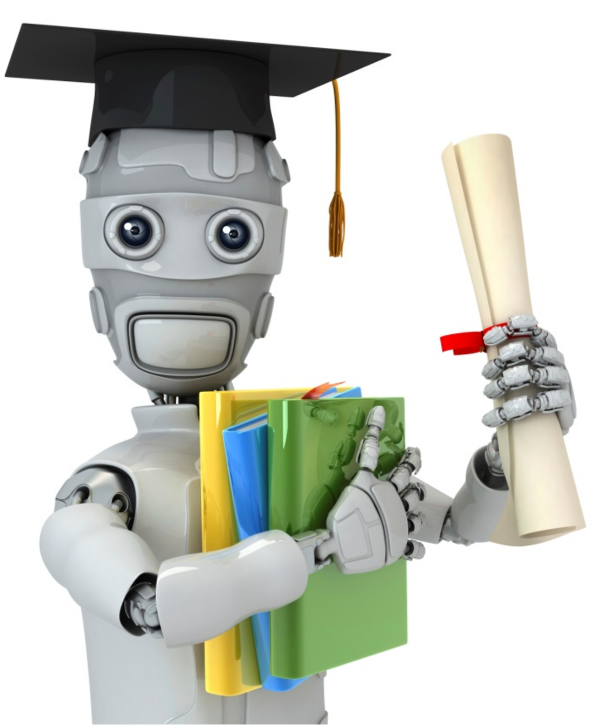


Machine Learning

# Anomaly detection

## Problem motivation

it's mainly for unsupervised problem, that there's some aspects of it that are also very similar to sort of the supervised learning problem.

## Anomaly detection example

Aircraft engine features:

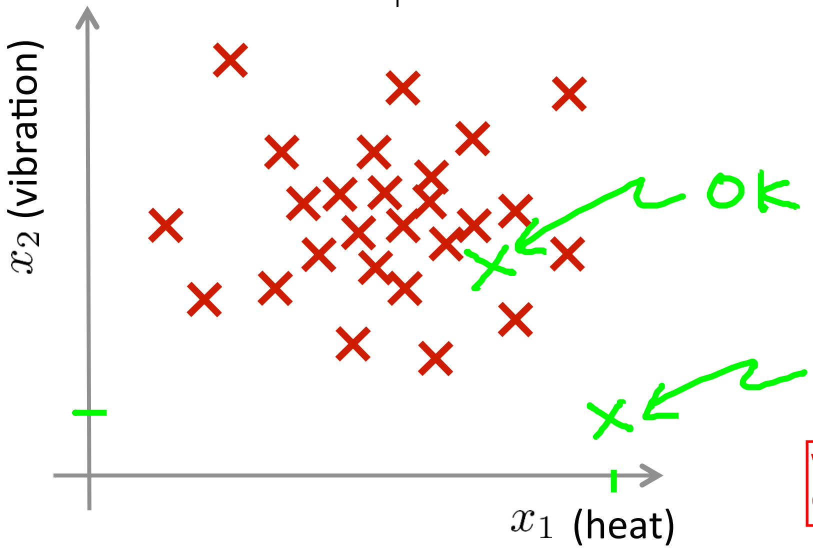
→ $x_1$ = heat generated

→ $x_2$ = vibration intensity

…

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

New engine: $\underline{\underline{x_{test}}}$



we want to know if this aircraft engine is anomalous

Andrew Ng

## Density estimation

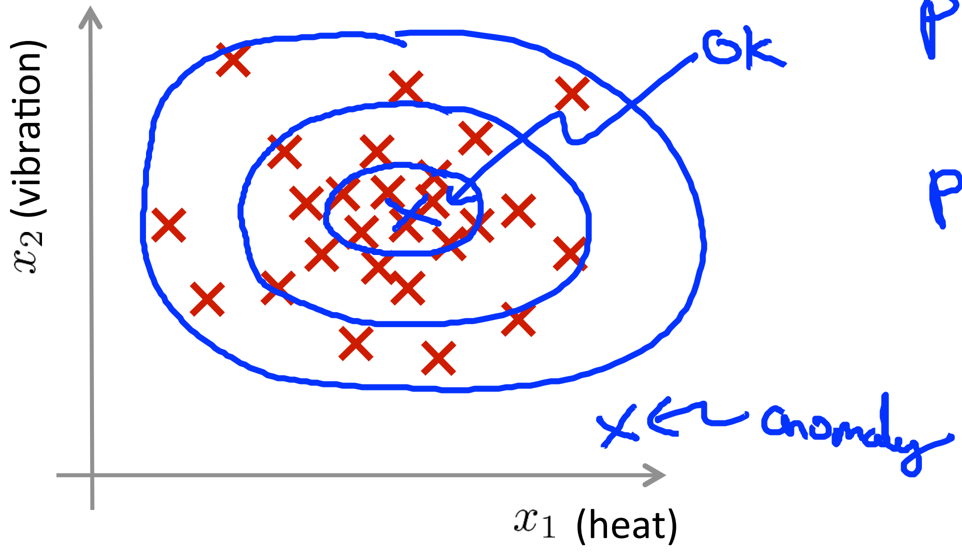
- Dataset: $\{x^{(1)}, x^{(2)}, \ldots, x^{(m)}\}$
- Is $x_{test}$ anomalous?

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

The approach is that given the unlabeled training set, we're going to build a model for p of x, having built a model of the probability of x we're then going to say that for the new aircraft engine, if p of x-test is less than some epsilon then we flag this as an anomaly.

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

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



Andrew Ng

## Anomaly detection example

$x_1$
$x_2$
$x_3$
$x_4$

$p(x)$

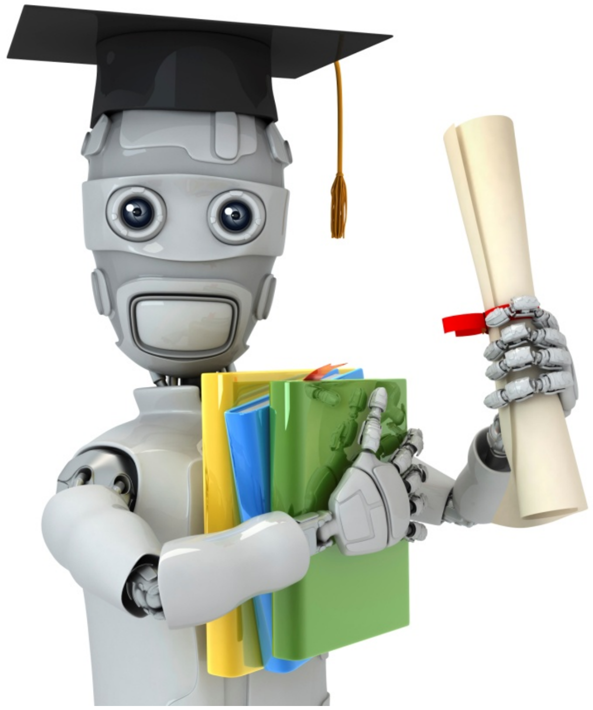
- → Fraud detection:
  - → $x^{(i)}$ = features of user $i$ 's activities
  - → Model $p(x)$ from data.
  - → Identify unusual users by checking which have $\underline{p(x) < \varepsilon}$
- → Manufacturing
- → Monitoring computers in a data center.
  - → $x^{(i)}$ = features of machine $i$
  - $x_1$ = memory use, $x_2$ = number of disk accesses/sec,
  - $x_3$ = CPU load, $x_4$ = $\underline{\text{CPU load}}$/$\underline{\text{network traffic}}$.
  - …

$p(x) < \varepsilon$

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 $\epsilon$.
- ● Try decreasing $\epsilon$. ✓

**Correct**



Machine Learning

# Anomaly detection

## Gaussian distribution

or normal distribution

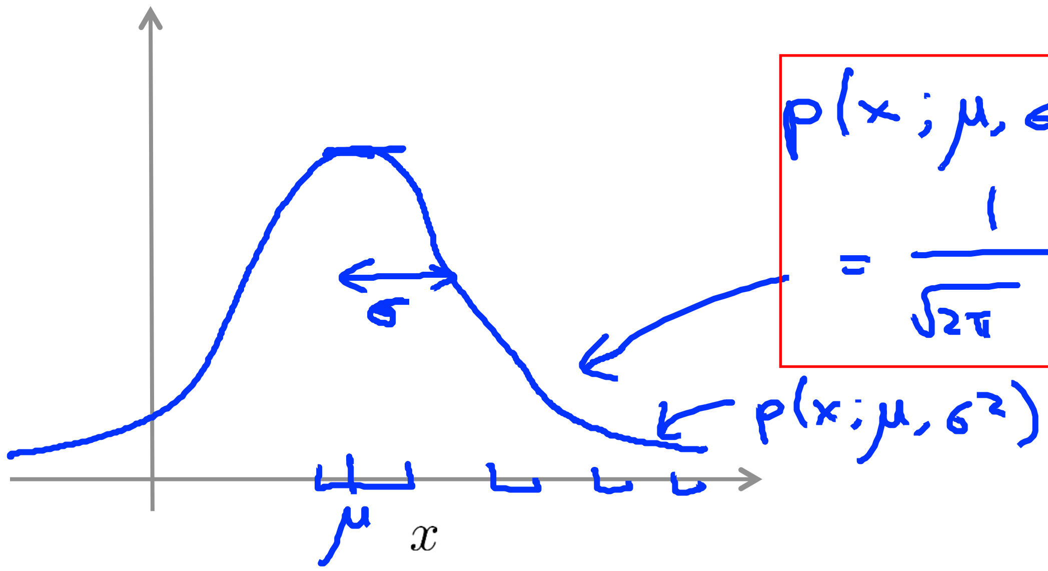
## Gaussian (Normal) distribution

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

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

"distributed as"

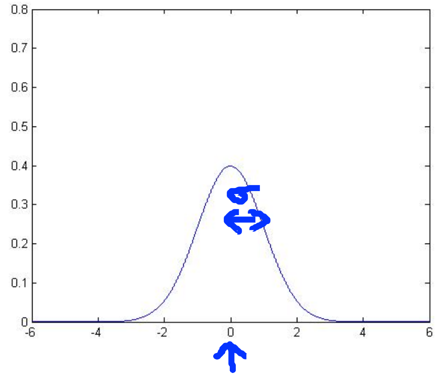
$\sigma$ standard deviation



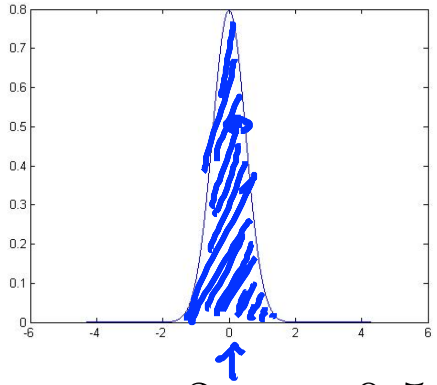
$$\boxed{p(x;\mu,\sigma^2) = \frac{1}{\sqrt{2\pi}\,\sigma} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)}$$

## Gaussian distribution example

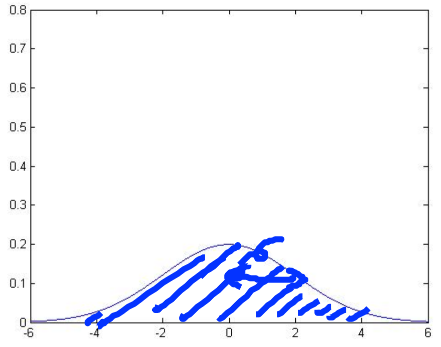
$\rightarrow \mu = 0, \sigma = 1$



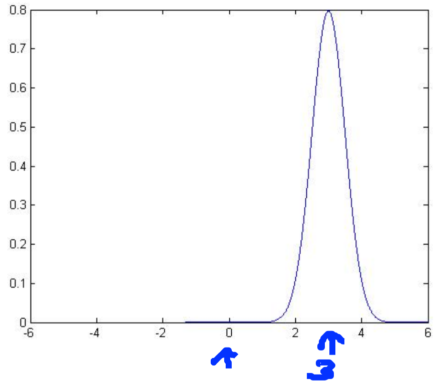
$\rightarrow \mu = 0, \sigma = \underline{0.5}$  $\sigma^2 = 0.25$



$\rightarrow \mu = 0, \sigma = 2$



$\rightarrow \mu = 3, \sigma = 0.5$

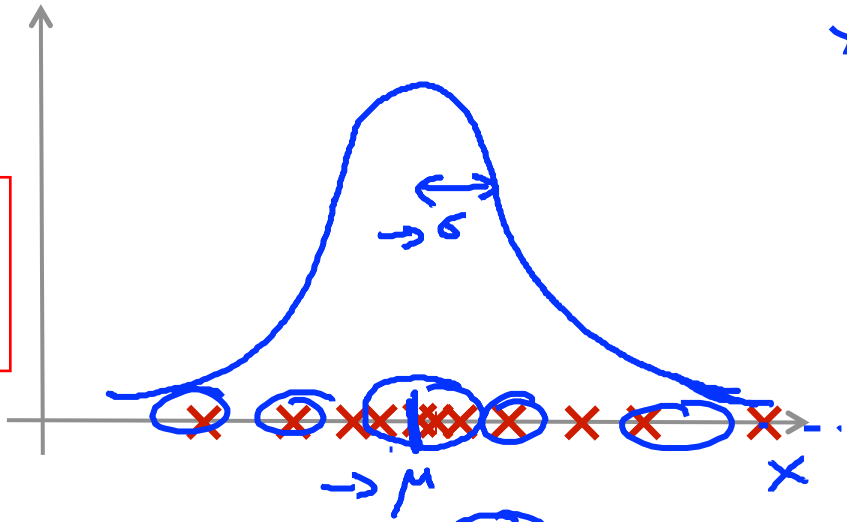


Andrew Ng

## Parameter estimation

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

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



these parameters, these estimates, are actually the maximum likelihood estimates of the parameters mu and sigma square

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

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

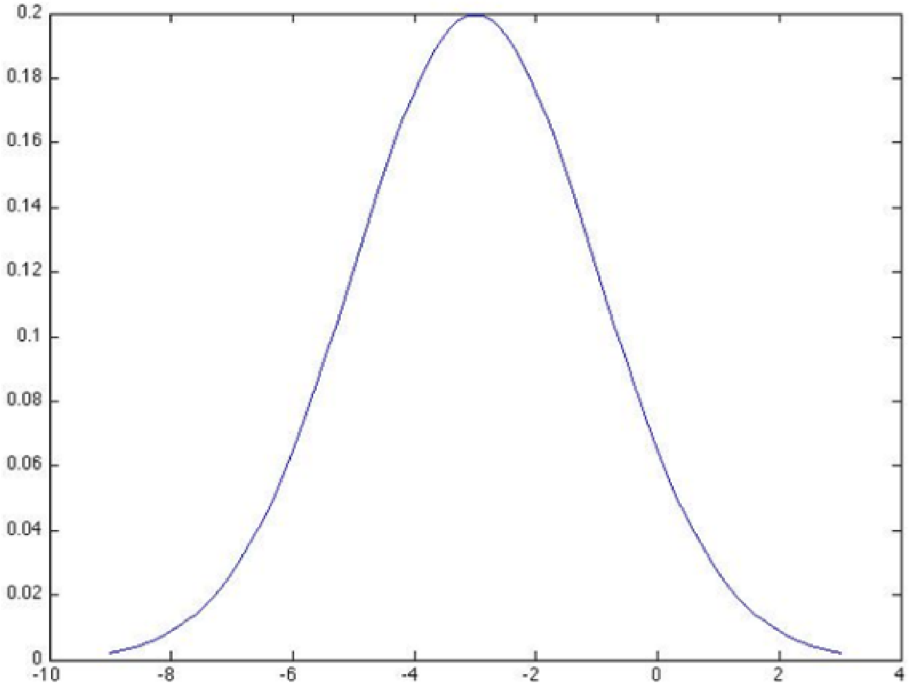
$m-1$ $\frac{1}{m-1} \leftarrow$

Some of you may have seen the formula here where this is M-1. In machine learning people tend to use 1/M formula

The formula for the Gaussian density is:

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

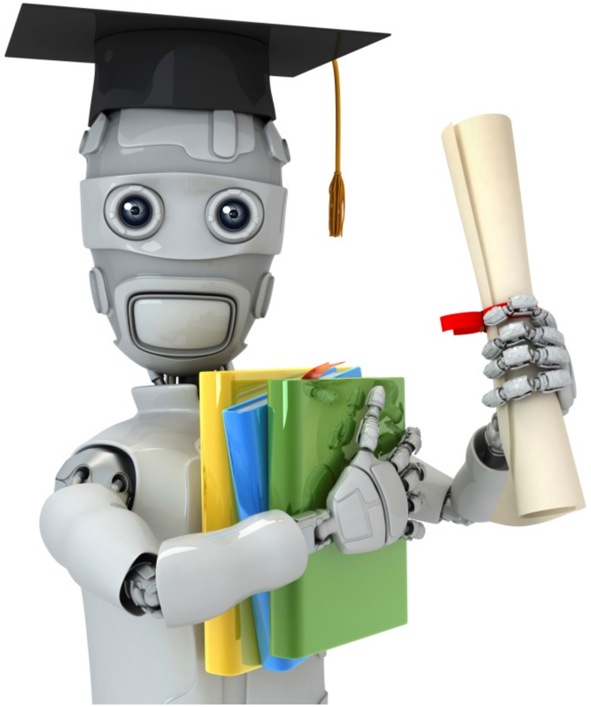
Which of the following is the formula for the density to the right?



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

**Correct**

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



Machine Learning

## Anomaly detection

## Algorithm

## $\rightarrow$ Density estimation

$\rightarrow$ Training set: $\{x^{(1)}, \ldots, x^{(m)}\}$
Each example is $\underline{x \in \mathbb{R}^n}$

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

$\rightarrow p(x)$

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

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

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

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

it turns out that this equation corresponds to an independence assumption on the values of the features x1 through xn. But in practice it turns out that the algorithm, it works just fine, whether or not these features are anywhere close to independent and even if independence assumption doesn't hold true this algorithm works just fine.

Andrew Ng

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

## Anomaly detection algorithm

$\rightarrow$ 1. Choose features $\underline{x_i}$ that you think might be indicative of anomalous examples.

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

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

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

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

$$p(\underset{\uparrow}{x_j}; \underset{\uparrow}{\mu_j}, \sigma_j^2)$$

$\mu_1, \mu_2, \dots, \mu_n$

$$\underset{\uparrow}{\mu} = \begin{bmatrix} \mu_1 \\ \mu_2 \\ \vdots \\ \mu_n \end{bmatrix} = \frac{1}{m}\sum_{i=1}^{m} \underset{\uparrow}{x^{(i)}}$$

$\rightarrow$ 3. Given new example $x$, compute $\underline{p(x)}$:

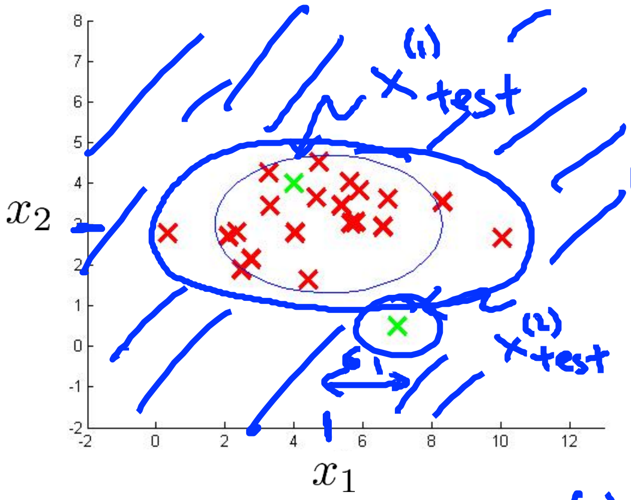
$$\underline{p(x)} = \underbrace{\prod_{j=1}^{n} \underbrace{p(x_j; \mu_j, \sigma_j^2)}}_{} = \underbrace{\prod_{j=1}^{n} \underbrace{\frac{1}{\sqrt{2\pi}\sigma_j} \exp\left(-\frac{(x_j - \mu_j)^2}{2\sigma_j^2}\right)}}_{}$$

Anomaly if $\underline{p(x) < \varepsilon}$

Andrew Ng

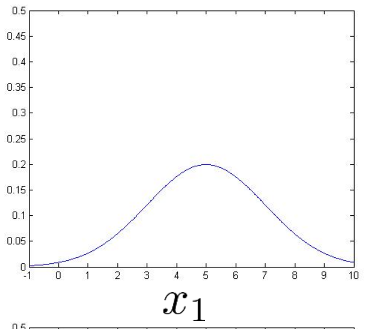
$\rightarrow p(x) = p(x_1; \mu_1, \sigma_1^2)$
$\times p(x_2; \mu_2, \sigma_2^2)$

## Anomaly detection example



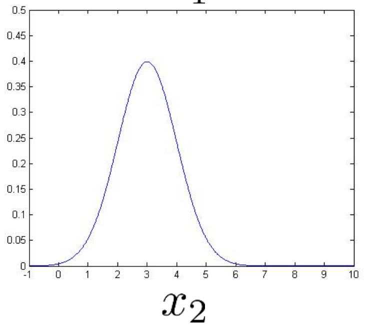
$\sigma_1^2, \sigma_2^2$
$= 4$

$$\boxed{\begin{aligned}\mu_1 &= 5, \underline{\sigma_1} = 2\\ \mu_2 &= 3, \underline{\sigma_2} = 1\end{aligned}}$$

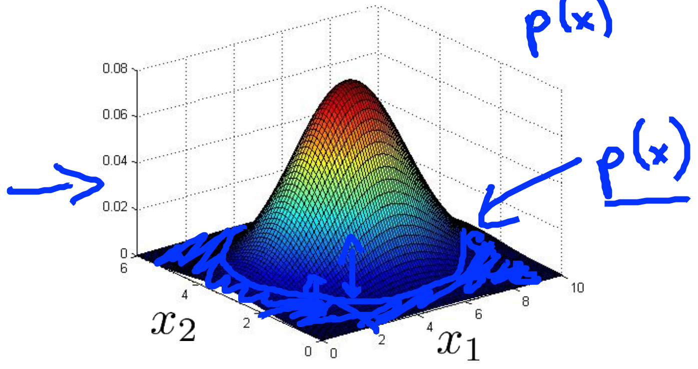


$\underline{p(x_1; \mu_1, \sigma_1^2)}$

$\updownarrow$

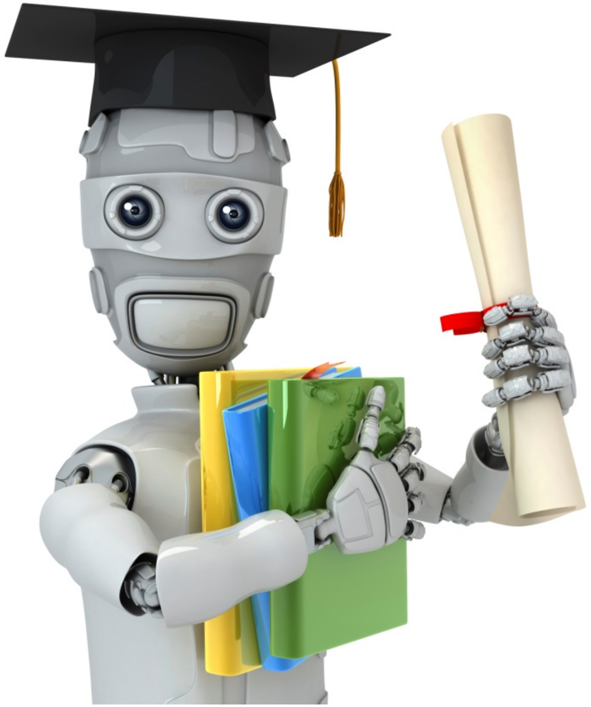


$\underline{p(x_2; \mu_2, \sigma_2^2)}$



$$\underline{\varepsilon = 0.02}$$

$$p(x_{test}^{(1)}) = \underline{0.0426} \quad \geq \varepsilon$$
$$p(x_{test}^{(2)}) = \underline{0.0021} \quad < \varepsilon$$



Machine Learning

# Anomaly detection

## Developing and evaluating 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.

in order to evaluate an anomaly detection system :

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

→ Training set: $x^{(1)}, x^{(2)}, \ldots, x^{(m)}$ (assume normal examples/not anomalous) but it's actually okay even if a few anomalies slip into your unlabeled training set

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

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

$y=1$

Andrew Ng

use these 6000 engines to fit p of x

## Aircraft engines motivating example

10000 good (normal) engines

20 flawed engines (anomalous) 2-50

$y=1$

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

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

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

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

we like to think of the cross validation set and the test set as being completely different data sets to each other.

Alternative: not recommended

Training set: 6000 good engines

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

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

sometimes people use the same set of good engines in the cross validation sets, and the test sets, and sometimes people use exactly the same sets of anomalous engines in the cross validation set and the test set. And so, all of these are considered less good practices and definitely less recommended.

I am calling them unlabeled examples, but these are really examples that we're assuming our goods are the normal aircraft engines.

## Algorithm evaluation

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

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

→ On a cross validation/test example $\underline{x}$, predict

labels will be skewed because y equals zero, usually be much more common than y equals 1

because the data is very skewed, classification accuracy would not be a good the evaluation metrics.

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

$\underline{y=0}$

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 $\boxed{\varepsilon}$ ←

if you have a cross validation set, another way to choose this parameter epsilon, would be to try a different values of epsilon, and then pick the value of epsilon that, maximizes f1 score, or that otherwise does well on your cross validation set.

Andrew Ng

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

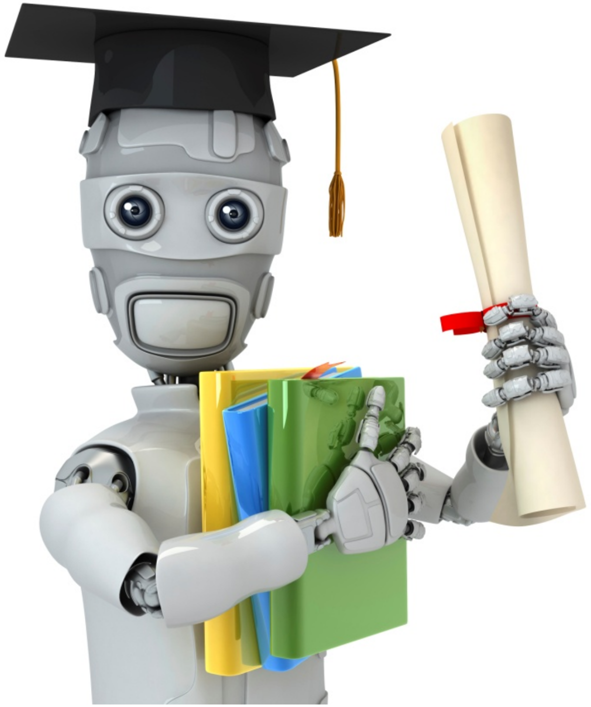
$$y = \begin{cases} 1 & \text{if } p(x) \leq \epsilon \\ 0 & \text{if } p(x) > \epsilon \end{cases}$$

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.



Machine Learning

# Anomaly detection

## Anomaly detection vs. supervised learning

## Anomaly detection vs. Supervised learning

**Anomaly detection**

- Very small number of positive examples ($y = 1$). (0-20 is common). anomaly 0-50
- Large number of negative ($y = 0$) examples. $\boxed{p(x)}$ normal
- 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.

**Supervised learning**

- 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.

Spam

for spam problem we usually have enough examples of spam email. that's why we usually think of spam as a supervised learning even though there are many different types of.

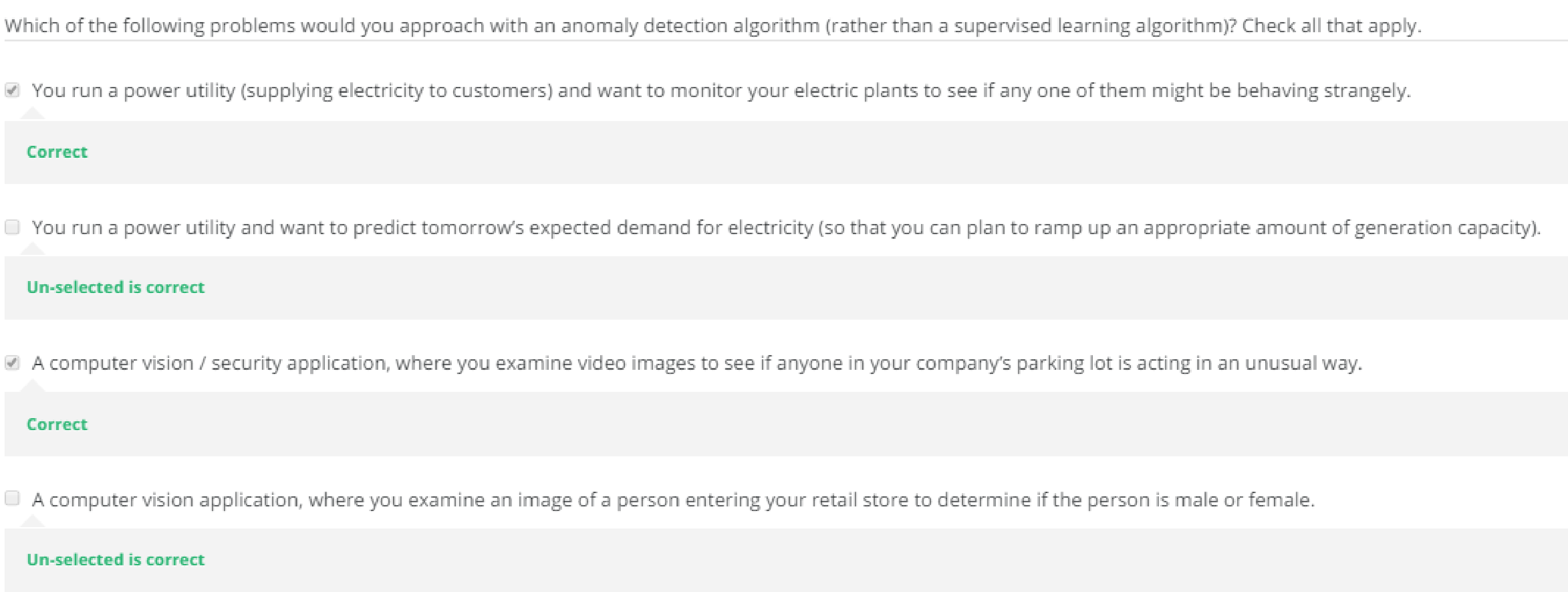
if you actually have had a lot of people commit fraud on your website, so you actually have a lot of examples of y=1, then sometimes fraud detection could actually shift over to the supervised learning column.

## Anomaly detection vs. Supervised learning

| Anomaly detection | Supervised learning |
|---|---|
| • Fraud detection $y=1$ | • Email spam classification |
| • Manufacturing (e.g. aircraft engines) | • Weather prediction (sunny/rainy/etc). |
| • Monitoring machines in a data center | • Cancer classification |
| ⋮ | ⋮ |

for many other problems that are faced by various technology companies and so on, we actually are in the settings where we have very few or sometimes zero positive training examples. There's just so many different types of anomalies that we've never seen them before. And for those sorts of problems, very often the algorithm that is used is an anomaly detection algorithm.

Andrew Ng



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.

**Correct**

☐ 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).

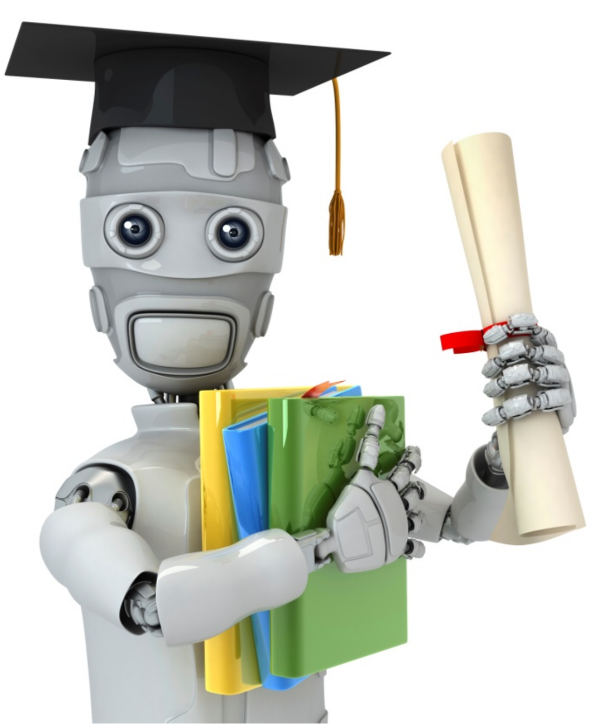
**Un-selected is correct**

☑ 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.

**Correct**

☐ 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.

**Un-selected is correct**



Machine Learning

# Anomaly detection

## Choosing what features to use

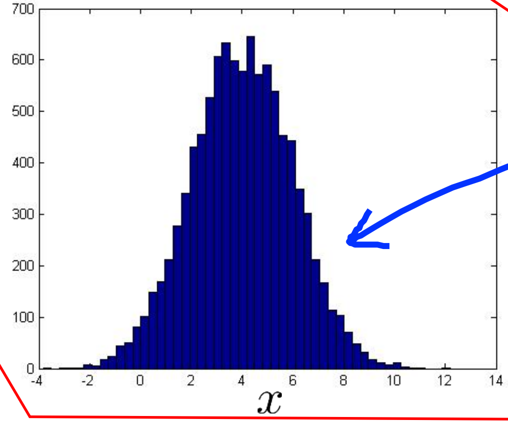
when you're applying anomaly detection, one of the things that has a huge effect on how well it does, is what features you use, and what features you choose, to give the anomaly detection algorithm.

In anomaly detection algorithm, one of the things we did was model the features using Gaussian distribution. One thing that I often do would be to plot the data (or histogram) to make sure that the data looks vaguely Gaussian before feeding it to algorithm.

## Non-gaussian features

$p(x_i; \mu_i, \sigma_i^2)$

hist



replace x with log(x+c)

$x_1 \leftarrow \log(x_1)$

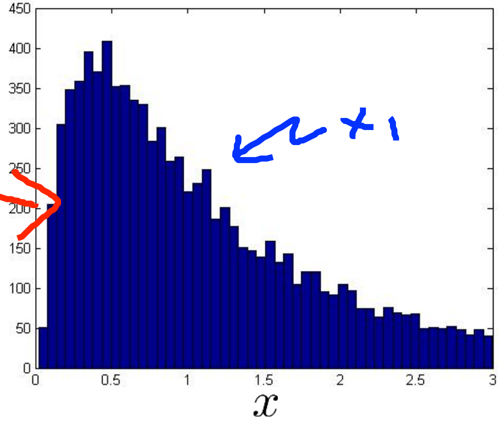
$x_2 \leftarrow \log(x_2+1)$ $\quad \log(x_2 + \boxed{c})$

$x_3 \leftarrow \sqrt{x_3} = x_3^{\frac{1}{2}}$

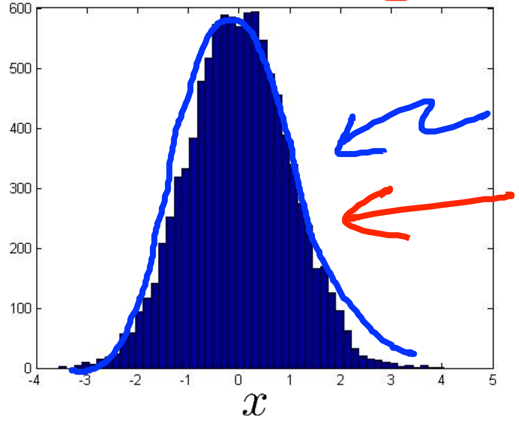
$x_4 \leftarrow x_4^{\frac{1}{3}}$

replace x with x^a

If this is what data looks like, what I'll often do is play with different transformations of the data in order to make it look more Gaussian.



$$\xrightarrow{\log(x)}$$



## Error analysis for anomaly detection

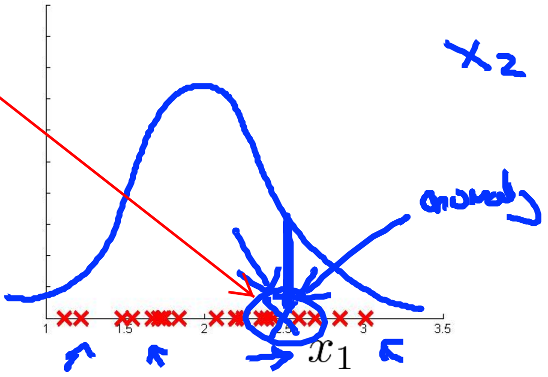
Want $p(x)$ large for normal examples $x$.

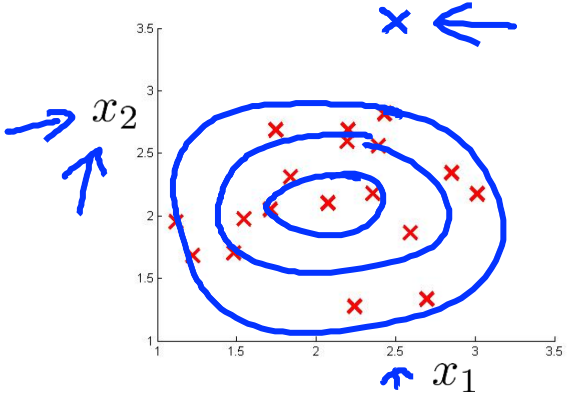
$p(x)$ small for anomalous examples $x$.

Most common problem:

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

this anomalous example (bad engine) gets a pretty high probability, where it's the height of the blue curve, and the algorithm fails to flag this as an anomalous example.





I would actually look at my training examples and look at what went wrong with that particular bad engine, and see if I can come up with a new feature X2, that helps to distinguish between this bad example, and my normal aircraft engines. And hope that I can create a new feature, X2, so that when I re-plot my data, I find that for my anomalous example, the feature X2 takes on the the unusual value. So for my bad example, my X1 value, is still 2.5. Then maybe my X2 value, is a very large value like 3.5 over there, or a very small value.

let me share with you my thinking on how I usually go about choosing features for anomaly detection.

## → 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 ←

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

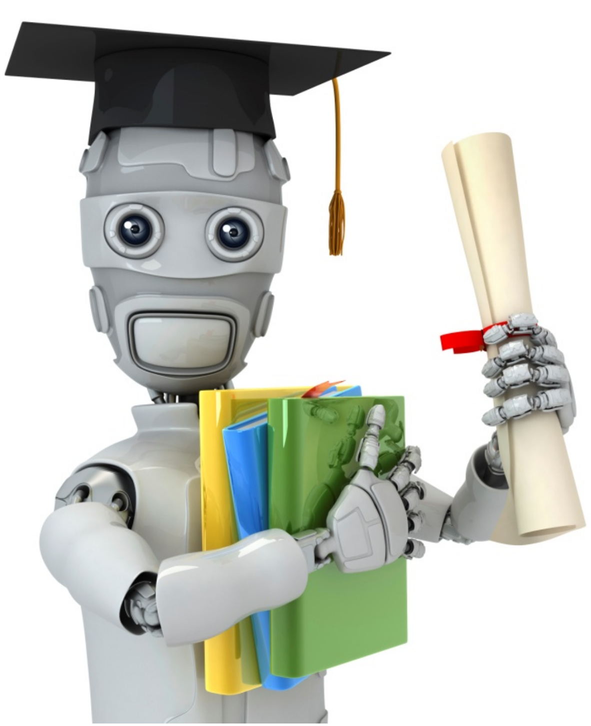
$$x_6 = \frac{(\text{CPU load})^2}{\text{network traffic}}$$

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 $\epsilon$.



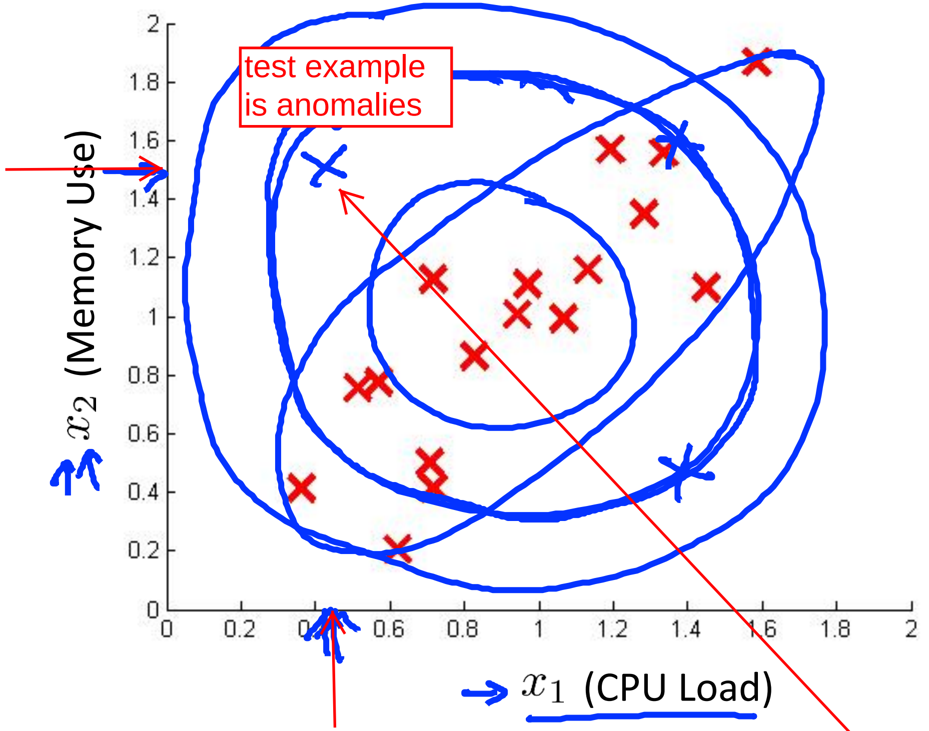
Machine Learning

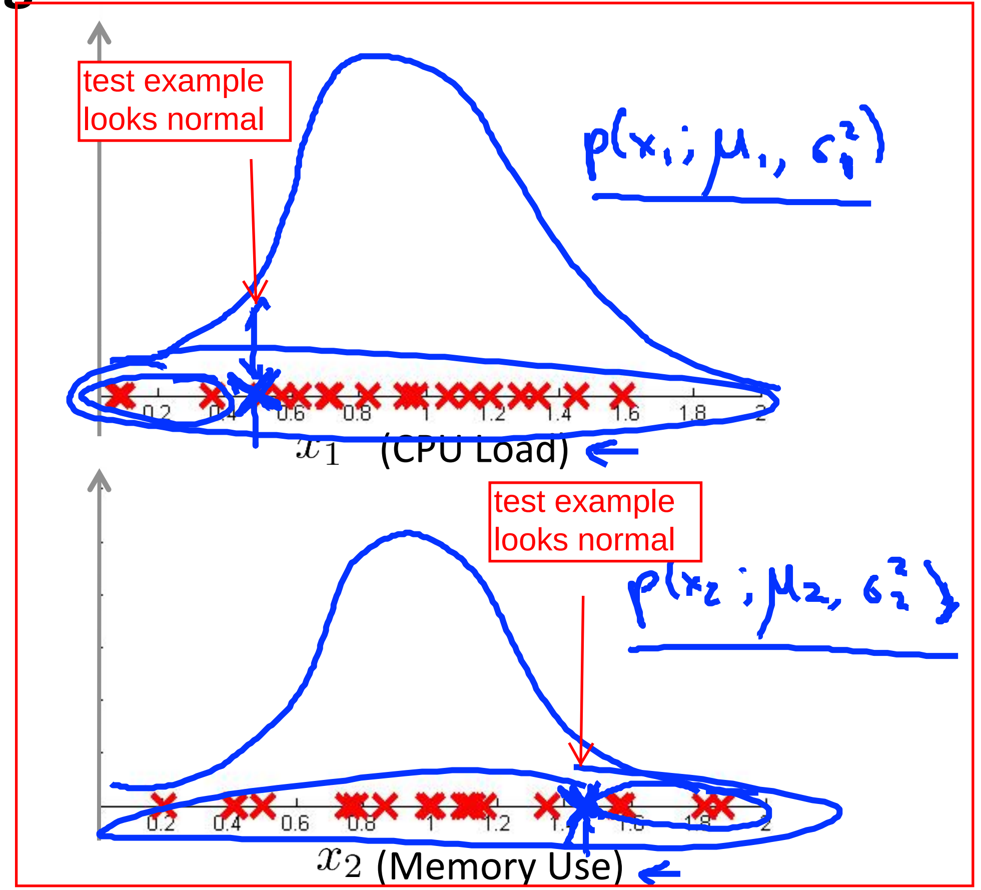
# Anomaly detection

## Multivariate Gaussian distribution

This extension uses something called the multivariate Gaussian distribution, and it has some advantages, and some disadvantages, and it can sometimes catch some anomalies that the earlier algorithm didn't.

# Motivating example: Monitoring machines in a data center





an anomaly detection algorithm will fail to flag this point as an anomaly

## Multivariate Gaussian (Normal) distribution

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

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

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

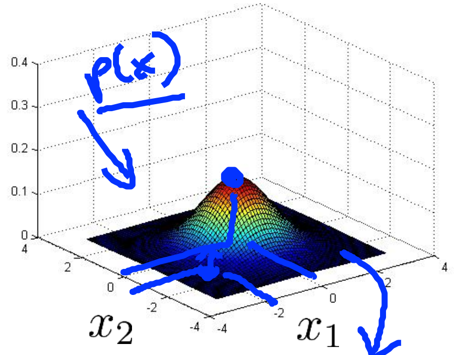
Andrew Ng

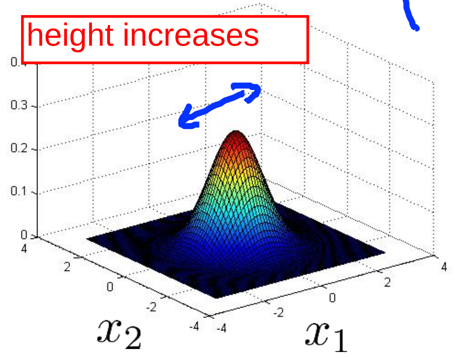
# Multivariate Gaussian (Normal) examples

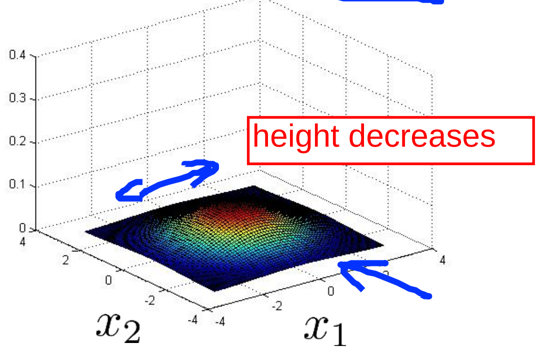
$\boxed{\mu = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \Sigma = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}}$ identity matrix

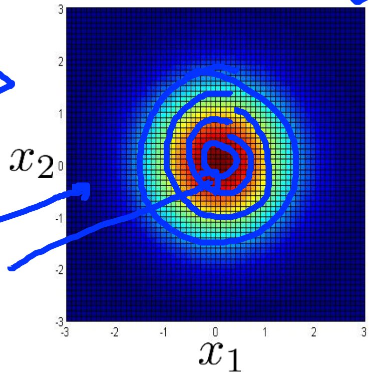
$\mu = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$ $\boxed{\Sigma = \begin{bmatrix} 0.6 & 0 \\ 0 & 0.6 \end{bmatrix}}$

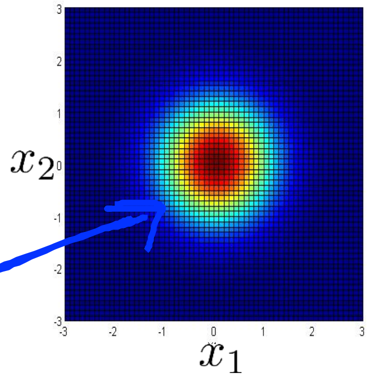
$\mu = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$ $\boxed{\Sigma = \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix}}$

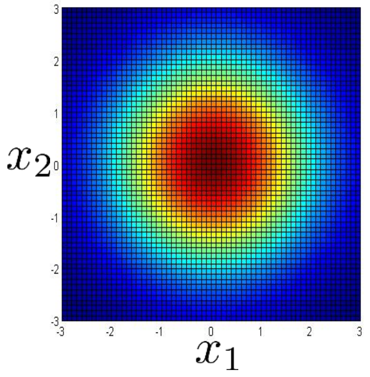












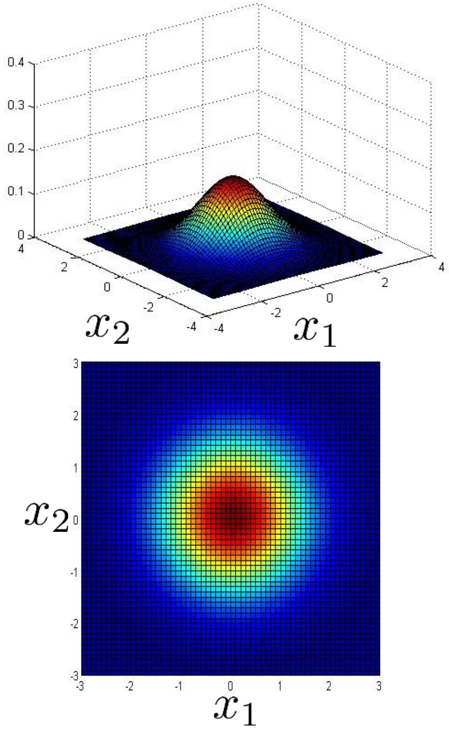
Andrew Ng

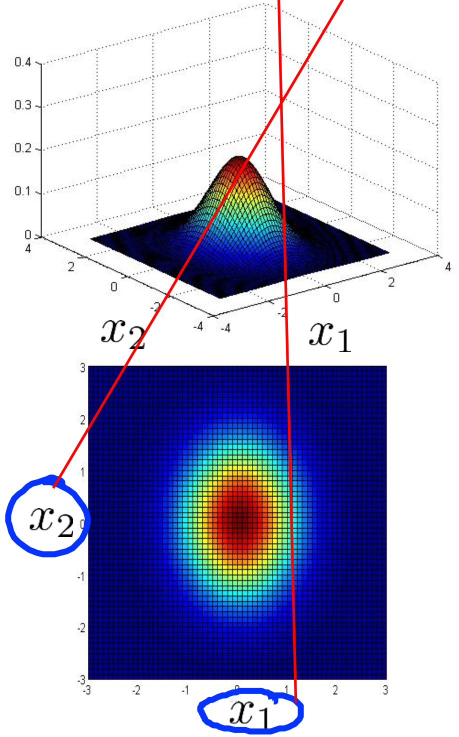
# Multivariate Gaussian (Normal) examples

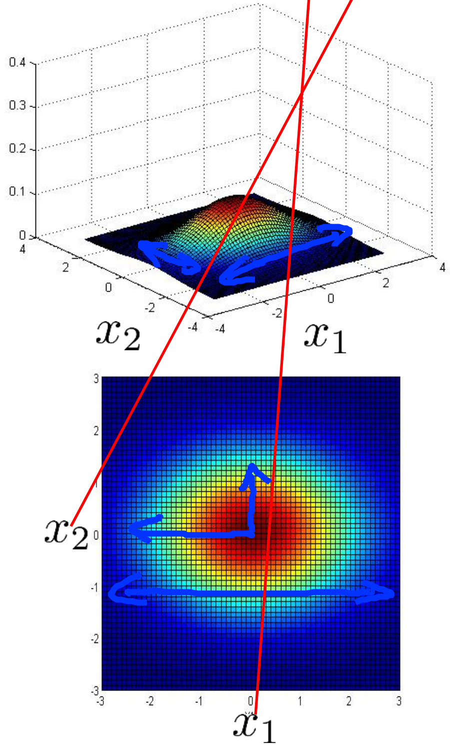
$\mu = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \Sigma = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$

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

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





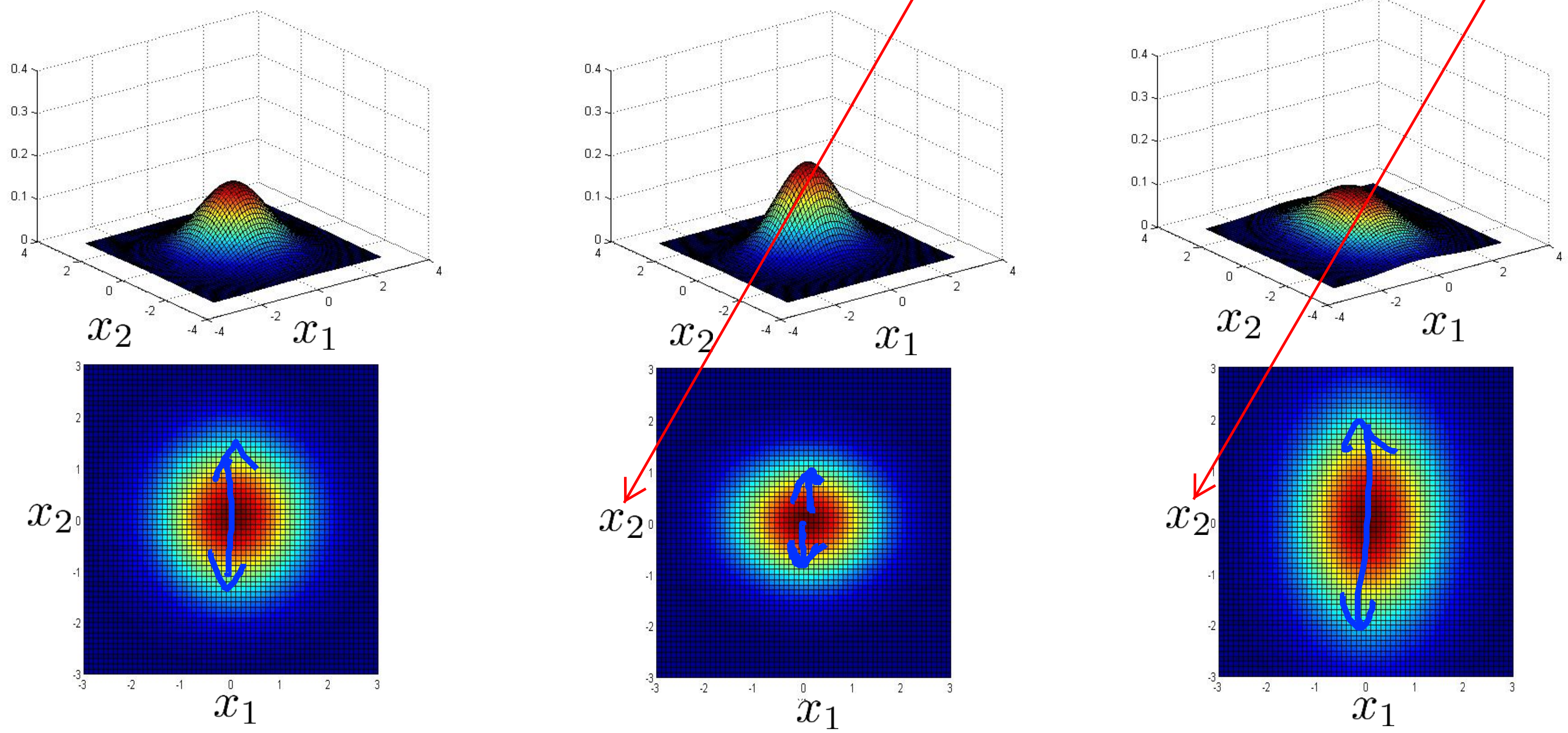


# Multivariate Gaussian (Normal) examples

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

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

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



Andrew Ng

one of the cool things about the multivariate Gaussian distribution is that you can also use it to model correlations between the data.

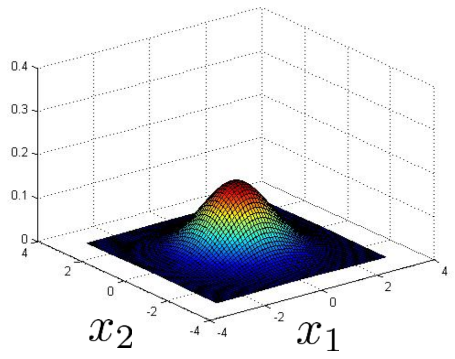
## Multivariate Gaussian (Normal) examples

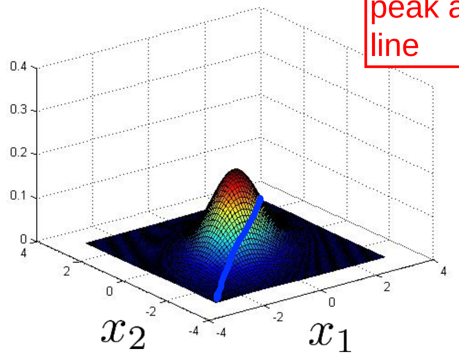
$\mu = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \Sigma = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$

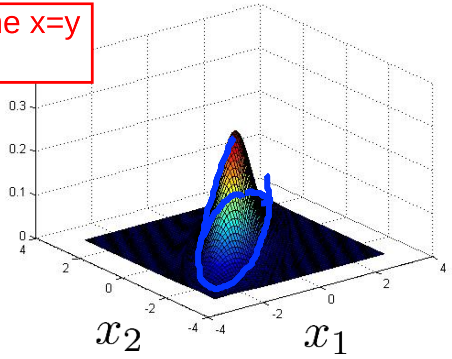
$\mu = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \Sigma = \begin{bmatrix} 1 & 0.5 \\ 0.5 & 1 \end{bmatrix}$

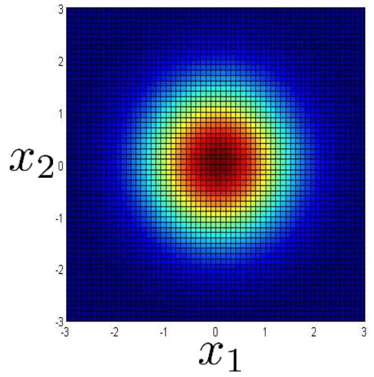
$\mu = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \Sigma = \begin{bmatrix} 1 & 0.8 \\ 0.8 & 1 \end{bmatrix}$

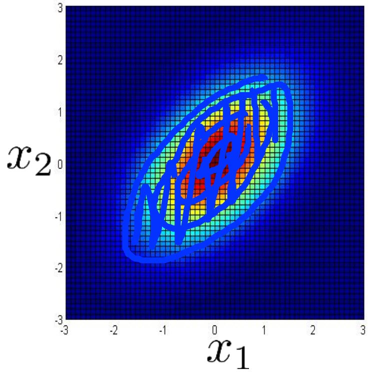
peak along the x=y line

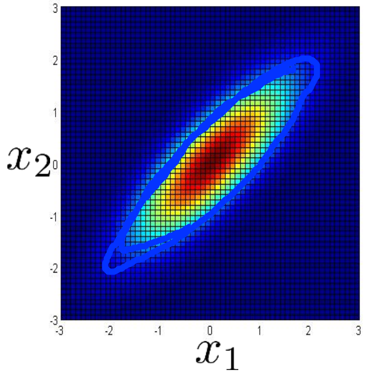








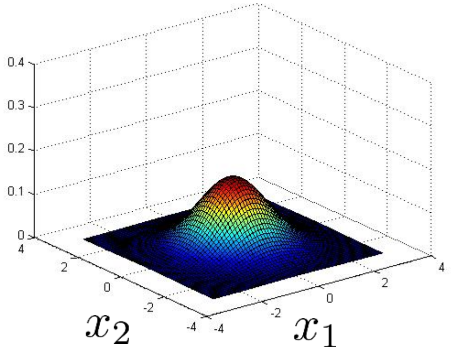


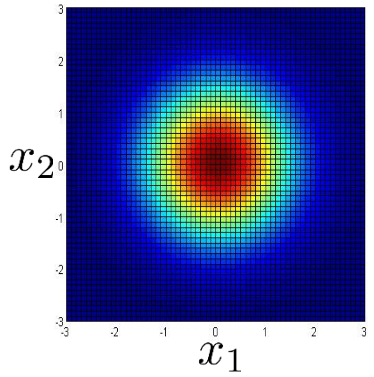


Andrew Ng

# Multivariate Gaussian (Normal) examples

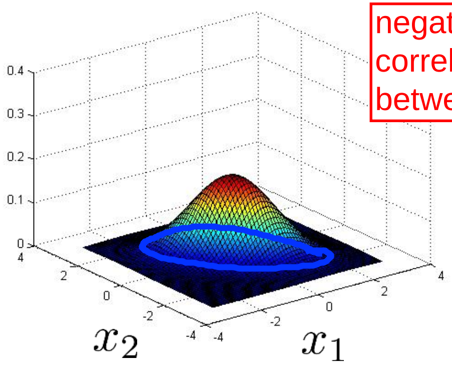
$\mu = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \; \Sigma = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$

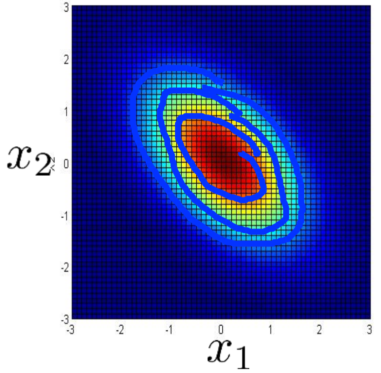




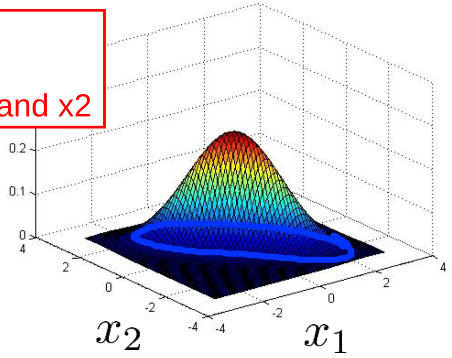
$\mu = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \; \Sigma = \begin{bmatrix} 1 & -0.5 \\ -0.5 & 1 \end{bmatrix}$

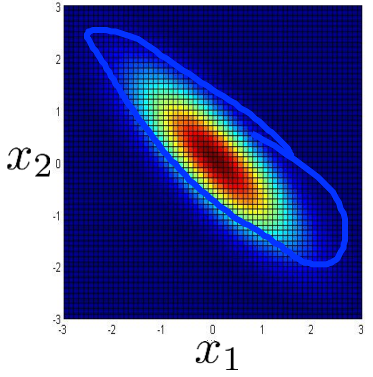
negative correlation between x1 and x2





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

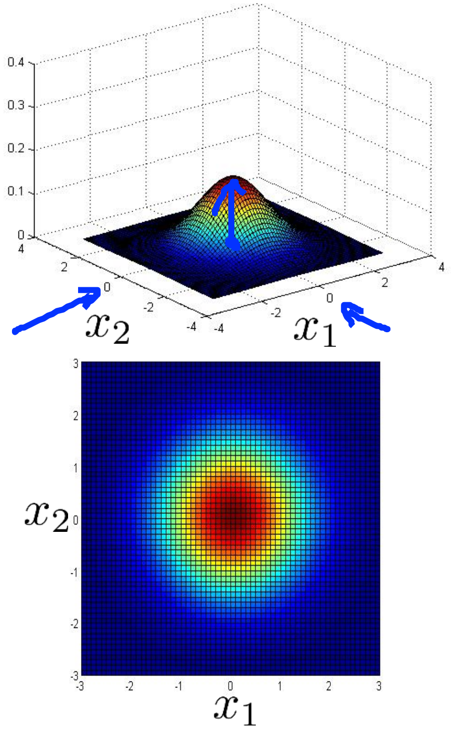




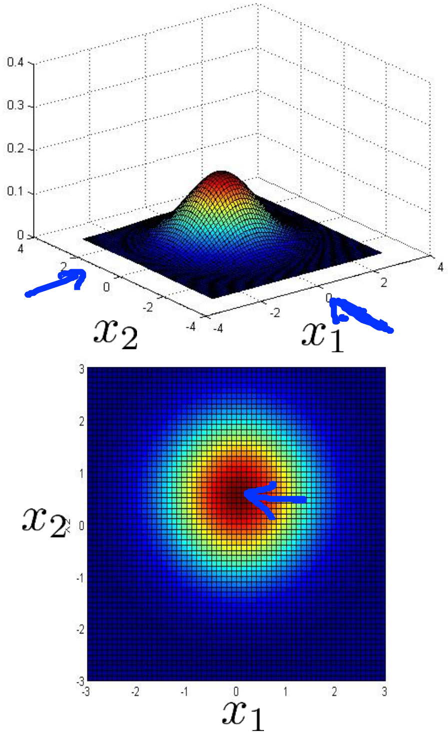
Andrew Ng

# Multivariate Gaussian (Normal) examples

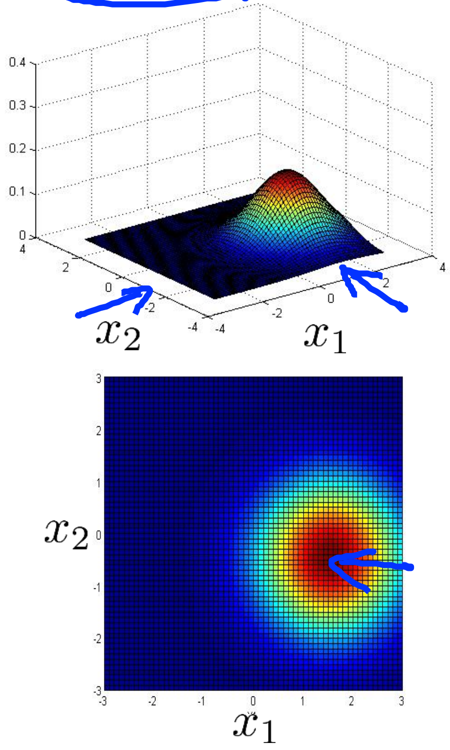
$\boxed{\mu = \begin{bmatrix} 0 \\ 0 \end{bmatrix}}\ \Sigma = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$



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

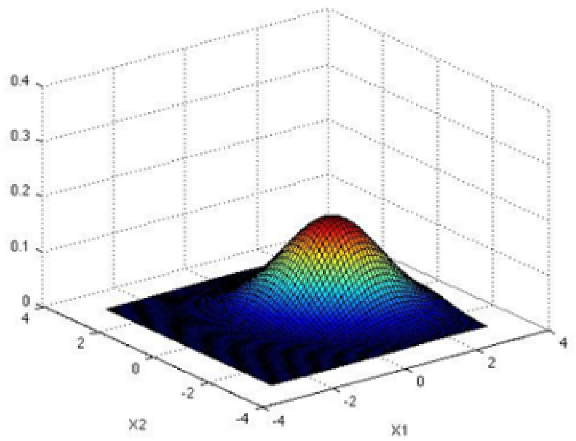


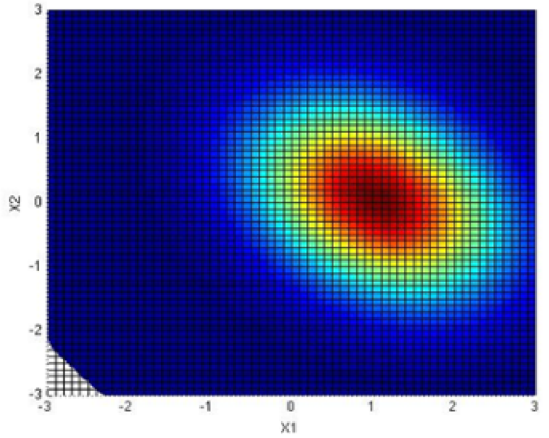
$\boxed{\mu = \begin{bmatrix} 1.5 \\ -0.5 \end{bmatrix}}\ \Sigma = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$



Andrew Ng

Consider the following multivariate Gaussian:



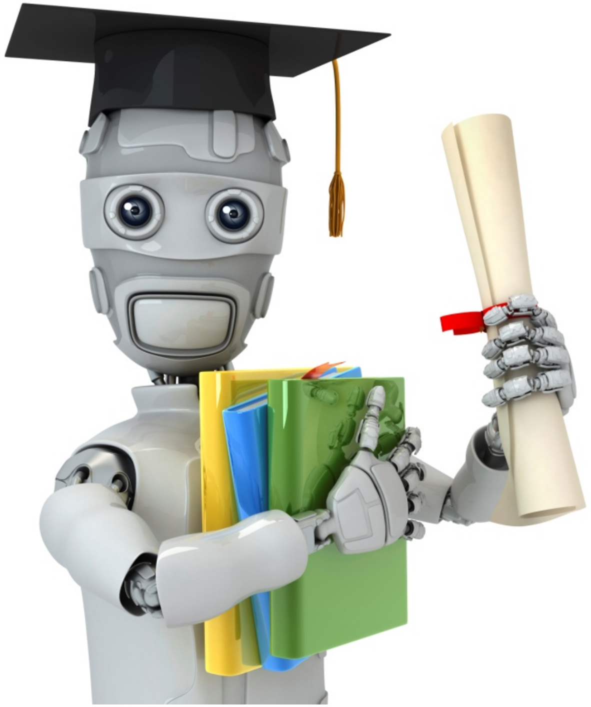


Which of the following are the $\mu$ and $\Sigma$ for this distribution?

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

**Correct**

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



Machine Learning

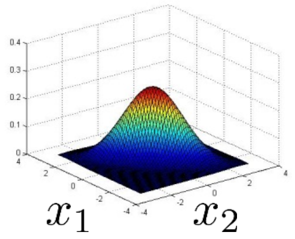
# Anomaly detection

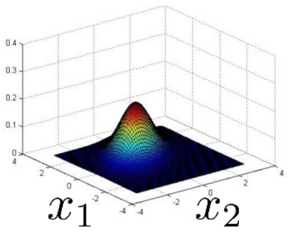
## Anomaly detection using the multivariate Gaussian distribution

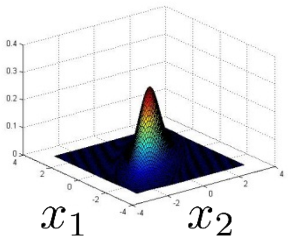
## Multivariate Gaussian (Normal) distribution

Parameters $\underline{\mu, \Sigma}$ $\mu \in \mathbb{R}^n$ $\Sigma \in \mathbb{R}^{n \times n}$

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







Parameter fitting:

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

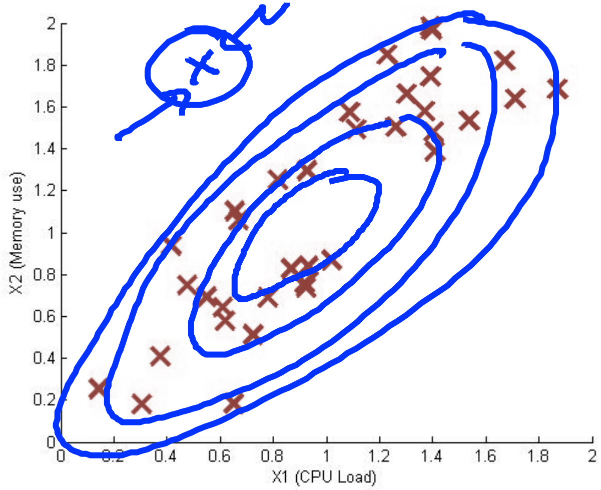
$$\boxed{\mu} = \frac{1}{m}\sum_{i=1}^{m} x^{(i)} \qquad \boxed{\Sigma} = \frac{1}{m}\sum_{i=1}^{m}(x^{(i)}-\mu)(x^{(i)}-\mu)^T$$

# 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

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

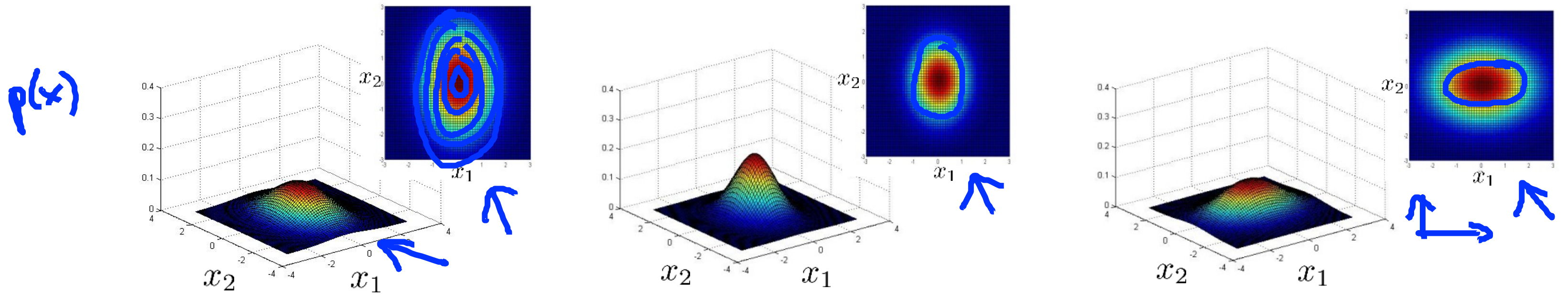
Flag an anomaly if $p(x) < \varepsilon$

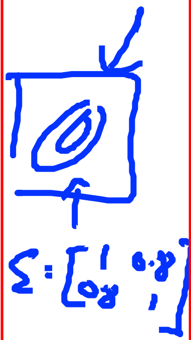
Andrew Ng

the original model corresponds to multivariate Gaussians, where the contours of the Gaussian are always axis aligned; these ellipses have their axes aligned with the X1 X2 ... axes.

## Relationship to original model

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



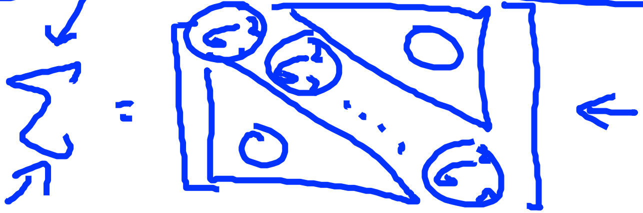


we will not have it using original model

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



if m is less than or equal to n, then this matrix isn't invertible, this matrix is singular,cant use multivariate gaussian

## → Original model vs. → Multivariate Gaussian

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

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

if you have...the original model work fine

Manually create features to capture anomalies where $\underline{x_1}, \underline{x_2}$ take unusual combinations of values.

$$\boxed{x_3=\frac{x_1}{x_2}=\frac{\text{CPU load}}{\text{memory}}}$$

→ Automatically captures correlations between features

disadvantage

$\Sigma \in \mathbb{R}^{n\times n}$ $\quad \underline{\Sigma^{-1}}$

→ Computationally cheaper (alternatively, scales better to large $n$) $n=10{,}000$, $\underline{n=100{,}000}$

Computationally more expensive

if n is large

$\to \Sigma \quad \sim \frac{n^2}{2}$

$\to x_1 = \cancel{x_2}$

$\cancel{x_3} = x_4 + x_5$

OK even if $m$ (training set size) is small

Must have $\boxed{m > n}$ or else $\Sigma$ is non-invertible. $\boxed{m \geq 10n}$

narrow mathematical requirement

in practice,I would use the multivariate Gaussian model, only if m were bigger than n.

in practice the original model shown on the left that is used more often. And if you suspect that you need to capture correlations between features what people will often do is just manually design extra features like these to capture specific unusual combinations of values.

----

But in problems where you have a very large training set or m is very large and n is not too large, then the multivariate Gaussian model is well worth considering and may work better as well, and can save you from having to spend your time to manually create extra features in case the anomalies turn out to be captured by unusual combinations of values of the features.

----

if you're fitting multivariate Gaussian model, and if you find that the covariance matrix sigma is singular (or non-invertible) they're usually 2 cases for this:

(1) it's failing to satisfy this m greater than n condition

(2) you have redundant features. (redundant features=features that are linearly dependent: if you have 2 features that are the same (x1=x2), Or if you have redundant features x3=x4+x5,x3 doesn't contain any extra information)

***

a debugging set--

(1) make sure that M is quite a bit bigger than N

(2) check for redundant features.

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

- [x] The original model $p(x_1; \mu_1, \sigma_1^2) \times \cdots \times p(x_n; \mu_n, \sigma_n^2)$ 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**

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

**Correct**