Week 6

- ## Evaluating a Learning Algorithm:

a. Deciding What to Try Next:

Debugging a learning algorithm:

Suppose you have implemented regularized linear regression to predict housing prices.

However, when you test your hypothesis on a new set of house, you find it makes unacceptably large errors in its predictions. What should try next?

- Get more training examples(sometimes it doesn't work)
- Try smaller sets of features
- Try getting additional features
- Try adding polynomial features(x_1^2, x_2^2, x_1x_2, etc)
- Try decreasing λ
- Try increasing λ

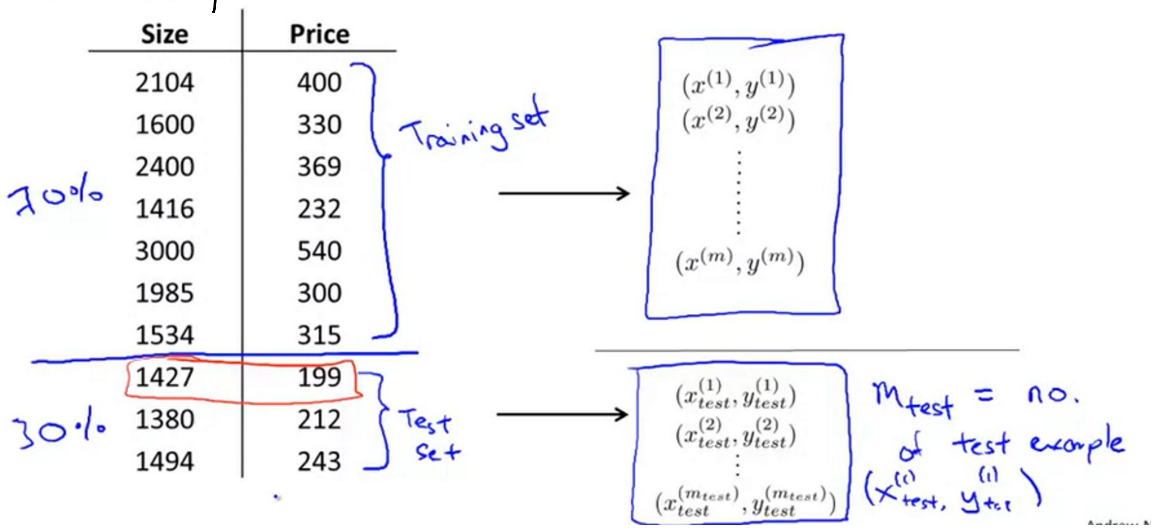
Machine learning diagnostic:

Diagnostic: A test that you can run to gain insight what is/isn't working with a learning algorithm, and gain guidance as to how best to improve its performance.

b. Evaluating a Hypothesis:

How to tell a hypothesis is overfitting?

Dataset: split datasets



Training/testing procedure for linear regression

- Learn parameter θ from training data (minimizing training error $J(\theta)$) 70%

- Compute test set error:

linear → $J_{test}(\theta) = \frac{1}{2m_{test}} \sum_{i=1}^{m_{test}} \left(h_\theta(x_{test}^{(i)}) - y_{test}^{(i)} \right)^2$

logistic → $J_{test}(\theta) = -\frac{1}{m_{test}} \sum_{i=1}^{m_{test}} y_{test}^{(i)} \log h_\theta(x_{test}^{(i)}) + (1 - y_{test}^{(i)}) \log h_\theta(x_{test}^{(i)})$.

Sometimes using misclassification error.

$$err(h_\theta(x), y) = \begin{cases} 1, & \text{if } h_\theta(x) \geq 0.5,\ y = 0 \text{ or if } h_\theta(x) < 0.5,\ y = 1 \\ 0 & \text{otherwise.} \end{cases}$$ } error.

Test error $= \frac{1}{m_{test}} \sum_{i=1}^{m_{test}} err(h_\theta(x_{test}^{(i)}), y_{test}^{(i)})$.

c. Model Selection and Train/Validation/Test Sets:
one more parameter d denote degree of polynomial.

Model selection

→ d = degree of polynomial

d=1 1. $h_\theta(x) = \theta_0 + \theta_1 x$ → $\theta^{(1)}$ → $J_{test}(\theta^{(1)})$

d=2 2. $h_\theta(x) = \theta_0 + \theta_1 x + \theta_2 x^2$ → $\theta^{(2)}$ → $J_{test}(\theta^{(2)})$

d=3 3. $h_\theta(x) = \theta_0 + \theta_1 x + \cdots + \theta_3 x^3$ → $\theta^{(3)}$ → $J_{test}(\theta^{(3)})$

⋮ ⋮ ⋮

d=10 10. $h_\theta(x) = \theta_0 + \theta_1 x + \cdots + \theta_{10} x^{10}$ → $\theta^{(10)}$ → $J_{test}(\theta^{(10)})$

Choose $\theta_0 + \ldots \theta_5 x^5$ ←

How well does the model generalize? Report test set error $J_{test}(\theta^{(5)})$.

$\theta^{(5)}$ We may choose an overly optimistic estimate of generalization error.

Problem: $J_{test}(\theta^{(5)})$ is likely to be an optimistic estimate of generalization error. I.e. our extra parameter (d = degree of polynomial) is fit to test set.

We choose the best J_{test} on test set, it by test set.

Evaluating your hypothesis

Dataset:

	Size	Price	
60%	2104	400	Training set
	1600	330	
	2400	369	
	1416	232	
	3000	540	
	1985	300	
20%	1534	315	Cross validation set (CV)
	1427	199	
20%	1380	212	test set
	1494	243	

split into 3 pieces.

$(x^{(1)}, y^{(1)})$
$(x^{(2)}, y^{(2)})$
$\vdots$
$(x^{(m)}, y^{(m)})$

$(x_{cv}^{(1)}, y_{cv}^{(1)})$
$(x_{cv}^{(2)}, y_{cv}^{(2)})$
$\vdots$
$(x_{cv}^{(m_{cv})}, y_{cv}^{(m_{cv})})$

m_{cv} = number of CV examples

$\rightarrow (x_{cv}^{(i)}, y_{cv}^{(i)})$.

$(x_{test}^{(1)}, y_{test}^{(1)})$
$(x_{test}^{(2)}, y_{test}^{(2)})$
$\vdots$
$(x_{test}^{(m_{test})}, y_{test}^{(m_{test})})$

training Set error: $J_{train}(\theta) = \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2$

cross validation error: $J_{cv}(\theta) = \frac{1}{2m_{cv}} \sum_{i=1}^{m_{cv}} (h_\theta(x_{cv}^{(i)}) - y_{cv}^{(i)})^2$

test error: $J_{test}(\theta) = \frac{1}{2m_{test}} \sum_{i=1}^{m_{test}} (h_\theta(x_{test}^{(i)}) - y_{test}^{(i)})^2$.

When we face a model selection problem:

Model selection

find min $J_{cv}(\theta^{(i)})$.

$d=1$ 1. $h_\theta(x) = \theta_0 + \theta_1 x$ $\rightarrow$ $\min_\theta J(\theta) \rightarrow \theta^{(1)} \rightarrow J_{cv}(\theta^{(1)})$

$d=2$ 2. $h_\theta(x) = \theta_0 + \theta_1 x + \theta_2 x^2$ $\rightarrow \theta^{(2)} \rightarrow J_{cv}(\theta^{(2)})$

$d=3$ 3. $h_\theta(x) = \theta_0 + \theta_1 x + \cdots + \theta_3 x^3$ $\rightarrow \theta^{(3)}$

$\vdots$ $\vdots$

$d=10$ 10. $h_\theta(x) = \theta_0 + \theta_1 x + \cdots + \theta_{10} x^{10}$ $\rightarrow \theta^{(10)} \rightarrow J_{cv}(\theta^{(10)})$

$J_{cv}(\theta^{(4)})$

$d=4$

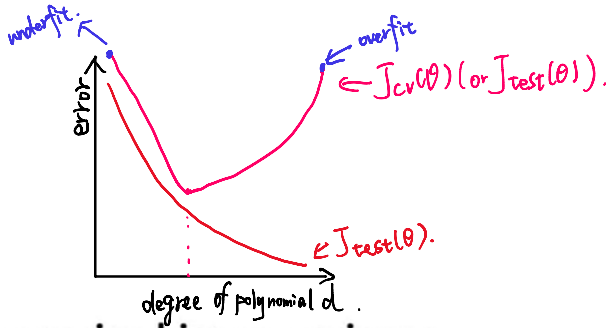
Pick $\theta_0 + \theta_1 x_1 + \cdots + \theta_4 x^4$ ⇐

Estimate generalization error for test set $J_{test}(\theta^{(4)})$ ⇐

- Bias vs. Variance:
  a. Diagnosing Bias vs. Variance:

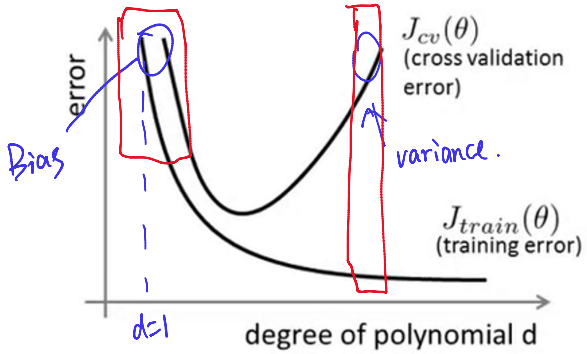
     High bias or high variance ————>underfitting or overfitting

     It's important to figure out which of these two problems is bias or variance or a bit of both. It would give a strong indicate for whether the useful and promising ways to try to improve your algorithm.



Diagnosing bias vs. variance

Suppose your learning algorithm is performing less well than you were hoping. ($J_{cv}(\theta)$ or $J_{test}(\theta)$ is high.) Is it a bias problem or a variance problem?



Bias (underfit):
$J_{train}(\theta)$ will be high.
$J_{cv}(\theta) \approx J_{train}(\theta)$

Variance (overfit):
$J_{train}(\theta)$ will be low
$J_{cv}(\theta) \gg J_{train}(\theta)$.

b. Regularizaiton and Bias/Variance:

Choose the regularization parameter λ

first:

$$J_{train}(\theta) = \frac{1}{2m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2$$

$$J_{cv}(\theta) = \frac{1}{2m_{cv}}\sum_{i=1}^{m_{cv}}(h_\theta(x_{cv}^{(i)}) - y^{(i)})^2$$

$$J_{test}(\theta) = \frac{1}{2m_{test}}\sum_{i=1}^{m_{test}}(h_\theta(x_{test}^{(i)}) - y^{(i)})^2$$

} no regularization.

second:

try different λ value.

Model: $h_\theta(x) = \theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4$

$$J(\theta) = \frac{1}{2m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2 + \frac{\lambda}{2m}\sum_{j=1}^{m}\theta_j^2$$

1. Try $\lambda = 0$ → $\min_\theta J(\theta)$ → $\theta^{(1)}$ → $J_{cv}(\theta^{(1)})$
2. Try $\lambda = 0.01$ → $\min_\theta J(\theta)$ → $\theta^{(2)}$ → $J_{cv}(\theta^{(2)})$
3. Try $\lambda = 0.02$ → $\theta^{(3)}$ → $J_{cv}(\theta^{(3)})$
4. Try $\lambda = 0.04$
5. Try $\lambda = 0.08$ → $\theta^{(5)}$
   ⋮

find $\min J_{cv}(\theta)$.
in this case, we we suppose 5th.
$J_{cv}(\theta^{(5)})$

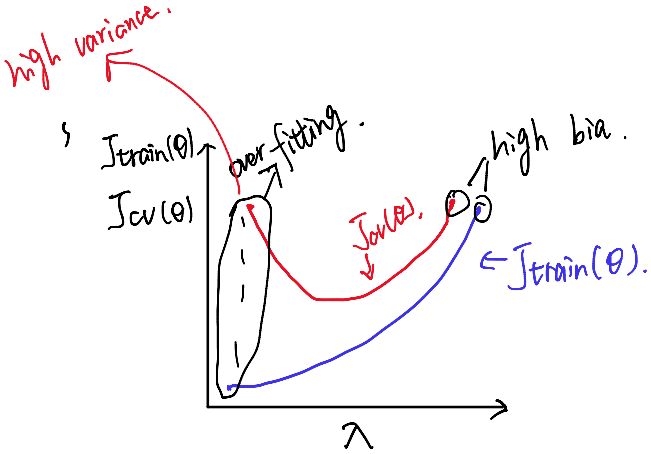
5. Try $\lambda = 0.08$ → $\theta^{(5)}$ → $J_{cv}(\theta^{(5)})$

⋮

12. Try $\lambda = 10$ → $\theta^{(12)}$ → $J_{cv}(\theta^{(12)})$

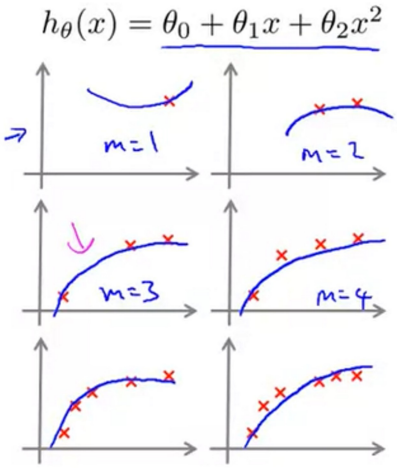
↑ 10.24

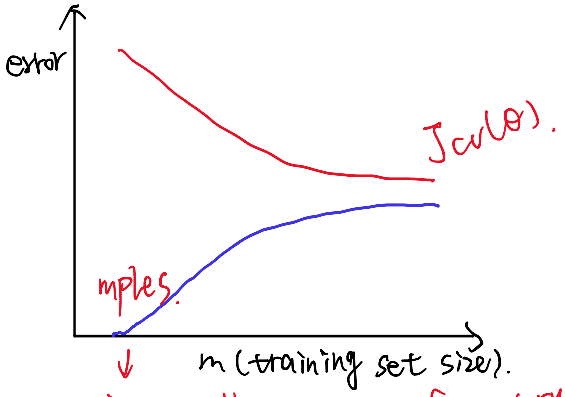
Pick (say) $\theta^{(5)}$. Test error: $J_{test}(\theta^{(5)})$



c. Learning Curves:

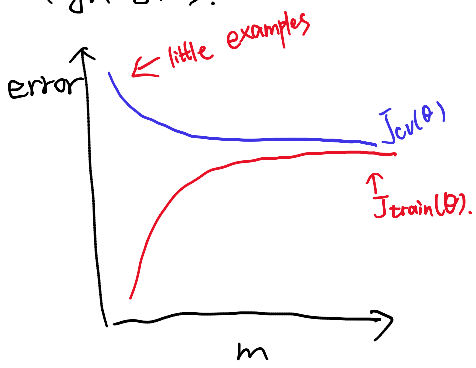
a very useful thing to plot, check algorithm working correctly or improve the performance of the algorithm.

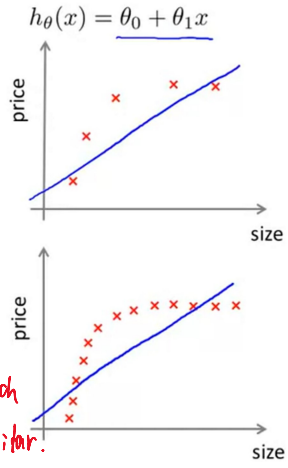




m is small, easy to fit every single exa

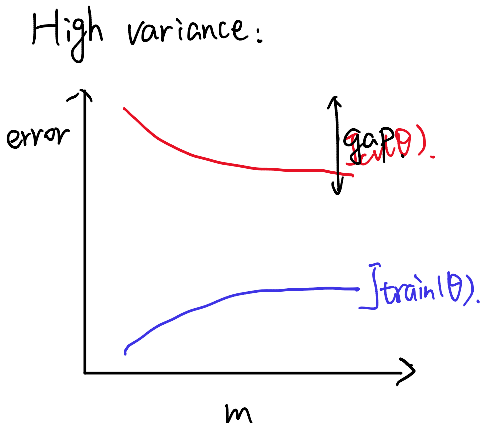
High bias:



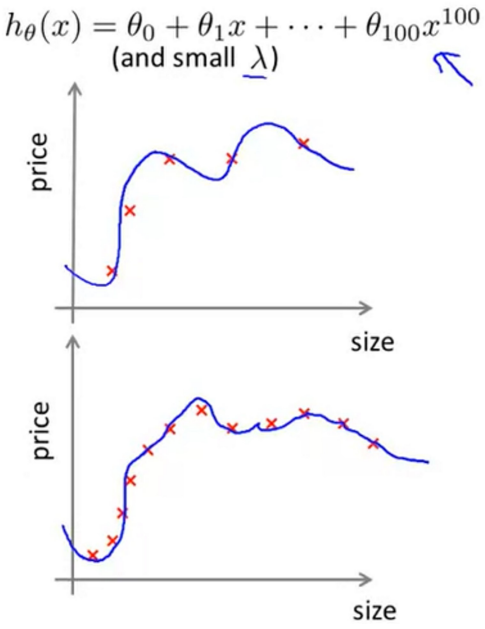


because you have few parameters and so much data, when m is large, train and CV will be similar.

if a learning algorithm is suffering from high bias, getting more training data will not help much.

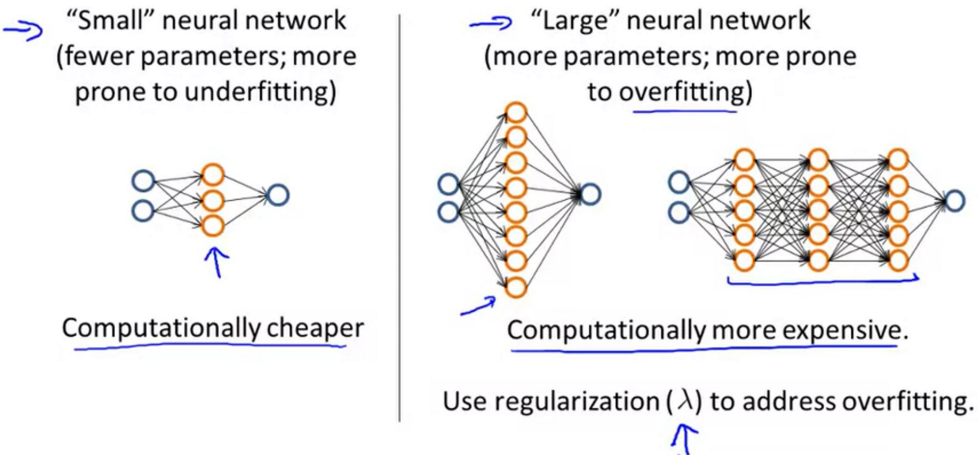


If a learning algorithm is suffering from high variance, getting more training data is likely to help.



b. Deciding What to Do Next Revisitied:
   - **Getting more training examples:** Fixes high variance
   - **Trying smaller sets of features:** Fixes high variance
   - **Adding features:** Fixes high bias
   - **Adding polynomial features:** Fixes high bias
   - **Decreasing λ:** Fixes high bias
   - **Increasing λ:** Fixes high variance

Neural networks and overfitting



using a larger neural network by using regularization to address is overfitting that's often more effective than using a smaller neural network.

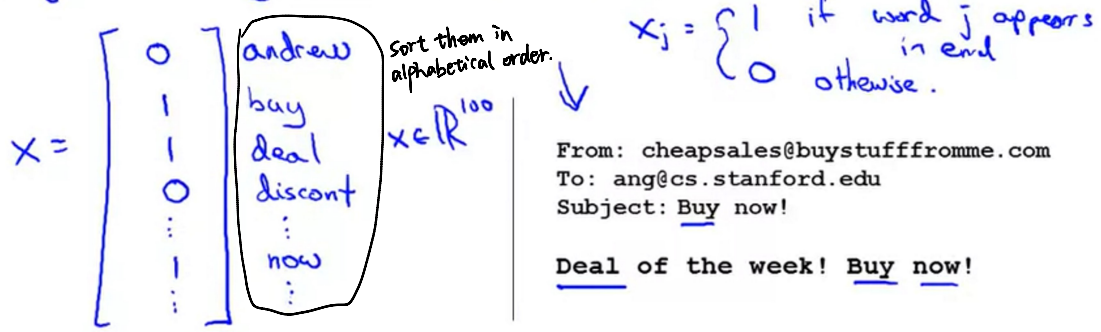
- Building a Spam Classifier:
  a. Prioritizing What to Work On:
     How do we want to represent x?

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, ...



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

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 "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.)

b. 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.

it will tell you current things or shortcomings of the system and give you inspiration to come up with improvements.

We should let evidence guide our decisions on where to spend our time rather than use gut feeling, which is often wrong.

Example:

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.

may not worth to spend time.

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

Deliberate misspellings: 5
(m0rgage, med1cine, etc.)
Unusual email routing: 16
Unusual (spamming) punctuation: 32

work on this problem.

The importance of numerical evaluation

through this evaluation, we'll know our method is good or not.

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

Error Analysis on Cross validation Set.

- Handling Skewed Data:
  a. Error Metrics for Skewed Classed:

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.

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

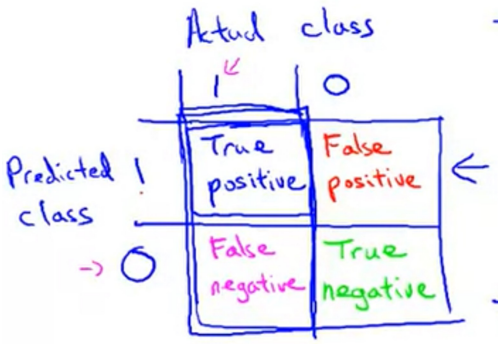
0.5% err.

In this case, means even thoug we just make y=0, it only has 0.5% error. This circumstance sometimes occurs when the ratio of

positive to negative examples is very close to one of two extremes.(above case: the number of positive is much much smaller than negative.)——skewed class.

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?)

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

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

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

$y = 0$
recall = 0

b. Trading Off Precision and Recall:

Trading off precision and recall

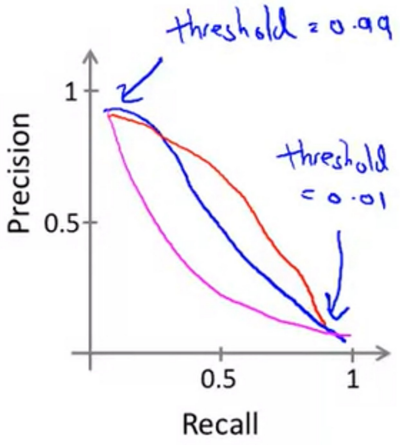
→ precision = $\frac{\text{true positives}}{\text{no. of predicted positive}}$

→ 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.



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

How to compare precision/recall number?

	Precision (P)	Recall (R)	Average	F1 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: ~~$\frac{P+R}{2}$~~

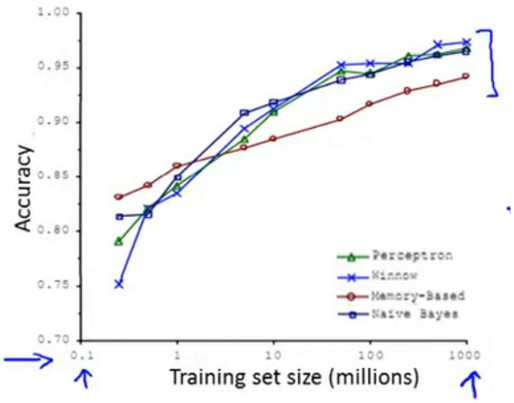
F1 Score: $2\frac{PR}{P+R}$

P=0 or R=0 → $F_1 = 0$ worst
P=1 and R=1 → $F_1 = 1$. best.

- Using Large Data Sets:
  a. 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

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.

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

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

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

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

Ask yourself can a human experts look at the features x and confidently predict the value of y? Because that's sort of a certification that y can be predicted accurately from the features x. secondly can we actually get a large training set, and train the learning algorithm with a lot of parameters in the training set.