## Bayesian Learning

- **Bayesian learning algorithms** use probability theory as an approach to concept classification.
- Bayesian classifiers generally produce probabilities for (possibly multiple) class assignments, rather than a single definite classification.
- Provides practical learning algorithms: Naive Bayes learning.
- Provides useful conceptual framework.
- Algorithms often computationally intractable.
- Provides "gold standard" for evaluating other learning algorithms.
- To use Bayesian techniques, we often need to make independence assumptions that often aren't valid.



## Prior Probability and Random Variables

- P(A) represents the **prior** or **unconditional** probability that statement A is true, in the absence of other information.
- A **random variable** represents different outcomes of an "event". Each random variable has a domain of possible values $x_1 \ldots x_n$, which each have a probability. The probabilities for all possible outcomes must sum to 1. For example:

  P(Disease=CAVITY) = 0.5
  P(Disease=GUM_DISEASE) = 0.3
  P(Disease=IMPACTED_TOOTH) = 0.1
  P(Disease=ROOT_INFECTION) = 0.1

**Conditional Probability**

P($A \mid B$) represents the probability of $A$ given that $B$ is known to be true. We call this a **conditional** or **posterior** probability.

P($A \mid B$) = 1 is equivalent to $B \Rightarrow A$.

---

For example, suppose Rover rarely howls:

P($Rover\_howls$) = 0.01

But when there is a full moon, he always howls!
Then P($Rover\_howls \mid full\_moon$) = 1.0

**Slide CS478–3**

**The Chain Rule**

$$P(A \mid B) = \frac{P(A \wedge B)}{P(B)}$$

We can rewrite this as: $P(A \wedge B) = P(A \mid B) * P(B)$
which is called **the chain rule** because we can chain together probabilities to compute the likelihood of conjunctions.

Example:
$P(A) = .0.5$
$P(B \mid A) = 0.6$
$P(C \mid A \wedge B) = 0.8$

What is $P(A \wedge B \wedge C)$?

**Slide CS478–4**

**Basic Formulas for Probabilities**

- **Product (or "Chain") Rule**: probability of a conjunction of two events:

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

- **Sum Rule**: probability of a disjunction of two events:

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

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

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

**Slide CS478–5**

**Bayes Rule**

Simple form:

$$P(A \mid B) = \frac{P(A)*P(B|A)}{P(B)}$$

General form:

$$P(A \mid B, X) = \frac{P(A|X)*P(B|A,X)}{P(B|X)}$$

**Slide CS478–6**

**Bayes Theorem applied to machine learning**

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

$P(h)$ = 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 $h$

**Slide CS478–7**

**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 $h_{MAP}$:

$$h_{MAP} = argmax_{h \in H} \ P(h|D)$$
$$h_{MAP} = argmax_{h \in H} \ \frac{P(D|h)P(h)}{P(D)}$$
$$h_{MAP} = argmax_{h \in H} \ P(D|h)P(h)$$

If assume $P(h_i) = P(h_j)$ then can further simplify. The **Maximum Likelihood** hypothesis:
$h_{ML} = argmax_{h_i \in H} \ P(D|h_i)$

**Slide CS478–8**

**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.

$$P(cancer) = \qquad P(\neg cancer) =$$
$$P(+|cancer) = \qquad P(-|cancer) =$$
$$P(+|\neg cancer) = \qquad P(-|\neg cancer) =$$

**Slide CS478–9**

$$P(cancer) = .008 \qquad P(\neg cancer) = .992$$
$$P(+|cancer) = .98 \qquad P(-|cancer) = .02$$
$$P(+|\neg cancer) = .03 \qquad P(-|\neg cancer) = .97$$

$h_{MAP} = argmax_{h \in H} \;\; P(D|h)P(h)$
$h_i$ is CANCER:

$h_i$ is $\neg$CANCER:

$h_{MAP} =$

**Slide CS478–10**

**Brute Force MAP Hypothesis Learner**

- For each hypothesis $h$ in $H$, calculate the posterior probability:

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

- Output the hypothesis $h_{MAP}$ with the highest posterior probability:

$$h_{MAP} = argmax_{h \in H} \quad P(h|D)$$



**Relation to Concept Learning**

Assume fixed set of instances $\langle x_1, \ldots, x_m \rangle$

Assume $D$ is the set of classifications $D = \langle c(x_1), \ldots, c(x_m) \rangle$

Choose $P(D|h)$:

- $P(D|h) = 1$ if $h$ consistent with $D$
- $P(D|h) = 0$ otherwise

Choose $P(h)$ to be *uniform* distribution

- $P(h) = \frac{1}{|H|}$ for all $h$ in $H$

Choose $P(D)$:

- $P(D) = \frac{|VS_{H,D}|}{|H|}$

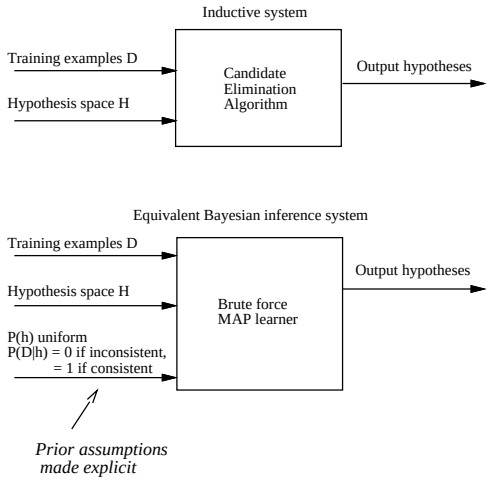
Then,

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

$$P(h|D) = \begin{cases} \frac{1}{|VS_{H,D}|} & \text{if } h \text{ is consistent with } D \\ 0 & \text{otherwise} \end{cases}$$

**Slide CS478–13**

## Characterizing Learning Algorithms by Equivalent MAP Learners




**Slide CS478–14**

## Most Probable Classification of New Instances

So far we've sought the most probable *hypothesis* given the data $D$ (i.e., $h_{MAP}$)

Given new instance $x$, what is its most probable *classification*?

$h_{MAP}(x)$ **is <u>not</u> the most probable classification!**

---

Consider this hypothesis space:

$P(h_1|D) = .4,\ P(h_2|D) = .3,\ P(h_3|D) = .3$

Given new instance $x$ $h_1(x) = +,\ h_2(x) = -,\ h_3(x) = -$

What's the most probable classification of $x$?

**Slide CS478–15**

## Bayes Optimal Classifier

$$argmax_{c_j \in C} \sum_{h_i \in H} P(h_i|D)P(c_j|h_i)$$

Example:

$$P(h_1|D) = .4, \quad P(-|h_1) = 0, \quad P(+|h_1) = 1$$
$$P(h_2|D) = .3, \quad P(-|h_2) = 1, \quad P(+|h_2) = 0$$
$$P(h_3|D) = .3, \quad P(-|h_3) = 1, \quad P(+|h_3) = 0$$

therefore

$$\sum_{h_i \in H} P(h_i|D)P(+|h_i) = .4$$
$$\sum_{h_i \in H} P(h_i|D)P(-|h_i) = .6 \textit{ and}$$

$$\arg\max_{c_j \in C} \sum_{h_i \in H} P(c_j|h_i)P(h_i|D) = -$$

**Slide CS478–16**

**Gibbs Classifier**

Bayes optimal classifier provides best result, but can be expensive if many hypotheses.

Gibbs algorithm:

1. Choose one hypothesis at random, according to $P(h|D)$
2. Use this to classify new instance

Surprising fact: Assume target concepts are drawn at random from $H$ according to priors on $H$. Then:

$$E[error_{Gibbs}] \leq 2E[error_{BayesOptimal}]$$

**Slide CS478–17**

**Naive Bayes Classifier**

- One of the most practical learning methods, along with decision trees, neural networks, nearest neighbor methods, etc.
- Requires:
  1. Moderate or large training set.
  2. Attributes that describe instances should be conditionally independent of the classification.
- Successful applications include diagnosis and text classification.

**Slide CS478–18**

**Naive Bayes Classifier**

Assume target function $f : X \rightarrow C$, where each instance $x$ is described by attributes $\langle a_1, a_2 \ldots a_n \rangle$.

Most probable value of $f(x)$ is:

$$c_{MAP} = argmax_{c_j \in C} \; P(c_j | a_1, a_2 \ldots a_n)$$

$$c_{MAP} = argmax_{c_j \in C} \; \frac{P(a_1, a_2 \ldots a_n | c_j) P(c_j)}{P(a_1, a_2 \ldots a_n)}$$

$$c_{MAP} = argmax_{c_j \in C} \; P(a_1, a_2 \ldots a_n | c_j) P(c_j)$$



**Naive Bayes Assumption**

To make the problem tractable, we often need to make the following independence assumption:

$$P(a_1, a_2 \ldots a_n | c_j) = \prod_i P(a_i | c_j)$$

which allows us to define the **Naive Bayes Classifier:**

$$c_{NB} = argmax_{c_j \in C} \;\; P(c_j) \prod_i P(a_i | c_j)$$

**Naive Bayes Algorithm**

Naive_Bayes_Learn($examples$)

For each possible class $c_j$

$\hat{P}(c_j) \leftarrow$ estimate $P(c_j)$

For each attribute value $a_i$ of each attribute $a$

$\hat{P}(a_i|c_j) \leftarrow$ estimate $P(a_i|c_j)$

---

Classify_New_Instance($x$)

$$c_{NB} = argmax_{c_j \in C} \hat{P}(c_j) \prod_{a_i \in x} \hat{P}(a_i|c_j)$$

**Slide CS478–21**

**Naive Bayes: Example**

Consider *PlayTennis* again, and new instance

$$\langle Outlk = sun, Temp = cool, Humid = high, Wind = strong \rangle$$

Want to compute:

$$c_{NB} = argmax_{c_j \in C} P(c_j) \prod_i P(a_i|c_j)$$

$$P(y)\ P(sun|y)\ P(cool|y)\ P(high|y)\ P(strong|y) = .005$$

$$P(n)\ P(sun|n)\ P(cool|n)\ P(high|n)\ P(strong|n) = .021$$

$$\rightarrow c_{NB} = n$$

**Slide CS478–22**

**Naive Bayes: Subtleties**

- Conditional independence assumption is often violated

$$P(a_1, a_2 \ldots a_n | c_j) = \prod_i P(a_i | c_j)$$

  ...but it works surprisingly well anyway.

- Naive Bayes posteriors often unrealistically close to 1 or 0



What if none of the training instances with class $c_j$ have attribute value $a_i$? Then

$$\hat{P}(a_i | c_j) = 0$$

$$so \quad \hat{P}(c_j) \prod_i \hat{P}(a_i | c_j) = 0$$

---

Typical solution is m-estimate for $\hat{P}(a_i|c_j) \leftarrow \frac{n_c + mp}{n+m}$ where

$n$ is number of training examples for which $c = c_j$

$n_c$ number of examples for which $c = c_j$ and $a = a_i$

$p$ is prior estimate for $\hat{P}(a_i|c_j)$

$m$ is weight given to prior (i.e. # of "virtual" examples)

**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??

**Slide CS478–25**

**Learning to Classify Text**

Target concept *Interesting*? : *Document*$\{+, -\}$

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

**Slide CS478–26**

Naive Bayes conditional independence assumption

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

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

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

**Slide CS478–27**

LEARN_NAIVE_BAYES_TEXT($Examples, C$)

1. *collect all words and other tokens that occur in Examples*
   - $Vocab \leftarrow$ ALL DISTINCT WORDS AND OTHER TOKENS IN $Examples$
2. *calculate the required $P(c_j)$ and $P(w_k|c_j)$ probability terms*
   - FOR EACH TARGET VALUE $c_j$ IN $C$ DO
     - $docs_j \leftarrow$ SUBSET OF $Examples$ FOR WHICH THE TARGET VALUE IS $c_j$
     - $P(c_j) \leftarrow \frac{|docs_j|}{|Examples|}$
     - $Text_j \leftarrow$ A SINGLE DOCUMENT CREATED BY CONCATENATING ALL MEMBERS OF $docs_j$
     - $n$ TOTAL NUMBER OF WORDS IN $Text_j$ (COUNTING DUPLICATE WORDS MULTIPLE TIMES)
     - FOR EACH WORD $w_k$ IN $Vocab$
       - $n_k \leftarrow$ NUMBER OF TIMES WORD $w_k$ OCCURS IN $Text_j$
       - $P(w_k|c_j) \leftarrow \frac{n_k+1}{n+|Vocab|}$

**Slide CS478–28**

CLASSIFY_NAIVE_BAYES_TEXT($Doc$)

- $positions \leftarrow$ ALL WORD POSITIONS IN $Doc$ THAT CONTAIN TOKENS FOUND IN $Vocab$
- RETURN $c_{NB}$, WHERE

$$c_{NB} = argmax_{c_j \in C} P(c_j) \prod_{i \in positions} P(a_i|c_j)$$

**Slide CS478–29**

**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 |

Naive Bayes: 89% classification accuracy

**Slide CS478–30**

## Article from rec.sport.hockey

```
Path: cantaloupe.srv.cs.cmu.edu!das-news.harvard.edu!ogicse!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
mediocre goaltender. A better choice would be
Tomas Sandstrom, though not through any fault of
his own, but because some thugs in Toronto decided
```

**Slide CS478–31**

## Learning Curve for 20 Newsgroups

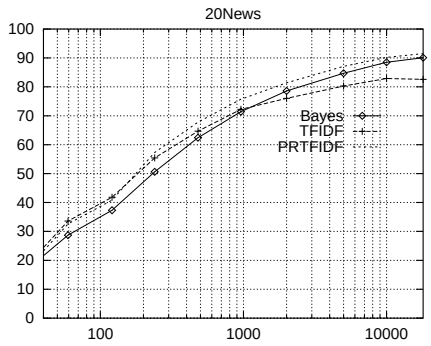



Figure 1: Accuracy vs. Training set size (1/3 withheld for test)

**Slide CS478–32**

**Statistical Speech Recognition Model**

What is the most likely sequence of words that the speech signal represents?

$$P(words \mid signal) = \frac{P(words)P(signal|words)}{P(signal)}$$

- $P(words)$ is the **language model**. For example, "fat cat" is more likely than "hat cat".
- $P(signal \mid words)$ is the **acoustic model**. For example, "cat" is likely to be pronounced as *kæt*.
- $P(signal)$ is the likelihood of the speech signal.



**The Language Model**

- Ideally, we'd like a complete model of the English language to tell us the exact probability of a sentence.
- But there is no complete model of English. (Or any other natural language.) You'd need to know the probability of every sentence that could ever be uttered!
- A grammar should be helpful, but there is no complete grammar for English and spoken language is notoriously ungrammatical anyway.
- Statistical speech recognition systems approximate the likelihood of sentences by collecting n-gram statistics of word usage from large spoken language corpora.

## N-grams

- A **unigram** represents a single word. We estimate:
  $P(w)$ = *frequency of w in corpus / number of words in corpus.*
- A **bigram** represents a pair of sequential words. We estimate:
  $P(w_2 \mid w_1)$ = *frequency of* $w_2$ *following* $w_1$ */ frequency of* $w_1$.
- A **trigram** represents a triple of sequential words. We estimate:
  $P(w_3 \mid w_1, w_2)$ = *frequency of* $w_3$ *following* $w_1$,$w_2$ */ frequency of* $w_{1,2}$.

**Slide CS478–35**

## Unigram/Bigram Statistics

| Word | Unigram count | Previous words | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | OF | IN | IS | ON | TO | FROM | THAT | WITH | LINE | VISION |
| THE | 367 | 179 | 143 | 44 | 44 | 65 | 35 | 30 | 17 | 0 | 0 |
| ON | 69 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| OF | 281 | 0 | 0 | 2 | 0 | 1 | 0 | 3 | 0 | 4 | 0 |
| TO | 212 | 0 | 0 | 19 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| IS | 175 | 0 | 0 | 0 | 0 | 0 | 0 | 13 | 0 | 1 | 3 |
| A | 153 | 36 | 36 | 33 | 23 | 21 | 14 | 3 | 15 | 0 | 0 |
| THAT | 124 | 0 | 3 | 18 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| WE | 105 | 0 | 0 | 0 | 1 | 0 | 0 | 12 | 0 | 0 | 0 |
| LINE | 17 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| VISION | 13 | 3 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 |

**Slide CS478–36**

**The N-gram language model**

Ideally, we'd like to compute:

$P(w_1...w_n) = P(w_1)P(w_2 \mid w_1)P(w_3 \mid w_1w_2)...P(w_n \mid w_1...w_{n-1})$

But we rarely have the data necessary to compute those statistics reliably. So we estimate by making independence assumptions and using bigrams (or trigams).

The bigram model is:

$P(w_1...w_n) = P(w_1)P(w_2 \mid w_1)P(w_3 \mid w_2)...P(w_n \mid w_{n-1})$

$$P(w_1...w_n) = \prod_{i=1}^{n} P(w_i \mid w_{i-1})$$