What is Machine Learning?

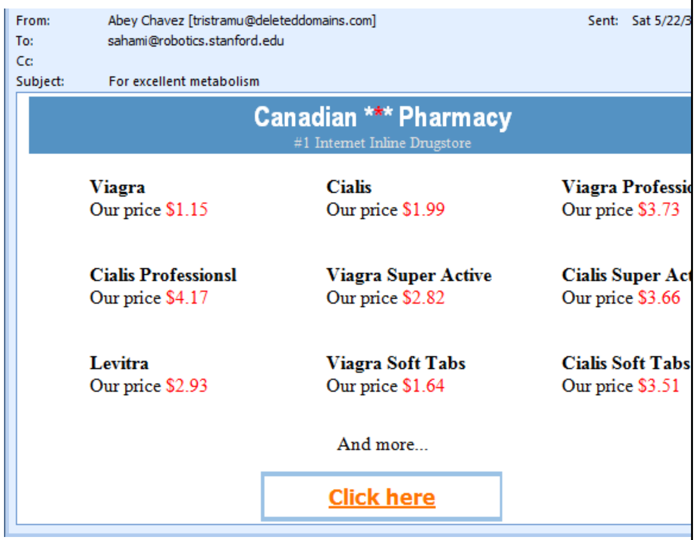
- Many different forms of “Machine Learning”
  - We focus on the problem of *prediction*
- Want to make a prediction based on observations
  - Vector **X** of *m* observed variables: $<X_1, X_2, \ldots, X_m>$
    - $X_1, X_2, \ldots, X_m$ are called “input features/variables”
    - Also called “independent variables,” but this can be misleading!
      - $X_1, X_2, \ldots, X_m$ need not be (and usually are not) independent
  - Based on observed **X**, want to predict unseen variable Y
    - Y called “output feature/variable” (or the “dependent variable”)
  - Seek to “learn” a function $g(\boldsymbol{X})$ to predict Y: $\hat{Y} = g(\boldsymbol{X})$
    - When Y is discrete, prediction of Y is called “classification”
    - When Y is continuous, prediction of Y is called “regression”

A (Very Short) List of Applications

- Machine learning widely used in many contexts
  - Stock price prediction
    - Using economic indicators, predict if stock will go up/down
  - Computational biology and medical diagnosis
    - Predicting gene expression based on DNA
    - Determine likelihood for cancer using clinical/demographic data
  - Predict people likely to purchase product or click on ad
    - *“Based on past purchases, you might want to buy…”*
  - Credit card fraud and telephone fraud detection
    - Based on past purchases/phone calls is a new one fraudulent?
      - Saves companies *billions(!)* of dollars annually
  - Spam E-mail detection (gmail, hotmail, many others)

What is Bayes Doing in My Mail Server?

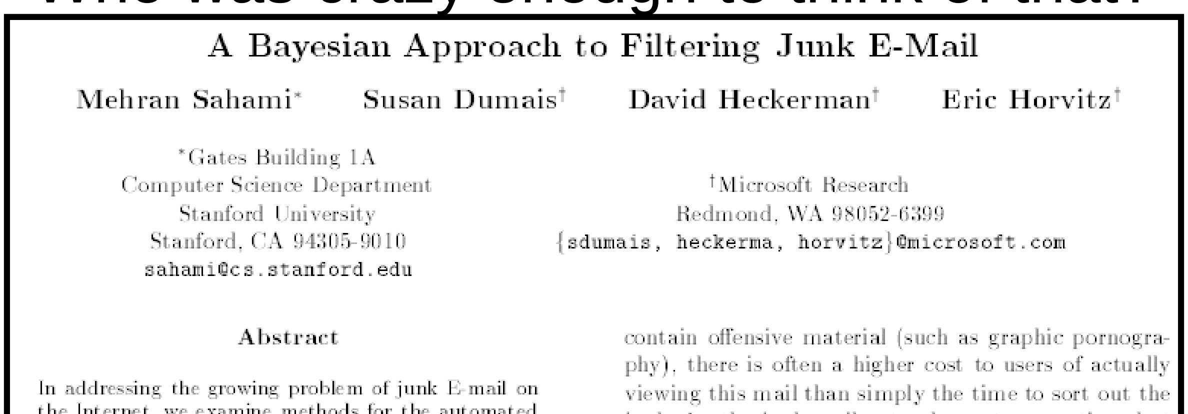
- This is spam:



Let's get Bayesian on your spam:

```
Content analysis details:   (49.5 hits, 7.0 required)
 0.9 RCVD_IN_PBL            RBL: Received via a relay in Spamhaus PBL
                            [93.40.189.29 listed in zen.spamhaus.org]
 1.5 URIBL_WS_SURBL         Contains an URL listed in the WS SURBL blocklist
                            [URIs: recragas.cn]
 5.0 URIBL_JP_SURBL         Contains an URL listed in the JP SURBL blocklist
                            [URIs: recragas.cn]
 5.0 URIBL_OB_SURBL         Contains an URL listed in the OB SURBL blocklist
                            [URIs: recragas.cn]
 5.0 URIBL_SC_SURBL         Contains an URL listed in the SC SURBL blocklist
                            [URIs: recragas.cn]
 2.0 URIBL_BLACK            Contains an URL listed in the URIBL blacklist
                            [URIs: recragas.cn]
 8.0 BAYES_99               BODY: Bayesian spam probability is 99 to 100%
                            [score: 1.0000]
```

Who was crazy enough to think of that?



A Bayesian Approach to Filtering Junk E-Mail

Mehran Sahami* Susan Dumais† David Heckerman† Eric Horvitz†

*Gates Building 1A
Computer Science Department
Stanford University
Stanford, CA 94305-9010
sahami@cs.stanford.edu

†Microsoft Research
Redmond, WA 98052-6399
{sdumais, heckerma, horvitz}@microsoft.com

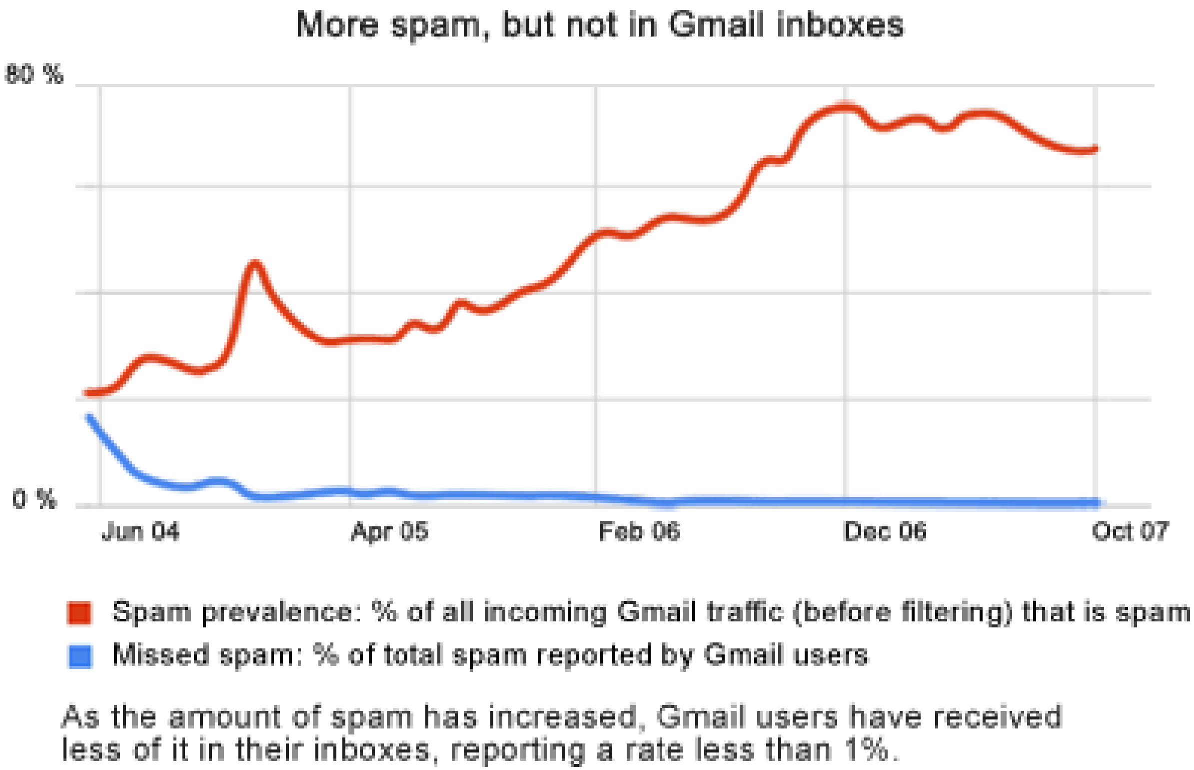
Abstract

In addressing the growing problem of junk E-mail on the Internet, we examine methods for the automated

contain offensive material (such as graphic pornography), there is often a higher cost to users of actually viewing this mail than simply the time to sort out the junk. Lastly, junk mail not only wastes user time, but

Spam, Spam… Go Away!

- The constant battle with spam



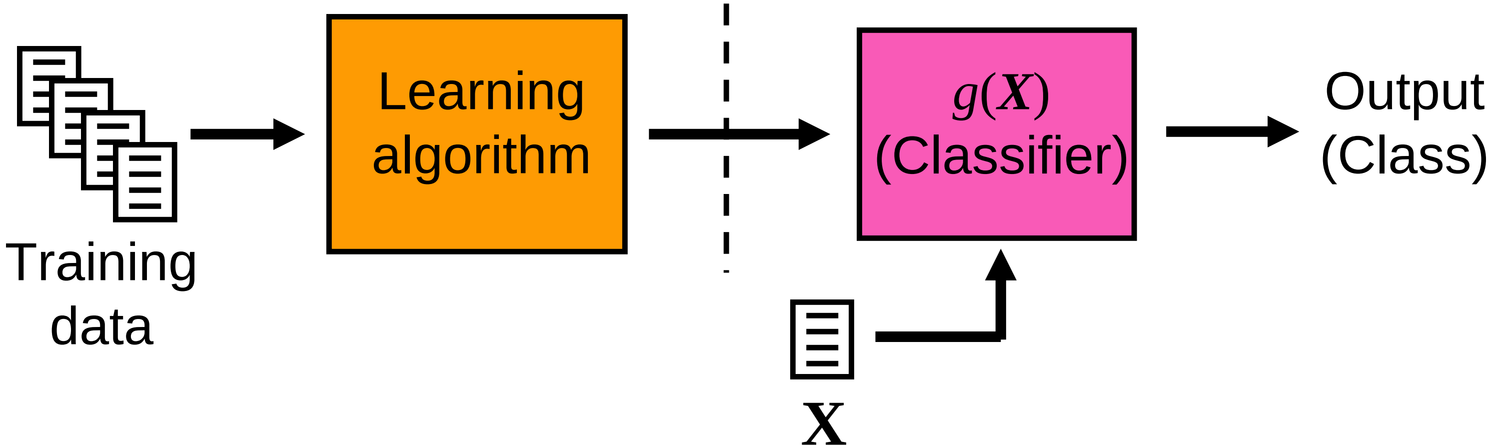
"And machine-learning algorithms developed to merge and rank large sets of Google search results allow us to combine hundreds of factors to classify spam."

Source: **`http://www.google.com/mail/help/fightspam/spamexplained.html`**

Training a Learning Machine

- We consider statistical learning paradigm here
  - We are given set of *N* “training” *instances*
    - Each training instance is pair: (<x_1, x_2, …, x_m>, y)
    - Training instances are *previously* observed data
    - Gives the output value *y* associated with each observed vector of input values <x_1, x_2, …, x_m>
  - Learning: use training data to specify $g(\boldsymbol{X})$
    - Generally, first select a parametric form for $g(\boldsymbol{X})$
    - Then, estimate parameters of model $g(\boldsymbol{X})$ using training data
    - For regression, usually want $g(\boldsymbol{X})$ that minimizes $E[(Y - g(\boldsymbol{X}))^2]$
      - *Mean squared error* (MSE) “loss” function. (Others exist.)
    - For classification, generally best choice of $g(\boldsymbol{X}) = \arg\max_y \hat{P}(Y \mid \boldsymbol{X})$

The Machine Learning Process



- Training data: set of *N* pre-classified data instances
  - *N* training pairs: $(\langle x\rangle^{(1)},y^{(1)}), (\langle x\rangle^{(2)},y^{(2)}), \ldots, (\langle x\rangle^{(N)}, y^{(N)})$
    - Use superscripts to denote *i*-th training instance
- Learning algorithm: method for determining $g(\boldsymbol{X})$
  - Given a new input observation of $\mathbf{X} = \langle X_1, X_2, \ldots, X_m\rangle$
  - Use $g(\boldsymbol{X})$ to compute a corresponding output (prediction)
  - When prediction is discrete, we call $g(\boldsymbol{X})$ a “classifier” and call the output the predicted “class” of the input

A Grounding Example: Linear Regression

- Predict real value Y based on observing variable X
  - Assume model is linear: $\hat{Y} = g(\boldsymbol{X}) = aX + b$
  - Training data
    - Each vector **X** has one observed variable: <X_1> (just call it X)
    - Y is continuous output variable
    - Given *N* training pairs: (<x>$^{(1)}$,y$^{(1)}$), (<x>$^{(2)}$,y$^{(2)}$), …, (<x>$^{(N)}$, y$^{(N)}$)
      - Use superscripts to denote *i*-th training instance
  - Determine *a* and *b* minimizing E[(Y – $g(\boldsymbol{X})$)^2]
    - First, minimize objective function:

$$E[(Y - g(X))^2] = E[(Y - (aX + b))^2] = E[(Y - aX - b)^2]$$

Don’t Make Me Get Non-Linear!

- Minimize objective function $E[(Y - aX - b)^2]$
  - Compute derivatives w.r.t. *a* and *b*

$$\frac{\partial}{\partial a} E[(Y - aX - b)^2] = E[-2X(Y - aX - b)] = -2E[XY] + 2aE[X^2] + 2bE[X]$$

$$\frac{\partial}{\partial b} E[(Y - aX - b)^2] = E[-2(Y - aX - b)] = -2E[Y] + 2aE[X] + 2b$$

  - Set derivatives to 0 and solve simultaneous equations:

$$a = \frac{E[XY] - E[X]E[Y]}{E[X^2] - (E[X])^2} = \frac{Cov(X,Y)}{Var(X)} = \rho(X,Y)\frac{\sigma_y}{\sigma_x}$$

$$b = E[Y] - aE[X] = \mu_y - \rho(X,Y)\frac{\sigma_y}{\sigma_x}\mu_x$$

  - Substitution yields: $Y = \rho(X,Y)\frac{\sigma_y}{\sigma_x}(X - \mu_x) + \mu_y$
  - Estimate parameters based on observed training data:

$$\hat{Y} = g(X = x) = \hat{\rho}(X,Y)\frac{\hat{\sigma}_y}{\hat{\sigma}_x}(x - \overline{X}) + \overline{Y}$$

A Simple Classification Example

- Predict Y based on observing variables X
  - X has discrete value from {1, 2, 3, 4}
    - X denotes temperature range today: <50, 50-60, 60-70, >70
  - Y has discrete value from {rain, sun}
    - Y denotes general weather outlook tomorrow
  - Given training data, estimate joint PMF: $\hat{p}_{X,Y}(x, y)$
  - Note Bayes' Thm.: $P(Y \mid X) = \dfrac{p_{X,Y}(x, y)}{p_X(x)} = \dfrac{p_{X|Y}(x \mid y)\, p_Y(y)}{p_X(x)}$
  - For new X, predict $\hat{Y} = g(\boldsymbol{X}) = \arg\max_y \hat{P}(Y \mid X)$
    - Note $p_X(x)$ is not affected by choice of y, yielding:

$$\hat{Y} = g(\boldsymbol{X}) = \arg\max_y \hat{P}(Y \mid X) = \arg\max_y \hat{P}(X, Y) = \arg\max_y \hat{P}(X \mid Y)\hat{P}(Y)$$

Estimating the Joint PMF

- Given training data, compute joint PMF: $p_{X,Y}(x, y)$
  - **MLE**: count number of times each pair (x, y) appears
  - **MAP using Laplace prior**: add 1 to all the MLE counts
  - Normalize to get true distribution (sums to 1)
  - Observed 50 data points:

Y \ X	1	2	3	4
rain	5	3	2	0
sun	3	7	10	20

$$\hat{p}_{MLE} = \frac{\text{count in cell}}{\text{total \# data points}}$$

$$\hat{p}_{Laplace} = \frac{\text{count in cell} + 1}{\text{total \# data points} + \text{total \# cells}}$$

MLE estimate

Y \ X	1	2	3	4	$p_Y(y)$
rain	0.10	0.06	0.04	0.00	0.20
sun	0.06	0.14	0.20	0.40	0.80
$p_X(x)$	0.16	0.20	0.24	0.40	1.00

Laplace (MAP) estimate

Y \ X	1	2	3	4	$p_Y(y)$
rain	0.103	0.069	0.052	0.017	0.241
sun	0.069	0.138	0.190	0.362	0.759
$p_X(x)$	0.172	0.207	0.242	0.379	1.00

Classify New Observation

- Say today's temperature is 75, so X = 4
  - Recall X temperature ranges: <50, 50-60, 60-70, >70
  - Prediction for Y (weather outlook tomorrow)

$$\hat{Y} = \arg\max_{y} \hat{P}(X, Y) = \arg\max_{y} \hat{P}(X \mid Y)\hat{P}(Y)$$

MLE estimate

Y \ X	1	2	3	4	$p_Y(y)$
rain	**0.10**	**0.06**	**0.04**	**0.00**	**0.20**
sun	**0.06**	**0.14**	**0.20**	**0.40**	**0.80**
$p_X(x)$	**0.16**	**0.20**	**0.24**	**0.40**	**1.00**

Laplace (MAP) estimate

Y \ X	1	2	3	4	$p_Y(y)$
rain	**0.103**	**0.069**	**0.052**	**0.017**	**0.241**
sun	**0.069**	**0.138**	**0.190**	**0.362**	**0.759**
$p_X(x)$	**0.172**	**0.207**	**0.242**	**0.379**	**1.00**

  - What if we asked what is probability of rain tomorrow?
    - MLE: absolutely, positively no chance of rain!
    - Laplace estimate: small chance → "never say never"

Classification with Multiple Observables

- Say, we have *m* input values $\mathbf{X} = \langle X_1, X_2, \ldots, X_m \rangle$
  - Note that variables $X_1, X_2, \ldots, X_m$ can be dependent!
  - In *theory*, could predict Y as before, using

$$\hat{Y} = \arg\max_{y} \hat{P}(\boldsymbol{X}, Y) = \arg\max_{y} \hat{P}(\boldsymbol{X} \mid Y)\hat{P}(Y)$$

    - Why won't this necessarily work in practice?
  - Need to estimate $P(X_1, X_2, \ldots, X_m \mid Y)$
    - Fine if *m* is small, but what if m = 10 or 100 or 10,000?
    - Note: size of PMF table is <u>exponential</u> in *m* (e.g. $O(2^m)$)
    - Need ridiculous amount of data for good probability estimates!
    - Likely to have many 0's in table (bad times)
  - Need to consider a simpler model

Naive Bayesian Classifier

- Say, we have *m* input values $\mathbf{X} = \langle X_1, X_2, \ldots, X_m \rangle$
  - Assume variables $X_1, X_2, \ldots, X_m$ are **<u>conditionally independent</u>** given Y
    - Really don't believe $X_1, X_2, \ldots, X_m$ are conditionally independent
    - Just an approximation we make to be able to make predictions
    - This is called the "Naive Bayes" assumption, hence the name
  - Predict Y using $\hat{Y} = \arg\max_y P(\boldsymbol{X}, Y) = \arg\max_y P(\boldsymbol{X} \mid Y) P(Y)$
    - But, we now have:

$$P(\boldsymbol{X} \mid Y) = P(X_1, X_2, \ldots X_m \mid Y) = \prod_{i=1}^{m} P(X_i \mid Y)$$ by conditional independence

  - Note: computation of PMF table is <u>linear</u> in *m* : O(*m*)
    - Don't need much data to get good probability estimates

Naive Bayes Example

- Predict Y based on observing variables X_1 and X_2
  - X_1 and X_2 are both indicator variables
    - X_1 denotes "likes Star Wars", X_2 denotes "likes Harry Potter"
  - Y is indicator variable: "likes Lord of the Rings"
    - Use training data to estimate PMFs: $\hat{p}_{X_i,Y}(x_i, y), \;\; \hat{p}_Y(y)$

Y \ X_1	0	1	MLE estimates	
0	3	10	0.10	0.33
1	4	13	0.13	0.43

Y \ X_2	0	1	MLE estimates	
0	5	8	0.17	0.27
1	7	10	0.23	0.33

Y	#	MLE est.
0	13	0.43
1	17	0.57

  - Say someone likes Star Wars ($X_1 = 1$), but not Harry Potter ($X_2 = 0$)
  - Will they like "Lord of the Rings"? Need to predict Y:

$$\hat{Y} = \arg\max_y \hat{P}(\mathbf{X} \mid Y)\hat{P}(Y) = \arg\max_y \hat{P}(X_1 \mid Y)\hat{P}(X_2 \mid Y)\hat{P}(Y)$$

"All Your Bayes Are Belong To Us"

Y \ X_1	0	1	MLE estimates	
0	3	10	0.10	0.33
1	4	13	0.13	0.43

Y \ X_2	0	1	MLE estimates	
0	5	8	0.17	0.27
1	7	10	0.23	0.33

Y	#	MLE est.
0	13	0.43
1	17	0.57

- Prediction for Y is value of Y maximizing P(**X**, Y):

$$\hat{Y} = \arg\max_y \hat{P}(\mathbf{X} \mid Y)\hat{P}(Y) = \arg\max_y \hat{P}(X_1 \mid Y)\hat{P}(X_2 \mid Y)\hat{P}(Y)$$

  - Compute P(**X**, Y=0): $\hat{P}(X_1 = 1 \mid Y = 0)\hat{P}(X_2 = 0 \mid Y = 0)\hat{P}(Y = 0)$

$$= \frac{\hat{P}(X_1 = 1, Y = 0)}{\hat{P}(Y = 0)} \frac{\hat{P}(X_2 = 0, Y = 0)}{\hat{P}(Y = 0)} \hat{P}(Y = 0) \approx \frac{0.33}{0.43}\frac{0.17}{0.43} 0.43 \approx 0.13$$

  - Compute P(**X**, Y=1): $\hat{P}(X_1 = 1 \mid Y = 1)\hat{P}(X_2 = 0 \mid Y = 1)\hat{P}(Y = 1)$

$$= \frac{\hat{P}(X_1 = 1, Y = 1)}{\hat{P}(Y = 1)} \frac{\hat{P}(X_2 = 0, Y = 1)}{\hat{P}(Y = 1)} \hat{P}(Y = 1) \approx \frac{0.43}{0.57}\frac{0.23}{0.57} 0.57 \approx 0.17$$

  - Since P(**X**, Y=1) > P(**X**, Y=0), we predict Ŷ = 1

Email Classification

- Want to predict if an email is spam or not
  - Start with the input data
    - Consider a lexicon of m words (Note: in English $m \approx 100{,}000$)
    - Define m indicator variables $\mathbf{X} = \langle X_1, X_2, \ldots, X_m \rangle$
    - Each variable X_i denotes if word i appeared in a document or not
    - Note: m is huge, so make "Naive Bayes" assumption
  - Define output classes Y to be: {spam, non-spam}
  - Given training set of N previous emails
    - For each email message, we have a training instance:
      $\mathbf{X} = \langle X_1, X_2, \ldots, X_m \rangle$ noting for each word, if it appeared in email
    - Each email message is also marked as spam or not (value of Y)

Training the Classifier

- Given *N* training pairs:

  $(\langle x\rangle^{(1)}, y^{(1)}), (\langle x\rangle^{(2)}, y^{(2)}), \ldots, (\langle x\rangle^{(N)}, y^{(N)})$

- Learning
  - Estimate probabilities P(Y) and each P(X_i | Y) for all *i*
    - Many words are likely to not appear at all in given set of email
  - Laplace estimate: $\hat{p}(X_i = 1 \mid Y = spam)_{Laplace} = \dfrac{(\#\,\text{spam emails with word } i) + 1}{\text{total}\,\#\,\text{spam emails} + 2}$
- Classification
  - For a new email, generate **X** = <X_1, X_2, …, X_m>
  - Classify as spam or not using: $\hat{Y} = \arg\max_{y} \hat{P}(\boldsymbol{X} \mid Y)\hat{P}(Y)$
  - Employ Naive Bayes assumption: $\hat{P}(\boldsymbol{X} \mid Y) = \prod_{i=1}^{m} \hat{P}(X_i \mid Y)$

How Does This Do?

- After training, can test with another set of data
  - "Testing" set also has known values for Y, so we can see how often we were right/wrong in predictions for Y
  - Spam data
    - Email data set: 1789 emails (1578 spam, 211 non-spam)
    - First, 1538 email messages (by time) used for training
    - Next 251 messages used to test learned classifier
  - Criteria:
    - Precision = # *correctly* predicted class Y/ # predicted class Y
    - Recall = # *correctly* predicted class Y / # real class Y messages

	Spam		**Non-spam**	
	Precision	**Recall**	**Precision**	**Recall**
Words only	**97.1%**	**94.3%**	**87.7%**	**93.4%**
Words + add'l features	**100%**	**98.3%**	**96.2%**	**100%**