目录

标记:
补: 需要补的知识
难点 (多看)
question: 疑问

一.ML Intuition

1.What is Machine Learning?

Arthur Samuel described it as: "the field of study that gives computers the ability to learn **without being explicitly programmed.**" This is an older, informal definition.

Tom Mitchell provides a more modern definition: "A computer program is said to learn from **experience E** with respect to some **class of tasks T** and performance measure **P**, if its performance at tasks in T, as measured by P, improves with experience E."

Example: playing checkers.

E = the experience of playing many games of checkers

T = the task of playing checkers.

P = the probability that the program will win the next game.

In general, any machine learning problem can be assigned to one of two broad classifications:1.Supervised learning and 2Unsupervised learning.

2.Supervised learning

Give the algorithm the dataset that the “corect answer” is given,try to learn a model(pattern.)

2.1 Regression and classification

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

A regression model predicts **continuous values.**(map input variables to some continuous function.
)

A classification model predicts **discrete values.**(map input variables into discrete categories.
)

3 Unsupervised learning

Datasets have no labels or all same label,not telling algorithm (machine)what to do, automatically try to find **structure** in it, which we don't know in advance

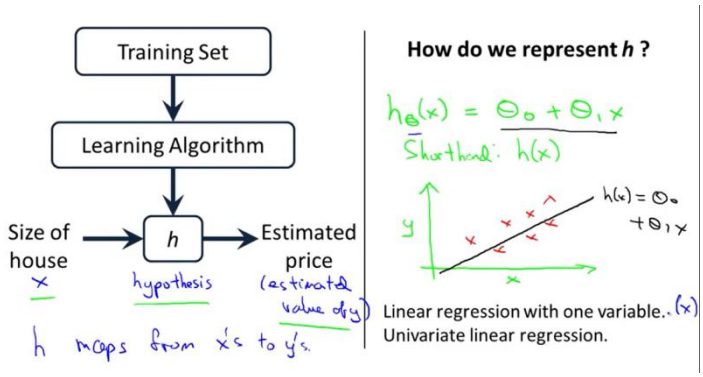
二.Linear Regression with One Variable

1.Model and Cost Function

1.1 Models

A **model** defines the **relationship** between features and label. e:

Training means **creating or** learning **the model.** That is, you show the model labeled examples and enable the model to gradually learn the **relationships** between features and label.**Training** a model simply means **learning (determining) good values for all the weights and the bias from labeled examples.**



train set feed algorithm, produce a **hypothesis**(function/model) that map x to y, use fomula to represent!

Hypothesis 就是 x 到 y 的映射

1.2 Inference

Inference means applying the trained model to unlabeled examples. That is, you use the trained model to make useful predictions (y').

1.3 cost function(L2 LOSS)

L2 LOSS：Mean square error (MSE)
1 首先 loss 的目的是为了寻参，所以 loss 一定是参数的函数!!! 寻参的方法一般是梯度下降！
2 LOSS 是针对所有 examples：要利用到所有数据来寻参
给定数据来寻求某种模型！用到全部数据：

mean(SUM (W1X1+W2X2 - Y)^2)

Loss 是 example data 值(x1,x2)和参数(w1,w2)组成，是参数的函数
参数产生预测值，example data 代表真实值，L2 loss 是代表两个值的误差
目的是让所有 example data 和 target data 误差最小!!!
example 的数值已知，因此变量是 parameter，需要搜索 parameter，使得 loss 最小!!!

区分概念，
Euclidean length == L2 norm == L2 distance：两个向量距离
元素差平方和，而 L2 loss 求了 mean

记住：
只要有 2，就意味着有平方，L2 loss 是预测向量和实际向量的差平方和均值：

Training Set

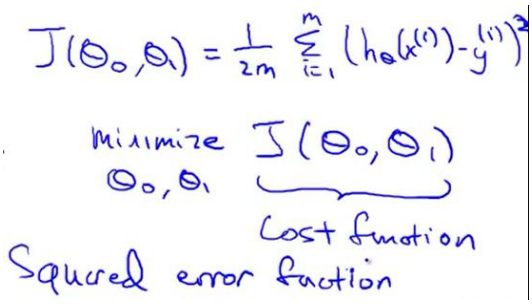
Size in feet² (x)	Price ($) in 1000's (y)
2104	460
1416	232
1534	315
852	178
…	…

m = 47

Hypothesis: $h_\theta(x) = \theta_0 + \theta_1 x$

θ_i's: Parameters

How to choose θ_i's ?



求 mean 的好处 1/m 均方误差!!! cost 不要随着 nums 增加而增加

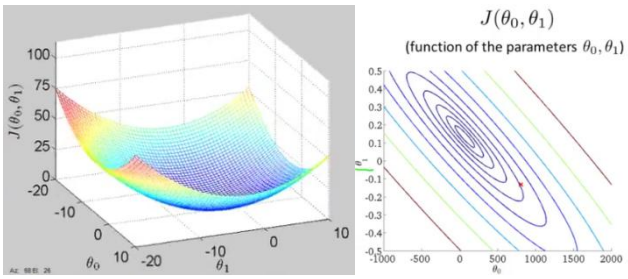
总结：

Hypothesis:	$h_\theta(x) = \theta_0 + \theta_1 x$
Parameters:	θ_0, θ_1
Cost Function:	$J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)}\right)^2$
Goal:	$\underset{\theta_0,\theta_1}{\text{minimize}}\ J(\theta_0, \theta_1)$

2.parameter learning

2.1 gradient descent

如果是两个参数 θ_1, θ_2 一起变化，则 cost function 是 3D 曲面



切开看就是 contour plot 等高线

每个 oval 代表**相同的 J value(cost 值)**

最里面的小圈是代表曲面的最低点，这一圈(θ_1, θ_2)组合都是 min of J(因为等高)

Gradient descent

应用于求目标函数最小或最大的参数选择方法。

gradient 是个**函数的偏导数向量**

Gradients

The **gradient** of a function, denoted as follows, is the vector of partial derivatives with respect to all of the independent variables:

$$\nabla f$$

For instance, if:

$$f(x,y) = e^{2y}sin(x)$$

then:

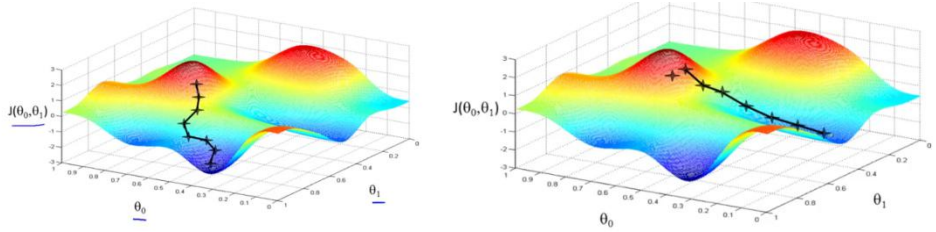
$$\nabla f(x,y) = (\frac{\partial f}{\partial x}(x,y), \frac{\partial f}{\partial y}(x,y)) = (e^{2y}cos(x), 2e^{2y}sin(x))$$

算法思路： initial parameter, keeping change a little bit

- Start with some θ_0, θ_1 (say $\theta_0 = 0, \theta_1 = 0$)
- Keep changing θ_0, θ_1 to reduce $J(\theta_0, \theta_1)$ until we hopefully end up at a minimum

find best direction to take tiny step dowm(reduce J), reach a new point。keep doing, until get down local minimum

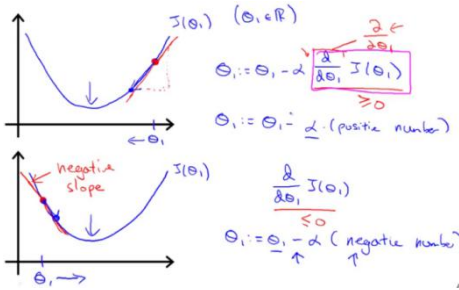
如何 change：只需要 1 方向，2stride



if change the initial point, you get another local minimum

1 方向：

对于一个参数好理解，沿导数负方向。



对于多维参数，每个参数都沿各自偏导数负方向 change。

Gradient descent algorithm

repeat until convergence {

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1)$$

(simultaneously update $j = 0$ and $j = 1$)

}

learning rate　derivative

参数更新时需要所有参数同时更新 simultaneously update

Correct: Simultaneous update

$\text{temp0} := \theta_0 - \alpha \frac{\partial}{\partial \theta_0} J(\theta_0, \theta_1)$

$\text{temp1} := \theta_1 - \alpha \frac{\partial}{\partial \theta_1} J(\theta_0, \theta_1)$

$\theta_0 := \text{temp0}$

$\theta_1 := \text{temp1}$

分别计算出右边结果 temp1，temp2，同属赋值给(更新) θ_1, θ_2。

错误的实现：compute temp1 update 一个参数，compute temp2 update 另一个参数 ，错误!!

Incorrect:

$\text{temp0} := \theta_0 - \alpha \frac{\partial}{\partial \theta_0} J(\theta_0, \theta_1)$

$\theta_0 := \text{temp0}$

$\text{temp1} := \theta_1 - \alpha \frac{\partial}{\partial \theta_1} J(\theta_0, \theta_1)$

$\theta_1 := \text{temp1}$

例：

Gradients

The **gradient** of a function, denoted as follows, is the vector of partial derivatives with respect to all of the independent variables:

$$\nabla f$$

For instance, if:

$$f(x,y) = e^{2y} sin(x)$$

then:

$$\nabla f(x,y) = (\frac{\partial f}{\partial x}(x,y), \frac{\partial f}{\partial y}(x,y)) = (e^{2y} cos(x), 2e^{2y} sin(x))$$

$$\frac{\partial f}{\partial x}(0,1) = e^2 \approx 7.4$$

x，y 是 weight，有初始值 (0,1)，计算该点 x 的偏导数值，并减去得到新的 x，x-7.4，因为只有这样 f 才会减少，同理 y

具体推导：

Gradient descent algorithm

repeat until convergence {

$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1)$

(for $j = 1$ and $j = 0$)

}

Linear Regression Model

$h_\theta(x) = \theta_0 + \theta_1 x$

$J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)}\right)^2$

$\min_{\theta_0, \theta_1} J(\theta_0, \theta_1)$

$$\frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1) = \frac{\partial}{\partial \theta_j} \cdot \frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)}\right)^2$$

$$= \frac{\partial}{\partial \theta_j} \frac{1}{2m} \sum_{i=1}^{m} \left(\theta_0 + \theta_1 x^{(i)} - y^{(i)}\right)^2$$

$$\theta_0\ j=0: \frac{\partial}{\partial\theta_0}J(\theta_0,\theta_1) = \frac{1}{m}\sum_{i=1}^{m}\left(h_\theta(x^{(i)})-y^{(i)}\right)$$

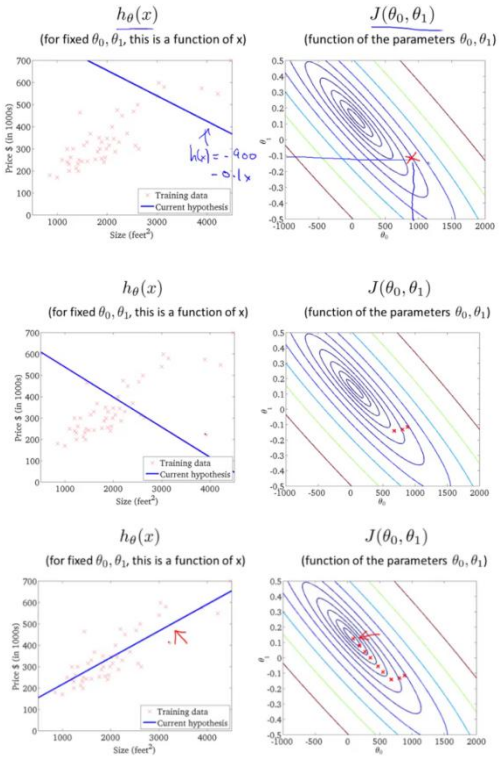
$$\theta_1\ j=1: \frac{\partial}{\partial\theta_1}J(\theta_0,\theta_1) = \frac{1}{m}\sum_{i=1}^{m}\left(h_\theta(x^{(i)})-y^{(i)}\right)\cdot x^{(i)}$$

注意：gradient 是所有 examples 的关于**一个参数偏导数值**的累加！

所有 example 偏导数的累加的均值就是，该参数的方向梯度 Δ

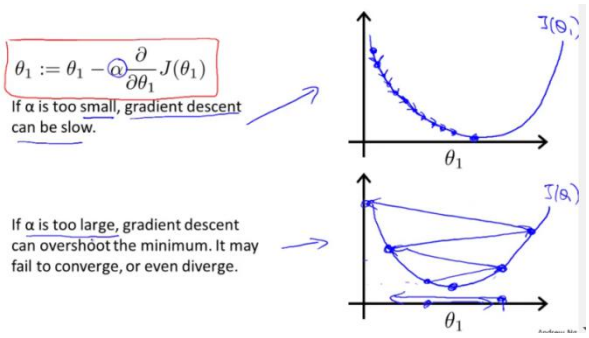
可视化例子：

参数每次调整，使得 cost function 减少，得到的直线更好拟合。



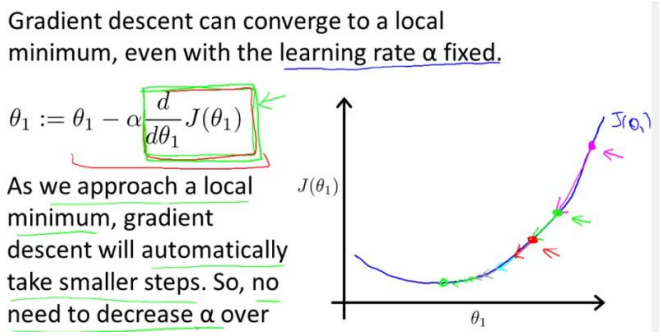
2.2 learning rate

multiply the gradient by a scalar known as the learning rate
LR 不能太大，会错过最小点
如果 α 过小，需要很多步才达底部。太大会跳过底部，甚至发散



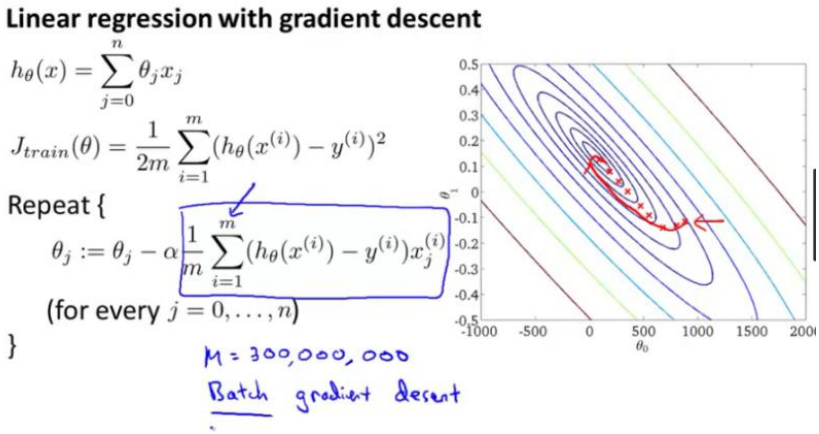
参数更新的值 step 会越来越小，因为 LR 不变，gradient 会越来越小(平稳，不陡峭)

如果到达了局部最小，参数就不再变了，因为导数为 0



2.3 batch gradient descent

batch gradient descent:



每次都要加总所有数据的 derivative

当数据量很多时，普通 batch gradient descent 计算量太大了

batch is the total number of examples you use to calculate the gradient in a single iteration

每一 step，都 look all example $\sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})$

所有 examples 都用到了，偏导数之和求梯度
有些 gradient descent，不是 batch，只 look small subset。
gradient descent adapt better to large datasets !!!

2.4 Stochastic Gradient Descent

adapt better to large datasets !!
A very large batch may cause even a single iteration to take a very long time to compute.
Stochastic gradient descent (SGD) takes this idea to the extreme--**only a single example (a batch size of 1)** per iteration. Given enough iterations, SGD works but is very noisy. The term "stochastic" indicates that the one example comprising each batch is chosen at random.

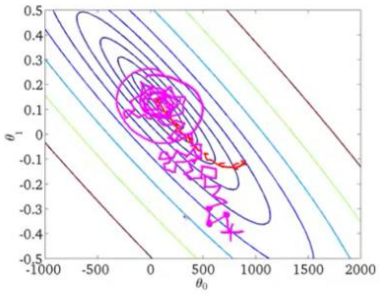
Mini-batch stochastic gradient descent (mini-batch SGD) reduces the amount of noise in SGD but is still more efficient than full-batch.

Stochastic Gradient Descent
也就是 **cost function** 不是所有 example 的 error 的 sum，
而是每一个 example 的 error

Stochastic gradient descent

1. Randomly shuffle (reorder) training examples

2. Repeat {
   for $i := 1, \ldots, m$ {
   $\theta_j := \theta_j - \alpha(h_\theta(x^{(i)}) - y^{(i)})x_j^{(i)}$
   (for every $j = 0, \ldots, n$)
   }
   }



会比较曲折，有时会达不到 global optima，会不断接近它，在它附近震荡，但是够用了！

2.5 Mini-Batch Gradient Descent

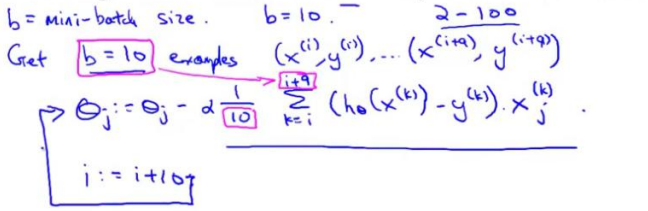
much more faster than stochastic

Mini-batch gradient descent

Batch gradient descent: Use all m examples in each iteration

Stochastic gradient descent: Use 1 example in each iteration

Mini-batch gradient descent: Use b examples in each iteration



b typical size: 2-100

假设取 10，遍历 example，每 10 个 example 更新一次参数，derivative 是 10 个的 sum

Mini-batch gradient descent

Say $b = 10, m = 1000$.

Repeat {

for $i = 1, 11, 21, 31, \ldots, 991$ {

$$\theta_j := \theta_j - \alpha \frac{1}{10} \sum_{k=i}^{i+9} (h_\theta(x^{(k)}) - y^{(k)}) x_j^{(k)}$$

(for every $j = 0, \ldots, n$)

}

}

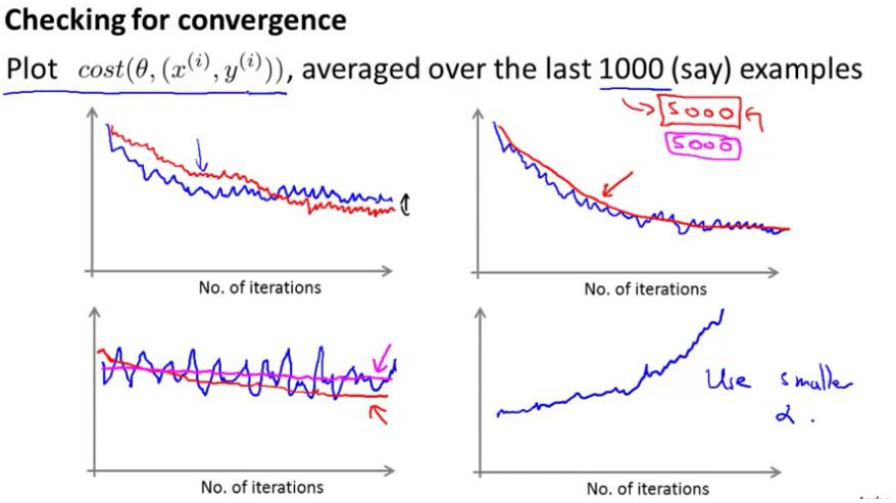
i 第一次是从 1 开始，第二次是从 11 开始…

速度比 stochastic 更快的原因是 vecterization 向量计算

计算 derivative 时 10 个 example 可以向量并行计算，效率更高

2.6 Stochastic Gradient Descent Convergence

batch size:



1 红色代表：如果用小的 learning rate，会更加收敛。接近 global optima

2 如果 batch size 1000 增加至 5000，会得到更平滑的曲线

3 波动比较大时代表算法没有学习，**试着增加 1000-5000**，可能会得到**平滑趋势线**，或 change learing rate 或 feature

4 上升的话，用更小的 learing rate

之前都是在 global optima 附近震荡，如果想 **actually converage to global optima**，需要在**每次迭代时降低 learning rate**，**震荡会越来越小，直到收敛 optima**

Learning rate α is typically held constant. Can slowly decrease α over time if we want θ to converge. (E.g. $\alpha = \frac{\text{const1}}{\text{iterationNumber} + \text{const2}}$)

但需要花时间去选择这两个常数

2.7 batch & step & peroids

hyperparameters:

1 **steps**, which is the total number of one training **iterations**(进行几次 batch). **One step 计算一次 batch gradient descent**

2 **batch size**, which is the number of examples for a **single step**. For example, the batch size for SGD is 1.

3 **epoch** 是几次 train set

4 **iteration** 是一个 train set 的 step

5 periods, which controls the granularity of **reporting**. For example, if `periods` is set to 7 and `steps` is set to 70

$$total\ number\ of\ trained\ examples = batch\ size * steps$$

$$number\ of\ training\ examples\ in\ each\ period = \frac{batch\ size * steps}{periods}$$

2.8 Tensorflow tf.estimator

TensorFlow consists of the following two components:

- a graph protocol buffer
- a runtime that executes the (distributed) graph

analogous to the Java compiler and the JVM

三.Linear Regression with Multiple Variables

1 Multipule feature

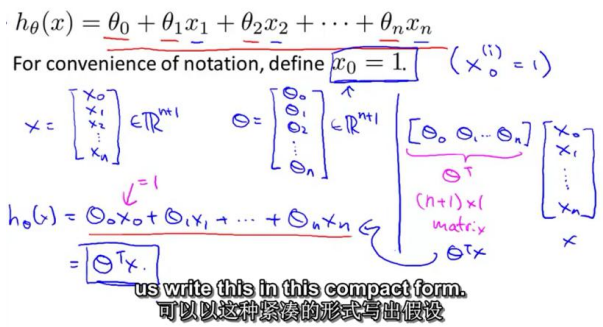
$x_j^{(i)}$ = value of feature j in the i^{th} training example
$x^{(i)}$ = the column vector of all the feature inputs of the i^{th} training example
m = the number of training examples
$n = \left|x^{(i)}\right|$; (the number of features)

$x^{(i)}$ 第 i 个行

Hypothesis:

$$h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_n x_n$$

哑变量 x0



向量都是列向量
多元线性回归 hypothesis 向量形式

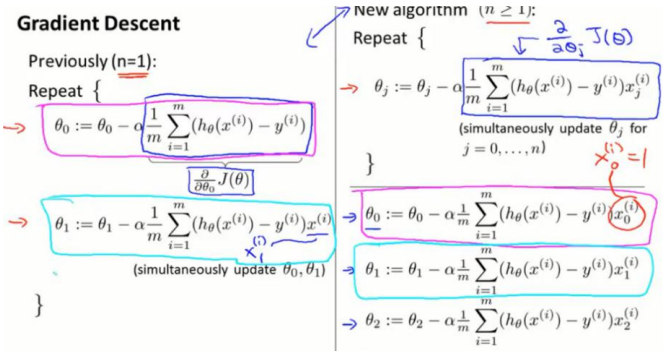
2 Gradient Descent for Multiple Variables

2.1 Gradient Descent

cost fun 和之前一样，只是多了更多参数
对每个参数分别求偏导数更新参数!!
要加总所有 exmples 的 cost，才能让我们的模型更好的 repesent all data

Cost function:

$$J(\theta_0, \theta_1, \ldots, \theta_n) = \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2$$



2.2 Gradient Descent in Practice - Feature Scaling

Scaling means converting floating-point feature values from their natural range (for example, 100 to 900) **into a standard range** (for example, 0 to 1 or -1 to +1). If a feature set consists of only a single feature, 不需要 scaling . If, however, a feature set consists of multiple features, then feature scaling provides the following benefits:

- 1 **Helps gradient descent converge more quickly.**
- 2 Helps avoid the **"NaN trap,"** in which one number in the model becomes a NaN(e.g., when a value exceeds the floating-point precision limit during training)
- 3 **Helps the model learn appropriate weights for each feature.**

Without feature scaling, the model **will pay too much attention to the features having a wider range.**

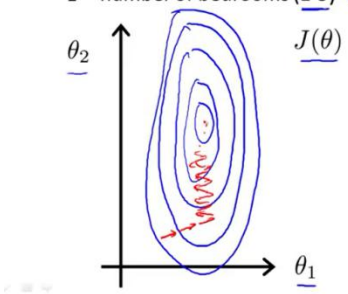
You don't have to give every floating-point feature exactly the same scale. Nothing terrible will happen if Feature A is scaled from -1 to +1 while Feature B is scaled from -3 to +3. calculate the Z score of each value:

Feature Scaling

Idea: Make sure features are on a similar scale.

E.g. x_1 = size (0-2000 feet^2)

x_2 = number of bedrooms (1-5)



如果两个变量的数量级别差别大

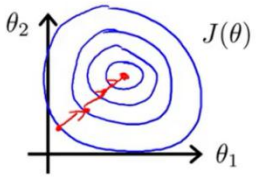
值越大，**参数取值范围会越偏小，形成狭窄** contour cost functiuon

因此，另一个参数 gradient descent 需要很长的时间。

we should **scale the features** to the similar scale

$$x_1 = \frac{\text{size (feet}^2\text{)}}{2000}$$

$$x_2 = \frac{\text{number of bedrooms}}{5}$$



会更直接的寻找到最下值，converage faster

calculate the Z score of each value:

$$scaledvalue = (value - mean)/stddev.$$

Scaling with Z scores means that most scaled values will be between -std and +std, but a few values will be a little higher or lower than that range.

2.3 Gradient Descent in Practice - Model Tuning

1 If the training has not converged, try **running it for longer.**

2 If the training error decreases too slowly, **increasing the learning rate** may help it decrease faster.

But sometimes the exact opposite may happen if the learning rate is too high.

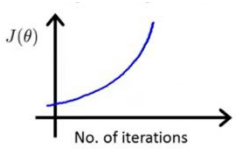
3 If the training error varies wildly, try **decreasing the learning rate.**

Lower learning rate plus larger number of steps or larger batch size

is often a good combination.

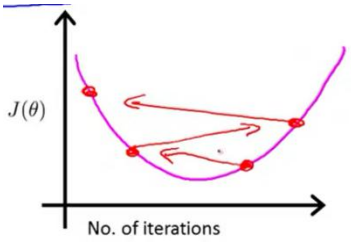
Very small batch sizes can also cause instability. First try larger values like 100 or 1000, and decrease until you see degradation.

It has been proven that if learning rate α is sufficiently small, then J(θ) will decrease on every iteration.
when not working, try smaller alpha。

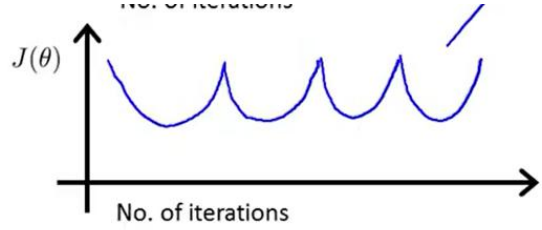


Gradient descent not working.
Use smaller α.

原因在于



还有下面这种：一直波动



For sufficiently small α, $J(\theta)$ should decrease on every iteration.
But if α is too small, gradient descent can be slow to converge.

To choose α, try

$\ldots, 0.001, 0.003, 0.01, 0.03, 0.1, 0.3, 1, \ldots$

每次 3 倍的增加 0.003 的 3 倍 roughly 是 0.01

四 特征工程+数据清洗

1 Feature Engineering (Representation)

Adding and improving its features.

1 Mapping Raw Data to Features vector

2 Mapping string values：machine learning models typically represent each categorical feature as a separate Boolean value，每个 string 一个单独的 feature 代表是不是这个 string----也就是 onehot encoding

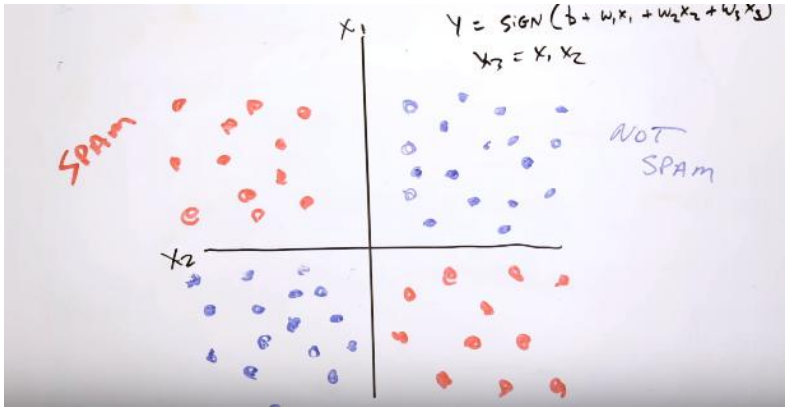
2 Feature Crosses/Synthetic feature+Polynomial Regression 高阶多项式回归(第三章多元回归进阶)

对于非线性的数据!!!!

Crossing two (or more) features is a clever way to learn non-linear relations using a linear model.

2.1 Feature cross

feature cross 可以帮助建立更复杂的模型,更好对现有的数据进行预测或分类，是的 loss 更好的下降，如果模型不理想 loss 太高，可以尝试 cross feature



线性模型分不开，但增加一个 feature，两个 feature 相乘！就可以

线性模型里 learn 非线性的东西，完成非线性数据分类

通过 Feature Crosses(feature product) (synthetic feature)

X3 为正是蓝，为负时红

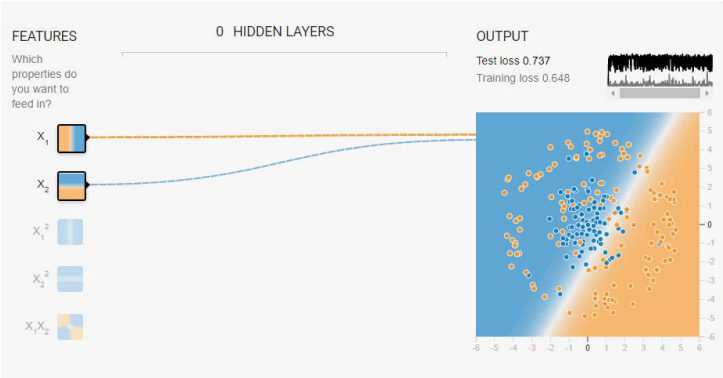
A linear algorithm can learn a weight for w3 , although w3 encodes nonlinear information, you don’t need to change how the linear model trains to determine the value of w3.

最后的结果可能是，x1，x2 的权重是 0，只有 x3 有权重，因为只需要 x3 就可以分开数据

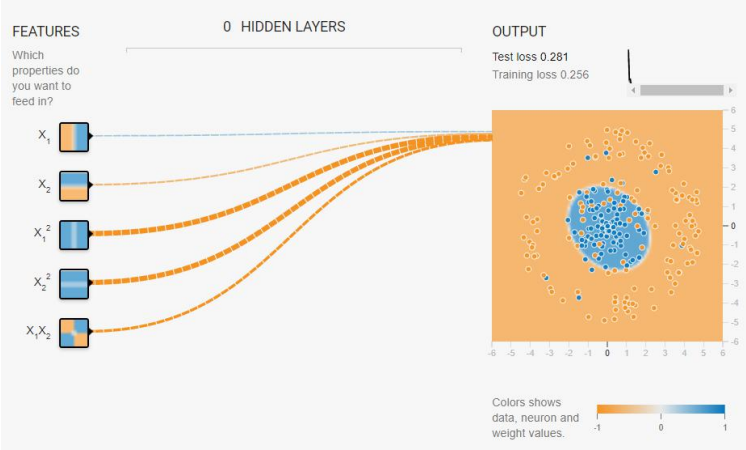
Kinds of feature crosses

We can create many different kinds of feature crosses. For example:

- `[A X B]` : a feature cross formed by multiplying the values of two features.
- `[A x B x C x D x E]` : a feature cross formed by multiplying the values of five features.
- `[A x A]` : a feature cross formed by squaring a single feature.



只用 x1x2 线性模型，loss 不会降低!!



假如 feature cross 后，就能够大致的分开!!!

In practice, machine learning models seldom cross continuous features. However, machine learning models do frequently cross one-hot feature vectors

实际，我们很少对连续数据进行 feature cross!!!

更多对离散数据的 onehot 数据进行 cross!! 因为 feature 更有意义!! 更好的理解

2.2 Crossing One-Hot Vectors

In practice, machine learning models **seldom cross continuous features**. However, machine learning models do **frequently cross one-hot feature vectors**

For example, two features: **country and language**.

country=USA, country=France or language=English, language=Spanish. if you do a **feature cross of these one-hot encodings**, you get binary features that can be interpreted as logical conjunctions, such as: **country:usa AND language:spanish** （新 feature 的含义）

例：

binned_latitude = [0, 0, 0, 1, 0]

binned_longitude = [0, 1, 0, 0, 0]

binned_latitude X binned_longitude 是一个 25-element one-hot vector (24 zeroes and 1 one). 上面 vector 每个元素的两两组合！！代表 and

The single 1 in the cross identifies **a particular conjunction of latitude and longitude**. Your model can then **learn particular associations about that conjunction.**

例子：

Now suppose our model needs to predict how satisfied dog owners will be with dogs based on two features:

- Behavior type (barking, crying, snuggling, etc.)
- Time of day

If we build a feature cross from both these features:

```
[behavior type X time of day]
```

then we'll end up with vastly more predictive ability than either feature on its own. For example, if a dog cries (happily) at 5:00 pm when the owner returns from work will likely be a great positive predictor of owner satisfaction. Crying (miserably, perhaps) at 3:00 am when the owner was sleeping soundly will likely be a strong negative predictor of owner satisfaction.

新的特征，比之前的任意两个特征更 powerful!!

对于离散数据：

连续数据也可以先离散化再这样做

In our problem, if we just use the **feature latitude** for learning, the model might learn that city blocks at a particular latitude (or within a particular range of latitudes since we have bucketized it) are more likely

to be expensive than others. Similarly for the feature **longitude**. However, if we **cross longitude by latitude**, the **crossed feature represents a well defined city block**. If the model learns that certain city blocks are more likely to be more expensive than others, **it is a stronger signal than two features considered individually.**

Currently, the feature columns **API only supports discrete features for crosses**. To cross two continuous values, like latitude or longitude, we can **bucketize them.**

1 先 bucketize

2 转成 onehotencoding

3 onehotencoding feature cross

for example, that longitude was bucketized into 2 buckets,
while latitude has 3 buckets, If we cross
the latitude and longitude features (supposing,), we actually get **six crossed binary features(onehot encoding)**_. Each of these features will get its own separate weight when we train the model.

原来的 featuure 也一同加进模型里面

代码：

```
long_x_lat = tf.feature_column.crossed_column(
set([bucketized_longitude, bucketized_latitude]), hash_bucket_size=1000)
```

linear model 的 powerful thing!!

1 Thanks to stochastic gradient descent, linear models can be trained efficiently. (an efficient way to train on massive-scale data sets.)

Linear learners scale well to massive data.

2supplementing scaled linear models with **feature crosses** 可以完成非线性的数据

Using feature crosses on massive data sets is one efficient strategy for learning highly **complex models**. Neural networks provide another strategy.

We can **combine** multiple features into one. For example, we can combine x1 and x2 into a new feature x3 by taking x1· x2.

Housing prices prediction

$h_\theta(x) = \theta_0 + \theta_1 \times frontage + \theta_2 \times depth$

Area

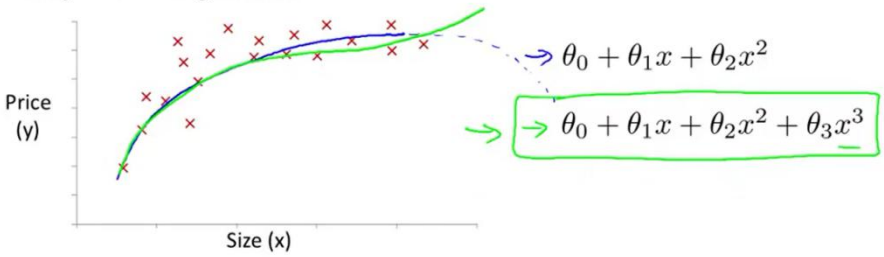
x = frontage * depth

$h_\theta(x) = \theta_0 + \theta_1 x$

↑ land area

我们真正关心的是面积，因此 we create a new feature
两个 feature 相乘得到的二次项，这就是 polynomial 的来源。

Polynomial regression



$$h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3$$
$$= \theta_0 + \theta_1 (size) + \theta_2 (size)^2 + \theta_3 (size)^3$$

$x_1 = (size)$
$x_2 = (size)^2$
$x_3 = (size)^3$

2.3 feature selection

Models with fewer features use fewer resources and are easier to maintain

What's the best performance you can get with just 2 or 3 features?

A correlation matrix shows pairwise correlations, both for each feature compared to the target and for each feature compared to other features.

$$\rho_{X,Y} = \frac{\operatorname{cov}(X,Y)}{\sigma_X \sigma_Y}$$

where:

- cov is the covariance
- σ_X is the standard deviation of X
- σ_Y is the standard deviation of Y

The formula for ρ can be expressed in terms of mean and expectation. Since

$$\operatorname{cov}(X,Y) = \mathrm{E}[(X - \mu_X)(Y - \mu_Y)],$$ [5]

the formula for ρ can also be written as

$$\rho_{X,Y} = \frac{\mathrm{E}[(X - \mu_X)(Y - \mu_Y)]}{\sigma_X \sigma_Y}$$

```
correlation_dataframe = training_examples.copy()
correlation_dataframe["target"] = training_targets["median_house_value"]

correlation_dataframe.corr()
```

	latitude	longitude	housing_median_age	total_rooms	total_bedrooms	population	households	median_income	rooms_per_person	target
latitude	1.0	-0.9	0.0	-0.0	-0.1	-0.1	-0.1	-0.1	0.1	-0.1
longitude	-0.9	1.0	-0.1	0.0	0.1	0.1	0.1	-0.0	-0.1	-0.0
housing_median_age	0.0	-0.1	1.0	-0.4	-0.3	-0.3	-0.3	-0.1	-0.1	0.1
total_rooms	-0.0	0.0	-0.4	1.0	0.9	0.9	0.9	0.2	0.1	0.1
total_bedrooms	-0.1	0.1	-0.3	0.9	1.0	0.9	1.0	-0.0	0.0	0.0
population	-0.1	0.1	-0.3	0.9	0.9	1.0	0.9	-0.0	-0.1	-0.0
households	-0.1	0.1	-0.3	0.9	1.0	0.9	1.0	0.0	-0.0	0.1
median_income	-0.1	-0.0	-0.1	0.2	-0.0	-0.0	0.0	1.0	0.2	0.7
rooms_per_person	0.1	-0.1	-0.1	0.1	0.0	-0.1	-0.0	0.2	1.0	0.2
target	-0.1	-0.0	0.1	0.1	0.0	-0.0	0.1	0.7	0.2	1.0

1 we'd like to have features that are strongly correlated with the target.

2 We'd also like to have features that **aren't so strongly correlated with each other, so that they add independent information.**

Use this information to try **removing features**. You can also **try developing additional synthetic features**, such as ratios of two raw features.

下面两个 feature 不错

"median_income",

"latitude",

2.4 Qualities of Good Features(包括缺失值)

1 Avoid rarely used discrete feature values
好的 string feature **至少要多余 5 个值，越多越好**
值太少，model **比较难 learn how this feature value relates to the label.**
除非这个 label 就是由这个 feature 决定，一般情况下都是由多个 feature，不会被少数简单的 feature 决定，建模也没意义了

2 feature 不能含有太多的 outlier!!! cut 或 transform

看分布!!! 通过变化将极端的分布去掉!!
outlier，做标准化的时候会影响 scale 的结果!!

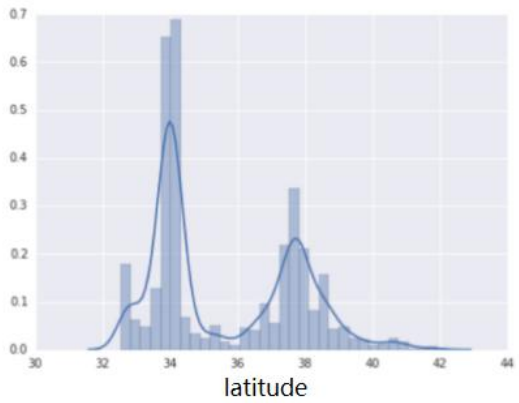
3 缺失值或特殊值处理
quality_rating: 0.82
quality_rating: 0.37
对于 absence data 填-1
quality_rating: -1 **这个值没意义!!**

convert the feature into **two features:**

- 1 One feature holds only **quality ratings(还是原来的数值)**, never **magic values. say -1**
- 2 One feature holds **a boolean value** indicating **whether or not** a `quality_rating` **was absent** Give this boolean feature a name like `is_quality_rating_defined`. (0 或 1)

2.5 Binning 分箱技术 Bucketized

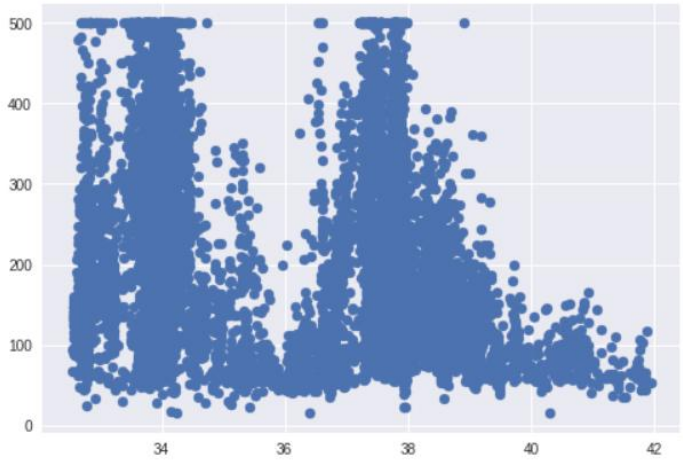
房子数量在维度的分布



In the data set, `latitude` **is a floating-point value**. However, it doesn't make sense to represent `latitude` as a **floating-point feature** in our model. That's because **no linear relationship exists between latitude and housing values**. 维度和房价并没有线性关系!!!

```
plt.scatter(training_examples["latitude"], training_targets["median_house_value"])
```

```
<matplotlib.collections.PathCollection at 0x7f13524f29d0>
```



所以加到线性模型没有什么意义，需要转换成有意义的：**synthetic features**

Try creating some synthetic features that do a better job with latitude.

1 create a feature that maps latitude to a value of |latitude - 38|, and call this **distance_from_san_francisco.**

Or

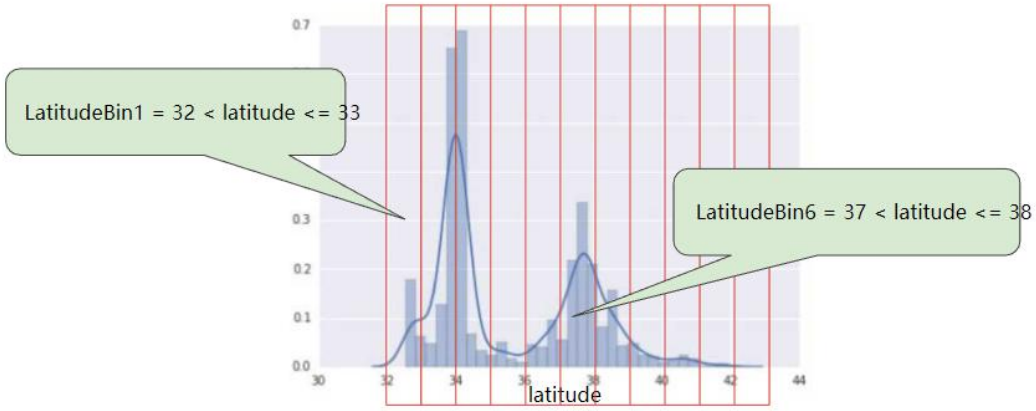
2 break the space into 10 different bins latitude_32_to_33, latitude_33_to_34, etc., 然后 onehot!

Use the **correlation matrix** to help guide development

What's the best validation performance you can get?

And yet, individual latitudes probably are a pretty good predictor of house values.

To make latitude a helpful predictor, let's divide latitudes into "bins" as suggested by the following figure:



然后将 13 个 value 热编码

Bucketized (Binned) Features

Bucketization is also known as binning.

We can bucketize `population` into the following 3 buckets (for instance):

- `bucket_0` (< 5000): corresponding to less populated blocks
- `bucket_1` (5000 - 25000): corresponding to mid populated blocks
- `bucket_2` (> 25000): corresponding to highly populated blocks

Given the preceding bucket definitions, the following `population` vector:

```
[[10001], [42004], [2500], [18000]]
```

becomes the following bucketized feature vector:

```
[[1], [2], [0], [1]]
```

The feature values are now the bucket indices. Note that these indices are considered to be discrete features. Typically, these will be further converted in one-hot representations as above, but this is done transparently.

3 Cleaning Data

3.1 detect bad data

you must also **detect bad data** in the aggregate. **Histograms** are a great mechanism for visualizing your data in the aggregate. In addition, getting statistics like the following can help:

- Maximum and minimum
- Mean and median
- Standard deviation

3.2 Scaling feature values

是为了让所有 feature in 同一个 scale，一个量级！！！使得 feature 地位相同，不能因为值大就对模型影响大

Scaling means converting floating-point feature values from their **natural range** (for example, 100 to 900) into a **standard range** (for example, 0 to 1 or -1 to +1). If a feature set consists of **only a single feature**, 不需要 scaling . If, however, a feature set consists of **multiple features**, then feature scaling provides the following benefits:

- 1 Helps gradient descent converge more quickly.
- 2 Helps avoid the "NaN trap," in which one number in the model becomes a NaN(e.g., when a value exceeds the floating-point precision limit during training)
- 3 Helps the model learn appropriate weights for each feature. Without feature scaling, **the model will pay too much attention to the features having a wider range.**

You don't have to give every floating-point feature exactly the same scale. Nothing terrible will happen if Feature A is scaled from -1 to +1 while Feature B is scaled from -3 to +3. However, your model will react poorly if Feature B is scaled from 5000 to 100000.

3.3 Handling extreme outliers

1 先看 feature 分布：

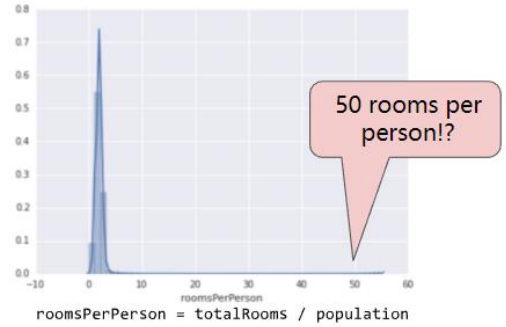


Figure 4. A verrrrry lonnnnnng tail.

长尾分布，很多异常值！

2 先做 log，但还有 tail，再换个方法!!!

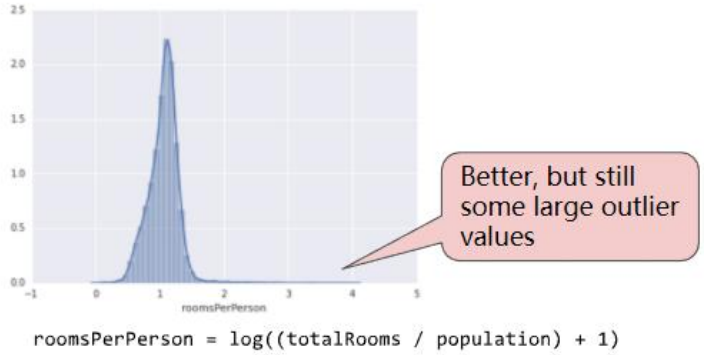


Figure 5. Logarithmic scaling still leaves a tail.

3 Clipping feature 大于 4 的全部变为 4（clipping gradient）

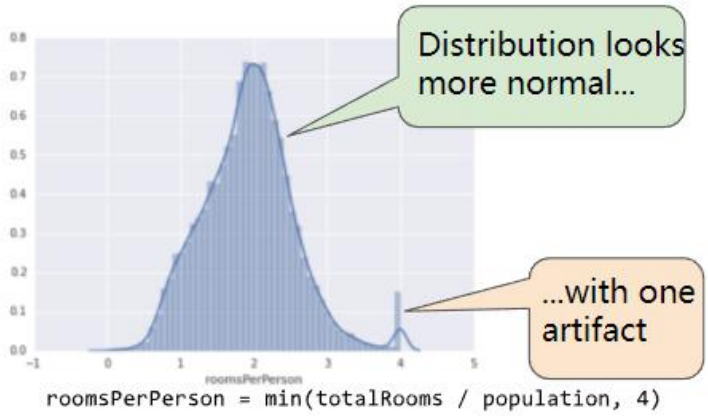


Figure 6. Clipping feature values at 4.0

Clipping the feature value at 4.0 doesn't mean that we ignore all values greater than 4.0. Rather, it means that all values that were greater than 4.0 now become 4.0. This explains the funny hill at 4.0. Despite that hill, the scaled feature set is now more useful than the original data.

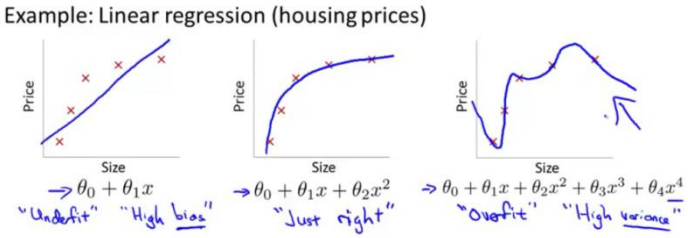
```
california_housing_dataframe["rooms_per_person"] = (
california_housing_dataframe["rooms_per_person"]).apply(lambda x: min(x, 5))
```

五. Regularization

If we use a model that is too complicated, such as one with too many crosses, we give it the opportunity to fit to the noise in the training data(overfit), often at the cost of making the model perform badly on test data.

在保证模型准确率的前提下，不能上模型太过复杂，前面的方法就是剔除特征，另外方法正则化，正则化可以让我们在保留全部特征的前提下，模型不过于复杂(相当于自动 feature selection)
Regularization can help avoiding overfitting, even we have a lot of features
In addition to avoiding overfitting, the resulting model will be more efficient.

1 The Problem of Overfitting(fit to the noise of training dat)



1 high bias(underfit)：have a strong preconception or bias 偏见，that price vary linearly with size, 但事实相反，poor fit the data
bias：对 data 有强烈偏见，不考虑 data
2 high variance(overfit)：the hypotesis to complex with too much variable, have not enough data to fit
variance:模型随 data 变化波动大

Overfitting: If we have too many features, the learned hypothesis may fit the training set very well ($J(\theta) = \frac{1}{2m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2 \approx 0$), but fail to generalize to new examples (predict prices on new examples).

regularization 核心思想是：不要太相信我们的 train set!!!

太相信 train set 会影响 generlization!!

1 early stopping （the train）再 converge 之前停止

2 prefer the small weight!!! penerlize the weight

因为 train 的过程本身就是 train loss：focus on train set，mimimize the empirical error。我们需要在 train 的过程中再加点东西 penerlty，prevent overfitting by penalizing complex models:

$$\text{minimize}(\text{Loss}(\text{Data}|\text{Model}) + \lambda\, \text{complexity}(\text{Model}))$$

在学习 weight 过程中，让 weight 尽量的小(simple model),需要折中！

2 model complexity function(L2 regularization)

1 Model complexity as a function of the weights of all the features in the model(a feature weight with a high absolute value is more complex than a feature weight with a low absolute value.)

2 Model complexity as a function of the total number of features with nonzero weights

L_2 regularization

对于第一种情况：We can quantify complexity using the L_2 regularization formula:

$$L_2 \text{ regularization term} = ||\boldsymbol{w}||_2^2 = w_1^2 + w_2^2 + \ldots + w_n^2$$

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

L_2 regularization has the following effect on a model

1 Encourages weight values toward 0 (but not exactly 0)

让参数都尽量小，会得到最好的近似最准确最低阶的 polynomial!!! 其他的参数接近 0.

have small value of parameter, make the function more smooth, 减少 variation 避免了 overfitting。

- 2 Encourages the mean of the weights toward 0, with a normal (bell-shaped or Gaussian) distribution(weight 正态分布！).

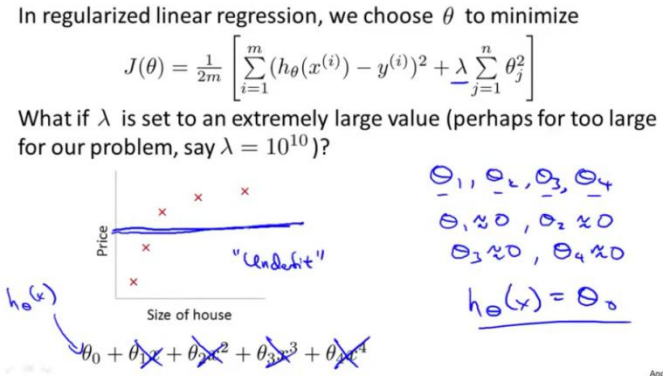
$$\text{minimize}(\text{Loss}(\text{Data}|\text{Model}) + \lambda\, \text{complexity}(\text{Model}))$$

-

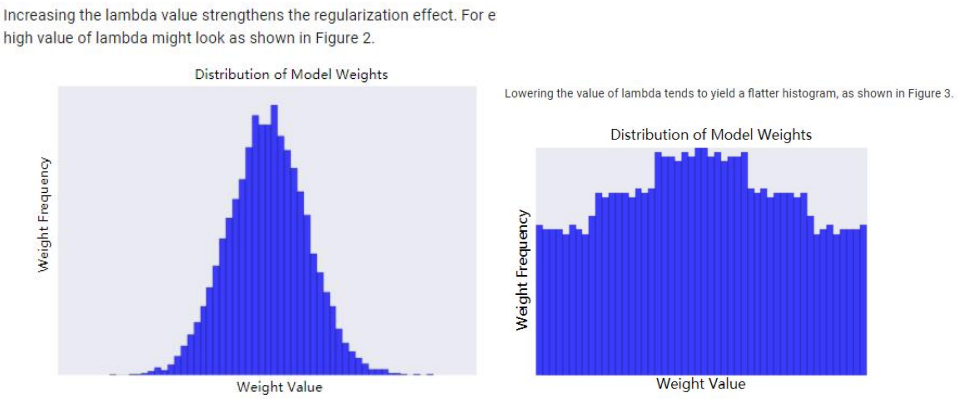
Lambda

λ越大，the prarameter 越 small

如果λ太大，参数都接近0，就会underfit:



λ对参数的影响



lambda 越大，weight 值为 0 的频率越大

lambda value is to strike the right balance between **simplicity** and **training-data fit:**

1 lambda value is too high, your model will be **simple,** but you run the risk of underfitting your data. Your model won't learn enough about the training data to make useful predictions.

2 lambda value is too low, your model will be more **complex,** and you run the risk of **overfitting** your data. Your model will learn too much about the particularities of the training data, and won't be able to generalize to new data.

The ideal value of lambda produces a model that **generalizes well to new, previously unseen data**

3 Regularized Linear Regression (L2 的原理实质)

加入 L2 regularization term 后的 cost function

Regularized linear regression

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

$$\min_\theta J(\theta)$$

bias 不加 regularization term

计算 gradient

Repeat {

$$\theta_0 := \theta_0 - \alpha\frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})x_0^{(i)}$$

$$\theta_j := \theta_j - \alpha\left[\left(\frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})x_j^{(i)}\right) + \frac{\lambda}{m}\theta_j\right] \qquad j \in \{1,2...n\}$$

}

将第二项合并：合并

$$\theta_j := \theta_j(1 - \alpha\tfrac{\lambda}{m}) - \alpha\tfrac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})x_j^{(i)}$$

$1-\alpha\frac{\lambda}{m}$ 是一个非常接近 1 的数值，say 0.98，会先 shrinking the θ a little bit。

后面那一项和原来的一样，唯一不同在于，每次迭代前，先让 θ shrink!!!

4 Nomal Equation

To add in regularization, the equation is the same as our original, except that we add another term inside the parentheses:

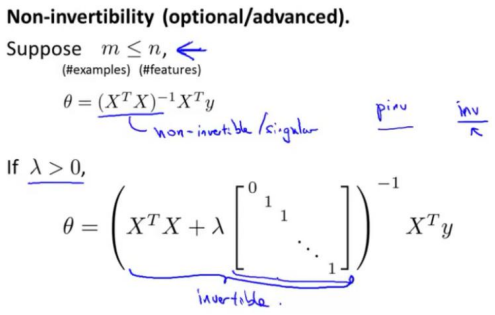
$$\theta = \left(X^TX + \lambda\cdot L\right)^{-1}X^Ty$$

$$\text{where } L = \begin{bmatrix} 0 & & & & \\ & 1 & & & \\ & & 1 & & \\ & & & \ddots & \\ & & & & 1 \end{bmatrix}$$

L 是(n+1)*(n+1)

如果不可逆

Recall that if m ≤ n, then X^TX is non-invertible. However, when we add the term λ· L, then X^TX + λ· L becomes invertible.



4 L1 Regularization (Sparsity):

One way to **reduce model complexity** is to **use a regularization function that encourages weights** to be exactly zero(sparsity).
任何 regularization 都是这个目的!!!
For linear models such as regression, a zero weight is equivalent to not using the corresponding feature at all. In addition to 1 **avoiding overfitting**, the model also will be 2 **more efficient**.

L1 regularization is a good way to increase sparsity.

好处：去掉 noise coefficient(会引起 overfit)，避免 overfitting

In a high-dimensional sparse vector，尤其是做完 cross 的 feature, it would be nice to encourage weights to **drop to exactly 0** where possible. A weight of exactly 0 essentially removes the corresponding feature from the model. **Zeroing out features** will save RAM and may **reduce noise** in the model.(其实就是将一些没用的，没意义的 feature 去除)

penalizes the count of non-zero coefficient values in a model

L2 会让参数小，但是不会让参数接近 0

L0 regularization ：An alternative regularization term that **penalizes the count of non-zero coefficient values** in a model. Increasing this count would only be justified if there was a sufficient gain in the model's ability to fit the data. Unfortunately, it would **turn our convex optimization problem into a non-convex optimization problem that's NP-hard.**

L_1 regularization serves as an **approximation to L_0**, but has the advantage of being convex and thus efficient to compute

L_2 and L_1 penalize weights differently:

- L_2 penalizes weight^2.
- L_1 penalizes |weight|.

Consequently, L_2 and L_1 have different derivatives:

- The derivative of L_2 is 2 * weight.
- The derivative of L_1 is k (value of weight).

QUESTION: L1 L2 的解释

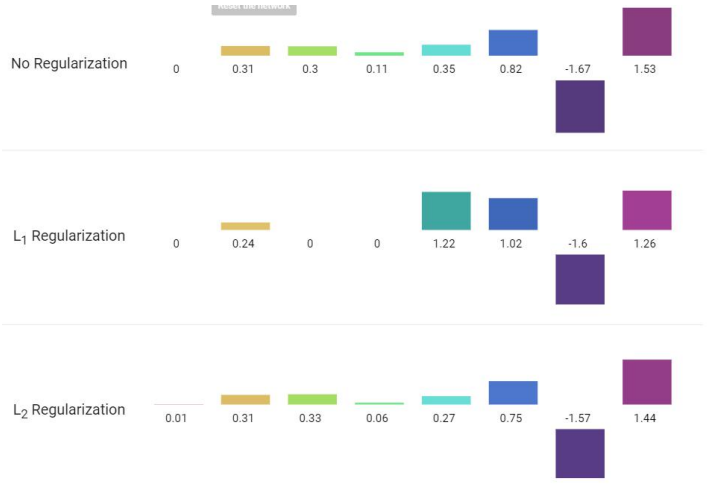
You can think of the **derivative of L_2** as a force that **removes x% of the weight every time**. As Zeno knew, **even if you remove x percent of a number *billions of times*, the diminished number will still never quite reach zero.** At any rate, L_2 does not normally drive weights to zero.

You can think of the **derivative of L_1** as a force that **subtracts some constant** from the weight every time. However, thanks to absolute values, L_1 has a discontinuity at 0, which causes subtraction results that cross 0 to become zeroed out. For example, if subtraction would have forced a weight from +0.1 to -0.2, L_1 will set the weight to exactly 0. Eureka, L_1 zeroed out the weight.

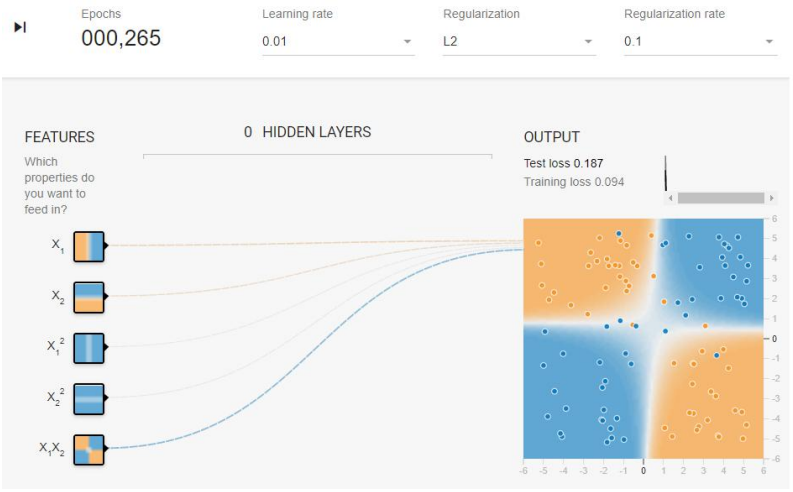
L_1 regularization—**penalizing the absolute value** of all the weights—turns out to be quite efficient for wide models.

Note that this description is true for a one-dimensional model.

三种不同的训练结果：



L2

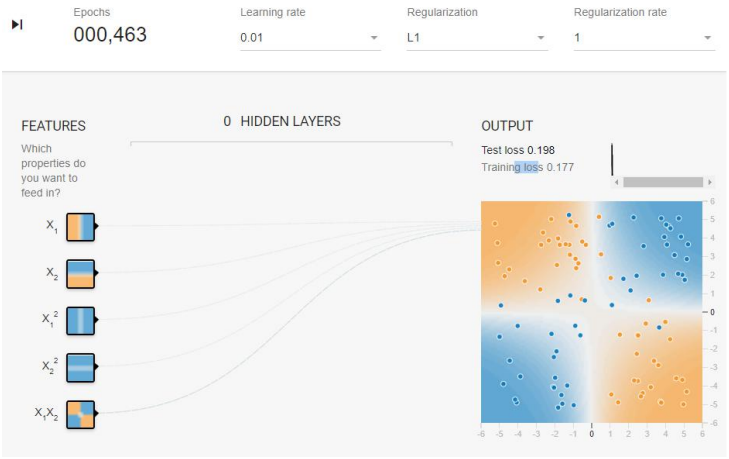


L1

L2---》L1

1 reduces the delta between test loss and training loss.泛化能力增强，减少过拟合

2 dampens all of the learned weights. 现都变细了，weight 变小了



3 Increasing the L1 regularization rate lambda generally dampens the learned weights; however, if the regularization rate goes too high, the model can't converge and both losses are very high. weight 太小，以至于模型有效 feature 很小，模型太简单！！学不到东西

Calculate the size of a model

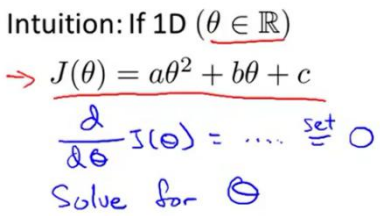
Apply L1 regularization to reduce the size of a model by increasing sparsity

六. Computing Parameters Analytically

1 Normal Equation(解析方法)求参数

Method to solve for parameter analytically 解析的(其实就是求函数极值！)，而不是 iteration，just one step

如果 θ 是一个数，也就是单个参数，可以对 J 求导，令导数=0，求极值即可



吧

如果 θ 是一个向量，也就是多个参数：

$\theta \in \mathbb{R}^{n+1}$ $J(\theta_0, \theta_1, \ldots, \theta_m) = \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2$

$\frac{\partial}{\partial \theta_j} J(\theta) = \cdots \overset{\text{set}}{=} 0$ (for every j)

Solve for $\theta_0, \theta_1, \ldots, \theta_n$

分别求每个 θ 的 partial derivative ，令为 0，求出每个参数值

Examples: $m = 4$.

x_0	Size (feet²) x_1	Number of bedrooms x_2	Number of floors x_3	Age of home (years) x_4	Price ($1000) y
1	2104	5	1	45	460
1	1416	3	2	40	232
1	1534	3	2	30	315
1	852	2	1	36	178

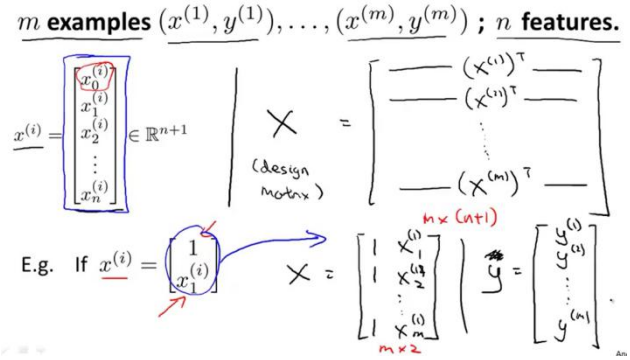
$X = \begin{bmatrix} 1 & 2104 & 5 & 1 & 45 \\ 1 & 1416 & 3 & 2 & 40 \\ 1 & 1534 & 3 & 2 & 30 \\ 1 & 852 & 2 & 1 & 36 \end{bmatrix}$ $y = \begin{bmatrix} 460 \\ 232 \\ 315 \\ 178 \end{bmatrix}$

m x (n+1) m-dimensional vector

$\theta = (X^T X)^{-1} X^T y$

固定公式：对 cost fun 求矩阵形式的偏导数推导得到

$$\theta = (X^T X)^{-1} X^T y$$



上标代表行，下标代表列(feature)
每一行是一个 n+1 维向量
X design matrix，其实就是数据的原始结构

但因为我们 $x^{(i)}$，每一个记录都是向量形式，要转成原始结构！转置

$$\theta = (X^T X)^{-1} X^T y$$

2 两种方法对比

Gradient Descent	Normal Equation
Need to choose alpha	No need to choose alpha
Needs many iterations	No need to iterate
O (kn^2)	O (n^3), need to calculate inverse of $X^T X$
Works well when n is large	Slow if n is very large

Normal Equation 只对回归有用，因为需要一个 y 值(y 不是向量)

In practice, when do regression, number of feature less than 1000 use normal equation。
when n exceeds 10,000 it might be a good time to go from a normal solution to an iterative process.

for other learning algritm，而且一些复杂的学习算法，无法用 normal equation 计算

3 Normal Equation Noninvertibility

$$\theta = (X^T X)^{-1} X^T y$$

pinv： $X^T X$ 即使 non-invertible

What if $X^T X$ is non-invertible?

- Redundant features (linearly dependent).
  E.g. x_1 =size in feet^2
  x_2 =size in m^2
  $1_m = 3.28$ feet
  $x_1 = (3.28)^2 x_2$

矩阵内有线性关联，矩阵不可逆，需删除一个 feature

- Too many features (e.g. $m \leq n$).
  - Delete some features, or use regularization.

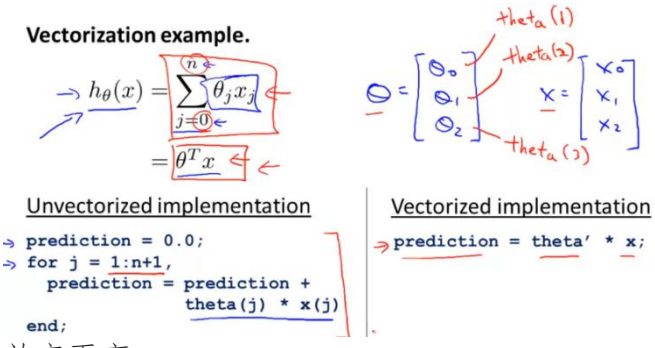
m 小于 n，也不可逆，删除些，或者正则化

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.

词汇:
involved 复杂的 go into the detail 深入细节

4 Vecterization 化(将迭代转成矩阵相乘！)

线性代数，矩阵计算，效率更高，代码更简单



效率更高

Gradient descent

$$\theta_j := \theta_j - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)} \qquad \text{(for all } j\text{)}$$

$j = 0, 1, 2$

- $\theta_0 := \theta_0 - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_0^{(i)}$
- $\theta_1 := \theta_1 - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_1^{(i)}$
- $\theta_2 := \theta_2 - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_2^{(i)}$

Simultaneous updates.

例：

$$u = \begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix}, \; v = \begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix}, \; w = \begin{bmatrix} w_1 \\ w_2 \\ w_3 \end{bmatrix}.$$

Your code implements the following:

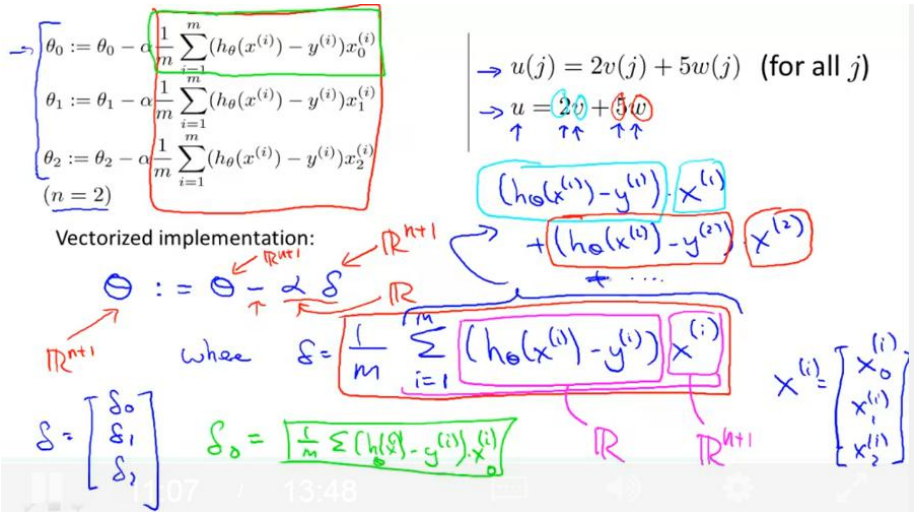
for j = 1:3,

u(j) = 2 * v(j) + 5 * w(j);

end

vectorize the code 难点

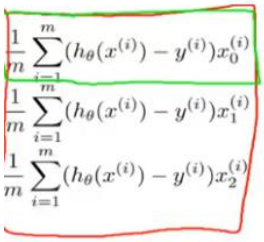
u = 2 * v + 5 * w



只有同时更新参数，才能保证 $\frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})$ 是常数，因为 example 都一样参数也固定

上面推导的证明：δ 是 Rn+1 维向量

可以每一行先展开



$$(h_\theta(x^{(1)})-y^{(1)})x_0^{(1)}+(h_\theta(x^{(2)})-y^{(2)})x_0^{(2)}+\dots$$

$$(h_\theta(x^{(1)})-y^{(1)})x_1^{(1)}+(h_\theta(x^{(2)})-y^{(2)})x_1^{(2)}+\dots$$

$$(h_\theta(x^{(1)})-y^{(1)})x_2^{(1)}+(h_\theta(x^{(2)})-y^{(2)})x_2^{(2)}+\dots$$

……..

每一列都是一个相同实数 R，乘以 $x^{(i)}$ 向量，向量运算：

u=2*v+3*x+6*y　每个向量都是 Rn+1 维度，刚好组成 δ

v 就是第一个 example 向量

词汇：初级的 elementary

七 Logistic Regression 二分类

1 Classification and Representation

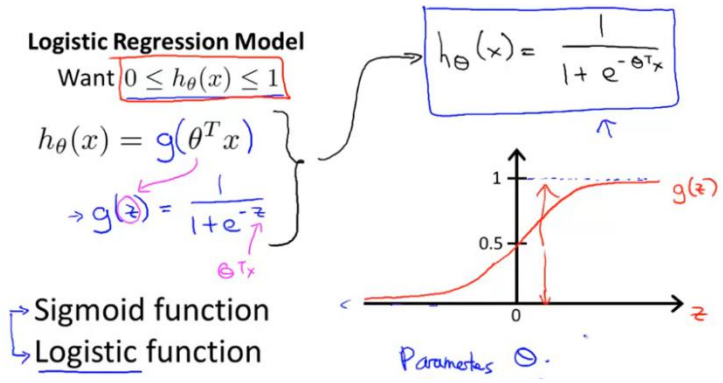
1.1 linear logistics Classification

在 linear regression 基础上进行延伸：其实就是将 y 属于 R 的值域，**压缩到 y 是 0,1，这样的好处是可以用 cross entropy 作为 loss，因为对于分类 L2 不好**

Practically speaking, you can use the **returned probability** in either of the following two ways:

- 1 "As is" 某件**事情(正样本)发生的概率**，(regression) label 是概率值
- 2 Converted to a **binary category(加个 treshold)**. (classification) label 是 0 或 1

hypothesis representation:



$$h_\theta(x) = p(y=1 \mid x;\theta)$$

sigmoid function:

$$y' = \frac{1}{1+e^{-(z)}}$$

z is also referred to as the log-odds（几率的 log！）

$$z = log(\frac{y}{1-y})$$

1.2 Decision Boundary(logistics 实质)+ threshold +线性分类+几何意义！

Suppose predict "$y=1$" if $h_\theta(x) \ge 0.5$

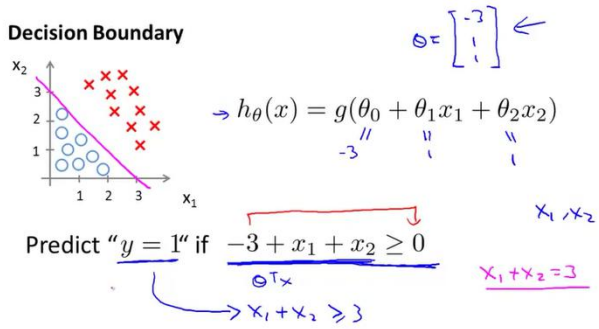
predict "$y=0$" if $h_\theta(x) < 0.5$

有了 threshold 就成了 binary 分类了！

进一步，将判定条件可以写成 y=1, $\theta^T x$>0　y=0, $\theta^T x$<0

$\theta^T x$=0 是判定边界 Decision boundary，为正 y=1，为负 y=0

几何意义：实质还是 boundary 起到了分类作用!!!



假设我们的线性模型，最终得到如下参数：-3,1,1

就是一条直线：

边界上方分类：y=1

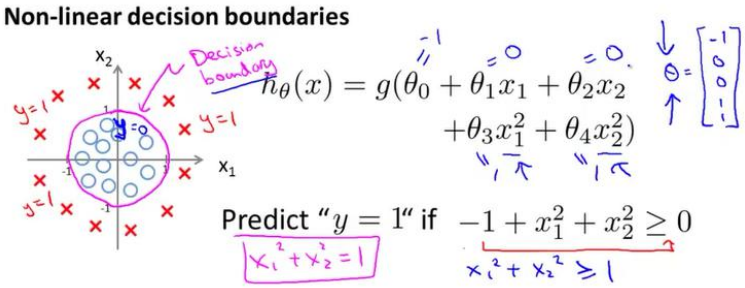
参数的作用是调节移动直线位置

$\theta^T x = 0$就是 Decision Boundary

线性分类指的是 boundary 是直线

1.3 非线性 logistics 分类(data 非线性可分)

由 boundary 的形状决定!!! boundary 起到了分类作用!!

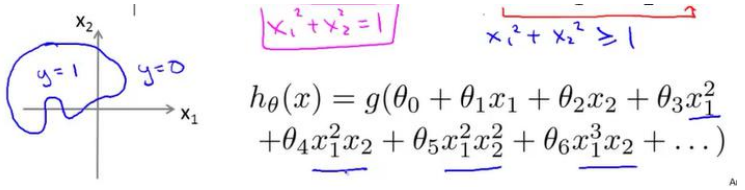


假设用多项式拟合得到如下参数：-1 0 0 1 1

可以看出 boundary 最终是个 circle

Decision Boundary 跟数据没关系，只跟假设本身及参数有关系!! 给定了参数就决定了 boundary。数据只是帮我们 fit，找到 boundary，但 boundary 已经存在。

我们可以得到更加**复杂的** Decision Boundary



f(x)无论是什么函数(Polynomial)，f(x)=0 就是 boundary，将数据点划分 0,1

1.4 logistics regression 的实质

就是找到一个 Decision boundray，这个边界是由原函数决定！(由原函数画出)边界内，边界外将数据分成两类!!!

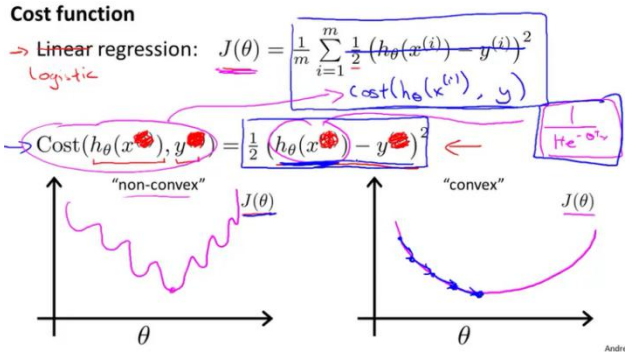
2 Logistic regression model

2.1 Cost Function(定制)

如果还用之前的 linear regression 用 **L2 loss**，doesn't do a great job at penalizing misclassifications when the **output is interpreted as a probability 不合适了**

对于 y 输出是概率的不适合用 L2，gradicent 不能很好收敛，for instance:

there should be a huge difference with outputing a probability of 0.9 vs 0.9999, but L2 loss doesn't strongly differentiate these cases.



如果用 L2 loss 此时因为，$h_\theta(x) = \frac{1}{1+e^{\theta^T x}}$，所以 $J(\theta)$ 是 non-convex 非凹函数，不能用 gradient descent，不保证全局最小。

因此为了更好的 penalizes misclassfing：下面的更合适

Log Loss:

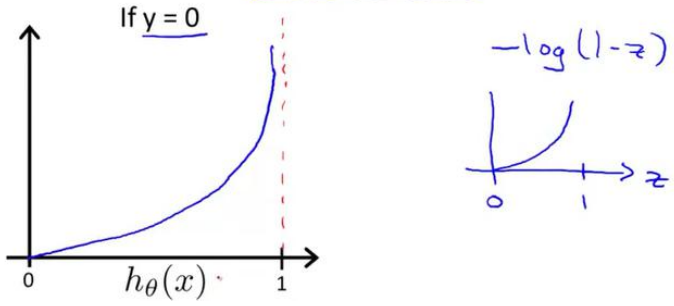
$$LogLoss = \sum_{(x,y)\in D} -y \cdot log(y_{pred}) - (1-y) \cdot log(1-y_{pred})$$

The equation for Log Loss is closely related to Shannon's Entropy measure from Information Theory. It is also the negative logarithm of the likelihood function, assuming a Bernoulli distribution of y. Indeed, minimizing the loss function **yields a maximum likelihood estimate**

2.2 cost function 定义思路

Logistic regression cost function

$$\text{Cost}(h_\theta(x), y) = \begin{cases} -\log(h_\theta(x)) & \text{if } y = 1 \\ -\log(1 - h_\theta(x)) & \text{if } y = 0 \end{cases}$$



这个 $\text{Cost}(h_\theta(x^{(i)}), y^{(i)})$ derive from 统计学的 principle of most likelyhood estimation
巧妙的数学模型!!!

2.3 Simplified 合并 cost function and gradient descent

Logistic regression cost function

$$J(\theta) = \frac{1}{m} \sum_{i=1}^{m} \text{Cost}(h_\theta(x^{(i)}), y^{(i)})$$

$$\text{Cost}(h_\theta(x), y) = \begin{cases} -\log(h_\theta(x)) & \text{if } y = 1 \\ -\log(1 - h_\theta(x)) & \text{if } y = 0 \end{cases}$$

总的 cost function 需要每个 example 累加 cost
又因为 y 只能取 0,1
可以合并：

$$\begin{aligned} J(\theta) &= \frac{1}{m} \sum_{i=1}^{m} \text{Cost}(h_\theta(x^{(i)}), y^{(i)}) \\ &= -\frac{1}{m} [\sum_{i=1}^{m} y^{(i)} \log h_\theta(x^{(i)}) + (1 - y^{(i)}) \log (1 - h_\theta(x^{(i)}))] \end{aligned}$$

$J(\theta)$ 是个 convex 函数

coding:

```
label=np.array([1,0,0,1,1,0])
output=np.array([4,-7,12,24,-9,10])
h=1/(np.exp(-output)+1)
h
```

```
array([  9.82013790e-01,   9.11051194e-04,   9.99993856e-01,
         1.00000000e+00,   1.23394576e-04,   9.99954602e-01])
```

```
cross_entropy=label*np.log(h)+(1-label)*np.log(1-h)
cross_entropy
```

```
array([ -1.81499279e-02,  -9.11466454e-04,  -1.20000061e+01,
        -3.77513576e-11,  -9.00012340e+00,  -1.00000454e+01])
```

```
loss=-1*np.mean(cross_entropy)
loss
```

```
5.1698727232823218
```

loss func 结果是向量(每个 example 的)

```
label*np.log(h)+(1-label)*np.log(1-h)
```

```
array([ -1.81499279e-02,  -9.11466454e-04,  -1.20000061e+01,
        -3.77513576e-11,  -9.00012340e+00,  -1.00000454e+01])
```

h(x)和 y 都是向量(每个 example 一个 h(x),y)

为了找到 θ, 只需 minimize $J(\theta)$

推导:

minimize $J(\theta)$ to find the θ:

Gradient Descent

$$J(\theta) = -\frac{1}{m}[\sum_{i=1}^{m} y^{(i)} \log h_\theta(x^{(i)}) + (1 - y^{(i)}) \log (1 - h_\theta(x^{(i)}))]$$

Want $\min_\theta J(\theta)$:

Repeat {

$$\theta_j := \theta_j - \alpha \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)}$$

(simultaneously update all θ_j)

}

求偏导:

$\frac{\partial}{\partial \theta_j} J(\theta) = \frac{1}{m} \sum_{I}^{M} (h_\theta(x^{(i)}) - y^{(i)}) x^{(i)}$ 带入上面式子:

Gradient Descent

$$J(\theta) = -\frac{1}{m}[\sum_{i=1}^{m} y^{(i)} \log h_\theta(x^{(i)}) + (1 - y^{(i)}) \log (1 - h_\theta(x^{(i)}))]$$

Want $\min_\theta J(\theta)$:

Repeat {

$$\theta_j := \theta_j - \alpha \underbrace{\sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)}}_{\text{(simultaneously update all } \theta_j)}$$

}

和我们之前线性回归的形式一模一样，但 $h_\theta(x^{(i)})$ 不同了

之前和现在：

3 Regularized Logistic Regression

Regularization is extremely important in logistic regression modeling. Without regularization, the model will become completely overfit。the asymptotic nature of logistic regression would keep driving loss towards 0 in high dimensions. driving the weights for each indicator feature to +infinity or -infinity. **This can happen in high dimensional data with feature crosses**

因此需要：

- L_2 regularization.
- **Early stopping, that is, limiting the number of training steps or the learning rate.**

We can regularize this equation by adding a term to the end:

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

尽管 high level dedgree polynimial，最终得到的 decision boundary smooth

the sum explicitly exclude 排除忽略了 θ_0，不用参与正则化，其他参数在梯度下降，更新参数时需要正则化!!!

因此要 separate： 求偏导

Gradient descent

Repeat {

$$\theta_0 := \theta_0 - \alpha \frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})x_0^{(i)}$$

$$\theta_j := \theta_j - \alpha \left[\frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})x_j^{(i)} + \frac{\lambda}{m}\theta_j\right] \quad (j = 1, 2, 3, \ldots, n)$$

}

$\theta_1 \cdots \theta_n$

$\frac{\partial}{\partial \theta_j} J(\theta)$

$h_\theta(x) = \frac{1}{1 + e^{-\theta^T x}}$

4 Advanced Optimization Algorithm 优化算法

FTRL Optimization Algorithm

High dimensional linear models benefit from using **a variant of gradient-based** optimization called **FTRL**. This algorithm has the benefit of **scaling the learning rate differently for different coefficients**, 会自动调整 learning rate

which can be useful if some features rarely take non-zero values (it also is well suited to support L1 regularization). We can apply FTRL using the FtrlOptimizer.

tf.train.FtrlOptimizer(learning_rate=learning_rate)

其他 advance 优化算法 minimize $J(\theta)$, 来求得 θ_j

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

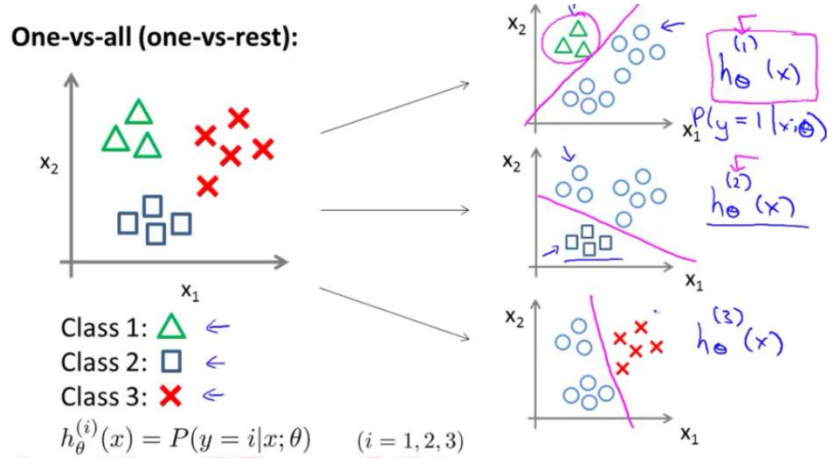
5 logistics Multi-class claasification One vs All (rest)

logistics regression 为我们做分类提供了基础!!

One vs All 分类算法:

用 logistics 解决多分类问题

Since y = {0,1...n}, we divide our problem into n+1 binary classification problems



其实就是一个神经网络：input+3 个输出节点

训练三个 logistics 二元分类器，分别代表三个类型，将 x 带入三个 $h_\theta^{(i)}(x)$，选值最大的来就是该类型。
神经网络的好处是：不需要训练三次，只需要和一起训练就可以了

One-vs-all

Train a logistic regression classifier $h_\theta^{(i)}(x)$ for each class i to predict the probability that $y = i$.

On a new input x, to make a prediction, pick the class i that maximizes

$$\max_i h_\theta^{(i)}(x)$$

带入每个分类器，选最大的

$$y \in \{0, 1 \ldots n\}$$
$$h_\theta^{(0)}(x) = P(y=0|x;\theta)$$
$$h_\theta^{(1)}(x) = P(y=1|x;\theta)$$
$$\cdots$$
$$h_\theta^{(n)}(x) = P(y=n|x;\theta)$$
$$\text{prediction} = \max_i(h_\theta^{(i)}(x))$$

6 logistics 为分类问题打开了一个世界，提高了基础(如何确定 threshold？)

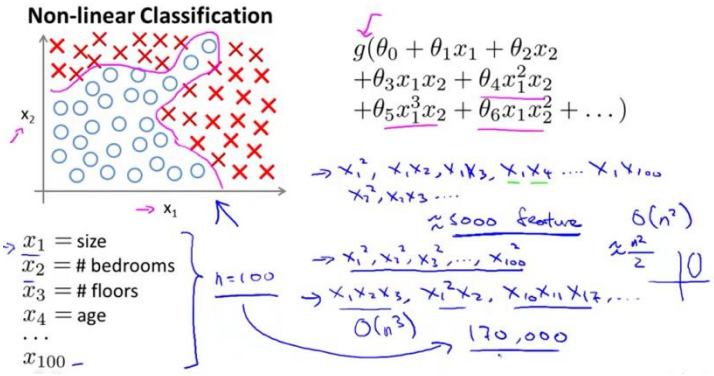
参看 evaluate model 部分

八 Neural Networks 的本质

1.Motivation

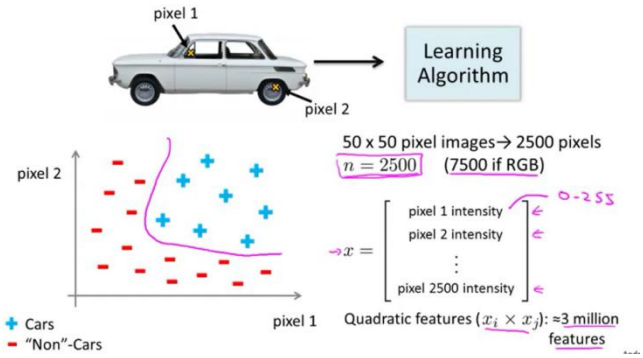
1.1 传统 non-liear(feature 合成) hypothesis weekness

传统分类非线性 logistics regression 的弱点：



对于两个 feature，logistics 可能没啥问题

对于**多个 feature，polynomial 需要包含太多的组合项**，不然就不能很好的 fit data，但这样**计算量太大！**



图像识别：选取两个像素点作为 feature

但对于一个图像，有 $x_1 \times x_2$ 个 feature，太多了，logistics 不太行！

2 Neural Networks

a more powerful non-linear hypothesis

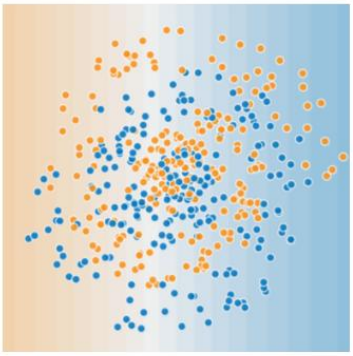


Figure 2. A more difficult nonlinear classification problem.

对于更复杂的非线性分类(更复杂的 data)，feature cross 不行了， can't be solved with a linear model，只能用非线性分类器神经网络了!!

神经网络各个圆圈 node:就是 input 和 weight 相乘产生的 output

Even with Neural Nets, some amount of **feature engineering** (特征合成)is often needed to achieve best performance. Try **adding in additional cross product features** or other transformations like **sin(X1) and sin(X2)**.

we will get a better model, and output surface any smoother!!

2.1 Model Representation

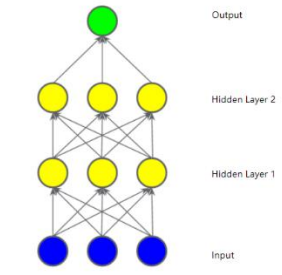


Figure 5. Graph of three-layer model.

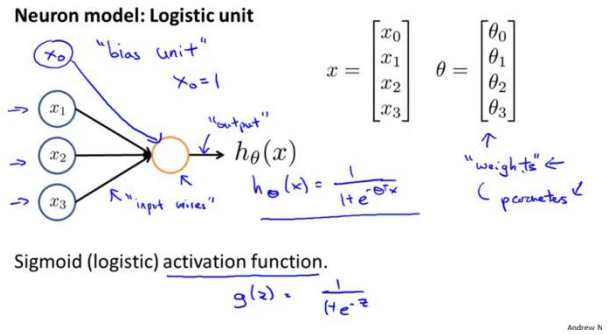
这样仍然是一个 **linear model:**
output is still a linear combination of its inputs.上面式子化简后，得到还是 input 的线性组合

Activation Functions 转化为 non-linear model
pipe each hidden layer node through a nonlinear function.
We now can model very **complicated relationships** between the inputs and the outputs

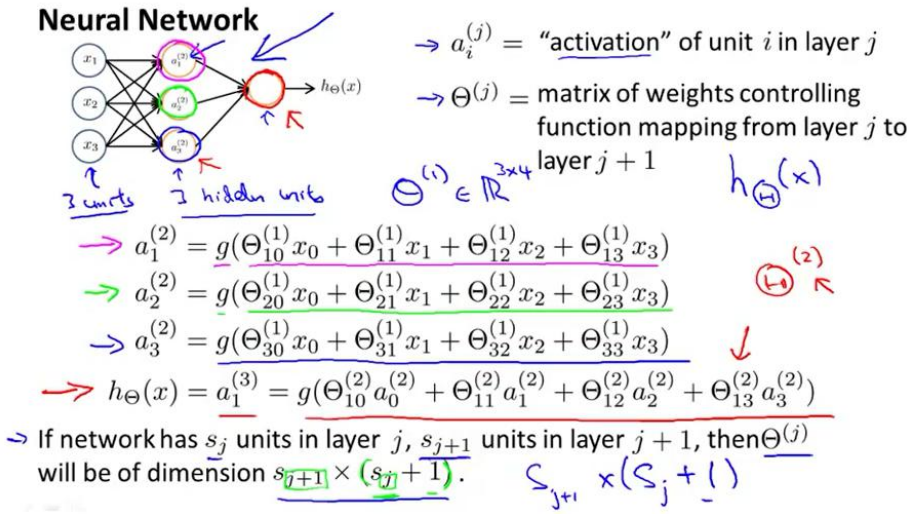
最简单的神经网络就是 logistics classification!!!
神经网络其实本质就是多个二分类 logistics!!! 但多了很多层，多了很多

feature 变换

back propagation:其实就是让我们更方便的做 gradient descent,更新参数!
可以多个 logistics 一起训练!!



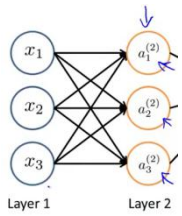
四层:



3 Relation betwen Neural Network and logistic regression(区别) 迁移学习

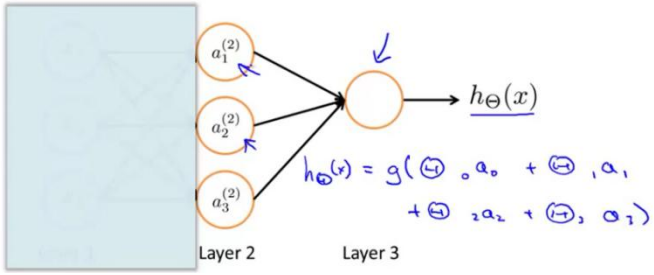
神经网络其实本质就是多个二分类 logistics!!! 但多了很多层，多了很多 feature 变换

1 一层的神经网络其实本质就是多个二分类 logistics!!!
2 多层 Neural Network(多个 neuron)唯一不同的是: logistic regression 用原始 raw 的 feature,而 Neural Network 用的 计算过的 feature a(activatioon value)。

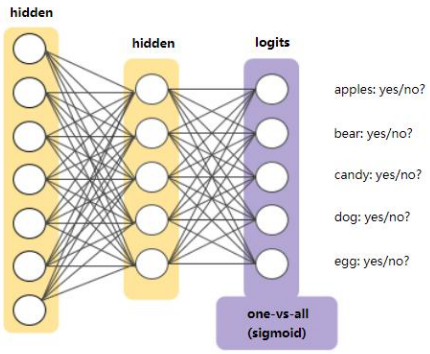


这也是神经网络的强大之处!!!!

Neural Network learning its own features



feature 是通过 input 自己学习得到，learn its own feature，create more complex feature，form more powerful hypothesis!!!



其实 deep network 本质其实就是多个 logistic，但所有 logistics model 共享前面一套参数，只是最后一层参数不同!!! 导致每个 output unit 输出不同的类别

因此迁移学习也是只更改最后一层，或最后几层的参数!!!

4 linear classification 神经网络的实质(boundary)

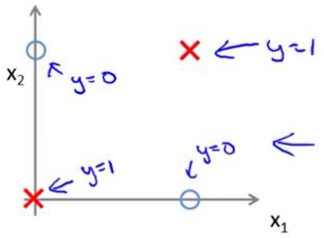
一层神经网络：核函数就是线性

A Xor B :只有其中一个是 1，才为 True

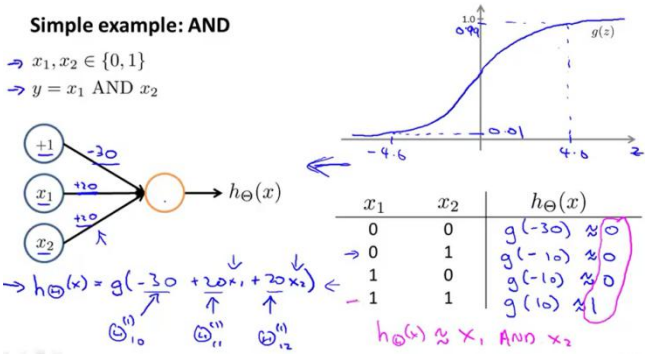
A Xnor B: not A or B :NOT(A Xor B)

Xnor:

x_1, x_2 are binary (0 or 1).



And function implement:



核函数就是边界：

输入层和参数，就确定了边界 boundary：$-30+20x_1+20x_2=0$

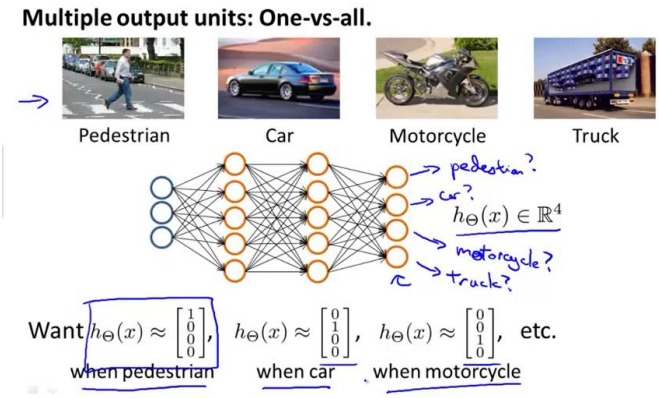
边界不一定都过原点，因此需要有截距项!!!

多层神经网络，核函数是复杂的函数，解决非线性分类，其实和 cross feature 的目的一样，将 kernel 变成非线性，完成非线性的分类!!!!

最终得到的**这些参数**，确定了 nonlinear boundary

5 Multiclass Classification softmax

Multiclass Classification Neural Network 本质是 One versus ALL method



图像识别，分成 4 类

需要 4 个 output unit，也是输出 0 或 1，但每个输出对应是否为某一类？

每一个输出是每一类的概率!!!

softmax : generalize of **single logistics regression**

$$p(y=j|\mathbf{x}) = \frac{e^{(\mathbf{w}_j^T\mathbf{x}+b_j)}}{\sum_{k\in K} e^{(\mathbf{w}_k^T\mathbf{x}+b_k)}}$$

Softmax 的假设： each example is a member of exactly one class.

一个 example 只有一个类别

6 One Label vs. Many Labels

Softmax assumes that each example is a member of exactly one class. 当一个 example 多个 label 时：

You may not use Softmax.

You must rely on multiple logistic regressions.

7 Softmax Options&random sample

训练一个分类器：

1Full Softmax

2random sample

Candidate sampling means that **Softmax calculates a probability for all the positive labels but only for a random sample of negative labels**. For example, if we are interested in **determining whether an input image is a beagle or a bloodhound**, we don't have to provide probabilities for every non-doggy example. 当 train 图片类别数量较少时，完整 Softmax 代价很小，但随着 train 图片类别数量的增加，它的代价会变得极其高昂。候选采样可以提高处理具有大量类别的问题的效率。

The motivation for candidate sampling is a **computational efficiency** win from not computing predictions for all negatives.

train: candidate sample

词汇：

resurge 复活 process 过程 处理 analogous to 类似于 term 术语 不是 二是 rather than.., is

九 ML System Design 和模型评估(Model Evaluation 最佳实践)

1 Evaluating a Learning Algorithm/ Hypothesis

Debugging a learning algorithm:
Suppose you have implemented regularized linear regression to predict housing prices.

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

However, when you test your hypothesis on a new set of houses, you find that it makes unacceptably large errors in its predictions. What should you try next?

如果预测 test error 很大，如何办？
bias and variance :model performance 表现差的原因！

- Get more training examples
- Try smaller sets of features
- Try getting additional features
- Try adding polynomial features $(x_1^2, x_2^2, x_1x_2, \text{etc.})$
- Try decreasing λ
- Try increasing λ

到底改选哪一个？

2 Generalization

refers to your model's ability to adapt properly to new, previously unseen data, drawn from the same distribution as the one used to create the model.
model 应该尽量简单 ，模型太复杂 overfit!!! Low loss, but still a bad model?does not generalize well to new data.

The fundamental tension of machine learning is between fitting our data well, but also fitting the data as simply as possible.

3 train/validation/test split and model selection+Bias/Variance (model performance 表现差的原因！)

how to detect underfit or overfit? when we have 2 features we can draw graph, but with too much, we can ‘t。
the way is: separate data
用 empirical evaluation to judge a model's ability to generalize to new data.

- training set—a subset to train a model.
- test set—a subset to test the mode

假设:

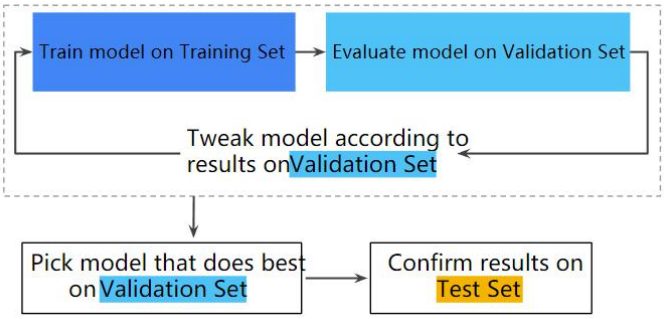
1 **i.i.d. assumption**: We draw examples **independently and identically** (i.i.d) at random from the distribution. In other words, examples don't influence each other.

2 The distribution is **stationary**; that is the distribution doesn't change within the data set

- Consider a model that chooses ads to display. The i.i.d. assumption would be violated if the model **bases its choice of ads on what ads the user has previously** seen. example 直接相互影响，有特定顺序模式
- Consider a data set that contains retail sales information for a year. User's purchases change **seasonally,** which would violate **stationarity.**
- if the key assumptions of supervised ML are not met, then we **lose important theoretical guarantees** on our ability to predict on new data.

when spliting data, we should make sure a random choose, should randomize the data in advance!

validation **and model selection**



1. **Pick the model** that **does best (matrix)**on the **validation set.**

   对于 regression 表现是 L2 Error(均方差)

   对于 classification 表现是分类准确率

   不能只用 test 来 select model，这个过程中 test data 信息已经利用了!!

   generalization 只能对于 unsee data

2. Double-check that model against the **test set.**

   用来调参，如果 test 表现不好，说明 overfit on validation set

   validation 也需要 iid

   If we don't randomize the data properly before creating training and validation splits, then we may be in trouble if the data is given to us in some sorted order, which appears to be the case here.

   数据分布就不一样，train 分布在 0-100，validation 分布在 100-120

Tip

Test sets and validation sets "wear out" 用坏 with repeated use. That is, the more you use the same data to make decisions about hyperparameter settings or model improvements, the less confidence you'll have that these results actually generalize to **new, unseen data**. Note that validation sets typically wear out more slowly than test sets.

If possible, it's a good idea to **collect more data** to "refresh" the test set and validation set. Starting anew is a great reset.重新开始

需要分布相同!! 只是数量不同，所以数据需要先 shuffle!!

Training targets summary:

	median_house_value
count	12000.0
mean	206.8
std	116.0
min	15.0
25%	118.8
50%	179.9
75%	264.5
max	500.0

Validation targets summary:

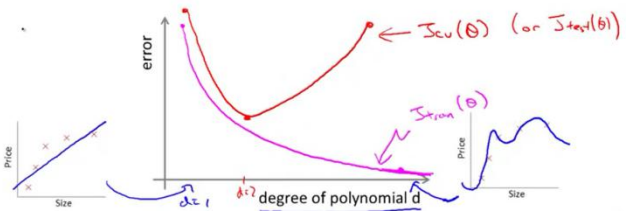
	median_house_value
count	5000.0
mean	208.6
std	116.1
min	32.5
25%	121.4
50%	181.3
75%	266.4
max	500.0

Bias vs. Variance 这两个是 model performance 表现差的原因!
也是提高 performance 的关键!!

Bias/variance

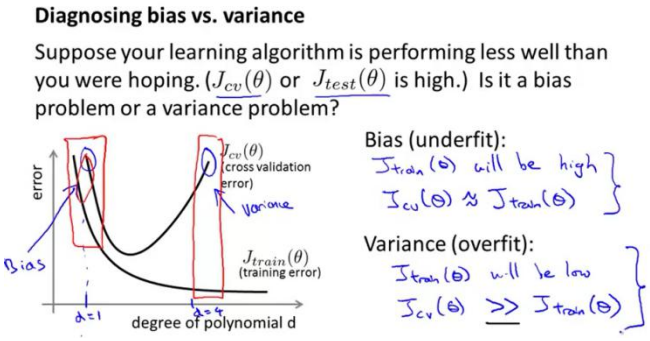
Training error: $J_{train}(\theta) = \frac{1}{2m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2$

Cross validation error: $J_{cv}(\theta) = \frac{1}{2m_{cv}}\sum_{i=1}^{m_{cv}}(h_\theta(x_{cv}^{(i)}) - y_{cv}^{(i)})^2$ (or $J_{test}(\theta)$)



train error 随着 d 增加，**而一直下降**，最后 overfit

cv error 作为 test，test 每个 d model 的 error，是先下降，**后上升。**
model select point



难点!!! 要记住这个图。

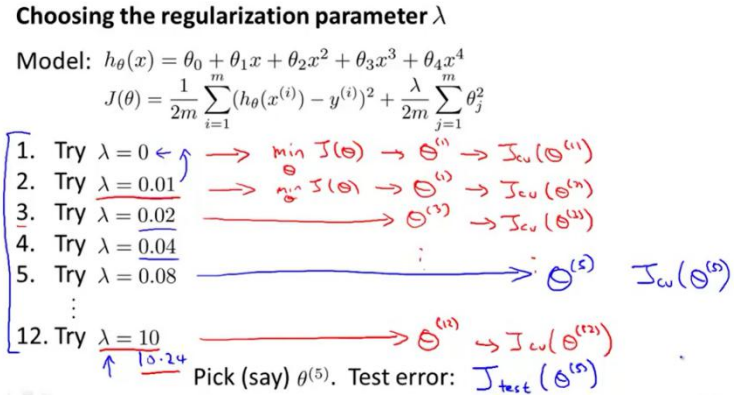
只要 cv error 或$J_{cv}(\theta)$很高的情况下，说明模型存在 Bias or Variance

1 Bias：**模型不理会数据,无论数据怎么增加，**变动 variance 很小，模型随着数据的增加波动很小 underfit

2 Variance： **Variance 大，随着数据增加模型波动很大，不稳定。数据增加 overfit 减轻，模型会变化。**
是因为：**数据的方差太大，我们的得到只是一小部分 range，需要更多的数据，才能获得总体的 range 近似，**因此 train data 的方差不代表真实 data，test set 的方差

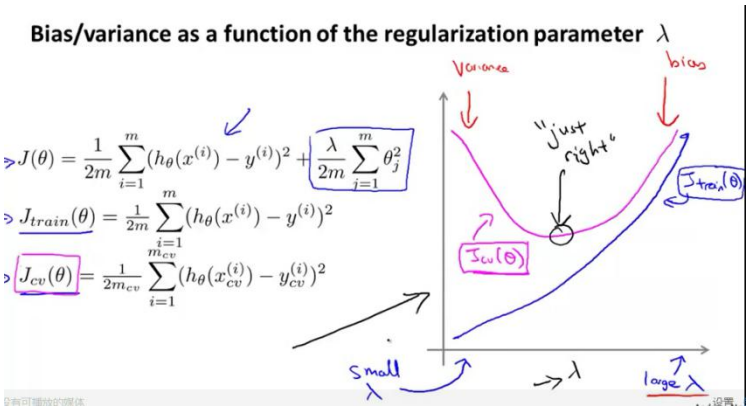
4 how to choose Regularization lambda(use validation)

how to choode λ value? valid and train error



选择一系列 lamda，训练不同的模型，得到不同的参数
然后计算 **cv 的 cost**(也就是 performance maxtrix)，用 cv 来选择模型，**选择 cost 最小的模型**

然后再用 test set $J_{test}(\theta)$，从而对模型 generalize



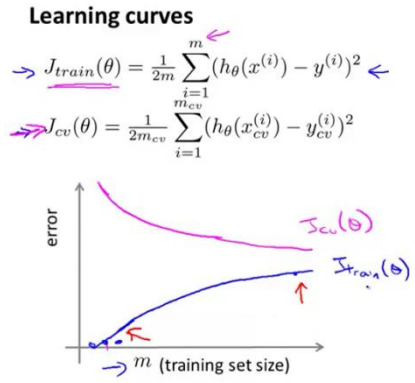
纵坐标是 train error(cost function)and cv error (cost function)与 λ 的关系：

当 λ 很大，模型 bias/underfitting，J train is high， J cv also high。

当 λ 很小，模型 **overfitting**，J train is low，and increase with λ increasing。and J cv also high，因为 overfitting。

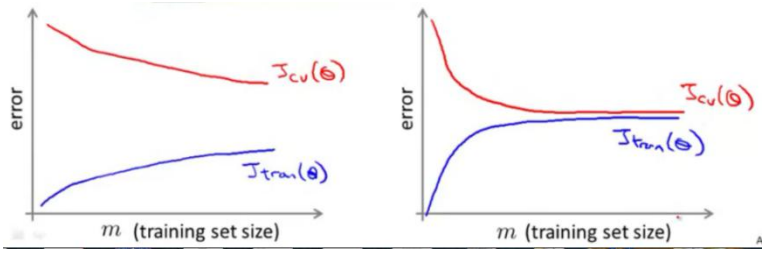
5 Learning Curves

Learn curve 用来诊断模型 overfit 还是 underfit.
to improve performance of our algorithm 难点
假设 train set 和 test set 已经分好了，train set have m examples，we **gradually add the number of our train set**
The #test set fixed.
The #train set to compute the train error **increase gradually**



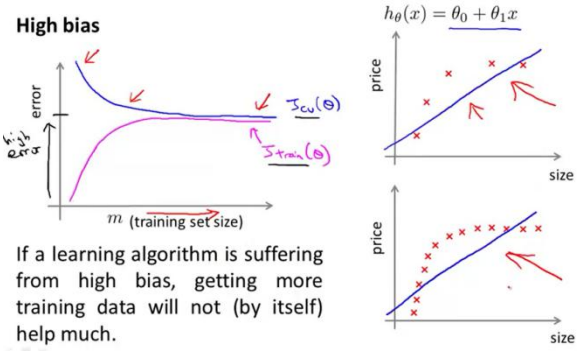
对于 J train
1 当 m 很小时，曲线都能很好的拟合点，J train 很小，但因为模型是 underfit ，J cv 却很大
2 随着 m 增大，**不能完全拟合所有点**，产生 error，J train 逐渐增大，但趋于稳定。而模型越来越好的拟合数据，J cv 减少
最后都趋于稳定。

根据 learning curve 诊断： bias，variance 共同点 vc error 都很大!! 模型表现都很差!!



左边 high-variance 过拟合了(train error 小，vc error 大)，缺少数据
右边 high-bias underfitting，缺少 feature

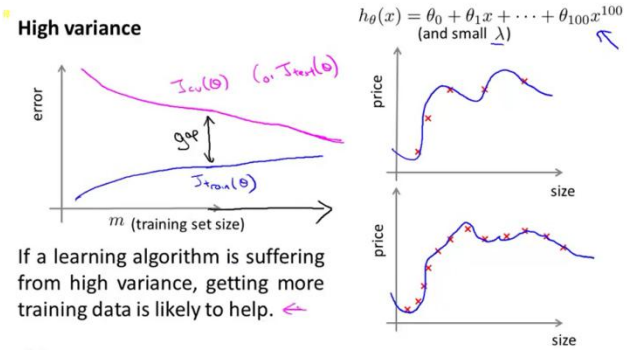
1 high bias：



vc error 和 Train error 都很高！

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

2 high variance：



vc error 很高！
train eror 低(因为 overfit)，会逐渐升高(overfit 逐渐减轻)
2 lines have a big gap
getting more training data may get 2 lines converage!!!

6 ML System Design(最佳实践)快速迭代法

Do a quick and dirt the first learning algorithm!!!Once we get initial implementation ,is a powerful tools to decide where to spent your time next:

1 look errors makes
2 error analysis
3 see the performance

numerical martix to test different ida to improve!!

7 classifiction 核心问题：threshold 确定+模型评估 metrics(Skewed Data)

logistics regression 为我们做分类提供了基础!! In order to map a logistic regression value to a binary category, you must define a 合理 classification threshold

Thresholds are problem-dependent, and are therefore values that you must tune.

To "Tuning" a threshold, we need metrics，来 evalurate model

we need metrics you can use to evaluate a classification model's predictions, as well as the impact of changing the classification threshold on these predictions.

Confusion matrix

true positive true negative

false positive false negative

都是针对模型的预测结果来定义

后面单词的代表模型预测 label!

前面代表模型预测对还是错，可以推出真实 label!

迅速得到

诀窍：当 threshold 提高,更少的预测 positive,会导致预测的 negative 增加，TN FN 都会增加!

预测 positive 的增多，TP,NP 都增加!

Accuracy 准确率

$$\text{Accuracy} = \frac{\text{Number of correct predictions}}{\text{Total number of predictions}}$$

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

但对于 class-imbalanced(Skewed Data)data set，accuracy 没有意义了

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} = \frac{1 + 90}{1 + 90 + 1 + 8} = 0.91$$

癌症诊断准确率 0.91

100 个人 91 个好，9 个癌症

一个模型如果全部预测好，准确率也是 0.91!! 但这样不合理，**因此不能仅仅看准确率了**

需要更细致的 assess 模型结果：需要更复杂的 metrics：

准确率是对所有类，精确率和召回率精确到**一类**

我们设定 y=1(正样本)是 rare class，这个是计算 precision 和 recall 的前提条件，**precision 和 recall 都是是针对 y=1 正样本这类计算的：**

1 Precision:所有预测为 P 中，正确的比例(范围限制在预测为 p 的准确率！)

从模型的角度(模型预测的 precision)

$$\text{Precision} = \frac{TP}{TP + FP}$$

2 Recall: 真实为 P，中被召回的比例!!

从正样本的角度(正样本的 recall)

$$\text{Recall} = \frac{TP}{TP + FN}$$

Trading Off Precision and Recall and F1 score

两者矛盾的关系：

Precision：保守预测（提高 precision，要谨慎预测，threshold 要大）

Recall：乐观预测（提高 recall，就要多预测 1，threshold 小）

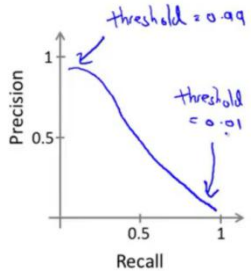
从而根据实际问题需求，可以用来确定我们算法的 threshold

如果两个都高！说明算法好!!!

上面癌症问题，recall 很低 Precision/Recall 很好的避免了 **skewed class** 情况

1 如果我们想减少**误判为癌症的概率**(增加 precision)，也就是 **we predict y=1, when very confident**，就要增加 threshold，maybe 0.7 .---->**higer precision**

2 如果想尽量多的将 y=1 的诊断出来，就要减少 threshold
通过改变 threshold，---->higer recall



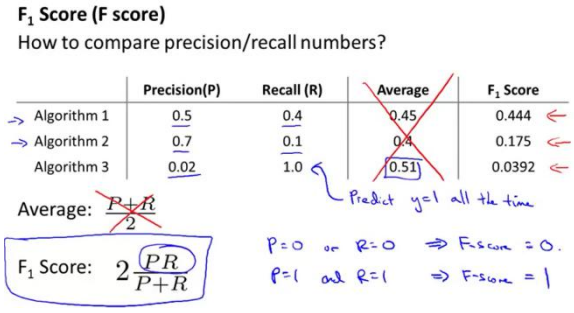
我们可以画出 threshold 和 precision 和 recall 的关系图

How to automatically pick the threshold? 利用 f1 score

we have 2 numerical matrics, difficult to compare with 2 algorithm。**we want a single numerical matrics**

平均值不行：

例如 predict y=1 all the time，recall very high，precision very low，but **average also high**，it’s not a good Algorithm!!



F1 score:**when P or R very low ,the prodcut also low**,when F1 score high, P and R both high!!!

当 P,R 全为 0，F1 Score=0

P,R 全为 1，F1 Score=1
F1 Score 取值在[0,1]

Choose Threshold by using vc data and F1 score!!

1 we try **different range of threshold**,ues **VC data to evaluate**,**to calculate different F1 Score**
2 pick the **highest F1 score thresholds!!**

ROC and AUC（不同 threshold 的表现）question

ROC curve(receiver operating characteristic curve)
is a graph showing the **performance of a classification** model **at all classification thresholds**

This curve plots two **parameters:横纵坐标值**

rate 是啥？？question

- True Positive Rate 就是 TP(只不过是一直变化)
- False Positive Rate 就是 FP

An ROC curve **plots TPR vs. FPR at different classification thresholds.** Lowering the classification threshold(预测更多 P)，will classifies more items as **positive,** thus increasing both **False Positives and True Positives.**

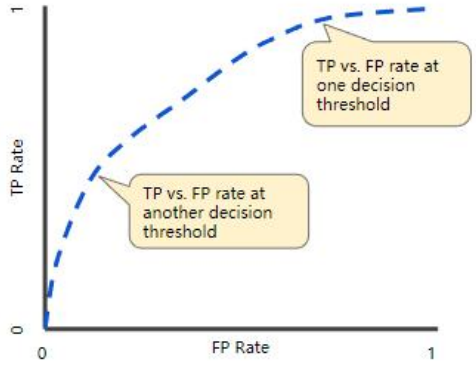


Figure 4. TP vs. FP rate at different classification thresholds.

To compute the **points** in an ROC curve, we could evaluate a logistic regression model **many times with different classification thresholds,** but this would be **inefficient.**

Fortunately, there's an efficient, sorting-based algorithm that can provide this information for us, called AUC.

AUC: Area Under the ROC Curve(from (0,0) to (1,1).)

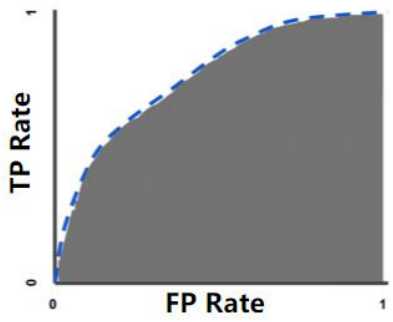


Figure 5. AUC (Area under the ROC Curve).

AUC provides an aggregate 综合的 measure of performance across all possible classification thresholds.

AUC is as the probability , that if we random choose one positive and one negative examples , the probability of model can correctly ranks (or give the higher score to)a random positive example more highly than a random negative example. to get the pairwise order correct

AUC is desirable for the following two reasons:

- AUC is scale-invariant. 跟数据的 scale 无关, It measures how well predictions are ranked, rather than their absolute values.
- AUC is classification-threshold-invariant. 综合指标 It measures the quality of the model's predictions irrespective of 不考虑 what classification threshold is chosen.

Classification-threshold invariance is not always desirable.

In cases where there are wide disparities in the cost of 代价不同 false negatives vs. false positives,it may be critical to minimize one type of classification error.

When doing email spam detection, you likely want to minimizing false positives ,根据诀窍:其实就是减少 positive 的预测,增加 negative 的预测,导致 TN FN 增大,再去想真实含义: FN 增大就是 真实是 P(垃圾), 预测为 N(普通)AUC isn't a useful metric for this type of optimization.
coding:

#先用模型 evaluate validate
evaluation_metrics =
linear_classifier.evaluate(input_fn=predict_validation_input_fn)
#predict_validation_input_fn 包含了 X 和真实的 label
#打印 auc 和 accuracy
print evaluation_metrics['auc']
print evaluation_metrics['accuracy']

画图：
#计算出

```
validation_probabilities = linear_classifier.predict(input_fn=predict_validation_input_fn)
# Get just the probabilities for the positive class
validation_probabilities = np.array([item['probabilities'][1] for item in validation_probabilities])

false_positive_rate, true_positive_rate, thresholds = metrics.roc_curve(
    validation_targets, validation_probabilities)
plt.plot(false_positive_rate, true_positive_rate, label="our model")
plt.plot([0, 1], [0, 1], label="random classifier")
_ = plt.legend(loc=2)
```

sklearn 的方法

8 Prediction Bias question

Logistic regression **predictions should be unbiased**. That is:

"**average** of predictions" should ≈ "**average** of observations"

Prediction bias is a quantity that measures **how far apart those two averages are**. That is:

prediction bias=average of predictions−average of labels in data set

A significant nonzero prediction bias tells you there is a **bug** somewhere in your **model**, as it indicates that the model **is wrong about how frequently positive labels occur.预测正负的频率有问题：**

For example, let's say 1% of all emails are spam. For a given email, a good spam model should **predict on average that emails are 1% likely** to be spam.)If instead, the model's average **prediction is 20% likelihood** of being spam, we can conclude that it **exhibits prediction bias.**

Possible **root causes** of prediction bias are:

- Incomplete feature set
- Noisy data set

- Buggy pipeline
- Biased training sample
- Overly strong regularization

A **good model** will usually have **near-zero bias**. That said, a low prediction bias does not prove that your model is good. A really terrible model could have a **zero prediction bias**.

9 Bucketing and Prediction Bias 不太懂 question

Logistic regression predicts a value 聽 *between* 聽 0 and 1. However, all labeled examples are either exactly 0 (meaning, for example, "not spam") or exactly 1 (meaning, for example, "spam"). Therefore, when examining prediction bias, you cannot accurately determine the prediction bias based on only one example; you must examine the prediction bias on a "bucket" of examples. That is, prediction bias for logistic regression only makes sense when grouping enough examples together to be able to compare a predicted value (for example, 0.392) to observed values (for example, 0.394).

You can form buckets in the following ways:

- Linearly breaking up the target predictions.
- Forming quantiles.

Consider the following calibration plot from a particular model. Each dot represents a bucket of 1,000 values. The axes have the following meanings:

- The x-axis represents the average of values the model predicted for that bucket.
- The y-axis represents the actual average of values in the data set for that bucket.

Both axes are logarithmic scales.

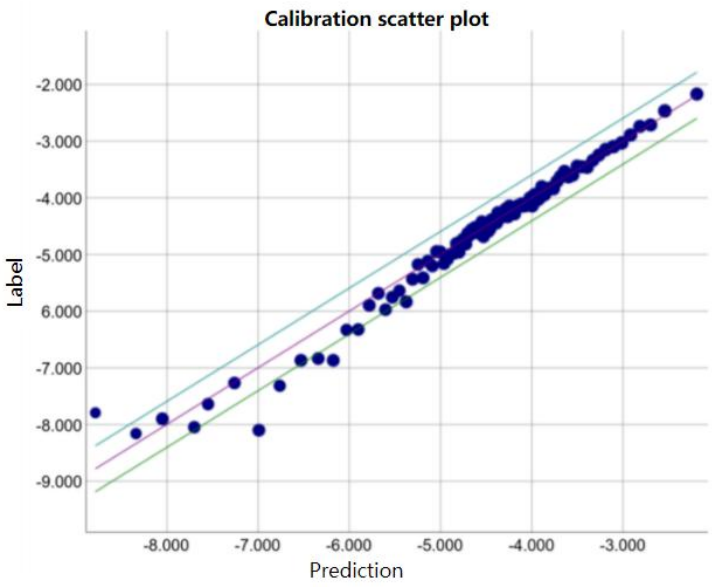


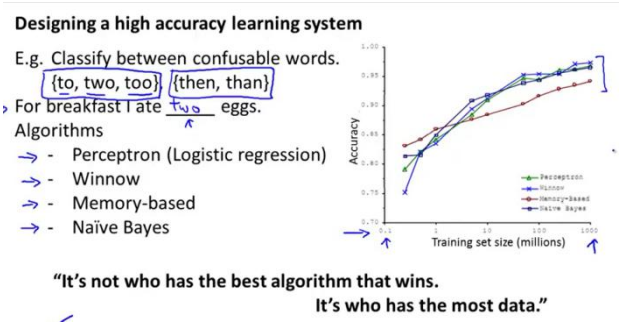
Figure 8. Prediction bias curve (logarithmic scales)

Why are the predictions so poor for only *part* of the model? Here are a few possibilities:

- The training set doesn't adequately represent certain subsets of the data space.
- Some subsets of the data set are noisier than others.
- The model is overly regularized. (Consider reducing the value of lambda.)

10 When to use Large Data Sets

how much data need to train on ?



但也不是所有情况下，data 越多，performance 越好！rationale 原理

Large data rationale

Assume feature $x \in \mathbb{R}^{n+1}$ has sufficient information to predict y accurately.

Example: For breakfast I ate two eggs.

Counterexample: Predict housing price from only size (feet^2) and no other features.

Useful test: Given the input x, can a human expert confidently predict y?

如何判断我么的 feature 足够的多？让人去测试下！看能否容易推断出来假如有足够的 feature，就会有很多的 parameter 是 low bias algorithm(feature 多的算法)随着 data 增加，因为 low bias，拟合 data 很好，J train 会很小

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.

$\rightarrow J_{train}(\theta)$ will be small.

Use a very large training set (unlikely to overfit)

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

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

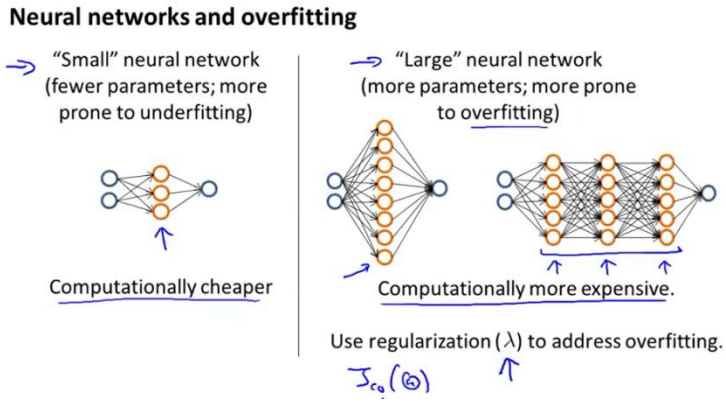
又因为，用大数据！所以 unlikely to overfit!!! 因此，J train 和 J test 比较接近，因此 J test 也很小!!! high performance algorithm, no high bias no high variance。we have sufficient feature to avoiding bias, we get large data to avoid variance use large data to train the algoritm with lots of features。1The key test is can humind look the feature, have confidence to predict。2get lage data 两者结合!!!

11 Deciding What to Do with model

todo:

- Get more training examples → fixes high variance
- Try smaller sets of features → fixes high variance
- Try getting additional features → fixes high bias
- Try adding polynomial features $(x_1^2, x_2^2, x_1x_2, \text{etc})$ → fixes high bias.
- Try decreasing λ → fixes high bias
- Try increasing λ → fixes high variance

对于 Nueral network



large network tend to overfitting, we can use regularizatong to avoid!!

Using a single hidden layer is a good starting default. We can train your neural network on 不同 number of hidden layers ,and use cross validation set to select the one that performs best.

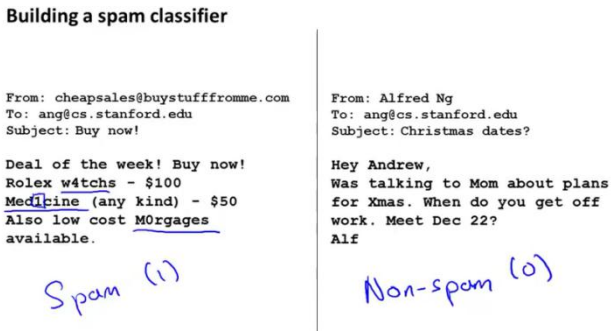
1 If the training has not converged, try running it for longer.
2 If the training error decreases too slowly, increasing the learning rate may help it decrease faster.
But sometimes the exact opposite may happen if the learning rate is too high.

3 If the training error varies wildly, try decreasing the learning rate.
Lower learning rate plus larger number of steps or larger batch size is often a good combination.

Very small batch sizes can also cause instability. First try larger values like 100 or 1000, and decrease until you see degradation.

12 Build a spam classifier(待看)

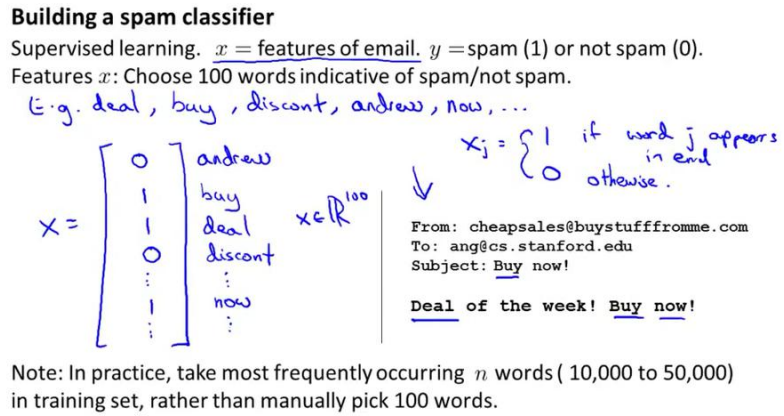
Prioritizing What to Work On



1: 确定 features and labels

feature 敏感词，每个词代表一个 feature x_i

x 则是个向量。



我们在 train set 邮件中寻找到最高频的词作为 feature

2 Build a classifier

So how could you spend your time to improve the accuracy of this classifier?
如何提高模型准确性？

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

4 更精细

- Develop sophisticated features for message body, e.g. should "discount" and "discounts" be treated as the same word? How about "deal" and "Dealer"? Features about punctuation?

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

Error Analysis（test on cv set ,rather than test set）
manully examming the mistaked examples,to guide us a more fruitful way to pursue.
recommand firstly quick and dirty implement of algorithm,then identify what is the most diffcult examples to classify,?focus effort on it

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.

识别错误的邮件中，有哪些系统性 pattern，give us inspiration to design new features

Error Analysis example:

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

cv=500，有 100 个错误分类
手动查看错误 example，分类，看哪些类出错频率大，从该类中提取新的 feature。

numerical evaluation
四个意思相同的词，词根一样

Should discount/discounts/discounted/discounting be treated as the same word?
Can use "stemming" software (E.g. "Porter stemmer")

词根提取 stemming software，提取前几个单词(porter stemmer 软件)，treat as same word which first servral alphabet are same。

也会出问题，如：universe/university. not same thing
我们每次改进自己算法时，如加上 stemmer 软件，需要 manully exam to see better or worse? very hard!!!
可以用一些数值指标去衡量和对比算法 performance

Need numerical evaluation (e.g., cross validation error) of algorithm's performance with and without stemming.
Without stemming: 5% error With stemming: 3% error

来测试我们的改进建议是否有效.

词汇；rule out 排除可能性 address 处理 be prone to(tend to) strategize doing 制订战略 touch on 触及 obscure 遮掩 掩盖 模糊的 fixate 固定下

里,in the absense of 缺少 as a reminder 回顾一下 when in doubt 当存在疑问时 保守的 conservativeincorporate st into 融入 合并到 caution 告诫 blindly 盲目的 confusable 容易混淆的 rationale 基本原理

十.SVM（解决高维非线性问题）

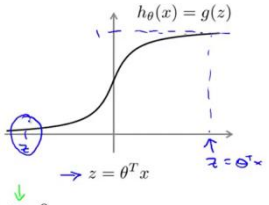
SVM 的处理方法是选择一个核函数 κ()，通过将**数据映射到高维空间，来解决在原始空间中线性不可分的**

1 Large Margin Classification represent

其实也是要从最基本的二分类 lr 模型开始

Alternative view of logistic regression

$$h_\theta(x) = \frac{1}{1+e^{-\theta^T x}}$$



If $y = 1$, we want $h_\theta(x) \approx 1$, $\theta^T x \gg 0$
If $y = 0$, we want $h_\theta(x) \approx 0$, $\theta^T x \ll 0$

要求更高！判断更严格，增加一个距离边界，也就是θ^Tx 要么很大，要么很小，不会出现中间的值(margin)
我们想找到这种 theta！
更加 confident to our result

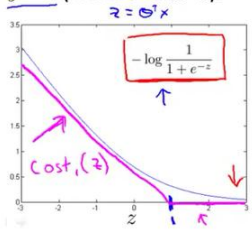
先从 Lr 模型开始：
如果想减少错判率，就要使得线性组合部分远大于或小于 0
通过改变 cost function，来改变对 theta 的要求：

Alternative view of logistic regression

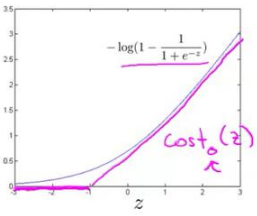
Cost of example: $-(y \log h_\theta(x) + (1-y)\log(1-h_\theta(x)))$

$$= -y \log \frac{1}{1+e^{-\theta^T x}} - (1-y)\log\left(1 - \frac{1}{1+e^{-\theta^T x}}\right)$$

If $y = 1$ (want $\theta^T x \gg 0$):



If $y = 0$ (want $\theta^T x \ll 0$):

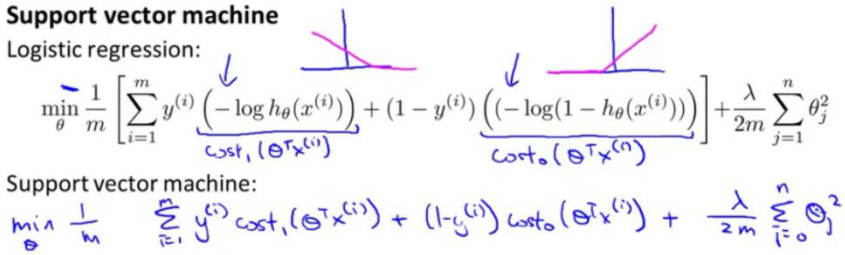


纵坐标 cost fun，横坐标 z
修改前：当 y=1，就算 z 值很大，cost 不为 0，也会惩罚。

修改后：我们对 cost fun 做修改。**当 z 大于某个值时，cost 始终为 0，不做惩罚，鼓励 z 是大的值！**

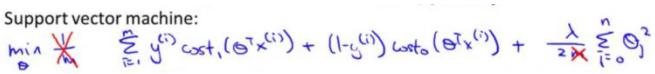
引入新的 cost： y=1 时 $\mathrm{Cost}_1(z)$，y=0 同样记为 $\mathrm{Cost}_0(z)$

用新的 cost fun 去代替！就是 SVM

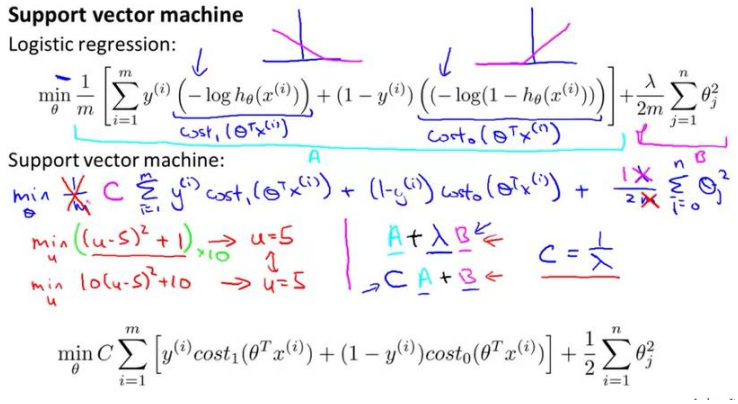


by convention:

1. 去掉 1/m，因为优化的话影响不大



2. 加一个 C 因子，类似于 lambda，也是起到 regularization 作用



对于 SVM，我们用另一个 parameter C replace λ

C 和 lambda 相反!!

$$C = \frac{1}{\lambda}$$

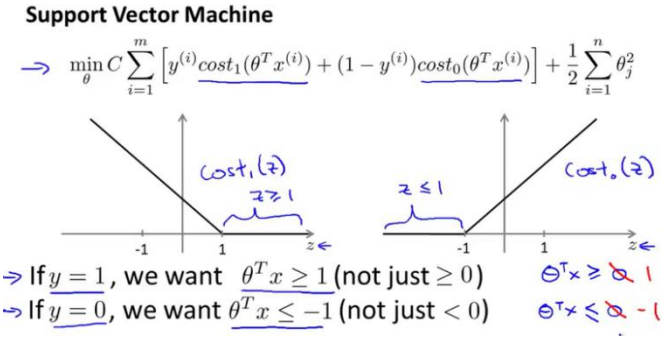
SVM 的 cost fun：

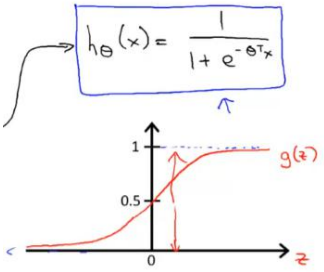
$$\min_\theta C\sum_{i=1}^{m}\left[y^{(i)} cost_1(\theta^T x^{(i)}) + (1-y^{(i)}) cost_0(\theta^T x^{(i)})\right] + \frac{1}{2}\sum_{i=1}^{n}\theta_j^2$$

C 越大，lamda 越小，**参数值越大**

1.1 convert to optimal proble

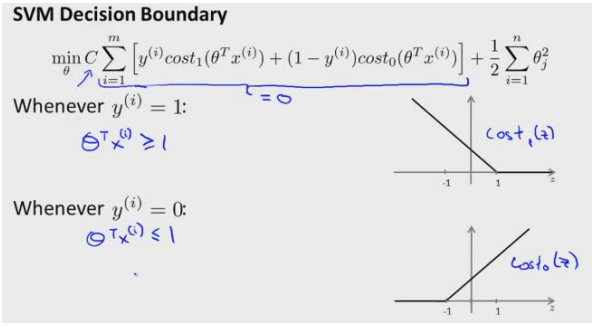
假如：我们设定一个 margin 为 1





但 SVM 对 $\theta^T x$ 要求更高

如果 C 非常大!! 下面蓝色框式子 A 应该趋向于 0，才能 min cost



左：当 y=1，蓝色框式子 A=0 的条件是 $\theta^T x$>=1

右：当 y=0，蓝色框式子 A=0 的条件是 $\theta^T x$<=-1

所以当满足 $\theta^T x$>=1， $\theta^T x$<=-1 时，A 一定为 0，**我们只需 min regularization 项**

SVM Decision Boundary

$$\min_\theta C\sum_{i=1}^{m}\left[y^{(i)}cost_1(\theta^T x^{(i)})+(1-y^{(i)})cost_0(\theta^T x^{(i)})\right]+\frac{1}{2}\sum_{i=1}^{n}\theta_j^2$$

Whenever $y^{(i)}=1$:

$\Theta^T x^{(i)} \geq 1$

Whenever $y^{(i)}=0$:

$\Theta^T x^{(i)} \leq -1$

$\min \frac{1}{2}\sum_{i=1}^{n}\Theta_j^2$

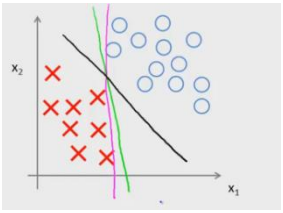
$s.t.\ \Theta^T x^{(i)} \geq 1 \quad if\ y^{(i)}=1$

$\Theta^T x^{(i)} \leq -1 \quad if\ y^{(i)}=0.$

minimize cost function 就转换成右下角优化问题

通过上面求解我们可以得到 Large Margin Boundary(more rubost)separate the data as large margin as impossible

对于应下面黑色的线：



1.2 直观理解 z 和 margin

margin 是 data 和 boundary 的距离！

距离之间没有数据点，也就是边界上的数据带入 $\theta^T x=1$(就是边界值!!)

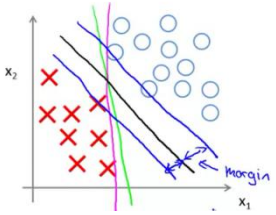
z 远大于 0，z 远离边界 0，其实就是找到离数据最远的边界!!! 这样子错判率最低!!

margin is the distance from the decision boundary

max the margin:

boundary 离正负数据的边界距离一样(正负边界)

SVM Decision Boundary: Linearly separable case

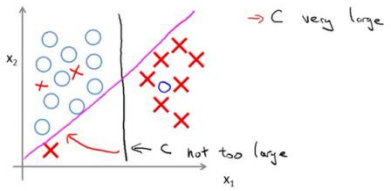


如果 C 很大，容易过拟合，更受 data noise 影响。得到紫色线。但我们想要黑色的线！

即使 data 不能线性可分，也能得到最优边界

如果 C 不是很大，就得到黑色线

even the data not linear separable!! 也能得到最优的边界



1.3 Mathematics Behind Large Margin Classification

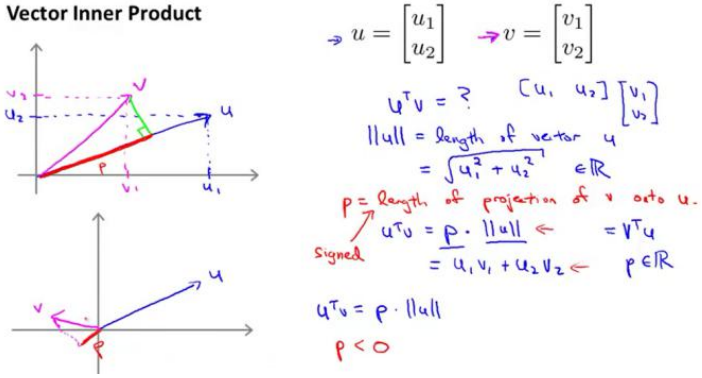
假设正确分类是 1，错误分类是-1

向量数量积和投影的关系：

(内积)数量积=a 在 b 上**投影***b 长度

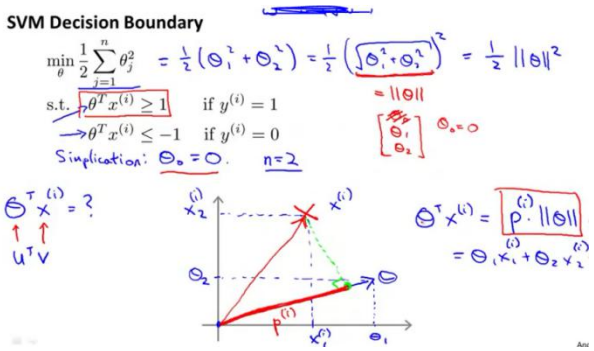
向量的投影 P 是长度，有正负!!

norm of u ||u||**范数:向量长度**



下面是 SVM 的 cost fun 及条件

假设我们只有两个**参数 theta**，cost fun 可以写成 **theta 距离形式!!**



条件$\theta^T x^{(i)}$是向量积，可以**写成 x 向量到 theta 向量的投影形式**$p^{(i)}\|\theta\|$

p 就是向量 x 到 theta 的投影。

因此可以写成如下形式：

SVM Decision Boundary

$$\min_\theta \frac{1}{2}\sum_{j=1}^{n}\theta_j^2 = \frac{1}{2}\|\theta\|^2$$

$$\text{s.t.}\quad p^{(i)}\cdot\|\theta\| \geq 1 \quad \text{if } y^{(i)}=1$$

$$p^{(i)}\cdot\|\theta\| \leq -1 \quad \text{if } y^{(i)}=1$$

where $p^{(i)}$ is the projection of $x^{(i)}$ onto the vector θ.

Simplification: $\theta_0 = 0$

根据前面的 logistics，我们知道我们的边界是 z(线性组合) $\theta^T x$

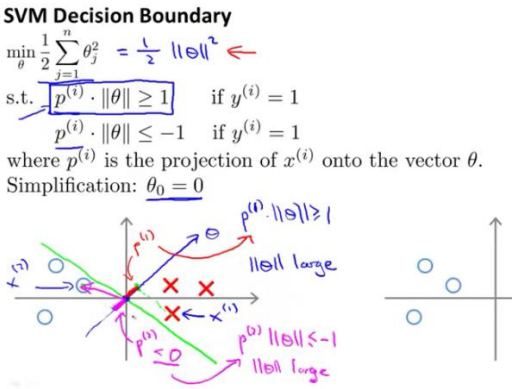
decision boundary 是直线： $\theta^T x+\theta_0=0$ (也就是边界线上的 data 带入为 0)，

其实就是： θ 确定下来以后，各个 feature x 和 θ 线性组合产生的超平面(x 是未知数，theta 是常数)!!! 也就是 $\theta^T x+\theta_0=0$， θ_0=0 时过原点的线

假设超平面上 2 个点(两个 feature 向量) x_1, x_2，则 $\theta^T x_1$ 减 $\theta^T x_2$ =0

$\theta^T(x_1-x_2)=0$ x_1-x_2 是超平面上的**线段向量**，则 θ 向量一定垂直于 x_1-x_2，垂直于平面

1.4 min cost function 直观意义

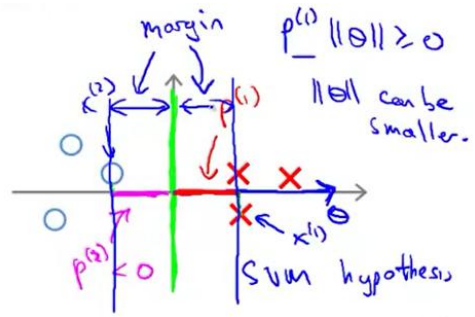


假设**绿线是 boundary**，

点(也就是到原点的向量)到 theta 向量(boudary 法线)的投影 **p 很小**，

如果想要 $p\|\theta\| \geq 1$，则需要 $\|\theta\|$ 很大。但我们的目标函数是为了 min $\|\theta\|$，所以不是好的 boundary

因此为了 minimize cost fun 又要满足 st 条件，需要将 **p 投影变的尽量大**，也就是 max margin!!!! **p 就是 margin!!!** (自己画一下图就可以)



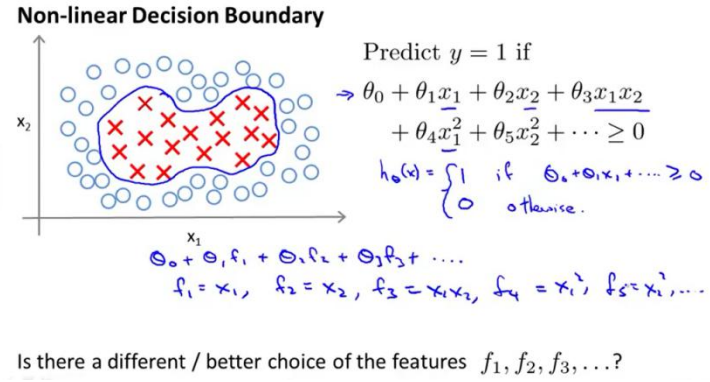
SVM will choose this hypothesis
此时 p(margin)最大.
p is the data distance from the decision boundary, is also the margin, which we want to maximize

2 Kernels(非线性分类)

前面是 linear kernel，其实也就是不用 kernel

2.1 非线性问题

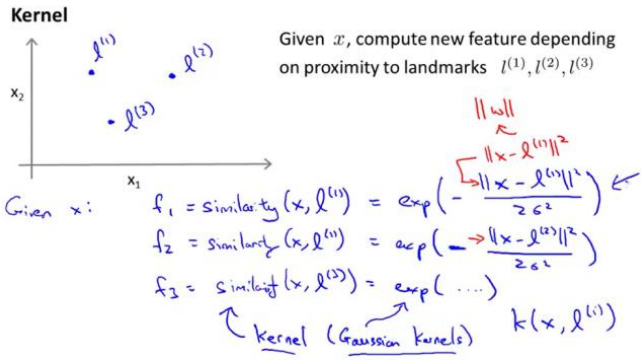
对于非线性可分数据：如何找到最大边界？？



假设我们拟合一个多项式边界，大于 0 预测 1
我们需要 synthesize 新的 feature(feature cross) :f1 f2 f3...
但因为多项式，计算量特别大尤其对计算机视觉图片处理时，有没其他 feature 替代(不需要这么复杂的计算)？

2.2 相似度代替原 feature(转化为高维)

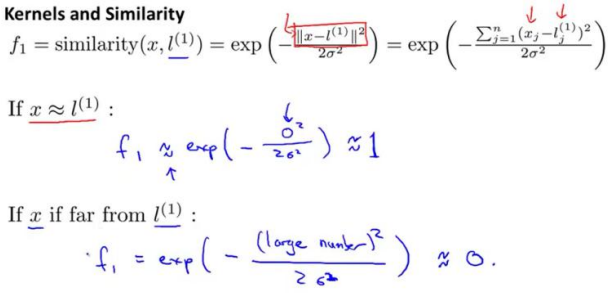
我们来计算合成新的 feature ：



landmark 是原来 x 维度里任意几个点
Kernel 是 similarity 相似度 function，是任意一点 l (landmark)到给定一点 x 的相似度相似度：距离越近相似度越高，值越大
相似度和距离一样，相似度就是距离取反
similarity function 就是 kernel function!!

因此原来的任意一个点 x，都可以计算多个 landmark 相似度 f
f 值作为新的 feature 值

不同的 kernel 相似度计算表示：
我们用的 Gaussian kernels:

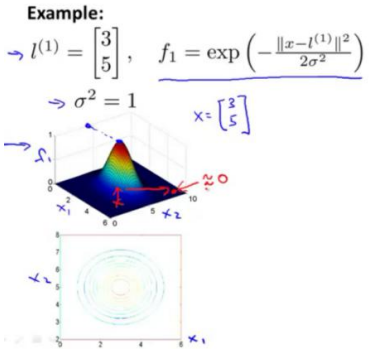


$\|x-\gamma\|$是 euclidean 两个点的欧氏距离

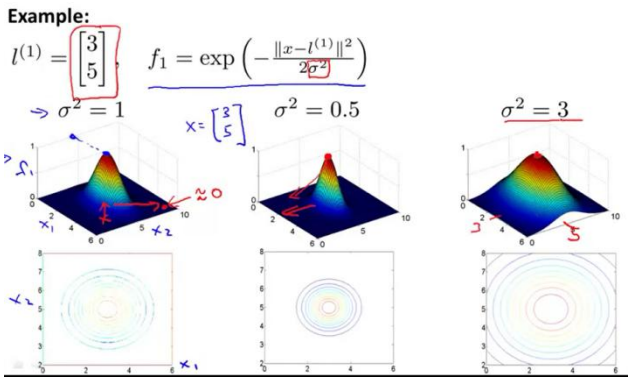
距离越近，相似度值越大(越接近 1)，代表两个点距离越近

为什么忽略了 x0？？？是截距项？X0 恒等于 1
因此在选定三个 lanmark 后，随便给一个 x 就可以计算三个新 feature ，二维 feature 变三维 feature(低维度变高维度)
想变换成几维就选几个 landmark

直观：

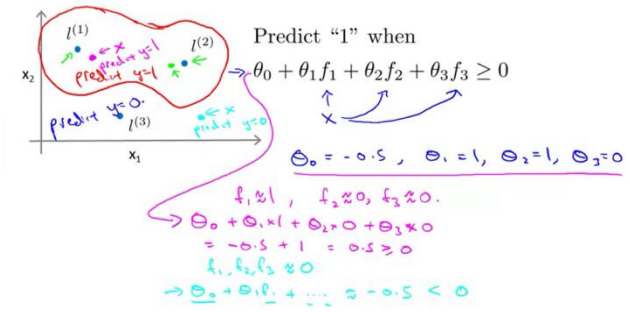


suppose we have 2 features: x1, x2
一个固定的 l1:(3,5)(相当于是多元正态分布的两个均值)
多元正态分布，均值是向量，因为中心需要两个坐标 x1x2 确定!! 多元正态分布其实就是多元函数，多个 x，一个 y!!
图中纵坐标是 f 值,下面是 x1,x2
每个 x1 x2 值，对应一个 f 相似度
可以看出当 x1，x2 是 3,5 时，f 最高，和 l 相似度最大!!
调解下参数σ：影响新 feature f 的值



alpha 越大，从 1 变 0 越慢!!

用新的 feature f 替代：



假设我们找到了 theta：-0.5,1,1,0
对于所有 train data 计算出 f ，带入上面式子
左上角红点离 l1 近，f1 几乎为 1，离 l2，l3 远，f2，f3 几乎为 0
因此带入式子=0.5>0,因此预测为 1
同理右下蓝点，预测为 0

所有 data 最终可以画出 boundary!! 是由 3 方面决定!!

是由 1 参数 θ 或以及 2 landmark 和 3kernel 决定!!

增加了 kernel 函数和 lanmark,可以 learn 更加 complex decision boundary。代替了上面的多项式回归!!!!!

2.3 landmark location(how to choose)

我们选 train data 的每个 example，都作为 landmark(same point)

$x^{(i)} = l^{(i)}$ 得到 m 个 l

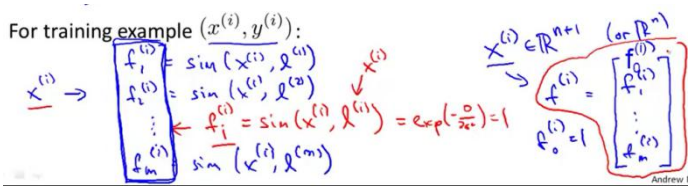
上面显示的是 14 个 2 维 feature 的 data examples，选择 14 个 landmark，每个 example 计算 14 个 f，得到 14 个由 2 维变成了 14 维

SVM with Kernels

Given $(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \ldots, (x^{(m)}, y^{(m)})$,

choose $l^{(1)} = x^{(1)}, l^{(2)} = x^{(2)}, \ldots, l^{(m)} = x^{(m)}$.

given training examples，选择每个点作为 landmark



对于 train set,任意一个数据 $x^{(i)}$，因为有 m 个 l，会有 m 个新 f，原来的 n 维 x 变成新的 m 维 f

新 feature，也要考虑作为截距项 $x^{(0)}$ f0，新 feature m+1 维度

3 SVM with kernels hypothesis

前面的 SVM 加上了 kernel 得到了下面的 model:

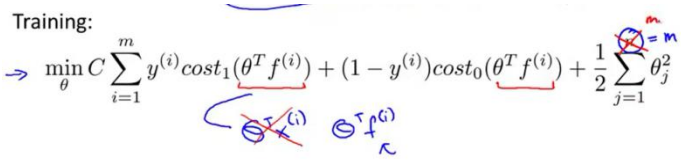
SVM with Kernels

Hypothesis: Given x, compute features $f \in \mathbb{R}^{m+1}$ $\quad \theta \in \mathbb{R}^{m+1}$

→ Predict "y=1" if $\theta^T f \geq 0$ $\quad \theta_0 f_0 + \theta_1 f_1 + \cdots + \theta_m f_m$

相应的θ也变成了 m+1 维度：

SVM cost fun 改变：



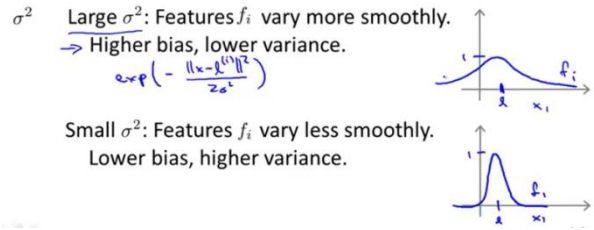
用新的 feature f 新的参数 theta 代替，后面改成 m 个参数

因为 SVM 对**数据量敏感，因为要计算 m 个 feature，**当 m 很大计算量也很大

实际中 regularization 项会稍微变化：中间会加个矩阵，**可以让 svm 适应更大的数据**

tune 参数：
1 对 model 的影响
C 越大 越容易过拟合

2 对 feature 的影响



sigma 越大,f 变化越平稳，不容易过度拟合!!! 容易得到很平稳的 hypothesis，容易 bias，例如得到直线模型拟合曲线的点
sigma 越小，**f 变化越大，variance 越大，得到很复杂的 hypothesis**
则我们得到的 hypothesis **容易 high variacne，容易过度的拟合！**
sigma 越小,数据分布越集中，数据值变化越大

4 SVM in Practice

Using An SVM

Use SVM software package (e.g. liblinear, libsvm, ...) to solve for parameters θ.

Need to specify:

→ Choice of parameter C.

Choice of kernel (similarity function):

我们需要选定 C，kernel

E.g. No kernel ("linear kernel")

Predict "y = 1" if $\theta^T x \geq 0$

$\theta_0 + \theta_1 x_1 + \cdots + \theta_n x_n \geq 0$

→ n large, m small

$x \in \mathbb{R}^{n+1}$

当 feature 数量 n 很多，数据量 m 很少，最好用 linear kernel(不用 kernel)，因为用别的 kernel 容易过拟合，产生复杂模型，但数据量太少

Gaussian kernel:

$f_i = \exp\left(-\frac{||x - l^{(i)}||^2}{2\sigma^2}\right)$, where $l^{(i)} = x^{(i)}$.

Need to choose σ^2.

$x \in \mathbb{R}^n$, n small and/or m large

当 n 很少，m 很多时，并且非线性可分时，可以用 kernel，**来产生 no-linear boundary!!**

如图 n=2，m 很多

自定义 kernel function

Kernel (similarity) functions:

```
function f = kernel(x1,x2)
    f = exp( -||x1 - x2||^2 / (2σ^2) )
return
```

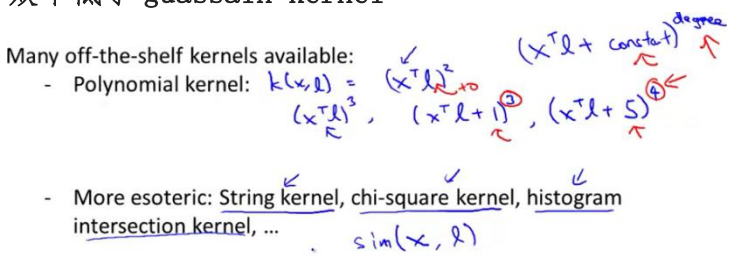
$x \rightarrow \begin{bmatrix} f_1 \\ f_2 \\ \vdots \\ f_m \end{bmatrix}$

我们可以自己定义 **similarity 函数**，其实就是两个点参数，返回距离的函数

不是所有我们定义的 kernel 都有效，必须要遵循技术标准：mercer' s theorem，保证快速的得到参数 θ

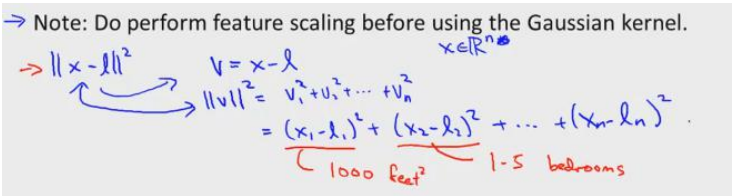
还有些其他满足 theorem 的 kernel，**但用的很少：**

效率低于 guassain kernel



String kernel input 字符串类型

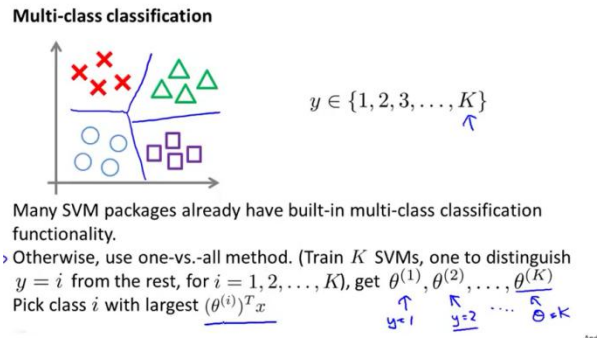
用之前标准化



在求新 feature 之前**必须要标准化**，不然：**x 到 l 的距离**仅仅由几个数值比较大的 feature 决定!!! 保证 feature 平等，每个 feature 平等的决定距离

multi-class classification(one vs all)

也是 one vs all 思路，train 多个 svm 进行分类



SVM 一般是提供了 muti class 分类实现了 one vs all
也可以自己实现：one vs all
每一类拟合一个 SVM，得到多个参数向量。

判断一个新的数据时，只需看最大的 $\theta^T x$

5 logistics and SVM(本质关系)

Logistic regression vs. SVMs

n =number of features ($x \in \mathbb{R}^{n+1}$), m =number of training examples

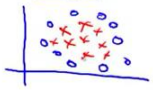
- If n is large (relative to m): (e.g. $n \geq m$, $n = 10{,}000$, $m = 10 \cdots 1000$)
- Use logistic regression, or SVM without a kernel ("linear kernel")

1 当 n 很多，m 很少用 logistics 或 SVM linear kernel，因为数据量少，没法拟合复杂的模型

2 m 不是很多，可以用 kernel

If n is small, m is intermediate: ($n = 1\text{-}1000$, $m = 10\text{-}10{,}000$)

→ Use SVM with Gaussian kernel



3 m 太大也不适合用 kernel

If n is small, m is large: ($n = 1\text{-}1000$, $m = 50{,}000+$)

Create/add more features, then use logistic regression or SVM without a kernel

logistics 和 svm(linear kernel)是非常相似的算法。

svm power 在于 ： 可以用不同 kernel，来学习非线性分类，计算量更小，更有效!!!

Neural network likely to work well for most of these settings, but may be slower to train.

十一. Embeddings(见练习还是不太懂实现，question)

1 Collaborative filtering 协同过滤 and Embedding

1 all users collaboratively help system get better movie rating for everyone

2 is the task of making predictions about the interests of a user based on interests of many other users data.

解决方案：determine which movies are similar to each other. We can achieve this goal by embedding the movies into a low-dimensional space

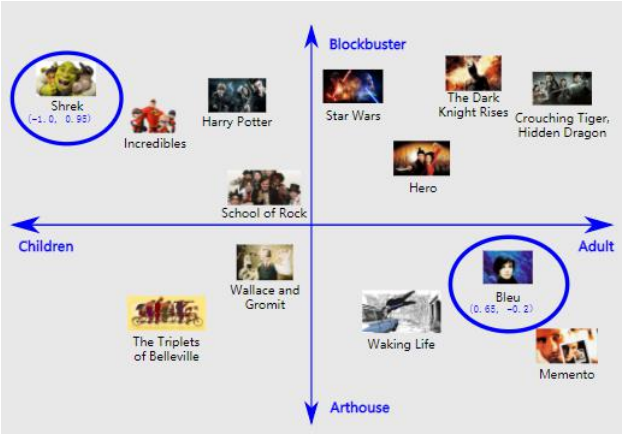
Mapped these movies into an embedding space

embadding 其实就是 product 的 feature!!!目的是可以**计算 product 的相似性，**从而进行推送!!

例如：

Movie recommendation. Suppose we have 1,000,000 users, and a list of the movies each user has watched, Our goal is to recommend movies to users.

我们为电影设定两个 feature（受众年龄，理想现实）



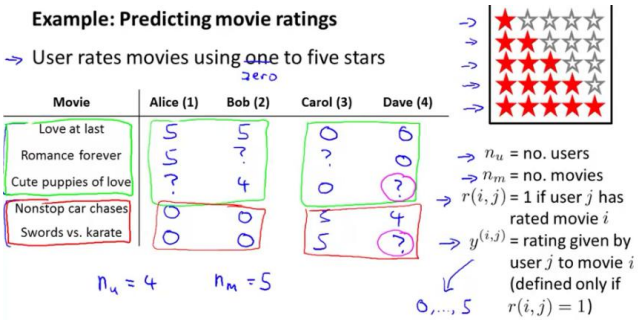
For example, in this space, "Shrek" maps to (-1.0, 0.95) and "Bleu" maps to (0.65, -0.2)，**用 embedding 来 represent movie**

这个 feature(embedding)需要我们自己去 learn

When learning embeddings, the individual dimensions **are not learned with names.** Sometimes, we will assign semantic meanings to the dimensions, and other times we cannot.

Each such dimension is called a latent dimension,不是从 data 中直接得来，二是我们学习推断出来的。

2 应用 Recommender System



我们需要预测所有 missing data，其实就是利用已有的数据来预测用户对新电影的评分！

可以看出规律，alice 和 bob 喜欢前三部 romantic

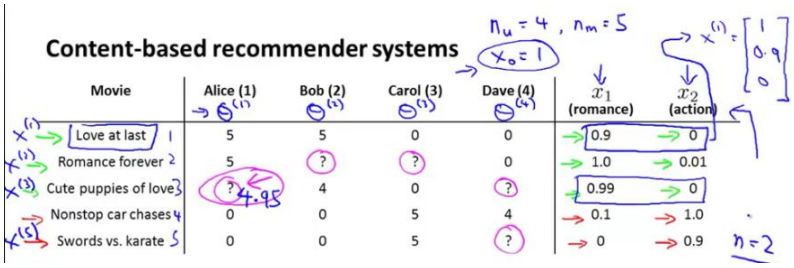
carol and dava 喜欢后两部，可以大概的预测

预测没有看过的用户是否喜欢这个产品

需要建立一个 learning algorithm

方法 1 Content Based Recommendations

feature 就是 embedding，只不过我们自己给出值



我们可以根据 movie 的特点提取两个特征，并且手工确定其值，这样每个 movie 有两个特征 共 nm 个 movie

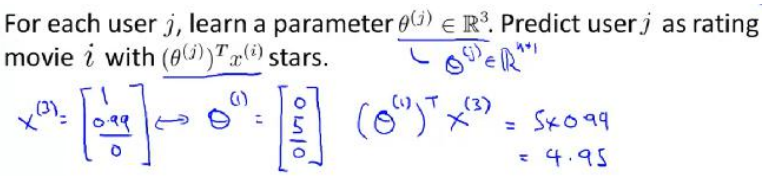
每个 movie 对应 2 个 feature: romance|action

the degree of kind of movie

加入一个 intercepter 值为 1，一共 3 个 feature

第一个电影 feature(1,0.9,0)

然后对 user 训练参数，使得参数乘以电影 feature 等于最终评分



1 先考虑：learn 一个人的参数，将一个人评论过得电影计算误差

Problem formulation

$r(i,j) = 1$ if user j has rated movie i (0 otherwise)

$y^{(i,j)}$ = rating by user j on movie i (if defined)

$\theta^{(j)}$ = parameter vector for user j

$x^{(i)}$ = feature vector for movie i

For user j, movie i, predicted rating: $(\theta^{(j)})^T(x^{(i)})$

$m^{(j)}$ = no. of movies rated by user j

To learn $\theta^{(j)}$:

$$\min_{\theta^{(j)}} \frac{1}{2}\sum_{i:r(i,j)=1}\left((\theta^{(j)})^T(x^{(i)}) - y^{(i,j)}\right)^2 + \frac{\lambda}{2}\sum_{k=1}^{n}(\theta_k^{(j)})^2$$

用户 j 评论过的所有电影： sum(i:r(i,j)=1)

m(j)是个常数，对于优化可以省略，得到如下：

To learn $\theta^{(j)}$ (parameter for user j):

$$\min_{\theta^{(j)}} \frac{1}{2}\sum_{i:r(i,j)=1}\left((\theta^{(j)})^T x^{(i)} - y^{(i,j)}\right)^2 + \frac{\lambda}{2}\sum_{k=1}^{n}(\theta_k^{(j)})^2$$

2 learn 所有人的参数,将所有评论过得电影计算误差,求全部误差最小,sum(所有 user 的 error)

To learn $\theta^{(1)}, \theta^{(2)}, \ldots, \theta^{(n_u)}$:

$$\min_{\theta^{(1)},\ldots,\theta^{(n_u)}} \frac{1}{2}\sum_{j=1}^{n_u}\sum_{i:r(i,j)=1}\left((\theta^{(j)})^T x^{(i)} - y^{(i,j)}\right)^2 + \frac{\lambda}{2}\sum_{j=1}^{n_u}\sum_{k=1}^{n}(\theta_k^{(j)})^2$$

3 最优化：

Optimization algorithm:

$$\min_{\theta^{(1)},\ldots,\theta^{(n_u)}} \frac{1}{2}\sum_{j=1}^{n_u}\sum_{i:r(i,j)=1}\left((\theta^{(j)})^T x^{(i)} - y^{(i,j)}\right)^2 + \frac{\lambda}{2}\sum_{j=1}^{n_u}\sum_{k=1}^{n}(\theta_k^{(j)})^2$$

$J(\theta^{(1)},\ldots,\theta^{(n_u)})$

Gradient descent update:

$$\theta_k^{(j)} := \theta_k^{(j)} - \alpha\sum_{i:r(i,j)=1}((\theta^{(j)})^T x^{(i)} - y^{(i,j)})x_k^{(i)} \quad (\text{for } k = 0)$$

$$\theta_k^{(j)} := \theta_k^{(j)} - \alpha\left(\sum_{i:r(i,j)=1}((\theta^{(j)})^T x^{(i)} - y^{(i,j)})x_k^{(i)} + \lambda\theta_k^{(j)}\right) \quad (\text{for } k \neq 0)$$

分两种情况 k=0 是截距项，没有正则化项

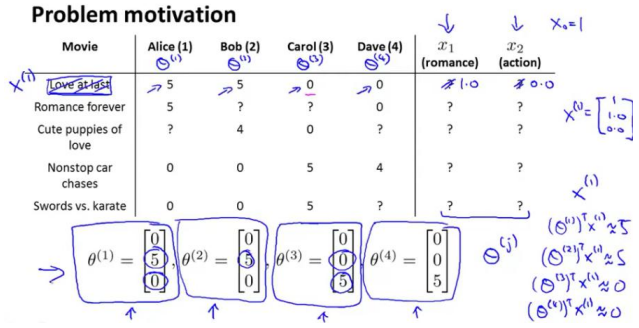
方法 2 Collaborative Filtering

上面的方法是人工确认 feature，有时候很难

我们需要自动 learn feature

all users collaboratively help system get better movie rating for

everyone



假设：

每个 user 告诉了 **对所有 feature**(两种电影 romance 和 action)的**喜好评分**，5 是喜爱，0 不喜欢

theta1(050)说明对 romance 喜欢，对 action 不喜欢

我们可以用 θ^1 来推断 x^1，**也是**为了让误差更少

Optimization algorithm

Given $\theta^{(1)}, \ldots, \theta^{(n_u)}$, to learn $x^{(i)}$:

$$\min_{x^{(i)}} \frac{1}{2} \sum_{j:r(i,j)=1} ((\theta^{(j)})^T x^{(i)} - y^{(i,j)})^2 + \frac{\lambda}{2} \sum_{k=1}^{n} (x_k^{(i)})^2$$

x 是参数，一部电影的 feature

Given $\theta^{(1)}, \ldots, \theta^{(n_u)}$, to learn $x^{(1)}, \ldots, x^{(n_m)}$:

$$\min_{x^{(1)}, \ldots, x^{(n_m)}} \frac{1}{2} \sum_{i=1}^{n_m} \sum_{j:r(i,j)=1} ((\theta^{(j)})^T x^{(i)} - y^{(i,j)})^2 + \frac{\lambda}{2} \sum_{i=1}^{n_m} \sum_{k=1}^{n} (x_k^{(i)})^2$$

所有电影的 feature

两种方法结合：

Collaborative filtering

Given $x^{(1)}, \ldots, x^{(n_m)}$ (and movie ratings),
can estimate $\theta^{(1)}, \ldots, \theta^{(n_u)}$

Given $\theta^{(1)}, \ldots, \theta^{(n_u)}$,
can estimate $x^{(1)}, \ldots, x^{(n_m)}$

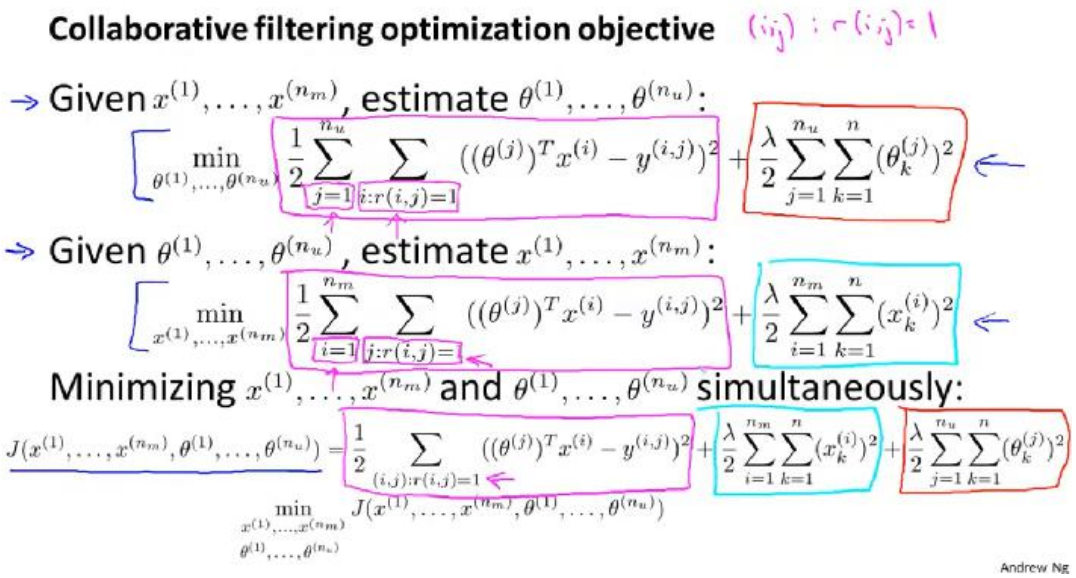
Guess $\Theta \to X \to \Theta \to X \to \Theta \to X \to \cdots$

通过初始化 theta，得到 x feature，再利用第一种方法得到更好的 theta，一直迭代直到聚合，直到误差很小!!

Collaborative Filtering Algorithm

之前是通过初始化 theta,得到 x feature,再利用第一种方法得到更好的 theta,一直迭代直到聚合

有更好的方法，合并一起优化：



put together，同时 learn **theta 和 x**

红色部分是一样，只是下标不同 i 是 movie，j 是 user

上面的下标：对于所有用户　sum 评论过的电影

下面的下标：对于所有电影　sum 评论过的用户

上下其实是等价的都是计算**(i,j):r(i,j)=1**

所以上面三个红框式子都是等价的!!

合起来是 all pairs(i,j)

唯一的不同是，没之前没那么麻烦，一直迭代，**此时可以同时训练参数!!!**

此时要把截距项 x0 去掉

Collaborative filtering algorithm

1. Initialize $x^{(1)},\dots,x^{(n_m)},\theta^{(1)},\dots,\theta^{(n_u)}$ to small random values.
2. Minimize $J(x^{(1)},\dots,x^{(n_m)},\theta^{(1)},\dots,\theta^{(n_u)})$ using gradient descent (or an advanced optimization algorithm). E.g. for every $j=1,\dots,n_u, i=1,\dots,n_m$:

$$x_k^{(i)} := x_k^{(i)} - \alpha\left(\sum_{j:r(i,j)=1}((\theta^{(j)})^T x^{(i)} - y^{(i,j)})\theta_k^{(j)} + \lambda x_k^{(i)}\right)$$

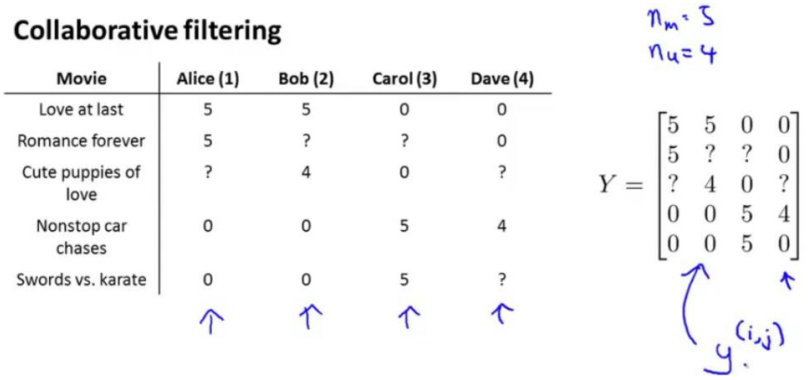
$$\theta_k^{(j)} := \theta_k^{(j)} - \alpha\left(\sum_{i:r(i,j)=1}((\theta^{(j)})^T x^{(i)} - y^{(i,j)})x_k^{(i)} + \lambda\theta_k^{(j)}\right)$$

$\frac{\partial}{\partial x_k^{(i)}} J(\dots)$

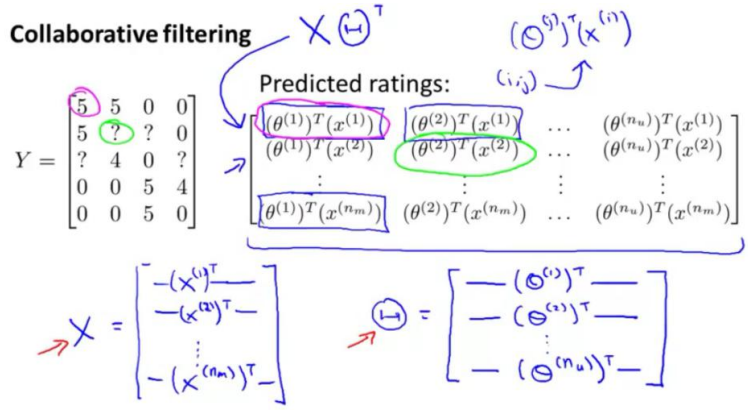
其实，还是求每个参数的偏导数，然后更新参数

3. For a user with parameters θ and a movie with (learned) features x, predict a star rating of $\theta^T x$.

Low Rank Matrix Factorization



评分转化成矩阵



x1 是 movie1 的 feature 向量

theta1 是 user1 的评分向量

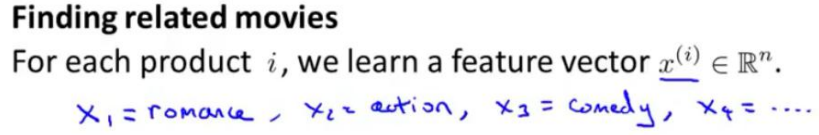
这个算法也叫：Low Rank Matrix Factorization

因为我们得到的 Y 是个 low rank matrix 低秩矩阵

low rank matrix 分解为两个矩阵，因此是 Factorization

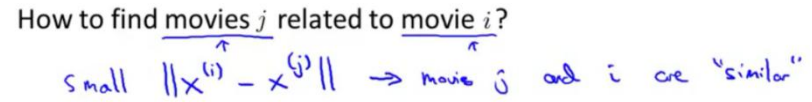
Recommand(相似性计算)

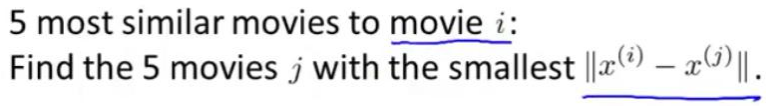
推荐相关联产品



学习到的 feature，我们并不知道它的具体含义，但确实是会代表 product 的一些重要特征！

找到相关产品



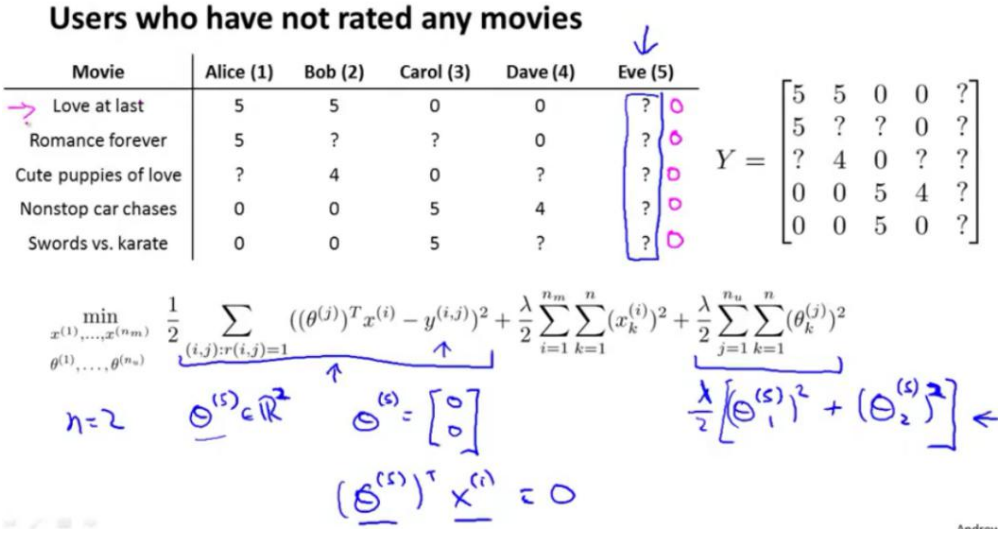


small distance 意味着相似度

Implementational Detail----（Mean Normalization）

make the algorithm better

假如 一个新 user 没有评分任何 movie



cost function 第一项和 theta5 没有关系，因为 user5 没有评分就没有 y，第一项只是对电影有评分的求 sum

但每个 user 都会对 x 有个评分，theta 向量

因此 theta5 也存在，只会受正则项影响，如果最小化，应该 theta5 都为 0（因为参数 theta5 的 cost 只有正则项，0 肯定最小）

theta5*x 评分结果也是 0，这样不是很合理

需要更好的预测值，才能更好的 recommand movie 给新用户

Mean Normalization:

只对有值的计算均值

Mean Normalization:

$$Y = \begin{bmatrix} 5 & 5 & 0 & 0 & ? \\ 5 & ? & ? & 0 & ? \\ ? & 4 & 0 & ? & ? \\ 0 & 0 & 5 & 4 & ? \\ 0 & 0 & 5 & 0 & ? \end{bmatrix} \quad \mu = \begin{bmatrix} 2.5 \\ 2.5 \\ 2 \\ 2.25 \\ 1.25 \end{bmatrix} \rightarrow Y = \begin{bmatrix} 2.5 & 2.5 & -2.5 & -2.5 & ? \\ 2.5 & ? & ? & -2.5 & ? \\ ? & 2 & -2 & ? & ? \\ -2.25 & -2.25 & 2.75 & 1.75 & ? \\ -1.25 & -1.25 & 3.75 & -1.25 & ? \end{bmatrix}$$

计算每一行有数据的均值，有数据的减去均值

normalize the average to 0

新的 Y 的每个元素**加上均值 u 就是真实值**

把新的矩阵 Y 假装成为我们的**用户评分**

用新的 Y 来进行 collaborative filter algorithm，去 learn 参数 **SUM 误差 (theta*x-Y)**

最后得到的 theta 和 x，相乘以后再加上我们的均值！

For user j, on movie i predict:

$$(\theta^{(j)})^T(x^{(i)}) + \mu_i$$

User 5 (Eve):

$$\theta^{(5)} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \qquad \underbrace{(\theta^{(5)})^T(x^{(i)})}_{=0} + \mu_i$$

这样 user5 的 rating 就不是 0 了

3 Categorical Input Data(encoding embedding)

Categorical data : input features that represent one or more discrete items from a finite set of choices. （固定集合里选几个）

Categorical data is most efficiently represented via sparse tensors, with very few non-zero elements.就是上面[1,4,19]

比如：

1 User 数据(看过哪些电影) : [1,4,10]看过第 1,4,10 部电影

每个人，看过的电影记录，每一行代表一个 user(user 的观影习惯)

The last row corresponds to the sparse tensor [1, 3, 999999]

2 the set of words in a document（一个 vocabulary，每个 document 出现的 word:[1,2,5,11.....]）

我们需要将上面编码为 ML 能够利用的 input!!

In order to use such representations within a machine learning system, we need a way to represent each sparse vector as a vector of numbers so that semantically similar items have similar distances in the vector space. 相似的事务，有较近的距离!! 上面不能表示距离!!

上面的电影[1,2,999]，我们想获得电影的相似性，而不是电视相似性!!

方法一：bag of words

假如 dict 100 个词汇

vector 的长度是 100.文章中出现了词汇，就标记 1(或者标记频率)，没出现的标记为 0

但是有时太稀疏了!! 0 太多!!

更刚才的电影就一样，每个人一个 bag of words：每个人每个观看记录一个 onehot vector

会带来如下几个问题

1 Huge input vectors mean a super-huge number of weights for a neural network.

需要大的计算量，而且需要 large Amount of data. The more weights in your model, the more data you need to train effectively.

2Lack of Meaningful Relations Between Vectors

If you feed the pixel values of RGB channels into an image classifier, it makes sense to talk about "close" values. Reddish blue is close to pure blue, both semantically and in terms of the geometric distance between vectors. But a vector with a 1 at index 1247 for "horse" is not any closer to a vector with a 1 at index 50,430 for "antelope" than it is to a vector with a 1 at index 238 for "television".

The Solution: Embeddings

Motivation: Why Learn Word Embeddings

Representing words as unique, discrete ids furthermore leads to data sparsity(data 见没啥关系)

If we represent 'cat' as Id537 and 'dog' as Id143. will treat words as discrete atomic symbols, and therefore These encodings provide no useful information to the system regarding the relationships that may exist between the individual symbols.

The solution to these problems is to use **embeddings**, which translate **large sparse vectors** into a **lower-dimensional space** that preserves **semantic relationships.** 需要训练，需要定义相似性 loss

This sort of meaningful space gives your machine learning system opportunities to detect patterns that may help with the learning task

例如：从 word of bag 是个大的向量，转向小的向量：

While we want **enough dimensions to encode rich semantic relations**, we also want an **embedding space that is small enough to allow us to train our system more quickly.** A useful embedding may be on the order of hundreds of dimensions. This is likely **several orders of magnitude smaller than the size of your vocabulary** for a natural language task.(比我们的词向量小了很多!!)

Embeddings as lookup tables

An embedding is a matrix in which **each column is the vector that corresponds to an item in your vocabulary.**

But how would you translate **a sparse bag of words vector**?

假如一段话,每个 word 一个 onehot encoding:

To get the **dense vector** for a sparse vector representing multiple vocabulary items (all the words in a sentence or paragraph), **you could retrieve the embedding for each individual item and then add them together.** (所有 word 的 embedding 相加)得到一个 embedding

If the sparse vector contains **counts** of the vocabulary items, you could **multiply each embedding by the count** of its corresponding item , 然后 adding it to the sum.

通过乘以 Embedding 矩阵获取 embedding 向量!!

Embedding lookup as matrix multiplication

The lookup, multiplication, and addition procedure we've just described is equivalent to **matrix multiplication**. Given a **1*N sparse representation S** and an **N*M embedding table E**, the matrix **multiplication S X E gives you the 1*M dense vector.**

3 How to Embedding?

There are many existing **mathematical techniques** for capturing the important structure of a high-dimensional space in a low dimensional space.

PCA

1 principal component analysis (PCA)

Given a set of instances like bag of words vectors, PCA tries to find highly correlated dimensions that can be collapsed into a single dimension.

Word2vec（Google）question

Word2vec relies on the **distributional hypothesis** to map semantically similar words to geometrically close embedding vectors.

理解为把 word onehot 转成 embedding vector

The distributional hypothesis states that words which often have the same neighboring words tend to be semantically similar. Both "dog" and "cat" frequently appear close to the word "vet", and this fact reflects their semantic similarity 语意相似

Word2Vec exploits contextual information: By training a neural net to distinguish actually co-occurring groups of words from randomly grouped words.

The input layer takes a sparse representation of a target word together with one or more context words.

This input connects to a single, smaller (一层 node 少的层)hidden layer(就是 Embedding)

1. In one version of the algorithm, the system makes a negative example by substituting a random noise word for the target word. Given the positive

example "the plane flies", the system might swap in "jogging" to create the contrasting negative example "the jogging flies".

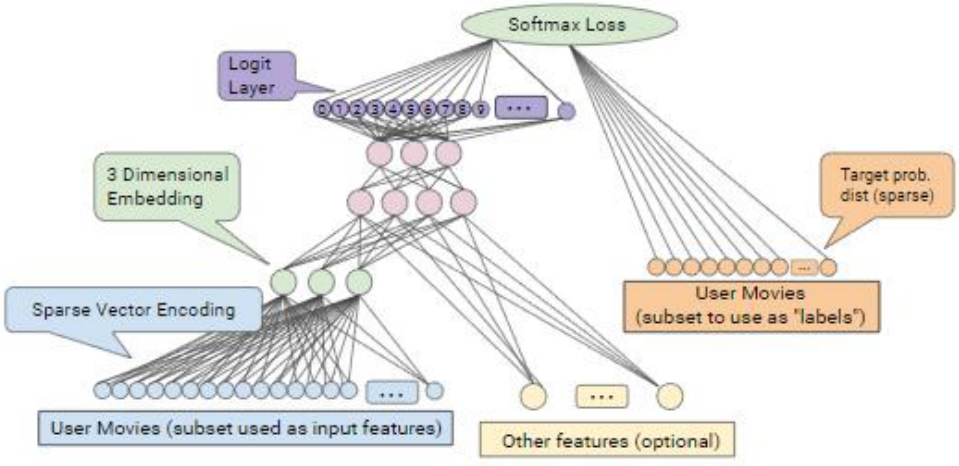
2 The other version of the algorithm creates negative examples by pairing the true target word with randomly chosen context words. So it might take the positive examples (the, plane), (flies, plane) and the 随机选取 context word ：negative examples (compiled, plane), (who, plane) and learn to identify which pairs actually appeared together in text.

After the model has been trained, you have an embedding. You can use the weights connecting the input layer with the hidden layer to map sparse representations of words to smaller vectors. This embedding can be reused in other classifiers.

Training an Embedding as Part of a Larger Model

You can also learn an embedding as part of the neural network for your target task. This approach gets you an embedding well customized for your particular system,

In general, when you have sparse data (or dense data that you'd like to embed), you can create an embedding unit (hidden unit of size d). This embedding layer can be combined with any other features and hidden layers.



The final layer will be the loss that is being optimized.

For example, collaborative filtering,predict a user's interests .

We can model this as a supervised learning problem by randomly setting aside (or holding out) a small number of the movies that the user has watched as the positive labels, and then optimize a softmax loss.

As another example if you want to create an embedding layer for the words in a real-estate ad as part of a DNN to predict housing prices then you'd

optimize an L2 Loss using the known sale price of homes in your training data as the label.

For more information about word2vec, see the tutorial on tensorflow.org ：https://www.tensorflow.org/tutorials/word2vec

原文作者：

Efficient Estimation of Word Representations in Vector Space https://arxiv.org/pdf/1301.3781.pdf

十二 Large Scale Machine Learning

1 Online Learning

和 stochasitic gradient descent 很类似

learning from 1 user，then discard，moving on，不再有固定的数据集。是持续的数据流

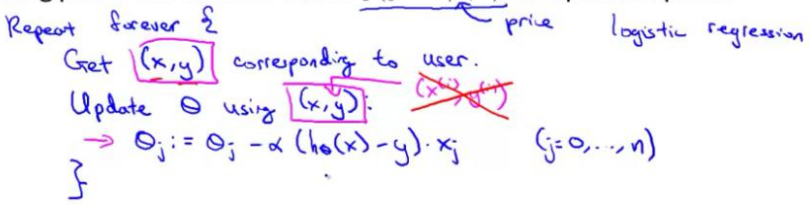
website learn from user data stream 数据流，一直源源不断，可以帮我们 do some decision

每一个 example 的 error 作为 cost，计算偏导数

Online learning

Shipping service website where user comes, specifies origin and destination, you offer to ship their package for some asking price, and users sometimes choose to use your shipping service ($y = 1$), sometimes not ($y = 0$).

Features x capture properties of user, of origin/destination and asking price. We want to learn $p(y = 1|x; \theta)$ to optimize price.



货运服务，

x:用户提供起点和终点，我们提供价格。

label:用户是否选择我们的服务

我们想建个模型，在给定 x 起点、终点、价格条件下，用户**选择我们的概率**，用 logistics 建模

每一个用户访问我们得到一个(x,y)，用该(x,y)来更新参数 theta，这个数据就不需要再保留(和传统算法)，再用下一个 user 数据，会持续学习。

不像其他算法，需要保存我们的 train set

而且 online learning 可以 adapt to user perference

可以自动适应用户的偏好变化

predict CTR click-through rate

the probability user click on

Other online learning example:

Product search (learning to search)

User searches for "Android phone 1080p camera"

Have 100 phones in store. Will return 10 results.

一共有 100 个手机，每次请求返回 10 款最匹配

learning search

x = features of phone, how many words in user query match name of phone, how many words in query match description of phone, etc.

$y = 1$ if user clicks on link. $y = 0$ otherwise.

Learn $p(y = 1|x; \theta)$.

在给定用户搜索内容基础上，对于所有手机，**构建 x：**

1 这款手机的 feature，2 以及用户搜索和这款手机的匹配度

然后记录用户的 click(每次 10 个 train examples)，得到每个手机的 label y

学习到： Learn $p(y = 1|x; \theta)$.

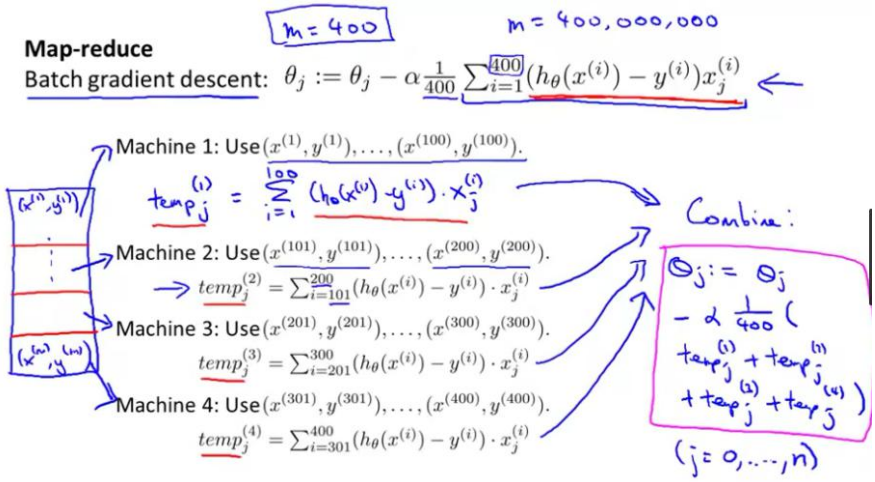
最终根据 x 来预测：用户点击某款手机的概率

根据这个值来推荐 10 款概率最高的手机

也就是每次用 10 training examples，进行 online learning algorithm 梯度更新参数，相当于 mini-batch=10。

2 Map Reduce and Data Parallelism

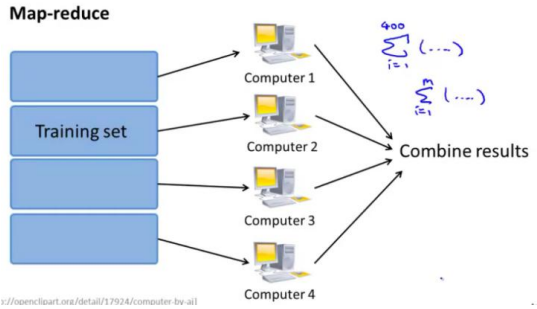
run ML on many computer



将 400 数据集分成四份 100，在多台机器上进行并行计算，计算每个 batch 的 gradient，**然后 sum 所有的 gradient，更新参数**

master server combine result together

会有 network delay



适用条件：

Map-reduce and summation over the training set

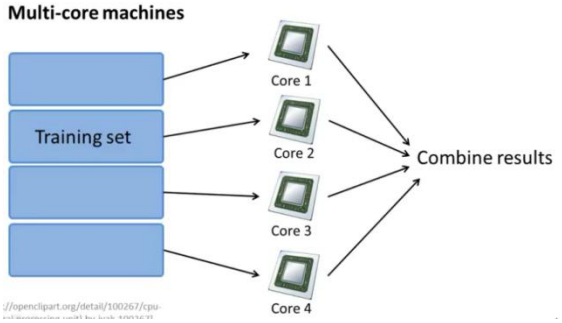
Many learning algorithms can be expressed as computing sums of functions over the training set.

E.g. for advanced optimization, with logistic regression, need:

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

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

更新参数是 sum 求和的形式



单个多核计算机

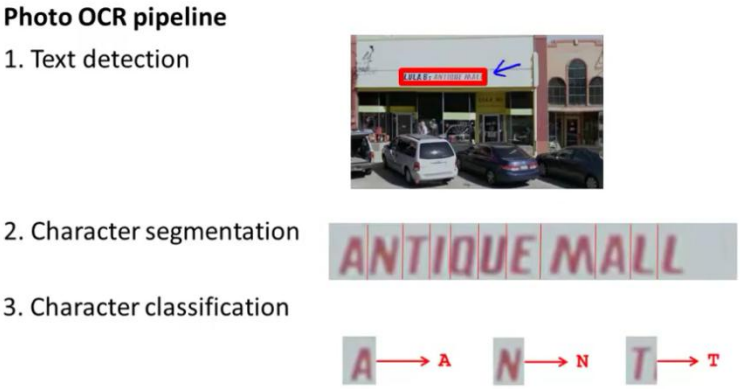
十三 Photo OCR (optical character recognition)项目

视觉字符识别，应用：

1 图片搜索：通过输入字符串就可以搜索到，照片中出现该字符串的照片

2 盲人道路识别

3 car 道路识别

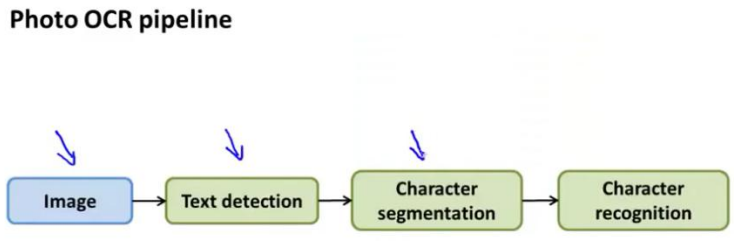


1 先文字识别，识别该区域是否是文字

2 字符分割

3 字符识别

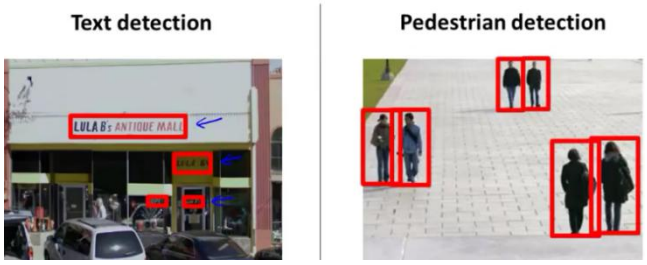
以上被称为 machine learning pipeline 流水线



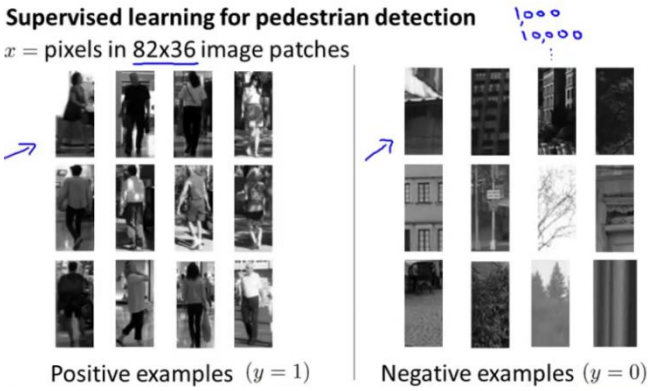
break problem into serveral moduel，每部分由不同人完成

1 Sliding Windows

1 先文字识别，识别该区域是否是文字



每个行人的窗口大小不同



先用统一的 size82*36 图片 train a neuron network，then classify

用 82*36window 来 silde，进行 detect

then 用一个更大的 window 去扫描

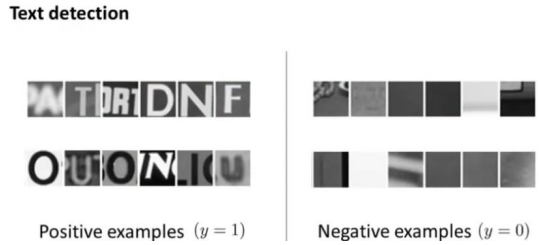
然后 resize down to 82*36，feed 分类器分类

然后用更大的 window

最后准确的得到某个区域是否是行人

2 Text detection

同上面的 sliding window 算法：先训练 classifier 固定 window



siiding 完得到每个字符的区域

将识别出的字符像素 patch 填充白色

通过 exspansion operator **膨胀算子，将左边的白色 region，**扩展成白色区域。

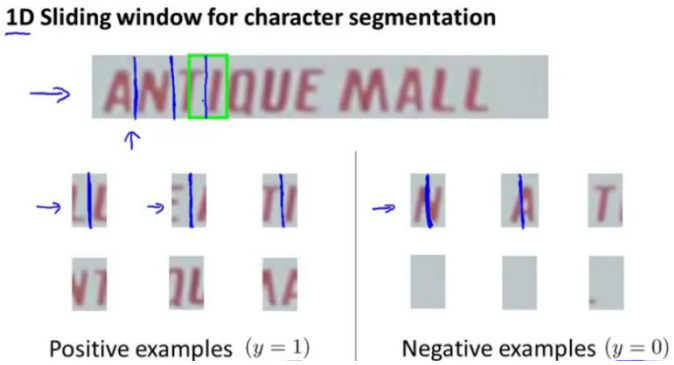
将左边相邻像素也填充成白色

然后丢弃一些，不规则的区域，剩下的是**文字区域**

完成了文字区域的识别

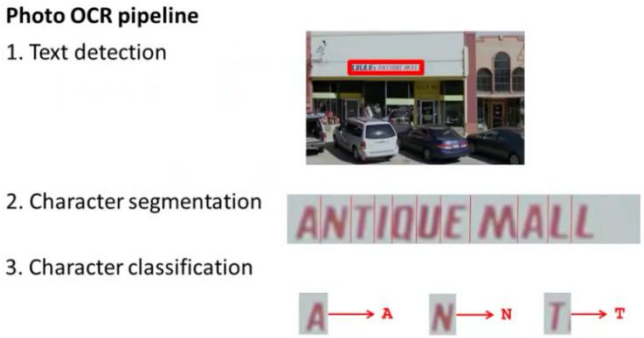
3 character segment 字符分割

也是 slide the window，先训练判断两字母直接是否有分割线，有的话就在这里分割字符图片。



我们自己制作 data set，去训练分类器

4 ocr pipeline

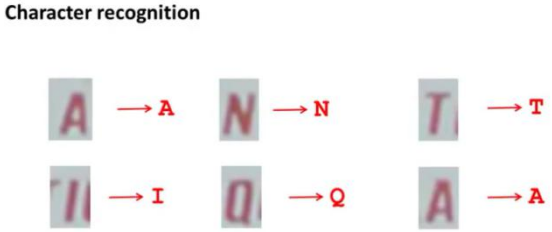


最后字符识别

最后 character recognize

5 Getting Lots of Data and Artificial Data(amplify train data)

performance well way: use no-bias algorithm, with large data



为了识别字符，我们**人工合成 dataset，电脑也自带了很多字体**

take char from different front, paste on random background(bg 不重要)

Artificial data synthesis for photo OCR

Real data

Artificial data synthesis for photo OCR

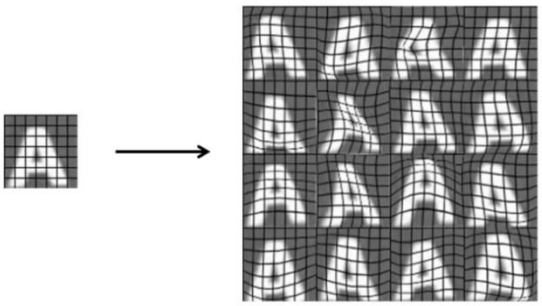
Real data　　Synthetic data

右边是合成 dataset，从电脑里找到字体粘贴上去的

第二个方法；

使字符失真：

Synthesizing data by introducing distortions



要根据实际应用来确定 distortion

amplify 扩充 train set

语音识别例子：

Synthesizing data by introducing distortions: Speech recognition

Original audio:

Audio on bad cellphone connection

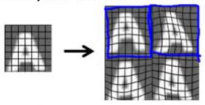
Noisy background: Crowd

Noisy background: Machinery

通过加各种背景音来扩充 train set

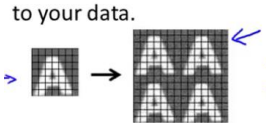
Synthesizing data by introducing distortions

Distortion introduced should be representation of the type of noise/distortions in the test set.



Audio:
Background noise,
bad cellphone connection

Usually does not help to add purely random/meaningless noise to your data.



x_i = intensity (brightness) of pixel i

$x_i \leftarrow x_i +$ random noise

synthesize 的前提是，这种 distortion 在 test set 里也出现了

不然就没意义，distortion 目的就是服务 test set

在搜集更多数据之前，先问几个问题：

Discussion on getting more data

1. Make sure you have a low bias classifier before expending the effort. (Plot learning curves). E.g. keep increasing the number of features/number of hidden units in neural network until you have a low bias classifier.
2. "How much work would it be to get 10x as much data as we currently have?"
   - Artificial data synthesis
   - Collect/label it yourself
   - "Crowd source" (E.g. Amazon Mechanical Turk)

1low-bias，如果不是增加 feature 或 unit

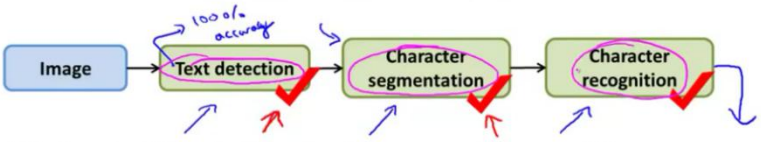
2 增加 10 倍的数据需要花费多少时间和工作量

众包 labeling，如网站验证，让用户去 labeling

6 Ceiling Analysis_ What Part of the Pipeline to Work on Next

find potential of improving each components

Estimating the errors due to each component (ceiling analysis)



What part of the pipeline should you spend the most time trying to improve?

Component	Accuracy
Overall system	72%
Text detection	89%
Character segmentation	90%
Character recognition	100%

查改进哪一部分对准确率的提升最有效!!!

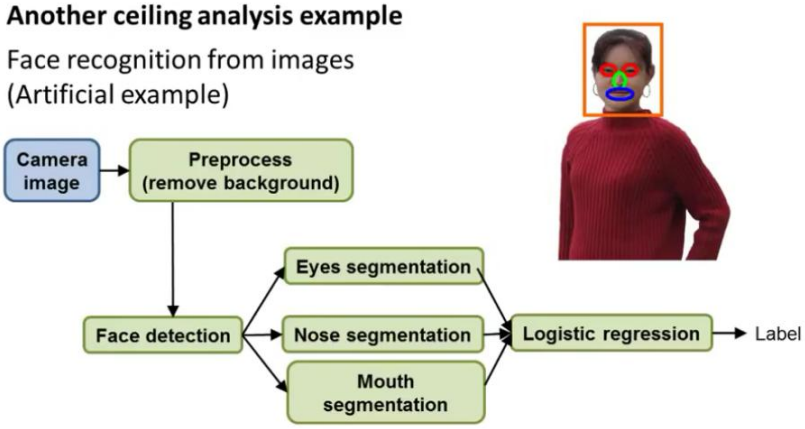
在 text detection 过程中，我们 manully 告诉分类器正确的答案，使其该过程准确率 100%。 看看我们总的准确率多少？能提高多少

计算每个模块的改进空间!!!让我们把更多的精力投放在改进空间最大的过程!!

例子：

Another ceiling analysis example

Face recognition from images (Artificial example)

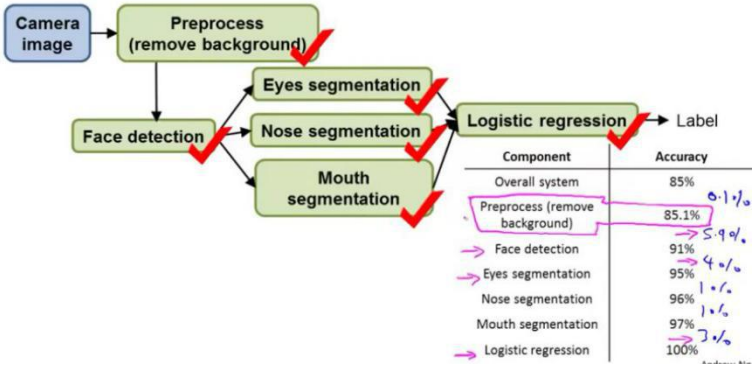


1 用 ps 去掉背景 preprocess

2 manully 给出正确的脸的区域

3 正确五官的区域

Another ceiling analysis example



查改进哪一部分对准确率的提升最有效!!!