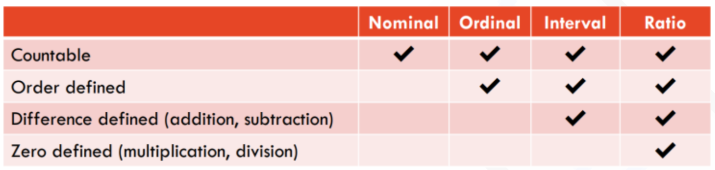
L2-L3: **Nominal Data** : Values are names; No ordering is implied; Eg jersey numbers; industry worked in; key experience you have

Ordinal Data: Values are ordered No distance is implied – Eg rank, agreement – central tendency can be measured by mode or median – **the mean cannot be defined from an ordinal set** – dispersion can be estimated by the Inter-Quartile Range (IQR) *The IQR is the difference between the first and third quartile*

Interval Data Interval scales provide information about order, and also possess equal intervals – Values encode differences – equal intervals between values – No true zero – Addition is defined – Eg **Celsius temperature** central tendency can be measured by mode, median, or mean

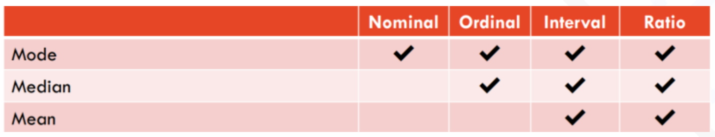
Ratio Data – Values encode differences – Zero is defined – Multiplication defined – Ratio is meaningful – Eg length, weight, income

Level of measurement



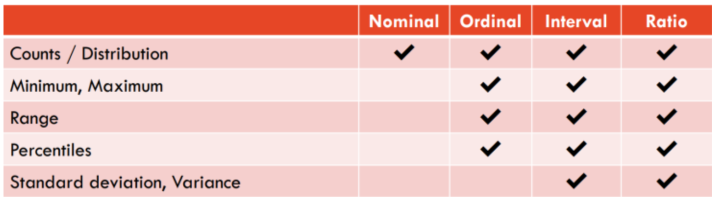
	Nominal	Ordinal	Interval	Ratio
Countable	✓	✓	✓	✓
Order defined		✓	✓	✓
Difference defined (addition, subtraction)			✓	✓
Zero defined (multiplication, division)				✓

Measure of central tendency



	Nominal	Ordinal	Interval	Ratio
Mode	✓	✓	✓	✓
Median		✓	✓	✓
Mean			✓	✓

Measure of Dispersion



	Nominal	Ordinal	Interval	Ratio
Counts / Distribution	✓	✓	✓	✓
Minimum, Maximum		✓	✓	✓
Range		✓	✓	✓
Percentiles		✓	✓	✓
Standard deviation, Variance			✓	✓

First sort values, then:

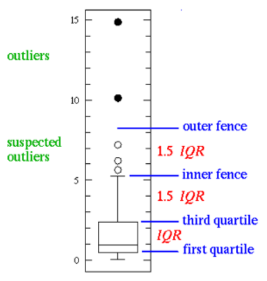
- ***Median*** is the middle value (or average of two middle values)
- ***Minimum*** is the first value
- ***Maximum*** is the last value
- ***10th percentile*** is item at index 0.1*N
- ***90th percentile*** is item at index 0.9*N
- ***Range*** is Maximum minus Minimum

Box plots summarise data based on 5 numbers:

- Lower inner fence – Q1–1.5*IQR
- First quartile (Q1) – equivalent to 25th percentile
- Median (Q2) – equivalent to 50th percentile
- Third quartile (Q3) – equivalent to 75th percentile
- Upper inner fence – Q3+1.5*IQR

Values outside fences are outliers

Sometimes include outer fences at 3*IQR



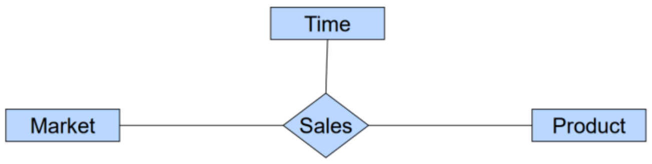
Relational data model is the most widely used model today – Main concept: relation, basically a table with rows and columns – Every relation has a schema, which describes the columns, or fields

Not all tables qualify as a relation:

– Every relation must have a **unique** name.
– **Attributes (columns)** in tables must have **unique names**.
=> The order of the columns is irrelevant.
– All tuples in a relation have the same structure; constructed from the same set of attributes
– Every attribute value is atomic (not multivalued, not composite).
– **Every row is unique** (can't have two rows with exactly the same values for all their fields)
– The order of the rows is immaterial

ETL Process: Capture/Extract - Data Cleansing - Transform – Load

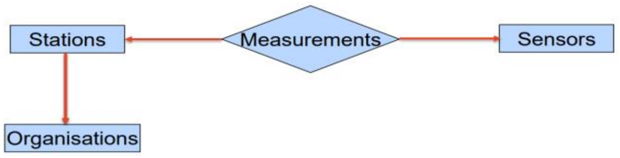
The fact and dimension relations linked to it looks like a star; – this is called a star schema



If we map this to relations

- 1 central fact table
- *n* dimension tables with foreign key relationships from the fact table *(the fact table holds the FKs referencing the dimension tables)*

Snowflake schema: A refinement of star schema where some dimensional hierarchy is **normalized** into a set of smaller dimension tables, forming a shape similar to snowflake measurements are the facts, rest describes the dimensions



Fact constellations: Multiple fact tables share dimension tables, viewed as a collection of stars, therefore called **galaxy schema** or **fact constellation.**

DDL (Data Definition Language)

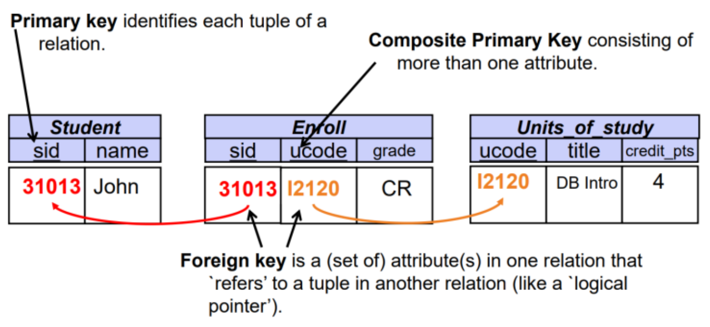
CREATE TABLE name (list_of_columns)

DML (Data Manipulation Language) for retrieval of information also called **query language** SELECT … FROM … WHERE

Primary key: unique, minimal identifier of a relation.
- Examples include employee numbers, social security numbers, etc. This is how we can guarantee that all rows are unique.

Foreign keys are identifiers that enable a dependent relation (on the many side of a relationship) to refer to its parent relation (on the one side of the relationship)
- Must refer to a candidate key of the parent relation
- Like a `logical pointer'

Keys can be **simple** (single attribute) or **composite** (multiple attributes)


Primary key identifies each tuple of a relation.

Composite Primary Key consisting of more than one attribute.

Foreign key is a (set of) attribute(s) in one relation that `refers' to a tuple in another relation (like a `logical pointer').

– **SELECT**	Lists the attributes (and expressions) that should be returned from the query
– **FROM**	Indicate the table(s) from which data will be obtained
– **WHERE**	Indicate the conditions to include a tuple in the result
– **GROUP BY**	Indicate the categorization of tuples
– **HAVING**	Indicate the conditions to include a category
– **ORDER BY**	Sorts the result according to specified criteria

SQL Statement	Meaning
SELECT COUNT(*) FROM *T*	count how many tuples are stored in table *T*
SELECT * FROM *T*	list the content of table *T*
SELECT * FROM *T* LIMIT *n*	only list *n* tuples from a table
SELECT * FROM *T* ORDER BY *a*	order the result by attribute *a* (in ascending order; add DESC for descending order)

SQL Aggregate Function	Meaning
COUNT(*attr*) ; COUNT(*)	Number of *Not-null-attr* ; or of all values
MIN(*attr*)	Minimum value of *attr*
MAX(*attr*)	Maximum value of *attr*
AVG(*attr*)	Average value of *attr* (arithmetic mean)
MODE() WITHIN GROUP (ORDER BY *attr*)	mode function over *attr*
PERCENTILE_DISC(0.5) WITHIN GROUP (ORDER BY *attr*)	median of the *attr* values

How many measurements we have done?
SELECT COUNT(*) FROM Measurement
List top five measurements ordered by date in descending order
SELECT * FROM Measurement ORDER BY date DESC limit 5;

e.g1: SELECT * FROM TelescopeConfig
WHERE (mindec BETWEEN -90 AND -50) AND (maxdec >= -45) AND (tele_array = 'H168')

e.g2 SELECT * FROM TelescopeConfig
WHERE tele_array LIKE 'H%';

EXTRACT(year FROM startDate)

TO_DATE('01-03-2012', 'DD-Mon-YYYY')

'2012-04-01' + INTERVAL '36 HOUR'

SELECT gid, band, epoch FROM Measurement WHERE intensity IS NULL
5 + null returns null

SELECT sitename, commence, organisation
FROM Station **JOIN** Organisation
ON orgcode = code; (inner join)

SELECT **uos_code** as unit_of_study, AVG(mark)
F

ROM Assessment NATURAL JOIN UnitOfStudy
WHERE credit_points = 6
GROUP BY **uos_code**
HAVING COUNT(*) > 2

```
SELECT COUNT(value),
       MIN(value),
       Max(value),
       MAX(value) - MIN(value)                              AS Range,
       AVG(value)                                              AS Mean,
       MODE()  WITHIN GROUP (ORDER BY value)                   AS Mode,
       PERCENTILE_DISC(0.5) WITHIN GROUP (ORDER BY value)      AS Median,
       PERCENTILE_DISC(0.25) WITHIN GROUP (ORDER BY value)     AS Percentile25,
       PERCENTILE_DISC(0.75) WITHIN GROUP (ORDER BY value)     AS Percentile75,
       PERCENTILE_DISC(0.75) WITHIN GROUP (ORDER BY value)
       - PERCENTILE_DISC(0.25) WITHIN GROUP (ORDER BY value) AS IQR
  FROM Measurement WHERE sensor='temp';
```

In which **time period** were all the measurement done?
SELECT MIN(date), MAX(date) FROM Measurement;

How many distinct Stations the temperature were measured
SELECT COUNT(DISTINCT station)
FROM Measurement WHERE sensor = 'temp';

How many measurements of *distinct* stations were done *per each sensor*?
SELECT sensor, COUNT(DISTINCT station)
FROM Measurement
GROUP BY sensor
ORDER BY count DESC;

self join - lists all film sub-categories and their corresponding parent categories

```
+------------+-------------+-----------+
| category_id |    name     | parent_cat |
+------------+-------------+-----------+
|          0 | Fiction     |           |
|          1 | Non-Fiction |           |
|          2 | Action      |         0 |
|         20 | Disaster    |         2 |
|         21 | Thriller    |         2 |
```

```
select cate.category_id,cate.name as category,pa.name as parent
from category cate join category pa
on pa.category_id = cate.parent_cat
where cate.parent_cat is not null
```

Determines the usage of Film categories throughout our database

```
CREATE VIEW LengthyDramas AS
  Select film_id as id,title,release_year as year,length as minutes
  From Film
  where lower(description) LIKE '%drama%' and length > 90
  order by minutes desc,title;
```

1. `category_id`;
2. `name` of the `Category`;
3. `count` of films that belong to this `Category` (or zero if there are none).

```
select c.category_id,c.name, count(fc.film_id) as count
from Category c LEFT OUTER JOIN Film_Category fc
on c.category_id = fc.category_id
group by c.category_id,c.name
order by count desc, c.name
```

lists all Actor nationalities and how many actors are of each nationality. Only show nationalities with at least 2 associated actors.

```
select nationality,count(actor_id) as number_of_actors
from Actor
group by nationality
having count(actor_id)>=2
order by count(actor_id) desc,nationality
```

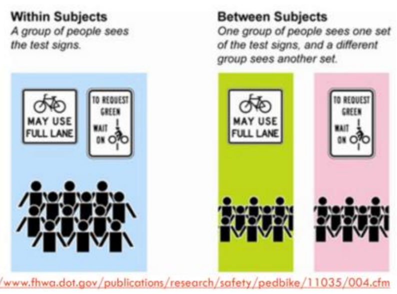
lists every Film which has at least five actors playing in it.

```
select fa.film_id,f.title,count(actor_id)
from Film_Actor fa inner join Film f on f.film_id = fa.film_id
group by fa.film_id,f.title
having count(actor_id)>=5
order by f.title
```

Hypothesis Testing

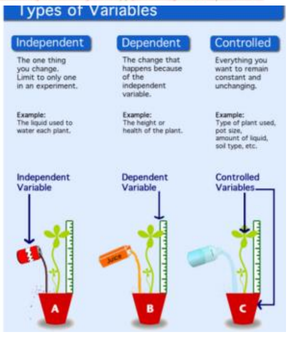
Experiment design

- ***Between subjects***: Each subject sees one and only one condition
- ***Within subjects***: Subjects see more than one or all conditions


Within Subjects
A group of people sees the test signs.
Between Subjects
One group of people sees one set of the test signs, and a different group sees another set.
MAY USE FULL LANE
TO REQUEST GREEN WAIT ON
MAY USE FULL LANE
TO REQUEST GREEN WAIT ON
http://www.fhwa.dot.gov/publications/research/safety/pedbike/11035/004.cfm

Types of variables

- ***Dependent variable*** is the measure of interest
- ***Independent variable*** is manipulated to observe the effect on dependent variable
- ***Controlled variables*** are materials, measurements and methods that don't change


Types of Variables
Independent
The one thing you change. Limit to only one in an experiment.
Example: The liquid used to water each plant.
Dependent
The change that happens because of the independent variable.
Example: The height or health of the plant.
Controlled
Everything you want to remain constant and unchanging.
Example: Type of plant used, pot size, amount of liquid, soil type, etc.
Independent Variable
Dependent Variable
Controlled Variables
Juice
A
B
C

- Research question (Q):
  - Asks whether the independent variable has an effect
  - "If there is a change in the independent variable, will there also be a change in the dependent variable?"
- Null hypothesis (H_0):
  - The assumption that there is no effect
  - "There is no change in the dependent variable when the independent variable changes."

Testing reliability with p-values

- Most tests calculate a p-value for measuring observation extremity
- Compare to significance level threshold α
  - α is the probability of (wrongly) rejecting H_0 given that it is true
  - aka Type I error rate (false positive)
  - Commonly use α of 5% or 1%

Truth	Decision: Accept H_0	Decision: Reject H_0
H_0 (No difference)	Right Decision	Type I error
H_1 (Difference exists)	Type II error	Right Decision

P-value	Indicates	Reject H_0?
<α	Strong evidence against the null hypothesis	Yes
>α	Weak evidence against the null hypothesis	No
=α	Marginal	NA

Increase the power of a significance test
– Obtain a larger sample
– Larger N means more reliable statistics
– Less likely to have errors
– Type I: Reject true H0
– Type II: Fail to reject false H0
Unpaired or independent : separate individuals
Paired: same individual at different points in time.

Unpaired Student's t-test
null hypothesis that two **population** means are equal
Assumes
– The samples are **independent**
– Populations are **normally distributed**
– **Standard deviations** are equal
– Note – Multiply **two-tailed p-value by 0.5** for one-tailed p-value (e.g., to test A>B, rather than A>B OR A<B)
scipy.stats.**ttest_ind**(a, b, axis=0, equal_var=True, nan_policy='propagate', alternative='two-sided'

Mann-Whitney U test
Nonparametric version of unpaired t-test
Assumes
– The samples are **independent**
– Note – N should be **at least 20**
scipy.stats.**mannwhitneyu**(x, y, use_continuity=True, alternative=None

Analysis of variance (ANOVA)
null hypothesis **two or more groups** have the same population mean
Assumes:
– The samples are independent
– Populations are normally distributed
– Standard deviations are equal
scipy.stats.**f_oneway**(*args, axis=0

Kruskall-Wallis H-test

Nonparametric version of ANOVA
– Assumes samples are independent
– also one-way ANOVA on ranks – as the ranks of the data values are used in the test rather than the actual data
Note:
– Not recommended for samples smaller than 5
– Not as statistically powerful as ANOVA
– Both ANOVA and Kruskall-Wallis H-test are extension of the Mann-Whitney test and Unpaired Student's t-test used to **compare the means of more than two populations.**
scipy.stats.**kruskal**(*args, nan_policy='propagate

Paired Student's t-test
null hypothesis that two population means are equal
Assumes
– The samples are **paired**
– Populations are normally distributed
– Standard deviations are equal
Multiply two-tailed p-value by 0.5 for one-tailed p-value (to test A>B, rather than A>B OR A<B)
*scipy.stats.**ttest_rel**(a, b, axis=0, nan_policy='propagate', alternative='two-sided*
Wilcoxon signed-rank test
Nonparametric version of paired t-test
– Assumes
– The samples are paired
– Note – Often used for ordinal data, e.g., Likert ratings
– N should be large, e.g., ≥20
*scipy.stats.**wilcoxon**(x, y=None, zero_method='wilcox', correction=False, alternative='two-sided', mode='auto'*

Measurement: Accuracy, precision, recall, f1

	s=1	s=0
g=1	TP (true positives)	FN (false negatives)
g=0	FP (false positives)	TN (true negatives)

– Accuracy: (TP+TN) / N
% correct over all instances
– Precision: TP / (TP+FP)
% correct system predictions
– Recall: TP / (TP+FN)
% correct gold labels
– F1: 2PR / (P+R)
Harmonic mean of Precision and Recall

Holdout method – Splits the data randomly into two independent sets
• Training set (e.g., 2/3) for model construction
• Test set (e.g., 1/3) for accuracy estimation
– Random sampling: a variation of holdout
• Repeat holdout k times, accuracy = avg. of the accuracies obtained

Cross-validation (k-fold, where k = 10 is most popular)
– Randomly partition the data into k **mutually exclusive** subsets, each approximately equal size
– Leave-One-Out is a particular form of cross-validation:
• k folds where k = # of tuples, for small sized data

Tutorial 6: Compute significance for H1 sys1 > sys2

Determine which classifier is better. (paired t-test)
stats.**ttest_rel**(sys1_scores, sys2_scores).pvalue*0.5

- Would you expect this variation in a real experiment?

Average scores should only change if the sample is not fixed, or **if folds are sampled randomly**.

- What does this variation say about reliability of experiments?

The variation highlights the fact that we always need to be careful generalising results to unseen data.
# It also highlights the importance of **selecting samples** that are **representative** of the population.

- How can we increase reliability?

Significance testing helps us quantify reliability. **Larger sample sizes** help ensure reliability.

Linear Regression

Fitting SLR: learn α and β

$$Y_i = \alpha + \beta X_i + \varepsilon_i$$

– **α**: Intercept (where the line crosses the y axis)
– **β**: Slope (direction and steepness of the line)
– **ε_i**: Error (error term describing variation of data)
– **Y_i** is the dependent variable (response).
– **X_i** is the independent variable (predictor).

– Sum of squared errors:

$$\varepsilon = SSE = \underbrace{\sum}_{\text{Sum}} \underbrace{(y_i - \hat{y}_i)}_{\text{Error}} \underbrace{{}^2}_{\text{Square}}$$

Error/residual: difference between the observed value and predicted value ($yactual - ypredicted$)

– Our goal is to find the optimal value of $\hat{\alpha}$ and $\hat{\beta}$ such that

$$\frac{1}{n}\sum_{i=1}^{n}\left((\hat{\alpha} + \hat{\beta}\,x_i) - y_i\right)^2 \text{ is the minimum}$$

– Using calculus

$$\hat{\beta} = \frac{\sum(x_i - \bar{x})(y_i - \bar{y})}{(x_i - \bar{x})^2} = \frac{\operatorname{cov}(x,y)}{Var(x)} = \frac{r(x,y) * sd(y)}{sd(x)} \qquad \hat{\alpha} = \bar{y} - \hat{\beta}\bar{x}$$

Where ***sd*** is the standard deviation and ***r*** is the correlation

R²: ratio of **explained variation in y** to **total variation in y**

$$R^2 = 1 - \frac{\sum(y_i - \hat{y}_i)^2}{\sum(y_i - \bar{y})^2} = 1 - \frac{SSE}{SST}$$

Range from 0 to 1
goodness of fit not precision

$$S = \sqrt{\frac{SSE}{N}}$$

Standard error (S):prediction accuracy
Expressed in units of the response variable
Prediction interval: range that should contain the response value of a new observation

If sample size is large enough useful rule-of-thumb: approximately 95% of predictions should fall within $\hat{y}_i \pm 2 * S$

Suppose: S = $2k and requirement is predictions within $5k – S must be <= $2.5k to produce a **sufficiently narrow 95% prediction interval**

Multiple linear regression P482 (L9 – p25)

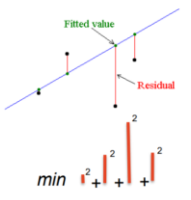
How to learn θ

$$Y = h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \ldots + \theta_d x_d = \sum_{j=0}^{d} \theta_j x_j = \theta^T x$$

Assume $x_0 = 1$

- Cost function: $J(\theta) = \frac{1}{2n}\sum_{i=1}^{n}(h_\theta(x^{(i)}) - y^{(i)})^2$
- Fit model by solving $\min_\theta J(\theta)$
- Basic search procedure
  - Choose initial value for θ
  - Until we reach a minimum:
    - Choose a new value for θ to reduce $J(\theta)$

predicted y



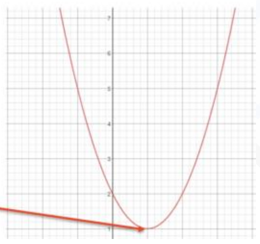
Gradient Descent

Gradient descent is a first-order iterative optimization algorithm for **finding a local minimum of a differentiable function**. The idea is to **take repeated steps** in the opposite direction of the gradient (or approximate gradient) of the function at the current point, because this is the direction of steepest descent.

Example:

- Given $y(x) = x^2 - 2x + 2$, find the value of X to minimise $y(x)$
- From Calculus, by finding the derivative and set it equal to zero:

$\frac{dy(x)}{dx} = 2x - 2 = 0 => x = 1$



With gradient descent, we don't know the optimal value of x,

So we pick a random number

- Let $x = 3$, which obviously is wrong
- Step 1 : we take the derivative of the function
  - $\frac{dy(x)}{dx} = 2x - 2$
- Step 2: we study the derivative at the point we guessed ($x = 3$)
  - $\frac{dy(x)}{dx} = 2 * 3 - 2 = 4$, but the derivative at min should be zero

Given that the derivative is positive, we know that the value is getting larger

Therefore we need to go backword

If we have guessed $x = -1$ instead, the derivative would have been -4

- then we would know that the function is getting smaller

By studying the derivative of the current guess, we know if we are getting closer or further away from the minimum

So here is the equation

- $x_{i+1} = x_i - \alpha * \frac{dy(x)}{dx}$ # α = learning rate, e.g. $\alpha = 0.2$

Given our example, we guessed $x_0 = 3$

- $x_{i+1} = x_i - \boxed{0.2} * \frac{dy(x)}{dx} \rightarrow 2x - 2$
- $x_1 = 3 - 0.2 * 4 = 2.2$

We repeat this process again at $x_1 = 2.2$

- $\frac{dy(x)}{dx} = 2x - 2$
- $\frac{dy(x=2.2)}{dx} = 2 * 2.2 - 2 = 2.4$
- $x_2 = 2.2 - 0.2 * 2.4 = 1.72$, we moved closer

At at $x_2 = 1.72$

- $\frac{dy(x)}{dx} = 2x - 2$
- $\frac{dy(x=1.72)}{dx} = 2 * 1.72 - 2 = 1.44$
- $x_3 = 1.72 - 0.2 * 1.44 = 1.432$, we moved closer

If we keep repeating this process, we can find the minimum point of the solution.

If α is small, gradient descent can be slow

If α is too large, gradient descent might **overshoot** the minimum

Batch descent

Slow but more accurate, costly

stochastic gradient descent

fast, may not converged to the min. training set is large.

Make sure features are on a similar scale

Logistic Regression

Applying linear regression for classification is often not useful

$h_\theta(x)$ can be a large positive or negative value while y is Yes or No (0 or 1) in case of binary classification

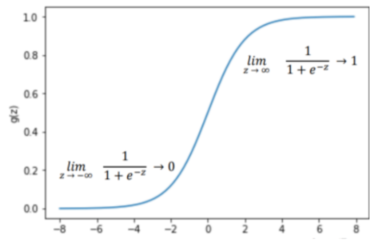
Logistic or sigmoid function

$0 \le h_\theta(x) \le 1$

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

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

$g(z)' = g(z)(1 - g(z))$



A threshold is defined to classify

- If y >= 0.5, predict y = 1
- If y <0.5 , predict y = 0

We can see:

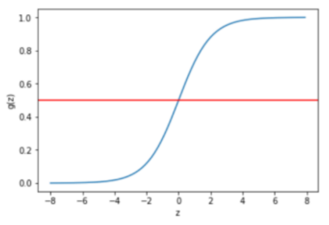
$g(z) \ge 0.5$ if $Z \ge 0$

$g(z) \le 0.5$ if $Z < 0$

Assume

$P(y = 1 | x; \theta) = h\theta(x)$

$P(y = 0 | x; \theta) = 1 - h\theta(x)$



It can be written as :

$P(y | x; \theta) = (h\theta(x))^y (1 - h\theta(x))^{1-y}$

Convex cost function - Cross – Entropy (Log Loss) P509 P9-50

Tutorial:

If R = 0.39: The value here is 0.329. This suggests that our model only partly explains the data so there **must be other factors at play**.

If R=0.755: Yes, r_squared indicates that our model explains the data reasonable well. But we should look at **standard error** as well

According to the 95% prediction interval, how close will our predictions be to the actual value? What if we calculate over the test data instead?
Answer: Note we have a **fairly small data set** (339 in training, 167 in test). So this value will vary depending on our split.

Unstructured Data– Naïve Bayes

Tokenisation

– Split a string (document) into pieces called tokens
– Possibly remove some characters, e.g., punctuation

Normalisation

Map similar words to the same token
– **Stemming/lemmatisation**
  – Avoid grammatical and derivational sparseness
  – E.g., "was" => "be"
– **Lower casing, encoding**
  – E.g., "Naïve" => "naive"

Term frequency weighting

["friend",
"roman",
"roman",
"countrymen"]

↓

{"friend": 1,
"roman": 2,
"countryman": 1}

– Term frequency
  – Give more weight to terms that are common in document
  – *TF = |occurrences of term in doc|*
– Damping
  – Sometimes want to reduce impact of high counts
  – *TF = log(|occurrences of term in doc|)*

TFIDF Weighting

["friend",
"roman",
"countrymen"]

↓

{"friend": 0.1,
"roman": 0.8,
"countrymen": 0.2}

– Inverse document frequency
  – Give less weight to terms that are common across documents
  – *IDF = log(|docs|/|docs containing term|)*
– TFIDF
  – *TFIDF = TF * IDF*

Naïve Bayes

$$P(H|\mathbf{X}) = \frac{P(\mathbf{X}|H)P(H)}{P(\mathbf{X})} = P(\mathbf{X}|H) \times P(H) / P(\mathbf{X})$$

Level	Lang	Tweets	PhD	Interviewed well
Senior	Java	No	No	False
Senior	Java	No	Yes	False
Mid	Java	No	No	True
Junior	Python	No	No	True
Junior	R	Yes	No	True
Junior	R	Yes	Yes	False
Mid	R	Yes	Yes	True
Senior	Python	No	No	False
Senior	R	Yes	No	True
Junior	Python	Yes	No	True
Senior	Python	Yes	Yes	True
Mid	Python	No	Yes	True
Mid	Java	Yes	No	True
Junior	Python	No	Yes	False

Class:
C1: Interviewed well = 'False'
C2: Interviewed well = 'True'

Data to be classified:
X = (Level =Senior,
Lang = Python,
Tweets = yes
PhD = No)

Calculation

X = (Level =Senior, Lang = Python, Tweets = yes, PhD = No)
We need to compute **$P(C_i | X) = P(X | C_i) * P(C_i)$**

$P(C_{True})$: P(Interviewed well = "True") = 9/14 = 0.643

$P(C_{Fasle})$: P(Interviewed well = "False") = 5/14= 0.357

Compute $P(X | C_{True})$
P(Level = "Senior" | Interviewed well = "True") = 2/9 = 0.222
P(Lang = "Python" | Interviewed well = "True") = 4/9 = 0.444
P(Tweets = "Yes" | Interviewed well = "True") = 6/9 = 0.667
P(PhD = "No" | Interviewed well = "True") = 6/9 = 0.667

Compute $P(X | C_{False})$
P(Level = "Senior" | Interviewed well = "False") = 3/5 = 0.6
P(Lang = "Python" | Interviewed well = "False") = 2/5 = 0.4
P(Tweets = "Yes" | Interviewed well = "False") = 1/5 = 0.2
P(PhD = "No" | Interviewed well = "False") = 2/5 = 0.4

$P(X | C_i)$:
P(X | Interviewed well = "True") = 0.222 x 0.444 x 0.667 x 0.667 = 0.044
P(X | Interviewed well = "False") = 0.6 x 0.4 x 0.2 x 0.4 = 0.019

$P(C_i | X) = P(X | C_i) * P(C_i)$:
P(X | Interviewed well = "True") * P(Interviewed well = "True") = 0.044 * 0.643 = **0.028**
P(X | Interviewed well = "False") * P(Interviewed well = "False") = 0.019 * 0.357 = 0.007

Since **$P(C_{true} |X) > P(C_{false} |X)$**, therefore X belongs to class (Interviewed well = "True")

Text Classification

Text	Category
"A great game"	Sports
"the election was over"	Not sports
"Very clean match"	Sports
"A clean but forgettable game"	Sports
"It was a close election"	Not sports

Our goal is to build a Naïve Bayes classifier that will tell us which category the sentence " A very close game" belongs to.

$$p(Sports|a\ \ very\ close\ game) = \frac{p(a\ very\ close\ game|Sports) \times p(Sports)}{p(a\ very\ close\ game)}$$

$$p(Not\ sports|a\ \ very\ close\ game) = \frac{p(a\ very\ close\ game|Not\ sports) \times p(Not\ sports)}{p(a\ very\ close\ game)}$$

$$p(a\ very\ close\ game|\ Sports) = p(a|Sports) \times p(very|Sports) \times p(close|Sports) \times p(game|Sports)$$

Similarly

$$p(a\ very\ close\ game|\ Not\ Sports) = p(a|Not\ Sports) \times p(very|Not\ Sports) \times p(close|Not\ Sports) \times p(game|Not\ Sports)$$

But the word "close" does not exist in the category Sports, thus $\boldsymbol{p}$ ($\boldsymbol{close}|\ \boldsymbol{Sports}$)= **0**, leading to $\boldsymbol{p}$ ($\boldsymbol{a\ very\ close\ game\ Sports}$) = 0

Laplace smoothing

– $p(w|c) = \frac{\text{count(w,c)+1}}{\text{count(word,c)+count(word)}}$

count(w,c) = Count of word w in class c
count(word,c) = Count of words in class c.
count(word) = Count all the possible words in the dataset
– **Now we can calculate the probability again :**

$$p(close|Sports) = \frac{(0+1)}{(11+14)}$$

11 : how may words in class Sports

14: how many words in whole datasets without repetition

Calculation

p(Sports|a very close game) =?
p(Not Sports|a very close game) =?

w	$p(w\|Sports)$	$p(w\|Not\ Sports)$
a	$\frac{(2+1)}{(11+14)}$	$\frac{(1+1)}{(9+14)}$
very	$\frac{(1+1)}{(11+14)}$	$\frac{(0+1)}{(9+14)}$
close	$\frac{(0+1)}{(11+14)}$	$\frac{(1+1)}{(9+14)}$
game	$\frac{(2+1)}{(11+14)}$	$\frac{(0+1)}{(9+14)}$

Let Z = A Very Close Game ; S = Sport ; NS = Non Sport

Therefore probability of "A Very Close Game " being a Sport => P(S|Z) = P(Z|S) P(S)

P(A|S) x P(V|S) x P(C|S) x P(G|S) x P(S) = [3/25 x 2/25 x 1/25 x 3/x25] x 3/5 = 0.0004608 x 0.6 = 0.000027648

And the probability of "A Very Close Game " being a Non Sport => P(NS|Z) = P(Z|NS) P(NS)

P(A|NS) x P(V|NS) x P(C|NS) x P(G|NS) x P(NS) = [2/23 x 1/23 x 2/23 x 1/23] x 2/5 = 0.00001429 x 0.4=0.00000571

Then "a very close game" is belong to the Sports class

Decision Tree (P522)

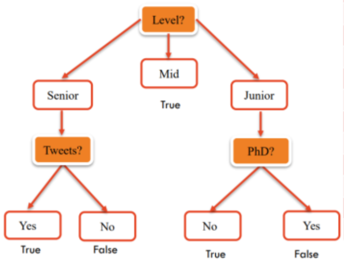
Information Gain (IG)
IG calculates effective change in entropy after making a decision based on the value of an attribute.
IG(Y|X) = H(Y) – H(Y|X)
where Y is a class label

X is an attribute
H(Y) is the entropy of Y
H(Y|X) is the cconditional entropy of Y given X

$H(Y) = -\sum_{i=i}^{m} p_i \log_2 (p_i)$, where p_i = P(Y =y_i) and m is the number of classes

– Higher entropy => higher uncertainty
– Lower entropy => lower uncertainty

The resulting tree:



Applicant	Level	Lang	Tweets	PhD	Interviewed well
A1	Senior	Java	No	No	False
A2	Senior	Java	No	Yes	False
A3	Mid	Java	No	No	True
A4	Junior	Python	No	No	True
A5	Junior	R	Yes	No	True
A6	Junior	R	Yes	Yes	False
A7	Mid	R	Yes	Yes	True
A8	Senior	Python	No	No	False
A9	Senior	R	Yes	No	True
A10	Junior	Python	Yes	No	True
A11	Senior	Python	Yes	Yes	True
A12	Mid	Python	No	Yes	True
A13	Mid	Java	Yes	No	True
A14	Junior	Python	No	Yes	False

Calculation

– H(Interviewed)=H(9,5)= $-\frac{9}{14} log_2(\overset{P(Y = True)}{\frac{9}{14}}) - \frac{5}{14} log_2(\overset{P(Y = False)}{\frac{5}{14}}) = 0.94$

– $H(\text{Interviewed}|_{X=level}) = \frac{5}{14} H(2,3) + \frac{4}{14} H(4,0) + \frac{5}{14} H(3,2) = 0.7$

$level$	$p(X = level)$	$H(Y\|X = level)$
Senior	0.365	$H(2,3)$ =0.971
Mid	0.27	$H(4,0)$=0
Junior	0.365	$H(3,2)$ = 0.971

– IG (level) = H(Interviewed) - $H(\text{Interviewed}|_{X=level})$ = **0.24**
– IG (tweets) = H(Interviewed) - $H(\text{Interviewed}|_{X=tweets})$ = 0.15
– IG (PhD) = H(Interviewed) - $H(\text{Interviewed}|_{X=PhD})$ = 0.048
– IG (lang) = H(Interviewed) - $H(\text{Interviewed}|_{X=lang})$= 0.029

Setting up a reliable evaluation (P549 – L10 p29)
Generalization error should model application as closely and reliably as possible • **Sample must be representative**
• **Larger sample better**

Data drift (non-stationary data)

What it is:
– Typical train/test setups assume stationarity
– Should be near-true for train and test samples
– Only near-true in production for a little while

What to do:
– Monitor offline metric on live data
– May require monitoring/annotation
– If there are large changes, then retrain on new data
– Online/incremental learning

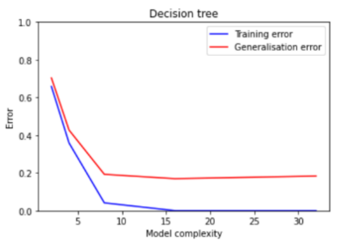
Building a good solution
Build a simple model first, evaluate, iterate
Ensembles of predictors often do very well
-random forest (bootstrap many trees, more biased, lower variance, lose explain ability of trees, boost performance)

Tutorial:

- Does training or generalisation error level out first? Why?

that **higher** values of max_depth may lead to **overfitting**.

- What is the best value of max_depth based on this plot



max_depth=8. This gives the best generalisation error with lower model complexity and less risk of overfitting.

- Why doesn't generalisation error increase on the right

The algorithm has other mechanisms to prevent overfitting. And overfitting does seem to hurt generalisation too much on this data. Nevertheless, decision trees can overfit so use with caution.

- Would it be useful to collect more training data?

Yes, almost always. However, it looks like both classifiers are close to their asymptotes. So the benefit might not be worth the cost. The decision tree would benefit more from additional data.

- The decision tree has a larger spread between training and generalisation error. Why is this?.

The decision tree suffers more from overfitting.The random forest on this particular data has 0 training error. This is a bit of a surprise as **random forests tend to increas bias**. With high bias, we would expect **underfitting** which tends to be characterised by both high training and high generalisation error. However, **random forests generally also reduce variance** enough to **cancel out any increase in bias**. Here we end up with a nice generalisation error plot that seems to be close to its asymptote and not too different from the training error.

- When is it OK to use the held-out test data from our train/dev/test split?

As little as possible. Ideally only once for our final generalisation error/accuracy calculation.

L8b – PCA (P446)

Aim: transforming the original data from high dimensional space into lower dimensional space.

Principal components (PC)
The new variables in the lower dimensional space corresponds to a linear combination of the originals

PCA helps in
-Visualization
-uncovering clusters
-dimensionality reduction
– PCA method is particularly useful when the variables within the data set are **highly correlated**.
– **Correlation indicates that there is redundancy in the data**.
– Correlation is captured by the covariance matrix1 .
– PCA is traditionally performed on covariance matrix or correlation matrix.

Covariance Matrix
three attributes (x,y,z):

$$C = \begin{pmatrix} \mathrm{cov}(x,x) & \mathrm{cov}(x,y) & \mathrm{cov}(x,z) \\ \mathrm{cov}(y,x) & \mathrm{cov}(y,y) & \mathrm{cov}(y,z) \\ \mathrm{cov}(z,x) & \mathrm{cov}(z,y) & \mathrm{cov}(z,z) \end{pmatrix} \text{Variances}$$

The covariance between one dimension and itself is the variance

– cov(x,y) = cov(y,x) hence matrix is symmetrical about the diagonal
– N-dimensional data will result in NxN covariance matrix

Covariance Matrix Example

Below is the covariance matrix of some 3 variables.
Their variances are on the diagonal, and the sum of the 3 values (3.448) is the overall variability

1.343730	-.1601522	.1864702
-.1601522	.61920562	-.1266842
.1864702	-.1266842	1.485549

The diagonal elements are the **variances** of the different variables.
In the covariance table above, the off-diagonal values are different from zero. This indicates the presence of redundancy in the data.
In other words, there is a certain amount of correlation between variables.

PCA Example
– PCA creates **uncorrelated** PC variables (**eigenvectors**) having **zero covariations and variances** (**eigenvalues**) sorted in **decreasing order**.
– The first PC captures the greatest variance , the second greatest variance is the second PC, and so on.
– By eliminating the later PCs we can achieve dimensionality reduction.

1.65135	.000000	.000000
.000000	1.220288	.000000
.000000	.0000000	.576843

The covariance matrix between the principal components

– The 1st PC accounts for or "explains" 1.651/3.448 = 47.9% of the overall variability; – the 2nd one explains 35.4% of it; the 3rd one explains 16.7% of it.

L8 – Clustering
Group data points into clusters such that
– Data points in one cluster are more similar to one another.
– Data points in separate clusters are less similar to one another.
– Distance function specifies the "closeness" of two objects.

– Distances are normally used to measure the similarity or dissimilarity between two data objects
– Some popular ones include: *Minkowski distance*:

$$d(i,j) = \sqrt[q]{(|x_{i_1} - x_{j_1}|^q + |x_{i_2} - x_{j_2}|^q + \dots + |x_{i_p} - x_{j_p}|^q)}$$

where $i = (x_{i1}, x_{i2}, \dots, x_{ip})$ and $j = (x_{j1}, x_{j2}, \dots, x_{jp})$ are two p-dimensional data objects, and *q* is a positive integer

– If *q* = *1*, *d* is Manhattan distance

$$d(i,j) = |x_{i_1} - x_{j_1}| + |x_{i_2} - x_{j_2}| + \dots + |x_{i_p} - x_{j_p}|$$

The University of Sydney Page 9

– *If q* = *2*, *d* is Euclidean distance:

$$d(i,j) = \sqrt{(|x_{i_1} - x_{j_1}|^2 + |x_{i_2} - x_{j_2}|^2 + \dots + |x_{i_p} - x_{j_p}|^2)}$$

– Properties
- *d(i,j)* ≥ 0
- *d(i,i)* = 0
- *d(i,j)* = *d(j,i)*
- *d(i,j)* ≤ *d(i,k)* + *d(k,j)*

Hierarchical clustering A method of cluster analysis which seeks to build a hierarchy of clusters. It produces a set of nested clusters organized as a hierarchical tree
Agglomerative (bottom up), Divisive (top down)
Partitional clustering A division data objects into non-overlapping subsets (clusters) such that each data object is in exactly one subset

Agglomerative
Initial – Each point in its own cluster, until: single cluster

Distance Calculation between two hierarchical clusters:
– single linkage:
 – The minimum distance between elements of each cluster
– complete linkage:
 – The maximum distance between elements of each cluster
– average linkage: i.e. mean distance calculation.

K-Means Clustering
Complexity is O(n * k * i * d)
n = number of points, k = number of clusters, i = number of iterations, d = number of attributes (or dimensions)

Measures of Cluster Validity
– External Index: Measure the extent to which cluster labels match externally supplied class labels (e.g., **accuracy, precision, recall, F1-score**)
– Internal Index: Measure the goodness of a clustering structure without respect to external information (e.g., **Sum of Squared Error**)
– Relative Index: Compare two different clusterings or clusters (often an external or internal index is used)

Homogeneity ranges from 0 to 1, measuring whether clusters contain data points that are part of a single class (analogous to precision, P = TP / (TP+FP))
Completeness ranges from 0 to 1, measuring whether classes contain data points that are part of a single cluster (analogous to recall, R = TP / (TP+FN))
V-measure is the harmonic mean of homogeneity and completeness (analogous to F1 score = 2PR / (P+R))

Internal: Sum of squared Error (SSE, Inertia)

$$SSE = \sum_{i=1}^{K} \sum_{x \in C_i} dist^2(m_i, x)$$

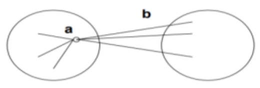
- Suppose we have 3 clusters:
  - Cluster 1: [2, 4] with centroid at 3
  - Cluster 2: [5, 6, 7] with centroid at 6
  - Cluster 3: [8, 10, 12] with centroid at 10
- Squared error for each cluster:
  - SE1 = $(2-3)^2 + (4-3)^2 = 1 + 1 = 2$
  - SE2 = $(5-6)^2 + (7-6)^2 = 1 + 1 = 2$
  - SE3 = $(8-10)^2 + (12-10)^2 = 4 + 4 = 8$
- SSE = SE1 + SE2 + SE3 = 12

For an individual point *i*

- Calculate ***a*** = average distance of *i* to points in its cluster
- Calculate ***b*** = average distance of *i* to points in the next nearest cluster
- The silhouette coefficient for a point is then given by

$s = 1 - a/b$ if $a < b$, (or $s = b/a - 1$ if $a \geq b$, not the usual case)

- The closer to 1 the better



Silhouette coefficient for dataset is average across all *i*

Silhouette Coefficient Example P38
Using Silhouettes to choose k
High average silhouette indicates points far away from neighbouring clusters

Pre-Processing for Clustering
Data cleansing/ Data Transformation/Data normalisation/Dimensionality Reduction / choice or projection of dimensions

L7 Association Rule Mining (P343)
Application of ML:
Creating and using models that are learned from data
– Predicting whether an email is spam or not
– Discovering hidden rules in complex datasets
– Predicting whether a credit card transaction is fraudulent
– Predicting tumour cells as benign or malignant

Supervised vs. Unsupervised Learning
Supervision: The training data are accompanied by labels indicating the class of the observations
Unsupervised learning (e.g. clustering and association rules)
– The class labels of training data is unknown
– Given a set of measurements, observations, etc. with the aim of • Establishing the existence of classes or clusters in the data • Discovering hidden patterns or rules

Itemset is a collection of one or more items e.g {Milk,Bread,Diaper}
A k-itemset is an itemset containing k items, 3-itemset

Support count (σ) Support (s)
A frequent itemset has **s ≥ min_support**

TID	Items
1	Bread, Milk
2	Bread, Diaper, Beer, Eggs
3	Milk, Diaper, Beer, Coke
4	Bread, Milk, Diaper, Beer
5	Bread, Milk, Diaper, Coke

Market-basket transactions
TID: Transaction Identifier
Items: Transaction item set

- **Support count** (σ) is the itemset frequency
  $\sigma(\{\text{Milk,Diaper,Beer}\}) = 2$
- **Support** (s) is the normalised itemset frequency
  $$s = \frac{\sigma(\{\text{Milk,Diaper,Beer}\})}{|T|} = \frac{2}{5}$$
- A **frequent itemset** has *s* ≥ *min_support*

Finding Frequent Itemsets

TID	Items
1	Beer, Nuts, Diaper
2	Beer, Coffee, Diaper
3	Beer, Diaper, Eggs
4	Nuts, Eggs, Milk
5	Nuts, Coffee, Diaper, Eggs, Milk

Market-basket transactions
TID: Transaction Identifier
Items: Transaction item set

- Let *min_support* = 50%
- Freq. 1-itemsets:
  - Beer:3(60%);
    Nuts:3(60%);
    Diaper:4(80%);
    Eggs:3(60%)
- Freq. 2-itemsets:
  - {Beer, Diaper}:3(60%)

An association rule is an implication of the form X◊Y where X and Y are itemsets {Milk,Diaper}◊{Beer}
Confidence (c) **c ≥ min_conf**

Definition: Association Rule

TID	Items
1	Bread, Milk
2	Bread, Diaper, Beer, Eggs
3	Milk, Diaper, Beer, Coke
4	Bread, Milk, Diaper, Beer
5	Bread, Milk, Diaper, Coke

Market-basket transactions
TID: Transaction Identifier
Items: Transaction item set

- An **association rule** is an implication of the form X→Y where X and Y are itemsets
  {Milk,Diaper}→{Beer}
- **Confidence** (c) measures how often Y occurs in transactions with X
  $$c = \frac{\sigma(\{\text{Milk,Diaper,Beer}\})}{\sigma(\{\text{Milk,Diaper}\})} = \frac{2}{3}$$
- An **association rule** has c ≥ min_conf

Finding Association Rules

TID	Items
1	Beer, Nuts, Diaper
2	Beer, Coffee, Diaper
3	Beer, Diaper, Eggs
4	Nuts, Eggs, Milk
5	Nuts, Coffee, Diaper, Eggs, Milk

Market-basket transactions
TID: Transaction Identifier
Items: Transaction item set

- Let *min_support* = 50%, min_conf = 50%
- Freq. Pat.: Beer:3, Nuts:3, Diaper:4, Eggs:3, {Beer, Diaper}:3
- Association rules:
  - Beer → Diaper (100%)
  - Diaper → Beer (75%)

Support count of {beer,diaper}/ support count of {beer}
Support count of {beer,diaper}/ support count of {diaper}

Mining Association Rules
1. **Frequent itemset generation** – Generate all itemsets with s ≥ min_support
2. **Rule generation** – Generate high-confidence rules from each frequent itemset – Each rule is a binary partitioning of a frequent itemset
Easy! But brute force enumerate is **computationally prohibitive**.

Apriori Principle
If an itemset is frequent, then all of its subsets are also frequent

anti-monotone property of support
If an itemset is infrequent, then its supersets are also infrequent

Gradient Descent

Gradient Descent is a very generic optimization algorithm capable of finding **optimal solutions** to a wide range of problems. The general idea of Gradient Descent is to **tweak parameters iteratively** in order to **minimize a cost function.**

Suppose you are lost in the mountains in a dense fog; you can only feel the slope of the ground below your feet. A good strategy to get to the bottom of the valley quickly is to go downhill in the direction of the steepest slope. This is exactly what Gradient Descent does: it measures the local gradient of the error function with regards to the parameter vector θ, and it goes in the direction of **descending gradient**. Once the gradient is zero, you have reached a minimum!

Concretely, you start by filling θ with **random values** (this is called **random initialization**), and then you improve it **gradually**, taking **one baby step** at a time, each step attempting to decrease the **cost function (e.g., the MSE),** until the algorithm **converges to a minimum**.

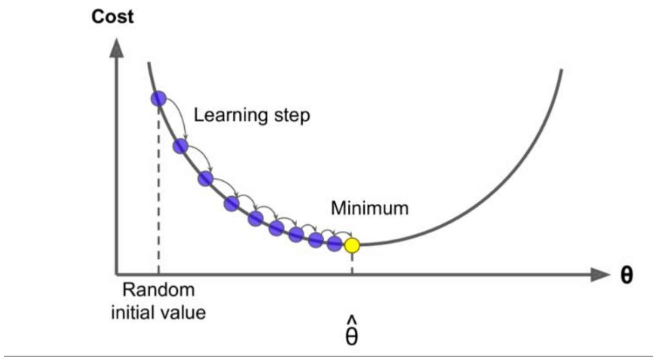


Figure 4-3. Gradient Descent

An important parameter in Gradient Descent is the size of the steps, determined by the **learning rate** hyperparameter. If the learning rate is too **small**, then the algorithm will have to go through many iterations to converge, which will take a **long time**.

On the other hand, if the learning rate is **too high**, you might jump across the valley and end up on the other side, possibly even higher up than you were before. This might make the algorithm **diverge**, with larger and larger values, failing to find a good solution.