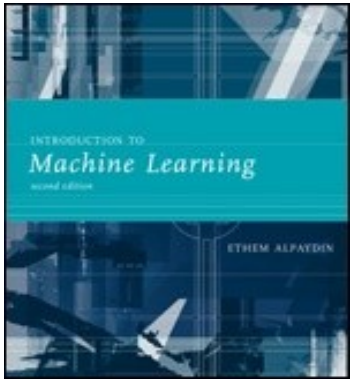
Lecture Slides for

INTRODUCTION TO

Machine Learning

2nd Edition



ETHEM ALPAYDIN, modified by Leonardo Bobadilla and some parts from http://www.cs.tau.ac.il/~apartzin/MachineLearning/ and www.cs.princeton.edu/courses/archive/fall01/cs302/notes/11.../EM.ppt

 alpaydin@boun.edu.tr
http://www.cmpe.boun.edu.tr/~ethem/i2m

Outline

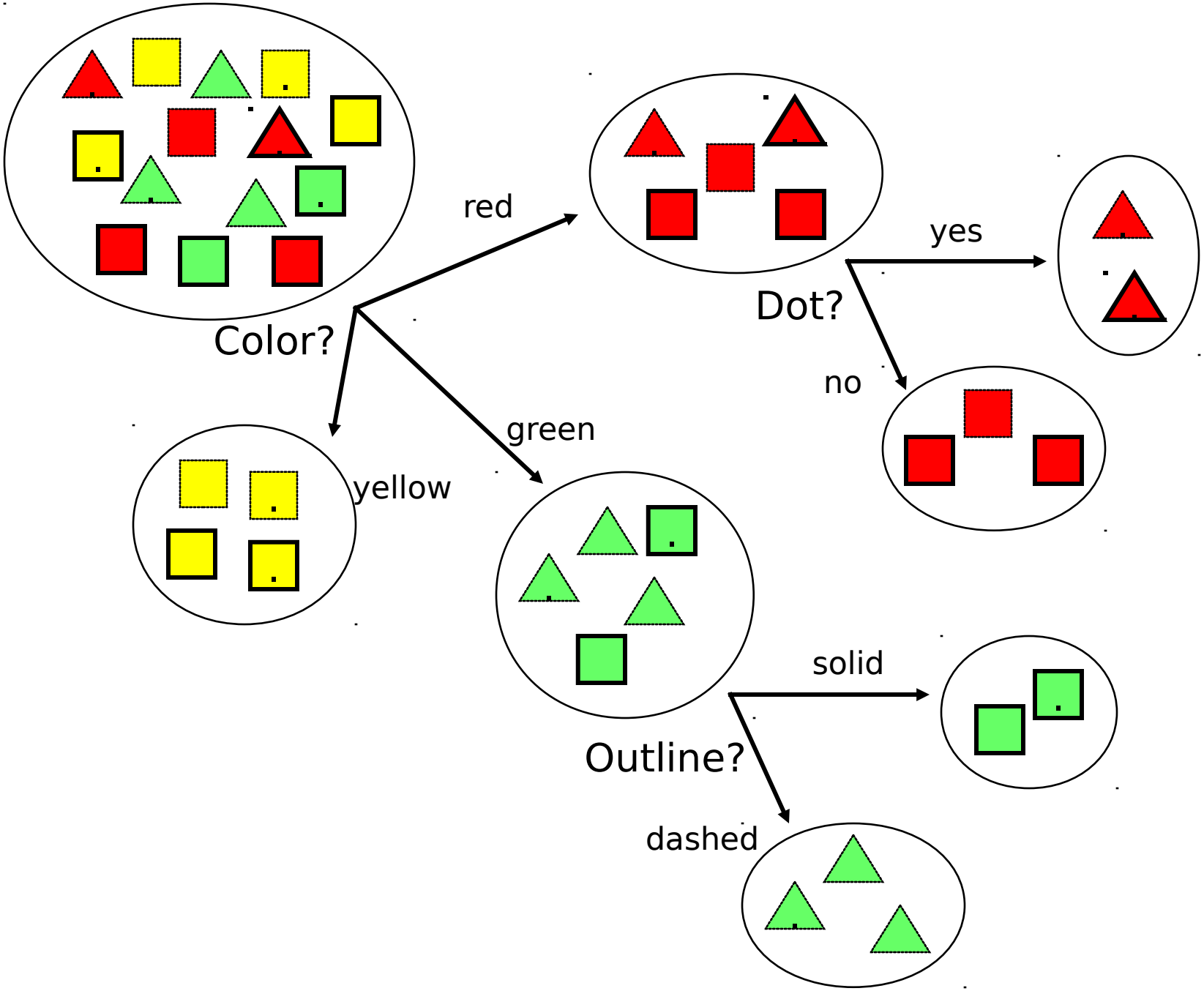
Previous class
Ch 8: Decision Trees
This class:
Ch 9: Decision Trees

CHAPTER 9:

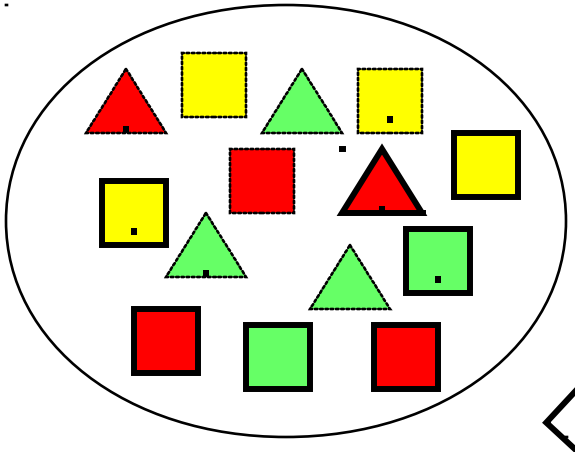
Decision Trees

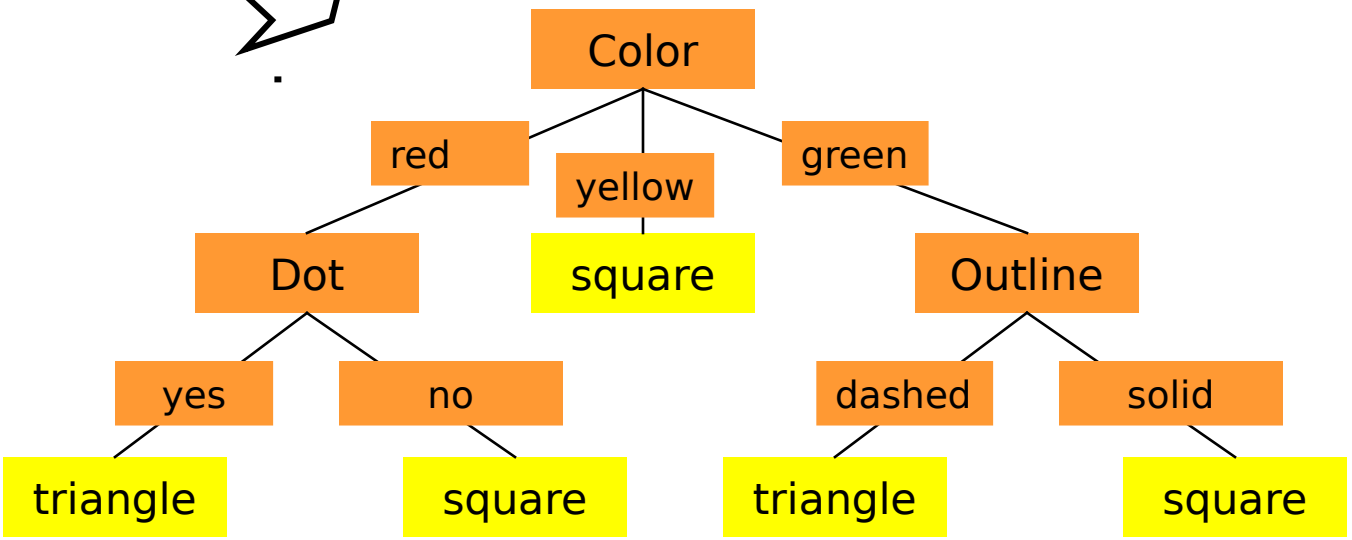
Slides from: Blaž Zupan and Ivan Bratko

magix.fri.uni-lj.si/predavanja/uisp


Color?
red
yellow
green
Dot?
yes
no
Outline?
solid
dashed

Decision Tree





Classification Trees

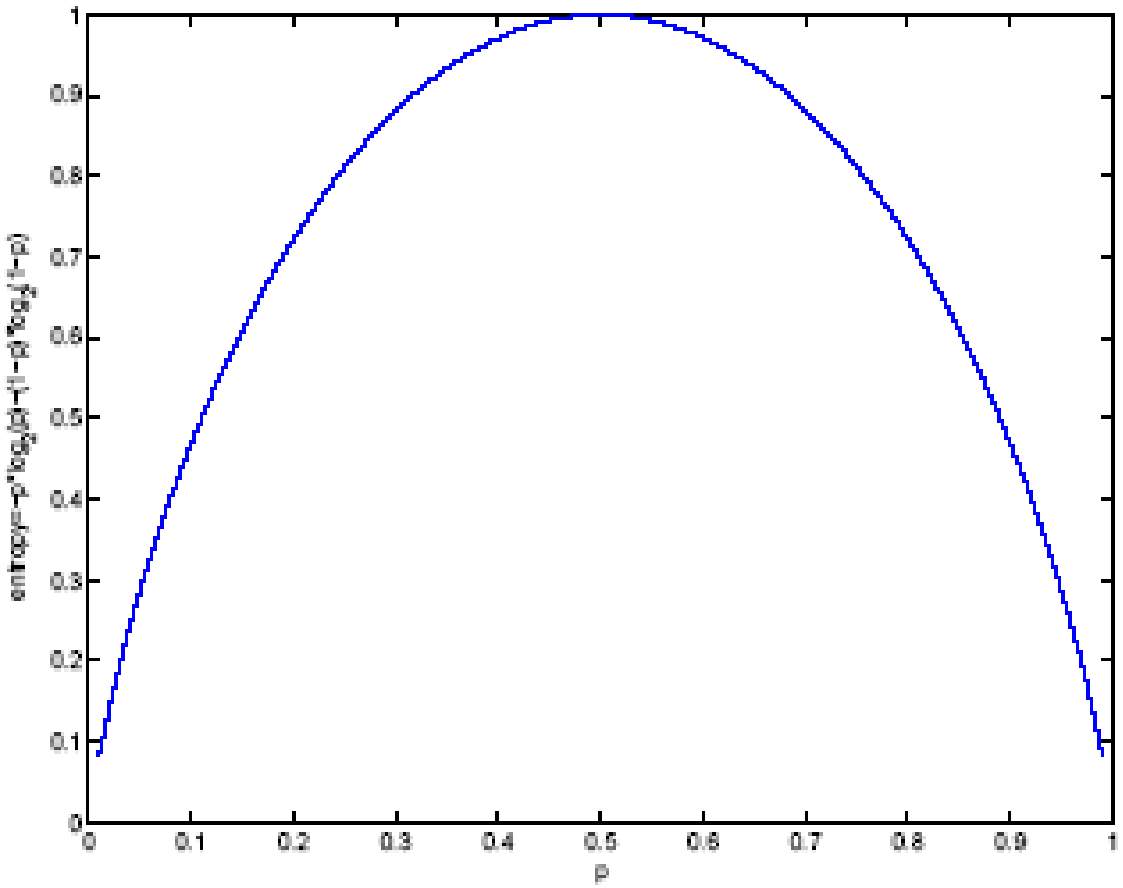
- What is the good split function?
- Use Impurity measure
- Assume N_m training samples reach node m
- N_m^i of N_m belong to class C_i, with $\sum_i N_m^i = N_m$.
- $$\hat{P}(C_i|\mathbf{x}, m) \equiv p_m^i = \frac{N_m^i}{N_m}$$
- Node m is pure if for all classes either 0 or 1

Entropy

- Measure amount of uncertainty on a scale from 0 to 1
- Example: 2 events
- If $p_1=p_2=0.5$, entropy is 1 which is maximum uncertainty
- If $p_2=1=1-p_1$, entropy is 0 , which is no uncertainty (Eq 9.3)

$$\mathcal{I}_m = -\sum_{i=1}^{K} p_m^i \log_2 p_m^i$$

Entropy



Best Split

- Node is impure, need to split more
- Have several split criteria (coordinates), have to choose optimal
- Minimize impurity (uncertainty) after split
- Stop when impurity is small enough
  - Zero stop impurity=>complex tree with large variance
  - Larger stop impurity=>small tress but large bias

Best Split

- Impurity after split: N_{mj} of N_m take branch j.
- N^i_{mj} belong to C_i

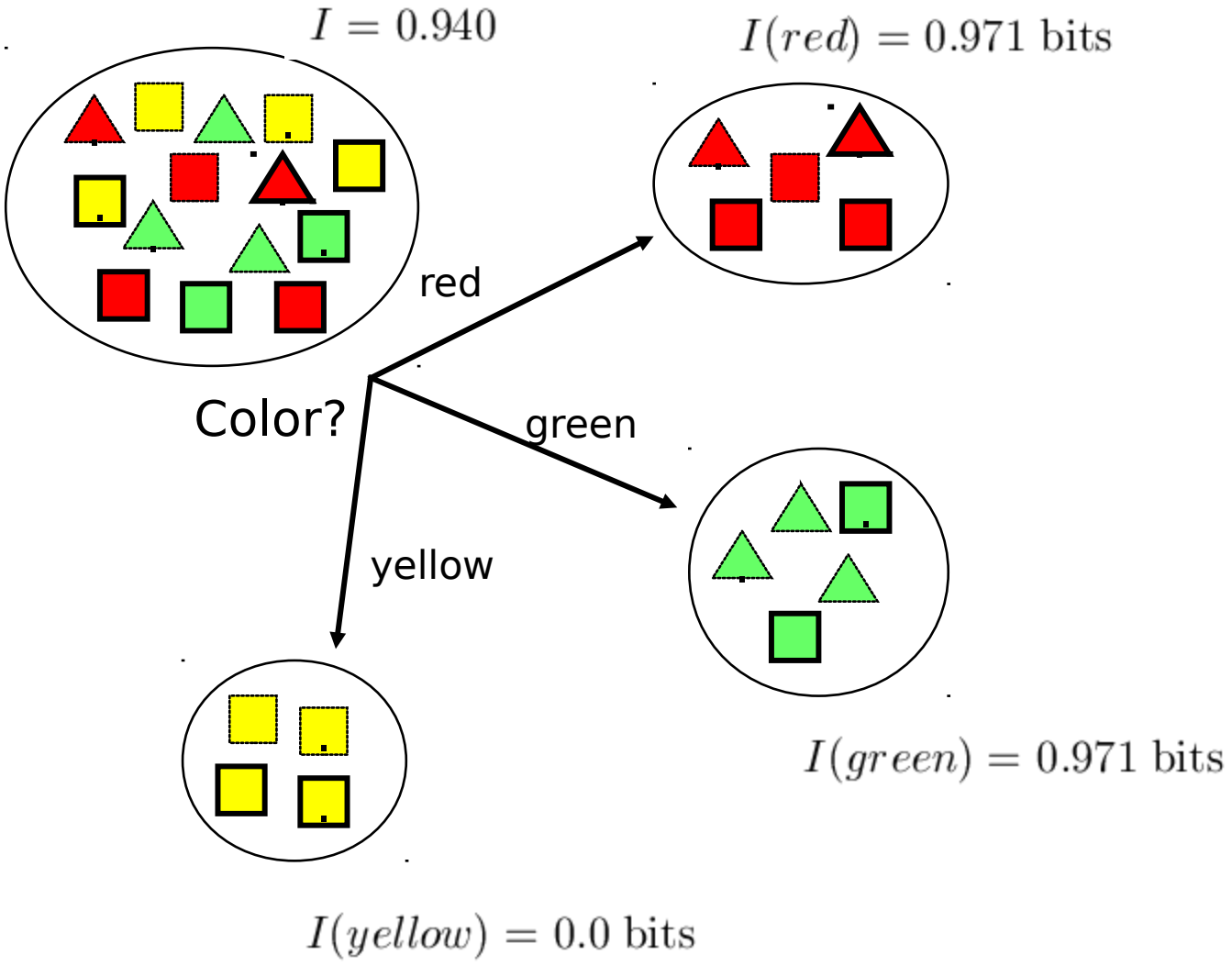
$$\hat{P}(C_i \mid \boldsymbol{x}, m, j) \equiv p^i_{mj} = \frac{N^i_{mj}}{N_{mj}}$$

  - (Eq 9.8)

$$I'_m = -\sum_{j=1}^{n} \frac{N_{mj}}{N_m} \sum_{i=1}^{K} p^i_{mj} \log_2 p^i_{mj}$$

- Find the variable and split that min impurity
  - among all variables
  - split positions for numeric variables

Information Gain



$$Gain(\text{Color}) = I - I_{res}(\text{Color}) = 0.940 - 0.694 = 0.246\ bits$$

```
GenerateTree(X)
    If NodeEntropy(X)< θ_I /* eq. 9.3
        Create leaf labelled by majority class in X
        Return
    i ← SplitAttribute(X)
    For each branch of x_i
        Find X_i falling in branch
        GenerateTree(X_i)
SplitAttribute(X)
    MinEnt← MAX
    For all attributes i = 1,...,d
        If x_i is discrete with n values
            Split X into X_1,...,X_n by x_i
            e ← SplitEntropy(X_1,...,X_n) /* eq. 9.8 */
            If e<MinEnt MinEnt ← e; bestf ← i
        Else /* x_i is numeric */
            For all possible splits
                Split X into X_1, X_2 on x_i
                e←SplitEntropy(X_1, X_2)
                If e<MinEnt MinEnt ← e; bestf ← i
    Return bestf
```

Regression Trees

- Value not a label in a leaf nodes
- Need other impurity measure
- Use Average Error

$$b_m(x)=\begin{cases}1 & \text{if } x\in X_m : x \text{ reaches node } m\\ 0 & \text{otherwise}\end{cases}$$

$$E_m=\frac{1}{N_m}\sum_t\left(r^t-g_m\right)^2 b_m\left(x^t\right) \qquad g_m=\frac{\sum_t b_m\left(x^t\right)r^t}{\sum_t b_m\left(x^t\right)}$$

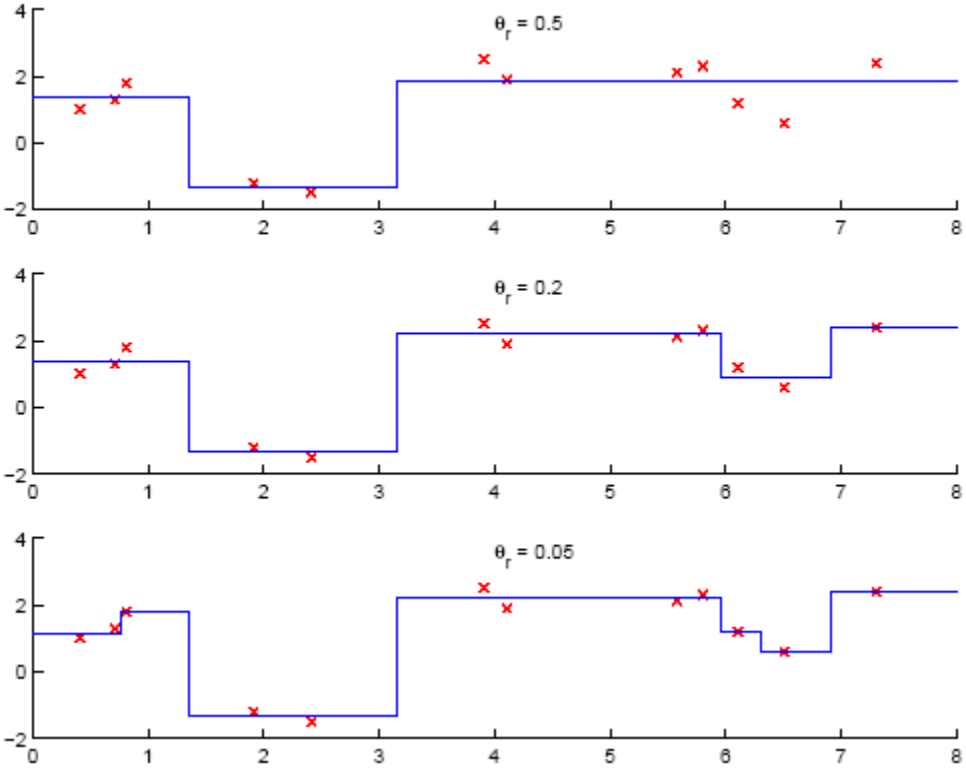
Regression Trees

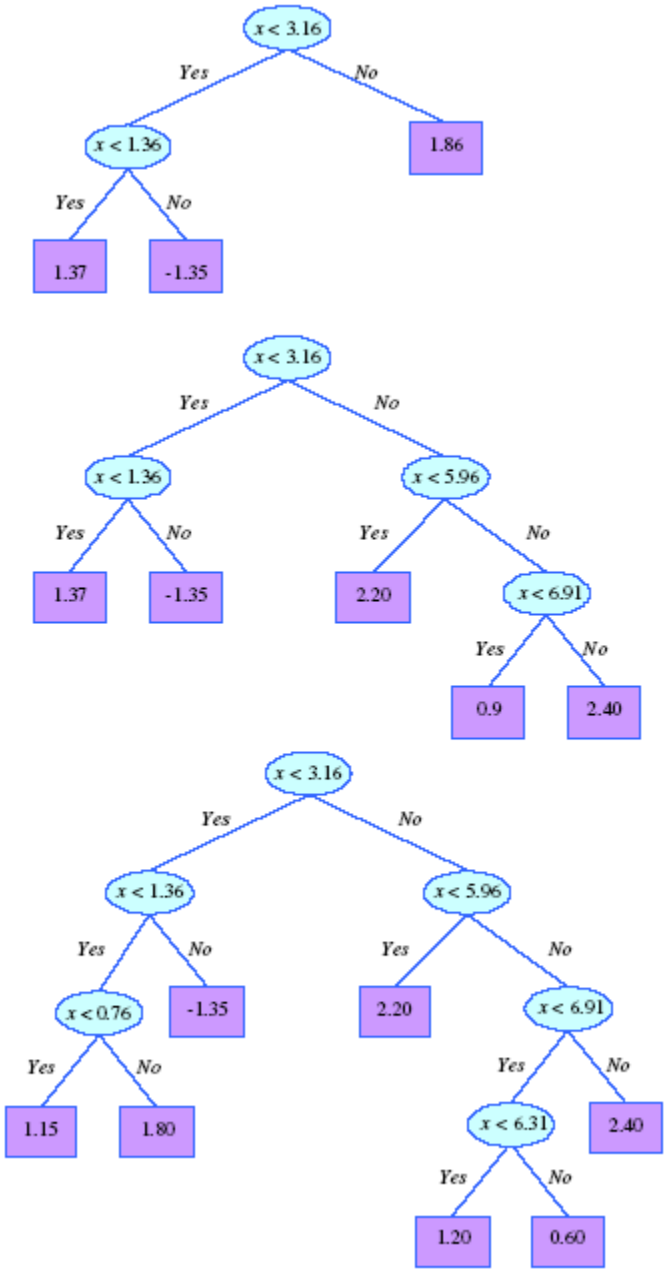
- After splitting:

$$b_{mj}\left(\mathbf{x}\right)=\begin{cases}1 & \text{if } \mathbf{x}\in \mathsf{X}_{mj}: \mathbf{x} \text{ reaches node } m \text{ and branch } j\\ 0 & \text{otherwise}\end{cases}$$

$$E'_m=\frac{1}{N_m}\sum_j\sum_t\left(r^t-g_{mj}\right)^2 b_{mj}\left(\mathbf{x}^t\right) \qquad g_{mj}=\frac{\sum_t b_{mj}\left(\mathbf{x}^t\right) r^t}{\sum_t b_{mj}\left(\mathbf{x}^t\right)}$$

Example





Pruning Trees

- Number of data instances reach a node is small
  - Less then 5% of training data
  - Don’t want to split further regardless of impurity
- Remove subtrees for better generalization
  - Prepruning: Early stopping
  - Postpruning: Grow the whole tree then prune subtrees
    - Set aside pruning set
    - Make sure pruning does not significantly increase error

Decision Trees and Feature Extraction

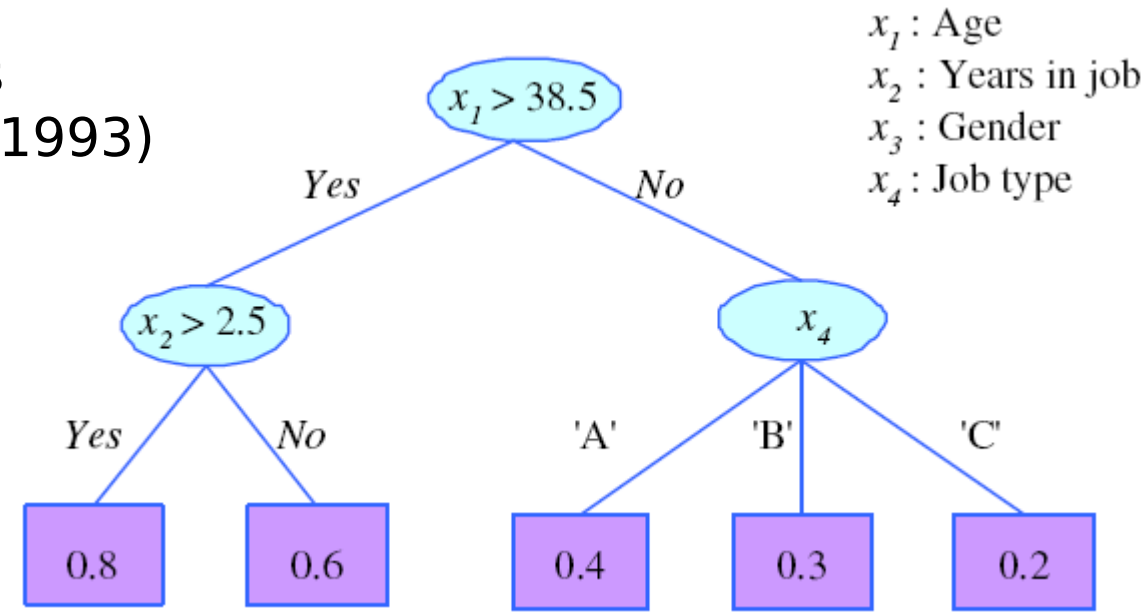
- Univariate Tree uses only certain variable
- Some variables might not get used
- Features closer to the root have greater importance

Interpretability

- Conditions that are simple to understand
- Path from the root =>one conjunction of test
- All paths can be defined using set of IF_THEN rules
  - Form a rule base
- Percentage of training data covered by the rule
  - Rule support
- Tool for a Knowledge Extraction
- Can be verified by experts

Rule Extraction from Trees

C4.5Rules
(Quinlan, 1993)



R1: IF (age>38.5) AND (years-in-job>2.5) THEN y =0.8
R2: IF (age>38.5) AND (years-in-job≤2.5) THEN y =0.6
R3: IF (age≤38.5) AND (job-type='A') THEN y =0.4
R4: IF (age≤38.5) AND (job-type='B') THEN y =0.3
R5: IF (age≤38.5) AND (job-type='C') THEN y =0.2

Learning Rules

- Rule induction is similar to tree induction but
  - tree induction is breadth-first,
  - rule induction is depth-first; one rule at a time

- Rule set contains rules; rules are conjunctions of terms
  - A rule covers an example if all terms of the rule evaluate to true for the example.
  - A rule is said to cover an example if the example satisfies all the conditions of the rule.

- Sequential covering:
  - Generate rules one at a time until all positive examples are covered

Rule-Based Classifier

- Classify records by using a collection of “if… then…” rules

- Rule: (*Condition*) $\rightarrow y$
  - where
    - *Condition* is a conjunctions of attributes
    - *y* is the class label
  - Examples of classification rules:
    - (Blood Type=Warm) $\wedge$ (Lay Eggs=Yes) $\rightarrow$ Birds
    - (Taxable Income < 50K) $\wedge$ (Refund=Yes) $\rightarrow$ Evade=No

Rule-based Classifier (Example)

Name	Blood Type	Give Birth	Can Fly	Live in Water	Class
human	warm	yes	no	no	mammals
python	cold	no	no	no	reptiles
salmon	cold	no	no	yes	fishes
whale	warm	yes	no	yes	mammals
frog	cold	no	no	sometimes	amphibians
komodo	cold	no	no	no	reptiles
bat	warm	yes	yes	no	mammals
pigeon	warm	no	yes	no	birds
cat	warm	yes	no	no	mammals
leopard shark	cold	yes	no	yes	fishes
turtle	cold	no	no	sometimes	reptiles
penguin	warm	no	no	sometimes	birds
porcupine	warm	yes	no	no	mammals
eel	cold	no	no	yes	fishes
salamander	cold	no	no	sometimes	amphibians
gila monster	cold	no	no	no	reptiles
platypus	warm	no	no	no	mammals
owl	warm	no	yes	no	birds
dolphin	warm	yes	no	yes	mammals
eagle	warm	no	yes	no	birds

R1: (Give Birth = no) $\wedge$ (Can Fly = yes) $\rightarrow$ Birds

R2: (Give Birth = no) $\wedge$ (Live in Water = yes) $\rightarrow$ Fishes

R3: (Give Birth = yes) $\wedge$ (Blood Type = warm) $\rightarrow$ Mammals

R4: (Give Birth = no) $\wedge$ (Can Fly = no) $\rightarrow$ Reptiles

R5: (Live in Water = sometimes) $\rightarrow$ Amphibians

From: Luke Huan 2006

Application of Rule-Based Classifier

- A rule is said to cover an example if the example satisfies all the conditions of the rule.

R1: (Give Birth = no) $\wedge$ (Can Fly = yes) $\rightarrow$ Birds

R2: (Give Birth = no) $\wedge$ (Live in Water = yes) $\rightarrow$ Fishes

R3: (Give Birth = yes) $\wedge$ (Blood Type = warm) $\rightarrow$ Mammals

R4: (Give Birth = no) $\wedge$ (Can Fly = no) $\rightarrow$ Reptiles

R5: (Live in Water = sometimes) $\rightarrow$ Amphibians

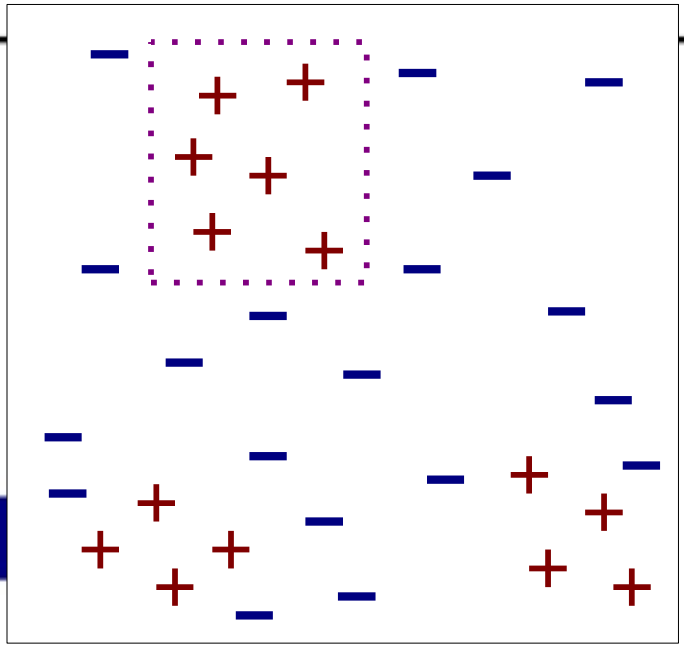
Name	Blood Type	Give Birth	Can Fly	Live in Water	Class
hawk	warm	no	yes	no	?
grizzly bear	warm	yes	no	no	?

The rule R1 covers a hawk => Bird

The rule R3 covers the grizzly bear => Mammal

From: Luke Huan 2006

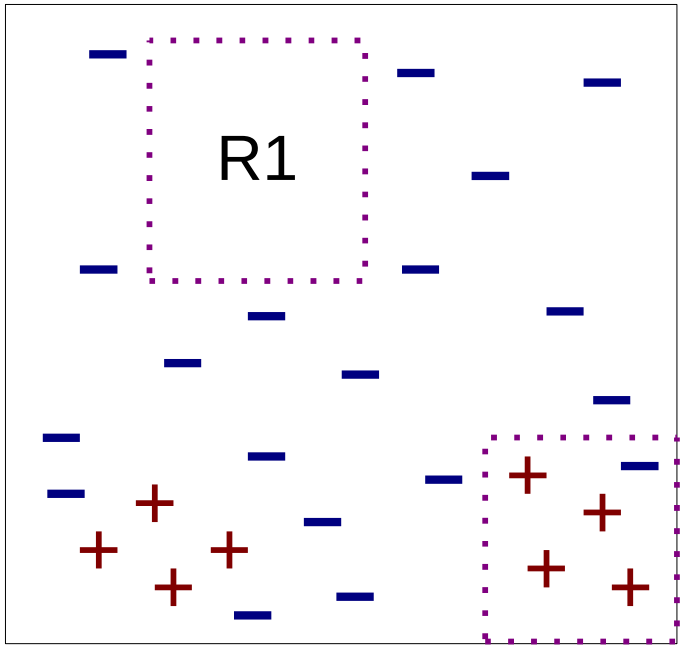
Example of Sequential Covering



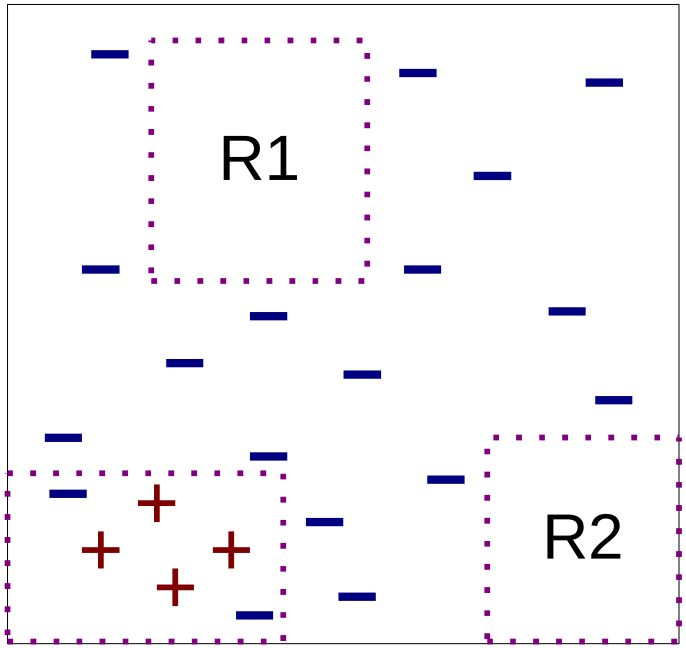
(ii) Step 1

From: Luke Huan 2006

Example of Sequential Covering…



(iii) Step 2



(iv) Step 3

From: Luke Huan 2006

Ripper Algorithm

- There are two kinds of loop in Ripper algorithm (Cohen, 1995):
  - Outer loop : adding one rule at a time to the rule base
  - Inner loop : adding one condition at a time to the current rule
    - Conditions are added to the rule to maximize an information gain measure.
    - Conditions are added to the rule until it covers no negative example.

Ripper Algorithm

- In Ripper, conditions are added to the rule to
  - Maximize an information gain measure

$$Gain(R', R) = s \cdot (\log_2 \frac{N'_+}{N'} - \log_2 \frac{N_+}{N})$$

    - R : the original rule
    - R' : the candidate rule after adding a condition
    - N (N'): the number of instances that are covered by R (R')
    - N_+ (N'_+): the number of true positives in R (R')
    - s : the number of true positives in R and R' (after adding the condition)
  - Until it covers no negative example

Stopping Criterion and Rule Pruning

- Stopping criterion
  - Compute the gain
  - If gain is not significant, discard the new rule

- Rule Pruning
  - Similar to post-pruning of decision trees
  - Reduced Error Pruning:
    - Remove one of the conjuncts in the rule
    - Compare error rate on validation set before and after pruning
    - If error improves, prune the conjunct

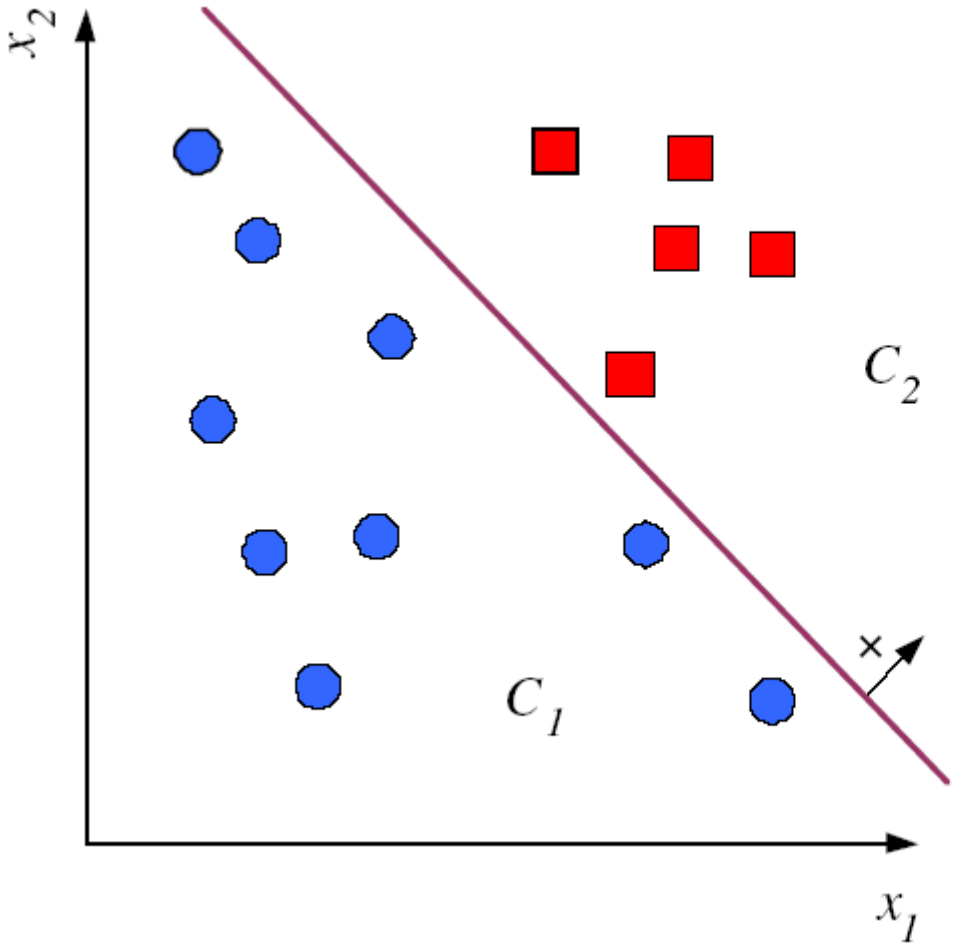
Summary of Direct Method

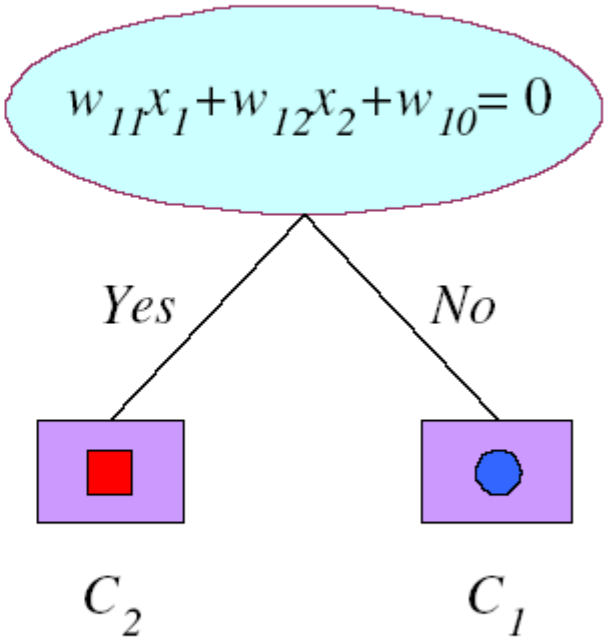
- Grow a single rule
- Remove Instances from rule
- Prune the rule (if necessary)
- Add rule to Current Rule Set
- Repeat

Direct Method: RIPPER

- For 2-class problem, choose one of the classes as positive class, and the other as negative class
  - Learn rules for positive class
  - Negative class will be default class
- For multi-class problem
  - Order the classes according to increasing class prevalence (fraction of instances that belong to a particular class)
  - Learn the rule set for smallest class first, treat the rest as negative class
  - Repeat with next smallest class as positive class

Multivariate Trees





Likelihood- vs. Discriminant-based Classification

- Likelihood-based: Assume a model for $p(\boldsymbol{x}|C_i)$, use Bayes' rule to calculate $P(C_i|\boldsymbol{x})$

$$g_i(\boldsymbol{x}) = \log P(C_i|\boldsymbol{x})$$

- Discriminant-based: Assume a model for $g_i(\boldsymbol{x}|\Phi_i)$; no density estimation
  - Estimating the boundaries is enough; no need to accurately estimate the densities inside the boundaries
  - Learning is the optimization of the model parameters Φ_i to maximize the classification accuracy on a given labeled training set.

Linear Discriminant

- Linear discriminant:

$$g_i(\boldsymbol{x}\,|\,\boldsymbol{w}_i, w_{i0}) = \boldsymbol{w}_i^T\boldsymbol{x} + w_{i0} = \sum_{j=1}^{d} w_{ij}x_j + w_{i0}$$

- Advantages:
  - Simple: O(d) space/computation
  - Easy to understand:
    - The final output is weighted sum of several factors.
  - Accurate:
    - When $p(\boldsymbol{x}|C_i)$ are Gaussian with a shared covariance matrix, the optimal discriminant is linear.
    - The linear discriminant can be used even when this assumption does not hold.

Generalized Linear Model

- Quadratic discriminant:

$$g_i(\boldsymbol{x} \mid \mathbf{W}_i, \boldsymbol{w}_i, w_{i0}) = \boldsymbol{x}^T \mathbf{W}_i \boldsymbol{x} + \boldsymbol{w}_i^T \boldsymbol{x} + w_{i0}$$

- Higher-order (product) terms:

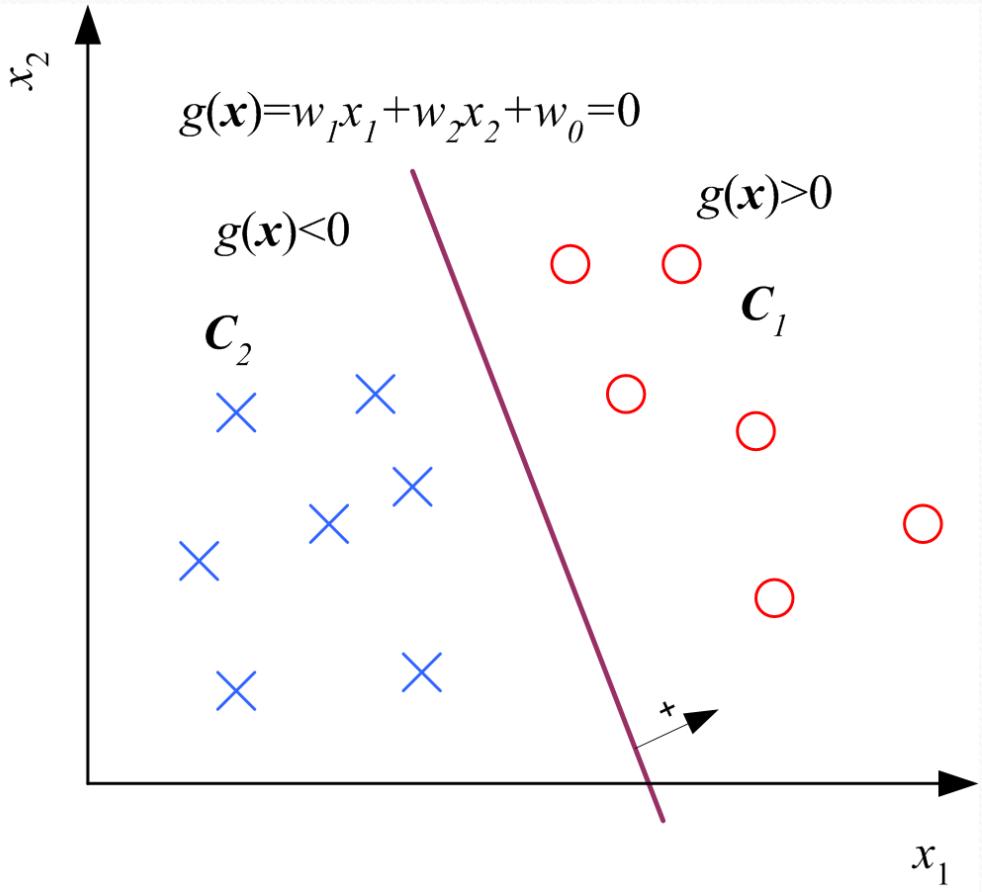
$$z_1 = x_1,\ z_2 = x_2,\ z_3 = x_1^2,\ z_4 = x_2^2,\ z_5 = x_1 x_2$$

Map from $\boldsymbol{x}$ to $\boldsymbol{z}$ using nonlinear basis functions and use a linear discriminant in $\boldsymbol{z}$-space

$$g_i(\boldsymbol{x}) = \sum_{j=1}^{k} w_j \phi_{ij}(\boldsymbol{x})$$

$\phi_{ij}(\boldsymbol{x})$: basis functions

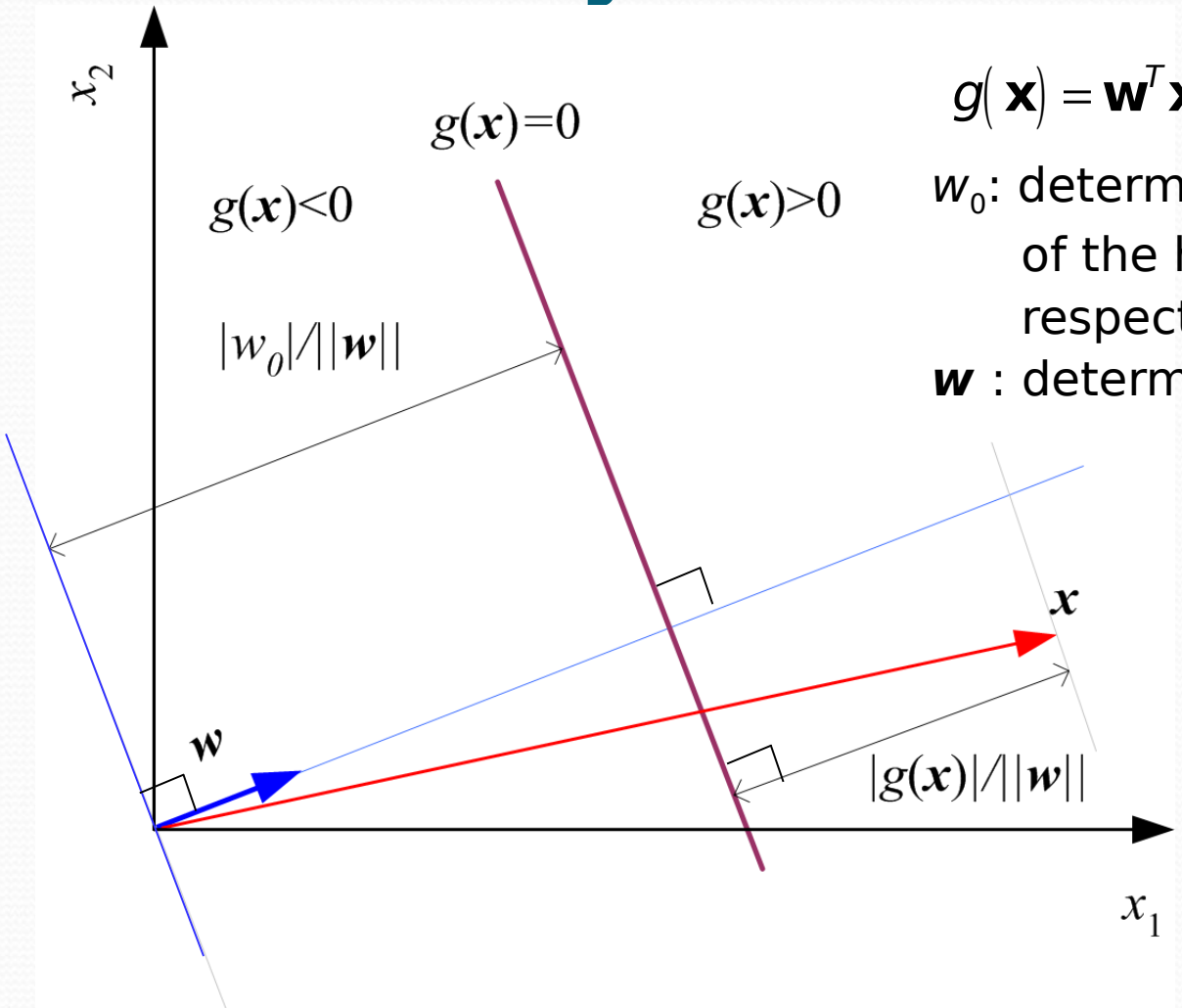
Two Classes



$$
\begin{aligned}
g(\boldsymbol{x}) &= g_1(\boldsymbol{x}) - g_2(\boldsymbol{x}) \\
&= \left(\boldsymbol{w}_1^T\boldsymbol{x} + w_{10}\right) - \left(\boldsymbol{w}_2^T\boldsymbol{x} + w_{20}\right) \\
&= \left(\boldsymbol{w}_1 - \boldsymbol{w}_2\right)^T\boldsymbol{x} + \left(w_{10} - w_{20}\right) \\
&= \boldsymbol{w}^T\boldsymbol{x} + w_0
\end{aligned}
$$

$$
\text{choose}\begin{cases} C_1 & \text{if } g(\boldsymbol{x}) > 0 \\ C_2 & \text{otherwise} \end{cases}
$$

Geometry

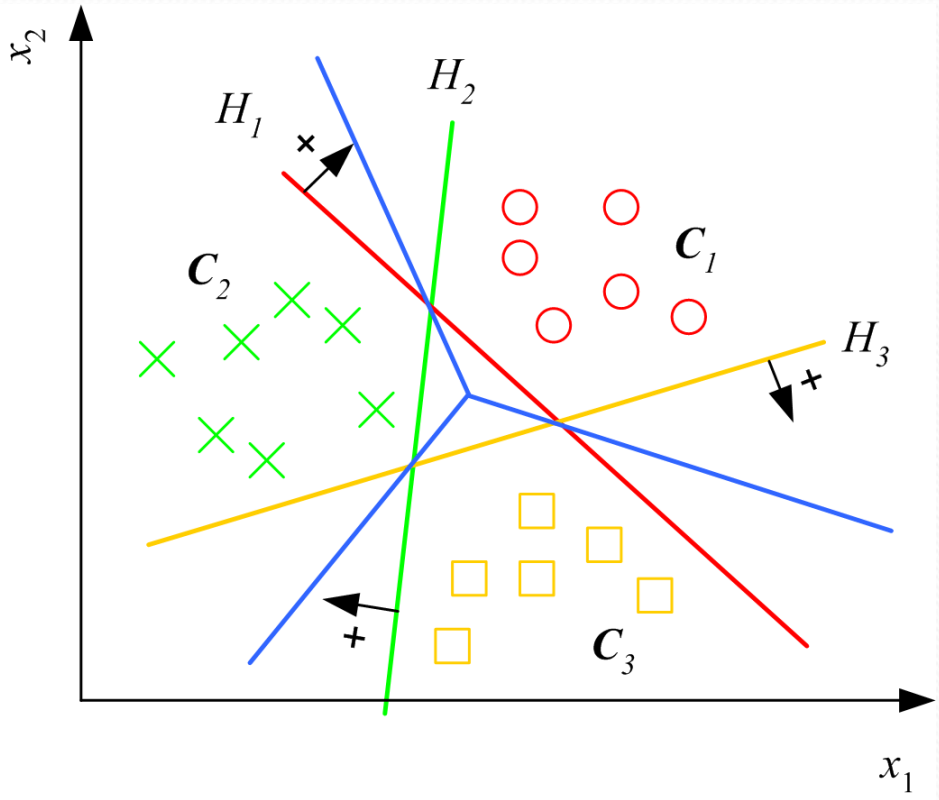


$$g(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + w_0$$

w_0: determines the location of the hyper-plane with respect to the origin

$\boldsymbol{w}$: determines its orientation

Multiple Classes



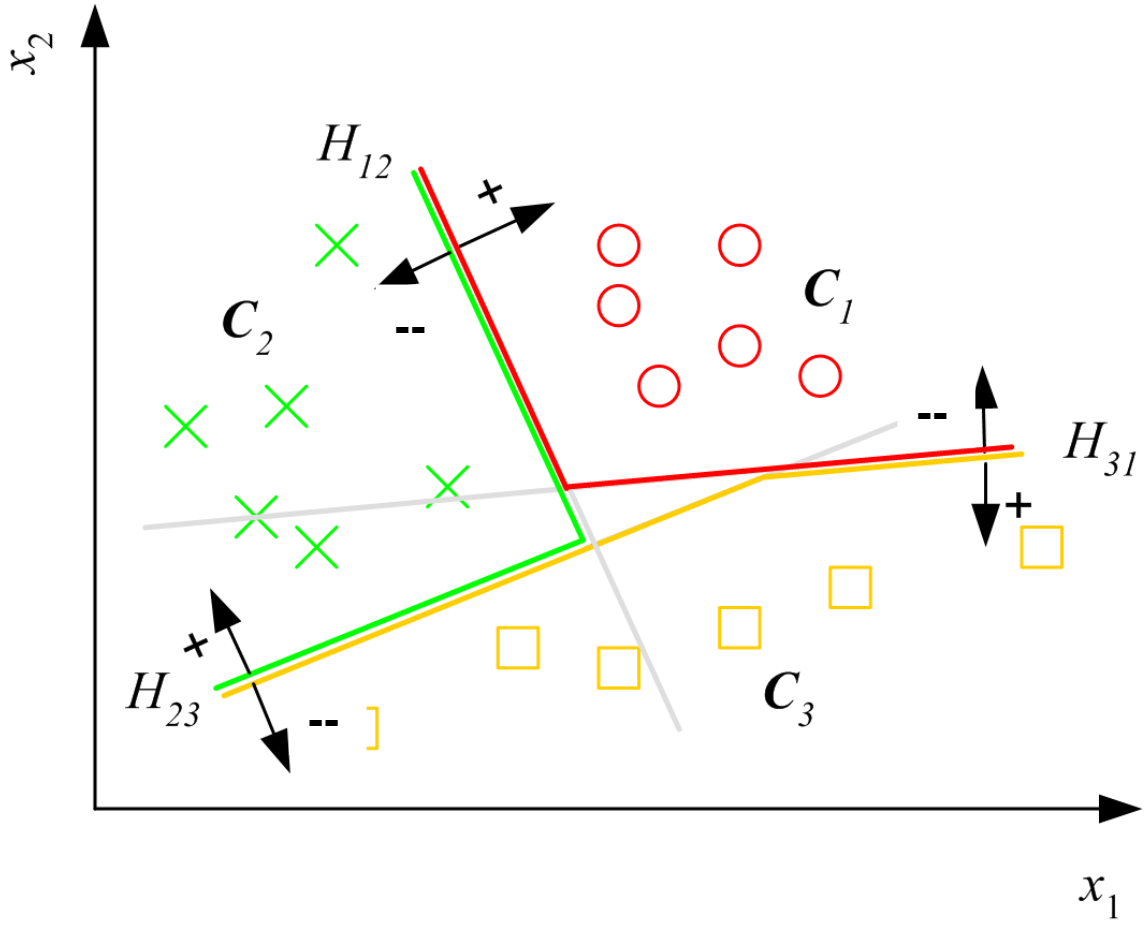
$$g_i(\mathbf{x} \mid \mathbf{w}_i, w_{i0}) = \mathbf{w}_i^T \mathbf{x} + w_{i0}$$

$$\text{Choose } C_i \text{ if } g_i(\mathbf{x}) = \max_{j=1}^{K} g_j(\mathbf{x})$$

H_i : the hyperplane separate the examples of C_i from the examples of all other classes

Classes are linearly separable

Pairwise Separation



$$g_{ij}\left(\boldsymbol{x}\,|\,\boldsymbol{w}_{ij}, w_{ij\,0}\right) = \boldsymbol{w}_{ij}^T\boldsymbol{x} + w_{ij\,0}$$

$$g_{ij}(\boldsymbol{x}) = \begin{cases} > 0 & \text{if } \boldsymbol{x} \in C_i \\ \leq 0 & \text{if } \boldsymbol{x} \in C_j \\ \text{don't care} & \text{otherwise} \end{cases}$$

choose C_i if $\forall j \neq i, g_{ij}(\mathbf{x}) > 0$

H_{ij} : the hyperplane separate
the examples of C_i and
the examples of C_j