第6周主要讲述如何诊断学习算法

Evaluating a Hypothesis

Once we have done some trouble shooting for errors in our predictions by:

- Getting more training examples
- Trying smaller sets of features
- Trying additional features
- Trying polynomial features
- Increasing or decreasing λ

We can move on to evaluate our new hypothesis.

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 %.

选取测试用的数据主要用三七分法，取70%的数据作为训练数据，剩下30%作为测试数据，

如果数据集排列有一定的规律性，务必打乱排序后再进行分割

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 $J_{test}(\Theta)$

一般情况下，数据集应该分为训练集，交叉验证集，测试集，因为，我们能会假设有好几种可能的模型，然后用数据集分别去训练这几个模型（用trainset得到theta），然后利用交叉验证集去选择一个比较好的模型（用cvset得到lambda），最后用测试集去测试选出最优模型的性能（用testset得到最合适的theta+lambda组合）。

如果，我们只假设了一个模型，那么就没有选择模型这个过程，那就把数据集分为训练集，测试集就可以了，训练集训练模型，测试集测试模型。

The test set error

1. For linear regression: $J_{test}(\Theta) = \frac{1}{2m_{test}} \sum_{i=1}^{mtest} (h_\Theta(x_{test}^{(i)}) - y_{test}^{(i)})^2$

2. For classification ~ Misclassification error (aka 0/1 misclassification error):

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

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

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

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

Train/validation/test error

Training error:

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

J(θ)

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:

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

我们在模型训练完，计算训练误差，交叉验证误差，测试误差的时候，即使有正则化参数，也不必加进去，只需按照上面的式子计算各种误差即可，很容易理解，就是比较训练结果与实际结果的差异。

Model Selection and Train/Validation/Test Sets

Just because a learning algorithm fits a training set well, that does not mean it is 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 you trained the parameters will be lower than the error on any other data set.

Given many models with different polynomial degrees, we can use a systematic approach to identify the 'best' function. In order to choose the model of your hypothesis, 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%

先用测试集得出好几组theta值，然后用得到的theta去计算交叉验证集（cv）的cost function，然后选出最小的那一组，就是最理想的参数值和次方数
We can now calculate three 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);

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

Diagnosing Bias vs. Variance

In this section we examine the relationship between the degree of the polynomial d and the underfitting or overfitting of our hypothesis.

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

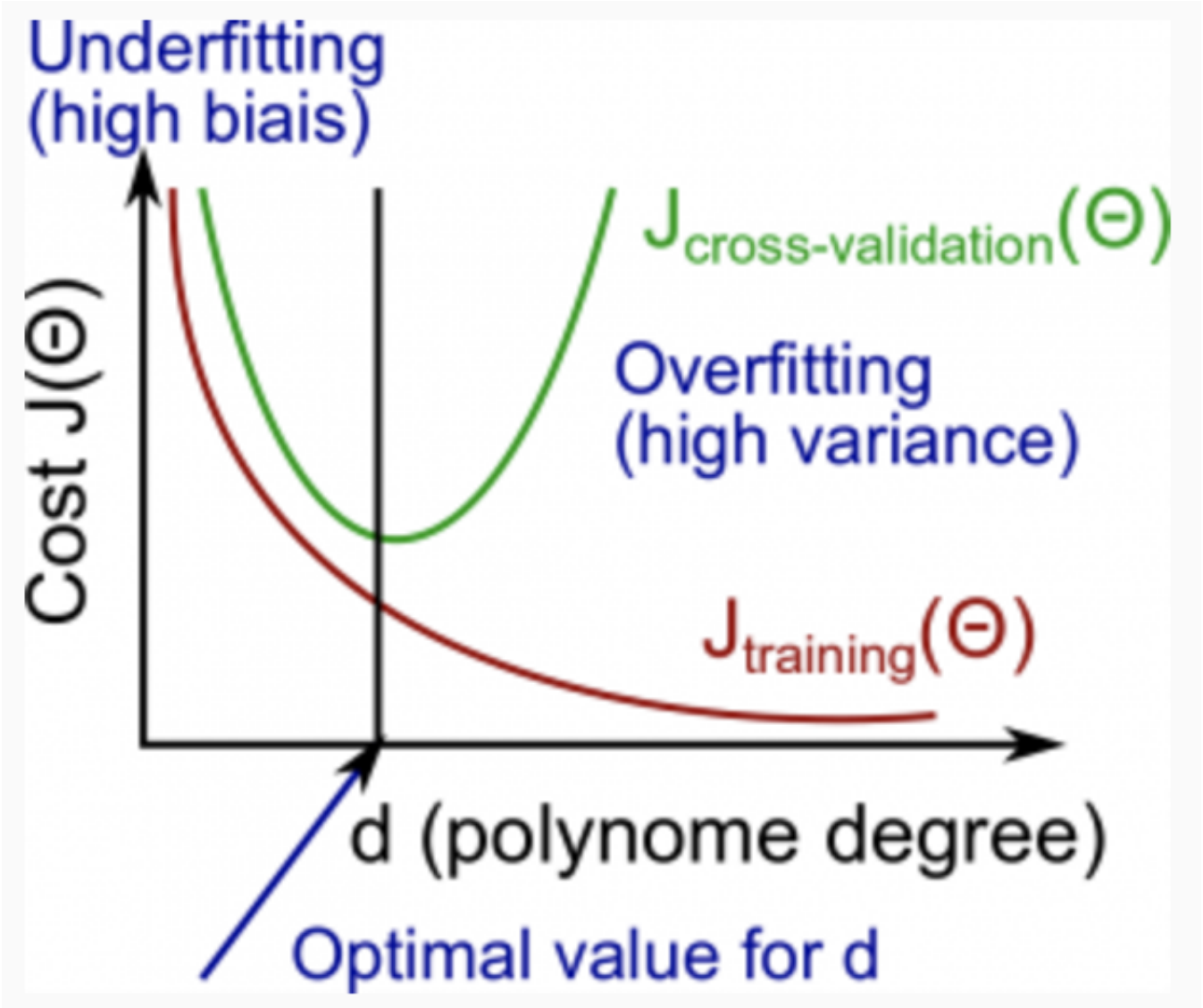
The training error will tend to **decrease** as we increase the degree d of the polynomial.

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, forming a convex curve.

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 $J_{train}(\Theta)$.

The is summarized in the figure below:

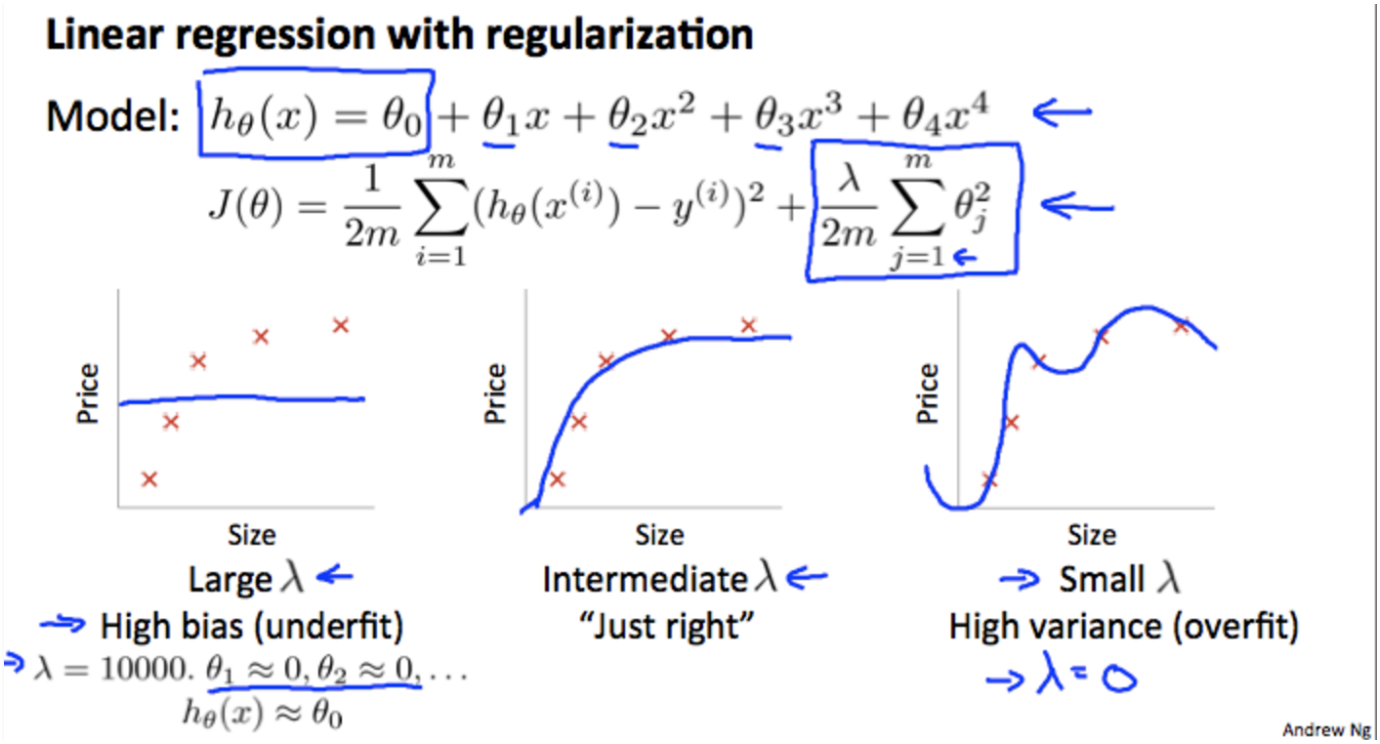


偏差很高且几乎相同：非拟合

交叉检测远高于测试集：过拟合

Regularization and Bias/Variance

Note: [The regularization term below and through out the video should be $\frac{\lambda}{2m}\sum_{j=1}^{n}\theta_j^2$ and **NOT** $\frac{\lambda}{2m}\sum_{j=1}^{m}\theta_j^2$]



In the figure above, we see that as λ increases, our fit becomes more rigid. On the other hand, as λ approaches 0, we tend to over overfit the data. So how do we choose our parameter λ to get it 'just right' ? In order to choose the model and the regularization term λ, we need to:

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 λs and for each λ go through all the models to learn some Θ.
4. Learn the parameter Θ for the model selected, using $J_{train}(\Theta)$ with the λ selected.
5. Compute the train error using the learned Θ (computed with λ) on the $J_{train}(\Theta)$ **without** regularization or λ = 0.
6. Compute the cross validation error using the learned Θ (computed with λ) on the $J_{CV}(\Theta)$ **without** regularization or λ = 0.
7. Select the best combo that produces the lowest error on the cross validation set.
8. Using the best combo Θ and λ, apply it on $J_{test}(\Theta)$ to see if it has a good generalization of the problem.

Learning Curves 学习曲线

Training an algorithm on a very few number of data points (such as 1, 2 or 3) will easily have 0 errors because we can always find a quadratic curve that touches exactly those number of points. Hence:

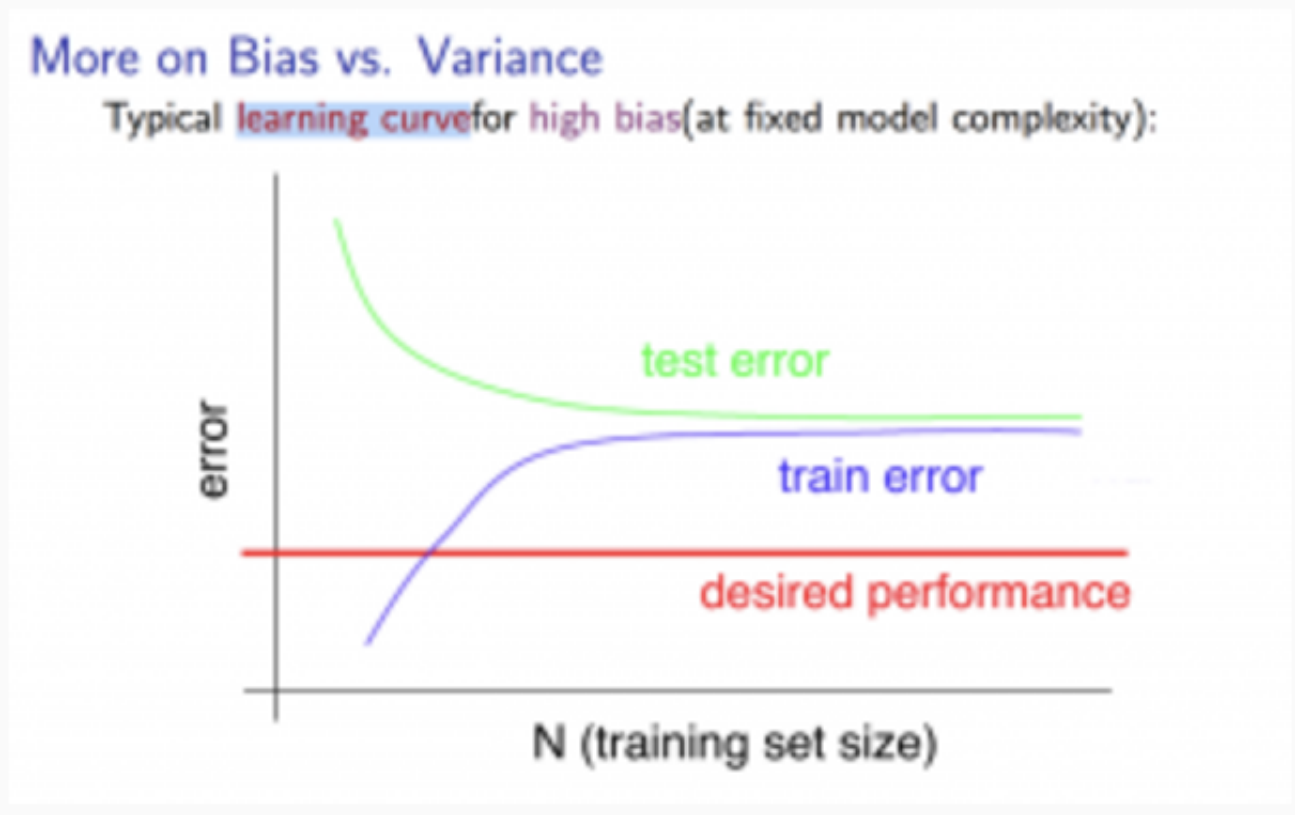
- As the training set gets larger, the error for a quadratic function increases.
- The error value will plateau out after a certain m, or training set size.

Experiencing high bias:

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

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

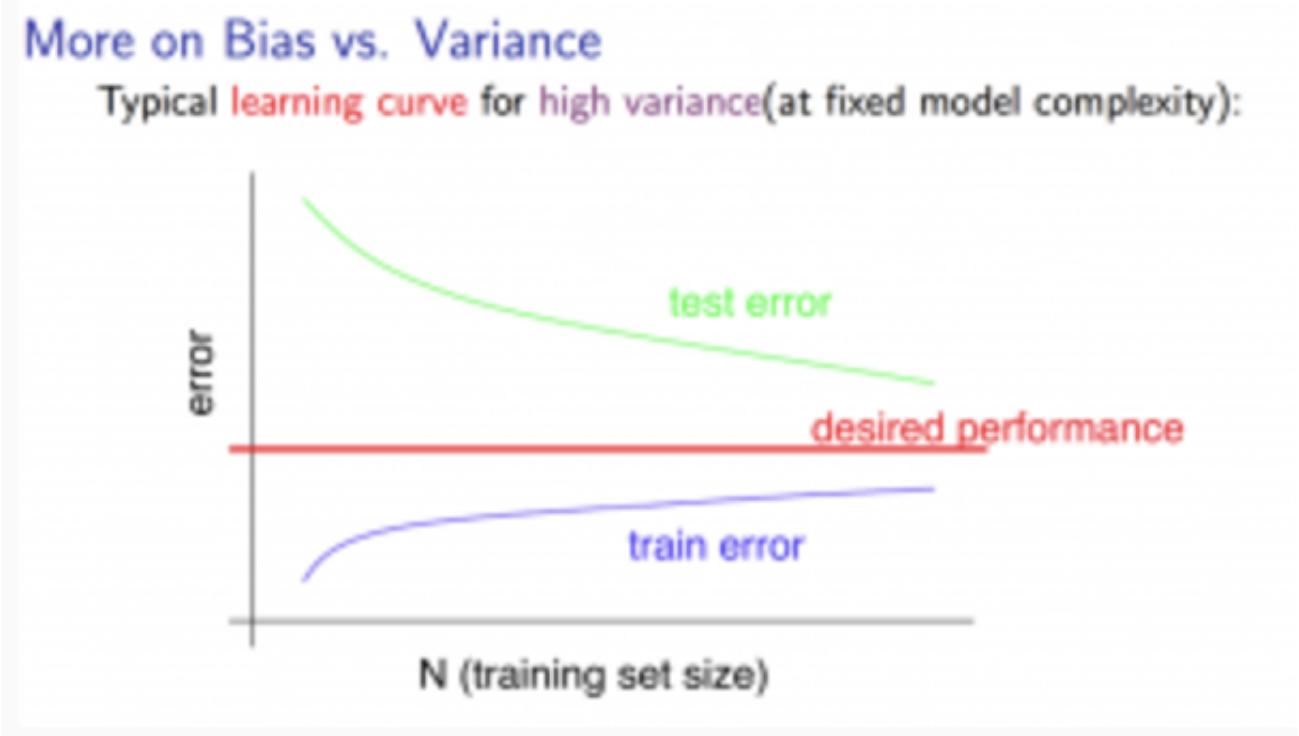
If a learning algorithm is suffering from **high bias**, getting more training data will not **(by itself)** help much.



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.



学习曲线可以帮助判断算法的表现，

（1）当训练集和交叉检测集的结果相近但误差很大，说明算法处于高偏差（high bias），此时增加训练集数量没有任何帮助

（2）当训练集和交叉检测的结果相差较多，说明算法处于高方差（high variance），此时增加训练集数量可以使差距变小

实际运用中会有很多噪音，但是基本可以看出这样一个趋势

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 is also **computationally cheaper**.
- A large neural network with more parameters is **prone to overfitting**. It is 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.

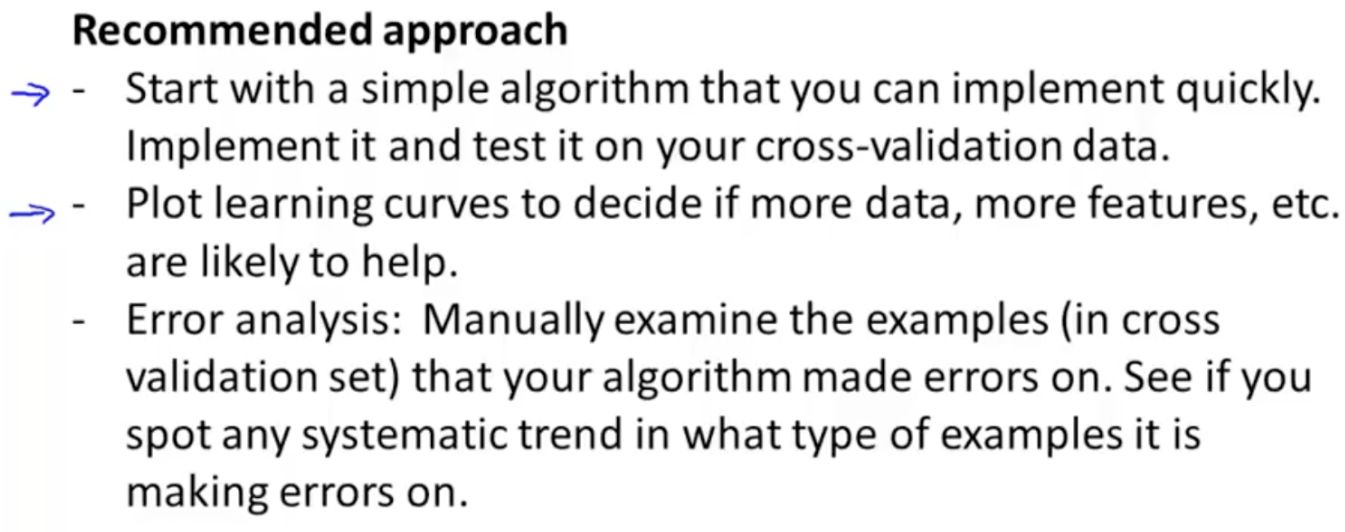
Model Complexity Effects:

- 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.

应对不同情况可以采取的方法：

高偏差时（high bias）： 1.增加特征数量　2.增加最高次方数　3.减小lambda（正规化）

高方差时（high variance）： 1.增加训练集数量　2.减少特征数量　3.增大lambda（正规化）



迅速作出原型进行计算，并通过绘制学习曲线得出优化的方向，避免过早优化（premature optimization）
然后通过错误分析来发现被错误分类的数据有什么特征，并以此提高算法的表现

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 examples it is making errors on.

下图：

错误分析可以让我们发现那些总是被错误分类的数据有哪些特征，以便采取相应的措施。

另外还可以通过观察被诊断为真的数据是否真的具有我们选取的某些特征，以便决定是否要继续使用那些特征

Error Analysis

m_{CV} = 500 examples in cross validation set

Algorithm misclassifies 100 emails.

Manually examine the 100 errors, and categorize them based on:

(i) What type of email it is — pharma, replica, steal passwords, ...

(ii) What cues (features) you think would have helped the algorithm classify them correctly.

Pharma: 12

Replica/fake: 4

Steal passwords: 53

Other: 31

Deliberate misspellings: 5
(m0rgage, med1cine, etc.)

Unusual email routing: 16

Unusual (spamming) punctuation: 32

下图：

我们需要有一个数学化的评估来帮助我们判断一个方法是否有效，

例如我们不使用词干提取法时错误率为5%，使用时为3%，则我们知道应该使用词干提取法（Porter stemmer）

再比如当我们停止使用大小写区分时，错误率变成了3.2%，这有助于我们思考是否要区分大小写

The importance of numerical evaluation

Should discount/discounts/discounted/discounting be treated as the same word?

Can use "stemming" software (E.g. "Porter stemmer")

universe/university.

Error analysis may not be helpful for deciding if this is likely to improve performance. Only solution is to try it and see if it works.

Need numerical evaluation (e.g., cross validation error) of algorithm's performance with and without stemming.

Without stemming: 5% error With stemming: 3% error

Distinguish upper vs. lower case (Mom/mom): 3.2%

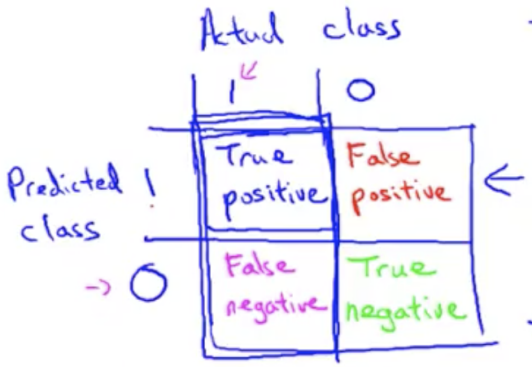
然而这样的做法是有例外的，例如当我们预测病人是否患有癌症时，我们的准确率可能达到了99%，
但是如果实际上得癌症的人只有0.5%，那么99%这样一个概率显然是不够的。
更为严重的是，当你不用任何算法，只预测所有人都没有得癌症，那么你也可以得到一个高达99.5%的准确率，
在这种情况下，显然普通的数学化评估失去了作用，我们很难判断一个算法是否真的起到了作用。

拥有像这里的0.5%这样一个非常低的正样本的概率（即y=1的概率）的情况被称为偏斜类。
偏斜类的意思是一个样本中一种数据相比另一种数据的数量呈一种极端的分布，
通过只预测y=1或者y=0也能让算法的表现非常好，因此使用分类误差或者分类精确度将起不到很好的效果。

这里就需要引入两个新的概念，查准率（Precision）和召回率（Recall），见下图

Precision/Recall

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



→ **Precision**

(Of all patients where we predicted $y = 1$, what fraction actually has cancer?)

$$\frac{\text{True positives}}{\#\text{predicted positive}} = \frac{\text{True positive}}{\text{True pos} + \text{False pos}}$$

→ **Recall**

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

$$\frac{\text{True positives}}{\#\text{actual positives}} = \frac{\text{True positives}}{\text{True pos} + \text{False neg}}$$

$y = 0$

recall = 0

通过比较预测（Predicted Class）和实际结果（Actual Class），我们可以得到以下四个数据：

真阳性（True Positive）：预测为真，实际也为真

假阳性（False Positive）：预测为真，实际为假

假阴性（False Negative）：预测为假，实际为真

真阴性（True Negative）：预测为假，实际也为假

通过这四个数据，我们可以计算出以下两个数值：

查准率（Precision）：在所有被预测为患了癌症的病人里，有多少人是真正患了癌症的?

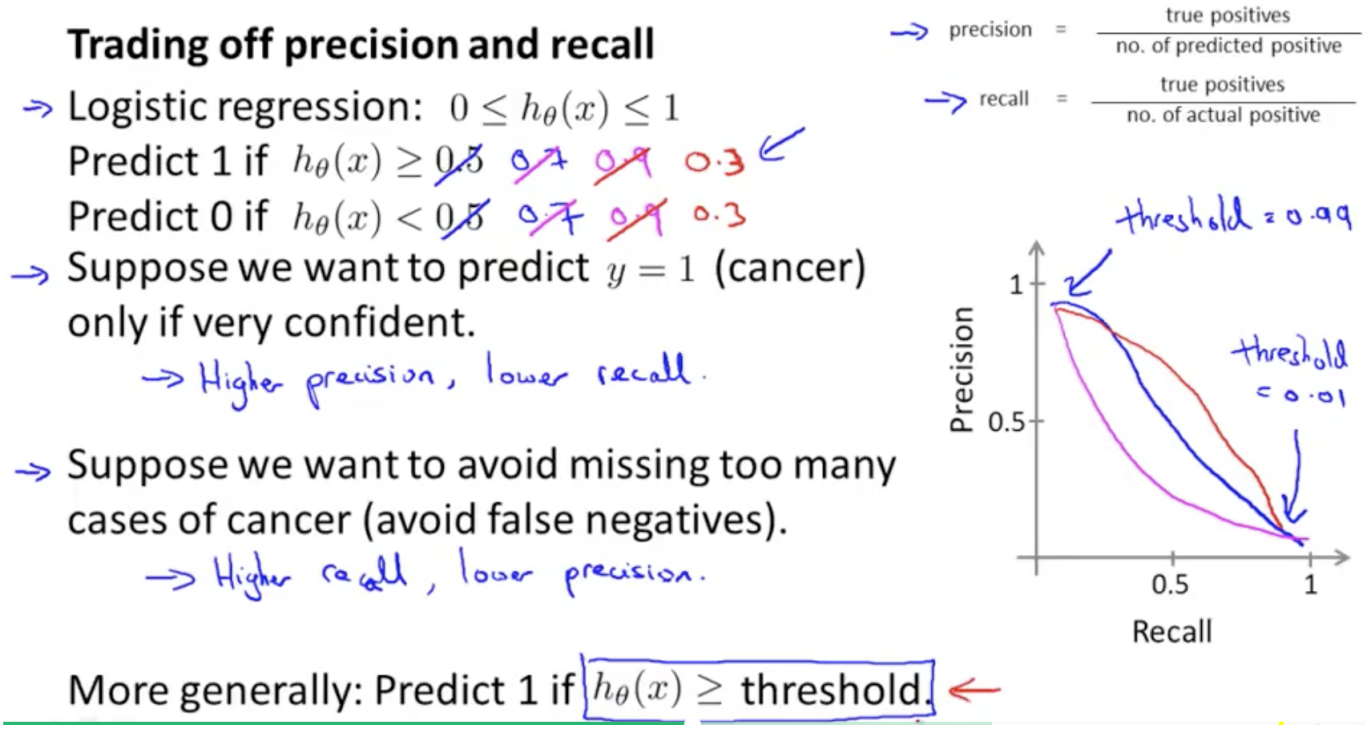
计算方法：真阳性的数量除以所有预测为阳性的数量（True Positives / #Predicted Positives，Predicted Positives = True Positives + False Positives）

召回率（Recall）：在所有实际患了癌症的病人里，有多少是被正确预测到的?

计算方法：真阳性的数量除以所有实际为阳性的数量（True Positives / #Actual Positives，Actual Positives = True Positives + False Negatives）

需要注意的是，我们应该通过交叉检测集（Cross Validation）来计算P和R的值

查准率和召回率是两个相互矛盾的概率，为了调整他们之间的平衡以得到一个表现良好的算法，我们需要考虑以下问题，见下图



在之前的课程中我们通常选择在有50%的确信时就判断样本为真，
但是想象一下判断癌症患者的情况，我们可能希望有70%甚至90%的确信时才判断一个患者得了癌症，
因为宣告癌症是一个非常严重的行为，我们可能希望能提高准确率，以避免给患者多余的担心。
通俗点说，在较高的确信下只要我们判断一个病人得了癌症，那么他有很大的概率真的得了癌症，
另一方面，对真正得了癌症的病人来说，却可能由于特征不明显使得确信程度较低导致被判定为没有得癌症。
也就是说此时我们会得到一个较高的查准率和一个较低的召回率（Higher Precision，Lower Recall）
相反，我们也可能希望不要放过任何一个可能患有癌症的病人，以确保他们能及时得到治疗，这时我们可能在有30%的确信时就判断患者得了癌症。
通俗点说，在较低的确信下当我们判断一个病人得了癌症，他不一定真的得了癌症，而仅仅是由于某些特征使得他看起来像是得了癌症，
另一方面，对真正得了癌症的病人来说，即使癌症的特征不明显，也会有很大的概率被判定为得了癌症，因此只要得了癌症几乎都能被判定出来。
此时我们会得到一个较低的查准率和一个较高的召回率（Lower Precision，Higher Recall）
最后，这里的确信值也被称为临界值（threshold）

那么有没有一个办法可以帮助我们自动选取合适的查准率和召回率呢？见下图

F_1 Score (F score)

How to compare precision/recall numbers?

	Precision(P)	Recall (R)	Average	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

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

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

P=0 or R=0 ⇒ F-score = 0.

P=1 and R=1 ⇒ F-score = 1

在这里，取平均值并不是一个好方法，因为当其中一个值非常大而另一个值非常小的时候也可能会得到一个比较高的平均值，这对我们的判断没有帮助。

因此，我们需要一个被称为F或F1的数值，F1 = 2 * （P * R / （P + R）），这个数值的特征是当P或R的其中一方比较小的时候会得到更高的权重，

而当P或者R等于0的时候F也会等于0

注意：F这个字母并没有什么实际意义，只是一些历史原因造成的

大量数据的作用，见下图

Large data rationale

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

→ $J_{train}(\theta)$ will be small.

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

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

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

建立学习算法时，我们应该首先用很多的特征来避免高偏差，然后再用大量的数据集来避免高方差

当算法的特征数不足以作出良好的预测，大量的数据也不会起到任何帮助