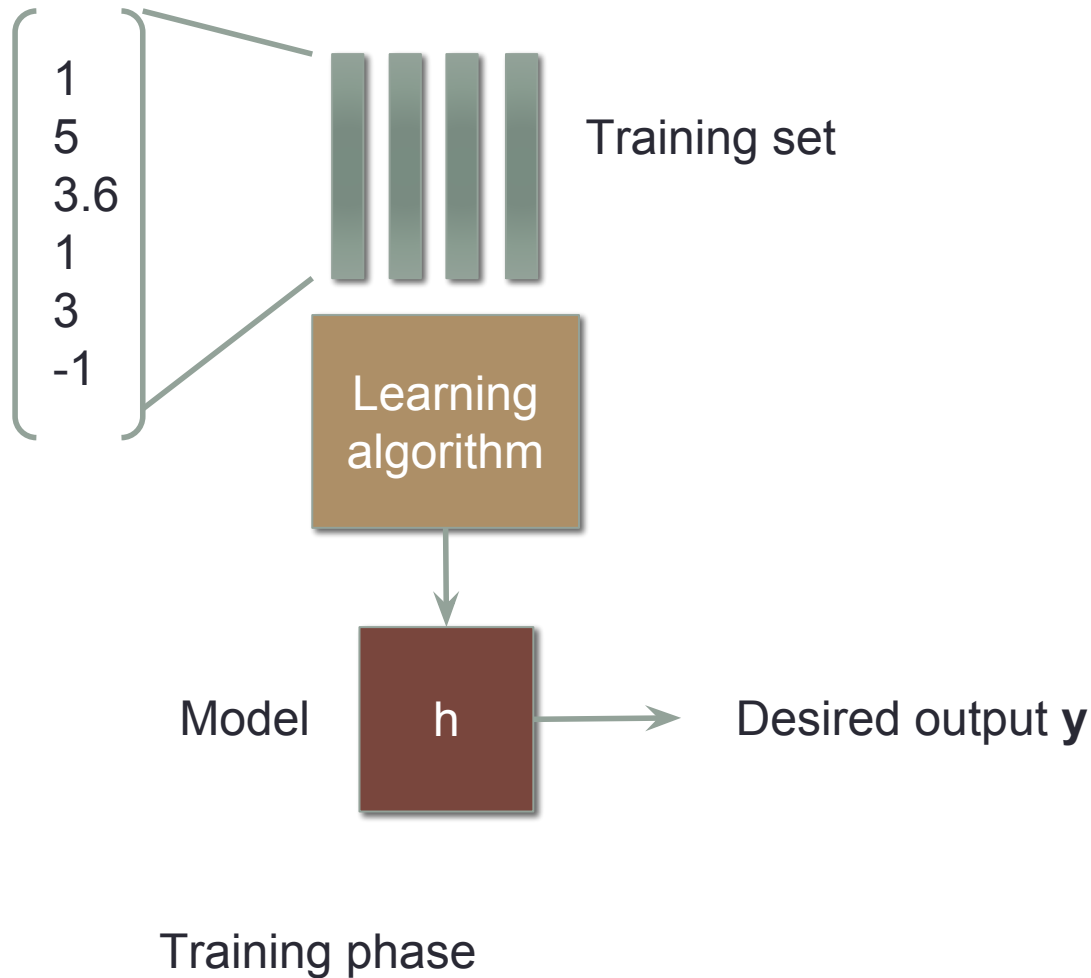


# REGRESSION

---

with some K-mean clustering

# How do we learn from data?



# Types of machine learning

1. Supervised learning

Learn a model  $F$  from pairs of  $(\mathbf{x}, y)$

2. Unsupervised learning

Discover the hidden structure in unlabeled data  $\mathbf{x}$  (no  $y$ )

3. Reinforcement learning

Train an agent to take appropriate actions in an environment by maximizing rewards

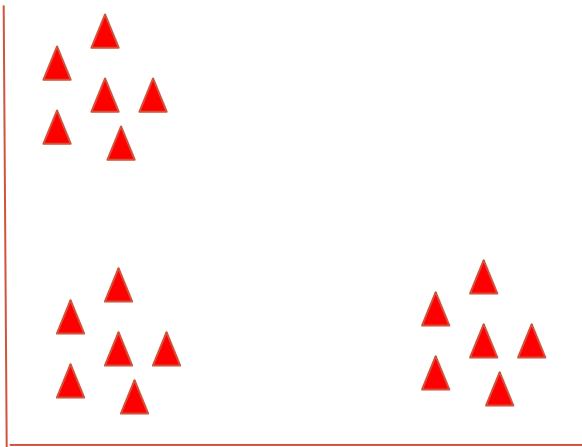
# Our first model - Unsupervised learning

Discover the hidden structure in unlabeled data  $X$  (no  $y$ )

- Customer/product segmentation
- Data analysis for ...
- Identify number of speakers in a meeting recording
- Helps supervised learning in some task

# Example - Customer analysis

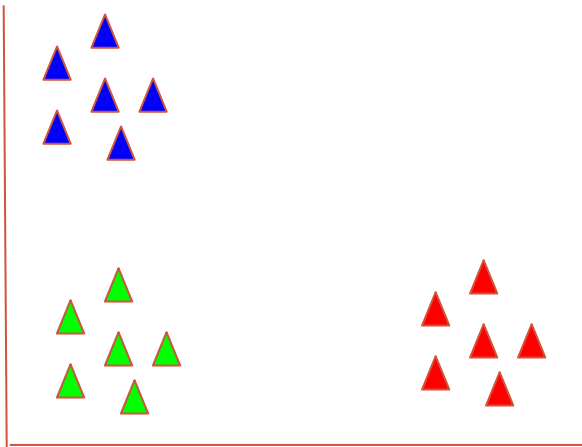
Brand royalty



Price sensitivity

# Example - Customer analysis

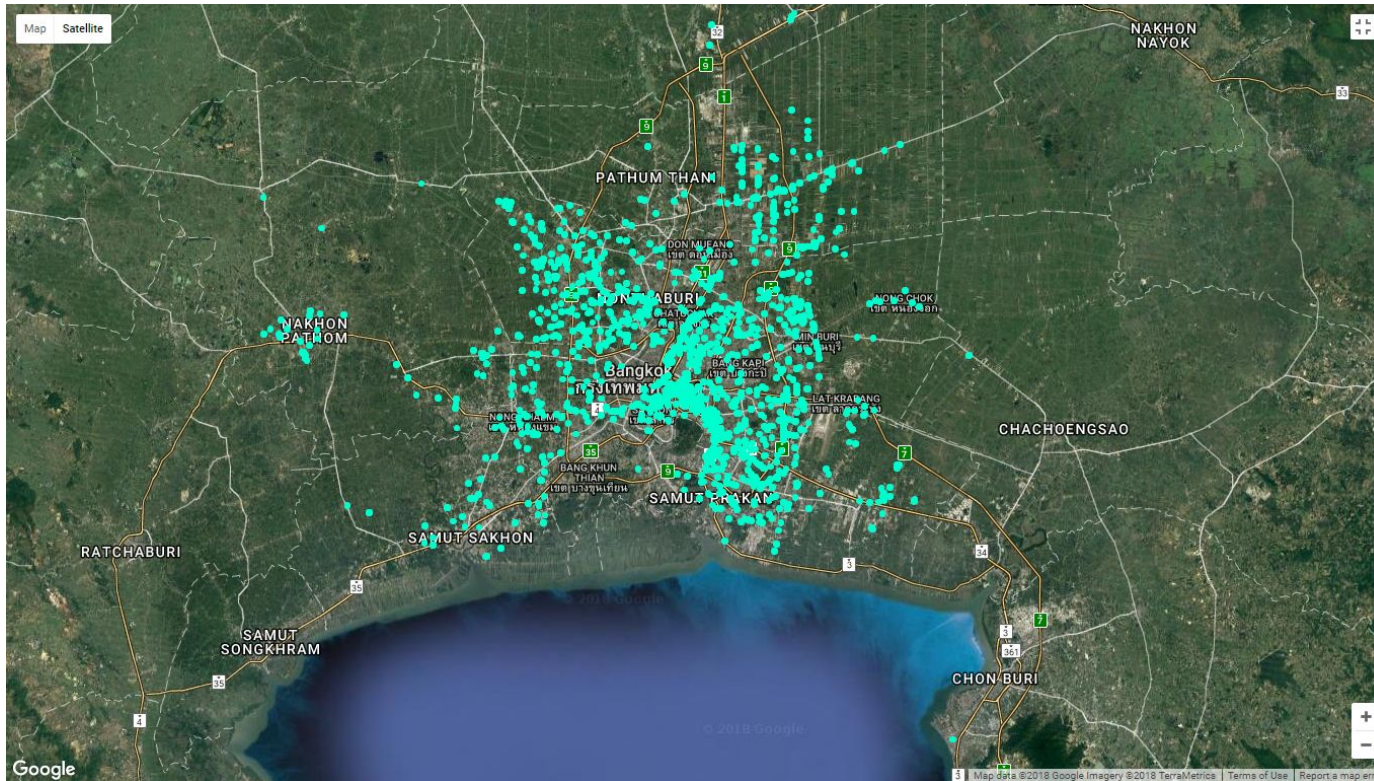
Brand royalty



Price sensitivity

# Example - Real Estate segmentation in Thailand

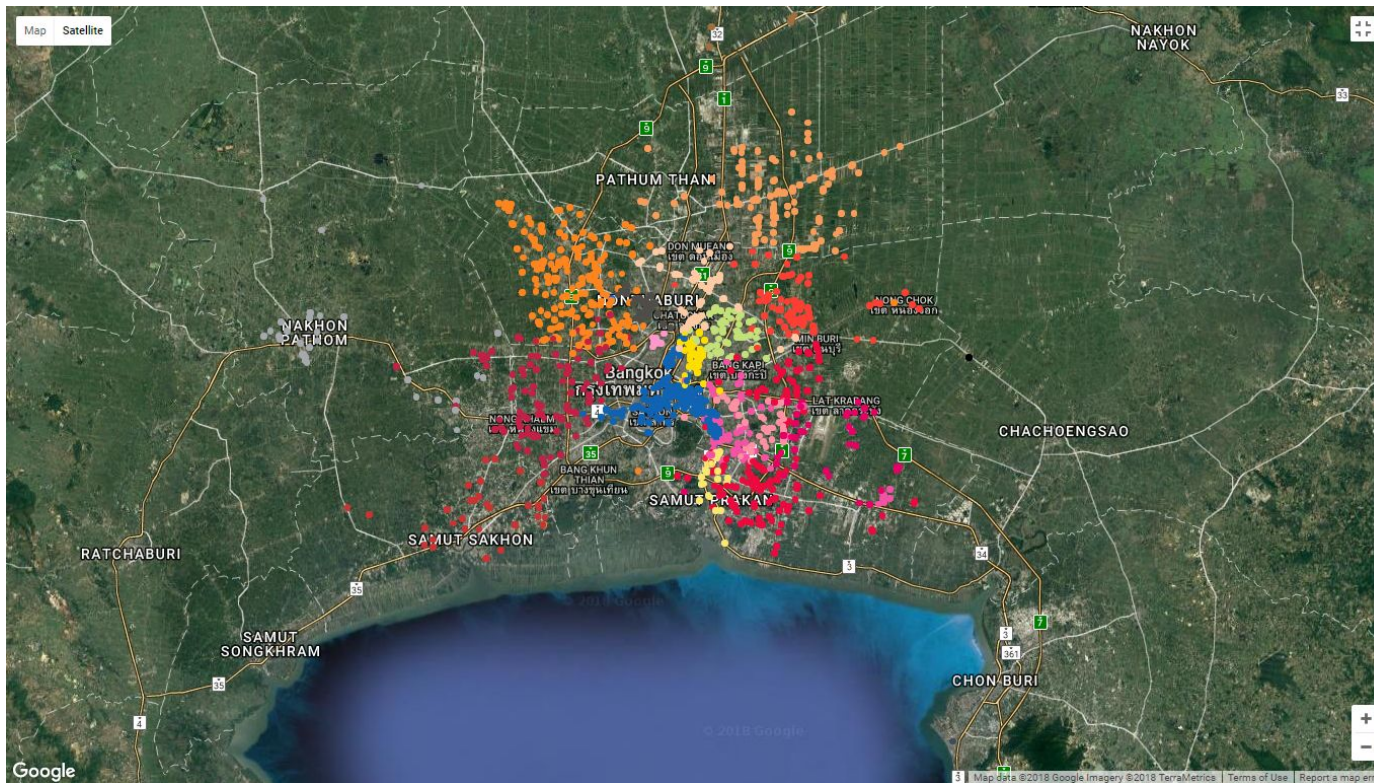
What should be the input feature of this?





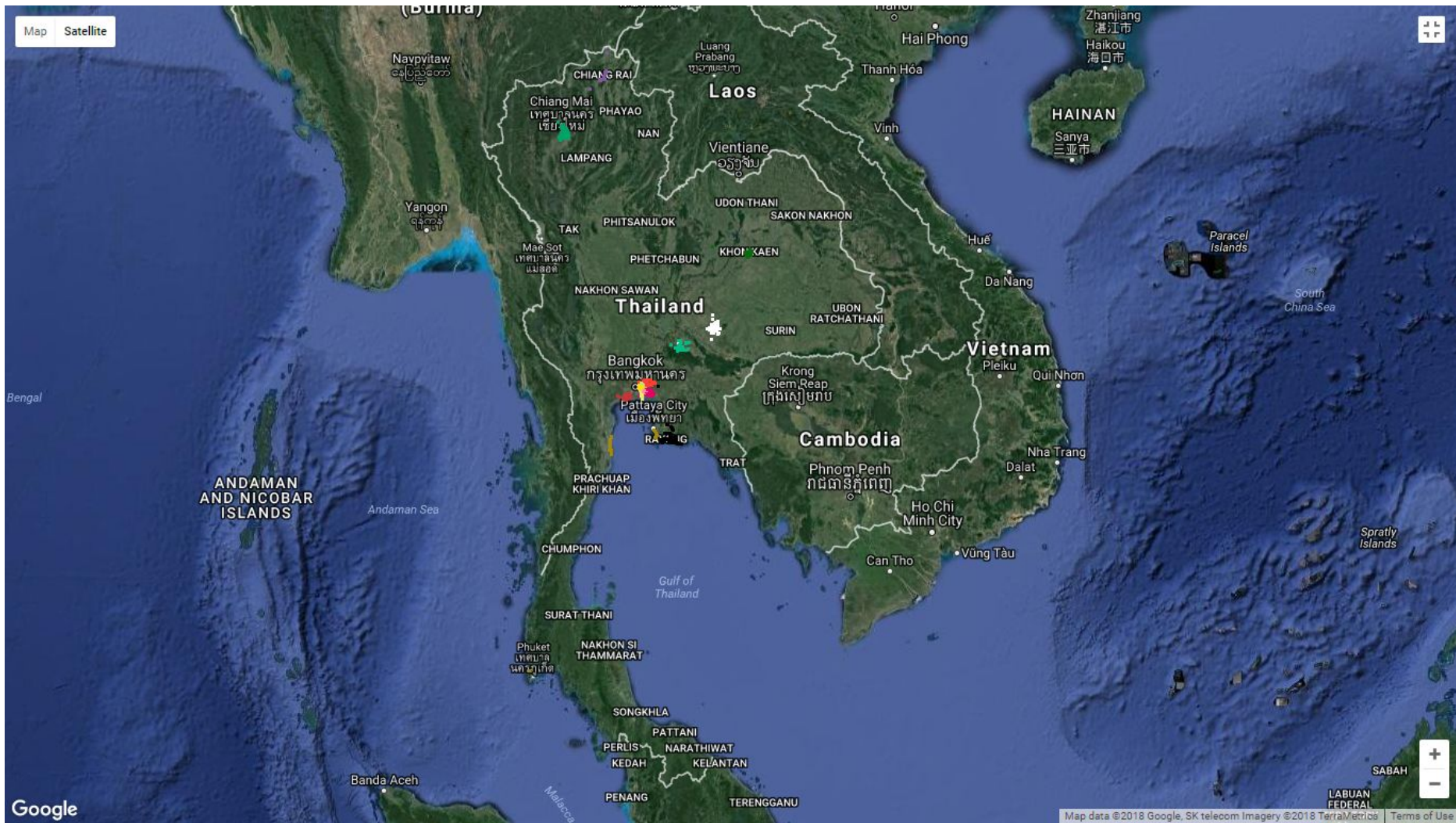
# Example - Real Estate segmentation in Thailand

What should be the input feature of this?



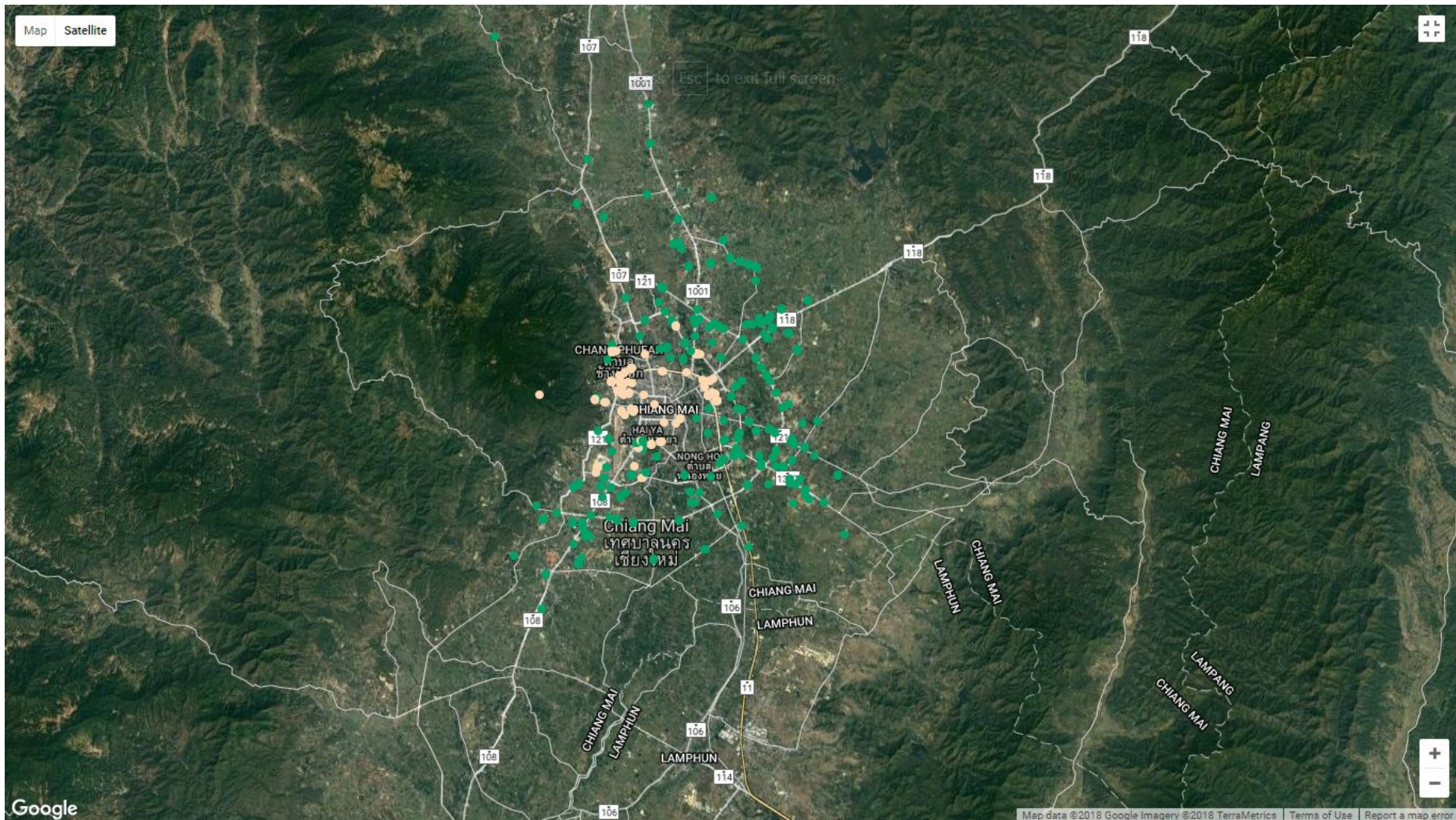


# Example - Real Estate segmentation in Thailand



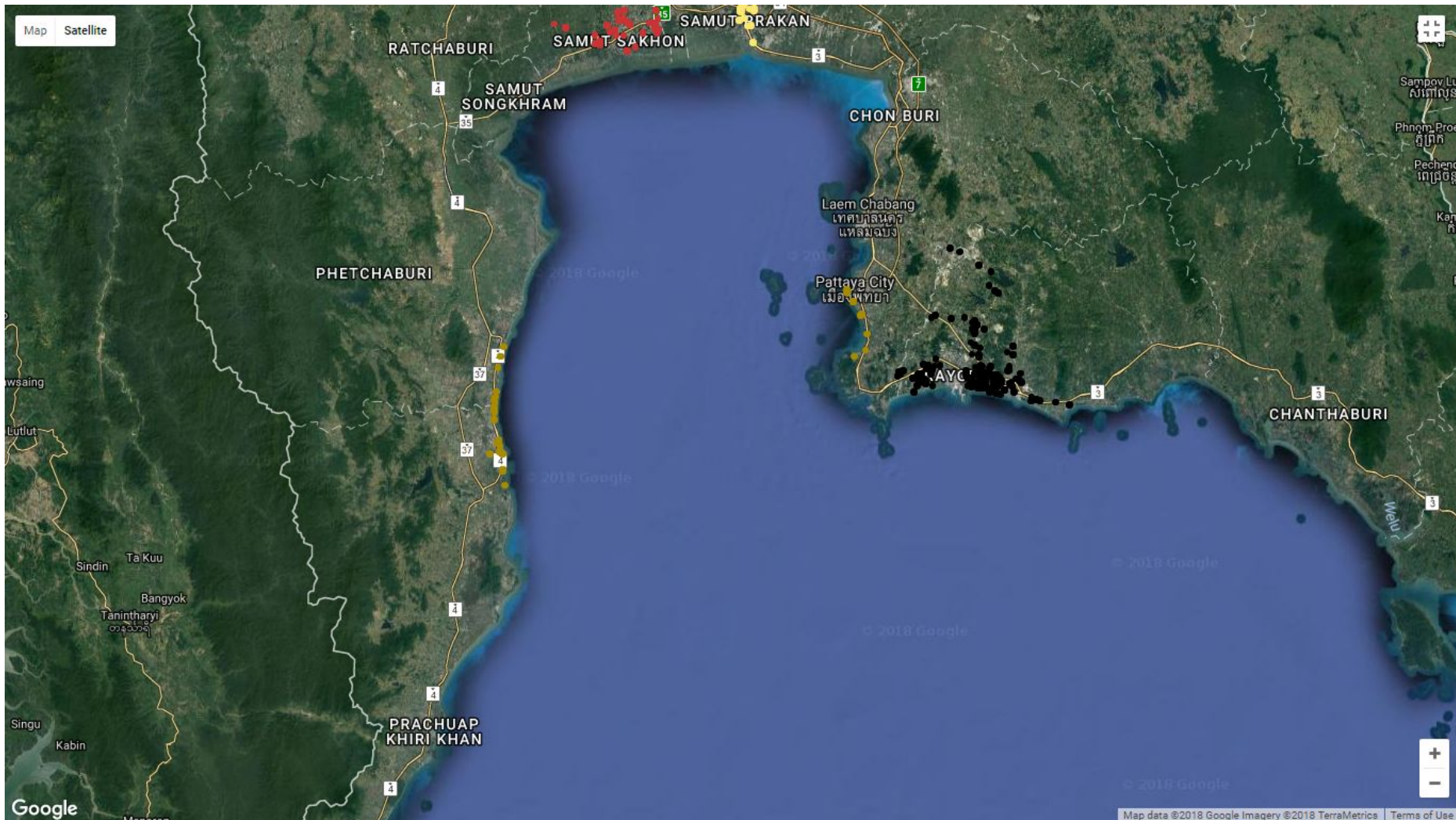


# Example - Real Estate segmentation in Thailand



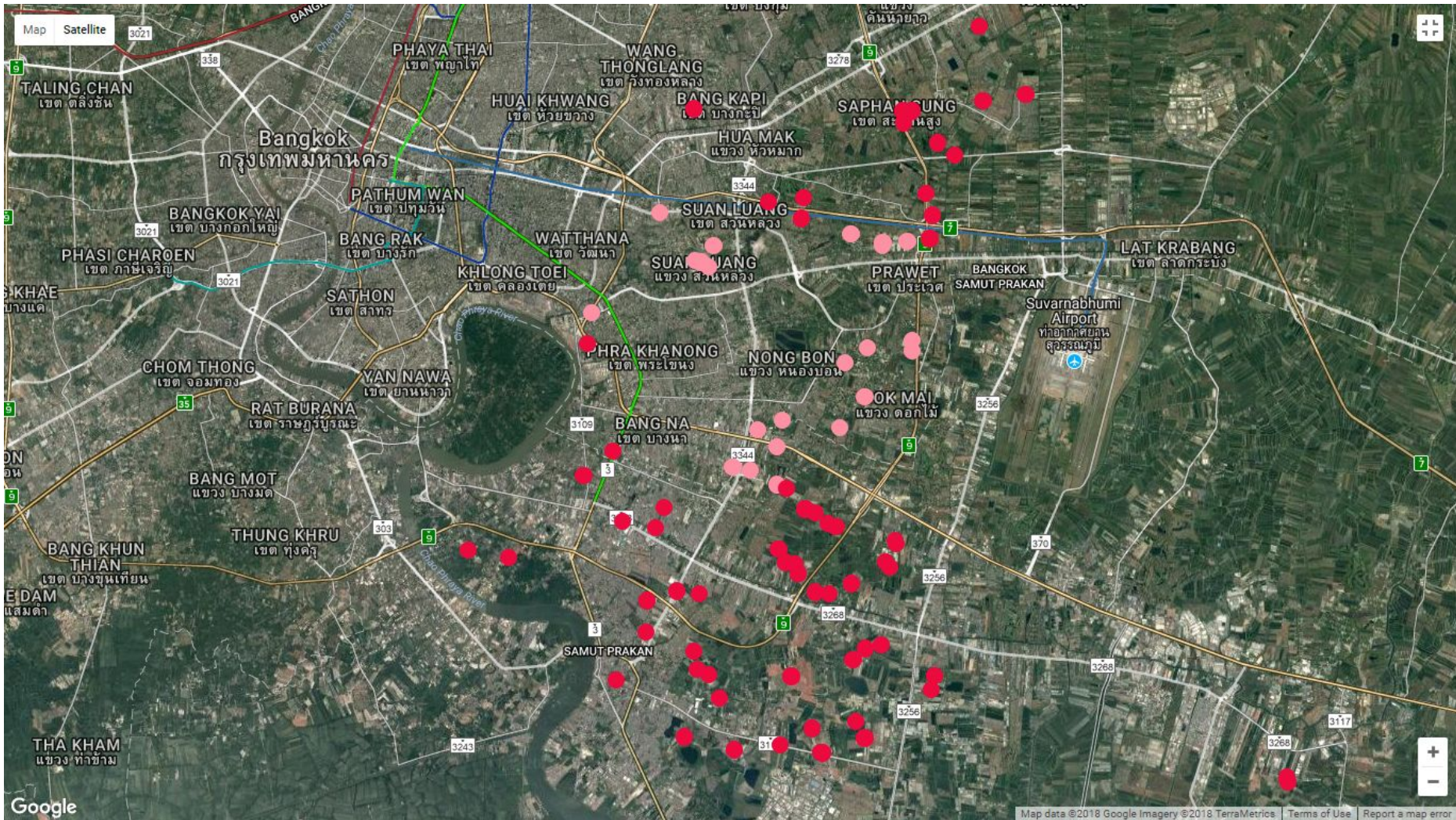


# Example - Real Estate segmentation in Thailand





# Example - Real Estate segmentation in Thailand

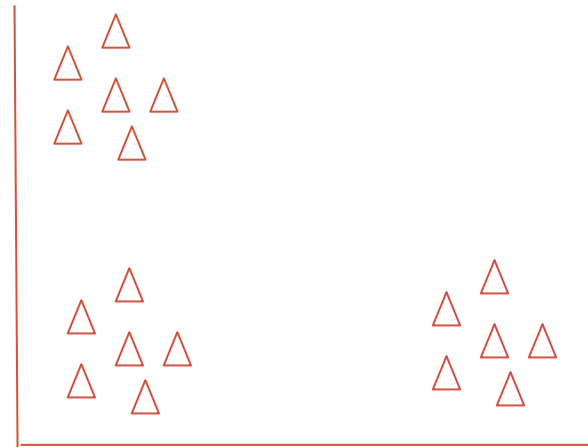


# K-mean clustering

Clustering - task that tries to automatically discover groups within the data

Too hard...

Brand royalty



Price sensitivity

# K-mean clustering

Clustering - task that tries to automatically discover groups within the data

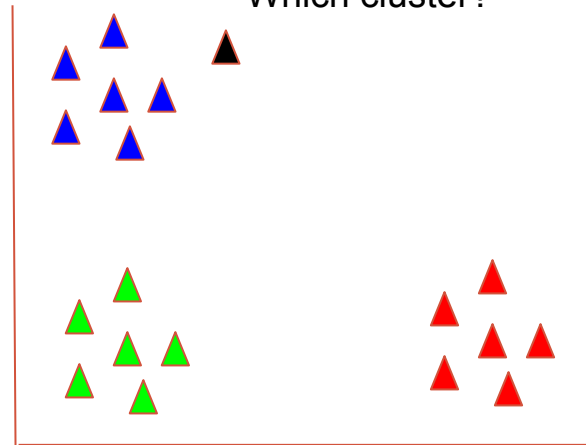
Too hard...

Easier if we know the  
grouping beforehand  
(supervised)

How?

Brand royalty

Which cluster?



Price sensitivity



# Nearest Neighbour classification

Find the closest training data, assign the same label as the training data

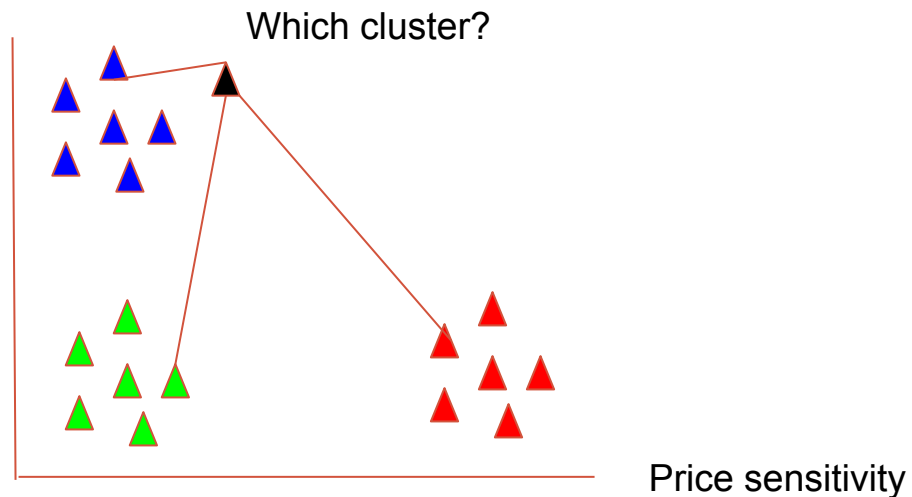
Given query data

For every point in the training data

Compute the distance with the query

Assign label of the smallest distance

Brand royalty

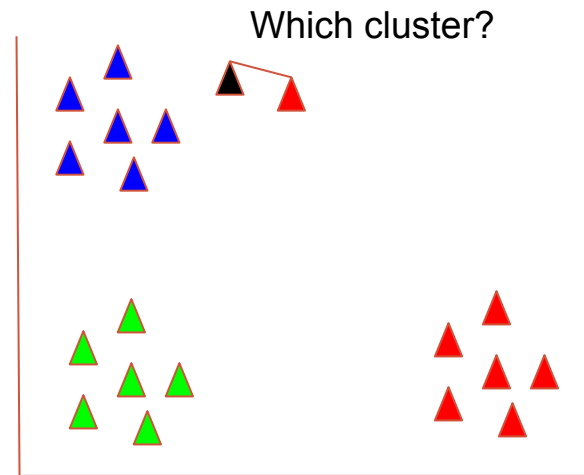


# K-Nearest Neighbour (kNN) classification

Nearest Neighbour is susceptible to noise in the training data

Use a voting scheme instead

Brand royalty



# K-Nearest Neighbour (kNN) classification

Nearest Neighbour is susceptible to noise in the training data

Use a voting scheme instead

Given query data

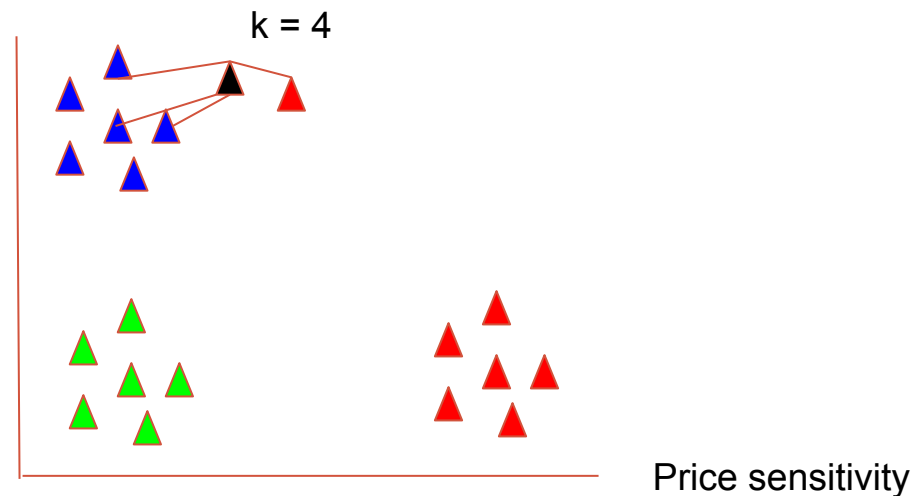
For every point in the training data

Compute the distance with the query

Find the K closest data points

Assign label by voting

Brand royalty



# K-Nearest Neighbour (kNN) classification

Nearest Neighbour is susceptible to noise in the training data

Use a voting scheme instead

Given query data

For every point in the training data

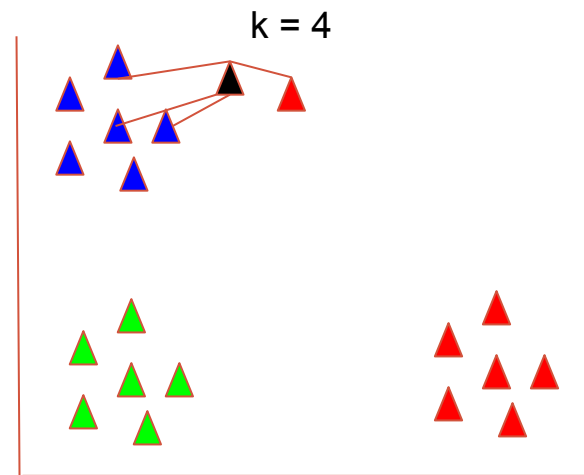
Compute the distance with the query

Find the K closest data points

Assign label by voting

The votes can be weighted by the  
inverse distance (weighted k-NN)

Brand royalty



Price sensitivity

# Closest?

We need some kind of **distance** or **similarity** measures

$$F(\mathbf{X}_1, \mathbf{X}_2) = d \quad \mathbf{X}_1 = [x_{1,1}, x_{1,2}, \dots, x_{1,n}]$$

$$\mathbf{X}_2 = [x_{2,1}, x_{2,2}, \dots, x_{2,n}]$$

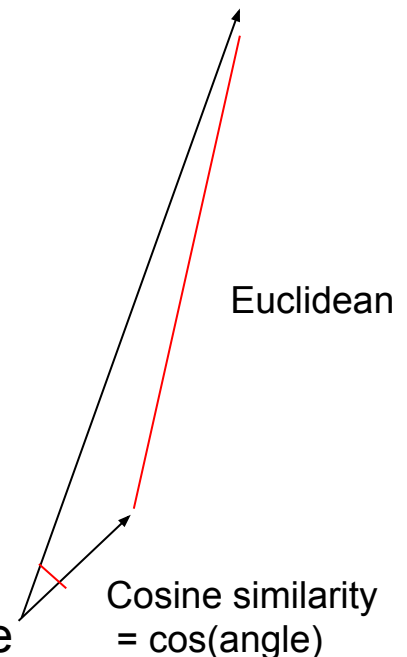
Euclidean distance

$$\sqrt{\sum_i (x_{1,i} - x_{2,i})^2}$$

Cosine similarity

$$\frac{\mathbf{X}_1 \cdot \mathbf{X}_2}{|\mathbf{X}_1| |\mathbf{X}_2|} = \frac{\sum_i x_{1,i} x_{2,i}}{\sqrt{\sum_i x_{1,i}^2} \sqrt{\sum_i x_{2,i}^2}}$$

Many more distances, Jaccard distance, Earth mover distance



# KNN runtime

For every point in the training data

- Compute the distance with the query

- Find the K closest data points

- Assign label by voting

$O(N)$

$O(JN)$  - If we have J queries

Expensive

Ways to make it faster

- Kernelized KNN

- Locally Sensitive Hashing (LSH)

- Use centroids



# Centroids

Basically the **representative of the cluster**

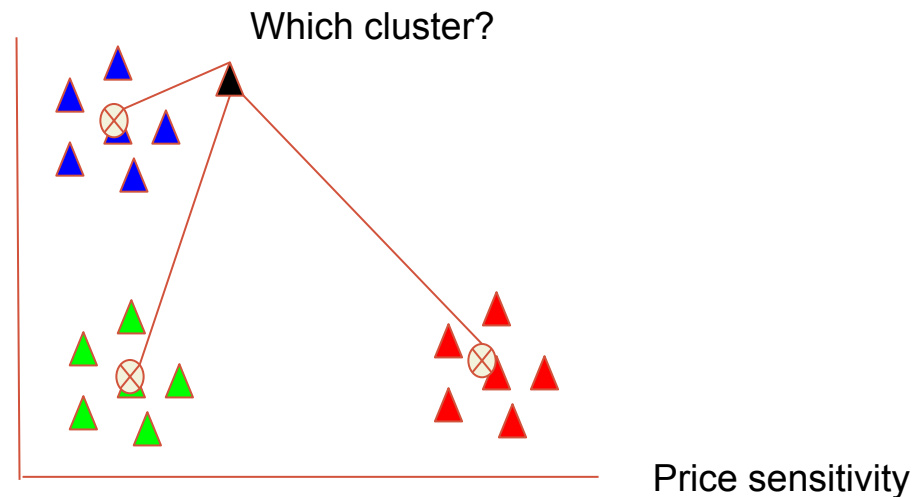
Find the mean location of the cluster by averaging

Can use mode or median depending on the data

$O(JL)$

L - number of clusters

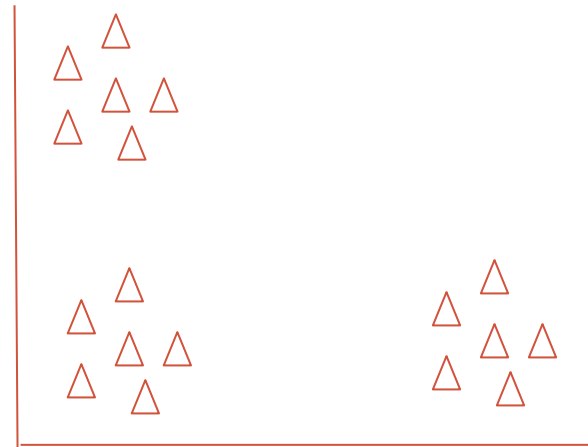
Brand royalty



# K-mean clustering

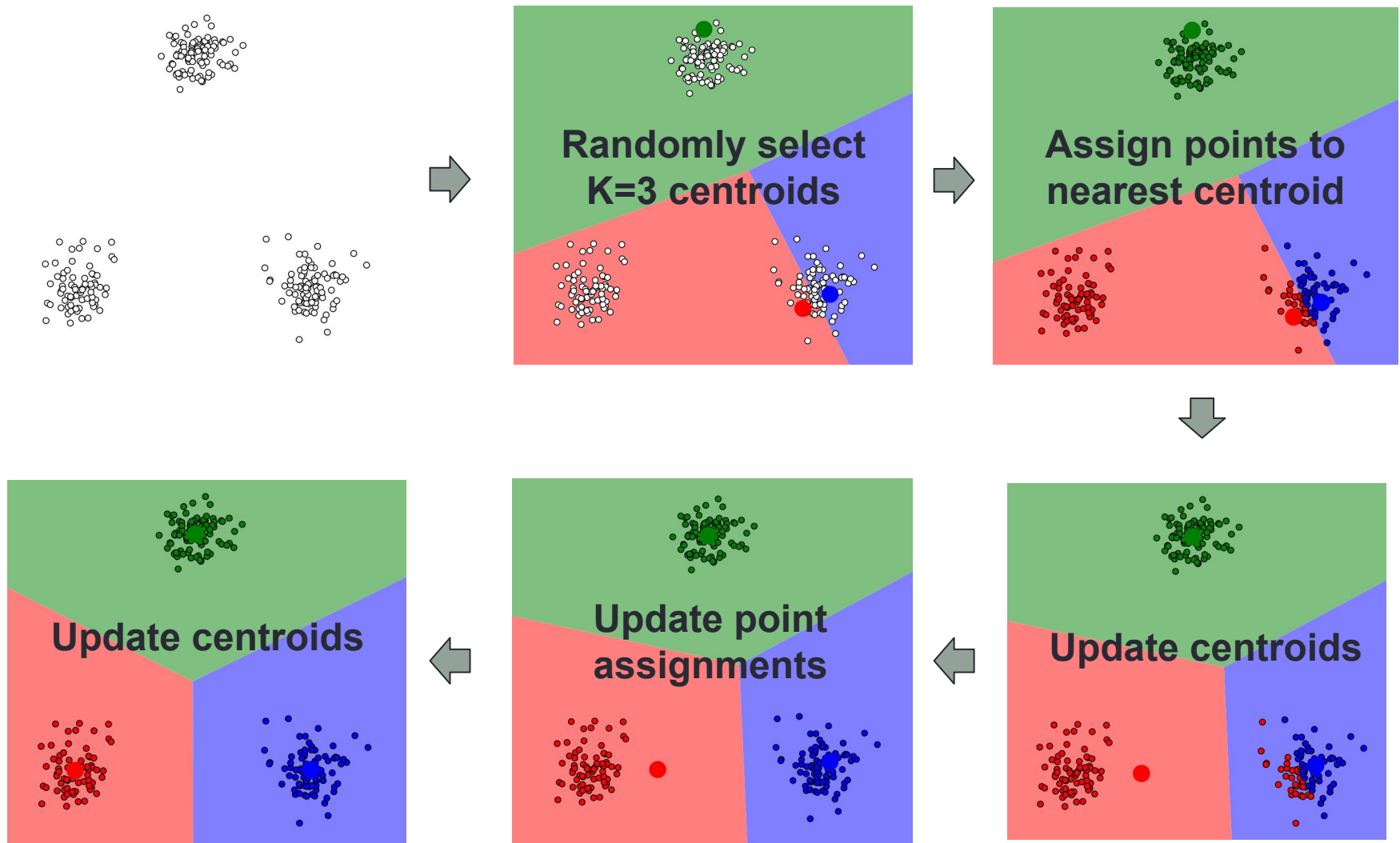
1. Randomly k centroids by picking from data points
2. Assign each data points to centroids
3. Update centroids for each cluster
4. Repeat 2-3 until centroids does not change

Brand royalty



Price sensitivity

# An Illustration Of K-Mean Clustering



# Characteristics of K-means

- The number of clusters,  $K$ , is specified in advance.
- Always converge to a (local) minimum.
  - Poor starting centroid locations can lead to incorrect minima.

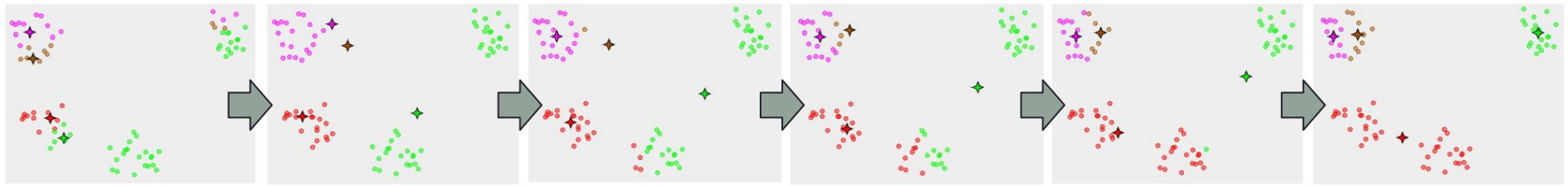
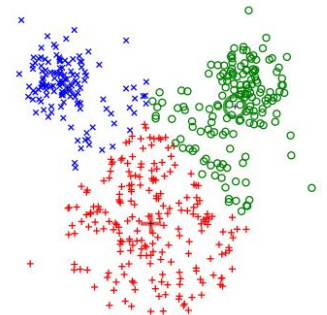
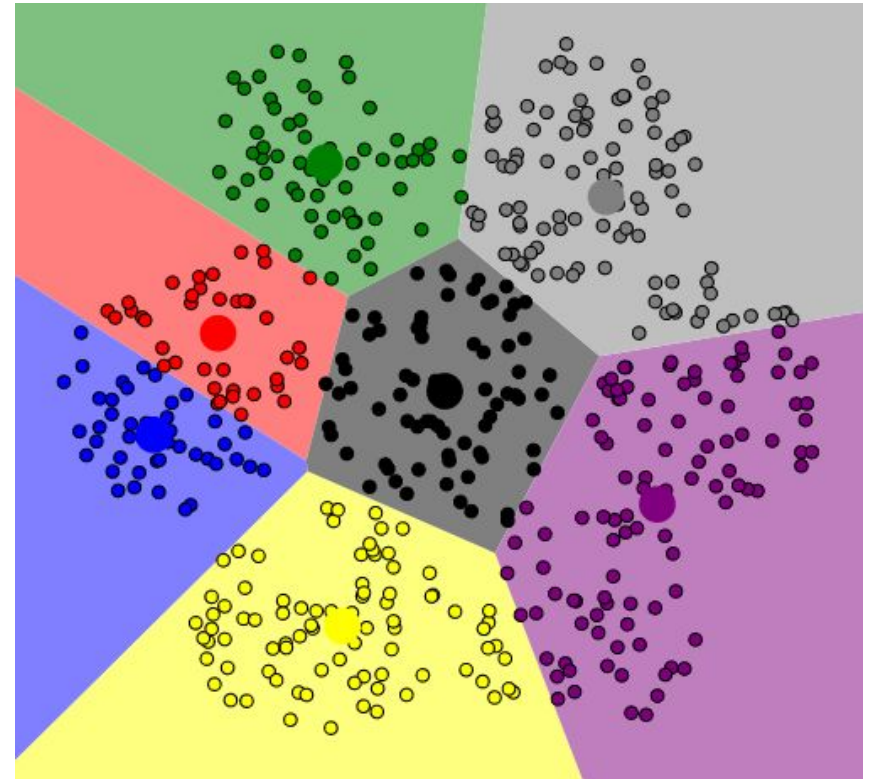
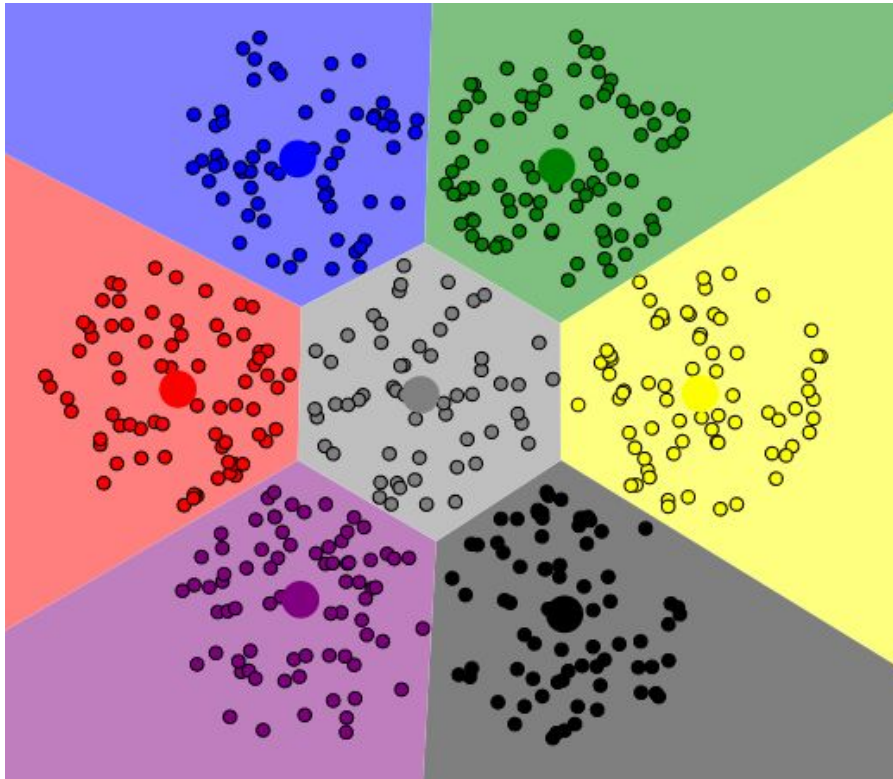


Image from [https://en.wikipedia.org/wiki/K-means\\_clustering](https://en.wikipedia.org/wiki/K-means_clustering)

- The model has several implicit assumptions:
  - Data points scatter around cluster's centers.
  - Boundary between adjacent clusters is always halfway between the cluster centroids.

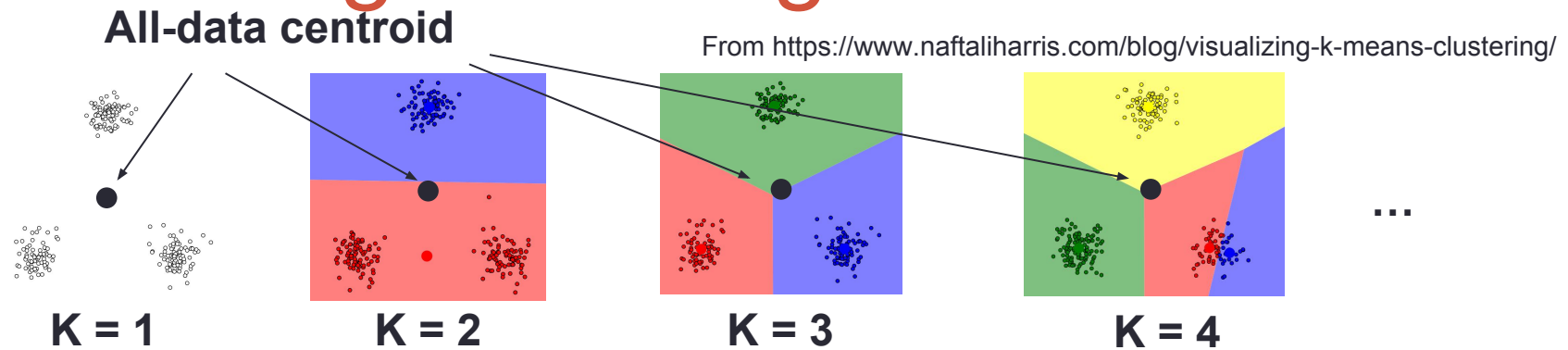


# Effect of bad initializations



Solution, try different randomization and pick the best

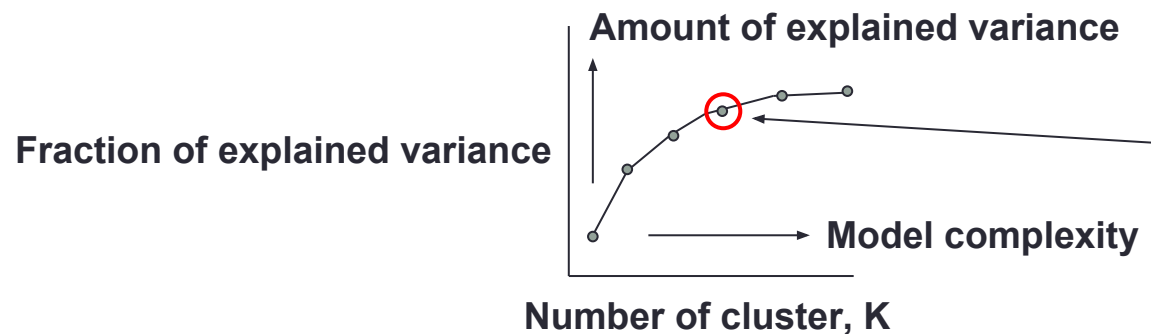
# Selecting K - Using Elbow method



**fraction of explained variance =** 
$$\frac{\text{between-cluster variance}}{\text{all-data variance}}$$

**between-cluster variance** = 
$$\sum_{i=1}^K \frac{n_i (M_i - M)^2}{N - 1}$$
, where  $n_i$  = size of  $i^{\text{th}}$  cluster,  
 $M_i$  = centroid of  $i^{\text{th}}$  cluster, and  
 $M$  = all-data centroid.

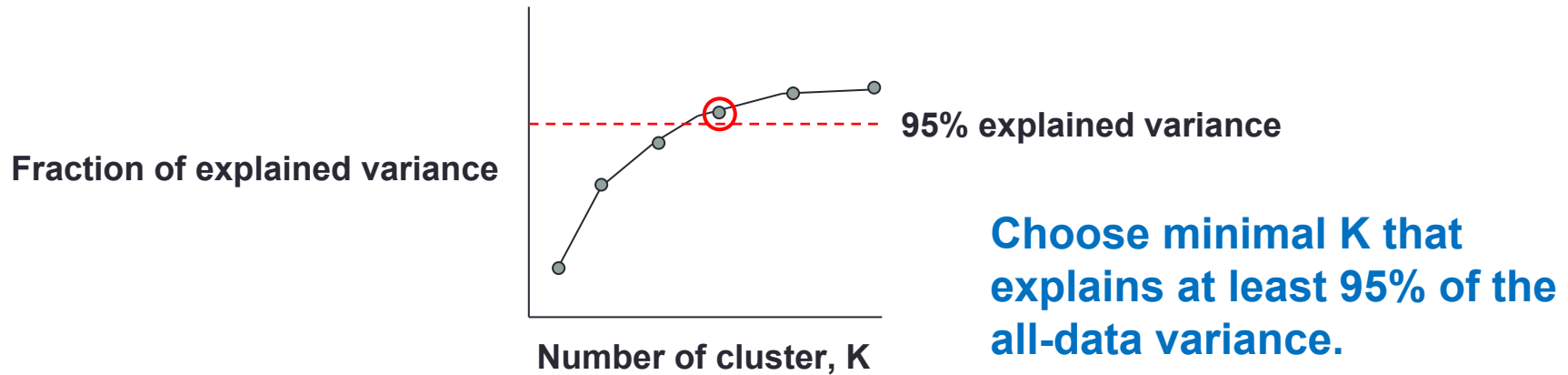
**all-data variance** = 
$$\sum_{i=1}^N \frac{(x_i - M)^2}{N - 1}$$
, where  $x_i$  =  $i^{\text{th}}$  data point and  $N$  = # of data.



**The elbow method chooses K where increasing complexity doesn't yield much in return.**



# Selecting K - other methods

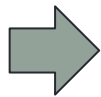


K = 2

K = 3

K = 4

⋮



Training  
K-mean  
Clustering  
Model



Testing /  
Cross-validation



K	Accuracy
2	50%
3	68%
4	83%
⋮	⋮

Choose K that maximizes  
certain objective (e.g.  
accuracy on testing data)

Best method

# REGRESSION

---

with some K-mean clustering

# Let's look at another example



<https://soclaimon.wordpress.com/2015/07/24/%E0%B9%82%E0%B8%A1%E0%B9%80%E0%B8%94%E0%B8%A5%E0%B8%99%E0%B9%89%E0%B8%B33%E0%B8%A2%E0%B8%B8%E0%B8%84%E0%B8%A1%E0%B8%B2%E0%B8%A3%E0%B9%8C%E0%B8%84-%E0%B8%9B%E0%B8%B9/>

# Predicting amount of rainfall



# Predicting amount of rainfall

Cloth	Corn	Grass	Water	Beer	Rainfall
4	6	3	10	0	<b>76950</b>
5	1	0	0	7	<b>30234</b>
6	0	3	5	7	<b>123456</b>
5	0	3	12	0	<b>89301</b>
4	3	0	6	7	<b>?</b>

We assume the input features have some correlation with the amount of rainfall.

Can we create a model that predict the amount of rainfall?

What is the output?

What is the input (features)?

# Predicting the amount of rainfall

- The correlation can be positive or negative

เสี่ยงทายผ้าหุง

 ผ้า 6 คืบ น้ำจะน้อย  
นาที่ลุ่มจะได้ผลดี  
นาที่ดอนจะเสียหาย

 ผ้า 5 คืบ น้ำปริมาณพอดี  
ข้าวกล้าในนาจะได้บุญ

 ผ้า 4 คืบ น้ำจะมาก  
นาที่ดอนจะได้ผลดี  
นาที่ลุ่มจะเสียหาย

 ประเทศไทยอยู่ตรงไหน?  
[whereisthailand.info](http://whereisthailand.info)



# Predicting the amount of rainfall

Cloth	Corn	Grass	Water	Beer	Rainfall
4	6	3	10	0	<b>76950</b>
5	1	0	0	7	<b>30234</b>
6	0	3	5	7	<b>123456</b>
5	0	3	12	0	<b>89301</b>
4	3	0	6	7	<b>?</b>

Can we create a model that predict the amount of rainfall?

What is the output?

What is the input (features)?

# Predicting the amount of rainfall

Cloth	Corn	Grass	Water	Beer	Rainfall
4	6	3	10	0	<b>76950</b>
5	1	0	0	7	<b>30234</b>
6	0	3	5	7	<b>123456</b>
5	0	3	12	0	<b>89301</b>
4	3	0	6	7	<b>?</b>

- $$h_{\theta}(\mathbf{x}_1) = \theta_0 + \theta_1 x_{1,1} + \theta_2 x_{1,2} + \theta_3 x_{1,3} + \theta_4 x_{1,4} + \theta_5 x_{1,5}$$

1 refers to index of the data (the first in the training/test set)

- Where  $\theta$ s are the parameter of the model
- $X$ s are values in the table

# (Linear) Regression

- $h_{\theta}(\mathbf{x}_1) = \theta_0 + \theta_1 x_{1,1} + \theta_2 x_{1,2} + \theta_3 x_{1,3} + \theta_4 x_{1,4} + \theta_5 x_{1,5}$

- $\theta$ s are the parameter (or weights)

Assume  $x_0$  is always 1

- We can rewrite  $n$  is dimension of  $x$

$$h_{\theta}(\mathbf{x}_i) = \sum_{j=0}^n \theta_j x_{i,j} = \theta^T \mathbf{x}_i$$

$h$  is parameterized by  $\theta$

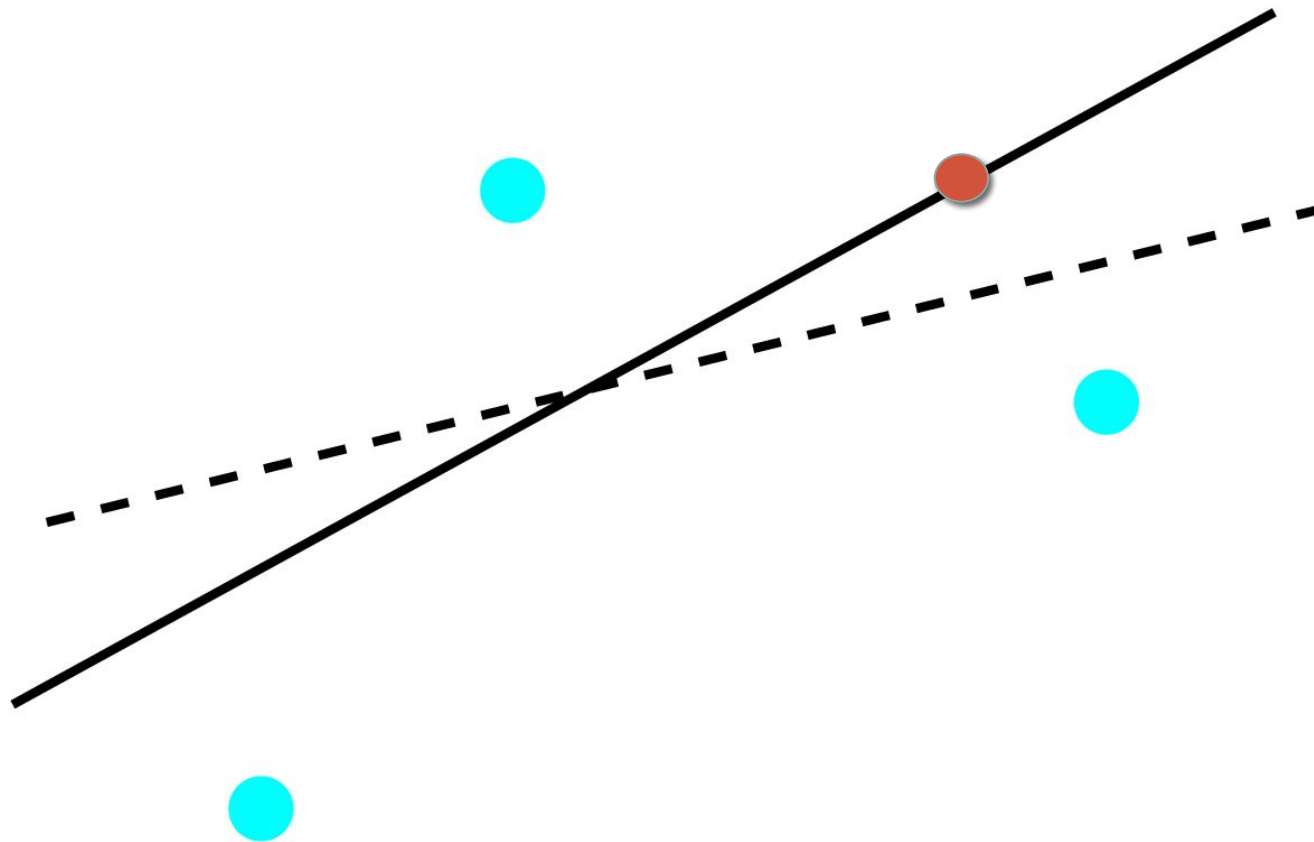
- Notation: vectors are bolded
- Notation: vectors are column vectors

# Picking $\theta$

- Random until you get the best performance?
- How to quantify best performance?

$$h_{\theta}(\mathbf{x}_i) = \sum_{j=0}^n \theta_j x_{i,j} = \theta^T \mathbf{x}_i$$

$$h_{\theta}(\mathbf{x}_i) = \sum_{j=0}^n \theta_j x_{i,j} = \theta^T \mathbf{x}_i$$



# Cost function (Loss function)

- Let's use the mean square error (MSE)

$$J(\theta) = \frac{1}{m} \sum_{i=1}^m (y_i - \theta^T \mathbf{x}_i)^2$$

*m is the number of training examples*



We want to pick  $\theta$  that minimize the loss

*i here is the index of the training example*

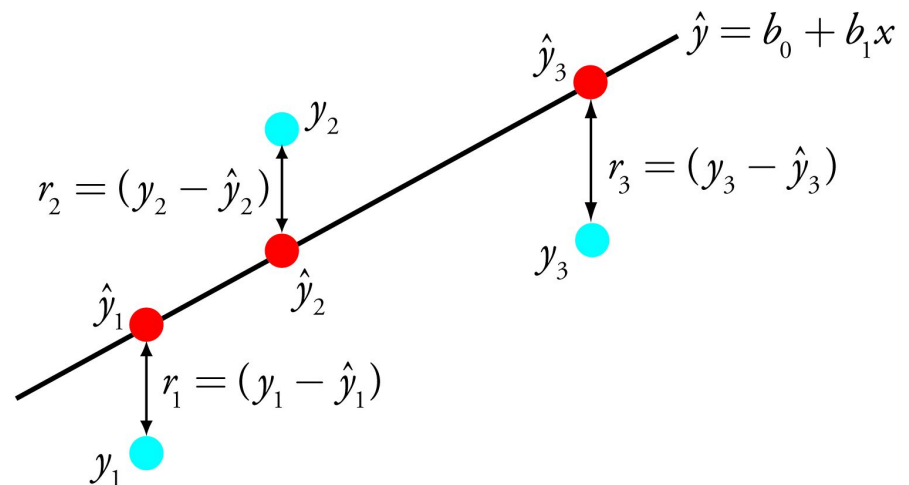
*Note how  $\mathbf{x}$  is bolded*

# Cost function (Loss function)

- Let's use the mean square error (MSE)

$$J(\theta) = \frac{1}{m} \sum_{i=1}^m (y_i - \theta^T \mathbf{x}_i)^2$$

We want to pick  $\theta$  that minimize the loss



# Cost function (Loss function)

- Let's use the mean square error (MSE)

$$J(\theta) = \frac{1}{m} \sum_{i=1}^m (y_i - \theta^T \mathbf{x}_i)^2$$



We want to pick  $\theta$  that minimize the loss

$$\frac{m}{2} J(\theta) = \frac{1}{2} \sum_{i=1}^m (y_i - \theta^T \mathbf{x}_i)^2$$



# Picking $\theta$

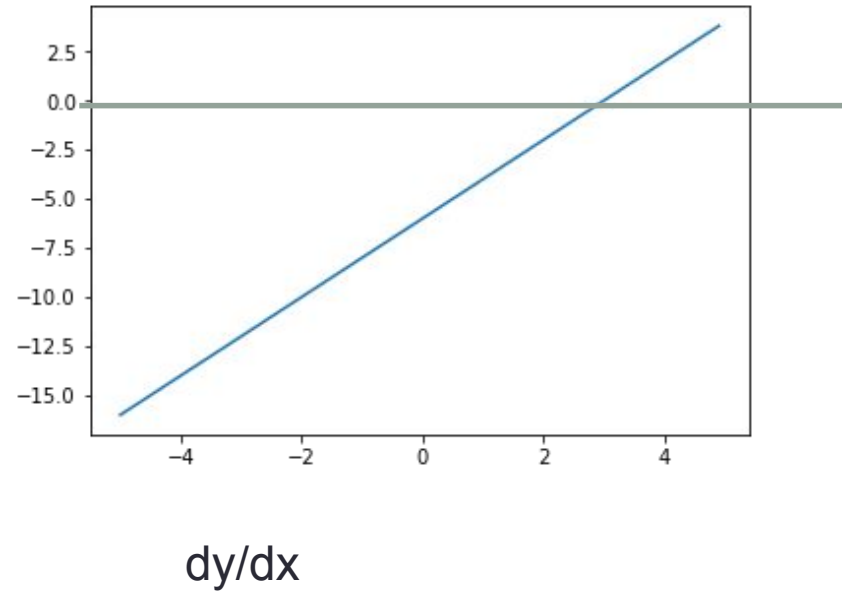
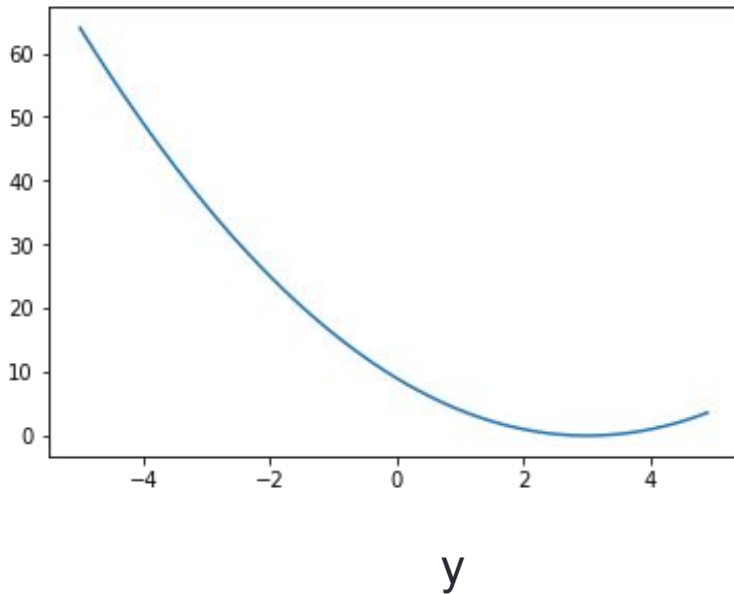
- Random until you get the best performance?
  - Can we do better than random chance?
- How to quantify best performance?

$$\frac{m}{2} J(\theta) = \frac{1}{2} \sum_{i=1}^m (y_i - \theta^T \mathbf{x}_i)^2$$

# Minimizing a function

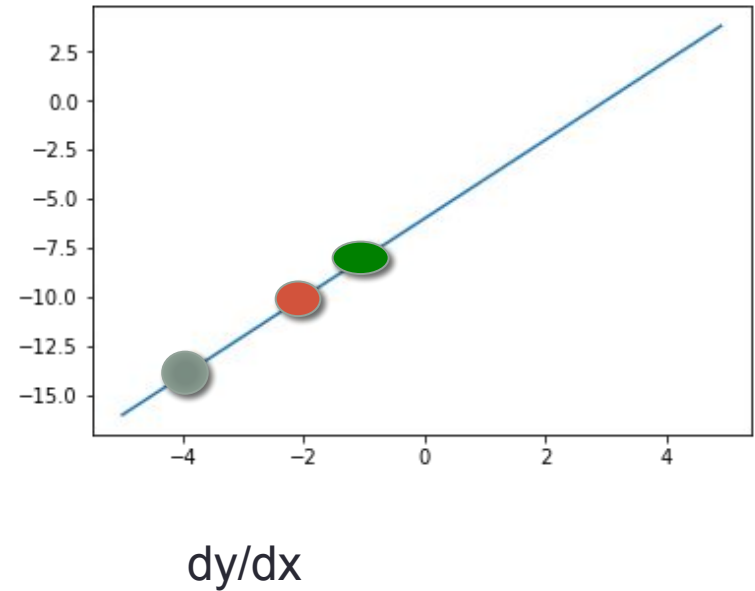
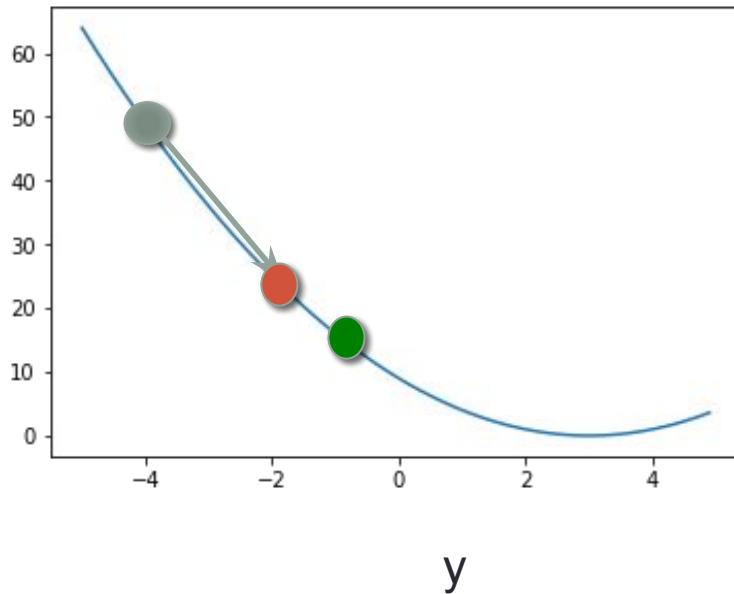
- You have a function
  - $y = (x - a)^2$
- You want to minimize Y with respect to x
  - $dy/dx = 2x - 2a$
  - Take the derivative and set the derivative to 0
    - (And maybe check if it's a minima, maxima or saddle point)
- We can also go with an iterative approach

# Gradient descent



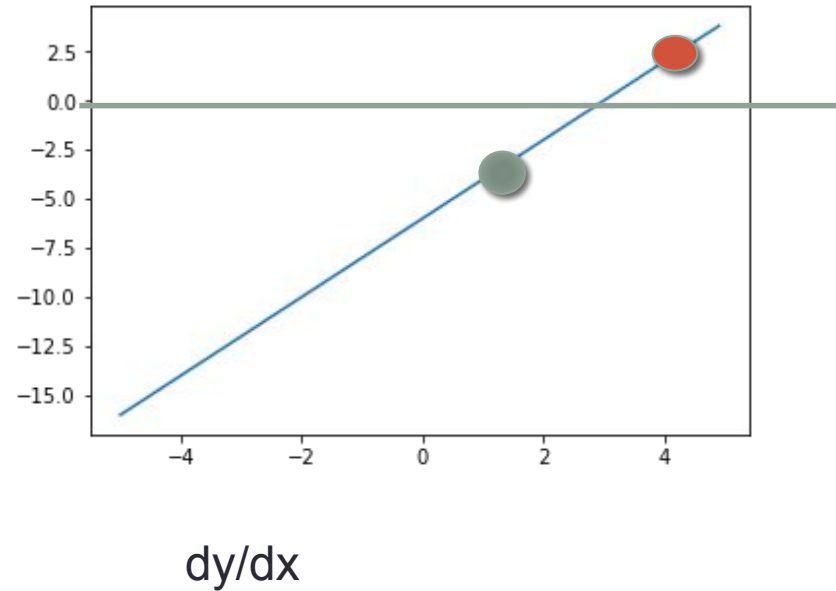
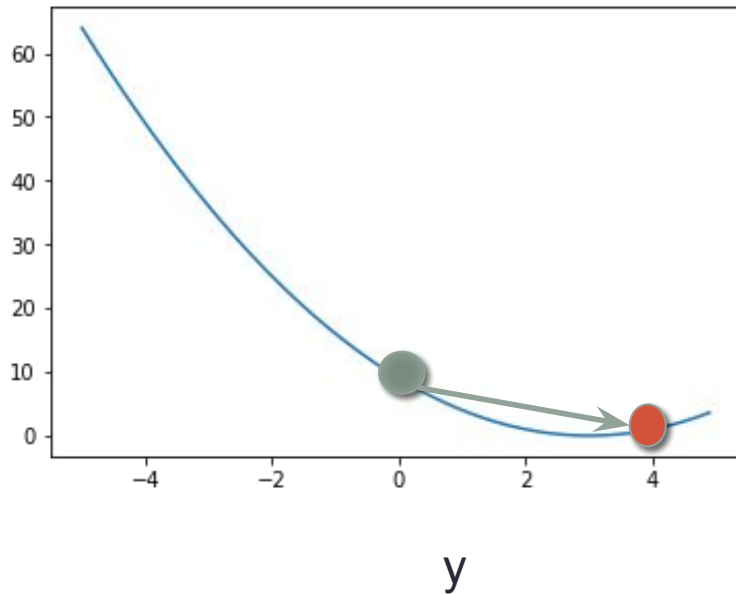
First what does  $dy/dx$  means?

# Gradient descent



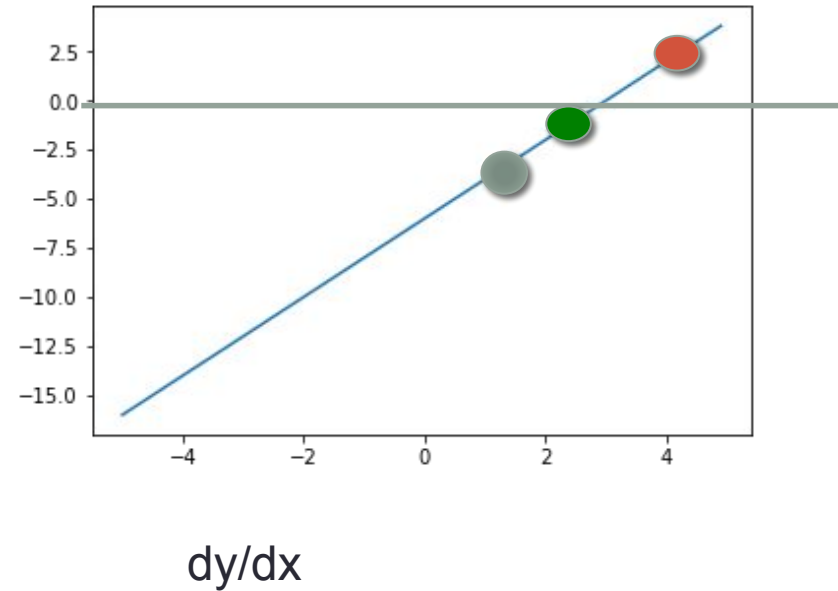
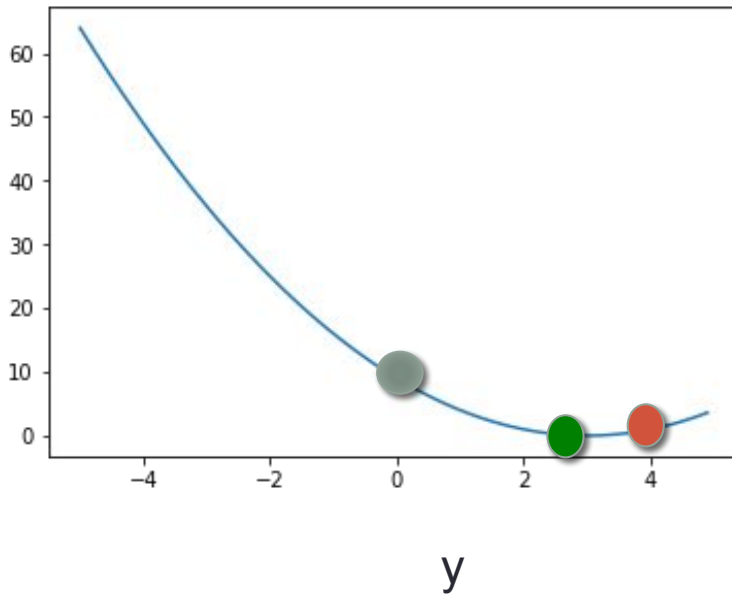
Move along the negative direction of the gradient  
The bigger the gradient the bigger step you move

# Gradient descent



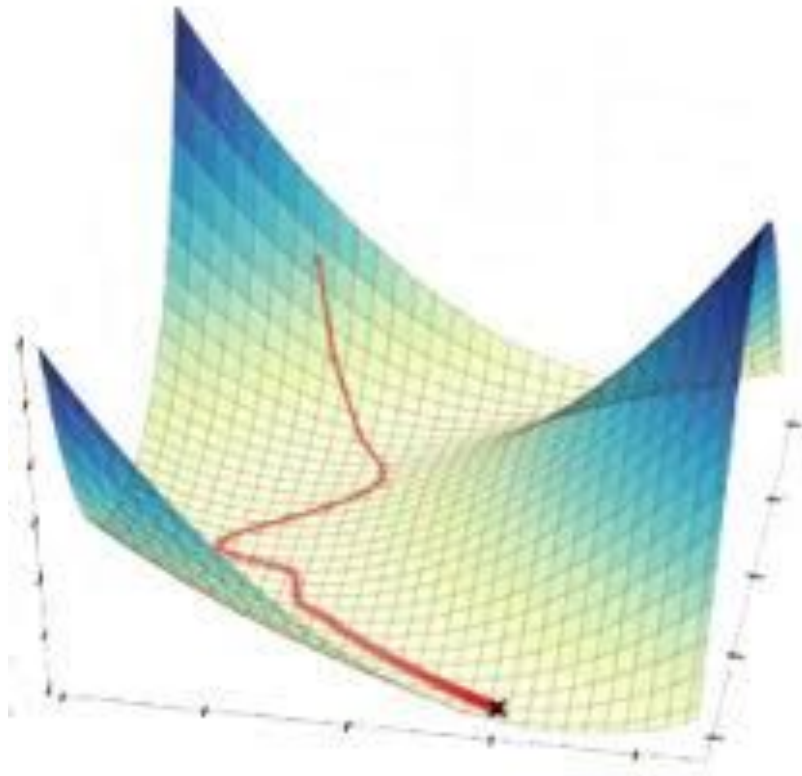
What happens when you overstep?

# Gradient descent



If you over step you can move back

# Gradient descent in 3d



# Formal definition

- $y = f(x)$
- Pick a starting point  $x_0$
- Moves along  $-dy/dx$
- $x_{n+1} = x_n - r * dy/dx$
- Repeat till convergence
- $r$  is the learning rate

Big  $r$  means you might overstep

Small  $r$  and you need to take more steps



# Picking $\theta$

- Random until you get the best performance?
  - Can we do better than random chance?
    - Gradient descent (a better guess!)
- How to quantify best performance?

$$\frac{m}{2} J(\theta) = \frac{1}{2} \sum_{i=1}^m (y_i - \theta^T \mathbf{x}_i)^2$$

## LMS regression with gradient descent

$$J(\theta) = \frac{1}{2} \sum_{i=1}^m (y_i - \theta^T \mathbf{x}_i)^2$$

$$\frac{\partial J}{\partial \theta_j} = -\sum_{i=1}^m (y_i - \theta^T \mathbf{x}_i) x_i^{(j)}$$



# LMS regression with gradient descent

$$\frac{\partial J}{\partial \theta_j} = -\sum_{i=1}^m (y_i - \theta^T \mathbf{x}_i) x_i^{(j)}$$

$$\theta_j \Leftarrow \theta_j + r \sum_{i=1}^m (y_i - \theta^T \mathbf{x}_i) x_i^{(j)}$$

Interpretation?

# Batch updates vs mini-batch

$$\theta_j \Leftarrow \theta_j + r \sum_{i=1}^m (y_i - \theta^T \mathbf{x}_i) x_i^{(j)}$$

- Batch updates (considering the whole training data) estimate the Loss function precisely
  - Can takes a long time if m is large
- Updates with a subset of m
  - We now have an estimate of the loss function
    - This can lead to a wrong direction, but we get faster updates
  - Called Stochastic Gradient Descent (SGD) or incremental gradient descent



# Other loss functions

- MSE

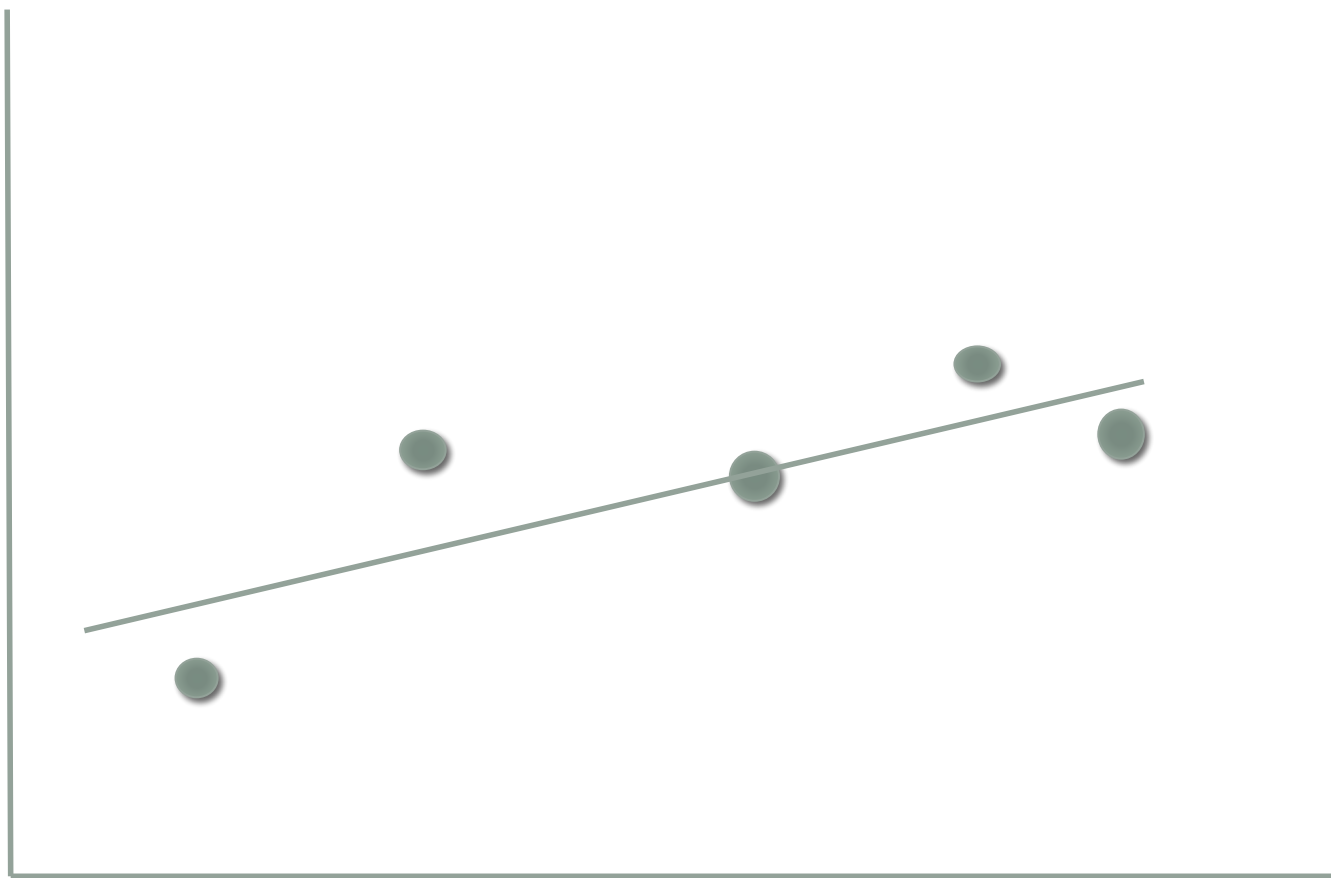
$$J(\theta) = \frac{1}{m} \sum_{i=1}^m (y_i - \theta^T \mathbf{x}_i)^2$$

- Also called L2 loss

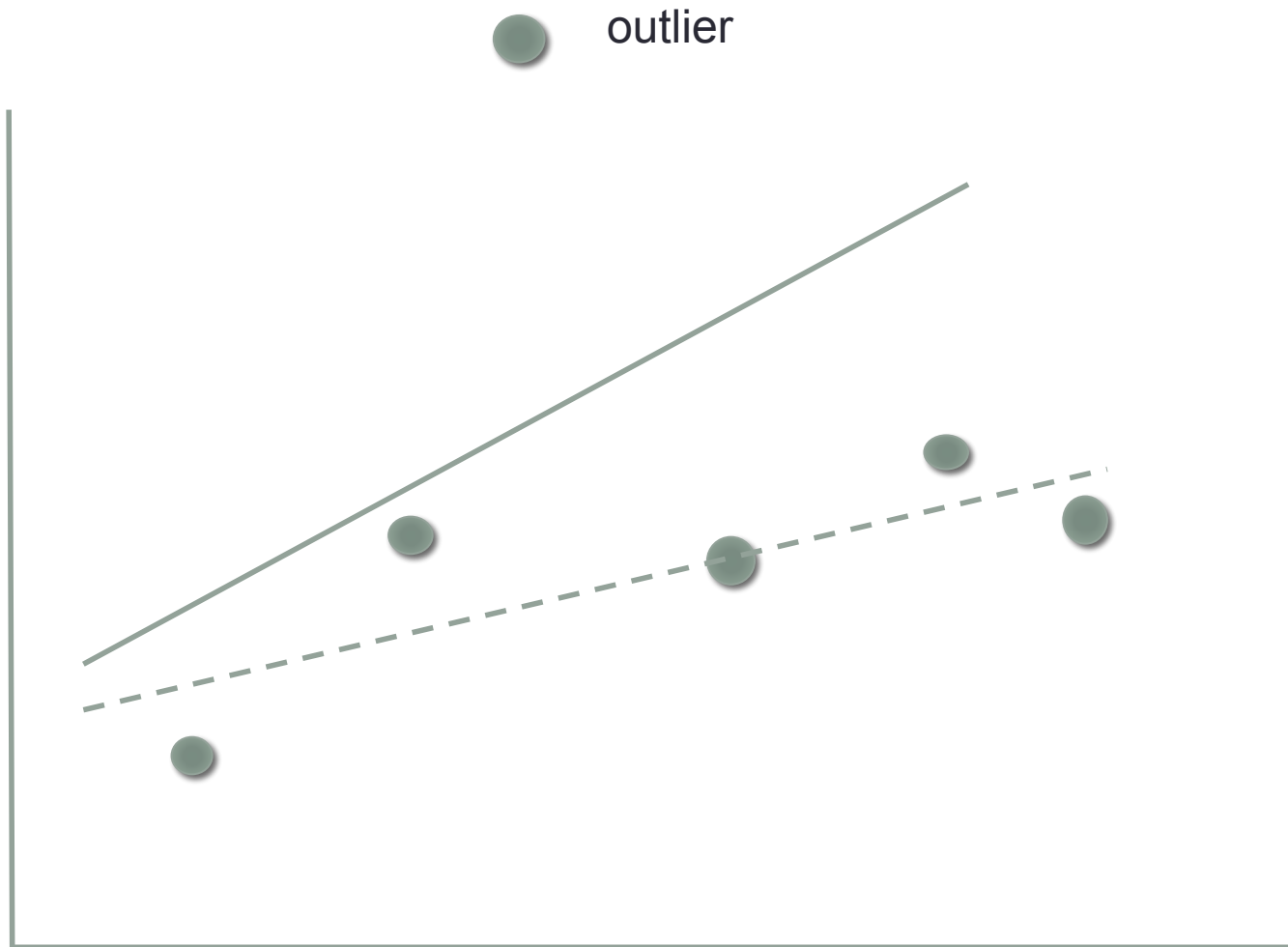
- L1 loss

$$\frac{1}{m} \sum_{i=1}^m |y_i - \theta^T \mathbf{x}_i|$$

# L2 vs L1 loss



# L2 vs L1 loss



Outlier frequently happens in the real world

# Norms (p-norm or Lp-norm)

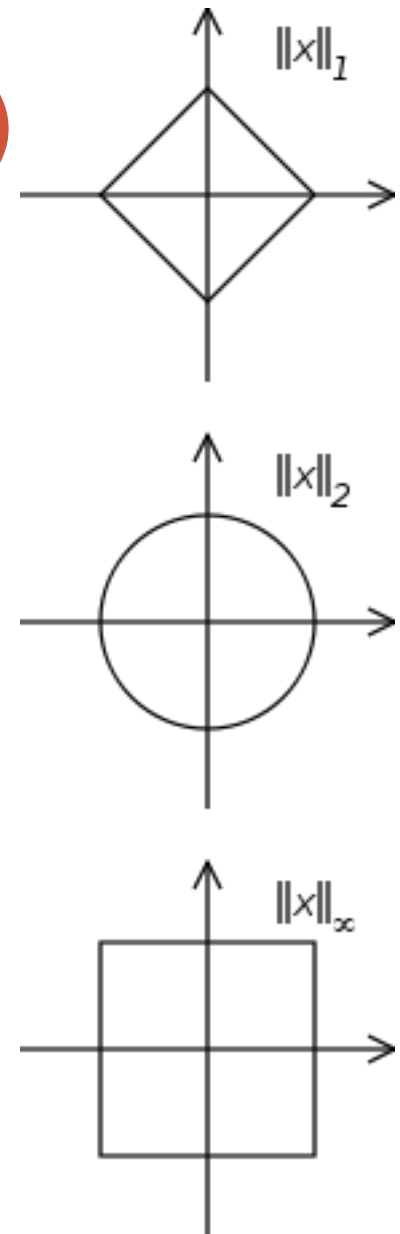
- For any real number  $p > 1$

$$\|\mathbf{x}\|_p = (|x_1|^p + |x_2|^p + \dots + |x_n|^p)^{\frac{1}{p}}$$

- For  $p = \infty$

$$\|x\|_{\infty} = \max \{|x_1|, |x_2|, \dots, |x_n|\}$$

- We'll see more of p-norms when we get to neural networks



# Minimizing a function

- You have a function
  - $y = (x - a)^2$
- You want to minimize Y with respect to x
  - $dy/dx = 2x - 2a$
  - Take the derivative and set the derivative to 0
    - (And maybe check if it's a minima, maxima or saddle point)
- We can also go with an iterative approach (Gradient descent)



# LMS regression with matrix derivatives

- First let's define what's a derivative of a matrix
- For a function  $f : \mathbb{R}^{m \times n} \mapsto \mathbb{R}$
- The derivative wrt to  $A$  is

$$\nabla_A f(A) = \begin{bmatrix} \frac{\partial f}{\partial A_{11}} & \dots & \frac{\partial f}{\partial A_{1n}} \\ \vdots & \ddots & \vdots \\ \frac{\partial f}{\partial A_{m1}} & \dots & \frac{\partial f}{\partial A_{mn}} \end{bmatrix}$$

# Example

- Suppose

$$A = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix}$$

$$f : \mathbb{R}^{m \times n} \mapsto \mathbb{R}$$

$$\nabla_A f(A) = \begin{bmatrix} \frac{\partial f}{\partial A_{11}} & \cdots & \frac{\partial f}{\partial A_{1n}} \\ \vdots & \ddots & \vdots \\ \frac{\partial f}{\partial A_{m1}} & \cdots & \frac{\partial f}{\partial A_{mn}} \end{bmatrix}$$

$$f(A) = \frac{3}{2}A_{11} + 5A_{12}^2 + A_{21}A_{22}$$

$$\nabla_A f(A) = \begin{bmatrix} \frac{3}{2} & 10A_{12} \\ A_{22} & A_{21} \end{bmatrix}$$

# Trace of a matrix

- $\text{tr}A$  is the sum of the diagonals of matrix  $A$  ( $A$  must be a square matrix)

$$\text{tr} A = \sum_i^N A_{ii}$$

- Trace of a real number? (1x1 matrix)

# Trace properties

- $\text{tr}(a) = a$
- $\text{tr}A = \text{tr}A^T$
- $\text{tr}(A+B) = \text{tr}A + \text{tr}B$
- $\text{tr}(aA) = a\text{tr}(A)$

$$\nabla_A \text{tr} AB = B^T$$

$$\nabla_{A^T} f(A) = (\nabla_A f(A))^T$$

$$\nabla_A \text{tr} ABA^T C = CAB + C^T AB^T$$

$$\nabla_{A^T} \text{tr} ABA^T C = B^T A^T C^T + BA^T C$$

# LMS regression with matrix derivatives

$$X = \begin{bmatrix} - & x_1^T & - \\ - & x_2^T & - \\ - & x_m^T & - \end{bmatrix} \quad y = \begin{bmatrix} y_1 \\ y_2 \\ y_m \end{bmatrix}$$

$$X\theta - y = \begin{bmatrix} x_1^T\theta \\ | \\ x_m^T\theta \end{bmatrix} - \begin{bmatrix} y_1 \\ | \\ y_m \end{bmatrix}$$

# LMS regression with matrix derivatives

$$X\theta - y = \begin{bmatrix} x_1^T \theta & y_1 \\ & \\ & \\ x_m^T \theta & y_m \end{bmatrix} - \begin{bmatrix} & \\ & \\ & \\ & \end{bmatrix}$$

$$\frac{1}{2}(X\theta - y)^T (X\theta - y) = \frac{1}{2} \sum_{i=1}^m (y_i - \theta^T x)^2$$

We want to minimize this term wrt to  $\theta$

# LMS regression with matrix derivatives

$$\theta = (X^T X)^{-1} X^T y$$



# Trace properties

1 •  $\text{tr}(a) = a$

2 •  $\text{tr}A = \text{tr}A^T$

3 •  $\text{tr}(A+B) = \text{tr}A + \text{tr}B$

4 •  $\text{tr}(aA) = a\text{tr}(A)$

5  $\nabla_A \text{tr}AB = B^T$

6  $\nabla_{A^T} f(A) = (\nabla_A f(A))^T$

7  $\nabla_A \text{tr}ABA^T C = CAB + C^T AB^T$

8  $\nabla_{A^T} \text{tr}ABA^T C = B^T A^T C^T + BA^T C$

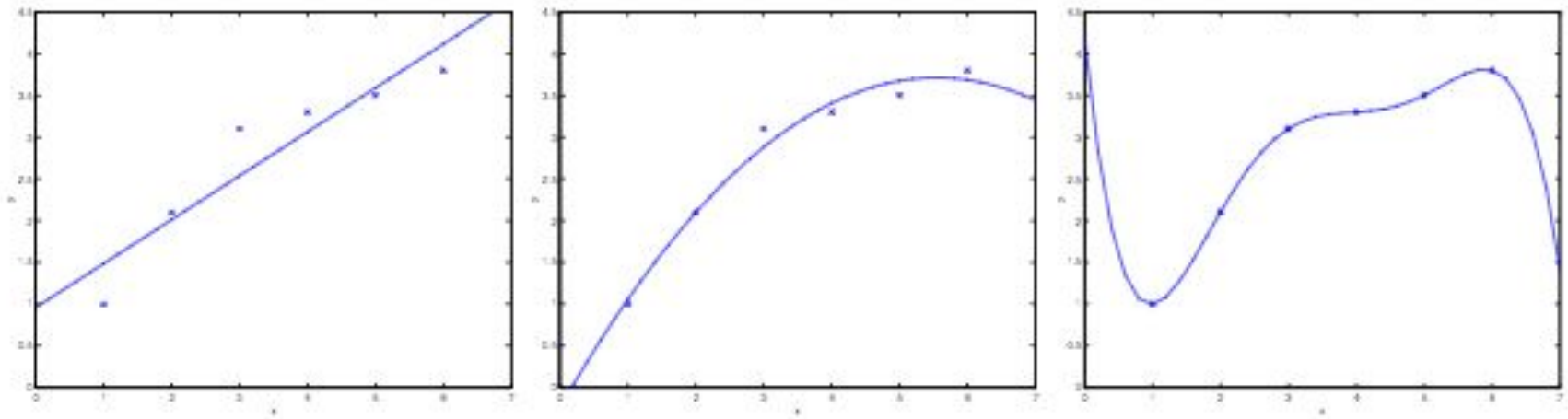
# Regression with non-linear features

- If we add extra features that are non-linear
  - For example  $x^2$

Cloth	Corn	Grass	Water	Beer	Rainfall
4	6	3	10	0	<b>76950</b>
5	1	0	0	7	<b>30234</b>
6	0	3	5	7	<b>123456</b>
5	0	3	12	0	<b>89301</b>
4	3	0	6	7	<b>?</b>

- $$h_{\theta}(\mathbf{x}_1) = \theta_0 + \theta_1 x_{1,1} + \theta_2 x_{1,2} + \theta_3 x_{1,3} + \theta_4 x_{1,4} + \theta_5 x_{1,5} + \theta_6 x_{1,1}^2 + \dots$$
- These can be considered as additional features
- We can now have a line that is non-linear

# Overfitting Underfitting



Adding more non-linear features makes the line more curvy  
(Adding more features also means more model parameters)

The curve can go directly to the outliers with enough parameters.

We call this effect **overfitting**

For the opposite case, having not enough parameters to model the data is called **underfitting**

# Predicting floods

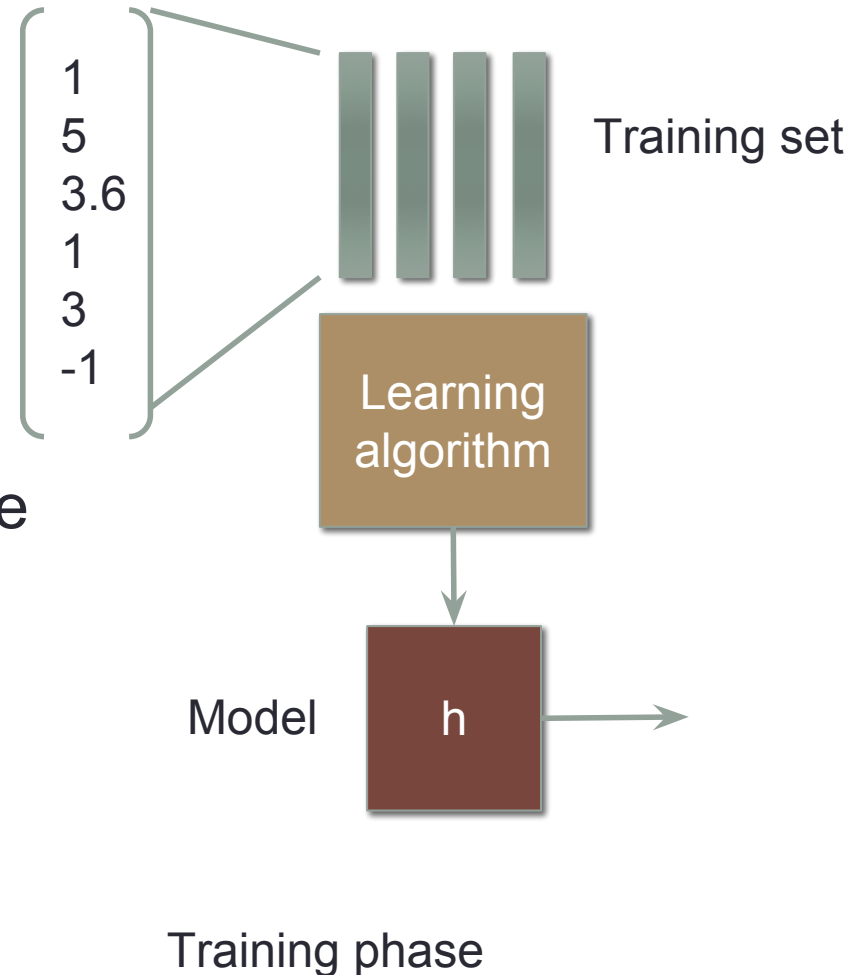
Cloth	Corn	Grass	Water	Beer	Flood?
4	6	3	10	0	<b>yes</b>
5	1	0	0	7	<b>yes</b>
6	0	3	5	7	<b>no</b>
5	0	3	12	0	<b>yes</b>
4	3	0	6	7	<b>?</b>

So far we talk about predicting an amount what if we want to do classification

Let's start with a binary choice. Flood or no flood

# Flood or no flood

- What would be the output?
- $y = 0$  if not flooded
- $y = 1$  if flooded
- Anything in between is a score for how likely it is to flood

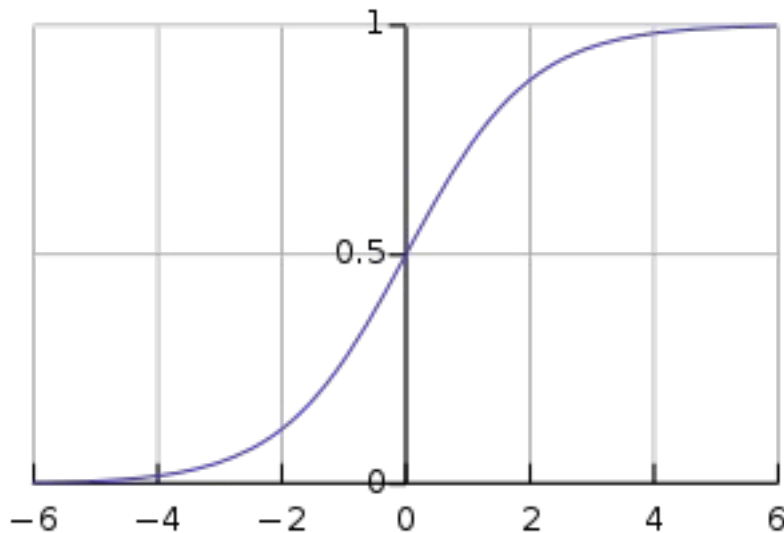


# Can we use regression?

- Yes
- $h_{\theta}(\mathbf{x}_1) = \theta_0 + \theta_1 x_{1,1} + \theta_2 x_{1,2} + \theta_3 x_{1,3} + \theta_4 x_{1,4} + \theta_5 x_{1,5}$
- But
- What does it mean when h is higher than 1?
- Can h be negative? What does it mean to have a negative flood value?

# Logistic function

- Let's force  $h$  to be between 0 and 1 somehow
- Introducing the logistic function (sigmoid function)



$$\begin{aligned} f(x) &= \frac{1}{1 + e^{-x}} \\ &= \frac{e^x}{1 + e^x} \end{aligned}$$



# Logistic Regression

- Pass  $\theta^T \mathbf{x}$  through the logistic function

$$h_{\theta}(x) = \frac{1}{1 + e^{-\theta^T x}}$$

# Loss function?

- MSE error no longer a good candidate

# Logistic Regression update rule

$$\theta_j \Leftarrow \theta_j + r \sum_{i=1}^m (y_i - h_{\theta}(x_i)) x_i^{(j)}$$

Update rule for linear regression

$$\theta_j \Leftarrow \theta_j + r \sum_{i=1}^m (y_i - \theta^T \mathbf{x}_i) x_i^{(j)}$$



# Homework

Some matrix, some K-mean, and some regression



# Office hours

- Wednesdays 16.30-18.00 starting from Aug 22<sup>nd</sup>
- 29th moved to 31st
- Location Sky Cafe

