

Anomaly Detection meaning and Example:

Suppose you run an aircraft engine manufacturing factory. Now, consider that there is a batch of aircraft engines ready. Though before sending them to the assembly unit, you want to test whether they work properly or not. Now, we can have features x_1, x_2, \dots, x_n as heat generated by the engine, vibration intensity of the engine etc.

Suppose there is only x_1 and x_2 present. Now we have some already plotted some data points for x_1 and x_2 values as red crosses. The red crosses together represent a region of already tested engines, which means if a new data point falls in this region, then the probability of the engine being ok is significant, otherwise it is not. The x_1 and x_2 values of new points /new engine are recorded and plotted on the x_1 vs x_2 curve. If a point is in or near the middle of the red crosses, that engine is ok, if a point is too far from the red cross cluster, it means that the probability of that new engine being ok is really less.

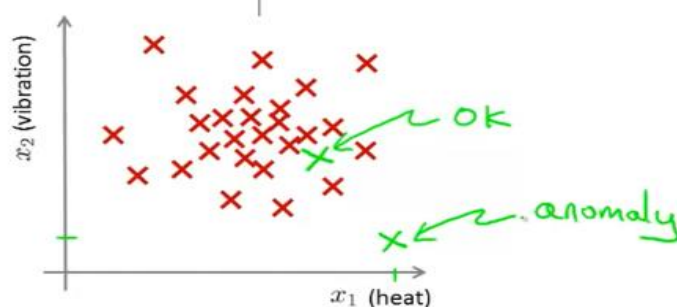
Anomaly detection example

Aircraft engine features:

- x_1 = heat generated
- x_2 = vibration intensity
- ...

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

New engine: x_{test}



Andrew Ng

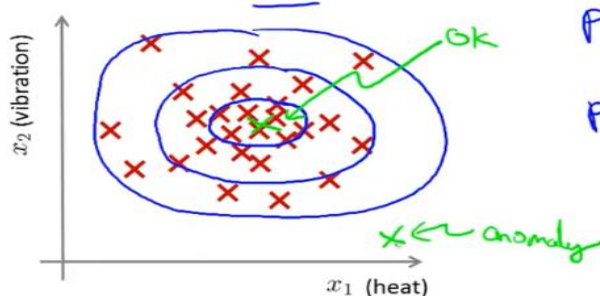
We selected a threshold probability value called epsilon and the new engines for whom probability of being in the region of the red cross cluster is greater than or equal to epsilon are classified as ok, while the other engines are classified as an anomaly. This is illustrated below:

Density estimation

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

→ Is x_{test} anomalous?

Model $p(x)$.



$p(x_{test}) < \epsilon \rightarrow \text{flag anomaly}$
 $p(x_{test}) \geq \epsilon \rightarrow \text{OK}$

Andrew Ng

More examples of anomaly detection :

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 $p(x) < \epsilon$

→ 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 = CPU load/network traffic.

...

$p(x) < \epsilon$

x_1
 x_2
 x_3
 x_4 $p(x)$

⏪ ⏩ ⏴ ⏵

Andrew Ng

If we set some features of computers as x_1, x_2, \dots, x_n and set a value of threshold probability as epsilon, we can track any unusual activity if a computer shows $p(x) < \epsilon$.

Question:

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 .

Correct

Explanation of the above is that, the epsilon value is incorrectly high because most of the $p(x)$ values, even the ones that are well within being OK range are less than it. Hence, we need to decrease epsilon.

Gaussian/Normal Distribution:

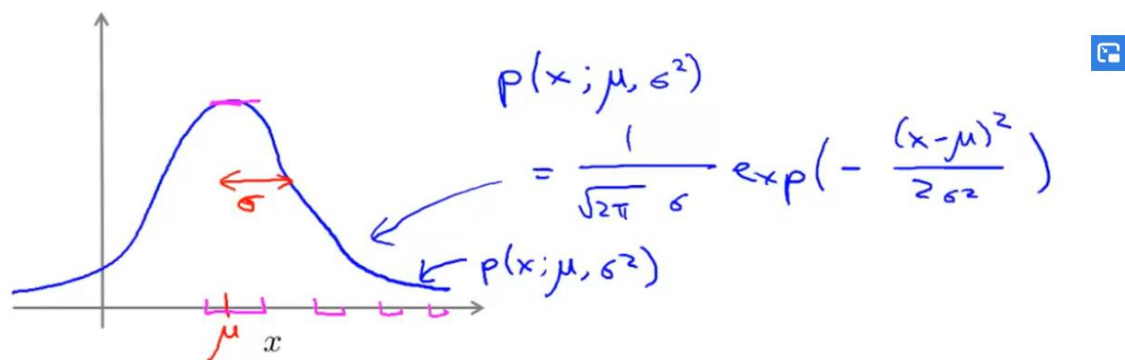
Gaussian (Normal) distribution

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



Andrew Ng

The spread of a normal curve is given by its standard deviation.

The height of a normal distribution (y) can be defined by its corresponding value of x (refer to Figure 2) by the following equation:

$$y = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

- where:
- y = vertical height of a point on the normal distribution
 - x = distance along the horizontal axis
 - σ = standard deviation of the data distribution
 - μ = mean of the data distribution
 - e = exponential constant = 2.71828...
 - π = pi = 3.14159....

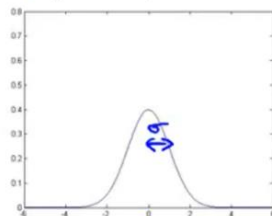
How Mean and Standard Deviation influence the shape of a Normal Curve:

JOJI

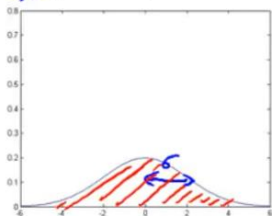
FILE

Gaussian distribution example

→ $\mu = 0, \sigma = 1$

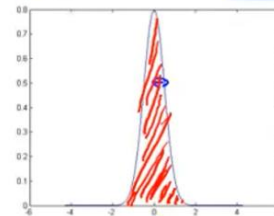


→ $\mu = 0, \sigma = 2$

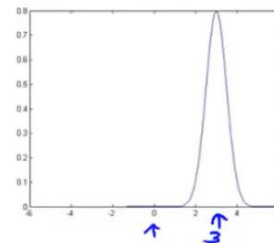


→ $\mu = 0, \sigma = 0.5$

$\sigma^2 = 0.25$



→ $\mu = 3, \sigma = 0.5$



Andrew Ng

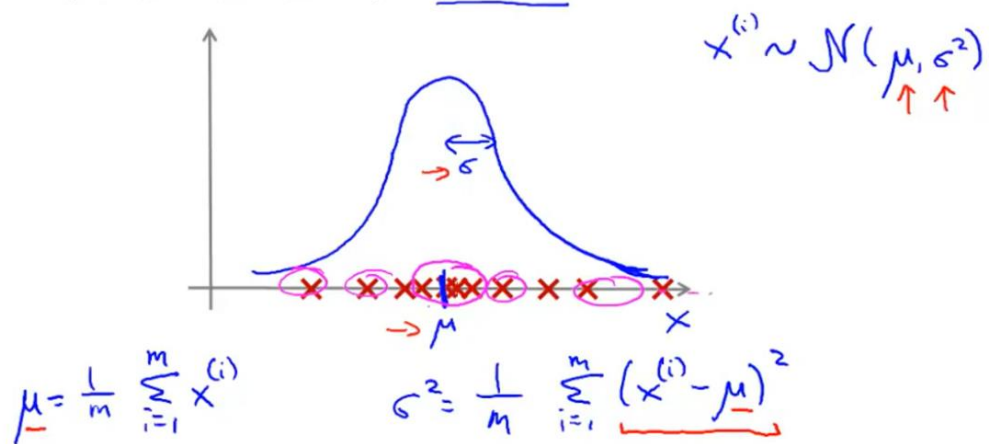
MEAN AND STANDARD DEVIATION:

JOJI

FILE

Parameter estimation

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



Andrew Ng

ANOMALY DETECTION ALGORITHM:

Density estimation

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

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

$$\begin{aligned}
 p(x) &= p(x_1; \mu_1, \sigma_1^2) p(x_2; \mu_2, \sigma_2^2) p(x_3; \mu_3, \sigma_3^2) \dots p(x_n; \mu_n, \sigma_n^2) \leftarrow \\
 &= \prod_{j=1}^n p(x_j; \mu_j, \sigma_j^2)
 \end{aligned}$$

$x_1 \sim \mathcal{N}(\mu_1, \sigma_1^2)$
 $x_2 \sim \mathcal{N}(\mu_2, \sigma_2^2)$
 $x_3 \sim \mathcal{N}(\mu_3, \sigma_3^2)$

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

Note: 1. x_1 means data entered in 1st column of all the rows i.e. x_1^1, x_1^2 etc.

2. μ_1 is the mean of all the data present in column 1, μ_2 is the mean of all the data present in column 2 etc.

3.

Anomaly detection algorithm

→ 1. Choose features x_i that you think might be indicative of anomalous examples. $\{x^{(1)}, \dots, x^{(m)}\}$

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

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

$p(x_j; \mu_j, \sigma_j^2)$
 $\mu = \begin{bmatrix} \mu_1 \\ \mu_2 \\ \vdots \\ \mu_n \end{bmatrix} = \frac{1}{m} \sum_{i=1}^m x^{(i)}$

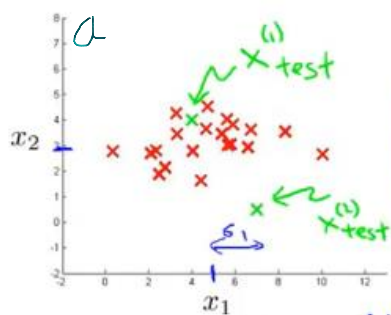
→ 3. Given new example x , compute $p(x)$:

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

Anomaly if $p(x) < \epsilon$

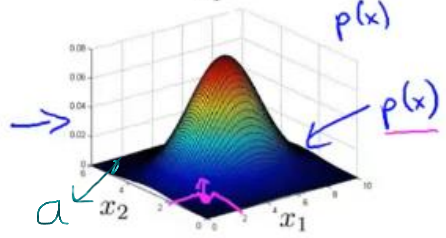
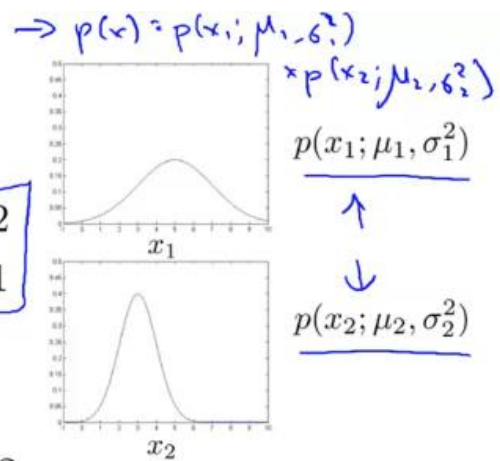
Example:

Anomaly detection example



$$\mu_1 = 5, \sigma_1 = 2$$

$$\mu_2 = 3, \sigma_2 = 1$$



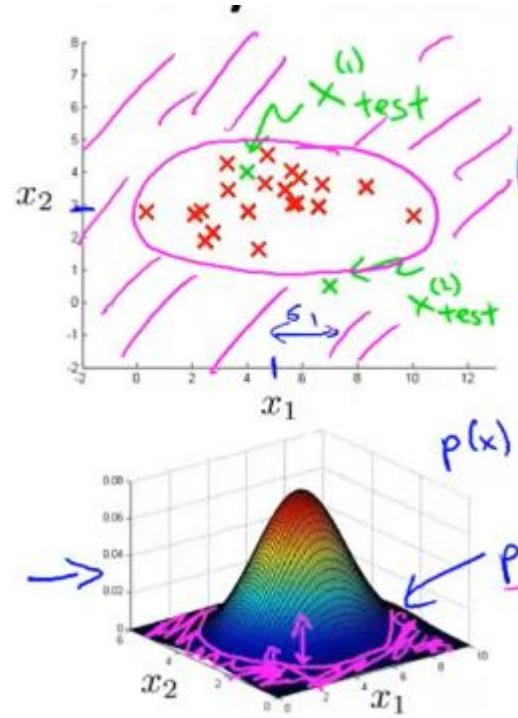
$$\varepsilon = 0.02$$

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

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

Andrew Ng

Note the graph on the lower left corner gives probability value for corresponding x_1 and x_2 values. If a point is in the middle of the plane represent by x_1 vs x_2 , then it's probability value is very high, whereas if a point is to the top left corner like point a, then it's probability is very less.



The area in pink represents points whose probability is very low.

Question:

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

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

Correct

Continue

How to Develop and Evaluate an Anomaly Detection System?

Note: We have an already labelled set of data (labelled as anomaly or normal) and now we are trying to predict the labels/values of y (0 or 1) for some new values of x .

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: $(x_{test}^{(1)}, y_{test}^{(1)}), \dots, (x_{test}^{(m_{test})}, y_{test}^{(m_{test})})$

We evaluate our features using the cross-validation set. Then we finally test our model using the test set.

Aircraft engines motivating example

- 10000 good (normal) engines
- 20 flawed engines (anomalous) 2-50 $y=1$
- Training set: 6000 good engines ($y=0$) $\mu_1, \sigma_1^2, \dots, \mu_n, \sigma_n^2$ $p(x) = p(x_1; \mu_1, \sigma_1^2) \dots p(x_n; \mu_n, \sigma_n^2)$
 - 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 ($y=1$)

Transferring data from b.6sc.co...

Andrew Ng

The alternative method is usually not preferred, since we prefer having different cross-validation and test sets. Now, usually we have unlabelled data, but here we have assumed that most of the values of $p(x)$ correspond to a normal engine and only a small proportion of them correspond to an anomaly.

Algorithm evaluation

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

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

Possible evaluation metrics:

- - True positive, false positive, false negative, true negative
- - Precision/Recall CV
- - F_1 -score Test set.

Can also use cross validation set to choose parameter ϵ \leftarrow

Transferring data from b.6sc.co...

Andrew Ng

We evaluate our model for different values of epsilon and different features of X using the cross-validation set. We can also check the precision/recall value using F_1 score metric for different features/epsilon values.

Note: We do not prefer to use classification accuracy as a method of evaluating the performance of our model, since we are assuming that our model has data X that is skewed i.e. most of the values of $p(x)$ correspond to a normal engine and only a small proportion of them correspond to an anomaly.

Question

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

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

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).

Correct

- ☐ No for the cross validation set; yes for the test set.

Anomaly Detection VS Supervised Learning:

Anomaly detection	vs.	Supervised learning
<ul style="list-style-type: none"> → Very small number of positive examples ($y = 1$). (0-20 is common). → Large number of negative ($y = 0$) examples. $p(x) \ll$ → Many different "types" of anomalies. Hard for any algorithm to learn from positive examples what the anomalies look like; → future anomalies may look nothing like any of the anomalous examples we've seen so far. 		<ul style="list-style-type: none"> Large number of positive and negative examples. ← 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. ←

Andrew Ng

When we have a dataset in which there are very few positive examples, we train our model using the negative examples and reserve the positive examples for the cross validation and test sets. In such a case, we use anomaly detection algorithm.

Note: Here positive examples mean that there is an anomaly/problem and negative examples mean that there is no such problem i.e. the examples are normal.

Also note that many different types of anomalies means that there may be certain parameters that we have not identified, but can contribute to a faulty engine. Now since we are training our algorithm on the normal/negative examples, we have defined a certain value of epsilon below which we classify an example as an anomaly using our normal examples. So, we need not worry about further unidentified parameters that can cause an anomaly because our model is not dependent on the parameters that contribute to a faulty engine but rather the parameters that contribute to a non-faulty engine.

More examples:

Anomaly detection	vs.	Supervised learning
<ul style="list-style-type: none"> → • Fraud detection • Manufacturing (e.g. aircraft engines) • Monitoring machines in a data center ⋮ 		<ul style="list-style-type: none"> • Email spam classification • Weather prediction (sunny/rainy/etc). • Cancer classification ⋮

Question:

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

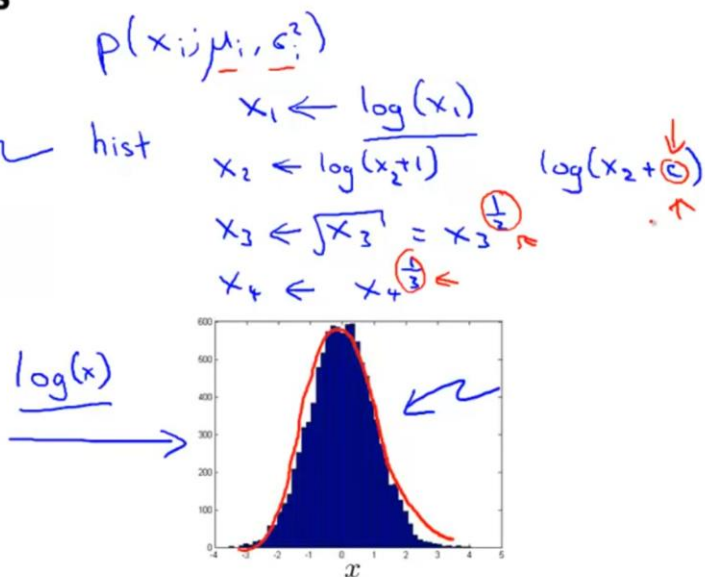
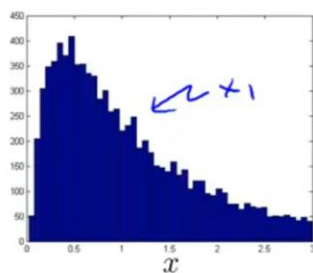
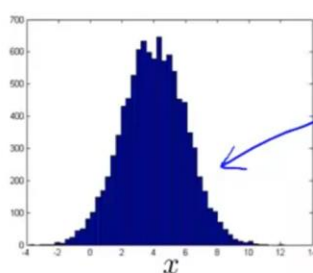
- ☒ You run a power utility (supplying electricity to customers) and want to monitor your electric plants to see if any one of them might be behaving strangely.
- ☐ You run a power utility and want to predict tomorrow's expected demand for electricity (so that you can plan to ramp up an appropriate amount 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:

1. Dealing with non-gaussian features: The anomaly detection algorithm works fine with both non-gaussian as well as gaussian features. Although, it works a bit better with gaussian features.

We can convert non-gaussian features to gaussian by taking certain transformations such as log transformation, square root, cube root etc.

Non-gaussian features



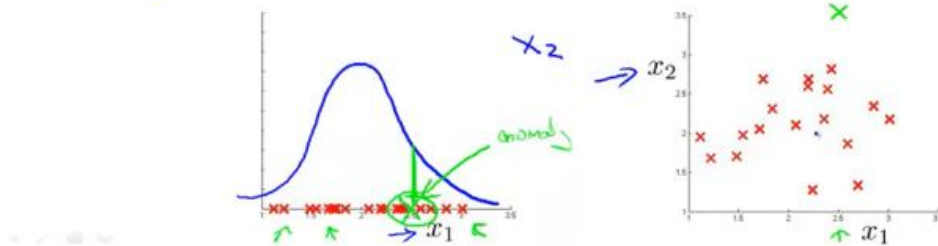
Error Analysis:

→ 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



Suppose our algorithm wrongly classified a data point x (shown in green). Now, we have to introduce a new feature say x^2 which classifies this example as an anomaly. So though x^1 is still=2.5, the data point x is classified as an anomaly due to the new feature x^2 .

Example:

→ 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 ←

$$x_5 = \frac{\text{CPU load}}{\text{network traffic}}$$

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

Let us consider that we face a common problem at our data center that our CPU load increases a lot maybe due to some infinite loop or some other error. Yet when the CPU load increases, the other features like network traffic remain almost constant and thus a computer with high CPU load alone cannot be classified as an anomaly. So, in order to classify or

highlight such a computer as an anomaly, we introduce new features like x_5 or x_6 which can help in correct anomaly detection.

Question:

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?

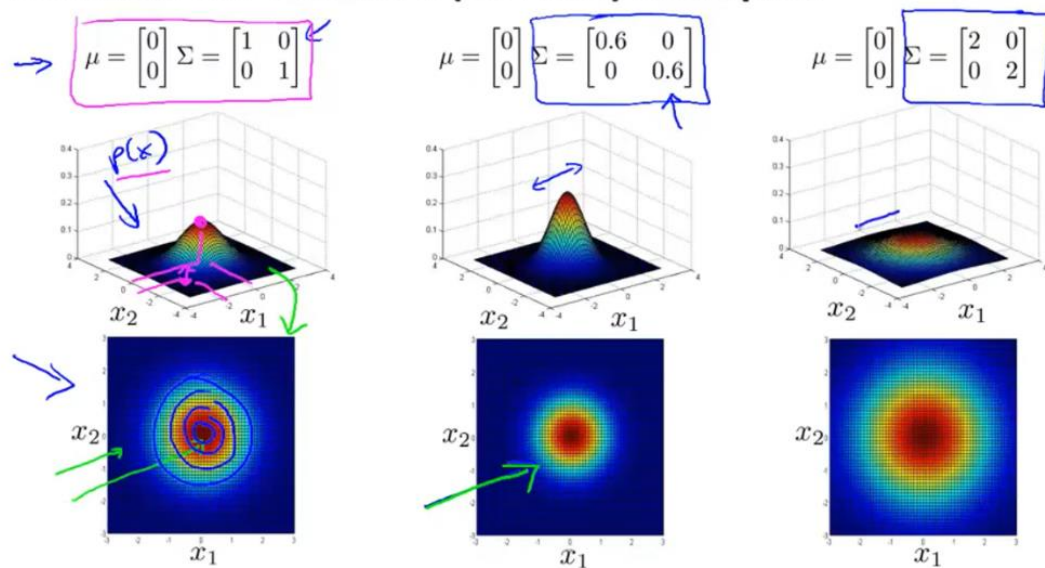
- ☐ Try using fewer features.
- ☒ Try coming up with more features to distinguish between the normal and the anomalous examples.

Correct

- ☐ Get a larger training set (of normal examples) with which to fit $p(x)$.
- ☐ Try changing ϵ .

Different types of Gaussian Distributions depending on values of μ and σ :

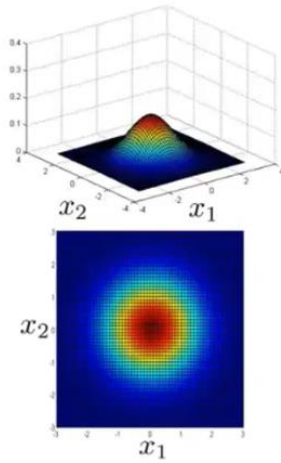
Multivariate Gaussian (Normal) examples



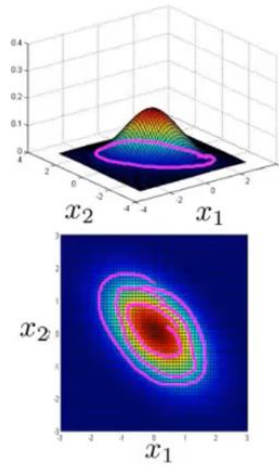
Andrew Ng

Multivariate Gaussian (Normal) examples

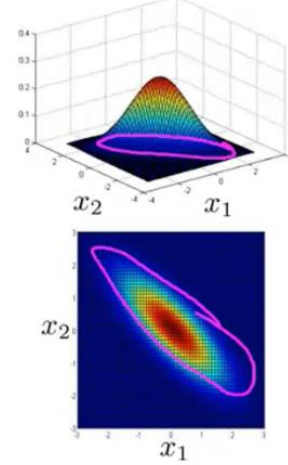
$$\mu = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \quad \Sigma = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$



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



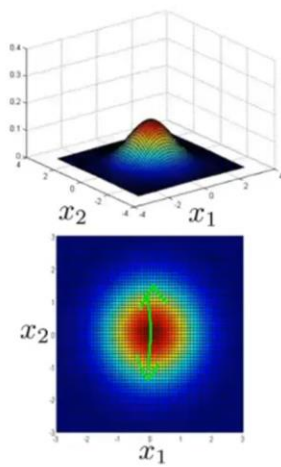
$$\mu = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \quad \Sigma = \begin{bmatrix} 1 & -0.8 \\ -0.8 & 1 \end{bmatrix}$$



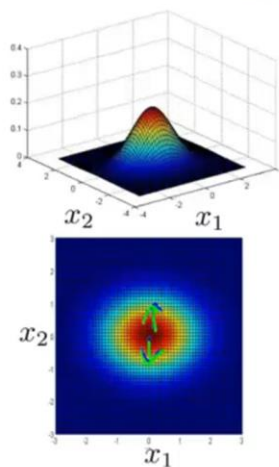
Andrew Ng

Multivariate Gaussian (Normal) examples

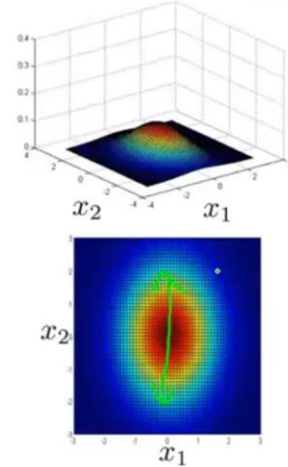
$$\mu = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \quad \Sigma = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$



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

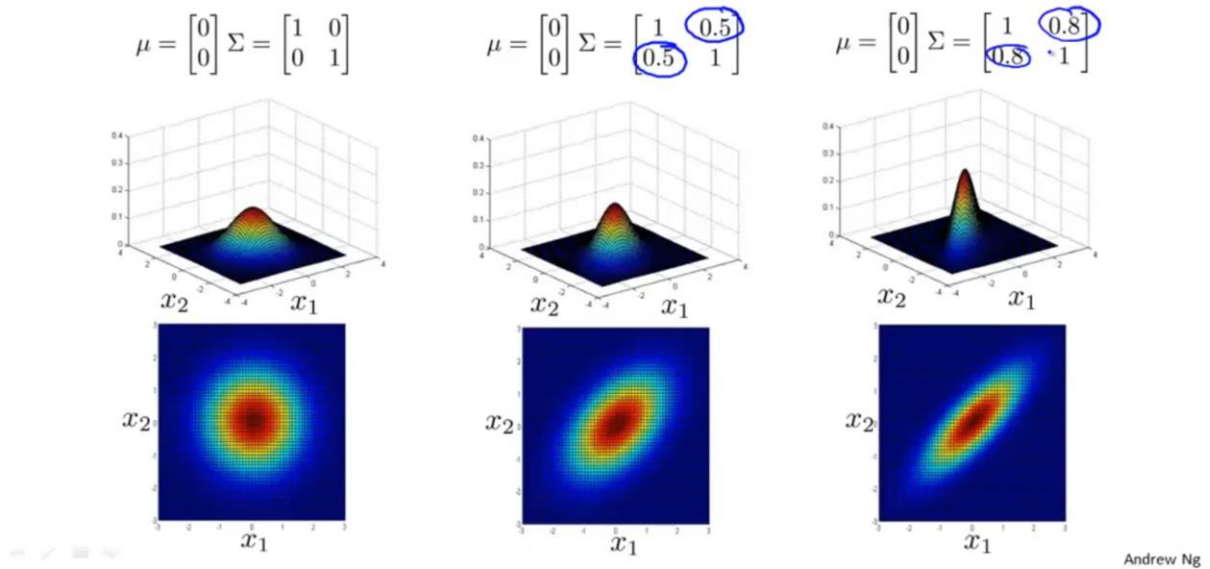


$$\mu = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \quad \Sigma = \begin{bmatrix} 1 & 0 \\ 0 & 2 \end{bmatrix}$$



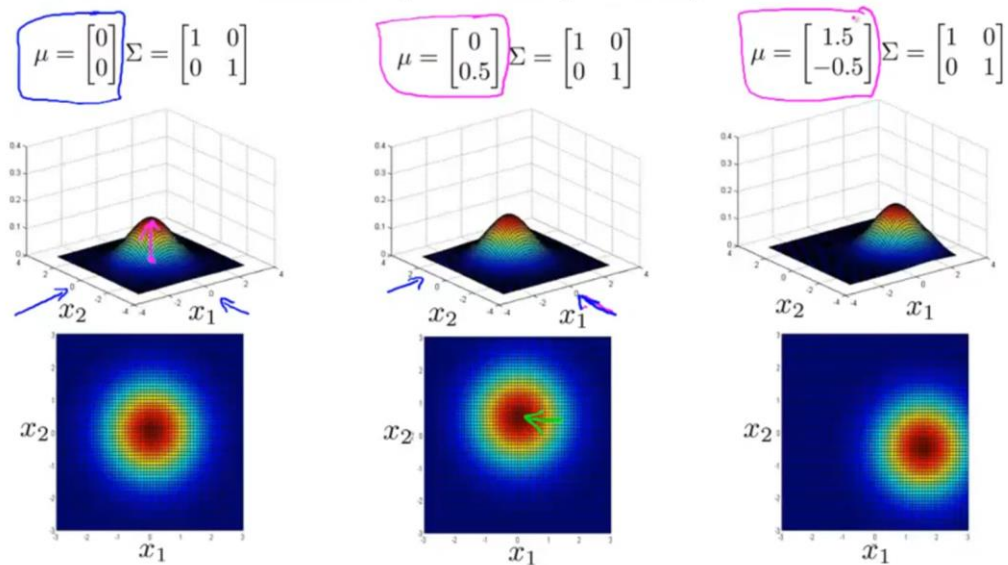
Andrew Ng

Multivariate Gaussian (Normal) examples



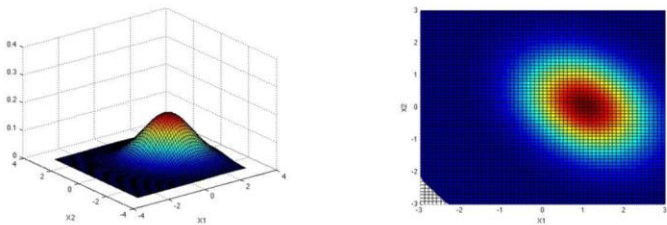
Above distribution can be used to check the correlation between x_1 and x_2 .

Multivariate Gaussian (Normal) examples



Question:

Consider the following multivariate Gaussian:



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

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

Correct

Continue

Detecting Anomalies using Multivariate Gaussian Model:

Anomaly detection with the multivariate Gaussian

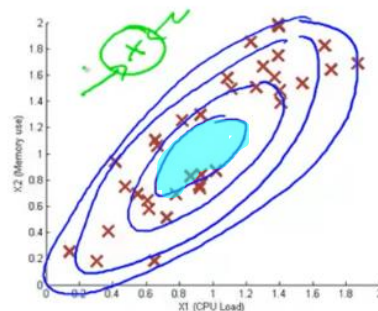
1. Fit model $p(x)$ by setting

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

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 $p(x) < \epsilon$

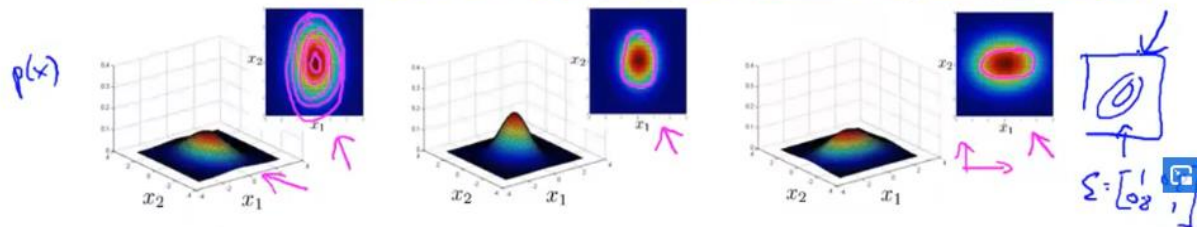


Andrew Ng

The region in sea green is the area of highest probability, as we move away from this area, the probability keeps on decreasing. Point X is really far from this area and thus has a really low probability and hence is labelled as an anomaly.

Relationship to original model

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



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

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

Andrew Ng

Note: The above original model is a special case of the multivariate gaussian model, in which the diagonals of sigma are equal to 0 or sigma varies linear along the axis. The graph in blue on the top right corner of the screen shows a graph in which value of diagonals of sigma is not 0, it is 0.8 .

Difference between the Original Model and Multivariate Gaussian Model:

→ Original model	vs. → <u>Multivariate Gaussian</u>
$p(x_1; \mu_1, \sigma_1^2) \times \dots \times p(x_n; \mu_n, \sigma_n^2)$ <p>Manually create features to capture anomalies where x_1, x_2 take unusual combinations of values.</p> <div style="border: 1px solid black; padding: 5px; display: inline-block;"> $x_3 = \frac{x_1}{x_2} = \frac{\text{CPU load}}{\text{memory}}$ </div> <p>→ Computationally cheaper (alternatively, scales better to large n) $n=10,000, \quad n=100,000$</p> <p>OK even if m (training set size) is small</p>	$p(x; \mu, \Sigma) = \frac{1}{(2\pi)^{\frac{n}{2}} \Sigma ^{\frac{1}{2}}} \exp\left(-\frac{1}{2}(x-\mu)^T \Sigma^{-1} (x-\mu)\right)$ <p>→ Automatically captures correlations between features</p> <div style="display: flex; justify-content: space-around; margin: 10px 0;"> $\Sigma \in \mathbb{R}^{n \times n}$ Σ^{-1} </div> <p>Computationally more expensive</p> <div style="display: flex; align-items: center; margin: 10px 0;"> $\rightarrow \Sigma \sim \frac{n^2}{2}$ <div style="margin-left: 20px;"> $\rightarrow x_1 = x_2$ $x_3 = x_4 + x_5$ </div> </div> <p>Must have $m > n$ or else Σ is non-invertible. $m \geq 10n$</p>

Andrew Ng

Note:

1. Multivariate gaussian model is computationally expensive because, in it we have to compute σ^{-1} and if sigma is very large like 100,000, then computing the inverse of a 100,000 by 100,000 matrix is really difficult.
2. If $x_1, x_2 \dots x_n$ are correlated then multivariate gaussian model won't be able to invert sigma and hence won't be able to calculate its inverse. Eg: As given above if $x_1 = x_2$ or $x_3 = x_4 + x_5$ then features x_2, x_4 and x_5 are redundant or useless.
3. For sigma to be invertible m should be a lot greater than n . Say $m \geq 10n$.

Questions:

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

- ☒ The original model $p(x_1; \mu_1, \sigma_1^2) \times \dots \times p(x_n; \mu_n, \sigma_n^2)$ corresponds to a multivariate Gaussian where the contours of $p(x; \mu, \Sigma)$ are axis-aligned.
Correct
- ☐ Using the multivariate Gaussian model is advantageous when m (the training set size) is very small ($m < n$).
Un-selected is correct
- ☒ The multivariate Gaussian model can automatically capture correlations between different features in x .
Correct
- ☒ 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).
Correct

Continue

1. For which of the following problems would anomaly detection be a suitable algorithm?

- ☒ Given an image of a face, determine whether or not it is the face of a particular famous individual.
- ☐ Given data from credit card transactions, classify each transaction according to type of purchase (for example: food, transportation, clothing).
- ☒ From a large set of primary care patient records, identify individuals who might have unusual health conditions.
- ☒ Given a dataset of credit card transactions, identify unusual transactions to flag them as possibly fraudulent.

2. Suppose you have trained an anomaly detection system for fraud detection, and your system that flags anomalies when $p(x)$ is less than ϵ , and you find on the cross-validation set that it is missing many fraudulent transactions (i.e., failing to flag them as anomalies). What should you do?

- ☒ Increase ϵ
- ☐ Decrease ϵ

1. For which of the following problems would anomaly detection be a suitable algorithm?

- ☒ Given an image of a face, determine whether or not it is the face of a particular famous individual.



This should not be selected

This problem is more suited to traditional supervised learning, as you want both famous and non-famous images in the training set.

- ☐ Given data from credit card transactions, classify each transaction according to type of purchase (for example: food, transportation, clothing).
- ☒ From a large set of primary care patient records, identify individuals who might have unusual health conditions.



Correct

Since you are just looking for unusual conditions instead of a particular disease, this is a good application of anomaly detection.

- ☒ Given a dataset of credit card transactions, identify unusual transactions to flag them as possibly fraudulent.

3. Suppose you are developing an anomaly detection system to catch manufacturing defects in airplane engines. Your model uses

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

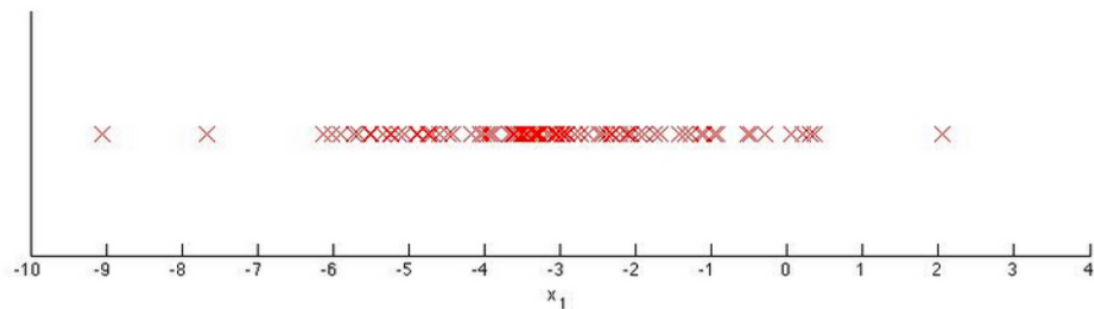
You have two features x_1 = vibration intensity, and x_2 = heat generated. Both x_1 and x_2 take on values between 0 and 1 (and are strictly greater than 0), and for most "normal" engines you expect that $x_1 \approx x_2$. One of the suspected anomalies is that a flawed engine may vibrate very intensely even without generating much heat (large x_1 , small x_2), even though the particular values of x_1 and x_2 may not fall outside their typical ranges of values. What additional feature x_3 should you create to capture these types of anomalies:

- ☐ $x_3 = \frac{1}{x_1}$
- ☐ $x_3 = \frac{1}{x_2}$
- ☒ $x_3 = \frac{x_1}{x_2}$
- ☐ $x_3 = x_1 + x_2$

4. Which of the following are true? Check all that apply.

- ☐ If you have a large labeled training set with many positive examples and many negative examples, the anomaly detection algorithm will likely perform just as well as a supervised learning algorithm such as an SVM.
- ☒ If you do not have any labeled data (or if all your data has label $y = 0$), then it is still possible to learn $p(x)$, but it may be harder to evaluate the system or choose a good value of ϵ .
- ☒ When choosing features for an anomaly detection system, it is a good idea to look for features that take on unusually large or small values for (mainly the) anomalous examples.
- ☐ If you are developing an anomaly detection system, there is no way to make use of labeled data to improve your system.

5. You have a 1-D dataset $\{x^{(1)}, \dots, x^{(m)}\}$ and you want to detect outliers in the dataset. You first plot the dataset and it looks like this:

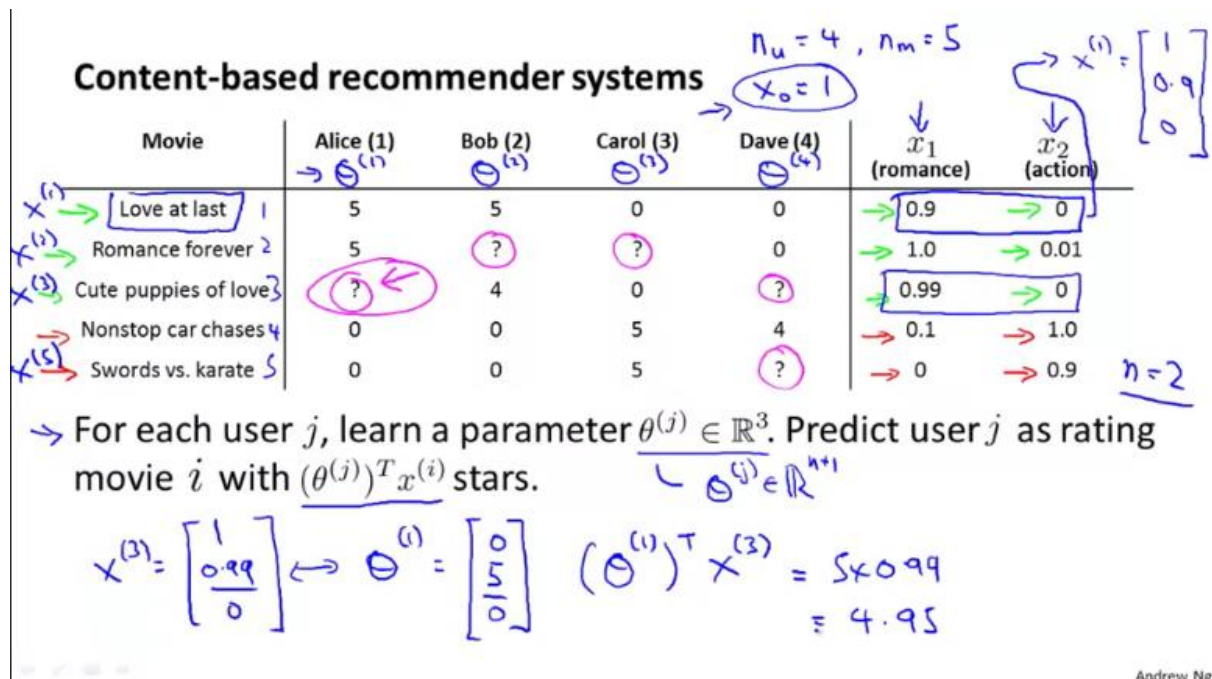


Suppose you fit the gaussian distribution parameters μ_1 and σ_1^2 to this dataset. Which of the following values for μ_1 and σ_1^2 might you get?

- ☒ $\mu_1 = -3, \sigma_1^2 = 4$
- ☐ $\mu_1 = -6, \sigma_1^2 = 4$
- ☐ $\mu_1 = -3, \sigma_1^2 = 2$
- ☐ $\mu_1 = -6, \sigma_1^2 = 2$

Content Based Recommendations:

Consider the following figure:



Here we are trying to predict whether a person say Alice would like a movie like Cute puppies of love, for which they have not voted.

As parameters x , we have certain features x_1 and x_2 . The y values for Alice are the values below the column Alice i.e. 5, 5, 0, 0.

We try to determine the value of theta for which the cost function is the minimum.

The Cost Function:

Week 9 > Content Based Recommendations

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 $\theta^{(j)}$:

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

$\theta^{(j)} \in \mathbb{R}^{n+1}$

Andrew Ng

We remove m from the regularized cost function of linear regression to get the above given cost function.

$i : r(i, j) = 1$ means that we take only those values of movies i for which user j has given a rating.

Each movie has its own feature vector x^i which when multiplied by theta values corresponding to each person gives the person's predicted opinion about the movie.

Example:

The screenshot shows a quiz interface with a table of movie ratings and a question about the value of $\theta^{(3)}$.

Movie	Alice (1)	Bob (2)	Carol (3)	David (4)	x_0	(romance) x_1	(action) x_2
Love at last	5	5	0	0	1	0.9	0
Romance forever	5	?	?	0	1	1.0	0.01
Cute puppies of love	?	4	0	?	1	0.99	0
Nonstop car chases	0	0	5	4	1	0.1	1.0
Swords vs. karate	0	0	5	?	1	0	0.9

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

- ☐ 0
- ☐ $\theta^{(3)} = 5$
- ☐ 0
- ☐ 0
- ☐ $\theta^{(3)} = 0$
- ☐ 1
- ☐ 1
- ☐ $\theta^{(3)} = 0$
- ☐ 4
- ☒ 0
- ☐ $\theta^{(3)} = 0$
- ☐ 5

Correct

Continue

Now, for each movie, we have feature vector x_0, x_1, x_2 . Theta1 corresponds to Alice, Theta2 corresponds to Bob and so on.

When we multiply Theta3 with x values for romance forever, we get the predicted rating given by Carol for this movie. When we multiply Theta3 with x values for Nonstop car chases, we get the predicted rating given by Carol for this movie.

In the Question above, the best value of theta3 is option d) because when we multiply these values of theta3 with corresponding x values for each movie, we get predicted movie rating of each movie for Carol:

$$\text{Love at last} = 1 \cdot 0 + 0.9 \cdot 0 + 0 \cdot 5 = 0$$

$$\text{Romance forever} = 0 \cdot 1 + 1 \cdot 0 + 0.01 \cdot 5 = 0.05$$

.

.

.

$$\text{Swords vs Karate} = 1 \cdot 0 + 0 \cdot 0 + 0.9 \cdot 5 = 4.95$$

Optimizing values of Theta:

Week 9 > Content Based Recommendations

Optimization algorithm:

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

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

Gradient descent update:

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

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

$\frac{\partial}{\partial \theta_k^{(j)}} J(\theta^{(1)}, \dots, \theta^{(n_u)})$

Andrew Ng

Collaborative Filtering:

In the above slides, we tried to predict the best values of Theta, given parameters x. Now, we will see a reverse method, i.e. given Theta values we will try to predict the best values of parameter x that will minimize the cost function.

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	?	?	?

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

$\theta^{(j)}$

$x^{(1)} = \begin{bmatrix} 1.0 \\ 0.0 \end{bmatrix}$
 $(\theta^{(1)})^T x^{(1)} \approx 5$
 $(\theta^{(2)})^T x^{(1)} \approx 5$
 $(\theta^{(3)})^T x^{(1)} \approx 0$
 $(\theta^{(4)})^T x^{(1)} \approx 0$

Andrew Ng

The lower right corner of the above image shows that for the movie Love at Last, we are trying to best values of parameters x, which when multiplied by the theta values for each of the 4 persons, predicts their review about it and also minimizes the cost function.

Example:

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

Correct

☐ 1

☐ 2

☐ Any of these values would be equally reasonable.

Continue

We know the values of $\theta_1, \theta_2, \theta_3$. Now we want to find the best values of x so that when θ^T and x are multiplied, we obtain the most accurate values of y .

We find that the best value of x is $[1 \ 0.5]$, because when x is multiplied by θ^T , we obtain:

For user 1: $0*1 + 0*0.5=0$

For user 2: $0*1 + 3*0.5=1.5$

For user 3: $0*1 + 5*0.5=2.5$

Cost Function:

Optimization algorithm

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

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

Gradient Descent:

Suppose you use gradient descent to minimize:

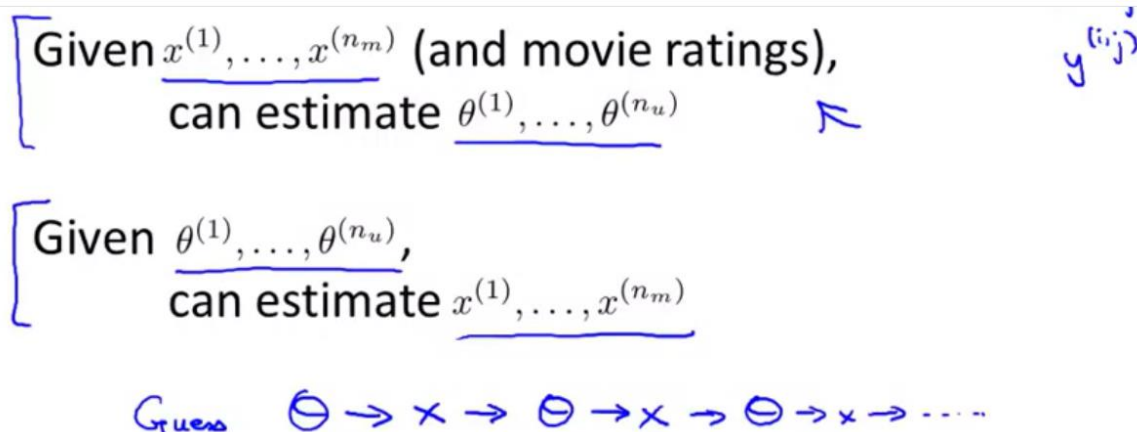
$$\min_{x^{(1)}, \dots, x^{(n_m)}} \frac{1}{2} \sum_{i=1}^{n_m} \sum_{j:r(i,j)=1} (\theta^{(j)})^T x^{(i)} - y^{(i,j)} + \frac{\lambda}{2} \sum_{i=1}^{n_m} \sum_{k=1}^n (x_k^{(i)})^2$$

Which of the following is a correct gradient descent update rule for $i \neq 0$?

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

Conclusion for Collaborative Filtering:

Check out the image below:



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

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

Guess $\Theta \rightarrow x \rightarrow \Theta \rightarrow x \rightarrow \Theta \rightarrow x \rightarrow \dots$

Practicing Collaborative filtering:

1. What we can do is first assume a value of theta, then calculate x using it.
2. Then using this value of x, we can estimate a better value of theta and then again use this value of theta to determine an even more accurate x.
3. We continue this process until we obtain an optimal value of theta and x.
4. In this process, with each step, we filter the value of theta and x hence it is called collaborative filtering.

Cost Function for Collaborative Filtering:

Collaborative filtering optimization objective

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

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

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

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

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

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

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

Andrew Ng

Collaborative filtering optimization objective

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

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

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

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

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

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

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

Andrew Ng

We have combined the 2 cost functions 1 and 2 to get the cost function for collaborative filtering.

Conclusion:

Collaborative filtering algorithm

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

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

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

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

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

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

Andrew Ng

Question:

In the algorithm we described, we initialized $x^{(1)}, \dots, x^{(n_m)}$ and $\theta^{(1)}, \dots, \theta^{(n_u)}$ 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 $x^{(1)}, \dots, x^{(n_m)}$ that are different from each other.

Correct

Continue

Andrew Ng

