WEEK - 6

⇒ Machine learning 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.

⇒ Training / testing procedure for linear regression

1) = Learn parameter θ from training data (minimizing training error J(θ))
2) = Compute test set error:

Test Set Error.

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

⇒ Training / Testing for Logistic Regression.

1) = Learn parameter θ from training data.
2) = Compute test set error :-

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

P.T.O.

3) = Misclassification error (0/1 misclassification error):

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

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

⇒ We generally expect $J_{CV}(\theta)$ to be lower than $J_{test}(\theta)$ because — An extra parameter (d - is the degree of polynomial) has been fit to the cross validation set.

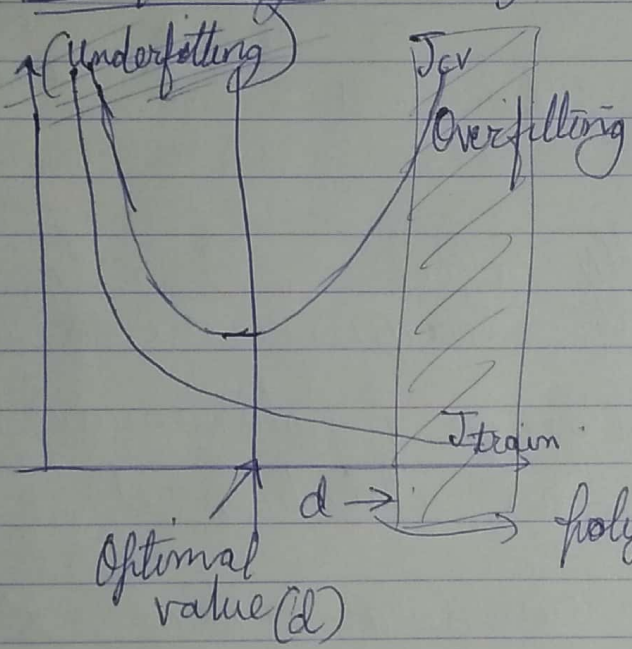
⇒ We break our dataset into 3 sets :-
Training set = 60%, Test set → 20%.
Cross Validation set → 20%

→ We can now calculate separate error values for the three different sets using the 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 cross validation set.
3) = Estimate the generalization error using the test set with $J_{test}(\theta^{(d)})$, (d = theta from polynomial with lower error)

⇒ Our algo has – high Bias ie $J_{train}(\theta)$ is high & $J_{test}\theta$ or $J_{cv}\theta$ is also high. ($J_{test} \geq J_{train}$) → underfit

⇒ If our algo has $J_{train}(\theta)$ low but $J_{cv}(\theta)$ or $J_{test}(\theta)$ very high, then its overfitting or High Variance.

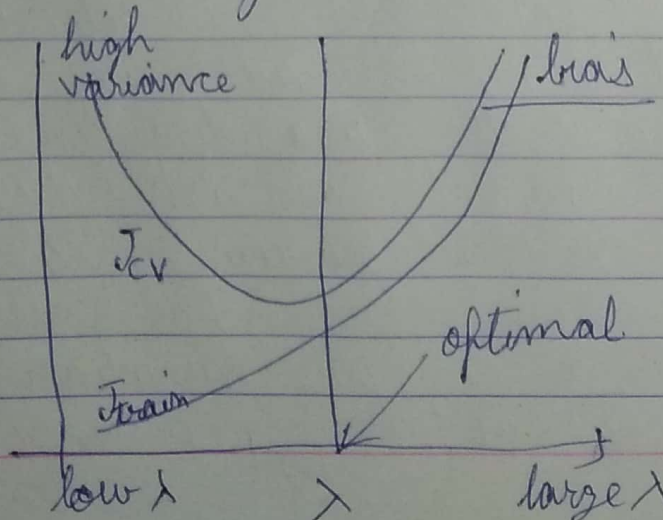
⇒



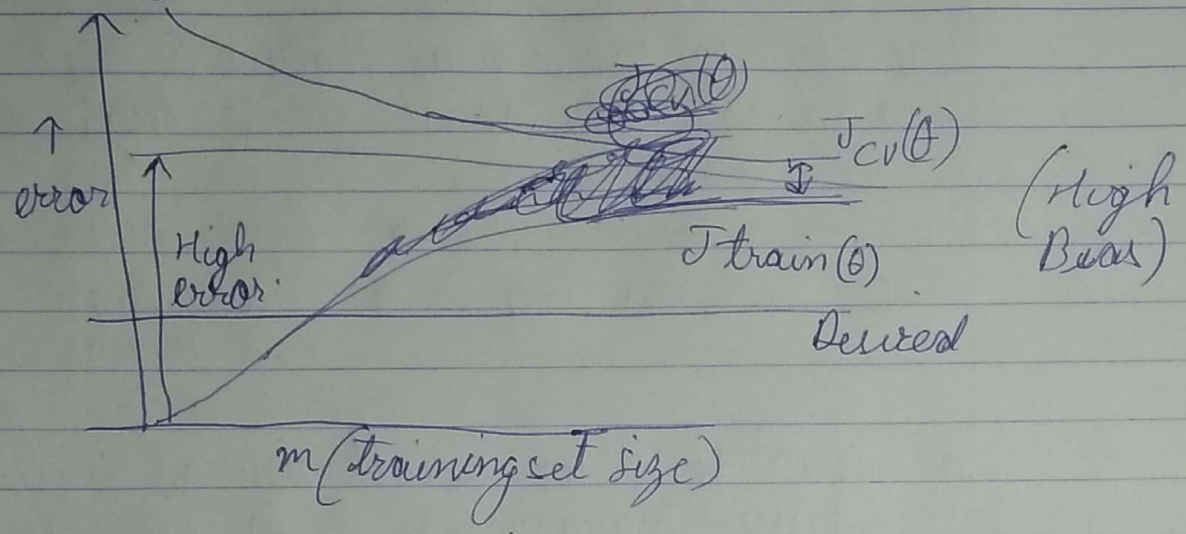
⇒ Large λ → High bias (underfitting) → $\lambda = 10000$

⇒ Small λ → High Variance (overfitting), $\lambda = 0$

⇒

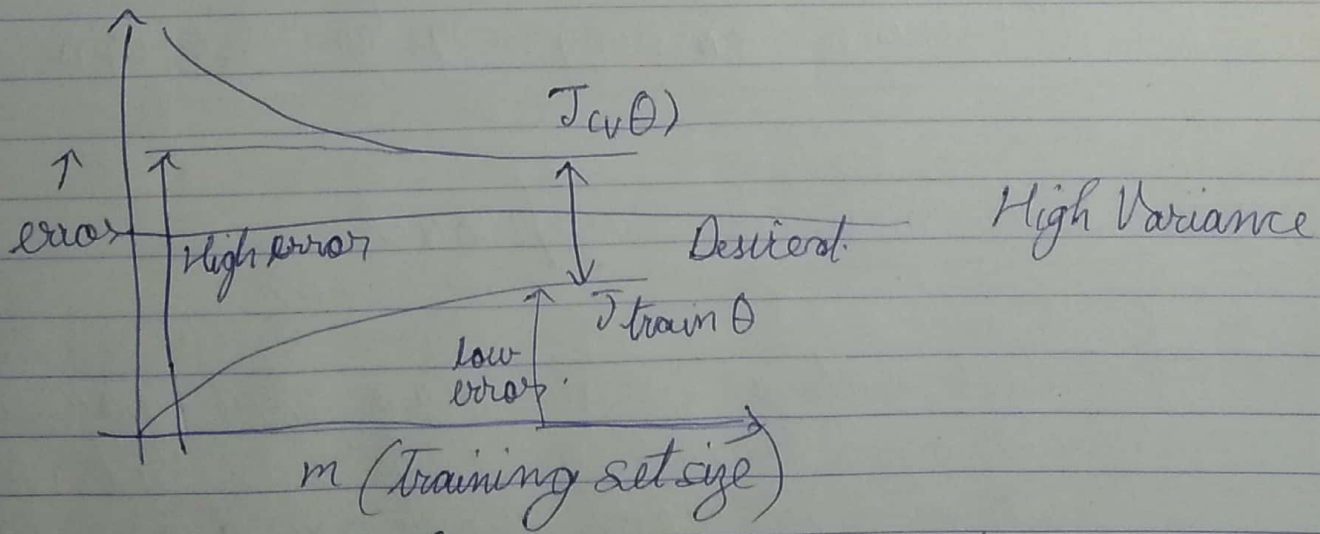


⇒ Learning curve



⇒ as training set size increases, the error in training set also increases. but the error in validation set decreases.

⇒ In High bias, the curve of $J_{CV}\theta$ & $J_{train}\theta$ will be close and their performance will be close (Both will be high) (Underfit)



⇒ In this case (High Variance), getting more training data will help in algo's performance

Fixes

1) Get more training examples → high variance
2) Try smaller set of features → high variance
3) Try getting additional features → high bias
4) Try adding polynomial " → high bias.
   (x_1^2, x_2^2, x_1x_2)
5) Try decreasing λ → fixes high bias.
6) Try increasing λ → fixes high variance
   ie. increases bias

→ So if we make as small neural network (fewer parameters & less hidden layers) → prone to underfitting

→ So if we make a large neural network, (more parameters; more prone to overfitting).
we can use regularization

Machine Learning System Design (W-6)

→ Recommended Approach

1) Start with a simple algo that you can implement quickly. Implement it and test it on your cross-validation test.

2) Plot learning curves to decide if more data, more features, etc are likely to help.

3) Error analysis → Manually examine the examples (in cross validation set) that your algo made errors on.
See if you spot any systematic trend in what type of examples it is making errors on.

⇒ Stemming software → lets you identify words that have same meaning.

⇒ Skewed classes → When the dataset has a huge amount of single classification cases and very little amount of other classification cases. So even if the algo only predicts for the class with higher cases, and ignores the other cases, it has a pretty good accuracy & low error.

⇒ Precision/Recall

		Actual Class 1	Actual Class 0
Predicted Class	1	True Positive	False Positive
	0	False Negative	True Negative

⇒ Precision → $\frac{\text{True positives}}{\text{predicted +ves}}$

$\Rightarrow \frac{\text{True +ves}}{\text{True +ves + False +ve}}$

⇒ Recall ⇒ $\frac{\text{True positive}}{\text{actual +ves}}$

$\Rightarrow \frac{\text{True positives}}{\text{True +ves + False -ves}}$

⇒ Accuracy = $\frac{\text{True positives} + \text{True negatives}}{\text{(total examples)}}$

⇒ Precision ⇒ Of all patients where we predicted $y=1$, what actually has cancer?

⇒ Recall ⇒ Of all patients that actually have cancer, what fraction did we detect as having cancer?

⇒ If an algo is getting high precision and high recall, then the algo is doing well.

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

→ Predict 1 if $h_\theta(x) \geq 0.7$
Predict 0 if $h_\theta(x) < 0.7$

↓

High precision, low recall.

→ Predict 1 if $h_\theta(x) \geq 0.3$
Predict 0 if $h_\theta(x) < 0.3$

↓

Low precision, high recall.

⇒ F_1 Score $= \frac{2RP}{P+R}$ (P = Precision, R = Recall)

⇒ When the dataset is very large, the training set is large which makes the algo unlikely to overfit.
∴ Its $J_{train}(\theta)$ will be small &
$J_{train}(\theta) \approx J_{test}(\theta)$