Data Mining
Classification: Alternative Techniques

Lecture Notes for Chapter 5

Introduction to Data Mining
by
Tan, Steinbach, Kumar

(modified by Predrag Radivojac, 2016)
Rule-Based Classifiers

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

- Rule: \((Condition) \rightarrow y\)
  - where
    - \(Condition\) is a conjunction of attributes
    - \(y\) is the class label
  - \(LHS\): rule antecedent or condition
  - \(RHS\): rule consequent

- Examples of classification rules:
  - \((Blood\ Type = Warm) \land (Lay\ Eggs = Yes) \rightarrow Birds\)
  - \((Taxable\ Income < 50K) \land (Refund = Yes) \rightarrow Evade = No\)
Rule-based Classifiers (Example)

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

$R_1$: (Give Birth = no) $\land$ (Can Fly = yes) $\rightarrow$ Birds

$R_2$: (Give Birth = no) $\land$ (Live in Water = yes) $\rightarrow$ Fishes

$R_3$: (Give Birth = yes) $\land$ (Blood Type = warm) $\rightarrow$ Mammals

$R_4$: (Give Birth = no) $\land$ (Can Fly = no) $\rightarrow$ Reptiles

$R_5$: (Live in Water = sometimes) $\rightarrow$ Amphibians
Application of Rule-Based Classifiers

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

$R_1$: (Give Birth = no) $\land$ (Can Fly = yes) $\rightarrow$ Birds
$R_2$: (Give Birth = no) $\land$ (Live in Water = yes) $\rightarrow$ Fishes
$R_3$: (Give Birth = yes) $\land$ (Blood Type = warm) $\rightarrow$ Mammals
$R_4$: (Give Birth = no) $\land$ (Can Fly = no) $\rightarrow$ Reptiles
$R_5$: (Live in Water = sometimes) $\rightarrow$ Amphibians

<table>
<thead>
<tr>
<th>Name</th>
<th>Blood Type</th>
<th>Give Birth</th>
<th>Can Fly</th>
<th>Live in Water</th>
<th>Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>hawk</td>
<td>warm</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>?</td>
</tr>
<tr>
<td>grizzly bear</td>
<td>warm</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>?</td>
</tr>
</tbody>
</table>

The rule $R_1$ covers a hawk $\Rightarrow$ Bird
The rule $R_3$ covers the grizzly bear $\Rightarrow$ 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

<table>
<thead>
<tr>
<th>Tid</th>
<th>Refund</th>
<th>Marital Status</th>
<th>Taxable Income</th>
<th>Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Yes</td>
<td>Single</td>
<td>125K</td>
<td>No</td>
</tr>
<tr>
<td>2</td>
<td>No</td>
<td>Married</td>
<td>100K</td>
<td>No</td>
</tr>
<tr>
<td>3</td>
<td>No</td>
<td>Single</td>
<td>70K</td>
<td>No</td>
</tr>
<tr>
<td>4</td>
<td>Yes</td>
<td>Married</td>
<td>120K</td>
<td>No</td>
</tr>
<tr>
<td>5</td>
<td>No</td>
<td>Divorced</td>
<td>95K</td>
<td>Yes</td>
</tr>
<tr>
<td>6</td>
<td>No</td>
<td>Married</td>
<td>60K</td>
<td>No</td>
</tr>
<tr>
<td>7</td>
<td>Yes</td>
<td>Divorced</td>
<td>220K</td>
<td>No</td>
</tr>
<tr>
<td>8</td>
<td>No</td>
<td>Single</td>
<td>85K</td>
<td>Yes</td>
</tr>
<tr>
<td>9</td>
<td>No</td>
<td>Married</td>
<td>75K</td>
<td>No</td>
</tr>
<tr>
<td>10</td>
<td>No</td>
<td>Single</td>
<td>90K</td>
<td>Yes</td>
</tr>
</tbody>
</table>

\[ R_1: (\text{Status} = \text{Single}) \rightarrow \text{No} \]

Coverage = 40\%,  Accuracy = 50\%
How Rule-based Classifiers Work

\[ R_1: (\text{Give Birth} = \text{no}) \land (\text{Can Fly} = \text{yes}) \rightarrow \text{Birds} \]
\[ R_2: (\text{Give Birth} = \text{no}) \land (\text{Live in Water} = \text{yes}) \rightarrow \text{Fishes} \]
\[ R_3: (\text{Give Birth} = \text{yes}) \land (\text{Blood Type} = \text{warm}) \rightarrow \text{Mammals} \]
\[ R_4: (\text{Give Birth} = \text{no}) \land (\text{Can Fly} = \text{no}) \rightarrow \text{Reptiles} \]
\[ R_5: (\text{Live in Water} = \text{sometimes}) \rightarrow \text{Amphibians} \]

<table>
<thead>
<tr>
<th>Name</th>
<th>Blood Type</th>
<th>Give Birth</th>
<th>Can Fly</th>
<th>Live in Water</th>
<th>Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>lemur</td>
<td>warm</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>?</td>
</tr>
<tr>
<td>turtle</td>
<td>cold</td>
<td>no</td>
<td>no</td>
<td>sometimes</td>
<td>?</td>
</tr>
<tr>
<td>dogfish shark</td>
<td>cold</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
<td>?</td>
</tr>
</tbody>
</table>

A lemur triggers rule \( R_3 \), so it is classified as a mammal
A turtle triggers both \( R_4 \) and \( R_5 \)
A dogfish shark triggers none of the rules
Characteristics of Rule-Based Classifiers

- **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
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 are mutually exclusive and exhaustive. The rule set contains as much information as the tree.
Rules Can Be Simplified

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

Simplified Rule: \((\text{Status} = \text{Married}) \rightarrow \text{No}\)
Effect of Rule Simplification

- Rules are no longer mutually exclusive
  - A record may trigger more than one rule
  - Solution?
    - Ordered rule set
    - Unordered rule set – use voting schemes

- Rules are no longer exhaustive
  - A record may not trigger any rules
  - Solution?
    - Use a default class
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

| R_1: (Give Birth = no) \land (Can Fly = yes) \rightarrow Birds |
| R_2: (Give Birth = no) \land (Live in Water = yes) \rightarrow Fishes |
| R_3: (Give Birth = yes) \land (Blood Type = warm) \rightarrow Mammals |
| R_4: (Give Birth = no) \land (Can Fly = no) \rightarrow Reptiles |
| R_5: (Live in Water = sometimes) \rightarrow Amphibians |

<table>
<thead>
<tr>
<th>Name</th>
<th>Blood Type</th>
<th>Give Birth</th>
<th>Can Fly</th>
<th>Live in Water</th>
<th>Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>turtle</td>
<td>cold</td>
<td>no</td>
<td>no</td>
<td>sometimes</td>
<td>?</td>
</tr>
</tbody>
</table>
Rule Ordering Schemes

- Rule-based ordering
  - Individual rules are ranked based on their quality
- Class-based ordering
  - Rules that belong to the same class appear together

**Rule-based Ordering**

(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

**Class-based Ordering**

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

(ii) Step 1
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

We need a criterion to pick the best path:

Data set: 60 positives and 100 negatives

R₁: covers 50 positives and 5 negatives
R₂: covers 2 positives and no negatives

Which rule is better?
Rule Growing (Examples)

- **CN2 Algorithm:**
  - Start from an empty conjunct: \{\}
  - Add conjuncts that minimizes the entropy measure: \{A\}, \{A,B\}, …
  - Determine the rule consequent by taking majority class of instances covered by the rule

- **RIPPER Algorithm:**
  - Start from an empty rule: \{} \Rightarrow \text{class}
  - Add conjuncts that maximizes FOIL’s information gain measure:
    - \(R_0: \{} \Rightarrow \text{class} \) (initial rule)
    - \(R_1: \{A\} \Rightarrow \text{class} \) (rule after adding conjunct)
    - Gain\((R_0, R_1) = t \cdot \left[ \log \left( \frac{p_1}{p_1 + n_1} \right) - \log \left( \frac{p_0}{p_0 + n_0} \right) \right] \)
    - where \(t\): number of positive instances covered by both \(R_0\) and \(R_1\)
      - \(p_0\): number of positive instances covered by \(R_0\)
      - \(n_0\): number of negative instances covered by \(R_0\)
      - \(p_1\): number of positive instances covered by \(R_1\)
      - \(n_1\): number of negative instances covered by \(R_1\)
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** = \( \frac{n_c}{n} \)
  - **Laplace** = \( \frac{n_c + 1}{n + k} \)
  - **M-estimate** = \( \frac{n_c + kp}{n + k} \)

\( n \): Number of instances covered by rule
\( n_c \): Number of instances covered by rule, in class
\( k \): 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
Direct Method: RIPPER

- For 2-class problems, 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 problems
  - 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
Direct Method: RIPPER

- Growing a rule:
  - Start from empty rule
  - Add conjuncts as long as they improve FOIL’s information gain
  - Stop when rule no longer covers negative examples
  - Prune the rule immediately using incremental reduced error pruning
  - Measure for pruning: $v = \frac{p-n}{p+n}$
    - p: number of positive examples covered by the rule in the validation set
    - n: number of negative examples covered by the rule in the validation set
  - Pruning method: delete any final sequence of conditions that maximizes $v$
Direct Method: RIPPER

- Building a Rule Set:
  - Use sequential covering algorithm
    - Finds the best rule that covers the current set of positive examples
    - Eliminate both positive and negative examples covered by the rule
  - Each time a rule is added to the rule set, compute the new description length
    - stop adding new rules when the new description length is $d$ bits longer than the smallest description length obtained so far
Direct Method: RIPPER

- Optimize the rule set:
  - For each rule \( r \) in the rule set \( R \)
    - Consider 2 alternative rules:
      - Replacement rule (\( r^* \)): grow new rule from scratch
      - Revised rule (\( r' \)): add conjuncts to extend the rule \( r \)
    - Compare the rule set for \( r \) against the rule set for \( r^* \) and \( r' \)
    - Choose rule set that minimizes MDL principle
  - Repeat rule generation and rule optimization for the remaining positive examples
Indirect Methods

**Rule Set**

r1: \((P=No, Q=No) \implies -\)

r2: \((P=No, Q=Yes) \implies +\)

r3: \((P=Yes, R=No) \implies +\)

r4: \((P=Yes, R=Yes, Q=No) \implies -\)

r5: \((P=Yes, R=Yes, Q=Yes) \implies +\)
Indirect Method: C4.5rules

- Extract rules from an unpruned decision tree
- For each rule, r: A → y,
  - consider an alternative rule r’: A’ → y where A’ is obtained by removing one of the conjuncts in A
  - Compare the pessimistic error rate for r against all r’s
  - Prune if one of the r’s has lower pessimistic error rate
  - Repeat until we can no longer improve generalization error
Indirect Method: C4.5rules

- Instead of ordering the rules, order subsets of rules (class ordering)
  - Each subset is a collection of rules with the same rule consequent (class)
  - Compute description length of each subset
    - Description length = $L(\text{error}) + g \cdot L(\text{model})$
    - $g$ is a parameter that takes into account the presence of redundant attributes in a rule set (default value = 0.5)
<table>
<thead>
<tr>
<th>Name</th>
<th>Give Birth</th>
<th>Lay Eggs</th>
<th>Can Fly</th>
<th>Live in Water</th>
<th>Have Legs</th>
<th>Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>human</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>mammals</td>
</tr>
<tr>
<td>python</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>reptiles</td>
</tr>
<tr>
<td>salmon</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>fishes</td>
</tr>
<tr>
<td>whale</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>mammals</td>
</tr>
<tr>
<td>frog</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>sometimes</td>
<td>yes</td>
<td>amphibians</td>
</tr>
<tr>
<td>komodo</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>reptiles</td>
</tr>
<tr>
<td>bat</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
<td>mammals</td>
</tr>
<tr>
<td>pigeon</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
<td>birds</td>
</tr>
<tr>
<td>cat</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>mammals</td>
</tr>
<tr>
<td>leopard shark</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>fishes</td>
</tr>
<tr>
<td>turtle</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>sometimes</td>
<td>yes</td>
<td>reptiles</td>
</tr>
<tr>
<td>penguin</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>sometimes</td>
<td>yes</td>
<td>birds</td>
</tr>
<tr>
<td>porcupine</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>sometimes</td>
<td>yes</td>
<td>mammals</td>
</tr>
<tr>
<td>eel</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>fishes</td>
</tr>
<tr>
<td>salamander</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>sometimes</td>
<td>yes</td>
<td>amphibians</td>
</tr>
<tr>
<td>gila monster</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>reptiles</td>
</tr>
<tr>
<td>platypus</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>mammals</td>
</tr>
<tr>
<td>owl</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
<td>birds</td>
</tr>
<tr>
<td>dolphin</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>mammals</td>
</tr>
<tr>
<td>eagle</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
<td>birds</td>
</tr>
</tbody>
</table>
C4.5 versus C4.5rules versus RIPPER

C4.5rules:
(Give Birth=No, Can Fly=Yes) → Birds
(Give Birth=No, Live in Water=Yes) → Fishes
(Give Birth=Yes) → Mammals
(Give Birth=No, Can Fly=No, Live in Water=No) → Reptiles
( ) → Amphibians

RIPPER:
(Live in Water=Yes) → Fishes
(Have Legs=No) → Reptiles
(Give Birth=No, Can Fly=No, Live In Water=No) → Reptiles
(Can Fly=Yes,Give Birth=No) → Birds
( ) → Mammals
## C4.5 versus C4.5rules versus RIPPER

### C4.5 and C4.5rules:

<table>
<thead>
<tr>
<th>ACTUAL CLASS</th>
<th>Amphibians</th>
<th>Fishes</th>
<th>Reptiles</th>
<th>Birds</th>
<th>Mammals</th>
</tr>
</thead>
<tbody>
<tr>
<td>Amphibians</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Fishes</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Reptiles</td>
<td>1</td>
<td>0</td>
<td>3</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Birds</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td>Mammals</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>6</td>
</tr>
</tbody>
</table>

### RIPPER:

<table>
<thead>
<tr>
<th>ACTUAL CLASS</th>
<th>Amphibians</th>
<th>Fishes</th>
<th>Reptiles</th>
<th>Birds</th>
<th>Mammals</th>
</tr>
</thead>
<tbody>
<tr>
<td>Amphibians</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>Fishes</td>
<td>0</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Reptiles</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Birds</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>Mammals</td>
<td>0</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>4</td>
</tr>
</tbody>
</table>
Advantages of Rule-Based Classifiers

- As highly expressive as decision trees
- Easy to interpret
- Easy to generate
- Can classify new instances rapidly
- Performance comparable to decision trees
Instance-Based Classifiers

Set of Stored Cases

<table>
<thead>
<tr>
<th>Atr1</th>
<th>………</th>
<th>AtrN</th>
<th>Class</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>A</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>B</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>B</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>C</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>A</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>C</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>B</td>
</tr>
</tbody>
</table>

- Store the training records
- Use training records to predict the class label of unseen cases

Unseen Case

<table>
<thead>
<tr>
<th>Atr1</th>
<th>………</th>
<th>AtrN</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Instance Based Classifiers

- **Examples:**
  - **Rote-learner**
    - Memorizes entire training data and performs classification only if attributes of record match one of the training examples exactly
  
  - **Nearest neighbor**
    - Uses $k$ “closest” points (nearest neighbors) for performing classification
Nearest Neighbor Classifiers

- Basic idea:
  - If it walks like a duck, quacks like a duck, then it’s probably a duck
Nearest-Neighbor Classifiers

- Requires three things
  - The set of stored records
  - Distance Metric to compute distance between records
  - The value of $k$, the number of nearest neighbors to retrieve

- To classify an unknown record:
  - Compute distance to other training records
  - Identify $k$ nearest neighbors
  - Use class labels of nearest neighbors to determine the class label of unknown record (e.g., by taking majority vote)
Definition of Nearest Neighbor

K-nearest neighbors of a record $x$ are data points that have the $k$ smallest distance to $x$.
1 nearest-neighbor

Voronoi Diagram
Nearest Neighbor Classification

- Compute distance between two points:
  - Euclidean distance
  
  \[ d(p, q) = \sqrt{\sum_i (p_i - q_i)^2} \]

- Determine the class from nearest neighbor list
  - take the majority vote of class labels among the k-nearest neighbors
  - Weigh the vote according to distance
    - weight factor, \( w = 1/d^2 \)
Nearest Neighbor Classification...

Choosing the value of k:
- If k is too small, sensitive to noise points
- If k is too large, neighborhood may include points from other classes
Nearest Neighbor Classification...

• Scaling issues
  – Attributes may have to be scaled to prevent distance measures from being dominated by one of the attributes
  – Example:
    ◆ height of a person may vary from 1.5m to 1.8m
    ◆ weight of a person may vary from 90lb to 300lb
    ◆ income of a person may vary from $10K to $1M
Nearest Neighbor Classification...

- Problem with Euclidean measure:
  - High dimensional data
    - curse of dimensionality
  - Can produce counter-intuitive results

\[
\begin{array}{ccccccccccccccc}
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \\
0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1
\end{array}
\quad \quad \quad
\begin{array}{ccccccccccccccc}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1
\end{array}
\]

\[d = 1.4142\]  
\[d = 1.4142\]

- Solution: Normalize the vectors to unit length
Nearest neighbor Classification...

- k-NN classifiers are lazy learners
  - It does not build models explicitly
  - Unlike eager learners such as decision tree induction and rule-based systems
  - Classifying unknown records are relatively expensive
Example: PEBLS

- PEBLS: Parallel Examplar-Based Learning System (Cost & Salzberg)
  - Works with both continuous and nominal features
    - For nominal features, distance between two nominal values is computed using modified value difference metric (MVDM)
  - Each record is assigned a weight factor
  - Number of nearest neighbor, \( k = 1 \)
**Example: PEBLS**

Distance between nominal attribute values:

\[
d_{\text{Single, Married}} = |\frac{2}{4} - \frac{0}{4}| + |\frac{2}{4} - \frac{4}{4}| = 1
\]

\[
d_{\text{Single, Divorced}} = |\frac{2}{4} - \frac{1}{2}| + |\frac{2}{4} - \frac{1}{2}| = 0
\]

\[
d_{\text{Married, Divorced}} = |\frac{0}{4} - \frac{1}{2}| + |\frac{4}{4} - \frac{1}{2}| = 1
\]

\[
d_{\text{Refund=Yes, Refund=No}} = |\frac{0}{3} - \frac{3}{7}| + |\frac{3}{3} - \frac{4}{7}| = \frac{6}{7}
\]

**Table:**

<table>
<thead>
<tr>
<th>Tid</th>
<th>Refund</th>
<th>Marital Status</th>
<th>Taxable Income</th>
<th>Cheat</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Yes</td>
<td>Single</td>
<td>125K</td>
<td>No</td>
</tr>
<tr>
<td>2</td>
<td>No</td>
<td>Married</td>
<td>100K</td>
<td>No</td>
</tr>
<tr>
<td>3</td>
<td>No</td>
<td>Single</td>
<td>70K</td>
<td>No</td>
</tr>
<tr>
<td>4</td>
<td>Yes</td>
<td>Married</td>
<td>120K</td>
<td>No</td>
</tr>
<tr>
<td>5</td>
<td>No</td>
<td>Divorced</td>
<td>95K</td>
<td>Yes</td>
</tr>
<tr>
<td>6</td>
<td>No</td>
<td>Married</td>
<td>60K</td>
<td>No</td>
</tr>
<tr>
<td>7</td>
<td>Yes</td>
<td>Divorced</td>
<td>220K</td>
<td>No</td>
</tr>
<tr>
<td>8</td>
<td>No</td>
<td>Single</td>
<td>85K</td>
<td>Yes</td>
</tr>
<tr>
<td>9</td>
<td>No</td>
<td>Married</td>
<td>75K</td>
<td>No</td>
</tr>
<tr>
<td>10</td>
<td>No</td>
<td>Single</td>
<td>90K</td>
<td>Yes</td>
</tr>
</tbody>
</table>

**Table:**

<table>
<thead>
<tr>
<th>Class</th>
<th>Marital Status</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Single</td>
</tr>
<tr>
<td>Yes</td>
<td>2</td>
</tr>
<tr>
<td>No</td>
<td>2</td>
</tr>
</tbody>
</table>

**Table:**

<table>
<thead>
<tr>
<th>Class</th>
<th>Refund</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Yes</td>
</tr>
<tr>
<td>Yes</td>
<td>0</td>
</tr>
<tr>
<td>No</td>
<td>3</td>
</tr>
</tbody>
</table>

\[
d(V_1, V_2) = \sum_i \left| \frac{n_{1i}}{n_1} - \frac{n_{2i}}{n_2} \right|
\]
Example: PEBLS

<table>
<thead>
<tr>
<th>$Tid$</th>
<th>Refund</th>
<th>Marital Status</th>
<th>Taxable Income</th>
<th>Cheat</th>
</tr>
</thead>
<tbody>
<tr>
<td>$X$</td>
<td>Yes</td>
<td>Single</td>
<td>125K</td>
<td>No</td>
</tr>
<tr>
<td>$Y$</td>
<td>No</td>
<td>Married</td>
<td>100K</td>
<td>No</td>
</tr>
</tbody>
</table>

Distance between record $X$ and record $Y$:

$$\Delta(X, Y) = w_X \cdot w_Y \sum_{i=1}^{d} d(X_i, Y_i)^2$$

where:

$$w_X = \frac{\text{Number of times } X \text{ is used for prediction}}{\text{Number of times } X \text{ predicts correctly}}$$

- $w_X \approx 1$ if $X$ makes accurate prediction most of the time
- $w_X > 1$ if $X$ is not reliable for making predictions
Bayes Classifier

- A probabilistic framework for solving classification problems

  - Conditional Probability:
    \[
    P(C \mid A) = \frac{P(A, C)}{P(A)}
    \]
    \[
    P(A \mid C) = \frac{P(A, C)}{P(C)}
    \]

- Bayes theorem:
  \[
  P(C \mid A) = \frac{P(A \mid C)P(C)}{P(A)}
  \]
Example of Bayes Theorem

Given:
- A doctor knows that meningitis causes stiff neck 50% of the time
- Prior probability of any patient having meningitis is 1/50,000
- Prior probability of any patient having stiff neck is 1/20

If a patient has stiff neck, what’s the probability he/she has meningitis?

\[
P(M \mid S) = \frac{P(S \mid M)P(M)}{P(S)} = \frac{0.5 \times 1/50000}{1/20} = 0.0002
\]
Bayesian Classifiers

- Consider each attribute and class label as random variables

- Given a record with attributes \((A_1, A_2, \ldots, A_n)\)
  - Goal is to predict class \(C\)
  - Specifically, we want to find the value of \(C\) that maximizes \(P(C|A_1, A_2, \ldots, A_n)\)

- Can we estimate \(P(C|A_1, A_2, \ldots, A_n)\) directly from data?
Bayesian Classifiers

- Approach:
  - compute the posterior probability $P(C \mid A_1, A_2, \ldots, A_n)$ for all values of $C$ using the Bayes theorem

\[
P(C \mid A_1 A_2 \ldots A_n) = \frac{P(A_1 A_2 \ldots A_n \mid C)P(C)}{P(A_1 A_2 \ldots A_n)}
\]

- Choose value of $C$ that maximizes $P(C \mid A_1, A_2, \ldots, A_n)$

- Equivalent to choosing value of $C$ that maximizes $P(A_1, A_2, \ldots, A_n | C) P(C)$

- How to estimate $P(A_1, A_2, \ldots, A_n \mid C)$?
Naïve Bayes Classifier

- Assume independence among attributes $A_i$ when class is given:
  - $P(A_1, A_2, \ldots, A_n \mid C) = P(A_1 \mid C_j) P(A_2 \mid C_j) \ldots P(A_n \mid C_j)$

  - Can estimate $P(A_i \mid C_j)$ for all $A_i$ and $C_j$.

  - New point is classified to $C_j$ if $P(C_j) \prod P(A_i \mid C_j)$ is maximal.
How to Estimate Probabilities from Data?

- **Class:** \( P(C) = \frac{N_c}{N} \)
  - e.g., \( P(\text{No}) = \frac{7}{10}, \quad P(\text{Yes}) = \frac{3}{10} \)

- **For discrete attributes:**
  \( P(A_i | C_k) = \frac{|A_{ik}|}{N_{ck}} \)
  - where \(|A_{ik}|\) is number of instances having attribute \(A_i\) and belongs to class \(C_k\)

**Examples:**

\[
P(\text{Status}=\text{Married}|\text{No}) = \frac{4}{7}
\]
\[
P(\text{Refund}=\text{Yes}|\text{Yes})=0
\]

<table>
<thead>
<tr>
<th>Tid</th>
<th>Refund</th>
<th>Marital Status</th>
<th>Taxable Income</th>
<th>Evade</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Yes</td>
<td>Single</td>
<td>125K</td>
<td>No</td>
</tr>
<tr>
<td>2</td>
<td>No</td>
<td>Married</td>
<td>100K</td>
<td>No</td>
</tr>
<tr>
<td>3</td>
<td>No</td>
<td>Single</td>
<td>70K</td>
<td>No</td>
</tr>
<tr>
<td>4</td>
<td>Yes</td>
<td>Married</td>
<td>120K</td>
<td>No</td>
</tr>
<tr>
<td>5</td>
<td>No</td>
<td>Divorced</td>
<td>95K</td>
<td>Yes</td>
</tr>
<tr>
<td>6</td>
<td>No</td>
<td>Married</td>
<td>60K</td>
<td>No</td>
</tr>
<tr>
<td>7</td>
<td>Yes</td>
<td>Divorced</td>
<td>220K</td>
<td>No</td>
</tr>
<tr>
<td>8</td>
<td>No</td>
<td>Single</td>
<td>85K</td>
<td>Yes</td>
</tr>
<tr>
<td>9</td>
<td>No</td>
<td>Married</td>
<td>75K</td>
<td>No</td>
</tr>
<tr>
<td>10</td>
<td>No</td>
<td>Single</td>
<td>90K</td>
<td>Yes</td>
</tr>
</tbody>
</table>
How to Estimate Probabilities from Data?

- For continuous attributes:
  - **Discretize** the range into bins
    - one ordinal attribute per bin
    - violates independence assumption
  - Two-way split: \((A < v)\) or \((A > v)\)
    - choose only one of the two splits as new attribute
  - **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?

- Normal distribution:
  \[
P(A_i \mid c_j) = \frac{1}{\sqrt{2\pi\sigma^2_{ij}}} e^{-\frac{(A_i - \mu_j)^2}{2\sigma^2_{ij}}}
\]
  - One for each \((A_i, c_i)\) pair

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

\[
P(\text{Income} = 120 \mid \text{No}) = \frac{1}{\sqrt{2\pi(54.54)}} e^{-\frac{(120-110)^2}{2(2975)}} = 0.0072
\]
**Example of Naïve Bayes Classifier**

Given a Test Record:

\[ X = (\text{Refund} = \text{No}, \text{Married}, \text{Income} = 120\text{K}) \]

Naive Bayes Classifier:

\[
\begin{align*}
\text{P(Refund=Yes|No)} &= 3/7 \\
\text{P(Refund=No|No)} &= 4/7 \\
\text{P(Refund=Yes|Yes)} &= 0 \\
\text{P(Refund=No|Yes)} &= 1 \\
\text{P(Marital Status=Single|No)} &= 2/7 \\
\text{P(Marital Status=Divorced|No)} &= 1/7 \\
\text{P(Marital Status=Married|No)} &= 4/7 \\
\text{P(Marital Status=Single|Yes)} &= 2/7 \\
\text{P(Marital Status=Divorced|Yes)} &= 1/7 \\
\text{P(Marital Status=Married|Yes)} &= 0
\end{align*}
\]

For taxable income:

If class=No:
- sample mean=110
- sample variance=2975

If class=Yes:
- sample mean=90
- sample variance=25

\[ \begin{align*}
\text{P(X|Class=No)} &= \text{P(Refund=No|Class=No)} \\
&\quad \times \text{P(Married| Class=No)} \\
&\quad \times \text{P(Income=120K| Class=No)} \\
&= 4/7 \times 4/7 \times 0.0072 = 0.0024 \\
\text{P(X|Class=Yes)} &= \text{P(Refund=No| Class=Yes)} \\
&\quad \times \text{P(Married| Class=Yes)} \\
&\quad \times \text{P(Income=120K| Class=Yes)} \\
&= 1 \times 0 \times 1.2 \times 10^{-9} = 0
\end{align*} \]

Since \( P(X|\text{No})P(\text{No}) > P(X|\text{Yes})P(\text{Yes}) \)

Therefore \( P(\text{No}|X) > P(\text{Yes}|X) \)

\( \Rightarrow \) Class = No
Naïve Bayes Classifier

- If one of the conditional probability is zero, then the entire expression becomes zero
- Probability estimation:

  Original: \( P(A_i \mid C) = \frac{N_{ic}}{N_c} \)

  Laplace: \( P(A_i \mid C) = \frac{N_{ic} + 1}{N_c + c} \)

  m-estimate: \( P(A_i \mid C) = \frac{N_{ic} + mp}{N_c + m} \)

  c: number of classes
  p: prior probability
  m: parameter
## Example of Naïve Bayes Classifier

<table>
<thead>
<tr>
<th>Name</th>
<th>Give Birth</th>
<th>Can Fly</th>
<th>Live in Water</th>
<th>Have Legs</th>
<th>Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>human</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>mammals</td>
</tr>
<tr>
<td>python</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>non-mammals</td>
</tr>
<tr>
<td>salmon</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>non-mammals</td>
</tr>
<tr>
<td>whale</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>mammals</td>
</tr>
<tr>
<td>frog</td>
<td>no</td>
<td>no</td>
<td>sometimes</td>
<td>yes</td>
<td>non-mammals</td>
</tr>
<tr>
<td>komodo</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>non-mammals</td>
</tr>
<tr>
<td>bat</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
<td>mammals</td>
</tr>
<tr>
<td>pigeon</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
<td>non-mammals</td>
</tr>
<tr>
<td>cat</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>mammals</td>
</tr>
<tr>
<td>leopard shark</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>non-mammals</td>
</tr>
<tr>
<td>turtle</td>
<td>no</td>
<td>no</td>
<td>sometimes</td>
<td>yes</td>
<td>non-mammals</td>
</tr>
<tr>
<td>penguin</td>
<td>no</td>
<td>no</td>
<td>sometimes</td>
<td>yes</td>
<td>non-mammals</td>
</tr>
<tr>
<td>porcupine</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>mammals</td>
</tr>
<tr>
<td>eel</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>non-mammals</td>
</tr>
<tr>
<td>salamander</td>
<td>no</td>
<td>no</td>
<td>sometimes</td>
<td>yes</td>
<td>non-mammals</td>
</tr>
<tr>
<td>gila monster</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>non-mammals</td>
</tr>
<tr>
<td>platypus</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>mammals</td>
</tr>
<tr>
<td>owl</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
<td>non-mammals</td>
</tr>
<tr>
<td>dolphin</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>mammals</td>
</tr>
<tr>
<td>eagle</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
<td>non-mammals</td>
</tr>
</tbody>
</table>

**A:** attributes  
**M:** mammals  
**N:** non-mammals

\[
P(A | M) = \frac{6}{7} \times \frac{6}{7} \times \frac{2}{7} = 0.06 \\
P(A | N) = \frac{1}{13} \times \frac{10}{13} \times \frac{3}{13} \times \frac{4}{13} = 0.0042 \\
P(A | M)P(M) = 0.06 \times \frac{7}{20} = 0.021 \\
P(A | N)P(N) = 0.004 \times \frac{13}{20} = 0.0027 \\
\]

\[P(A|M)P(M) > P(A|N)P(N)\]

=> Mammals
Naïve Bayes (Summary)

- Robust to isolated noise points
- Handle missing values by ignoring the instance during probability estimate calculations
- Robust to irrelevant attributes
- Independence assumption may not hold for some attributes
  - Use other techniques such as Bayesian Belief Networks (BBN)
Artificial Neural Networks (ANN)

Output $Y$ is 1 if at least two of the three inputs are equal to 1.
Artificial Neural Networks (ANN)

<table>
<thead>
<tr>
<th>X₁</th>
<th>X₂</th>
<th>X₃</th>
<th>Y</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

\[
Y = I(0.3 X_1 + 0.3 X_2 + 0.3 X_3 - 0.4 > 0)
\]

where \( I(z) = \begin{cases} 
1 & \text{if } z \text{ is true} \\
0 & \text{otherwise}
\end{cases} \)
Artificial Neural Networks (ANN)

- Model is an assembly of inter-connected nodes and weighted links.
- Output node sums up each of its input value according to the weights of its links.
- Compare output node against some threshold $t$.

**Perceptron Model**

$$Y = I\left(\sum_i w_i X_i - t\right) \quad \text{or} \quad Y = \text{sign}\left(\sum_i w_i X_i - t\right)$$
General Structure of ANN

Training ANN means learning the weights of the neurons.
Algorithm for learning ANN

• Initialize the weights \((w_0, w_1, \ldots, w_k)\)

• Adjust the weights in such a way that the output of ANN is consistent with class labels of training examples
  
  – Objective function:  
  \[ E = \sum_i [Y_i - f(w_i, X_i)]^2 \]

  – Find the weights \(w_i\)'s that minimize the above objective function
    
    ◆ e.g., backpropagation algorithm (see lecture notes)
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?
Support Vector Machines

- Find hyperplane *maximizes* the margin => B1 is better than B2
Support Vector Machines

\[ \overrightarrow{w} \cdot \overrightarrow{x} + b = 0 \]

\[ \overrightarrow{w} \cdot \overrightarrow{x} + b = -1 \]

\[ f(\overrightarrow{x}) = \begin{cases} 
  1 & \text{if } \overrightarrow{w} \cdot \overrightarrow{x} + b \geq 1 \\
  -1 & \text{if } \overrightarrow{w} \cdot \overrightarrow{x} + b \leq -1 
\end{cases} \]

Margin = \frac{2}{\| \overrightarrow{w} \|^2}
Support Vector Machines

- We want to maximize:  
  \[ \text{Margin} = \frac{2}{\| \mathbf{w} \|^2} \]

- Which is equivalent to minimizing:  
  \[ L(w) = \frac{\| \mathbf{w} \|^2}{2} \]

- But subjected to the following constraints:

  \[ f(\mathbf{x}_i) = \begin{cases} 
  1 & \text{if } \mathbf{w} \cdot \mathbf{x}_i + b \geq 1 \\
  -1 & \text{if } \mathbf{w} \cdot \mathbf{x}_i + b \leq -1 
  \end{cases} \]

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

- What if the problem is not linearly separable?
Support Vector Machines

- What if the problem is not linearly separable?
  - Introduce slack variables
    - Need to minimize:
      \[
      L(w) = \frac{||\vec{w}||^2}{2} + C \left( \sum_{i=1}^{N} \xi_i \right)
      \]
    - Subject to:
      \[
      f(\vec{x}_i) = \begin{cases} 
      1 & \text{if } \vec{w} \cdot \vec{x}_i + b \geq 1 - \xi_i \\
      -1 & \text{if } \vec{w} \cdot \vec{x}_i + b \leq -1 + \xi_i 
      \end{cases}
      \]
Nonlinear Support Vector Machines

- What if decision boundary is not linear?
Nonlinear Support Vector Machines

- Transform data into higher dimensional space
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

Step 1: Create Multiple Data Sets

D_1 → C_1
D_2 → C_2
... → ...
D_{t-1} → C_{t-1}
D_t → C_t

Step 2: Build Multiple Classifiers

Step 3: Combine Classifiers

C^*
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:

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

- How to generate an ensemble of classifiers?
  - Bagging
  - Boosting
Bagging

- Sampling with replacement

<table>
<thead>
<tr>
<th>Original Data</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bagging (Round 1)</td>
<td>7</td>
<td>8</td>
<td>10</td>
<td>8</td>
<td>2</td>
<td>5</td>
<td>10</td>
<td>10</td>
<td>5</td>
<td>9</td>
</tr>
<tr>
<td>Bagging (Round 2)</td>
<td>1</td>
<td>4</td>
<td>9</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>2</td>
<td>7</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>Bagging (Round 3)</td>
<td>1</td>
<td>8</td>
<td>5</td>
<td>10</td>
<td>5</td>
<td>5</td>
<td>9</td>
<td>6</td>
<td>3</td>
<td>7</td>
</tr>
</tbody>
</table>

- Build classifier on each bootstrap sample

- Each sample has probability \((1 - 1/n)^n\) of being selected
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.

<table>
<thead>
<tr>
<th>Original Data</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boosting (Round 1)</td>
<td>7</td>
<td>3</td>
<td>2</td>
<td>8</td>
<td>7</td>
<td>9</td>
<td>4</td>
<td>10</td>
<td>6</td>
<td>3</td>
</tr>
<tr>
<td>Boosting (Round 2)</td>
<td>5</td>
<td>4</td>
<td>9</td>
<td>4</td>
<td>2</td>
<td>5</td>
<td>1</td>
<td>7</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>Boosting (Round 3)</td>
<td>4</td>
<td>4</td>
<td>8</td>
<td>10</td>
<td>4</td>
<td>5</td>
<td>4</td>
<td>6</td>
<td>3</td>
<td>4</td>
</tr>
</tbody>
</table>

- 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:

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

- 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:

\[
C^{*}(x) = \arg \max_{y} \sum_{j=1}^{T} \alpha_{j} \delta(C_{j}(x) = y)
\]
Illustrating AdaBoost

Original Data

Initial weights for each data point

Data points for training

Boosting Round 1

\( \alpha = 1.9459 \)
Illustrating AdaBoost

Boosting Round 1

Boosting Round 2

Boosting Round 3

Overall

\[ \alpha = 1.9459 \]

\[ \alpha = 2.9323 \]

\[ \alpha = 3.8744 \]