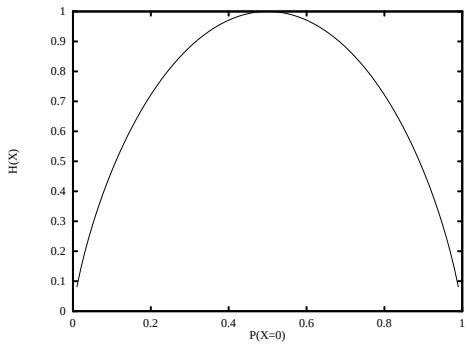
Entropy

- Let X be a discrete random variable
- The *surprise* of observing X = x is defined as
  - $-\log_2 P(X=x)$
- Surprise of probability 1 is zero.
- Surprise of probability 0 is ∞

Expected Surprise

- What is the expected surprise of X?
  - $\sum_x P(X=x) \cdot [-\log_2 P(X=x)]$
  - $\sum_x -P(X=x) \cdot \log_2 P(X=x)$
- This is known as the *entropy* of X: H(X)



Shannon's Experiment

- Measure the entropy of English
  - Ask humans to rank-order the next letter given all of the previous letters in a text.
  - Compute the position of the correct letter in this rank order
  - Produce a histogram
  - Estimate P(X| …) from this histogram
  - Compute the entropy H(X) = expected number of bits of "surprise" of seeing each new letter

Predicting the Next Letter

Everything_in_camp_was_drenched_the_camp_fire_as_well_for_they_were but heedless lads like their generation and had made no provision against rain Here was matter for dismay for they were soaked through and chilled They were eloquent in their distress but they presently discovered that the fire had eaten so far up under the great log it had been built against where it curved upward and separated itself from the ground that a hand breadth or so of it had escaped wetting so they patiently wrought until with shreds and bark gathered from the under sides of sheltered logs they coaxed the fire to burn again Then they piled on great dead boughs till they had a roaring furnace and were glad hearted once more They dried their boiled ham and had a feast and after that they sat by the fire and expanded and glorified their midnight adventure until morning for there was not a dry spot to sleep on anywhere around

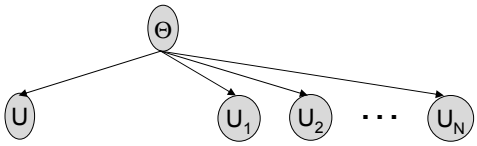
Statistical Learning Methods

- The Density Estimation Problem:
  - Given:
    - a set of random variables $U = \{V_1, \ldots, V_n\}$
    - A set S of training examples $\{U_1, \ldots, U_N\}$ drawn independently according to unknown distribution $\underline{P}(U)$
  - Find:
    - A bayesian network with probabilities Θ that is a good approximation to $\underline{P}(U)$
- Four Cases:
  - Known Structure; Fully Observable
  - Known Structure; Partially Observable
  - Unknown Structure; Fully Observable
  - Unknown Structure; Partially Observable

Bayesian Learning Theory

- Fundamental Question: Given S how to choose Θ?
- Bayesian Answer: Don't choose a single Θ.

A Bayesian Network for Learning Bayesian Networks



$P(U|U_1,\ldots,U_N) = P(U|S) = P(U \wedge S) / P(S)$

$= [\sum_\Theta P(U|\Theta) \prod_i P(U_i|\Theta) \cdot P(\Theta)] / P(S)$

$P(U|S) = \sum_\Theta P(U|\Theta) \cdot [P(S|\Theta) \cdot P(\Theta) / P(S)]$

$P(U|S) = \sum_\Theta P(U|\Theta) \cdot P(\Theta \mid S)$

Each Θ votes for U according to its posterior probability.

"Bayesian Model Averaging"

Approximating Bayesian Model Averaging

- Summing over all possible Θ's is usually impossible.
- Approximate this sum by the single most likely Θ value, Θ_{MAP}
- $\Theta_{MAP} = \text{argmax}_\Theta P(\Theta|S)$
  $= \text{argmax}_\Theta P(S|\Theta) P(\Theta)$
- $P(U|S) \approx P(U|\Theta_{MAP})$
- "Maximum Aposteriori Probability" – MAP

Maximum Likelihood Approximation

- If we assume $P(\Theta)$ is a constant for all Θ, then MAP become MLE, the Maximum Likelihood Estimate

  $\Theta_{MLE} = \text{argmax}_\Theta P(S|\Theta)$
- $P(S|\Theta)$ is called the "likelihood function"
- We often take logarithms

  $\Theta_{MLE} = \text{argmax}_\Theta P(S|\Theta)$

  $= \text{argmax}_\Theta \log P(S|\Theta)$

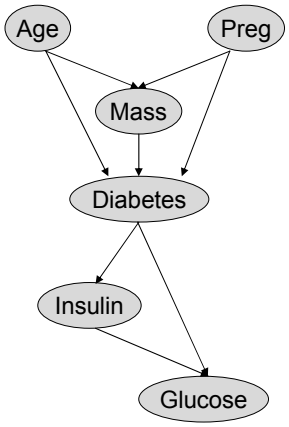
  $= \text{argmax}_\Theta \log \prod_i P(U_i|\Theta)$

  $= \text{argmax}_\Theta \sum_i \log P(U_i|\Theta)$

Experimental Methodology

- Collect data
- Divide data randomly into training and testing sets
- Choose Θ to maximize log likelihood of the training data
- Evaluate log likelihood on the test data

Known Structure, Fully Observable



Preg	Glucose	Insulin	Mass	Age	Diabetes?
3	3	1	2	3	0
3	2	3	3	6	0
3	4	0	3	3	0
3	1	1	3	4	0
3	5	0	3	4	1
1	1	0	3	4	1
2	0	3	3	2	0
2	8	2	4	2	1
1	2	0	3	4	0
3	2	1	3	4	0

Learning Process

- Simply count the cases:

$$P(Age = 2) = \frac{N(Age = 2)}{N}$$

$$P(Mass = 0 | Preg = 1, Age = 2) = \frac{N(Mass = 0, Preg = 1, Age = 2)}{N(Preg = 1, Age = 2)}$$

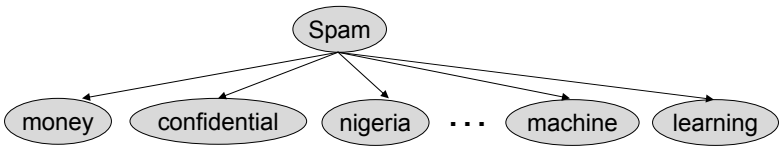
Laplace Corrections

- Probabilities of 0 and 1 are undesirable because they are too strong. To avoid them, we can apply the Laplace Correction. Suppose there are *k* possible values for age:

$$P(Age = 2) = \frac{N(Age = 2) + 1}{N + k}$$

- Implementation: Initialize all counts to 1. When the counts are normalized, this automatically computes *k*.

Spam Filtering using Naïve Bayes



- Spam $\in$ {0,1}
- One random variable for each possible word that could appear in email
- P(money=1 | Spam=1); P(money=1 | Spam=0)

Probabilistic Reasoning

- All of the variables are observed except Spam, so the reasoning is very simple:

  $P(spam=1|w_1,w_2,\ldots,w_n) = \alpha\, P(w_1|spam=1) \cdot P(w_2|spam=1) \cdot \cdots \cdot P(w_n|spam=1) \cdot P(spam=1)$

Likelihood Ratio

- To avoid normalization, we can compute the "log odds":

$$\frac{P(spam=1|w_1,\ldots,w_n)}{P(spam=0|w=1,\ldots,w_n)} = \frac{\alpha P(w_1|spam=1)\cdots P(w_n|spam=1)\cdot P(spam=1)}{\alpha P(w_1|spam=0)\cdots P(w_n|spam=0)\cdot P(spam=0)}$$

$$\frac{P(spam=1|w_1,\ldots,w_n)}{P(spam=0|w=1,\ldots,w_n)} = \frac{\alpha}{\alpha}\frac{P(w_1|spam=1)}{P(w_1|spam=0)}\cdots\frac{P(w_n|spam=1)}{P(w_n|spam=0)}\frac{P(spam=1)}{P(spam=0)}$$

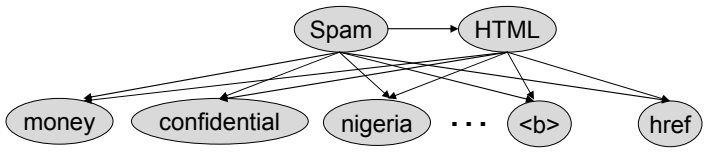
$$\log\frac{P(spam=1|w_1,\ldots,w_n)}{P(spam=0|w=1,\ldots,w_n)} = \log\frac{P(w_1|spam=1)}{P(w_1|spam=0)}+\ldots+\log\frac{P(w_n|spam=1)}{P(w_n|spam=0)}+\log\frac{P(spam=1)}{P(spam=0)}$$

Design Issues

- What to consider “words”?
  - Read Paul Graham’s articles (see web page)
  - Read about CRM114
  - Do we define w_j to be the number of times w_j appears in the email? Or do we just use a boolean: presence/absence of the word?
- How to handle previously unseen words?
  - Laplace estimates will assign them probabilities of 0.5 and 0.5 and NB will therefore ignore them.
- Efficient implementation
  - Two hash tables: one for spam and one for non-spam that contain the counts of the number of messages in which the word was seen.

Correlations?

- Naïve Bayes assumes that each word is generated independently given the class.
- HTML tokens are not generated independently. Should we model this?



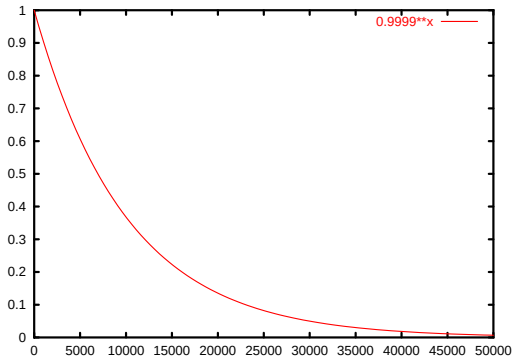
Dynamics

- We are engaged in an "arms race" between the spammers and the spam filters. Spam is changing all the time, so we need our estimates $P(w_i|spam)$ to be changing too.
- One Method: Exponential moving average. Each time we process a new training message, we decay the previous counts slightly. For every w_i:
  - $N(w_i|spam=1) := N(w_i|spam=1) \cdot 0.9999$
  - $N(w_i|spam=0) := N(w_i|spam=0) \cdot 0.9999$

  Then add in the counts for the new words.

  Choose the constant (0.9999) carefully.

Decay Parameter



- "half life" is 6930 updates (how did I compute that?)

Architecture

- .procmailrc is read by sendmail on engr accounts. This allows you to pipe your email into a program you write yourself.

```
# .procmail recipe
# pipe mail into myprogram, then continue processing it
:0fw: .msgid.lock
| /home/tgd/myprogram
# if myprogram added the spam header, then file into
# the spam mail file
:0: * ^X-SPAM-Status: SPAM.*
mail/spam
```


Architecture (2)

- Tokenize
- Hash
- Classify

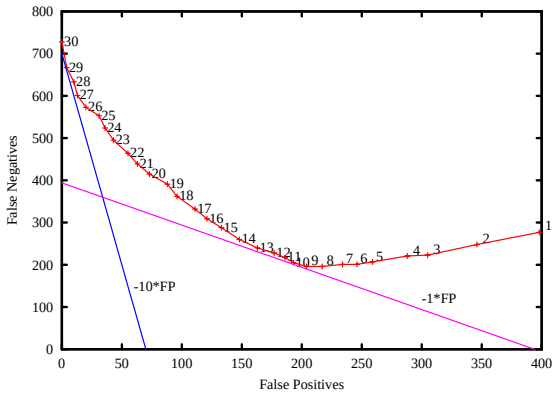
Classification Decision

- False positives: good email misclassified as spam
- False negatives: spam misclassified as good email
- Choose a threshold θ

$$\log \frac{P(spam = 1|w_1, \ldots, w_n)}{P(spam = 0|w = 1, \ldots, w_n)} > \theta$$

Plot of False Positives versus False Negatives

As we vary θ, we change the number of false positives and false negatives. We can choose the threshold that achieves the desired ratio of FP to FN



Methodology

- Collect data (spam and non-spam)
- Divide data into training and testing (presumably by choosing a cutoff date)
- Train on the training data
- Test on the testing data
- Compute Confusion Matrix:

Predicted Class	True Class	
	spam	nonspam
spam	TP	FP
nonspam	FN	TN

Choosing θ by internal validation

- Subdivide Training Data into “subtraining” set and “validation” set
- Train on subtraining set
- Classify validation set and record the predicted log odds of spam for each validation example
- Sort and construct FP/FN graph
- Choose θ
- Now retrain on entire training set