

# Intro. to machine learning (CSI 5325)

## Lecture 14: Bayesian learning

Greg Hamerly

Some content from Tom Mitchell.

- 1 Naive bayes classifier example: Learning over text data
- 2 Bayesian belief networks
- 3 Expectation Maximization algorithm

# 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

- SVMs are even better...
- but NB is much simpler...

What attributes shall we use to represent text documents?

# Learning to Classify Text

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

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

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

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

# Learning to Classify Text

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
- 3 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  $Text_j$
    - $P(w_k|v_j) \leftarrow \frac{n_k+1}{n+|Vocabulary|}$

# Learning to Classify Text

Classify\_naive\_Bayes\_text(*Doc*)

- *positions* ← all word positions in *Doc* that contain tokens found in *Vocabulary*
- Return  $v_{NB}$ , where

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

# Bag of Words model

The representation of a text document as a...

- vector of a fixed vocabulary
- where each element indicates the presence or count of the word in the document
- and each word's position is discarded

is called the 'bag of words' model. Also known as 'unigram.'

Why is it useful? What does it lose?

Alternative methods which also discard position but improve context: bi-grams, tri-grams, etc.

## 20 news groups dataset

Given 1000 training documents from each group, learn to classify new documents according to which newsgroup it came from.

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

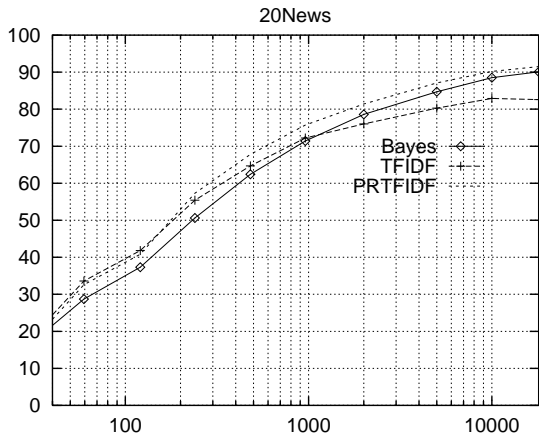
Naive Bayes achieves 89% classification accuracy.

# from rec.sport.hockey

```
Path: cantaloupe.srv.cs.cmu.edu!das-news.harvard.edu!logicse!uwm.edu
From: xxx@yyy.zzz.edu (John Doe)
Subject: Re: This year's biggest and worst (opinion)...
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

# Curve for 20 Newsgroups



Accuracy vs. Training set size (1/3 withheld for test)

## Naive Bayes with other probability distributions

Usually, the naive Bayes classifier is described with histogram-type probability distributions, i.e.

$$P(w|v) = \frac{|\text{times } w \text{ seen in class } v|}{|\text{times class } v \text{ seen}|}$$

We could alternatively use any probability distribution like Gaussian, binomial, etc.

We could even use a different distribution for each attribute.

## Naive Bayes classifier applications

### Text classification

- Spam classification
- Finding articles 'of interest'
- News article classification (juggling!?)
- Identifying 'high-quality' posts in a forum
- Grading papers?

Failure prediction (e.g. hard disk drives, motors, software, etc.)

Others?

## Conditional independence

The NB classifier builds a distribution of the attributes for each class.

- $k$  classes  $\rightarrow k$  distributions

Each distribution considers the attributes to be independent, **given the class**.

However, it doesn't mean that the attributes are considered completely independent!

That is,

$$P(a_1, a_2|v) = P(a_1|v)P(a_2|v)$$

but **not**

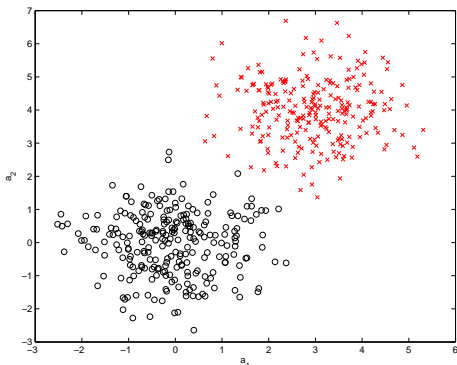
$$P(a_1, a_2) = P(a_1)P(a_2)$$

## Conditional independence example

$$P(a_1, a_2 | \text{red}) = P(a_1 | \text{red})P(a_2 | \text{red})$$

but **not**

$$P(a_1, a_2) = P(a_1)P(a_2)$$



# Bayesian Belief Networks

Interesting because:

- Naive Bayes assumption of conditional independence too restrictive
  - But learning is intractable without some such 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 Nets)

## 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

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

## Conditional Independence

$X$  is conditionally independent of  $Y$  given  $Z$ :

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

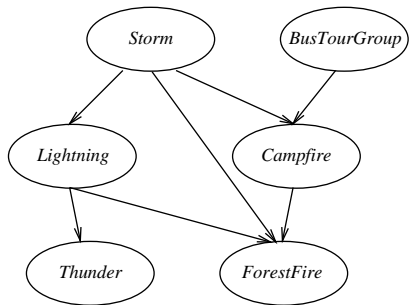
Example: *Thunder* cond. independent of *Rain*, given *Lightning*

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

Naive Bayes uses conditional independence to justify

$$\begin{aligned} P(X, Y|Z) &= P(X|Y, Z)P(Y|Z) \\ &= P(X|Z)P(Y|Z) \end{aligned}$$

# Bayesian Belief Network

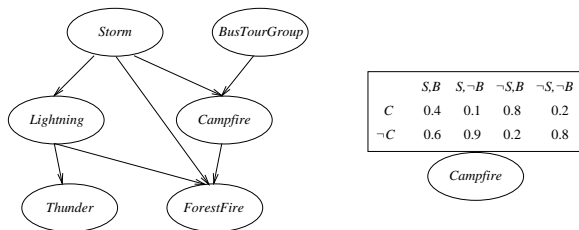


	$S, B$	$S, \neg B$	$\neg S, B$	$\neg S, \neg B$
$C$	0.4	0.1	0.8	0.2
$\neg C$	0.6	0.9	0.2	0.8



Network represents a set of conditional independence assertions:

- Each node is asserted to be conditionally independent of its nondescendants, given its immediate predecessors.
- Directed acyclic graph



Represents joint probability distribution over all variables

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

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

$\text{Parents}(Y_i)$  denotes immediate predecessors of  $Y_i$  in graph

- joint dist. fully defined by graph and  $P(y_i | \text{Parents}(Y_i))$

# 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, can succeed in many cases

- Exact inference methods work well for some network structures
- Monte Carlo methods “simulate” the network randomly to calculate approximate solutions

# Learning of Bayesian Networks

Several variants of this learning task

- Network structure might be *known* or *unknown*
- Training examples might provide values of *all* network variables, or just *some*

If structure known and observe all variables

- Then it's as easy as training a Naive Bayes classifier

# Learning Bayes Nets

Suppose structure known, variables partially observable

e.g., observe *ForestFire*, *Storm*, *BusTourGroup*, *Thunder*, but not *Lightning*, *Campfire*...

- Similar to training neural network with hidden units
- In fact, we can learn network conditional probability tables using gradient ascent.
- Converge to network  $h$  that (locally) maximizes  $P(D|h)$

# Gradient Ascent for Bayes Nets

Let  $w_{ijk}$  denote one entry in the conditional probability table for variable  $Y_i$  in the network

$$w_{ijk} = P(Y_i = y_{ij} | Parents(Y_i) = \text{the list } u_{ik} \text{ of values})$$

e.g., if  $Y_i = \text{Campfire}$ , then  $u_{ik}$  might be  $\langle \text{Storm} = T, \text{BusTourGroup} = F \rangle$

Perform gradient ascent by repeatedly

- 1 update all  $w_{ijk}$  using training data  $D$

$$w_{ijk} \leftarrow w_{ijk} + \eta \sum_{d \in D} \frac{P_h(y_{ij}, u_{ik} | d)}{w_{ijk}}$$

- 2 then, renormalize the  $w_{ijk}$  to assure

- $\sum_j w_{ijk} = 1$
- $0 \leq w_{ijk} \leq 1$

## More on Learning Bayes Nets

EM algorithm can also be used. Repeatedly:

- 1 Calculate probabilities of unobserved variables, assuming  $h$
- 2 Calculate new  $w_{ijk}$  to maximize  $E[\ln P(D|h)]$  where  $D$  now includes both observed and (calculated probabilities of) unobserved variables

When structure unknown...

- Algorithms use greedy search to add/subtract edges and nodes

## Summary: Bayesian Belief Networks

- Combine prior knowledge with observed data
- Impact of prior knowledge (when correct!) is to lower the sample complexity
- Further extensions:
  - Extend from boolean to real-valued variables
  - Parameterized distributions instead of tables
  - Extend to first-order instead of propositional systems
  - More effective inference methods
  - ...

# Expectation Maximization (EM)

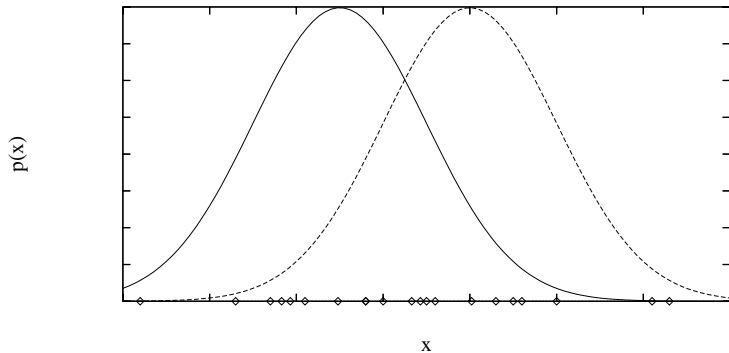
When to use:

- Data is only partially observable
- Unsupervised clustering (target value unobservable)
- Supervised learning (some instance attributes unobservable)

Some uses:

- Train Bayesian Belief Networks
- Unsupervised clustering (AUTOCLASS)
- Learning Hidden Markov Models

# Generating Data from Mixture of $k$ Gaussians



Each instance  $x$  generated by

- 1 Choosing one of the  $k$  Gaussians with uniform probability
- 2 Generating an instance at random according to that Gaussian

## EM for Estimating $k$ Means

Given:

- Instances from  $X$  generated by mixture of  $k$  Gaussian distributions
- Unknown means  $\langle \mu_1, \dots, \mu_k \rangle$  of the  $k$  Gaussians
- Don't know which instance  $x_i$  was generated by which Gaussian

Determine:

- Maximum likelihood estimates of  $\langle \mu_1, \dots, \mu_k \rangle$

Think of each instance as  $y_i = \langle x_i, z_{i1}, z_{i2} \rangle$ , where

- $z_{ij}$  is 1 if  $x_i$  generated by  $j$ th Gaussian
- $x_i$  observable
- $z_{ij}$  unobservable

# (Mitchell's) EM for Estimating $k$ Means

EM Algorithm: Pick random initial  $h = \langle \mu_1, \mu_2 \rangle$ , then iterate

**E step:** Calculate the expected value  $E[z_{ij}]$  of each hidden variable  $z_{ij}$ , assuming the current hypothesis  $h = \langle \mu_1, \mu_2 \rangle$  holds.

$$\begin{aligned}
 E[z_{ij}] &= \frac{p(x = x_i | \mu = \mu_j)}{\sum_{n=1}^2 p(x = x_i | \mu = \mu_n)} \\
 &= \frac{e^{-\frac{1}{2\sigma^2}(x_i - \mu_j)^2}}{\sum_{n=1}^2 e^{-\frac{1}{2\sigma^2}(x_i - \mu_n)^2}}
 \end{aligned}$$

**M step:** Calculate a new maximum likelihood hypothesis  $h' = \langle \mu'_1, \mu'_2 \rangle$ , assuming the value taken on by each hidden variable  $z_{ij}$  is its expected value  $E[z_{ij}]$  calculated above. Replace  $h = \langle \mu_1, \mu_2 \rangle$  by  $h' = \langle \mu'_1, \mu'_2 \rangle$ .

$$\mu_j \leftarrow \frac{\sum_{i=1}^m E[z_{ij}] x_i}{\sum_{i=1}^m E[z_{ij}]}$$

## The 'real' $k$ -means algorithm

Choose  $k$  initial centers  $\mu_1, \mu_2, \dots, \mu_k$ .

Repeat until convergence:

**E-step:** Assign each example  $x_i$  to its closest center  $\mu_j$ . That is,

$$E[z_{ij}] = \begin{cases} 1 & \text{if } j = \operatorname{argmax}_{j'} \|x_i - \mu_{j'}\| \\ 0 & \text{otherwise} \end{cases}$$

**M-step:** Update each center  $\mu_j$  to be the mean of its assigned points:

$$\mu_j = \frac{\sum_i E[z_{ij}] x_i}{\sum_i E[z_{ij}]}$$

Difference from Mitchell:  $z_{ij} \in \{0, 1\}$  versus  $z_{ij} \in [0, 1]$ .

Another way to look at it:  $\sigma^2 = 0$  for each Gaussian.

# EM Algorithm

Converges to local maximum likelihood  $h$   
and provides estimates of hidden variables  $z_{ij}$

In fact, local maximum in  $E[\ln P(Y|h)]$

- $Y$  is complete (observable plus unobservable variables) data
- Expected value is taken over possible values of unobserved variables in  $Y$

# General EM Problem

Given:

- Observed data  $X = \{x_1, \dots, x_m\}$
- Unobserved data  $Z = \{z_1, \dots, z_m\}$
- Parameterized probability distribution  $P(Y|h)$ , where
  - $Y = \{y_1, \dots, y_m\}$  is the full data  $y_i = x_i \cup z_i$
  - $h$  are the parameters

Determine:

- $h$  that (locally) maximizes  $E[\ln P(Y|h)]$

Many uses:

- Train Bayesian belief networks
- Unsupervised clustering (e.g.,  $k$  means)
- Hidden Markov Models

## General EM Method

Define likelihood function  $Q(h'|h)$  which calculates  $Y = X \cup Z$  using observed  $X$  and current parameters  $h$  to estimate  $Z$

$$Q(h'|h) \leftarrow E[\ln P(Y|h')|h, X]$$

EM Algorithm:

*Estimation (E) step:* Calculate  $Q(h'|h)$  using the current hypothesis  $h$  and the observed data  $X$  to estimate the probability distribution over  $Y$ .

$$Q(h'|h) \leftarrow E[\ln P(Y|h')|h, X]$$

*Maximization (M) step:* Replace hypothesis  $h$  by the hypothesis  $h'$  that maximizes this  $Q$  function.

$$h \leftarrow \underset{h'}{\operatorname{argmax}} Q(h'|h)$$

## EM example – waiting for the bus

If it comes, you take it; otherwise you get tired and take a cab.

- let  $T$  be a random variable indicating the time you spend waiting for the bus
- assume that bus arrives according to exponential distribution:

$$Pr[T = t] = \lambda e^{-\lambda t}$$

- exponential distribution has Markov property of history independence:

$$Pr[T = t_1 + t_2 | T > t_1] = Pr[T = t_2]$$

Observe: 

wait 5 mins, bus arrives	wait 8 mins, take a cab
wait 7 mins, bus arrives	wait 4 mins, take a cab

What is  $\lambda$ ?

## EM example – waiting for the bus

Observe: 

wait 5 mins, bus arrives		wait 8 mins, take a cab
wait 7 mins, bus arrives		wait 4 mins, take a cab

What is  $\lambda$ ? Apply the EM algorithm:

- Initialize: guess  $\lambda = 5$
- E-step: compute expected value of each waiting time  
→ for censored values, add  $\lambda$  (due to Markov property)
- M-step: compute maximum likelihood value of  $\lambda$   
→ estimate  $\lambda$  based on average of data
- repeat E and M steps

## EM clustering example

Matlab demo...