Debug and improve learning algorithm.

some promising avenue :

- Get more training examples → fix high variance
- trying a smaller sets of features → fix high variance
- trying additional features → fix high bias
- adding polynomial features → fix high bias
- increasing $\lambda$ → fix high variance
- decreasing $\lambda$ → fix high bias

a hypothesis may have low error for the training examples but still be inaccurate, because of overfitting, thus we split the dataset into two sets : a training set and a test set

training Set (70%) → learn $\theta$ and minimize $J_{train}(\theta)$

test Set (30%) → compute the test set error $J_{test}(\theta)$

The Test Set Error :

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

for classification : misclassification error (aka 0/1 misclassification error)

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

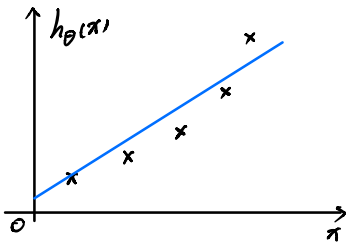
the average test error is :

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

which gives us the proportion of the test data be misclassified.

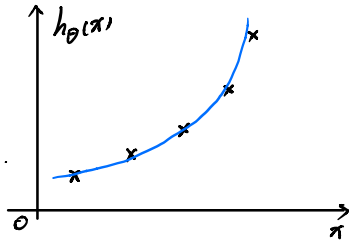
## Model Selection and train / validation / test sets

choose the degree of polynomial function to decide the model.



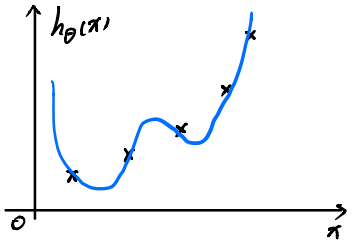

$\theta_0 + \theta_1 x$

$d = 1$




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

$d = 2$




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

$d = 4$

60%

training set

learn $\theta$ and minimize $J(\theta)$ for each polynomial degree, $\Rightarrow$ $\theta^{(1)}, \theta^{(2)}, \theta^{(3)}, \ldots$

20%

Cross validation set

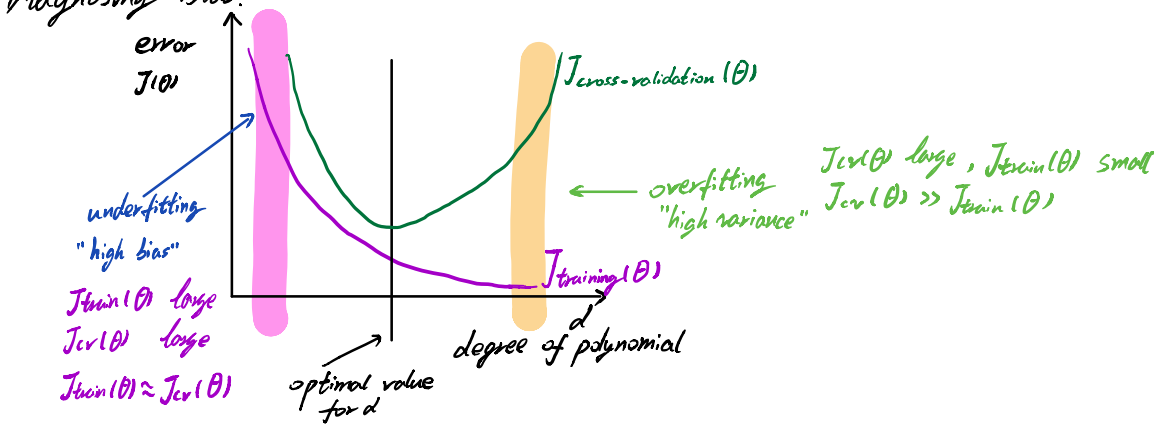
select the degree $d$ with least error using cross validation set. $\min_{vc} J(\theta) = J(\theta^{(d)})$

20%

test set

estimate the generalization error using the test set with $J_{test}(\theta^{(d)})$ $\Rightarrow$

## Diagnosing Bias:

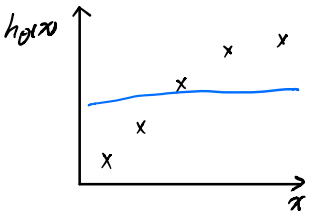


# Regularization and Bias/Variance

## choosing the regularization parameter $\lambda$

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

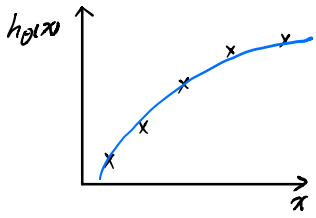
$$J(\theta) = \frac{1}{2m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2 + \frac{\lambda}{2n}\sum_{j=1}^{n}\theta_j^2$$




large $\lambda$

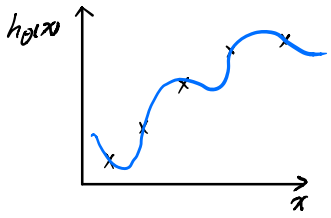
high bias (underfit)

$\lambda = 10000$




intermediate $\lambda$

just right




small $\lambda$

high variance (overfit)

$\lambda = 0$

split the dataset by 60%, 20%, 20%, compute $\min_k J_{cv}(\theta^{(k)})$ then select $\lambda$

try $\lambda = 0$ $\xrightarrow{\times 2}$ fit $\min_\theta J(\theta)$ $\rightarrow$ $\theta^{(1)}$ $\rightarrow$ compute $J_{cv}(\theta^{(1)})$

$\lambda = 0.01$ $\xrightarrow{\times 2}$ $\min_\theta J(\theta)$ $\rightarrow$ $\theta^{(2)}$ $\rightarrow$ $J_{cv}(\theta^{(2)})$

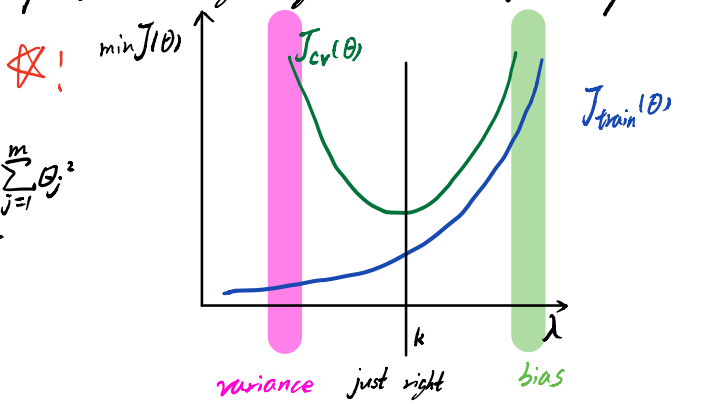
$\lambda = 0.02$ $\xrightarrow{\times 2}$ $\min_\theta J(\theta)$ $\rightarrow$ $\theta^{(3)}$ $\rightarrow$ $J_{cv}(\theta^{(3)})$

$\lambda = 0.04$ $\xrightarrow{\times 2}$ $\min_\theta J(\theta)$ $\rightarrow$ $\theta^{(4)}$ $\rightarrow$ $J_{cv}(\theta^{(4)})$

$\vdots$ $\quad$ $\vdots$ $\quad$ $\vdots$

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

then compute $J_{test}(\theta)$ to see if it has a good generalization of the problem.




if let:

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

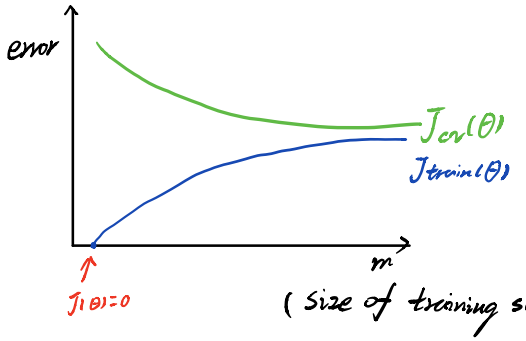
$$J_{train}(\theta) = \frac{1}{2m_{train}}\sum_{i=1}^{m_{train}}(h_\theta(x_{train}^{(i)}) - y_{train}^{(i)})^2$$

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

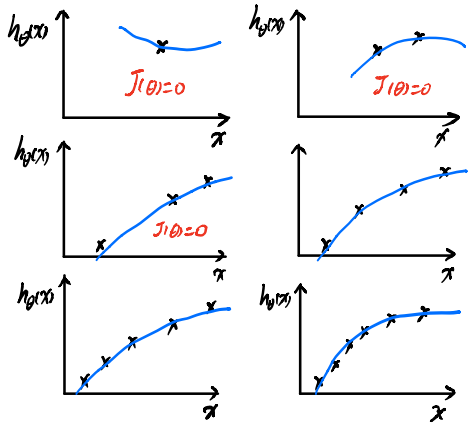
# Learning Curves

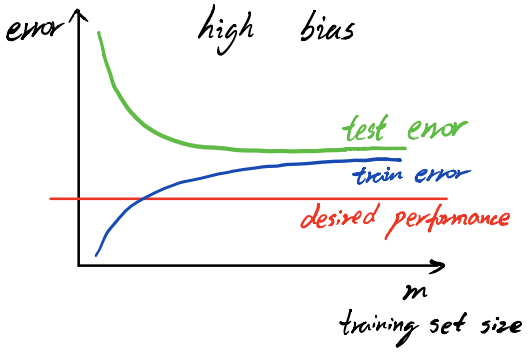
$$J_{train}(\theta) = \frac{1}{2m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2$$

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




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




m large, $J_{train}(\theta) \approx J_{cv}(\theta)$

getting more training data doesn't help


m large, $J_{train}(\theta) < J_{cv}(\theta)$

getting more training data helps

Diagnose neural networks

neural network with fewer parameters, layers is prone to underfitting, it's also computationally cheap.

neural network with more parameters, layers is prone to overfitting, it's also computationally expensive. you can use regularization to address fitting.

neural network with mutiple hidden layer using regularization prone to perform better than single hidden layer network. You can train mutiple hidden layer network using cross validation set and then choose the one perform best.

Machine Learning System design.

Build a spam classifier

a supervised learning example, $x$ = features of the email, $y$ = spam(1) or not spam(0).

feature $x$: choose $n$ ($n>100$) words indicative of spam/not spam.

$$x = \begin{bmatrix} 0 \\ 1 \\ 1 \\ 1 \\ \vdots \\ 0 \end{bmatrix} \begin{matrix} \text{andrew} \\ \text{buy} \\ \text{deal} \\ \text{discount} \\ \vdots \\ \text{now} \end{matrix} \qquad x \in \mathbb{R}^n \qquad x_i = 1 \text{ if } i\text{th word appears}$$

how to spend your time to make it have low error:

- collect lots of data
- develop sophisticated features based on email routing information or message body
- develop sophiscated algorithm to detect misspelling.

Recommended approach on starting a machine learning problem:

- start with a simple algorithm that you can implement quickly. Implement it and test it on your cross-validation dataset.
- plot learning curves to decide more data, more features etc. may help.
- error analysis: manually examine the examples in cross-validation set that your algorithms made errors on. See if you spot any systematic trend in what type of example it is making errors on.

Error Analysis:

$m_{cv} = 500$, algorithm misclassified 100 emails, manually examine the 100 errors, and categorize them based on:

(1) what types of the misclassified email? ⇒
- pharma: 12
- replica/fake: 4
- steal password: 53
- other: 31

(2) what features you think would have helped algorithm classify them correctly? ⇒
- misspelling: 5
- unusual email routing: 16
- unusual punctuation: 32

need numerical evaluation of algorithm's performance:

distinguish upper/lower case? Y 3% error  N 5% error

Error Matrics for skewed classes:

e.g. cancer classification problem:

use a logistic regression model $h_\theta(x)$ to predit cancer (1), otherwise(0)

find that you got 1% error on test set (99% correctly diagnoses)

actually, only 0.5% of patients have cancer.

if we define function predicting cancer : $y=0$, means for all cases just return 0 (not cancer), the accuracy of hypothesis should be 99.5%, and the error rate = 1 - 99.5% = 0.5%, it looks pretty good but it is obviously not a proper hypothesis function. Accuracy should not be used to measure the performance of hypothesis for they are skew classes (the difference of cancer and not cancer is huge), hence we need to introduce a new measure:

## Precision / Recall

we usually set $y=1$ in presence of rare class (cancer)

| predicted class \ actual class | 1 | 0 |
|---|---|---|
| 1 | true positive | false positive |
| 0 | false negative | true negative |

$$\text{Accuracy} = \frac{\text{true positive} + \text{true negative}}{\text{all example}}$$

Precision

(of all patients where we predicted $y=1$, the fraction actually has cancer)

$$= \frac{\text{true positive}}{\text{predicted positive}} = \frac{\text{true positive}}{\text{true positive} + \text{false positive}}$$

Recall

(of all patients that actually have cancer, the fraction did we correct detect as having cancer)

$$= \frac{\text{true positive}}{\text{actually positive}} = \frac{\text{true positive}}{\text{true positive} + \text{false negative}}$$

## Trade off precision and recall

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

predict 1 if $h_\theta(x) \ge$ threshold (usually 0.5)

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

to select the most suitable threshold, we need to trade off between precision and recall, and use a single number matric.

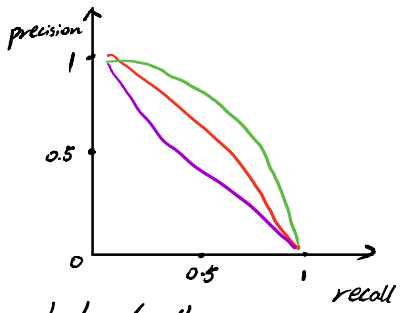
suppose we want to predict y=1 (cancer) only if very confident:

threshold = 0.7 ↑ precision ↑ recall ↓

suppose we want to avoid missing too many cancer:

threshold = 0.3 ↓ precision ↓ recall ↑

| e.g. | Precision (P) | Recall (R) |
|---|---|---|
| Algorithm1 | 0.5 | 0.4 |
| Algorithm2 | 0.7 | 0.1 |
| Algorithm3 | 0.02 | 1.0 |




if y always =1 actually this is a bad algorithm, so we can't use average $\frac{R+P}{2}$ to measure.

we use $F_1$ score: $2\frac{PR}{P+R}$

when $P=0$ or $R=0$, $F_1\text{ score}=0$

$P=1$ and $R=1$, $F_1\text{ score}=1$

## Data For Machine Learning

—— how many data to use.

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

I: useful test: given the input test x, can a human expert confidently predict y;

II: use a learning algorithm with many parameters (many features, many hidde layers)

→ this tends to be a low bias algorithm

→ $J_{train}(\theta)$ will be small.

III. use a large training set (unlikely to overfit)

→ this tends to be a low variance algorithm

→ $J_{test}(\theta) \simeq J_{train}(\theta)$, $J_{test}(\theta)$ will be small

} meet both can get a good algorithm