Data Mining and Data Warehousing

2. Machine Learning I: Supervised Learning

Drd. Horia Modran

Contact: horia.modran@unitbv.ro / modranhoria@gmail.com

Tel: 0770171577

Universitatea Transilvania din Brașov

FACULTATEA DE INGINERIE ELECTRICĂ ȘI ȘTIINȚA CALCULATOARELOR

2022 - 2023

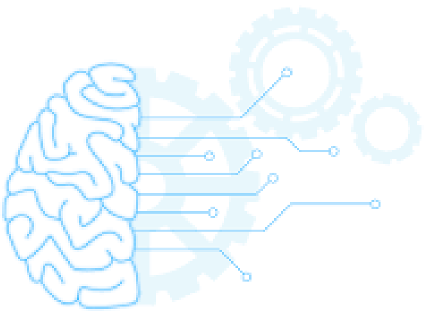
What is learning ?

- Herbert Simon: “Learning is any process by which a system improves performance from experience.”
- “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

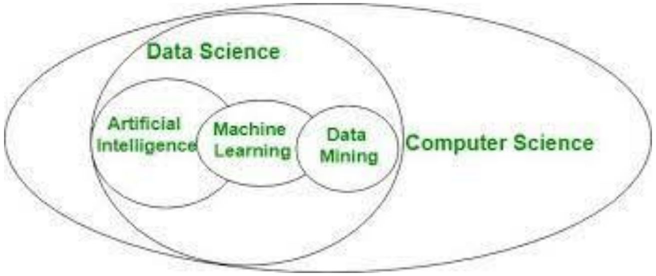
Machine Learning

- Machine learning: study of computer algorithms that can improve automatically through experience and by use of data
  - 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)



Machine Learning and Data Mining

- Machine learning and data mining often employ the same methods and overlap significantly
- Machine learning focuses on prediction, based on known properties learned from the training data
- Data mining focuses on the discovery of (previously) unknown properties in the data (analysis step of knowledge discovery in databases)
- Data mining uses many machine learning methods, but with different goals



Machine Learning Areas

- Supervised Learning: data and corresponding labels are given
- Unsupervised Learning: only data is given, no labels provided
- 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

- Supervised learning (SL) is the machine learning task of learning a function that maps an input to an output based on example input-output pairs
- it infers a function from labeled training data
- each example consists of an input object and a desired output
- an algorithm analyzes the training data and produces an inferred function, used for mapping new examples
- optimal scenario: correctly determine the class labels for unseen instances
- statistical quality - measured through generalization error

Important Concepts

- Data: labeled instances <x, y>, e. g. emails marked spam/not spam -> split into training/(hold-out)/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
- Evaluation
  - accuracy: fraction of instances predicted correctly
- Overfitting and generalization
  - want a classifier which does well on test data

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.

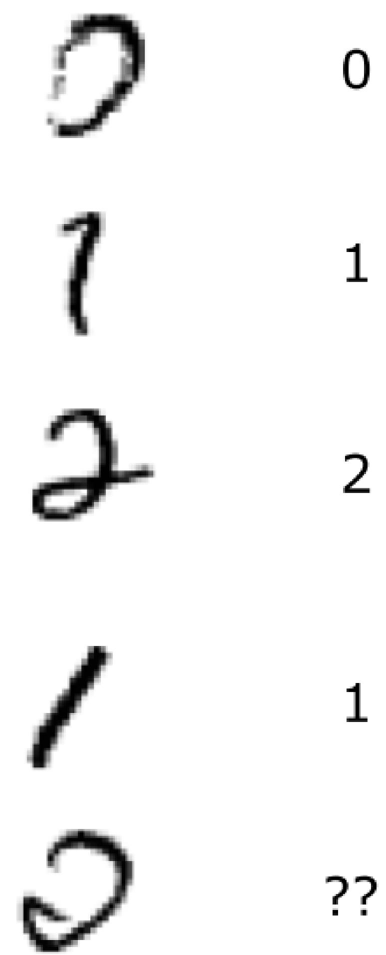
Example: Digit Recognition

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
- ...

Classification Examples

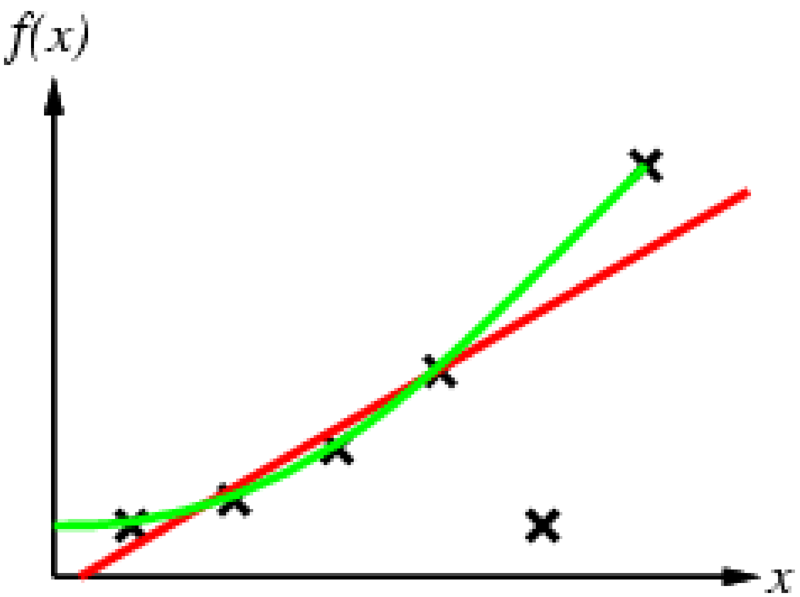
- In classification, we predict labels y (classes) for inputs x
- examples:
  - OCR (input: images, classes: characters)
  - Medical diagnosis (input: symptoms, classes: diseases)
  - Automatic essay grader (input: document, classes: grades)
  - Fraud detection (input: account activity, classes: fraud / no fraud)
  - Customer service email routing
  - recommended articles in a newspaper, books
  - DNA and protein sequence identification
  - financial investments

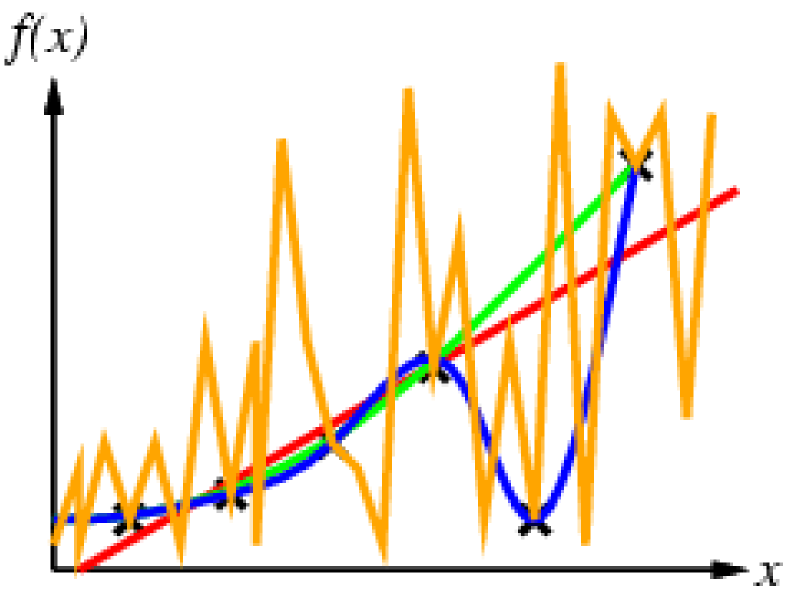
Inductive learning

- Simplest form: learn a function from examples
  - 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 ≈ f, given a training set of examples
- this is a highly simplified model of real learning:
  - ignores prior knowledge
  - assumes examples are given

Inductive learning

- construct/adjust h to agree with f on training set
  - h is consistent if it agrees with f on all examples
  - e. g. curve fitting

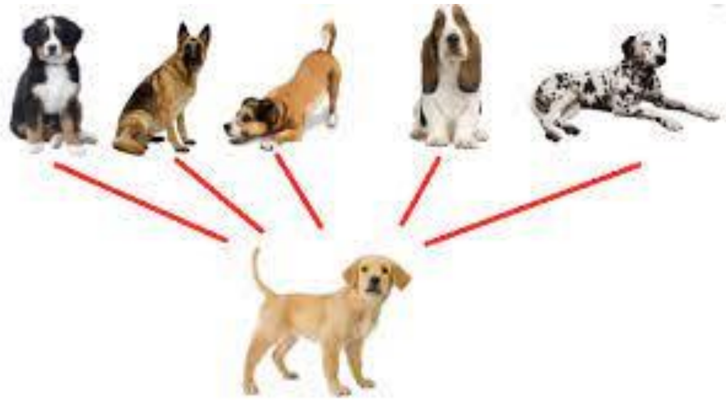




Ockham's razor: prefer the simplest hypothesis consistent with data

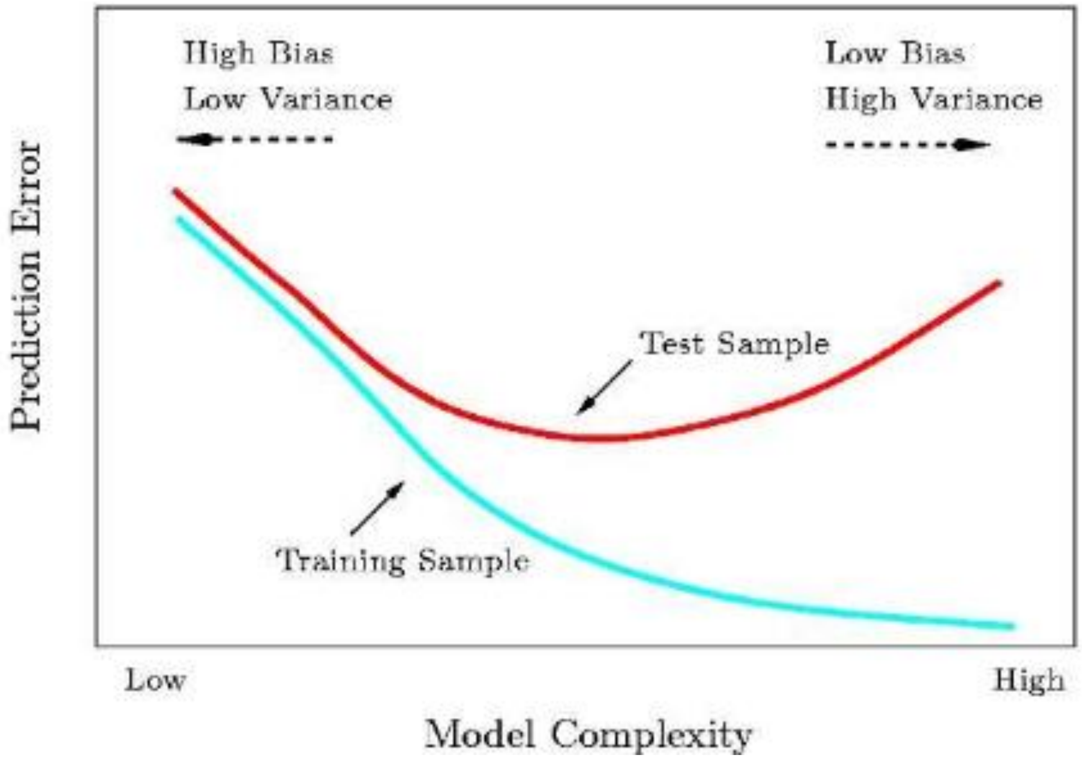
Generalization

- 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



Training error vs test error

- Low bias/high variance – **overfitting** ("fitting" the training set)
- High bias/low variance – **underfitting** (the model cannot capture the structure of the data)



Classification/Regression

- Learning a discrete function: classification
  - boolean classification:
    - each example is classified as true(positive) or false(negative)
  - can also predict multiple classes (3, 4, 5…)
- Learning a continuous function: Regression
  - E.g. Linear Regression, Logistic regression

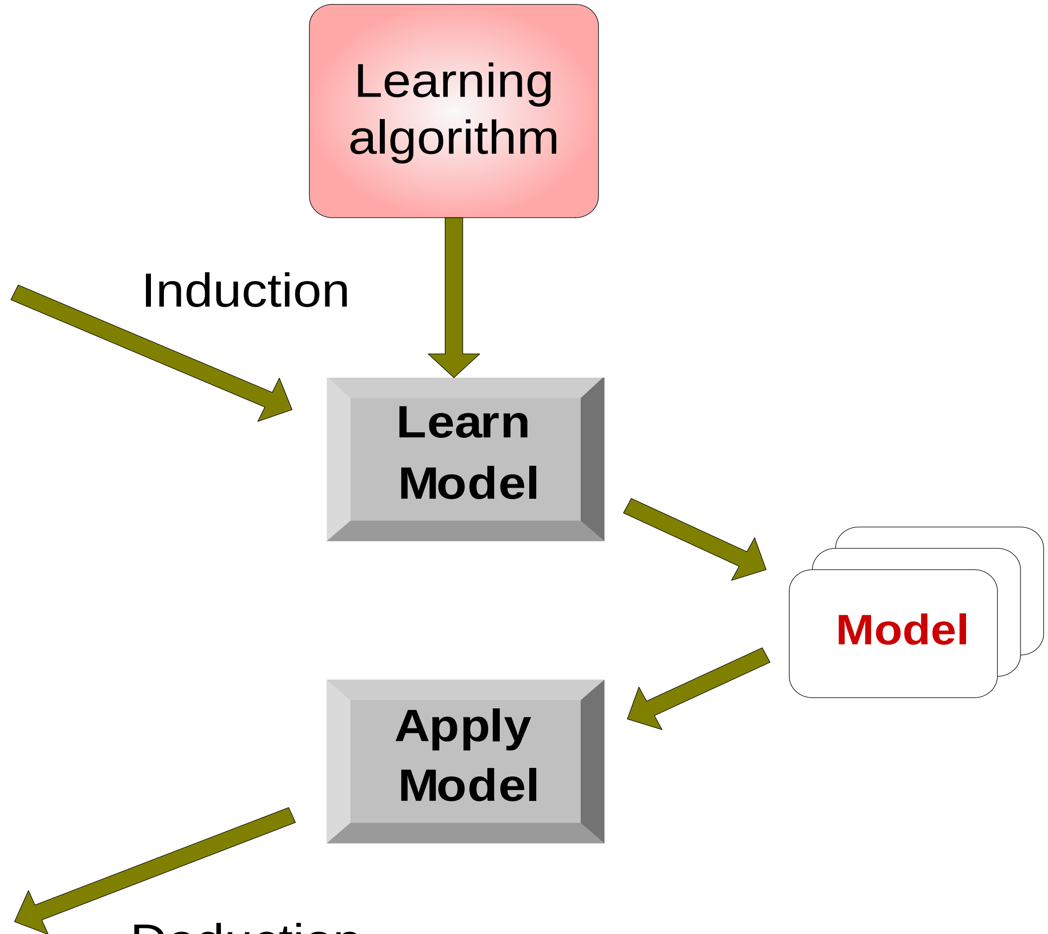
Classification

- Model construction: describing a set of predetermined classes
  - 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
- Model usage: for classifying future or unknown objects
  - estimate accuracy of the model
  - if the accuracy is acceptable, use the model to classify data tuples whose class labels are not known

Classification example

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

Data preparation

- Data cleaning
  - preprocess data in order to reduce noise and handle missing values
- Relevance analysis (feature selection)
  - remove the irrelevant or redundant attributes
- Data transformation
  - generalize data to (higher concepts, discretization)
  - normalize attribute values

Classification techniques

- Decision Tree based Methods
  - Random Forests
- Rule-based Methods
- Naive Bayes
- Bayesian Belief Networks
- Support Vector Machines
- and many more…

Decision trees

- Example Problem: decide whether to wait for a table at a restaurant, based on the following attributes:
  - alternate: is there an alternative restaurant nearby?
  - Bar: is there a comfortable bar area to wait in?
  - Fri/Sat: is today Friday or Saturday?
  - hungry: are we hungry?
  - patrons: number of people in the restaurant (None, Some, Full)
  - price: price range ($, $$, $$$)
  - raining: is it raining outside?
  - reservation: have we made a reservation?
  - type: kind of restaurant (French, Italian, Thai, Burger)
  - waitEstimate: estimated waiting time (0-10, 10-30, 30-60, >60)

Feature base representation

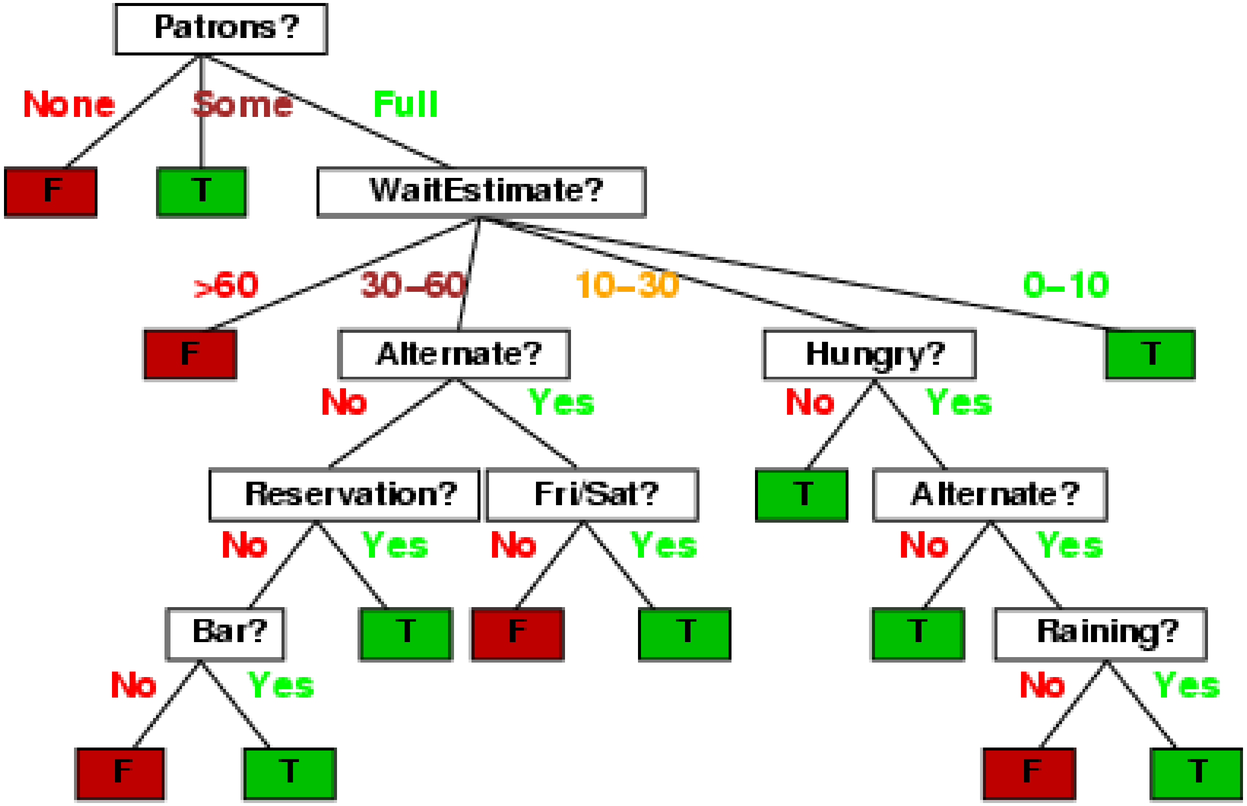
- examples described by feature(attribute) values (Boolean, discrete, continuous)
  - e.g., situations where I will/won't wait for a table:

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

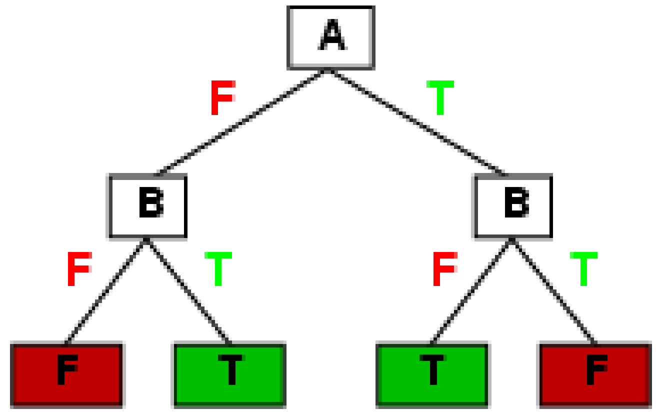
- one possible representation for hypotheses
  - e.g., here is the “true” tree for deciding whether to wait:



Expressivness

- 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

Decision tree algorithm

- Principle – 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
  - 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
  - there are no samples left

Advantages/disadvantages

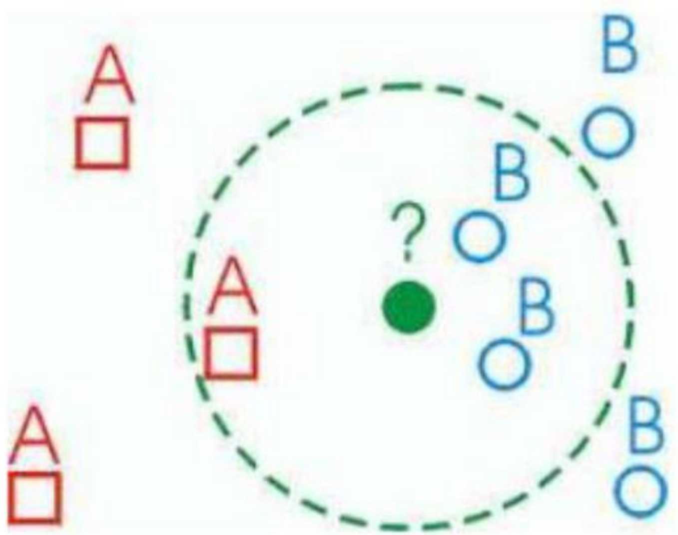
- Advantages
  - easy to construct/implement
  - extremely fast at classifying unknown records
  - accuracy is comparable to other classification techniques for many simple data sets
- Disadvantages
  - computationally expensive to train
  - some decision trees can be overly complex that do not generalize the data well
  - less expressivity

K-Nearest Neighbor (KNN)

- simple, but a very powerful classification algorithm
- classifies based on a similarity measure
- non-parametric
- lazy learning
  - does not “learn” until the test example is given
  - whenever we have a new data to classify, we find its K nearest neighbors from the training data

KNN: Classification

- classified by “MAJORITY VOTES” for its neighbor classes
  - assigned to the most common class amongst its K nearest neighbors (by measuring “distant” between data)



KNN: Steps

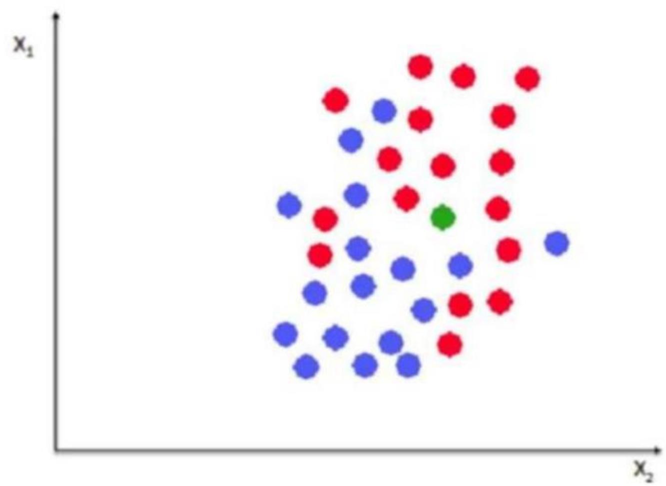
Step 1: Determine parameter K = number of nearest neighbors

Step 2: Calculate the distance between the query-instance and all the training examples.

Step 3: Sort the distance and determine nearest neighbors based on the k-th minimum distance.

Step 4:Gather the category Y of the nearest neighbors.

Step 5: Use simple majority of the category of nearest neighbors as the prediction value of the query instance.



KNN: Example

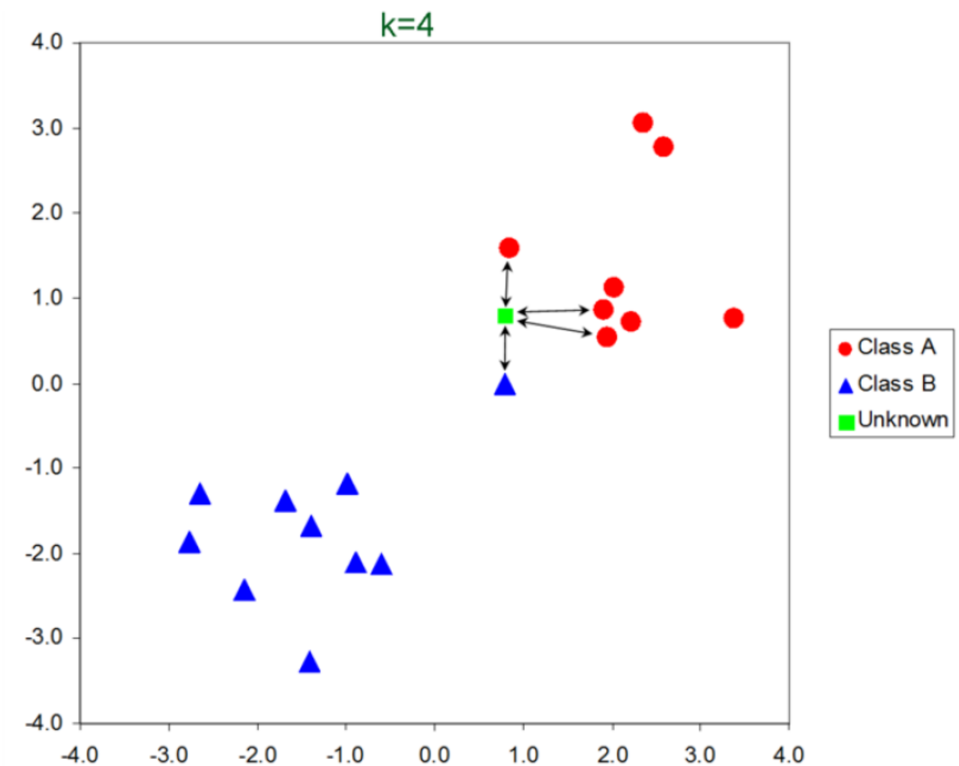


Table 1. Euclidean distance matrix D listing all possible pairwise Euclidean distances between 19 samples.

	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}	x_{11}	x_{12}	x_{13}	x_{14}	x_{15}	x_{16}	x_{17}	x_{18}
x_2	1.5																	
x_3	1.4	1.6																
x_4	1.6	1.4	1.3															
x_5	1.7	1.4	1.5	1.5														
x_6	1.3	1.4	1.4	1.5	1.4													
x_7	1.6	1.3	1.4	1.4	1.5	1.8												
x_8	1.5	1.4	1.6	1.3	1.7	1.6	1.4											
x_9	1.4	1.3	1.4	1.5	1.2	1.4	1.3	1.5										
x_{10}	2.3	2.4	2.5	2.3	2.6	2.7	2.8	2.7	3.1									
x_{11}	2.9	2.8	2.9	3.0	2.9	3.1	2.9	3.1	3.0	1.5								
x_{12}	3.2	3.3	3.2	3.1	3.3	3.4	3.3	3.4	3.5	3.3	1.6							
x_{13}	3.3	3.4	3.2	3.2	3.3	3.4	3.2	3.3	3.5	3.6	1.4	1.7						
x_{14}	3.4	3.2	3.5	3.4	3.7	3.5	3.6	3.3	3.5	3.6	1.5	1.8	0.5					
x_{15}	4.2	4.1	4.1	4.1	4.1	4.1	4.1	4.1	4.1	4.1	1.7	1.6	0.3	0.5				
x_{16}	4.1	4.1	4.1	4.1	4.1	4.1	4.1	4.1	4.1	4.1	1.6	1.5	0.4	0.5	0.4			
x_{17}	5.9	6.2	6.2	5.8	6.1	6.0	6.1	5.9	5.8	6.0	2.3	2.3	2.5	2.3	2.4	2.5		
x_{18}	6.1	6.3	6.2	5.8	6.1	6.0	6.1	5.9	5.8	6.0	3.1	2.7	2.6	2.3	2.5	2.6	3.0	
x_{19}	6.0	6.1	6.2	5.8	6.1	6.0	6.1	5.9	5.8	6.0	3.0	2.9	2.7	2.4	2.5	2.8	3.1	0.4

$$d(x,\ y) = \sqrt{\sum_{i=1}^{n}(y_i - x_i)^2}$$

n– number of dimensions (in our case – 2)

Pros and cons

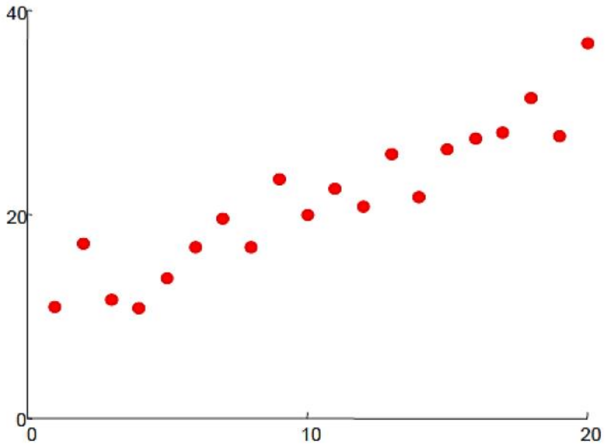
- Pros
  - learning and implementation is extremely simple and intuitive
  - flexible decision boundaries
- Cons
  - irrelevant or correlated features have high impact and must be eliminated
  - difficult to handle high dimensionality
  - computational costs: memory and classification time computation

Correlation

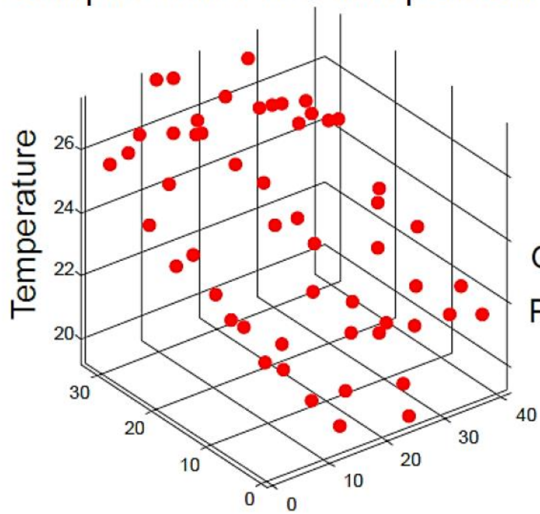
- linear association between two variables
- show how to determine both the nature and strength of relationship between two variables
- correlation lies between -1 to +1
- zero correlation indicates that there is no relationship between the variables
- Pearson correlation coefficient
  - most familiar measure of dependence between two quantities

Linear Regression

Samples with ONE independent variable

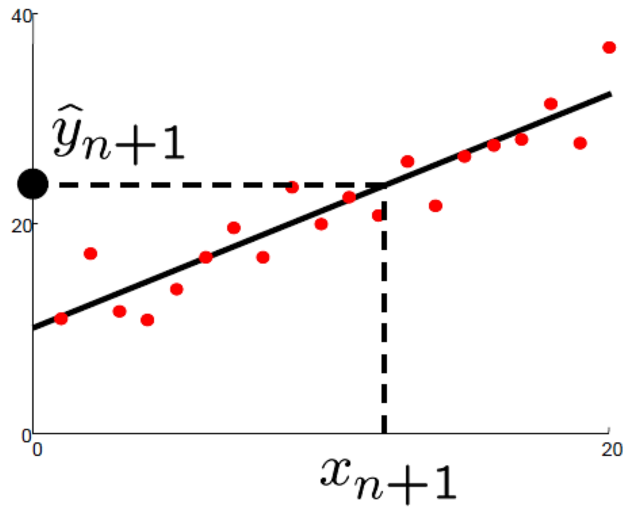


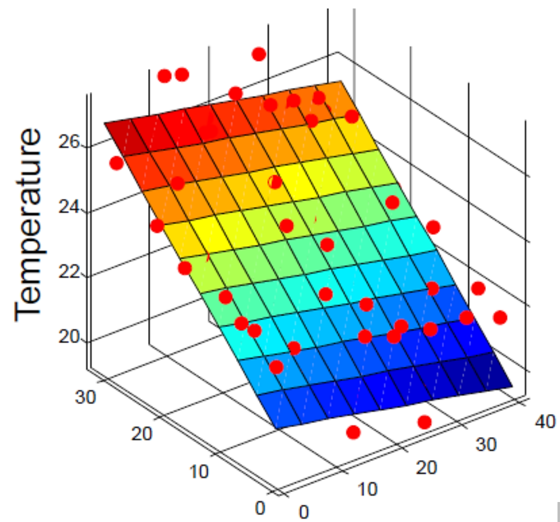
Samples with TWO independent variables



Given examples $(x_i, y_i)_{i=1\ldots n}$

Predict y_{n+1} given a new point x_{n+1}





Universitatea Transilvania din Brașov
FACULTATEA DE INGINERIE ELECTRICĂ ȘI ȘTIINȚA CALCULATOARELOR

Linear Regression

- how to represent the data as a vector/matrix
- we assume a model

$$\mathbf{y} = \mathrm{b}_0 + \mathbf{bX} + \epsilon$$

, where b_0 and $\mathbf{b}$ are intercept and slope, known as coefficients or parameters. ϵ is the error term (typically assumes that $\epsilon \sim N(\mu,\sigma^2)$)

- Simple Linear Regression
  - a single independent variable is used to predict
- Multiple linear regression
  - two or more independent variables are used to predict

Linear Regression

- find the optimal coefficient vector b that makes the most similar observation: **y=Xb+e** (vector multiplication)

$$\begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix} = \begin{bmatrix} 1 & x_{11} & \cdots & x_{1p} \\ 1 & \vdots & \ddots & \vdots \\ 1 & x_{n1} & \cdots & x_{np} \end{bmatrix} \begin{bmatrix} b_0 \\ \vdots \\ b_p \end{bmatrix} + \begin{bmatrix} e_1 \\ \vdots \\ e_n \end{bmatrix}$$

- need to minimize the error (sum squared error)

$$\min J(\mathbf{b}) = \sum_{i=1}^{n} (y_i - \mathbf{x}_{i,*}\mathbf{b})^2$$

Linear regression with categorical variables

- we assumed that all variables are continuous variables
- categorical variables:
  - ordinal variables - encode data with continuous values
    - Evaluation: Excellent (5), Very good (4), Good (3), Poor (2), Very poor (1)
- nominal variables - use dummy variables
  - department: Computer, Biology, Physics

	Computer	Biology	Physics
Computer	1	0	0
Biology	0	1	0
Physics	0	0	1

Linear regression for classification

- for binary classification
  - encode class labels as y=0,1 or {−1,1}
  - apply formula: $\mathbf{y} = \mathbf{X} * \mathbf{b} + \mathbf{e}$
  - check which class the prediction is closer to
    - if class 1 is encoded to 1 and class 2 is – 1

$$\begin{array}{ll} class\ 1 & if\ f(x) \geq 0 \\ class\ 2 & if\ f(x) < 0 \end{array}$$

    - **linear models are NOT optimized for classification**

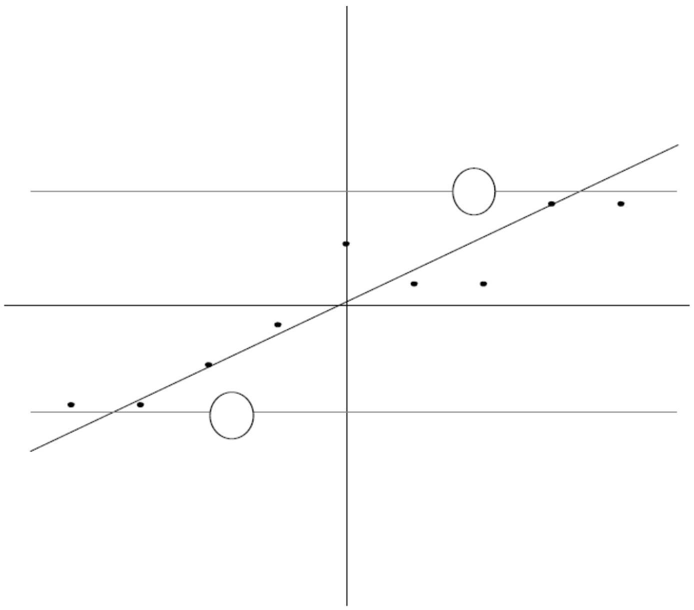
Logistic regression

Logistic regression

- Predict results on a binary outcome variable
  - e.g., whether or not a patient has a disease
  - whether a new applicant would succeed in the program or not
  - the outcome is not continuous or distributed normally
- when we have a binary response variables
  - we code "disease" as 1 and "no disease" as 0, can we just fit a line through those points as we would with linear regression? Possible! But some problems.

Linear Regression Problem

- the problem of fitting a regular regression line to a binary dependent variable
  - the line seems to oversimplify the relationship
  - it gives predictions that cannot be observable values of Y for extreme values of X
  - the approach is analogous to fitting a linear model to the probability of the event



  - produces unobservable predictions for extreme values of dependent variable

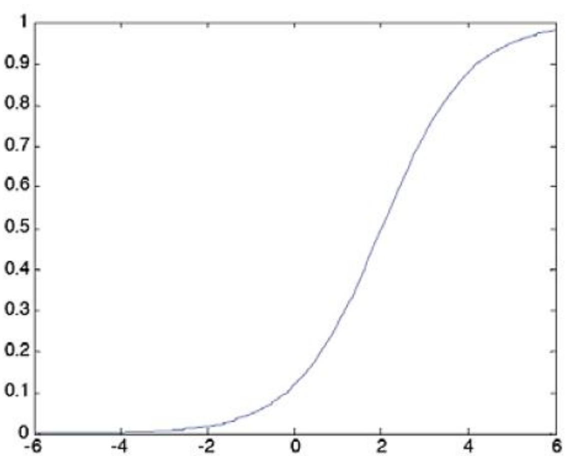
Probabilistic approach

- Learn P(Y|X) directly
  - cumulative probability distribution
  - using a sigmoid function $P(Y=0|X) = \frac{1}{1+\exp(\mathbf{Xb})}$

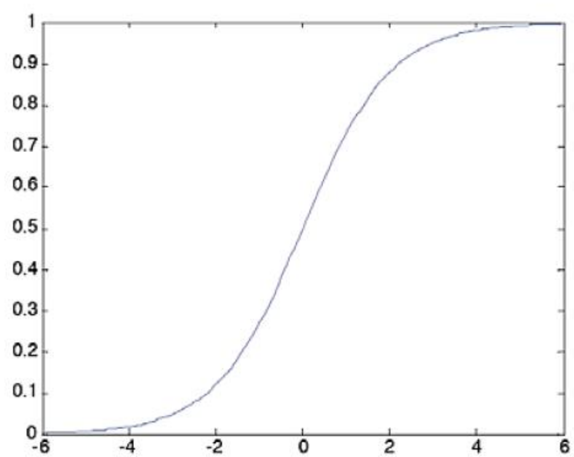
$$P(Y=1|X) = \frac{1}{1+\exp(-\mathbf{Xb})}$$

Sigmoid: $g(w_0 + \sum_i w_i x_i) = \frac{1}{1+e^{w_0+\sum_i w_i x_i}}$

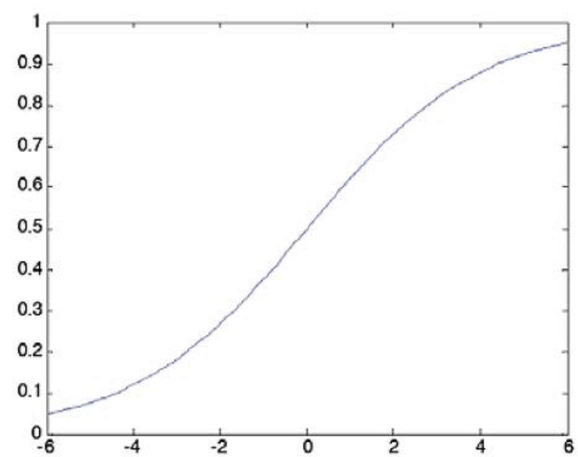
w_0=-2, w_1=-1



w_0=0, w_1=-1



w_0=0, w_1=-0.5



Logistic regression model

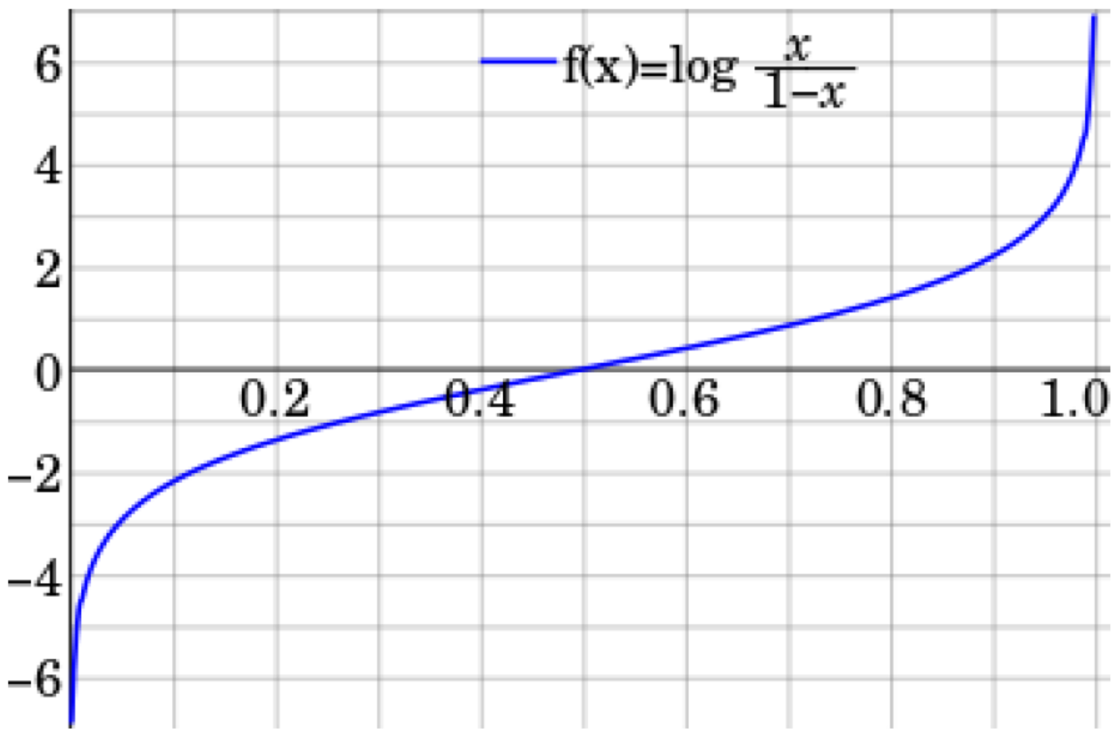
$$\log\left(\frac{p}{1-p}\right) = \mathbf{Xb}$$

- p is the probability that an event Y occurs, p(Y=1)
- p/(1 p) is the "Odds ratio" -> range of [0 to infinite]
- Log(p/(1 p)) is log odds ratio, or "logit" -> range: $[-\infty, +\infty]$

Odd	Logit score
0.99	1.996
0.5	0
0.25	-0.477
0.01	-1.996

Logistic function

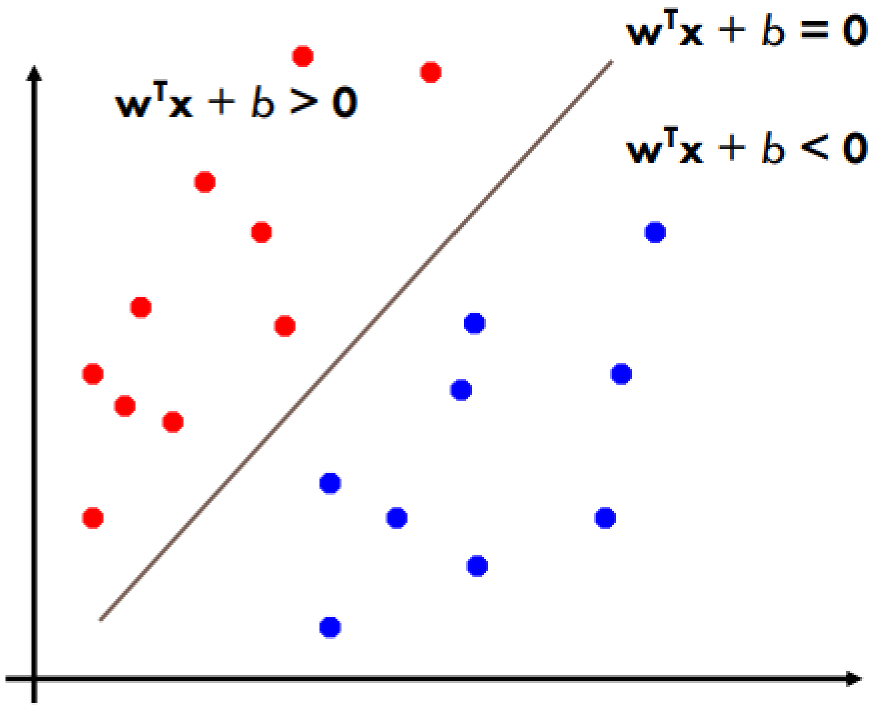
$$Y = \log(p/(1\text{-}p))$$



- Logistic regression predicts probabilities rather than classes: **stochastic approach**

Linear Separators

- Binary classification can be viewed as the task of separating classes in feature space:



$$f(\mathbf{x}) = \text{sign}(\mathbf{w}^T\mathbf{x} + b)$$

Training set:

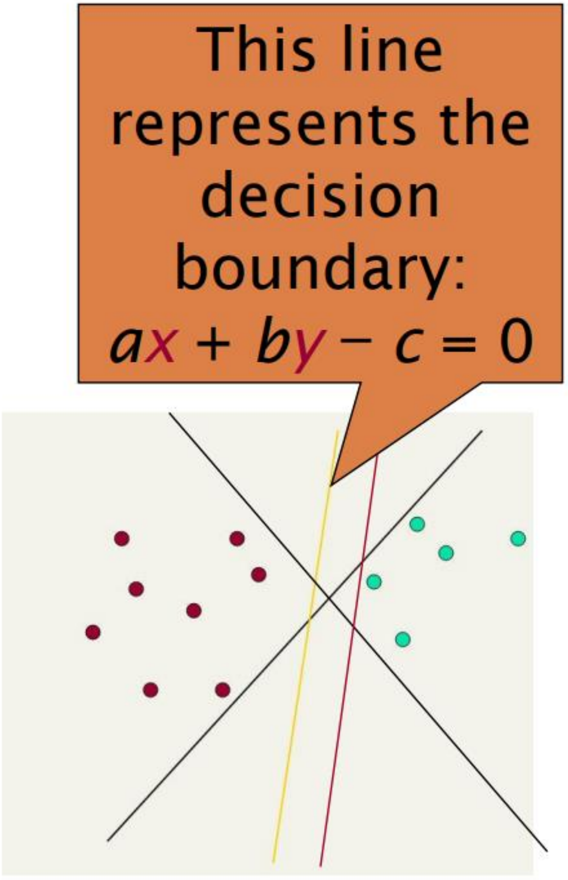
$$(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n),$$

y (either 1 or –1) - indicating the class to which the point x_i belongs

w - normal vector to the hyperplane

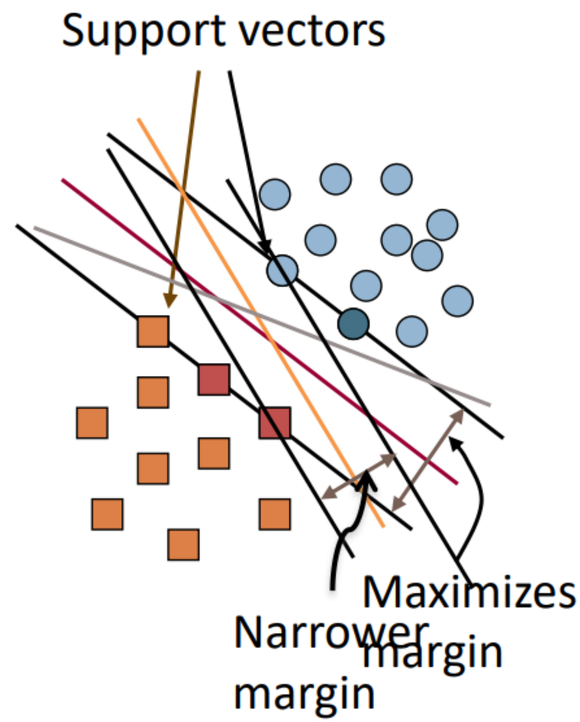
Hyperplane

- lots of possible choices for a, b, c
- a Support Vector Machine (SVM) finds an optimal solution
  - maximizes the distance between the hyperplane and the "difficult points" close to decision boundary
  - one intuition: if there are no points near the decision surface, then there are no very uncertain classification decisions



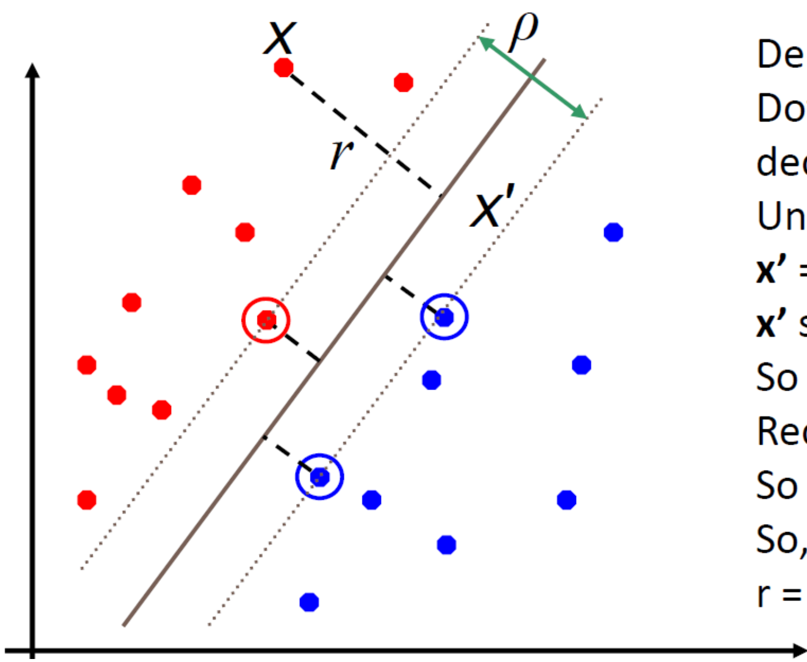
SVM

- SVMs maximize the margin around the separating hyperplane
  - a.k.a. large margin classifiers
- the decision function is fully specified by a subset of training samples, the support vectors
- solving SVMs is a quadratic programming problem



Geometric Margin

- distance from example to the separator is $r = y\dfrac{\mathbf{w}^T\mathbf{x}+b}{\|\mathbf{w}\|}$
- examples closest to the hyperplane are **support vectors**
- **margin** ρ of the separator is the width of separation between support vectors of classes



Derivation of finding *r:*
Dotted line **x’** – **x** is perpendicular to decision boundary so parallel to **w.**
Unit vector is **w**/|**w**|, so line is r**w**/|**w**|
x’ = **x** – yr**w**/|**w**|.
x’ satisfies $\mathbf{w}^T\mathbf{x}' + b = 0$.
So $\mathbf{w}^T(\mathbf{x} - yr\mathbf{w}/|\mathbf{w}|) + b = 0$
Recall that $|\mathbf{w}| = \mathrm{sqrt}(\mathbf{w}^T\mathbf{w})$.
So $\mathbf{w}^T\mathbf{x} - yr|\mathbf{w}| + b = 0$
So, solving for r gives:
$r = y(\mathbf{w}^T\mathbf{x} + b)/|\mathbf{w}|$

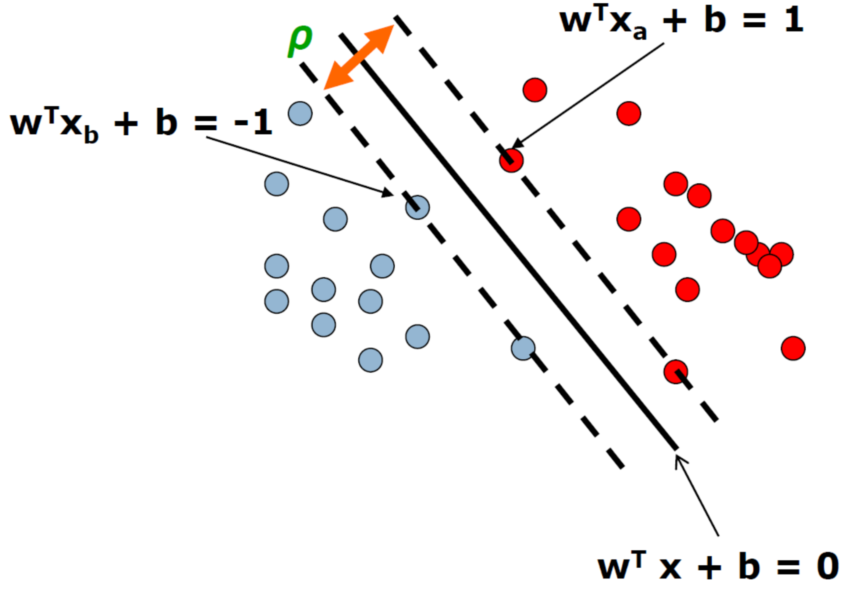
Linear SVM

- assume that the functional margin of each data item is at least 1, then the following two constraints follow for a training set $\{(x_i , y_i)\}$:

$$\mathbf{w}^{\mathbf{T}}\mathbf{x}_{\mathbf{i}} + b \geq 1 \quad \text{if } y_i = 1$$

$$\mathbf{w}^{\mathbf{T}}\mathbf{x}_{\mathbf{i}} + b \leq -1 \quad \text{if } y_i = -1$$

- for support vectors, the inequality becomes an equality



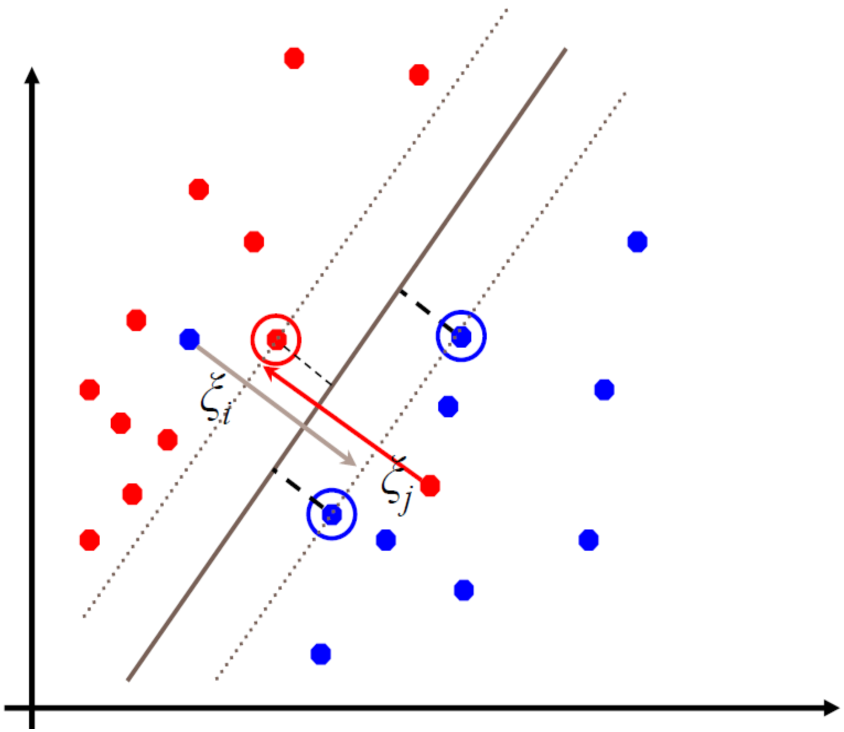
Hyperplane
$\mathbf{w}^T\mathbf{x} + b = 0$

Extra scale constraint:
$\min_{i=1,\dots,n} |\mathbf{w}^T\mathbf{x}_i + \mathbf{b}| = 1$

This implies:
$\mathbf{w}^T(\mathbf{x}_a - \mathbf{x}_b) = 2$
$\rho = \|\mathbf{x}_a - \mathbf{x}_b\|_2 = 2/\|\mathbf{w}\|_2$

Soft Margin Classification

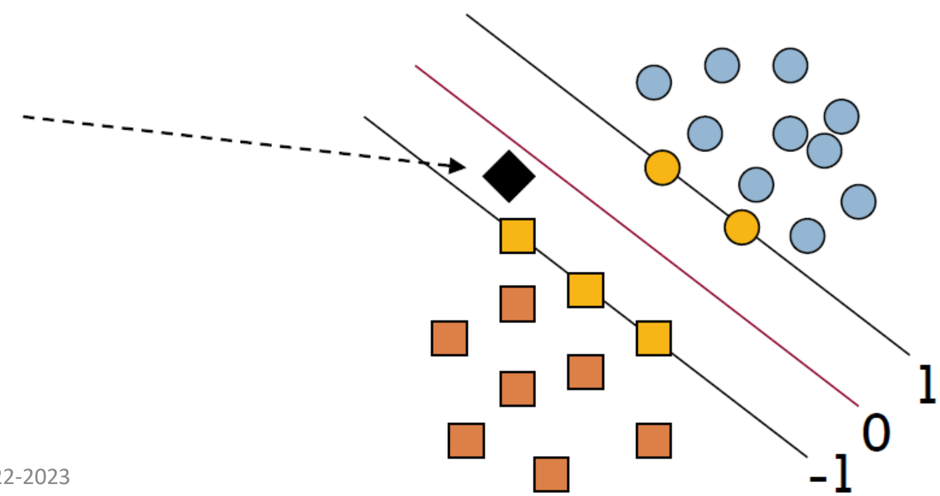
- if the training data is not linearly separable, slack variables ξ_i can be added to allow misclassification of difficult or noisy examples
- allow some errors
  - let some points be moved to where they belong, at a cost
- still, try to minimize training set errors, and to place hyperplane "far" from each class (large margin)



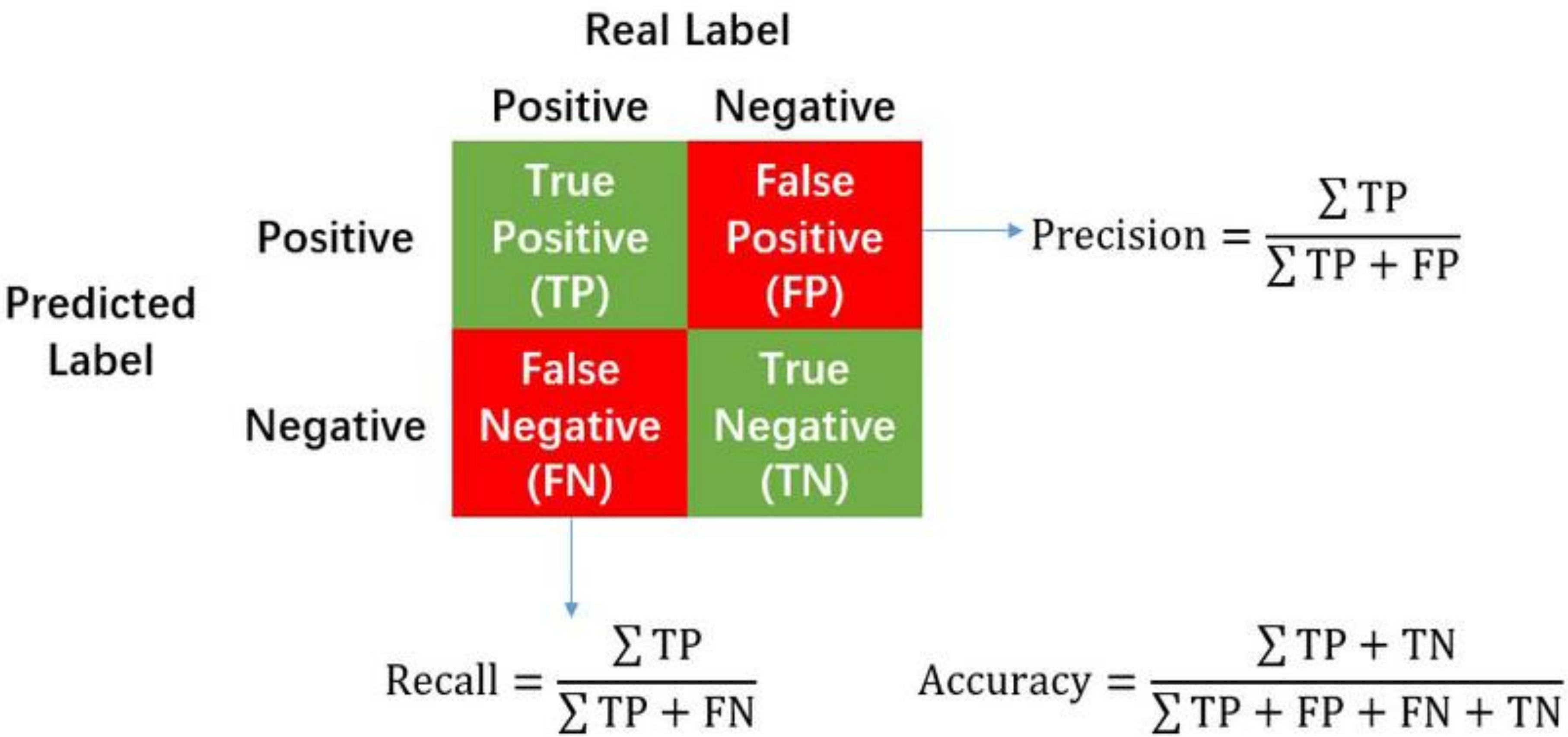
Classification with SVM

- given a new point x , we can score its projection onto the hyperplane normal:
  - i. e. compute score: $\mathbf{w}^{\mathbf{T}}\mathbf{x} + b = \Sigma \alpha_i y_i \mathbf{x}_{\mathbf{i}}^{\mathbf{T}}\mathbf{x} + b$
    - decide class based on whether < or > 0
- can set confidence threshold

Score > t: **yes**
Score < $-t$: **no**
Else: **don't know**



Metrics


Real Label
Positive
Negative
Predicted Label
Positive
Negative
True Positive (TP)
False Positive (FP)
False Negative (FN)
True Negative (TN)
$\text{Precision} = \frac{\sum \text{TP}}{\sum \text{TP} + \text{FP}}$
$\text{Recall} = \frac{\sum \text{TP}}{\sum \text{TP} + \text{FN}}$
$\text{Accuracy} = \frac{\sum \text{TP} + \text{TN}}{\sum \text{TP} + \text{FP} + \text{FN} + \text{TN}}$

QUESTIONS ?