(1)}$

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

$\theta^{(j)}$

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

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

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

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

Andrew Ng

Optimization algorithm

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

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

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

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

Collaborative filtering

$r^{(i,j)}$
$y^{(i,j)}$

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

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

Guess $\Theta \rightarrow x \rightarrow \Theta \rightarrow x \rightarrow \Theta \rightarrow x \rightarrow \cdots$

Collaborative filtering algorithm :

In the previous section we learned that by collaborative filtering we can train theta and X one after another. That means we need to use X to get min theta and to get min X need to use theta. But in collaborative filtering algorithm we can minimize our perimeter theta and X simultaneously.

Collaborative filtering optimization objective

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

$\cancel{x_0 = 1}$ $\quad x \in \mathbb{R}^n \quad \cancel{x \in \mathbb{R}^{n+1}}$

$\cancel{\theta_0}$ $\quad \theta \in \mathbb{R}^n \quad x_1 = 1$

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

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

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

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

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

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

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

Andrew Ng

Collaborative filtering algorithm

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

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

→ 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 $j = 1, \ldots, n_u, i = 1, \ldots, n_m$:

$\cancel{\theta_0}$, θ_1, …, θ_n

$$x_k^{(i)} := x_k^{(i)} - \alpha \underbrace{\left(\sum_{j:r(i,j)=1} \left((\theta^{(j)})^T x^{(i)} - y^{(i,j)} \right) \theta_k^{(j)} + \lambda x_k^{(i)} \right)}_{\frac{\partial}{\partial x_k^{(i)}} J(\ldots)} \leftarrow$$

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

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

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

Andrew Ng

Vectorize low rank matrix factorization :
Instead of using loops to calculate collaborative filtering we can use vectorize implementation. It will make computation easier.

Collaborative filtering

$n_m = 5$
$n_u = 4$

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	?

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

$y^{(i,j)}$

Andrew Ng

Collaborative filtering

$X \Theta^T$

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

(i,j)

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

Predicted ratings:

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

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

→ Low rank matrix factorization

Andrew Ng

To improve user experience online tvs need to suggest related movies or videos. In that case they use existing user data to do that. Suppose, movie x1 is watched by a user. Now if you want to test if x2 movie is related to x1 then you need to following steps.

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

Andrew Ng

Implementation details : mean normalization :
Suppose a new user just signed up. He hasn't rated any movie yet. How do you suggest movies or other stuff? In that case, mean normalization will help us.

Users who have not rated any movies

Movie	Alice (1)	Bob (2)	Carol (3)	Dave (4)	Eve (5)
→ Love at last	5	5	0	0	? 0
Romance forever	5	?	?	0	? 0
Cute puppies of love	?	4	0	?	? 0
Nonstop car chases	0	0	5	4	? 0
→ Swords vs. karate	0	0	5	?	? 0

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

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

$n = 2$ $\quad \theta^{(5)} \in \mathbb{R}^2$ $\quad \theta^{(5)} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$ $\quad \frac{\lambda}{2}\left[(\theta_1^{(5)})^2 + (\theta_2^{(5)})^2\right]$ ←

$(\theta^{(5)})^T x^{(i)} = 0$

Andrew Ng

To apply this algorithm you need to compute the average rating for every single movie. Then subtract the avg value from original value. Then use this formula for user j, on movie i predict: