Bayes Classifiers

Bayes Classifier

- A probabilistic framework for solving classification problems
- Conditional Probability:

$$P(C \mid X) = \frac{P(X, C)}{P(X)}$$

$$P(X \mid C) = \frac{P(X, C)}{P(C)}$$

- Bayes theorem:

Posterior probability

Prior probability

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

Towards Naïve Bayesian Classifier

- Let D be a training set of tuples and their associated class labels, and each tuple is represented by an n attribute vector **X** = (x_1, x_2, …, x_n)
- Suppose there are *m* classes C_1, C_2, …, C_m.
- Classification is to derive the maximum posteriori, i.e., the maximal P(C_i|**X**)
- This can be derived from Bayes' theorem

$$P(C_i|\mathbf{X}) = \frac{P(\mathbf{X}|C_i)P(C_i)}{P(\mathbf{X})}$$

- Since P(X) is constant for all classes, only

$$P(C_i|\mathbf{X}) = P(\mathbf{X}|C_i)P(C_i)$$

needs to be maximized

Derivation of Naïve Bayes Classifier

- A simplified assumption: attributes are conditionally independent (i.e., no dependence relation between attributes):

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

- This greatly reduces the computation cost: Only counts the class distribution
- If A_k is categorical: $P(x_k|C_i)$ is the # of tuples in C_i having value x_k for A_k divided by $|C_{i, D}|$ (# of tuples of C_i in D)
- If A_k is continous-valued: $P(x_k|C_i)$ is usually computed based on Gaussian distribution with a mean μ and standard deviation σ

$$g(x, \mu, \sigma) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

and $P(x_k|C_i)$ is

$$P(\mathbf{X}|C_i) = g(x_k, \mu_{C_i}, \sigma_{C_i})$$

How to Estimate Probabilities from Data?

Tid	Refund	Marital Status	Taxable Income	Evade
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

- Class: $P(C) = N_c/N$
  - e.g., P(No) = 7/10,
    P(Yes) = 3/10

- For discrete attributes:

  $$P(A_i \mid C_k) = |A_{ik}| / N_{c_k}$$

  - where $|A_{ik}|$ is number of instances having attribute A_i and belongs to class C_k
  - Examples:

    P(Status=Married|No) = 4/7
    P(Refund=Yes|Yes)=0

How to Estimate Probabilities from Data?

- For continuous attributes:
  - Discretization:
    - Replace the attribute with discrete intervals
  - Probability density estimation:
    - Assume attribute follows a normal distribution
    - Use data to estimate parameters of distribution (e.g., mean and standard deviation)
    - Once probability distribution is known, can use it to estimate the conditional probability $P(A_i|c)$

How to Estimate Probabilities from Data?

Tid	Refund	Marital Status	Taxable Income	Evade
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

- Normal distribution:

$$P(A_i \mid c_j) = \frac{1}{\sqrt{2\pi\sigma_{ij}^2}} e^{-\frac{(A_i - \mu_{ij})^2}{2\sigma_{ij}^2}}$$

  - One for each (A_i,c_j) pair

- For (Income, Class=No):
  - If Class=No
    - sample mean = 110
    - sample variance = 2975

$$P(Income = 120 \mid No) = \frac{1}{\sqrt{2\pi}(54.54)} e^{-\frac{(120-110)^2}{2(2975)}} = 0.0072$$

Naïve Bayesian Classifier: Training Dataset

Class:
C1:buys_computer = 'yes'
C2:buys_computer = 'no'

Data sample
X = (age <=30,
Income = medium,
Student = yes
Credit_rating = Fair)

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

Naïve Bayesian Classifier: An Example

X = (age <=30, Income = medium, Student = yes, Credit_rating = Fair)

- Naïve Bayes Classifier $$P(C_i|\mathbf{X}) = \frac{P(\mathbf{X}|C_i)P(C_i)}{P(\mathbf{X})}$$
- P(Buy=Yes|X) = [P(X|Buy=yes) P(P(buy=yes)]/P(X)
- P(Buy=No|X) = [P(X|Buy=No) P(P(buy=No)]/P(X)

age	income	std	c_rating	buy
<=30	high	no	fair	no
<=30	high	no	excellent	no
31…40	high	no	fair	yes
>40	medium	no	fair	yes
>40	low	yes	fair	yes
>40	low	yes	excellent	no
31…40	low	yes	excellent	yes
<=30	medium	no	fair	no
<=30	low	yes	fair	yes
>40	medium	yes	fair	yes
<=30	medium	yes	excellent	yes
31…40	medium	no	excellent	yes
31…40	high	yes	fair	yes
>40	medium	no	excellent	no

Naïve Bayesian Classifier: An Example

X = (age <=30, Income = medium, Student = yes, Credit_rating = Fair)

- Naïve Bayes Classifier $$P(C_i|\mathbf{X}) = \frac{P(\mathbf{X}|C_i)P(C_i)}{P(\mathbf{X})}$$
- P(Buy=Yes|X) = [P(X|Buy=yes) P(P(buy=yes)]/P(X)
- P(Buy=No|X) = [P(X|Buy=No) P(P(buy=No)]/P(X)
- P(X) would be same for both cases

age	income	std	c_rating	buy
<=30	high	no	fair	no
<=30	high	no	excellent	no
31…40	high	no	fair	yes
>40	medium	no	fair	yes
>40	low	yes	fair	yes
>40	low	yes	excellent	no
31…40	low	yes	excellent	yes
<=30	medium	no	fair	no
<=30	low	yes	fair	yes
>40	medium	yes	fair	yes
<=30	medium	yes	excellent	yes
31…40	medium	no	excellent	yes
31…40	high	yes	fair	yes
>40	medium	no	excellent	no

Naïve Bayesian Classifier: An Example

X = (age <=30, Income = medium, Student = yes, Credit_rating = Fair)

- Naïve Bayes Classifier $$P(C_i|\mathbf{X}) = \frac{P(\mathbf{X}|C_i)P(C_i)}{P(\mathbf{X})}$$
- P(Buy=Yes|X)   = P(X|Buy=yes) P(buy=yes)
- P(Buy=No|X)   = P(X|Buy=No) P(buy=No)
- $P(C_i)$:   P(buy = "yes") = 9/14 = 0.643
  P(buy = "no") = 5/14= 0.357

age	income	std	c_rating	buy
<=30	high	no	fair	no
<=30	high	no	excellent	no
31…40	high	no	fair	yes
>40	medium	no	fair	yes
>40	low	yes	fair	yes
>40	low	yes	excellent	no
31…40	low	yes	excellent	yes
<=30	medium	no	fair	no
<=30	low	yes	fair	yes
>40	medium	yes	fair	yes
<=30	medium	yes	excellent	yes
31…40	medium	no	excellent	yes
31…40	high	yes	fair	yes
>40	medium	no	excellent	no

Naïve Bayesian Classifier: An Example

X = (age <=30, Income = medium, Student = yes, Credit_rating = Fair)

- Naïve Bayes Classifier $$P(C_i|\mathbf{X}) = \frac{P(\mathbf{X}|C_i)P(C_i)}{P(\mathbf{X})}$$
- C = {Yes, No}

- P(Buy=Yes|X) = P(X|Buy=yes) P(buy=yes)
- P(Buy=No|X) = P(X|Buy=No) P(buy=No)

- $P(C_i)$: P(buy = yes) = 9/14 = 0.643
  P(buy = no) = 5/14= 0.357
- Compute P(X|Buy = yes) for each class
  - P(age = "<=30" | buys_computer = "yes")
  - P(income = "medium" | buys_computer = "yes")
  - P(student = "yes" | buys_computer = "yes)
  - P(credit_rating = "fair" | buys_computer = "yes")

Naïve Bayesian Classifier: An Example

X = (age <=30, Income = medium, Student = yes, Credit_rating = Fair)

- Naïve Bayes Classifier $$P(C_i|\mathbf{X}) = \frac{P(\mathbf{X}|C_i)P(C_i)}{P(\mathbf{X})}$$
- C = {Yes, No}

- P(Buy=Yes|X)   = P(X|Buy=yes) P(buy=yes)
- P(Buy=No|X)   = P(X|Buy=No) P(buy=No)

- $P(C_i)$:   P(buy = yes) = 9/14 = 0.643
  P(buy = no) = 5/14= 0.357
- Compute P(X|Buy = yes) for each class
  - P(age = "<=30" | buys_computer = "yes")
  - P(income = "medium" | buys_computer = "yes")
  - P(student = "yes" | buys_computer = "yes)
  - P(credit_rating = "fair" | buys_computer = "yes")
- Compute P(X|Buy = No) same as above

Naïve Bayesian Classifier: An Example

age	income	std	c_rating	buy
<=30	high	no	fair	no
<=30	high	no	excellent	no
31…40	high	no	fair	yes
>40	medium	no	fair	yes
>40	low	yes	fair	yes
>40	low	yes	excellent	no
31…40	low	yes	excellent	yes
<=30	medium	no	fair	no
<=30	low	yes	fair	yes
>40	medium	yes	fair	yes
<=30	medium	yes	excellent	yes
31…40	medium	no	excellent	yes
31…40	high	yes	fair	yes
>40	medium	no	excellent	no

- $P(C_i)$: P(buys_computer = “yes”) = 9/14 = 0.643
  P(buys_computer = “no”) = 5/14= 0.357

- Compute $P(X|C_i)$ for each class

  P(age = “<=30” | buys_computer = “yes”) = 2/9 = 0.222
  P(age = “<= 30” | buys_computer = “no”) = 3/5 = 0.6

  P(income = “medium” | buys_computer = “yes”) = 4/9 = 0.444
  P(income = “medium” | buys_computer = “no”) = 2/5 = 0.4

  P(student = “yes” | buys_computer = “yes) = 6/9 = 0.667
  P(student = “yes” | buys_computer = “no”) = 1/5 = 0.2

  P(credit_rating = “fair” | buys_computer = “yes”) = 6/9 = 0.667
  P(credit_rating = “fair” | buys_computer = “no”) = 2/5 = 0.4

X = (age <=30, Income = medium, Student = yes, Credit_rating = Fair)

Naïve Bayesian Classifier: An Example

age	income	std	c_rating	buy
<=30	high	no	fair	no
<=30	high	no	excellent	no
31…40	high	no	fair	yes
>40	medium	no	fair	yes
>40	low	yes	fair	yes
>40	low	yes	excellent	no
31…40	low	yes	excellent	yes
<=30	medium	no	fair	no
<=30	low	yes	fair	yes
>40	medium	yes	fair	yes
<=30	medium	yes	excellent	yes
31…40	medium	no	excellent	yes
31…40	high	yes	fair	yes
>40	medium	no	excellent	no

- $P(C_i)$: P(buys_computer = “yes”) = 9/14 = 0.643
  P(buys_computer = “no”) = 5/14= 0.357

- Compute $P(X|C_i)$ for each class
  P(age = “<=30” | buys_computer = “yes”) = 2/9 = 0.222
  P(age = “<= 30” | buys_computer = “no”) = 3/5 = 0.6
  P(income = “medium” | buys_computer = “yes”) = 4/9 = 0.444
  P(income = “medium” | buys_computer = “no”) = 2/5 = 0.4
  P(student = “yes” | buys_computer = “yes) = 6/9 = 0.667
  P(student = “yes” | buys_computer = “no”) = 1/5 = 0.2
  P(credit_rating = “fair” | buys_computer = “yes”) = 6/9 = 0.667
  P(credit_rating = “fair” | buys_computer = “no”) = 2/5 = 0.4

- **X = (age <= 30 , income = medium, student = yes, credit_rating = fair)**

$P(X|C_i)$: P(X|buys_computer = “yes”) = 0.222 x 0.444 x 0.667 x 0.667 = 0.044
P(X|buys_computer = “no”) = 0.6 x 0.4 x 0.2 x 0.4 = 0.019

$P(X|C_i)*P(C_i)$: P(X|buys_computer = “yes”) * P(buys_computer = “yes”) = 0.028
P(X|buys_computer = “no”) * P(buys_computer = “no”) = 0.007

Avoiding the 0-Probability Problem

- Naïve Bayesian prediction requires each conditional prob. be non-zero. Otherwise, the predicted prob. will be zero

$$P(X|C_i) = \prod_{k=1}^{n} P(x_k|C_i)$$

- Ex. Suppose a dataset with 1000 tuples, income=low (0), income= medium (990), and income = high (10),
- Use Laplacian correction (or Laplacian estimator)
  - Adding 1 to each case

    Prob(income = low) = 1/1003

    Prob(income = medium) = 991/1003

    Prob(income = high) = 11/1003
  - The "corrected" prob. estimates are close to their "uncorrected" counterparts

Naïve Bayesian Classifier: Comments

- Advantages
  - Easy to implement
  - Good results obtained in most of the cases
- Disadvantages
  - Assumption: class conditional independence, therefore loss of accuracy
  - Practically, dependencies exist among variables
    - E.g., hospitals: patients: Profile: age, family history, etc.

      Symptoms: fever, cough etc., Disease: lung cancer, diabetes, etc.
    - Dependencies among these cannot be modeled by Naïve Bayesian Classifier
- How to deal with these dependencies?
  - Bayesian Belief Networks

Rule Based Classifiers

Dr. Faisal Kamiran

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
  - *LHS*: rule antecedent or condition
  - *RHS*: rule consequent
  - 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

Application of Rule-Based Classifier

- A rule *r* covers an instance **x** if the attributes of the instance satisfy the condition 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

Rule Coverage and Accuracy

- Coverage of a rule:
  – Fraction of records that satisfy the antecedent of a rule
- Accuracy of a rule:
  – Fraction of records that satisfy both the antecedent and consequent of a rule

Tid	Refund	Marital Status	Taxable Income	Class
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**

(Status=Single) $\rightarrow$ No

Coverage = 40%, Accuracy = 50%

How does Rule-based Classifier Work?

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
lemur	warm	yes	no	no	?
turtle	cold	no	no	sometimes	?
dogfish shark	cold	yes	no	yes	?

A lemur triggers rule R3, so it is classified as a mammal

A turtle triggers both R4 and R5

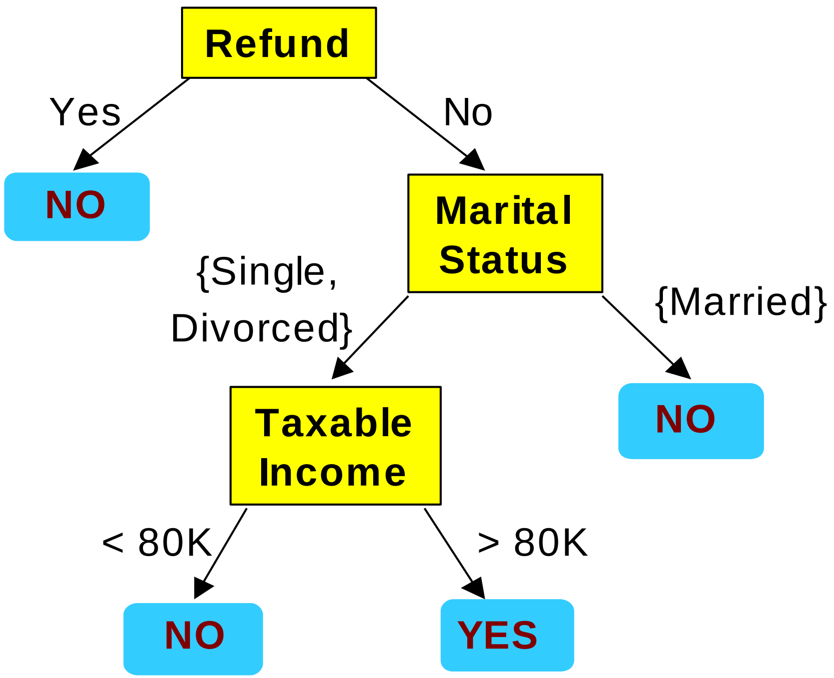
A dogfish shark triggers none of the rules

Characteristics of Rule-Based Classifier

- Mutually exclusive rules
  - Classifier contains mutually exclusive rules if the rules are independent of each other
  - Every record is covered by at most one rule

- Exhaustive rules
  - Classifier has exhaustive coverage if it accounts for every possible combination of attribute values
  - Each record is covered by at least one rule

From Decision Trees To Rules



Classification Rules

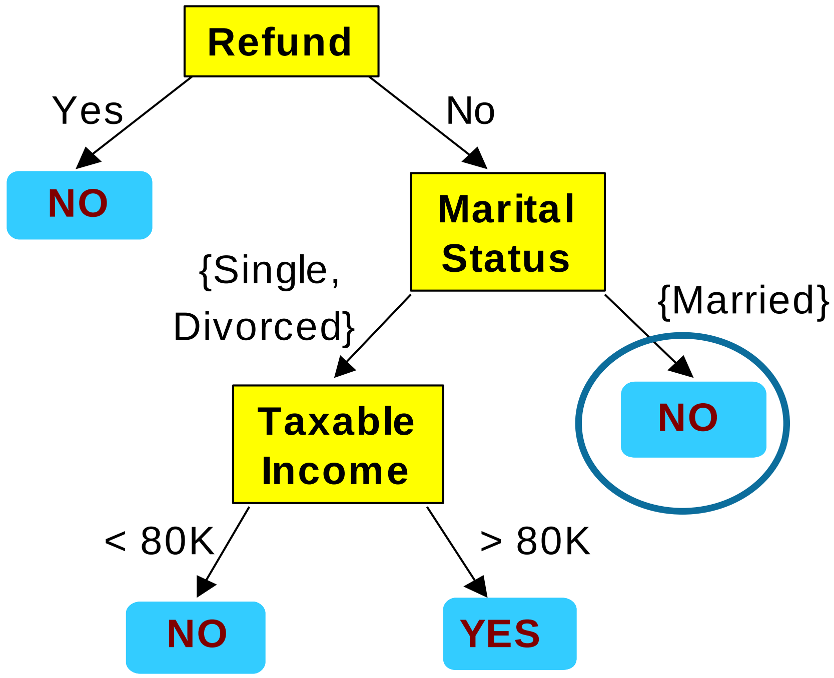
(Refund=Yes) ==> No

(Refund=No, Marital Status={Single,Divorced}, Taxable Income<80K) ==> No

(Refund=No, Marital Status={Single,Divorced}, Taxable Income>80K) ==> Yes

(Refund=No, Marital Status={Married}) ==> No

Rules Can Be Simplified



Tid	Refund	Marital Status	Taxable Income	Cheat
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

Initial Rule: $(\text{Refund=No}) \wedge (\text{Status=Married}) \rightarrow \text{No}$

Simplified Rule: $(\text{Status=Married}) \rightarrow \text{No}$

Ordered Rule Set

- Rules are rank ordered according to their priority
  - An ordered rule set is known as a decision list
- When a test record is presented to the classifier
  - It is assigned to the class label of the highest ranked rule it has triggered
  - If none of the rules fired, it is assigned to the default class

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
turtle	cold	no	no	sometimes	?

Rule Ordering Schemes

- Size based ordering
  - High priority to the rules with toughest requirement
  - Toughness is measured by the size of antecedent
- Rule-based ordering
  - Individual rules are ranked based on their quality, i.e., accuracy, coverage, size and advice from the domain experts.
- Class-based ordering
  - Priority to the rules with most prevalent classes or most rare classes

Building Classification Rules

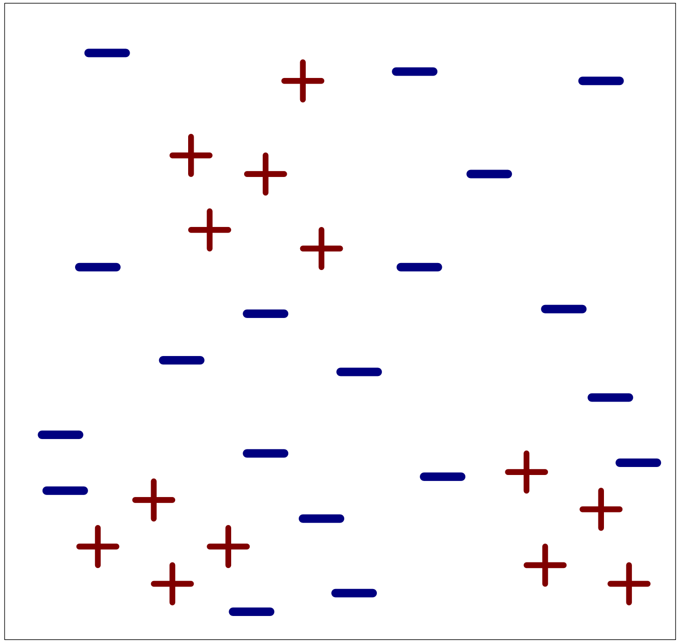
- Direct Method:
  - Extract rules directly from data
  - e.g.: RIPPER, CN2, Holte's 1R

- Indirect Method:
  - Extract rules from other classification models (e.g. decision trees, neural networks, etc).
  - e.g: C4.5rules

Direct Method: Sequential Covering

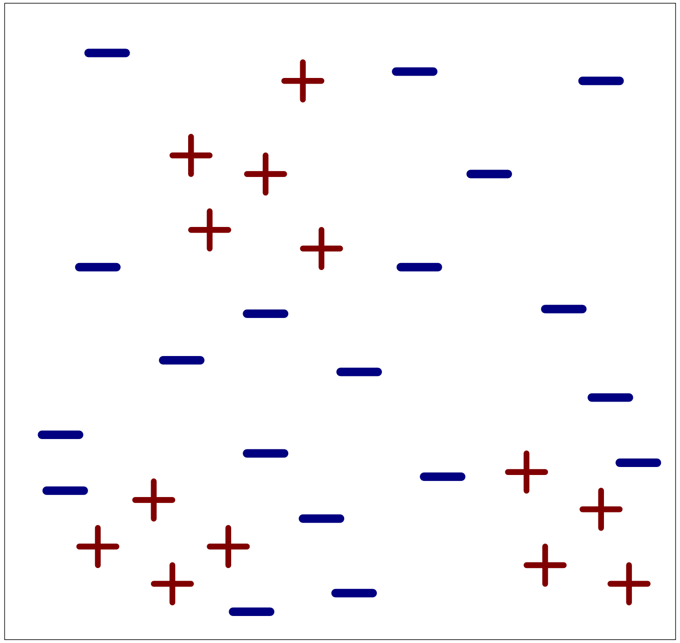
1. Start from an empty rule
2. Grow a rule using the Learn-One-Rule function
3. Remove training records covered by the rule
4. Repeat Step (2) and (3) until stopping criterion is met

Example of Sequential Covering

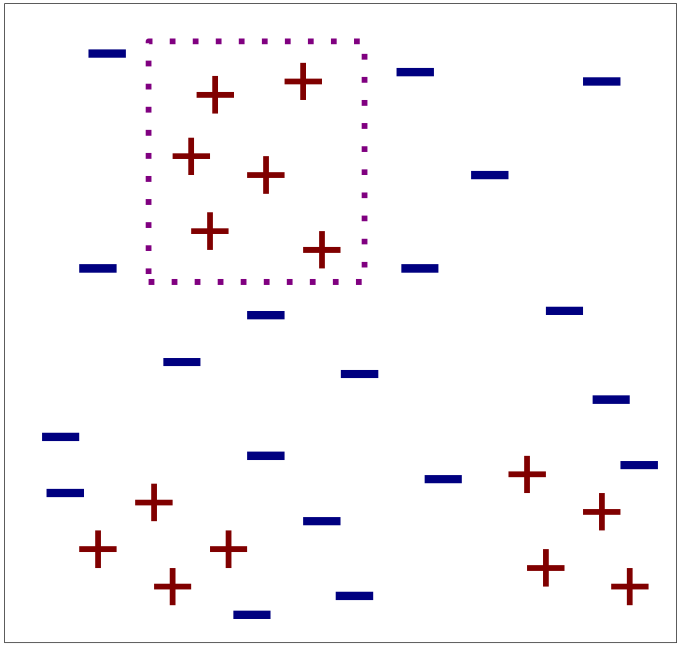


(i) Original Data

Example of Sequential Covering

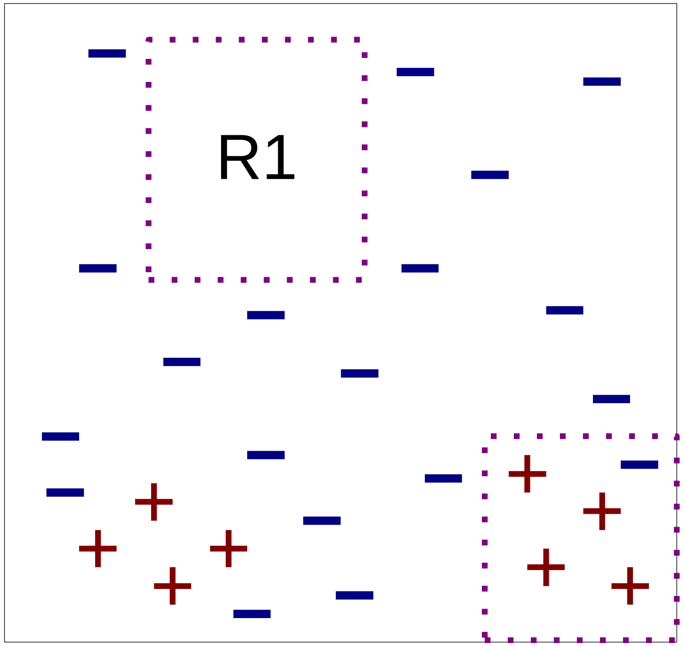


(i) Original Data



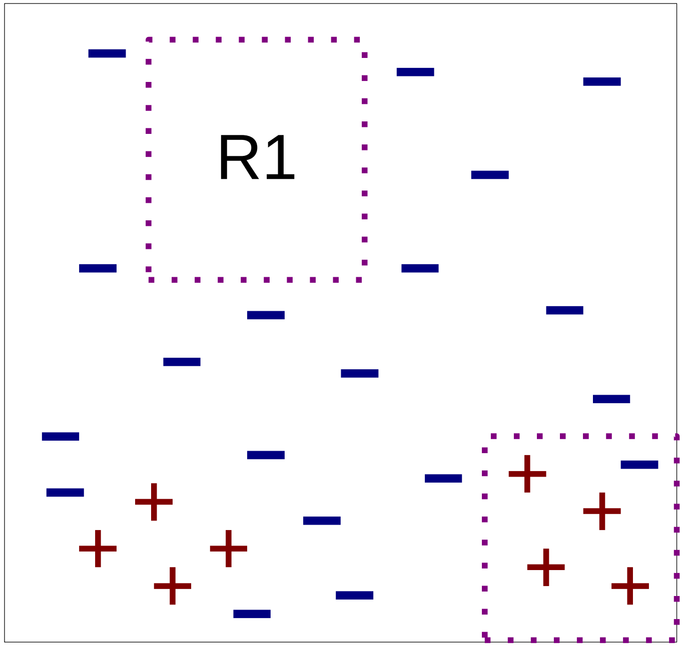
(ii) Step 1

Example of Sequential Covering…

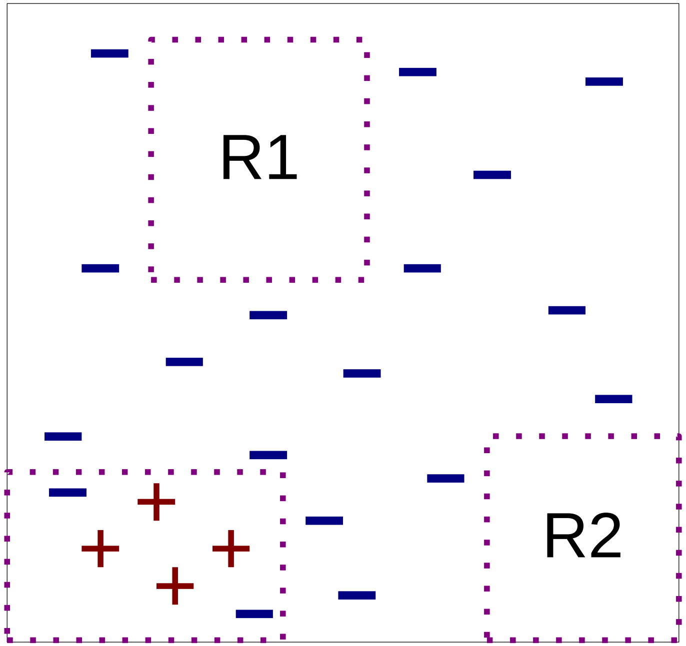


(iii) Step 2

Example of Sequential Covering…



(iii) Step 2



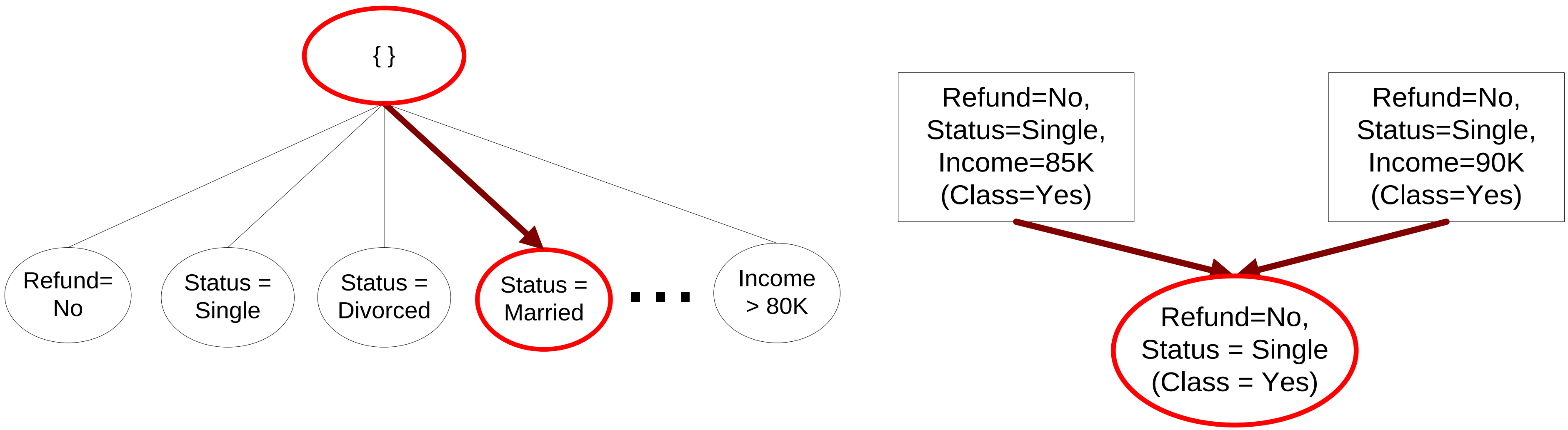
(iv) Step 3

Aspects of Sequential Covering

- Rule Growing
- Instance Elimination
- Rule Evaluation
- Stopping Criterion
- Rule Pruning

Rule Growing

- Two common strategies

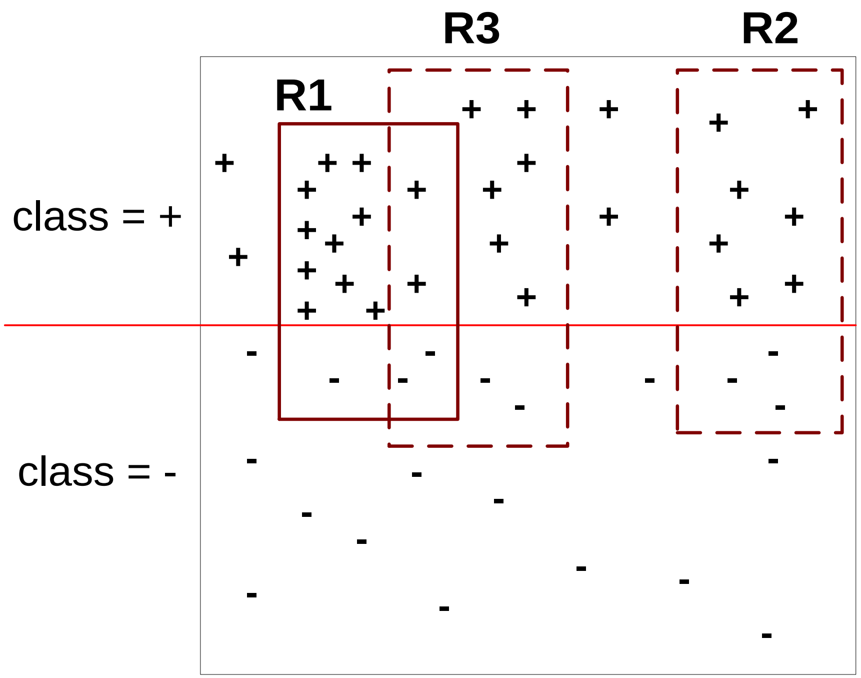


(a) General-to-specific

(b) Specific-to-general

Instance Elimination

- Why do we need to eliminate instances?
  - Otherwise, the next rule is identical to previous rule
- Why do we remove positive instances?
  - Ensure that the next rule is different
- Why do we remove negative instances?
  - Prevent underestimating accuracy of rule
  - Compare rules R2 and R3 in the diagram



Rule Evaluation

- Metrics:
  - Accuracy $= \dfrac{n_c}{n}$
  - Laplace $= \dfrac{n_c + 1}{n + k}$

n : Number of instances covered by rule

n_c : Number of instances correctly covered by rule

k : Number of classes

p : Prior probability

Stopping Criterion and Rule Pruning

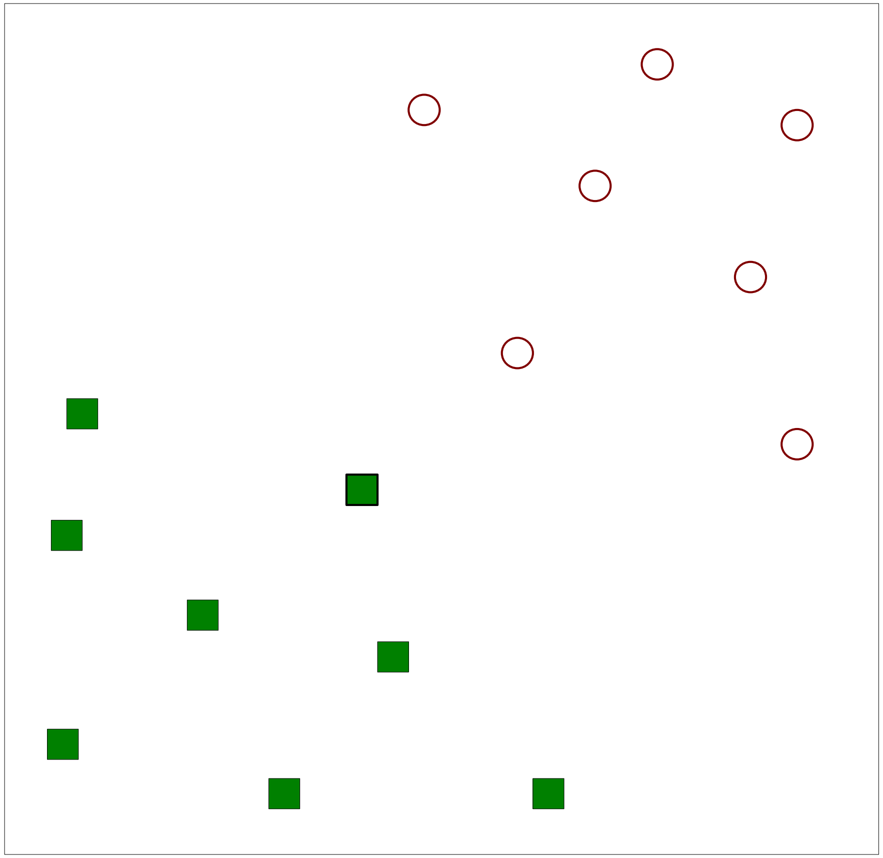
- Stopping criterion
  - Compute the gain
  - If gain is not significant, discard the new rule

- 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

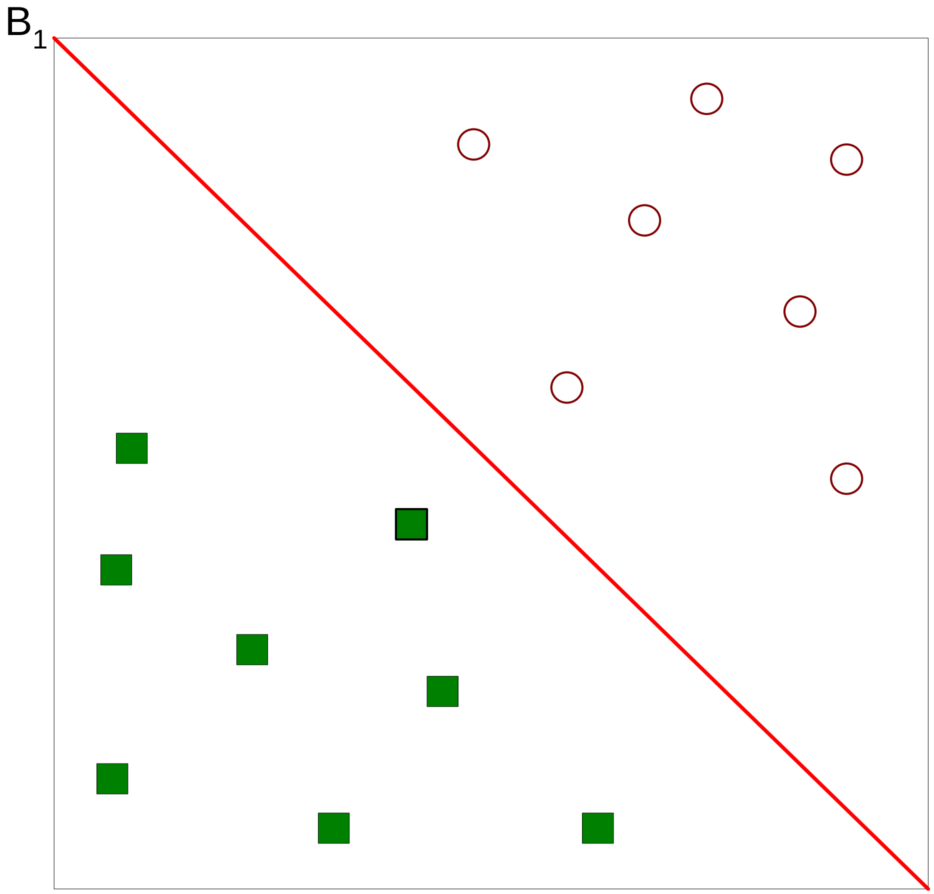
- Grow a single rule
- Remove Instances from rule
- Prune the rule (if necessary)
- Add rule to Current Rule Set
- Repeat

Support Vector Machines



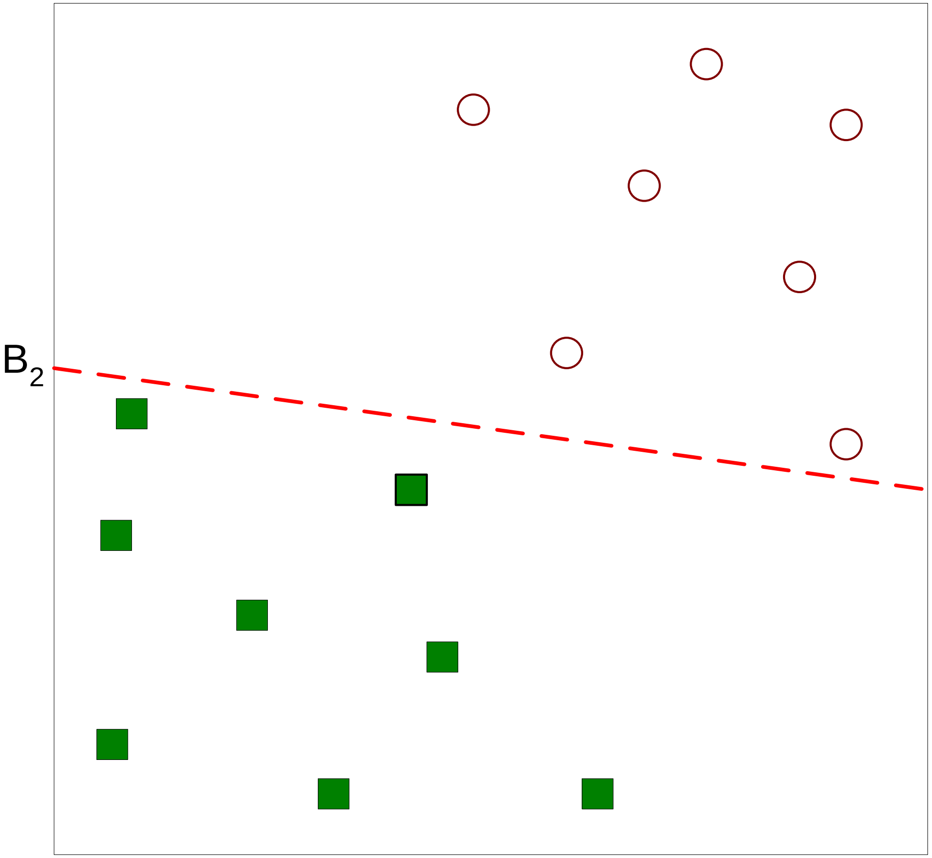
- Find a linear hyperplane (decision boundary) that will separate the data

Support Vector Machines



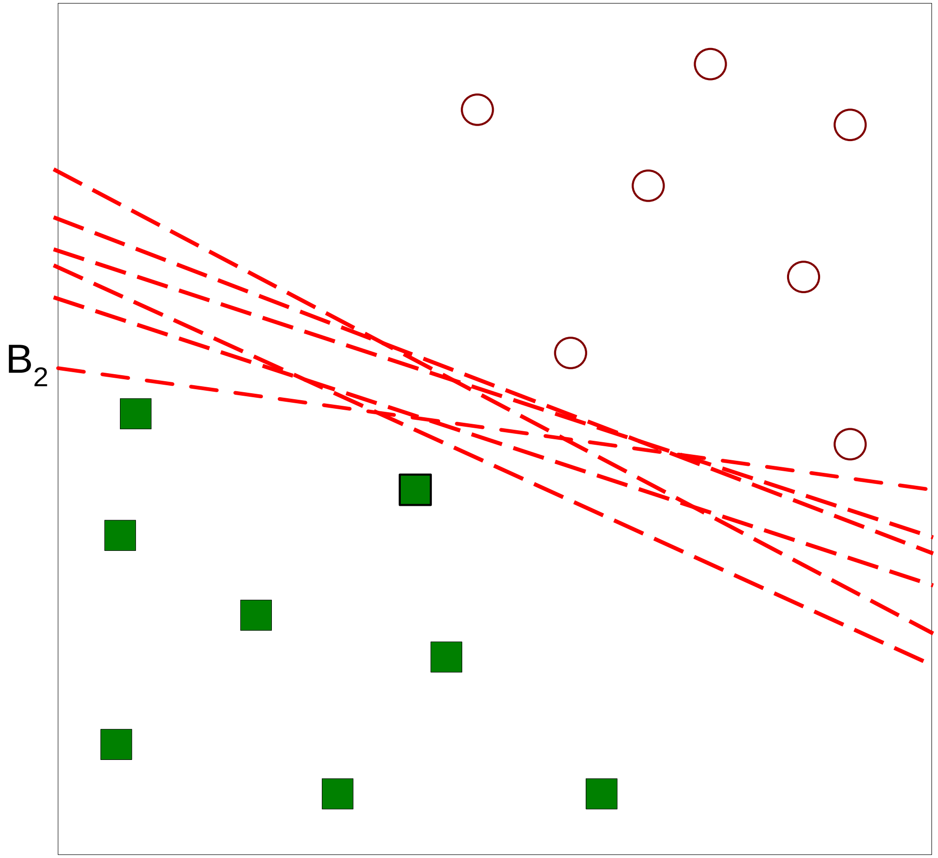
- One Possible Solution

Support Vector Machines



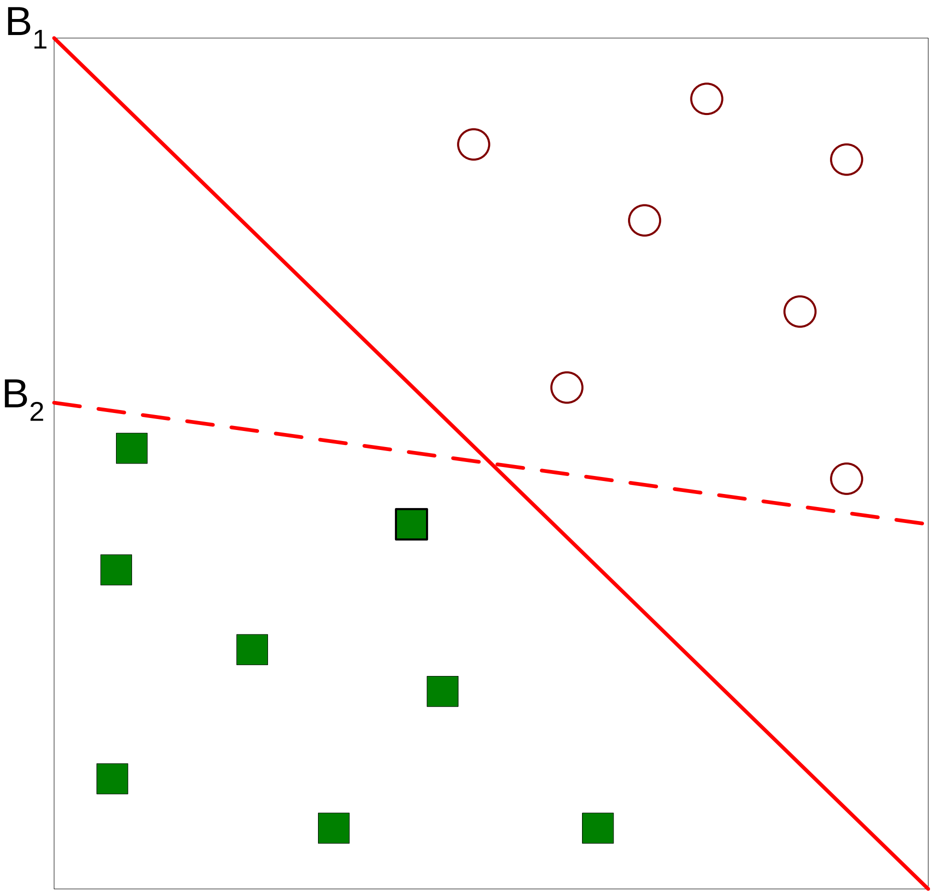
- Another possible solution

Support Vector Machines



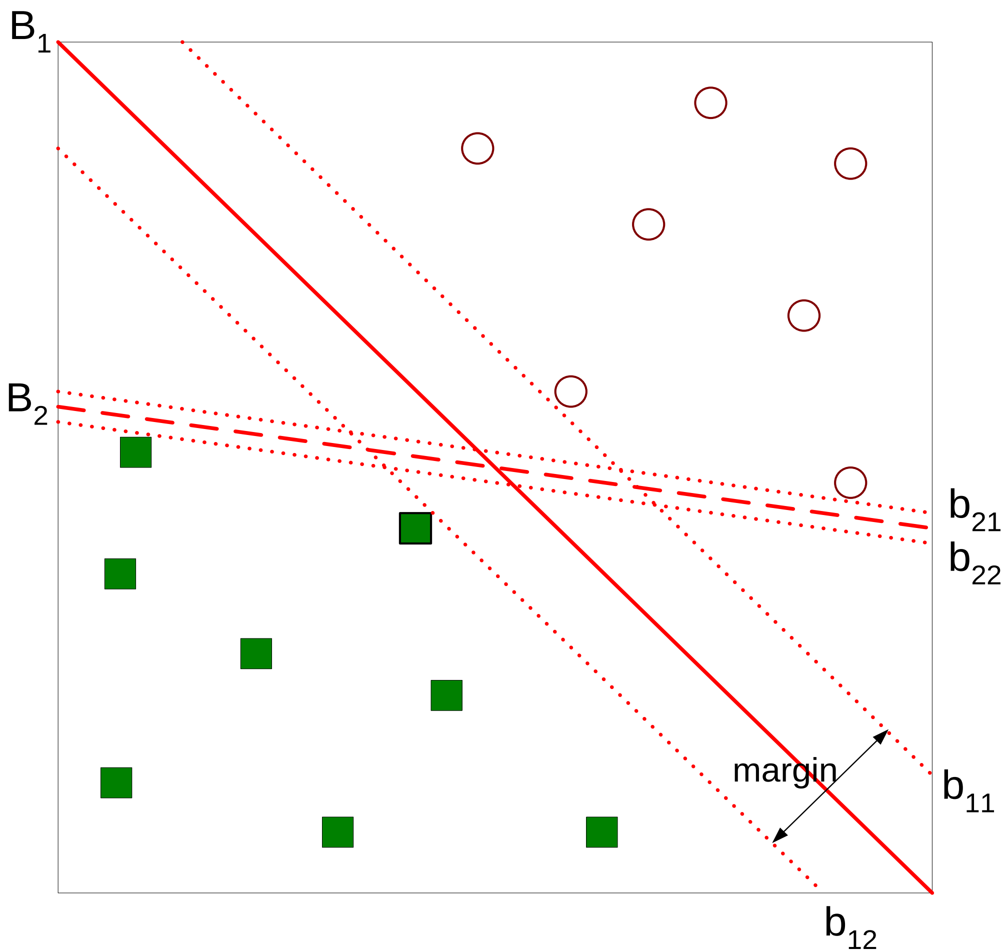
- Other possible solutions

Support Vector Machines



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

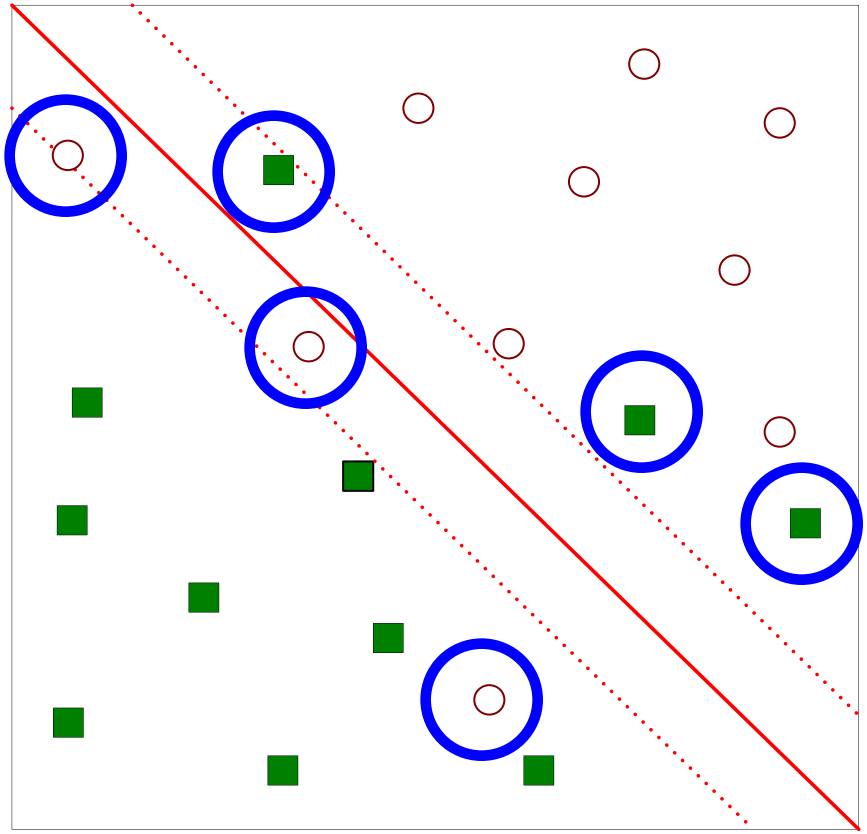
SVM: Maximum Margin Hyperplane



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

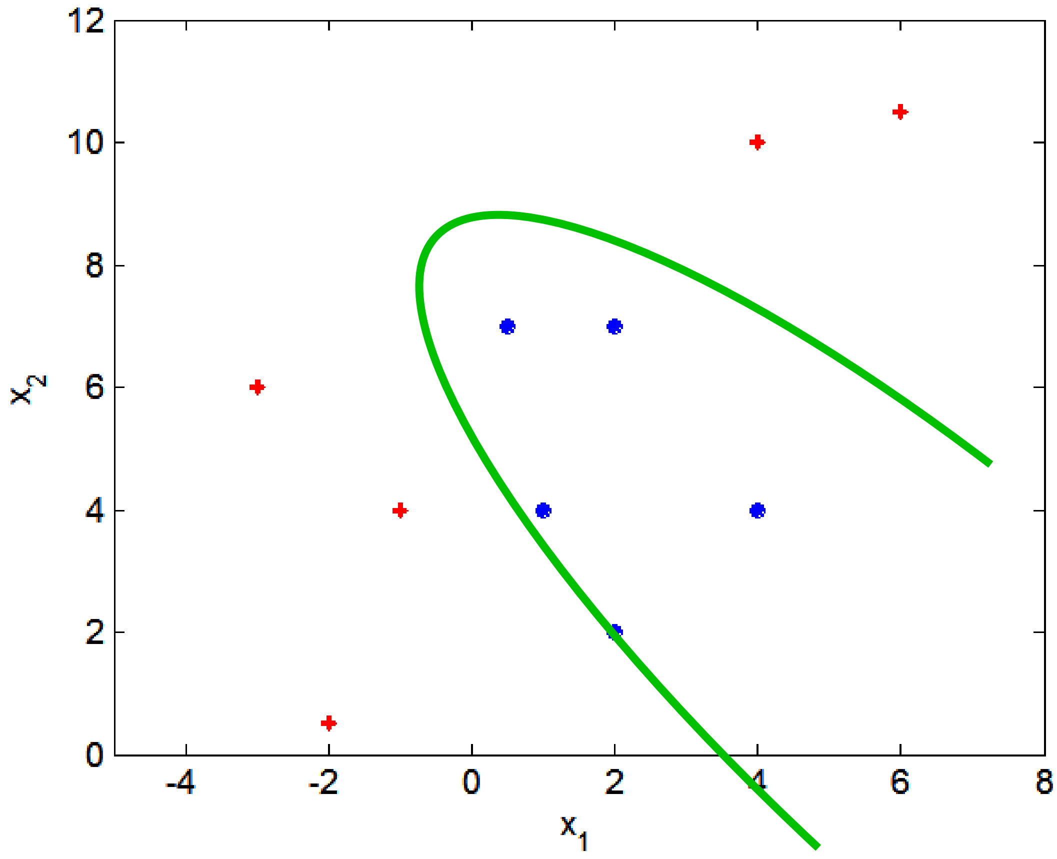
Support Vector Machines

- What if the problem is not linearly separable?



Nonlinear Support Vector Machines

- What if decision boundary is not linear?



Rule Based Classifiers

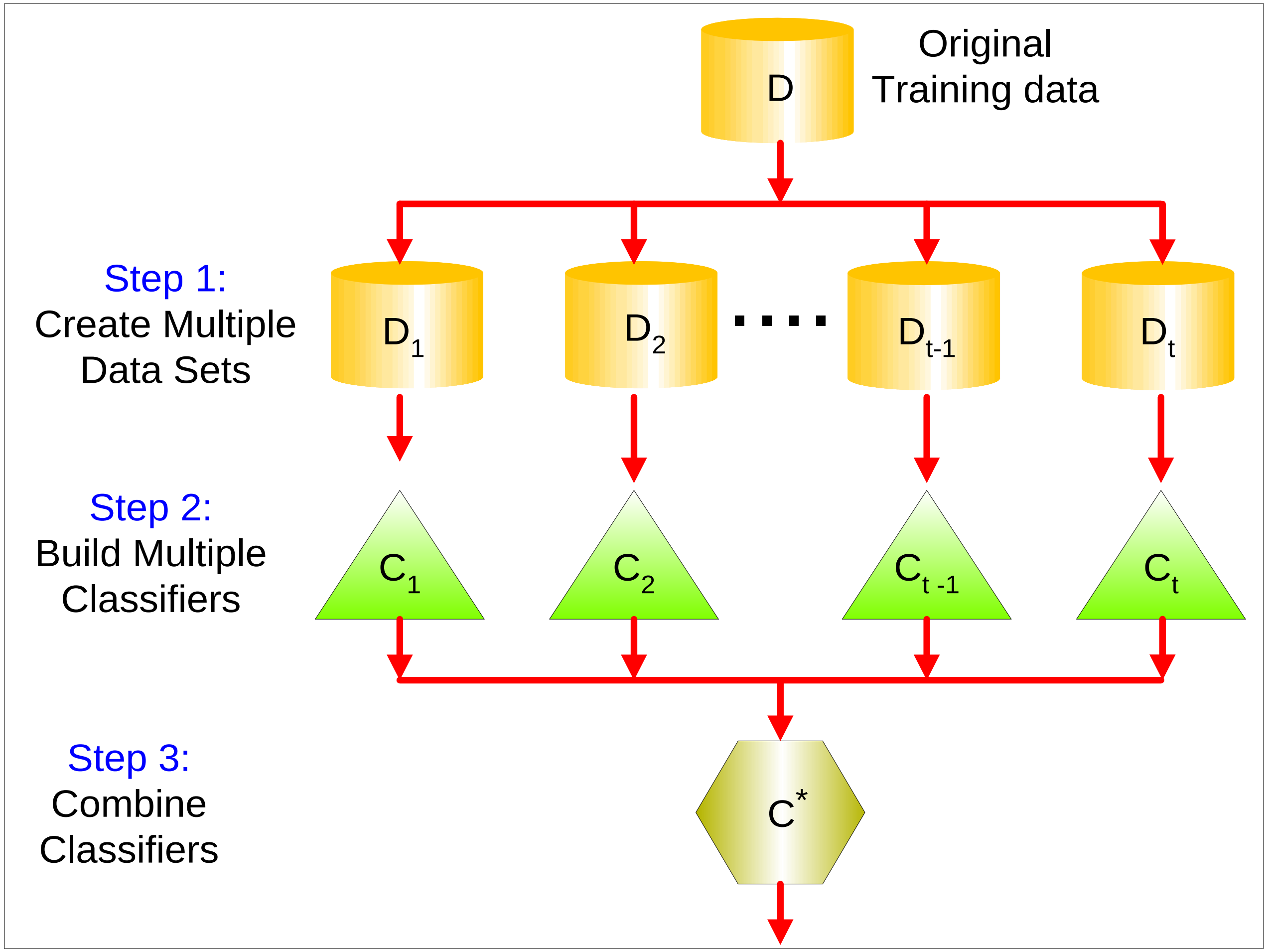
Dr. Faisal Kamiran

Ensemble Methods

- Construct a set of classifiers from the training data

- Predict class label of previously unseen records by aggregating predictions made by multiple classifiers

General Idea



Why does it work?

- Suppose there are 25 base classifiers
  - Each classifier has error rate, $\varepsilon = 0.35$
  - Assume classifiers are independent
  - Probability that the ensemble classifier makes a wrong prediction by **majority voting**:

$$\sum_{i=13}^{25} \binom{25}{i} \varepsilon^i (1-\varepsilon)^{25-i} = 0.06$$

Methods for Ensemble Classifiers

- By manipulating the training data:
  - Bagging
  - Boosting
- By manipulating the input features
  - Random forest
- By manipulating the class labels
  - Error correcting output coding
- By manipulating the learning algorithm

Bagging

- Create multiple samples of original data (almost of same size) with replacement

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

- Each sample, D_i, is called bootstrap sample
- Build classifier on each bootstrap sample

Bagging Algorithm

Algorithm 5.6 Bagging algorithm.

1: Let k be the number of bootstrap samples.
2: **for** $i = 1$ to k **do**
3: Create a bootstrap sample of size N, D_i.
4: Train a base classifier C_i on the bootstrap sample D_i.
5: **end for**
6: $C^*(x) = \underset{y}{\operatorname{argmax}} \sum_i \delta\big(C_i(x) = y\big)$.
$\{\delta(\cdot) = 1$ if its argument is true and 0 otherwise$\}$.

Boosting

- An iterative procedure to adaptively change distribution of training data by focusing more on previously misclassified records
  - Initially, all N records are assigned equal weights
  - Unlike bagging, weights may change at the end of boosting round

Boosting

- Records that are wrongly classified will have their weights increased
- Records that are classified correctly will have their weights decreased

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

- Example 4 is hard to classify
- Its weight is increased, therefore it is more likely to be chosen again in subsequent rounds

Example: AdaBoost

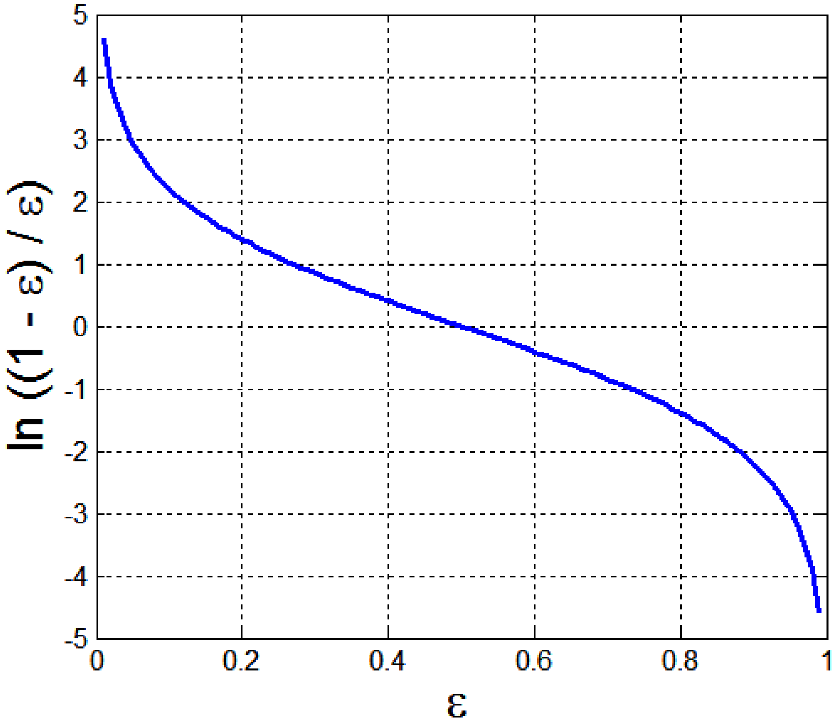
- Base classifiers: $C_1, C_2, \ldots, C_T$

- Error rate:

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

- Importance of a classifier:

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



Example: AdaBoost

- Weight update:

$$w_i^{(j+1)} = \frac{w_i^{(j)}}{Z_j} \begin{cases} \exp^{-\alpha_j} & \text{if } C_j(x_i) = y_i \\ \exp^{\alpha_j} & \text{if } C_j(x_i) \neq y_i \end{cases}$$

where Z_j is the normalization factor

- If any intermediate rounds produce error rate higher than 50%, the weights are reverted back to 1/n and the resampling procedure is repeated
- Classification: $C^*(x) = \arg\max_y \sum_{j=1}^{T} \alpha_j \delta\left(C_j(x) = y\right)$

AdaBoost Algorithm

Algorithm 5.7 AdaBoost algorithm.

1: $\mathbf{w} = \{w_j = 1/N \mid j = 1, 2, \ldots, N\}$. {Initialize the weights for all N examples.}
2: Let k be the number of boosting rounds.
3: **for** $i = 1$ to k **do**
4:     Create training set D_i by sampling (with replacement) from D according to $\mathbf{w}$.
5:     Train a base classifier C_i on D_i.
6:     Apply C_i to all examples in the original training set, D.
7:     $\epsilon_i = \frac{1}{N}\left[\sum_j w_j \; \delta\big(C_i(x_j) \neq y_j\big)\right]$ {Calculate the weighted error.}
8:     **if** $\epsilon_i > 0.5$ **then**
9:         $\mathbf{w} = \{w_j = 1/N \mid j = 1, 2, \ldots, N\}$. {Reset the weights for all N examples.}
10:         Go back to Step 4.
11:     **end if**
12:     $\alpha_i = \frac{1}{2} \ln \frac{1-\epsilon_i}{\epsilon_i}$.
13:     Update the weight of each example according to Equation 5 **given on the prev slide**
14: **end for**
15: $C^*(\mathbf{x}) = \underset{y}{\operatorname{argmax}} \sum_{j=1}^{T} \alpha_j \delta\big(C_j(\mathbf{x}) = y\big)$.