算法：

最好的算法永远在心中不断产生

通过技术使自己更便捷

考虑使用卡方方式去解决变量之间线性相关问题（非线性相关问题怎么解决？）

1model selection

2feature selection

可以考虑先使用一个算法，再采用另一个继续

adjust: when should we

add more samples(me:need more to reduce overfit)

add/reduce features(me:reduce when high overfit)

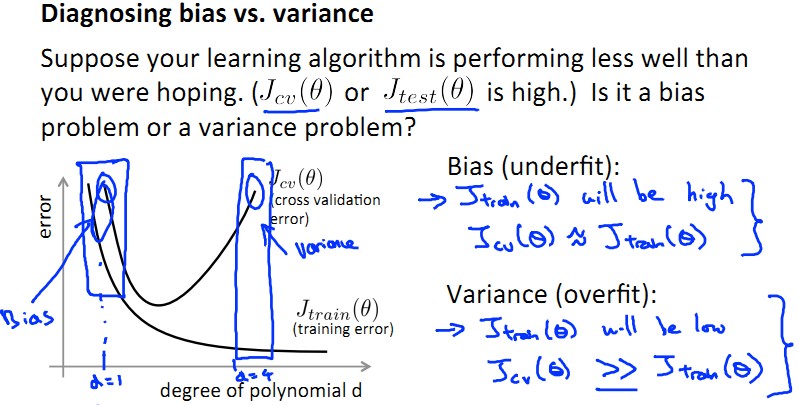
impose/dismipose regularizations(need more reg when overfit)

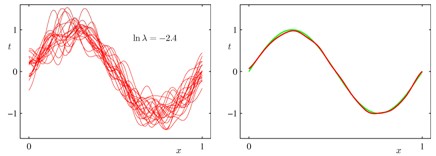
Along with the increasing degree of poly, the overfit increasing(essentially due to the features increased)

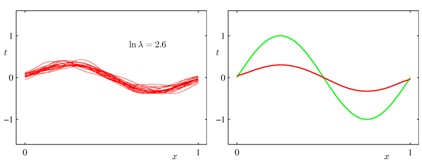
The training error is down, but test error first down then up, has a best fit

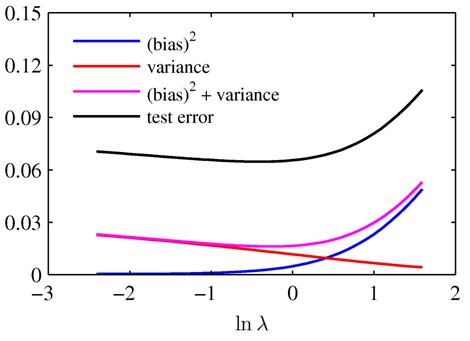
The sign of underfit: train error and test error both high

The sign of overfit: train error much smaller than test error





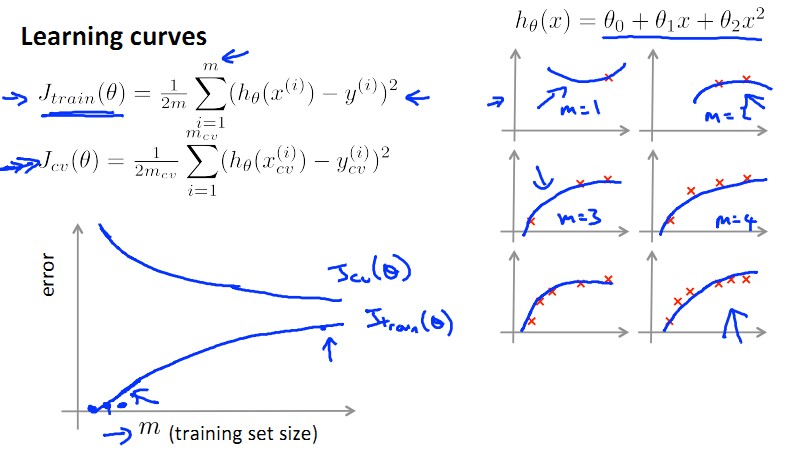




将误差（预测值与实际值（是个随机变量，但是有平均）的差距）分解为

（预测值与预测值本身的平均的差距，预测值本身的平均与实际值（概率模型的期望）的差距）

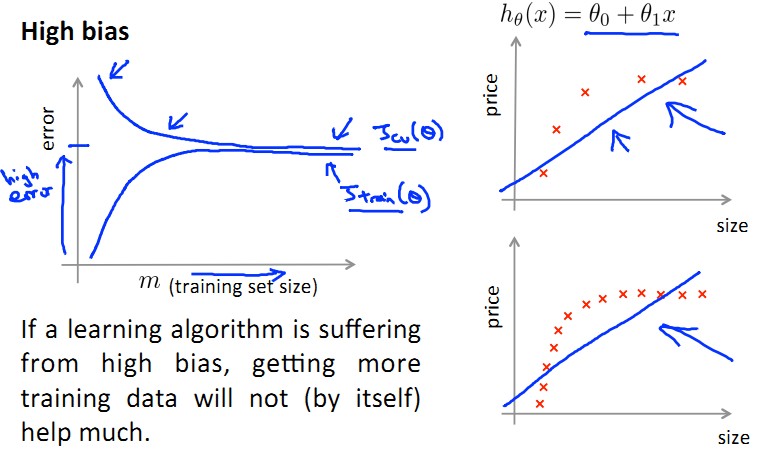
在选定模型的情况下，什么时候增加样本量有比较明显的效果？



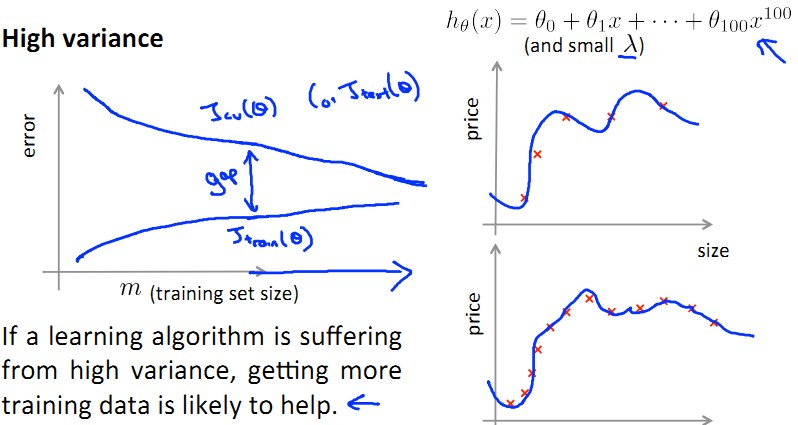
一般来说，随着样本量的增加，试验误差总是不断下降的，只是如果模型选的不对，这个降速会很慢，或者当一个模型样本量已经积累到一定程度时，单纯增加样本量已经不能使得模型持续快速下降了，上面的图是一个特例

训练误差总是随着样本量上升而上升的

当模型high bias,增加样本量不能提升，从图上看，训练误差和试验误差压在一起



当模型high variance,增加样本量可以提升，从图上看，训练误差永远和试验误差距离很大



分类重要指标：

sensitivity = true positive/actual positive

specificity = true negative/actual negative

precision = true positive/predicted positive

recall = true positive/actual positive

F1 score = 2\*P\*R/(P+R)

semi supervised learning

one class classification(one class SVM)

PU learning:In PU learning, two sets of examples are assumed to be available for training: the positive set {\displaystyle P} P and a mixed set {\displaystyle U} U, which is assumed to contain both positive and negative samples, but without these being labeled as such.

I am thinking about naive baysian and EM algorithm

semi supervised learning can refer to transductive learning or inductive learning. with labeled sample and unlabeled sample

Adaboost:

https://qizeresearch.wordpress.com/2013/12/05/short-example-for-adaboost/

http://blog.csdn.net/haidao2009/article/details/7514787

may read: https://baike.baidu.com/item/adaboost/4531273?fr=aladdin

loop: t

1Give the train set a weight to distribute Di = wi/sum(wi)

2Train the weak model ht

3Get the error by Err(t)= sum(D(i)\*I(ht(xi)!=yi))

4Get the weight of the tree a(t) = ln((1-Err(t))/Err(t))

5Get the weight of the next tree wi(t+1) = Di(t)\*exp(-a(t)\*yi\*ht(xi))

loop end

6 H(x) = sum(a\*h(x))

The different between it and gradient boost, gradient boost will have add gradient concern will this just set the weight of the sample

EM:

intuitive explanation: you have group of people with boys and girls, you want to figure out the mean of height of boys and girls

you need a guess parameter, then with each sample, you calculate the chance of girl/boy with that parameter; this is called the estimation step, then you use the estimation to calculate the new parameter, it would get close to the real value but may not converge to it.

another example is with the coins, two types of coins, each has own probability of Head and Tail, you initially set a prob for them, and using that to estimate the prob of type of a coin by "HTTHT", (P1=H)^2(P1=T)^3/((P1=H)^2(P1=T)^3 + (P2=H)^2(P2=T)^3); Then using this prob to calculate the more real parameter : all type I's head/ all type I get a better M(the idea is,since we got the frequece,so we get the prob using the freq)

it has some math background, to be solved tonight, ideally.

PCA:http://www.360doc.com/content/13/1124/02/9482\_331688889.shtml

A matrix M times a vector is to put the vector into the space with M's row as the basis

put all the samples with n dim,to a n\*m matrix X (better to scale it, then X\*X' is cov)

after Y = P\*X

A = Y\*Y' = P\*X\*X'\*P'

find the P to make A Diag, with cov = 0 and var arrange from max to min

then we pick the first r rows, it gives us an smaller space with each dim independent.

For nonlinear case, we need to using kernal PCA

using PCA to do the scores(?)

LDA

dimention reduce with tag on the class

maximize: w'(mu1-mu2)/w'(sigma1+sigma2)w

calculate the character value lambda (mu1-mu2)\*(mu1-mu2)'/sigma1+sigma2

and the char vector W which made the diag matrix

we extract first several matrix

ROC seems did greate job

it has true positive rate as y(sensitivity) and false positive as the x(1-specificity)

if you increase the threshold then you get low sensitivity and high specificity; the other way is the moving up

the area decided the better solution(how to calculate?)

even with bad (x.y) down the line, we could using the mirror of it(using the totally converse prediction)

Controversy prob baysian

when you know P(x|y) --to be more specific,you get P(ai|y),and you get P(x|y) = prod(P(ai)|y) you want to get P(y|x)

using P(x|yi)\*P(yi)/P(x) and compare them since P(x) is a constant, we could eliminate it.

So finally we got prod(P(aj|yi))\*P(yi) and compare it we got the answer

For the continous cases, we need to calculate the mean and var of the sample

then using that prob density function f(x,mu,sigma,yi); and using that to be p(ai|yi)

for the laplace correction, we add 1 for those type without some feature shows up

terminology:

posterior = prior \* likelihood/evidence

decision tree:

In mind you can imagin it as a big super rectangle with many small super rectangle in it/ or partition of the space

precut of the tree: stop to divide; post prune of the tree: make the cost of the test set down

With BIC it is possible to evaluate the posibility of the model

In R the BIC formula is given by:

2 \* loglik - nparams \* log(n)

which is reverse sign of original standard formula

so the higer the better

According to the test\_set, We get the model shape

then using that shape we using the sample to train a model

apply that model to the test set

We use our own way to decide if it works

Even it is simple~But sometimes works

study some example(may be practical,representative)

study a theory

apply it to more and more example to deepen the understanding

pretrain the model, and then using the better model may avoid overfit