Welcome to:

Machine Learning

Welcome to:

Unit – 4: Introduction to Classification & Classification Algorithms

Outline

- General Approach to Classification
- k-Nearest Neighbor Algorithm
- Decision Trees
- Naive Bayesian Classifier
- Ensemble Methods
- Advanced Classification Methods
- Classification Model Evaluation and Selection:
- Misclassification Cost Adjustment to Reflect Real-World Concerns
- Decision Cost/Benefit Analysis

Objectives

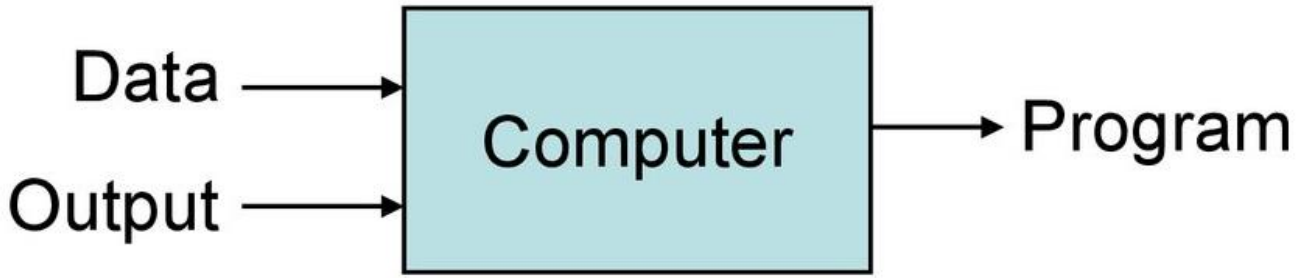
- In this unit, we planned to made the students to understand about:
  - General Concepts on Classification
  - Applications of classification algorithms
  - Classification techniques such as k-Nearest Neighbor Algorithm, Decision Trees, Naive Bayesian Classifier, Ensemble Methods namely bagging and boosting, and advanced classification methods namely support vector machines, artificial neural networks.
  - Metrics for evaluation of classification methods

Preamble – Machine Learning

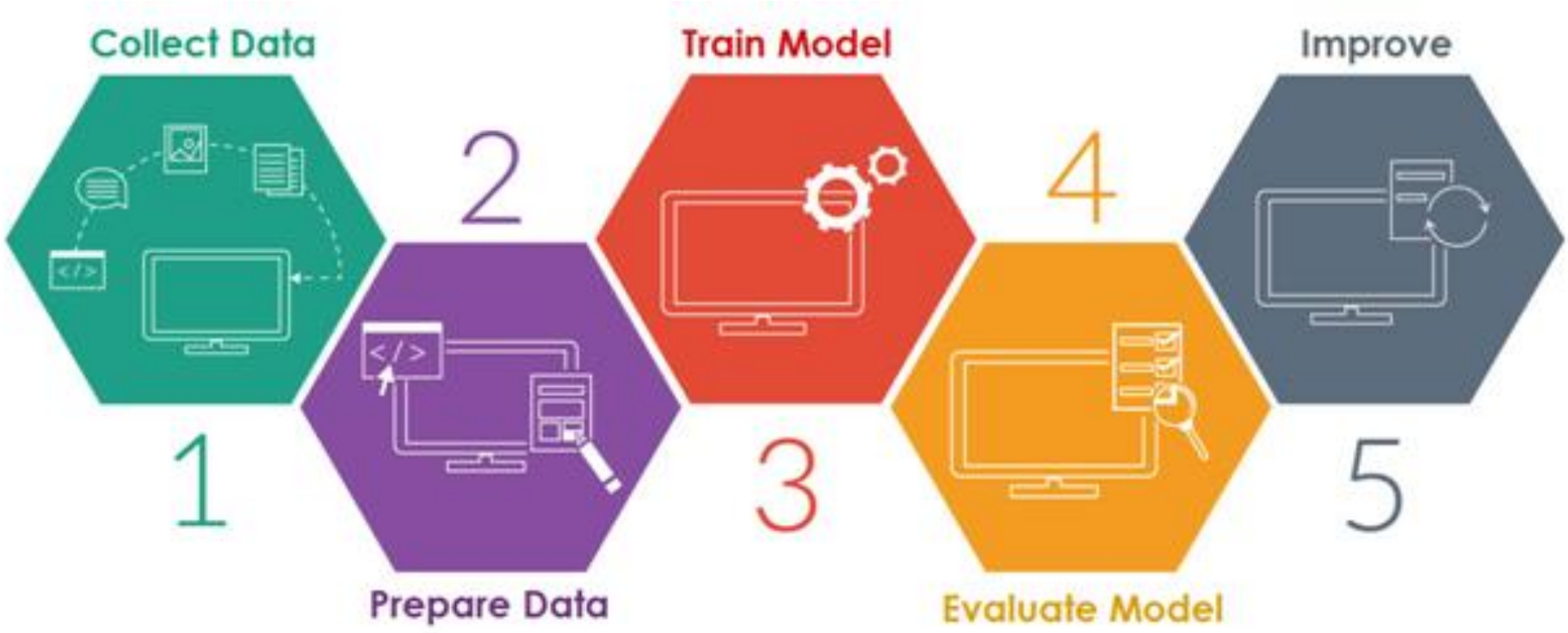
What is Machine Learning?

- A major focus of machine learning research is to automatically learn to recognize complex patterns and make intelligent decisions based on data.

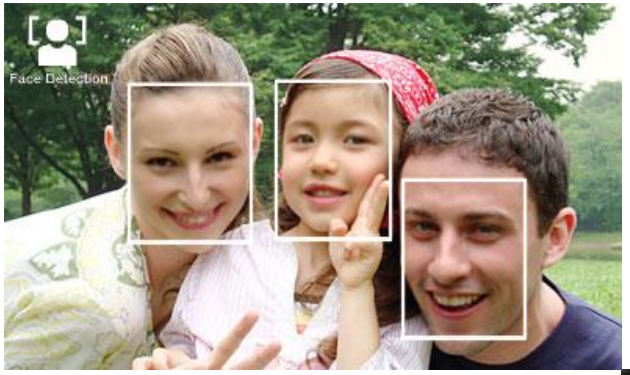
- The goal of machine learning is to build computer systems that can adapt and learn from their experience - Tom Dietterich

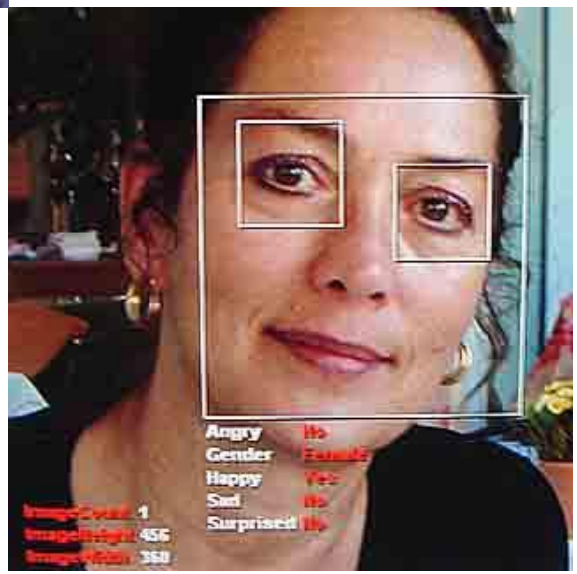


Phases in Machine Learning

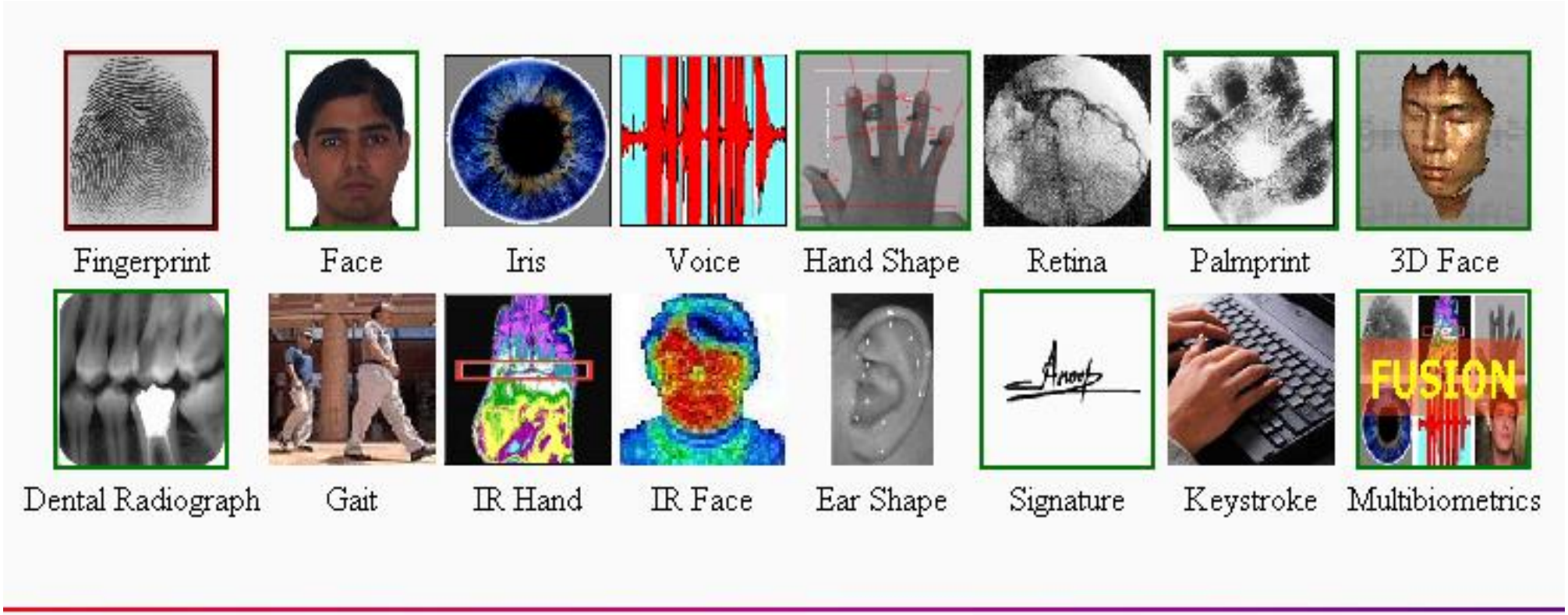


To Classify Faces and Expressions

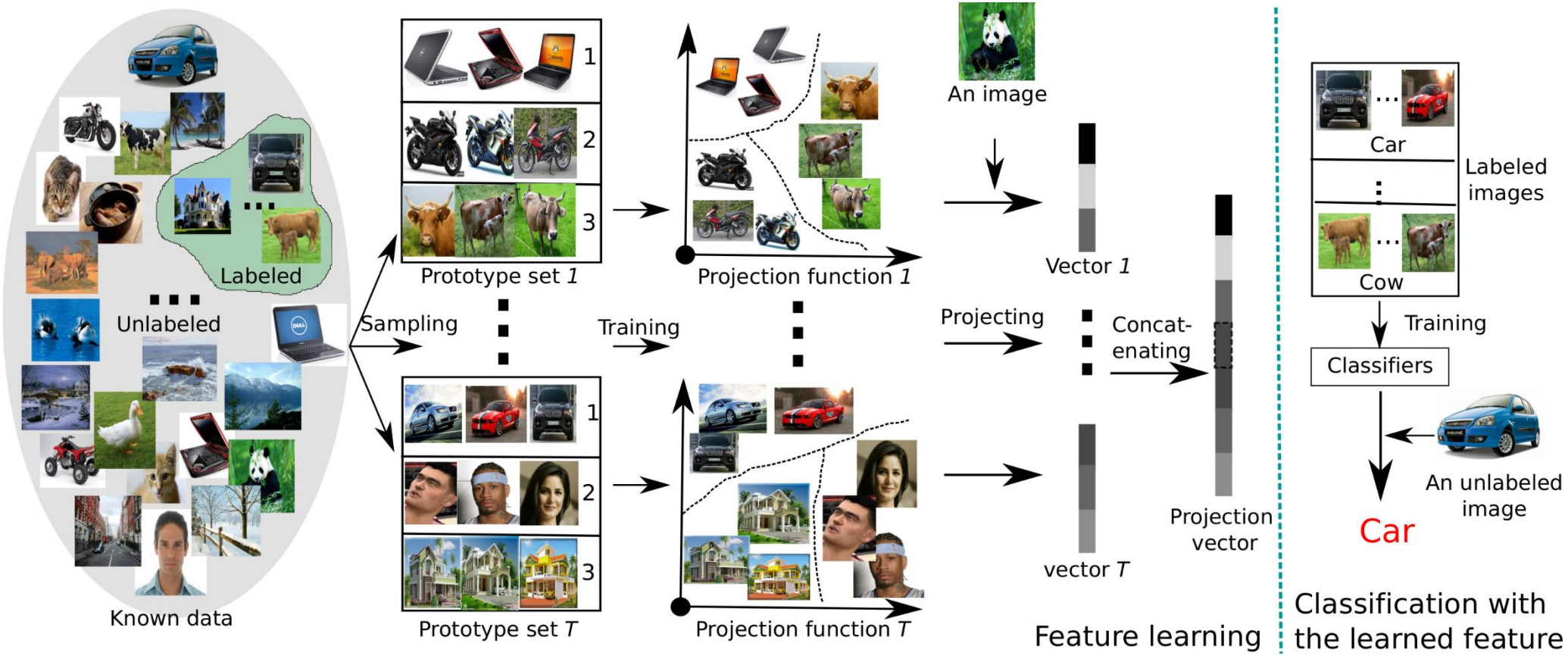




Biometrics


Fingerprint
Face
Iris
Voice
Hand Shape
Retina
Palmprint
3D Face
Dental Radiograph
Gait
IR Hand
IR Face
Ear Shape
Signature
Keystroke
Multibiometrics
FUSION

Introduction



Source: *Dengxin Dai , and Luc Van Gool*

Unsupervised High-level Feature Learning by Ensemble Projection for Semi-supervised Image Classification and Image Clustering.

Problems in classifying data

- Often high dimension of data.
- Hard to put up simple rules.
- Amount of data.
- Need automated ways to deal with the data.

Use computers – data processing, statistical analysis, try to learn patterns from the data (Machine Learning Methods)

Example: Document classification

- A Person talked about ways to make rules that classify documents. Examples of companies that have such systems are:
  - LexisNexis
  - Verity
  - Smart-Logic
  - Interwoven
- Machine Learning is another way of getting computers to classify documents.
- Machine learning is normally not rule based. Instead, it is normally statistically based.
- Definition of Machine Learning from dictionary.com
  - “The ability of a machine to improve its performance based on previous results.”
- Machine learning document classification is “the ability of a machine to improve its document classification performance based on previous results of document classification”.

You as an ML Classifier

- **Topic 1 words:**

  baseball, owners, sports, selig, ball, bill, indians, isringhausen, mets, minors, players, specter, stadium, power, send, new, bud, comes, compassion, game, headaches, lite, nfl, powerful, strawberry, urges, home, ambassadors, building, calendar, commish, costs, day, dolan, drive, hits, league, little, match, payments, pitch, play, player, red, stadiums, umpire, wife, youth, field, leads

- **Topic 2 words:**

  merger, business, bank, buy, announces, new, acquisition, finance, companies, com, company, disclosure, emm, news, us, acquire, chemical, inc, results, shares, takeover, corporation, european, financial, investment, market, quarter, two, acquires, bancorp, bids, communications, first, mln, purchase, record, stake, west, sale, bid, bn, brief, briefs, capital, control, europe, inculab

Use the previous slide's topics & related words to classify the following titles

- CYBEX-Trotter merger creates fitness equipment powerhouse
- WSU RECRUIT CHOOSES BASEBALL INSTEAD OF FOOTBALL
- FCC chief says merger may help pre-empt Internet regulation
- Vision of baseball stadium growing
- Regency Realty Corporation Completes Acquisition Of Branch properties
- Red Sox to punish All-Star scalpers
- Canadian high-tech firm poised to make $415-million acquisition
- Futures-selling hits the Footsie for six
- A'S NOTEBOOK; Another Young Arm Called Up
- All-American SportPark Reaches Agreement for Release of Corporate Guarantees

Titles & Their Classifications

- (2) CYBEX-Trotter merger creates fitness equipment powerhouse
- (1) WSU RECRUIT CHOOSES BASEBALL INSTEAD OF FOOTBALL
- (2) FCC chief says merger may help pre-empt Internet regulation
- (1) Vision of baseball stadium growing
- (2) Regency Realty Corporation Completes Acquisition Of Branch properties
- (1) Red Sox to punish All-Star scalpers
- (2) Canadian high-tech firm poised to make $415-million acquisition
- (2) Futures-selling hits the Footsie for six
- (1) A'S NOTEBOOK; Another Young Arm Called Up
- (1) All-American SportPark Reaches Agreement for Release of Corporate Guarantees

Uses of ML classifiers

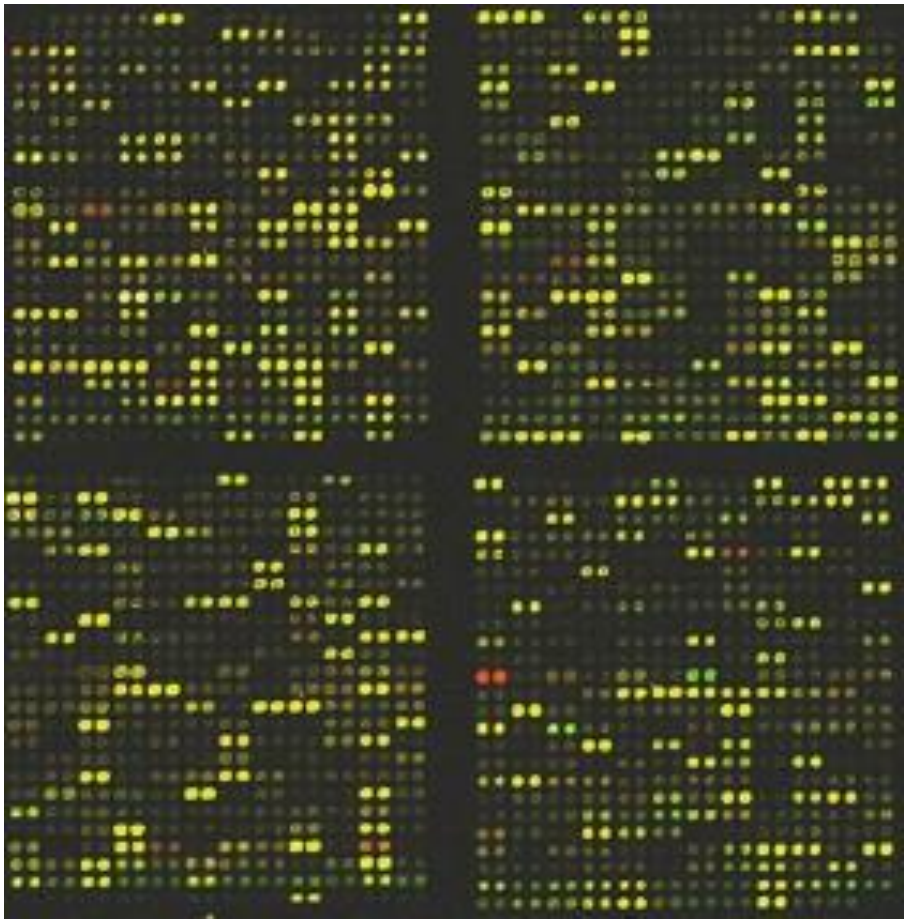
- Automatically classify documents.
- Find important information in a document.
  - For example, rules of law in a case law document, or the facts of the case.
- text categorization (e.g., spam filtering)
- fraud detection
- optical character recognition
- machine vision (e.g., face detection)
- natural-language processing
  - (e.g., spoken language understanding)
- market segmentation
  - (e.g.: predict if customer will respond to promotion)
- bioinformatics
  - (e.g., classify proteins according to their function)

Some Real life applications

- Systems Biology – Gene expression microarray data:
- Text categorization: spam detection
- Face detection: Signature recognition: Customer discovery
- Medicine: Predict if a patient has heart ischemia by a spectral analysis of his/her ECG.

Microarray data

Separate malignant from healthy tissues based on the mRNA expression profile of the tissue.



Text Categorization (multi category)

Categorize text documents into predefined categories . For example, categorize news into ‘sports’, ‘politics’, ‘science’, etc.

Soft tissue found in T-rex fossil
Find may reveal details about cells and blood vessels
of di
Thur
WAS
stud
bone
year
vessels a

Health may be concern when giving kids cell phones
Wednesday, March 23, 2005 Posted: 11:14 AM EST
SEATTLE, Washington (AP) -- Parents should think
twic
for a
long

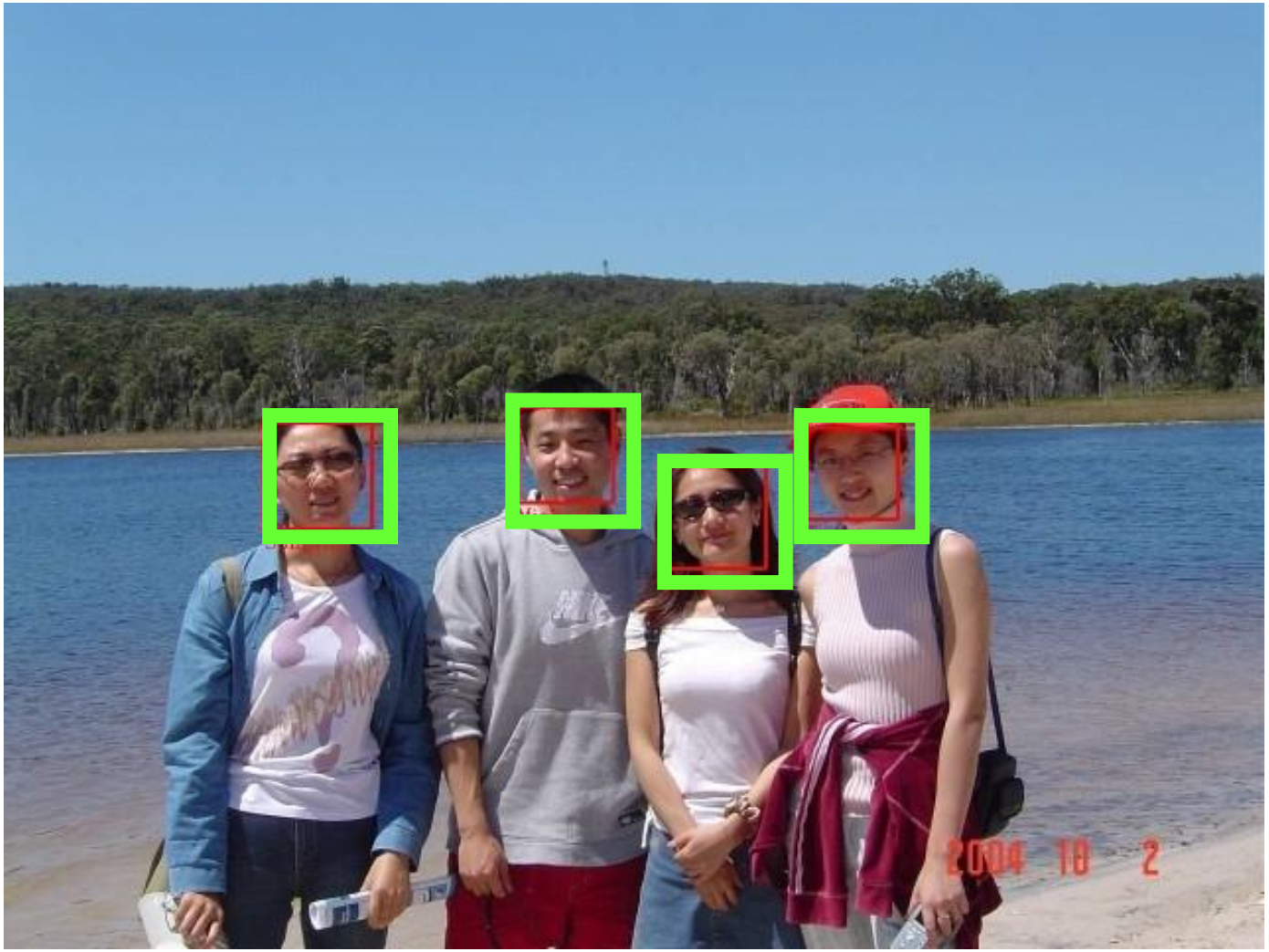
Wall Street gears up for jobs
Saturday, March 26, 2005: 11:41 AM EST

NEW
2005
-- de

Probe finds atmosphere on Saturn moon
Thursday, March 17, 2005 Posted: 11:17 AM EST
LOS ANGELES, California (Reuters) -- The space probe Cassini discovered a significant atmosphere around Saturn's moon Enceladus during two recent passes close by, the Jet Propulsion Laboratory said on Wednesday

Face detection

- Discriminating human faces from non-faces

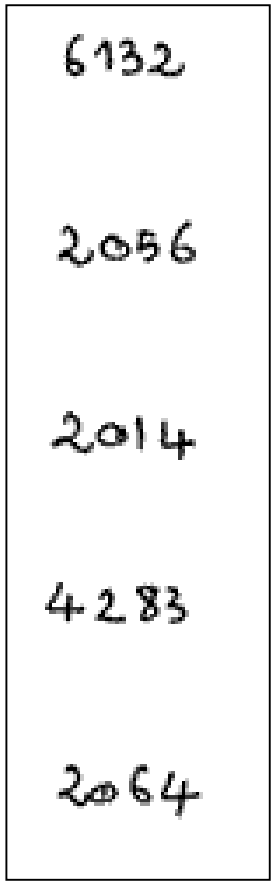


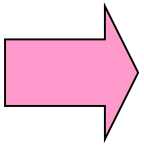
Signature recognition

- Recognize signatures by structural similarities which are difficult to quantify.
- does a signature belongs to a specific person, say Amir, or not.

Character recognition (multi category)

- Identify handwritten characters: classify each image of character into one of 10 categories ‘0’, ‘1’, ‘2’ …





6132

2056

2014

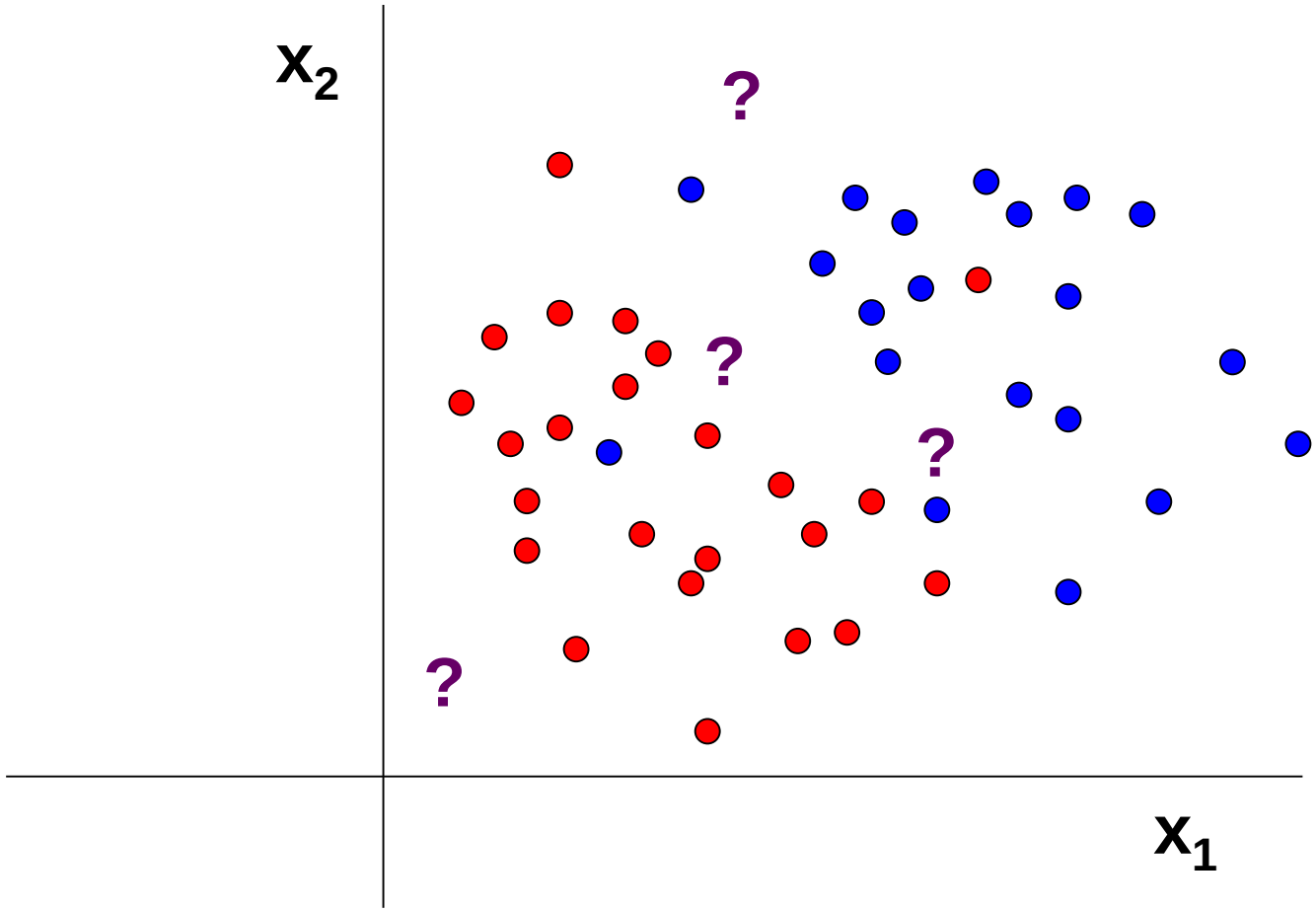
4283

2064

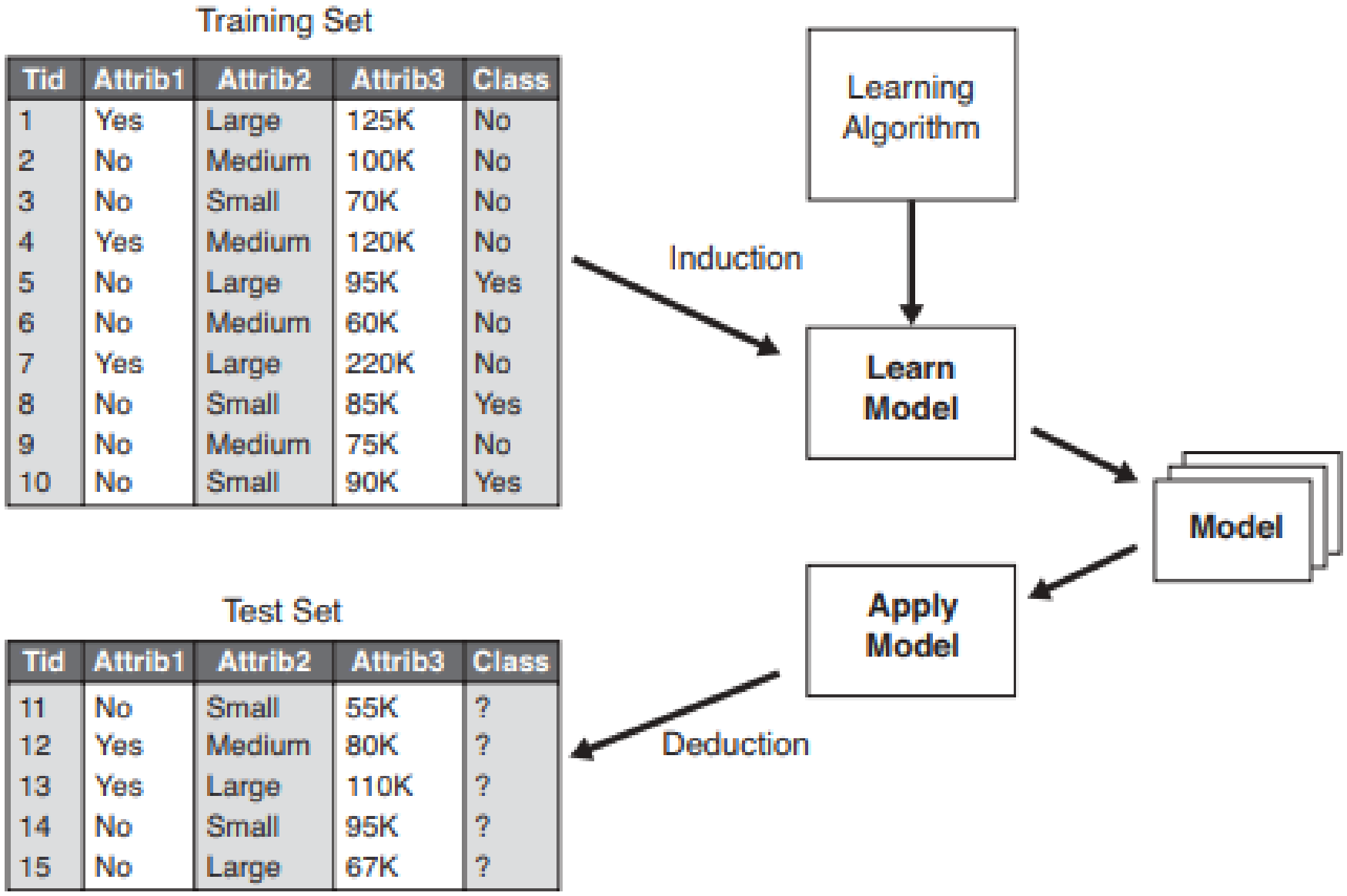
Advantages & Disadvantages

- Advantage over classification by humans: Once the system is trained, classification is done automatically with no or little human intervention – saving human resources.
- Advantage over classification by humans: Consistent classification.
- Advantage over rule based classification: Human resources are not needed to make rules.
- Disadvantage: Not always obvious why it classified a document in a certain way and not obvious how to keep it from doing the same type of classification in the future (i.e. don't know how to modify it.)
- Disadvantage: Human resources must be used to manually classify documents for the training set. Furthermore, the number and type of document that should be in the training set isn't straightforward.

Classification problem



General approach



Classification algorithms

- Fisher linear discriminant
- KNN
- Decision tree
- Neural networks
- SVM
- Naïve Bayes
- Adaboost etc.

Each one has its properties wrt bias, speed, accuracy, transparency…

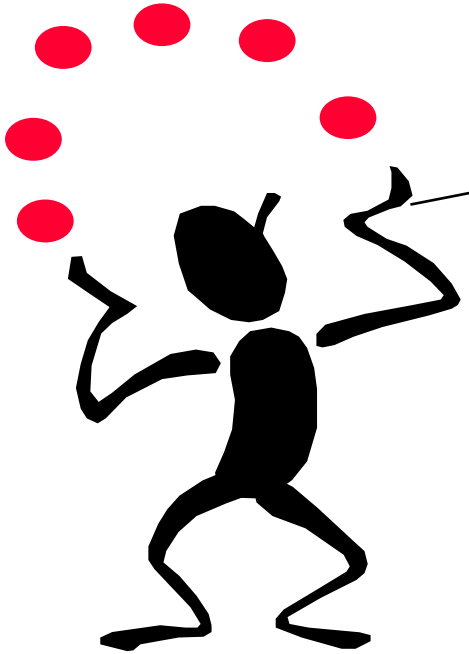
Classification algorithms

- **Naïve Bayes** – This method computes the probability that a document is about a particular topic, T, using a) the words of the document to be classified and b) the estimated probability of each of these words as they appeared in the set of training documents for the topic, T – like the example previously given.
- **Neural networks** – During training, a neural network looks at the patterns of features (e.g. words, phrases, or N-grams) that appear in a document of the training set and attempts to produce classifications for the document. If its attempt doesn't match the set of desired classifications, it adjusts the weights of the connections between neurons. It repeats this process until the attempted classifications match the desired classifications.
- **Instance based** – Saves documents of the training set and compares new documents to be classified with the saved documents. The document to be classified gets tagged with the highest scoring classifications. One way to do this is to implement a search engine using the documents of the training set as the document collection. A document to be classified becomes a query/search. A classification, C, is picked if a large number of its training set documents are at the top of the returned answer set.

Characteristics of Classification Algorithms

- Primary goal: highly accurate predictions on test data
- Goal is not to uncover underlying “truth”
- Methods should be general purpose, fully automatic and “off-the-shelf”
- However, in practice, incorporation of prior, human knowledge is crucial
- Rich interplay between theory and practice
- Emphasis on methods that can handle large datasets

Instance based learning



Its very similar to a Desktop!!

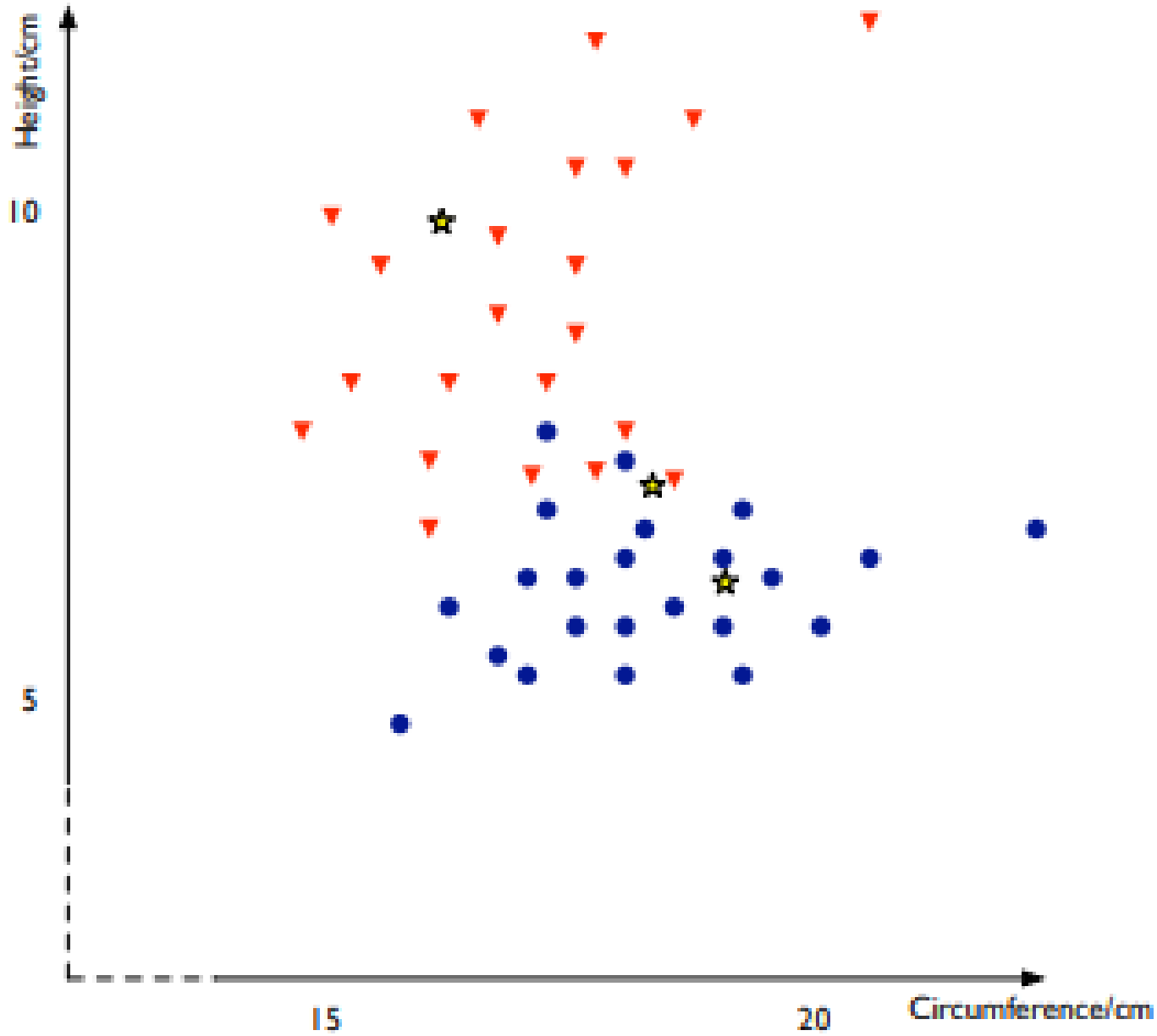
K-Nearest Neighbor – Introduction

- Most basic instance-based method
- Data are represented in a vector space
- Supervised learning
- **Feature Space:**
  - All instances correspond to points in an n-dimensional Euclidean space
  - Classification is delayed till a new instance arrives
  - Classification done by comparing feature vectors of the different points
  - Target function may be discrete or real-valued

$$\{< \vec{x}^{(1)}, f(\vec{x}^{(1)}) >, < \vec{x}^{(2)} f(\vec{x}^{(2)}) >, ..., < \vec{x}^{(n)}, f(\vec{x}^{(n)}) >\}$$

$$\vec{x} = \begin{cases} x_1 \\ x_2 \\ .. \\ .. \\ x_d \end{cases} \in \Re^d \qquad \left\| \vec{x} - \vec{y} \right\| = \sqrt{\sum_{i=1}^{d} (x_i - y_i)^2}$$

K-Nearest Neighbors – Number of neighbors



K-Nearest Neighbor algorithm

- In nearest-neighbor learning the target function may be either discrete-valued or real valued
- Learning a discrete valued function
- $f : \Re^d \rightarrow V$, V is the finite set $\{v_1,\ldots\ldots,v_n\}$
- For discrete-valued, the *k*-NN returns the most common value among the k training examples nearest to x_q.

K-Nearest Neighbor algorithm

- Training algorithm
  - For each training example <*x,f(x)*> add the example to the list
- Classification algorithm
  - Given a query instance x_q to be classified
    - Let $x_1,..,x_k$ *k* instances which are nearest to x_q

$$\hat{f}(x_q) \leftarrow \underset{v \in V}{\operatorname{argmax}} \sum_{i=1}^{k} \delta(v, f(x_i))$$

    - Where $\delta(a,b)=1$ if $a=b$, else $\delta(a,b)=0$ (Kronecker function)

KNN Example

	Food (3)	Chat (2)	Fast (2)	Price (3)	Bar (2)	BigTip
1	great	yes	yes	normal	no	yes
2	great	no	yes	normal	no	yes
3	mediocre	yes	no	high	no	no
4	great	yes	yes	normal	yes	yes

Similarity metric: Number of matching attributes (k=2)

- **New examples**:
  - Example 1 (great, no, no, normal, no) Yes

    → most similar: number 2 (1 mismatch, 4 match) → Yes

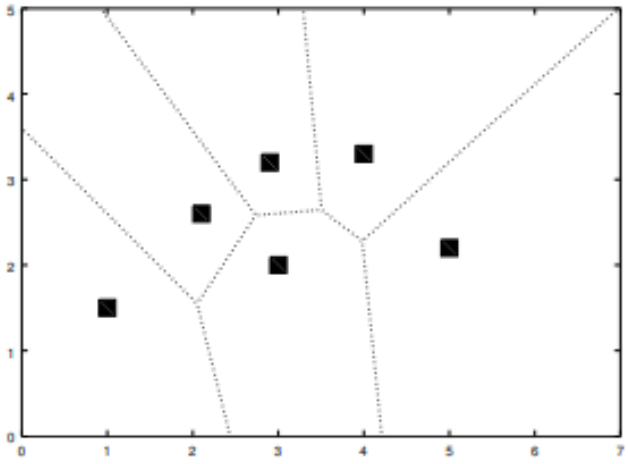
    →Second most similar example: number 1 (2 mismatch, 3 match) → Yes

  - Example 2 (mediocre, yes, no, normal, no) Yes/No

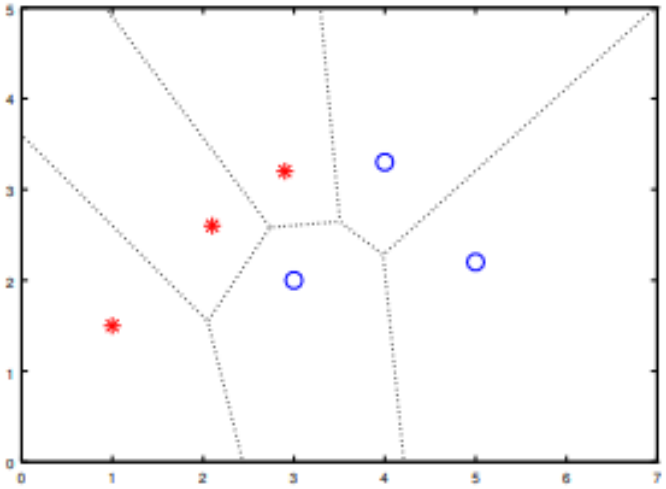
    → Most similar: number 3 (1 mismatch, 4 match) → No

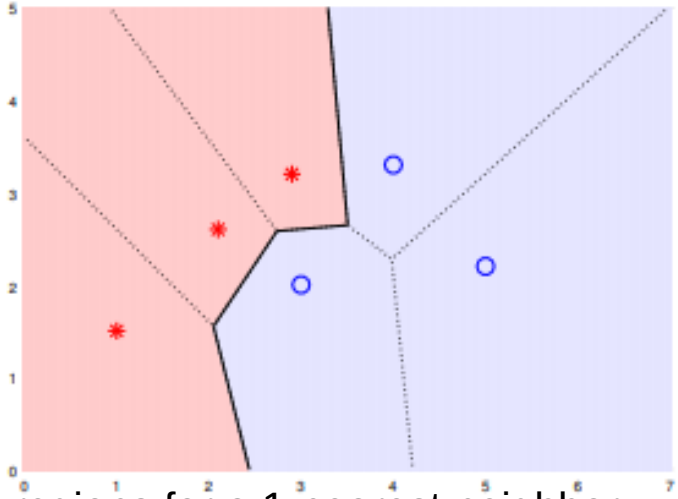
    →Second most similar example: number 1 (2 mismatch, 3 match) → Yes

Decision Boundaries: Voronoi diagram

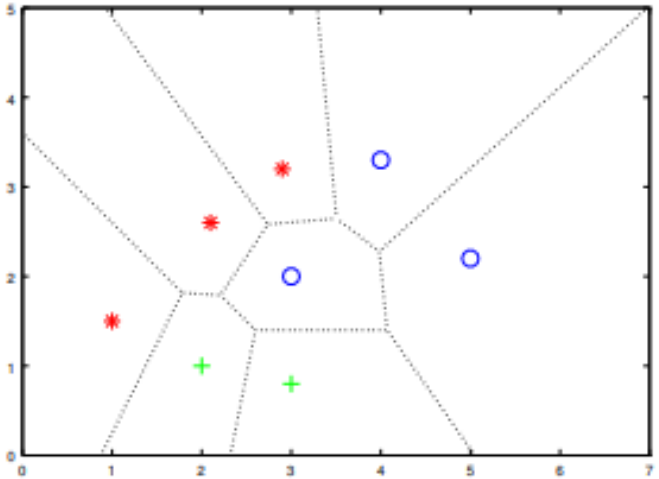


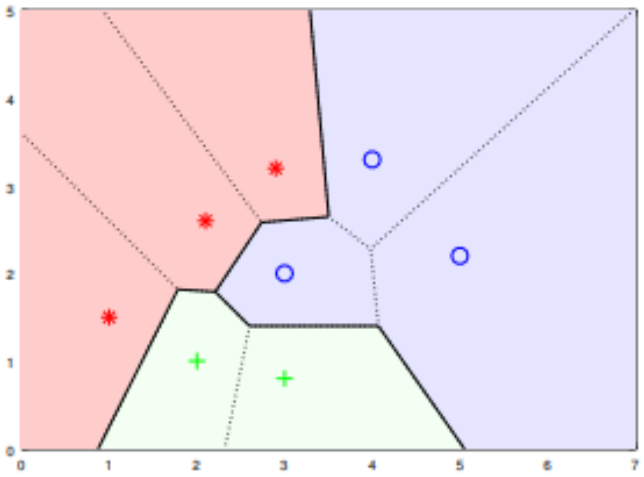
Decision boundaries by 1-NN for data points of distinct classes from each other





Decision boundary and decision regions for a 1-nearest neighbor classifier for two classes.





Decision boundary and decision regions for a 1-nearest neighbor classifier for three classes.

How to determine the good value for k?

- Determined experimentally
- Start with k=1 and use a test set to validate the error rate of the classifier
- Repeat with k=k+2
- Choose the value of k for which the error rate is minimum
- Note: k should be odd number to avoid ties

Continuous-valued target functions

- kNN approximating continuous-valued target functions
- Calculate the mean value of the *k* nearest training examples rather than calculate their most common value

$$f : \Re^d \rightarrow \Re$$

$$\hat{f}(x_q) \leftarrow \frac{\sum_{i=1}^{k} f(x_i)}{k}$$

Distance Weighted

- Refinement to kNN is to weight the contribution of each *k* neighbor according to the distance to the query point x_q
  - Greater weight to closer neighbors
  - For discrete target functions

$$\hat{f}(x_q) \leftarrow \underset{v \in V}{\arg\max} \sum_{i=1}^{k} w_i \delta(v, f(x_i))$$

$$w_i = \begin{cases} \dfrac{1}{d(x_q, x_i)^2} & if \quad x_q \neq x_i \\ 1 & else \end{cases}$$

  - For real valued functions

$$\hat{f}(x_q) \leftarrow \frac{\sum_{i=1}^{k} w_i f(x_i)}{\sum_{i=1}^{k} w_i}$$

$$w_i = \begin{cases} \dfrac{1}{d(x_q, x_i)^2} & if \quad x_q \neq x_i \\ 1 & else \end{cases}$$

Curse of Dimensionality

- Imagine instances described by 20 features (attributes) but only 3 are relevant to target function
- Curse of dimensionality: nearest neighbor is easily misled when instance space is high-dimensional
- Dominated by large number of irrelevant features

Possible solutions

- Stretch j-th axis by weight z_j, where $z_1,\ldots,z_n$ chosen to minimize prediction error (weight different features differently)
- Use cross-validation to automatically choose weights $z_1,\ldots,z_n$
- Note setting z_j to zero eliminates this dimension altogether (feature subset selection)
- Principal Component Analysis

When to Consider Nearest Neighbors

- Instances map to points in R^d
- Less than 20 features (attributes) per instance, typically normalized
- Lots of training data

Advantages:

- Training is very fast
- Learn complex target functions
- Do not loose information

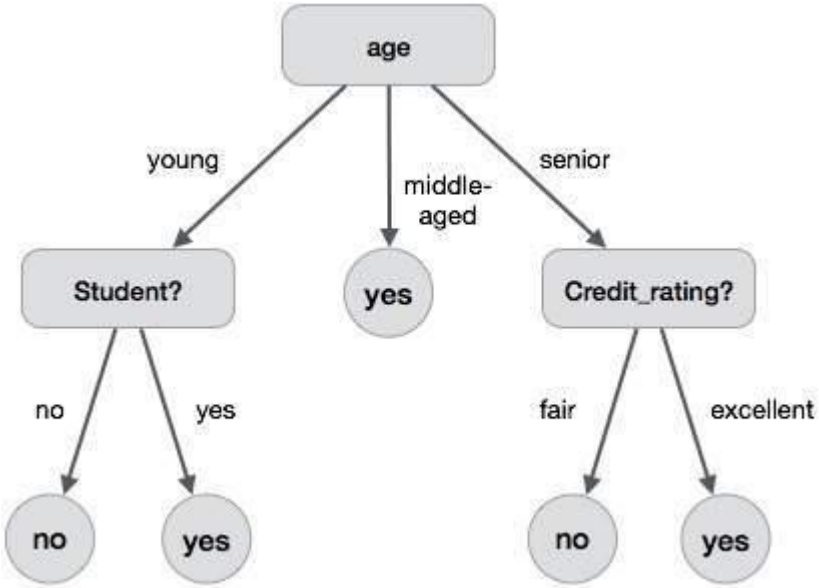
Disadvantages:

- Slow at query time
  - Presorting and indexing training samples into search trees reduces time
- Easily fooled by irrelevant features (attributes)

Decision Trees: Background

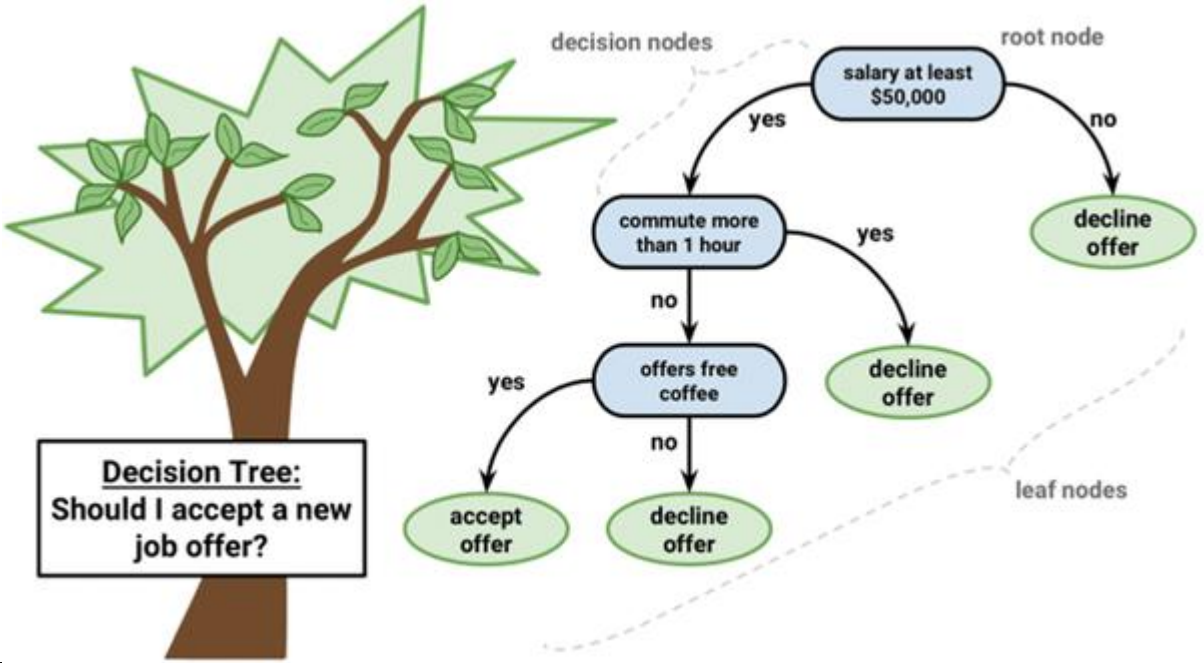
Goal: Categorization

- Given an event, predict is category. Examples:
  - Who won a given ball game?
  - How should we file a given email?
  - What word sense was intended for a given occurrence of a word?
- Event = list of features. Examples:
  - Ball game: Which players were on offense?
  - Email: Who sent the email?
  - Disambiguation: What was the preceding word?



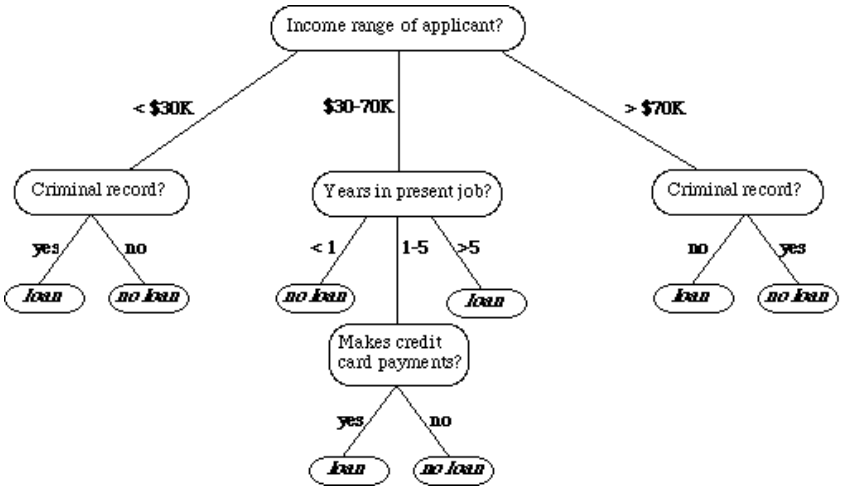
Decision Tree Algorithms

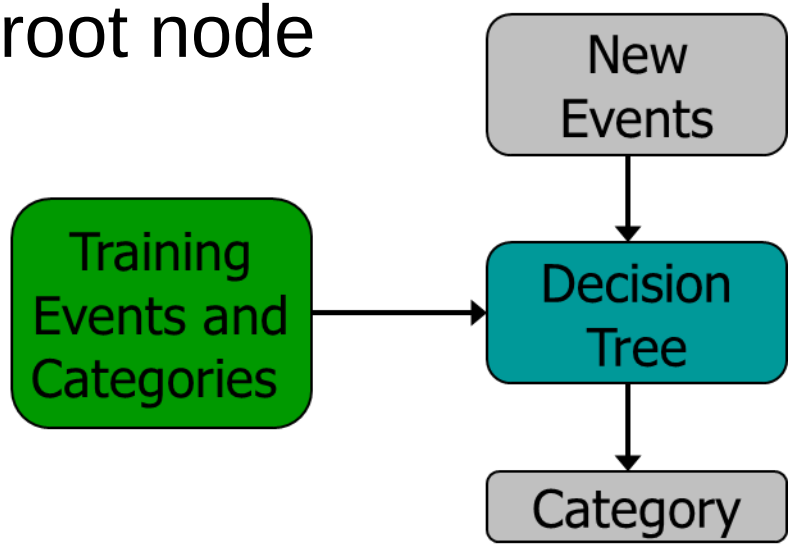
- The Decision Tree based methods develop a model of decisions considering the actual values of attributes in the data.
- Here decisions fork in tree structures until a prediction decision is made for a given record.
- These decision trees are trained on data for classification and regression problems.
- Decision trees are often fast and accurate highly used in machine learning.
- Well known decision tree algorithms are,
  - Classification and Regression Tree (CART)
  - Iterative Dichotomiser 3 (ID3)
  - C4.5 and C5.0
  - Chi-squared Automatic Interaction Detection (CHAID)
  - Decision Stump
  - M5
  - Conditional Decision Trees



Decision Trees

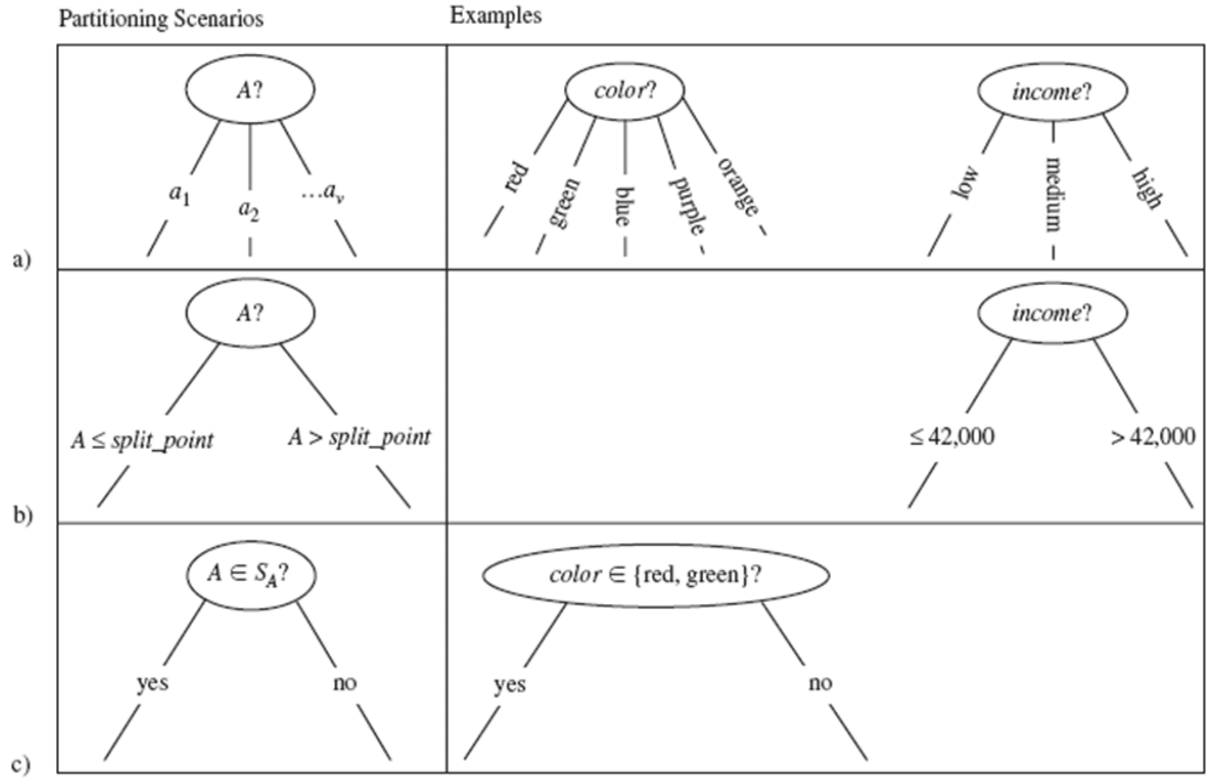
- A decision tree is a flowchart-like tree structure, where each internal node (nonleaf node) denotes a test on attribute, each branch represents an outcome of the test, and leaf node (or terminal) holds a class label.
- Decision tree induction is the learning of decision trees from class-labeled training tuples(instances).
- Decision tree is used to predict categories for new events.
- The training data is used to build the decision tree.
- The topmost node in a tree is the root node





Decision Tree Induction

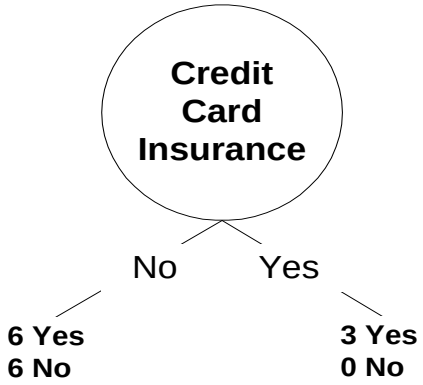
- During late 1970's, Ross Quinlan, a researcher in Machine Learning, developed a decision tree algorithm as ID3 (Iterative Dichotomiser).
- This work is expanded and presented C4.5, which became a benchmark to which newer supervised learning algorithms are often compared.
- Three possibilities for partitioning tubles based on the splitting criterion
  - a- If A is discrete-valued
  - b- If A is continuous-value, then two branchs are grown
  - c- If A is discrete-valued and binary tree must be produced



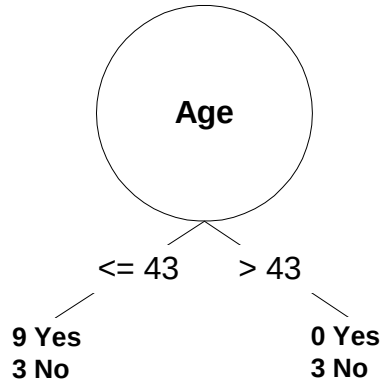
Decision Trees for Credit card promotion

Table: **The Credit Card Promotion Database**

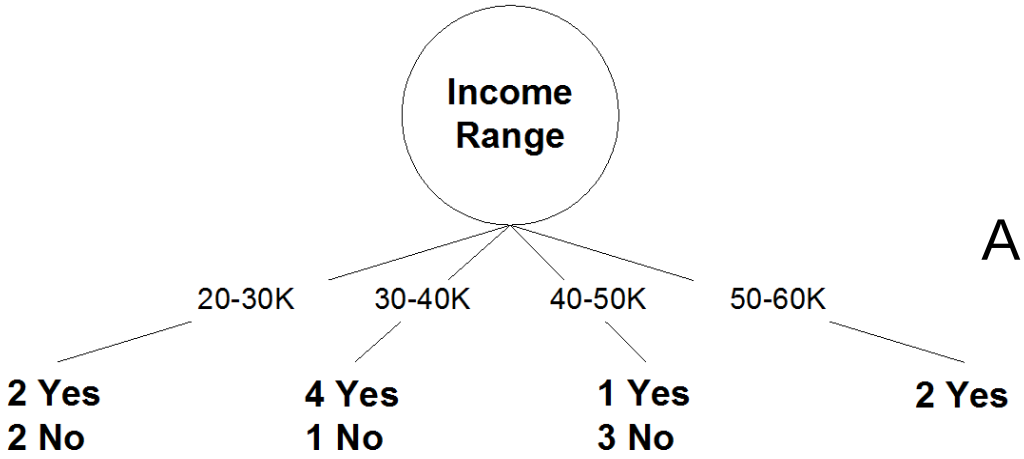
Income Range	Life Insurance Promotion	Credit Card Insurance	Sex	Age
40–50K	No	No	Male	45
30–40K	Yes	No	Female	40
40–50K	No	No	Male	42
30–40K	Yes	Yes	Male	43
50–60K	Yes	No	Female	38
20–30K	No	No	Female	55
30–40K	Yes	Yes	Male	35
20–30K	No	No	Male	27
30–40K	No	No	Male	43
30–40K	Yes	No	Female	41
40–50K	Yes	No	Female	43
20–30K	Yes	No	Male	29
50–60K	Yes	No	Female	39
40–50K	No	No	Male	55
20–30K	Yes	Yes	Female	19



A partial decision tree with root node=CreditCardInsurance

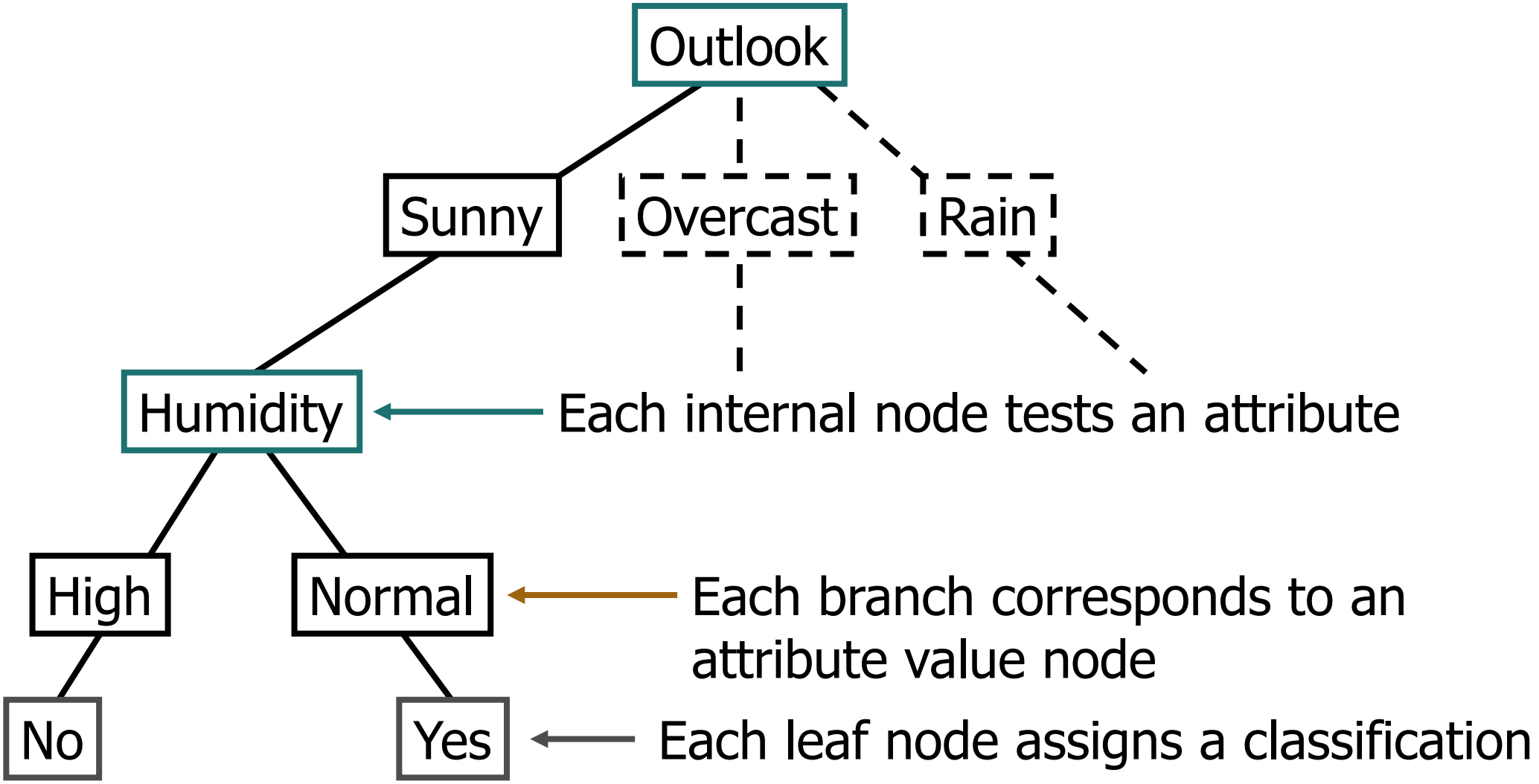


A partial decision tree with root node=age



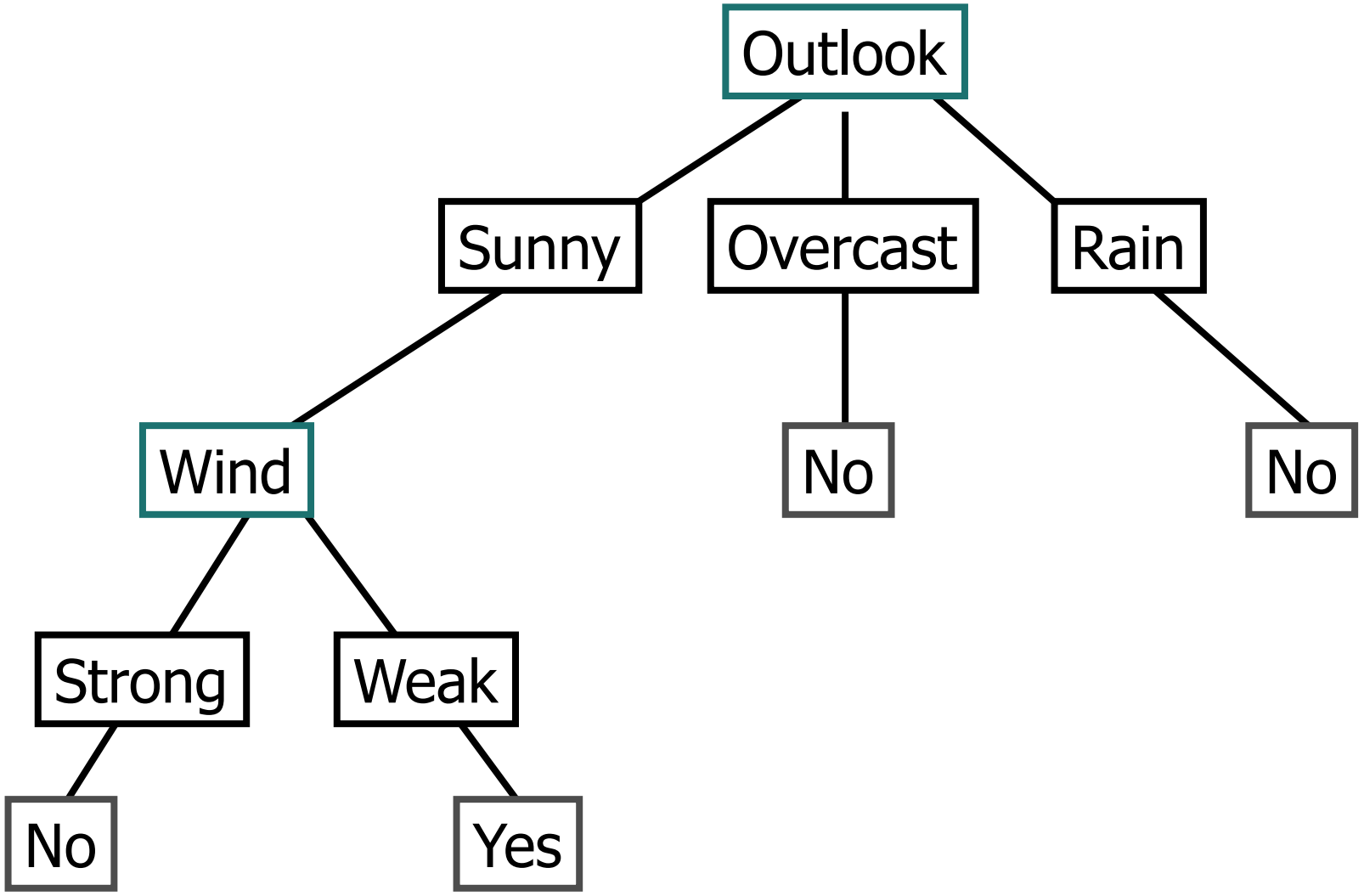
A partial decision tree with root node=income range

Decision Tree for PlayTennis



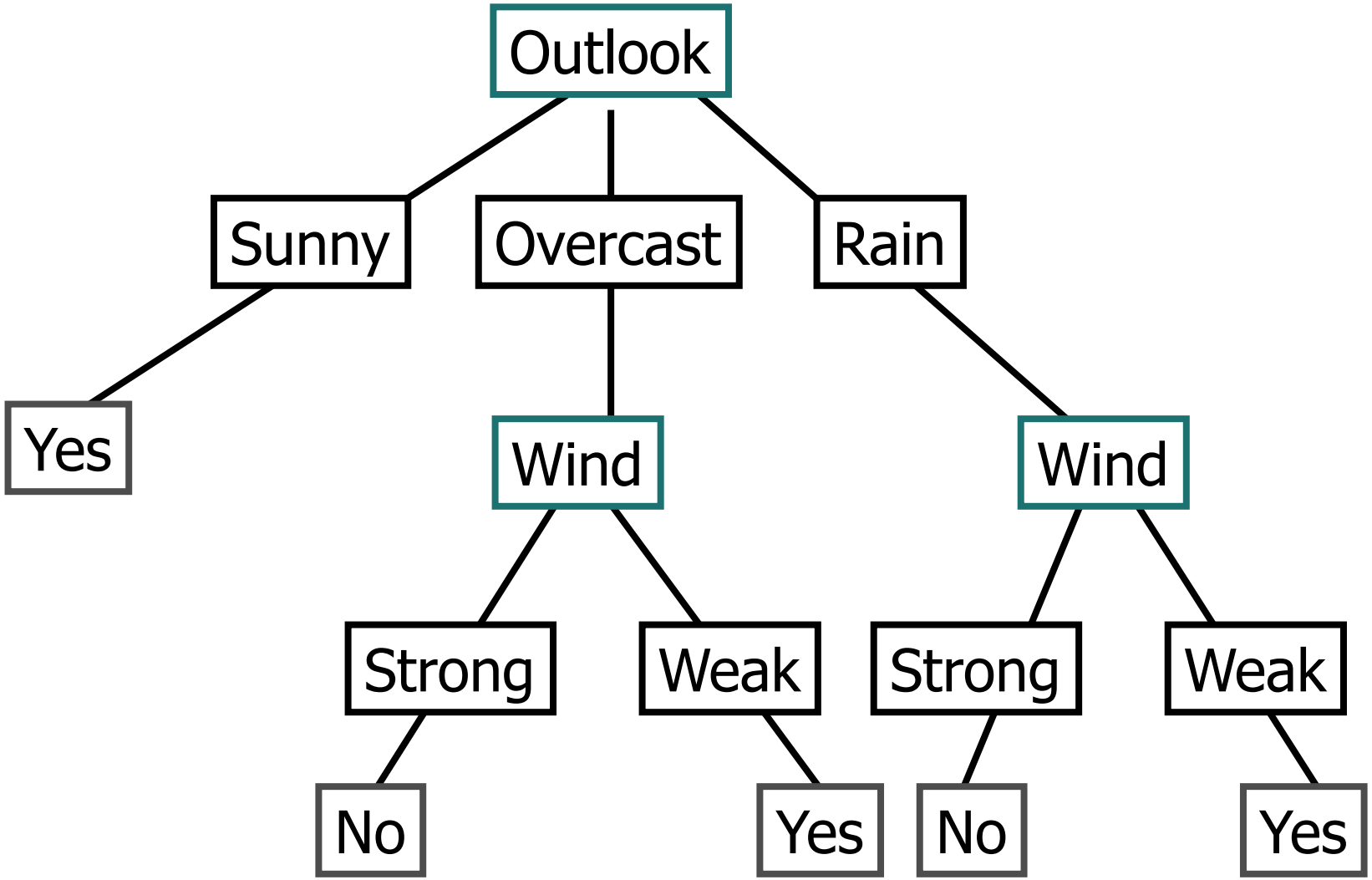
Decision Tree for Conjunction

Outlook=Sunny $\wedge$ Wind=Weak



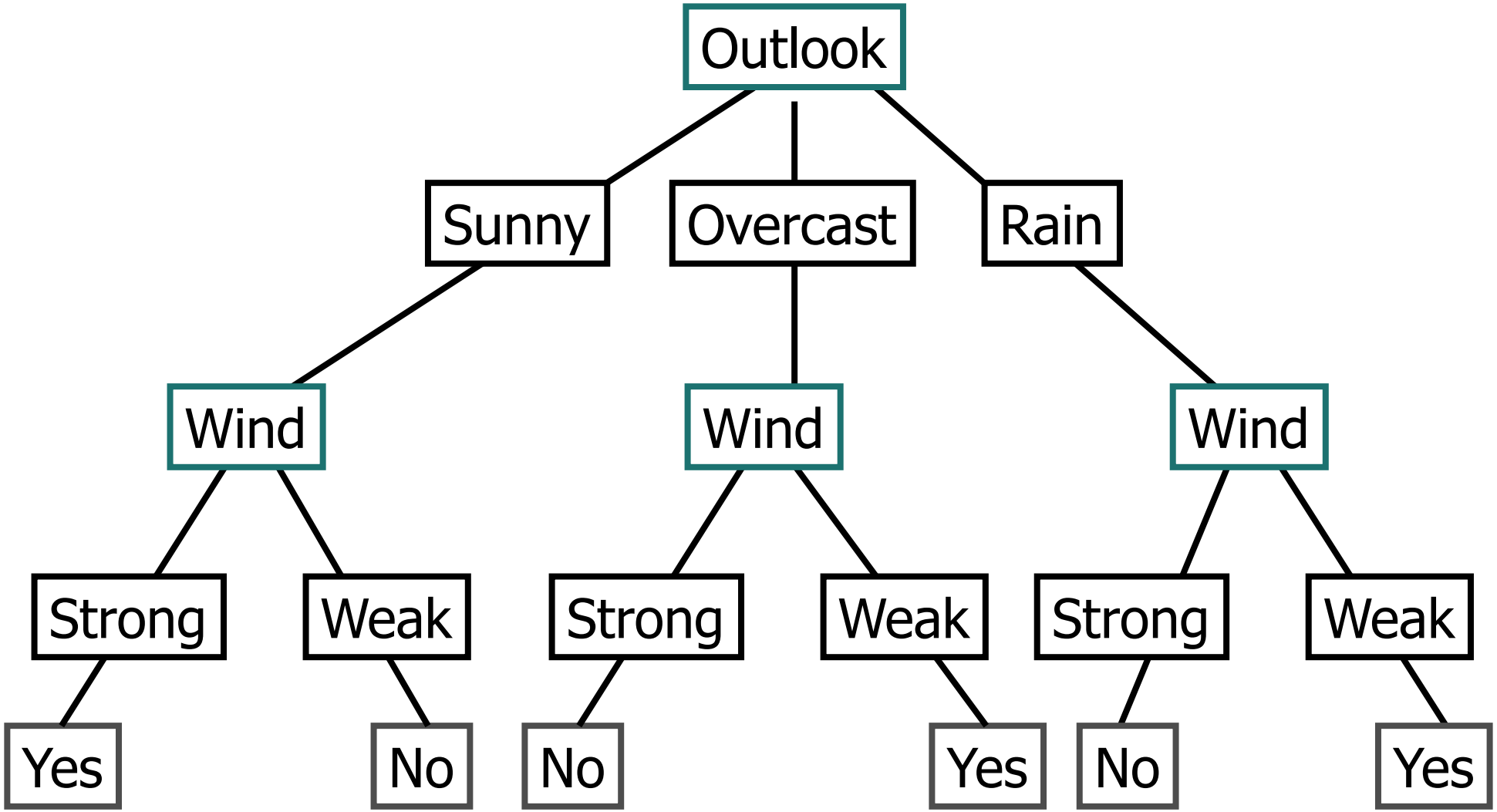
Decision Tree for Disjunction

Outlook=Sunny $\vee$ Wind=Weak



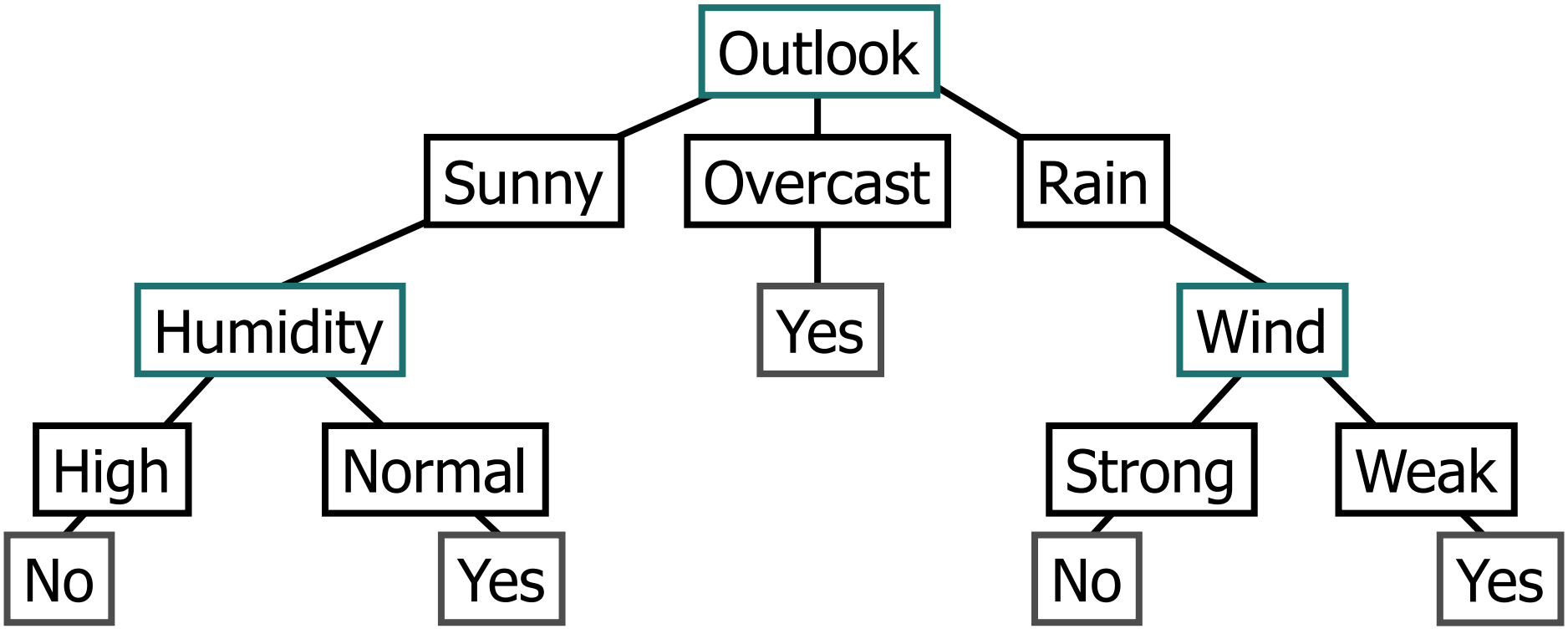
Decision Tree for XOR

Outlook=Sunny XOR Wind=Weak



Decision Tree

Decision trees represent disjunctions of conjunctions



(Outlook=Sunny $\wedge$ Humidity=Normal)
$\vee$ (Outlook=Overcast)
$\vee$ (Outlook=Rain $\wedge$ Wind=Weak)

When to consider Decision Trees

- Instances describable by attribute-value pairs
- Target function is discrete valued
- Disjunctive hypothesis may be required
- Possibly noisy training data
- Missing attribute values
- Examples:
  - Medical diagnosis
  - Credit risk analysis
  - Object classification for robot manipulator (Tan 1993)

Top-Down Induction of Decision Trees: ID3

- A ← the “best” decision attribute for next *node*
- Assign A as decision attribute for *node*
- For each value of A create new descendant
- Sort training examples to leaf node according to the attribute value of the branch
- If all training examples are perfectly classified (same value of target attribute) stop, else iterate over new leaf nodes.

An Algorithm for Building Decision Trees

1. Let *T* be the set of training instances.
2. Choose an attribute that best differentiates the instances in *T.*
3. Create a tree node whose value is the chosen attribute.
   a. Create child links from this node where each link represents a unique value for the chosen attribute.
   b. Use the child link values to further subdivide the instances into subclasses.
4. For each subclass created in step 3:
   a. If the instances in the subclass satisfy predefined criteria or if the set of remaining attribute choices for this path is null, specify the classification for new instances following this decision path.
   b. If the subclass does not satisfy the criteria and there is at least one attribute to further subdivide the path of the tree, let *T* be the current set of subclass instances and return to step 2.

Attribute Selection Measure: Information Gain

- Select the attribute with the highest information gain
- Let p_i be the probability that an arbitrary tuple in D belongs to class C_i, estimated by $|C_{i,D}|/|D|$

- Expected information (entropy) needed to classify a tuple in D:

$$Info(D) = -\sum_{i=1}^{m} p_i \log_2(p_i)$$

- Information needed (after using A to split D into v partitions) to classify D:

$$Info_A(D) = \sum_{j=1}^{v} \frac{|D_j|}{|D|} \times I(D_j)$$

- Information gained by branching on attribute A

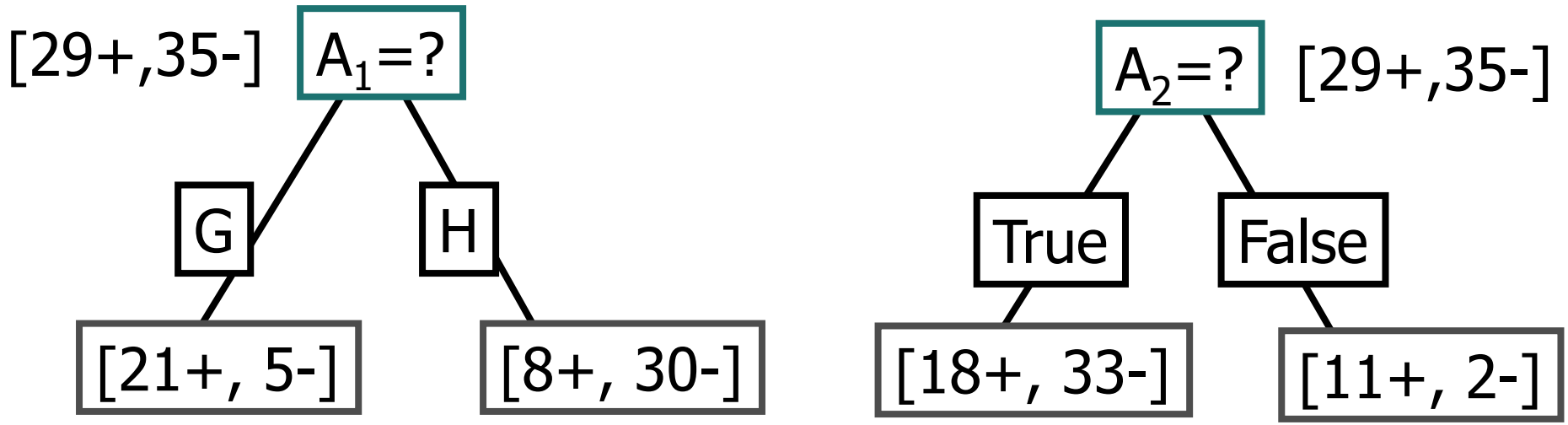
$$Gain(A) = Info(D) - Info_A(D)$$

Information Gain

- Gain(S,A): expected reduction in entropy due to sorting S on attribute A

$$Gain(S, A) \equiv Entropy(S) - \sum_{v \in D_A} \frac{|S_v|}{|S|} Entropy(S_v)$$

Entropy([29+,35-]) = -29/64 log2 29/64 – 35/64 log2 35/64
= 0.99



Information Gain

Entropy([21+,5-]) = 0.71

Entropy([8+,30-]) = 0.74

Gain(S,A_1)=Entropy(S)

-26/64*Entropy([21+,5-])

-38/64*Entropy([8+,30-])

=0.27

Entropy([18+,33-]) = 0.94

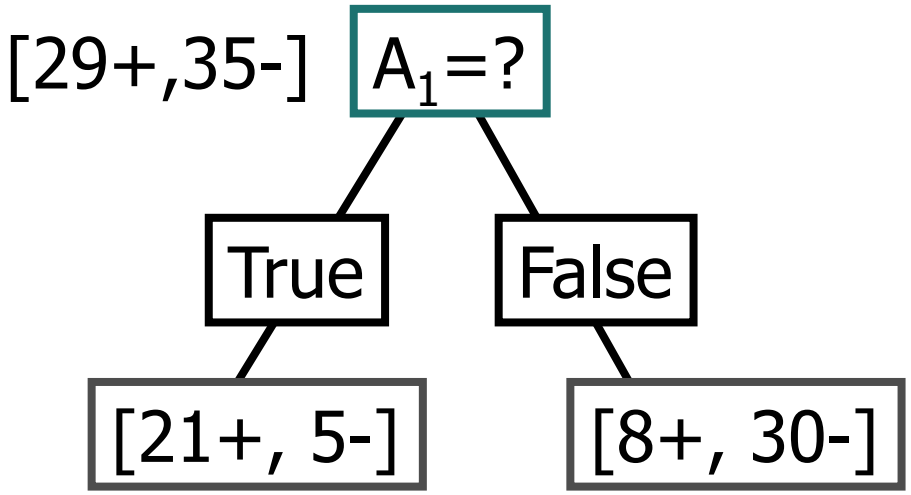
Entropy([11+,2-]) = 0.62

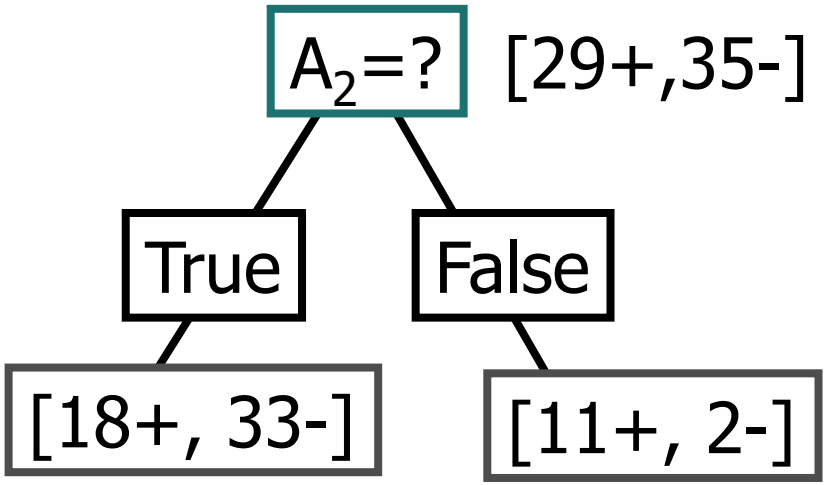
Gain(S,A2)=Entropy(S)

-51/64*Entropy([18+,33-])

-13/64*Entropy([11+,2-])

=0.12





Information Gain

- Class P: buys_computer = “yes”
- Class N: buys_computer = “no”

$$Info_{age}(D) = \frac{5}{14} I(2,3) + \frac{4}{14} I(4,0)$$

$$+ \frac{5}{14} I(3,2) = 0.694$$

$$Info(D) = I(9,5) = -\frac{9}{14}\log_2(\frac{9}{14}) - \frac{5}{14}\log_2(\frac{5}{14}) = 0.940$$

$\frac{5}{14} I(2,3)$ means “age <=30” has 5 out of 14 samples, with 2 yes’es and 3 no’s. Hence

$$Gain(age) = Info(D) - Info_{age}(D) = 0.246$$

Similarly, $Gain(income) = 0.029$

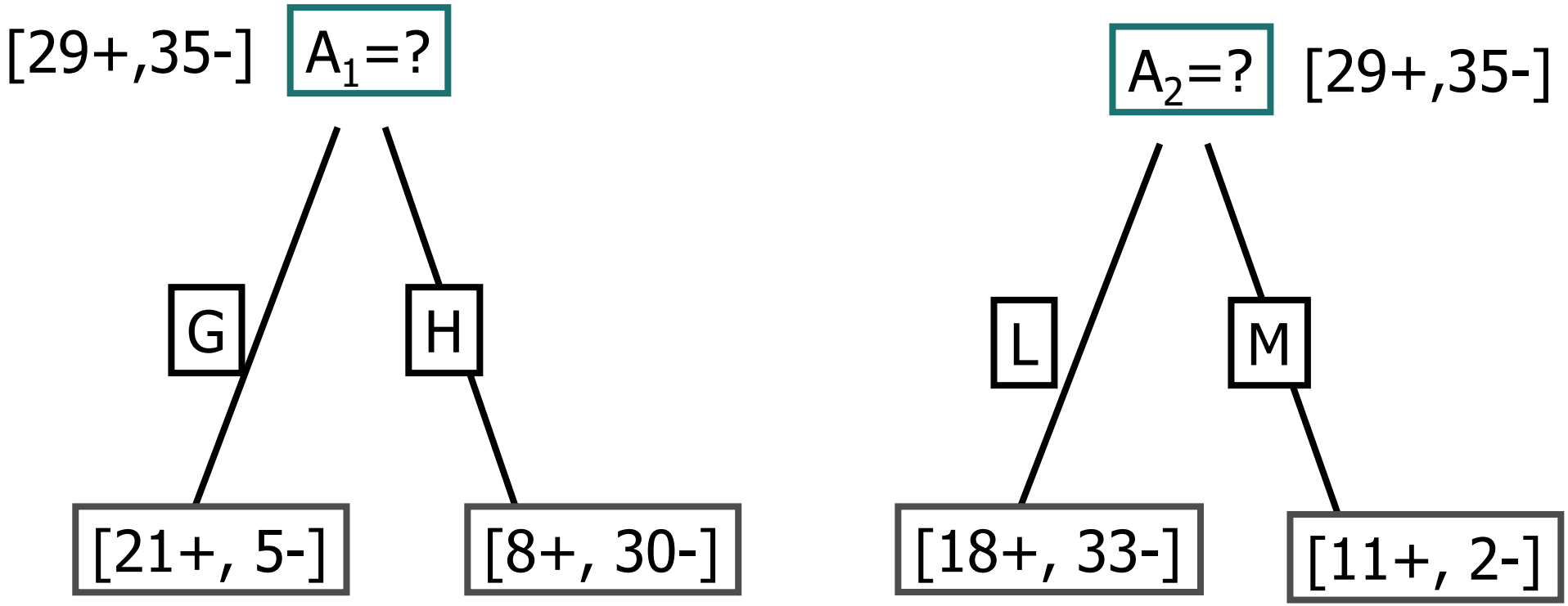
$Gain(student) = 0.151$

$Gain(credit\_rating) = 0.048$

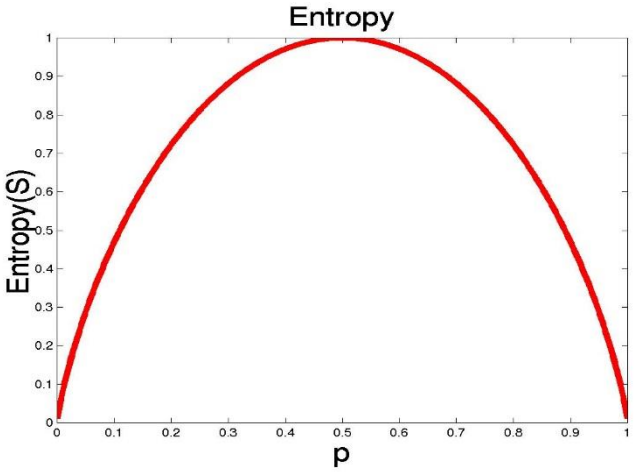
age	p_i	n_i	$I(p_i, n_i)$
<=30	2	3	0.971
31…40	4	0	0
>40	3	2	0.971

age	income	student	credit_rating	buys_computer
<=30	high	no	fair	no
<=30	high	no	excellent	no
31…40	high	no	fair	yes
>40	medium	no	fair	yes
>40	low	yes	fair	yes
>40	low	yes	excellent	no
31…40	low	yes	excellent	yes
<=30	medium	no	fair	no
<=30	low	yes	fair	yes
>40	medium	yes	fair	yes
<=30	medium	yes	excellent	yes
31…40	medium	no	excellent	yes
31…40	high	yes	fair	yes
>40	medium	no	excellent	no

Which attribute is best?



Entropy



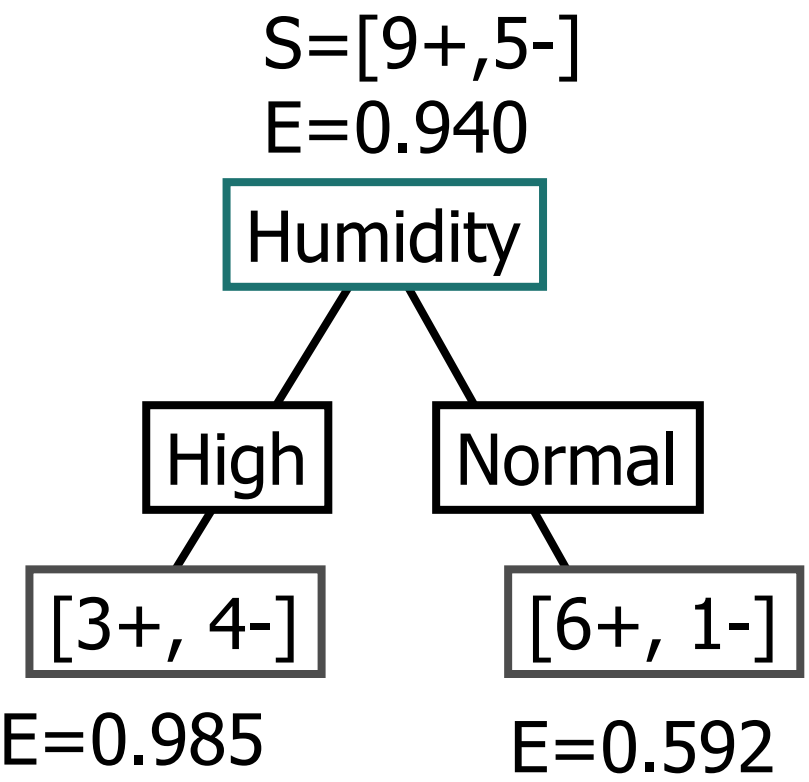
- S is a sample of training examples
- p_+ is the proportion of positive examples
- p_- is the proportion of negative examples
- Entropy measures the impurity of S
  - Entropy(S) = $-p_+ \log_2 p_+ - p_- \log_2 p_-$

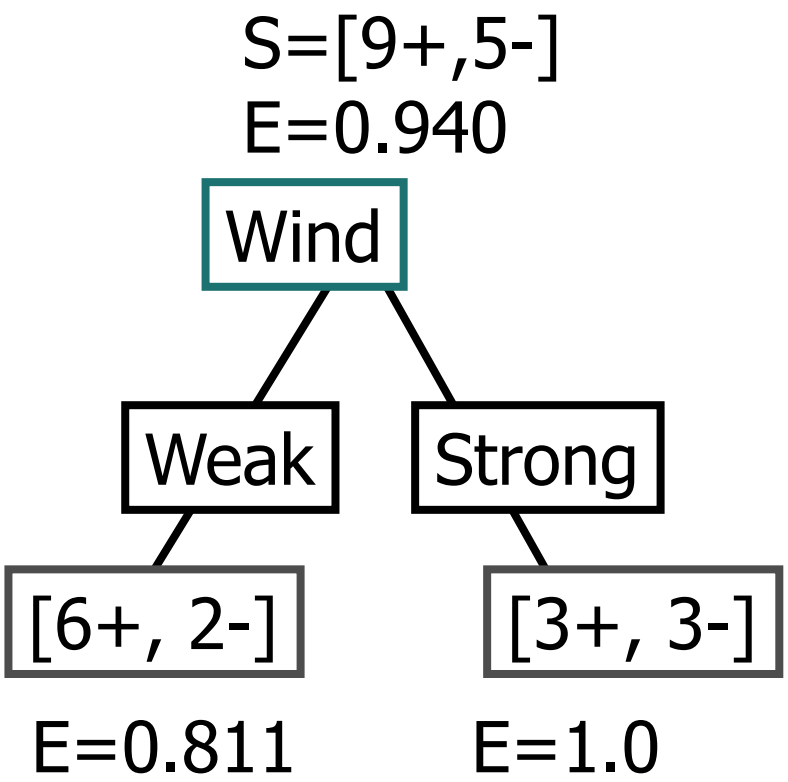
- Entropy(S)= expected number of bits needed to encode class (+ or -) of randomly drawn members of S.
- Information theory optimal length code assign $-\log_2 p$ bits to messages having probability p.
- So the expected number of bits to encode (+ or -) of random member of S:
  - $-p_+ \log_2 p_+ - p_- \log_2 p_-$

Training Examples

Day	Outlook	Temp.	Humidity	Wind	Play Tennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Weak	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cold	Normal	Weak	Yes
D10	Rain	Mild	Normal	Strong	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

Selecting the Next Attribute



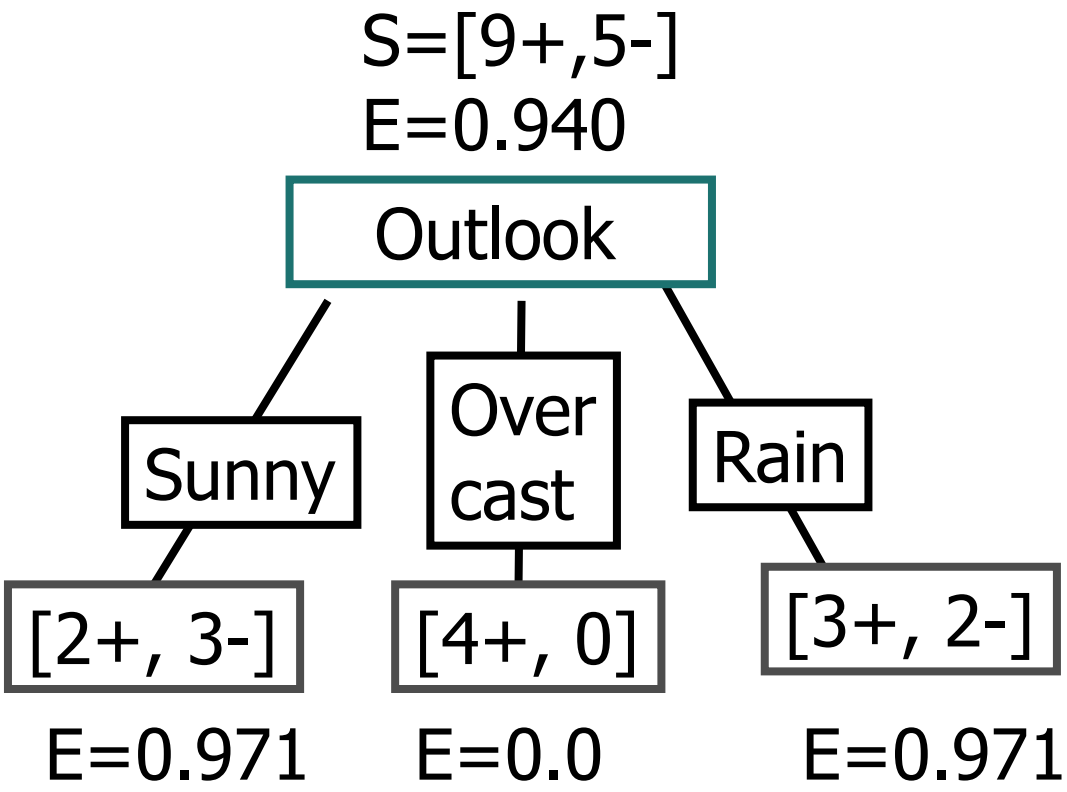


Gain(S,Humidity)
=0.940-(7/14)*0.985
– (7/14)*0.592
=0.151

Gain(S,Wind)
=0.940-(8/14)*0.811
– (6/14)*1.0
=0.048

Humidity provides greater info. gain than Wind, w.r.t target classification.

Selecting the Next Attribute



Gain(S,Outlook)
=0.940-(5/14)*0.971
-(4/14)*0.0 – (5/14)*0.0971
=0.247

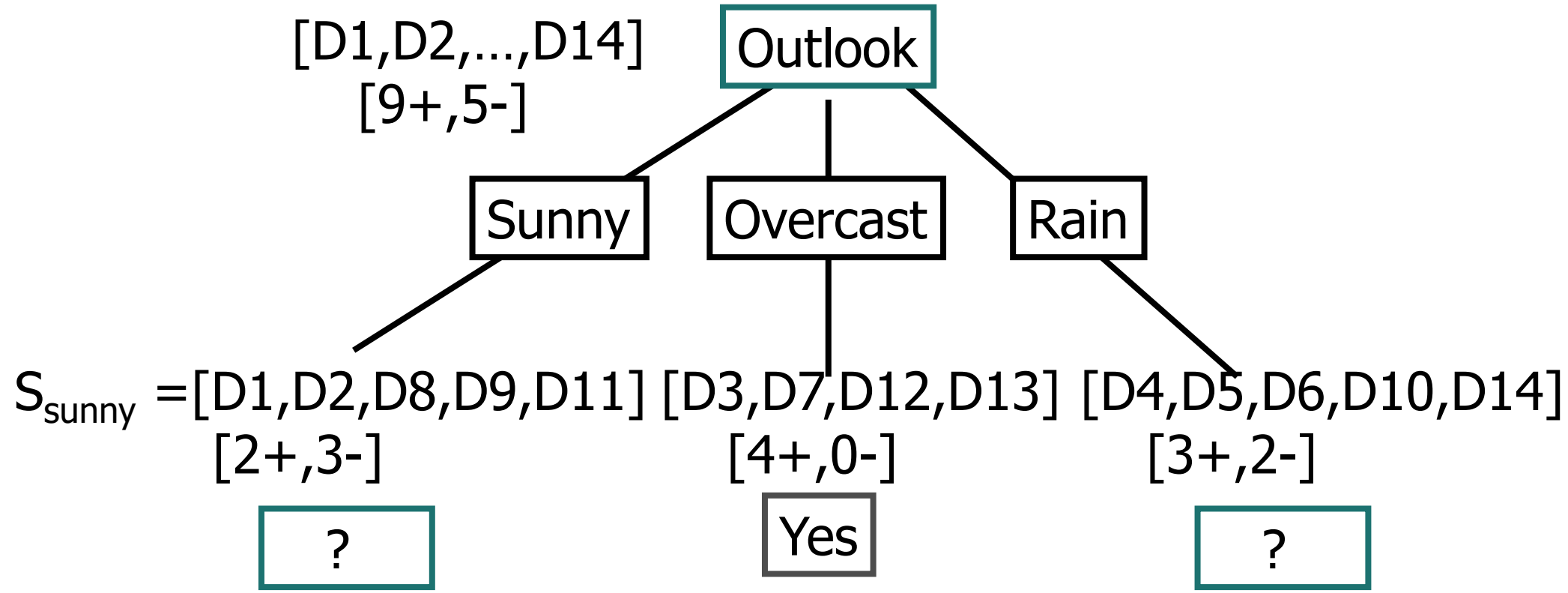
Selecting the Next Attribute

The information gain values for the 4 attributes are:

- Gain(S,Outlook) =0.247
- Gain(S,Humidity) =0.151
- Gain(S,Wind) =0.048
- Gain(S,Temperature) =0.029

where S denotes the collection of training examples

ID3 Algorithm

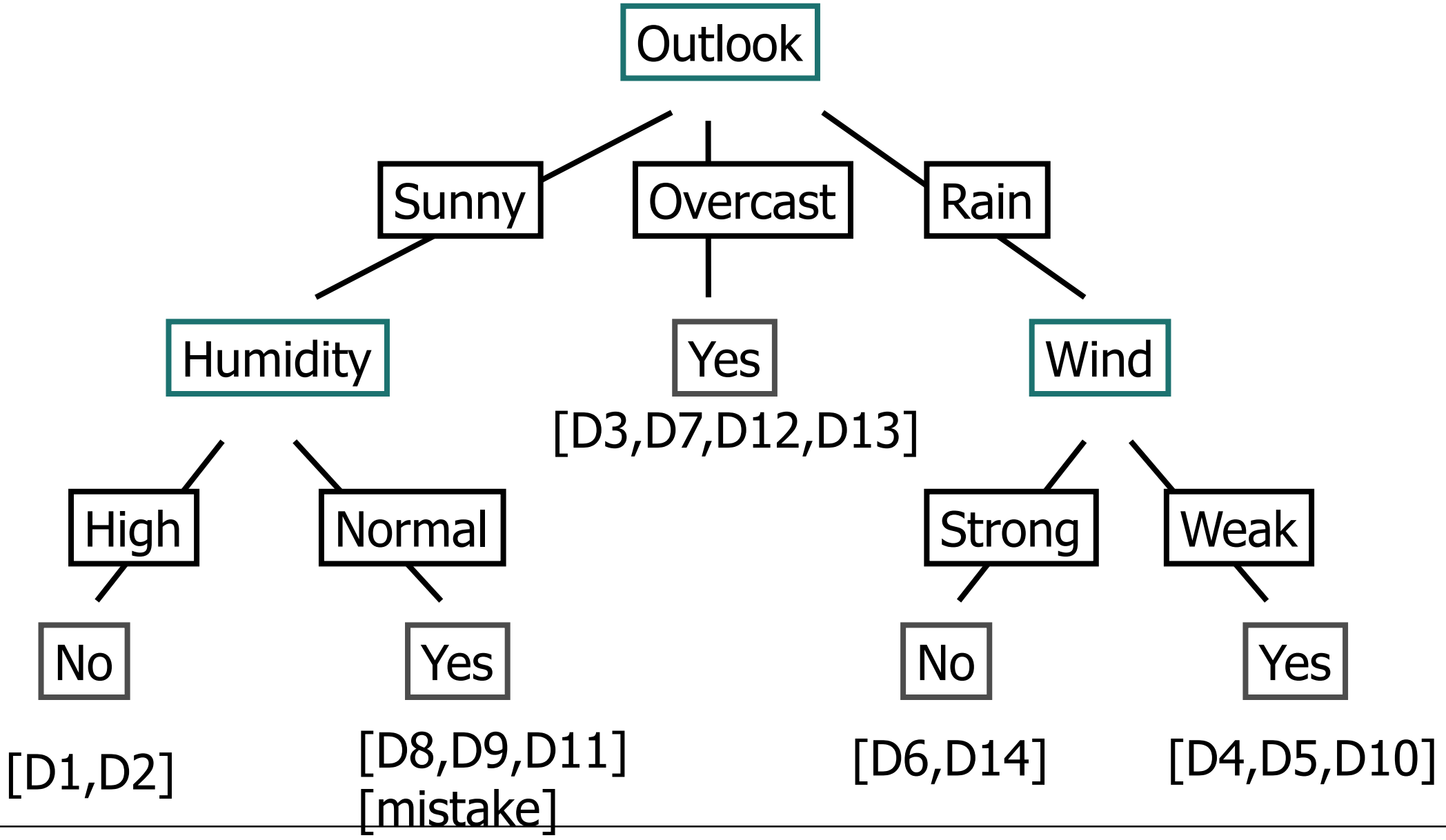


Gain(S_{sunny}, Humidity)=0.970-(3/5)0.0 – 2/5(0.0) = 0.970

Gain(S_{sunny}, Temp.)=0.970-(2/5)0.0 –2/5(1.0)-(1/5)0.0 = 0.570

Gain(S_{sunny}, Wind)=0.970= -(2/5)1.0 – 3/5(0.918) = 0.019

ID3 Algorithm



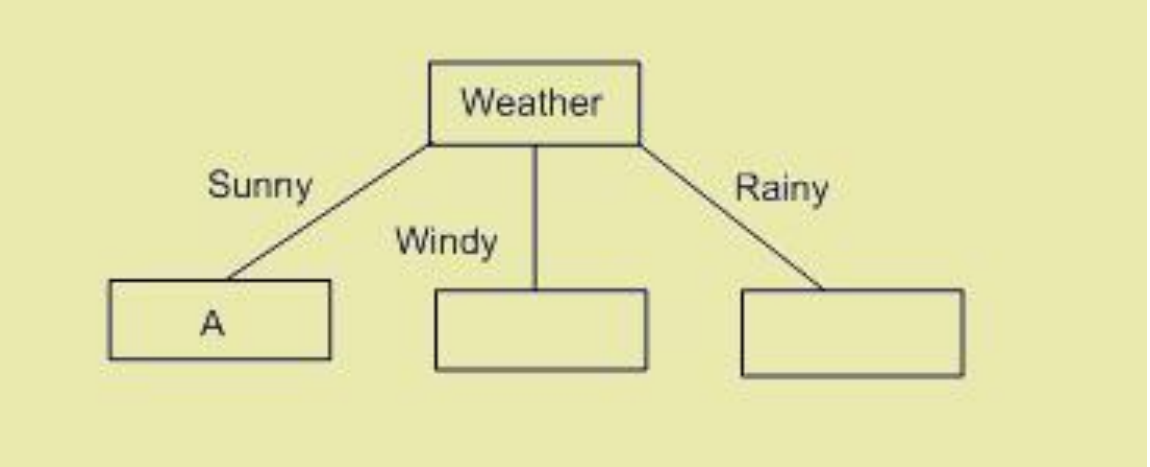
Decison Tree: Weekend example

- **Step-1:** Let *T* be the set of training instances.

Weekend (Example)	Weather	Parents	Money	Decision (Category
W1	Sunny	Yes	Rich	Cinema
W2	Sunny	No	Rich	Tennis
W3	Windy	Yes	Rich	Cinema
W4	Rainy	Yes	Poor	Cinema
W5	Rainy	No	Rich	Stay in
W6	Rainy	Yes	Poor	Cinema
W7	Windy	No	Poor	Cinema
W8	Windy	No	Rich	Shopping
W9	Windy	Yes	Rich	Cinema
W10	Sunny	No	Rich	Tennis

Decison Tree: Weekend example

- Step-2: Choose an attribute that best differentiates the instances in



- Step-3: Create a tree node whose value is the chosen attribute.

Weekend (Example)	Weather	Parents	Money	Decision (Category)
W1	Sunny	Yes	Rich	Cinema
W2	Sunny	No	Rich	Tennis
W10	Sunny	No	Rich	Tennis

Decison Tree: Entropy

- For the 10 Training instances
  - 6 : Go to Cinema
  - 2 : Play Tennis
  - 1 : Stay at Home
  - 1 : Go to Shopping
- Entropy:
  - $H(T)= - (6/10) \log_2(6/10) - (2/10) \log_2(2/10) - (1/10) \log_2(1/10) - (1/10) \log_2(1/10)$
  - $H(T)= 1,571$
- Gain(T, weather)= ?
  - Sunny=3 (1 Cinema, 2 Tennis)
  - Windy=4 (3 Cinema, 1 Shopping)
  - Rainy=3 (2 Cinema, 1 Stay in)
  - $Entropy(T_{sunny})= - (1/3) \log_2 (1/3) - (2/3) \log_2 (2/3)=0,918$
  - $Entropy(T_{windy})= - (3/4) \log_2 (3/4) - (1/4) \log_2 (1/4) =0,811$
  - $Entropy(T_{rainy})= - (2/3) \log_2 (2/3) - (1/3) \log_2 (1/3) =0,918$
- $Gain(T, weather) = Entropy(T)- ((P(sunny)Entropy(T_{sunny}) + P(windy) Entropy(T_{windy})+ P(rainy) Entropy(T_{rainy}))$
  $=1,571- ((3/10)Entropy(T_{sunny})+(4/10)Entropy(T_{windy})+ (3/10)Entropy(T_{rainy}))$
- Gain(T, weather) = 0,70

Decision Tree: Information Gain

- Gain(T, parents)= ?
  - Yes=5 (5 Cinema)
  - No =5 (2 Tennis, 1 Cinema, 1 Shopping, 1 Stay in)
  - Entropy(T_{yes})= - (5/5) $\log_2$ (5/5) = 0
  - Entropy(T_{no})= - (2/5) $\log_2$ (2/5) - 3(1/5) $\log_2$ (1/5) =1,922

- Gain(T, parents) = Entropy(T)- ((P(yes)Entropy(T_{yes}) + P(no) Entropy(T_{no}))
  =1,571- ((5/10)Entropy(T_{yes})+(5/10)Entropy(T_{no}))
- Gain(T, parents)=0,61

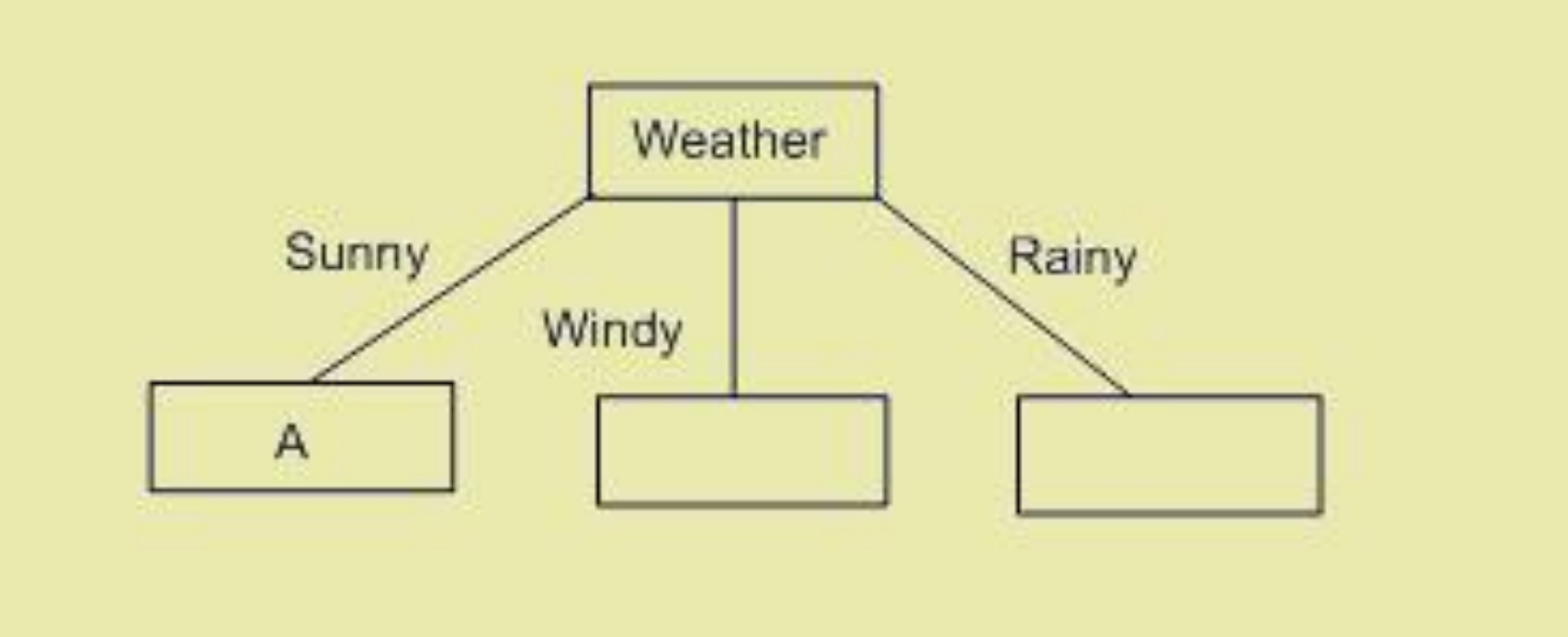
- Gain(T, money)= ?
  - Rich=7 (3 Cinema, 2 Tennis, 1 Shopping, 1 Stay in)
  - Poor=3 (3 Cinema)
  - Entropy(T_{rich})= 1,842
  - Entropy(T_{poor})= 0

- Gain(T, money) = Entropy(T)- ((P(rich)Entropy(T_{rich}) +P(poor) Entropy(T_{poor}))
  =1,571- ((5/10)Entropy(T_{rich})+(5/10)Entropy(T_{poor}))
- Gain(T, money)=0,2816

Decision Tree: Information Gain

- Gain(T, weather) =0,70
- Gain(T, parents)=0,61
- Gain(T, money)=0,2816

Create a tree node whose value is the chosen attribute.



Decision Trees:

- Create child links from this node where each link represents a unique value for the chosen attribute.

Weekend (Example)	Weather	Parents	Money	Decision (Category)
W1	Sunny	Yes	Rich	Cinema
W2	Sunny	No	Rich	Tennis
W10	Sunny	No	Rich	Tennis

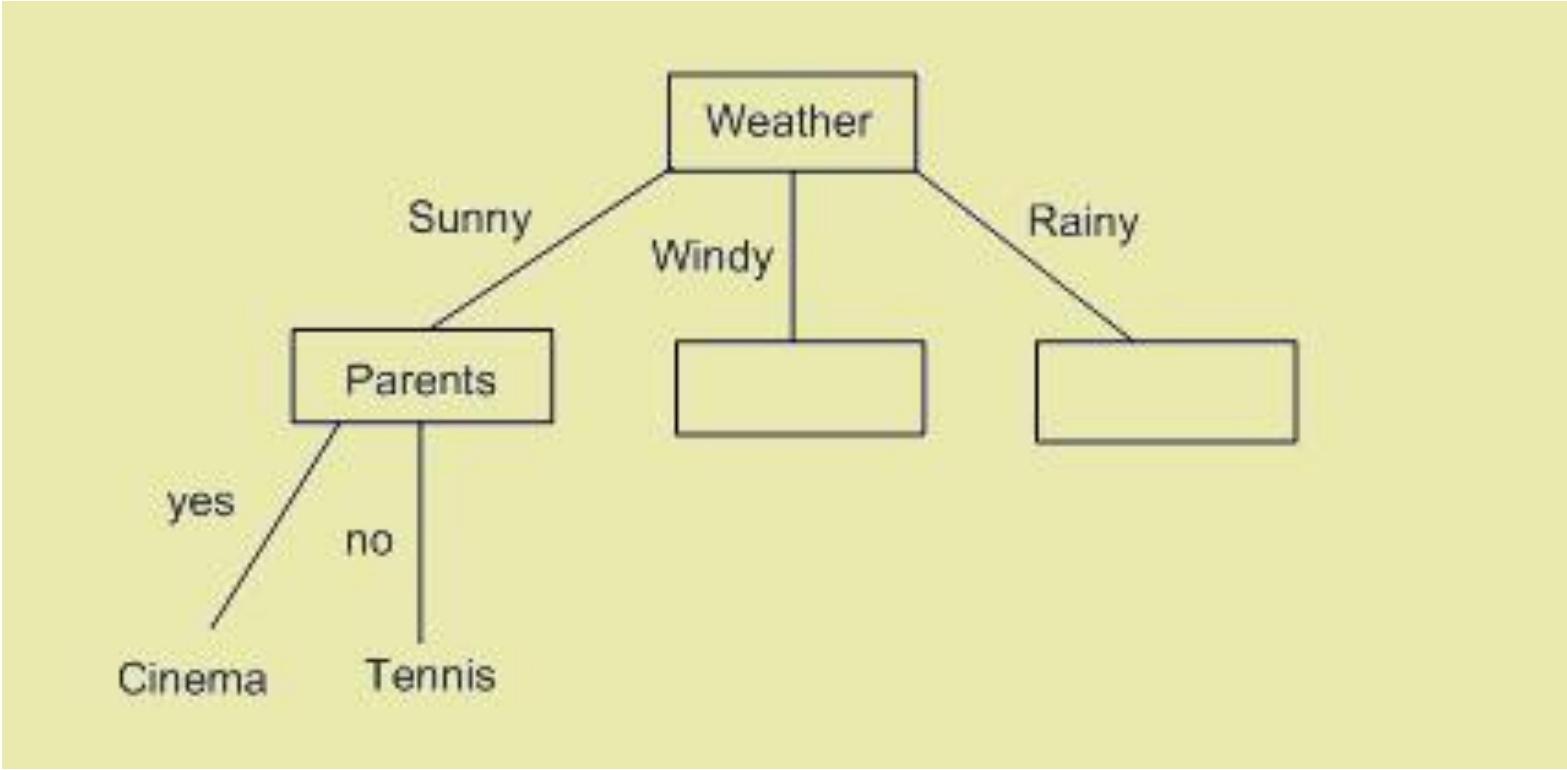
- Use the child link values to further subdivide the instances into subclasses.

$$Gain(S_{sunny}, parents) = 0.918 - (|S_{yes}|/|S|)*Entropy(S_{yes}) - (|S_{no}|/|S|)*Entropy(S_{no})$$
$$= 0.918 - (1/3)*0 - (2/3)*0 = 0.918$$

$$Gain(S_{sunny}, money) = 0.918 - (|S_{rich}|/|S|)*Entropy(S_{rich}) - (|S_{poor}|/|S|)*Entropy(S_{poor})$$
$$= 0.918 - (3/3)*0.918 - (0/3)*0 = 0.918 - 0.918 = 0$$

Decision Trees:

- If the instances in the subclass satisfy predefined criteria or if the set of remaining attribute choices for this path is null, specify the classification for new instances following this decision path.
- If the subclass does not satisfy the criteria and there is at least one attribute to further subdivide the path of the tree, let *T* be the current set of subclass instances and return to step 2.



Occam's Razor

”If two theories explain the facts equally weel, then the simpler theory is to be preferred”

Arguments in favor:

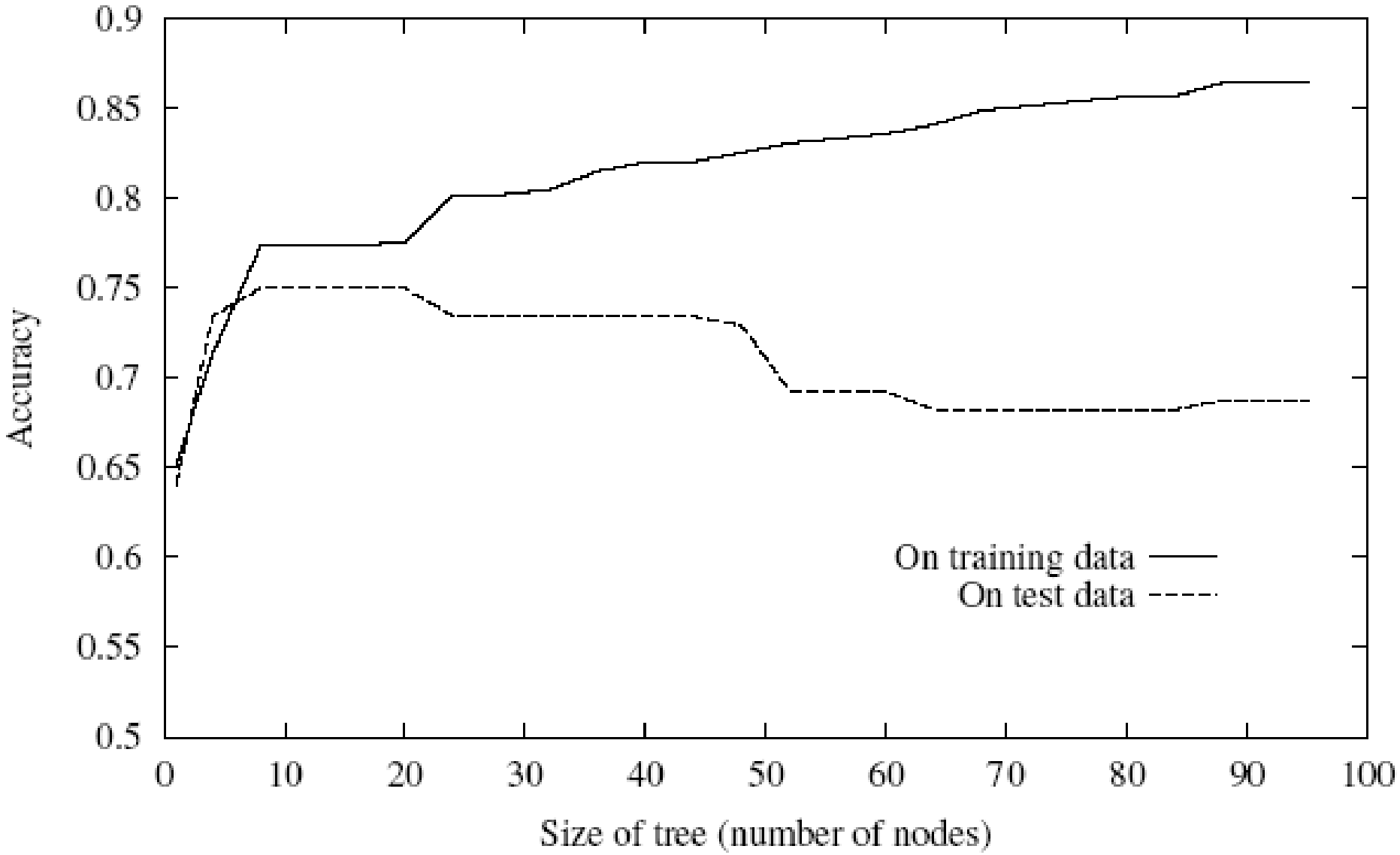
- Fewer short hypotheses than long hypotheses
- A short hypothesis that fits the data is unlikely to be a coincidence
- A long hypothesis that fits the data might be a coincidence

Arguments opposed:

- There are many ways to define small sets of hypotheses

Overfitting

- One of the biggest problems with decision trees is **Overfitting**



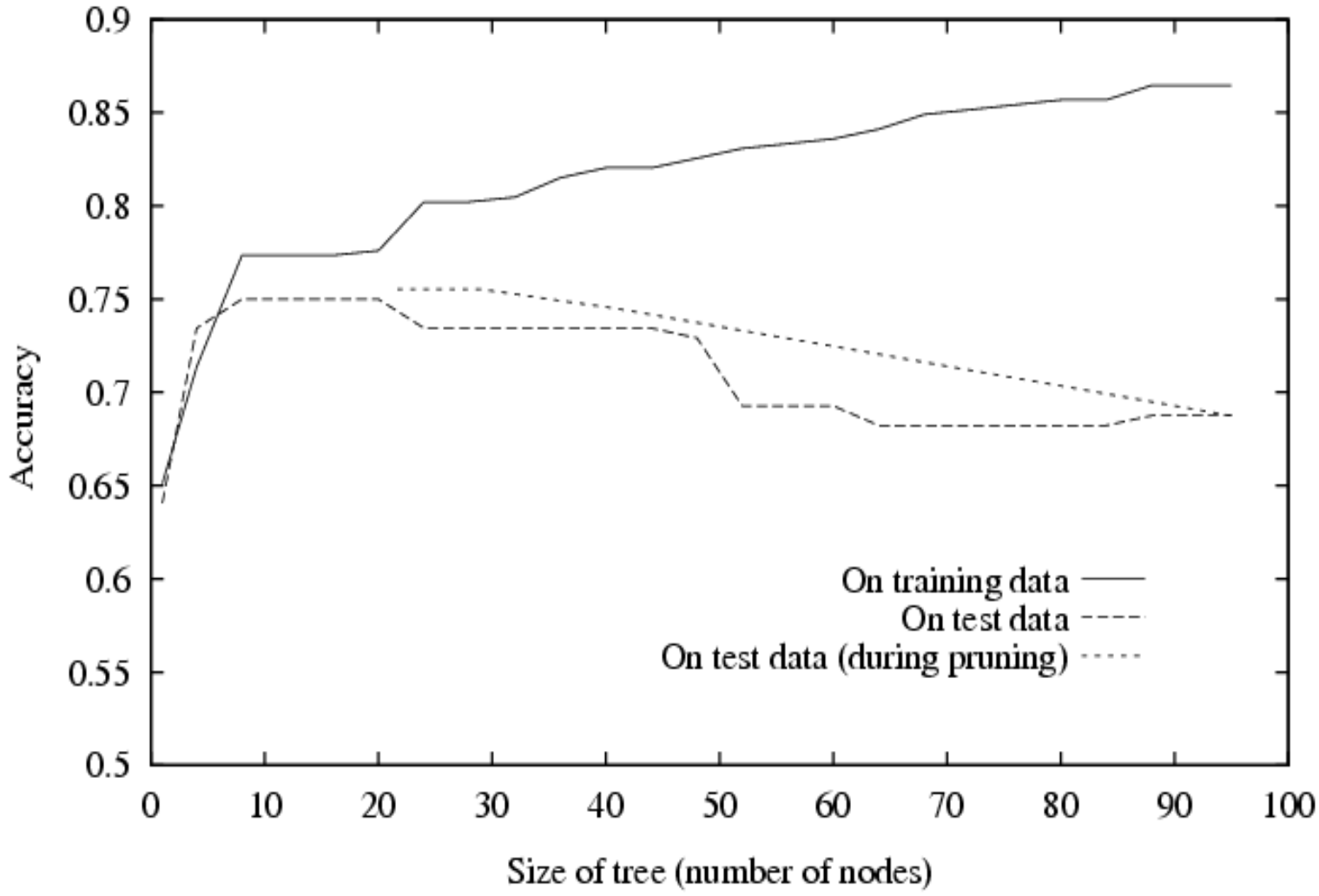
Avoid Overfitting

- Stop growing when split not statistically significant
- Grow full tree, then post-prune

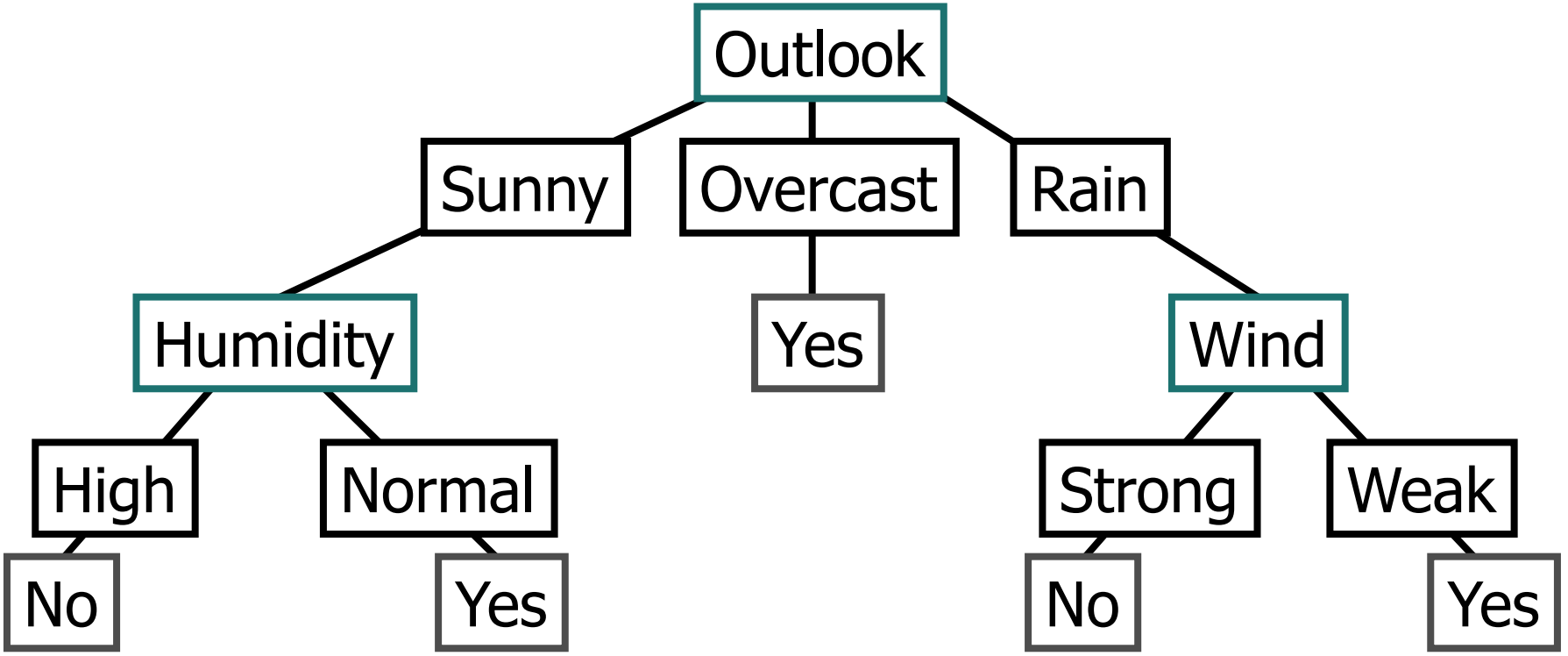
Select “best” tree:

- Measure performance over training data
- Measure performance over separate validation data set
- Min(|tree|+|misclassifications(tree)|)

Effect of Reduced Error Pruning



Converting a Tree to Rules



R_1: If (Outlook=Sunny) $\wedge$ (Humidity=High) Then PlayTennis=No
R_2: If (Outlook=Sunny) $\wedge$ (Humidity=Normal) Then PlayTennis=Yes
R_3: If (Outlook=Overcast) Then PlayTennis=Yes
R_4: If (Outlook=Rain) $\wedge$ (Wind=Strong) Then PlayTennis=No
R_5: If (Outlook=Rain) $\wedge$ (Wind=Weak) Then PlayTennis=Yes

Continuous Valued Attributes

Create a discrete attribute to test continuous

- Temperature = 24.5^0C
- (Temperature > 20.0^0C) = {true, false}

Where to set the threshold?

Temperature	15^0C	18^0C	19^0C	22^0C	24^0C	27^0C
PlayTennis	No	No	Yes	Yes	Yes	No

Unknown Attribute Values

What if some examples have missing values of A?

Use training example anyway sort through tree

- If node *n* tests A, assign most common value of A among other examples sorted to node *n*.
- Assign most common value of A among other examples with same target value
- Assign probability p_i to each possible value v_i of A
  - Assign fraction p_i of example to each descendant in tree

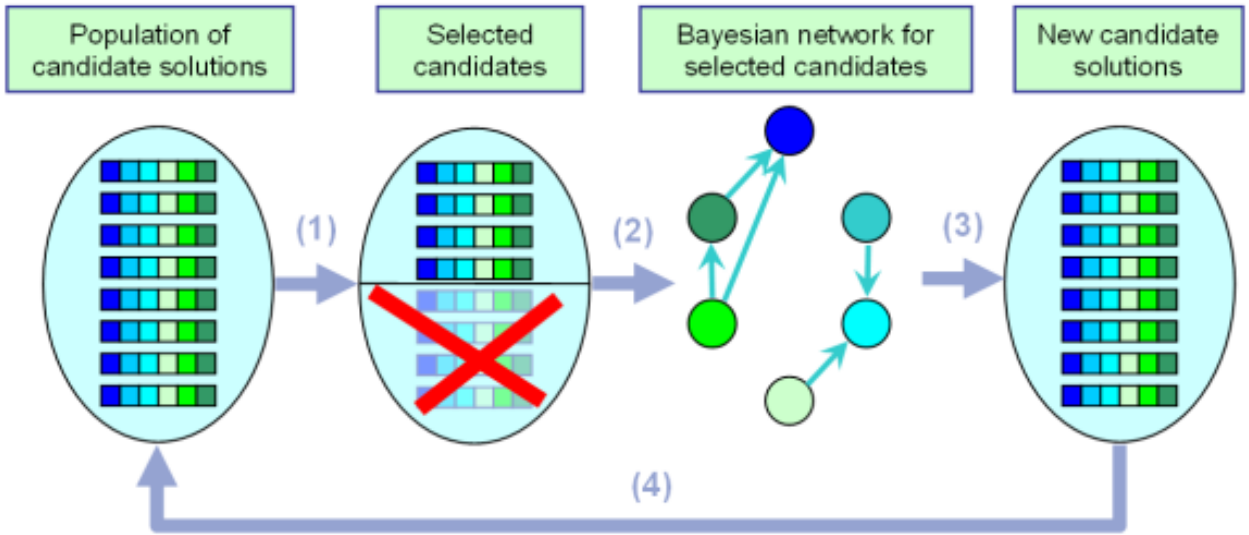
Classify new examples in the same fashion

Cross-Validation

- Estimate the accuracy of an hypothesis induced by a supervised learning algorithm
- Predict the accuracy of an hypothesis over future unseen instances
- Select the optimal hypothesis from a given set of alternative hypotheses
  - Pruning decision trees
  - Model selection
  - Feature selection
- Combining multiple classifiers (boosting)

Bayesian Algorithms

- These methods apply Bayes' Theorem for problems such as classification and regression.
- Bays theorem considers the prior probability and evidence for predicting the posterior probability.
- Highly used versions of Bayesian approaches are,
  - Naive Bayes
  - Gaussian Naive Bayes
  - Multinomial Naive Bayes
  - Averaged One-Dependence Estimators (AODE)
  - Bayesian Belief Network (BBN)
  - Bayesian Network (BN)



NAÏVE BAYESIAN CLASSIFIER

Bayes Theorem

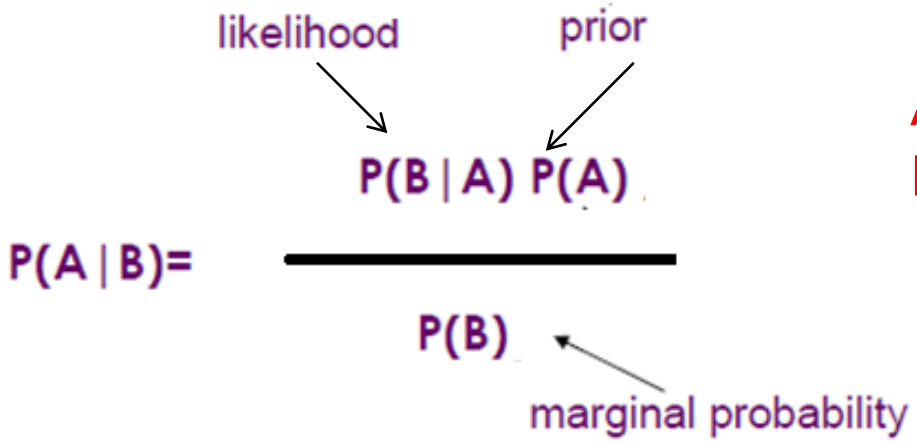
- Provide a mathematical rule explaining how you should change your existing beliefs in the light of new evidence
- Ex:
  - Without looking at the sky we ask you what you think about it
  - You believe that it is completely cloudy
  - Let us call your belief Hypothesis A
  - A : The sky is completely cloudy
  - P(A): The probability of the sky is being completely cloudy without having any observation. It is called Prior probability
- Now you observe that it is raining. we call this observation evidence B
- Now you would like to compute the probability of Hypothesis A given evidence B

Bayes Theorem

P(sky is completely cloudy | it rains) can be computed using:

prior × likelihood / marginal probability

- → **Prior** probability that the sky is completely cloudy
- → **Likelihood** that it rains given that the sky is completely cloudy
- → **Marginal probability**: used for normalization. It is the unconditional probability of the rain, regardless of the situation of the sky



A : The sky is completely cloudy
B : It is raining : Evidence

Naïve Bayesian Classifier

- X is a data tuple. In Bayesian term it is considered **"evidence"**
- H is some **hypothesis** that X belongs to a specified class C

$$P(H \mid X) = \frac{P(X \mid H)P(H)}{P(X)}$$

X ➔ (Age=youth, student=yes, creditRate=fair)

Example: **predict** whether a costumer **will buy a computer** or **not**

Likelihood: Given that **X** will buy computer, the prob. that X is youth, yes, fair

YES: X

NO :X

Naïve Bayesian Classifier

Principle

- Given a tuple **X**, the classifier will predict that X belongs to the class having the **highest posterior probability** conditioned on X

- Predict that tuple X belongs to the class C_i if and only if

$$P(C_i \mid X) > P(C_j \mid X) \quad \text{for} \quad 1 \le j \le m, j \ne i$$

- Maximize **$P(C_i \mid X)$**: find the **maximum posteriori hypothesis**

$$P(C_i \mid X) = \frac{P(X \mid C_i)P(C_i)}{P(X)}$$

Naïve Bayesian Classifier

- Maximize $P(C_i \mid X)$: find the **maximum posteriori hypothesis**

$$P(C_i \mid X) = \frac{P(X \mid C_i)P(C_i)}{P(X)}$$

  - P(X) is **constant** for all classes, thus, **maximize $P(X \mid C_i)P(C_i)$**
  - To maximize **$P(X \mid Ci)P(Ci)$**, we need to know class prior probabilities

→ Class prior probabilities can be estimated by $P(C_i) = |C_{i,D}| / |D|$

→ If the probabilities are not known, assume that
$P(C_1)=P(C_2)=\ldots=P(C_m) \Rightarrow$ **maximize $P(X \mid C_i)$**

Naïve Bayesian Classifier

- Assume **Class Conditional Independence** to reduce computational cost of **$P(X \mid C_i)$**
  - → given **$X(x_1,\ldots,x_n)$**, $P(X \mid C_i)$ is:

$$P(X \mid C_i) = \prod_{k=1}^{n} P(x_k \mid C_i)$$
$$= P(x_1 \mid C_i) \times P(x_2 \mid C_i) \times \ldots \times P(x_n \mid C_i)$$

  - → The probabilities $P(x_1 \mid C_i)$, …$P(x_n \mid C_i)$ can be estimated from the training tuples

Example

RID	age	income	student	credit-rating	class:buy_computer
1	youth	high	no	fair	no
2	youth	high	no	excellent	no
3	middle-aged	high	no	fair	yes
4	senior	medium	no	fair	yes
5	senior	low	yes	fair	yes
6	senior	low	yes	excellent	no
7	middle-aged	low	yes	excellent	yes
8	youth	medium	no	fair	no
9	youth	low	yes	fair	yes
10	senior	medium	yes	fair	yes
11	youth	medium	yes	excellent	yes
12	middle-aged	medium	no	excellent	yes
13	middle-aged	high	yes	fair	yes
14	senior	medium	no	excellent	no

Tuple to classify is

X (age=youth, income=medium, student=yes, credit=fair)

Maximize $P(X \mid C_i)P(C_i)$, for i=1,2

Example

RID	age	income	student	credit-rating	class:buy_computer
1	youth	high	no	fair	no
2	youth	high	no	excellent	no
3	middle-aged	high	no	fair	yes
4	senior	medium	no	fair	yes
5	senior	low	yes	fair	yes
6	senior	low	yes	excellent	no
7	middle-aged	low	yes	excellent	yes
8	youth	medium	no	fair	no
9	youth	low	yes	fair	yes
10	senior	medium	yes	fair	yes
11	youth	medium	yes	excellent	yes
12	middle-aged	medium	no	excellent	yes
13	middle-aged	high	yes	fair	yes
14	senior	medium	no	excellent	no

Tuple to classify is

X (age=youth, income=medium, student=yes, credit=fair)

Maximize $P(X \mid C_i)P(C_i)$, for i=1,2

Example

RID	age	income	student	credit-rating	class:buy_computer
1	youth	high	no	fair	no
2	youth	high	no	excellent	no
3	middle-aged	high	no	fair	yes
4	senior	medium	no	fair	yes
5	senior	low	yes	fair	yes
6	senior	low	yes	excellent	no
7	middle-aged	low	yes	excellent	yes
8	youth	medium	no	fair	no
9	youth	low	yes	fair	yes
10	senior	medium	yes	fair	yes
11	youth	medium	yes	excellent	yes
12	middle-aged	medium	no	excellent	yes
13	middle-aged	high	yes	fair	yes
14	senior	medium	no	excellent	no

Tuple to classify is

X (age=youth, income=medium, student=yes, credit=fair)

Maximize $P(X \mid C_i)P(C_i)$, for i=1,2

Naïve Bayesian Classifier

Second step: compute $P(X \mid C_i)$

$P(X \mid buys\_computer=no) = P(age=youth \mid buys\_computer=no) \times$
$P(income=medium \mid buys\_computer=no) \times$
$P(student=yes \mid buys\_computer=no) \times$
$P(credit\_rating=fair \mid buys\_computer=no)$
$= 0.019$

$P(age=youth \mid buys\_computer=no)=3/5=0.666$

$P(income=medium \mid buys\_computer=no)=2/5=0.400$

$P(tudent=yes \mid buys\_computer=no)=1/5=0.200$

$P(credit\_rating=fair \mid buys\_computer=no)=2/5=0.400$

Example

We have computed in the first and second steps:

$P(X \mid C_1)\, P(C_1)$: C1 is Yes Class
$P(X \mid C_2)\, P(C_2)$: C2 No Class

P(buys_computer=yes)=9/14=0.643

P(buys_computer=no)=5/14=0.357

P(X | buys_computer=yes)= 0.044

P(X | buys_computer=no)= 0.019

Third step: compute $P(X \mid C_i)P(C_i)$ for each class

P(X | buys_computer=yes)P(buys_computer=yes)=0.044 ×0.643=**0.028**

P(X | buys_computer=no)P(buys_computer=no)=0.019 ×0.357=**0.007**

The naïve Bayesian Classifier predicts buys_computer=yes for tuple X

Classical Example: Play Tennis?

Outlook	Temperature	Humidity	Windy	Class
sunny	hot	high	false	N
sunny	hot	high	true	N
overcast	hot	high	false	P
rain	mild	high	false	P
rain	cool	normal	false	P
rain	cool	normal	true	N
overcast	cool	normal	true	P
sunny	mild	high	false	N
sunny	cool	normal	false	P
rain	mild	normal	false	P
sunny	mild	normal	true	P
overcast	mild	high	true	P
overcast	hot	normal	false	P
rain	mild	high	true	N

Outlook=Rain temp= hot
Humidity =high windy= false
?????

Training set from Quinlan's book

Example

Calculate $P(X \mid C_1)\, P(C_1)$
Calculate $P(X \mid C_2)\, P(C_2)$ Select maximum

Outlook=Rain temp= hot Humid=High windy= false

Outlook	Temperature	Humidity	Windy	Class
sunny	hot	high	false	N
sunny	hot	high	true	N
overcast	hot	high	false	P
rain	mild	high	false	P
rain	cool	normal	false	P
rain	cool	normal	true	N
overcast	cool	normal	true	P
sunny	mild	high	false	N
sunny	cool	normal	false	P
rain	mild	normal	false	P
sunny	mild	normal	true	P
overcast	mild	high	true	P
overcast	hot	normal	false	P
rain	mild	high	true	N

Play-tennis Example: Estimating $P(x_i|C)$

Outlook	Temperature	Humidity	Windy	Class
sunny	hot	high	false	N
sunny	hot	high	true	N
overcast	hot	high	false	P
rain	mild	high	false	P
rain	cool	normal	false	P
rain	cool	normal	true	N
overcast	cool	normal	true	P
sunny	mild	high	false	N
sunny	cool	normal	false	P
rain	mild	normal	false	P
sunny	mild	normal	true	P
overcast	mild	high	true	P
overcast	hot	normal	false	P
rain	mild	high	true	N

P(p) = 9/14
P(n) = 5/14

outlook	
P(sunny|p) = 2/9	P(sunny|n) = 3/5
P(overcast|p) = 4/9	P(overcast|n) = 0
P(rain|p) = 3/9	P(rain|n) = 2/5
temperature	
P(hot|p) = 2/9	P(hot|n) = 2/5
P(mild|p) = 4/9	P(mild|n) = 2/5
P(cool|p) = 3/9	P(cool|n) = 1/5
humidity	
P(high|p) = 3/9	P(high|n) = 4/5
P(normal|p) = 6/9	P(normal|n) = 2/5
windy	
P(true|p) = 3/9	P(true|n) = 3/5
P(false|p) = 6/9	P(false|n) = 2/5

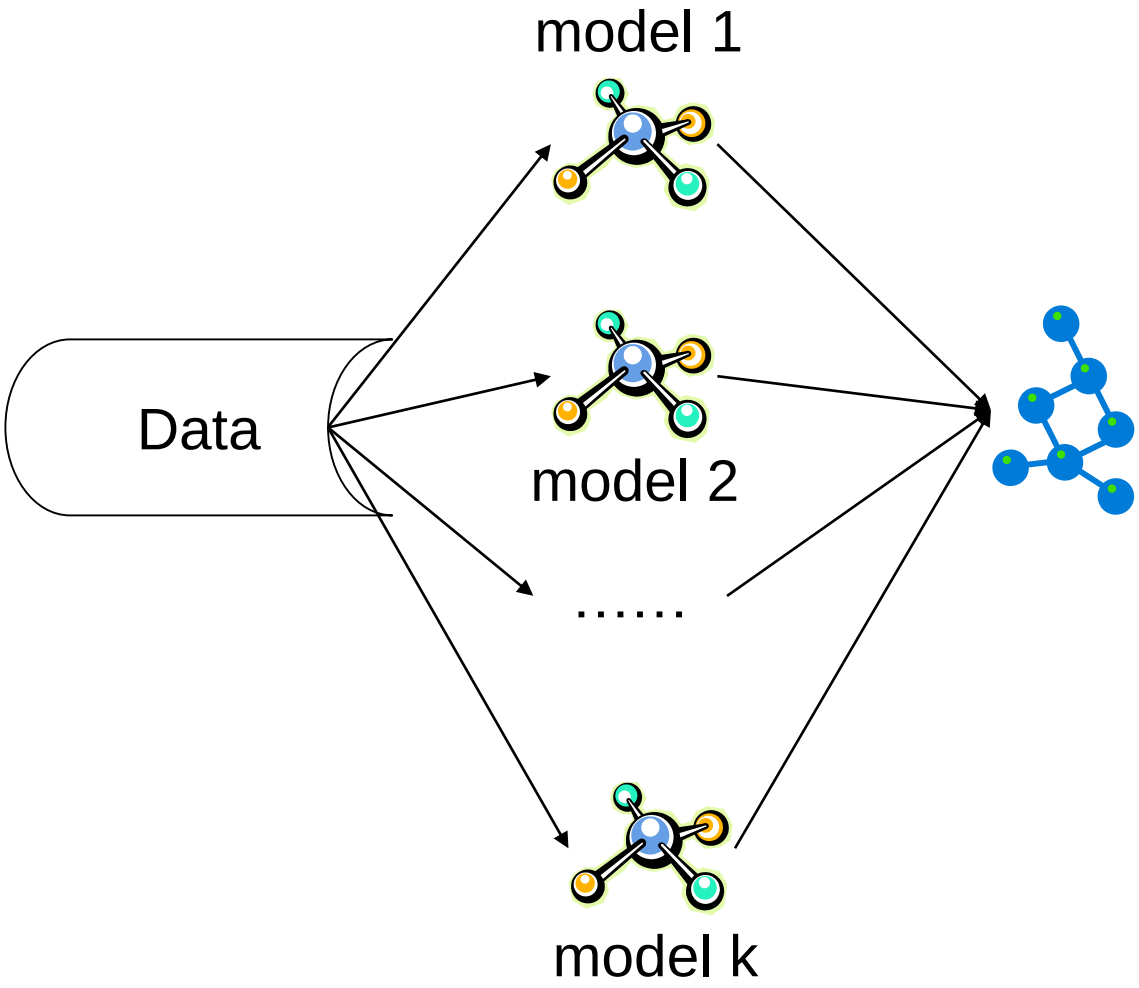
Play-tennis Example: Classifying X

- An unseen sample X = <rain, hot, high, false>
- P(X|p)·P(p) =
  P(rain|p)·P(hot|p)·P(high|p)·P(false|p)·P(p) =
  3/9·2/9·3/9·6/9·9/14 = 0.010582
- P(X|n)·P(n) =
  P(rain|n)·P(hot|n)·P(high|n)·P(false|n)·P(n) =
  2/5·2/5·4/5·2/5·5/14 = 0.018286
- Sample X is classified in class n (don't play)

Naïve Bayesian Classifier: Comments

- Advantages
  - Easy to implement
  - Good results obtained in most of the cases
- Disadvantages
  - Assumption: class conditional independence, therefore loss of accuracy
  - Practically, dependencies exist among variables
    - E.g., hospitals: patients: Profile: age, family history, etc.
    - Symptoms: fever, cough etc., Disease: lung cancer, diabetes, etc.
    - Dependencies among these cannot be modeled by Naïve Bayesian Classifier
- How to deal with these dependencies?
  - Bayesian Belief Networks

Ensemble

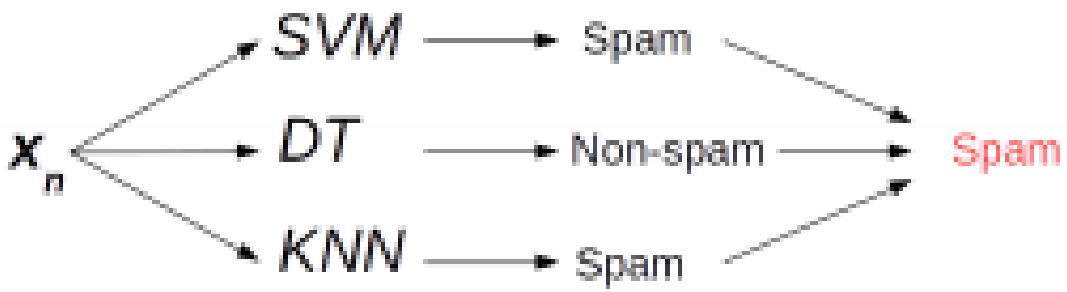


Combine multiple models into one!

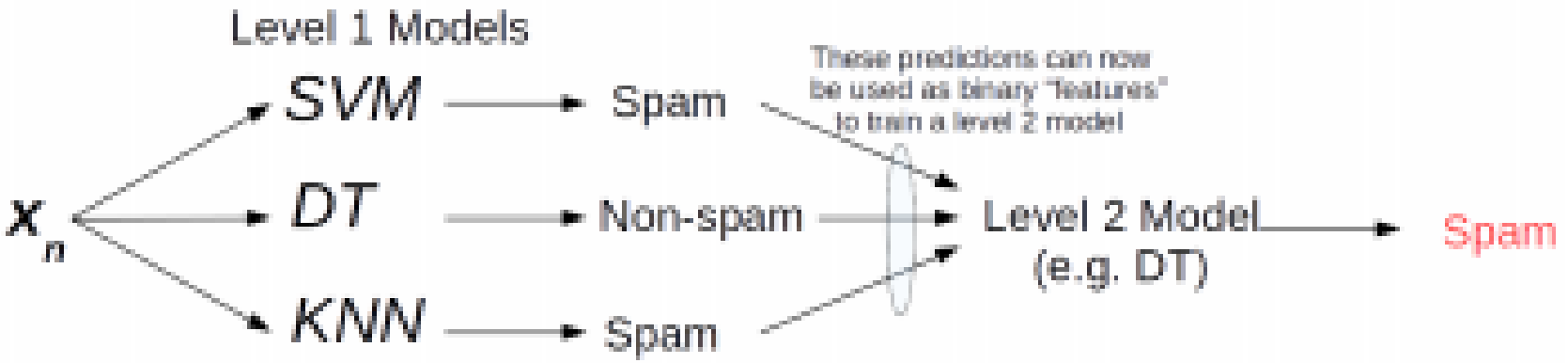
Applications: Classification, Clustering, Collaborative Filtering, Anomaly Detection……

Some simple ensembles

Voting or Averaging of predictions of multiple pre-trained models

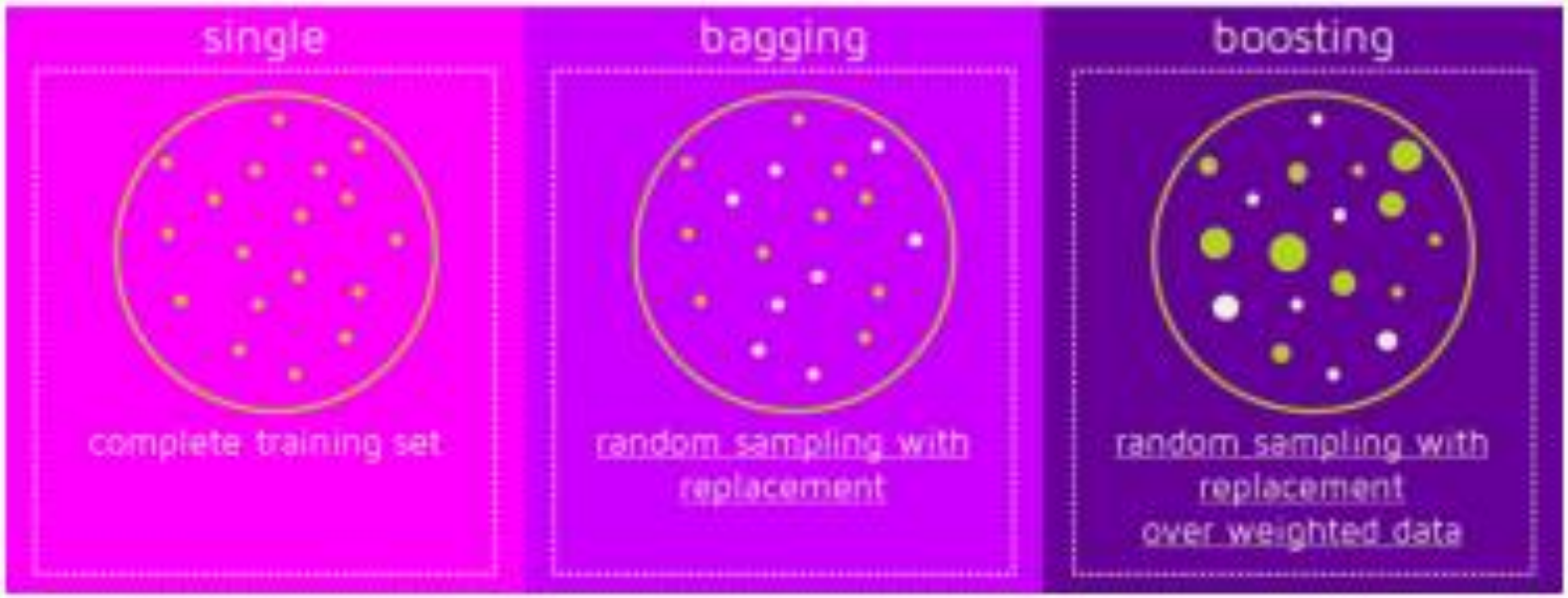


"Stacking": Use predictions of multiple models as "features" to train a new model and use the new model to make predictions on test data



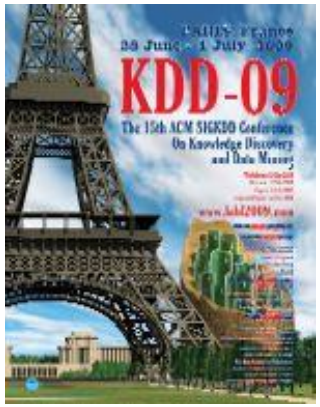
Another approach

- Instead of training different models on same data, train same model multiple times on different data sets, and “combine” these “different” models
- We can use some simple/weak model as the base model
- How do we get multiple training data sets (in practice, we only have one data set at training time)?



Stories of Success

- Million-dollar prize
  - Improve the baseline movie recommendation approach of Netflix by 10% in accuracy
  - The top submissions all combine several teams and algorithms as an ensemble



- Data mining competitions
  - Classification problems
  - Winning teams employ an ensemble of classifiers

Netflix Prize

- Supervised learning task
  - Training data is a set of users and ratings (1,2,3,4,5 stars) those users have given to movies.
  - Construct a classifier that given a user and an unrated movie, correctly classifies that movie as either 1, 2, 3, 4, or 5 stars
  - $1 million prize for a 10% improvement over Netflix's current movie recommender
- Competition
  - At first, single-model methods are developed, and performances are improved
  - However, improvements slowed down
  - Later, individuals and teams merged their results, and significant improvements are observed

Leaderboard

Rank	Team Name	Best Test Score	% Improvement	Best Submit Time
Grand Prize - RMSE = 0.8567 - Winning Team: BellKor's Pragmatic Chaos				
1	BellKor's Pragmatic Chaos	0.8567	10.06	2009-07-26 18:18:28
2	The Ensemble	0.8567	10.06	2009-07-26 18:38:22
3	Grand Prize Team	0.8582	9.90	2009-07-10 21:24:40
4	Opera Solutions and Vandelay United	0.8588	9.84	2009-07-10 01:12:31
5	Vandelay Industries !	0.8591	9.81	2009-07-10 00:32:20
6	PragmaticTheory	0.8594	9.77	2009-06-24 12:06:56
7	BellKor in BigChaos	0.8601	9.70	2009-05-13 08:14:09
8	Dace	0.8612	9.59	2009-07-24 17:18:43
9	Feeds2	0.8622	9.48	2009-07-12 13:11:51
Progress Prize 2008 - RMSE = 0.8627 - Winning Team: BellKor in BigChaos				
13	xiangliang	0.8642	9.27	2009-07-15 14:53:22
14	Gravity	0.8643	9.26	2009-04-22 18:31:32
15	Ces	0.8651	9.18	2009-06-21 19:24:53

"Our final solution (RMSE=0.8712) consists of blending 107 individual results. "

"Predictive accuracy is substantially improved when blending multiple predictors. Our experience is that most efforts should be concentrated in deriving substantially different approaches, rather than refining a single technique. "

Motivations

- Motivations of ensemble methods
  - Ensemble model improves accuracy and robustness over single model methods
  - Applications:
    - distributed computing
    - privacy-preserving applications
    - large-scale data with reusable models
    - multiple sources of data
  - Efficiency: a complex problem can be decomposed into multiple sub-problems that are easier to understand and solve (divide-and-conquer approach)

Relationship with Related Studies

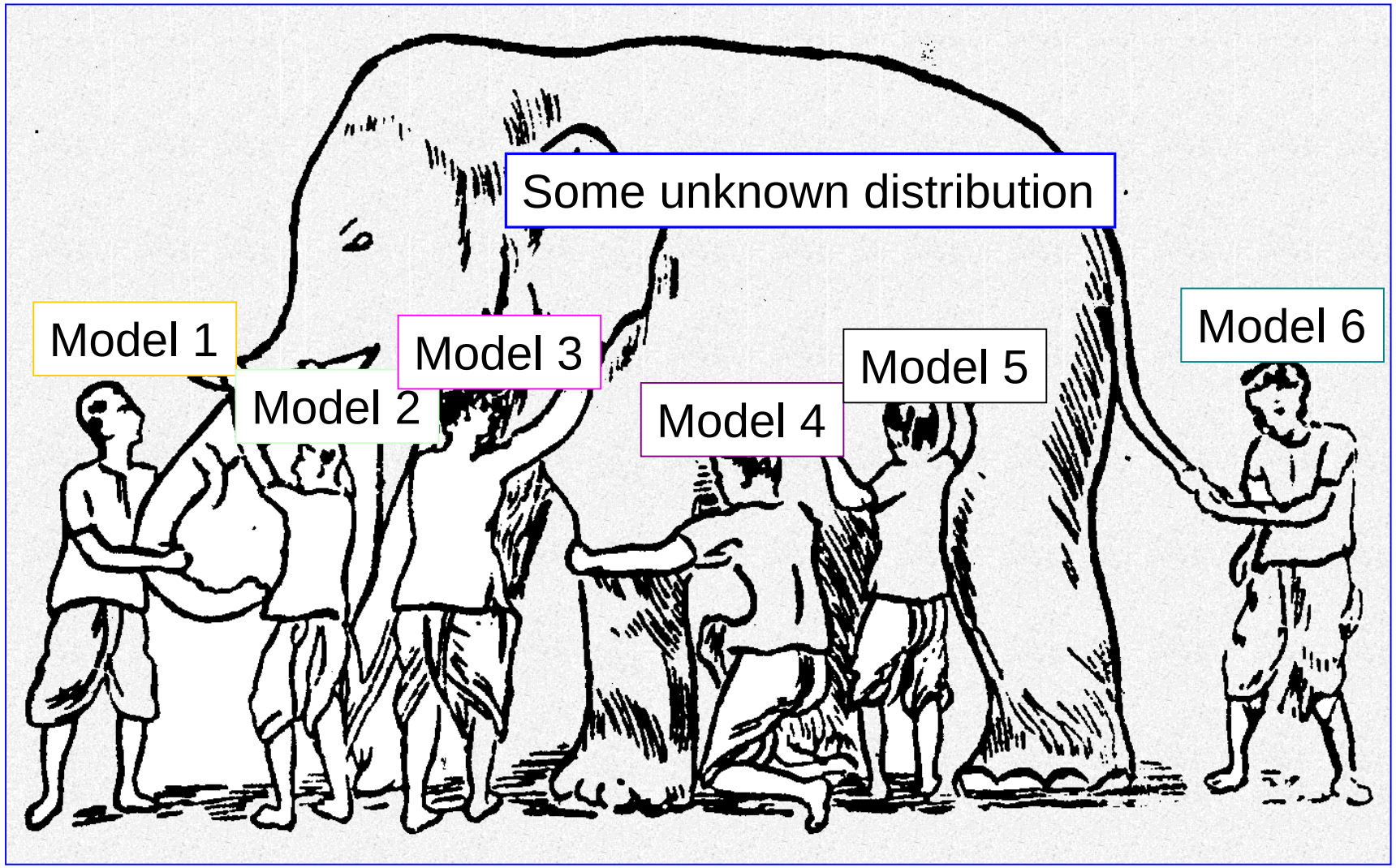
- Multi-task learning
  - Learn **multiple** tasks simultaneously
  - Ensemble methods: use multiple models to learn **one** task
- Data integration
  - Integrate raw data
  - Ensemble methods: integrate information at the **model** level
- Meta learning
  - **Learn** on meta-data (include base model output)
  - Ensemble methods: besides learn a joint model based on model output, we can also combine the output by **consensus**
- Non-redundant clustering
  - Give **multiple** non-redundant clustering solutions to users
  - Ensemble methods: give **one** solution to users which represents the consensus among all the base models

Why Ensemble Works? (1)

- Intuition
  - combining diverse, independent opinions in human decision-making as a protective mechanism (e.g. stock portfolio)
- Uncorrelated error reduction
  - Suppose we have 5 completely independent classifiers for majority voting
  - If accuracy is 70% for each
    - 10 (.7^3)(.3^2)+5(.7^4)(.3)+(.7^5)
    - **83.7% majority vote accuracy**
  - 101 such classifiers
    - **99.9% majority vote accuracy**

from T. Holloway, Introduction to Ensemble Learning, 2007.

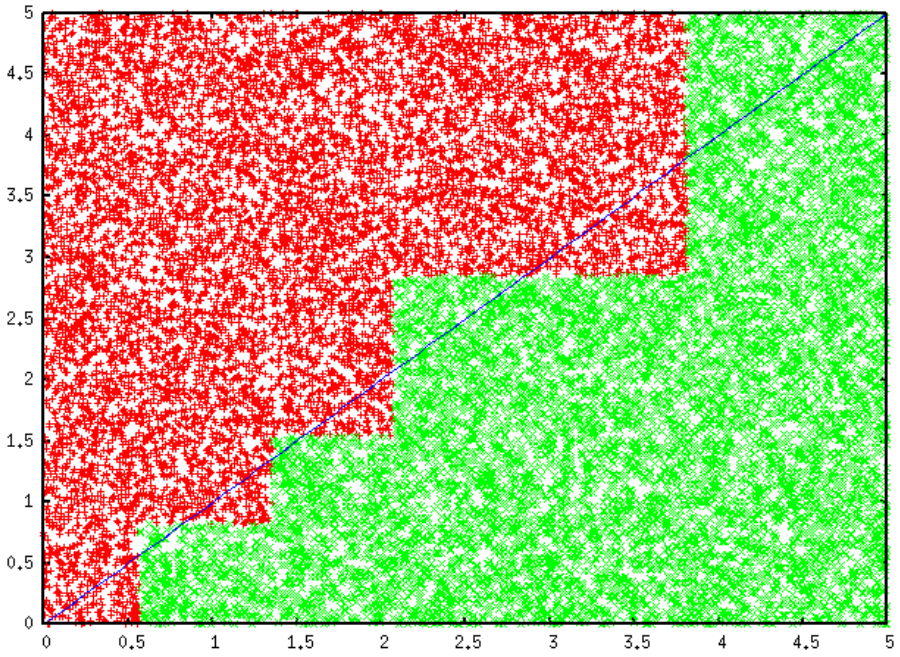
Why Ensemble Works? (2)



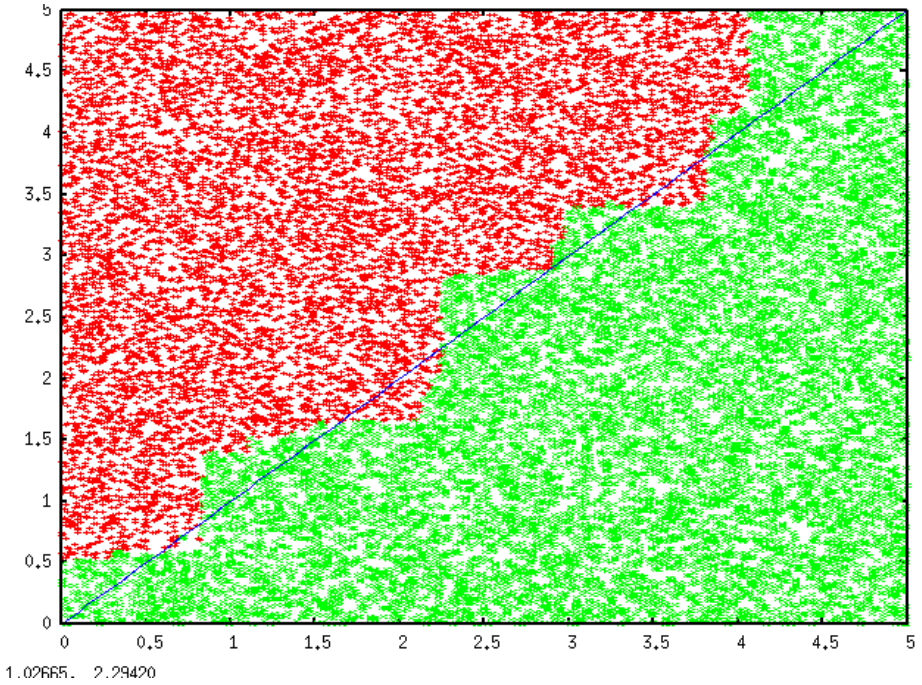
Ensemble gives the global picture!

Why Ensemble Works? (3)

- Overcome limitations of single hypothesis
  - The target function may not be implementable with individual classifiers, but may be approximated by model averaging

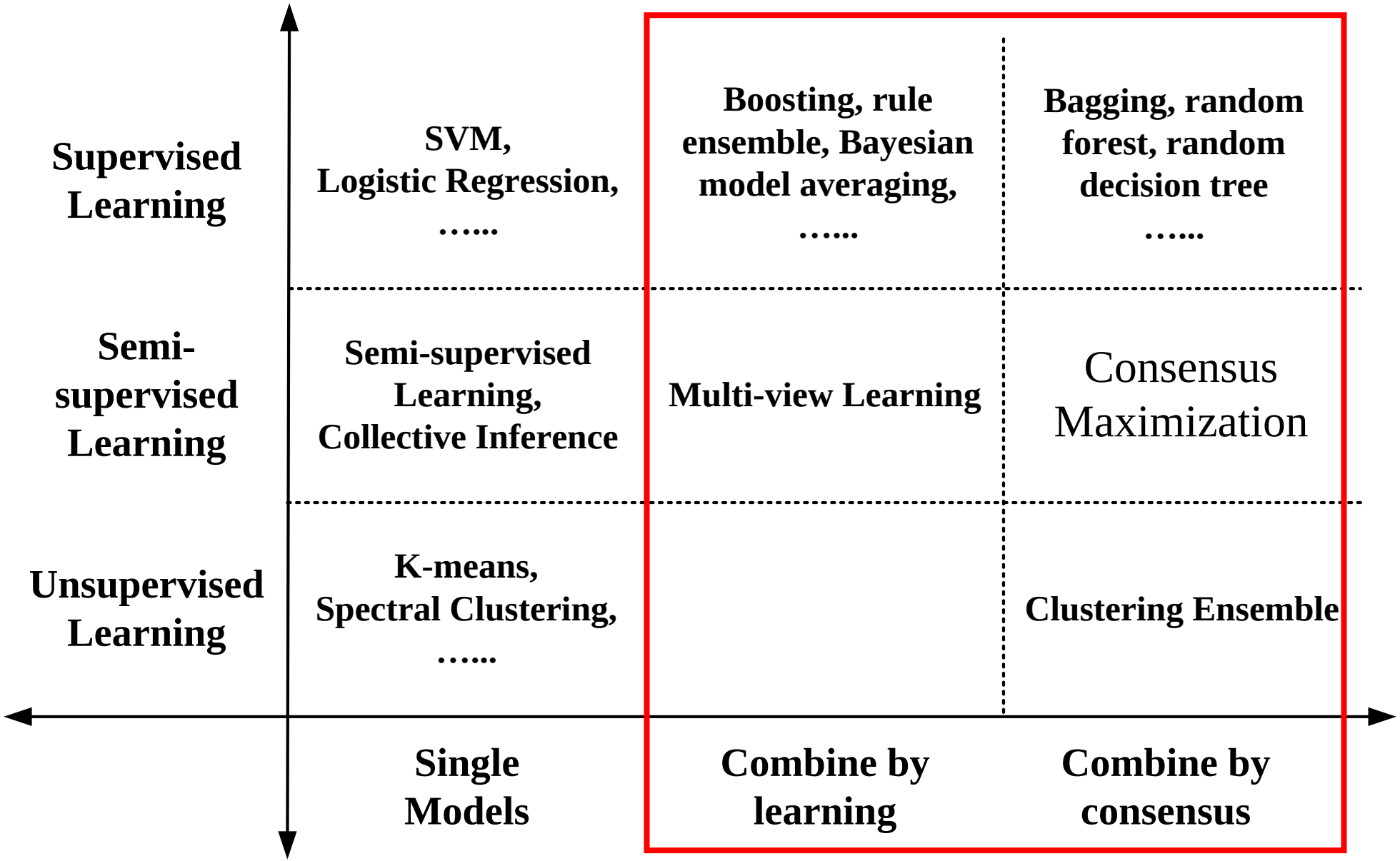


Decision Tree

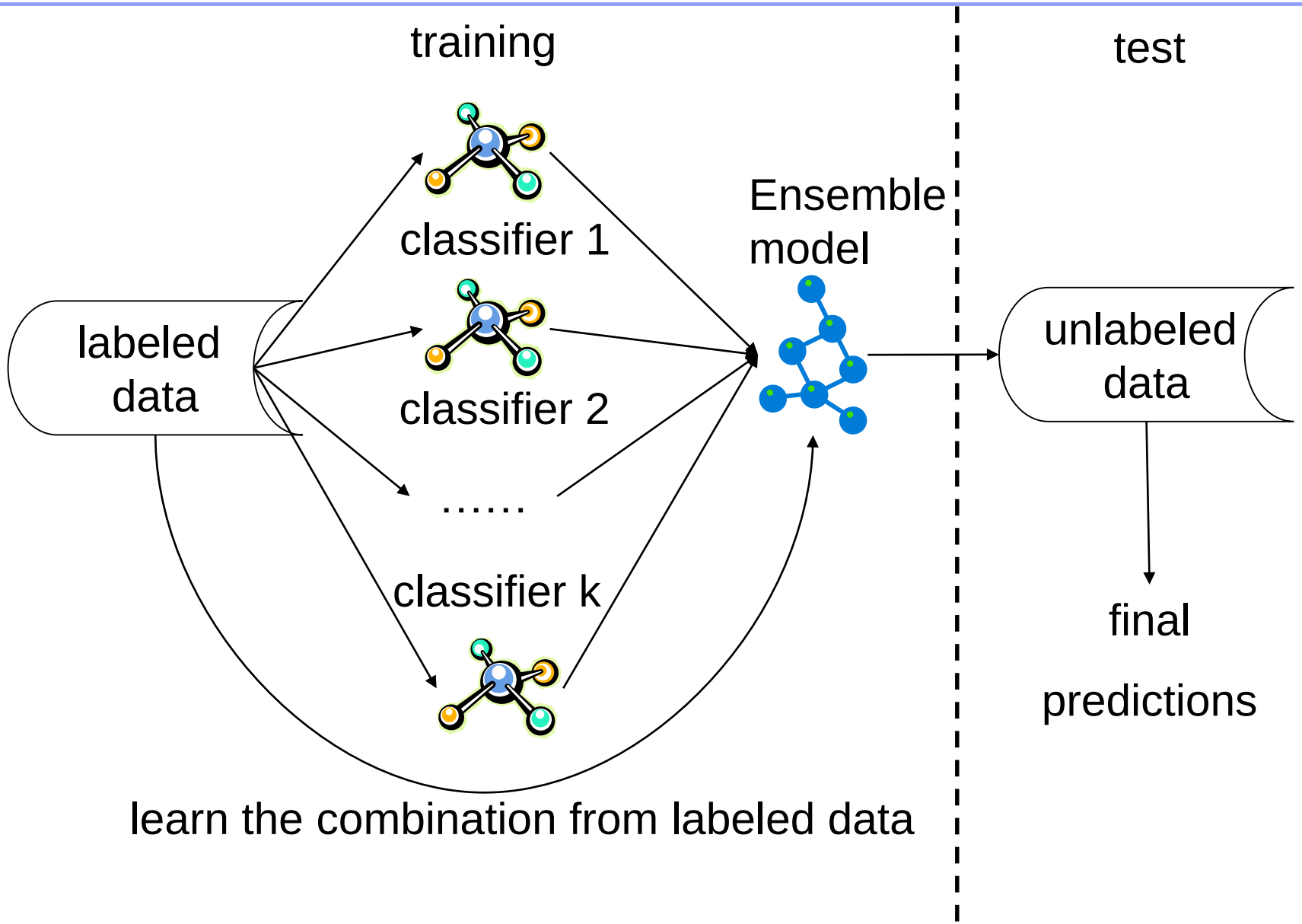


Model Averaging

Summary

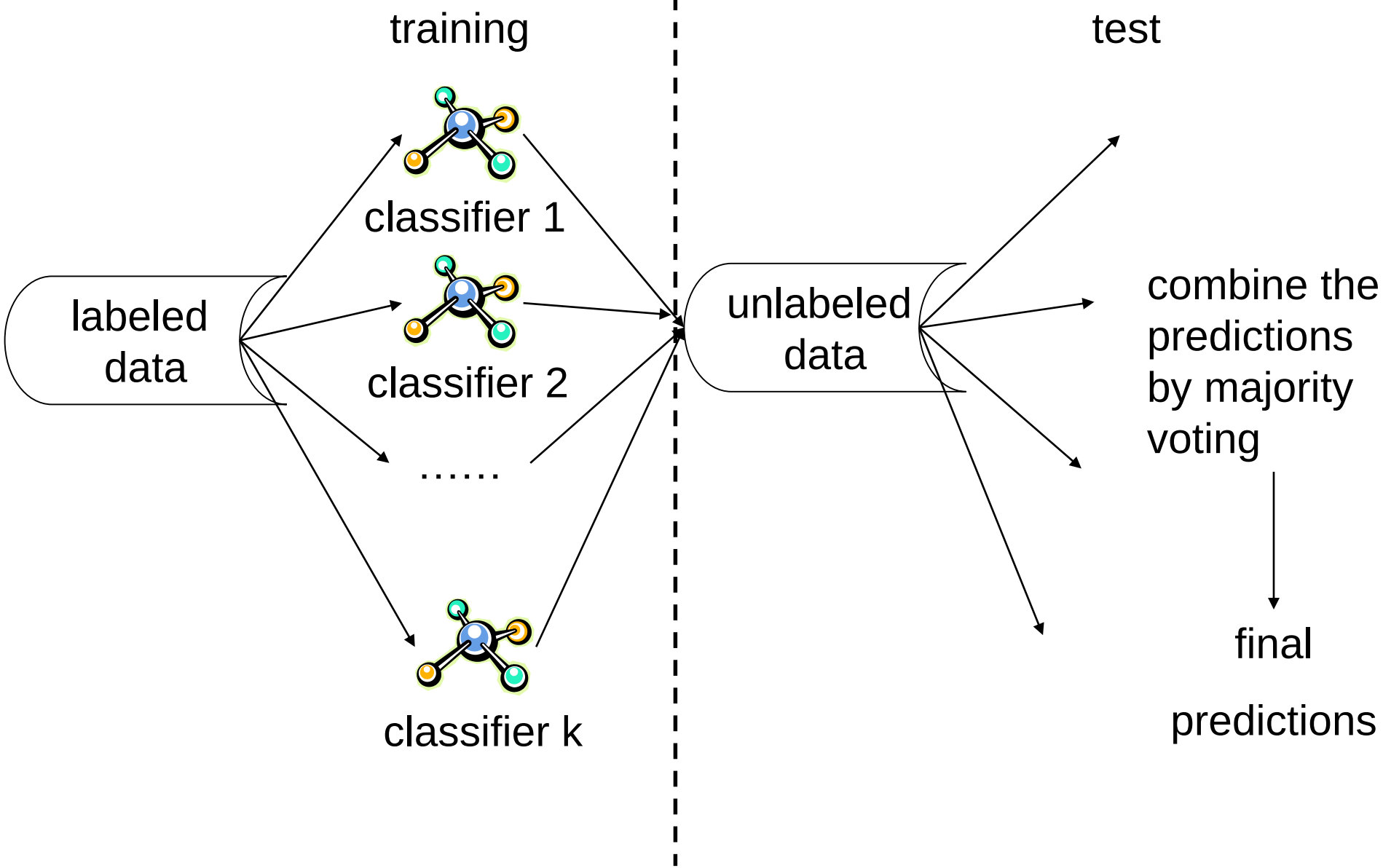

Supervised Learning
Semi-supervised Learning
Unsupervised Learning
SVM, Logistic Regression, ……
Boosting, rule ensemble, Bayesian model averaging, ……
Bagging, random forest, random decision tree ……
Semi-supervised Learning, Collective Inference
Multi-view Learning
Consensus Maximization
K-means, Spectral Clustering, ……
Clustering Ensemble
Single Models
Combine by learning
Combine by consensus

Ensemble of Classifiers—Learn to Combine



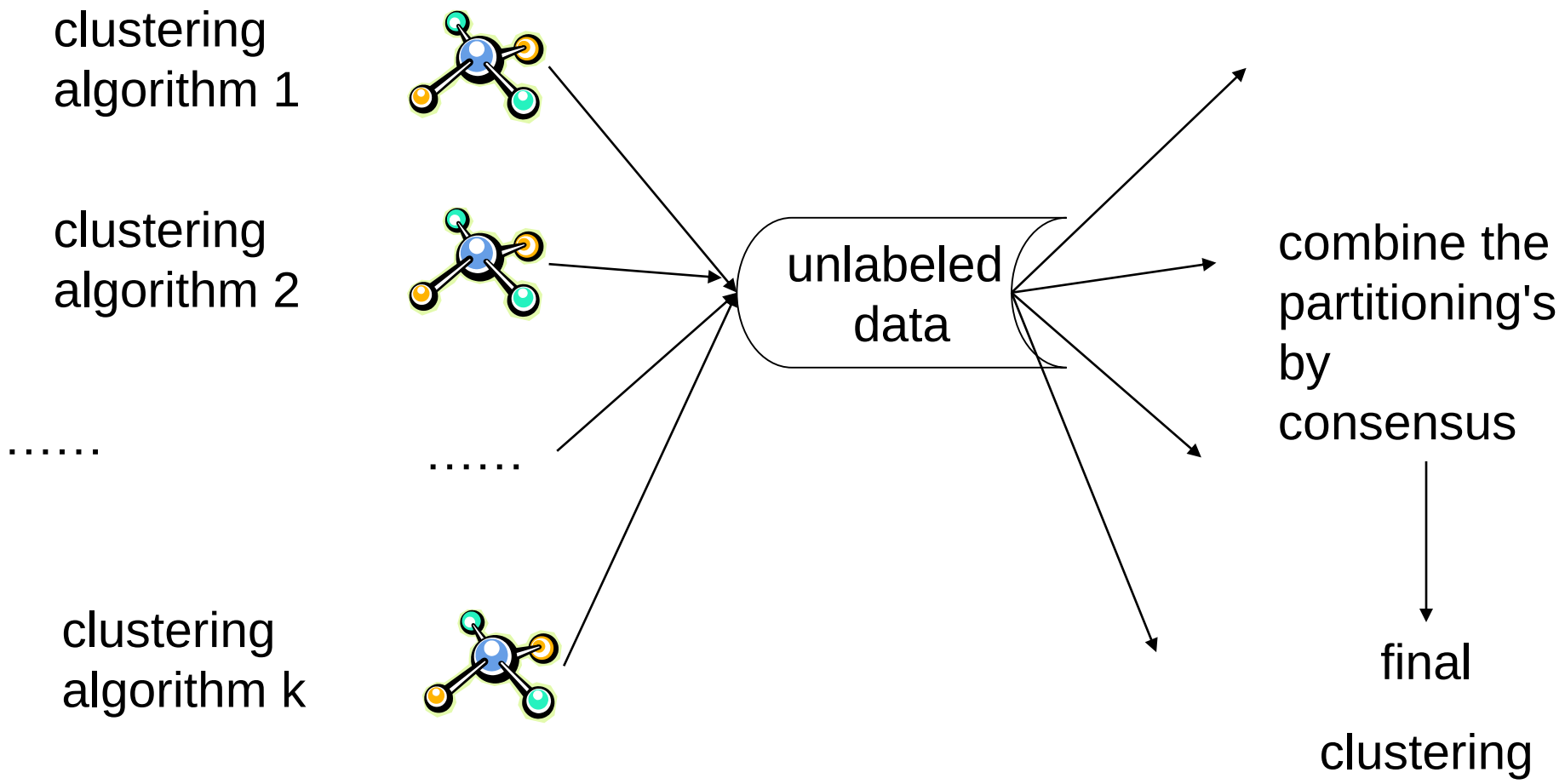
Algorithms: Boosting, Stacked Generalization, Rule Ensemble, Bayesian Model Averaging

Ensemble of Classifiers—Consensus



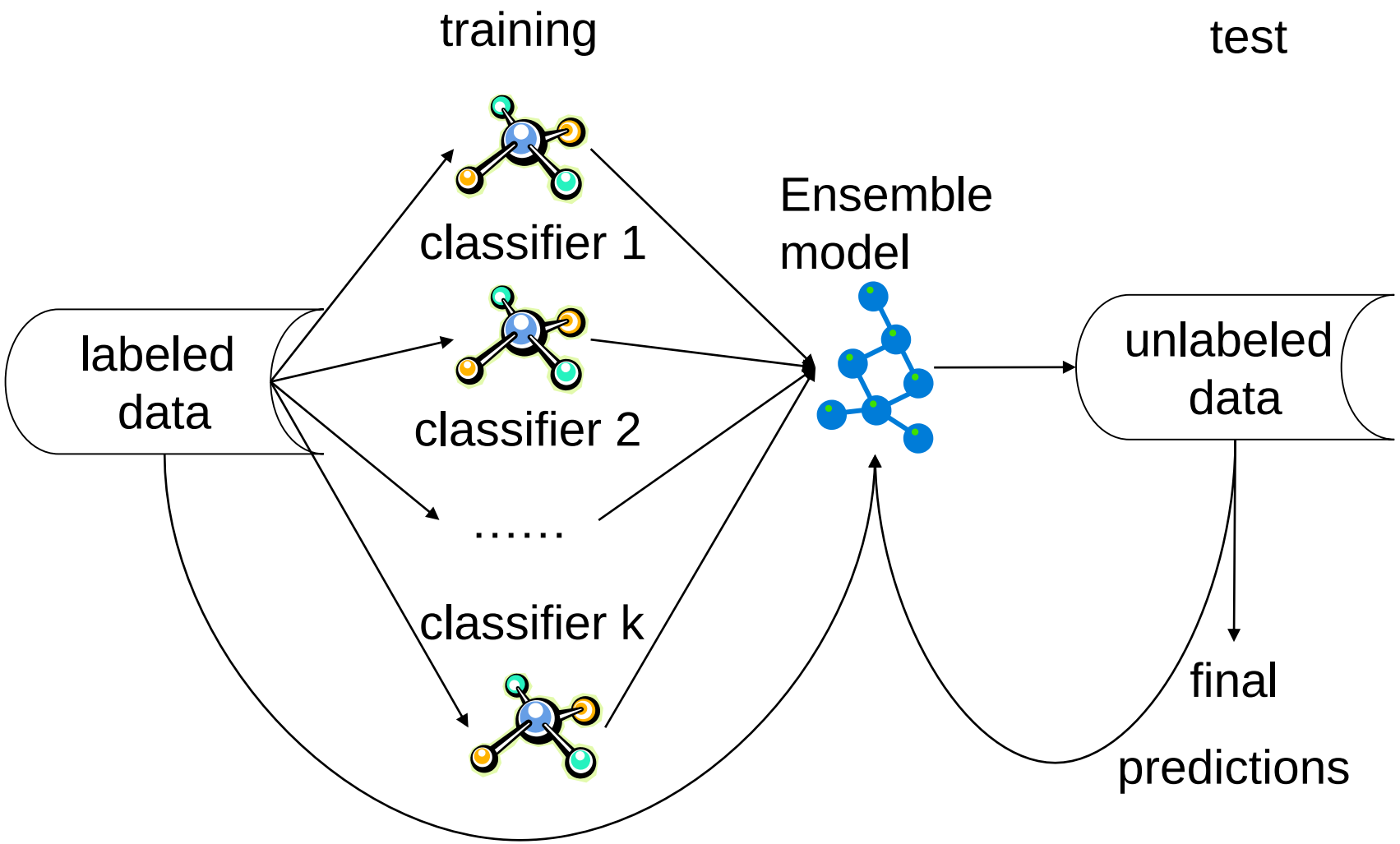
Algorithms: Bagging, Random Forest, Random Decision Tree, Model Averaging Of Probabilities

Clustering Ensemble—Consensus



Algorithms: Direct Approach, Object-based, Cluster-based, Object-cluster-based Approaches, Generative Models

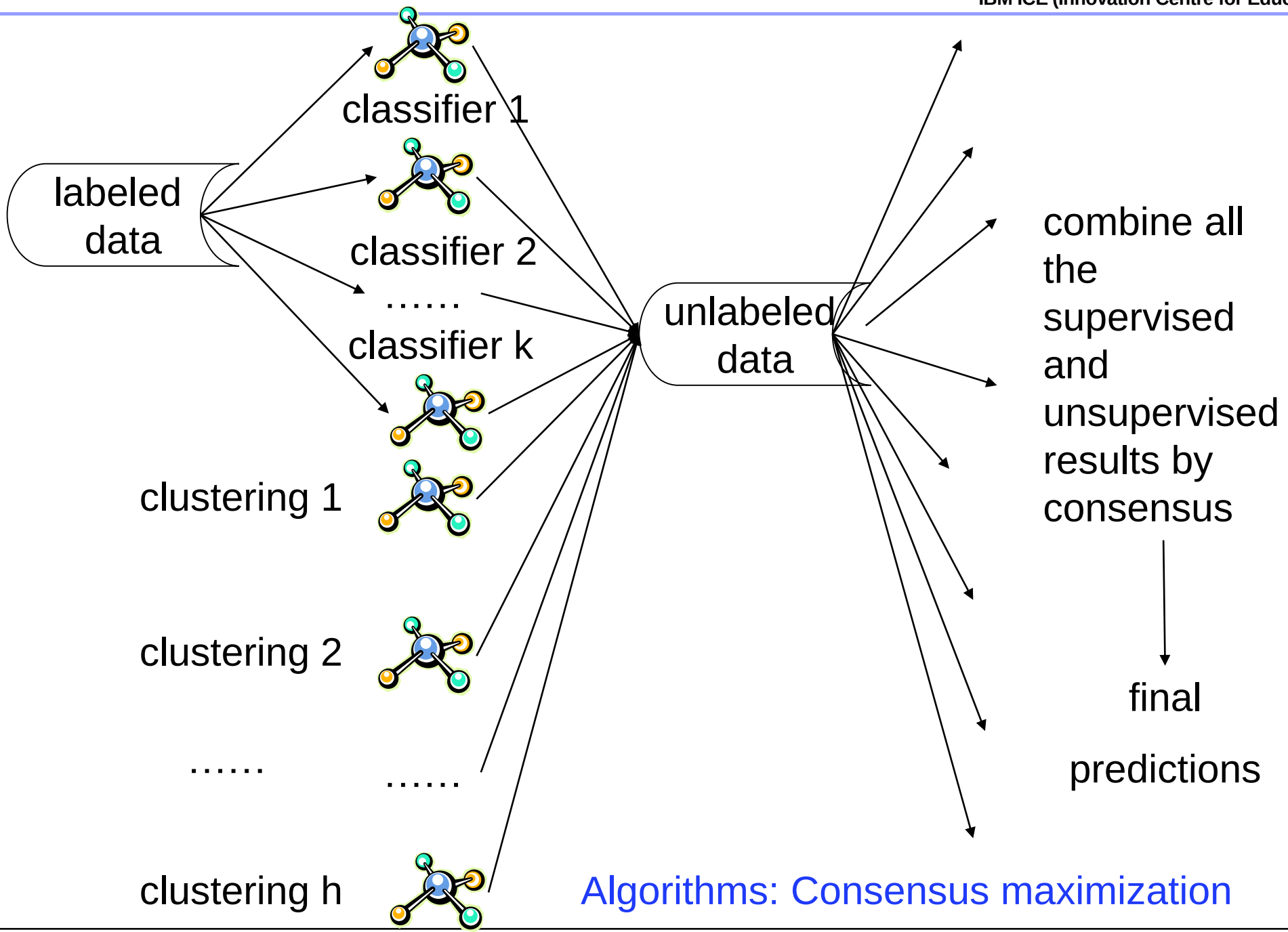
Semi-Supervised Ensemble—Learn to Combine



Learn the combination from both labeled and unlabeled data

Algorithms: Multi-view learning

Semi-supervised Ensemble—Consensus



Pros and Cons

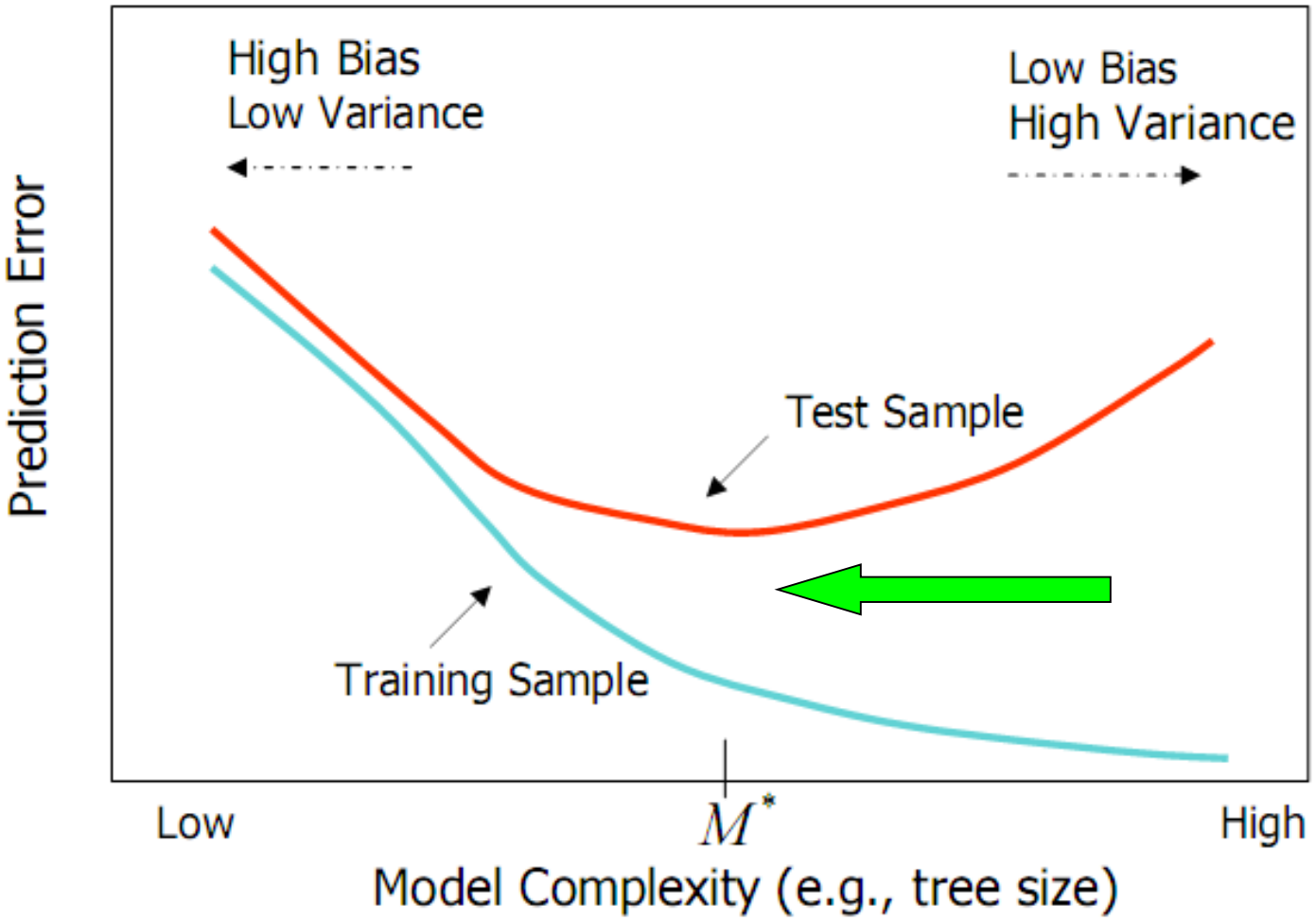
	Combine by learning	Combine by consensus
Pros	Get useful feedbacks from labeled data Can potentially improve accuracy	Do not need labeled data Can improve the generalization performance
Cons	Need to keep the labeled data to train the ensemble May overfit the labeled data Cannot work when no labels are available	No feedbacks from the labeled data Require the assumption that consensus is better

Supervised Ensemble Methods

- Problem
  - Given a data set $D=\{x_1,x_2,\ldots,x_n\}$ and their corresponding labels $L=\{l_1,l_2,\ldots,l_n\}$
  - An ensemble approach computes:
    - A set of classifiers $\{f_1,f_2,\ldots,f_k\}$, each of which maps data to a class label: $f_j(x)=l$
    - A combination of classifiers f^* which minimizes generalization error: $f^*(x)= w_1f_1(x)+ w_2f_2(x)+\ldots+ w_kf_k(x)$

Bias and Variance

- Ensemble methods
  - Combine learners to reduce variance



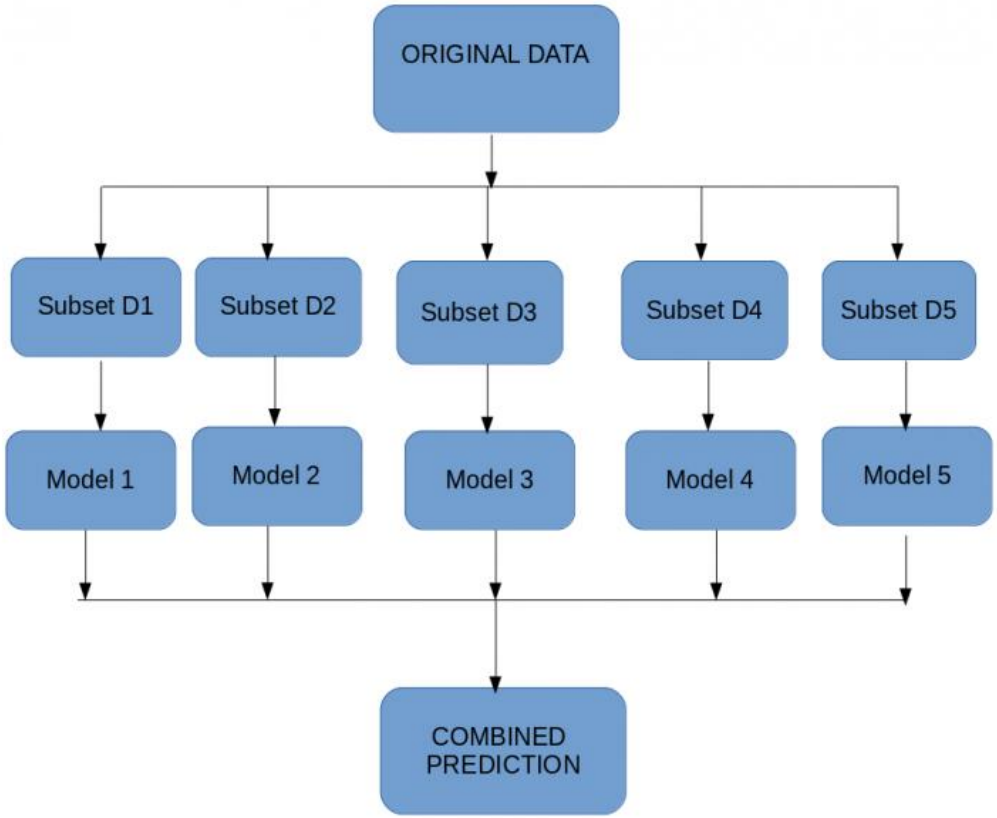
from Elder, John. From Trees to Forests and Rule Sets - A Unified Overview of Ensemble Methods. 2007.

Generating Base Classifiers

- Sampling training examples
  - Train k classifiers on k subsets drawn from the training set
- Using different learning models
  - Use all the training examples, but apply different learning algorithms
- Sampling features
  - Train k classifiers on k subsets of features drawn from the feature space
- Learning “randomly”
  - Introduce randomness into learning procedures

Bagging (1)

- Bootstrap
  - Sampling with replacement
  - Contains around 63.2% original records in each sample
- Bootstrap Aggregation
  - Train a classifier on each bootstrap sample
  - Use majority voting to determine the class label of ensemble classifier



Bagging

- Bagging stands for Bootstrap Aggregation
- Takes original data set D with N training examples
- Creates M copies: D_m, m=1…M
- Each D_m is generated from D by sampling with replacement
- Each data set D_m has the same number of examples as in data set D
- These data sets are reasonably different from each other (since only about 63% of the original examples appear in any of these data sets)
- Train models $h_{1...}h_M$ using $D_{....}D_M$, respectively
- Use an averaged model $h = 1/M \Sigma h_m$ m=1…M as the final model
- Useful for models with high variance and noisy data

Bagging (2)

Original Data:

x	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
y	1	1	1	-1	-1	-1	-1	1	1	1

Bootstrap samples and classifiers:

x	0.1	0.2	0.2	0.3	0.4	0.4	0.5	0.6	0.9	0.9
y	1	1	1	1	-1	-1	-1	-1	1	1

x	0.1	0.2	0.3	0.4	0.5	0.5	0.9	1	1	1
y	1	1	1	-1	-1	-1	1	1	1	1

x	0.1	0.2	0.3	0.4	0.4	0.5	0.7	0.7	0.8	0.9
y	1	1	1	-1	-1	-1	-1	-1	1	1

x	0.1	0.2	0.5	0.5	0.5	0.7	0.7	0.8	0.9	1
y	1	1	-1	-1	-1	-1	-1	1	1	1

Combine predictions by majority voting

From P. Tan et al. Introduction to Data Mining.

Bagging (3)

- Error Reduction
  - Under mean squared error, bagging reduces variance and leaves bias unchanged
  - Consider idealized bagging estimator: $\bar{f}(x) = E(\hat{f}_z(x))$
  - The error is

$$E[Y - \hat{f}_z(x)]^2 = E[Y - \bar{f}(x) + \bar{f}(x) - \hat{f}_z(x)]^2$$

$$= E[Y - \bar{f}(x)]^2 + E[\bar{f}(x) - \hat{f}_z(x)]^2 \geq E[Y - \bar{f}(x)]^2$$

  - Bagging usually decreases MSE

Boosting

- The basic idea
  - Take a weak learning algorithm
    - Only requirement: Should be slightly better than random
  - Turn it into an awesome one by making it focus on difficult cases
- Most boosting algorithms follow these steps:
  1. Train a weak model on some training data
  2. Compute the error of the model on each training example
  3. Give higher importance to examples on which the model made mistakes
  4. Re-train the model using \importance weighted" training examples
  5. Go back to step 2

Boosting (1)

- Principles
  - Boost a set of weak learners to a strong learner
  - Make records currently misclassified more important
- Example
  - Record 4 is hard to classify
  - Its weight is increased, therefore it is more likely to be chosen again in subsequent rounds

Original Data	1	2	3	4	5	6	7	8	9	10
Boosting (Round 1)	7	3	2	8	7	9	4	10	6	3
Boosting (Round 2)	5	4	9	4	2	5	1	7	4	2
Boosting (Round 3)	4	4	8	10	4	5	4	6	3	4

From P. Tan et al. Introduction to Data Mining.

Boosting (2)

- AdaBoost
  - Initially, set uniform weights on all the records
  - At each round
    - Create a bootstrap sample based on the weights
    - Train a classifier on the sample and apply it on the original training set
    - Records that are wrongly classified will have their weights increased
    - Records that are classified correctly will have their weights decreased
    - If the error rate is higher than 50%, start over
  - Final prediction is weighted average of all the classifiers with weight representing the training accuracy

Boosting (3)

- Determine the weight
  - For classifier *i*, its error is

$$\varepsilon_i = \frac{\sum_{j=1}^{N} w_j \delta(C_i(x_j) \neq y_j)}{\sum_{j=1}^{N} w_j}$$

  - The classifier's importance is represented as:

$$\alpha_i = \frac{1}{2} \ln\left(\frac{1-\varepsilon_i}{\varepsilon_i}\right)$$

  - The weight of each record is updated as:

$$w_j^{(i+1)} = \frac{w_j^{(i)} \exp\left(-\alpha_i y_j C_i(x_j)\right)}{Z^{(i)}}$$

  - Final combination:

$$C^*(x) = \arg\max_y \sum_{i=1}^{K} \alpha_i \delta\left(C_i(x) = y\right)$$

Adaboost Algorithm

- Given: Training data $(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_N, y_N)$ with $y_n \in \{-1, +1\}$, $\forall n$
- Initialize weight of each example $(\boldsymbol{x}_n, y_n)$: $D_1(n) = 1/N$, $\forall n$
- For round $t = 1 : T$
  - Learn a weak $h_t(\boldsymbol{x}) \rightarrow \{-1, +1\}$ using training data weighted as per D_t
  - Compute the weighted fraction of errors of h_t on this training data

$$\epsilon_t = \sum_{n=1}^{N} D_t(n) \mathbb{1}[h_t(\boldsymbol{x}_n) \neq y_n]$$

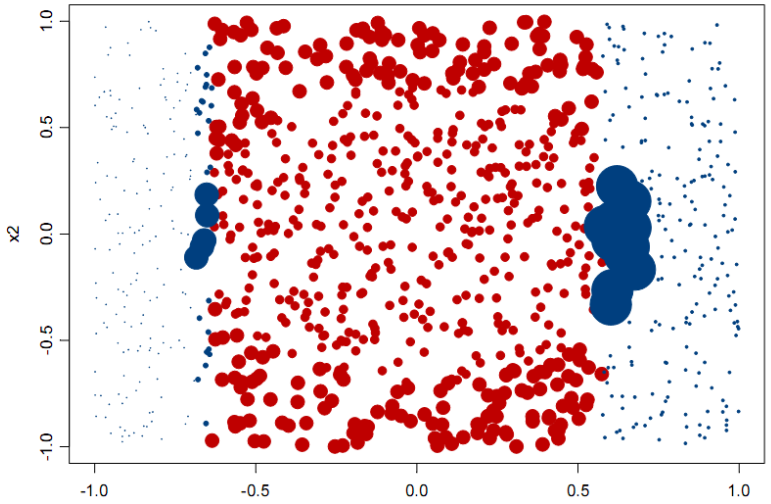
  - Set "importance" of h_t: $\alpha_t = \frac{1}{2} \log(\frac{1-\epsilon_t}{\epsilon_t})$(gets larger as ϵ_t gets smaller)
  - Update the weight of each example

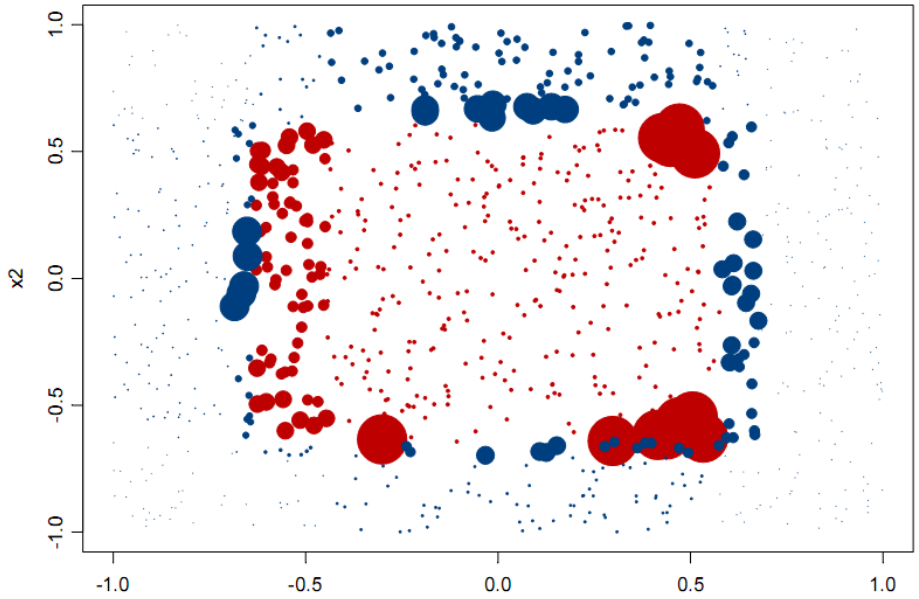
$$\begin{aligned} D_{t+1}(n) \quad &\propto \quad \begin{cases} D_t(n) \times \exp(-\alpha_t) & \text{if } h_t(\boldsymbol{x}_n) = y_n \quad \text{(correct prediction: decrease weight)} \\ D_t(n) \times \exp(\alpha_t) & \text{if } h_t(\boldsymbol{x}_n) \neq y_n \quad \text{(incorrect prediction: increase weight)} \end{cases} \\ &= \quad D_t(n) \exp(-\alpha_t y_n h_t(\boldsymbol{x}_n)) \end{aligned}$$

  - Normalize D_{t+1} so that it sums to 1: $D_{t+1}(n) = \frac{D_{t+1}(n)}{\sum_{m=1}^{N} D_{t+1}(m)}$
- Output the "boosted" final hypothesis $H(\boldsymbol{x}) = \text{sign}(\sum_{t=1}^{T} \alpha_t h_t(\boldsymbol{x}))$

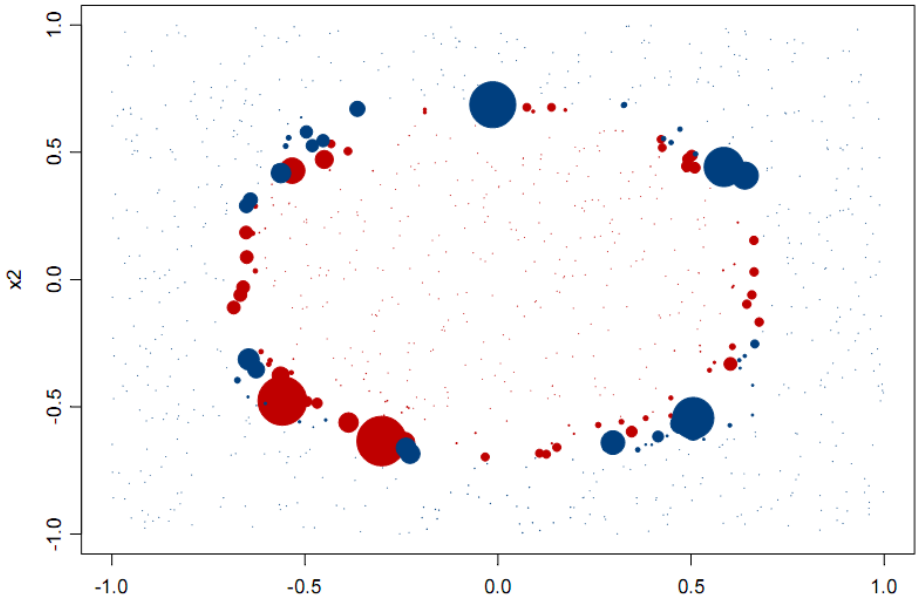
Example – 1

Classifications (colors) and Weights (size) after *1 iteration* Of AdaBoost





3 iterations



20 iterations

From Elder, John. From Trees to Forests and Rule Sets - A Unified Overview of Ensemble Methods. 2007.

Boosting (4)

- Explanation
  - Among the classifiers of the form:

$$f(x) = \sum_{i=1}^{K} \alpha_i C_i(x)$$

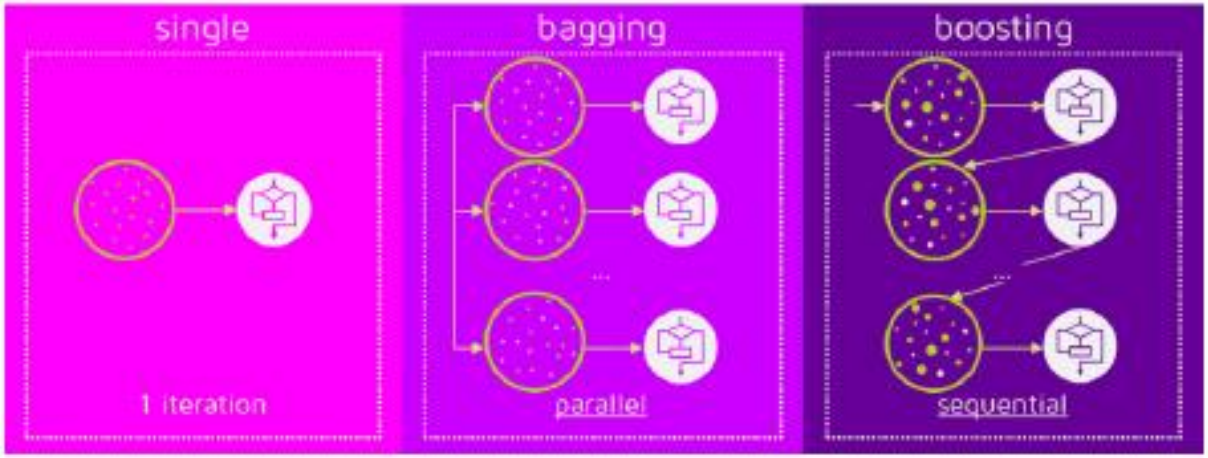
  - We seek to minimize the exponential loss function:

$$\sum_{j=1}^{N} \exp\left(-y_j f(x_j)\right)$$

  - Not robust in noisy settings

Bagging vs Boosting

- No clear winner; usually depends on the data
- Bagging is computationally more efficient than boosting (note that bagging can train the M models in parallel, boosting can't)



- Both reduce variance (and over-fitting) by combining different models
  - The resulting model has higher stability as compared to the individual ones
- Bagging usually can't reduce the bias, boosting can (note that in boosting, the training error
- steadily decreases)
- Bagging usually performs better than boosting if we don't have a high bias and only want to reduce variance (i.e., if we are over-fitting)

Random Forests (1)

- Algorithm
  - Choose *T*—number of trees to grow
  - Choose *m*<*M* (M is the number of total features) — number of features used to calculate the best split at each node (typically 20%)
  - For each tree
    - Choose a training set by choosing *N* times (*N* is the number of training examples) with replacement from the training set
    - For each node, randomly choose *m* features and calculate the best split
    - Fully grown and not pruned
  - Use majority voting among all the trees

Random Forests (2)

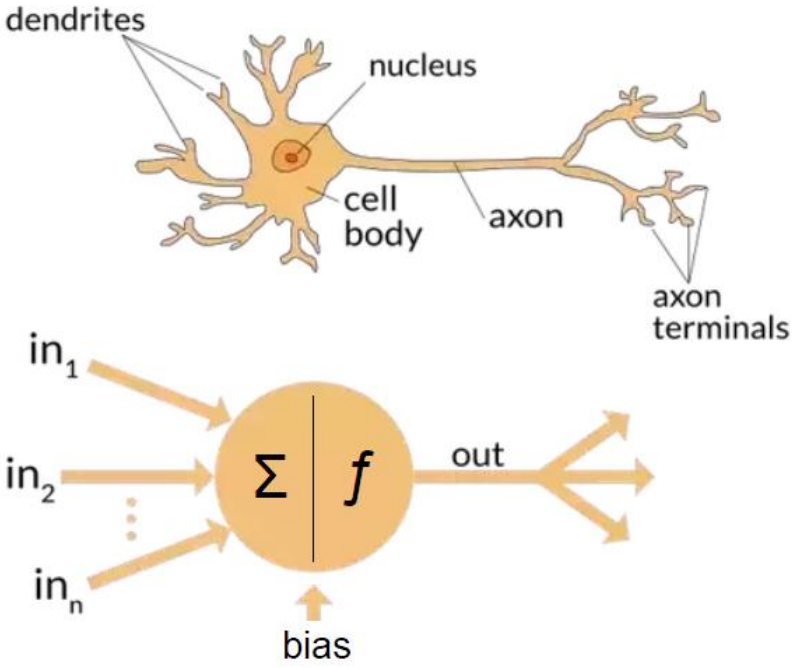
- Discussions
  - Bagging + random features
  - Improve accuracy
    - Incorporate more diversity and reduce variances
  - Improve efficiency
    - Searching among subsets of features is much faster than searching among the complete set

Neural Networks – Introduction

- Neural network learning methods provide a robust approach to approximating real-valued, discrete-valued, and vector-valued target functions.
- For certain types of problems, such as learning to interpret complex real-world sensor data, artificial neural networks are among the most effective learning methods currently known.
- For example, the Back-propagation algorithm described in this module has proven surprisingly successful in many practical problems such as learning to recognize handwritten characters, learning to recognize spoken words and learning to recognize faces.
- The study of artificial neural networks (ANNs) has been inspired in part by the observation that biological learning systems are built of very complex webs of interconnected **neurons**.
- In rough analogy, artificial neural networks are built out of a densely interconnected set of simple units, where each unit takes a number of real-valued inputs (possibly the outputs of other units) and produces a single real-valued output (which may become the input to many other units).

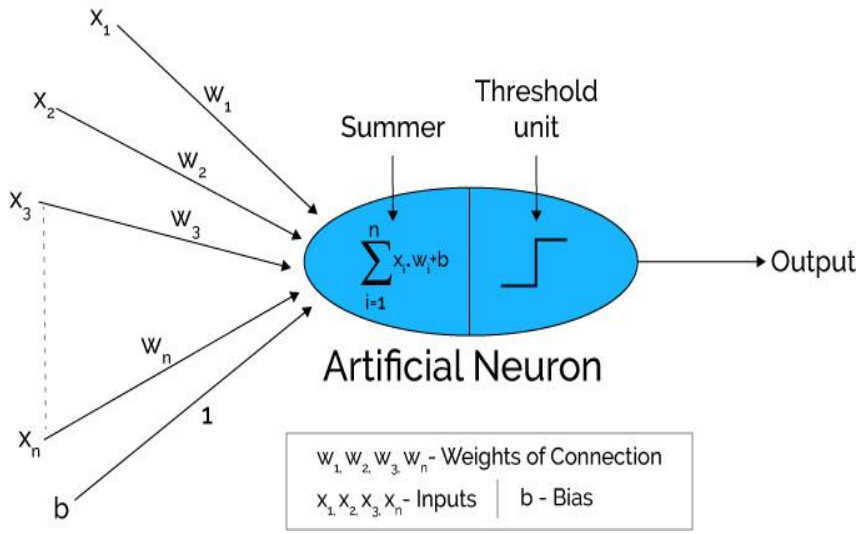
Artificial Neural Network

- It is a computational model inspired by the way biological neural networks in the human brain process information
- It has a lot of excitement and breakthrough in
  - Machine Learning research and industry
  - Speech recognition
  - Computer vision
  - Text processing
- Used for regression and classification problems
- It combines hundreds of algorithms and their variants



Appropriate Problems for Neural Network Learning

- ANN learning is well-suited to problems in which the training data corresponds to noisy, complex sensor data, such as inputs from cameras and microphones.
- It is also applicable to problems for which more symbolic representations are often used, such as the decision tree learning tasks.
- In these cases, ANN and decision tree learning often produce results of comparable accuracy.
- The back-propagation algorithm is the most commonly used ANN learning technique.

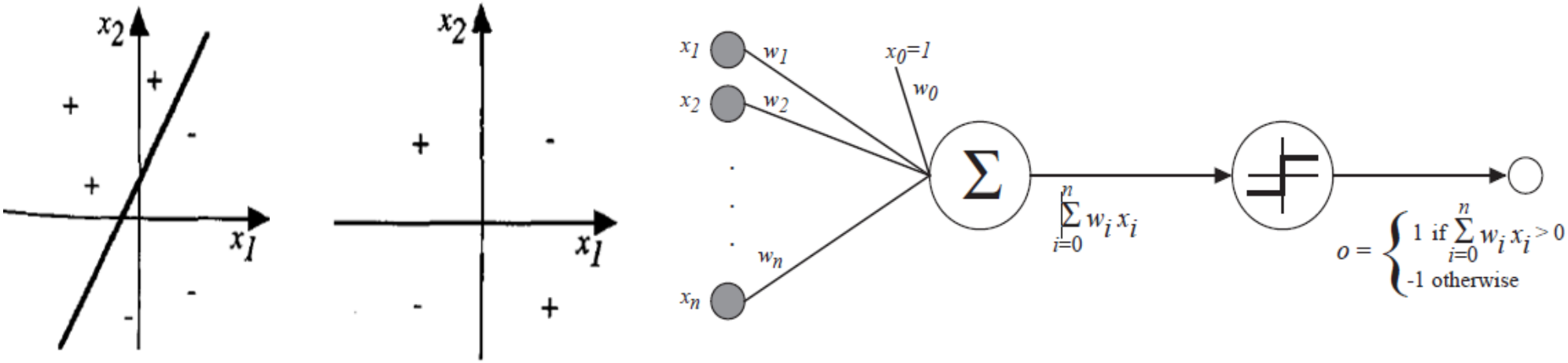


Characteristics of the Problems

- Instances are represented by many attribute-value pairs.
- The target function output may be discrete-valued, real-valued, or a vector of several real- or discrete-valued attributes.
- The training examples may contain errors.
- Long training times are acceptable
- Fast evaluation of the learned target function may be required.
- The ability of humans to understand the learned target function is not important.

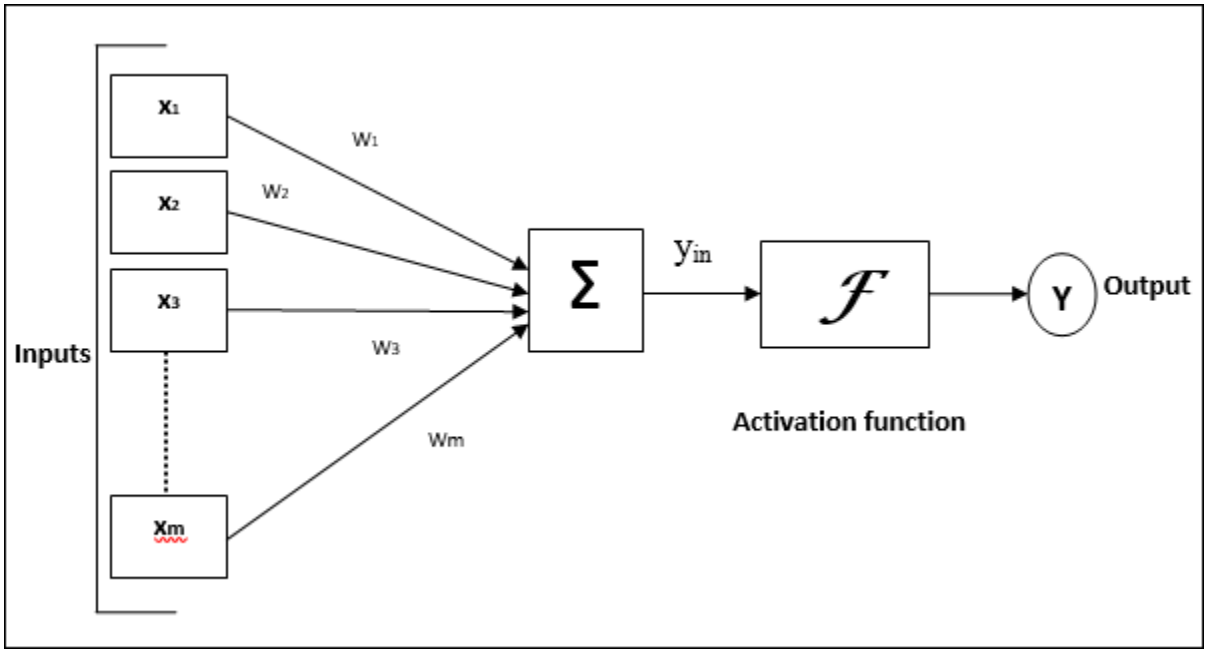
Basic understanding of Neural Networks

- Perceptron is the basic operational unit of artificial neural networks. It employs supervised learning rule and is able to classify the data into two classes.
- Perceptron thus has the following three basic elements −
  - Links − It would have a set of connection links, which carries a weight including a bias always having weight 1.
  - Adder − It adds the input after they are multiplied with their respective weights.
  - Activation function − It limits the output of neuron. The most basic activation function is a Heaviside step function that has two possible outputs. This function returns 1, if the input is positive, and 0 for any negative input.
- We can view the perceptron as representing a hyperplane decision surface in the n-dimensional space of instances (i.e., points). The perceptron outputs a 1 for instances lying on one side of the hyperplane and outputs a -1 for instances lying on the other side, as illustrated in Figure given below.



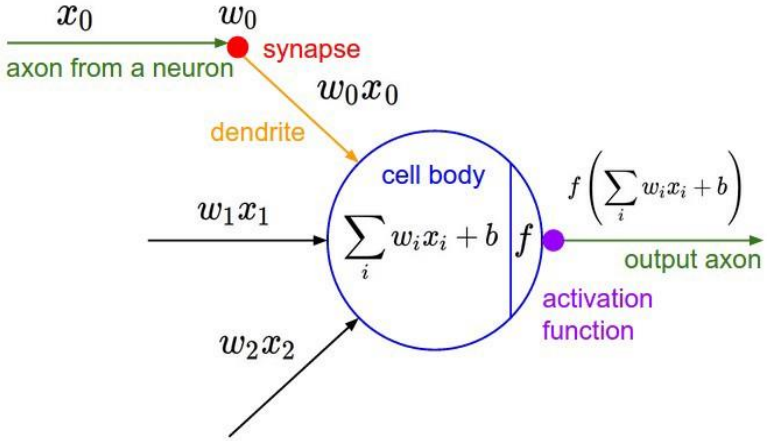
A Single Neuron (1 of 2)

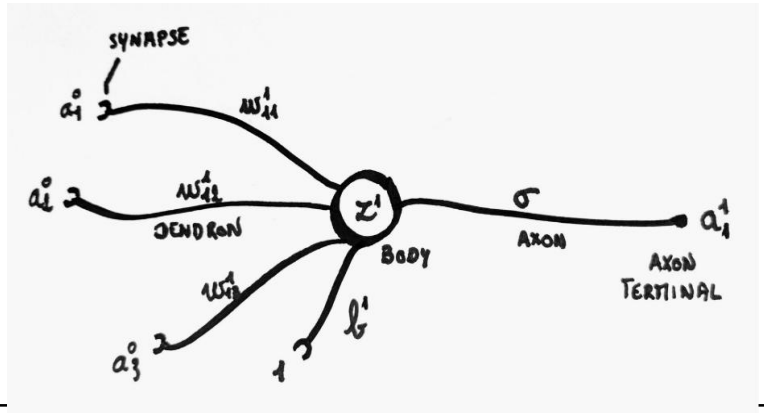
- The basic unit of computation in a neural network is the neuron, often called a node or unit
- Receives input from some other nodes, or from an external source and computes an output
- Each input has an associated weight (w) which is assigned on the basis of its relative importance to other inputs
- The node applies a function f to the weighted sum of its inputs as shown below



A Single Neuron (2 of 2)

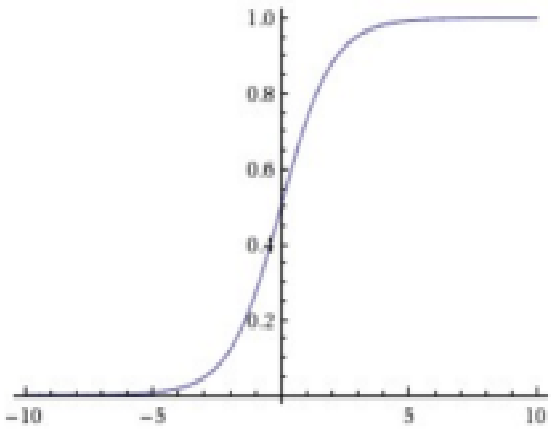
- The output Y from the neuron is computed
- The function f is an Activation Function
- The purpose of the activation function is to introduce non-linearity into the output of a neuron
- This is important because most real world data is non-linear and we want neurons to learn those non-linear representations
- Every activation function (non-linearity) takes a single number and performs a certain fixed mathematical operation on it
- There are several activation functions you may encounter in practice
  - Types of activations (Sigmoid, ReLU, Tanh, softmax)



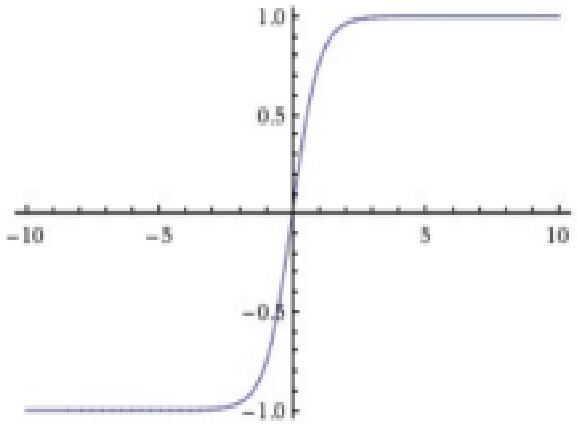


Activation Functions

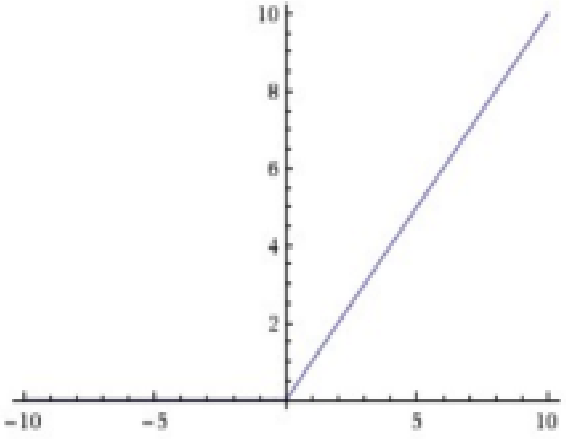
- Sigmoid: Takes a real-valued input and squashes it to range between [0,1]
- tanh: Takes a real-valued input and squashes it to the range [-1, 1]
- ReLU: It stands for Rectified Linear Unit
- It takes a real-valued input and thresholds it at zero (replaces negative values with zero)



Sigmoid



tanh



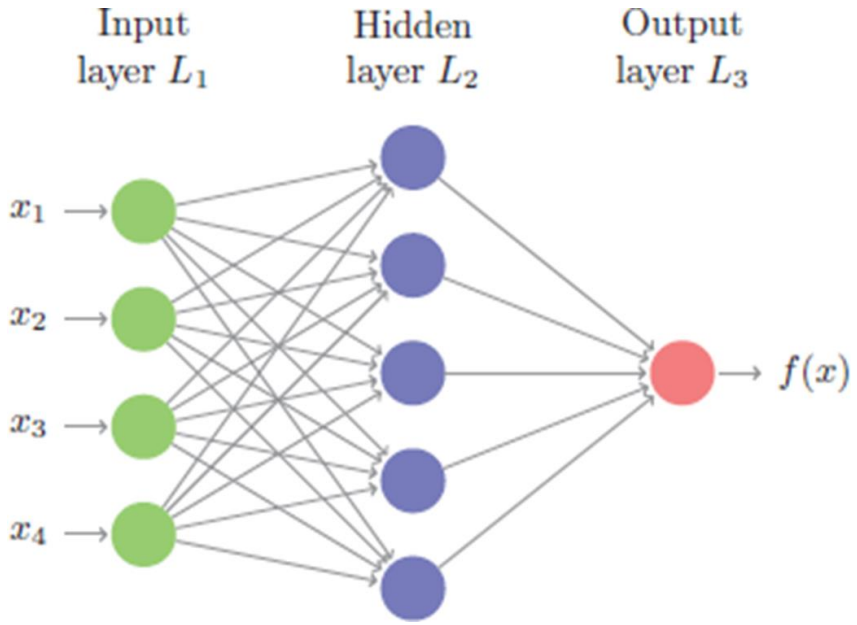
ReLU

Architectures of neural networks

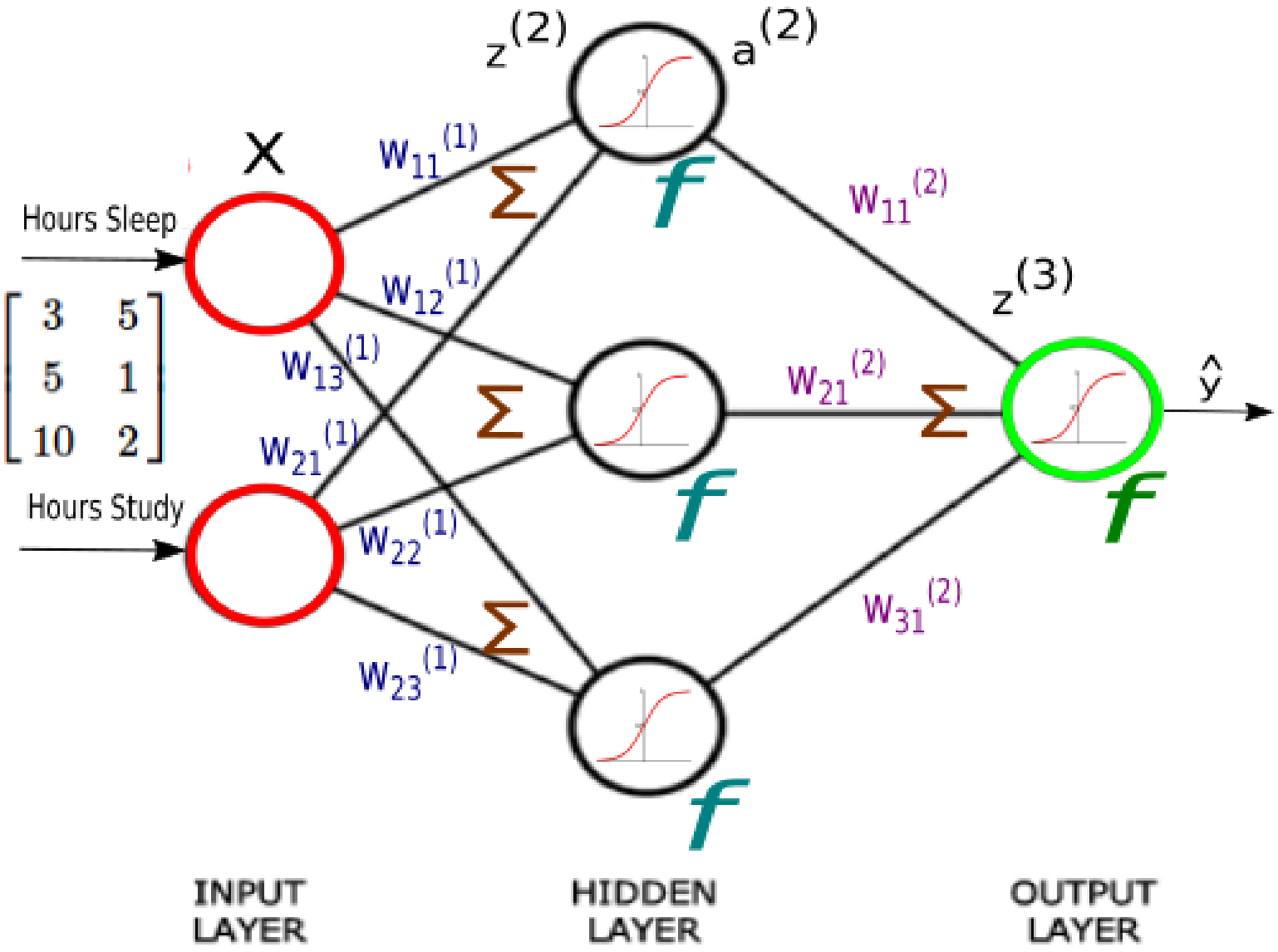
- The main architectures of artificial neural networks, considering the neuron disposition, as well as how they are interconnected and how its layers are composed, classified as follows:
  - single-layer feedforward network,
  - multilayer feedforward networks,
  - recurrent networks and
  - mesh networks.

Feedforward Neural Network

- The simplest type of artificial neural network
- Contains multiple neurons (nodes) arranged in layers
- Nodes from adjacent layers have connections or edges between them
- All these connections have associated weights
- A feedforward neural network consists of three types of layers:



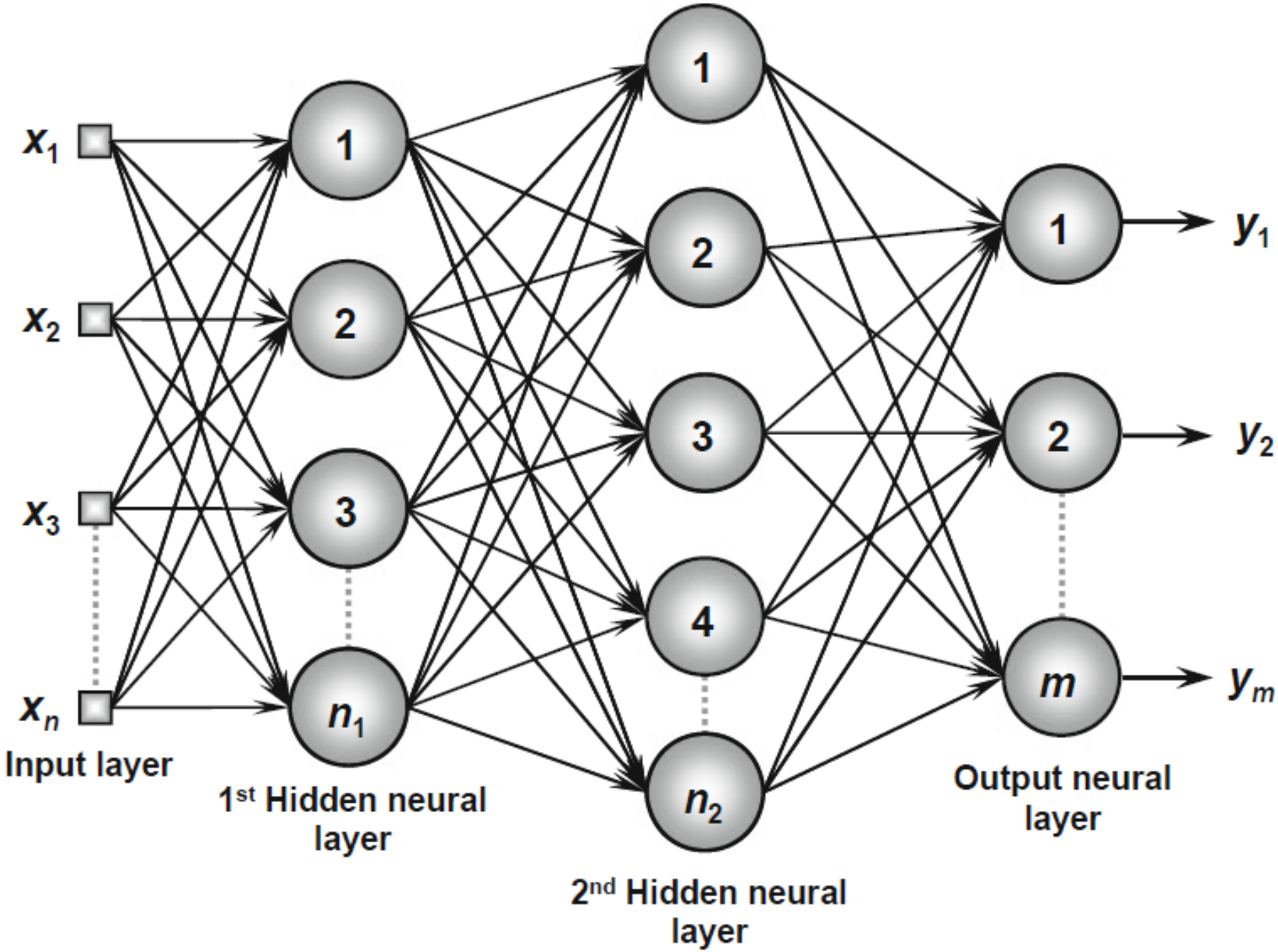
Single-Layer Feedforward Architecture



Types of nodes

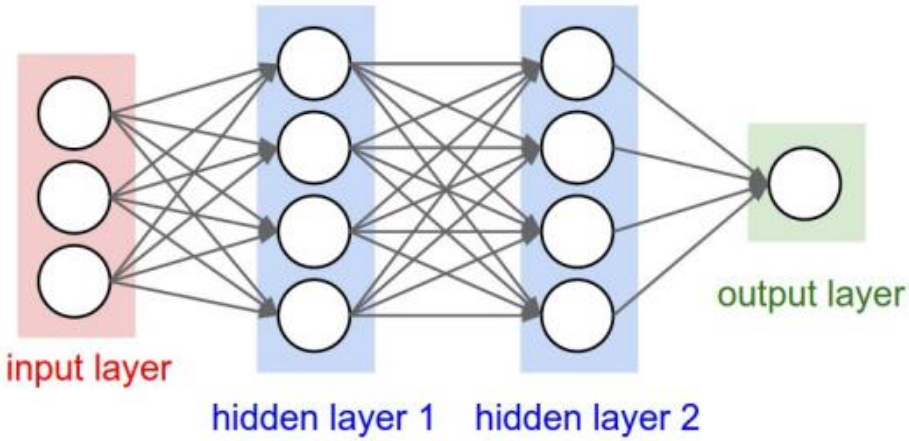
- Input Nodes:
  - Provide information from the outside world to the network, hence input layer
  - No computation is performed in any of the Input nodes
  - They just pass on the information to the hidden nodes
- Hidden Nodes:
  - Those nodes have no direct connection with the outside world, hence the name hidden nodes
  - They perform computations and transfer information from the input nodes to the output nodes
  - A collection of hidden nodes forms a hidden Layer
  - The feedforward network have a single input layer and a single output layer, and zero or multiple hidden Layers
- Output Nodes:
  - The Output nodes are collectively referred to as the output layer and are responsible for computations and transferring information from the network to the outside world (classification/regression)
- In a feedforward network, the information moves in only one direction, i.e, Input nodes → Hidden nodes → output nodes. There are no cycles (loops) in the network, unlike to other networks such as Recurrent Neural Networks (RNN).

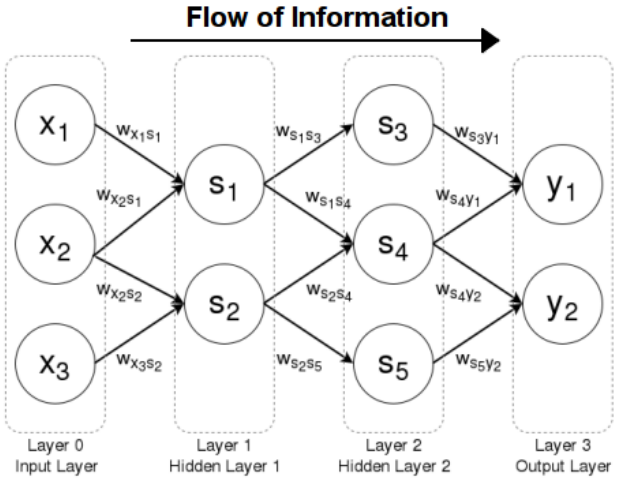
Multiple-Layer Feedforward Architecture

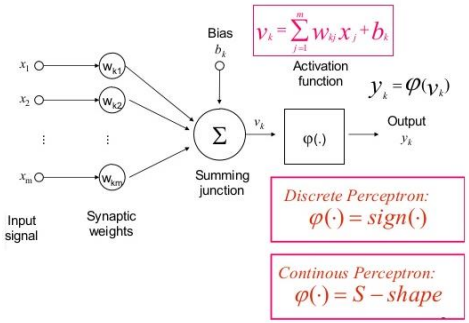


Types of feedforward networks

- Single Layer Perceptron(SLP):
  - The simplest feedforward neural network with no hidden layers
  - Not as such applicable in problem solving
- Multi Layer Perceptron (MLP):
  - Has one or more hidden layers
  - MLP is more applicable than single layer perceptron as many of the problems in the real world deserves more number of computing units
  - MLP can learn non-linear functions (non-linearity)







Multi Layer Perceptron

- Input Layer:
  - The Input layer has three nodes
  - The Bias node has a value of 1
  - The other two nodes take X1 and X2 as external inputs
- Hidden Layer:
  - It has three nodes with the Bias node having an output of 1
  - The output of the other two nodes in the Hidden layer depends on the outputs from the Input layer (1, X1, X2) as well as the weights associated with the connections (edges).
  - The output of the highlighted node is calculated
  - Similarly, the output from other hidden node can be calculated
  - f refers to the activation function
  - These outputs are then fed to the nodes in the Output layer
- Output Layer:
  - It has two nodes which take inputs from the hidden layer and perform similar computations as shown for the highlighted hidden node
  - The values calculated (Y1 and Y2) as a result of these computations are outputs of the MLP
  - Given a set of features X = (x1, x2, … xn) and a target y, the MLP can learn the relationship between the features and the target for classification

MLP: Example

- Suppose we have the following student-marks dataset

Hours Studied	Mid Term Marks	Final Term Result
35	67	1 (Pass)
12	75	0 (Fail)
16	89	1 (Pass)
45	56	1 (Pass)
10	90	0 (Fail)

- The two input columns show the number of hours the student has studied and the mid term marks obtained by the student
- The Final Result column can have two values 1 or 0 indicating whether the student passed in the final term
- It is observed that if the student studied 35 hours and had obtained 67 marks in the mid term, the student will pass the final term
- Question: Suppose, Will a student studying 25 hours and having 70 marks in the mid term pass the final term?

Hours Studied	Mid Term Marks	Final Term Result
25	70	?

- This is a binary classification problem where the MLP can learn from the given examples (training data) and make prediction given a new data point (test data)

Training MLP: The Back-Propagation Algorithm

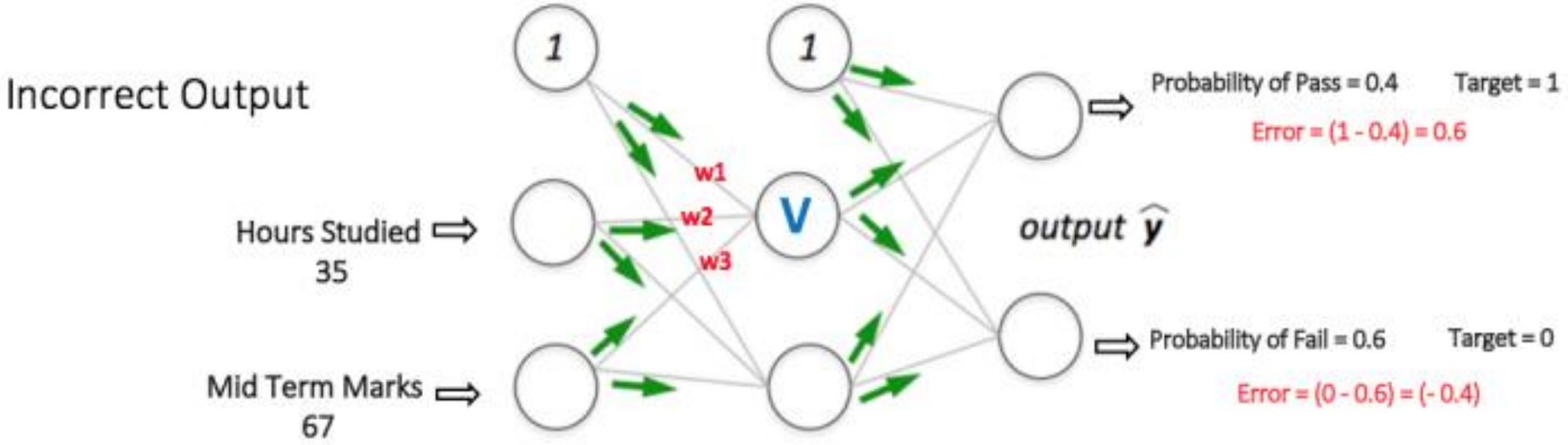
- The process by which the MLP learns is called the Backpropagation algorithm
- Backward Propagation of Errors is one of the several ways in which an artificial neural network (ANN) can be trained
- It is a supervised training scheme, that learns from labeled training data
- BackProp is like “learning from mistakes“
- The supervisor corrects the ANN whenever it makes mistakes

BackProp Algorithm

- Initially all the edge weights are randomly assigned
- For every input in the training dataset, the ANN is activated and its output is observed
- This output is compared with the desired output that we already know, and the error is “propagated” back to the previous layer
- This error is noted and the weights are adjusted (updated) accordingly
- This process is repeated until the output error is below a predetermined threshold
- Once the above algorithm terminates, it is considered as the MLP has “learned” and is ready to predict when new data point (**new input**) comes.
- This ANN is said to have learned from several examples (labelled data) and from its mistakes (error propagation)

Step 1: Forward Propagation

- All weights in the network are randomly assigned
- Lets consider the hidden layer node marked V
- Assume the weights of the connections from the inputs to that node are w1, w2 and w3 (as shown)



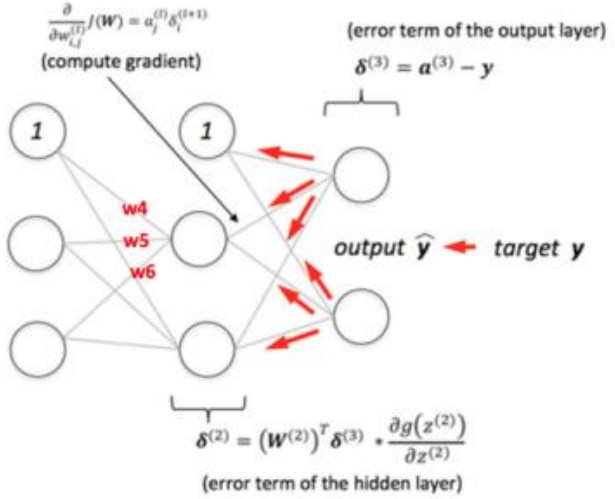
Step 1: Forward Propagation

- The network takes the first training example as input (it is know that for inputs 35 and 67, the probability of Pass is 1)
  - Input to the network = [35, 67]
  - Desired output from the network (target) = [1, 0]
- Then output V from the node in consideration can be calculated as (f is an activation function using sigmoid function):
  - V = f (1*w1 + 35*w2 + 67*w3)
- Similarly, outputs from the other node in the hidden layer is also calculated
- The outputs of the two nodes in the hidden layer act as inputs to the two nodes in the output layer
- This enables us to calculate output probabilities from the two nodes in output layer
- Suppose the output probabilities from the two nodes in the output layer are 0.4 and 0.6 respectively (since the weights are randomly assigned, outputs will also be random)
- We can see that the calculated probabilities (0.4 and 0.6) are very far from the desired probabilities (1 and 0 respectively)
  - Hence the network is has wrong output

Step 2: Back Propagation and Weight updation

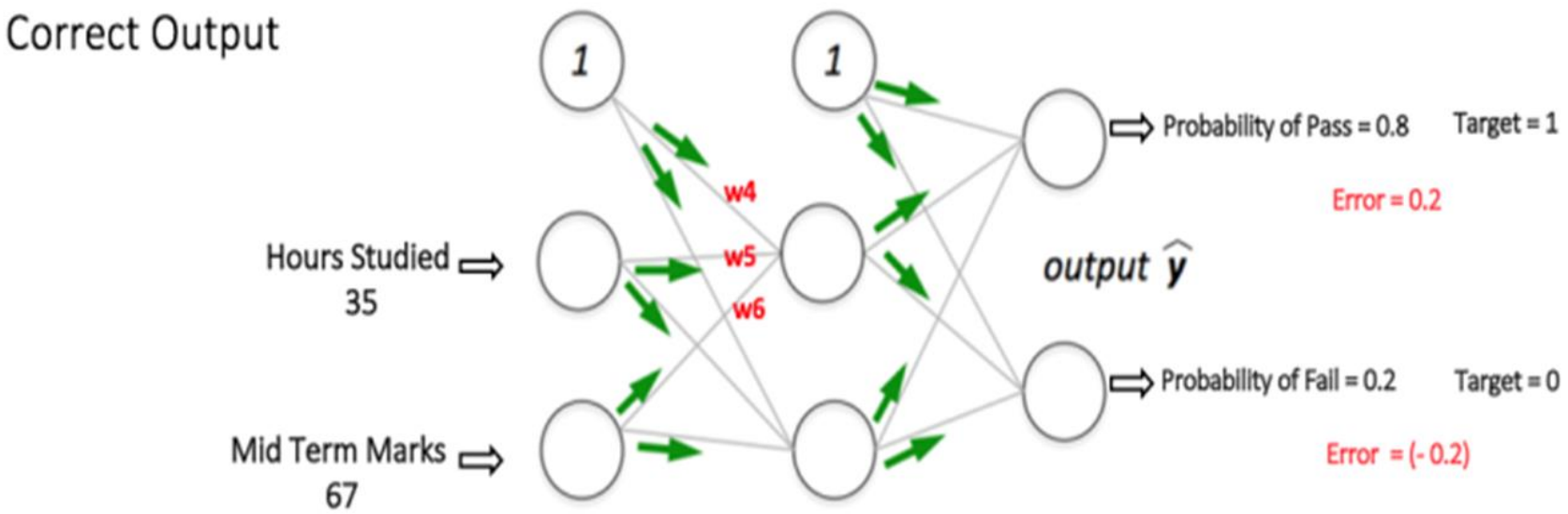
- Calculate the total error at the output nodes and propagate these errors back through the network using Backpropagation to calculate the gradients
- Then, use an optimization method such as Gradient Descent to 'adjust' all weights in the network with an aim of reducing the error at the output layer
- Suppose that the new weights associated with the node in consideration are w4, w5 and w6 (after Backpropagation and adjusting weights)

Backpropagation
+
Weights Adjusted

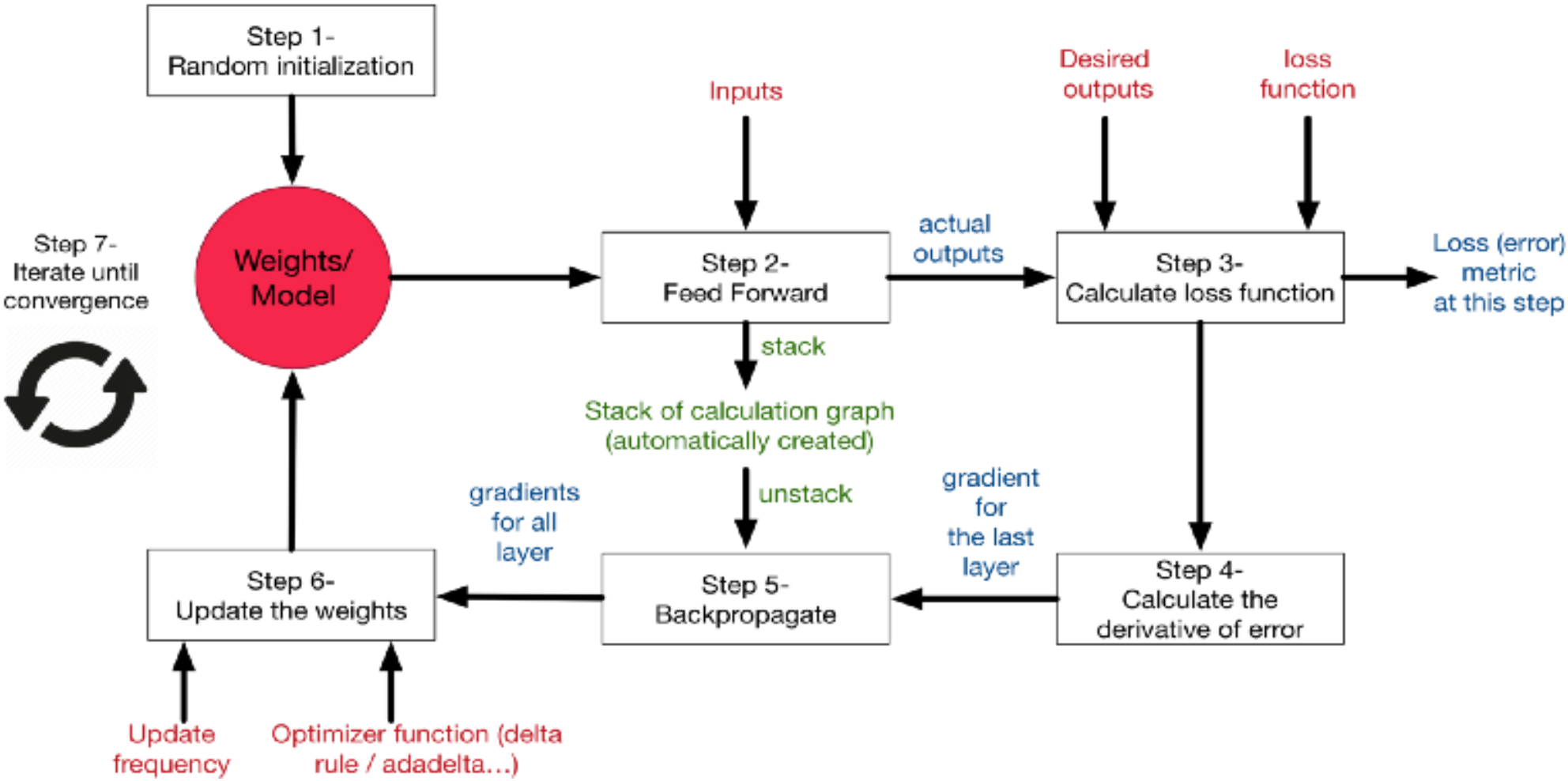


Step 2: Back Propagation and Weight updation

- If we now input the same example to the network again, the network should perform better than before since the weights have now been adjusted to minimize the error in prediction
- As shown in Figure below, the errors at the output nodes now reduce to [0.2, -0.2] as compared to [0.6, -0.4] earlier and hence the network has learnt to correctly classify the first training example.
- Repeat this process with all other training examples in the dataset and the network is said to have learnt those examples.



Process of Learning in Neural Network



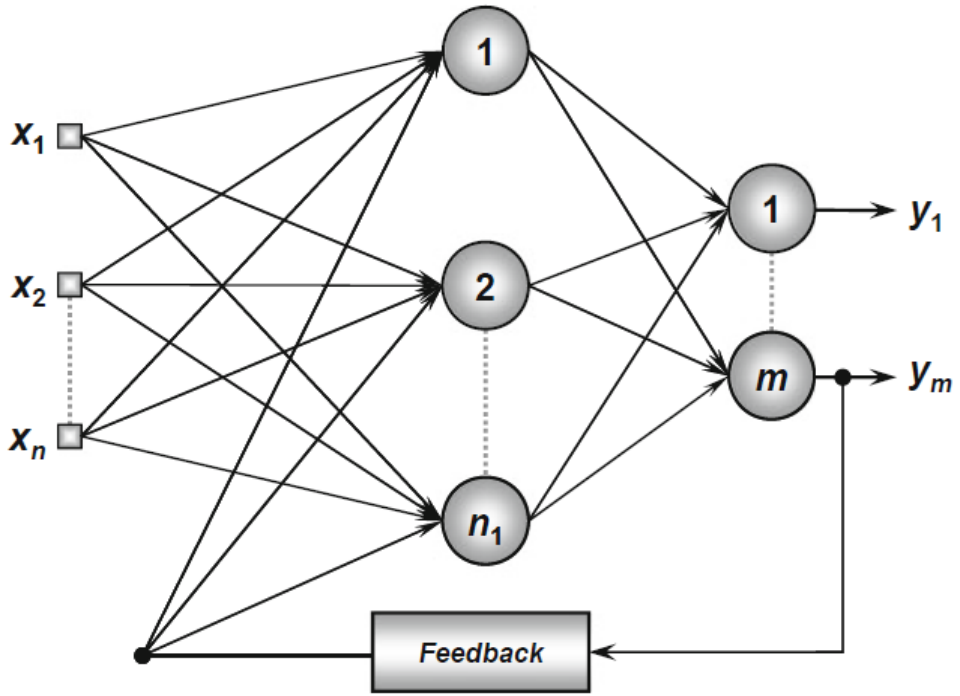
Courtesy: https://medium.com

Limitations of Neural Networks

- Random initialization + densely connected networks lead to:
- High computational cost
  - Each neuron in the neural network is considered as a logistic regression
  - Training the entire neural network is to train all the interconnected logistic regressions
- Difficult to train as the number of hidden layers increases
  - Recall that logistic regression is trained by gradient descent
- Stuck in local optima before getting the optimal (global) solution
  - The random initialization does not guarantee starting from the proximity of global optima
- Solution:
  - Deep Learning/Learning multiple levels of representation

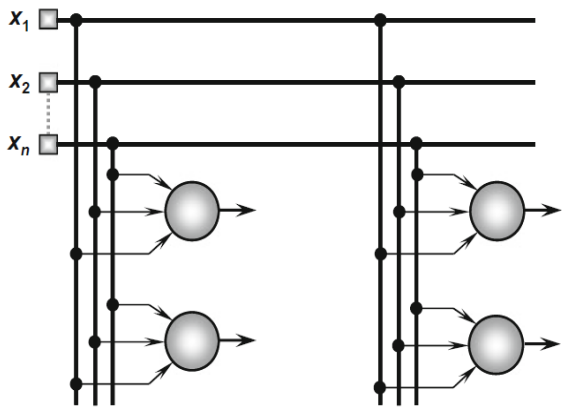
Recurrent or Feedback Architecture

- Among the main feedback networks are the Hopfield and the Perceptron with feedback between neurons from distinct layers, whose learning algorithms used in their training processes are respectively based on energy function minimization and generalized delta rule, as will be investigated in the next chapters.
- Thus, using the feedback process, the networks with this architecture produce current outputs also taking into consideration the previous output values.



Mesh Architectures

- The main features of networks with mesh structures reside in considering the spatial arrangement of neurons for pattern extraction purposes, that is, the spatial localization of the neurons is directly related to the process of adjusting their synaptic weights and thresholds.
- Although we are interested in learning networks of many interconnected units, let us begin by understanding how to learn the weights for a single perceptron.
- Here the precise learning problem is to determine a weight vector that causes the perceptron to produce the correct ±1 output for each of the given training examples.
- Several algorithms are known to solve this learning problem. Here we consider two:
  - The perceptron rule and
  - The delta rule



Perceptron rule

- One way to learn an acceptable weight vector is to begin with random weights, then iteratively apply the perceptron to each training example, modifying the perceptron weights whenever it misclassifies an example.
- This process is repeated, iterating through the training examples as many times as needed until the perceptron classifies all training examples correctly.
- Weights are modified at each step according to the perceptron training rule, which revises the weight w_i associated with input xi according to the rule
- $w_i \leftarrow w_i + \Delta w$ where $\Delta w_i = \eta(t - o)x_i$
- Here t is the target output for the current training example, o is the output generated by the perceptron, and η is a positive constant called the learning rate.
- The role of the learning rate is to moderate the degree to which weights are changed at each step.
- It is usually set to some small value (e.g., 0.1) and is sometimes made to decay as the number of weight-tuning iterations increases.

Gradient Descent and the Delta Rule

- Although the perceptron rule finds a successful weight vector when the training examples are linearly separable, it can fail to converge if the examples are not linearly separable.
- A second training rule, called the **delta rule**, is designed to overcome this difficulty.
- If the training examples are not linearly separable, the delta rule converges toward a best-fit approximation to the target concept.
- The key idea behind the delta rule is to use **gradient descent** to search the hypothesis space of possible weight vectors to find the weights that best fit the training examples.
- This rule is important because gradient descent provides the basis for the Backpropagation
- An algorithm which can learn networks with many interconnected units.
- It is also important because gradient descent can serve as the basis for learning algorithms that must search through hypothesis spaces containing many different types of continuously parameterized hypotheses.

Delta Rule

- The delta training rule is best understood by considering the task of training an un-thresholded perceptron; that is, a linear unit for which the output o is given by

$$o(\vec{x}) = \vec{w} \cdot \vec{x}$$

- Thus, a linear unit corresponds to the first stage of a perceptron, without the threshold.
- In order to derive a weight learning rule for linear units, let us begin by specifying a measure for the training error of a hypothesis (weight vector), relative to the training examples.

GRADIENT-DESCENT(training_examples, η)

- Each training example is a pair of the form <x, t > , where x is the vector of input values, and t is the target output value. η is the learning rate.

Method

- Initialize each w_i to some small random value
- Until the termination condition is met, Do
  - Initialize each Δw_i to zero
  - For each < x, t > in training_examples, Do
    - Input the instance x to the unit and compute the output o
    - For each linear unit weight wi, Do $\Delta w_i = \Delta w_i + \eta(t - o)x_i$
- For each linear unit weight w_i, Do $w_i = w_i + \Delta w_i$




Stochastic Approximation to Gradient Descent

- Gradient descent is an important general paradigm for learning. It is a strategy for searching through a large or infinite hypothesis space that can be applied whenever
  - the hypothesis space contains continuously parameterized hypotheses (e.g., the weights in a linear unit), and
  - the error can be differentiated with respect to these hypothesis parameters.
- The key practical difficulties in applying gradient descent are
  - converging to a local minimum can sometimes be quite slow (i.e., it can require many thousands of gradient descent steps), and
  - if there are multiple local minima in the error surface, then there is no guarantee that the procedure will find the global minimum

Stochastic GRADIENT-DESCENT(training_examples, η)

- Each training example is a pair of the form <x, t > , where x is the vector of input values, and t is the target output value. η is the learning rate.
- **Method**
- Initialize each w_i to some small random value
- Until the termination condition is met, Do
- Initialize each Δw_i to zero
- For each < x, t > in training_examples, Do
  - Input the instance x to the unit and compute the output o
  - For each linear unit weight wi, Do $w_i = w_i + \eta(t - o)x_i$

Gradient Descent: Some key differences

- Standard gradient descent
  - Error is summed over all examples before updating weights
  - Requires more computation per weight update step
  - Converges to local minima
- Stochastic gradient descent
  - Weights are updated upon examining each training example
  - Require less computation
  - Sometimes avoid falling into these local minima

Perceptron training rule

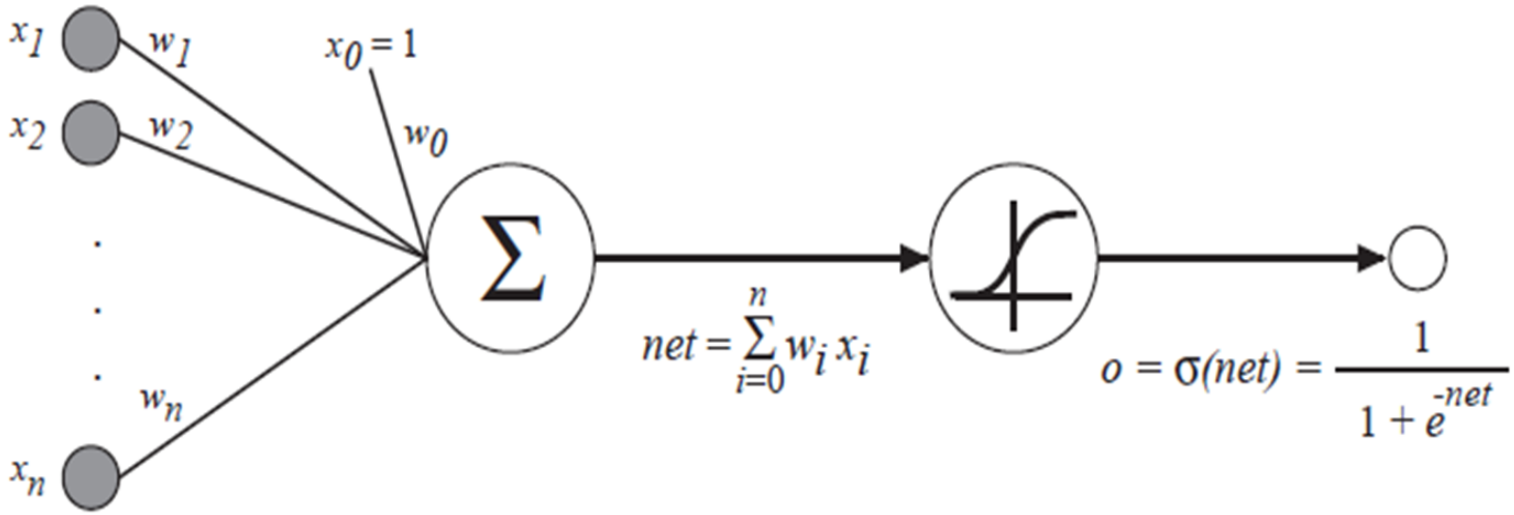
- Updates weights based on the error in the thresholded perceptron output
- Converges after a finite number of iterations to a hypothesis that perfectly classifies the training data, provided the training examples are linearly separable.

Delta rule

- Updates weights based on the error in the un-thresholded linear combination of inputs
- Converges only asymptotically toward the minimum error hypothesis, possibly requiring unbounded time, but converges regardless of whether the training data are linearly separable.

Multilayer Networks and Backpropagation Algorithm

- Single perceptrons can only express linear decision surfaces.
- In contrast, the kind of multilayer networks learned by the BACKPROPACATION algorithm are capable of expressing a rich variety of nonlinear decision surfaces.



The Backpropagation Algorithm

- Because we are considering networks with multiple output units rather than single units as before, we begin by redefining E to sum the errors over all of the network output units

$$E(\vec{w}) \equiv \frac{1}{2} \sum_{d \in D} \sum_{k \in outputs} (t_{kd} - o_{kd})^2$$

- where outputs are the set of output units in the network, and tkd and okd are the target and output values associated with the kth output unit and training example d.
- The learning problem faced by Backpropagation search a large hypothesis space defined by all possible weight values for all the units in the network.

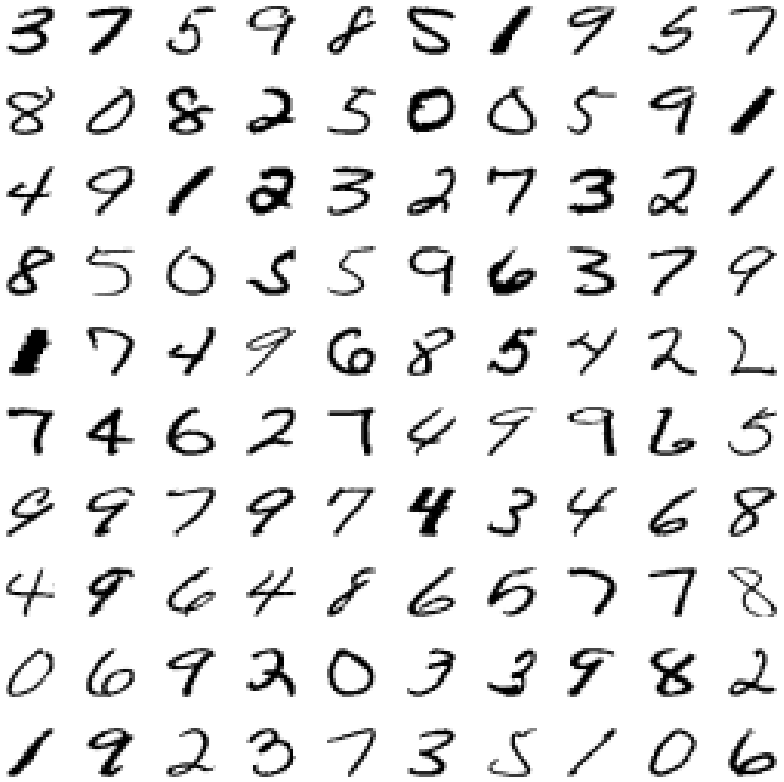
Backpropagation
(training_examples, η, n_{in} , n_{out}, n_{hidden})

- Input
  - Each training η example is a pair of the form (x, t), where x is the vector of network input values, and t is the vector of target network output values.
  - η is the learning rate (e.g., .05). n_{in} is the number of network inputs, n_{hidden} the number of units in the hidden layer, and n_{out} the number of output units.
  - The input from unit i into unit j is denoted x_{ji}, and the weight from unit i to unit j is denoted w_{ji}.
- Method
  - Create a feed-forward network with n_{in} inputs, n_{hidden} hidden units, and n_{out} output units.
  - Initialize all network weights to small random numbers (e.g., between -.05 and .05).
  - Until the termination condition is met, Do
  - For each (x,t) in training examples, Do
  - Propagate the input forward through the network:
    - Input the instance x to the network and compute the output o, of every unit u in the network.
  - Propagate the errors backward through the network:
    - For each network output unit k, calculate its error term δ_k
  - $\delta_k \leftarrow o_k(1 - o_k)(t_k - o_k)$
  - For each hidden unit h, calculate its error term δ_h
  - $\delta_h \leftarrow o_h(1 - o_h) \sum_{k \in outputs} w_{kh} \delta_k$
    - Update each network weight w_{ji}
  - $w_{ji} \leftarrow w_{ji} + \Delta w_{ji}$ where $\Delta w_{ji} = \eta \delta_j x_{ji}$

Support Vector Machine

- A Supervised Classifier

Handwritten Digit Clustering and Recognition



How can we develop algorithms that will automatically cluster these images?

Use a training set of labeled images to learn to classify new images?

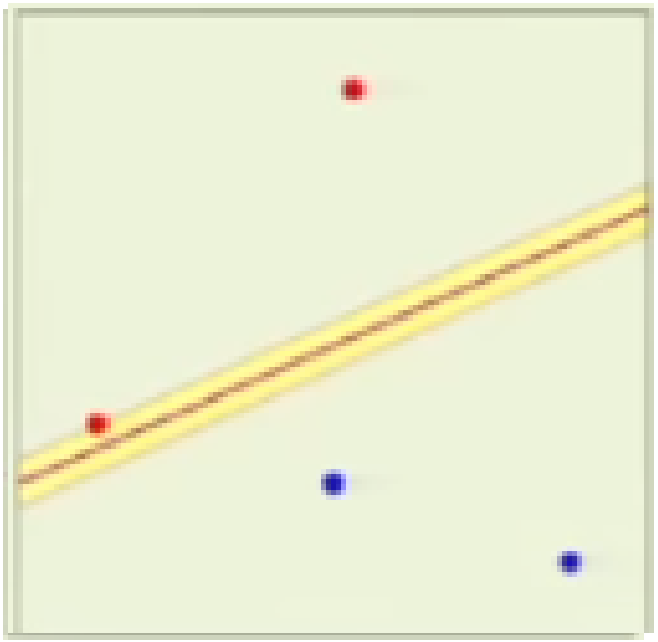
Discover how to account for variability in writing style?

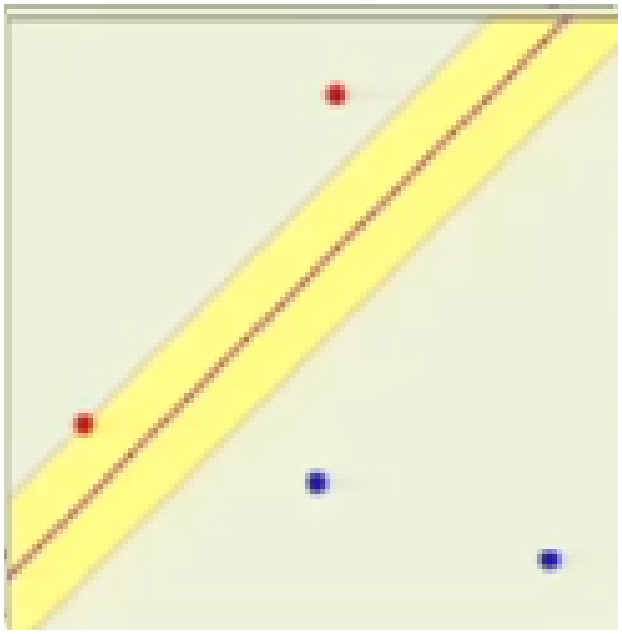
Support Vector Machines (SVM)

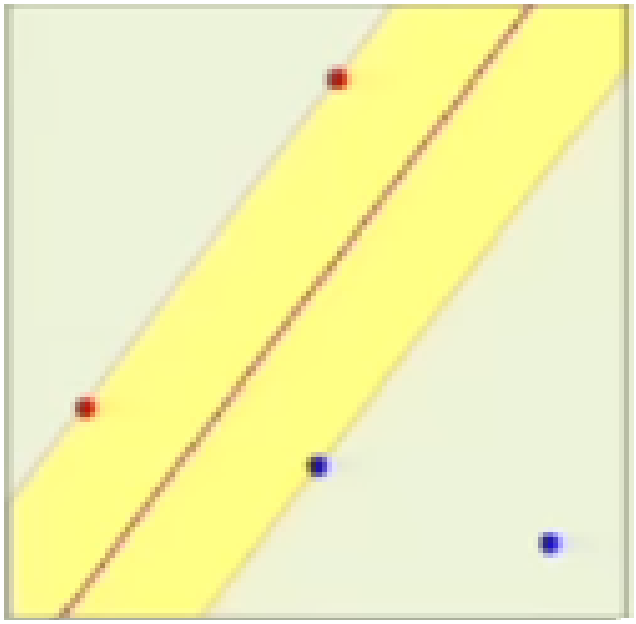
- Supervised learning methods for classification and relatively new class of successful learning methods
- They can represent non-linear functions and they have an efficient training algorithm
- Derived from statistical learning theory by Vapnik and Chervonenkis (COLT-92)
- SVM got into mainstream because of their exceptional performance in Handwritten Digit Recognition

Support Vector Machine

Consider a linearly separable data points







Different separating lines and which is the best among these?

Why bigger margin is better?

Which w maximizes the margin?

Finding $\mathbf{w}$ with large margin

Let x_n be the nearest point to the plane $\mathbf{w}^T\mathbf{x} = 0$

Without loss of generality:

1. Normalize w: $|\mathbf{w}^T x_n| = 1$
2. The plane is $\mathbf{w}^T\mathbf{x} + b = 0$

Computing the distance

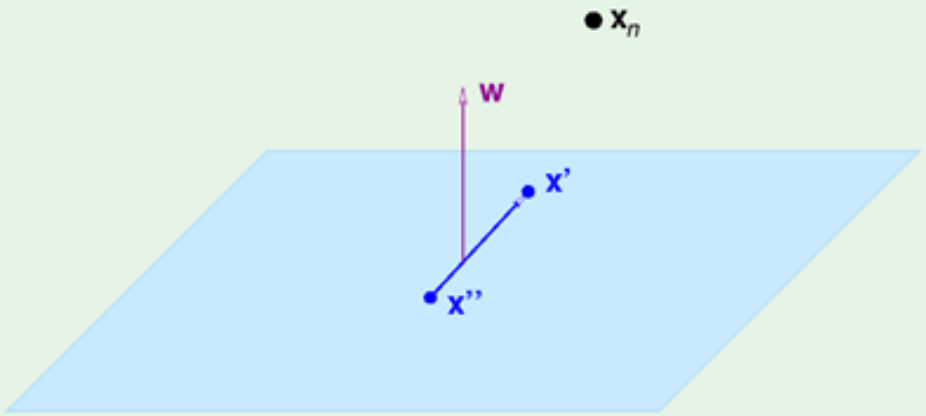
The distance between $\mathbf{x}_n$ and the plane $\mathbf{w}^\intercal\mathbf{x} + b = 0$ where $|\mathbf{w}^\intercal\mathbf{x}_n + b| = 1$

The vector $\mathbf{w}$ is $\perp$ to the plane in the $\mathcal{X}$ space:

Take $\mathbf{x}'$ and $\mathbf{x}''$ on the plane

$\mathbf{w}^\intercal\mathbf{x}' + b = 0$ and $\mathbf{w}^\intercal\mathbf{x}'' + b = 0$

$\implies \mathbf{w}^\intercal(\mathbf{x}' - \mathbf{x}'') = 0$

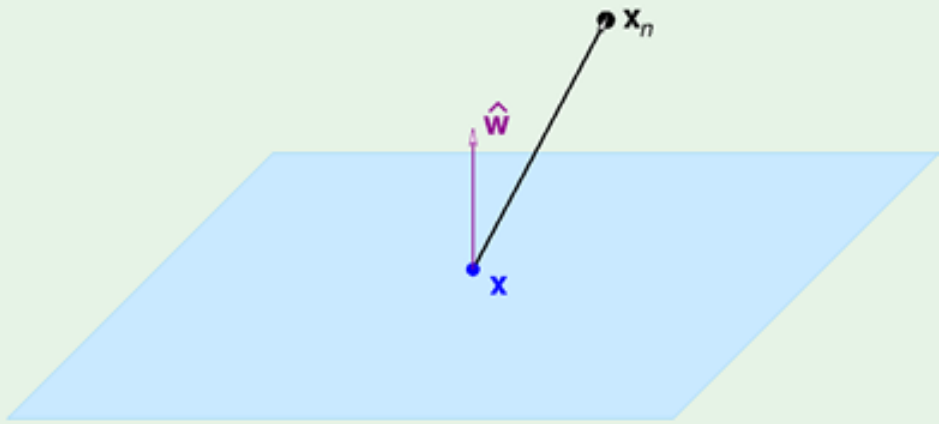


The distance is

Distance between $\mathbf{x}_n$ and the plane:

Take any point $\mathbf{x}$ on the plane

Projection of $\mathbf{x}_n - \mathbf{x}$ on $\mathbf{w}$



$$\hat{\mathbf{w}} = \frac{\mathbf{w}}{\|\mathbf{w}\|} \implies \text{distance} = \left|\hat{\mathbf{w}}^\intercal(\mathbf{x}_n - \mathbf{x})\right|$$

$$\text{distance} = \frac{1}{\|\mathbf{w}\|}\left|\mathbf{w}^\intercal\mathbf{x}_n - \mathbf{w}^\intercal\mathbf{x}\right| = \frac{1}{\|\mathbf{w}\|}\left|\mathbf{w}^\intercal\mathbf{x}_n + b - \mathbf{w}^\intercal\mathbf{x} - b\right| = \frac{1}{\|\mathbf{w}\|}$$

The optimization problem

Maximize $\dfrac{1}{\|\mathbf{w}\|}$

subject to $\min_{n=1,2,\ldots,N} |\mathbf{w}^{\mathsf{T}}\mathbf{x}_n + b| \;=\; 1$

Notice: $|\mathbf{w}^{\mathsf{T}}\mathbf{x}_n + b| \;= y_n\,(\mathbf{w}^{\mathsf{T}}\mathbf{x}_n + b)$

Minimize $\dfrac{1}{2}\,\mathbf{w}^{\mathsf{T}}\mathbf{w}$

subject to $y_n\,(\mathbf{w}^{\mathsf{T}}\mathbf{x}_n + b) \geq 1 \quad$ for $\quad n = 1, 2, \ldots, N$

Constrained Optimization

$$\text{Minimize} \quad \frac{1}{2}\,\mathbf{w}^{\mathsf{T}}\mathbf{w}$$

$$\text{subject to} \quad y_n\left(\mathbf{w}^{\mathsf{T}}\mathbf{x}_n + b\right) \geq 1 \quad \text{for} \quad n = 1, 2, \ldots, N$$

$$\mathbf{w} \in \mathbb{R}^d, \; b \in \mathbb{R}$$

Solution is: Lagrange Multipliers

A strategy for finding the local maxima and minima of a function subject to equality constraints

Lagrange Multipliers

For the case of only one constraint and only two choice variables consider the optimization problem

- maximize f(x, y)
- subject to g(x, y) = 0.

$\mathcal{L}(x, y, \lambda) = f(x, y) - \lambda . g(x, y)$ where λ is the Lagrange multiplier

Minimize $$\mathcal{L}(\mathbf{w}, b, \boldsymbol{\alpha}) = \frac{1}{2}\mathbf{w}^{\intercal}\mathbf{w} - \sum_{n=1}^{N} \alpha_n (y_n (\mathbf{w}^{\intercal}\mathbf{x}_n + b) - 1)$$

w.r.t. $\mathbf{w}$ and b and maximize w.r.t. each $\alpha_n \geq 0$

$$\nabla_{\mathbf{w}}\mathcal{L} = \mathbf{w} - \sum_{n=1}^{N} \alpha_n y_n \mathbf{x}_n = \mathbf{0}$$

$$\frac{\partial \mathcal{L}}{\partial b} = - \sum_{n=1}^{N} \alpha_n y_n = 0$$

Upon substitution

$$\mathbf{w} = \sum_{n=1}^{N} \alpha_n y_n \mathbf{x}_n \qquad \text{and} \qquad \sum_{n=1}^{N} \alpha_n y_n = 0$$

in the Lagrangian $$\mathcal{L}(\mathbf{w}, b, \boldsymbol{\alpha}) = \frac{1}{2}\mathbf{w}^\intercal\mathbf{w} - \sum_{n=1}^{N} \alpha_n \left(y_n \left(\mathbf{w}^\intercal\mathbf{x}_n + b\right) - 1\right)$$

we get $$\mathcal{L}(\boldsymbol{\alpha}) = \sum_{n=1}^{N} \alpha_n - \frac{1}{2}\sum_{n=1}^{N}\sum_{m=1}^{N} y_n y_m \, \alpha_n \alpha_m \, \mathbf{x}_n^\intercal \mathbf{x}_m$$

Maximize w.r.t. to $\boldsymbol{\alpha}$ subject to $\alpha_n \geq 0$ for $n = 1, \cdots, N$ **and** $\sum_{n=1}^{N} \alpha_n y_n = 0$

The Solution – Quadratic Programming

$$\min_{\boldsymbol{\alpha}} \quad \frac{1}{2}\,\boldsymbol{\alpha}^{\intercal}\underbrace{\begin{bmatrix} y_1y_1\,\mathbf{x}_1^{\intercal}\mathbf{x}_1 & y_1y_2\,\mathbf{x}_1^{\intercal}\mathbf{x}_2 & \dots & y_1y_N\,\mathbf{x}_1^{\intercal}\mathbf{x}_N \\ y_2y_1\,\mathbf{x}_2^{\intercal}\mathbf{x}_1 & y_2y_2\,\mathbf{x}_2^{\intercal}\mathbf{x}_2 & \dots & y_2y_N\,\mathbf{x}_2^{\intercal}\mathbf{x}_N \\ \dots & \dots & \dots & \dots \\ y_Ny_1\,\mathbf{x}_N^{\intercal}\mathbf{x}_1 & y_Ny_2\,\mathbf{x}_N^{\intercal}\mathbf{x}_2 & \dots & y_Ny_N\,\mathbf{x}_N^{\intercal}\mathbf{x}_N \end{bmatrix}}_{\text{quadratic coefficients}}\boldsymbol{\alpha} + \underbrace{(-\mathbf{1}^{\intercal})}_{\text{linear}}\boldsymbol{\alpha}$$

subject to $\underbrace{\mathbf{y}^{\intercal}\boldsymbol{\alpha} = 0}_{\text{linear constraint}}$

$$\underbrace{\mathbf{0}}_{\text{lower bounds}} \leq \boldsymbol{\alpha} \leq \underbrace{\infty}_{\text{upper bounds}}$$

QP gives us α

Solution: $\boldsymbol{\alpha} = \alpha_1, \cdots, \alpha_N$

$$\Longrightarrow \quad \mathbf{w} = \sum_{n=1}^{N} \alpha_n y_n \mathbf{x}_n$$

KKT condition: For $n = 1, \cdots, N$

$$\alpha_n \left(y_n \left(\mathbf{w}^{\intercal} \mathbf{x}_n + b \right) - 1 \right) = 0$$

$\alpha_n > 0 \Longrightarrow \mathbf{x}_n$ is a **support vector**

Allowing inequality constraints, the **Karush–Kuhn–Tucker (KKT) (**KKT) conditions approach to nonlinear programming generalizes the method of Lagrange multipliers, which allows only equality constraints.

Support vectors

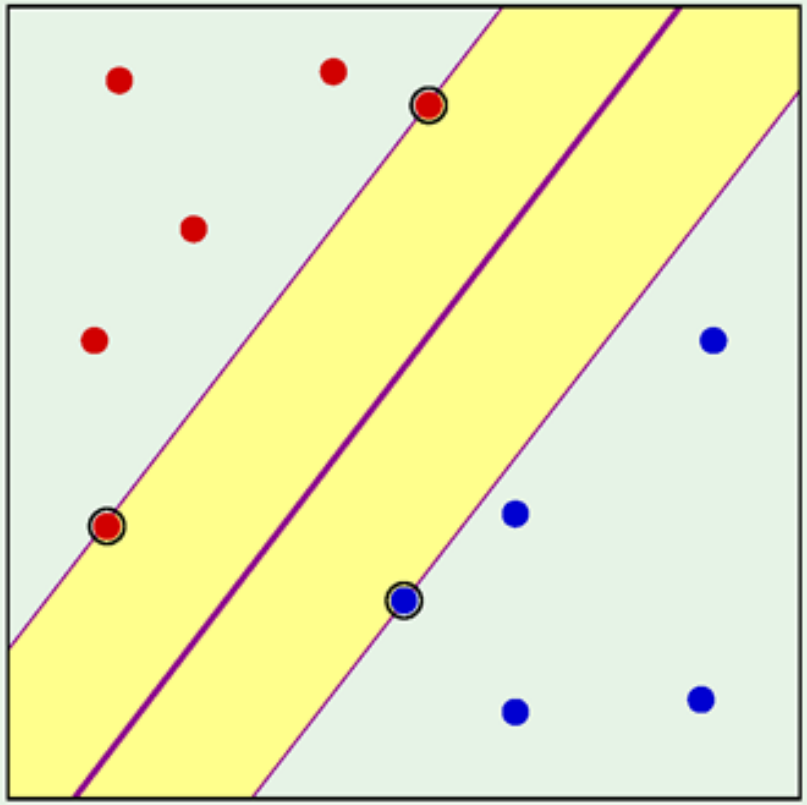
Closest $\mathbf{x}_n$'s to the plane: achieve the margin

$$\implies \quad y_n \left(\mathbf{w}^\intercal \mathbf{x}_n + b\right) = 1$$

$$\mathbf{w} \;=\; \sum_{\mathbf{x}_n \text{ is SV}} \alpha_n y_n \mathbf{x}_n$$

Solve for b using any SV:

$$y_n \left(\mathbf{w}^\intercal \mathbf{x}_n + b\right) = 1$$



Classification Model Evaluation and Selection

- Need a way to choose between models: different model types, tuning parameters, and features
- Use a model evaluation procedure to estimate how well a model will generalize to out-of-sample data
- Requires a model evaluation metric to quantify the model performance
- **Model evaluation procedures**
- **Training and testing on the same data**
  - Rewards overly complex models that "overfit" the training data and won't necessarily generalize
- **Train/test split**
  - Split the dataset into two pieces, so that the model can be trained and tested on different data
  - Better estimate of out-of-sample performance, but still a "high variance" estimate
  - Useful due to its speed, simplicity, and flexibility
- **K-fold cross-validation**
  - Systematically create "K" train/test splits and average the results together
  - Even better estimate of out-of-sample performance
  - Runs "K" times slower than train/test split
- Model evaluation metrics
  - Regression problems: Mean Absolute Error, Mean Squared Error, Root Mean Squared Error
  - Classification problems: Classification accuracy

Classification Model Evaluation and Selection

- **Classification accuracy:** percentage of correct predictions
- **Confusion matrix:** Table that describes the performance of a classification model

		Actual Values: Positive (1)	Actual Values: Negative (0)
Predicted Values	Positive (1)	TP	FP
Predicted Values	Negative (0)	FN	TN

- Recall: $$Recall = \frac{TP}{TP + FN}$$
- Precision: $$Precision = \frac{TP}{TP + FP}$$
- F-measure: $$F\text{-}measure = \frac{2*Recall*Precision}{Recall + Precision}$$

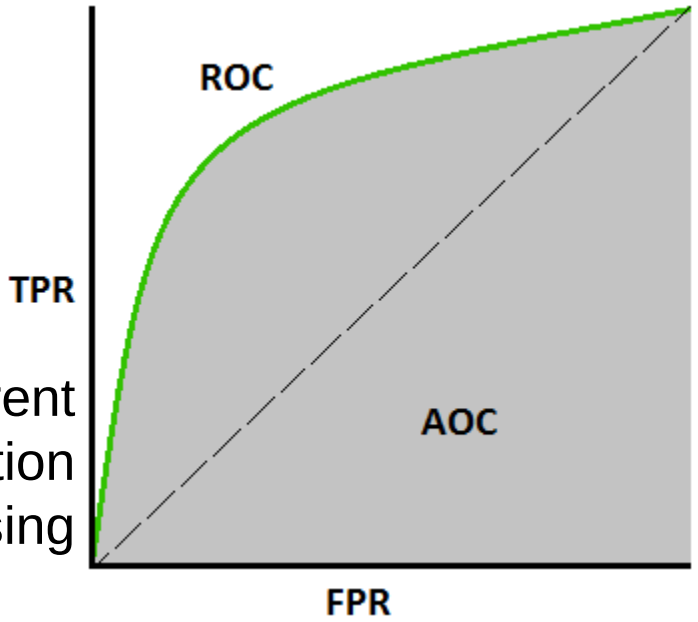
Lift curve and Gain curve

- Lift and Gain Charts are a useful way of visualizing how good a predictive model is.
- Cumulative gains and lift curves are a simple and useful approach to understand what returns you are likely to get from a predictive model.
- For example: These approaches could similarly be applied in the context of predicting which individuals will default on a personal loan in order to decide who could be offered a credit card. In this case, the aim is to minimize the number of people likely to default on the loan, whilst maximizing the number of credit cards offered to those who will not default.
- The predictive model in each case could be any appropriate statistical approach for generating a probability for a binary outcome, be that a logistic regression model, a random forest, or a neural network.

ROC Curves

- ROC curve is a performance measurement for classification problem at various thresholds settings.
- ROC is a probability curve and AUC represents degree or measure of separability.
- It tells how much model is capable of distinguishing between classes. Higher the AUC, better the model is at predicting 0s as 0s and 1s as 1s.
- The ROC curve is plotted with TPR against the FPR where TPR is on y-axis and FPR is on the x-axis.
- True Positive Rate: $TPR = \frac{TP}{TP + FN}$
- False Positive Rate: $FPR = \frac{FP}{FP + TN}$

An ROC curve plots TPR vs. FPR at different classification thresholds. Lowering the classification threshold classifies more items as positive, thus increasing both False Positives and True Positives.



Cost-benefit analysis

- Cost–benefit analysis (CBA), sometimes called benefit costs analysis (BCA), is a systematic approach to estimating the strengths and weaknesses of alternatives.
- A CBA may be used to compare potential (or completed) courses of actions, or to estimate (or evaluate) the value against the cost of a decision, project, or policy.
- Cost–benefit analysis is often used by organizations to appraise the desirability of a given policy.
- The value of a cost–benefit analysis depends on the accuracy of the individual cost and benefit estimates.
- In health economics, CBA may be an inadequate measure because willingness-to-pay methods of determining the value of human life can be influenced by income level. Variants, such as cost–utility analysis, QALY and DALY to analyze the effects of health policies, may be more suitable.

Checkpoints (1 of 2)

Fill in the blanks:

1. _____ learning is used when you build component classifiers that are more accurate and independent from each other.
2. _____ are supervised learning algorithms used for classification and regression analysis.
3. _____ learning is a learning framework that has been introduced to analyze learning algorithms and their statistical efficiency.

State True or False:

1. Designing and developing algorithms according to the behaviors based on empirical data are known as Machine Learning.
2. Sequence learning is a method of teaching and learning in a logical manner.
3. Instance based learning algorithm is also referred as Lazy learning algorithm as they delay the induction or generalization process until classification is performed.

Checkpoint Solutions (1 of 2)

Fill in the blanks:

1. **Ensemble** learning is used when you build component classifiers that are more accurate and independent from each other.
2. **Support vector machines** are supervised learning algorithms used for classification and regression analysis.
3. **Probably Approximately Correct** learning is a learning framework that has been introduced to analyze learning algorithms and their statistical efficiency.

State True or False:

1. Designing and developing algorithms according to the behaviors based on empirical data are known as Machine Learning **– True.**
2. Sequence learning is a method of teaching and learning in a logical manner – **True.**
3. Instance based learning algorithm is also referred as Lazy learning algorithm as they delay the induction or generalization process until classification is performed **– True.**

Checkpoints (2 of 2)

Multiple Choice Questions

1. Decision trees are appropriate for the problems where:
   a. Attributes are both numeric and nominal
   b. Target function takes on a discrete number of values.
   c. Data may have errors
   d. All of the mentioned

2. Which of the following SVM model can be used to detect the outliers?
   a. nu-SVM classification
   b. nu-SVM regression
   c. one-classification
   d. None of the above

3. Suppose your model is over-fitting. Which of the following is NOT a valid way to try and reduce the over-fitting?
   a. Increase the amount of training data
   b. Improve the optimization algorithm being used for error minimization.
   c. Decrease the model complexity
   d. Reduce the noise in the training data

Checkpoints Solutions (2 of 2)

Multiple Choice Questions

1. Decision trees are appropriate for the problems where:
   a. Attributes are both numeric and nominal
   b. Target function takes on a discrete number of values.
   c. Data may have errors
   d. All of the mentioned

2. Which of the following SVM model can be used to detect the outliers?
   a. nu-SVM classification
   b. nu-SVM regression
   c. one-classification
   d. None of the above

3. Suppose your model is overfitting. Which of the following is NOT a valid way to try and reduce the overfitting?
   a. Increase the amount of training data
   b. Improve the optimization algorithm being used for error minimization.
   c. Decrease the model complexity
   d. Reduce the noise in the training data

Two Marks Questions

1. Define machine learning. Give an example.
2. Define instance based learning. Give an example.
3. Define curse of dimensionality. How do you address this issue in ML?
4. Define information gain. Illustrate.
5. Define entropy. Illustrate.
6. Define ensemble of classifiers.
7. Define random forest. What is its importance in ML?
8. Define neural network. Give an example.
9. Define ROC curves. Illustrate.
10. Define precision and recall. Give an example.

Four Marks Questions

1. Discuss the problems in classifying data.
2. What are the advantages and disadvantages of classification?
3. What are the characteristics of classification algorithms?
4. Discuss in brief k-nearest neighbor technique.
5. Explain bagging with an example.
6. Explain adaboost algorithm.
7. Discuss the different types of activation functions used in neural networks.
8. What are the limitations of neural networks?

Eight Marks Questions

1. Discuss in brief, various applications of classification.
2. With its block diagram, explain the general classification model.
3. With an example, discuss the procedure of constructing decision tree.
4. Explain Naïve- Bayesian classifier.
5. Discuss feedforward neural network based classification.
6. Explain MLP with an example.
7. Explain gradient descent technique of classification.
8. Discuss SVM as a classifier.