Machine Learning System Design

Andrew Ng

Machine learning system design

Prioritizing what to work on: Spam classification example

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
Med1cine (any kind) - $50
Also low cost M0rgages
available.
```

```
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
```

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

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

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

Deal of the week! Buy now!
```

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
    - E.g. “honey pot” 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.)

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

Error analysis

$m_{CV} =$ 500 examples in cross validation set

Algorithm misclassifies 100 emails.

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

(i) What type of email it is

(ii) What cues (features) you think would have helped the algorithm classify them correctly

Pharma: 12
Replica/fake: 4
Steal passwords: 53
Other: 31

$\longrightarrow$

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 (wrong)

- 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

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.

```
def predictCancer(x):
    y = 0    # ignore x!
    return y
```

0.5% error is better than 1% error?

Precision/Recall

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

		Actual class	
		1	0
Predicted class	1	True positive	False Positive
	0	False negative	True negative

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

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

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

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

Machine learning system design

Trading off precision and recall

Trading off precision and recall

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

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

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

Predict 1 if $h_\theta(x) \geq 0.5$

Predict 0 if $h_\theta(x) < 0.5$

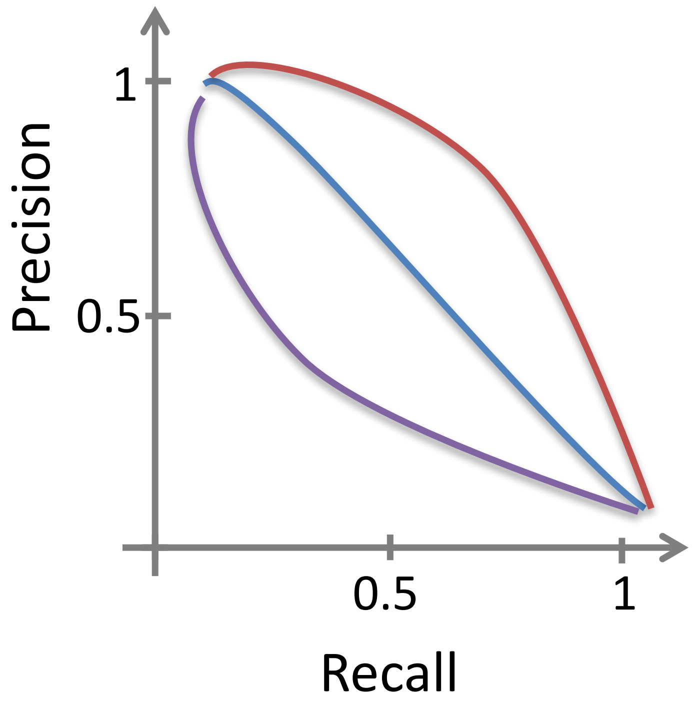
Suppose we want to predict $y = 1$ (cancer) only if very confident (threshold = 0.6, 0.7, ...)

Higher precision, lower recall

Suppose we want to avoid missing too many cases of cancer (avoid false negatives) (threshold = 0.4, 0.3, ...)

Higher recall, lower precision

More generally: Predict 1 if $h_\theta(x) \geq$ threshold



F_1 Score (F score)

How to compare precision/recall numbers?

	Precision(P)	**Recall (R)**	**Average**	**F_1 Score**
Algorithm 1	0.5	0.4	0.45	0.444
Algorithm 2	0.7	0.1	0.4	0.175
Algorithm 3	0.02	1.0	0.51	0.0392

Average: $\frac{P+R}{2}$

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

(harmonic mean)

Machine learning system design

Data for machine learning

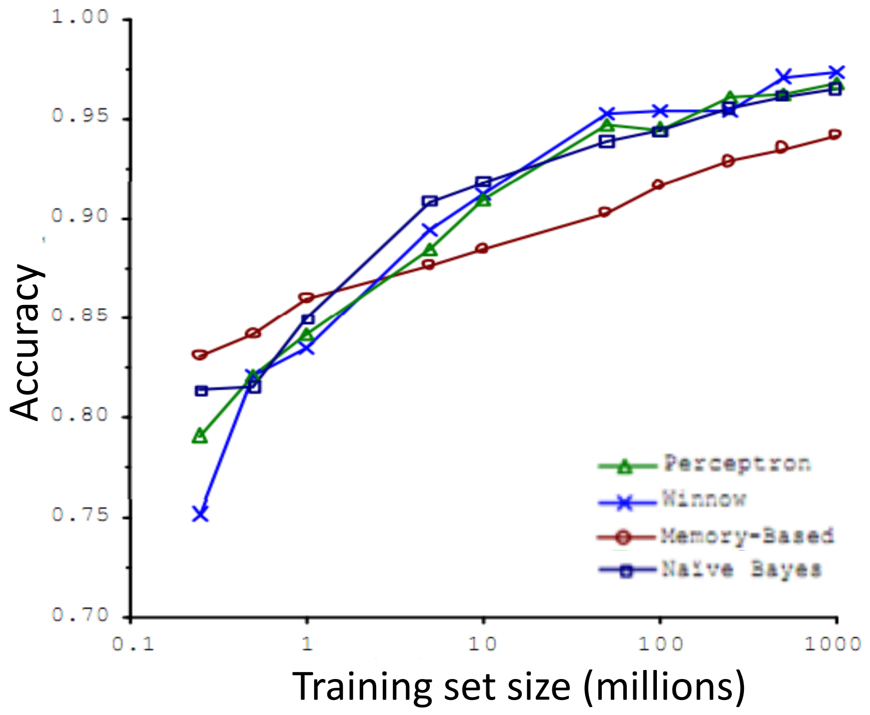
Designing a high accuracy learning system

E.g. Classify between confusable words
{to, two, too}, {then, than}

For breakfast I ate ______ 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."

[Banko and Brill, 2001]

Large data rationale

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

Example: For breakfast I ate ______ eggs

Counterexample: Predict housing price from only size (feet^2) 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) (unlikely to underfit)

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

- Use a very large training set (unlikely to overfit)

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

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