Week6

Table of Contents

Overview

How to systematically improving our learning algorithm.

how to tell when a learning algorithms is doing poorly, and describe the 'best practices' for how to 'debug' your learning algorithm and go about improving its performance.

Also covering machine learning system design. To optimize a machine learning algorithm, you will need to first understand the performance of machine learning system with multiple parts, and also how to deal with skewed data.

Evaluating a Learning hypothesis

What's Machine learning diagnostic:

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

A hypothesis may have a low error for the training examples but still be inaccurate (because of overfitting). Thus, to evaluate a hypothesis, given a dataset of training examples, we can split up the data into two sets: a **training set** and a **test set**. Typically, the training set consists of 70 % of your data and the test set is the remaining 30 %.

The new procedure using these two sets is then:

1. Learn Θ and minimize $J_{train}(\Theta)$ using the training set
2. Compute the test set error $J_{test}(\Theta)$

The test set error

1. For linear regression: $J_{test}(\Theta) = \frac{1}{2m_{test}} \sum_{i=1}^{2m_{test}} (h_\Theta(x_{test}^{(i)}) - y_{test}^{(i)})^2$
2. For classification ~ Misclassification error (aka 0/1 misclassification error):

$$err(h\Theta(x), y) = \begin{cases} 1 : if\ h_\Theta \geq 0.5\, and\, h_\Theta\ \leq\ 0.5\, and\, y\ =\ 1 \\ 0 :\ otherwise \end{cases}$$

This gives us a binary 0 or 1 error result based on a misclassification. The average test error for the test set is:

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

This gives us the proportion of the test data that was misclassified.

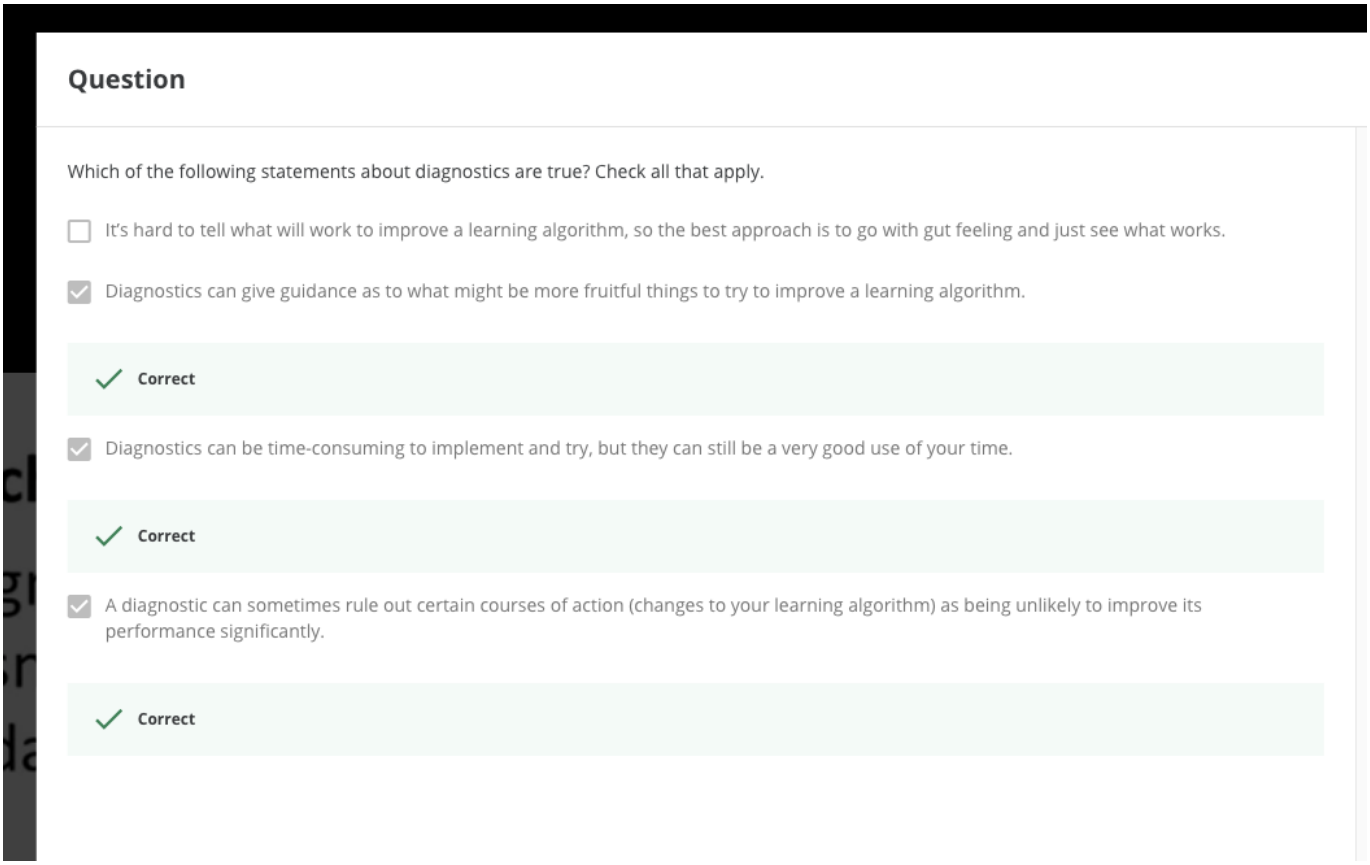
Question:

Question

Suppose an implementation of linear regression (without regularization) is badly overfitting the training set. In this case, we would expect:

- (●) The training error $J(\theta)$ to be **low** and the test error $J_{\text{test}}(\theta)$ to be **high**
- ( ) The training error $J(\theta)$ to be **low** and the test error $J_{\text{test}}(\theta)$ to be **low**
- ( ) The training error $J(\theta)$ to be **high** and the test error $J_{\text{test}}(\theta)$ to be **low**
- ( ) The training error $J(\theta)$ to be **high** and the test error $J_{\text{test}}(\theta)$ to be **high**

✓ **Correct**



Model Selection and Train/Validation/test sets

- A learning algorithm fits a training set well, that does not mean it's a good hypothesis. It could over fit and as a result your predictions on the test set would be poor. The error of your hypothesis as measured on the data set with which trained the parameters will be lower than the error on any other data set.
- How to systematic approach to identify the 'best' function between many models with different polynomial degrees. You can test each degree of polynomial and look at the error result.
- One way to break down our dataset into the three sets is :
  - Training set: 60%
  - Cross validation set: 20%
  - Test set: 20%
- **We can calculate three separate error values for the three different sets using the following method:**
  - Optimize the parameters in Θ using the training set for each polynomial degree.
  - Find the polynomial degree d with the least error using the cross validation set.
  - Estimate the generalization error using the test set with $J(\Theta^d)$,(d = theta from polynomial with lower error);

This way, the degree of the polynomial d has not been training using the test set.

Question:

Consider the model selection procedure where we choose the degree of polynomial using a cross validation set. For the final model (with parameters θ), we might generally expect $J_{\text{CV}}(\theta)$ To be lower than $J_{\text{test}}(\theta)$ because:

- (●) An extra parameter (d, the degree of the polynomial) has been fit to the cross validation set.
- ( ) An extra parameter (d, the degree of the polynomial) has been fit to the test set.
- ( ) The cross validation set is usually smaller than the test set.
- ( ) The cross validation set is usually larger than the test set.

✓ **Correct**

Bias vs. Variance[key]

Diagnosing Bias vs. Variance

if you run a learning algorithms and it doesn't do as your hoping, it will because you have either a high bias problem or a high variance problem.[underfiting or overfiting]

This section we examine the relationship between the degree of the polynomial d and the underfitting of our hypothesis.

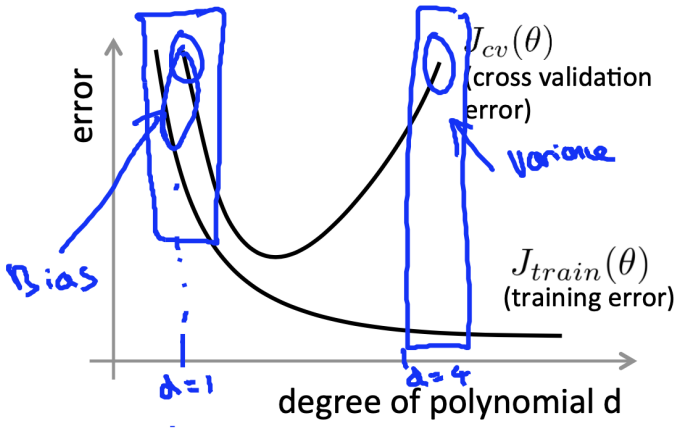
- We need to distinguish whether bias or variance is the problem contributing to bas predictions.
- **High bias is underfitting and high variance is overfitting**. Ideally, we need to find a golden mean between these two.

The training error will tend to decrease as we increase the degree d of the polynomial.

At the same time, the cross validation error will tend to decrease as we increase d up to a point, and then it will increase as d is increased.

Diagnosing bias vs. variance

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

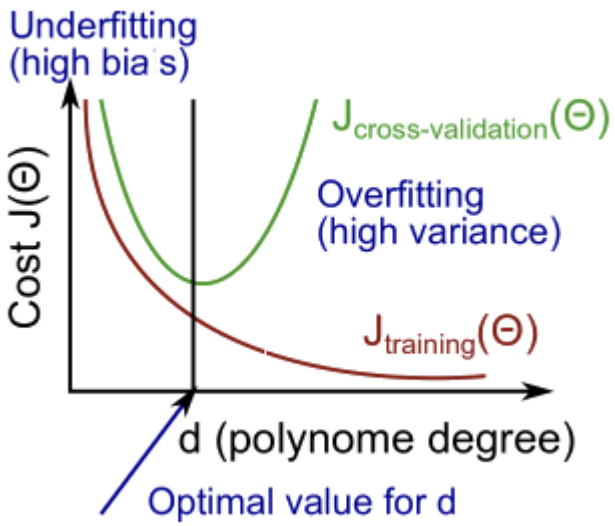
$J_{cv}(\Theta) >> J_{train}(\Theta)$

>>

High bias(underfitting): both $J_{train}(\Theta)$ and $J_{CV}(\Theta)$ will be high. Also, $J_{CV}(\Theta) \approx J_{train}(\Theta)$

High variance(overfitting): $J_{train}(\Theta)$ will be low and $J_{CV}(\Theta)$ will be much greater than $J_{train}(\Theta)$。

Summarized in figure below:



Question:

Suppose you have a classification problem. The (misclassification) error is defined as $\frac{1}{m}\sum_{i=1}^{m} \text{err}(h_\theta(x^{(i)}), y^{(i)})$, and the cross validation (misclassification) error is similarly defined, using the cross validation examples $(x_{\text{cv}}^{(1)}, y_{\text{cv}}^{(1)}), \ldots, (x_{\text{cv}}^{(m_{\text{cv}})}, y_{\text{cv}}^{(m_{\text{cv}})})$. Suppose your training error is 0.10, and your cross validation error is 0.30. What problem is the algorithm most likely to be suffering from?

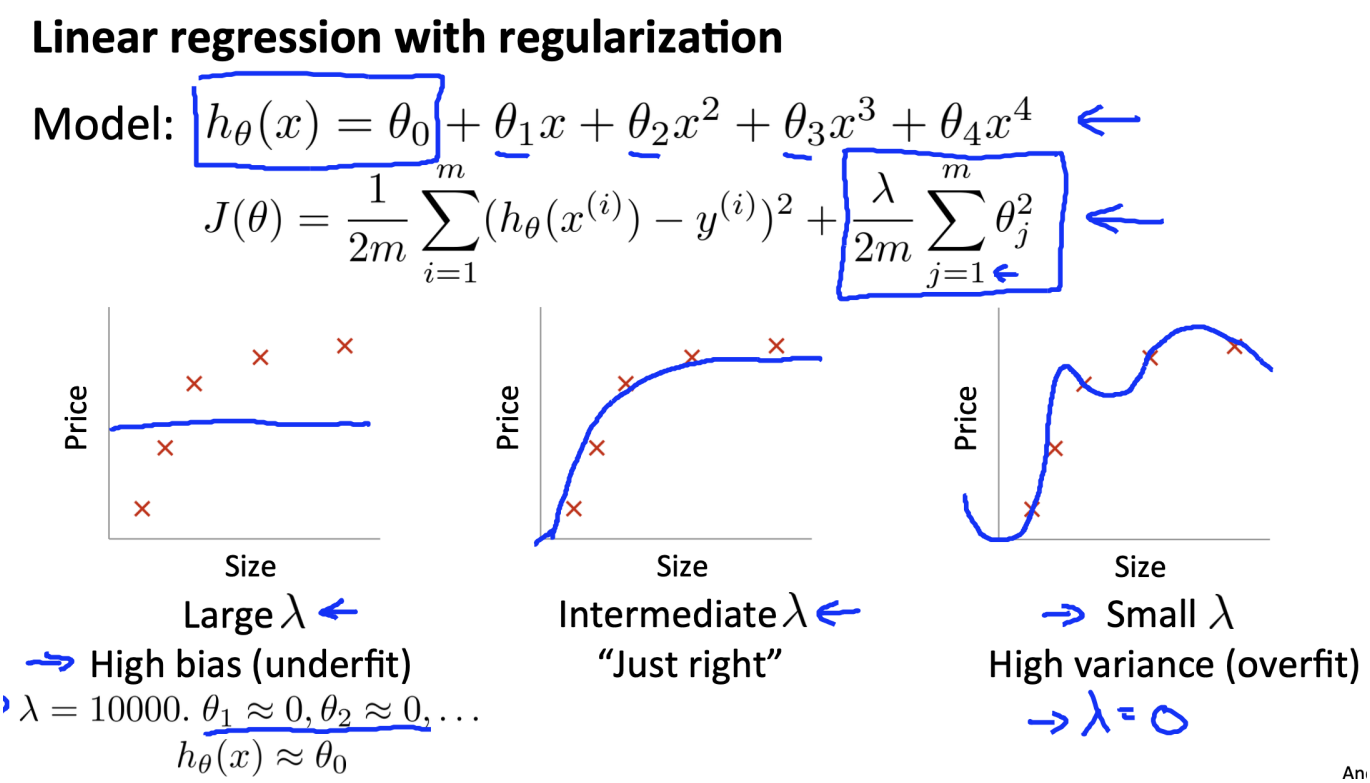
- ○ High bias (overfitting)
- ○ High bias (underfitting)
- ◉ High variance (overfitting)
- ○ High variance (underfitting)

✓ Correct

Exp: The cross-validation error is much greater than train validation.

Regularization and Bias/Variance

Note: This section will go deeper into bias and variances. also talk about how it interacts with and is affected by the regularization of learning algorithm.



As λ increases, our fit becomes more rigid. As λ approaches 0, we tend to overfit the data.

The point is: how do we choose our parameter λ to get it 'just right'.

1. Create a list of lambdas (i.e. λ∈{0,0.01,0.02,0.04,0.08,0.16,0.32,0.64,1.28,2.56,5.12,10.24});
2. Create a set of models with different degrees or any other variants.
3. Iterate through the λ s and for each λ go through all the models to learn some Θ.
4. Compute the cross validation error using the learned Θ (computed with λ) on the $J_{CV}(\Theta)$ **without** regularization or λ = 0.
5. Select the best combo that produces the lowest error on the cross validation set.
6. Using the best combo Θ and λ, apply it on $J_{test}(\Theta)$ to see if it has a good generalization of the problem.

Choosing the regularization parameter λ

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

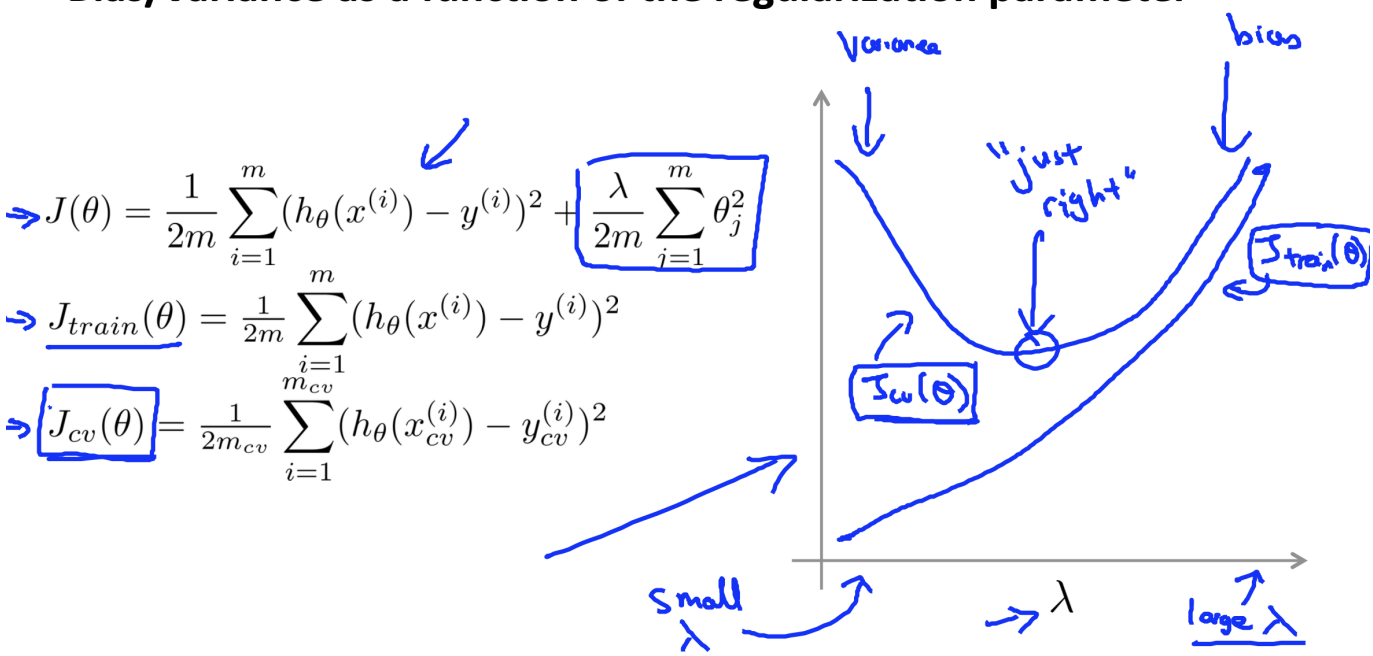
$$J(\theta) = \frac{1}{2m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2 + \frac{\lambda}{2m}\sum_{j=1}^{m}\theta_j^2$$

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

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

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

Bias/variance as a function of the regularization parameter λ



Learning Curves

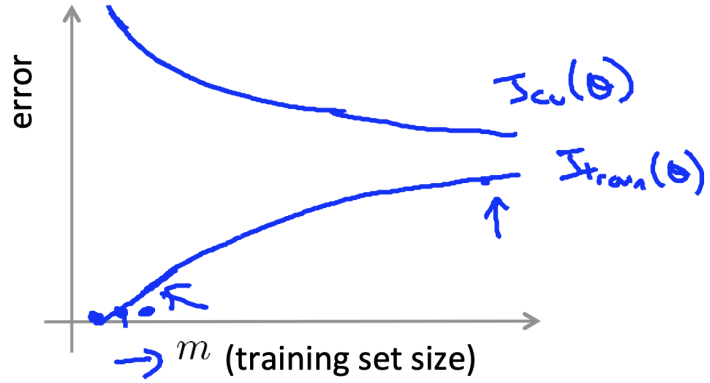
Learning curves is often a very useful thing to plot. If either want to sanity check that you algorithm is working correctly. or if you want to improve the performance of algorithm.

Also, learning curves is a tool to diagnose if a physical learning algorithms maybe suffering from bias, sort of variance problem or a bit of both.

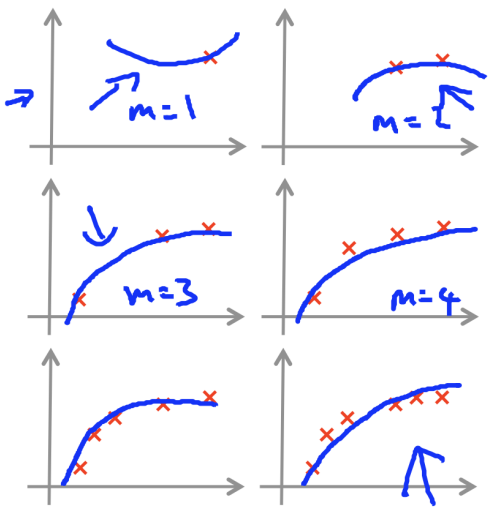
Learning curves

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

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



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



Training an algorithm on a very few number of data points will easily have 0 errors because we can always find quadratic curve that touches exactly those number of points. Hence:

- As the training set gets larger, the error for a quadratic function increases.
- The error value will plateau out after a certain m, or training set size.

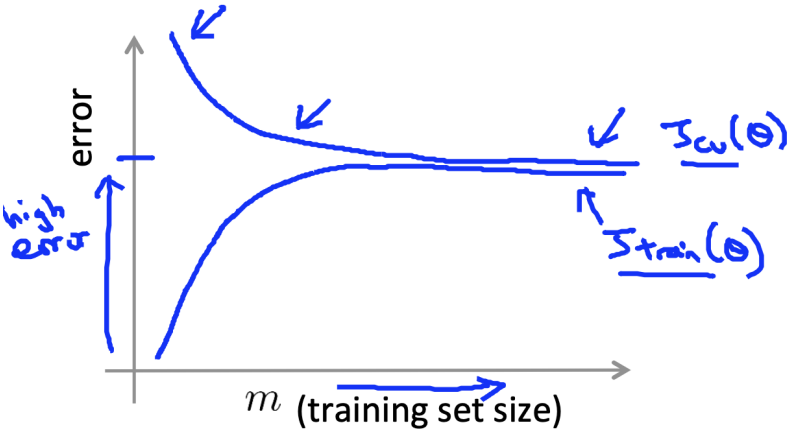
Experiencing high bias:

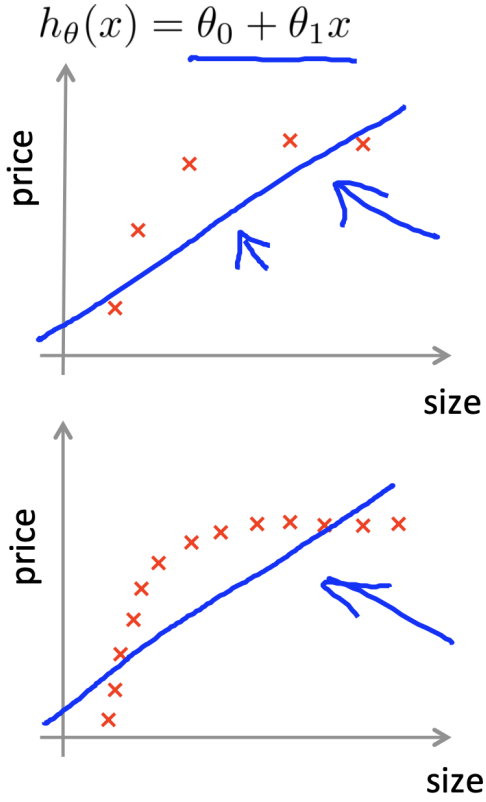
Low training set size: causes $J_{train}(\Theta)$ to be low and J_{CV} to be high.

Large training set size: causes both $J_{train}(\Theta)$ and $J_{CV}(\Theta)$ to be high with $J_{train}(\Theta) \approx J_{CV}(\Theta)$.

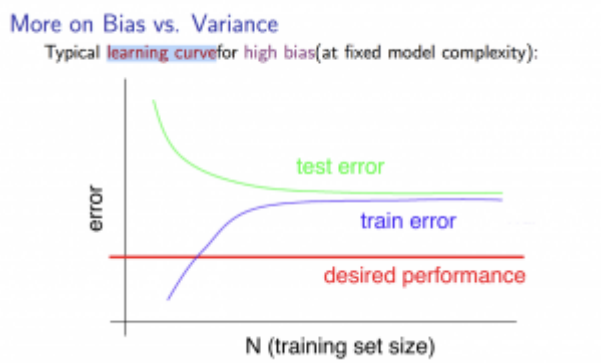
- if a learning algorithm is suffering from high bias, getting more training data will not help much.]

High bias





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



Experiencing high variance:

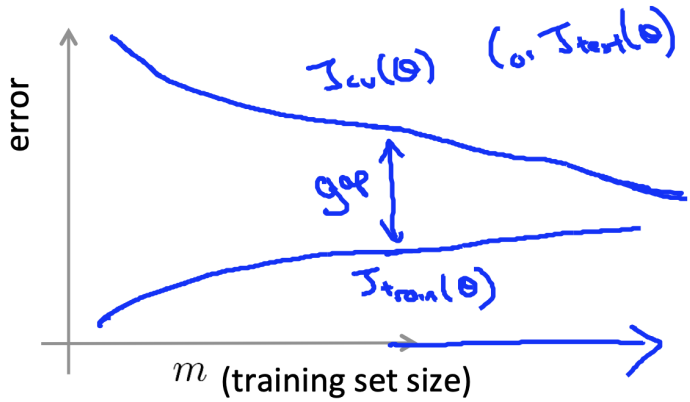
Low training set size: $J_{train}(\Theta)$ will be low and $J_{CV}(\Theta)$ will be high.

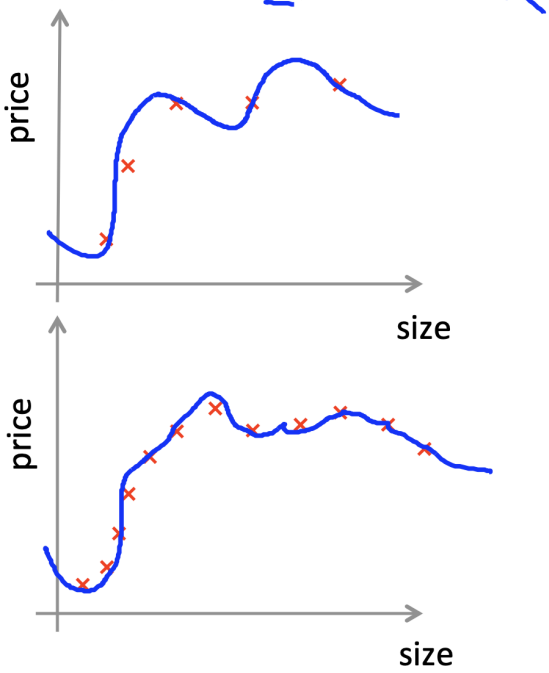
Large training set size: $J_{train}(\Theta)$ increases with training set size and $J_{CV}(\Theta)$ continues to decrease without leveling off. Also, $J_{train}(\Theta) < J_{CV}(\Theta)$ but the difference between them remains significant.

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

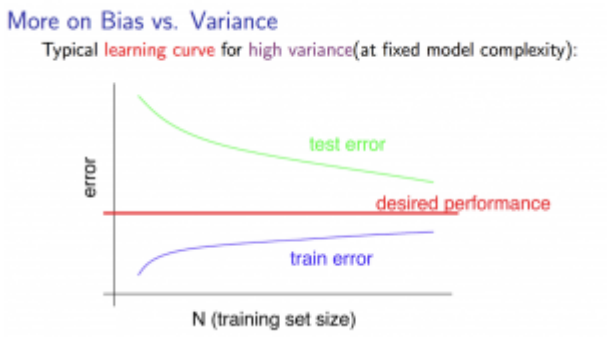
High variance

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





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



Question:

In which of the following circumstances is getting more training data likely to significantly help a learning algorithm's performance?

- [ ] Algorithm is suffering from high bias.
- [x] Algorithm is suffering from high variance.

✓ **Correct**

- [x] $J_{\text{CV}}(\theta)$ (cross validation error) is much larger than $J_{\text{train}}(\theta)$ (training error).

✓ **Correct**

- [ ] $J_{\text{CV}}(\theta)$ (cross validation error) is about the same as $J_{\text{train}}(\theta)$ (training error).

Deciding What to Do Next Revisited

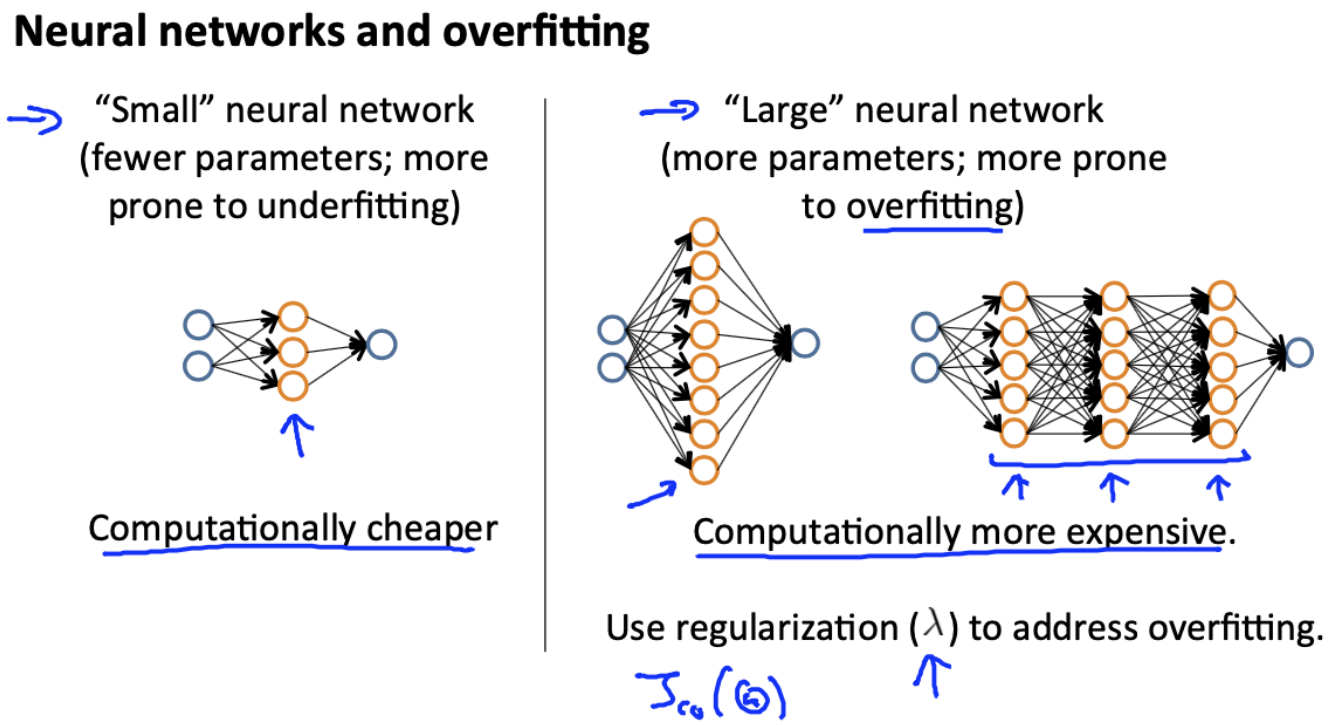
Our decision process can be broken down as follows:

- **Getting more training examples:** Fixes high variance
- **Trying smaller sets of features:** Fixes high variance
- **Adding features:** Fixes high bias
- **Adding polynomial features:** Fixes high bias
- **Decreasing λ:** Fixes high bias
- **Increasing λ:** Fixes high variance.

Diagnosing Neural Networks

- A neural network with fewer parameters is prone to underfitting. It's also **computationally cheaper.**
- A large neural network with more parameters is prone to overfitting. It's also **computationally expensive**. In this case you can use regularization (Increase λ) to address the overfitting.

Using a single hidden layer is a good starting default. You can train your neural network on a number of hidden layers using your cross validation set. You can then select the one that performs best.

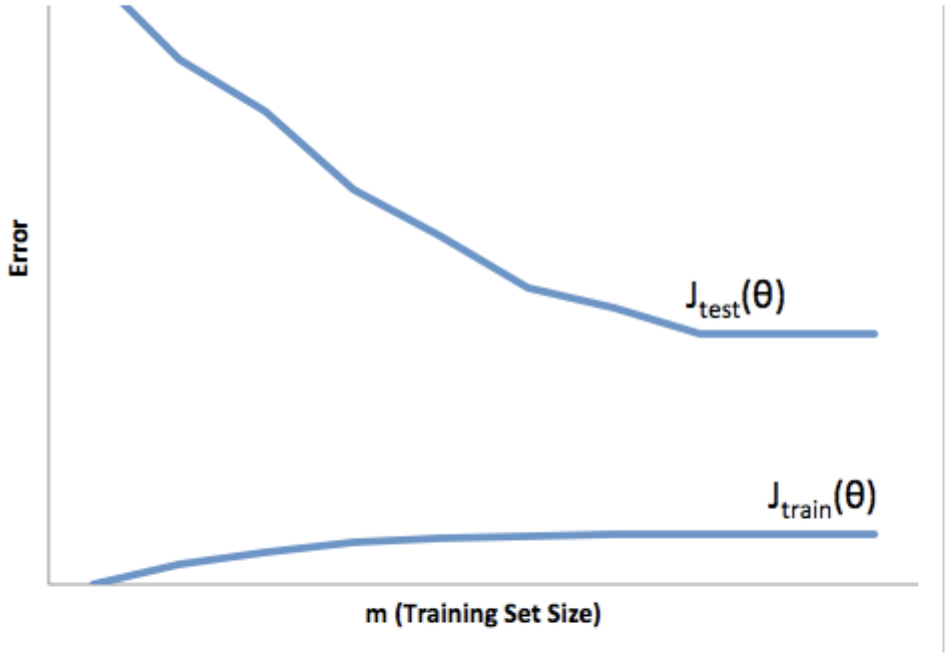


Model Complexity Effects:

- Lower-order polynomials (low model complexity) have high bias and low variance. In this case, the model fits poorly consistently.
- Higher-order polynomials (high model complexity) fit the training data extremely well and the test data extremely poorly. These have low bias on the training data, but very high variance.
- In reality, we would want to choose a model somewhere in between, that can generalize well but also fits the data reasonably well.

Quiz - Advice for applying Machine Learning

You train a learning algorithm, and find that it has unacceptably high error on the test set. You plot the learning curve, and obtain the figure below. Is the algorithm suffering from high bias, high variance, or neither?



- ○ High bias
- ◉ High variance
- ○ Neither

Suppose you have implemented regularized logistic regression to classify what object is in an image (i.e., to do object recognition). However, when you test your hypothesis on a new set of images, you find that it makes unacceptably large errors with its predictions on the new images. However, your hypothesis performs **well** (has low error) on the training set. Which of the following are promising steps to take? Check all that apply.

- [ ] Try adding polynomial features.
- [x] Get more training examples.
- [ ] Use fewer training examples.
- [x] Try using a smaller set of features.

Suppose you have implemented regularized logistic regression to predict what items customers will purchase on a web shopping site. However, when you test your hypothesis on a new set of customers, you find that it makes unacceptably large errors in its predictions. Furthermore, the hypothesis performs **poorly** on the training set. Which of the following might be promising steps to take? Check all that apply.

- [ ] Try increasing the regularization parameter λ.
- [x] Try adding polynomial features.
- [x] Try to obtain and use additional features.
- [ ] Try using a smaller set of features.

Which of the following statements are true? Check all that apply.

- [x] A typical split of a dataset into training, validation and test sets might be 60% training set, 20% validation set, and 20% test set.
- [x] Suppose you are using linear regression to predict housing prices, and your dataset comes sorted in order of increasing sizes of houses. It is then important to randomly shuffle the dataset before splitting it into training, validation and test sets, so that we don't have all the smallest houses going into the training set, and all the largest houses going into the test set.
- [ ] Suppose you are training a logistic regression classifier using polynomial features and want to select what degree polynomial (denoted d in the lecture videos) to use. After training the classifier on the entire training set, you decide to use a subset of the training examples as a validation set. This will work just as well as having a validation set that is separate (disjoint) from the training set.
- [ ] It is okay to use data from the test set to choose the regularization parameter λ, but not the model parameters (θ).

Which of the following statements are true? Check all that apply.

- [x] If a learning algorithm is suffering from high variance, adding more training examples is likely to improve the test error.
- [x] When debugging learning algorithms, it is useful to plot a learning curve to understand if there is a high bias or high variance problem.
- [x] If a learning algorithm is suffering from high bias, only adding more training examples may **not** improve the test error significantly.
- [ ] We always prefer models with high variance (over those with high bias) as they will able to better fit the training set.

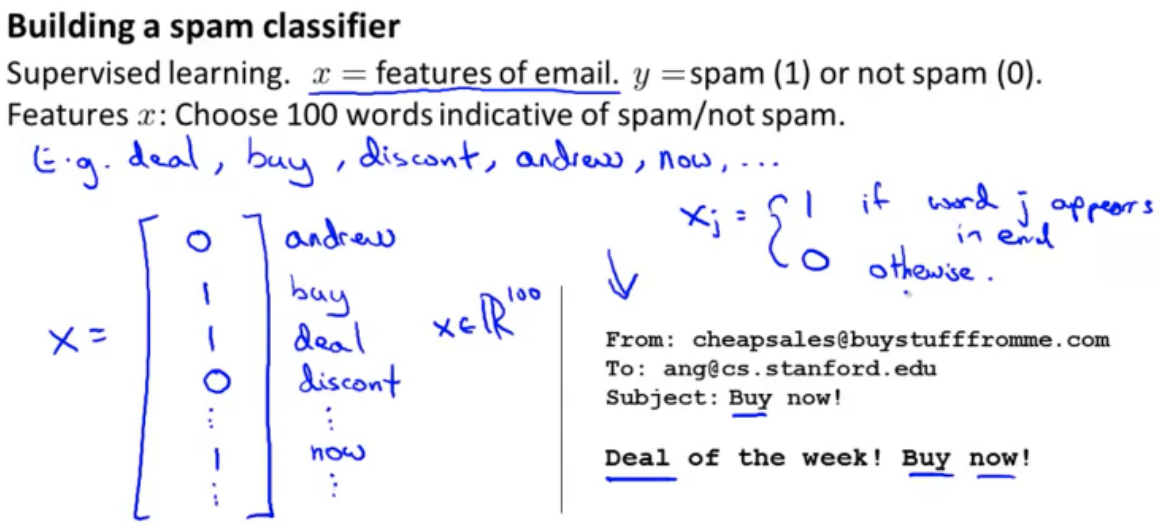
* Build a Spam Classifier

This video will touch on the main issues that you may face when designing a complex machine learning system.

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 if it is not found, 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 a spam or not.



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

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

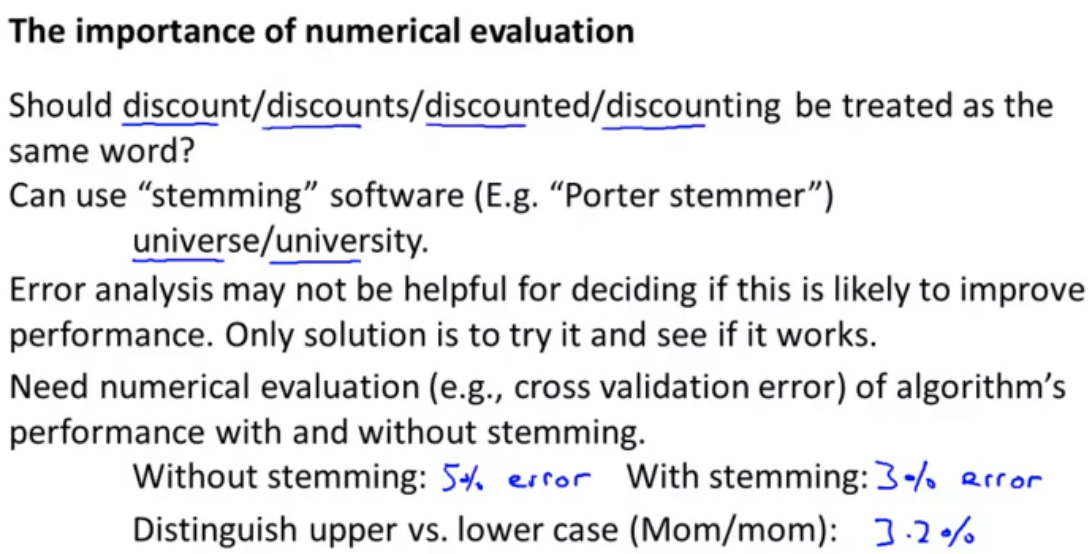
It is difficult to tell which of the options will be most helpful.

Error Analysis

The recommended approach to solving machine learning problems is to:

- Start with a simple algorithm, implement it quickly, and test it early on your cross validation data.
- Plot learning curves to decide if more data, more features, etc. are likely to help.
- Manually examine the errors on examples in the cross validation set and try to spot a trend where most of the errors were made.

For example, assume that we have 500 emails and our algorithm misclassifies a 100 of them. We could manually analyze the 100 emails and categorize them based on what type of emails they are. We could then try to come up with new cues and features that would help us classify these 100 emails correctly. Hence, if most of our misclassified emails are those which try to steal passwords, then we could find some features that are particular to those emails and add them to our model. We could also see how classifying each word according to its root changes our error rate:



It is very important to get error results as a single, numerical value. Otherwise it is difficult to assess your algorithm's performance. For example if we use stemming, which is the process of treating the same word with different forms (fail/failing/failed) as one word (fail), and get a 3% error rate instead of 5%, then we should definitely add it to our model. However, if we try to distinguish between upper case and lower case letters and end up getting a 3.2% error rate instead of 3%, then we should avoid using this new feature. Hence, we should try new things, get a numerical value for our error rate, and based on our result decide whether we want to keep the new feature or not.

Quiz: Machine Learning System Design

Coursera: Machine Learning (Week 6) Quiz - Machine Learning System Design | Andrew NG

1. You are working on a spam classification system using regularized logistic regression. "Spam" is a positive class (y = 1) and "not spam" is the negative class (y = 0). You have trained your classifier and there are m = 1000 examples in the cross-validation set. The chart of predicted class vs. actual class is: 1 point

	Actual Class: 1	Actual Class: 0
Predicted Class: 1	85	890
Predicted Class: 0	15	10

For reference:

- Accuracy = (true positives + true negatives) / (total examples)
- Precision = (true positives) / (true positives + false positives)
- Recall = (true positives) / (true positives + false negatives)
- F_1 score = (2 * precision * recall) / (precision + recall)

What is the classifier's recall (as a value from 0 to 1)?

Enter your answer in the box below. If necessary, provide at least two values after the decimal point.

0.85

NOTE:

Accuracy = (85 + 10) / (1000) = 0.095

Precision = (85) / (85 + 890) = 0.087

Recall = There are 85 true positives and 15 false negatives, so recall is 85 / (85 + 15) = 0.85.

F1 Score = (2 * (0.087 * 0.85)) / (0.087 + 0.85) = 0.16

2. Suppose a massive dataset is available for training a learning algorithm. Training on a lot of data is likely to give good performance when two of the following conditions hold true. 1 point

Which are the two?

- [ ] When we are willing to include high order polynomial features of x (such as x_1^2, x_2^2, $x_1 x_2$, etc.).
- [x] We train a learning algorithm with a large number of parameters (that is able to learn/represent fairly complex functions).
- [ ] We train a learning algorithm with a small number of parameters (that is thus unlikely to overfit).
- [x] The features x contain sufficient information to predict y accurately. (For example, one way to verify this is if a human expert on the domain can confidently predict y when given only x).

3. Suppose you have trained a logistic regression classifier which is outputting $h_\theta(x)$. 1 point

Currently, you predict 1 if $h_\theta(x) \geq \text{threshold}$, and predict 0 if $h_\theta(x) < \text{threshold}$, where currently the threshold is set to 0.5.

Suppose you **decrease** the threshold to 0.3. Which of the following are true? Check all that apply.

- [ ] The classifier is likely to have unchanged precision and recall, but higher accuracy.
- [x] The classifier is likely to now have higher recall.
- [ ] The classifier is likely to have unchanged precision and recall, but lower accuracy.
- [ ] The classifier is likely to now have higher precision.

4. Suppose you are working on a spam classifier, where spam emails are positive examples ($y = 1$) and non-spam emails are negative examples ($y = 0$). You have a training set of emails in which 99% of the emails are non-spam and the other 1% is spam. Which of the following statements are true? Check all that apply.

- [ ] If you always predict non-spam (output $y = 0$), your classifier will have 99% accuracy on the training set, but it will do much worse on the cross validation set because it has overfit the training data.
- [ ] If you always predict non-spam (output $y = 0$), your classifier will have 99% accuracy on the training set, and it will likely perform similarly on the cross validation set.
- [x] A good classifier should have both a high precision and high recall on the cross validation set.
- [x] If you always predict non-spam (output $y = 0$), your classifier will have an accuracy of 99%.

5. Which of the following statements are true? Check all that apply. 1 point

- [x] The "error analysis" process of manually examining the examples which your algorithm got wrong can help suggest what are good steps to take (e.g., developing new features) to improve your algorithm's performance.
- [ ] It is a good idea to spend a lot of time collecting a **large** amount of data before building your first version of a learning algorithm.
- [x] Using a **very large** training set makes it unlikely for model to overfit the training data.
- [ ] After training a logistic regression classifier, you **must** use 0.5 as your threshold for predicting whether an example is positive or negative.
- [ ] If your model is underfitting the training set, then obtaining more data is likely to help.

Summary with Chinese

中文整理：

神经网络与深度学习：偏差与方差

为了避免过拟合，我们需要在拟合能力和复杂度之间权衡。拟合能力强的模型一般复杂度比较高，容易过拟合。相反，如果限制模型复杂度，会降低拟合能力，可能导致欠拟合。因此，在模型拟合能力和复杂度之间取得一个较好的平衡，对于一个机器学习算法来说非常重要。

概念：

偏差（Bias）：指一个模型在不同训练集上的平均性能和最优模型的差异，**用来衡量一个模型的拟合能力。**

方差（Variance）：指一个模型在不同训练集上的差异，**用来衡量一个模型是否容易过拟合**，即泛化能力。

机器学习模型中一般包含以下四种偏差和方差的组合情况。每个图中心点为最优模型。

(a)图给出一种理想情况，偏差和方差都相对比较低。

(b)图为高偏差低方差的情况，表示泛化能力很好，但拟合能力不足。

(c)图为低偏差高方差的情况，表示模型拟合能力很好，但泛化能力不足，当训练数据较少时，容易过拟合。

(d)图表示高偏差高方差，是最差的情况。

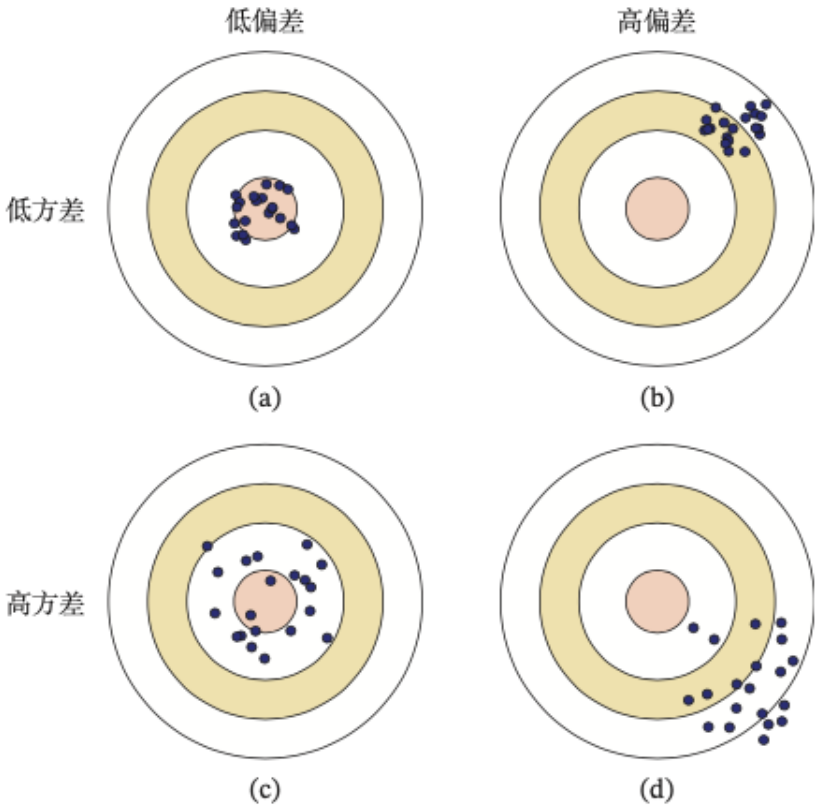


图 2.6　机器学习模型的四种偏差和方差组合情况

方差一般会随着训练样本的增加而减少。样本较多时，方差比较小，这时候可以选择能力强的模型来减少偏差。

随着模型复杂度增加，模型的拟合能力变强，偏差减少而方差增大，从而导致过拟合。以结构风险最小化为例，我们可以调整正则化系数 λ 来控制模型的复杂度。

- 当 λ 变大时，模型复杂度会降低，可以有效减少方差，避免过拟合，但偏差会升高。

- 当 λ 过大时，总的期望错误反而会上升，因此，一个好的正则化系数 λ 需要在偏差和方差之间取得比较好的平衡。

因此，一个好的正则化系数 λ 需要在偏差和方差之间取得比较好的平衡。下图给出来机器学习模型期望错误、偏差和方差随复杂度的变化情况，其中红色虚线表示最优模型，最优模型并不一定是偏差曲线和方差曲线的交点。

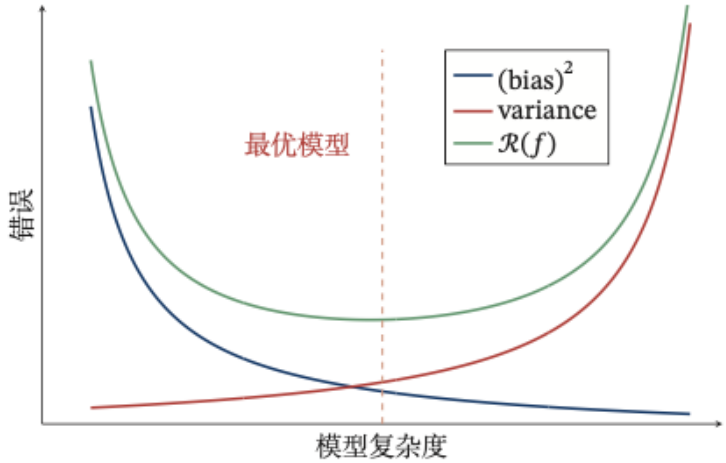


图 2.7　机器学习模型的期望错误、偏差和方差随复杂度的变化情况

偏差和方差分解给机器学习模型提供了一种分析途径，但在实际操作中难以直接衡量。

一个模型在训练集上错误率比较高时，说明模型的拟合能力不够，偏差比较高。一般通过以下方式进行改进：

- 增加数据特征
- 提高模型复杂度
- 减小正则化系数
- 尝试多项式特征【Adding polynomial features】
- 集成模型【通过多个高方差模型的平均来降低方差】

当模型在训练集上错误率比较低，但验证集上错误率比较高时，说明模型过拟合，方差比较高，一般通过以下方式进行改进：

- 降低模型复杂度
- 加大正则化系数
- 引入先验