WEEK - 6

WEEK 6: Deciding what to try next

#1 Debugging a learning algorithm.

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

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

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

- → Get more training examples
- → Try smaller sets of features
- → Try getting additional features
- → Try adding polynomial features.
- → Try decreasing λ.
- → Try increasing λ

#2 Evaluating a hypothesis.

Overfitting: fail to generalize to new examples not in training set.

#) Data set → Training set (70%) : } randomly sorted.
Test set (30%) : }

So, in case of overfitting, $J(\theta)$ will be low and $J_{test}(\theta)$ will be high.

#) Training / testing procedure for LR / logistic regression :-

① learn parameter θ from training data ($J(\theta)$) → 70%.

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

③ $J_{test}(\theta) = -\frac{1}{m_t}\left[\sum_{i=1}^{m_t} y^i_{test} \log(h_\theta(x^i_{test})) + (1 - y^i_{test})(\log(h_\theta(x^i_{test})))\right]$

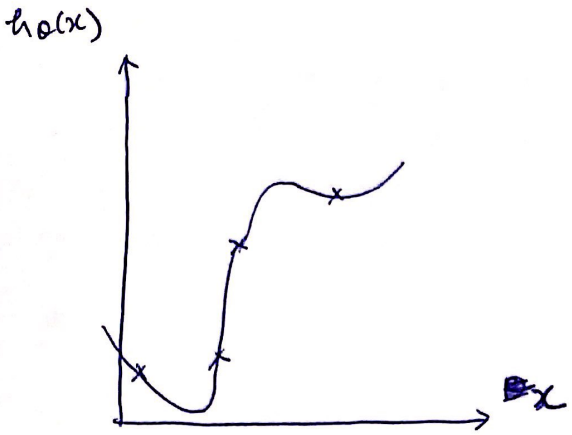
↓

Another metric : Misclassification error :-

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

$$\therefore \text{Test error} = \frac{1}{M_{test}} \sum_{i=1}^{M_{test}} err(h_\theta(x^i_{test}), y^i_{test})$$

Model selection and training/Validation/test sets.
(parameters / λ)



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

* Once parameters $\theta_0, \theta_1, \theta_2 .. \theta_4$ were fit to some set of data (training) set, the error of the parameters as measured on that data ($J(\theta)$) is likely to be lower than the actual generalization error.

(1) Model Selection.

degree of Polynomial

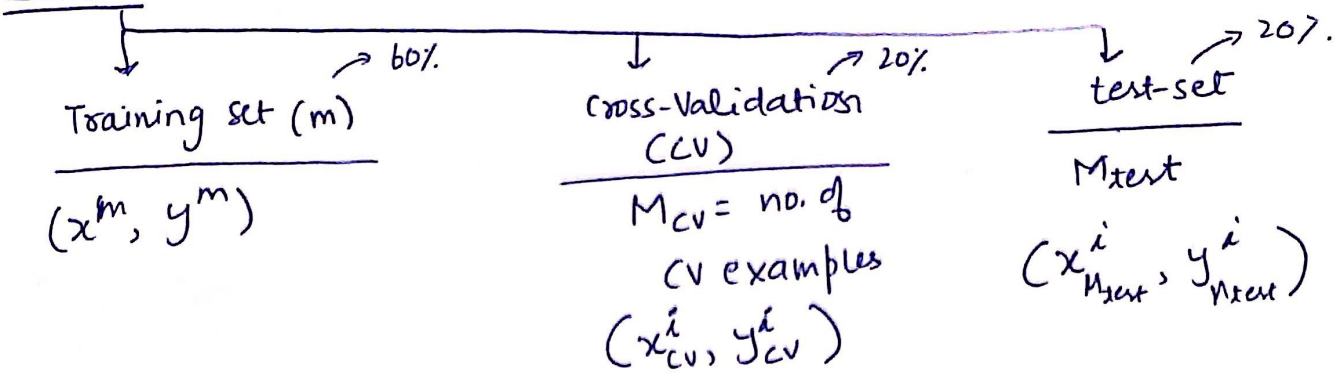
1. $h_\theta(x) = \theta_0 + \theta_1(x) \rightarrow J_{test}(\theta^1)$ ← d=1
2. $h_\theta(x) = \theta_0 + \theta_1 x + \theta_2 x^2 \rightarrow J_{test}(\theta^2)$
⋮
10. $h_\theta(x) = \theta_0 + \theta_1 x + \theta_2 x^2 + \dots \theta_{10} x^{10} \rightarrow J_{test}(\theta^{10})$

→ Measure the performance, $J_{test}(\theta^i)$

→ choose the ~~loose~~ lowest cost among the options.

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.

- Dataset



① Training error:

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

Optimize the parameters in θ using the training for each polynomial degree

② Cross-Validation error:

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

Find the polynomial degree d with least error

③ Test error

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

Estimate J using test set.

Model Selection:

→ compute, $J_{cv}(\theta)$, $J_{cv}(\theta^2)$ ... $J_{cv}(\theta^{10})$

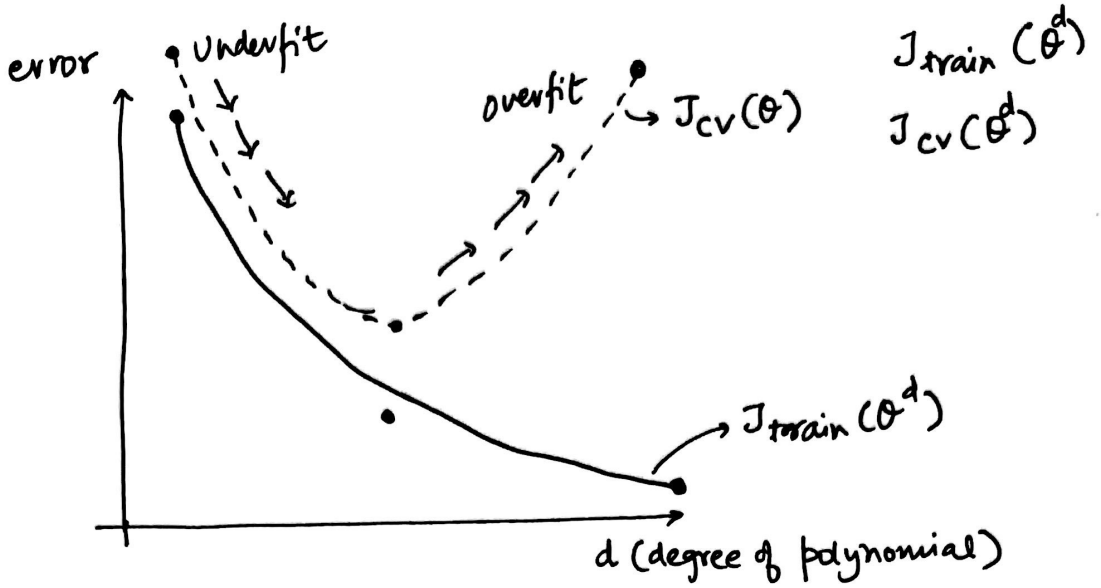
→ select lowest order poct. for inst. $J_{cv}(\theta^4)$.

→ Estimate generalization error for test set

$J_{test}(\theta^{(4)})$

BIAS Vs. VARIANCE

(Underfit) (overfit)



#: After having trained the data, for smaller d, $J_{cv}(\theta)$ may/will be high. As we go on increasing d, we may find a suitable d for which $J_{cv}(\theta)$ is low.

Bias: $J_{train}(\theta)$ will be high. $J_{cv}(\theta)$ will be high. } $J_{cv}(\theta) \approx J_{train}(\theta)$

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

Regularization & Bias/Variance

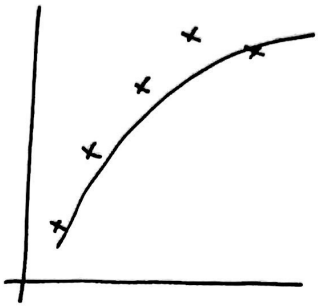
(problem of Overfitting)

Model: $h_\theta(x) = \theta_0^T x$. $\quad \theta \in \{\theta_0 \dots \theta_4\}$

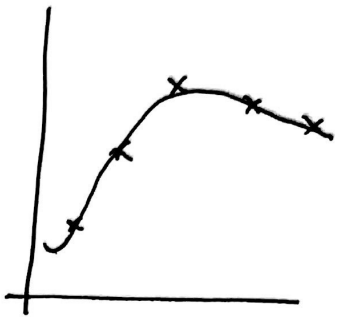
$$J(\theta) = \frac{1}{2m}\left[\sum_{i=1}^{m}(h_\theta(x)-y)^2 + \lambda\, \theta_j^2\right] \qquad j \in \{1, \dots n\}$$

λ = very large

(High Bias)



λ = just right



$\lambda = 0$

(High Variance)

#: How to choose λ:

① $\quad [\lambda = 0]$

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

②

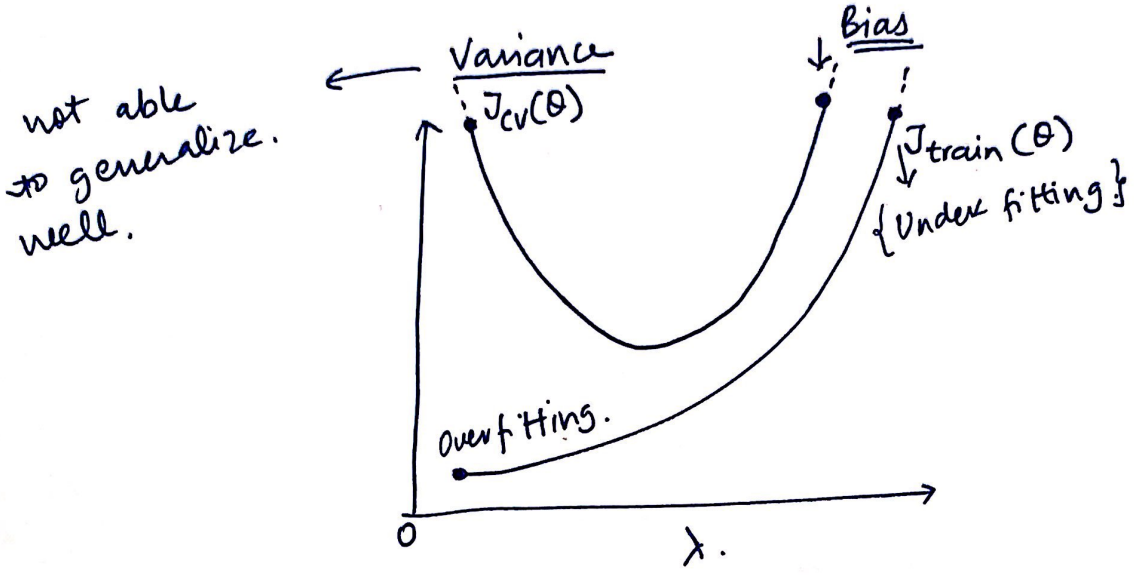
1. $\lambda = 0 \longrightarrow$ $\min_\theta J(\theta) \longrightarrow \theta^1 \longrightarrow J_{CV}(\theta^1)$
2. $\lambda = 0.01 \longrightarrow$ $\min (J(\theta)) \longrightarrow \theta^2 \longrightarrow$ ⋮
3. $\lambda = 0.02$
4. ⋮
12. $\lambda = 10 \longrightarrow$ $\longrightarrow \theta^{12} \longrightarrow J_{CV}(\theta^{12})$

$\longrightarrow$ lowest $J_{CV}(\theta^d)$ ex. (θ^5)

③ Pick θ^5. Test error : $J_{test}(\theta^5)$

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

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



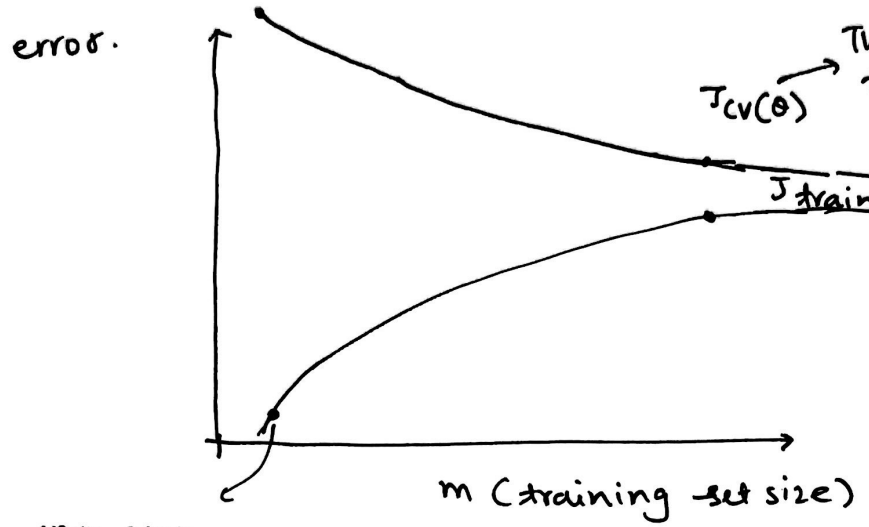
Steps to Model Selection:

1. Create a list of $\lambda \in \{0, 0.01, 0.02 \ldots 10.24\}$
2. Create a model ~~for~~ with different degrees
3. Iterate through the λ's & for each λ compute $\theta^{learned}$
4. Compute $J_{cv}(\theta^{learned})$ with $\lambda = 0$ with diff degrees.
5. Select lowest $J_{cv}(\theta^{learned})$.
6. Using the best θ & λ, apply it on the $J_{test}(\theta)$ to see if it has good generalization of the problem.

LEARNING CURVES

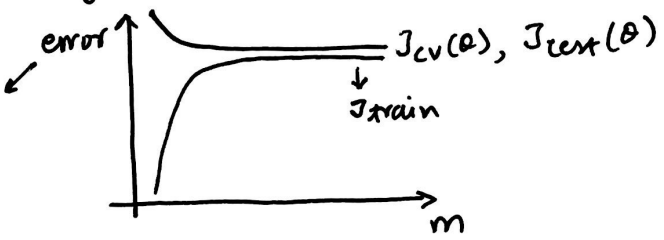
$$J_{train} = \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x) - y)^2$$

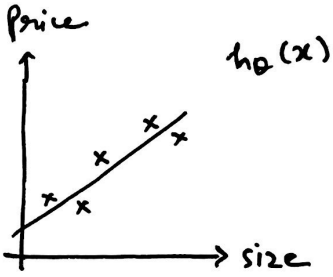
$$J_{CV} = \frac{1}{2m} \sum_{i=1}^{m_{cv}} (h_\theta(x^i_{cv}) - y^i_{cv})^2$$



very easy to fit the curve.

#1: High Bias:

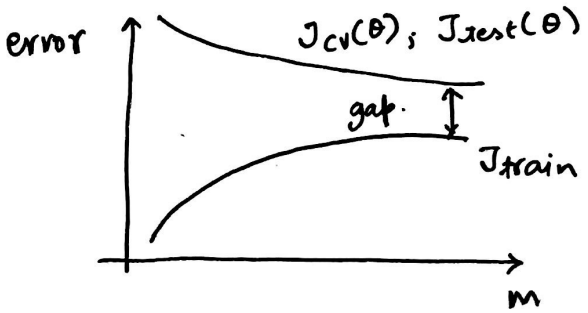


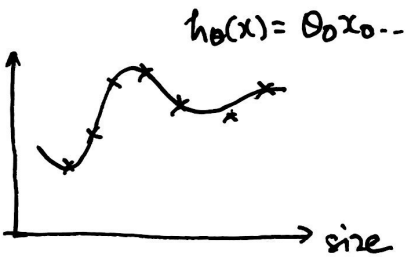


$h_\theta(x) = \theta_0 + \theta_1 x$

getting more training data will not (by itself) help much.

#2 High variance:





$h_\theta(x) = \theta_0 x_0 \ldots + \theta_{100} x^{100}$

getting more ~~data~~ training data is likely to help.

Debugging a learning Algorithm

① Get more training examples → fixes high variance.

② Try smaller sets of features → fixes high variance.

③ Try getting additional features. → fixes high bias

④ Try adding polynomial features → fixes high bias.

⑤ Try decreasing λ → fixes high bias.

⑥ Try increasing λ → fixes high variance

EXAMPLE : **BUILDING A SPAM CLASSIFIER**:
what to prioritize

Non-spam (0)
Spam (1)

① Supervised learning:

x = features of email $\quad y \in \{0, 1\}$

{choose 100 words indicative of spam / not spam}

Eg. deal, buy, discount, now, andrew, ...

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

***

Note: In practice, take most frequently occuring (n) words (10,000 to 50000) in training set, rather than manually pick 100 words

ERROR ANALYSIS

- Start with a simple algorithm that you can implement quickly. Implement it and test it on your CV data

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

- Error Analysis:

  Manually examine the examples that the algorithm made errors on (in your CV set). See if you spot any systematic trend in what type of examples it is making errors on. { if we develop new features by examining the test set then we may end up choosing features that work well wrt $J_{test}(\theta)$, and it no longer is a good est. }

  Ex:

  $m_{CV} = 500$

  error = 100 emails.

  ↙ what type of email it is?

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

- 'stemming' Software: discount/discounted/discounting ✓
  universe/university ✗

  CV error → { without stemming 5% ; with stemming 3% ✓ }

  [Real number evaluation Metric]

Handling Skewed data

Error metric for skewed classes.

① Precision / Recall:

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

Predicted class \ Actual Class.	1	0
1	True positive	False positive
0	False negative.	True negative

* Precision: of all patients where we predicted $y=1$, what fraction actually has cancer.

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

* Recall: of all the patients that actually have cancer, what fraction did we correctly detect as having cancer?

$$\left\{ \frac{\text{True positives}}{\#\text{ Actual positive}} = \frac{\text{True Posit}}{\text{True post False neg.}} \right\}$$

Trading off Precision & Recall :

→ Logistic regression : $0 \leq h_\theta(x) \leq 1 = \begin{cases} 1 & \text{if } h_\theta(x) \geq 0.5 \\ 0 & \text{if } h_\theta(x) \leq 0.5 \end{cases}$

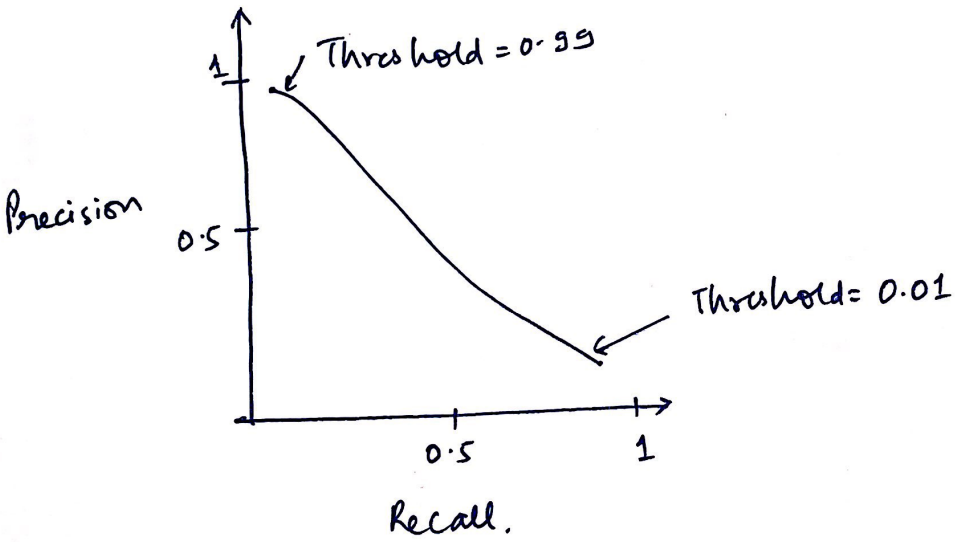
→ Suppose we want to predict $y=1$ only if very confident

① if we change, $h_\theta(x) \geq 0.7 \Rightarrow$ Higher precision., lower recall.

② Suppose we want to avoid missing too many cases of cancer (avoid false negative)

{ ↳ lower $h_\theta(x) < 0.3$ }

$\Rightarrow$ Higher recall, lower precision



F_1 Score: How to compare precision/recall numbers?

	(P)	(R)	Avg	F_1 Score
Algorithm 1	0.5	0.4	0.45	0.444
Algorithm 2	0.7	0.1	0.4	0.175
Algorithm 3	0.02	1.0	0.51	0.0392

Predict y=1 all the time.

* Avg. = $\frac{P+R}{2}$ (Not a good way)

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