Week 6

If when you test your hypothesis you get a high accuracy, but it fails on new data, try the following:

(1)* - Get more training examples
- Try smaller sets of features
- Try adding polynomial features
- Try adding additional features
- Try decreasing λ
- Try increasing λ

This is used later in the notes

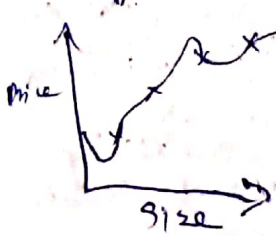
How do you decide which to do?

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.

Evaluating a hypothesis

If overfitting occurs, then it may show high accuracy but fail on new data



On small no. of features we can plot and see, but what if we have many features?

We split the data:

	notation
Training set - 70%	m
Test set - 30%	m_{test} $(x^{(i)}_{test}, y^{(i)}_{test})$

Distribute them randomly (first randomly sort the data, then distribute)

Training/Testing Procedure (for linear regression)

- Learn parameter θ from training data (Minimising training error $J(\theta)$)

- Compute test set error

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

For logistic regression,

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

- Misclassification error

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

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

Model selection

How to choose polynomical features,

d - degree of polynomial

d=1 1. $h_\theta(x) = \theta_0 + \theta_1 x \xrightarrow{\min_\theta J(\theta)} \theta^{(1)}$ - That is the minimized cost function's θ

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

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

$\vdots$

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

Find θ for the different models

Then $J_{test}(\theta^{(1)})$, $J_{test}(\theta^{(2)}) \ldots J_{test}(\theta^{(3)})$

and we choose which has the least cost

But note that if fit well to the training set, and may not fit well for new data

∴ We split the data as follows

- Training set - 60% - m
- Cross validation set - 20% - m_{cv} $(x_{cv}^{(i)}, y_{cv}^{(i)})$
- Test set - 20% - m_{test} $(x_{test}^{(i)}, y_{test}^{(i)})$

Train / validation / test error

Training error:

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

Cross validation error:

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

Test error:

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

Now we choose model as follows:

1. $h_\theta(x) = \theta_0 + \theta_1 x \rightarrow \theta^{(1)} \rightarrow J_{cv}(\theta^{(1)})$
2. $h_\theta(x) = \theta_0 + \theta_1 x + \theta_2 x^2 \rightarrow \theta^{(2)} \rightarrow J_{cv}(\theta^{(2)})$
3. $h_\theta(x) = \theta_0 + \theta_1 x + \ldots + \theta_3 x^3 \rightarrow \theta^{(3)} \rightarrow J_{cv}(\theta^{(3)})$

⋮

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

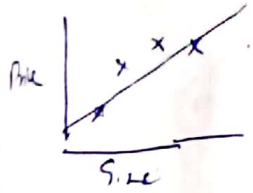
So we fit the θ on the cross validation test and pick the best model

Let's say 4 was the best,

Now we estimate generalization error for test set $J_{test}(\theta^{(4)})$

Diagnosing bias vs variance:

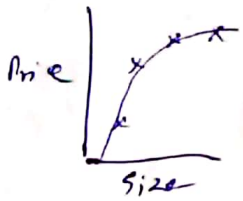
Bias / Variance



d=1 $\theta_0 + \theta_1 x$

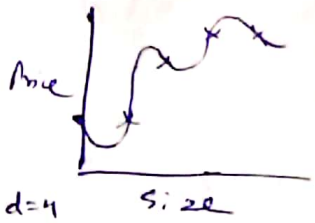
High Bias

Results in underfit



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

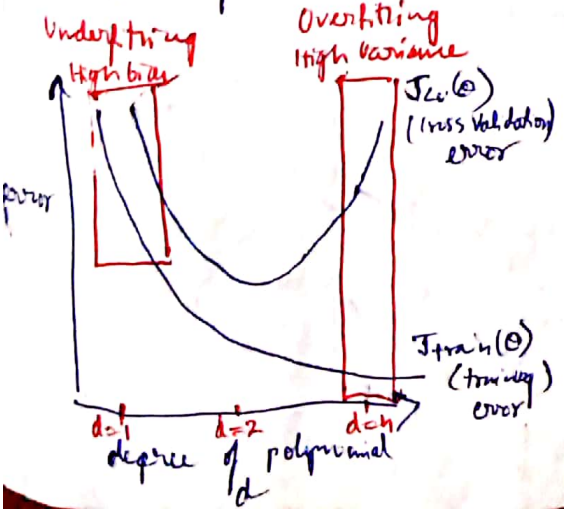
Just right



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

High variance

Results in overfit



Bias (underfit):

→ $J_{train}(\theta)$ will be high,

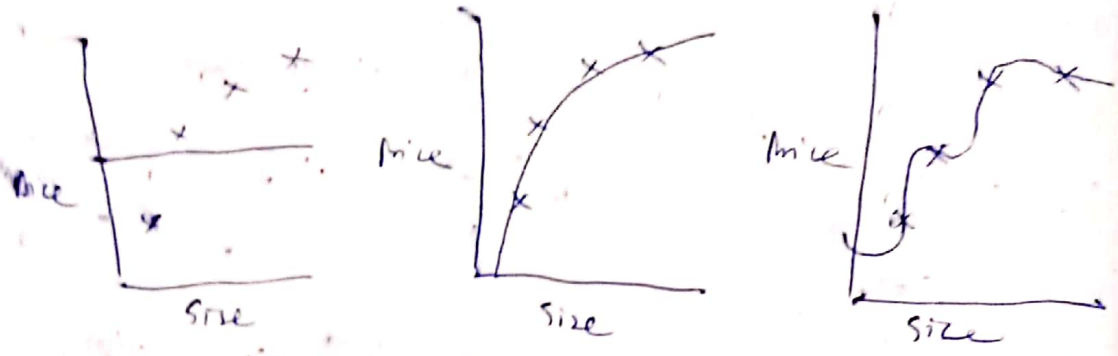
→ $J_{cv}(\theta)$ will be high, $J_{cv}(\theta) \approx J_{train}(\theta)$

Variance (overfit):

→ J_{train} will be low

→ $J_{cv}(\theta) \gg J_{train}(\theta)$

Regularization and bias/variance



Large λ

High bias (underfit)

$\lambda = 10000$, $\theta_1 \approx 0$, $\theta_2 \approx 0$ ...

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

Intermediate λ

Small λ.

High variance (overfit)

$\lambda = 0$

Choosing λ:

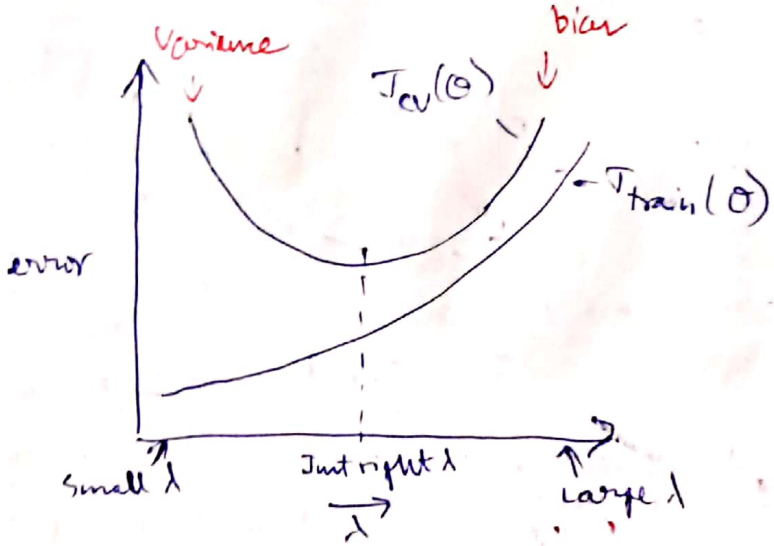
1. Try $\lambda = 0$ $\xrightarrow{\min_\theta J(\theta)}$ $\theta^{(1)}$ $\rightarrow$ $J_{cv}(\theta^{(1)})$
2. Try $\lambda = 0.01$ $\xrightarrow{\min_\theta J(\theta)}$ $\theta^{(2)}$ $\rightarrow$ $J_{cv}(\theta^{(2)})$
3. Try $\lambda = 0.02$ $\xrightarrow{\min_\theta J(\theta)}$ $\theta^{(3)}$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

⋮ ⋮ ⋮

12. try $\lambda = 10$ $\xrightarrow{\min_\theta J(\theta)}$ $\theta^{(12)}$ $\rightarrow$ $J_{cv}(\theta^{(12)})$

Pick the best θ (Lets say $\theta^{(5)}$)

Then $J_{test}(\theta^{(5)})$



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

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

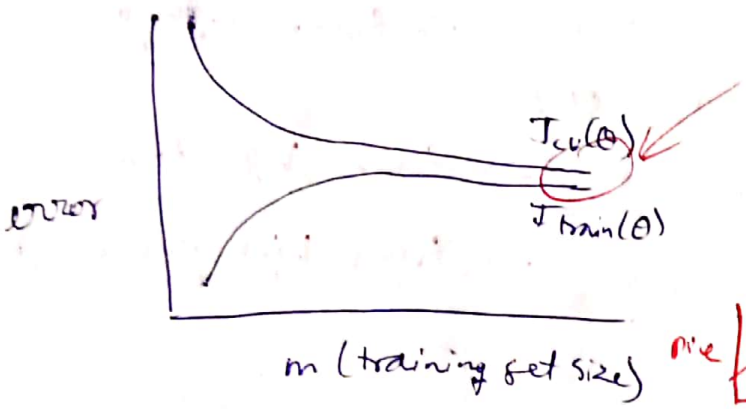
$$J_{train}(\theta) = \frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)}\right)^2$$

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

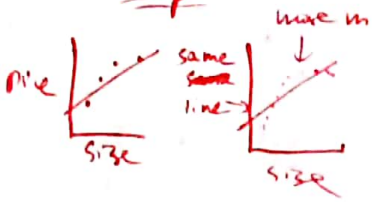
Learning curves

- is a tool to find if the model is suffering from underfitting or overfitting

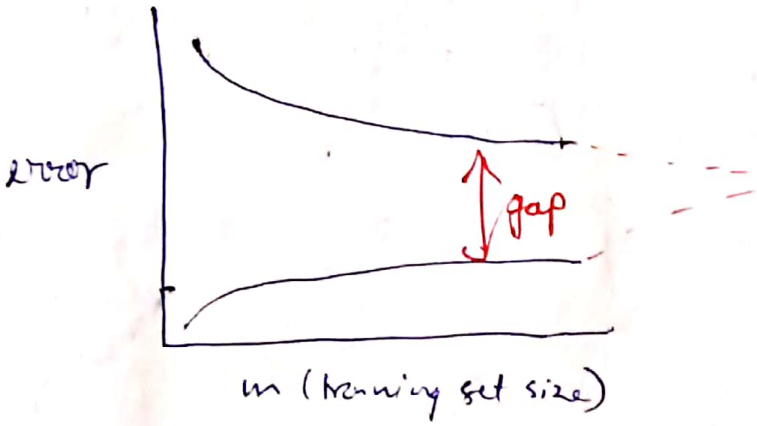
Experiencing high bias:

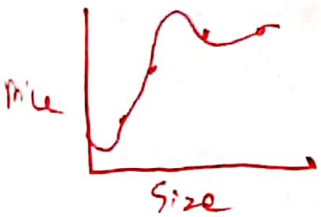


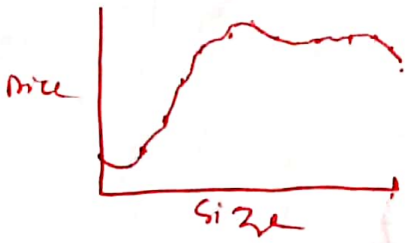
After certain m, the graph flattens out, so getting more data won't help



Experiencing high variance







∴ GETTING MORE TRAINING EXAMPLES (INCREASING M) HELPS ONLY FOR HIGH VARIENCE

Looking back at (1)*

→ Get More training examples → fixes high variance

→ Try smaller sets of features → fixes high variance

→ Try getting additional features → fixes high bias

→ Try adding polynomial features → fixes high bias
$(x_1^2, x_2^2, x_1x_2, etc)$

→ Try decreasing λ → fixes high bias

→ Try increasing λ → fixes high variance

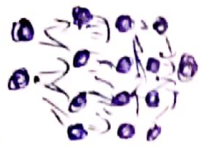
Neural Networks:

Small NN

- Few parameters
- Computationally cheaper

→ Prone to underfitting

Large NN



→ More units

→ More layers

→ Computationally more expensive

→ Prone to overfitting

→ Use regularization

To decide no. of layers:
- Plot $J_{cv}(\theta)$ with different no. of layers and see which fits best

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 (eg. deal, buy, discount, andrew, now, ...)

Not spam example

$x = \begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \\ \vdots \\ 1 \\ \vdots \end{bmatrix}$ andrew, buy, deal, discount, ..., now, ...

Indicates if the word appeared in the email

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

Deal of the week! Buy 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
  - Eg. Honeypot project
- Develop sophisticated features based on email routing information (from email header)
- Develop sophisticated features for message body, eg. should "discount" and "discounts" be treated as the same word? How about "deal" and "dealer"? Features about punctuation?
- Develop sophisticated algorithm to detect misspellings (eg. m0rtgage, med1cine, w4tch).

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 errors on examples in the cross validation set and try to spot a trend where most of the errors were made

The importance of numerical evaluation

It is important to get error results as a single, numerical value, inorder to access the algorithm's performance.

For example, in the spam classifier, if we use stemming (treating similar words like discount/discounts/discounted as same word) we get a error rate of 3% instead of 5% without it, then we can decide to add it to the model. So using numerical evaluation we can determine whether to keep a feature or not.

Error metrics for skewed classes:

Consider a cancer classification example. And let's say we get an error of 1%. This seems great since accuracy is 99%.

However lets say only 0.50% ($\frac{0.5}{100}$) patients have cancer.

Then a non-learning algorithm that always returns $y = 0$ (person has no cancer) will have an error of 0.5%.

So skewed classes is when one class is substantially lower in number than the other class such that an algorithm that always returns the majority class will have less error.

∴ We need another evaluation metric

Precision/Recall

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

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

Precision (what fraction actually has cancer? of all the ppl we predicted $y=1$)

$$\frac{\text{True Positives}}{\#\text{predicted positives}} = \frac{\text{True +ve}}{\text{True +ve + False +ve}}$$

∴ A model that always predicts $y=0$, will have recall = 0

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

$$\frac{\text{True +ve}}{\#\text{actual +ve}} = \frac{\text{True +ve}}{\text{True +ve + False -ve}}$$

Trading off precision and recall

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

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

Predict 0 if $h_\theta(x) <$ threshold

- usually we used to set threshold as 0.5

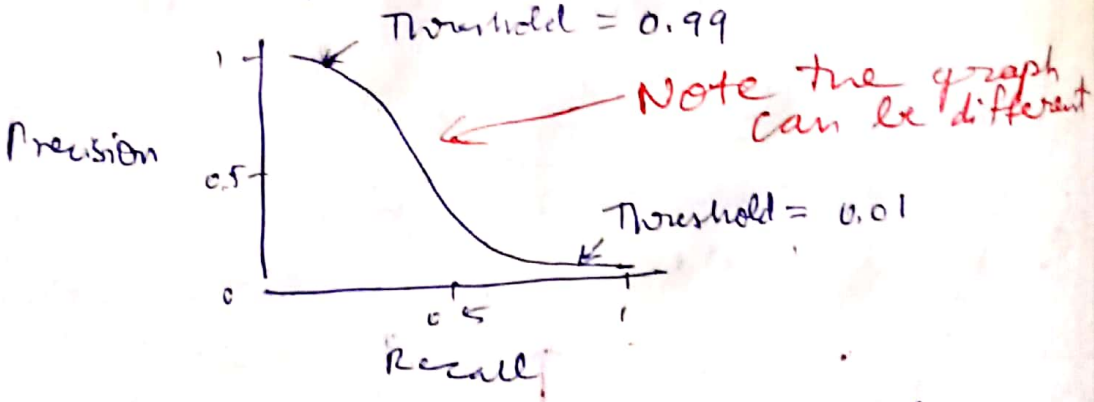
Consider two scenarios :

1) Suppose we want to predict $y=1$ (cancer) only if we are very confident. Then we set threshold = 0.7 or 0.9 or some high number.

   → higher precision, lower recall

2) Suppose we want to avoid missing too many cases of cancer (avoid false negatives), then we use a low threshold like 0.3.

   → higher recall, lower precision



Can we choose the threshold automatically?

Before we had 1 numerical evaluation but now we have 2, which makes choosing an algorithm from another hard.

Using average isn't a good idea.

Using precision and recall: F_1 score: $2\frac{PR}{P+R}$ ← F score method

	Precision (P)	Recall (R)	F_1 Score
Algo 1	0.5	0.4	0.444
Algo 2	0.7	0.1	0.175
Algo 3	0.02	1.0	0.0392
	✗	—	✗ —

Data for ML

learning

1) We use a low-bias algorithm (logistic/linear regression with many features, NN with many hidden ~~layers~~ units). Due to the large number of parameters it'll have low bias. $J_{train}(\theta)$ will be small

2) Use a large training set (~~unlikely to overfit~~) so overfitting is unlikely to happen.

$$J_{train}(\theta) \simeq J_{test}(\theta)$$

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

We need to also have enough features such that a human expert can predict y. Eg: Just given a house's size, a human expert can't predict the price (since locality, furnished or not, etc matters). So we need to add all these features.