Linear regression:

1. State the assumptions in linear regression model

4个：

1. No multicollinearity. The variables are independent.

For example, the total asset and fixed asset are correlated, we can’t use both of them.

We can use PCA, Lasso or Ridge to reduce features.

1. The variance of residuals should be constant. No heteroscedasticity. The residuals are not correlated with X or y. We omitted features that could predict y.

For example, we build a model to predict the consumption based on one’s income level.

Y(real) =a+b\*x+e(residual)

The problem is if a group people have lower income, the variation of their consumption is small. But if we use the model to predict a group of people are rich, their variation of consumption (variance of residuals) will be large. So, residuals are correlated with X. We need add features to predict y.

1. No auto correlation. The residuals are not correlated with each other. There are no seasonal signs. If there is, then we should consider season as another factor.

For example, if we use sales=a+ price\*b as our model, and plot the residuals, they are more like a wave with peaks and bottoms, then we may need to consider time factor to better predict sales.

1. The residuals are normally distributed with a mean of 0 and a constant covariance.
2. The variables are not random.

2. How to avoid overfitting in linear regression? Please list 3-5 practical experience:

First, we need to shuffle and split the data into train. validation and test. And if the dataset is small, we’d better to do cross validation to split the data into 5 folds. A,B,C,D,E . E is the final test data.

Then MODEL1

1. A~C train D valid；   
2. ABD train C valid;

….

Get Model 1’s the average e.g. MSE on validation data set. Choose the model with smallest average MSE.

Then apply model on test dataset. If there’s only one model, then choose the parameter that have better performance in most validation data.

If MSE in train data set is small but large in validation, it is likely to have overfitting problem. Usually, it’s because we use too many variables to predict.

It's like we are preparing for a test. And in our daily life, we did a lot of exercise on very weird questions and finally we can do them well. But the test is full of questions that are not like the question in our train set and our way to solve weird questions can’t be used on the test. So, we failed the test although we did well during our training.

So, we can use L1 or L2 to penalize the number of variables.

The L1 (loess) is to reduce the number of variables. That means some theta will be zero.

Min (MSE+1/N(|β1|+|β2|+…+|βn|)

The L2 (ridge) is to make the beta become more average. There won’t be an extremely large theta.

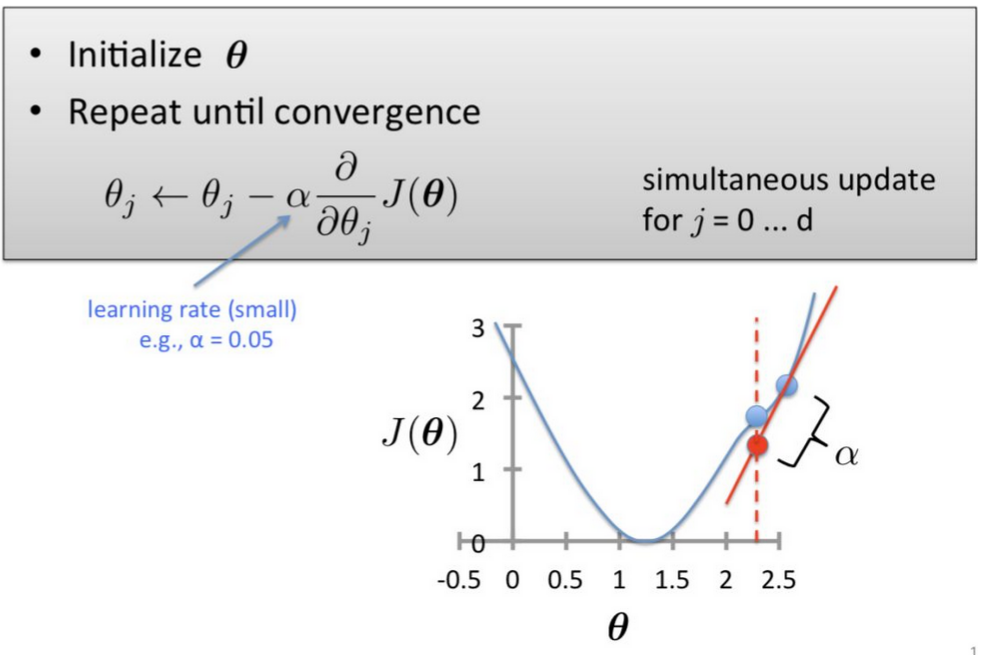
Min (MSE+1/N(β1²+β2²+....+ βn²)

I used L1, PCA and enlarge dataset to avoid overfitting.

1. Predict the house price with over 10 variables. And I used L1 to predict the house price. The MSE is smaller than not using it on both validation and the test data.
2. Predict if a company will go bankrupt with over 165 variables. I used PCA to find the number of variables that could lead to a minimum BER rate. And I compressed the variables to 25 dimensions.
3. Predict if a book is read or not. Because I increased the features from 2 to 7 to my model, I also increase train data size from 20000 to 50000. And the result on validation data become better.

3. Explain gradient descent with respect to linear regression

* Goal is to min (MSE). The tangent of the line that cuts the bottom of the function is 0, so our goal is to decrease theta until the tangent is 0. Our way is to use gradient descent. It's like we coming down from a mountain. First, we initialize our standing point. Then we choose different directions and steps to go down from the mountain. As we begin stepping down, every step we generate a new theta. Finally, we found the theta that won’t make J(θ) become smaller.



4. How to choose the value of the parameter learning rate α?

* The steps should not be too large because we may miss the bottom and the cost function may become larger and larger. It can’t be too small because we may waste a lot of time and find a local but not global optima.
* We can use feature scaling (standardize each feature to same unit) to increase the learning rate.
* But first we need to remove outliers and make sure the data in test and train have the same distribution. Otherwise, they have different means and different units of standard error. The model can’t fit on the test data.

5. How to choose the regularization parameter L?

Tune the parameter on validation set to find the one that min MSE of validation data.

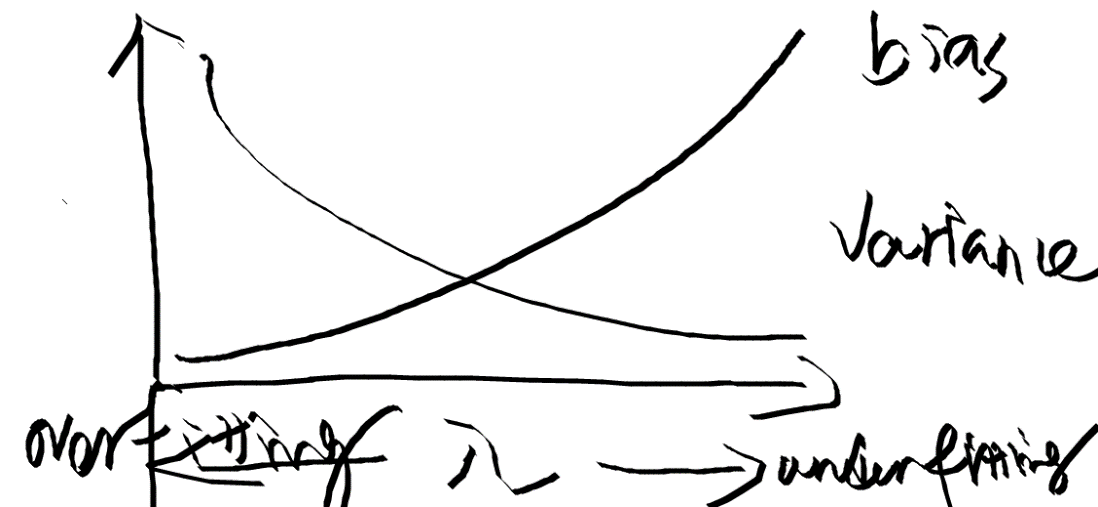
Lambda, can’t be too large because it will penalize the model too much and make it become underfitting.

Can't too small because it can’t fit validation data.

1. What is bias/ variance:

Bias means accuracy. If you want to have a low bias, you need fit your data perfectly. But it will cause high variance because it’s like your unique experience can’t be applied to other situations. This is usually happened in overfitting.

If you want to have a low variance, that means you want to create a common rule that can be used widely. But the downside is that the model may have a high bias because it can’t predict accurately. This is usually happened in underfitting.



1. What is dimension curse? How to prevent?

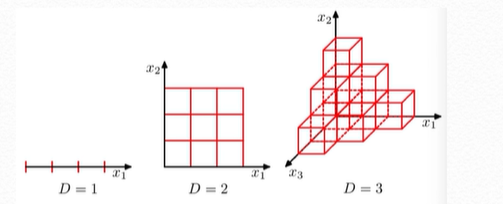
As the dimension increases, the data we need to better conduct some classification will grow exponentially.

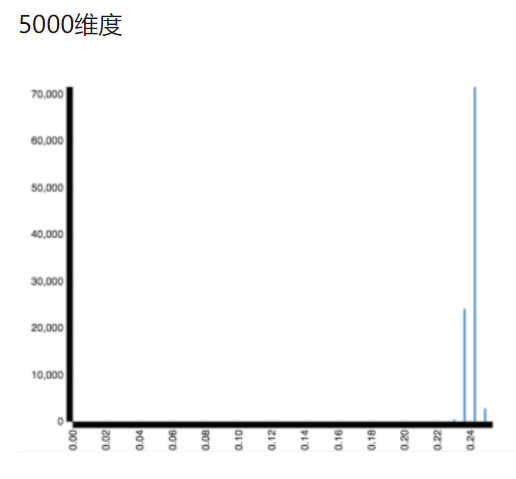
If our data is only one dimension and we got a new number. Say 0.1,2 and we got a number of 3. Clearly, we know 3 is closer to 2 and thus we can pose it. But if data becomes two dimensional. For example, we use taste and smell, the two features to define the goodness of a kind of beer. And each feature has 3 levels, then we need more than 3\*3=9 sample data to better classify a kind of beer. If the dimension becomes n d, then we need more than 3^n sample data to do prediction.

In reality, it’s hard to get a dataset that cover all the possibilities of the data. So high dimension will cause our train data set not representative and thus cause the model overfitting with a small sample size.

Even we get the large sample, it’s difficult to do classification.

Say K-means algorithm. We calculate the distance among points and assign the point to the nearest one. But if the dimension becomes higher, every one becomes equally far from each other. (the distribution of the distances of 2 points becomes concentrated) And thus this algorithm has no meaning. Just like your relationship with everyone in each group is the same and thus we can’t predict which group are you in.





8. How to prevent?

We can do feature engineering to reduce dimension.

such as PCA to compress features and random forest to select features.

In addition, large dataset small feature number: logistic regression + regularization svm

Small dataset large feature number: random forest, XG boost

Large dataset large feature number: neural network

Because the risk of overfitting decreases (you’ve done all kinds of homework and thus handle the test) as the data size increases and the more emphasis is put on the power of the model.

9. describe …. how did it work? pros and cons in practical implementation:

Logistic regression，SVM, PCA, 随机森林，GBM tree model 的pros and cons

Naive bayes:

LG: classifier: we want to have a classifier that predict only 0 and 1. So suppose Xi are the features of one row. Then, we build logit models:

Pr(Y=1| Xi)= e(Xi\*θ） / 1+ e(Xi\*θ)

Pr(Y=0| Xi)= 1- Pr(Y=1| Xi) = 1/1+ e(Xi\*θ)

They can limit our result from 0 to 1.

Pr(Y=1| Xi) is also equal to y=1/(1+e^(-X\* θ))

Why we write like this? Because the logic is we want our model predict

Y={1, if X\*θ >0 ; 0, if X\*θ <0 (threshold can be change))

The function can meet the logic. See if X\*θ >0, the y is larger than X\*θ<0.

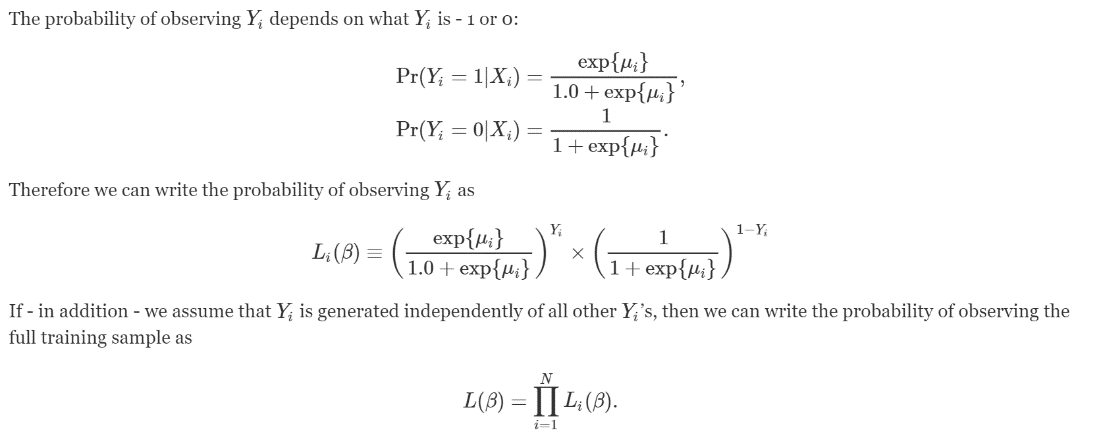
So the probability of overserving y of a given row:

Li( θ）= ( e(Xi\*θ） / 1+ e(Xi\*θ) ) ^ yi + ( 1 / 1+ e(Xi\*θ) ) ^ (1-yi )

If the yi is independent from other ys. Then the probability of overserving the training data is : (time all the possibilities)

L(θ)= Πi=1 N Li( θ）

We want to find theta that maximize the probability. Actually, the algorithm is to maximize log(L(θ)) and sovle theta.



Why Logistic regression doesn’t use R square?

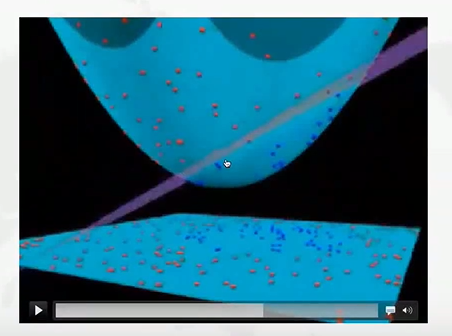
Because logistic regression predicts binary outcomes, not an exact value. R square is the model’s explained part except residuals. The features we put in the model explained how much of the variation of the y. There're no residuals in the model. So, there is no r square in lg.

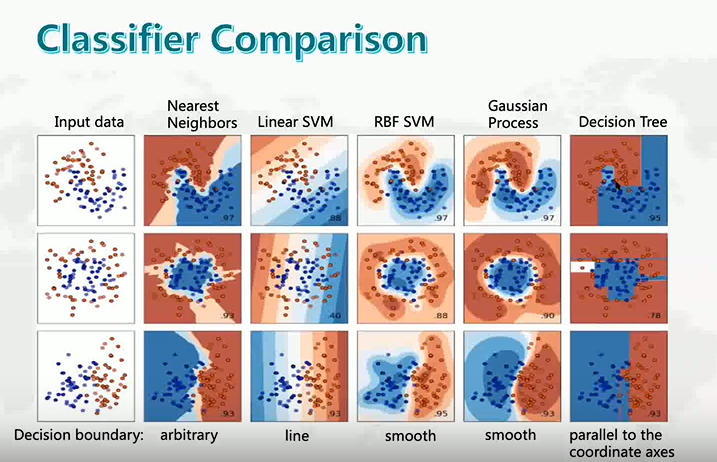
If we use the R square to measure the difference of predicted possibility and the true possibility (0 ,1), it is meaningless because we don’t care the difference, we just care whether you predict right or wrong. Even the difference is 0.0000000001, and you predict 0 but in fact it is 1. then you predict wrong. So, it makes sense that we use confusion matrix to evaluate our model.

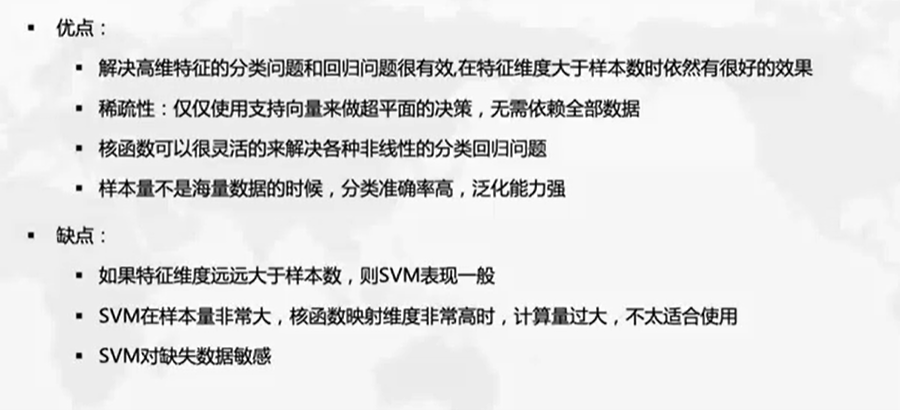
Pro: easy and useful to predict binary result

Cons: with high dimension data, the classifier becomes weak

SVM:







10. Missing value:

11. Metrics (cse midterm.)