UNSW
AUSTRALIA

Regression

Never Stand Still

COMP9417: Machine Learning & Data Mining

Term 1, 2020

Adapted from slides by Dr Michael Bain

Administration

- Lecturers:
    - Dr. Gelareh Mohammadi (Lecturer in Charge)
    - Dr. Yang Song

- Course admin:
    - Dr. Alexandra Vassar

- Tutors: Peng Yi, Payal Bawa, Deepansh, Ce Song, Xuefeng Li, Andrew Lau, Anant Mathur, Zhengyue Liu

Communication

- Course website: Moodle
- Email: cs9417@cse.unsw.edu.au (use your UNSW email for all correspondence)
- Moodle forums
- Consult with your tutor in the time slots
- Consultation

Lectures & Tutorials

- Tutorials start in week 2 (No tutorials this week)
- There is no lectures or tutorials in week 6 (term break)
- Attend your allocated tutorial session
- The assignments and homework are in Python
- References: a list is provided in the course outline

Assessments

- Homework 1 (5%)
  - Due in week 4
- Homework 2 (5%)
  - Due in week 7
- Assignment (group project) (30%)
  - Due in week 10
- Final Exam (60%)

Late submission penalties will apply to all assessable works.

All submissions go through Turnitin and will be checked for plagiarism.

A Brief Course Introduction

Overview

This course will introduce you to machine learning, covering some of the core ideas, methods and theory currently used and understood by practitioners, including, but not limited to:

- Categories of learning (supervised / unsupervised learning, etc.)
- Widely-used machine learning techniques and algorithms
- Batch vs. online settings
- Parametric vs. non-parametric approaches
- Generalisation in machine learning
- Training, validation and testing phases in applications

What we will cover

- Core algorithms and model types in machine learning
- Foundational concepts regarding learning from data
- Relevant theory to inform and generalise understanding
- Practical applications

What we will NOT cover

- Probability and statistics
- Lots of neural nets and deep learning
- “big data”
- Ethical aspects of AI and ML

although all of these are interesting and important topics!

Some definitions

The field of machine learning is concerned with the question of how to construct computer programs that automatically improve from experience.

“Machine Learning”. T. Mitchell (1997)

Machine learning, then, is about making computers modify or adapt their actions (whether these actions are making predictions, or controlling a robot) so that these actions get more accurate, where accuracy is measured by how well the chosen actions reflect the correct ones.

“Machine Learning”. S. Marsland (2015)

Some definitions

The term machine learning refers to the automated detection of meaningful patterns in data.

"Understanding Machine Learning". S. Shwartz and S. David (2014)

Data mining is the extraction of implicit, previously unknown, and potentially useful information from data.

"Data Mining". I. Witten et al. (2016)

Trying to get programs to work in a reasonable way to predict stuff.

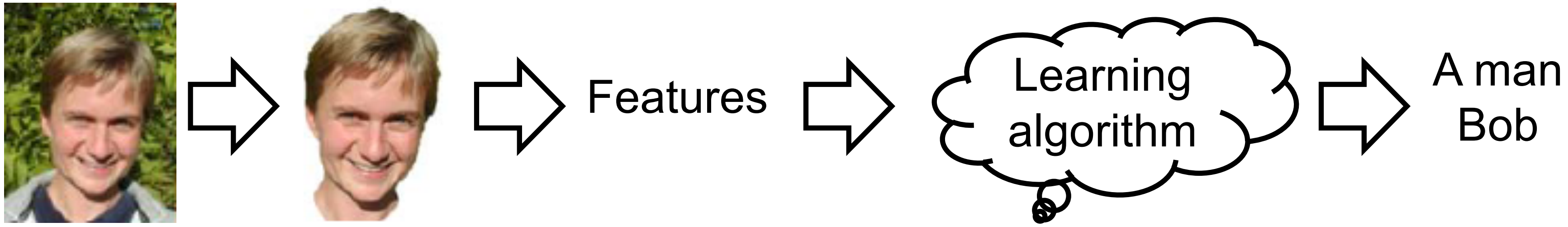
R. Kohn (2015)

Examples

- Object recognition / object classification
- Text classification (e.g. spam/non-spam emails)
- Speech recognition
- Event detection
- Recommender systems
- Human behaviour recognition (emotions, state of mind, etc.)
- Automatic medical diagnosis

Example

Simple face recognition pipeline:

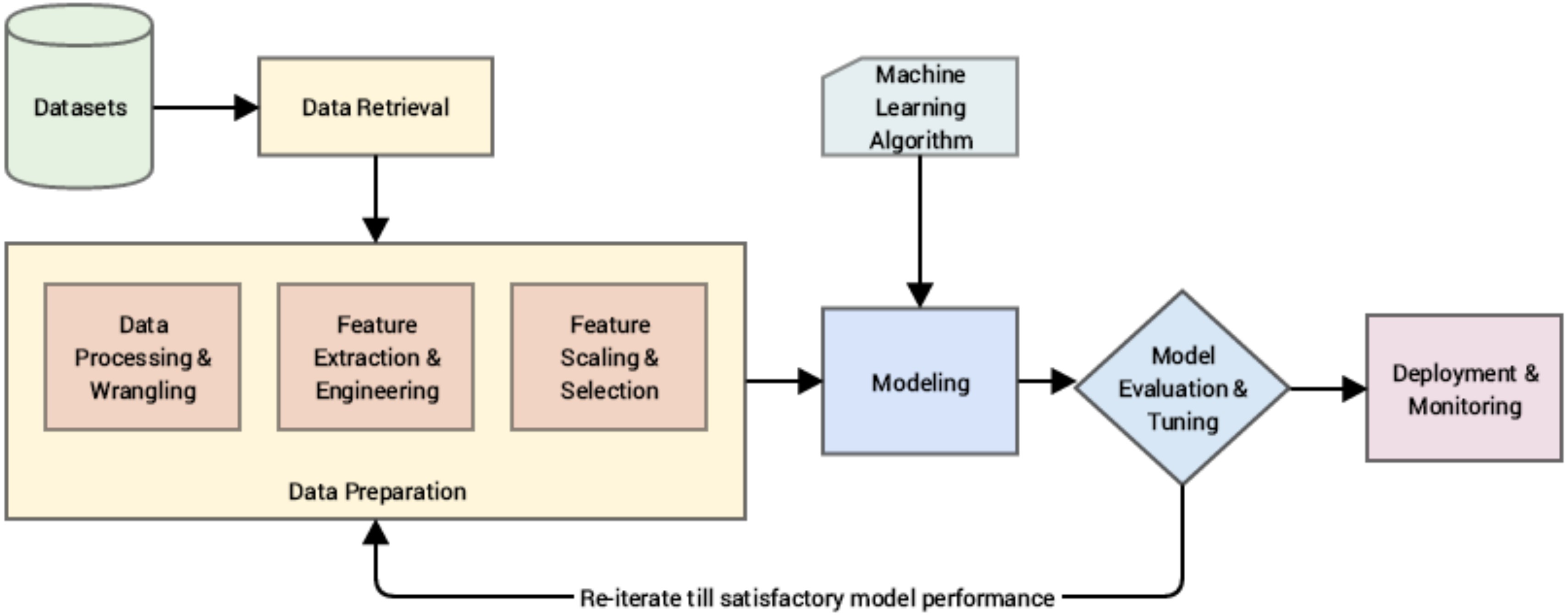


Features

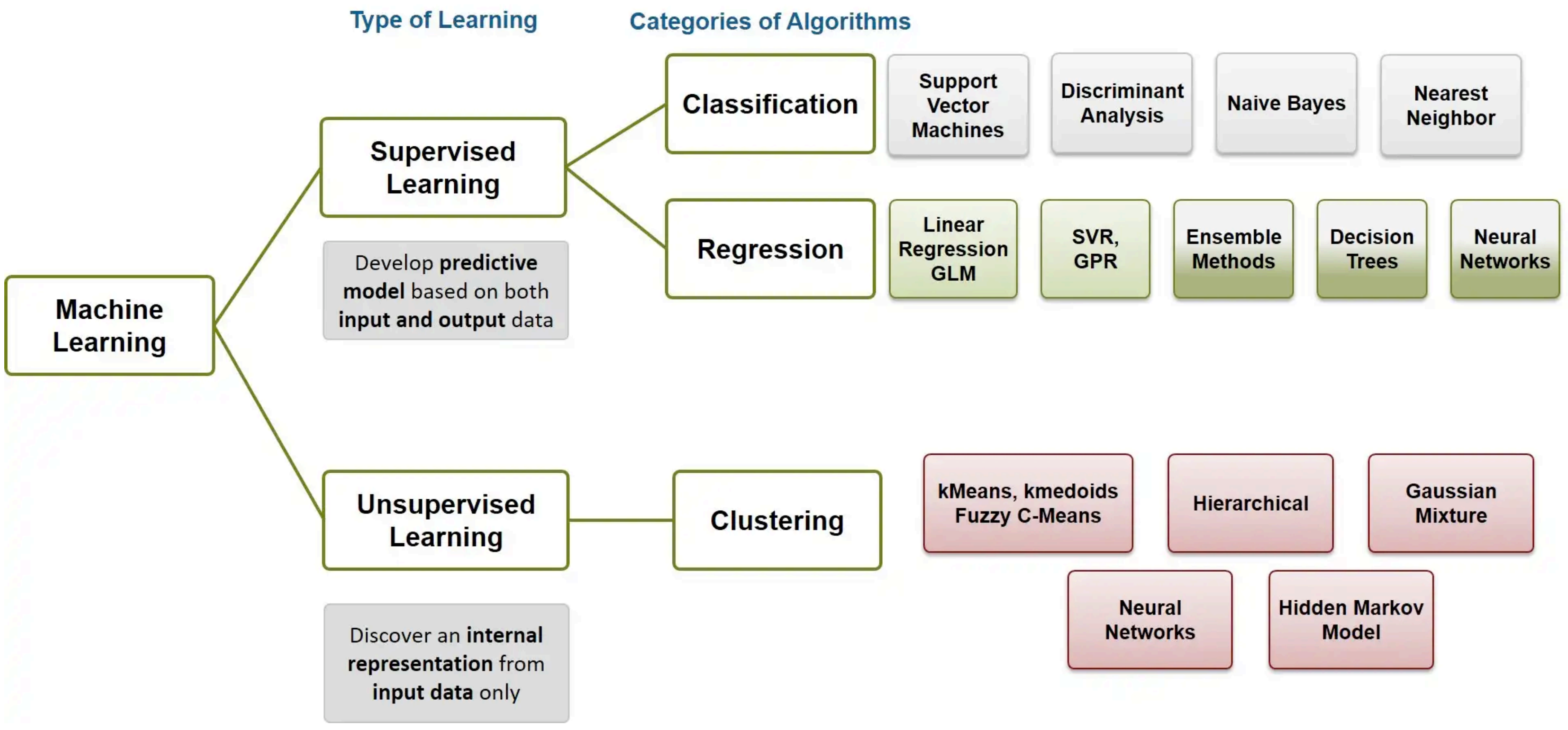
Learning algorithm

A man
Bob

Machine learning pipeline


Datasets
Data Retrieval
Machine Learning Algorithm
Data Processing & Wrangling
Feature Extraction & Engineering
Feature Scaling & Selection
Data Preparation
Modeling
Model Evaluation & Tuning
Deployment & Monitoring
Re-iterate till satisfactory model performance

Supervised and unsupervised learning



Type of Learning

Categories of Algorithms

Machine Learning

Supervised Learning

Develop **predictive model** based on both **input and output** data

Unsupervised Learning

Discover an **internal representation** from **input data** only

Supervised and unsupervised learning

The most widely used categories of machine learning algorithms are:

- Supervised learning – output class (or label) is given
- Unsupervised learning – no output class is given

There are also hybrids, such as semi-supervised learning, and alternative strategies to acquire data, such as reinforcement learning and active learning.

Note: output class can be real-valued or discrete, scalar, vector, or other structure . . .

Supervised and unsupervised learning

Supervised learning tends to dominate in applications.

Why ?

Supervised and unsupervised learning

Supervised learning tends to dominate in applications because generally, it is much easier to define the problem and develop an error measure (loss function) to evaluate different algorithms, parameter settings, data transformations, etc. for supervised learning than for unsupervised learning.

Supervised and unsupervised learning

Unfortunately . . .

In the real world it is often difficult to obtain good labelled data in sufficient quantities

So in such cases unsupervised learning is really what you want . . .

but currently, finding good unsupervised learning algorithms for complex machine learning tasks remains a research challenge.

Aims

This lecture will introduce you to machine learning approaches to the problem of numerical prediction. Following it you should be able to reproduce theoretical results, outline algorithmic techniques and describe practical applications for the topics:

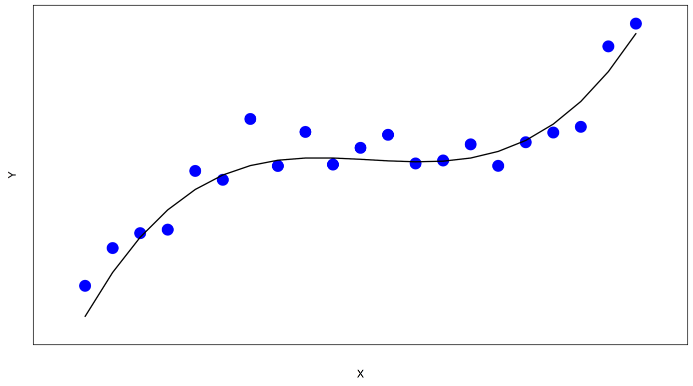
- the supervised learning task of numeric prediction
- how linear regression solves the problem of numeric prediction
- fitting linear regression by least squares error criterion
- non-linear regression via linear-in-the-parameters models
- parameter estimation for regression
- local (nearest-neighbour) regression

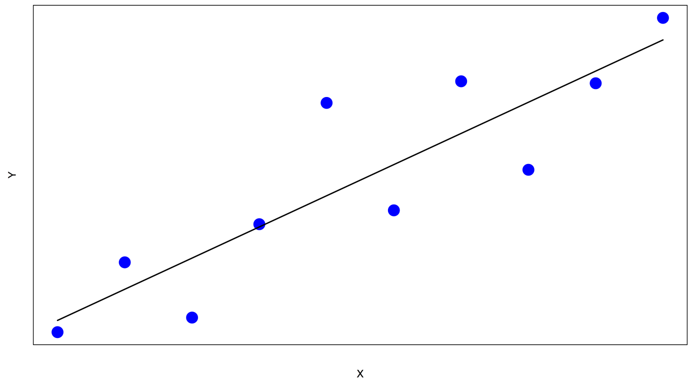
Introduction to Regression

Introduction to Regression

The “most typical” machine learning approach is to apply supervised learning methods for classification, where the task is to learn a model to predict a discrete value for data instances . . .

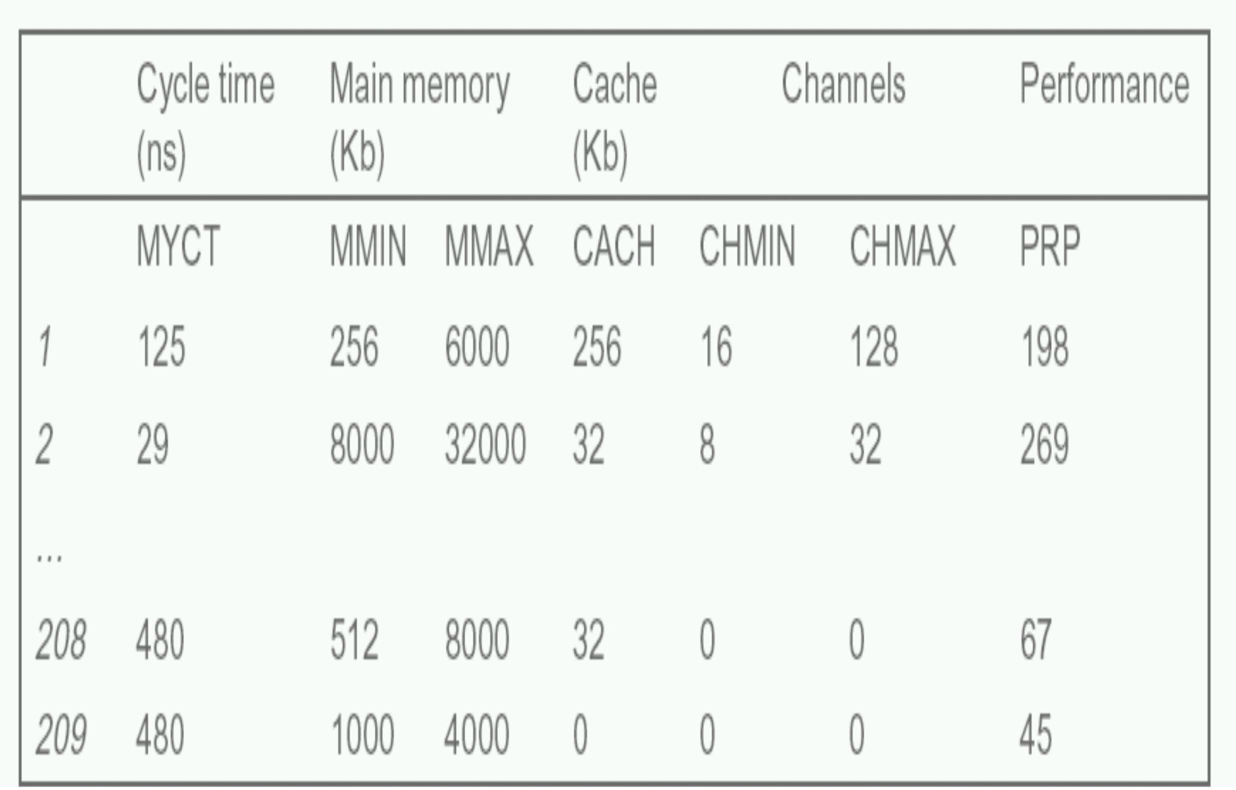
. . . however, we often find tasks where the most natural representation is that of prediction of numeric values. So we need regression in those tasks





Introduction to Regression

Example – task is to learn a model to predict CPU performance from a dataset of example of 209 different computer configurations:



	Cycle time (ns)	Main memory (Kb)		Cache (Kb)	Channels		Performance
	MYCT	MMIN	MMAX	CACH	CHMIN	CHMAX	PRP
1	125	256	6000	256	16	128	198
2	29	8000	32000	32	8	32	269
...							
208	480	512	8000	32	0	0	67
209	480	1000	4000	0	0	0	45

Introduction to Regression

Result: a linear regression equation fitted to the CPU dataset.

$$
\begin{aligned}
PRP \quad = \quad & -\ 56.1 \\
& +\ 0.049\ MYCT \\
& +\ 0.015\ MMIN \\
& +\ 0.006\ MMAX \\
& +\ 0.630\ CACH \\
& -\ 0.270\ CHMIN \\
& +\ 1.46\ CHMAX
\end{aligned}
$$

Introduction to Regression

For the class of symbolic representations, machine learning is viewed as:

searching a space of **hypotheses** . . .

represented in a formal hypothesis language (trees, rules, graphs . . .).

Introduction to Regression

For the class of numeric representations, machine learning is viewed as:

“searching” a space of **functions** . . .

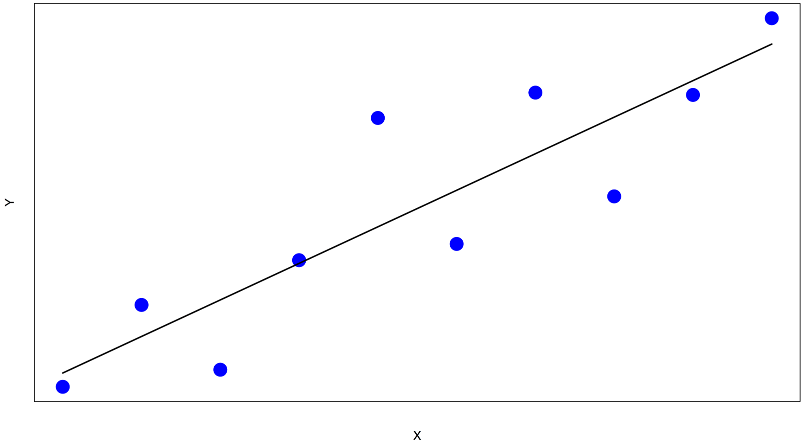
represented as mathematical models (linear equations, neural nets, . . .).

Note: in both settings, the models may be probabilistic . . .

Learning Linear Regression Models

Linear Regression

In linear regression, we assume there is a linear relationship between the output and feature(s)/input variable(s); this means the expected value of the output given an input, $E[y|x]$ is linear in input x



Example with one variable: $\hat{y} = bx + c$

Problem: given the data, estimate b and c

Linear regression

- Training instances: $\langle x_j, y_j \rangle, j = 1, \dots, m$
- x_j is the input
- y_j is the output to be predicted
- Each input has n attributes/features $x_j = [x_{j1}, x_{j2}, \dots, x_{jn}]^T$
- x and y are a general observation with $x = [x_1, x_2, \dots, x_n]^T$
- $\theta = [\theta_0, \theta_1, \dots, \theta_n]^T$
- $\mathbb{y}$ is the vector of outputs:$\mathbb{y} = [y_1, \dots, y_m]^T$
- X is the matrix of inputs where each row is one observation

Linear regression

- Linear regression can be used for numeric prediction from numeric attributes

- In linear models, the outcome is a **linear combination of attributes**:

$$\hat{y} = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_n x_n = h(x)$$

- **Weights** are estimated **from the observed data** (training data)

- **Predicted value** for the first training instance x_i is:

$$\hat{y}_1 = \theta_0 + \theta_1 x_{11} + \theta_2 x_{12} + \cdots + \theta_n x_{1n} = \sum_{i=0}^{n} \theta_i x_{1i} = x_1^T \theta = h(x_1)$$

To simplify the notation, we set $x_0 = 1$ and define $x = [x_0, x_1, \ldots, x_n]^T$

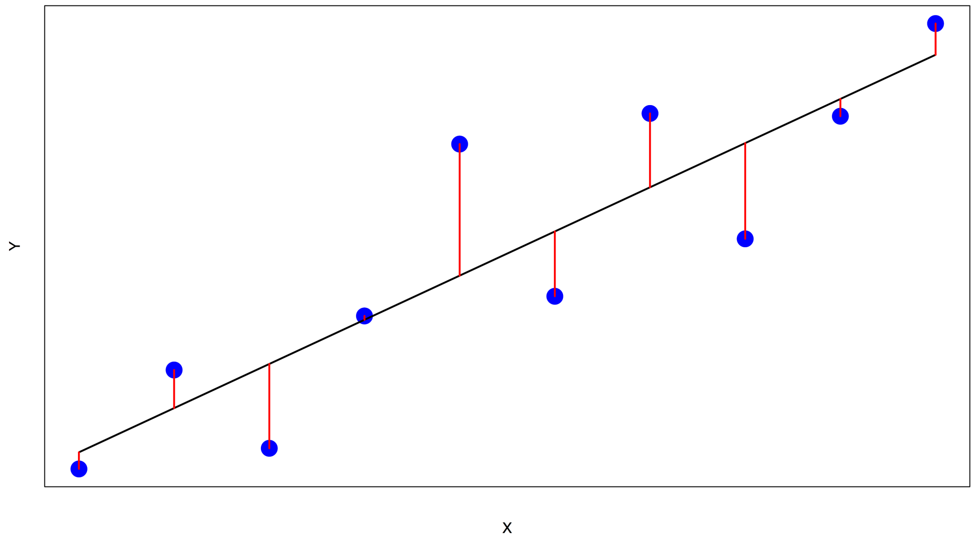
Linear regression

- **Univariate regression**: one input variable/feature/attribute is used to predict one output variable
- **Multiple regression**: more than one variables/features/attributes are used to predict one output variable

Linear Regression

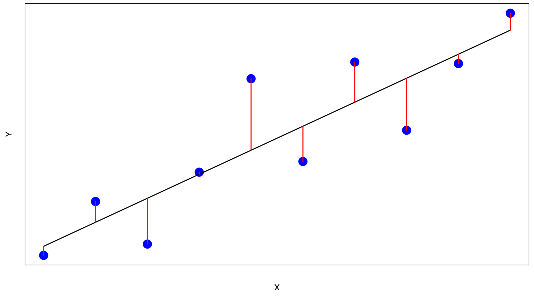
Infinite number of lines can be fitted, depending on how we define the best fit criteria

…but the most popular estimation model is “Least Squares”, also known as “Ordinary Least Squares” (OLS) regression



Minimizing Error

- Error = Difference between the predicted value and the actual value



- We want to minimize the error over all samples!
- We define the total error as the sum of squared errors and searching for n+1 parameters to minimize that
- Sum of squared error for m samples/observations

$$J(\theta) = \sum_{j=1}^{m} (y_j - \sum_{i=0}^{n} \theta_i x_{ji})^2 = \sum_{j=1}^{m} (y_j - x_j^T \theta)^2 = (\mathbb{y} - X\theta)^T (\mathbb{y} - X\theta)$$

$J(\theta)$ is called cost function or loss function. This concept generalizes to all functions that measure the distance between the predicted and true output. (in OLS framework, it is also called Residual Sum of Squares (RSS))

Minimizing Squared Error

This cost function can be also written as:

$$J(\theta) = \frac{1}{m}\sum_{j=1}^{m}(y_j - x_j^T\theta)^2$$

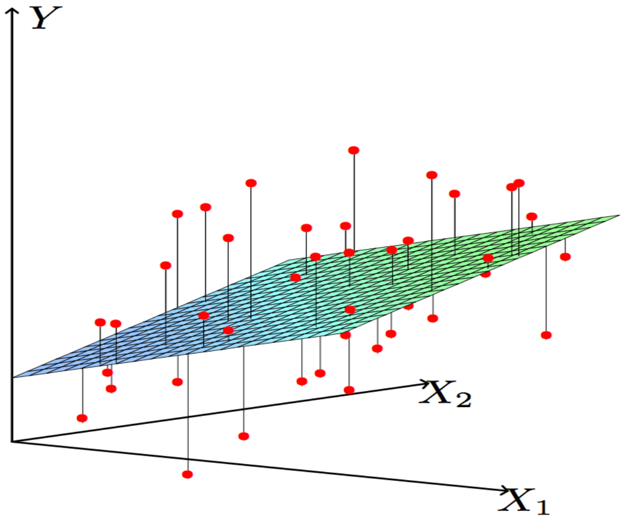
Which is the average of squared error and minimizing it, leads to the same θ, that we get with $J(\theta)$ without taking the average.

The $\frac{1}{m}\sum_{j=1}^{m}(y_j - x_j^T\theta)^2$ is called mean squared error or briefly MSE.

Minimizing Squared Error

Multiple regression example:

Given 2 real-values variables/attributes x_1, x_2, labelled with a real-valued variable y, find “plane of best fit” that captures the dependency of y on x_1 and x_2.



Again we can use MSE to find the best plane that fits the data.

For more than two variables, we find the best hyper-plane.

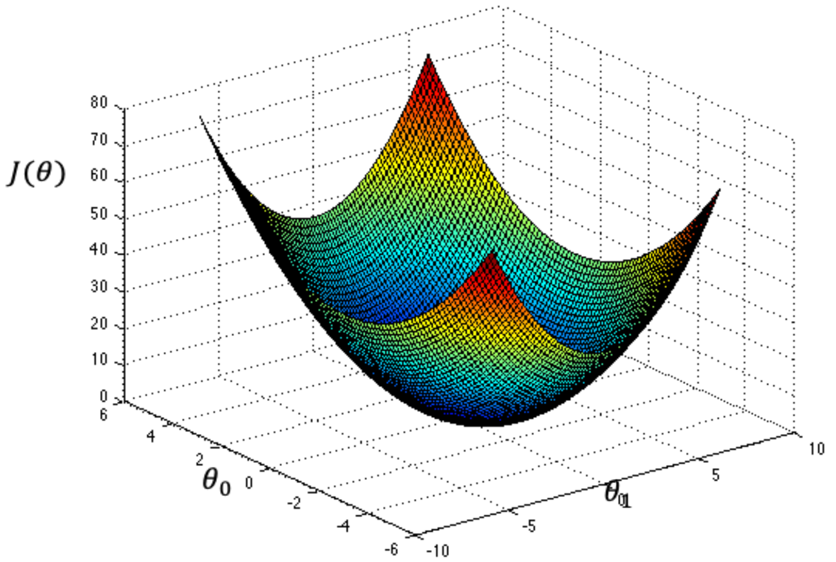
Least Squares Regression

Question: How to estimate the parameters such to minimize the cost function $J(\theta)$?

If you compute $J(\theta)$ for different values of θ in linear regression problem, you will end up with a convex function which in this particular problem has one global minima.

…and we can use a search algorithm which starts with some "initial guess" for θ and repeatedly changes θ to make $J(\theta)$ smaller, until it converge to minimum value.

One of such algorithms is gradient descent



Gradient Descent

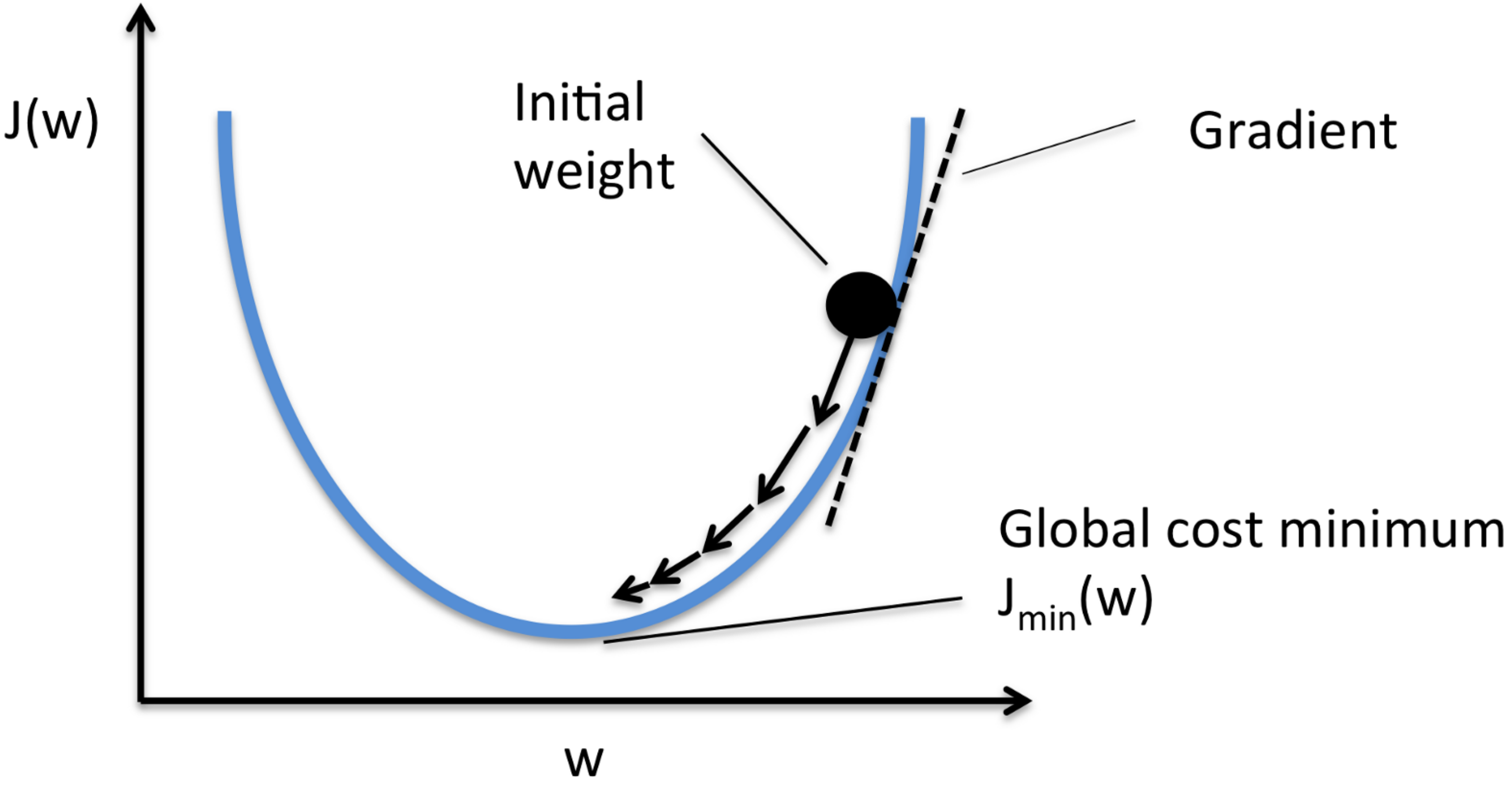
Gradient descent starts with some initial θ, and repeatedly performs an update:

$$\theta_i := \theta_i - \alpha \frac{\partial}{\partial \theta_i} J(\theta)$$

α is the learning rate. This algorithm takes a step in the direction of the steepest decrease in $J(\theta)$

To implement this algorithm, we have to work out the partial derivative term for $J(\theta) = \frac{1}{m} \sum_{j=1}^{m} (y_j - x_j^T \theta)^2$

Gradient Descent



From: Gradient Descent: All you need to know, by S. Suryansh

Gradient Descent

If we focus on only one sample out of m samples, then the cost function is:

$$J(\theta) = \ (y_j - h_\theta\left(x_j\right))^2 \ = \ (y_j - \sum_{i=1}^{n} x_{ji}\theta_i)^2$$

Taking the derivative will give:

$$\frac{\partial}{\partial\theta_i}J(\theta) = 2(h_\theta\left(x_j\right) - y_j)x_{ji}$$

So, for a **single** training sample, the update rule is:

$$\theta_i \coloneqq \theta_i + \alpha(y_j - h_\theta\left(x_j\right))x_{ji} \qquad \text{(for every } i\text{)}$$

The update rule for squared distance is called "**Least Mean Squares**" (LMS), also known as **Widrow-Hoff**

Gradient Descent

We looked at LMS rule for only a single sample. But what about other samples? There are two ways…

1. **Batch Gradient Descent:**

$$\theta_i = \theta_i + \alpha \sum_{j=1}^{m} (y_j - h_\theta(x_j)) x_{ji} \quad \text{(for every } i\text{)}$$

Replace the gradient with the sum of gradient for all samples and continue until convergence.

Convergence means that, the estimated θ will be stabilized.

Gradient Descent

2. Stochastic Gradient Descent:

```
$for\ j = 1\ \ to\ \ m\ \{$
    $\theta_i = \theta_i + \alpha(y_j - h_\theta(x_j))x_{ji}$   (for every $i$)
$\}$
```

Repeat this algorithm until convergence.

What this algorithm does?

Gradient Descent

In stochastic gradient descent, θ gets updated at any sample separately. This algorithm is much less costly than batch gradient descent, however it may never converge to the minimum.

Gradient Descent

In both algorithms:

- You can start with an arbitrary (random) values for your parameters θ_i (initialization)
- You have to be careful with the selection of learning rate, α, as a small value can make the algorithm very slow, and a big value may stop the algorithm from convergence

Minimizing Squared Error (normal equations)

Gradient descent is one way of minimizing our cost function $J(\theta)$ which is an iterative algorithm.

But maybe you can find the minimum of $J(\theta)$ explicitly by taking its derivatives and setting them to zero. This is also called exact or closed-form solution:

$$\frac{\partial}{\partial\theta}J(\theta) = 0$$

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

$$\frac{\partial}{\partial\theta}J(\theta) = -2X^T(\mathbb{y} - X\theta) = 0$$

$$X^T(\mathbb{y} - X\theta) = 0$$

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

Minimizing Squared Error (normal equations)

Here is how matrix X and vector $\mathbb{y}$ look like:

$$X = \begin{bmatrix} 1 & x_{11} & x_{12} & \dots & x_{1n} \\ 1 & x_{21} & x_{22} & \dots & x_{2n} \\ & & \vdots & & \\ 1 & x_{m1} & x_{m2} & \dots & x_{mn} \end{bmatrix} \qquad \mathbb{y} = \begin{bmatrix} y_1 \\ y_2 \\ \\ y_m \end{bmatrix}$$

You have to be careful to **add a column of ones** in X to count for the intercept parameter.