

# The 10 Commandments for Java Programmers

VII: Thou shalt study thy libraries with care and strive not to reinvent them without cause, that thy code may be short and readable and thy days pleasant and productive.

Loosely adapted from "10 Commandments for C Programmers", by Henry Spencer

# Bayesian Decision Making 101

- Deciding between 2 classes,  $C_N$  and  $C_S$
- Based on some data  $\mathbf{X}$  (email message)
- Decision criterion is probability test:  
$$\Pr[C_S|\mathbf{X}] \text{ <?> } \Pr[C_N|\mathbf{X}]$$
- $>$   $\square$  SPAM;  $<$   $\square$  NORMAL
- $\Pr[C_S|\mathbf{X}]$  and  $\Pr[C_N|\mathbf{X}]$  are *posterior probabilities* (after the fact) -- I.e., after you see the data  $\mathbf{X}$

# Email Probabilities

- So how do you get  $\Pr[C_i|\mathbf{X}]$  ?
- Need a TRAINING phase:
  - Given LABELED data (known SPAM/NORMAL)
  - Measure statistics for each class
- Later, in CLASSIFICATION phase, label UNLABELED (unknown) messages
  - Compare stats of UNLABELED email msg to stats of TRAINING data

# Finding the Posteriors

- Problem: LABELED email only tells you a little bit -- namely  $\Pr[C_i|\mathbf{X}] = 1$  or  $0$  for a specific  $\mathbf{X}$ .
- But through the magic of Bayes' rule, we can turn the question around:

$$\Pr[C_i|\mathbf{X}] = \Pr[\mathbf{X} | C_i] * \Pr[C_i] / \Pr[\mathbf{X}]$$

- This is called the "Posterior probability" of  $C_i$  (after the data)

# The Parts of the Posterior

- Now we have 3 parts. Is life better?
- Look at them one at a time:
- $\Pr[\mathbf{X}]$ 
  - This is the “data prior”
  - Represents the probability of the data “independent of SPAM or NORMAL”.
  - I.e., averaged over SPAM/NORMAL
- This one is irrelevant to us. Why?

# Irrelevance of the Data Prior

- Recall our classification rule:

$$\Pr[C_S|\mathbf{X}] \langle? \rangle \Pr[C_N|\mathbf{X}]$$

- Now substitute in from the Bayes' rule expression:

$$\Pr[\mathbf{X}|C_S]\Pr[C_S]/\Pr[\mathbf{X}] \langle? \rangle$$

$$\Pr[\mathbf{X}|C_N]\Pr[C_N]/\Pr[\mathbf{X}]$$

- Look!  $\Pr[\mathbf{X}]$  cancels. How fortunate...

# The Class Prior

- Ok, now what about  $\Pr[C_S]$  and  $\Pr[C_N]$ ?
- These are called CLASS PRIORS, or sometimes just PRIORS (from “before the data” -- before you’ve looked at a specific **X**)
- These are easy -- just the probability of any message being SPAM or NORMAL
- E.g.:

$$\Pr[C_S] = \frac{(\# \text{SPAM emails})}{(\# \text{all emails})}$$

## 2 Down, 1 to Go...

- That leaves  $\Pr[\mathbf{X} | C_i]$
- This is called the *generative model*
- It basically says:
  - Think of two machines -- a SPAM machine and a NORMAL machine
  - If you turn on the SPAM machine, it spits out an email message.
  - Given that you turned on the SPAM machine, what's the chance you saw this particular  $\mathbf{X}$ ?

# Measuring $\Pr[\mathbf{X}|C_i]$

- To measure  $\Pr[\mathbf{X}|C_i]$ :
  - Split the training data into SPAM/NORMAL
  - For each class, count the number of email messages of type  $\mathbf{X}$
  - Then  $\Pr[\mathbf{X}|C_i] = (\#\mathbf{X} \text{ in Class } i) / (\#\text{all msgs in } i)$
- Problem: each message is unique
- No big deal if you have *infinite* data
- Sadly, my hard drive isn't that big (yet)

## *Approximating* $\Pr[\mathbf{X} | C_i]$

- Massive, ruthless approximation time...
- Break up  $\mathbf{X}$  into a bunch of small pieces called features:  $\mathbf{X} = [x_1, x_2, \dots, x_k]$
- Features might be:
  - Words in an email
  - N-grams (sequences of N characters)
  - Line lengths
  - Etc.
- Now write:  $\Pr[\mathbf{X} | C_i] = \prod_{j=1}^k \Pr[x_j | C_i]$
- This is the “naïve Bayes” approximation

# Measuring Feature Probs

- Now you need  $\Pr[x_j | C_i]$
- But that's easy to measure: frequency of a given feature (e.g., a word) in the TRAINING data
- E.g., suppose features are "every word you might see".
- Then,

$$\Pr["viagra" | C_S] = \frac{(\# \text{ "viagra" in SPAM})}{(\# \text{ all words in SPAM})}$$

# Putting it All Together

- Now you have all the parts you need
- When you see an unlabeled email message:
  - Break msg down into features  $\mathbf{X} = [x_1, x_2, \dots, x_k]$
  - For each class (SPAM or NORMAL)
  - Calculate the posterior for that class:  
$$\Pr[C_S | \mathbf{X}] = \Pr[C_S] \prod_{j=1}^k \Pr[x_j | C_S]$$
$$\Pr[C_N | \mathbf{X}] = \Pr[C_N] \prod_{j=1}^k \Pr[x_j | C_N]$$
  - Label the message according to the max posterior

# Practical Issue #1

- Underflow
- Look back at the generative model equation again:

$$\Pr[C_S|\mathbf{X}] = \Pr[C_S] \prod_{j=1}^k \Pr[x_j|C_S]$$

- What happens when, say  $k=500$ ? (E.g., your message has 500 different words)
- Product of 500 factors, each between 0.0 and 1.0...
- The variables that you're storing  $\Pr[C_i|\mathbf{X}]$  in (e.g., `doubles`) quickly go to 0...