TDT4173 Machine Learning

Lecture 5 – Bayesian Methods

Norwegian University of Science and Technology

Helge L.
IT-VEST 310
helgel@idi.ntnu.no

NTNU

Outline

Why use for Bayesian Methods

Provides practical learning algorithms:

- Naïve Bayes learning
- Bayesian belief network learning
- Combine prior knowledge (prior probabilities) with observed data
- Requires prior probabilities
- Provides "gold standard" for evaluating other learning algorithms

Basic Formulas for Probabilities

- **Product Rule**: probability $P(A \wedge B)$ of a conjunction of two events A and B:

  $$P(A \wedge B) = P(A|B) \cdot P(B) = P(A) \cdot P(B|A)$$

  If A and B are **independent**, then

  $$P(A \wedge B) = P(A) \cdot P(B) \text{ – because } P(A|B) = P(A)$$

- **Sum Rule**: probability of a disjunction of two events A and B:

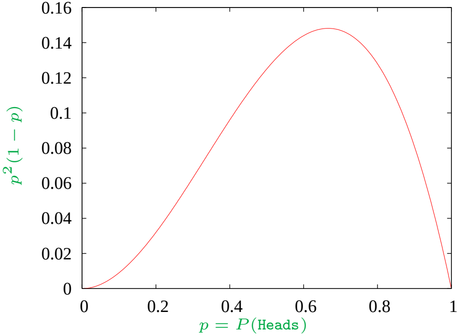
  $$P(A \vee B) = P(A) + P(B) - P(A \wedge B)$$

- **Theorem of total probability**: if events $A_1, \ldots, A_n$ are mutually exclusive with $\sum_{i=1}^{n} P(A_i) = 1$, then

  $$P(B) = \sum_{i=1}^{n} P(B|A_i)P(A_i)$$

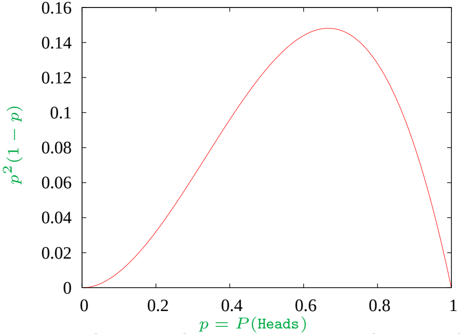
Maximum Likelihood Estimation

- How should we determine the parameters in a model, e.g., the probabilities?
- **Class task:** Flipping a coin three times gives `Heads` twice and `Tails` once. What is $p =$ probability of a coin-flip being `Heads`?



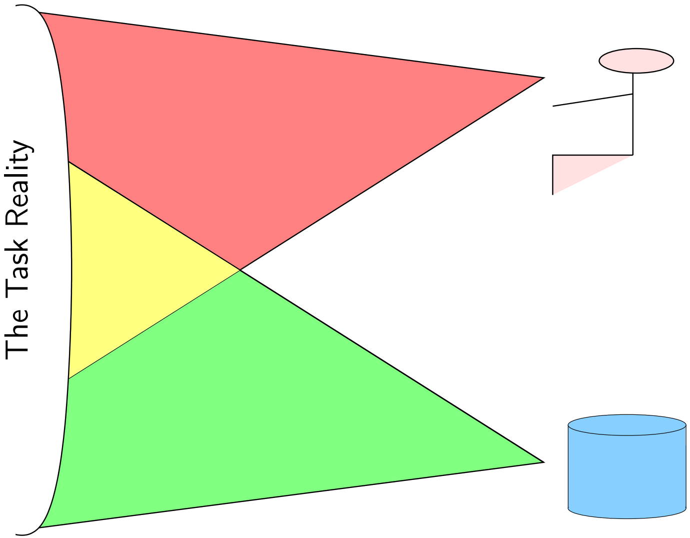
Maximum Likelihood Estimation

- How should we determine the parameters in a model, e.g., the probabilities?
- **Class task:** Flipping a coin three times gives `Heads` twice and `Tails` once. What is $p =$ probability of a coin-flip being `Heads`?



- **Classic answer:** Choose the parameters that makes our observations most likely. Here: $\hat{p} = .67$. **Reasonable?**

Data and User views



Bayes Theorem

- $P(h)$ = prior probability of hypothesis h
- $P(D)$ = prior probability of training data D
- $P(h|D)$ = probability of h given D
- $P(D|h)$ = probability of D given h

$$P(h|D) = \frac{P(D|h)P(h)}{P(D)}$$

Choosing Hypotheses

$$P(h|D) = \frac{P(D|h)P(h)}{P(D)}$$

Generally want the most probable hypothesis given the training data

Maximum a posteriori hypothesis h_{MAP}:

$$\begin{aligned} h_{\text{MAP}} &= \operatorname*{argmax}_{h \in H} P(h|D) \\ &= \operatorname*{argmax}_{h \in H} \frac{P(D|h)P(h)}{P(D)} \\ &= \operatorname*{argmax}_{h \in H} P(D|h)P(h) \end{aligned}$$

Choosing Hypotheses

$$P(h|D) = \frac{P(D|h)P(h)}{P(D)}$$

Generally want the most probable hypothesis given the training data

Maximum a posteriori hypothesis h_{MAP}:

$$\begin{aligned} h_{\text{MAP}} &= \operatorname*{argmax}_{h \in H} P(h|D) \\ &= \operatorname*{argmax}_{h \in H} \frac{P(D|h)P(h)}{P(D)} \\ &= \operatorname*{argmax}_{h \in H} P(D|h)P(h) \end{aligned}$$

If we assume $P(h_i) = P(h_j)$, then h_{MAP} is equal to the *Maximum likelihood* (ML) hypothesis

$$h_{\text{ML}} = \operatorname*{argmax}_{h \in H} P(D|h).$$

Bayes Theorem

Example: Does patient have cancer or not?

A patient takes a lab test and the result comes back positive. The test returns a correct positive result in only 98% *of the cases in which the disease is actually present, and a correct negative result in only* 97% *of the cases in which the disease is not present. Furthermore,* $.008$ *of the entire population have this cancer.*

$$P(\texttt{cancer}) = \qquad P(\neg\texttt{cancer}) =$$
$$P(\oplus|\texttt{cancer}) = \qquad P(\ominus|\texttt{cancer}) =$$
$$P(\oplus|\neg\texttt{cancer}) = \qquad P(\ominus|\neg\texttt{cancer}) =$$

Conditional Independence

Definition:

X is conditionally independent *of* Y *given* Z *if the probability distribution governing* X *is independent of the value of* Y *given the value of* Z*; that is, if*

$$(\forall x_i, y_j, z_k)\; P(X = x_i | Y = y_j, Z = z_k) = P(X = x_i | Z = z_k)$$

more compactly, we write

$$P(X|Y, Z) = P(X|Z)$$

Example:

`Thunder` is conditionally independent of `Rain`, given `Lightning`

$$P(\texttt{Thunder}|\texttt{Rain}, \texttt{Lightning}) = P(\texttt{Thunder}|\texttt{Lightning})$$

Bayesian Networks

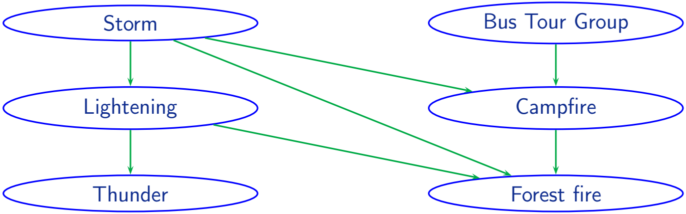
Interesting because:

- General infference in probabilistic models is intractable without some assumptions. . .
- Bayesian Belief networks describe conditional independence among *subsets* of variables

→ allows combining prior knowledge about (in)dependencies among variables with observed training data

(also called Bayes Belief Nets)

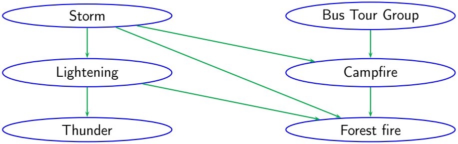
Bayesian Networks (2)



Network represents a set of conditional independence assertions:

- Each node is asserted to be conditionally independent of its non-descendants, given its parents.
- Directed acyclic graph

Bayesian Networks (3)



Represents joint probability distribution over all variables

- e.g., $P(\text{Storm}, \text{BusTourGroup}, \ldots, \text{ForestFire})$
- in general,

$$P(y_1, \ldots, y_n) = \prod_{i=1}^{n} P(y_i | \text{Pa}(Y_i))$$

where $\text{Pa}(Y_i)$ denotes *parents* of Y_i in graph

- Joint distribution is fully defined by the graph and $\{P(y_i | \text{Pa}(Y_i))\}$

Inference in Bayesian Networks

How can one infer the (probabilities of) values of one or more network variables, given observed values of others?

- Bayes net contains all information needed for this inference
- If only one variable with unknown value, easy to infer it
- In general case, problem is NP hard

In practice, we can succeed in many ways

- Exact inference methods work well for some network structures **(HUGIN)**
- Monte Carlo methods "simulate" the network randomly to calculate approximate solutions

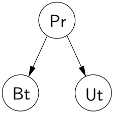
Learning probabilities from a database

We have:

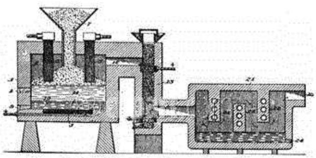
- A Bayesian network structure.
- A database of cases over (some of) the variables.

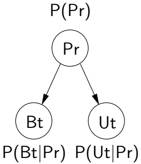
We want:

- A Bayesian network model (with probabilities) representing the database.



Cases	Pr	Bt	Ut
1.	?	pos	pos
2.	yes	neg	pos
3.	yes	pos	?
4.	yes	pos	neg
5.	?	neg	?





Complete data: Maximum likelihood estimation

We have tossed a thumb tack 100 times. It has landed pin up 80 times, and we now look for the model that best fits the observations/data:

Structure	....	T	T	T	....
Probability, $P(\text{pin up}) =$		0.1	0.2	0.3	
Model	....	$M_{0.1}$	$M_{0.2}$	$M_{0.3}$	....

Complete data: Maximum likelihood estimation

We have tossed a thumb tack 100 times. It has landed pin up 80 times, and we now look for the model that best fits the observations/data:

Structure	....	T	T	T	....
Probability, $P(\text{pin up}) =$		0.1	0.2	0.3	
Model	....	$M_{0.1}$	$M_{0.2}$	$M_{0.3}$	....

We can measure how well a model fits the data using:

$$
\begin{aligned}
P(\mathcal{D}|M_\theta) &= P(\text{pin up}, \text{pin down}, \text{pin down}, \ldots, \text{pin up}|M_\theta) \\
&= P(\text{pin up}|M_\theta) \cdot P(\text{pin down}|M_\theta) \cdot \ldots \cdot P(\text{pin up}|M_\theta)
\end{aligned}
$$

This is also called the likelihood of M_θ given $\mathcal{D}$.

Complete data: Maximum likelihood estimation

We have tossed a thumb tack 100 times. It has landed pin up 80 times, and we now look for the model that best fits the observations/data:

Structure	....	T	T	T	....
Probability, $P(\text{pin up}) =$		0.1	0.2	0.3	
Model	....	$M_{0.1}$	$M_{0.2}$	$M_{0.3}$	....

We select the parameter $\hat{\theta}$ that maximizes:

$$\hat{\theta} = \arg\max_{\theta} P(\mathcal{D}|M_\theta) = \arg\max_{\theta} \prod_{i=1}^{100} P(d_i|M_\theta)$$
$$= \arg\max_{\theta} \mu \cdot \theta^{80}(1-\theta)^{20}.$$

Complete data: Maximum likelihood estimation

We have tossed a thumb tack 100 times. It has landed pin up 80 times, and we now look for the model that best fits the observations/data:

Structure	....	T	T	T	....
Probability, $P(\text{pin up}) =$		0.1	0.2	0.3	
Model	....	$M_{0.1}$	$M_{0.2}$	$M_{0.3}$	....

By setting:

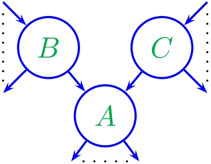
$$\frac{d}{d\theta}\mu \cdot \theta^{80}(1-\theta)^{20} = 0$$

we get the **maximum likelihood estimate:**

$$\hat{\theta} = 0.8 \quad \equiv \frac{\#\textbf{pin up}}{\#\textbf{tosses}}.$$

Complete data: General maximum likelihood estimation

In general, you get a maximum likelihood estimate as the fraction of counts over the total number of counts.



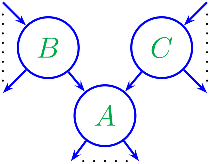
We want $P(A = a|B = b, C = c)$!

To find the maximum likelihood estimate $\hat{P}(A = a|B = b, C = c)$ we simply calculate:

$$\hat{P}(A = a|B = b, C = c) =$$

Complete data: General maximum likelihood estimation

In general, you get a maximum likelihood estimate as the fraction of counts over the total number of counts.



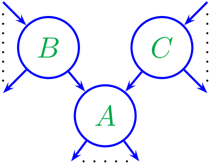
We want $P(A = a|B = b, C = c)$!

To find the maximum likelihood estimate $\hat{P}(A = a|B = b, C = c)$ we simply calculate:

$$\hat{P}(A = a|B = b, C = c) = \frac{\hat{P}(A = a, B = b, C = c)}{\hat{P}(B = b, C = c)}$$

Complete data: General maximum likelihood estimation

In general, you get a maximum likelihood estimate as the fraction of counts over the total number of counts.



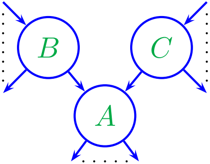
We want $P(A = a|B = b, C = c)$!

To find the maximum likelihood estimate $\hat{P}(A = a|B = b, C = c)$ we simply calculate:

$$\hat{P}(A = a|B = b, C = c) = \frac{\hat{P}(A = a, B = b, C = c)}{\hat{P}(B = b, C = c)} = \frac{\left[\frac{N(A=a,B=b,C=c)}{N}\right]}{\left[\frac{N(B=b,C=c)}{N}\right]}$$

Complete data: General maximum likelihood estimation

In general, you get a maximum likelihood estimate as the fraction of counts over the total number of counts.



We want $P(A = a|B = b, C = c)$!

To find the maximum likelihood estimate $\hat{P}(A = a|B = b, C = c)$ we simply calculate:

$$\hat{P}(A = a|B = b, C = c) = \frac{\hat{P}(A = a, B = b, C = c)}{\hat{P}(B = b, C = c)} = \frac{\left[\frac{N(A=a,B=b,C=c)}{N}\right]}{\left[\frac{N(B=b,C=c)}{N}\right]}$$

$$= \frac{N(A = a, B = b, C = c)}{N(B = b, C = c)}.$$

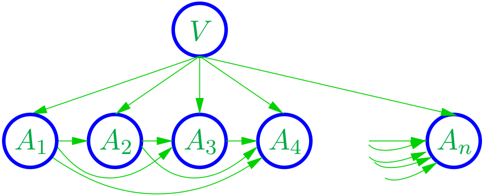
So we have a simple counting problem!

General classification – and the Naïve Bayes Classifier

Assume target function $f : \mathcal{X} \to V$, where each instance x described by attributes $\langle a_1, a_2 \ldots a_n \rangle$. Most probable value of $f(x)$:

$$v_{\mathsf{MAP}} = \operatorname*{argmax}_{v_j \in V} P(v_j | a_1, a_2 \ldots a_n)$$

$$\begin{aligned} v_{\mathsf{MAP}} &= \operatorname*{argmax}_{v_j \in V} \frac{P(a_1, a_2 \ldots a_n | v_j) P(v_j)}{P(a_1, a_2 \ldots a_n)} \\ &= \operatorname*{argmax}_{v_j \in V} P(a_1, a_2 \ldots a_n | v_j) P(v_j) \end{aligned}$$

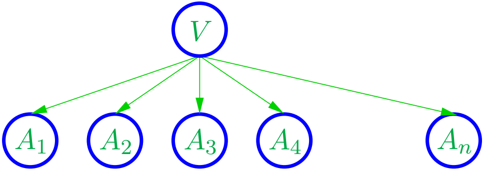


General classification – and the Naïve Bayes Classifier

Assume target function $f : \mathcal{X} \rightarrow V$, where each instance x described by attributes $\langle a_1, a_2 \ldots a_n \rangle$. Most probable value of $f(x)$:

$$v_{\mathsf{MAP}} = \underset{v_j \in V}{\operatorname{argmax}} P(v_j | a_1, a_2 \ldots a_n)$$

$$\begin{aligned} v_{\mathsf{MAP}} &= \underset{v_j \in V}{\operatorname{argmax}} \frac{P(a_1, a_2 \ldots a_n | v_j) P(v_j)}{P(a_1, a_2 \ldots a_n)} \\ &= \underset{v_j \in V}{\operatorname{argmax}} P(a_1, a_2 \ldots a_n | v_j) P(v_j) \end{aligned}$$



General classification – and the Naïve Bayes Classifier

Assume target function $f : \mathcal{X} \rightarrow V$, where each instance x described by attributes $\langle a_1, a_2 \ldots a_n \rangle$. Most probable value of $f(x)$:

$$
\begin{aligned}
v_{\text{MAP}} &= \operatorname*{argmax}_{v_j \in V} P(v_j | a_1, a_2 \ldots a_n) \\
v_{\text{MAP}} &= \operatorname*{argmax}_{v_j \in V} \frac{P(a_1, a_2 \ldots a_n | v_j) P(v_j)}{P(a_1, a_2 \ldots a_n)} \\
&= \operatorname*{argmax}_{v_j \in V} P(a_1, a_2 \ldots a_n | v_j) P(v_j)
\end{aligned}
$$

Naïve Bayes assumption: $P(a_1, a_2 \ldots a_n | v_j) = \prod_i P(a_i | v_j)$, which gives

Naïve Bayes classifier: $v_{\text{NB}} = \operatorname*{argmax}_{v_j \in V} P(v_j) \prod_i P(a_i | v_j)$

Naïve Bayes Algorithm

`Naïve_Bayes_Learn(examples)`

- For each target value v_j
  - $\hat{P}(v_j) \leftarrow$ estimate $P(v_j)$
  - For each attribute value a_i of each attribute a
    - $\hat{P}(a_i|v_j) \leftarrow$ estimate $P(a_i|v_j)$

$\Longrightarrow$ Learning in $\mathcal{O}\left(\#\text{classes } \#\text{attributes } \#\text{training-examples}\right)$

`Classify_New_Instance(`x`)`

$$v_{\text{NB}} = \operatorname*{argmax}_{v_j \in V} \hat{P}(v_j) \prod_{a_i \in x} \hat{P}(a_i|v_j)$$

$\Longrightarrow$ Classification in $\mathcal{O}\left(\#\text{classes } \#\text{attributes}\right)$

Naïve Bayes: Subtleties

Naïve Bayes uses Conditional independence assumption to justify

$$
\begin{aligned}
P(a_i, a_j|v) &= P(a_i|a_j, v)P(a_j|v) \\
&= P(a_i|v)P(a_j|v)
\end{aligned}
$$

It is often violated, but NB works surprisingly well anyway.

Why?

We don't need estimated posteriors $\hat{P}(v_j|x)$ to be correct; only that

$$\operatorname*{argmax}_{v_j \in V} \hat{P}(v_j) \prod_i \hat{P}(a_i|v_j) = \operatorname*{argmax}_{v_j \in V} P(v_j)P(a_1 \ldots, a_n|v_j)$$

$\rightarrow$ Naïve Bayes posteriors are typically unrealistically close to 0 or 1, but this will often not harm classification ability!

Example: Learning to Classify Text

Why consider how to classify text?

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

Naïve Bayes is among most effective algorithms

Important question:

What attributes shall we use to represent text documents?

Learning to Classify Text - Definition

Target concept `Interesting?`: `Document` $\rightarrow \{\oplus, \ominus\}$

1. Represent each document by vector of words
   - one attribute per word position in document
2. Learning: Use training examples to estimate
   - $P(\oplus)$
   - $P(\ominus)$
   - $P(\texttt{doc}|\oplus)$
   - $P(\texttt{doc}|\ominus)$

Naïve Bayes conditional independence assumption

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

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

One more assumption: $P(a_i = w_k|v_j) = P(a_m = w_k|v_j), \forall i, m$

Naïve Bayes algorithm - learning

`Learn_Naïve_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. *Estimate the $P(v_j)$ and $P(w_k|v_j)$ probability terms*

- For each target value v_j in V do
  - $\texttt{docs}_j \leftarrow$ subset of `Examples` for which the target value is v_j
  - $P(v_j) \leftarrow \frac{|\texttt{docs}_j|}{|\texttt{Examples}|}$
  - $\texttt{Text}_j \leftarrow$ a single document created by concatenating all members of $\texttt{docs}_j$
  - $n \leftarrow$ total number of words in $\texttt{Text}_j$ (counting duplicate words multiple times)
  - for each word w_k in `Vocabulary`
    - $n_k \leftarrow$ number of times word w_k occurs in $\texttt{Text}_j$
    - $P(w_k|v_j) \leftarrow \frac{n_k}{n}$

Naïve Bayes algorithm - classification

`Classify_Naïve_Bayes_text(Doc)`

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

$$v_{\mathsf{NB}} = \operatorname*{argmax}_{v_j \in V} P(v_j) \prod_{i \in \mathtt{positions}} P(a_i | v_j)$$

Twenty NewsGroups

Given 1000 training documents from each group

Learn to classify new documents according to which newsgroup it came from

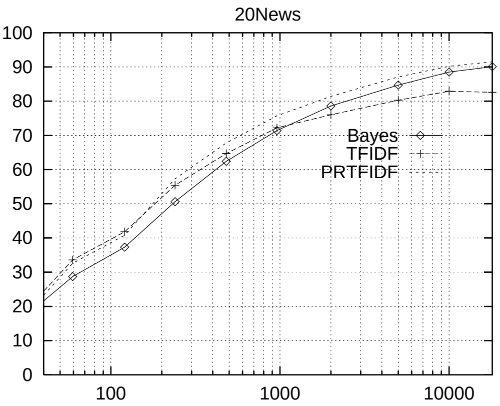
comp.graphics	misc.forsale
comp.os.ms-windows.misc	rec.autos
comp.sys.ibm.pc.hardware	rec.motorcycles
comp.sys.mac.hardware	rec.sport.baseball
comp.windows.x	rec.sport.hockey
alt.atheism	sci.space
soc.religion.christian	sci.crypt
talk.religion.misc	sci.electronics
talk.politics.mideast	sci.med
talk.politics.misc	
talk.politics.guns	

Article from rec.sport.hockey

```
From: xxx@yyy.zzz.edu (John Doe)
Subject: Re: This year's biggest and worst
Date: 5 Apr 93 09:53:39 GMT
```

I can only comment on the Kings, but the most obvious candidate for pleasant surprise is Alex Zhitnik. He came highly touted as a defensive defenseman, but he's clearly much more than that. Great skater and hard shot (though wish he were more accurate). In fact, he pretty much allowed the Kings to trade away that huge defensive liability Paul Coffey. Kelly Hrudey is only the biggest disappointment if you thought he was any good to begin with. But, at best, he's only a mediocre goaltender. A better choice would be Tomas Sandstrom, though not through any fault of his own, but because some thugs in Toronto decided

Learning Curve for 20 Newsgroups



Accuracy vs. Training set size

Incomplete data

How do we handle cases with missing values:

- Faulty sensor readings.
- Values have been intentionally removed.
- Some variables may be unobservable.

Why don't we just throw away the cases with missing values?

Incomplete data

How do we handle cases with missing values:

- Faulty sensor readings.
- Values have been intentionally removed.
- Some variables may be unobservable.

Why don't we just throw away the cases with missing values?

A	B	A	B
a_1	b_1	a_2	b_1
a_1	b_2	a_2	b_1
a_1	b_2	a_2	b_2
a_1	b_1	a_2	b_1
a_1	b_1	a_2	b_1
a_1	b_2	a_2	?
a_1	b_1	a_2	?
a_1	b_2	a_2	?
a_1	b_1	a_2	?
a_1	b_2	a_2	?

$\Rightarrow$

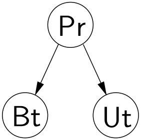
Using the entire database:

$$\hat{P}(a_1) = \frac{N(a_1)}{N(a_1) + N(a_2)} = \frac{10}{10+10} = 1/2.$$

Having removed the cases with missing values:

$$\hat{P}'(a_1) = \frac{N'(a_1)}{N'(a_1) + N'(a_2)} = \frac{10}{10+5} = 2/3.$$

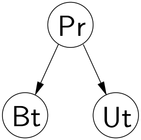
The EM algorithm



Cases	Pr	Bt	Ut
1.	?	pos	pos
2.	yes	neg	pos
3.	yes	pos	?
4.	yes	pos	neg
5.	?	neg	?

Estimate the required probability distributions for the network

The EM algorithm



Cases	Pr	Bt	Ut
1.	?	pos	pos
2.	yes	neg	pos
3.	yes	pos	?
4.	yes	pos	neg
5.	?	neg	?

If the database was complete we would estimate the required probabilities, $P(\text{Pr})$, $P(\text{Ut}|\text{Pr})$ and $P(\text{Bt}|\text{Pr})$ as:

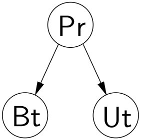
$$P(\text{Pr} = \text{yes}) = \frac{N(\text{Pr} = \text{yes})}{N}$$

$$P(\text{Ut} = \text{yes}|\text{Pr} = \text{yes}) = \frac{N(\text{Ut} = \text{yes}, \text{Pr} = \text{yes})}{N(\text{Pr} = \text{yes})}$$

$$P(\text{Bt} = \text{yes}|\text{Pr} = \text{no}) = \frac{N(\text{Bt} = \text{yes}, \text{Pr} = \text{no})}{N(\text{Pr} = \text{no})}$$

So estimating the probabilities is basically a counting problem!

The EM algorithm



Cases	Pr	Bt	Ut
1.	?	pos	pos
2.	yes	neg	pos
3.	yes	pos	?
4.	yes	pos	neg
5.	?	neg	?

Estimate $P(\text{Pr})$ from the database above:

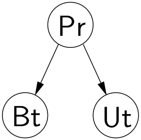
Case 2, 3 and 4 contributes with a value 1 to $N(\text{Pr} = \text{yes})$, but what is the contribution from case 1 and 5?

- Case 1 contributes with $P(\text{Pr} = \text{yes} | \text{Bt} = \text{pos}, \text{Ut} = \text{pos})$.
- Case 5 contributes with $P(\text{Pr} = \text{yes} | \text{Bt} = \text{neg})$.

To find these probabilities we assume some initial distributions, $P_0(\cdot)$, have been assigned to the network.

We are basically calculating the expectation for $N(\text{Pr} = \text{yes})$, denoted $\mathbb{E}[N(\text{Pr} = \text{yes})]$

The EM algorithm



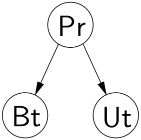
Cases	Pr	Bt	Ut
1.	?	pos	pos
2.	yes	neg	pos
3.	yes	pos	?
4.	yes	pos	neg
5.	?	neg	?

Using $P_0(\text{Pr}) = (0.5, 0.5)$, $P_0(\text{Bt}|\text{Pr} = \text{yes}) = (0.5, 0.5)$ etc., as starting distributions we get:

$$\begin{aligned}
\mathbb{E}[N(\text{Pr} = \text{y})] &= P_0(\text{Pr} = \text{y}|\text{Bt} = \text{Ut} = \text{pos}) + 1 + 1 + 1 \\
&\quad + P_0(\text{Pr} = \text{y}|\text{Bt} = \text{neg}) \\
&= 0.5 + 1 + 1 + 1 + 0.5 = 4 \\
\mathbb{E}[N(\text{Pr} = \text{no})] &= P_0(\text{Pr} = \text{no}|\text{Bt} = \text{Ut} = \text{pos}) + 0 + 0 + 0 \\
&\quad + P_0(\text{Pr} = \text{no}|\text{Bt} = \text{neg}) \\
&= 0.5 + 0 + 0 + 0 + 0.5 = 1
\end{aligned}$$

So we e.g. get $\hat{P}_1(\text{Pr} = \text{yes}) = \mathbb{E}[N(\text{Pr} = \text{yes})]/N = 0.8$.

The EM algorithm



Cases	Pr	Bt	Ut
1.	?	pos	pos
2.	yes	neg	pos
3.	yes	pos	?
4.	yes	pos	neg
5.	?	neg	?

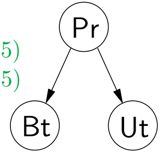
To estimate $\hat{P}_1(\mathsf{Ut}|\mathsf{Pr}) = \mathbb{E}[N(\mathsf{Ut},\mathsf{Pr})]/\mathbb{E}[N(\mathsf{Pr})]$ we e.g. need:

$$\mathbb{E}[N(\mathsf{Ut}=\mathsf{p},\mathsf{Pr}=\mathsf{y})] = P_0(\mathsf{Ut}=\mathsf{p},\mathsf{Pr}=\mathsf{y}|\mathsf{Bt}=\mathsf{Ut}=\mathsf{p}) + 1$$
$$+ P_0(\mathsf{Ut}=\mathsf{p},\mathsf{Pr}=\mathsf{y}|\mathsf{Bt}=\mathsf{p},\mathsf{Pr}=\mathsf{y}) + 0 + P_0(\mathsf{Ut}=\mathsf{p},\mathsf{Pr}=\mathsf{y}|\mathsf{Bt}=\mathsf{n})$$
$$= 0.5 + 1 + 0.5 + 0 + 0.25 = 2.25$$
$$\mathbb{E}[N(\mathsf{Pr}=\mathsf{yes})] = P_0(\mathsf{Pr}=\mathsf{yes}|\mathsf{Bt}=\mathsf{Ut}=\mathsf{pos}) + 1 + 1 + 1$$
$$+ P_0(\mathsf{Pr}=\mathsf{yes}|\mathsf{Bt}=\mathsf{neg}) = 0.5 + 1 + 1 + 1 + 0.5 = 4$$

$$\hat{P}_1(\mathsf{Ut}=\mathsf{pos}|\mathsf{Pr}=\mathsf{yes}) = \frac{\mathbb{E}[N(\mathsf{Ut}=\mathsf{p},\mathsf{Pr}=\mathsf{y})]}{\mathbb{E}[N(\mathsf{Pr}=\mathsf{yes})]} = \frac{2.25}{4} = 0.5625$$

The EM algorithm

$P_0(Pr) = (0.5, 0.5)$
$P_0(Ut = \text{pos}|Pr) = (0.5, 0.5)$
$P_0(Bt = \text{pos}|Pr) = (0.5, 0.5)$

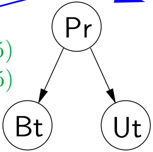


Cases	Pr	Bt	Ut
1.	?	pos	pos
2.	yes	neg	pos
3.	yes	pos	?
4.	yes	pos	neg
5.	?	neg	?

The EM algorithm

E-step 1

$P_0(Pr) = (0.5, 0.5)$
$P_0(Ut = \text{pos}|Pr) = (0.5, 0.5)$
$P_0(Bt = \text{pos}|Pr) = (0.5, 0.5)$

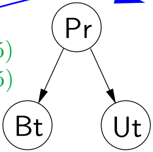


$\mathbb{E}_0[N(Pr)] = (4, 1)$
$\mathbb{E}_0[N(Ut = \text{pos}, Pr)] = (2.25, 0.5 + 0 + 0 + 0 + 0.25)$
$\mathbb{E}_0[N(Bt = \text{pos}, Pr)] = (0.5 + 0 + 1 + 1 + 0 = 2.5$
$, 0.5 + 0 + 0 + 0 + 0 = 0.5)$

Cases	Pr	Bt	Ut
1.	?	pos	pos
2.	yes	neg	pos
3.	yes	pos	?
4.	yes	pos	neg
5.	?	neg	?

The EM algorithm

$P_0(Pr) = (0.5, 0.5)$
$P_0(Ut = \text{pos}|Pr) = (0.5, 0.5)$
$P_0(Bt = \text{pos}|Pr) = (0.5, 0.5)$



E-step 1

$\mathbb{E}_0[N(Pr)] = (4, 1)$
$\mathbb{E}_0[N(Ut = \text{pos}, Pr)] = (2.25, 0.5 + 0 + 0 + 0 + 0.25)$
$\mathbb{E}_0[N(Bt = \text{pos}, Pr)] = (0.5 + 0 + 1 + 1 + 0 = 2.5$
$, 0.5 + 0 + 0 + 0 + 0 = 0.5)$

M-step 2

$P_1(Pr) = (\frac{4}{5}, \frac{1}{5})$
$P_1(Ut = \text{pos}|Pr) = (\frac{2.25}{4}, \frac{0.75}{1})$
$P_1(Bt = \text{pos}|Pr) = (\frac{2.5}{4}, \frac{0.5}{1})$

Cases	Pr	Bt	Ut
1.	?	pos	pos
2.	yes	neg	pos
3.	yes	pos	?
4.	yes	pos	neg
5.	?	neg	?

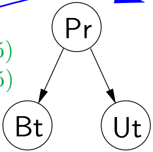
The EM algorithm

E-step 1

$P_0(Pr) = (0.5, 0.5)$

$P_0(Ut = \text{pos}|Pr) = (0.5, 0.5)$

$P_0(Bt = \text{pos}|Pr) = (0.5, 0.5)$



$\mathbb{E}_0[N(Pr)] = (4, 1)$

$\mathbb{E}_0[N(Ut = \text{pos}, Pr)] = (2.25, 0.5 + 0 + 0 + 0 + 0.25)$

$\mathbb{E}_0[N(Bt = \text{pos}, Pr)] = (0.5 + 0 + 1 + 1 + 0 = 2.5$
$, 0.5 + 0 + 0 + 0 + 0 = 0.5)$

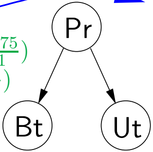
M-step 2

E-step 3

$P_1(Pr) = (\frac{4}{5}, \frac{1}{5})$

$P_1(Ut = \text{pos}|Pr) = (\frac{2.25}{4}, \frac{0.75}{1})$

$P_1(Bt = \text{pos}|Pr) = (\frac{2.5}{4}, \frac{0.5}{1})$



$\mathbb{E}_1[N(Pr)] = (\quad , \quad)$

$\mathbb{E}_1[N(Ut = \text{pos}, Pr)] = (\quad , \quad)$

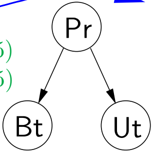
$\mathbb{E}_1[N(Bt = \text{pos}, Pr)] = (\quad , \quad)$

Cases	Pr	Bt	Ut
1.	?	pos	pos
2.	yes	neg	pos
3.	yes	pos	?
4.	yes	pos	neg
5.	?	neg	?

The EM algorithm

E-step 1

$P_0(Pr) = (0.5, 0.5)$
$P_0(Ut = \text{pos}|Pr) = (0.5, 0.5)$
$P_0(Bt = \text{pos}|Pr) = (0.5, 0.5)$

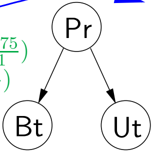


$\mathbb{E}_0[N(Pr)] = (4, 1)$
$\mathbb{E}_0[N(Ut = \text{pos}, Pr)] = (2.25, 0.5 + 0 + 0 + 0 + 0.25)$
$\mathbb{E}_0[N(Bt = \text{pos}, Pr)] = (0.5 + 0 + 1 + 1 + 0 = 2.5$
$, 0.5 + 0 + 0 + 0 + 0 = 0.5)$

M-step 2

E-step 3

$P_1(Pr) = (\frac{4}{5}, \frac{1}{5})$
$P_1(Ut = \text{pos}|Pr) = (\frac{2.25}{4}, \frac{0.75}{1})$
$P_1(Bt = \text{pos}|Pr) = (\frac{2.5}{4}, \frac{0.5}{1})$



$\mathbb{E}_1[N(Pr)] = (\quad , \quad)$
$\mathbb{E}_1[N(Ut = \text{pos}, Pr)] = (\quad , \quad)$
$\mathbb{E}_1[N(Bt = \text{pos}, Pr)] = (\quad , \quad)$

M-step 4

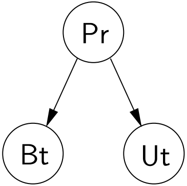
$P_2(Pr) = (\cdot, \cdot)$
$P_2(Ut = \text{pos}|Pr) = (\cdot, \cdot)$
$P_2(Bt = \text{pos}|Pr) = (\cdot, \cdot)$

$\vdots$

Until convergence

Cases	Pr	Bt	Ut
1.	?	pos	pos
2.	yes	neg	pos
3.	yes	pos	?
4.	yes	pos	neg
5.	?	neg	?

The EM algorithm



Cases	Pr	Bt	Ut
1.	?	pos	pos
2.	yes	neg	pos
3.	yes	pos	?
4.	yes	pos	neg
5.	?	neg	?

- Let $\boldsymbol{\theta}^0 = \{\theta_{ijk}\}$ be start estimates ($P(X_i = j|\mathrm{pa}(X_i) = k) = \theta_{ijk}$).
- Repeat until convergence:
  - **E-step:** For each variable X_i calculate the table of expected counts: $\mathbb{E}_{\boldsymbol{\theta}^t}[N(X_i, \mathrm{pa}(X_i)\,|\mathcal{D}] = \sum_{\boldsymbol{d} \in \mathcal{D}} P(X_i, \mathrm{pa}(X_i)\,|\boldsymbol{d}, \boldsymbol{\theta}^t)$.
  - **M-step:** Use the expected counts as if they were actual counts: $\hat{\theta}_{ijk} = \frac{\mathbb{E}_{\boldsymbol{\theta}^i}[N(X_i=k,\mathrm{pa}(X_i)=j|\mathcal{D}]}{\sum_{k=1}^{|sp(X_i)|}\mathbb{E}_{\boldsymbol{\theta}^i}[N(X_i=k,\mathrm{pa}(X_i)=j|\mathcal{D}]}$.

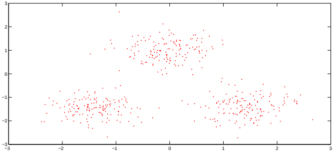
EM for Estimating k Gaussians

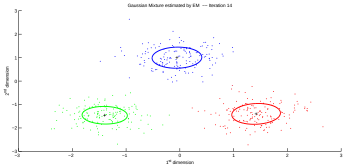
Given:

- Instances from $\mathcal{X}$ generated by mixture of k Gaussian distributions with unknown means
- Unknown means $\langle \boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_k \rangle$ of the k Gaussians
- Don't know which instance $\boldsymbol{x}_i$ was generated by which Gaussian

Determine:

- Maximum likelihood estimates of $\langle \boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_k \rangle$

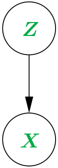




EM for Estimating k Gaussians (2)

Think of each instance as $\boldsymbol{y}_i = \langle \boldsymbol{x}_i, z_{i1}, \ldots, z_{ik} \rangle$, where

- z_{ij} is 1 if $\boldsymbol{x}_i$ generated by jth Gaussian
- $\boldsymbol{x}_i$ observable, z_{ij} unobservable



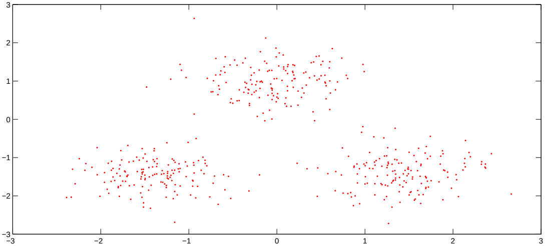
Cases	$\boldsymbol{x}$	$\boldsymbol{z}$
1.	(1.2, 2.0)	(?,?,?)
2.	(2.2, 0.1)	(?,?,?)
3.	(2.3, 0.1)	(?,?,?)
4.	(-0.2, -1.2)	(?,?,?)
5.	(6.7, -3.0)	(?,?,?)

Idea:

Use the EM algorithm to learn the Maximum Likelihood parameters. As a side-effect we "fill in" values for $\langle z_{i1}, \ldots, z_{ik} \rangle$, which are the labels for each observation's mixture belonging.

EM for Estimating k Gaussians (3)

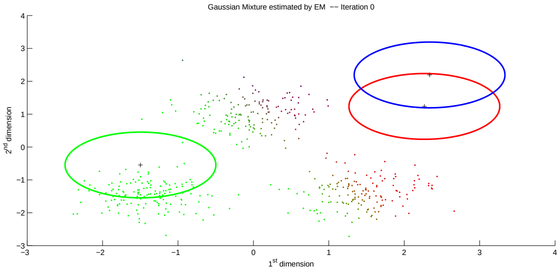
Initialize: $\mu_1, \ldots, \mu_k$ picked on random



EM for Estimating k Gaussians (3)

Initialize: $\mu_1, \ldots, \mu_k$ picked on random

Calculate the *responsibilities* each Gaussian takes for each datapoint

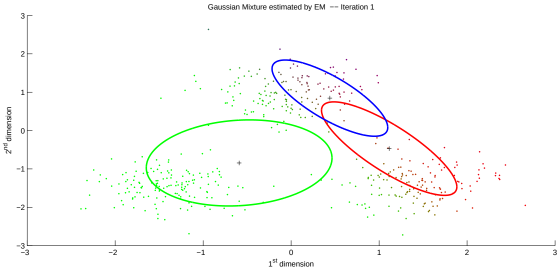


EM for Estimating k Gaussians (3)

Initialize: $\mu_1, \ldots, \mu_k$ picked on random

Calculate the *responsibilities* each Gaussian takes for each datapoint

Update μ_j based on those datapoints Gaussian j takes responsibility for



EM for Estimating k Gaussians (3)

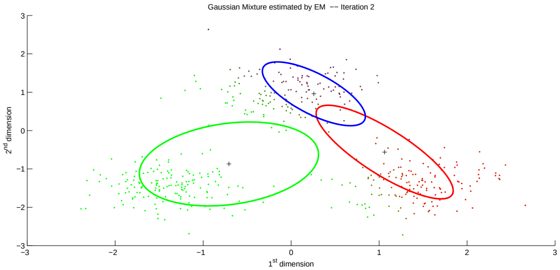
Initialize: $\mu_1, \ldots, \mu_k$ picked on random

Calculate the *responsibilities* each Gaussian takes for each datapoint

Update μ_j based on those datapoints Gaussian j takes responsibility for

Calculate the *responsibilities* each Gaussian takes for each datapoint

And so on ...



EM for Estimating k Gaussians (3)

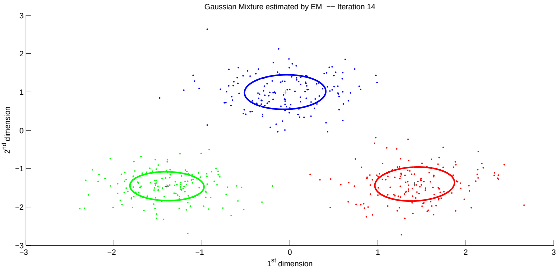
Initialize: $\mu_1, \ldots, \mu_k$ picked on random

Calculate the *responsibilities* each Gaussian takes for each datapoint

Update μ_j based on those datapoints Gaussian j takes responsibility for

Calculate the *responsibilities* each Gaussian takes for each datapoint

And so on . . .
Until convergence



EM for Estimating k Gaussians (4)

EM Algorithm: Pick random initial $h = \langle \boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_k \rangle$, then iterate

E step: Calculate the expected value $E[z_{ij}]$ of each hidden variable z_{ij}, assuming the current hypothesis $h = \langle \boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_k \rangle$ holds.

$$
\begin{aligned}
E[z_{ij}] &= \frac{p(\boldsymbol{x} = \boldsymbol{x}_i | \boldsymbol{\mu} = \boldsymbol{\mu}_j)}{\sum_{n=1}^{k} p(\boldsymbol{x} = \boldsymbol{x}_i | \boldsymbol{\mu} = \boldsymbol{\mu}_n)} \\
p(\boldsymbol{x} = \boldsymbol{x}_i | \boldsymbol{\mu} = \boldsymbol{\mu}_j) &\propto \exp(-||\boldsymbol{x}_i - \boldsymbol{\mu}_j||^2)
\end{aligned}
$$

M step: Calculate a new maximum likelihood hypothesis assuming the value taken on by each hidden variable z_{ij} is its expected value $E[z_{ij}]$ calculated above:
$h \leftarrow \langle \boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_k \rangle$, where

$$
\boldsymbol{\mu}_j \leftarrow \frac{\sum_{i=1}^{m} E[z_{ij}] \;\; \boldsymbol{x}_i}{\sum_{i=1}^{m} E[z_{ij}]}
$$

Summary

- Bayesian Learning
  - Combine prior knowledge with observed data
  - Impact of prior knowledge (when correct!) is to lower the sample complexity
- Bayesian Networks
  - Learn parameters using counting (complete data) or EM (incomplete data)
  - Important special case: *Naïve Bayes Classifier*