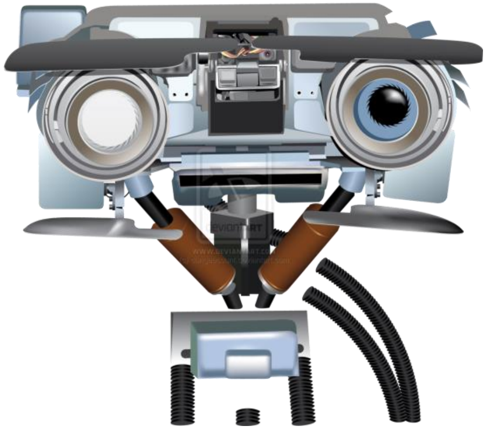
Today: Outline

- **Unsupervised Learning**

- Announcements:

  - Pre-lecture Material 4, *due: Jun 10*
  - Exam Jun 22 in class

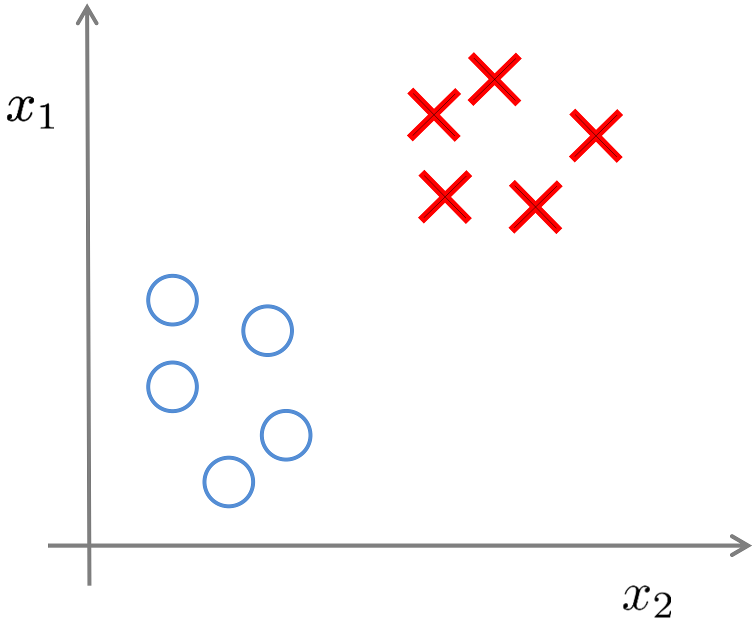
    (and ~12 hrs before for remote only students)



Unsupervised Learning

Clustering

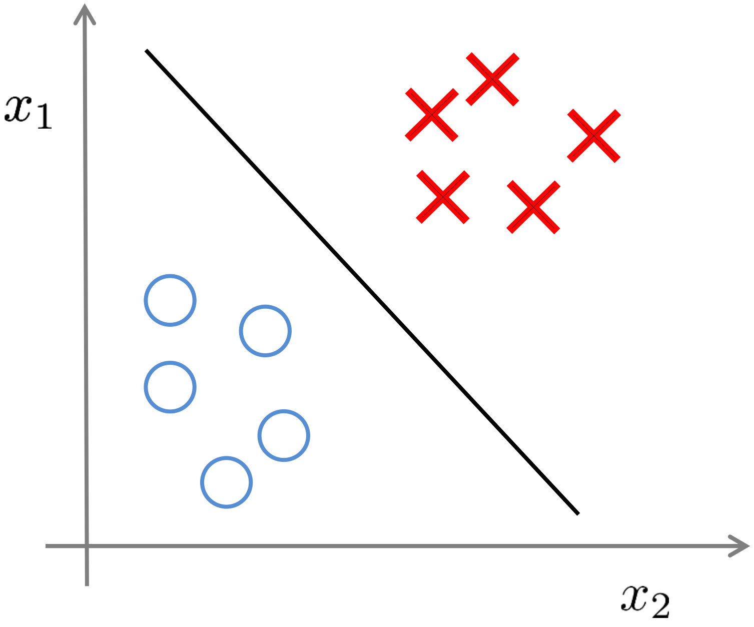
Supervised learning



Training set: $\{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), (x^{(3)}, y^{(3)}), \ldots, (x^{(m)}, y^{(m)})\}$

slide credit: Andrew Ng

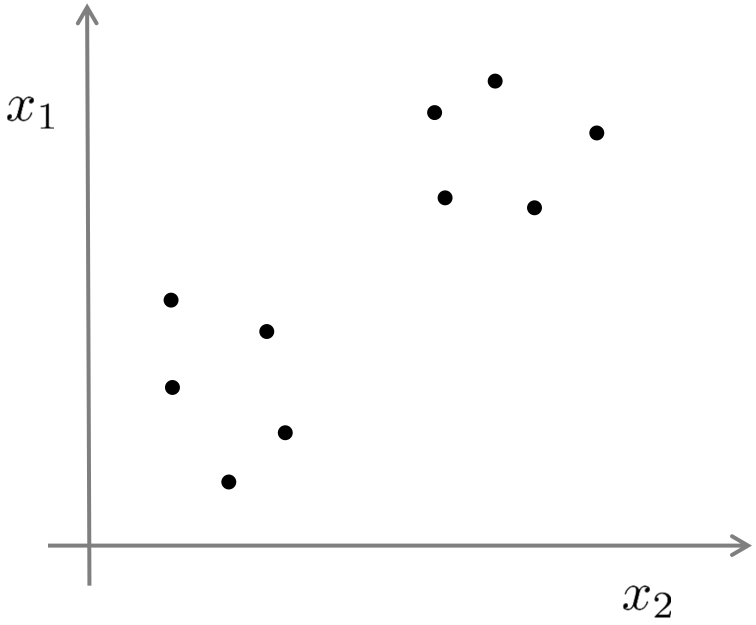
Goal of Supervised learning



Training set: $\{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), (x^{(3)}, y^{(3)}), \ldots, (x^{(m)}, y^{(m)})\}$

slide credit: Andrew Ng

Unsupervised learning

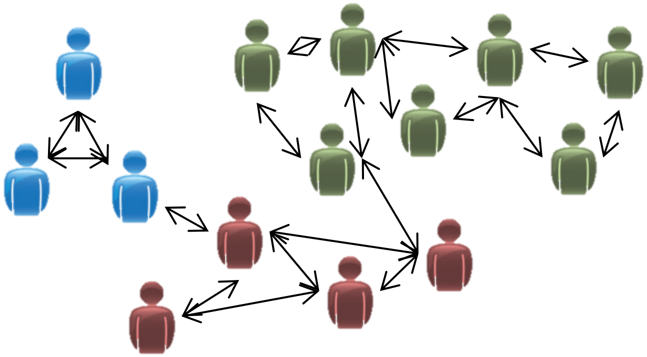


Training set: $\{x^{(1)}, x^{(2)}, x^{(3)}, \ldots, x^{(m)}\}$

slide credit: Andrew Ng

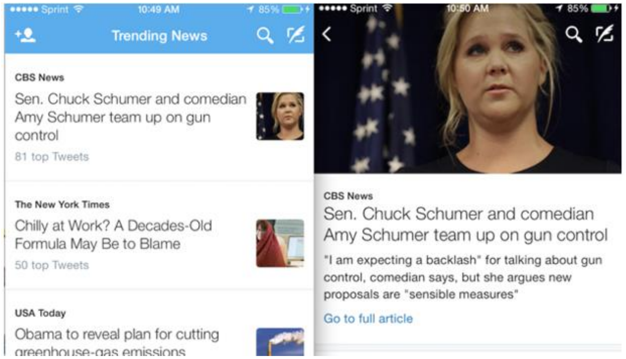
Clustering

Gene analysis



Social network analysis

Types of voters



Trending News

CBS News

Sen. Chuck Schumer and comedian Amy Schumer team up on gun control

81 top Tweets

The New York Times

Chilly at Work? A Decades-Old Formula May Be to Blame

50 top Tweets

USA Today

Obama to reveal plan for cutting greenhouse-gas emissions

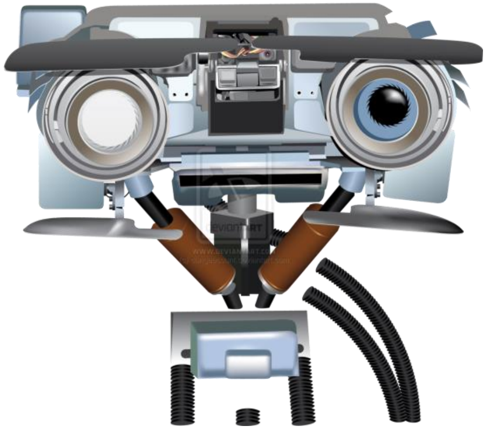
CBS News

Sen. Chuck Schumer and comedian Amy Schumer team up on gun control

"I am expecting a backlash" for talking about gun control, comedian says, but she argues new proposals are "sensible measures"

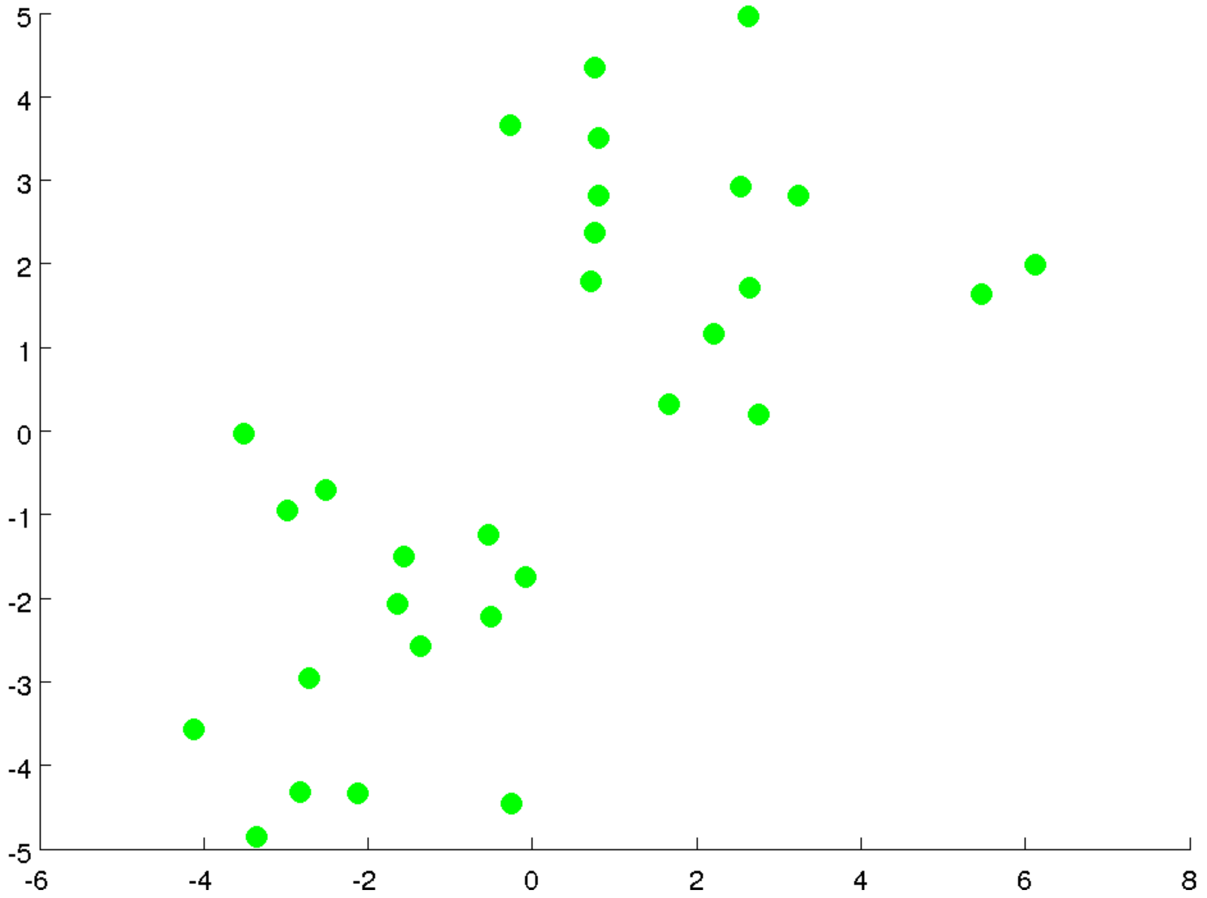
Go to full article

Trending news

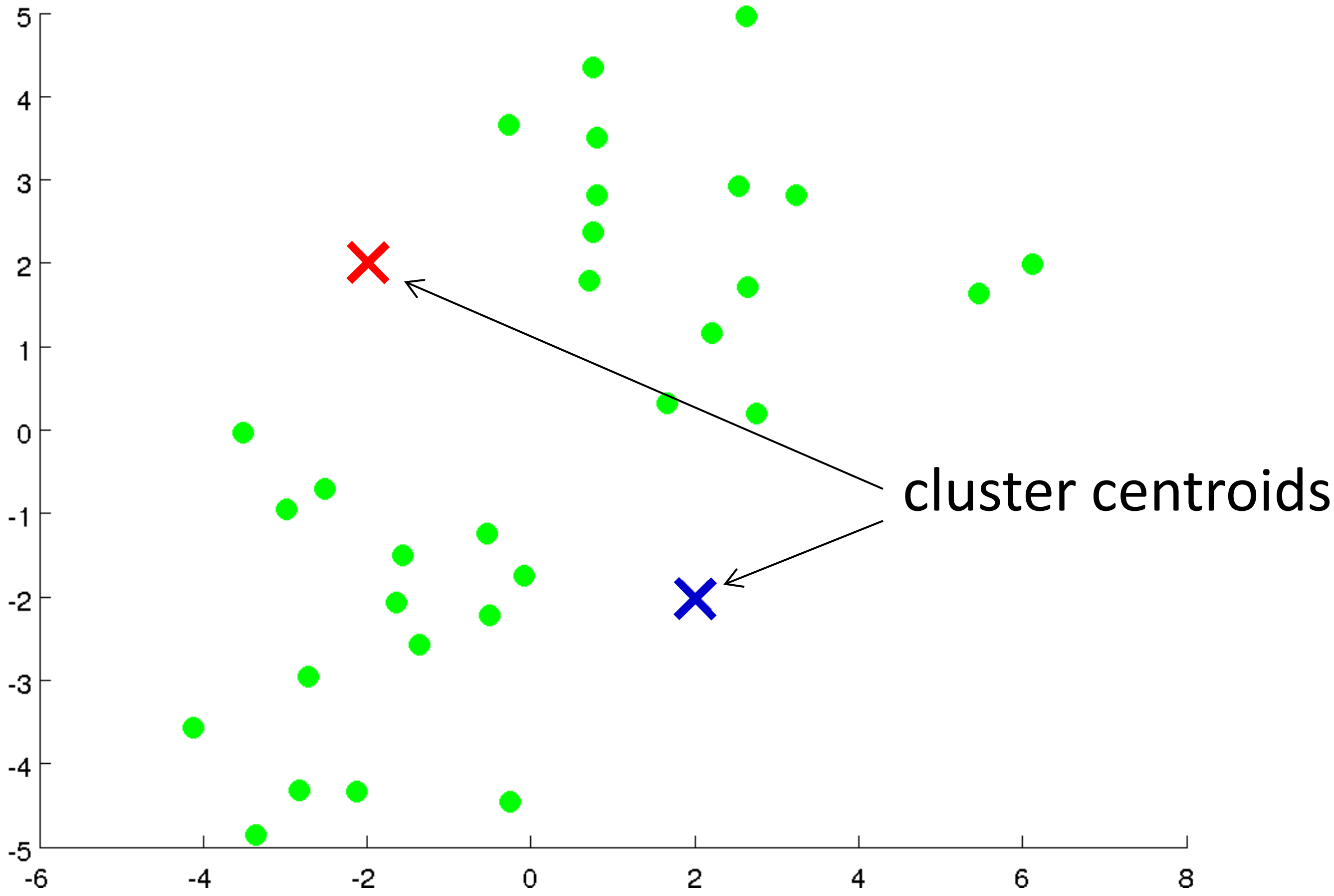


Unsupervised Learning

K-means Algorithm

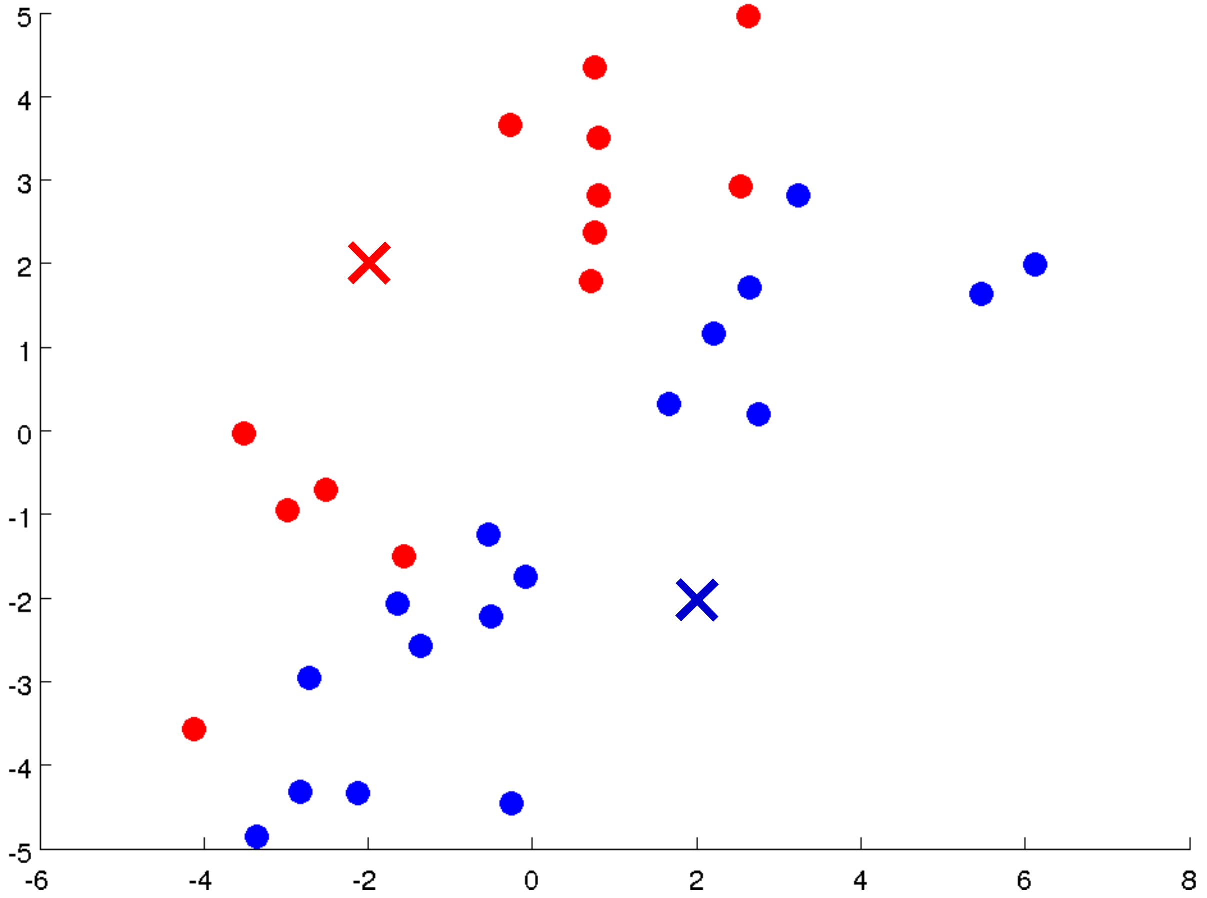


slide credit: Andrew Ng

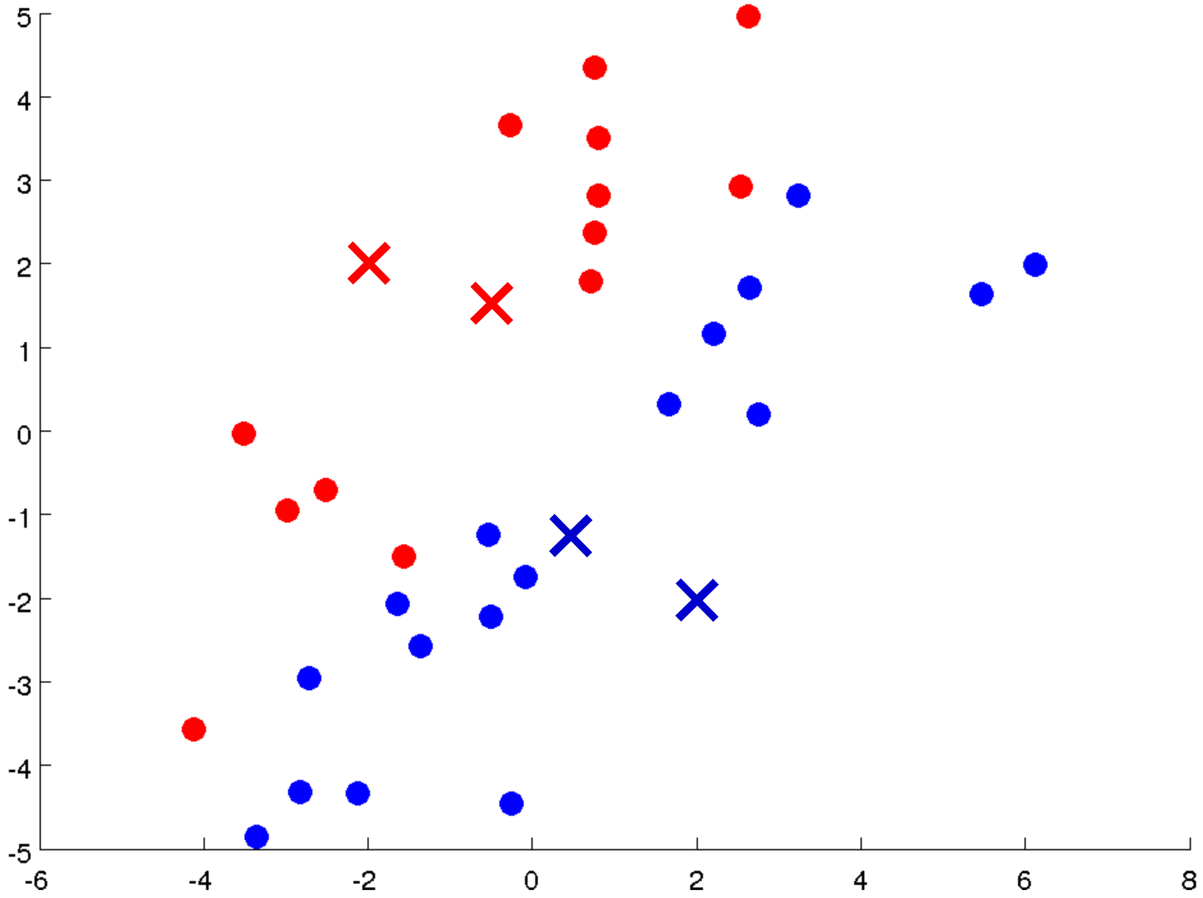


cluster centroids

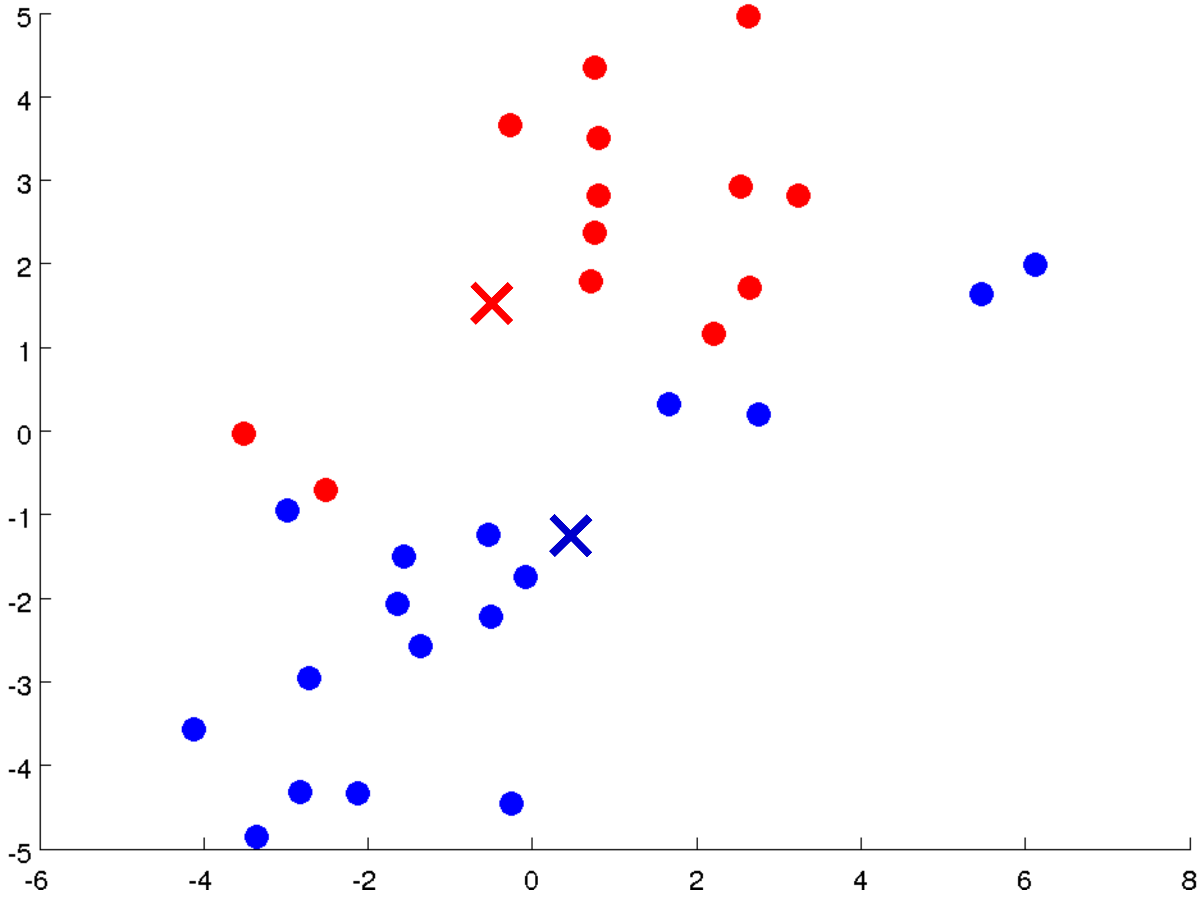
slide credit: Andrew Ng



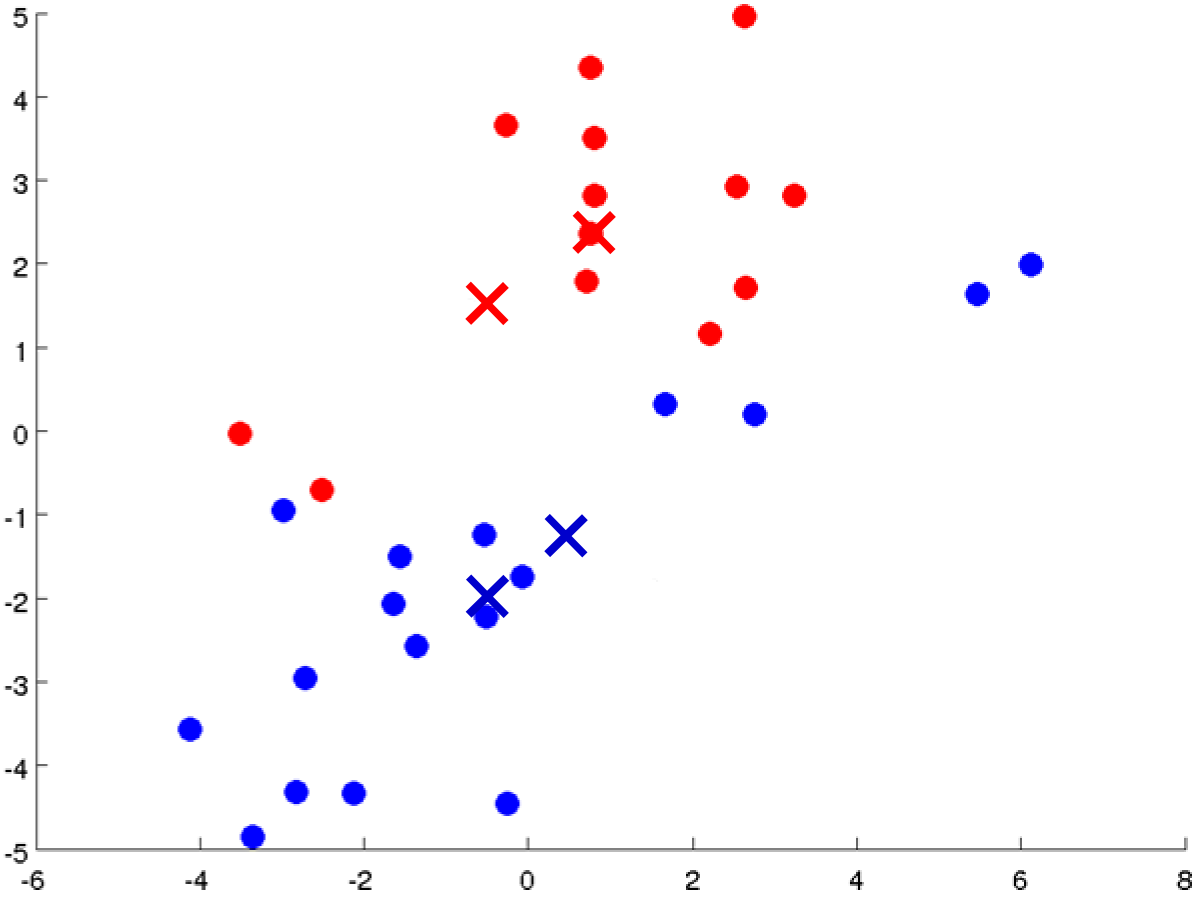
slide credit: Andrew Ng



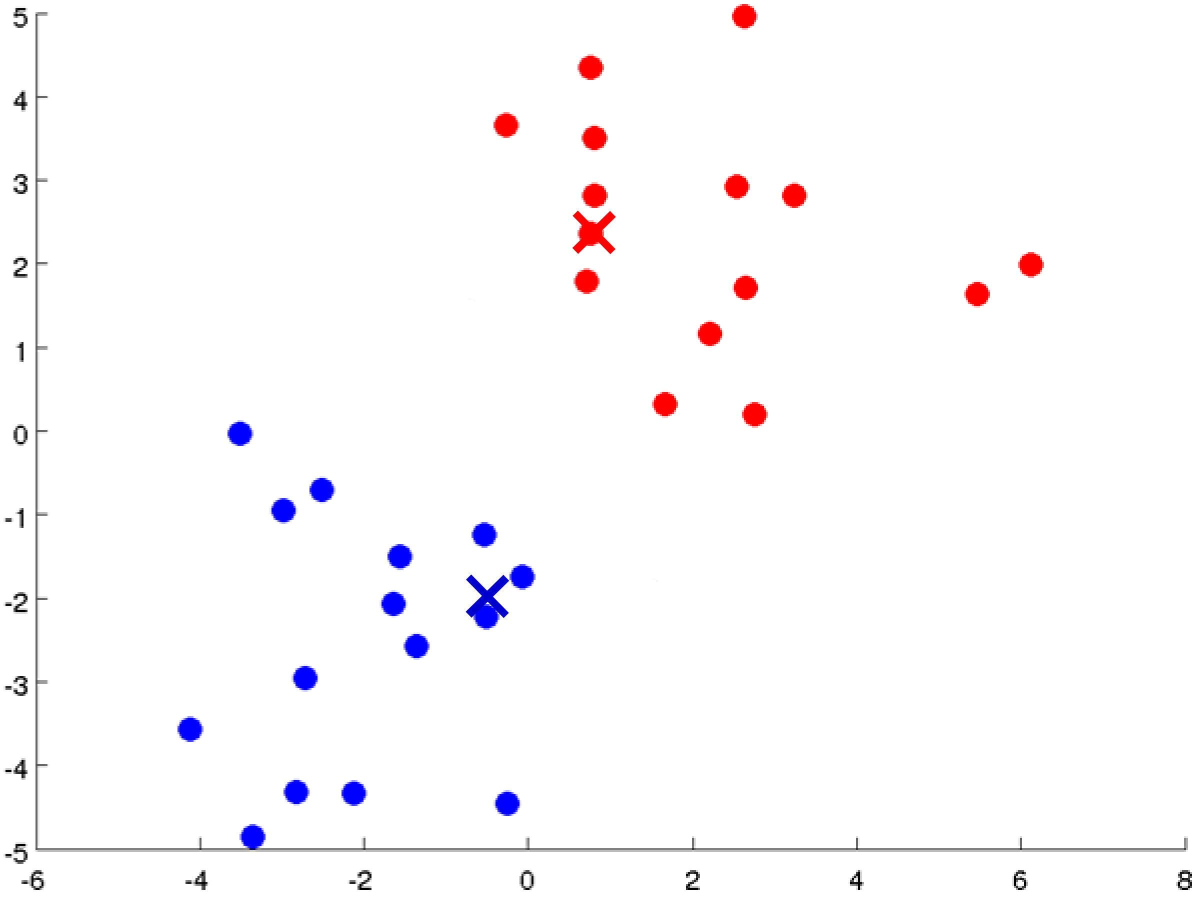
slide credit: Andrew Ng



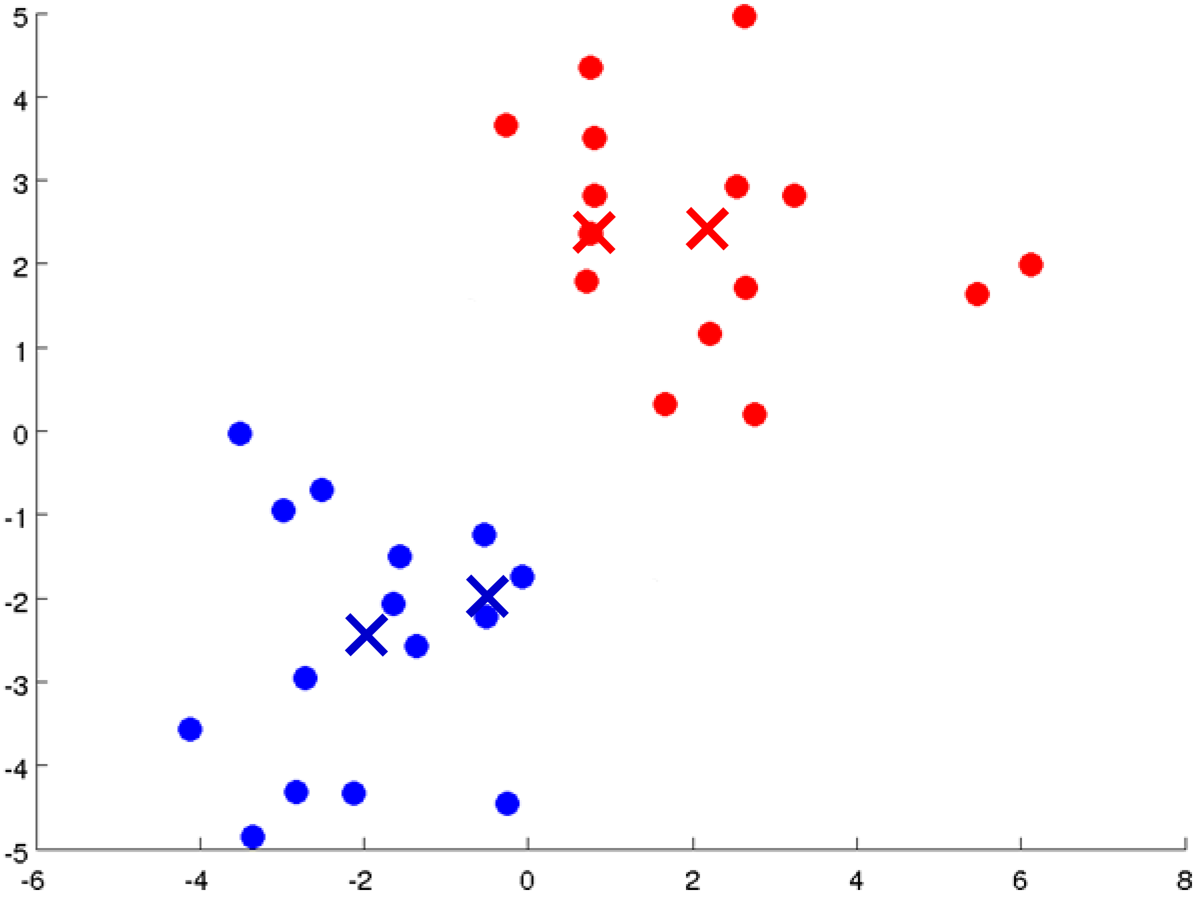
slide credit: Andrew Ng



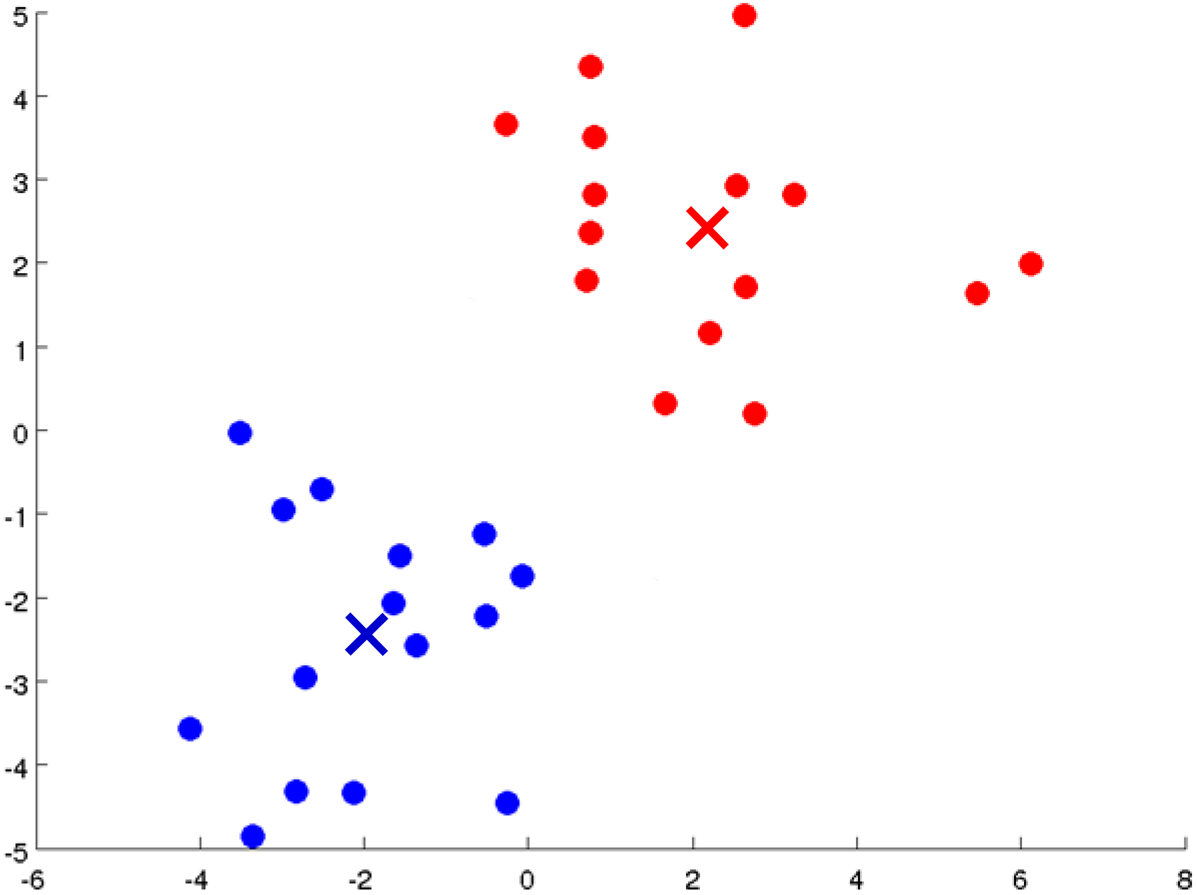
slide credit: Andrew Ng



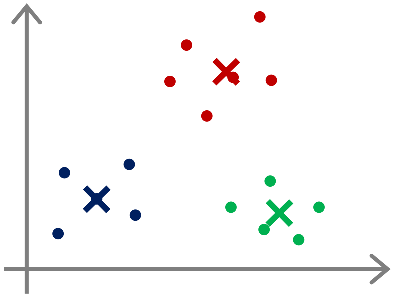
slide credit: Andrew Ng



slide credit: Andrew Ng



slide credit: Andrew Ng



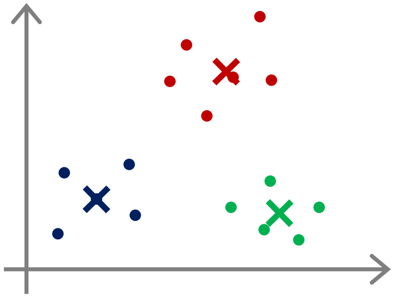
K-means algorithm

Input:

- K (number of clusters)
- Training set $\{x^{(1)}, x^{(2)}, \ldots, x^{(m)}\}$

$x^{(i)} \in \mathbb{R}^n$

slide credit: Andrew Ng



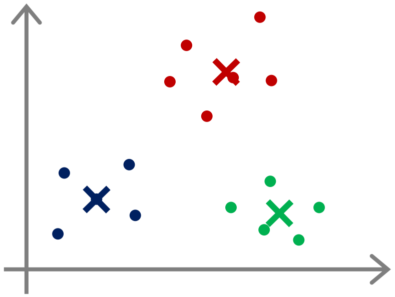
K-means algorithm

Randomly initialize K cluster centroids $\mu_1, \mu_2, \ldots, \mu_K \in \mathbb{R}^n$

```
Repeat {
        for $i$ = 1 to $m$
            $c^{(i)}$ := index (from 1 to $K$) of cluster centroid
                    closest to $x^{(i)}$
        for $k$ = 1 to $K$
            $\mu_k$ := average (mean) of points assigned to cluster $k$
        }
```

slide credit: Andrew Ng

K-means Cost Function



$c^{(i)}$ = index of cluster (1,2,…,K) to which example $x^{(i)}$ is currently assigned

μ_k = cluster centroid k ($\mu_k \in \mathbb{R}^n$)

$\mu_{c^{(i)}}$ = cluster centroid of cluster to which example $x^{(i)}$ has been assigned

Optimization cost: "distortion"

$$J(c^{(1)}, \dots, c^{(m)}, \mu_1, \dots, \mu_K) = \frac{1}{m} \sum_{i=1}^{m} ||x^{(i)} - \mu_{c^{(i)}}||^2$$

$$\min_{\substack{c^{(1)}, \dots, c^{(m)}, \\ \mu_1, \dots, \mu_K}} J(c^{(1)}, \dots, c^{(m)}, \mu_1, \dots, \mu_K)$$

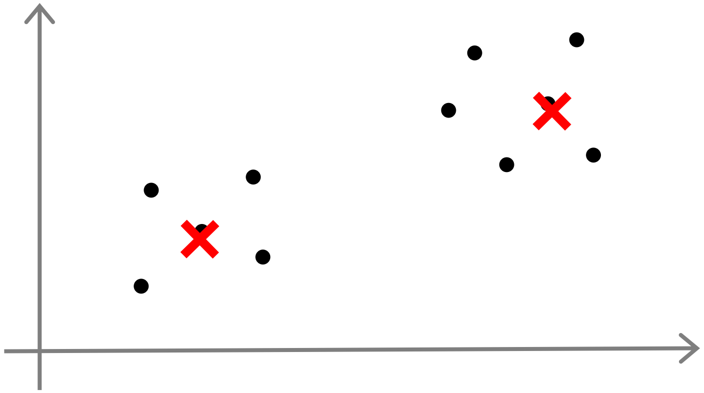
slide credit: Andrew Ng

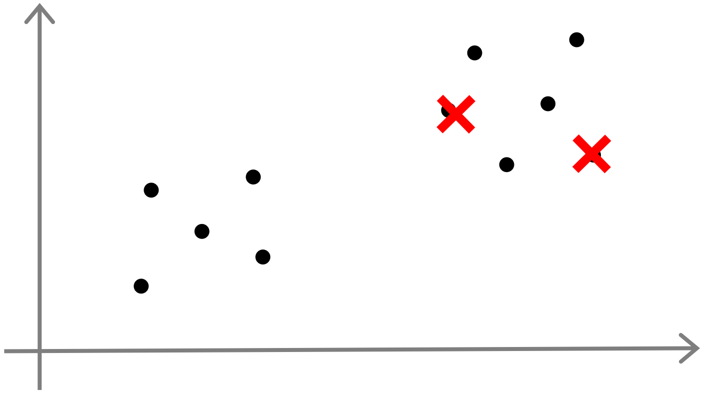
Random initialization

Should have $K < m$

Randomly pick K training examples.

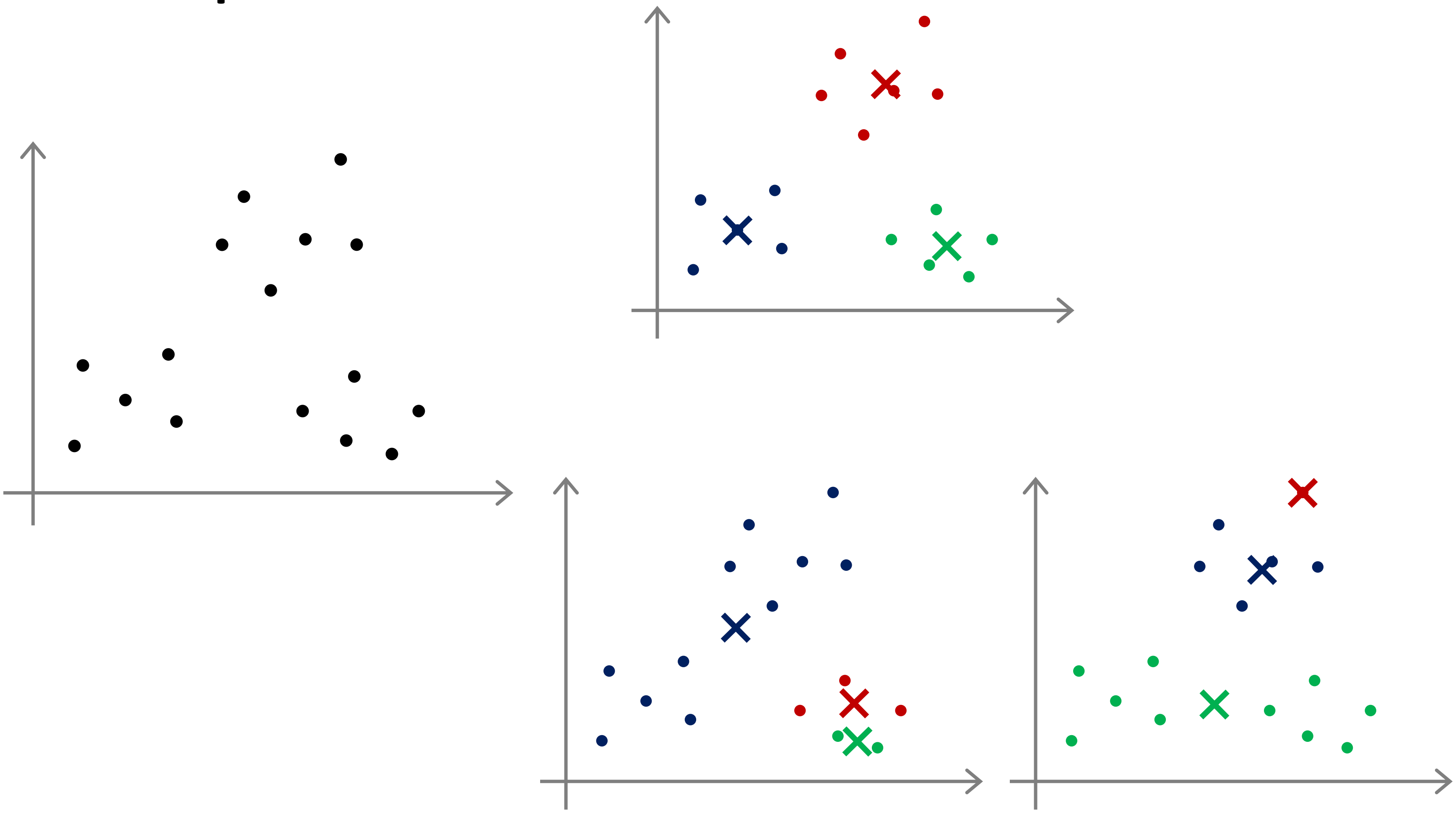
Set $\mu_1, \ldots, \mu_K$ equal to these K examples.





slide credit: Andrew Ng

Local Optima



slide credit: Andrew Ng

Avoiding Local Optima with Random Initialization

```
For i = 1 to 100 {

        Randomly initialize K-means.
        Run K-means. Get $c^{(1)}, \ldots, c^{(m)}, \mu_1, \ldots, \mu_K$.
        Compute cost function (distortion)
            $J(c^{(1)}, \ldots, c^{(m)}, \mu_1, \ldots, \mu_K)$
        }
```

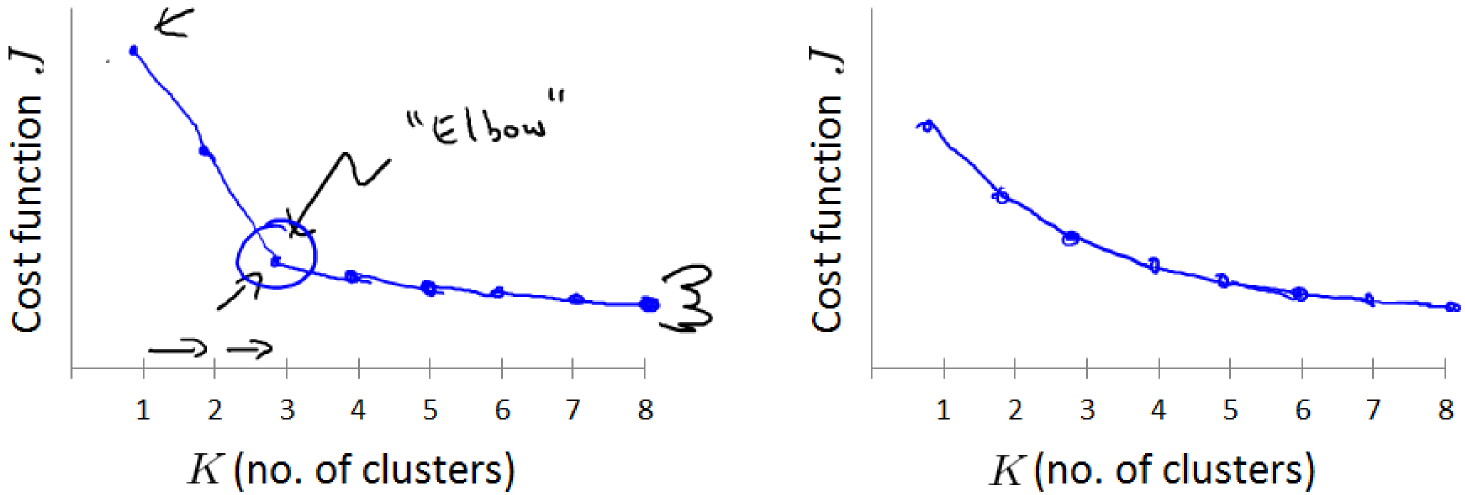
Pick clustering that gave lowest cost $J(c^{(1)}, \ldots, c^{(m)}, \mu_1, \ldots, \mu_K)$

slide credit: Andrew Ng

How to choose K?

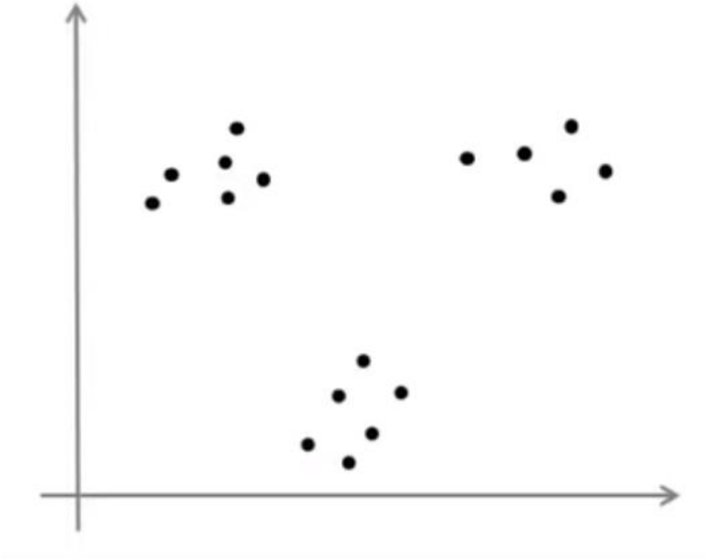
Elbow method:

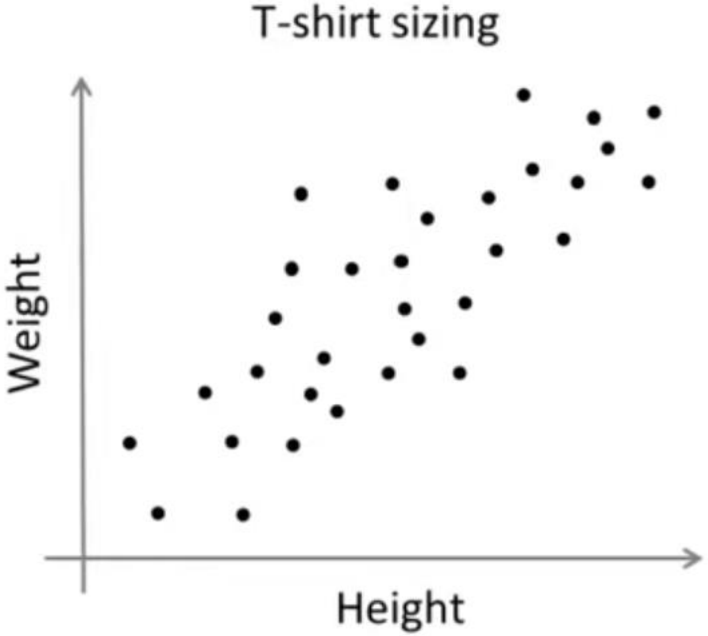
Elbow method:



slide credit: Andrew Ng

K-means for Non-Separated Clusters





slide credit: Andrew Ng

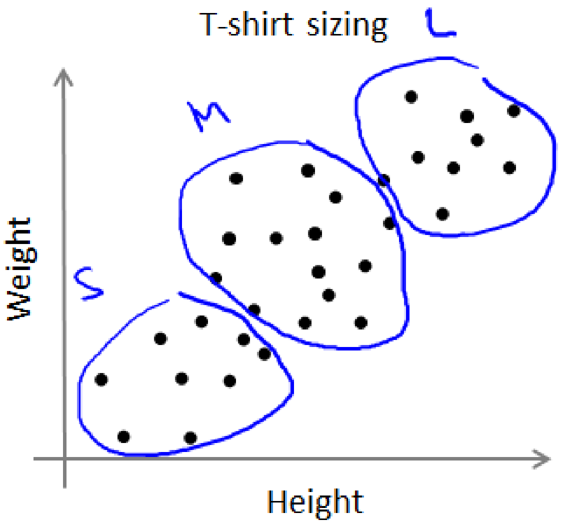
How to choose K?

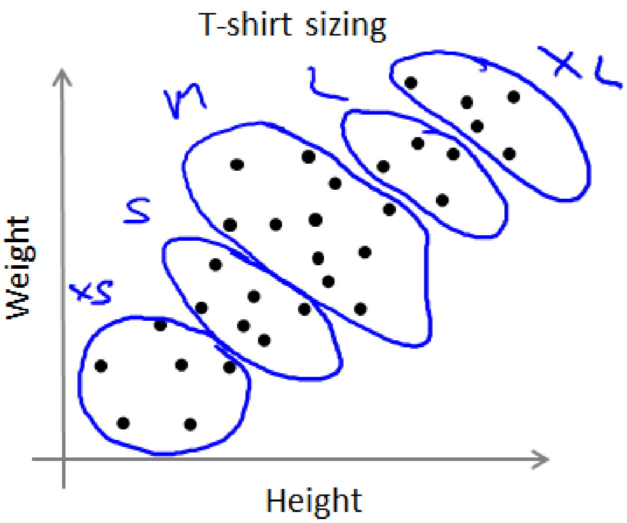
Sometimes, you’re running K-means to get clusters to use for some later/downstream purpose. Evaluate K-means based on a metric for how well it performs for that later purpose.

K=3 S, M, L

K=5 XS, S, M, L, XL

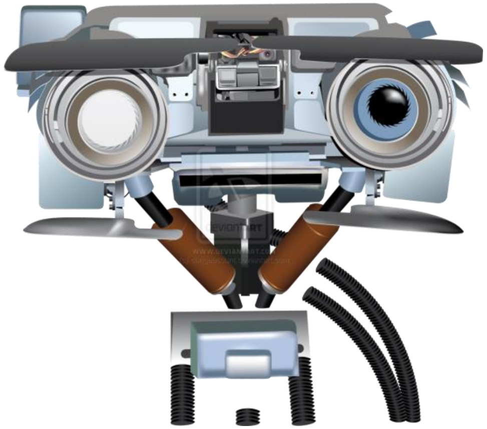
E.g.





Cheaper t-shirts ***vs.*** *Better fitting t-shirts*

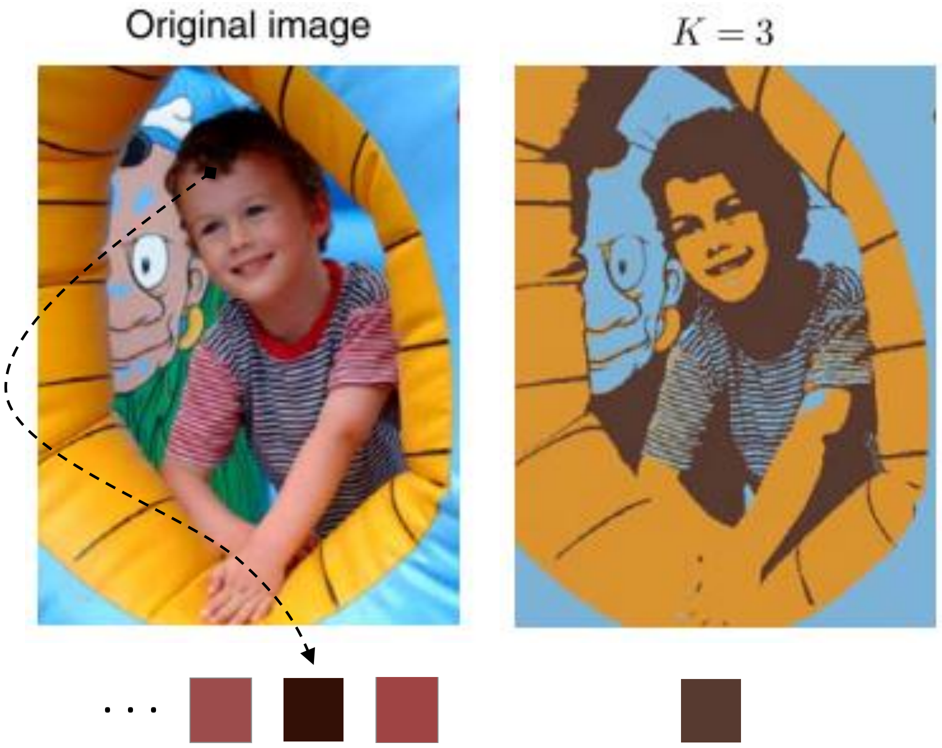
slide credit: Andrew Ng



Unsupervised Learning

Applications of Clustering

Application of Clustering: Vector Quantization



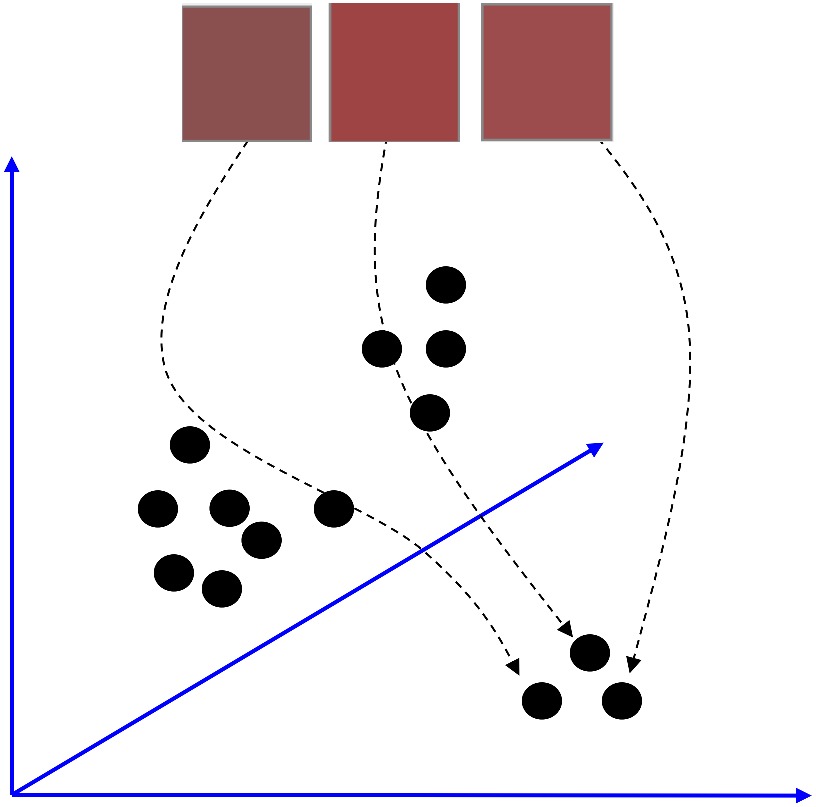
$$x^{(i)} = \begin{bmatrix}138\\80\\79\end{bmatrix} \rightarrow \mu_{c^i}$$

- Compress an image using clustering
- Each {R, G, B} pixel value is an input vector $x^{(i)}$ (255 x 255 x 255 possible values)
- Cluster into K clusters (using k-means)
- Replace each vector by its cluster's index $c^{(i)}$ (K possible values)
- For display, show the mean μ_{c^i}

Vector quantization: color values

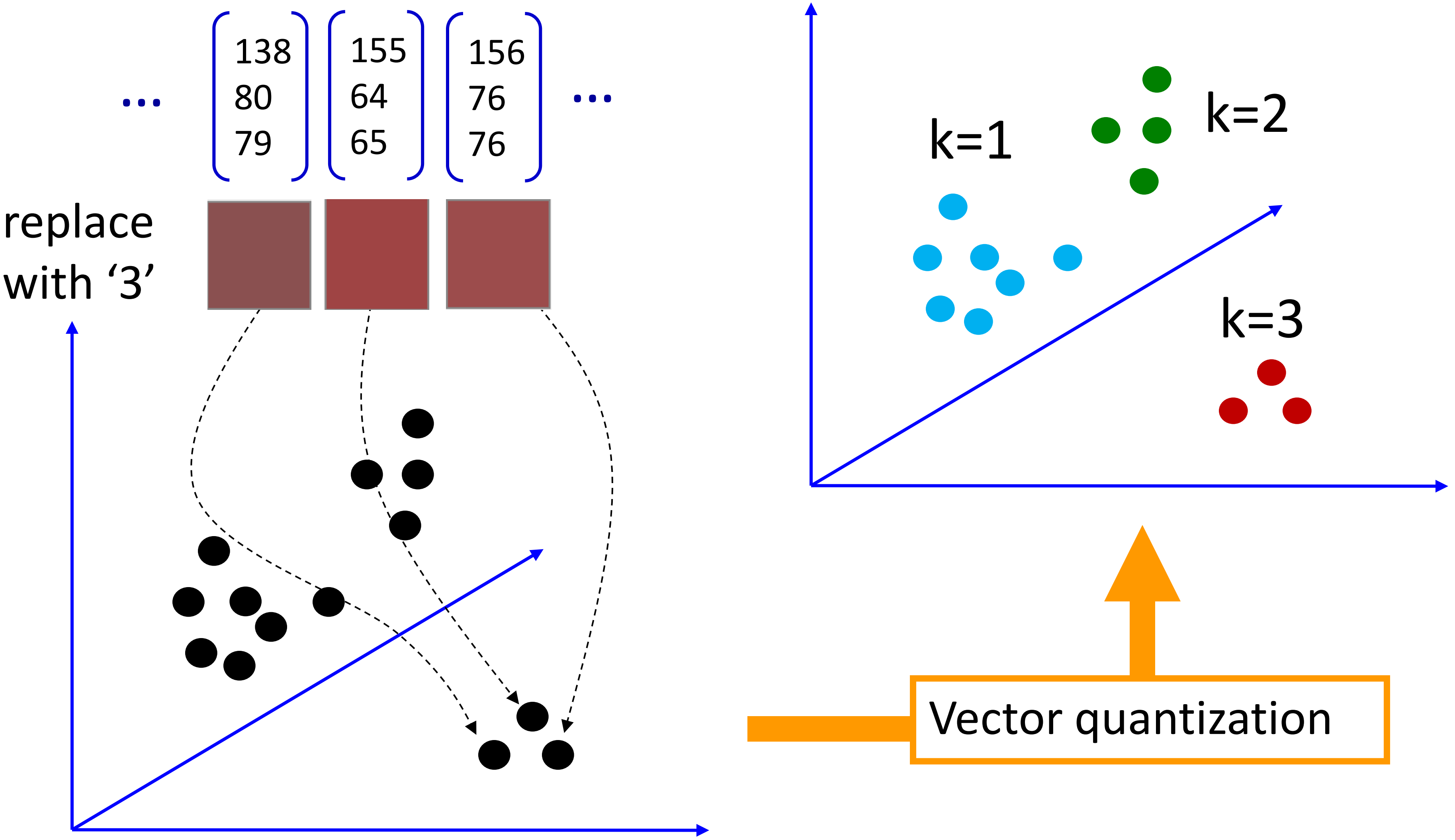
Example: R, G, B vectors

$$\ldots \begin{pmatrix}138\\80\\79\end{pmatrix} \begin{pmatrix}155\\64\\65\end{pmatrix} \begin{pmatrix}156\\76\\76\end{pmatrix} \ldots$$



Slide credit: Josef Sivic

Vector quantization: color values



$$\cdots \begin{pmatrix}138\\80\\79\end{pmatrix} \begin{pmatrix}155\\64\\65\end{pmatrix} \begin{pmatrix}156\\76\\76\end{pmatrix} \cdots$$

replace
with ‘3’

Vector quantization

Slide credit: Josef Sivic

K-Means for Image Compression

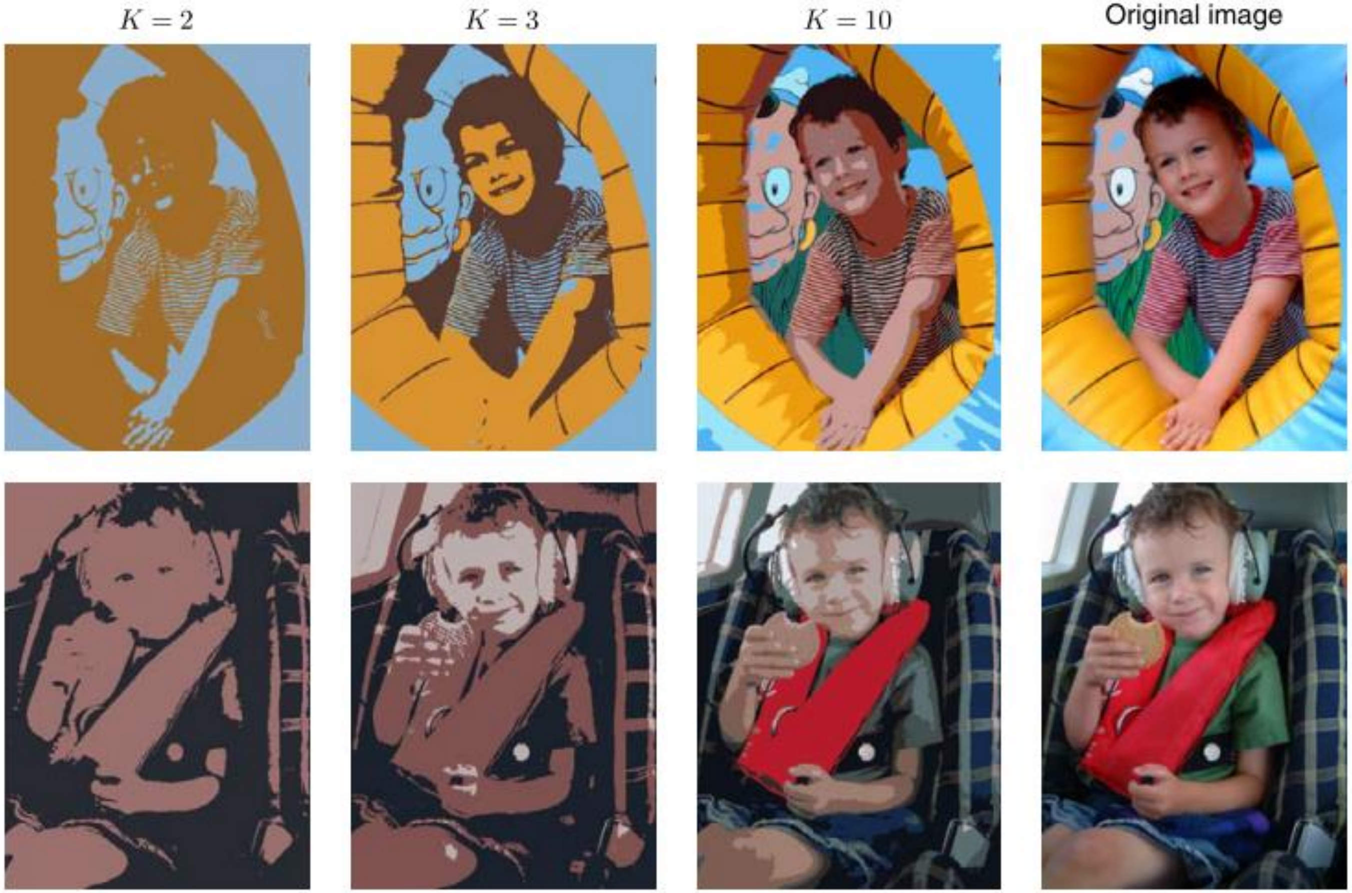
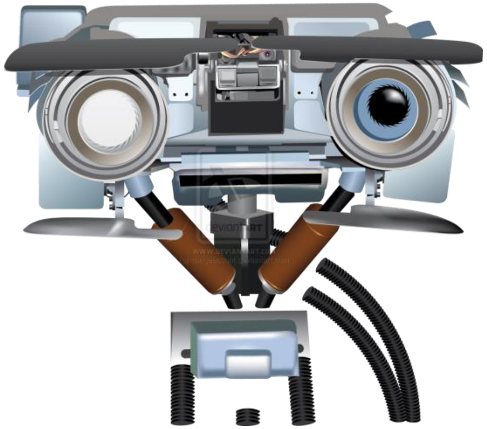


Figure 9.3 Two examples of the application of the K-means clustering algorithm to image segmentation showing the initial images together with their K-means segmentations obtained using various values of K. This also illustrates of the use of vector quantization for data compression, in which smaller values of K give higher compression at the expense of poorer image quality.



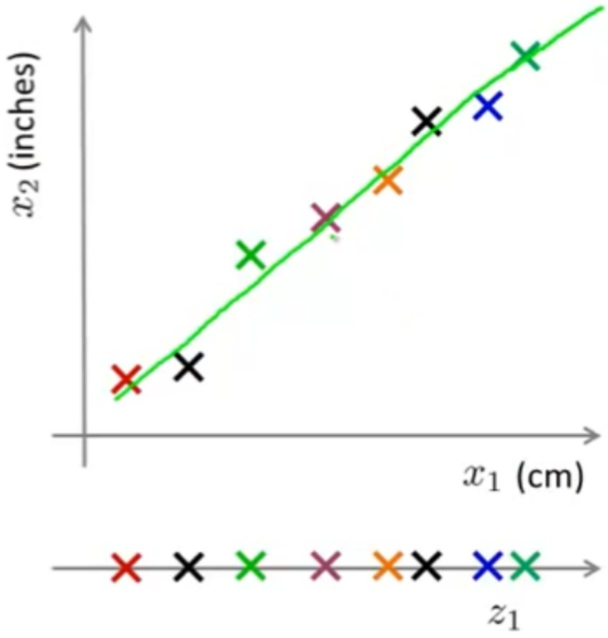
Unsupervised Learning

Dimensionality Reduction

Data Compression

For each data point, we could reduce the memory requirement for storage and having learning algorithms run faster.

E.g.

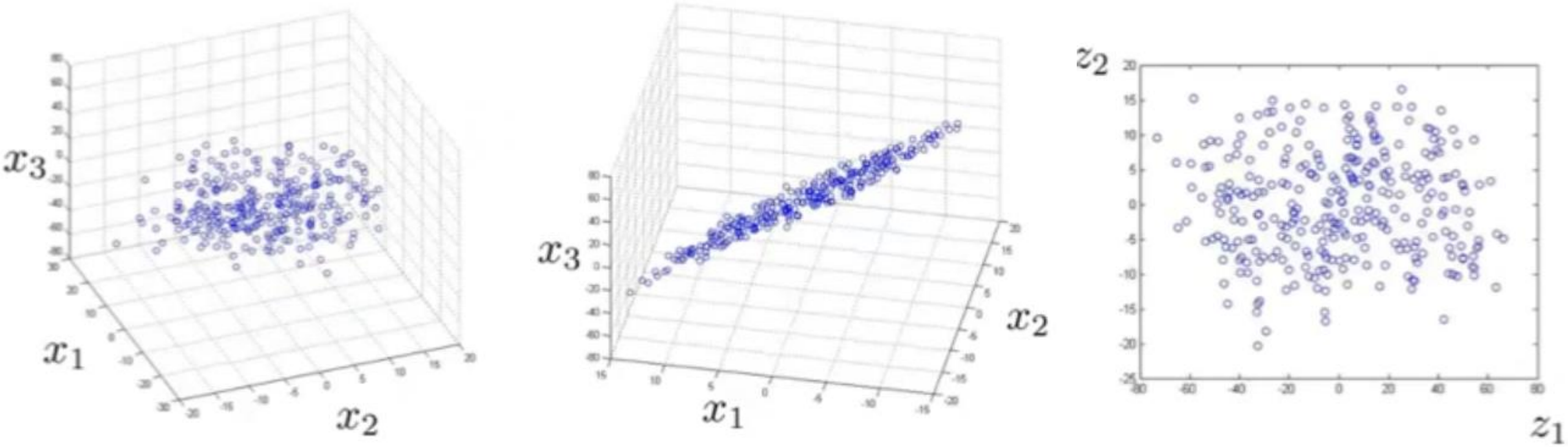


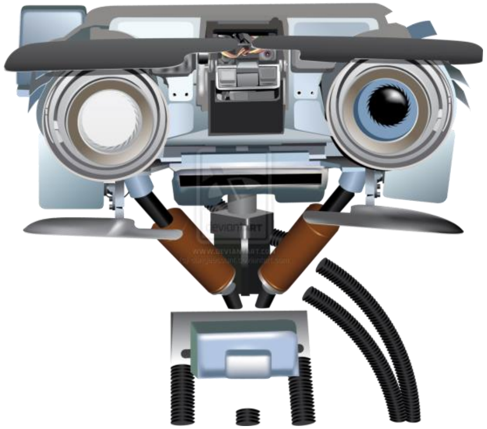
Reduce data from 2D to 1D

slide credit: Andrew Ng

Data Compression

Reduce data from 3D to 2D

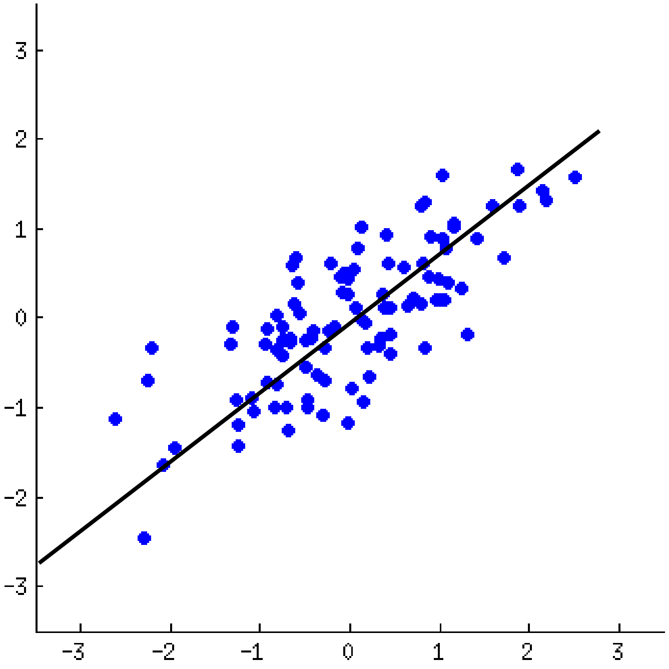


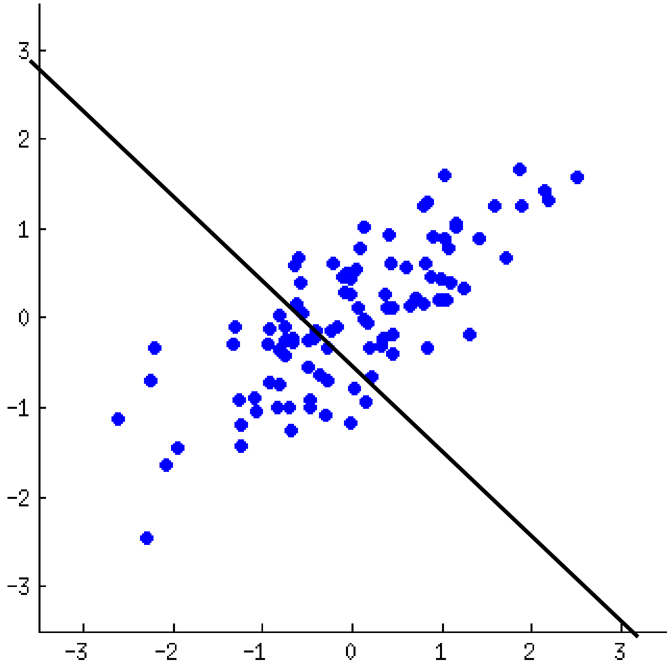


Unsupervised Learning

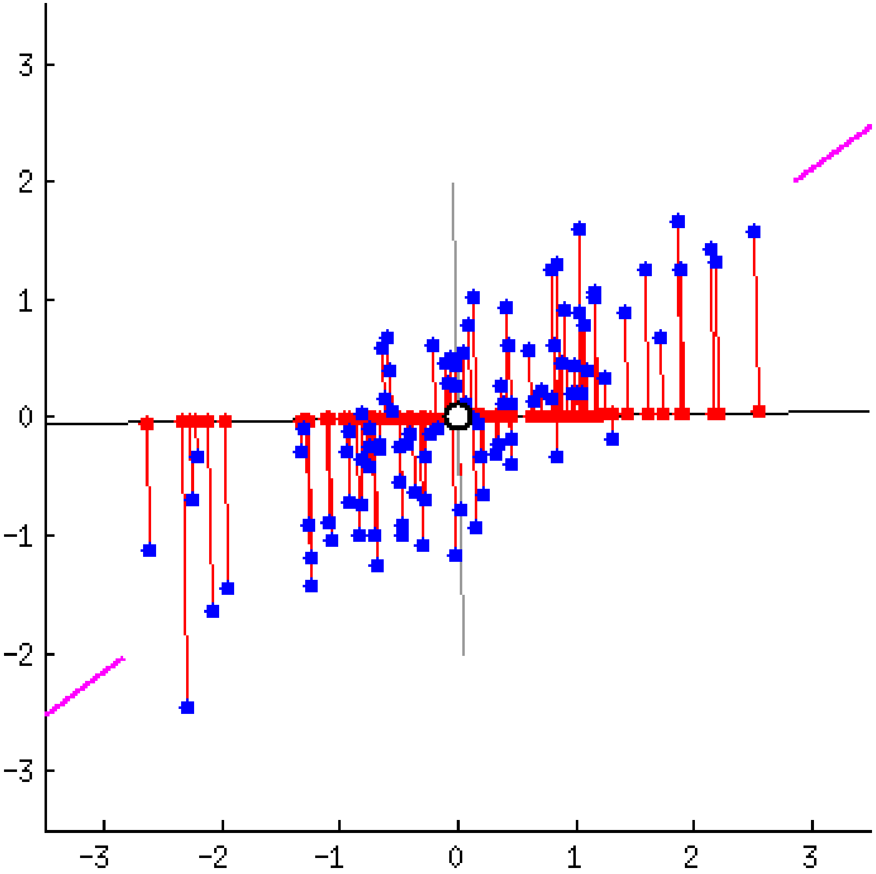
Principal Component Analysis

How to choose lower-dim subspace?

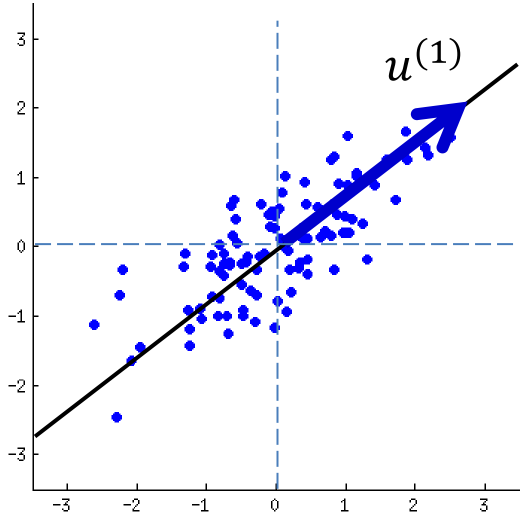




Minimize “error”

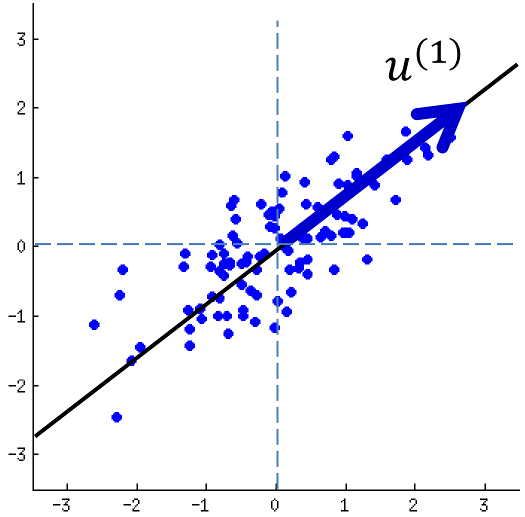


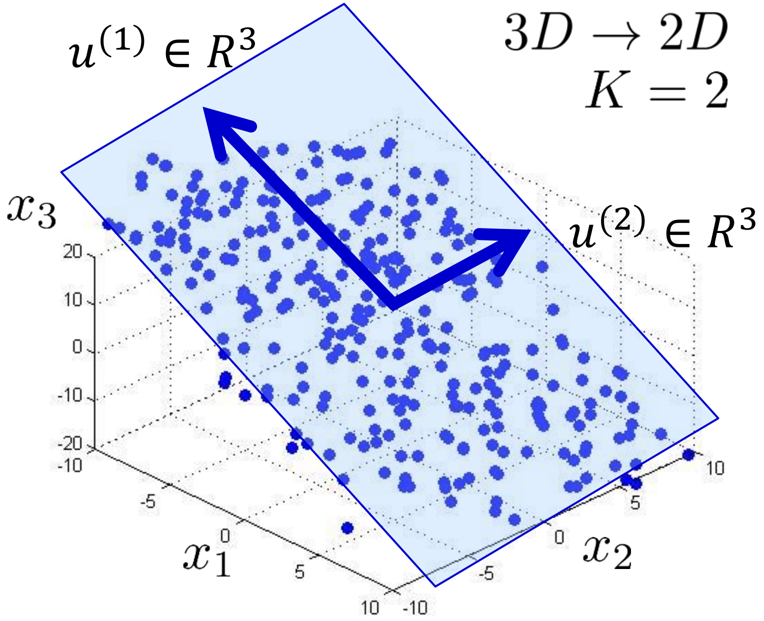
Choose subspace with minimal “information loss”



Reduce from 2-dimension to 1-dimension: Find a direction (a vector $u^{(1)}$) onto which to project the data, so as to minimize the projection error.

Choose subspace with minimal “information loss”



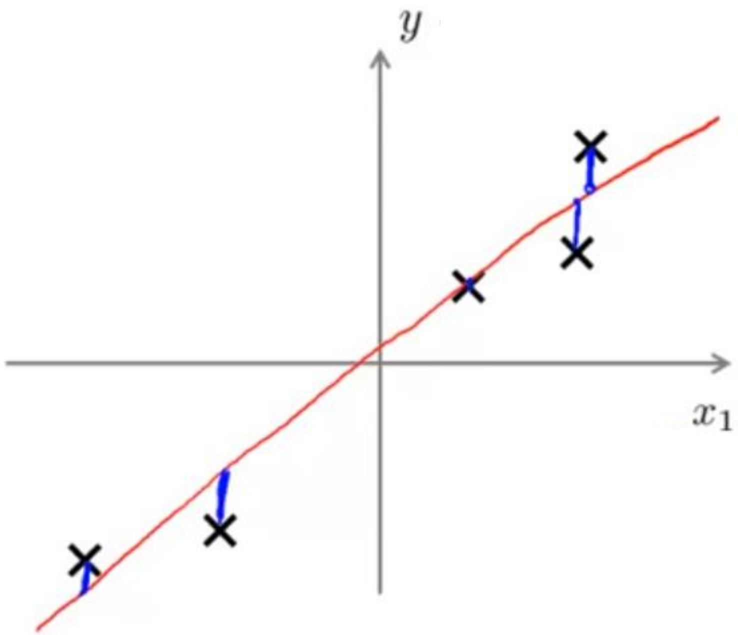


Reduce from 2-dimension to 1-dimension: Find a direction (a vector $u^{(1)}$) onto which to project the data, so as to minimize the projection error.

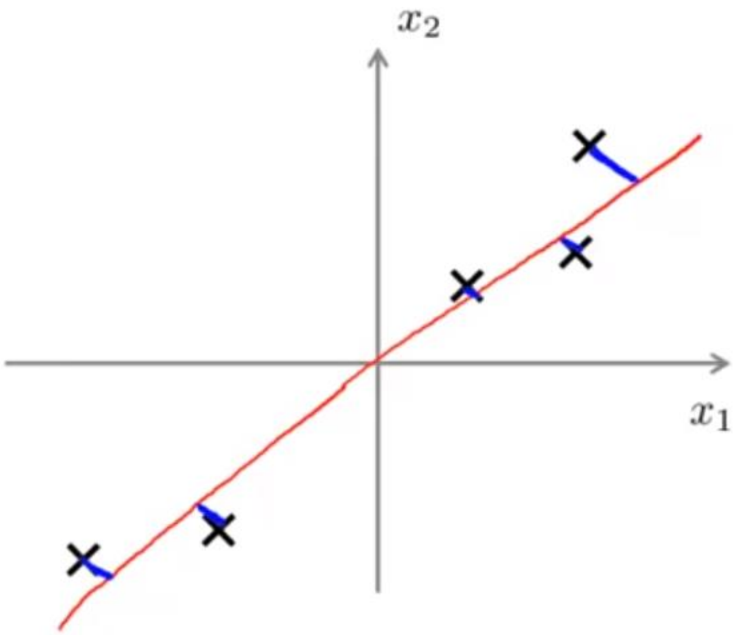
Reduce from n-dimension to K-dimension: Find K vectors $u^{(1)}, u^{(2)}, \dots, u^{(K)}$ onto which to project the data so as to minimize the projection error.

PCA is not Linear Regression

Linear Regression: predicting y



PCA: no predicted variable



PCA Algorithm

Data preprocessing

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

Preprocessing (feature scaling/mean normalization):

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

Replace each $x_j^{(i)}$ with $x_j - \mu_j$.

If different features on different scales (e.g., $x_1 =$ size of house, $x_2 =$ number of bedrooms), scale features to have comparable range of values.

PCA Solution

- The solution turns out to be the first K eigenvectors of the data covariance matrix
  (see Bishop 12.1 for details)

- Closed-form, use Singular Value Decomposition (SVD) on covariance matrix

PCA Algorithm

Normalize features (ensure every feature has zero mean) and optionally scale feature

Compute “covariance matrix” Σ:

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

Compute its “eigenvectors”:

`[U,S,V] = svd(Sigma);` $$U = \begin{bmatrix} | & | & & | \\ u^{(1)} & u^{(2)} & \ldots & u^{(n)} \\ | & | & & | \end{bmatrix} \in \mathbb{R}^{n \times n}$$

Keep first K eigenvectors and project to get new features **`z`**

```
Ureduce = U(:,1:K);
z = Ureduce'*x;
```

Applications for PCA

- Compression
  - Reduce memory/disk needed to store data
  - Speed up learning algorithm

- Visualization

PCA *vs.* t-SNE

Principle Component Analysis

1933

- linear dimension reduction technique
- seeks to maximize variance and preserves large pairwise distances, i.e. things that are different end up far apart.
- This can lead to poor visualization especially when dealing with non-linear manifold structures. Examples: geometric shape like cylinder, ball, curve, etc.

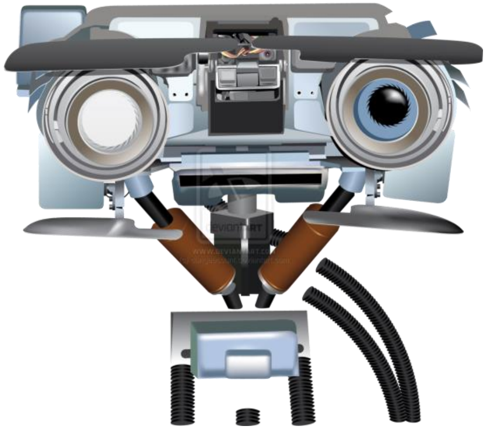
t-Distributed Stochastic Neighbor Embedding (t-SNE)

2008

- Non-linear technique

- t-SNE differs from PCA by preserving only small pairwise distances or local similarities
- t-SNE is not a clustering approach since it does not preserve the inputs like PCA
- the values may often change between runs so it's purely for exploration.

[https://towardsdatascience.com/an-introduction-to-t-sne-with-python-example-5a3a293108d1]



Unsupervised Learning

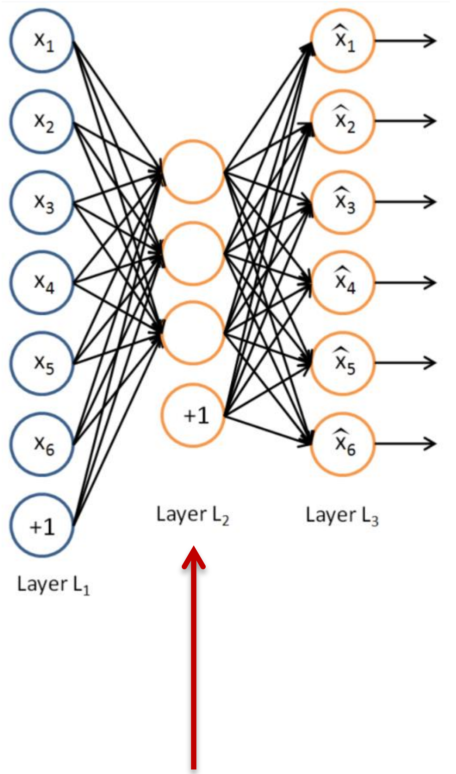
Autoencoders

Autoencoders

- Representation Learning

- We *impose a bottleneck in the network which forces a* ***compressed*** *knowledge representation of the original input*.

- If the input features were each independent of one another, this compression and subsequent reconstruction would be a very difficult task.



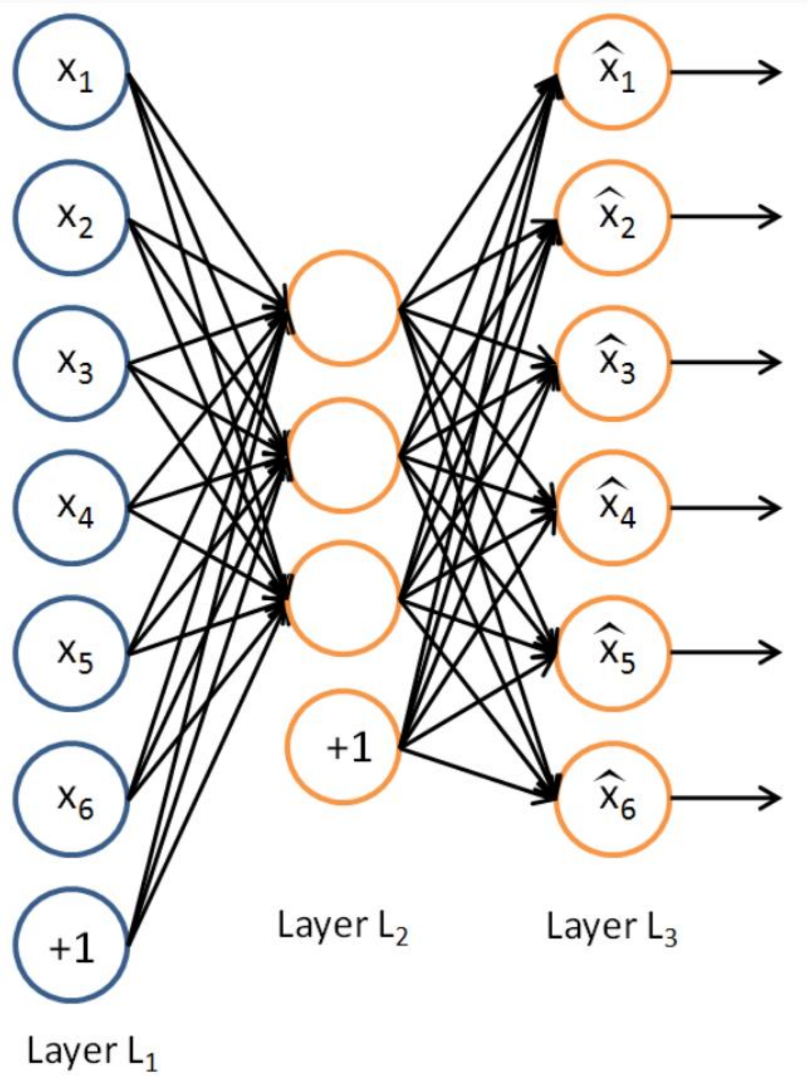
bottleneck

[https://www.jeremyjordan.me/autoencoders/]

Autoencoders

An **autoencoder** neural network is an unsupervised learning algorithm that applies backpropagation, setting the target values to be equal to the inputs.

i.e., it uses $y^{(i)}$=$x^{(i)}$.



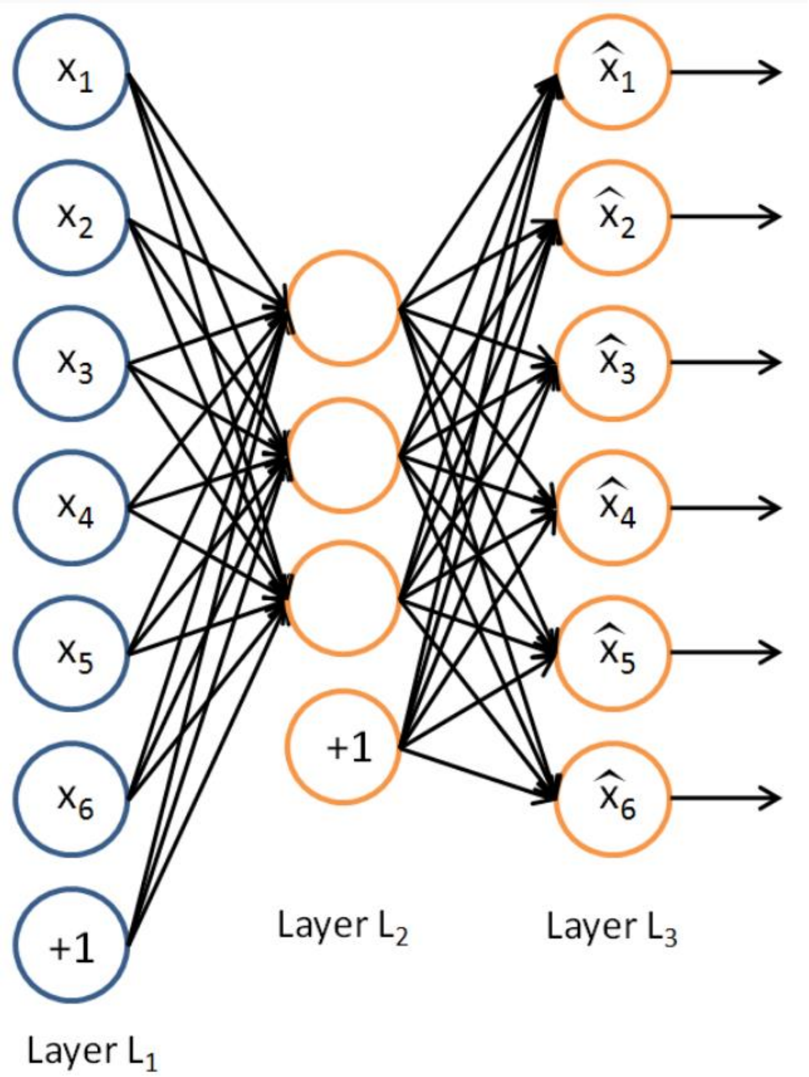
[https://www.jeremyjordan.me/autoencoders/]

Autoencoders

$$\mathcal{L}\left(x, \hat{x}\right)$$

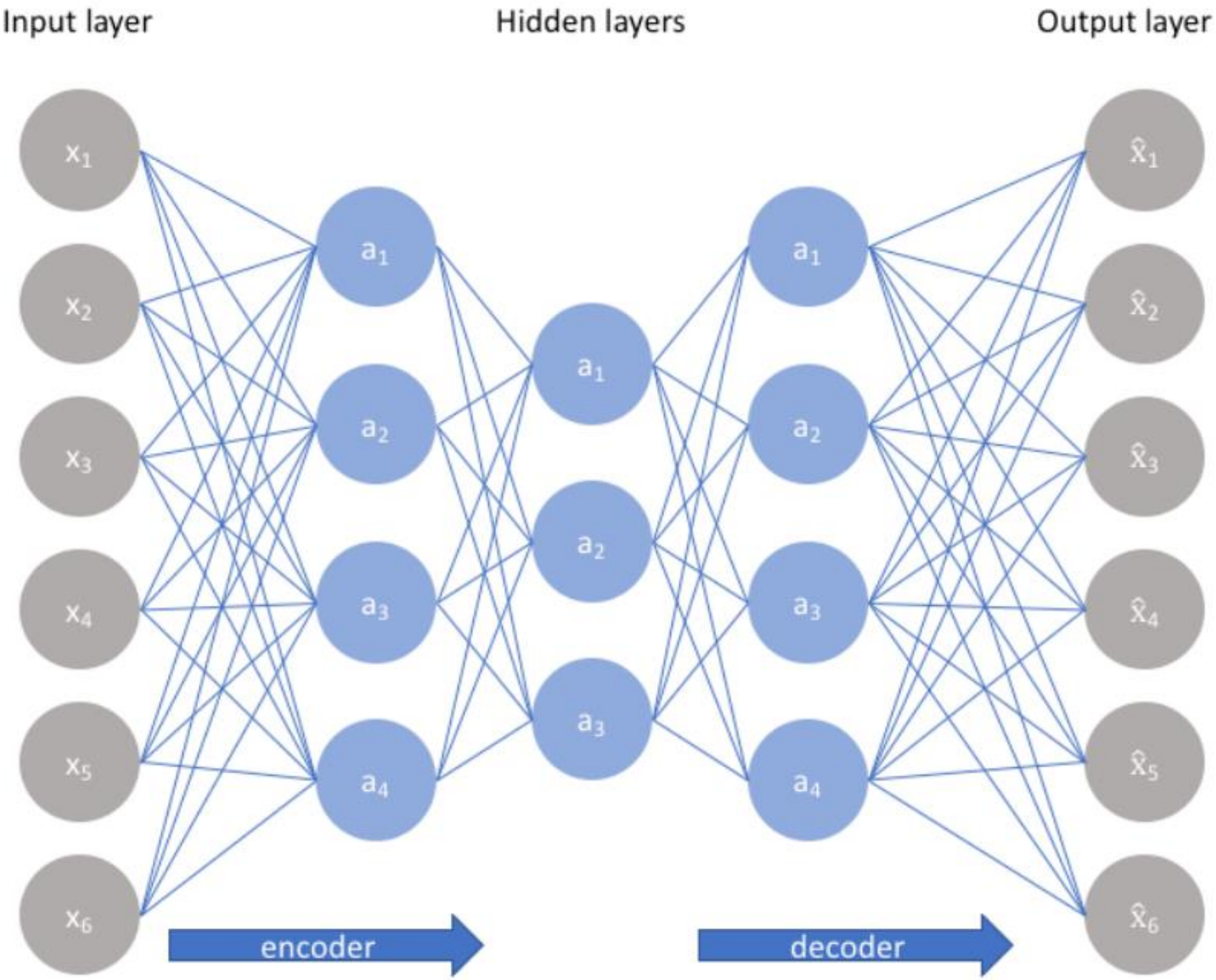
measures the differences between our original input and the consequent reconstruction

$$\mathcal{L}\left(x, \hat{x}\right) + regularizer$$



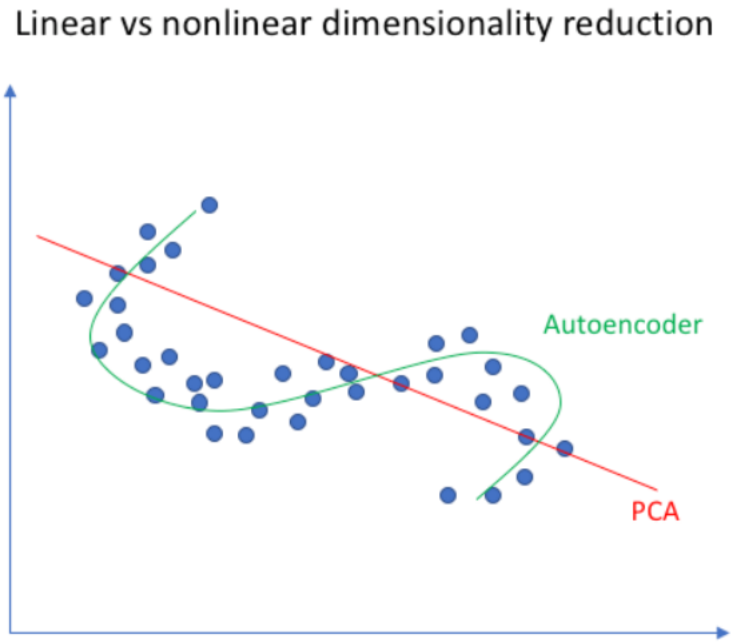
[https://www.jeremyjordan.me/autoencoders/]

Autoencoders



PCA *vs.* Autoencoders

- Because neural networks are capable of learning nonlinear relationships, this can be thought of as a more powerful (nonlinear) generalization of PCA.



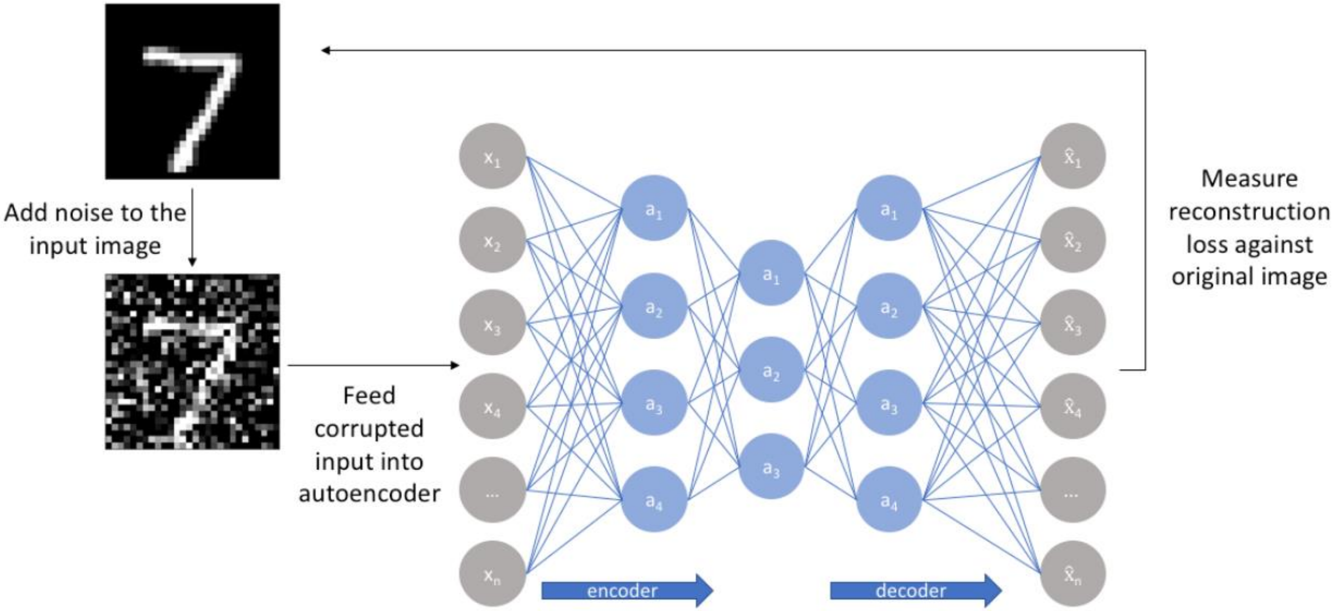
- For higher dimensional data, autoencoders are capable of learning a complex representation of the data (manifold) which can be used to describe observations in a lower dimensionality

[https://www.jeremyjordan.me/autoencoders/]

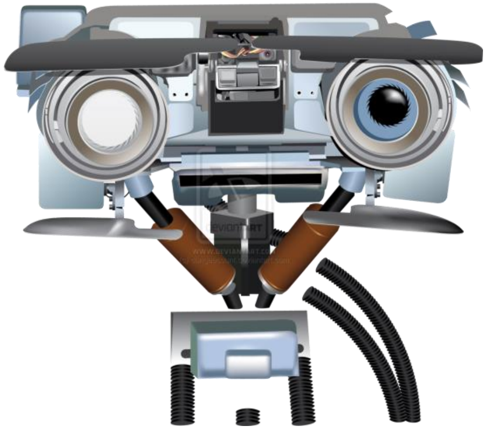
Autoencoders

Applications of autoencoders include:

- Anomaly detection
- Data denoising (ex. images, audio)
- Image inpainting



[https://www.jeremyjordan.me/autoencoders/]

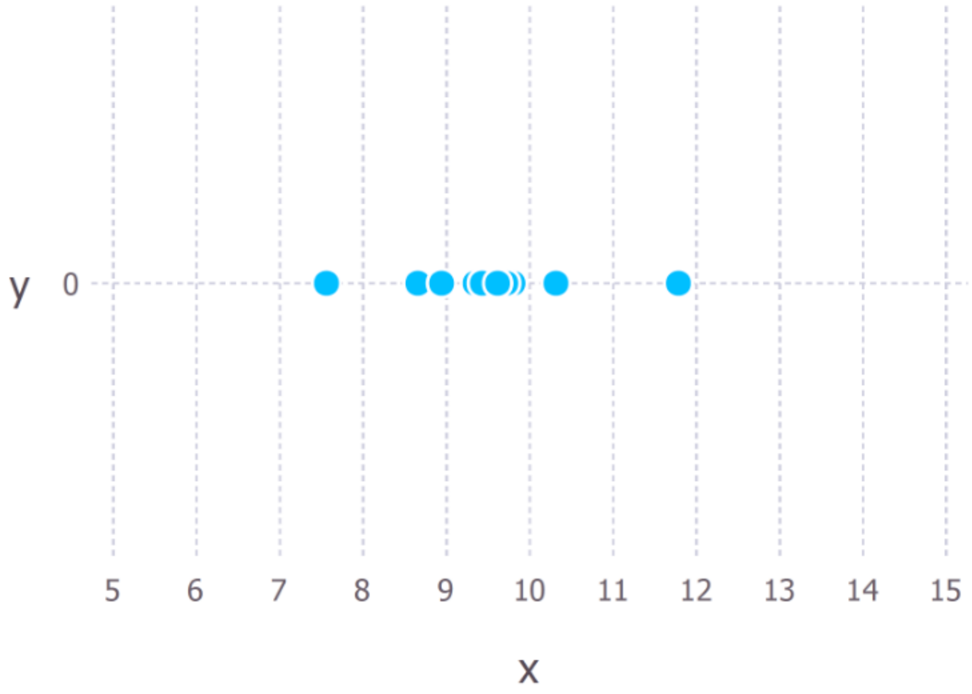


Unsupervised Learning

Mixtures of Gaussians

Observed Data from a Single Gaussian

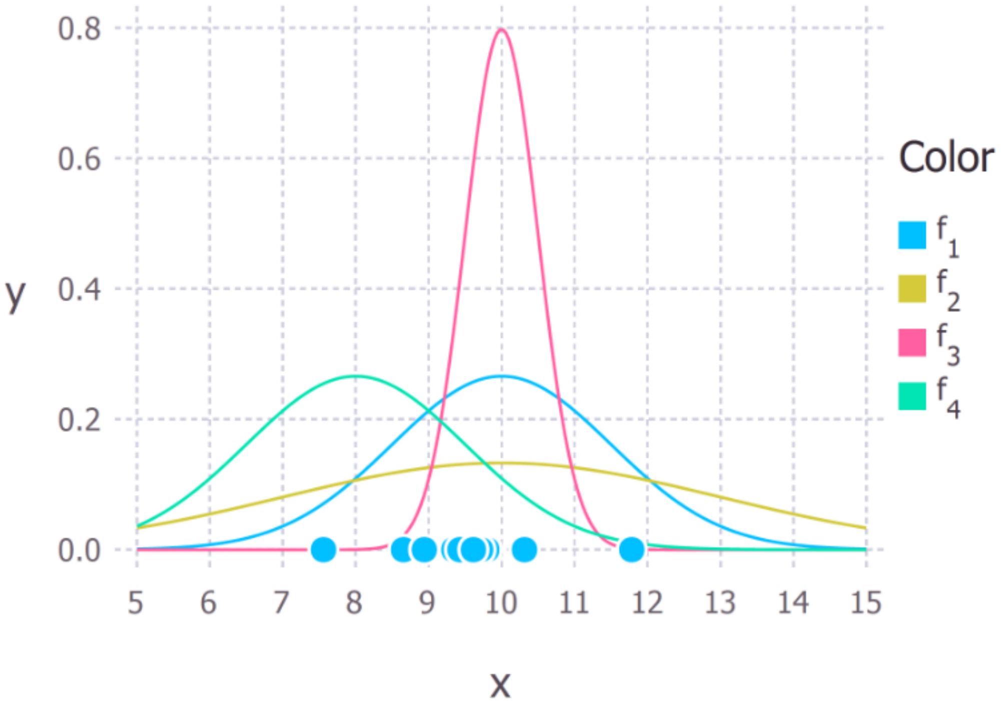
- Ten observed data points from some process



Jonny Brooks-Bartlett

Learning the Model

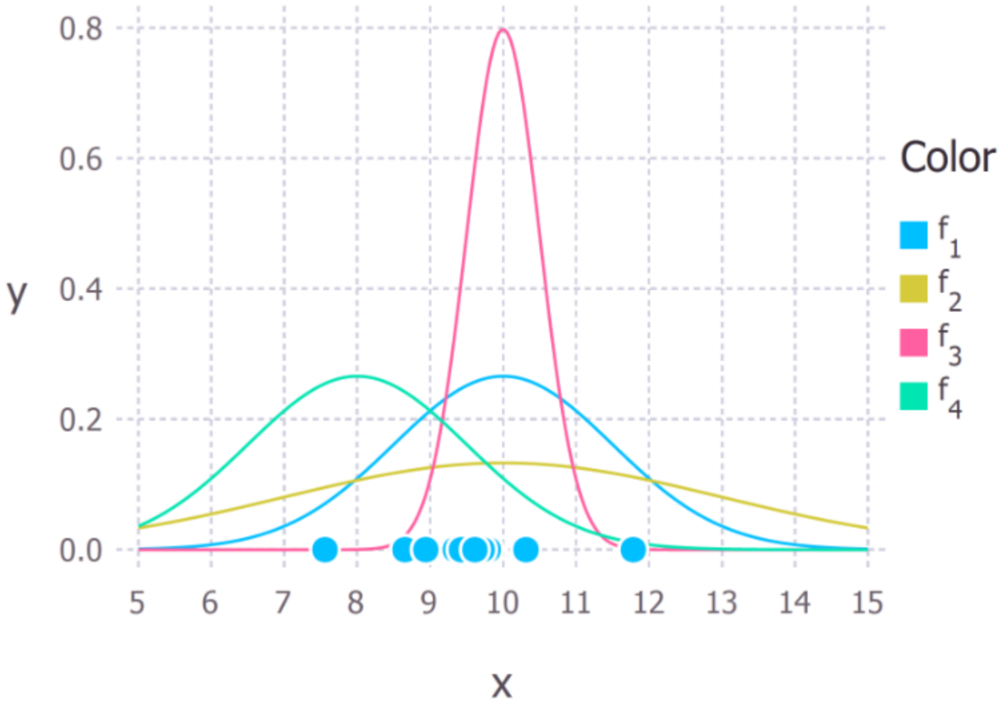
- We want to know *which curve was most likely responsible for creating the data points that we observed?*



Jonny Brooks-Bartlett

Maximum Likelihood

- Maximum likelihood estimation is a method that will find the values of μ and σ that result in the curve that best fits the data.



Jonny Brooks-Bartlett

Calculating Maximum Likelihood Estimates

- What we want to calculate is the total probability of observing all of the data, *i.e.* the joint probability distribution of all observed data points.

- To do this we would need to calculate some conditional probabilities, which can get very difficult.

- So it is here that we'll make our first assumption. *The assumption is that each data point is generated independently of the others*.

Calculating
Maximum Likelihood Estimates

The probability density of observing a single data point *x*, that is generated from a Gaussian distribution is given by:

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

In our example the total (joint) probability density of observing the three data points is given by:

$$P(9, 9.5, 11; \mu, \sigma) = \boxed{\begin{aligned}&\frac{1}{\sigma\sqrt{2\pi}}\exp\left(-\frac{(9-\mu)^2}{2\sigma^2}\right) \times \frac{1}{\sigma\sqrt{2\pi}}\exp\left(-\frac{(9.5-\mu)^2}{2\sigma^2}\right)\\ &\qquad\qquad\qquad\qquad \times \frac{1}{\sigma\sqrt{2\pi}}\exp\left(-\frac{(11-\mu)^2}{2\sigma^2}\right)\end{aligned}}$$

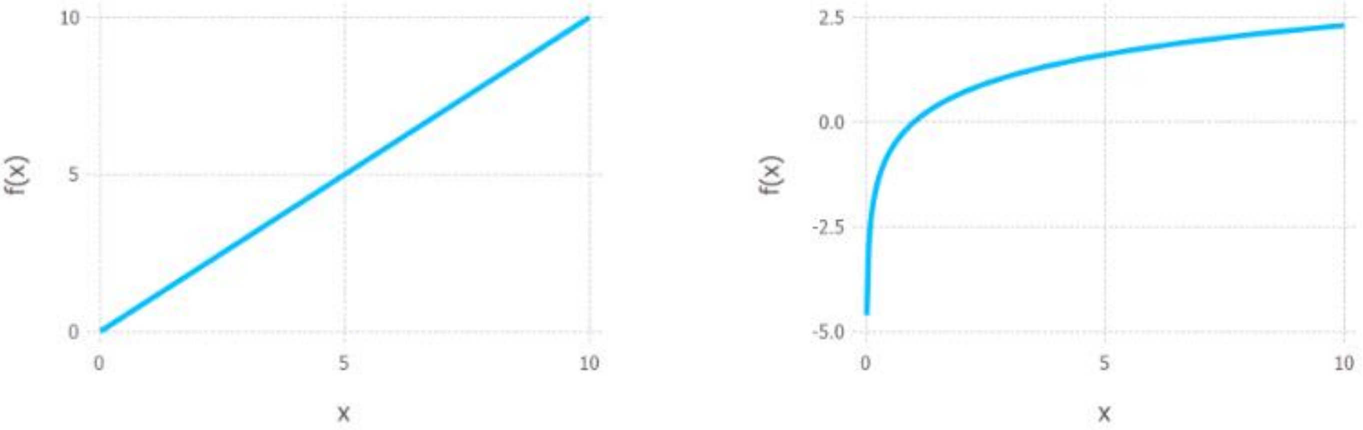
Jonny Brooks-Bartlett

Calculating Maximum Likelihood Estimates

- We need to find the values of μ and σ that results in giving the maximum value of the above expression.
- The above expression for the total probability is difficult to differentiate.
- It is almost always simplified by taking the natural logarithm of the expression.

Log Likelihood

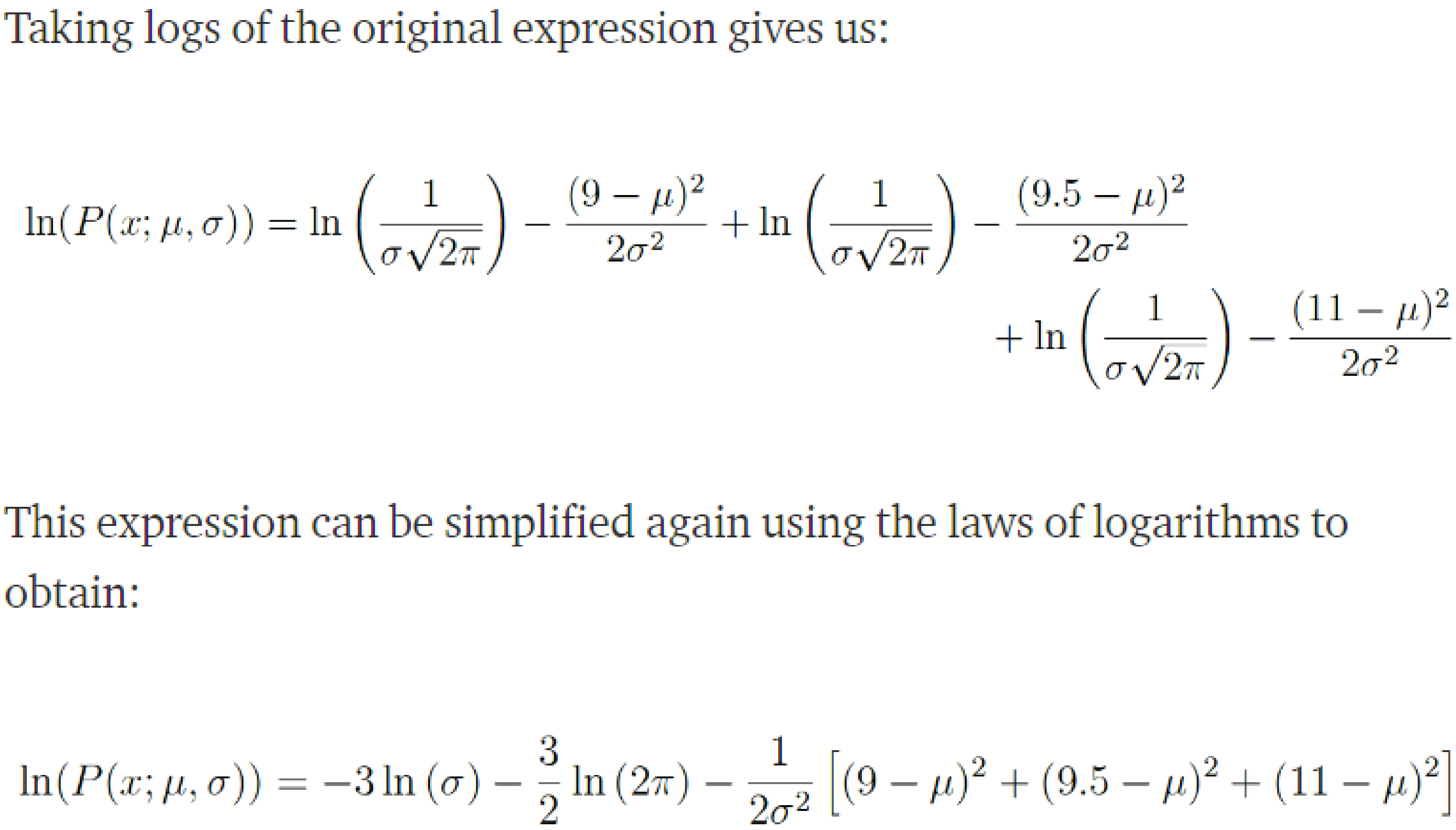
- This is absolutely fine because the natural logarithm is a monotonically increasing function.



(a) $f(x) = x$

(b) $f(x) = \ln(x)$

Log Likelihood



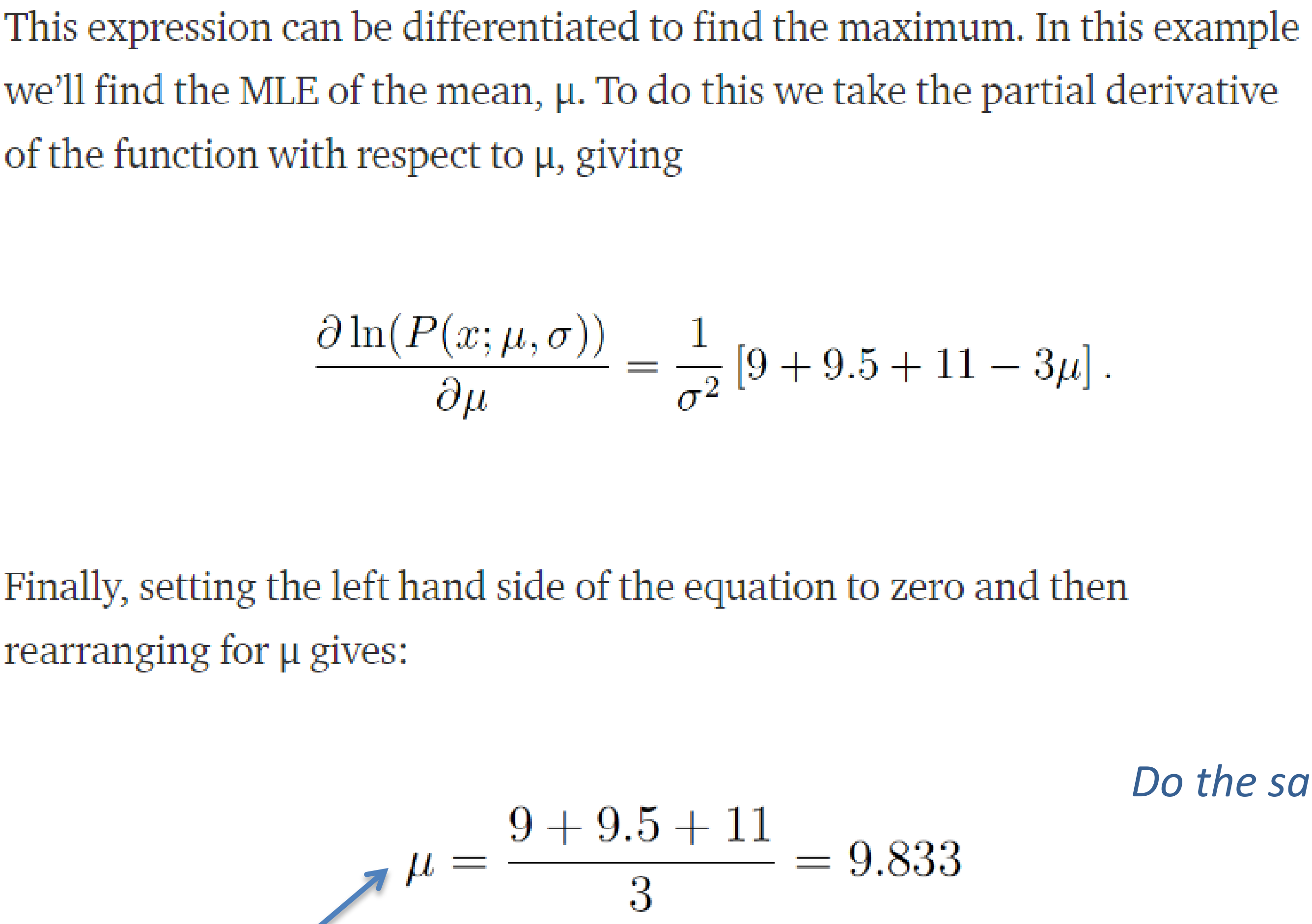
Taking logs of the original expression gives us:

$$\ln(P(x;\mu,\sigma)) = \ln\left(\frac{1}{\sigma\sqrt{2\pi}}\right) - \frac{(9-\mu)^2}{2\sigma^2} + \ln\left(\frac{1}{\sigma\sqrt{2\pi}}\right) - \frac{(9.5-\mu)^2}{2\sigma^2} \\ + \ln\left(\frac{1}{\sigma\sqrt{2\pi}}\right) - \frac{(11-\mu)^2}{2\sigma^2}$$

This expression can be simplified again using the laws of logarithms to obtain:

$$\ln(P(x;\mu,\sigma)) = -3\ln(\sigma) - \frac{3}{2}\ln(2\pi) - \frac{1}{2\sigma^2}\left[(9-\mu)^2 + (9.5-\mu)^2 + (11-\mu)^2\right]$$

Computing μ_{ML}



This expression can be differentiated to find the maximum. In this example we'll find the MLE of the mean, μ. To do this we take the partial derivative of the function with respect to μ, giving

$$\frac{\partial \ln(P(x;\mu,\sigma))}{\partial \mu} = \frac{1}{\sigma^2}\left[9 + 9.5 + 11 - 3\mu\right].$$

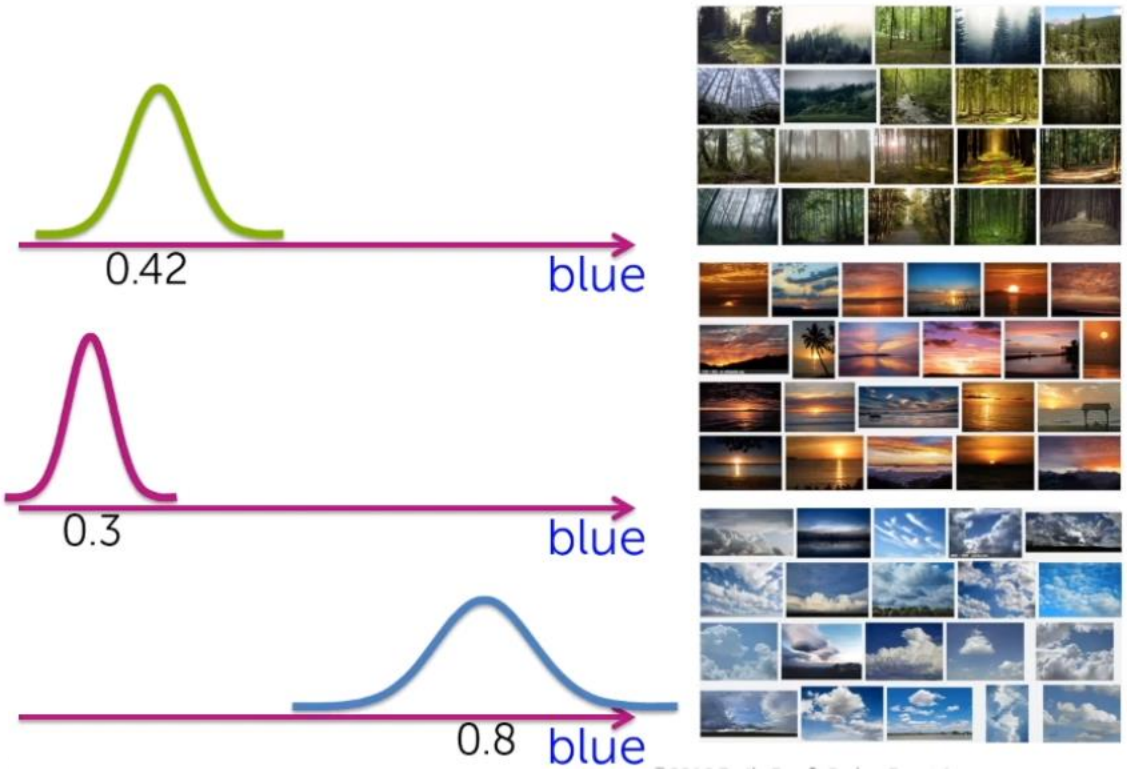
Finally, setting the left hand side of the equation to zero and then rearranging for μ gives:

$$\mu = \frac{9 + 9.5 + 11}{3} = 9.833$$

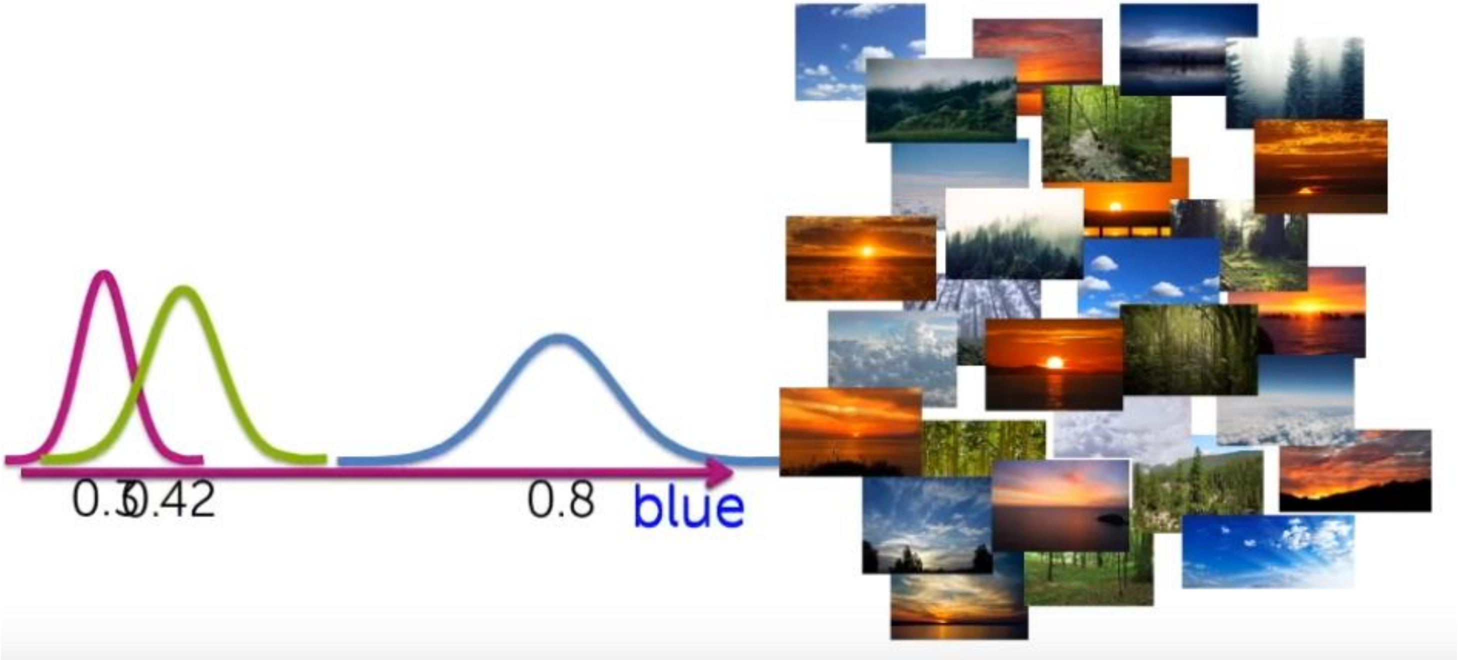
μ_{ML}

Do the same for σ

Mixtures of Gaussians

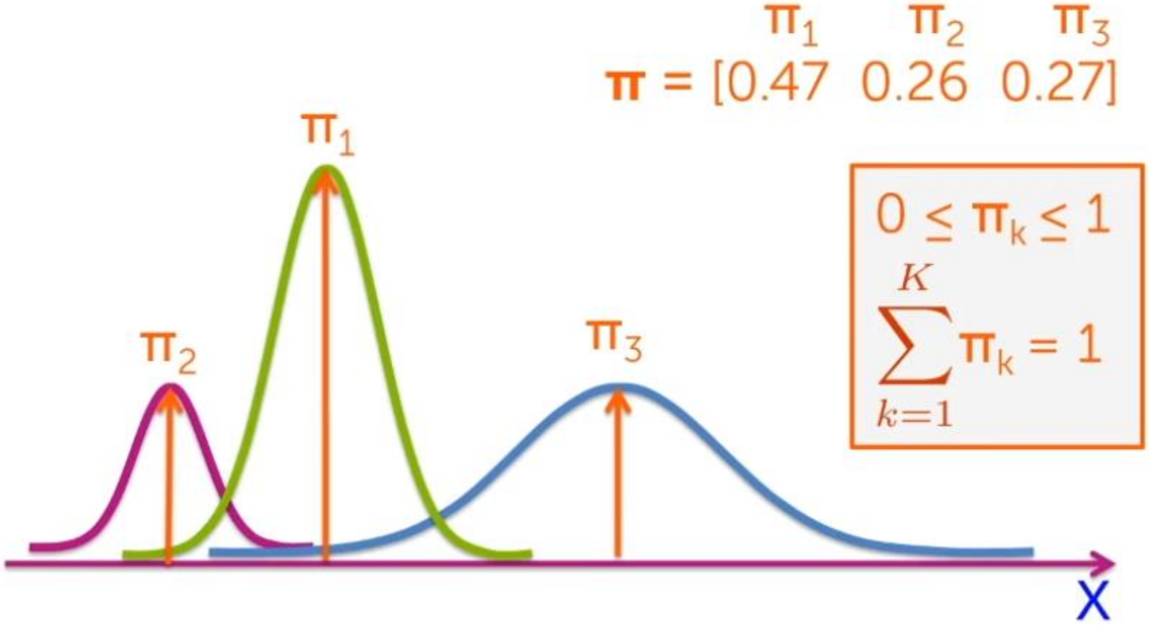


Mixtures of Gaussians



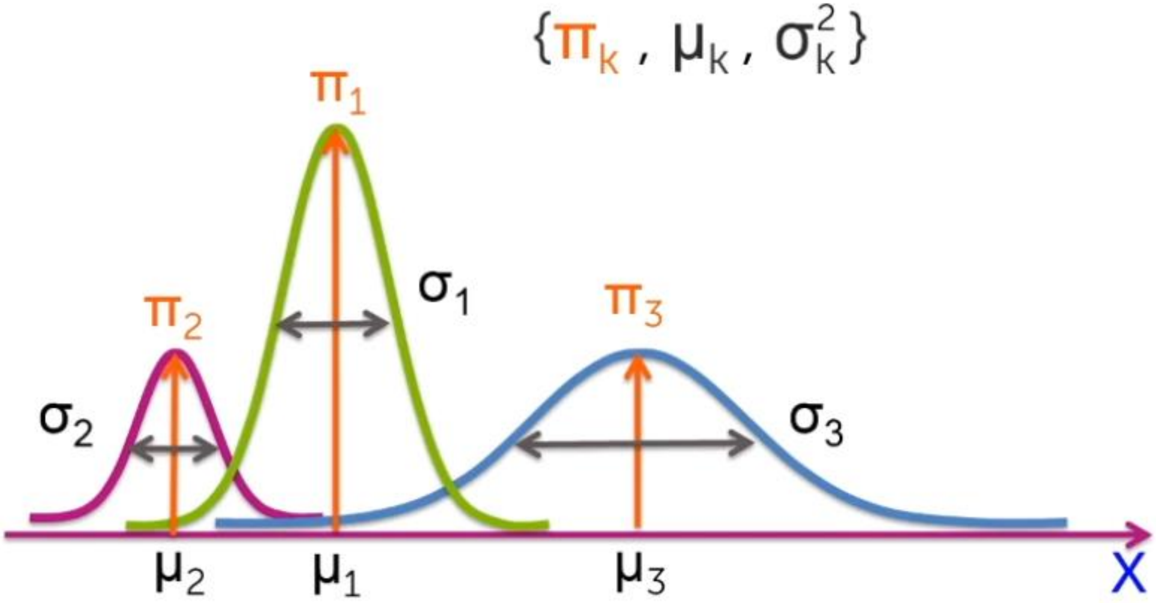
Mixtures of Gaussians

- Associate a weight π_k with each Gaussian Component: “The mixing coefficients”

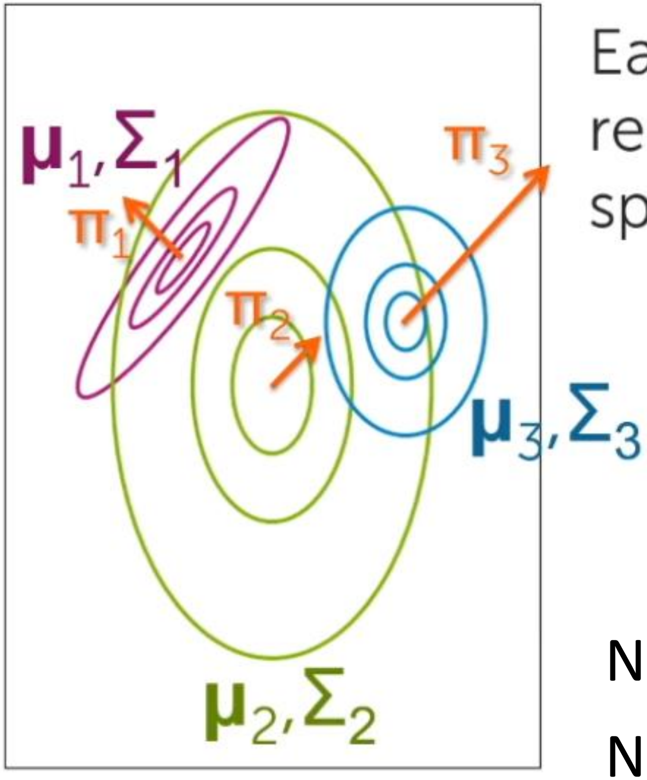


Mixtures of Gaussians

- Location and spread for the distributions comprising the Gaussians



Higher Dimensions



Each mixture component represents a unique cluster specified by:

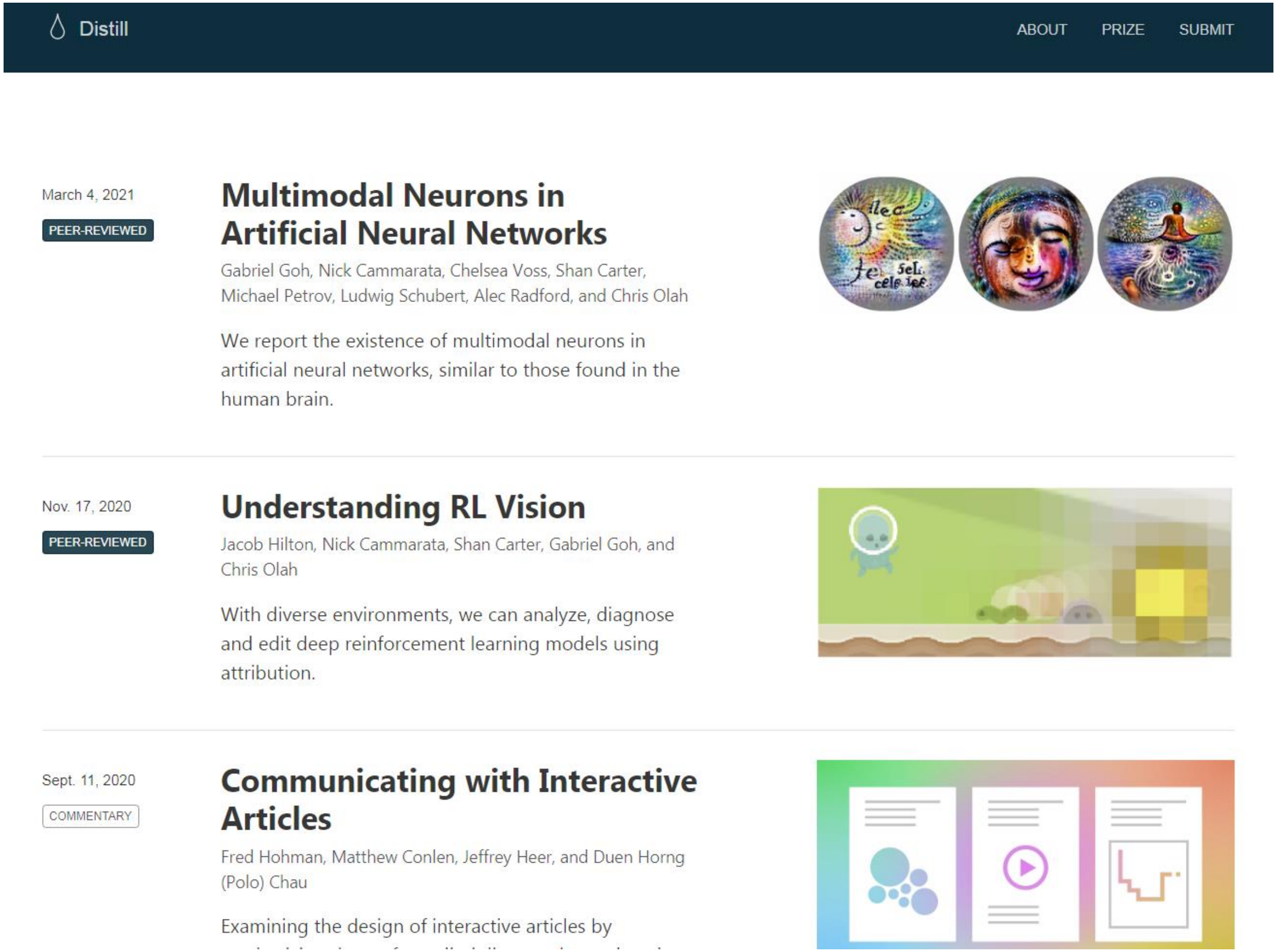
$$\{\pi_k , \mu_k , \Sigma_k\}$$

Naturally generated clusters!

Naturally a generative model!

vs. discriminative models

https://distill.pub/



Distill

ABOUT PRIZE SUBMIT

March 4, 2021

PEER-REVIEWED

Multimodal Neurons in Artificial Neural Networks

Gabriel Goh, Nick Cammarata, Chelsea Voss, Shan Carter, Michael Petrov, Ludwig Schubert, Alec Radford, and Chris Olah

We report the existence of multimodal neurons in artificial neural networks, similar to those found in the human brain.

Nov. 17, 2020

PEER-REVIEWED

Understanding RL Vision

Jacob Hilton, Nick Cammarata, Shan Carter, Gabriel Goh, and Chris Olah

With diverse environments, we can analyze, diagnose and edit deep reinforcement learning models using attribution.

Sept. 11, 2020

COMMENTARY

Communicating with Interactive Articles

Fred Hohman, Matthew Conlen, Jeffrey Heer, and Duen Horng (Polo) Chau

Examining the design of interactive articles by