# AGI

监督学习和不监督学习居多

强化学习很少

监督学习指 从X到Y输出映射的学习，最终给定输入而预测相当准确度的输出

### 监督学习中拟合曲线称为——回归

### 另一种应用：分类算法

### 结果不必只是数字 对可能的结果进行分类

## 无监督学习

给定数据与输出的Y没有关系：比如聚类算法：不提前设定分类依据，仅仅依据数据集本身的分布特征进行聚类

Jupyter Notebook

分类中只有少数给定集合中的输出

而在回归中，理论上该有连续的无限多的输出

### “训练集”

训练示例

**包括训练特征和输出目标**

Produce some function to take new x and produce a guess for y

Function is the model

Simple y means a true values in reality

代价函数，用于评估预测结果的函数

平方误差代价函数：J(W,B) = 1/2M\* SIGEMA从1-m（yhat-y）^2

## 梯度下降算法

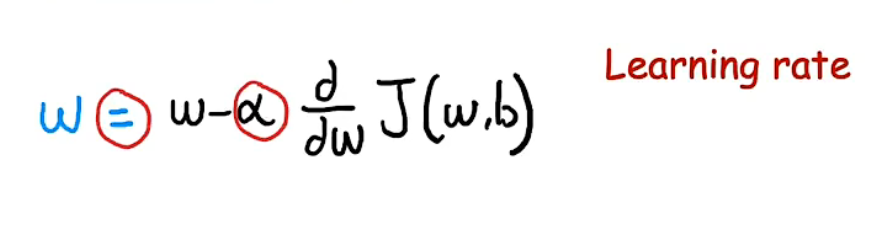
选定初始猜测值

持续改变w,b来改变J，直到得到合适的J

改变的依据是，每次选择斜率最大的方向进行迭代

但是在这里只考虑w和b自己的梯度变化

### 因为在该方向上，w的变化会引起代价的增加，因此w要往反方向变动，由是能够使得代价减少。



# 多元线性回归

F(x) = 向量w.向量x+ b

## 矢量化思想

即矩阵化数据集然后使用numpy数组运算

专用于线性回归的方法：

正规方程法 不需要迭代的方法

但是不适用大量数据集的分析

## →特征缩放

缩放某个自变量的值范围来防止不同变量之间特征值相差过大

使得每个自变量的系数的值都在一个大致相当的范围

除以最大值、

### 均值归一化：

求完均值mean

x\_ = x- mean/(max- min)、

### Z值标准化

Mu sigma

X1 = (x1- mu1)/sigma1

当特征值比较大的时候好的模型会试着使用比较小的参数。

根据经验，执行特征缩放的时候总是希望把缩放后的值控制在-1:1之间

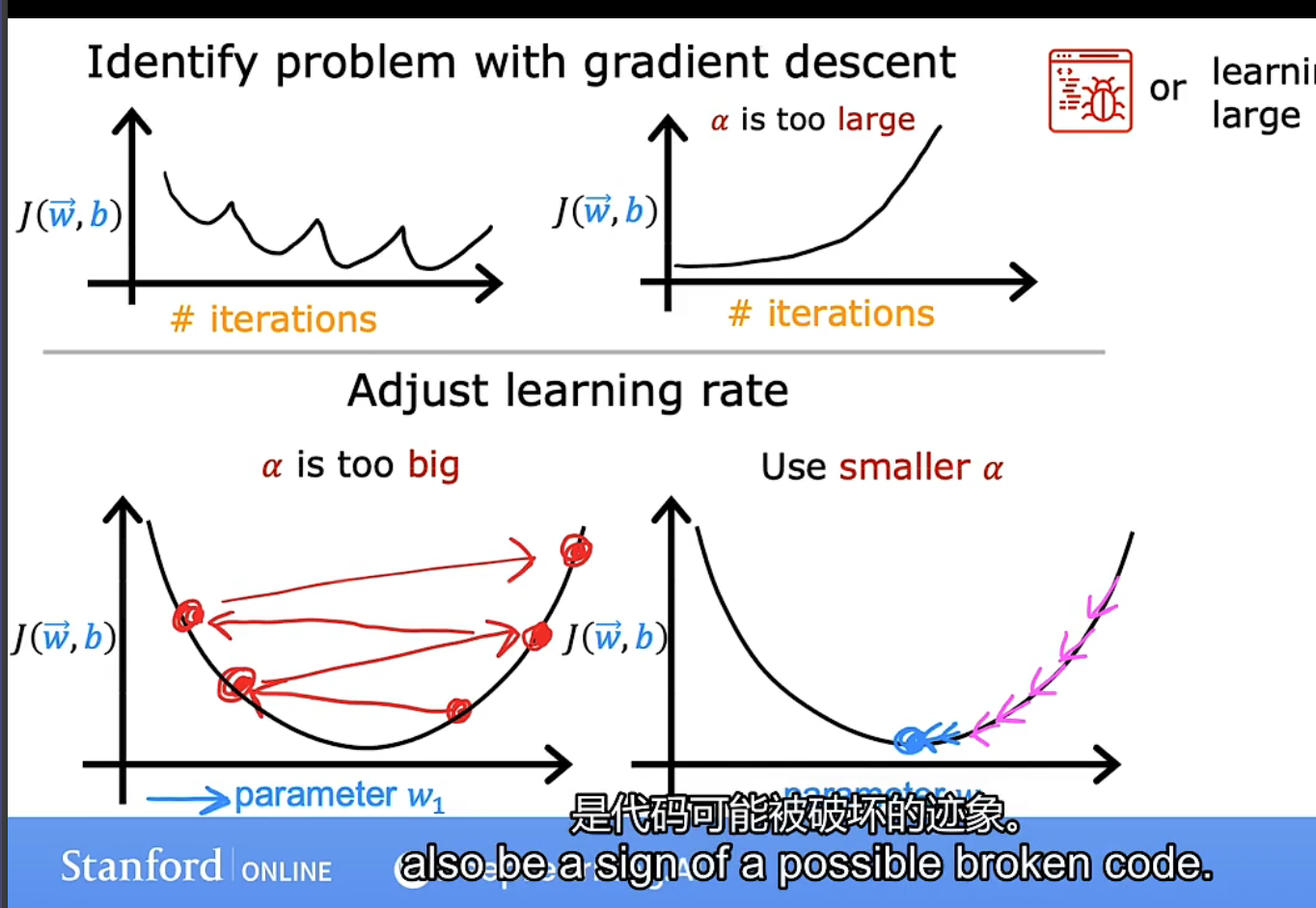
或者避免值域过于松散或者紧凑的时候可以适当的缩放

特征缩放可以加快梯度下降的速度

通过绘制成本函数关于迭代次数的图像来观察

正常情况下随着迭代次数的增多，成本函数会逐渐收敛于某一值

若随迭代次数成本忽大忽小可能说明代码有问题



若梯度下降不起作用，尝试把学习系数设置的很小

推荐从小到大尝试系数

## 特征工程

选择正确的、有效的特征

预设多项式函数形式次数越大特征缩放越重要

## 逻辑回归

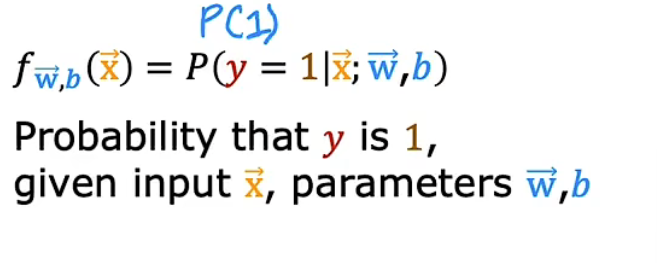
一种分类算法的解决方案

逻辑函数 输出值在0-1

### G(z) = 1/ (1+e^-z)

### Z = wx+b

输入特征值得到0-1的概率

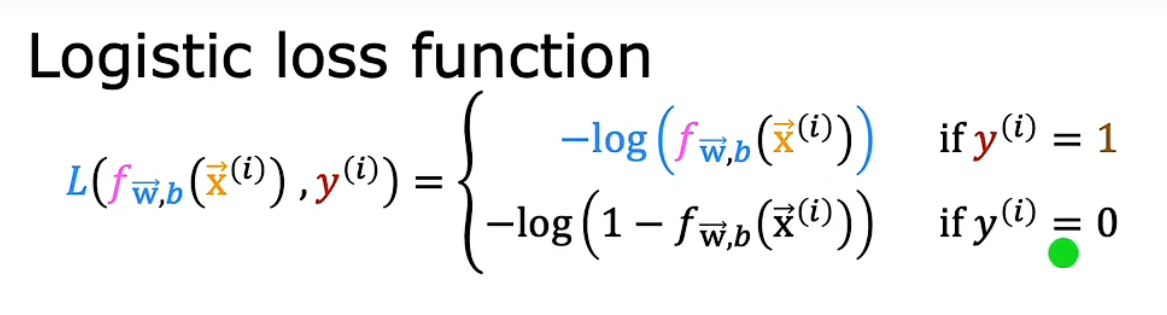


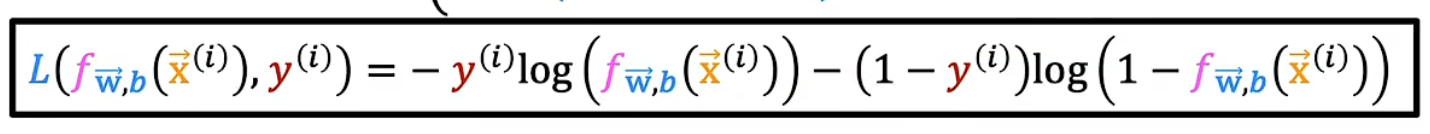
也存在g(z) = g(w1x1+w2x2 + b)

Z = f(x1, x2)

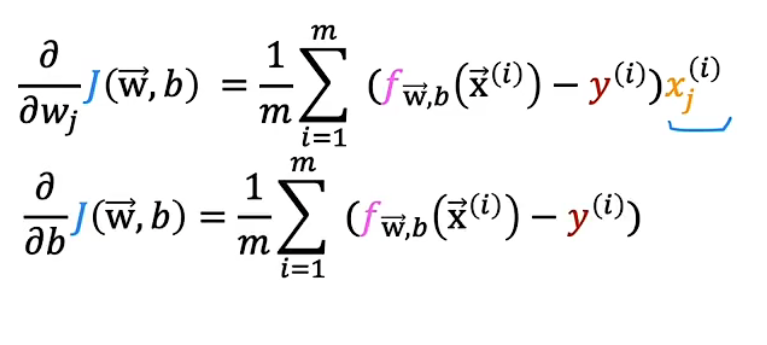
逻辑回归的成本函数

平方误差成本函数并不适用





仍然使用梯度下降的方法接近最优解



寻找决策边界：即使得z = w\*x +b 恰好= 0的那一条线

为**凸函数**的成本函数才适用梯度下降，否则容易陷入局部最优

利用矢量化和特征缩放来优化代码

过拟合和欠拟合

### 泛化

对训练集之外的数据进行比较好的拟合

**不希望有太高的方差也不希望有太大的偏差**

### 高方差很可能意味着过拟合

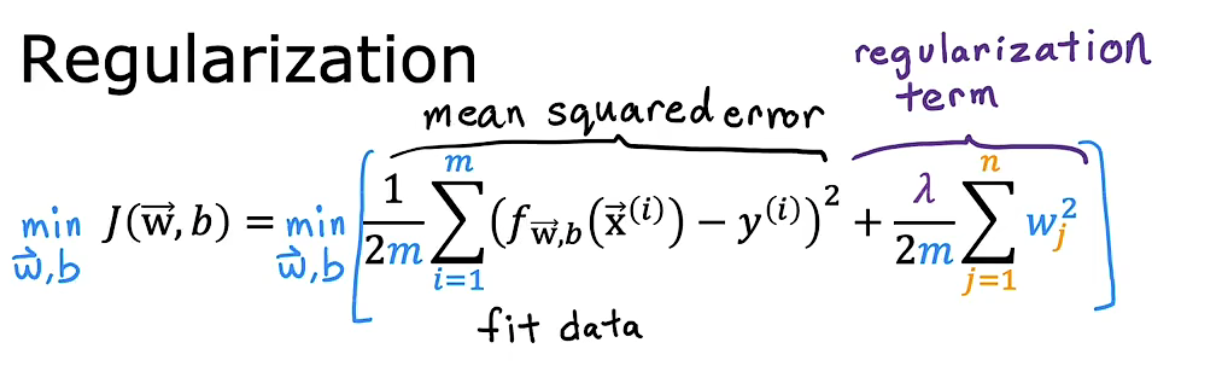
Just right

通过寻找更大的训练集或者减少特征数目来防止过拟合

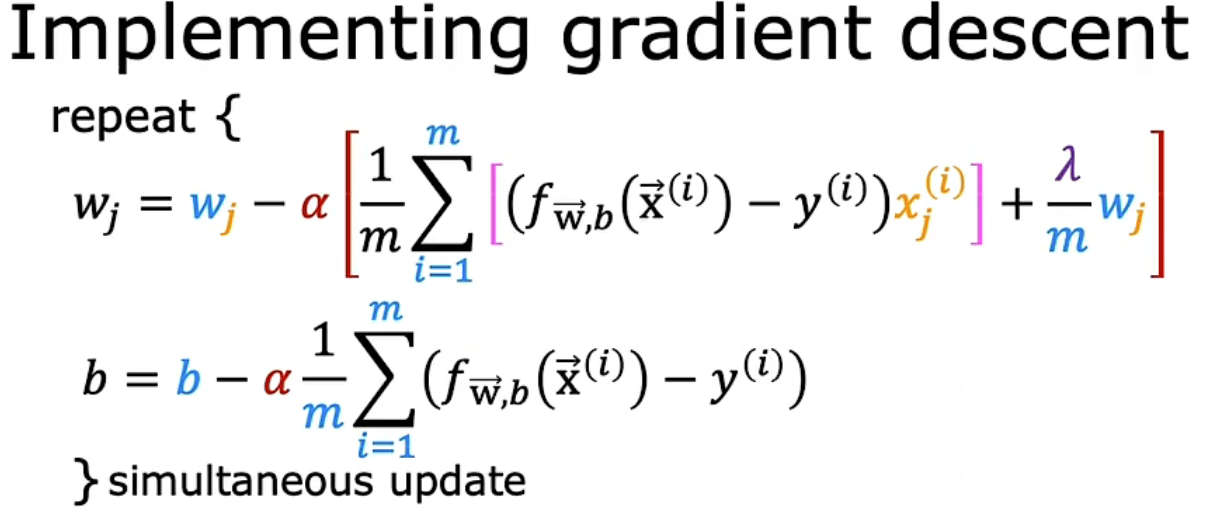
特征选择

### 正则化：鼓励缩小参数而非去除特征

修改成本函数的形式，使特定系数前的参数放大，可以使迭代过程中该参数趋于相对小的值



对其求偏导，得到对应的梯度下降关系式



在每次迭代中稍微缩小一点点wj

### 深度学习/ 神经网络

### 下载别人训练好了的神经网络——推理

以及训练自己的神经网络

训练自己的神经网络

predict + train

决策树

同时模拟多神经元

普通的机器学习无法太有效地利用所有训练集数据

是一类新的算法

GPU是很好的深度学习的硬件

给定输出，经过训练好的神经网络得到输出

demand prediction

price/shipping cost/marketing/material

↓

layer

↓

“activation”

affordability/awareness/perceived quality

↓

layer

↓

probability of being a top seller

input layer-hidden layer-…-output layer

实践中，每一个特征都会被layer访问，而非被手动选择,隐藏层的激活特征也不需要手动设计

X-A-output

每一个层中有多少元需要设计神经网络架构

## 脸部识别

亮度0-255

训练一个神经网络，以百万像素亮度值为特征向量识别人脸特征

### 可能发生的计算：

### 第一个层：寻找低垂直线或者类似垂直边缘，寻找非常短小的线

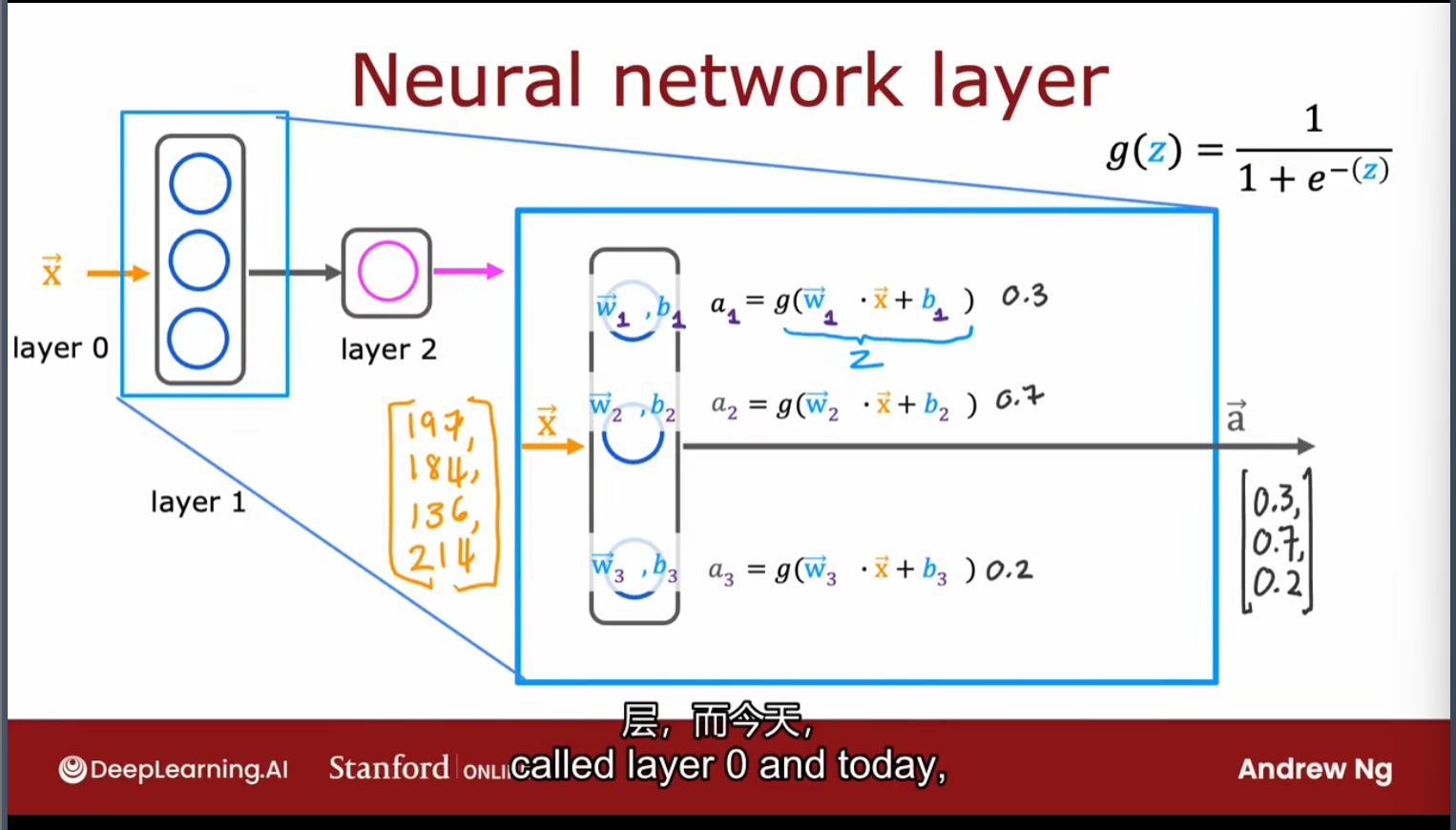
### 第二层：把小短线组合在一起

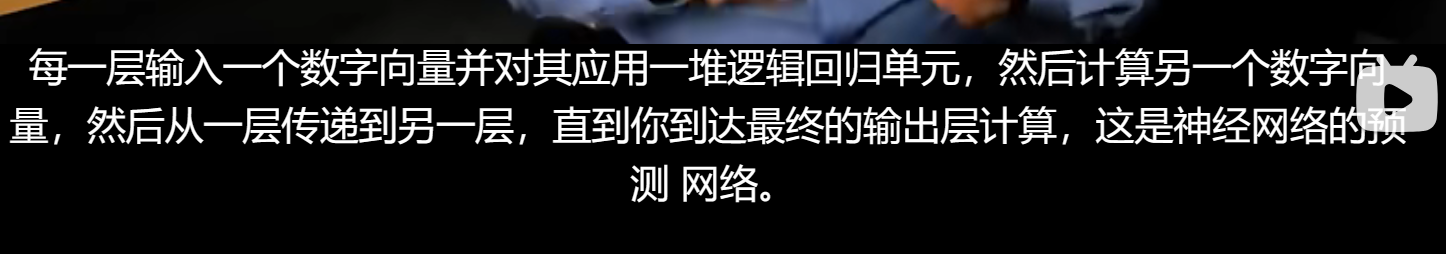
### 第三层：再组合

每一层都在尝试实现一个小逻辑回归函数

给定输入再输出的是“激活值”

“特征识别”





说神经网络有几层时，通常包括输出层但是不包括输入层

激活函数是指输出这些激活值的函数

手写识别

识别手写的0和1

隐藏单元会随着计算的推进越来越少

## 最重要的是如何设计“特征识别”的算法，一步一步拆解问题到可编程的地步，再一步一步组装起来输出需要的结果

Tensorflow和numpy表达矩阵的方式不完全一样

一个tensor对象可以通过.numpy()来转换

上一层激活值→激活函数→激活值→下一个密集层

Np.dot()

ANI+AGI= AI

代码的矢量化实现：即直接进行矩阵计算，以加快代码效率

Np.matmul()

A\_T = A.T

@ = np.matmul

计算机长于矩阵运算

## FP网络

### dense的作用是什么？

def dense(a\_0, W, b, g)：

units = W.shape[1]

a\_out = np.zeros(units)

for j in range(units):

w = W[:, j]

z = np.dot(w, a\_0)+ b[j]

a\_out[j] = g(z)

return a\_out

# 神经网络

def sequential(x):

a1 = dense(x, W1, b1)

a2 = dense(a1, W2, b2)

a3 = dense(a2, W3, b3)

f\_x= a3

return f\_x

二分类：建议sigmoid

线性回归：linear regression

假如y只能非负，则推荐ReLU

# 现在开始总结一下deepLearning到底学了什么。

首先和普通的机器学习不同的是，深度学习他提出了一个“神经元”的概念，并且模仿大脑中神经冲动的传送机制设计了神经网络的学习框架，即几个不同的神经元构成一个神经网络，通过这样一个神经网络来完成对训练集的学习并且在之后接收输入，达成输出。

而对于每一个神经元（Dense）来说，Dense内部的拟合过程和原先的机器学习没有什么不同，也是通过调整*y*=*σ*(*Wx*+*b*)中的W和b来最小化损失函数，然后再接着往下

## 卷积层

网络中的单个元只会接收局部的数据

而后向的神经网络也可能只接收上一个神经网络的局部输出。

BP网络是一种高效的计算LOSS关于某Param导数的方法

可以在测试集上计算损失函数由是避免针对训练集的过拟合

# 接下来是一堆模型调优的操作

## 1. 当J\_train无法客观描述模型的性能时，引入J\_cv在交叉数据集上测试模型，从而得到相对准确的评价

## 2. 主要观察bias和variance，排列组合之后有四种情况

bias 高 variance高

模型完全就不行，可能是方程形式不对，也可能是别的问题导致欠拟合

bias 高 variance 低

模型很奇怪，对交叉集有效但对训练集无效

bias 低 variance 高

常见情况，模型只对训练集有效，可能过拟合

bias 低 variance 低

理想情况，模型对训练集和交叉集都有效

## 3. 应对方法

### 修改方程形式

### 扩充数据集

1. 给没打标签的数据打标签

2. 略微修改已有数据得到新数据，比如在图像识别时可以把已有的数据进行放大、缩小、旋转等操作，在图片上放置网格，进行随机扭曲；以上操作同样适用于语音识别（添加噪声

3. 正则化方法可以减少w，防止过拟合，减少variance，而当需要降低bias时，可能就需要复杂化方程形式，或者增加数据集规模等。

4. 学习曲线，模型性能关于某东西（比如正则化参数、学习率、数据集规模等）的曲线

# 迁移学习

将相似问题的神经网络的参数传递给自己，使得神经网络能以一组相对准确的参数开局

用自己的输出层替换别人的输出层，然后进行微调和训练

机器学习的完整周期

## 1. 确定项目范围

## 2. 定义/收集数据

## 3. 训练模型

## 3-2-3循环直到模型精度足够

## 4. 部署使用，并持续运维

## 4-3-2-3-4循环

## MLOps

处理倾斜数据集时我们通常使用不同的错误度量而非单纯的分类误差

常见的有精确率和召回率

给出混淆矩阵

true positive

true negative

false positive

false negative

precision = true positive/ predicted positive

recall = true positives/ actual positive(true pos+ false neg)

理想情况就是hi precision+ hi recall

但是实际上二者不可得兼

precision-recall曲线

一个选择模型的方法，看precision-recall的平均值，但是这个方法并不好。

一种常见的方法，F1 SCORE

结合了PR两个值，但是更看重其中的较小值。

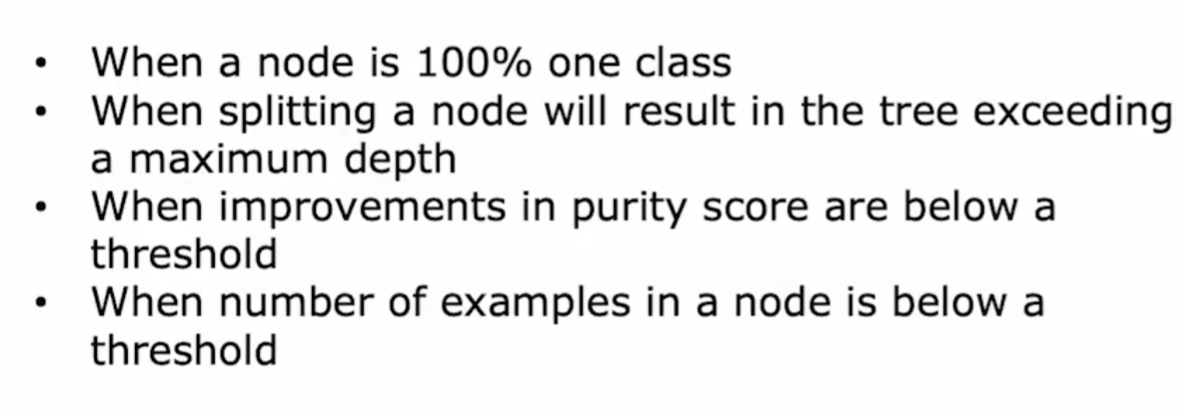
**F1 = 1/ 0.5(1/P + 1/R)**

**= 2PR/(P+R)**

## 决策树和决策森林

1. 如何选择feature作为树节点

2. 何时停止分裂？



当每一个特征非二元，或者非离散时如何呢？

### one-hot编码

当离散变量非二元时，对离散值编码再投入神经网络

当变量为连续时，对数据集依据某特征进行排序，逐个检视数据集中该特征的值，得到各区间内不同的信息熵，然后行前故事

### 回归树

可以换一个指标，熟悉的MRE什么的都可以用来作为选择分裂特征的指标

### 决策森林

得到多个决策树模型，在每次预测或者判断时综合所有树的结果给出输出

类似你最喜欢的聚类集成

### 1. 放回抽样

从训练集中随机抽取样例，放回抽样，如何一直重复

会得到一个新训练集，然后再训练一个新决策树

### 2. 随机选择feature-随机森林

### XGBoost

对于决策森林中表现不佳的决策树

### 何时使用决策树？

对于结构化数据来说，效果甚佳

但是对于非结构化数据，比如文本、语音、图片而言，就不行了

### 对计算速度有要求时，可以考虑决策树

## 神经网络的通用性就比较强

但是计算时间比较长

需要和迁移学习结合

# \*\*XGBoost原理\*\*

基于GBDT改进的一种快速优化决策树的算法，实际原理相当复杂

## 无监督学习

聚类/推荐系统

### K-Means

1. 随机猜一个簇质心

2. 初步分一类

3. 修改质心位置，为类中心

4. 接着重复2.3.

### 肘部法则

## 异常检测算法

**找出变量在训练集中的概率密度**

对于新的数据点，比对其值在数据集中出现的概率，

假定训练集各特征符合正态分布，连乘得到每个特征点的概率，如何查看其是否小于既定阈值

假如各种数据都充足的话，可以考虑直接用监督学习，反之无监督学习。

但是监督学习，可能无法发现未来新特征导致的值变化

选择特征，对于无监督学习很重要

假如某特征表现得并非正态分布，可以考虑**对其作变化以更接近高斯分布，比如取对数**

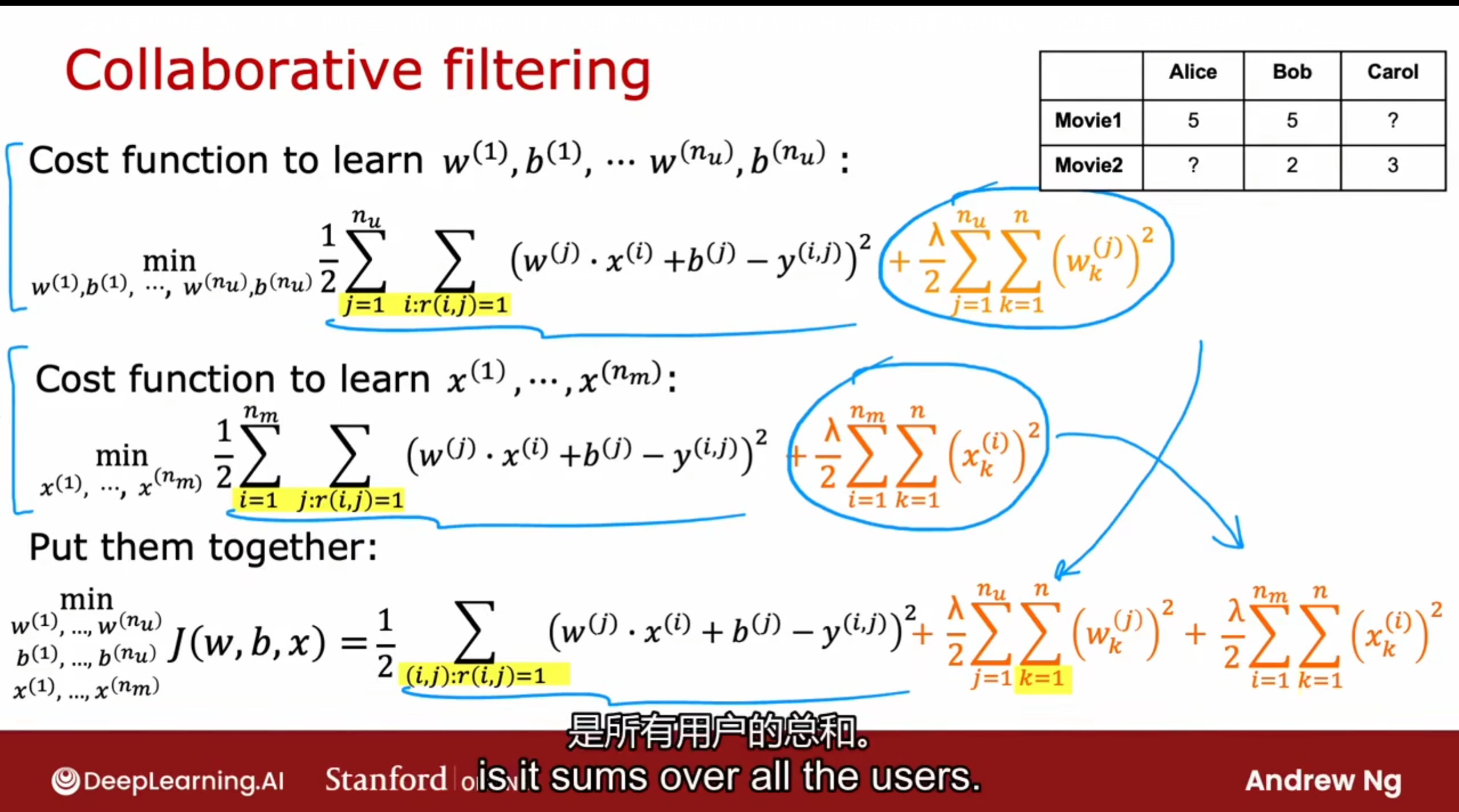
但也有可能使得异常点和正常点混淆

## 推荐系统

抽象出对象的特征(feature)

定义损失函数

协同过滤算法



一种方法，用梯度下降直接干三个参数w,b,x

# 技术总结一下

吴恩达这个协同过滤，已知信息非常少，各种参数（parameter）都是纯用机器学习算法算出来的，和之前数据挖掘学到的有所不同，光看视频不太好理解，现在开始看代码

有空的话要把以前数据挖掘的代码粗略看一下，最近的代码过一遍，强化学习看完，清明之后就要开始干大模型了。

现在的问题是，X的维度是怎么确定的？因为有M部电影，所以会有M行，这我知道，但是会有多少列呢？

现在知道他怎么处理的了，是直接假定一个列数来操作的。

# 基于内容的过滤算法

UserNetwork

给定Xu输出vu，vu有32个单元

给定Xm输出vm，vm也有32个单元

vu表示用户的32个评价字段

vm表示电影在这32个字段上的得分

相似度寻找相似的movie

# PCA非线性回归

state s -> action a

reward function

terminal state

discount factor gama

state --policy🡪 action

## 马尔科夫决策过程MDP

## 状态动作值函数

## 贝尔曼方程

## 随机马尔科夫过程

## 预期回报

## 小批量梯度下降

## 软更新