Introduction to Neural Networks

By Vipul Goyal

Why Another Technique?

Linear and Logistics regression are “one-shot”

Give the input, the output comes out “right away”

- In linear regression: output is a simple linear function of the input
- In logistic regression: you apply the logistic function

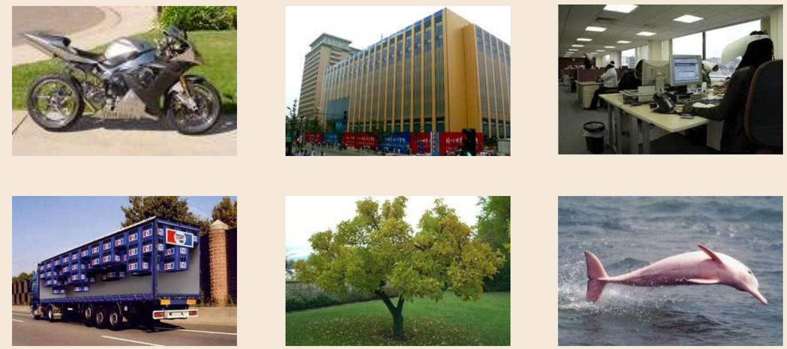
Why Another Technique?

- But many computations are more complex! Might involve millions of steps to go from input to output.
- Thinking about a program with millions of lines of code (iPhone apps, Zoom software…)
- Think about self driving cars. Car decides whether to apply brakes or not!

Computer Vision: Classification

Training Examples

Cars

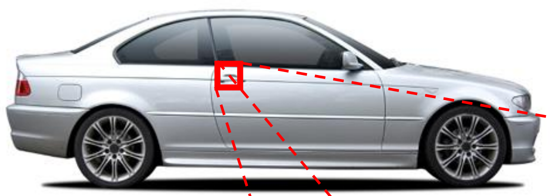


Not a car

Input:

Computer Vision: Classification

We see this:



194	210	201	212	199	213	215	195	178	158	182	209
180	189	190	221	209	205	191	167	147	115	129	163
114	126	140	188	176	165	152	140	170	106	78	88
87	103	115	154	143	142	149	153	173	101	57	57
102	112	106	131	122	138	152	147	128	84	58	66
94	95	79	104	105	124	129	113	107	87	69	67
68	71	69	98	89	92	98	95	89	88	76	67
41	56	68	99	63	45	60	82	58	76	75	65
20	43	69	75	56	41	51	73	55	70	63	44
50	50	57	69	75	75	73	74	53	68	59	37
72	59	53	66	84	92	84	74	57	72	63	42
67	61	58	65	75	78	76	73	59	75	69	50

But the program sees this

Seems unlikely that within a “single shot”, you get the answer from these numbers!

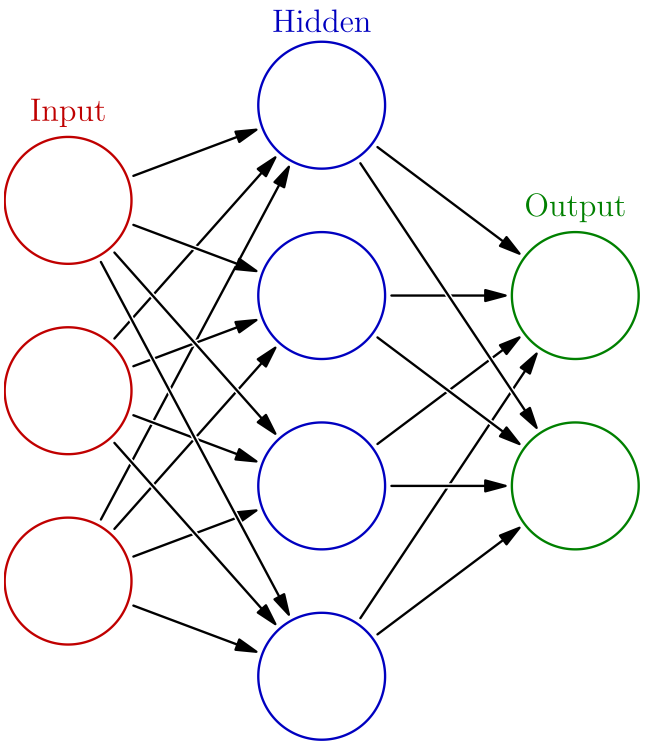
Neural Networks

Short story: several instances of “one-shot” learning algorithms connected with each other.

Example: many logistic regression “gates” arranged like a circuit or a directed acyclic graph.

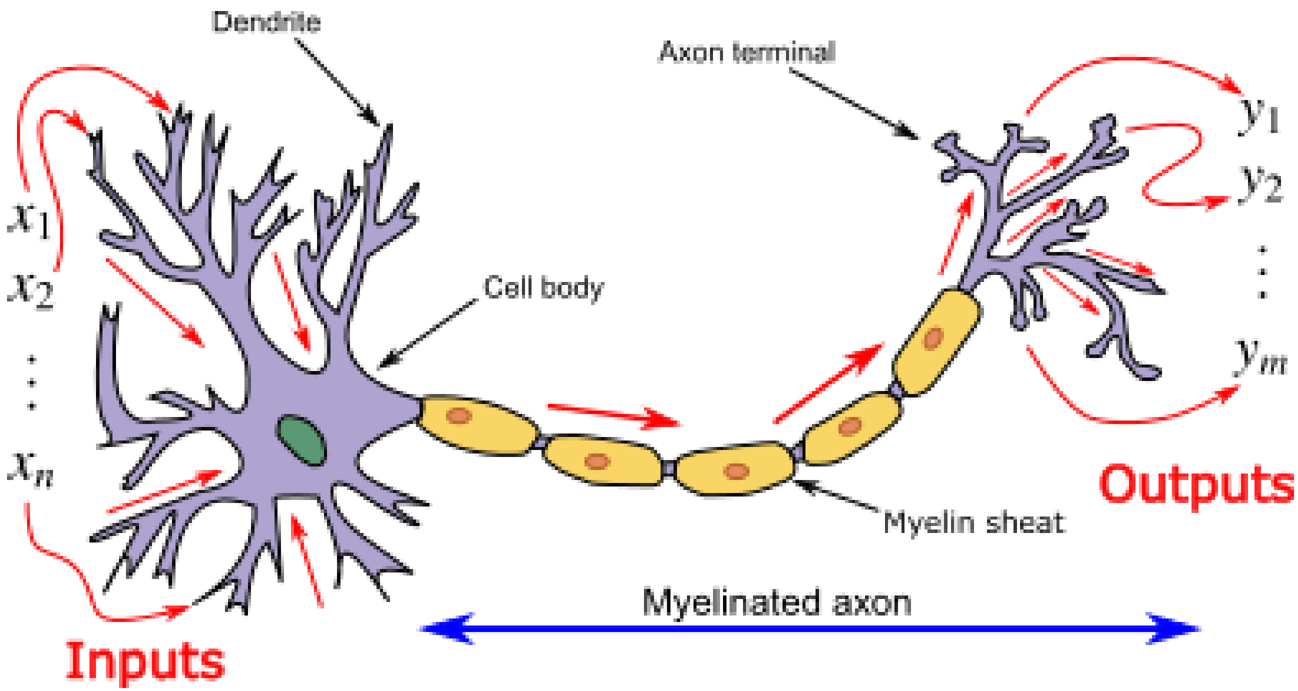
Also called “artificial neural networks”. “Natural” neural networks are inside our brain.

Long story?



“Natural” Neural Networks

We are all born with neural networks inside our brain

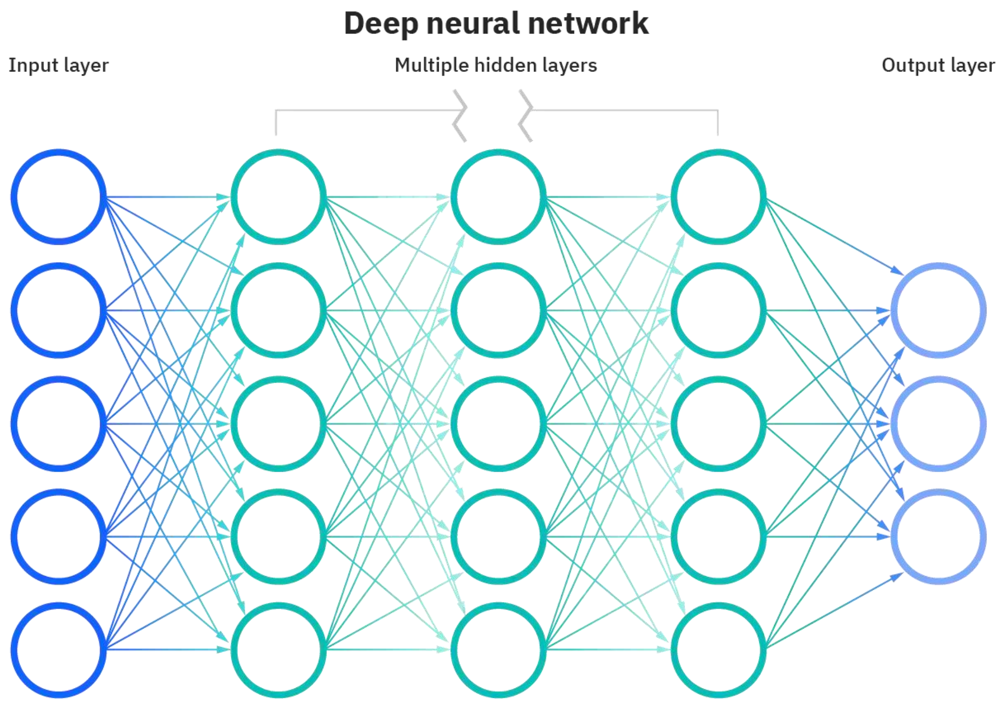


- Dendrites can be seen as input wires. Axon is the output wire. Based on the inputs, a neuron may “fire” or “stay quiet”
- Output of one neuron goes as input to another.

(Artificial) Neural Networks

- Composed of (artificial) neurons. Each artificial neuron has inputs and produces a single output which can be sent to multiple other neurons.

- The inputs can be the feature values of a sample of external data, such as images or documents, or they can be the outputs of other neurons.



Neural Networks

- The outputs of the final output neurons of the neural net accomplish the task, such as recognizing an object in an image.

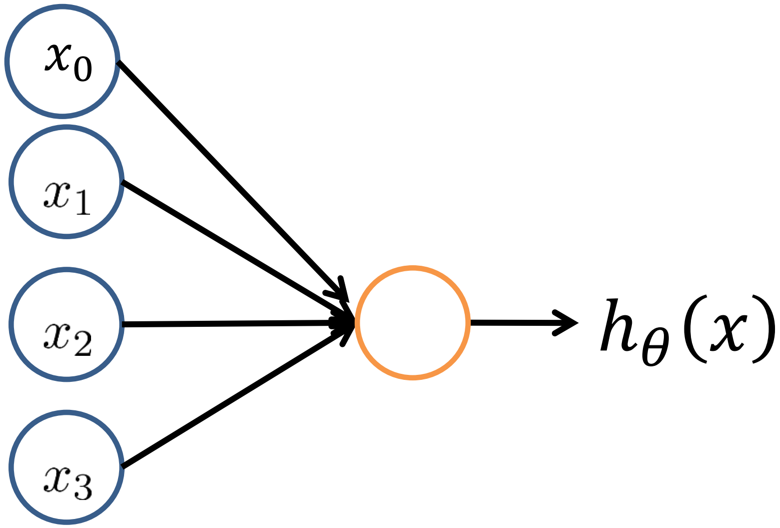
- To find the output of the neuron, first we take the weighted sum of all the inputs, weighted by the weights of the connections from the inputs to the neuron.

- We add a bias term to this sum. Similar to constant θ_0 in linear regression.

- This weighted sum is then passed through a (usually nonlinear) activation function to produce the output.

- The initial inputs are external data, such as images and documents. The ultimate outputs accomplish the task, such as recognizing an object in an image.

Using Logistic Unit as Neuron



$$x = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{bmatrix} \quad \theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \\ \theta_3 \end{bmatrix}$$

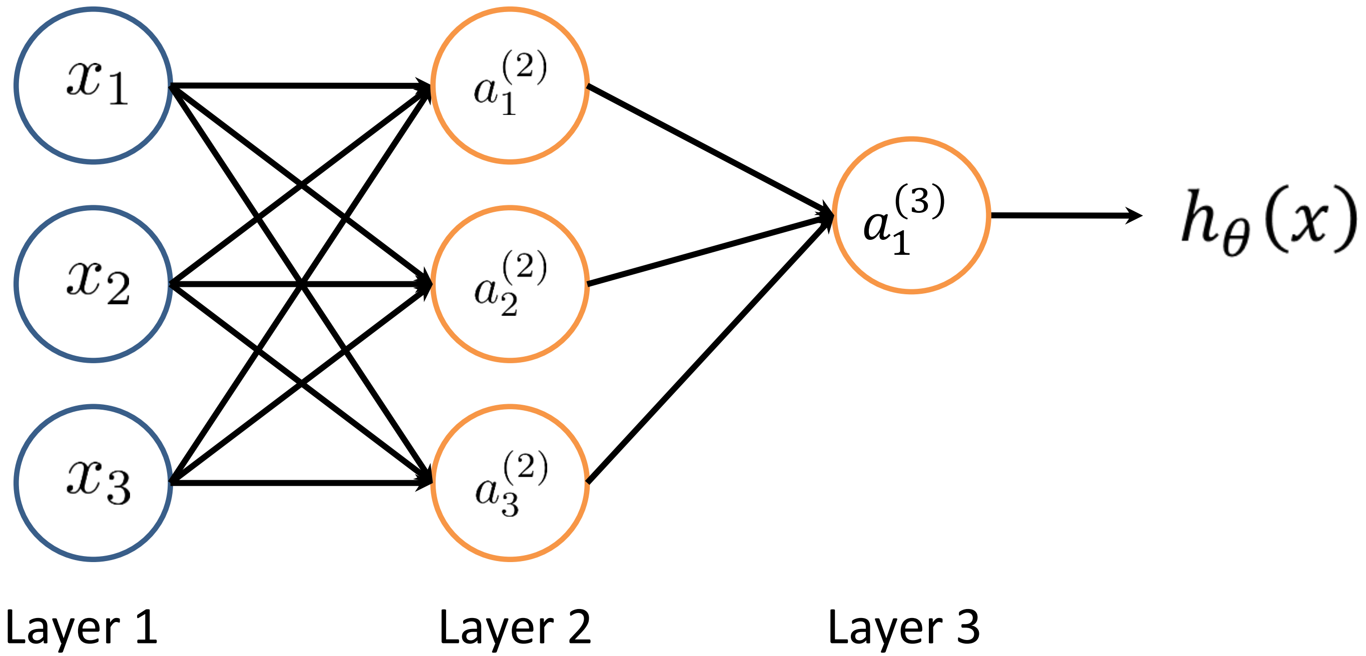
Sigmoid (logistic) activation function.

- x_1, x_2, x_3 are input (features). x_0=1 added as bias term.
- θ_0, θ_2 .... θ_3 are the weights.

Step1: compute weighted sum $z = \theta^T x$

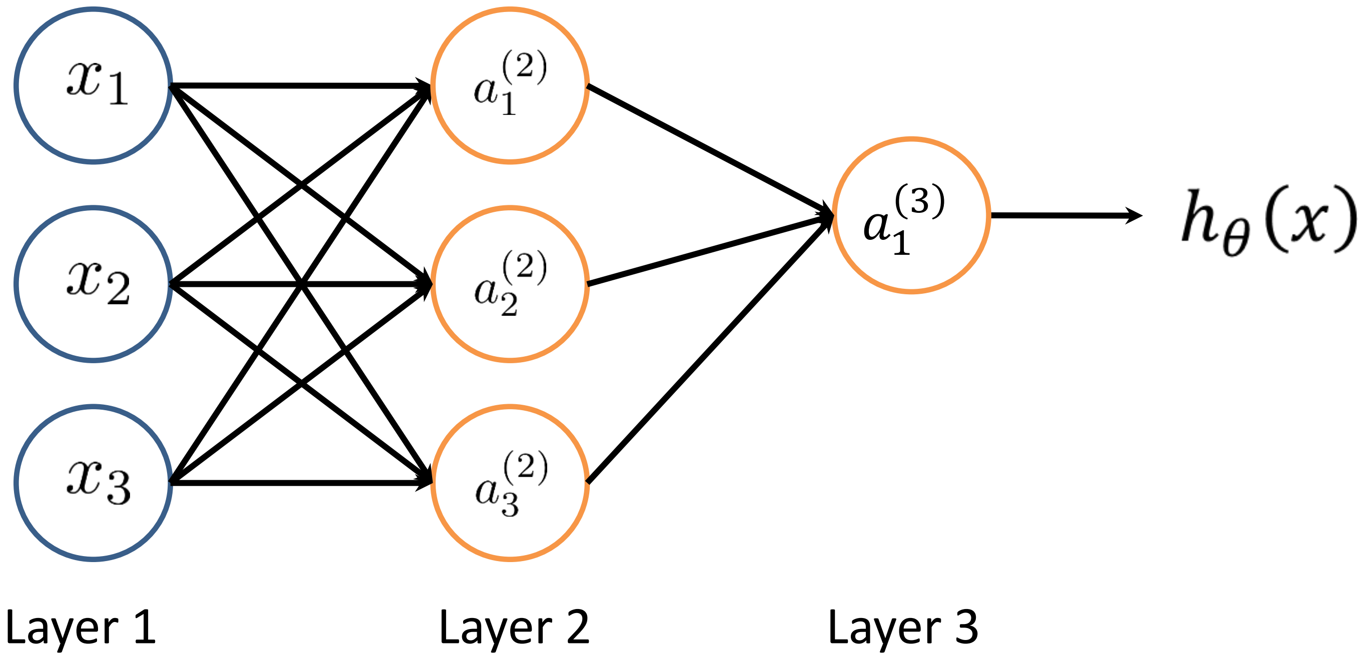
Step2: compute activation function $\mathrm{g}(z) = \frac{1}{1+e^{-z}}$ giving us the final output $h_\theta(x)$

Neural Networks



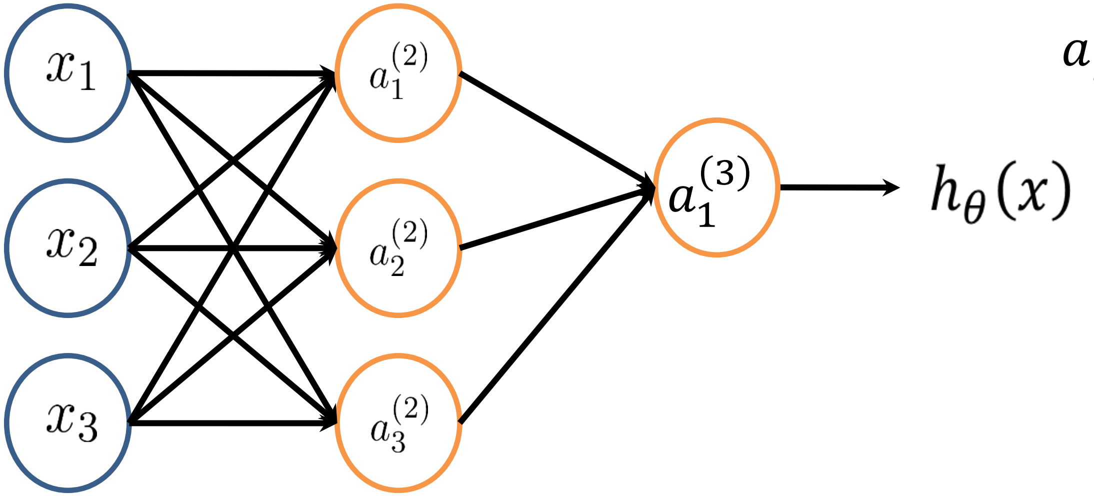
- Logistic units are in orange color. No computation in initial layer.
- Input layer, hidden layer(s), output layer
- Need to add x_0 as the bias term to all logistic units.

Neural Networks



- Each $a_{\mathrm{i}}^{(\mathrm{j})}$ unit has its own weights ($\theta_{\mathrm{i1}}^{(\mathrm{j})}, \theta_{\mathrm{i2}}^{(\mathrm{j})}$, …)
- $a_{\mathrm{i}}^{(2)}$ units take x_i's as input (plus bias term)
- $a_1^{(3)}$ unit takes $a_{\mathrm{i}}^{(2)}$'s as input (output of the previous layer)

Running the Neural Network



$a_i^{(j)}$ = "activation" of unit i in layer j

$\theta^{(j)}$ = matrix of weights controlling function mapping from layer j to layer $j+1$

$$a_1^{(2)} = g(\theta_{10}^{(1)} x_0 + \theta_{11}^{(1)} x_1 + \theta_{12}^{(1)} x_2 + \theta_{13}^{(1)} x_3)$$

$$a_2^{(2)} = g(\theta_{20}^{(1)} x_0 + \theta_{21}^{(1)} x_1 + \theta_{22}^{(1)} x_2 + \theta_{23}^{(1)} x_3)$$

$$a_3^{(2)} = g(\theta_{30}^{(1)} x_0 + \theta_{31}^{(1)} x_1 + \theta_{32}^{(1)} x_2 + \theta_{33}^{(1)} x_3)$$

$$h_\theta(x) = a_1^{(3)} = g(\theta_{10}^{(2)} a_0^{(2)} + \theta_{11}^{(2)} a_1^{(2)} + \theta_{12}^{(2)} a_2^{(2)} + \theta_{13}^{(2)} a_3^{(2)})$$

Question: how do we compute θ's? Training the neural network.

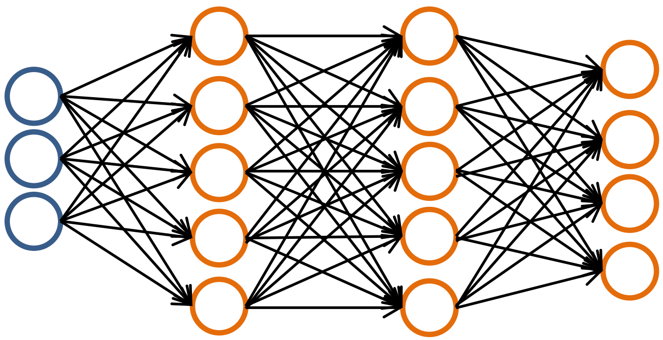
More General Neural Networks

Pedestrian

Car

Motorcycle

Truck

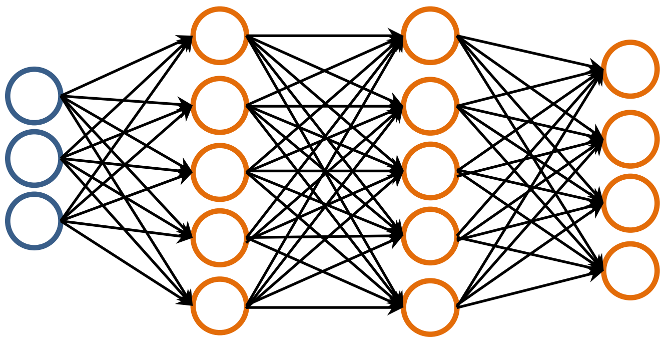


$h_\theta(x) \in R^4$

Want $h_\theta(x) \approx \begin{bmatrix}1\\0\\0\\0\end{bmatrix}$, $h_\theta(x) \approx \begin{bmatrix}0\\1\\0\\0\end{bmatrix}$, $h_\theta(x) \approx \begin{bmatrix}0\\0\\1\\0\end{bmatrix}$ etc.

Pedestrian, car, motorcycle

More General Neural Networks



$h_\theta(x) \in R^4$

Want $h_\theta(x) \approx \begin{bmatrix}1\\0\\0\\0\end{bmatrix}$, $h_\theta(x) \approx \begin{bmatrix}0\\1\\0\\0\end{bmatrix}$, $h_\theta(x) \approx \begin{bmatrix}0\\0\\1\\0\end{bmatrix}$, etc.

when pedestrian, when car, when motorcycle, etc.

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

$y^{(i)}$ is one of $\begin{bmatrix}1\\0\\0\\0\end{bmatrix}$, $\begin{bmatrix}0\\1\\0\\0\end{bmatrix}$, $\begin{bmatrix}0\\0\\1\\0\end{bmatrix}$, $\begin{bmatrix}0\\0\\0\\1\end{bmatrix}$

Pedestrian, car, motorcycle, truck

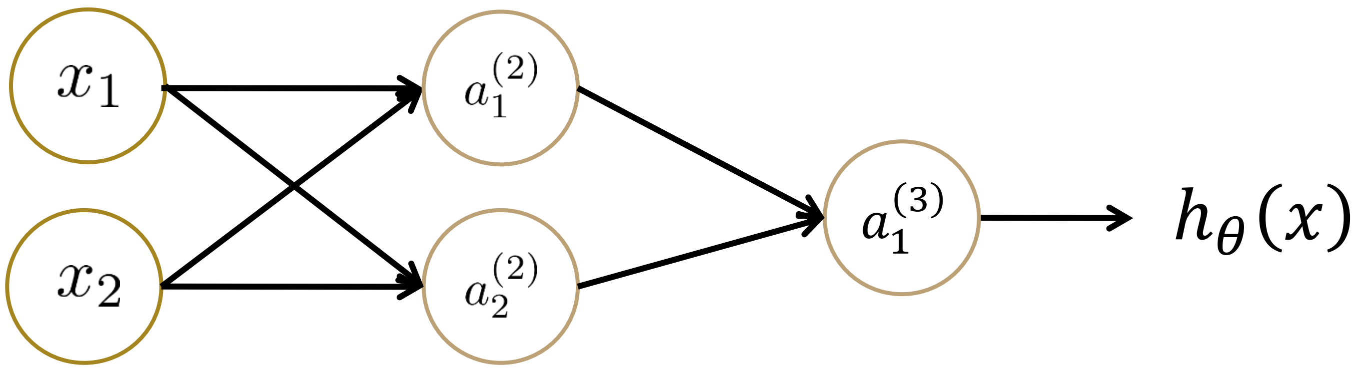
Why Non-Linear Activation?

- Can the neurons be linear regression units?

- Turns out that in this case, even arbitrarily deep neural networks are (roughly) equivalent to a *single* linear regression unit

- This is because composition of many linear functions is still a linear function. Why?

Why Non-Linear Activation?



$$a_1^{(2)} = \theta_{11}^{(1)} x_1 + \theta_{12}^{(1)} x_2$$
$$a_2^{(2)} = \theta_{21}^{(1)} x_1 + \theta_{22}^{(1)} x_2$$

$$\begin{aligned}
a_1^{(3)} &= \theta_{11}^{(2)} a_1^{(2)} + \theta_{12}^{(2)} a_2^{(2)} \\
&= \theta_{11}^{(2)} \left(\theta_{11}^{(1)} x_1 + \theta_{12}^{(1)} x_2\right) + \theta_{12}^{(2)} \left(\theta_{21}^{(1)} x_1 + \theta_{22}^{(1)} x_2\right) \\
&= \left(\theta_{11}^{(2)} \theta_{11}^{(1)} + \theta_{12}^{(2)} \theta_{21}^{(2)}\right) x_1 + \left(\theta_{11}^{(2)} \theta_{12}^{(1)} + \theta_{12}^{(2)} \theta_{22}^{(1)}\right) x_2 \\
&= \theta_1' x_1 + \theta_2' x_2
\end{aligned}$$

Training the Neural Network

A Million (Trillion?) Dollar Question

We have learnt how to evaluate a neural network

Question: how does one compute the weights $\theta_{ij}^{(k)}$?

- First step: define a cost function
- Second step: select $\theta_{ij}^{(k)}$ carefully to minimize the cost function

Cost Function for Neural Network*

Logistics regression:

$$J(\theta) = -\frac{1}{m}\left[\sum_{i=1}^{m} y^{(i)} log\, h_\theta\left(x^{(i)}\right) + \left(1 - y^{(i)}\right) log\left(1 - h_\theta\left(x^{(i)}\right)\right)\right]$$

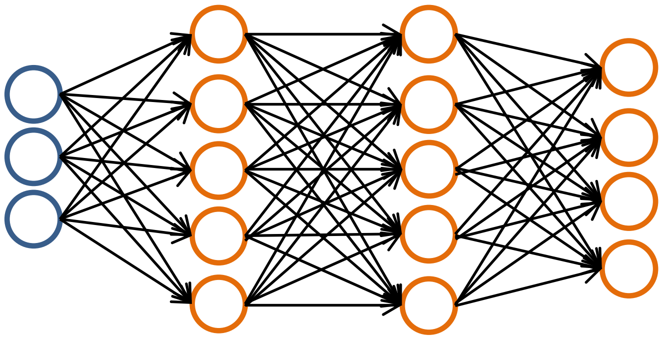
Neural Networks

$$h_\theta(x) \in R^K \qquad (h_\theta(x))_i = i^{\text{th}} \text{ output}$$

$$J(\theta) = -\frac{1}{m}\left[\sum_{i=1}^{m}\sum_{k=1}^{K} y_k^{(i)} \log\left(h_\theta\left(x^{(i)}\right)\right)_k + \left(1 - y_k^{(i)}\right)\log\left(1 - \left(h_\theta\left(x^{(i)}\right)\right)_k\right)\right]$$

*somewhat oversimplified, ignores bias terms

Intuition



$h_\theta(x) \in R^K$ $\qquad\qquad (h_\theta(x))_i = i^{\text{th}}$ output

$$J(\theta) = -\frac{1}{m}\left[\sum_{i=1}^{m}\sum_{k=1}^{K} y_k^{(i)} \log\left(h_\theta\left(x^{(i)}\right)\right)_k + \left(1 - y_k^{(i)}\right)\log(1 - \left(h_\theta\left(x^{(i)}\right)\right)_k\right]$$

Our cost function is just the sum of the cost function for each individual logistic unit in the output.

Optimizing the Cost

$$J(\theta) = -\frac{1}{m}\left[\sum_{i=1}^{m}\sum_{k=1}^{K} y_k^{(i)} \log\left(h_\theta\left(x^{(i)}\right)\right)_k + \left(1 - y_k^{(i)}\right)\log(1 - \left(h_\theta\left(x^{(i)}\right)\right)_k\right]$$

- To minimize the cost function, we need to make sure partial derivatives w.r.t. EACH $\theta_{ij}^{(k)}$ is close to 0
- Thus, we need to be able to compute the cost function and all the partial derivatives
- Afterwards: run gradient descent

Computing the Partial Derivatives

- Partial derivatives can be computed using what is known as the backpropagation algorithm
- Idea: compute an error term for each node in the network
- For output nodes: error term computed using the training examples. For hidden layer nodes: computed by going backwards from output layer by utilizing some known as chain rule and dynamic programming
- Exact math: messy and involved

Gradient Descent for NN

Want $min_\theta \, J(\theta)$:

Repeat $\{$

$$\theta_{ij}^{(k)} := \theta_{ij}^{(k)} - \alpha \frac{\partial}{\partial \theta_{ij}^{(k)}} J(\theta)$$

$\}$

(simultaneously update all $\theta_{ij}^{(k)}$)

- Keep in mind: it's important for all $\theta_{ij}^{(k)}$ to be initialized at random for symmetry breaking
- If initialized to the same value: all nodes within a given layer become identical and may remain identical

Flavors of Gradient Descent

- Batch gradient descent: all available data is injected at once.

- Stochastic gradient descent (SGD): a single random sample is introduced on each iteration.

- (Stochastic) Mini-batch gradient descent: instead of feeding the network with single samples, N random items are introduced on each iteration.

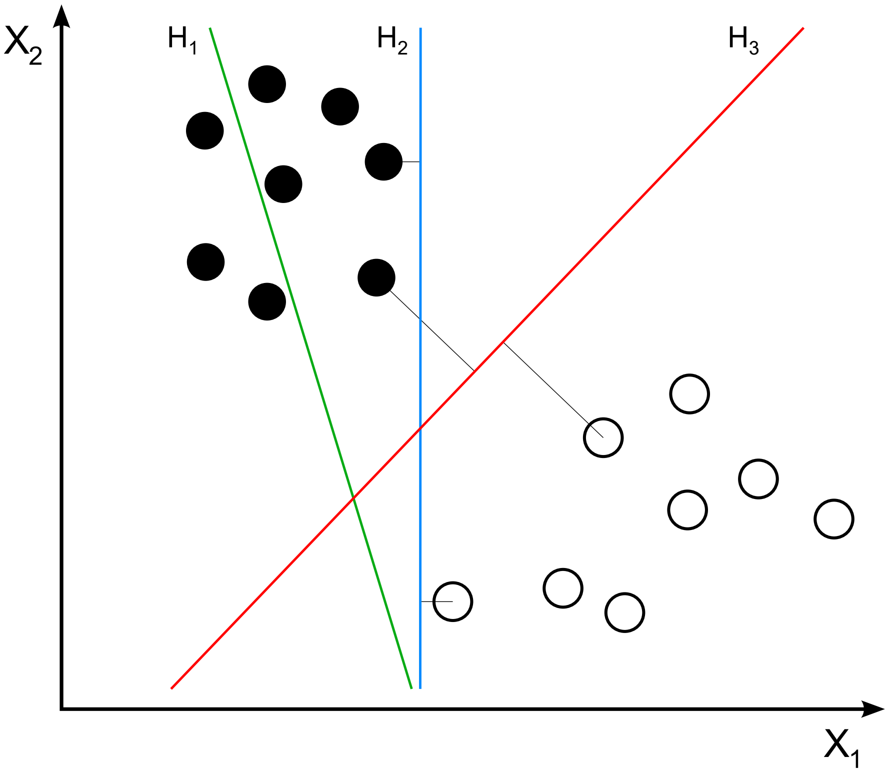
Support Vector Machines

Support Vector Machine (SVM)

Classification: black=0, white=1

Linear classifiers: H_1, H_2 and H_3

Which one is the best?



Are H_2 and H_3 equally good?
Probably Not.

Maximum-Margin Hyperplane

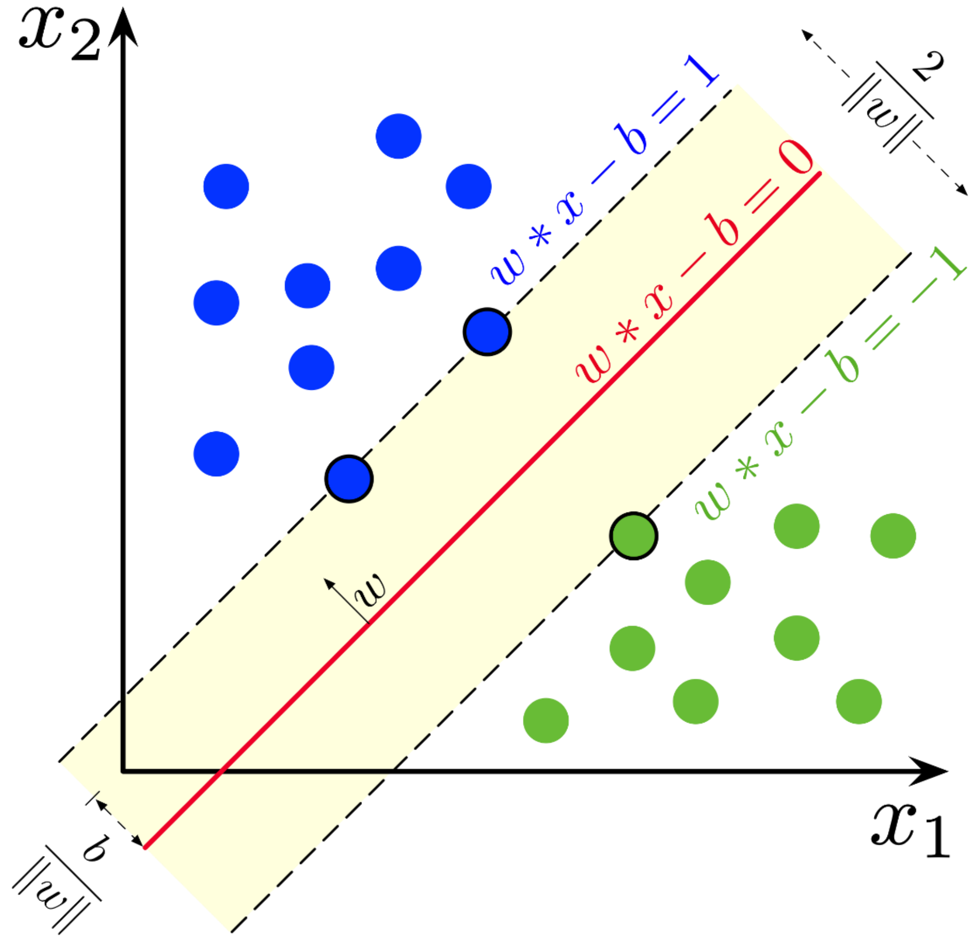
- Hyperplane: equivalent of line in higher dimensions (many features)

- A reasonable choice as the best hyperplane: the one that represents the largest separation, or margin, between the two classes. It is known as the maximum-margin hyperplane

- The linear classifier it defines is known as a maximum-margin classifier

- To compute such a hyperplane: define a cost function which increases if the hyperplane is “close” to any data point

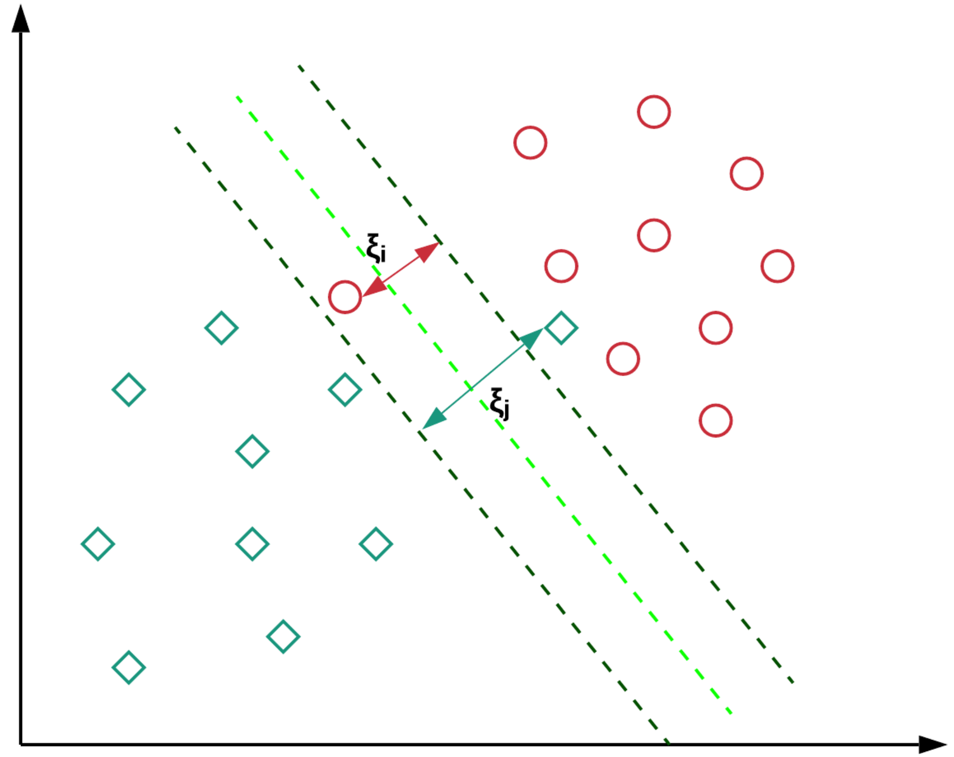
Maximum-Margin Hyperplane



Soft-Margin SVMs

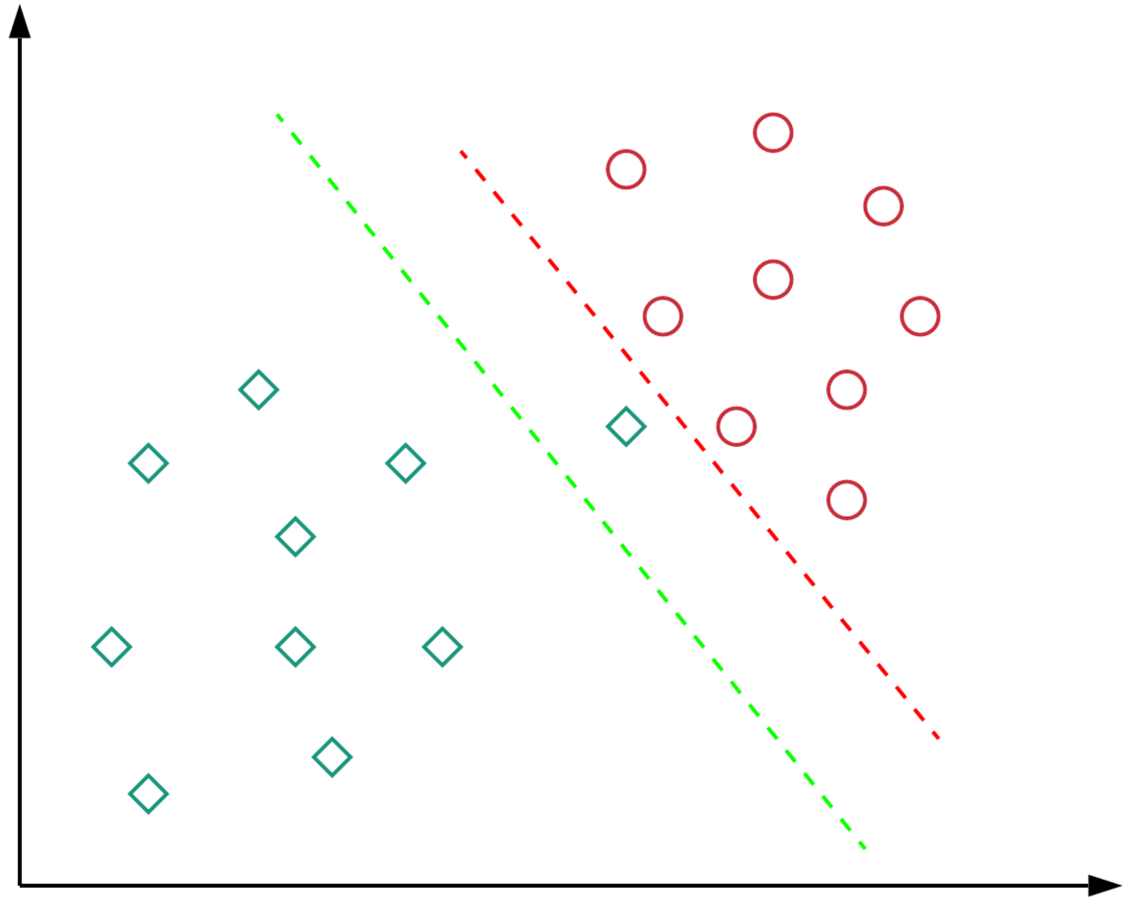
- Sometimes data may not be linearly separable

- Data points on the wrong side are known as outliers



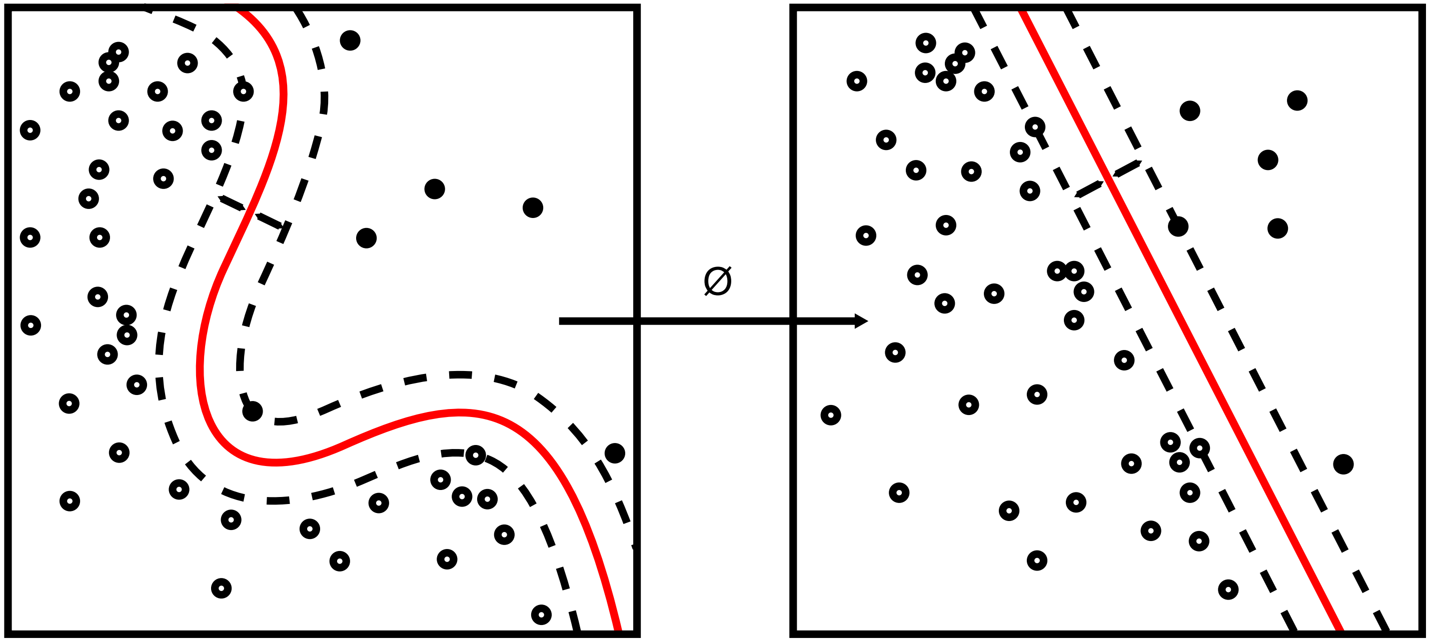
- To compute SVMs in such cases: we define a hinge loss function. Cost increases if the hyperplane: (a) doesn't separate the two classes, (b) is "close" to any data point

Outliers Maybe Acceptable



Green line maybe preferred over red since it has a higher margin (even though it results in 1 outlier) !!!

Nonlinear Classification



SVMs can be used for non-linear classification as well using so called “kernel functions”. These functions transform space which can change its shape.

Uses: higher dimensional linear algebra, inner products, vector spaces….

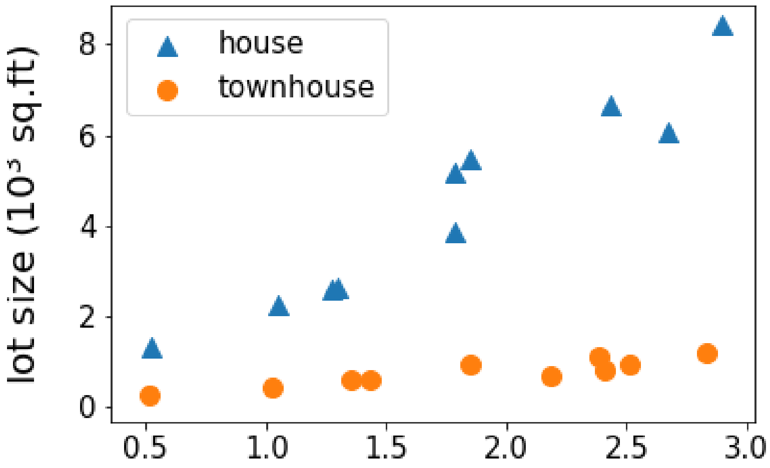
Unsupervised Learning (Clustering)

Unsupervised Learning

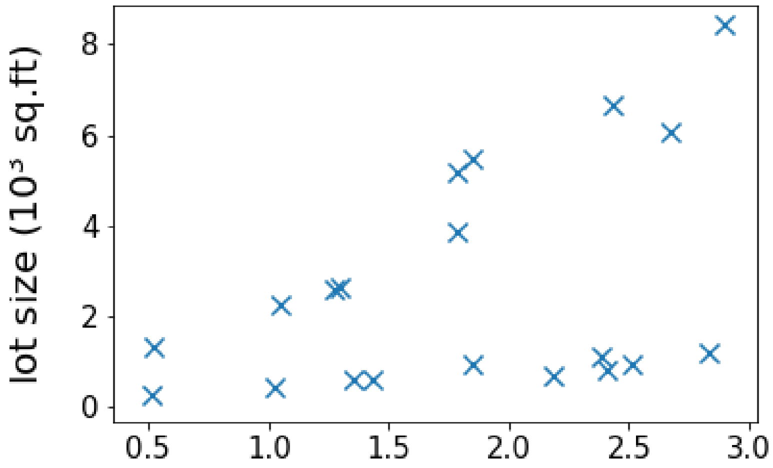
- Dataset contains no labels: $x^{(1)}, \dots x^{(m)}$

- Goal (vaguely-posed): to find interesting structures in the data

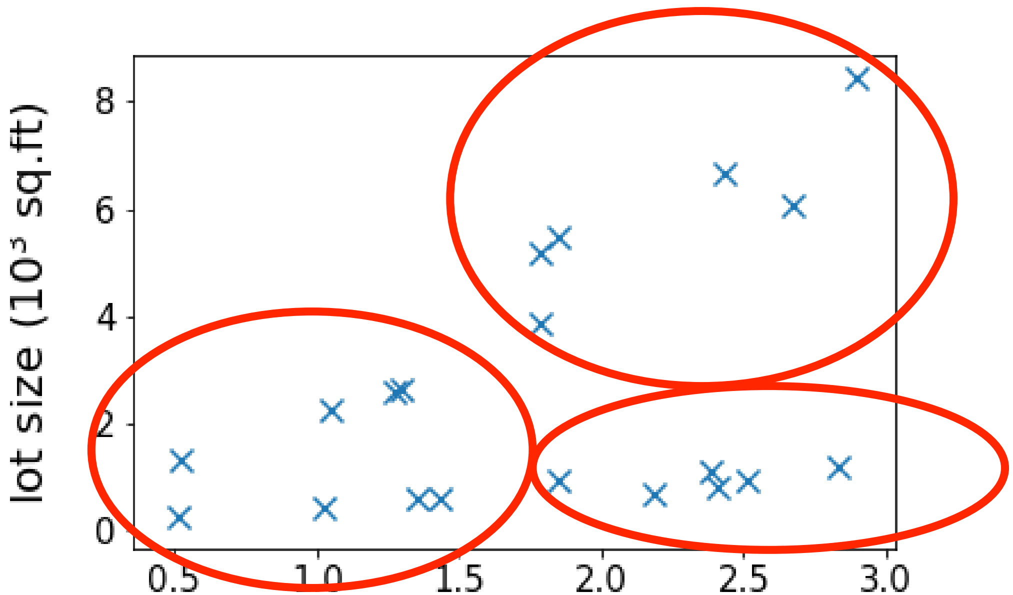
supervised



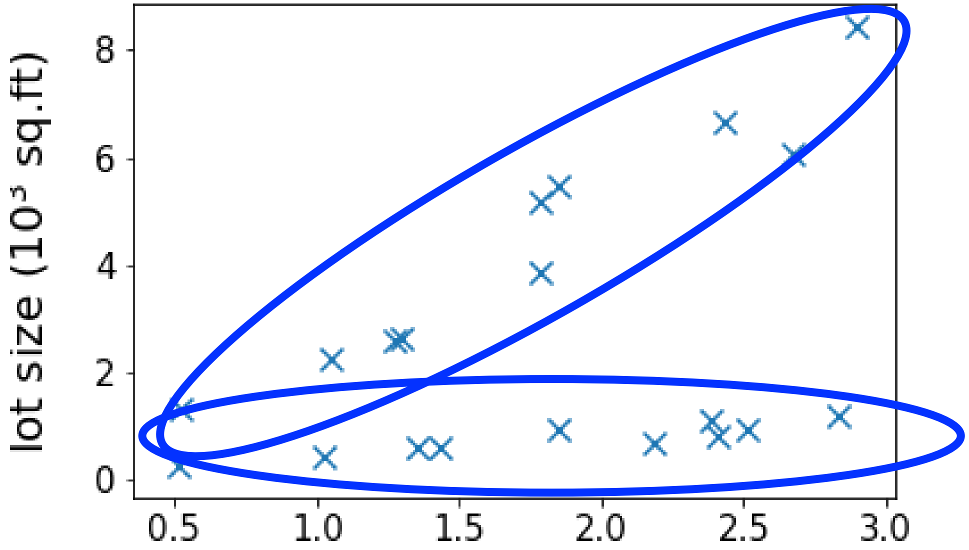
unsupervised



Clustering



Clustering



Google News

Headlines

More Headlines

COVID-19 news: See the latest coverage of the coronavirus (COVID-19)

Here's How the Senate Pared Back Biden's Stimulus Plan

The New York Times · 1 hour ago

- Third stimulus check is in COVID relief bill | USA TODAY
  USA TODAY · 1 hour ago
- Senate passes $1.9 trillion Covid relief bill, including $1,400 stimulus checks, with no Republican support
  NBC News · 6 hours ago
- Biden's historic victory for America -- no thanks to GOP
  CNN · 4 hours ago · Opinion
- Senate Democrats eke out 50-49 COVID-19 relief bill victory
  The Week · 5 hours ago

View Full Coverage

Amanda Gorman says she was "tailed" by security guard on her way home: "This is the reality of black girls"

CBS News · 7 hours ago

- Amanda Gorman, inaugural poet, 'tailed' by security guard on her walk home
  CNN · 8 hours ago

View Full Coverage

Jacksonville

Rain
55° F

Today	Sun	Mon	Tue	Wed
57°F	61°F	62°F	67°F	72°F
47°F	45°F	47°F	53°F	56°F

C | F | K

More on weather.com

Fact check

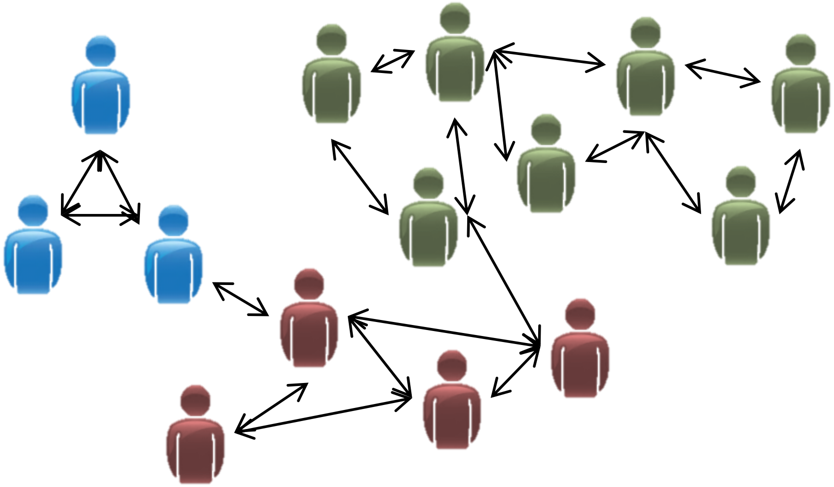
Did Kyrsten Sinema Bring Cake to the Senate and Vote Against Raising Minimum Wage?
Snopes.com

Fact Check: Are COVID-Positive Migrants Allowed to Cross Southern Border Into US?
Newsweek

Fitzgerald overstates claim on pork in COVID-19 relief bill
PolitiFact

Other Examples

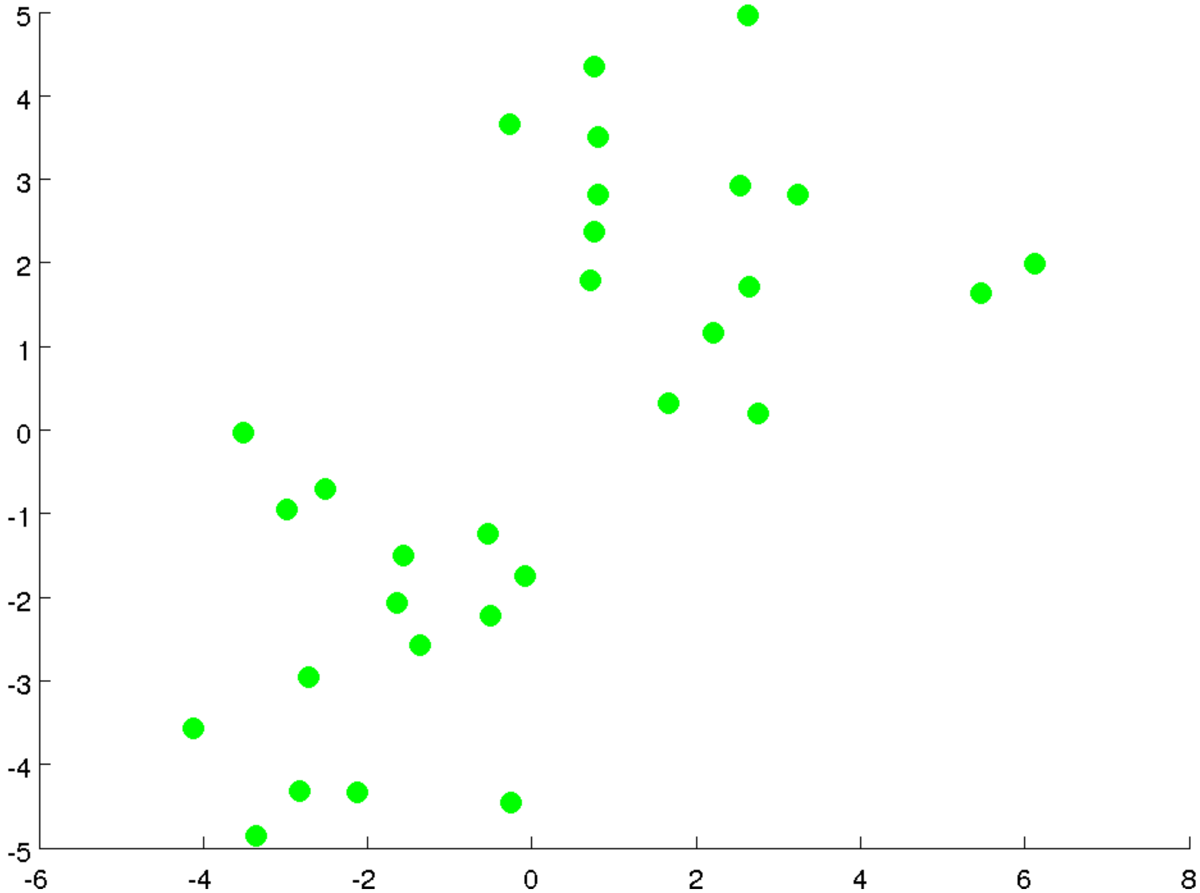
Clustering computer servers



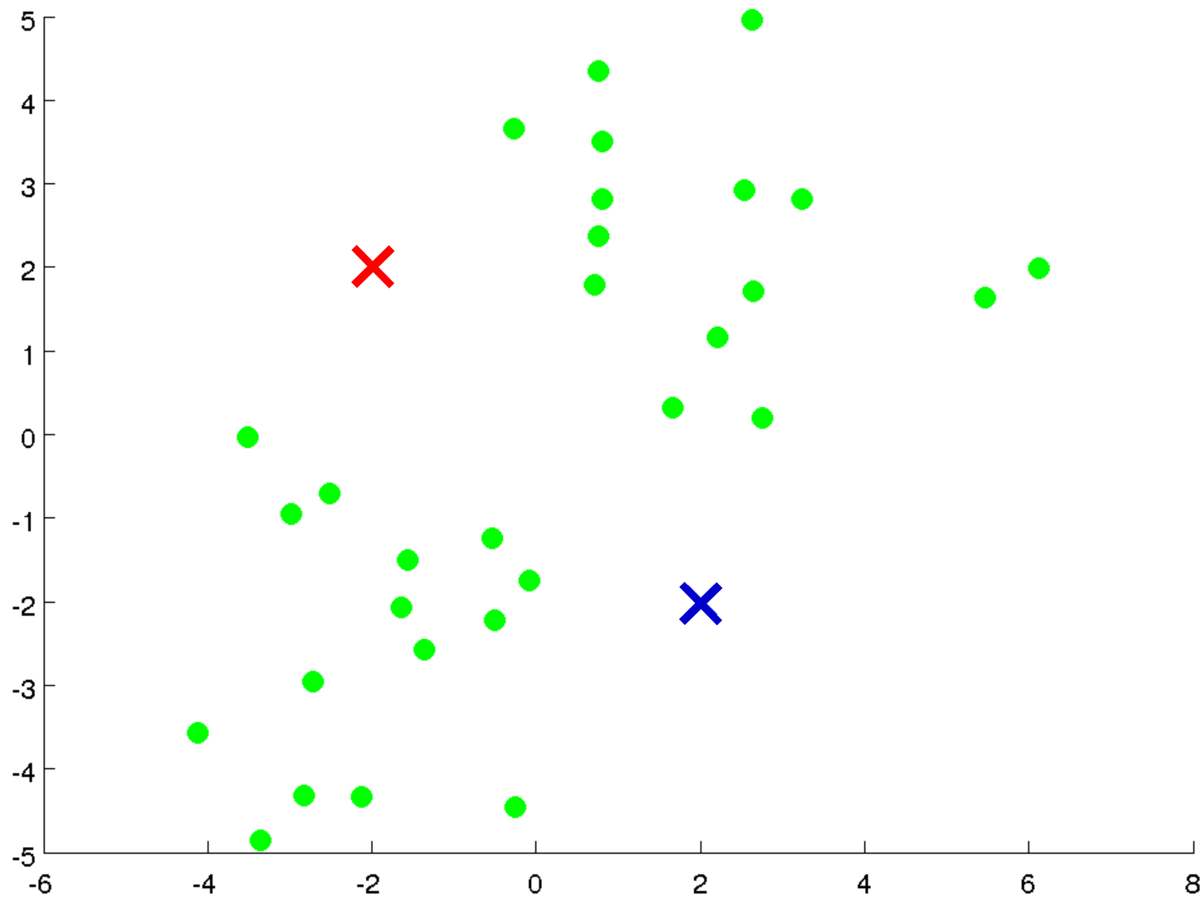
Clustering users in a
social network

K-means Clustering Algorithm

K-means Clustering

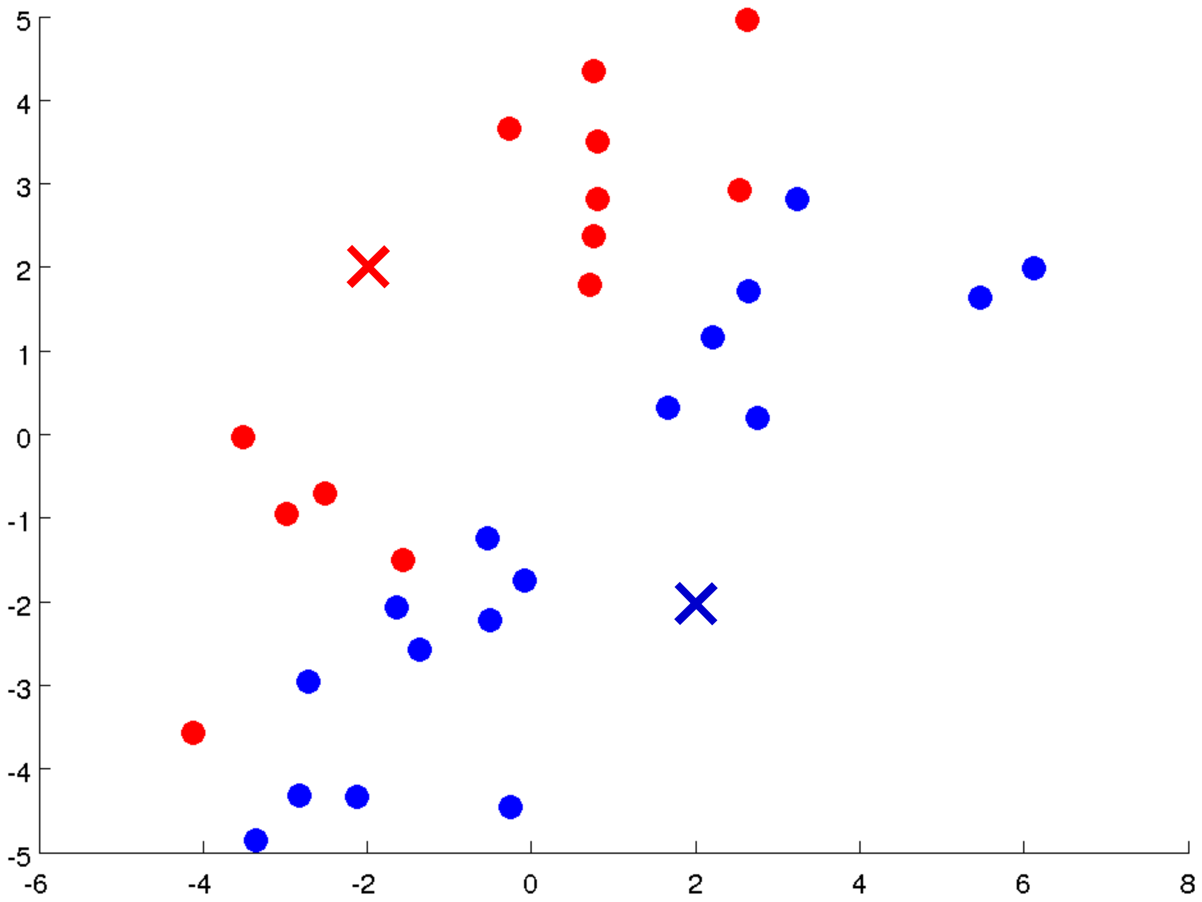


K-means Clustering



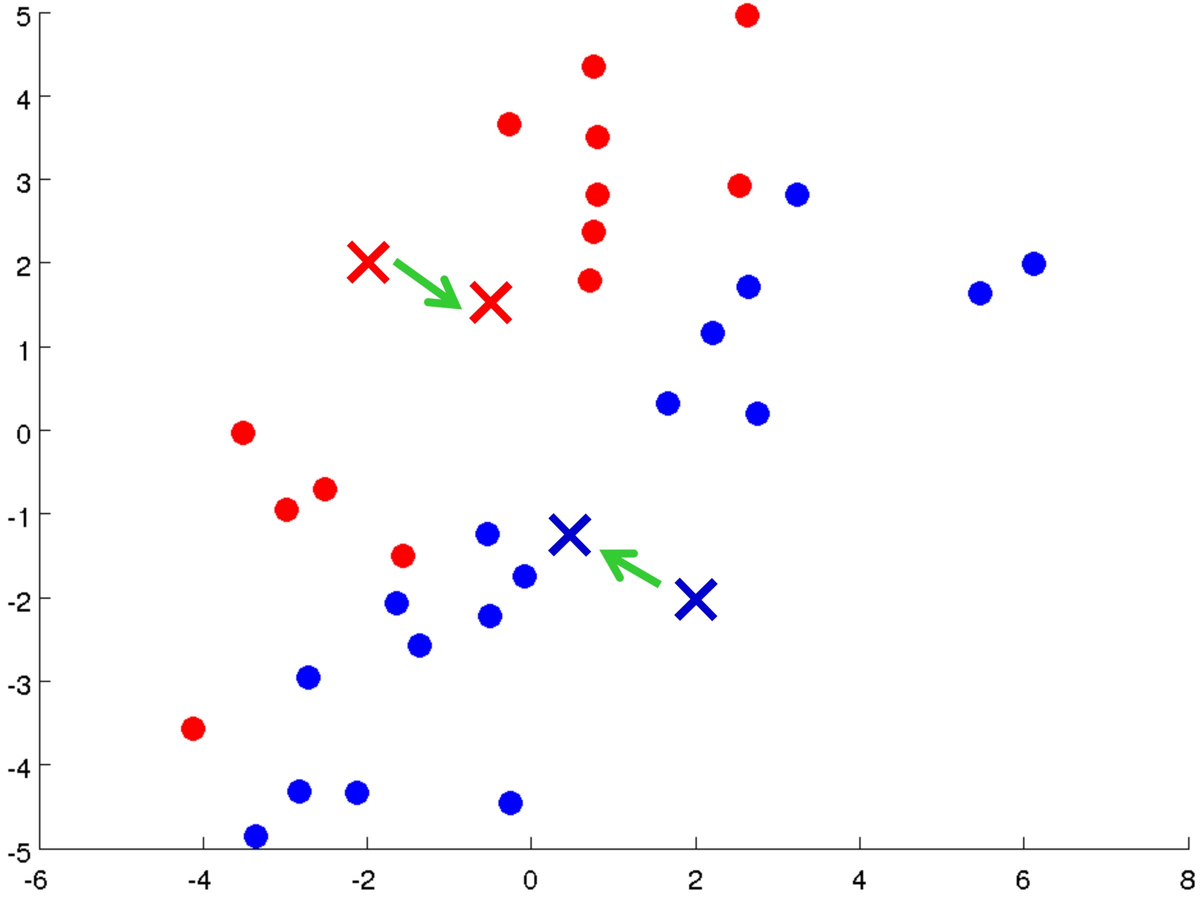
K= number of clusters. Start with centroid for each.

K-means Clustering



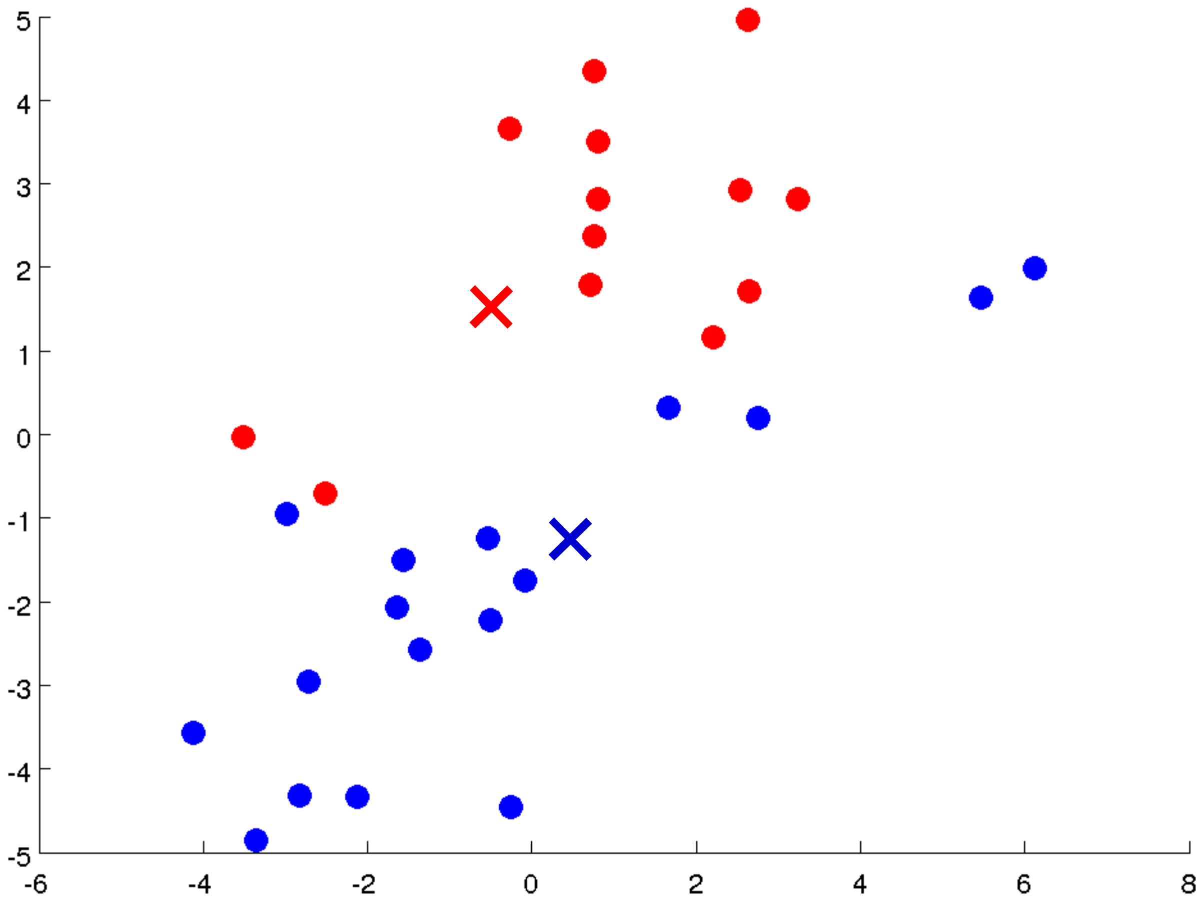
Match each point to the centroid which is “closer”

K-means Clustering



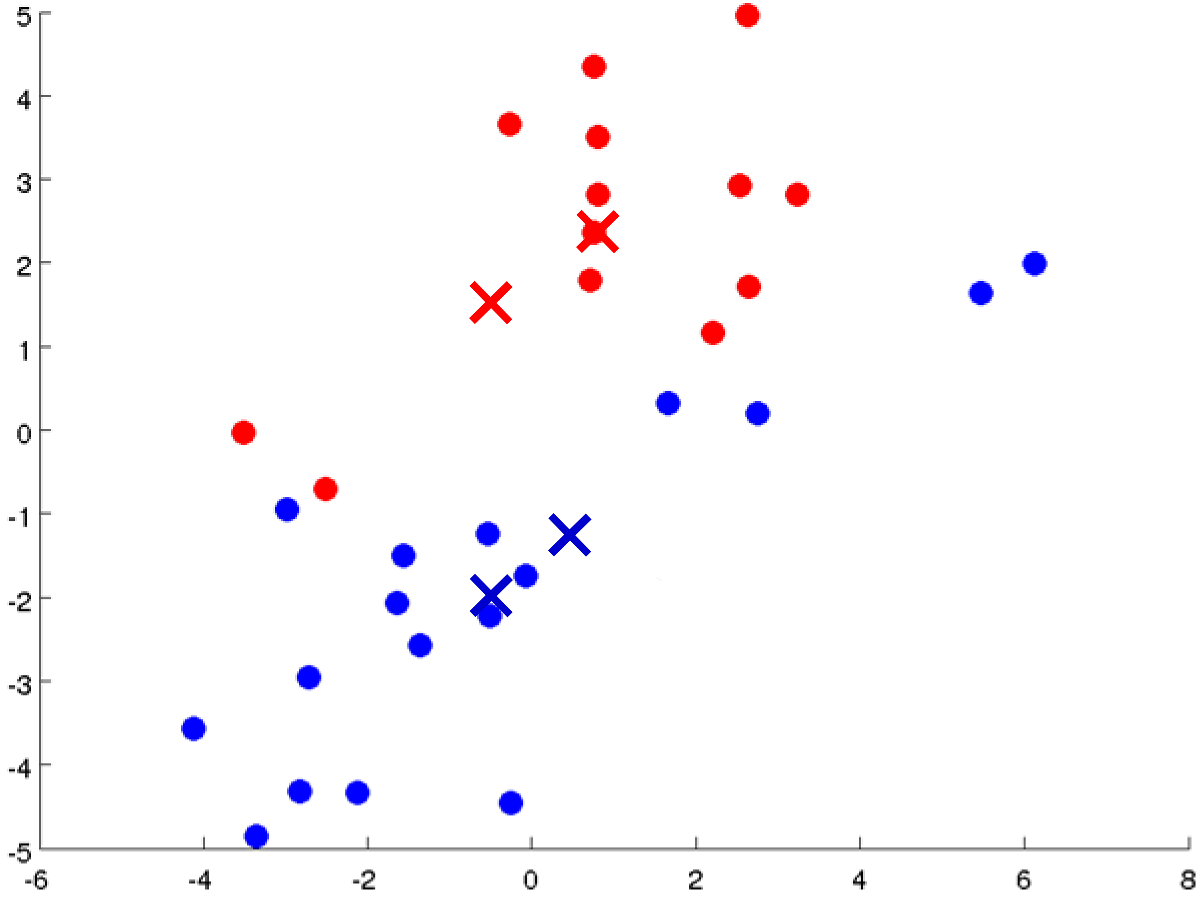
Compute new centroids as the “average” of each cluster

K-means Clustering

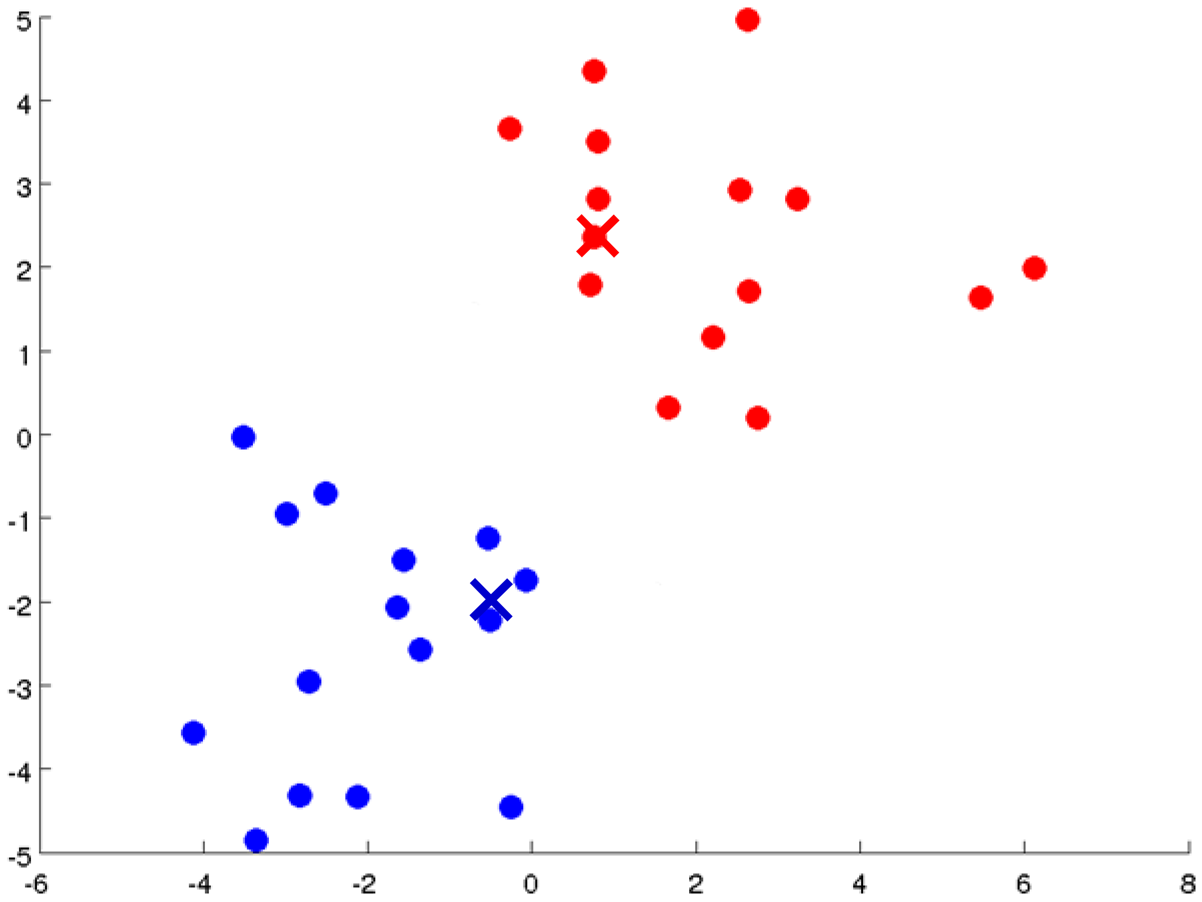


Restart with new centroids: Match each point to the centroid which is “closer”

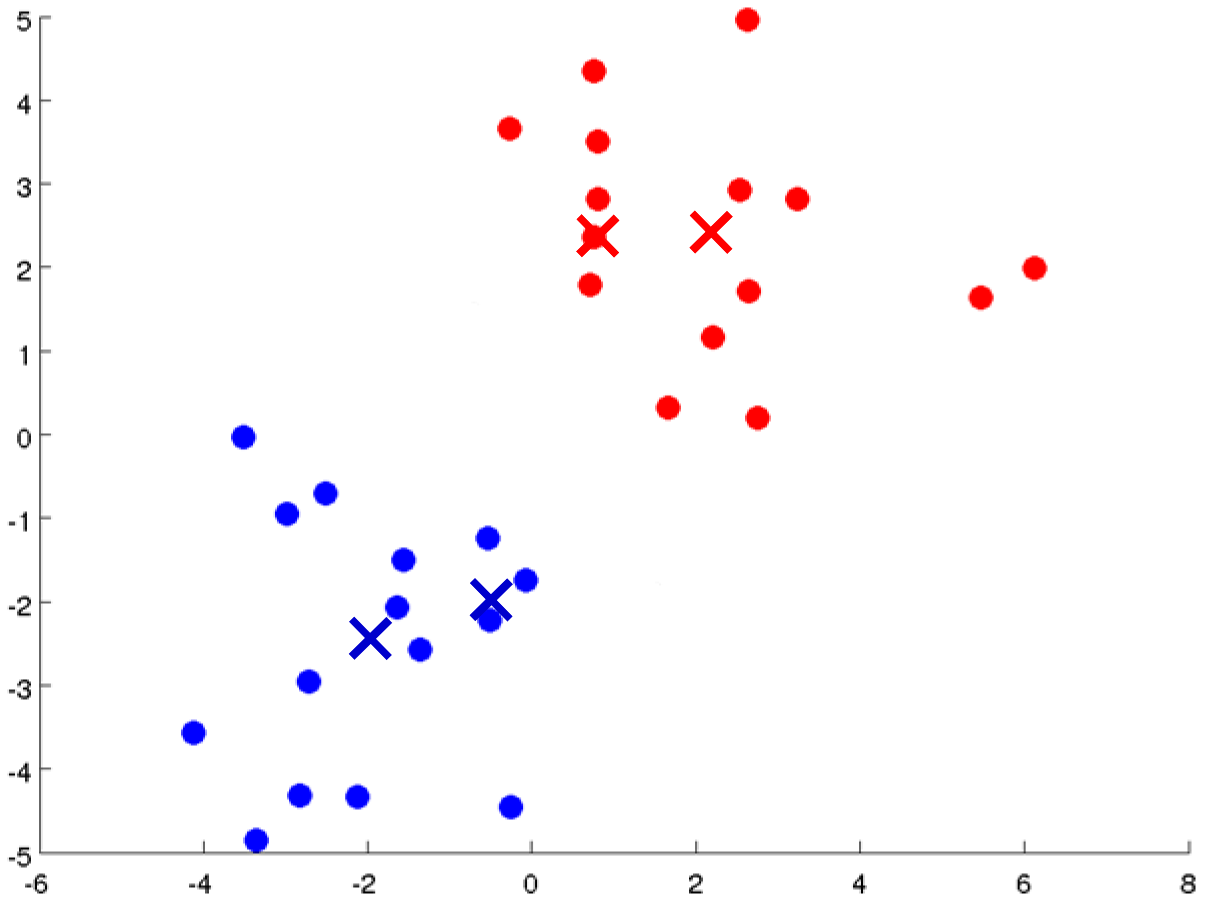
K-means Clustering



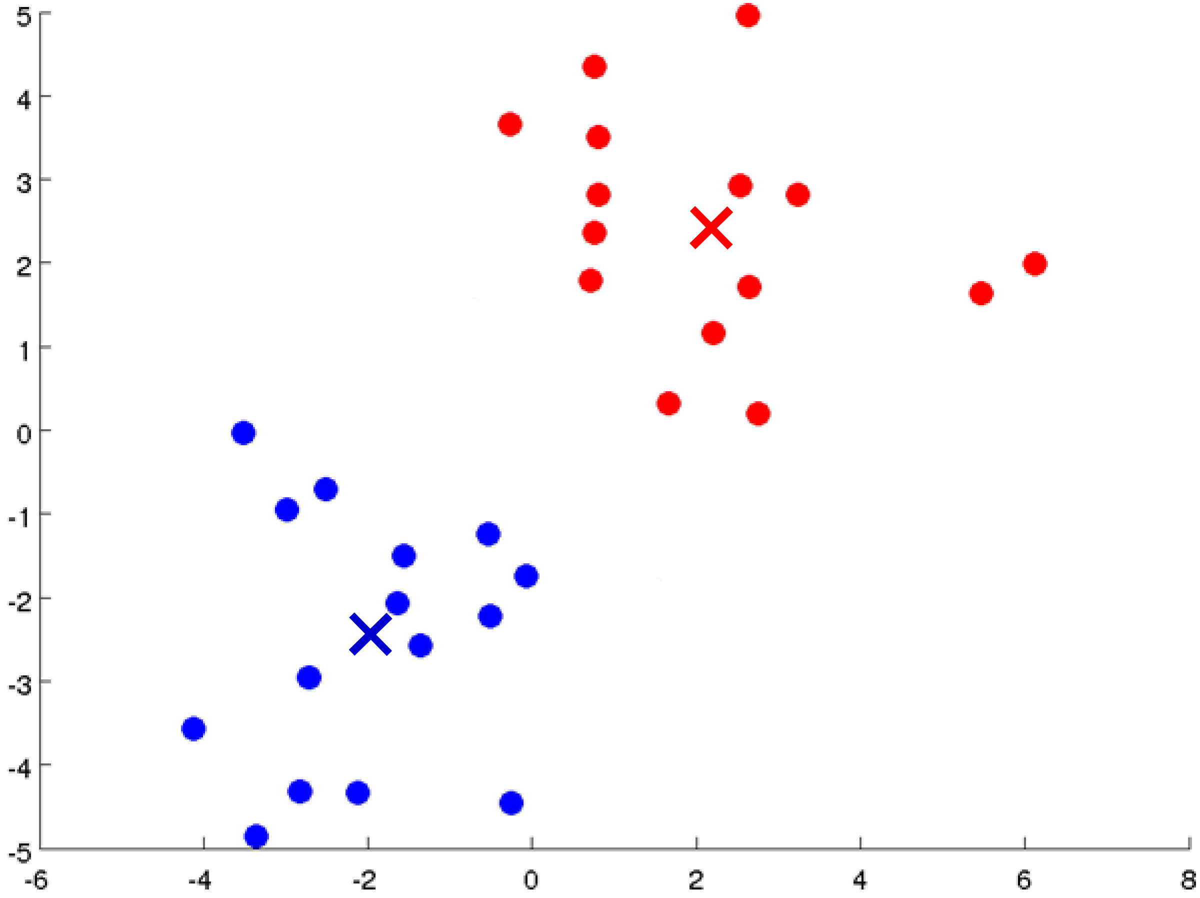
K-means Clustering



K-means Clustering



K-means Clustering



K-means Concepts

Input:

- K (number of clusters)
- Training set $\{x^{(1)}, x^{(2)}, \dots, x^{(\mathrm{m})}\}$ $\quad x^{(\mathrm{i})} \in R^n$

(Algorithm running in n-dimensional space)

Compute distance: Take L_2 or Euclidean norm of the difference

$$x = (x_1, x_2, \dots, x_n)$$
$$\mathrm{y} = (\mathrm{y}_1, \mathrm{y}_2, \dots, \mathrm{y}_n)$$
$$\|x - y\| = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2 + \cdots \ (x_n - y_n)^2}$$

K-means Concepts

Input:

- K (number of clusters)
- Training set $\{x^{(1)}, x^{(2)}, \dots, x^{(\mathrm{m})}\}$ $\qquad x^{(\mathrm{i})} \in R^n$

(Algorithm running in n-dimensional space)

Taking average: Average of each coordinate

$$x = (x_1, x_2, \dots, x_n)$$
$$\mathrm{y} = (\mathrm{y}_1, \mathrm{y}_2, \dots, \mathrm{y}_n)$$

average = $\left(\frac{x_1+y_1}{2}, \frac{x_2+y_2}{2} \dots \frac{x_n+y_n}{2}\right)$

K-means Algorithm

Randomly initialize K cluster centroids $\mu_1,\ \mu_2, \dots,\ \mu_k \in R^n$

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

μ_k := average (mean) of points assigned to cluster k

}

K-means Optimization Objective

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

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

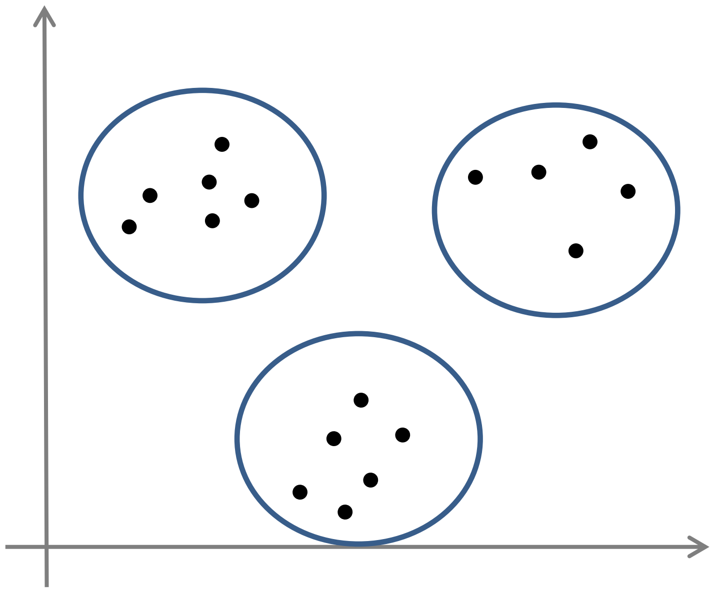
$\mu_{c^{(i)}}$= cluster centroid of cluster to which example $x^{(i)}$ has been assigned

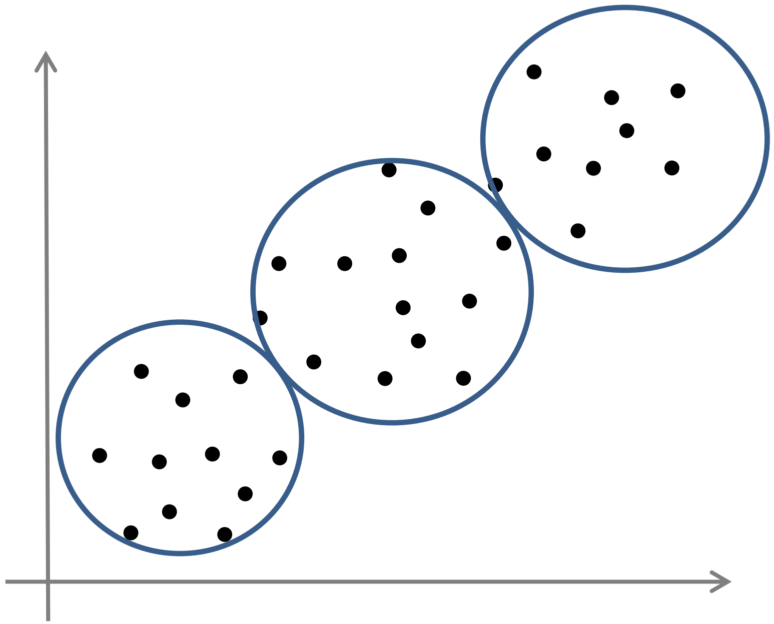
Optimization objective:

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

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

No Natural Clusters?





Questions?