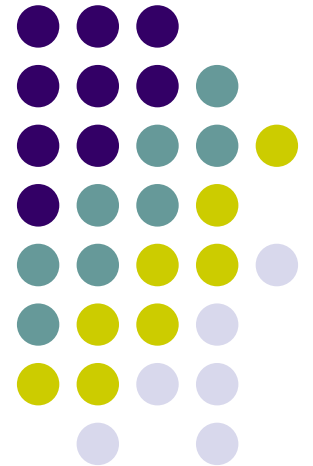
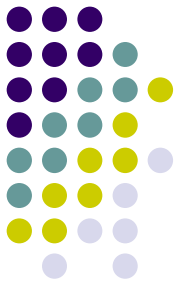


# Naïve Bayes Classification

By Dr. Zhang Hongxin  
State Key Lab of CAD&CG, ZJU  
2005-06-16



# Outline



- Supervised learning:
  - classification & regression
- Bayesian Spam Filtering: an example
- Naïve Bayes classification



# Supervised learning

- **Given:** training examples (*instances*) for which *attribute* values and target are known
  - E.g. age, weight, etc., for a patient and whether that patient has diabetes or not
  - **Instance:**  $\langle \mathbf{x}_i, t_i \rangle$
- **Goal:** function that predicts target variable accurately given new attribute values
  - **Learn:** mapping from  $\mathbf{x}$  to  $t(\mathbf{x})$ .
- Accuracy is measured on *fresh* data

# Regression vs. Classification



- Regression: predicting a *numeric* target variable given some attributes
  - E.g. predicting auto price given make, horse power, weight, etc.
- Classification: predicting a *nominal* target variable given some attributes
  - E.g. predicting type of car based on job and wealth of owner, type of best friend's car, etc.



# Bayesian Spam Filtering: An example

- The materials of this part are mainly from
  - <http://www.cs.mun.ca/~donald/slug/2005-04-07/slugspam.pdf>
- Resources:
  - <http://www.cs.mun.ca/~donald/buryspam/>
    - extensive documentation on Donald's buryspam.rb script.
  - <http://www.paulgraham.com/spam/>:
    - Paul Graham's spam proposal.
  - <http://www.mozilla.org/>:
    - Free Mozilla Thunderbird e-mail client.
  - <http://spambayes.sourceforge.net/>:
    - Plugins for other e-mail clients.
  - <http://spamassassin.sourceforge.net/doc/sa-learn.html>:
    - SpamAssassin's Bayesian classifier

# You've Got Mail! 😊 or ☹️



From uunet.uu.net!national-alliance.org!Crusader Sat Sep 30 17:26:42 1995

Received: from asso.nis.garr.it (@asso.nis.garr.it [192.12.192.10]) by

garfield.cs.mun.ca with SMTP id <102189>; Sat, 30 Sep 1995 17:26:31 -0230

Received: by asso.nis.garr.it (4.1/1.34/ABB950929)

id AA18937; Sat, 30 Sep 95 19:03:23 +0100

Received: by mercury.sfsu.edu (5.0/SMI-SVR4)

id AA21676; Sat, 30 Sep 95 11:03:27 -0700

Date: Sat, 30 Sep 1995 15:33:27 -0230

From: Crusader@national-alliance.org

Message-Id: <913247217488@National-Alliance.org>

To: <donald@cs.mun.ca>

To: <XXXXXXX@cs.mun.ca>

Subject: Piranhas

Apparently-To: [Crusader@National-Alliance.org](mailto:Crusader@National-Alliance.org)

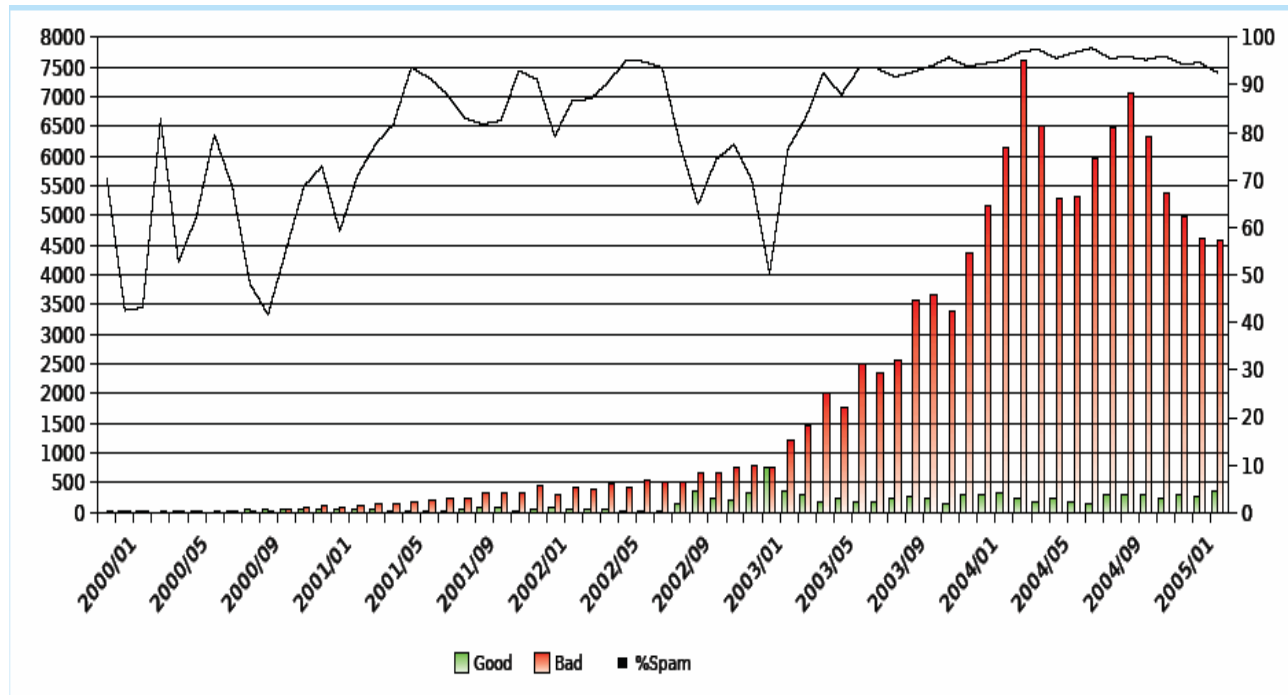
What would you say if a Liberal "social scientist" told you to jump into a pool filled with five hundred ravenous piranhas?

If you valued your life, you'd certainly refuse the invitation.

But what if the Liberal "social scientist" tried to convince you to go ahead and jump in, with the argument that "not all of the piranhas are aggressive. Some of them probably just want to make friends with you, and really aren't hungry either. To say that a

... *blah, blah, blah...*

# Spam, Spam, and Spam!



- The volume of spam was increasing exponentially.
  - Currently, about 95% of my e-mail is spam.
  - I currently receive about 100 to 150 spam messages per day, over 5MB a week (down from about 200 to 250 per day).

# Filtering Attempt: key words and regular expressions



- An early attempt to identify spam was to use *procmail*.
- This involved storing regular expressions in one's .procmailrc which would match phrases that occurred frequently in spam messages. e.g.

```
:0B:
* (mailto:|subject=(3D?)(abuse|nosale|(un)?sub|delete|remov(e|al)|exclude|discontinue|nomore|
(additional|more)[-_.]info)|annuler=|(reply|yes|subscribe|remove)["']? (yourself|instructions|
.*from .*mailing|.*(in|as).*subject)|To remove.*(e-?mail|message)|to.* be.*(r.?e.?m.?o.?v.?e.?d?|
taken (from|off)|extracted)|for removal|removal instructions|remove:mailto|to (be taken out|
get off|opt out|remove from|unsubscribe)|remove (requests|me)|(do not|don.t) respond|(global|
no need to) remove|remove[a-z0-9]*@|cancel subscription|remove(me|you|\. (php|html?))|opt.?out|
perman(ent|tently) remov(ed?|al)|remov(ed?|al) (notification|requests|administrator)|$removal:|
remove in the|if you wish to receive|(/|mail)-?(refuse|remove|unsu(b|s))|["']remove["']|off.?list|
reject.?mail
junk/junk.a
```

- Problems
  - Its binary nature could very easily generate false positives.
  - Spam kept changing too rapidly for this strategy to be effective.
  - As a result, the **.procmailrc** le became very large and cumbersome to change.

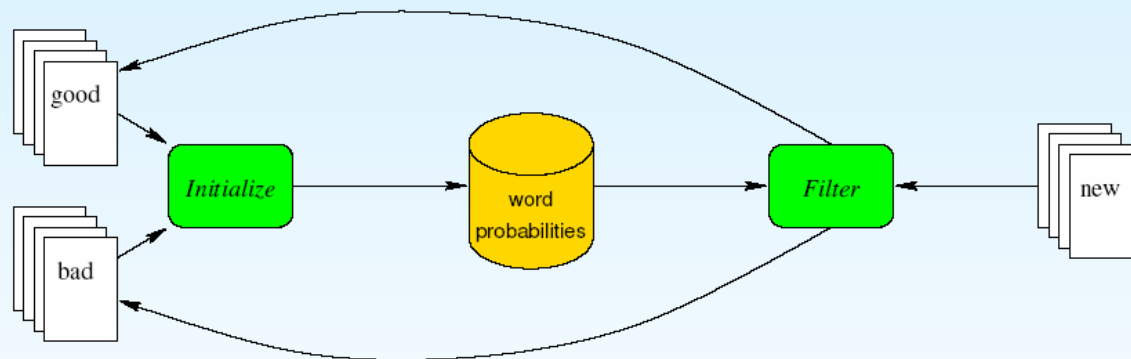




# A Better Solution

- In August 2002, Paul Graham advocated Bayesian filtering as an effective means of combating the spam problem. He didn't actually discover the technique, but he did help popularize it.
- Accuracy rate was claimed to be in excess of 99%.
- The technique has two distinct phases:

A collection of good and bad messages are examined and statistical probabilities for all words is derived based upon the relative frequencies of each word in the two collection of messages.



The filter uses these statistical probabilities during its analysis of incoming messages to classify them as spam or non-spam.



# The Initializing Phase

If we know:

- $msg_b$  = total number of bad messages,
- $msg_g$  = total number of good messages,
- $w_b$  = occurrences of word  $w$  in bad messages and
- $w_g$  = occurrences of word  $w$  in good messages,

then we can assign a probability to each word  $w$  based upon the relative frequencies that  $w$  occurred in all the bad and good messages.

$$P(w) = \max \left( \min \left( \frac{r_b}{r_g + r_b}, 0.99 \right), 0.01 \right)$$

where:

$$r_b = \min \left( \frac{w_b}{msg_b}, 1.0 \right), \quad r_g = \min \left( \frac{2w_g}{msg_g}, 1.0 \right)$$

The further away  $P(w)$  is from 0.5 (neutral), the more useful the word  $w$  is in determining whether or not the message is spam.



# Initializing Examples

$$msg_b = 69,449 \quad msg_g = 9,580$$

$w = \text{buy} :$

$$w_b = 4,434 \quad w_g = 171$$

$$\begin{aligned} r_b &= \frac{4,434}{69,449} & r_g &= \frac{2 \times 171}{9,580} & P(w) &= \frac{0.063845}{0.035699 + 0.063845} \\ &= 0.063845 & &= 0.035699 & &= \mathbf{0.641374} \end{aligned}$$

$w = \text{university} :$

$$w_b = 198 \quad w_g = 1,243$$

$$\begin{aligned} r_b &= \frac{198}{69,449} & r_g &= \frac{2 \times 1243}{9,580} & P(w) &= \frac{0.002851}{0.002851 + 0.259499} \\ &= 0.002851 & &= 0.259499 & &= \mathbf{0.0108672} \end{aligned}$$

$w = \text{and} :$

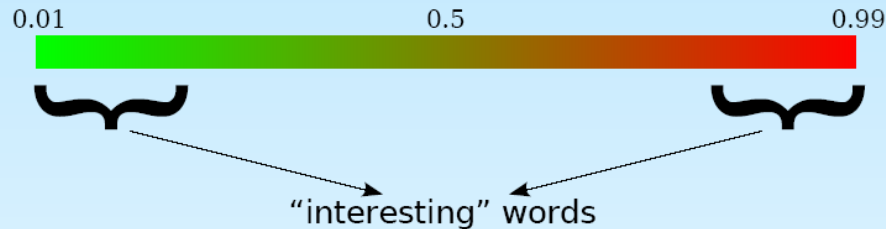
$$w_b = 158,729 \quad w_g = 70,828$$

$$\begin{aligned} r_b &= \min\left(\frac{158,729}{69,449}, 1.0\right) & r_g &= \min\left(\frac{2 \times 70,828}{9,580}, 1.0\right) & P(w) &= \frac{1}{1+1} \\ &= 1 & &= 1 & &= \mathbf{0.5} \end{aligned}$$



# The Filtering Phase

Once we have generated a database of words and their corresponding probabilities, we can start to filter incoming messages. When we receive a message, we identify the  $n$  most *interesting* words,  $w_i$  (Graham's algorithm sets  $n = 15$ ). These are the words whose probabilities are furthest away from neutral.



We then apply the following formula:

$$p = \frac{prod_1}{prod_1 + prod_2}$$

where:

$$prod_1 = \prod_{i=1}^n P(w_i), \quad prod_2 = \prod_{i=1}^n (1 - P(w_i))$$

The closer that  $p$  is to one, the more likely the message is spam.

# Filtering Example #1



From: Michael Rayment  
To: Donald Craig, Geoff Holden  
Organization: Memorial University  
Subject: Abstract for talk

What's with Python Anyway?  
A Look at Why Python is so Popular  
by  
Michael Rayment

This talk will look at how the designers of Python got it right when they designed the language and wrote the Python interpreter. A quick tour of the language will reveal a symmetric object oriented data structure model at the heart of the language along with a rich set of program constructs that allow the programmer to manipulate the data easily. All data are dynamically typed objects that can be assigned to variables without worrying about type declarations. Python "does the right thing" according to the context in which the data is used. The variable name is simply a handle. Among other things this talk will describe some of the salient features of the language such as list manipulation, creating classes, handling exceptions, scope of variables and most importantly the rich set of predefined module libraries.



# Filtering Example #1 (Cont'd)

$P(\text{variables}) = 0.01$   
 $P(\text{Memorial}) = 0.01$   
 $P(\text{declarations}) = 0.01$   
 $P(\text{Craig}) = 0.01$   
 $P(\text{wrote}) = 0.01$   
 $P(\text{Geoff}) = 0.01$   
 $P(\text{Abstract}) = 0.01$   
 $P(\text{Python}) = 0.01$   
 $P(\text{dynamically}) = 0.0104355365$   
 $P(\text{University}) = 0.0108397943$   
 $P(\text{Rayment}) = 0.0108484296$   
 $P(\text{Rayment}) = 0.0108484296$   
 $P(\text{exceptions}) = 0.0132622019$   
 $P(\text{typed}) = 0.0139836777$   
 $P(\text{Anyway}) = 0.0198030382$

$$prod_1 = 4.88921 \times 10^{-30}$$

$$prod_2 = 0.842786$$

$$p = \frac{4.88921 \times 10^{-30}}{4.88921 \times 10^{-30} + 0.842786}$$
$$= 5.80125 \times 10^{-30}$$

Message is not spam!

# Filtering Example #2



```
From cubitus@realtywebsites.com Tue Mar 22 02:16:29 2005
Return-Path: <cubitus@realtywebsites.com>
Received: from realtywebsites.com (unknown [218.85.167.135])
        by mercury.cs.mun.ca (Postfix) with SMTP id OD3DB18B6E3
        for <dona1d@garfield.cs.mun.ca>; Tue, 22 Mar 2005 02:16:18 -0330 (NST)
From: <cubitus@realtywebsites.com>
To: <dona1d@cs.mun.ca>
Subject: At your service, online pharm
Date: Tue, 22 Mar 2005 13:50:20 -0500
Mime-Version: 1.0
Content-Type: text/plain; charset=us-ascii
Message-Id: <20050322054619.0D3DB18B6E3@mercury.cs.mun.ca>
X-Virus-Scanned: by amavisd-new at cs.mun.ca
```

If you need high quality medication and would love to save on outrageous retail pricing, then OnlinePharmacy is for you.

Shopping online for your drug needs saves you the hassle of going to the doctor, answering embarrassing questions and waiting in line to receive good treatment? OnlinePharmacy has been dedicated to serving its online clientel since 2001.

Try us to find out why ordering your medications online has never been easier.

<http://www.fastmail336.com/rx/?76>



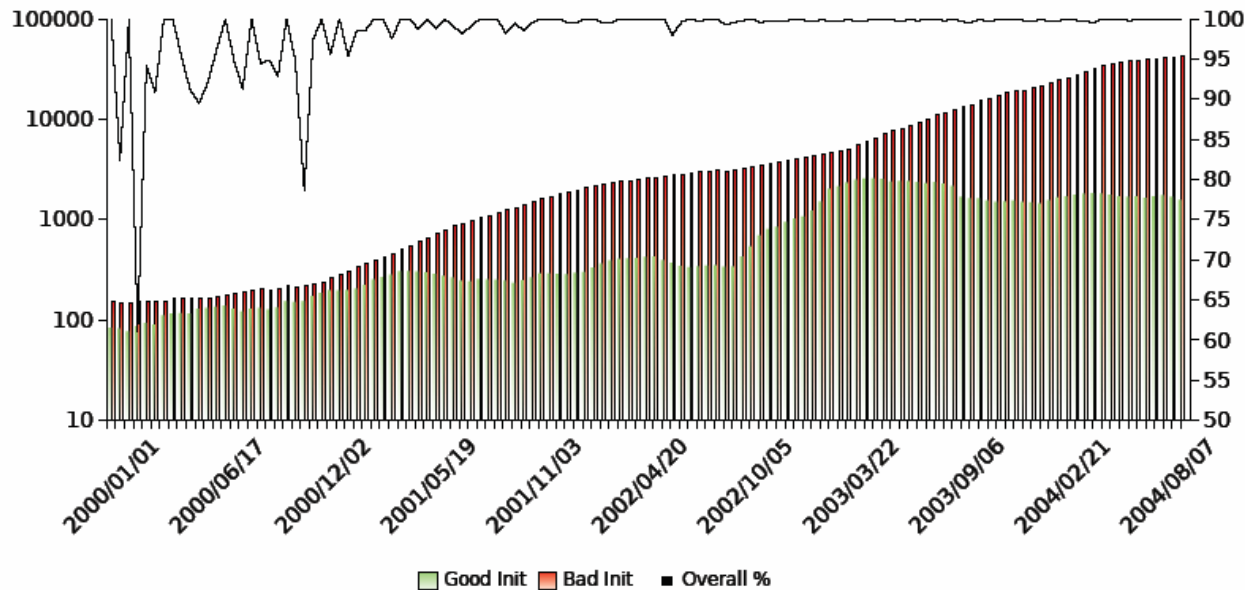
# Filtering Example #2 (cont'd)

$P(\text{OnlinePharmacy})$	$= 0.99$	
$P([218])$	$= 0.99$	
$P(\text{pharm})$	$= 0.0420049316$	
$P(\text{answering})$	$= 0.0459504311$	
$P(\text{unknown})$	$= 0.9534104082$	
$P(\text{retail})$	$= 0.9132323275$	$prod_1 = 3.38881 \times 10^{-9}$
$P(\text{medications})$	$= 0.9128298221$	$prod_2 = 1.28555 \times 10^{-10}$
$P(\text{medication})$	$= 0.9104607169$	$p = \frac{3.38881 \times 10^{-9}}{3.38881 \times 10^{-9} + 1.28555 \times 10^{-10}}$
$P(\text{would})$	$= 0.0919725162$	$= 0.963451$
$P(\text{doctor})$	$= 0.9036781616$	
$P(\text{questions})$	$= 0.1016689722$	
$P(\text{serving})$	$= 0.1121766169$	
$P(\text{dedicated})$	$= 0.1185694551$	
$P(\text{easier})$	$= 0.1433214124$	
$P(\text{going})$	$= 0.1536839685$	

Message is spam!



# Performance



- Initially, the filter performs very poorly, sometimes misclassifying up to one-third of all messages.
- After being initialized with about 2000 bad messages and 200 good messages, the filter hits very close to 100% effectiveness.

# Bayesian filtering: Advantages & Disadvantages



- Advantages
  - Conceptually very easy to understand.
  - Very effective (filters more than 99.691%)
  - Everyone's filter is essentially customized, making it very difficult for spammers to defeat everyone's filter with a single message.
  - Many e-mail clients now either directly or indirectly support Bayesian filtering.
- Disadvantages
  - You need to have a corpus of good and bad messages to initialize the filter.
  - Initialization is a bit time consuming (but this can be made quicker by caching word counts for each mail folder).
  - On each message, a user-specific database of word probabilities has to be consulted. This makes Bayesian filtering somewhat resource intensive and probably not ideal for sites with large user bases.
  - False positives do happen (but are rare).

# Regression and Classification



- Function Approximation
  - Learn  $f: X \rightarrow Y$
  - Learn  $P(Y | X)$

# Bayes Rule



$$P(Y | X) = \frac{P(X | Y)P(Y)}{P(X)}$$

$$(\forall i, j)P(Y = y_i | X = x_j) = \frac{P(X = x_j | Y = y_i)P(Y = y_i)}{P(X = x_j)}$$

How will we represent  $P(X|Y)$ ,  $P(Y)$ ?

How many parameters must we estimate?



# Naïve Bayes

- Suppose  $X = \{X_1, X_2, \dots, X_n\}$
- Naïve Bayes assumption:

$$P(X | Y) = \prod_i P(X_i | Y)$$

i.e., that  $X_i$  and  $X_j$  are conditionally independent, given  $Y$

# Naïve Bayes Classifier



Assume target function  $f : X \rightarrow V$ , where each instance  $x$  described by attributes  $\langle a_1, a_2 \dots a_n \rangle$ .  
Most probable value of  $f(x)$  is:

$$\begin{aligned}v_{MAP} &= \operatorname{argmax}_{v_j \in V} P(v_j | a_1, a_2 \dots a_n) \\v_{MAP} &= \operatorname{argmax}_{v_j \in V} \frac{P(a_1, a_2 \dots a_n | v_j) P(v_j)}{P(a_1, a_2 \dots a_n)} \\ &= \operatorname{argmax}_{v_j \in V} P(a_1, a_2 \dots a_n | v_j) P(v_j)\end{aligned}$$

Naive Bayes assumption:

$$P(a_1, a_2 \dots a_n | v_j) = \prod_i P(a_i | v_j)$$

which gives

**Naive Bayes classifier:**  $v_{NB} = \operatorname{argmax}_{v_j \in V} P(v_j) \prod_i P(a_i | v_j)$

# Naïve Bayes Algorithm



Naive\_Bayes\_Learn(*examples*)

For each target value  $v_j$

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

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

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

Classify\_New\_Instance( $x$ )

$$v_{NB} = \operatorname{argmax}_{v_j \in V} \hat{P}(v_j) \prod_{a_i \in x} \hat{P}(a_i|v_j)$$

# Naïve Bayes: a simple example



Sky	Temp	Humid	Wind	Water	Forecst	EnjoySpt
Sunny	Warm	Normal	Strong	Warm	Same	Yes
Sunny	Warm	High	Strong	Warm	Same	Yes
Rainy	Cold	High	Strong	Warm	Change	No
Sunny	Warm	High	Strong	Cool	Change	Yes

Consider *PlayTennis* again, and new instance

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

Want to compute:

$$v_{NB} = \operatorname{argmax}_{v_j \in V} P(v_j) \prod_i P(a_i | v_j)$$

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

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

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





# What you should know:

- Learning (generative) classifiers based on Bayes rule
- Conditional independence
- Naïve Bayes assumption and its consequences
- Naïve Bayes with discrete inputs, continuous inputs



# Homework

- What if we have continuous  $X_i$

$$P(x | \mu, \delta) \sim \frac{1}{\delta \sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\delta^2}}$$