Week 6 - part 2

Machine Learning System design: Prioritizing what to work on: Spam classification example

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

Features x: Choose 100 words indicative of spam / not spam.
e.g deal, buy, discount... = spam
Andrew, now... = not spam

$$x = \begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \\ \vdots \\ \vdots \end{bmatrix} \begin{matrix} \text{Andrew} \\ \text{Buy} \\ \text{deal} \\ \text{discount} \\ \vdots \\ \text{now} \end{matrix} \qquad x \in \mathbb{R}^{100}$$

example email
From:
To
subject: buy now!
Deal of the week!

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

Note: In practise, take most frequency occurring n words (10000 to 50000) 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 "honeypot" project.
- Develop sophisticated features based on email routing information. (from email header)
- develop sophisticated features for message body
  e.g should "discounts" and "discount" be treated as the same word.
- develop sophisticated algorithms to detect misspellings

Week 6

Machine learning: Error Analysis

Recommended approach

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

→ ways of avoiding "premature - optimization" //
→ lets evidence decide on where to spend our time rather than use gut feeling.

---

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 — pharma, replica, steal passwords...
(ii) what cues (features) you think would have helped the algorithm classify 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

strong signal, maybe worth the time working on this problem.

Week 6

Machine learning: error analysis

The importance of numerical evaluation

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

Can use "stemming" software (eg. "porterstemmer")

universe/university - potential error here.

error analysis may not be helpful for deciding if this is likely to improve performance. Only solution is to try 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% error

if we use this as another parameter, by evaluating the error that the algorithm produces, we can see if the choice of the feature was correct.

Recommend implementing a "quick & dirty" implementation of your learning algorithm.

Week 6

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

I.e hardly ever 1 (the patient has cancer) ∴ data is "skewed"

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

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

$y=0$
recall $=0$

not a good classifier

Computing precision and recall will give us a good sense of how well our classifier is doing.

Precision

(of all the patients that we predicted $y=1$, what fraction actually has cancer?)

divided by

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

Recall

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

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

high recall is a good thing

Week 6

Machine learning: trading off precision and recall

Trading off precision and recall

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

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

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

Predict 1 if $h_\theta(x) \ge \cancel{0.5}$ | 0.7 | 0.9 | 0.3

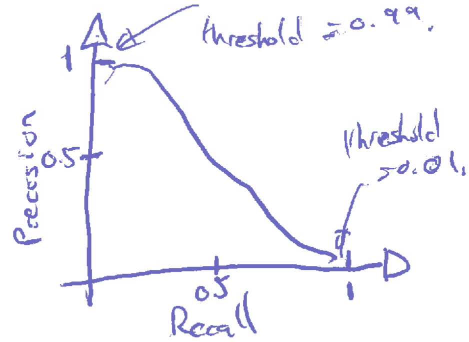
Predict 0 if $h_\theta(x) < \cancel{0.5}$ | 0.7 | 0.9 | 0.3

Suppose we want to predict $y=1$ (cancer)
only if very confident. → one way of doing this
→ higher precision, lower recall?

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

higher recall, lower precision

more generally: Predict 1 if $h_\theta(x) \ge$ 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

Predict $y=1$ all the time — not a good algorithm get high average ∴

Average: $\frac{P+R}{2}$ ← this is one way

F_1 score: $2\frac{PR}{P+R}$ → $P=0$ or $R=0$
⇒ Fscore $= 0$
$P=1$ & $R=1$ ⇒ Fscore $=1$

→ averaging is not a good method of evaluating the algorithm's overall performance.

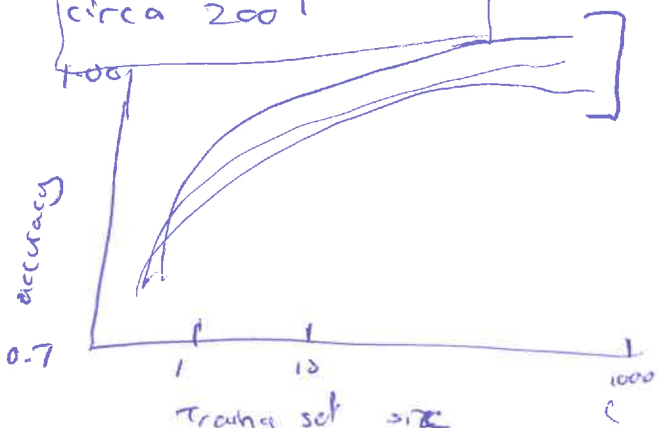
Week 6

Machine learning system design: data for machine learning

Design 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
- Naive based.

Banko e Brill study
circa 2001



"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 ~~to~~ ~~too~~

Counter example: Predict housing price from only size ($feet^2$) and no other features. → this is an example where it would be very difficult to predict the price accurately.

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

- So ask a human expert if the chosen parameter is "good enough" to make the prediction.

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.

→ $J_{train}(\theta)$ will be small.

Use a very large training set (unlikely to overfit) ← because we have such a massive training set.

low variance

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

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