Sadguru

Deciding what to try next - - -

★ Debugging a learning algorithm.

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

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

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? low accuracy

- Get more training examples.
- Try smaller set of features
- Try getting additional features.
- Try adding polynomial features (x_1^2, x_2^2, x_1x_2, etc)
- Try changing λ. (increase & decrease)

Machine learning diagnostic.

Diagnostic:

A test that you can run to gain insight what is/isn't working with a learning algorithm, & 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.

9 Which of following statements about diagnostic are ~~true~~ true?

✓. It's hard to tell what will work to improve a learning algorithm, so the best approach is to go with gut feeling and just see what works.

✓. Diagnostics can give guidance as to what might be more fruitful things to try to improve a learning algorithm.

✓. Diagnostics ~~can't~~ can be time consuming to implement and try, but they can still be a very good use of your time.

✓. A diagnostic can sometimes rule out certain courses of action (changes to learning algo) as being unlikely to improve its performance significantly.

⑥ Evaluating a hypothesis

Getting low value of training error → good thing
→ ~~under~~ over fitting

Not generalized for new examples

(Training error $J(\theta) \rightarrow$ low
Test error $J_{test}(\theta) \rightarrow$ high)

$h_\theta(x) = \theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4$, Here we can plot & get to know about overfitting.

But when we have many features
$x_1, x_2, x_3, x_4 \ldots x_{100}$
We cannot plot how hypothesis looks.

Dataset:

size	price

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

Training set (70%)

$(x^{(m)}, y^{(m)})$

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

Test set (30%)

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

representation.

Just remember to randomly choose train, test.

★ Training / testing procedure for linear regression.

1) learn parameter θ from training data (minimizing training error $J(\theta)$) → for 70% of data

2) Compute test set error

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

plug θ here, from 1)

★ Training / testing procedure for logistic regression.

1) learn parameter θ from training data

2) Compute test set error

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

Misclassification error (0/1 misclassification error)

getting example wrong / right

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

hypothesis classified example y correctly

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

→ count of how many examples classified wrongly

@ Model Selection and Train/Validation/Test Sets

Model Selection process

- Degree of polynomial?
- What features to include?
- λ^2.

Once your parameter is what fit to some set of data. Maybe the training set, (may be something else), then the ~~hypothesis~~ error of your hypothesis as measured on that same data set, (such as training error) That's unlikely a good estimate of your actual generalization error.

i.e. How well the hypothesis will generalize to new examples.

The training error $J(\theta)$ is likely to be lower than the actual generalization team.

Model Selection.

Minimizing training error gives some parameter vector θ.

1) $h_\theta(x) = \theta_0 + \theta_1 x$ — $\theta_0^{(1)} \rightarrow J_{test}(\theta^{(1)})$
2) $h_\theta(x) = \theta_0 + \theta_1 x + \theta_2 x^2$ — $\theta^{(2)} \rightarrow J_{test}(\theta^{(2)})$
3) $h_\theta(x) = \theta_0 + \theta_1 x + \theta_2 x^3$ — $\theta^{(3)} \rightarrow J_{test}(\theta^{(3)})$

Test set

10) $h_\theta(x) = \theta_0 + \cdots + \theta_{10} x^{10}$ — $\theta^{10} \rightarrow J_{test}(\theta^{(10)})$

d = degree of polynomial — (one more parameter)

Here we choosed model with lowest test error (d=5)
i.e. choose $\theta_0 + \cdots + \theta_5 x^5$

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

→ This will not be a fair estimate of how well my hypothesis generalizes.

because, we ~~choos~~ fit value of d (d=5) according to test set → that gave us the best performance on test set

So, Hypothesis of d=5, is likely to do better on this test set, than it would on new examples that it has n't seen before.

NOT Predictive
[How well ~~How well~~ generalizes to

fit $a_0, a_1, \ldots$ on → training set → Hypothesis → New examples

a_0, a_1 will go well on training set, but not well on new examples.

fit d on → test set → Hypothesis Examples we haven't seen before.

Performance of hypothesis on that test set may not be fair estimate of how well the hypothesis is likely to do on

Generalising this fault.

★ Evaluating your hypothesis.

Dataset: size Price

Training set (60%) $(x^{(1)}, y^{(1)})$ $(x^{(m)}, y^{(m)})$

cross Validation set / Validation set (cv) (20%) $(x^{(1)}_{cv}, y^{(1)}_{cv})$ $(x^{(m_{cv})}_{cv}, y^{(m_{cv})}_{cv})$

test set. $(x^{(1)}_{test}, y^{(1)}_{test})$ $(x^{(m_{test})}_{test}, y^{(m_{test})}_{test})$

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

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

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

Model selection

~~We~~ ~~we~~ Instead of ~~test~~ set to select d,
↳ we will use cv set to select d.
i.e. to select model.

1) $h_\theta(x) = \theta_0 + \theta_1 x$ → $\min_\theta J(\theta)$ → $\theta^{(1)}$ → $J_{cv}(\theta^{(1)})$
2) $h_\theta(x) = \theta_0 + \theta_1 x + \theta_2 x^2$ → $\theta^{(2)}$ → $J_{cv}(\theta^{(2)})$
3) → $\theta^{(3)}$ → $J_{cv}(\theta^{(3)})$
selected → 4) $h_\theta(x) = \theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4$ → $\theta^{(4)}$ → $J_{cv}(\theta^{(4)})$ lowest
10) $h_\theta(x) = \theta_0 + \theta_1 x + \dots + \theta_{10} x^{10}$ → $\theta^{(10)}$ → $J_{cv}(\theta^{(10)})$

d=4, Pick $\theta_0 + \theta_1 x_1 + \dots + \theta_4 x^4$

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

↳ Here we are left only with test data.
We will use test data to measure generalization error of model.

9" We might generally expect $J_{cv}(\theta) < J_{test}(\theta)$?.

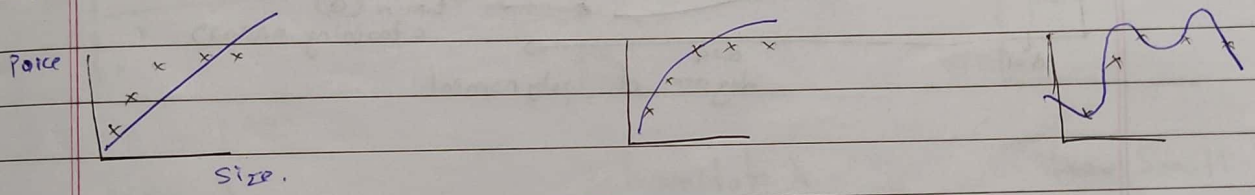
→ An extra parameter d has been fit to cross validation ~~to~~ set, not according to test set.

--@ Diagnosis Bias vs. Variance. - - - - - - -

Most of time algo is in underfitting / overfitting problem

Bias or Variance or a bit of both ?.

✱ Bias / Variance.

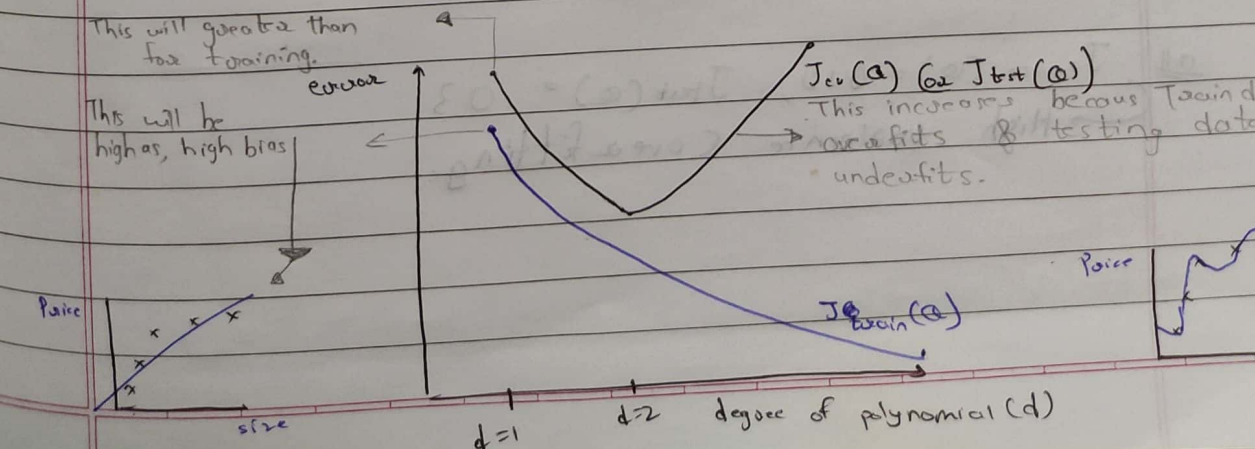


$\theta_0 + \theta_1 x$ | $\theta_0 + \theta_1 x + \theta_2 x^2$ | $\theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4$

High bias (underfit) | Just right $d=2$ | High variance. (overfit)

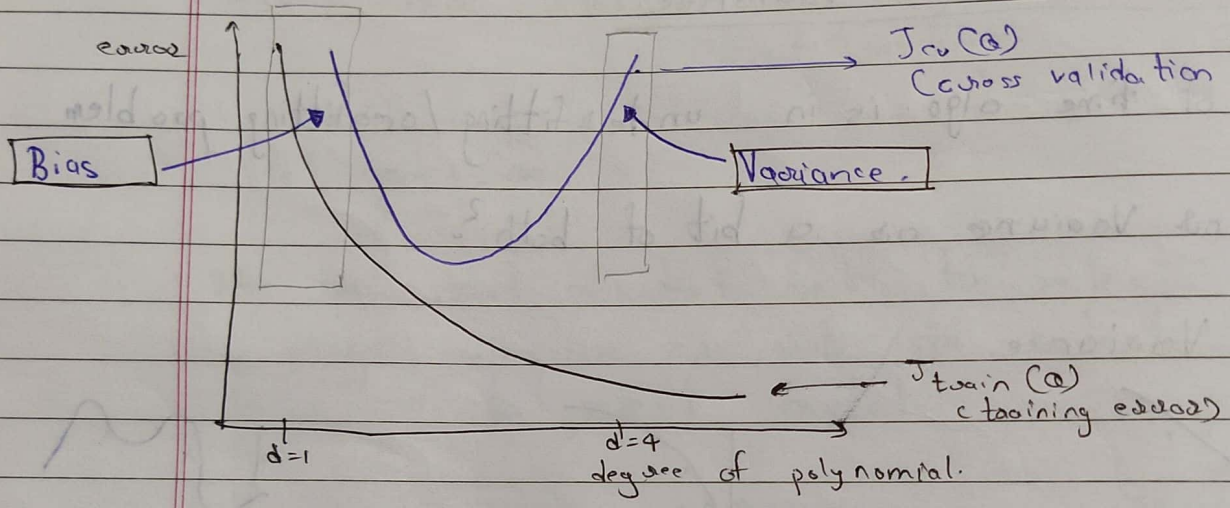
Training error : $J_{train}(\theta) = \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2$

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



☆ 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. It is 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
$J_{cv}(\theta) >> J_{train}(\theta)$

eg: $J_{train}(\theta) = 0.10$, $J_{test}(\theta) = 0.3$
→ High variance (overfitting.

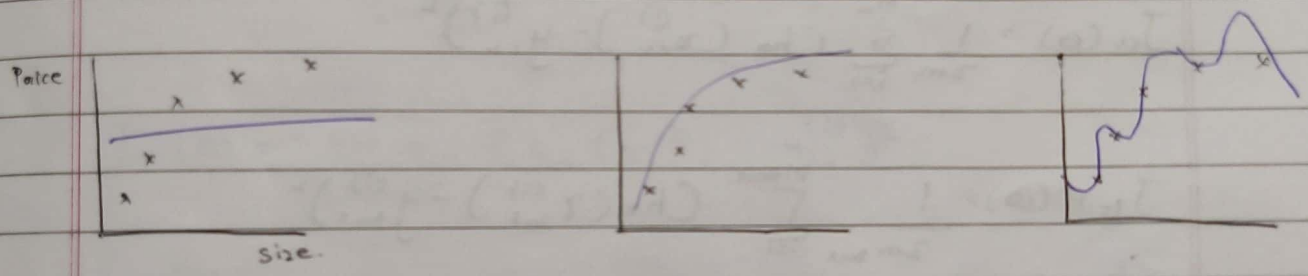
- - ⑥ Regularization and Bias / Variance. — — —

→ How regularization affects Bias & Variance.

★ Linear regression with regularization.

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

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



large λ | Intermidate λ | ~~larg~~. Small λ

High bias (underfit) | Just right | High variance. (overfit)

$\lambda = 1000$, $\theta_1 \approx 0$, $\theta_2 \approx 0$, ...

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

$\lambda = 0$ $(\lambda \to 0)$

So, no regularization term.

☆ Choosing a regularization parameter. λ.

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

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

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

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

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

Choosing regularization parameter.

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

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

we can go even less ↑

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 $\lambda = 0.02$
4) Try $\lambda = 0.04$
5) Try $\lambda = 0.08$

→ $J_{cv}(\theta^{(5)})$ lowest $J_{cv}(\theta$

12) Try $\lambda = 10.24$ (10) → $\theta^{(12)}$ → $J_{cv}(\theta^{(12)})$

↓

X2

~~Test~~ Training data → | choosing λ | choosing Θ |

Page No: 88
Date: / /

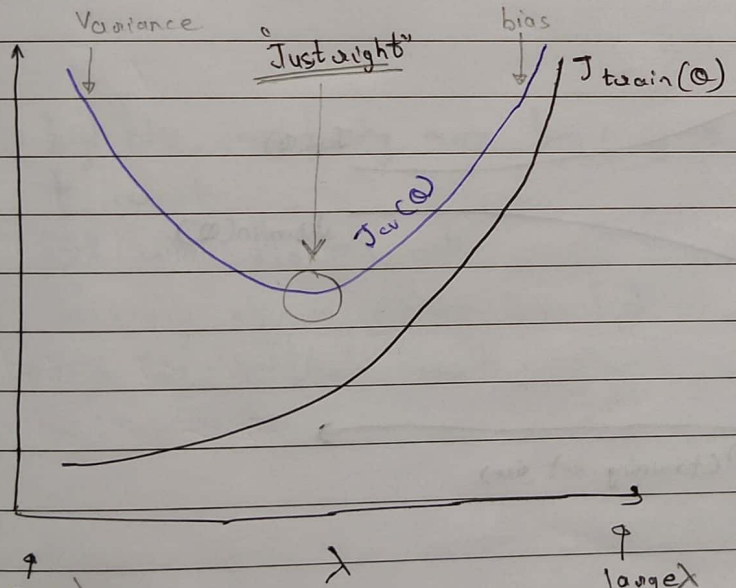
Pick (say) $\Theta^{(5)}$. Test error : $J_{test}(\Theta^{(5)})$

★ Bias / Variance as a function of the regularization parameter λ.

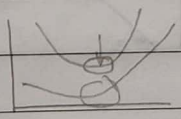
$$J(\Theta) = \frac{1}{2m}\sum_{i=1}^{m}(h_\Theta(x^{(i)}) - y^{(i)})^2 + \frac{\lambda}{2m}\sum_{j=1}^{m}\Theta_j^2$$

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

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



Here, we don't have considered regularization term in J_{train} & J_{cv}. I thinks its by mistake. We should have considered regularization.



And Also, for right λ, J_{train} will also be lowest.

learning curve will tell about bias/variance or both.

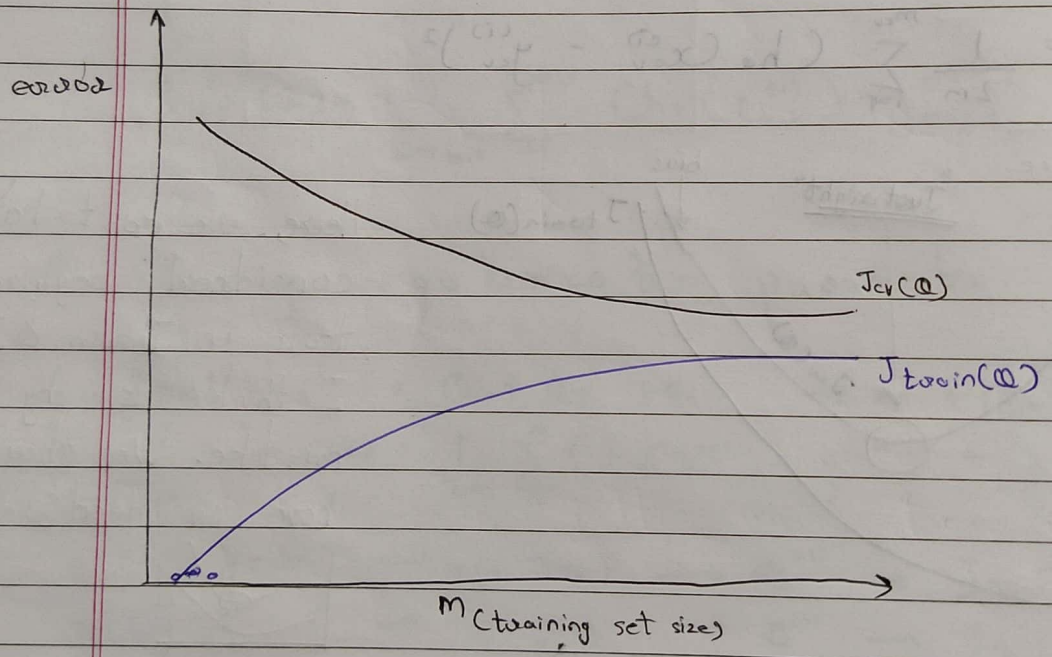
-- ⑫ Learning Curves

To diagnose → ~~Algo~~ Algo suffering from bias
→ Variance problem
→ or bit of both

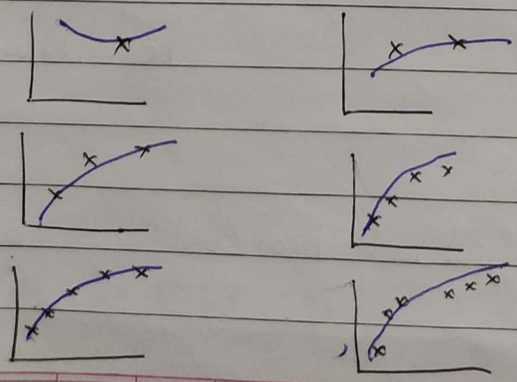
✯ learning curves

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

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



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

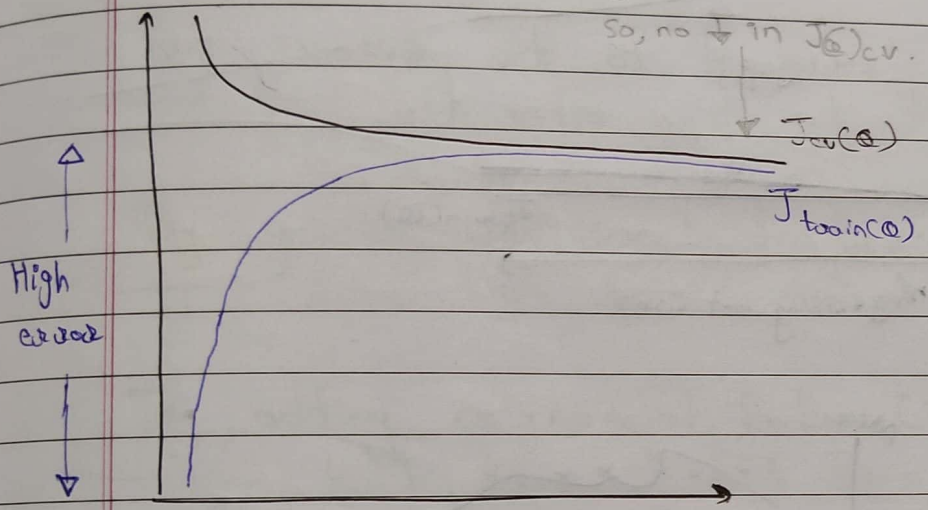


So, $m = 1/2/3$, { ~~training~~ $J_{train}(\theta) = 0$ Without regularization

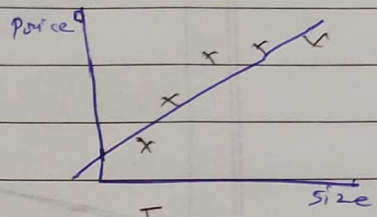
$J_{train}(\theta) \approx 0$ ~~$(\lambda = 0)$~~ $(\lambda > 0)$ Regularization

☆ High bias

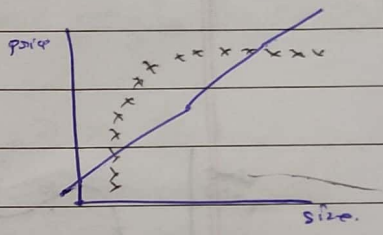
increasing m will given nearby same hypothesis, so same line, So, no ↓ in $J_{cv}(\theta)$.



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



J_{ex}



more ex.

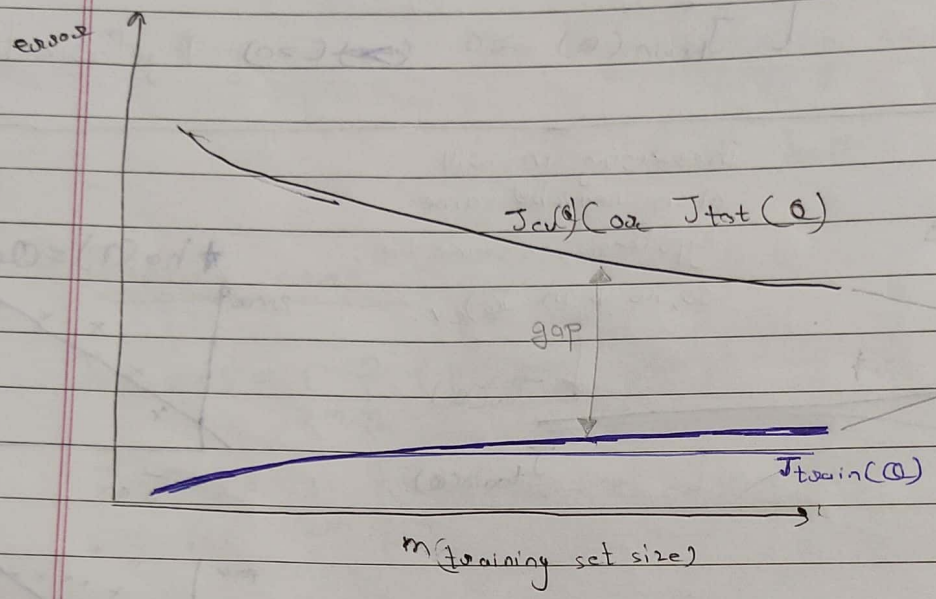
but same line

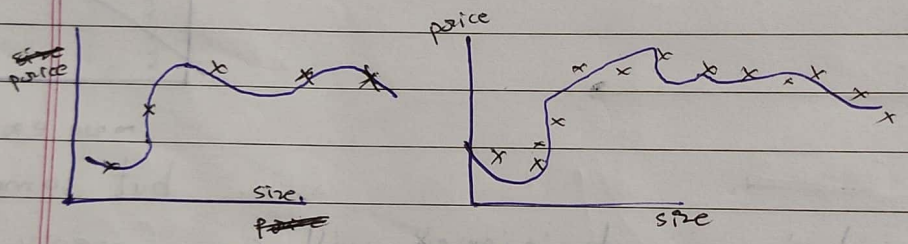
In high bias, considering more training ex. will ← (pretty close to previous) not work.

$J_{cv}(\theta)$ will flatten out.

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

☆ High Variance.





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

In high variance, getting more training data is likely to help.

→ $J_{train}(\theta)$↑, $J_{cv}(\theta)$↓ → This is what actually matters.

Q// Getting more ~~data~~ training data will help learning algo's performance

- Algo is suffering from high bias. (2.)
- Algo is ——— Variance.
- $J_{cv}(\theta) >> J_{train}(\theta)$
- $J_{cv}(\theta) \approx J_{train}(\theta)$ (2.)

Deciding What to do Next Revisited - - - -

If model is making very high error than expected

Get more training examples
→ fixes high variance, not fixes high bias.

Try smaller sets of features
→ fixes high variance, not fixes high bias

Try getting additional features
→ fixes high bias (current hypothesis may be too simple)

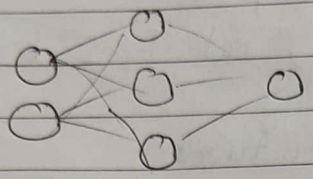
Try adding polynomial features (x_1^2, x_2^2, x_1x_2, etc)
→ fixes high bias, not fixes high variance.

decreasing λ → fixes high bias

Increasing λ → fixes high variance.

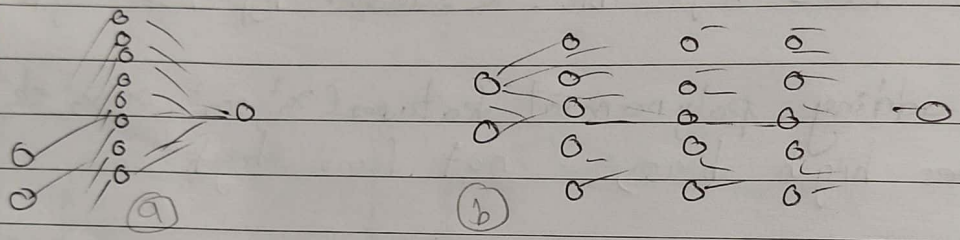
☆ Neural networks and overfitting.

"small" neural network
fewer parameters; more prone to underfitting



Computationally cheaper

"large" neural network.
more ~~compu~~ parameters; more prone to overfitting



Computationally more expensive

The best way is using large neural network with regularization (disadvantage → Time complexity)

(a) or (b)
Use a train, test, cv,

Try training with one/2/3 hidden layers.
See $J_{cv}(\theta)$

Q// We fit a neural network, $J_{cv}(\theta) >> J_{train}(\theta)$
Is increasing no of hidden layers likely to help?

→ No, because it is currently suffering from high ~~bias~~ variance, so adding hidden units is unlikely to help.

Better to skip this video.

⑥ Prioritizing what to work on

* 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

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

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

"Deal of the Day! Buy now!

In practice, take most significant (frequently) occurring n words (10,000 to 50,000) in training set rather than manually pick 100 words & use them as features.

- Collect lots of data. "Honeypot" proj.
  Features → email routing

- Is discount ↔ Discounts same., Punctuation's.

- Detecting misspellings, medicine, watches

(11) visit courses P.T.O.

☆ Recommended approach.

- Take simple algo; implement & use cv. set. → test
- Plot curves (more data/ features)
- Error analysis Manually examine ~~eg~~ examples.

e.g.

$m_{cv} = 500$

wrongly classified cv = 100

examine 100 → 1) Type

2) which features will help

In NLP → Porter stemmer is used

Q11 Which of the following do you agree? Check all that apply.

✓ For some learning applications, it is possible to imagine coming up with many different features. But it can be hard to guess in advance which feature will be most useful.

- For spam classific^n, algo to detect and correct deliberate misspellings will make a significant improvement in accuracy.

- Because spam classif^n uses very high dimensional features vectors, (n = 50,000 ...), a significant effort to collect a massive training set will always be a good idea

✓ There are often many possible ideas for how to develop a high accuracy learning system; "gut feeling is not recommended" to choose among alternatives.

Error Metrices for skewed example.

Train logistic regression model $h_\theta(x)$.
($y=1$ if cancer / $y=0$ else otherwise)

Found that we got 1% error on test set.
→ (99% correct diagnoses)
but from 1%
→ only 0.5% only had actually cancer → 0.5% penalty

If we have said, no one has cancer, → 0.5% penalty.

☆ Skewed Class

Accuracy is not a really good measure.

99.2% accuracy	(0.8% error)
99.5% accuracy	(0.5% error)

→ we cannot tell which one gives more quality to classifier.
Because predicting 0 all time, is not a good classifier.

☆ Precision (Recall)

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

y=1 Positive
y=0 Negative

		Actual class 1	Actual class 0	
Predicted class	1	True Positive	False Positive	→ Predicts 1 really 0
	0	False negative	True negative	→ Telling cancer patient that you don't have cancer

FN → must be as low as possible

False negative must be ↓,
False pos, not must, but should be↓

☆ Precision

Of all patients where we predicted y=1, what fraction actually have cancer.

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

☆ Recall

Of all patients ~~where~~ that actually have cancer, what fraction did we correctly detect as having cancer.

$$\text{Recall} = \frac{\text{True positives}}{\#\text{ actual positive}} = \frac{\text{True positives}}{\text{True positive + False negative}}$$

•) If we make a classifier which predicts $y=0$ all the time.

→ recall=0 as there will no true positive.
As there will no y predicted as 1.

(ii) Precision = ?

80	20
80	820

$= \frac{80}{80+20} = 0.8$

(ii) Recall = ?

80	20
80	820

$= \frac{80}{80+80} = 0.5$

• Classifier with high precision / high recall is actually a good classifier -- (cannot decide by accuracy)

• While defining precision / recall,
We assign $y=1$ to rare class that we are trying to detect

An algorithm cannot cheat (predicts $y=1$ all the time) and get high precision / recall

High precision / High recall ⟶ good classifier

--- (a) Trading off Precision & Recall ---

for some applications we will want to somehow control trade-off b/w precision & recall.
↳ balance, maybe

☆ Trading off precision & Recall.

logistic regn : $0 \leq h_\theta(x) \leq 1$
Predict $y=1$ If $h_\theta(x) \geq$ ~~0.5~~ ~~0.7~~ 0.9
Predict $y=0$ If $h_\theta(x) <$ ~~0.5~~ ~~0.7~~ 0.9

1) Suppose we want to predict $y=1$ (cancer) only if very confident.
↳

High precision,	lower recall
We are classifying $y=1$, only if we are more confident. So, more no. of predictions will be correct.	We are predicting $y=1$, on smaller no of patients
(False positive ↓, maybe)	(False negative ↓, maybe)

2)# Suppose we want to avoid missing too many cases of cancer (avoid false negative)
false negative must be as low as possible.

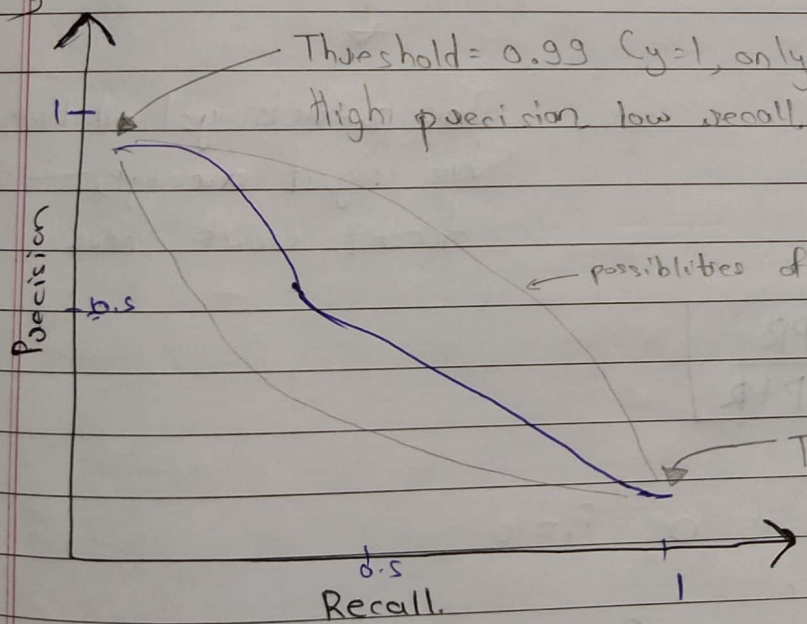
In doubt we want to predict $y=1$. (low confident)

predict 1 if $h_\theta(x) \geq 0.3$
predict 0 if $h_\theta(x) \leq 0.3$

High recall , lower precision.

We have to choose according to situation 1)#/2)#

A general trade off b/w precision vs Recall



Note: Try diff thresholds & pick which maximizes F_1

F_1 score (F score)

How to compare precision/recall numbers.

	Precision (P)	Recall (R)	Avg	F_1 score
Algorithm 1	0.5	0.4	0.45	0.444
Algo 2	0.7	0.1	0.4	0.175
Algo 3	0.02	1.0	0.51	0.039

Precision / Recall → 2 no's

F_1 score → 1 no

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

This predicts y=1 all time, still has higest average, → average doesn't work here

$$F_1 \text{ score } 2\frac{PR}{P+R}$$

If any of R/P is 0, $F_1 = 0$

$P=0$ & $R=0$ ⇒ F_1 score $= 0$

$P=1$ & $R=1$ ⇒ F_1 score $= 1$

We measure R & P of Cross Validation set which maximizes $\frac{R+P}{2}$

@ Data for Machine learning -

It's not who has the best algorithm that wins, It's who has the most data.

A inferior algo can beat superior algo, if more data is given, (more data to both algo, still inferior can win)

★ large data rationale.

→ To test this, question, can human predict y, given x.

- Assume x has all features to predict y correctly.
- Use learning algorithm with many parameters (e.g. logistic/linear, nn)
  low bias algo. → $J_{train}(\theta)$ will be small.
- Use a very large training set (unlikely to overfit)
  low variance → $J_{train}(\theta) \approx J_{test}(\theta)$

→ $J_{(test)}(\theta)$ will be small.

@ Having large data can improve algo, but unlikely to help when,

- ✓ feature x ~~do not~~ cannot predict accurately. (linear regn)
- ✓ Algo with large no of features. (low bias)
- ✓ feature x cannot predict y accurately (nn, big #hidden layer)
- ✓ Not using regularization.