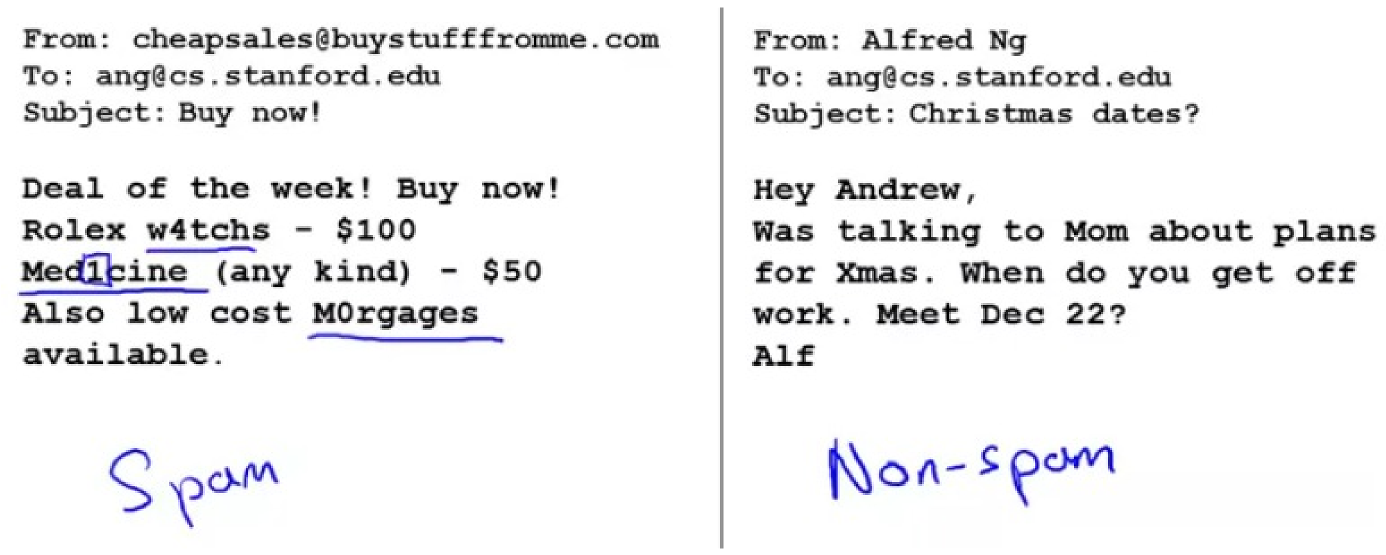
Prioritizing What to Work On: Spam classification example

- The idea of prioritizing what to work on is one of the most important skill programmers typically need to develop.
  - Concretely, it's so easy to have many ideas we want to work on, and as a result do none of them well, because doing one well is harder than doing six superficially

Building a spam classifier

Let's say we want to build a spam classifier.



```
From: cheapsales@buystufffromme.com
To: ang@cs.stanford.edu
Subject: Buy now!

Deal of the week! Buy now!
Rolex w4tchs - $100
Med1cine (any kind) - $50
Also low cost M0rgages
available.
```

Spam

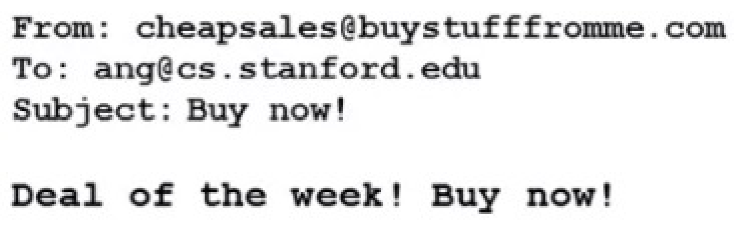
```
From: Alfred Ng
To: ang@cs.stanford.edu
Subject: Christmas dates?

Hey Andrew,
Was talking to Mom about plans
for Xmas. When do you get off
work. Meet Dec 22?
Alf
```

Non-spam

Let's say we have a labeled training set of some number of spam emails and some non-spam emails denoted with labels $y = 1$ or $y = 0$, how do we build a classifier using supervised learning to distinguish between spam and non-spam

In order to apply supervised learning, the first decision we must make is how do we want to represent x, that is the features of the email. Given the features x and the labels y in our training set, we can then train a classifier, for example using logistic regression.

- Supervised learning. x = features of email. y = spam (1) or not spam (0).
- Features x: Choose 100 words indicative of spam/not spam.
  - E.g. deal, buy, discont, andrew, now, ...
  - All these words go into one long vector



```
From: cheapsales@buystufffromme.com
To: ang@cs.stanford.edu
Subject: Buy now!

Deal of the week! Buy now!
```

- Define a feature vector x
  - Which is 0 or 1 if a word corresponding word in the reference vector is present or not
- This is a bitmap of the word content of your email

$$x = \left[\begin{array}{c} 0 \\ 1 \\ 1 \\ 0 \\ \vdots \\ 1 \\ \vdots \end{array}\right] \begin{array}{l} \text{andrew} \\ \text{buy} \\ \text{deal} \\ \text{discont} \\ \vdots \\ \text{now} \\ \vdots \end{array} \qquad x \in \mathbb{R}^{100}$$

Note: In practice, take most frequently occurring n words (10,000 to 50,000) in training set, rather than manually pick 100 words.

What's the best use of our time to improve system accuracy?

If we're building a spam classifier one question that we may face is, what's the best use of our time in order to make our spam classifier have higher accuracy, and have lower error?.

- Collect lots of data (in fact there's this tendency to think that, when the more data we have the better the algorithm will do, but we've already seen that getting lots of data will often help, but not all the time).
    - E.g. "honeypot" project (Honey pot anti-spam projects try and get fake email addresses into spammers' hands, collect loads of spam).
- Develop sophisticated features based on email routing information (from email header).
    - Spammers often try and obscure origins of email
    - Send through unusual routes
- Develop sophisticated features for message body, e.g. should "discount" and "discounts" be treated as the same word? How about "deal" and "Dealer"? Features about punctuation?
- Develop sophisticated algorithm to detect misspellings (e.g. m0rtgage, med1cine, w4tches.)

Video Question: Which of the following statements do you agree with? Check all that apply.

> For some learning applications, it is possible to imagine coming up with many different features (e.g. email body features, email routing features, etc.). But it can be hard to guess in advance which features will be the most helpful.

- For spam classification, algorithms to detect and correct deliberate misspellings will make a significant improvement in accuracy.
- Because spam classification uses very high dimensional feature vectors (e.g. n = 50,000 if the features capture the presence or absence of 50,000 different words), a significant effort to collect a massive training set will always be a good idea.

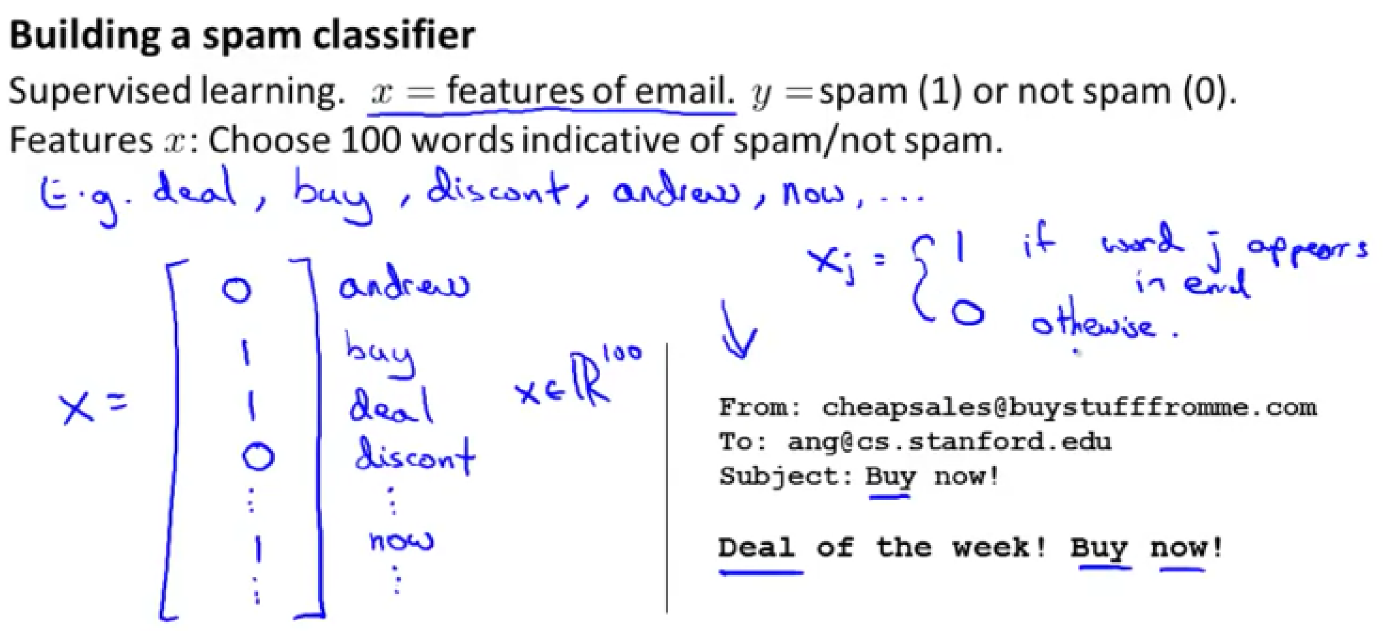
> There are often many possible ideas for how to develop a high accuracy learning system; “gut feeling” is not a recommended way to choose among the alternatives.

Summary

System Design Example:

Given a data set of emails, we could construct a vector for each email.

Each entry in this vector represents a word. The vector normally contains 10,000 to 50,000 entries gathered by finding the most frequently used words in our data set. If a word is to be found in the email, we would assign its respective entry a 1, else if it is not found, that entry would be a 0. Once we have all our x vectors ready, we train our algorithm and finally, we could use it to classify if an email is a spam or not.



Building a spam classifier

Supervised learning. $x =$ features of email. $y =$ spam (1) or not spam (0).

Features x: Choose 100 words indicative of spam/not spam.

E.g. deal, buy, discont, andrew, now, ...

$$x = \begin{array}{c|l} \begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \\ \vdots \\ 1 \\ \vdots \end{bmatrix} & \begin{array}{l} \text{andrew} \\ \text{buy} \\ \text{deal} \\ \text{discont} \\ \vdots \\ \text{now} \\ \vdots \end{array} \end{array} \qquad x \in \mathbb{R}^{100}$$

$$x_j = \begin{cases} 1 & \text{if word } j \text{ appears in email} \\ 0 & \text{otherwise.} \end{cases}$$

```
From: cheapsales@buystufffromme.com
To: ang@cs.stanford.edu
Subject: Buy now!

Deal of the week! Buy now!
```

So how could you spend your time to improve the accuracy of this classifier?

- Collect lots of data (for example "honeypot" project but doesn't always work)
- Develop sophisticated features (for example: using email header data in spam emails)
- Develop algorithms to process your input in different ways (recognizing misspellings in spam).

It is difficult to tell which of the options will be most helpful.