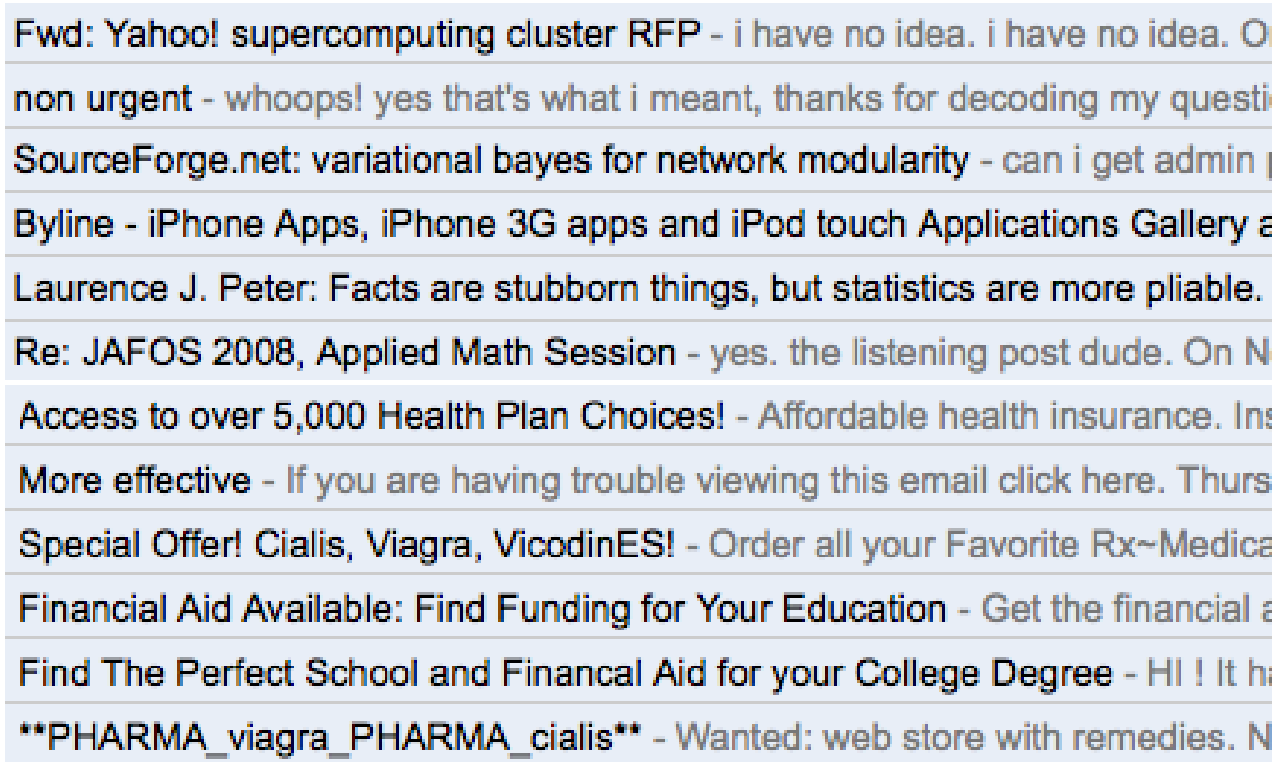
Classification with Naive Bayes

Jake Hofman

Microsoft Research

March 16, 2018

Learning by example



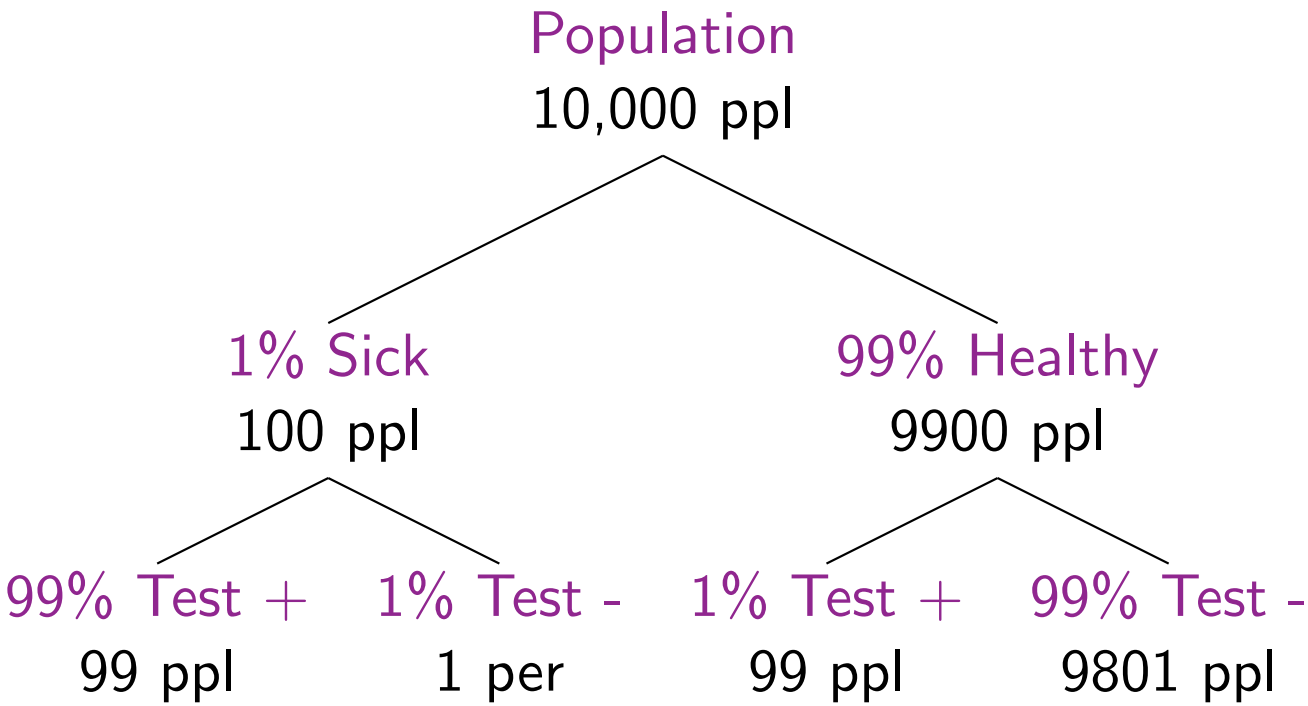
- How did you solve this problem?
- Can you make this process explicit (e.g. write code to do so)?

Diagnoses a la Bayes[1]

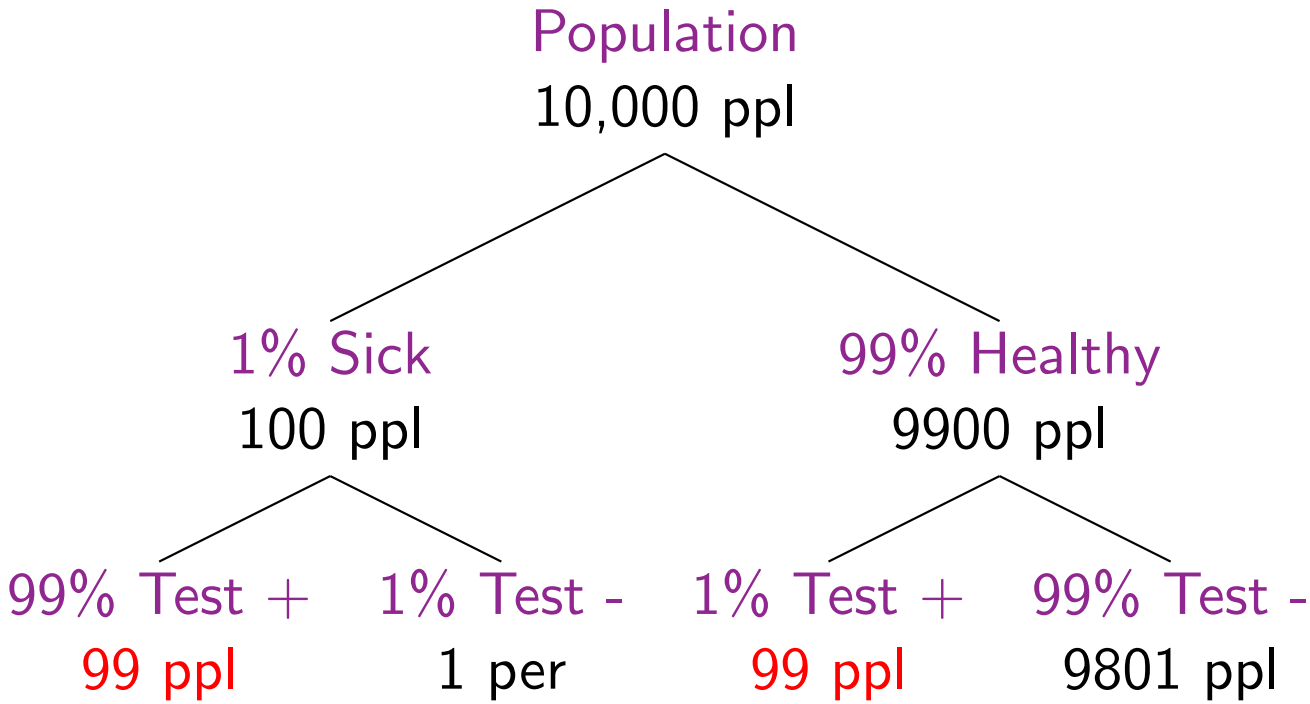
- You're testing for a rare disease:
  - 1% of the population is infected
- You have a highly sensitive and specific test:
  - 99% of sick patients test positive
  - 99% of healthy patients test negative
- Given that a patient tests positive, what is probability the patient is sick?

[1] Wiggins, SciAm 2006

Diagnoses a la Bayes

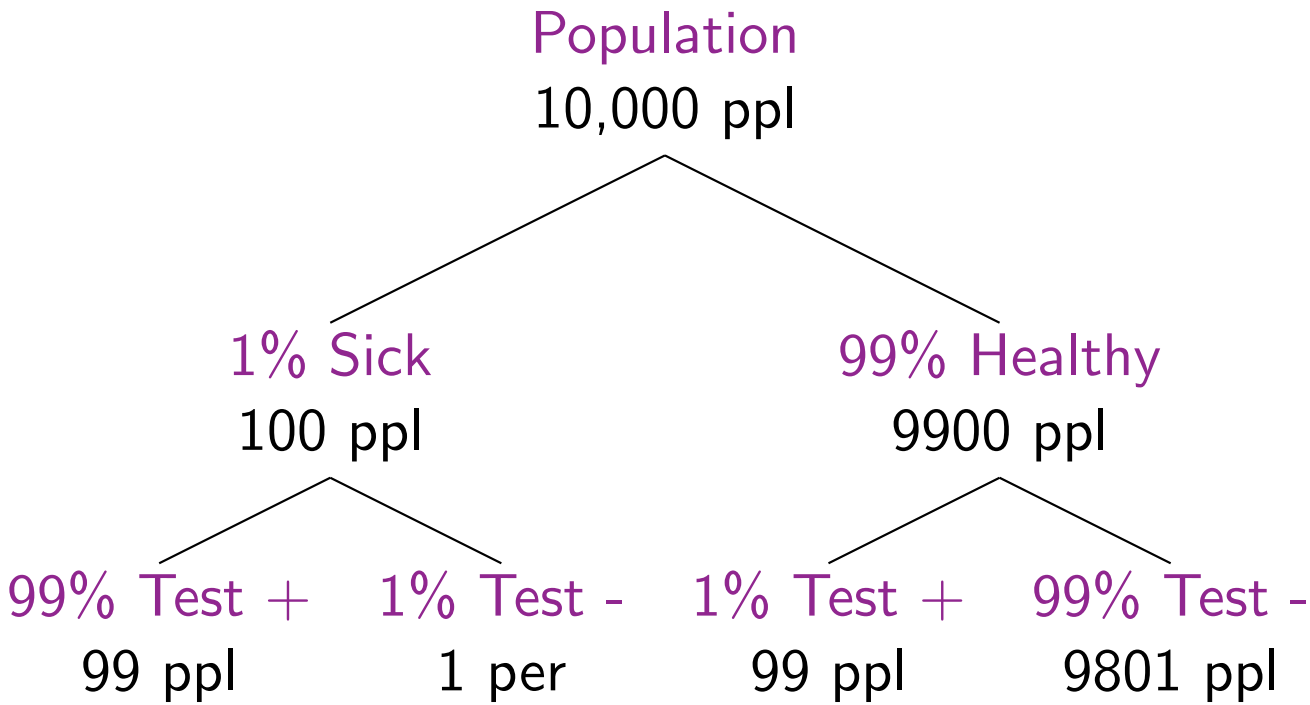


Diagnoses a la Bayes



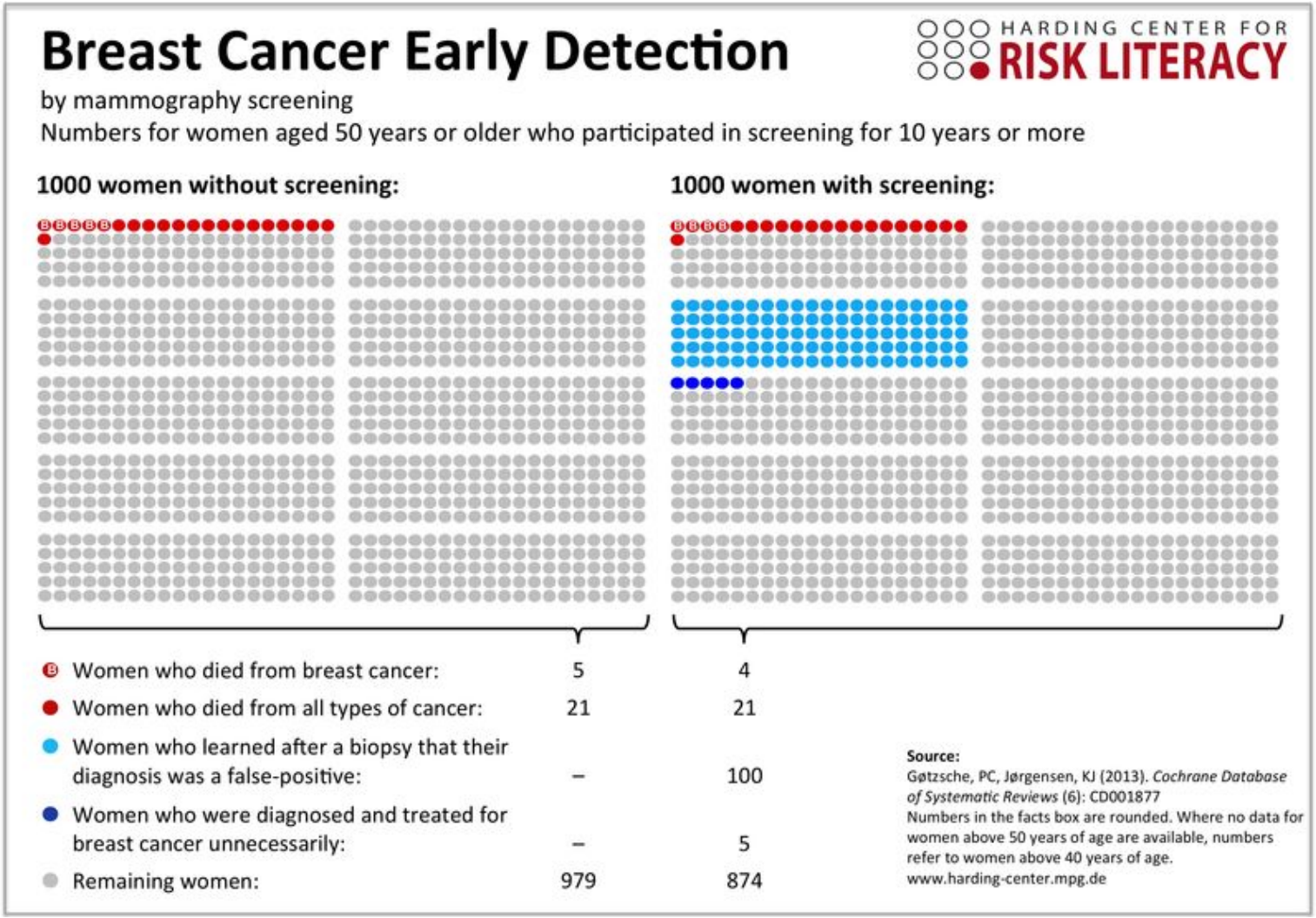
So given that a patient tests positive (198 ppl), there is a 50% chance the patient is sick (99 ppl)!

Diagnoses a la Bayes



The small error rate on the large healthy population produces many false positives.

Natural frequencies a la Gigerenzer[2]



[2] http://bit.ly/ggbbc

Inverting conditional probabilities

Bayes' Theorem

Equate the far right- and left-hand sides of product rule

$$p(y|x)\,p(x) = p(x,y) = p(x|y)\,p(y)$$

and divide to get the probability of y given x from the probability of x given y:

$$p(y|x) = \frac{p(x|y)\,p(y)}{p(x)}$$

where $p(x) = \sum_{y \in \Omega_Y} p(x|y)\,p(y)$ is the normalization constant.

Diagnoses a la Bayes

Given that a patient tests positive, what is probability the patient is sick?

$$p\left(sick|+\right) = \frac{\overbrace{p\left(+|sick\right)}^{99/100}\overbrace{p\left(sick\right)}^{1/100}}{\underbrace{p\left(+\right)}_{99/100^2+99/100^2=198/100^2}} = \frac{99}{198} = \frac{1}{2}$$

where $p\left(+\right) = p\left(+|sick\right)p\left(sick\right) + p\left(+|healthy\right)p\left(healthy\right)$.

(Super) Naive Bayes

We can use Bayes' rule to build a one-word spam classifier:

$$p\left(spam|word\right) = \frac{p\left(word|spam\right)p\left(spam\right)}{p\left(word\right)}$$

where we estimate these probabilities with ratios of counts:

$$\begin{aligned}
\hat{p}(word|spam) &= \frac{\#\text{ spam docs containing word}}{\#\text{ spam docs}} \\
\hat{p}(word|ham) &= \frac{\#\text{ ham docs containing word}}{\#\text{ ham docs}} \\
\hat{p}(spam) &= \frac{\#\text{ spam docs}}{\#\text{ docs}} \\
\hat{p}(ham) &= \frac{\#\text{ ham docs}}{\#\text{ docs}}
\end{aligned}$$

(Super) Naive Bayes

```
$ ./enron_naive_bayes.sh meeting
1500 spam examples
3672 ham examples
16 spam examples containing meeting
153 ham examples containing meeting

estimated P(spam) = .2900
estimated P(ham) = .7100
estimated P(meeting|spam) = .0106
estimated P(meeting|ham) = .0416

P(spam|meeting) = .0923
```

(Super) Naive Bayes

```
$ ./enron_naive_bayes.sh money
1500 spam examples
3672 ham examples
194 spam examples containing money
50 ham examples containing money

estimated P(spam) = .2900
estimated P(ham) = .7100
estimated P(money|spam) = .1293
estimated P(money|ham) = .0136

P(spam|money) = .7957
```

(Super) Naive Bayes

```
$ ./enron_naive_bayes.sh enron
1500 spam examples
3672 ham examples
0 spam examples containing enron
1478 ham examples containing enron

estimated P(spam) = .2900
estimated P(ham) = .7100
estimated P(enron|spam) = 0
estimated P(enron|ham) = .4025

P(spam|enron) = 0
```

Naive Bayes

Represent each document by a binary vector $\vec{x}$ where $x_j = 1$ if the j-th word appears in the document ($x_j = 0$ otherwise).

Modeling each word as an *independent* Bernoulli random variable, the probability of observing a document $\vec{x}$ of class c is:

$$p(\vec{x}|c) = \prod_j \theta_{jc}^{x_j}(1-\theta_{jc})^{1-x_j}$$

where θ_{jc} denotes the probability that the j-th word occurs in a document of class c.

Naive Bayes

Using this likelihood in Bayes' rule and taking a logarithm, we have:

$$\begin{aligned}\log p\left(c|\vec{x}\right) &= \log \frac{p\left(\vec{x}|c\right)p\left(c\right)}{p\left(\vec{x}\right)} \\ &= \sum_j x_j \log \frac{\theta_{jc}}{1-\theta_{jc}} + \sum_j \log(1-\theta_{jc}) + \log \frac{\theta_c}{p\left(\vec{x}\right)}\end{aligned}$$

where θ_c is the probability of observing a document of class c.

Naive Bayes

We can eliminate $p(\vec{x})$ by calculating the log-odds:

$$\log \frac{p(1|\vec{x})}{p(0|\vec{x})} = \sum_j x_j \underbrace{\log \frac{\theta_{j1}(1-\theta_{j0})}{\theta_{j0}(1-\theta_{j1})}}_{w_j} + \underbrace{\sum_j \log \frac{1-\theta_{j1}}{1-\theta_{j0}} + \log \frac{\theta_1}{\theta_0}}_{w_0}$$

which gives a linear classifier of the form $\vec{w} \cdot \vec{x} + w_0$

Naive Bayes

We train by counting words and documents within classes to estimate θ_{jc} and θ_c:

$$\begin{aligned} \hat{\theta}_{jc} &= \frac{n_{jc}}{n_c} \\ \hat{\theta}_c &= \frac{n_c}{n} \end{aligned}$$

and use these to calculate the weights $\hat{w}_j$ and bias $\hat{w}_0$:

$$\begin{aligned} \hat{w}_j &= \log \frac{\hat{\theta}_{j1}(1-\hat{\theta}_{j0})}{\hat{\theta}_{j0}(1-\hat{\theta}_{j1})} \\ \hat{w}_0 &= \sum_j \log \frac{1-\hat{\theta}_{j1}}{1-\hat{\theta}_{j0}} + \log \frac{\hat{\theta}_1}{\hat{\theta}_0}. \end{aligned}$$

We we predict by simply adding the weights of the words that appear in the document to the bias term.

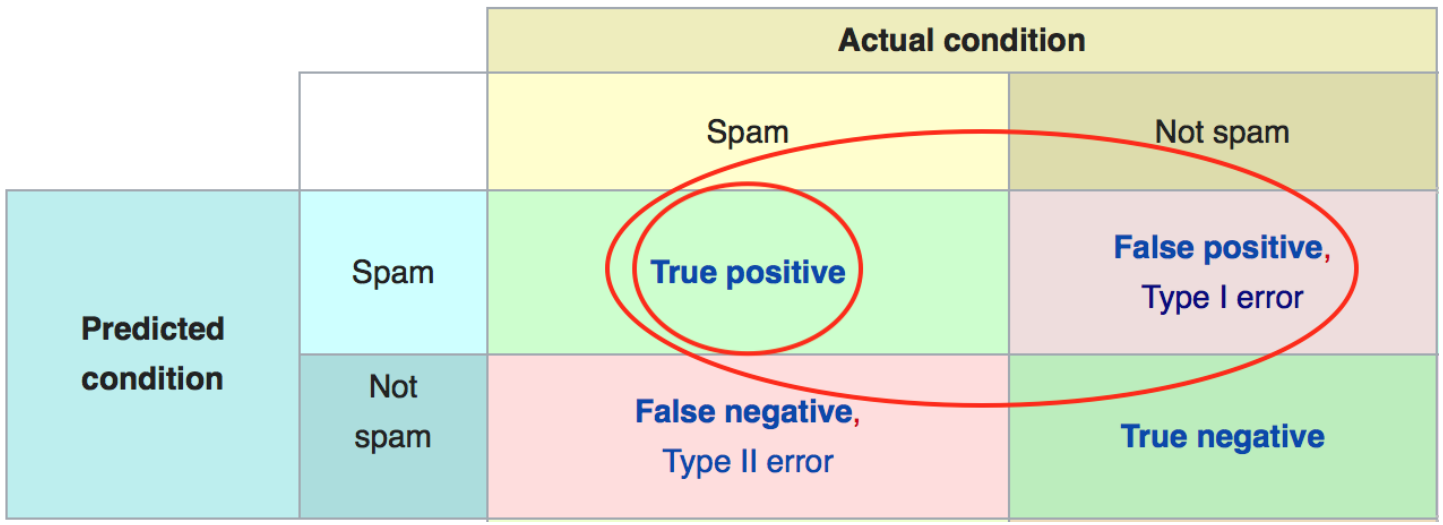
Measures of success

Accuracy: The fraction of examples correctly classified

		Actual condition	
		Spam	Not spam
Predicted condition	Spam	**True positive**	**False positive**, Type I error
	Not spam	**False negative**, Type II error	**True negative**

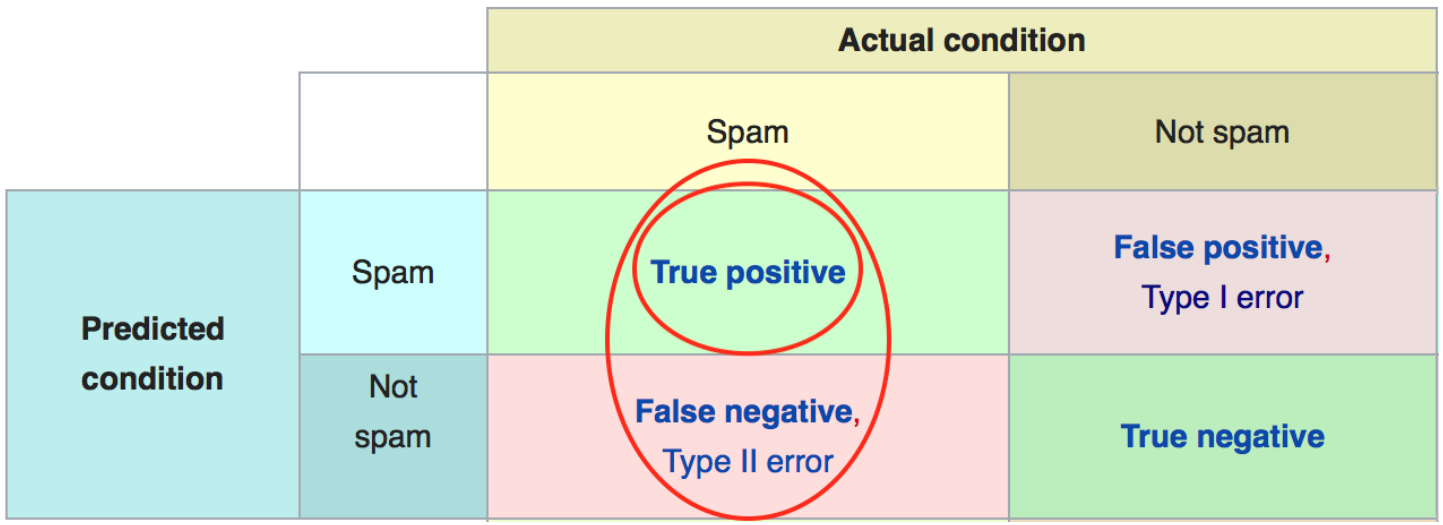
Measures of success

Precision: The fraction of predicted spam that's actually spam



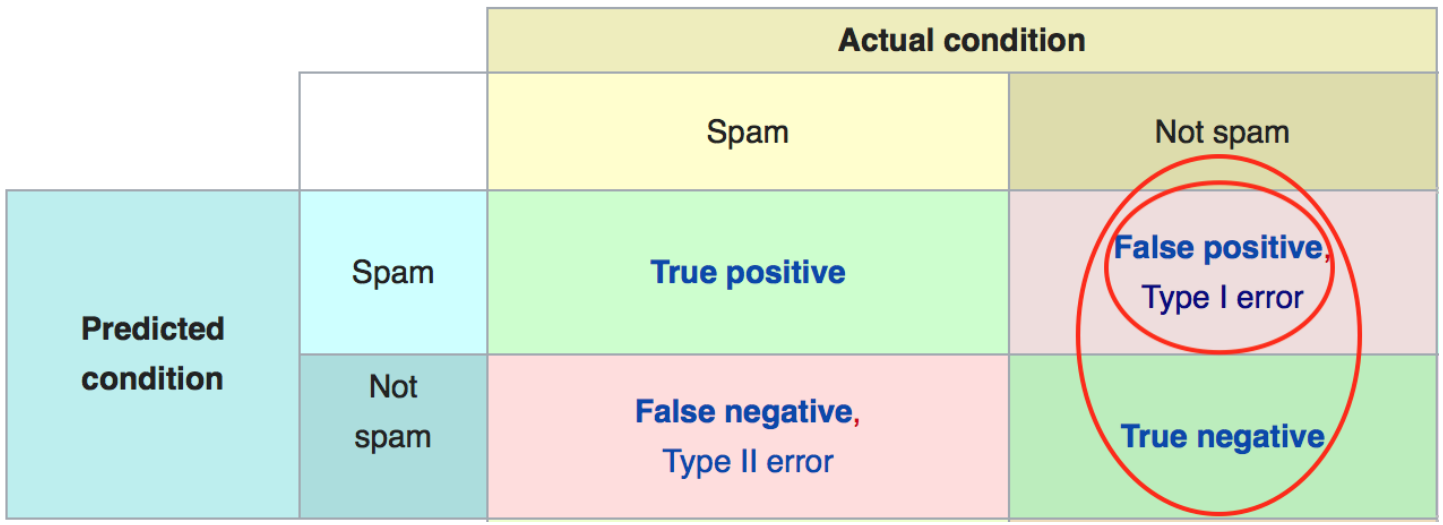
Measures of success

Recall: The fraction of all spam that's predicted to be spam



Measures of success

False positive rate: The fraction of legitimate email that's predicted to be spam



Naive Bayes

In practice, this works better than one might expect given its simplicity[3]

[3] http://www.jstor.org/pss/1403452

Naive Bayes

Training is computationally cheap and scalable, and the model is easy to update given new observations[3]

[3] http://www.springerlink.com/content/wu3g458834583125/

Naive Bayes

Performance varies with document representations and corresponding likelihood models[3]

[3] `http://ceas.cc/2006/15.pdf`

Naive Bayes

It's often important to smooth parameter estimates (e.g., by adding pseudocounts) to avoid overfitting

$$\hat{\theta}_{jc} = \frac{n_{jc} + \alpha}{n_c + \beta}$$