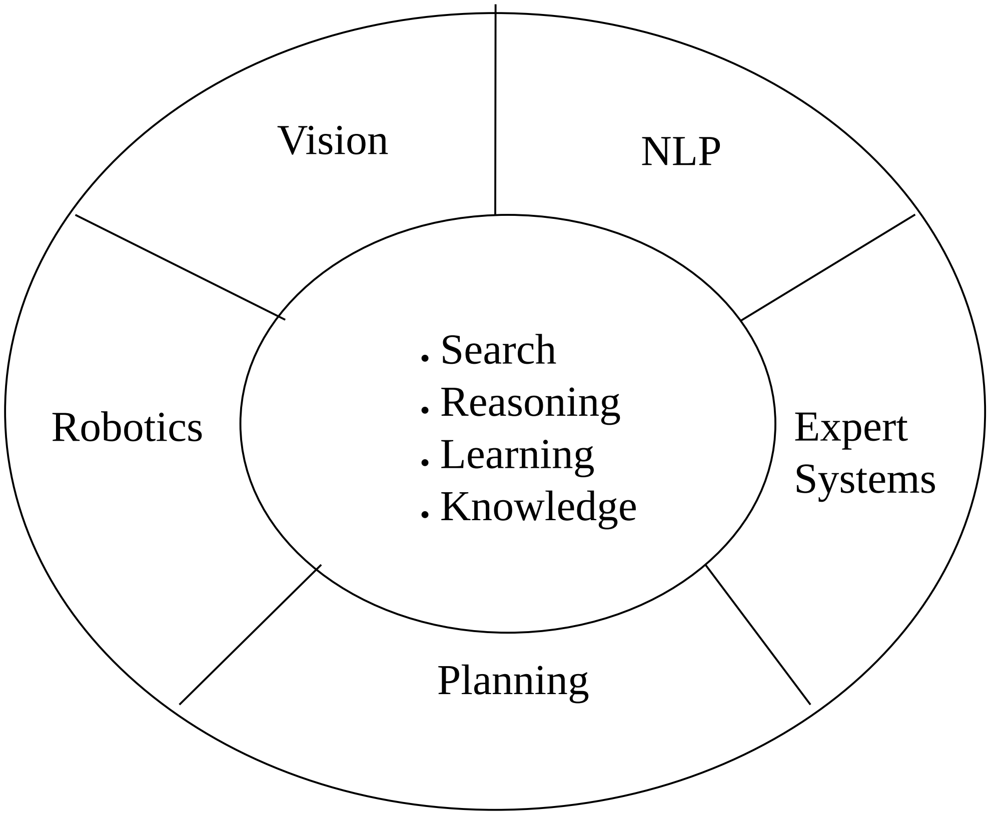
CS 564: Machine Learning

Naïve Bayes Classification

AI: The various Components



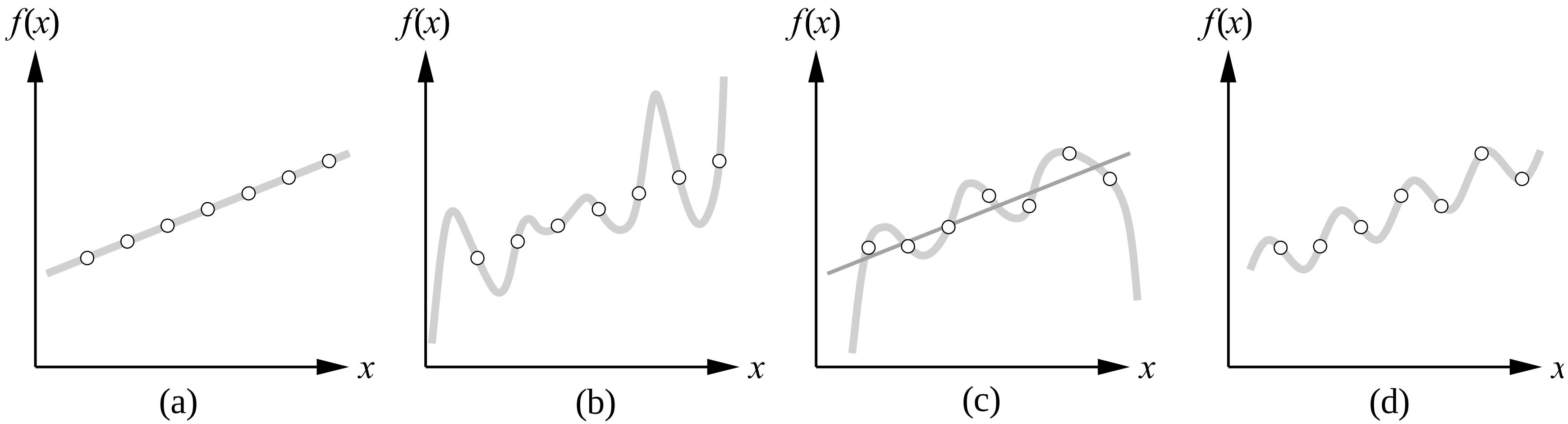
Machine Learning

- Machine learning: how to acquire a model on the basis of data / experience?
  - Learning parameters (e.g. probabilities)
  - Learning structure (e.g. BN graphs)
  - Learning hidden concepts (e.g. clustering)

Machine Learning

- Unsupervised Learning
  - No feedback from teacher; detect patterns
- Reinforcement Learning
  - Feedback consists of rewards/punishment
- Supervised Learning
  - Examples of correct answers are given
  - Discrete answers: *Classification*
  - Continuous answers: *Regression*

Supervised Machine Learning



Given a training set:

$(x_1, y_1), (x_2, y_2), (x_3, y_3), \ldots (x_n, y_n)$

Where each y_i was generated by an unknown $y = f(x)$,
Discover a function h that approximates the true function f

Example: Spam Filter

- Input: x = email
- Output: y = “spam” or “ham”
- Setup:
  - Get a large collection of example emails, each labeled “spam” or “ham”
  - Note: someone has to hand label all this data!
  - Want to learn to predict labels of new, future emails

- Features: The attributes used to make the ham / spam decision
  - Words: FREE!
  - Text Patterns: $dd, CAPS
  - Non-text: SenderInContacts
  - …

> Dear Sir.
>
> First, I must solicit your confidence in this transaction, this is by virture of its nature as being utterly confidencial and top secret. …

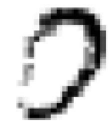
> TO BE REMOVED FROM FUTURE MAILINGS, SIMPLY REPLY TO THIS MESSAGE AND PUT "REMOVE" IN THE SUBJECT.
>
> 99 MILLION EMAIL ADDRESSES FOR ONLY $99

> Ok, Iknow this is blatantly OT but I'm beginning to go insane. Had an old Dell Dimension XPS sitting in the corner and decided to put it to use, I know it was working pre being stuck in the corner, but when I plugged it in, hit the power nothing happened.

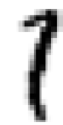
Example: Digit Recognition

- Input: x = images (pixel grids)
- Output: y = a digit 0-9
- Setup:
  - Get a large collection of example images, each labeled with a digit
  - Note: someone has to hand label all this data!
  - Want to learn to predict labels of new, future digit images

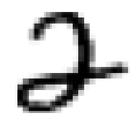
- Features: The attributes used to make the digit decision
  - Pixels: (6,8)=ON
  - Shape Patterns: NumComponents, AspectRatio, NumLoops
  - …



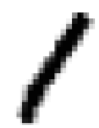
0



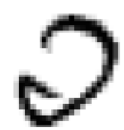
1



2



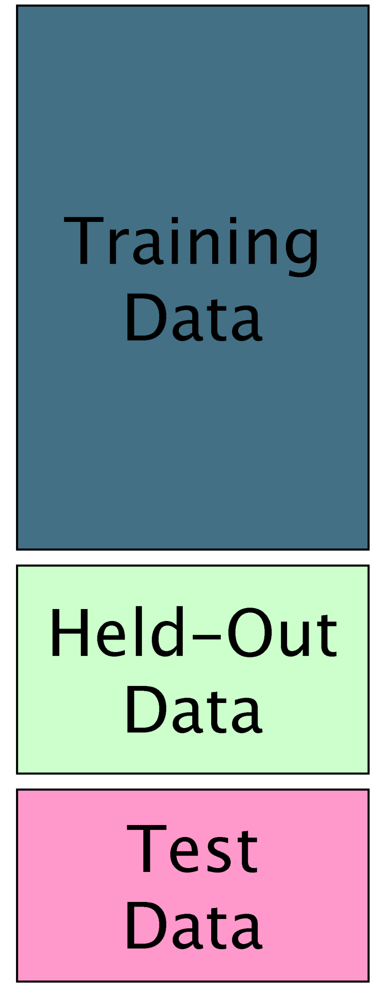
1



??

How to Learn

- Data: labeled instances, e.g. emails marked spam/ham
  - Training set
  - Held out (validation) set
  - Test set
- Features: attribute-value pairs which characterize each x
- Experimentation cycle
  - Learn parameters (e.g. model probabilities) on training set
  - Tune hyperparameters on held-out set
  - Compute accuracy on test set
  - Very important: never “peek” at the test set!
- Evaluation
  - Accuracy: fraction of instances predicted correctly
- Overfitting and generalization
  - Want a classifier which does well on *test* data
  - Overfitting: fitting the training data very closely, but not generalizing well to test data

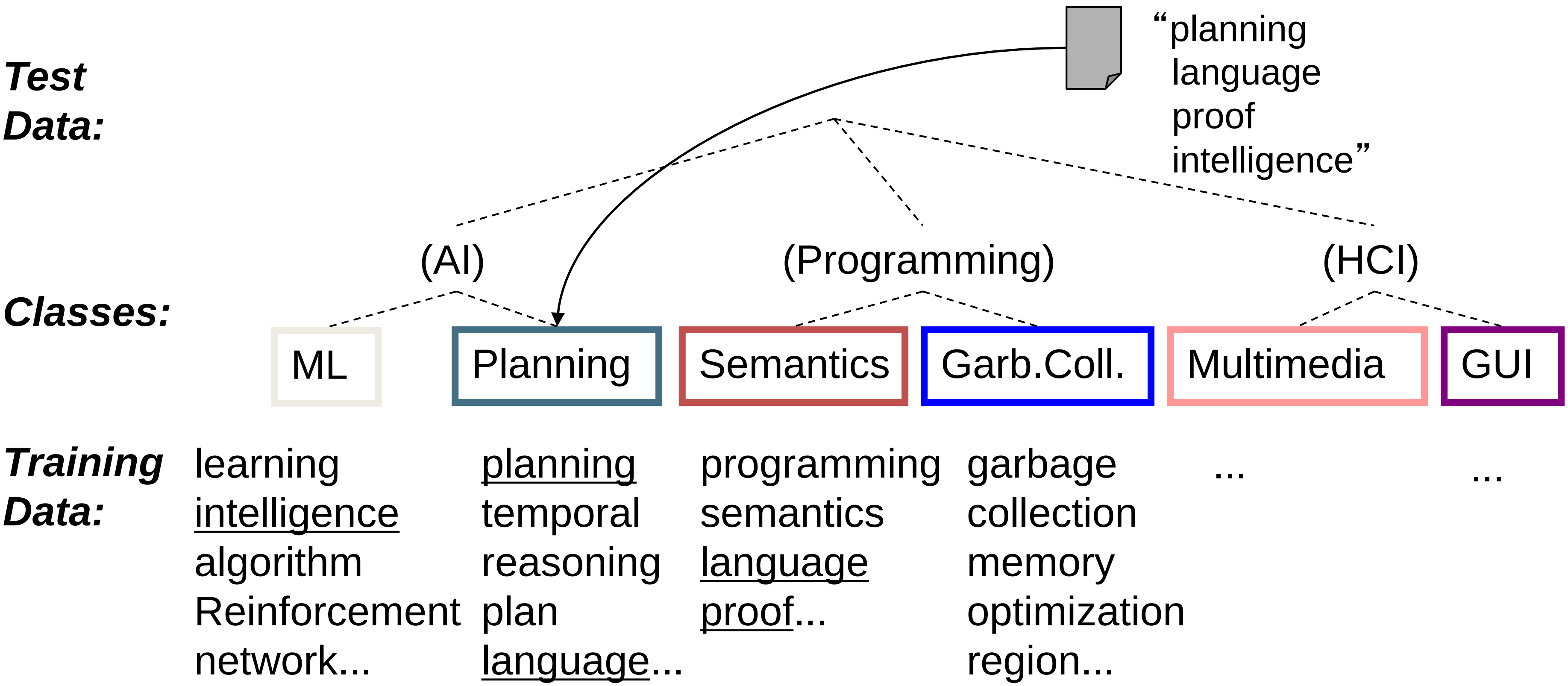


Supervised Classification

- Given:
  - A description of an instance, $d \in X$
    - X is the *instance language* or *instance space*.
  - A fixed set of classes:

    $C = \{c_1, c_2, \ldots, c_J\}$
  - A training set D of labeled documents with each labeled document $\langle d,c \rangle \in X \times C$
- Determine:
  - A learning method or algorithm which will enable us to learn a classifier $\gamma: X \rightarrow C$
  - For a test document d, we assign it the class $\gamma(d) \in C$

Document Classification



More Text Classification Examples

Many search engine functionalities use classification

Assigning labels to documents or web-pages:

- Labels are most often topics such as Yahoo-categories
  - *"finance," "sports," "news>world>asia>business"*
- Labels may be genres (or, categories)
  - *"editorials" "movie-reviews" "news"*
- Labels may be opinion on a person/product
  - *"like", "hate", "neutral"*
- Labels may be domain-specific
  - *"interesting-to-me" : "not-interesting-to-me"*
  - *language identification: English, French, Chinese, …*
  - *search vertical: about Linux versus not*
  - *"link spam" : "not link spam"*

Classification Methods (1)

- Manual classification
  - Used by the original Yahoo! Directory
  - Looksmart, about.com, ODP, PubMed
  - Very accurate when job is done by experts
  - Consistent when the problem size and team is small
  - Difficult and expensive to scale
    - Means we need automatic classification methods for big problems

Classification Methods (2)

- Automatic classification
  - Hand-coded rule-based systems
    - One technique used by Reuters, CIA, etc.
    - It's what Google Alerts is doing
      - Widely deployed in government and enterprise
    - Companies provide "IDE" (integrated development environment) for writing such rules
    - E.g., assign category if document contains a given boolean combination of words
    - Standing queries: Commercial systems have complex query languages (everything in IR query languages +score accumulators)
    - Accuracy is often very high if a rule has been carefully refined over time by a subject expert
    - Building and maintaining these rules is expensive
    - Rules could vary with the change of domain

Classification Methods (3)

- *Supervised learning of a document*-label assignment function
  - Many systems *partly rely on machine learning* (Autonomy, Microsoft, Enkata, Yahoo!, Google News, …)
    - k-Nearest Neighbors (simple, powerful)
    - Naive Bayes (simple, common method)
    - Support-vector machines (new, more powerful)
    - … plus many other methods
    - Requirement: requires hand-classified training data
    - But data can be built up (and refined) by amateurs
- Many commercial systems use a mixture of methods

Recall a few probability basics

- For events *a* and *b:*
- Bayes' Rule

$$p(a,b) = p(a \cap b) = p(a \mid b)\, p(b) = p(b \mid a)\, p(a)$$

$$p(\bar{a} \mid b)\, p(b) = p(b \mid \bar{a})\, p(\bar{a})$$

$$\underbrace{p(a \mid b)}_{\text{Posterior}} = \frac{p(b \mid a)\, p(a)}{p(b)} = \frac{p(b \mid a)\, \overbrace{p(a)}^{\text{Prior}}}{\sum_{x=a,\bar{a}} p(b \mid x)\, p(x)}$$

- Odds: $O(a) = \dfrac{p(a)}{p(\bar{a})} = \dfrac{p(a)}{1 - p(a)}$

Probabilistic Methods

- Learning and classification methods based on probability theory
- Bayes theorem plays a critical role in probabilistic learning and classification
- Builds a *generative model* that approximates how data is produced
- Uses *prior* probability of each category given no information about an item
- Categorization produces a *posterior* probability distribution over the possible categories given a description of an item

Bayes' Rule for text classification

- For a document *d* and a class *c*

$$P(c,d) = P(c \mid d)P(d) = P(d \mid c)P(c)$$

$$\boxed{P(c \mid d) = \frac{P(d \mid c)P(c)}{P(d)}}$$

Naive Bayes Classifiers

Task: Classify a new instance *d* based on a tuple of attribute values into one of the classes $c_j \in C$

$$d = \rangle x_1, x_2, \ldots, x_n \langle$$

$$c_{MAP} = \underset{c_j \in C}{\operatorname{argmax}} P(c_j \mid x_1, x_2, \ldots, x_n)$$

$$= \underset{c_j \in C}{\operatorname{argmax}} \frac{P(x_1, x_2, \ldots, x_n \mid c_j) P(c_j)}{P(x_1, x_2, \ldots, x_n)}$$

$$= \underset{c_j \in C}{\operatorname{argmax}} P(x_1, x_2, \ldots, x_n \mid c_j) P(c_j)$$

MAP is "maximum a posteriori" = most likely class

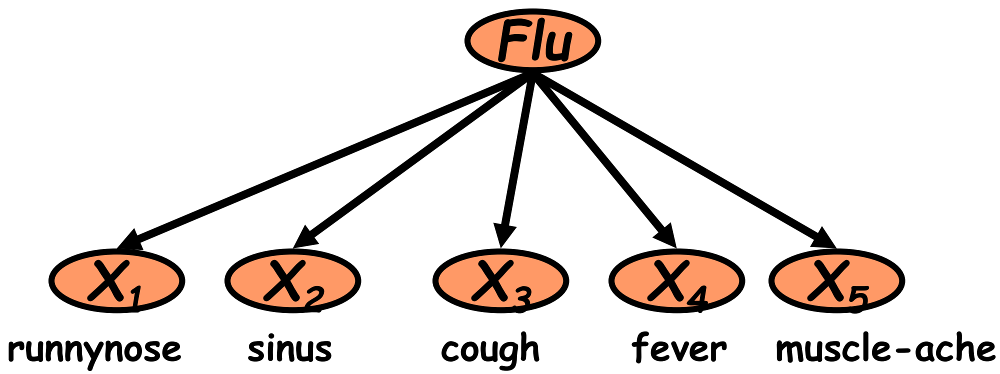
Naive Bayes Classifier: Naive Bayes Assumption

- $P(c_j)$
  - Can be estimated from the frequency of classes in the training examples
- $P(x_1,x_2,\ldots,x_n|c_j)$
  - $O(|X|^n \bullet |C|)$ parameters
  - Could only be estimated if a very, very large number of training examples were available

Naive Bayes Conditional Independence Assumption:

- Assume that the probability of observing the conjunction of attributes is equal to the product of the individual probabilities $P(x_i|c_j)$

The Naive Bayes Classifier

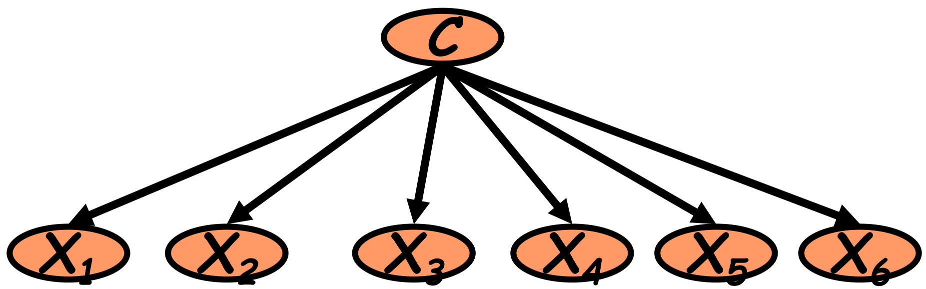


- **Conditional Independence Assumption:** features detect term presence and are **independent** of each other **given the class**:

$$P(X_1,\ldots,X_5 \mid C) = P(X_1 \mid C) \bullet P(X_2 \mid C) \bullet \cdots \bullet P(X_5 \mid C)$$

- This model is appropriate for binary variables
  - Multivariate Bernoulli model

Learning the Model

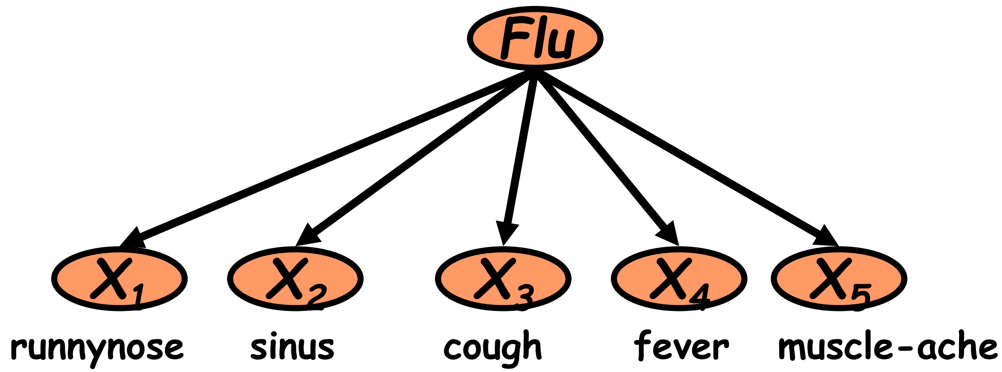


- First attempt: maximum likelihood estimates
  - simply use the frequencies in the data

$$\hat{P}(c_j) = \frac{N(C = c_j)}{N}$$

$$\hat{P}(x_i \mid c_j) = \frac{N(X_i = x_i, C = c_j)}{N(C = c_j)}$$

Problem with Maximum Likelihood



$$P(X_1,\ldots,X_5 \mid C) = P(X_1 \mid C) \bullet P(X_2 \mid C) \bullet \cdots \bullet P(X_5 \mid C)$$

- What if we have seen no training documents with the ***fever*** and classified in the topic ***Flu***?

$$\hat{P}(X_4 = t \mid C = Flu) = \frac{N(X_4 = t, C = Flu)}{N(C = Flu)} = 0$$

- Zero probabilities cannot be conditioned away, no matter the other evidence!

$$\ell = \arg\max_c \hat{P}(c) \prod_i \hat{P}(x_i \mid c)$$

Smoothing to Avoid Overfitting

$$\hat{P}(x_i \mid c_j) = \frac{N(X_i = x_i, C = c_j) + 1}{N(C = c_j) + k}$$

# of values of X_i

- Somewhat more subtle version

overall fraction in data where X_i=$x_{i,k}$

$$\hat{P}(x_{i,k} \mid c_j) = \frac{N(X_i = x_{i,k}, C = c_j) + mp_{i,k}}{N(C = c_j) + m}$$

extent of "smoothing"

Generalization and Overfitting

- Raw counts will overfit the training data!
  - Just because we never saw a 3 with pixel (15,15) on during training doesn't mean we won't see it at test time
  - Unlikely that every occurrence of "minute" is 100% spam
  - Unlikely that every occurrence of "seriously" is 100% ham
  - What about all the words that don't occur in the training set at all? 0/0?
  - In general, we can't go around giving unseen events zero probability

- At the extreme, imagine using the entire email as the only feature
  - Would get the training data perfect (if deterministic labeling)
  - Wouldn't *generalize* at all
  - Just making the bag-of-words assumption gives us some generalization, but isn't enough

- To generalize better: we need to smooth or regularize the estimates

- **Regularization** involves introducing additional information in order to solve an ill-posed problem or to prevent **overfitting**

- From a Bayesian point of view: imposing certain prior distributions on model parameters

Estimation: Smoothing

- Maximum likelihood estimates:

$$P_{\mathsf{ML}}(x) = \frac{\mathsf{count}(x)}{\mathsf{total\ samples}}$$

$$P_{\mathsf{ML}}(\mathsf{r}) = 1/3$$

- Problems with maximum likelihood estimates:
  - If I flip a coin once, and it's head, what's the estimate for P(head)?
  - What if I flip 10 times with 8 heads?
  - What if I flip 10M times with 8M heads?

- Basic idea:
  - We have some prior expectation about parameters (here, the probability of heads)
  - Given little evidence, we should skew towards our prior
  - Given a lot of evidence, we should listen to the data

Estimation: Laplace Smoothing

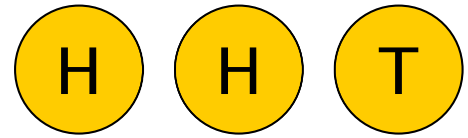
- Laplace's estimate (extended):
  - Pretend you saw every outcome *k* extra times

$$P_{LAP,k}(x) = \frac{c(x) + k}{N + k|X|}$$

  - What's Laplace with *k* = 0?
  - *k* is the strength of the prior

- Laplace for conditionals:
  - Smooth each condition independently:

$$P_{LAP,k}(x|y) = \frac{c(x,y) + k}{c(y) + k|X|}$$



$$P_{LAP,0}(X) =$$

$$P_{LAP,1}(X) =$$

$$P_{LAP,100}(X) =$$

Estimation: Linear Interpolation

- In practice, Laplace often performs poorly for P(X|Y):
  - When |X| is very large
  - When |Y| is very large

- Another option: linear interpolation
  - Also get P(X) from the data
  - Make sure the estimate of P(X|Y) isn't too different from P(X)

$$P_{LIN}(x|y) = \alpha \widehat{P}(x|y) + (1.0 - \alpha)\widehat{P}(x)$$

  - What if α is 0? 1?

Stochastic Language Models

- Model *probability* of generating strings (each word in turn) in a language (commonly all strings over alphabet ∑). E.g., a unigram model

Model M

0.2	the
0.1	a
0.01	man
0.01	woman
0.03	said
0.02	likes
…	

the	man	likes	the	woman
0.2	0.01	0.02	0.2	0.01

multiply

$P(s \mid M) = 0.00000008$

Stochastic Language Models

- Model *probability* of generating any string

Model M1

0.2	the
0.01	class
0.0001	sayst
0.0001	pleaseth
0.0001	yon
0.0005	maiden
0.01	woman

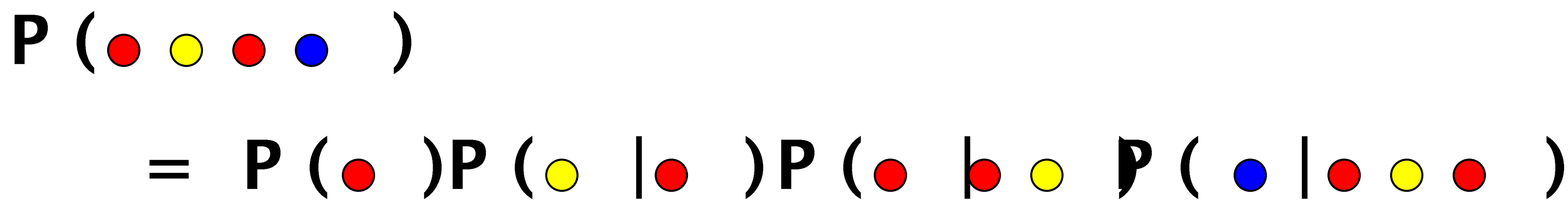
Model M2

0.2	the
0.0001	class
0.03	sayst
0.02	pleaseth
0.1	yon
0.01	maiden
0.0001	woman

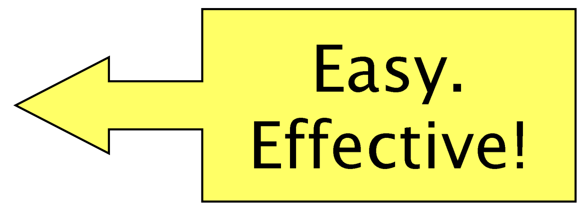
the	class	pleaseth	yon	maiden
0.2	0.01	0.0001	0.0001	0.0005
0.2	0.0001	0.02	0.1	0.01

$$P(s|M2) > P(s|M1)$$

Unigram and higher-order models



- Unigram Language Models



- Bigram (generally, *n*-gram) Language Models

- Other Language Models
  - Grammar-based models (PCFGs), etc.
    - Probably not the first thing to try in IR

Two Naive Bayes Models

- Model 1: Multivariate Bernoulli
  - One feature X_w for each word in dictionary
  - X_w = true in document d if w appears in d
  - Naive Bayes assumption:
    - Given the document's topic, appearance of one word in the document tells us nothing about chances that another word appears

Two Models

- Model 2: Multinomial = Class conditional unigram
  - One feature X_i for each word position in document
    - feature's values are all words in dictionary
  - Value of X_i is the word in position i
  - Naive Bayes' assumption:
    - Given the document's topic, word in one position in the document tells us nothing about words in other positions
  - Second assumption:
    - Word appearance does not depend on position

$$P(X_i = w \mid c) = P(X_j = w \mid c)$$

for all positions i,j, word w, and class c

- Just have one multinomial feature predicting all words

Parameter estimation

- Multivariate Bernoulli model:

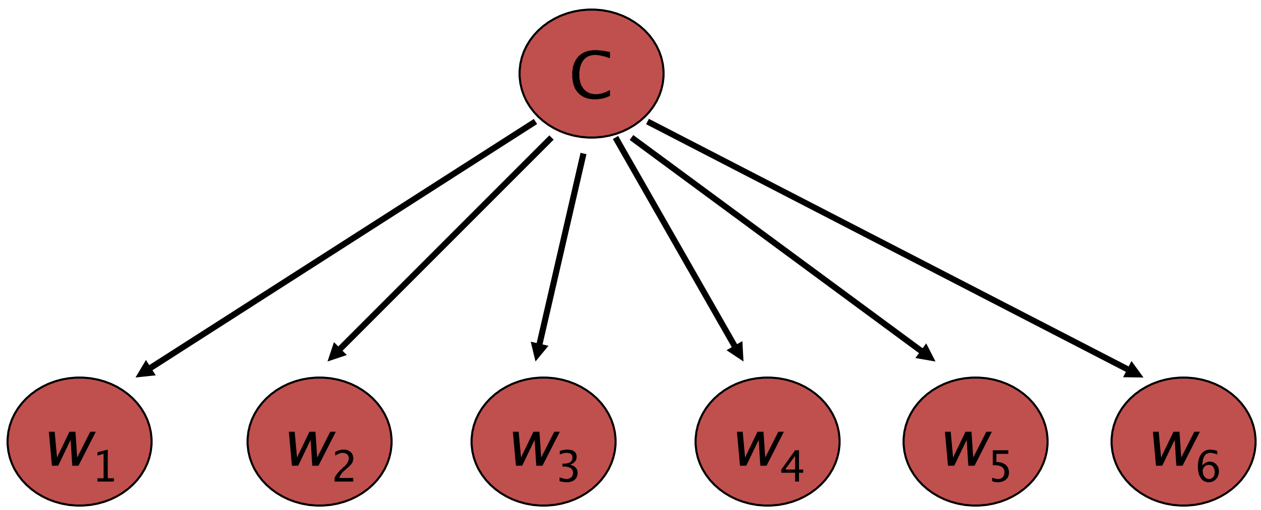
$$\hat{P}(X_w = t \mid c_j) = \text{fraction of documents of topic } c_j \text{ in which word } w \text{ appears}$$

- Multinomial model:

$$\hat{P}(X_i = w \mid c_j) = \text{fraction of times in which word } w \text{ appears among all words in documents of topic } c_j$$

  - Can create a mega-document for topic *j* by concatenating all documents in this topic
  - Use frequency of *w* in mega-document

Naive Bayes via a class conditional language model = multinomial NB



- Effectively, the probability of each class is done as a class-specific unigram language model

Using Multinomial Naive Bayes Classifiers to Classify Text: Basic method

- Attributes are text positions, values are words

$$c_{NB} = \underset{c_j \in C}{\operatorname{argmax}} P(c_j) \prod_i P(x_i \mid c_j)$$

$$= \underset{c_j \in C}{\operatorname{argmax}} P(c_j) P(x_1 = \text{"our"} \mid c_j) \cdots P(x_n = \text{"text"} \mid c_j)$$

- Still too many possibilities
- Assume that classification is *independent* of the positions of the words
  - Use same parameters for each position
  - Result is *bag of words* model (*over tokens and not types*)

Naive Bayes: Learning

- From training corpus, extract *Vocabulary*
- Calculate required $P(c_j)$ and $P(x_k \mid c_j)$ terms
  - For each c_j in C do
    - $docs_j \leftarrow$ subset of documents for which the target class is c_j

$$P(c_j) \leftarrow \frac{|docs_j|}{|\text{total \# documents}|}$$

- $Text_j \leftarrow$ single document containing all $docs_j$
- for each word x_k in *Vocabulary*
  - $n_k \leftarrow$ number of occurrences of x_k in $Text_j$

$$P(x_k \mid c_j) \leftarrow \frac{n_k + \alpha}{n + \alpha |Vocabulary|}$$

Naive Bayes: Classifying

- positions $\leftarrow$ all word positions in current document which contain tokens found in *Vocabulary*
- Return c_{NB}, where

$$c_{NB} = \operatorname*{argmax}_{c_j \in C} P(c_j) \prod_{i \in positions} P(x_i \mid c_j)$$

Naive Bayes: Time Complexity

- Training Time: $O(|D|L_d + |C||V|))$
  where L_d is the average length of a document in D
  - Assumes V and all D_i, n_i, and n_{ij} pre-computed in $O(|D|L_d)$ time during one pass through all of the data
  - Generally just $O(|D|L_d)$ since usually $|C||V| << |D|L_d$
    - |C| |V| = Complexity of computing all probability values (loop over terms and classes)

- Test Time: $O(|C|\ L_t)$
  where L_t is the average length of a test document
- Very efficient overall, linearly proportional to the time needed to just read in all the data

An Example

	docID	words in document	in $c = China$?
training set	1	Chinese Beijing Chinese	yes
	2	Chinese Chinese Shanghai	yes
	3	Chinese Macao	yes
	4	Tokyo Japan Chinese	no
test set	5	Chinese Chinese Chinese Tokyo Japan	?

- Estimate parameters of Naive Bayes classifier
- Classify the test document

Example: Parameter estimates

Priors: $\hat{P}(c) = 3/4$ and $\hat{P}(\overline{c}) = 1/4$ Conditional probabilities:

$$\begin{aligned}
\hat{P}(\textsc{Chinese}|c) &= (5+1)/(8+6) = 6/14 = 3/7 \\
\hat{P}(\textsc{Tokyo}|c) = \hat{P}(\textsc{Japan}|c) &= (0+1)/(8+6) = 1/14 \\
\hat{P}(\textsc{Chinese}|\overline{c}) &= (1+1)/(3+6) = 2/9 \\
\hat{P}(\textsc{Tokyo}|\overline{c}) = \hat{P}(\textsc{Japan}|\overline{c}) &= (1+1)/(3+6) = 2/9
\end{aligned}$$

The denominators are (8 + 6) and (3 + 6) because the lengths of $text_c$ and $text_{\overline{c}}$ are 8 and 3, respectively, and because the constant B is 6 as the vocabulary consists of six terms.

42

Example: Classification

$$\hat{P}(c|d_5) \quad \propto \quad 3/4 \cdot (3/7)^3 \cdot 1/14 \cdot 1/14 \approx 0.0003$$
$$\hat{P}(\overline{c}|d_5) \quad \propto \quad 1/4 \cdot (2/9)^3 \cdot 2/9 \cdot 2/9 \approx 0.0001$$

Thus, the classifier assigns the test document to *c* = *China*. The reason for this classification decision is that the three occurrences of the positive indicator `CHINESE` in d_5 outweigh the occurrences of the two negative indicators `JAPAN` and `TOKYO`

43

Underflow Prevention: using logs

- Multiplying lots of probabilities, which are between 0 and 1 by definition, can result in floating-point underflow
- Since log(*xy*) = log(*x*) + log(*y*), it is better to perform all computations by summing logs of probabilities rather than multiplying probabilities
- Class with highest final un-normalized log probability score is still the most probable

$$c_{NB} = \underset{c_j \in C}{\operatorname{argmax}} [\log P(c_j) + \sum_{i \in positions} \log P(x_i \mid c_j)]$$

- Note that model is now just max of sum of weights…

Naive Bayes Classifier

$$c_{NB} = \underset{c_j \in C}{\mathrm{argmax}}[\log P(c_j) + \sum_{i \in positions} \log P(x_i \mid c_j)]$$

- The prior log $P(c_j)$ is a weight that indicates the relative frequency of c_j
- Simple interpretation: Each conditional parameter log $P(x_i|c_j)$ is a weight that indicates how good an indicator x_i is for c_j
- The sum is then a measure of how much evidence there is for the document being in the class
- We select the class with the most evidence for it

Multi-variate NB Model

TRAINBERNOULINB (C, D)

1. V←ExtractVocabulary (D)
2. N←CountDocs (D)
3. for each c in C
4. do Nc← CountDocsInClass (D, c)
5. Prior [c]←Nc/N
6. for each t in V
7. Nct←CountDocsInClassContainingTerm (D,c,t)
8. condprob [t][c]←(Nct+1)/(Nc+2)
9. return V, prior, condprob

Multi-variate NB Model

ApplyBernoullliNB (C,V,prior,condprob,d)

1. Vd←ExtractTermsFromDoc(V,d)
2. for each c in C
3. do score [c]←log prior [c]
4. for each t in v
5. do if t in vd
6. then score[c]+= log condprob[t][c]
7. else score [c]+=log (1-condprob[t][c])
8. return argmax score[c]

Classification

- Multinomial vs Multivariate Bernoulli?

- Multinomial model is almost always more effective in text applications!
- While classifying a test document
  - Bernoulli model uses binary occurrence information, ignoring the number of occurrences
  - Multinomial model keeps track of multiple occurrences
  - Bernoulli makes many mistakes while classifying long documents (as it ignores counts)

Naive Bayes is Not So Naive

- Naive Bayes won 1st and 2nd place in KDD-CUP 97 competition out of 16 systems

  Goal: Financial services industry direct mail response prediction model: Predict if the recipient of mail will actually respond to the advertisement – 750,000 records.

- More robust to irrelevant features than many learning methods

  Irrelevant Features cancel each other without affecting results

  Decision Trees can suffer heavily from this.

- More robust to concept drift (changing class definition over time)
- Very good in domains with many equally important features

  Decision Trees suffer from *fragmentation* in such cases – especially if little data

- A good dependable baseline for text classification (but not the best)!
- Optimal if the Independence Assumptions hold: Bayes Optimal Classifier

  Never true for text, but possible in some domains

- Very Fast Learning and Testing (basically just count the data)
- Low Storage requirements

Resources for lecture

- IIR 13
- Fabrizio Sebastiani. Machine Learning in Automated Text Categorization. *ACM Computing Surveys*, 34(1):1-47, 2002.
- Yiming Yang & Xin Liu, A re-examination of text categorization methods. *Proceedings of SIGIR*, 1999.
- Andrew McCallum and Kamal Nigam. A Comparison of Event Models for Naive Bayes Text Classification. In *AAAI/ICML-98 Workshop on Learning for Text Categorization*, pp. 41-48.
- Tom Mitchell, Machine Learning. McGraw-Hill, 1997.
  - Clear simple explanation of Naive Bayes
- Open Calais: Automatic Semantic Tagging
  - Free (but they can keep your data), provided by Thompson/Reuters (ex-ClearForest)
- Weka: A data mining software package that includes an implementation of Naive Bayes
- Reuters-21578 – the most famous text classification evaluation set
  - Still widely used by lazy people (but now it's too small for realistic experiments – you should use Reuters RCV1)