Learning probabilistic models
Chapter 20
Bayesian learning: Bayes’ rule

- Given some model space (set of hypotheses $h_i$) and evidence (data $D$):
  - $P(h_i|D) = \alpha P(D|h_i) P(h_i)$

- We assume that observations are independent of each other, given a model (hypothesis), so:
  - $P(h_i|D) = \alpha \prod_j P(d_j|h_i) P(h_i)$

- To predict the value of some unknown quantity, $X$ (e.g., the class label for a future observation):
  - $P(X|D) = \sum_i P(X|D, h_i) P(h_i|D) = \sum_i P(X|h_i) P(h_i|D)$

These are equal by our independence assumption
Bayesian learning

We can apply Bayesian learning in three basic ways:

- **BMA (Bayesian Model Averaging):** Don’t just choose one hypothesis; instead, make predictions based on the weighted average of all hypotheses (or some set of best hypotheses).

- **MAP (Maximum A Posteriori) hypothesis:** Choose the hypothesis with the highest \textit{a posteriori} probability, given the data.

- **MLE (Maximum Likelihood Estimate):** Assume that all hypotheses are equally likely \textit{a priori}; then the best hypothesis is just the one that maximizes the likelihood (i.e., the probability of the data given the hypothesis).

- **MDL (Minimum Description Length) principle:** Use some encoding to model the complexity of the hypothesis, and the fit of the data to the hypothesis, then minimize the overall description of $h_i + D$. 
Naïve Bayes
Naïve Bayes

- Use Bayesian modeling
- Make the simplest possible independence assumption:
  - Each attribute is independent of the values of the other attributes, given the class variable
  - In the restaurant domain: Cuisine is independent of Patrons, \textit{given} a decision to stay (or not)
Bayesian Formulation

- \[ p(C \mid F_1, \ldots, F_n) = \frac{p(C) \ p(F_1, \ldots, F_n \mid C)}{P(F_1, \ldots, F_n)} \]
  \[ = \alpha \ p(C) \ p(F_1, \ldots, F_n \mid C) \]

- Naïve Bayes assumption
  - Assume that each feature \( F_i \) is \textit{conditionally independent of the other features given the class} \( C \).

- Then we have:
  \[ p(C \mid F_1, \ldots, F_n) = \alpha \ p(C) \ \Pi_i \ p(F_i \mid C) \]
  - remember that \( \alpha \) is a normalization factor

- We can estimate each of these conditional probabilities from the observed counts in the training data:
  \[ p(F_i \mid C) = \frac{\#(F_i \land C)}{\#(C)} \]
### Restaurant example (training set)

<table>
<thead>
<tr>
<th>Example</th>
<th>Attributes</th>
<th>Target</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Alt Bar Fri Hun Pat Price Rain Res Type Est Wait</td>
<td></td>
</tr>
<tr>
<td>$X_1$</td>
<td>T F F T Some $$$$ F T French 0–10 T</td>
<td></td>
</tr>
<tr>
<td>$X_2$</td>
<td>T F F T Full $ F F Thai 30–60 F</td>
<td></td>
</tr>
<tr>
<td>$X_3$</td>
<td>F T F F Some $ F F Burger 0–10 T</td>
<td></td>
</tr>
<tr>
<td>$X_4$</td>
<td>T F T T Full $ F F Thai 10–30 T</td>
<td></td>
</tr>
<tr>
<td>$X_5$</td>
<td>T F T F Full $$$$ F T French $&gt;$60 F</td>
<td></td>
</tr>
<tr>
<td>$X_6$</td>
<td>F T F T Some $$ T T Italian 0–10 T</td>
<td></td>
</tr>
<tr>
<td>$X_7$</td>
<td>F T F F None $ T F Burger 0–10 F</td>
<td></td>
</tr>
<tr>
<td>$X_8$</td>
<td>F F F T Some $$ T T Thai 0–10 T</td>
<td></td>
</tr>
<tr>
<td>$X_9$</td>
<td>F T T F Full $ T F Burger $&gt;$60 F</td>
<td></td>
</tr>
<tr>
<td>$X_{10}$</td>
<td>T T T T Full $$$$ F T Italian 10–30 F</td>
<td></td>
</tr>
<tr>
<td>$X_{11}$</td>
<td>F F F F None $ F F Thai 0–10 F</td>
<td></td>
</tr>
<tr>
<td>$X_{12}$</td>
<td>T T T T Full $ F F Burger 30–60 T</td>
<td></td>
</tr>
</tbody>
</table>

Naive Bayes: Example

- \( p(Wait \mid Cuisine, Patrons, Rain) = \)
  \[ \alpha \ p(Wait) \ p(Cuisine \mid Wait) \ p(Patrons \mid Wait) \ p(Rain \mid Wait) \]
  - remember that \( \alpha \) is a normalization factor

- \( P(Rain \mid Wait) = \#(Wait \land Rain) / \#(Rain) \)
- \( P(Patrons \mid Wait) = \#(Patrons \land Rain) / \#(Rain) \)

- Substitute \( Wait \) for \( Wait=T \) and \( Wait=F \)
- This gives us the actual probabilities (of \( Wait=T \) and \( Wait=F \))
Assume target function $f: X \rightarrow V$, where each instance (example) $x$ is described by attributes $<a_1, a_2, \ldots, a_n>$.

The most probable value of $f(x)$ is:

\[
\begin{align*}
 v_{MAP} &= \underset{v_j \in V}{\text{argmax}} \ P(v_j|a_1, a_2 \ldots a_n) \\
 v_{MAP} &= \underset{v_j \in V}{\text{argmax}} \ \frac{P(a_1, a_2 \ldots a_n|v_j)P(v_j)}{P(a_1, a_2 \ldots a_n)} \\
 &= \underset{v_j \in V}{\text{argmax}} \ P(a_1, a_2 \ldots a_n|v_j)P(v_j)
\end{align*}
\]

Naive Bayes assumption:

\[P(a_1, a_2 \ldots a_n|v_j) = \prod_i P(a_i|v_j)\]

which gives

**Naive Bayes classifier**: \[v_{NB} = \underset{v_j \in V}{\text{argmax}} \ P(v_j) \prod_i P(a_i|v_j)\]
Naïve Bayes Classifier

Naive Bayes classifier:  

\[ v_{NB} = \arg \max_{v_j \in V} P(v_j) \prod_{i} P(a_i | v_j) \]

- \( v_{NB} \): Naïve Bayes value
  - The value returned by the Naïve Bayes classifier (T or F for the restaurant example)

- Remember that we can estimate each of these conditional probabilities from the observed counts in the training data:

\[ p(a_i \mid v_j) = \#(a_i \land v_j) / \#(v_j) \]
Naïve Bayes Learning

- To learn from the examples, we estimate the probabilities from the observed counts in the training data:

\[
\text{Naive Bayes Learn}(examples)
\]

For each target value \( v_j \)

\[
\hat{P}(v_j) \leftarrow \text{estimate } P(v_j)
\]

For each attribute value \( a_i \) of each attribute \( a \)

\[
\hat{P}(a_i|v_j) \leftarrow \text{estimate } P(a_i|v_j)
\]

\[
\text{Classify New Instance}(x)
\]

\[
v_{NB} = \arg\max_{v_j \in V} \hat{P}(v_j) \prod_{a_i \in x} \hat{P}(a_i|v_j)
\]
Learning to classify text

Why?

- Learn which news articles are of interest
- Learn to classify web pages by topic

Naive Bayes is among most effective algorithms

What attributes shall we use to represent text documents??
Target concept *Interesting?* : Document $\rightarrow \{+, -\}$

1. Represent each document by vector of words
   - one attribute per word position in document

2. Learning: Use training examples to estimate
   - $P(+)$
   - $P(-)$
   - $P(doc|+)$
   - $P(doc|-)$

Naive Bayes conditional independence assumption

$$P(doc|v_j) = \prod_{i=1}^{\text{length}(doc)} P(a_i = w_k|v_j)$$

where $P(a_i = w_k|v_j)$ is probability that word in position $i$ is $w_k$, given $v_j$
Learning to classify text

\textbf{Classify\_naive\_Bayes\_text}(\textit{Doc})

- \textit{positions} $\leftarrow$ all word positions in \textit{Doc} that contain tokens found in \textit{Vocabulary}
- Return $v_{NB}$, where

$$v_{NB} = \arg\max_{v_j \in V} P(v_j) \prod_{i \in \text{positions}} P(a_i | v_j)$$
Learning to classify text

\textsc{learn\_naive\_bayes\_text}(Examples, V)

1. collect all words and other tokens that occur in Examples

- Vocabulary $\leftarrow$ all distinct words and other tokens in Examples

2. calculate the required $P(v_j)$ and $P(w_k|v_j)$ probability terms

   For each target value $v_j$ in $V$ do

   - $docs_j \leftarrow$ subset of Examples for which the target value is $v_j$
   - $P(v_j) \leftarrow \frac{|docs_j|}{|Examples|}$
   - $Text_j \leftarrow$ a single document created by concatenating all members of $docs_j$
   - $n \leftarrow$ total number of words in $Text_j$ (counting duplicate words multiple times)
   - for each word $w_k$ in Vocabulary
     * $n_k \leftarrow$ number of times word $w_k$ occurs in $Text_j$
     * $P(w_k|v_j) \leftarrow \frac{n_k+1}{n+|Vocabulary|}$
Naive Bayes: Analysis

- Naive Bayes is amazingly easy to implement (once you understand the bit of math behind it)
- Remarkably, naive Bayes can outperform many much more complex algorithms—it’s a baseline that should pretty much always be used for comparison
- Naive Bayes can’t capture interdependencies between variables (obviously)—for that, we need Bayes nets!
Naïve Bayes in practice

- **When to use**
  - Moderate or large training set available
  - Attributes that describe instances are conditionally independent given classification

- **Successful applications**
  - Diagnosis
  - Classifying text documents
  - Detecting spam email
Exercise:

Play Tennis example again!

- We to use NaiveBayes to decide whether the weather is amenable to playing tennis. Over the course of 2 weeks, data is collected to help ID3 build a decision tree.

- The target (binary) classification is
  - "should we play PlayTennis?" which can be Yes or No

- The weather attributes are outlook, temperature, humidity, and wind. They can have the following values:
  - Outlook = { sunny, overcast, rain }
  - Temperature = { hot, mild, cool }
  - Humidity = { high, normal }
  - Wind = { weak, strong }
# Training examples for the target concept PlayTennis

<table>
<thead>
<tr>
<th>Day</th>
<th>Outlook</th>
<th>Temperature</th>
<th>Humidity</th>
<th>Wind</th>
<th>PlayTennis</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Sunny</td>
<td>Hot</td>
<td>High</td>
<td>Weak</td>
<td>No</td>
</tr>
<tr>
<td>2</td>
<td>Sunny</td>
<td>Hot</td>
<td>High</td>
<td>Strong</td>
<td>No</td>
</tr>
<tr>
<td>3</td>
<td>Overcast</td>
<td>Hot</td>
<td>High</td>
<td>Weak</td>
<td>Yes</td>
</tr>
<tr>
<td>4</td>
<td>Rain</td>
<td>Mild</td>
<td>High</td>
<td>Weak</td>
<td>Yes</td>
</tr>
<tr>
<td>5</td>
<td>Rain</td>
<td>Cool</td>
<td>Normal</td>
<td>Weak</td>
<td>Yes</td>
</tr>
<tr>
<td>6</td>
<td>Rain</td>
<td>Cool</td>
<td>Normal</td>
<td>Strong</td>
<td>No</td>
</tr>
<tr>
<td>7</td>
<td>Overcast</td>
<td>Cool</td>
<td>Normal</td>
<td>Strong</td>
<td>Yes</td>
</tr>
<tr>
<td>8</td>
<td>Sunny</td>
<td>Mild</td>
<td>High</td>
<td>Weak</td>
<td>No</td>
</tr>
<tr>
<td>9</td>
<td>Sunny</td>
<td>Cool</td>
<td>Normal</td>
<td>Weak</td>
<td>Yes</td>
</tr>
<tr>
<td>10</td>
<td>Rain</td>
<td>Mild</td>
<td>Normal</td>
<td>Weak</td>
<td>Yes</td>
</tr>
<tr>
<td>11</td>
<td>Sunny</td>
<td>Mild</td>
<td>Normal</td>
<td>Strong</td>
<td>Yes</td>
</tr>
<tr>
<td>12</td>
<td>Overcast</td>
<td>Mild</td>
<td>High</td>
<td>Strong</td>
<td>Yes</td>
</tr>
<tr>
<td>13</td>
<td>Overcast</td>
<td>Hot</td>
<td>Normal</td>
<td>Weak</td>
<td>Yes</td>
</tr>
<tr>
<td>14</td>
<td>Rain</td>
<td>Mild</td>
<td>High</td>
<td>Strong</td>
<td>No</td>
</tr>
</tbody>
</table>
Exercise

- Consider a new instance (observation):
  
  <Outlook=sunny, Temperature=cool, Humidity=high, Wind=strong>

- We want to compute ($v_{NB}$: Naïve Bayes value):
  
  $$v_{NB} = \arg\max_{v_j \in V} P(v_j) \prod_{i} P(a_i|v_j)$$

  - where $V = \{y, n\}$ and the $a_i$’s are the values for each attribute given in the observation vector (<Outlook=sunny, Temperature=cool, Humidity=high, Wind=strong> )

- That is, the max of:

  - $P(y) P(sun|y) P(cool|y) P(high|y) P(strong|y) :$
  
  - $P(n) P(sun|n) P(cool|n) P(high|n) P(strong|n)$
Exercise

- \( P(\text{strong} \mid y) = \frac{3}{9} = .33 \)
  - Remember this is abbrev. for \( P(\text{Wind}=\text{strong} \mid \text{PlayTennis}=\text{yes}) \)
  - Also remember that \( p(a_i \mid v_j) = \frac{\#(a_i \land v_j)}{\#(v_j)} \)
  - Then, we have: \( p(\text{strong} \mid y) = \frac{\#(\text{strong} \land y)}{\#(y)} = \frac{3}{9} \)

- \( P(\text{strong} \mid n) = \frac{3}{5} = .60 \)

- Therefore: \( P(y) P(\text{sun} \mid y) P(\text{cool} \mid y) P(\text{high} \mid y) P(\text{strong} \mid y) = .005 \)
  \[ P(n) P(\text{sun} \mid n) P(\text{cool} \mid n) P(\text{high} \mid n) P(\text{strong} \mid n) = .021 \]

- So, the Naïve Bayes value \((v_{NB})\) is:
  - \textit{Playtennis=\textbf{no}}
  - What is its probability?
  \[
  \frac{.021}{.021+.005} = .795
  \]

\[
\alpha P(v_j) \prod_i P(a_i \mid v_j)
\]
Learning Bayesian Networks
Learning Bayesian networks

- Given training set
- Find B that best matches $D$
  - Learn the structure - *model selection*
  - Structure is given, learn the conditional probabilities - *parameter estimation*

![Diagram of Bayesian network]

Data D

MCMAMBMECAE
Learning Bayesian networks

- **Known structure, fully observable**: only need to do parameter estimation
- **Unknown structure, fully observable**: do heuristic search through structure space, then parameter estimation
- **Known structure, missing values**: use expectation maximization (EM) to estimate parameters
- **Known structure, hidden variables**: apply adaptive probabilistic network (APN) techniques
- **Unknown structure, hidden variables**: too hard to solve!
Learning Bayesian networks

- **Known structure, fully observable**: only need to do parameter estimation
- **Unknown structure, fully observable**: do heuristic search through structure space, then parameter estimation
- **Known structure, missing values**: use expectation maximization (EM) to estimate parameters
- **Known structure, hidden variables**: apply adaptive probabilistic network (APN) techniques
- **Unknown structure, hidden variables**: too hard to solve!
Parameter estimation

- **Known structure, fully observable**: only need to do parameter estimation
Parameter estimation

- Assume known structure
- Goal: estimate BN parameters $\theta$
  - entries in local probability models, $P(X \mid \text{Parents}(X))$
- A parameterization $\theta$ is good if it is likely to generate the observed data:

$$L(\theta : D) = P(D \mid \theta) = \prod_{m} P(d_m \mid \theta)$$

- Maximum Likelihood Estimation (MLE) Principle: Choose $\theta^*$ so as to maximize $L$
The likelihood **decomposes** according to the structure of the network

→ we get a separate estimation task for each parameter

The MLE (maximum likelihood estimate) solution:

- for each value $x$ of a node $X$
- and each instantiation $u$ of $Parents(X)$

\[
\theta_{x|u}^* = \frac{N(x, u)}{N(u)} \quad \text{sufficient statistics}
\]

- Just need to collect the counts for every combination of parents and children observed in the data
- MLE is equivalent to an assumption of a uniform prior over parameter values
Sufficient statistics: Example

\[ \theta^*_{x|u} = \frac{N(x, u)}{N(u)} \]

\[ \theta^*_{A|E, B} = \frac{N(A, E, B)}{N(E, B)} \]
Learning Bayesian networks

- **Known structure, fully observable**: only need to do parameter estimation
- **Unknown structure, fully observable**: do heuristic search through structure space, then parameter estimation
- **Known structure, missing values**: use expectation maximization (EM) to estimate parameters
- **Known structure, hidden variables**: apply adaptive probabilistic network (APN) techniques
- **Unknown structure, hidden variables**: too hard to solve!
Model selection

- **Unknown structure, fully observable:** do heuristic search through structure space, then parameter estimation

**Goal:** Select the best network structure, given the data

**Input:**
- Training data
- Scoring function

**Output:**
- A network that maximizes the score
Structure selection: Scoring

- Bayesian: prior over parameters and structure
  - get balance between model complexity and fit to data as a byproduct

- Score \((G:D) = \log P(G|D) \propto \log [P(D|G) P(G)]\)

- Marginal likelihood just comes from our parameter estimates

- Prior on structure can be any measure we want; typically a function of the network complexity

**Same key property: Decomposability**

\[
\text{Score(structure)} = \sum_i \text{Score(family of } X_i) \]

Heuristic search

- Add $E \rightarrow C$
- $\Delta \text{score}(C)$

- Reverse $E \rightarrow A$
- $\Delta \text{score}(A, E)$

- Delete $E \rightarrow A$
- $\Delta \text{score}(A)$
Exploiting decomposability

To recompute scores, only need to re-score families that changed in the last move
Learning Bayesian networks

- **Known structure, fully observable:** only need to do parameter estimation
- **Unknown structure, fully observable:** do heuristic search through structure space, then parameter estimation
- **Known structure, missing values:** use expectation maximization (EM) to estimate parameters
- **Known structure, hidden variables:** apply adaptive probabilistic network (APN) techniques
- **Unknown structure, hidden variables:** too hard to solve!
Handling missing data

- Suppose that in some cases, we observe earthquake, alarm, light-level, and moon-phase, but not burglary
- Should we throw that data away??
- **Idea**: Guess the missing values based on the other data
EM (expectation maximization)

- **Guess** probabilities for nodes with missing values (e.g., based on other observations)
- **Compute the probability distribution** over the missing values, given our guess
- **Update the probabilities** based on the guessed values
- **Repeat** until convergence
Suppose we have observed Earthquake and Alarm but not Burglary for an observation on November 27.

We estimate the CPTs based on the rest of the data.

We then estimate $P(\text{Burglary})$ for November 27 from those CPTs.

Now we recompute the CPTs as if that estimated value had been observed.

Repeat until convergence!
Unsupervised Learning: Clustering
Unsupervised learning

- Learn without a “supervisor” who labels instances
  - Clustering
  - Scientific discovery
  - Pattern discovery
  - Associative learning

- Clustering:
  - Given a set of instances without labels, partition them such that each instance is:
    - similar to other instances in its partition (intra-cluster similarity)
    - dissimilar from instances in other partitions (inter-cluster dissimilarity)
Clustering techniques

- **Agglomerative clustering**
  - Single-link clustering
  - Complete-link clustering
  - Average-link clustering

- **Partitional clustering**
  - k-means clustering

- **Spectral clustering**
Agglomerative clustering

- **Agglomerative:**
  - Start with each instance in a cluster by itself
  - Repeatedly combine pairs of clusters until some stopping criterion is reached (or until one “super-cluster” with substructure is found)
  - Often used for non-fully-connected data sets (e.g., clustering in a social network)

- **Single-link:**
  - At each step, combine the two clusters with the smallest minimum distance between any pair of instances in the two clusters (i.e., find the shortest “edge” between each pair of clusters)

- **Average-link:**
  - Combine the two clusters with the smallest average distance between all pairs of instances

- **Complete-link:**
  - Combine the two clusters with the smallest maximum distance between any pair of instances
**k-Means**

- **Partitional:**
  - Start with all instances in a set, and find the “best” partition

- **k-Means:**
  - Basic idea: use expectation maximization to find the best clusters
  - Objective function: Minimize the within-cluster sum of squared distances
  - Initialize $k$ clusters by choosing $k$ random instances as cluster “centroids” (where $k$ is an input parameter)
  - E-step: Assign each instance to its nearest cluster (using Euclidean distance to the centroid)
  - M-step: Recompute the centroid as the center of mass of the instances in the cluster
  - Repeat until convergence is achieved