# Machine Learning
# Apprentissage Machine
# INFO415

Dr. Amani RAAD

Amani.raad@ul.edu.lb

2023

# Chapter 1

General Concepts

# Introduction

Arithmetics, algorithms

An abacus

Logic and logical reasoning

Aristotle

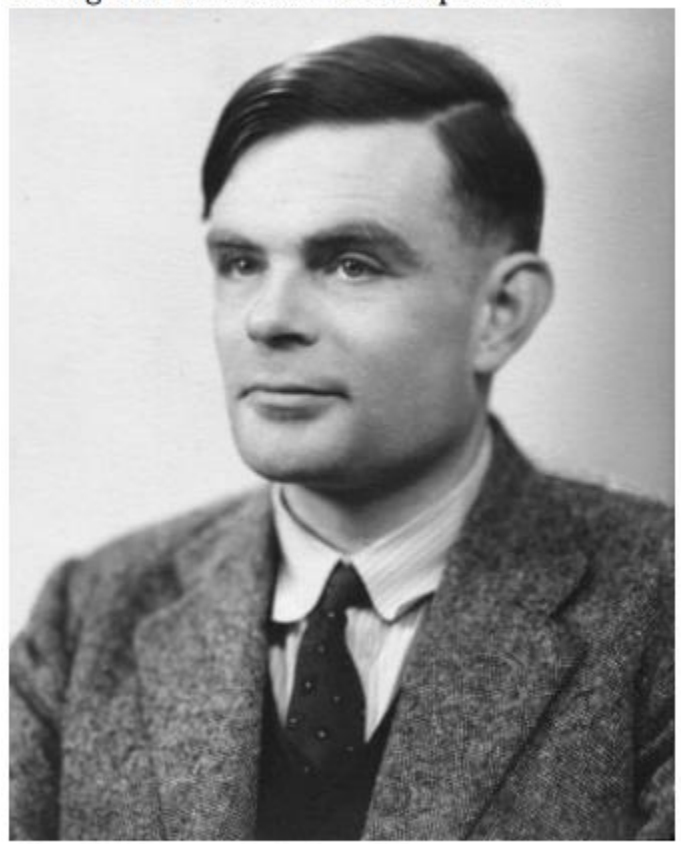
Automata

A writing automaton

Programmable machines

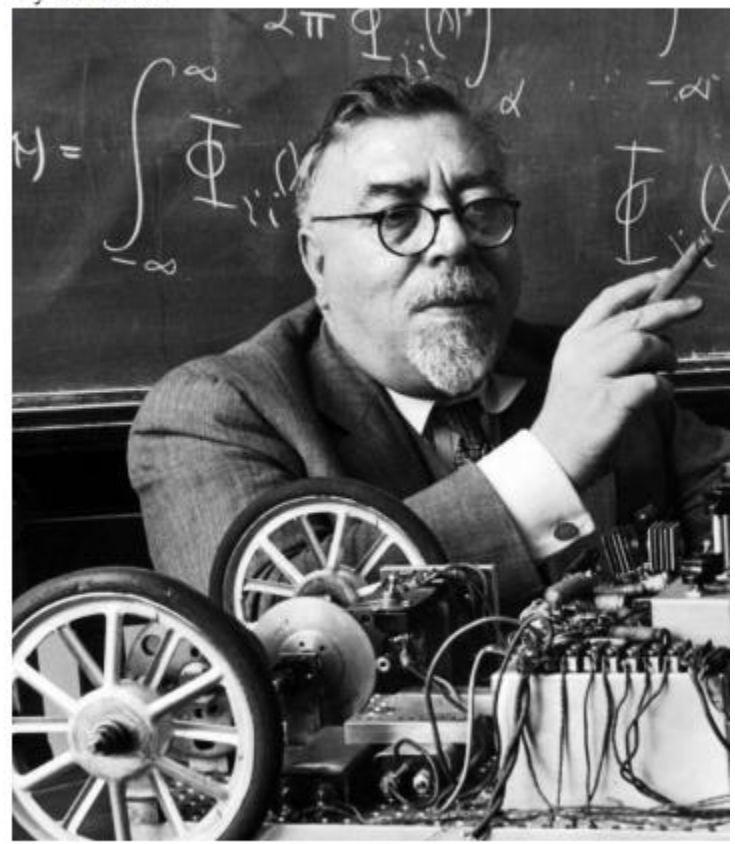
The Jaquard programmable loom

Turing and the science of computation



Alan M. Turing (1912 - 1954)

Cybernetics



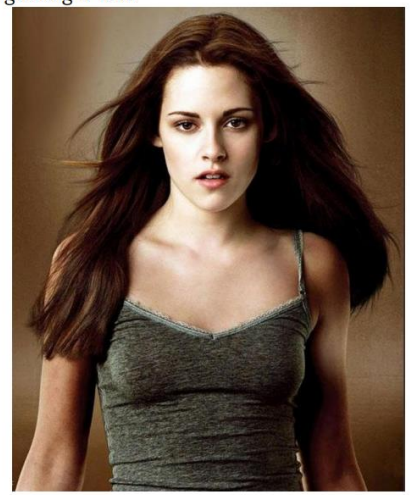
Norbert Wiener (1894 - 1964)

# For these problems we do have algorithms

- Sorting → insert sort, bubble sort, Shell sort, radix sort, heapsort, bogosort. . .
- Spectral analysis of periodic signals → FFT
- Database filtering → SQL SELECT

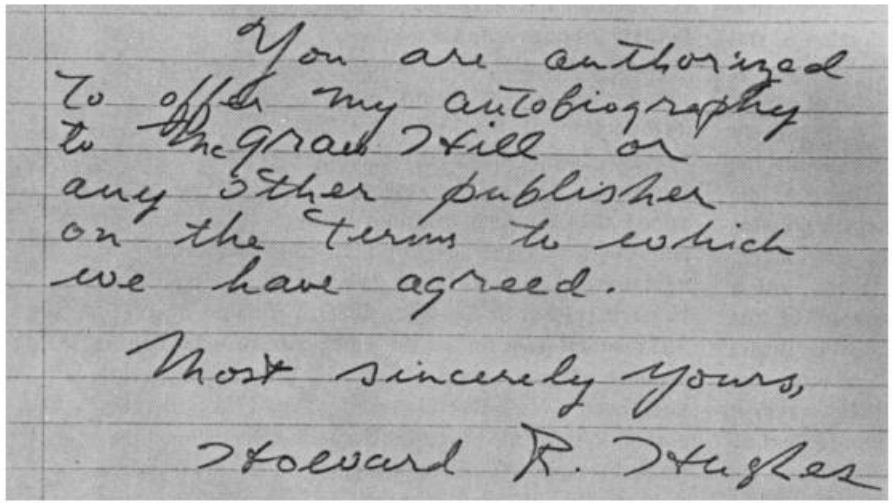
# We have no algorithms for

Recognizing a face



# We have no algorithms for

Distinguishing between genuine works by a given author and fakes



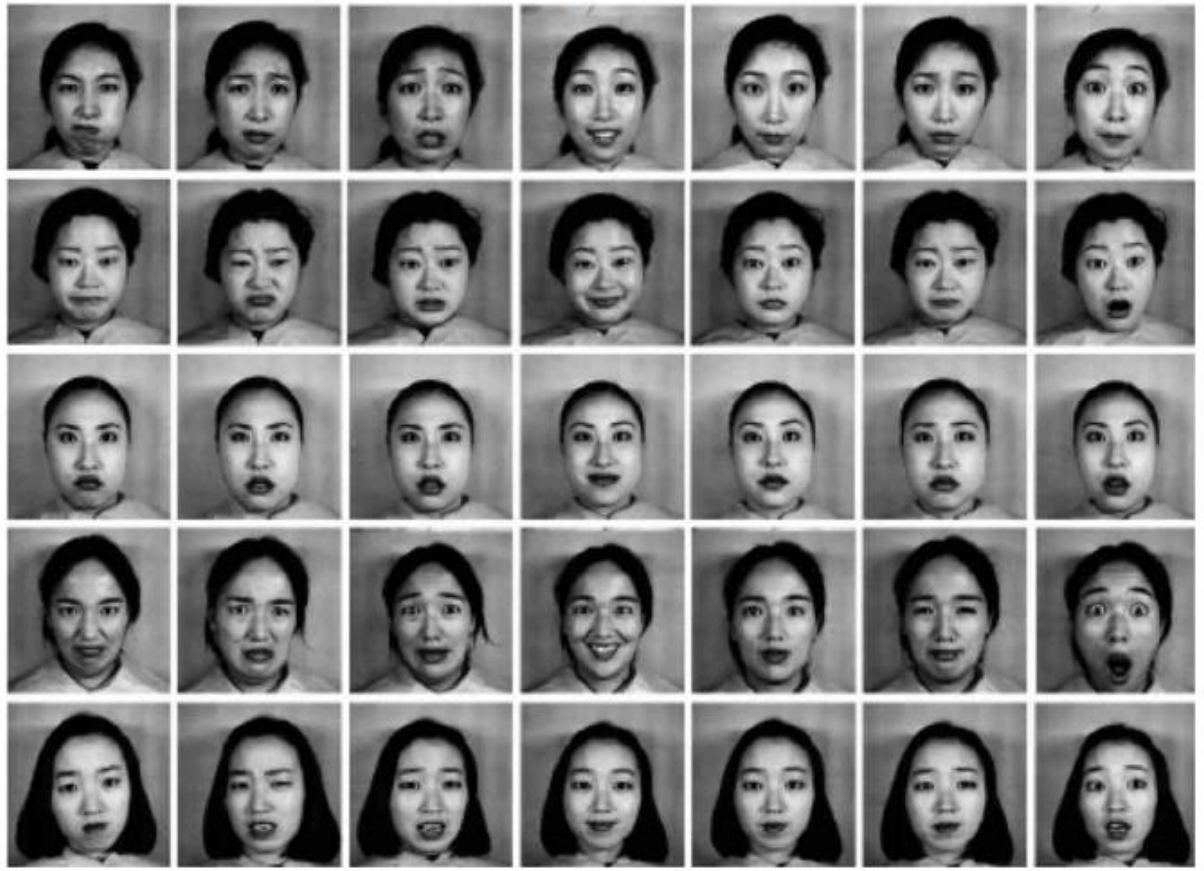
You are authorized to offer my autobiography to McGraw Hill or any other publisher on the terms to which we have agreed.

Most sincerely yours,

Howard R. Hughes

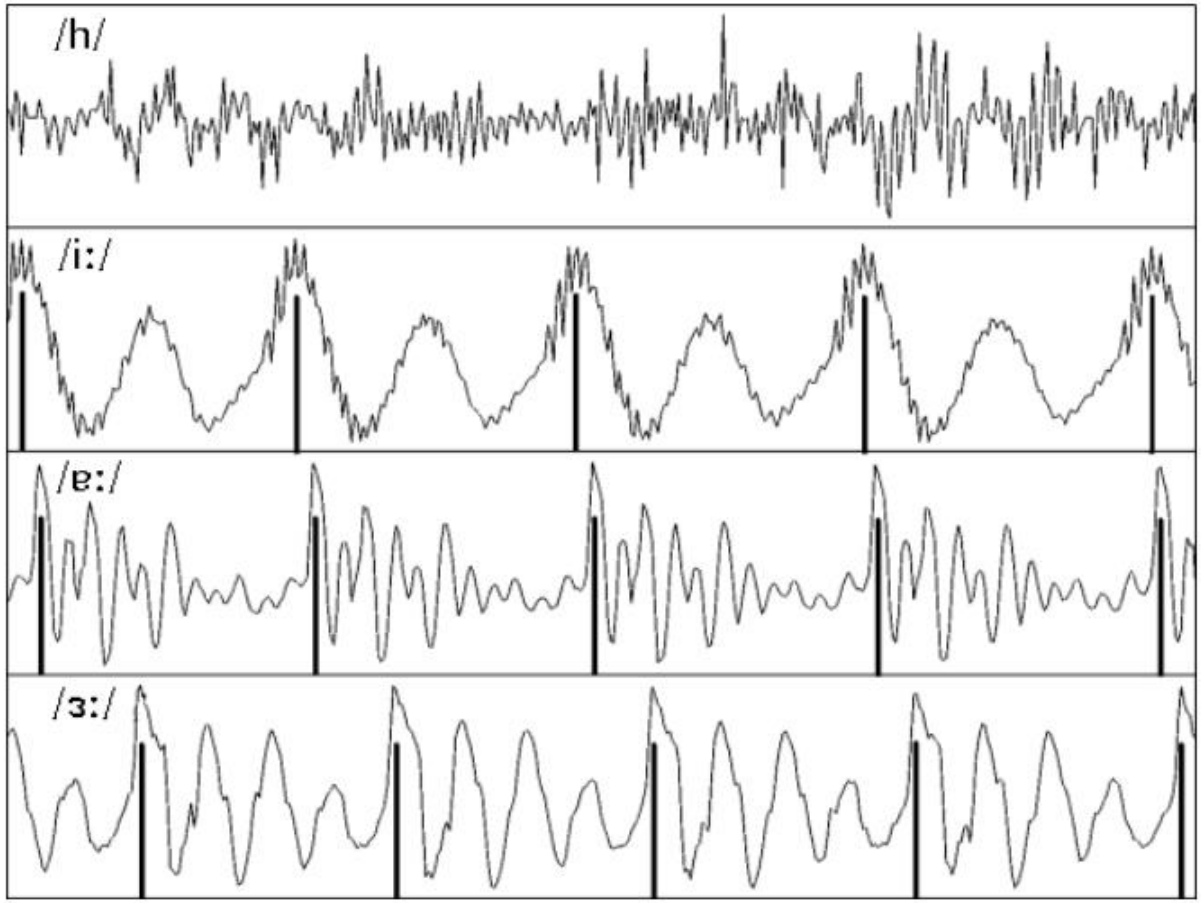
# We have no algorithms for

Recognizing emotions from gestures or facial expressions



# We have no algorithms for

Understanding speech



# We have no algorithms for machine translation

# We have no algorithms for


Spam filtering

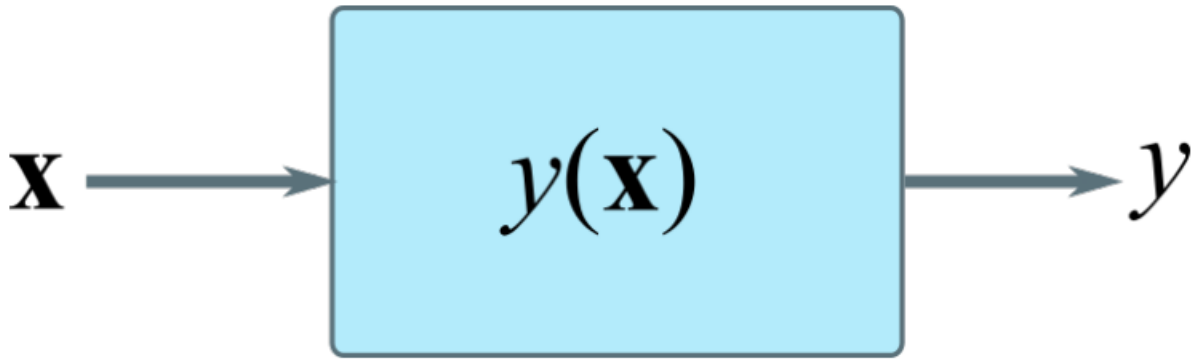
- Many interesting problems are too complex to admit an algorithmic solution or even a complete description
- For these problems, only data are available
- Nowadays we have LOTS OF DATA FROM LOTS OF SOURCES

- Machine learning is about using data to solve problem

- Instead of writing a program by hand for each specific task, we collect lots of examples that specify the correct output for a given input.
- A machine learning algorithm then takes these examples and produces a program that does the job.
- Massive amounts of computation are now cheaper than paying someone to write a task-specific program.

# General structure of a learning machine

A rule that receives an observation $\mathbf{x}$
and emits an output $y = y(\mathbf{x})$

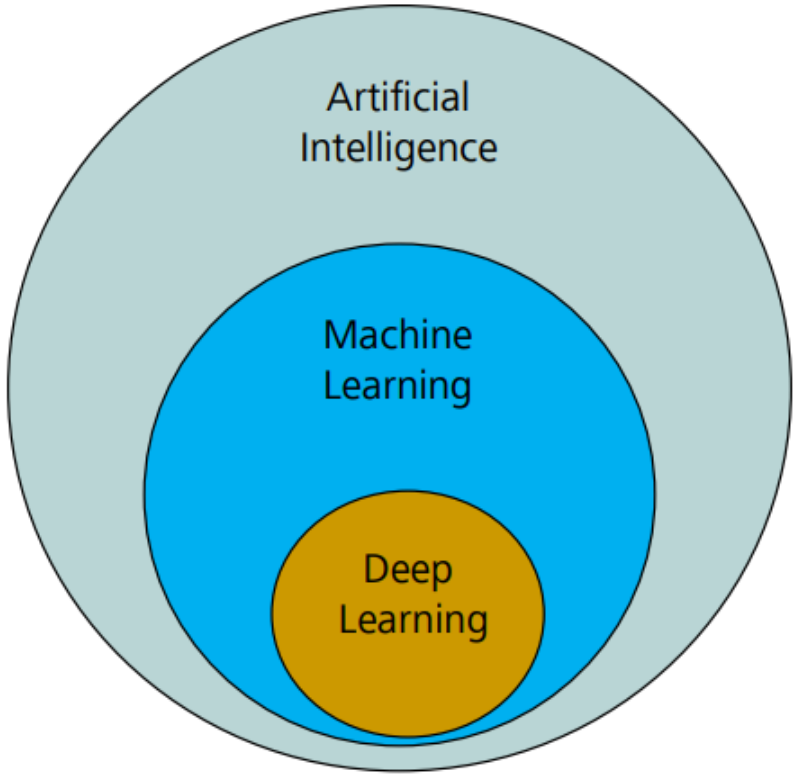



- The rule can be implemented as a software, as a procedure, as a hardware machine, as an embedded computer...
- The rule normally has a predefined structure, but depends on a set of parameters.
- Learning consists in fitting these parameters to a specific problem.

# Machine Learning

- Machine Learning is a subset of  Artificial Intelligence et is coined by Arthur Samuel in 1959, an American pioneer in the field of computer gaming and artificial intelligence
- It stated that
- **“it gives computers the ability to learn without being explicitly programmed”.**

# Relationship betweeen AI, Machine Learning and Deep Learning



- **Learning machine** – also “learner”.
  Not necessarily a real machine! Maybe a program

- **Task** – a problem to be solved.
  We don’t have a description of the problem, but **data**

- **Learning** – adjusting quantities “inside the machine”
  (e.g., algorithm parameters) to do a certain task

# Tasks

- Mapping a stimulus to a response:
  **supervised learning**

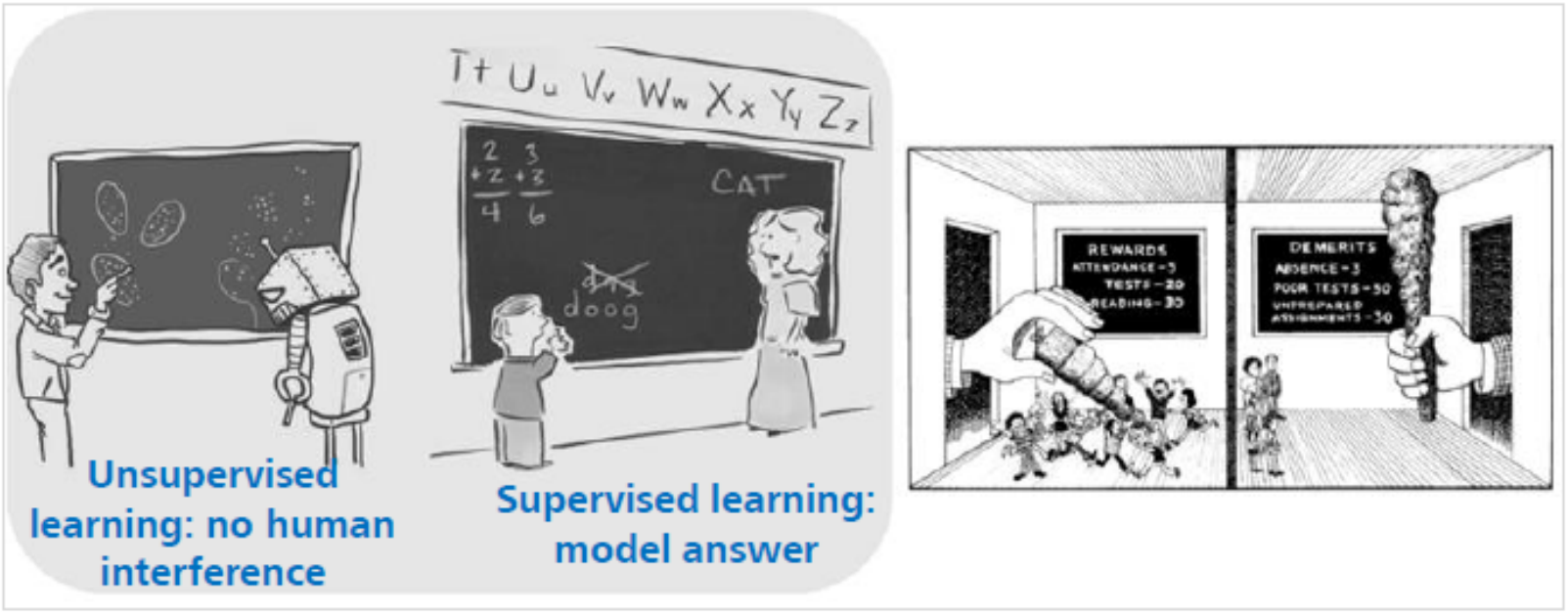
- Describing the data:
  **unsupervised learning**

# Tasks

- In **supervised tasks**
  the data contain both input patterns and output values

- In **unsupervised tasks**
  the data contain just input patterns

# Types of Machine learning




Traditional machine learning

Reinforcement learning: reward and punishment

# Tasks

- Mapping a stimulus to a response (from input to output):
  - Classification
  - Regression

- Describing the data (from input to a more compact representation of the input itself):
  - Clustering
  - Mapping in lower dimensionality

# Supervised learning

- This approach is indeed similar to human learning under the supervision of a teacher. The teacher provides good examples for the student to memorize, and the student then derives general rules from these specific examples.
- In Supervised learning, an AI system is presented with data which is labeled, which means that each data tagged with the correct label.
- Categories: classification, regression

# Classification

- When inputs are divided into two or more classes, and the learner must produce a model that assigns unseen inputs to one or more of these classes. This is typically tackled in a supervised way.
- Example: Spam filtering is an example of classification, where the inputs are email (or other) messages and the classes are “spam” and “not spam”.
- Example: Medical and industrial diagnosis : predict heart disease or predict the failure of a machine in industry in the case of labelled data.

# Example

Iris (flower) recognition

- The Iris dataset has been in use since 1936
- Collected by botanist Edgar Anderson in 1935
- Used by statistician Sir Ronald A. Fisher in 1936

Sources:

*Edgar Anderson (1935). "The irises of the Gaspe Peninsula".* Bulletin of the American Iris Society *59: 25.*

*Fisher, R.A. (1936). "The Use of Multiple Measurements in Taxonomic Problems".* Annals of Eugenics *7: 179-188.*

Download it from the University of California - Irvine repository at:
`http://archive.ics.uci.edu/ml/datasets/Iris`

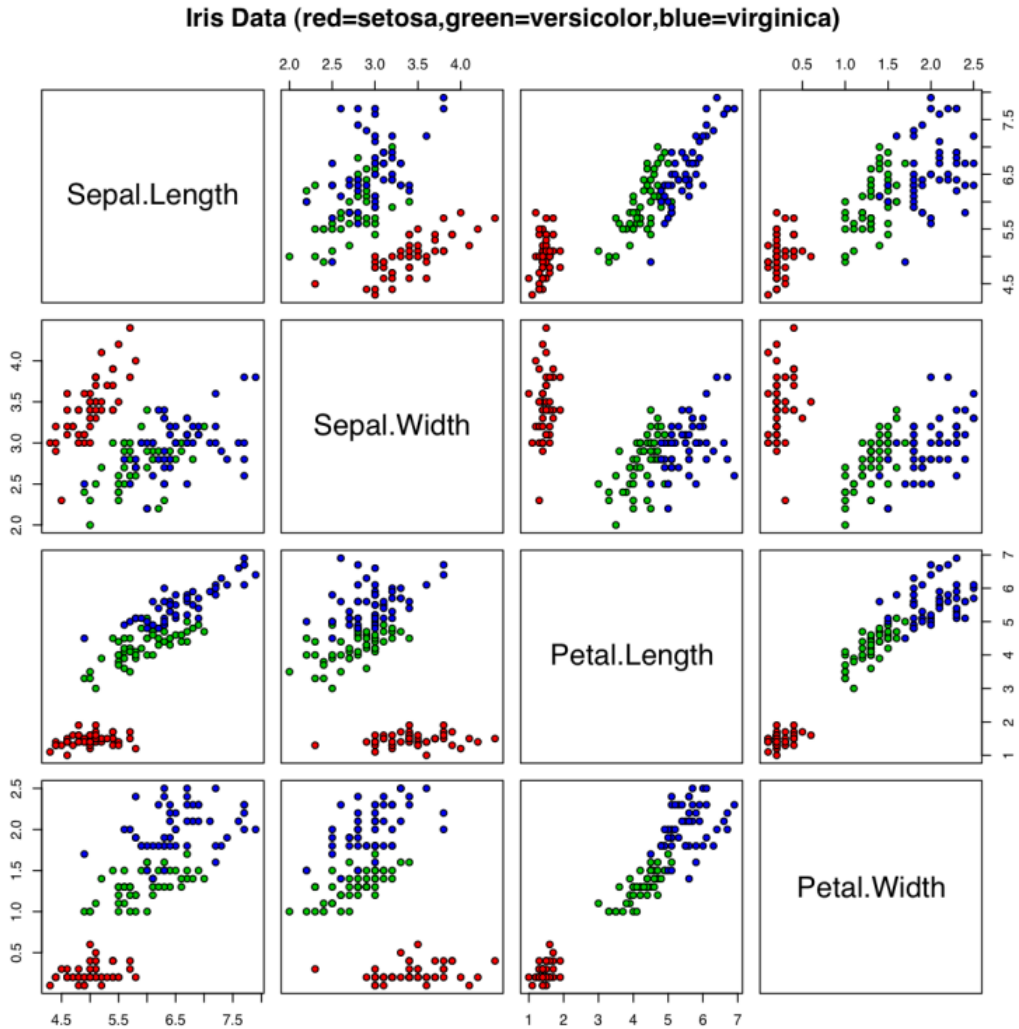
*Iris Setosa*

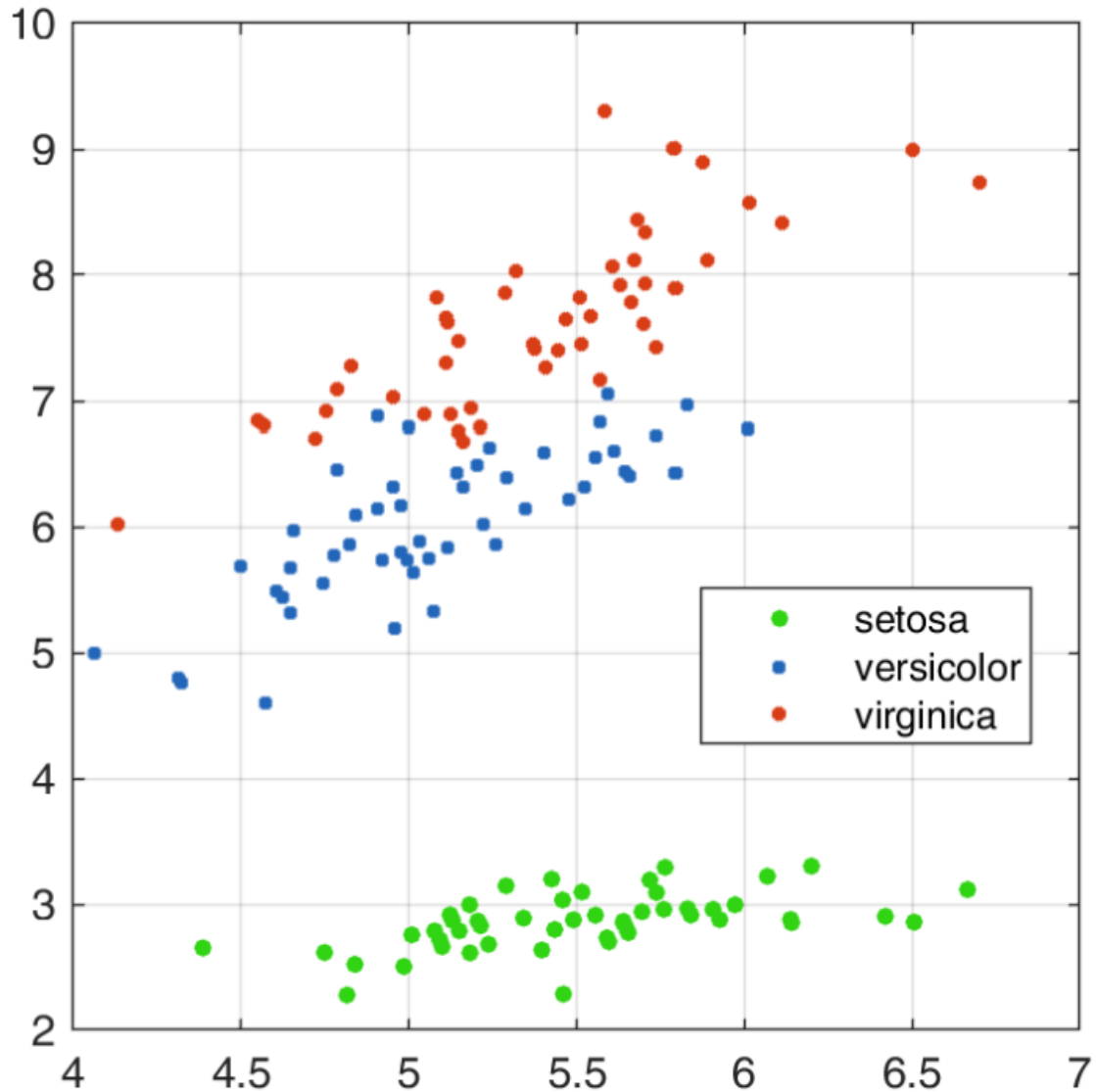
*Iris Virginica*

*Iris Versicolor*

Table I

| *Iris setosa* | | | | *Iris versicolor* | | | | *Iris virginica* | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Sepal length | Sepal width | Petal length | Petal width | Sepal length | Sepal width | Petal length | Petal width | Sepal length | Sepal width | Petal length | Petal width |
| 5·1 | 3·5 | 1·4 | 0·2 | 7·0 | 3·2 | 4·7 | 1·4 | 6·3 | 3·3 | 6·0 | 2·5 |
| 4·9 | 3·0 | 1·4 | 0·2 | 6·4 | 3·2 | 4·5 | 1·5 | 5·8 | 2·7 | 5·1 | 1·9 |
| 4·7 | 3·2 | 1·3 | 0·2 | 6·9 | 3·1 | 4·9 | 1·5 | 7·1 | 3·0 | 5·9 | 2·1 |
| 4·6 | 3·1 | 1·5 | 0·2 | 5·5 | 2·3 | 4·0 | 1·3 | 6·3 | 2·9 | 5·6 | 1·8 |
| 5·0 | 3·6 | 1·4 | 0·2 | 6·5 | 2·8 | 4·6 | 1·5 | 6·5 | 3·0 | 5·8 | 2·2 |
| 5·4 | 3·9 | 1·7 | 0·4 | 5·7 | 2·8 | 4·5 | 1·3 | 7·6 | 3·0 | 6·6 | 2·1 |
| 4·6 | 3·4 | 1·4 | 0·3 | 6·3 | 3·3 | 4·7 | 1·6 | 4·9 | 2·5 | 4·5 | 1·7 |
| 5·0 | 3·4 | 1·5 | 0·2 | 4·9 | 2·4 | 3·3 | 1·0 | 7·3 | 2·9 | 6·3 | 1·8 |
| 4·4 | 2·9 | 1·4 | 0·2 | 6·6 | 2·9 | 4·6 | 1·3 | 6·7 | 2·5 | 5·8 | 1·8 |
| 4·9 | 3·1 | 1·5 | 0·1 | 5·2 | 2·7 | 3·9 | 1·4 | 7·2 | 3·6 | 6·1 | 2·5 |
| 5·4 | 3·7 | 1·5 | 0·2 | 5·0 | 2·0 | 3·5 | 1·0 | 6·5 | 3·2 | 5·1 | 2·0 |
| 4·8 | 3·4 | 1·6 | 0·2 | 5·9 | 3·0 | 4·2 | 1·5 | 6·4 | 2·7 | 5·3 | 1·9 |
| 4·8 | 3·0 | 1·4 | 0·1 | 6·0 | 2·2 | 4·0 | 1·0 | 6·8 | 3·0 | 5·5 | 2·1 |
| 4·3 | 3·0 | 1·1 | 0·1 | 6·1 | 2·9 | 4·7 | 1·4 | 5·7 | 2·5 | 5·0 | 2·0 |
| 5·8 | 4·0 | 1·2 | 0·2 | 5·6 | 2·9 | 3·6 | 1·3 | 5·8 | 2·8 | 5·1 | 2·4 |
| 5·7 | 4·4 | 1·5 | 0·4 | 6·7 | 3·1 | 4·4 | 1·4 | 6·4 | 3·2 | 5·3 | 2·3 |
| 5·4 | 3·9 | 1·3 | 0·4 | 5·6 | 3·0 | 4·5 | 1·5 | 6·5 | 3·0 | 5·5 | 1·8 |
| 5·1 | 3·5 | 1·4 | 0·3 | 5·8 | 2·7 | 4·1 | 1·0 | 7·7 | 3·8 | 6·7 | 2·2 |
| 5·7 | 3·8 | 1·7 | 0·3 | 6·2 | 2·2 | 4·5 | 1·5 | 7·7 | 2·6 | 6·9 | 2·3 |
| 5·1 | 3·8 | 1·5 | 0·3 | 5·6 | 2·5 | 3·9 | 1·1 | 6·0 | 2·2 | 5·0 | 1·5 |
| 5·4 | 3·4 | 1·7 | 0·2 | 5·9 | 3·2 | 4·8 | 1·8 | 6·9 | 3·2 | 5·7 | 2·3 |


Iris Data (red=setosa,green=versicolor,blue=virginica)
Sepal.Length
Sepal.Width
Petal.Length
Petal.Width


10
9
8
7
6
5
4
3
2
4
4.5
5
5.5
6
6.5
7
setosa
versicolor
virginica

# Example 2- MNIST DATA

Recognizing handwritten digits

Images of size 28x28 pixels, grayscale

# Example 3

- A wall-following robot has to make decisions as to the direction to take, depending on a circular array of ultrasound sensors
- The robot has 24 such sensors evenly spread over 360 degrees.
- Possible directions are:

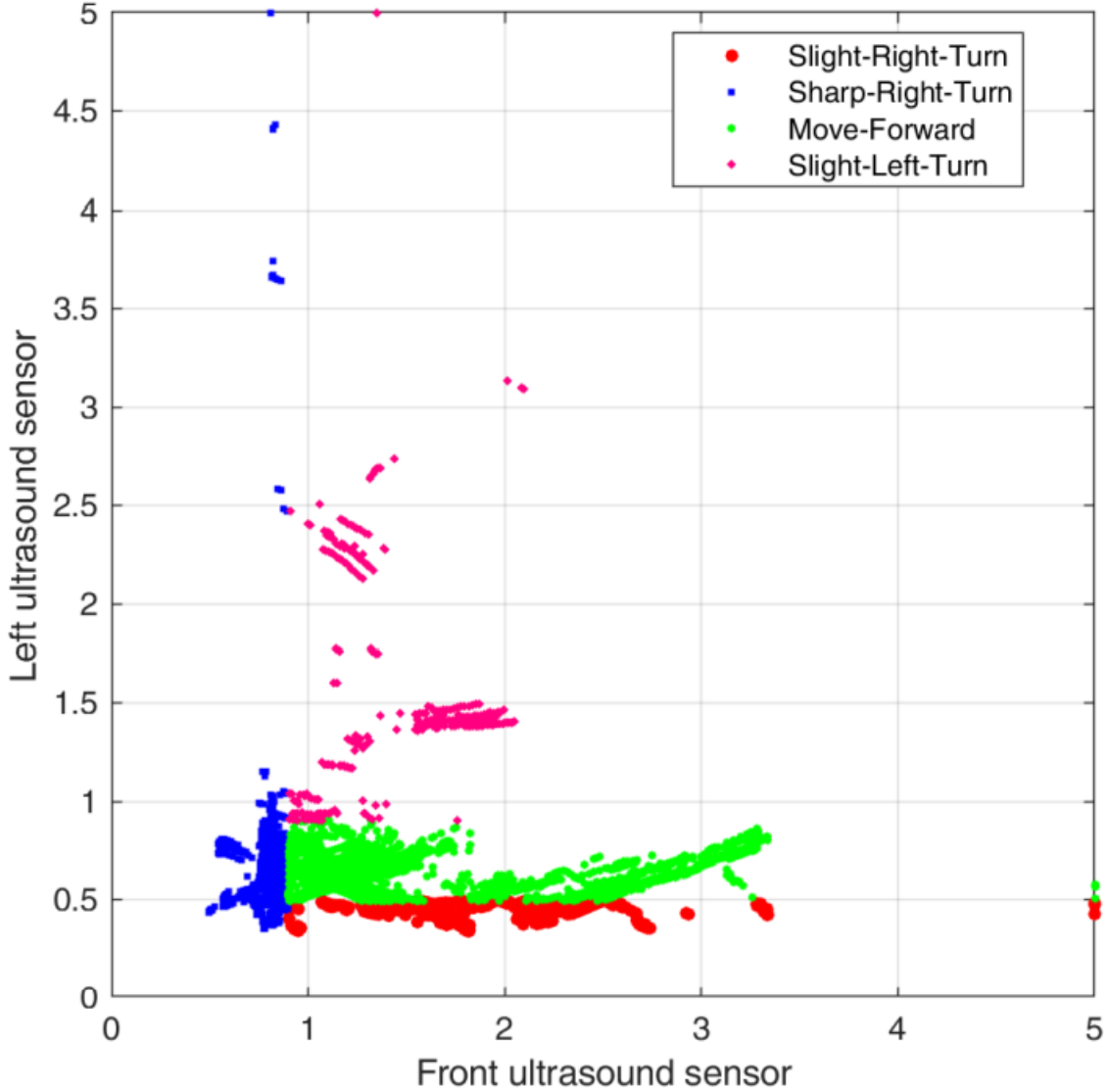
| Sharp-Left-Turn | Slight-Left-Turn | Move-forward | Slight-Right-Turn | Sharp-Right-Turn |
|---|---|---|---|---|

The scitos G5 robot is a multipurpose, modular platform for robotic research and development.

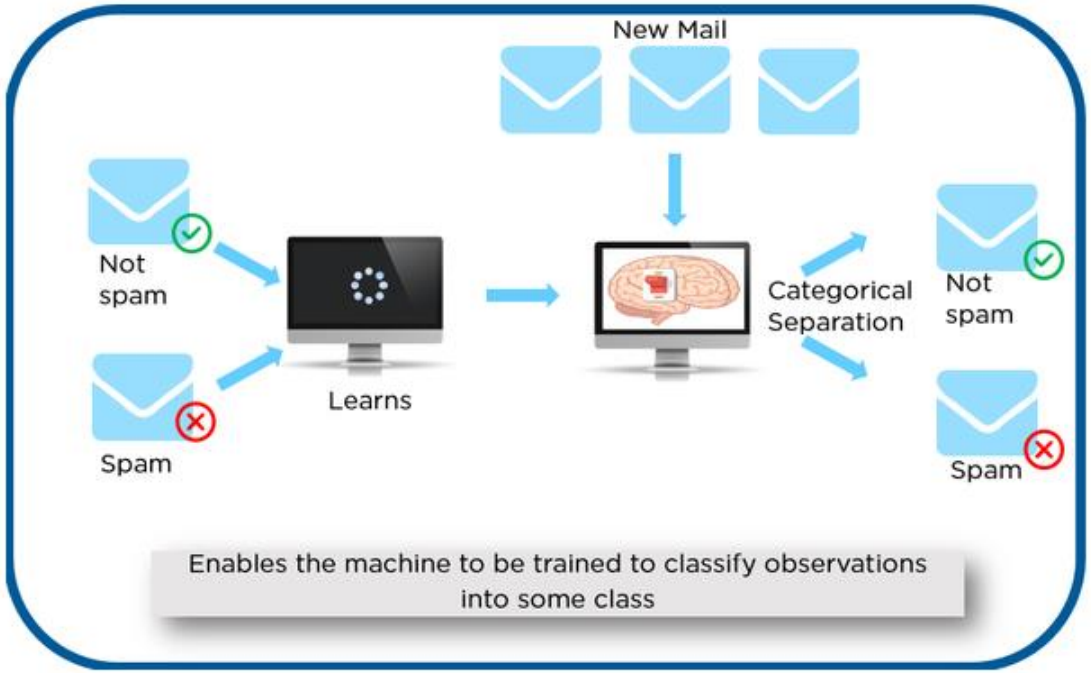

https://www.metralabs.com/en/mobile-robot-scitos-g5/

The ultrasonic sensor's output is available as a voltage in the range 0 . . . 5V

Minimum readings from two groups of sensors, on the forward and on the left.
Colors correspond to directions to take.



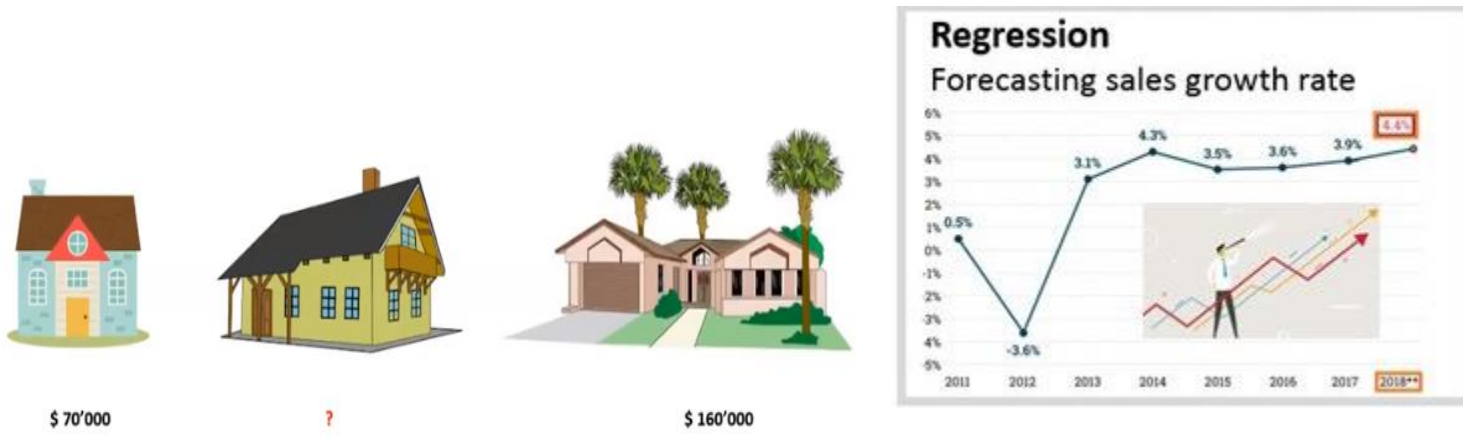
# Example 4




Example of Supervised Learning

# Regression

- **Regression :** Which is also a supervised problem, A case when the outputs are continuous rather than discrete.
- Example: to predict the price of a house

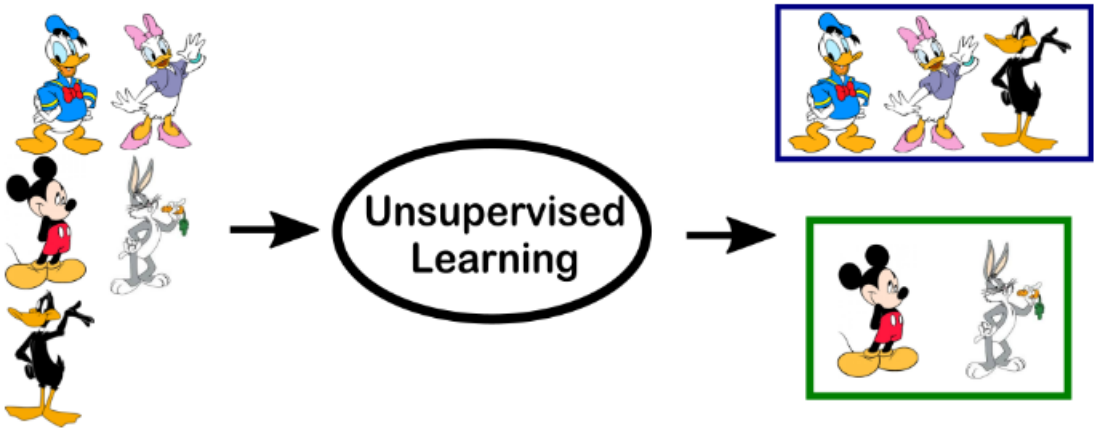


# Unsupervised learning

- In unsupervised learning, an AI system is presented with unlabeled, uncategorized data and the system's algorithms act on the data without prior training.

- Category: Clustering, Association

# Unsupervised learning




Example of Unsupervised Learning

# Clustering


- When a set of inputs is to be divided into groups. Unlike in classification, the groups are not known beforehand, making this typically an unsupervised task.
- **Example : YouTube**:  uses our search history or watched history and suggests videos we might like. Feature data set for Facebook contains is people we follow, pages we follow, comments we input, photos or videos we like, pictures or photos we tag at.
- **Example: Banking:** With the help of clustering, insurance companies can find fraud, acknowledge customers about it and understand policies brought by the customer.
- **Example: Google** is one of the search engine people uses. Let’s take an example when we search for some information like pet store in the area, Google will provide us different options.

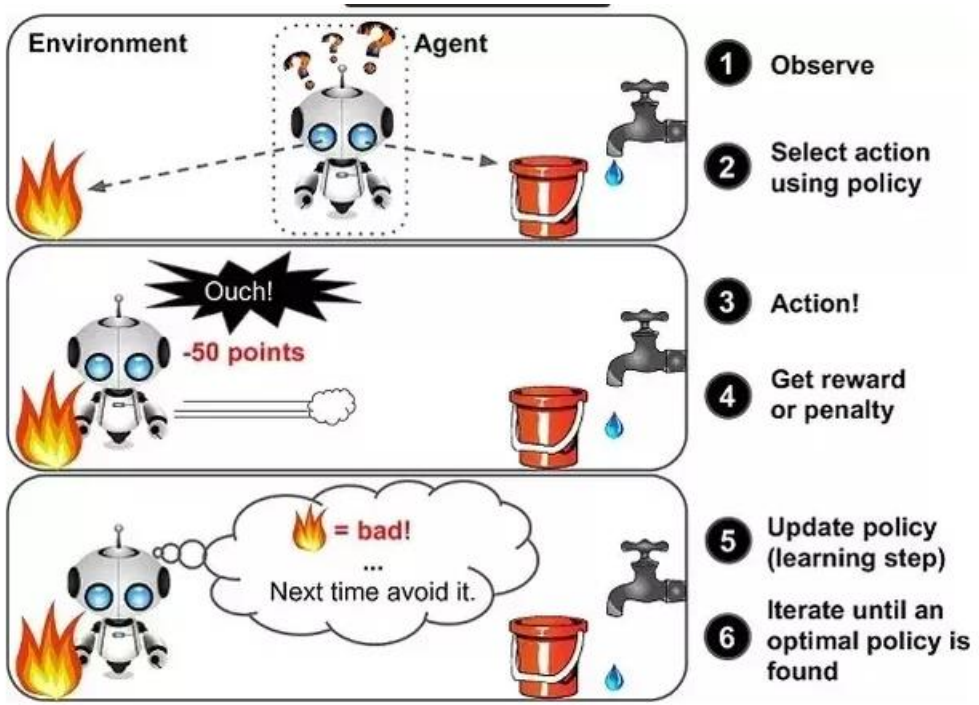
# Reinforcement learning

- A reinforcement learning algorithm, or agent, learns by interacting with its environment. The agent receives rewards by performing correctly and penalties for performing incorrectly. The agent learns without intervention from a human by maximizing its reward and minimizing its penalty. It is a type of dynamic programming that trains algorithms using a system of reward and punishment.

- In the human world, it is just like learning by trial and error.
  Errors help you learn because they have a penalty added (cost, loss of time, regret, pain, and so on), teaching you that a certain course of action is less likely to succeed than others.

- An interesting example of reinforcement learning occurs when computers learn to play video games by themselves.

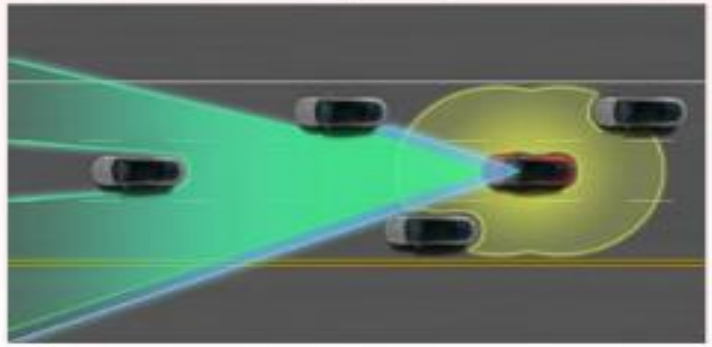
# Reinforcement learning




Example of Reinforcement Learning

# Self Driving cars

- Autonomous cars(self-driving car)

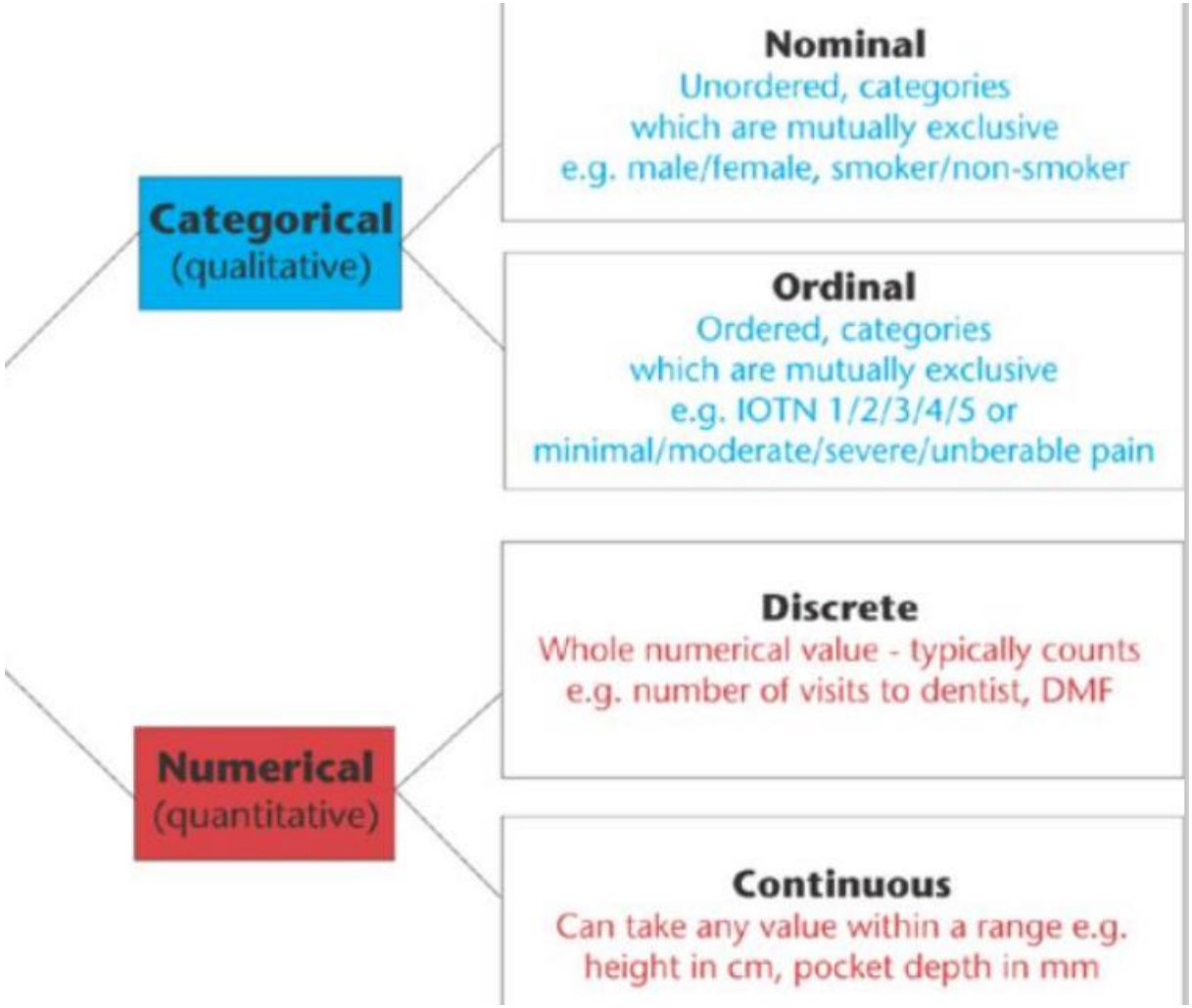



- Google Self-driving cars
- Tesla Model S – Autopilot mode

# Gaming bots

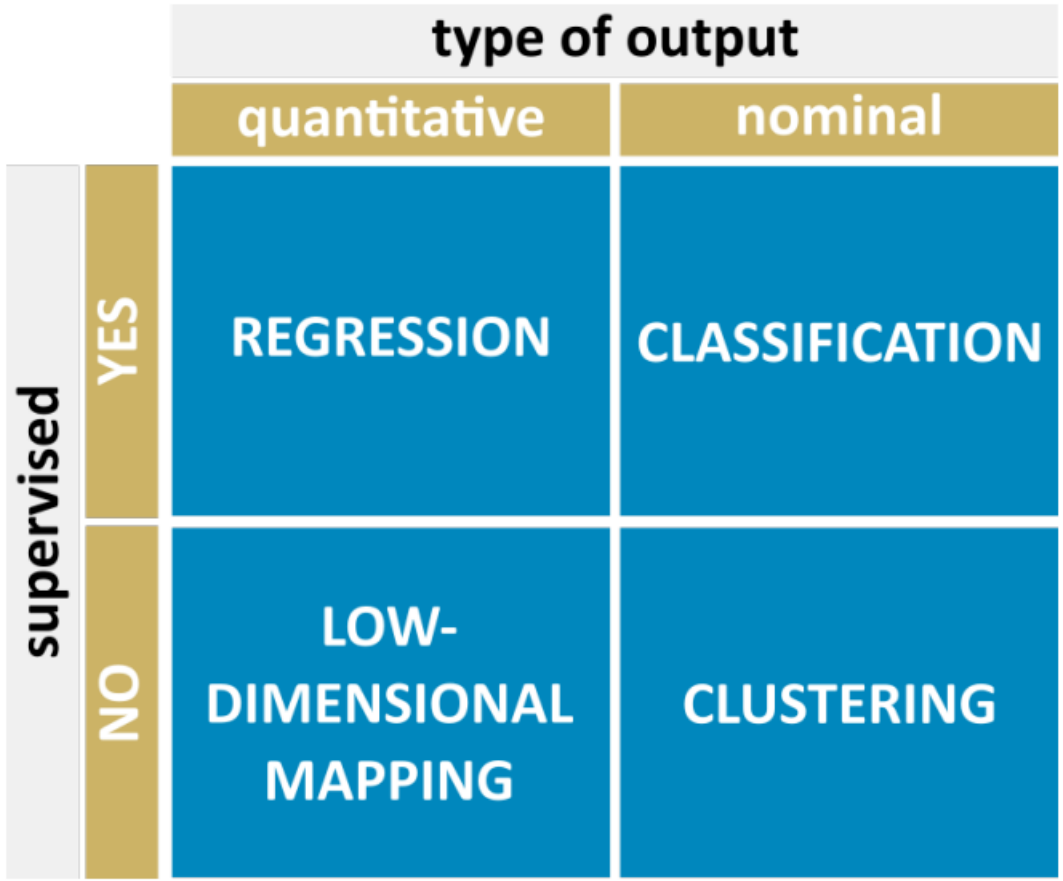
- **Deep blue** : Chess playing computer developed by IBM. It the masterpiece of a truly intelligent player. It **won against** Garry Kasparov(Chess grandmaster) in a match becoming the first computer system to defeat a reigning world champion in a match under standard chess tournament time controls.

# Types of variables



examples of categorical information:

- colour $= \{\text{red}, \text{green}, \text{blue}, \text{cyan}, \text{magenta}, \text{yellow}, \text{black}\}$
- name $= \{\text{Socrates}, \text{Plato}\}$
- truth value $= \{\text{true}, \text{false}\}$


type of output
quantitative
nominal
supervised
YES
NO
REGRESSION
CLASSIFICATION
LOW-DIMENSIONAL MAPPING
CLUSTERING

# A generic Python machine learning library

Scikit-learn
`https://scikit-learn.org/`

- A simple-to-use, yet fairly complete, machine learning library
- Built upon other standard libraries:
  - numpy (Matlab-like vectors, arrays, linear algebra)
  - scipy (computational methods)
  - matplotlib (Matlab-like plotting)

# A python data analysis library

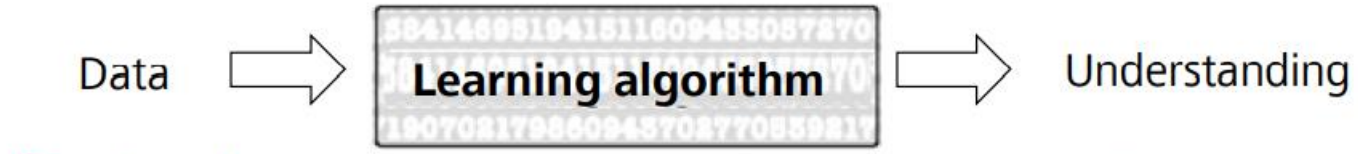
**Pandas**, a Python data analysis library

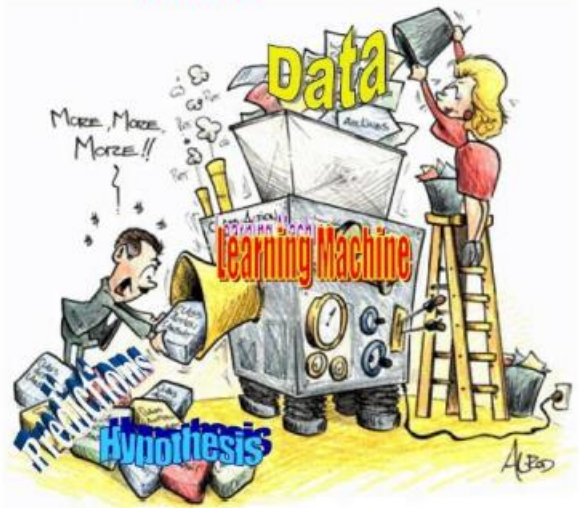
```
import pandas as pd
```

- Suitable for tabular data
- `Series` and `DataFrame` data structures
- Heterogeneous columns, missing data
- Intrinsic data alignment (data with their labels)
- Works with rows, columns, slices, boolean indexing
- Data types interoperable with numpy

# Learning algorithms

- Machine learning (including deep learning) is a study of learning algorithms. A computer program is said to learn from experience $E$ with respect to some class of tasks $T$ and performance measure $P$ if its performance at tasks in $T$, as measured by $P$, improves with experience $E$.



# Basic Terms and Concepts

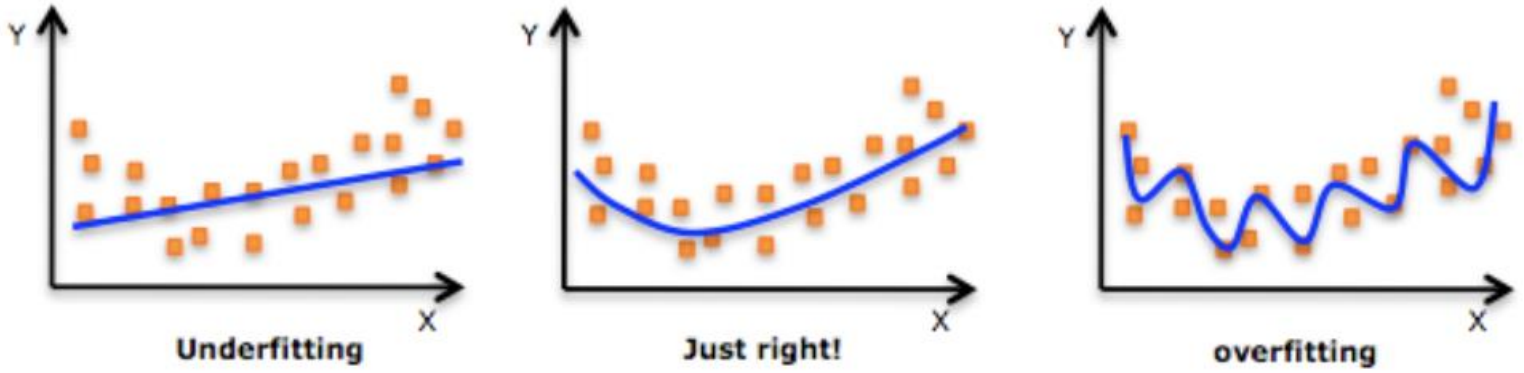
- **Dataset:** refers to a set of data used in machine learning tasks. Each piece of data is called a sample. The event or attribute that reflects the performance or nature of a sample in a certain aspect is called a **feature**.
- **Training set:** refers to a dataset used in the training process, where each sample is referred to as a training sample. The process of learning a model from data is called **learning (training)**.
- **Test set:** Test refers to the process of using the learnt model for prediction. The dataset used is called a test set, and each sample is called a test sample.

# Basic Terms and Concepts

- **Generalization capability:** The goal of machine learning is that the learnt model should perform well on new samples, not just those on which the model has been trained. The ability to perform well on new samples is called generalization capability.
- **Error:** refers to the difference between the sample result predicted by the learnt model and the actual sample result.
  - Training error: error of the model on the training set
  - Generalization error: error on the new sample. Obviously, we prefer a model with a smaller generalization error.
- **Underfitting:** occurs when the training error is too large.
- **Overfitting:** occurs when the training error of the learnt model is small but the generalization error is large (weak generalization capability).

# Basic Terms and Concepts

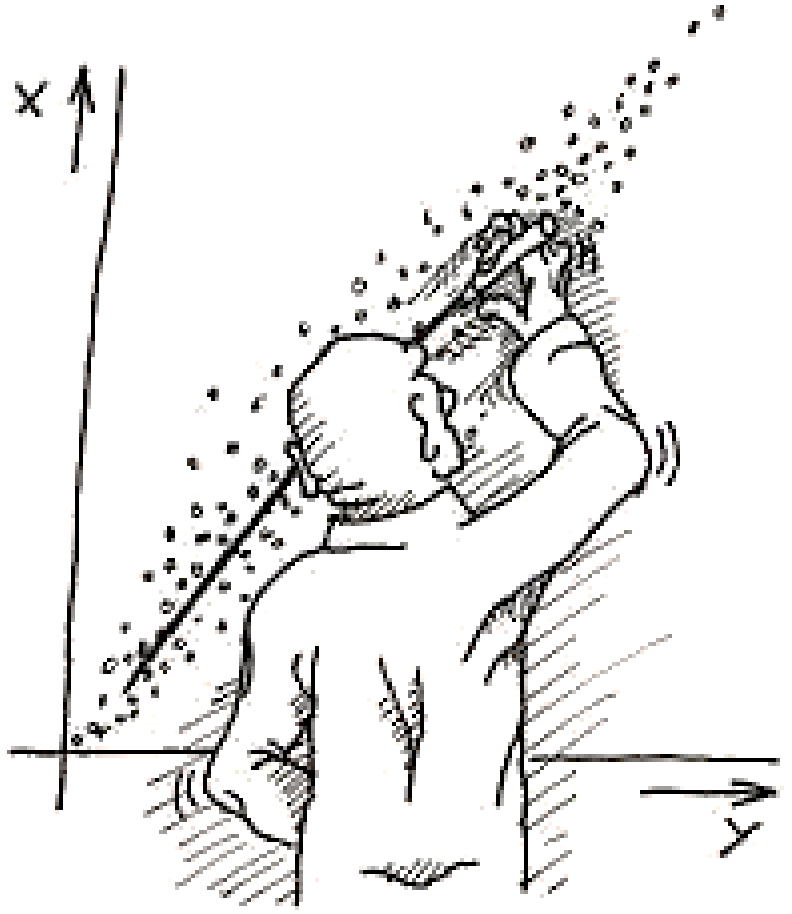
- **Capacity of a model:** refers to the ability to fit a wide variety of functions. Machine learning algorithms will generally perform best when their capacity is appropriate for the true complexity of the task they need to perform and the amount of training data they are provided with. Models with an insufficient capacity are unable to solve complex tasks. Models with a high capacity can solve complex tasks, but when their capacity is higher than that is needed to solve the present task, they may overfit.




Failing to learn features

Learning noise

# Régression



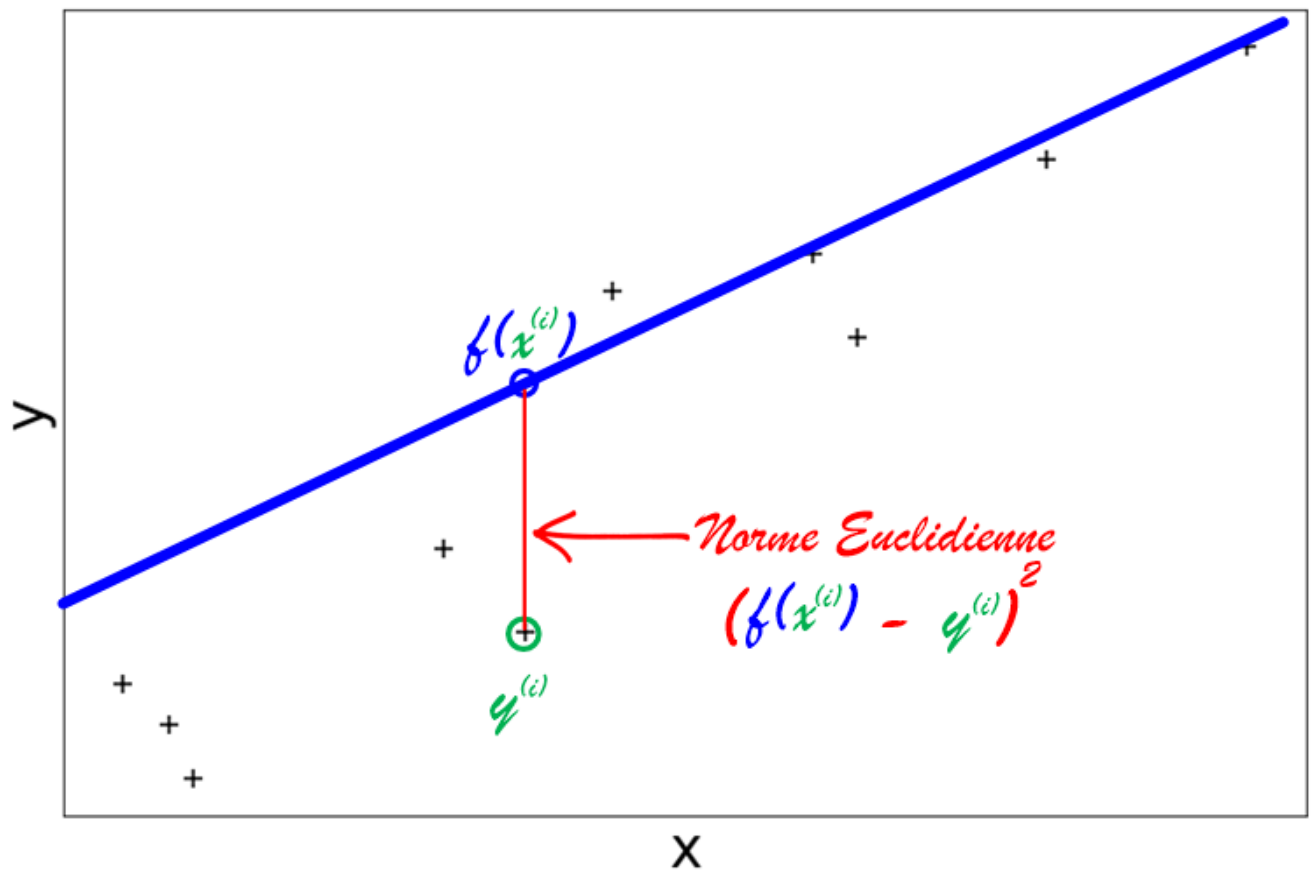
# Régression

- La régression sert à trouver la relation d'**une** variable par rapport à **une** ou **plusieurs** autres. Le but est d'estimer une valeur (numérique) de sortie à partir des valeurs d'un ensemble de caractéristiques en entrée. Par exemple, estimer le prix d'une maison en se basant sur sa surface, nombre des étages, son emplacement, etc.

Donc, le problème revient à:

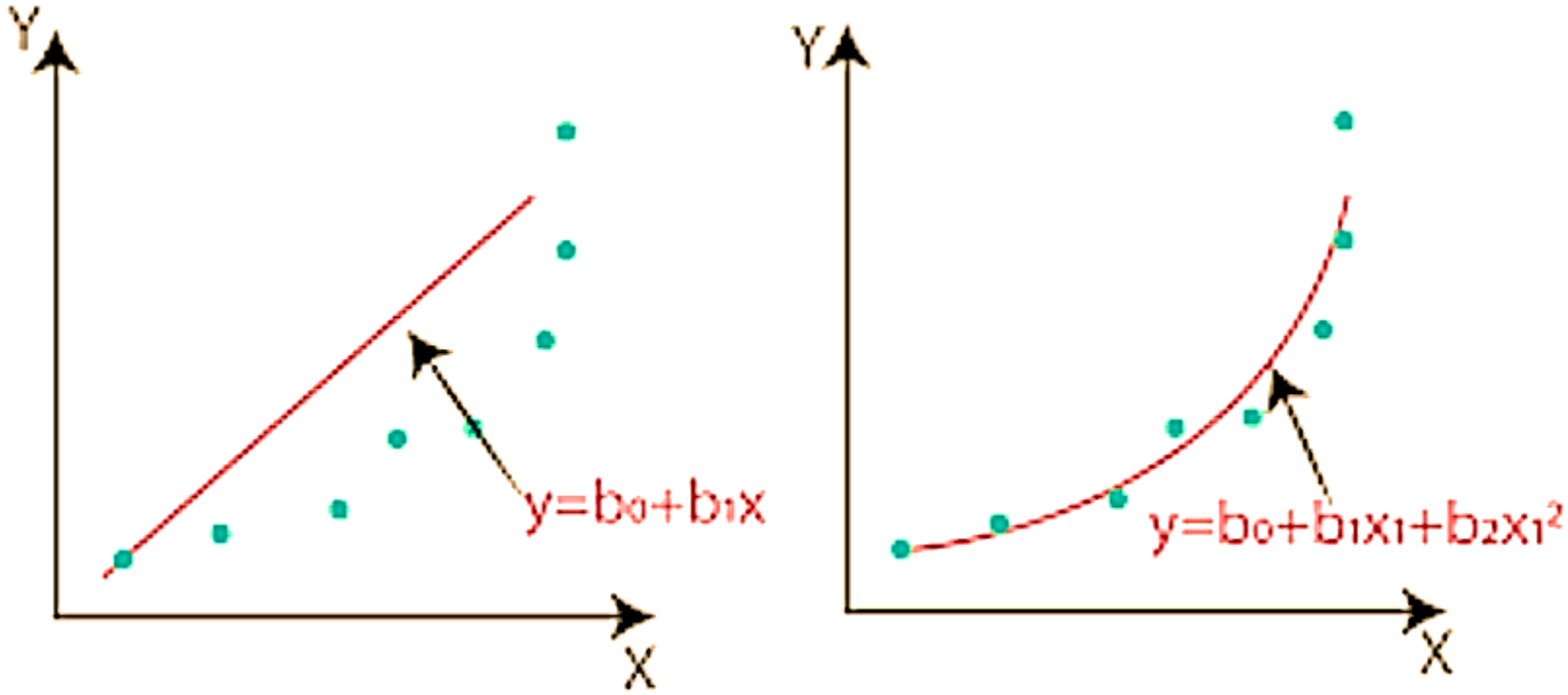
- **Estimer une fonction de calcul (fonction d'hypothèse)** en se basant sur des données d'entraînement.
- **Définir une fonction coût** : c'est une fonction mathématique qui mesure l'erreur que nous commettons en approximant les données. Nous parlons aussi d'erreur induite par la modélisation.

- **Minimiser cette fonction coût** : il faut trouver les bons paramètres de notre modèle pour minimiser l'erreur de modélisation.
- **Choisir une méthode de résolution du problème**. Il existe deux méthodes :
  - une méthode de résolution numérique, la descente de gradient,
  - une méthode analytique, la méthode des moindres carrés.


y
x
$f(x^{(i)})$
$y^{(i)}$
Norme Euclidienne
$(f(x^{(i)}) - y^{(i)})^2$

- Il existe plusieurs algorithmes pour la régression:
- Régression linéaire: Dans le cadre d'un modèle linéaire simple, on peut représenter graphiquement la relation entre x et y à travers un nuage de points. L'estimation du modèle linéaire permet de tracer la droite de régression, d'équation: $y=w_1X+w_0$. Le paramètre $w_0$ représente l'ordonnée à l'origine et $w_1$ le coefficient directeur de la droite.
- Régression polynomiale: La régression polynomiale est une approche statistique qui est employée pour modéliser une relation de forme **non-linéaire** entre la réponse (y) et les variables explicatives (x). Pour prendre en charge cette forme non-linéaire de la relation entre y et x, ces modèles de régression intègrent des polynômes dans leurs équations : $y=w_2X^2+w_1X+w_0$

# Regression lineaire et polynomiale

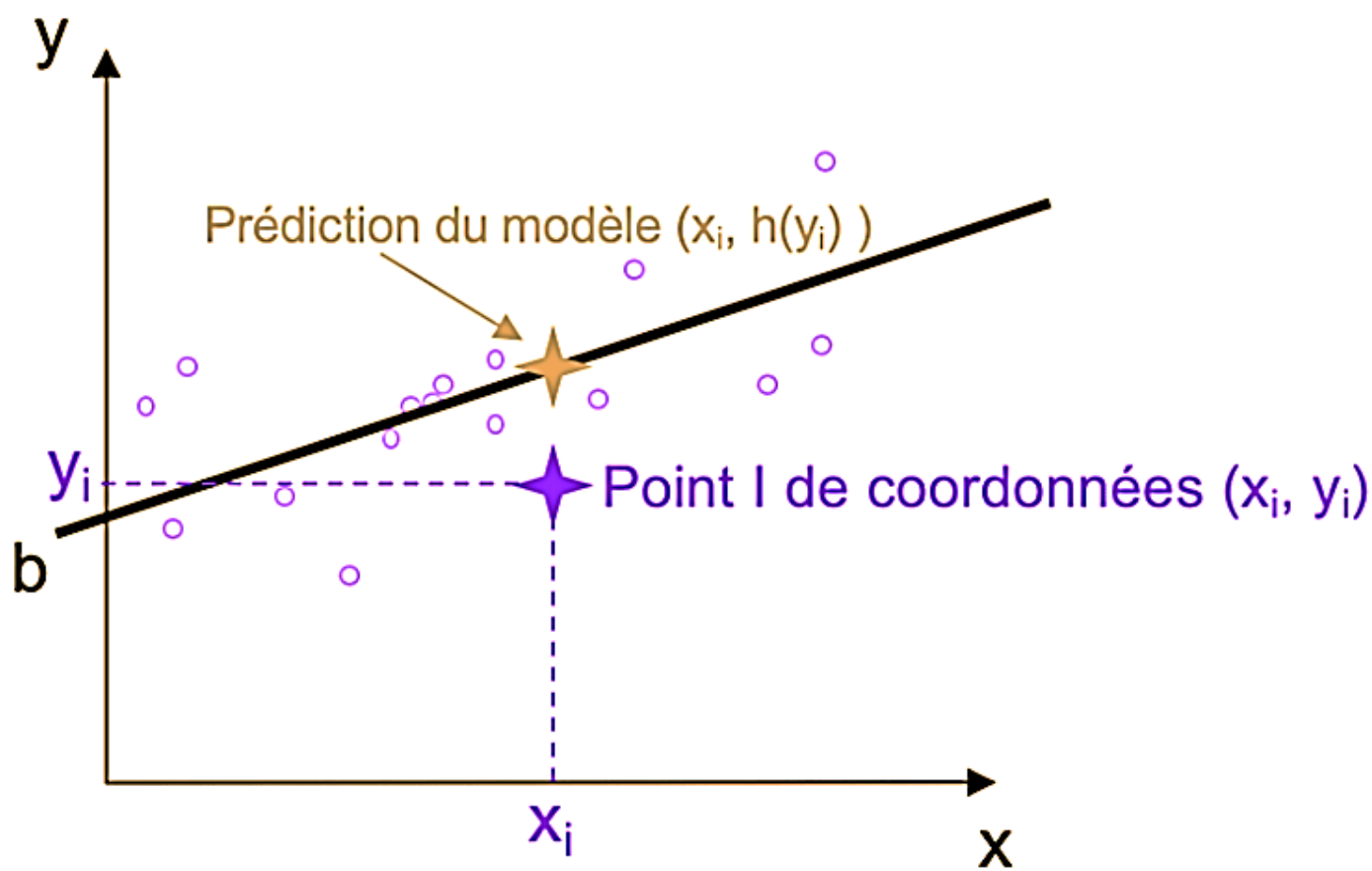


# Régression linéaire

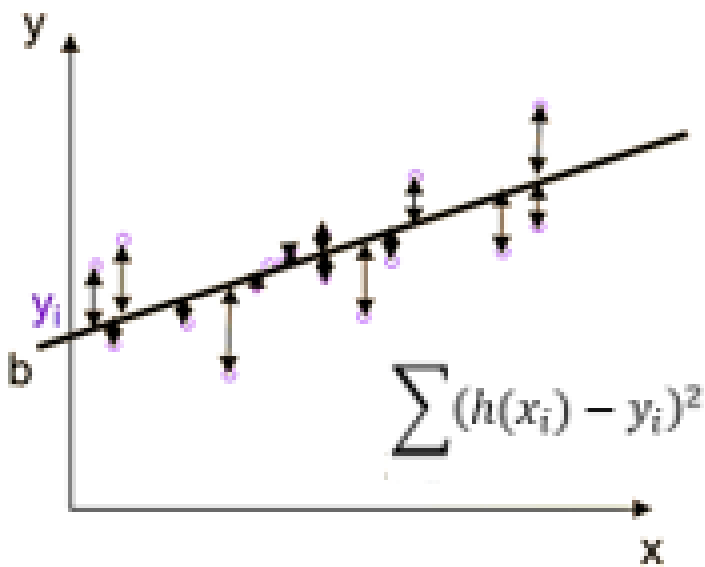
Pour chaque point $x_i$, la fonction hypothèse associe une valeur définie par $h(x_i)$ qui est plus ou moins proche de la variable cible $y_i$.

l'erreur unitaire pour $x_i$ ***est définie par*** par $h(x_i) - y_i$.

Or, chaque erreur unitaire pouvant être positive ou négative.. Il faut donc faire en sorte que la contribution de chaque erreur soit systématiquement pénalisante. Alors, on élève au carré l'erreur unitaire


y
Prédiction du modèle ($x_i$, h($y_i$) )
$y_i$
Point I de coordonnées ($x_i$, $y_i$)
b
$x_i$
x

).


y
$y_i$
b
$\sum (h(x_i) - y_i)^2$
x

On somme l'ensemble des erreurs unitaires pour l'ensemble des points de données : on parle alors d'erreur quadratique (erreur au carré) :

$$\sum (h(x_i) - y_i)^2$$

La fonction de coût s'écrit alors en pondérant la somme des erreurs quadratiques par le nombre de points dans la base d'apprentissage :

$$\frac{1}{n}\sum (h(x_i) - y_i)^2$$

# Régression linéaire univariée

Dans notre cas de régression linéaire univariée, la fonction de coût devient :

$$\frac{1}{n}\sum(w0 + w1x - y_i)^2$$

Déterminer les meilleurs paramètres **($w_0$, $w_1$)** pour la fonction hypothèse ***h*** revient à trouver la meilleure **droite**, celle qui minimise la somme de toutes les erreurs unitaires.

Du point de vue arithmétique, il s'agit de trouver le minimum de la fonction de coût. Pour cela il existe plusieurs méthodes:

- **Méthode Analytique par OLS** (ordinary least square): l'idée est de réduire le carré des erreurs et estimer les coefficients $b_0$ et $b_1$ par des relations mathématiques fermées. (Il n'y a pas besoin de faire d'algorithmes compliqués qui sont itératifs)
- **Méthode numérique par gradient descent** (gradient descendant). C'est la plus utilisée en machine learning pour résoudre les problèmes. C'est un algorithme qui procède par itération pour déterminer les valeurs des coefficients.

# Linear regression

Regression means:
**approximating a functional dependency based on measured data.**

“learning a function” or “predicting continuous values”

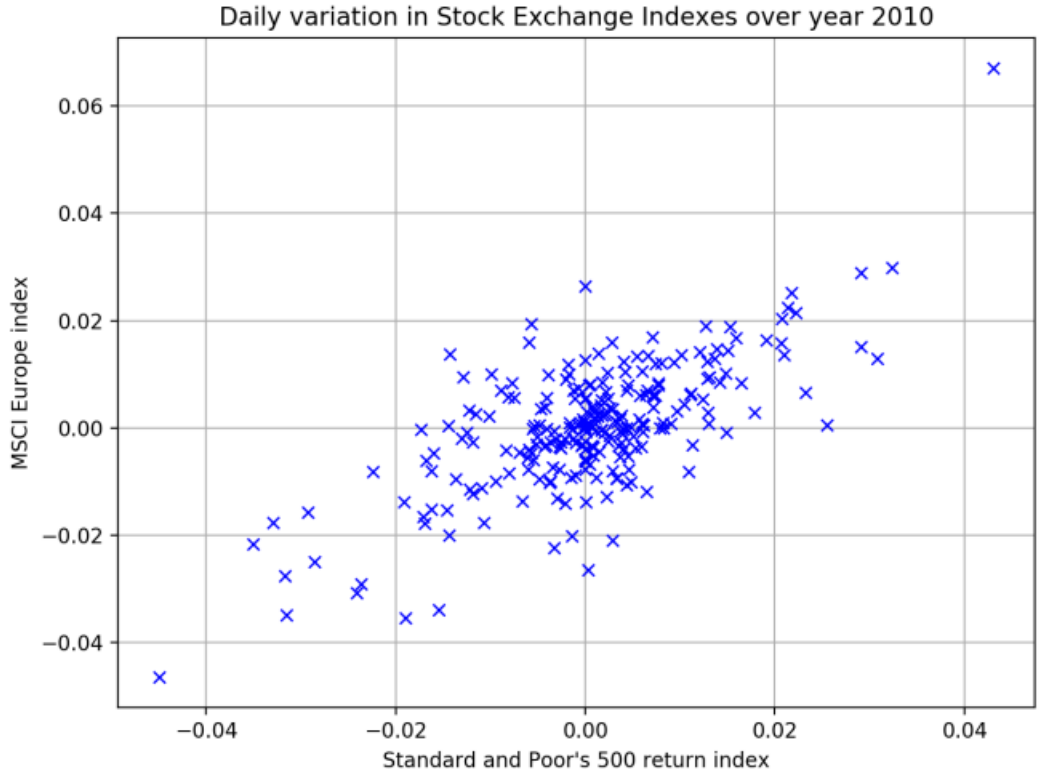
a typical supervised problem

# Data

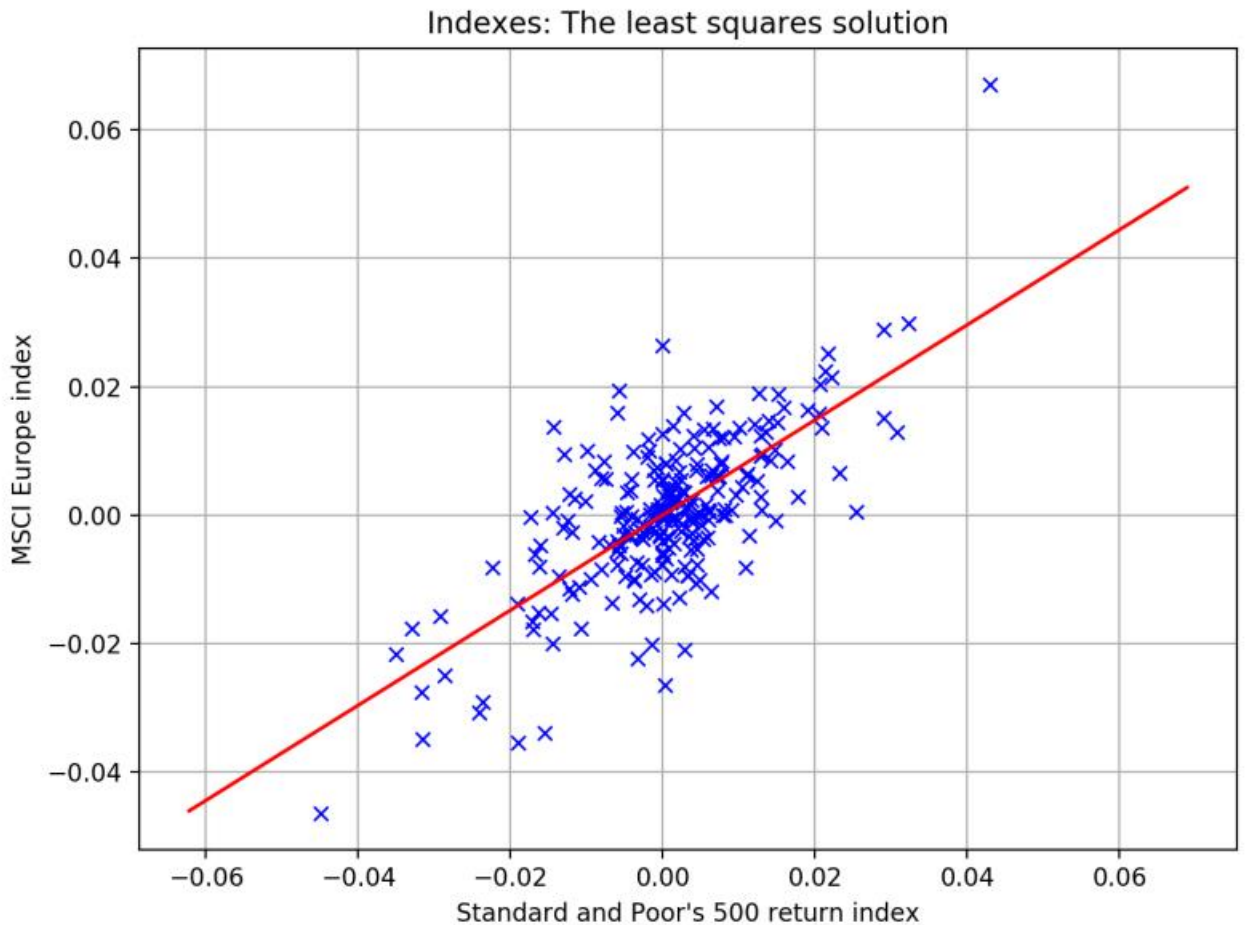
$$\begin{pmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \\ \mathbf{x}_3 \\ \vdots \\ \mathbf{x}_N \end{pmatrix} \quad \begin{pmatrix} \mathbf{t}_1 \\ \mathbf{t}_2 \\ \mathbf{t}_3 \\ \vdots \\ \mathbf{t}_N \end{pmatrix}$$

Observations Target

# Exemple 1  One dimensional Linear regression




249 observations in year 2010. (Source: UCI)
We want to predict the variation of the MSCI European index by observing Standard and Poor 500 return index. quad


Indexes: The least squares solution
MSCI Europe index
Standard and Poor's 500 return index
0.06
0.04
0.02
0.00
−0.02
−0.04
−0.06
−0.04
−0.02
0.00
0.02
0.04
0.06

- Observation: $x$ is the value of the variation of Standard and Poor's (SP) 500 return index on a given day.
- Target: $t$ is the value of the variation of the MSCI European index (MSCI) on the same day.

There is clearly some relationship between the two values.

But not one-to-one

# A model for approximating the data

A linear model $y(x)$ that predicts $t$ given $x$:

$$t \approx y \quad \text{where} \quad y = wx$$

For instance

$t_1 \approx y_1$ where $y_1 = wx_1$

or $t_2 \approx y_2$ where $y_2 = wx_2$

**We want $y(x)$ to be similar to $t(x)$ for any $x$.**

# A solution based on optimization

Idea:

- quantify **how wrong** is each estimate using some **measure**,
- make this measure as small as possible **on average**.

Measure = **loss function**

Overall evaluation (e.g. average) = **objective function**

# Properties of the square error loss

- is even: $(t-y)^2 = (y-t)^2$
- grows more than linearly, giving heavier weight to larger errors
- is differentiable with respect to the model output:

$$\frac{d}{dy}\lambda_{\text{SE}}(y,t) = 2(y-t)$$

**Square error loss**

$$\lambda_{\text{SE}}(y,t) = (y-t)^2$$

# The objective function

Generic goal: **minimize the mean value of the loss over the whole data set**

$$J = \frac{1}{N}\sum_{l=1}^{N}\lambda(y_l, t_l)\,.$$

$J$ = Objective function or cost function

In the specific case of the square error loss:

$$J_{\text{MSE}} = \frac{1}{N}\sum_{l=1}^{N}(y_l - t_l)^2\,.$$

$J_{\text{MSE}}$ = mean square error objective.

# The objective function

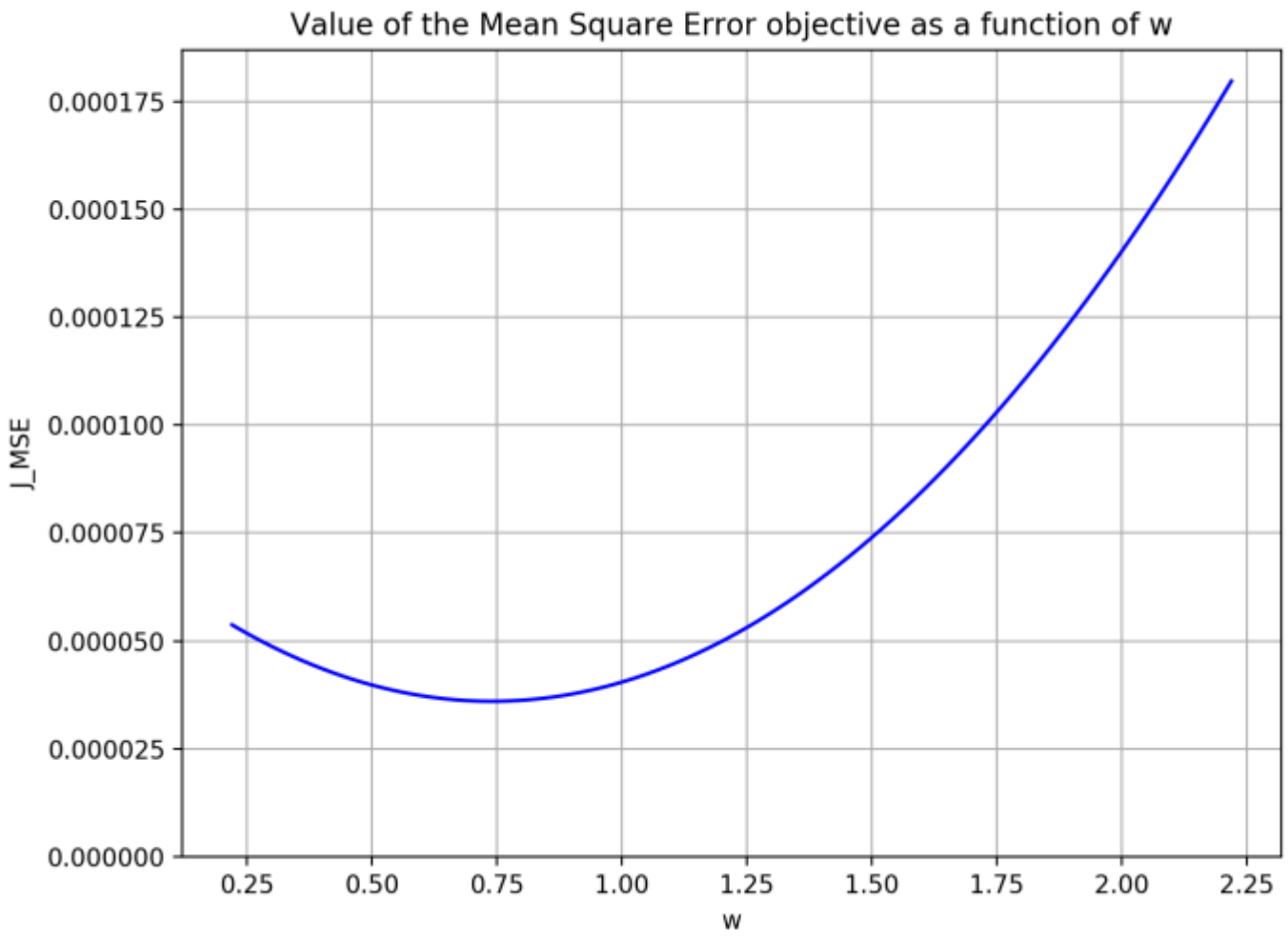
A loss function $\lambda(y, t)$ is a function of the two arguments $y$ and $t$.

We can look at the objective in two complementary situations:

- Given the data set, the targets $t_1, \ldots, t_N$ are fixed
  $\longrightarrow$ **The objective depends only on the model parameter(s)**

- Given a model, the parameters are fixed.
  However we can apply this model to various data sets.
  $\longrightarrow$ **The objective depends only on the data.**

# The objective as a function of the model parameter(s)



- When **building a model (= training)**, the objective is a function of the parameters in the model and the data are fixed

- When **using a model (= inference)**, the objective is a function of the data, which uses the (now fixed) model parameters.

# The least squares method

**Problem:**
minimize $J_{\text{MSE}}$ with respect to the parameters with fixed data

we know that $J_{\text{MSE}}$ is a parabola

**Unique solution:** $w$ such that

$$\frac{d}{dw} J_{\text{MSE}} = 0$$

# Computing $\frac{d}{dw}J_{\text{MSE}}$

$$\frac{d}{dw}\lambda_{\text{SE}}(y,t) = \frac{d}{dy}\lambda_{\text{SE}}(y,t) \times \frac{d}{dw}y$$

Here we have used the chain rule of differentiation to write the derivative:

$$\frac{df(g(x))}{dx} = \left.\frac{df(y)}{dy}\right|_{y=g(x)} \frac{dg(x)}{dx}$$

$$\left.\frac{d}{dy}(y-t)^2\right|_{y=wx} \quad \frac{d}{dw}(wx)$$

$$2(xw-t) \quad x$$

# Derivative of the loss

This is for one observation

$$\frac{d}{dw}\lambda_{\mathrm{SE}}(t,y) = 2(xw-t)(x)$$
$$= 2x^2 w - 2xt\ .$$

# Derivative of the objective

This is for the average over all observations

$$\frac{d}{dw} J_{\mathrm{MSE}} = \frac{d}{dw} \frac{1}{N} \sum_{l=1}^{N} \lambda_{\mathrm{SE}}(y_l, t_l)$$

Exchange sum and derivative:

$$= \frac{1}{N} \sum_{l=1}^{N} \frac{d}{dw} \lambda_{\mathrm{SE}}(y_l, t_l)$$

$$= \frac{1}{N} \sum_{l=1}^{N} 2\left( x_l^2 w - x_l t_l \right) ,$$

where the constant coefficient $(2/N)$ is irrelevant and can be disregarded.

This equation is solved by bringing $w$ outside the sum, since it does not depend on $l$:

$$w\sum_{l=1}^{N} x_l^2 - \sum_{l=1}^{N} x_l t_l = 0$$

$$w = \frac{\sum_{l=1}^{N} x_l t_l}{\sum_{l=1}^{N} x_l^2}$$

This is the least squares solution to the linear regression problem.

# Metrics to evaluate the regression

| Regression | |
|---|---|
| 'explained_variance' | `metrics.explained_variance_score` |
| 'max_error' | `metrics.max_error` |
| 'neg_mean_absolute_error' | `metrics.mean_absolute_error` |
| 'neg_mean_squared_error' | `metrics.mean_squared_error` |
| 'neg_root_mean_squared_error' | `metrics.mean_squared_error` |
| 'neg_mean_squared_log_error' | `metrics.mean_squared_log_error` |
| 'neg_median_absolute_error' | `metrics.median_absolute_error` |
| 'r2' | `metrics.r2_score` |
| 'neg_mean_poisson_deviance' | `metrics.mean_poisson_deviance` |
| 'neg_mean_gamma_deviance' | `metrics.mean_gamma_deviance` |
| 'neg_mean_absolute_percentage_error' | `metrics.mean_absolute_percentage_error` |
| 'd2_absolute_error_score' | `metrics.d2_absolute_error_score` |
| 'd2_pinball_score' | `metrics.d2_pinball_score` |
| 'd2_tweedie_score' | `metrics.d2_tweedie_score` |

# R2-score

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2}$$

# Exemple 2

## 1974 “Motor Trends” car data (four columns)

| Model | mpg | disp | hp | weight |
|---|---|---|---|---|
| Mazda RX4 | 21 | 160 | 110 | 2.62 |
| Mazda RX4 Wag | 21 | 160 | 110 | 2.875 |
| Datsun 710 | 22.8 | 108 | 93 | 2.32 |
| Hornet 4 Drive | 21.4 | 258 | 110 | 3.215 |
| Hornet Sportabout | 18.7 | 360 | 175 | 3.44 |
| Valiant | 18.1 | 225 | 105 | 3.46 |
| Duster 360 | 14.3 | 360 | 245 | 3.57 |
| Merc 240D | 24.4 | 146.7 | 62 | 3.19 |
| Merc 230 | 22.8 | 140.8 | 95 | 3.15 |
| Merc 280 | 19.2 | 167.6 | 123 | 3.44 |
| Merc 280C | 17.8 | 167.6 | 123 | 3.44 |
| Merc 450SE | 16.4 | 275.8 | 180 | 4.07 |
| Merc 450SL | 17.3 | 275.8 | 180 | 3.73 |
| Merc 450SLC | 15.2 | 275.8 | 180 | 3.78 |
| Cadillac Fleetwood | 10.4 | 472 | 205 | 5.25 |
| Lincoln Continental | 10.4 | 460 | 215 | 5.424 |
| Chrysler Imperial | 14.7 | 440 | 230 | 5.345 |
| Fiat 128 | 32.4 | 78.7 | 66 | 2.2 |
| Honda Civic | 30.4 | 75.7 | 52 | 1.615 |
| Toyota Corolla | 33.9 | 71.1 | 65 | 1.835 |
| Toyota Corona | 21.5 | 120.1 | 97 | 2.465 |
| Dodge Challenger | 15.5 | 318 | 150 | 3.52 |
| AMC Javelin | 15.2 | 304 | 150 | 3.435 |
| Camaro Z28 | 13.3 | 350 | 245 | 3.84 |
| Pontiac Firebird | 19.2 | 400 | 175 | 3.845 |
| Fiat X1-9 | 27.3 | 79 | 66 | 1.935 |
| Porsche 914-2 | 26 | 120.3 | 91 | 2.14 |
| Lotus Europa | 30.4 | 95.1 | 113 | 1.513 |
| Ford Pantera L | 15.8 | 351 | 264 | 3.17 |
| Ferrari Dino | 19.7 | 145 | 175 | 2.77 |
| Maserati Bora | 15 | 301 | 335 | 3.57 |
| Volvo 142E | 21.4 | 121 | 109 | 2.78 |

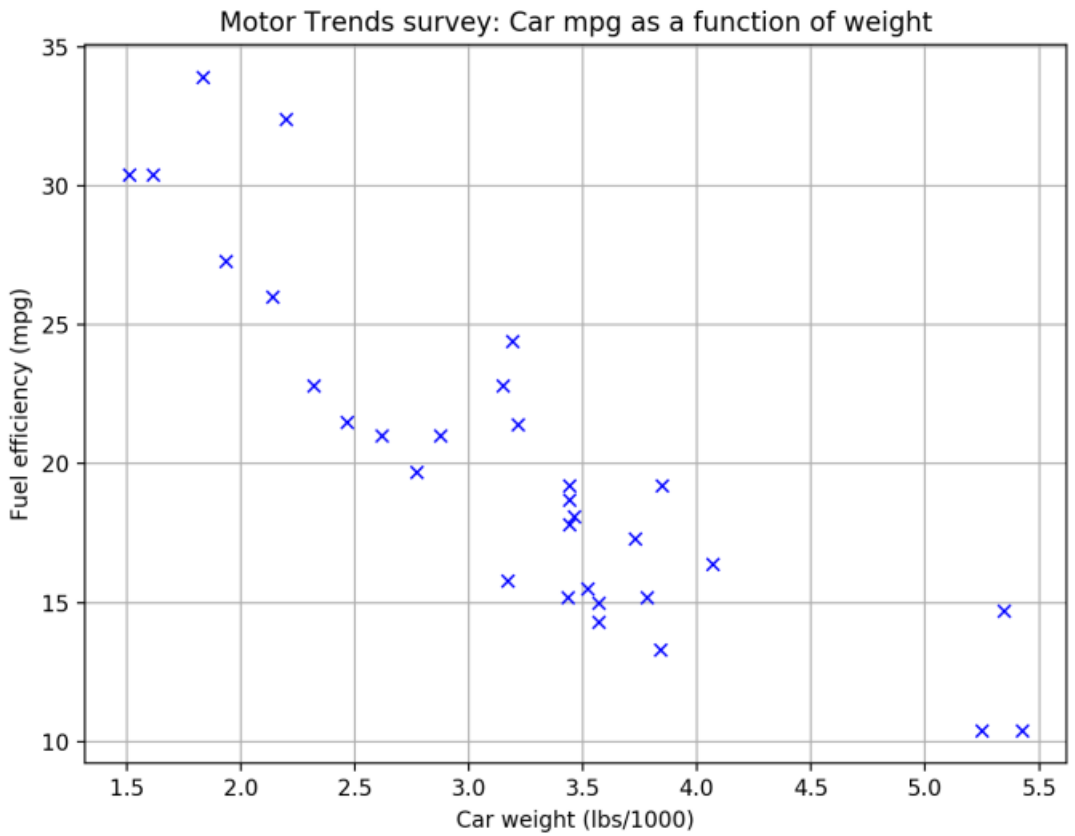
1974 survey on some car models.

Among other variables:

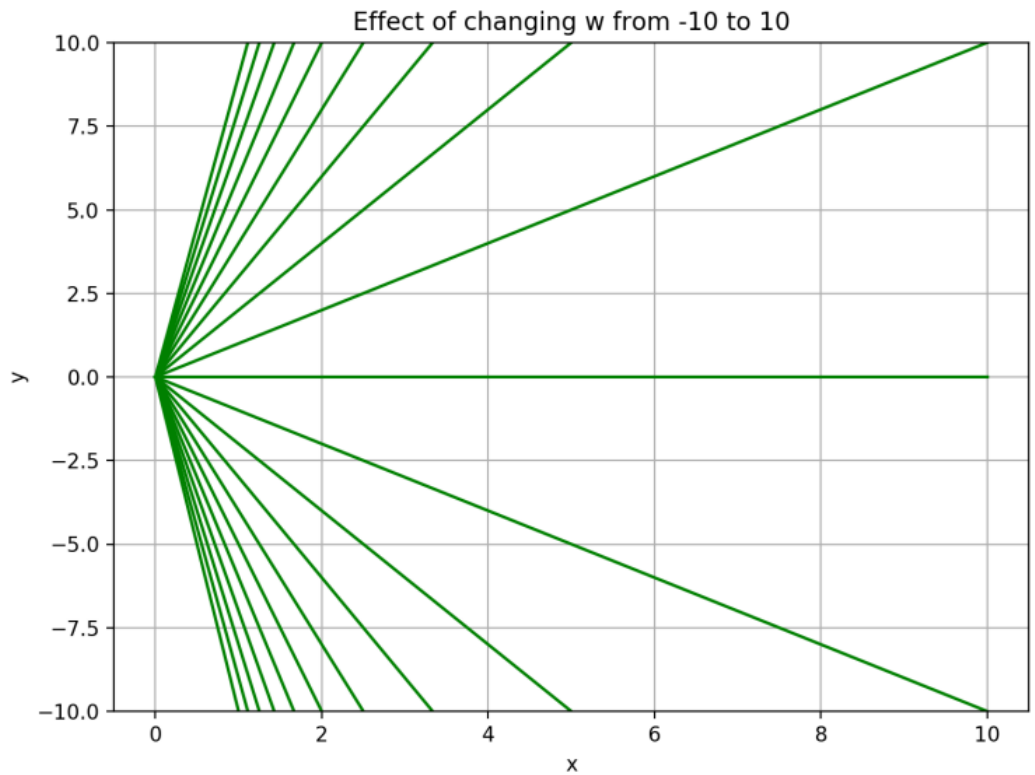
- mpg or miles-per-gallon
- disp or displacement (in cu.in)
- hp or horse-power
- weight, total (in lbs/1000)

(Note that these are USA units.)

# Forecasting mpg with weight

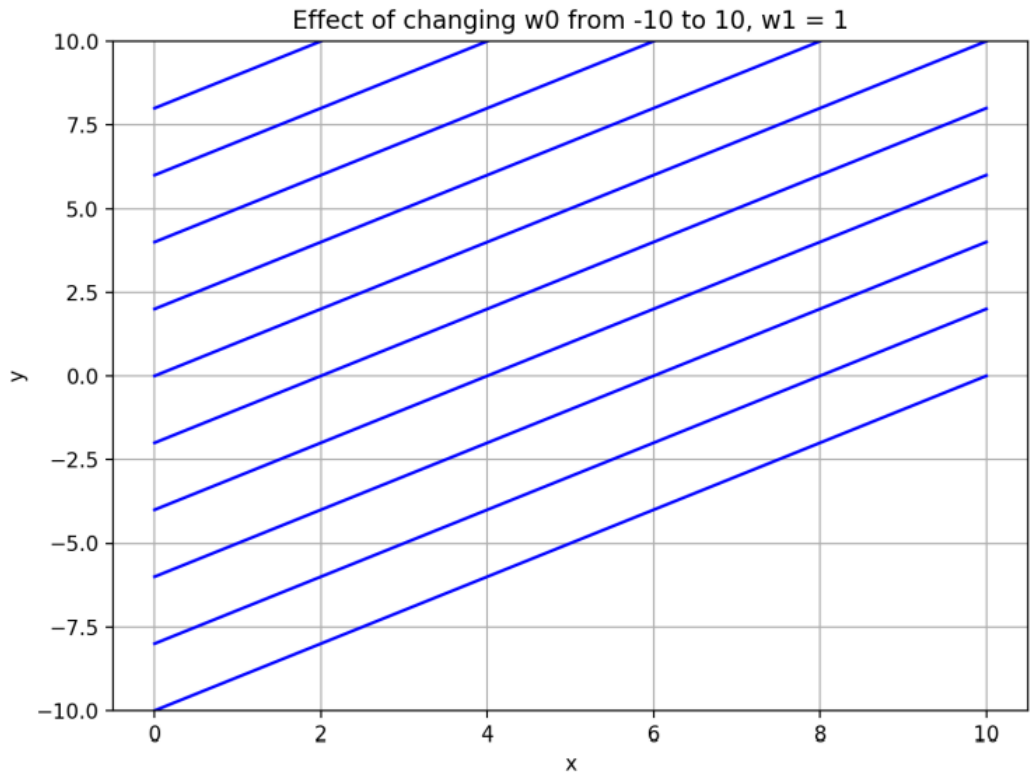


$$y = wx$$


Effect of changing w from -10 to 10
10.0
7.5
5.0
2.5
0.0
−2.5
−5.0
−7.5
−10.0
y
0
2
4
6
8
10
x

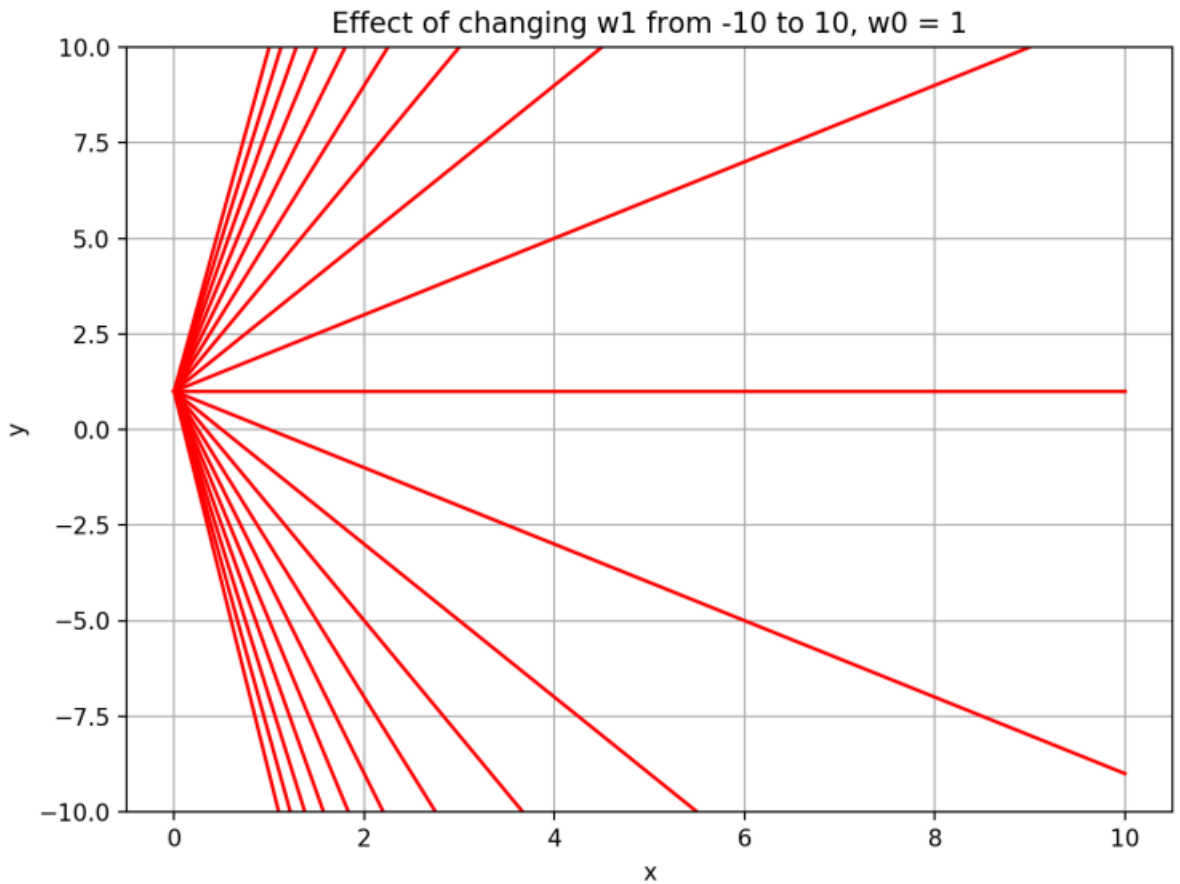
# A more flexible model

$$y = w_1 x + w_0$$



# A more flexible model

$$y = w_1 x + w_0$$



# Solving

The solution in this case can be found by **centering** around the mean $\bar{x}$ of $x$ and $\bar{t}$ of $t$

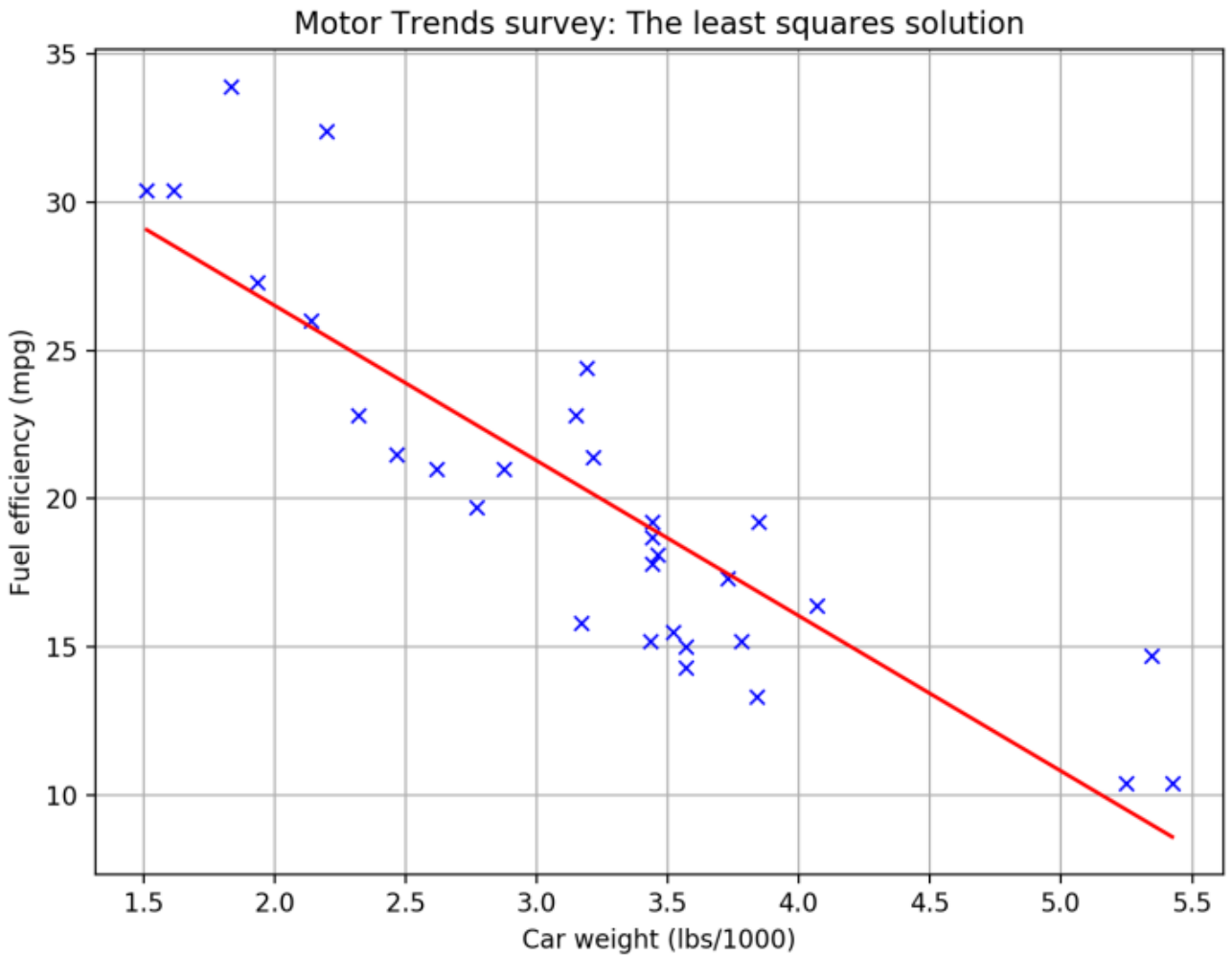
$$\bar{x} = \frac{1}{N}\sum_{l=1}^{N} x_l \qquad \bar{t} = \frac{1}{N}\sum_{l=1}^{N} t_l$$

$$w_1 = \frac{\sum_{l=1}^{N}(x_l - \bar{x})(t_l - \bar{t})}{\sum_{l=1}^{N}(x_l - \bar{x})^2}$$

$$w_0 = \bar{t} - w_1\bar{x}$$

- w1 = slope; gain

w0 = intercept, offset; bias


Motor Trends survey: The least squares solution
Fuel efficiency (mpg)
Car weight (lbs/1000)
35
30
25
20
15
10
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0
5.5

# Multidimensional linear regression

The data is now composed of $d$-dimensional vectors:

$$\mathbf{x}_1 = [\, x_{1,1}, x_{1,2}, \ldots, x_{1,d} \,]$$
$$\mathbf{x}_2 = [\, x_{2,1}, x_{2,2}, \ldots, x_{2,d} \,]$$
$$\ldots$$
$$\mathbf{x}_N = [\, x_{N,1}, x_{N,2}, \ldots, x_{N,d} \,]$$

so we can organize them into a $N \times d$ matrix:

$$X = \begin{pmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \\ \mathbf{x}_3 \\ \vdots \\ \mathbf{x}_N \end{pmatrix} = \begin{pmatrix} x_{1,1} & x_{1,2} & \ldots & x_{1,d} \\ x_{2,1} & x_{2,2} & \ldots & x_{2,d} \\ x_{3,1} & x_{3,2} & \ldots & x_{3,d} \\ & & \vdots & \\ x_{N,1} & x_{N,2} & \ldots & x_{N,d} \end{pmatrix}$$

Since the data are now $d$-dimensional, we have $d$ parameters in a $d$-dimensional vector:

$$\mathbf{w} = \begin{pmatrix} w_1 \\ w_2 \\ w_3 \\ \vdots \\ w_d \end{pmatrix}$$

This can be expressed as a matrix-vector multiplication between the data matrix $X$ and the parameter vector $\mathbf{w}$:

$$\mathbf{y} = \begin{pmatrix} y_1 \\ y_2 \\ y_3 \\ \vdots \\ y_N \end{pmatrix} = \begin{pmatrix} x_{1,1} & x_{1,2} & \dots & x_{1,d} \\ x_{2,1} & x_{2,2} & \dots & x_{2,d} \\ x_{3,1} & x_{3,2} & \dots & x_{3,d} \\ & & \vdots & \\ x_{N,1} & x_{N,2} & \dots & x_{N,d} \end{pmatrix} \begin{pmatrix} w_1 \\ w_2 \\ w_3 \\ \vdots \\ w_d \end{pmatrix}$$

$$= \begin{pmatrix} x_{1,1}w_1 + x_{1,2}w_2 + & \dots & +x_{1,d}w_d \\ x_{2,1}w_1 + x_{2,2}w_2 + & \dots & +x_{2,d}w_d \\ x_{3,1}w_1 + x_{3,2}w_2 + & \dots & +x_{3,d}w_d \\ & & \vdots \\ x_{N,1}w_1 + x_{N,2}w_2 + & \dots & +x_{N,d}w_d \end{pmatrix} = X\mathbf{w}$$

Finally, our goal is to make this model's prediction $\mathbf{y}$ as similar as possible to the measured outputs for each observation, which again can be organized as a vector, this time $N$-dimensional:

$$\mathbf{t} = \begin{pmatrix} t_1 \\ t_2 \\ t_3 \\ \vdots \\ t_N \end{pmatrix}$$

## The square error objective in matrix-vector form

$$
\begin{aligned}
J_{\text{MSE}} &= \frac{1}{2}\|\mathbf{y}-\mathbf{t}\|^2 \\
&= \frac{1}{2}\|X\mathbf{w}-\mathbf{t}\|^2 \\
&= \frac{1}{2}(X\mathbf{w}-\mathbf{t})^T(X\mathbf{w}-\mathbf{t}) \\
&= \frac{1}{2}(\mathbf{w}^T X^T-\mathbf{t}^T)(X\mathbf{w}-\mathbf{t}) \\
&= \frac{1}{2}\left(\mathbf{w}^T X^T X\mathbf{w}-\mathbf{w}^T X^T\mathbf{t}-\mathbf{t}^T X\mathbf{w}+\mathbf{t}^T\mathbf{t}\right) \\
&= \frac{1}{2}\|X\mathbf{w}\|^2-\mathbf{w}^T X^T\mathbf{t}\frac{1}{2}\|\mathbf{t}\|^2
\end{aligned}
$$

# Closed-form solution

It can be proven that we can write:

$$\begin{aligned}\nabla J_{\text{MSE}} &= \frac{\partial}{\partial \mathbf{w}} J_{\text{MSE}} \\ &= X^T X \mathbf{w} - X^T \mathbf{t}\end{aligned}$$

Setting $\nabla J_{\text{MSE}} = \mathbf{0}$ we get

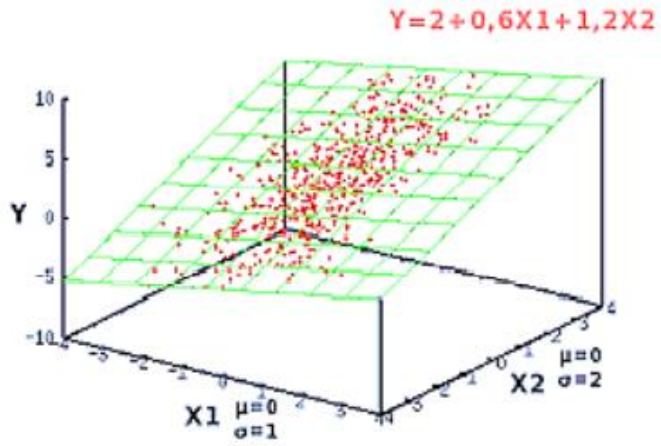
$$X^T X \mathbf{w} = X^T \mathbf{t}$$

By premultiplying both sides by $(X^T X)^{-1}$, we obtain the closed-form solution:

$$\mathbf{w} = (X^T X)^{-1} X^T \mathbf{t}$$

The matrix form of the **normal equations** for the least squares problem.

# Example



# Problem!!

## Invertibility and stability

- $X^T X$ might not be invertible. This is when the data have exactly linearly dependent components
- What if the variables are **correlated**?
- Even if $X^T X$ has full rank, in the case of correlated variables it will have a high **condition number** $\lambda_{\text{first}}/\lambda_{\text{last}}$.
- DIFFICULT TO INVERT (numerical precision must be too high)
- This is a problem of **numeric instability**: even very small numeric errors are amplified by the condition number and cause large errors on the result

**SOLUTION: Iterative computation by successive approximations**

# Problem!!


To solve the problem of overfitting in linear regression, the technique of **regularization** is used in such cases.

# Polynomial Regression

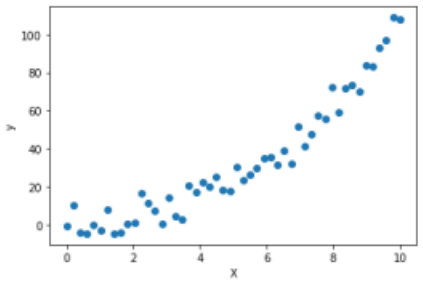
$$y = h(x) = b_0 + b_1 x + b_2 x^2 + \ldots . + b_n x^n$$

# Python

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import PolynomialFeatures
from sklearn.metrics import mean_squared_error, r2_score
```

Maintenant, en utilisant les lignes de code suivantes, nous pouvons générer des nuages de points ayant un rapport linéaire :

```
# Étape 1: génération de données
m = 50    #création de 50 échantillons
X = np.linspace(0,10, m).reshape(m, 1)  # Vecteur X
y = X**2 + 6*np.random.randn(m, 1)        # Y = X2 + la déviation par
#rapport à x
plt.ylabel('y')
plt.xlabel('X')
plt.scatter(X,y)
```




```
# Étape 2:  la préparation des données
nb_degree = 2    # degré de polynome (1 -> y=b1X+b0  2-> y=b2X²+b1X+b0 )
polynomial_features = PolynomialFeatures(degree = nb_degree)
X_TRANSF = polynomial_features.fit_transform(X)

# Étape 3: définir et entraîner le modèle
model = LinearRegression() #L'utilisation de la régression linéaire sur les #données transformées
nous permet d'estimer la fonction de régression des #polynômes.

model.fit(X_TRANSF, y)   # Entraîner le modèle sur les données X,y

# Étape 4: calculer le biais et la variance
Y_NEW = model.predict(X_TRANSF)
rmse = np.sqrt(mean_squared_error(y,Y_NEW))
r2 = r2_score(y,Y_NEW)

print('RMSE: ', rmse)
print('R2: ', r2)
```

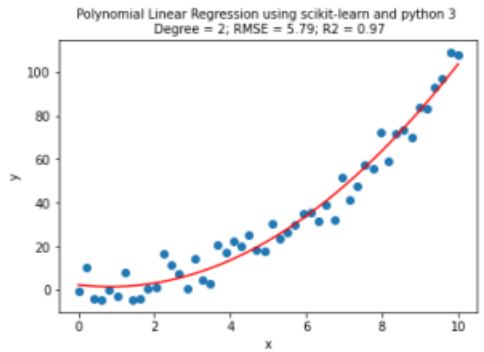
```
model.score(X_TRANSF,y)      # Evaluer le modèle sur les même données
predictions = model.predict(X_TRANSF)
plt.scatter(X,y)
title = 'Degree = {}; RMSE = {}; R2 = {}'.format(nb_degree,
round(rmse,2), round(r2,2))

plt.plot(X,predictions,c='r')
plt.title("Polynomial Linear Regression using scikit-learn and python 3
\n " + title,
          fontsize=10)
plt.xlabel('x')
plt.ylabel('y')
```

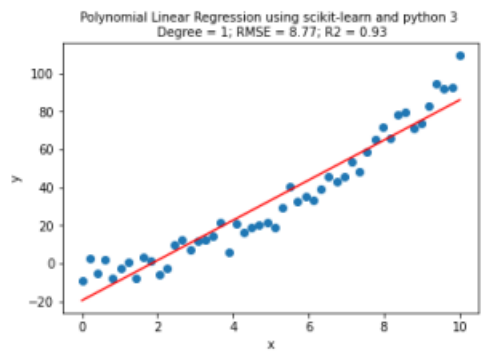
Sortie: 0.965828030085421

l'exécution de ses lignes de code, on obtient un score de 96% qui est le coefficient de détermination $R^2$ de moindre carré.

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2}$$




Maintenant si on change nb_degree pour quelle soit =1, notre modèle n'est autre que la régression linéaire univariée. Et le score sera dans ce cas de 82%.

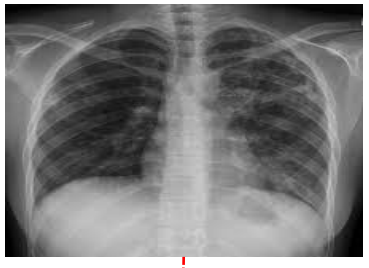


# Logistic Regression

# Example 1: Malignant or Benign

- Consider the example of recognizing whether a tumor in an input image is malignant or benign

**Input**:



Can be represented as a matrix of pixels

Binary Classifier

**Output**:

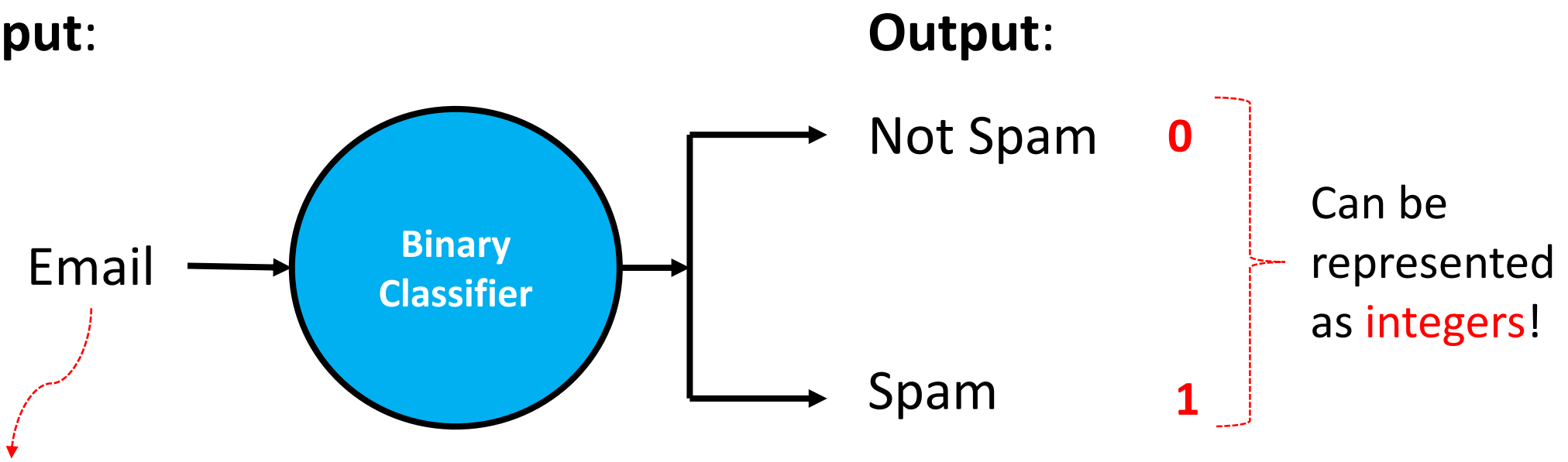
Benign **0**

Malignant **1**

Can be represented as integers!

# Example 2: Spam or Not Spam

- As another example, consider the problem of detecting whether an email is a spam or not a spam




Can be represented as a vector $\mathbf{x} = [x_1, x_2, \ldots, x_d]$, with each component $x_i$ corresponding to the presence ($x_i = 1$) or absence ($x_i = 0$) of a particular *word* (or *feature*) in the email

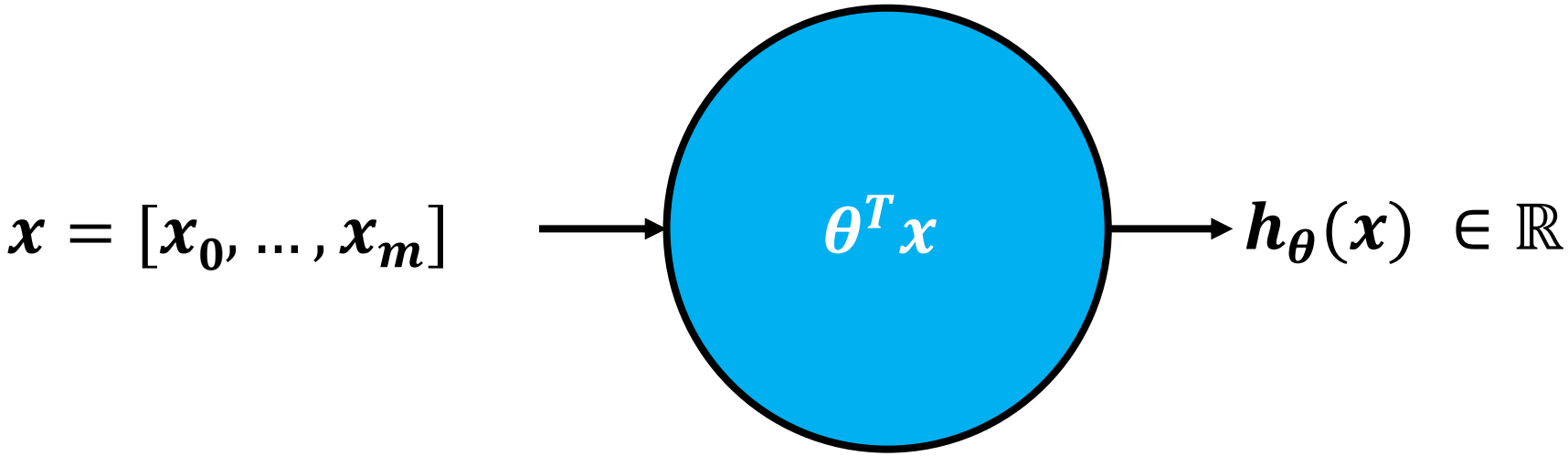
# Regression vs. Classification

- What are the possible output values of the ***linear regression model*** $\boldsymbol{h_\theta(x) = \theta^T x}$?

$\boldsymbol{\theta}$ is the ***parameter vector***, that is, $\boldsymbol{\theta} = [\boldsymbol{\theta_0}, \boldsymbol{\theta_1}, \ldots, \boldsymbol{\theta_m}]$ (assuming $m + 1$ parameters) and $\boldsymbol{x}$ is the ***feature vector***, that is, $\boldsymbol{x} = [\boldsymbol{x_0}, \boldsymbol{x_1}, \ldots, \boldsymbol{x_m}]$ (assuming $m + 1$ features and $\boldsymbol{x_0} = 1$ to account for the intercept term, namely, $\boldsymbol{\theta_0}$)

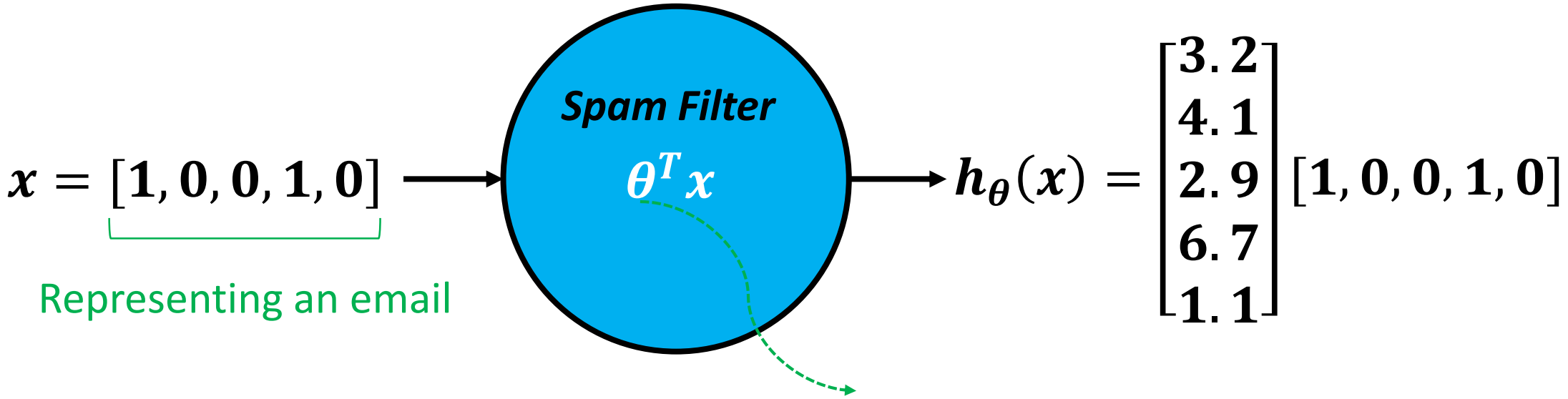
# Regression vs. Classification

- What are the possible output values of the ***linear regression model*** $\boldsymbol{h_\theta(x) = \theta^T x}$?
  - Real-valued outputs



# Regression vs. Classification

- What are the possible output values of the ***linear regression model*** $\boldsymbol{h_\theta(x) = \theta^T x}$?
  - Real-valued outputs

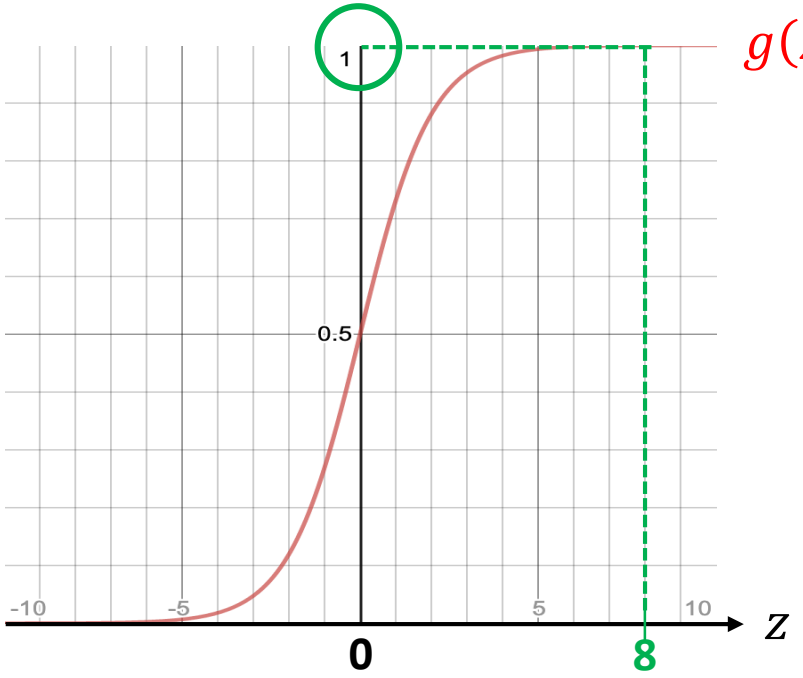


# Regression vs. Classification

- How can we make the possible outputs of $\boldsymbol{h_\theta(x) = \theta^T x}$ discrete-valued (as opposed to real-valued)?
  - By using an ***activation function*** (e.g., ***sigmoid or logistic function***)

$$g(z) = \frac{1}{1 + e^{-z}}$$

$z \in \mathbb{R}, but$
$g(z) \in [0,1]$




Assume a labeled example ($\boldsymbol{x}$, y):

If y = **1**, we want $g(z) \approx 1$ (i.e., we want a correct prediction)

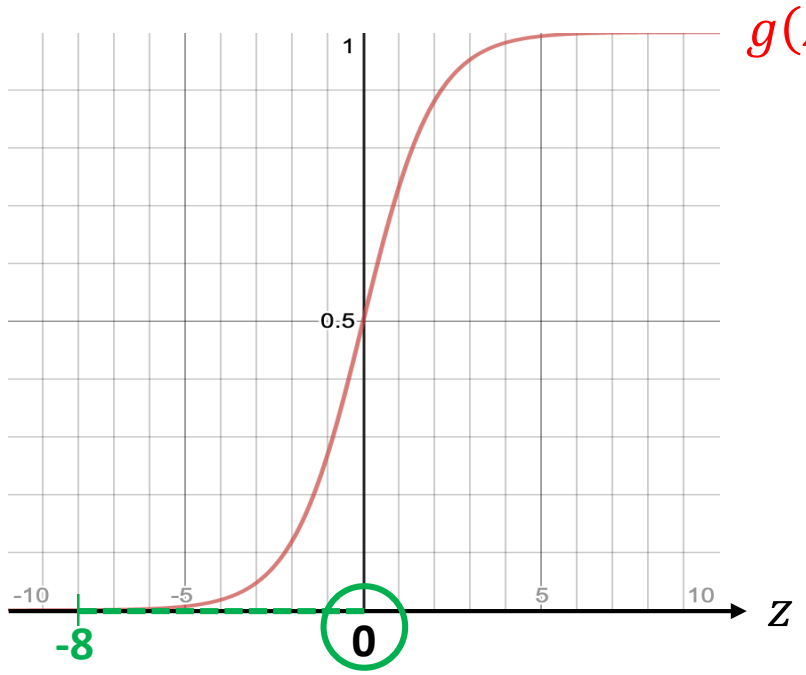
For this to happen, $\boldsymbol{z \gg 0}$

# Regression vs. Classification

- How can we make the possible outputs of $\boldsymbol{h_\theta(x) = \theta^T x}$ discrete-valued (as opposed to real-valued)?
  - By using an ***activation function*** (e.g., ***sigmoid or logistic function***)

$$g(z) = \frac{1}{1 + e^{-z}}$$

$z \in \mathbb{R}, but$
$g(z) \in [0,1]$



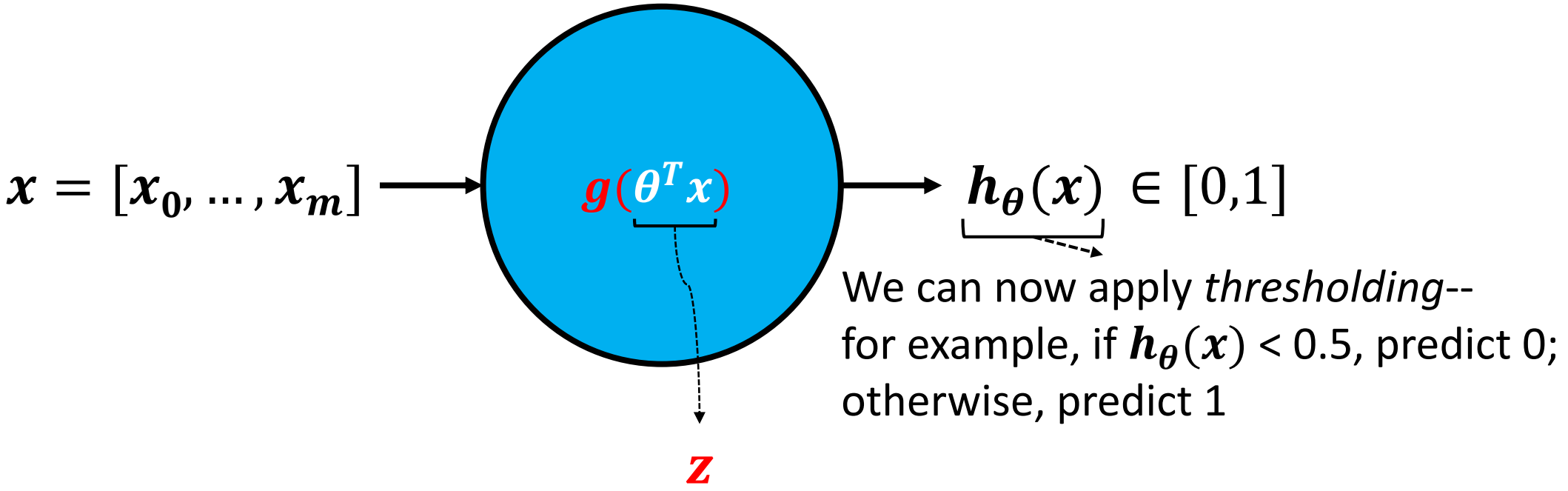

Assume a labeled example ($\boldsymbol{x}$, y):

If y = **0**, we want $g(z) \approx 0$ (i.e., we want a correct prediction)

For this to happen, $\boldsymbol{z \ll 0}$

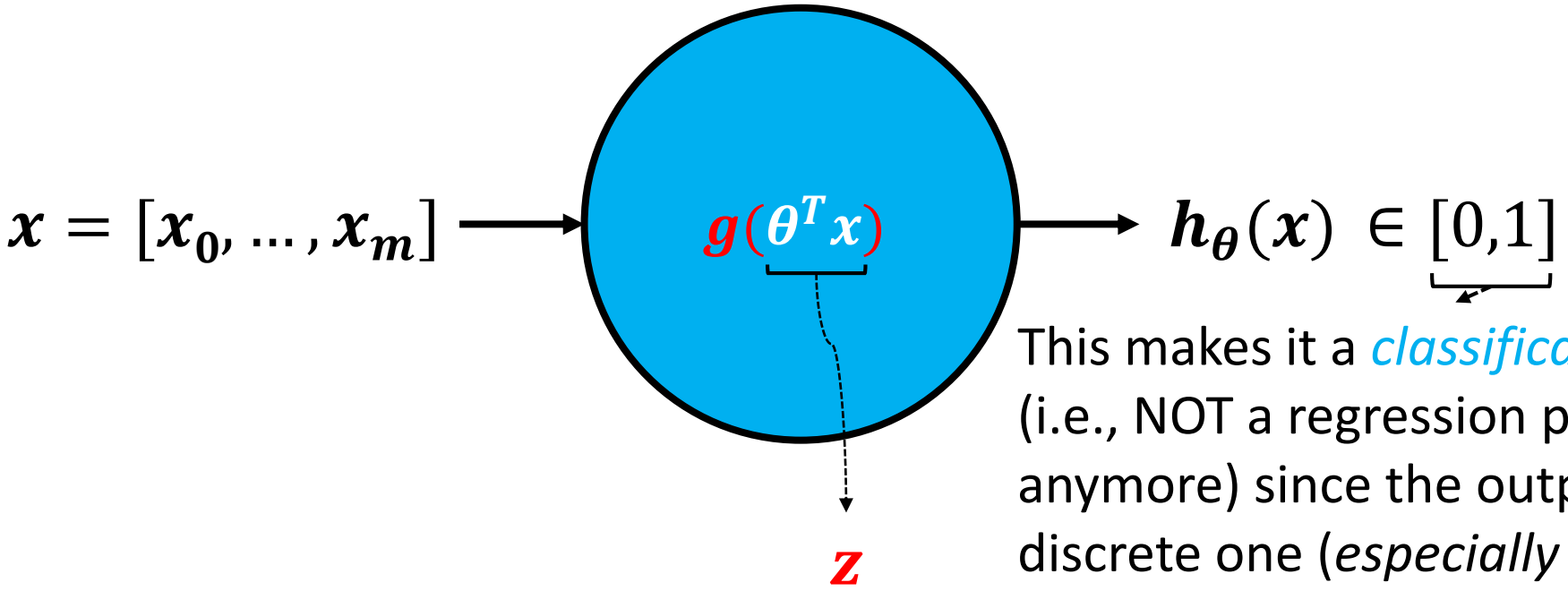
# Regression vs. Classification

- How can we make the possible outputs of $\boldsymbol{h_\theta(x) = \theta^T x}$ discrete-valued (as opposed to real-valued)?
  - By using an ***activation function*** (e.g., ***sigmoid or logistic function***)



# Regression vs. Classification

- How can we make the possible outputs of $\boldsymbol{h_\theta(x) = \theta^T x}$ discrete-valued (as opposed to real-valued)?
  - By using an ***activation function*** (e.g., ***sigmoid or logistic function***)




This makes it a *classification problem* (i.e., NOT a regression problem anymore) since the output renders a discrete one (*especially after applying thresholding*)

# So, What is Logistic Regression?

- Logistic regression is a machine learning algorithm that can be used to *classify* input data into discrete output (e.g., *input emails into spam or non-spam and tumour images into benign or malignant*)
  - **Note**: The word "regression" in the name does not mean that the algorithm is a regression algorithm (rather it is a classification algorithm)

- Major questions about logistic regression:
  - What is the ***hypothesis function*** (or ***model***)?
  - What is the ***cost function***?
  - How can we learn the parameters of the model?

# The Logistic Regression Model

- What will be the output of the model $\boldsymbol{h_\theta(x) = \theta^T x}$, where $\boldsymbol{\theta}$ = $\boldsymbol{[\theta_0, \dots, \theta_m]}$ and $\boldsymbol{x = [x_0, \dots, x_m]}$?
  - Real-valued

- How can we make the output of $\boldsymbol{h_\theta(x)}$ discrete?
  - By using the logistic function as follows:

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

***This is the logistic regression model or hypothesis function***

  - And then applying thresholding *after* learning the model to predict the output as follows:

$$\begin{cases} \boldsymbol{if\ h_\theta(x) < 0.5\ predict\ 0} \\ \boldsymbol{if\ h_\theta(x) \geq 0.5\ predict\ 1} \end{cases}$$

# Towards Identifying the Logistic Regression Cost Function

- How to learn a *logistic regression model* $\boldsymbol{h_\theta(x) = g(\theta^T x)}$, where $\boldsymbol{\theta} = [\boldsymbol{\theta_0}, \ldots, \boldsymbol{\theta_m}]$ and $\boldsymbol{x} = [\boldsymbol{x_0}, \ldots, \boldsymbol{x_m}]$?
  - Perhaps, by minimizing ***Mean Squared Error (MSE)***. That is:

$$\text{minimize} \frac{1}{2n} \sum_{i=1}^{n} \left( y'^{(i)} - y^{(i)} \right)^2$$

$$\equiv$$

$$\underset{\theta}{\text{minimize}} \frac{1}{2n} \sum_{i=1}^{n} \left( (g(\theta^T x))^{(i)} - y^{(i)} \right)^2$$

***objective function*** $\boldsymbol{J(\theta)}$

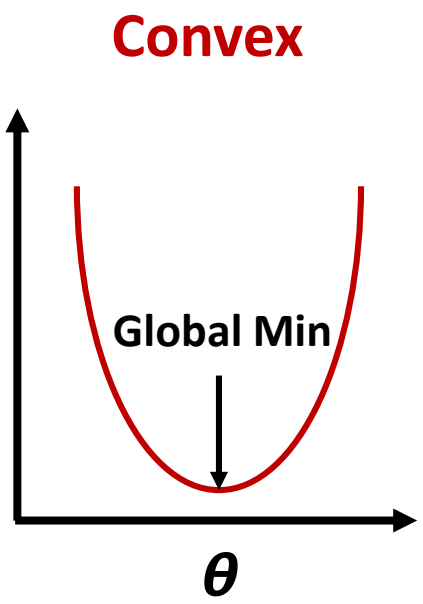
$$\equiv$$

$$\underset{\theta}{\text{minimize}}\, \boldsymbol{J(\theta)}$$

# Towards Identifying the Logistic Regression Cost Function

- How to learn a *logistic regression model* $\boldsymbol{h_\theta(x) = g(\theta^T x)}$, where $\boldsymbol{\theta} = [\boldsymbol{\theta_0}, \ldots, \boldsymbol{\theta_m}]$ and $\boldsymbol{x} = [\boldsymbol{x_0}, \ldots, \boldsymbol{x_m}]$?
  - Perhaps, by minimizing ***Mean Squared Error (MSE)***. That is:

**Easy for Gradient Descent to locate!**




$$\text{minimize} \frac{1}{2n} \sum_{i=1}^{n} \left( y'^{(i)} - y^{(i)} \right)^2$$

$$\equiv$$

$$\underset{\theta}{\text{minimize}} \frac{1}{2n} \sum_{i=1}^{n} \left( (g(\theta^T x))^{(i)} - y^{(i)} \right)^2$$

$$\equiv$$

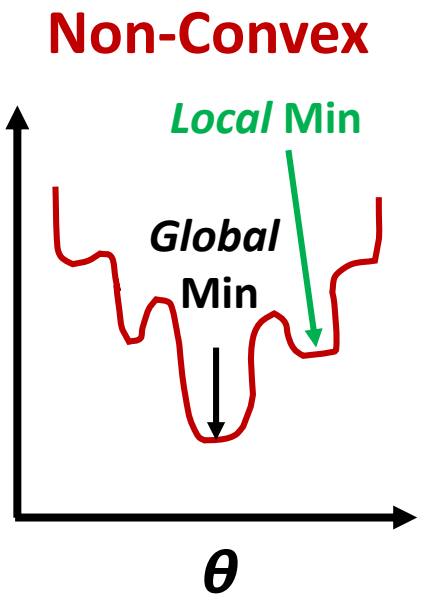
$$\underset{\theta}{\text{minimize}}\, J(\boldsymbol{\theta})$$

Unfortunately, if we plot this cost function, it will turn out to be "**non-convex**"

# Towards Identifying the Logistic Regression Cost Function

- How to learn a *logistic regression model* $\boldsymbol{h_\theta(x) = g(\theta^T x)}$, where $\boldsymbol{\theta} = [\boldsymbol{\theta_0}, \ldots, \boldsymbol{\theta_m}]$ and $\boldsymbol{x} = [\boldsymbol{x_0}, \ldots, \boldsymbol{x_m}]$?
  - Perhaps, by minimizing ***Mean Squared Error (MSE)***. That is:

**Gradient Descent might get stuck at a *local* min and fail to locate the *global* min!**




$$\text{minimize} \frac{1}{2n} \sum_{i=1}^{n} \left( y'^{(i)} - y^{(i)} \right)^2$$

$$\equiv$$

$$\underset{\theta}{\text{minimize}} \frac{1}{2n} \sum_{i=1}^{n} \left( (g(\theta^T x))^{(i)} - y^{(i)} \right)^2$$

$$\equiv$$

$$\underset{\theta}{\text{minimize}}\, J(\boldsymbol{\theta})$$

Unfortunately, if we plot this cost function, it will turn out to be "**non-convex**"

# The Logistic Regression Cost Function

- How to learn a *logistic regression model* $\boldsymbol{h_\theta(x) = g(\theta^T x)}$, where $\boldsymbol{\theta} = [\boldsymbol{\theta_0}, \ldots, \boldsymbol{\theta_m}]$ and $\boldsymbol{x} = [\boldsymbol{x_0}, \ldots, \boldsymbol{x_m}]$?
  - Let us try a different cost function. That is:

$$\text{Cost}(\boldsymbol{h_\theta(x)}, \text{y}) = \begin{cases} -\log\big(\boldsymbol{h_\theta(x)}\big) \; \boldsymbol{if} \; \boldsymbol{y = 1} \\ -\log\big(\boldsymbol{1 - \; h_\theta(x)}\big) \; \boldsymbol{if} \; \boldsymbol{y = 0} \end{cases}$$

*Equivalent To*

$$\text{Cost}(\boldsymbol{h_\theta(x)}, \text{y}) = -y \log\big(\boldsymbol{h_\theta(x)}\big) - (1 - y) \log\big(\boldsymbol{1 - \; h_\theta(x)}\big)$$

# The Logistic Regression Cost Function

- How to learn a *logistic regression model* $\boldsymbol{h_\theta(x) = g(\theta^T x)}$, where $\boldsymbol{\theta} = [\boldsymbol{\theta_0}, \ldots, \boldsymbol{\theta_m}]$ and $\boldsymbol{x} = [\boldsymbol{x_0}, \ldots, \boldsymbol{x_m}]$?
  - Let us try a different cost function. That is:

$$\text{Cost}(\boldsymbol{h_\theta(x)}, \text{y}) = \quad -y \log\big(\boldsymbol{h_\theta(x)}\big) - (1-y) \log\big(\mathbf{1} - \boldsymbol{h_\theta(x)}\big)$$

This function still assumes real-valued outputs for $\boldsymbol{h_\theta(x)}$ (i.e., still entails *a regression problem*), while logistic regression should predict discrete values (i.e., logistic regression is a *classification problem*)

# The Logistic Regression Cost Function

- How to learn a *logistic regression model* $\boldsymbol{h_\theta(x) = g(\theta^T x)}$, where $\boldsymbol{\theta} = [\boldsymbol{\theta_0}, \ldots, \boldsymbol{\theta_m}]$ and $\boldsymbol{x} = [\boldsymbol{x_0}, \ldots, \boldsymbol{x_m}]$?
  - Let us try a different cost function. That is:

$$\text{Cost}(\boldsymbol{h_\theta(x)}, \text{y}) = \; -y \log\big(\boldsymbol{h_\theta(x)}\big) - (1-y) \log\big(\boldsymbol{1 - h_\theta(x)}\big)$$

We still need to apply to it the logistic function:

$$\boldsymbol{g}(z) = \frac{1}{1 + e^{-z}}$$

# The Logistic Regression Cost Function

- How to learn a *logistic regression model* $\boldsymbol{h_\theta(x) = g(\theta^T x)}$, where $\boldsymbol{\theta} = [\boldsymbol{\theta_0}, \ldots, \boldsymbol{\theta_m}]$ and $\boldsymbol{x} = [\boldsymbol{x_0}, \ldots, \boldsymbol{x_m}]$?
  - By minimizing the following cost function:

$$\text{Cost}(\boldsymbol{h_\theta(x)}, \text{y}) = -y \log\left(\boldsymbol{h_\theta(x)}\right) - (1-y)\log\left(\boldsymbol{1} - \boldsymbol{h_\theta(x)}\right)$$

$$= -y \log\left(\boldsymbol{g(\theta^T x)}\right) - (1-y)\log\left(1 - \boldsymbol{g(\theta^T x)}\right)$$

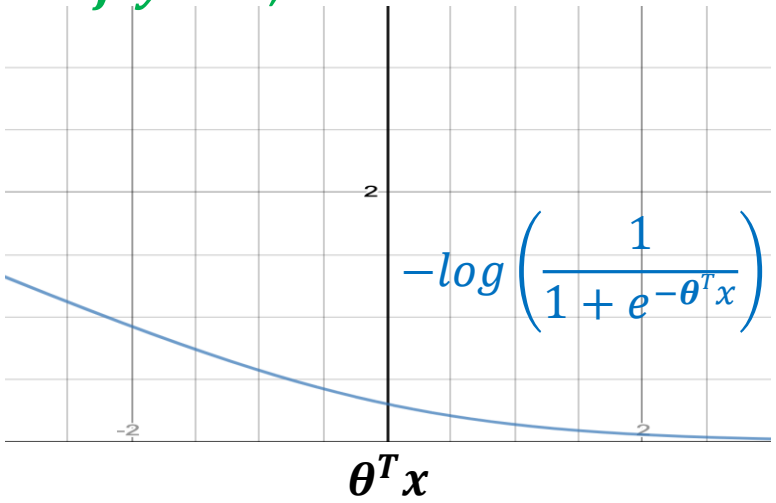
$$= -y \log\left(\frac{1}{1+e^{-\boldsymbol{\theta}^T x}}\right) - (1-y)\log\left(1 - \frac{1}{1+e^{-\boldsymbol{\theta}^T x}}\right)$$

# The Logistic Regression Cost Function

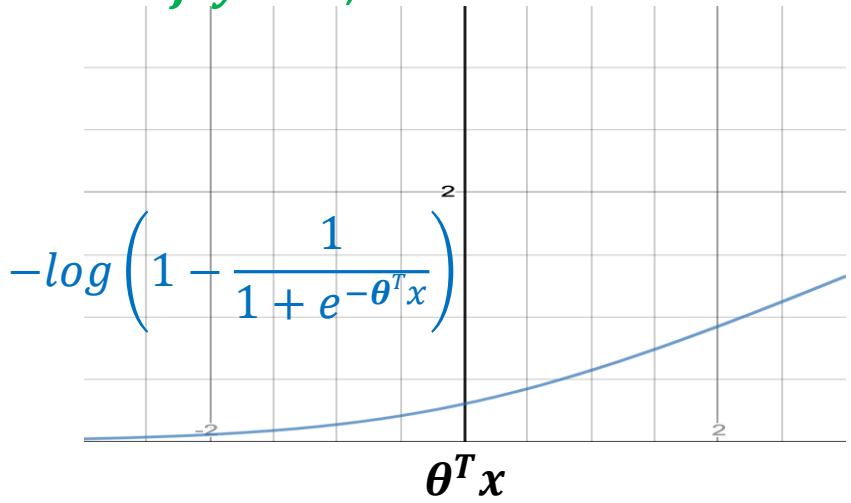
- How to learn a *logistic regression model* $\boldsymbol{h_\theta(x) = g(\theta^T x)}$, where $\boldsymbol{\theta} = [\boldsymbol{\theta_0}, \ldots, \boldsymbol{\theta_m}]$ and $\boldsymbol{x} = [\boldsymbol{x_0}, \ldots, \boldsymbol{x_m}]$?
  - By minimizing the following cost function:

$$\text{Cost}(\boldsymbol{h_\theta(x)}, \text{y}) = -y \log\left(\frac{1}{1+e^{-\boldsymbol{\theta}^T x}}\right) - (1-y)\log\left(1 - \frac{1}{1+e^{-\boldsymbol{\theta}^T x}}\right)$$

***If*** $\boldsymbol{y = 1}$, we want $\boldsymbol{\theta^T x >> 0}$




***If*** $\boldsymbol{y = 0}$, we want $\boldsymbol{\theta^T x << 0}$



# Learning a Logistic Regression Model

- How to learn a *logistic regression model* $\boldsymbol{h_\theta(x) = g(\theta^T x)}$, where $\boldsymbol{\theta} = [\boldsymbol{\theta_0}, \ldots, \boldsymbol{\theta_m}]$ and $\boldsymbol{x} = [\boldsymbol{x_0}, \ldots, \boldsymbol{x_m}]$?
  - By minimizing the following cost function:

$$\text{Cost}(\boldsymbol{h_\theta(x)}, \text{y}) = -y \log\left(\frac{1}{1+e^{-\boldsymbol{\theta}^T x}}\right) - (1-y)\log\left(1 - \frac{1}{1+e^{-\boldsymbol{\theta}^T x}}\right)$$

  - That is:

$$\underset{\theta}{\text{minimize}} \frac{1}{n}\sum_{i=1}^{n} \text{Cost}(\boldsymbol{h_\theta(x)}^{(i)}, y^{(i)})$$

$$\equiv$$

$$\underset{\theta}{\text{minimize}} \frac{1}{n}\sum_{i=1}^{n} -y^{(i)} \log\left(\frac{1}{1+e^{-\boldsymbol{\theta}^T x^{(i)}}}\right) - (1-y)\log\left(1 - \frac{1}{1+e^{-\boldsymbol{\theta}^T x^{(i)}}}\right)$$

***Cost function $J(\boldsymbol{\theta})$***

# Gradient Descent For Logistic Regression

- **Outline**:
  - Have the objective function $\boldsymbol{J(\theta)}$, where $\boldsymbol{\theta} = [\boldsymbol{\theta_0}, \ldots, \boldsymbol{\theta_m}]$
  - Start off with some guesses for $\theta_0, \ldots, \theta_m$
    - It does not really matter what values you start off with, but a common choice is to set them all initially to zero
  - Repeat until convergence{

$$\theta_j = \theta_j - \alpha \frac{\partial \boldsymbol{J(\theta)}}{\partial\, \theta_j}$$

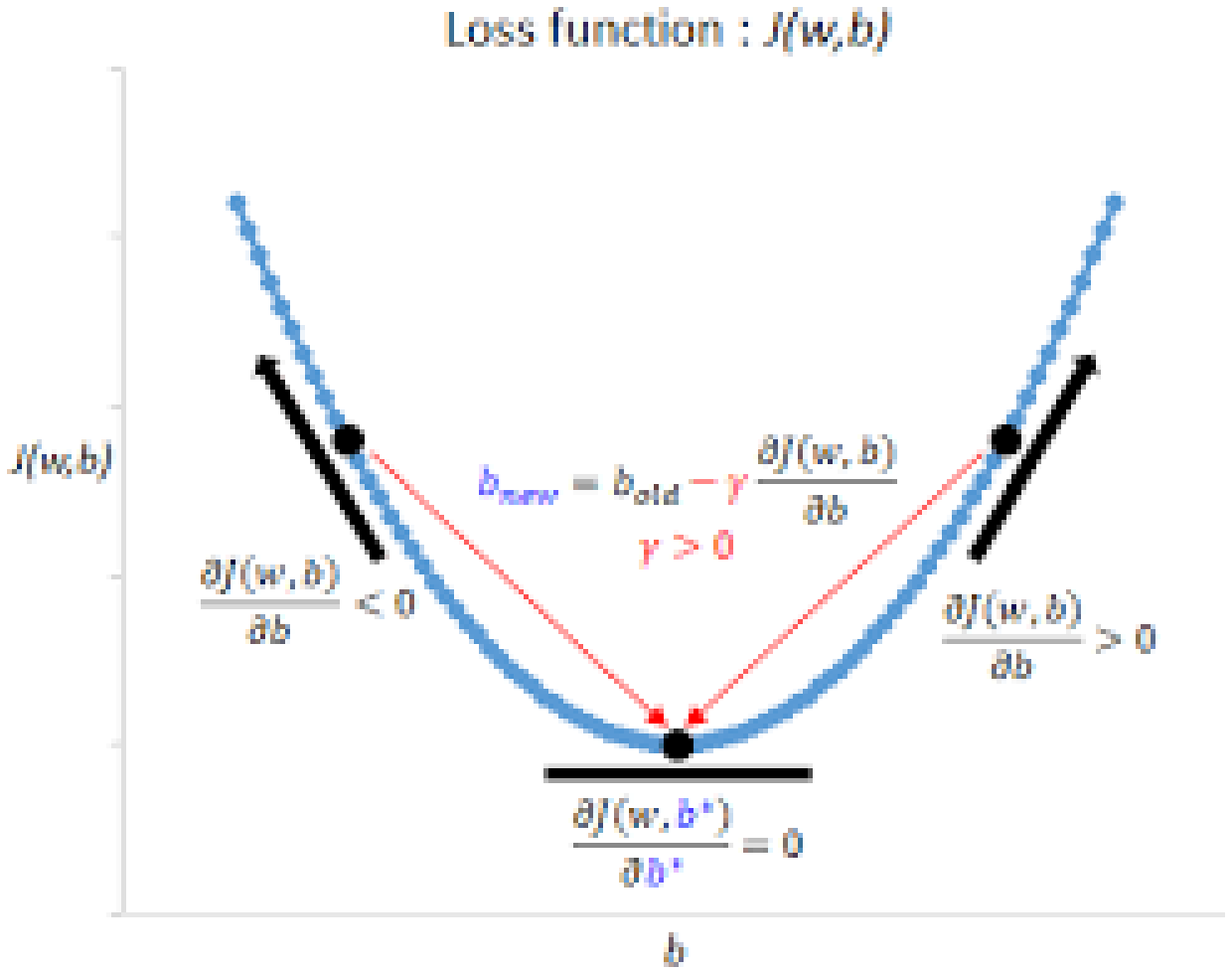
***Partial derivative***

**Note: Update all $\theta_j$ simulatenously**

***Learing rate**, which controls how big a step we take when we update $\theta_j$*

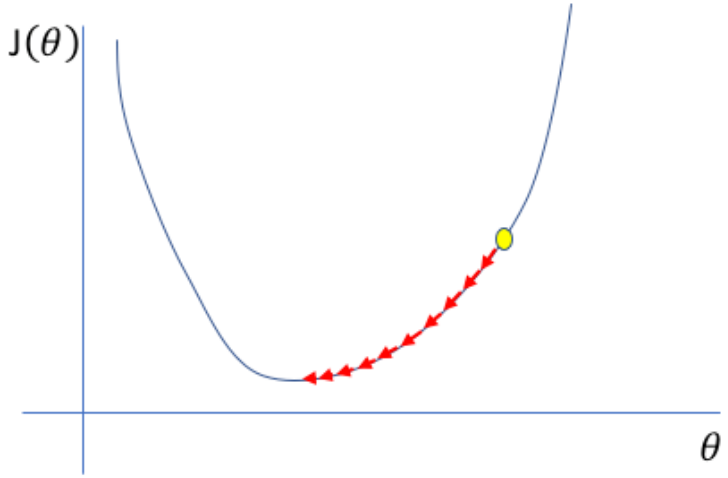
}

# Gradient Descent



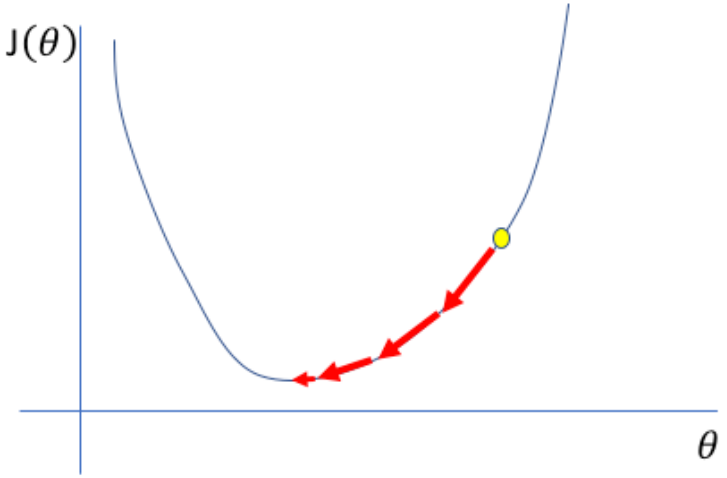
# Gradient Descent

**Too low**



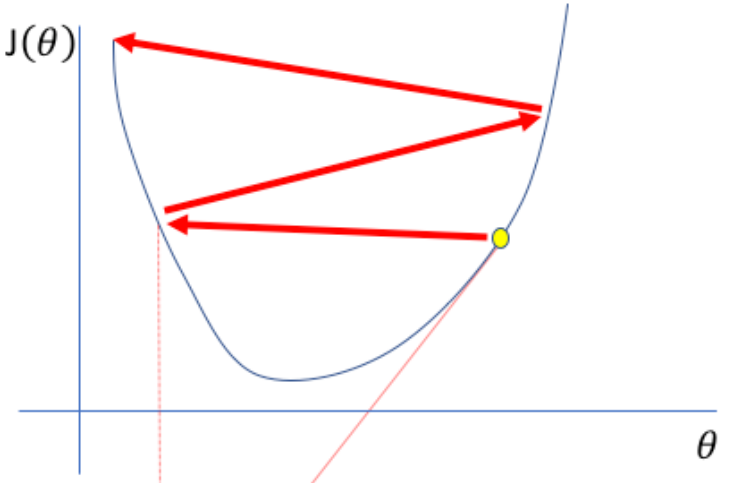

A small learning rate requires many updates before reaching the minimum point

**Just right**




The optimal learning rate swiftly reaches the minimum point

**Too high**




Too large of a learning rate causes drastic updates which lead to divergent behaviors

# Gradient Descent For Logistic Regression

- **Outline**:
  - Have cost function $\boldsymbol{J(\theta)}$, where $\boldsymbol{\theta} = [\boldsymbol{\theta_0}, \ldots, \boldsymbol{\theta_m}]$
  - Start off with some guesses for $\theta_0, \ldots, \theta_m$
    - It does not really matter what values you start off with, but a common choice is to set them all initially to zero
  - Repeat until convergence{

$$\theta_j = \theta_j - \alpha \frac{1}{n} \sum_{i=1}^{n} \left( \frac{1}{1 + e^{-\boldsymbol{\theta}^T x^{(i)}}} - y^{(i)} \right) x_j{}^{(i)}$$

***The final formula after applying partial derivatives***

}

# Inference After Learning

- After learning the parameters $\boldsymbol{\theta} = [\theta_0, \dots, \theta_m]$, we can predict the output of any new unseen $\boldsymbol{x} = [\boldsymbol{x_0}, \dots, \boldsymbol{x_m}]$ as follows:

$$\begin{cases} \boldsymbol{if}\ \boldsymbol{h_\theta(x)} = \dfrac{1}{1 + e^{-\boldsymbol{\theta}^T \boldsymbol{x}}} < \mathbf{0.5}\ \mathbf{predict\ 0} \\ \boldsymbol{Else\ if}\ \boldsymbol{h_\theta(x)} = \dfrac{1}{1 + e^{-\boldsymbol{\theta}^T \boldsymbol{x}}} \geq \mathbf{0.5}\ \mathbf{predict\ 1} \end{cases}$$

# A Concrete Example: The Training Phase

- Let us apply logistic regression on the spam email recognition problem, assuming $\alpha$ = 0.5 and starting with $\theta$ = [0, 0, 0, 0, 0, 0]

| | and | vaccine | the | of | nigeria | y |
|---|---|---|---|---|---|---|
| Email **a** | 1 | 1 | 0 | 1 | 1 | 1 |
| Email **b** | 0 | 0 | 1 | 1 | 0 | 0 |
| Email **c** | 0 | 1 | 1 | 0 | 0 | 1 |
| Email **d** | 1 | 0 | 0 | 1 | 0 | 0 |
| Email **e** | 1 | 0 | 1 | 0 | 1 | 1 |
| Email **f** | 1 | 0 | 1 | 1 | 0 | 0 |

**A Training Dataset**

# A Concrete Example: The Training Phase

- Let us apply logistic regression on the spam email recognition problem, assuming $\boldsymbol{\alpha}$ = 0.5 and starting with $\boldsymbol{\theta}$ = [0, 0, 0, 0, 0, 0]

| | and | vaccine | the | of | nigeria | y |
|---|---|---|---|---|---|---|
| Email **a** | 1 | 1 | 0 | 1 | 1 | 1 |
| Email **b** | 0 | 0 | 1 | 1 | 0 | 0 |
| Email **c** | 0 | 1 | 1 | 0 | 0 | 1 |
| Email **d** | 1 | 0 | 0 | 1 | 0 | 0 |
| Email **e** | 1 | 0 | 1 | 0 | 1 | 1 |
| Email **f** | 1 | 0 | 1 | 1 | 0 | 0 |

**1** entails that a word (i.e., “and”) is *present* in an email (i.e., “Email **a**”)

# A Concrete Example: The Training Phase

- Let us apply logistic regression on the spam email recognition problem, assuming $\boldsymbol{\alpha}$ = 0.5 and starting with $\boldsymbol{\theta}$ = [0, 0, 0, 0, 0, 0]

| | and | vaccine | the | of | nigeria | y |
|---|---|---|---|---|---|---|
| Email **a** | 1 | 1 | 0 | 1 | 1 | 1 |
| Email **b** | 0 | 0 | 1 | 1 | 0 | 0 |
| Email **c** | 0 | 1 | 1 | 0 | 0 | 1 |
| Email **d** | 1 | 0 | 0 | 1 | 0 | 0 |
| Email **e** | 1 | 0 | 1 | 0 | 1 | 1 |
| Email **f** | 1 | 0 | 1 | 1 | 0 | 0 |

**0** entails that a word (i.e., “and”) is *absent* in an email (i.e., “Email **b**”)

# A Concrete Example: The Training Phase

- Let us apply logistic regression on the spam email recognition problem, assuming $\boldsymbol{\alpha}$ = 0.5 and starting with $\boldsymbol{\theta}$ = [0, 0, 0, 0, 0, 0]

We define 6 parameters (the first one, i.e., $\theta_0$, is the intercept)

5 words (or ***features***) = $[\boldsymbol{x_1}, \boldsymbol{x_2}, \boldsymbol{x_3}, \boldsymbol{x_4}, \boldsymbol{x_5}]$

| | $x_1$ = and | $x_2$ = vaccine | $x_3$ = the | $x_4$ = of | $x_5$ = nigeria | y |
|---|---|---|---|---|---|---|
| Email **a** | 1 | 1 | 0 | 1 | 1 | 1 |
| Email **b** | 0 | 0 | 1 | 1 | 0 | 0 |
| Email **c** | 0 | 1 | 1 | 0 | 0 | 1 |
| Email **d** | 1 | 0 | 0 | 1 | 0 | 0 |
| Email **e** | 1 | 0 | 1 | 0 | 1 | 1 |
| Email **f** | 1 | 0 | 1 | 1 | 0 | 0 |

# A Concrete Example: The Training Phase

- Let us apply logistic regression on the spam email recognition problem, assuming $\alpha$ = 0.5 and starting with $\theta$ = [0, 0, 0, 0, 0, 0]

*The parameter vector:*
$\theta = [\theta_0, \theta_1, \theta_2, \theta_3, \theta_4, \theta_5]$

$x = [x_0, x_1, x_2, x_3, x_4, x_5]$ → *The feature vector*

| | $x_0 = 1$ | $x_1$ = and | $x_2$ = vaccine | $x_3$ = the | $x_4$ = of | $x_5$ = nigeria | y |
|---|---|---|---|---|---|---|---|
| Email **a** | 1 | 1 | 1 | 0 | 1 | 1 | 1 |
| Email **b** | 1 | 0 | 0 | 1 | 1 | 0 | 0 |
| Email **c** | 1 | 0 | 1 | 1 | 0 | 0 | 1 |
| Email **d** | 1 | 1 | 0 | 0 | 1 | 0 | 0 |
| Email **e** | 1 | 1 | 0 | 1 | 0 | 1 | 1 |
| Email **f** | 1 | 1 | 0 | 1 | 1 | 0 | 0 |

To account for the intercept

# Recap: Gradient Descent For Logistic Regression

- **Outline**:
  - Have cost function $\boldsymbol{J(\theta)}$, where $\boldsymbol{\theta} = [\boldsymbol{\theta_0}, \ldots, \boldsymbol{\theta_m}]$
  - Start off with some guesses for $\theta_0, \ldots, \theta_m$
    - It does not really matter what values you start off with, but a common choice is to set them all initially to zero
  - Repeat until convergence{

$$\theta_j = \theta_j - \alpha \frac{1}{n} \sum_{i=1}^{n} \left( \frac{1}{1 + e^{-\boldsymbol{\theta}^T x^{(i)}}} - y^{(i)} \right) x_j{}^{(i)}$$

}

**First, let us calculate this factor for every example in our training dataset**

# A Concrete Example: The Training Phase

- Let us apply logistic regression on the spam email recognition problem, assuming $\alpha$ = 0.5 and starting with $\theta$ = [0, 0, 0, 0, 0, 0]

| $x$ | $y$ | $\theta^T x$ |
|---|---|---|
| [1,1,1,0,1,1] | **1** | [0,0,0,0,0,0]×[1,1,1,0,1,1]=0 |
| [1,0,0,1,1,0] | **0** | [0,0,0,0,0,0]×[1,0,0,1,1,0]=0 |
| [1,0,1,1,0,0] | **1** | [0,0,0,0,0,0]×[1,0,1,1,0,0]=0 |
| [1,1,0,0,1,0] | **0** | [0,0,0,0,0,0]×[1,1,0,0,1,0]=0 |
| [1,1,0,1,0,1] | **1** | [0,0,0,0,0,0]×[1,1,0,1,0,1]=0 |
| [1,1,0,1,1,0] | **0** | [0,0,0,0,0,0]×[1,1,0,1,1,0]=0 |

# Recap: Gradient Descent For Logistic Regression

- **Outline**:
  - Have cost function $\boldsymbol{J(\theta)}$, where $\boldsymbol{\theta} = [\boldsymbol{\theta_0}, \dots, \boldsymbol{\theta_m}]$
  - Start off with some guesses for $\theta_0, \dots, \theta_m$
    - It does not really matter what values you start off with, but a common choice is to set them all initially to zero
  - Repeat until convergence{

$$\theta_j = \theta_j - \alpha \frac{1}{n} \sum_{i=1}^{n} \left( \frac{1}{1 + e^{-\boldsymbol{\theta}^T x^{(i)}}} - y^{(i)} \right) x_j{}^{(i)}$$

}

**Second, let us calculate this equation for every example in our training dataset and for every $\boldsymbol{\theta_j}$, where $\boldsymbol{j}$ is between $\boldsymbol{0}$ and $\boldsymbol{m}$**

# A Concrete Example: The Training Phase

- Let us apply logistic regression on the spam email recognition problem, assuming $\boldsymbol{\alpha}$ = 0.5 and starting with $\boldsymbol{\theta}$ = [0, 0, 0, 0, 0, 0]

| $x$ | $y$ | $\theta^T x$ | $(\frac{1}{1+e^{-\theta^T x}} - y)x_0$ |
|---|---|---|---|
| [1,1,1,0,1,1] | **1** | [0,0,0,0,0,0]×[1,1,1,0,1,1]=0 | |
| [1,0,0,1,1,0] | **0** | [0,0,0,0,0,0]×[1,0,0,1,1,0]=0 | |
| [1,0,1,1,0,0] | **1** | [0,0,0,0,0,0]×[1,0,1,1,0,0]=0 | |
| [1,1,0,0,1,0] | **0** | [0,0,0,0,0,0]×[1,1,0,0,1,0]=0 | |
| [1,1,0,1,0,1] | **1** | [0,0,0,0,0,0]×[1,1,0,1,0,1]=0 | |
| [1,1,0,1,1,0] | **0** | [0,0,0,0,0,0]×[1,1,0,1,1,0]=0 | |

# A Concrete Example: The Training Phase

- Let us apply logistic regression on the spam email recognition problem, assuming $\boldsymbol{\alpha}$ = 0.5 and starting with $\boldsymbol{\theta}$ = [0, 0, 0, 0, 0, 0]

| $\boldsymbol{x}$ | $\boldsymbol{y}$ | $\boldsymbol{\theta^T x}$ | $(\frac{1}{1+e^{-\theta^T x}} - \boldsymbol{y})\boldsymbol{x_0}$ |
|---|---|---|---|
| [1,1,1,0,1,1] | **1** | [0,0,0,0,0,0]×[1,1,1,0,1,1]=0 | $(\frac{1}{1+e^{-\mathbf{0}}} - \mathbf{1}) \times \mathbf{1}$ = -0.5 |
| [1,0,0,1,1,0] | **0** | [0,0,0,0,0,0]×[1,0,0,1,1,0]=0 | $(\frac{1}{1+1} - \mathbf{0}) \times \mathbf{1}$ = 0.5 |
| [1,0,1,1,0,0] | **1** | [0,0,0,0,0,0]×[1,0,1,1,0,0]=0 | $(\frac{1}{1+1} - \mathbf{1}) \times \mathbf{1}$ = -0.5 |
| [1,1,0,0,1,0] | **0** | [0,0,0,0,0,0]×[1,1,0,0,1,0]=0 | $(\frac{1}{1+1} - \mathbf{0}) \times \mathbf{1}$ = 0.5 |
| [1,1,0,1,0,1] | **1** | [0,0,0,0,0,0]×[1,1,0,1,0,1]=0 | $(\frac{1}{1+1} - \mathbf{1}) \times \mathbf{1}$ = -0.5 |
| [1,1,0,1,1,0] | **0** | [0,0,0,0,0,0]×[1,1,0,1,1,0]=0 | $(\frac{1}{1+1} - \mathbf{0}) \times \mathbf{1}$ = 0.5 |

# Recap: Gradient Descent For Logistic Regression

- **Outline**:
  - Have cost function $\boldsymbol{J(\theta)}$, where $\boldsymbol{\theta} = [\boldsymbol{\theta_0}, \ldots, \boldsymbol{\theta_m}]$
  - Start off with some guesses for $\theta_0, \ldots, \theta_m$
    - It does not really matter what values you start off with, but a common choice is to set them all initially to zero
  - Repeat until convergence{

$$\theta_j = \theta_j - \alpha \frac{1}{n} \sum_{i=1}^{n} \left( \frac{1}{1 + e^{-\boldsymbol{\theta}^T x^{(i)}}} - y^{(i)} \right) x_j{}^{(i)}$$

Third, let us compute every $\boldsymbol{\theta_j}$

}

# A Concrete Example: The Training Phase

- Let us apply logistic regression on the spam email recognition problem, assuming $\boldsymbol{\alpha}$ = 0.5 and starting with $\boldsymbol{\theta}$ = [0, 0, 0, 0, 0, 0]

| $x$ | $y$ | $\boldsymbol{\theta^T x}$ | $(\frac{1}{1+e^{-\theta^T x}} - y)x_0$ |
|---|---|---|---|
| [1,1,1,0,1,1] | **1** | [0,0,0,0,0,0]×[1,1,1,0,1,1]=0 | -0.5 |
| [1,0,0,1,1,0] | **0** | [0,0,0,0,0,0]×[1,0,0,1,1,0]=0 | 0.5 |
| [1,0,1,1,0,0] | **1** | [0,0,0,0,0,0]×[1,0,1,1,0,0]=0 | -0.5 |
| [1,1,0,0,1,0] | **0** | [0,0,0,0,0,0]×[1,1,0,0,1,0]=0 | 0.5 |
| [1,1,0,1,0,1] | **1** | [0,0,0,0,0,0]×[1,1,0,1,0,1]=0 | -0.5 |
| [1,1,0,1,1,0] | **0** | [0,0,0,0,0,0]×[1,1,0,1,1,0]=0 | 0.5 |

$$\frac{1}{n}\sum_{i=1}^{n}\left(\frac{1}{1+e^{-\boldsymbol{\theta}^T x^{(i)}}} - y^{(i)}\right)x_0{}^{(i)} = \mathbf{0}$$

***Then,***

$$\theta_0 = \theta_0 - \alpha \times \mathbf{0}$$

***New $\theta_0$***

# A Concrete Example: The Training Phase

- Let us apply logistic regression on the spam email recognition problem, assuming $\boldsymbol{\alpha}$ = 0.5 and starting with $\boldsymbol{\theta}$ = [0, 0, 0, 0, 0, 0]

| $x$ | $y$ | $\boldsymbol{\theta^T x}$ | $(\frac{1}{1+e^{-\theta^T x}} - y)x_0$ |
|---|---|---|---|
| [1,1,1,0,1,1] | **1** | [0,0,0,0,0,0]×[1,1,1,0,1,1]=0 | -0.5 |
| [1,0,0,1,1,0] | **0** | [0,0,0,0,0,0]×[1,0,0,1,1,0]=0 | 0.5 |
| [1,0,1,1,0,0] | **1** | [0,0,0,0,0,0]×[1,0,1,1,0,0]=0 | -0.5 |
| [1,1,0,0,1,0] | **0** | [0,0,0,0,0,0]×[1,1,0,0,1,0]=0 | 0.5 |
| [1,1,0,1,0,1] | **1** | [0,0,0,0,0,0]×[1,1,0,1,0,1]=0 | -0.5 |
| [1,1,0,1,1,0] | **0** | [0,0,0,0,0,0]×[1,1,0,1,1,0]=0 | 0.5 |

$$\frac{1}{n}\sum_{i=1}^{n}\left(\frac{1}{1+e^{-\boldsymbol{\theta}^T x^{(i)}}} - y^{(i)}\right)x_0{}^{(i)} = \mathbf{0}$$

***Then,***

$$\theta_0 = \theta_0 - \alpha \times \mathbf{0}$$

***Old*** $\boldsymbol{\theta_0}$

# A Concrete Example: The Training Phase

- Let us apply logistic regression on the spam email recognition problem, assuming $\boldsymbol{\alpha}$ = 0.5 and starting with $\boldsymbol{\theta}$ = [0, 0, 0, 0, 0, 0]

| $\boldsymbol{x}$ | $\boldsymbol{y}$ | $\boldsymbol{\theta^T x}$ | $(\frac{1}{1+e^{-\theta^T x}} - \boldsymbol{y})\boldsymbol{x_0}$ |
|---|---|---|---|
| [1,1,1,0,1,1] | **1** | [0,0,0,0,0,0]×[1,1,1,0,1,1]=0 | -0.5 |
| [1,0,0,1,1,0] | **0** | [0,0,0,0,0,0]×[1,0,0,1,1,0]=0 | 0.5 |
| [1,0,1,1,0,0] | **1** | [0,0,0,0,0,0]×[1,0,1,1,0,0]=0 | -0.5 |
| [1,1,0,0,1,0] | **0** | [0,0,0,0,0,0]×[1,1,0,0,1,0]=0 | 0.5 |
| [1,1,0,1,0,1] | **1** | [0,0,0,0,0,0]×[1,1,0,1,0,1]=0 | -0.5 |
| [1,1,0,1,1,0] | **0** | [0,0,0,0,0,0]×[1,1,0,1,1,0]=0 | 0.5 |

$$\frac{1}{n}\sum_{i=1}^{n}\left(\frac{1}{1+e^{-\boldsymbol{\theta}^T x^{(i)}}} - y^{(i)}\right)x_0{}^{(i)} = \mathbf{0}$$

***Then,***

$$\theta_0 = \theta_0 - \alpha \times \mathbf{0}$$

$$= 0 - 0.5 \times \mathbf{0} = \mathbf{0}$$

**New Parameter Vector**:

$$\boldsymbol{\theta} = [\mathbf{0}, \boldsymbol{\theta_1}, \boldsymbol{\theta_2}, \boldsymbol{\theta_3}, \boldsymbol{\theta_4}, \boldsymbol{\theta_5}]$$

# A Concrete Example: The Training Phase

- Let us apply logistic regression on the spam email recognition problem, assuming $\boldsymbol{\alpha}$ = 0.5 and starting with $\boldsymbol{\theta}$ = [0, 0, 0, 0, 0, 0]

| $x$ | $y$ | $\theta^T x$ | $(\frac{1}{1+e^{-\theta^T x}} - y)x_1$ |
|---|---|---|---|
| [1,1,1,0,1,1] | **1** | [0,0,0,0,0,0]×[1,1,1,0,1,1]=0 | |
| [1,0,0,1,1,0] | **0** | [0,0,0,0,0,0]×[1,0,0,1,1,0]=0 | |
| [1,0,1,1,0,0] | **1** | [0,0,0,0,0,0]×[1,0,1,1,0,0]=0 | |
| [1,1,0,0,1,0] | **0** | [0,0,0,0,0,0]×[1,1,0,0,1,0]=0 | |
| [1,1,0,1,0,1] | **1** | [0,0,0,0,0,0]×[1,1,0,1,0,1]=0 | |
| [1,1,0,1,1,0] | **0** | [0,0,0,0,0,0]×[1,1,0,1,1,0]=0 | |

# A Concrete Example: The Training Phase

- Let us apply logistic regression on the spam email recognition problem, assuming $\boldsymbol{\alpha}$ = 0.5 and starting with $\boldsymbol{\theta}$ = [0, 0, 0, 0, 0, 0]

| $x$ | $y$ | $\boldsymbol{\theta^T x}$ | $(\frac{1}{1+e^{-\theta^T x}} - y)x_1$ |
|---|---|---|---|
| [1,1,1,0,1,1] | **1** | [0,0,0,0,0,0]×[1,1,1,0,1,1]=0 | -0.5 |
| [1,0,0,1,1,0] | **0** | [0,0,0,0,0,0]×[1,0,0,1,1,0]=0 | 0 |
| [1,0,1,1,0,0] | **1** | [0,0,0,0,0,0]×[1,0,1,1,0,0]=0 | 0 |
| [1,1,0,0,1,0] | **0** | [0,0,0,0,0,0]×[1,1,0,0,1,0]=0 | 0.5 |
| [1,1,0,1,0,1] | **1** | [0,0,0,0,0,0]×[1,1,0,1,0,1]=0 | -0.5 |
| [1,1,0,1,1,0] | **0** | [0,0,0,0,0,0]×[1,1,0,1,1,0]=0 | 0.5 |

$$\frac{1}{n}\sum_{i=1}^{n}\left(\frac{1}{1+e^{-\boldsymbol{\theta}^T x^{(i)}}} - y^{(i)}\right){x_1}^{(i)} = \mathbf{0}$$

***Then,***

$$\theta_1 = \theta_1 - \alpha \times \mathbf{0}$$

$$= 0 - 0.5 \times \mathbf{0} = \mathbf{0}$$

**New Paramter Vector**:

$$\boldsymbol{\theta} = [\mathbf{0}, \mathbf{0}, \boldsymbol{\theta_2}, \boldsymbol{\theta_3}, \boldsymbol{\theta_4}, \boldsymbol{\theta_5}]$$

# A Concrete Example: The Training Phase

- Let us apply logistic regression on the spam email recognition problem, assuming $\boldsymbol{\alpha}$ = 0.5 and starting with $\boldsymbol{\theta}$ = [0, 0, 0, 0, 0, 0]

| $x$ | $y$ | $\boldsymbol{\theta}^T \boldsymbol{x}$ | $(\frac{1}{1+e^{-\theta^T x}} - y)x_2$ |
|---|---|---|---|
| [1,1,1,0,1,1] | **1** | [0,0,0,0,0,0]×[1,1,1,0,1,1]=0 | |
| [1,0,0,1,1,0] | **0** | [0,0,0,0,0,0]×[1,0,0,1,1,0]=0 | |
| [1,0,1,1,0,0] | **1** | [0,0,0,0,0,0]×[1,0,1,1,0,0]=0 | |
| [1,1,0,0,1,0] | **0** | [0,0,0,0,0,0]×[1,1,0,0,1,0]=0 | |
| [1,1,0,1,0,1] | **1** | [0,0,0,0,0,0]×[1,1,0,1,0,1]=0 | |
| [1,1,0,1,1,0] | **0** | [0,0,0,0,0,0]×[1,1,0,1,1,0]=0 | |

# A Concrete Example: The Training Phase

- Let us apply logistic regression on the spam email recognition problem, assuming $\boldsymbol{\alpha}$ = 0.5 and starting with $\boldsymbol{\theta}$ = [0, 0, 0, 0, 0, 0]

| $x$ | $y$ | $\boldsymbol{\theta^T x}$ | $(\frac{1}{1+e^{-\theta^T x}} - y)x_2$ |
|---|---|---|---|
| [1,1,1,0,1,1] | **1** | [0,0,0,0,0,0]×[1,1,1,0,1,1]=0 | -0.5 |
| [1,0,0,1,1,0] | **0** | [0,0,0,0,0,0]×[1,0,0,1,1,0]=0 | 0 |
| [1,0,1,1,0,0] | **1** | [0,0,0,0,0,0]×[1,0,1,1,0,0]=0 | -0.5 |
| [1,1,0,0,1,0] | **0** | [0,0,0,0,0,0]×[1,1,0,0,1,0]=0 | 0 |
| [1,1,0,1,0,1] | **1** | [0,0,0,0,0,0]×[1,1,0,1,0,1]=0 | 0 |
| [1,1,0,1,1,0] | **0** | [0,0,0,0,0,0]×[1,1,0,1,1,0]=0 | 0 |

$$\frac{1}{n}\sum_{i=1}^{n}\left(\frac{1}{1+e^{-\boldsymbol{\theta}^T x^{(i)}}} - y^{(i)}\right)x_2{}^{(i)} = \mathbf{-1}$$

***Then,***

$$\theta_2 = \theta_2 - \alpha \times (\mathbf{-1})$$

$$= 0 - 0.5 \times (\mathbf{-1})/6 = \mathbf{0.5/6}$$

**New Paramter Vector**:

$$\boldsymbol{\theta} = [\mathbf{0}, \mathbf{0}, \mathbf{0.5/6}, \boldsymbol{\theta_3}, \boldsymbol{\theta_4}, \boldsymbol{\theta_5}]$$

# A Concrete Example: The Training Phase

- Let us apply logistic regression on the spam email recognition problem, assuming $\boldsymbol{\alpha}$ = 0.5 and starting with $\boldsymbol{\theta}$ = [0, 0, 0, 0, 0, 0]

| $x$ | $y$ | $\boldsymbol{\theta}^T \boldsymbol{x}$ | $(\frac{1}{1+e^{-\theta^T x}} - y)x_3$ |
|---|---|---|---|
| [1,1,1,0,1,1] | **1** | [0,0,0,0,0,0]×[1,1,1,0,1,1]=0 | |
| [1,0,0,1,1,0] | **0** | [0,0,0,0,0,0]×[1,0,0,1,1,0]=0 | |
| [1,0,1,1,0,0] | **1** | [0,0,0,0,0,0]×[1,0,1,1,0,0]=0 | |
| [1,1,0,0,1,0] | **0** | [0,0,0,0,0,0]×[1,1,0,0,1,0]=0 | |
| [1,1,0,1,0,1] | **1** | [0,0,0,0,0,0]×[1,1,0,1,0,1]=0 | |
| [1,1,0,1,1,0] | **0** | [0,0,0,0,0,0]×[1,1,0,1,1,0]=0 | |

# A Concrete Example: The Training Phase

- Let us apply logistic regression on the spam email recognition problem, assuming $\boldsymbol{\alpha}$ = 0.5 and starting with $\boldsymbol{\theta}$ = [0, 0, 0, 0, 0, 0]

| $x$ | $y$ | $\boldsymbol{\theta^T x}$ | $(\frac{1}{1+e^{-\theta^T x}} - y)x_3$ |
|---|---|---|---|
| [1,1,1,0,1,1] | **1** | [0,0,0,0,0,0]×[1,1,1,0,1,1]=0 | 0 |
| [1,0,0,1,1,0] | **0** | [0,0,0,0,0,0]×[1,0,0,1,1,0]=0 | 0.5 |
| [1,0,1,1,0,0] | **1** | [0,0,0,0,0,0]×[1,0,1,1,0,0]=0 | -0.5 |
| [1,1,0,0,1,0] | **0** | [0,0,0,0,0,0]×[1,1,0,0,1,0]=0 | 0 |
| [1,1,0,1,0,1] | **1** | [0,0,0,0,0,0]×[1,1,0,1,0,1]=0 | -0.5 |
| [1,1,0,1,1,0] | **0** | [0,0,0,0,0,0]×[1,1,0,1,1,0]=0 | 0.5 |

$$\frac{1}{n}\sum_{i=1}^{n}\left(\frac{1}{1+e^{-\boldsymbol{\theta}^T x^{(i)}}} - y^{(i)}\right)x_3{}^{(i)} = \mathbf{0}$$

***Then,***

$$\theta_3 = \theta_3 - \alpha \times \mathbf{0}$$

$$= 0 - 0.5 \times 0 = \mathbf{0}$$

**New Paramter Vector**:

$$\boldsymbol{\theta} = [\mathbf{0}, \mathbf{0}, \mathbf{0.1}, \mathbf{0}, \boldsymbol{\theta_4}, \boldsymbol{\theta_5}]$$

# A Concrete Example: The Training Phase

- Let us apply logistic regression on the spam email recognition problem, assuming $\boldsymbol{\alpha}$ = 0.5 and starting with $\boldsymbol{\theta}$ = [0, 0, 0, 0, 0, 0]

| $x$ | $y$ | $\boldsymbol{\theta}^T \boldsymbol{x}$ | $(\frac{1}{1+e^{-\theta^T x}} - y)x_4$ |
|---|---|---|---|
| [1,1,1,0,1,1] | **1** | [0,0,0,0,0,0]×[1,1,1,0,1,1]=0 | |
| [1,0,0,1,1,0] | **0** | [0,0,0,0,0,0]×[1,0,0,1,1,0]=0 | |
| [1,0,1,1,0,0] | **1** | [0,0,0,0,0,0]×[1,0,1,1,0,0]=0 | |
| [1,1,0,0,1,0] | **0** | [0,0,0,0,0,0]×[1,1,0,0,1,0]=0 | |
| [1,1,0,1,0,1] | **1** | [0,0,0,0,0,0]×[1,1,0,1,0,1]=0 | |
| [1,1,0,1,1,0] | **0** | [0,0,0,0,0,0]×[1,1,0,1,1,0]=0 | |

# A Concrete Example: The Training Phase

- Let us apply logistic regression on the spam email recognition problem, assuming $\boldsymbol{\alpha}$ = 0.5 and starting with $\boldsymbol{\theta}$ = [0, 0, 0, 0, 0, 0]

| $x$ | $y$ | $\boldsymbol{\theta^T x}$ | $(\frac{1}{1+e^{-\theta^T x}} - y)x_4$ |
|---|---|---|---|
| [1,1,1,0,1,1] | **1** | [0,0,0,0,0,0]×[1,1,1,0,1,1]=0 | -0.5 |
| [1,0,0,1,1,0] | **0** | [0,0,0,0,0,0]×[1,0,0,1,1,0]=0 | 0.5 |
| [1,0,1,1,0,0] | **1** | [0,0,0,0,0,0]×[1,0,1,1,0,0]=0 | 0 |
| [1,1,0,0,1,0] | **0** | [0,0,0,0,0,0]×[1,1,0,0,1,0]=0 | 0.5 |
| [1,1,0,1,0,1] | **1** | [0,0,0,0,0,0]×[1,1,0,1,0,1]=0 | 0 |
| [1,1,0,1,1,0] | **0** | [0,0,0,0,0,0]×[1,1,0,1,1,0]=0 | 0.5 |

$$\frac{1}{n}\sum_{i=1}^{n}\left(\frac{1}{1+e^{-\boldsymbol{\theta}^T x^{(i)}}} - y^{(i)}\right)x_4{}^{(i)} = \mathbf{1}$$

***Then,***

$$\theta_4 = \theta_4 - \alpha \times \mathbf{1}$$

$$= 0 - 0.5 \times 1/6 = \mathbf{-0.5/6}$$

**New Paramter Vector**:

$$\boldsymbol{\theta} = [\mathbf{0}, \mathbf{0}, \mathbf{0.5/6}, \mathbf{0}, \mathbf{-0.5/6}, \boldsymbol{\theta_5}]$$

# A Concrete Example: The Training Phase

- Let us apply logistic regression on the spam email recognition problem, assuming $\boldsymbol{\alpha}$ = 0.5 and starting with $\boldsymbol{\theta}$ = [0, 0, 0, 0, 0, 0]

| $x$ | $y$ | $\theta^T x$ | $(\frac{1}{1+e^{-\theta^T x}} - y)x_5$ |
|---|---|---|---|
| [1,1,1,0,1,1] | **1** | [0,0,0,0,0,0]×[1,1,1,0,1,1]=0 | |
| [1,0,0,1,1,0] | **0** | [0,0,0,0,0,0]×[1,0,0,1,1,0]=0 | |
| [1,0,1,1,0,0] | **1** | [0,0,0,0,0,0]×[1,0,1,1,0,0]=0 | |
| [1,1,0,0,1,0] | **0** | [0,0,0,0,0,0]×[1,1,0,0,1,0]=0 | |
| [1,1,0,1,0,1] | **1** | [0,0,0,0,0,0]×[1,1,0,1,0,1]=0 | |
| [1,1,0,1,1,0] | **0** | [0,0,0,0,0,0]×[1,1,0,1,1,0]=0 | |

# A Concrete Example: The Training Phase

- Let us apply logistic regression on the spam email recognition problem, assuming $\boldsymbol{\alpha}$ = 0.5 and starting with $\boldsymbol{\theta}$ = [0, 0, 0, 0, 0, 0]

| $\boldsymbol{x}$ | $\boldsymbol{y}$ | $\boldsymbol{\theta^T x}$ | $(\frac{1}{1+e^{-\theta^T x}} - \boldsymbol{y})\boldsymbol{x_5}$ |
|---|---|---|---|
| [1,1,1,0,1,1] | **1** | [0,0,0,0,0,0]×[1,1,1,0,1,1]=0 | -0.5 |
| [1,0,0,1,1,0] | **0** | [0,0,0,0,0,0]×[1,0,0,1,1,0]=0 | 0 |
| [1,0,1,1,0,0] | **1** | [0,0,0,0,0,0]×[1,0,1,1,0,0]=0 | 0 |
| [1,1,0,0,1,0] | **0** | [0,0,0,0,0,0]×[1,1,0,0,1,0]=0 | 0 |
| [1,1,0,1,0,1] | **1** | [0,0,0,0,0,0]×[1,1,0,1,0,1]=0 | -0.5 |
| [1,1,0,1,1,0] | **0** | [0,0,0,0,0,0]×[1,1,0,1,1,0]=0 | 0 |

$$\frac{1}{n}\sum_{i=1}^{n}\left(\frac{1}{1+e^{-\boldsymbol{\theta}^T x^{(i)}}} - y^{(i)}\right)x_5{}^{(i)} = -\mathbf{1}$$

***Then,***

$$\theta_5 = \theta_5 - \alpha \times (-\mathbf{1})$$

$$= 0 - 0.5 \times (-1)/6 = \mathbf{0.5/6}$$

**New Paramter Vector**:

$$\boldsymbol{\theta} = [\mathbf{0}, \mathbf{0}, \mathbf{0.5/6}, \mathbf{0}, \mathbf{-0.5/6}, \mathbf{0.5/6}$$

# A Concrete Example: Inference

- Let us infer whether a given new email, say, ***k*** = [1, 0, 1, 0, 0, 1] is a spam or not, using logistic regression with the just learnt parameter vector $\boldsymbol{\theta} = [\mathbf{0}, \mathbf{0}, \mathbf{0.5/6}, \mathbf{0}, \mathbf{-0.5/6}, \mathbf{0.5/6}]$

| | $x_0 = 1$ | $x_1$ = and | $x_2$ = vaccine | $x_3$ = the | $x_4$ = of | $x_5$ = nigeria | y |
|---|---|---|---|---|---|---|---|
| Email **a** | 1 | 1 | 1 | 0 | 1 | 1 | 1 |
| Email **b** | 1 | 0 | 0 | 1 | 1 | 0 | 0 |
| Email **c** | 1 | 0 | 1 | 1 | 0 | 0 | 1 |
| Email **d** | 1 | 1 | 0 | 0 | 1 | 0 | 0 |
| Email **e** | 1 | 1 | 0 | 1 | 0 | 1 | 1 |
| Email **f** | 1 | 1 | 0 | 1 | 1 | 0 | 0 |

**Our Training Dataset**

# A Concrete Example: Inference

- Let us infer whether a given new email, say, ***k*** = [1, 0, 1, 0, 0, 1] is a spam or not, using logistic regression with the just learnt parameter vector $\boldsymbol{\theta} = [\mathbf{0}, \mathbf{0}, \mathbf{0.5/6}, \mathbf{0}, \mathbf{-0.5/6}, \mathbf{0.5/6}]$

| | $x_0 = 1$ | $x_1$ = and | $x_2$ = vaccine | $x_3$ = the | $x_4$ = of | $x_5$ = nigeria | y |
|---|---|---|---|---|---|---|---|
| Email **a** | 1 | 1 | 1 | 0 | 1 | 1 | 1 |
| Email **b** | 1 | 0 | 0 | 1 | 1 | 0 | 0 |
| Email **c** | 1 | 0 | 1 | 1 | 0 | 0 | 1 |
| Email **d** | 1 | 1 | 0 | 0 | 1 | 0 | 0 |
| Email **e** | 1 | 1 | 0 | 1 | 0 | 1 | 1 |
| Email **f** | 1 | 1 | 0 | 1 | 1 | 0 | 0 |
| Email **k** | 1 | 0 | 1 | 0 | 0 | 1 | ? |

# A Concrete Example: Inference

- Let us infer whether a given new email, say, ***k*** = [1, 0, 1, 0, 0, 1] is a spam or not, using logistic regression with the just learnt parameter vector $\boldsymbol{\theta} = [\mathbf{0}, \mathbf{0}, \mathbf{0.5/6}, \mathbf{0}, \mathbf{-0.5/6}, \mathbf{0.5/6}]$

$$\boldsymbol{h_\theta(x)} = \frac{1}{1 + e^{-\boldsymbol{\theta^T x}}} = \frac{1}{1 + e^{\mathbf{-1/6}}}$$

$$\begin{bmatrix} \mathbf{0} \\ \mathbf{0} \\ \mathbf{0.5/6} \\ \mathbf{0} \\ \mathbf{-0.5/6} \\ \mathbf{0.5/6} \end{bmatrix} [\mathbf{1, 0, 1, 0, 0, 1}] = (\mathbf{0.5/6 \times 1}) + (\mathbf{0.5/6 \times 1}) = \mathbf{1/6}$$

$\geq 0.5$ ➔ Class 1 (i.e., Spam)

# TF-IDF (Term frequency-Inverse dense or document frequency)

TF-IDF is scored between 0 and 1. The **higher the numerical weight value, the rarer the term**. The **smaller the weight, the more common the term**.

For a term $i$ in document $j$:

$$w_{i,j} = tf_{i,j} \times \log\left(\frac{N}{df_i}\right)$$

$tf_{ij}$ = number of occurrences of $i$ in $j$

$df_i$ = number of documents containing $i$

$N$ = total number of documents

# Example

**Let's cover an example of 3 documents -**

**Document 1** It is going to rain today.

**Document 2** Today I am not going outside.

**Document 3** I am going to watch the season premiere.

| Word | Count |
| --- | --- |
| going | 3 |
| to | 2 |
| today | 2 |
| i | 2 |
| am | 2 |
| it | 1 |
| is | 1 |
| rain | 1 |

Vocab of document

| Words/ Documents | Document 1 | Document 2 | Document 3 |
|---|---|---|---|
| going | 0.16 | 0.16 | 0.12 |
| to | 0.16 | 0 | 0.12 |
| today | 0.16 | 0.16 | 0 |
| i | 0 | 0.16 | 0.12 |
| am | 0 | 0.16 | 0.12 |
| it | 0.16 | 0 | 0 |
| is | 0.16 | 0 | 0 |
| rain | 0.16 | 0 | 0 |

TF for the document

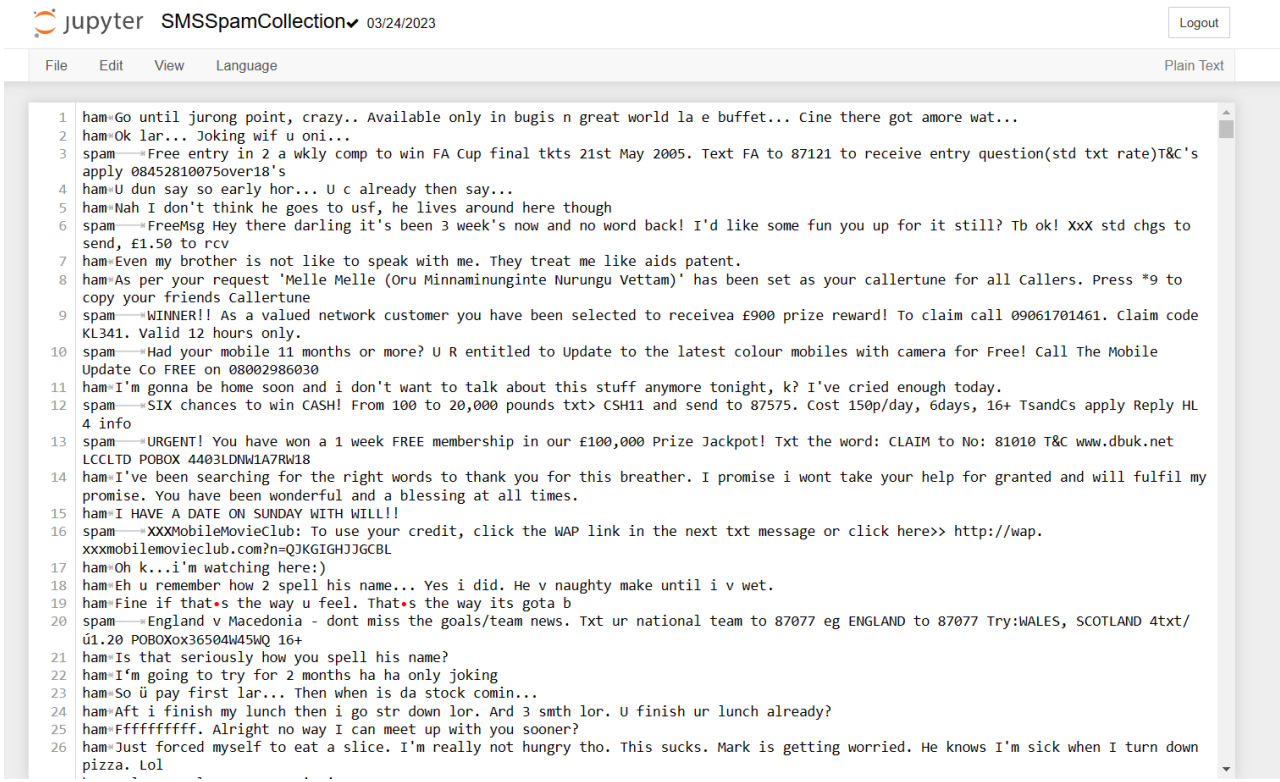
| Words | IDF Value |
| --- | --- |
| going | $\log(3/3)$ |
| to | $\log(3/2)$ |
| today | $\log(3/2)$ |
| i | $\log(3/2)$ |
| am | $\log(3/2)$ |
| It | $\log(3/1)$ |
| is | $\log(3/1)$ |
| rain | $\log(3/1)$ |

IDF for document

| Words/ Documents | going | to | today | i | am | it | is | rain |
|---|---|---|---|---|---|---|---|---|
| Document 1 | 0 | 0.07 | 0.07 | 0 | 0 | 0.17 | 0.17 | 0.17 |
| Document 2 | 0 | 0 | 0.07 | 0.07 | 0.07 | 0 | 0 | 0 |
| Document 3 | 0 | 0.05 | 0 | 0.05 | 0.05 | 0 | 0 | 0 |

Remember, the final equation = TF-IDF = TF * IDF

# SMSSpamCollection database



In [2]:
```
import pandas as pd
import numpy as np
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.linear_model  import LogisticRegression
from sklearn.model_selection import train_test_split, cross_val_score

df = pd.read_csv('SMSSpamCollection', delimiter='\t',header=None)
```

In [3]:
```
X_train_raw, X_test_raw, y_train, y_test = train_test_split(df[1],df[0])

vectorizer = TfidfVectorizer()
X_train = vectorizer.fit_transform(X_train_raw)
```

In [4]:
```
classifier = LogisticRegression()
classifier.fit(X_train, y_train)

X_test = vectorizer.transform( ['URGENT! Your Mobile No 1234 was awarded a Prize', 'Hey honey, whats up?','how are you'] )
print(X_test)
```

In [3]:
```
X_train_raw, X_test_raw, y_train, y_test = train_test_split(df[1],df[0])

vectorizer = TfidfVectorizer()
X_train = vectorizer.fit_transform(X_train_raw)
```

In [4]:
```
classifier = LogisticRegression()
classifier.fit(X_train, y_train)

X_test = vectorizer.transform( ['URGENT! Your Mobile No 1234 was awarded a Prize', 'Hey honey, whats up?','how are you'] )
print(X_test)
```

```
  (0, 7511)     0.25466740289936407
  (0, 7192)     0.3461278313956526
  (0, 7014)     0.4264937405685157
  (0, 5300)     0.4190022423989018
  (0, 4679)     0.2984334164110299
  (0, 4430)     0.3820593750636906
  (0, 1142)     0.47206819237223135
  (1, 7276)     0.5599731789780006
  (1, 6995)     0.33337693489630504
  (1, 3403)     0.6304810448926348
  (1, 3346)     0.42164381904103376
  (2, 7505)     0.3925217496458997
  (2, 3439)     0.679214073335713
  (2, 1021)     0.6201571725277635
```

In [5]:
```
predictions = classifier.predict(X_test)
print(predictions)
```

```
['spam' 'ham' 'ham']
```

# Classification metrics

- **Terms:**
  - $P$: positive examples, involving major interesting classes
  - $N$: negative examples (other examples)
  - TP: true positive examples (positive examples that are correctly classified by the classifier)
  - $TN$: true negative examples (negative examples that are correctly classified by the classifier)
  - $FP$: false positive examples (negative examples that are incorrectly marked as positive examples)
  - $FN$: false negative examples (positive examples that are incorrectly marked as negative examples)

| Predicted / Observed | $Yes$ | $No$ | Total |
|---|---|---|---|
| $Yes$ | $TP$ | $FN$ | $P$ |
| $No$ | $FP$ | $TN$ | $N$ |
| Total | $P'$ | $N'$ | $P + N$ |

Confusion matrix

- Confusion matrix: a table (at least $m \times m$). $CM_{i,j}$, element of the first $i$-th line and $j$-th column is the number of examples known to be in class $i$ but marked as $j$ by the classifier.
  - Ideally, for a highly accurate classifier, most data samples should be represented by the elements on the diagonal from $CM_{1,1}$ and $CM_{m,m}$. Other elements are 0 or close to 0. In other words, $FP$ and $FN$ are close to 0.

# Classification metrics

| Measure | Formula |
|---|---|
| Accuracy or correct classification rate | $\frac{TP + TN}{P + N}$ |
| Error rate or false classification rate | $\frac{FP + FN}{P + N}$ |
| Sensitivity, true positive rate, or $recall$ | $\frac{TP}{P}$ |
| Specificity or true negative rate | $\frac{TN}{N}$ |
| $Precision$ | $\frac{TP}{TP + FP}$ |
| $F$ score: harmonic mean of precision or recall | $\frac{2 \times precision \times recall}{precision + recall}$ |
| $F_\beta$ ($\beta$ is a non-negative real number) | $\frac{(1 + \beta^2) \times precision \times recall}{\beta^2 \times precision + recall}$ |

# Example

- We have trained a machine learning model to identify that whether the object in an image is a cat. Now we use 200 images to test its performance. Among 200 images, 170 contain cats and 30 do not contain cats. The identification result of the model is that 160 images contain cats and 40 do not contain cats.

Precision: $P = \frac{TP}{TP+FP} = \frac{140}{140+20} = 87.5.\%$

Recall: $R = \frac{TP}{P} = \frac{140}{170} = 93.3\%.$

Accuracy: $ACC = \frac{TP+TN}{P+N} = \frac{140+10}{170+30} = 85\%$

| Predicted / Observed | *Yes* | *No* | Total |
|---|---|---|---|
| *Yes* | 140 | 30 | 170 |
| *No* | 20 | 10 | 30 |
| Total | 160 | 40 | 200 |

# Debugging a learning algorithm

$$J(\theta) = \frac{1}{2m}\sum_{i=1}^{m}\left(h_\theta\left(x^{(i)}\right) - y^{(i)}\right)^2$$

- You have built you awesome linear regression model predicting price
- Work perfectly on you testing data

Source: Andrew Ng

# Debugging a learning algorithm

$$J(\theta) = \frac{1}{2m}\sum_{i=1}^{m}\left(h_\theta\left(x^{(i)}\right) - y^{(i)}\right)^2$$

- You have built you awesome linear regression model predicting price
- Work perfectly on your training data
- Then it fails miserably when you test it on the new data you collected
- What to do now?

Source: Andrew Ng

# Things You Can Try

- Get more data
- Try different features
- Try tuning your hyperparameter

- But which should I try first?

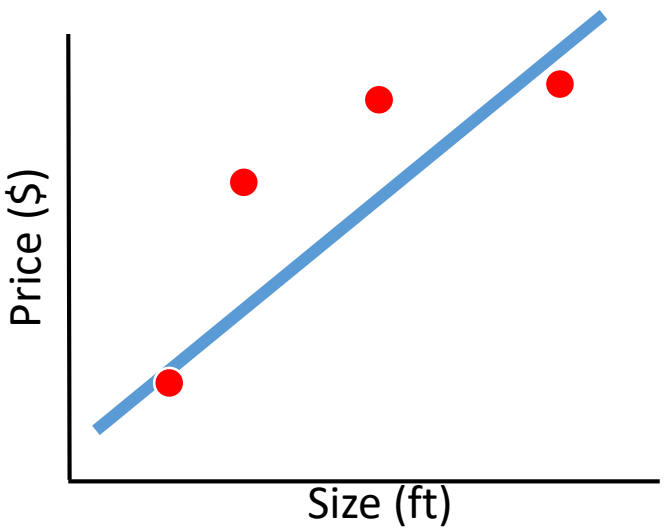
# Diagnosing Machine Learning System

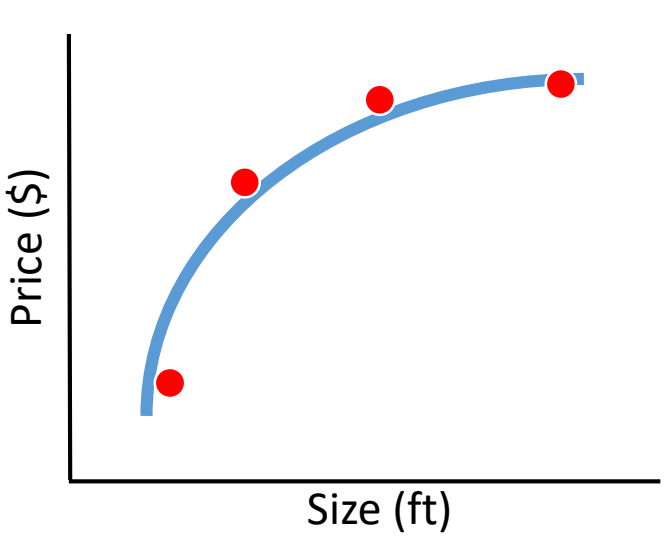
- Figure out what is wrong first
- Diagnosing your system takes time, but it can save your time as well
- Ultimate goal: low generalization error

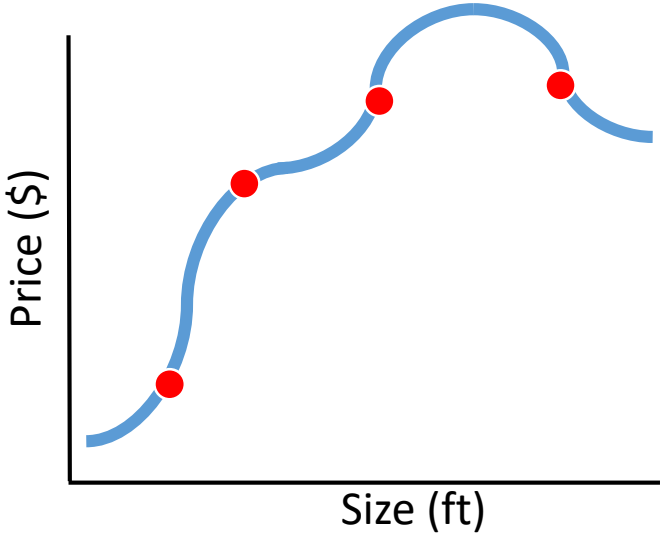
# Problem: Fail to Generalize

- Model does not generalize to unseen data
  - Fail to predict things that are not in training sample
  - Pick a model that has **lower generalization error**

# Evaluate Your Hypothesis

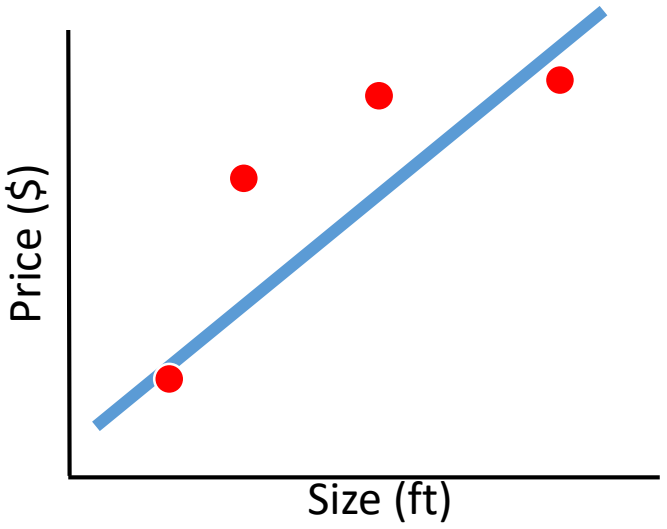


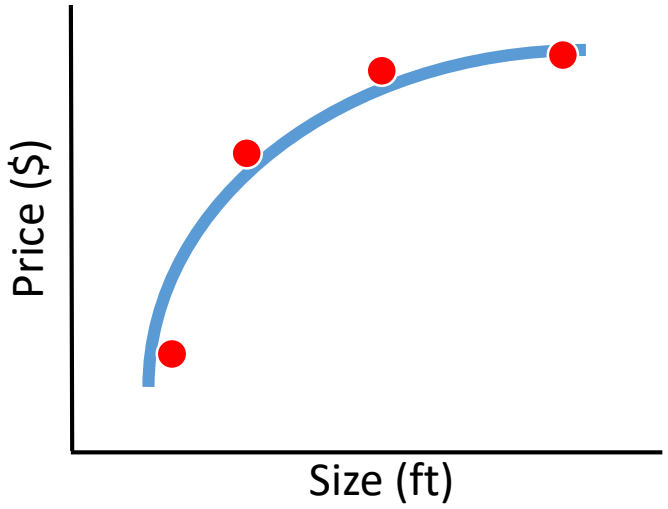

Source: Andrew Ng

# Evaluate Your Hypothesis



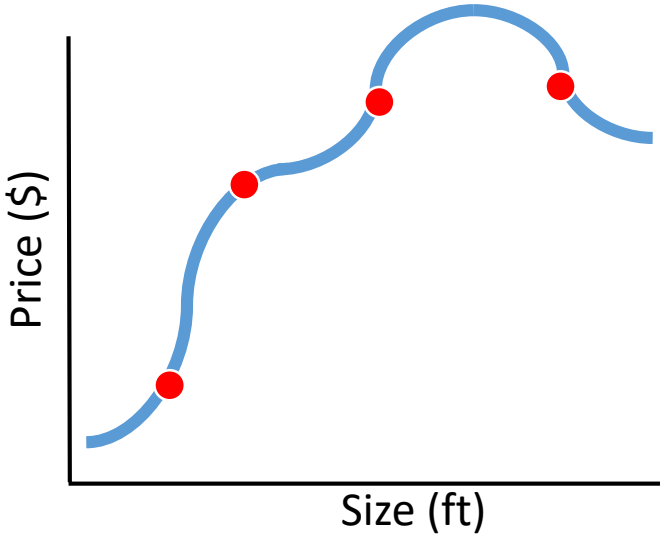

$\theta_0 + \theta_1 x$

Underfit




$\theta_0 + \theta_1 x + \theta_2 x^2$

Just right

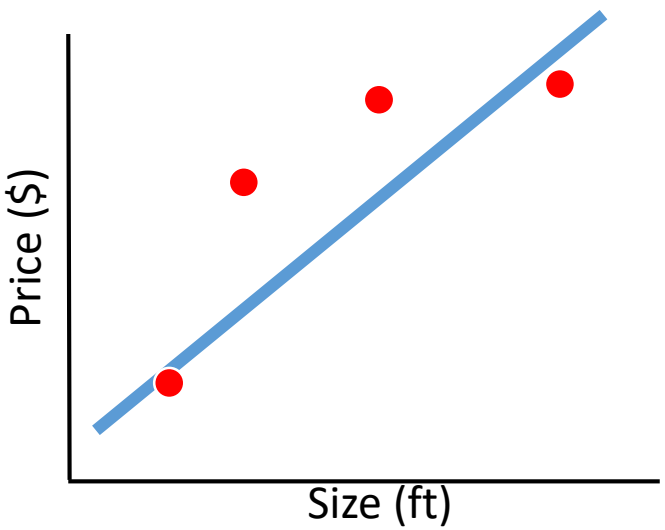



$\theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4$

Overfit

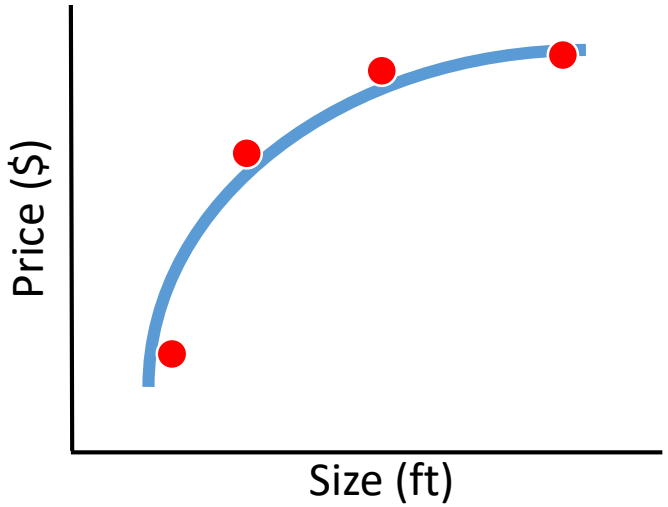
Source: Andrew Ng

# Evaluate Your Hypothesis



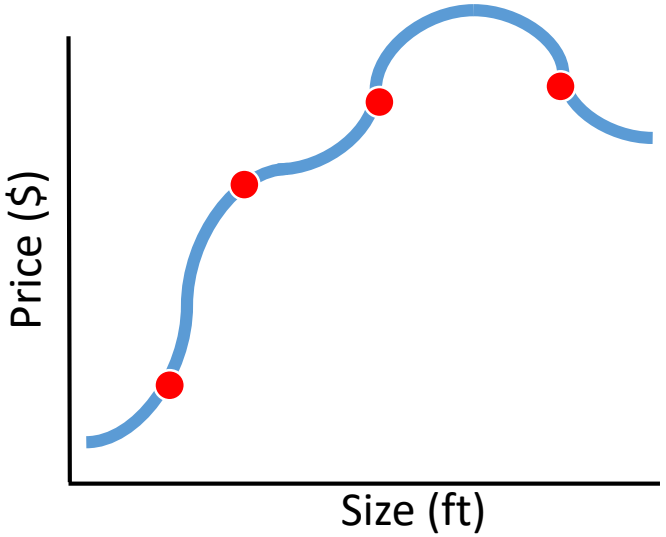

$\theta_0 + \theta_1 x$

Underfit




$\theta_0 + \theta_1 x + \theta_2 x^2$

Just right




$\theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4$

Overfit

- What if the feature dimension is too high?

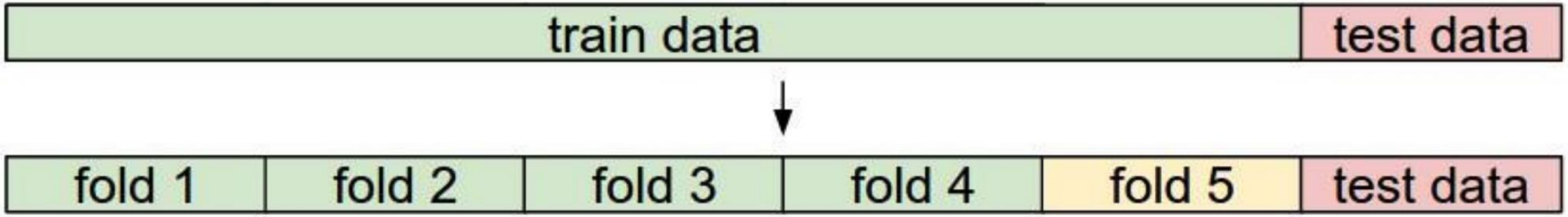
Source: Andrew Ng

# Model Selection

- Model does not generalize to unseen data
  - Fail to predict things that are not in training sample
  - Pick a model that has **lower generalization error**

# Model Selection

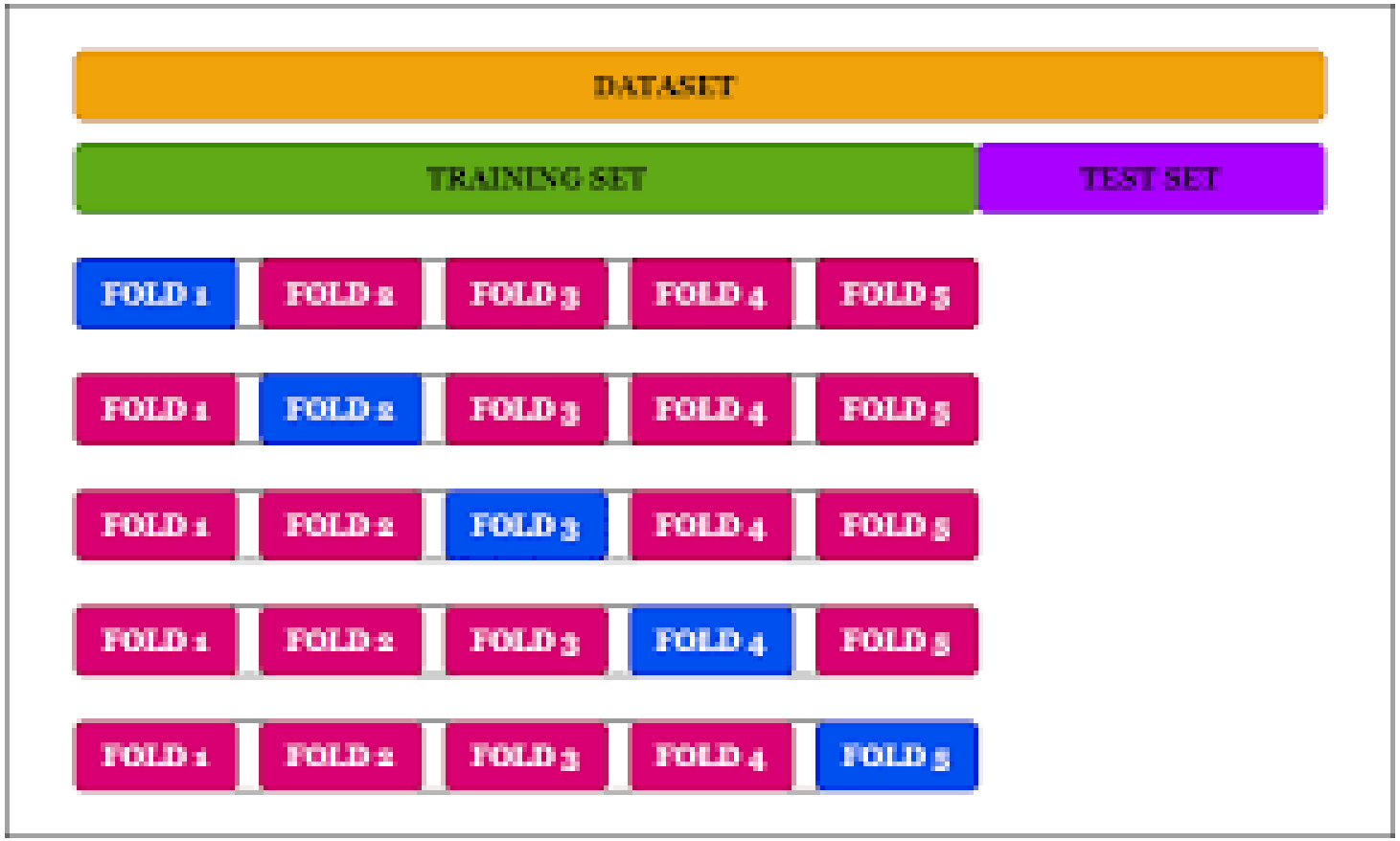
- Model does not generalize to unseen data
  - Fail to predict things that are not in training sample
  - Pick a model that has **lower generalization error**



# Model Selection

- Model does not generalize to unseen data
  - Fail to predict things that are not in training sample
  - Pick a model that has **lower generalization error**

- How to evaluate generalization error?
  - Split your data into *train*, *validation*, and *test set*.
  - Use test set error as an *estimator* of generalization error
  - Technique K-fold cross validation


DATASET
TRAINING SET
TEST SET
FOLD 1
FOLD 2
FOLD 3
FOLD 4
FOLD 5
FOLD 1
FOLD 2
FOLD 3
FOLD 4
FOLD 5
FOLD 1
FOLD 2
FOLD 3
FOLD 4
FOLD 5
FOLD 1
FOLD 2
FOLD 3
FOLD 4
FOLD 5
FOLD 1
FOLD 2
FOLD 3
FOLD 4
FOLD 5

# Model Selection

- Training error

$$J_{train}(\theta) = \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2$$

- Validation error

$$J_{cv}(\theta) = \frac{1}{2m_{cv}} \sum_{i=1}^{m_{cv}} (h_\theta(x_{cv}^{(i)}) - y_{cv}^{(i)})^2$$

- Test error

$$J_{test}(\theta) = \frac{1}{2m_{test}} \sum_{i=1}^{m_{test}} (h_\theta(x_{test}^{(i)}) - y_{test}^{(i)})^2$$

# Model Selection

- Training error

$$J_{train}(\theta) = \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2$$

- Validation error

$$J_{cv}(\theta) = \frac{1}{2m_{cv}} \sum_{i=1}^{m_{cv}} (h_\theta(x_{cv}^{(i)}) - y_{cv}^{(i)})^2$$

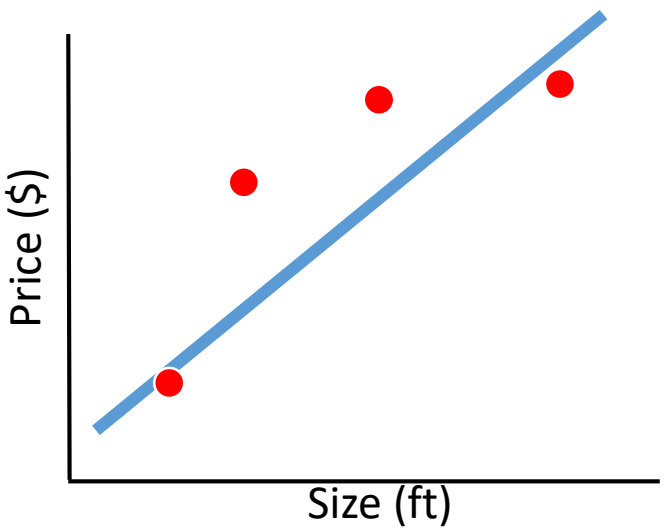
- Test error

$$J_{test}(\theta) = \frac{1}{2m_{test}} \sum_{i=1}^{m_{test}} (h_\theta(x_{test}^{(i)}) - y_{test}^{(i)})^2$$

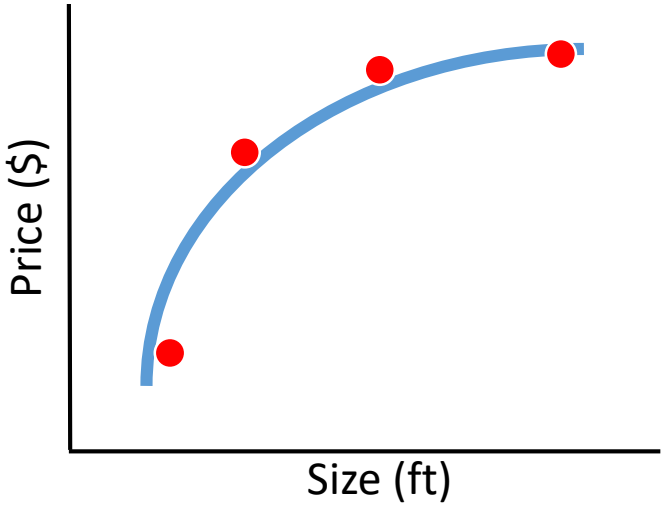
Procedure:

Step 1. Train on training set
Step 2. Evaluate validation error
Step 3. Pick the best model based on Step 2.
Step 4. Evaluate the test error

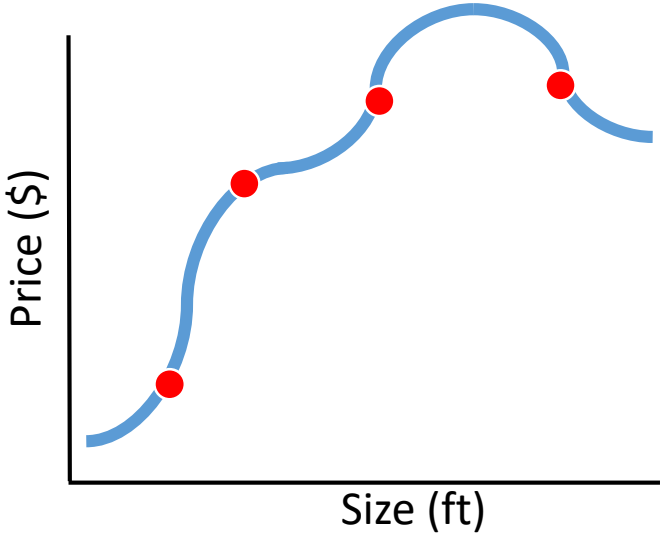
# Bias/Variance Trade-off




Underfit



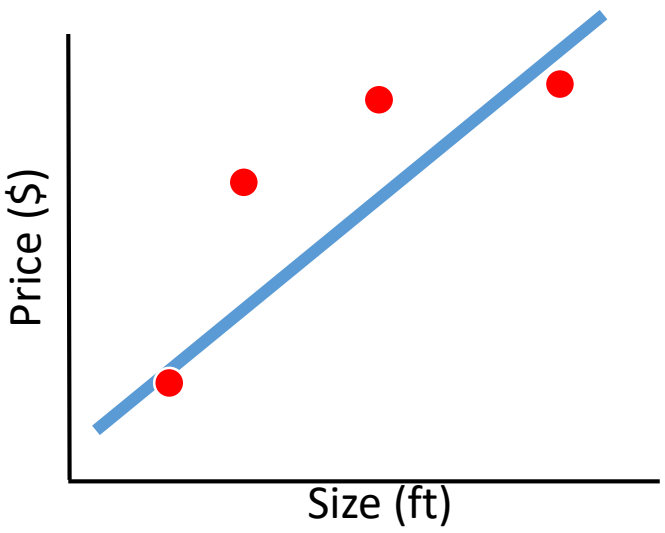

Just right



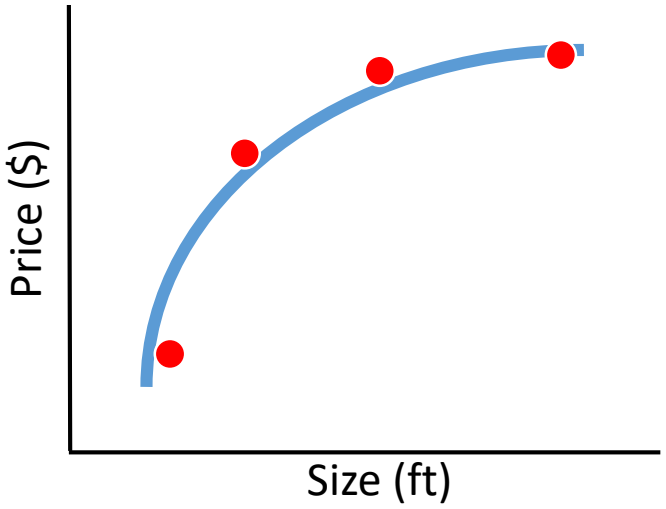

Overfit

Source: Andrew Ng

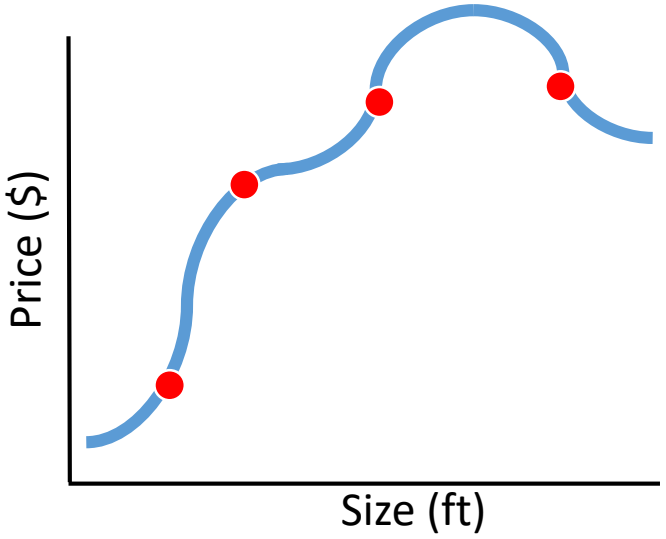
# Bias/Variance Trade-off




Underfit
High bias



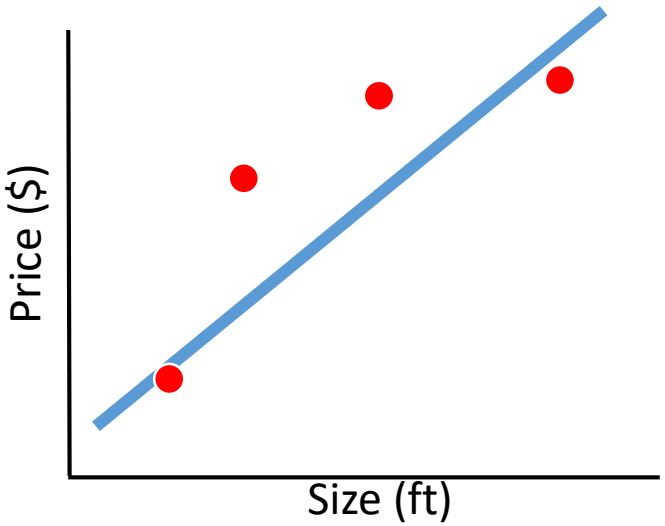

Just right



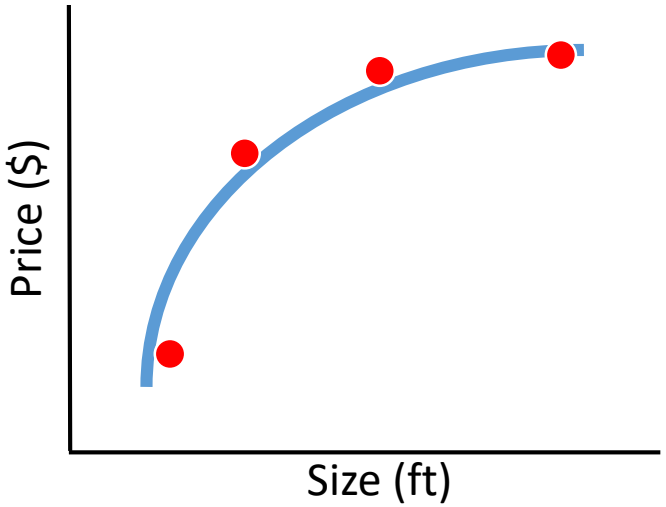

Overfit
High Variance

Source: Andrew Ng

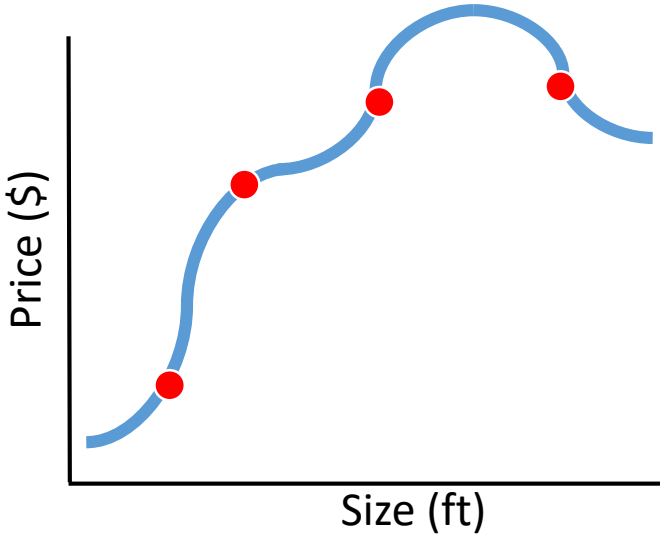
# Bias/Variance Trade-off




Underfit
High bias
Too simple




Just right

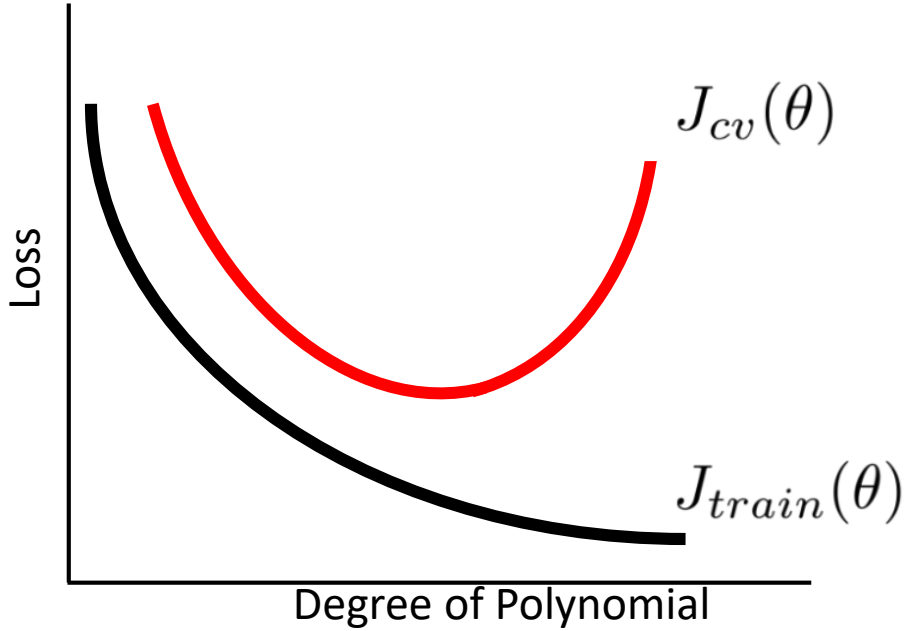



Overfit
High Variance
Too Complex

Source: Andrew Ng

# Bias / Variance Trade-off

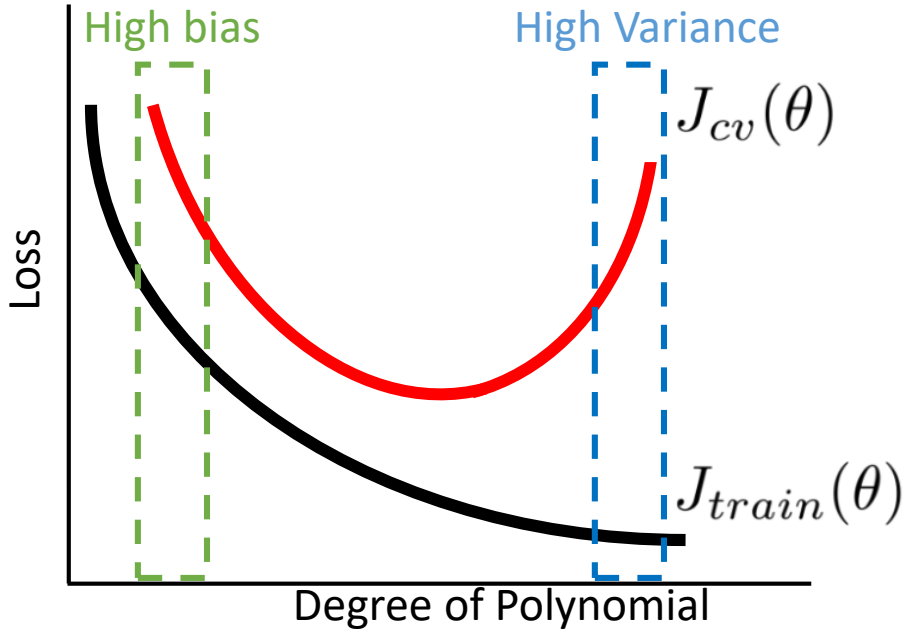
- Training error $J_{train}(\theta) = \frac{1}{2m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2$
- Cross-validation error $J_{cv}(\theta) = \frac{1}{2m_{cv}}\sum_{i=1}^{m_{cv}}(h_\theta(x_{cv}^{(i)}) - y_{cv}^{(i)})^2$




Source: Andrew Ng

# Bias / Variance Trade-off

- Training error $J_{train}(\theta) = \frac{1}{2m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2$
- Cross-validation error $J_{cv}(\theta) = \frac{1}{2m_{cv}}\sum_{i=1}^{m_{cv}}(h_\theta(x_{cv}^{(i)}) - y_{cv}^{(i)})^2$

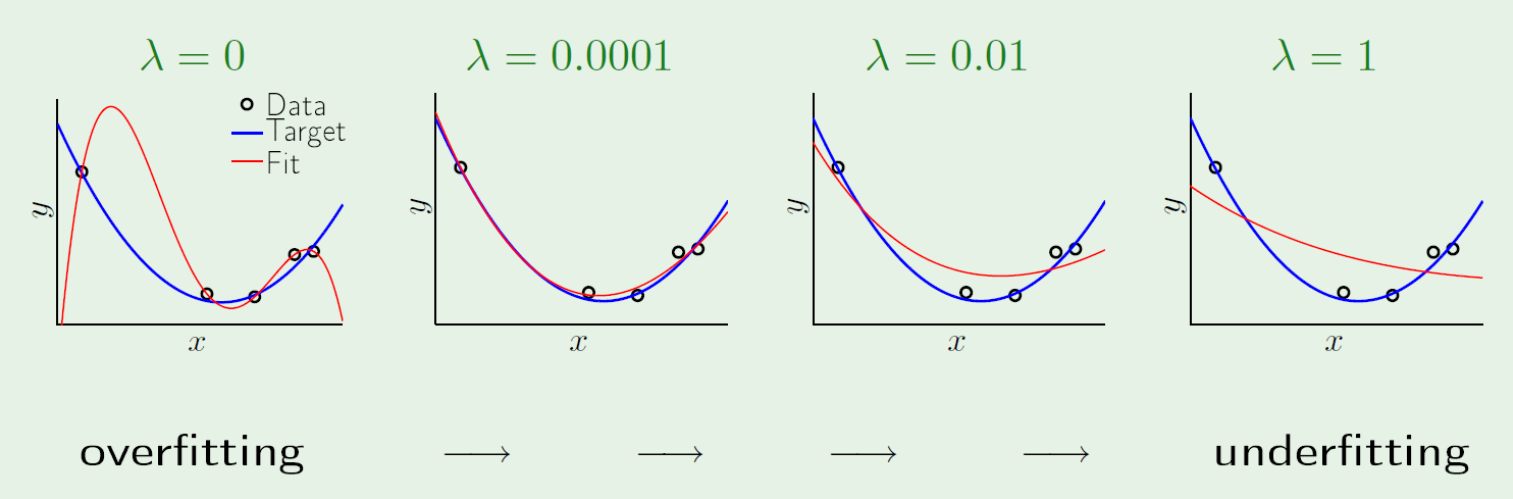


# Regularization

- Involves shrinking the estimated coefficients toward zero relative to the OLS estimates; has the effect of reducing variance.
- Adds another term to the loss function to be reduced= $\lambda$. Regularization term, where $\lambda \geq 0$ is a *tuning parameter*, to be determined separately.
- The weight decay coefficient $\lambda$ determines how dominant the regularization is during the gradient computation
- It could be
  - L1 regularization : Lasso (order 1)
  - L2 regularization:  Ridge (order 2)

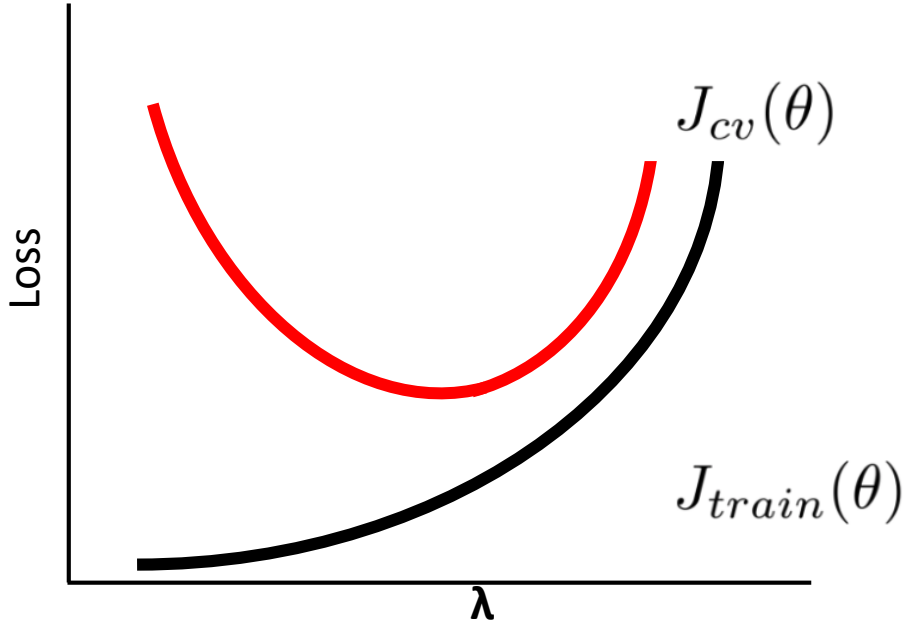
# Regularization or Shrinkage

- Look at what a little regularization can do:



# Bias / Variance Trade-off with Regularization L2

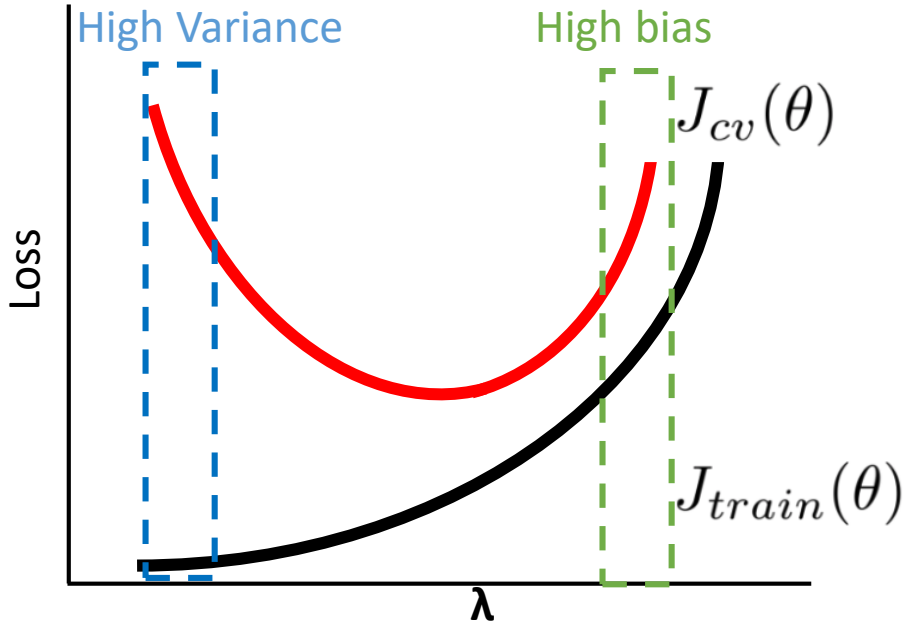
- Training error $J_{train}(\theta) = \frac{1}{2m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2 + \frac{\lambda}{2m}\sum_{j=1}^{m}\theta_j^2$
- Cross-validation error $J_{cv}(\theta) = \frac{1}{2m_{cv}}\sum_{i=1}^{m_{cv}}(h_\theta(x_{cv}^{(i)}) - y_{cv}^{(i)})^2 + \frac{\lambda}{2m_{cv}}\sum_{j=1}^{m_{cv}}\theta_j^2$




Source: Andrew Ng

# Bias / Variance Trade-off with Regularization L2

- Training error $J_{train}(\theta) = \frac{1}{2m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2 + \frac{\lambda}{2m}\sum_{j=1}^{m}\theta_j^2$
- Cross-validation error $J_{cv}(\theta) = \frac{1}{2m_{cv}}\sum_{i=1}^{m_{cv}}(h_\theta(x_{cv}^{(i)}) - y_{cv}^{(i)})^2 + \frac{\lambda}{2m_{cv}}\sum_{j=1}^{m_{cv}}\theta_j^2$




Source: Andrew Ng

# Problem: Fail to Generalize

- Should we get more data?

# Problem: Fail to Generalize

- Should we get more data?

- Getting more data does not always help

# Problem: Fail to Generalize

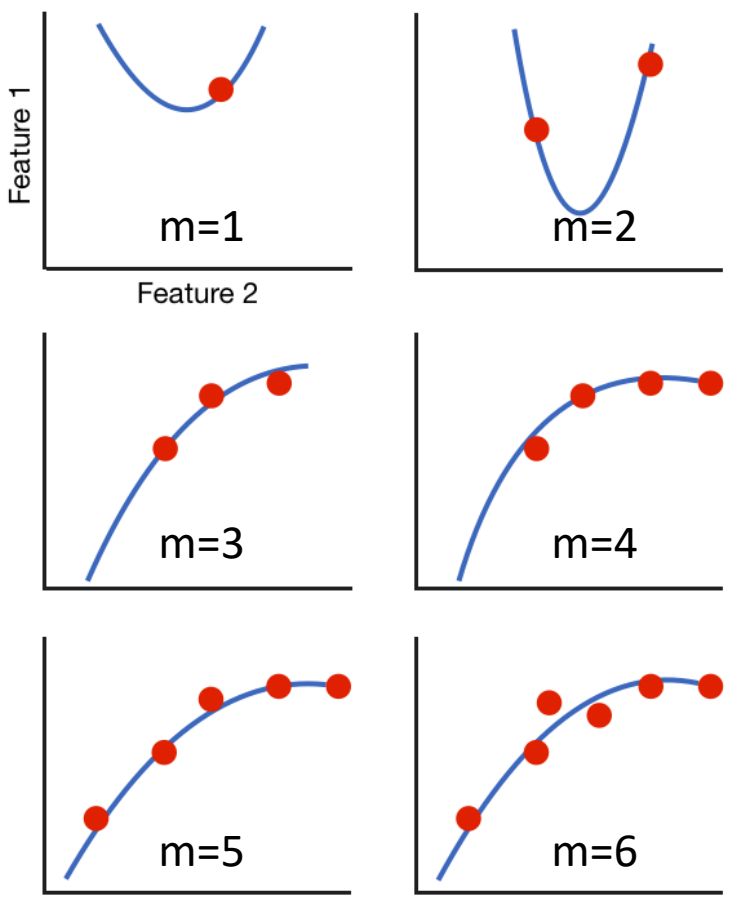
- Should we get more data?
- Getting more data does not always help
- How do we know if we should collect more data?

# Learning Curve

$$J_{train}(\theta) = \frac{1}{2m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2$$

$$J_{cv}(\theta) = \frac{1}{2m_{cv}}\sum_{i=1}^{m_{cv}}(h_\theta(x_{cv}^{(i)}) - y_{cv}^{(i)})^2$$

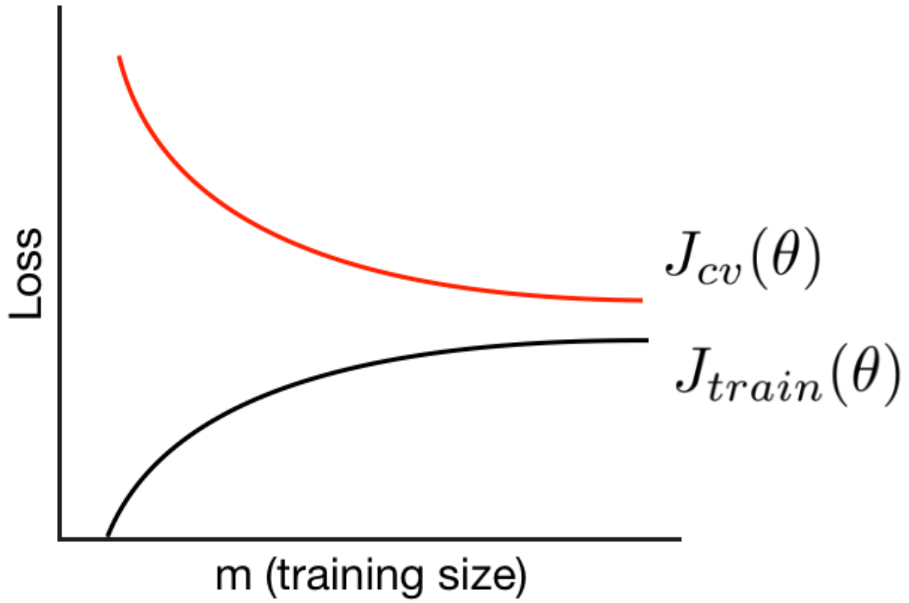
$\theta_0 + \theta_1 x + \theta_2 x^2$



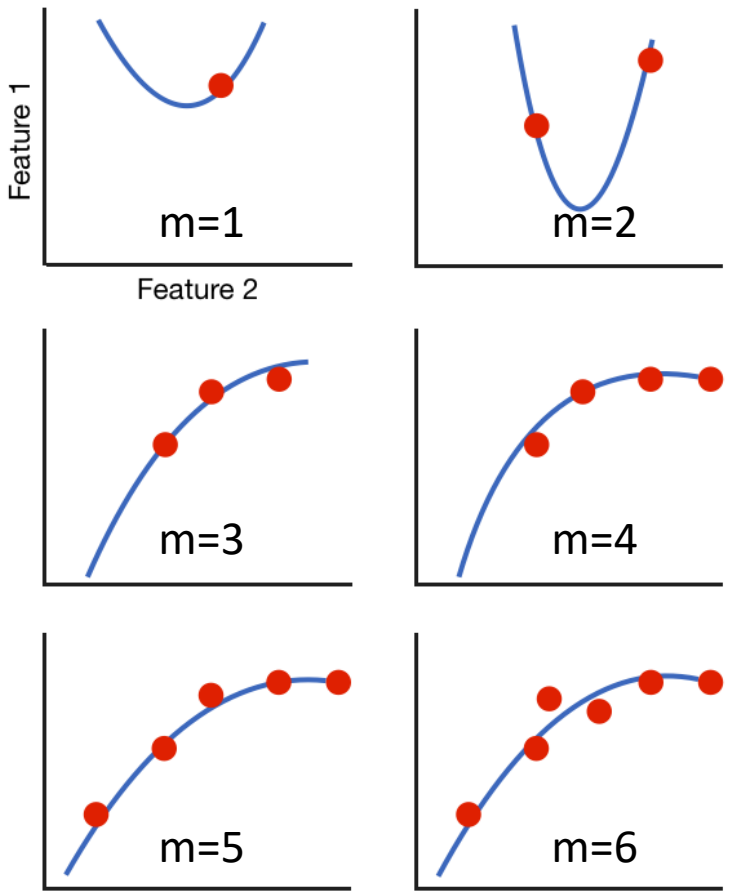
# Learning Curve

$$J_{train}(\theta) = \frac{1}{2m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2$$

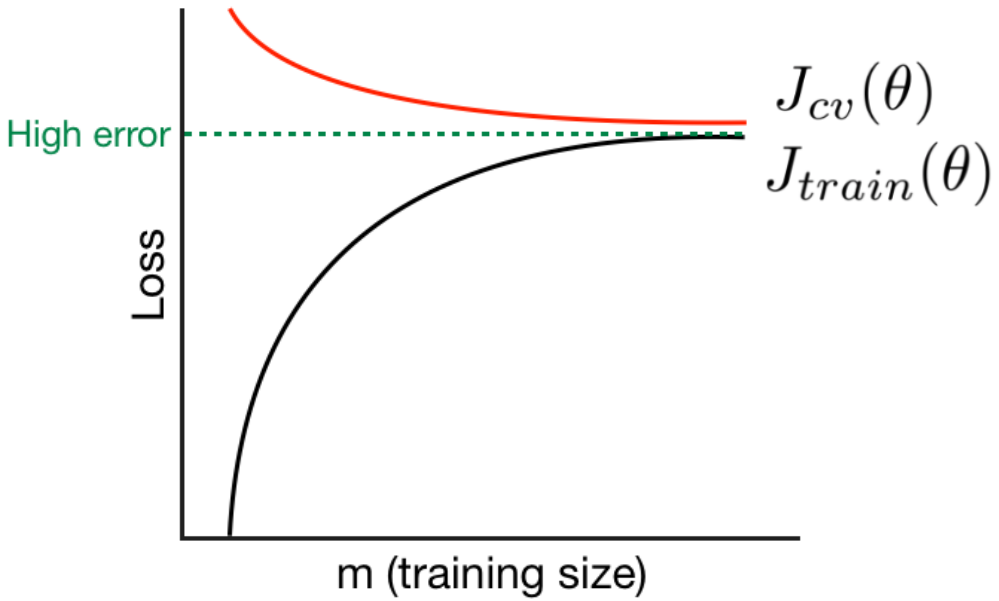
$$J_{cv}(\theta) = \frac{1}{2m_{cv}}\sum_{i=1}^{m_{cv}}(h_\theta(x_{cv}^{(i)}) - y_{cv}^{(i)})^2$$

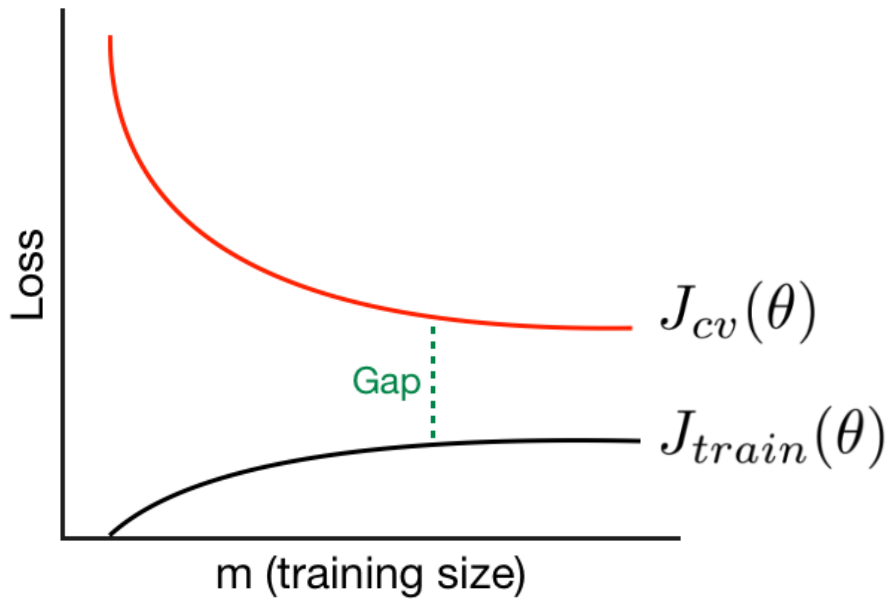



$$\theta_0 + \theta_1 x + \theta_2 x^2$$

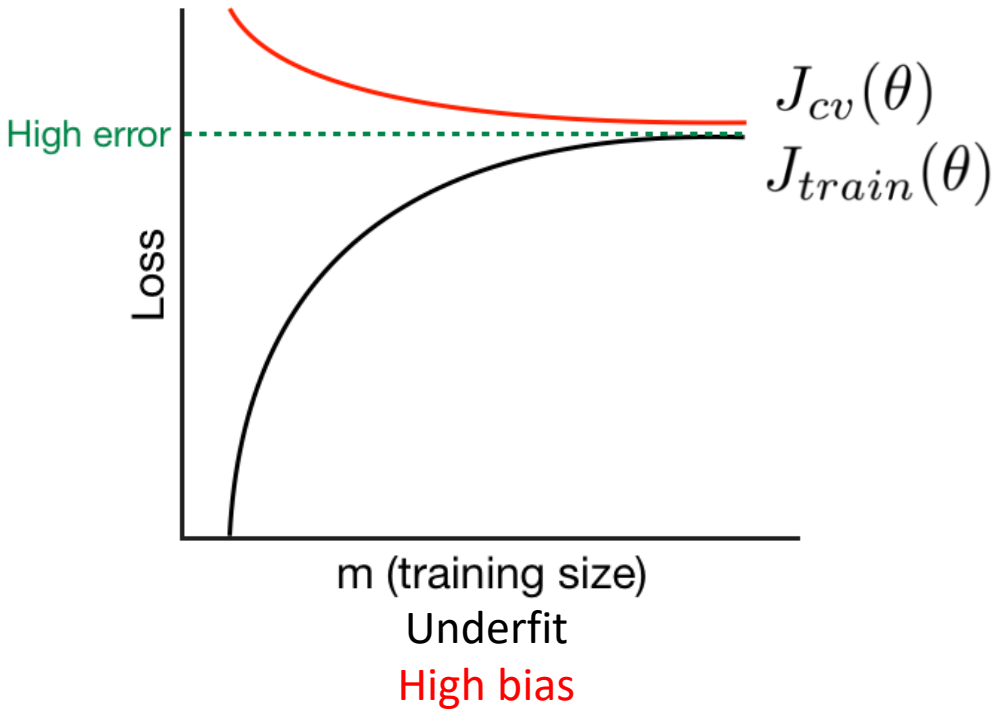


# Learning Curve



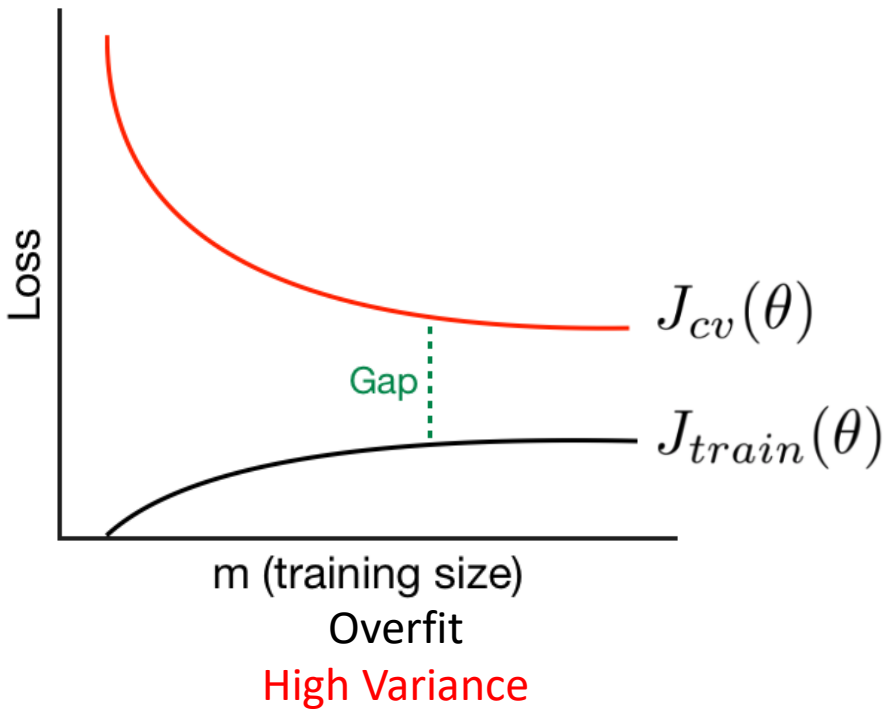



# Learning Curve




Underfit

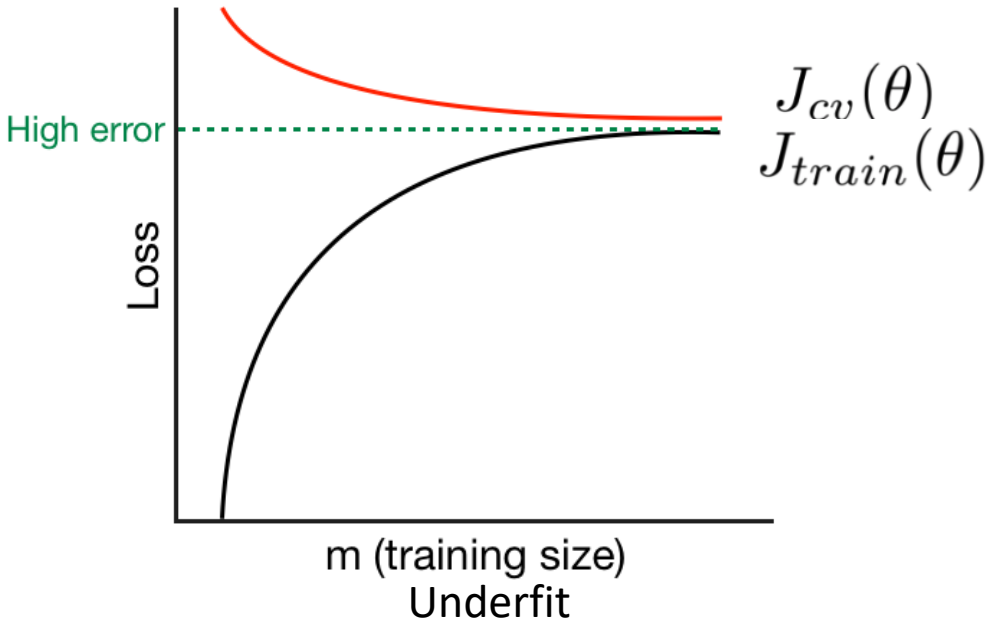
High bias




Overfit

High Variance

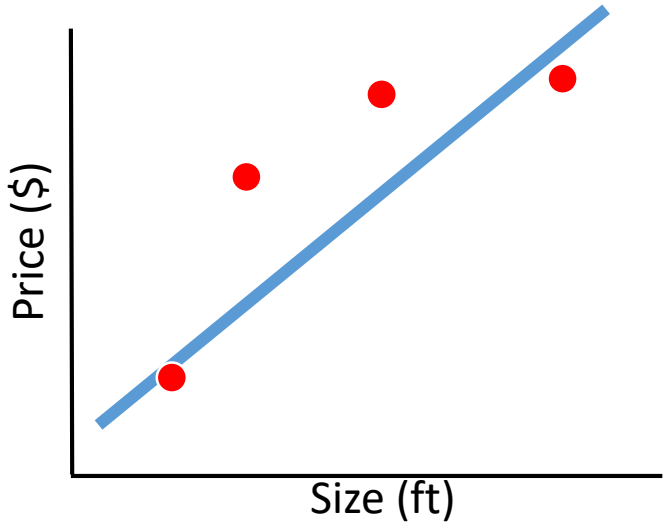
# Learning Curve



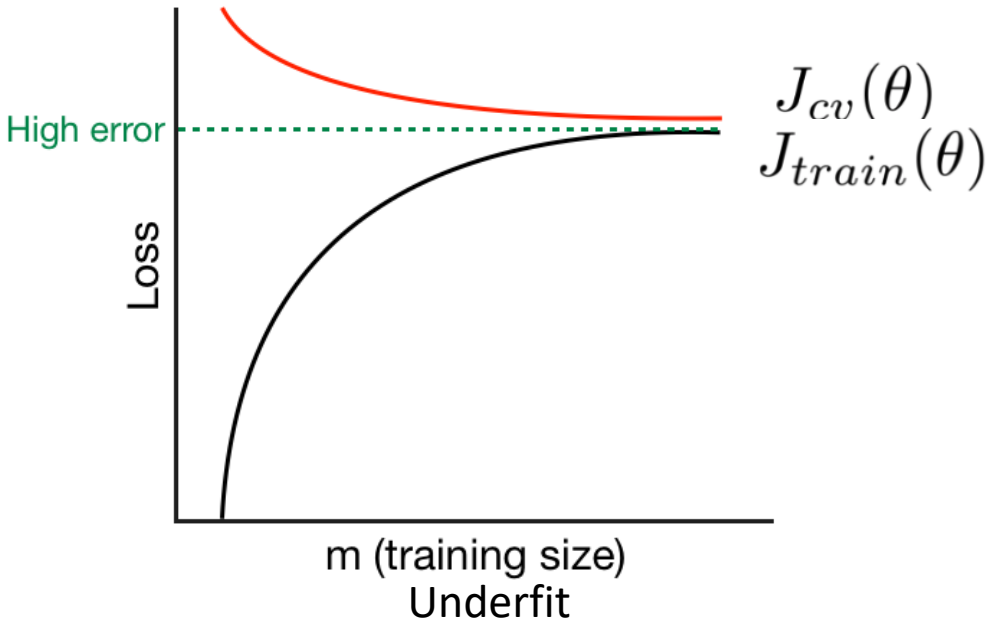

Underfit

High bias

Does adding more data help?



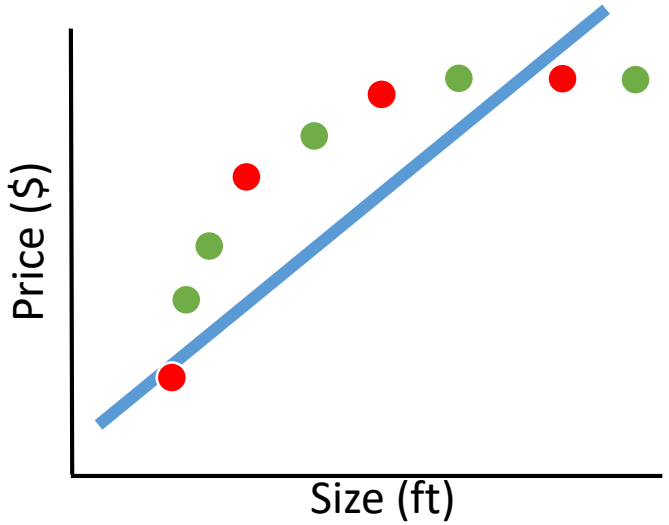
# Learning Curve



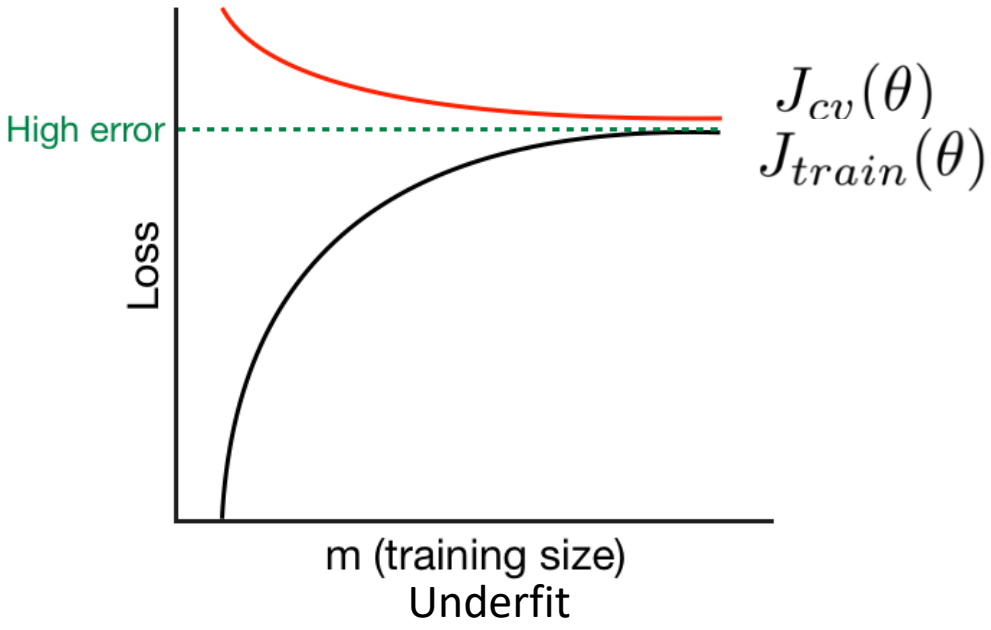

Underfit

High bias

Does adding more data help?



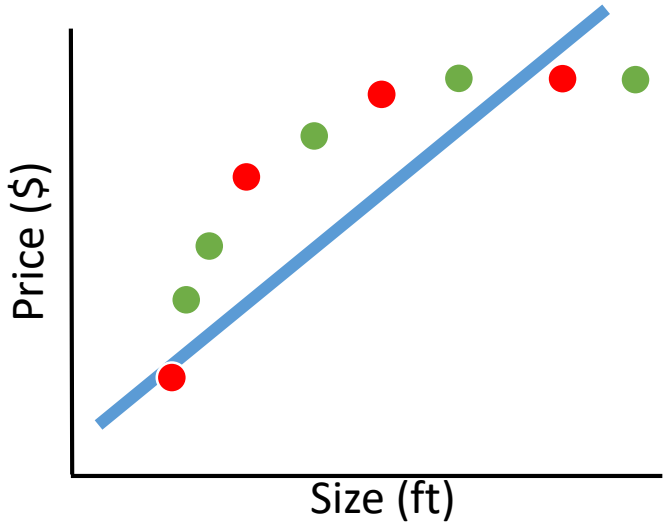
# Learning Curve




Underfit

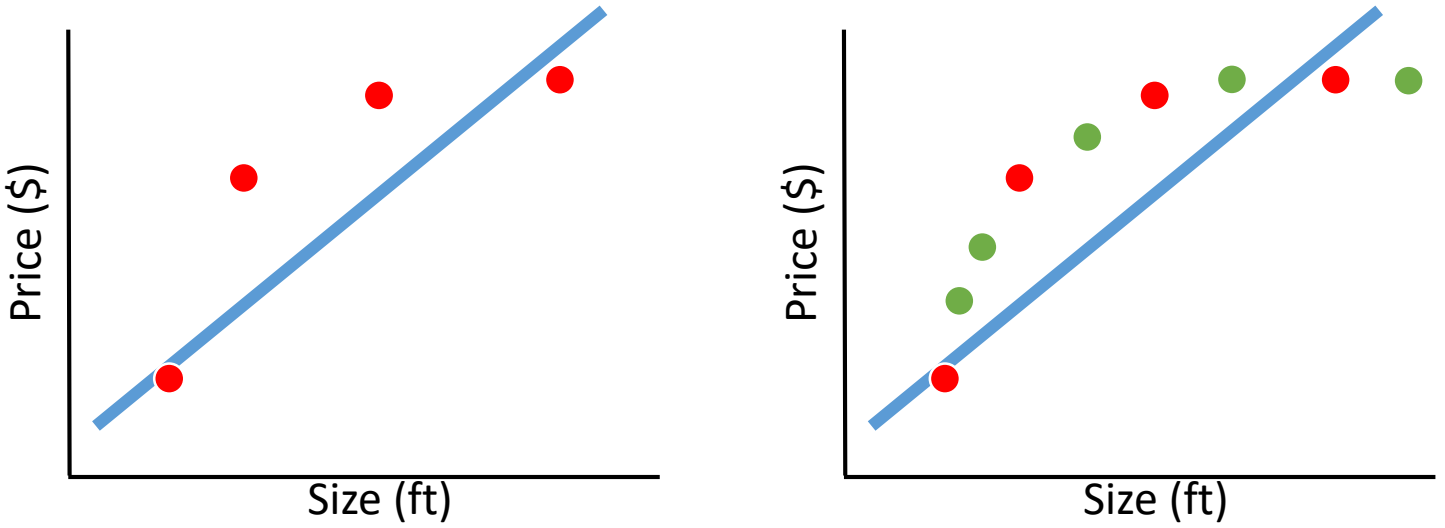
High bias

Does adding more data help?



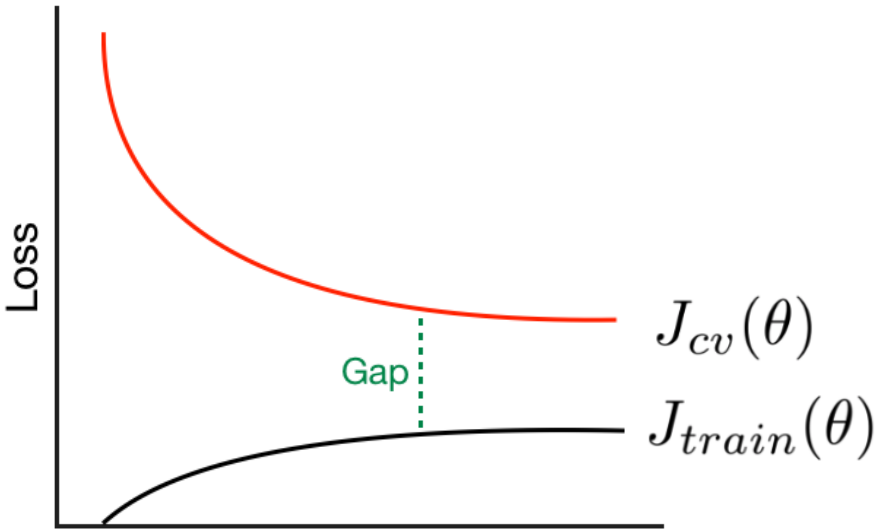
# Learning Curve

Does adding more data help?




More data doesn't help when your model has high bias

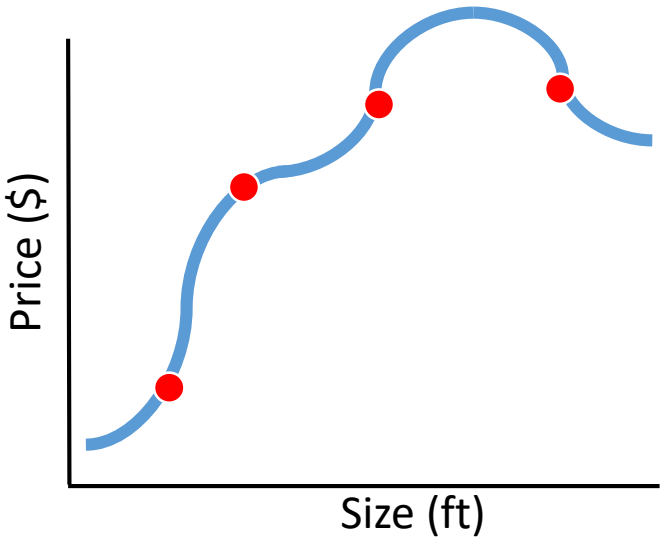
# Learning Curve



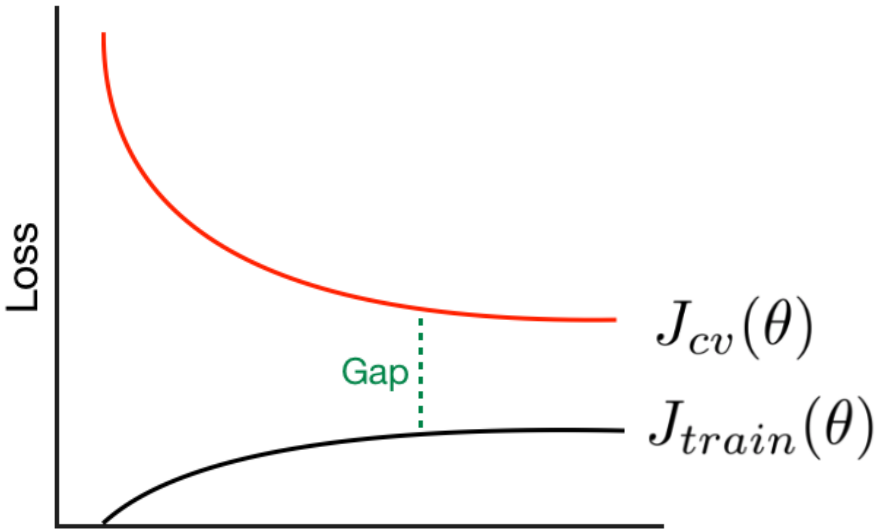

Overfit

High Variance

Does adding more data help?



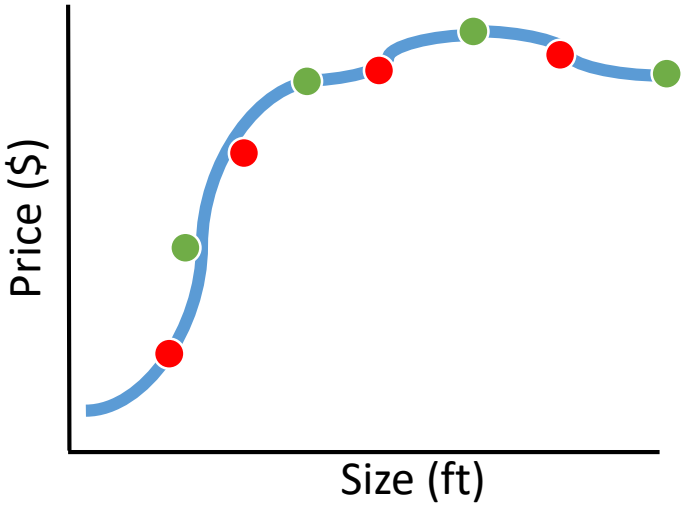
# Learning Curve




Overfit

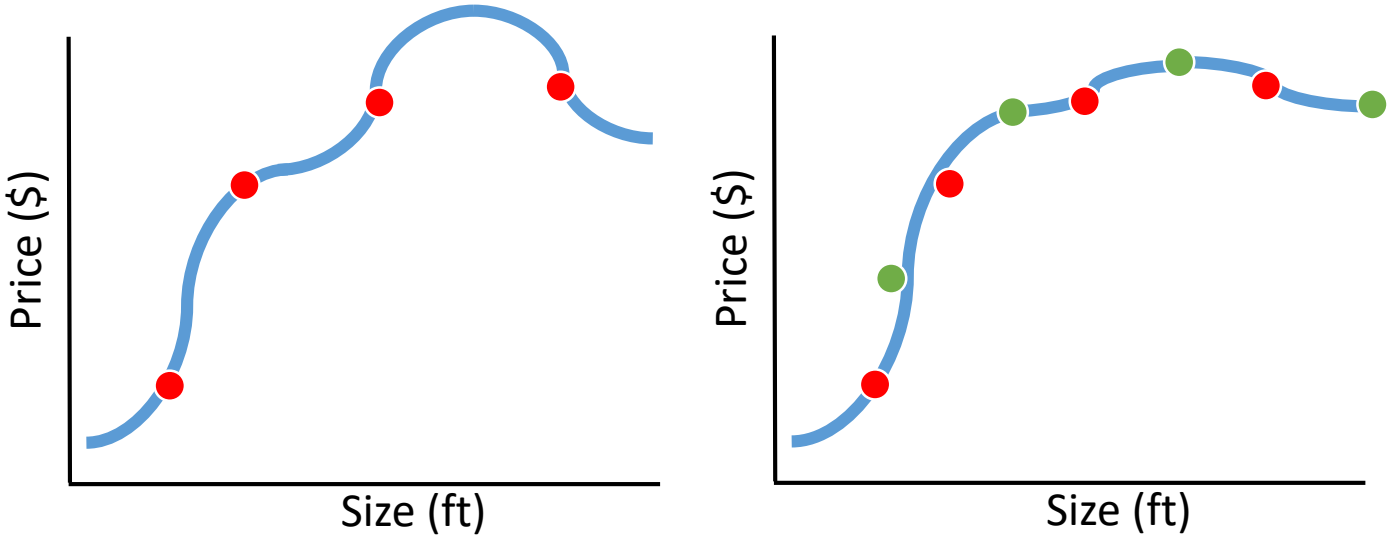
High Variance

Does adding more data help?



# Learning Curve

Does adding more data help?



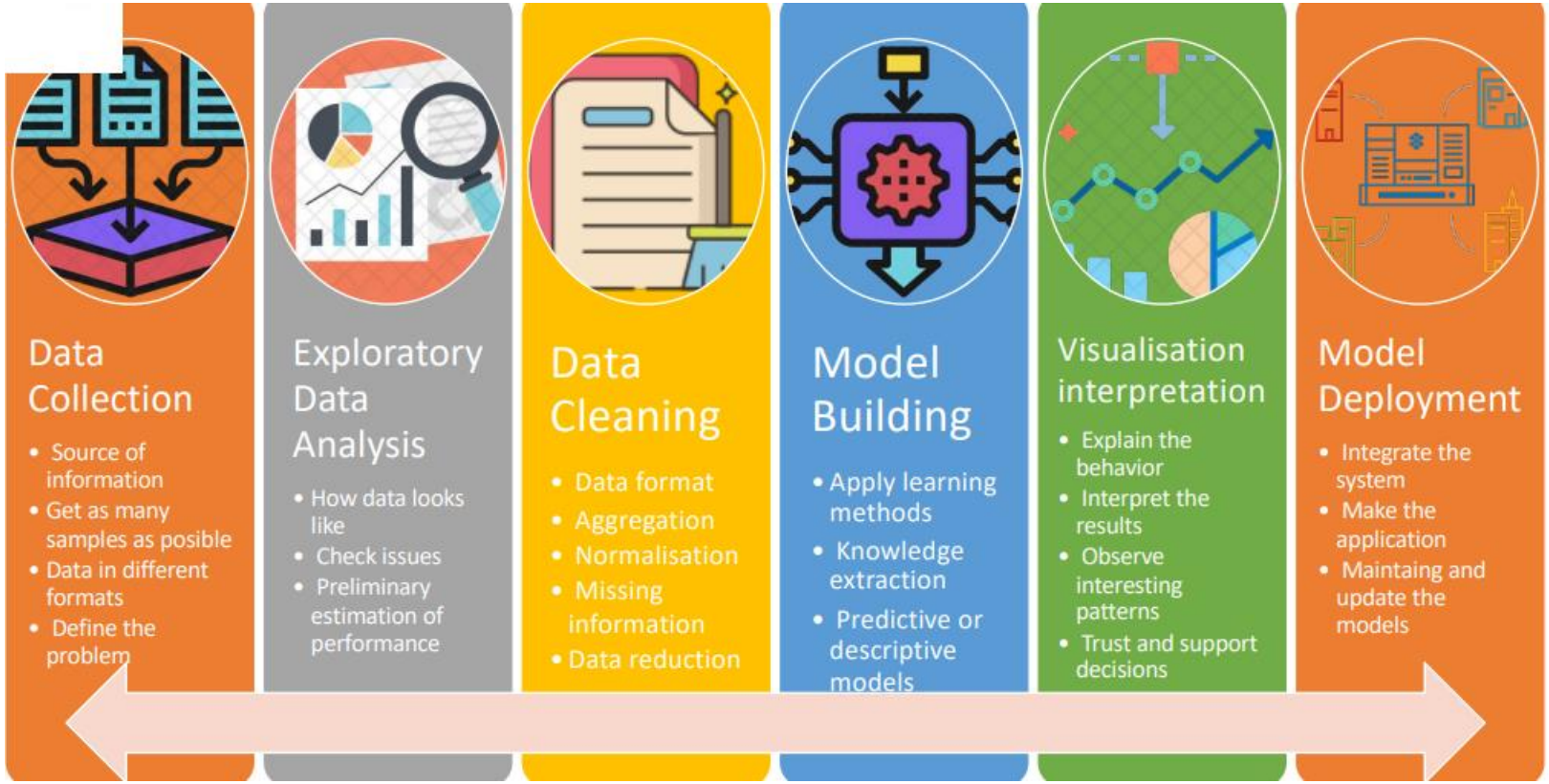

More data is likely to help when your model has high variance

# Things You Can Try

- Get more data
  - When you have high variance
- Try different features
  - Adding feature helps fix high bias
  - Using smaller sets of feature fix high variance
- Try tuning your decay parameter
  - Decrease regularization when bias is high
  - Increase regularization when variance is high

Analyze your model before you act

# ML pipeline




Data preprocessing

# Data Preprocessing

**What does Data Preprocessing include?**

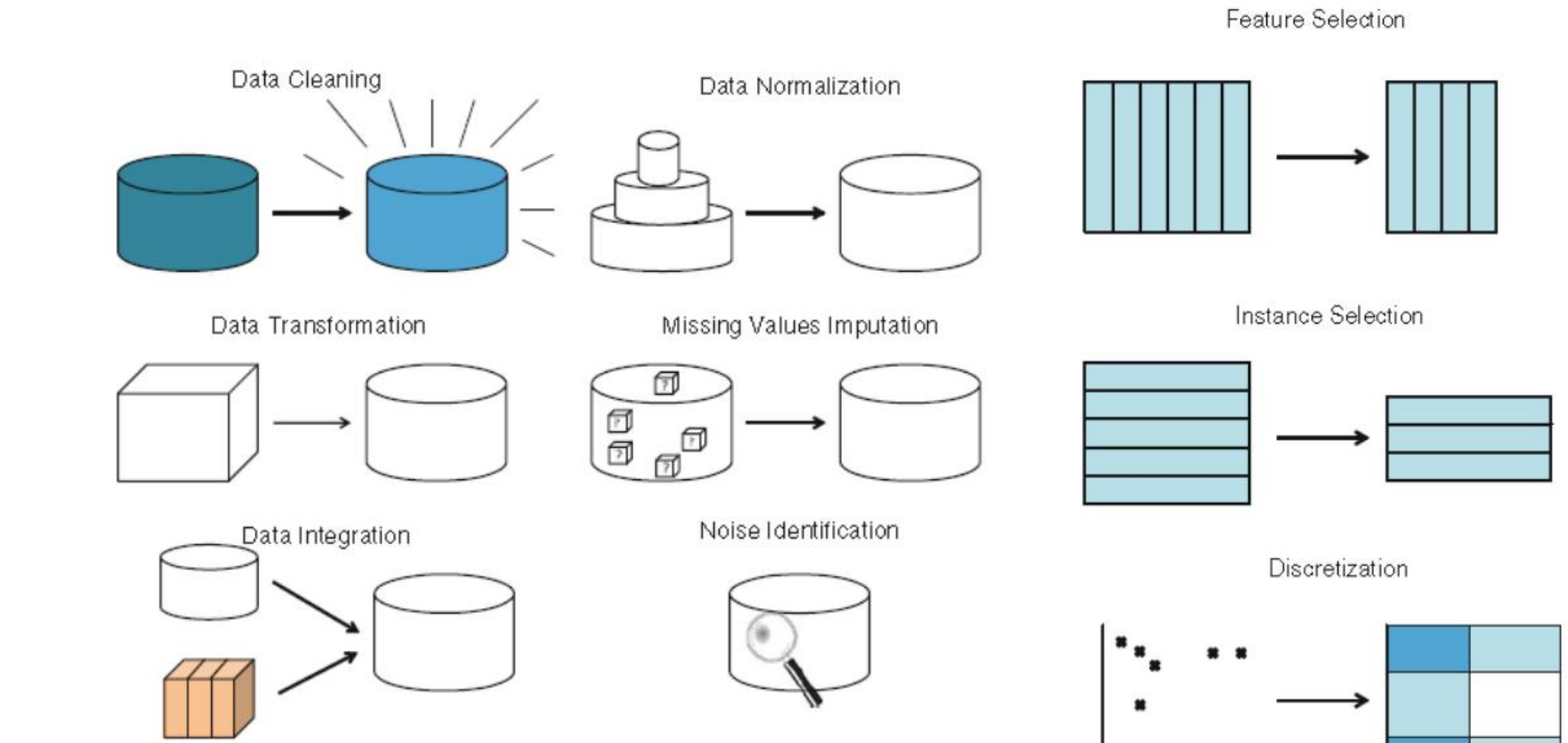



**Fig. 1.3** Forms of data preparation

# Data Integration

- Obtains data from different sources of information
- Solves representation and coding problems
- Integrates data from different tables to create homogeneous information, ...

# Data cleansing

- Objectives:
  - resolving inconsistencies
  - Fill/impute missing values,
  - smoothing the noise in the data,
  - identify or eliminate *outliers* ...

# Data cleansing

- Data need to be formatted for a given software tool
- Data need to be made adequate for a given method
- Data in the real world is dirty
  - incomplete: lacking attribute values, lacking certain attributes of interest, or containing only aggregate data
    - e.g., occupation=“”
  - noisy: containing errors or outliers
    - e.g., Salary=“-10”, Age=“222”
  - inconsistent: containing discrepancies in codes or names
    - e.g., Age=“42” Birthday=“03/07/1997”
    - e.g., Was rating “1,2,3”, now rating “A, B, C”
    - e.g., discrepancy between duplicate records

# Normalisation (MinMaxScaling)

- Objective: To move the values of an attribute to a better rank.
- Useful for some techniques such as AANN or distance-based methods (nearest neighbours,...).
- Some standardisation techniques:
  - min-max normalisation: Performs a linear transformation of the original data.

$$[\min_A, \max_A] \rightarrow [nuevo_{\min_A}, nuevo_{\max_A}]$$

$$v' = \frac{v - \min_A}{\max_A - \min_A}(nuevo_{\max_A} - nuevo_{\min_A}) + nuevo_{\min_A}$$

Relationships between the original data are preserved.

# Standardisation

- *Zero-mean* normalisation. It is normalised according to the mean and the standard deviation.

$$v' = \frac{v - \overline{A}}{\sigma_A}$$

Useful when limits are unknown or when outliers may dominate the min-max normalisation.

- Normalisation by decimal scale. Normalises by moving the decimal point of the attribute values. The number of decimal points moved depends on the maximum absolute value of A.

$$v' = \frac{v}{10^j}$$

with *j* equal to the smallest integer such that max(|v'|)<1.

E.g. : if the data are in [-986,917], then *j*=3.

# Data Transformation

- Objective: To transform the data in the best possible way for the application of DM algorithms.
- Some typical operations:
  - Aggregation. E.g. aggregation of monthly sales into a single attribute that is annual sales,...
  - Data generalisation. The aim is to obtain higher level data from current data, using hierarchies of concepts.
    - City streets
    - Numerical age {young, adult, middle-aged, old}
  - Normalisation: Change the range [-1,1] or [0,1].
  - Linear, quadratic, polynomial transformations, ...

# Different examples of variables or features

- ID numbers, Names of people
- eye color, zip codes
- rankings (e.g., taste of potato chips on a scale from 1-10), grades,  height in {tall, medium, short}
- calendar dates, temperatures in Celsius or Fahrenheit, GRE (Graduate Record Examination) and IQ scores
- temperature in Kelvin, length, time, counts

# Different examples of features

- Examples:
  - US State Code (50 values)
  - Profession Code (7,000 values, but only few frequent)

- Ignore ID-like fields whose values are unique for each record
- For other fields, group values “naturally”:
  - e.g. 50 US States = 3 or 5 regions
  - Profession – select most frequent ones, group the rest

# Encoding

- the machine learning model can’t process categorical variables.
- So when we have categorical variables in our dataset then we need to convert them into numerical variables.
- There are many ways to convert categorical values into numerical values.  Most popular

  - One hot encoding
  - Label encoding

# One hot encoding

| Country |
| --- |
| India |
| Australia |
| Russia |
| America |

| Country | India | Australia | Russia | America |
| --- | --- | --- | --- | --- |
| India | 1 | 0 | 0 | 0 |
| Australia | 0 | 1 | 0 | 0 |
| Russia | 0 | 0 | 1 | 0 |
| America | 0 | 0 | 0 | 1 |

# Label encoding

| State | Confirmed | Recovered | Deaths |
|---|---|---|---|
| Punjab | 2000 | 1500 | 200 |
| Haryana | 2321 | 1222 | 345 |
| Kerala | 3455 | 2365 | 400 |

| State |
|---|
| Punjab |
| Haryana |
| Kerala |

| State |
|---|
| 0 |
| 1 |
| 2 |