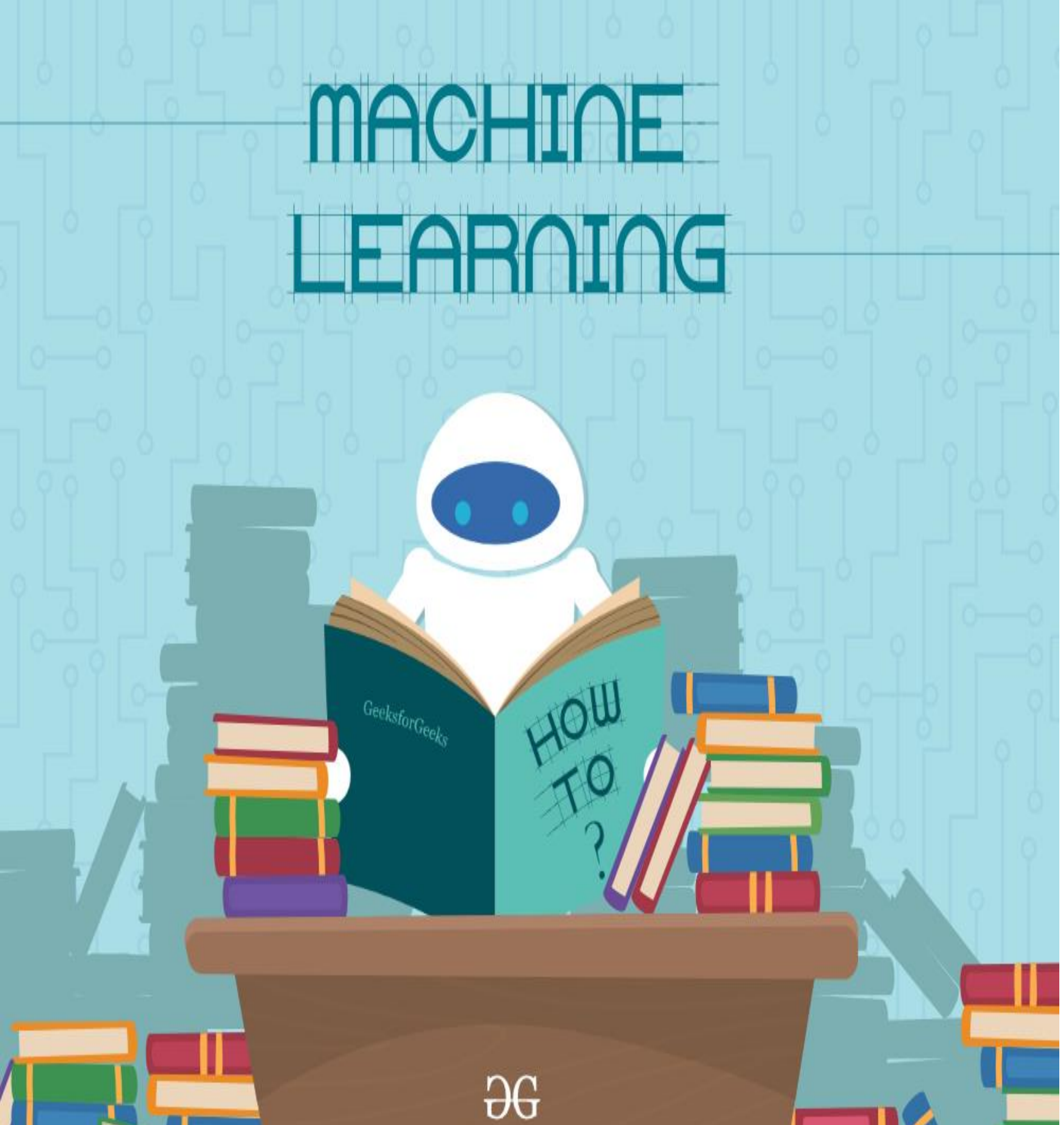


THE CODERS' CLUB

Anomaly Detection
Recommender Systems

WEEK 9

Anomaly Detection

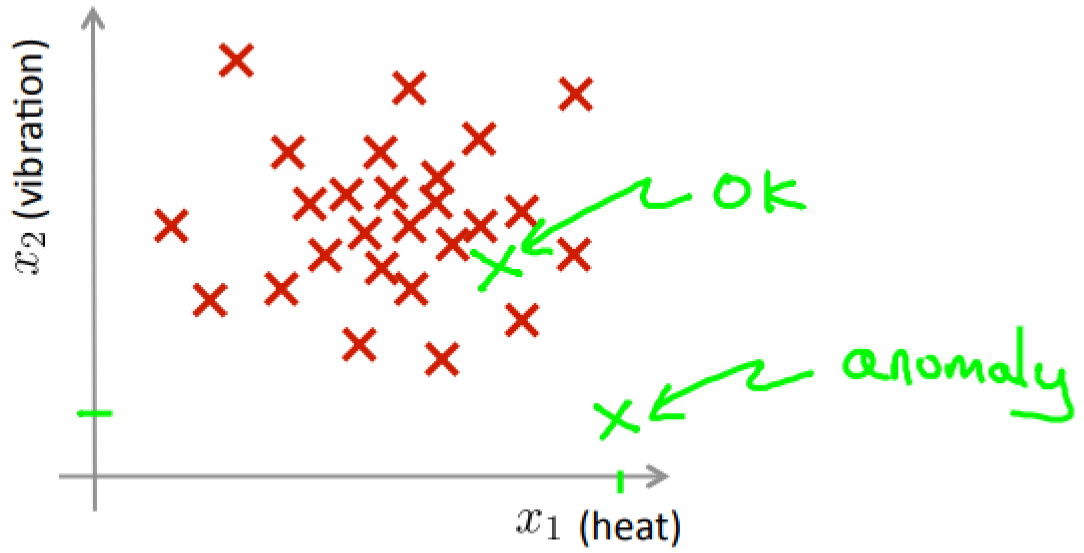
Consider an example of anomaly detection.

We have an unlabeled dataset $\{x^{(1)}, x^{(2)}, \dots, x^{(m)}\}$ and features x_1, x_2, … for an aircraft engine.

x_1 = heat generated

x_2 = vibration intensity

…

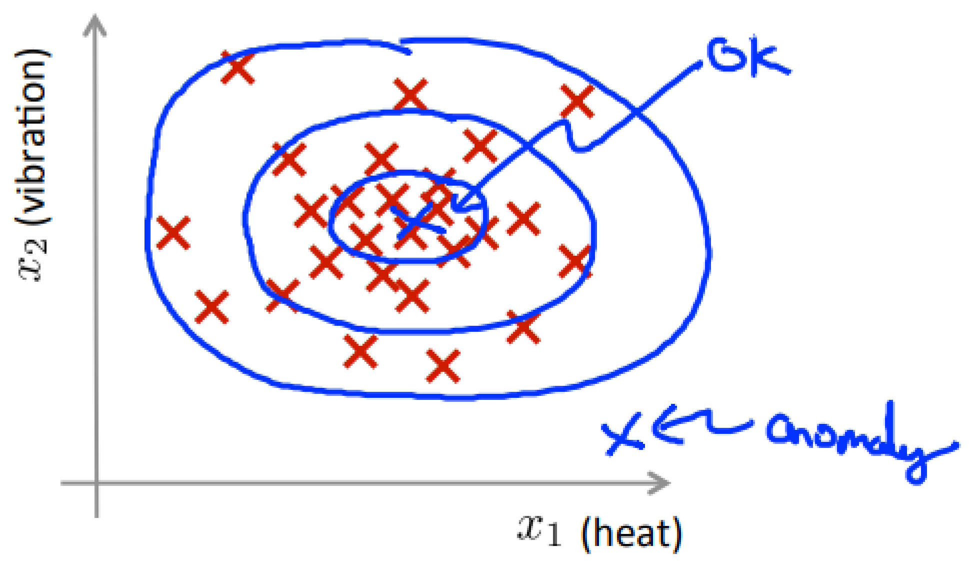


From the figure, it is evident that the example which lies far from other examples in the dataset is anomalous.

Density Estimation:

Consider we have a dataset $\{x^{(1)}, x^{(2)}, \dots, x^{(m)}\}$

We have to find whether x_{test} is anomalous or not?



- If $p(x_{test}) < \varepsilon$, then flag it as anomalous
- If $p(x_{test}) > \varepsilon$, then consider it as "OK"

Note: ε is some threshold

Anomaly Detection Examples:

1. **Fraud Detection:**

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

2. **Manufacturing**

3. **Monitoring computers in a data center**

- $x^{(i)}$ = features of machine i
- x_1 = memory use, x_2 = no. of disk accesses/sec, x_3 = CPU load, x_4 = CPU load/network traffic
  …

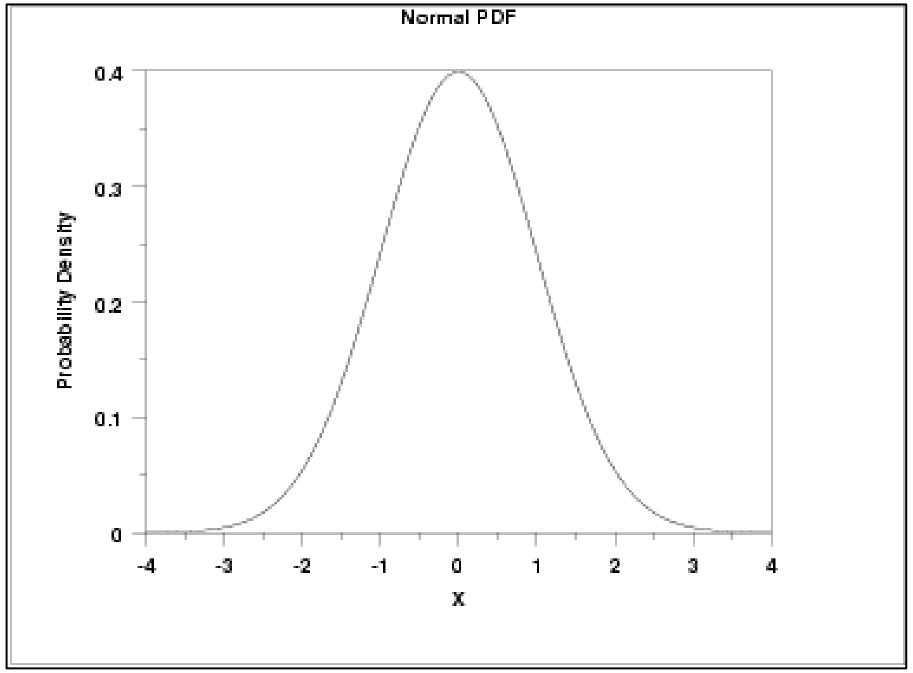
Your anomaly detection system flags x as anomalous whenever $p(x) \leq \epsilon$.
Suppose your system is flagging too many things as anomalous that are not actually so (similar to supervised learning, these mistakes are called false positives).
What should you do?

- Try increasing ϵ
- Try decreasing ϵ

Gaussian Distribution

Gaussian Distribution is also called "Normal Distribution".

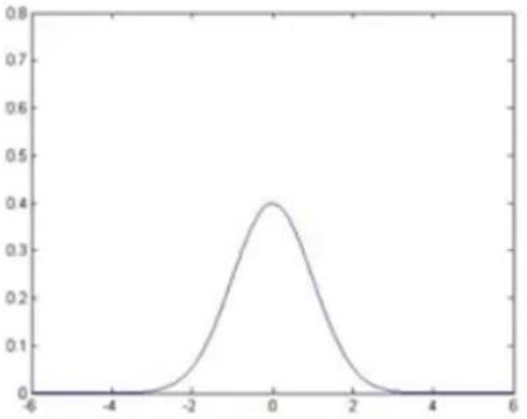
Say x ∈ R. If x is a distributed Gaussian with mean μ, variance σ^2 , then:



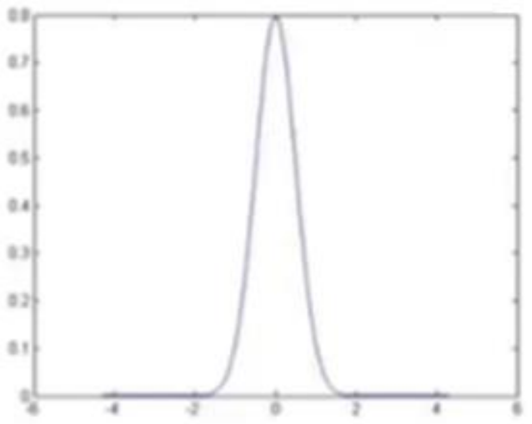
Gaussian Distribution is given by: $\mathcal{N}(x\ ;\ \mu,\sigma) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left[-\frac{1}{2}(x-\mu)^2/\sigma^2\right]$

Gaussian Distribution Examples:

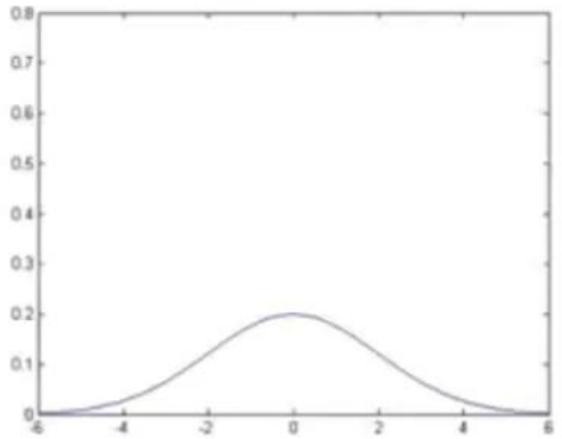
$\mu = 0, \sigma = 1$



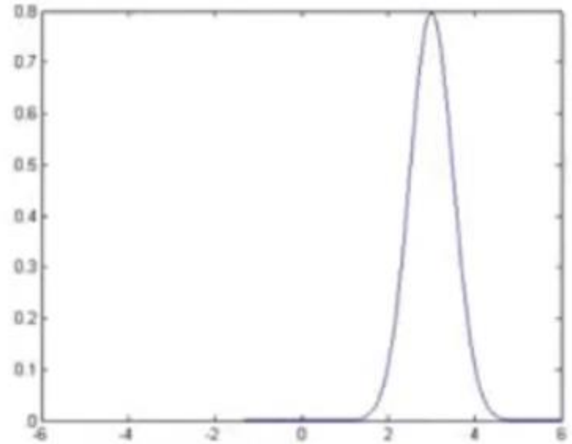
$\mu = 0, \sigma = 0.5$



$\mu = 0, \sigma = 2$



$\mu = 3, \sigma = 0.5$



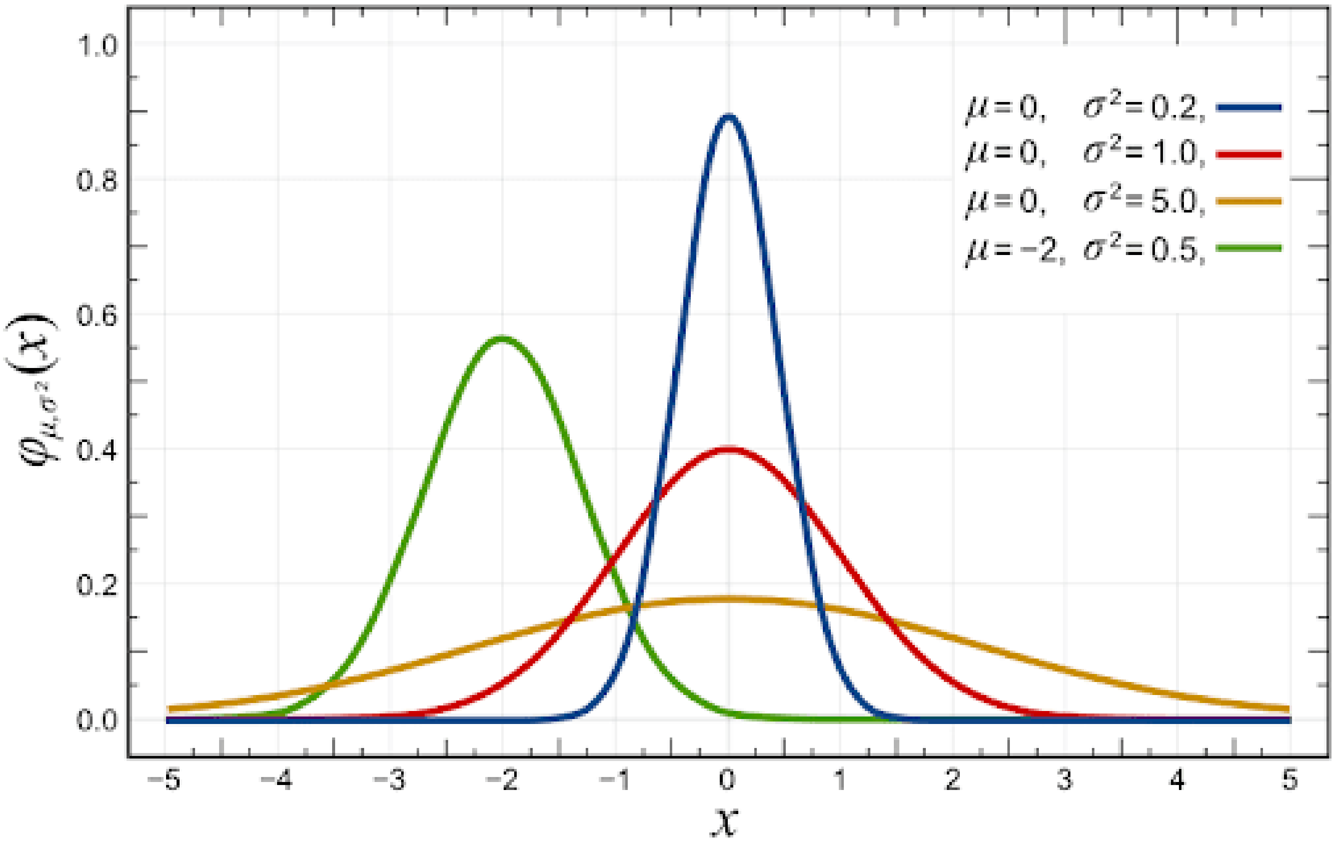
Parameter Estimation (Maximum Likelihood Estimation):

For the dataset $\{x^{(1)}, x^{(2)}, \dots , x^{(m)}\}$ where $x^{(i)} \in R$

The Maximum Likelihood Estimation is given by,

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

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



The formula for the Gaussian density is:

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

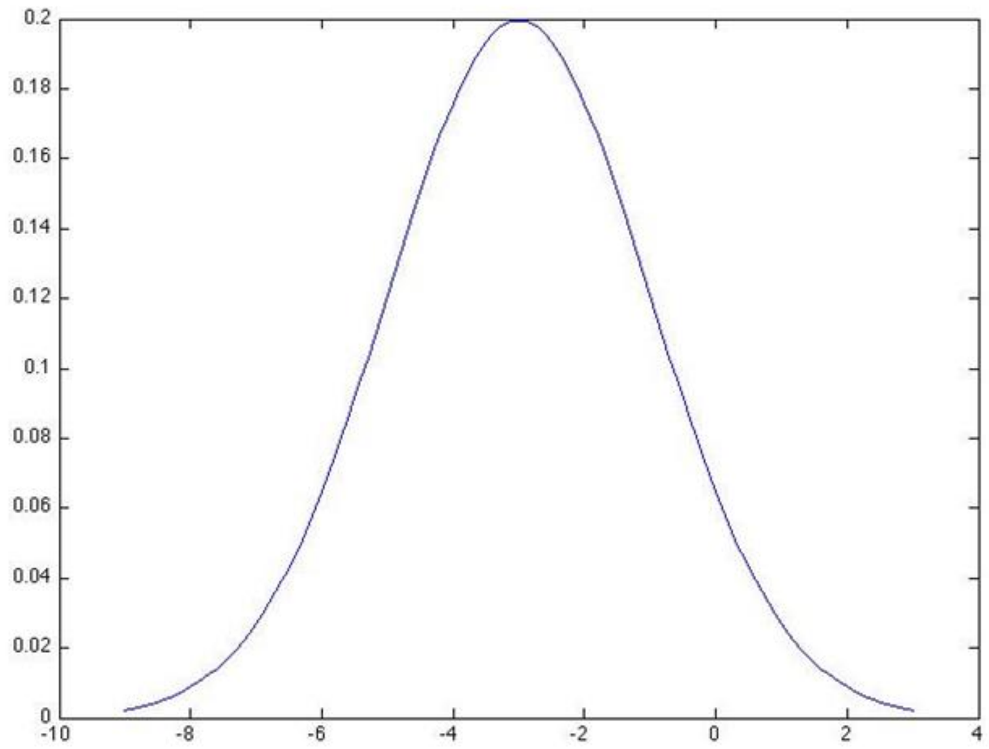
Which of the following is the formula for the density to the right?

$$p(x) = \frac{1}{\sqrt{2\pi}\times 2} \exp\left(-\frac{(x-3)^2}{2\times 4}\right)$$

$$p(x) = \frac{1}{\sqrt{2\pi}\times 4} \exp\left(-\frac{(x-3)^2}{2\times 2}\right)$$

[✓ $p(x) = \frac{1}{\sqrt{2\pi}\times 2} \exp\left(-\frac{(x+3)^2}{2\times 4}\right)$]

$$p(x) = \frac{1}{\sqrt{2\pi}\times 4} \exp\left(-\frac{(x+3)^2}{2\times 2}\right)$$



Algorithm

Density Estimation:

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

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

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

…

$x_n \sim N(\mu_n, \sigma_n{}^2)$

The probability p(x) is given by:

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

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

Anomaly Detection Algorithm:

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

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

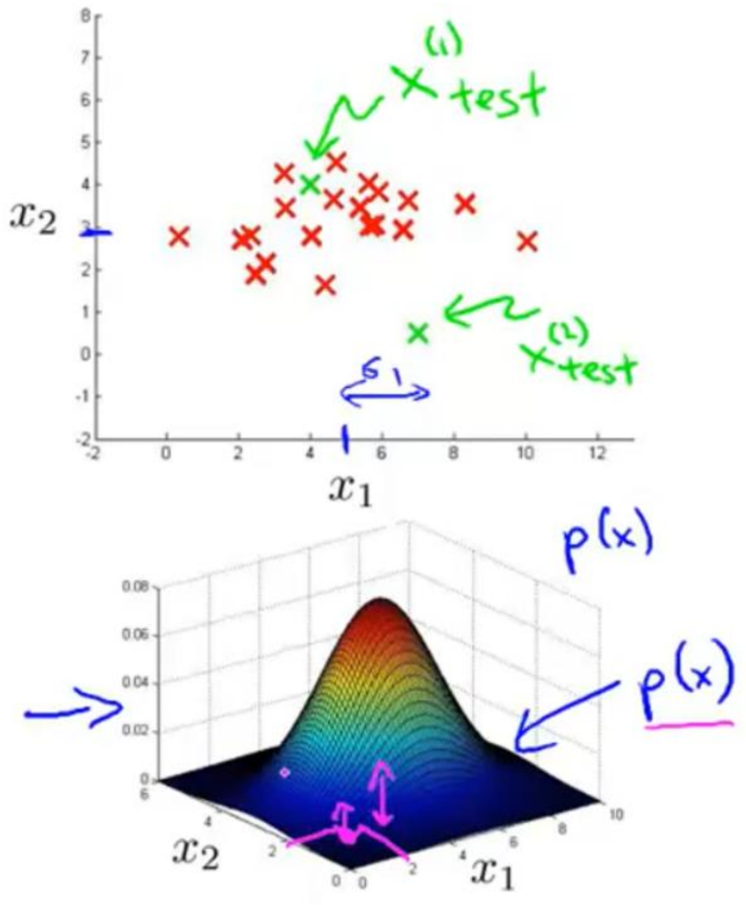
- $\mu_j = \frac{1}{m}\sum_{i=1}^{m} x_j^{(i)}$
- $\sigma_j^2 = \frac{1}{m}\sum_{i=1}^{m}(x_j^{(i)} - \mu_j)^2$

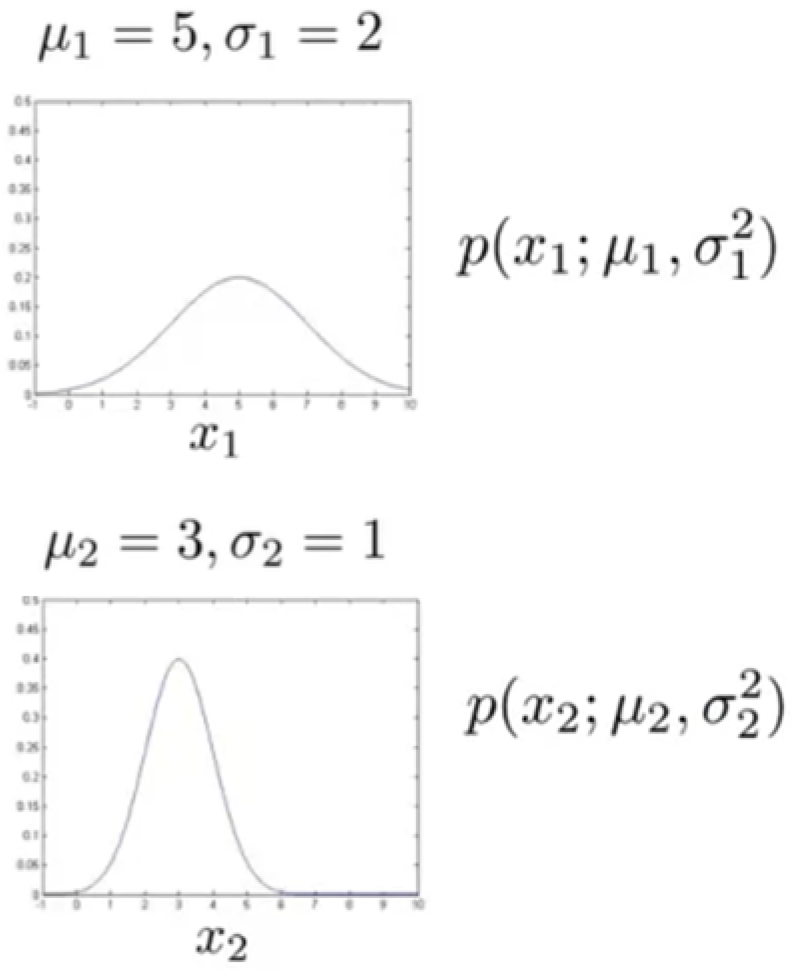
3. Given new example x, compute P(x):

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

4. Anomaly if $p(x) < \varepsilon$

Anomaly Detection Example:





$\varepsilon = 0.02$

- $p(x^{(1)}_{test}) = 0.0426 \geq \varepsilon \Rightarrow x^{(1)}_{test}$ is not anomalous
- $p(x^{(2)}_{test}) = 0.0021 < \varepsilon \Rightarrow x^{(2)}_{test}$ is anomalous

Given a training set $\{x^{(1)}, x^{(2)}, \ldots, x^{(m)}\}$, how would you estimate each μ_j and σ_j^2
(Note: $\mu_j \in R$, $\sigma_j^2 \in R$)

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

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

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

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

Building an Anomaly Detection System

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 labelled 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)
- Cross-validation set: $(x_{CV}^{(1)}, y_{CV}^{(1)}), \ldots (x_{CV}^{(mcv)}, y_{CV}^{(mcv)})$
- Test set: $\underbrace{(x_{test}^{(1)}, y_{test}^{(1)}), \ldots (x_{test}^{(mtest)}, y_{test}^{(mtest)})}_{y=1}$

Aircraft Engine Example:

Consider in a dataset, there are 10,000 good (normal) engines and 20 flawed (anomalous) engines with about 2-50 features for example.

This dataset could be divided as follows:

- Training set: 6,000 good engines (y=0)
- Cross-validation set: 2,000 good engines (y=0), 10 anomalous (y=1)
- Test set: 2,000 good engines (y=0), 10 anomalous (y=1)

Alternative Method:

- Training set: 6,000 good engines (y=0)
- Cross-validation set: 4,000 good engines (y=0), 10 anomalous (y=1)
- Test set: 4,000 good engines (y=0), 10 anomalous (y=1)

$\}$ Using same set of examples for cross-validation and test sets

GENERALLY NOT RECOMMENDED

Algorithm Evaluation:

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

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

- Possible evaluation metrics are:

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

We can also use cross-validation set to choose parameter ε

Suppose you have fit a model p(x). When evaluating on the cross validation set or test set, your algorithm predicts:

$$y = \begin{cases} 1 & \text{if } p(x) \le \epsilon \\ 0 & \text{if } p(x) > \epsilon \end{cases}$$

Is classification accuracy a good way to measure the algorithm's performance?

- Yes, because we have labels in the cross-validation/test sets.
- No, because we do not have labels in the cross-validation/test sets.
- No, because of skewed classes (so an algorithm that always predicts y=0 will have high accuracy).
- No for the cross-validation set; yes for the test set.

Anomaly Detection vs. Supervised Learning

Anomaly Detection	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.	
• Many different "types" of anomalies. Hard for any algorithm to learn from positive examples what anomalies look like; future anomalies may look nothing like any of the anomalous examples we've seen so far.	• 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.
• Fraud Detection	• Email Spam Classification
• Manufacturing (e.g. aircraft engines)	• Weather prediction (sunny, rainy, etc.)
• Monitoring machines in a data center …	• Cancer classification …

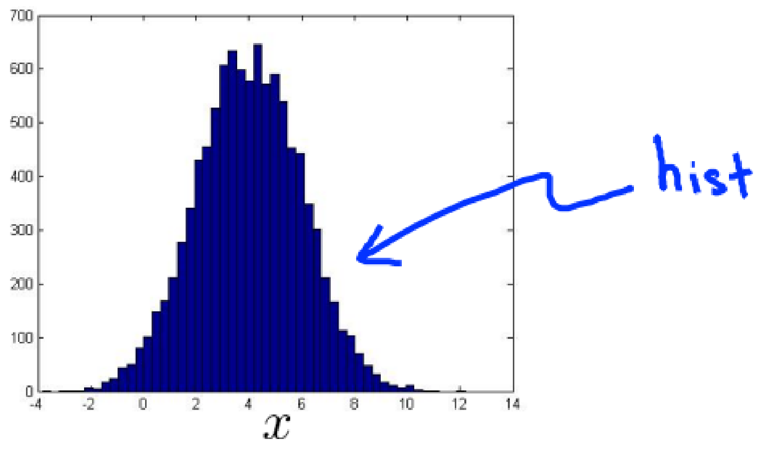
Which of the following problems would you approach with an anomaly detection algorithm (rather than a supervised learning algorithm)? Check all that apply.

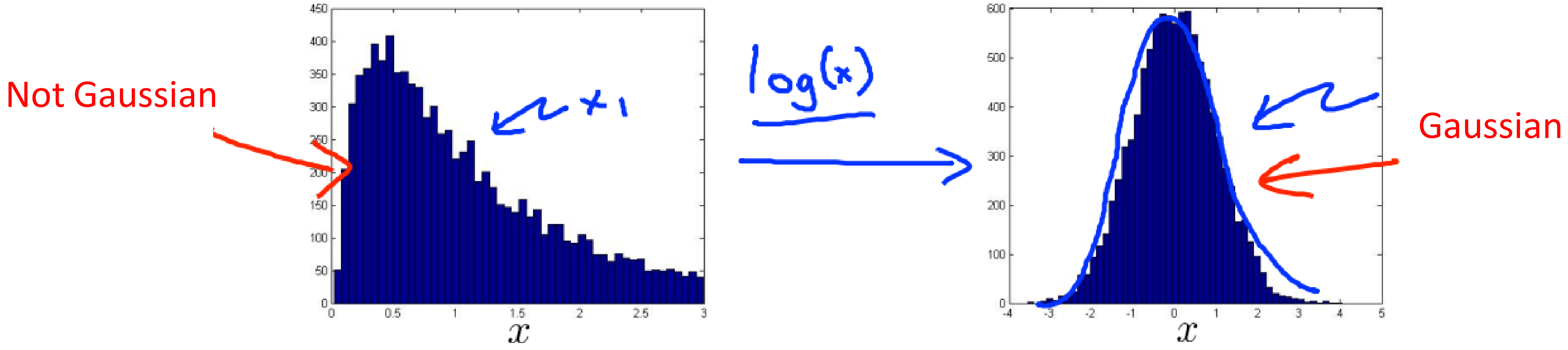
- You run a power utility (supplying electricity to customers) and want to monitor your electric plants to see if any one of them might be behaving strangely.
- You run a power utility and want to predict tomorrow's expected demand for electricity (so that you can plan to ramp up an appropriate amount of generation capacity).
- A computer vision/security application, where you examine video images to see if anyone in your company's parking lot is acting in an unusual way.
- A computer vision application, where you examine an image of a person entering your retail store to determine if the person is male or female.

Choosing What Features to Use

Non-Gaussian Features:

This plot is a histogram of some data which looks vaguely Gaussian.



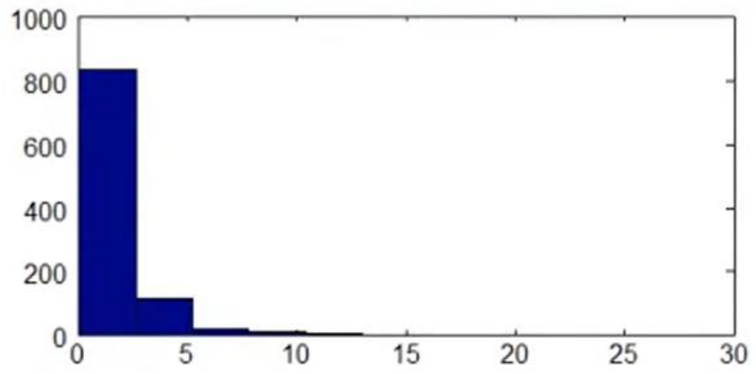


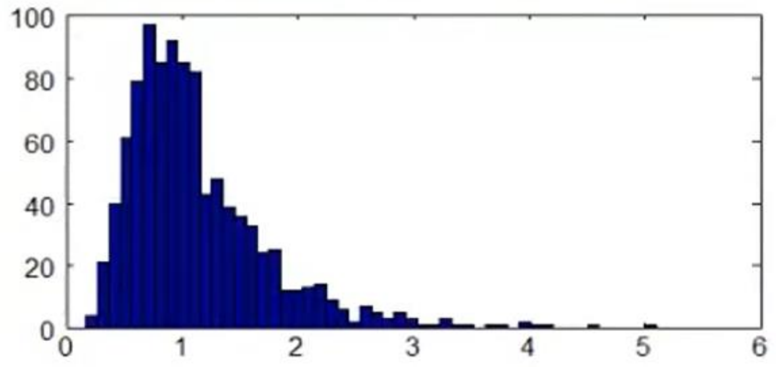
We apply transformation to this plot to make it look more Gaussian.

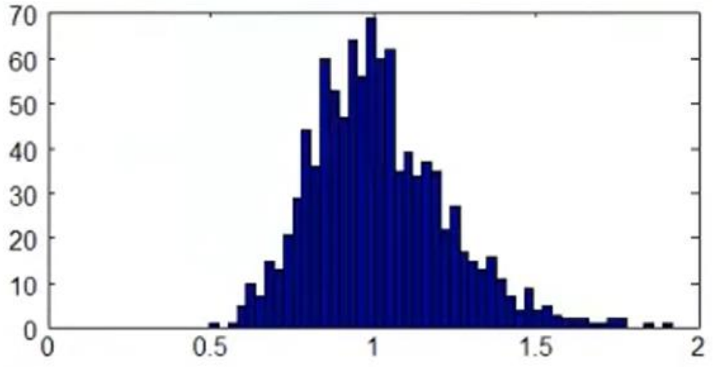
Some transformations that could be applied to make the plot look more Gaussian:

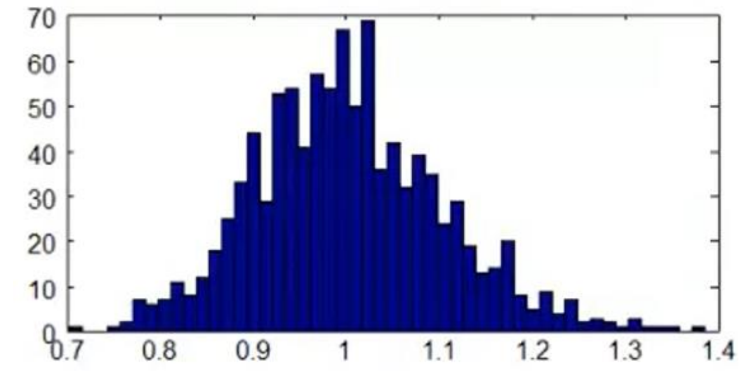
- $x \longrightarrow \log(x)$
- $x \longrightarrow \log(x+c)$
- $x \longrightarrow x^{1/2}$
- $x \longrightarrow x^{1/3}$

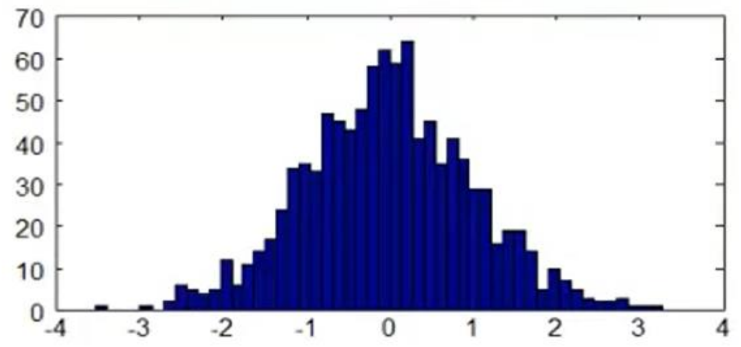
Example:


hist(x)


hist(x.^0.5, 50)


hist(x.^0.2, 50)


hist(x.^0.1, 50)


hist(log(x), 50)

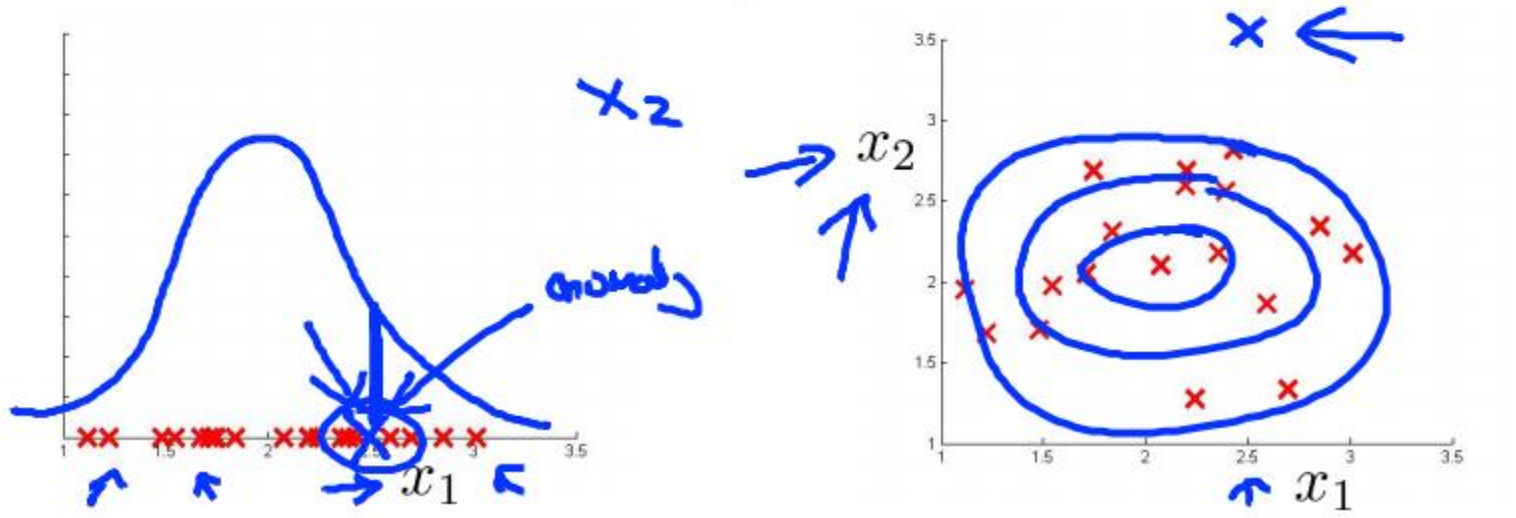
Error Analysis for Anomaly Detection:

We want –

- p(x) large for normal examples x.
- p(x) small for anomalous examples x.

Most common problem:

p(x) is comparable (say, both large) for normal and anomalous examples.



In other words, the probability of having normal and anomalous examples is similar.

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 = no. of disk accesses/sec

x_3 = CPU load

x_4 = network traffic

x_5 = CPU load/network traffic

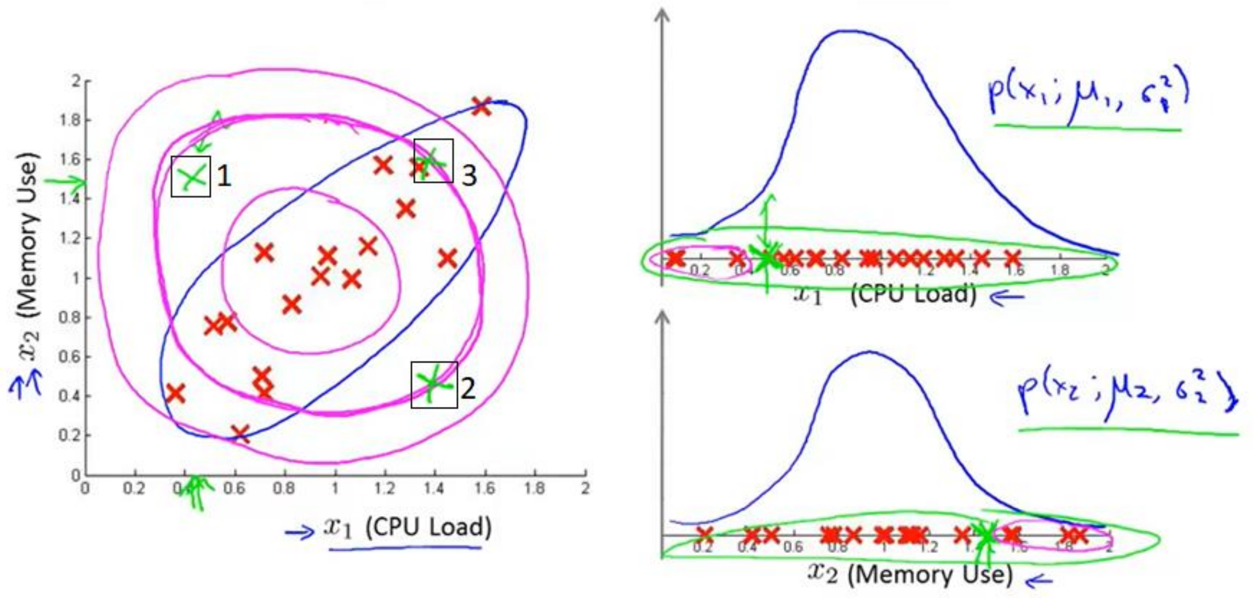
x_6 = (CPU load)^2/network traffic

Suppose your anomaly detection algorithm is performing poorly and outputs a large value of p(x) for many normal examples and for many anomalous examples in your cross validation dataset. Which of the following changes to your algorithm is most likely to help?

- Try using fewer features.
- Try coming up with more features to distinguish between the normal and the anomalous examples.
- Get a larger training set (of normal examples) with which to fit p(x).
- Try changing ε

Multivariate Gaussian Distribution

Example: Monitoring machines in a data center



- Consider the three examples in green cross marked as **1, 2** and **3** respectively.
- **"1"** is anomalous example compared to examples in red cross (normal examples) because for less CPU load, the memory use is high.
- The innermost ellipse has the highest probability of normal example and lowest for anomalous example and vice-versa.
- **"2"** and **"3"** lie on the same ellipse, yet probability of **"2"** being anomalous is more than that of **"3".**
- On the other hand, the bell curve shows that the green examples are **NOT AT ALL ANAMALOUS.**

- $p(x_1)$ and $p(x_2)$ when modelled separately showed no anomaly which is not true.
- In order to fix this, we develop a modified version of the anomaly detection algorithm using **"Multivariate Gaussian Distribution".**

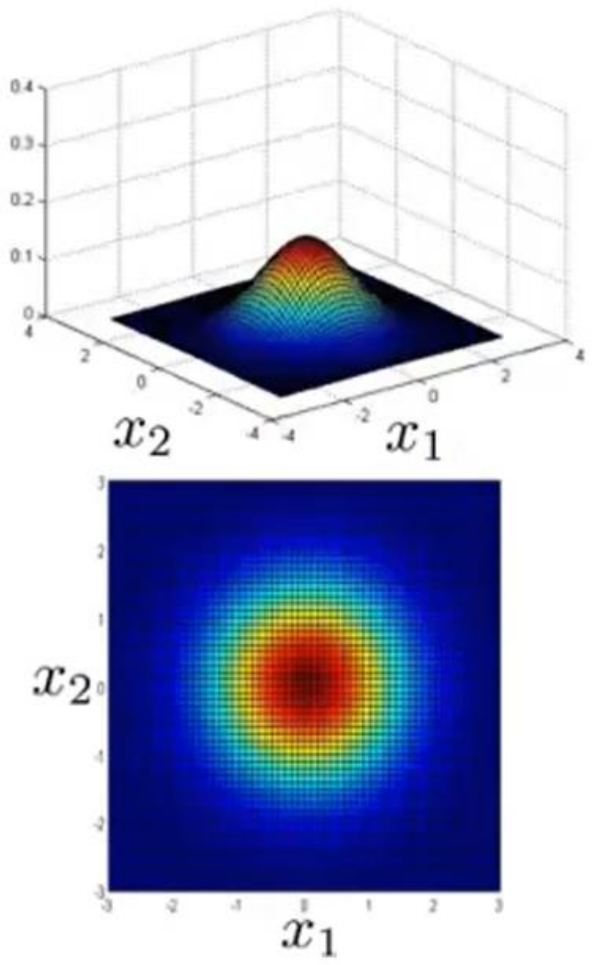
Multivariate Gaussian Distribution:

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

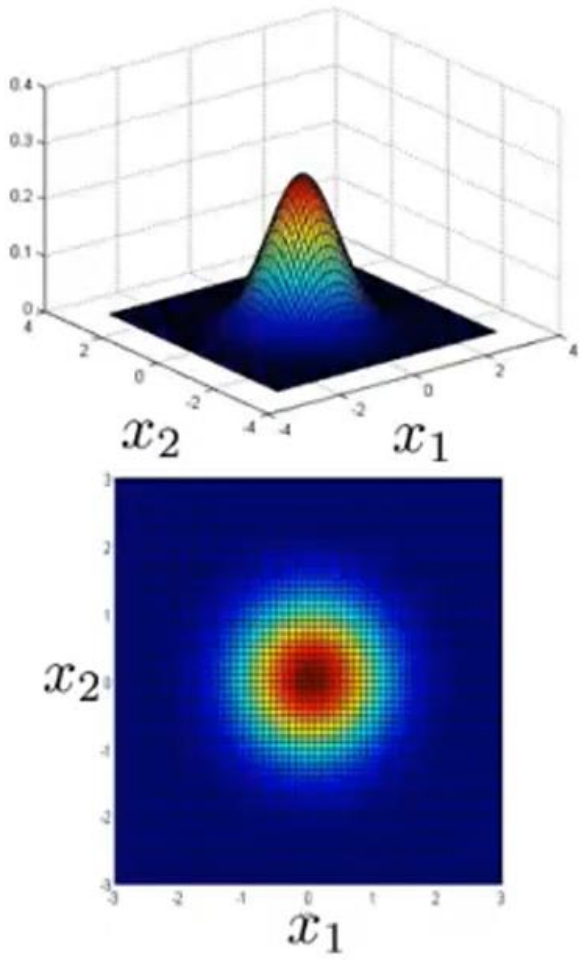
$$\boxed{p(x; \mu, \Sigma) = \frac{1}{(2\pi)^{\frac{n}{2}}.\,|\Sigma|^{\frac{1}{2}}}\ \exp\left(\frac{-1}{2}(x-\mu)^T\,\Sigma^{-1}(x-\mu)\right)}$$

Multivariate Gaussian (Normal) Examples:

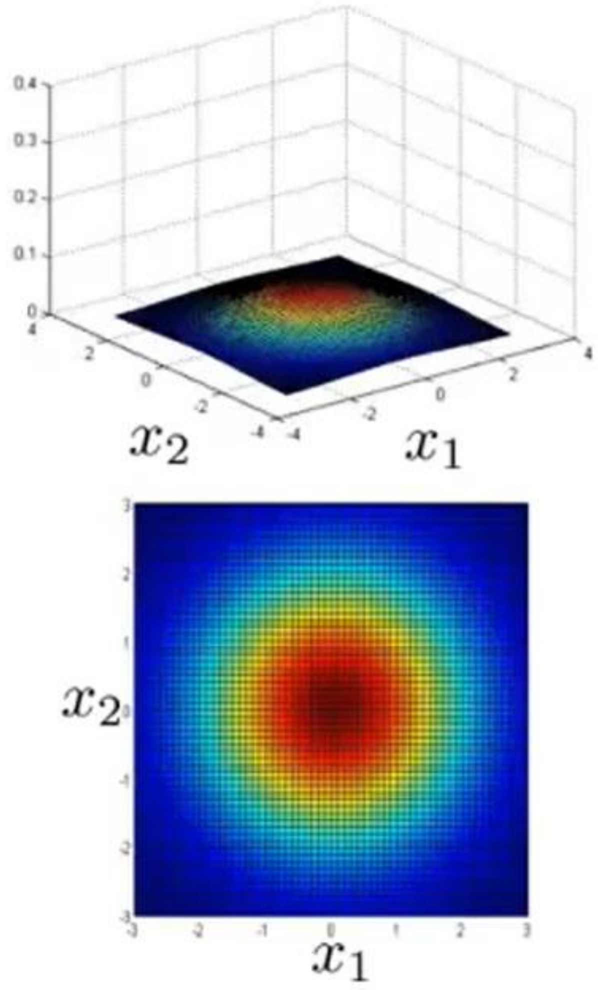
$$\mu = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \Sigma = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$



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

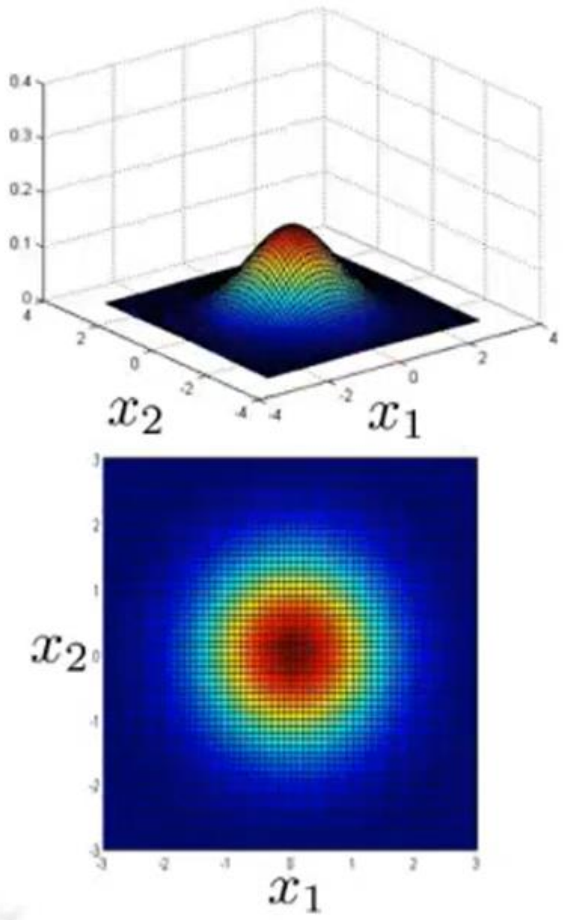


$$\mu = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \Sigma = \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix}$$

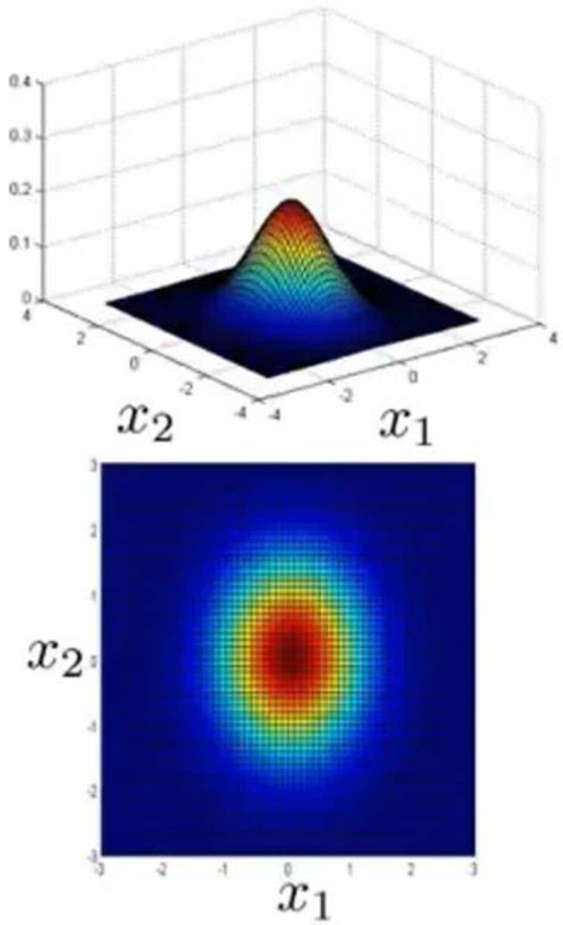


Multivariate Gaussian (Normal) Examples:

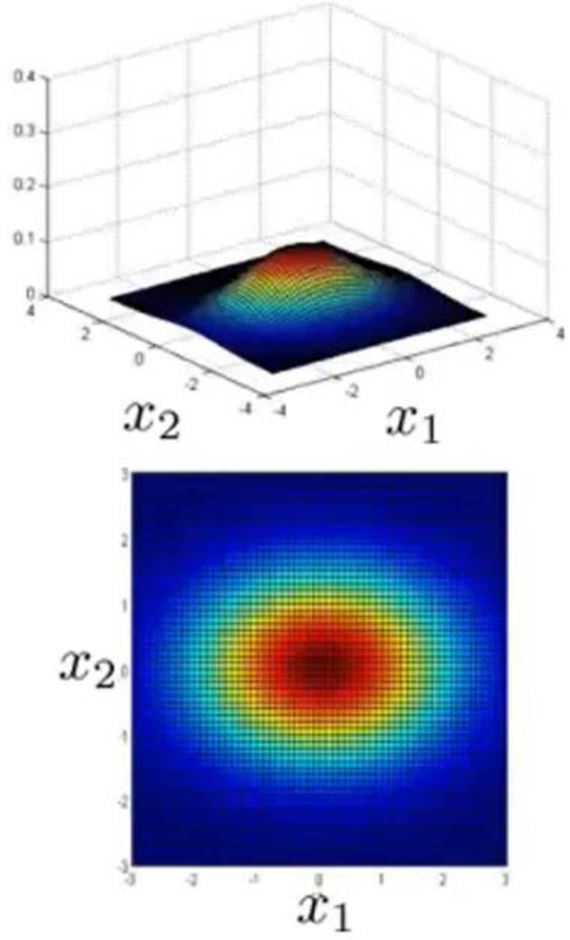
$$\mu = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \Sigma = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$



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

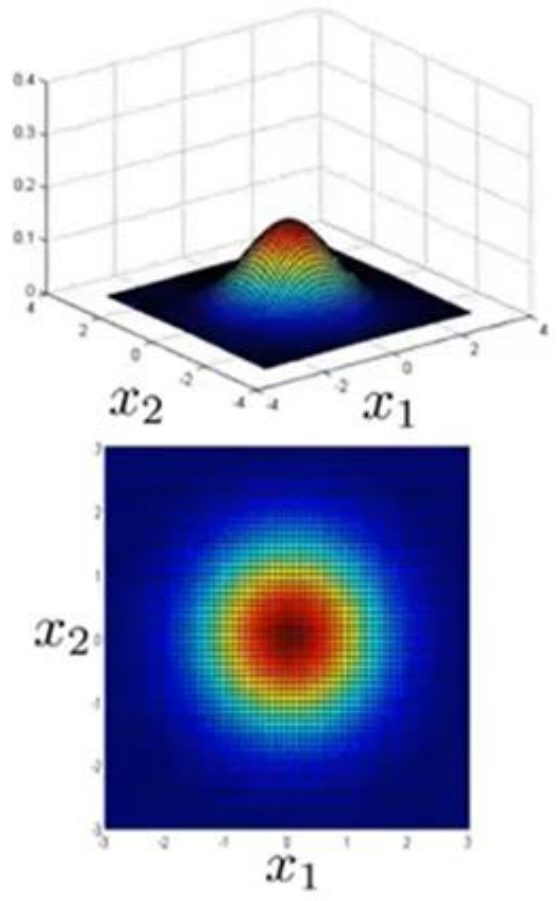


$$\mu = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \Sigma = \begin{bmatrix} 2 & 0 \\ 0 & 1 \end{bmatrix}$$

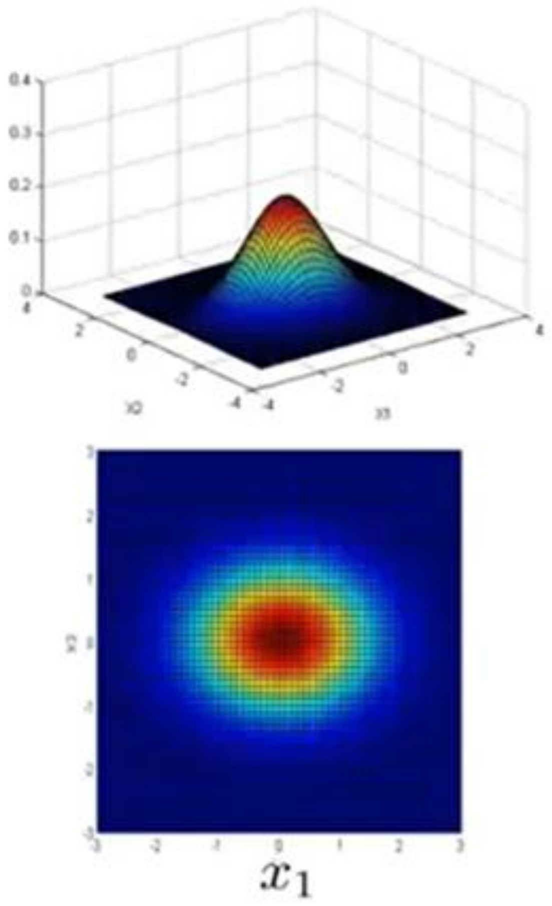


Multivariate Gaussian (Normal) Examples:

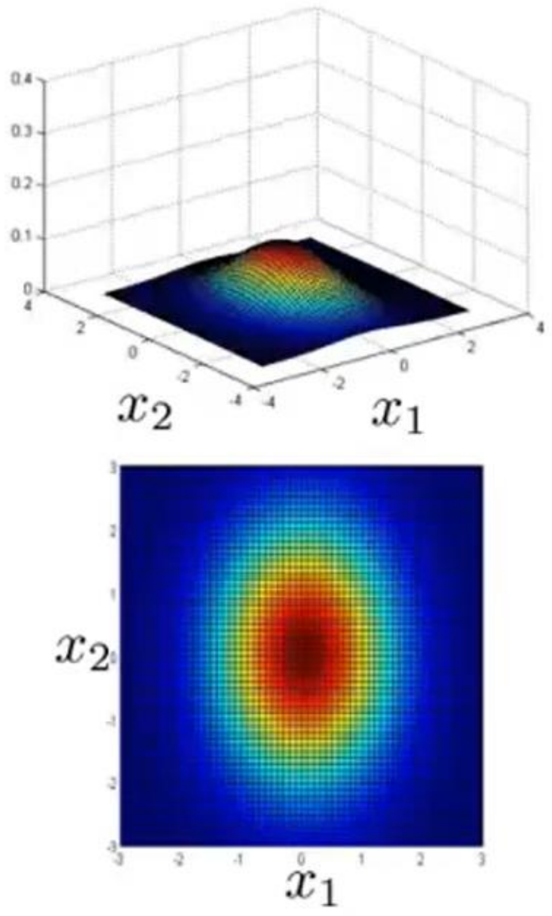
$\mu = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \Sigma = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$



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

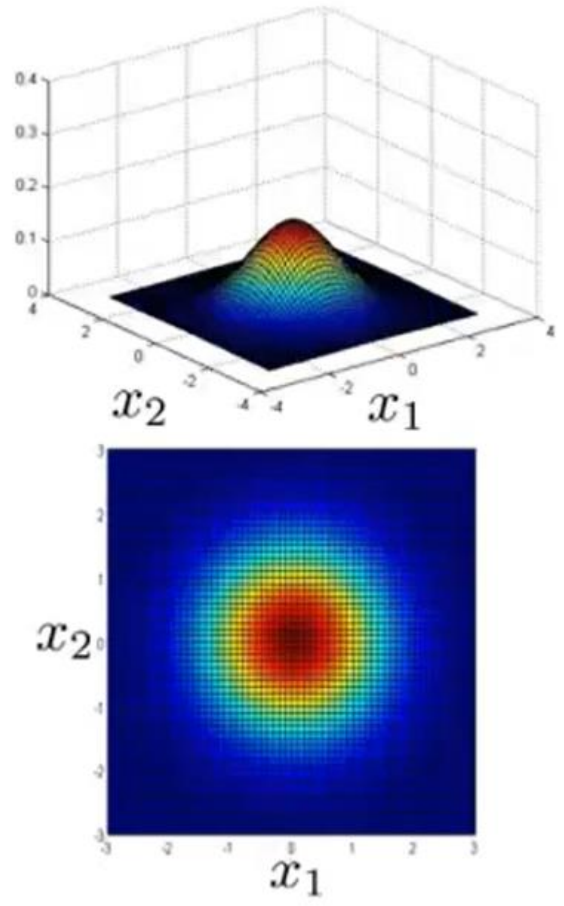


$\mu = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \Sigma = \begin{bmatrix} 1 & 0 \\ 0 & 2 \end{bmatrix}$

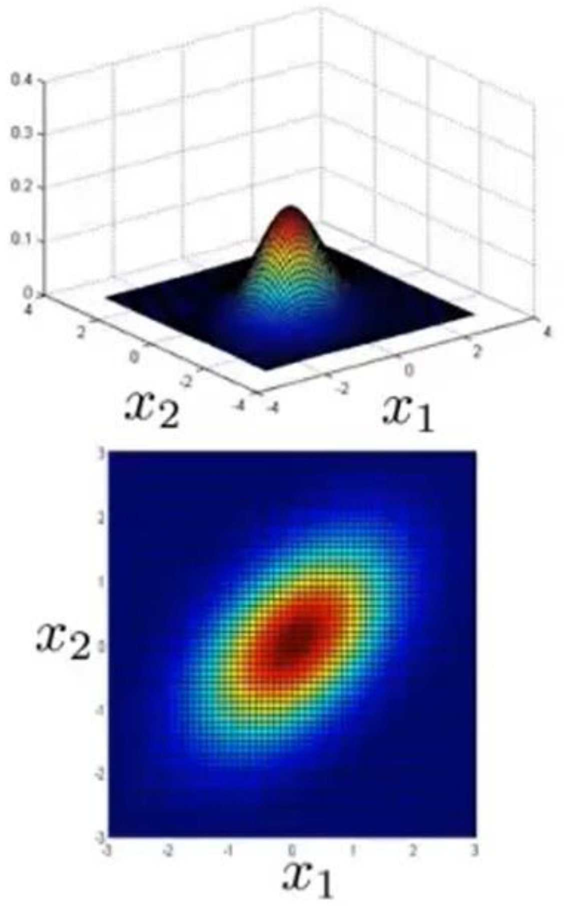


Multivariate Gaussian (Normal) Examples:

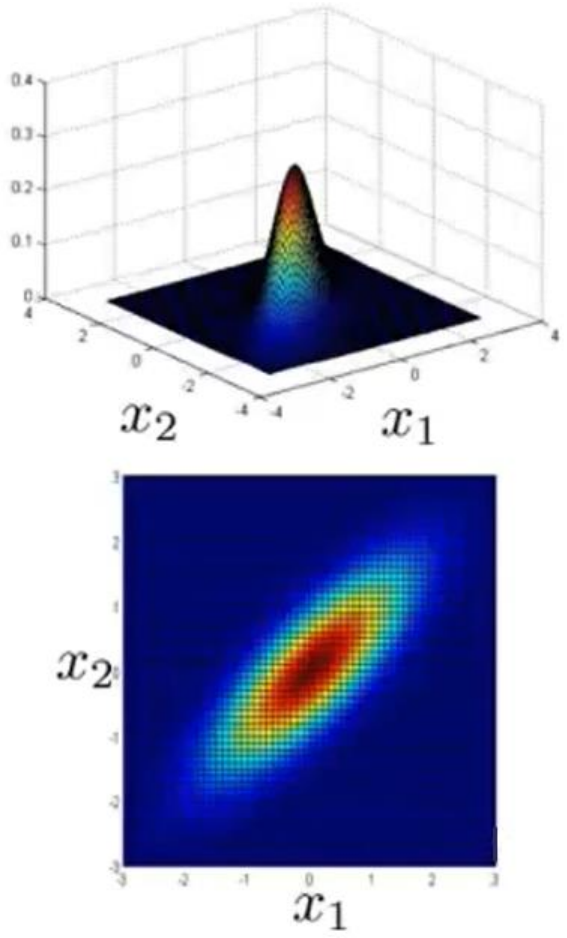
$$\mu = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \Sigma = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$



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



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

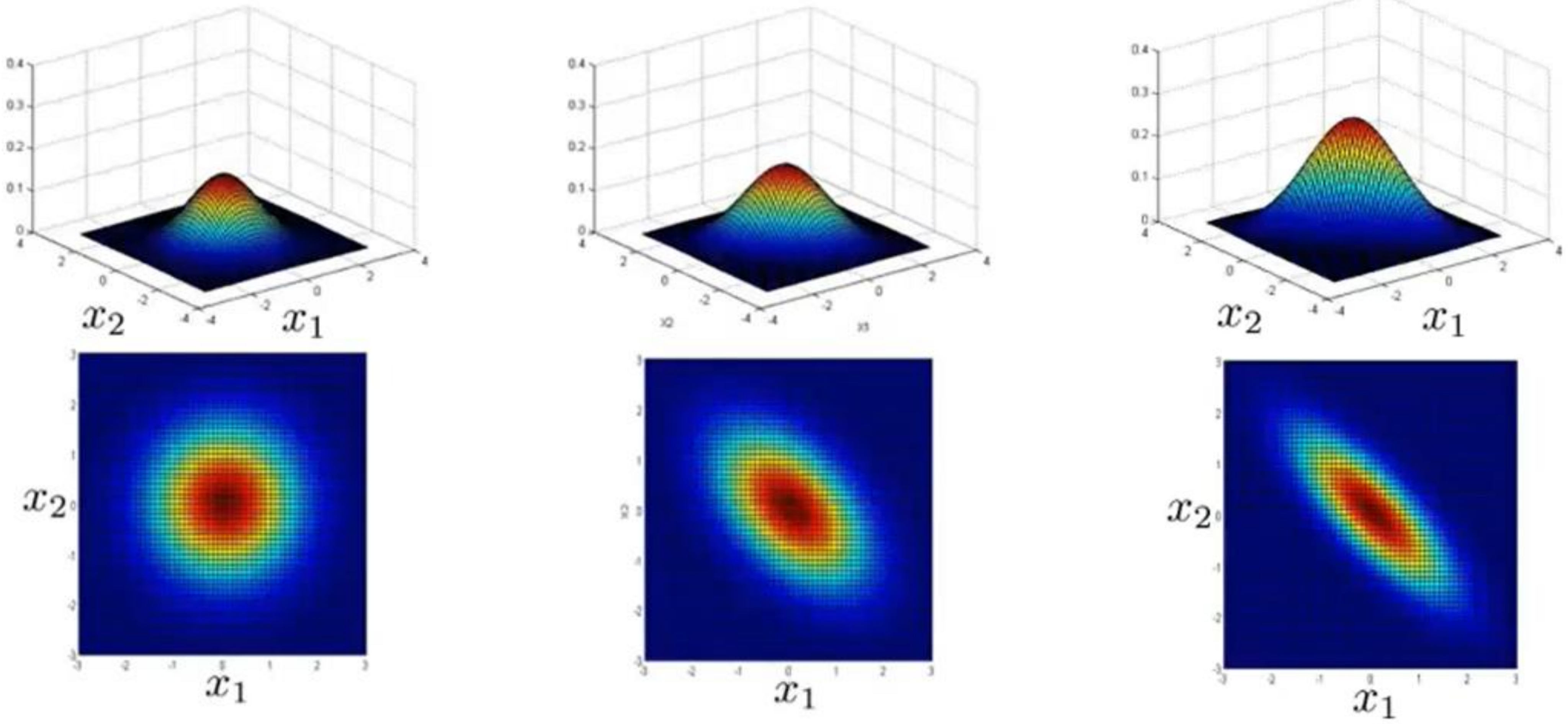


Multivariate Gaussian (Normal) Examples:

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

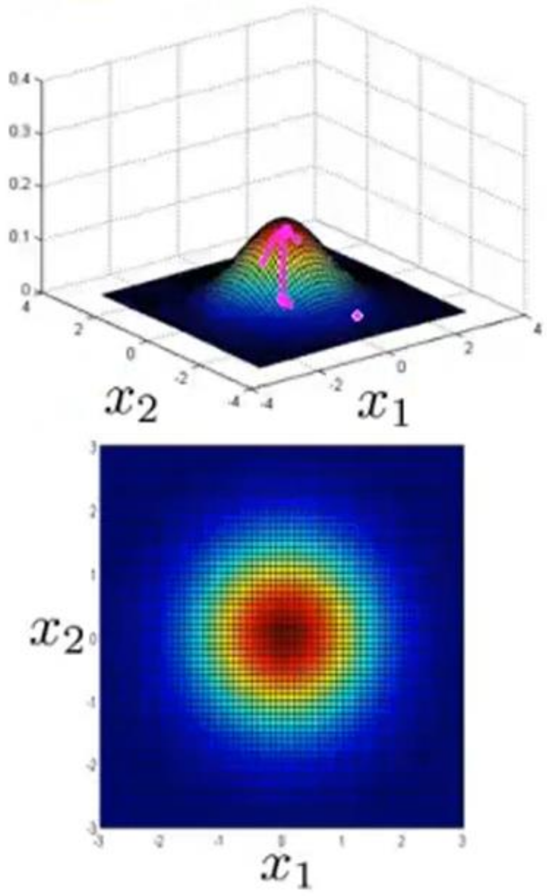
$\mu = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \Sigma = \begin{bmatrix} 1 & -0.5 \\ -0.5 & 1 \end{bmatrix}$

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

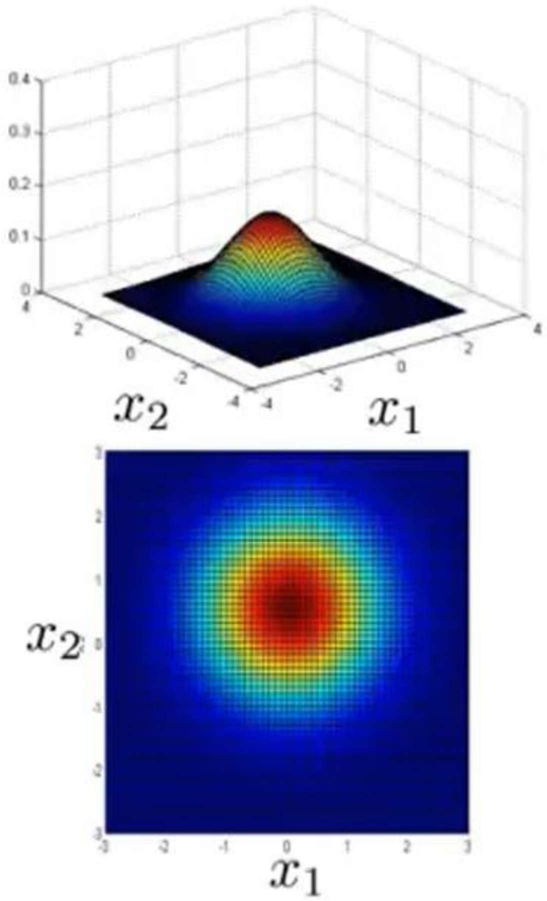


Multivariate Gaussian (Normal) Examples:

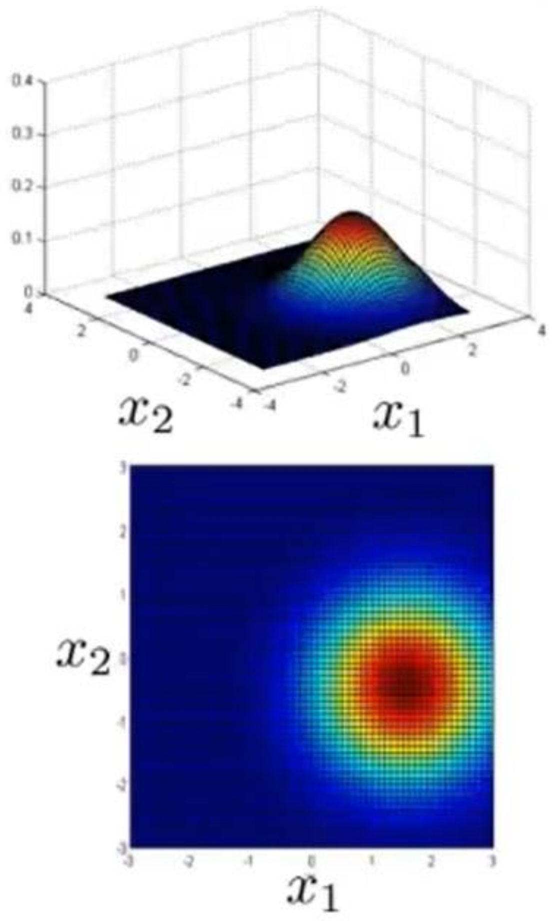
$$\mu = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \Sigma = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$



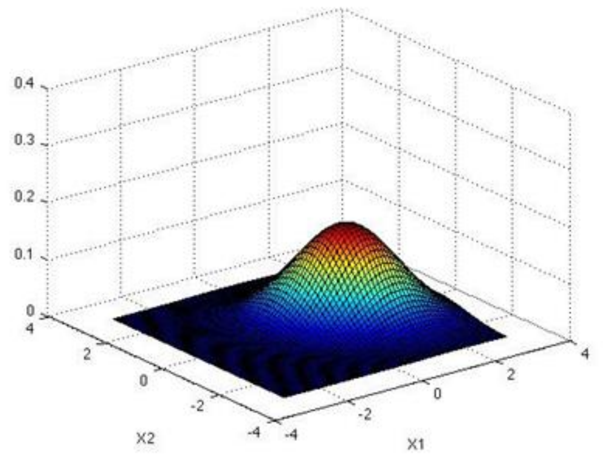
$$\mu = \begin{bmatrix} 0 \\ 0.5 \end{bmatrix} \Sigma = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

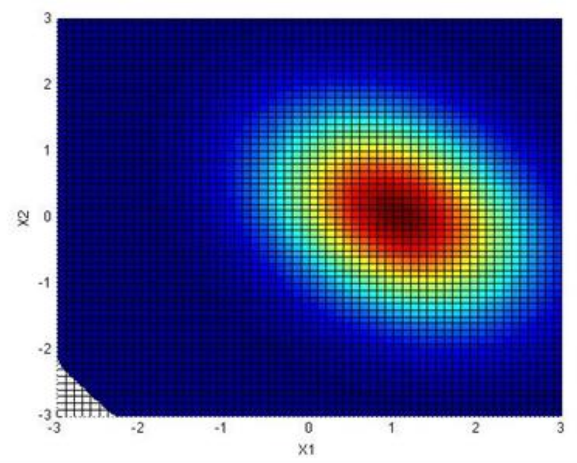


$$\mu = \begin{bmatrix} 1.5 \\ -0.5 \end{bmatrix} \Sigma = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$



Consider the following multivariate Gaussian:





Which of the following are the μ and Σ for this distribution?

$$\mu = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \Sigma = \begin{bmatrix} 1 & 0.3 \\ 0.3 & 1 \end{bmatrix}$$

$$\mu = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \Sigma = \begin{bmatrix} 1 & 0.3 \\ 0.3 & 1 \end{bmatrix}$$

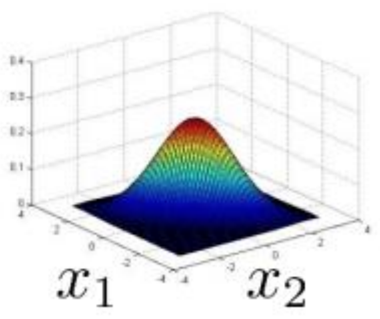
[✓ $\mu = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \Sigma = \begin{bmatrix} 1 & -0.3 \\ -0.3 & 1 \end{bmatrix}$]

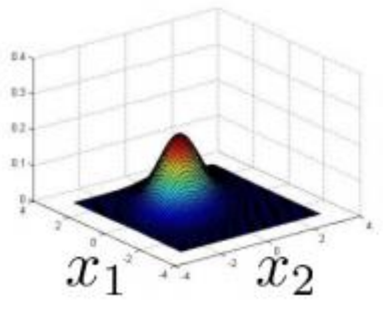
$$\mu = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \Sigma = \begin{bmatrix} 1 & -0.3 \\ -0.3 & 1 \end{bmatrix}$$

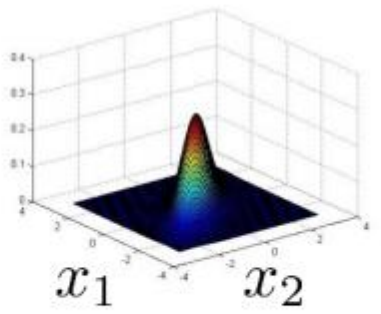
Anomaly Detection using Multivariate Gaussian Distribution

The parameters of multivariate Gaussian distribution are μ, Σ.

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







Parameter Fitting:

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

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

Anomaly Detection with multivariate Gaussian:

1. Fit model p(x) by setting

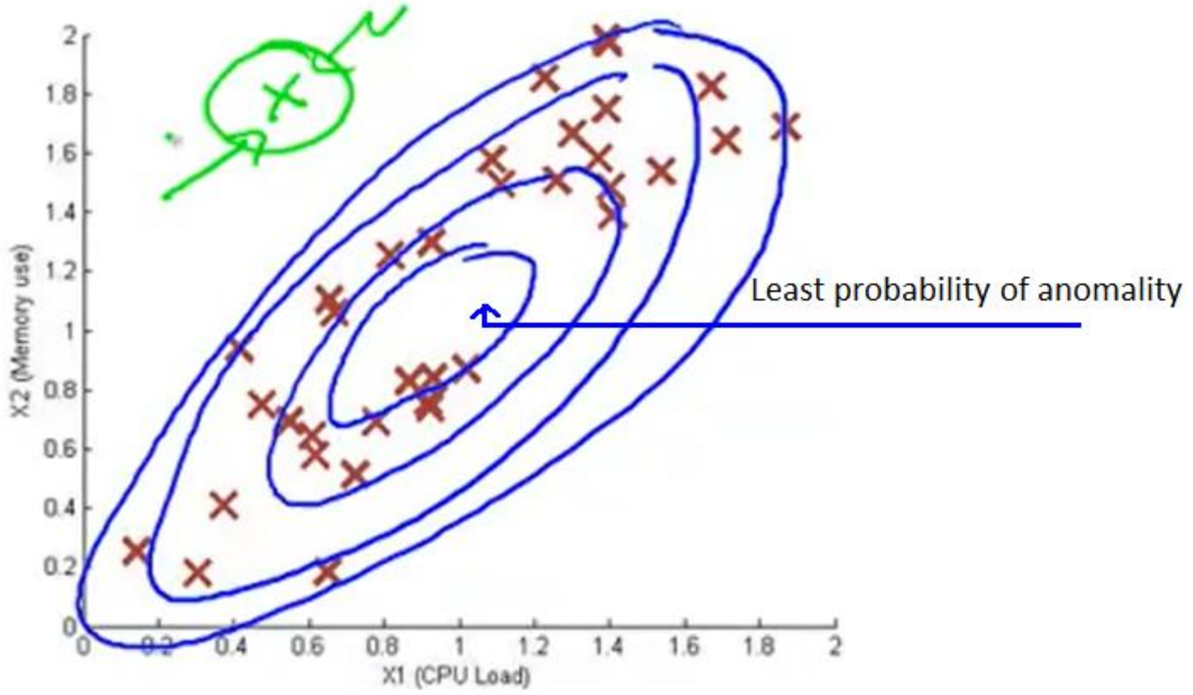
$$\mu = \frac{1}{m}\sum_{i=1}^{m} x^{(i)}$$

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

2. Given a new example x, compute

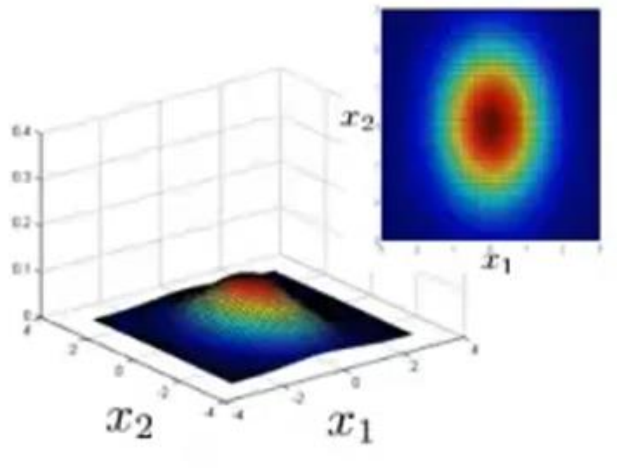
$$p(x) = \frac{1}{(2\pi)^{\frac{n}{2}}|\Sigma|^{\frac{1}{2}}}\exp\left(-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)\right)$$

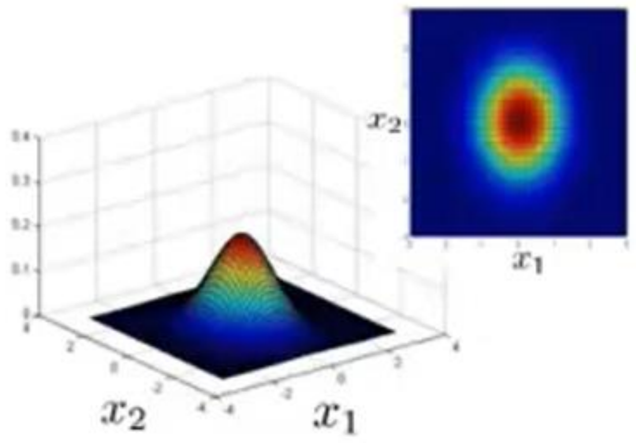
Flag an anomaly if p(x) < ε

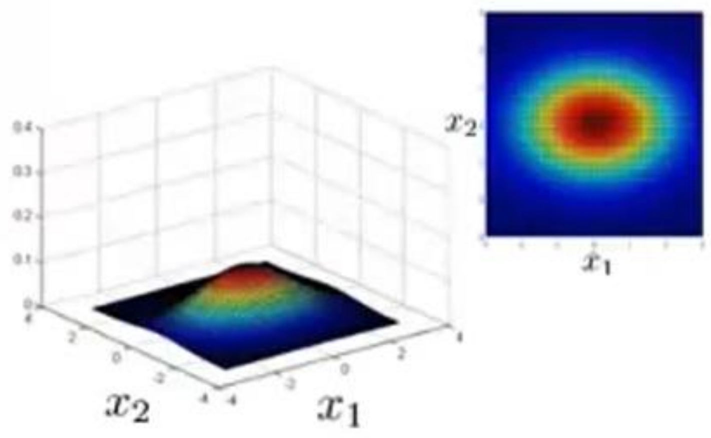


Relationship to original model:

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







Corresponds to multivariate Gaussian

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

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

Original Model vs. Multivariate Gaussian

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

Consider applying anomaly detection using a training set $\{x^{(1)}, x^{(2)}, \ldots , x^{(m)}\}$ where $x^{(i)} \in R^n$. Which of the following statements are true? Check all that apply.

- The original model $p(x_1 ; \mu_1, \sigma_1^2) \times \ldots \times p(x_n ; \mu_n, \sigma_n^2)$ corresponds to a multivariate Gaussian where the contours of $p(x; \mu, \Sigma)$ are axis-aligned.

- Using the multivariate Gaussian model is advantageous when m (the training set size) is very small ($m<n$).

- The multivariate Gaussian model can automatically capture correlations between different features in x.

- The original model can be more computationally efficient than the multivariate Gaussian model, and thus might scale better to very large values of n (number of features).

Recommender Systems

Example: Predicting Movie Ratings

User rates movies using 0 to 5 stars

Movie	Alice (1)	Bob (2)	Carol (3)	Dave (4)
The Godfather	5	5	0	0
The Fall of Max Payne	5	?	?	0
Avengers	?	4	0	?
Jurassic World	0	0	5	4
The Matrix	0	0	5	?

- n_u = no. of users
- n_m = no. of movies
- r(i, j) = 1 if user j has rated movie i
- $y^{(i, j)}$ = rating given by user j to move i (defined only if r(i, j) = 1)

In our notation, r(i, j) = 1 if user j has rated movie i, and $y^{(i, j)}$ is his rating on that movie. Consider the following example (no. of movies $n_m = 2$, no. of users $n_u = 3$):

	User 1	User 2	User 3
Movie 1	0	1	?
Movie 2	?	5	5

What is r(2, 1)? How about $y^{(2, 1)}$?

- r(2, 1) = 0, $y^{(2, 1)} = 1$
- r(2, 1) = 1, $y^{(2, 1)} = 1$
- r(2, 1) = 0, $y^{(2, 1)}$ = undefined
- r(2, 1) = 1, $y^{(2, 1)}$ = undefined

Content Based Recommendations

How do we predict the unknown values in the table?

Movie	**Alice (1)** $\theta^{(1)}$	**Bob (2)** $\theta^{(2)}$	**Carol (3)** $\theta^{(3)}$	**David (4)** $\theta^{(4)}$	x_1 **(thriller)**	x_2 **(action)**
The Godfather	5	5	0	0	0.9	0
The Fall of Max Payne	5	?	?	0	1.0	0.01
Avengers	?	4	0	?	0.99	0
Jurassic World	0	0	5	4	0.1	1.0
The Matrix	0	0	5	?	0	0.9

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

Example:

$$x^{(3)} = \begin{bmatrix} 1 \\ 0.99 \\ 0 \end{bmatrix}, \quad \theta^{(1)} = \begin{bmatrix} 0 \\ 5 \\ 0 \end{bmatrix} \quad \Rightarrow (\theta^{(1)})^T x^{(3)} = 5 \times 0.99 = 4.95$$

Note: By default, $x_0 = 1$

Consider the following set of movie ratings:

Movie	Alice (1)	Bob (2)	Carol (3)	David (4)	x_1 (thriller)	x_2 (action)
The Godfather	5	5	0	0	0.9	0
The Fall of Max Payne	5	?	?	0	1.0	0.01
Avengers	?	4	0	?	0.99	0
Jurassic World	0	0	5	4	0.1	1.0
The Matrix	0	0	5	?	0	0.9

Which of the following is a reasonable value for $\theta^{(3)}$? Recall that as $x_0 = 1$.

$$\theta^{(3)} = \begin{bmatrix} 0 \\ 5 \\ 0 \end{bmatrix}$$

$$\theta^{(3)} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

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

[✓ $\theta^{(3)} = \begin{bmatrix} 0 \\ 0 \\ 5 \end{bmatrix}$]

Problem Formulation:

- r(i, j) = 1 if user j has rated movie i (0 otherwise)
- $y^{(i, j)}$ = rating by user j on movie i (if defined)
- $\theta^{(j)}$ = parameter vector for user j
- $x^{(i)}$ = feature vector for movie i
- For user j, movie i, predicted rating: $(\theta^{(j)})^T(x^{(i)})$
- $m^{(j)}$ = no. of movies rated by user j

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

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

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

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

Optimization Algorithm:

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

Gradient Descent Update:

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

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

Collaborative Filtering

Collaborative Filtering is an algorithm that can start to learn for itself what features to use.

Suppose we have the following dataset but have no idea how thriller or action-packed the movies are.

Movie	Alice (1) $\theta^{(1)}$	Bob (2) $\theta^{(2)}$	Carol (3) $\theta^{(3)}$	David (4) $\theta^{(4)}$	x_1 (thriller)	x_2 (action)
The Godfather	5	5	0	0	?	?
The Fall of Max Payne	5	?	?	0	?	?
Avengers	?	4	0	?	?	?
Jurassic World	0	0	5	4	?	?
The Matrix	0	0	5	?	?	?

Now suppose we have gone to each of our users, and each of our users has told us how much they like thriller movies and how much they like action-packed movies

$$\theta^{(1)} = \begin{bmatrix} 0 \\ 5 \\ 0 \end{bmatrix}, \theta^{(2)} = \begin{bmatrix} 0 \\ 5 \\ 0 \end{bmatrix}, \theta^{(3)} = \begin{bmatrix} 0 \\ 0 \\ 5 \end{bmatrix}, \theta^{(4)} = \begin{bmatrix} 0 \\ 0 \\ 5 \end{bmatrix} \xrightarrow{\text{Find } x^{(1)} \text{ such that}} \boxed{\begin{array}{l} (\theta^{(1)})^T x^{(1)} \approx 5 \\ (\theta^{(2)})^T x^{(1)} \approx 5 \\ (\theta^{(3)})^T x^{(1)} \approx 0 \\ (\theta^{(4)})^T x^{(1)} \approx 0 \end{array}}$$

Optimization Algorithm:

Given $\theta^{(1)}$, $\theta^{(2)}$, … , $\theta^{(n_u)}$, to learn $x^{(i)}$:

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

Given $\theta^{(1)}$, $\theta^{(2)}$, … , $\theta^{(n_u)}$, to learn $x^{(1)}$, … , $x^{(n_m)}$

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

Collaborative Filtering:

- Given $x^{(1)}, \ldots, x^{(n_m)}$ (and movie ratings), we can estimate $\theta^{(1)}, \theta^{(2)}, \ldots, \theta^{(nu)}$
- Given $\theta^{(1)}, \theta^{(2)}, \ldots, \theta^{(nu)}$, we can estimate $x^{(1)}, \ldots, x^{(nm)}$

We first make an initial random guess for θ and then go ahead with the procedure in order to learn about the features for different movies.

Now that we have some initial set of features, we try to get a better estimate for our parameters θ.

Now, we have a better setting of our parameter θ for our users which can be used to get a better set of features.

We repeat this procedure until we get a better set of features and parameter θ.

$$\text{Initial guess for } \theta \longrightarrow x \longrightarrow \theta \longrightarrow x \longrightarrow \theta \longrightarrow x \longrightarrow \ldots \text{ and so on}$$

$$\xrightarrow{\hspace{8em}}$$

Improvement in θ and x

Consider the following movie ratings:

	User 1	User 2	User 3	(romance)
Movie 1	0	1.5	2.5	?

Note that there is only one feature x_1. Suppose that:

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

What would be a reasonable value for $x_1^{(1)}$ (the value denoted "?" in the table above)?

- 0.5
- 1
- 2
- Any of these values would be equally reasonable

Suppose you use gradient descent to minimize:

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

Which of the following is a correct gradient descent update rule for i ≠ 0?

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

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

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

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

Collaborative Filtering Optimization Objective

In this algorithm, instead of going back and forth and minimizing w.r.t θ and then minimizing w.r.t x, we simultaneously minimize w.r.t both θ and x.

- Given $x^{(1)}, \dots, x^{(nm)}$, estimate $\theta^{(1)}, \dots, \theta^{(nu)}$:

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

- Given $\theta^{(1)}, \dots, \theta^{(nu)}$, estimate $x^{(1)}, \dots, x^{(nm)}$:

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

- Minimizing $x^{(1)}, \dots, x^{(nm)}$ and $\theta^{(1)}, \dots, \theta^{(nu)}$ simultaneously:

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

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

Collaborative Filtering Algorithm

1. Initialize $x^{(1)}, \dots, x^{(nm)}, \theta^{(1)}, \dots, \theta^{(nu)}$ to some random values.

2. Minimize $J(x^{(1)}, \dots, x^{(nm)}, \theta^{(1)}, \dots, \theta^{(nu)})$ using gradient descent (or an advanced optimization algorithm).

E.g. for every $j = 1, \dots, n_u$, $i = 1, \dots, n_m$:

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

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

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

In the algorithm we described, we initialized $x^{(1)}, \dots, x^{(nm)}$ and $\theta^{(1)}, \theta^{(2)}, \dots, \theta^{(nu)}$ to small random values. Why is this?

- This step is optional. Initializing to all 0's would work just as well.
- Random initialization is always necessary when using gradient descent on any problem.
- This ensures that $x^{(i)} \neq \theta^{(j)}$ for any i, j
- This serves as symmetry breaking (similar to the random initialization of a neural network's parameters) and ensures the algorithm learns features $x^{(1)}, \dots, x^{(nm)}$ that are different from each other.

Vectorization: Low Rank Matrix Factorization

Collaborative Filtering:

Consider the following dataset.

Movie	Alice (1)	Bob (2)	Carol (3)	Dave (4)
The Godfather	5	5	0	0
The Fall of Max Payne	5	?	?	0
Avengers	?	4	0	?
Jurassic World	0	0	5	4
The Matrix	0	0	5	?

Take all the ratings by all the users and group them into a matrix.

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

Collaborative Filtering:

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

Predicted ratings:

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

Given the matrix of predicted ratings, there is vectorized way of writing this out called **"Low Rank Matrix Vectorization"**

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

Low Rank Matrix Vectorization: $X\Theta^T$

Finding related movies:

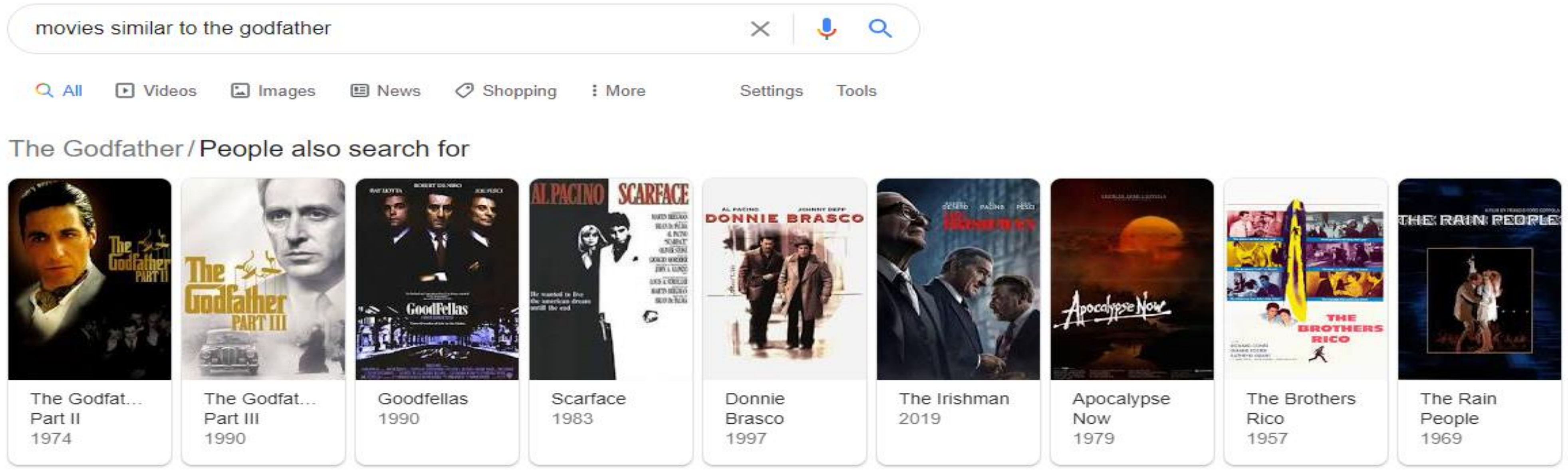
For each product i, we learn a feature vector $x^{(i)} \in R^n$.

Example: x_1 = romance, x_2 = action, x_3 = comedy, etc.

Suppose a user has watched some movie i and we have to find movies j related to movie i.
How do we do that?

Answer: Now that we have learned these feature vectors, this gives us a very convenient way to measure how similar two movies are. In particular, movie i has a feature vector $x^{(i)}$ and so if you can find a different movie, j, so that the distance between $x^{(i)}$ and $x^{(j)}$ is small i.e. $\|x^{(i)} - x^{(j)}\|$ is small, then this is a pretty strong indication that, movies j and i are somehow similar.

Example: Find movies j with the smallest $\|x^{(i)} - x^{(j)}\|$ similar to **"The Godfather"**



$$\text{Let } X = \begin{bmatrix} - & (x^{(1)})^T & - \\ & \vdots & \\ - & (x^{(n_m)} & - \end{bmatrix}, \ \Theta = \begin{bmatrix} - & (\theta^{(1)})^T & - \\ & \vdots & \\ - & (\theta^{(n_u)} & - \end{bmatrix}.$$

What is another way of writing the following:

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

- $X\Theta$
- $X^T\Theta$
- $X\Theta^T$
- $\Theta^T X^T$

Implementational Detail: Mean Normalization

Users who have not rated any movie:
Consider the following dataset. We have a new user “Eve” who has not rated any movie yet.

Movie	Alice (1)	Bob (2)	Carol (3)	Dave (4)	Eve (5)
The Godfather	5	5	0	0	?
The Fall of Max Payne	5	?	?	0	?
Avengers	?	4	0	?	?
Jurassic World	0	0	5	4	?
The Matrix	0	0	5	?	?

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

Let’s see what the collaborative filtering algorithm will do on this user.

Collaborative Filtering algorithm on Eve:

Let no. of features (n) = 2. We want to a learn parameter vector $\theta^{(5)}$

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

Zero since user has not rated any movie

Zero

This is the only term that affects $\theta^{(5)}$

$$\min\left(\frac{\lambda}{2}[(\theta_1^{(5)})^2 + (\theta_2^{(5)})^2]\right) \Longrightarrow \theta^{(5)} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

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

This means that, **Eve will rate every single movie as “0”.**

Movie	Eve (5)
The Godfather	0
The Fall of Max Payne	0
Avengers	0
Jurassic World	0
The Matrix	0

This doesn’t make sense because other users have rated the movies with 5 stars, 4 stars, etc.

Mean Normalization:

In order to fix this issue, we use Mean Normalization method so that Eve does not rate all movies with 0 stars.

- For the matrix Y, compute the mean matrix μ.

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

- Update Y as Y : = Y - μ

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

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

Hence, for Eve (User 5), predict: $(\theta^{(5)})^T x^{(i)} + \mu_i$

Since $\theta^{(5)} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$, $(\theta^{(5)})^T x^{(i)} = 0$

This means, the predicted movie rating by Eve is μ_i

Hence, instead of rating all movies with 0 stars, Eve will predict the movies as:

Movie	Eve (5)
The Godfather	2.5
The Fall of Max Payne	2.5
Avengers	2
Jurassic World	2.25
The Matrix	1.25

We talked about mean normalization. However, unlike some other applications of feature scaling, we did not scale the movie ratings by dividing by the range (max – min value). This is because:

- This sort of scaling is not useful when the value being predicted is real-valued.
- All the movie ratings are already comparable (e.g. 0 to 5 stars), so they are already on similar cases.
- Subtracting the mean is mathematically equivalent to dividing by the range.
- The makes the overall algorithm significantly more computationally efficient.