

# CS 188: Artificial Intelligence

## Fall 2006

Lecture 22: Naïve Bayes  
11/14/2006

Dan Klein – UC Berkeley

## Announcements

---

- **Optional midterm**
  - On Tuesday 11/21 in class
  - Review session 11/19, 7 pm, in 306 Soda
- **Projects**
  - 3.3 due 11/15
  - 3.4 due 11/27
- **Contest details on web!**

## Machine Learning

---

- Up till now: how to reason or make decisions using a model
- **Machine learning: how to select 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)

## Classification

---

- In classification, we learn to predict labels (classes) for inputs
- **Examples:**
  - Spam detection (input: document, classes: spam / ham)
  - OCR (input: images, classes: characters)
  - Medical diagnosis (input: symptoms, classes: diseases)
  - Automatic essay grader (input: document, classes: grades)
  - Fraud detection (input: account activity, classes: fraud / no fraud)
  - Customer service email routing
  - ... many more
- **Classification is an important commercial technology!**

## Classification

---

- **Data:**
  - Inputs  $x$ , class labels  $y$
  - We imagine that  $x$  is something that has a lot of structure, like an image or document
  - In the basic case,  $y$  is a simple N-way choice
- **Basic Setup:**
  - Training data:  $D =$  bunch of  $\langle x, y \rangle$  pairs
  - Feature extractors: functions  $f_i$  which provide attributes of an example  $x$
  - Test data: more  $x$ 's, we must predict  $y$ 's
    - During development, we actually know the  $y$ 's, so we can check how well we're doing, but when we deploy the system, we don't

## Bayes Nets for Classification

---

- **One method of classification:**
  - Features are values for observed variables
  - $Y$  is a query variable
  - Use probabilistic inference to compute most likely  $Y$

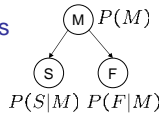
$$y = \operatorname{argmax}_y P(y|f_1 \dots f_n)$$

- **You already know how to do this inference**

## Simple Classification

- Simple example: two binary features

- This is a naïve Bayes model



$P(m|s, f)$  ← direct estimate

$P(m|s, f) = \frac{P(s, f|m)P(m)}{P(s, f)}$  ← Bayes estimate (no assumptions)

$P(m|s, f) = \frac{P(s|m)P(f|m)P(m)}{P(s, f)}$  ← Conditional independence

+  $\begin{cases} P(m, s, f) = P(s|m)P(f|m)P(m) \\ P(\bar{m}, s, f) = P(s|\bar{m})P(f|\bar{m})P(\bar{m}) \end{cases}$

## General Naïve Bayes

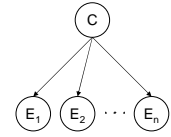
- A general naïve Bayes model:

$|C| \times |E|^n$  parameters

$P(\text{Cause}, \text{Effect}_1 \dots \text{Effect}_n) =$

$P(\text{Cause}) \prod_i P(\text{Effect}_i | \text{Cause})$

$|C|$  parameters       $n \times |E| \times |C|$  parameters



- We only specify how each feature depends on the class
- Total number of parameters is **linear** in n

## Inference for Naïve Bayes

- Goal: compute posterior over causes

- Step 1: get joint probability of causes and evidence

$P(C, e_1 \dots e_n) =$

$\begin{bmatrix} P(c_1, e_1 \dots e_n) \\ P(c_2, e_1 \dots e_n) \\ \vdots \\ P(c_k, e_1 \dots e_n) \end{bmatrix} \Rightarrow \begin{bmatrix} P(c_1) \prod_i P(e_i|c_1) \\ P(c_2) \prod_i P(e_i|c_2) \\ \vdots \\ P(c_k) \prod_i P(e_i|c_k) \end{bmatrix}$

- Step 2: get probability of evidence

$P(e_1 \dots e_n)$

- Step 3: renormalize

$P(C|e_1 \dots e_n)$

## General Naïve Bayes

- What do we need in order to use naïve Bayes?

- Some code to do the inference (you know this part)

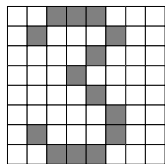
- For fixed evidence, build  $P(C, e)$
- Sum out C to get  $P(e)$
- Divide to get  $P(C|e)$

- Estimates of local conditional probability tables

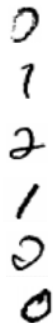
- $P(C)$ , the prior over causes
- $P(E|C)$  for each evidence variable
- These probabilities are collectively called the **parameters** of the model and denoted by  $\theta$
- These typically come from observed data: we'll look at this now

## A Digit Recognizer

- Input: pixel grids



- Output: a digit 0-9



## Naïve Bayes for Digits

- Simple version:

- One feature  $F_{ij}$  for each grid position  $\langle i, j \rangle$
- Feature values are on / off based on whether intensity is more or less than 0.5
- Input looks like:

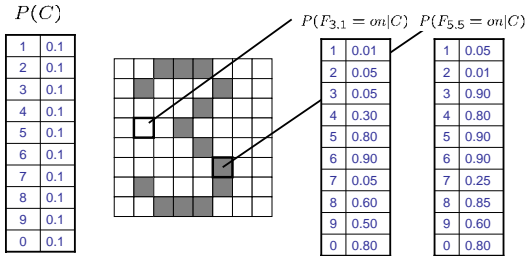
$1 \rightarrow \langle F_{0,0} = 0 \ F_{0,1} = 0 \ F_{0,2} = 1 \ F_{0,3} = 1 \ F_{0,4} = 0 \ \dots \ F_{15,15} = 0 \rangle$

- Naïve Bayes model:

$P(C, F_{0,0} \dots F_{15,15}) = P(C) \prod_{i,j} P(F_{i,j}|C)$

- What do we need to learn?

## Examples: CPTs



## Parameter Estimation

- Estimating the distribution of a random variable X or X|Y
- Empirically:** use training data
  - For each value x, look at the **empirical rate** of that value:

$$\hat{P}(x) = \frac{\text{count}(x)}{\text{total samples}}$$

r g g  
 $\hat{P}(r) = 1/3$

- This estimate maximizes the **likelihood of the data**

$$L(x, \theta) = \prod_i P_{\theta}(x_i)$$

- Elicitation:** ask a human!
  - Usually need domain experts, and sophisticated ways of eliciting probabilities (e.g. betting games)
  - Trouble calibrating

## A Spam Filter

- Naïve Bayes spam filter

### Data:

- Collection of emails, labeled spam or ham
- Note: someone has to hand label all this data!
- Split into training, held-out, test sets

### Classifiers

- Learn on the training set
- (Tune it on a held-out set)
- Test it on new emails

Dear Sir.

First, I must solicit your confidence in this transaction, this is by virtue of its nature as being utterly confidential 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, I know 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.

## Naïve Bayes for Text

- Naïve Bayes:
  - Predict unknown cause (spam vs. ham)
  - Independent evidence from observed variables (e.g. the words)

- Generative model\*

$$P(C, W_1 \dots W_n) = P(C) \prod_i P(W_i|C)$$

*Word at position i, not i<sup>th</sup> word in the dictionary*

- Tied distributions and bag-of-words
  - Usually, each variable gets its own conditional probability distribution
  - In a bag-of-words model
    - Each position is identically distributed
    - All share the same distributions
    - Why make this assumption?

\*Minor detail: technically we're conditioning on the length of the document here

## Example: Spam Filtering

- Model:  $P(C, W_1 \dots W_n) = P(C) \prod_i P(W_i|C)$
- What are the parameters?

$P(C)$	$P(W \text{spam})$	$P(W \text{ham})$
ham : 0.66 spam: 0.33	the : 0.0156 to : 0.0153 and : 0.0115 of : 0.0095 you : 0.0093 a : 0.0086 with: 0.0080 from: 0.0075 ...	the : 0.0210 to : 0.0133 of : 0.0119 2002: 0.0110 with: 0.0108 from: 0.0107 and : 0.0105 a : 0.0100 ...

- Where do these tables come from?

## Spam Example




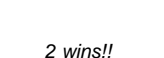
Word	P(w spam)	P(w ham)	Tot Spam	Tot Ham
(prior)	0.33333	0.66666	-1.1	-0.4

P(spam | w) = 98.9

## Example: Overfitting

$P(\text{features}, C = 2)$

$P(\text{features}, C = 3)$

$P(\text{on} C = 2) = 0.8$		$P(\text{on} C = 3) = 0.8$
$P(\text{on} C = 2) = 0.1$		$P(\text{on} C = 3) = 0.9$
$P(\text{off} C = 2) = 0.1$		$P(\text{off} C = 3) = 0.7$
$P(\text{on} C = 2) = 0.01$		$P(\text{on} C = 3) = 0.0$

2 wins!!

## Example: Spam Filtering

- Raw probabilities don't affect the posteriors; relative probabilities (odds ratios) do:

$\frac{P(W \text{ham})}{P(W \text{spam})}$	$\frac{P(W \text{spam})}{P(W \text{ham})}$																												
<table border="1"> <tr><td>south-west</td><td>: inf</td></tr> <tr><td>nation</td><td>: inf</td></tr> <tr><td>morally</td><td>: inf</td></tr> <tr><td>nicely</td><td>: inf</td></tr> <tr><td>extent</td><td>: inf</td></tr> <tr><td>seriously</td><td>: inf</td></tr> <tr><td>...</td><td></td></tr> </table>	south-west	: inf	nation	: inf	morally	: inf	nicely	: inf	extent	: inf	seriously	: inf	...		<table border="1"> <tr><td>screens</td><td>: inf</td></tr> <tr><td>minute</td><td>: inf</td></tr> <tr><td>guaranteed</td><td>: inf</td></tr> <tr><td>\$205.00</td><td>: inf</td></tr> <tr><td>delivery</td><td>: inf</td></tr> <tr><td>signature</td><td>: inf</td></tr> <tr><td>...</td><td></td></tr> </table>	screens	: inf	minute	: inf	guaranteed	: inf	\$205.00	: inf	delivery	: inf	signature	: inf	...	
south-west	: inf																												
nation	: inf																												
morally	: inf																												
nicely	: inf																												
extent	: inf																												
seriously	: inf																												
...																													
screens	: inf																												
minute	: inf																												
guaranteed	: inf																												
\$205.00	: inf																												
delivery	: inf																												
signature	: inf																												
...																													

What went wrong here?

## Generalization and Overfitting

- Relative frequency parameters will overfit the training data!
  - 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?
  - In general, we can't go around giving unseen events zero probability
- As an extreme case, 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

## Estimation: Smoothing

- Problems with maximum likelihood estimates:
  - If I flip a coin once, and it's heads, what's the estimate for P(heads)?
  - 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: Smoothing

- Relative frequencies are the maximum likelihood estimates

$$\theta_{ML} = \arg \max_{\theta} P(\mathbf{X}|\theta) \Rightarrow \hat{P}(x) = \frac{\text{count}(x)}{\text{total samples}}$$

$$= \arg \max_{\theta} \prod_i P_{\theta}(X_i)$$

- In Bayesian statistics, we think of the parameters as just another random variable, with its own distribution

$$\theta_{MAP} = \arg \max_{\theta} P(\theta|\mathbf{X})$$

$$= \arg \max_{\theta} P(\mathbf{X}|\theta)P(\theta)/P(\mathbf{X}) \Rightarrow \text{???}$$

$$= \arg \max_{\theta} P(\mathbf{X}|\theta)P(\theta)$$

## Estimation: Laplace Smoothing

- Laplace's estimate:
  - Pretend you saw every outcome once more than you actually did



$$P_{LAP}(x) = \frac{c(x) + 1}{\sum_x [c(x) + 1]}$$

$$= \frac{c(x) + 1}{N + |X|}$$

$$P_{ML}(X) =$$

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

- Can derive this as a MAP estimate with Dirichlet priors (see cs281a)

## 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  $P(X)$

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

- What if  $\alpha$  is 0? 1?

- For even better ways to estimate parameters, as well as details of the math see cs281a, cs294-7

## Real NB: Smoothing

- For real classification problems, smoothing is critical

- New odds ratios:

$$\frac{P(W|\text{ham})}{P(W|\text{spam})}$$

helvetica	: 11.4
seems	: 10.8
group	: 10.2
ago	: 8.4
areas	: 8.3
...	

$$\frac{P(W|\text{spam})}{P(W|\text{ham})}$$

verdana	: 28.8
Credit	: 28.4
ORDER	: 27.2
<FONT>	: 26.9
money	: 26.5
...	

Do these make more sense?

## Classification

- Data: labeled instances, e.g. emails marked spam/ham

- Training set
- Held out set
- Test set

- Experimentation

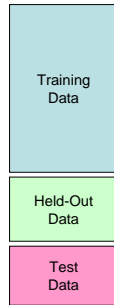
- Learn model parameters (probabilities) on training set
- (Tune performance on held-out set)
- Run a single test on the test set
- Very important: never "peek" at the test set!

- 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
- We'll investigate overfitting and generalization formally in a few lectures



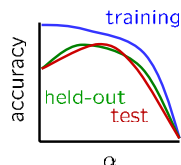
## Tuning on Held-Out Data

- Now we've got two kinds of unknowns

- Parameters: the probabilities  $P(Y|X)$ ,  $P(Y)$
- Hyper-parameters, like the amount of smoothing to do:  $k$ ,  $\alpha$

- Where to learn?

- Learn parameters from training data
- Must tune hyper-parameters on different data
  - Why?
- For each value of the hyper-parameters, train and test on the held-out data
- Choose the best value and do a final test on the test data



## Baselines

- First task: get a **baseline**

- Baselines are very simple "straw man" procedures
- Help determine how hard the task is
- Help know what a "good" accuracy is

- Weak baseline: most frequent label classifier

- Gives all test instances whatever label was most common in the training set
- E.g. for spam filtering, might label everything as ham
- Accuracy might be very high if the problem is skewed

- For real research, usually use previous work as a (strong) baseline

## Confidences from a Classifier

- The **confidence** of a probabilistic classifier:

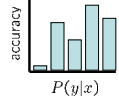
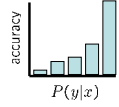
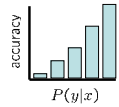
- Posterior over the top label

$$\text{confidence}(x) = \arg \max_y P(y|x)$$

- Represents how sure the classifier is of the classification
- Any probabilistic model will have confidences
- No guarantee confidence is correct

- Calibration**

- Weak calibration: higher confidences mean higher accuracy
- Strong calibration: confidence predicts accuracy rate
- What's the value of calibration?



## Errors, and What to Do

- Examples of errors

Dear GlobalSCAPE Customer,  
GlobalSCAPE has partnered with ScanSoft to offer you the latest version of OmniPage Pro, for just \$99.99\* - the regular list price is \$499! The most common question we've received about this offer is - Is this genuine? We would like to assure you that this offer is authorized by ScanSoft, is genuine and valid. You can get the . . .

. . . To receive your \$30 Amazon.com promotional certificate, click through to  
<http://www.amazon.com/apparel>  
and see the prominent link for the \$30 offer. All details are there. We hope you enjoyed receiving this message. However, if you'd rather not receive future e-mails announcing new store launches, please click . . .

## What to Do About Errors?

- Need more features- words aren't enough!

- Have you emailed the sender before?
- Have 1K other people just gotten the same email?
- Is the sending information consistent?
- Is the email in ALL CAPS?
- Do inline URLs point where they say they point?
- Does the email address you by (your) name?

- Can add these information sources as new variables in the NB model

- Next class we'll talk about classifiers which let you easily add arbitrary features more easily

## Summary

- Bayes rule lets us do diagnostic queries with causal probabilities
- The naïve Bayes assumption makes all effects independent given the cause
- We can build classifiers out of a naïve Bayes model using training data
- Smoothing estimates is important in real systems
- Classifier confidences are useful, when you can get them