Machine Learning

Lecture 01

Min-Kuan Chang

minkuanc@nchu.edu.tw

EE, College of EECS

Fundamentals in Machine Learning
Topic 01
Basic Concepts

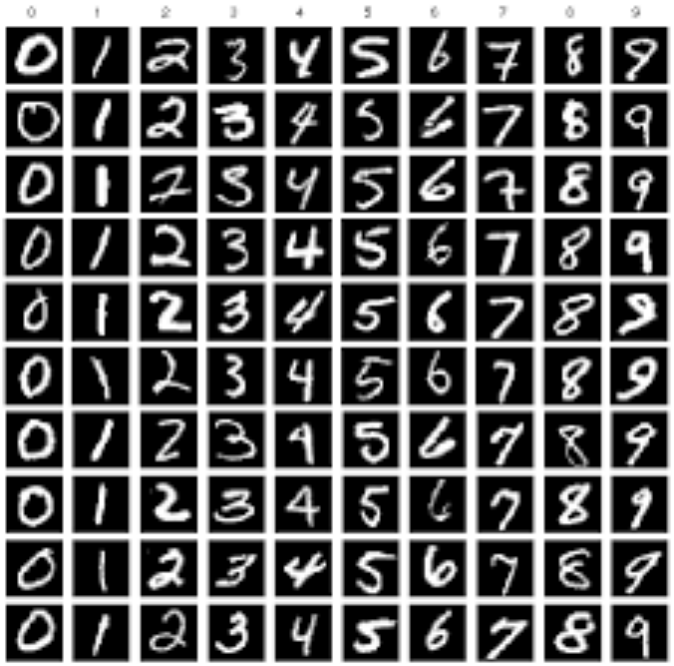
Supervised Learning

- As known as the predictive learning
- The goal is to map an input x to an output y based on the observations of (x_i, y_i) for $i = 1,2,\cdots,N$
  - these observations are called the training set
  - N is called the number of examples
  - x_i is a D-dimensional vector of numbers, which are called features, attributes or covariates
  - y_i is is a categorical or nominal variable from some finite set or a real-valued scalar
    - classification and regression

Supervised Learning

Where is Bob?

Supervised Learning

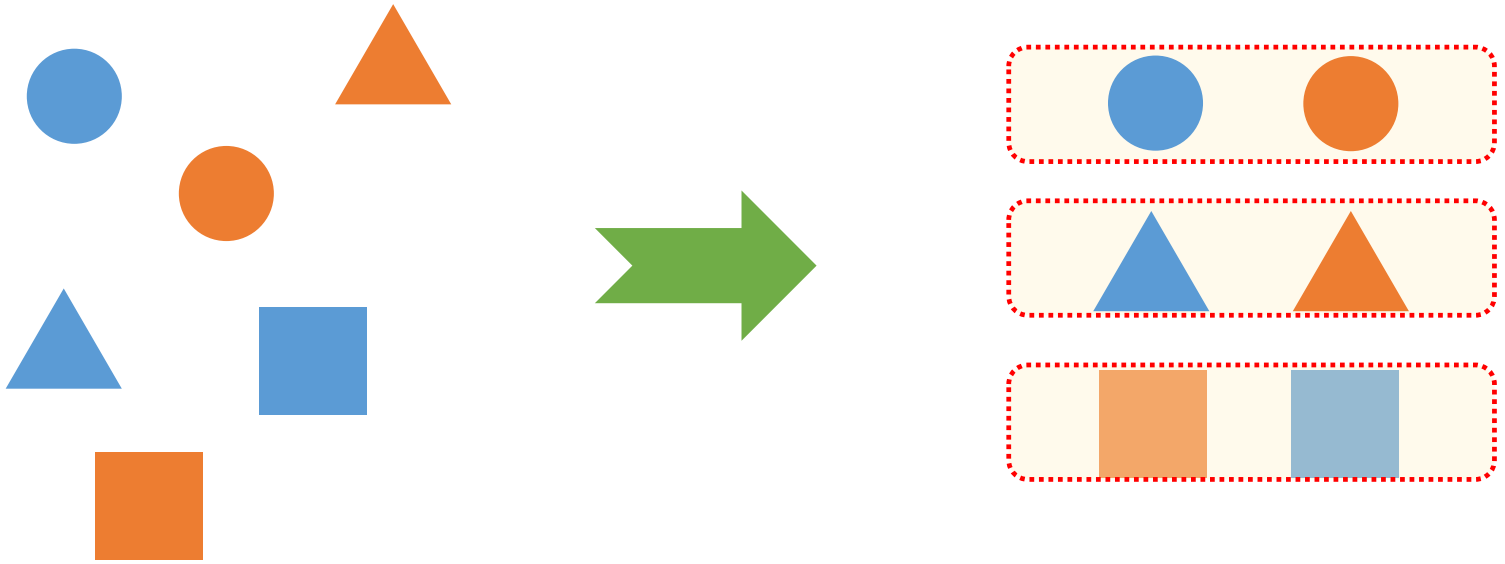


Is this 3?

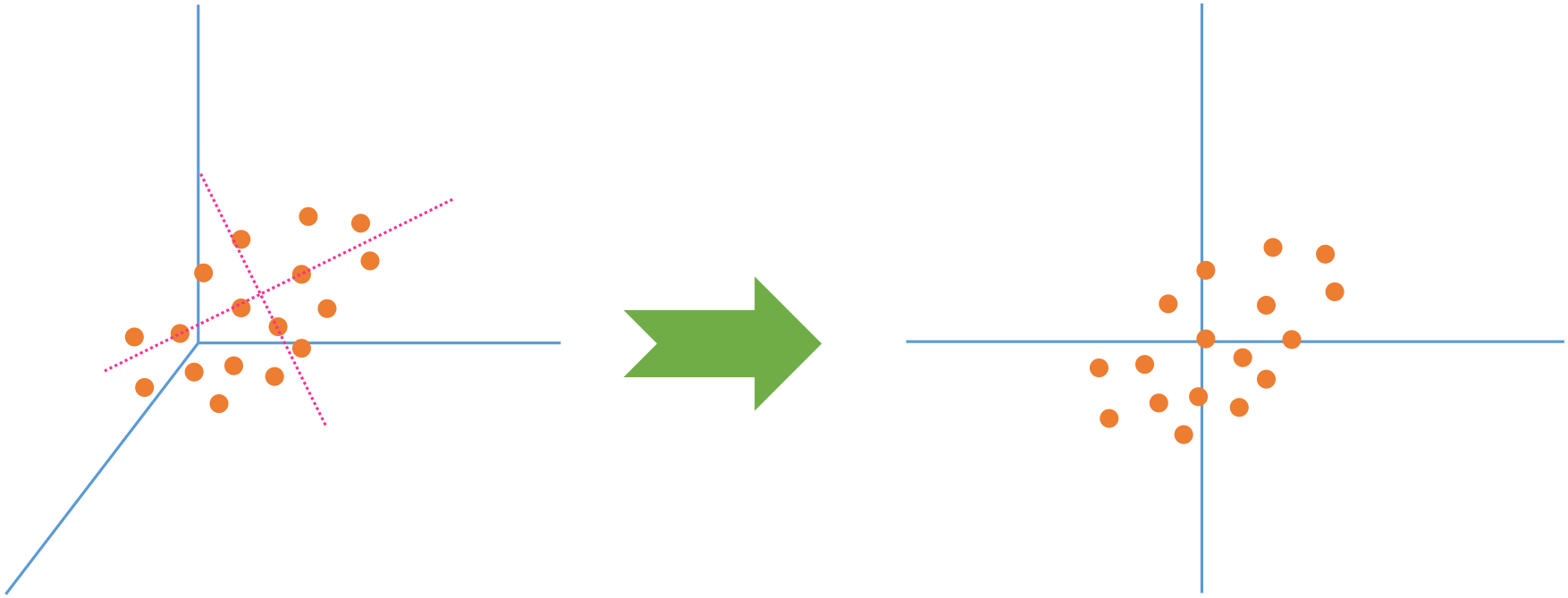
Unsupervised Learning

- Also known as the descriptive learning
- The goal is to find the "interesting information" given the input data x_i for for $i = 1,2,\cdots,N$
  - to discover groups of similar examples within the data (clustering)
  - to determine the distribution of data within the input space (density estimation)
  - to project the data from a high-dimensional space down to two or three dimensions (dimension reduction)

Unsupervised Learning



Unsupervised Learning



Parametric Models

- Models come with fixed parameters
- Simplify the learning process but also limit what can be learned from the training data
- Designing such a model involves two steps:
  - step 1: assume the type of functions used in the model
  - step 2: learn the parameters assumed in the chosen model
- Examples of parametric models
  - Linear regression
  - Logistic regression
  - Perceptron
  - Naive Bayes

Parametric Models

- Advantages of parametric models
  - easier to understand and interpret results
  - very fast to learn from data
  - less training data needed
- Disadvantages of parametric models
  - constrained to the specified form
  - suited and limited to simpler problems
  - unlikely to match the underlying 'true' mapping function between the input data and their respective output label or values

Non-parametric Models

- Do not make strong assumptions about the form of the mapping function
  - seek to best fit the training data in constructing the mapping function
  - maintain the ability to generalize to unseen data
- Suitable for the situation where large training dataset is available
- Examples of non-parametric models
  - k-Nearest Neighbors
  - Decision Trees
  - Support Vector Machines

Non-parametric Models

- Advantages of non-parametric models
  - capable of fitting a large number of functional forms
  - no or less assumptions about the underlying mapping function
  - higher performance models for prediction
- Disadvantages of non-parametric models
  - require much more training
  - much more parameters to train
  - easier to overfit the training data
  - harder to explain the results

No Free Lunch Theorem

- It states that no single machine learning algorithm is universally the best-performing algorithm for all problems
  - need to develop many different types of models
    - to cover the wide variety of data that occurs in the real world
  - for each model, different algorithms are developed to to train the model
    - speed-accuracy-complexity tradeoffs
  - imply that each machine learning algorithm has its own advantages and disadvantages due to its prior assumptions

Inductive Reasoning

- A form of reasoning where we draw conclusions about the world based on past observations
- This is exactly what machine learning algorithms do
- However, the black swan paradox shows how limited machine learning algorithms are
  - a machine learning algorithm trained with images of all white swan can be disproved by just one counter-example (the image of a black swan)
  - we cannot apply a conclusion about a particular set of observations to a more general set of observations

Fundamentals in Machine Learning
Topic 02
Mathematical Fundamentals

Probability Theory

- What is probability?
- What does it means when we say the probability that a coin land heads is 0.5?
  - the frequentist viewpoint
    - probabilities represent long run frequencies of events
    - this means if we flip the coin many times, we expect it to land heads about half the time
  - the Bayesian viewpoint
    - probabilities are used to quantify the uncertainty about events
      - fundamentally related to information rather than repeated trials
    - this means that we believe the coin is equally likely to land heads or tails on the next toss

Probability Theory

- Axioms of probability
  - the probability of an event is between 0 and 1
  - the probability of the sample space is 1
  - the probability of the union of mutually exclusive events is the sum of the probabilities of them
- The probability of the union of two events
- The joint probability of two events
- The conditional probability
- Independence and conditional independence
- The Bayes rule

Probability Theory

- A random variable is a variable whose possible values are numerical outcomes of a random phenomenon
  - discrete random variables
    - take on only a countable number of distinct values
    - probability mass function
    - example:
      - Bernoulli random variable, binominal random variable, Poisson random variable
  - continuous random variables
    - take on a uncountable number of real values
    - probability density function
    - example:
      - uniform random variable, Gaussian random variable, exponential random variable

Probability Theory

- Statistical property of random variables
  - mean
  - variance
  - moments and joint moments
  - covariance function
  - autocorrelation function
  - correlation coefficient
  - orthogonality and uncorrelated-ness

Mathematical Programming

- A mathematical programming has the following form

$$\min_{\boldsymbol{x}} J(\boldsymbol{x})$$

subject to

$$f_i(\boldsymbol{x}) \leq b_i \text{ for } i = 1,2,\cdots,m$$

Mathematical Programming

- A mathematical programming can be categorized into
  - linear and non-linear programming
    - a mathematical programming is a linear programming if $J(\boldsymbol{x})$ and $f_i(\boldsymbol{x})$ for $i = 1,2,\cdots,m$ are all linear
  - convex and non-convex programming
    - a mathematical programming is a convex programming if

      $J(\lambda \boldsymbol{x}_1 + (1-\lambda)\boldsymbol{x}_2) \leq \lambda J(\boldsymbol{x}_1) + (1-\lambda)J(\boldsymbol{x}_2)$

      $f_i(\lambda \boldsymbol{x}_1 + (1-\lambda)\boldsymbol{x}_2) \leq \lambda f_i(\boldsymbol{x}_1) + (1-\lambda)f_i(\boldsymbol{x}_2)$ for $i = 1,2,\cdots,m$
  - smooth and non-smooth optimization
    - a mathematical programming is a smooth optimization if programming if $J(\boldsymbol{x})$ and $f_i(\boldsymbol{x})$ for $i = 1,2,\cdots,m$ are all differentiable

Mathematical Programming

- How to solve a mathematical programming?

$$\min_{\boldsymbol{x}} J(\boldsymbol{x})$$

subject to

$$f_i(\boldsymbol{x}) \leq b_i \text{ for } i = 1,2,\cdots,m$$

- It is difficult to find the optimal solution when it comes to the non-smooth optimization
- The optimal solution is attainable in the smooth convex optimization
  - the gradient approach can always be adopted to find the optimal solution iteratively

Mathematical Programming

- Now we meet an important approach called the gradient approach
- What is the gradient? and What is the gradient approach?
- The gradient of $f(\boldsymbol{x})$ for $\boldsymbol{x} \in \mathbb{R}^D$ and $\boldsymbol{x} = [x_1, x_2, \cdots, x_D]^T$ is defined as

$$\nabla f = \left[\frac{\partial f}{\partial x_1}, \quad \frac{\partial f}{\partial x_2}, \quad \cdots, \quad \frac{\partial f}{\partial x_D}\right]^T$$

  - the rate of change in each direction
  - the direction of the greatest increase of $f(\boldsymbol{x})$
- What does the gradient have do with the gradient approach?

Mathematical Programming

- Recall that a mathematical programming has the following form

$$\min_{\boldsymbol{x}} J(\boldsymbol{x})$$

subject to

$$f_i(\boldsymbol{x}) \leq b_i \text{ for } i = 1,2,\cdots,m$$

- When $\boldsymbol{x}$ is updated in the direction of the decrease of $J(\boldsymbol{x})$ subject to the constraints, the minimum of $J(\boldsymbol{x})$ can be reached
- The constraints complicate the search of the appropriate direction for $\boldsymbol{x}$ to be updated

Mathematical Programming

- With the help of the Lagrange multiplier, we can transform the constrained optimization problem into an unconstrained one

$$\min_{\boldsymbol{x}} J(\boldsymbol{x}) \qquad \text{subject to} \qquad f_i(\boldsymbol{x}) \le b_i \ \text{ for } \ i = 1,2,\cdots,m$$

Lagrange multiplier

$$\min_{\boldsymbol{x}} \hat{J}(\boldsymbol{x}), \text{where } \hat{J}(\boldsymbol{x}) = J(\boldsymbol{x}) + \sum_{i=1}^{m} \lambda_i (f_i(\boldsymbol{x}) - b_i)$$

Gradient approach

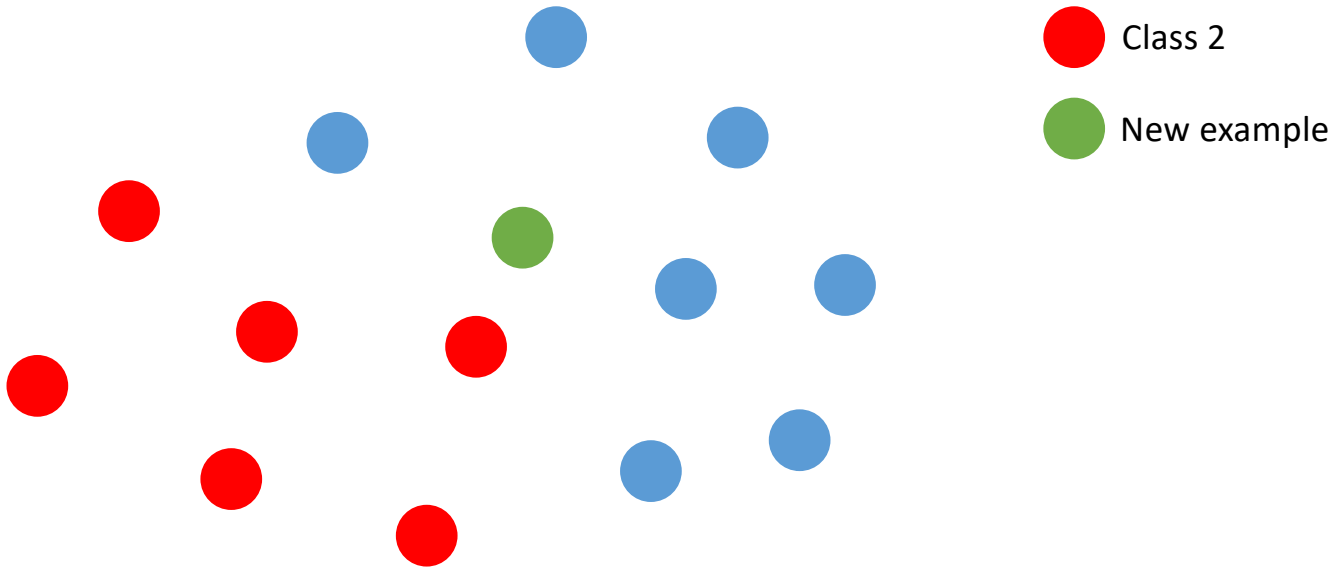
$$\boldsymbol{x}^{(i+1)} = \boldsymbol{x}^{(i)} - \gamma \nabla \hat{J}(\boldsymbol{x})$$

Fundamentals in Machine Learning
Topic 03
K-Nearest Neighbors Algorithm

- KNN is one of the simplest machine learning algorithm
  - assume the similarity between of the new example (data) and the examples in dataset exists
  - this similarity enables KNN to put this new example to the corresponding class
- KNN is an non-parametric algorithm
  - no assumption on underlying data
- KNN is sometimes called the lazy learner algorithm
  - does not learn from the training data
    - no model is built explicitly
  - training data are stored and used when a new example needs classifying
  - an example of memory-based learning or instance-based learning

- The basic idea of KNN
  - due to the similarity among examples from the same class, 'ideally' they should stay in the similar region in 'space'
- The way KNN classifies a new example is based on two factors
  - the "distance" between this new example and the examples in the dataset
  - the information of labels of the "closest" neighbors
- KNN algorithm classifies an unlabeled test sample based on the majority of similar samples among the K-nearest neighbors that are the closest to test sample
  - K is a number that should be chosen wisely

- How does KNN works?



- Basic KNN Algorithm
  - Input: Dataset $\mathcal{D} = \{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \cdots, (\boldsymbol{x}_N, y_N)\}$, K, test example $\boldsymbol{x}$
  - Step 1: Compute the distance between $\boldsymbol{x}$ and $\boldsymbol{x}_i$ for $i = 1,2, \ldots, N$

x=(1,1,2)	distance	class
$\boldsymbol{x}_1$=(1,2,3)	$\sqrt{2}$	1
$\boldsymbol{x}_2$=(1,3,2)	$\sqrt{4}$	2
$\boldsymbol{x}_3$=(2,4,5)	$\sqrt{9}$	2
$\boldsymbol{x}_4$=(2,2,3)	$\sqrt{3}$	2

- Basic KNN Algorithm
  - Input: Dataset $\mathcal{D} = \{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \cdots, (\boldsymbol{x}_N, y_N)\}$, K, test example $\boldsymbol{x}$
  - Step 1: Compute the distance between $\boldsymbol{x}$ and $\boldsymbol{x}_i$ for $i = 1,2, \ldots, N$
  - Step 2: Choose K samples in $\mathcal{D}$ that are nearest to $\boldsymbol{x}$

x=(1,1,2)	distance	class	K=1	K=3
$\boldsymbol{x}_1$=(1,2,3)	$\sqrt{2}$	1		
$\boldsymbol{x}_2$=(1,3,2)	$\sqrt{4}$	2		
$\boldsymbol{x}_3$=(2,4,5)	$\sqrt{9}$	2		
$\boldsymbol{x}_4$=(2,2,3)	$\sqrt{3}$	2		

- Basic KNN Algorithm
  - Input: Dataset $\mathcal{D} = \{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \cdots, (\boldsymbol{x}_N, y_N)\}$, K, test example $\boldsymbol{x}$
  - Step 1: Compute the distance between $\boldsymbol{x}$ and $\boldsymbol{x}_i$ for $i = 1,2, \ldots, N$
  - Step 2: Choose K samples in $\mathcal{D}$ that are nearest to $\boldsymbol{x}$
  - Step 3: Assign $\boldsymbol{x}$ the class it that is the most frequent class

x=(1,1,2)	distance	class	$K = 1$	$K = 3$
$\boldsymbol{x}_1$=(1,2,3)	$\sqrt{2}$	1		
$\boldsymbol{x}_2$=(1,3,2)	$\sqrt{4}$	2		
$\boldsymbol{x}_3$=(2,4,4)	$\sqrt{9}$	2		
$\boldsymbol{x}_4$=(2,2,3)	$\sqrt{3}$	2		

x=(1,1,2)	Assigned class
$K = 1$	1
$K = 3$	2

- The distance is an important metric to evaluate how similar two examples are
- In general, the choices of distance
  - Minkowski Distance

$$d(\boldsymbol{x}, \boldsymbol{y}) = \sqrt[p]{\sum_{i=1}^{D} |x_i - y_i|^p}$$

  - when $p = 1$, it becomes the Manhattan distance
  - when $p = 2$, it becomes the Euclidean distance
  - when $p = \infty$, it becomes the Chebyshev distance, $d(\boldsymbol{x}, \boldsymbol{y}) = \max_i |x_i - y_i|$

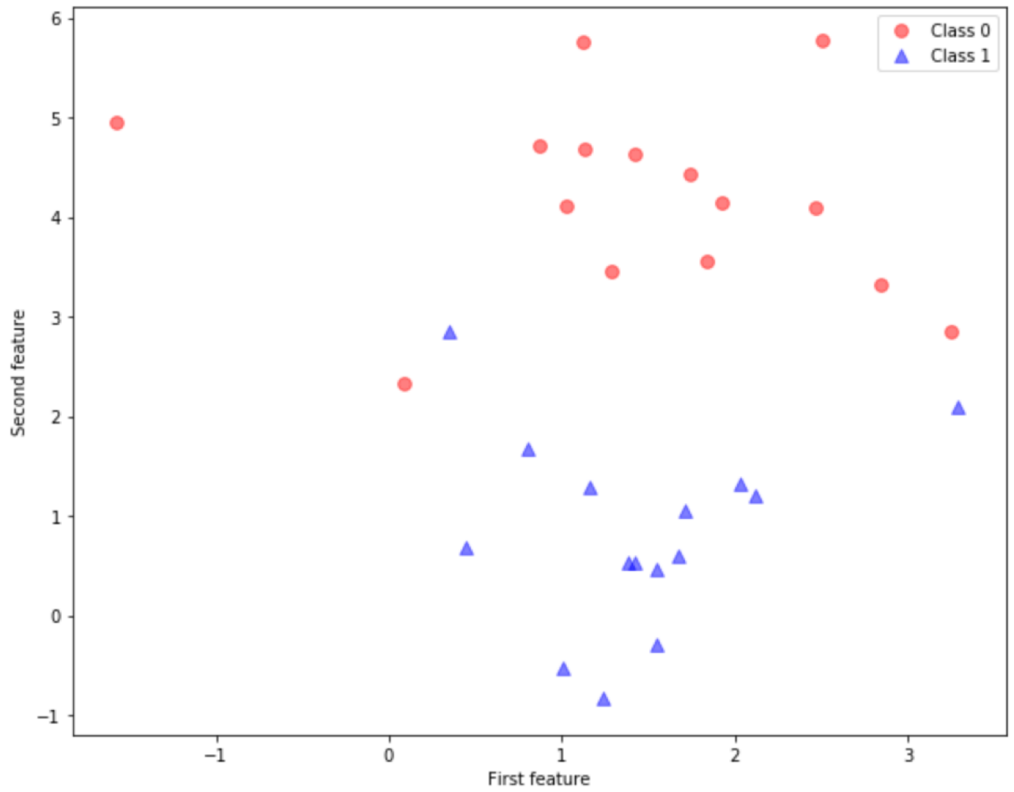
- The distance is an important metric to evaluate how similar two examples are
- In general, the choices of distance
  - Cosine Distance

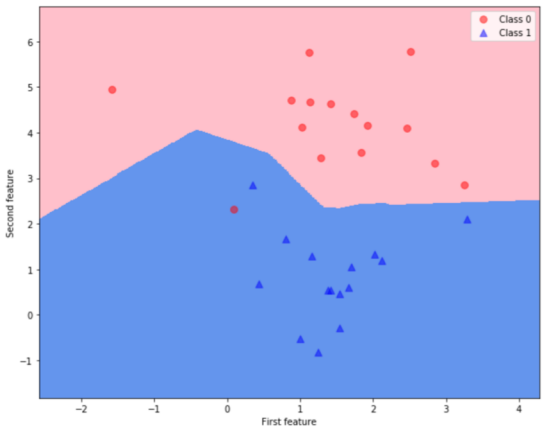
$$d(\boldsymbol{x}, \boldsymbol{y}) = 1 - \frac{\langle \boldsymbol{x}, \boldsymbol{y} \rangle}{\|\boldsymbol{x}\| \cdot \|\boldsymbol{y}\|}$$

    - also called the angular distance
  - There are a lot more
    - Hassanat Distance, Hamming Distance, Correlation Distance, Kullback-Leibler distance, to name a few

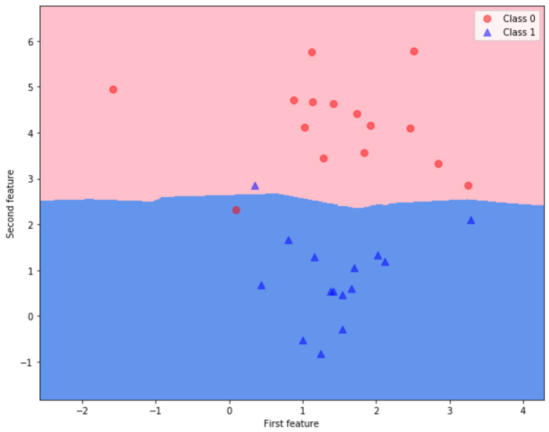
- Comments on the distance measure in KNN
  - the performance of KNN classifier depends significantly on the distance used
    - large gaps between the performances of different distances
  - no optimal distance metric that can be used for all types of datasets
    - complies with the no-free-lunch theorem
  - scaling issue
    - features may have to be scaled
      - prevent distance measures from being dominated by one of the features

- The advantage of KNN
  - simple to implement
    - no need to build a model, tune several parameters, or make additional assumptions
  - robust to the noisy training data
  - could be more effective if the training data is large
  - versatile
    - can be used for classification, regression, and search
- The disadvantage of KNN
  - need to determine the value of *K*
    - may be complex
  - high computation cost
    - calculating the distance between the data points for all the training samples
  - do not work well with high dimensional inputs

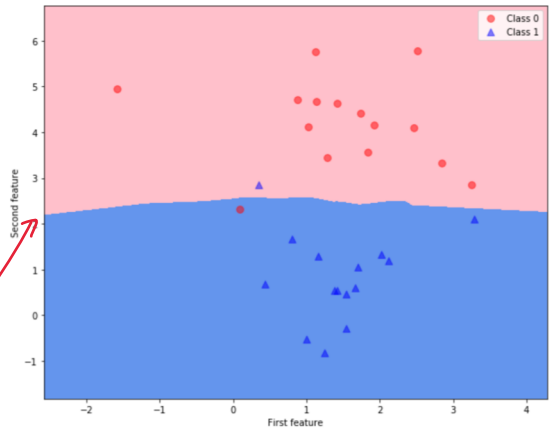

Class 0
Class 1
Second feature
First feature
6
5
4
3
2
1
0
−1
−1
0
1
2
3



$K = 1$



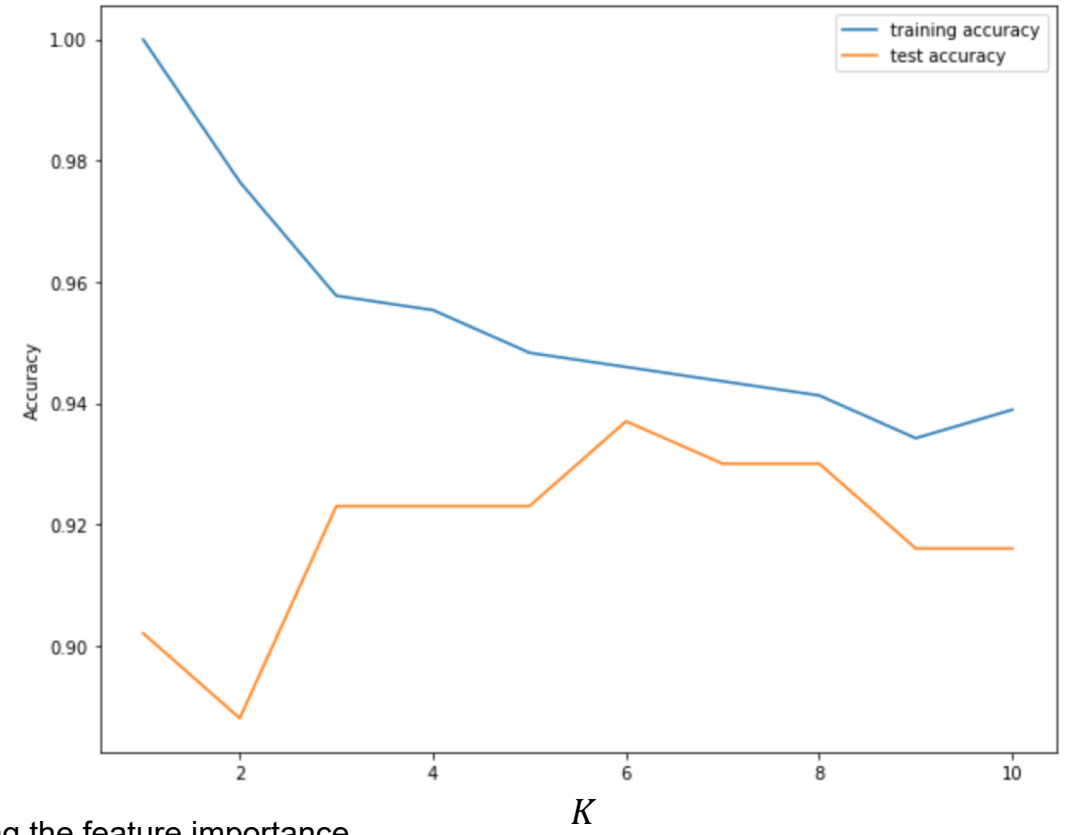
$K = 3$



$K = 12$

decision boundary

Breast Cancer



KNN
=> No intormation regarding the feature importance

Fundamentals in Machine Learning
Topic 04
Decision Tree – Part I

Tree => features (or subset of features) => explain

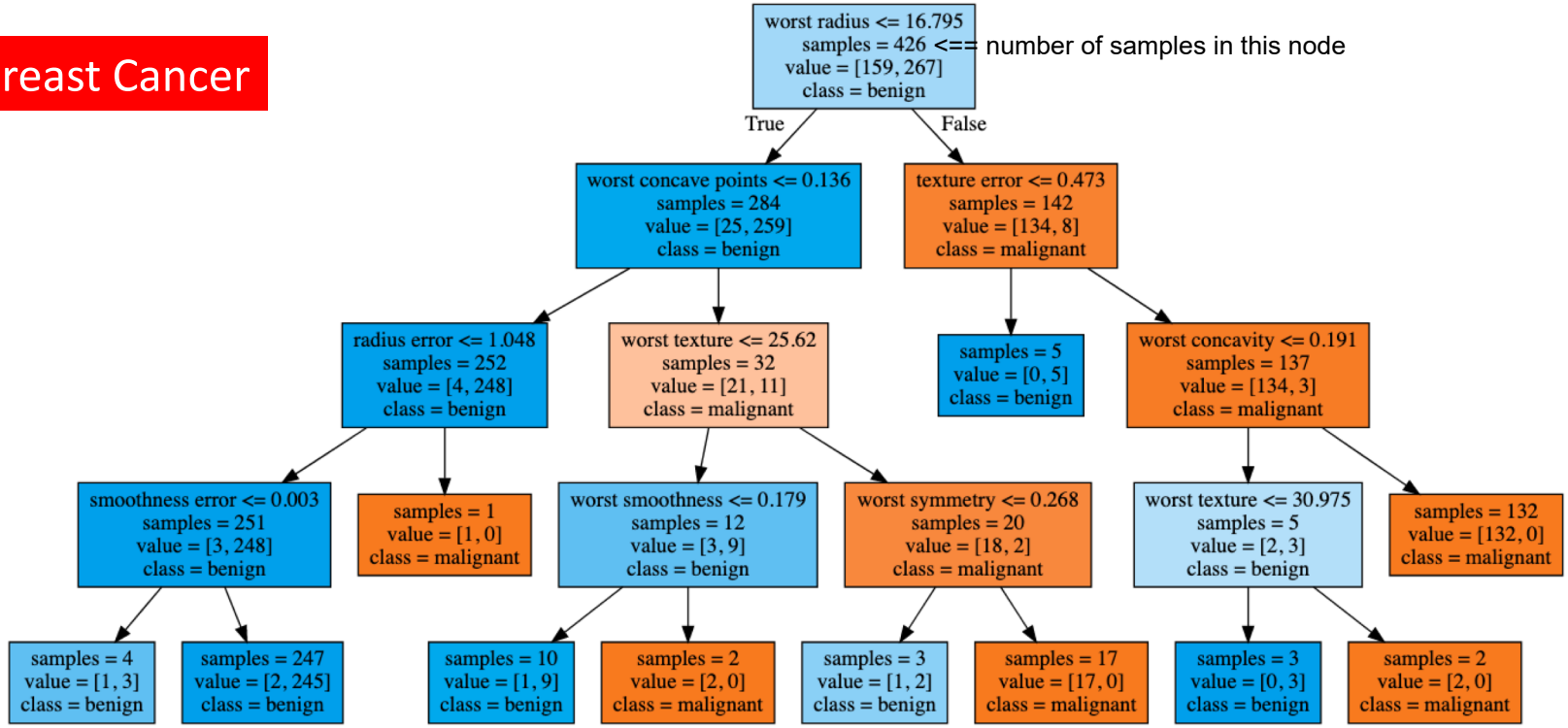
Overfitting when tree is too big

- A supervised machine learning
- Use a tree structure to model decisions, outcomes and predictions
- The decision tree consists of nodes, branches and leaves (無下一節點的 node)
  - the tree is constructed via a set of if-else statements
    - split, classify, and visualize a dataset based on different conditions
  - each internal node represents a test on a feature of a dataset
    - split the instance space into two (or more) sub-spaces according to a certain discrete function of the input attributes values
  - each leaf node represents an outcome
  - branches represent the decision rules or feature conjunctions that lead to the respective class labels

"optimal tree (最佳決策樹)" => bcz its hard => sub-optimal tree

每次做分支時 找到最有利的特徵做區分

Breast Cancer



- Advantages of decision tree
  - simple to understand and to interpret
    - trees can be visualized
  - require little data preparation
  - able to handle both numerical and categorical data
  - perform well

scalar

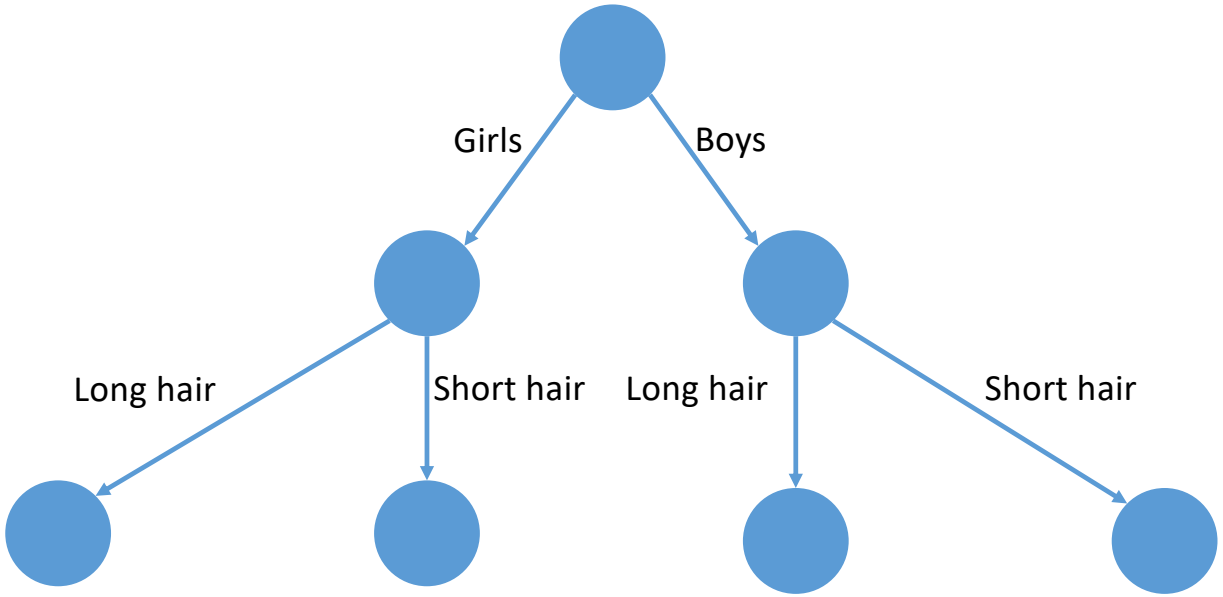
like labels {a, b} , {cat, dog} ...

- Disadvantages of decision tree

沒有對節點做限制的話容易overfitting

  - decision-tree learners can create over-complex trees that do not generalize the data well
  - decision trees can be unstable ==> 改用 "random forest" or "ensemble tree"
    - small variations in the data might result in a completely different tree
  - decision tree learners create biased trees if some classes dominate 偏向正確率較多的資料
  - the problem of learning an optimal decision tree is known to be NP-complete
    - heuristic algorithms cannot guarantee to return the globally optimal decision tree

suboptimal solution


Girls
Boys
Long hair
Short hair
Long hair
Short hair

How to Build a Decision Tree

- In principle, there are many decision trees that can be built based on the available set of features in data
- Among these possible decision tree, there exists an optimal decision tree that can provide the best performance
  - finding such an optimal decision tree is computationally infeasible
  - efficient algorithms are developed to find a suboptimal decision tree and provide reasonable performance
    - the greedy strategy is commonly adopted find the optimal splitting based on current tree
    - the decision tree is grown by a series of locally optimum decisions about which features to use to partition the data

How to Build a Decision Tree

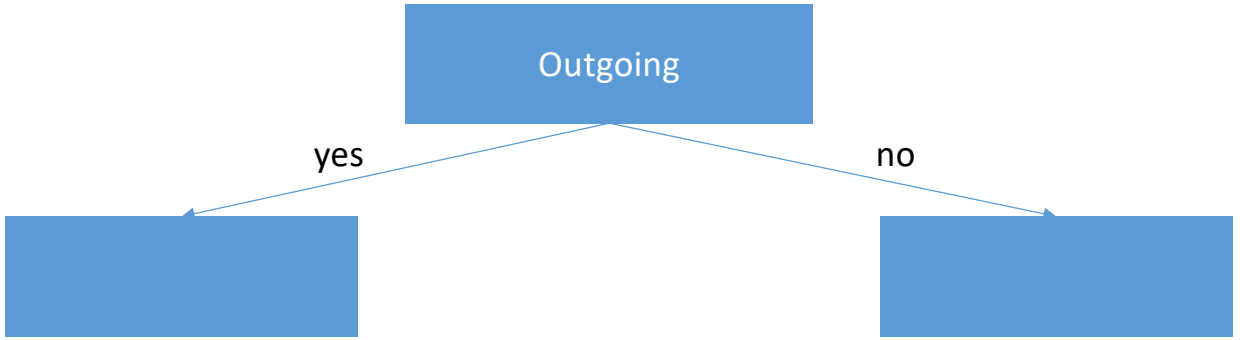
- Hunt's Algorithm
  - the basis of many existing decision tree algorithms such as ID3, CART, C4.5, C5 and etc.
  - consist of two steps:
    - let $\mathcal{D}_s$ be the training examples at node s and $C = \{C_0, C_1, \cdots, C_{K-1}\}$ be the class labels
    - step 1: if all training examples in $\mathcal{D}_s$ belong to the same class, C_j. Node s is a leaf labelled as C_j
    - step 2: if if all training examples in $\mathcal{D}_s$ do not belong to the same class,
      - step 2.1: select an attribute test condition to partition $\mathcal{D}_s$ into smaller ones
      - step 2.2: create a node for each test outcome and repeat step 1 for each child node

Illustrative Example for Hunt's Algorithm

ID	Study Hours	Outgoing	Relationship	Age	X'mas party
1	3	No	No	18	No
2	8	Yes	Yes	19	Yes
3	4	No	Yes	20	Yes
4	10	No	No	22	No
5	9	Yes	Yes	18	No

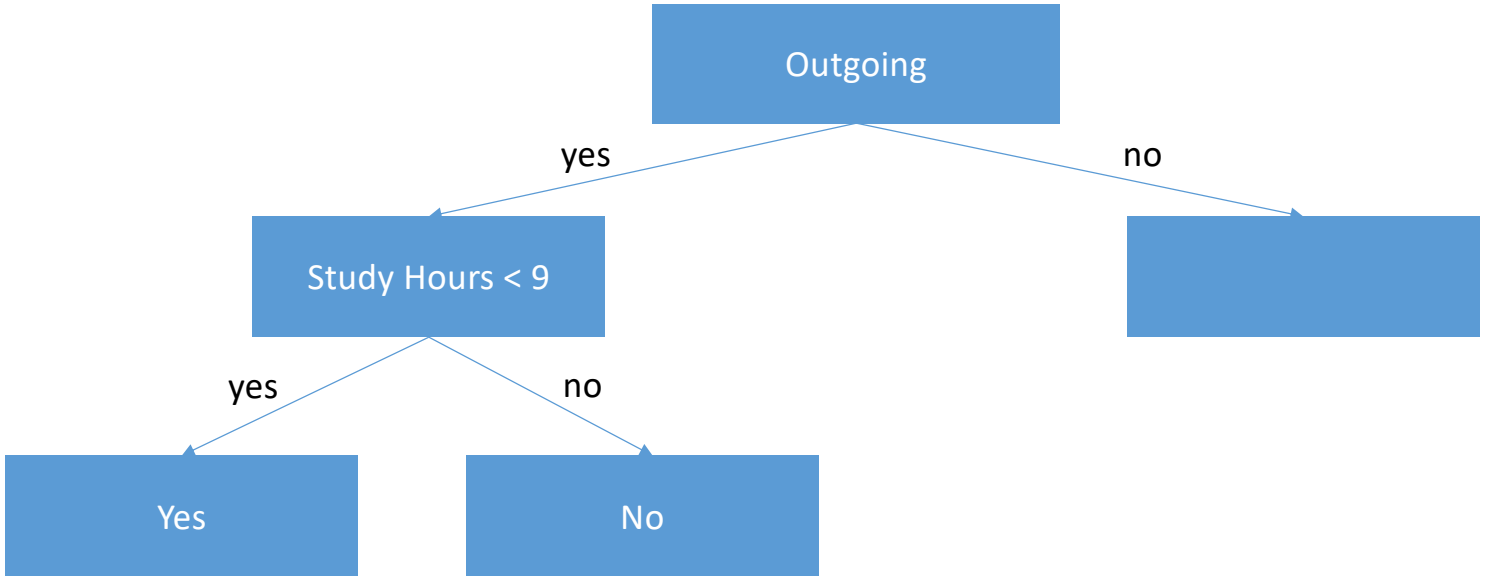
Root

Illustrative Example for Hunt's Algorithm

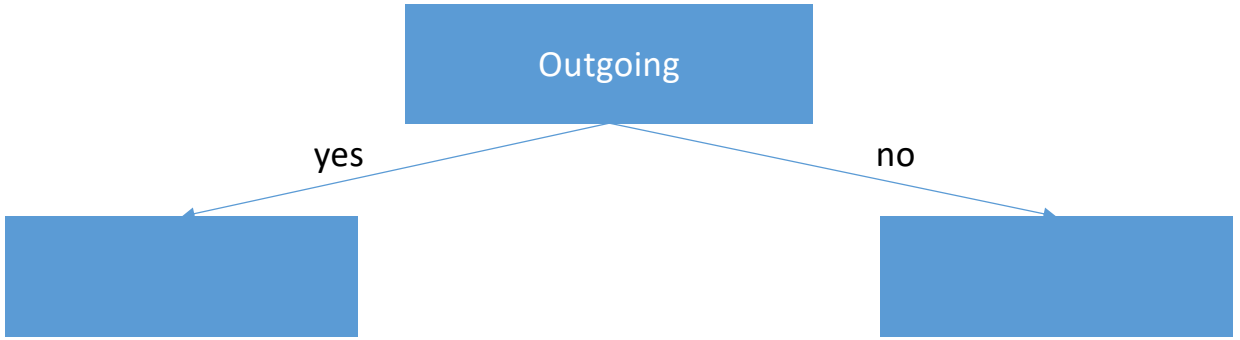


ID	Study Hours	Outgoing	Relationship	Age	X'mas party
2	8	Yes	Yes	19	Yes
5	9	Yes	Yes	18	No

Illustrative Example for Hunt's Algorithm

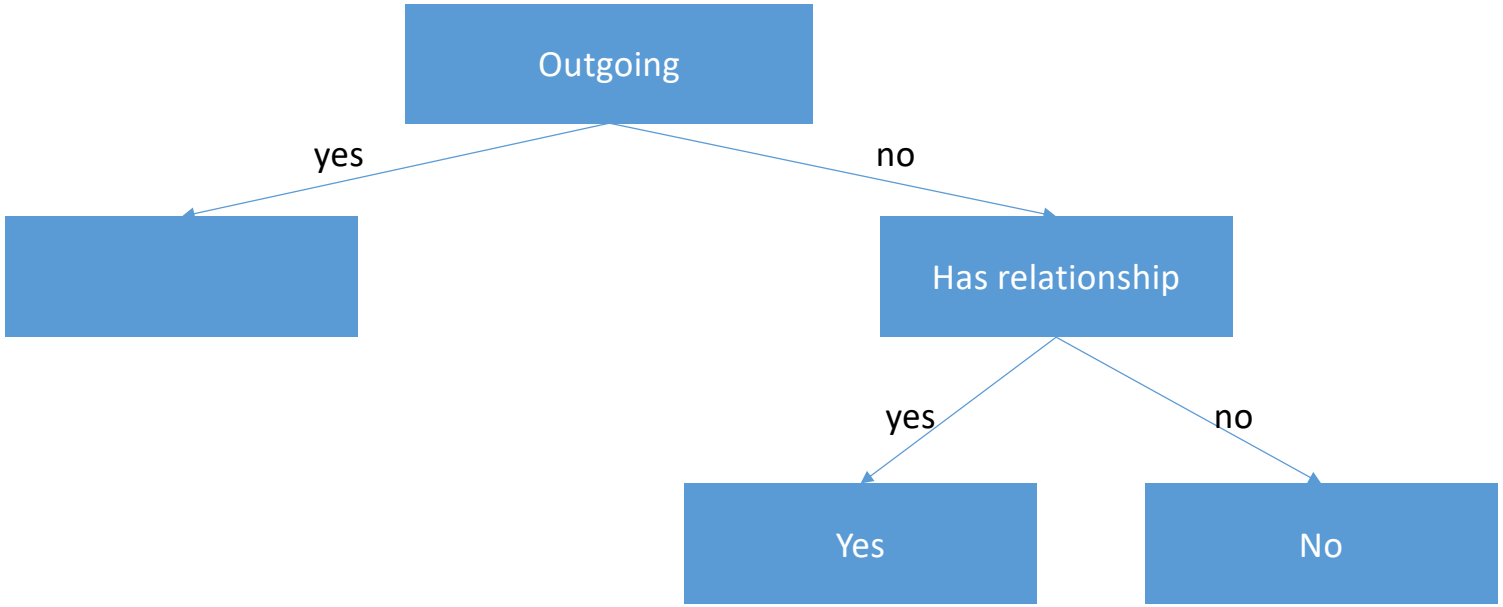


Illustrative Example for Hunt's Algorithm

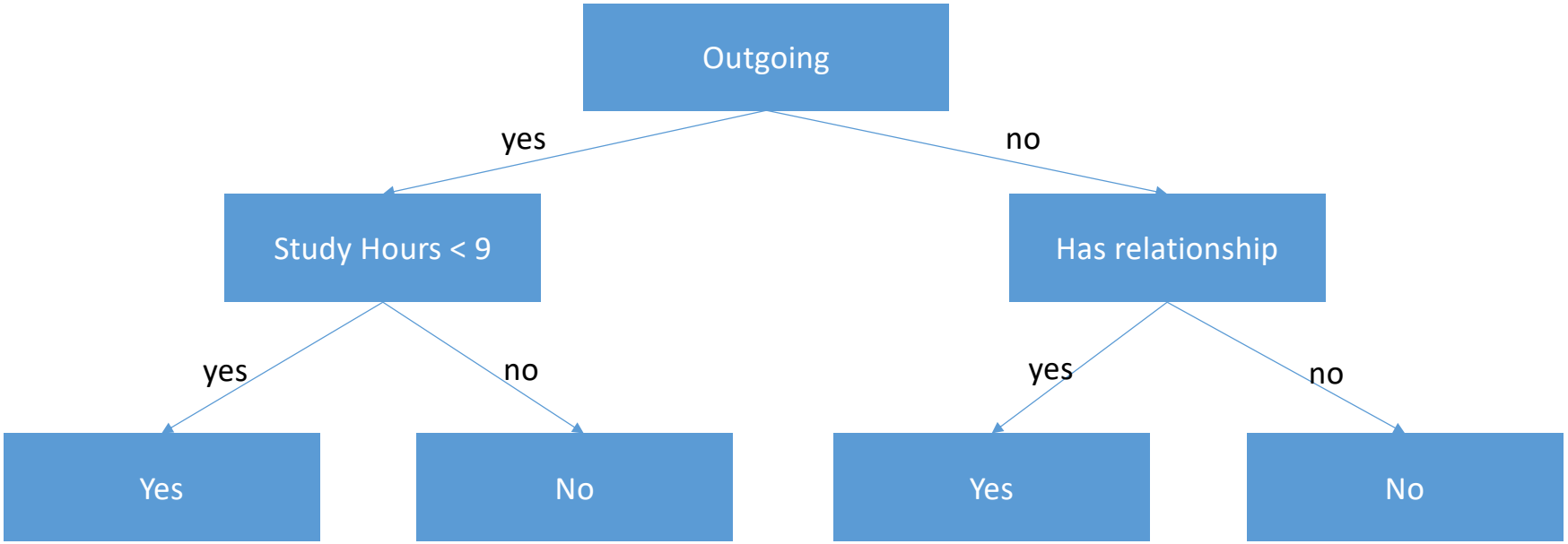


ID	Study Hours	Outgoing	Relationship	Age	X'mas party
1	3	No	No	18	No
3	4	No	Yes	20	Yes
4	10	No	No	22	No

Illustrative Example for Hunt's Algorithm

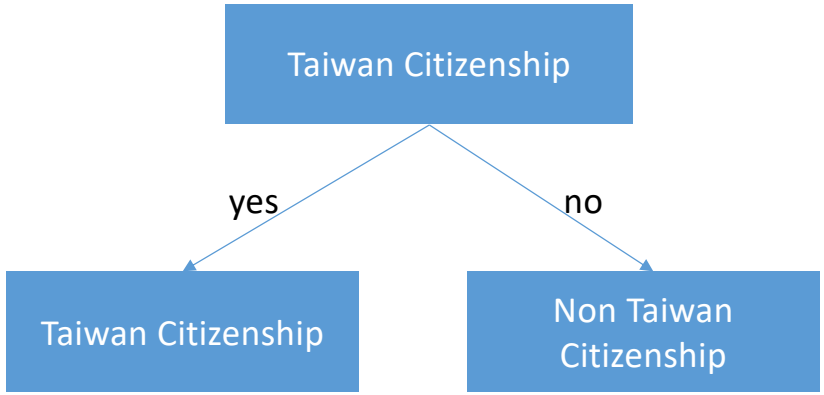


Illustrative Example for Hunt's Algorithm

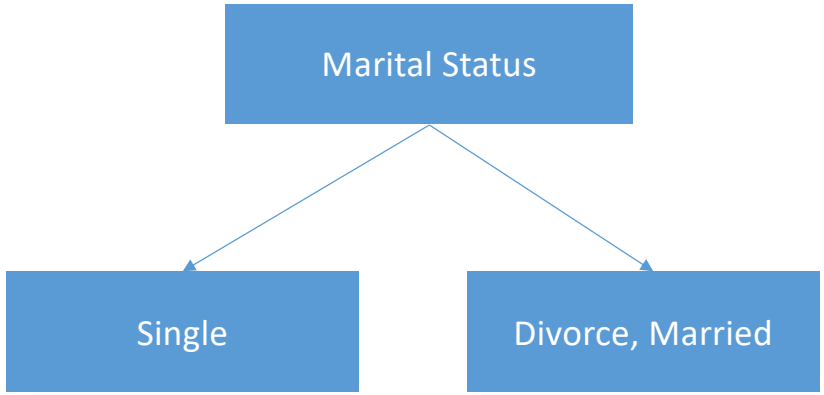


- Design issues of decision trees
  - how should the training examples be split?
    - each step must select an attribute test condition
      - a method for specifying the test condition of a attribute
      - an objective measurement for evaluating the goodness of the attribute test condition
  - how should the splitting procedure stop?
    - a stopping criterion for the tree to stop growing
    - a possible stopping strategy is when all training examples in nodes belong to the same class

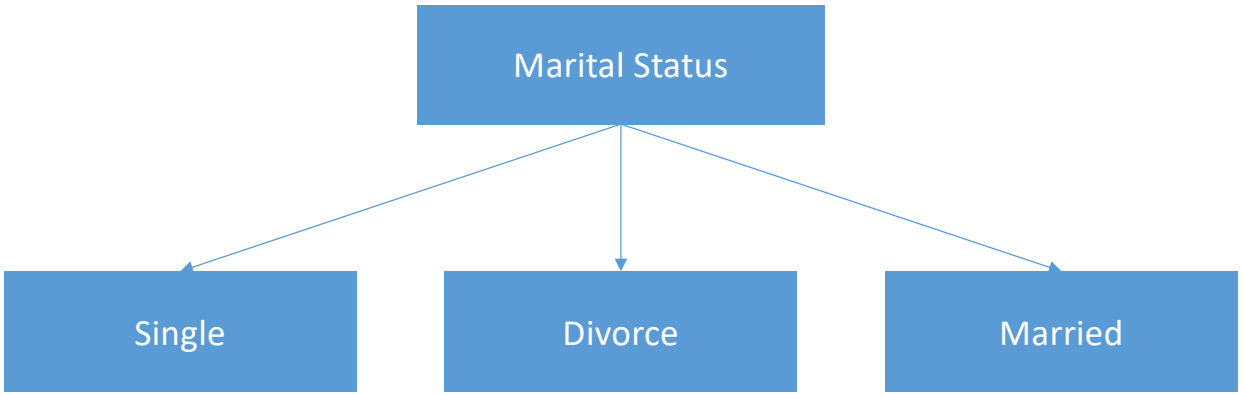
- How to express the attribute test conditions
  - binary attribute: create two potential outcomes



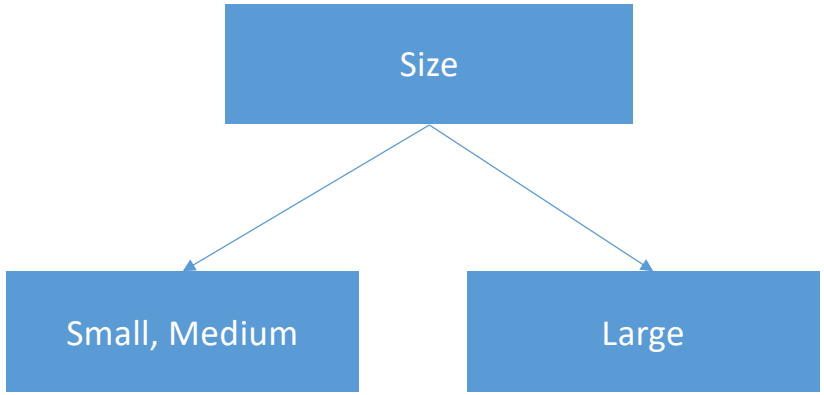
- How to express the attribute test conditions
  - nominal or ordinal or continuous attribute:
    - binary or multi-way splits



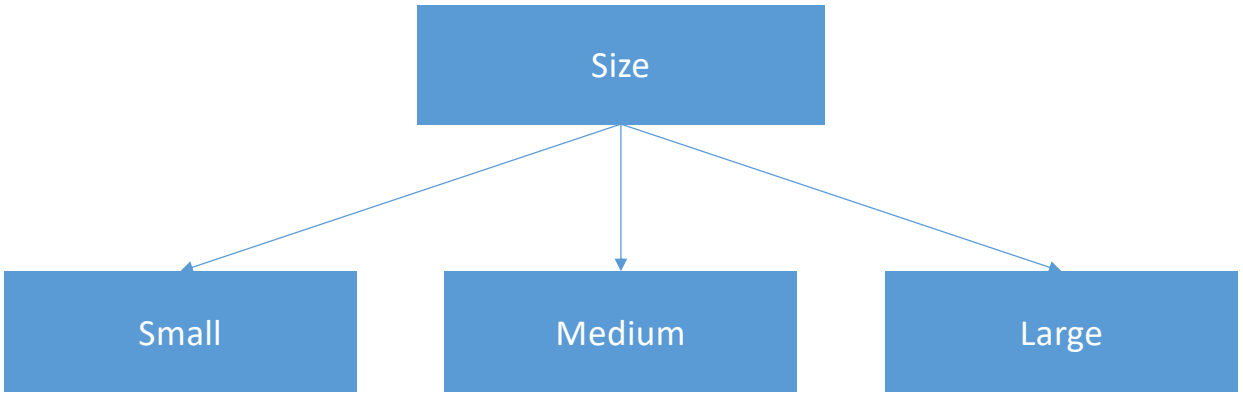
- How to express the attribute test conditions
  - nominal or ordinal or continuous attribute:
    - binary or multi-way splits



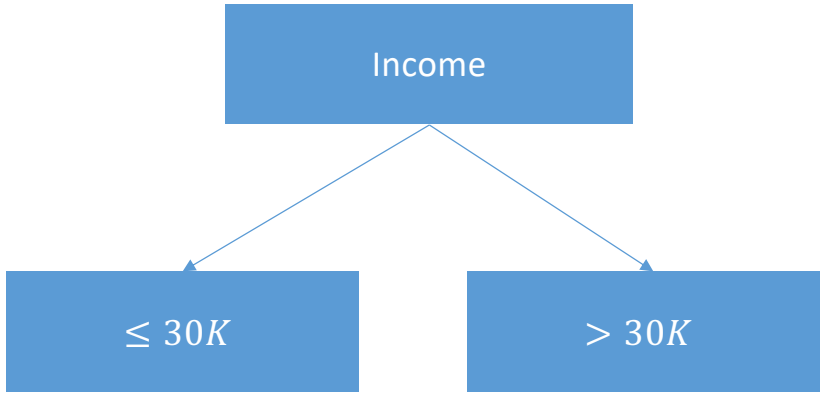
- How to express the attribute test conditions
  - nominal or ordinal or continuous attribute:
    - binary or multi-way splits



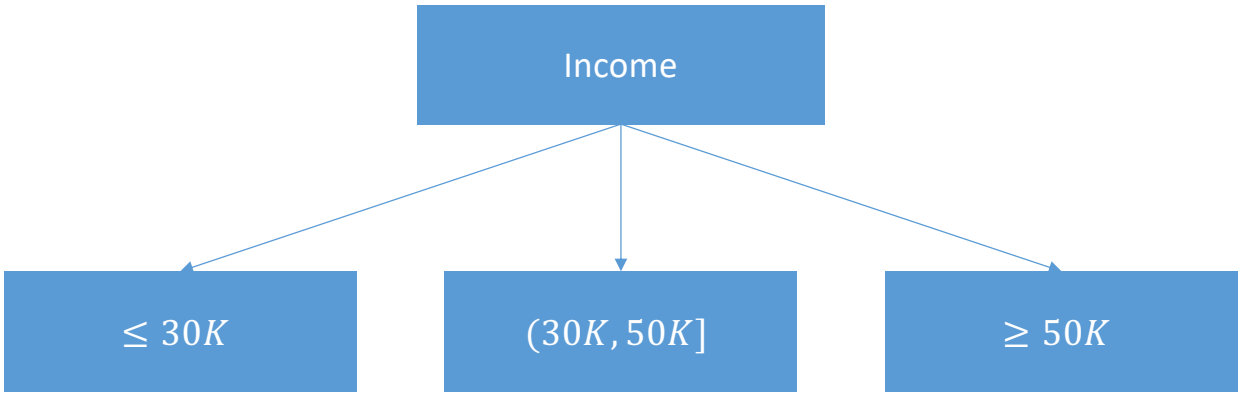
- How to express the attribute test conditions
  - nominal or ordinal or continuous attribute:
    - binary or multi-way splits



- How to express the attribute test conditions
  - nominal or ordinal or continuous attribute:
    - binary or multi-way splits

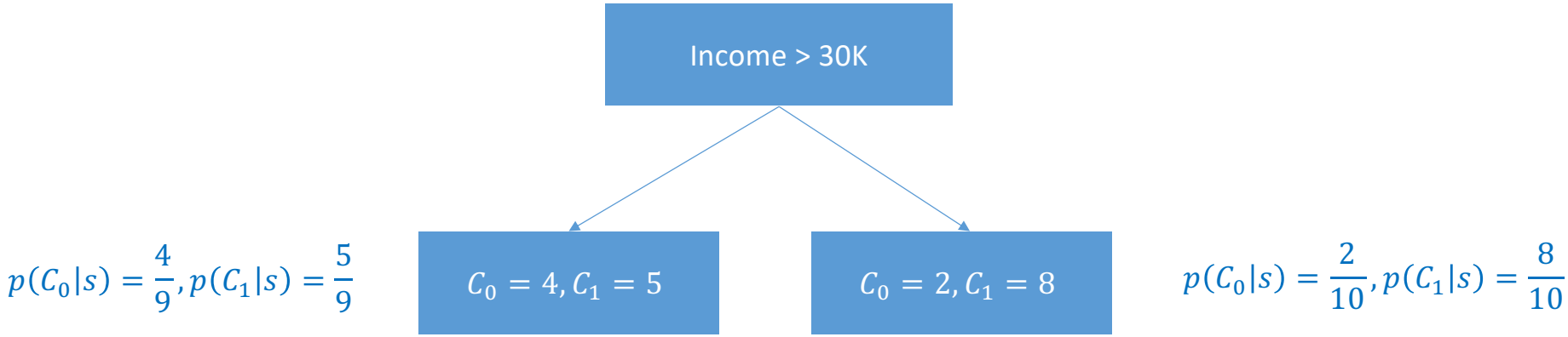


- How to express the attribute test conditions
  - nominal or ordinal or continuous attribute:
    - binary or multi-way splits



Fundamentals in Machine Learning
Topic 05
Decision Tree – Part II

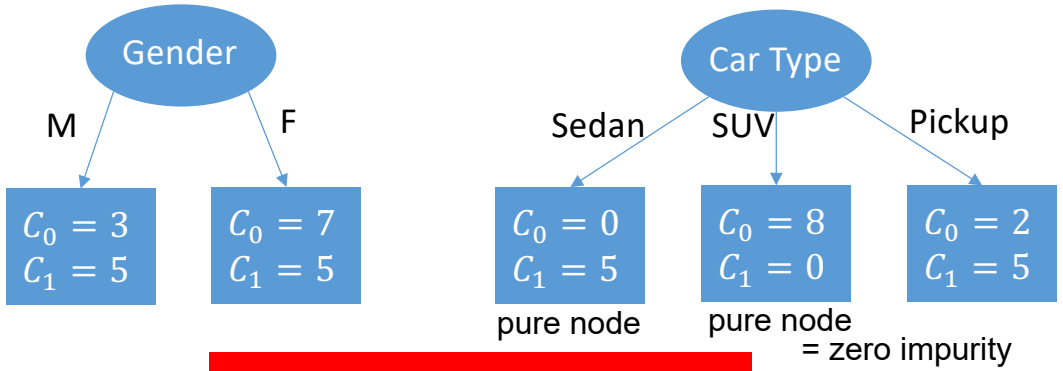
- What are the measures for selecting the best split?
- These measures normally has to do with the distribution of classes
- We first define probability of class k, where $k \in \{0,1,2,\cdots,K-1\}$ at node s as $p(C_k|s)$



Two classes: C_0 and C_1

Number of total examples: 10

Before splitting, $p(C_0) = \frac{1}{2}$ and $p(C_1) = \frac{1}{2}$



Which split is better?

- Splitting on the car type results in purer partition
- The measures for selecting the best split are often based on the degree of the impurity of the child nodes
  - the node with class distribution (0,1) has zero impurity
  - the node with class distribution (0.5, 0.5) has highest impurity
  - in the case of binary classification, if p is the probability of class 0 at a node, we will need a measure which has the peak at $p = 0.5$ and has lowest value when either $p = 0$ or $p = 1$

- To reflect the impurity of node s, three common measures are
  - Entropy: ID3, C4.5, C5

level of randomness => harder to predict

= amount of information
越亂 => 越多資訊

$$-\sum_{i=0}^{K-1} p(C_i|s)\log_2 p(C_i|s)$$

  - Gini index: CART

$$\sum_{i=0}^{K-1} p(C_i|s)\big(1-p(C_i|s)\big) = 1-\sum_{i=0}^{K-1}\big(p(C_i|s)\big)^2$$

  - Classification error:

$$1-\max p(C_i|s)$$

$C_0 = 0$ $C_1 = 10$	Entropy	0
	Gini	0
	Classification error	0
$C_0 = 2$ $C_1 = 8$	Entropy	0.8113
	Gini	0.32
	Classification error	0.2
$C_0 = 5$ $C_1 = 5$	Entropy	1
	Gini	0.5
	Classification error	0.5

- So far, we only talk about how to calculate the degree of impurity of a node
- How to split a node in the best way is still unanswered
  - the best splitting results in the largest decrease in the degree of impurity before and after splitting among all possible ways of splitting
  - Two commonly adopted measures
    - information gain <= possible features
    - Gini index after splitting

Before splitting:
total number of example is N
the number of example in C_i is n_i

After splitting:
S nodes are created
total number of example in node s is N_s
the number of example in C_i at node s is n_{si}
the probability of example in C_i at node s is $p(C_i|s)$

parent entropy

the entropy after splitting given a feature

$$\text{Information Gain} = \left[-\sum_{i=0}^{K-1}\left(\frac{n_i}{N}\right)\log_2\left(\frac{n_i}{N}\right)\right] - \sum_{s=1}^{S}\left(\frac{N_s}{N}\right)\left[-\sum_{i=0}^{K-1}\left(\frac{n_{si}}{N_s}\right)\log_2\left(\frac{n_{si}}{N_s}\right)\right]$$

node's weight

entropy in a node

$$\text{Gini index after splitting} = \sum_{s=1}^{S}\left(\frac{N_s}{N}\right)\left(1 - \sum_{i=0}^{K-1}\left(\frac{n_{si}}{N_s}\right)^2\right)$$

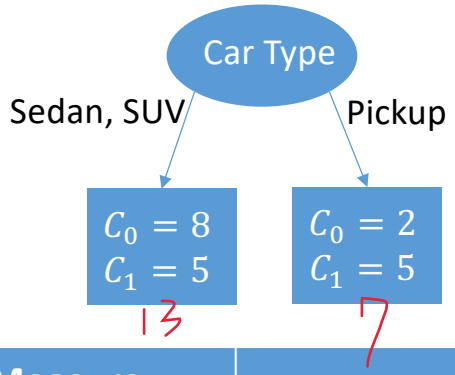
gini index within a node

$C_0 = 10$

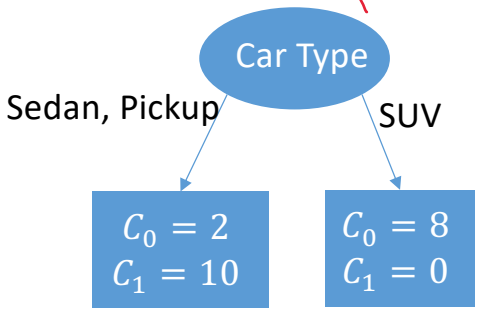
$C_1 = 10$

$-\left[\frac{10}{20}\log\frac{10}{21} + \frac{10}{20}\log\frac{10}{21}\right] = 1$

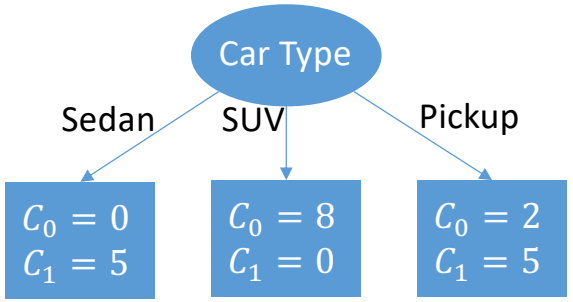
$\Rightarrow \frac{13}{20}\cdot\left[-\left(\frac{8}{13}\log \cdots\right.\right.$



Measure	
Information gain	0.0731
Gini	0.6005

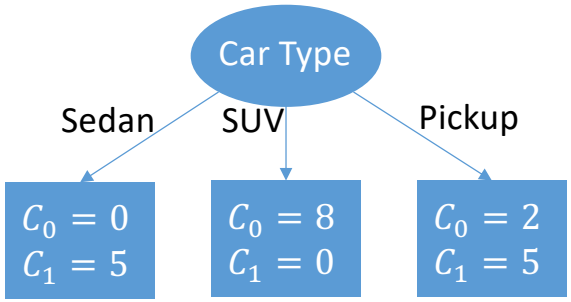


Measure	
Information gain	0.6100
Gini	0.1667

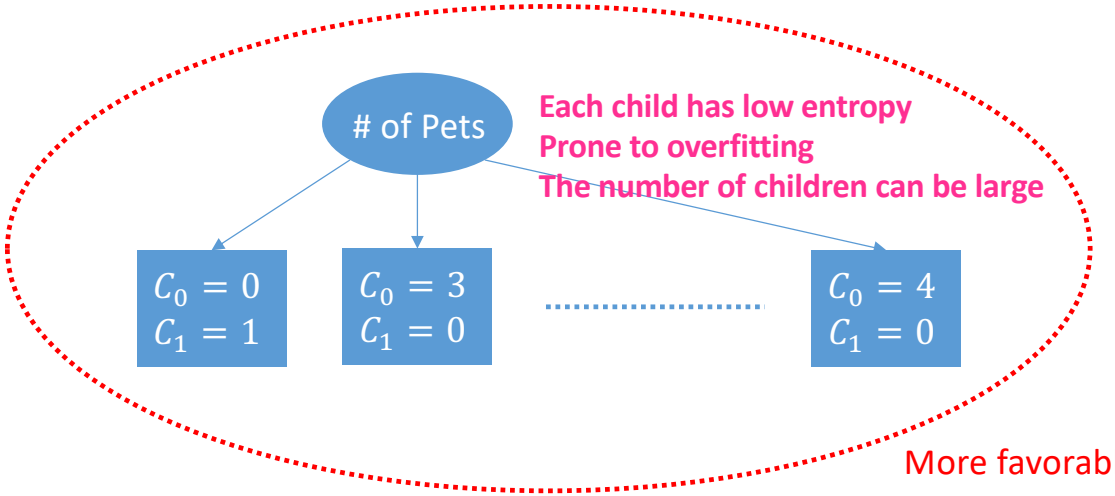


Measure	
Information gain	0.6979
Gini	0.1429

- Gain ratio
  - the impurity measures such as entropy and Gini index tend to favor attribute that have a large number of distinct values



Less favorable



More favorable

- Two ways to deal with this:
  - limit the test condition to binary split only
    - CART
  - modify the splitting condition to take into account the number of outcomes
    - in C4.5, gain ratio is adopted as the criterion to determine how good a split is
    - gain ratio is defined as

$$\frac{\text{Information Gain}}{\text{Intrinsic Information}}$$

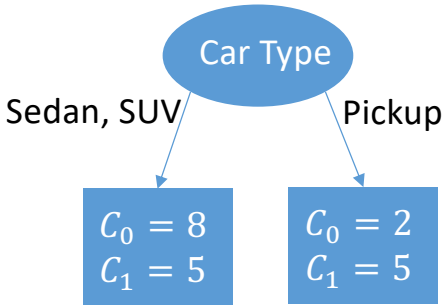
Before splitting:
total number of example is N
the number of example in C_i is n_i

After splitting:
S nodes are created
total number of example in node s is N_s
the number of example in C_i at node s is n_{si}
the probability of example in C_i at node s is $p(C_i|s)$

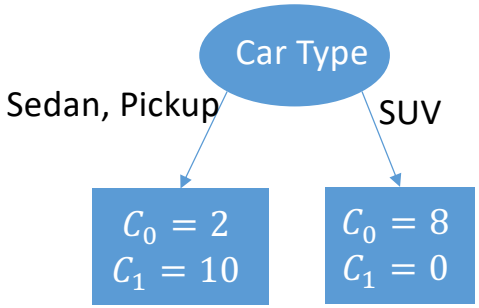
$$\text{Information Gain} = \left[-\sum_{i=0}^{K-1}\left(\frac{n_i}{N}\right)\log_2\left(\frac{n_i}{N}\right)\right] - \sum_{s=1}^{S}\left(\frac{N_s}{N}\right)\left[-\sum_{i=0}^{K-1}\left(\frac{n_{si}}{N_s}\right)\log_2\left(\frac{n_{si}}{N_s}\right)\right]$$

recall

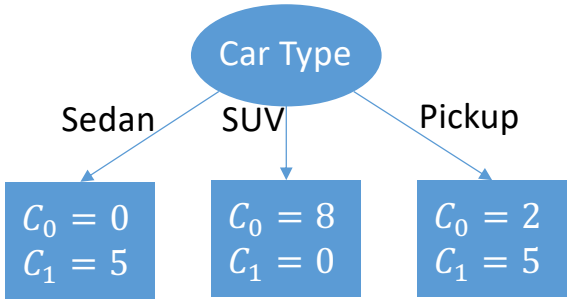
$$\text{Instrinsic Informtion} = -\sum_{s=1}^{S}\left(\frac{N_s}{N}\right)\log_2\left(\frac{N_s}{N}\right)$$



Measure	
Information gain	0.0731
Gain ratio	0.0783



Measure	
Information gain	0.6100
Gain ratio	0.6282



Measure	
Information gain	0.6979
Gain ratio	0.5748

Algorithm for Decision Tree Induction

- Let $\mathcal{D}$ be the training examples and $\mathcal{A}$ be the set of attributes

TreeGrowing($\mathcal{D}$, $\mathcal{A}$)

1. if stop($\mathcal{D}$, $\mathcal{A}$) = True, then
2. leaf = create_node()
3. leaf.class = classify($\mathcal{D}$)
4. return leaf
5. else
6. root = create_node()
7. root.test = best_split($\mathcal{D}$, $\mathcal{A}$)
8. let $\mathcal{S} = \{s | s \text{ is a possible outcome of root.test}\}$
9. for each s in $\mathcal{S}$ do
10. $\mathcal{D}_s = \{d | \text{root.test}(d) = s \text{ and } d \text{ in } \mathcal{D}\}$
11. child = TreeGrowing($\mathcal{D}_s$, $\mathcal{A}$)
12. end for
13. end if
14. return root

Q&A