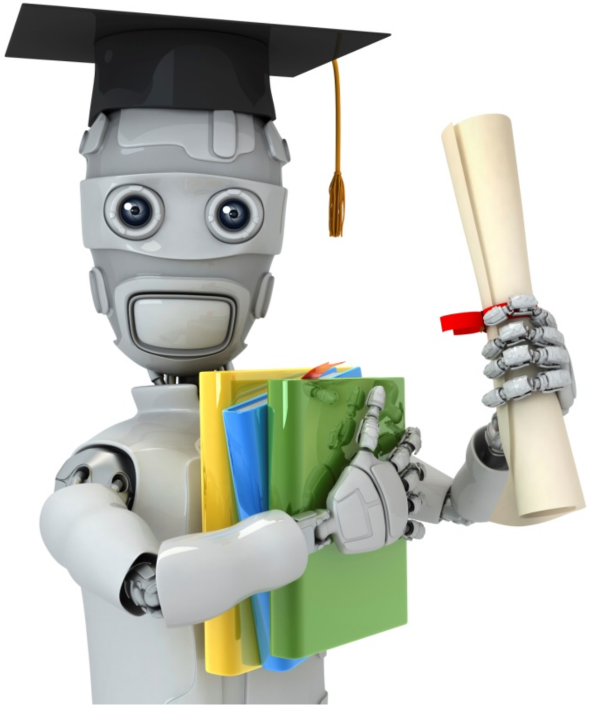


Machine Learning

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.
```

Spam (1)

```
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

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, ...

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

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

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

Deal of the week! Buy now!
```

not gonna count how many times a word shows up

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

Andrew Ng

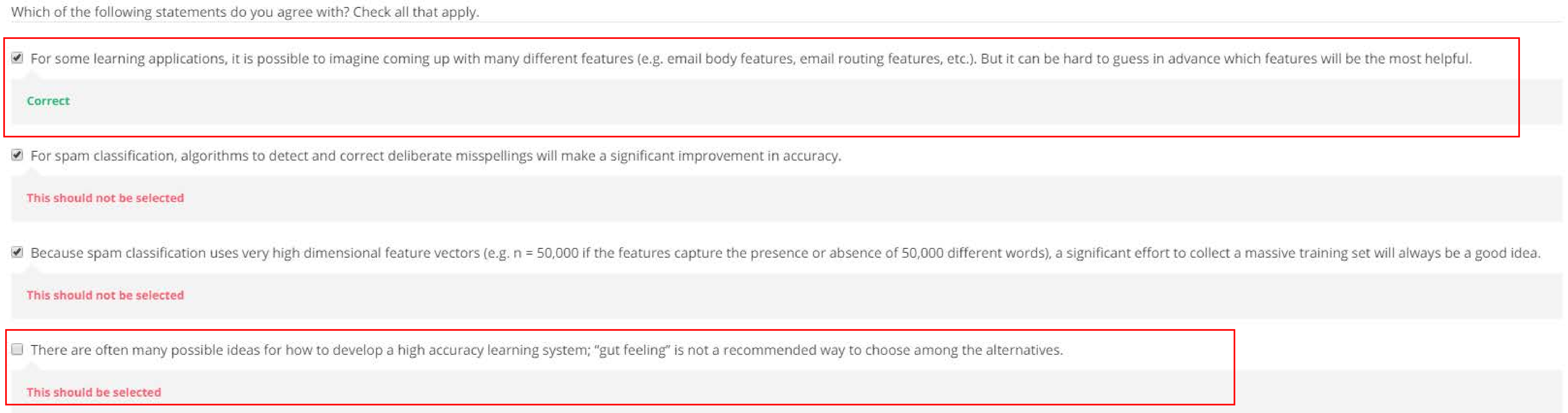
if you're building a spam classifier one question that you may face is, what's the best use of your time in order to make your spam classifier have higher accuracy, you have lower error.

Building a spam classifier

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

- Collect lots of data
  that getting lots of data will often help, but not all the time.
  - 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.)

I would actually have a very hard time telling you of these four options which is the best use of your time. what most people do is instead of trying to list out the options of things you might try, what far too many people do is wake up one morning and have a weird gut feeling that, "Oh let's have a huge honeypot project to go and collect tons more data" and for whatever strange reason just sort of wake up one morning and randomly fixate on one thing and just work on that for six months.
But I think we can do better. And in particular what I'd like to do is tell you about the concept of "error analysis"



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.

Correct

[✓] For spam classification, algorithms to detect and correct deliberate misspellings will make a significant improvement in accuracy.

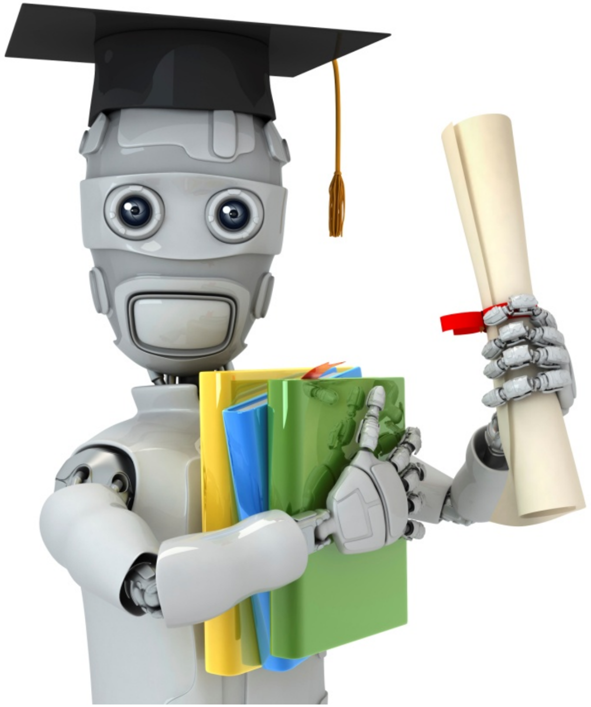
This should not be selected

[✓] 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.

This should not be selected

[ ] 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.

This should be selected



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.

And it's often by implementing even a simple implementation. And by plotting learning curves, that helps you make these decisions. So if you like you can to think of this as a way of avoiding "premature optimization" in computer programming. And this idea that says we should let evidence guide our decisions on where to spend our time rather than use gut feeling, which is often wrong. I

I will often look at my cross validation set and manually look at the emails that my algorithm is making errors on. So look at the spam e-mails and non-spam e-mails that the algorithm is misclassifying and see if you can spot any systematic patterns in what type of examples it is misclassifying. And often, by doing that, this is the process that will inspire you to design new features. Or they'll tell you what are the current shortcomings of the system. And give you the inspiration you need to come up with improvements to it.

Error Analysis

What we really want to do is figure out what are the most difficult examples for an algorithm to classify, very often for different algorithms, for different learning algorithms they'll often find similar categories of examples difficult. And by having a quick and dirty implementation, that's often a quick way to let you identify some errors and quickly identify what are the hard examples. So that you can focus your effort on those.

$m_{CV} =$ 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 them correctly.

And it's by counting up the number of emails in these different categories that you might discover that the algorithm is doing poorly on emails trying to steal passwords. And that may suggest that it might be worth your effort to look more carefully at that type of email

Pharma: 12

Replica/fake: 4

→ Steal passwords: 53

Other: 31

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

→ Unusual email routing: 16

→ Unusual (spamming) punctuation: 32

⋮

In natural language processing, the way that this is done is using stemming software. And if you ever want to do this yourself, search for the porter stemmer, and that would be one reasonable piece of software for doing this sort of stemming

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

using a stemming software that basically looks at the first few alphabets of a word,it can help, but it can hurt, because the software may mistake the words universe and university as being the same thing.

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%

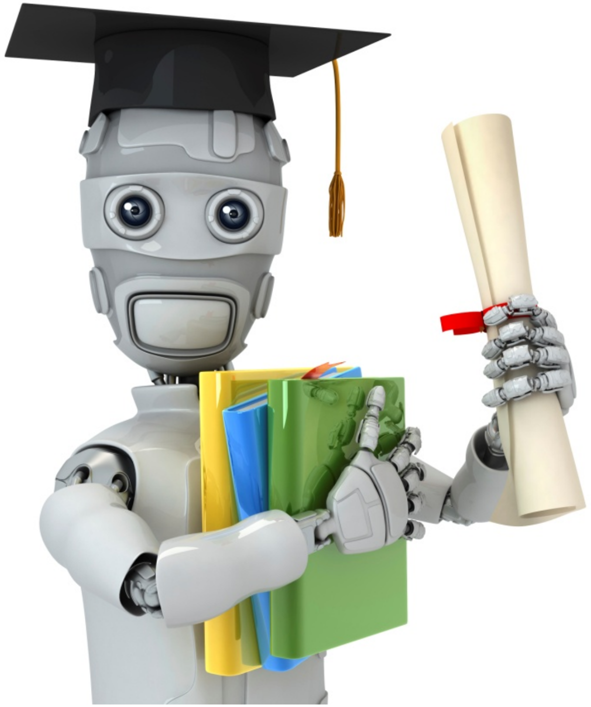
Do you distinguish upper and lower case or not? But by having a single real number evaluation metric, you can then just look and see, oh,did the error go up or did it go down? And you can use that to much more rapidly try out new ideas and almost right away tell if your new idea has improved or worsened the performance of the learning algorithm.

Why is the recommended approach to perform error analysis using the cross validation data used to compute $J_{\text{CV}}(\theta)$ rather than the test data used to compute $J_{\text{test}}(\theta)$?

- The cross validation data set is usually large.
- This process will give a lower error on the test set.
- [✓ If we develop new features by examining the test set, then we may end up choosing features that work well specifically for the test set, so Jtest(θ) is no longer a good estimate of how well we generalize to new examples.]

  Correct

- Doing so is less likely to lead to choosing an excessive number of features.



Machine Learning

Machine learning system design

Error metrics for skewed classes

if you have skewed classes it becomes harder to use classification accuracy, because you can get very high classification accuracies or very low errors, and it's not clear if doing so is improving the quality of classifier.

Cancer classification example

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

Find that you got $\boxed{\underline{1\%}\text{ error}}$ on test set.

($\underline{99\%}$ correct diagnoses)

But now,we find out that only 0.5 percent of patients in our "training test sets" actually have cancer. So only half a percent of the patients that come through our screening process have cancer. In this case, the 1% error no longer looks so impressive.

Only $\underline{0.50\%}$ of patients have cancer. We just have a lot more of examples from one class than from the other class.

$\rightarrow$ skewed classes.

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

$\rightarrow$ $\underline{0.5\% \text{ error}}$

$\rightarrow$ 99.2% accuracy (0.8% error)

$\rightarrow$ 99.5% accuracy (0.5% error)

By going from 99.2% accuracy to 99.5% accuracy.did we just do something useful or did we just replace our code with something that just predicts y equals zero more often?if you have skewed classes it becomes much harder to use just classification accuracy.

Precision/Recall

usually we use the convention y=1, in the presence of the more rare class.

So if we are trying to detect rare conditions such as cancer,precision and recall are defined setting y equals 1, to be the presence of that rare class that we're trying to detect.

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

Actual class

Predicted class	1	0
1	True positive	False positive
→ 0	False negative	True negative

→ **Precision**

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

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

high precision would be good. high precision means that of that group of patients most of them we had actually made accurate predictions on them and they do have cancer.

→ **Recall**

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

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

$y = 0$

recall $= 0$

if y=0 all the time, then recall = 0

And we find, that even if we have skewed classes, if a classifier is getting high precision and high recall, then we are actually confident that the algorithm has to be doing well, even if we have very skewed classes.

https://www.quora.com/What-is-the-best-way-to-understand-the-terms-precision-and-recall

Imagine that, your girlfriend gave you a birthday surprise every year in last 10years.
However, one day, your girlfriend asks you: ‘Sweetie, do you remember all birthday surprises from me?’
This simple question makes your life in danger. To extend your life, you need to recall all 10 surprising events from your memory.
So, recall is the ratio of a number of events you can correctly recall to a number of all correct events.
If you can recall all 10 events correctly, then, your recall ratio is 100%. If you can recall 7 events correctly, your recall ratio is 70%.
However, you might be wrong in some answers.
For example, you answers 15 times, 10 events are correct and 5 events are wrong. This means you can recall all events but it’s not so precise.
So, precision is the ratio of a number of events you can correctly recall to a number all events you recall (mix of correct and wrong recalls). In other words, it is how precise of your recall.
From the previous example (10 real events, 15 answers: 10 correct answers, 5 wrong answers), you get 100% recall but your precision is only 66.67% (10 / 15).
If a machine learning algorithm is good at recall, it doesn’t mean that algorithm is good at precision. That’s why we also need F1 score which is the (harmonic) mean of recall and precision to evaluate an algorithm.

1.

Learning System Design

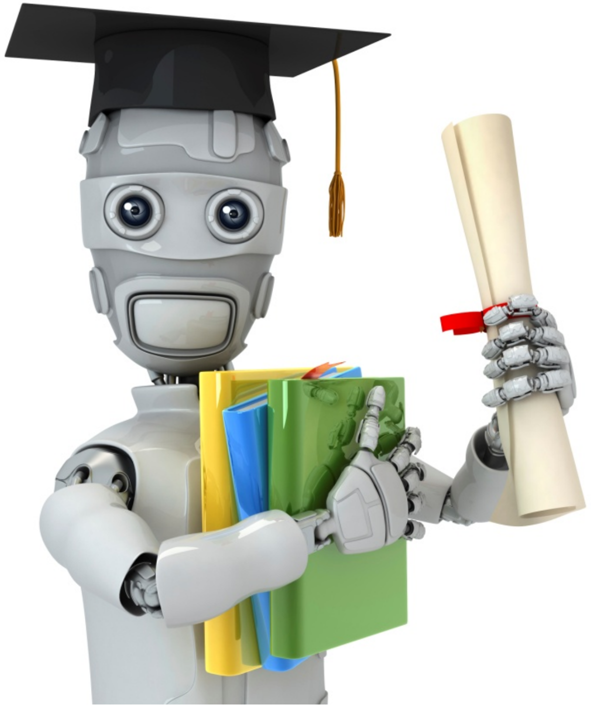
You are working on a spam classification system using regularized logistic regression. "Spam" is a positive class (y = 1) and "not spam" is the negative class (y = 0). You have trained your classifier and there are m = 1000 examples in the cross-validation set. The chart of predicted class vs. actual class is:

85+890+15+10=1000

	Actual Class: 1	**Actual Class: 0**
Predicted Class: 1	85 true positive	890 false positive
Predicted Class: 0	15 false negative	10 true negative

For reference:

- Accuracy = (true positives + true negatives) / (total examples)
- Precision = (true positives) / (true positives + false positives)
- Recall = (true positives) / (true positives + false negatives)
- F_1 score = (2 * precision * recall) / (precision + recall)



Machine Learning

Machine learning system design

Trading off precision and recall

Trading off precision and recall

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

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

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

  Predict 1 if $h_\theta(x) \ge$ ~~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.

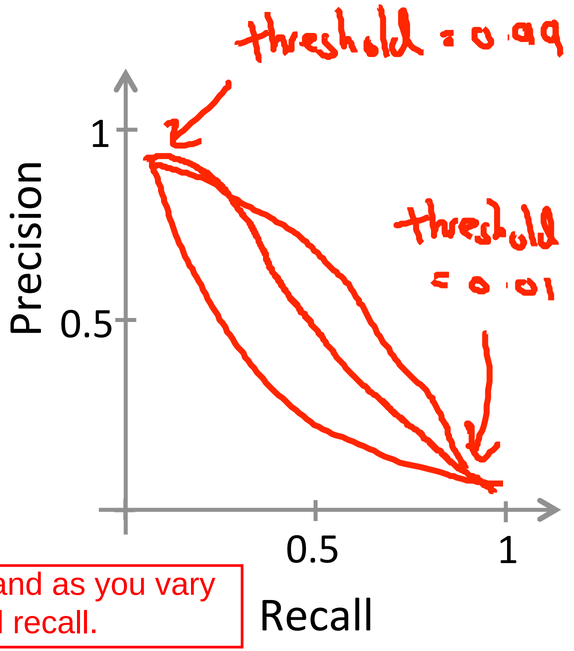
  → Higher precision, lower recall

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

  → Higher recall, lower precision.

in general, for most classifiers there is going to be a trade off between precision and recall, and as you vary the value of this threshold, you can actually plot out some curve that trades off precision and recall.

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



Andrew Ng

if we have a few different algorithms or a few different ideas for algorithms, how do we compare different precision recall numbers?

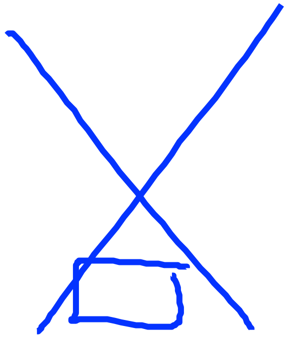
F_1 Score (F score)

One of the things we talked about earlier is the importance of a single real number evaluation metric. And that is the idea of having "a number" that just tells you how well is your classifier doing.

How to compare precision/recall numbers?

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

Predict y=1 all the time



Average: $\cancel{\dfrac{P+R}{2}}$

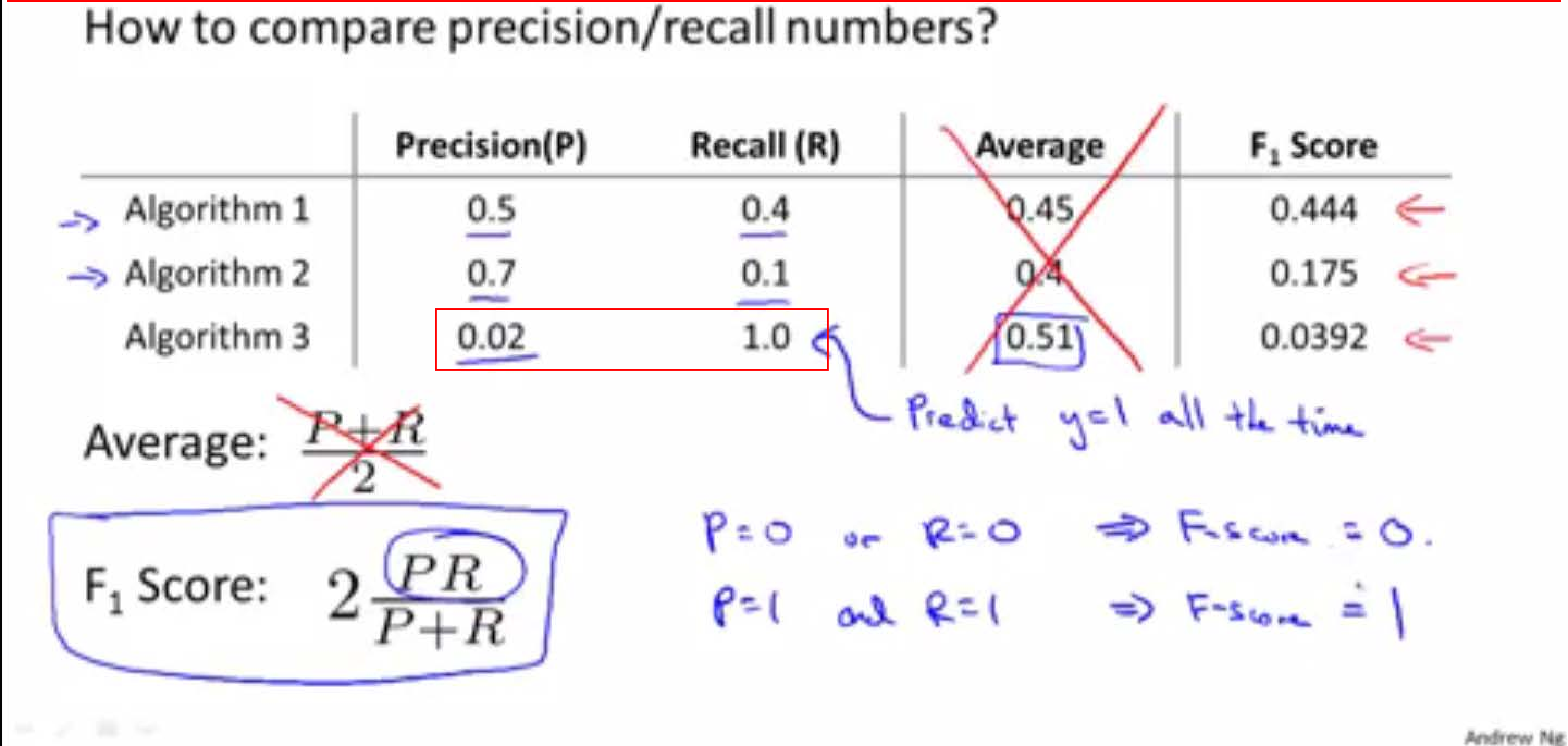
$$\boxed{F_1 \text{ Score: } \quad 2\frac{PR}{P+R}}$$

P=0 or R=0 ⇒ F-score = 0

P=1 and R=1 ⇒ F-score = 1

Andrew Ng

if we have a classifier that predicts y=1 all the time, then if you do that you can get a very high recall, but you end up with a very low value of precision. Conversely, if you have a classifier that predicts y equals zero, almost all the time, that is that it predicts y=1 very sparingly, this corresponds to setting a very high threshold using the notation of the previous y. Then you can actually end up with a very high precision with a very low recall. So, the two extremes of either a very high threshold or a very low threshold, neither of that will give a particularly good classifier.



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

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

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

$P=0$ or $R=0$ $\Rightarrow$ F-score $=0$.

$P=1$ and $R=1$ $\Rightarrow$ F-score $\doteq 1$

Andrew Ng

You have trained a logistic regression classifier and plan to make predictions according to:

- Predict $y = 1$ if $h_\theta(x) \geq \text{threshold}$
- Predict $y = 0$ if $h_\theta(x) < \text{threshold}$

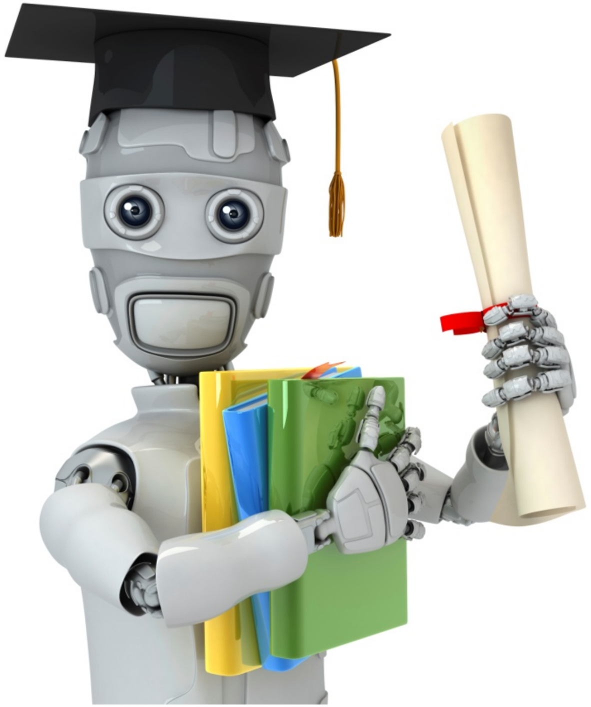
For different values of the threshold parameter, you get different values of precision (P) and recall (R). Which of of the following would be a reasonable way to pick the value to use for the threshold?

○ Measure precision (P) and recall (R) on the **test set** and choose the value of threshold which maximizes $\frac{P+R}{2}$

○ Measure precision (P) and recall (R) on the **test set** and choose the value of threshold which maximizes $2\frac{PR}{P+R}$

○ Measure precision (P) and recall (R) on the **cross validation set** and choose the value of threshold which maximizes $\frac{P+R}{2}$

[✓ Measure precision (P) and recall (R) on the **cross validation set** and choose the value of threshold which maximizes $2\frac{PR}{P+R}$]



Machine Learning

Machine learning system design

Data for machine learning

Designing a high accuracy learning system

E.g. Classify between confusable words.

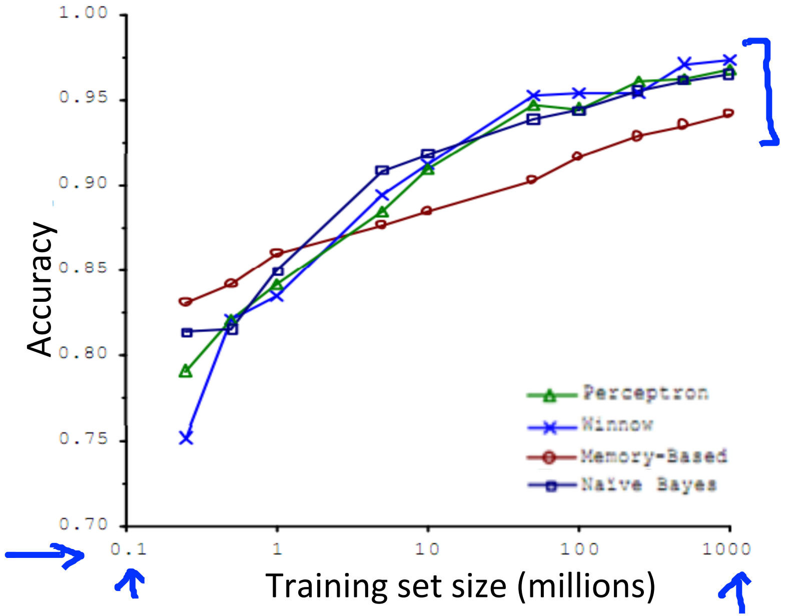
{to, two, too}, {then, than}

For breakfast I ate two ____ eggs.

should use: to, two. or too

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]

a set of assumptions under which having a massive training set we think will be able to help.

Large data rationale

Training on a lot of data is likely to give good performance when two of the following conditions hold true.

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

Example: For breakfast I ate two ______ eggs.

~~to~~, ~~too~~

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?

Suppose the features have enough information to predict the value of y. And let's suppose we use a learning algorithm with a large number of parameters so maybe logistic regression or linear regression with a large number of features. Or one thing that I often do actually is using neural network with many hidden units.....these are all powerful learning algorithms with a lot of parameters that can fit very complex functions.

$\rightarrow$ 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. $\leftarrow$

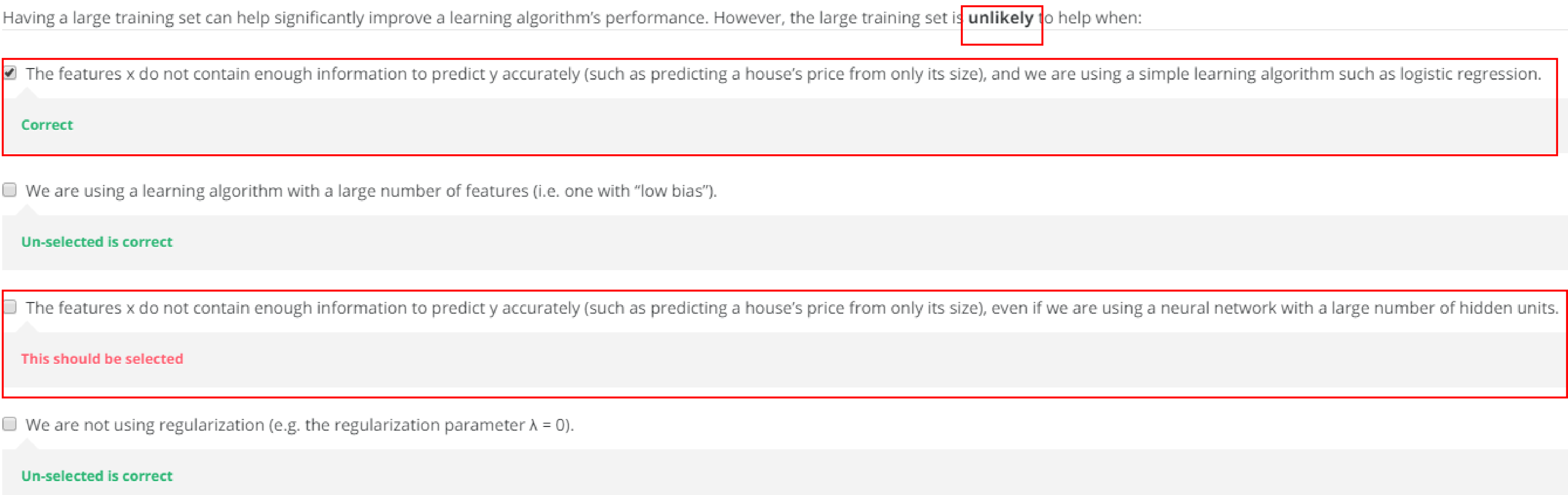
$\rightarrow J_{train}(\theta)$ will be small.

Use a very large training set (unlikely to overfit) low variance $\swarrow$

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

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

Another way to think about this is that in order to have a high performance learning algorithm we want it not to have (high bias and high variance), So the bias problem we're going to address by making sure we have a learning algorithm with many parameters (gives us a low bias) and by using a very large training set (don't have a variance problem).



Having a large training set can help significantly improve a learning algorithm's performance. However, the large training set is **unlikely** to help when:

[✓] The features x do not contain enough information to predict y accurately (such as predicting a house's price from only its size), and we are using a simple learning algorithm such as logistic regression.

Correct

☐ We are using a learning algorithm with a large number of features (i.e. one with "low bias").

Un-selected is correct

☐ The features x do not contain enough information to predict y accurately (such as predicting a house's price from only its size), even if we are using a neural network with a large number of hidden units.

This should be selected

☐ We are not using regularization (e.g. the regularization parameter $\lambda = 0$).

Un-selected is correct