*NOTES:*

**Week1:**

**Definition**

A computer program is said to learn form experience E with respect to some Task T and some performance measure P, if its performance on T, as measured by P, improves with experience E.

一个程序被认为能从经验E中学习, 解决任务T, 达到性能度量值P，当且仅当，有了经验E后, 经过P评判, 程序在处理 T 时的性能有所提升.

**Supervised Learning**

In supervised learning, we are given a data set and already know what our correct output should look like, having the idea that there is a relationship between the input and the output.

Supervised learning problems are categorized into "regression" and "classification" problems.

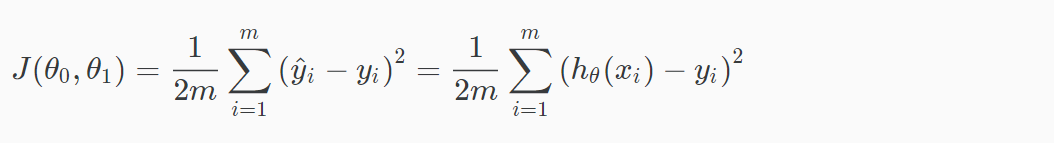
**Unsupervised Learning**

Unsupervised learning allows us to approach problems with little or no idea what our results should look like. We can derive (得到) structure from data where we don't necessarily know the effect of the variables. We can derive this structure by clustering the data based on relationships among the variables in the data.

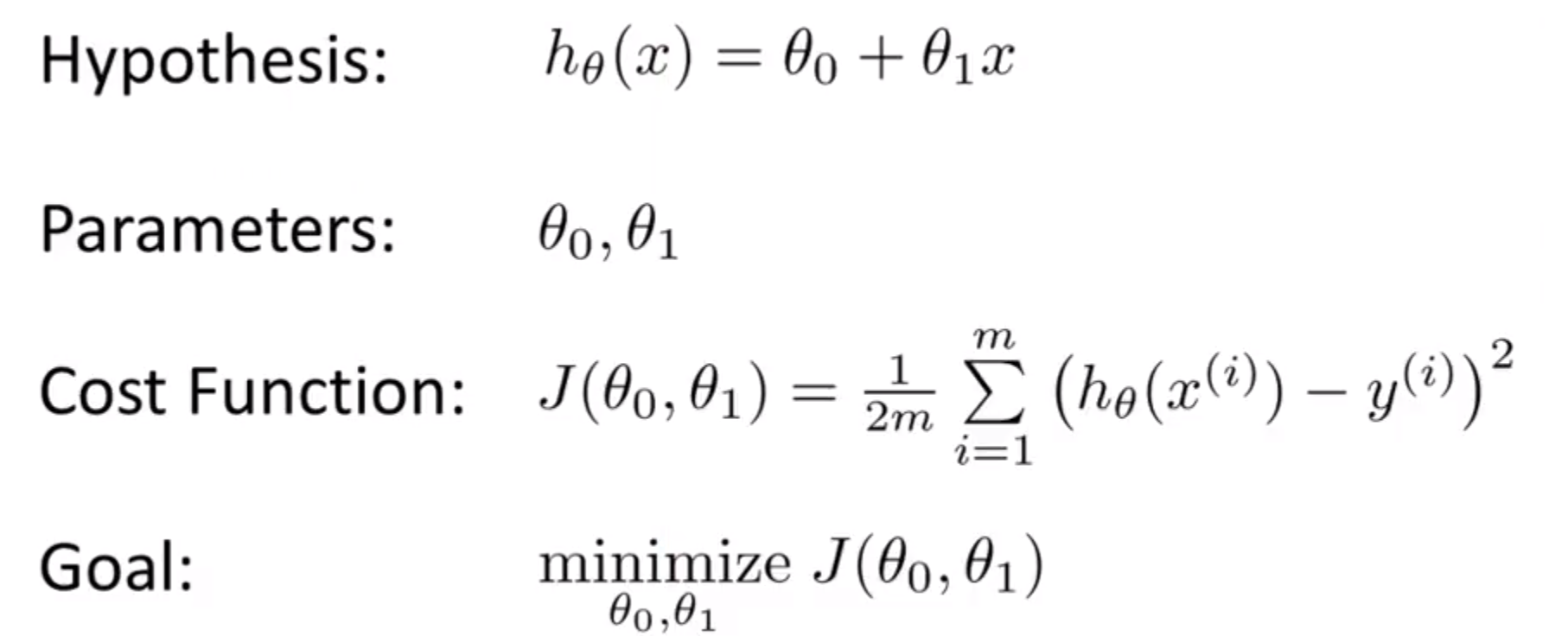
With unsupervised learning there is no feedback based on the prediction results.

**Learning Regression**

Cost function:



The 1/2 is as a convenience for the computation of the gradient descent, as the derivative term of the square function will cancel out the 1/2 term.

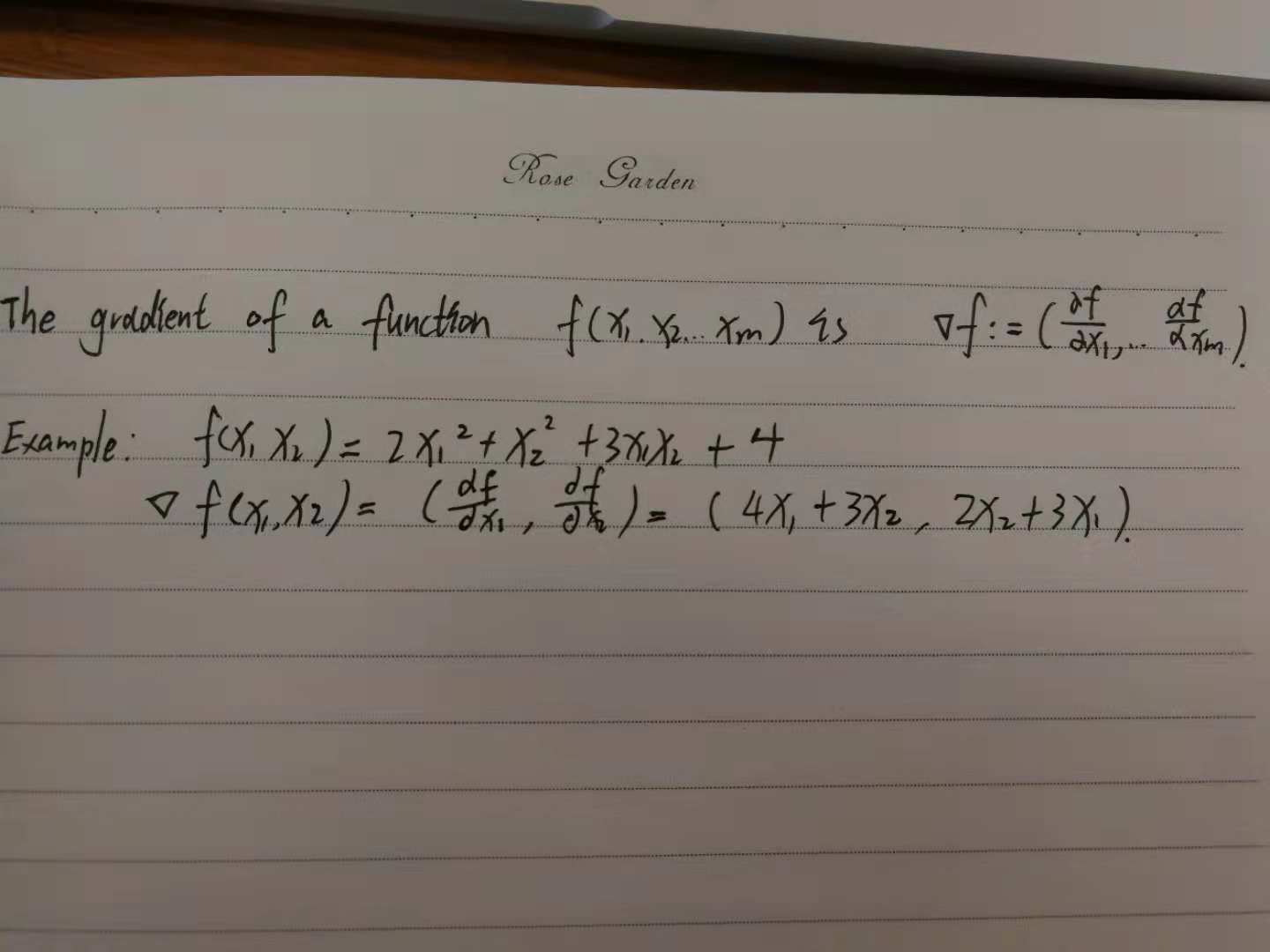


In order to minimize cost function, using gradient descent algorithm.

Hint: Difference between =(判断为真的声明) and :=(赋值)

**Gradient descent algorithm**

Definition of gradient (From neural computation course):

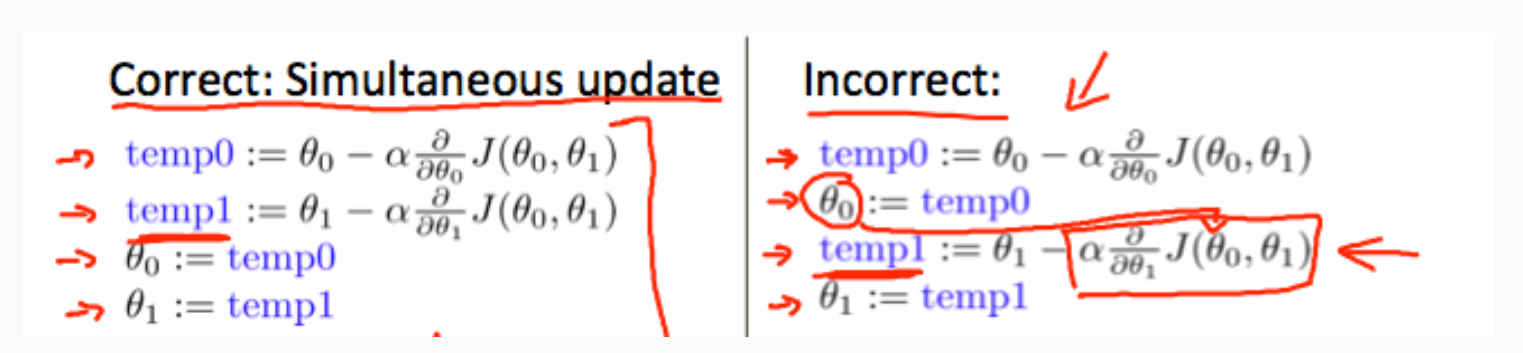


标量场中某一点的梯度指向在这点标量场增长最快的方向

The gradient descent algorithms show that where j = 0,1 and is learning rate. Its needs to repeat until convergence.



The parameters need to update simultaneously because >>



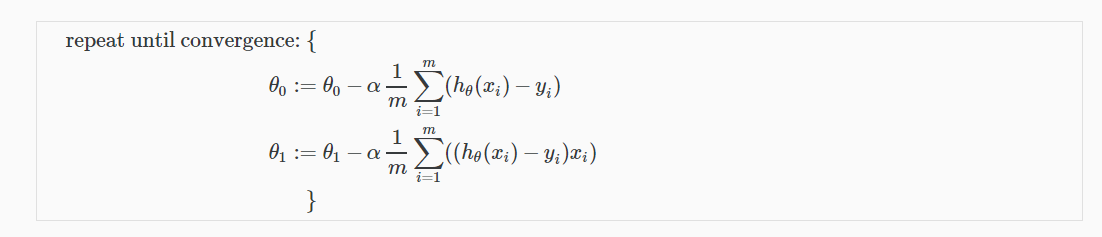
If learning rate is too small, gradient descent can be slow

If leaning rate is too large, gradient descent can overshoot the minimum. It may fail to converge, or even diverge(偏离，相异).

When point approach a local minimum, gradient descent will automatically take smaller steps because the slope(斜坡，倾斜，斜率) will small.

**Gradient descent for linear regression**

事实证明The cost function for linear regression is always going to be a bow shaped function (凸函数convex function). It does not have any local optimal except for one global optimum.



Batch gradient descent refers to each step of gradient descent uses all the training examples. Using m.

**Linear Algebra Review**

Matrix is rectangular array of numbers。

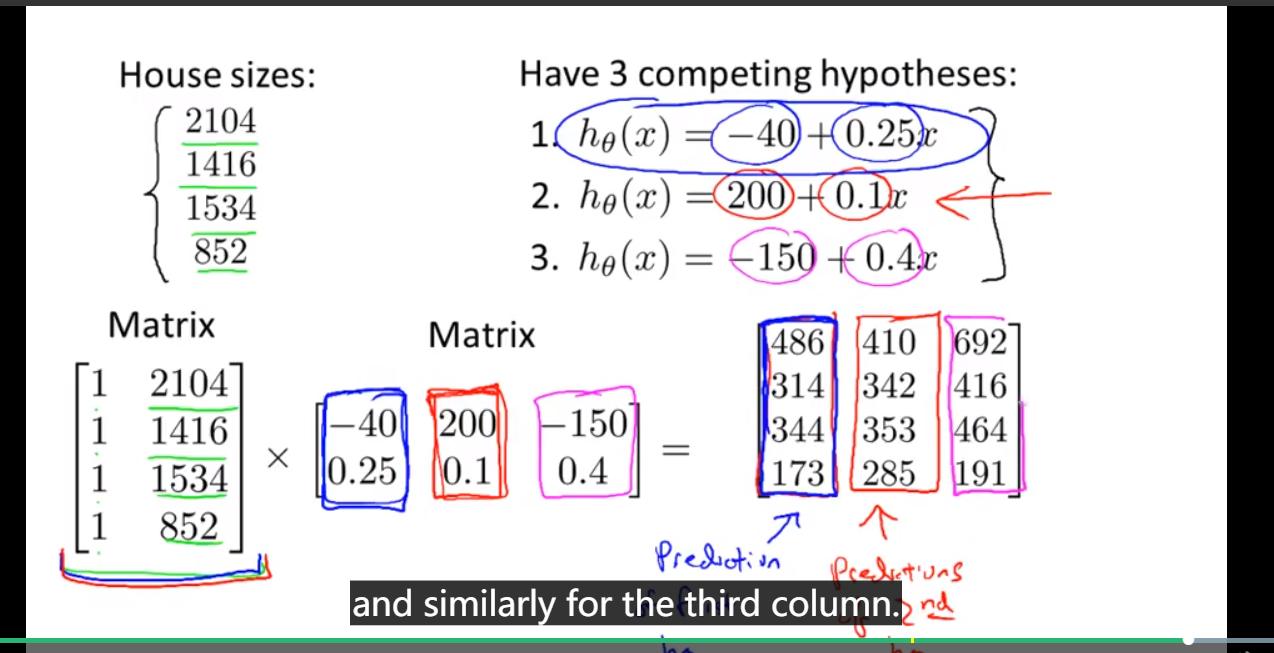
A vector is turns out to be a special case of matrix. A vector is a matrix that only has one column.

Matrix addition need matrix have same dimension.

Matrix multiplication. 新矩阵的行数是第一个矩阵的行，列数是第二个矩阵的列.

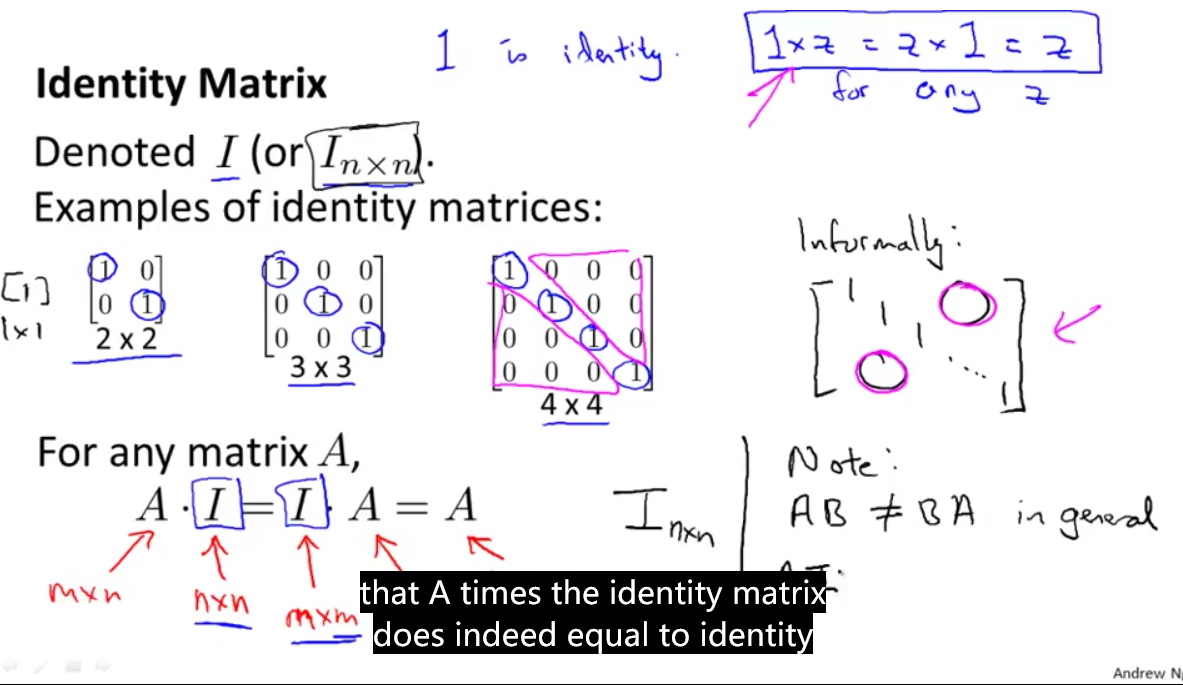
第一个矩阵的列数必须等于第二个矩阵的行数.

矩阵乘法的妙用在线性回归中~~



矩阵乘法不服从交换律, 但是符合结合律

Identity matrix



**Matrix Inverse and Transpose**

Not all numbers have an inverse like zero 0.

If A is an m x m matrix (square matrix), and if it has in inverse.

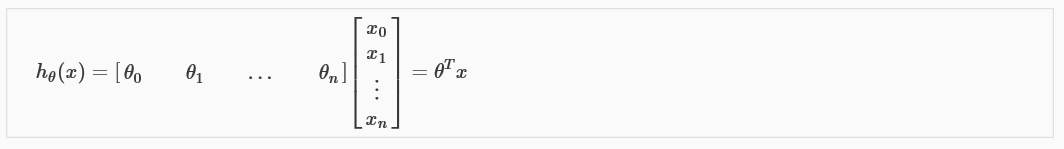
*A(A-1) = A-1A = I*

Matrices that don’t have an inverse are “singular” (奇异矩阵 比如全为0)

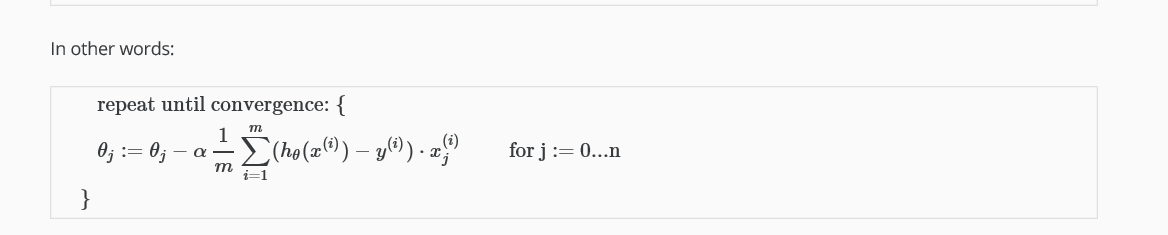
For transpose, B*ij* = A*ji*

Week2

**Linear Regression with multiple variables**



For multiple variables,



**Feature Scaling**

Make sure features are on a similar scale. Then the gradient descent can converge quickly.

Feature scaling involves dividing the input values by the range (i.e. the maximum value minus the minimum value) of the input variable, resulting in a new range of just 1. Mean normalization involves subtracting the average value for an input variable from the values for that input variable resulting in a new average value for the input variable of just zero. To implement both of these techniques, adjust your input values as shown in this formula:

Where μi​ is the **average** of all the values for feature (i) and si​ is the range of values (max - min), or si​ is the standard deviation.

特征缩放到1的范围，均值归一到0的mean,都可以用上述公式实现

**Learning rate**

If α is too small: slow convergence.

If α is too large: cost may not decrease on every iteration and thus may not converge.

To choose α, try 0.001,0.003.0.01,0.3,0.03,1….

**Features and Polynomial Regression**

We can improve our features and the form of our hypothesis function in a couple different ways.

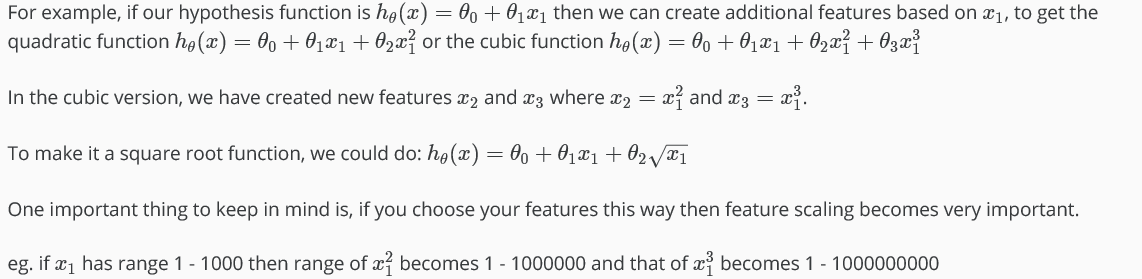
We can combine multiple features into one. For example, 土地的长和宽用面积表示

Polynomial Regression

Our hypothesis function need not be linear (a straight line) if that does not fit the data well.

We can change the behavior or curve of our hypothesis function by making it a quadratic, cubic or square root function (or any other form).

For example, cubic 立方

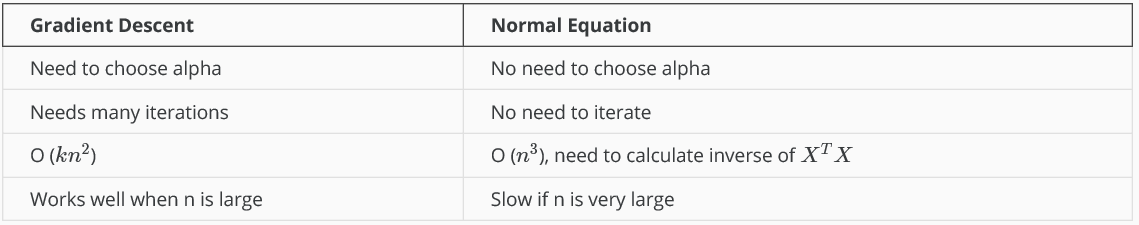


**Normal equation (Learning in machine learning module)**



NO need to do feature scaling with the normal equation.

Comparison of gradient and the normal equation:



Suggestion: 10000 以上用gradient descent

**Normal equation noninevitability**

If *XTX* is noninvertible, the common causes might be having :

1.Redundant features, where two features are very closely related (i.e. they are linearly dependent)

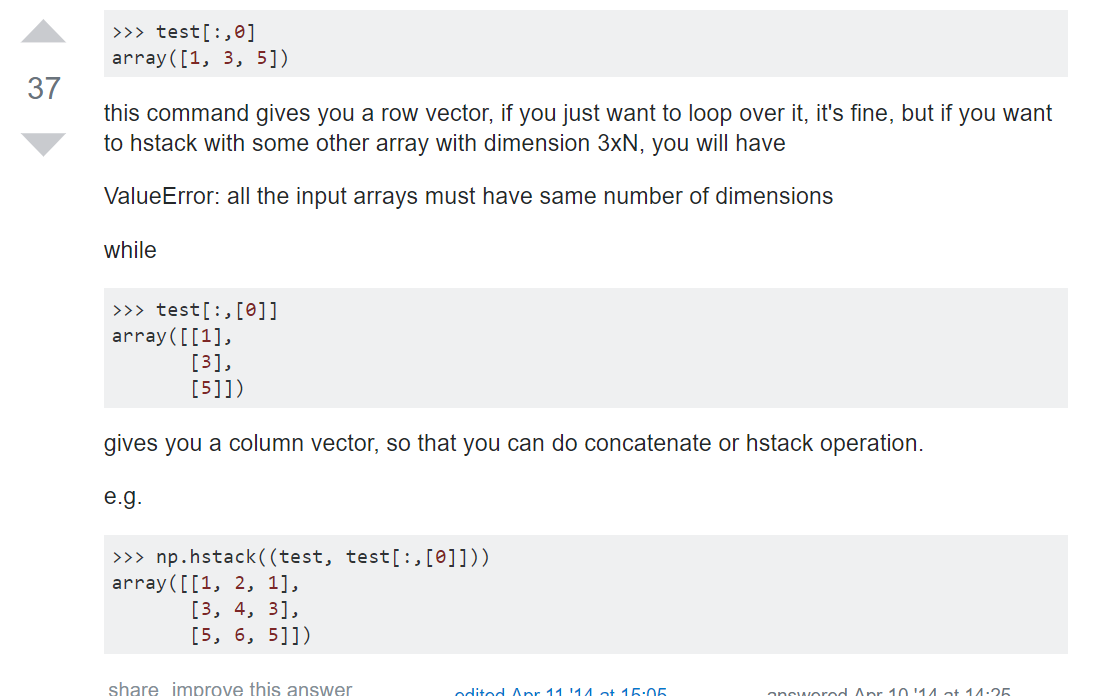
2.Too many features (e.g. m ≤ n). In this case, delete some features or use "regularization".

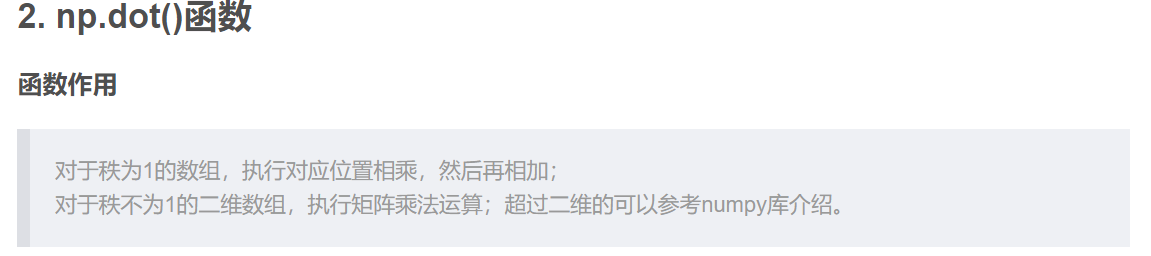
Solutions to the above problems include deleting a feature that is linearly dependent with another or deleting one or more features when there are too many features.

**Practice**

Levels off 平稳

valid commands 有效的命令



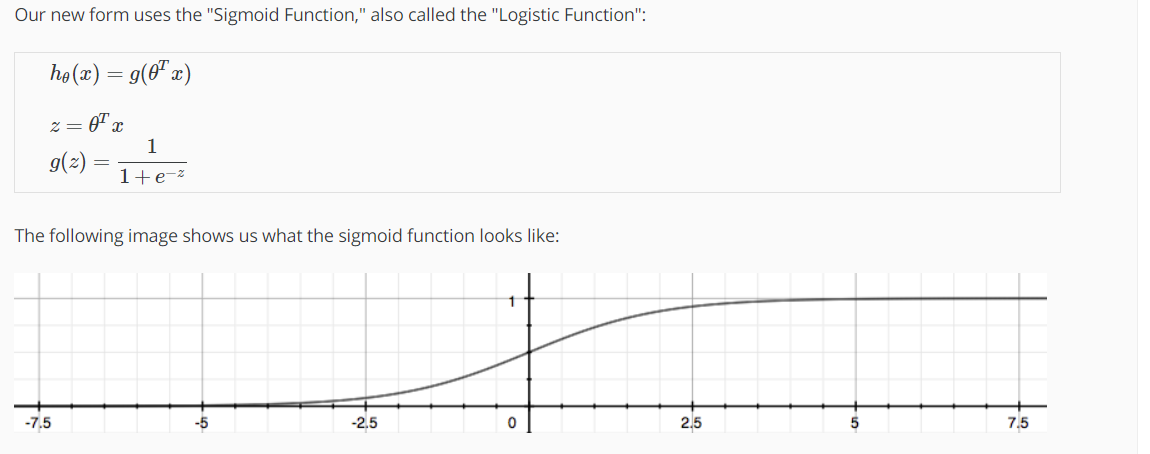


Week3

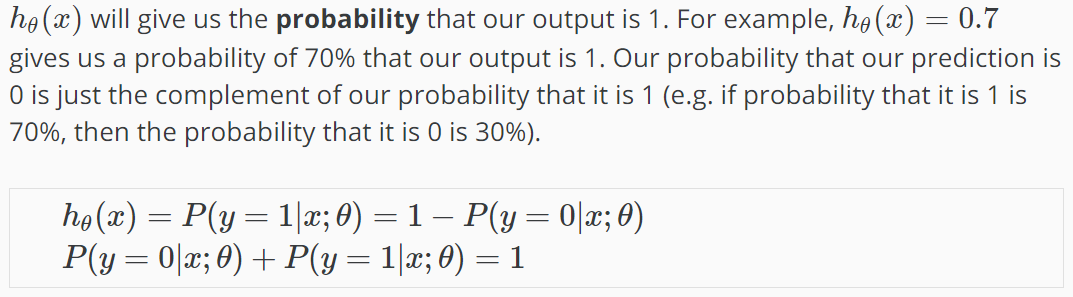
**Classification**

Predictions between 0 and 1

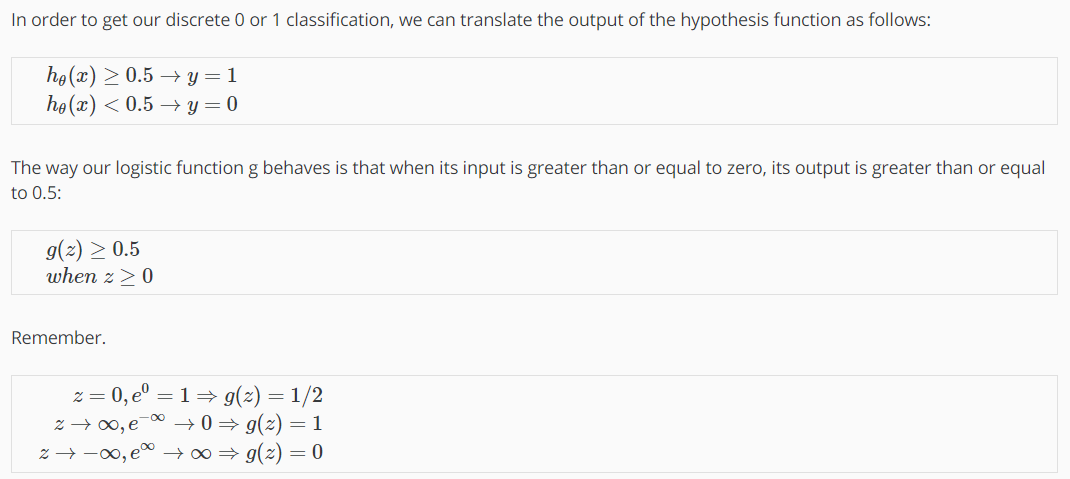
**Hypothesis Representation of logistic regression**



This function could map any real number to the (0,1) interval.



**Decision boundary**



So when h(x) = WTX >=0 is y=1

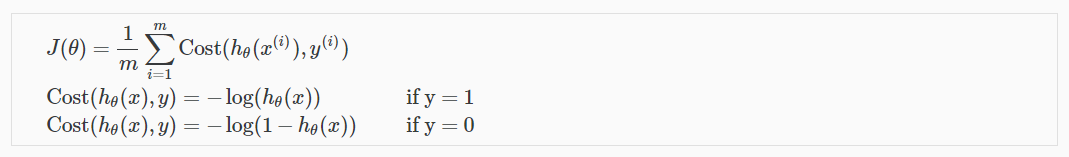
The decision boundary is the line that separates the area where y = 0 and where y = 1. Its create by hypothesis function.

By the way, the input to the sigmoid function doesn’t to be linear, and could be a function that describe the circle or any shape to fit our data.

**Cost function**

It is not possible to use same cost function that use for linear regression because the logistic function will cause the output to be **wavy(起伏的)**, causing **many local optima**. In other words, it will not be a convex function (凸函数).

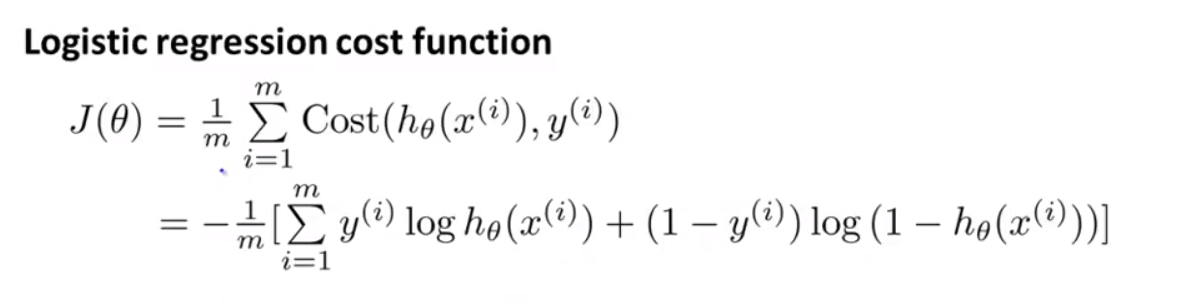
So the cost function is look like



Express the cost function in this way guarantees that J() is convex for logistic regression.

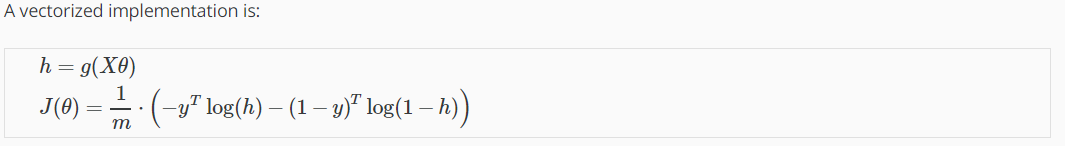
**Simplified cost function and gradient descent**

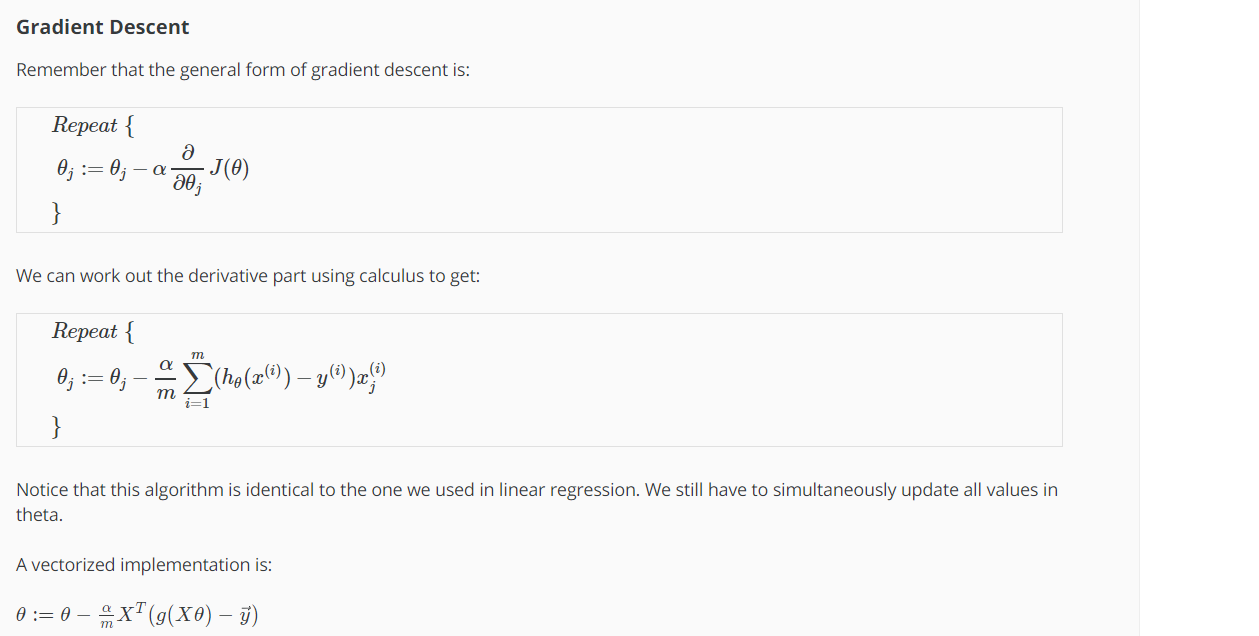
A more complex form of cost function shows that:



Note that when y = 1, the second term will be 0 and will not affect the result. By contrast, if y is 0 the first term will be 0 and will not affect the result.

A vectorized implementation is:





**Advanced optimization**

"Conjugate gradient", "BFGS", and "L-BFGS" are more sophisticated, faster ways to optimize θ that can be used instead of gradient descent.

**Multi-classification one-vs-all**

Basically, choosing one class and then lumping all the others into single second class, applying binary logistic regression to each case. Then use the hypothesis that returned the highest value as our prediction.

To sum up, training a logistic regression classifier h() for each class to predict the probability that y = i.

To make a prediction on a new x, pick the class that maximizes h(x).

**Solving the problem of overfitting**

Underfitting has high bias. Its not fitting very well.

Two main way to address the issue of overfitting:

1.Reduce the number of features

Manually select which features to keep.

Use a model selection algorithms.

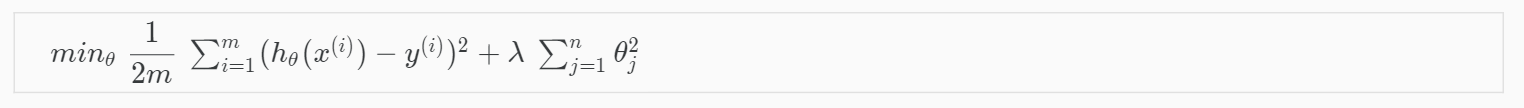
2.Regularization

Keep all features, but reduce the magnitude of parameters theta.

Regularization works well when we have lot of slightly useful features.

**Cost function**

Regularize all of our theta parameters in a single summation as



The lambda, is the regularization parameter, it is determines how much the costs of theta parameters are inflated (膨胀的).