北京航空航天大學
BEIHANG UNIVERSITY

大数据分析方法和技术

Classification

经济管理学院 刘冠男

Classification: Definition

北京航空航天大學
BEIHANG UNIVERSITY

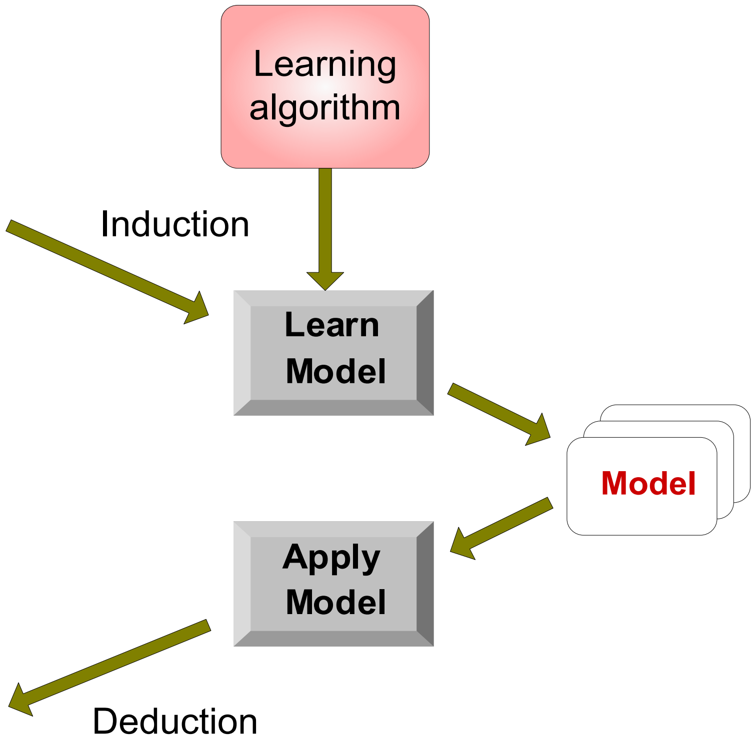
- **Given a collection of records (training set)**
  - Each record is by characterized by a tuple ($\boldsymbol{x}$, y), where $\boldsymbol{x}$ is the attribute set and y is the class label
    - $\boldsymbol{x}$: attribute, predictor, independent variable, input
    - y: class, response, dependent variable, output

- **Task:**
  - Learn a model that maps each attribute set $\boldsymbol{x}$ into one of the predefined class labels y

General Approach for Building Classification Model

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

Classification Techniques

- **Base Classifiers**
  - Rule-based Methods
  - Decision Tree based Methods
  - Support Vector Machines
  - Nearest-neighbor
  - Neural Networks
  - Naïve Bayes and Bayesian Belief Networks
- **Ensemble Classifiers**
  - Boosting, Bagging, Random Forests

Decision Tree Induction

- **Many Algorithms:**
  - Hunt’s Algorithm (one of the earliest)
  - CART
  - ID3, C4.5
  - SLIQ,SPRINT

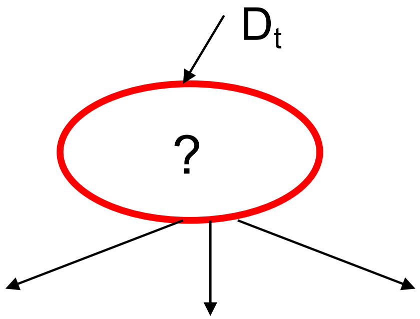
General Structure of Hunt's Algorithm

北京航空航天大學
BEIHANG UNIVERSITY

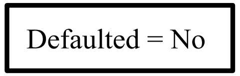
- Let D_t be the set of training records that reach a node t

- General Procedure:
  - If D_t contains records that belong the same class y_t, then t is a leaf node labeled as y_t
  - If D_t contains records that belong to more than one class, use an attribute test to split the data into smaller subsets. Recursively apply the procedure to each subset.

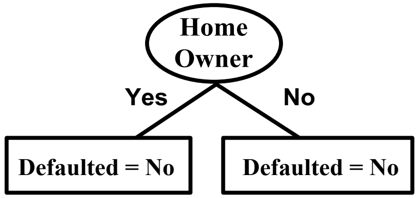
ID	Home Owner	Marital Status	Annual Income	Defaulted Borrower
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes



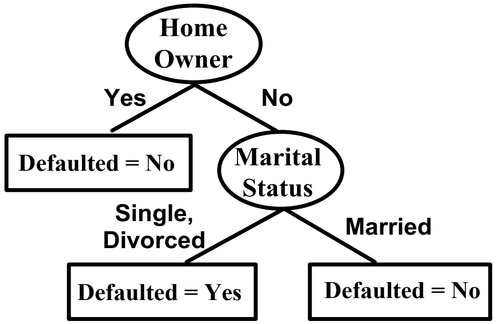
Hunt’s Algorithm



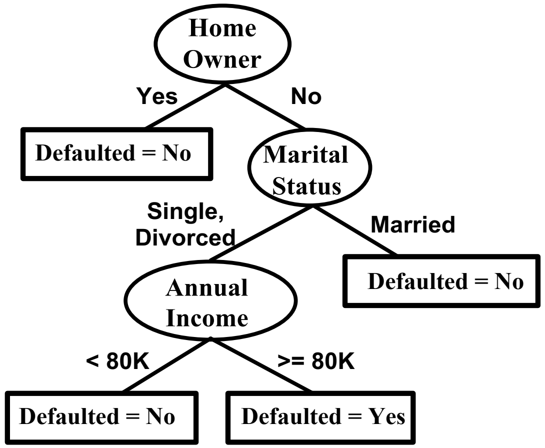
(a)



(b)



(c)



(d)

ID	Home Owner	Marital Status	Annual Income	Defaulted Borrower
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

Design Issues of Decision Tree Induction

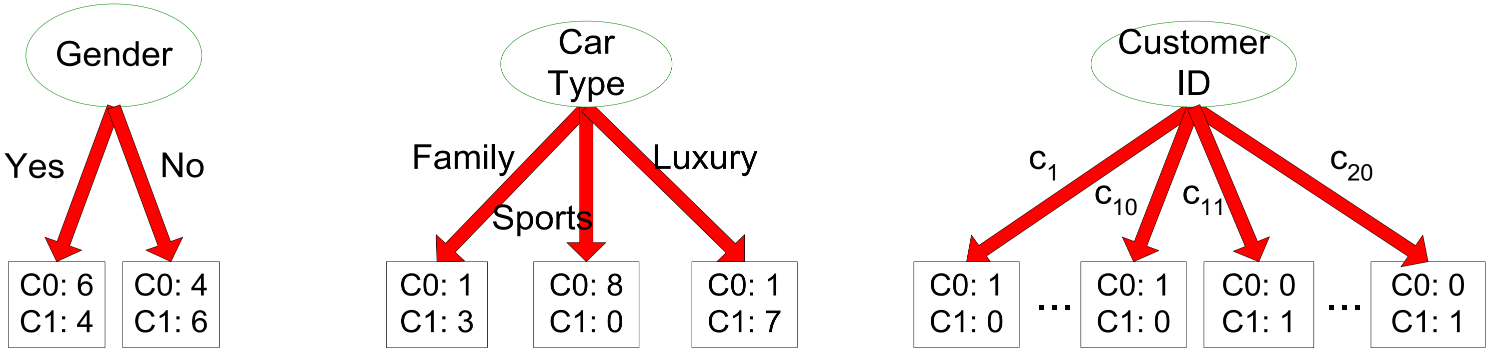
- **How should training records be split?**
  - Method for specifying test condition
    - depending on attribute types
  - Measure for evaluating the goodness of a test condition

- **How should the splitting procedure stop?**
  - Stop splitting if all the records belong to the same class or have identical attribute values
  - Early termination

How to determine the Best Split

Customer Id	Gender	Car Type	Shirt Size	Class
1	M	Family	Small	C0
2	M	Sports	Medium	C0
3	M	Sports	Medium	C0
4	M	Sports	Large	C0
5	M	Sports	Extra Large	C0
6	M	Sports	Extra Large	C0
7	F	Sports	Small	C0
8	F	Sports	Small	C0
9	F	Sports	Medium	C0
10	F	Luxury	Large	C0
11	M	Family	Large	C1
12	M	Family	Extra Large	C1
13	M	Family	Medium	C1
14	M	Luxury	Extra Large	C1
15	F	Luxury	Small	C1
16	F	Luxury	Small	C1
17	F	Luxury	Medium	C1
18	F	Luxury	Medium	C1
19	F	Luxury	Medium	C1
20	F	Luxury	Large	C1

**Before Splitting: 10 records of class 0,
10 records of class 1**



Which test condition is the best?

How to determine the Best Split

- Greedy approach:
  - Nodes with purer class distribution are preferred

- Need a measure of node impurity:

C0: 5 C1: 5	C0: 9 C1: 1
High degree of impurity	Low degree of impurity

Measures of Node Impurity

- Gini Index

$$GINI(t) = 1 - \sum_j [p(j\,|\,t)]^2$$

- Entropy

$$Entropy(t) = -\sum_j p(j\,|\,t)\log p(j\,|\,t)$$

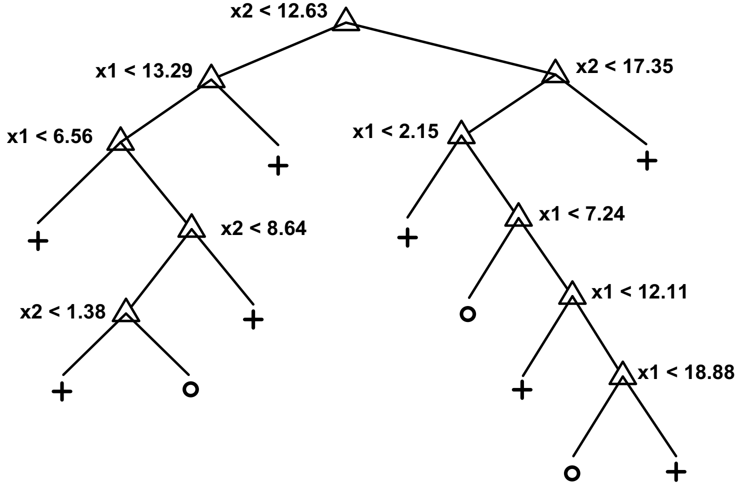
- Misclassification error

$$Error(t) = 1 - \max_i P(i\,|\,t)$$

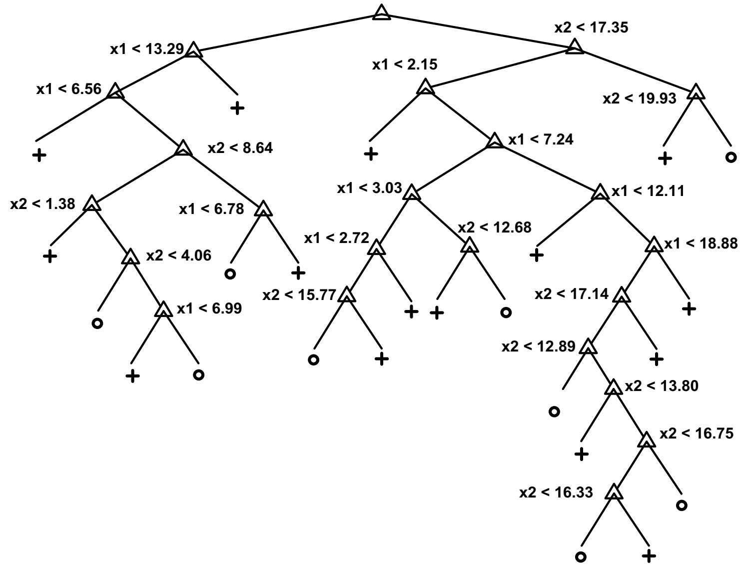
Classification Errors

- **Training errors (apparent errors)**
  - Errors committed on the training set
- **Test errors**
  - Errors committed on the test set
- **Generalization errors**
  - Expected error of a model over random selection of records from same distribution

Decision Tree



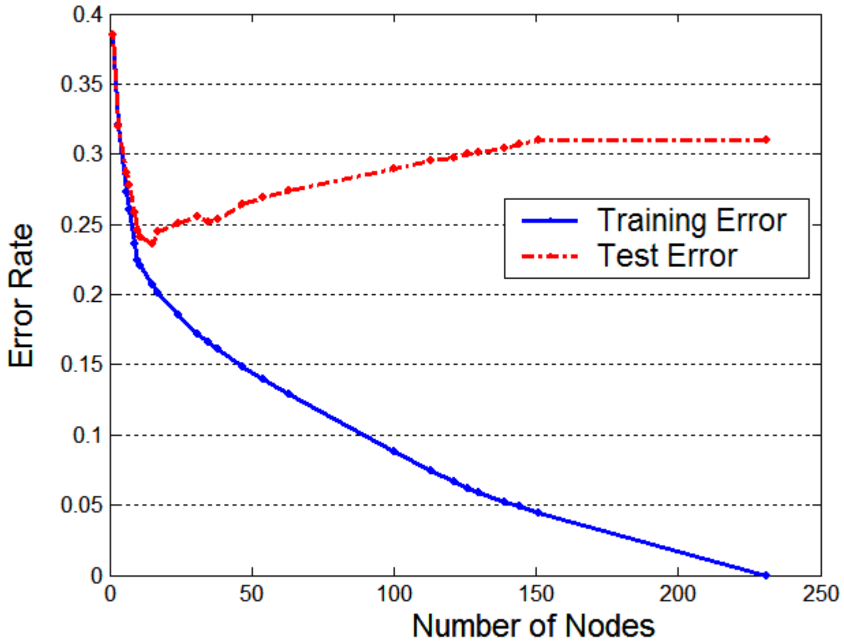
Decision Tree with 11 leaf nodes



Decision Tree with 24 leaf nodes

Which tree is better?

Model Overfitting

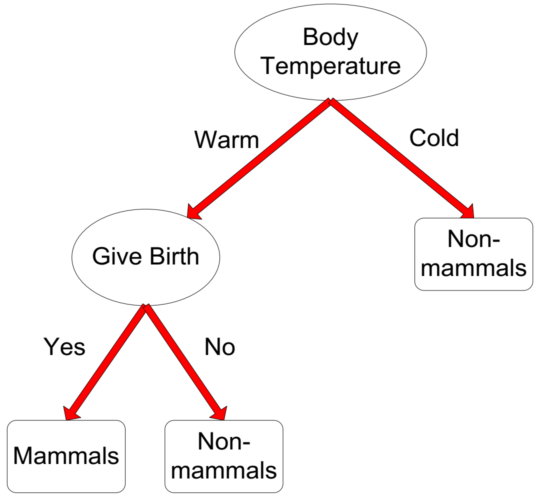


Underfitting: when model is too simple, both training and test errors are large

Overfitting: when model is too complex, training error is small but test error is large

Mammal Classification Problem

Name	Body Temperature	Skin Cover	Gives Birth	Aquatic Creature	Aerial Creature	Has Legs	Hiber-nates	Mammal
human	warm-blooded	hair	yes	no	no	yes	no	yes
python	cold-blooded	scales	no	no	no	no	yes	no
salmon	cold-blooded	scales	no	yes	no	no	no	no
whale	warm-blooded	hair	yes	yes	no	no	no	yes
frog	cold-blooded	none	no	semi	no	yes	yes	no
komodo dragon	cold-blooded	scales	no	no	no	yes	no	no
bat	warm-blooded	hair	yes	no	yes	yes	yes	yes
pigeon	warm-blooded	feathers	no	no	yes	yes	no	no
cat	warm-blooded	fur	yes	no	no	yes	no	yes
leopard shark	cold-blooded	scales	yes	yes	no	no	no	no
turtle	cold-blooded	scales	no	semi	no	yes	no	no
penguin	warm-blooded	feathers	no	semi	no	yes	no	no
porcupine	warm-blooded	quills	yes	no	no	yes	yes	yes
eel	cold-blooded	scales	no	yes	no	no	no	no
salamander	cold-blooded	none	no	semi	no	yes	yes	no



Training Set

Decision Tree Model

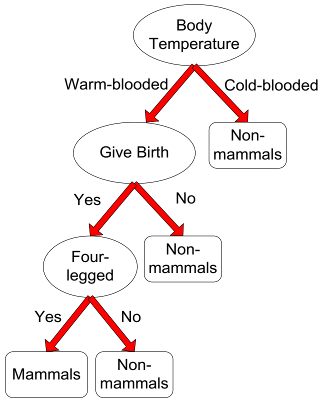
training error = 0%

Effect of Noise

Example: Mammal Classification problem

Training Set:

Name	Body Temperature	Gives Birth	Four-legged	Hibernates	Class Label
porcupine	warm-blooded	yes	yes	yes	yes
cat	warm-blooded	yes	yes	no	yes
bat	warm-blooded	yes	no	yes	no*
whale	warm-blooded	yes	no	no	no*
salamander	cold-blooded	no	yes	yes	no
komodo dragon	cold-blooded	no	yes	no	no
python	cold-blooded	no	no	yes	no
salmon	cold-blooded	no	no	no	no
eagle	warm-blooded	no	no	no	no
guppy	cold-blooded	yes	no	no	no



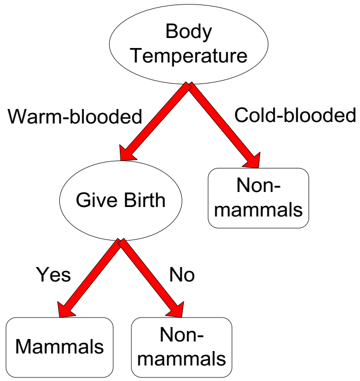
Model M1:

train err = 0%,

test err = 30%

Test Set:

Name	Body Temperature	Gives Birth	Four-legged	Hibernates	Class Label
human	warm-blooded	yes	no	no	yes
pigeon	warm-blooded	no	no	no	no
elephant	warm-blooded	yes	yes	no	yes
leopard shark	cold-blooded	yes	no	no	no
turtle	cold-blooded	no	yes	no	no
penguin	cold-blooded	no	no	no	no
eel	cold-blooded	no	no	no	no
dolphin	warm-blooded	yes	no	no	yes
spiny anteater	warm-blooded	no	yes	yes	yes
gila monster	cold-blooded	no	yes	yes	no



Model M2:

train err = 20%,

test err = 10%

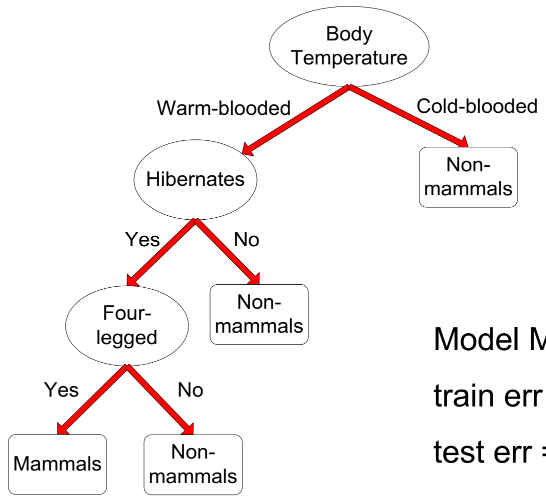
Lack of Representative Samples

Training Set:

Name	Body Temperature	Four-legged	Hibernates	Class Label
salamander	cold-blooded	yes	yes	no
guppy	cold-blooded	no	no	no
eagle	warm-blooded	no	no	no
poorwill	warm-blooded	no	yes	no
platypus	warm-blooded	yes	yes	yes

Test Set:

Name	Body Temperature	Four-legged	Hibernates	Class Label
human	warm-blooded	no	no	yes
pigeon	warm-blooded	no	no	no
elephant	warm-blooded	yes	no	yes
leopard shark	cold-blooded	no	no	no
turtle	cold-blooded	yes	no	no
penguin	cold-blooded	no	no	no
eel	cold-blooded	no	no	no
dolphin	warm-blooded	no	no	yes
spiny anteater	warm-blooded	yes	yes	yes
gila monster	cold-blooded	yes	yes	no



Model M3:

train err = 0%,

test err = 30%

Lack of training records at the leaf nodes for making reliable classification

The Source of Model Overfitting: Effect of Multiple Comparison Procedure

- **Consider the task of predicting whether stock market will rise/fall in the next 10 trading days**

- **Random guessing:**
  - P(correct) = 0.5

- **Make 10 random guesses in a row:**

$$P(\#correct \geq 8) = \frac{\binom{10}{8} + \binom{10}{9} + \binom{10}{10}}{2^{10}} = 0.0547$$

Day 1	Up
Day 2	Down
Day 3	Down
Day 4	Up
Day 5	Down
Day 6	Down
Day 7	Up
Day 8	Up
Day 9	Up
Day 10	Down

Effect of Multiple Comparison Procedure

北京航空航天大學 BEIHANG UNIVERSITY

- **Approach:**
  - Get 50 analysts
  - Each analyst makes 10 random guesses
  - Choose the analyst that makes the most number of correct predictions

- **Probability that at least one analyst makes at least 8 correct predictions**

$$P(\#correct \geq 8) = 1 - (1 - 0.0547)^{50} = 0.9399$$

Effect of Multiple Comparison Procedure

北京航空航天大學 BEIHANG UNIVERSITY

- **Many algorithms employ the following greedy strategy:**
  - Initial model: M
  - Alternative model: $M' = M \cup \gamma$, where γ is a component to be added to the model (e.g., a test condition of a decision tree)
  - Keep M' if improvement, $\Delta(M,M') > \alpha$
- **Often times, γ is chosen from a set of alternative components, $\Gamma = \{\gamma_1, \gamma_2, \ldots, \gamma_k\}$**
- **If many alternatives are available, one may inadvertently add irrelevant components to the model, resulting in model overfitting**

Multiple comparison in Decision Tree

北京航空航天大學 BEIHANG UNIVERSITY

- **T_0: initial decision tree**
- **T_x: x added to the tree**
  - Compare the gain $\Delta(T_0, T_x)$
- **x is determined from a set of candidates: x_{max}**
  - $x_1, x_2, \cdots, x_k$
- **The chance of finding $\Delta(T_0, T_x) \geq \alpha$ increases due to multiple comparisons.**

Notes on Overfitting

- Overfitting results in decision trees that are more complex than necessary

- Training error no longer provides a good estimate of how well the tree will perform on previously unseen records

- Need new ways for estimating generalization errors

Evaluating Performance of Classifier

- **Model Selection**
  - Performed during model building
  - Purpose is to ensure that model is not overly complex (to avoid overfitting)
  - Need to estimate generalization error

- **Model Evaluation**
  - Performed after model has been constructed
  - Purpose is to estimate performance of classifier on previously unseen data (e.g., test set)

Methods for Classifier Evaluation

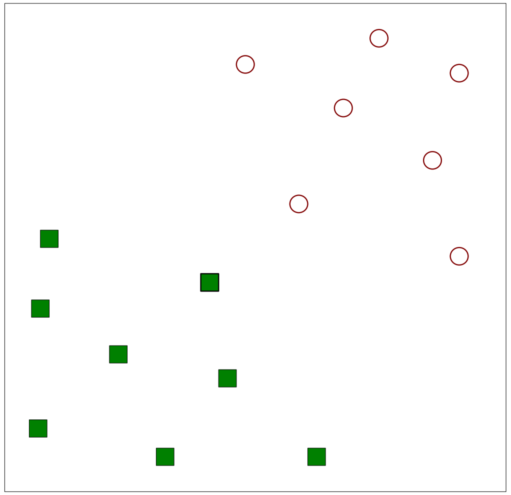
- **Holdout**
  - Reserve k% for training and (100-k)% for testing
- **Random subsampling**
  - Repeated holdout
- **Cross validation**
  - Partition data into k disjoint subsets
  - k-fold: train on k-1 partitions, test on the remaining one
  - Leave-one-out: k=n
- **Bootstrap**
  - Sampling with replacement
  - .632 bootstrap: $acc_{boot} = \frac{1}{b}\sum_{i=1}^{b}\left(0.632 \times acc_i + 0.368 \times acc_s\right)$

Support Vector Machine (SVM)

Support Vector Machines (SVMs)

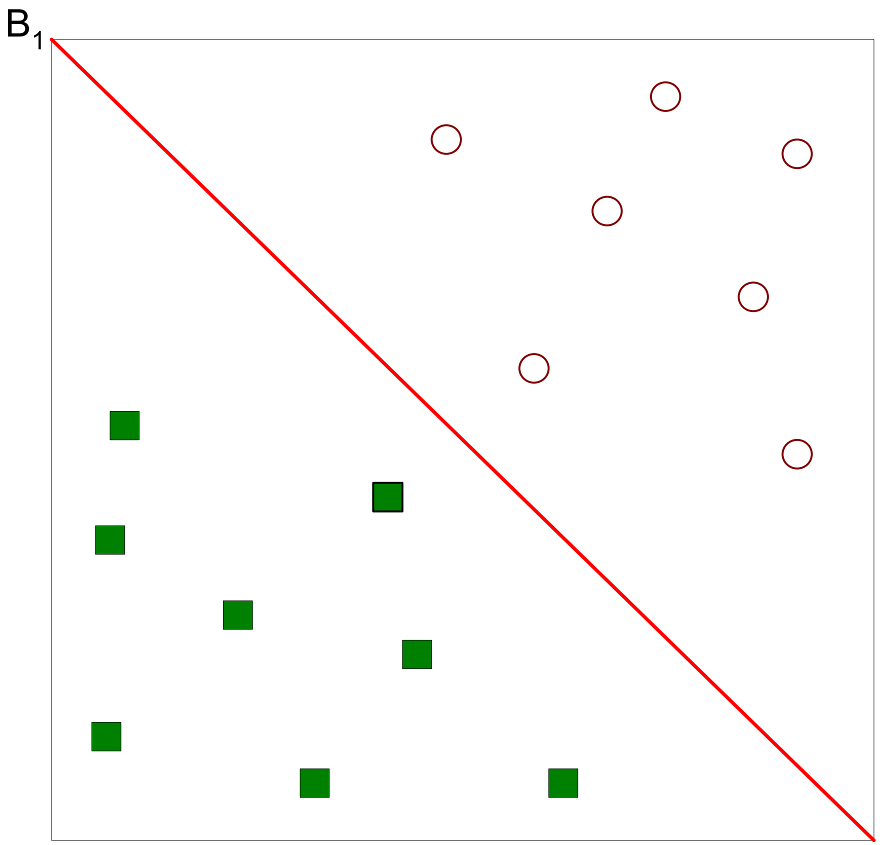
北京航空航天大學
BEIHANG UNIVERSITY

SVMs are a rare example of a methodology where geometric intuition, elegant mathematics, theoretical guarantees, and practical use meet.



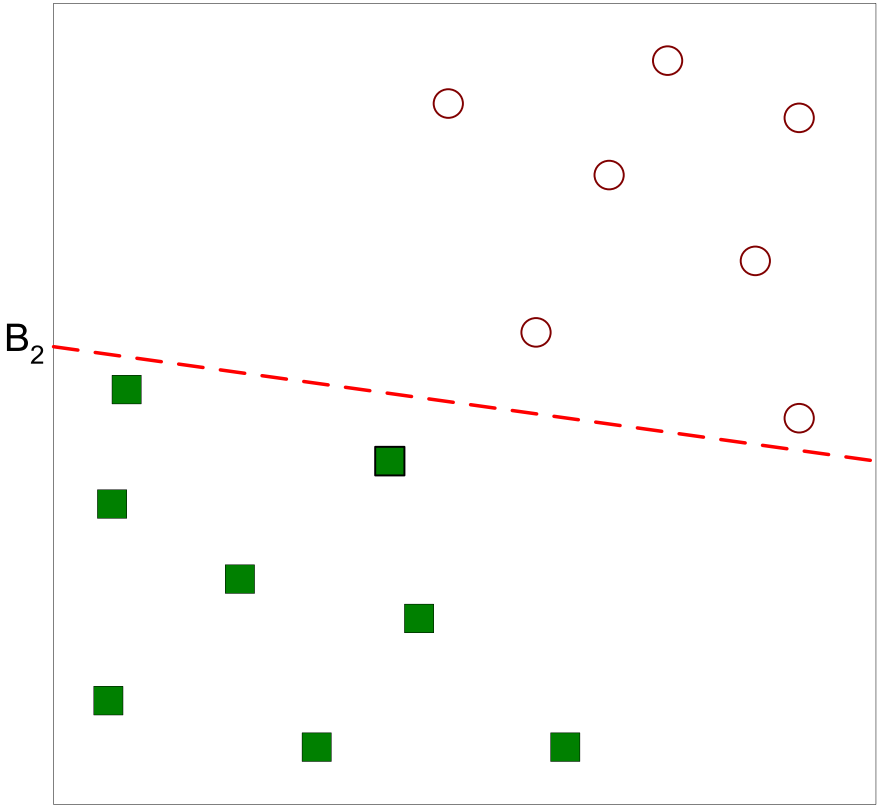
- **Core idea: find a linear hyperplane (decision boundary) that separates the data**

Support Vector Machines



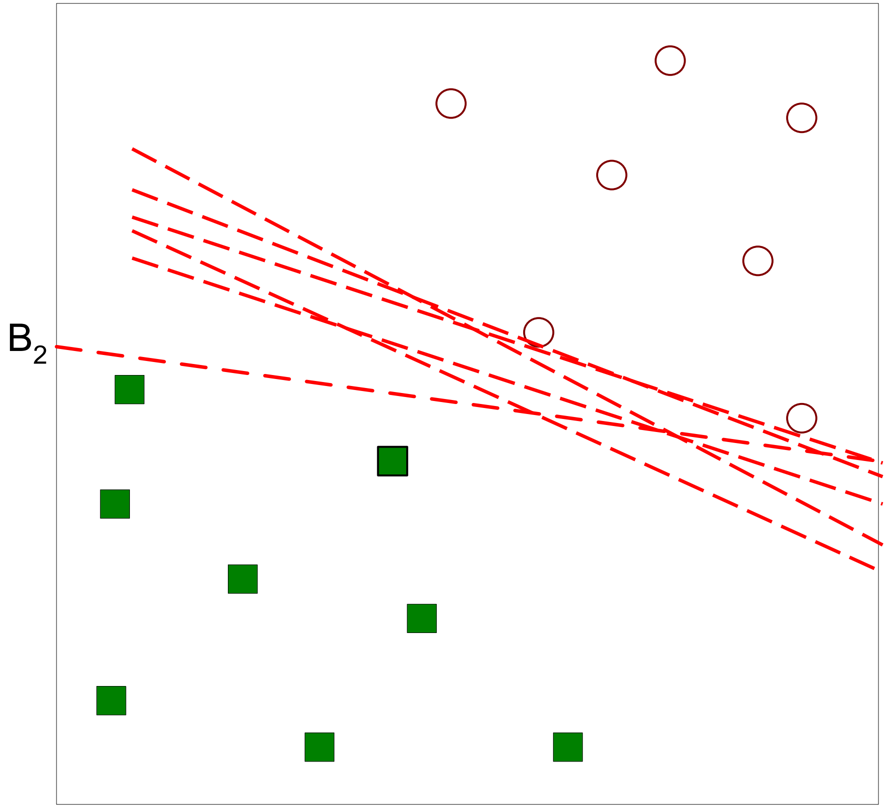
- One Possible Solution

Support Vector Machines



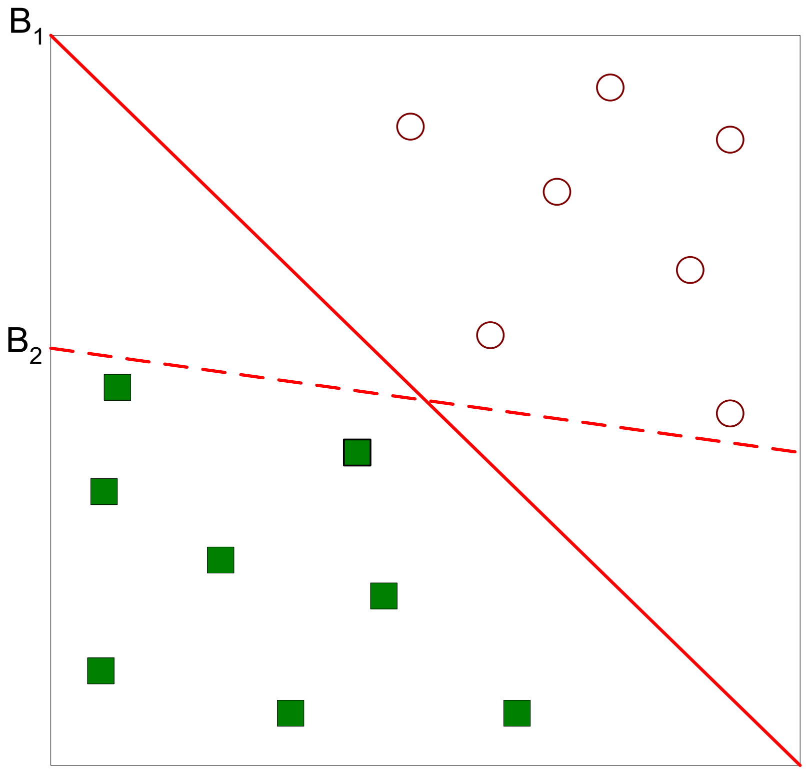
- Another possible solution

Support Vector Machines



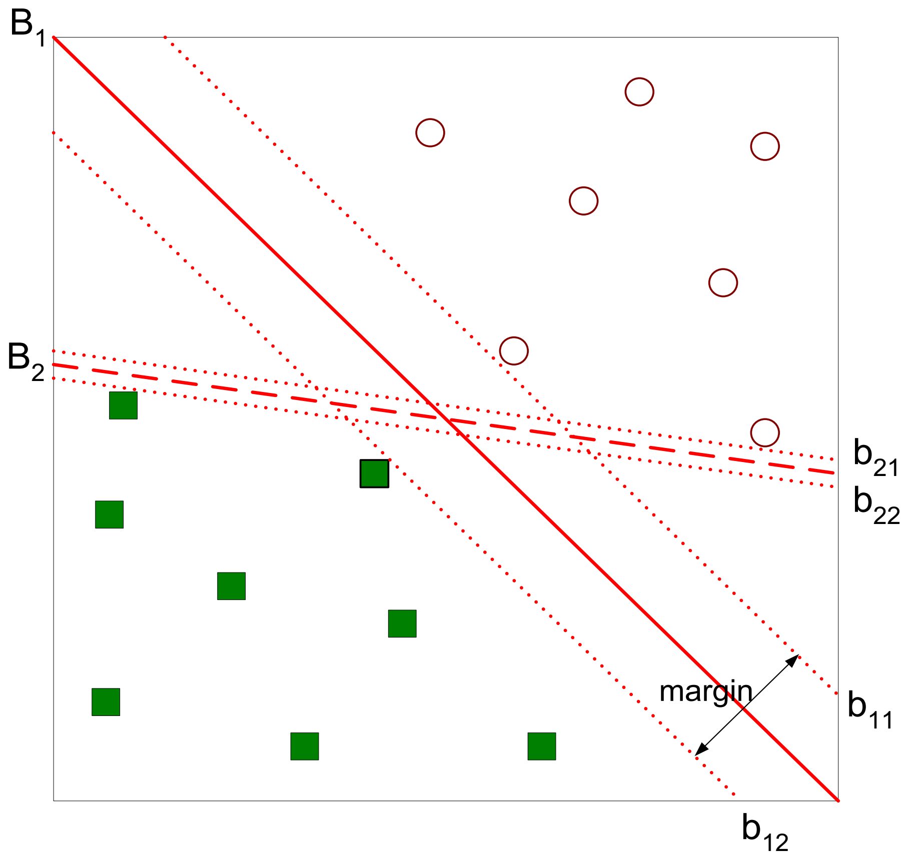
- Other possible solutions

Support Vector Machines



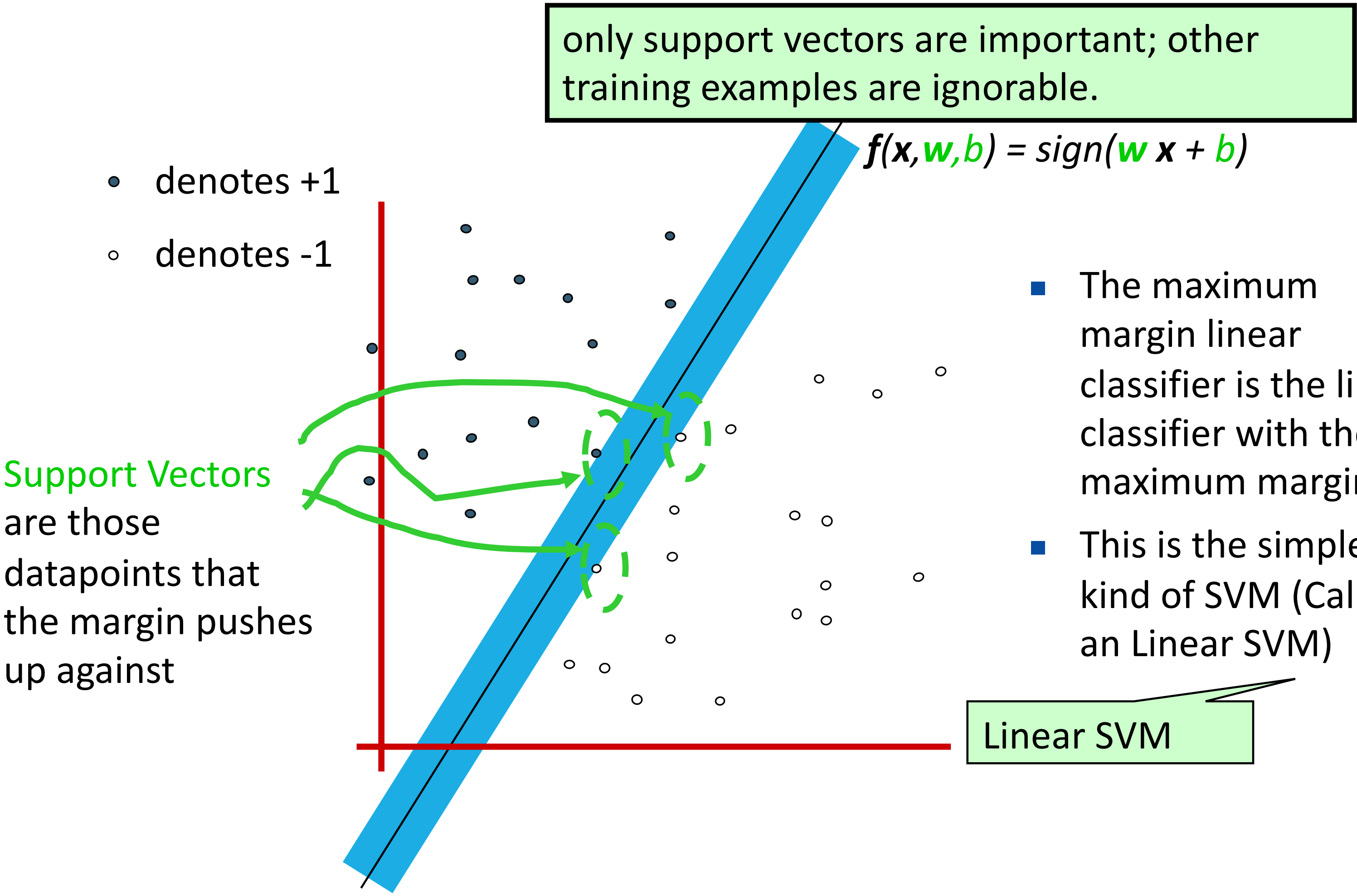
- Which one is better? B1 or B2?
- How do you define better?

Support Vector Machines



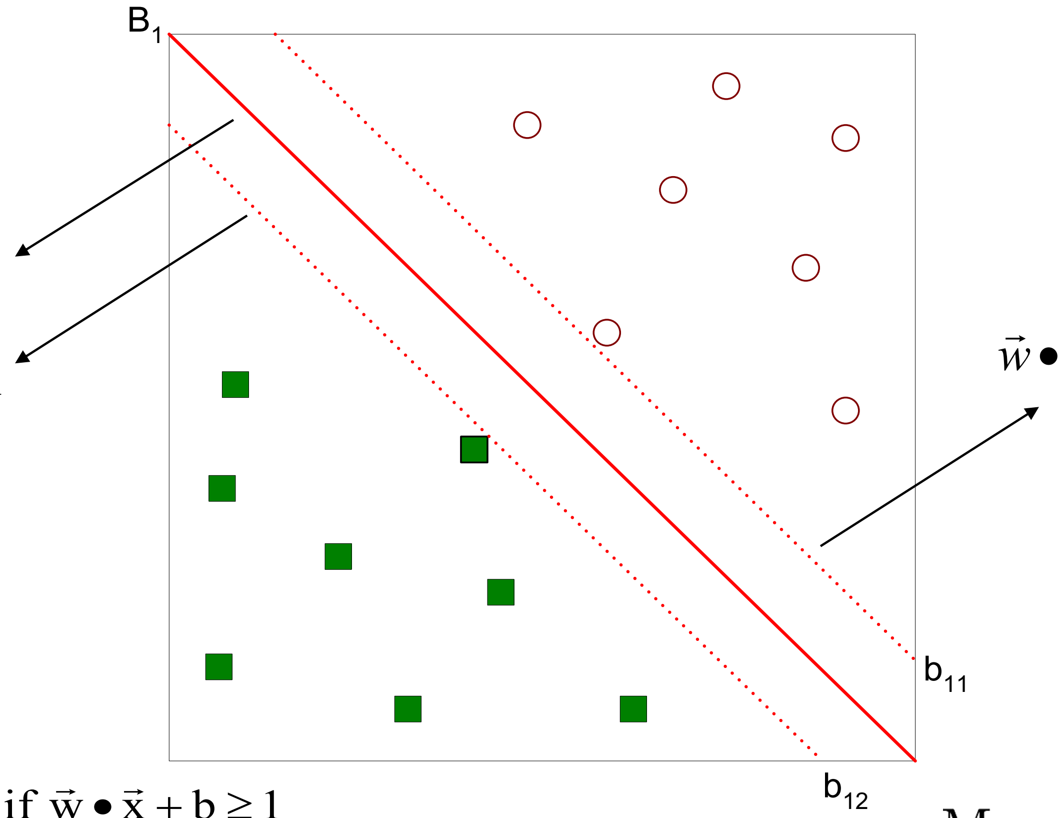
- Find hyperplane maximizes the margin => B1 is better than B2

Maximum Margin



- The maximum margin linear classifier is the linear classifier with the maximum margin.
- This is the simplest kind of SVM (Called an Linear SVM)

Support Vector Machines



$$\vec{w} \bullet \vec{x} + b = 0$$

$$\vec{w} \bullet \vec{x} + b = -1$$

$$\vec{w} \bullet \vec{x} + b = +1$$

$$f(\vec{x}) = \begin{cases} 1 & \text{if } \vec{\mathrm{w}} \bullet \vec{\mathrm{x}} + \mathrm{b} \geq 1 \\ -1 & \text{if } \vec{\mathrm{w}} \bullet \vec{\mathrm{x}} + \mathrm{b} \leq -1 \end{cases}$$

$$\text{Margin} = \frac{2}{||\vec{\mathrm{w}}||}$$

Support Vector Machines

- **We want to maximize:**

$$\text{Margin} = \frac{2}{||\vec{\mathrm{w}}||}$$

- **Which is equivalent to minimizing:**

$$L(w) = \frac{\|\vec{w}\|^2}{2}$$

- **But subjected to the following constraints:**

$$f(\vec{x}_i) = \begin{cases} 1 & \text{if } \vec{\mathrm{w}} \bullet \vec{\mathrm{x}}_\mathrm{i} + \mathrm{b} \geq 1 \\ -1 & \text{if } \vec{\mathrm{w}} \bullet \vec{\mathrm{x}}_\mathrm{i} + \mathrm{b} \leq -1 \end{cases} \Rightarrow y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1, \qquad \forall i$$

- **This is a constrained optimization problem**
  - Numerical approaches to solve it (e.g., quadratic programming)

Lagrangian of Original Problem

北京航空航天大學 BEIHANG UNIVERSITY

- **Lagrange multiplier**

Lagrangian multipliers

$$\mathcal{L} = \frac{1}{2}\mathbf{w}^T\mathbf{w} + \sum_{i=1}^{n} \alpha_i \left(1 - y_i(\mathbf{w}^T\mathbf{x}_i + b)\right) \qquad \alpha_i \geq 0$$

- **Partial derivative of L w.r.t w and b**

$$\mathbf{w} + \sum_{i=1}^{n} \alpha_i(-y_i)\mathbf{x}_i = \mathbf{0} \quad \Rightarrow \quad \boxed{\mathbf{w} = \sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i}$$

$$\boxed{\sum_{i=1}^{n} \alpha_i y_i = 0}$$

Dual Problem

Dot product of X

$$\max.\ W(\boldsymbol{\alpha}) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1,j=1}^{n} \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j$$

$$\text{subject to } \alpha_i \geq 0, \sum_{i=1}^{n} \alpha_i y_i = 0$$

α's → New variables
(Lagrangian multipliers)

- **Quadratic programming**
  - Global maximum of α_i can always be found
  - Well established tools for solving this optimization problem

KKT Condition

北京航空航天大學 BEIHANG UNIVERSITY

$$\min.: f(\mathbf{x})$$

$$\begin{aligned} \text{s.t.}: \quad & g_i(\mathbf{x}) \leq 0, i = 1, 2, \ldots, p, \\ & h_j(\mathbf{x}) = 0, k = 1, 2, \ldots, q, \\ & \mathbf{x} \in \Omega \subset \mathbf{R}^n \end{aligned}$$

1). 约束条件满足$g_i(\mathbf{x}^*) \leq 0, i = 1, 2, \ldots, p$, 以及, $h_j(\mathbf{x}^*) = 0, j = 1, 2, \ldots, q$

2). $\nabla f(\mathbf{x}^*) + \sum_{i=1}^{p} \mu_i \nabla g_i(\mathbf{x}^*) + \sum_{j=1}^{q} \lambda_j \nabla h_j(\mathbf{x}^*) = \mathbf{0}$, 其中$\nabla$为梯度算子;

3). $\lambda_j \neq 0$且不等式约束条件满足$\mu_i \geq 0$, $\boxed{\mu_i g_i(\mathbf{x}^*) = 0}$, $i = 1, 2, \ldots, p$

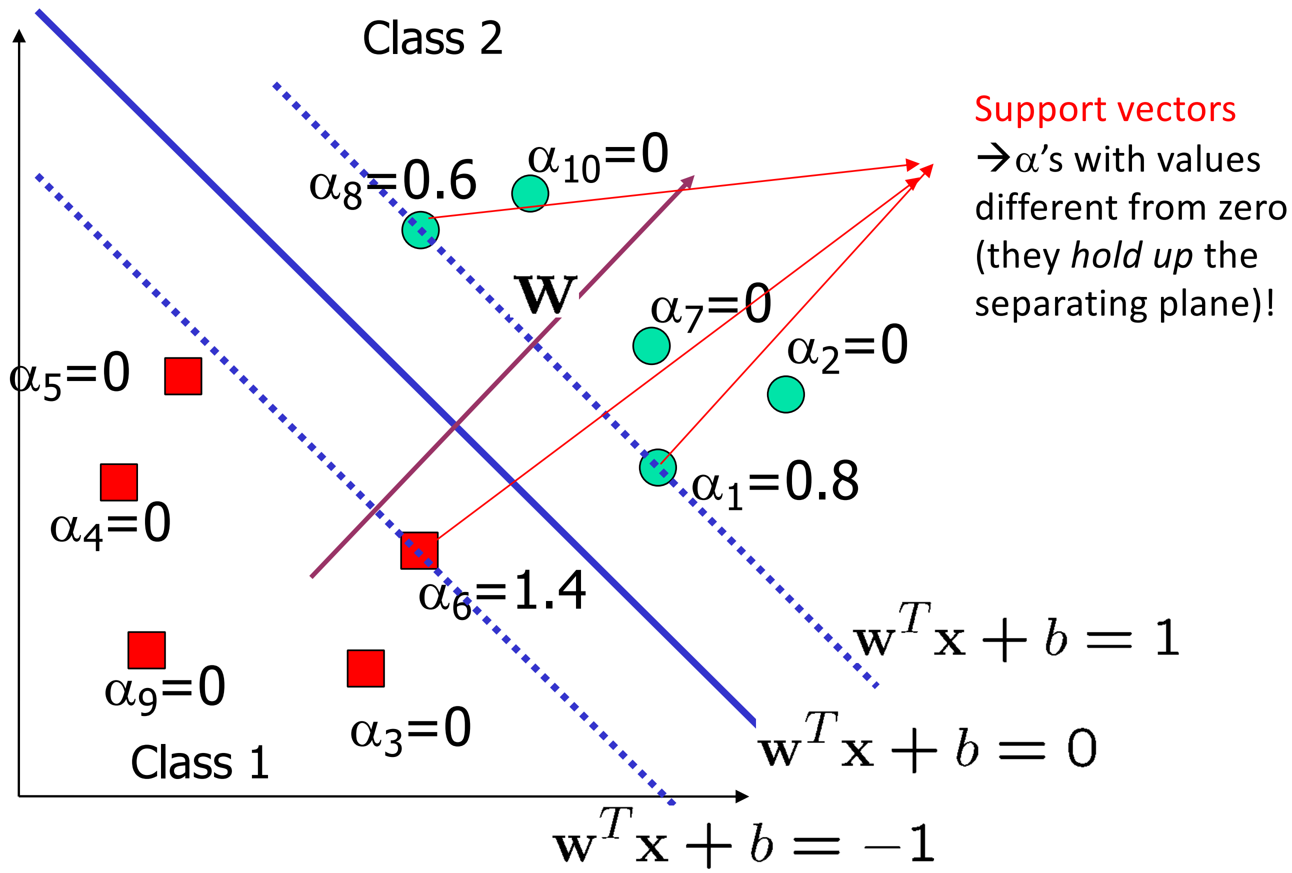
Support Vectors

北京航空航天大學 BEIHANG UNIVERSITY

- **KKT Conditions for SVM**

$$\begin{cases} \alpha_i \geq 0 \\ y_i f(\mathbf{x}_i) - 1 \geq 0 \\ \boxed{\alpha_i (y_i f(\mathbf{x}_i) - 1) = 0} \end{cases}$$

Support Vectors



Sequential Minimal Optimization

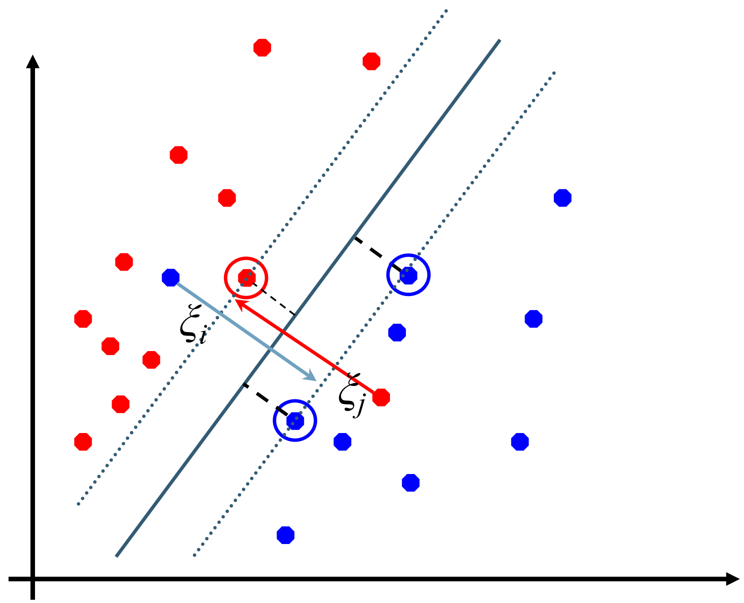
$$\text{max. } W(\boldsymbol{\alpha}) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1,j=1}^{n} \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j$$

$$\text{subject to } \alpha_i \geq 0, \sum_{i=1}^{n} \alpha_i y_i = 0$$

- **Fix the variables except for α_i, obtain the value that maximizes the objective**
  - May not enough because of the constraints
- **Choose a pair of variables α_i and α_j**
- **Fix the other to obtain the optima**
- $\alpha_i y_i + \alpha_j y_j = c, \alpha_i \geq 0, \alpha_j \geq 0$ → uni-variable QP

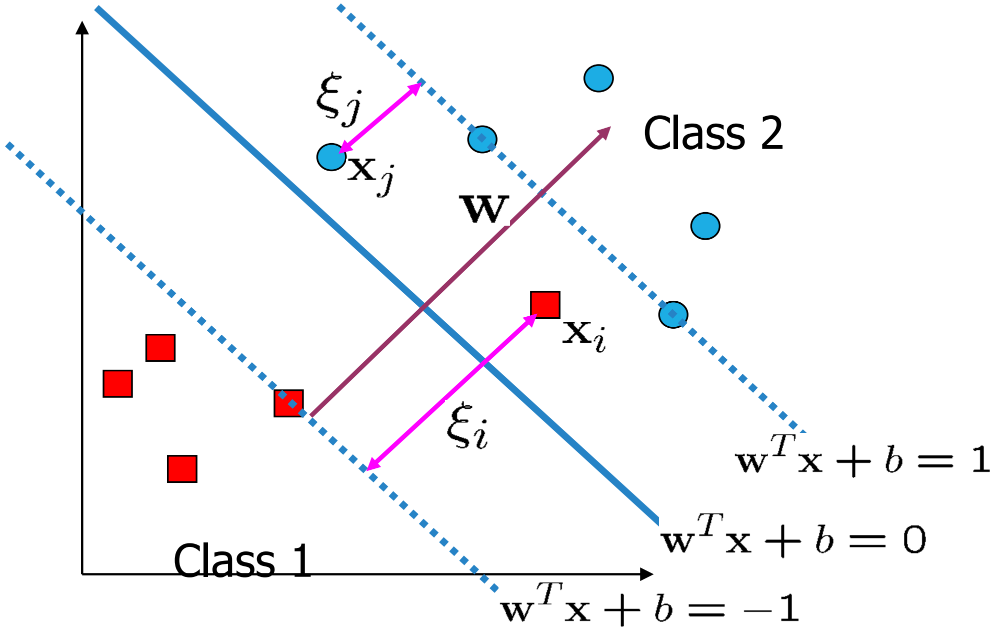
Soft Margin

- **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)**



Non-linearly Separable Problems

- We allow "error" ξ_i in classification; it is based on the output of the discriminant function w^Tx+b
- ξ_i approximates the number of misclassified samples



New objective function:

$$\frac{1}{2}||\mathbf{w}||^2 + C\sum_{i=1}^{n}\xi_i$$

C : tradeoff parameter between error and margin; chosen by the user; large C means a higher penalty to errors

Soft Margin Optimization

北京航空航天大學 BEIHANG UNIVERSITY

$$\text{minimize} \quad \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{n}\xi_i$$

$$\mathbf{w}\mathbf{x}_i + b \geq 1 - \xi_i \quad \text{if } y_i = 1$$

$$\mathbf{w}\mathbf{x}_i + b \leq -1 + \xi_i \quad \text{if } y_i = -1$$

$$C > 0$$

$$y_i(w^t x_i + b) \geq 1 - \xi_i \quad \forall i \quad \xi_i \geq 0$$

The Optimization Problem

- The dual of the problem is

$$\max.\ W(\boldsymbol{\alpha}) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1,j=1}^{n} \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j$$

$$\text{subject to } \boxed{C \geq \alpha_i \geq 0,} \sum_{i=1}^{n} \alpha_i y_i = 0$$

$$\mathbf{w} = \sum_{j=1}^{s} \alpha_{t_j} y_{t_j} \mathbf{x}_{t_j}$$

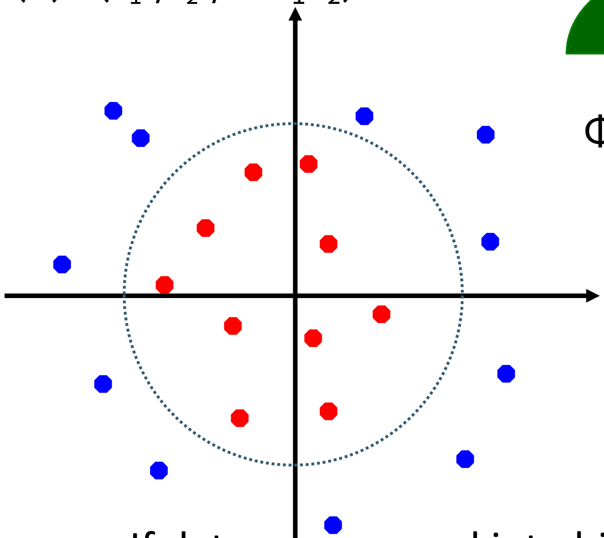
- The only difference with the linear separable case is that there is an upper bound *C* on α_i
- Once again, a **QP solver can be used to find α_i efficiently!!!**

Non Linear SVM

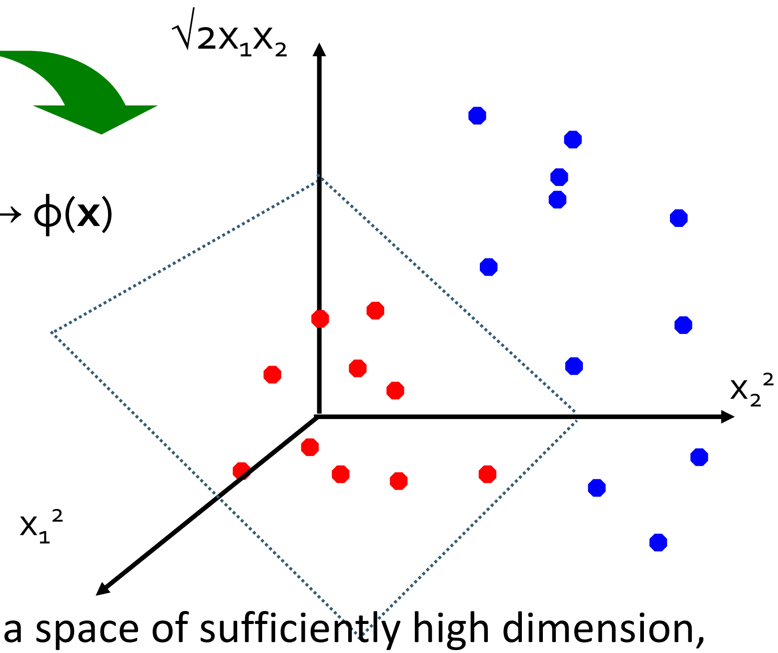
General idea: the original input space (x) can be mapped to some higher-dimensional feature space $\phi(\mathbf{x})$ where the training set is separable:

$x=(x_1,x_2)$

$\phi(\mathbf{x}) = (x_1^2, x_2^2, \sqrt{2}x_1x_2)$



$\Phi\colon\ \mathbf{x} \rightarrow \phi(\mathbf{x})$



If data are mapped into higher a space of sufficiently high dimension, then they will in general be linearly separable;

N data points are in general separable in a space of N-1 dimensions or more!!!

This slide is courtesy of *www.iro.umontreal.ca/~pift6080/documents/papers/**svm**_tutorial.**ppt***

Mapping into a New Feature Space

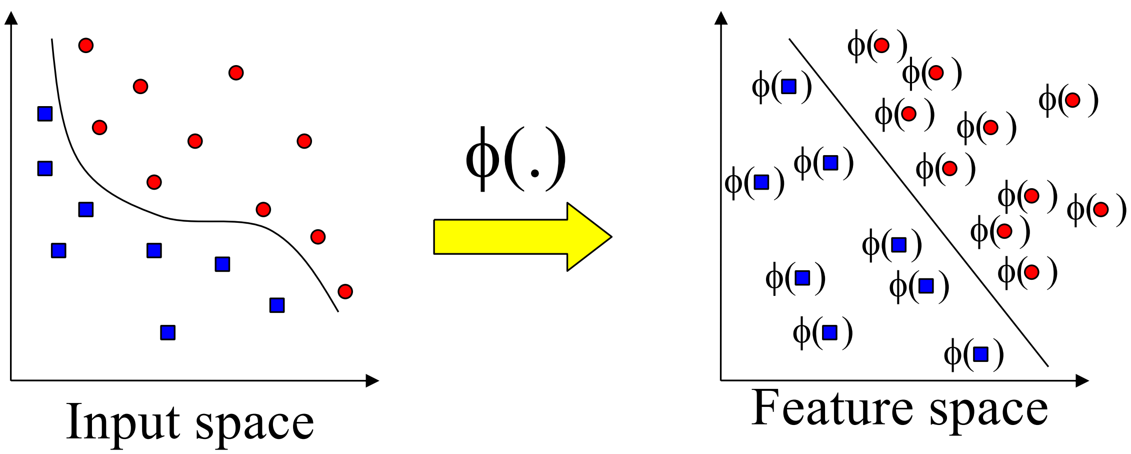
$$\Phi : x \rightarrow X = \Phi(x)$$

$$\Phi(x_1,x_2) = (x_1,x_2,x_1^2,x_2^2,x_1x_2)$$

- Rather than run SVM on x_i, run it on $\Phi(x_i)$
- Find non-linear separator in input space
- What if $\Phi(x_i)$ is really big?
- Use kernels!

Transformation to Feature Space

- **Possible problem of the transformation**
  - High computation burden due to high-dimensionality and hard to get a good estimate
- **SVM solves these two issues simultaneously**
  - “Kernel tricks” for efficient computation
  - Minimize $||\mathbf{w}||^2$ can lead to a “good” classifier



Example Transformation

- Consider the following transformation

$$\phi(\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}) = (1, \sqrt{2}x_1, \sqrt{2}x_2, x_1^2, x_2^2, \sqrt{2}x_1 x_2)$$

$$\phi(\begin{bmatrix} y_1 \\ y_2 \end{bmatrix}) = (1, \sqrt{2}y_1, \sqrt{2}y_2, y_1^2, y_2^2, \sqrt{2}y_1 y_2)$$

$$\begin{aligned} \langle \phi(\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}), \phi(\begin{bmatrix} y_1 \\ y_2 \end{bmatrix}) \rangle &= (1 + x_1 y_1 + x_2 y_2)^2 \\ &= K(\mathbf{x}, \mathbf{y}) \end{aligned}$$

- Define the kernel function *K* (x,y) as

$$K(\mathbf{x}, \mathbf{y}) = (1 + x_1 y_1 + x_2 y_2)^2$$

- The inner product ϕ(.)ϕ(.) can be computed by *K* **without going through the map ϕ(.) explicitly!!!**
- Principle: The Kernel function can always be expressed as the **dot product between two input vectors**

Examples of Kernel Functions

- **Mercer's Theorem**
  - Kernel function for a pair of vectors == dot product in transformed space
- **Polynomial kernel with degree *d***

$$K(\mathbf{x}, \mathbf{y}) = (\mathbf{x}^T \mathbf{y} + 1)^d$$

- **Radial basis function kernel with width σ**

$$K(\mathbf{x}, \mathbf{y}) = \exp(-||\mathbf{x} - \mathbf{y}||^2 / (2\sigma^2))$$

- **Sigmoid with parameter κ and θ**
  - It does not satisfy the Mercer condition on all κ and θ

$$K(\mathbf{x}, \mathbf{y}) = \tanh(\kappa \mathbf{x}^T \mathbf{y} + \theta)$$

Kernel Trick ☺

Note that data only appears as dot products

$$\sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=j=1}^{N} \alpha_i \alpha_j y_i y_j x_i x_j$$

$$C \geq \alpha_i \geq 0, \sum_{i=1}^{N} \alpha_i y_i = 0$$

$$K(x_i, x_j) = \phi(x_i) \cdot \phi(x_j)$$

Modification Due to Kernel Function

- **Change all inner products to kernel functions**
- **For training,**

Original

$$\text{max. } W(\boldsymbol{\alpha}) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1,j=1}^{n} \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j$$

$$\text{subject to } C \geq \alpha_i \geq 0, \sum_{i=1}^{n} \alpha_i y_i = 0$$

With kernel function

$$\text{max. } W(\boldsymbol{\alpha}) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1,j=1}^{n} \alpha_i \alpha_j y_i y_j K(\mathbf{x}_i, \mathbf{x}_j)$$

$$\text{subject to } C \geq \alpha_i \geq 0, \sum_{i=1}^{n} \alpha_i y_i = 0$$

Choosing the Kernel Function

- Probably the most tricky part of using SVM.
- The kernel function is important because it creates the kernel matrix, which summarizes all the data
- Many principles have been proposed (diffusion kernel, Fisher kernel, string kernel, …)
- There is even research to estimate the kernel matrix from available information
- In practice, a low degree polynomial kernel or RBF kernel with a reasonable width is a good initial try
- Note that SVM with RBF kernel is closely related to RBF neural networks, with the centers of the radial basis functions automatically chosen for SVM

SVM Summary

- **Mathematical formulation**
- **Optimization**
  - Non linear programming
  - Quadratic programming
- **Regularization**
- **Kernel tricks**

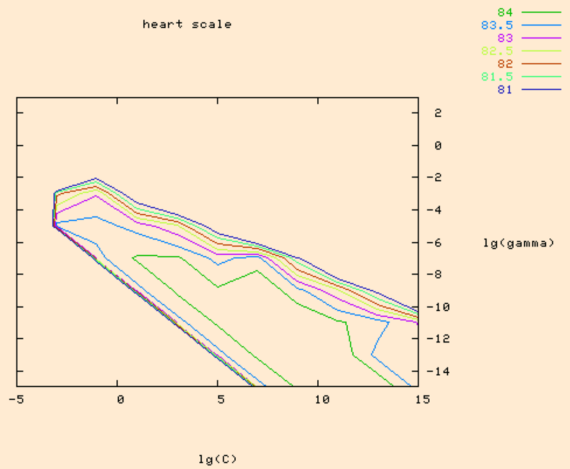
Tools

- LIBSVM

LIBSVM -- A Library for Support Vector Machines

Chih-Chung Chang and Chih-Jen Lin

NEW Version 3.21 released on December 14, 2015. It conducts some minor fixes.
NEW LIBSVM tools provides **many extensions** of LIBSVM. Please check it if you need some functions not supported in LIBSVM.
NEW We now have a nice page LIBSVM data sets providing problems in LIBSVM format.
NEW A practical guide to SVM classification is available now! (mainly written for beginners)
We now have an easy script (easy.py) for users who know NOTHING about SVM. It makes everything automatic--from data scaling to parameter selection.
The parameter selection tool grid.py generates the following contour of cross-validation accuracy. To use this tool, you also need to install python and gnuplot.



Naïve Bayes Classification

Background

- **There are three methods to establish a classifier**

_a_) Model a classification rule directly

Examples: k-NN, decision trees, perceptron, SVM

b) Model the probability of class memberships given input data

Example: perceptron with the cross-entropy cost

_c_) Make a probabilistic model of data within each class

Examples: naive Bayes, model based classifiers

- ***a*) and *b*) are examples of discriminative classification**
- ***c*) is an example of generative classification**
- ***b*) and *c*) are both examples of probabilistic classification**

Probability Basics

- Prior, conditional and joint probability for random variables
  - Prior probability: $P(X)$
  - Conditional probability: $P(X_1 \mid X_2), P(X_2 \mid X_1)$
  - Joint probability: $\mathbf{X} = (X_1, X_2), P(\mathbf{X}) = P(X_1, X_2)$
  - Relationship: $P(X_1, X_2) = P(X_2 \mid X_1)P(X_1) = P(X_1 \mid X_2)P(X_2)$
  - Independence: $P(X_2 \mid X_1) = P(X_2), P(X_1 \mid X_2) = P(X_1), P(X_1, X_2) = P(X_1)P(X_2)$
- Bayesian Rule

$$P(C \mid \mathbf{X}) = \frac{P(\mathbf{X} \mid C)P(C)}{P(\mathbf{X})} \qquad Posterior = \frac{Likelihood \times Prior}{Evidence}$$

Does patient have cancer or not?

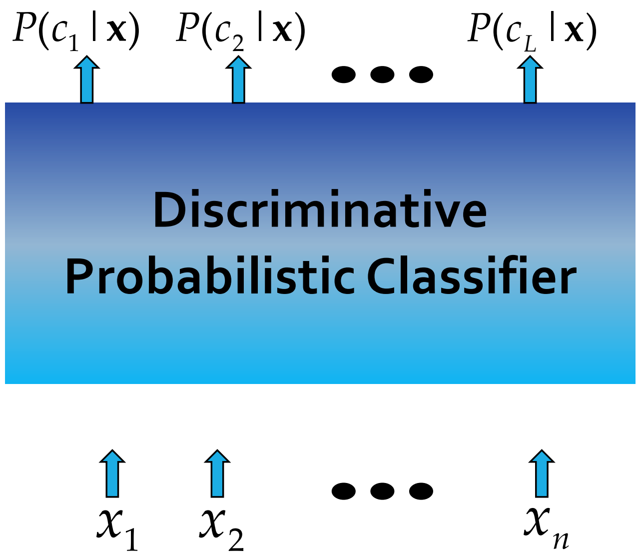
- **A patient takes a lab test and the result comes back positive. It is known that the test returns a correct positive result in only 99% of the cases and a correct negative result in only 95% of the cases. Furthermore, only 0.03 of the entire population has this disease.**

1. What is the probability that this patient has cancer?
2. What is the probability that he does not have cancer?
3. What is the diagnosis?

Probabilistic Classification

- Establishing a probabilistic model for classification
  - **Discriminative model**

$$\boxed{P(C\,|\,\mathbf{X}) \quad C = c_1,\cdots,c_L,\ \mathbf{X}=(X_1,\cdots,X_n)}$$

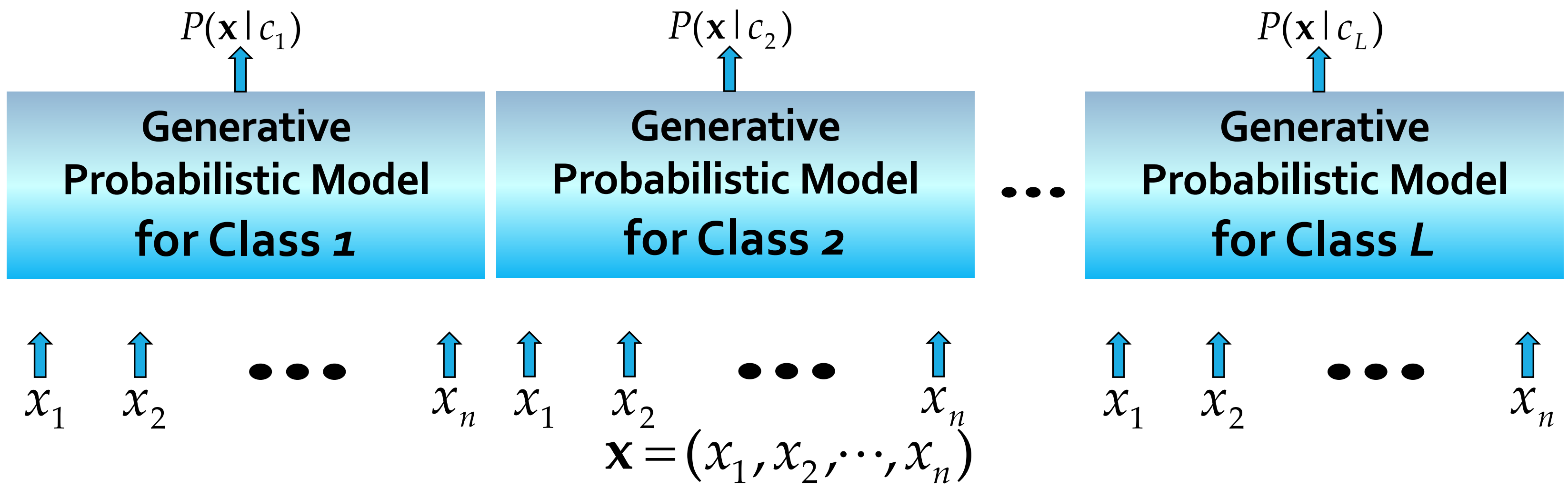


$$\mathbf{x}=(x_1,x_2,\cdots,x_n)$$

Probabilistic Classification

- Establishing a probabilistic model for classification
  - Generative model

$$P(\mathbf{X}\mid C) \quad C = c_1,\cdots,c_L,\ \mathbf{X} = (X_1,\cdots,X_n)$$



Maximum A Posterior

- **Based on Bayes Theorem, we can compute the *Maximum A Posterior* (MAP) hypothesis/parameters for the data**
- **We are interested in the best hypothesis for some space H given observed training data D.**

$$
\begin{aligned}
h_{MAP} &\equiv \underset{h \in H}{\operatorname{argmax}}\, P(h \mid D) \\
&= \underset{h \in H}{\operatorname{argmax}} \frac{P(D \mid h) P(h)}{P(D)} \\
&= \underset{h \in H}{\operatorname{argmax}}\, P(D \mid h) P(h)
\end{aligned}
$$

H: set of all hypothesis.

Note that we can drop *P(D)* as the probability of the data is constant (and independent of the hypothesis).

Bayes Classifiers

- **Assumption: training set consists of instances of different classes described *cj* as conjunctions of attributes values**
- **Task: Classify a new instance *d* based on a tuple of attribute values into one of the classes *cj* ∈ *C***
- **Key idea: assign the most probable class c_{MAP} using Bayes Theorem.**

$$
\begin{aligned}
c_{MAP} &= \underset{c_j \in C}{\operatorname{argmax}} P(c_j \mid x_1, x_2, \ldots, x_n) \\
&= \underset{c_j \in C}{\operatorname{argmax}} \frac{P(x_1, x_2, \ldots, x_n \mid c_j) P(c_j)}{P(x_1, x_2, \ldots, x_n)} \\
&= \underset{c_j \in C}{\operatorname{argmax}} P(x_1, x_2, \ldots, x_n \mid c_j) P(c_j)
\end{aligned}
$$

Parameters estimation

- $P(c_j)$
  - Can be estimated from the frequency of classes in the training examples.
- $P(x_1,x_2,\ldots,x_n|c_j)$
  - $O(|X|^n \bullet |C|)$ parameters
  - Could only be estimated if a very, very large number of training examples was available.
- **Independence Assumption: attribute values are conditionally independent given the target value: *naïve Bayes*.**

$$P(x_1, x_2, \ldots, x_n \mid c_j) = \prod_i P(x_i \mid c_j)$$

$$c_{NB} = \operatorname*{arg\,max}_{c_j \in C} P(c_j) \prod_i P(x_i \mid c_j)$$

Properties

- **Estimating $P(x_i \mid c_j)$ instead of $P(x_1, x_2, \ldots, x_n \mid c_j)$ greatly reduces the number of parameters (and the data sparseness).**
- **The learning step in Naïve Bayes consists of estimating**

  $P(x_i \mid c_j)$ and $P(c_j)$ based on the frequencies in the training data
- **An unseen instance is classified by computing the class that maximizes the posterior**
- **When conditioned independence is satisfied, Naïve Bayes corresponds to MAP classification.**

Example

北京航空航天大學 BEIHANG UNIVERSITY

- Example: Play Tennis

PlayTennis: training examples

Day	Outlook	Temperature	Humidity	Wind	PlayTennis
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	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	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

Example

- Learning Phase

Outlook	Play=*Yes*	Play=*No*
Sunny	2/9	3/5
Overcast	4/9	0/5
Rain	3/9	2/5

Temperature	Play=*Yes*	Play=*No*
Hot	2/9	2/5
Mild	4/9	2/5
Cool	3/9	1/5

Humidity	Play=*Yes*	Play=No
High	3/9	4/5
Normal	6/9	1/5

Wind	Play=*Yes*	Play=*No*
Strong	3/9	3/5
Weak	6/9	2/5

$P(\text{Play}=Yes) = 9/14 \qquad P(\text{Play}=No) = 5/14$

Example

北京航空航天大學 BEIHANG UNIVERSITY

- Test Phase
  - Given a new instance, predict its label
  - x'=(Outlook=Sunny, Temperature=Cool, Humidity=High, Wind=Strong)
  - Look up tables achieved in the learning phrase

$P(\text{Outlook}=\textit{Sunny} \mid \text{Play}=\textit{Yes}) = 2/9$	$P(\text{Outlook}=\textit{Sunny} \mid \text{Play}=\textit{No}) = 3/5$
$P(\text{Temperature}=\textit{Cool} \mid \text{Play}=\textit{Yes}) = 3/9$	$P(\text{Temperature}=\textit{Cool} \mid \text{Play}==\textit{No}) = 1/5$
$P(\text{Huminity}=\textit{High} \mid \text{Play}=\textit{Yes}) = 3/9$	$P(\text{Huminity}=\textit{High} \mid \text{Play}=\textit{No}) = 4/5$
$P(\text{Wind}=\textit{Strong} \mid \text{Play}=\textit{Yes}) = 3/9$	$P(\text{Wind}=\textit{Strong} \mid \text{Play}=\textit{No}) = 3/5$
$P(\text{Play}=\textit{Yes}) = 9/14$	$P(\text{Play}=\textit{No}) = 5/14$

  - Decision making with the MAP rule

$$P(\textit{Yes} \mid \mathbf{x}') \approx [P(\textit{Sunny} \mid \text{Yes})P(\textit{Cool} \mid \textit{Yes})P(\textit{High} \mid \text{Yes})P(\textit{Strong} \mid \textit{Yes})]P(\text{Play}=\textit{Yes}) = 0.0053$$

$$P(\textit{No} \mid \mathbf{x}') \approx [P(\textit{Sunny} \mid \text{No})\, P(\textit{Cool} \mid \text{No})P(\textit{High} \mid \textit{No})P(\textit{Strong} \mid \textit{No})]P(\text{Play}=\textit{No}) = 0.0206$$

Given the fact $P(\textit{Yes} \mid \mathbf{x}') < P(\textit{No} \mid \mathbf{x}')$, we label $\mathbf{x}'$ to be "*No*".

- Algorithm: Continuous-valued Features
  - Numberless values for a feature
  - Conditional probability often modeled with the normal distribution

$$\hat{P}(X_j \mid C = c_i) = \frac{1}{\sqrt{2\pi}\sigma_{ji}} \exp\left(-\frac{(X_j - \mu_{ji})^2}{2\sigma_{ji}^2} \right)$$

μ_{ji} : mean (avearage) of feature values X_j of examples for which $C = c_i$

σ_{ji} : standard deviation of feature values X_j of examples for which $C = c_i$

  - Learning Phase: for $\mathbf{X} = (X_1, \cdots, X_n),\ C = c_1, \cdots, c_L$
  - Output: $n \times L$ normal distributions and $P(C = c_i)\ i = 1, \cdots, L$
  - Test Phase: Given an unknown instance $\mathbf{X}' = (a'_1, \cdots, a'_n)$
  - Instead of looking-up tables, calculate conditional probabilities with all the normal distributions achieved in the learning phrase Apply the MAP rule to make a decision

- Example: Continuous-valued Features
  - Temperature is naturally of continuous value.
  - Yes: 25.2, 19.3, 18.5, 21.7, 20.1, 24.3, 22.8, 23.1, 19.8
  - No: 27.3, 30.1, 17.4, 29.5, 15.1
  - Estimate mean and variance for each class

$$\mu = \frac{1}{N}\sum_{n=1}^{N} x_n, \quad \sigma^2 = \frac{1}{N}\sum_{n=1}^{N}(x_n - \mu)^2$$

$$\mu_{Yes} = 21.64, \ \sigma_{Yes} = 2.35$$

$$\mu_{No} = 23.88, \ \sigma_{No} = 7.09$$

  - Learning Phase: output two Gaussian models for P(temp|C)

$$\hat{P}(x \mid Yes) = \frac{1}{2.35\sqrt{2\pi}} \exp\left(-\frac{(x-21.64)^2}{2\times 2.35^2}\right) = \frac{1}{2.35\sqrt{2\pi}} \exp\left(-\frac{(x-21.64)^2}{11.09}\right)$$

$$\hat{P}(x \mid No) = \frac{1}{7.09\sqrt{2\pi}} \exp\left(-\frac{(x-23.88)^2}{2\times 7.09^2}\right) = \frac{1}{7.09\sqrt{2\pi}} \exp\left(-\frac{(x-23.88)^2}{50.25}\right)$$

Relevant Issues

北京航空航天大學 BEIHANG UNIVERSITY

- Violation of Independence Assumption
  - For many real world tasks, $P(X_1,\cdots,X_n|C) \neq P(X_1|C)\cdots P(X_n|C)$
  - Nevertheless, naïve Bayes works surprisingly well anyway!
- Zero conditional probability Problem
  - If no example contains the feature value $X_j = a_{jk}$, $\hat{P}(X_j = a_{jk}|C = c_i) = 0$
  - In this circumstance, $\hat{P}(x_1|c_i)\cdots\hat{P}(a_{jk}|c_i)\cdots\hat{P}(x_n|c_i) = 0$ during test
  - For a remedy, conditional probabilities re-estimated with

$$\hat{P}(X_j = a_{jk}|C = c_i) = \frac{n_c + mp}{n + m}$$

n_c : number of training examples for which $X_j = a_{jk}$ and $C = c_i$

n : number of training examples for which $C = c_i$

p : prior estimate (usually, $p = 1/t$ for t possible values of X_j)

m : weight to prior (number of "virtual" examples, $m \geq 1$)

Summary of Naïve Bayes

- **Naïve Bayes: the conditional independence assumption**
  - Training is very easy and fast; just requiring considering each attribute in each class separately
  - Test is straightforward; just looking up tables or calculating conditional probabilities with estimated distributions
- **A popular generative model**
  - Performance competitive to most of state-of-the-art classifiers even in presence of violating independence assumption
  - Many successful applications, e.g., spam mail filtering
  - A good candidate of a base learner in ensemble learning
  - Apart from classification, naïve Bayes can do more…

Bayesian Belief Network

- **What if the conditional independence condition does not hold?**
- **Directed Acyclic Graph (DAG)**
- **Conditional Probability distribution on the edge**
- **Conditional Independence (d-separation)**
  - A node in BN is conditionally independent of its non-descendants, if its parents are known/observed

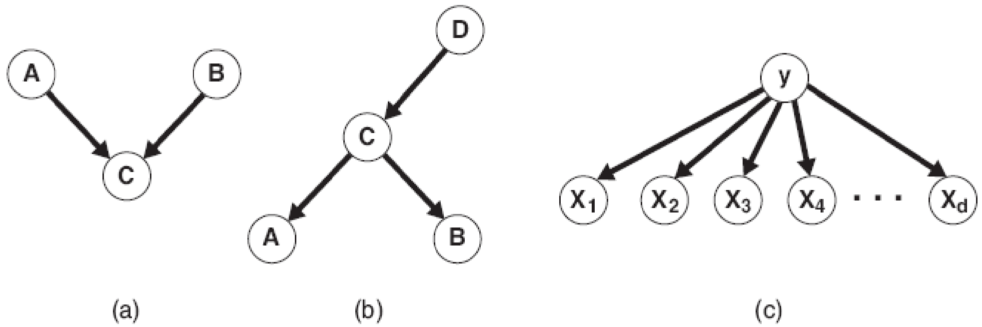


Figure 5.12. Representing probabilistic relationships using directed acyclic graphs.

Example of BBN

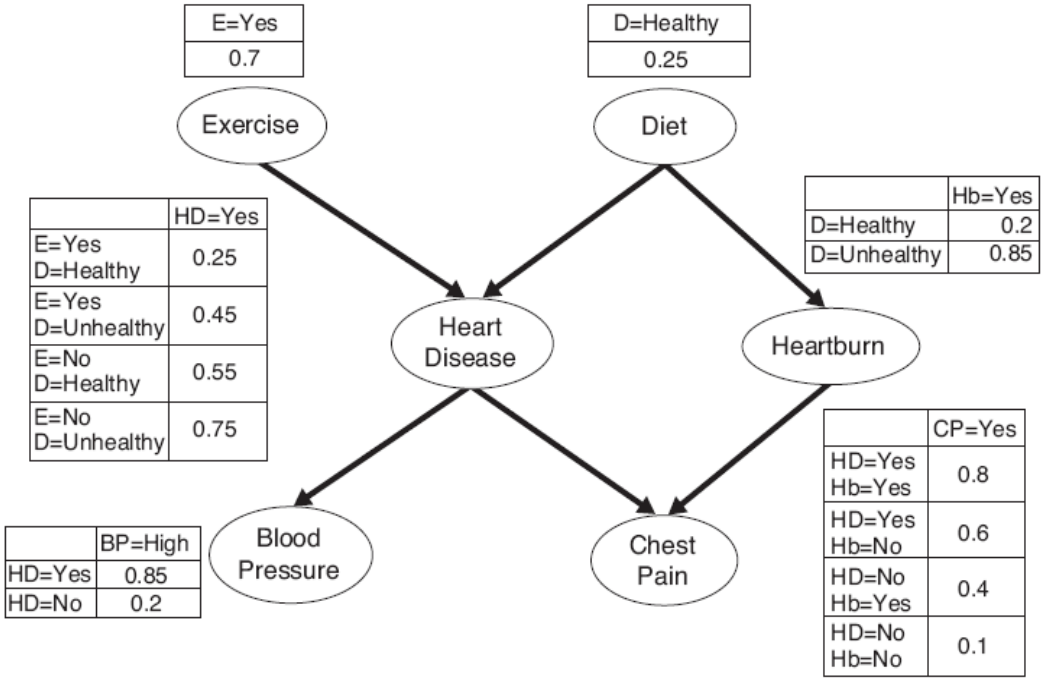


Figure 5.13. A Bayesian belief network for detecting heart disease and heartburn in patients.

Characteristics of BBN

北京航空航天大學 BEIHANG UNIVERSITY

- **Prior knowledge**
- **Graphical model**
  - Dependencies among variables
- **Robust to overfitting**

THANK YOU

经管学院 刘冠男