Statistical learning

Chapter 20 (plus 18.1-2)
Outline

♦ Forms of Learning
♦ Bayesian learning
♦ Maximum likelihood and linear regression
♦ Expectation Maximization
Learning

Learning is essential for unknown environments, i.e., when designer lacks omniscience.

Learning is useful as a system construction method, i.e., expose the agent to reality rather than trying to write it down.

Learning modifies the agent’s decision mechanisms to improve performance.
Learning agents

Performance standard

Critic

Sensors

Environment

Performance element

Learning element

Problem generator

feedback

changes

learning goals

knowledge

experiments

Effectors

Agent

Chapter 20 (plus 18.1-2)  4
Learning element

Design of learning element is dictated by

♦ what type of performance element is used
♦ which functional component is to be learned
♦ how that functional component is represented
♦ what kind of feedback is available

Example scenarios:

<table>
<thead>
<tr>
<th>Performance element</th>
<th>Component</th>
<th>Representation</th>
<th>Feedback</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alpha–beta search</td>
<td>Eval. fn.</td>
<td>Weighted linear function</td>
<td>Win/loss</td>
</tr>
<tr>
<td>Logical agent</td>
<td>Transition model</td>
<td>Successor–state axioms</td>
<td>Outcome</td>
</tr>
<tr>
<td>Utility–based agent</td>
<td>Transition model</td>
<td>Dynamic Bayes net</td>
<td>Outcome</td>
</tr>
<tr>
<td>Simple reflex agent</td>
<td>Percept–action fn</td>
<td>Neural net</td>
<td>Correct action</td>
</tr>
</tbody>
</table>
Types of learning

**Supervised learning**: learn a function from examples labeled with the correct answers (requires “teacher”)

**Unsupervised learning**: learn patterns in the input when no specific output (or answers) are given (no “teacher”)

**Reinforcement learning**: learn from occasional rewards (harder, but does not require a teacher)
Inductive learning (a.k.a. Science)

Simplest form: learn a function from examples (tabula rasa)

\( f \) is the target function

An example is a pair \( x, f(x) \), e.g.,

\[
\begin{array}{c|c|c}
O & O & X \\
X & X & \ \\
X & & +1 \\
\end{array}
\]

Problem: find a(n) hypothesis \( h \) such that \( h \approx f \)
given a training set of examples

(This is a highly simplified model of real learning:
  - Ignores prior knowledge
  - Assumes a deterministic, observable “environment”
  - Assumes examples are given
  - Assumes that the agent wants to learn \( f \)—why?)
Inductive learning method

Construct/adjust $h$ to agree with $f$ on training set
($h$ is consistent if it agrees with $f$ on all examples)

E.g., curve fitting:

\[ f(x) \]

\[ x \]
Inductive learning method

Construct/adjust $h$ to agree with $f$ on training set
($h$ is consistent if it agrees with $f$ on all examples)

E.g., curve fitting:

\[ f(x) \]
Inductive learning method

Construct/adjust $h$ to agree with $f$ on training set
($h$ is **consistent** if it agrees with $f$ on all examples)

E.g., curve fitting:
Inductive learning method

Construct/adjust $h$ to agree with $f$ on training set
($h$ is consistent if it agrees with $f$ on all examples)

E.g., curve fitting:

\[
f(x)
\]
Inductive learning method

Construct/adjust $h$ to agree with $f$ on training set
($h$ is consistent if it agrees with $f$ on all examples)

E.g., curve fitting:
Inductive learning method

Construct/adjust \( h \) to agree with \( f \) on training set
\((h \text{ is consistent if it agrees with } f \text{ on all examples})\)

E.g., curve fitting:

Ockham’s razor: maximize a combination of consistency and simplicity
Moving on to: Statistical learning

Chapter 20
Full Bayesian learning

(This is a form of unsupervised learning.)

View learning as Bayesian updating of probability distribution over the hypothesis space

Prior \( P(H) \), data evidence given as \( d = d_1, \ldots, d_N \)

Given the data so far, each hypothesis has a posterior probability:

\[
P(h_i|d) = \alpha P(d|h_i) P(h_i)
\]

Predictions use a likelihood-weighted average over the hypotheses:

\[
P(X|d) = \sum_i P(X|d, h_i) P(h_i|d) = \sum_i P(X|h_i) P(h_i|d)
\]

Assume observations are independently and identically distributed (i.e., i.i.d.):

\[
P(d|h_i) = \prod_j P(d_j|h_i)
\]
Suppose there are five kinds of bags of candies:

10% are $h_1$: 100% lime candies
20% are $h_2$: 75% lime candies + 25% cherry candies
40% are $h_3$: 50% lime candies + 50% cherry candies
20% are $h_4$: 25% lime candies + 75% cherry candies
10% are $h_5$: 100% cherry candies

Then we observe candies drawn from some bag: ●●●●●●●●●●●

What kind of bag is it? What flavor will the next candy be?
Posterior probability of hypotheses

For example, since here we have 10 cherry candies in a row, the likelihood that this was generated by a given hypothesis is:

\[ P(d|h_1) = 0^{10} = 0 \]
\[ P(d|h_2) = 0.25^{10} = 0.954 \times 10^{-7} \]
\[ P(d|h_3) = 0.5^{10} = 0.001 \]
\[ P(d|h_4) = 0.75^{10} = 0.0563 \]
\[ P(d|h_5) = 1^{10} = 1 \]

Then, we take into account the prior probabilities of each hypothesis.

Assume that the prior distribution over \( h_1, \ldots, h_5 \) (i.e., \( P(h_i) \)) is given by:
\(< 0.1, 0.2, 0.4, 0.2, 0.1 >\)

Computing \( P(d|h_i)P(h_i) \) and normalizing, we have ...
Posterior probability of hypotheses

Posters given data generated from $h_5$

$P(h_1|E)$
$P(h_2|E)$
$P(h_3|E)$
$P(h_4|E)$
$P(h_5|E)$

Number of samples
Let’s say we now want to know the probability that the next candy is lime, given that we’ve seen 10 limes so far. Here our unknown quantity, $X$, is “next candy is lime”.

As we’ve already seen, we calculate predictions using a likelihood-weighted average over the hypotheses:

$$P(X|d) = \sum_i P(X|h_i)P(h_i|d)$$

After 10 lime candies, we have:

$$P(d_{N+1} = lime|d_1...d_N = lime)$$

$$= \sum_i P(d_{N+1} = lime|h_i)P(h_i|d_1...d_N = lime)$$
Prediction probability

\[ P(\text{next candy is lime} \mid \text{d}) \]

Number of samples in \( \text{d} \)
MAP approximation for predictions

Summing over the hypothesis space is often intractable (e.g., 18,446,744,073,709,551,616 Boolean functions of 6 attributes)

**Maximum a posteriori** (MAP) learning: choose \( h_{\text{MAP}} \) maximizing \( P(h_i|d) \)

Remember: \( P(h_i|d) = \alpha P(d|h_i)P(h_i) \)

So, maximize \( P(d|h_i)P(h_i) \) or, equivalently,

minimize \( -\log P(d|h_i) - \log P(h_i) \)

[By taking logarithms, we reduce the product to a sum over the data, which is usually easier to optimize.]

Log terms can be viewed as:

bits to encode data given hypothesis + bits to encode hypothesis

This is the basic idea of **minimum description length** (MDL) learning

For deterministic hypotheses, \( P(d|h_i) \) is 1 if consistent, 0 otherwise

\( \Rightarrow \) MAP = simplest consistent hypothesis
ML approximation

For large data sets, prior becomes irrelevant

**Maximum likelihood (ML) learning:** choose $h_{ML}$ maximizing $P(d|h_i)$

I.e., simply get the best fit to the data; identical to MAP for uniform prior (which is reasonable if all hypotheses are of the same complexity)

ML is the “standard” (non-Bayesian) statistical learning method
Summarizing these 3 types of learning

**Bayesian learning:**

Calculate $P(h_i|d) = \alpha P(d|h_i)P(h_i)$.

**MAP (Maximum a posteriori) learning:**

Choose $h_{MAP}$ that maximizes $P(d|h_i)P(h_i)$ or, equivalently, minimizes $-\log P(d|h_i) - \log P(h_i)$.

(This avoids summing over all hypotheses.)

**ML (Maximum likelihood) learning:**

Choose $h_{ML}$ maximizing $P(d|h_i)$.

(This assumes uniform prior for hypotheses, which is reasonable for large data sets.)
Maximum Likelihood Parameter Learning

Objective: Find numerical parameters for a probability model whose structure is fixed.

Example: A bag of candy

- Unknown fraction of lime/cherry
- Parameter = $\theta = \text{proportion of cherry candies}$
- Hypothesis = $h_\theta = \text{proportion of cherry candies}$
- If assume all proportions are equally likely \textit{a priori}, then ML approach is feasible
- If model as Bayesian network, just need one random variable, $\textit{Flavor}$
Problem Modeled as a Bayesian Network

\[ P(F=\text{cherry}) \]

\[ \emptyset \]

Flavor
Example: Bags of Candy

Suppose unwrap $N$ candies, of which $c$ are cherries, and $l$ are limes.

Remember, we have likelihood of data (assuming i.i.d.) is:

$$P(d|h_i) = \prod_j P(d_j|h_i)$$

So, $P(d|h_\theta) = \prod_{j=1}^N P(d_j|h_\theta) = \theta^c \times (1 - \theta)^l$

The maximum-likelihood hypothesis is given by the value of $\theta$ that maximizes this expression. This is equivalent to maximizing the log likelihood:

$$L(d|h_\theta) = \log P(d|h_\theta) = \sum_{j=1}^N P(d_j|h_\theta) = c \log \theta + l \log(1 - \theta)$$

To find maximum-likelihood value of $\theta$, differentiate $L$ wrt $\theta$, and set result to zero:

$$\frac{dL(d|h_\theta)}{d\theta} = \frac{c}{\theta} - \frac{l}{1 - \theta} = 0 \Rightarrow \theta = \frac{c}{c + l} = \frac{c}{N}$$

Thus, $h_{ML}$ says proportion of cherries in bag $=$ proportion observed so far.
Standard Method for ML Parameter Learning

1. Write down an expression for the likelihood of the data as a function of the parameter(s).

2. Write down the derivative of the log likelihood with respect to each parameter.

3. Find the parameter values such that the derivatives are zero.

Problem: When data set is small enough so that some events have not yet been observed, the maximum likelihood hypothesis assigns zero probability to those events.

Possible solutions: initialize counts for each event to 1 instead of 0.
Another Example: Add Wrappers

Red and green wrappers assigned probabilistically, depending on flavor.

Now have 3 parameters: $\theta$, $\theta_1$, $\theta_2$.

Corresponding Bayesian network:

![Bayesian network diagram]

- $P(F=\text{cherry})$
- $\theta$
- Flavor
- $F$  $P(W=\text{red} \mid F)$
  - cherry  $\theta_1$
  - lime  $\theta_2$
Remember standard semantics of Bayesian Nets... (Note that here we’re showing the parameters as given info, just to make it explicit; normally, the parameters are implicitly assumed as given info.)

\[ P(Flavor = cherry, Wrapper = green | h_\theta, h_{\theta_1}, h_{\theta_2}) \]
Another Example: Add Wrappers

Remember standard semantics of Bayesian Nets... (Note that here we’re showing the parameters as given info, just to make it explicit; normally, the parameters are implicitly assumed as given info.)

\[ P(Flavor = \text{cherry}, \text{Wrapper} = \text{green}| h_\theta, h_{\theta_1}, h_{\theta_2}) \]

\[ = P(Flavor = \text{cherry}| h_\theta, h_{\theta_1}, h_{\theta_2}) \times P(\text{Wrapper} = \text{green}| Flavor = \text{cherry}, h_\theta, h_{\theta_1}, h_{\theta_2}) \]
Another Example: Add Wrappers

Remember standard semantics of Bayesian Nets... (Note that here we’re showing the parameters as given info, just to make it explicit; normally, the parameters are implicitly assumed as given info.)

\[
P(Flavor = cherry, Wrapper = green|h_\theta, h_{\theta_1}, h_{\theta_2})
\]

\[
= P(Flavor = cherry|h_\theta, h_{\theta_1}, h_{\theta_2}) \times P(Wrapper = green|Flavor = cherry, h_\theta, h_{\theta_1}, h_{\theta_2})
\]

\[
= \theta \cdot (1 - \theta_1)
\]
Another Example: Add Wrappers

Now, unwrap \( N \) candies, of which \( c \) are cherry and \( l \) are limes.

\( r_c \) of cherries have red wrappers;
\( g_c \) of cherries have green wrappers;
\( r_l \) of limes have red wrappers;
\( g_l \) of limes have green wrappers.

Likelihood of data:

\[
P(\mathbf{d}|h_{\theta}, h_{\theta_1}, h_{\theta_2}) = \theta^c (1 - \theta)^l \cdot \theta^{r_c}_{1} (1 - \theta_1)^{g_c} \cdot \theta^{r_l}_{2} (1 - \theta_2)^{g_l}
\]

Taking logs gives us:

\[
L = [c \log \theta + l \log(1 - \theta)] + [r_c \log \theta_1 + g_c \log(1 - \theta_1)] + [r_l \log \theta_2 + g_l \log(1 - \theta_2)]
\]
Another Example: Add Wrappers

\[ L = \left[c \log \theta + l \log(1 - \theta)\right] + \left[r_c \log \theta_1 + g_c \log(1 - \theta_1)\right] + \left[r_l \log \theta_2 + g_l \log(1 - \theta_2)\right] \]

Take derivatives wrt each parameter and set to 0 gives us:

\[ \frac{\partial L}{\partial \theta} = \frac{c}{\theta} - \frac{l}{1 - \theta} = 0 \Rightarrow \theta = \frac{c}{c + l} \]

\[ \frac{\partial L}{\partial \theta_1} = \frac{r_c}{\theta_1} - \frac{g_c}{1 - \theta_1} = 0 \Rightarrow \theta_1 = \frac{r_c}{r_c + g_c} \]

\[ \frac{\partial L}{\partial \theta_2} = \frac{r_l}{\theta_2} - \frac{g_l}{1 - \theta_2} = 0 \Rightarrow \theta_2 = \frac{r_l}{r_l + g_l} \]

**Important point 1:** with complete data, ML parameter learning for a Bayesian network decomposes into separate learning problems, one for each parameter.

**Important point 2:** parameter values for a variable, given its parents, are the observed freq. of the variable values for each setting of the parent values.
Naive Bayes Models

**Naive Bayes Model** = Most common Bayesian network model

**Representation:** “Class” variable \((C)\) is the root, and “attribute” variables \((X_i)\) are the leaves.

**Example** of Naive Bayes Model with one attribute:

\[
P(C|x_1, \ldots x_n) = \alpha P(C) \prod_i P(x_i|C)
\]

“Naive” \(\Rightarrow\) assumes attributes are conditionally independent, given the class.

**Objective:** after learning, predict “Class” of new examples
Naive Bayes Models (con’t)

Nice characteristics of Naive Bayes learning:

• Works surprisingly well in wide range of applications
• Scales well to very large problems \((n \text{ Boolean attributes} \Rightarrow 2n + 1 \text{ parameters})\)
• No search is required to find \(h_{ML}\)
• No difficulty with noisy data
• Can give probabilistic predictions when appropriate
Naive Bayes Classifier Example

We want to build a classifier that can classify days according to whether someone will play tennis. How do we start?
## Naive Bayes Classifier Example

<table>
<thead>
<tr>
<th>Day</th>
<th>Outlook</th>
<th>Temp</th>
<th>Humidity</th>
<th>Wind</th>
<th>Play?</th>
</tr>
</thead>
<tbody>
<tr>
<td>D1</td>
<td>Sunny</td>
<td>Hot</td>
<td>High</td>
<td>Weak</td>
<td>No</td>
</tr>
<tr>
<td>D2</td>
<td>Sunny</td>
<td>Hot</td>
<td>High</td>
<td>Strong</td>
<td>No</td>
</tr>
<tr>
<td>D3</td>
<td>Overcast</td>
<td>Hot</td>
<td>High</td>
<td>Weak</td>
<td>Yes</td>
</tr>
<tr>
<td>D4</td>
<td>Rain</td>
<td>Mild</td>
<td>High</td>
<td>Weak</td>
<td>Yes</td>
</tr>
<tr>
<td>D5</td>
<td>Rain</td>
<td>Cool</td>
<td>Normal</td>
<td>Weak</td>
<td>Yes</td>
</tr>
<tr>
<td>D6</td>
<td>Rain</td>
<td>Cool</td>
<td>Normal</td>
<td>Strong</td>
<td>No</td>
</tr>
<tr>
<td>D7</td>
<td>Overcast</td>
<td>Cool</td>
<td>Normal</td>
<td>Strong</td>
<td>Yes</td>
</tr>
<tr>
<td>D8</td>
<td>Sunny</td>
<td>Mild</td>
<td>High</td>
<td>Weak</td>
<td>No</td>
</tr>
<tr>
<td>D9</td>
<td>Sunny</td>
<td>Cool</td>
<td>Normal</td>
<td>Weak</td>
<td>Yes</td>
</tr>
<tr>
<td>D10</td>
<td>Rain</td>
<td>Mild</td>
<td>Normal</td>
<td>Weak</td>
<td>Yes</td>
</tr>
<tr>
<td>D11</td>
<td>Sunny</td>
<td>Mild</td>
<td>Normal</td>
<td>Strong</td>
<td>Yes</td>
</tr>
<tr>
<td>D12</td>
<td>Overcast</td>
<td>Mild</td>
<td>High</td>
<td>Strong</td>
<td>Yes</td>
</tr>
<tr>
<td>D13</td>
<td>Overcast</td>
<td>Hot</td>
<td>Normal</td>
<td>Weak</td>
<td>Yes</td>
</tr>
<tr>
<td>D14</td>
<td>Rain</td>
<td>Mild</td>
<td>High</td>
<td>Strong</td>
<td>No</td>
</tr>
</tbody>
</table>
Now, we have a new instance we want to classify:
(Outlook = Sunny, Temp = Cool, Humidity = High, Wind = Strong)

Need to predict 'Yes' or 'No'.

What now?
Naive Bayes Classifier Example

Now, we have a new instance we want to classify:
(Outlook = Sunny, Temp = Cool, Humidity = High, Wind = Strong)

Need to predict 'Yes' or 'No'.

What now?

Let $v$ be either 'Yes' or 'No'.

Then,

$$v = \arg\max_{v_j \in \{Yes, No\}} P(v_j) \prod_i P(x_i | v_j)$$

$$= \arg\max_{v_j \in \{Yes, No\}} P(v_j) \times P(\text{Outlook} = \text{Sunny} | v_j) \times P(\text{Temp} = \text{Cool} | v_j)$$

$$\times P(\text{Humidity} = \text{High} | v_j) \times P(\text{Wind} = \text{Strong} | v_j)$$
Naive Bayes Classifier Example

\[ P(\text{Play?}=\text{Yes}) = \]
\[ P(\text{Play?}=\text{No}) = \]
\[ P(\text{Outlook}=\text{Sunny}|\text{Play?}=\text{Yes}) = \]
\[ P(\text{Outlook}=\text{Sunny}|\text{Play?}=\text{No}) = \]
\[ P(\text{Temp}=\text{Cool}|\text{Play?}=\text{Yes}) = \]
\[ P(\text{Temp}=\text{Cool}|\text{Play?}=\text{No}) = \]
\[ P(\text{Humidity}=\text{High}|\text{Play?}=\text{Yes}) = \]
\[ P(\text{Humidity}=\text{High}|\text{Play?}=\text{No}) = \]
\[ P(\text{Wind}=\text{Strong}|\text{Play?}=\text{Yes}) = \]
\[ P(\text{Wind}=\text{Strong}|\text{Play?}=\text{No}) = \]
Naive Bayes Classifier Example

\[ P(\text{Play?=Yes}) = \frac{9}{14} = 0.64 \]

\[ P(\text{Play?=No}) = \frac{5}{14} = 0.36 \]

\[ P(\text{Outlook=Sunny}|\text{Play?=Yes}) = \frac{2}{9} = 0.22 \]

\[ P(\text{Outlook=Sunny}|\text{Play?=No}) = \frac{3}{5} = 0.60 \]

\[ P(\text{Temp=Cool}|\text{Play?=Yes}) = \frac{3}{9} = 0.33 \]

\[ P(\text{Temp=Cool}|\text{Play?=No}) = \frac{1}{5} = 0.20 \]

\[ P(\text{Humidity=High}|\text{Play?=Yes}) = \frac{3}{9} = 0.33 \]

\[ P(\text{Humidity=High}|\text{Play?=No}) = \frac{4}{5} = 0.80 \]

\[ P(\text{Wind=Strong}|\text{Play?=Yes}) = \frac{3}{9} = 0.33 \]

\[ P(\text{Wind=Strong}|\text{Play?=No}) = \frac{3}{5} = 0.60 \]
Naive Bayes Classifier Example

And, what is decision? Yes or No?
Naive Bayes Classifier Example

And, what is decision? Yes or No?

\[
P(\text{Yes})P(\text{Sunny}|\text{Yes})P(\text{Cool}|\text{Yes})P(\text{High}|\text{Yes})P(\text{Strong}|\text{Yes})
\]
\[
= 0.64 \times 0.22 \times 0.33 \times 0.33 \times 0.33 = 0.0051
\]

\[
P(\text{No})P(\text{Sunny}|\text{No})P(\text{Cool}|\text{No})P(\text{High}|\text{No})P(\text{Strong}|\text{No})
\]
\[
= 0.36 \times 0.60 \times 0.20 \times 0.80 \times 0.60 = 0.0207
\]

Conditional probability that value is “No”:

\[
\frac{0.0207}{0.0207 + 0.0051} = 0.80
\]

No tennis today.
How to learn continuous models from data? Very similar to discrete case.

Data: pairs \((x_1, y_1), \ldots, (x_N, y_N)\)

Hypotheses: straight lines \(y = ax + b\) with Gaussian noise

Want to choose parameters \(\theta = (a, b)\) to maximize likelihood of data (this is linear regression)
Recall: Method for ML Parm Learning

1. Write down an expression for the likelihood of the data as a function of the parameter(s).

2. Write down the derivative of the log likelihood with respect to each parameter.

3. Find the parameter values such that the derivatives are zero.
Data assumed i.i.d. (independently and identically distributed)
\[ \Rightarrow \text{ likelihood } P(d|h_i) = \prod_j P(d_j|h_i) \]

Maximizing likelihood \( P(d|h_i) \) ⇔ maximizing log likelihood
\[ L = \log P(d|h_i) = \log \prod_j P(d_j|h_i) = \sum_j \log P(d_j|h_i) \]

For a continuous hypothesis space, set \( \partial L/\partial \theta = 0 \) and solve for \( \theta \)

For Gaussian noise, \( P(d_j|h_i) = \alpha \exp \left( -\frac{(y_j - (ax_j + b))^2}{2\sigma^2} \right) \), so
\[ L = \sum_j \log P(d_j|h_i) = -\alpha' \sum_j (y_j - (ax_j + b))^2 \]
so maximizing \( L = \) minimizing sum of squared errors

Note: This is just standard linear regression! That is, linear regression is
the same thing as maximum-likelihood (ML) learning (as long as data are
generated with Gaussian noise of fixed variance).
To find the maximum, set derivatives to zero:

\[
\frac{\partial L}{\partial a} = -\alpha' \sum_j 2(y_j -(ax_j+b)) \cdot (-x_j) = 0
\]

\[
\frac{\partial L}{\partial b} = -\alpha' \sum_j 2(y_j -(ax_j+b)) \cdot (-1) = 0
\]

Solve for parameters \(a\) and \(b\). Solutions are:

\[
a = \frac{\sum_j x_j \sum_j y_j - N \sum_j x_j y_j}{(\sum_j x_j)^2 - N \sum_j x_j^2}; \quad b = \frac{(\sum_j y_j - a \sum_j x_j)}{N}
\]
ML learning with Continuous Models

Let’s assume we have $m$ data points $(x_j, y_j)$, where $y_j$’s generated from $x_j$’s according to the following linear Gaussian model:

$$P(y|x) = \frac{1}{\sqrt{2\pi\sigma}} e^{-\frac{(y-(\theta_1 x + \theta_2))^2}{2\sigma^2}}$$

Find the values of $\theta_1$, $\theta_2$, and $\sigma$ that maximize the conditional log likelihood of the data.
ML learning with Continuous Models

Let’s assume we have $m$ data points $(x_j, y_j)$, where $y_j$’s generated from $x_j$’s according to the following linear Gaussian model:

$$P(y|x) = \frac{1}{\sqrt{2\pi}\sigma}e^{-\frac{(y-(\theta_1 x + \theta_2))^2}{2\sigma^2}}$$

Find the values of $\theta_1$, $\theta_2$, and $\sigma$ that maximize the conditional log likelihood of the data.

Likelihood of this data set is:

$$\prod_{j=1}^{m} P(y_j|x_j)$$

Then, compute log likelihood, take derivative, set equal to 0, and solve for the 3 parameters.
Learning Bayes Net structures

What if structure of Bayes net is not known?

Techniques for learning are not well-established.

General idea: search for good model, then measure its quality.

**Searching for a good model**

Option 1: Start with no links, and add parents for each node, fitting the parameters, and then measuring the accuracy of the result.

Option 2: Guess a structure, then use hill-climbing, simulated annealing, etc., to improve, re-tuning the parameters after each step.
Learning Bayes Net structures (con’t.)

How to measure quality of structure?

Option 1: Test whether conditional independence assertions implicit in the structure are actually satisfied in the data.

Option 2: Measure the degree to which model explains the data (probabilistically).

Problem: fully connected network will have high correlation to data. So, need to penalize for complexity. Usually, use MCMC for sampling over structures.
Learning with Hidden Variables

Hidden (or latent) variables: not observable in data

Hidden variables can dramatically reduce number of parameters needed to specify a Bayesian network ⇒ dramatically reduce amount of data needed to learn parameters

(a)

(b)
Expectation-Maximization (EM) Algorithm

Solves problem in general way

Applicable to huge range of learning problems

2-step process in each iteration (i.e., repeat until convergence):

- E-step (i.e., Expectation step): compute expected values of hidden variables (below, it’s the summation)
- M-step (i.e., Maximization step): find new values of parameters that maximize log likelihood of data, given expected values of hidden variables

General form of EM algorithm:

\[
\theta^{(i+1)} = \arg \max_\theta \sum_z P(Z = z|x, \theta^{(i)}) L(x, Z = z|\theta)
\]
Using EM Algorithm

Starting from the general form, it is possible to derive an EM algorithm for a specific application once the appropriate hidden variables have been identified.

Examples:

♦ Learning mixtures of Gaussians (unsupervised clustering)
♦ Learning Bayesian networks with hidden variables
♦ Learning hidden Markov models
Unsupervised clustering: discerning multiple categories in a collection of objects

Mixtures of Gaussians: combination of Gaussian “component” distributions

Example of Gaussian mixture model with 2 components:
Unsupervised Clustering (con’t.)

Data are generated from mixture distribution $P$.

Distribution $P$ has $k$ components, each of which is itself a distribution

Let $C = \text{component}$, with values $1, \ldots, k$.

Let $x = \text{values of attributes of data point}$.

Then, mixture distribution is:

$$P(x) = \sum_{i=1}^{k} P(C = i) P(x | C = i)$$

Parameters of a mixture of Gaussians:

- $w_i = P(C = i)$: the weight of each component
- $\mu_i$: mean of each component
- $\Sigma_i$: the covariance of each component

Unsupervised clustering objective: learn mixture model from raw data.
Unsupervised Clustering Challenge

If we knew which component generated each data point, then it is easy to recover the component Gaussians – just select all the data points from a given component and apply ML parameter learning (like eqn. (20.4)).

Or, if we know the parameters of each component, we could probabilistically assign each data point to a component.

Problem:
Unsupervised Clustering Challenge

If we knew which component generated each data point, then it is easy to recover the component Gaussians – just select all the data points from a given component and apply ML parameter learning (like eqn. (20.4)).

Or, if we know the parameters of each component, we could probabilistically assign each data point to a component.

Problem: We don’t know either.
Unsupervised Clustering (con’t.)

Approach:

- Pretend we know the parameters of the model, and infer the probability that each data point belongs to each component.
- Then, retrofit the components to the data, where each component is fitted to entire data set with each point weighted by the probability that it belongs to that component.

Iterate until convergence.

What is “hidden” in unsupervised clustering?
Unsupervised Clustering (con’t.)

Approach:

• Pretend we know the parameters of the model, and infer the probability that each data point belongs to each component

• Then, retrofit the components to the data, where each component is fitted to entire data set with each point weighted by the probability that it belongs to that component.

Iterate until convergence.

What is “hidden” in unsupervised clustering?

We don’t know which component each data set belongs to.
Unsupervised Clustering (con’t.)

Two-step algorithm:

1. **E-step**: Compute $p_{ij} = P(C = i | x_j)$ (i.e., probability that $x_j$ was generated by component $i$)

   $$p_{ij} = \alpha P(x_j | C = i) P(C = i)$$

2. **M-step**: Compute new mean, covariance, and component weights:

   \[
   \begin{align*}
   \mu_i & \leftarrow \sum_j p_{ij} x_j / p_i \\
   \Sigma_i & \leftarrow \sum_j p_{ij} x_j x_j^\top / p_i \\
   w_i & \leftarrow p_i
   \end{align*}
   \]

Iterate until converge to a solution.
Some potential problems

♦ One Gaussian component could shrink to cover just a single data point; then, variance goes to 0 and its likelihood goes to infinity!

♦ Two components can merge, acquiring identical means and variances, and sharing data points.

Possible solutions:

♦ Place priors on model parameters and apply the MAP version of EM.

♦ Restart a component with new random parameters if it gets too small or too close to another component.

♦ Initialize the parameters with reasonable values.
EM Ex. 2: Bayesian NW with Hidden Vars

Two bags of candies mixed together.

Three features of candy: **Flavor, Wrapper, Hole**.

Distribution of candies in each bag described by Naive Bayes model (i.e., features are independent, given the bag, but CPT depends on bag)
Parameters:

\( \theta \): prior probability that candy comes from Bag 1

\( \theta_{F1}, \theta_{F2} \): probabilities that flavor is cherry, given it comes from Bag 1 and Bag 2, respectively

\( \theta_{W1}, \theta_{W2} \): probabilities that wrapper is red, given it comes from Bag 1 and Bag 2, respectively

\( \theta_{H1}, \theta_{H2} \): probabilities that candy has hole, given it comes from Bag 1 and Bag 2, respectively

Hidden variable: the bag

Objective: Learn the descriptions of the two bags by observing candies from mixture
Let’s step through iteration of EM.

Data: 1000 samples from model whose true parameters are:

\[ \theta = 0.5 \]
\[ \theta_{F1} = \theta_{W1} = \theta_{H1} = 0.8 \]
\[ \theta_{F2} = \theta_{W2} = \theta_{H2} = 0.3 \]

Data counts:

<table>
<thead>
<tr>
<th></th>
<th>W=red</th>
<th>W=green</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>H=1</td>
<td>H=0</td>
</tr>
<tr>
<td>F = cherry</td>
<td>273</td>
<td>93</td>
</tr>
<tr>
<td>F = lime</td>
<td>79</td>
<td>100</td>
</tr>
</tbody>
</table>
Learning Bayesian NW with HV, con’t.

First, initialize parameters:

\[
\begin{align*}
\theta^{(0)} &= 0.6 \\
\theta_{F_1}^{(0)} &= \theta_{W_1}^{(0)} = \theta_{H_1}^{(0)} = 0.6 \\
\theta_{F_2}^{(0)} &= \theta_{W_2}^{(0)} = \theta_{H_2}^{(0)} = 0.4
\end{align*}
\]

Now, work on \( \theta \).

Because the bag is a hidden variable, we estimate this from expected counts:

\[
\hat{N}(Bag = 1) = \text{sum, over all candies, of probability that candy came from bag 1:}
\]

\[
\theta^{(1)} = \frac{\hat{N}(Bag = 1)}{N}
\]

\[
= \sum_{j=1}^{N} \frac{P(B = 1|f_j, w_j, h_j)}{N}
\]

\[
\theta^{(1)} = \frac{1}{N} \sum_{j=1}^{N} \frac{P(f_j|B = 1)P(w_j|B = 1)P(h_j|B = 1)P(B = 1)}{\sum_{i=1}^{N} P(f_j|B = i)P(w_j|B = i)P(h_j|B = i)P(B = i)}
\]
Learning Bayesian NW with HV, con’t.

Apply to the 273 red-wrapped cherry candies with holes:

\[
\frac{273}{1000} \cdot \frac{\theta_{F1}^{(0)} \theta_{W1}^{(0)} \theta_{H1}^{(0)} \theta^{(0)}}{\theta_{F1}^{(0)} \theta_{W1}^{(0)} \theta_{H1}^{(0)} + \theta_{F2}^{(0)} \theta_{W2}^{(0)} \theta_{H2}^{(0)} (1 - \theta^{(0)})} \approx 0.22797
\]

Applying to remaining 7 kinds of candy, we get \( \theta^{(1)} = 0.6124 \).

Now, look at \( \theta_{F1} \). Expected count of cherry candies from Bag 1 is:

\[
\sum_{j:Flavor_j=cherry} P(Bag = 1 | Flavor_j = cherry, wrapper_j, holes_j)
\]

Calculate these probabilities (using inference alg. for Bayes net):

\[
\begin{align*}
\theta^{(1)} &= 0.6124 \\
\theta_{F1}^{(1)} &= 0.6684 & \theta_{W1}^{(1)} &= 0.6483 & \theta_{H1}^{(1)} &= 0.6558 \\
\theta_{F2}^{(1)} &= 0.3887 & \theta_{W2}^{(1)} &= 0.3817 & \theta_{H2}^{(1)} &= 0.3827
\end{align*}
\]
Main Lesson from Bayesian NW Learning

Main points:

- Parameter updates for Bayesian network learning with hidden variables are directly available from the results of inference on each example (so, no extra computations specific to learning).
- Only local posteriori probabilities are needed for each parameter.

In general, for learning conditional probability parameters for each variable $X_i$, given its parents (i.e., $\theta_{ijk} \leftarrow P(X_i = x_{ij}, Pa_i = pa_{ik})$), the update is given by normalized expected counts:

$$\theta_{ijk} \leftarrow \frac{\hat{N}P(X_i = x_{ij}, Pa_i = pa_{ik})}{\hat{N}(Pa_i = pa_{ik})}$$

Expected counts are obtained by summing over the examples and computing the probabilities $P(X_i = x_{ij}, Pa_i = pa_{ik})$ using a Bayes net inference algorithm.
In Hidden Markov Models, hidden variables are the $i \to j$ transitions.

Each data point: finite observation sequence

Objective: learn transition probabilities from sequences

Similar to learning Bayesian nets with hidden variables.

Complication: In Bayes nets, each parameter is distinct. In HMM, individual transition probabilities from state $i$ to state $j$ are repeated across time (i.e., $\theta_{ijt} = \theta_{ij}$ for all $t$)

Thus, to estimate transition probability from state $i$ to state $j$, we calculate expected proportion of times that the system undergoes transition to state $j$ when in state $i$:

$$\theta_{ij} \leftarrow \frac{\sum_t \hat{N}(X_{t+1} = j, X_t = i)}{\sum_t \hat{N}(X_t = i)}$$
Learning HMMs (con’t.)

Similar to learning Bayesian NWs, here we can compute expected counts by any HMM inference algorithm.

E.g., the forward-backward algorithm of smoothing (Fig. 15.4) can be used (modified easily to compute the required probabilities). This algorithm is also known as **Baum-Welch** algorithm.

Note: we’re making use of **smoothing** instead of **filtering**, since we need to pay attention to subsequent evidence in estimating the probability that a particular transition occurred.

The forward-backward (Baum-Welch) algorithm is a particular case of the generalized EM. It computes maximum likelihood estimate of the parameters of the HMM given a sequence of outputs.
Expectation Maximization algorithm:

2-step process:

- E-step (i.e., Expectation step): compute expected values of hidden variables (below, it’s the summation)
- M-step (i.e., Maximization step): find new values of parameters that maximize log likelihood of data, given expected values of hidden variables

General form of EM algorithm:

\[ \theta^{(i+1)} = \operatorname{argmax}_\theta \sum_z P(Z = z|x, \theta^{(i)}) L(x, Z = z|\theta) \]
Summary

Full Bayesian learning gives best possible predictions but is intractable.

MAP learning balances complexity with accuracy on training data.

Maximum likelihood assumes uniform prior, OK for large data sets.

ML for continuous spaces using gradient of log likelihood.

Regression with Gaussian noise → minimize sum-of-squared errors.

When some variables are hidden, local maximum likelihood solutions can be found using Expectation Maximization algorithm.