Classification

k-nearest neighbor classifier
Naïve Bayes
Logistic Regression
Support Vector Machines
NEAREST NEIGHBOR CLASSIFICATION
Illustrating Classification Task

<table>
<thead>
<tr>
<th>Tid</th>
<th>Attrib1</th>
<th>Attrib2</th>
<th>Attrib3</th>
<th>Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Yes</td>
<td>Large</td>
<td>125K</td>
<td>No</td>
</tr>
<tr>
<td>2</td>
<td>No</td>
<td>Medium</td>
<td>100K</td>
<td>No</td>
</tr>
<tr>
<td>3</td>
<td>No</td>
<td>Small</td>
<td>70K</td>
<td>No</td>
</tr>
<tr>
<td>4</td>
<td>Yes</td>
<td>Medium</td>
<td>120K</td>
<td>No</td>
</tr>
<tr>
<td>5</td>
<td>No</td>
<td>Large</td>
<td>95K</td>
<td>Yes</td>
</tr>
<tr>
<td>6</td>
<td>No</td>
<td>Medium</td>
<td>60K</td>
<td>No</td>
</tr>
<tr>
<td>7</td>
<td>Yes</td>
<td>Large</td>
<td>220K</td>
<td>No</td>
</tr>
<tr>
<td>8</td>
<td>No</td>
<td>Small</td>
<td>85K</td>
<td>Yes</td>
</tr>
<tr>
<td>9</td>
<td>No</td>
<td>Medium</td>
<td>75K</td>
<td>No</td>
</tr>
<tr>
<td>10</td>
<td>No</td>
<td>Small</td>
<td>90K</td>
<td>Yes</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Tid</th>
<th>Attrib1</th>
<th>Attrib2</th>
<th>Attrib3</th>
<th>Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>11</td>
<td>No</td>
<td>Small</td>
<td>55K</td>
<td>?</td>
</tr>
<tr>
<td>12</td>
<td>Yes</td>
<td>Medium</td>
<td>80K</td>
<td>?</td>
</tr>
<tr>
<td>13</td>
<td>Yes</td>
<td>Large</td>
<td>110K</td>
<td>?</td>
</tr>
<tr>
<td>14</td>
<td>No</td>
<td>Small</td>
<td>95K</td>
<td>?</td>
</tr>
<tr>
<td>15</td>
<td>No</td>
<td>Large</td>
<td>67K</td>
<td>?</td>
</tr>
</tbody>
</table>
## 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>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<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 classifier**
    - 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:
  1. **Compute distance** to other training records
  2. Identify \( k \) nearest neighbors
  3. Use class labels of nearest neighbors to determine the class label of unknown record (e.g., by taking majority vote)
Definition of Nearest Neighbor

(a) 1-nearest neighbor (b) 2-nearest neighbor (c) 3-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 defines the classification boundary

The area takes the class of the green point
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

\[
\begin{array}{cccccccccccccccc}
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
\end{array}
\]
\[d = 1.4142\]

\[
\begin{array}{cccccccccccccccc}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
\end{array}
\]
\[d = 1.4142\]

- 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 trees

• Classifying unknown records are relatively expensive
  • Naïve algorithm: O(n)
  • Need for structures to retrieve nearest neighbors fast.
    • The **Nearest Neighbor Search** problem.
Nearest Neighbor Search

- Two-dimensional \textit{kd-trees}
  - A data structure for answering nearest neighbor queries in $\mathbb{R}^2$

- \textit{kd-tree} construction algorithm
  - Select the $x$ or $y$ dimension (alternating between the two)
  - Partition the space into two with a line passing from the median point
  - Repeat recursively in the two partitions as long as there are enough points
Nearest Neighbor Search

2-dimensional kd-trees
Nearest Neighbor Search

2-dimensional \( kd \)-trees
Nearest Neighbor Search

2-dimensional kd-trees
Nearest Neighbor Search

2-dimensional kd-trees
Nearest Neighbor Search

2-dimensional kd-trees
Nearest Neighbor Search

2-dimensional kd-trees
Nearest Neighbor Search

2-dimensional kd-trees

region(u) – all the black points in the subtree of u
Nearest Neighbor Search

2-dimensional \( \text{kd-trees} \)

- A binary tree:
  - Size \( \mathcal{O}(n) \)
  - Depth \( \mathcal{O}(\log n) \)
  - Construction time \( \mathcal{O}(n \log n) \)
  - Query time: worst case \( \mathcal{O}(n) \), but for many cases \( \mathcal{O}(\log n) \)

Generalizes to \( d \) dimensions

- Example of Binary Space Partitioning
SUPPORT VECTOR MACHINES
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

\[ \tilde{w} \cdot \tilde{x} + b = 0 \]
\[ \tilde{w} \cdot \tilde{x} + b = -1 \]
\[ \tilde{w} \cdot \tilde{x} + b = +1 \]

\[
\begin{align*}
\text{Margin} &= \frac{2}{\| \tilde{w} \|} \\
\end{align*}
\]

\[
\begin{cases}
1 & \text{if } \tilde{w} \cdot \tilde{x} + b \geq 1 \\
-1 & \text{if } \tilde{w} \cdot \tilde{x} + b \leq -1 
\end{cases}
\]
Support Vector Machines

- We want to maximize: \( \text{Margin} = \frac{2}{\| \vec{w} \|^2} \)

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

- But subjected to the following constraints:

  \[
  \vec{w} \cdot \vec{x}_i + b \geq 1 \text{ if } y_i = 1 \\
  \vec{w} \cdot \vec{x}_i + b \leq -1 \text{ if } y_i = -1
  \]

- 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?
Support Vector Machines

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

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

- Transform data into higher dimensional space

Use the Kernel Trick
LOGISTIC REGRESSION
Classification via regression

- Instead of predicting the class of an record we want to predict the probability of the class given the record.
- The problem of predicting continuous values is called regression problem.
- General approach: find a continuous function that models the continuous points.
Example: Linear regression

• Given a dataset of the form \{\( (x_1, y_1), \ldots, (x_n, y_n) \)\} find a linear function that given the vector \( x_i \) predicts the \( y_i \) value as \( y_i' = w^T x_i \)
  
  • Find a vector of weights \( w \) that minimizes the sum of square errors
    \[
    \sum_i (y_i' - y_i)^2
    \]
  
  • Several techniques for solving the problem.
Classification via regression

• Assume a linear classification boundary

For the positive class the bigger the value of $w \cdot x$, the further the point is from the classification boundary, the higher our certainty for the membership to the positive class

• Define $P(C_+|x)$ as an increasing function of $w \cdot x$

For the negative class the smaller the value of $w \cdot x$, the further the point is from the classification boundary, the higher our certainty for the membership to the negative class

• Define $P(C_-|x)$ as a decreasing function of $w \cdot x$
Logistic Regression

The **logistic function**

\[ f(t) = \frac{1}{1 + e^{-t}} \]

\[ P(C_+ | x) = \frac{1}{1 + e^{-w \cdot x}} \]

\[ P(C_- | x) = \frac{e^{-w \cdot x}}{1 + e^{-w \cdot x}} \]

\[ \log \frac{P(C_+ | x)}{P(C_- | x)} = w \cdot x \]

Linear regression on the log-odds ratio

**Logistic Regression**: Find the vector \( w \) that maximizes the probability of the observed data
Logistic Regression

• Produces a probability estimate for the class membership which is often very useful.
• The weights can be useful for understanding the feature importance.
• Works for relatively large datasets
• Fast to apply.
NAÏVE BAYES CLASSIFIER
Bayes Classifier

- A probabilistic framework for solving classification problems
- A, C random variables
- Joint probability: \( \Pr(A=a, C=c) \)
- Conditional probability: \( \Pr(C=c \mid A=a) \)
- Relationship between joint and conditional probability distributions

\[
\Pr(C, A) = \Pr(C \mid A) \times \Pr(A) = \Pr(A \mid C) \times \Pr(C)
\]

- Bayes Theorem:

\[
P(C \mid A) = \frac{P(A \mid C)P(C)}{P(A)}
\]
Bayesian Classifiers

- Consider each attribute and class label as random variables

<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>

**Evade C**
Event space: \{Yes, No\}
P(C) = (0.3, 0.7)

**Refund A_1**
Event space: \{Yes, No\}
P(A_1) = (0.3,0.7)

**Martial Status A_2**
Event space: \{Single, Married, Divorced\}
P(A_2) = (0.4,0.4,0.2)

**Taxable Income A_3**
Event space: \(\mathbb{R}\)
P(A_3) \sim \text{Normal}(\mu,\sigma)
Bayesian Classifiers

• Given a record $X$ over attributes $(A_1, A_2, \ldots, A_n)$
  • E.g., $X = ('Yes', 'Single', 125K)$

• The goal is to predict class $C$
  • Specifically, we want to find the value $c$ of $C$ that maximizes $P(C=c| X)$
    • Maximum Aposteriori Probability estimate

• Can we estimate $P(C| X)$ directly from data?
  • This means that we estimate the probability for all possible values of the class variable.
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 $P(A_1, A_2, \ldots, A_n \mid C) \cdot P(C)$

• 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:
  
  \[ P(A_1, A_2, ..., A_n | C) = P(A_1 | C) \cdot P(A_2 | C) \cdot ... \cdot P(A_n | C) \]

• We can estimate $P(A_i | C)$ for all values of $A_i$ and $C$.

• New point $X$ is classified to class $c$ if
  
  \[ P(C = c | X) = P(C = c) \prod_i P(A_i | c) \]

  is maximum over all possible values of $C$. 
How to Estimate Probabilities from Data?

• **Class Prior Probability:**
  \[ P(C = c) = \frac{N_c}{N} \]
  e.g., \( P(C = \text{No}) = \frac{7}{10} \), \( P(C = \text{Yes}) = \frac{3}{10} \)

• **For discrete attributes:**
  \[ P(A_i = a \mid C = c) = \frac{N_{a,c}}{N_c} \]
  where \( N_{a,c} \) is number of instances having attribute \( A_i = a \) and belongs to class \( c \)

  • **Examples:**
    \[
    P(\text{Status}=\text{Married} \mid \text{No}) = \frac{4}{7} \\
    P(\text{Refund}=\text{Yes} \mid \text{Yes}) = 0
    \]
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
    (i.e., **mean** \(\mu\) and **standard deviation** \(\sigma\))
  - Once probability distribution is known, we can use it to estimate the conditional probability \(P(A_i|c)\)
How to Estimate Probabilities from Data?

• Normal distribution:

\[ P(A_i = a \mid c_j) = \frac{1}{\sqrt{2\pi\sigma^2_{ij}}} e^{-\frac{(a-\mu_{ij})^2}{2\sigma^2_{ij}}} \]

• One for each \((a_i, c_i)\) pair

• For \((\text{Income}, \text{Class}=\text{No})\):
  • If \text{Class}=\text{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

Creating a Naïve Bayes Classifier, essentially means to compute **counts**:

Total number of records: \( N = 10 \)

**Class No:**
- Number of records: 7
- **Attribute Refund:**
  - Yes: 3
  - No: 4
- **Attribute Marital Status:**
  - Single: 2
  - Divorced: 1
  - Married: 4
- **Attribute Income:**
  - mean: 110
  - variance: 2975

**Class Yes:**
- Number of records: 3
- **Attribute Refund:**
  - Yes: 0
  - No: 3
- **Attribute Marital Status:**
  - Single: 2
  - Divorced: 1
  - Married: 0
- **Attribute Income:**
  - mean: 90
  - variance: 25
Example of Naïve Bayes Classifier

Given a Test Record:

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

naive Bayes Classifier:

- \[ P(Refund=Yes|\text{No}) = \frac{3}{7} \]
- \[ P(Refund=No|\text{No}) = \frac{4}{7} \]
- \[ P(Refund=Yes|\text{Yes}) = 0 \]
- \[ P(Refund=No|\text{Yes}) = 1 \]

- \[ P(Marital \text{ Status}=Single|\text{No}) = \frac{2}{7} \]
- \[ P(Marital \text{ Status}=Divorced|\text{No}) = \frac{1}{7} \]
- \[ P(Marital \text{ Status}=Married|\text{No}) = \frac{4}{7} \]
- \[ P(Marital \text{ Status}=Single|\text{Yes}) = \frac{2}{7} \]
- \[ P(Marital \text{ Status}=Divorced|\text{Yes}) = \frac{1}{7} \]
- \[ P(Marital \text{ Status}=Married|\text{Yes}) = 0 \]

For taxable income:

If class=\text{No}:
- sample mean=110
- sample variance=2975

If class=\text{Yes}:
- sample mean=90
- sample variance=25

\[ P(\text{No}) = 0.3, \ P(\text{Yes}) = 0.7 \]

Since \[ P(X|\text{No})P(\text{No}) > P(X|\text{Yes})P(\text{Yes}) \]
Therefore \[ P(\text{No}|X) > P(\text{Yes}|X) \]
\[ \Rightarrow \text{Class} = \text{No} \]
Naïve Bayes Classifier

• If one of the conditional probability is zero, then the entire expression becomes zero

• Probability estimation:

Original: \( P(A_i = a \mid C = c) = \frac{N_{ac}}{N_c} \)

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

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

\( N_i \): number of attribute values for attribute \( A_i \)
\( p \): prior probability
\( m \): parameter
Example of Naïve Bayes Classifier

Given a Test Record:

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

naive Bayes Classifier:

\[
\begin{align*}
\text{P(Refund=Yes | No)} &= 4/9 \\
\text{P(Refund=No | No)} &= 5/9 \\
\text{P(Refund=Yes | Yes)} &= 1/5 \\
\text{P(Refund=No | Yes)} &= 4/5 \\
\text{P(Marital Status=Single | No)} &= 3/10 \\
\text{P(Marital Status=Divorced | No)} &= 2/10 \\
\text{P(Marital Status=Married | No)} &= 5/10 \\
\text{P(Marital Status=Single | Yes)} &= 3/6 \\
\text{P(Marital Status=Divorced | Yes)} &= 2/6 \\
\text{P(Marital Status=Married | Yes)} &= 1/6 \\
\end{align*}
\]

For taxable income:

If class=No: sample mean=110 \\
    sample variance=2975 \\
If class=Yes: sample mean=90 \\
    sample variance=25 \\

\[
\begin{align*}
\text{P(X|Class=No)} &= \text{P(Refund=No|Class=No)} \times \text{P(Married| Class=No)} \times \text{P(Income=120K| Class=No)} \\
&= 5/9 \times 5/10 \times 0.0072 \\
\text{P(X|Class=Yes)} &= \text{P(Refund=No| Class=Yes)} \times \text{P(Married| Class=Yes)} \times \text{P(Income=120K| Class=Yes)} \\
&= 4/5 \times 1/6 \times 1.2 \times 10^{-9} \\
\end{align*}
\]

\[
\begin{align*}
\text{P(No)} &= 0.7, \quad \text{P(Yes)} = 0.3 \\
\text{Since P(X|No)P(No) > P(X|Yes)P(Yes)} \\
\text{Therefore P(No|X) > P(Yes|X)} \\
\Rightarrow \text{Class} = \text{No}
\end{align*}
\]
Implementation details

- Computing the conditional probabilities involves multiplication of many very small numbers
  - Numbers get very close to zero, and there is a danger of numeric instability
- We can deal with this by computing the logarithm of the conditional probability

\[
\log P(C|A) \sim \log P(A|C) + \log P(A) \\
= \sum_i \log(A_i|C) + \log P(A)
\]
Naïve Bayes for Text Classification

- Naïve Bayes is commonly used for text classification
- For a document $d = (t_1, \ldots, t_k)$

$$P(c|d) = P(c) \prod_{t_i \in d} P(t_i|c)$$

- $P(t_i|c) =$ Fraction of terms from all documents in $c$ that are $t_i$.

- Easy to implement and works relatively well
- Limitation: Hard to incorporate additional features (beyond words).
TRAINMULTINOMIALNB(C, D)
  1 $V \leftarrow$ EXTRACTVOCABULARY(D)
  2 $N \leftarrow$ COUNTDOCS(D)
  3 for each $c \in C$
  4 do $N_c \leftarrow$ COUNTDOCSINCLASS(D, c)
  5 prior[$c$] $\leftarrow$ $N_c / N$
  6 $text_c \leftarrow$ CONCATENATETEXTOFALLDOSCSINCLASS(D, c)
  7 for each $t \in V$
  8 do $T_{ct} \leftarrow$ COUNTTOKENSOFTERM(text$_c$, t)
  9 for each $t \in V$
 10 do condprob[t][c] $\leftarrow$ $\frac{T_{ct} + 1}{\sum_{t'}(T_{ct'} + 1)}$
 11 return $V$, prior, cond prob

APPLYMULTINOMIALNB(C, V, prior, condprob, d)
  1 $W \leftarrow$ EXTRACTTOKENSFROMDOC(V, d)
  2 for each $c \in C$
  3 do score[c] $\leftarrow$ log prior[c]
  4 for each $t \in W$
  5 do score[c] $+=$ log condprob[t][c]
  6 return arg max$_{c \in C}$ score[c]

Figure 13.2  Naive Bayes algorithm (multinomial model): Training and testing.
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)

- Naïve Bayes can produce a probability estimate, but it is usually a very biased one
  - Logistic Regression is better for obtaining probabilities.
Generative vs Discriminative models

- Naïve Bayes is a type of a generative model
  - Generative process:
    - First pick the category of the record
    - Then given the category, generate the attribute values from the distribution of the category
  - Conditional independence given C

- We use the training data to learn the distribution of the values in a class
Generative vs Discriminative models

- Logistic Regression and SVM are **discriminative** models
  - The goal is to find the boundary that discriminates between the two classes from the training data

- In order to classify the language of a document, you can
  - Either learn the two languages and find which is more likely to have generated the words you see
  - Or learn what differentiates the two languages.