



Machine Learning

Machine learning system design

Prioritizing what to
work on: Spam
classification example

These videos are less mathematical but very useful!!!

Building a spam classifier

From: cheapsales@buystufffromme.com

To: ang@cs.stanford.edu

Subject: Buy now!

Deal of the week! Buy now!

Rolex w4tchs - \$100

Medicine (any kind) - \$50

Also low cost M0rgages
available.

Spam (1)

we have a training set with spam and non-spam emails

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 (0)

Building a spam classifier

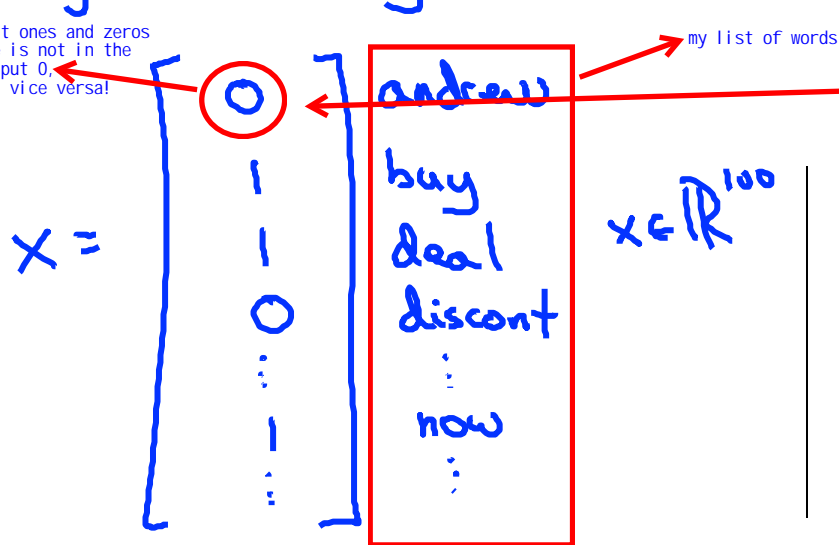
how to choose a list features for emails.

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, discount, andrew, now, ...

We just put ones and zeros
if my name is not in the
email, we put 0,
and put 1 vice versa!



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


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

Deal of the week! Buy now!

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

Building a spam classifier

How to spend your time to make it have low error?

- Collect lots of data often helps but not always!
 - E.g. “honeypot” project.
- Develop sophisticated features based on email routing information (from email header).
- 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.)  spammers actually intentionally misspell words to trick the ML algorithm



Machine Learning

Machine learning system design

Error analysis

Recommended approach

- Start with a simple algorithm that you can implement quickly. Implement it and test it on your cross-validation data.
- Plot learning curves to decide if more data, more features, etc. are likely to help.
- **Error analysis:** Manually examine the examples (in cross validation set) that your algorithm made errors on. See if you spot any systematic trend in what type of examples it is making errors on.

No more than one day!

Error Analysis

m_{CV} = 500 examples in cross validation set

Algorithm misclassifies 100 emails.

manually exam the mistake that a algorithm made and to inspire new algorithm.

Manually examine the 100 errors, and categorize them based on:

- (i) What type of email it is — *pharma, replica, steal passwords, ...*
- (ii) What cues (features) you think would have helped the algorithm classify them correctly.

Pharma: 12

Replica/fake: 4

→ Steal passwords: 53

Other: 31

→ Deliberate misspellings: 5
(m0rgage, med1cine, etc.)

→ Unusual email routing: 16

→ Unusual (spamming) punctuation: 32

The importance of numerical evaluation

Should discount/discounts/discounted/discounting be treated as the same word?

Can use “stemming” software (E.g. “Porter stemmer”) universe/university.

Error analysis may not be helpful for deciding if this is likely to improve performance. Only solution is to try it and see if it works.

Need numerical evaluation (e.g., cross validation error) of algorithm's performance with and without stemming.

Without stemming: 5% error With stemming: 3% error

Distinguish upper vs. lower case (Mom/mom): 3.2%



Machine Learning

Machine learning system design

Error metrics for
skewed classes

Cancer classification example

Train logistic regression model $h_{\theta}(x)$. ($y = 1$ if cancer, $y = 0$ otherwise)

Find that you got 1% error on test set.
(99% correct diagnoses)

Only 0.50% of patients have cancer.

→ skewed classes.

when we have a lot more examples from one class than from the other class.

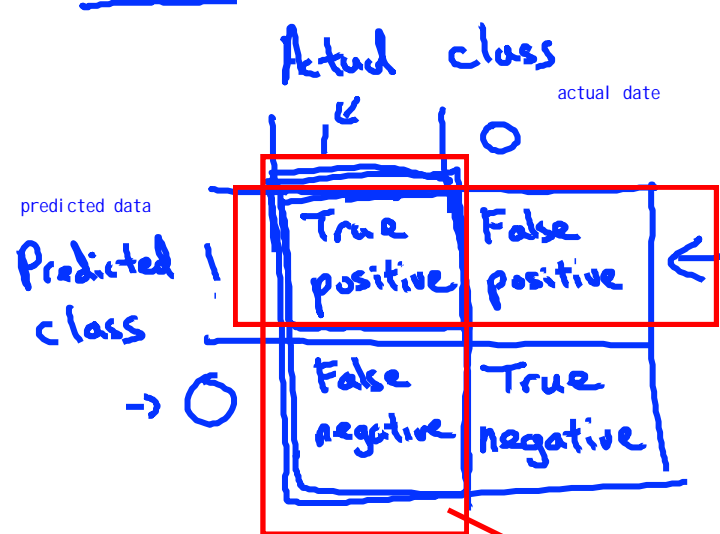
```
function y = predictCancer(x)
    → y = 0; %ignore x!
    return
```

0.5% error

→ 99.2% accuracy (0.8% error)
→ 99.5% accuracy (0.5% error)

Precision/Recall

$y = 1$ in presence of rare class that we want to detect



→ Precision

(Of all patients where we predicted $y = 1$, what fraction actually has cancer?)

number of people have cancer and we predicted this.

high precision is good!

$$\frac{\text{True positives}}{\text{\#predicted positive}} = \frac{\text{True positive}}{\text{True pos} + \text{False pos}}$$

predicted to have cancer

→ Recall

(Of all patients that actually have cancer, what fraction did we correctly detect as having cancer?)

$$\frac{\text{True positives}}{\text{\#actual positives}} = \frac{\text{True positives}}{\text{True pos} + \text{False neg}}$$

High recall is good!



Machine Learning

Machine learning system design

Trading off precision
and recall

Trading off precision and recall

→ Logistic regression: $0 \leq h_{\theta}(x) \leq 1$

Predict 1 if $h_{\theta}(x) \geq 0.5$ ~~0.7~~ ~~0.9~~ 0.3 ←

Predict 0 if $h_{\theta}(x) < 0.5$ ~~0.7~~ ~~0.9~~ 0.3

→ Suppose we want to predict $y = 1$ (cancer) only if **very confident**.

setting a higher cutoff value!

→ Higher precision, lower recall

small false positive, large false negative

→ Suppose we want to **avoid missing too many cases of cancer** (avoid false negatives).

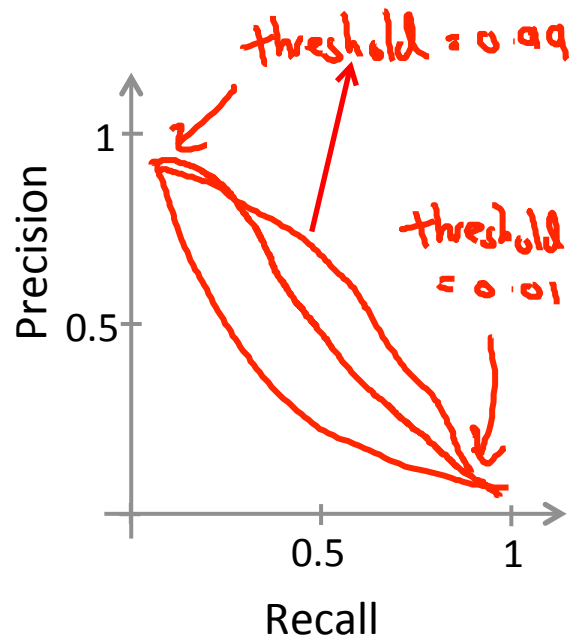
→ Higher recall, lower precision.

small false negative, large false positive

More generally: Predict 1 if $h_{\theta}(x) \geq \text{threshold}$ ←

→ precision = $\frac{\text{true positives}}{\text{no. of predicted positive}}$

→ recall = $\frac{\text{true positives}}{\text{no. of actual positive}}$



F₁ Score (F score)

How to compare precision/recall numbers?

	Precision(P)	Recall (R)
→ Algorithm 1	<u>0.5</u>	<u>0.4</u>
→ Algorithm 2	<u>0.7</u>	<u>0.1</u>
Algorithm 3	<u>0.02</u>	1.0

Average: ~~$\frac{P+R}{2}$~~ not good!

$$F_1 \text{ Score: } 2 \frac{PR}{P+R}$$

$$P=0 \text{ or } R=0 \Rightarrow F\text{-score} = 0$$
$$P=1 \text{ and } R=1 \Rightarrow F\text{-score} = 1$$

This is usually people in machine learning use!

which algorithm is better?



can eliminate the extreme cases

Predict $y=1$ all the time



Machine Learning

Machine learning system design

Data for machine learning

Designing a high accuracy learning system

E.g. Classify between confusable words.

{to, two, too}, {then, than}

→ For breakfast I ate two eggs.

Algorithms

- - Perceptron (Logistic regression)
- - Winnow
- - Memory-based
- - Naïve Bayes



“It’s not who has the best algorithm that wins.

It’s who has the most data.”

Large data rationale

→ Assume feature $x \in \mathbb{R}^{n+1}$ has sufficient information to predict y accurately. ↗

Example: For breakfast I ate two eggs. ↗

Counterexample: Predict housing price from only size (feet²) and no other features. ↗

Useful test: Given the input x , can a human expert confidently predict y ?

Large data rationale

→ Use a learning algorithm with many parameters (e.g. logistic regression/linear regression with many features; neural network with many hidden units). low bias algorithms. ←

→ $J_{\text{train}}(\theta)$ will be small.

Use a very large training set (unlikely to overfit) low variance ←

→ $J_{\text{train}}(\theta) \approx J_{\text{test}}(\theta)$

→ $J_{\text{test}}(\theta)$ will be small