Machine Learning - Week 6 - Advice for Applying Machine Learning

11 - Evaluating a Learning Algorithm

L1 - Deciding what to try Next

Debugging a learning algorithm:

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

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

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?

- Get more training examples
- Try smaller sets of features
- Try getting additional features
- Try adding polynomial features (x_1^2, x_2^2, x_1x_2, etc...)
- Try decreasing λ
- Try increasing λ

Machine learning diagnostic:

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

L2 - Evaluating a hypothesis

Dataset:	Size	Price	
	2104	400	
	1600	330	
	2400	369	
70%	1416	232	Training set →
	3000	540	
	1985	300	
	1534	315	
	1427	199	Test set →
30%	1380	212	
	1494	243	

Select Random examples

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

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

m_{test} = # of test examples

Training/testing procedure for linear regression
- Learn parameter θ from training data (minimize training error $J(\theta)$)
- Compute test set error:

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

Training/testing procedure for logistic regression
- Learn parameter θ from training data
- Compute test set error:

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

- Misclassification error (0/1 misclassification error):

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

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

L3 - Model Selection and training/validation/test sets

d = degree of polynomial

Model Selection

d=1	1. $h_\theta(x) = \theta_0 + \theta_1 x$	$\theta^{(1)}$	$J_{test}(\theta^{(1)})$
d=2	2. $h_\theta(x) = \theta_0 + \theta_1 x + \theta_2 x^2$	$\theta^{(2)}$	$J_{test}(\theta^{(2)})$
d=3	3. $h_\theta(x) = \theta_0 + \theta_1 x + \dots + \theta_3 x^3$	$\theta^{(3)}$	$J_{test}(\theta^{(3)})$
⋮	⋮	⋮	⋮
d=10	10. $h_\theta(x) = \theta_0 + \theta_1 x + \dots + \theta_{10} x^{10}$	$\theta^{(10)}$	$J_{test}(\theta^{(10)})$

Select min

Ex. Choose $\theta_0 + \dots + \theta_5 x^5$ (d=5)

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

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.

Evaluating your hypothesis

Dataset:

	Size	Price	
60%	2104	400	Training set
	1600	330	
	2400	369	
	1416	232	
	3000	540	
	1985	300	
20%	1534	315	Cross-validation set (CV)
	1427	199	
20%	1380	212	Test set
	1494	243	

Training set → $(x^{(1)}, y^{(1)}), \ldots, (x^{(m)}, y^{(m)})$

Cross-validation set → $(x_{cv}^{(1)}, y_{cv}^{(1)}), \ldots, (x_{cv}^{(m)}, y_{cv}^{(m)})$

Test set → $(x_{test}^{(1)}, y_{test}^{(1)}), \ldots, (x_{test}^{(m)}, y_{test}^{(m)})$

Train/Validation/Test error.

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

1. $h_\theta(x) = \theta_0 + \theta_1 x \rightarrow \min_\theta J(\theta) \rightarrow \theta^{(1)} \rightarrow J_{cv}(\theta^{(1)})$

⋮

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

e.g. Pick $\theta_0 + \theta_1 x_1 + \cdots + \theta_4 x^4$ $(d=4)$

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

T2- Bias vs Variance

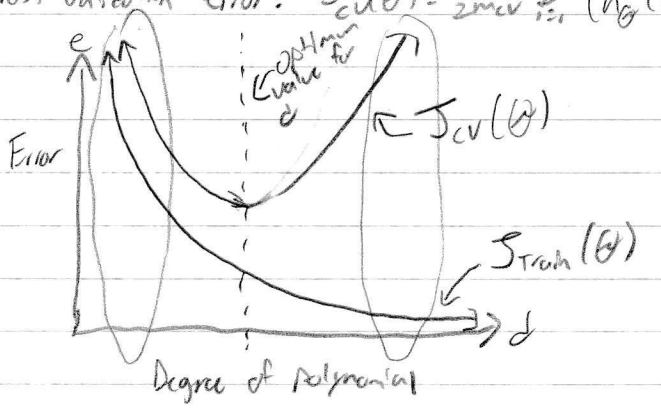
L1 - Diagnosing Bias vs Variance

Recall:

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



Suppose your learning algorithm is performing less well than you were hoping. ($J_{cv}(\theta)$ or $J_{test}(\theta)$ is high). Is it a bias problem or a variance problem?

Bias (underfit)
- $J_{train}(\theta)$ will be high
- $J_{cv}(\theta) \approx J_{train}(\theta)$

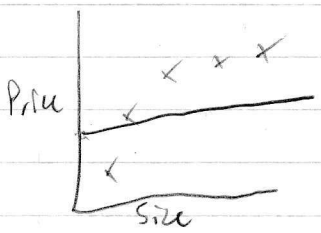
Variance (overfit)
- $J_{train}(\theta)$ will be low
- $J_{cv}(\theta) \gg J_{train}(\theta)$

L2 - Regularization and bias/variance

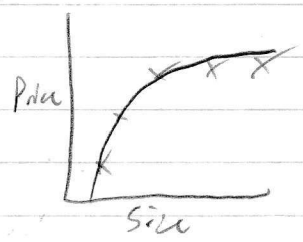
Linear regression with regularization

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

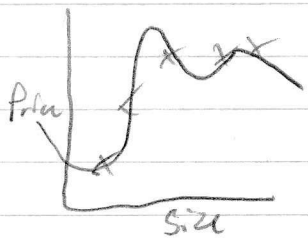
$$J(\theta) = \frac{1}{2m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2 + \frac{\lambda}{2m}\sum_{j=1}^{n}\theta_j^2$$



Large λ
High bias (underfit)
$\lambda = 10000$, $\theta_1 \approx 0, \theta_2 \approx 0, \ldots$
$h_\theta(x) \approx \theta_0$



Intermediate λ
"just right"



Small λ
High variance (overfit)
$\lambda = 0$

Choosing the regularization parameter λ

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

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

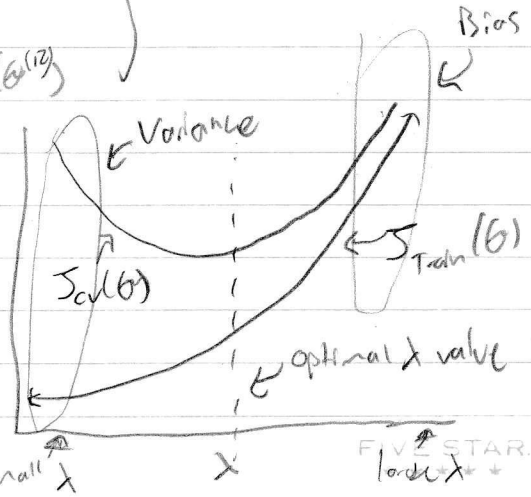
$$J_{test}(\theta) = \frac{1}{2m_{test}}\sum_{i=1}^{m_{test}}(h_\theta(x_{test}^{(i)}) - y_{test}^{(i)})^2$$

} Note there is no λ

1. Try $\lambda = 0 \rightarrow \min_\theta J(\theta) \rightarrow \theta^{(1)} \rightarrow J_{cv}(\theta^{(1)})$
2. Try $\lambda = 0.01 \rightarrow \min_\theta J(\theta) \rightarrow \theta^{(2)} \rightarrow J_{cv}(\theta^{(2)})$
3. Try $\lambda = 0.02 \rightarrow \min_\theta J(\theta) \rightarrow \theta^{(3)} \rightarrow J_{cv}(\theta^{(3)})$
4. Try $\lambda = 0.04 \rightarrow \min_\theta J(\theta) \rightarrow \theta^{(4)} \rightarrow J_{cv}(\theta^{(4)})$
⋮
12. Try $\lambda = 10 \rightarrow \min_\theta J(\theta) \rightarrow \theta^{(12)} \rightarrow J_{cv}(\theta^{(12)})$

} Select min

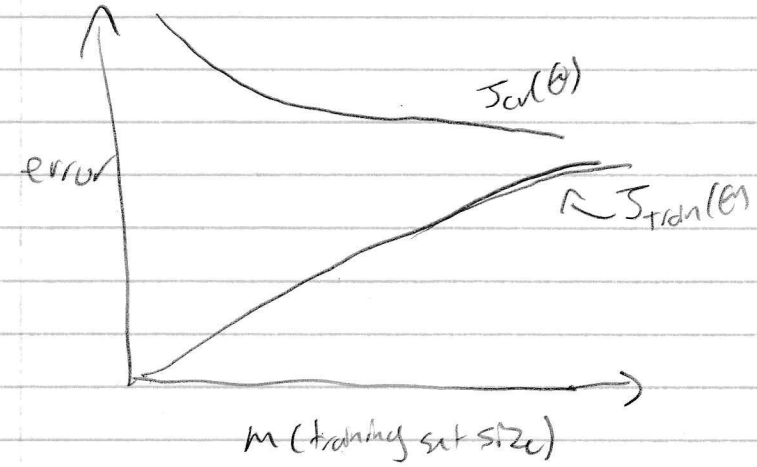
eg. Pick $\theta^{(5)}$. Test error: $J_{test}(\theta^{(5)})$



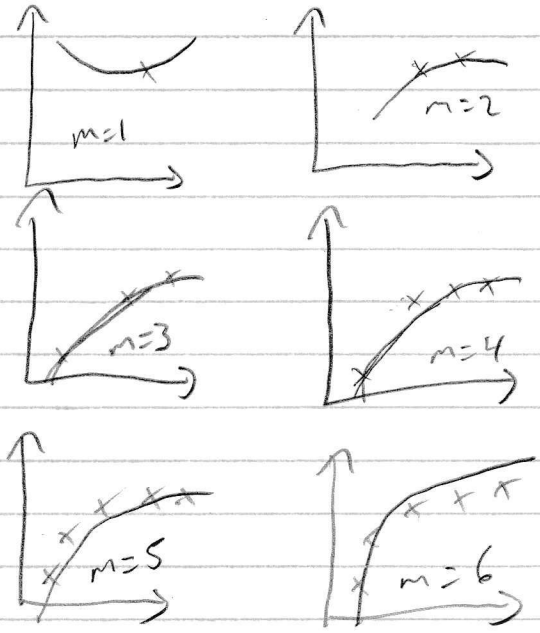
L3- Learning Curves

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

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

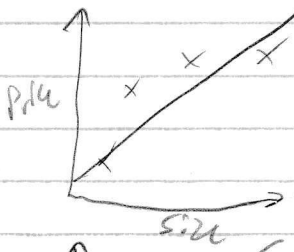


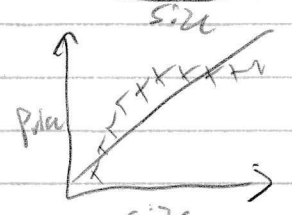
$h_\theta(x) = \theta_0 + \theta_1 x + \theta_2 x^2$



High Bias

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

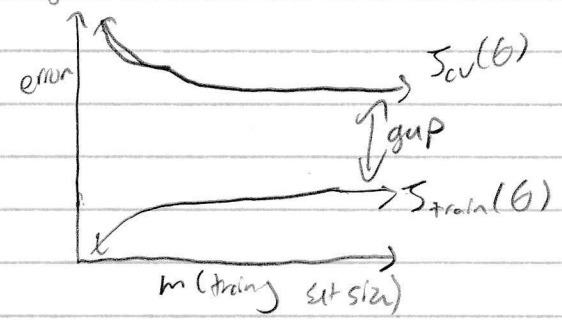




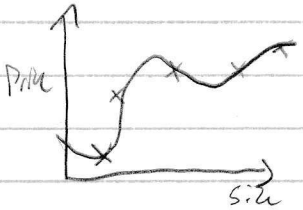
If a learning algorithm is suffering from high bias, getting more data will not (by itself) help much

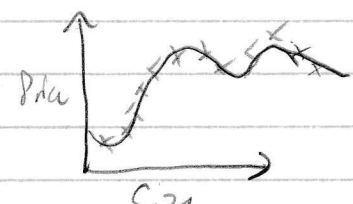
---

High Variance



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





If a learning algorithm is suffering from high variance, getting more training data is likely to help.

L4- Deciding what to try next (revisited)

Recall:

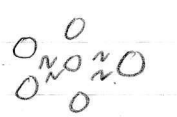
Suppose you have implemented regularized linear regression to predict housing prices. However, when you test your hypothesis in a new set of houses, you find that it makes unacceptably large errors in its predictions.

What should you try next?

- Get more training examples
  - → Fixes high variance
- Try smaller sets of features
  - → Fixes high variance
- Try to get additional features
  - → Fixes high bias
- Try adding polynomial features (x_1^2, x_2^2, x_1x_2, etc)
  - → Fixes high bias
- Try decreasing λ
  - → Fixes high bias
- Try increasing λ
  - → Fixes high variance

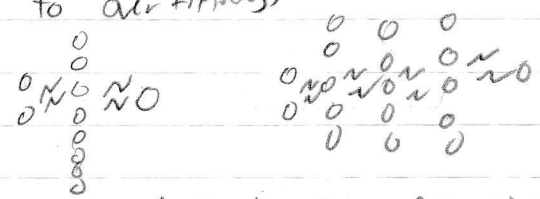
Neural Networks and Overfitting

"Small" nn (fewer parameters; more prone to underfitting)



- Computationally cheaper

"Large" nn (more parameters; more prone to overfitting)



- computationally more expensive
- Use regularization (λ) to address overfitting

Machine Learning - Week 6 - Machine Learning System Design

T3 - Building a Spam Classifier

L1 - Prioritizing what to work on

System Design Example:

Given a data set of emails, we could construct a vector for each email. Each entry in this vector represents a word. The vector normally contains 10 000 to 50 000 entries gathered by finding the most frequently used words in our data set. If a word is to be found in the email, we would assign its respective entry a 1, else that entry would be a 0. Once we have all our x vectors ready, we train our algorithm and finally, we could use it to classify if an email is spam or not.

Building a spam classifier.

Supervised learning. x = features of email. y = spam (1) or not spam (0)
Features x: Chose 100 words indicative of spam/not spam

So how could you spend your time to improve the accuracy of the classifier?

- Collect lots of data (e.g. honeypot project but doesn't always work)
- Develop sophisticated features (e.g. using email header data in spam emails).
- Develop algorithms to process your input in different ways (recognizing misspellings in spam).

It is difficult to tell which of the options will be most helpful.

L2 - 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 examples (in cross validation set) that your algorithm made errors on. See if you spot any systematic trend in what type of example it is making errors on.

T4 - Handling Skewed Data

L1 - Error Metrics for Skewed Classes

Cancer classification example

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

Find that you get 1% error on test set. (99% correct diagnoses)

But only 0.50% of patients have cancer.

↳ skewed classes.

```
function y = predictCancer(x)
    y=0; % ignore x!
return
```

} 0.5 % error

Precision/Recall

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

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

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

i.e. $\frac{\text{True positives}}{\text{True positives} + \text{False Positives}}$

← Predicted positives

High Precision and high recall is good

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

i.e. $\frac{\text{True positives}}{\text{True pos} + \text{False neg}}$

← Actual positives

L2 - Trading off Precision and Recall

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

Predict 1 if $h_\theta(x) \ge 0.5$

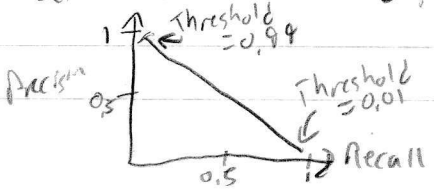
Predict 0 if $h_\theta(x) < 0.5$

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

- Could set a higher threshold, resulting in higher precision and lower recall

Suppose we want to avoid missing too many cases of cancer (avoid false negatives).

- Could set a lower threshold, resulting in higher recall and lower precision



F, Score (F score)

How to compare precision/recall numbers?

	Precision (P)	Recall (R)	Average	F, 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

Average: $\frac{P+R}{2}$

- Not very effective

F, Score: $2\frac{P \cdot R}{P+R}$

$P=0$ or $R=0 \Rightarrow$ F-Score $= 0$

$P=1$ and $R=1 \Rightarrow$ F-Score $= 1$

18 - Using Large Data Sets

11 - Data For Machine Learning

Large data rationale

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

Ex. For breakfast I ate ____ eggs. (two, too, or to)?

Counter Ex. Predict housing price from only size (feet^2) and no other features.

Useful test: Given the input x, can a human expert confidently predict y?

Use a learning algorithm with many parameters (eg. logistic regression/linear regression with many features; neural network with many hidden units). Low bias algorithms.

$\rightarrow J_{train}(\theta)$ will be small.

Use a very large training set (unlikely to overfit). Low variance

$\rightarrow J_{train}(\theta) \approx J_{test}(\theta)$

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