No.

Date

Advice for applying ML

- Deciding what to try next

Debugging a learning algorithm:

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

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

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

- Get more training examples
- Try smaller sets of features → To prevent overfitting.
- Try getting additional features (maybe current one not informative enough)
- Try adding polynomial features (x_1^2, x_2^2, x, x_2, etc.)
- Try decreasing λ
- Try increasing λ

Technique to decide take which option

- 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.

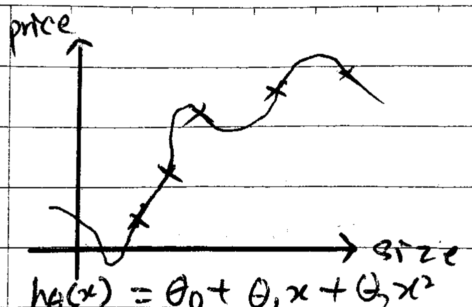
Diagnostics can take time to implement, but doing so can be a very good use of your time. (用这个会吃时间可是好过浪费在没有用的一个一个test option，因为那样的话吃更多时间)

Quiz: True statements about diagnostics:

- Diagnostics can give guidance as to what might be more fruitful things to try to improve a learning algorithm.
- Diagnostics can be time-consuming to implement and try, but still a very good use of time
- A diagnostic can sometimes rule out certain courses of action (changes to learning algorithm) as being unlikely to improve its performance significantly.

Advice for applying machine learning

Evaluating a hypothesis



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

⇒ Fail to generalize to new examples not in training set

x_1 = size of house
x_2 = no. of bedrooms
x_3 = no. of floors
x_4 = age of house
x_5 = average income in neighborhood
x_6 = kitchen size
⋮
x_{100}

Problem with large num of features, will become hard / impossible to plot what hypothesis looks like.
So: Need another way

Evaluating your hypothesis
Dataset:

	Size	Price	
70%	2104	400	Training Set
	1600	330	
	2400	369	
	1416	232	
	3000	540	
	1985	300	
	1534	315	
30%	1427	199	Test Set
	1380	212	
	1494	243	

In order to make sure can evaluate hypothesis we split data into two portions.

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

$(x^{(1)}_{test}, y^{(1)}_{test})$
$(x^{(2)}_{test}, y^{(2)}_{test})$
⋮
$(x^{(m_{test})}_{test}, y^{(m_{test})}_{test})$

m_{test} = no. of test example

It is better to randomly shuffle the Data first before sending the 70% and 30%.

Quiz: Suppose an implementation of linear regression (without regularization) is badly overfitting the training set, in this case we would expect:
The traing error [ ] to be low and the
test error [ ] to be high.

Training / testing procedure for linear regression

- Learn parameter θ from training data (minimizing training error $J(\theta)$) ↳ 70%
- compute test set error: plug in here and start computing

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

↑ definition of test set error in linear regression and using squared error metric

Training / testing procedure for logistic regression

- Learn parameter θ from training data (From 70%)
- compute test set error:

$$J_{test}(\theta) = -\frac{1}{m_{test}} \sum_{i=1}^{m_{test}} y^{(i)}_{test} \log h_\theta(x^{(i)}_{test}) + (1 - y^{(i)}_{test}) \log h_\theta(x^{(i)}_{test})$$

- Misclassification error (0/1 misclassification error):

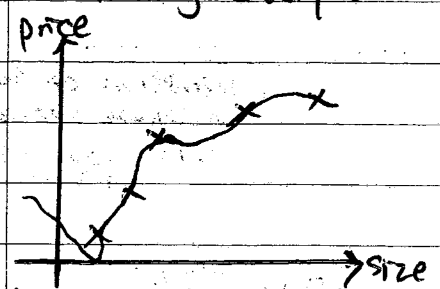
$$err(h_\theta(x), y) = \begin{cases} 1 & \text{if } h_\theta(x) \geq 0.5,\ y = 0 \\ & \text{or if } h_\theta(x) < 0.5,\ y = 1 \end{cases} \text{error}$$
$$\qquad 0 \quad \text{otherwise (hypothesis classified example y correctly)}$$

$$\text{Test error} = \frac{1}{m_{test}} \sum_{i=1}^{m_{test}} err(h_\theta(x^{(i)}_{test}), y^{(i)}_{test}).$$

Advice for applying ML

Model selection and training/validation/test sets

Overfitting example



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

Once parameters $\theta_0, \theta_1, \ldots, \theta_4$ were fit to come set of data(training set), the error of the parameters as measured on that data(the training error $J(\theta)$) is likely to be lower than the actual generalization error.

Model selection

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

d: degree of polynomial

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

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

$\vdots$

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

measure performance on the test set

Can take the model and minimize the training error and will get parameter vector theta

After getting the model with the lowest test set error, Let say, choose fifth order polynomial (value of d that best)

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

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

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

Evaluating your hypothesis

Dataset:

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

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

$(x^{(1)}_{cv}, y^{(1)}_{cv})$
$(x^{(2)}_{cv}, y^{(2)}_{cv})$
$\vdots$
$(x^{(m_{cv})}_{cv}, y^{(m_{cv})}_{cv})$

m_{cv} = no. of cv example

$(x^{(i)}_{cv}, y^{(i)}_{cv})$

$(x^{(1)}_{test}, y^{(1)}_{test})$
$(x^{(2)}_{test}, y^{(2)}_{test})$
$\vdots$
$(x^{(m_{test})}_{test}, y^{(m_{test})}_{test})$

Train/validation/test error

Training error:

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

Cross Validation error:

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

Test error:

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

~~Going to use the cv set to select model~~

Model Selection

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

d=3 3. $h_\theta(x) = \theta_0 + \theta_1 x + \ldots + \theta_3 x^3 \xrightarrow{\text{same}} \theta^{(3)}$

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

$\vdots$

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

Pick the hypothesis with the lowest cv error

exp: Pick $\theta_0 + \theta_1 x_1 + \ldots + \theta_4 x^4$ $d = 4$ fit into

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

Quiz: Consider ~~model selection procedure where we choose the degree of polynomial~~ using a cv set. For the final model (with parameters θ), we might generally expect $J_{cv}(\theta)$ To be lower than $J_{test}(\theta)$ because:

An extra parameter(d, the degree of polynomial) has been fit into the cv set.

$>$ greater than

$>>$ double greater than (much greater than)

Note:

We can now calculate 3 separate error values for the 3 different sets using following method:

1. Optimize the parameters in Θ using the training set for each polynomial degree.
2. Find the polynomial degree d with the least error using the cv set.
3. Estimate the generalization error using the test set with $J_{test}(\theta^{(d)})$, (d = theta from polynomial with lower error);

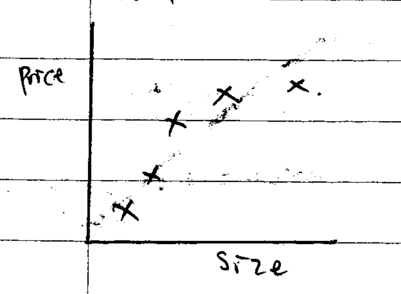
This way, the degree of the polynomial d has not been trained using the test set.

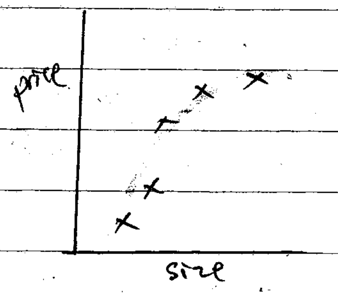
Advice for applying machine learning

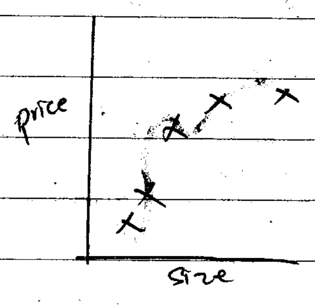
Diagnosing bias vs. variance

- If the learning algorithm doesn't do as well as you wish, most of the time because you have high bias problem or high variance problem. (underfitting / overfitting problem)

Bias / variance







$\theta_0 + \theta_1 x$

High bias (underfit)

d=1

$\theta_0 + \theta_1 x + \theta_2 x^2$

"Just right"

d=2

$\theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4$

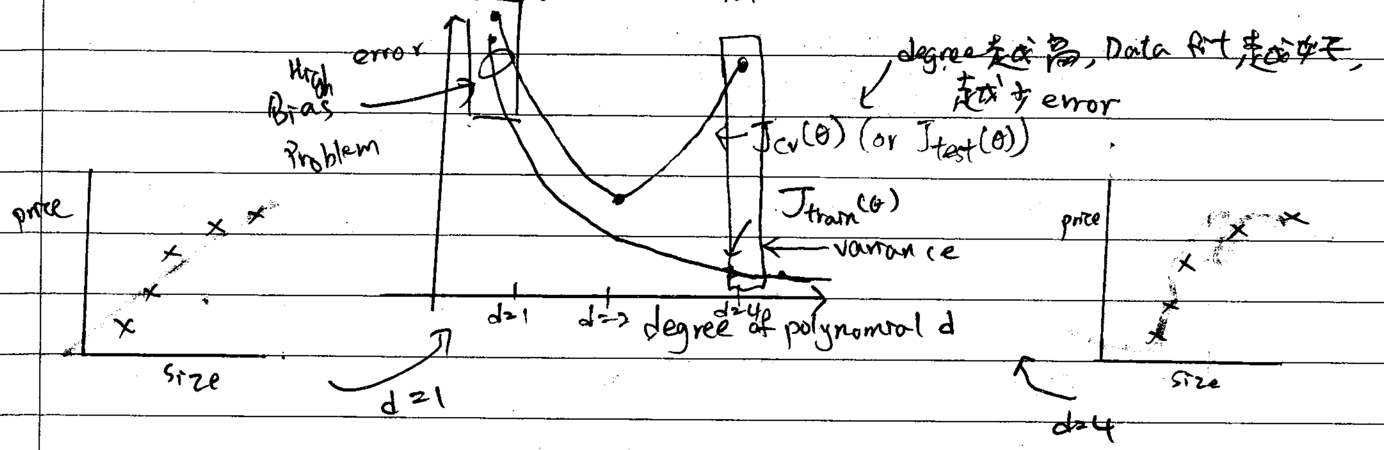
High variance (overfit)

d=4

Bias / variance

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

Cross validation error : $J_{cv}(\theta) = \frac{1}{2m_{cv}} \sum_{i=1}^{m_{cv}} (h_\theta(x^{(i)}_{cv}) - y^{(i)}_{cv})^2$ (or $J_{test}(\theta)$)



$J_{train}(\theta)$ plot 的 是 ~~Non~~ Training set 的 Data, so d 越小, error 就多, d越 error 就 $J_{cv}(\theta)$ plot test set / cv set data, d 小的时候, error 大 因为 underfitting, d 中间 OK 的时候, error 比较小, d 大的时候, error 就回去大 因为 overfitting

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):	Variance (overfit):
$J_{train}(\theta)$ will be high	$J_{train}(\theta)$ will be low
$J_{cv}(\theta)$ will be high as well	$J_{cv}(\theta)$ will be high
$J_{cv}(\theta) \approx J_{train}(\theta)$	$J_{cv}(\theta) >> J_{train}(\theta)$

Advice for applying machine learning

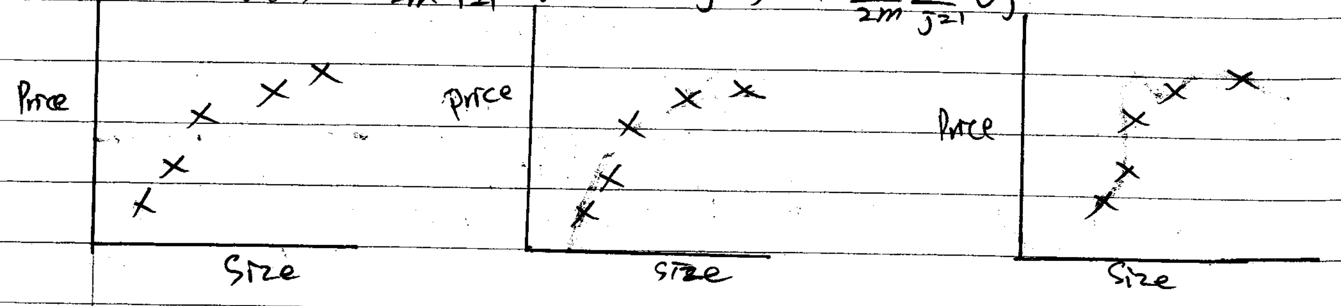
Regularization and bias/variance

Regularization prevent overfitting, but how does it affect bias and variances of a learning algorithms?

Linear regression with regularization

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

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



Large λ — Intermediate λ — Small λ

High bias (underfit) — "Just right" — High variance (overfit)

$\lambda = 10000$: $\theta_1 \simeq 0, \theta_2 \simeq 0 \ldots$ — $\lambda = 0$

$h_\theta(x) \approx \theta_0$

Choosing the regularization parameter λ

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

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

$$J_{train}(\theta) = \frac{1}{2m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2$$ [to be optimization objective, but without regularization term]

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

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

Model selection apply to ~~the~~ selecting regularization parameter lambda (λ)

Choosing the regularization parameter λ

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

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

Put a range of values of λ that you want to try out

1. Try $\lambda = 0 \longrightarrow \min_\theta J(\theta) \rightarrow \theta^{(1)} \rightarrow J_{cv}(\theta^{(1)})$
2. Try $\lambda = 0.01 \longrightarrow \min_\theta J(\theta) \rightarrow \theta^{(2)} \rightarrow J_{cv}(\theta^{(2)})$
3. Try $\lambda = 0.02$ Normally ×2 $\longrightarrow \theta^{(3)} \rightarrow J_{cv}(\theta^{(3)})$
4. Try $\lambda = 0.04$
5. Try $\lambda = 0.08 \longrightarrow$ end up, choose lowest $\longrightarrow \theta^{(5)} \rightarrow J_{cv}(\theta^{(5)})$

⋮

12. Try $\lambda = 10$ (10.24) but close enough $\longrightarrow \theta^{(12)} \rightarrow J_{cv}(\theta^{(12)})$

Pick the model that gives the lowest error on the cv set.

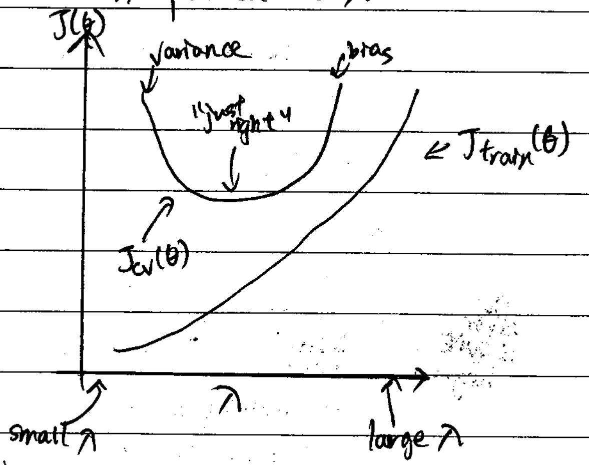
Pick (say) $\theta^{(5)}$. Test error: $J_{test}(\theta^{(5)})$ ← Put back into test set

Bias/variance as a function of the regularization parameter λ

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

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

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



In order to choose the model and the regularization term λ, we need to:

1. Create a list of lambdas (i.e. $\lambda \in \{0, 0.01, 0.02, 0.04, 0.08, 0.16, 0.32, 0.64, 1.28, 2.56, 5.12, 10.24\}$
2. Create a set of models with different degrees or any other variants.
3. Iterate through the λs and for each λ go through all the models to learn some θ.
4. Compute the cross validation error using the learned θ (computed with λ) on the $J_{cv}(\theta)$ without regularization or $\lambda = 0$.
5. Select the best combo that produces the lowest error on the cv set.
6. Using the best combo θ and λ, apply it on $J_{test}(\theta)$ to see if it has a good generalization of the problem.

因为 $J_{train}(\theta)$ 和 $J_{cv}(\theta)$ 用的 Data set 不同，数量也不同，所以得出来的 average squared error 也会不同.

Advice for applying ML

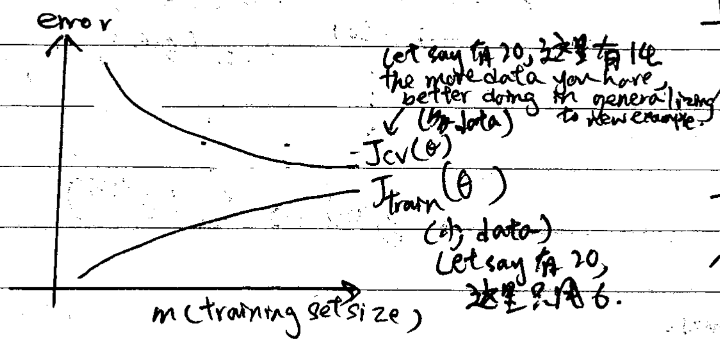
Learning curves ⟶ check/improve the performance of the algorithm.

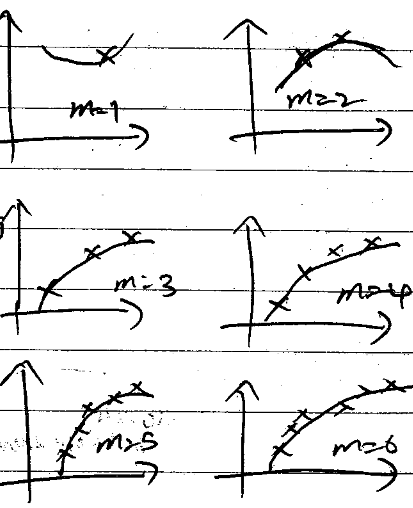
⇒ To diagnose if a physical ~~learning~~ algorithm may be suffering from bias, sort of variance problem or a bit of both.

Example: $h_\theta(x) = \theta_0 + \theta_1 x + \theta_2 x^2$

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

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

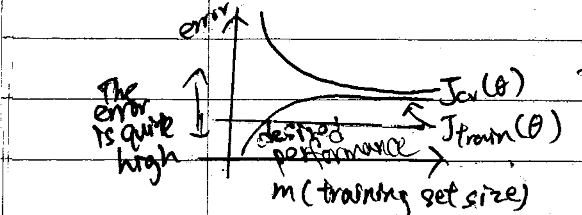




when training set size is small, training error will be small as well.

When training set size become larger, harder to ensure ~~to~~ find quadratic function that fit perfectly.

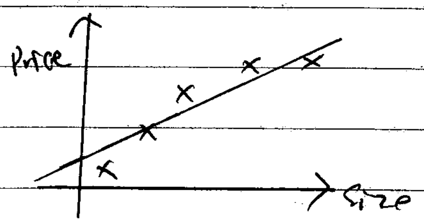
High bias



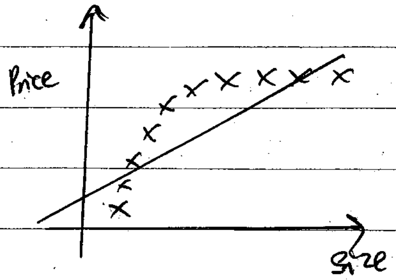
} because has many data but less parameters, and when m is large, performance on $J_{train}(\theta)$ and $J_{cv}(\theta)$ will be very similar

If a learning algorithm is suffering from high bias, getting more training data will not (by itself) help much.

$h_\theta(x) = \theta_0 + \theta_1 x$

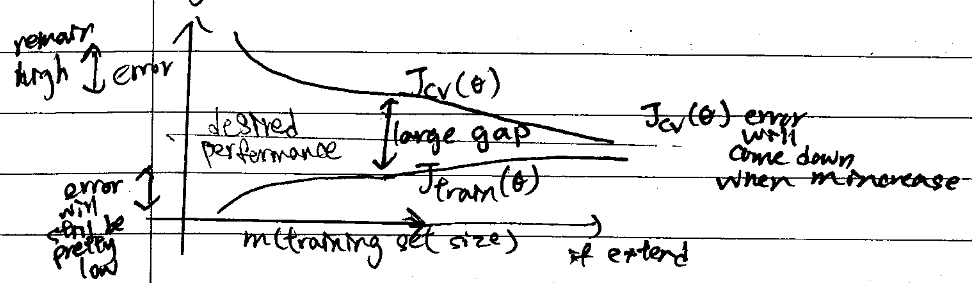


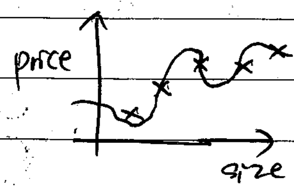
Try increase training set size

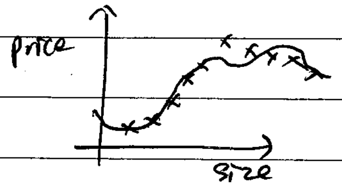


High variance

$h_\theta(x) = \theta_0 + \theta_1 x + \dots + \theta_{100} x^{100}$
(and small λ)







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

Quiz:

In which of the following circumstances is getting more training data likely to significantly help a learning algorithm's performance?

Answer: Algorithm is suffering from high variance
$J_{cv}(\theta)$ (cross validation error) is much more larger than $J_{train}(\theta)$ (training error)

Advice for applying ML ⇒ Deciding what to try next (revisited)

Debugging a learning algorithm:

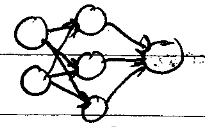
Suppose you have implemented regularized linear regression to predict housing prices. However, when you test your hypothesis in a new set of houses, you find that it makes unacceptably large errors in its prediction. What should you try next?

- Get more training examples ⟶ fix high variance
- Try smaller sets of features ⟶ fix high variance
- Try getting additional features → fix high bias
- Try adding polynomial features (x_1^2, x_2^2, x_1x_2, etc) → fix high bias
- Try decreasing λ → fix high bias
- Try increasing λ → fix high variance

Neural networks and overfitting

"Small" neural network

(fewer parameters; more prone to underfitting)

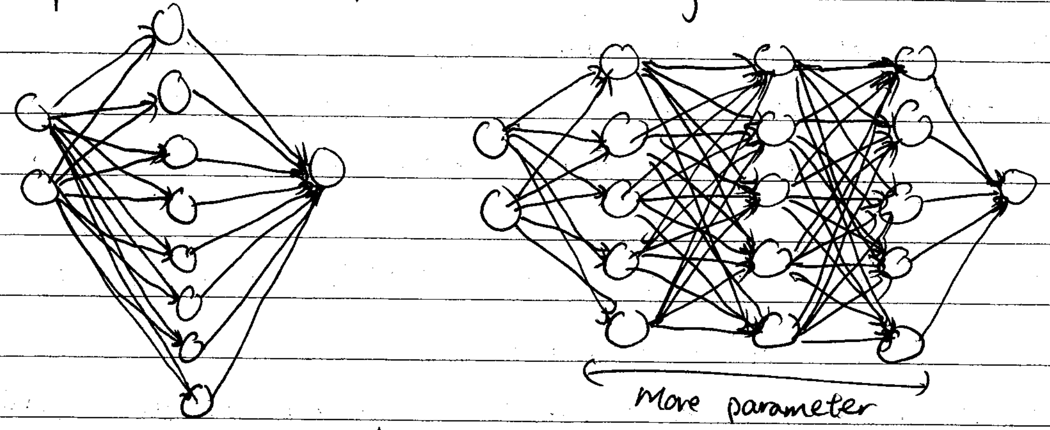


only 1 hidden layer with 3 hidden units

Computationally cheaper

"Large" neural network

(more parameters; more prone to overfitting)



Computationally more expensive

Use regularization(λ) to address overfitting

- usually using a larger neural network by using regularization to address is overfitting that's often more effective than using a smaller neural network.
- Main disadvantage is that it can be more computationally expensive.

Can use the model selecting (cv set)
to check which works best in neural network.

Quiz:

Suppose you fit a neural network with one hidden layer to a training set. You find that the cv error $J_{cv}(\theta)$ is much larger than training error $J_{train}(\theta)$. Is increasing the number of hidden units likely to help? Answer: No, it is currently suffering from high variance, adding hidden units is unlikely to help.

Note:

Model Complexity Effects:

- Lower order polynomials (low model complexity) have high bias and low variance. In this case, the model fits poorly consistently.
- Higher-order polynomials (high model complexity) fit the training data extremely well and the test data extremely poorly. These have low-bias on the training data, but very high variance.
- In reality, we would want to choose a model somewhere in between, that can generalize well but also fits the data reasonably well.

Machine learning system design (How to strategize putting together a complex ML system)

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, ...

spam feature

Not spam as the name of you are there or "now" as urgent.

$$x = \begin{array}{c|l} \left[\begin{array}{c} 0 \\ 1 \\ 1 \\ 0 \\ \vdots \\ 1 \\ \vdots \end{array}\right] & \begin{array}{l} \text{andrew} \\ \text{buy} \\ \text{deal} \\ \text{discount} \\ \vdots \\ \text{now} \\ \vdots \end{array} \end{array}$$

if choose 100 words to use for representation

$x \in \mathbb{R}^{100}$

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

Deal of the week! Buy now!

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

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

features

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 "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.)

a project which create fake email address and try to get these fake email address into the hands of spammers and use that to try to collect tons of spam email, to get a lot of spam data to train learning algorithm.

Quiz: Correct statement

- 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.

Summarize Note:

System Design Example:

Given a data set of emails, we could construct a vector for each email. Each entry in this vector represents a word. The vector normally contains 10,000 to 50,000 entries gathered by finding the most frequently used words in our data set. If a word is to be found in the email, we would assign its repective entry a 1, else if it is not found, that entry would be a 0. Once we have all our x vectors ready, we train our algorithm and finally, we could use it to classify if an email is a spam or not.

So how could you spend your time to improve the accuracy of this classifier?

- Collect lots of data (for example "honeypot" project but doesn't always work)
- Develop sophisticated features (for example: using email header data in spam emails)
- Develop algorithms to process your input in different ways (recognizing misspellings in spam).

It is difficult to tell which of the options will be most helpful.

Machine Learning System Design

Error analysis

When starting working on ML problem, 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 (pattern) in what type of examples it is making errors on.

Error Analycrs.

Example: m_{cv} = 500 examples in cv set

Algorithm misclassifies 100 emails.

Manually examine 100 errors, and categorize them based on:

(i) what type of email it is → pharmacies (sell drug), replica (fake thing), steal passwords, ...

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

Pharma : 12

Replica/fake : 4

⇒ Steal passwords : 53

Others 31

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

Unusual email routing: 16

Unusual (spamming) punctuation: 32

看哪一个 type error 最多, focus on it because algorithm works poor on this type.

有这个 features 的 email 较多, 所以 可以 develop more sophisticated features

The importance of numerical evaluation (it will be better if we can evaluate our learning algorithm that give back a single real number / accuracy (error that tell how well your learning algorithm is doing

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

For this problem (in NLP) → Can use "stemming" software (E.g. "Porter stemmer")

Error might happen ⇒ universe/university

So, it is not easy to tell that whether decide or not to use stemming software for a spam class classifier.

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.

Quiz: Why is the recommended approach to perform error analysis using the cv data used to compute $J_{cv}(\theta)$ rather than the test data used to compute $J_{test}(\theta)$?

Answer: 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 $J_{test}(\theta)$ is no longer a good estimate of how well we generalize to new examples.

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

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

⇒ Then stemming is a good idea

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

If this worse when using stemming, then can quickly decide to go ahead to distinguish or to not distinguish between upper and lower case.

(当然是不用, 如果在这种情况)

Notes:

Do error analysis on cross validation set rather than test set.

Machine Learning system design

Error matrics for skewed classes (having a single real number evaluation metric)

Cancer Classification example (Malignant vs. Benign tumor)

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% patients have cancer

this setting of when ratio of the +ve -ve examples is very close to one of two extremes where in this case num. of +ve example is much smaller than num. of -ve examples because y=1 so rarely, this is case of skewed classes

a non learning code that predict y=0 all the time

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

will give 0.5% error

本来的 algorithm 有 99.2% accuracy (0.8% error)

改了一点东西变 99.5% accuracy (0.5% error)

有 real number evaluation metric 就能告诉我们知道改了到底好不好. 可是, Although can get high classification accuracy or very low errors, not always clear if doing so is really improving quality of classifier because predicting y=0 all the time doesn't seem like a particularly good classifier.

把 predict $y=0$ 更的况 maybe can bring error down. but maybe as low as 0.5%. When we faced with such a skewed classes therefore we would want to come up with a different error metric / different evaluation metric.

Evaluation metric ~~for~~ classification problem with skewed constant

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?)

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

如果像刚才的 algorithm, predict $y=0$ all the time, then recall $= 0$, and will not a good classifier

Recall

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

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

Quiz:

		Actual class	
		1	0
Predicted Class	1	80	20
	0	80	820

Algorithm's precision

$$= \frac{80}{80+20}$$

$$= 0.8$$

Algorithm's recall

$$= \frac{80}{80+80}$$

$$= 0.5$$

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. so we have to modify the threshold to 0.7, so we will only tell the patient that they have cancer only if we think there's a greater than or equal to, 70% chance that they have cancer.

→ Higher precision, lower recall

(你predict有cancer的人是真的有cancer 因为 we only make those decision only if we are pretty confident)

(因为我们 predict $y=1$ on a smaller number of patients)

Suppose we want to avoid missing too many cases of cancer (avoid false negative)
(如果病人有cancer, 我们和他们讲没有就gg了)

~~We~~ For this case, we will set the value to lower value, maybe 0.3. By doing so, we think that more than a 30% of the patient have cancer, and we better be more conservative and tell them that they may have cancer so that they can seek treatment if necessary.

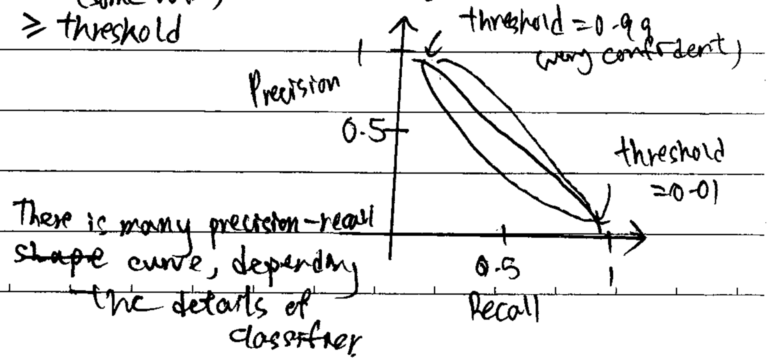
→ Higher recall, lower precision

(Because we're going to be correctly flagging a higher fraction of all the patients that actually do have cancer.)

(Because a higher fraction of patients that we said have cancer, a high fraction of them will turn out not to have cancer after all.)

Depending on where you want, higher precision – lower recall (some value) or higher recall – lower precision

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



There is many precision-recall ~~shape~~ curve, depending the details of classifier

F_1 Score (F score)

How to compare precision/recall numbers? How do we decide which of these algorithm is best?

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

Predict y=1 all the time

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

But this is not that good because this is high and the performance by predicting y=1 all the time not really a good classifier

Average is not a good way to evaluate learning algorithm. There's a different way for combining precision and recall.

Formula for combining precision and recall

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

For F_1 Score to be large, both precision and recall have to be pretty large. 因为如果个太小，结果偏 0，F_1 Score 就变 0 了.

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

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

Intermediate values between 0 and 1, this usually gives a reasonable rank ordering of different classifiers.

⇒ 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 system design

Data for machine learning

Designing a high accuracy learning system

[This experiment was done by 2 researchers in [studying the effect of using different learning algorithms versus trying them out on diff. training set sciences]

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

They vary the training set and try out the learning algorithm, when training set is small, all the algorithm accuracy seem to be the same and low, but when the training set increase, the accuracy will increase. [range: small is 0.1 million and big is 1000 millions].

"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.

Ask yourself ⇒ 如果给这些 input 给 human expert，他能不能 predict 到

Example: For breakfast I ate two eggs. ← feature x is the word behind the blank. enough info to tell me is two but not, to or too.

Counterexample: Predict housing price from only size ($feet^2$) and no other features

only give size of house, not enough info to predict.

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

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

- $\boxed{\text{low bias}}$ algorithms. [can fit very complex functions]

→ $J_{train}(\theta)$ will be small (because it will be able to fit the training set well, so error will be small)

Use a very large training set (unlikely to overfit) ($m > n$) ensure $\boxed{\text{low variance}}$

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

→ $J_{test}(\theta)$ will be small