CS3121 - Introduction to Data Science

Supervised Learning

Dr. Nisansa de Silva,
Department of Computer Science & Engineering
http://nisansads.staff.uom.lk/

Basic concepts

What is Learning?

“Learning is any process by which a system improves performance from experience.”

– Herbert Simon

“A computer program is said to **learn** from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E.”

– Tom Mitchell

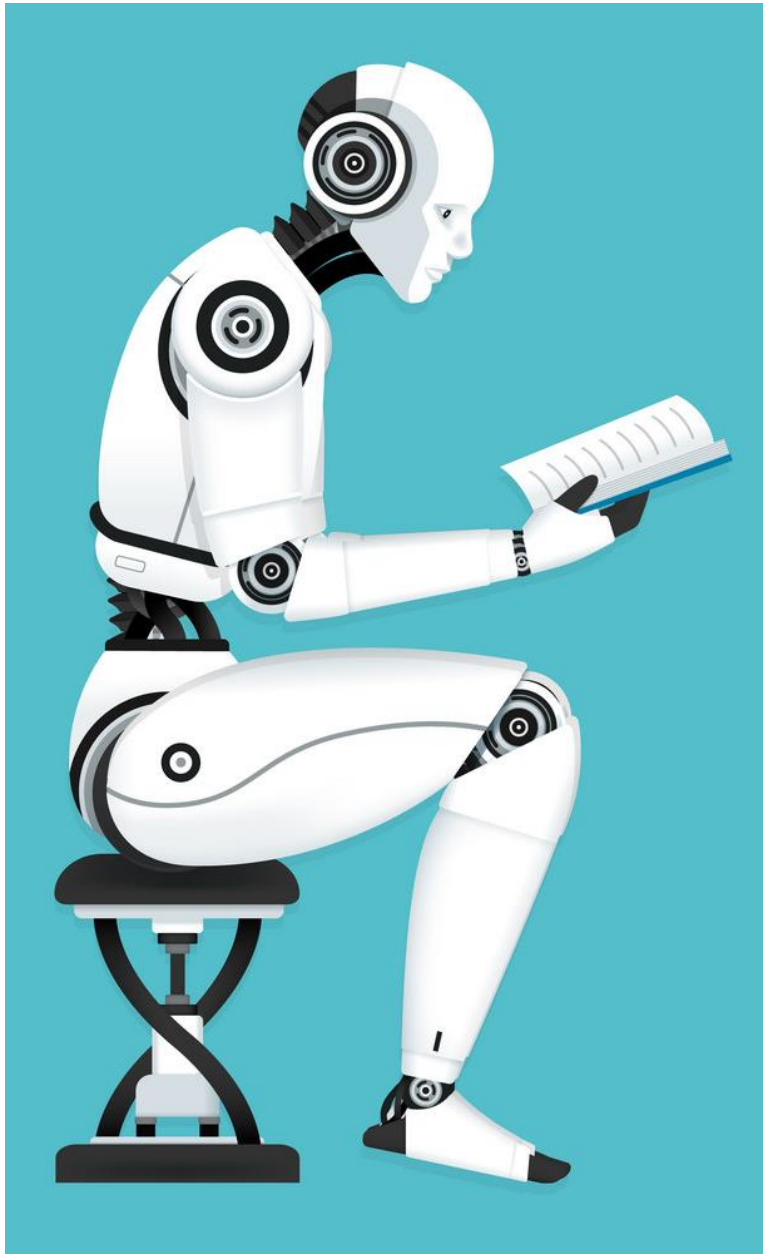
3

Learning

- Learning is essential for unknown environments,
  - i.e., when designer lacks omniscience
- Learning is useful as a system construction method,
  - i.e., expose the agent to reality rather than trying to write it down
- Learning modifies the agent's decision mechanisms to improve performance

Machine Learning

- Machine learning: how to acquire a model on the basis of data / experience
  - Learning parameters (e.g. probabilities)
  - Learning structure (e.g. BN graphs)
  - Learning hidden concepts (e.g. clustering)
- Like human learning from past experiences.
- A computer does not have “experiences”.
- A computer system learns from data, which represent some “past experiences” of an application domain.
- Our focus: learn a target function that can be used to predict the values of a discrete class attribute, e.g., approve or not-approved, and high-risk or low risk.
- The method is commonly called: inductive learning.



Machine Learning

- Given
  - a data set *D*,
  - a task *T,* and
  - a performance measure *M*,

  a computer system is said to **learn** from *D* to perform the task *T* if after learning the system's performance on *T* improves as measured by *M*.
- In other words, the learned model helps the system to perform *T* better as compared to no learning.
- An example
  - Data: Loan application data
  - Task: Predict whether a loan should be approved or not.
  - Performance measure: accuracy.

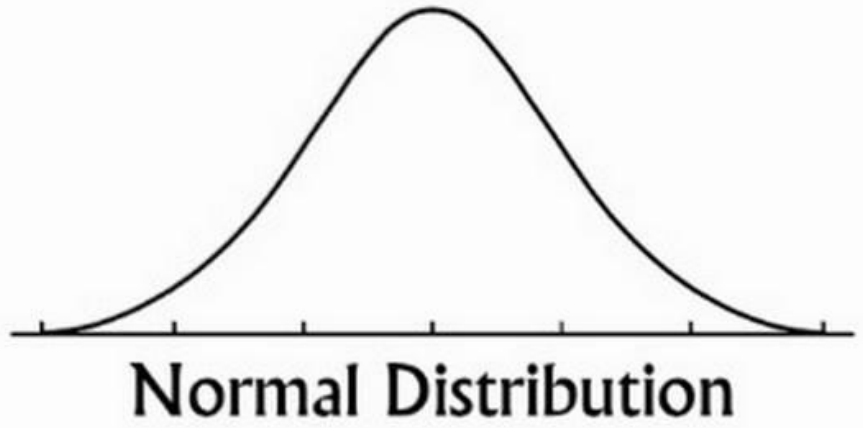
  No learning: classify all future applications (test data) to the majority class (i.e., Yes):

  Accuracy = 9/15 = 60%.
  - We can do better than 60% with learning.

Fundamental Assumption of Machine Learning

Assumption: The distribution of training examples is identical to the distribution of test examples (including future unseen examples).

- In practice, this assumption is often violated to certain degree.
- Strong violations will clearly result in poor classification accuracy.
- To achieve good accuracy on the test data, training examples must be sufficiently representative of the test data.



An Example Application

- An emergency room in a hospital measures 17 variables (e.g., blood pressure, age, etc) of newly admitted patients.
- **A decision is needed**: whether to put a new patient in an intensive-care unit.
- Due to the high cost of ICU,
  - Those patients who may survive less than a month are given higher priority.
  - On the other hand, the patience who may survive only a few hours are given lower priority.
- **Problem**: to predict **high-risk patients** and discriminate them from **low-risk patients**.

Another Application

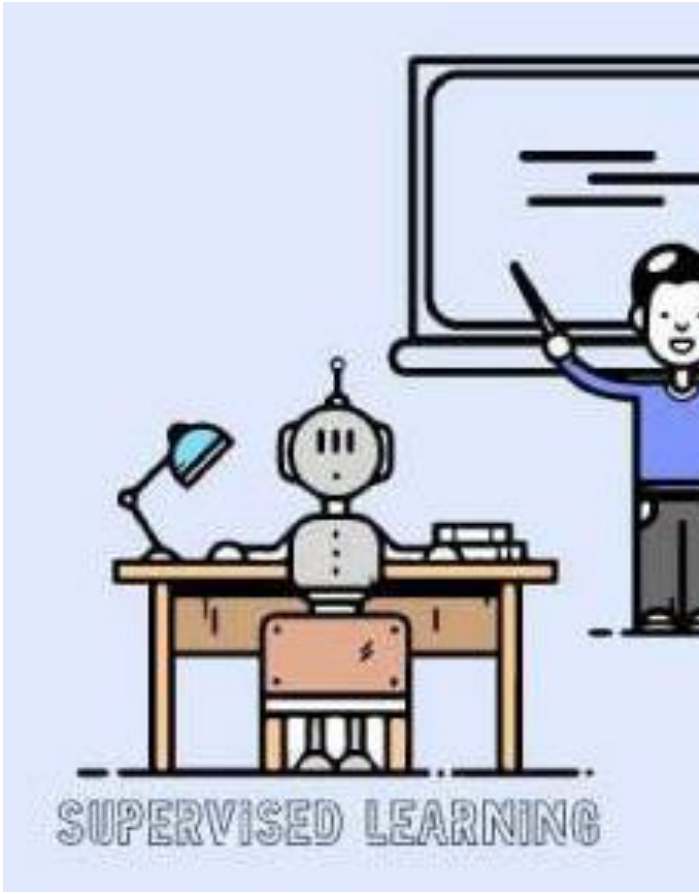
- A credit card company receives thousands of applications for new cards. Each application contains information about an applicant,
  - age
  - Marital status
  - annual salary
  - outstanding debts
  - credit rating
  - etc.
- Problem: to decide whether an application should approved, or to classify applications into two categories, approved and not approved.

Machine Learning Areas

- **Supervised Learning**: Data and corresponding labels are given
  - Supervision: The data (observations, measurements, etc.) are labeled with pre-defined classes. It is like that a “teacher” gives the classes (supervision).
  - Test data are classified into these classes too.
  - The process of inferring a function from labeled training data drawn from set of training examples.
  - <Input object, Desired Output Value>
- **Unsupervised Learning**: Only data is given, no labels provided
  - Class labels of the data are unknown
  - Given a set of data, the task is to establish the existence of classes or clusters in the data
- **Semi-supervised Learning**: Some (if not all) labels are present
- **Reinforcement Learning**: An agent interacting with the world makes observations, takes actions, and is rewarded or punished; it should learn to choose actions in such a way as to obtain a lot of reward

Supervised Learning : Important Concepts

- Data: labeled instances <x_i, y>, e.g. emails marked spam/not spam
  - Training Set
  - Held-out Set / Validation Set
  - Test Set
- Features: attribute-value pairs which characterize each x
- Experimentation cycle
  - Learn parameters (e.g. model probabilities) on training set
  - (Tune hyper-parameters on held-out set)
  - Compute accuracy of test set
  - Very important: never “peek” at the test set!
- Evaluation
  - Accuracy: fraction of instances predicted correctly
- Overfitting and generalization
  - Want a classifier which does well on test data
  - Overfitting: fitting the training data very closely, but not generalizing well



Example: Spam Filter

Input: email

Output: spam/ham

Setup:

- Get a large collection of example emails, each labeled "spam" or "ham"
- Note: someone has to hand label all this data!
- Want to learn to predict labels of new, future emails

Features: The attributes used to make the ham / spam decision

- Words: FREE!
- Text Patterns: $dd, CAPS
- Non-text: SenderInContacts
- …

Dear Sir.

First, I must solicit your confidence in this transaction, this is by virture of its nature as being utterly confidencial and top secret. …

TO BE REMOVED FROM FUTURE MAILINGS, SIMPLY REPLY TO THIS MESSAGE AND PUT "REMOVE" IN THE SUBJECT.

99 MILLION EMAIL ADDRESSES FOR ONLY $99

Ok, Iknow this is blatantly OT but I'm beginning to go insane. Had an old Dell Dimension XPS sitting in the corner and decided to put it to use, I know it was working pre being stuck in the corner, but when I plugged it in, hit the power nothing happened.

Slide from Mackassy

Example: Spam Filter

Michigan woman checks spam folder and learns she won the lottery

Laura Spears, 55, has now added the lottery to her safe senders list in case of future jackpots

Lottery tickets at a convenience store in Illinois on 6 January 2021. Photograph: Nam Y Huh/AP

Slide from Mackassy

Example: Digit Recognition

Input: images / pixel grids

Output: a digit 0-9

Setup:

- Get a large collection of example images, each labeled with a digit
- Note: someone has to hand label all this data!
- Want to learn to predict labels of new, future digit images

Features: The attributes used to make the digit decision

- Pixels: (6,8)=ON
- Shape Patterns: NumComponents, AspectRatio, NumLoops
- …

0

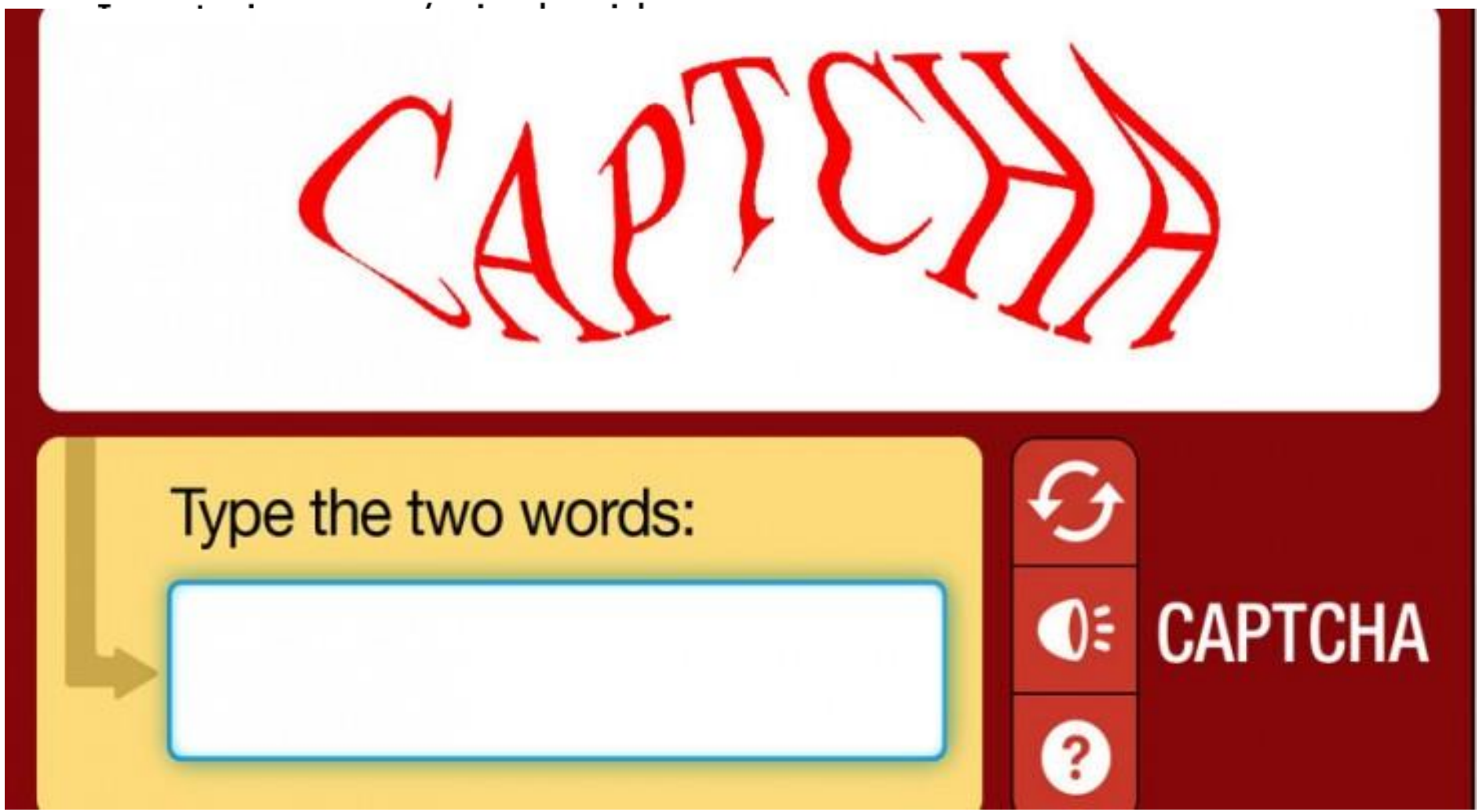
1

2

1

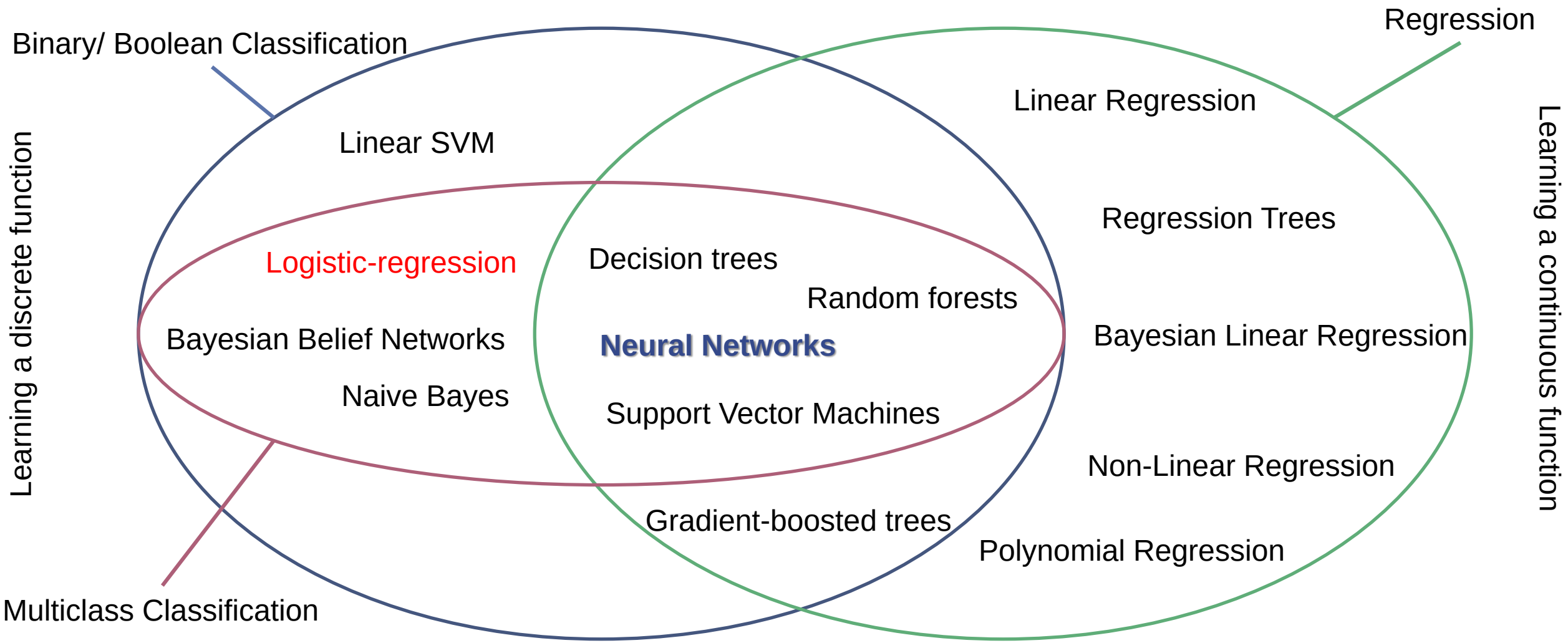
??

Example: Digit Recognition



Slide from Mackassy

Supervised Learning: Classification & Regression

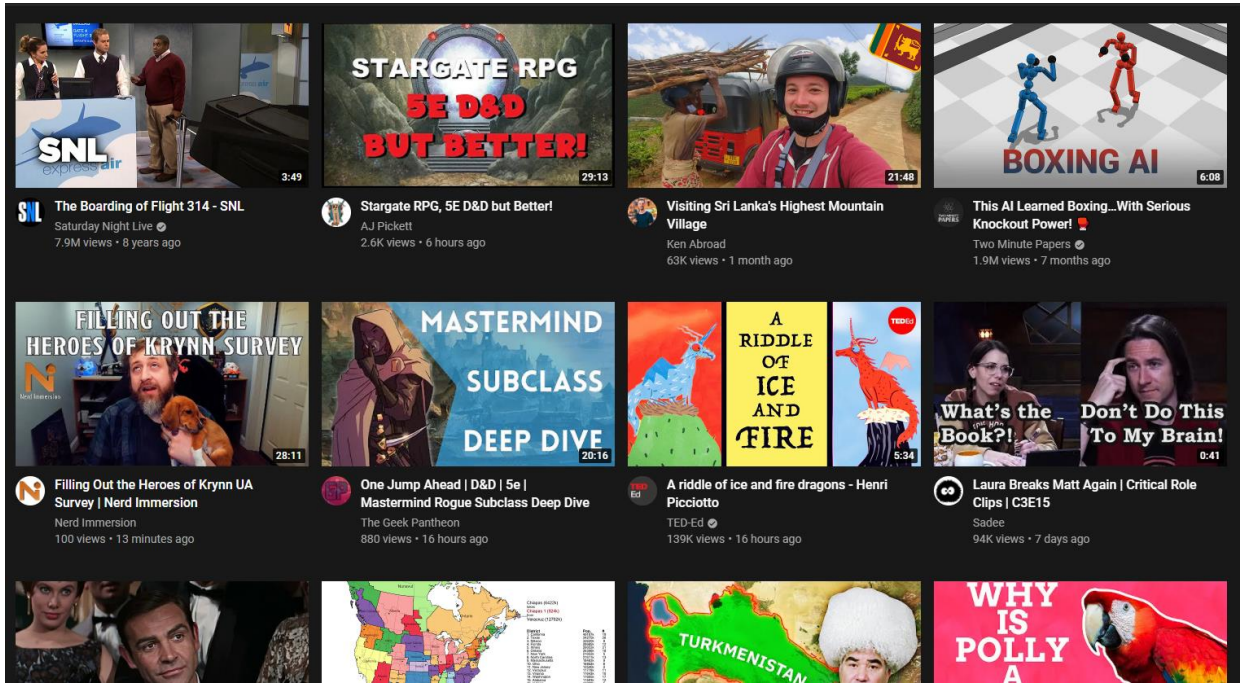


Supervised Learning: Classification

- In classification, we predict labels y (classes) for inputs x
- Binary Classification
  - Given: a set of m examples (x_i, y_i) $i = 1,2, \dots, m$ sampled from some distribution D, where $x_i \in R^n$ and $y_i \in \{-1, +1\}$.
  - Find: a function f, $f: R^n \rightarrow \{-1, +1\}$which classifies examples x_j sampled from D well.
- Comments
  - The function f is usually a statistical model, whose parameters are learnt from the set of examples.
  - Thy set of examples are called ‘training‘ se’
  - y is called – ‘target variable’ or ‘target’
  - Examples with $y_i = +1$ are called ‘positive examples’
  - Examples with $y_i = -1$ are called ‘negative examples’

Supervised Learning: Classification

- Examples:
  - OCR (input: images, classes: characters)
  - Medical diagnosis (input: symptoms, classes: diseases)
  - Automatic essay grader (input: document, classes: grades)
  - Fraud detection (input: account activity, classes: fraud / no fraud)
  - Customer service email routing
  - Recommended articles in a newspaper, recommended books
  - DNA and protein sequence identification
  - Categorization and identification of astronomical images
  - Financial investments
  - … many more



Supervised Learning: Regression

- Estimation of relationships between a dependent variable and one or more independent variables.

- Linear Model Assumptions
  - The dependent and independent variables show a linear relationship between the slope and the intercept.
  - The independent variable is not random.
  - The value of the residual (error) is zero.
  - The value of the residual (error) is constant across all observations.
  - The value of the residual (error) is not correlated across all observations.
  - The residual (error) values follow the normal distribution.

- Simple Linear Regression: $Y = a + bX_1 + \epsilon$

- Multiple Linear Regression: $Y = a + bX_1 + cX_2 + dX_3 + \epsilon$

- Y – Dependent variable
- X_1, X_2, X_3 – Independent (explanatory) variables
- a – Intercept
- b, c, d – Slopes
- ϵ – Residual (error)

Inductive learning Method

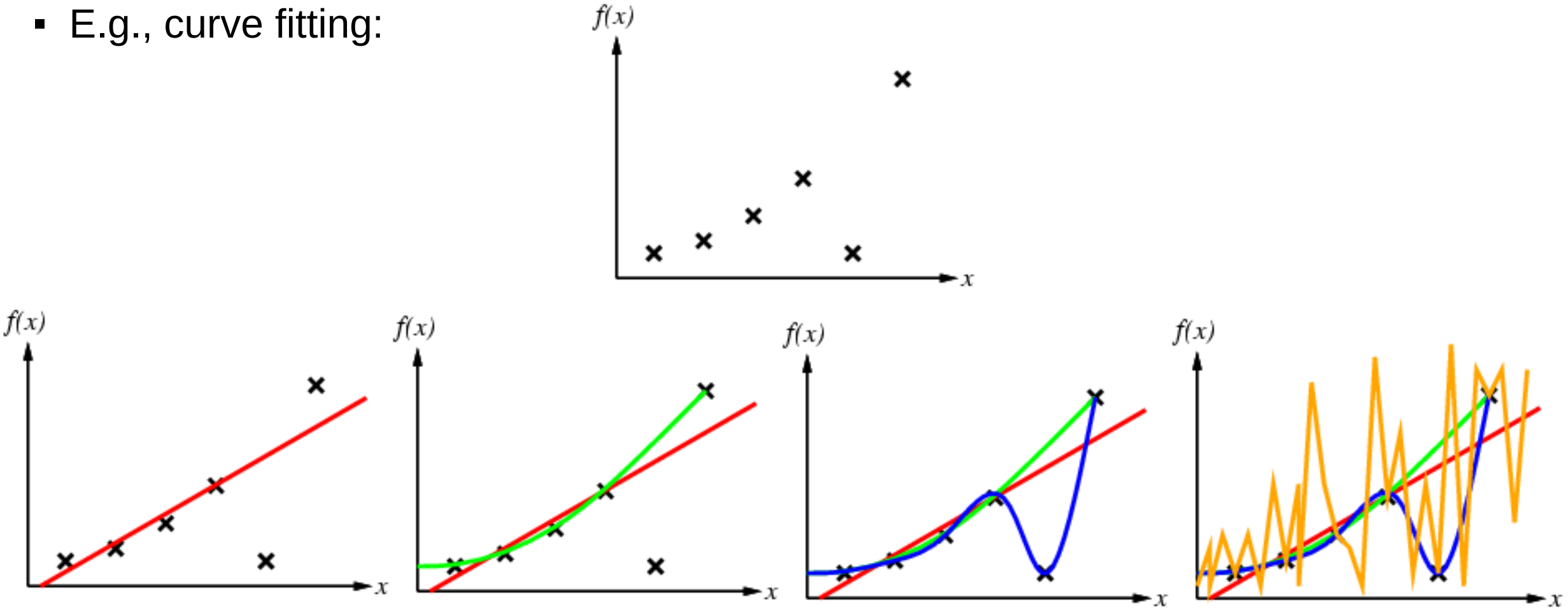
- Simplest form: learn a function from example
  - f is the target function
  - An example is a pair $(x, f(x))$

- **Pure induction task:**
  - **Given a collection of examples of f, return a function h that approximates f.**
  - find a hypothesis h, such that $h \approx f$, given a training set of examples
  - Construct/adjust h to agree with f on training set
  - h is consistent if it agrees with f on all examples

- This is a highly simplified model of real learning:
  - Ignores prior knowledge
  - Assumes examples are given

Inductive learning Method: Regression Example

- E.g., curve fitting:



Ockham's razor: prefer the simplest hypothesis consistent with data

Generalization

- Hypotheses must generalize to correctly classify instances not in the training data.
- Simply memorizing training examples is a consistent hypothesis that does not generalize.
- *Occam's razor*:
  - Finding a *simple* hypothesis helps ensure generalization.

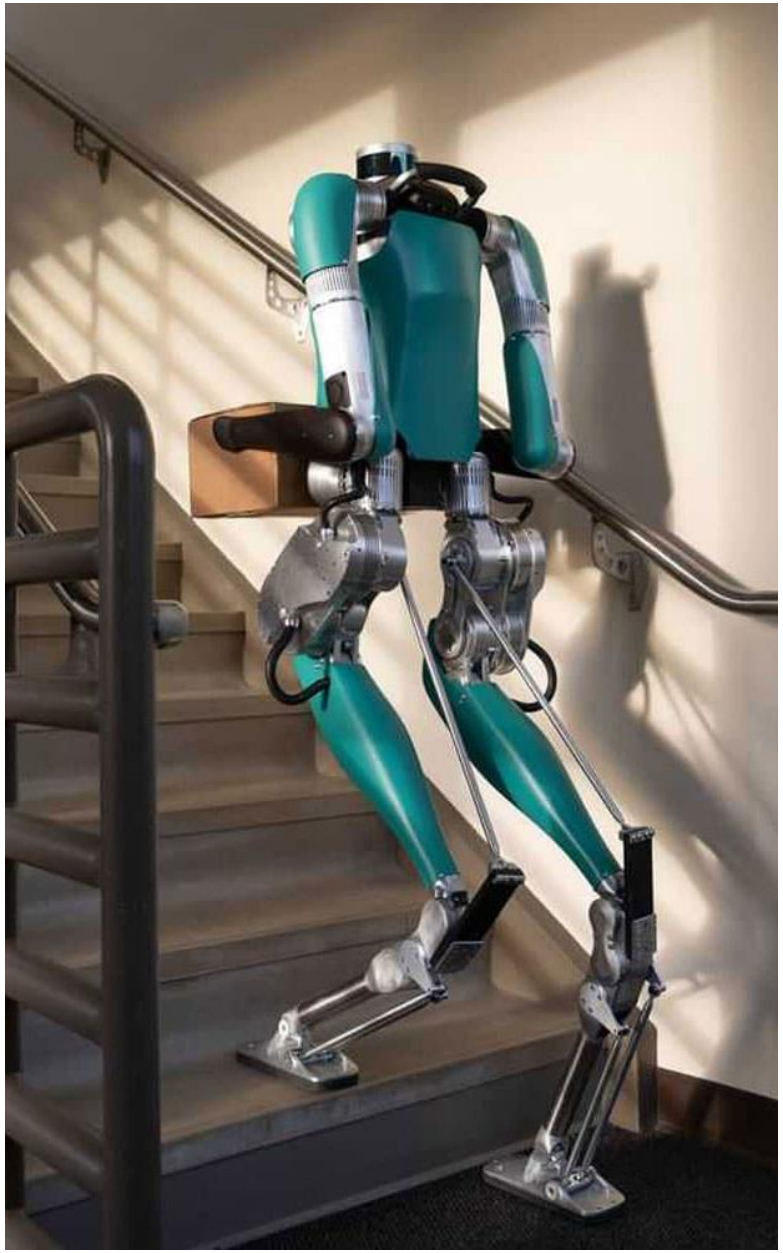
OCCAM'S RAZOR

"WHEN FACED WITH TWO POSSIBLE EXPLANATIONS, THE SIMPLER OF THE TWO IS THE ONE MOST LIKELY TO BE TRUE."

22

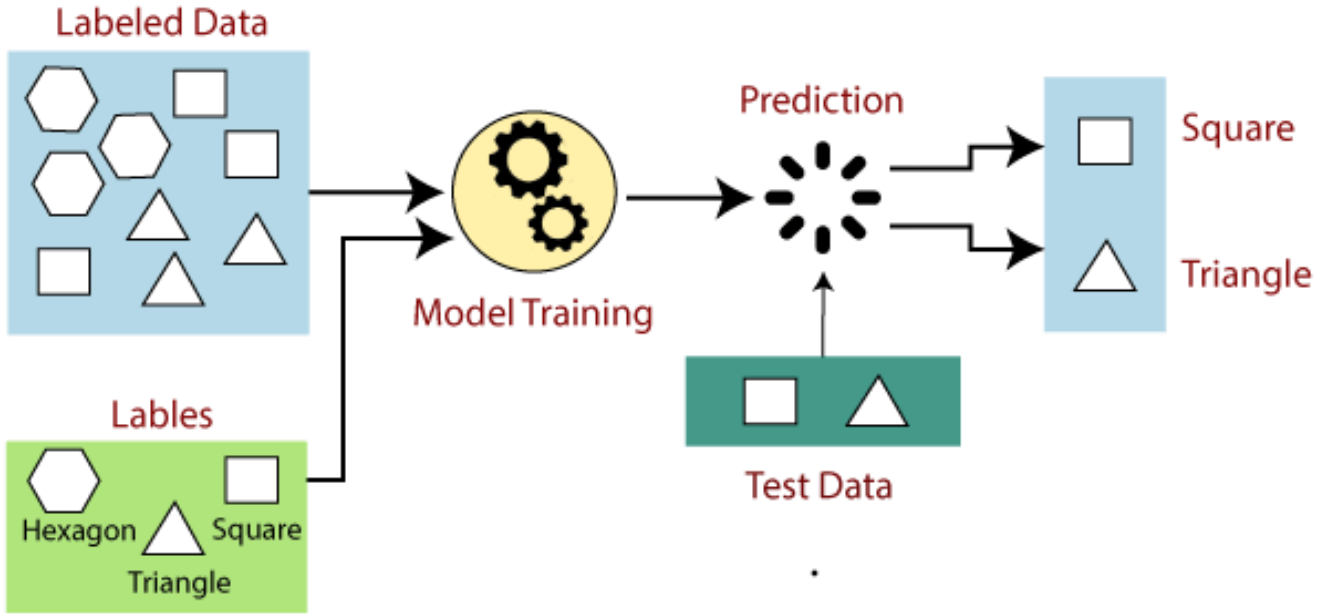
Supervised Learning Steps

- Determine the type of the training examples. Before doing anything else, the user should decide what kind of data is to be used as a training set.
- Gather training set. Thus, a set of input objects is gathered and corresponding outputs are also gathered, either from human experts or from measurements.
- Determine the structure of the learned function and corresponding learning algorithm.
- Complete the design. Run the learning algorithm on the gathered training set.
- Evaluate the accuracy of the learned function. After the parameter adjustment and learning, the performance of the resulting function should be measured on a test set that is separate from the training set.

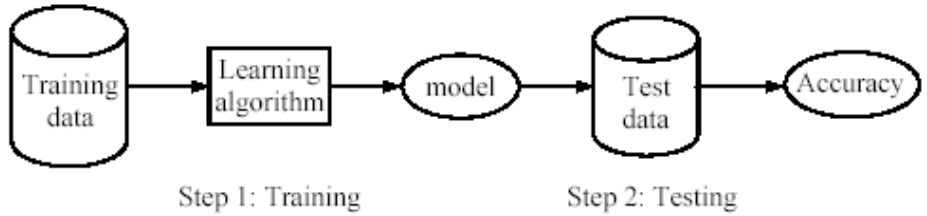


Inductive learning Method: Classification

- Data: A set of data records (also called examples, instances or cases) described by
  - *k* attributes: A_1, A_2, … A_k.
  - a class: Each example is labelled with a pre-defined class.
- Goal: To learn a classification model from the data that can be used to predict the classes of new (future, or test) cases/instances.



Classification: A Two-Step Process



- Model construction: describing a set of predetermined classes
  - Learning (training): Learn a model using the training data
  - Each tuple/sample is assumed to belong to a predefined class, as determined by the class label
  - The set of tuples used for model construction is training set
  - The model is represented as **classification rules**, **decision trees**, or **mathematical formulae**
  - $\text{Accuracy} = \frac{Number\ of\ Correct\ Classifications}{Total\ Number\ of\ Test\ Cases}$
- Model usage: for classifying future or unknown objects
  - Testing: Test the model using unseen test data to assess the model accuracy
  - Estimate accuracy of the model
    - The known label of test sample is compared with the classified result from the model
    - **Test set is independent of training set**, otherwise over-fitting will occur
  - If the accuracy is acceptable, use the model to classify data tuples whose class labels are not known

An Example: The Learning Task

- Learn a classification model from the data
- Use the model to classify future loan applications into
  - Yes (approved) and
  - No (not approved)
- What is the class for following case/instance?

Age	Has_Job	Own_house	Credit-Rating	Class
young	false	false	good	?

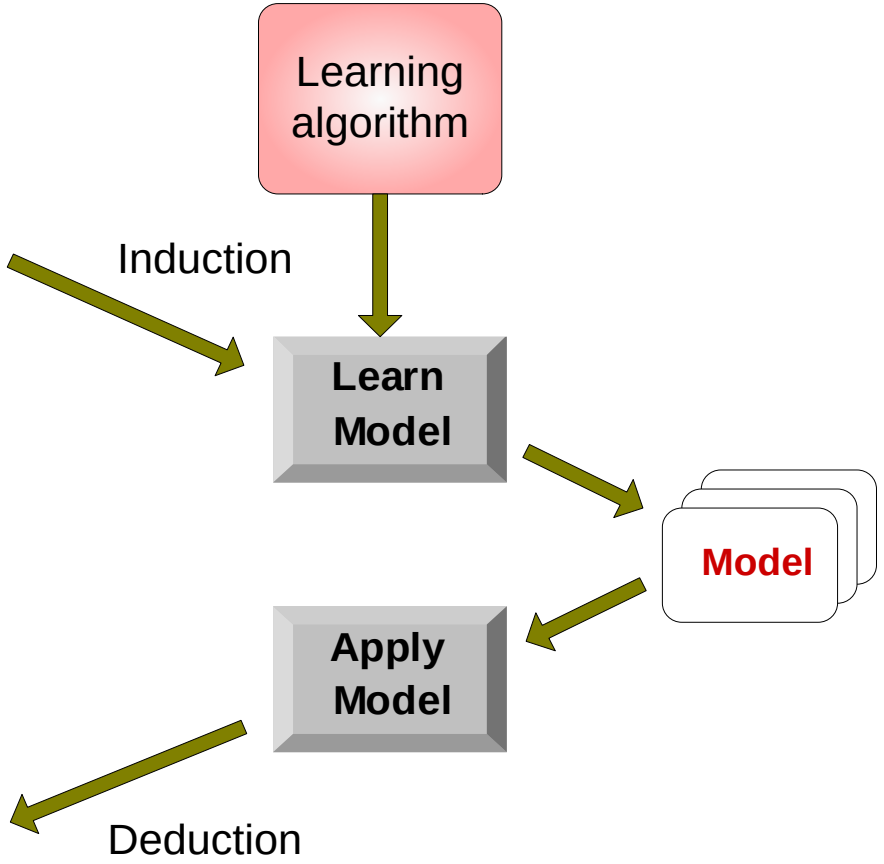
Illustrating Classification Task

Tid	Attrib1	Attrib2	Attrib3	Class
1	Yes	Large	125K	No
2	No	Medium	100K	No
3	No	Small	70K	No
4	Yes	Medium	120K	No
5	No	Large	95K	Yes
6	No	Medium	60K	No
7	Yes	Large	220K	No
8	No	Small	85K	Yes
9	No	Medium	75K	No
10	No	Small	90K	Yes

Training Set

Tid	Attrib1	Attrib2	Attrib3	Class
11	No	Small	55K	?
12	Yes	Medium	80K	?
13	Yes	Large	110K	?
14	No	Small	95K	?
15	No	Large	67K	?

Test Set



Evaluation of Classifiers

Evaluating classification methods

- **Predictive accuracy**

$$\text{Accuracy} = \frac{True\ Positive + TrueNegative}{Total}$$

- Efficiency
  - time to construct the model
  - time to use the model
- Robustness: handling noise and missing values
- Scalability: efficiency in disk-resident databases
- Interpretability:
  - understandable and insight provided by the model
- Compactness of the model: size of the tree, or the number of rules.

Evaluation methods

- **Holdout set**: The available data set *D* is divided into two disjoint subsets,
  - the *training set* D_{train} (for learning a model)
  - the *test set* D_{test} (for testing the model)
- **Important:** training set should not be used in testing and the test set should not be used in learning.
  - Unseen test set provides a unbiased estimate of accuracy.
- The test set is also called the holdout set. (the examples in the original data set *D* are all labeled with classes.)
- This method is mainly used when the data set *D* is large.

Evaluation methods (cont…)

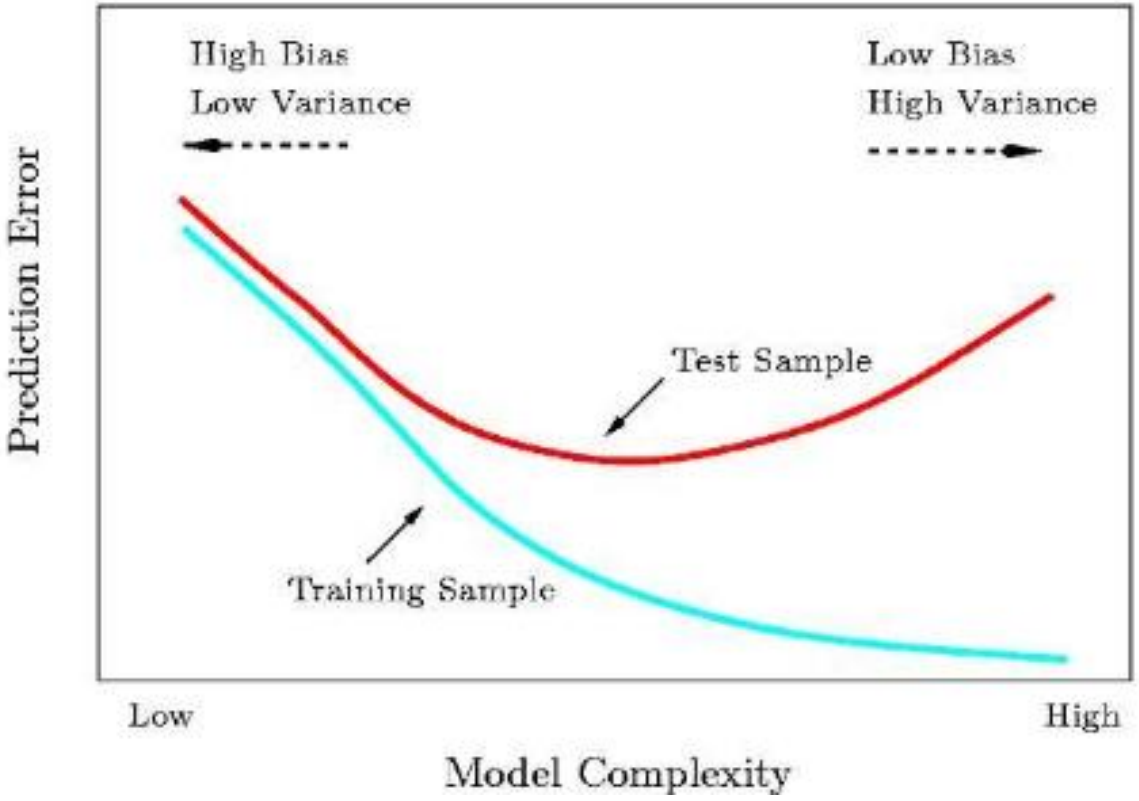
- **n-fold cross-validation**: The available data is partitioned into *n* equal-size disjoint subsets.
- Use each subset as the test set and combine the rest *n*-1 subsets as the training set to learn a classifier.
- The procedure is run *n* times, which give *n* accuracies.
- The final estimated accuracy of learning is the average of the *n* accuracies.
- 10-fold and 5-fold cross-validations are commonly used.
- This method is used when the available data is not large.

Evaluation methods (cont…)

- **Leave-one-out cross-validation**: This method is used when the data set is very small.
- It is a special case of cross-validation
- Each fold of the cross validation has only a single test example and all the rest of the data is used in training.
- If the original data has *m* examples, this is *m*-fold cross-validation

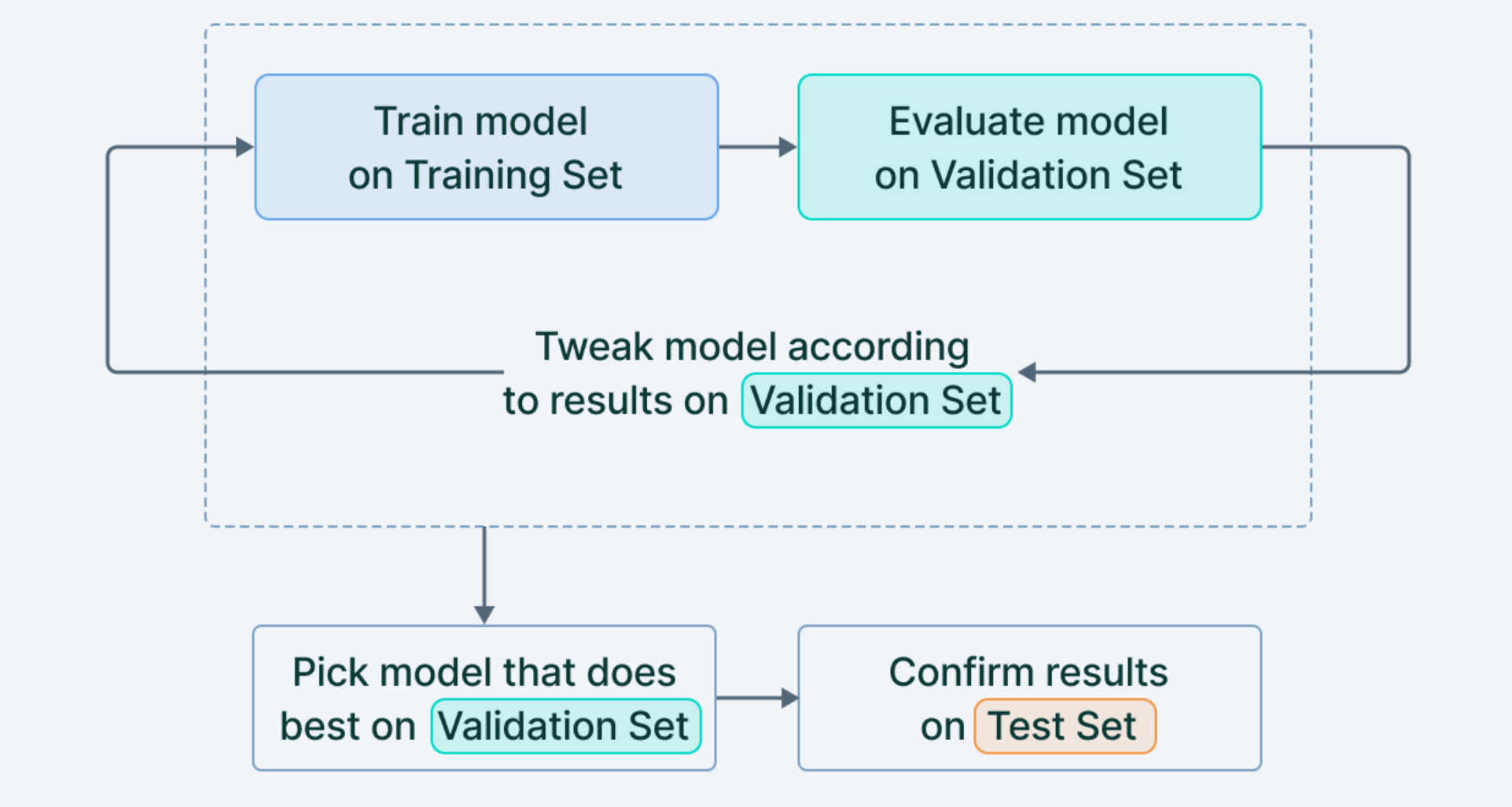
Evaluation methods (cont…)

- **Validation set**: the available data is divided into three subsets,
  - a training set,
  - a validation set and
  - a test set.
- A validation set is used frequently for estimating parameters in learning algorithms.



- In such cases, the values that give the best accuracy on the validation set are used as the final parameter values.
- Cross-validation can be used for parameter estimating as well.

Training, Validation, and Test



Classification measures

- Accuracy is only one measure (error = 1-accuracy).
- **Accuracy is not suitable in some applications**.
- In text mining, we may only be interested in the documents of a particular topic, which are only a small portion of a big document collection.
- In classification involving skewed or highly imbalanced data, e.g., network intrusion and financial fraud detections, we are interested only in the minority class.
  - High accuracy does not mean any intrusion is detected.
  - E.g., 1% intrusion. Achieve 99% accuracy by doing nothing.
- The class of interest is commonly called the **positive class**, and the rest **negative classes.**

Binary Classification: Evaluation

- Used in information retrieval and text classification.
- We use a **confusion matrix** to introduce them.

		Predicted	
		Negative	**Positive**
Actual	**Negative**	True Negative (TN)	False Positive (FP)
	Positive	False Negative (FN)	True Positive (TP)

- TP - True Positive – Number of **Positive** (correct) classifications of **Positive** examples
- FN - False Negative – Number of **Negative** (incorrect) classifications of **Positive** examples
- FP - False Positive – Number of **Positive** (incorrect) classifications of **Negative** examples
- TN - True Negative – Number of **Negative** (correct) classifications of **Negative** examples

Binary Classification: Evaluation

		Predicted	
		Negative	**Positive**
Actual	**Negative**	True Negative (TN)	False Positive (FP)
	Positive	False Negative (FN)	True Positive (TP)

- Precision *p* is the number of correctly classified positive examples divided by the total number of examples that are classified as positive.
- $\mathrm{P} = \mathrm{Precision} = \frac{True\ Positive}{True\ Positive + False\ Positive} = \frac{TP}{TP + FP}$
- Recall *r* is the number of correctly classified positive examples divided by the total number of actual positive examples in the test set.
- $\mathrm{R} = \mathrm{Recall} = \frac{True\ Positive}{True\ Positive + False\ Negative} = \frac{TP}{TP + FN}$

An example

- This confusion matrix gives
  - precision p = 100% and
  - recall r = 1%

  because we only classified one positive example correctly and no negative examples wrongly.

- Note: precision and recall only measure classification on the positive class.

	Classified Positive	Classified Negative
Actual Positive	1	99
Actual Negative	0	1000

F_1-value (also called F_1-score)

- It is hard to compare two classifiers using two measures. F_1 score combines precision and recall into one measure

$$F_1 = \frac{2pr}{p + r}$$

F_1-score is the harmonic mean of precision and recall.

$$F_1 = \frac{2}{\frac{1}{p} + \frac{1}{r}}$$

- The harmonic mean of two numbers tends to be closer to the smaller of the two.
- For F_1-value to be large, both *p* and *r* much be large.

Binary Classification: Evaluation

		Predicted	
		Negative	**Positive**
Actual	**Negative**	True Negative (TN)	False Positive (FP)
	Positive	False Negative (FN)	True Positive (TP)

- Accuracy $= \frac{True\ Positive + TrueNegative}{Total}$
- Precision $= \frac{True\ Positive}{True\ Positive + False\ Positive}$
- Recall $= \frac{True\ Positive}{True\ Positive + False\ Negative}$
- $F1 = 2 \times \frac{\text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}}$

Some Supervised Learning Algorithms

k-Nearest Neighbor Classification (kNN)

- Unlike most learning methods, kNN does not build model from the training data.
- To classify a test instance *d*, define *k*-neighborhood *P* as *k* nearest neighbors of *d*
- Count number *n* of training instances in *P* that belong to class c_j
- Estimate Pr(c_j|*d*) as *n*/*k*
- No training is needed.
- Classification time is linear in training set size for each test case.

kNNAlgorithm

Algorithm kNN(D, d, k)

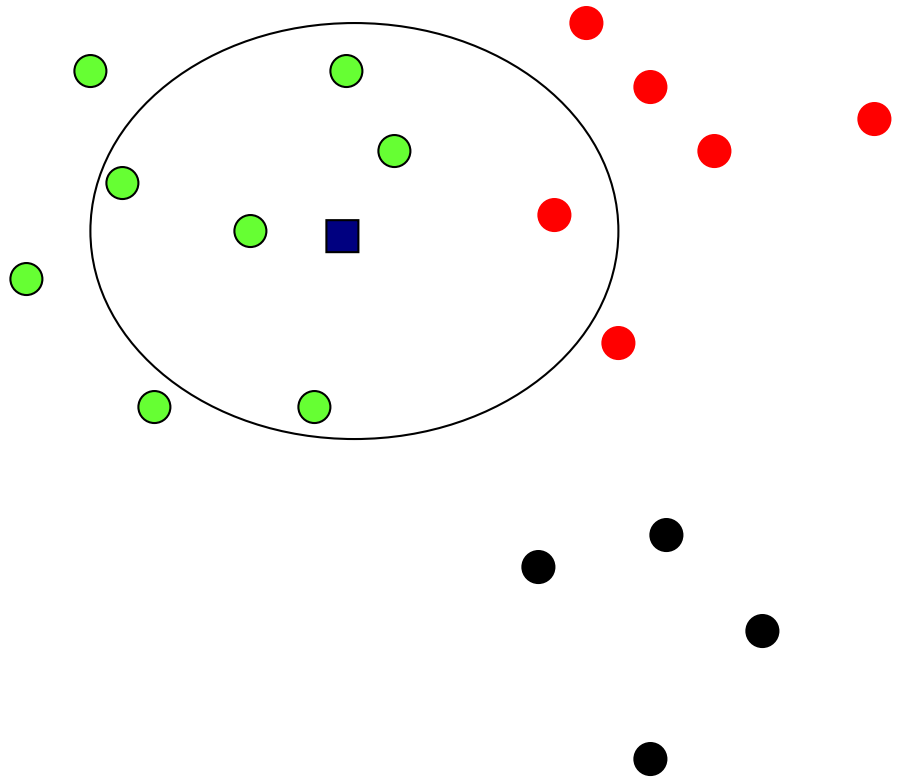
1 Compute the distance between d and every example in D;

2 Choose the k examples in D that are nearest to d, denote the set by P ($\subseteq D$);

3 Assign d the class that is the most frequent class in P (or the majority class);

- k is usually chosen empirically via a validation set or cross-validation by trying a range of k values.
- Distance function is crucial, but depends on applications.

Example: k=6 (6NN)

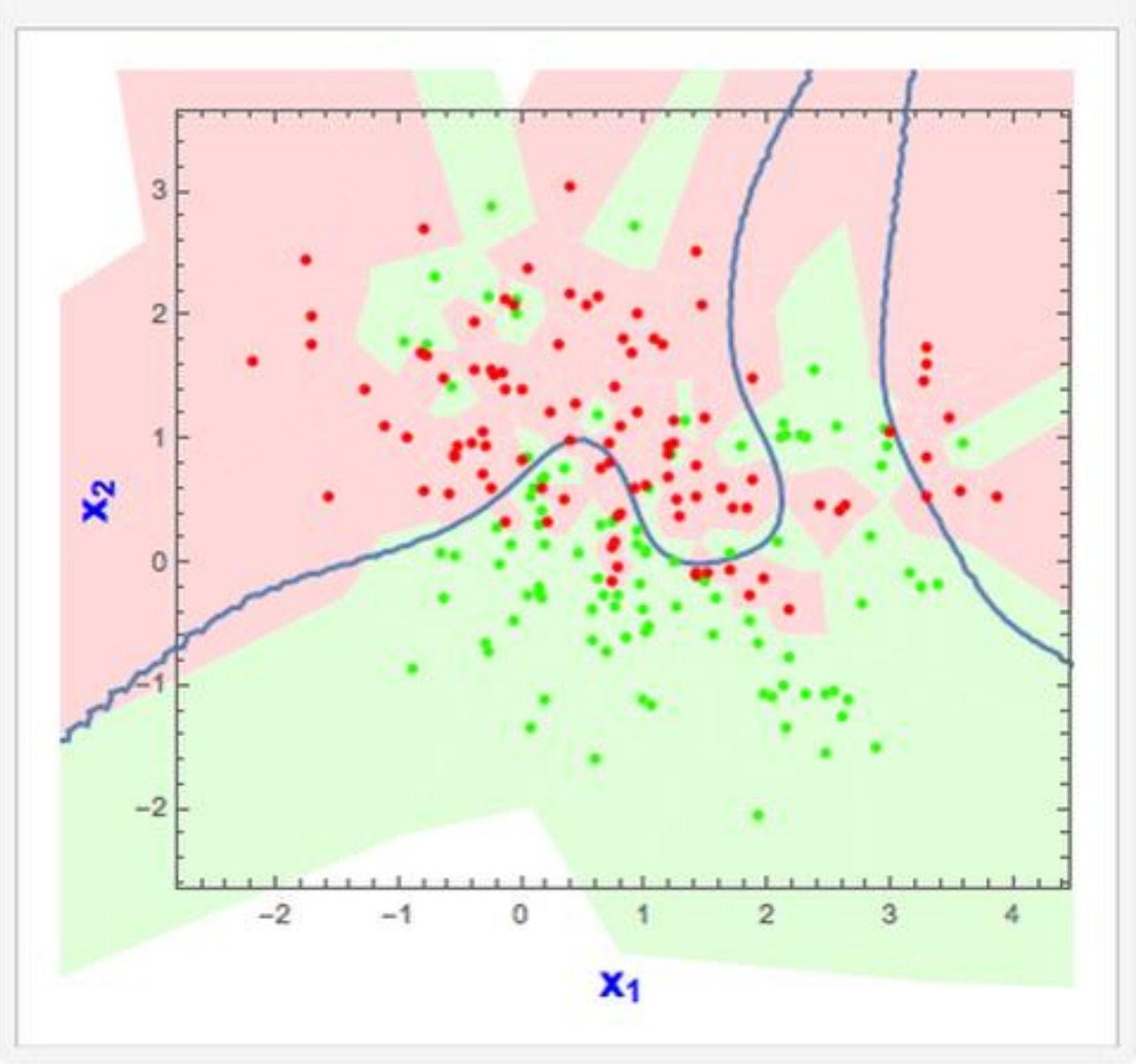


- Government
- Science
- Arts

A new point ■
Pr(science| ■)?

Discussions

- kNN can deal with complex and arbitrary decision boundaries.
- Despite its simplicity, researchers have shown that the classification accuracy of kNN can be quite strong and in many cases as accurate as those elaborated methods.
- kNN is slow at the classification time
- kNN does not produce an understandable model



Decision Trees Introduction

- Decision tree learning is one of the most widely used techniques for classification.
  - Its classification accuracy is competitive with other methods, and
  - it is very efficient.
- The classification model is a tree, called decision tree.
- C4.5 by Ross Quinlan is perhaps the best known system. It can be downloaded from the Web.

Learning decision trees

Example Problem: decide whether to wait for a table at a restaurant, based on the following attributes:

1. **Alternate**: is there an alternative restaurant nearby?
2. **Bar**: is there a comfortable bar area to wait in?
3. **Fri/Sat**: is today Friday or Saturday?
4. **Hungry**: are we hungry?
5. **Patrons**: number of people in the restaurant (None, Some, Full)
6. **Price**: price range ($, $$, $$$)
7. **Raining**: is it raining outside?
8. **Reservation**: have we made a reservation?
9. **Type**: kind of restaurant (French, Italian, Thai, Burger)
10. **WaitEstimate**: estimated waiting time (0-10, 10-30, 30-60, >60)

Feature(Attribute)-based representations

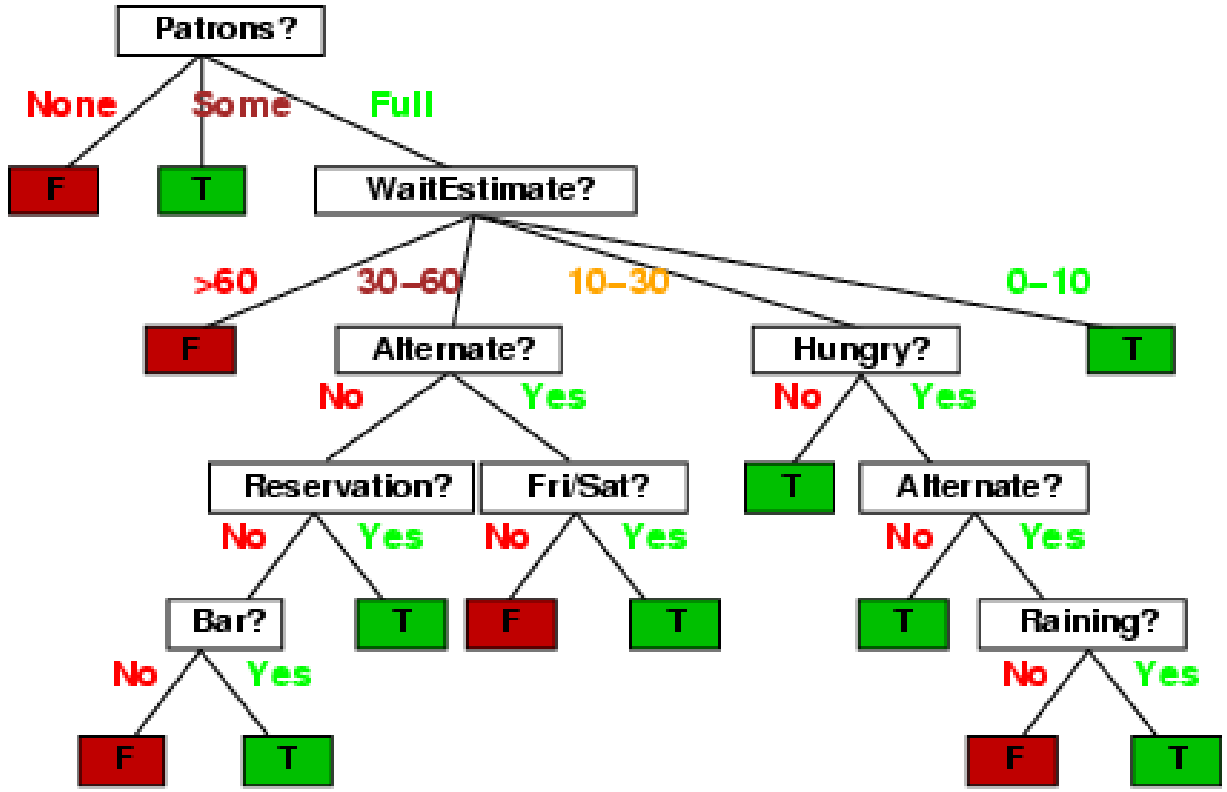
- Examples described by feature(attribute) values
  - (Boolean, discrete, continuous)
- E.g., situations where I will/won't wait for a table:

Example	Attributes										Target
	Alt	*Bar*	*Fri*	*Hun*	*Pat*	*Price*	*Rain*	*Res*	*Type*	*Est*	*Wait*
X_1	T	F	F	T	Some	$$$	F	T	French	0–10	T
X_2	T	F	F	T	Full	$	F	F	Thai	30–60	F
X_3	F	T	F	F	Some	$	F	F	Burger	0–10	T
X_4	T	F	T	T	Full	$	F	F	Thai	10–30	T
X_5	T	F	T	F	Full	$$$	F	T	French	>60	F
X_6	F	T	F	T	Some	$$	T	T	Italian	0–10	T
X_7	F	T	F	F	None	$	T	F	Burger	0–10	F
X_8	F	F	F	T	Some	$$	T	T	Thai	0–10	T
X_9	F	T	T	F	Full	$	T	F	Burger	>60	F
X_{10}	T	T	T	T	Full	$$$	F	T	Italian	10–30	F
X_{11}	F	F	F	F	None	$	F	F	Thai	0–10	F
X_{12}	T	T	T	T	Full	$	F	F	Burger	30–60	T

- Classification of examples is positive (T) or negative (F)

Decision trees

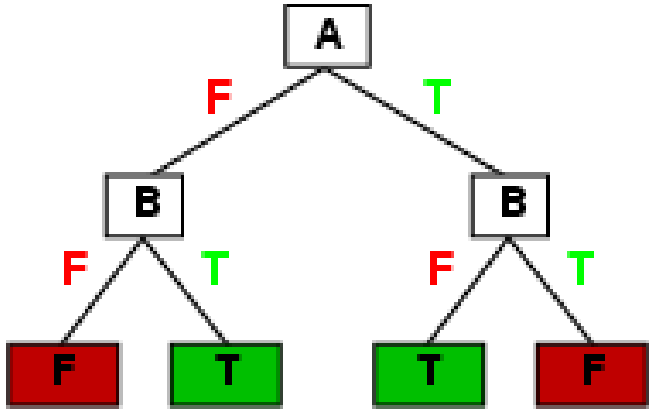
- One possible representation for hypotheses
- E.g., here is the “true” tree for deciding whether to wait:



Expressiveness

- Decision trees can express any function of the input attributes.
- E.g., for Boolean functions, truth table row → path to leaf:

A	B	A xor B
F	F	F
F	T	T
T	F	T
T	T	F



- Trivially, there is a consistent decision tree for any training set with one path to leaf for each example (unless *f* nondeterministic in *x*) but it probably won't generalize to new examples
- Prefer to find more compact decision trees

Decision tree learning

- Aim: find a small tree consistent with the training examples
- Idea: (recursively) choose "most significant" attribute as root of (sub)tree

```
function DTL(examples, attributes, default) returns a decision tree
    if examples is empty then return default
    else if all examples have the same classification then return the classification
    else if attributes is empty then return MODE(examples)
    else
        best ← CHOOSE-ATTRIBUTE(attributes, examples)
        tree ← a new decision tree with root test best
        for each value v_i of best do
            examples_i ← {elements of examples with best = v_i}
            subtree ← DTL(examples_i, attributes − best, MODE(examples))
            add a branch to tree with label v_i and subtree subtree
        return tree
```

Decision Tree Construction Algorithm

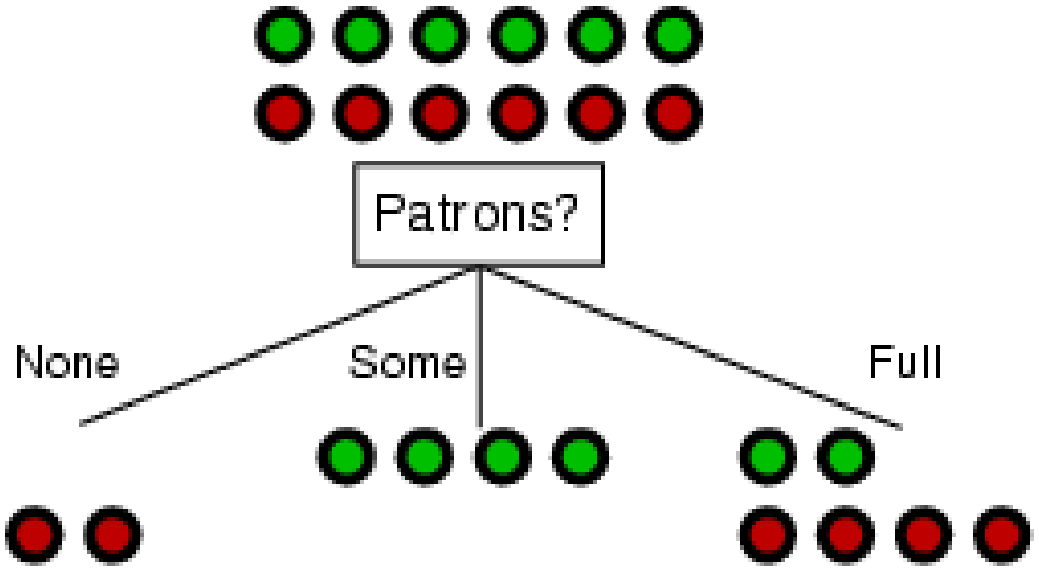
- **Principle**
  - Basic algorithm (adopted by ID3, C4.5 and CART): a **greedy algorithm**
  - Tree is constructed in a *top-down recursive divide-and-conquer* manner
- **Iterations**
  - At start, all the training tuples are at the root
  - Tuples are partitioned recursively based on selected attributes
  - Test attributes are selected on the basis of a heuristic or statistical measure (e.g, information gain)
- **Stopping conditions**
  - All samples for a given node belong to the same class
  - There are no remaining attributes for further partitioning – **majority voting** is employed for classifying the leaf
  - There are no samples left

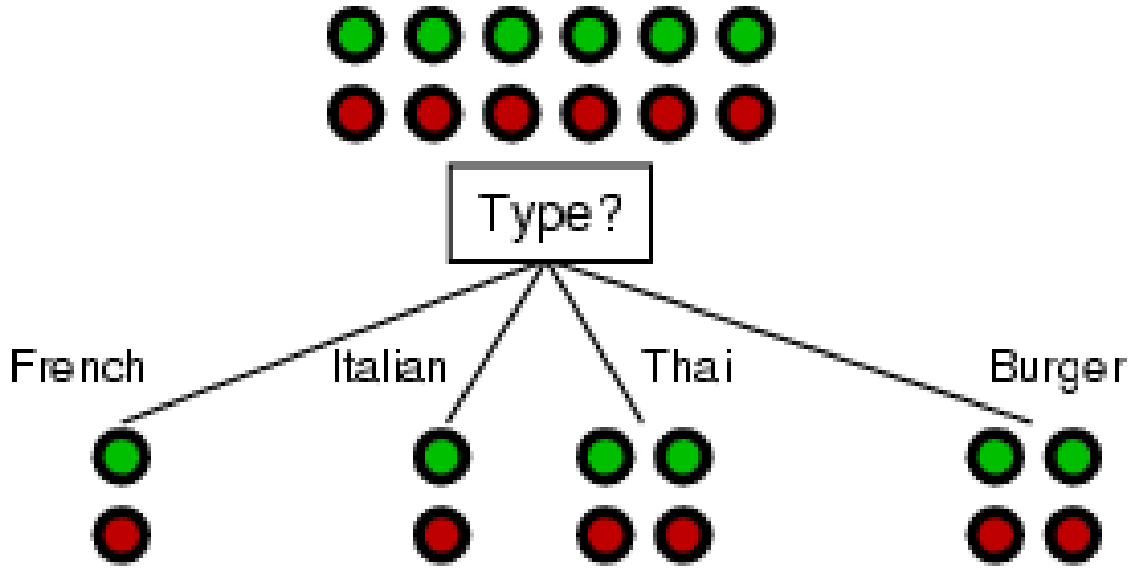
Tree Induction

- Greedy strategy.
  - Split the records based on an attribute test that optimizes certain criterion.
  - Nodes with homogeneous class distribution are preferred
- Issues
  - Determine how to split the records
    - How to specify the attribute test condition?
    - How to determine the best split?
  - Determine when to stop splitting

Choosing an attribute

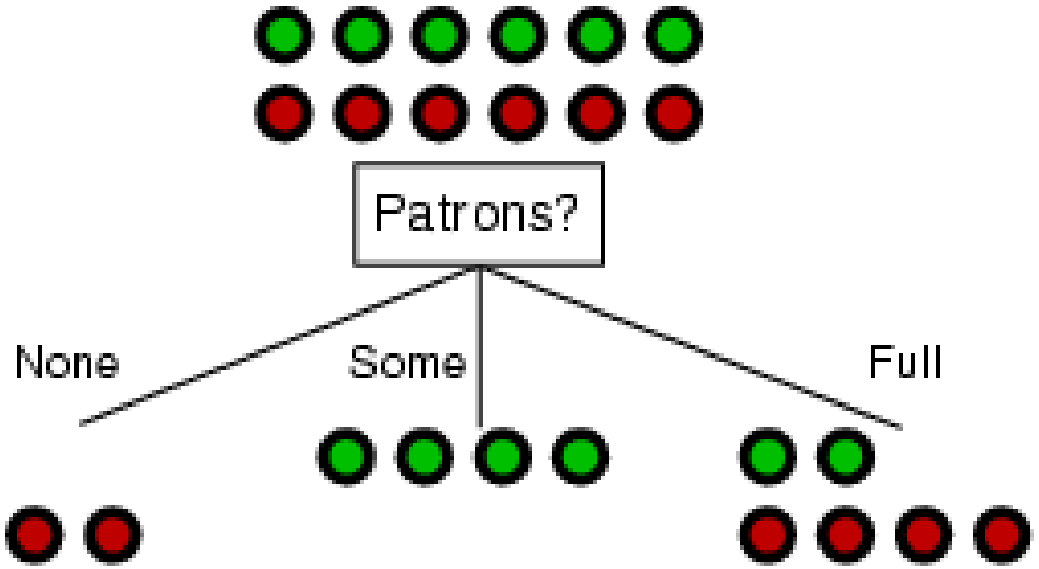
- Idea: a good attribute splits the examples into subsets that are (ideally) "all positive" or "all negative"

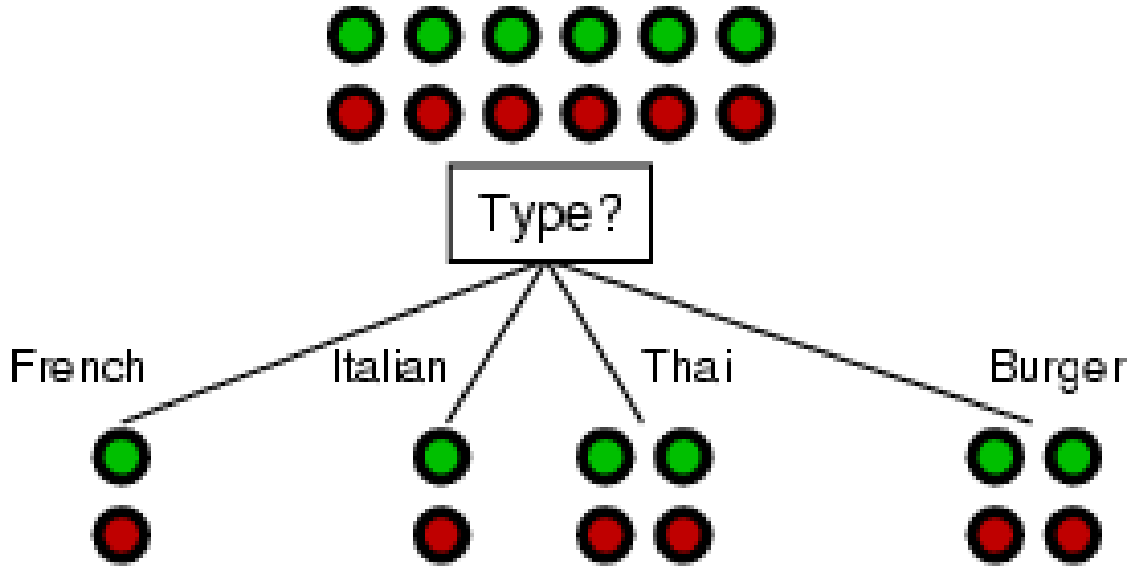




Choosing an attribute

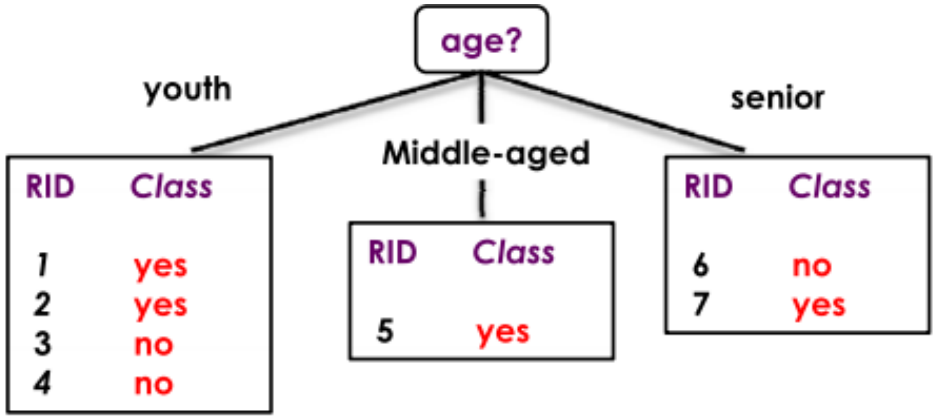
- Idea: a good attribute splits the examples into subsets that are (ideally) "all positive" or "all negative"





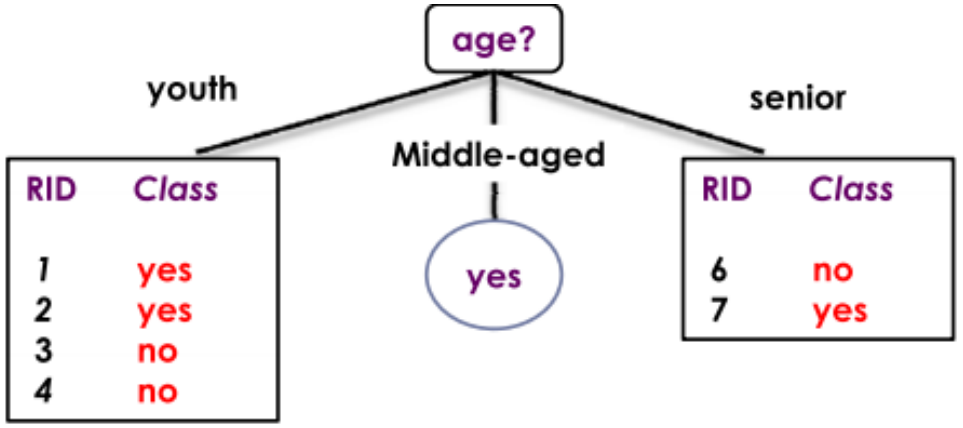
- *Patrons?* is a better choice

Example



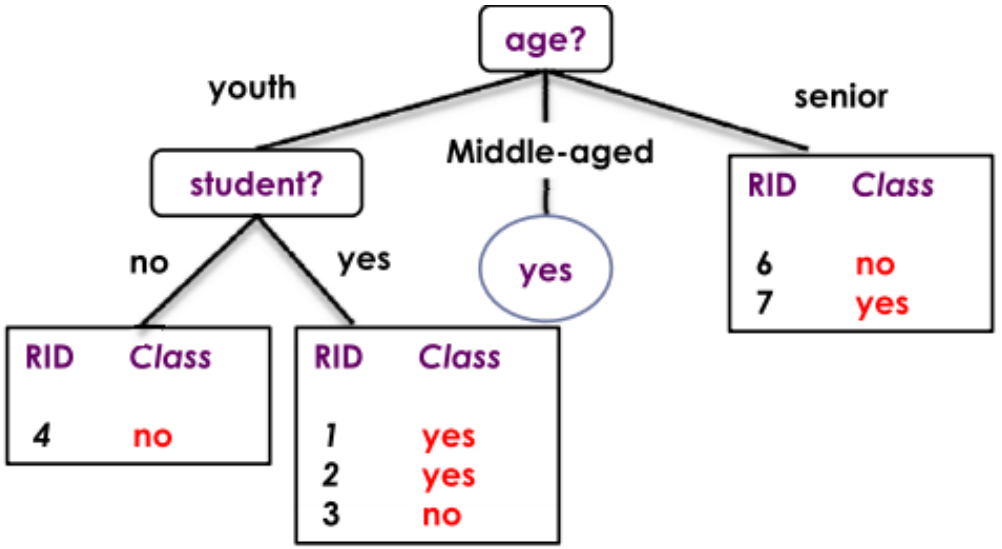
RID	age	student	credit-rating	Class: buys_computer
1	youth	yes	fair	yes
2	youth	yes	fair	yes
3	youth	yes	fair	no
4	youth	no	fair	no
5	middle-aged	no	excellent	yes
6	senior	yes	fair	no
7	senior	yes	excellent	yes

Example



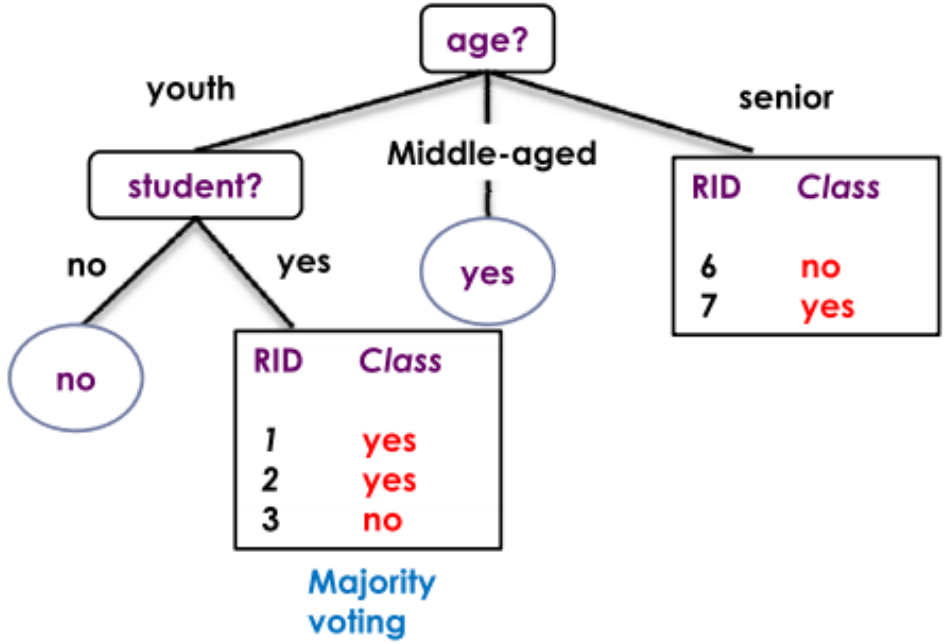
RID	age	student	credit-rating	Class: buys_computer
1	youth	yes	fair	yes
2	youth	yes	fair	yes
3	youth	yes	fair	no
4	youth	no	fair	no
5	middle-aged	no	excellent	yes
6	senior	yes	fair	no
7	senior	yes	excellent	yes

Example



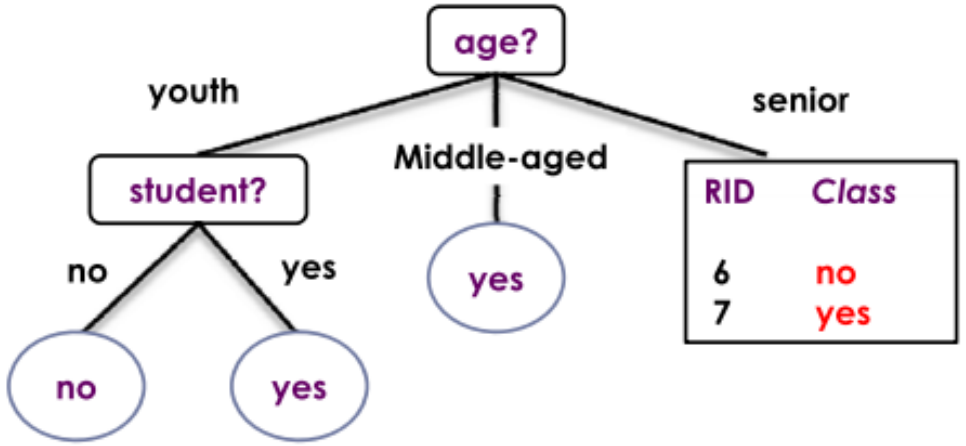
RID	age	student	credit-rating	Class: buys_computer
1	youth	yes	fair	yes
2	youth	yes	fair	yes
3	youth	yes	fair	no
4	youth	no	fair	no
5	middle-aged	no	excellent	yes
6	senior	yes	fair	no
7	senior	yes	excellent	yes

Example



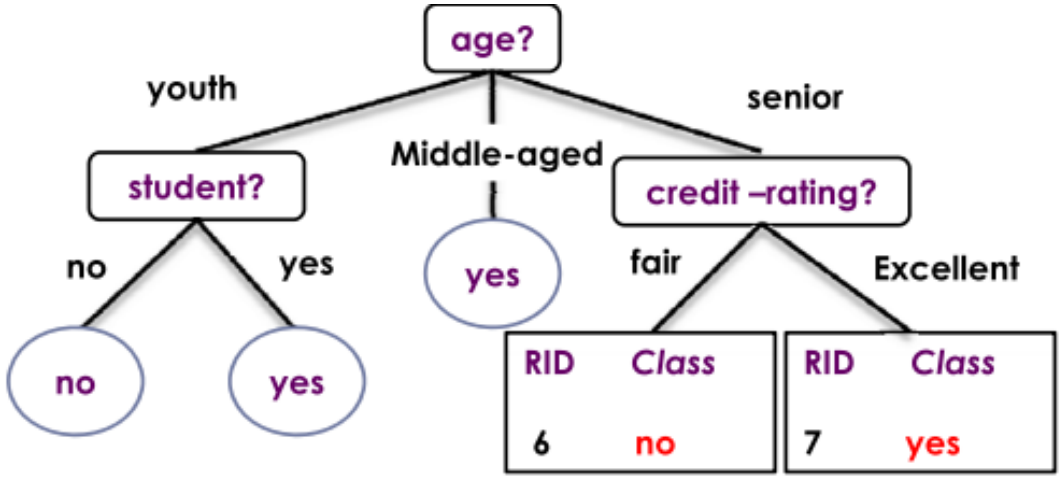
RID	age	student	credit-rating	Class: buys_computer
1	youth	yes	fair	yes
2	youth	yes	fair	yes
3	youth	yes	fair	no
4	youth	no	fair	no
5	middle-aged	no	excellent	yes
6	senior	yes	fair	no
7	senior	yes	excellent	yes

Example



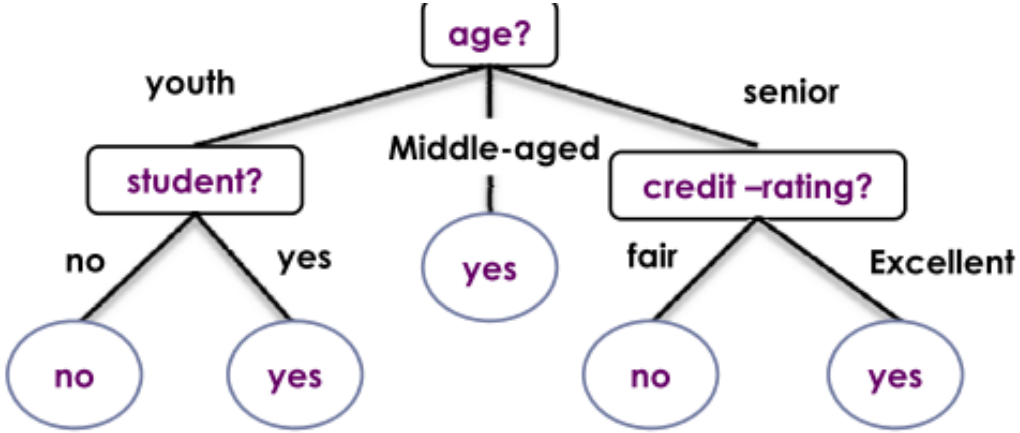
RID	age	student	credit-rating	Class: buys_computer
1	youth	yes	fair	yes
2	youth	yes	fair	yes
3	youth	yes	fair	no
4	youth	no	fair	no
5	middle-aged	no	excellent	yes
6	senior	yes	fair	no
7	senior	yes	excellent	yes

Example



RID	age	student	credit-rating	Class: buys_computer
1	youth	yes	fair	yes
2	youth	yes	fair	yes
3	youth	yes	fair	no
4	youth	no	fair	no
5	middle-aged	no	excellent	yes
6	senior	yes	fair	no
7	senior	yes	excellent	yes

Example



RID	age	student	credit-rating	Class: buys_computer
1	youth	yes	fair	yes
2	youth	yes	fair	yes
3	youth	yes	fair	no
4	youth	no	fair	no
5	middle-aged	no	excellent	yes
6	senior	yes	fair	yes
7	senior	yes	excellent	no

Measures of Node Impurity

- Information Gain
- Gini Index
- Misclassification error

Choose attributes to split to achieve minimum impurity

Attribute Selection Measure: Information Gain (ID3/C4.5)

- 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:

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

- 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

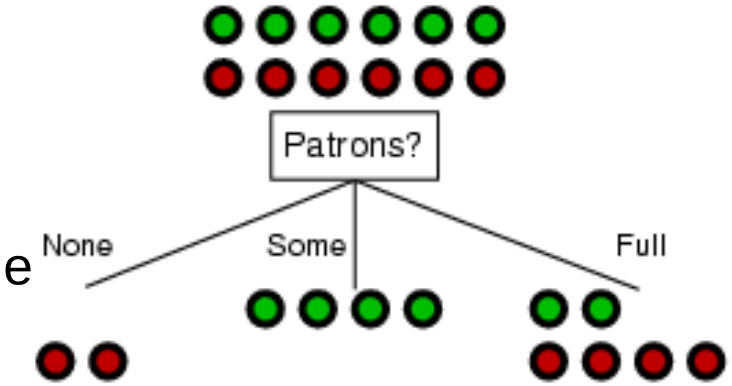
For the training set, $p = n = 6$, *I(6/12, 6/12) = 1* bit

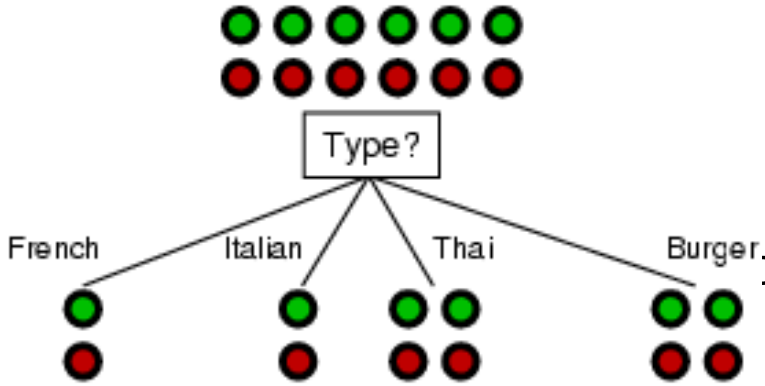
Consider the attributes *Patrons* and *Type* (and others too):

$$IG(Patrons) = 1 - [\frac{2}{12} I(0,1) + \frac{4}{12} I(1,0) + \frac{6}{12} I(\frac{2}{6}, \frac{4}{6})] = .0541 \text{ bits}$$

$$IG(Type) = 1 - [\frac{2}{12} I(\frac{1}{2}, \frac{1}{2}) + \frac{2}{12} I(\frac{1}{2}, \frac{1}{2}) + \frac{4}{12} I(\frac{2}{4}, \frac{2}{4}) + \frac{4}{12} I(\frac{2}{4}, \frac{2}{4})] = 0 \text{ bits}$$

Patrons has the highe





oot

Decision Tree Based Classification

- Advantages:
  - **Easy** to construct/implement
  - Extremely **fast** at classifying unknown records
  - Models are **easy to interpret for small-sized trees**
  - **Accuracy is comparable** to other classification techniques for many simple data sets
  - Tree models make no assumptions about the distribution of the underlying data : **nonparametric**
  - Have a **built-in feature selection** method that makes them immune to the presence of useless variables
- Disadvantages
  - Computationally **expensive to train**
  - **Some decision trees can be overly complex** that do not generalise the data well.
  - **Less expressivity**: There may be concepts that are hard to learn with limited decision trees

References

- CS583 - Chapter 3: Supervised Learning by Bing Liu
- “Supervised Learning” by Amar Tripathi
- “Learning from Observations” Nazli Ikizler-cinbis
- “What is Regression Analysis?” Corporate Finance Institute.
- “Supervised Machine Learning” by javatpoint
- “k-Nearest Neighbor (kNN) Classifier” by WOLFRAM Demonstrations Project

Two More Supervised Learning Algorithms

(Not Covered in Class)

Naïve Bayesian Classification

- Probabilistic view: Supervised learning can naturally be studied from a probabilistic point of view.
- Let A_1 through A_k be attributes with discrete values. The class is C.
- Given a test example *d* with observed attribute values a_1 through a_k.
- Classification is basically to compute the following posteriori probability. The prediction is the class c_j such that

$$\Pr(C = c_j \mid A_1 = a_1, \ldots, A_{|A|} = a_{|A|})$$

is maximal

Apply Bayes' Rule

$$\Pr(C = c_j \mid A_1 = a_1, \ldots, A_{|A|} = a_{|A|})$$

$$= \frac{\Pr(A_1 = a_1, \ldots, A_{|A|} = a_{|A|} \mid C = c_j)\Pr(C = c_j)}{\Pr(A_1 = a_1, \ldots, A_{|A|} = a_{|A|})}$$

$$= \frac{\Pr(A_1 = a_1, \ldots, A_{|A|} = a_{|A|} \mid C = c_j)\Pr(C = c_j)}{\sum_{r=1}^{|C|} \Pr(A_1 = a_1, \ldots, A_{|A|} = a_{|A|} \mid C = c_r)\Pr(C = c_r)}$$

- $\Pr(C=c_j)$ is the class *prior* probability: easy to estimate from the training data.

Computing probabilities

- The denominator P(A1=a1,...,Ak=ak) is irrelevant for decision making since it is the same for every class.
- We only need P(A1=a1,...,Ak=ak | C=ci), which can be written as
- Pr(A1=a1|A2=a2,...,Ak=ak, C=cj)* Pr(A2=a2,...,Ak=ak |C=cj)
- Recursively, the second factor above can be written in the same way, and so on.
- Now an assumption is needed.

Conditional independence assumption

- All attributes are conditionally independent given the class $C = c_j$.
- Formally, we assume,

$$\Pr(A_1=a_1 \mid A_2=a_2, ..., A_{|A|}=a_{|A|}, C=c_j) = \Pr(A_1=a_1 \mid C=c_j)$$

and so on for A_2 through $A_{|A|}$. I.e.,

$$\Pr(A_1 = a_1,...,A_{|A|} = a_{|A|} \mid C = c_i) = \prod_{i=1}^{|A|} \Pr(A_i = a_i \mid C = c_j)$$

Final naïve Bayesian classifier

- We are done!
- How do we estimate P($A_i = a_i | C=c_j$)? Easy!.

$$\Pr(C = c_j \mid A_1 = a_1, \ldots, A_{|A|} = a_{|A|})$$

$$= \frac{\Pr(C = c_j)\prod_{i=1}^{|A|} \Pr(A_i = a_i \mid C = c_j)}{\sum_{r=1}^{|C|} \Pr(C = c_r)\prod_{i=1}^{|A|} \Pr(A_i = a_i \mid C = c_r)}$$

Classify a test instance

- If we only need a decision on the most probable class for the test instance, we only need the numerator as its denominator is the same for every class.
- Thus, given a test example, we compute the following to decide the most probable class for the test instance

$$c = \arg\max_{c_j} \Pr(c_j) \prod_{i=1}^{|A|} \Pr(A_i = a_i \mid C = c_j)$$

An example

- Compute all probabilities required for classification

A	B	C
m	b	t
m	s	t
g	q	t
h	s	t
g	q	t
g	q	f
g	s	f
h	b	f
h	q	f
m	b	f

$\Pr(C = t) = 1/2$, $\Pr(C = f) = 1/2$

$\Pr(A=m \mid C=t) = 2/5$ $\Pr(A=g \mid C=t) = 2/5$ $\Pr(A=h \mid C=t) = 1/5$

$\Pr(A=m \mid C=f) = 1/5$ $\Pr(A=g \mid C=f) = 2/5$ $\Pr(A=h \mid C=n) = 2/5$

$\Pr(B=b \mid C=t) = 1/5$ $\Pr(B=s \mid C=t) = 2/5$ $\Pr(B=q \mid C=t) = 2/5$

$\Pr(B=b \mid C=f) = 2/5$ $\Pr(B=s \mid C=f) = 1/5$ $\Pr(B=q \mid C=f) = 2/5$

Now we have a test example:

$A = m \quad B = q \quad C = ?$

An Example (cont …)

- For C = t, we have

$$\Pr(C=t)\prod_{j=1}^{2}\Pr(A_j=a_j\,|\,C=t)=\frac{1}{2}\times\frac{2}{5}\times\frac{2}{5}=\frac{2}{25}$$

- For class C = f, we have

$$\Pr(C=f)\prod_{j=1}^{2}\Pr(A_j=a_j\,|\,C=f)=\frac{1}{2}\times\frac{1}{5}\times\frac{2}{5}=\frac{1}{25}$$

- C = t is more probable. t is the final class.

Additional issues

- **Numeric attributes**: Naïve Bayesian learning assumes that all attributes are categorical. Numeric attributes need to be discretized.
- **Zero counts**: An particular attribute value never occurs together with a class in the training set. We need smoothing.
- **Missing values**: Ignored

$$\Pr(A_i = a_i \mid C = c_j) = \frac{n_{ij} + \lambda}{n_j + \lambda n_i}$$

On naïve Bayesian classifier

- Advantages:
  - Easy to implement
  - Very efficient
  - Good results obtained in many applications
- Disadvantages
  - Assumption: class conditional independence, therefore loss of accuracy when the assumption is seriously violated (those highly correlated data sets)

Support Vector Machines Introduction

- Support vector machines were invented by V. Vapnik and his co-workers in 1970s in Russia and became known to the West in 1992.
- SVMs are linear classifiers that find a hyperplane to separate two class of data, positive and negative.
- Kernel functions are used for nonlinear separation.
- SVM not only has a rigorous theoretical foundation, but also performs classification more accurately than most other methods in applications, especially for high dimensional data.
- It is perhaps the best classifier for text classification.

Basic concepts

- Let the set of training examples D be

$$\{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_r, y_r)\},$$

where $\mathbf{x}_i = (x_1, x_2, \ldots, x_n)$ is an **input vector** in a real-valued space $X \subseteq R^n$ and y_i is its **class label** (output value), $y_i \in \{1, -1\}$.

1: positive class and -1: negative class.

- SVM finds a linear function of the form (**w**: weight vector)

$$f(\mathbf{x}) = \langle \mathbf{w} \cdot \mathbf{x} \rangle + b$$

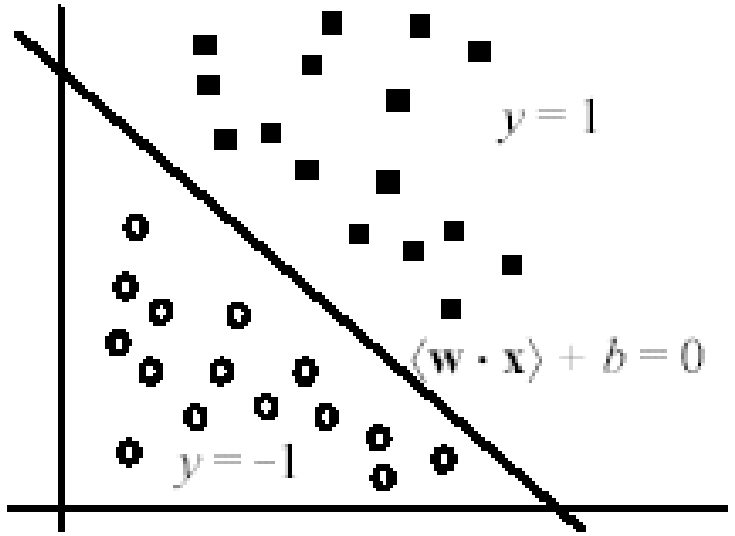
$$y_i = \begin{cases} 1 & if \langle \mathbf{w} \cdot \mathbf{x}_i \rangle + b \geq 0 \\ -1 & if \langle \mathbf{w} \cdot \mathbf{x}_i \rangle + b < 0 \end{cases}$$

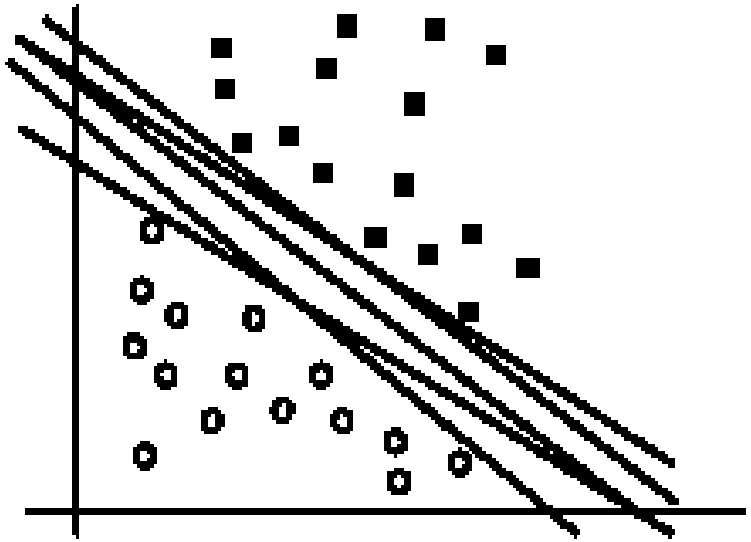
The hyperplane

- The hyperplane that separates positive and negative training data is

$$\langle \mathbf{w} \cdot \mathbf{x} \rangle + b = 0$$

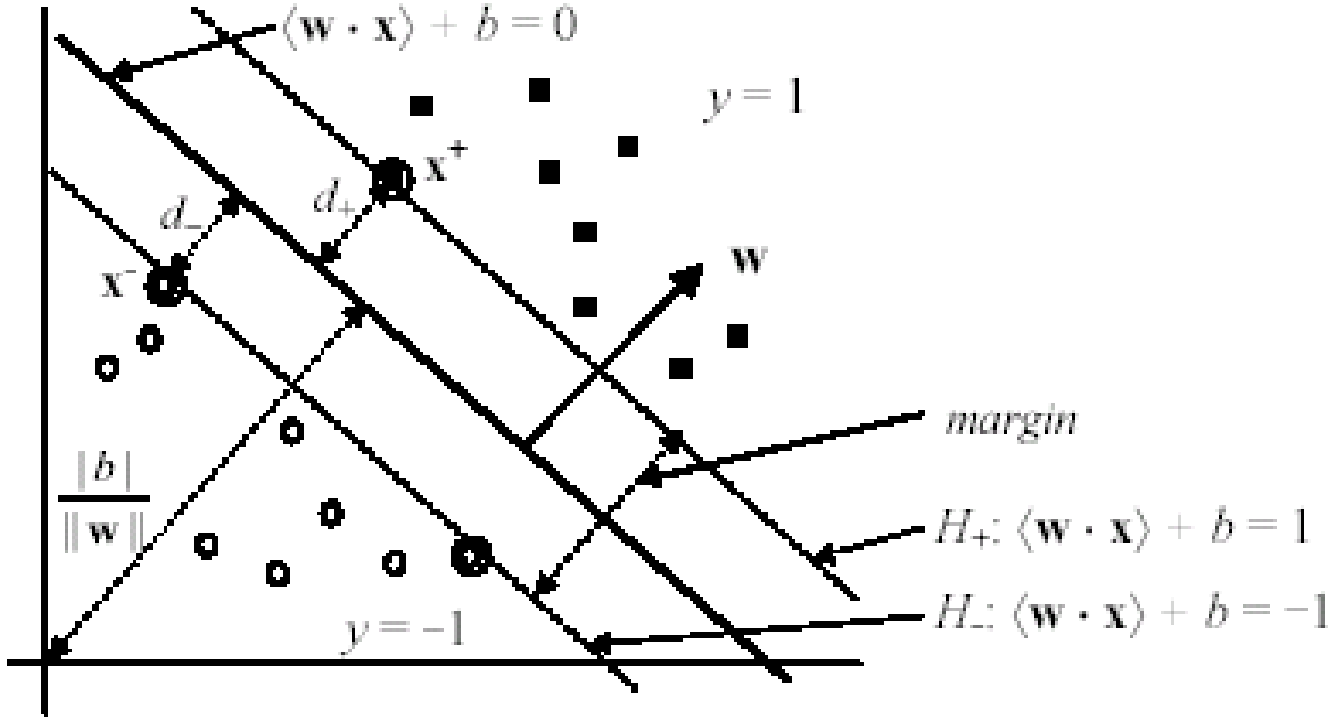
- It is also called the decision boundary (surface).
- So many possible hyperplanes, which one to choose?





Maximal margin hyperplane

- SVM looks for the separating hyperplane with the largest margin.
- Machine learning theory says this hyperplane minimizes the error bound



Linear SVM: separable case

- Assume the data are linearly separable.
- Consider a positive data point ($\mathbf{x}^+$, 1) and a negative ($\mathbf{x}^-$, -1) that are closest to the hyperplane

$$<\mathbf{w} \cdot \mathbf{x}> + b = 0.$$

- We define two parallel hyperplanes, H_+ and H_-, that pass through $\mathbf{x}^+$ and $\mathbf{x}^-$ respectively. H_+ and H_- are also parallel to $<\mathbf{w} \cdot \mathbf{x}> + b = 0$.

$$H_+: \quad \langle \mathbf{w} \cdot \mathbf{x}^+ \rangle + b = 1$$
$$H_-: \quad \langle \mathbf{w} \cdot \mathbf{x}^- \rangle + b = -1$$
$$\text{such that} \quad \langle \mathbf{w} \cdot \mathbf{x}_i \rangle + b \geq 1 \quad \text{if } y_i = 1$$
$$\langle \mathbf{w} \cdot \mathbf{x}_i \rangle + b \leq -1 \quad \text{if } y_i = -1,$$

Compute the margin

- Now let us compute the distance between the two margin hyperplanes H_+ and H_-. Their distance is the **margin** ($d_+ + d_-$ in the figure).
- Recall from vector space in algebra that the (perpendicular) **distance** from a point $\mathbf{x}_i$ to the hyperplane $\langle \mathbf{w} \cdot \mathbf{x} \rangle + b = 0$ is:

where $\|\mathbf{w}\|$ is the norm of $\mathbf{w}$,

$$\frac{|\langle \mathbf{w} \cdot \mathbf{x}_i \rangle + b|}{\|\mathbf{w}\|} \qquad (36)$$

$$\|\mathbf{w}\| = \sqrt{<\mathbf{w} \cdot \mathbf{w}>} = \sqrt{w_1^2 + w_2^2 + \ldots + w_n^2} \qquad (37)$$

Compute the margin (cont …)

- Let us compute d_+.
- Instead of computing the distance from $\mathbf{x}^+$ to the separating hyperplane $\langle \mathbf{w} \cdot \mathbf{x} \rangle + b = 0$, we pick up any point $\mathbf{x}_s$ on $\langle \mathbf{w} \cdot \mathbf{x} \rangle + b = 0$ and compute the distance from $\mathbf{x}_s$ to $\langle \mathbf{w} \cdot \mathbf{x}^+ \rangle + b = 1$ by applying the distance Eq. (36) and noticing $\langle \mathbf{w} \cdot \mathbf{x}_s \rangle + b = 0$,

$$d_+ = \frac{|\langle \mathbf{w} \cdot \mathbf{x}_s \rangle + b - 1|}{\| \mathbf{w} \|} = \frac{1}{\| \mathbf{w} \|} \tag{38}$$

$$margin = d_+ + d_- = \frac{2}{\| \mathbf{w} \|} \tag{39}$$

A optimization problem!

Definition (Linear SVM: separable case): Given a set of linearly separable training examples,

$D = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_r, y_r)\}$

Learning is to solve the following constrained minimization problem,

$$\text{Minimize}: \ \frac{\langle \mathbf{w} \cdot \mathbf{w} \rangle}{2} \qquad (40)$$

$$\text{Subject to}: \ y_i(\langle \mathbf{w} \cdot \mathbf{x}_i \rangle + b) \geq 1, \quad i = 1, 2, \ldots, r$$

$y_i(\langle \mathbf{w} \cdot \mathbf{x}_i \rangle + b \geq 1$, summarizes $i = 1, 2, \ldots, r$

$\langle \mathbf{w} \cdot \mathbf{x}_i \rangle + b \geq 1$ for $y_i = 1$

$\langle \mathbf{w} \cdot \mathbf{x}_i \rangle + b \leq -1$ for $y_i = -1$.

Solve the constrained minimization

- Standard Lagrangian method

$$L_P = \frac{1}{2}\langle \mathbf{w} \cdot \mathbf{w} \rangle - \sum_{i=1}^{r} \alpha_i [y_i (\langle \mathbf{w} \cdot \mathbf{x}_i \rangle + b) - 1] \qquad (41)$$

  where $\alpha_i \geq 0$ are the **Lagrange multipliers**.

- Optimization theory says that an optimal solution to (41) must satisfy certain conditions, called **Kuhn-Tucker conditions**, which are necessary (but not sufficient)
- Kuhn-Tucker conditions play a central role in constrained optimization.

Kuhn-Tucker conditions

$$\frac{\partial L_p}{\partial w_j} = w_j - \sum_{i=1}^{r} y_i \alpha_i \mathbf{x}_i = 0, \quad j = 1, 2, \ldots, m \tag{48}$$

$$\frac{\partial L_p}{\partial b} = -\sum_{i=1}^{r} y_i \alpha_i = 0 \tag{49}$$

$$y_i(\langle \mathbf{w} \cdot \mathbf{x}_i \rangle + b) - 1 \geq 0, \quad i = 1, 2, \ldots, r \tag{50}$$

$$\alpha_i \geq 0, \quad i = 1, 2, \ldots, r \tag{51}$$

$$\alpha_i (y_i(\langle \mathbf{w} \cdot \mathbf{x}_i \rangle + b) - 1) = 0, \quad i = 1, 2, \ldots, r \tag{52}$$

- Eq. (50) is the original set of constraints.
- The complementarity condition (52) shows that only those data points on the margin hyperplanes (i.e., H_+ and H_-) can have $\alpha_i > 0$ since for them $y_i(\langle \mathbf{w} \cdot \mathbf{x}_i \rangle + b) - 1 = 0$.
- These points are called the **support vectors**, All the other parameters $\alpha_i = 0$.

Solve the problem

- In general, Kuhn-Tucker conditions are necessary for an optimal solution, but not sufficient.
- However, for our minimization problem with a convex objective function and linear constraints, the Kuhn-Tucker conditions are both **necessary** and **sufficient** for an optimal solution.
- Solving the optimization problem is still a difficult task due to the inequality constraints.
- However, the Lagrangian treatment of the convex optimization problem leads to an alternative **dual** formulation of the problem, which is easier to solve than the original problem (called the **primal**).

Dual formulation

- From primal to a dual: Setting to zero the partial derivatives of the Lagrangian (41) with respect to the **primal variables** (i.e., **w** and *b*), and substituting the resulting relations back into the Lagrangian.
  - I.e., substitute (48) and (49), into the original Lagrangian (41) to eliminate the primal variables

$$L_D = \sum_{i=1}^{r} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{r} y_i y_j \alpha_i \alpha_j \langle \mathbf{x}_i \cdot \mathbf{x}_j \rangle, \qquad (55)$$

Dual optimization prolem

$$\text{Maximize: } L_D = \sum_{i=1}^{r} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{r} y_i y_j \alpha_i \alpha_j \langle \mathbf{x}_i \cdot \mathbf{x}_j \rangle. \tag{56}$$

$$\text{Subject to: } \sum_{i=1}^{r} y_i \alpha_i = 0$$

$$\alpha_i \geq 0, \quad i = 1, 2, \ldots, r.$$

- This dual formulation is called the **Wolfe dual**.
- For the convex objective function and linear constraints of the primal, it has the property that the maximum of L_D occurs at the same values of **w**, b and α_i, as the minimum of L_P (the primal).
- Solving (56) requires numerical techniques and clever strategies, which are beyond our scope.

The final decision boundary

- After solving (56), we obtain the values for α_i, which are used to compute the weight vector **w** and the bias *b* using Equations (48) and (52) respectively.
- **The decision boundary**

$$\langle \mathbf{w} \cdot \mathbf{x} \rangle + b = \sum_{i \in sv} y_i \alpha_i \langle \mathbf{x}_i \cdot \mathbf{x} \rangle + b = 0 \qquad (57)$$

- **Testing**: Use (57). Given a test instance **z**,

$$sign(\langle \mathbf{w} \cdot \mathbf{z} \rangle + b) = sign\left(\sum_{i \in sv} \alpha_i y_i \langle \mathbf{x}_i \cdot \mathbf{z} \rangle + b \right) \qquad (58)$$

- If (58) returns 1, then the test instance **z** is classified as positive; otherwise, it is classified as negative.

Linear SVM: Non-separable case

- Linear separable case is the ideal situation.
- Real-life data may have noise or errors.
  - Class label incorrect or randomness in the application domain.
- Recall in the separable case, the problem was

$$\text{Minimize}: \frac{\langle \mathbf{w} \cdot \mathbf{w} \rangle}{2}$$

$$\text{Subject to}: y_i(\langle \mathbf{w} \cdot \mathbf{x}_i \rangle + b) \geq 1, \quad i = 1, 2, \ldots, r$$

- With noisy data, the constraints may not be satisfied. Then, no solution!

Relax the constraints

- To allow errors in data, we relax the margin constraints by introducing **slack** variables, ξ_i (≥ 0) as follows:

$$\langle \mathbf{w} \cdot \mathbf{x}_i \rangle + b \geq 1 - \xi_i \quad \text{for } y_i = 1$$

$$\langle \mathbf{w} \cdot \mathbf{x}_i \rangle + b \leq -1 + \xi_i \text{ for } y_i = -1.$$

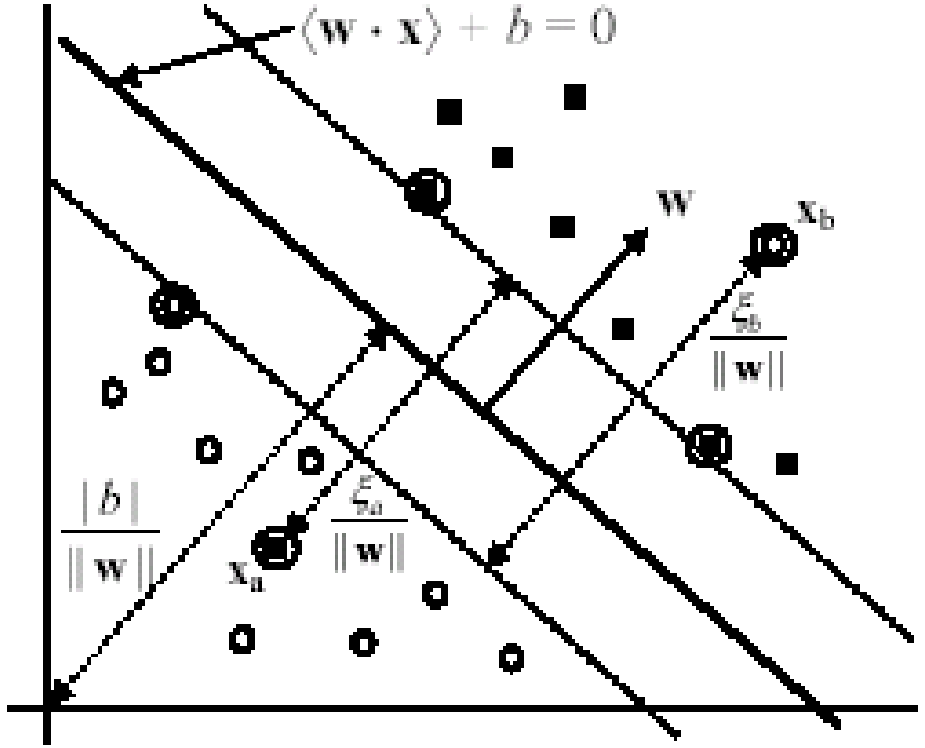
- The new constraints:

Subject to: $y_i(\langle \mathbf{w} \cdot \mathbf{x}_i \rangle + b) \geq 1 - \xi_i$, $i = 1, \ldots, r$,

$$\xi_i \geq 0, \quad i = 1, 2, \ldots, r.$$

Geometric interpretation

- Two error data points $\mathbf{x}_a$ and $\mathbf{x}_b$ (circled) in wrong regions



Penalize errors in objective function

- We need to penalize the errors in the objective function.
- A natural way of doing it is to assign an extra cost for errors to change the objective function to

$$\text{Minimize}: \frac{\langle \mathbf{w} \cdot \mathbf{w} \rangle}{2} + C\left(\sum_{i=1}^{r} \xi_i\right)^k \quad (60)$$

- $k = 1$ is commonly used, which has the advantage that neither ξ_i nor its Lagrangian multipliers appear in the dual formulation.

New optimization problem

- This formulation is called the **soft-margin SVM**. The primal Lagrangian is

(61)

$$\text{Minimize}: \quad \frac{\langle \mathbf{w} \cdot \mathbf{w} \rangle}{2} + C\sum_{i=1}^{r} \xi_i$$

$$\text{Subject to}: \quad y_i(\langle \mathbf{w} \cdot \mathbf{x}_i \rangle + b) \geq 1 - \xi_i, \quad i = 1, 2, \ldots, r$$

$$\xi_i \geq 0, \quad i = 1, 2, \ldots, r$$

where α_i, $\mu_i \geq 0$ are the **Lagrange multipliers**

(62)

$$L_P = \frac{1}{2}\langle \mathbf{w} \cdot \mathbf{w} \rangle + C\sum_{i=1}^{r} \xi_i - \sum_{i=1}^{r} \alpha_i [y_i(\langle \mathbf{w} \cdot \mathbf{x}_i \rangle + b) - 1 + \xi_i] - \sum_{i=1}^{r} \mu_i \xi_i$$

Kuhn-Tucker conditions

$$\frac{\partial L_P}{\partial w_j} = w_j - \sum_{i=1}^{r} y_i \alpha_i \mathbf{x}_i = 0, \quad j = 1, 2, \ldots, m \tag{63}$$

$$\frac{\partial L_P}{\partial b} = -\sum_{i=1}^{r} y_i \alpha_i = 0 \tag{64}$$

$$\frac{\partial L_P}{\partial \xi_i} = C - \alpha_i - \mu_i = 0, \quad i = 1, 2, \ldots, r \tag{65}$$

$$y_i(\langle \mathbf{w} \cdot \mathbf{x}_i \rangle + b) - 1 + \xi_i \geq 0, \quad i = 1, 2, \ldots, r \tag{66}$$

$$\xi_i \geq 0, \quad i = 1, 2, \ldots, r \tag{67}$$

$$\alpha_i \geq 0, \quad i = 1, 2, \ldots, r \tag{68}$$

$$\mu_i \geq 0, \quad i = 1, 2, \ldots, r \tag{69}$$

$$\alpha_i(y_i(\langle \mathbf{w} \cdot \mathbf{x}_i \rangle + b) - 1 + \xi_i) = 0, \quad i = 1, 2, \ldots, r \tag{70}$$

$$\mu_i \xi_i = 0, \quad i = 1, 2, \ldots, r \tag{71}$$

From primal to dual

- As the linear separable case, we transform the primal to a dual by setting to zero the partial derivatives of the Lagrangian (62) with respect to the primal variables (i.e., w, b and ξi), and substituting the resulting relations back into the Lagrangian.
- Ie.., we substitute Equations (63), (64) and (65) into the primal Lagrangian (62).
- From Equation (65), C – αi – μi = 0, we can deduce that αi $\leq$ C because μi $\geq$ 0.

Dual

- The dual of (61) is

$$\text{Maximize: } L_D(\boldsymbol{\alpha}) = \sum_{i=1}^{r} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{r} y_i y_j \alpha_i \alpha_j \langle \mathbf{x}_i \cdot \mathbf{x}_j \rangle. \qquad (72)$$

$$\text{Subject to: } \sum_{i=1}^{r} y_i \alpha_i = 0$$

$$0 \le \alpha_i \le C, \quad i = 1, 2, \ldots, r.$$

- Interestingly, ξ_i and its Lagrange multipliers μ_i are not in the dual. The objective function is identical to that for the separable case.
- The only difference is the constraint $\alpha_i \le C$.

Find primal variable values

- The dual problem (72) can be solved numerically.
- The resulting α_i values are then used to compute **w** and *b*. **w** is computed using Equation (63) and *b* is computed using the Kuhn-Tucker complementarity conditions (70) and (71).
- Since no values for ξ_i, we need to get around it.
  - From Equations (65), (70) and (71), we observe that if $0 < \alpha_i < C$ then both $\xi_i = 0$ and $y_i\langle \mathbf{w} \cdot \mathbf{x}_i \rangle + b - 1 + \xi_i = 0$. Thus, we can use any training data point for which $0 < \alpha_i < C$ and Equation (69) (with $\xi_i = 0$) to compute *b.*

$$b = \frac{1}{y_i} - \sum_{i=1}^{r} y_i \alpha_i \langle \mathbf{x}_i \cdot \mathbf{x}_j \rangle = 0. \qquad (73)$$

(65), (70) and (71) in fact tell us more

$$\begin{aligned} \alpha_i = 0 &\Rightarrow y_i(\langle \mathbf{w} \cdot \mathbf{x}_i \rangle + b) \geq 1 \text{ and } \xi_i = 0 \\ 0 < \alpha_i < C &\Rightarrow y_i(\langle \mathbf{w} \cdot \mathbf{x}_i \rangle + b) = 1 \text{ and } \xi_i = 0 \\ \alpha_i = C &\Rightarrow y_i(\langle \mathbf{w} \cdot \mathbf{x}_i \rangle + b) \leq 1 \text{ and } \xi_i \geq 0 \end{aligned} \tag{74}$$

- (74) shows a very important property of SVM.
  - The solution is **sparse** in α_i. Many training data points are outside the margin area and their α_i's in the solution are 0.
  - Only those data points that are on the margin (i.e., $y_i(\langle \mathbf{w} \cdot \mathbf{x}_i \rangle + b) = 1$, which are support vectors in the separable case), inside the margin (i.e., $\alpha_i = C$ and $y_i(\langle \mathbf{w} \cdot \mathbf{x}_i \rangle + b) < 1$), or errors are non-zero.
  - Without this sparsity property, SVM would not be practical for large data sets.

The final decision boundary

- The final decision boundary is (we note that many α_i's are 0)

$$\langle \mathbf{w} \cdot \mathbf{x} \rangle + b = \sum_{i=1}^{r} y_i \alpha_i \langle \mathbf{x}_i \cdot \mathbf{x} \rangle + b = 0 \qquad (75)$$

- The decision rule for classification (testing) is the same as the separable case, i.e.,

  sign($\langle \mathbf{w} \cdot \mathbf{x} \rangle + b$).

- Finally, we also need to determine the parameter *C* in the objective function. It is normally chosen through the use of a validation set or cross-validation.

How to deal with nonlinear separation?

- The SVM formulations require linear separation.
- Real-life data sets may need nonlinear separation.
- To deal with nonlinear separation, the same formulation and techniques as for the linear case are still used.
- We only transform the input data into another space (usually of a much higher dimension) so that
  - a linear decision boundary can separate positive and negative examples in the transformed space,
- The transformed space is called the **feature space**. The original data space is called the **input space**.

Space transformation

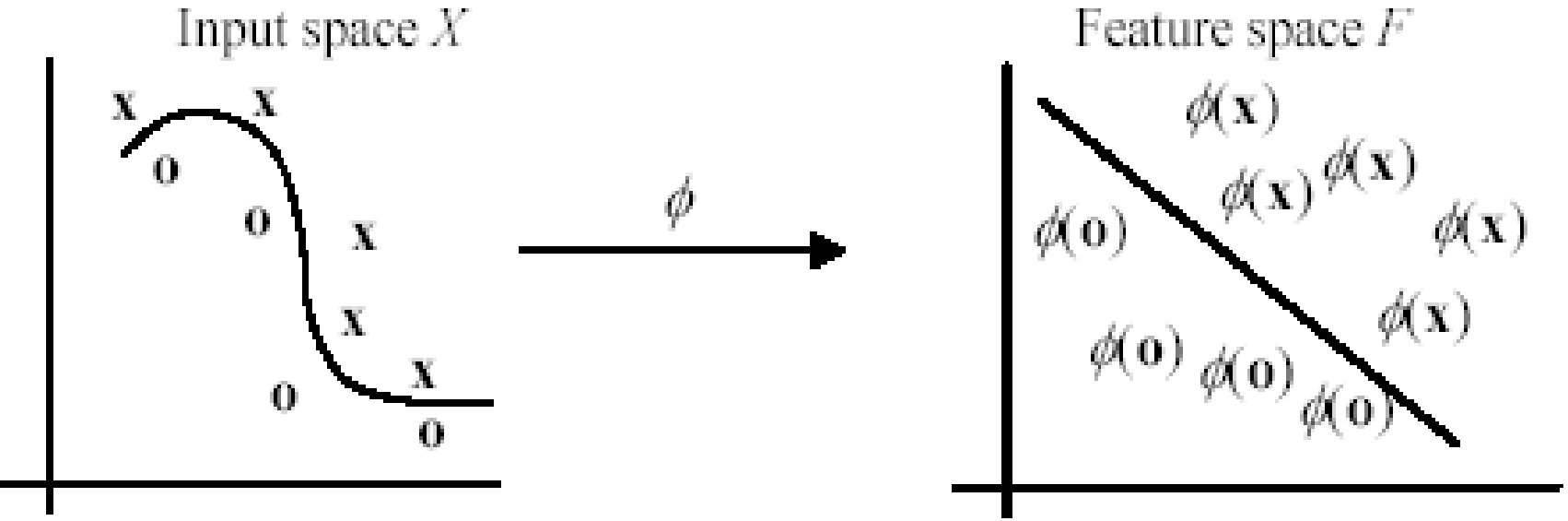
- The basic idea is to map the data in the input space X to a feature space F via a nonlinear mapping ϕ,

$$\begin{aligned} \phi &: X \rightarrow F \\ \mathbf{x} &\mapsto \phi(\mathbf{x}) \end{aligned} \qquad (76)$$

- After the mapping, the original training data set $\{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_r, y_r)\}$ becomes:

$$\{(\phi(\mathbf{x}_1), y_1), (\phi(\mathbf{x}_2), y_2), \ldots, (\phi(\mathbf{x}_r), y_r)\} \qquad (77)$$

Geometric interpretation



- In this example, the transformed space is also 2-D. But usually, the number of dimensions in the feature space is much higher than that in the input space

Optimization problem in (61) becomes

With the transformation, the optimization problem in (61) becomes

$$\text{Minimize}: \frac{\langle \mathbf{w} \cdot \mathbf{w} \rangle}{2} + C\sum_{i=1}^{r} \xi_i \tag{78}$$

$$\text{Subject to}: y_i(\langle \mathbf{w} \cdot \phi(\mathbf{x}_i) \rangle + b) \geq 1 - \xi_i, \quad i = 1, 2, ..., r$$

$$\xi_i \geq 0, \quad i = 1, 2, ..., r$$

The dual is

$$\text{Maximize: } L_D = \sum_{i=1}^{r} \alpha_i - \frac{1}{2}\sum_{i,j=1}^{r} y_i y_j \alpha_i \alpha_j \langle \phi(\mathbf{x}_i) \cdot \phi(\mathbf{x}_j) \rangle. \tag{79}$$

$$\text{Subject to: } \sum_{i=1}^{r} y_i \alpha_i = 0$$

$$0 \leq \alpha_i \leq C, \quad i = 1, 2, ..., r.$$

The final decision rule for classification (testing) is

$$\sum_{i=1}^{r} y_i \alpha_i \langle \phi(\mathbf{x}_i) \cdot \phi(\mathbf{x}) \rangle + b \tag{80}$$

An example space transformation

- Suppose our input space is 2-dimensional, and we choose the following transformation (mapping) from 2-D to 3-D:

$$(x_1, x_2) \mapsto ({x_1}^2, {x_2}^2, \sqrt{2}x_1x_2)$$

- The training example ((2, 3), -1) in the input space is transformed to the following in the feature space:

((4, 9, 8.5), -1)

Problem with explicit transformation

- The potential problem with this explicit data transformation and then applying the linear SVM is that it may suffer from the curse of dimensionality.
- The number of dimensions in the feature space can be huge with some useful transformations even with reasonable numbers of attributes in the input space.
- This makes it computationally infeasible to handle.
- Fortunately, explicit transformation is not needed.

Kernel functions

- We notice that in the dual formulation both
  - the construction of the optimal hyperplane (79) in F and
  - the evaluation of the corresponding decision function (80)

  only require dot products $\langle \phi(\mathbf{x}) \cdot \phi(\mathbf{z}) \rangle$ and never the mapped vector $\phi(\mathbf{x})$ in its explicit form. This is a crucial point.
- Thus, if we have a way to compute the dot product $\langle \phi(\mathbf{x}) \cdot \phi(\mathbf{z}) \rangle$ using the input vectors **x** and **z** directly,
  - no need to know the feature vector $\phi(\mathbf{x})$ or even ϕ itself.
- In SVM, this is done through the use of **kernel functions**, denoted by K,

$$K(\mathbf{x}, \mathbf{z}) = \langle \phi(\mathbf{x}) \cdot \phi(\mathbf{z}) \rangle \tag{82}$$

An example kernel function

- Polynomial kernel

$$K(\mathbf{x}, \mathbf{z}) = \langle \mathbf{x} \cdot \mathbf{z} \rangle^d \tag{83}$$

- Let us compute the kernel with degree $d = 2$ in a 2-dimensional space: $\mathbf{x} = (x_1, x_2)$ and $\mathbf{z} = (z_1, z_2)$.

$$\begin{aligned} \langle \mathbf{x} \cdot \mathbf{z} \rangle^2 &= (x_1 z_1 + x_2 z_2)^2 \\ &= x_1^2 z_1^2 + 2 x_1 z_1 x_2 z_2 + x_2^2 z_2^2 \\ &= \langle (x_1^2, x_2^2, \sqrt{2} x_1 x_2) \cdot (z_1^2, z_2^2, \sqrt{2} z_1 z_2) \rangle \\ &= \langle \phi(\mathbf{x}) \cdot \phi(\mathbf{z}) \rangle, \end{aligned} \tag{84}$$

- This shows that the kernel $\langle \mathbf{x} \cdot \mathbf{z} \rangle^2$ is a dot product in a transformed feature space

Kernel trick

- The derivation in (84) is only for illustration purposes.
- We do not need to find the mapping function.
- We can simply apply the kernel function directly by
  - replace all the dot products $\langle \phi(\mathbf{x}) \cdot \phi(\mathbf{z}) \rangle$ in (79) and (80) with the kernel function $K(\mathbf{x}, \mathbf{z})$ (e.g., the polynomial kernel $\langle \mathbf{x} \cdot \mathbf{z} \rangle^d$ in (83)).
- This strategy is called the **kernel trick**.

Is it a kernel function?

- The question is: how do we know whether a function is a kernel without performing the derivation such as that in (84)? I.e,
  - How do we know that a kernel function is indeed a dot product in some feature space?
- This question is answered by a theorem called the Mercer's theorem, which we will not discuss here.

Commonly used kernels

- It is clear that the idea of kernel generalizes the dot product in the input space. This dot product is also a kernel with the feature map being the identity

$$K(\mathbf{x}, \mathbf{z}) = \langle \mathbf{x} \cdot \mathbf{z} \rangle. \tag{85}$$

Commonly used kernels include

$$\text{Polynomial:} \quad K(\mathbf{x}, \mathbf{z}) = (\langle \mathbf{x} \cdot \mathbf{z} \rangle + \theta)^d \tag{86}$$

$$\text{Gaussian RBF:} \quad K(\mathbf{x}, \mathbf{z}) = e^{-\|\mathbf{x}-\mathbf{z}\|^2/2\sigma} \tag{87}$$

$$\text{Sigmoidal:} \quad K(\mathbf{x}, \mathbf{z}) = \tanh(k\langle \mathbf{x} \cdot \mathbf{z} \rangle - \delta) \tag{88}$$

where $\theta \in R$, $d \in N$, $\sigma > 0$, and $k, \delta \in R$.

Some other issues in SVM

- SVM works only in a real-valued space. For a categorical attribute, we need to convert its categorical values to numeric values.
- SVM does only two-class classification. For multi-class problems, some strategies can be applied, e.g., one-against-rest, and error-correcting output coding.
- The hyperplane produced by SVM is hard to understand by human users. The matter is made worse by kernels. Thus, SVM is commonly used in applications that do not required human understanding.