Never Stand Still

Regression

COMP9417 Machine Learning & Data Mining

Term 3, 2019

Adapted from slides by Dr Michael Bain

Administration

- Lecturers:
  - Dr. Gelareh Mohammadi (Lecturer in Charge)
  - Dr. Yang Song
- Course admin:
  - Anahita Namvar
- Tutors: Dr Alfred Krzywicki, Anahita Namvar, Payal Bawa

Communication

- Course website: Moodle
- Email: cs9417@cse.unsw.edu.au (use your UNSW email for all correspondence)
- Moodle forum
- 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 4 (the Monday of that week is a public holiday)
- Attend your allocated tutorial session
- The assignments ang 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

- lots of probability and statistics
- lots of neural nets and deep learning
- “big data”
- commercial and business aspects of “analytics”
- 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.

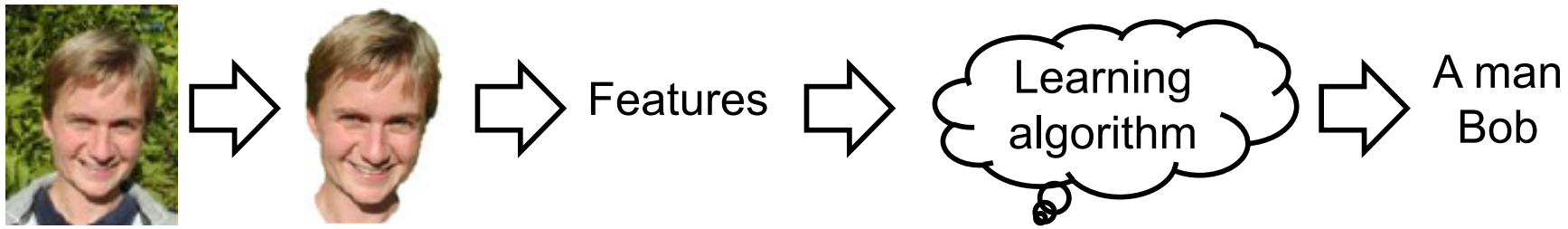
R. Kohn (2015)

Examples

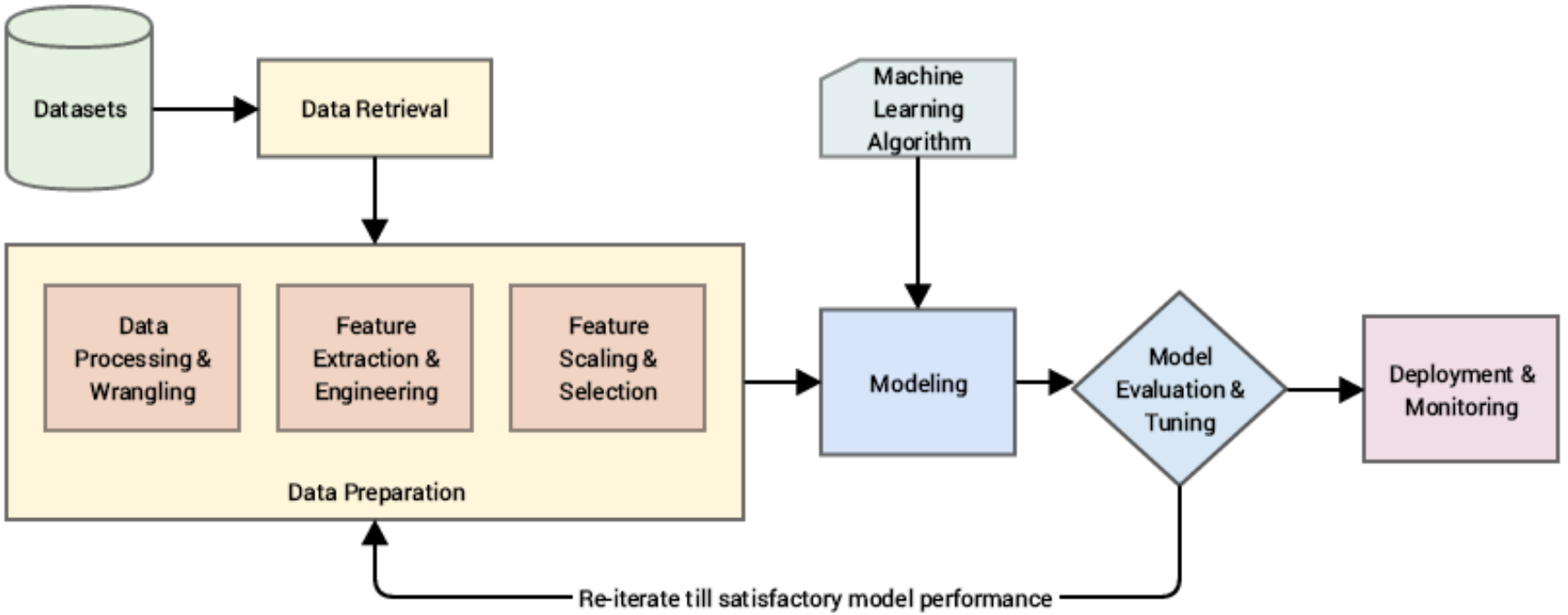
- Object recognition / object classification
- Text classification (e.g. spam/non-spam emails)
- Speech recognition
- Event detection
- Recommender systems
- Human behaviour recognition (emotions, state of mind, etc.)
- Automatic medical diagnosis

Example

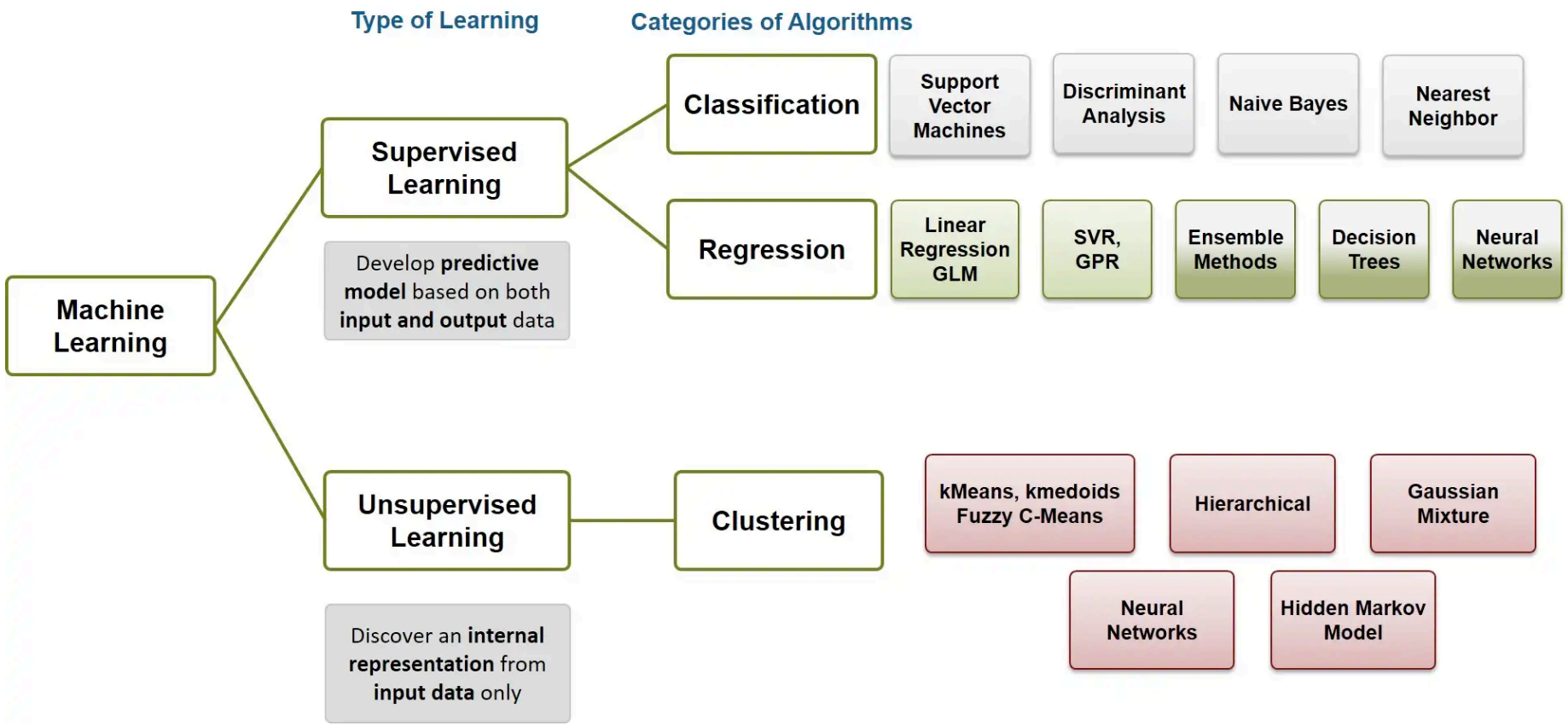
Simple face recognition pipeline:



Machine learning pipeline



Supervised and unsupervised learning



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 ?

Generally, because 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:

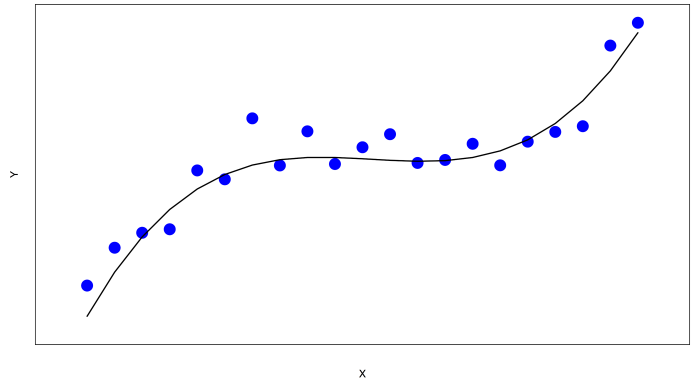
- the supervised learning task of numeric prediction
- how linear regression solves the problem of numeric prediction
- fitting linear regression by least squares error criterion
- non-linear regression via linear-in-the-parameters models
- parameter estimation for regression
- local (nearest-neighbour)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

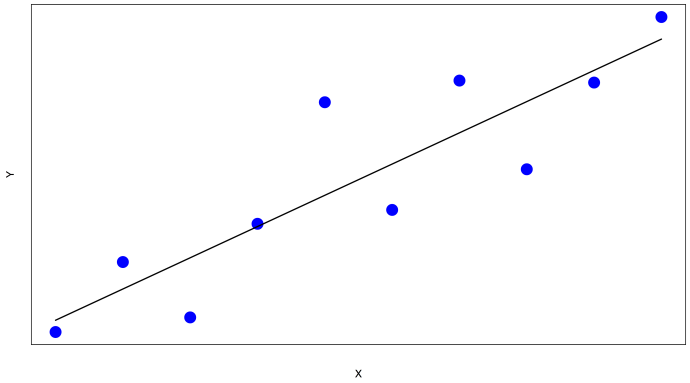
Introduction to Regression

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.

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

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** . . .

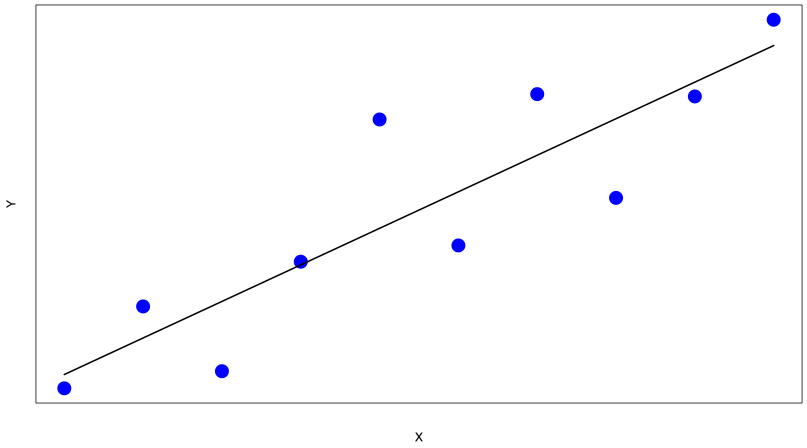
represented as mathematical models (linear equations, neural nets, . . .).

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 features; this means the expected value of the output given an input $E[y|X]$ is linear in the input $X = [x_1, \dots, x_n]$



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

Problem: given the data, estimate b

Note: $x_1, \dots, x_n$ are called with different names like attributes, features, covariates, predictors, independent variables, etc.

y is called output or dependent variable.

Note: each sample j has n attributes and is denoted as $X_j = [\, x_1^{(j)}, \dots, x_n^{(j)}]$

Linear Regression

- Numeric attributes and numeric prediction, i.e., regression
- Linear models, i.e. outcome is *linear* combination of attributes:

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

- Weights are calculated from the training data
- **Predicted** value for first training instance $x^{(1)}$ is:

$$\theta_0 x_0^{(1)} + \theta_1 x_1^{(1)} + \cdots + \theta_n x_n^{(1)} = \sum_{i=1}^{n} \theta_i x_i^{(1)} = X\theta = h(X)$$

$$\theta = [\theta_0, \ldots, \theta_n]^T$$

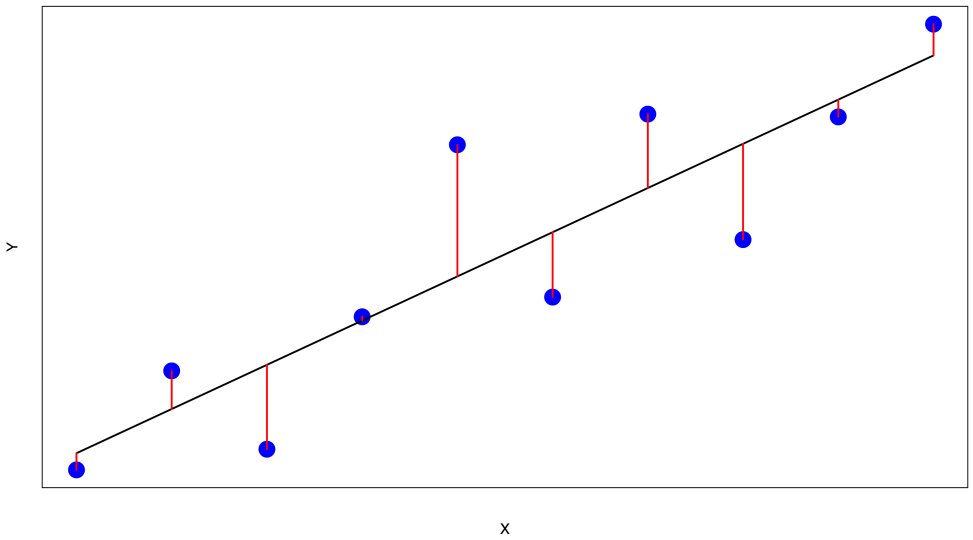
$$X = [x_0, \ldots, x_n]$$

Note: We can define/add $x_0 = 1$ for simplicity

Linear Regression

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

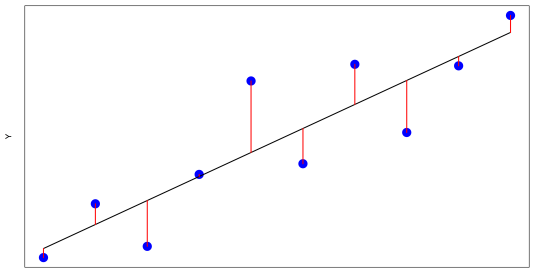
…but the most popular estimation model is “Least Squares”, also known as “Ordinary Least Squares” (OLS) regression



Minimizing Squared Error

Difference between predicted and actual values is the error !

$n + 1$ coefficients are chosen so that sum of squared error on all instances in training data is minimized.



Squared error for m samples/observations:

$$J(\theta) = \sum_{j=1}^{m} (y^{(j)} - \sum_{i=1}^{n} \theta_i x_i^{(j)})^2 = \sum_{j=1}^{m} (y^{(j)} - X\theta)^2 = (y - X\theta)^T (y - X\theta)$$

$J(\theta)$ is called cost function or loss function. This concept generalises to all functions that measures the distance between the predicted and true output. (in the OLS framework, it is also called Residual Sum of Squares (RSS)).This is also called mean squared error (MSE) if gets divided

Minimizing Squared Error

This cost function can be also written as:

$$J(\theta) = \frac{1}{m}\sum_{j=1}^{m}(y^{(j)} - X\theta)^2$$

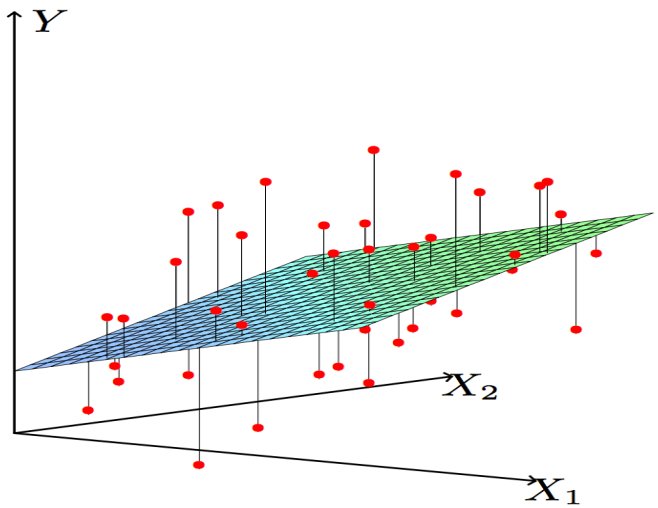
Which is the average of squared error and minimizing it, leads to the same θ, that we get with $J(\theta)$ without taking the average.

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

Minimizing Squared Error

Multivariable example:

Given 2 real-valued variables 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, x_2.



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

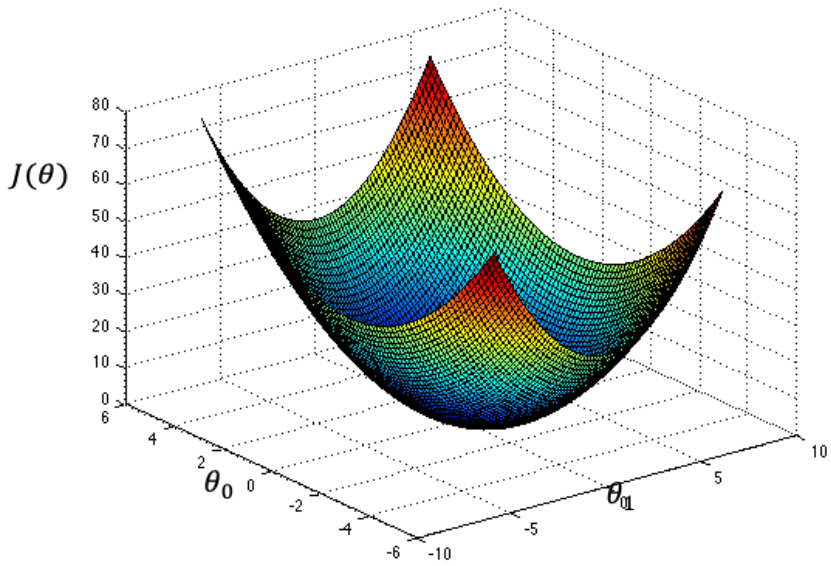
For more than 2 variables, we have to estimate a hyper-plane.

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

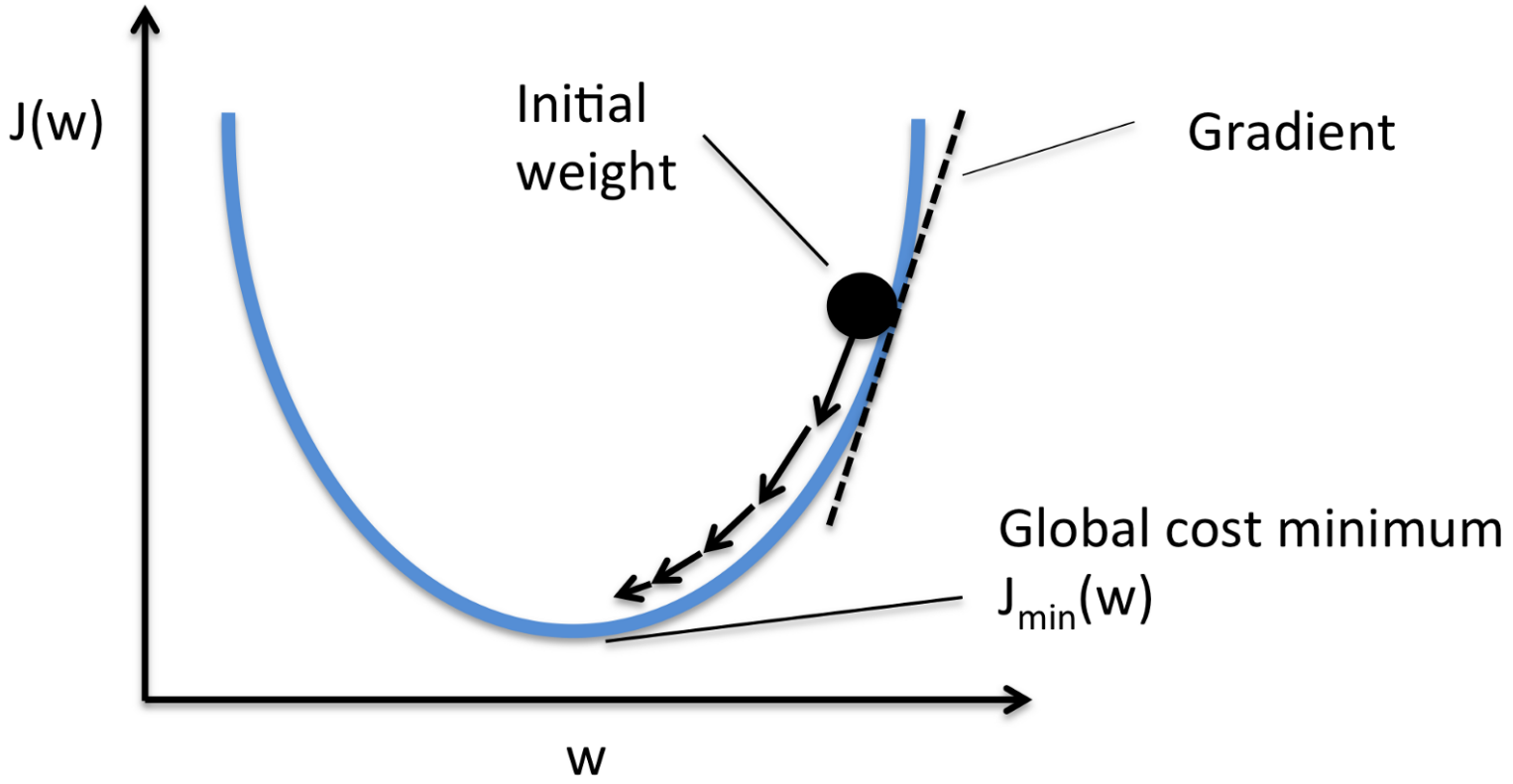
Gradient descent starts with some initial θ, and repeatedly performs an update:

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

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

To implement this algorithm, we have to work out the partial derivative term for $J(\theta) = \sum_{j=1}^{m} (y^{(j)} - \sum_{i=1}^{n} \theta_i x_i^{(j)})^2 = \sum_{j=1}^{m} (y^{(j)} - h_\theta(X^{(j)}))$

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, the cost function is:

$$J(\theta) = (y^{(j)} - h(X^{(j)}))^2$$

And take the derivative, then:

$$\frac{\partial}{\partial \theta_i} J(\theta) = 2(h_\theta(X^{(j)}) - y^{(j)}) x_i^{(j)}$$

So for **single** training example, the update rule is:

$$\theta_i := \theta_i + \alpha \left(y^{(j)} - h_\theta\left(X^{(j)}\right) \right) x_i^{(j)}$$

For squared distance, the update rule is called "**Least Mean Squares**" (LMS), also known as "**Windrow-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:

$$\theta_i := \theta_i + \alpha \sum_{j=1}^{m} \left(y^{(j)} - h_\theta\left(X^{(j)}\right) \right) x_i^{(j)} \quad \text{(for every } i\text{)}$$

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:

$$\text{for j = 1 to m \{}$$

$$\theta_i := \theta_i + \alpha \left(y^{(j)} - h_\theta \left(X^{(j)} \right) \right) x_i^{(j)} \quad \text{(for every } i\text{)}$$

$$\}$$

Repeat this algorithm until convergence.

The difference with batch gradient descent is that, here, we update θ 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 algorithm:

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

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

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

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

$$\boxed{\theta = (X^TX)^{-1}X^Ty}$$

Minimizing Squared Error (normal equations)

Here is how your matrices of X and y look like:

$$X = \begin{bmatrix} 1 & x_1^1 & x_2^1 & x_3^1 & \dots & x_n^1 \\ 1 & x_1^2 & x_2^2 & x_3^2 & \dots & x_n^2 \\ & & & \cdot & & \\ & & & \cdot & & \\ & & & \cdot & & \\ 1 & x_1^m & x_2^m & x_3^m & \dots & x_n^m \end{bmatrix} \quad \mathrm{y} = \begin{bmatrix} y^1 \\ y^2 \\ \cdot \\ \cdot \\ \cdot \\ y^m \end{bmatrix}$$

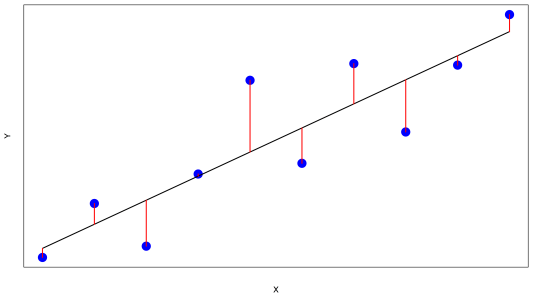
You have to add a **column of ones** in X to count for intercept parameter

Minimizing Squared Error (probabilistic interpretation)

We can write the relation between input variable X and output variable y as:

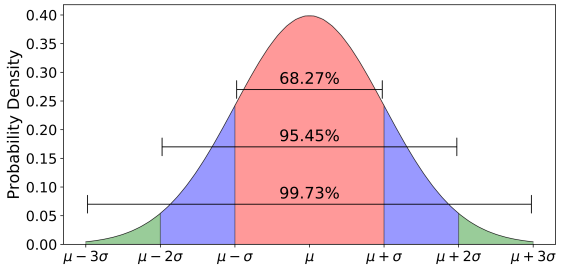
$$y^{(j)} = X^{(j)}\theta + \epsilon^{(j)}$$

and $\epsilon^{(j)}$ is an error term which might be unmodeled effect or random noise. Let's assume $\epsilon^{(j)}$ are independently and identically distributed (i.i.d) according to a Gaussian distribution:



$$\epsilon^{(j)} \sim \mathrm{N}(0, \sigma^2)$$

$$p(\epsilon^{(j)}) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(\epsilon^{(j)})^2}{2\sigma^2}\right)$$



Minimizing Squared Error (probabilistic interpretation)

This is equivalent to:

$$p\left(y^{(j)}\middle|X^{(j)};\theta\right) = \frac{1}{\sqrt{2\pi\sigma^2}}\exp\left(-\frac{(y^{(j)} - X^{(j)}\theta)^2}{2\sigma^2}\right)$$

We want to estimate θ such that we maximize the probability of output y given input X over all m training samples:

$$L(\theta) = p(y|X;\theta)$$ (this called **likelihood** function)

Since we assumed that our samples are independent, we can write $L(\theta)$ as follow:

$$L(\theta) = \prod_{j=1}^{m} p(\left(y^{(j)}\middle|X^{(j)};\theta\right)$$

$$= \prod_{j=1}^{m} \frac{1}{\sqrt{2\pi\sigma^2}}\exp\left(-\frac{(y^{(j)} - X^{(j)}\theta)^2}{2\sigma^2}\right)$$

Minimizing Squared Error (probabilistic interpretation)

Now, we have to estimate θ, such that it maximizes $L(\theta)$. This is called maximum likelihood.

We know (from math) that to find θ, we can also maximize any strictly increasing function of $L(\theta)$. In this case, it would be easier if we maximize the log likelihood $l(\theta)$:

$$l(\theta) = \log L(\theta) = \log \prod_{j=1}^{m} \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(y^{(j)} - X^{(j)}\theta)^2}{2\sigma^2}\right)$$

$$= m \log\left(\frac{1}{\sqrt{2\pi\sigma^2}}\right) - \frac{1}{\sigma^2}\frac{1}{2}\sum_{j=1}^{m}(y^{(j)} - X^{(j)}\theta)^2$$

So, maximizing $l(\theta)$ is equal to minimizing $\sum_{j=1}^{m}(y^{(j)} - X^{(j)}\theta)^2$ which is the squared error we defined earlier.

Minimizing Squared Error (probabilistic interpretation)

- This simply shows that under certain assumptions ($\epsilon^{(j)} \sim \mathrm{N}(0, \sigma^2)$ and i.i.d), the least-squared regression is equivalent to find the maximum likelihood estimate of θ.

- The value of σ^2 do not affect the choice of θ.

Step back: Statistical Techniques for Data Analysis

Probability vs Statistics: The Difference

Probability versus Statistics

- **Probability**: reasons from populations to samples
  - This is deductive reasoning, and is usually sound (in the logical sense of the word)
- **Statistics**: reasons from samples to populations
  - This is inductive reasoning, and is usually unsound (in the logical sense of the word)

Statistical Analyses

- Statistical analyses usually involve one of 3 things:
    1. The study of populations;
    2. The study of variation; and
    3. Techniques for data abstraction and data reduction
- Statistical analysis is more than statistical computation:
    1. What is the question to be answered?
    2. Can it be quantitative (i.e., can we make measurements about it)?
    3. How do we collect data?
    4. What can the data tell us?

Sampling

Where do the Data come from? (Sampling)

- For groups (populations) that are fairly homogeneous, we do not need to collect a lot of data. (We do not need to sip a cup of tea several times to decide that it is too hot.)
- For populations which have irregularities, we will need to either take measurements of the entire group, or find some way of get a good idea of the population without having to do so
- *Sampling* is a way to draw conclusions about the population without having to measure all of the population. The conclusions need not be completely accurate
- All this is possible if the sample closely resembles the population about which we are trying to draw some conclusions

What We Want From a Sampling Method

- No systematic bias, or at least no bias that we cannot account for in our calculations
- The chance of obtaining an unrepresentative sample can be calculated. (So, if this chance is high, we can choose not to draw any conclusions.)
- The chance of obtaining an unrepresentative sample decreases with the size of the sample

Estimation

Estimation from a Sample

- Estimating some aspect of the population using a sample is a common task. Along with the estimate, we also want to have some idea of the accuracy of the estimate (usually expressed in terms of confidence limits)
- Some measures calculated from the sample are very good estimates of corresponding population values. For example, the sample mean m is a very good estimate of the population mean μ. But this is not always the case. For example, the range of a sample usually under-estimates the range of the population
- We will have to clarify what is meant by a “good estimate”. One meaning is that an estimator is correct on average. For example, on average, the mean of a sample is a good estimator of the mean of the population

Estimation from a Sample

- For example, when a number of samples are drawn and the mean of each is found, then average of these means is equal to the population mean
- Such an estimator is said to be *statistically unbiased*

Estimates of the Mean and Variance

Mean: This is calculated as follows.

- Find the total T of N observations. Estimate the (arithmetic) mean by $m = T/N$.
- This works very well when the data follow a symmetric *bell-shaped frequency distribution* (of the kind modelled by "normal" distribution)
- A simple mathematical expression of this is $m = \frac{1}{N}\sum_i x_i$, where the observations are $x_1, x_2, \ldots, x_N$
- If we can group the data so that the observation x_1 occurs f_1 times, x_2 occurs f_2 times and so on, then the mean is calculated even easier as $m = \frac{1}{N}\sum_i x_i f_i$

Estimates of the Mean and Variance

- If, instead of frequencies, you had relative frequencies (i.e. instead of f_i you had $p_i = \frac{f_i}{N}$), then the mean is simply the observations weighted by relative frequency. That is, $m = \sum_i x_i \, p_i$

- We want to connect this up to computing the mean value of observations modelled by some theoretical probability distribution function. That is, we want to a similar counting method for calculating the mean of random variables modelled using some known distribution

Estimates of the Mean and Variance

- Correctly, this is the mean value of the values of the random variable function. But this is a bit cumbersome, so we will just say the "mean value of the r.v." For discrete r.v.'s this is:

$$E(X) = \sum_i x_i p(X = x_i)$$

- **Variance:** This is calculated as follows:
- Calculate the total T and the sum of squares of N observations. The estimate of the standard deviation is $s = \sqrt{\frac{1}{N-1}\sum_i (x_i - m)^2}$
- Again, this is a very good estimate when the data are modelled by a normal distribution

Estimates of the Mean and Variance

- For grouped data, this is modified to

$$s = \sqrt{\frac{1}{N-1}\sum_i (x_i - m)^2 f_i}$$

- Again, we have a similar formula in terms of expected values, for the scatter (spread) of values of a r.v. X around a mean value $E(X)$:

$$Var(X) = E((X - E(X))^2) = E(X^2) - [E(X)]^2$$

- You can remember this as “the mean of squares minus the square of means”

Correlation

- The formula for computing correlation between x and y is:

$$r = \frac{cov(x, y)}{\sqrt{var(x)}\sqrt{var(y)}}$$

This is sometimes also called *Pearson's correlation coefficient*

- The terms in the denominator are simply the standard deviations of x and y. But the numerator is different. This is the covariance, calculated as the average of the product of deviations from the mean:

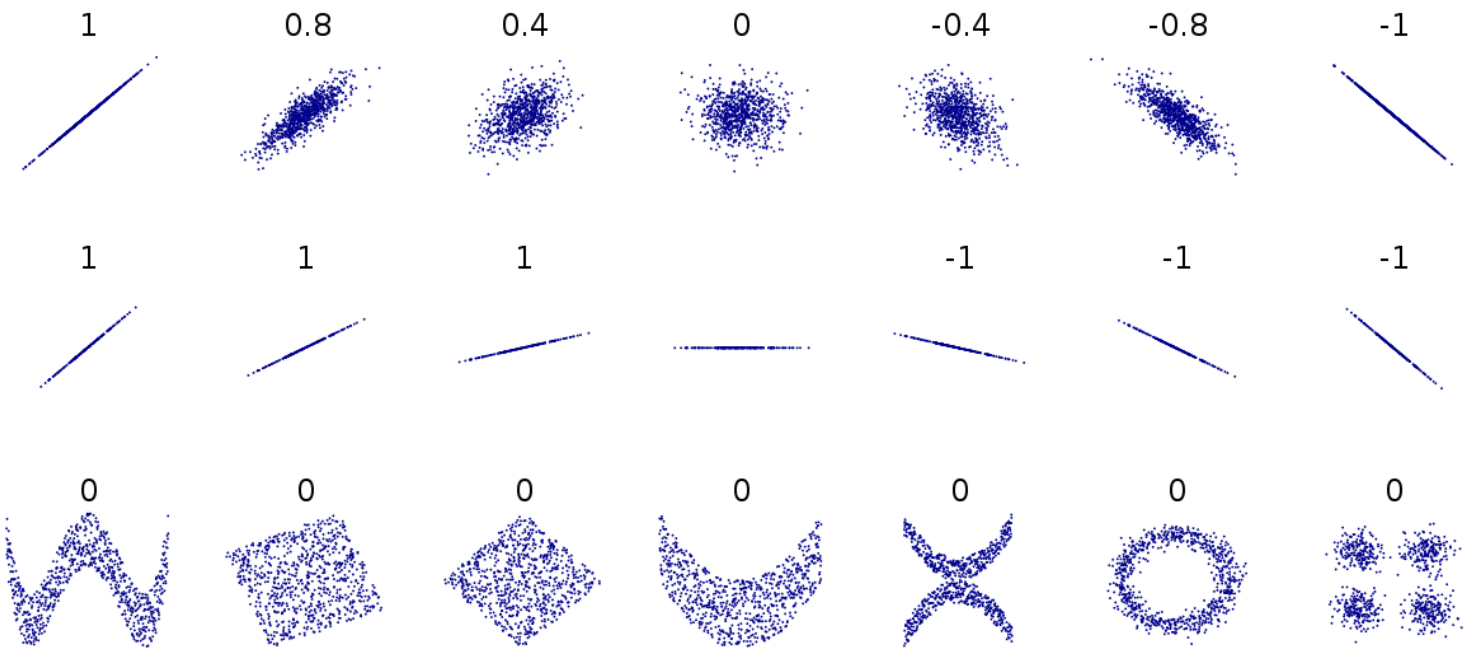
$$cov(x, y) = \frac{\sum_i (x_i - \bar{x})(y_i - \bar{y})}{N - 1}$$

$$r = \frac{\sum_i x_i y_i - n\bar{x}\bar{y}}{n - 1}$$

Correlation

- The same kinds of calculations can be done if the data were not actual values but ranks instead (i.e. ranks for x's and y's)
  - This is called *Spearman's rank correlation*, but we don't do these calculations here

Correlation



Pearson correlation in some sets of (x,y). [Wikipedia]

Covariance and Correlation

- The correlation coefficient is a number between -1 and +1 that indicates whether a pair of variables x and y are associated or not, and whether the scatter in the association is high or low
  - High values of x are associated with high values of y and low values of x are associated with low values of y, and scatter is low
  - A value near 0 indicates that there is no particular association and that there is a large scatter associated with the values
  - A value close to -1 suggests an inverse association between x and y
- Only appropriate when x and y are roughly linearly associated (doesn't work well when the association is curved)

What Does Correlation Mean?

- r is a quick way of checking whether there is some linear association between x and y
- The sign of the value tells you the direction of the association
- All that the numerical value tells you is about the scatter in the data
- The correlation coefficient does not model any relationship. That is, given a particular x you cannot use the r value to calculate a y value
  - It is possible for two datasets to have the same correlation, but different relationships
  - It is possible for two datasets to have different correlations but the same relationship
- MORAL: Do not use correlations to compare datasets. All you can derive is whether there is a positive or negative relationship between x and y
- ANOTHER MORAL: Do not use correlation to imply x causes y or the other way around

Regression

Univariate linear regression: (only one independent variable)

Goal: fit a line such that $\hat{y} = \theta_0 + \theta_1 x$

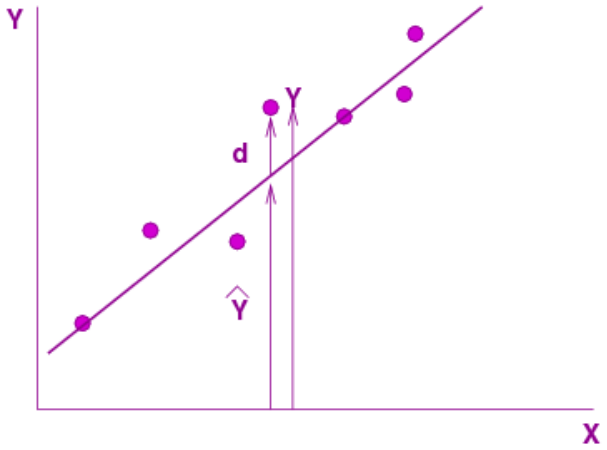
How: minimize $J(\theta) = \sum_j (y_j - \hat{y}_j)^2$ (Least squares estimator)

- If we expand the explicit solution we saw before, we can see:

$$\theta_1 = \frac{cov(x, y)}{var(x)}$$

Where $cov(x, y)$ is the covariance of x and y

- and $\theta_0 = \bar{y} - \theta_1 \bar{x}$



Meaning of the coefficients

- θ_1: change in y that accompanies a unit change in x
- If the values of x were assigned at random, then θ_1 estimates the unit change in y caused by a unit change in x
- If the values of x were not assigned at random (for example, they were data somebody observed), then the change in y will include the change in x and any other confounding variables that may have changed as a result of changing x by 1 unit. So, you cannot say for example, that a change of x by 1 unit causes θ_1 units of change in y
- $\theta_1 = 0$ means there is no linear relationship between x and y , and then best we can do is simply say is $\theta_0 = \bar{y}$. Estimating the sample mean is therefore a special case of the MSE criterion.

Finding the parameters for univariate linear regression

Univariate linear regression

In univariate regression, we aim to find the relationship between y and one independent variable x.

Example:

Suppose we want to investigate the relationship between people's height and weight. We collect m height and weight measurements $(h_j, w_j), 1 \leq j \leq m$

Univariate linear regression assumes a linear equation $w = \theta_0 + \theta_1 h$, with parameters θ_0 and θ_1 chosen such that the sum of squared residuals $\sum_{j=1}^{m}(w_j - (\theta_0 + \theta_1 h_j))^2$ is minimised.

Univariate linear regression

In order to find the parameters we take partial derivatives, set the partial derivatives to 0 and solve for θ_0. As we saw before, this will lead to:

$$\theta_1 = \frac{cov(h,w)}{var(h)} = \frac{\sum_{j=1}^{m}(h_j - \bar{h})(w_j - \bar{w})}{\sum_{j=1}^{m}(h_j - \bar{h})^2}$$

$$\theta_0 = \bar{w} - \theta_1 \bar{h}$$

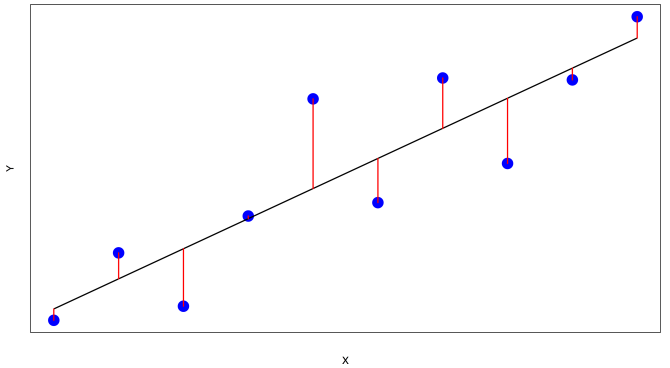
Linear regression: intuitions

- Adding a constant to all x-values (a translation) will affect only the intercept but not the regression coefficient (since it is defined in terms of deviations from the mean, which are unaffected by a translation).
- So we could zero-centre the x -values by subtracting x, in which case the intercept is equal to y .
- We could even subtract y from all y -values to achieve a zero intercept, without changing the problem in an essential way.

Linear regression: intuitions

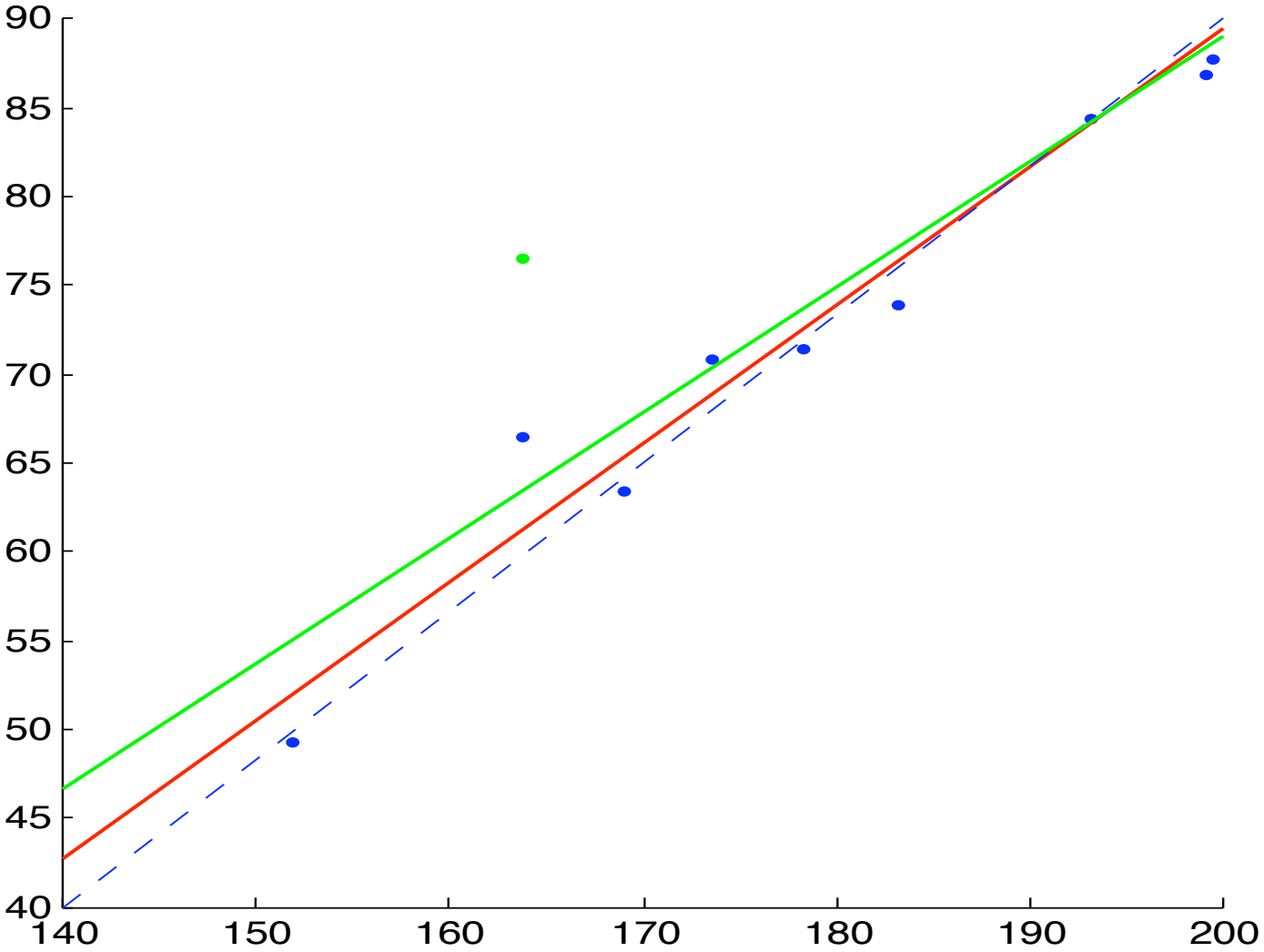
Another important point to note is that the **sum of the residuals** of the least-squares solution **is zero**:

$$\sum_{j=1}^{m}(y_j - X_j\theta) = 0$$



While this property is intuitively appealing, it is worth keeping in mind that it also makes linear regression susceptible to *outliers*: points that are far removed from the regression line, often because of measurement errors.

The effect of outliers



The effect of outliers

Shown on previous slide:

- Suppose that, as the result of a transcription error, one of the weight values from the previous example of univariate regression is increased by 10 kg. The diagram shows that this has a considerable effect on the least-squares regression line.
- Specifically, we see that one of the blue points got moved up 10 units to the green point, changing the red regression line to the green line.

Multiple linear regression

Multiple Regression

Often, we are interesting in modelling the relationship of y to several other variables. In observational studies, the value of y may be affected by the values of several variables.

Example:

Suppose we want to predict people's weight from their height and body frame size (usually measured by wrist size). We collect m height and weight measurements $(h_j, w_j, f_j), 1 \leq j \leq m$

Multiple Regression

Similar to before, the equation we want to solve is:

$$w = \theta_0 + \theta_1 h + \theta_2 f$$

And similar to univariate we have to optimize for squared error:

$$\sum_{j=1}^{m} (w_j - (\ \theta_0 + \theta_1 h_j + \theta_2 f_j))^2$$

- Including more variables can give a narrower confidence interval on the prediction being made
- With many variables, the regression equation and expressions for the θ are expressed better using a matrix representation (as we used before) for sets of equations.

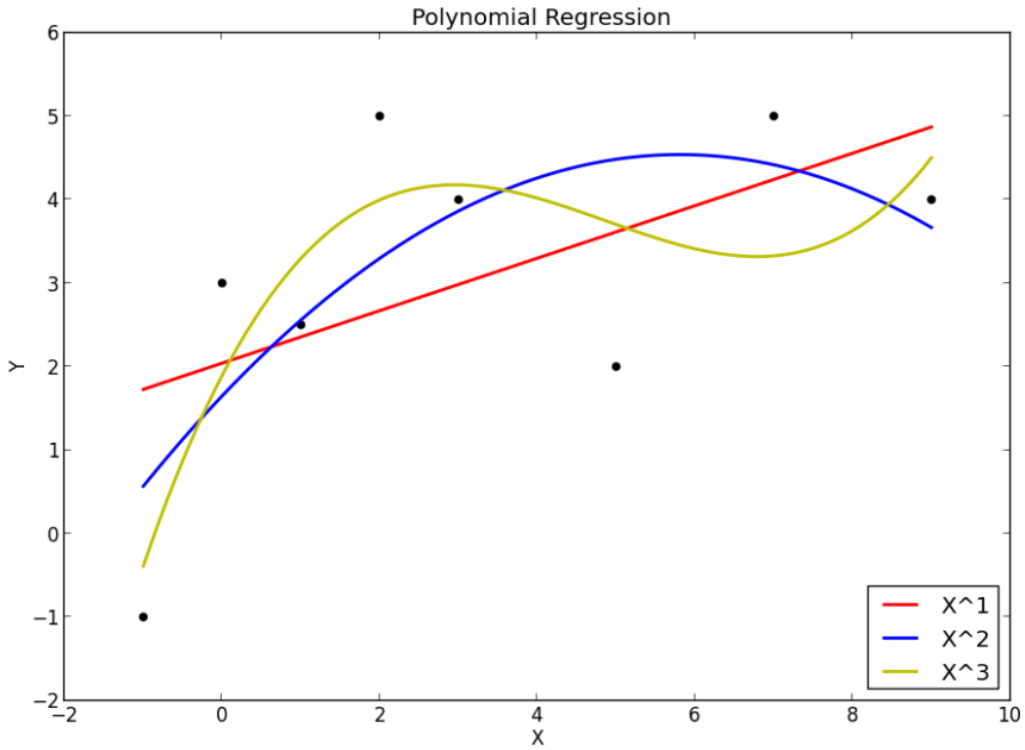
Linear Regression for curve shapes

You may think that linear regression produces straight lines or plains and nonlinear equations models produce curvature!!

Well that's not completely correct. With some tricks we can produce curves with linear regression

Linear Regression for curve shapes

Example



Linear Regression for curve shapes

In this example, we can predict the output with different models:

$$y = \theta_0 + \theta_1 x_1$$

$$y = \theta_0 + \theta_1 x_1 + \theta_2 x_1^2 \ , \ x_2 = x_1^2. \ \rightarrow y = \theta_0 + \theta_1 x_1 + \theta_2 x_2$$

$$y = \theta_0 + \theta_1 x_1 + \theta_2 x_1^2 + \theta_3 x_1^3 \ , \ x_2 = \ x_1^2, x_3 = \ x_1^3 \rightarrow$$
$$y = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3$$

As you can see, these nonlinear models can still be treated like linear regression and they can fit curvatures. They are still linear in parameters. (nonlinear regression is not linear in parameters, e.g. $y = \frac{ax}{b+x}$)

Linear Regression for curve shapes

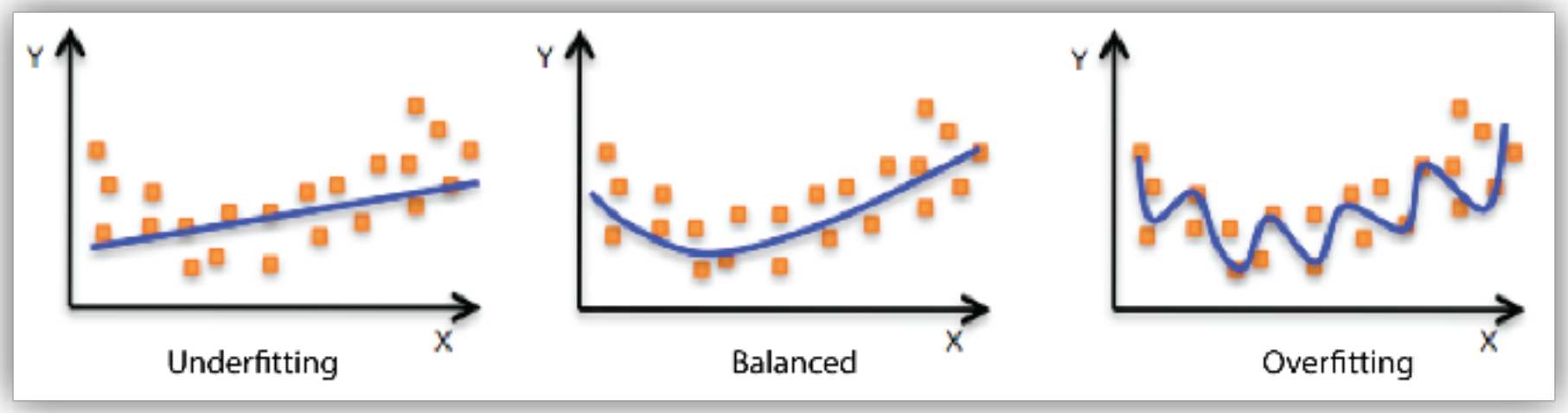


Image from AWS website

Linear Regression for curve shapes

Question:

How to control for degree of complexity of the model to avoid overfitting?

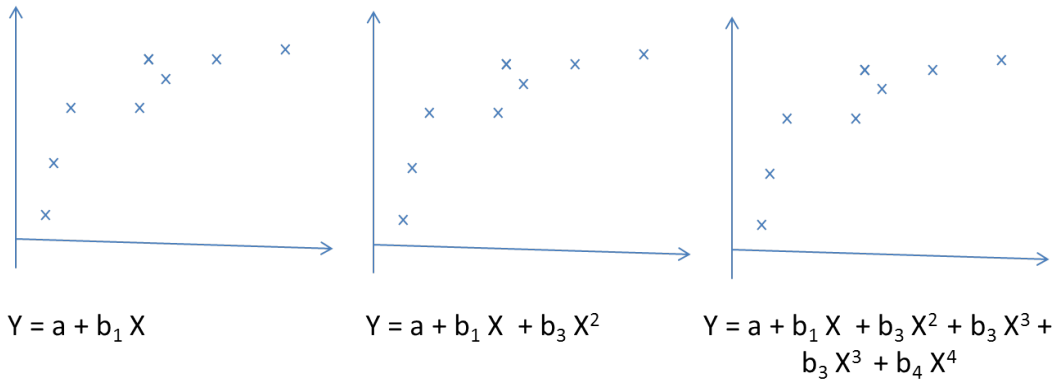
Regularisation

Regularisation

Regularisation is a general method to avoid overfitting by applying additional constraints to the weight vector. A common approach is to make sure the weights are, on average, small in magnitude: this is referred to as *shrinkage*.

Recall the setting for regression in terms of cost minimization.

- Can add penalty terms to a cost function, forcing coefficients to shrink to zero



Regularisation

- MSE as a cost function, given data $(x_1, y_1), \ldots, (x_m, y_m)$

$$J(\theta) = \sum_j (y_j - h_\theta(x_j))^2 + \lambda \sum_i \theta_i^2$$

- Parameter estimation by optimisation will attempt to values for $\theta_0, \ldots, \theta_n$ s.t. $J(\theta)$ is a minimum

- Similar to before, this can be solved by gradient descent or take the derivatives and set them to zero

Regularisation

The multiple least-squares regression problem is an optimisation problem, as we saw, and can be written as:

$$\theta^* = \arg\min_{\theta}(y - X\theta)^T(y - X\theta)$$

The regularised version of this is then as follows:

$$\theta^* = \arg\min_{\theta}((y - X\theta)^T(y - X\theta) + \lambda||\theta||^2$$

Where $||\theta||^2 = \sum_i \theta_i^2$ is the squared norm of the vector θ, or equivalently, the dot product $\theta^T\theta$; λ is a scalar determining the amount of regularisation.

Regularisation

This regularised problem still has a closed-form solution:

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

where I denotes the identity matrix. Regularisation amounts to adding λ to the diagonal of $X^T X$, a well-known trick to improve the numerical stability of matrix inversion. This form of least-squares regression is known as ***ridge regression***.

An interesting alternative form of regularised regression is provided by the ***lasso***, which stands for 'least absolute shrinkage and selection operator'. It replaces the ridge regularisation term $\sum_i \theta_i^2$ with the sum of absolute weights $\sum_i |\theta_i|$. The result is that some weights are shrunk, but others are set to 0, and so the *lasso regression favours sparse solutions*.

Train, Validation & Test Data

- **Train Data:** the data that we use to learn our model and its parameters
- **Validation Data**: The data (unseen by the model) used to provide an unbiased evaluation of a model fit on the training dataset while tuning model hyperparameters.
- **Test Data:** unseen data by the model that we use to test the model and shows how well our model generalizes. In practice, we don't know the ground truth (label) for this data

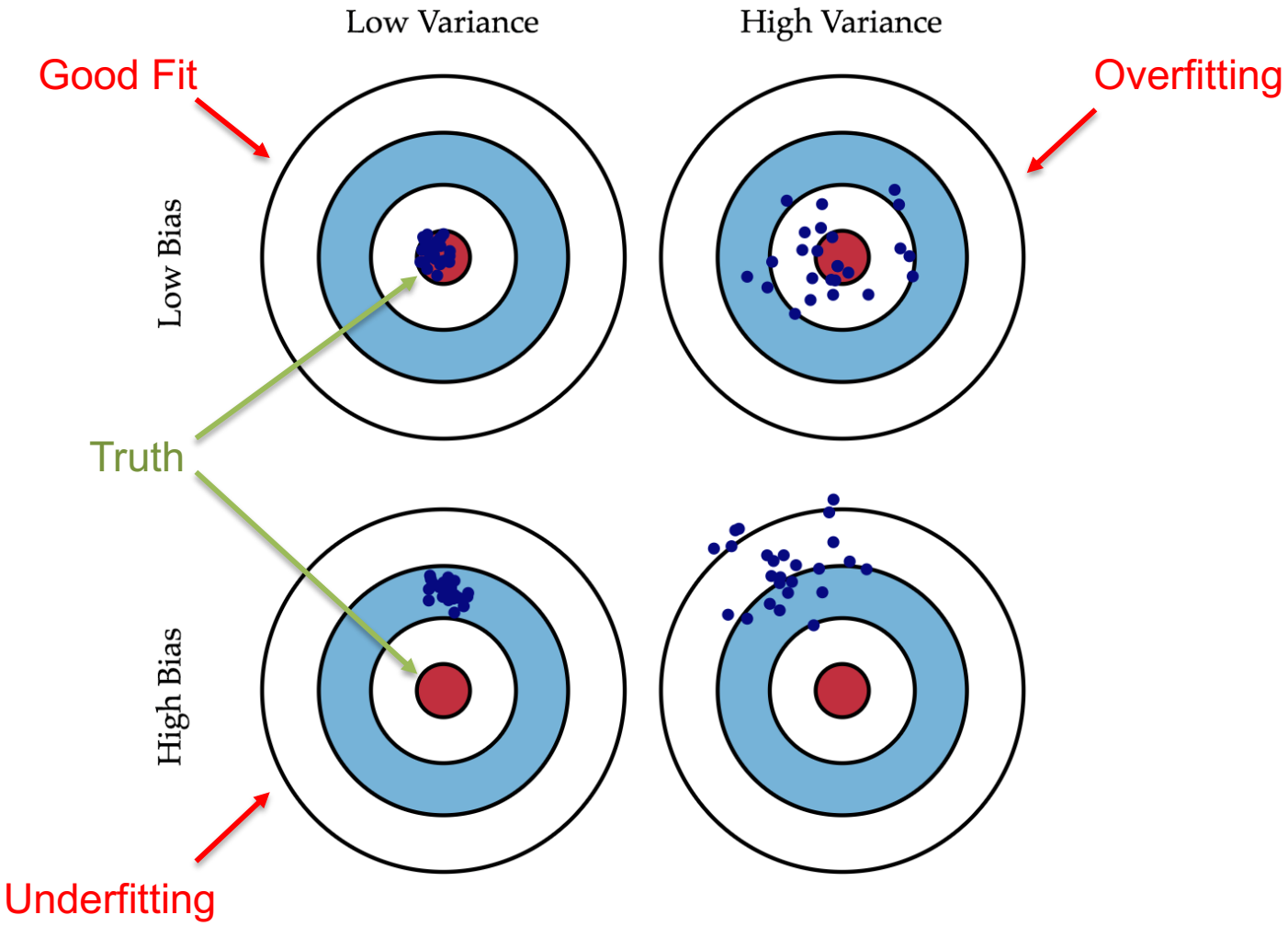
Train, Validation & Test Data

Ideally, we would like, to get the same performance on test set as we get on validation/test set. This is called *Generalization*.

Generalization is the model ability to adapt properly to new, previously unseen data drawn from the same distribution as the one used to create the model.

Bias-Variance Tradeoff

Bias-Variance Tradeoff



Source: Scott-Fortmann,Understanding Bias-variance tradeoff

Bias-Variance Tradeoff

Bias is the difference between the average prediction of our model and the correct value which we are trying to predict. Model with high bias pays very little attention to the training data and oversimplifies the model. It always leads to high error on training and test data.

Variance is the variability of model prediction for a given data point or a value which tells us spread of our data. Model with high variance pays a lot of attention to training data and does not generalize on the data which it hasn't seen before. As a result, such models perform very well on training data but has high error rates on test data.

Bias-Variance Tradeoff

- In supervised learning, **underfitting** happens when a model is unable to capture the underlying pattern of the data. These models usually have high bias and low variance.
- In supervised learning, **overfitting** happens when our model captures the noise along with the underlying pattern in data.

Bias-Variance Decomposition

It can be shown that, when we assume $y = f + \epsilon$ and we estimate f, with $\hat{f}$, then the expectation of error:

$$E[(y - \hat{f})^2] = (f - E[\hat{f}])^2 + Var(\hat{f}) + Var(y)$$

So., the mean of squared error (MSE) can be written as:

$$MSE = Bias^2 + Variance + irreducable\ error$$

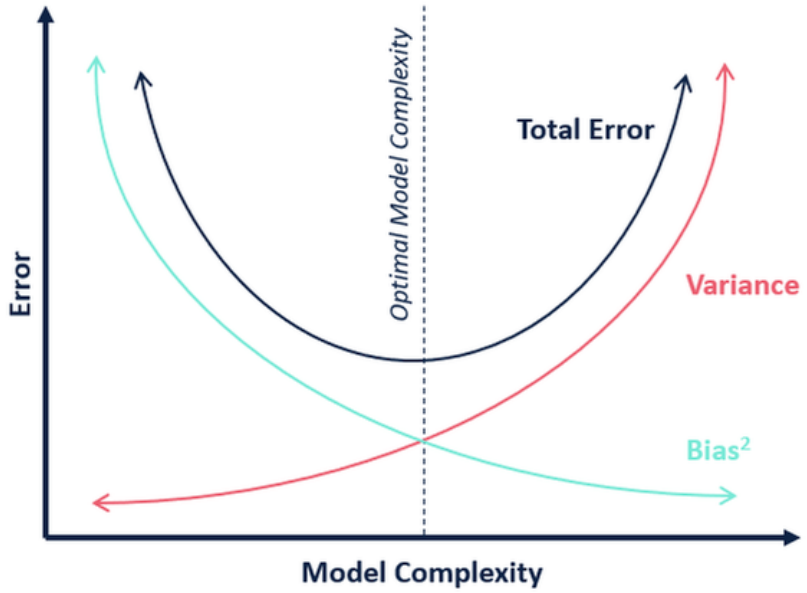
- **Irreducible error** or inherent uncertainty is associated with a natural variability in a system (noise). It can not be not reduced since it is due to unknown/unpredictable factors or simply due to chance.
- **Reducible error**, as the name suggests, can be and should be minimized further by adjustments to the model.

Bias-Variance Tradeoff

- When comparing unbiased estimators, we would like to select the one with minimum variance
- In general, we would be comparing estimators that have some bias and some variance
- If our model is too simple and has very few parameters then it may have high bias and low variance. On the other hand if our model has large number of parameters then it's going to have high variance and low bias. So we need to find the right/good balance without overfitting and underfitting the data.
- We have to choose a complexity that makes a good tradeoff between bias and variance.

Bias-Variance Tradeoff

- Often, the is to find the balance between bias and variance such that we minimize the overall (total) error.
- There is not a quantitative way to find this balanced error point. Instead, you will need to leverage (preferably on an unseen validation data set) and adjust your model's complexity until you find the iteration that minimizes overall error.



Model Evaluation

Model Evaluation

The most popular metrics are:

- Root Mean Square Error (RMSE)

$$RMSE = \sqrt{\frac{1}{m}\sum_{j=1}^{m}(y_j - \hat{y}_j)^2}$$

- Mean Absolute Error (MAE)

$$MAE = \frac{1}{m}\sum_{j=1}^{m}\left|(y_j - \hat{y}_j)\right|$$

- R-squared ($[-\infty, 1]$)

$$R^2 = 1 - \frac{\sum_{j=1}^{m}(y_j - \hat{y}_j)^2}{\sum_{j=1}^{m}(y_j - \bar{y}_j)^2}$$

- Adjuster R-squared

$$R^2_{adjusted} = 1 - [\frac{(1 - R^2)(m-1)}{m-n-1}]$$

Where m is the total number of samples and n is the number of predictors/features.

- R-squared represents the portion of variance in the output that has been explained by the model

Model Evaluation

Example:

Case 1		Case 2			Case 3		
Var1	Y	Var1	Var2	Y	Var1	Var2	Y
x1	y1	x1	2*x1	y1	x1	2*x1+0.1	y1
x2	y2	x2	2*x2	y2	x2	2*x2	y2
x3	y3	x3	2*x3	y3	x3	2*x3 + 0.1	y3
x4	y4	x4	2*x4	y4	x4	2*x4	y4
x5	y5	x5	2*x5	y5	x5	2*x5 + 0.1	y5

	Case 1	Case 2	Case 3
R_squared	0.985	0.985	0.987
Adj_R_squared	0.981	0.971	0.975

- As can be seen adjusted R-squared does a better job at penalizing case 2 & 3 for having more variables without adding any extra information

Model Evaluation

- the absolute value of RMSE does not actually tell how bad a model is. It can only be used to compare across two models
- Adjusted R-squared easily tells about the quality of the model as well. For example, if a model has adjusted R-squared equal to 0.05 then it is definitely poor.
- However, if you care only about prediction accuracy then RMSE is best. It is computationally simple, easily differentiable and present as default metric for most of the models.

Some further issues in learning linear regression models

Categoric Variables

- "Indicator" variables are those that take on the values 0 or 1
- They are used to include the effects of categoric variables. For example, if D is a variable that takes the value 1 if a patient takes a drug and 0 if the patient does not. Suppose you want to know the effect of drug D on blood pressure y keeping age (X) constant

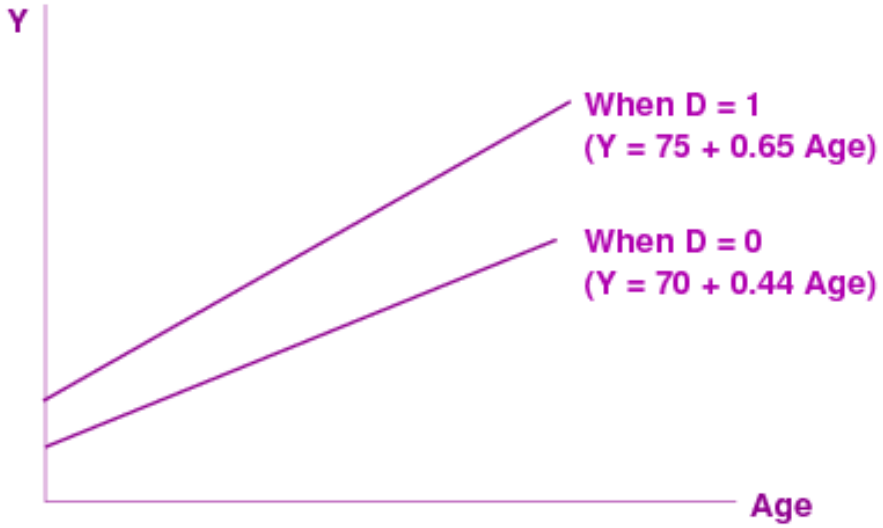
$$y = 70 + 5D + 0.44X$$

- So, taking the drug (a unit change in D) makes a difference of 5 units, provided age is held constant

Categoric Variables

- How do we capture any interaction effect between age and drug intake? Introduce a new indicator variable $DX = D \times X$

$$y = 70 + 5D + 0.44X + 0.21DX$$



Model Selection

Suppose there are a lot of variables/features (x), some of which may be representing products, powers, etc.

- Taking all the features will lead to an overly complex model. There are 3 ways to reduce complexity:
  1. Subset-selection, by search over subset lattice. Each subset results in a new model, and the problem is one of model-selection
  2. Shrinkage, or regularization of coefficients to zero, by optimization. There is a single model, and unimportant variables have near-zero coefficients.
  3. Dimensionality-reduction, by projecting points into a lower dimensional space (this is different to subset-selection, and we will look at it later)

Model Selection

Subset-selection: we want to find a subset of variables/features which performs well and get rid of redundant features. This is also called stepwise regression.

- Historically, model-selection for regression has been done using “forward-selection”, “backward-elimination”, or “bidirectional” methods
  - These are greedy search techniques that either:
    - (a) start with no variable and at each step add one variable whose addition gives the most improvement to the fit
    - (b) start with all variables and at each step remove one whose loss gives the most insignificant deterioration of the model fit;
    - (c) a combination of the above, testing at each step for variables to be included or excluded.

Model Selection

- This greedy selection done on the basis of calculating the fit quality (often using R-squared, which denotes the proportion of total variation in the dependent variable Y that is explained by the model)
- Given a model formed with a subset of variables X, it is possible to compute the observed change in R-squared due to the addition or deletion of some variable x
- This is used to select greedily the next best move in the graph-search

To set other hyper-parameters, such as shrinkage parameter in regularisation, we can use grid search

Local (nearest-neighbor) regression

Local Learning

- Related to the simplest form of learning: rote learning, or memorization
- Training instances are searched for instance that most closely resembles query or test instance
- The instances themselves represent the knowledge
- Called: nearest-neighbor, instance-based, memory-based or case-based learning; all forms of local learning
- The similarity or distance function defines “learning”, i.e., how to go beyond simple memorization
- Intuition — classify an instance similarly to examples “close by” — neighbors or exemplars
- A form of lazy learning – don’t need to build a model!

Nearest neighbor for numeric prediction

- Store all training examples $(f(X_i), X_i)$

Nearest neighbor:

- Given query instance X_q,
- first locate nearest training example x_n,
- then estimate $\hat{y}_q = f(X_n)$

k-Nearest neighbor:

- Given X_q, take mean of f values of k nearest neighbors

$$\hat{y}_q = \frac{\sum_{i=1}^{k} f(X_i)}{k}$$

Distance function

The distance function defines what is “learned”, i.e., predicted.

- Most commonly used distance function is Euclidean distance, where
- If instance X_i is defined with n feature values:

$$\left\langle x_1^{(i)}, \ldots, x_n^{(i)} \right\rangle$$

- the distance between two instances X_i , X_j is defined as

$$d(X_i, X_j) = \sqrt{\sum_{k=1}^{n} (x_k^{(i)} - x_k^{(j)})^2}$$

Local regression

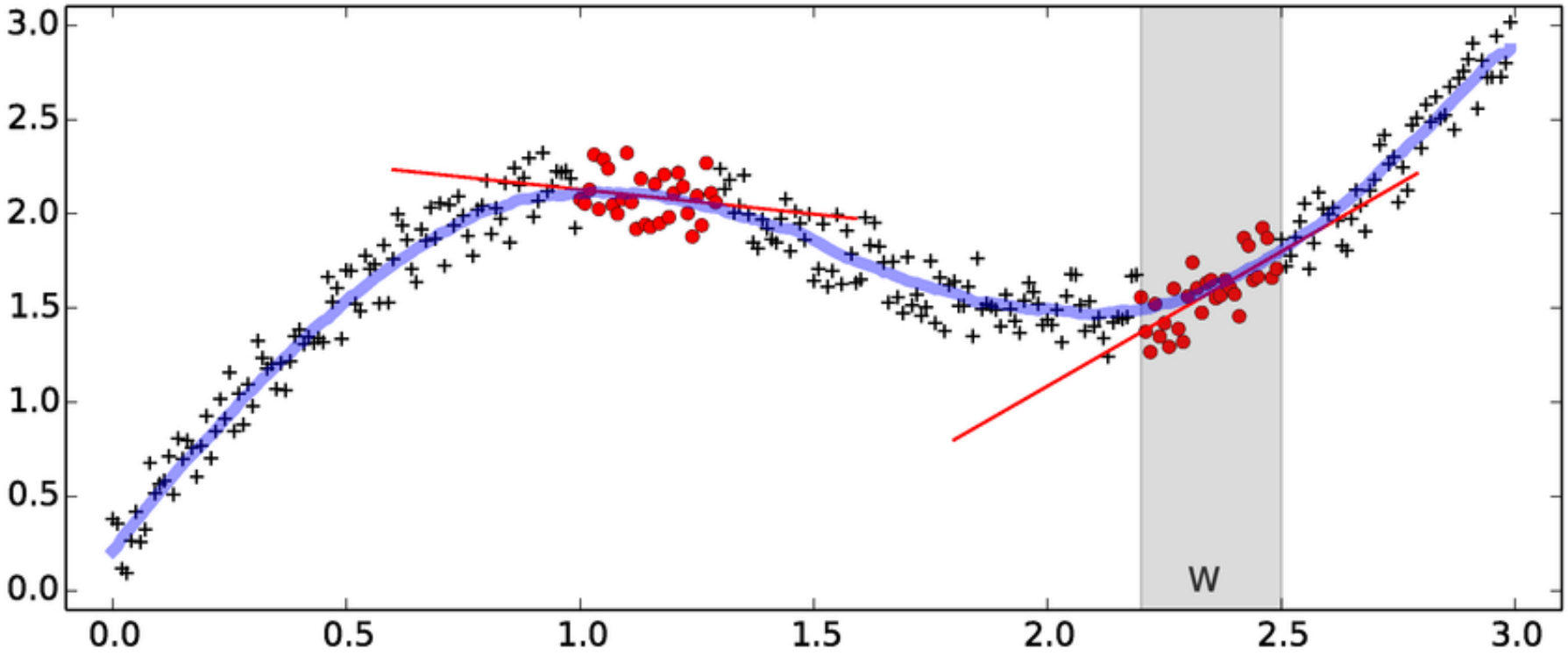
Use kNN to form a local approximation to f for each query point X_q using a linear function of the form:

$$\hat{y} = \hat{f}(X) = \theta_0 + \theta_1 x_1 + \cdots + + \theta_m x_m$$

Where does this linear regression model come from ?

- fit linear function to k nearest neighbors
- quadratic or higher-order polynomial . . .
- produces "piecewise approximation" to f

Local regression



[From, Locally-weighted homographies for calibration of imaging systems, by P. Ranganathan & E. Olson]

Acknowledgements

- Material derived from slides for the book "Elements of Statistical Learning (2nd Ed.)" by T. Hastie, R. Tibshirani & J. Friedman. Springer (2009) http://statweb.stanford.edu/~tibs/ElemStatLearn/
- Material derived from slides for the book "Machine Learning: A Probabilistic Perspective" by P. Murphy MIT Press (2012) http://www.cs.ubc.ca/~murphyk/MLbook
- Material derived from slides for the book "Machine Learning" by P. Flach Cambridge University Press (2012) http://cs.bris.ac.uk/~flach/mlbook
- Material derived from slides for the book "Bayesian Reasoning and Machine Learning" by D. Barber Cambridge University Press (2012) http://www.cs.ucl.ac.uk/staff/d.barber/brml
- Material derived from slides for the book "Machine Learning" by T. Mitchell McGraw-Hill (1997) http://www- 2.cs.cmu.edu/~tom/mlbook.html
- Material derived from slides for the course "Machine Learning" by A. Ng, Stanford University, USA (2015)
- Material derived from slides for the course "Machine Learning" by A. Srinivasan BITS Pilani, Goa, India (2016)