BAT机器学习面试1000题系列

来源：https://blog.csdn.net/v_july_v/article/details/78121924

整理：July、元超、立娜、德伟、贾茹、王剑、AntZ、孟莹等众人。本系列大部分题目来源于公开网络，取之分享，用之分享，且在撰写答案过程中若引用他人解析则必注明原作者及来源链接。另，不少答案得到寒小阳、管博士、张雨石、王赟、褚博士等七月在线名师审校。
说明：本系列作为国内首个AI题库，首发于七月在线实验室公众号上：julyedulab，并部分更新于本博客上，且已于17年双十二当天上线七月在线官网、**七月在线Android APP**、**七月在线iPhone APP**，后本文暂停更新和维护，新题都已更新到七月在线APP或七月在线官网题库板块上，欢迎天天刷题。另，可以转载，注明来源链接即可。

前言

July我又回来了。

之前本博客整理过数千道微软等公司的面试题，侧重数据结构、算法、海量数据处理，详见：微软面试100题系列，今17年，近期和团队整理BAT机器学习面试1000题系列，侧重机器学习、深度学习。我们将通过这个系列索引绝大部分机器学习和深度学习的笔试面试题、知识点，它将更是一个足够庞大的机器学习和深度学习面试库/知识库，通俗成体系且循序渐进。

此外，有四点得强调下：

1. 虽然本系列主要是机器学习、深度学习相关的考题，其他类型的题不多，但不代表应聘机器学习或深度学习的岗位时，公司或面试官就只问这两项，虽说是做数据或AI相关，但基本的语言（比如Python）、编码coding能力（对于开发，编码coding能力怎么强调都不过分，比如最简单的手写快速排序、手写二分查找）、数据结构、算法、计算机体系结构、操作系统、概率统计等等也必须掌握。对于数据结构和算法，一者 重点推荐前面说的微软面试100题系列（后来这个系列整理成了新书《编程之法：面试和算法心得》），二者 多刷leetcode，看1000道题不如实际动手刷100道。
2. 本系列会尽量让考察同一个部分（比如同是模型/算法相关的）、同一个方向（比如同是属于最优化的算法）的题整理到一块，为的是让大家做到举一反三、构建完整知识体系，在准备笔试面试的过程中，通过懂一题懂一片。
3. 本系列每一道题的答案都会确保逻辑清晰、通俗易懂（当你学习某个知识点感觉学不懂时，十有八九不是你不够聪明，十有八九是你所看的资料不够通俗、不够易懂），如有更好意见，欢迎在评论下共同探讨。
4. 关于如何学习机器学习，最推荐机器学习集训营系列。从Python基础、数据分析、爬虫，到数据可视化、spark大数据，最后实战机器学习、深度学习等一应俱全。

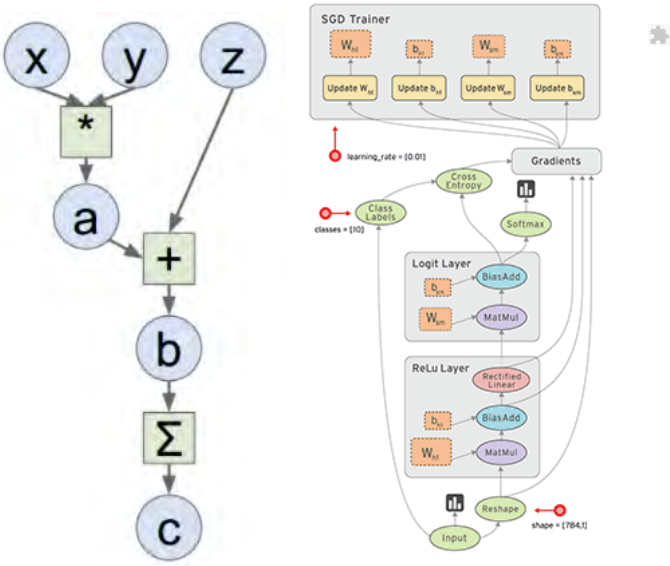
另，本系列会长久更新，直到上千道、甚至数千道题，欢迎各位于评论下留言分享你在自己笔试面试中遇到的题，或你在网上看到或收藏的题，共同分享帮助全球更多人，thanks。

BAT机器学习面试1000题系列

1. 请简要介绍下SVM，机器学习 ML模型 易SVM，全称是support vector machine，中文名叫支持向量机。SVM是一个面向数据的分类算法，它的目标是为确定一个分类超平面，从而将不同的数据分隔开。
   扩展：这里有篇文章详尽介绍了SVM的原理、推导，《

支持向量机通俗导论（理解SVM的三层境界）》。此外，这里有个视频也是关于SVM的推导：《纯白板手推SVM》请简要介绍下tensorflow的计算图，深度学习 DL框架 中
@寒小阳&AntZ：Tensorflow是一个通过计算图的形式来表述计算的编程系统，计算图也叫数据流图，可以把计算图看做是一种有向图，Tensorflow中的每一个节点都是计算图上的一个Tensor, 也就是张量，而节点之间的边描述了计算之间的依赖关系(定义时)和数学操作(运算时)。如下两图表示：
[plain] view plain copy print?

1. a=x*y; b=a+z; c=tf.reduce_sum(b);



在k-means或kNN，我们常用欧氏距离来计算最近的邻居之间的距离，有时也用曼哈顿距离，请对比下这两种距离的差别。机器学习 ML模型 中

欧氏距离，最常见的两点之间或多点之间的距离表示法，又称之为欧几里得度量，它定义于欧几里得空间中，如点 x = (x1,...,xn) 和 y = (y1,...,yn) 之间的距离为：

$$d(x,y) := \sqrt{(x_1-y_1)^2+(x_2-y_2)^2+\cdots+(x_n-y_n)^2} = \sqrt{\sum_{i=1}^{n}(x_i-y_i)^2}$$

欧氏距离虽然很有用，但也有明显的缺点。它将样品的不同属性（即各指标或各变量量纲）之间的差别等同看待，这一点有时不能满足实际要求。例如，在教育研究中，经常遇到对人的分析和判别，个体的不同属性对于区分个体有着不同的重要性。因此，欧氏距离适用于向量各分量的度量标准统一的情况。

> 曼哈顿距离，我们可以定义曼哈顿距离的正式意义为L1-距离或城市区块距离，也就是在欧几里得空间的固定直角坐标系上两点所形成的线段对轴产生的投影的距离总和。例如在平面上，坐标（x1, y1）的点P1与坐标（x2, y2）的点P2的曼哈顿距离为：$|x_1-x_2|+|y_1-y_2|$，要注意的是，曼哈顿距离依赖座标系统的转度，而非系统在座标轴上的平移或映射。当坐标轴变动时，点间的距离就会不同。

通俗来讲，想象你在曼哈顿要从一个十字路口开车到另外一个十字路口，驾驶距离是两点间的直线距离吗？显然不是，除非你能穿越大楼。而实际驾驶距离就是这个“曼哈顿距离”，这也是曼哈顿距离名称的来源， 同时，曼哈顿距离也称为城市街区距离(City Block distance)。

曼哈顿距离和欧式距离一般用途不同，无相互替代性。另，关于各种距离的比较参看《从K近邻算法、距离度量谈到KD树、SIFT+BBF算法》。CNN的卷积核是单层的还是多层的？深度学习 DL模型 中

@AntZ：卷积运算的定义和理解可以看下这篇文章《CNN笔记：通俗理解卷积神经网络》，链接：http://blog.csdn.net/v_july_v/article/details/51812459，在CNN中,卷积计算属于离散卷积, 本来需要卷积核的权重矩阵旋转180度, 但我们并不需要旋转前的权重矩阵形式, 故直接用旋转后权重矩阵作为卷积核表达, 这样的好处就离散卷积运算变成了矩阵点积运算。

一般而言，深度卷积网络是一层又一层的。层的本质是特征图, 存贮输入数据或其中间表示值。一组卷积核则是联系前后两层的网络参数表达体, 训练的目标就是每个卷积核的权重参数组。

描述网络模型中某层的厚度，通常用名词通道channel数或者特征图feature map数。不过人们更习惯把作为数据输入的前层的厚度称之为通道数（比如RGB三色图层称为输入通道数为3），把作为卷积输出的后层的厚度称之为特征图数。

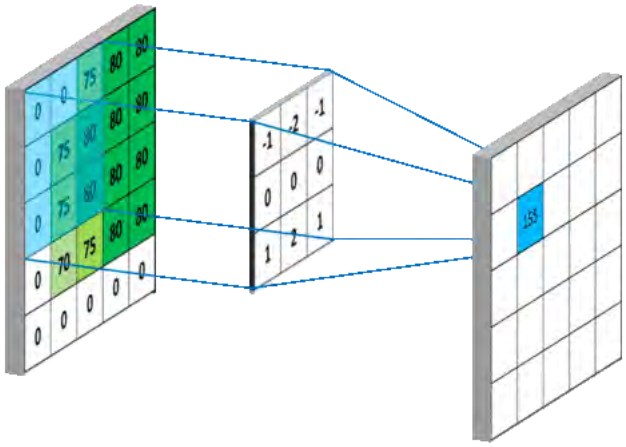
卷积核(filter)一般是3D多层的，除了面积参数, 比如3x3之外, 还有厚度参数H（2D的视为厚度1). 还有一个属性是卷积核的个数N。

卷积核的厚度H, 一般等于前层厚度M(输入通道数或feature map数). 特殊情况M > H。

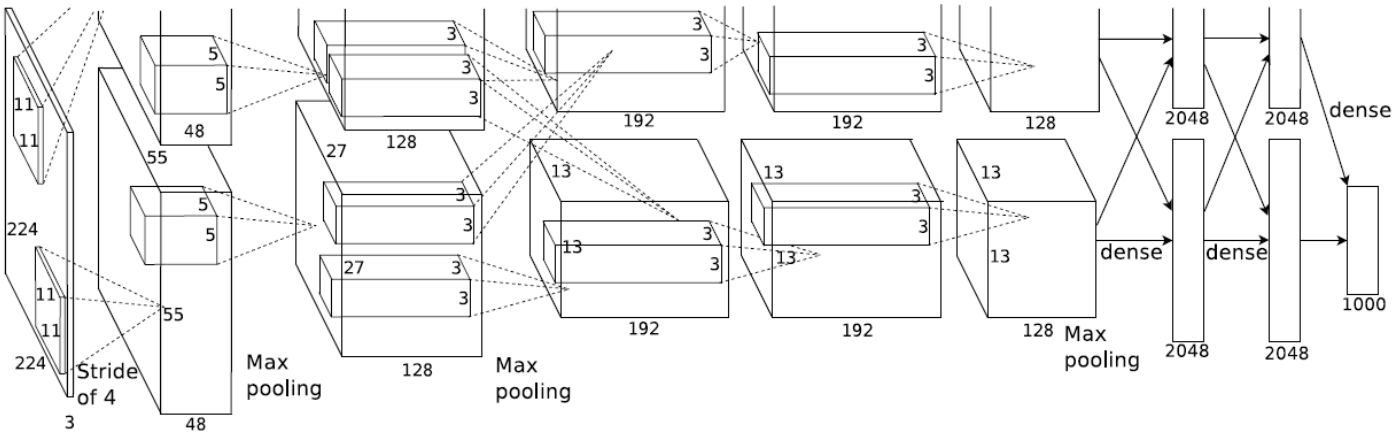
卷积核的个数N, 一般等于后层厚度(后层feature maps数，因为相等所以也用N表示)。

卷积核通常从属于后层，为后层提供了各种查看前层特征的视角，这个视角是自动形成的。

卷积核厚度等于1时为2D卷积，也就是平面对应点分别相乘然后把结果加起来，相当于点积运算. 各种2D卷积动图可以看这里 https://github.com/vdumoulin/conv_arithmetic



卷积核厚度大于1时为3D卷积(depth-wise)，每片平面分别求2D卷积，然后把每片卷积结果加起来，作为3D卷积结果；1x1卷积属于3D卷积的一个特例(point-wise)，有厚度无面积, 直接把每层单个点相乘再相加。
归纳之，卷积的意思就是把一个区域，不管是一维线段，二维方阵，还是三维长方块，全部按照卷积核的维度形状，从输入挖出同样维度形状, 对应逐点相乘后求和，浓缩成一个标量值也就是降到零维度，作为输出到一个特征图的一个点的值. 这个很像渔夫收网。
可以比喻一群渔夫坐一个渔船撒网打鱼，鱼塘是多层水域，每层鱼儿不同。
船每次移位一个stride到一个地方，每个渔夫撒一网，得到收获，然后换一个距离stride再撒，如此重复直到遍历鱼塘。
A渔夫盯着鱼的品种，遍历鱼塘后该渔夫描绘了鱼塘的鱼品种分布；
B渔夫盯着鱼的重量，遍历鱼塘后该渔夫描绘了鱼塘的鱼重量分布；
还有N-2个渔夫，各自兴趣各干各的；
最后得到N个特征图，描述了鱼塘的一切！
2D卷积表示渔夫的网就是带一圈浮标的渔网，只打上面一层水体的鱼；
3D卷积表示渔夫的网是多层嵌套的渔网，上中下层水体的鱼儿都跑不掉；
1x1卷积可以视为每次移位stride，甩钩钓鱼代替了撒网；
下面解释一下特殊情况的 M > H：
实际上，除了输入数据的通道数比较少之外，中间层的feature map数很多，这样中间层算卷积会累死计算机（鱼塘太深，每层鱼都打，需要的鱼网太重了）。所以很多深度卷积网络把全部通道/特征图划分一下，每个卷积核只看其中一部分（渔夫A的渔网只打捞深水段，渔夫B的渔网只打捞浅水段）。这样整个深度网络架构是横向开始分道扬镳了，到最后才又融合。这样看来，很多网络模型的架构不完全是突发奇想，而是是被参数计算量逼得。特别是现在需要在移动设备上进行AI应用计算(也叫推断), 模型参数规模必须更小, 所以出现很多减少握手规模的卷积形式, 现在主流网络架构大都如此。比如AlexNet：



另，附百度2015校招机器学习笔试题：http://www.itmian4.com/thread-7042-1-1.html
关于LR。机器学习 ML模型 难@rickjin：把LR从头到脚都给讲一遍。建模，现场数学推导，每种解法的原理，正则化，LR和maxent模型啥关系，lr为啥比线性回归好。有不少会背答案的人，问逻辑细节就糊涂了。原理都会？那就问工程，并行化怎么做，有几种并行化方式，读过哪些开源的实现。还会，那就准备收了吧，顺便逼问LR模型发展历史。

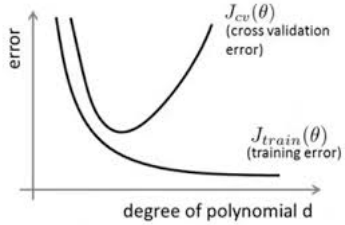
$$\begin{aligned}\ell(\theta) &= \log L(\theta) \\ &= \sum_{i=1}^{m} y^{(i)} \log h(x^{(i)}) + (1-y^{(i)}) \log(1-h(x^{(i)}))\end{aligned}$$

$$\begin{aligned}\frac{\partial}{\partial\theta_j}\ell(\theta) &= \left(y\frac{1}{g(\theta^T x)} - (1-y)\frac{1}{1-g(\theta^T x)}\right)\frac{\partial}{\partial\theta_j} g(\theta^T x) \\ &= \left(y\frac{1}{g(\theta^T x)} - (1-y)\frac{1}{1-g(\theta^T x)}\right) g(\theta^T x)(1-g(\theta^T x))\frac{\partial}{\partial\theta_j}\theta^T x \\ &= \left(y(1-g(\theta^T x)) - (1-y)g(\theta^T x)\right) x_j \\ &= \left(y - h_\theta(x)\right) x_j\end{aligned}$$

Logistic回归参数迭代公式：

$$\theta_j := \theta_j + \alpha\left(y^{(i)} - h_\theta(x^{(i)})\right) x_j^{(i)}$$

另外，这两篇文章可以做下参考：Logistic Regression 的前世今生（理论篇）、机器学习算法与Python实践之（七）逻辑回归（Logistic Regression）。overfitting怎么解决？机器学习 ML基础 中
dropout、regularization、batch normalizatin
@AntZ: overfitting就是过拟合, 其直观的表现如下图所示，随着训练过程的进行，模型复杂度增加，在training data上的error渐渐减小，但是在验证集上的error却反而渐渐增大——因为训练出来的网络过拟合了训练集, 对训练集外的数据却不work, 这称之为泛化(generalization)性能不好。泛化性能是训练的效果评价中的首要目标，没有良好的泛化，就等于南辕北辙, 一切都是无用功。



过拟合是泛化的反面，好比乡下快活的刘姥姥进了大观园会各种不适应，但受过良好教育的林黛玉进贾府就不会大惊小怪。实际训练中, 降低过拟合的办法一般如下：
正则化(Regularization)
L2正则化：目标函数中增加所有权重w参数的平方之和, 逼迫所有w尽可能趋向零但不为零. 因为过拟合的时候, 拟合函数需要顾忌每一个点, 最终形成的拟合函数波动很大, 在某些很小的区间里, 函数值的变化很剧烈, 也就是某些w非常大. 为此, L2正则化的加入就惩罚了权重变大的趋势.
L1正则化：目标函数中增加所有权重w参数的绝对值之和, 逼迫更多w为零(也就是变稀疏. L2因为其导数也趋0, 奔向零的速度不如L1给力了). 大家对稀疏规则化趋之若鹜的一个关键原因在于它能实现特征的自动选择。一般来说，xi的大部分元素（也就是特征）都是和最终的输出yi没有关系或者不提供任何信息的，在最小化目标函数的时候考虑xi这些额外的特征，虽然可以获得更小的训练误差，但在预测新的样本时，这些没用的特征权重反而会被考虑，从而干扰了对正确yi的预测。稀疏规则化算子的引入就是为了完成特征自动选择的光荣使命，它会学习地去掉这些无用的特征，也就是把这些特征对应的权重置为0。
随机失活(dropout)
在训练的运行的时候，让神经元以超参数p的概率被激活(也就是1-p的概率被设置为0), 每个w因此随机参与, 使得任意w都不是不可或缺的, 效果类似于数量巨大的模型集成。
逐层归一化(batch normalization)
这个方法给每层的输出都做一次归一化(网络上相当于加了一个线性变换层), 使得下一层的输入接近高斯分布. 这个方法相当于下一层的w训练时避免了其输入以偏概全, 因而泛化效果非常好.
提前终止(early stopping)
理论上可能的局部极小值数量随参数的数量呈指数增长, 到达某个精确的最小值是不良泛化的一个来源. 实践表明, 追求细粒度极小值具有较高的泛化误差。这是直观的，因为我们通常会希望我们的误差函数是平滑的, 精确的最小值处所见相应误差曲面具有高度不规则性, 而我们的泛化要求减少精确度去获得平滑最小值, 所以很多训练方法都提出了提前终止策略. 典型的方法是根据交叉验证提前终止: 若每次训练前, 将训练数据划分为若干份, 取一份为测试集, 其他为训练集, 每次训练完立即拿此次选中的测试集自测. 因为每份都有一次机会当测试集, 所以此方法称之为交叉验证. 交叉验证的错误率最小时可以认为泛化性能最好, 这时候训练错误率虽然还在继续下降, 但也得终止继续训练了.
LR和SVM的联系与区别。机器学习 ML模型 中
@朝阳在望，联系：
1、LR和SVM都可以处理分类问题，且一般都用于处理线性二分类问题（在改进的情况下可以处理多分类问题）
2、两个方法都可以增加不同的正则化项，如l1、l2等等。所以在很多实验中，两种算法的结果是很接近的。
区别：
1、LR是参数模型，SVM是非参数模型。

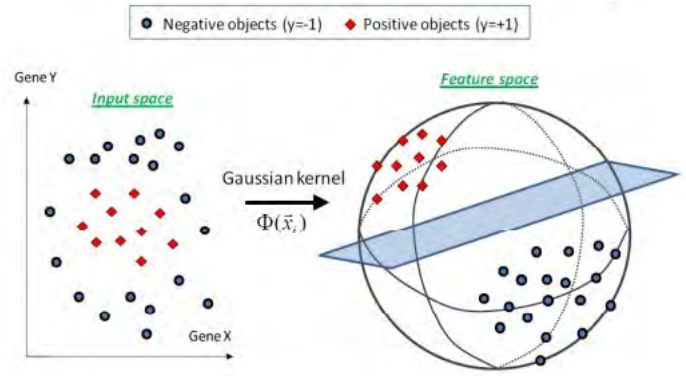
2、从目标函数来看，区别在于逻辑回归采用的是logistical loss，SVM采用的是hinge loss，这两个损失函数的目的都是增加对分类影响较大的数据点的权重，减少与分类关系较小的数据点的权重。
3、SVM的处理方法是只考虑support vectors，也就是和分类最相关的少数点，去学习分类器。而逻辑回归通过非线性映射，大大减小了离分类平面较远的点的权重，相对提升了与分类最相关的数据点的权重。
4、逻辑回归相对来说模型更简单，好理解，特别是大规模线性分类时比较方便。而SVM的理解和优化相对来说复杂一些，SVM转化为对偶问题后,分类只需要计算与少数几个支持向量的距离,这个在进行复杂核函数计算时优势很明显,能够大大简化模型和计算。
5、logic 能做的 svm能做，但可能在准确率上有问题，svm能做的logic有的做不了。
来源：http://blog.csdn.net/timcompp/article/details/62237986说说你知道的核函数。机器学习 ML基础 易

通常人们会从一些常用的核函数中选择（根据问题和数据的不同，选择不同的参数，实际上就是得到了不同的核函数），例如：

多项式核，显然刚才我们举的例子是这里多项式核的一个特例（R = 1，d = 2）。虽然比较麻烦，而且没有必要，不过这个核所对应的映射实际上是可以写出来的，该空间的维度是$\binom{m+d}{d}$，其中 是原始空间的维度。

$$\kappa(x_1, x_2) = \exp\left(-\frac{\|x_1 - x_2\|^2}{2\sigma^2}\right)$$

高斯核，这个核就是最开始提到过的会将原始空间映射为无穷维空间的那个家伙。不过，如果σ选得很大的话，高次特征上的权重实际上衰减得非常快，所以实际上（数值上近似一下）相当于一个低维的子空间；反过来，如果σ选得很小，则可以将任意的数据映射为线性可分——当然，这并不一定是好事，因为随之而来的可能是非常严重的过拟合问题。不过，总的来说，通过调控参数σ，高斯核实际上具有相当高的灵活性，也是使用最广泛的核函数之一。下图所示的例子便是把低维线性不可分的数据通过高斯核函数映射到了高维空间：

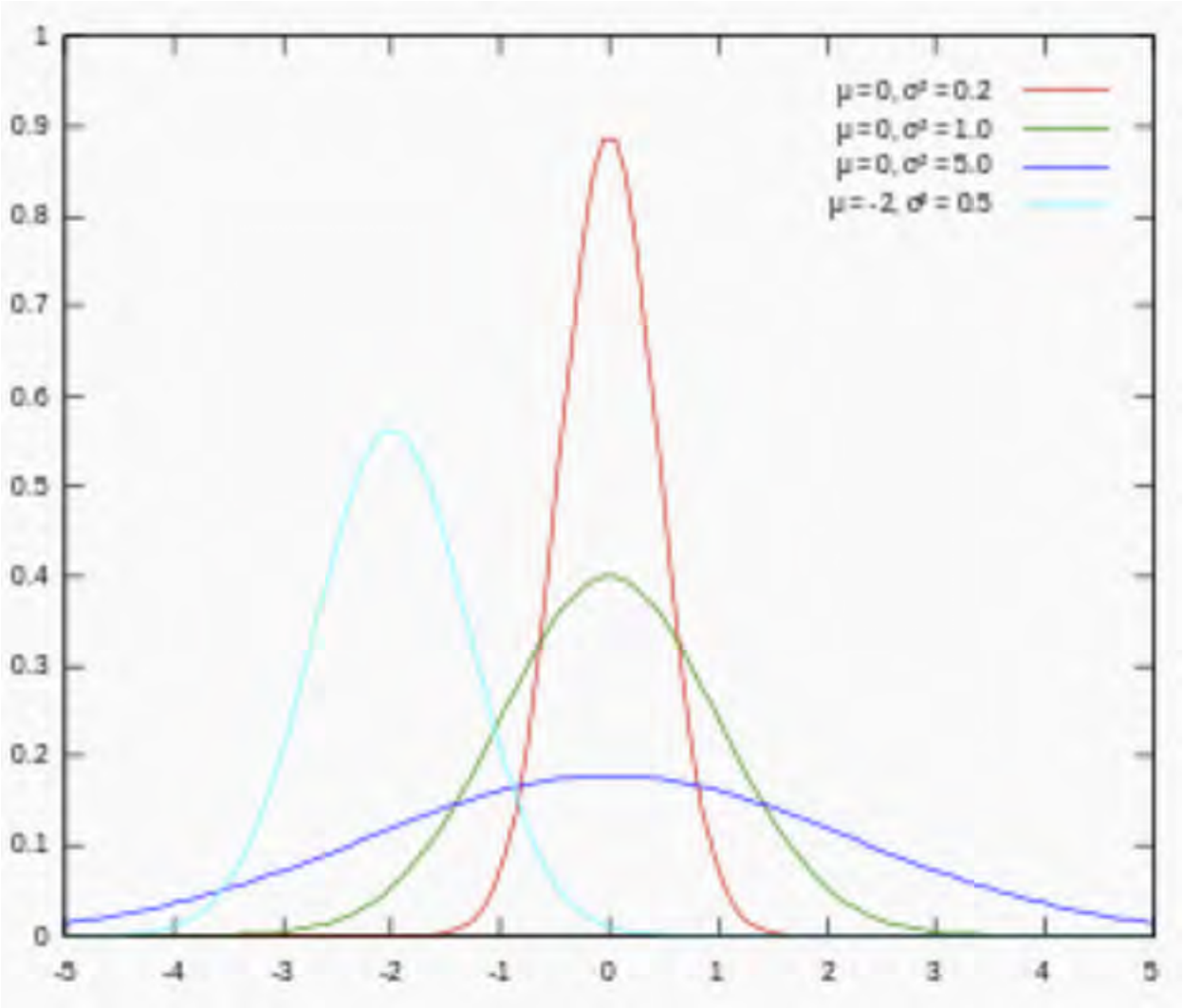


线性核$\kappa(x_1, x_2) = \langle x_1, x_2 \rangle$，这实际上就是原始空间中的内积。这个核存在的主要目的是使得“映射后空间中的问题”和“映射前空间中的问题”两者在形式上统一起来了(意思是说，咱们有的时候，写代码，或写公式的时候，只要写个模板或通用表达式，然后再代入不同的核，便可以了，于此，便在形式上统一了起来，不用再分别写一个线性的，和一个非线性的)。

LR与线性回归的区别与联系。机器学习　　ML模型 中等
@AntZ: LR工业上一般指Logistic Regression(逻辑回归)而不是Linear Regression(线性回归). LR在线性回归的实数范围输出值上施加sigmoid函数将值收敛到0~1范围, 其目标函数也因此从差平方和函数变为对数损失函数, 以提供最优化所需导数（sigmoid函数是softmax函数的二元特例, 其导数均为函数值的f*(1-f)形式）。请注意, LR往往是解决二元0/1分类问题的, 只是它和线性回归耦合太紧, 不自觉也冠了个回归的名字(马甲无处不在). 若要求多元分类,就要把sigmoid换成大名鼎鼎的softmax了。
@nishizhen：个人感觉逻辑回归和线性回归首先都是广义的线性回归，
其次经典线性模型的优化目标函数是最小二乘，而逻辑回归则是似然函数，
另外线性回归在整个实数域范围内进行预测，敏感度一致，而分类范围，需要在[0,1]。逻辑回归就是一种减小预测范围，将预测值限定为[0,1]间的一种回归模型，因而对于这类问题来说，逻辑回归的鲁棒性比线性回归的要好。
@乖乖癞皮狗：逻辑回归的模型本质上是一个线性回归模型，逻辑回归都是以线性回归为理论支持的。但线性回归模型无法做到sigmoid的非线性形式，sigmoid可以轻松处理0/1分类问题。
请问（决策树、Random Forest、Booting、Adaboot）GBDT和XGBoost的区别是什么？机器学习 ML模型 难
@AntZ
集成学习的集成对象是学习器. Bagging和Boosting属于集成学习的两类方法. Bagging方法有放回地采样同数量样本训练每个学习器, 然后再一起集成(简单投票); Boosting方法使用全部样本(可调权重)依次训练每个学习器, 迭代集成(平滑加权).
决策树属于最常用的学习器, 其学习过程是从根建立树, 也就是如何决策叶子节点分裂. ID3/C4.5决策树用信息熵计算最优分裂, CART决策树用基尼指数计算最优分裂, xgboost决策树使用二阶泰勒展开系数计算最优分裂.
下面所提到的学习器都是决策树:
Bagging方法:
　学习器间不存在强依赖关系, 学习器可并行训练生成, 集成方式一般为投票;
　Random Forest属于Bagging的代表, 放回抽样, 每个学习器随机选择部分特征去优化;
Boosting方法:
　学习器之间存在强依赖关系、必须串行生成, 集成方式为加权和;
　Adaboost属于Boosting, 采用指数损失函数替代原本分类任务的0/1损失函数;
　GBDT属于Boosting的优秀代表, 对函数残差近似值进行梯度下降, 用CART回归树做学习器, 集成为回归模型;
　xgboost属于Boosting的集大成者, 对函数残差近似值进行梯度下降, 迭代时利用了二阶梯度信息, 集成模型可分类也可回归. 由于它可在特征粒度上并行计算, 结构风险和工程实现都做了很多优化, 泛化, 性能和扩展性都比GBDT要好。
关于决策树，这里有篇《决策树算法》。而随机森林Random Forest是一个包含多个决策树的分类器。至于AdaBoost，则是英文"Adaptive Boosting"（自适应增强）的缩写，关于AdaBoost可以看下这篇文章《Adaboost 算法的原理与推导》。GBDT（Gradient Boosting Decision Tree），即梯度上升决策树算法，相当于融合决策树和梯度上升boosting算法。
@Xijun LI：xgboost类似于gbdt的优化版，不论是精度还是效率上都有了提升。与gbdt相比，具体的优点有：
1.损失函数是用泰勒展式二项逼近，而不是像gbdt里的就是一阶导数
2.对树的结构进行了正则化约束，防止模型过度复杂，降低了过拟合的可能性
3.节点分裂的方式不同，gbdt是用的gini系数，xgboost是经过优化推导后的
更多详见：https://xijunlee.github.io/2017/06/03/%E9%9B%86%E6%88%90%E5%AD%A6%E4%B9%A0%E6%80%BB%E7%BB%93/为什么xgboost要用泰勒展开，优势在哪里？机器学习 ML模型 难
@AntZ：xgboost使用了一阶和二阶偏导, 二阶导数有利于梯度下降的更快更准. 使用泰勒展开取得函数做自变量的二阶导数形式, 可以在不选定损失函数具体形式的情况下, 仅仅依靠输入数据的值就可以进行叶子分裂优化计算, 本质上也就把损失函数的选取和模型算法优化/参数选择分开了. 这种去耦合增加了xgboost的适用性, 使得它按需选取损失函数, 可以用于分类, 也可以用于回归。

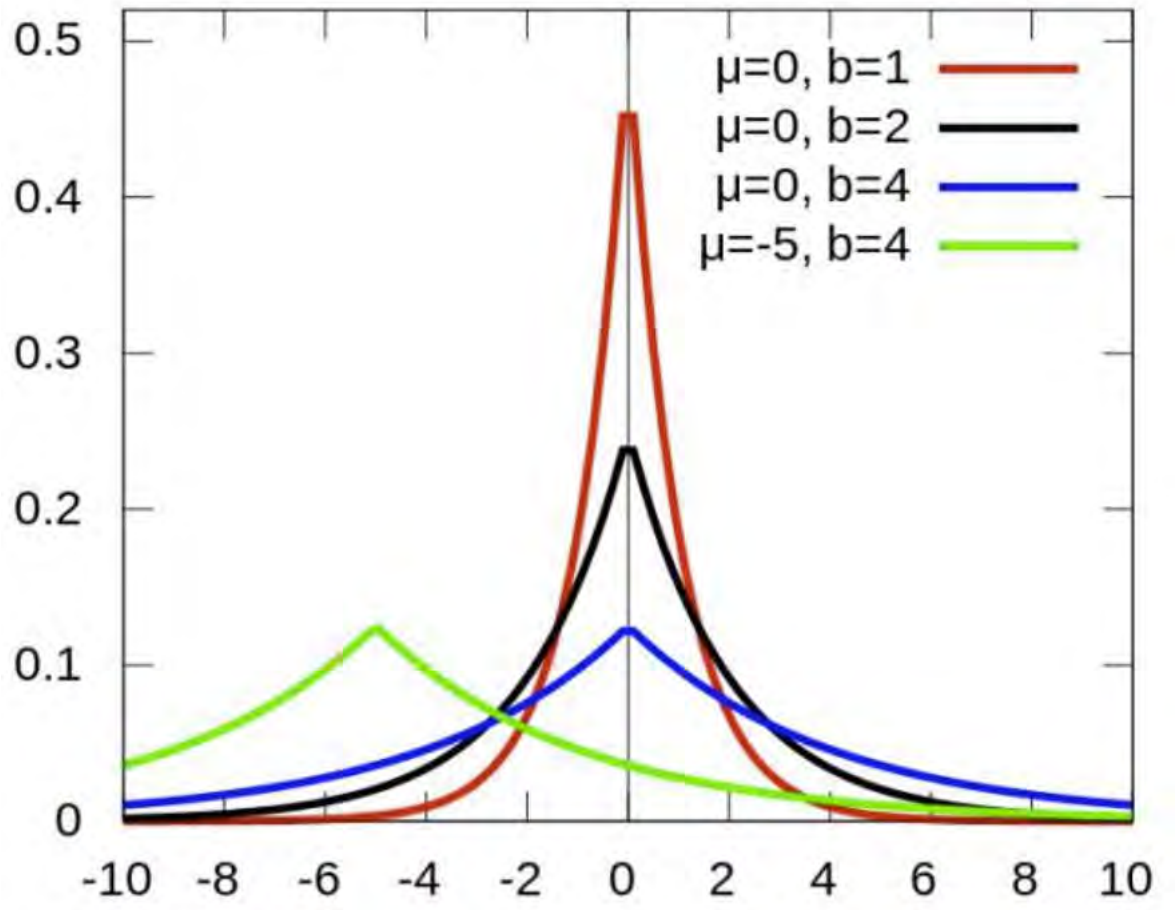
xgboost如何寻找最优特征？是又放回还是无放回的呢？机器学习 ML模型 难
@AntZ：xgboost在训练的过程中给出各个特征的增益评分，最大增益的特征会被选出来作为分裂依据, 从而记忆了每个特征对在模型训练时的重要性 -- 从根到叶子中间节点涉及某特征的次数作为该特征重要性排序.
xgboost属于boosting集成学习方法, 样本是不放回的, 因而每轮计算样本不重复. 另一方面, xgboost支持子采样, 也就是每轮计算可以不使用全部样本, 以减少过拟合. 进一步地, xgboost 还有列采样, 每轮计算按百分比随机采样一部分特征, 既提高计算速度又减少过拟合。
谈谈判别式模型和生成式模型？机器学习 ML基础 易
判别方法：由数据直接学习决策函数 Y = f（X），或者由条件分布概率 P（Y|X）作为预测模型，即判别模型。
生成方法：由数据学习联合概率密度分布函数 P（X,Y）,然后求出条件概率分布P(Y|X)作为预测的模型，即生成模型。
由生成模型可以得到判别模型，但由判别模型得不到生成模型。
常见的判别模型有：K近邻、SVM、决策树、感知机、线性判别分析（LDA）、线性回归、传统的神经网络、逻辑斯蒂回归、boosting、条件随机场
常见的生成模型有：朴素贝叶斯、隐马尔可夫模型、高斯混合模型、文档主题生成模型（LDA）、限制玻尔兹曼机L1和L2的区别。机器学习 ML基础 易
L1范数（L1 norm）是指向量中各个元素绝对值之和，也有个美称叫“稀疏规则算子”（Lasso regularization）。
比如 向量A=[1，-1，3]， 那么A的L1范数为 |1|+|-1|+|3|.
简单总结一下就是：
L1范数: 为x向量各个元素绝对值之和。
L2范数: 为x向量各个元素平方和的1/2次方，L2范数又称Euclidean范数或者Frobenius范数
Lp范数: 为x向量各个元素绝对值p次方和的1/p次方.
在支持向量机学习过程中，L1范数实际是一种对于成本函数求解最优的过程，因此，L1范数正则化通过向成本函数中添加L1范数，使得学习得到的结果满足稀疏化，从而方便人类提取特征。
L1范数可以使权值稀疏，方便特征提取。
L2范数可以防止过拟合，提升模型的泛化能力。
@AntZ: L1和L2的差别，为什么一个让绝对值最小，一个让平方最小，会有那么大的差别呢？看导数一个是1一个是w便知, 在靠进零附近, L1以匀速下降到零, 而L2则完全停下来了. 这说明L1是将不重要的特征(或者说, 重要性不在一个数量级上)尽快剔除, L2则是把特征贡献尽量压缩最小但不至于为零. 两者一起作用, 就是把重要性在一个数量级(重要性最高的)的那些特征一起平等共事(简言之, 不养闲人也不要超人)。
L1和L2正则先验分别服从什么分布。机器学习 ML基础 易
@齐同学：面试中遇到的，L1和L2正则先验分别服从什么分布，L1是拉普拉斯分布，L2是高斯分布。
@AntZ: 先验就是优化的起跑线, 有先验的好处就是可以在较小的数据集中有良好的泛化性能，当然这是在先验分布是接近真实分布的情况下得到的了，从信息论的角度看，向系统加入了正确先验这个信息，肯定会提高系统的性能。
对参数引入高斯正态先验分布相当于L2正则化, 这个大家都熟悉：

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

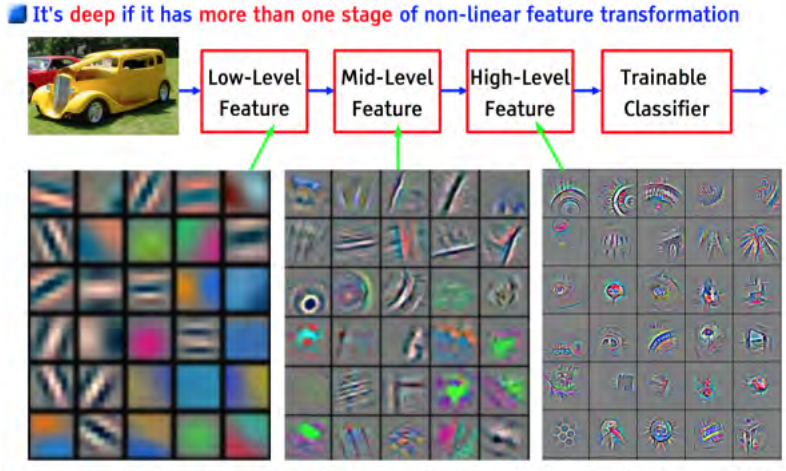


对参数引入拉普拉斯先验等价于 L1正则化, 如下图：

$$f(x \mid \mu, b) = \frac{1}{2b}\exp\left(-\frac{|x-\mu|}{b}\right)$$

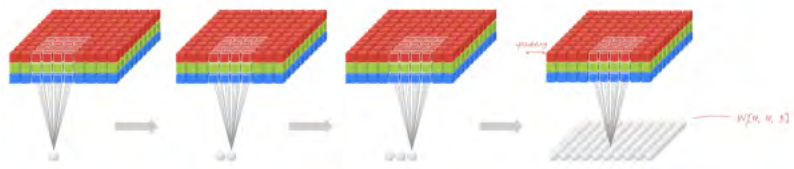


从上面两图可以看出, L2先验趋向零周围, L1先验趋向零本身。
CNN最成功的应用是在CV，那为什么NLP和Speech的很多问题也可以用CNN解出来？为什么AlphaGo里也用了CNN？这几个不相关的问题的相似性在哪里？CNN通过什么手段抓住了这个共性？深度学习 DL应用 难
@许韩，来源：https://zhuanlan.zhihu.com/p/25005808
Deep Learning -Yann LeCun, Yoshua Bengio & Geoffrey Hinton
Learn TensorFlow and deep learning, without a Ph.D.
The Unreasonable Effectiveness of Deep Learning -LeCun 16 NIPS Keynote
以上几个不相关问题的相关性在于，都存在局部与整体的关系，由低层次的特征经过组合，组成高层次的特征，并且得到不同特征之间的空间相关性。
如下图：低层次的直线 / 曲线等特征，组合成为不同的形状，最后得到汽车的表示。



Feature visualization of convolutional net trained on ImageNet from [Zeiler & Fergus 2013]

CNN抓住此共性的手段主要有四个：局部连接 / 权值共享 / 池化操作 / 多层次结构。
局部连接使网络可以提取数据的局部特征；权值共享大大降低了网络的训练难度，一个Filter只提取一个特征，在整个图片（或者语音 / 文本） 中进行卷积；池化操作与多层次结构一起，实现了数据的降维，将低层次的局部特征组合成为较高层次的特征，从而对整个图片进行表示。如下图：



上图中，如果每一个点的处理使用相同的Filter，则为全卷积，如果使用不同的Filter，则为Local-Conv。
另，关于CNN，这里有篇文章《CNN笔记：通俗理解卷积神经网络》。
说一下Adaboost，权值更新公式。当弱分类器是Gm时，每个样本的的权重是w1，w2...，请写出最终的决策公式。机器学习 ML模型 难

给定一个训练数据集T={(x1,y1), (x2,y2)...(xN,yN)}，其中实例$x \in \mathcal{X}$，而实例空间$\mathcal{X} \subset \mathbb{R}^n$，yi属于标记集合{-1,+1}，Adaboost的目的就是从训练数据中学习一系列弱分类器或基本分类器，然后将这些弱分类器组合成一个强分类器。

Adaboost的算法流程如下：

步骤1. 首先，初始化训练数据的权值分布。每一个训练样本最开始时都被赋予相同的权值：1/N。

$$D_1 = (w_{11}, w_{12} \cdots w_{1i} \cdots, w_{1N}), \quad w_{1i} = \frac{1}{N}, \quad i = 1,2,\cdots,N$$

步骤2. 进行多轮迭代，用m = 1,2, ..., M表示迭代的第多少轮

a. 使用具有权值分布Dm的训练数据集学习，得到基本分类器（选取让误差率最低的阈值来设计基本分类器）：

$$G_m(x): \ \chi \to \{-1,+1\}$$

b. 计算Gm(x)在训练数据集上的分类误差率

$$e_m = P(G_m(x_i) \neq y_i) = \sum_{i=1}^{N} w_{mi} I(G_m(x_i) \neq y_i)$$

由上述式子可知，Gm(x)在训练数据集上的误差率em就是被Gm(x)误分类样本的权值之和。

c. 计算Gm(x)的系数，am表示Gm(x)在最终分类器中的重要程度（目的：得到基本分类器在最终分类器中所占的权重）：

$$\alpha_m = \frac{1}{2} \log \frac{1 - e_m}{e_m}$$

由上述式子可知，em <= 1/2时，am >= 0，且am随着em的减小而增大，意味着分类误差率越小的基本分类器在最终分类器中的作用越大。

d. 更新训练数据集的权值分布（目的：得到样本的新的权值分布），用于下一轮迭代

$$D_{m+1} = (w_{m+1,1}, w_{m+1,2} \cdots w_{m+1,i} \cdots, w_{m+1,N})$$

$$w_{m+1,i} = \frac{w_{mi}}{Z_m} \exp(-\alpha_m y_i G_m(x_i)), \quad i = 1,2,\cdots,N$$

使得被基本分类器Gm(x)误分类样本的权值增大，而被正确分类样本的权值减小。就这样，通过这样的方式，AdaBoost方法能“重点关注”或“聚焦于”那些较难分的样本上。

其中，Zm是规范化因子，使得Dm+1成为一个概率分布：

$$Z_m = \sum_{i=1}^{N} w_{mi} \exp(-\alpha_m y_i G_m(x_i))$$

步骤3. 组合各个弱分类器

$$f(x) = \sum_{m=1}^{M} \alpha_m G_m(x)$$

从而得到最终分类器，如下：

$$G(x) = sign(f(x)) = sign\left(\sum_{m=1}^{M} \alpha_m G_m(x)\right)$$

更多请查看此文：《Adaboost 算法的原理与推导》。

LSTM结构推导，为什么比RNN好？深度学习 DL模型 难
推导forget gate，input gate，cell state， hidden information等的变化；因为LSTM有进有出且当前的cell informaton是通过input gate控制之后叠加的，RNN是叠乘，因此LSTM可以防止梯度消失或者爆炸

经常在网上搜索东西的朋友知道，当你不小心输入一个不存在的单词时，搜索引擎会提示你是不是要输入某一个正确的单词，比如当你在Google中输入“Julw”时，系统会猜测你的意图：是不是要搜索“July”，如下图所示：

这叫做拼写检查。根据谷歌一员工写的文章显示，Google的拼写检查基于贝叶斯方法。请说说的你的理解，具体Google是怎么利用贝叶斯方法，实现"拼写检查"的功能。 机器学习 ML应用 难

用户输入一个单词时，可能拼写正确，也可能拼写错误。如果把拼写正确的情况记做c（代表correct），拼写错误的情况记做w（代表wrong），那么"拼写检查"要做的事情就是：在发生w的情况下，试图推断出c。换言之：已知w，然后在若干个备选方案中，找出可能性最大的那个c，也就是求$P(c|w)$的最大值。

而根据贝叶斯定理，有：

$$P(c|w) = P(w|c) * P(c)/P(w)$$

由于对于所有备选的c来说，对应的都是同一个w，所以它们的P(w)是相同的，因此我们只要最大化

$$P(w|c) * P(c)$$

即可。其中：

P(c)表示某个正确的词的出现"概率"，它可以用"频率"代替。如果我们有一个足够大的文本库，那么这个文本库中每个单词的出现频率，就相当于它的发生概率。某个词的出现频率越高，P(c)就越大。比如在你输入一个错误的词"Julw"时，系统更倾向于去猜测你可能想输入的词是"July"，而不是"Jult"，因为"July"更常见。
P(w|c)表示在试图拼写c的情况下，出现拼写错误w的概率。为了简化问题，假定两个单词在字形上越接近，就有越可能拼错，P(w|c)就越大。举例来说，相差一个字母的拼法，就比相差两个字母的拼法，发生概率更高。你想拼写单词July，那么错误拼成Julw（相差一个字母）的可能性，就比拼成Jullw高（相差两个字母）。值得一提的是，一般把这种问题称为"编辑距离"，参见博客中的这篇文章。

所以，我们比较所有拼写相近的词在文本库中的出现频率，再从中挑出出现频率最高的一个，即是用户最想输入的那个词。具体的计算过程及此方法的缺陷请参见这里。

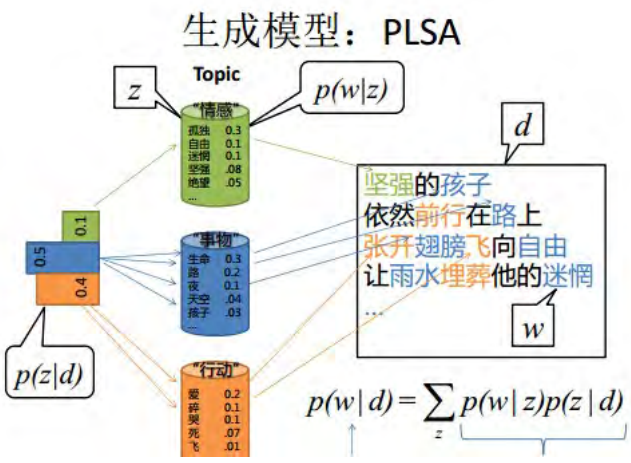
为什么朴素贝叶斯如此“朴素”？ 机器学习 ML模型 易
因为它假定所有的特征在数据集中的作用是同样重要和独立的。正如我们所知，这个假设在现实世界中是很不真实的，因此，说朴素贝叶斯真的很“朴素”。
@AntZ: 朴素贝叶斯模型(Naive Bayesian Model)的朴素(Naive)的含义是"很简单很天真"地假设样本特征彼此独立. 这个假设现实中基本上不存在, 但特征相关性很小的实际情况还是很多的, 所以这个模型仍然能够工作得很好。

请大致对比下plsa和LDA的区别。机器学习 ML模型 中等

pLSA中，主题分布和词分布确定后，以一定的概率（$P(z_k|d_i)$、$P(w_j|z_k)$）分别选取具体的主题和词项，生成好文档。而后根据生成好的文档反推其主题分布、词分布时，最终用EM算法（极大似然估计思想）求解出了两个未知但固定的参数的值：$\phi_{k,j}$（由$P(w_j|z_k)$转换而来）和$\theta_{i,k}$（由$P(z_k|d_i)$转换而来）。

文档d产生主题z的概率，主题z产生单词w的概率都是两个固定的值。

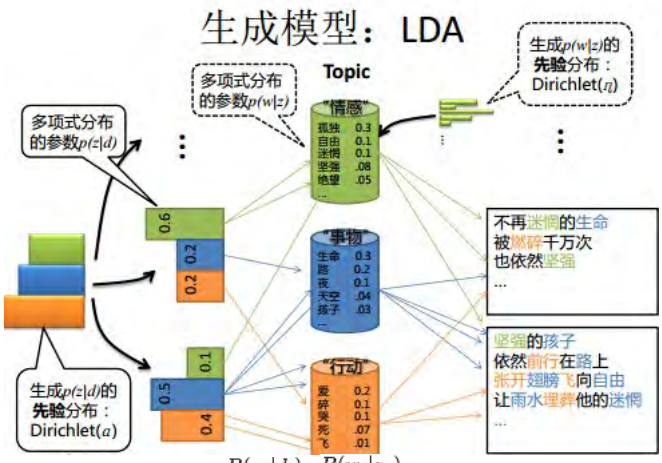
举个文档d产生主题z的例子。给定一篇文档d，主题分布是一定的，比如{ P(zi|d), i = 1,2,3 }可能就是{0.4,0.5,0.1}，表示z1、z2、z3，这3个主题被文档d选中的概率都是个固定的值：P(z1|d) = 0.4、P(z2|d) = 0.5、P(z3|d) = 0.1，如下图所示（图截取自沈博PPT上）：



但在贝叶斯框架下的LDA中，我们不再认为主题分布（各个主题在文档中出现的概率分布）和词分布（各个词语在某个主题下出现的概率分布）是唯一确定的（而是随机变量），而是有很多种可能。但一篇文档总得对应一个主题分布和一个词分布吧，怎么办呢？LDA为它们弄了两个Dirichlet先验参数，这个Dirichlet先验为某篇文档随机抽取出某个主题分布和词分布。

文档d产生主题z（准确的说，其实是Dirichlet先验为文档d生成主题分布Θ，然后根据主题分布Θ产生主题z）的概率，主题z产生单词w的概率都不再是某两个确定的值，而是随机变量。

还是再次举下文档d具体产生主题z的例子。给定一篇文档d，现在有多个主题z1、z2、z3，它们的主题分布{ P(zi|d), i = 1,2,3 }可能是{0.4,0.5,0.1}，也可能是{0.2,0.2,0.6}，即这些主题被d选中的概率都不再认为是确定的值，可能是P(z1|d) = 0.4、P(z2|d) = 0.5、P(z3|d) = 0.1，也有可能是P(z1|d) = 0.2、P(z2|d) = 0.2、P(z3|d) = 0.6等等，而主题分布到底是哪个取值集合我们不确定（为什么？这就是贝叶斯派的核心思想，把未知参数当作是随机变量，不再认为是某一个确定的值），但其先验分布是dirichlet 分布，所以可以从无穷多个主题分布中按照dirichlet 先验随机抽取出某个主题分布出来。如下图所示（图截取自沈博PPT上）：



换言之，LDA在pLSA的基础上给这两参数（$P(z_k|d_i)$、$P(w_j|z_k)$）加了两个先验分布的参数（贝叶斯化）：一个主题分布的先验分布Dirichlet分布α，和一个词语分布的先验分布Dirichlet分布β。

综上，LDA真的只是pLSA的贝叶斯版本，文档生成后，两者都要根据文档去推断其主题分布和词语分布，只是用的参数推断方法不同，在pLSA中用极大似然估计的思想去推断两未知的固定参数，而LDA则把这两参数弄成随机变量，且加入dirichlet先验。

更多请参见：《通俗理解LDA主题模型》。

请简要说说EM算法。机器学习 ML模型 中等

@tornadomeet，本题解析来源：http://www.cnblogs.com/tornadomeet/p/3395593.html

有时候因为样本的产生和隐含变量有关（隐含变量是不能观察的），而求模型的参数时一般采用最大似然估计，由于含有了隐含变量，所以对似然函数参数求导是求不出来的，这时可以采用EM算法来求模型的参数的（对应模型参数个数可能有多个），EM算法一般分为2步：

E步：选取一组参数，求出在该参数下隐含变量的条件概率值；

M步：结合E步求出的隐含变量条件概率，求出似然函数下界函数（本质上是某个期望函数）的最大值。

重复上面2步直至收敛。

公式如下所示：

$$\text{(E-step) For each } i, \text{ set}$$

$$Q_i(z^{(i)}) := p(z^{(i)}|x^{(i)};\theta).$$

$$\text{(M-step) Set}$$

$$\theta := \arg\max_\theta \sum_i \sum_{z^{(i)}} Q_i(z^{(i)}) \log \frac{p(x^{(i)}, z^{(i)};\theta)}{Q_i(z^{(i)})}.$$

M步公式中下界函数的推导过程：

$$\begin{aligned} \sum_i \log p(x^{(i)};\theta) &= \sum_i \log \sum_{z^{(i)}} p(x^{(i)}, z^{(i)};\theta) && (1) \\ &= \sum_i \log \sum_{z^{(i)}} Q_i(z^{(i)}) \frac{p(x^{(i)}, z^{(i)};\theta)}{Q_i(z^{(i)})} && (2) \\ &\geq \sum_i \sum_{z^{(i)}} Q_i(z^{(i)}) \log \frac{p(x^{(i)}, z^{(i)};\theta)}{Q_i(z^{(i)})} && (3) \end{aligned}$$

EM算法一个常见的例子就是GMM模型，每个样本都有可能由k个高斯产生，只不过由每个高斯产生的概率不同而已，因此每个样本都有对应的高斯分布（k个中的某一个），此时的隐含变量就是每个样本对应的某个高斯分布。

GMM的E步公式如下（计算每个样本对应每个高斯的概率）：

$$\text{(E-step) For each } i, j, \text{ set}$$

$$w_j^{(i)} := p(z^{(i)} = j|x^{(i)};\phi,\mu,\Sigma)$$

更具体的计算公式为：

$$p(z^{(i)} = j|x^{(i)};\phi,\mu,\Sigma) = \frac{p(x^{(i)}|z^{(i)} = j;\mu,\Sigma)p(z^{(i)} = j;\phi)}{\sum_{l=1}^k p(x^{(i)}|z^{(i)} = l;\mu,\Sigma)p(z^{(i)} = l;\phi)}$$

M步公式如下（计算每个高斯的比重，均值，方差这3个参数）：

(M-step) Update the parameters:

$$\phi_j := \frac{1}{m}\sum_{i=1}^{m} w_j^{(i)},$$
$$\mu_j := \frac{\sum_{i=1}^{m} w_j^{(i)} x^{(i)}}{\sum_{i=1}^{m} w_j^{(i)}},$$
$$\Sigma_j := \frac{\sum_{i=1}^{m} w_j^{(i)} (x^{(i)} - \mu_j)(x^{(i)} - \mu_j)^T}{\sum_{i=1}^{m} w_j^{(i)}}$$

KNN中的K如何选取的？机器学习 ML模型 易

关于什么是KNN，可以查看此文：《从K近邻算法、距离度量谈到KD树、SIFT+BBF算法》。KNN中的K值选取对K近邻算法的结果会产生重大影响。如李航博士的一书「统计学习方法」上所说：

1. 如果选择较小的K值，就相当于用较小的领域中的训练实例进行预测，“学习”近似误差会减小，只有与输入实例较近或相似的训练实例才会对预测结果起作用，与此同时带来的问题是“学习”的估计误差会增大，换句话说，K值的减小就意味着整体模型变得复杂，容易发生过拟合；
2. 如果选择较大的K值，就相当于用较大领域中的训练实例进行预测，其优点是可以减少学习的估计误差，但缺点是学习的近似误差会增大。这时候，与输入实例较远（不相似的）训练实例也会对预测器作用，使预测发生错误，且K值的增大就意味着整体的模型变得简单。
3. K=N，则完全不足取，因为此时无论输入实例是什么，都只是简单的预测它属于在训练实例中最多的累，模型过于简单，忽略了训练实例中大量有用信息。

在实际应用中，K值一般取一个比较小的数值，例如采用交叉验证法（简单来说，就是一部分样本做训练集，一部分做测试集）来选择最优的K值。

防止过拟合的方法。机器学习 ML基础 易

过拟合的原因是算法的学习能力过强；一些假设条件（如样本独立同分布）可能是不成立的；训练样本过少不能对整个空间进行分布估计。

处理方法：

- 早停止：如在训练中多次迭代后发现模型性能没有显著提高就停止训练
- 数据集扩增：原有数据增加、原有数据加随机噪声、重采样
- 正则化
- 交叉验证
- 特征选择/特征降维
- 创建一个验证集是最基本的防止过拟合的方法。我们最终训练得到的模型目标是要在验证集上面有好的表现，而不训练集。
- 正则化可以限制模型的复杂度。

机器学习中，为何要经常对数据做归一化。机器学习 ML基础 中等

@zhanlijun，本题解析来源：http://www.cnblogs.com/LBSer/p/4440590.html

机器学习模型被互联网行业广泛应用，如排序（参见：排序学习实践）、推荐、反作弊、定位（参见：基于朴素贝叶斯的定位算法）等。一般做机器学习应用的时候大部分时间是花费在特征处理上，其中很关键的一步就是对特征数据进行归一化，为什么要归一化呢？很多同学并未搞清楚，维基百科给出的解释：1）归一化后加快了梯度下降求最优解的速度；2）归一化有可能提高精度。下面再简单扩展解释下这两点。

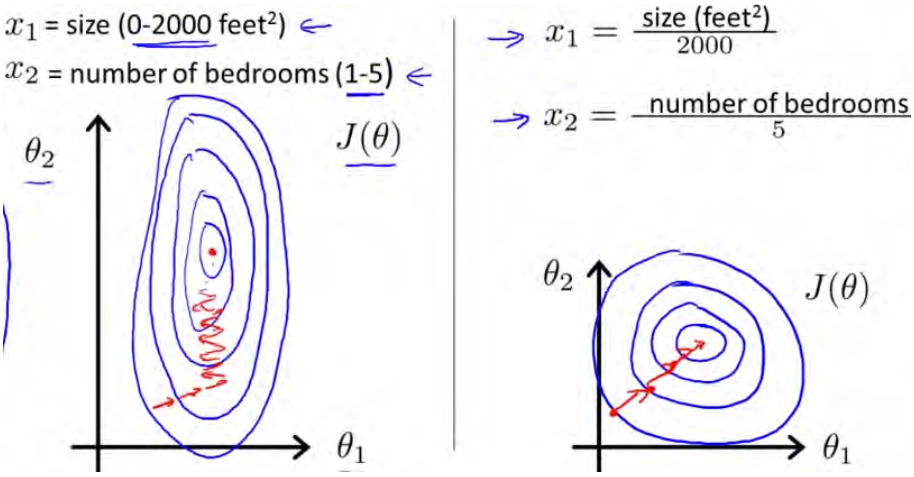
1 归一化为什么能提高梯度下降法求解最优解的速度？

斯坦福机器学习视频做了很好的解释：https://class.coursera.org/ml-003/lecture/21

如下图所示，蓝色的圈圈图代表的是两个特征的等高线。其中左图两个特征X1和X2的区间相差非常大，X1区间是[0,2000]，X2区间是[1,5]，其所形成的等高线非常尖。当使用梯度下降法寻求最优解时，很有可能走“之字型”路线（垂直等高线走），从而导致需要迭代很多次才能收敛；

而右图对两个原始特征进行了归一化，其对应的等高线显得很圆，在梯度下降进行求解时能较快的收敛。

因此如果机器学习模型使用梯度下降法求最优解时，归一化往往非常有必要，否则很难收敛甚至不能收敛。



2 归一化有可能提高精度

一些分类器需要计算样本之间的距离（如欧氏距离），例如KNN。如果一个特征值域范围非常大，那么距离计算就主要取决于这个特征，从而与实际情况相悖（比如这时实际情况是值域范围小的特征更重要）。

3 归一化的类型

1）线性归一化

这种归一化方法比较适用在数值比较集中的情况。这种方法有个缺陷，如果max和min不稳定，很容易使得归一化结果不稳定，使得后续使用效果也不稳定。实际使用中可以用经验常量值来替代max和min。

2）标准差标准化

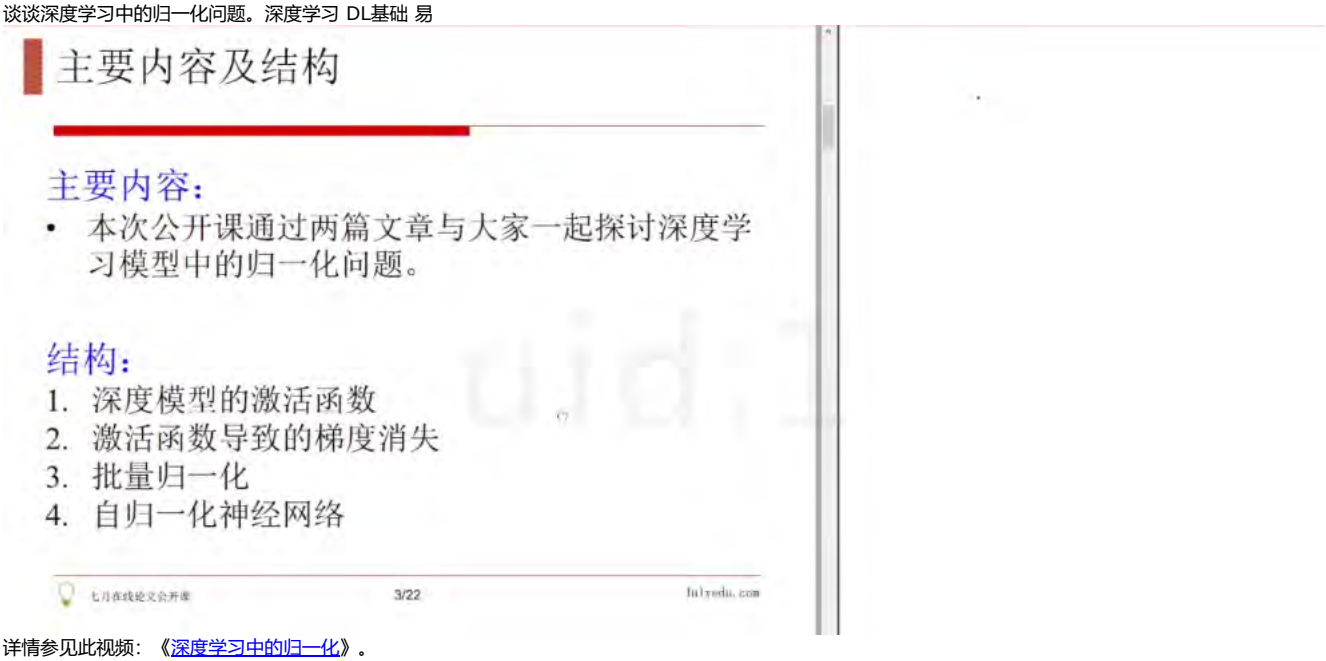
经过处理的数据符合标准正态分布，即均值为0，标准差为1，其转化函数为：

其中μ为所有样本数据的均值，σ为所有样本数据的标准差。

3）非线性归一化

经常用在数据分化比较大的场景，有些数值很大，有些很小。通过一些数学函数，将原始值进行映射。该方法包括 log、指数，正切等。需要根据数据分布的情况，决定非线性函数的曲线，比如log(V, 2)还是log(V, 10)等。

谈谈深度学习中的归一化问题。深度学习 DL基础 易



详情参见此视频：《深度学习中的归一化》。

哪些机器学习算法不需要做归一化处理？机器学习 ML基础 易

概率模型不需要归一化，因为它们不关心变量的值，而是关心变量的分布和变量之间的条件概率，如决策树、rf。而像adaboost、svm、lr、KNN、KMeans之类的最优化问题就需要归一化。

@管博士：我理解归一化和标准化主要是为了使计算更方便 比如两个变量的量纲不同 可能一个的数值远大于另一个那么他们同时作为变量的时候 可能会造成数值计算的问题，比如说求矩阵的逆可能很不精确 或者梯度下降法的收敛比较困难，还有如果需要计算欧式距离的话可能 量纲也需要调整 所以我估计lr 和 knn 保准话一下应该有好处。至于其他的算法 我也觉得如果变量量纲差距很大的话 先标准化一下会有好处。

@寒小阳：一般我习惯说树形模型，这里说的概率模型可能是差不多的意思。

对于树形结构为什么不需要归一化？机器学习 ML基础 易

答：数值缩放，不影响分裂点位置。因为第一步都是按照特征值进行排序的，排序的顺序不变，那么所属的分支以及分裂点就不会有不同。对于线性模型，比如说LR，我有两个特征，一个是(0,1)的，一个是(0,10000)的，这样运用梯度下降时候，损失等高线是一个椭圆的形状，这样我想迭代到最优点，就需要很多次迭代，但是如果进行了归一化，那么等高线就是圆形的，那么SGD就会往原点迭代，需要的迭代次数较少。

另外，注意树模型是不能进行梯度下降的，因为树模型是阶跃的，阶跃点是不可导的，并且求导没意义，所以树模型（回归树）寻找最优点事通过寻找最优分裂点完成的。

数据归一化（或者标准化，注意归一化和标准化不同）的原因。机器学习 ML基础 易

@我愛大泡泡，来源：http://blog.csdn.net/woaidapaopao/article/details/77806273

要强调：能不归一化最好不归一化，之所以进行数据归一化是因为各维度的量纲不相同。而且需要看情况进行归一化。

- 有些模型在各维度进行了不均匀的伸缩后，最优解与原来不等价（如SVM）需要归一化。
- 有些模型伸缩有与原来等价，如：LR则不用归一化，但是实际中往往通过迭代求解模型参数，如果目标函数太扁（想象一下很扁的高斯模型）迭代算法会发生不收敛的情况，所以最坏进行数据归一化。

补充：其实本质是由于loss函数不同造成的，SVM用了欧拉距离，如果一个特征很大就会把其他的维度dominated。而LR可以通过权重调整使得损失函数不变。

请简要说说一个完整机器学习项目的流程。机器学习 ML应用 中

@寒小阳、龙心尘

1 抽象成数学问题

明确问题是进行机器学习的第一步。机器学习的训练过程通常都是一件非常耗时的事情，胡乱尝试时间成本是非常高的。

这里的抽象成数学问题，指的我们明确我们可以获得什么样的数据，目标是一个分类还是回归或者是聚类的问题，如果都不是的话，如果划归为其中的某类问题。

2 获取数据

数据决定了机器学习结果的上限，而算法只是尽可能逼近这个上限。

数据要有代表性，否则必然会过拟合。

而且对于分类问题，数据偏斜不能过于严重，不同类别的数据数量不要有数个数量级的差距。

而且还要对数据的量级有一个评估，多少个样本，多少个特征，可以估算出其对内存的消耗程度，判断训练过程中内存是否能够放得下。如果放不下就得考虑改进算法或者使用一些降维的技巧了。如果数据量实在太大，那就要考虑分布式了。

3 特征预处理与特征选择

良好的数据要能够提取出良好的特征才能真正发挥效力。

特征预处理、数据清洗是很关键的步骤，往往能够使得算法的效果和性能得到显著提高。归一化、离散化、因子化、缺失值处理、去除共线性等，数据挖掘过程中很多时间就花在它们上面。这些工作简单可复制，收益稳定可预期，是机器学习的基础必备步骤。

筛选出显著特征、摒弃非显著特征，需要机器学习工程师反复理解业务。这对很多结果有决定性的影响。特征选择好了，非常简单的算法也能得出良好、稳定的结果。这需要运用特征有效性分析的相关技术，如相关系数、卡方检验、平均互信息、条件熵、后验概率、逻辑回归权重等方法。

4 训练模型与调优

直到这一步才用到我们上面说的算法进行训练。现在很多算法都能够封装成黑盒供人使用。但是真正考验水平的是调整这些算法的（超）参数，使得结果变得更加优良。这需要我们对算法的原理有深入的理解。理解越深入，就越能发现问题的症结，提出良好的调优方案。

5 模型诊断

如何确定模型调优的方向与思路呢？这就需要对模型进行诊断的技术。

过拟合、欠拟合 判断是模型诊断中至关重要的一步。常见的方法如交叉验证，绘制学习曲线等。过拟合的基本调优思路是增加数据量，降低模型复杂度。欠拟合的基本调优思路是提高特征数量和质量，增加模型复杂度。
误差分析 也是机器学习至关重要的步骤。通过观察误差样本，全面分析误差产生误差的原因:是参数的问题还是算法选择的问题，是特征的问题还是数据本身的问题……
诊断后的模型需要进行调优，调优后的新模型需要重新进行诊断，这是一个反复迭代不断逼近的过程，需要不断地尝试， 进而达到最优状态。

6 模型融合
一般来说，模型融合后都能使得效果有一定提升。而且效果很好。
工程上，主要提升算法准确度的方法是分别在模型的前端（特征清洗和预处理，不同的采样模式）与后端（模型融合）上下功夫。因为他们比较标准可复制，效果比较稳定。而直接调参的工作不会很多，毕竟大量数据训练起来太慢了，而且效果难以保证。

7 上线运行
这一部分内容主要跟工程实现的相关性比较大。工程上是结果导向，模型在线上运行的效果直接决定模型的成败。 不单纯包括其准确程度、误差等情况，还包括其运行的速度(时间复杂度)、资源消耗程度（空间复杂度）、稳定性是否可接受。
这些工作流程主要是工程实践上总结出的一些经验。并不是每个项目都包含完整的一个流程。这里的部分只是一个指导性的说明，只有大家自己多实践，多积累项目经验，才会有自己更深刻的认识。

故，基于此，七月在线每一期ML算法班都特此增加特征工程、模型调优等相关课。比如，这里有个公开课视频《特征处理与特征选择》。

逻辑斯特回归为什么要对特征进行离散化。机器学习 ML模型 中等
@严林，本题解析来源：https://www.zhihu.com/question/31989952

在工业界，很少直接将连续值作为逻辑回归模型的特征输入，而是将连续特征离散化为一系列0、1特征交给逻辑回归模型，这样做的优势有以下几点：

0. 离散特征的增加和减少都很容易，易于模型的快速迭代；

1. 稀疏向量内积乘法运算速度快，计算结果方便存储，容易扩展；

2. 离散化后的特征对异常数据有很强的鲁棒性：比如一个特征是年龄>30是1，否则0。如果特征没有离散化，一个异常数据“年龄300岁”会给模型造成很大的干扰；

3. 逻辑回归属于广义线性模型，表达能力受限；单变量离散化为N个后，每个变量有单独的权重，相当于为模型引入了非线性，能够提升模型表达能力，加大拟合；

4. 离散化后可以进行特征交叉，由M+N个变量变为M*N个变量，进一步引入非线性，提升表达能力；

5. 特征离散化后，模型会更稳定，比如如果对用户年龄离散化，20-30作为一个区间，不会因为一个用户年龄长了一岁就变成一个完全不同的人。当然处于区间相邻处的样本会刚好相反，所以怎么划分区间是门学问；

6. 特征离散化以后，起到了简化了逻辑回归模型的作用，降低了模型过拟合的风险。

李沐曾经说过：模型是使用离散特征还是连续特征，其实是一个“海量离散特征+简单模型” 同 “少量连续特征+复杂模型”的权衡。既可以离散化用线性模型，也可以用连续特征加深度学习。就看是喜欢折腾特征还是折腾模型了。通常来说，前者容易，而且可以n个人一起并行做，有成功经验；后者目前看很赞，能走多远还须拭目以待。

new 和 malloc的区别。编程开发 C/C++ 易
@Sommer_Xia，来源：http://blog.csdn.net/shymi1991/article/details/39432775
1. malloc与free是C++/C语言的标准库函数，new/delete是C++的运算符。它们都可用于申请动态内存和释放内存。
2. 对于非内部数据类型的对象而言，光用maloc/free无法满足动态对象的要求。对象在创建的同时要自动执行构造函数，对象在消亡之前要自动执行析构函数。由于malloc/free是库函数而不是运算符，不在编译器控制权限之内，不能够把执行构造函数和析构函数的任务强加于malloc/free。
3. 因此C++语言需要一个能完成动态内存分配和初始化工作的运算符new，以一个能完成清理与释放内存工作的运算符delete。注意new/delete不是库函数。
4. C++程序经常要调用C函数，而C程序只能用malloc/free管理动态内存

hash 冲突及解决办法。数据结构/算法 中等
@Sommer_Xia，来源：http://blog.csdn.net/shymi1991/article/details/39432775
关键字值不同的元素可能会映象到哈希表的同一地址上就会发生哈希冲突。解决办法：
1）开放定址法：当冲突发生时，使用某种探查(亦称探测)技术在散列表中形成一个探查(测)序列。沿此序列逐个单元地查找，直到找到给定 的关键字，或者碰到一个开放的地址(即该地址单元为空)为止（若要插入，在探查到开放的地址，则可将待插入的新结点存人该地址单元）。查找时探查到开放的 地址则表明表中无待查的关键字，即查找失败。
2） 再哈希法：同时构造多个不同的哈希函数。
3）链地址法：将所有哈希地址为i的元素构成一个称为同义词链的单链表，并将单链表的头指针存在哈希表的第i个单元中，因而查找、插入和删除主要在同义词链中进行。链地址法适用于经常进行插入和删除的情况。
4）建立公共溢出区：将哈希表分为基本表和溢出表两部分，凡是和基本表发生冲突的元素，一律填入溢出表。

下列哪个不属于CRF模型对于HMM和MEMM模型的优势（B ） 机器学习 ML模型 中等
A. 特征灵活 B. 速度快 C. 可容纳较多上下文信息 D. 全局最优
首先，CRF，HMM(隐马模型)，MEMM(最大熵隐马模型)都常用来做序列标注的建模.
隐马模型一个最大的缺点就是由于其输出独立性假设，导致其不能考虑上下文的特征，限制了特征的选择
最大熵隐马模型则解决了隐马的问题，可以任意选择特征，但由于其在每一节点都要进行归一化，所以只能找到局部的最优值，同时也带来了标记偏见的问题，即凡是训练语料中未出现的情况全都忽略掉
条件随机场则很好的解决了这一问题，他并不在每一个节点进行归一化，而是所有特征进行全局归一化，因此可以求得全局的最优值。
此外《机器学习工程师第八期》里有讲概率图模型。

什么是熵。机器学习 ML基础 易

从名字上来看，熵给人一种很玄乎，不知道是啥的感觉。其实，熵的定义很简单，即用来表示随机变量的不确定性。之所以给人玄乎的感觉，大概是因为为何要取这样的名字，以及怎么用。

熵的概念最早起源于物理学，用于度量一个热力学系统的无序程度。在信息论里面，熵是对不确定性的测量。

熵的引入

事实上，熵的英文原文为entropy，最初由德国物理学家鲁道夫·克劳修斯提出，其表达式为：

$$\Delta S = \frac{Q}{T}$$

它表示一个系系统在不受外部干扰时，其内部最稳定的状态。后来一中国学者翻译entropy时，考虑到entropy是能量Q跟温度T的商，且跟火有关，便把entropy形象的翻译成“熵”。

我们知道，任何粒子的常态都是随机运动，也就是"无序运动"，如果让粒子呈现"有序化"，必须耗费能量。所以，温度（热能）可以被看作"有序化"的一种度量，而"熵"可以看作是"无序化"的度量。

如果没有外部能量输入，封闭系统趋向越来越混乱（熵越来越大）。比如，如果房间无人打扫，不可能越来越干净（有序化），只可能越来越乱（无序化）。而要让一个系统变得更有序，必须有外部能量的输入。

1948年，香农Claude E. Shannon引入信息（熵），将其定义为离散随机事件的出现概率。一个系统越是有序，信息熵就越低；反之，一个系统越是混乱，信息熵就越高。所以说，信息熵可以被认为是系统有序化程度的一个度量。

更多请查看《最大熵模型中的数学推导》。

熵、联合熵、条件熵、相对熵、互信息的定义。机器学习 ML基础 中等

为了更好的理解，需要了解的概率必备知识有：

1. 大写字母X表示随机变量，小写字母x表示随机变量X的某个具体的取值；
2. P(X)表示随机变量X的概率分布，P(X,Y)表示随机变量X、Y的联合概率分布，P(Y|X)表示已知随机变量X的情况下随机变量Y的条件概率分布；
3. p(X = x)表示随机变量X取某个具体值的概率，简记为p(x)；
4. p(X = x, Y = y) 表示联合概率，简记为p(x,y)，p(Y = y|X = x)表示条件概率，简记为p(y|x)，且有：p(x,y) = p(x) * p(y|x)。

熵：如果一个随机变量X的可能取值为X = {x1, x2,…, xk}，其概率分布为P(X = xi) = pi（i = 1,2, ..., n），则随机变量X的熵定义为：

$$H(X)=-\sum_{x} p(x)\log p(x)$$

把最前面的负号放到最后，便成了：

$$H(X)=\sum_{x} p(x)\log \frac{1}{p(x)}$$

上面两个熵的公式，无论用哪个都行，而且两者等价，一个意思（这两个公式在下文中都会用到）。

联合熵：两个随机变量X，Y的联合分布，可以形成联合熵Joint Entropy，用H(X,Y)表示。
条件熵：在随机变量X发生的前提下，随机变量Y发生所新带来的熵定义为Y的条件熵，用H(Y|X)表示，用来衡量在已知随机变量X的条件下随机变量Y的不确定性。

且有此式子成立：H(Y|X) = H(X,Y) – H(X)，整个式子表示(X,Y)发生所包含的熵减去X单独发生包含的熵。至于怎么得来的请看推导：

$$\begin{aligned}
&H(X,Y)-H(X)\\
&=-\sum_{x,y} p(x,y)\log p(x,y)+\sum_{x} p(x)\log p(x)\\
&=-\sum_{x,y} p(x,y)\log p(x,y)+\sum_{x}\left(\sum_{y} p(x,y)\right)\log p(x)\\
&=-\sum_{x,y} p(x,y)\log p(x,y)+\sum_{x,y} p(x,y)\log p(x)\\
&=-\sum_{x,y} p(x,y)\log \frac{p(x,y)}{p(x)}\\
&=-\sum_{x,y} p(x,y)\log p(y\mid x)
\end{aligned}$$

简单解释下上面的推导过程。整个式子共6行，其中

第二行推到第三行的依据是边缘分布p(x)等于联合分布p(x,y)的和；
第三行推到第四行的依据是把公因子logp(x)乘进去，然后把x,y写在一起；
第四行推到第五行的依据是：因为两个sigma都有p(x,y)，故提取公因子p(x,y)放到外边，然后把里边的-（log p(x,y) - log p(x)）写成- log (p(x,y)/p(x))；
第五行推到第六行的依据是：p(x,y) = p(x) * p(y|x)，故p(x,y) / p(x) = p(y|x)。

相对熵：又称互熵，交叉熵，鉴别信息，Kullback熵，Kullback-Leible散度等。设p(x)、q(x)是X中取值的两个概率分布，则p对q的相对熵是：

$$D(p\,\|\,q)=\sum_{x} p(x)\log \frac{p(x)}{q(x)}=E_{p(x)}\log \frac{p(x)}{q(x)}$$

在一定程度上，相对熵可以度量两个随机变量的“距离”，且有D(p||q) ≠D(q||p)。另外，值得一提的是，D(p||q)是必然大于等于0的。

互信息：两个随机变量X，Y的互信息定义为X，Y的联合分布和各自独立分布乘积的相对熵，用I(X,Y)表示：

$$I(X,Y)=\sum_{x,y} p(x,y)\log \frac{p(x,y)}{p(x)p(y)}$$

且有I(X,Y)=D(P(X,Y) || P(X)P(Y))。下面，咱们来计算下H(Y)-I(X,Y)的结果，如下：

$$
\begin{aligned}
&H(Y)-I(X,Y)\\
&=-\sum_{y}p(y)\log p(y)-\sum_{x,y}p(x,y)\log\frac{p(x,y)}{p(x)p(y)}\\
&=-\sum_{y}\left(\sum_{x}p(x,y)\right)\log p(y)-\sum_{x,y}p(x,y)\log\frac{p(x,y)}{p(x)p(y)}\\
&=-\sum_{x,y}p(x,y)\log p(y)-\sum_{x,y}p(x,y)\log\frac{p(x,y)}{p(x)p(y)}\\
&=-\sum_{x,y}p(x,y)\log\frac{p(x,y)}{p(x)}\\
&=-\sum_{x,y}p(x,y)\log p(y\mid x)\\
&=H(Y\mid X)
\end{aligned}
$$

通过上面的计算过程，我们发现竟然有H(Y)-I(X,Y) = H(Y|X)。故通过条件熵的定义，有：H(Y|X) = H(X,Y) - H(X)，而根据互信息定义展开得到H(Y|X) = H(Y) - I(X,Y)，把前者跟后者结合起来，便有I(X,Y)= H(X) + H(Y) - H(X,Y)，此结论被多数文献作为互信息的定义。更多请查看《最大熵模型中的数学推导》。

什么是最大熵。机器学习 ML基础 易

熵是随机变量不确定性的度量，不确定性越大，熵值越大；若随机变量退化成定值，熵为0。如果没有外界干扰，随机变量总是趋向于无序，在经过足够时间的稳定演化，它应该能够达到的最大程度的熵。

为了准确的估计随机变量的状态，我们一般习惯性最大化熵，认为在所有可能的概率模型（分布）的集合中，熵最大的模型是最好的模型。换言之，在已知部分知识的前提下，关于未知分布最合理的推断就是符合已知知识最不确定或最随机的推断，其原则是承认已知事物（知识），且对未知事物不做任何假设，没有任何偏见。

例如，投掷一个骰子，如果问"每个面朝上的概率分别是多少"，你会说是等概率，即各点出现的概率均为1/6。因为对这个"一无所知"的色子，什么都不确定，而假定它每一个朝上概率均等则是最合理的做法。从投资的角度来看，这是风险最小的做法，而从信息论的角度讲，就是保留了最大的不确定性，也就是说让熵达到最大。

3.1 无偏原则

下面再举个大多数有关最大熵模型的文章中都喜欢举的一个例子。

例如，一篇文章中出现了“学习”这个词，那这个词是主语、谓语、还是宾语呢？换言之，已知“学习”可能是动词，也可能是名词，故“学习”可以被标为主语、谓语、宾语、定语等等。

令x1表示“学习”被标为名词， x2表示“学习”被标为动词。
令y1表示“学习”被标为主语， y2表示被标为谓语， y3表示宾语， y4表示定语。

且这些概率值加起来的和必为1，即 $p(x_1)+p(x_2)=1$ ， $\sum_{i=1}^{4}p(y_i)=1$ ， 则根据无偏原则，认为这个分布中取各个值的概率是相等的，故得到：

$$
\begin{aligned}
&p(x_1)=p(x_2)=0.5\\
&p(y_1)=p(y_2)=p(y_3)=p(y_4)=0.25
\end{aligned}
$$

因为没有任何的先验知识，所以这种判断是合理的。如果有了一定的先验知识呢?

即进一步，若已知：“学习”被标为定语的可能性很小，只有0.05，即 $p(y_4)=0.05$ ，剩下的依然根据无偏原则，可得：

$$
\begin{aligned}
&p(x_1)=p(x_2)=0.5\\
&p(y_1)=p(y_2)=p(y_3)=\frac{0.95}{3}
\end{aligned}
$$

再进一步，当“学习”被标作名词x1的时候，它被标作谓语y2的概率为0.95，即 $p(y_2\mid x_1)=0.95$ ，此时仍然需要坚持无偏见原则，使得概率分布尽量平均。但怎么样才能得到尽量无偏见的分布?

实践经验和理论计算都告诉我们，在完全无约束状态下，均匀分布等价于熵最大（有约束的情况下，不一定是概率相等的均匀分布。 比如，给定均值和方差，熵最大的分布就变成了正态分布 ）。

于是，问题便转化为了：计算X和Y的分布，使得H(Y|X)达到最大值，并且满足下述条件：

$$
\begin{aligned}
&p(x_1)+p(x_2)=1\\
&\sum_{i=1}^{4}p(y_i)=1\\
&p(y_4)=0.05\\
&p(y_2\mid x_1)=0.95
\end{aligned}
$$

因此，也就引出了最大熵模型的本质，它要解决的问题就是已知X，计算Y的概率，且尽可能让Y的概率最大（实践中，X可能是某单词的上下文信息，Y是该单词翻译成me，I，us、we的各自概率），从而根据已有信息，尽可能最准确的推测未知信息，这就是最大熵模型所要解决的问题。

相当于已知X，计算Y的最大可能的概率，转换成公式，便是要最大化下述式子H(Y|X)：

$$
\max H(Y\mid X)=\sum_{\substack{x\in\{x_1,x_2\}\\ y\in\{y_1,y_2,y_3,y_4\}}}p(x,y)\log\frac{1}{p(y\mid x)}
$$

且满足以下4个约束条件：

$$
\begin{aligned}
&p(x_1)+p(x_2)=1\\
&p(y_1)+p(y_2)+p(y_3)+p(y_4)=1\\
&p(y_4)=0.05\\
&p(y_2\mid x_1)=0.95
\end{aligned}
$$

简单说下有监督学习和无监督学习的区别。机器学习 ML基础 易
有监督学习：对具有标记的训练样本进行学习，以尽可能对训练样本集外的数据进行分类预测。 (LR,SVM,BP,RF,GBDT)
无监督学习：对未标记的样本进行训练学习，比发现这些样本中的结构知识。(KMeans,DL)

了解正则化么。机器学习 ML基础 易
正则化是针对过拟合而提出的，以为在求解模型最优的是一般优化最小的经验风险，现在在该经验风险上加入模型复杂度这一项（正则化项是模型参数向量的范数），并使用一个rate比率来权衡模型复杂度与以往经验风险的权重，如果模型复杂度越高，结构化的经验风险会越大，现在的目标就变为了结构经验风险的最优化，可以防止模型训

练过度复杂，有效的降低过拟合的风险。
奥卡姆剃刀原理，能够很好的解释已知数据并且十分简单才是最好的模型。

协方差和相关性有什么区别？机器学习 ML基础 易
相关性是协方差的标准化格式。协方差本身很难做比较。例如：如果我们计算工资（$）和年龄（岁）的协方差，因为这两个变量有不同的度量，所以我们会得到不能做比较的不同的协方差。

$$\Sigma_{ij} = \mathrm{cov}(X_i, X_j) = \mathrm{E}[\,(X_i - \mu_i)(X_j - \mu_j)\,] = \mathrm{E}[\,X_i X_j\,] - \mu_i \mu_j$$

为了解决这个问题，我们计算相关性来得到一个介于-1和1之间的值，就可以忽略它们各自不同的度量。

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

线性分类器与非线性分类器的区别以及优劣。机器学习 ML基础 易
@伟祺，线性和非线性是针对，模型参数和输入特征来讲的；比如输入x，模型y=ax+ax^2那么就是非线性模型，如果输入是x和X^2则模型是线性的。
线性分类器可解释性好，计算复杂度较低，不足之处是模型的拟合效果相对弱些。
非线性分类器效果拟合能力较强，不足之处是数据量不足容易过拟合、计算复杂度高、可解释性不好。
常见的线性分类器有：LR,贝叶斯分类，单层感知机、线性回归
常见的非线性分类器：决策树、RF、GBDT、多层感知机
SVM两种都有（看线性核还是高斯核）

数据的逻辑存储结构（如数组，队列，树等）对于软件开发具有十分重要的影响，试对你所了解的各种存储结构从运行速度、存储效率和适用场合等方面进行简要地分析。 数据结构/算法 中等

	运行速度	存储效率	适用场合	
数组	快	高	比较适合进行查找操作，还有像类似于矩阵等的操作	
链表	较快	较高	比较适合增删改频繁操作，动态的分配内存	
队列	较快	较高	比较适合进行任务类等的调度	
栈	一般	较高	比较适合递归类程序的改写	
二叉树（树）	较快	一般	一切具有层次关系的问题都可用树来描述	
图	一般	一般	除了像最小生成树、最短路径、拓扑排序等经典用途。还被用于像神经网络等人工智能领域等等。	

什么是分布式数据库？计算机基础 数据库 易
分布式数据库系统是在集中式数据库系统成熟技术的基础上发展起来的，但不是简单地把集中式数据库分散地实现，它具有自己的性质和特征。集中式数据库系统的许多概念和技术，如数据独立性、数据共享和减少冗余度、并发控制、完整性、安全性和恢复等在分布式数据库系统中都有了不同的、更加丰富的内容。
具体来说，集群文件系统是指运行在多台计算机之上，之间通过某种方式相互通信从而将集群内所有存储空间资源整合、虚拟化并对外提供文件访问服务的文件系统。其与NTFS、EXT等本地文件系统的目的不同，前者是为了扩展性，后者运行在单机环境，纯粹管理块和文件之间的映射以及文件属性。

集群文件系统分为多类，按照对存储空间的访问方式，可分为共享存储型集群文件系统和分布式集群文件系统，前者是多台计算机识别到同样的存储空间，并相互协调共同管理其上的文件，又被称为共享文件系统；后者则是每台计算机各自提供自己的存储空间，并各自协调管理所有计算机节点中的文件。Veritas的VxFS/VCS，昆腾Stornext，中科蓝鲸BWFS，EMC的MPFS，属于共享存储型集群文件系统。而HDFS、Gluster、Ceph、Swift等互联网常用的大规模集群文件系统无一例外都属于分布式集群文件系统。分布式集群文件系统可扩展性更强，目前已知最大可扩展至10K节点。

按照元数据的管理方式，可分为对称式集群文件系统和非对称式集群文件系统。前者每个节点的角色均等，共同管理文件元数据，节点间通过高速网络进行信息同步和互斥锁等操作，典型代表是Veritas的VCS。而非对称式集群文件系统中，有专门的一个或者多个节点负责管理元数据，其他节点需要频繁与元数据节点通信以获取最新的元数据比如目录列表文件属性等等，后者典型代表比如HDFS、GFS、BWFS、Stornext等。对于集群文件系统，其可以是分布式+对称式、分布式+非对称式、共享式+对称式、共享式+非对称式，两两任意组合。

按照文件访问方式来分类，集群文件系统可分为串行访问式和并行访问式，后者又被俗称为并行文件系统。
串行访问是指客户端只能从集群中的某个节点来访问集群内的文件资源，而并行访问则是指客户端可以直接从集群中任意一个或者多个节点同时收发数据，做到并行数据存取，加快速度。
HDFS、GFS、pNFS等集群文件系统，都支持并行访问，需要安装专用客户端，传统的NFS/CIFS客户端不支持并行访问。

简单说说贝叶斯定理。机器学习 ML模型 易
在引出贝叶斯定理之前，先学习几个定义：

条件概率（又称后验概率）就是事件A在另外一个事件B已经发生条件下的发生概率。条件概率表示为P(A|B)，读作“在B条件下A的概率”。

比如，在同一个样本空间Ω中的事件或者子集A与B，如果随机从Ω中选出的一个元素属于B，那么这个随机选择的元素还属于A的概率就定义为在B的前提下A的条件概率，所以：P(A|B) = |A∩B|/|B|，接着分子、分母都除以|Ω|得到

$$P(A|B) = \frac{P(A \cap B)}{P(B)}$$

联合概率表示两个事件共同发生的概率。A与B的联合概率表示为$P(A \cap B)$或者$P(A, B)$。
边缘概率（又称先验概率）是某个事件发生的概率。边缘概率是这样得到的：在联合概率中，把最终结果中那些不需要的事件通过合并成它们的全概率，而消去它们（对离散随机变量用求和得全概率，对连续随机变量用积分得全概率），这称为边缘化（marginalization），比如A的边缘概率表示为P(A)，B的边缘概率表示为P(B)。

接着，考虑一个问题：P(A|B)是在B发生的情况下A发生的可能性。

1. 首先，事件B发生之前，我们对事件A的发生有一个基本的概率判断，称为A的先验概率，用P(A)表示；
2. 其次，事件B发生之后，我们对事件A的发生概率重新评估，称为A的后验概率，用P(A|B)表示；
3. 类似的，事件A发生之前，我们对事件B的发生有一个基本的概率判断，称为B的先验概率，用P(B)表示；
4. 同样，事件A发生之后，我们对事件B的发生概率重新评估，称为B的后验概率，用P(B|A)表示。

贝叶斯定理便是基于下述贝叶斯公式：

$$P(A|B) = \frac{P(B|A)\,P(A)}{P(B)}$$

上述公式的推导其实非常简单，就是从条件概率推出。

根据条件概率的定义，在事件B发生的条件下事件A发生的概率是

$$P(A|B) = \frac{P(A \cap B)}{P(B)}$$

同样地，在事件A发生的条件下事件B发生的概率

$$P(B|A) = \frac{P(A \cap B)}{P(A)}.$$

整理与合并上述两个方程式，便可以得到：

$$P(A|B)\,P(B) = P(A \cap B) = P(B|A)\,P(A).$$

接着，上式两边同除以P(B)，若P(B)是非零的，我们便可以得到贝叶斯定理的公式表达式：

$$P(A|B) = \frac{P(B|A)\,P(A)}{P(B)}.$$

所以，贝叶斯公式可以直接根据条件概率的定义直接推出。即因为P(A,B) = P(A)P(B|A) = P(B)P(A|B)，所以P(A|B) = P(A)P(B|A) / P(B)。更多请参见此文：《从贝叶斯方法谈到贝叶斯网络》。

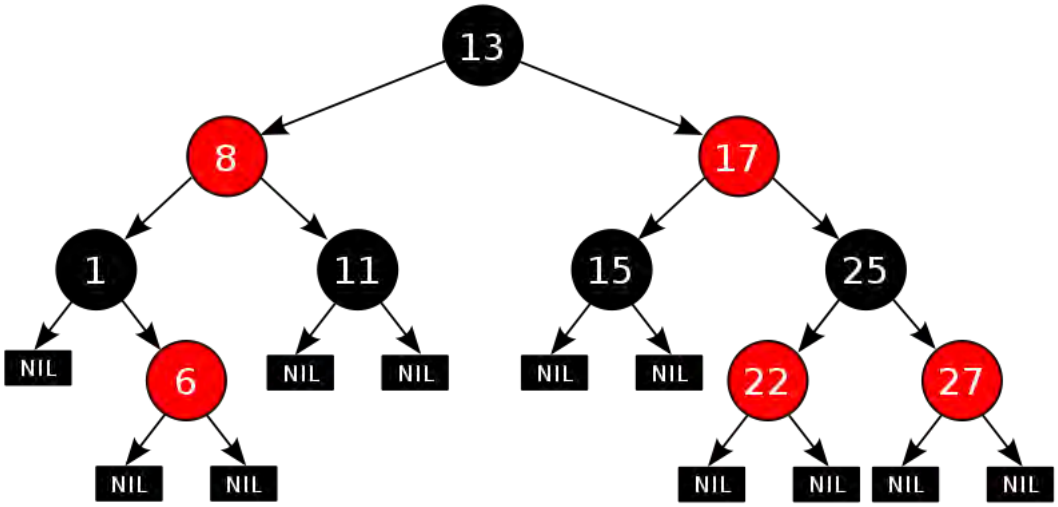
#include和#include “filename.h”有什么区别？计算机基础 编译原理 易
用 #include 格式来引用标准库的头文件（编译器将从标准库目录开始搜索）。
用 #include “filename.h” 格式来引用非标准库的头文件（编译器将从用户的工作目录开始搜索）。
某超市研究销售纪录数据后发现，买啤酒的人很大概率也会购买尿布，这种属于数据挖掘的哪类问题？(A) 数据挖掘 DM模型 易
A. 关联规则发现 B. 聚类
C. 分类 D. 自然语言处理
将原始数据进行集成、变换、维度规约、数值规约是在以下哪个步骤的任务？(C) 数据挖掘 DM基础 易
A. 频繁模式挖掘 B. 分类和预测 C. 数据预处理 D. 数据流挖掘
下面哪种不属于数据预处理的方法？ (D) 数据挖掘 DM基础 易
A变量代换 B离散化 C 聚集 D 估计遗漏值
什么是KDD？ (A) 数据挖掘 DM基础 易
A. 数据挖掘与知识发现 B. 领域知识发现
C. 文档知识发现 D. 动态知识发现
当不知道数据所带标签时，可以使用哪种技术促使带同类标签的数据与带其他标签的数据相分离？(B) 数据挖掘 DM模型 易
A. 分类 B. 聚类 C. 关联分析 D. 隐马尔可夫链
建立一个模型，通过这个模型根据已知的变量值来预测其他某个变量值属于数据挖掘的哪一类任务？(C) 数据挖掘 DM基础 易
A. 根据内容检索 B. 建模描述
C. 预测建模 D. 寻找模式和规则
以下哪种方法不属于特征选择的标准方法： (D) 数据挖掘 DM基础 易
A嵌入 B 过滤 C 包装 D 抽样
请用python编写函数find_string，从文本中搜索并打印内容，要求支持通配符星号和问号。Python Python语言 易
例子：

```
>>>find_string('hello\nworld\n','wor')
['wor']
>>>find_string('hello\nworld\n','l*d')
['ld']
>>>find_string('hello\nworld\n','o.')
['or']
```

答案

```
def find_string(str,pat):
 import re
 return re.findall(pat,str,re.I)
```

说下红黑树的五个性质。数据结构 树 易
红黑树，一种二叉查找树，但在每个结点上增加一个存储位表示结点的颜色，可以是Red或Black。
通过对任何一条从根到叶子的路径上各个结点着色方式的限制，红黑树确保没有一条路径会比其他路径长出俩倍，因而是接近平衡的。



红黑树，作为一棵二叉查找树，满足二叉查找树的一般性质。下面，来了解下 二叉查找树的一般性质。

二叉查找树，也称有序二叉树（ordered binary tree），或已排序二叉树（sorted binary tree），是指一棵空树或者具有下列性质的二叉树：

若任意节点的左子树不空，则左子树上所有结点的值均小于它的根结点的值；

若任意节点的右子树不空，则右子树上所有结点的值均大于它的根结点的值；

任意节点的左、右子树也分别为二叉查找树。

没有键值相等的节点（no duplicate nodes）。

因为一棵由n个结点随机构造的二叉查找树的高度为lgn，所以顺理成章，二叉查找树的一般操作的执行时间为O(lgn)。但二叉查找树若退化成了一棵具有n个结点的线性链后，则这些操作最坏情况运行时间为O(n)。

红黑树虽然本质上是一棵二叉查找树，但它在二叉查找树的基础上增加了着色和相关的性质使得红黑树相对平衡，从而保证了红黑树的查找、插入、删除的时间复杂度最坏为O(log n)。

但它是如何保证一棵n个结点的红黑树的高度始终保持在logn的呢？这就引出了红黑树的5个性质：

每个结点要么是红的要么是黑的。

根结点是黑的。

每个叶结点（叶结点即指树尾端NIL指针或NULL结点）都是黑的。

如果一个结点是红的，那么它的两个儿子都是黑的。

对于任意结点而言，其到叶结点树尾端NIL指针的每条路径都包含相同数目的黑结点。

正是红黑树的这5条性质，使一棵n个结点的红黑树始终保持了logn的高度，从而也就解释了上面所说的“红黑树的查找、插入、删除的时间复杂度最坏为O(log n)”这一结论成立的原因。更多请参见此文：《教你初步了解红黑树》。

简单说下sigmoid激活函数。深度学习 DL基础 易

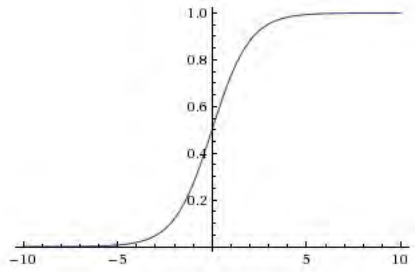
常用的非线性激活函数有sigmoid、tanh、relu等等，前两者sigmoid/tanh比较常见于全连接层，后者relu常见于卷积层。这里先简要介绍下最基础的sigmoid函数（btw，在本博客中SVM那篇文章开头有提过）。

sigmoid的函数表达式如下

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

其中z是一个线性组合，比如z可以等于：b + $w_1 * x_1 + w_2 * x_2$。通过代入很大的正数或很小的负数到g(z)函数中可知，其结果趋近于0或1。

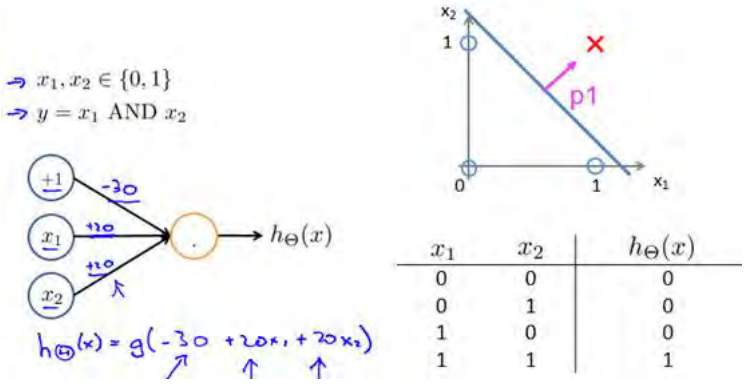
因此，sigmoid函数g(z)的图形表示如下（横轴表示定义域z，纵轴表示值域g(z)）：



也就是说，sigmoid 函数的功能是相当于把一个实数压缩至0到1之间。当z是非常大的正数时，g(z) 会趋近于1，而z是非常小的负数时，则g(z)会趋近于0。

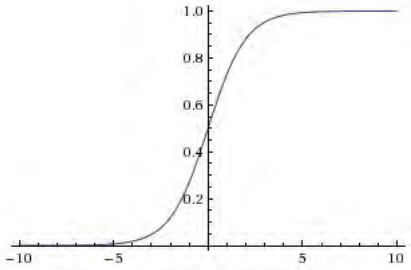
压缩至0到1有何用处呢？用处是这样一来便可以把激活函数看作一种“分类的概率”，比如激活函数的输出为0.9的话便可以解释为90%的概率为正样本。

举个例子，如下图（图引自Stanford机器学习公开课）



x_1	x_2	$h_\Theta(x)$
0	0	0
0	1	0
1	0	0
1	1	1

z = b + w_1*x_1 + w_2*x_2，其中b为偏置项 假定取-30，w_1、w_2都取为20



如果x_1 = 0 x_2 = 0，则z = -30，g(z) = 1/(1 + e^-z)趋近于0。此外，从上图sigmoid函数的图形上也可以看出，当z=-30的时候，g(z)的值趋近于0

如果x_1 = 0 x_2 = 1，或x_1 =1 x_2 = 0，则z = b + w_1*x_1 + w_2*x_2 = -30 + 20 = -10，同样，g(z)的值趋近于0

如果x_1 = 1 x_2 = 1，则z = b + w_1*x_1 + w_2*x_2 = -30 + 20*1 + 20*1 = 10，此时，g(z)趋近于1。

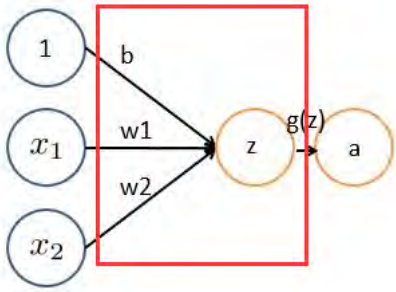
换言之，只有x_1和x_2都取1的时候，g(z)→1，判定为正样本；x_1或x_2取0的时候，g(z)→0，判定为负样本，如此达到分类的目的。

综上，sigmod函数，是逻辑斯蒂回归的压缩函数，它的性质是可以把分隔平面压缩到[0,1]区间一个数（向量），在线性分割平面值为0时候正好对应sigmod值为0.5，大于0对应sigmod值大于0.5、小于0对应sigmod值小于0.5；0.5可以作为分类的阀值；exp的形式最值求解时候比较方便，用相乘形式作为logistic损失函数，使得损失函数是凸函数；不足之处是sigmod函数在y趋于0或1时候有死区，控制不好在bp形式传递loss时候容易造成梯度弥撒。

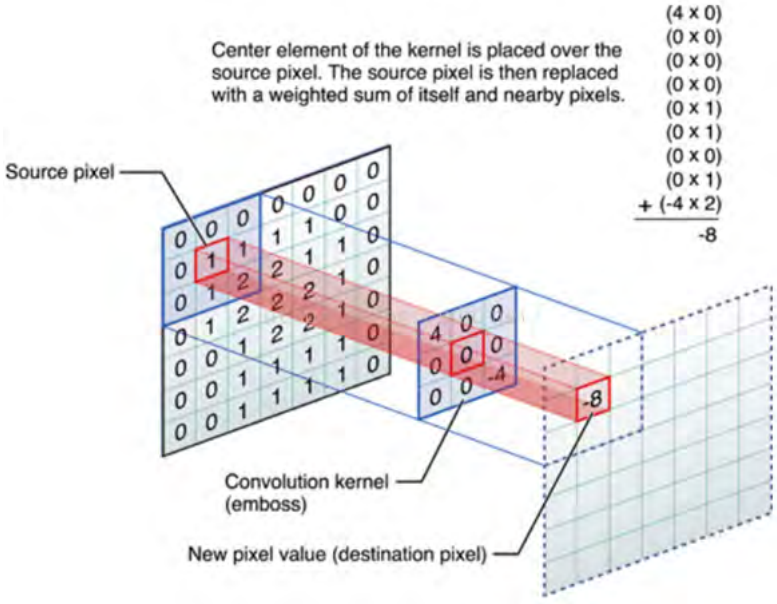
什么是卷积。深度学习 DL基础 易

对图像（不同的数据窗口数据）和滤波矩阵（一组固定的权重：因为每个神经元的多个权重固定，所以又可以看做一个恒定的滤波器filter）做内积（逐个元素相乘再求和）的操作就是所谓的『卷积』操作，也是卷积神经网络的名字来源。

非严格意义上来讲，下图中红框框起来的部分便可以理解为一个滤波器，即带着一组固定权重的神经元。多个滤波器叠加便成了卷积层。



OK，举个具体的例子。比如下图中，图中左边部分是原始输入数据，图中中间部分是滤波器filter，图中右边是输出的新的二维数据。



分解下上图

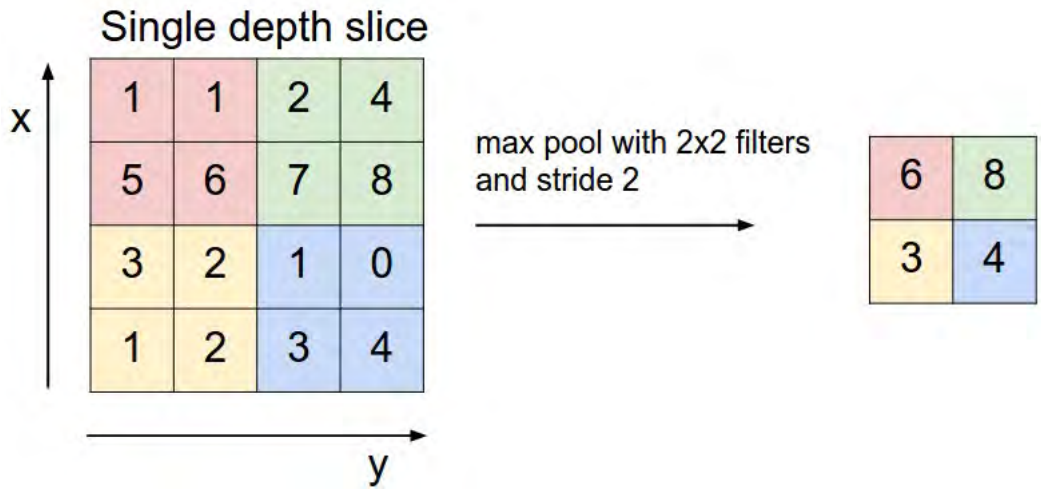
对应位置上是数字先相乘后相加

=

中间滤波器filter与数据窗口做内积，其具体计算过程则是：4*0 + 0*0 + 0*0 + 0*0 + 0*1 + 0*1 + 0*0 + 0*1 + -4*2 = -8

什么是CNN的池化pool层。深度学习 DL模型 易

池化，简言之，即取区域平均或最大，如下图所示（图引自cs231n）

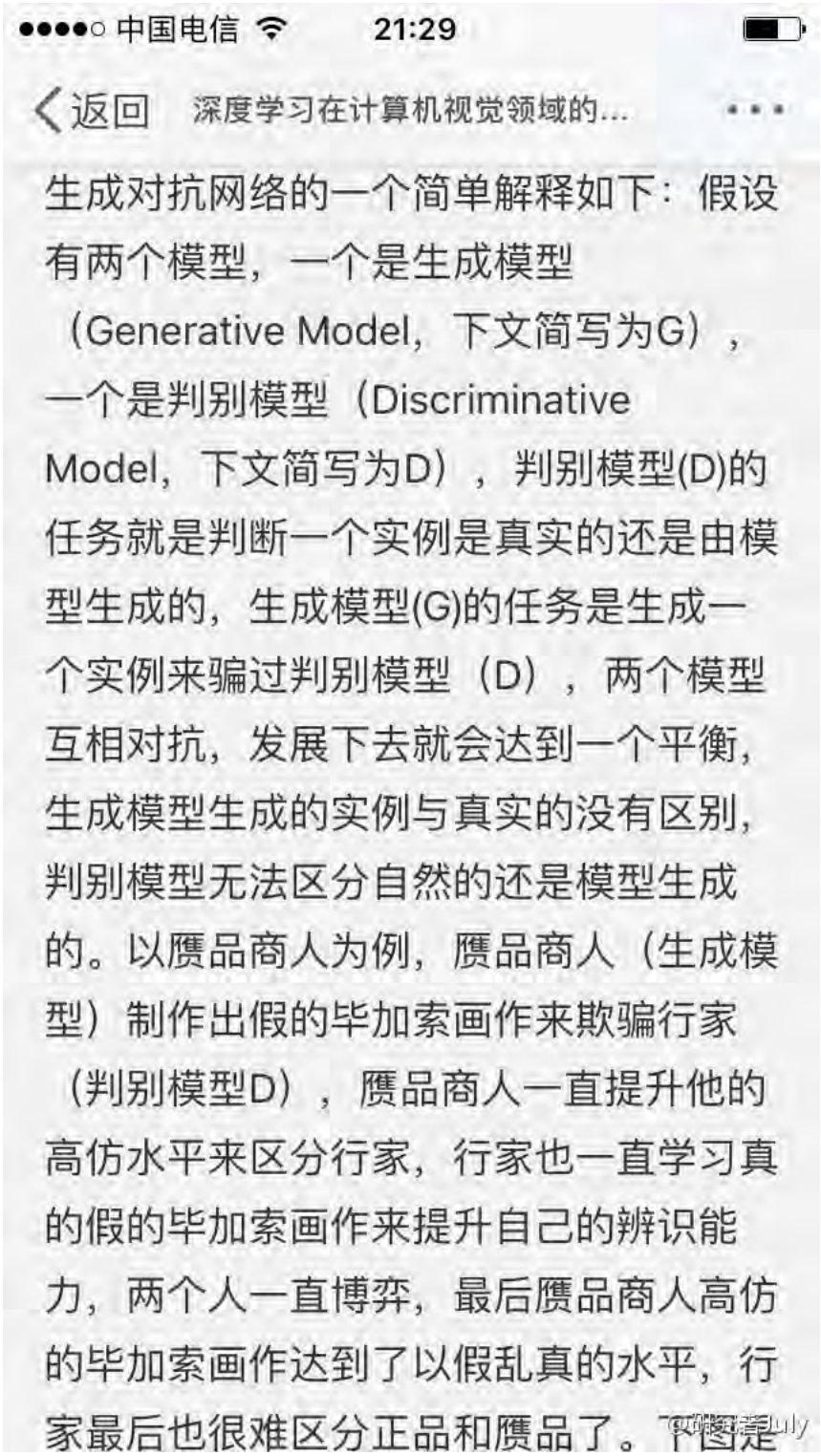


上图所展示的是取区域最大，即上图左边部分中 左上角2x2的矩阵中6最大，右上角2x2的矩阵中8最大，左下角2x2的矩阵中3最大，右下角2x2的矩阵中4最大，所以得到上图右边部分的结果：6 8 3 4。很简单不是?

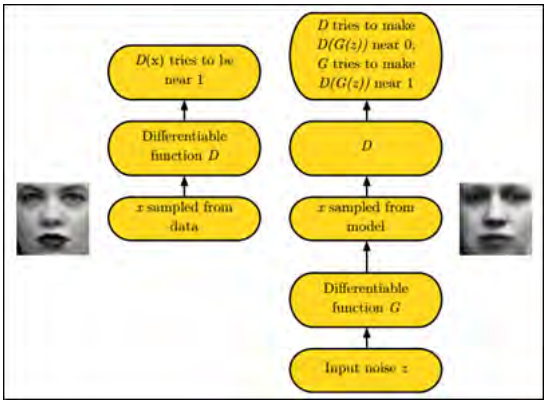
简述下什么是生成对抗网络。深度学习 DL扩展 中

GAN之所以是对抗的，是因为GAN的内部是竞争关系，一方叫generator，它的主要工作是生成图片，并且尽量使得其看上去是来自于训练样本的。另一方是discriminator，其目标是判断输入图片是否属于真实训练样本。

更直白的讲，将generator想象成假币制造商，而discriminator是警察。generator目的是尽可能把假币造的跟真的一样，从而能够骗过discriminator，即生成样本并使它看上去好像来自于真实训练样本一样。



如下图中的左右两个场景：



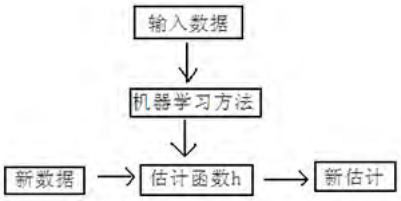
更多请参见此课程：《生成对抗网络班》。

学梵高作画的原理是啥？深度学习 DL应用 难
这里有篇如何做梵高风格画的实验教程《教你从头到尾利用DL学梵高作画：GTX 1070 cuda 8.0 tensorflow gpu版》，至于其原理请看这个视频：NeuralStyle艺术化图片（学梵高作画背后的原理）。

现在有 a 到 z 26 个元素， 编写程序打印 a 到 z 中任取 3 个元素的组合（比如 打印 a b c ， d y z等） 数理逻辑 排列组合 中
解析参考：http://blog.csdn.net/lvonve/article/details/53320680

说说梯度下降法。机器学习 ML基础 中

@LeftNotEasy，本题解析来源：http://www.cnblogs.com/LeftNotEasy/archive/2010/12/05/mathmatic_in_machine_learning_1_regression_and_gradient_descent.html下面是一个典型的机器学习的过程，首先给出一个输入数据，我们的算法会通过一系列的过程得到一个估计的函数，这个函数有能力对没有见过的新数据给出一个新的估计，也被称为构建一个模型。



我们用X1，X2..Xn 去描述feature里面的分量，比如x1=房间的面积，x2=房间的朝向等等，我们可以做出一个估计函数：

$$h(x) = h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2$$

θ在这儿称为参数，在这儿的意思是调整feature中每个分量的影响力，就是到底是房屋的面积更重要还是房屋的地段更重要。为了如果我们令X0 = 1，就可以用向量的方式来表示了：

$$h_\theta(x) = \theta^T X$$

我们程序也需要一个机制去评估我们θ是否比较好，所以说需要对我们做出的h函数进行评估，一般这个进行评估的函数称为损失函数（loss function），描述h函数不好的程度，在下面，我们称这个函数为J函数

在这儿我们可以做出下面的一个损失函数：

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

$$\min_\theta J_\theta$$

换言之，我们把对x(i)的估计值与真实值y(i)差的平方和作为损失函数，前面乘上的1/2是为了在求导的时候，这个系数就不见了。

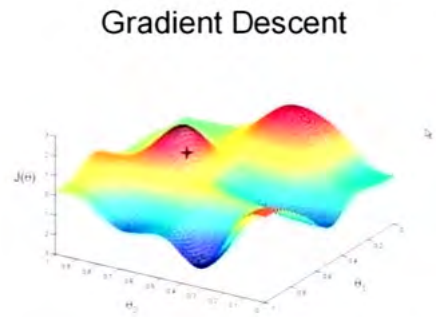
如何调整θ以使得J(θ)取得最小值有很多方法，其中有最小二乘法(min square)，是一种完全是数学描述的方法，另外一种就是梯度下降法。

梯度下降法的算法流程如下：

1）首先对θ赋值，这个值可以是随机的，也可以让θ是一个全零的向量。

2）改变θ的值，使得J(θ)按梯度下降的方向进行减少。

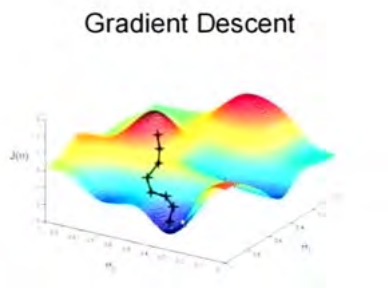
为了描述的更清楚，给出下面的图：



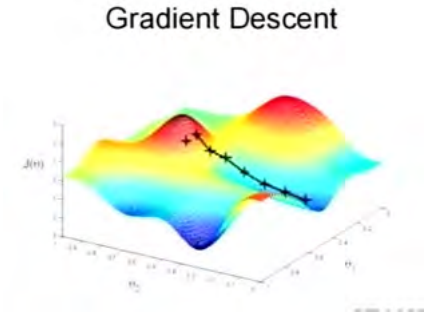
这是一个表示参数θ与误差函数J(θ)的关系图，红色的部分是表示J(θ)有着比较高的取值，我们需要的是，能够让J(θ)的值尽量的低，也就是达到深蓝色的部分。θ0，θ1表示θ向量的两个维度。

在上面提到梯度下降法的第一步是给θ给一个初值，假设随机给的初值是在图上的十字点。

然后我们将θ按照梯度下降的方向进行调整，就会使得J(θ)往更低的方向进行变化，如下图所示，算法的结束将是在θ下降到无法继续下降为止。



当然，可能梯度下降的最终点并非是全局最小点，即也可能是一个局部最小点，如下图所示：



上面这张图就是描述的一个局部最小点，这是我们重新选择了一个初始点得到的，看来我们这个算法将会在很大的程度上被初始点的选择影响而陷入局部最小点。

下面我将用一个例子描述一下梯度减少的过程，对于我们的函数J(θ)求偏导J：

$$\frac{\partial}{\partial\theta}J(\theta)=\frac{\partial}{\partial\theta}\frac{1}{2}\sum_{i=1}^{m}(h_\theta(x)-y)^2=(h_\theta(x)-y)x^{(i)}$$

下面是更新的过程，也就是θi会向着梯度最小的方向进行减少。θi表示更新之前的值，-后面的部分表示按梯度方向减少的量，α表示步长，也就是每次按照梯度减少的方向变化多少。

$$\theta_i=\theta_i-\alpha\frac{\partial}{\partial\theta}J(\theta)=\theta_i-\alpha(h_\theta(x)-y)x^{(i)}$$

一个很重要的地方值得注意的是，梯度是有方向的，对于一个向量θ，每一维分量θi都可以求出一个梯度的方向，我们就可以找到一个整体的方向，在变化的时候，我们就朝着下降最多的方向进行变化就可以达到一个最小点，不管它是局部的还是全局的。

用更简单的数学语言进行描述步骤2）是这样的：

$$\nabla_\theta J=\begin{bmatrix}\frac{\partial}{\partial\theta_0}J\\ \vdots\\ \frac{\partial}{\partial\theta_n}J\end{bmatrix}$$

$$\theta=\theta-\alpha\nabla_\theta J$$

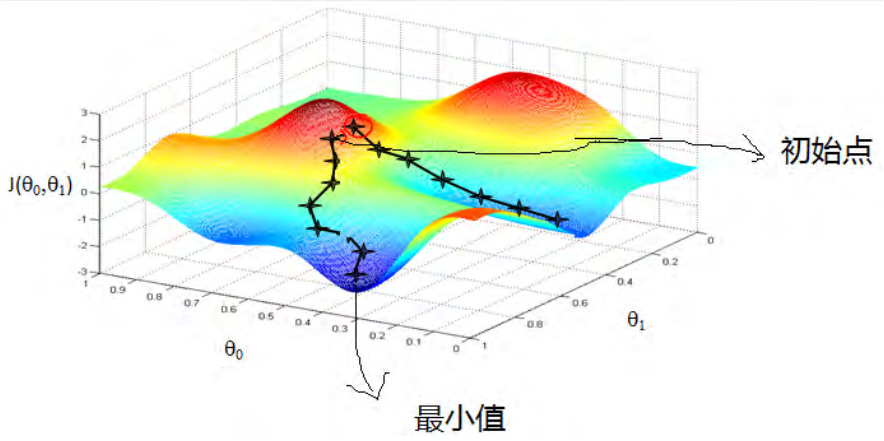
梯度下降法找到的一定是下降最快的方向么？机器学习 ML基础 中

梯度下降法并不是下降最快的方向，它只是目标函数在当前的点的切平面（当然高维问题不能叫平面）上下降最快的方向。在practical implementation中，牛顿方向（考虑海森矩阵）才一般被认为是下降最快的方向，可以达到superlinear的收敛速度。梯度下降类的算法的收敛速度一般是linear甚至sublinear的（在某些带复杂约束的问题）。by 林小溪（https://www.zhihu.com/question/30672734/answer/139689869）。

一般解释梯度下降，会用下山来举例。假设你现在在山顶处，必须抵达山脚下（也就是山谷最低处）的湖泊。但让人头疼的是，你的双眼被蒙上了无法辨别前进方向。换句话说，你不再能够一眼看出哪条路径是最快的下山路径，如下图（图片来源：http://blog.csdn.net/wemedia/details.html?id=45460）：

最好的办法就是走一步算一步，先用脚向四周各个方向都迈出一步，试探一下周围的地势，用脚感觉下哪个方向是下降最大的方向。换言之，每走到一个位置的时候，求解当前位置的梯度，沿着梯度的负方向（当前最陡峭的位置向下）走一步。就这样，每要走一步都根据上一步所在的位置选择当前最陡峭最快下山的方向走下一步，一步步走下去，一直走到我们感觉已经到了山脚。

当然这样走下去，我们走到的可能并不一定是真正的山脚，而只是走到了某一个局部的山峰低处。换句话说，梯度下降不一定能够找到全局的最优解，也有可能只是一个局部最优解。当然，如果损失函数是凸函数，梯度下降法得到的解就一定是全局最优解。



@zbxzc（http://blog.csdn.net/u014568921/article/details/44856915）：更进一步，我们来定义输出误差，即对于任意一组权值向量，那它得到的输出和我们预想的输出之间的误差值。定义误差的方法很多，不同的误差计算方法可以得到不同的权值更新法则，这里我们先用这样的定义：

$$E(\vec{w}) = \frac{1}{2}\sum_{d\in D}(t_d - o_d)^2$$

上面公式中D代表了所有的输入实例，或者说是样本，d代表了一个样本实例，od表示感知器的输出，td代表我们预想的输出。
这样，我们的目标就明确了，就是想找到一组权值让这个误差的值最小，显然我们用误差对权值求导将是一个很好的选择，导数的意义是提供了一个方向，沿着这个方向改变权值，将会让总的误差变大，更形象的叫它为梯度。

$$\nabla E(w_i) = \frac{\partial E}{\partial w} = \frac{1}{2}\frac{\partial \sum_{d\in D}(t_d - o_d)^2}{\partial w_i} = \frac{1}{2}\sum_{d\in D}\frac{\partial (t_d - o_d)^2}{\partial w_i}$$

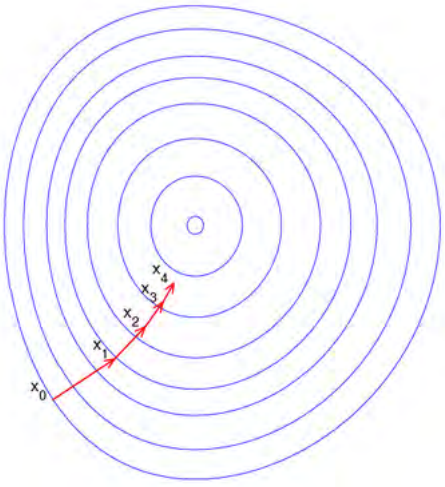
既然梯度确定了E最陡峭的上升的方向，那么梯度下降的训练法则是：

$$\vec{w_i} \leftarrow \vec{w_i} + \Delta\vec{w_i}\text{，其中}\Delta\vec{w_i} = -\eta\frac{\partial E}{\partial w_i}$$

梯度上升和梯度下降其实是一个思想，上式中权值更新的+号改为-号也就是梯度上升了。梯度上升用来求函数的最大值，梯度下降求最小值。
这样每次移动的方向确定了，但每次移动的距离却不知道。这个可以由步长（也称学习率）来确定，记为α。这样权值调整可表示为：

$$\boldsymbol{w} := \boldsymbol{w} + \alpha\nabla_{\mathbf{w}} f(w)$$

总之，梯度下降法的优化思想是用当前位置负梯度方向作为搜索方向，因为该方向为当前位置的最快下降方向，所以也被称为是“最速下降法”。最速下降法越接近目标值，步长越小，前进越慢。梯度下降法的搜索迭代示意图如下图所示：



正因为梯度度下降法在接近最优解的区域收敛速度明显变慢，所以利用梯度下降法求解需要很多次的迭代。在机器学习中，基于基本的梯度下降法发展了两种梯度下降方法，分别为随机梯度下降法和批量梯度下降法。by@wtq1993，http://blog.csdn.net/wtq1993/article/details/51607040

随机梯度下降

普通的梯度下降算法在更新回归系数时要遍历整个数据集，是一种批处理方法，这样训练数据特别忙庞大时，可能出现如下问题：

1）收敛过程可能非常慢；

2）如果误差曲面上有多个局极小值，那么不能保证这个过程会找到全局最小值。

为了解决上面的问题，实际中我们应用的是梯度下降的一种变体被称为随机梯度下降。

上面公式中的误差是针对于所有训练样本而得到的，而随机梯度下降的思想是根据每个单独的训练样本来更新权值，这样我们上面的梯度公式就变成了：

$$\frac{\partial E}{\partial w_i} = \frac{1}{2}\frac{\partial (t-o)^2}{\partial w_i} = -(t-o)\frac{\partial o}{\partial w_i}$$

经过推导后，我们就可以得到最终的权值更新的公式：

$$w_i = w_i + \Delta w_i$$
$$\delta = (t-o)o(1-o)$$
$$\Delta w_i = \eta \delta x_i$$

有了上面权重的更新公式后，我们就可以通过输入大量的实例样本，来根据我们预期的结果不断地调整权值，从而最终得到一组权值使得我们的算法能够对一个新的样本输入得到正确的或无限接近的结果。

这里做一个对比

设代价函数为

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

批量梯度下降

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

参数更新为：

$$\theta_j^{'} = \theta_j + \frac{1}{m}\sum_{i=1}^{m}(y^i - h_\theta(x^i))x_j^i$$

i是样本编号下标，j是样本维数下标，m为样例数目，n为特征数目。所以更新一个θj需要遍历整个样本集

Repeat until convergence {

$\theta_j := \theta_j + \alpha \sum_{i=1}^{m}\left(y^{(i)} - h_\theta(x^{(i)})\right)x_j^{(i)}$ (for every j).

}

随机梯度下降

参数更新为：

$$\theta_j^{'} = \theta_j + (y^i - h_\theta(x^i))x_j^i$$

i是样本编号下标，j是样本维数下标，m为样例数目，n为特征数目。所以更新一个θj只需要一个样本就可以。

Loop {

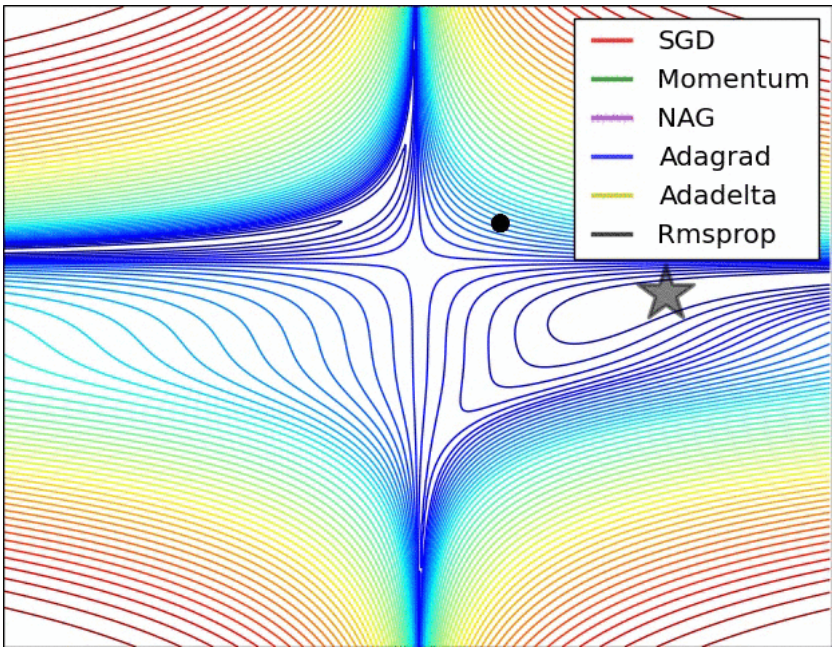
for i=1 to m, {

$\theta_j := \theta_j + \alpha \left(y^{(i)} - h_\theta(x^{(i)})\right)x_j^{(i)}$ (for every j).

}

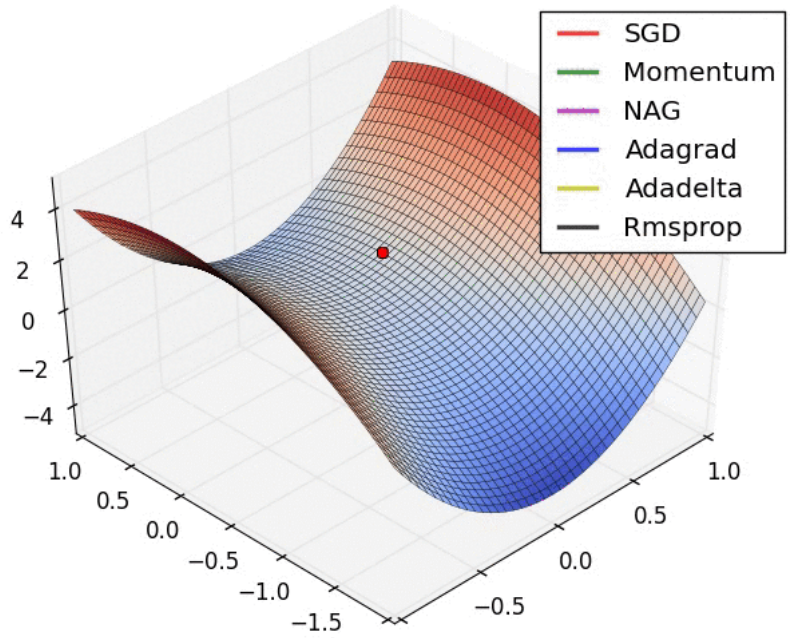
}

下面两幅图可以很形象的对比各种优化方法（图来源：http://sebastianruder.com/optimizing-gradient-descent/）：



SGD各优化方法在损失曲面上的表现

从上图可以看出， Adagrad、Adadelta与RMSprop在损失曲面上能够立即转移到正确的移动方向上达到快速的收敛。而Momentum 与NAG会导致偏离(off-track)。同时NAG能够在偏离之后快速修正其路线，因为其根据梯度修正来提高响应性。



SGD各优化方法在损失曲面鞍点处上的表现

牛顿法和梯度下降法有什么不同。机器学习 ML基础 中

@wtq1993，http://blog.csdn.net/wtq1993/article/details/51607040

1）牛顿法（Newton's method）

牛顿法是一种在实数域和复数域上近似求解方程的方法。方法使用函数$f(x)$的泰勒级数的前面几项来寻找方程$f(x)=0$的根。牛顿法最大的特点就在于它的收敛速度很快。

具体步骤：

首先，选择一个接近函数 $f(x)$零点的 x_0，计算相应的 $f(x_0)$ 和切线斜率$f'(x_0)$（这里f'表示函数 f 的导数）。然后我们计算穿过点$(x_0, f(x_0))$ 并且斜率为$f'(x_0)$的直线和 x 轴的交点的x坐标，也就是求如下方程的解：

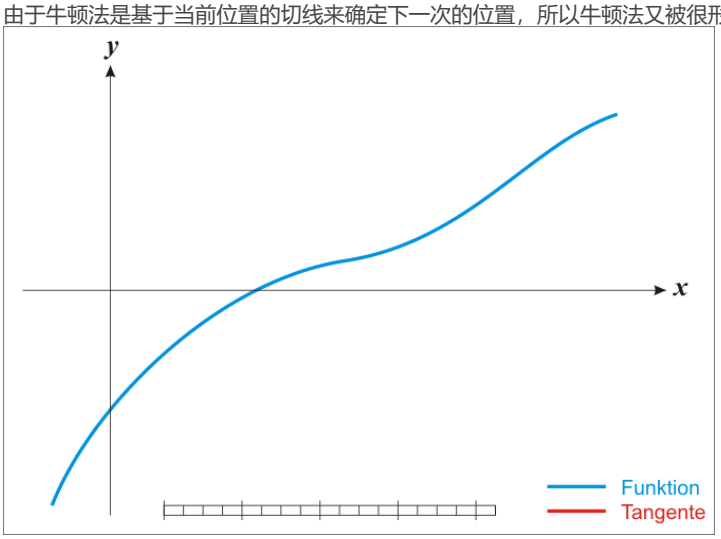
$$x \cdot f'(x_0) + f(x_0) - x_0 \cdot f'(x_0) = 0$$

我们将新求得的点的 x 坐标命名为x_1，通常x_1会比x_0更接近方程$f(x)=0$的解。因此我们现在可以利用x_1开始下一轮迭代。迭代公式可化简为如下所示：

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$$

已经证明，如果f ' 是连续的，并且待求的零点x是孤立的，那么在零点x周围存在一个区域，只要初始值x_0位于这个邻近区域内，那么牛顿法必定收敛。并且，如果f ' (x)不为0, 那么牛顿法将具有平方收敛的性能. 粗略的说，这意味着每迭代一次，牛顿法结果的有效数字将增加一倍。

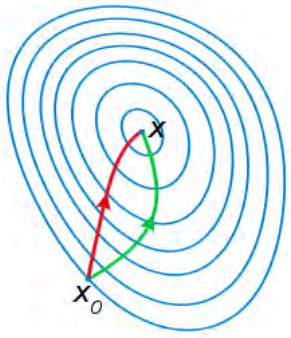
由于牛顿法是基于当前位置的切线来确定下一次的位置，所以牛顿法又被很形象地称为是"切线法"。牛顿法的搜索路径（二维情况）如下图所示：



关于牛顿法和梯度下降法的效率对比：

a）从收敛速度上看 ，牛顿法是二阶收敛，梯度下降是一阶收敛，前者牛顿法收敛速度更快。但牛顿法仍然是局部算法，只是在局部上看的更细致，梯度法仅考虑方向，牛顿法不但考虑了方向还兼顾了步子的大小，其对步长的估计使用的是二阶逼近。

b）根据wiki上的解释，从几何上说，牛顿法就是用一个二次曲面去拟合你当前所处位置的局部曲面，而梯度下降法是用一个平面去拟合当前的局部曲面，通常情况下，二次曲面的拟合会比平面更好，所以牛顿法选择的下降路径会更符合真实的最优下降路径。



注：红色的牛顿法的迭代路径，绿色的是梯度下降法的迭代路径。

牛顿法的优缺点总结：

优点：二阶收敛，收敛速度快；

缺点：牛顿法是一种迭代算法，每一步都需要求解目标函数的Hessian矩阵的逆矩阵，计算比较复杂。

什么是拟牛顿法（Quasi-Newton Methods）？ 机器学习 ML基础 中

@wtq1993，http://blog.csdn.net/wtq1993/article/details/51607040

拟牛顿法是求解非线性优化问题最有效的方法之一，于20世纪50年代由美国Argonne国家实验室的物理学家W.C.Davidon所提出来。Davidon设计的这种算法在当时看来是非线性优化领域最具创造性的发明之一。不久R. Fletcher和M. J. D. Powell证实了这种新的算法远比其他方法快速和可靠，使得非线性优化这门学科在一夜之间突飞猛进。

拟牛顿法的本质思想是改善牛顿法每次需要求解复杂的Hessian矩阵的逆矩阵的缺陷，它使用正定矩阵来近似Hessian矩阵的逆，从而简化了运算的复杂度。拟牛顿法和最速下降法一样只要求每一步迭代时知道目标函数的梯度。通过测量梯度的变化，构造一个目标函数的模型使之足以产生超线性收敛性。这类方法大大优于最速下降法，尤其对于困难的问题。另外，因为拟牛顿法不需要二阶导数的信息，所以有时比牛顿法更为有效。如今，优化软件中包含了大量的拟牛顿算法用来解决无约束，约束，和大规模的优化问题。

具体步骤：

拟牛顿法的基本思想如下。首先构造目标函数在当前迭代x_k的二次模型：

$$m_k(p) = f(x_k) + \nabla f(x_k)^T p + \frac{p^T B_k p}{2}$$

$$p_k = -B_k^{-1} \nabla f(x_k)$$

这里B_k是一个对称正定矩阵，于是我们取这个二次模型的最优解作为搜索方向，并且得到新的迭代点：

$$x_{k+1} = x_k + a_k p_k$$

其中我们要求步长a_k

满足Wolfe条件。这样的迭代与牛顿法类似，区别就在于用近似的Hessian矩阵B_k

代替真实的Hessian矩阵。所以拟牛顿法最关键的地方就是每一步迭代中矩阵B_k

的更新。现在假设得到一个新的迭代x_{k+1}，并得到一个新的二次模型：

$$m_{k+1}(p) = f(x_{k+1}) + \nabla f(x_{k+1})^T p + \frac{p^T B_{k+1} p}{2}$$

我们尽可能地利用上一步的信息来选取Bk。具体地，我们要求

$$\nabla f(x_{k+1}) - \nabla f(x_k) = a_k B_{k+1} p_k$$

从而得到

$$B_{k+1}(x_{k+1} - x_k) = \nabla f(x_{k+1}) - \nabla f(x_k)$$

这个公式被称为割线方程。常用的拟牛顿法有DFP算法和BFGS算法。

请说说随机梯度下降法的问题和挑战？机器学习 ML基础 中

梯度下降法：

如果J(θ)是一个多元函数，在θ_0点附近对J(θ)做线性逼近

$$J(\theta_0 + \Delta_\theta) = J(\theta_0) + \Delta_\theta^T \cdot \nabla J(\theta_0) + o(|\Delta_\theta|)$$

这个线性逼近不能告诉我们极值点在什么地方，他只能告诉我们极值点在什么方向.所以我们只能选取一个比较“小”的学习率η来沿着这个方向走下去，并得到梯度下降法的序列：

$$\theta_n = \theta_{n-1} - \eta_{n-1} \nabla J(\theta_{n-1})$$

困难：

1. **梯度的计算**

   在机器学习和统计参数估计问题中目标函数经常是求和函数的形式

   $$J_X(\theta) = \sum_i J_{x_i}(\theta)$$

   其中每一个函数$J_{x_i}(\theta)$都对应于一个样本x_i。

   - **当样本量极大时，梯度的计算就变得非常耗时耗力。**

2. **学习率的选择**

   学习率选择过小会导致算法收敛太慢，学习率选择过大容易导致算法不收敛。

   - **如何选择学习率需要 具体问题具体分析**

Chapter 2. 随机梯度下降法

随机梯度下降法主要为了解决第一个问题：梯度计算

由于随机梯度下降法的引入，我们通常将梯度下降法分为三种类型：

1. **批梯度下降法(GD)**
   原始的梯度下降法
2. **随机梯度下降法(SGD)**
   每次梯度计算只使用一个样本
   - 避免在类似样本上计算梯度造成的冗余计算
   - 增加了跳出当前的局部最小值的潜力
   - 在逐渐缩小学习率的情况下，有与批梯度下降法类似的收敛速度
3. **小批量随机梯度下降法(Mini Batch SGD)**
   每次梯度计算使用一个小批量样本
   - 梯度计算比单样本更加稳定
   - 可以很好的利用现成的高度优化的矩阵运算工具

注意：神经网络训练的文献中经常把 Mini Batch SGD 称为 SGD

Chapter 4. 随机梯度下降法的优化算法

为什么不用牛顿法？

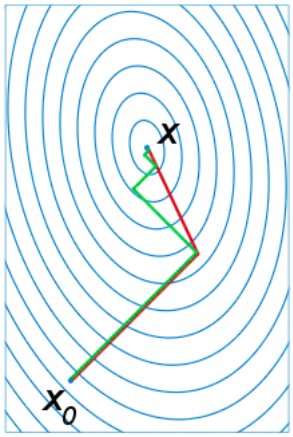
- 牛顿法要求计算目标函数的二阶导数(Hessian matrix)，在高维特征情形下这个**矩阵非常巨大**，计算和存储都成问题
- 在使用小批量情形下，牛顿法对于二阶导数的估计**噪音太大**
- 在目标函数非凸时，牛顿法更容易收到**鞍点**甚至最大值点的吸引

那到底如何优化随机梯度法呢？详情请点击：论文公开课第一期：详解梯度下降等各类优化算法（含视频和PPT下载）。

说说共轭梯度法? 机器学习 ML基础 中
@wtq1993，http://blog.csdn.net/wtq1993/article/details/51607040
共轭梯度法是介于梯度下降法（最速下降法）与牛顿法之间的一个方法，它仅需利用一阶导数信息，但克服了梯度下降法收敛慢的缺点，又避免了牛

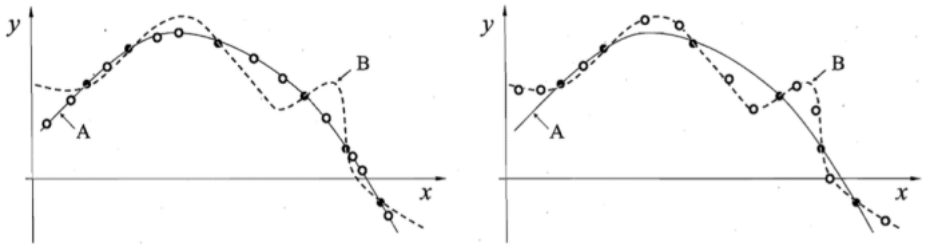
顿法需要存储和计算Hessian矩阵并求逆的缺点，共轭梯度法不仅是解决大型线性方程组最有用的方法之一，也是解大型非线性最优化最有效的算法之一。在各种优化算法中，共轭梯度法是非常重要的一种。其优点是所需存储量小，具有逐步收敛性，稳定性高，而且不需要任何外来参数。

下图为共轭梯度法和梯度下降法搜索最优解的路径对比示意图：



注：绿色为梯度下降法，红色代表共轭梯度法

对所有优化问题来说, 有没有可能找到比现在已知算法更好的算法？机器学习 ML基础 中
@抽象猴，来源：https://www.zhihu.com/question/41233373/answer/145404190



(a) A 优于 B　　(b) B 优于 A

图 1.4　没有免费的午餐. (黑点: 训练样本; 白点: 测试样本)

没有免费的午餐定理：
对于训练样本（黑点），不同的算法A/B在不同的测试样本（白点）中有不同的表现，这表示：对于一个学习算法A，若它在某些问题上比学习算法 B更好，则必然存在一些问题，在那里B比A好。
也就是说：对于所有问题，无论学习算法A多聪明，学习算法 B多笨拙，它们的期望性能相同。
但是：没有免费午餐定力假设所有问题出现几率相同，实际应用中，不同的场景，会有不同的问题分布，所以，在优化算法时，针对具体问题进行分析，是算法优化的核心所在。
什么最小二乘法？机器学习 ML基础 中

我们口头中经常说：一般来说，平均来说。如平均来说，不吸烟的健康优于吸烟者，之所以要加“平均”二字，是因为凡事皆有例外，总存在某个特别的人他吸烟但由于经常锻炼所以他的健康状况可能会优于他身边不吸烟的朋友。而最小二乘法的一个最简单的例子便是算术平均。

最小二乘法（又称最小平方法）是一种数学优化技术。它通过最小化误差的平方和寻找数据的最佳函数匹配。利用最小二乘法可以简便地求得未知的数据，并使得这些求得的数据与实际数据之间误差的平方和为最小。用函数表示为：

$$\min_{\vec{x}} \sum_{i=1}^{n} (y_m - y_i)^2.$$

使误差「所谓误差，当然是观察值与实际真实值的差量」平方和达到最小以寻求估计值的方法，就叫做最小二乘法，用最小二乘法得到的估计，叫做最小二乘估计。当然，取平方和作为目标函数只是众多可取的方法之一。

最小二乘法的一般形式可表示为：

$$\min_{\vec{x}} \|\vec{y}_m(\vec{x}) - \vec{y}\|_2 \ .$$

有效的最小二乘法是勒让德在 1805 年发表的，基本思想就是认为测量中有误差，所以所有方程的累积误差为

$$\text{累积误差} = \sum (\text{观测值} - \text{理论值})^2$$

我们求解出导致累积误差最小的参数即可：

$$\begin{aligned} \hat{\beta} &= argmin_{\beta} \sum_{i=1}^{n} e_i^2 \\ &= argmin_{\beta} \sum_{i=1}^{n} [y_i - (\beta_0 + \beta_1 x_{1i} + \cdots + \beta_p x_{pi})]^2 \end{aligned}$$

勒让德在论文中对最小二乘法的优良性做了几点说明：

- 最小二乘使得误差平方和最小，并在各个方程的误差之间建立了一种平衡，从而防止某一个极端误差取得支配地位
- 计算中只要求偏导后求解线性方程组，计算过程明确便捷
- 最小二乘可以导出算术平均值作为估计值

对于最后一点，从统计学的角度来看是很重要的一个性质。推理如下：假设真值为 , 为n次测量值, 每次测量的误差为，按最小二乘法，误差累积为

$$L(\theta) = \sum_{i=1}^{n} e_i^2 = \sum_{i=1}^{n} (x_i - \theta)^2$$

求解θ 使$L(\theta)$达到最小，正好是算术平均 $\bar{x} = \frac{\sum_{i-1}^{n} x_i}{n}$ 。

由于算术平均是一个历经考验的方法，而以上的推理说明，算术平均是最小二乘的一个特例，所以从另一个角度说明了最小二乘方法的优良性，使我们对最小二乘法更加有信心。
最小二乘法的原理之一：当估计误差服从正态分布时，最小二乘法等同于极大似然估计。 如果 y = f(x) + e, 其中y 是目标值，f(x)为估计值，e为误差项。如果e服从正态分布，那么 细节可以看：https://www.zhihu.com/question/20447622/answer/209839263，而由于中心极限定理的原因，很多误差分布确实服从正态分布，这也是最小二乘法能够十分有效的一个原因。

最小二乘法发表之后很快得到了大家的认可接受，并迅速的在数据分析实践中被广泛使用。不过历史上又有人把最小二乘法的发明归功于高斯，这又是怎么一回事呢。高斯在1809年也发表了最小二乘法，并且声称自己已经使用这个方法多年。高斯发明了小行星定位的数学方法，并在数据分析中使用最小二乘方法进行计算，准确的预测了谷神星的位置。
对了，最小二乘法跟SVM有什么联系呢？请参见《支持向量机通俗导论（理解SVM的三层境界）》。

看你T恤上印着：人生苦短，我用Python，你可否说说Python到底是什么样的语言？你可以比较其他技术或者语言来回答你的问题。
Python Python语言 易
@David 9，http://nooverfit.com/wp/15%E4%B8%AA%E9%87%8D%E8%A6%81python%E9%9D%A2%E8%AF%95%E9%A2%98-%E6%B5%8B%E6%B5%8B%E4%BD%A0%E9%80%82%E4%B8%8D%E9%80%82%E5%90%88%E5%81%9Apython%EF%BC%9F/
这里是一些关键点：Python是解释型语言。这意味着不像C和其他语言，Python运行前不需要编译。其他解释型语言包括PHP和Ruby。

Python是动态类型的，这意味着你不需要在声明变量时指定类型。你可以先定义x=111，然后 x="I' m a string"。
Python是面向对象语言，所有允许定义类并且可以继承和组合。Python没有访问访问标识如在C++中的public, private，这就非常信任程序员的素质，相信每个程序员都是“成人”了~
在Python中，函数是一等公民。这就意味着它们可以被赋值，从其他函数返回值，并且传递函数对象。类不是一等公民。
写Python代码很快，但是跑起来会比编译型语言慢。幸运的是，Python允许使用C扩展写程序，所以瓶颈可以得到处理。Numpy库就是一个很好例子，因为很多代码不是Python直接写的，所以运行很快。
Python使用场景很多 – web应用开发、大数据应用、数据科学、人工智能等等。它也经常被看做“胶水”语言，使得不同语言间可以衔接上。
Python能够简化工作，使得程序员能够关心如何重写代码而不是详细看一遍底层实现。

@July：Python目前早已成为AI时代的第一语言，为帮助大家更好的学习Python语言、数据分析、爬虫等相关知识，七月在线特开一系列Python课程，有需要的亲们可以看下，比如《Python数据分析集训营》。
Python是如何进行内存管理的？Python Python基础 中
@Tom_junsong，来源：http://www.cnblogs.com/tom-gao/p/6645859.html
答:从三个方面来说,一对象的引用计数机制,二垃圾回收机制,三内存池机制
一、对象的引用计数机制
Python内部使用引用计数，来保持追踪内存中的对象，所有对象都有引用计数。
引用计数增加的情况：
1，一个对象分配一个新名称
2，将其放入一个容器中（如列表、元组或字典）
引用计数减少的情况：
1，使用del语句对对象别名显示的销毁
2，引用超出作用域或被重新赋值
sys.getrefcount( )函数可以获得对象的当前引用计数
多数情况下，引用计数比你猜测得要大得多。对于不可变数据（如数字和字符串），解释器会在程序的不同部分共享内存，以便节约内存。
二、垃圾回收
1，当一个对象的引用计数归零时，它将被垃圾收集机制处理掉。
2，当两个对象a和b相互引用时，del语句可以减少a和b的引用计数，并销毁用于引用底层对象的名称。然而由于每个对象都包含一个对其他对象的应用，因此引用计数不会归零，对象也不会销毁。（从而导致内存泄露）。为解决这一问题，解释器会定期执行一个循环检测器，搜索不可访问对象的循环并删除它们。
三、内存池机制
Python提供了对内存的垃圾收集机制，但是它将不用的内存放到内存池而不是返回给操作系统。
1，Pymalloc机制。为了加速Python的执行效率，Python引入了一个内存池机制，用于管理对小块内存的申请和释放。
2，Python中所有小于256个字节的对象都使用pymalloc实现的分配器，而大的对象则使用系统的malloc。

3，对于Python对象，如整数，浮点数和List，都有其独立的私有内存池，对象间不共享他们的内存池。也就是说如果你分配又释放了大量的整数，用于缓存这些整数的内存就不能再分配给浮点数。
请写出一段Python代码实现删除一个list里面的重复元素。Python Python开发 中
@Tom_junsong，http://www.cnblogs.com/tom-gao/p/6645859.html
答：
1,使用set函数，set(list)
2，使用字典函数，
>>>a=[1,2,4,2,4,5,6,5,7,8,9,0]
>>> b={}
>>>b=b.fromkeys(a)
>>>c=list(b.keys())
>>> c
编程用sort进行排序，然后从最后一个元素开始判断？Python Python开发 中
a=[1,2,4,2,4,5,7,10,5,5,7,8,9,0,3]
@Tom_junsong，http://www.cnblogs.com/tom-gao/p/6645859.html
a.sort()
last=a[-1]
for i inrange(len(a)-2,-1,-1):
if last==a[i]:
del a[i]
else:last=a[i]
print(a)
Python里面如何生成随机数？Python Python开发 中
@Tom_junsong，http://www.cnblogs.com/tom-gao/p/6645859.html
答：random模块
随机整数：random.randint(a,b)：返回随机整数x,a<=x<=b
random.randrange(start,stop,[,step])：返回一个范围在(start,stop,step)之间的随机整数，不包括结束值。
随机实数：random.random( ):返回0到1之间的浮点数
random.uniform(a,b):返回指定范围内的浮点数。更多Python笔试面试题请看：http://python.jobbole.com/85231/
说说常见的损失函数？机器学习 ML基础 易

对于给定的输入X，由f(X)给出相应的输出Y，这个输出的预测值f(X)与真实值Y可能一致也可能不一致（要知道，有时损失或误差是不可避免的），用一个损失函数来度量预测错误的程度。损失函数记为L(Y, f(X))。

常用的损失函数有以下几种（基本引用自《统计学习方法》）：

（1）0-1 损失函数

$$L(Y, f(X)) = \begin{cases} 1, Y \neq f(X) \\ 0, Y = f(X) \end{cases}$$

（2）平方损失函数

$$L(Y, f(X)) = (Y - f(X))^2$$

（3）绝对损失函数

$$L(Y, f(X)) = |Y - f(X)|$$

（4）对数损失函数

$$L(Y, P(Y|X)) = -\log P(Y|X)$$

给定一个训练数据集

$$T = \{(x_1, y_1), (x_2, y_2), \ldots (x_i, y_i) \ldots, (x_N, y_N)\}$$

模型 f(X)关于训练数据集的平均损失称为经验风险，如下：

$$R_{emp}(f) = \frac{1}{N} \sum_{i=1}^{N} L(y_i, f(x_i))$$

关于如何选择模型，监督学习有两种策略：经验风险最小化和结构风险最小化。

经验风险最小化的策略认为，经验风险最小的模型就是最优的模型，则按照经验风险最小化求最优模型就是求解如下最优化问题：

$$\min_{f \in F} \frac{1}{N} \sum_{i=1}^{N} L(y_i, f(x_i)) \qquad (1)$$

当样本容量很小时，经验风险最小化的策略容易产生过拟合的现象。结构风险最小化可以防止过拟合。结构风险是在经验风险的基础上加上表示模型复杂度的正则化项或罚项，结构风险定义如下：

$$R_{srm}(f) = \frac{1}{N} \sum_{i=1}^{N} L(y_i, f(x_i)) + \lambda J(f)$$

其中 J(f)为模型的复杂度，模型 f 越复杂，J(f)值就越大，模型越简单，J(f)值就越小，也就是说 J(f)是对复杂模型的惩罚。$\lambda \geq 0$ 是系数，用以权衡经验风险和模型复杂度。结构风险最小化的策略认为结构风险最小的模型是最优的模型，所以求最优的模型就是求解下面的最优化问题：

$$\min_{f \in F} \frac{1}{N} \sum_{i=1}^{N} L(y_i, f(x_i)) + \lambda J(f) \qquad (2)$$

这样，监督学习问题就变成了经验风险或结构风险函数的最优化问题，如式（1）和式（2）。

如此，SVM有第二种理解，即最优化+损失最小，或如@夏粉_百度所说“可从损失函数和优化算法角度看SVM，boosting，LR等算法，可能会有不同收获”。关于SVM的更多理解请参考：支持向量机通俗导论（理解SVM的三层境界）

简单介绍下logistics回归？机器学习 ML模型 易

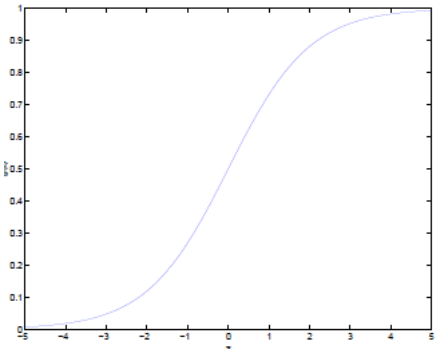
Logistic回归目的是从特征学习出一个0/1分类模型，而这个模型是将特性的线性组合作为自变量，由于自变量的取值范围是负无穷到正无穷。因此，使用logistic函数（或称作sigmoid函数）将自变量映射到(0,1)上，映射后的值被认为是属于y=1的概率。

假设函数

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

其中x是n维特征向量，函数g就是logistic函数。

而 $g(z) = \frac{1}{1 + e^{-z}}$ 的图像是



可以看到，将无穷映射到了(0,1)。

而假设函数就是特征属于y=1的概率。

$$\begin{aligned} P(y = 1 \mid x; \theta) &= h_\theta(x) \\ P(y = 0 \mid x; \theta) &= 1 - h_\theta(x) \end{aligned}$$

从而，当我们要判别一个新来的特征属于哪个类时，只需求$h_\theta(x)$即可，若$h_\theta(x)$大于0.5就是y=1的类，反之属于y=0类。

此外，$h_\theta(x)$只和$\theta^T x$有关，$\theta^T x$>0，那么$h_\theta(x) > 0.5$，而g(z)只是用来映射，真实的类别决定权还是在于$\theta^T x$。再者，当$\theta^T x \gg 0$时，$h_\theta(x)$=1，反之$h_\theta(x)$=0。如果我们只从$\theta^T x$出发，希望模型达到的目标就是让训练数据中y=1的特征$\theta^T x \gg 0$，而是y=0的特征$\theta^T x \ll 0$。Logistic回归就是要学习得到θ，使得正例的特征远远大于0，负例的特征远小于0，而且要在全部训练实例上达到这个目标。

接下来，尝试把logistic回归做个变形。首先，将使用的结果标签y = 0和y = 1替换为y = -1,y = 1，然后将$\theta^T x = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_n x_n$（$x_0 = 1$）中的$\theta_0$替换为b，最后将后面的$\theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_n x_n$替换为$\theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_n x_n$（即$w^T x$）。如此，则有了$\theta^T x = w^T x + b$。也就是说除了y由y=0变为y=-1外，线性分类函数跟logistic回归的形式化表示$h_\theta(x) = g(\theta^T x) = g(w^T x + b)$没区别。

进一步，可以将假设函数$h_{w,b}(x) = g(w^T x + b)$中的g(z)做一个简化，将其简单映射到y=-1和y=1上。映射关系如下：

$$g(z) = \begin{cases} 1, & z \geq 0 \\ -1, & z < 0 \end{cases}$$

最后补充一点，正态分布的极大似然估计 如果n维空间中两组点的分布各自服从多元正态分布，那么逻辑回归就等价于利用极大似然估计来对空间中的点进行分类。细节可以参考：http://blog.sciencenet.cn/blog-508318-633085.html。看你是搞视觉的，熟悉哪些CV框架，顺带聊聊CV最近五年的发展史如何？深度学习 DL应用 难

原英文：adeshpande3.github.io
作者：Adit Deshpande，UCLA CS研究生

译者：新智元闻菲、胡祥杰
译文链接：https://mp.weixin.qq.com/s?__biz=MzI3MTA0MTk1MA==&mid=2651986617&idx=1&sn=fddebd0f2968d66b7f424d6a435c84af&scene=0#wechat_redirect的本段结构如下：

AlexNet(2012年)

ZF Net(2013年)

VGG Net(2014年)

GoogLeNet (2015年)

微软 ResNet (2015年)

区域 CNN(R-CNN - 2013年，Fast R-CNN - 2015年，Faster R-CNN - 2015年)

生成对抗网络(2014年)

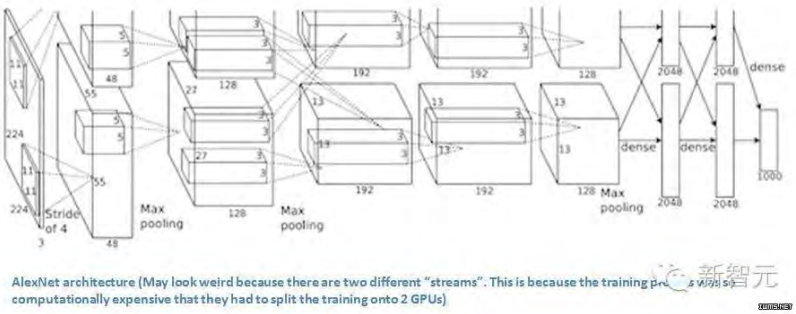
生成图像描述(2014年)

空间转化器网络(2015年)

AlexNet(2012 年)

一切都从这里开始(尽管有些人会说是Yann LeCun 1998年发表的那篇论文才真正开启了一个时代)。这篇论文，题目叫做“ImageNet Classification with Deep Convolutional Networks”，迄今被引用6184次，被业内普遍视为行业最重要的论文之一。Alex Krizhevsky、Ilya Sutskever和 Geoffrey Hinton创造了一个“大型的深度卷积神经网络”，赢得了2012 ILSVRC(2012年ImageNet 大规模视觉识别挑战赛)。稍微介绍一下，这个比赛被誉为计算机视觉的年度奥林匹克竞赛，全世界的团队相聚一堂，看看是哪家的视觉模型表现最为出色。2012年是CNN首次实现Top 5误差率15.4%的一年(Top 5误差率是指给定一张图像，其标签不在模型认为最有可能的5个结果中的几率)，当时的次优项误差率为26.2%。这个表现不用说震惊了整个计算机视觉界。可以说，是自那时起，CNN才成了家喻户晓的名字。

论文中，作者讨论了网络的架构(名为AlexNet)。相比现代架构，他们使用了一种相对简单的布局，整个网络由5层卷积层组成，最大池化层、退出层(dropout layer)和3层全卷积层。网络能够对1000种潜在类别进行分类。



AlexNet architecture (May look weird because there are two different “streams”. This is because the training pr… was split so computationally expensive that they had to split the training onto 2 GPUs)

AlexNet 架构：看上去有些奇怪，因为使用了两台GPU训练，因而有两股“流”。使用两台GPU训练的原因是计算量太大，只能拆开来。

要点

使用ImageNet数据训练网络，ImageNet数据库含有1500多万个带标记的图像，超过2.2万个类别。

使用ReLU代替传统正切函数引入非线性(ReLU比传统正切函数快几倍，缩短训练时间)。

使用了图像转化(image translation)、水平反射(horizontal reflection)和补丁提取(patch extraction)这些数据增强技术。

用dropout层应对训练数据过拟合的问题。

使用批处理随机梯度下降训练模型，注明动量衰减值和权重衰减值。

使用两台GTX 580 GPU，训练了5到6天

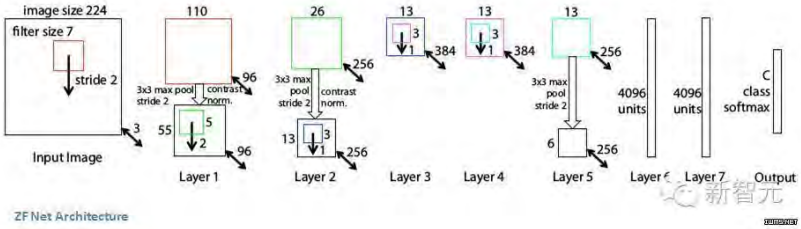
为什么重要?

Krizhevsky、Sutskever 和 Hinton 2012年开发的这个神经网络，是CNN在计算机视觉领域的一大亮相。这是史上第一次有模型在ImageNet 数据库表现这么好，ImageNet 数据库难度是出了名的。论文中提出的方法，比如数据增强和dropout，现在也在使用，这篇论文真正展示了CNN的优点，并且以破纪录的比赛成绩实打实地做支撑。

ZF Net(2013年)

2012年AlexNet出尽了风头，ILSVRC 2013就有一大批CNN模型冒了出来。2013年的冠军是纽约大学Matthew Zeiler 和 Rob Fergus设计的网络 ZF Net，错误率11.2%。ZF Net模型更像是AlexNet架构的微调优化版，但还是提出了有关优化性能的一些关键想法。还有一个原因，这篇论文写得非常好，论文作者花了大量时间阐释有关卷积神经网络的直观概念，展示了将滤波器和权重可视化的正确方法。

在这篇题为“Visualizing and Understanding Convolutional Neural Networks”的论文中，Zeiler和Fergus从大数据和GPU计算力让人们重拾对CNN的兴趣讲起，讨论了研究人员对模型内在机制知之甚少，一针见血地指出“发展更好的模型实际上是不断试错的过程”。虽然我们现在要比3年前知道得多一些了，但论文所提出的问题至今仍然存在!这篇论文的主要贡献在于提出了一个比AlexNet稍微好一些的模型并给出了细节，还提供了一些制作可视化特征图值得借鉴的方法。



要点

除了一些小的修改，整体架构非常类似AlexNet。

AlexNet训练用了1500万张图片，而ZFNet只用了130万张。

AlexNet在第一层中使用了大小为11×11的滤波器，而ZF使用的滤波器大小为7x7，整体处理速度也有所减慢。做此修改的原因是，对于输入数据来说，第一层卷积层有助于保留大量的原始象素信息。11×11的滤波器漏掉了大量相关信息，特别是因为这是第一层卷积层。

随着网络增大，使用的滤波器数量增多。

利用ReLU的激活函数，将交叉熵代价函数作为误差函数，使用批处理随机梯度下降进行训练。

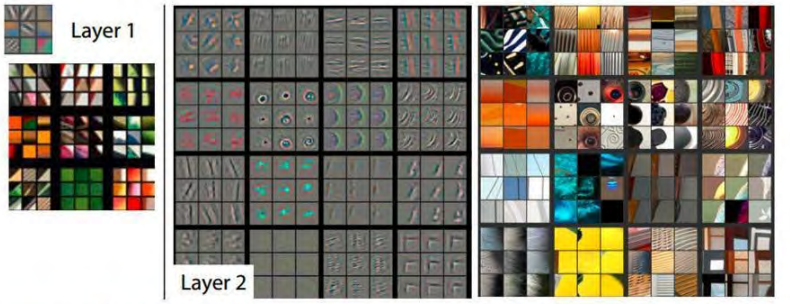
使用一台GTX 580 GPU训练了12天。

开发可视化技术“解卷积网络”(Deconvolutional Network)，有助于检查不同的特征激活和其对输入空间关系。名字之所以称为“deconvnet”，是因为它将特征映射到像素(与卷积层恰好相反)。

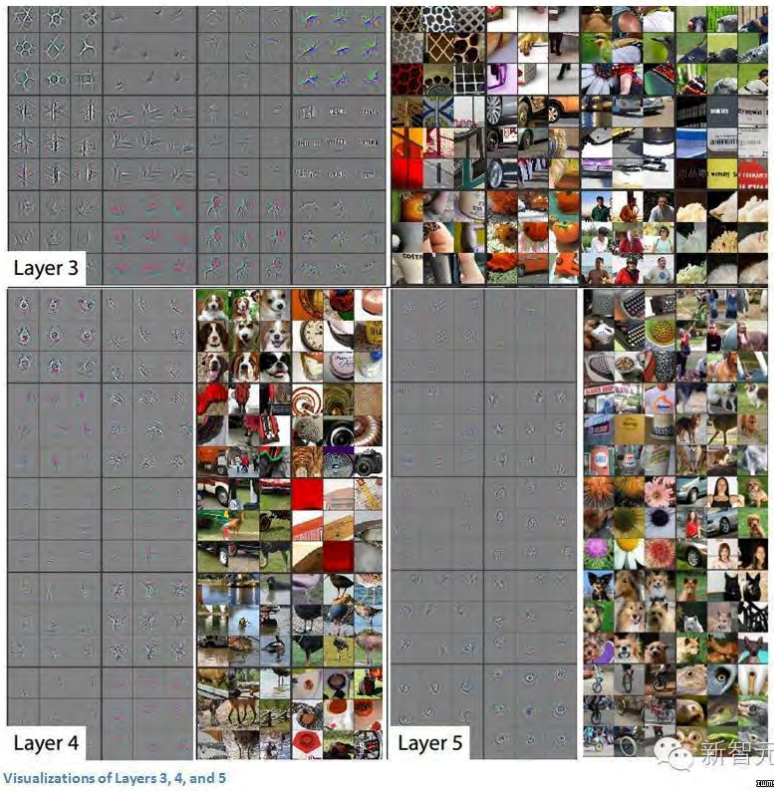
DeConvNet

DeConvNet工作的基本原理是，每层训练过的CNN后面都连一层“deconvet”，它会提供一条返回图像像素的路径。输入图像进入CNN之后，每一层都计算激活。然而向前传递。现在，假设我们想知道第4层卷积层某个特征的激活值，我们将保存这个特征图的激活值，并将这一层的其他激活值设为0，再将这张特征图作为输入送入deconvnet。Deconvnet与原来的CNN拥有同样的滤波器。输入经过一系列unpool(maxpooling倒过来)，修正，对前一层进行过滤操作，直到输入空间满。

这一过程背后的逻辑在于，我们想要知道是激活某个特征图的是什么结构。下面来看第一层和第二层的可视化。



ConvNet的第一层永远是低层特征检测器，在这里就是对简单的边缘、颜色进行检测。第二层就有比较圆滑的特征了。再来看第三、第四和第五层。



Visualizations of Layers 3, 4, and 5

这些层展示出了更多的高级特征，比如狗的脸和鲜花。值得一提的是，在第一层卷积层后面，我们通常会跟一个池化层将图像缩小(比如将 32x32x32 变为16x16x3)。这样做的效果是加宽了第二层看原始图像的视野。更详细的内容可以阅读论文。

为什么重要?

ZF Net不仅是2013年比赛的冠军，还对CNN的运作机制提供了极好的直观信息，展示了更多提升性能的方法。论文所描述的可视化方法不仅有助于弄清CNN的内在机理，也为优化网络架构提供了有用的信息。Deconv可视化方法和 occlusion 实验也让这篇论文成了我个人的最爱。

VGG Net(2015年)

简单、有深度，这就是2014年错误率7.3%的模型VGG Net(不是ILSVRC 2014冠军)。牛津大学的Karen Simonyan 和 Andrew Zisserman Main Points创造了一个19层的CNN，严格使用3x3的过滤器(stride =1，pad= 1)和2x2 maxpooling层(stride =2)。简单吧?

ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB imag)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

These different architectures of VGG Net. Configuration D produced the best results

要点

这里使用3x3的滤波器和AlexNet在第一层使用11x11的滤波器和ZF Net 7x7的滤波器作用完全不同。作者认为两个3x3的卷积层组合可以实现5x5的有效感受野。这就在保持滤波器尺寸较小的同时模拟了大型滤波器，减少了参数。此外，有两个卷积层就能够使用两层ReLU。

3卷积层具有7x7的有效感受野。

每个maxpool层后滤波器的数量增加一倍。进一步加强了缩小空间尺寸，但保持深度增长的想法。

图像分类和定位任务都运作良好。

使用Caffe工具包建模。

训练中使用scale jittering的数据增强技术。

每层卷积层后使用ReLU层和批处理梯度下降训练。

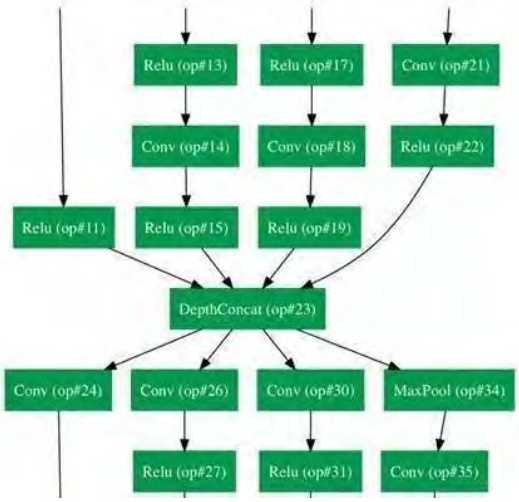
使用4台英伟达Titan Black GPU训练了两到三周。

为什么重要?

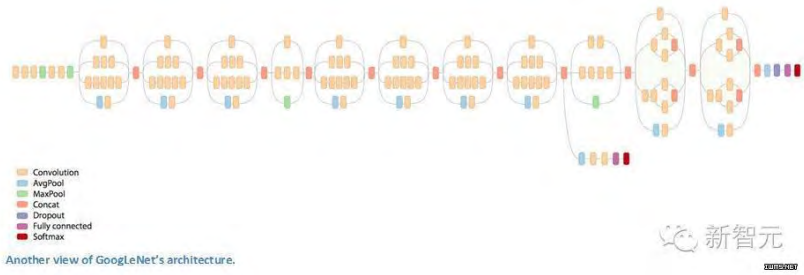
在我看来，VGG Net是最重要的模型之一，因为它再次强调CNN必须够深，视觉数据的层次化表示才有用。深的同时结构简单。

GoogLeNet(2015 年)

理解了我们刚才所说的神经网络架构中的简化的概念了吗?通过推出 Inception 模型，谷歌从某种程度上把这一概念抛了出来。GoogLeNet是一个22层的卷积神经网络，在2014年的ILSVRC2014上凭借6.7%的错误率进入Top 5。据我所知，这是第一个真正不使用通用方法的卷积神经网络架构，传统的卷积神经网络的方法是简单堆叠卷积层，然后把各层以序列结构堆积起来。论文的作者也强调，这种新的模型重点考虑了内存和能量消耗。这一点很重要，我自己也会经常忽略：把所有的层都堆叠、增加大量的滤波器，在计算和内存上消耗很大，过拟合的风险也会增加。



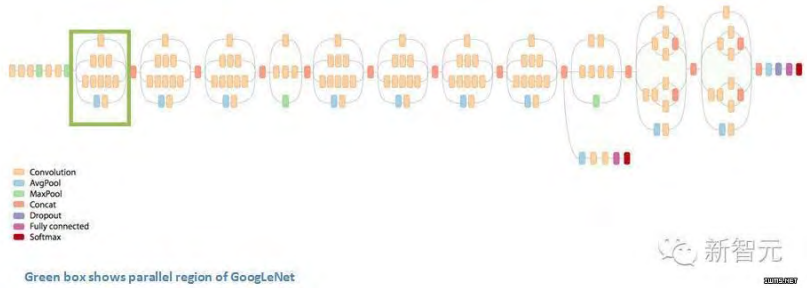
换一种方式看 GoogLeNet:



Another view of GoogLeNet's architecture.

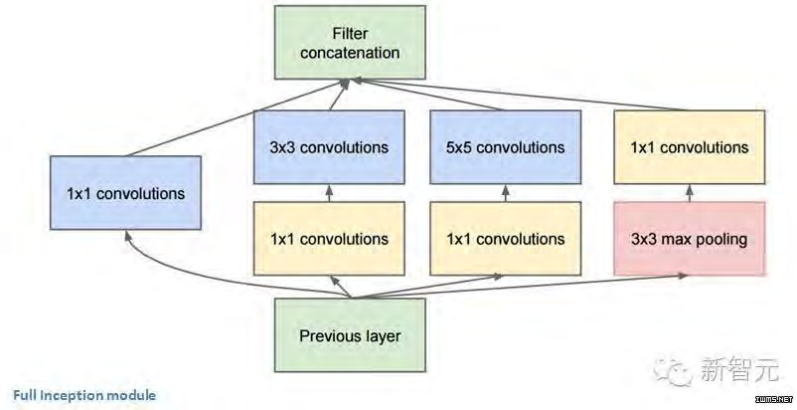
Inception 模型

第一次看到GoogLeNet的构造时，我们立刻注意到，并不是所有的事情都是按照顺序进行的，这与此前看到的架构不一样。我们有一些网络，能同时并行发生反应。



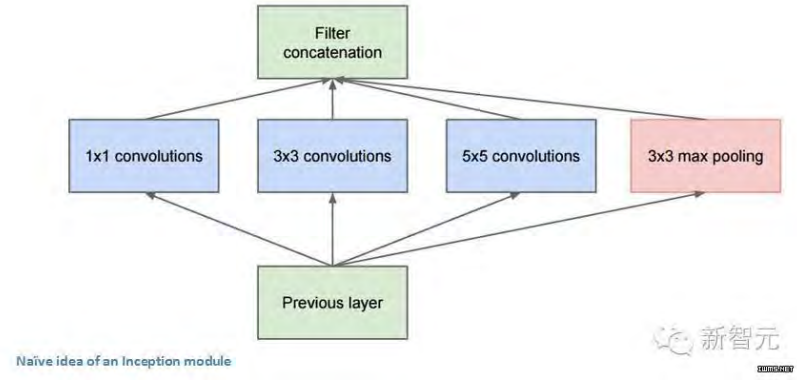
Green box shows parallel region of GoogLeNet

这个盒子被称为 Inception 模型。可以近距离地看看它的构成。



Full Inception module

底部的绿色盒子是我们的输入层，顶部的是输出层(把这张图片向右旋转90度，你会看到跟展示了整个网络的那张图片相对应的模型)。基本上，在一个传统的卷积网络中的每一层中，你必须选择操作池还是卷积操作(还要选择滤波器的大小)。Inception 模型能让你做到的就是并行地执行所有的操作。事实上，这就是作者构想出来的最“初始”的想法。



Naïve idea of an Inception module

现在，来看看它为什么起作用。它会导向许多不同的结果，我们会最后会在输出层体积上获得极端大的深度通道。作者处理这个问题的方法是，在3X3和5X5层前，各自增加一个1X1的卷积操作。1X1的卷积(或者网络层中的网络)，提供了一个减少维度的方法。比如，我们假设你拥有一个输入层，体积是100x100x60(这并不定是图像的三个维度，只是网络中每一层的输入)。增加20个1X1的卷积滤波器，会让你把输入的体积减小到100X100X20。这意味着，3X3层和5X5层不需要处理输入层那么大的体积。这可以被认为是“池特征”(pooling of feature)，因为我们正在减少体积的高度，这和使用常用的最大池化层(maxpooling layers)减少宽度和长度类似。另一个需要注意的是，这些1X1的卷积层后面跟着的是ReLU 单元，这肯定不会有害。

你也许会问，“这个架构有什么用?”这么说吧，这个模型由一个网络层中的网络、一个中等大小的过滤卷积、一个大型的过滤卷积、一个操作池(pooling operation)组成。网络卷积层中的网络能够提取输入体积中的每一个细节中的信息，同时 5x5 的滤波器也能够覆盖大部分接受层的的输入，进而能提起其中的信息。你也可以进行一个池操作，以减少空间大小，降低过度拟合。在这些层之上，你在每一个卷积层后都有一个ReLU，这能改进网络的非线性特征。基本上，网络在执行这些基本的功能时，还能同时考虑计算的能力。这篇论文还提供了更高级别的推理，包括的主题有稀疏和紧密联结(见论文第三和第四节)。

要点

整个架构中使用了9个Inception 模型，总共超过100层。这已经很深了……没有使用完全连接的层。他们使用一个平均池代替，从 7x7x1024 的体积降到了 1x1x1024，这节省了大量的参数。比AlexNet的参数少了12X在测试中，相同图像的多个剪裁建立，然后填到网络中，计算softmax probabilities的均值，然后我们可以获得最后的解决方案。在感知模型中，使用了R-CNN中的概念。Inception有一些升级的版本(版本6和7)，“少数高端的GPU”一周内就能完成训练。

为什么重要?

GoogLeNet 是第一个引入了“CNN 各层不需要一直都按顺序堆叠”这一概念的模型。用Inception模型，作者展示了一个具有创造性的层次机构，能带来性能和计算效率的提升。这篇论文确实为接下来几年可能会见到的令人惊叹的架构打下了基础。

微软 ResNet(2015年)

想象一个深度CNN架构，再深、再深、再深，估计都还没有 ILSVRC 2015 冠军，微软的152层ResNet架构深。除了在层数上面创纪录，ResNet 的错误率也低得惊人，达到了3.6%，人类都大约在5%~10%的水平。

为什么重要?

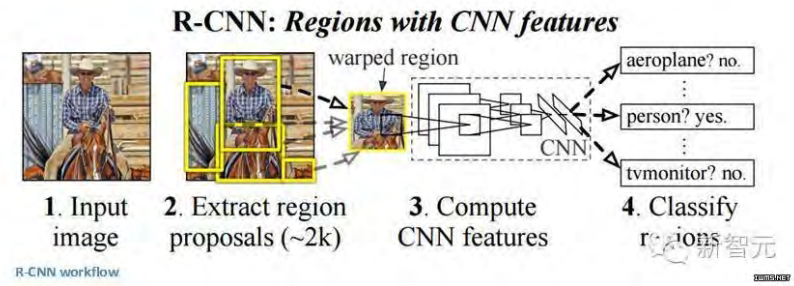
只有3.6%的误差率，这应该足以说服你。ResNet模型是目前最好的CNN架构，而且是残差学习理念的一大创新。从2012年起，错误率逐年下降，我怀疑到ILSVRC2016，是否还会一直下降。我相信，我们现在堆放更多层将不会实现性能的大幅提升。我们必须要创造新的架构。

区域 CNN：R-CNN(2013年)、Fast R-CNN(2015年)、Faster R-CNN(2015年)

一些人可能会认为，R-CNN的出现比此前任何关于新的网络架构的论文都有影响力。第一篇关于R-CNN的论文被引用了超过1600次。Ross Girshick 和他在UC Berkeley 的团队在机器视觉上取得了最有影响力的进步。正如他们的文章所写， Fast R-CNN 和 Faster R-CNN能够让模型变得更快，更好地适应现代的物体识别任务。

R-CNN的目标是解决物体识别的难题。在获得特定的一张图像后， 我们希望能够绘制图像中所有物体的边缘。这一过程可以分为两个组成部分，一个是区域建议，另一个是分类。

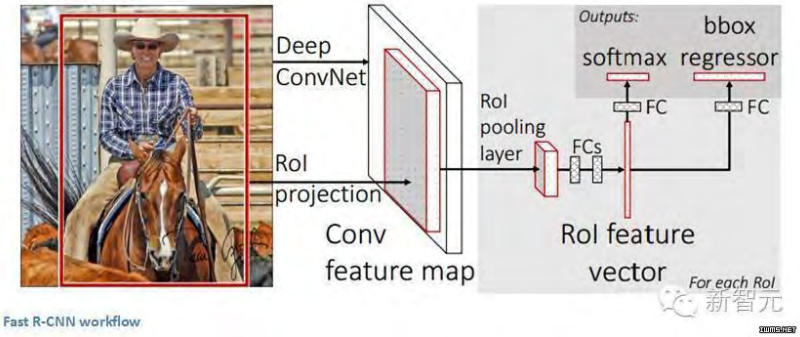
论文的作者强调，任何分类不可知区域的建议方法都应该适用。Selective Search专用于RCNN。Selective Search 的作用是聚合2000个不同的区域，这些区域有最高的可能性会包含一个物体。在我们设计出一系列的区域建议之后，这些建议被汇合到一个图像大小的区域，能被填入到经过训练的CNN(论文中的例子是AlexNet)，能为每一个区域提取出一个对应的特征。这个向量随后被用于作为一个线性SVM的输入，SVM经过了每一种类型和输出分类训练。向量还可以被填入到一个有边界的回归区域，获得最精准的一致性。



非极值压抑后被用于压制边界区域，这些区域相互之间有很大的重复。

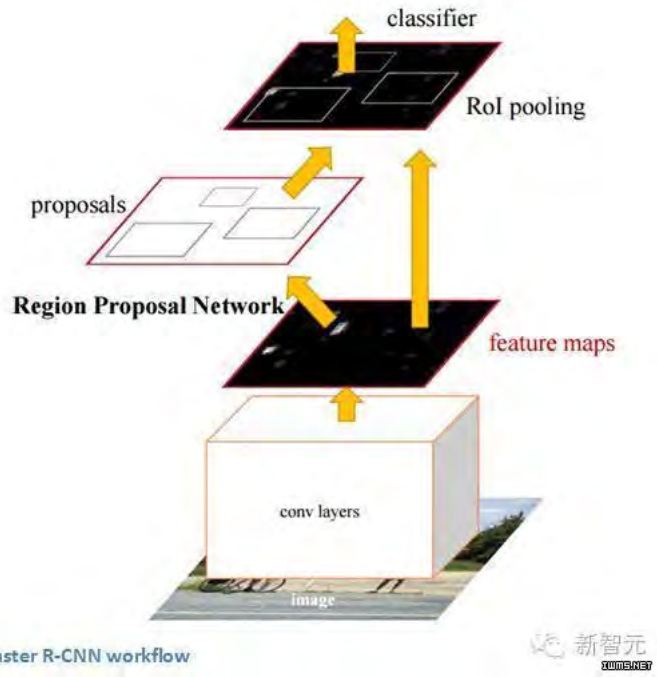
Fast R-CNN

原始模型得到了改进，主要有三个原因：训练需要多个步骤，这在计算上成本过高，而且速度很慢。Fast R-CNN通过从根本上在不同的建议中分析卷积层的计算，同时打乱生成区域建议的顺利以及运行CNN，能够快速地解决问题。



Faster R-CNN

Faster R-CNN的工作是克服R-CNN和 Fast R-CNN所展示出来的，在训练管道上的复杂性。作者 在最后一个卷积层上引入了一个区域建议网络(RPN)。这一网络能够只看最后一层的特征就产出区域建议。从这一层面上来说，相同的R-CNN管道可用。

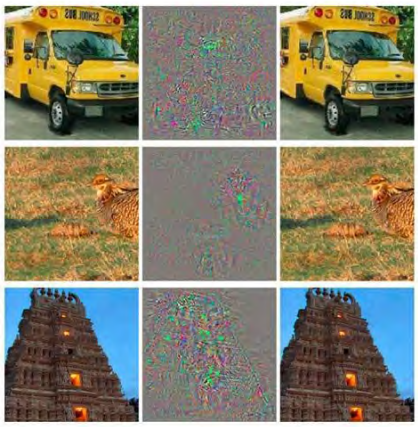


为什么重要?

能够识别出一张图像中的某一个物体是一方面，但是，能够识别物体的精确位置对于计算机知识来说是一个巨大的飞跃。更快的R-CNN已经成为今天标准的物体识别程序。

生成对抗网络(2015年)

按照Yann LeCun的说法，生成对抗网络可能就是深度学习下一个大突破。假设有两个模型，一个生成模型，一个判别模型。判别模型的任务是决定某幅图像是真实的(来自数据库)，还是机器生成的，而生成模型的任务则是生成能够骗过判别模型的图像。这两个模型彼此就形成了“对抗”，发展下去最终会达到一个平衡，生成器生成的图像与真实的图像没有区别，判别器无法区分两者。



The images in the left most column are correctly classified examples. The middle column represents the distortion between the left and right images. The images in the right most column are predicted to be of the class ostrich! Even though the difference between the images on the left and right is imperceptible to humans, the ConvNet makes drastic errors in classification.

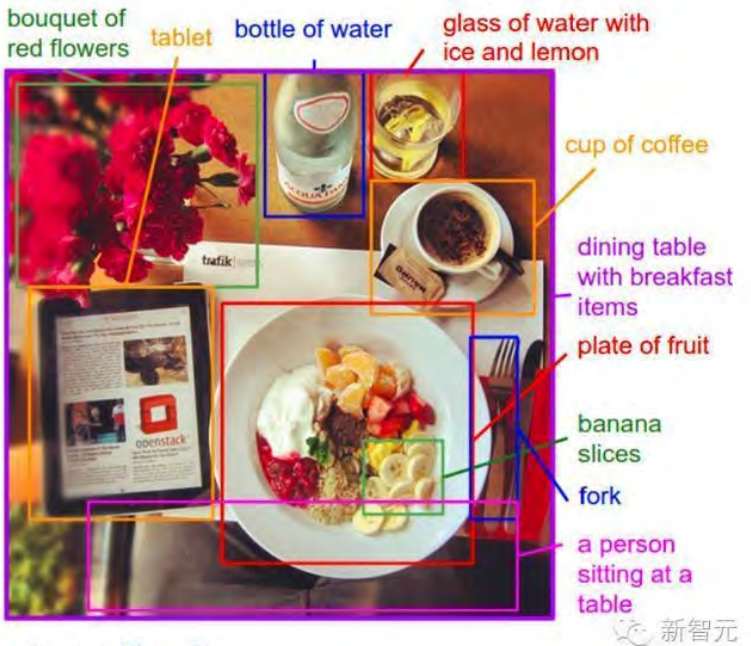
左边一栏是数据库里的图像，也即真实的图像，右边一栏是机器生成的图像，虽然肉眼看上去基本一样，但在CNN看起来却十分不同。

为什么重要?

听上去很简单，然而这是只有在理解了“数据内在表征”之后才能建立的模型，你能够训练网络理解真实图像和机器生成的图像之间的区别。因此，这个模型也可以被用于CNN中做特征提取。此外，你还能用生成对抗模型制作以假乱真的图片。

生成图像描述(2014年)

把CNN和RNN结合在一起会发生什么?Andrej Karpathy 和李飞飞写的这篇论文探讨了结合CNN和双向RNN生成不同图像区域的自然语言描述问题。简单说，这个模型能够接收一张图片，然后输出



Example output of the model

很神奇吧。传统CNN，训练数据中每幅图像都有单一的一个标记。这篇论文描述的模型则是每幅图像都带有一句话(或图说)。这种标记被称为弱标记，使用这种训练数据，一个深度神经网络“推断句子中的部分与其描述的区域之间的潜在对齐(latent alignment)”，另一个神经网络将图像作为输入，生成文本的描述。

为什么重要?

使用看似不相关的RNN和CNN模型创造了一个十分有用的应用，将计算机视觉和自然语言处理结合在一起。这篇论文为如何建模处理跨领域任务提供了全新的思路。

空间转换器网络(2015年)

最后，让我们来看该领域最近的一篇论文。本文是谷歌DeepMind的一个团队在一年前写的。这篇论文的主要贡献是介绍了空间变换器(Spatial Transformer)模块。基本思路是，这个模块会转变输入图像，使随后的层可以更轻松地进行分类。作者试图在图像到达特定层前改变图像，而不是更改主CNN架构本身。该模块希望纠正两件事：姿

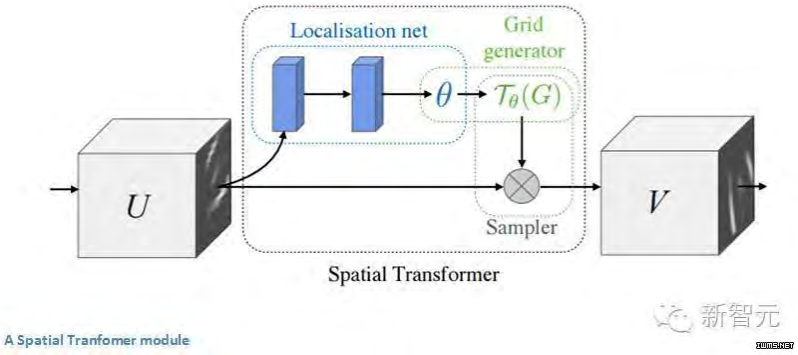
势标准化(场景中物体倾斜或缩放)和空间注意力(在密集的图像中将注意力集中到正确的物体)。对于传统的CNN，如果你想使你的模型对于不同规格和旋转的图像都保持不变，那你需要大量的训练样本来使模型学习。让我们来看看这个模块是如何帮助解决这一问题。

传统CNN模型中，处理空间不变性的是maxpooling层。其原因是，一旦我们知道某个特定特性还是起始输入量(有高激活值)，它的确切位置就没有它对其他特性的相对位置重要，其他功能一样重要。这个新的空间变换器是动态的，它会对每个输入图像产生不同的行为(不同的扭曲/变形)。这不仅仅是像传统 maxpool 那样简单和预定义。让我们来看看这个模块是如何工作的。该模块包括：

一个本地化网络，会吸收输入量，并输出应施加的空间变换的参数。参数可以是6维仿射变换。

采样网格，这是由卷曲规则网格和定位网络中创建的仿射变换(theta)共同产生的。

一个采样器，其目的是执行输入功能图的翘曲。



A Spatial Tranfomer module

该模块可以放入CNN的任何地方中，可以帮助网络学习如何以在训练过程中最大限度地减少成本函数的方式来变换特征图。

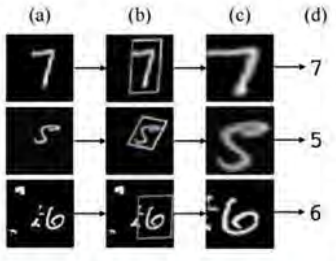


Figure 1: The result of using a spatial transformer as the first layer of a fully-connected network trained for distorted MNIST digit classification. (a) The input to the spatial transformer network is an image of an MNIST digit that is distorted with random translation, scale, rotation, and clutter. (b) The localisation network of the spatial transformer predicts a transformation to apply to the input image. (c) The output of the spatial transformer, after applying the transformation. (d) The classification prediction produced by the subsequent fully-connected network on the output of the spatial transformer. The spatial transformer network (a CNN including a spatial transformer module) is trained end-to-end with only class labels – no knowledge of the groundtruth transformations is given to the system.

Great overview of the function of a Spatial Tranfomer module

为什么重要?

CNN的改进不一定要到通过网络架构的大改变来实现。我们不需要创建下一个ResNet或者 Inception 模型。本文实现了对输入图像进行仿射变换的简单的想法，以使模型对平移，缩放和旋转保持不变。更多请查看《CNN十篇经典论文》。

深度学习在视觉领域有何前沿进展？深度学习 DL应用 难
@元峰，本题解析来源：https://zhuanlan.zhihu.com/p/24699780

引言

在今年的神经网络顶级会议NIPS2016上，深度学习三大牛之一的Yann Lecun教授给出了一个关于机器学习中的有监督学习、无监督学习和增强学习的一个有趣的比喻，他说：如果把智能（Intelligence）比作一个蛋糕，那么无监督学习就是蛋糕本体，增强学习是蛋糕上的樱桃，那么监督学习，仅仅能算作蛋糕上的糖霜（图1）。

图1. Yann LeCun 对监督学习，增强学习和无监督学习的价值的形象比喻

1. 深度有监督学习在计算机视觉领域的进展

1.1 图像分类（Image Classification）

自从Alex和他的导师Hinton（深度学习鼻祖）在2012年的ImageNet大规模图像识别竞赛（ILSVRC2012）中以超过第二名10个百分点的成绩(83.6%的Top5精度)碾压第二名（74.2%，使用传统的计算机视觉方法）后，深度学习真正开始火热，卷积神经网络（CNN）开始成为家喻户晓的名字，从12年的AlexNet（83.6%），到2013年ImageNet 大规模图像识别竞赛冠军的88.8%，再到2014年VGG的92.7%和同年的GoogLeNet的93.3%，终于，到了2015年，在1000类的图像识别中，微软提出的残差网（ResNet）以96.43%的Top5正确率，达到了超过人类的水平（人类的正确率也只有94.9%）.

Top5精度是指在给出一张图片，模型给出5个最有可能的标签，只要在预测的5个结果中包含正确标签，即为正确

HashMap与HashTable区别？数据结构 hash表 中
点评：HashMap基于Hashtable实现，不同之处在于HashMap是非同步的，并且允许null，即null value和null key，Hashtable则不允许null，详见：http://oznyang.iteye.com/blog/30690。此外，记住一点：hashmap/hashset等凡是带有hash字眼的均基于hashtable实现，没带hash字眼的如set/map均是基于红黑树实现，前者无序，后者有序，详见此文第一部分：《教你如何迅速秒杀掉：99%的海量数据处理面试题》。
不过，估计还是直接来图更形象点，故直接上图（图片来源：July9月28日在上海交大面试&算法讲座的PPThttp://vdisk.weibo.com/s/zrFL6OXKg_1me）：

在分类问题中，我们经常会遇到正负样本数据量不等的情况，比如正样本为10w条数据，负样本只有1w条数据，以下最合适的处理方法是( )？ 机器学习 ML基础 中
A 将负样本重复10次，生成10w样本量，打乱顺序参与分类
B 直接进行分类，可以最大限度利用数据
C 从10w正样本中随机抽取1w参与分类
D 将负样本每个权重设置为10，正样本权重为1，参与训练过程
@管博士：准确的说，其实选项中的这些方法各有优缺点，需要具体问题具体分析，有篇文章对各种方法的优缺点进行了分析，讲的不错 感兴趣的同学可以参考一下：https://www.analyticsvidhya.com/blog/2017/03/imbalanced-classification-problem/。

以下第69题~第83题来自：http://blog.csdn.net/u011204487

深度学习是当前很热门的机器学习算法，在深度学习中，涉及到大量的矩阵相乘，现在需要计算三个稠密矩阵A,B,C的乘积ABC,假设三个矩阵的尺寸分别为m*n，n*p，p*q，且m<n<p<q，以下计算顺序效率最高的是（）？ 深度学习 DL基础 中

A.(AB)C

B.AC(B)

C.A(BC)

D.所以效率都相同

正确答案：A

@BlackEyes_SGC： m*n*p<m*n*q,m*p*q< n*p*q, 所以 (AB)C 最小

Nave Bayes是一种特殊的Bayes分类器,特征变量是X,类别标签是C,它的一个假定是（）机器学习 ML模型 中

A.各类别的先验概率P(C)是相等的

B.以0为均值，sqr(2)/2为标准差的正态分布

C.特征变量X的各个维度是类别条件独立随机变量

D.P(X|C)是高斯分布

正确答案：C

@BlackEyes_SGC：朴素贝叶斯的基本假设就是每个变量相互独立。

关于支持向量机SVM,下列说法错误的是（） 机器学习 ML模型 中

A.L2正则项，作用是最大化分类间隔，使得分类器拥有更强的泛化能力

B.Hinge 损失函数，作用是最小化经验分类错误

C.分类间隔为1/||w||，||w||代表向量的模

D.当参数C越小时，分类间隔越大，分类错误越多，趋于欠学习

正确答案：C

@BlackEyes_SGC：

A正确。考虑加入正则化项的原因：想象一个完美的数据集，y>1是正类，y<-1是负类，决策面y=0，加入一个y=-30的正类噪声样本，那么决策面将会变“歪”很多，分类间隔变小，泛化能力减小。加入正则项之后，对噪声样本的容错能力增强，前面提到的例子里面，决策面就会没那么“歪”了，使得分类间隔变大，提高了泛化能力。

B正确。

C错误。间隔应该是2/||w||才对，后半句应该没错，向量的模通常指的就是其二范数。

D正确。考虑软间隔的时候，C对优化问题的影响就在于把a的范围从[0，+inf]限制到了[0,C]。C越小，那么a就会越小，目标函数拉格朗日函数导数为0可以求出w=求和ai?yi?xi，a变小使得w变小，因此间隔2/||w||变大

在HMM中,如果已知观察序列和产生观察序列的状态序列,那么可用以下哪种方法直接进行参数估计（）机器学习 ML模型 中

A.EM算法

B.维特比算法

C.前向后向算法

D.极大似然估计

正确答案：D

@BlackEyes_SGC：

EM算法： 只有观测序列，无状态序列时来学习模型参数，即Baum-Welch算法

维特比算法： 用动态规划解决HMM的预测问题，不是参数估计

前向后向算法：用来算概率

极大似然估计：即观测序列和相应的状态序列都存在时的监督学习算法，用来估计参数

注意的是在给定观测序列和对应的状态序列估计模型参数，可以利用极大似然发估计。如果给定观测序列，没有对应的状态序列，才用EM，将状态序列看不不可测的隐数据。

假定某同学使用Naive Bayesian（NB）分类模型时，不小心将训练数据的两个维度搞重复了，那么关于NB的说法中正确的是？机器学习 ML模型 中

A.这个被重复的特征在模型中的决定作用会被加强

B.模型效果相比无重复特征的情况下精确度会降低

C.如果所有特征都被重复一遍，得到的模型预测结果相对于不重复的情况下的模型预测结果一样。

D.当两列特征高度相关时，无法用两列特征相同时所得到的结论来分析问题

E.NB可以用来做最小二乘回归

F.以上说法都不正确

正确答案：BD

@BlackEyes_SGC：NB的核心在于它假设向量的所有分量之间是独立的。在贝叶斯理论系统中，都有一个重要的条件独立性假设：假设所有特征之间相互独立，这样才能将联合概率拆分

以下哪些方法不可以直接来对文本分类？机器学习 ML模型 易

A、Kmeans

B、决策树

C、支持向量机

D、KNN正确答案: A分类不同于聚类。

@BlackEyes_SGC：A：Kmeans是聚类方法，典型的无监督学习方法。分类是监督学习方法，BCD都是常见的分类方法。

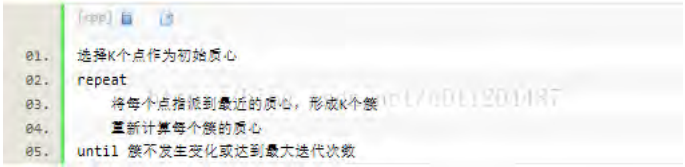
已知一组数据的协方差矩阵P,下面关于主分量说法错误的是（）机器学习 ML基础 易

A、主分量分析的最佳准则是对一组数据进行按一组正交基分解, 在只取相同数量分量的条件下,以均方误差计算截尾误差最小

B、在经主分量分解后,协方差矩阵成为对角矩阵

C、主分量分析就是K-L变换

D、主分量是通过求协方差矩阵的特征值得到
正确答案: C
@BlackEyes_SGC：K-L变换与PCA变换是不同的概念，PCA的变换矩阵是协方差矩阵，K-L变换的变换矩阵可以有很多种（二阶矩阵、协方差矩阵、总类内离散度矩阵等等）。当K-L变换矩阵为协方差矩阵时，等同于PCA。
kmeans的复杂度？机器学习 ML模型 易



时间复杂度：O(tKmn)，其中，t为迭代次数，K为簇的数目，m为记录数，n为维数空间复杂度：O((m+K)n)，其中，K为簇的数目，m为记录数，n为维数
关于logit 回归和SVM 不正确的是（A） 机器学习 ML模型 中
A. Logit回归本质上是一种根据样本对权值进行极大似然估计的方法，而后验概率正比于先验概率和似然函数的乘积。logit仅仅是最大化似然函数，并没有最大化后验概率，更谈不上最小化后验概率。A错误
B. Logit回归的输出就是样本属于正类别的几率，可以计算出概率，正确
C. SVM的目标是找到使得训练数据尽可能分开且分类间隔最大的超平面，应该属于结构风险最小化。
D. SVM可以通过正则化系数控制模型的复杂度，避免过拟合。
@BlackEyes_SGC：Logit回归目标函数是最小化后验概率，Logit回归可以用于预测事件发生概率的大小，SVM目标是结构风险最小化，SVM可以有效避免模型过拟合。
输入图片大小为200×200，依次经过一层卷积（kernel size 5×5，padding 1，stride 2），pooling（kernel size 3×3，padding 0，stride 1），又一层卷积（kernel size 3×3，padding 1，stride 1）之后，输出特征图大小为（） 深度学习 DL基础 中
A 95
B 96
C 97
D 98
E 99
F 100
正确答案：C
@BlackEyes_SGC：计算尺寸不被整除只在GoogLeNet中遇到过。卷积向下取整，池化向上取整。
本题 （200-5+2*1）/2+1 为99.5，取99
（99-3）/1+1 为97
（97-3+2*1）/1+1 为97
研究过网络的话看到stride为1的时候，当kernel为 3 padding为1或者kernel为5 padding为2 一看就是卷积前后尺寸不变。
计算GoogLeNet全过程的尺寸也一样。
影响聚类算法结果的主要因素有（B、C、D ） 机器学习 ML模型 易
A.已知类别的样本质量；
B.分类准则；
C.特征选取；
D.模式相似性测度 模式识别中，马式距离较之于欧式距离的优点是（C、D） 机器学习 ML模型 易
A.平移不变性；
B.旋转不变性；
C尺度不变性；
D.考虑了模式的分布影响基本K-均值算法的主要因素有(BD） 机器学习 ML模型 易
A.样本输入顺序；
B.模式相似性测度；
C.聚类准则；
D.初始类中心的选取在统计模式分类问题中，当先验概率未知时，可以使用（BD） 机器学习 ML模型 易
A. 最小损失准则；
B. 最小最大损失准则；
C. 最小误判概率准则；
D. N-P判决如果以特征向量的相关系数作为模式相似性测度，则影响聚类算法结果的主要因素有（BC） 机器学习 ML模型 易
A. 已知类别样本质量；
B. 分类准则；
C. 特征选取；
D. 量纲欧式距离具有（A B ）；马式距离具有（A B C D ） 机器学习 ML基础 易
A. 平移不变性；
B. 旋转不变性；
C. 尺度缩放不变性；
D. 不受量纲影响的特性
你有哪些deep learning（rnn、cnn）调参的经验？ 深度学习 DL基础 中
@萧瑟，来源：https://www.zhihu.com/question/41631631/answer/94816420

参数初始化

下面几种方式,随便选一个,结果基本都差不多。但是一定要做。否则可能会减慢收敛速度，影响收敛结果，甚至造成Nan等一系列问题。

下面的n_in为网络的输入大小，n_out为网络的输出大小，n为n_in或(n_in+n_out)*0.5

Xavier初始法论文：http://jmlr.org/proceedings/papers/v9/glorot10a/glorot10a.pdf

He初始化论文：https://arxiv.org/abs/1502.01852

- uniform均匀分布初始化：w = np.random.uniform(low=-scale, high=scale, size=[n_in,n_out])
  - Xavier初始法，适用于普通激活函数(tanh,sigmoid)：scale = np.sqrt(3/n)

    - He初始化，适用于ReLU：scale = np.sqrt(6/n)
- normal高斯分布初始化：w = np.random.randn(n_in,n_out) * stdev # stdev为高斯分布的标准差，均值设为0
    - Xavier初始法，适用于普通激活函数 (tanh,sigmoid)：stdev = np.sqrt(n)
    - He初始化，适用于ReLU：stdev = np.sqrt(2/n)
- svd初始化：对RNN有比较好的效果。参考论文：https://arxiv.org/abs/1312.6120

数据预处理方式

- zero-center ,这个挺常用的.X -= np.mean(X, axis = 0) # zero-centerX /= np.std(X, axis = 0) # normalize
- PCA whitening,这个用的比较少.

训练技巧

- 要做梯度归一化,即算出来的梯度除以minibatch size
- clip c(梯度裁剪): 限制最大梯度,其实是value = sqrt(w1^2+w2^2....),如果value超过了阈值,就算一个衰减系系数,让value的值等于阈值: 5,10,15
- dropout对小数据防止过拟合有很好的效果,值一般设为0.5,小数据上dropout+sgd在我的大部分实验中，效果提升都非常明显.因此可能的话，建议一定要尝试一下。 dropout的位置比较有讲究, 对于RNN,建议放到输入->RNN与RNN->输出的位置.关于RNN如何用dropout,可以参考这篇论文:http://arxiv.org/abs/1409.2329
- adam,adadelta等,在小数据上,我这里实验的效果不如sgd, sgd收敛速度会慢一些，但是最终收敛后的结果，一般都比较好。如果使用sgd的话,可以选择从1.0或者0.1的学习率开始,隔一段时间,在验证集上检查一下,如果cost没有下降,就对学习率减半. 我看过很多论文都这么搞,我自己实验的结果也很好. 当然,也可以先用ada系列先跑,最后快收敛的时候,更换成sgd继续训练.同样也会有提升.据说adadelta一般在分类问题上效果比较好，adam在生成问题上效果比较好。
- 除了gate之类的地方,需要把输出限制成0-1之外,尽量不要用sigmoid,可以用tanh或者relu之类的激活函数.1. sigmoid函数在-4到4的区间里，才有较大的梯度。之外的区间，梯度接近0，很容易造成梯度消失问题。2. 输入0均值，sigmoid函数的输出不是0均值的。
- rnn的dim和embdding size,一般从128上下开始调整. batch size,一般从128左右开始调整.batch size合适最重要,并不是越大越好.
- word2vec初始化,在小数据上,不仅可以有效提高收敛速度,也可以可以提高结果.
- 尽量对数据做shuffle
- LSTM 的forget gate的bias,用1.0或者更大的值做初始化,可以取得更好的结果,来自这篇论文:http://jmlr.org/proceedings/papers/v37/jozefowicz15.pdf, 我这里实验设成1.0,可以提高收敛速度.实际使用中,不同的任务,可能需要尝试不同的值.
- Batch Normalization据说可以提升效果，不过我没有尝试过，建议作为最后提升模型的手段，参考论文：Accelerating Deep Network Training by Reducing Internal Covariate Shift
- 如果你的模型包含全连接层（MLP），并且输入和输出大小一样，可以考虑将MLP替换成Highway Network,我尝试对结果有一点提升，建议作为最后提升模型的手段，原理很简单，就是给输出加了一个gate来控制信息的流动，详细介绍请参考论文: http://arxiv.org/abs/1505.00387
- 来自@张馨宇的技巧：一轮加正则，一轮不加正则，反复进行。

Ensemble

Ensemble是论文刷结果的终极核武器,深度学习中一般有以下几种方式

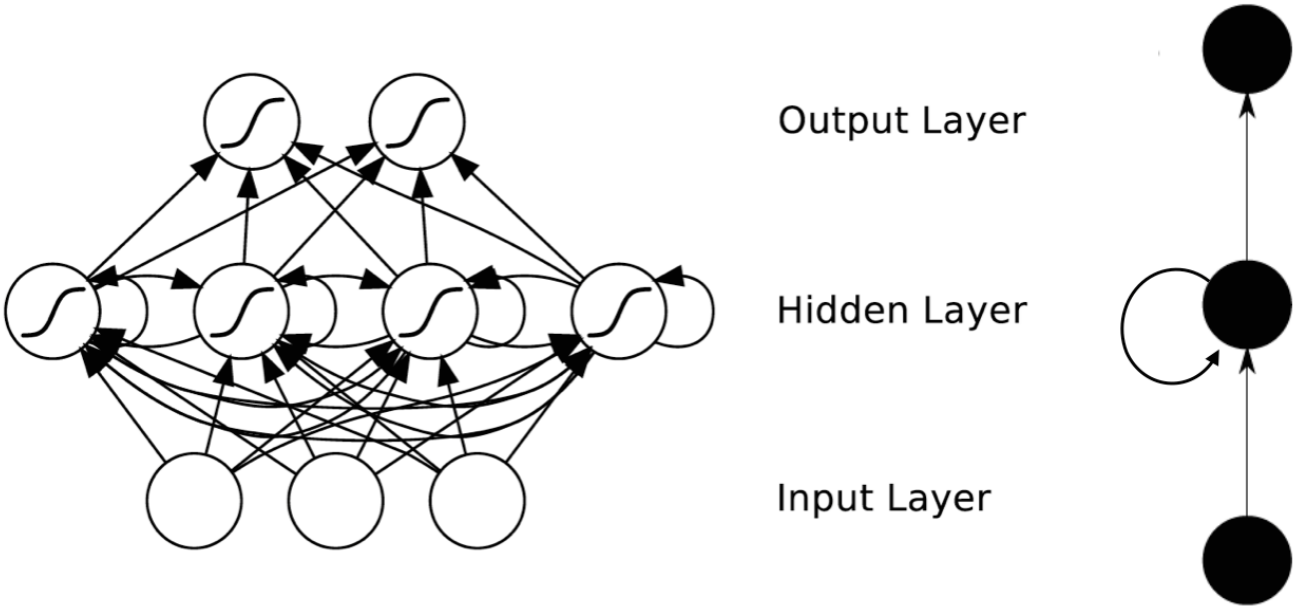
- 同样的参数,不同的初始化方式
- 不同的参数,通过cross-validation,选取最好的几组
- 同样的参数,模型训练的不同阶段，即不同迭代次数的模型。
- 不同的模型,进行线性融合. 例如RNN和传统模型.

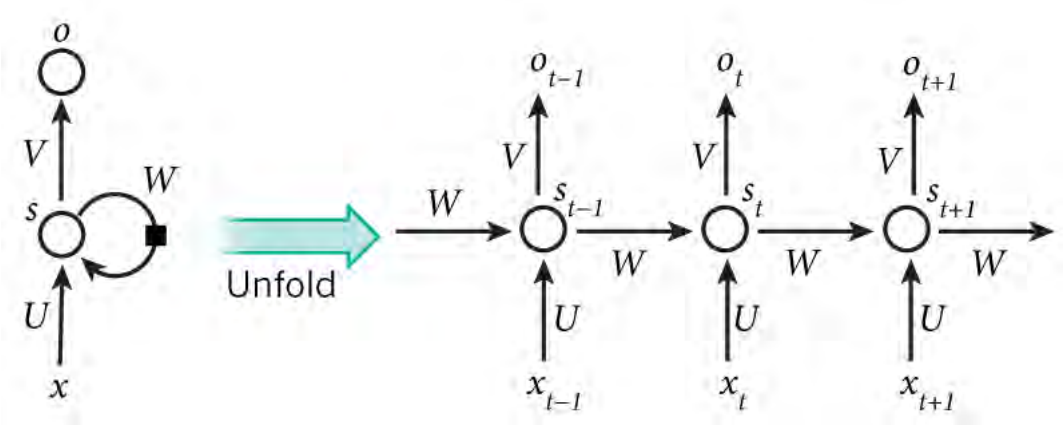
更多深度学习技巧，请参见专栏：炼丹实验室 - 知乎专栏

简单说说RNN的原理？深度学习 DL模型 中
我们升学到高三准备高考时，此时的知识是由高二及高二之前所学的知识加上高三所学的知识合成得来，即我们的知识是由前序铺垫，是有记忆的，好比当电影字幕上出现：“我是”时，你会很自然的联想到：“我是中国人”。

关于RNN，这里有课程详细讲RNN，包括RNN条件生成、attention，以及LSTM等等均有细致讲解：深度学习 [同品类最牛，培养DL工程师]。
什么是RNN？深度学习 DL模型 中
@一只鸟的天空，本题解析来源：http://blog.csdn.net/heyongluoyao8/article/details/48636251
RNNs的目的使用来处理序列数据。在传统的神经网络模型中，是从输入层到隐含层再到输出层，层与层之间是全连接的，每层之间的节点是无连接的。但是这种普通的神经网络对于很多问题却无能无力。例如，你要预测句子的下一个单词是什么，一般需要用到前面的单词，因为一个句子中前后单词并不是独立的。

RNNs之所以称为循环神经网路，即一个序列当前的输出与前面的输出也有关。具体的表现形式为网络会对前面的信息进行记忆并应用于当前输出的计算中，即隐藏层之间的节点不再无连接而是有连接的，并且隐藏层的输入不仅包括输入层的输出还包括上一时刻隐藏层的输出。理论上，RNNs能够对任何长度的序列数据进行处理。但是在实践中，为了降低复杂性往往假设当前的状态只与前面的几个状态相关，下图便是一个典型的RNNs：





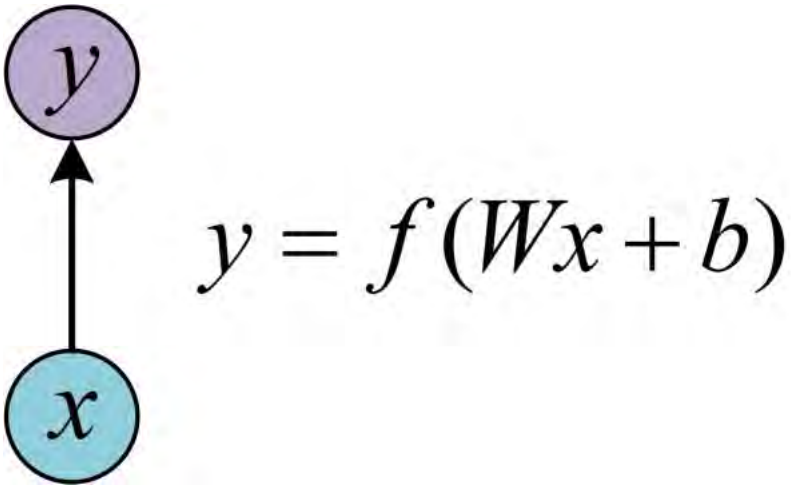
From Nature
??RNNs包含输入单元(Input units)，输入集标记为，而输出单元(Output units)的输出集则被标记为。RNNs还包含隐藏单元(Hidden units)，我们将其输出集标记为，这些隐藏单元完成了最为主要的工作。你会发现，在图中：有一条单向流动的信息流是从输入单元到达隐藏单元的，与此同时另一条单向流动的信息流从隐藏单元到达输出单元。**在某些情况下，RNNs会打破后者的限制，引导信息从输出单元返回隐藏单元，这些被称为“Back Projections”，并且隐藏层的输**入还包括上一隐藏层的状态，即隐藏层内的节点可以自连也可以互连。
??上图将循环神经网络进行展开成一个全神经网络。例如，对一个包含5个单词的语句，那么展开的网络便是一个五层的神经网络，每一层代表一个单词。对于该网络的计算过程如下：

- 表示第步(step)的输入。比如，为第二个词的one-hot向量(根据上图，为第一个词)；
- 为隐藏层的第步的状态，它是网络的记忆单元。 根据当前输入层的输出与上一步隐藏层的状态进行计算。，其中一般是非线性的激活函数，如tanh或ReLU，在计算时，即第一个单词的隐藏层状态，需要用到，但是其并不存在，在实现中一般置为0向量；
- 是第步的输出，如下个单词的向量表示，.
  更多请看此文：循环神经网络(RNN, Recurrent Neural Networks)介绍。

RNN是怎么从单层网络一步一步构造的？深度学习 DL模型 难
@何之源，本题解析来源：https://zhuanlan.zhihu.com/p/28054589

一、从单层网络谈起

在学习RNN之前，首先要了解一下最基本的单层网络，它的结构如图：



输入是x，经过变换Wx+b和激活函数f得到输出y。相信大家对这个已经非常熟悉了。

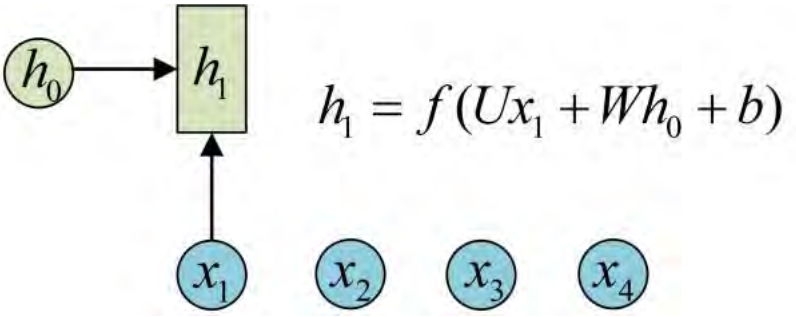
二、经典的RNN结构（N vs N）

在实际应用中，我们还会遇到很多序列形的数据：

如：

- 自然语言处理问题。x1可以看做是第一个单词，x2可以看做是第二个单词，依次类推。
- 语音处理。此时，x1、x2、x3……是每帧的声音信号。
- 时间序列问题。例如每天的股票价格等等。

序列形的数据就不太好用原始的神经网络处理了。为了建模序列问题，RNN引入了隐状态h（hidden state）的概念，h可以对序列形的数据提取特征，接着再转换为输出。先从h1的计算开始看：

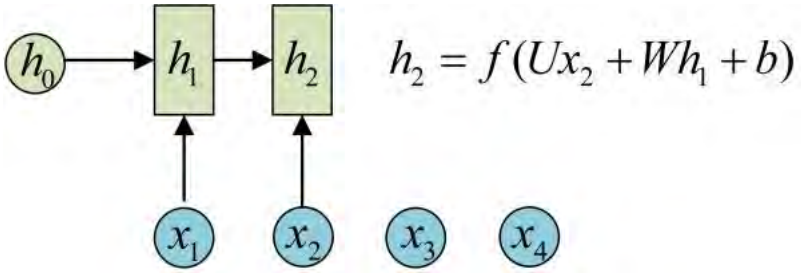


图示中记号的含义是：

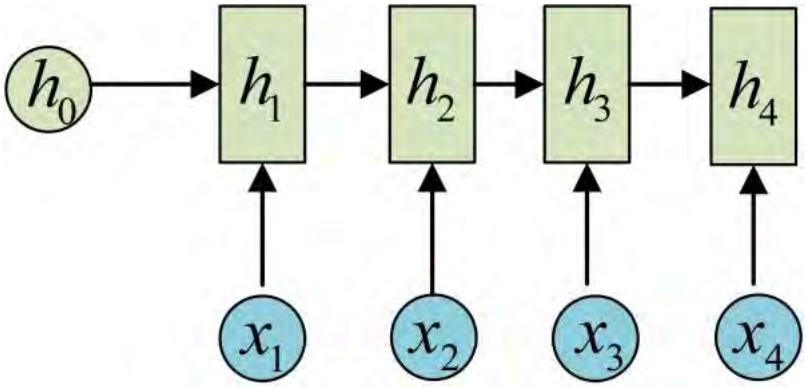
- 圆圈或方块表示的是向量。
- 一个箭头就表示对该向量做一次变换。如上图中h0和x1分别有一个箭头连接，就表示对h0和x1各做了一次变换。

在很多论文中也会出现类似的记号，初学的时候很容易搞乱，但只要把握住以上两点，就可以比较轻松地理解图示背后的含义。

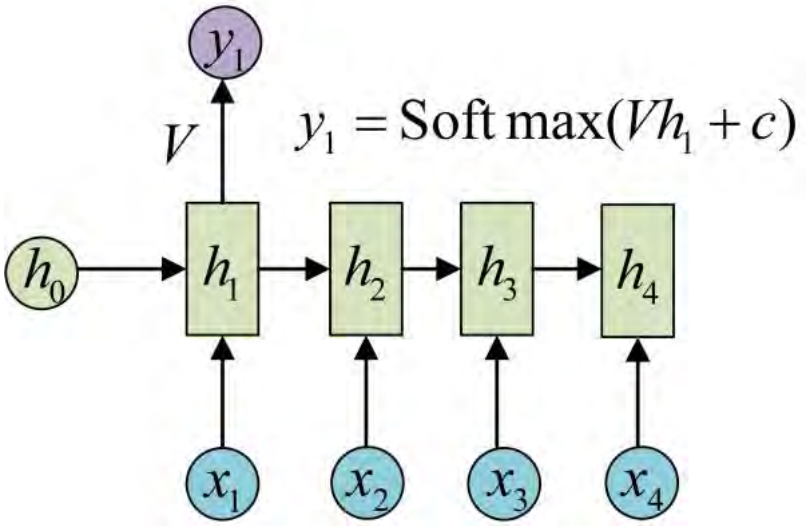
h2的计算和h1类似。要注意的是，在计算时，每一步使用的参数U、W、b都是一样的，也就是说每个步骤的参数都是共享的，这是RNN的重要特点，一定要牢记。



依次计算剩下来的（使用相同的参数U、W、b）：

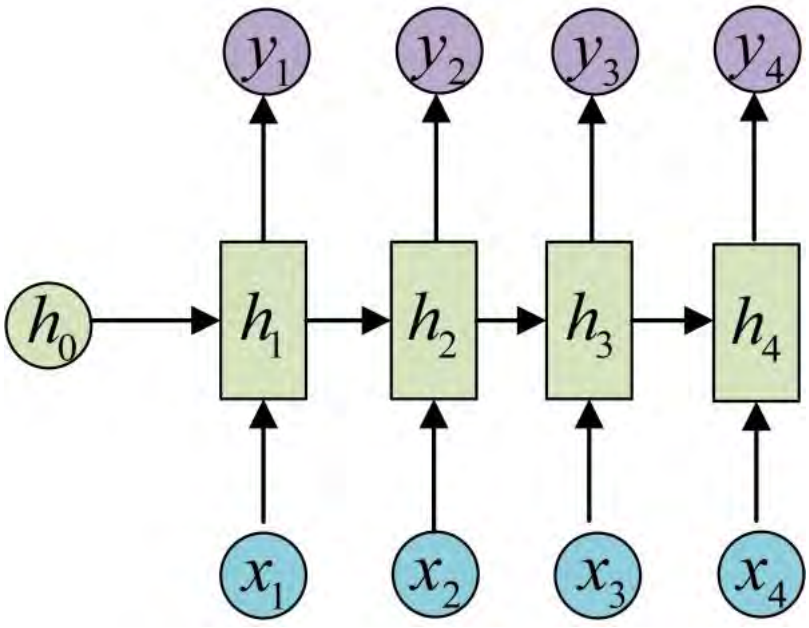


我们这里为了方便起见，只画出序列长度为4的情况，实际上，这个计算过程可以无限地持续下去。



我们目前的RNN还没有输出，得到输出值的方法就是直接通过h进行计算：

正如之前所说，一个箭头就表示对对应的向量做一次类似于f(Wx+b)的变换，这里的这个箭头就表示对h1进行一次变换，得到输出y1。



剩下的输出类似进行（使用和y1同样的参数V和c）：

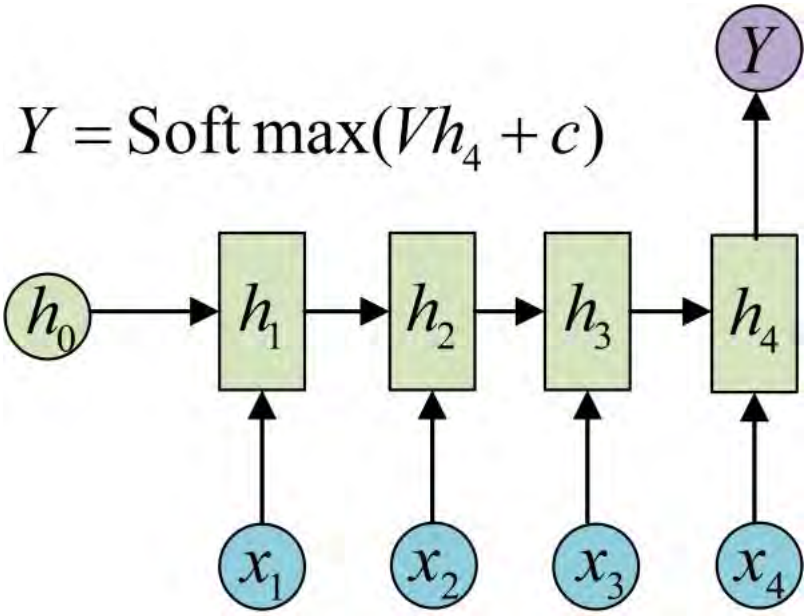
OK！大功告成！这就是最经典的RNN结构，我们像搭积木一样把它搭好了。它的输入是x1, x2, ...xn，输出为y1, y2, ...yn，也就是说，输入和输出序列必须要是等长的。

由于这个限制的存在，经典RNN的适用范围比较小，但也有一些问题适合用经典的RNN结构建模，如：

- 计算视频中每一帧的分类标签。因为要对每一帧进行计算，因此输入和输出序列等长。
- 输入为字符，输出为下一个字符的概率。这就是著名的Char RNN（详细介绍请参考：The Unreasonable Effectiveness of Recurrent Neural Networks，Char RNN可以用来生成文章、诗歌，甚至是代码。此篇博客里有自动生成歌词的实验教程《基于torch学汪峰写歌词、聊天机器人、图像着色/生成、看图说话、字幕生成》）。

三、N VS 1

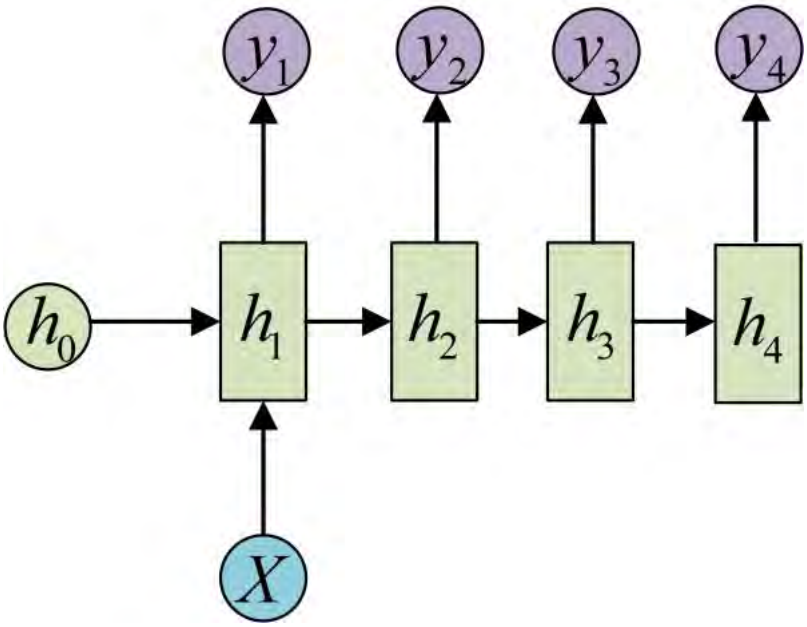
有的时候，我们要处理的问题输入是一个序列，输出是一个单独的值而不是序列，应该怎样建模呢？实际上，我们只在最后一个h上进行输出变换就可以了：



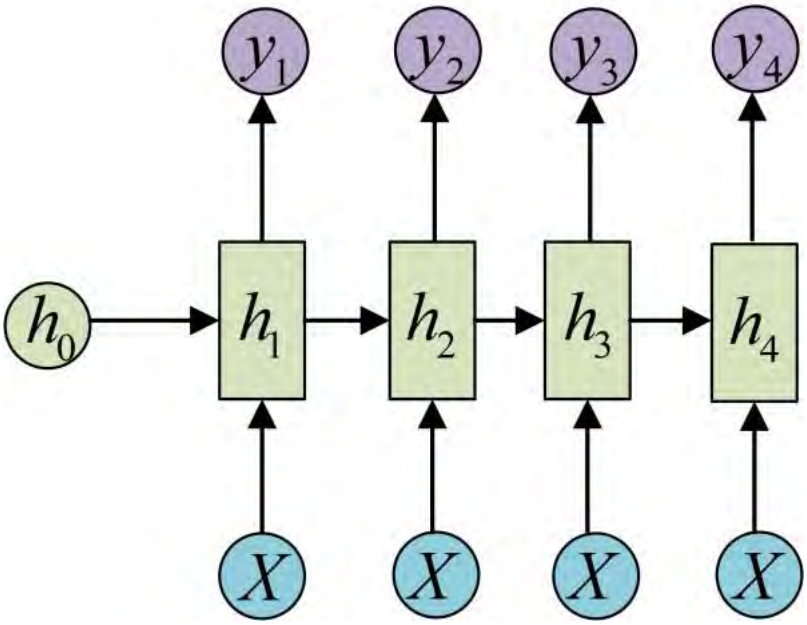
这种结构通常用来处理序列分类问题。如输入一段文字判别它所属的类别，输入一个句子判断其情感倾向，输入一段视频并判断它的类别等等。

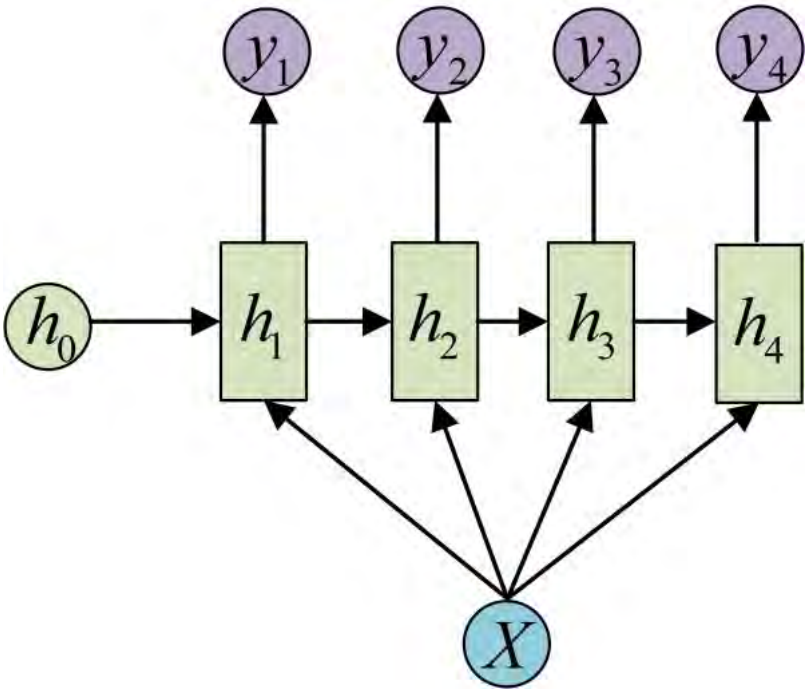
四、1 VS N

输入不是序列而输出为序列的情况怎么处理？我们可以只在序列开始进行输入计算：



还有一种结构是把输入信息X作为每个阶段的输入：





下图省略了一些X的圆圈，是一个等价表示：

这种1 VS N的结构可以处理的问题有：

- 从图像生成文字（image caption），此时输入的X就是图像的特征，而输出的y序列就是一段句子
- 从类别生成语音或音乐等

五、N vs M

下面我们来介绍RNN最重要的一个变种：N vs M。这种结构又叫Encoder-Decoder模型，也可以称之为Seq2Seq模型。

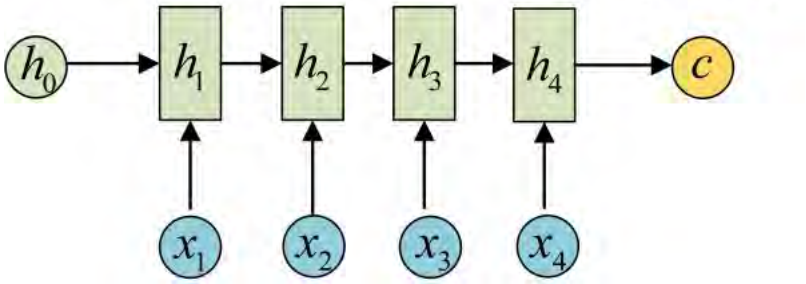
原始的N vs N RNN要求序列等长，然而我们遇到的大部分问题序列都是不等长的，如机器翻译中，源语言和目标语言的句子往往并没有相同的长度。

为此，Encoder-Decoder 结构先将输入数据编码成一个上下文向量c：

（1）$c = h_4$

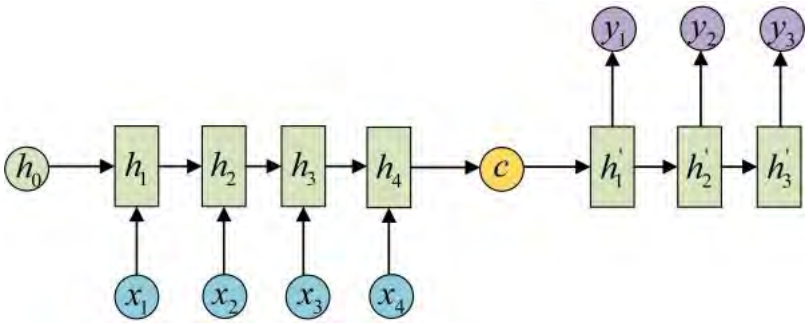
（2）$c = q(h_4)$

（3）$c = q(h_1, h_2, h_3, h_4)$

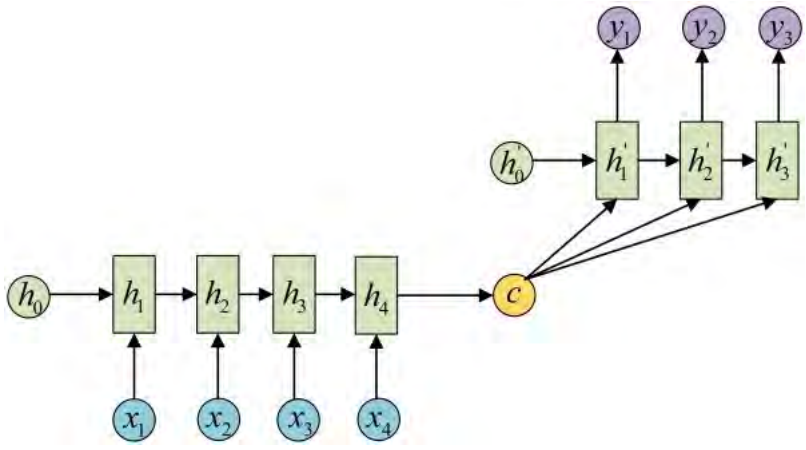


得到c有多种方式，最简单的方法就是把Encoder的最后一个隐状态赋值给c，还可以对最后的隐状态做一个变换得到c，也可以对所有的隐状态做变换。

拿到c之后，就用另一个RNN网络对其进行解码，这部分RNN网络被称为Decoder。具体做法就是将c当做之前的初始状态h0输入到Decoder中：



还有一种做法是将c当做每一步的输入：



由于这种Encoder-Decoder结构不限制输入和输出的序列长度，因此应用的范围非常广泛，比如：

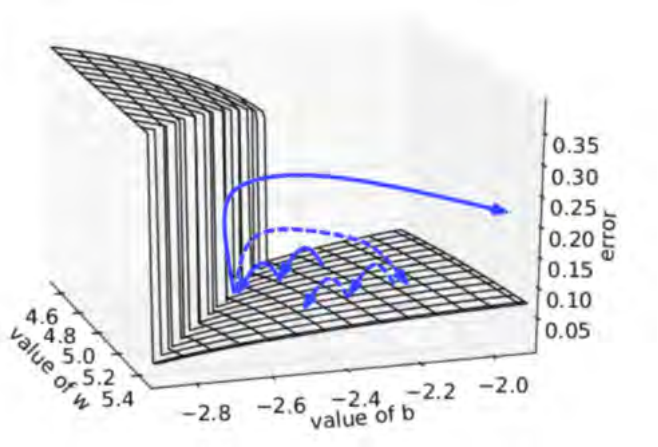
- 机器翻译。Encoder-Decoder的最经典应用，事实上这一结构就是在机器翻译领域最先提出的
- 文本摘要。输入是一段文本序列，输出是这段文本序列的摘要序列。
- 阅读理解。将输入的文章和问题分别编码，再对其进行解码得到问题的答案。
- 语音识别。输入是语音信号序列，输出是文字序列。

RNN中只能采用tanh而不是ReLu作为激活函数么？深度学习 DL模型 中
解析详见：https://www.zhihu.com/question/61265076
深度学习（CNN RNN Attention）解决大规模文本分类问题。深度学习 DL应用 难
https://zhuanlan.zhihu.com/p/25928551
如何解决RNN梯度爆炸和弥散的问题？深度学习 DL模型 难
本题解析来源：深度学习与自然语言处理(7)_斯坦福cs224d 语言模型，RNN，LSTM与GRU
为了解决梯度爆炸问题，Thomas Mikolov首先提出了一个简单的启发性的解决方案，就是当梯度大于一定阈值的的时候，将它截断为一个较小的数。具体如算法1所述：

算法：当梯度爆炸时截断梯度（伪代码）

if then

下图可视化了梯度截断的效果。它展示了一个小的rnn（其中W为权值矩阵，b为bias项）的决策面。这个模型是一个一小段时间的rnn单元组成；实心箭头表明每步梯度下降的训练过程。当梯度下降过程中，模型的目标函数取得了较高的误差时，梯度将被送到远离决策面的位置。截断模型产生了一个虚线，它将误差梯度拉回到离原始梯度接近的位置。



梯度爆炸，梯度截断可视化

为了解决梯度弥散的问题，我们介绍了两种方法。第一种方法是将随机初始化改为一个有关联的矩阵初始化。第二种方法是使用ReLU（Rectified Linear Units）代替sigmoid函数。ReLU的导数不是0就是1.因此，神经元的梯度将始终为1，而不会当梯度传播了一定时间之后变小。

如何理解LSTM网络。深度学习 DL模型 难
@Not_GOD，本题解析来源：http://www.jianshu.com/p/9dc9f41f0b29/

Recurrent Neural Networks

人类并不是每时每刻都从一片空白的大脑开始他们的思考。在你阅读这篇文章时候，你都是基于自己已经拥有的对先前所见词的理解来推断当前词的真实含义。我们不会将所有的东西都全部丢弃，然后用空白的大脑进行思考。我们的思想拥有持久性。
传统的神经网络并不能做到这点，看起来也像是一种巨大的弊端。例如，假设你希望对电影中的每个时间点的时间类型进行分类。传统的神经网络应该很难来处理这个问题——使用电影中先前的事件推断后续的事件。
RNN 解决了这个问题。RNN 是包含循环的网络，允许信息的持久化。

RNN 包含循环
RNN 包含循环

在上面的示例图中，神经网络的模块，A，正在读取某个输入 x_i，并输出一个值 h_i。循环可以使得信息可以从当前步传递到下一步。
这些循环使得 RNN 看起来非常神秘。然而，如果你仔细想想，这样也不比一个正常的神经网络难于理解。RNN 可以被看做是同一神经网络的多次复制，每个神经网络模块会把消息传递给下一个。所以，如果我们将这个循环展开：

展开的 RNN
展开的 RNN

链式的特征揭示了 RNN 本质上是与序列和列表相关的。他们是对于这类数据的最自然的神经网络架构。
并且 RNN 也已经被人们应用了！在过去几年中，应用 RNN 在语音识别，语言建模，翻译，图片描述等问题上已经取得一定成功，并且这个列表还在增长。我建议大家参考 Andrej Karpathy 的博客文章——The Unreasonable Effectiveness of Recurrent Neural Networks 来看看更丰富有趣的 RNN 的成功应用。
而这些成功应用的关键之处就是 LSTM 的使用，这是一种特别的 RNN，比标准的 RNN 在很多的任务上都表现得更好。几乎所有的令人振奋的关于 RNN 的结果都是通过 LSTM 达到的。这篇博文也会就 LSTM 进行展开。

长期依赖（Long-Term Dependencies）问题

RNN 的关键点之一就是他们可以用来连接先前的信息到当前的任务上，例如使用过去的视频段来推测对当前段的理解。如果 RNN 可以做到这个，他们就变得非常有用。但是真的可以么？答案是，还有很多依赖因素。
有时候，我们仅仅需要知道先前的信息来执行当前的任务。例如，我们有一个语言模型用来基于先前的词来预测下一个词。如果我们试着预测 “the clouds are in the sky” 最后的词，我们并不需要任何其他的上下文 —— 因此下一个词很显然就应该是 sky。在这样的场景中，相关的信息和预测的词位置之间的间隔是非常小的，RNN 可以学会使用先前的信息。

不太长的相关信息和位置间隔
不太长的相关信息和位置间隔

但是同样会有一些更加复杂的场景。假设我们试着去预测“I grew up in France... I speak fluent French”最后的词。当前的信息建议下一个词可能是一种语言的名字，但是如果我们需要弄清楚是什么语言，我们是需要先前提到的离当前位置很远的 France 的上下文的。这说明相关信息和当前预测位置之间的间隔就肯定变得相当的大。
不幸的是，**在这个间隔不断增大时，RNN 会丧失学习到连接如此远的信息的能力。**

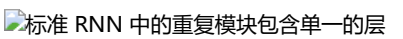
相当长的相关信息和位置间隔

相当长的相关信息和位置间隔

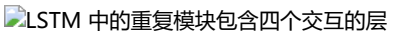
在理论上，RNN 绝对可以处理这样的 长期依赖 问题。人们可以仔细挑选参数来解决这类问题中的最初级形式，但在实践中，RNN 肯定不能够成功学习到这些知识。Bengio, et al. (1994)等人对该问题进行了深入的研究，他们发现一些使训练 RNN 变得非常困难的相当根本的原因。
然而，幸运的是，LSTM 并没有这个问题！

LSTM 网络

Long Short Term 网络—— 一般就叫做 **LSTM ——是一种 RNN 特殊的类型，可以学习长期依赖信息。**如@寒小阳所说：LSTM和基线RNN并没有特别大的结构不同，但是它们用了不同的函数来计算隐状态。LSTM的“记忆”我们叫做细胞/cells，你可以直接把它们想做黑盒，这个黑盒的输入为前状态和当前输入。这些“细胞”会决定哪些之前的信息和状态需要保留/记住，而哪些要被抹去。实际的应用中发现，这种方式可以有效地保存很长时间之前的关联信息。
LSTM 由Hochreiter & Schmidhuber (1997)提出，并在近期被Alex Graves进行了改良和推广。在很多问题，LSTM 都取得相当巨大的成功，并得到了广泛的使用。
LSTM 通过刻意的设计来避免长期依赖问题。记住长期的信息在实践中是 LSTM 的默认行为，而非需要付出很大代价才能获得的能力！
所有 RNN 都具有一种重复神经网络模块的链式的形式。在标准的 RNN 中，这个重复的模块只有一个非常简单的结构，例如一个 tanh 层。


标准 RNN 中的重复模块包含单一的层
标准 RNN 中的重复模块包含单一的层

LSTM 同样是这样的结构，但是重复的模块拥有一个不同的结构。不同于 单一神经网络层，这里是有四个，以一种非常特殊的方式进行交互。


LSTM 中的重复模块包含四个交互的层
LSTM 中的重复模块包含四个交互的层

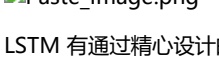
不必担心这里的细节。我们会一步一步地剖析 LSTM 解析图。现在，我们先来熟悉一下图中使用的各种元素的图标。


LSTM 中的图标
LSTM 中的图标

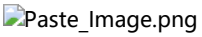
在上面的图例中，每一条黑线传输着一整个向量，从一个节点的输出到其他节点的输入。粉色的圈代表 pointwise 的操作，诸如向量的和，而黄色的矩阵就是学习到的神经网络层。合在一起的线表示向量的连接，分开的线表示内容被复制，然后分发到不同的位置。

LSTM 的核心思想

LSTM 的关键就是细胞状态，水平线在图上方贯穿运行。
细胞状态类似于传送带。直接在整个链上运行，只有一些少量的线性交互。信息在上面流传保持不变会很容易。


Paste_Image.png

LSTM 有通过精心设计的称作为“门”的结构来去除或者增加信息到细胞状态的能力。门是一种让信息选择式通过的方法。他们包含一个 sigmoid 神经网络层和一个 pointwise 乘法操作。


Paste_Image.png

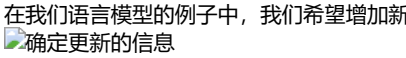
Sigmoid 层输出 0 到 1 之间的数值，描述每个部分有多少量可以通过。0 代表“不许任何量通过”，1 就指“允许任意量通过”！

LSTM 拥有三个门，来保护和控制细胞状态。

逐步理解 LSTM

在我们 LSTM 中的第一步是决定我们会从细胞状态中丢弃什么信息。这个决定通过一个称为忘记门层完成。该门会读取 h_{t-1} 和 x_t，输出一个在 0 到 1 之间的数值给每个在细胞状态 C_{t-1} 中的数字。1 表示“完全保留”，0 表示“完全舍弃”。
让我们回到语言模型的例子中来基于已经看到的预测下一个词。在这个问题中，细胞状态可能包含当前主语的性别，因此正确的代词可以被选择出来。当我们看到新的主语，我们希望忘记旧的主语。

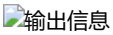

决定丢弃信息
决定丢弃信息

下一步是确定什么样的新信息被存放在细胞状态中。这里包含两个部分。第一，sigmoid 层称 “输入门层” 决定什么值我们将要更新。然后，一个 tanh 层创建一个新的候选值向量，$\tilde{C}_t$，会被加入到状态中。下一步，我们会讲这两个信息来产生对状态的更新。
在我们语言模型的例子中，我们希望增加新的主语的性别到细胞状态中，来替代旧的需要忘记的主语。

确定更新的信息
确定更新的信息

现在是更新旧细胞状态的时间了，C_{t-1} 更新为 C_t。前面的步骤已经决定了将会做什么，我们现在就是实际去完成。
我们把旧状态与 f_t 相乘，丢弃掉我们确定需要丢弃的信息。接着加上 $i_t * \tilde{C}_t$。这就是新的候选值，根据我们决定更新每个状态的程度进行变化。
在语言模型的例子中，这就是我们实际根据前面确定的目标，丢弃旧代词的性别信息并添加新的信息的地方。


更新细胞状态
更新细胞状态

最终，我们需要确定输出什么值。这个输出将会基于我们的细胞状态，但是也是一个过滤后的版本。首先，我们运行一个 sigmoid 层来确定细胞状态的哪个部分将输出出去。接着，我们把细胞状态通过 tanh 进行处理（得到一个在 -1 到 1 之间的值）并将它和 sigmoid 门的输出相乘，最终我们仅仅会输出我们确定输出的那部分。
在语言模型的例子中，因为他就看到了一个 代词，可能需要输出与一个 动词 相关的信息。例如，可能输出是否代词是单数还是负数，这样如果是动词的话，我们也知道动词需要进行的词形变化。


输出信息

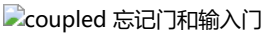
LSTM 的变体

我们到目前为止都还在介绍正常的 LSTM。但是不是所有的 LSTM 都长成一个样子的。实际上，几乎所有包含 LSTM 的论文都采用了微小的变体。差异非常小，但是也值得拿出来讲一下。
其中一个流形的 LSTM 变体，就是由 Gers & Schmidhuber (2000) 提出的，增加了 “peephole connection”。是说，我们让 门层 也会接受细胞状态的输入。


peephole 连接

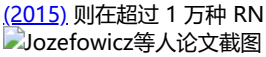
上面的图例中，我们增加了 peephole 到每个门上，但是许多论文会加入部分的 peephole 而非所有都加。

另一个变体是通过使用 coupled 忘记和输入门。不同于之前是分开确定什么忘记和需要添加什么新的信息，这里是一同做出决定。我们仅仅会当我们将要输入在当前位置时忘记。我们仅仅输入新的值到那些我们已经忘记旧的信息的那些状态 。


coupled 忘记门和输入门

另一个改动较大的变体是 Gated Recurrent Unit (GRU)，这是由 Cho, et al. (2014) 提出。它将忘记门和输入门合成了一个单一的 更新门。同样还混合了细胞状态和隐藏状态，和其他一些改动。最终的模型比标准的 LSTM 模型要简单，也是非常流行的变体。


GRU

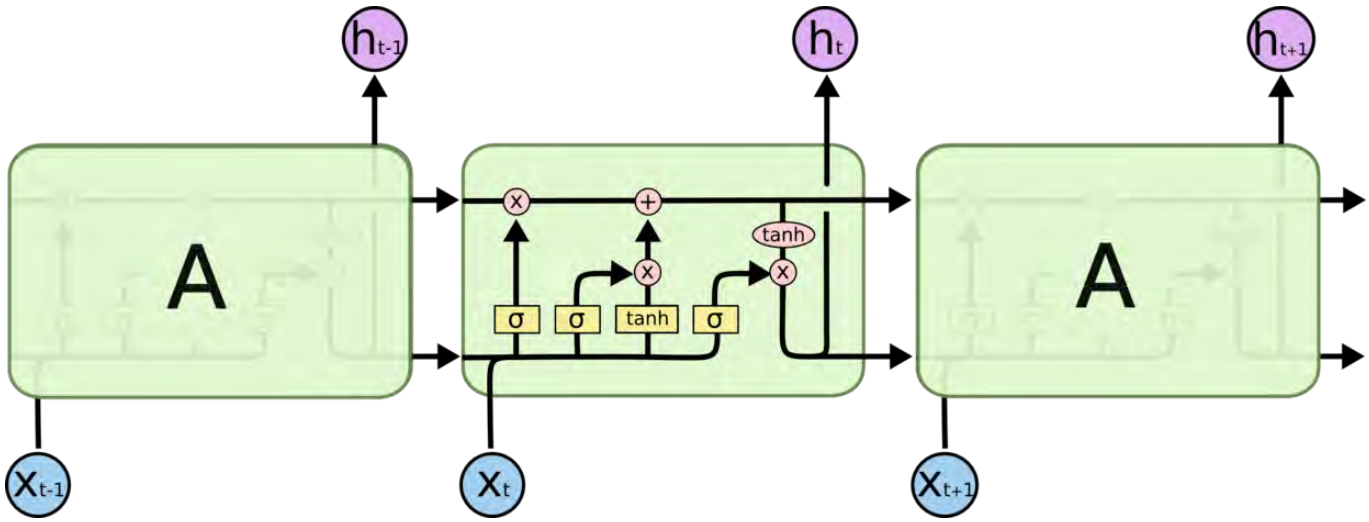
这里只是部分流行的 LSTM 变体。当然还有很多其他的，如Yao, et al. (2015) 提出的 Depth Gated RNN。还有用一些完全不同的观点来解决长期依赖的问题，如Koutnik, et al. (2014) 提出的 Clockwork RNN。
要问哪个变体是最好的？其中的差异性真的重要吗？Greff, et al. (2015) 给出了流行变体的比较，结论是他们基本上是一样的。Jozefowicz, et al. (2015) 则在超过 1 万种 RNN 架构上进行了测试，发现一些架构在某些任务上也取得了比 LSTM 更好的结果。

Jozefowicz等人论文截图

结论

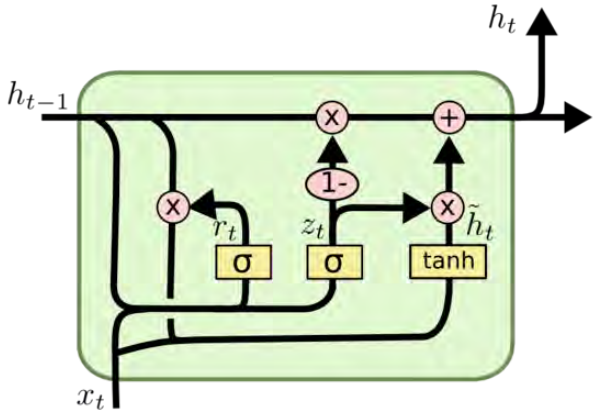
刚开始，我提到通过 RNN 得到重要的结果。本质上所有这些都可以使用 LSTM 完成。对于大多数任务确实展示了更好的性能！
由于 LSTM 一般是通过一系列的方程表示的，使得 LSTM 有一点令人费解。然而本文中一步一步地解释让这种困惑消除了不少。
LSTM 是我们在 RNN 中获得的重要成功。很自然地，我们也会考虑：哪里会有更加重大的突破呢？在研究人员间普遍的观点是：“Yes! 下一步已经有了——那就是注意力！” 这个想法是让 RNN 的每一步都从更加大的信息集中挑选信息。例如，如果你使用 RNN 来产生一个图片的描述，可能会选择图片的一个部分，根据这部分信息来产生输出的词。实际上，Xu, *et al.*(2015)已经这么做了——如果你希望深入探索注意力可能这就是一个有趣的起点！还有一些使用注意力的相当振奋人心的研究成果，看起来有更多的东西亟待探索……
注意力也不是 RNN 研究领域中唯一的发展方向。例如，Kalchbrenner, *et al.* (2015) 提出的 Grid LSTM 看起来也是很有前途。使用生成模型的 RNN，诸如Gregor, *et al.* (2015) Chung, *et al.* (2015) 和 Bayer & Osendorfer (2015) 提出的模型同样很有趣。在过去几年中，RNN 的研究已经相当的燃，而研究成果当然也会更加丰富！
再次说明下，本题解析基本取自Not_GOD翻译Christopher Olah 博文的《理解LSTM网络》，致谢。

RNN、LSTM、GRU区别。深度学习 DL模型 难
@我愛大泡泡，本题解析来源：http://blog.csdn.net/woaidapaopao/article/details/77806273

- RNN引入了循环的概念，但是在实际过程中却出现了初始信息随时间消失的问题，即长期依赖（Long-Term Dependencies）问题，所以引入了LSTM。
- LSTM：因为LSTM有进有出且当前的cell informaton是通过input gate控制之后叠加的，RNN是叠乘，因此LSTM可以防止梯度消失或者爆炸的变化是关键，下图非常明确适合记忆：



- GRU是LSTM的变体，将忘记门和输入们合成了一个单一的更新门。



$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right)$$
$$r_t = \sigma\left(W_r \cdot [h_{t-1}, x_t]\right)$$
$$\tilde{h}_t = \tanh\left(W \cdot [r_t * h_{t-1}, x_t]\right)$$
$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

当机器学习性能遭遇瓶颈时，你会如何优化的？机器学习 ML应用 难
可以从这4个方面进行尝试：、基于数据、借助算法、用算法调参、借助模型融合。当然能谈多细多深入就看你的经验心得了。这里有一份参考清单：机器学习性能改善备忘单。

如何提高深度学习的性能？深度学习 DL应用 难
http://blog.csdn.net/han_xiaoyang/article/details/52654879

做过什么样的机器学习项目？比如如何从零构建一个推荐系统。机器学习 ML应用 难
这里有一个推荐系统的公开课《推荐系统》，另，再推荐一个课程：机器学习项目班 [10次纯项目讲解，100%纯实战]。

什麽样的资料集不适合用深度学习？深度学习 DL应用 难
@抽象猴，来源：https://www.zhihu.com/question/41233373

1. 数据集太小，数据样本不足时，深度学习相对其它机器学习算法，没有明显优势。
2. 数据集没有局部相关特性，目前深度学习表现比较好的领域主要是图像 / 语音 / 自然语言处理等领域，这些领域的一个共性是局部相关性。图像中像素组成物体，语音信号中音位组合成单词，文本数据中单词组合成句子，这些特征元素的组合一旦被打乱，表示的含义同时也被改变。对于没有这样的局部相关性的数据集，不适于使用深度学习算法进行处理。举个例子：预测一个人的健康状况，相关的参数会有年龄、职业、收入、家庭状况等各种元素，将这些元素打乱，并不会影响相关的结果。

广义线性模型是怎被应用在深度学习中？深度学习 DL模型 中
@许韩，来源：https://www.zhihu.com/question/41233373/answer/145404190
A Statistical View of Deep Learning (I): Recursive GLMs
深度学习从统计学角度，可以看做递归的广义线性模型。
广义线性模型相对于经典的线性模型(y=wx+b)，核心在于引入了连接函数g(.)，形式变为：y=g?1(wx+b)。
深度学习时递归的广义线性模型，神经元的激活函数，即为广义线性模型的链接函数。逻辑回归（广义线性模型的一种）的Logistic函数即为神经元激活函数中的Sigmoid函数，很多类似的方法在统计学和神经网络中的名称不一样，容易引起初学者（这里主要指我）的困惑。下图是一个对照表

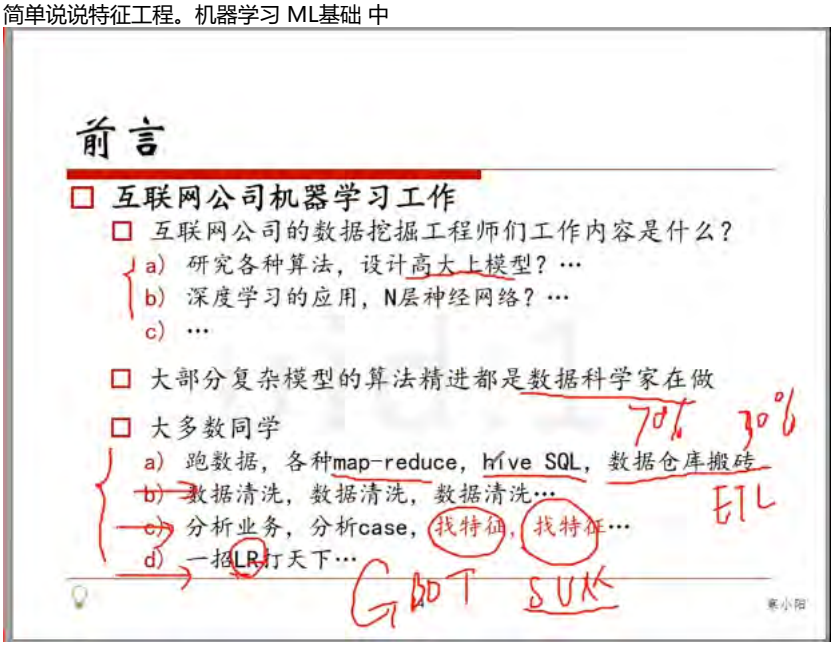
Target Type	Regression	Link	Inv link	Activation
Real	Linear	Identity	Identity	
Binary	Logistic	Logit $\log \frac{\mu}{1-\mu}$	Sigmoid σ $\frac{1}{1+\exp(-\eta)}$	Sigmoid
Binary	Probit	Inv Gauss CDF $\Phi^{-1}(\mu)$	Gauss CDF $\Phi(\eta)$	Probit
Binary	Gumbel	Compl. log-log $log(-log(\mu))$	Gumbel CDF $e^{-e^{-x}}$	
Binary	Logistic		Hyperbolic Tangent $\tanh(\eta)$	Tanh
Categorical	Multinomial		Multin. Logit $\frac{\eta_i}{\sum_j \eta_j}$	Softmax
Counts	Poisson	$log(\mu)$	$\exp(\nu)$	
Counts	Poisson	$\sqrt{(\mu)}$	ν^2	
Non-neg.	Gamma	Reciprocal $\frac{1}{\mu}$	$\frac{1}{\nu}$	
Sparse	Tobit		max $\max(0;\nu)$	ReLU
Ordered	Ordinal		Cum. Logit $\sigma(\phi_k - \eta)$	

准备机器学习面试应该了解哪些理论知识？机器学习 ML模型 中
@穆文，来源：https://www.zhihu.com/question/62482926

1. 【理论功底】主要考察对机器学习模型的理解，***选择性提问***（如果遇到面试者的研究方向自己不了解但感兴趣，会很欣喜，可以趁机学习一个哈哈）这块儿的问题会比较细碎，都是我自己深入思考过的（***背书是没用的，这里任何一个点我都可以给你展开问下去***），在此**全部手敲**
    1. 过拟合欠拟合（举几个例子让判断下，顺便问问交叉验证的目的、超参数搜索方法、EarlyStopping）、L1正则和L2正则的做法、正则化背后的思想（顺便问问BatchNorm、Covariance Shift）、L1正则产生稀疏解原理、逻辑回归为何线性模型（顺便问问LR如何解决低维不可分、从图模型角度看LR和朴素贝叶斯和无监督）、几种参数估计方法MLE/MAP/贝叶斯的联系和区别、简单说下SVM的支持向量（顺便问问KKT条件、为何对偶、核的通俗理解）、 GBDT随机森林能否并行（顺便问问bagging boosting）、 生成模型判别模型举个例子、聚类方法的掌握（顺便问问Kmeans的EM推导思路、谱聚类和Graph-cut的理解）、梯度下降类方法和牛顿类方法的区别（顺便问问Adam、L-BFGS的思路）、半监督的思想（顺便问问一些特定半监督算法是如何利用无标签数据的、从MAP角度看半监督）、常见的分类模型的评价指标（顺便问问交叉熵、ROC如何绘制、AUC的物理含义、类别不均衡样本）
    2. CNN中卷积操作和卷积核作用、maxpooling作用、卷积层与全连接层的联系、梯度爆炸和消失的概念（顺便问问神经网络权值初始化的方法、为何能减缓梯度爆炸消失、CNN中有哪些解决办法、LSTM如何解决的、如何梯度裁剪、dropout如何用在RNN系列网络中、dropout防止过拟合）、为何卷积可以用在图像/语音/语句上（顺便问问channel在不同类型数据源中的含义）
    3. 如果面试者跟我一样做NLP、推荐系统，我会继续追问 CRF跟逻辑回归 最大熵模型的关系、CRF的优化方法、CRF和MRF的联系、HMM和CRF的关系（顺便问问 朴素贝叶斯和HMM的联系、LSTM+CRF 用于序列标注的原理、CRF的点函数和边函数、CRF的经验分布）、WordEmbedding的几种常用方法和原理（顺便问问language model、perplexity评价指标、word2vec跟Glove的异同）、topic model说一说、为何CNN能用在文本分类、syntactic和semantic问题举例、常见Sentence embedding方法、注意力机制（顺便问问注意力机制的几种不同情形、为何引入、seq2seq原理）、序列标注的评价指标、语义消歧的做法、常见的跟word有关的特征、factorization machine、常见矩阵分解模型、如何把分类模型用于商品推荐（包括数据集划分、模型验证等）、序列学习、wide&deep model（顺便问问为何wide和deep）
2. 【**代码能力**】主要考察实现算法和优化代码的能力，我一般会先看面试者的github repo（如果简历给出来），看其代码风格、架构能力（**遇到大神会认真学习一个哈哈**），如果没有github，我会避免问典型的应试题，而是问一些 我本人从实际问题中抽象出的小算法题，比如：
    1. 给出节点的矩阵和边的矩阵，求路径和最大的路径（来源于 Viterbi 算法，本质就是个动态规划），至少给个思路和伪代码（顺便聊聊前向传播和反向传播）
    2. 给出一数组，数组元素是pair对儿，表示一个有向无环图的<父亲节点, 孩子节点>，用最优的方法，将其变成一个新的有序数组，数组元素是该有向无环图所有节点，数组的有序性体现在：父亲节点在孩子节点前面（来源于 贝叶斯网络实现时的小trick）
3. 【**项目能力**】主要考察解决实际问题的思路、填坑能力，这部分最考验面试官功底，要能从面试者浮夸的描述中寻找有意义的点，并一步步深挖。另外很多dirty work(数据预处理、文本清

看下来，这些问题的答案基本都在本BAT机器学习面试1000题系列里了。
标准化与归一化的区别？机器学习 ML基础 易
@艾华丰，本题解析来源：https://www.zhihu.com/question/20467170
归一化方法：
1、把数变为（0，1）之间的小数主要是为了数据处理方便提出来的，把数据映射到0～1范围之内处理，更加便捷快速。
2、把有量纲表达式变为无量纲表达式 归一化是一种简化计算的方式，即将有量纲的表达式，经过变换，化为无量纲的表达式，成为纯量。
标准化方法：　　数据的标准化是将数据按比例缩放，使之落入一个小的特定区间。由于信用指标体系的各个指标度量单位是不同的，为了能够将指标参与评价计算，需要对指标进行规范化处理，通过函数变换将其数值映射到某个数值区间。
随机森林如何处理缺失值？机器学习 ML模型 中
方法一（na.roughfix）简单粗暴，对于训练集,同一个class下的数据，如果是分类变量缺失，用众数补上，如果是连续型变量缺失，用中位数补。
方法二（rfImpute）这个方法计算量大，至于比方法一好坏？不好判断。先用na.roughfix补上缺失值，然后构建森林并计算proximity matrix，再回头看缺失值，如果是分类变量，则用没有缺失的观测实例的proximity中的权重进行投票。如果是连续型变量，则用proximity矩阵进行加权平均的方法补缺失值。然后迭代4-6次，这个补缺失值的思想和KNN有些类似12。
随机森林如何评估特征重要性？机器学习 ML模型 中
衡量变量重要性的方法有两种，Decrease GINI 和 Decrease Accuracy：
1) Decrease GINI：对于回归问题，直接使用argmax(VarVarLeftVarRight)作为评判标准，即当前节点训练集的方差Var减去左节点的方差VarLeft和右节点的方差VarRight。
2) Decrease Accuracy：对于一棵树Tb(x)，我们用OOB样本可以得到测试误差1；然后随机改变OOB样本的第j列：保持其他列不变，对第j列进行随机的上下置换，得到误差2。至此，我们可以用误差1-误差2来刻画变量j的重要性。基本思想就是，如果一个变量j足够重要，那么改变它会极大的增加测试误差；反之，如果改变它测试误差没有增大，则说明该变量不是那么的重要。
优化Kmeans？机器学习 ML模型 中
使用kd树或者ball tree
将所有的观测实例构建成一颗kd树，之前每个聚类中心都是需要和每个观测点做依次距离计算，现在这些聚类中心根据kd树只需要计算附近的一个局部区域即可
KMeans初始类簇中心点的选取。机器学习 ML模型 中
k-means++算法选择初始seeds的基本思想就是：初始的聚类中心之间的相互距离要尽可能的远。
1. 从输入的数据点集合中随机选择一个点作为第一个聚类中心
2. 对于数据集中的每一个点x，计算它与最近聚类中心(指已选择的聚类中心)的距离D(x)

3. 选择一个新的数据点作为新的聚类中心，选择的原则是：D(x)较大的点，被选取作为聚类中心的概率较大
4. 重复2和3直到k个聚类中心被选出来
5. 利用这k个初始的聚类中心来运行标准的k-means算法
解释对偶的概念。机器学习 ML基础 易
一个优化问题可以从两个角度进行考察，一个是primal 问题，一个是dual 问题，就是对偶问题，一般情况下对偶问题给出主问题最优值的下界，在强对偶性成立的情况下由对偶问题可以得到主问题的最优下界，对偶问题是凸优化问题，可以进行较好的求解，SVM中就是将primal问题转换为dual问题进行求解，从而进一步引入核函数的思想。
如何进行特征选择？机器学习 ML基础 中
特征选择是一个重要的数据预处理过程，主要有两个原因：一是减少特征数量、降维，使模型泛化能力更强，减少过拟合;二是增强对特征和特征值之间的理解
常见的特征选择方式：
1. 去除方差较小的特征
2. 正则化。1正则化能够生成稀疏的模型。L2正则化的表现更加稳定，由于有用的特征往往对应系数非零。
3. 随机森林，对于分类问题，通常采用基尼不纯度或者信息增益，对于回归问题，通常采用的是方差或者最小二乘拟合。一般不需要feature engineering、调参等繁琐的步骤。它的两个主要问题，1是重要的特征有可能得分很低（关联特征问题），2是这种方法对特征变量类别多的特征越有利（偏向问题）。
4. 稳定性选择。是一种基于二次抽样和选择算法相结合较新的方法，选择算法可以是回归、SVM或其他类似的方法。它的主要思想是在不同的数据子集和特征子集上运行特征选择算法，不断的重复，最终汇总特征选择结果，比如可以统计某个特征被认为是重要特征的频率（被选为重要特征的次数除以它所在的子集被测试的次数）。理想情况下，重要特征的得分会接近100%。稍微弱一点的特征得分会是非0的数，而最无用的特征得分将会接近于0。
数据预处理。机器学习 ML基础 易
1. 缺失值，填充缺失值fillna：
i. 离散：None,
ii. 连续：均值。
iii. 缺失值太多，则直接去除该列
2. 连续值：离散化。有的模型（如决策树）需要离散值
3. 对定量特征二值化。核心在于设定一个阈值，大于阈值的赋值为1，小于等于阈值的赋值为0。如图像操作
4. 皮尔逊相关系数，去除高度相关的列
简单说说特征工程。机器学习 ML基础 中

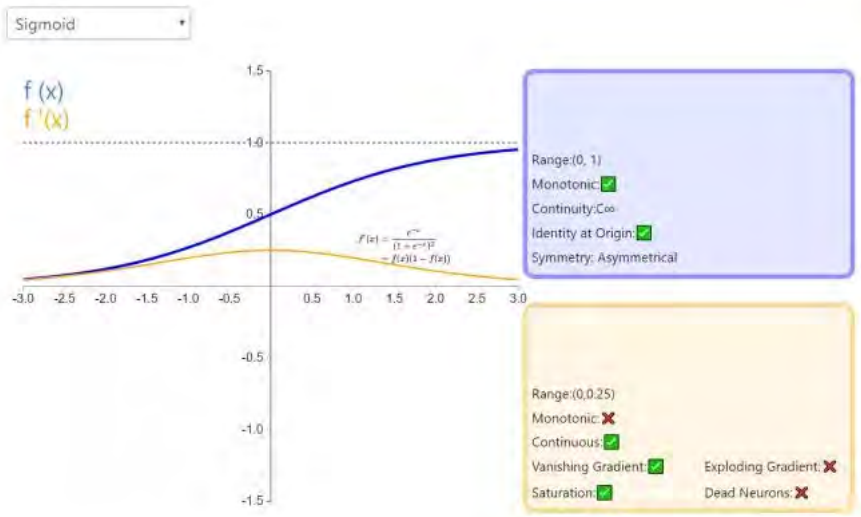


上图来源：http://www.julyedu.com/video/play/18

你知道有哪些数据处理和特征工程的处理？机器学习 ML应用 中

更多请查看此课程《机器学习工程师 第八期 [六大阶段、层层深入]》第7次课 特征工程。

请对比下Sigmoid、Tanh、ReLu这三个激活函数。深度学习 DL基础 中



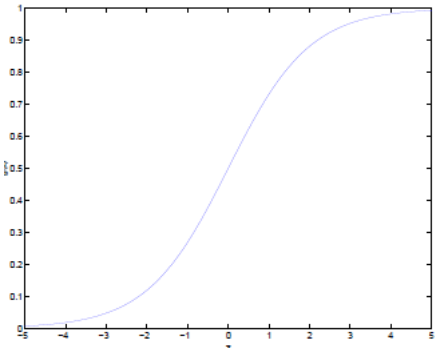
sigmoid函数又称logistic函数，应用在Logistic回归中。logistic回归的目的是从特征学习出一个0/1分类模型，而这个模型是将特性的线性组合作为自变量，由于自变量的取值范围是负无穷到正无穷。因此，使用logistic函数将自变量映射到(0,1)上，映射后的值被认为是属于y=1的概率。

假设函数

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

其中x是n维特征向量，函数g就是logistic函数。

而 $g(z) = \frac{1}{1 + e^{-z}}$ 的图像是

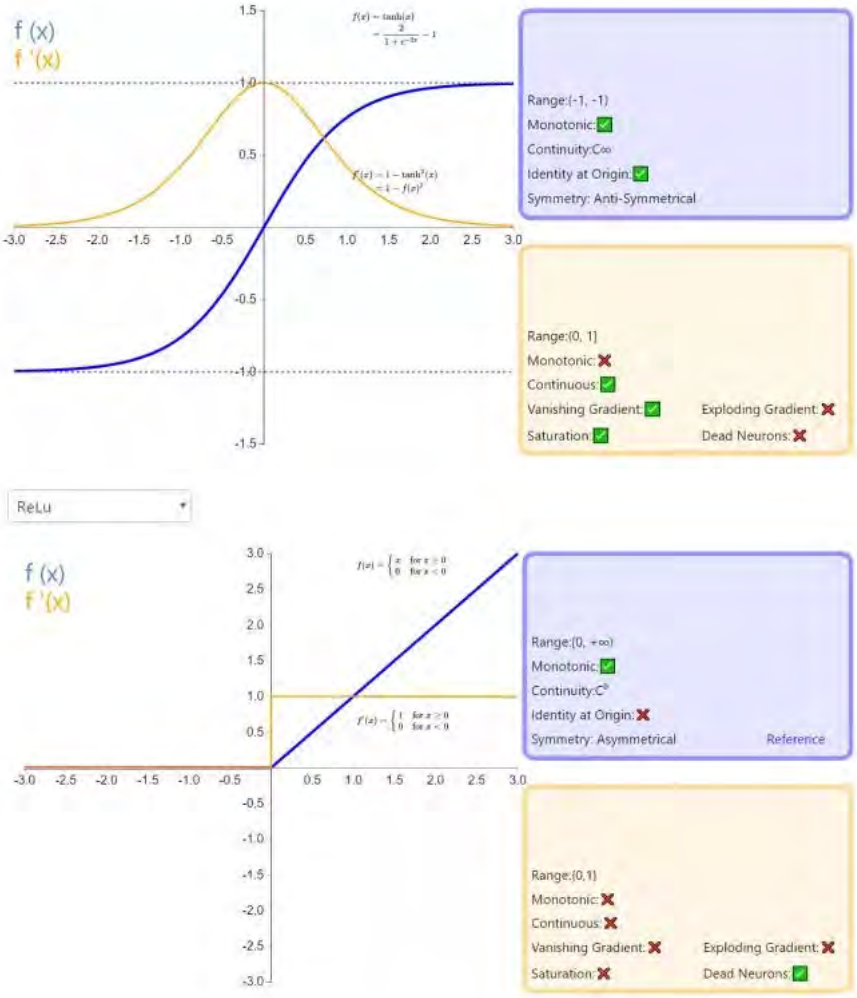


可以看到，将无穷映射到了(0,1)。

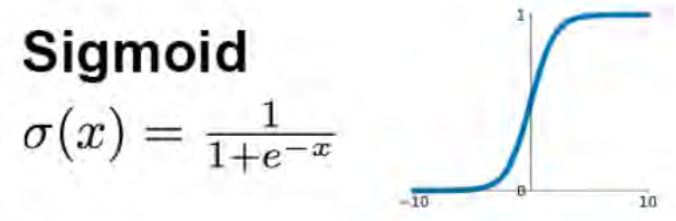
而假设函数就是特征属于y=1的概率。

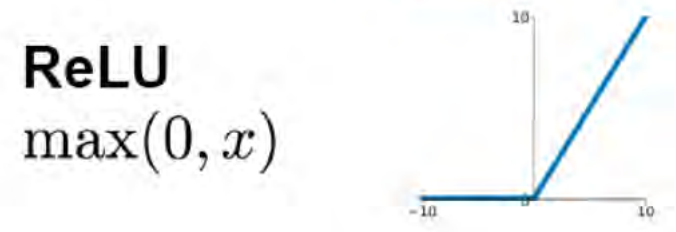
$$\begin{aligned} P(y = 1 \mid x; \theta) &= h_\theta(x) \\ P(y = 0 \mid x; \theta) &= 1 - h_\theta(x) \end{aligned}$$

从而，当我们要判别一个新来的特征属于哪个类时，只需求$h_\theta(x)$即可，若$h_\theta(x)$大于0.5就是y=1的类，反之属于y=0类。

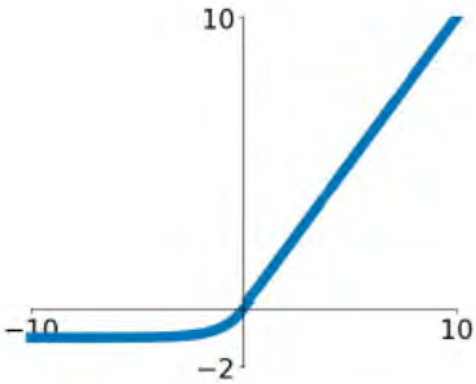


更多详见：https://mp.weixin.qq.com/s/7DgiXCNBS5vb07WIKTFYRQ
所以，sigmoid函数将输出映射到0-1范围之间，可以被看做是概率，因而，sigmoid函数是Logstic回归模型的激活函数。
但sigmoid函数有如下几个缺点：
正向计算包含指数，反向传播的导数也包含指数计算和除法运算，因而计算复杂度很高。
输出的均值非0。这样使得网络容易发生梯度消失或梯度爆炸。这也是batch normalization要解决的问题。
假如sigmoid函数为f(x)，那么f'(x)=f(x)(1-f(x))，因为f(x)输出在0-1之间，那么f'(x)恒大于0。 这就导致全部的梯度的正负号都取决于损失函数上的梯度。这样容易导致训练不稳定，参数一荣俱荣一损俱损。
同样的，f'(x)=f(x)(1-f(x))，因为f(x)输出在0-1之间，那么f'(x)输出也在0-1之间，当层次比较深时，底层的导数就是很多在0-1之间的数相乘，从而导致了梯度消失问题。
对于tanh来说，同sigmoid类似，但是输出值在-1到1之间，均值为0，是其相对于sigmoid的提升。但是因为输出在-1，1之间，因而输出不能被看做是概率。



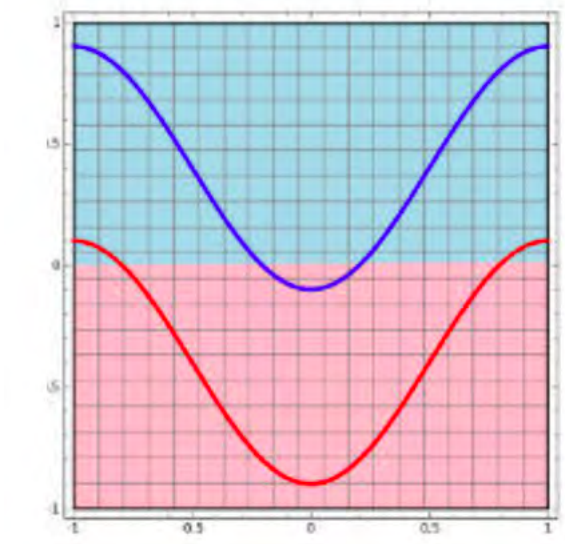


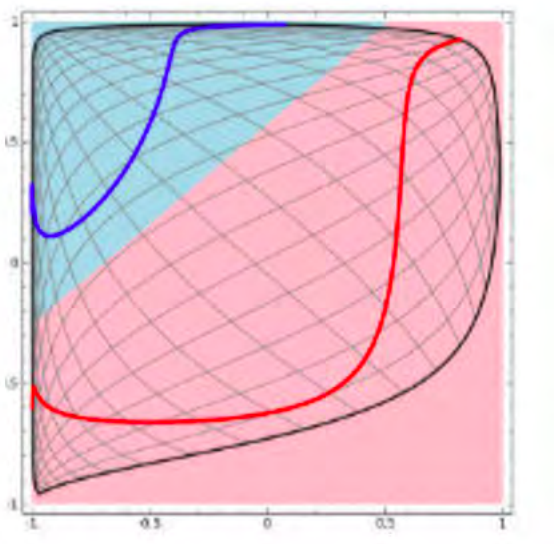
对于ReLU来说，相对于sigmoid和tanh来说，有如下优点：
计算量下，没有指数和除法运算。
不会饱和，因为在x>0的情况下，导数恒等于1
收敛速度快，在实践中可以得知，它的收敛速度是sigmoid的6倍。
Relu会使一部分神经元的输出为0，这样就造成了网络的稀疏性，并且减少了参数的相互依存关系，缓解了过拟合问题的发生
但是Relu也有缺点，缺点在于，
如果有一个特别大的导数经过神经单元使得输入变得小于0，这样会使得这个单元永远得不到参数更新，因为输入小于0时导数也是0. 这就形成了很多dead cell。Sigmoid、Tanh、ReLu这三个激活函数有什么缺点或不足，有没改进的激活函数。深度学习 DL基础 中
@张雨石：sigmoid、Tanh、ReLU的缺点在121问题中已有说明，为了解决ReLU的dead cell的情况，发明了Leaky Relu， 即在输入小于0时不让输出为0，而是乘以一个较小的系数，从而保证有导数存在。同样的目的，还有一个ELU，函数示意图如下。



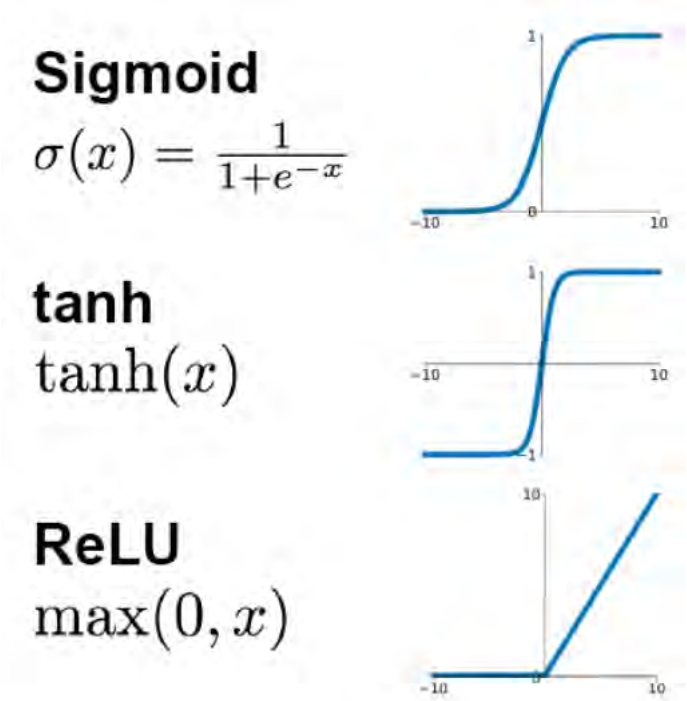
$$f(x) = \begin{cases} x & \text{if } x > 0 \\ \alpha\left(\exp(x) - 1\right) & \text{if } x \leq 0 \end{cases}$$

还有一个激活函数是Maxout，即使用两套w,b参数，输出较大值。本质上Maxout可以看做Relu的泛化版本，因为如果一套w,b全都是0的话，那么就是普通的ReLU。Maxout可以克服Relu的缺点，但是参数数目翻倍。
@我愛大泡泡，来源：http://blog.csdn.net/woaidapaopao/article/details/77806273怎么理解决策树、xgboost能处理缺失值？而有的模型(svm)对缺失值比较敏感。机器学习 ML模型 中
https://www.zhihu.com/question/58230411
为什么引入非线性激励函数？深度学习 DL基础 中
@张雨石：第一，对于神经网络来说，网络的每一层相当于f(wx+b)=f(w'x)，对于线性函数，其实相当于f(x)=x，那么在线性激活函数下，每一层相当于用一个矩阵去乘以x，那么多层就是反复的用矩阵去乘以输入。根据矩阵的乘法法则，多个矩阵相乘得到一个大矩阵。所以线性激励函数下，多层网络与一层网络相当。比如，两层的网络f(W1*f(W2x))=W1W2x=Wx。
第二，非线性变换是深度学习有效的原因之一。原因在于非线性相当于对空间进行变换，变换完成后相当于对问题空间进行简化，原来线性不可解的问题现在变得可以解了。
下图可以很形象的解释这个问题，左图用一根线是无法划分的。经过一系列变换后，就变成线性可解的问题了。
@Begin Again，来源：https://www.zhihu.com/question/29021768
如果不用激励函数（其实相当于激励函数是f(x) = x），在这种情况下你每一层输出都是上层输入的线性函数，很容易验证，无论你神经网络有多少层，输出都是输入的线性组合，与没有隐藏层效果相当，这种情况就是最原始的感知机（Perceptron）了。
正因为上面的原因，我们决定引入非线性函数作为激励函数，这样深层神经网络就有意义了（不再是输入的线性组合，可以逼近任意函数）。最早的想法是sigmoid函数或者tanh函数，输出有界，很容易充当下一层输入（以及一些人的生物解释）。

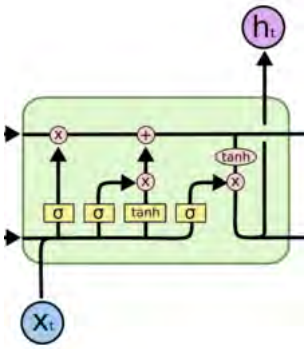




请问人工神经网络中为什么ReLu要好过于tanh和sigmoid function？深度学习 DL基础 中
先看sigmoid、tanh和RelU的函数图：



@Begin Again，来源：https://www.zhihu.com/question/29021768
第一，采用sigmoid等函数，算激活函数时（指数运算），计算量大，反向传播求误差梯度时，求导涉及除法和指数运算，计算量相对大，而采用Relu激活函数，整个过程的计算量节省很多。
第二，对于深层网络，sigmoid函数反向传播时，很容易就会出现梯度消失的情况（在sigmoid接近饱和区时，变换太缓慢，导数趋于0，这种情况会造成信息丢失），这种现象称为饱和，从而无法完成深层网络的训练。而ReLU就不会有饱和倾向，不会有特别小的梯度出现。
第三，Relu会使一部分神经元的输出为0，这样就造成了网络的稀疏性，并且减少了参数的相互依存关系，缓解了过拟合问题的发生（以及一些人的生物解释balabala）。当然现在也有一些对relu的改进，比如prelu，random relu等，在不同的数据集上会有一些训练速度上或者准确率上的改进，具体的大家可以找相关的paper看。
多加一句，现在主流的做法，会多做一步batch normalization，尽可能保证每一层网络的输入具有相同的分布[1]。而最新的paper[2]，他们在加入bypass connection之后，发现改变batch normalization的位置会有更好的效果。大家有兴趣可以看下。
[1] Ioffe S, Szegedy C. Batch normalization: Accelerating deep network training by reducing internal covariate shift[J]. arXiv preprint arXiv:15
[2] He, Kaiming, et al. "Identity Mappings in Deep Residual Networks." arXiv preprint arXiv:1603.05027 (2016).
为什么LSTM模型中既存在sigmoid又存在tanh两种激活函数？深度学习 DL模型 难
为什么不是选择统一一种sigmoid或者tanh，而是混合使用呢？这样的目的是什么？



本题解析来源：https://www.zhihu.com/question/46197687
@beanfrog：二者目的不一样
sigmoid 用在了各种gate上，产生0~1之间的值，这个一般只有sigmoid最直接了。
tanh 用在了状态和输出上，是对数据的处理，这个用其他激活函数或许也可以。
@hhhh：另可参见A Critical Review of Recurrent Neural Networks for Sequence Learning的section4.1，说了那两个tanh都可以替换成别的。
衡量分类器的好坏？机器学习 ML基础 中
@我愛大泡泡，来源：http://blog.csdn.net/woaidapaopao/article/details/77806273
这里首先要知道TP、FN（真的判成假的）、FP（假的判成真）、TN四种（可以画一个表格）。
几种常用的指标：

- 精度precision = TP/(TP+FP) = TP/~P （~p为预测为真的数量）
- 召回率 recall = TP/(TP+FN) = TP/ P
- F1值： 2/F1 = 1/recall + 1/precision
- ROC曲线：ROC空间是一个以伪阳性率（FPR，false positive rate）为X轴，真阳性率（TPR, true positive rate）为Y轴的二维坐标系所代表的平面。其中真阳率TPR = TP / P = recall， 伪阳率FPR = FP / N
  更详细请点击：
  https://siyaozhang.github.io/2017/04/04/%E5%87%86%E7%A1%AE%E7%8E%87%E3%80%81%E5%8F%AC%E5%9B%9E%E7%8E%87%E3%80%8

机器学习和统计里面的auc的物理意义是啥？机器学习 ML基础 中
https://www.zhihu.com/question/39840928
观察增益gain，alpha和gamma越大，增益越小？机器学习 ML基础 中
@AntZ：xgboost寻找分割点的标准是最大化gain. 考虑传统的枚举每个特征的所有可能分割点的贪心法效率太低，xgboost实现了一种近似的算法。大致的思想是根据百分位法列举几个可能成为分割点的候选者，然后从候选者中计算Gain按最大值找出最佳的分割点。它的计算公式分为四项，可以由正则化项参数调整(lamda为叶子权重平方

和的系数，gama为叶子数量）：

objective after adding the split is

$$Gain = \frac{G_L^2}{H_L+\lambda} + \frac{G_R^2}{H_R+\lambda} - \frac{(G_L+G_R)^2}{H_L+H_R+\lambda} - \gamma$$

The complexity cost by introducing additional leaf

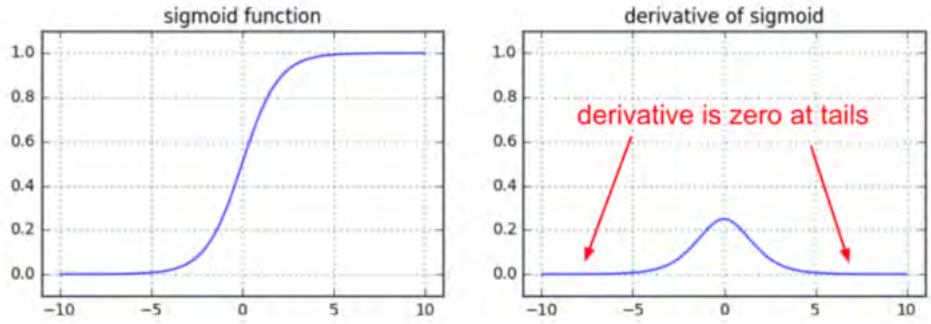
the score of left child

the score of right child

the score of if we do not split

第一项是假设分割的左孩子的权重分数，第二项为右孩子，第三项为不分割总体分数，最后一项为引入一个节点的复杂度损失
由公式可知，gama越大gain越小，lamda越大，gain可能小也可能大.
原问题是alpha而不是lambda，这里paper上没有提到，xgboost实现上有这个参数. 上面是我从paper上理解的答案,下面是搜索到的：
https://zhidao.baidu.com/question/2121727290086699747.html?fr=iks&word=xgboost+lamda&ie=gbk
lambda[默认1]权重的L2正则化项。(和Ridge regression类似)。 这个参数是用来控制XGBoost的正则化部分的。虽然大部分数据科学家很少用到这个参数，但是这个参数在减少过拟合上还是可以挖掘出更多用处的。11、alpha[默认1]权重的L1正则化项。(和Lasso regression类似)。 可以应用在很高维度的情况下，使得算法的速度更快。
gamma[默认0]在节点分裂时，只有分裂后损失函数的值下降了，才会分裂这个节点。Gamma指定了节点分裂所需的最小损失函数下降值。 这个参数的值越大，算法越保守。
什麽造成梯度消失问题? 推导一下。深度学习 DL基础 中
@许韩，来源：https://www.zhihu.com/question/41233373/answer/145404190

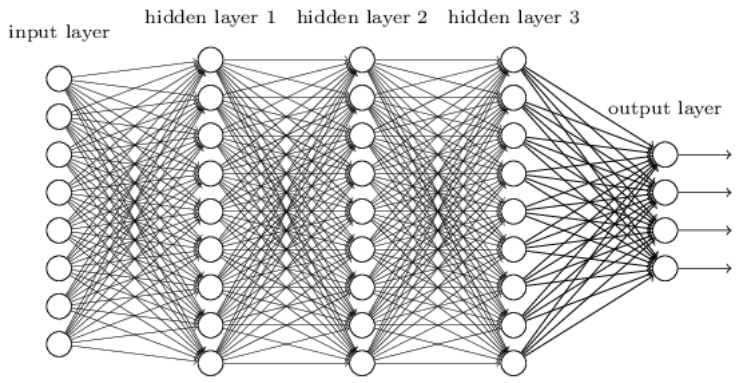
- **Yes you should understand backdrop - Andrej Karpathy**
- How does the ReLu solve the vanishing gradient problem?
- 神经网络的训练中，通过改变神经元的权重，使网络的输出值尽可能逼近标签以降低误差值，训练普遍使用BP算法，核心思想是，计算出输出与标签间的损失函数值，然后计算其相对于每个神经元的梯度，进行权值的迭代。
- 梯度消失会造成权值更新缓慢，模型训练难度增加。造成梯度消失的一个原因是，许多激活函数将输出值挤压在很小的区间内，在激活函数两端较大范围的定义域内梯度为0，造成学习停止。



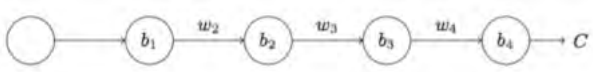
@张雨石：简而言之，就是sigmoid函数f(x)的导数为f(x)*(1-f(x))， 因为f(x)的输出在0-1之间，所以随着深度的增加，从顶端传过来的导数每次都乘以两个小于1的数，很快就变得特别特别小。

什么是梯度消失和梯度爆炸？深度学习 DL基础 中
@寒小阳，反向传播中链式法则带来的连乘，如果有数很小趋于0，结果就会特别小（梯度消失）；如果数都比较大，可能结果会很大（梯度爆炸）。
@单车，下段来源：https://zhuanlan.zhihu.com/p/25631496
层数比较多的神经网络模型在训练时也是会出现一些问题的，其中就包括梯度消失问题（gradient vanishing problem）和梯度爆炸问题（gradient exploding problem）。梯度消失问题和梯度爆炸问题一般随着网络层数的增加会变得越来越明显。

例如，对于下图所示的含有3个隐藏层的神经网络，梯度消失问题发生时，接近于输出层的hidden layer 3等的权值更新相对正常，但前面的hidden layer 1的权值更新会变得很慢，导致前面的层权值几乎不变，仍接近于初始化的权值，这就导致hidden layer 1相当于只是一个映射层，对所有的输入做了一个同一映射，这是此深层网络的学习就等价于只有后几层的浅层网络的学习了。



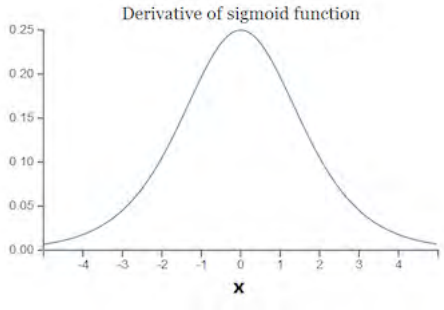
而这种问题为何会产生呢？以下图的反向传播为例（假设每一层$y_i = \sigma(z_i) = \sigma(w_i x_i + b_i)$，其中$\sigma$为sigmoid函数）



可以推导出

$$\frac{\partial C}{\partial b_1} = \frac{\partial C}{\partial y_4}\frac{\partial y_4}{\partial z_4}\frac{\partial z_4}{\partial x_4}\frac{\partial x_4}{\partial z_3}\frac{\partial z_3}{\partial x_3}\frac{\partial x_3}{\partial z_2}\frac{\partial z_2}{\partial x_2}\frac{\partial x_2}{\partial z_1}\frac{\partial z_1}{\partial b_1}$$
$$= \frac{\partial C}{\partial y_4}\sigma'(z_4)\,w_4\sigma'(z_3)\,w_3\sigma'(z_2)\,w_2\sigma'(z_1)$$

而sigmoid的导数$\sigma'(x)$如下图



可见，$\sigma'(x)$的最大值为$\frac{1}{4}$，而我们初始化的网络权值$|w|$通常都小于1，因此$|\sigma'(z)\,w| \le \frac{1}{4}$，因此对于上面的链式求导，层数越多，求导结果$\frac{\partial C}{\partial b_1}$越小，因而导致梯度消失的情况出现。

这样，梯度爆炸问题的出现原因就显而易见了，即$|\sigma'(z)\,w| > 1$，也就是w比较大的情况。但对于使用sigmoid激活函数来说，这种情况比较少。因为$\sigma'(z)$的大小也与w有关（$z = wx + b$），除非该层的输入值x在一直一个比较小的范围内。

其实梯度爆炸和梯度消失问题都是因为网络太深，网络权值更新不稳定造成的，本质上是因为梯度反向传播中的连乘效应。对于更普遍的梯度消失问题，可以考虑用ReLU激活函数取代sigmoid激活函数。另外，LSTM的结构设计也可以改善RNN中的梯度消失问题。

如何解决梯度消失和梯度膨胀？深度学习 DL基础 中
（1）梯度消失：
根据链式法则，如果每一层神经元对上一层的输出的偏导乘上权重结果都小于1的话，那么即使这个结果是0.99，在经过足够多层传播之后，误差对输入层的偏导会趋于0
可以采用ReLU激活函数有效的解决梯度消失的情况，也可以用Batch Normalization解决这个问题。关于深度学习中 Batch Normalization为什么效果好？参见：
https://www.zhihu.com/question/38102762
（2）梯度膨胀
根据链式法则，如果每一层神经元对上一层的输出的偏导乘上权重结果都大于1的话，在经过足够多层传播之后，误差对输入层的偏导会趋于无穷大
可以通过激活函数来解决，或用Batch Normalization解决这个问题。

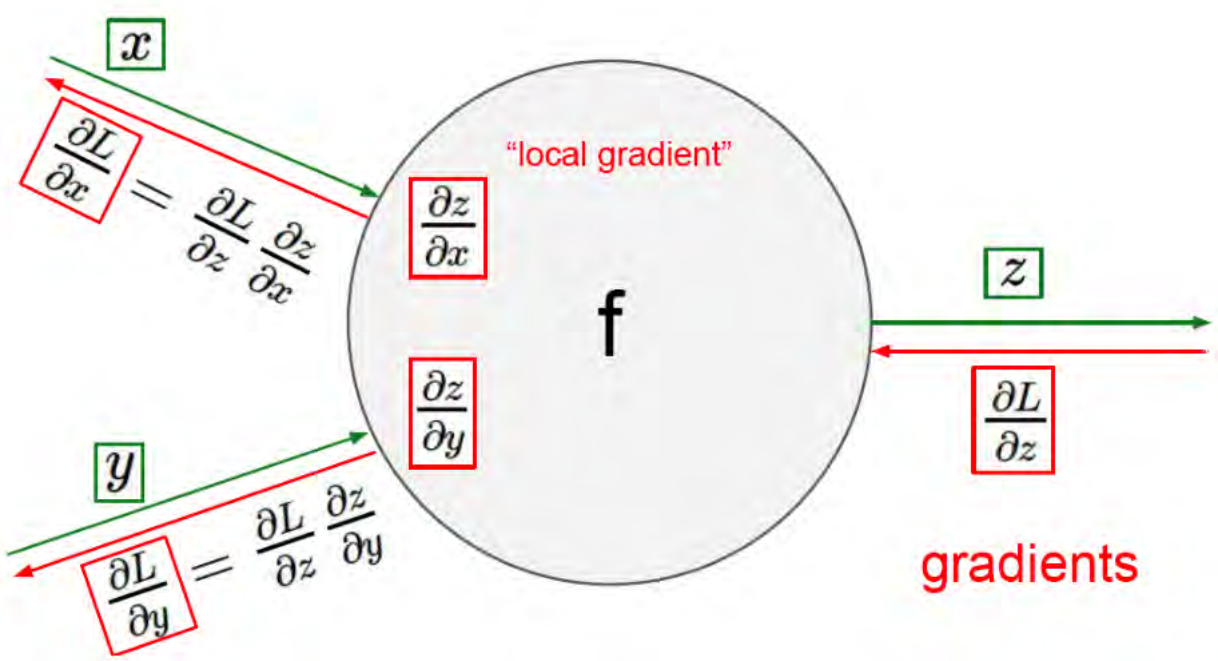
推导下反向传播Backpropagation。深度学习 DL基础 难
@我愛大泡泡，来源：http://blog.csdn.net/woaidapaopao/article/details/77806273

首先，要理解反向传播的基本原理，那就是求导的链式法则。

$$\frac{dy}{dx} = \frac{dy}{du}\cdot\frac{du}{dx}$$

$$\frac{dy}{dz} = \frac{dy}{du}\cdot\frac{du}{dx}\cdot\frac{dx}{dz}$$

反映到神经网络里：



下面从损失函数开始用公式进行推导。
反向传播是在求解损失函数L对参数w求导时候用到的方法，目的是通过链式法则对参数进行一层一层的求导。这里重点强调：要将参数进行随机初始化而不是全部置0，否则所有隐层的数值都会与输入相关，这称为对称失效。
大致过程是:

- 首先前向传导计算出所有节点的激活值和输出值，
  $z^{(l+1)} = W^{(l)}a^{(l)} + b^{(l)}$
  $a^{(l+1)} = f(z^{(l+1)})$
- 计算整体损失函数：
  $$J(W,b) = \left[\frac{1}{m}\sum_{i=1}^{m} J(W,b;x^{(i)},y^{(i)})\right] + \frac{\lambda}{2}\sum_{l=1}^{n_l-1}\sum_{i=1}^{s_l}\sum_{j=1}^{s_{l+1}}\left(W_{ji}^{(l)}\right)^2$$
  $$= \left[\frac{1}{m}\sum_{i=1}^{m}\left(\frac{1}{2}\left\|h_{W,b}(x^{(i)}) - y^{(i)}\right\|^2\right)\right] + \frac{\lambda}{2}\sum_{l=1}^{n_l-1}\sum_{i=1}^{s_l}\sum_{j=1}^{s_{l+1}}\left(W_{ji}^{(l)}\right)^2$$
- 然后针对第L层的每个节点计算出残差（这里是因为UFLDL中说的是残差，本质就是整体损失函数对每一层激活值Z的导数），所以要对W求导只要再乘上激活函数对W的导数即可
  $$\delta_i^{(l)} = \left(\sum_{j=1}^{s_{l+1}} W_{ji}^{(l)}\delta_j^{(l+1)}\right) f'(z_i^{(l)})$$
  $$\frac{\partial}{\partial W_{ij}^{(l)}} J(W,b;x,y) = a_j^{(l)}\delta_i^{(l+1)}$$
  $$\frac{\partial}{\partial b_i^{(l)}} J(W,b;x,y) = \delta_i^{(l+1)}.$$

SVD和PCA。机器学习 ML模型 中

PCA的理念是使得数据投影后的方差最大，找到这样一个投影向量，满足方差最大的条件即可。而经过了去除均值的操作之后，就可以用SVD分解来求解这样一个投影向量，选择特征值最大的方向。
PCA的本质是对于一个以矩阵为参数的分布进行似然估计，而SVD是矩阵近似的有效手段。
详见：https://www.zhihu.com/question/40043805

数据不平衡问题。机器学习 ML基础 易

这主要是由于数据分布不平衡造成的。解决方法如下：

- 采样，对小样本加噪声采样，对大样本进行下采样
- 数据生成，利用已知样本生成新的样本
- 进行特殊的加权，如在Adaboost中或者SVM中
- 采用对不平衡数据集不敏感的算法
- 改变评价标准：用AUC/ROC来进行评价
- 采用Bagging/Boosting/ensemble等方法
- 在设计模型的时候考虑数据的先验分布

简述神经网络的发展历史。深度学习 DL基础 中
1949年Hebb提出了神经心理学学习范式——Hebbian学习理论
1952年，IBM的Arthur Samuel写出了西洋棋程序
1957年，Rosenblatt的感知器算法是第二个有着神经系统科学背景的机器学习模型.
3年之后，Widrow因发明Delta学习规则而载入ML史册，该规则马上就很好的应用到了感知器的训练中
感知器的热度在1969被Minskey一盆冷水泼灭了。他提出了著名的XOR问题，论证了感知器在类似XOR问题的线性不可分数据的无力。

尽管BP的思想在70年代就被Linnainmaal以“自动微分的翻转模式”被提出来，但直到1981年才被Werbos应用到多层感知器(MLP)中，NN新的大繁荣。
1991年的Hochreiter和2001年的Hochreiter的工作，都表明在使用BP算法时，NN单元饱和之后会发生梯度损失。又发生停滞。
时间终于走到了当下，随着计算资源的增长和数据量的增长。一个新的NN领域——深度学习出现了。
简言之，MP模型+sgn—->单层感知机（只能线性）+sgn— Minsky 低谷 —>多层感知机+BP+sigmoid—- (低谷) —>深度学习+pre-training+ReLU/sigmoid
深度学习常用方法。深度学习 DL基础 中
@SmallisBig，来源：http://blog.csdn.net/u010496169/article/details/73550487
全连接DNN（相邻层相互连接、层内无连接）：
AutoEncoder(尽可能还原输入)、Sparse Coding（在AE上加入L1规范）、RBM（解决概率问题）——>特征探测器——>栈式叠加 贪心训练
RBM—->DBN
解决全连接DNN的全连接问题——>CNN
解决全连接DNN的无法对时间序列上变化进行建模的问题——>RNN—解决时间轴上的梯度消失问题——->LSTM
@张雨石：现在在应用领域应用的做多的是DNN，CNN和RNN。
DNN是传统的全连接网络，可以用于广告点击率预估，推荐等。其使用embedding的方式将很多离散的特征编码到神经网络中，可以很大的提升结果。
CNN主要用于计算机视觉(Computer Vision)领域，CNN的出现主要解决了DNN在图像领域中参数过多的问题。同时，CNN特有的卷积、池化、batch normalization、Inception、ResNet、DeepNet等一系列的发展也使得在分类、物体检测、人脸识别、图像分割等众多领域有了长足的进步。同时，CNN不仅在图像上应用很多，在自然语言处理上也颇有进展，现在已经有基于CNN的语言模型能够达到比LSTM更好的效果。在最新的AlphaZero中，CNN中的ResNet也是两种基本算法之一。
GAN是一种应用在生成模型的训练方法，现在有很多在CV方面的应用，例如图像翻译，图像超清化、图像修复等等。
RNN主要用于自然语言处理(Natural Language Processing)领域，用于处理序列到序列的问题。普通RNN会遇到梯度爆炸和梯度消失的问题。所以现在在NLP领域，一般会使用LSTM模型。在最近的机器翻译领域，Attention作为一种新的手段，也被引入进来。
除了DNN、RNN和CNN外， 自动编码器(AutoEncoder)、稀疏编码(Sparse Coding)、深度信念网络(DBM)、限制玻尔兹曼机(RBM)也都有相应的研究。

神经网络模型（Neural Network）因受人类大脑的启发而得名。深度学习 DL基础 易

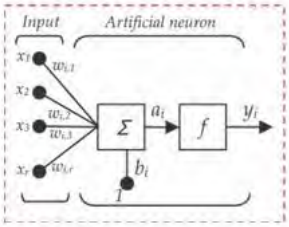
神经网络由许多神经元（Neuron）组成，每个神经元接受一个输入，对输入进行处理后给出一个输出，如下图所示。请问下列关于神经元的描述中，哪一项是正确的？

1. A 每个神经元可以有一个输入和一个输出
2. B 每个神经元可以有多个输入和一个输出
3. C 每个神经元可以有一个输入和多个输出
4. D 每个神经元可以有多个输入和多个输出
5. E 上述都正确

答案：（E）

每个神经元可以有一个或多个输入，和一个或多个输出。

下图是一个神经元的数学表示。深度学习 DL基础 易



这些组成部分分别表示为：

- x1, x2,…, xN：表示神经元的输入。可以是输入层的实际观测值，也可以是某一个隐藏层（Hidden Layer）的中间值

- w1, w2,…,wN：表示每一个输入的权重

- bi：表示偏差单元/偏移量（bias unit）。作为常数项加到激活函数的输入当中，类似截距（Intercept）

- a：作为神经元的激励函数（Activation），可以表示为

$$a = f(\sum_{i=0}^{N} w_i x_i)$$

- y：神经元输出

考虑上述标注，线性等式（y = mx + c）可以被认为是属于神经元吗：

A. 是

B. 否

答案：（A）

输入只有一个变量，激活函数为线性。所以可以被认为是线性回归函数。

在一个神经网络中，知道每一个神经元的权重和偏差是最重要的一步。如果知道了神经元准确的权重和偏差，便可以近似任何函数，但怎么获知每个神经的权重和偏移呢？
深度学习 DL基础 易
A 搜索每个可能的权重和偏差组合，直到得到最佳值
B 赋予一个初始值，然后检查跟最佳值的差值，不断迭代调整权重
C 随机赋值，听天由命
D 以上都不正确的
答案：（B）
选项B是对梯度下降的描述。

梯度下降算法的正确步骤是什么？深度学习 DL基础 易

1. 计算预测值和真实值之间的误差
2. 重复迭代，直至得到网络权重的最佳值
3. 把输入传入网络，得到输出值
4. 用随机值初始化权重和偏差
5. 对每一个产生误差的神经元，调整相应的（权重）值以减小误差

A. 1, 2, 3, 4, 5

B. 5, 4, 3, 2, 1

C. 3, 2, 1, 5, 4

D. 4, 3, 1, 5, 2

答案：（D）

已知：
- 大脑是有很多个叫做神经元的东西构成，神经网络是对大脑的简单的数学表达。
- 每一个神经元都有输入、处理函数和输出。
- 神经元组合起来形成了网络，可以拟合任何函数。
- 为了得到最佳的神经网络，我们用梯度下降方法不断更新模型

给定上述关于神经网络的描述，什么情况下神经网络模型被称为深度学习模型？深度学习 DL基础 易
A 加入更多层，使神经网络的深度增加
B 有维度更高的数据
C 当这是一个图形识别的问题时
D 以上都不正确
答案：（A）
更多层意味着网络更深。没有严格的定义多少层的模型才叫深度模型，目前如果有超过2层的隐层，那么也可以及叫做深度模型。

使用CNN时，是否需要对输入进行旋转、平移、缩放等预处理？深度学习 DL基础 易
A 需要
B 不需要
答案：（A）
把数据传入神经网络之前需要做一系列数据预处理（也就是旋转、平移、缩放）工作，神经网络本身不能完成这些变换。

下面哪项操作能实现跟神经网络中Dropout的类似效果？（B）深度学习 DL基础 易
A Boosting
B Bagging
C Stacking
D Mapping
Dropout可以认为是一种极端的Bagging，每一个模型都在单独的数据上训练，同时，通过和其他模型对应参数的共享，从而实现模型参数的高度正则化。

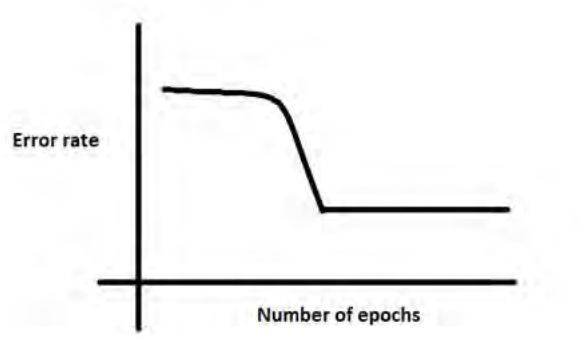
下列哪一项在神经网络中引入了非线性？深度学习 DL基础 易

1. A 随机梯度下降
2. B 修正线性单元（ReLU）
3. C 卷积函数
4. D 以上都不正确

答案：（B）

修正线性单元是非线性的激活函数。

在训练神经网络时，损失函数(loss)在最初的几个epochs时没有下降，可能的原因是？（D）深度学习 DL基础 易



A 学习率(learning rate)太低
B 正则参数太高
C 陷入局部最小值
D 以上都有可能

下列哪项关于模型能力（model capacity）的描述是正确的？（指神经网络模型能拟合复杂函数的能力）深度学习 DL基础 易

1. A 隐藏层层数增加，模型能力增加
2. B Dropout的比例增加，模型能力增加
3. C 学习率增加，模型能力增加
4. D 都不正确

答案：（A）

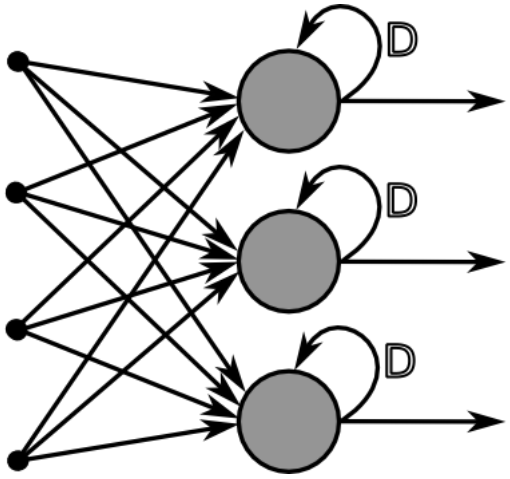
如果增加多层感知机（Multilayer Perceptron）的隐藏层层数，分类误差便会减小。这种陈述正确还是错误？深度学习 DL基础 易

1. A 正确
2. B 错误

答案：（B）

并不总是正确。层数增加可能导致过拟合，从而可能引起错误增加。

构建一个神经网络，将前一层的输出和它自身作为输入。深度学习 DL模型 易



下列哪一种架构有反馈连接？

1. A 循环神经网络
2. B 卷积神经网络
3. C 限制玻尔兹曼机
4. D 都不是

答案：（A）

在感知机中（Perceptron）的任务顺序是什么？深度学习 DL基础 易
1 随机初始化感知机的权重

2 去到数据集的下一批（batch）
3 如果预测值和输出不一致，则调整权重
4 对一个输入样本，计算输出值

A. 1, 2, 3, 4
B. 4, 3, 2, 1
C. 3, 1, 2, 4
D. 1, 4, 3, 2
答案：（D）

假设你需要调整参数来最小化代价函数（cost function），会使用下列哪项技术？深度学习 DL基础 易

A. 穷举搜索

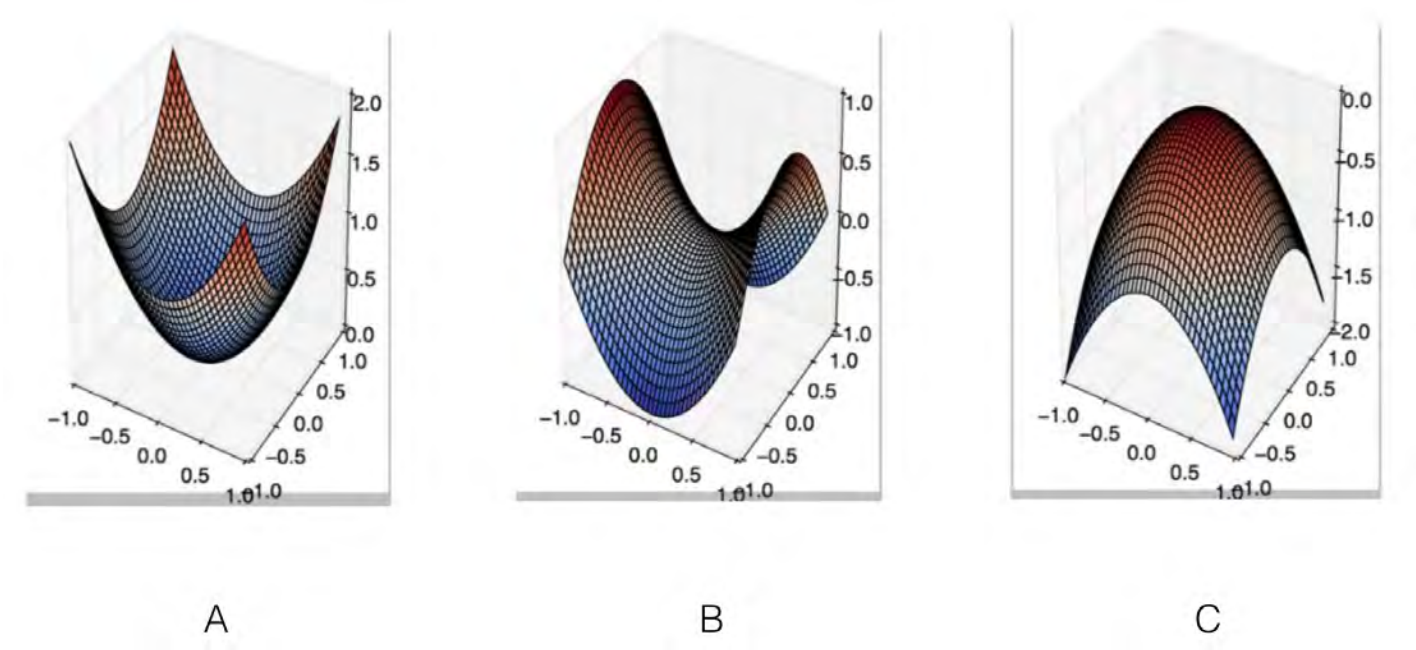
B. 随机搜索

C. Bayesian优化

D. 梯度下降

答案：（D）

在下面哪种情况下，一阶梯度下降不一定正确工作（可能会卡住）？深度学习 DL基础 易

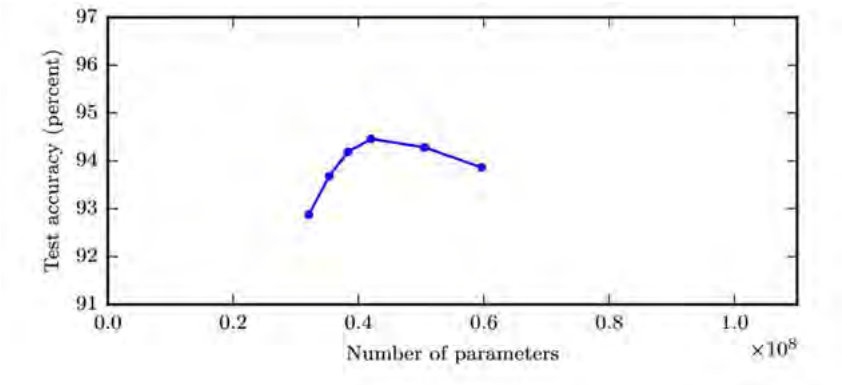


D. 以上都不正确

答案：（B）

这是鞍点（Saddle Point）的梯度下降的经典例子。另，本题来源于：https://www.analyticsvidhya.com/blog/2017/01/must-know-questions-deep-learning/。

下图显示了训练过的3层卷积神经网络准确度，与参数数量(特征核的数量)的关系。深度学习 DL基础 易



从图中趋势可见，如果增加神经网络的宽度，精确度会增加到一个特定阈值后，便开始降低。造成这一现象的可能原因是什么？

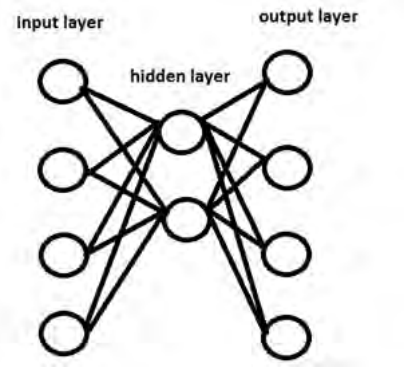
1. A 即使增加卷积核的数量，只有少部分的核会被用作预测
2. B 当卷积核数量增加时，神经网络的预测能力（Power）会降低

3. C 当卷积核数量增加时，导致过拟合
4. D 以上都不正确

答案：（C）

网络规模过大时，就可能学到数据中的噪声，导致过拟合

假设我们有一个如下图所示的隐藏层。隐藏层在这个网络中起到了一定的降纬作用。假如现在我们用另一种维度下降的方法，比如说主成分分析法(PCA)来替代这个隐藏层。
深度学习 DL基础 易



那么，这两者的输出效果是一样的吗？

A.是

B.否

答案：（B）

PCA 提取的是数据分布方差比较大的方向，隐藏层可以提取有预测能力的特征

下列哪个函数不可以做激活函数？深度学习 DL基础 易

A. y = tanh(x)
B. y = sin(x)
C. y = max(x,0)
D. y = 2x

答案：（D）

线性函数不能作为激活函数。

下列哪个神经网络结构会发生权重共享？深度学习 DL模型 易

A.卷积神经网络

B.循环神经网络

C.全连接神经网络

D.选项A和B

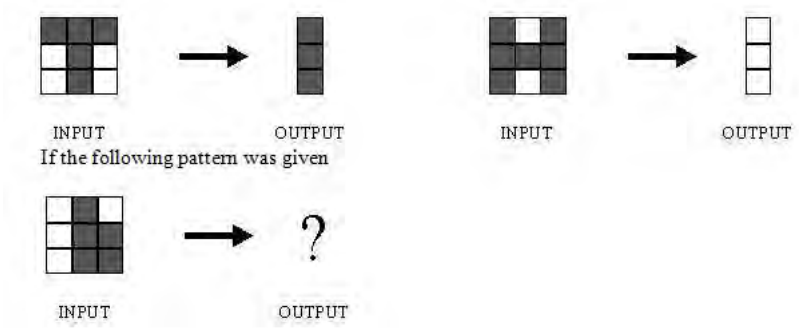
答案：（D）

批规范化(Batch Normalization)的好处都有啥？深度学习 DL基础 中
A.在将所有的输入传递到下一层之前对其进行归一化（更改）
B.它将权重的归一化平均值和标准差
C.它是一种非常有效的反向传播(BP)方法
D.这些均不是
答案：（A）

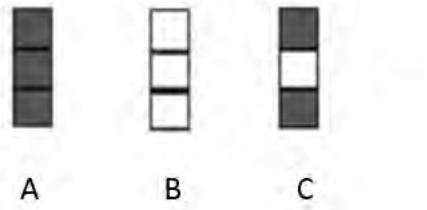
在一个神经网络中，下面哪种方法可以用来处理过拟合？（D） 深度学习 DL基础 易
A Dropout
B 分批归一化(Batch Normalization)
C 正则化(regularization)
D 都可以
对于选项C，分批归一化处理过拟合的原理，是因为同一个数据在不同批中被归一化后的值会有差别，相当于做了data augmentatio。

如果我们用了一个过大的学习速率会发生什么？深度学习 DL基础 易
A 神经网络会收敛
B 不好说
C 都不对
D 神经网络不会收敛

下图所示的网络用于训练识别字符H和T，如下所示（深度学习 DL基础 易）：



网络的输出是什么？



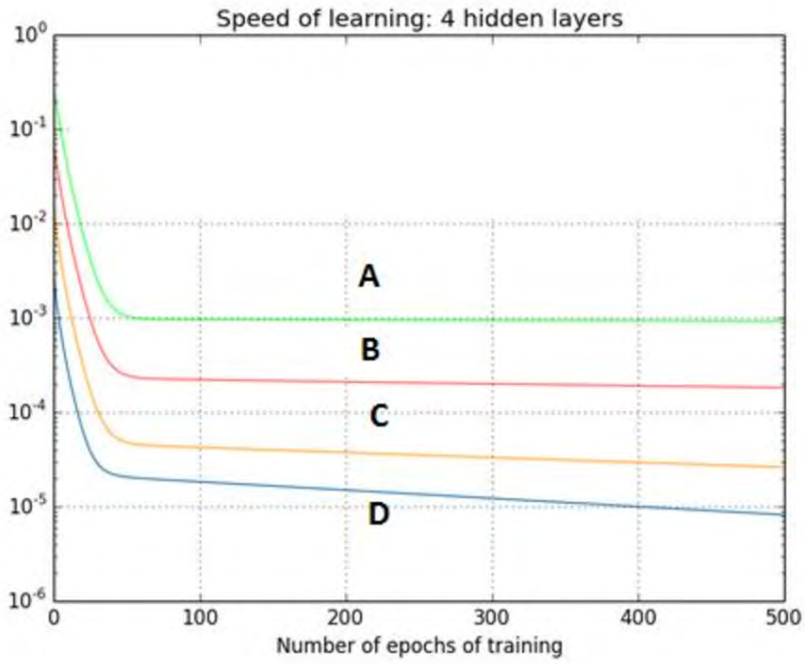
D.可能是A或B，取决于神经网络的权重设置
答案：（D）
不知道神经网络的权重和偏差是什么，则无法判定它将会给出什么样的输出。

假设我们已经在ImageNet数据集(物体识别)上训练好了一个卷积神经网络。然后给这张卷积神经网络输入一张全白的图片。对于这个输入的输出结果为任何种类的物体的可能性都是一样的，对吗？深度学习 DL模型 中
A 对的
B 不知道
C 看情况
D 不对
答案：D，已经训练好的卷积神经网络，各个神经元已经精雕细作完工，对于全白图片的输入，其j层层激活输出给最后的全连接层的值几乎不可能恒等，再经softmax转换之后也不会相等，所以"输出结果为任何种类的等可能性一样"也就是softmax的每项均相等，这个概率是极低的。

当在卷积神经网络中加入池化层(pooling layer)时，变换的不变性会被保留，是吗？深度学习 DL模型 中
A 不知道
B 看情况
C 是
D 否
答案：（C）
池化算法比如取最大值/取平均值等，都是输入数据旋转后结果不变，所以多层叠加后也有这种不变性。

当数据过大以至于无法在RAM中同时处理时，哪种梯度下降方法更加有效？（A）深度学习 DL基础 易
A 随机梯度下降法(Stochastic Gradient Descent)
B 不知道
C 整批梯度下降法(Full Batch Gradient Descent)
D 都不是
梯度下降法分随机梯度下降(每次用一个样本)、小批量梯度下降法(每次用一小批样本算出总损失，因而反向传播的梯度折中)、全批量梯度下降法则一次性使用全部样本。这三个方法，对于全体样本的损失函数曲面来说，梯度指向一个比一个准确．但是在工程应用中,受到内存/磁盘IO的吞吐性能制约，若要最小化梯度下降的实际运算时间,需要在梯度方向准确性和数据传输性能之间取得最好的平衡．所以，对于数据过大以至于无法在RAM中同时处理时，RAM每次只能装一个样本，那么只能选随机梯度下降法。

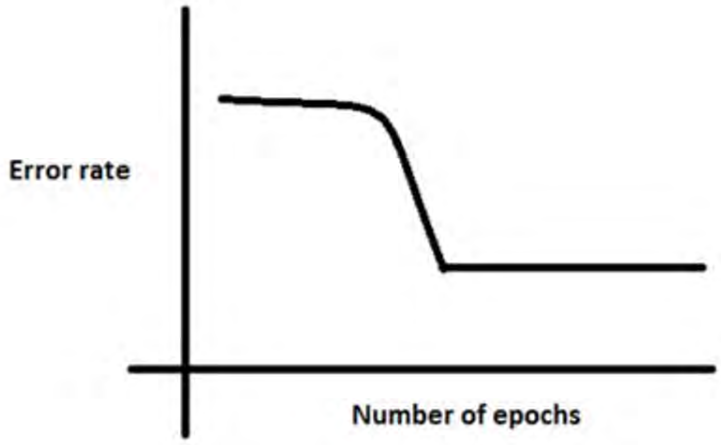
下图是一个利用sigmoid函数作为激活函数的含四个隐藏层的神经网络训练的梯度下降图。这个神经网络遇到了梯度消失的问题。下面哪个叙述是正确的？（A）深度学习 DL基础 中



第一隐藏层对应D，第二隐藏层对应C，第三隐藏层对应B，第四隐藏层对应A
第一隐藏层对应A，第二隐藏层对应C，第三隐藏层对应B，第四隐藏层对应D
第一隐藏层对应A，第二隐藏层对应B，第三隐藏层对应C，第四隐藏层对应D
第一隐藏层对应B，第二隐藏层对应D，第三隐藏层对应C，第四隐藏层对应A
由于反向传播算法进入起始层，学习能力降低，这就是梯度消失。换言之，梯度消失是梯度在前向传播中逐渐减为0, 按照图标题所说, 四条曲线是4个隐藏层的学习曲线, 那么第一层梯度最高(损失函数曲线下降明显), 最后一层梯度几乎为零(损失函数曲线变成平直线). 所以D是第一层, A是最后一层。

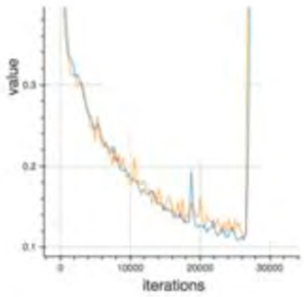
对于一个分类任务，如果开始时神经网络的权重不是随机赋值的，二是都设成0，下面哪个叙述是正确的？（C）深度学习 DL基础 易
A 其他选项都不对
B 没啥问题，神经网络会正常开始训练
C 神经网络可以训练，但是所有的神经元最后都会变成识别同样的东西
D 神经网络不会开始训练，因为没有梯度改变
令所有权重都初始化为0这个一个听起来还蛮合理的想法也许是一个我们假设中最好的一个假设了，但结果是错误的，因为如果神经网络计算出来的输出值都一个样，那么反向传播算法计算出来的梯度值一样，并且参数更新值也一样(w=w?α?dw)。更一般地说，如果权重初始化为同一个值，网络即是对称的，最终所有的神经元最后都会变成识别同样的东西。

下图显示，当开始训练时，误差一直很高，这是因为神经网络在往全局最小值前进之前一直被卡在局部最小值里。为了避免这种情况，我们可以采取下面哪种策略？深度学习 DL基础 易

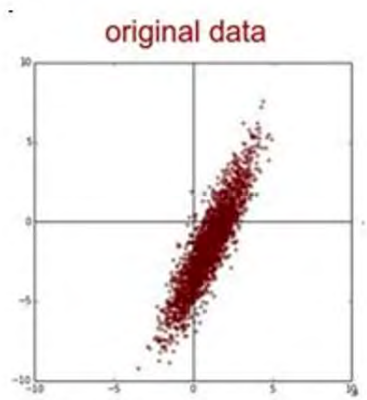


A 改变学习速率，比如一开始的几个训练周期不断更改学习速率
B 一开始将学习速率减小10倍，然后用动量项(momentum)
C 增加参数数目，这样神经网络就不会卡在局部最优处
D 其他都不对
答案：（A）
选项A可以将陷于局部最小值的神经网络提取出来。
对于一个图像识别问题(在一张照片里找出一只猫)，下面哪种神经网络可以更好地解决这个问题？（D）深度学习 DL基础 易
A 循环神经网络
B 感知机
C 多层感知机

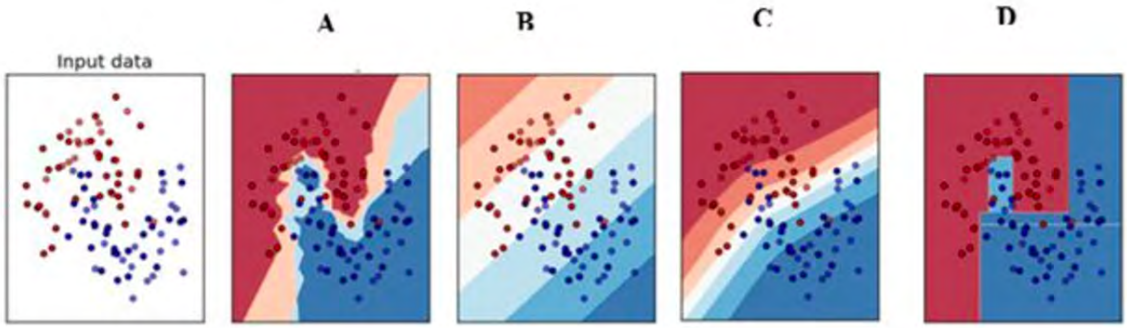
D 卷积神经网络
卷积神经网络将更好地适用于图像相关问题，因为考虑到图像附近位置变化的固有性质。
假设在训练中我们突然遇到了一个问题，在几次循环之后，误差瞬间降低



你认为数据有问题，于是你画出了数据并且发现也许是数据的偏度过大造成了这个问题。

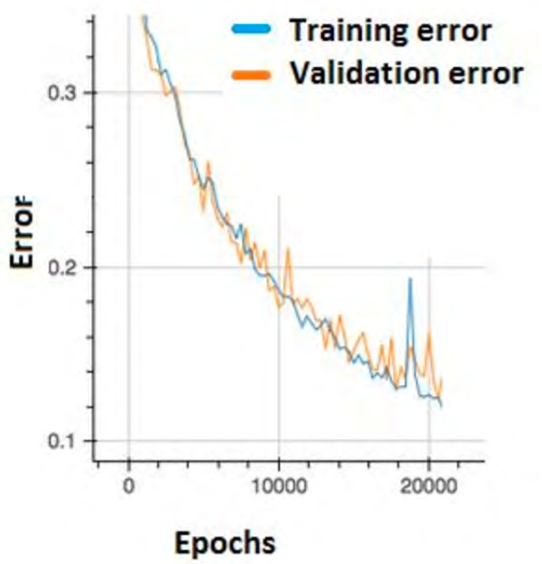


你打算怎么做来处理这个问题？深度学习 DL基础 易
A 对数据作归一化
B 对数据取对数变化
C 都不对
D 对数据作主成分分析(PCA)和归一化
答案：（D）
首先您将相关的数据去掉，然后将其置零。具体来说，误差瞬间降低, 一般原因是多个数据样本有强相关性且突然被拟合命中, 或者含有较大方差数据样本突然被拟合命中. 所以对数据作主成分分析(PCA)和归一化能够改善这个问题。
下面那个决策边界是神经网络生成的？ （E） 深度学习 DL基础 易



A A
B D
C C
D B
E 以上都有
神经网络可以逼近方式拟合任意函数, 所以以上图都可能由神经网络通过监督学习训练得到决策边界。

在下图中，我们可以观察到误差出现了许多小的"涨落"。 这种情况我们应该担心吗？深度学习 DL基础 易



A 需要，这也许意味着神经网络的学习速率存在问题
B 不需要，只要在训练集和交叉验证集上有累积的下降就可以了
C 不知道
D 不好说
答案：（B）
选项B是正确的，为了减少这些“起伏”，可以尝试增加批尺寸(batch size)。具体来说，在曲线整体趋势为下降时, 为了减少这些“起伏”，可以尝试增加批尺寸(batch size)以缩小batch综合梯度方向摆动范围. 当整体曲线趋势为平缓时出现可观的“起伏”, 可以尝试降低学习率以进一步收敛. “起伏”不可观时应该提前终止训练以免过拟合
在选择神经网络的深度时，下面那些参数需要考虑？深度学习 DL基础 易
1 神经网络的类型(如MLP,CNN)
2 输入数据
3 计算能力(硬件和软件能力决定)
4 学习速率
5 映射的输出函数

A 1,2,4,5
B 2,3,4,5
C 都需要考虑
D 1,3,4,5
答案：（C）
所有上述因素对于选择神经网络模型的深度都是重要的。特征抽取所需分层越多, 输入数据维度越高, 映射的输出函数非线性越复杂, 所需深度就越深. 另外为了达到最佳效果, 增加深度所带来的参数量增加, 也需要考虑硬件计算能力和学习速率以设计合理的训练时间。
考虑某个具体问题时，你可能只有少量数据来解决这个问题。不过幸运的是你有一个类似问题已经预先训练好的神经网络。可以用下面哪种方法来利用这个预先训练好的网络？（C）深度学习 DL基础 易
A 把除了最后一层外所有的层都冻住，重新训练最后一层
B 对新数据重新训练整个模型
C 只对最后几层进行调参(fine tune)
D 对每一层模型进行评估，选择其中的少数来用
如果有个预先训练好的神经网络, 就相当于网络各参数有个很靠谱的先验代替随机初始化. 若新的少量数据来自于先前训练数据(或者先前训练数据量很好地描述了数据分布, 而新数据采样自完全相同的分布), 则冻结前面所有层而重新训练最后一层即可; 但一般情况下, 新数据分布跟先前训练集分布有所偏差, 所以先验网络不足以完全拟合新数据时, 可以冻结大部分前层网络, 只对最后几层进行训练调参(这也称之为fine tune)。
增加卷积核的大小对于改进卷积神经网络的效果是必要的吗？（C）深度学习 DL基础 易
A 没听说过
B 是
C 否
D 不知道
答案：C，增加核函数的大小不一定会提高性能。这个问题在很大程度上取决于数据集。
请简述神经网络的发展史。深度学习 DL基础 易
@SIY.Z。本题解析来源：https://zhuanlan.zhihu.com/p/29435406
sigmoid会饱和，造成梯度消失。于是有了ReLU。
ReLU负半轴是死区，造成梯度变0。于是有了LeakyReLU，PReLU。
强调梯度和权值分布的稳定性，由此有了ELU，以及较新的SELU。
太深了，梯度传不下去，于是有了highway。
干脆连highway的参数都不要，直接变残差，于是有了ResNet。
强行稳定参数的均值和方差，于是有了BatchNorm。
在梯度流中增加噪声，于是有了 Dropout。
RNN梯度不稳定，于是加几个通路和门控，于是有了LSTM。
LSTM简化一下，有了GRU。
GAN的JS散度有问题，会导致梯度消失或无效，于是有了WGAN。
WGAN对梯度的clip有问题，于是有了WGAN-GP。

说说spark的性能调优。大数据 Hadoop/spark 中
https://tech.meituan.com/spark-tuning-basic.html
https://tech.meituan.com/spark-tuning-pro.html
常见的分类算法有哪些？ 机器学习 ML基础 易
SVM、神经网络、随机森林、逻辑回归、KNN、贝叶斯
常见的监督学习算法有哪些？ 机器学习 ML基础 易
感知机、svm、人工神经网络、决策树、逻辑回归
在其他条件不变的前提下，以下哪种做法容易引起机器学习中的过拟合问题（）机器学习 ML基础 易
A. 增加训练集量
B. 减少神经网络隐藏层节点数
C. 删除稀疏的特征
D. SVM算法中使用高斯核/RBF核代替线性核
正确答案：D
@刘炫320，本题题目及解析来源：http://blog.csdn.net/column/details/16442.html
一般情况下，越复杂的系统，过拟合的可能性就越高，一般模型相对简单的话泛化能力会更好一点。
B.一般认为，增加隐层数可以降低网络误差（也有文献认为不一定能有效降低），提高精度，但也使网络复杂化，从而增加了网络的训练时间和出现“过拟合”的倾向， svm高斯核函数比线性核函数模型更复杂，容易过拟合
D.径向基(RBF)核函数/高斯核函数的说明,这个核函数可以将原始空间映射到无穷维空间。对于参数 ，如果选的很大，高次特征上的权重实际上衰减得非常快，实际上（数值上近似一下）相当于一个低维的子空间；反过来，如果选得很小，则可以将任意的数据映射为线性可分——当然，这并不一定是好事，因为随之而来的可能是非常严重的过拟合问题。不过，总的来说，通过调整参数 ，高斯核实际上具有相当高的灵活性，也是 使用最广泛的核函数 之一。
下列时间序列模型中,哪一个模型可以较好地拟合波动性的分析和预测。机器学习 ML模型 易
A.AR模型
B.MA模型
C.ARMA模型
D.GARCH模型
正确答案：D
@刘炫320，本题题目及解析来源：http://blog.csdn.net/column/details/16442.html
AR模型是一种线性预测，即已知N个数据，可由模型推出第N点前面或后面的数据（设推出P点），所以其本质类似于插值。
MA模型(moving average model)滑动平均模型，其中使用趋势移动平均法建立直线趋势的预测模型。
ARMA模型(auto regressive moving average model)自回归滑动平均模型，模型参量法高分辨率谱分析方法之一。这种方法是研究平稳随机过程有理谱的典型方法。它比AR模型法与MA模型法有较精确的谱估计及较优良的谱分辨率性能，但其参数估算比较繁琐。
GARCH模型称为广义ARCH模型，是ARCH模型的拓展，由Bollerslev(1986)发展起来的。它是ARCH模型的推广。GARCH(p,0)模型，相当于ARCH(p)模型。GARCH模型是一个专门针对金融数据所量体订做的回归模型，除去和普通回归模型相同的之处，GARCH对误差的方差进行了进一步的建模。特别适用于波动性的分析和预测，这样的分析对投资者的决策能起到非常重要的指导性作用，其意义很多时候超过了对数值本身的分析和预测。
以下()属于线性分类器最佳准则？机器学习 ML模型 易
A.感知准则函数
B.贝叶斯分类
C.支持向量机
D.Fisher准则
正确答案：ACD
@刘炫320，本题题目及解析来源：http://blog.csdn.net/column/details/16442.html
线性分类器有三大类：感知器准则函数、SVM、Fisher准则，而贝叶斯分类器不是线性分类器。
感知准则函数 ： 准则函数以使错分类样本到分界面距离之和最小为原则。其优点是通过错分类样本提供的信息对分类器函数进行修正，这种准则是人工神经元网络多层感知器的基础。
支持向量机 ： 基本思想是在两类线性可分条件下，所设计的分类器界面使两类之间的间隔为最大，它的基本出发点是使期望泛化风险尽可能小。（使用核函数可解决非线性问题）
Fisher 准则 ： 更广泛的称呼是线性判别分析（LDA），将所有样本投影到一条远点出发的直线，使得同类样本距离尽可能小，不同类样本距离尽可能大，具体为最大化“广义瑞利商”。
根据两类样本一般类内密集，类间分离的特点，寻找线性分类器最佳的法线向量方向，使两类样本在该方向上的投影满足类内尽可能密集，类间尽可能分开。这种度量通过类内离散矩阵 Sw 和类间离散矩阵 Sb 实现。
基于二次准则函数的H-K算法较之于感知器算法的优点是()？深度学习 DL基础 易
A.计算量小
B.可以判别问题是否线性可分
C.其解完全适用于非线性可分的情况
D.其解的适应性更好
正确答案：BD
@刘炫320，本题题目及解析来源：http://blog.csdn.net/column/details/16442.html
HK算法思想很朴实,就是在最小均方误差准则下求得权矢量.
他相对于感知器算法的优点在于,他适用于线性可分和非线性可分得情况,对于线性可分的情况,给出最优权矢量,对于非线性可分得情况,能够判别出来,以退出迭代过程.
以下说法中正确的是() 机器学习 ML模型 中
A.SVM对噪声(如来自其他分布的噪声样本)鲁棒
B.在AdaBoost算法中,所有被分错的样本的权重更新比例相同
C.Boosting和Bagging都是组合多个分类器投票的方法,二都是根据单个分类器的正确率决定其权重
D.给定n个数据点,如果其中一半用于训练,一般用于测试,则训练误差和测试误差之间的差别会随着n的增加而减少
正确答案：BD
@刘炫320，本题题目及解析来源：http://blog.csdn.net/column/details/16442.html
A、SVM对噪声（如来自其他分布的噪声样本）鲁棒
SVM本身对噪声具有一定的鲁棒性，但实验证明，是当噪声率低于一定水平的噪声对SVM没有太大影响，但随着噪声率的不断增加，分类器的识别率会降低。
B、在AdaBoost算法中所有被分错的样本的权重更新比例相同
AdaBoost算法中不同的训练集是通过调整每个样本对应的权重来实现的。开始时，每个样本对应的权重是相同的，即其中n为样本个数，在此样本分

布下训练出一弱分类器。对于分类错误的样本，加大其对应的权重；而对于分类正确的样本，降低其权重，这样分错的样本就被凸显出来，从而得到一个新的样本分布。在新的样本分布下，再次对样本进行训练，得到弱分类器。以此类推，将所有的弱分类器重叠加起来，得到强分类器。
C、Boost和Bagging都是组合多个分类器投票的方法，二者均是根据单个分类器的正确率决定其权重。
Bagging与Boosting的区别：
取样方式不同。
Bagging采用均匀取样，而Boosting根据错误率取样。
Bagging的各个预测函数没有权重，而Boosting是有权重的。
Bagging的各个预测函数可以并行生成，而Boosing的各个预测函数只能顺序生成。
@AntZ
A. SVM解决的是结构风险最小, 经验风险处理较弱, 所以对数据噪声敏感.
B. AdaBoost算法中, 每个迭代训练一个学习器并按其误分类率得到该学习器的权重alpha, 这个学习器的权重算出两个更新比例去修正全部样本的权重: 正样本是exp(-alpha), 负样本是exp(alpha). 所以所有被分错的样本的权重更新比例相同.
C. bagging的学习器之间无权重不同, 简单取投票结果; Boosting的adaboost根据误分类率决定权重, boosting的gbdt则是固定小权重(也称学习率), 用逼近伪残差函数本身代替权重.
D: 根据中心极限定律, 随着n的增加, 训练误差和测试误差之间的差别必然减少 -- 这就是大数据训练的由来

> 输入图片大小为200×200，依次经过一层卷积（kernel size 5×5，padding 1，stride 2），pooling（kernel size 3×3，padding 0，stride 1），又一层卷积（kernel size 3×3，padding 1，stride 1）之后，输出特征图大小为：
>
> A. 95
>
> B. 96
>
> C. 97
>
> D. 98
>
> E. 99
>
> F. 100

深度学习 DL基础 易，正确答案：C
@刘炫320，本题题目及解析来源：http://blog.csdn.net/column/details/16442.html
首先我们应该知道卷积或者池化后大小的计算公式：

其中，padding指的是向外扩展的边缘大小，而stride则是步长，即每次移动的长度。

这样一来就容易多了，首先长宽一般大，所以我们只需要计算一个维度即可，这样，经过第一次卷积后的大小为:

经过第一次池化后的大小为：

经过第二次卷积后的大小为：

最终的结果为97。

在spss的基础分析模块中，作用是“以行列表的形式揭示数据之间的关系”的是（ ）大数据 Hadoop/spark 易
A. 数据描述
B. 相关
C. 交叉表
D. 多重相应
正确答案：C
一监狱人脸识别准入系统用来识别待进入人员的身份，此系统一共包括识别4种不同的人员：狱警，小偷，送餐员，其他。下面哪种学习方法最适合此种应用需求：（）机器学习 ML基础 易
A. 二分类问题
B. 多分类问题
C. 层次聚类问题
D. k-中心点聚类问题
E. 回归问题
F. 结构分析问题
正确答案：B
@刘炫320，本题题目及解析来源：http://blog.csdn.net/column/details/16442.html
二分类：每个分类器只能把样本分为两类。监狱里的样本分别为狱警、小偷、送餐员、其他。二分类肯 定行不通。瓦普尼克95年提出来基础的支持向量机就是个二分类的分类器，这个分类器学习过 程就是解一个基于正负二分类推导而来的一个最优规划问题（对偶问题），要解决多分类问题 就要用

决策树把二分类的分类器级联，VC维的概念就是说的这事的复杂度。
层次聚类： 创建一个层次等级以分解给定的数据集。监狱里的对象分别是狱警、小偷、送餐员、或者其 他，他们等级应该是平等的，所以不行。此方法分为自上而下（分解）和自下而上（合并）两种操作方式。
K-中心点聚类：挑选实际对象来代表簇，每个簇使用一个代表对象。它是围绕中心点划分的一种规则，所以这里并不合适。
回归分析：处理变量之间具有相关性的一种统计方法，这里的狱警、小偷、送餐员、其他之间并没有什 么直接关系。
结构分析： 结构分析法是在统计分组的基础上，计算各组成部分所占比重，进而分析某一总体现象的内部结构特征、总体的性质、总体内部结构依时间推移而表现出的变化规律性的统计方法。结构分析法的基本表现形式，就是计算结构指标。这里也行不通。
多分类问题： 针对不同的属性训练几个不同的弱分类器，然后将它们集成为一个强分类器。这里狱警、 小偷、送餐员 以及他某某，分别根据他们的特点设定依据，然后进行区分识别。
关于 logit 回归和 SVM 不正确的是（）机器学习 ML模型 易
A.Logit回归目标函数是最小化后验概率
B. Logit回归可以用于预测事件发生概率的大小
C. SVM目标是结构风险最小化
D.SVM可以有效避免模型过拟合
正确答案： A
@刘炫320，本题题目及解析来源：http://blog.csdn.net/column/details/16442.html
A. Logit回归本质上是一种根据样本对权值进行极大似然估计的方法，而后验概率正比于先验概率和似然函数的乘积。logit仅仅是最大化似然函数，并没有最大化后验概率，更谈不上最小化后验概率。而最小化后验概率是朴素贝叶斯算法要做的。A错误
B. Logit回归的输出就是样本属于正类别的几率，可以计算出概率，正确
C. SVM的目标是找到使得训练数据尽可能分开且分类间隔最大的超平面，应该属于结构风险最小化。
D. SVM可以通过正则化系数控制模型的复杂度，避免过拟合。
有两个样本点，第一个点为正样本,它的特征向量是(0,-1);第二个点为负样本,它的特征向量是(2,3),从这两个样本点组成的训练集构建一个线性SVM分类器的分类面方程是()机器学习 ML基础 易
A. 2x+y=4
B. x+2y=5
C. x+2y=3
D. 2x-y=0
正确答案：C
解析：这道题简化了，对于两个点来说，最大间隔就是垂直平分线，因此求出垂直平分线即可。斜率是两点连线的斜率的负倒数-1/((-1-3)/(0-2)) = -1/2, 可得y=-(1/2)x + c, 过中点((0+2)/2, (-1+3)/2) = (1, 1), 可得c=3/2, 故选C.
下面有关分类算法的准确率，召回率，F1 值的描述，错误的是？机器学习 ML基础 易
A.准确率是检索出相关文档数与检索出的文档总数的比率，衡量的是检索系统的查准率
B.召回率是指检索出的相关文档数和文档库中所有的相关文档数的比率，衡量的是检索系统的查全率
C.正确率、召回率和 F 值取值都在0和1之间，数值越接近0，查准率或查全率就越高
D.为了解决准确率和召回率冲突问题，引入了F1分数
正确答案：C
解析：
对于二类分类问题常用的评价指标是精准度（precision）与召回率（recall）。通常以关注的类为正类，其他类为负类，分类器在测试数据集上的预测或正确或不正确，4种情况出现的总数分别记作：
TP——将正类预测为正类数
FN——将正类预测为负类数
FP——将负类预测为正类数
TN——将负类预测为负类数
由此：
精准率定义为：P = TP / (TP + FP)
召回率定义为：R = TP / (TP + FN)
F1值定义为： F1 = 2 P R / (P + R)
精准率和召回率和F1取值都在0和1之间，精准率和召回率高，F1值也会高，不存在数值越接近0越高的说法，应该是数值越接近1越高。
以下几种模型方法属于判别式模型(Discriminative Model)的有() 机器学习 ML模型 易
1)混合高斯模型
2)条件随机场模型
3)区分度训练
4)隐马尔科夫模型
A.2,3
B.3,4
C.1,4
D.1,2
正确答案：A
@刘炫320，本题题目及解析来源：http://blog.csdn.net/column/details/16442.html
常见的判别式模型有：
Logistic regression（logistical 回归）
Linear discriminant analysis（线性判别分析）
Supportvector machines（支持向量机）
Boosting（集成学习）
Conditional random fields（条件随机场）
Linear regression（线性回归）
Neural networks（神经网络）
常见的生成式模型有:
Gaussian mixture model and othertypes of mixture model（高斯混合及其他类型混合模型）
Hidden Markov model（隐马尔可夫）
NaiveBayes（朴素贝叶斯）
AODE（平均单依赖估计）
Latent Dirichlet allocation（LDA主题模型）
Restricted Boltzmann Machine（限制波兹曼机）

生成式模型是根据概率乘出结果，而判别式模型是给出输入，计算出结果。
SPSS中，数据整理的功能主要集中在（ ）等菜单中。大数据 Hadoop/spark 易
A.数据
B.直销
C.分析
D.转换
正确答案：AD
@刘炫320，本题题目及解析来源：http://blog.csdn.net/column/details/16442.html
解析：对数据的整理主要在数据和转换功能菜单中。

> 深度学习是当前很热门的机器学习算法，在深度学习中，涉及到大量的矩阵相乘，现在需要计算三个稠密矩阵A,B,C的乘积ABC,假设三个矩阵的尺寸分别为，以下计算顺序效率最高的是（）
> A.(AB)C
>
> B.AC(B)
>
> C.A(BC)
>
> D.所以效率都相同

深度学习 DL基础 易，正确答案：A

@刘炫320，本题题目及解析来源：http://blog.csdn.net/column/details/16442.html
首先，根据简单的矩阵知识，因为 A*B ， A 的列数必须和 B 的行数相等。因此，可以排除 B 选项，
然后，再看 A 、 C 选项。在 A 选项中， 的矩阵 A 和的矩阵 B 的乘积，得到 的矩阵 A*B ，而 的每个元素需要 n 次乘法和 n-1 次加法，忽略加法，共需要 次乘法运算。同样情况分析 A*B 之后再乘以 C 时的情况，共需要 次乘法运算。因此， A 选项 (AB)C 需要的乘法次数是 。同理分析， C 选项 A(BC) 需要的乘法次数是 。
由于，显然 A 运算次数更少，故选 A 。

> Nave Bayes是一种特殊的Bayes分类器,特征变量是X,类别标签是C,它的一个假定是:()
> A.各类别的先验概率P(C)是相等的
>
> B.以0为均值，sqr(2)/2为标准差的正态分布
>
> C.特征变量X的各个维度是类别条件独立随机变量
>
> D.P(X|C)是高斯分布

机器学习 ML模型 中，正确答案：C

@刘炫320，本题题目及解析来源：http://blog.csdn.net/column/details/16442.html

朴素贝叶斯的条件就是每个变量相互独立。

> 关于支持向量机SVM,下列说法错误的是（）
> A.L2正则项，作用是最大化分类间隔，使得分类器拥有更强的泛化能力
>
> B.Hinge 损失函数，作用是最小化经验分类错误
>
> C.分类间隔为1/||w||，||w||代表向量的模
>
> D.当参数C越小时，分类间隔越大，分类错误越多，趋于欠学习

机器学习 ML模型，易，正确答案：C

@刘炫320，本题题目及解析来源：http://blog.csdn.net/column/details/16442.html

A正确。考虑加入正则化项的原因：想象一个完美的数据集，y>1是正类，y<-1是负类，决策面y=0，加入一个y=-30的正类噪声样本，那么决策面将会变“歪”很多，分类间隔变小，泛化能力减小。加入正则项之后，对噪声样本的容错能力增强，前面提到的例子里面，决策面就会没那么“歪”了，

使得分类间隔变大，提高了泛化能力。

B正确。

C错误。间隔应该是2/||w||才对，后半句应该没错，向量的模通常指的就是其二范数。

D正确。考虑软间隔的时候，C对优化问题的影响就在于把a的范围从[0，+inf]限制到了[0,C]。C越小，那么a就会越小，目标函数拉格朗日函数导数为0可以求出w=求和，a变小使得w变小，因此间隔2/||w||变大

在HMM中,如果已知观察序列和产生观察序列的状态序列,那么可用以下哪种方法直接进行参数估计() 机器学习 ML模型 易
A.EM算法
B.维特比算法
C.前向后向算法
D.极大似然估计
正确答案：D
@刘炫320，本题题目及解析来源：http://blog.csdn.net/column/details/16442.html
EM算法： 只有观测序列，无状态序列时来学习模型参数，即Baum-Welch算法
维特比算法： 用动态规划解决HMM的预测问题，不是参数估计
前向后向算法：用来算概率
极大似然估计：即观测序列和相应的状态序列都存在时的监督学习算法，用来估计参数
注意的是在给定观测序列和对应的状态序列估计模型参数，可以利用极大似然发估计。如果给定观测序列，没有对应的状态序列，才用EM，将状态序列看不不可测的隐数据。假定某同学使用Naive Bayesian（NB）分类模型时，不小心将训练数据的两个维度搞重复了，那么关于NB的说法中正确的是（）：机器学习 ML模型 易
A.这个被重复的特征在模型中的决定作用会被加强
B.模型效果相比无重复特征的情况下精确度会降低
C.如果所有特征都被重复一遍，得到的模型预测结果相对于不重复的情况下的模型预测结果一样。
D.当两列特征高度相关时，无法用两列特征相同时所得到的结论来分析问题
E.NB可以用来做最小二乘回归
F.以上说法都不正确
正确答案：BD
朴素贝叶斯的条件就是每个变量相互独立. 若高度相关的特征在模型中引入两次, 这样增加了这一特征的重要性, 则它的性能因数据包含高度相关的特征而下降。正确做法是评估特征的相关矩阵，并移除那些高度相关的特征。

L1与L2范数。机器学习 ML基础 易

> 在Logistic Regression 中,如果同时加入L1和L2范数,会产生什么效果()
>
> A.可以做特征选择,并在一定程度上防止过拟合
>
> B.能解决维度灾难问题
>
> C.能加快计算速度
>
> D.可以获得更准确的结果

正确答案:ABC

@刘炫320，本题题目及解析来源：http://blog.csdn.net/column/details/16442.html
Ｌ１范数具有系数解的特性，但是要注意的是， Ｌ１没有选到的特征不代表不重要，原因是两个高相关性的特征可能只保留一个。如果需要确定哪个特征重要，再通过交叉验证。它的优良性质是能产生稀疏性，导致 W 中许多项变成零。 稀疏的解除了计算量上的好处之外，更重要的是更具有“可解释性”。所以能加快计算速度和缓解维数灾难. 所以BC正确.

在代价函数后面加上正则项， Ｌ１即是Ｌｏｓｓｏ回归， Ｌ２是岭回归。L1范数是指向量中各个元素绝对值之和，用于特征选择。L2范数 是指向量各元素的平方和然后求平方根，用于 防止过拟合，提升模型的泛化能力。因此选择A。

对于机器学习中的范数规则化，也就是L0,L1,L2范数的详细解答，请参阅《范数规则化》。

正则化。机器学习 ML基础 易
机器学习中L1正则化和L2正则化的区别是?
A.使用L1可以得到稀疏的权值
B.使用L1可以得到平滑的权值
C.使用L2可以得到稀疏的权值
D.使用L2可以得到平滑的权值
正确答案:AD
@刘炫320，本题题目及解析来源：http://blog.csdn.net/column/details/16442.html

L1正则化偏向于稀疏，它会自动进行特征选择，去掉一些没用的特征，也就是将这些特征对应的权重置为0.
L2主要功能是为了防止过拟合，当要求参数越小时，说明模型越简单，而模型越简单则，越趋向于平滑，从而防止过拟合。
L1正则化/Lasso
L1正则化将系数w的l1范数作为惩罚项加到损失函数上，由于正则项非零，这就迫使那些弱的特征所对应的系数变成0。因此L1正则化往往会使学到的模型很稀疏（系数w经常为0），这个特性使得L1正则化成为一种很好的特征选择方法。
L2正则化/Ridge regression
L2正则化将系数向量的L2范数添加到了损失函数中。由于L2惩罚项中系数是二次方的，这使得L2和L1有着诸多差异，最明显的一点就是，L2正则化会让系数的取值变得平均。对于关联特征，这意味着他们能够获得更相近的对应系数。还是以Y=X1+X2为例，假设X1和X2具有很强的关联，如果用L1正则化，不论学到的模型是Y=X1+X2还是Y=2X1，惩罚都是一样的，都是2alpha。但是对于L2来说，第一个模型的惩罚项是2alpha，但第二个模型的是4*alpha。可以看出，系数之和为常数时，各系数相等时惩罚是最小的，所以才有了L2会让各个系数趋于相同的特点。
可以看出，L2正则化对于特征选择来说一种稳定的模型，不像L1正则化那样，系数会因为细微的数据变化而波动。所以L2正则化和L1正则化提供的价值是不同的，L2正则化对于特征理解来说更加有用：表示能力强的特征对应的系数是非零。
因此，一句话总结就是：L1会趋向于产生少量的特征，而其他的特征都是0，而L2会选择更多的特征，这些特征都会接近于0。Lasso在特征选择时候非常有用，而Ridge就只是一种规则化而已。
具体的，可以参阅《机器学习之特征选择》与《机器学习范数正则化》。势函数法。机器学习 ML基础 易
位势函数法的积累势函数K(x)的作用相当于Bayes判决中的()
A.后验概率
B.先验概率
C.类概率密度
D.类概率密度与先验概率的乘积
正确答案:AD
@刘炫320，本题题目及解析来源：http://blog.csdn.net/column/details/16442.html
事实上，AD说的是一回事。
具体的，势函数详解请看——《势函数法》。隐马尔可夫。机器学习 ML模型 易
隐马尔可夫模型三个基本问题以及相应的算法说法正确的是（ ）
A.评估—前向后向算法
B.解码—维特比算法
C.学习—Baum-Welch算法
D.学习—前向后向算法
正确答案:ABC
解析：评估问题，可以使用前向算法、后向算法、前向后向算法。
特征比数据量还大时，选择什么样的分类器？机器学习 ML基础 易
线性分类器，因为维度高的时候，数据一般在维度空间里面会比较稀疏，很有可能线性可分
来自http://blog.sina.com.cn/s/blog_178bcad000102x70r.html
下列属于无监督学习的是： 机器学习 ML基础 易
A.k-means
B.SVM
C.最大熵
D.CRF
正确答案：A
解析：
A是聚类，BC是分类，D是序列化标注，也是有监督学习。下列哪个不属于CRF模型对于HMM和MEMM模型的优势（ ） 机器学习 ML模型 中
A.特征灵活
B.速度快
C.可容纳较多上下文信息
D.全局最优
正确答案：B

解析：
CRF 的优点：特征灵活，可以容纳较多的上下文信息，能够做到全局最优CRF 的缺点：速度慢
CRF没有HMM那样严格的独立性假设条件，因而可以容纳任意的上下文信息。特征设计灵活（与ME一样） ————————与HMM比较
同时，由于CRF计算全局最优输出节点的条件概率，它还克服了最大熵马尔可夫模型标记偏置（Label-bias）的缺点。 --————————与MEMM比较

CRF是在给定需要标记的观察序列的条件下，使用维特比算法，计算整个标记序列的联合概率分布，而不是在给定当前状态条件下，定义下一个状态的状态分布。————————与ME比较
数据清理中，处理缺失值的方法是? 机器学习 ML基础 易
A.估算
B.整例删除
C.变量删除
D.成对删除
正确答案：ABCD

@刘炫320，本题题目及解析来源：http://blog.csdn.net/column/details/16442.html
由于调查、编码和录入误差，数据中可能存在一些无效值和缺失值，需要给予适当的处理。常用的处理方法有：估算，整例删除，变量删除和成对删除。

估算(estimation)。最简单的办法就是用某个变量的样本均值、中位数或众数代替无效值和缺失值。这种办法简单，但没有充分考虑数据中已有的信息，误差可能较大。另一种办法就是根据调查对象对其他问题的答案，通过变量之间的相关分析或逻辑推论进行估计。例如，某一产品的拥有情况可能与家庭收入有关，可以根据调查对象的家庭收入推算拥有这一产品的可能性。
整例删除(casewise deletion)是剔除含有缺失值的样本。由于很多问卷都可能存在缺失值，这种做法的结果可能导致有效样本量大大减少，无法充分利用已经收集到的数据。因此，只适合关键变量缺失，或者含有无效值或缺失值的样本比重很小的情况。
变量删除(variable deletion)。如果某一变量的无效值和缺失值很多，而且该变量对于所研究的问题不是特别重要，则可以考虑将该变量删除。这种做法减少了供分析用的变量数目，但没有改变样本量。
成对删除(pairwise deletion)是用一个特殊码(通常是9、99、999等)代表无效值和缺失值，同时保留数据集中的全部变量和样本。但是，在具体计算时

只采用有完整答案的样本，因而不同的分析因涉及的变量不同，其有效样本量也会有所不同。这是一种保守的处理方法，最大限度地保留了数据集中的可用信息。

采用不同的处理方法可能对分析结果产生影响，尤其是当缺失值的出现并非随机且变量之间明显相关时。因此，在调查中应当尽量避免出现无效值和缺失值，保证数据的完整性。
关于线性回归的描述,以下正确的有（） 机器学习 ML基础 易
A.基本假设包括随机干扰项是均值为0,方差为1的标准正态分布
B.基本假设包括随机干扰下是均值为0的同方差正态分布
C.在违背基本假设时,普通最小二乘法估计量不再是最佳线性无偏估计量
D.在违背基本假设时,模型不再可以估计
E.可以用DW检验残差是否存在序列相关性
F.多重共线性会使得参数估计值方差减小
正确答案：ACEF
@刘炫320，本题题目及解析来源：http://blog.csdn.net/column/details/16442.html
AB一元线性回归的基本假设有
1、随机误差项是一个期望值或平均值为0的随机变量；
2、对于解释变量的所有观测值，随机误差项有相同的方差；
3、随机误差项彼此不相关；
4、解释变量是确定性变量，不是随机变量，与随机误差项彼此之间相互独立；
5、解释变量之间不存在精确的（完全的）线性关系，即解释变量的样本观测值矩阵是满秩矩阵；
6、随机误差项服从正态分布

CD 违背基本假设的计量经济学模型还是可以估计的，只是不能使用普通最小二乘法进行估计。
当存在异方差时，普通最小二乘法估计存在以下问题： 参数估计值虽然是无偏的，但不是最小方差线性无偏估计。
E杜宾-瓦特森（DW）检验，计量经济，统计分析中常用的一种检验序列一阶 自相关 最常用的方法。

F所谓多重共线性（Multicollinearity）是指线性回归模型中的解释变量之间由于存在精确相关关系或高度相关关系而使模型估计失真或难以估计准确。影响
（1）完全共线性下参数估计量不存在
（2）近似共线性下OLS估计量非有效
多重共线性使参数估计值的方差增大，1/(1-r2)为方差膨胀因子(Variance Inflation Factor, VIF)
（3）参数估计量经济含义不合理
（4）变量的显著性检验失去意义，可能将重要的解释变量排除在模型之外
（5）模型的预测功能失效。变大的方差容易使区间预测的“区间”变大，使预测失去意义。
对于线性回归模型,当响应变量服从正态分布,误差项满足高斯–马尔科夫条件（零均值、等方差、不相关）时,回归参数的最小二乘估计是一致最小方差无偏估计。
当然，该条件只是理想化的假定，为的是数学上有相应的较为成熟的结论。其实大多数实际问题都不完全满足这些理想化的假定。

线性回归模型理论的发展正是在不断克服理想化条件不被满足时得到许多新方法。如加权LSE、岭估计、压缩估计、BOX_COX变换等一系列段。做实际工作时一定是要超越书本上的理想化条件的。
影响聚类算法效果的主要原因有：（ ） 机器学习 ML基础 易
A.特征选取
B.模式相似性测度
C.分类准则
D.已知类别的样本质量
正确答案：ABC
@刘炫320，本题题目及解析来源：http://blog.csdn.net/column/details/16442.html
解析：这道题应该是很简单的，D之所以不正确，是因为聚类是对无类别的数据进行聚类，不使用已经标记好的数据。
前面的ABC选项，可以参考：《聚类分析》与《各类算法的比较》。
以下哪个是常见的时间序列算法模型（）机器学习 ML模型 易
A.RSI
B.MACD
C.ARMA
D.KDJ
正确答案：C
解析：
自回归滑动平均模型(ARMA)
其建模思想可概括为：逐渐增加模型的阶数，拟合较高阶模型，直到再增加模型的阶数而剩余残差方差不再显著减小为止。

其他三项都不是一个层次的。
A.相对强弱指数 (RSI, Relative Strength Index) 是通过比较一段时期内的平均收盘涨数和平均收盘跌数来分析市场买沽盘的意向和实力，从而作出未来市场的走势 .
B.移动平均聚散指标 (MACD, Moving Average Convergence Divergence), 是根据均线的构造原理，对股票价格的收盘价进行平滑处理，求出算术平均值以后再进行计算，是一种趋向类指标 .
D. 随机指标 (KDJ) 一般是根据统计学的原理，通过一个特定的周期（常为 9 日 ,9 周等）内出现过的最高价，最低价及最后一个计算周期的收盘价及这三者之间的比例关系，来计算最后一个计算周期的未成熟随机值 RSV, 然后根据平滑移动平均线的方法来计算 K 值，D 值与 J 值，并绘成曲线图来研判股票走势 .
下列不是SVM核函数的是（） 机器学习 ML模型 易
A.多项式核函数
B.logistic核函数
C.径向基核函数
D.Sigmoid核函数
正确答案：B

@刘炫320，本题题目及解析来源：http://blog.csdn.net/column/details/16442.html
SVM核函数包括线性核函数、多项式核函数、径向基核函数、高斯核函数、幂指数核函数、拉普拉斯核函数、ANOVA核函数、二次有理核函数、多元二次核函数、逆多元二次核函数以及Sigmoid核函数.
核函数的定义并不困难，根据泛函的有关理论，只要一种函数 K (x i , x j) 满足Mercer条件，它就对应某一变换空间的内积．对于判断哪些函数是核函数到目前为止也取得了重要的突破，得到Mercer定理和以下常用的核函数类型：
(1)线性核函数
K (x , x i) = x ? x i
(2)多项式核
K (x , x i) = ((x ? x i) + 1) d
(3)径向基核（RBF）
K (x , x i) = exp (? ‖ x ? x i ‖ 2 σ 2)
Gauss径向基函数则是局部性强的核函数，其外推能力随着参数 σ 的增大而减弱。多项式形式的核函数具有良好的全局性质。局部性较差。
(4)傅里叶核
K (x , x i) = 1 ? q 2 2 (1 ? 2 q cos (x ? x i) + q 2)
(5)样条核
K (x , x i) = B 2 n + 1 (x ? x i)
(6)Sigmoid核函数
K (x , x i) = tanh (κ (x , x i) ? δ)
采用Sigmoid函数作为核函数时，支持向量机实现的就是一种多层感知器神经网络，应用SVM方法，隐含层节点数目(它确定神经网络的结构)、隐含层节点对输入节点的权值都是在设计(训练)的过程中自动确定的。而且支持向量机的理论基础决定了它最终求得的是全局最优值而不是局部最小值，也保证了它对于未知样本的良好泛化能力而不会出现过学习现象。

核函数的选择
在选取核函数解决实际问题时，通常采用的方法有：
一是利用专家的先验知识预先选定核函数；
二是采用Cross-Validation方法，即在进行核函数选取时，分别试用不同的核函数，归纳误差最小的核函数就是最好的核函数．如针对傅立叶核、RBF核，结合信号处理问题中的函数回归问题，通过仿真实验，对比分析了在相同数据条件下，采用傅立叶核的SVM要比采用RBF核的SVM误差小很多．
三是采用由Smits等人提出的混合核函数方法，该方法较之前两者是目前选取核函数的主流方法，也是关于如何构造核函数的又一开创性的工作．将不同的核函数结合起来后会有更好的特性，这是混合核函数方法的基本思想．
已知一组数据的协方差矩阵P,下面关于主分量说法错误的是（） 数据挖掘 DM基础 易
A.主分量分析的最佳准则是对一组数据进行按一组正交基分解, 在只取相同数量分量的条件下,以均方误差计算截尾误差最小
B.在经主分量分解后,协方差矩阵成为对角矩阵
C.主分量分析就是K-L变换
D.主分量是通过求协方差矩阵的特征值得到
正确答案：C
解析：K-L变换与PCA变换是不同的概念，PCA的变换矩阵是协方差矩阵，K-L变换的变换矩阵可以有很多种（二阶矩阵、协方差矩阵、总类内离散度矩阵等等）。当K-L变换矩阵为协方差矩阵时，等同于PCA。
在分类问题中,我们经常会遇到正负样本数据量不等的情况,比如正样本为10w条数据,负样本只有1w条数据,以下最合适的处理方法是（）机器学习 ML基础 易
A.将负样本重复10次,生成10w样本量,打乱顺序参与分类
B.直接进行分类,可以最大限度利用数据
C.从10w正样本中随机抽取1w参与分类
D.将负样本每个权重设置为10,正样本权重为1,参与训练过程
正确答案:ACD
解析：对于这一块我想还是有一些了解的
1. 重采样。
A可视作重采样的变形。改变数据分布消除不平衡，可能导致过拟合。
2. 欠采样。
C的方案 提高少数类的分类性能，可能丢失多数类的重要信息。
如果1：10算是均匀的话，可以将多数类分割成为1000份。然后将每一份跟少数类的样本组合进行训练得到分类器。而后将这1000个分类器用assemble的方法组合位一个分类器。A选项可以看作此方式，因而相对比较合理。
另：如果目标是 预测的分布 跟训练的分布一致，那就加大对分布不一致的惩罚系数。
3. 权值调整。
D方案也是其中一种方式。

当然，这只是在数据集上进行相应的处理，在算法上也有相应的处理方法。

> 在统计模式识分类问题中，当先验概率未知时，可以使用（）机器学习 ML基础 易
>
> A.最小损失准则
>
> B.N-P判决
>
> C.最小最大损失准则
>
> D.最小误判概率准则

正确答案:BC

@刘炫320，本题题目及解析来源：http://blog.csdn.net/column/details/16442.html

选项 A

最小损失准则中需要用到先验概率

选项B

在贝叶斯决策中，对于先验概率p(y)，分为已知和未知两种情况。

1. p(y)已知，直接使用贝叶斯公式求后验概率即可；

2. p(y)未知，可以使用聂曼-皮尔逊决策(N-P决策)来计算决策面。

聂曼-皮尔逊决策（N-P判决）可以归结为找阈值a，即：

如果，则 x属于w1；

如果，则 x属于w 2；

选项C

而最大最小损失规则主要就是使用解决最小损失规则时先验概率未知或难以计算的问题的。

解决隐马模型中预测问题的算法是（）机器学习 ML模型 中

A.前向算法

B.后向算法

C.Baum-Welch算法

D.维特比算法

正确答案：D

@刘炫320，本题题目及解析来源：http://blog.csdn.net/column/details/16442.html

A、B：前向、后向算法解决的是一个评估问题，即给定一个模型，求某特定观测序列的概率，用于评估该序列最匹配的模型。

C：Baum-Welch算法解决的是一个模型训练问题，即参数估计，是一种无监督的训练方法，主要通过EM迭代实现；

D：维特比算法解决的是给定 一个模型和某个特定的输出序列，求最可能产生这个输出的状态序列。如通过海藻变化（输出序列）来观测天气（状态序列），是预测问题，通信中的解码问题。一般，k-NN最近邻方法在（）的情况下效果较好 机器学习 ML模型 易

A.样本较多但典型性不好

B.样本较少但典型性好

C.样本呈团状分布

D.样本呈链状分布

正确答案：B

解析：K近邻算法主要依靠的是周围的点，因此如果样本过多，那肯定是区分不出来的。因此应当选择B

样本呈团状颇有迷惑性，这里应该指的是整个样本都是呈团状分布，这样kNN就发挥不出其求近邻的优势了，整体样本应该具有典型性好，样本较少，比较适宜。

下列方法中，可以用于特征降维的方法包括（） 深度学习 DL模型 易

A.主成分分析PCA

B.线性判别分析LDA

C.深度学习SparseAutoEncoder

D.矩阵奇异值分解SVD

E.最小二乘法LeastSquares

正确答案：ABCD

解析：降维的3种常见方法ABD，都是线性的。深度学习是降维的方法这个就比较新鲜了，事实上，细细想来，也是降维的一种方法，因为如果隐藏层中的神经元数目要小于输入层，那就达到了降维，但如果隐藏层中的神经元如果多余输入层，那就不是降维了。

最小二乘法是线性回归的一种解决方法，其实也是投影，但是并没有进行降维。下面哪些是基于核的机器学习算法？（）机器学习 ML模型 易

A.Expectation Maximization（EM）（最大期望算法）

B.Radial Basis Function（RBF）（径向基核函数）

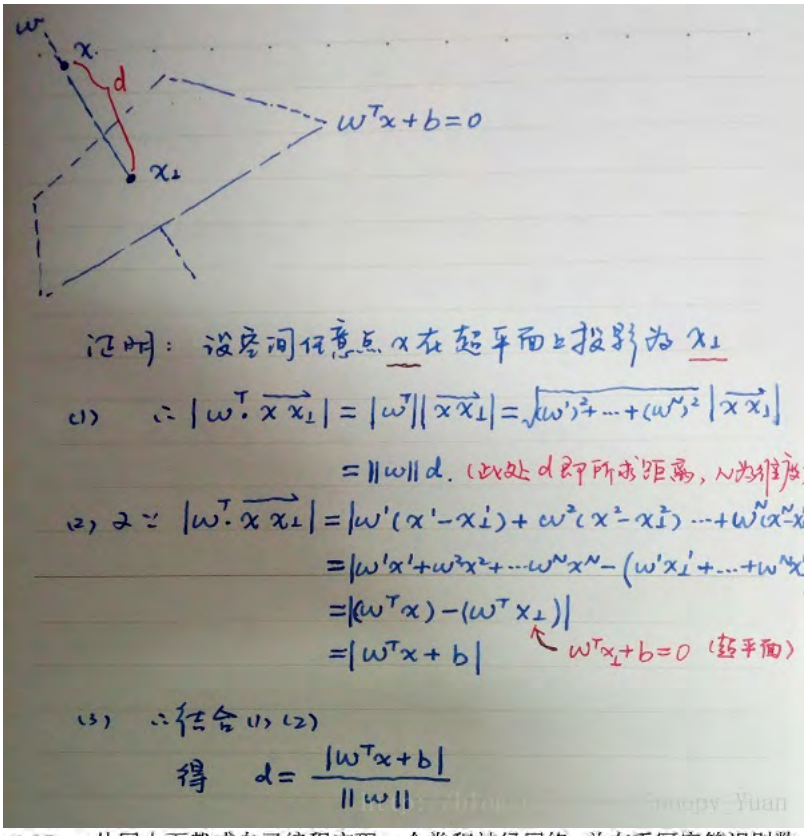
C.Linear Discrimimate Analysis（LDA）（主成分分析法）

D.Support Vector Machine（SVM）（支持向量机）

正确答案：BCD

解析：径向基核函数是非常常用的核函数，而主成分分析法的常规方法是线性的，但是当遇到非线性的时候，同样可以使用核方法使得非线性问题转化为线性问题。支持向量机处理非线性的问题的时候，核函数也是非常重要的。

6.1 试证明样本空间中任意点 $\boldsymbol{x}$ 到超平面 $(\boldsymbol{w}, b)$ 的距离为式(6.2). 机器学习 ML基础 易



5.10 从网上下载或自己编程实现一个卷积神经网络，并在手写字符识别数据 MNIST 上进行实验测试.

深度学习 DL模型 中

解析详见：http://blog.csdn.net/snoopy_yuan/article/details/71703019

神经网络中激活函数的真正意义？一个激活函数需要具有哪些必要的属性？还有哪些属性是好的属性但不必要的？深度学习 DL基础 中

@Hengkai Guo，本题解析来源：https://www.zhihu.com/question/67366051

说说我对一个好的激活函数的理解吧，有些地方可能不太严谨，欢迎讨论。（部分参考了Activation function。）

1. 非线性：即导数不是常数。这个条件前面很多答主都提到了，是多层神经网络的基础，保证多层网络不退化成单层线性网络。这也是激活函数的意义所在。

2. 几乎处处可微：可微性保证了在优化中梯度的可计算性。传统的激活函数如sigmoid等满足处处可微。对于分段线性函数比如ReLU，只满足几乎处处可微（即仅在有限个点处不可微）。对于SGD算法来说，由于几乎不可能收敛到梯度接近零的位置，有限的不可微点对于优化结果不会有很大影响[1]。

3. 计算简单：正如题主所说，非线性函数有很多。极端的说，一个多层神经网络也可以作为一个非线性函数，类似于Network In Network[2]中把它当做卷积操作的做法。但激活函数在神经网络前向的计算次数与神经元的个数成正比，因此简单的非线性函数自然更适合用作激活函数。这也是ReLU之流比其它使用Exp等操作的激活函数更受欢迎的其中一个原因。

4. 非饱和性（saturation）：饱和指的是在某些区间梯度接近于零（即梯度消失），使得参数无法继续更新的问题。最经典的例子是Sigmoid，它的导数在x为比较大的正值和比较小的负值时都会接近于0。更极端的例子是阶跃函数，由于它在几乎所有位置的梯度都为0，因此处处饱和，无法作为激活函数。ReLU在x>0时导数恒为1，因此对于再大的正值也不会饱和。但同时对于x<0，其梯度恒为0，这时候它也会出现饱和的现象（在这种情况下通常称为dying ReLU）。Leaky ReLU[3]和PReLU[4]的提出正是为了解决这一问题。

5. 单调性（monotonic）：即导数符号不变。这个性质大部分激活函数都有，除了诸如sin、cos等。个人理解，单调性使得在激活函数处的梯度方向不会经常改变，从而让训练更容易收敛。

6. 输出范围有限：有限的输出范围使得网络对于一些比较大的输入也会比较稳定，这也是为什么早期的激活函数都以此类函数为主，如Sigmoid、TanH。但这导致了前面提到的梯度消失问题，而且强行让每一层的输出限制到固定范围会限制其表达能力。因此现在这类函数仅用于某些需要特定输出范围的场合，比如概率输出（此时loss函数中的log操作能够抵消其梯度消失的影响[1]）、LSTM里的gate函数。

7. 接近恒等变换（identity）：即约等于x。这样的好处是使得输出的幅值不会随着深度的增加而发生显著的增加，从而使网络更为稳定，同时梯度也能够更容易地回传。这个与非线性是有点矛盾的，因此激活函数基本只是部分满足这个条件，比如TanH只在原点附近有线性区（在原点为0且在原点的导数为1），而ReLU只在x>0时为线性。这个性质也让初始化参数范围的推导更为简单[5][4]。额外提一句，这种恒等变换的性质也被其他一些网络结构设计所借鉴，比如CNN中的ResNet[6]和RNN中的LSTM。

8. 参数少：大部分激活函数都是没有参数的。像PReLU带单个参数会略微增加网络的大小。还有一个例外是Maxout[7]，尽管本身没有参数，但在同样输出通道数下k路Maxout需要的输入通道数是其它函数的k倍，这意味着神经元数目也需要变为k倍；但如果不考虑维持输出通道数的情况下，该激活函数又能将参数个数减少为原来的k倍。

9. 归一化（normalization）：这个是最近才出来的概念，对应的激活函数是SELU[8]，主要思想是使样本分布自动归一化到零均值、单位方差的分布，从而稳定训练。在这之前，这种归一化的思想也被用于网络结构的设计，比如Batch Normalization[9]。

参考文献：

[1] Goodfellow I, Bengio Y, Courville A. Deep learning[M]. MIT press, 2016.
[2] Lin M, Chen Q, Yan S. Network in network[J]. arXiv preprint arXiv:1312.4400, 2013.
[3] Maas A L, Hannun A Y, Ng A Y. Rectifier nonlinearities improve neural network acoustic models[C]//Proc. ICML. 2013, 30(1).
[4] He K, Zhang X, Ren S, et al. Delving
deep into rectifiers: Surpassing human-level performance on imagenet
classification[C]//Proceedings of the IEEE international conference on
computer vision. 2015: 1026-1034.
[5] Glorot X, Bengio Y. Understanding the
difficulty of training deep feedforward neural networks[C]//Proceedings
of the Thirteenth International Conference on Artificial Intelligence

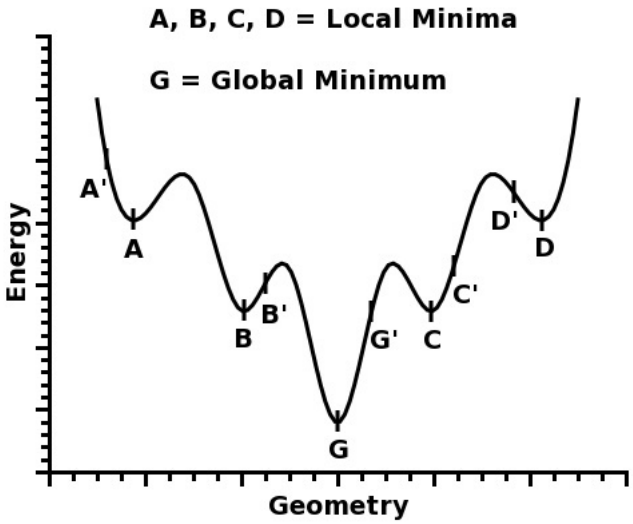
and Statistics. 2010: 249-256.
[6] He K, Zhang X, Ren S, et al. Deep
residual learning for image recognition[C]//Proceedings of the IEEE
conference on computer vision and pattern recognition. 2016: 770-778.
[7] Goodfellow I J, Warde-Farley D, Mirza M, et al. Maxout networks[J]. arXiv preprint arXiv:1302.4389, 2013.
[8] Klambauer G, Unterthiner T, Mayr A, et al. Self-Normalizing Neural Networks[J]. arXiv preprint arXiv:1706.02515, 2017.
[9] Ioffe S, Szegedy C. Batch
normalization: Accelerating deep network training by reducing internal
covariate shift[C]//International Conference on Machine Learning. 2015:
448-456.

梯度下降法的神经网络容易收敛到局部最优，为什么应用广泛？深度学习 DL基础 中

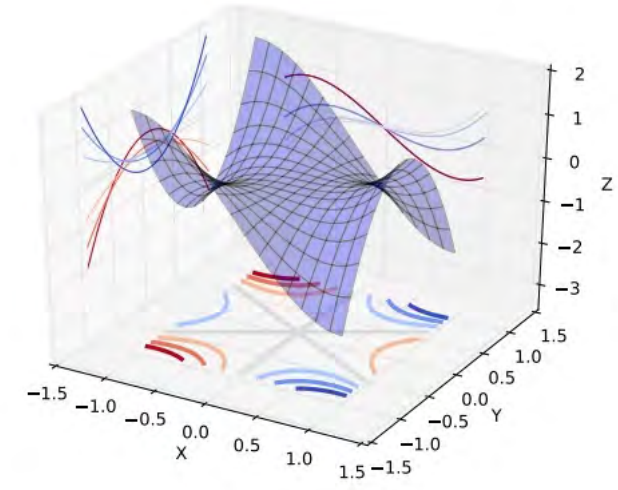
@李振华，https://www.zhihu.com/question/68109802/answer/262143638

深度神经网络“容易收敛到局部最优”，很可能是一种想象，实际情况是，我们可能从来没有找到过“局部最优”，更别说全局最优了。

很多人都有一种看法，就是“局部最优是神经网络优化的主要难点”。这来源于一维优化问题的直观想象。在单变量的情形下，优化问题最直观的困难就是有很多局部极值，如



人们直观的想象，高维的时候这样的局部极值会更多，指数级的增加，于是优化到全局最优就更难了。然而单变量到多变量一个重要差异是，单变量的时候，Hessian矩阵只有一个特征值，于是无论这个特征值的符号正负，一个临界点都是局部极值。但是在多变量的时候，Hessian有多个不同的特征值，这时候各个特征值就可能会有更复杂的分布，如有正有负的不定型和有多个退化特征值（零特征值）的半定型



在后两种情况下，是很难找到局部极值的，更别说全局最优了。

现在看来，神经网络的训练的困难主要是鞍点的问题。在实际中，我们很可能也从来没有真的遇到过局部极值。Bengio组这篇文章Eigenvalues of the Hessian in Deep Learning（https://arxiv.org/abs/1611.07476）里面的实验研究给出以下的结论：

> ? Training stops at a point that has a small gradient. The norm of the gradient is not zero, therefore it does not, technically speaking, converge to a critical point.
> ? There are still negative eigenvalues even when they are small in magnitude.

另一方面，一个好消息是，即使有局部极值，具有较差的loss的局部极值的吸引域也是很小的Towards Understanding Generalization of Deep Learning: Perspective of Loss Landscapes。（https://arxiv.org/abs/1706.10239）

> For the landscape of loss function for deep networks, the volume of basin of attraction of good minima dominates over that of poor minima, which guarantees optimization methods with random initialization to converge to good minima.

所以，很可能我们实际上是在“什么也没找到”的情况下就停止了训练，然后拿到测试集上试试，“咦，效果还不错”。

补充说明，这些都是实验研究结果。理论方面，各种假设下，深度神经网络的Landscape 的鞍点数目指数增加，而具有较差loss的局部极值非常少。

请比较下EM算法、HMM、CRF。机器学习 ML模型 中

这三个放在一起不是很恰当，但是有互相有关联，所以就放在这里一起说了。注意重点关注算法的思想。

（1）EM算法

EM算法是用于含有隐变量模型的极大似然估计或者极大后验估计，有两步组成：E步，求期望（expectation）；M步，求极大（maxmization）。本质上EM算法还是一个迭代算法，通过不断用上一代参数对隐变量的估计来对当前变量进行计算，直到收敛。

注意：EM算法是对初值敏感的，而且EM是不断求解下界的极大化逼近求解对数似然函数的极大化的算法，也就是说EM算法不能保证找到全局最优值。对于EM的导出方法也应该掌握。

（2）HMM算法

隐马尔可夫模型是用于标注问题的生成模型。有几个参数（π，A，B）：初始状态概率向量π，状态转移矩阵A，观测概率矩阵B。称为马尔科夫模型的三要素。
马尔科夫三个基本问题：

- 概率计算问题：给定模型和观测序列，计算模型下观测序列输出的概率。–》前向后向算法
- 学习问题：已知观测序列，估计模型参数，即用极大似然估计来估计参数。–》Baum-Welch(也就是EM算法)和极大似然估计。
- 预测问题：已知模型和观测序列，求解对应的状态序列。–》近似算法（贪心算法）和维比特算法（动态规划求最优路径）

（3）条件随机场CRF

给定一组输入随机变量的条件下另一组输出随机变量的条件概率分布密度。条件随机场假设输出变量构成马尔科夫随机场，而我们平时看到的大多是线性链条随机场，也就是由输入对输出进行预测的判别模型。求解方法为极大似然估计或正则化的极大似然估计。

之所以总把HMM和CRF进行比较，主要是因为CRF和HMM都利用了图的知识，但是CRF利用的是马尔科夫随机场（无向图），而HMM的基础是贝叶斯网络（有向图）。而且CRF也有：概率计算问题、学习问题和预测问题。大致计算方法和HMM类似，只不过不需要EM算法进行学习问题。

（4）HMM和CRF对比

其根本还是在于基本的理念不同，一个是生成模型，一个是判别模型，这也就导致了求解方式的不同。

CNN常用的几个模型。深度学习 DL模型 中

名称	特点
LeNet5	没啥特点-不过是第一个CNN应该要知道
AlexNet	引入了ReLU和dropout，引入数据增强、池化相互之间有覆盖，三个卷积一个最大池化+三个全连接层
VGGNet	采用1*1和3*3的卷积核以及2*2的最大池化使得层数变得更深。常用VGGNet-16和VGGNet19
Google Inception Net	这个在控制了计算量和参数量的同时，获得了比较好的分类性能，和上面相比有几个大的改进： 1、去除了最后的全连接层，而是用一个全局的平均池化来取代它； 2、引入Inception Module，这是一个4个分支结合的结构。所有的分支都用到了1*1的卷积，这是因为1*1性价比很高，可以用很少的参数达到非线性和特征变换。 3、Inception V2第二版将所有的5*5变成2个3*3，而且提出来著名的Batch Normalization； 4、Inception V3第三版就更变态了，把较大的二维卷积拆成了两个较小的一维卷积，加速运算、减少过拟合，同时还更改了Inception Module的结构。
微软ResNet残差神经网络(Residual Neural Network)	1、引入高速公路结构，可以让神经网络变得非常深 2、ResNet第二个版本将ReLU激活函数变成y=x的线性函数

带核的SVM为什么能分类非线性问题？
核函数的本质是两个函数的内积，而这个函数在SVM中可以表示成对于输入值的高维映射。注意核并不是直接对应映射，核只不过是一个内积 常用核函数及核函数的条件：
核函数选择的时候应该从线性核开始，而且在特征很多的情况下没有必要选择高斯核，应该从简单到难的选择模型。我们通常说的核函数指的是正定和函数，其充要条件是对于任意的x属于X，要求K对应的Gram矩阵要是半正定矩阵。
RBF核径向基，这类函数取值依赖于特定点间的距离，所以拉普拉斯核其实也是径向基核。
线性核：主要用于线性可分的情况
多项式核
Boosting和Bagging

（1）随机森林

随机森林改变了决策树容易过拟合的问题，这主要是由两个操作所优化的：

1）Boostrap从袋内有放回的抽取样本值

2）每次随机抽取一定数量的特征（通常为sqr(n)）。

分类问题：采用Bagging投票的方式选择类别频次最高的

回归问题：直接取每颗树结果的平均值。

常见参数	误差分析	优点	缺点
1、树最大深度 2、树的个数 3、节点上的最小样本数 4、特征数(sqr(n))	oob(out-of-bag) 将各个树的未采样样本作为预测样本统计误差 作为误分率	可以并行计算 不需要特征选择 可以总结出特征重要性 可以处理缺失数据 不需要额外设计测试集	在回归上不能输出连续结果

（2）Boosting之AdaBoost

Boosting的本质实际上是一个加法模型，通过改变训练样本权重学习多个分类器并进行一些线性组合。而Adaboost就是加法模型+指数损失函数+前项分布算法。Adaboost就是从弱分类器出发反复训练，在其中不断调整数据权重或者是概率分布，同时提高前一轮被弱分类器误分的样本的权值。最后用分类器进行投票表决（但是分类器的重要性不同）。

（3）Boosting之GBDT

将基分类器变成二叉树，回归用二叉回归树，分类用二叉分类树。和上面的Adaboost相比，回归树的损失函数为平方损失，同样可以用指数损失函数定义分类问题。但是对于一般损失函数怎么计算呢？GBDT（梯度提升决策树）是为了解决一般损失函数的优化问题，方法是用损失函数的负梯度在当前模型的值来模拟回归问题中残差的近似值。

注：由于GBDT很容易出现过拟合的问题，所以推荐的GBDT深度不要超过6，而随机森林可以在15以上。

（4）Xgboost

这个工具主要有以下几个特点：

- 支持线性分类器
- 可以自定义损失函数，并且可以用二阶偏导
- 加入了正则化项：叶节点数、每个叶节点输出score的L2-norm
- 支持特征抽样
- 在一定情况下支持并行，只有在建树的阶段才会用到，每个节点可以并行的寻找分裂特征。

逻辑回归相关问题

（1）公式推导一定要会

（2）逻辑回归的基本概念

这个最好从广义线性模型的角度分析，逻辑回归是假设y服从Bernoulli分布。

（3）L1-norm和L2-norm

其实稀疏的根本还是在于L0-norm也就是直接统计参数不为0的个数作为规则项，但实际上却不好执行于是引入了L1-norm；而L1norm本质上是假设参数先验是服从Laplace分布的，而L2-norm是假设参数先验为Gaussian分布，我们在网上看到的通常用图像来解答这个问题的原理就在这。

但是L1-norm的求解比较困难，可以用坐标轴下降法或是最小角回归法求解。

（4）LR和SVM对比

首先，LR和SVM最大的区别在于损失函数的选择，LR的损失函数为Log损失（或者说是逻辑损失都可以）、而SVM的损失函数为hinge loss。

$$\min_{w,b} \sum_{i}^{N} [1 - y_i(w \cdot x_i + b)]_+ + \lambda ||w||^2$$

其次，两者都是线性模型。

最后，SVM只考虑支持向量（也就是和分类相关的少数点）

（5）LR和随机森林区别

随机森林等树算法都是非线性的，而LR是线性的。LR更侧重全局优化，而树模型主要是局部的优化。

（6）常用的优化方法

逻辑回归本身是可以用公式求解的，但是因为需要求逆的复杂度太高，所以才引入了梯度下降算法。

一阶方法：梯度下降、随机梯度下降、mini 随机梯度下降降法。随机梯度下降不但速度上比原始梯度下降要快，局部最优化问题时可以一定程度上抑制局部最优解的发生。

二阶方法：牛顿法、拟牛顿法：

这里详细说一下牛顿法的基本原理和牛顿法的应用方式。牛顿法其实就是通过切线与x轴的交点不断更新切线的位置，直到达到曲线与x轴的交点得到方程解。在实际应用中我们因为常常要求解凸优化问题，也就是要求解函数一阶导数为0的位置，而牛顿法恰好可以给这种问题提供解决方法。实际应用中牛顿法首先选择一个点作为起始点，并进行一次二阶泰勒展开得到导数为0的点进行一个更新，直到达到要求，这时牛顿法也就成了二阶求解问题，比一阶方法更快。我们常常看到的x通常为一个多维向量，这也就引出了Hessian矩阵的概念（就是x的二阶导数矩阵）。缺点：牛顿法是定长迭代，没有步长因子，所以不能保证函数值稳定的下降，严重时甚至会失败。还有就是牛顿法要求函数一定是二阶可导的。而且计算Hessian矩阵的逆复杂度很大。

拟牛顿法： 不用二阶偏导而是构造出Hessian矩阵的近似正定对称矩阵的方法称为拟牛顿法。拟牛顿法的思路就是用一个特别的表达形式来模拟Hessian矩阵或者是他的逆使得表达式满足拟牛顿条件。主要有DFP法（逼近Hession的逆）、BFGS（直接逼近Hession矩阵）、 L-BFGS（可以减少BFGS所需的存储空间）。

用贝叶斯机率说明Dropout的原理

@许韩，来源：https://zhuanlan.zhihu.com/p/25005808

Dropout as a Bayesian Approximation: Insights and Applications

（http://mlg.eng.cam.ac.uk/yarin/PDFs/Dropout_as_a_Bayesian_approximation.pdf）

为什么很多做人脸的Paper会最后加入一个Local Connected Conv?

@许韩，来源：https://zhuanlan.zhihu.com/p/25005808

以FaceBook DeepFace 为例：

Figure 2. **Outline of the *DeepFace* architecture**. A front-end of a single convolution-pooling-convolution filtering on the rectified input, followed by three locally-connected layers and two fully-connected layers. Colors illustrate feature maps produced at each layer. The net includes more than 120 million parameters, where more than 95% come from the local and fully connected layers.

DeepFace 先进行了两次全卷积＋一次池化，提取了低层次的边缘 / 纹理等特征。后接了3个Local-Conv层，这里是用Local-Conv的原因是，人脸在不同的区域存在不同的特征（眼睛 / 鼻子 / 嘴的分布位置相对固定），当不存在全局的局部特征分布时，Local-Conv更适合特征的提取。

什么是共线性, 跟过拟合有什么关联?

@抽象猴，来源：https://www.zhihu.com/question/41233373/answer/145404190

共线性：多变量线性回归中，变量之间由于存在高度相关关系而使回归估计不准确。

共线性会造成冗余，导致过拟合。

解决方法：排除变量的相关性 / 加入权重正则。

为什么网络够深(Neurons 足够多)的时候，总是可以避开较差Local Optima?

参见：The Loss Surfaces of Multilayer Networks（https://arxiv.org/pdf/1412.0233.pdf）

机器学习中的正负样本

在分类问题中，这个问题相对好理解一点，比如人脸识别中的例子，正样本很好理解，就是人脸的图片，负样本的选取就与问题场景相关，具体而言，如果你要进行教室中学生的人脸识别，那么负样本就是教室的窗子、墙等等，也就是说，不能是与你要研究的问题毫不相关的乱七八糟的场景图片，这样的负样本并没有意义。负样本可以根据背景生成，有时候不需要寻找额外的负样本。一般3000-10000的正样本需要5，000,000-100,000,000的负样本来学习，在互金领域一般在入模前将正负比例通过采样的方法调整到3:1-5:1。

机器学习中，有哪些特征选择的工程方法?

数据和特征决定了机器学习的上限，而模型和算法只是逼近这个上限而已

1. 计算每一个特征与响应变量的相关性：工程上常用的手段有计算皮尔逊系数和互信息系数，皮尔逊系数只能衡量线性相关性而互信息系数能够很好地度量各种相关性，但是计算相对复杂一些，好在很多toolkit里边都包含了这个工具（如sklearn的MINE），得到相关性之后就可以排序选择特征了；

2. 构建单个特征的模型，通过模型的准确性为特征排序，借此来选择特征；

3.通过L1正则项来选择特征：L1正则方法具有稀疏解的特性，因此天然具备特征选择的特性，但是要注意，L1没有选到的特征不代表不重要，原因是两个具有高相关性的特征可能只保留了一个，如果要确定哪个特征重要应再通过L2正则方法交叉检验*；

4. 训练能够对特征打分的预选模型：RandomForest和Logistic Regression等都能对模型的特征打分，通过打分获得相关性后再训练最终模型；

5.通过特征组合后再来选择特征：如对用户id和用户特征最组合来获得较大的特征集再来选择特征，这种做法在推荐系统和广告系统中比较常见，这也是所谓亿级甚至十亿级特征的主要来源，原因是用户数据比较稀疏，组合特征能够同时兼顾全局模型和个性化模型，这个问题有机会可以展开讲。

6.通过深度学习来进行特征选择：目前这种手段正在随着深度学习的流行而成为一种手段，尤其是在计算机视觉领域，原因是深度学习具有自动学习特征的能力，这也是深度学习又叫unsupervised feature learning的原因。从深度学习模型中选择某一神经层的特征后就可以用来进行最终目标模型的训练了。

在一个n维的空间中， 最好的检测outlier(离群点)的方法是（）机器学习 ML基础 易

A. 作正态分布概率图

B. 作盒形图

C. 马氏距离

D. 作散点图

答案：C

> 马氏距离是基于卡方分布的，度量多元outlier离群点的统计方法。
> 有M个样本向量X1~Xm，协方差矩阵记为S，均值记为向量μ，则其中样本向量X到u的马氏距离表示为：
> $D(X) = \sqrt{(X-\mu)^T S^{-1}(X-\mu)}$
> （协方差矩阵中每个元素是各个矢量元素之间的协方差Cov(X,Y)，Cov(X,Y) = E{ [X-E(X)] [Y-E(Y)]}，其中E为数学期望）
> 而其中向量Xi与Xj之间的马氏距离定义为：
> $D(X_i,X_j) = \sqrt{(X_i-X_j)^T S^{-1}(X_i-X_j)}$
> 若协方差矩阵是单位矩阵（各个样本向量之间独立同分布）,则公式就成了：
> $D(X_i,X_j) = \sqrt{(X_i-X_j)^T (X_i-X_j)}$
> 也就是欧氏距离了。
> 若协方差矩阵是对角矩阵，公式变成了标准化欧氏距离。
> (2)马氏距离的优缺点：量纲无关，排除变量之间的相关性的干扰。

更多请详见：这里和”各种距离“。

对数几率回归（logistics regression）和一般回归分析有什么区别？机器学习 ML基础 易

A. 对数几率回归是设计用来预测事件可能性的

B. 对数几率回归可以用来度量模型拟合程度

C. 对数几率回归可以用来估计回归系数

D. 以上所有

答案：D

A: 这个在这篇文章里提到过，对数几率回归其实是设计用来解决分类问题的

B: 对数几率回归可以用来检验模型对数据的拟合度

C: 虽然对数几率回归是用来解决分类问题的，但是模型建立好后，就可以根据独立的特征，估计相关的回归系数。就我认为，这只是估计回归系数，不能直接用来做回归模型。

bootstrap数据是什么意思？（提示：考“bootstrap”和“boosting”区别）机器学习 ML模型 易

A. 有放回地从总共M个特征中抽样m个特征

B. 无放回地从总共M个特征中抽样m个特征

C. 有放回地从总共N个样本中抽样n个样本

D. 无放回地从总共N个样本中抽样n个样本

答案：C。boostrap是提鞋自举的意思(武侠小说作者所说的左脚踩右脚腾空而起). 它的过程是对样本(而不是特征)进行有放回的抽样, 抽样次数等同于样本总数. 这个随机抽样过程决定了最终抽样出来的样本, 去除重复之后, 占据原有样本的1/e比例。

“过拟合”只在监督学习中出现，在非监督学习中，没有“过拟合”，这是（）机器学习 ML基础 易

A. 对的

B. 错的

答案：B

我们可以评估无监督学习方法通过无监督学习的指标，如：我们可以评估聚类模型通过调整兰德系数（adjusted rand score）

对于k折交叉验证, 以下对k的说法正确的是（）机器学习 ML基础 易

A. k越大, 不一定越好, 选择大的k会加大评估时间

B. 选择更大的k, 就会有更小的bias (因为训练集更加接近总数据集)

C. 在选择k时, 要最小化数据集之间的方差

D. 以上所有

答案：D

k越大, bias越小, 训练时间越长. 在训练时, 也要考虑数据集间方差差别不大的原则. 比如, 对于二类分类问题, 使用2-折交叉验证, 如果测试集里的数据都是A类的, 而训练集中数据都是B类的, 显然, 测试效果会很差.

如果不明白bias和variance的概念, 务必参考下面链接:

Gentle Introduction to the Bias-Variance Trade-Off in Machine Learning

Understanding the Bias-Variance Tradeoff

回归模型中存在多重共线性, 你如何解决这个问题？机器学习 ML模型 中

A. 去除这两个共线性变量

B. 我们可以先去除一个共线性变量

C. 计算VIF(方差膨胀因子), 采取相应措施

D. 为了避免损失信息, 我们可以使用一些正则化方法, 比如, 岭回归和lasso回归.

以下哪些是对的:

A. 1

B. 2

C. 2和3

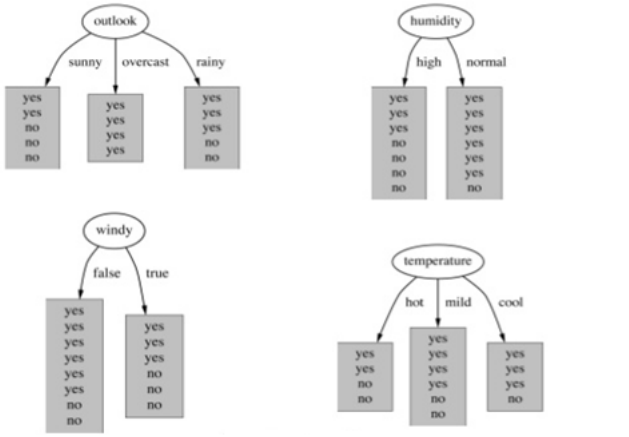
D. 2, 3和4

答案: D

解决多重公线性, 可以使用相关矩阵去去除相关性高于75%的变量 (有主观成分). 也可以VIF, 如果VIF值<=4说明相关性不是很高, VIF值>=10说明相关性较高.

我们也可以用 岭回归和lasso回归的带有惩罚正则项的方法. 我们也可以在一些变量上加随机噪声, 使得变量之间变得不同, 但是这个方法要小心使用, 可能会影响预测效果。

模型的高bias是什么意思, 我们如何降低它？ 机器学习 ML基础 易
A. 在特征空间中减少特征
B. 在特征空间中增加特征
C. 增加数据点
D. B和C
E. 以上所有
答案: B
bias太高说明模型太简单了, 数据维数不够, 无法准确预测数据, 所以, 升维吧！
训练决策树模型, 属性节点的分裂, 具有最大信息增益的图是下图的哪一个（）机器学习 ML模型 易

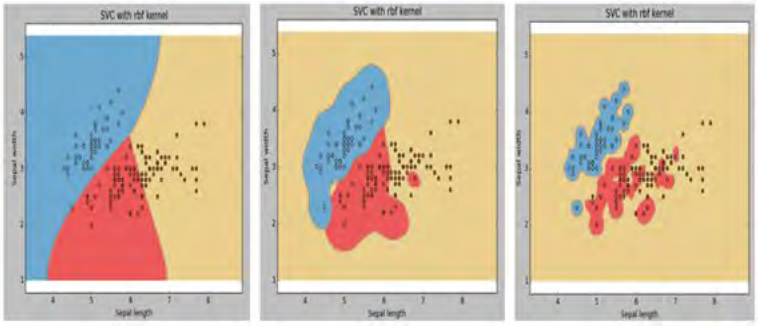


A. Outlook
B. Humidity
C. Windy
D. Temperature
答案: A信息增益, 增加平均子集纯度, 详细研究, 请戳下面链接:
A Complete Tutorial on Tree Based Modeling from Scratch (in R & Python)
Lecture 4 Decision Trees (2): Entropy, Information Gain, Gain Ratio
对于信息增益, 决策树分裂节点, 下面说法正确的是（）机器学习 ML模型 易
A. 纯度高的节点需要更多的信息去区分
B. 信息增益可以用”1比特-熵”获得
C. 如果选择一个属性具有许多归类值, 那么这个信息增益是有偏差的
A. 1
B. 2
C.2和3
D. 所有以上
答案: C
详细研究, 请戳下面链接:
A Complete Tutorial on Tree Based Modeling from Scratch (in R & Python)
Lecture 4 Decision Trees (2): Entropy, Information Gain, Gain Ratio
如果SVM模型欠拟合, 以下方法哪些可以改进模型 （） 机器学习 ML模型 中
A. 增大惩罚参数C的值
B. 减小惩罚参数C的值
C. 减小核系数(gamma参数)
@David 9，本题解析来源：http://nooverfit.com/wp/12-%E6%9C%BA%E5%99%A8%E5%AD%A6%E4%B9%A0%E8%83%BD%E5%8A%9B%E8%87%AA%E6%B5%8B%E9%A2%98-%E7%9C%8B%E7%9C%8B%E4%BD%A0%E7%9A%84%E6%9C%BA%E5%99%A8%E5%AD%A6%E4%B9%A0%E7%9F%A5%E8%AF%86%E8%8A如果SVM模型欠拟合, 我们可以调高参数C的值, 使得模型复杂度上升.LibSVM中，SVM的目标函数是:

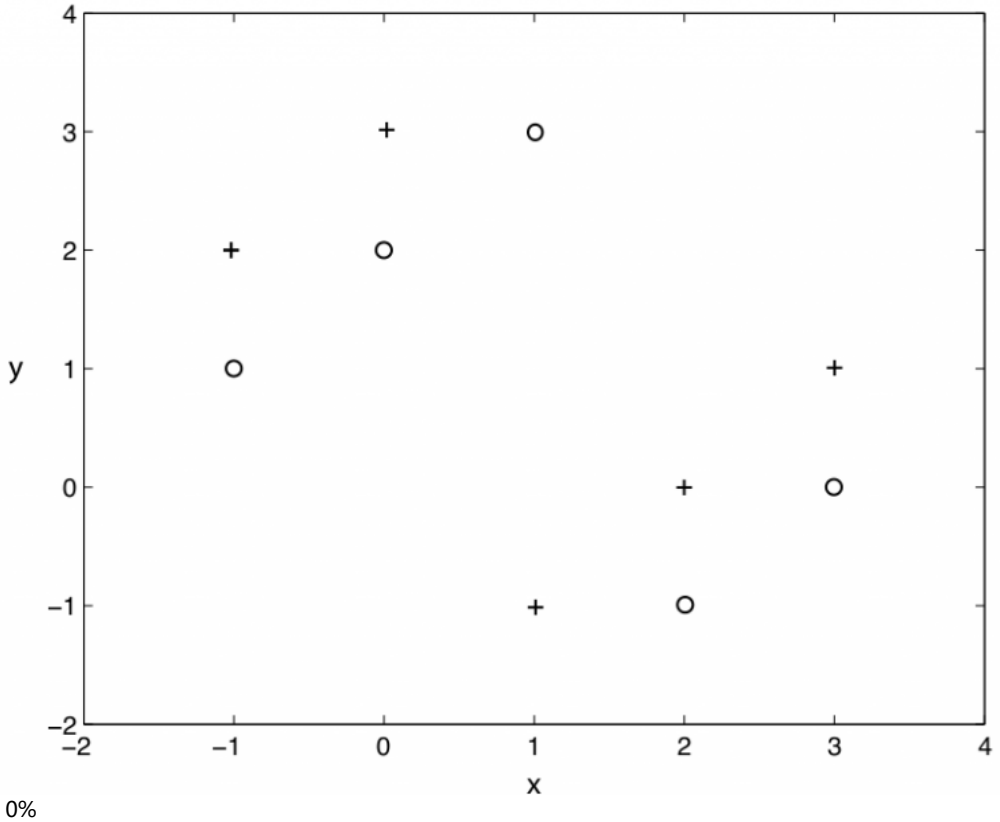
$$\begin{aligned} \min_{\boldsymbol{w},b,\boldsymbol{\xi}} \quad & \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w} + C\sum_{i=1}^{l}\xi_i \\ \text{subject to} \quad & y_i(\boldsymbol{w}^T\phi(\boldsymbol{x}_i) + b) \geq 1 - \xi_i, \\ & \xi_i \geq 0, i = 1, \ldots, l, \end{aligned}$$

而, gamma参数是你选择径向基函数作为kernel后,该函数自带的一个参数.隐含地决定了数据映射到新的特征空间后的分布.
gamma参数与C参数无关. gamma参数越高, 模型越复杂.

下图是同一个SVM模型, 但是使用了不同的径向基核函数的gamma参数, 依次是g1, g2, g3 , 下面大小比较正确的是:



A. g1 > g2 > g3
B. g1 = g2 = g3
C. g1 < g2 < g3
D. g1 >= g2 >= g3E. g1 <= g2 <= g3
答案: C参考Q10题
假设我们要解决一个二类分类问题, 我们已经建立好了模型, 输出是0或1, 初始时设阈值为0.5, 超过0.5概率估计, 就判别为1, 否则就判别为0 ; 如果我们现在用另一个大于0.5的阈值, 那么现在关于模型说法, 正确的是 :
A. 模型分类的召回率会降低或不变
B. 模型分类的召回率会升高
C. 模型分类准确率会升高或不变
D. 模型分类准确率会降低
A. 1
B. 2
C.1和3
D. 2和4
E. 以上都不是
答案: C
这篇文章讲述了阈值对准确率和召回率影响 :
Confidence Splitting Criterions Can Improve Precision And Recall in Random Forest Classifiers“点击率问题”是这样一个预测问题, 99%的人是不会点击的, 而1%的人是会点击进去的, 所以这是一个非常不平衡的数据集. 假设, 现在我们已经建了一个模型来分类, 而且有了99%的预测准确率, 我们可以下的结论是 :
A. 模型预测准确率已经很高了, 我们不需要做什么了
B. 模型预测准确率不高, 我们需要做点什么改进模型
C. 无法下结论
D. 以上都不对
答案: B
99%的预测准确率可能说明, 你预测的没有点进去的人很准确 (因为有99%的人是不会点进去的, 这很好预测). 不能说明你的模型对点进去的人预测准确, 所以, 对于这样的非平衡数据集, 我们要把注意力放在小部分的数据上, 即那些点击进去的人.
详细可以参考这篇文章: article
使用k=1的knn算法, 下图二类分类问题, “+” 和 “o” 分别代表两个类, 那么, 用仅拿出一个测试样本的交叉验证方法, 交叉验证的错误率是多少:



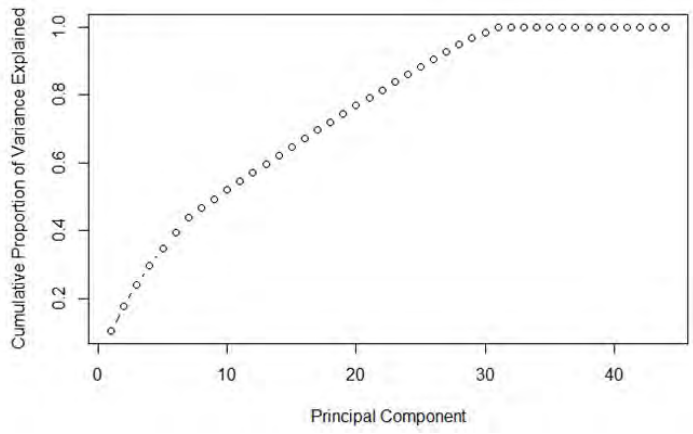
A. 0%
B. 100%

C. 0% 到 100
D. 以上都不是答案: B
knn算法就是, 在样本周围看k个样本, 其中大多数样本的分类是A类, 我们就把这个样本分成A类. 显然, k=1 的knn在上图不是一个好选择, 分类的错误率始终是100%
我们想在大数据集上训练决策树, 为了使用较少时间, 我们可以:
A. 增加树的深度
B. 增加学习率 (learning rate)
C. 减少树的深度
D. 减少树的数量
答案: C
增加树的深度, 会导致所有节点不断分裂, 直到叶子节点是纯的为止. 所以, 增加深度, 会延长训练时间.
决策树没有学习率参数可以调. (不像集成学习和其它有步长的学习方法)
决策树只有一棵树, 不是随机森林.
对于神经网络的说法, 下面正确的是 :
1. 增加神经网络层数, 可能会增加测试数据集的分类错误率
2. 减少神经网络层数, 总是能减小测试数据集的分类错误率
3. 增加神经网络层数, 总是能减小训练数据集的分类错误率
A. 1
B. 1 和 3
C. 1 和 2
D. 2
答案: A
深度神经网络的成功, 已经证明, 增加神经网络层数, 可以增加模型范化能力, 即, 训练数据集和测试数据集都表现得更好. 但更多的层数, 也不一定能保证有更好的表现（https://arxiv.org/pdf/1512.03385v1.pdf）. 所以, 不能绝对地说层数多的好坏, 只能选A
假如我们使用非线性可分的SVM目标函数作为最优化对象, 我们怎么保证模型线性可分?
A. 设C=1
B. 设C=0
C. 设C=无穷大
D. 以上都不对
答案: C
C无穷大保证了所有的线性不可分都是可以忍受的.
训练完SVM模型后, 不是支持向量的那些样本我们可以丢掉, 也可以继续分类:
A. 正确
B. 错误
答案: A
SVM模型中, 真正影响决策边界的是支持向量
以下哪些算法, 可以用神经网络去构造:
1. KNN
2. 线性回归
3. 对数几率回归
A. 1和 2
B. 2 和 3
C. 1, 2 和 3
D. 以上都不是
答案: B
1. KNN算法不需要训练参数, 而所有神经网络都需要训练参数, 因此神经网络帮不上忙
2. 最简单的神经网络, 感知器, 其实就是线性回归的训练
3. 我们可以用一层的神经网络构造对数几率回归
请选择下面可以应用隐马尔科夫(HMM)模型的选项:
A. 基因序列数据集
B. 电影浏览数据集
C. 股票市场数据集
D. 所有以上
答案: D
只要是和时间序列问题有关的 , 都可以试试HMM
我们建立一个5000个特征, 100万数据的机器学习模型. 我们怎么有效地应对这样的大数据训练 :
A. 我们随机抽取一些样本, 在这些少量样本之上训练
B. 我们可以试用在线机器学习算法
C. 我们应用PCA算法降维, 减少特征数
D. B 和 C
E. A 和 B
F. 以上所有
答案: F252.我们想要减少数据集中的特征数, 即降维. 选择以下适合的方案 :
1. 使用前向特征选择方法
2. 使用后向特征排除方法
3. 我们先把所有特征都使用, 去训练一个模型, 得到测试集上的表现. 然后我们去掉一个特征, 再去训练, 用交叉验证看看测试集上的表现. 如果表现比原来还要好, 我们可以去除这个特征.
4. 查看相关性表, 去除相关性最高的一些特征
A. 1 和 2
B. 2, 3和4
C. 1, 2和4
D. All

答案: D

1.前向特征选择方法和后向特征排除方法是我们特征选择的常用方法
2.如果前向特征选择方法和后向特征排除方法在大数据上不适用, 可以用这里第三种方法.
3.用相关性的度量去删除多余特征, 也是一个好方法
所有D是正确的对于随机森林和GradientBoosting Trees, 下面说法正确的是:
1.在随机森林的单个树中, 树和树之间是有依赖的, 而GradientBoosting Trees中的单个树之间是没有依赖的.
2.这两个模型都使用随机特征子集, 来生成许多单个的树.
3.我们可以并行地生成GradientBoosting Trees单个树, 因为它们之间是没有依赖的, GradientBoosting Trees训练模型的表现总是比随机森林好
A. 2
B. 1 and 2
C. 1, 3 and 4
D. 2 and 4

答案: A
1.随机森林是基于bagging的, 而Gradient Boosting trees是基于boosting的, 所有说反了,在随机森林的单个树中, 树和树之间是没有依赖的, 而GradientBoosting Trees中的单个树之间是有依赖关系.
2.这两个模型都使用随机特征子集, 来生成许多单个的树.
所有A是正确的对于PCA(主成分分析)转化过的特征 , 朴素贝叶斯的”不依赖假设” 总是成立, 因为所有主要成分是正交的, 这个说法是 :
A. 正确的
B. 错误的
答案: B.
这个说法是错误的, 首先, “不依赖”和”不相关”是两回事, 其次, 转化过的特征, 也可能是相关的对于PCA说法正确的是 :
1. 我们必须在使用PCA前规范化数据
2. 我们应该选择使得模型有最大variance的主成分
3. 我们应该选择使得模型有最小variance的主成分
4. 我们可以使用PCA在低维度上做数据可视化
A. 1, 2 and 4
B. 2 and 4
C. 3 and 4
D. 1 and 3
E. 1, 3 and 4
答案: A
1）PCA对数据尺度很敏感, 打个比方, 如果单位是从km变为cm, 这样的数据尺度对PCA最后的结果可能很有影响(从不怎么重要的成分变为很重要的成分).
2）我们总是应该选择使得模型有最大variance的主成分
3）有时在低维度上左图是需要PCA的降维帮助的
对于下图, 最好的主成分选择是多少 ? :



A. 7

B. 30

C. 35

D. Can’t Say

答案: B

- 主成分选择使variance越大越好， 在这个前提下， 主成分越少越好。

数据科学家可能会同时使用多个算法（模型）进行预测， 并且最后把这些算法的结果集成起来进行最后的预测（集成学习），以下对集成学习说法正确的是 :

A. 单个模型之间有高相关性

B. 单个模型之间有低相关性

C. 在集成学习中使用“平均权重”而不是“投票”会比较好

D. 单个模型都是用的一个算法

答案: B

- 详细请参考下面文章:
  - Basics of Ensemble Learning Explained in Simple English
  - Kaggle Ensemble Guide
  - 5 Easy questions on Ensemble Modeling everyone should know

在有监督学习中， 我们如何使用聚类方法? :

A. 我们可以先创建聚类类别， 然后在每个类别上用监督学习分别进行学习

B. 我们可以使用聚类“类别id”作为一个新的特征项， 然后再用监督学习分别进行学习

C. 在进行监督学习之前， 我们不能新建聚类类别

D. 我们不可以使用聚类“类别id”作为一个新的特征项， 然后再用监督学习分别进行学习

A. 2 和 4

B. 1 和 2

C. 3 和 4

D. 1 和 3

答案: B

我们可以为每个聚类构建不同的模型， 提高预测准确率。

“类别id”作为一个特征项去训练， 可以有效地总结了数据特征。

所以B是正确的

以下说法正确的是：

A. 一个机器学习模型，如果有较高准确率，总是说明这个分类器是好的

B. 如果增加模型复杂度， 那么模型的测试错误率总是会降低

C. 如果增加模型复杂度， 那么模型的训练错误率总是会降低

D. 我们不可以使用聚类“类别id”作为一个新的特征项， 然后再用监督学习分别进行学习

A. 1

B. 2

C. 3

D. 1 and 3

答案: C

考的是过拟合和欠拟合的问题。

对应GradientBoosting tree算法， 以下说法正确的是：

A. 当增加最小样本分裂个数，我们可以抵制过拟合

B. 当增加最小样本分裂个数，会导致过拟合

C. 当我们减少训练单个学习器的样本个数，我们可以降低variance

D. 当我们减少训练单个学习器的样本个数，我们可以降低bias

A. 2 和 4

B. 2 和 3

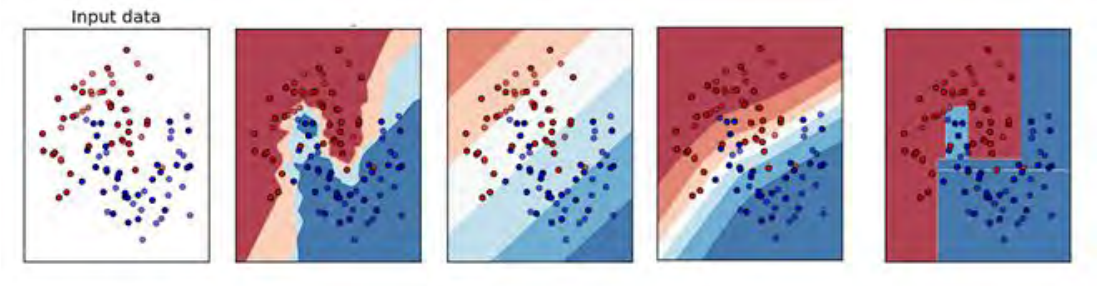
C. 1 和 3

D. 1 和 4

答案: C

- 最小样本分裂个数是用来控制“过拟合”参数。太高的值会导致“欠拟合”，这个参数应该用交叉验证来调节。
- 第二点是靠bias和variance概念的。

以下哪个图是KNN算法的训练边界：



A) B

B) A

C) D

D) C

E) 都不是

答案: B

KNN算法肯定不是线性的边界， 所以直的边界就不用考虑了。另外这个算法是看周围最近的k个样本的分类用以确定分类，所以边界一定是坑坑洼洼的。

如果一个训练好的模型在测试集上有100%的准确率， 这是不是意味着在一个新的数据集上，也会有同样好的表现？ ：

A. 是的，这说明这个模型的范化能力已经足以支持新的数据集合了

B. 不对，依然后其他因素模型没有考虑到，比如噪音数据

答案: B
没有一个模型是可以总是适应新数据的。我们不可能可到100%准确率。

Q33：下面的交叉验证方法：

i. 有放回的Bootstrap方法

ii. 留一个测试样本的交叉验证

iii. 5折交叉验证

iv. 重复两次的5折教程验证

当样本是1000时，下面执行时间的顺序，正确的是：

A. i > ii > iii > iv

B. ii > iv > iii > i

C. iv > i > ii > iii

D. ii > iii > iv > i

答案：B

- Boostrap方法是传统地随机抽样，验证一次的验证方法，只需要训练1次模型，所以时间最少。
- 留一个测试样本的交叉验证，需要n次训练过程（n是样本个数），这里，要训练1000个模型。
- 5折交叉验证需要训练5个模型。
- 重复2次的5折交叉验证，需要训练10个模型。

所有B是正确的

变量选择是用来选择最好的判别器子集， 如果要考虑模型效率，我们应该做哪些变量选择的考虑？ ：

1. 多个变量其实有相同的用处
2. 变量对于模型的解释有多大作用
3. 特征携带的信息
4. 交叉验证

A. 1 和 4

B. 1, 2 和 3

C. 1,3 和 4

D. 以上所有

答案：C

注意， 这题的题眼是考虑模型效率，所以不要考虑选项2.

对于线性回归模型，包括附加变量在内，以下的可能正确的是：

1. R-Squared 和 Adjusted R-squared都是递增的
2. R-Squared 是常量的，Adjusted R-squared是递增的
3. R-Squared 是递减的， Adjusted R-squared 也是递减的
4. R-Squared 是递减的， Adjusted R-squared是递增的

A. 1 和 2

B. 1 和 3

C. 2 和 4

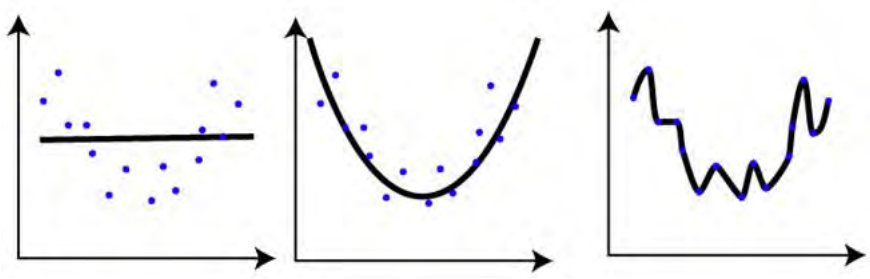
D. 以上都不是

答案：D

R-squared不能决定系数估计和预测偏差，这就是为什么我们要估计残差图。但是，R-squared有R-squared 和 predicted R-squared 所没有的问题。
每次你为模型加入预测器，R-squared递增或不变.

详细请看这个链接：discussion.

对于下面三个模型的训练情况， 下面说法正确的是：



1. 第一张图的训练错误与其余两张图相比，是最大的
2. 最后一张图的训练效果最好，因为训练错误最小
3. 第二张图比第一和第三张图鲁棒性更强，是三个里面表现最好的模型
4. 第三张图相对前两张图过拟合了
5. 三个图表现一样，因为我们还没有测试数据集

A. 1 和 3

B. 1 和 3

C. 1, 3 和 4

D. 5

答案：C

对于线性回归，我们应该有以下哪些假设？ ：

1. 找到利群点很重要，因为线性回归对利群点很敏感
2. 线性回归要求所有变量必须符合正态分布
3. 线性回归假设数据没有多重线性相关性

A. 1 和 2

B. 2 和 3

C. 1,2 和 3

D. 以上都不是

答案: D

- 利群点要着重考虑, 第一点是对的
- 不是必须的, 当然, 如果是正态分布, 训练效果会更好
- 有少量的多重线性相关性是可以的, 但是我们要尽量避免

当我们构造线性模型时, 我们注意变量间的相关性. 在相关矩阵中搜索相关系数时, 如果我们发现3对变量的相关系数是(Var1 和Var2, Var2和Var3, Var3和Var1)是-0.98, 0.45, 1.23. 我们可以得出什么结论:

1. Var1和Var2是非常相关的
2. 因为Var和Var2是非常相关的, 我们可以去除其中一个
3. Var3和Var1的1.23相关系数是不可能的

A. 1 and 3

B. 1 and 2

C. 1,2 and 3

D. 1

答案: C

- Var1和Var2相关系数是负的, 所以这是多重线性相关, 我们可以考虑去除其中一个.
- 一般地, 如果相关系数大于0.7或者小于-0.7, 是高相关的
- 相关性系数范围应该是 [-1,1]

如果在一个高度非线性并且复杂的一些变量中, 一个树模型可能比一般的回归模型效果更好. 只是:

A. 对的

B. 错的

答案: A

对于维度极低的特征，选择线性还是非线性分类器?
非线性分类器，低维空间可能很多特征都跑到一起了，导致线性不可分。
1. 如果Feature的数量很大，跟样本数量差不多，这时候选用LR或者是Linear Kernel的SVM
2. 如果Feature的数量比较小，样本数量一般，不算大也不算小，选用SVM+Gaussian Kernel
3. 如果Feature的数量比较小，而样本数量很多，需要手工添加一些feature变成第一种情况。

特征向量的缺失值处理
1. 缺失值较多.直接将该特征舍弃掉，否则可能反倒会带入较大的noise，对结果造成不良影响。
2. 缺失值较少,其余的特征缺失值都在10%以内，我们可以采取很多的方式来处理:
1) 把NaN直接作为一个特征，假设用0表示;
2) 用均值填充;
3) 用随机森林等算法预测填充

SVM、LR、决策树的对比。
模型复杂度：SVM支持核函数，可处理线性非线性问题;LR模型简单，训练速度快，适合处理线性问题;决策树容易过拟合，需要进行剪枝
损失函数：SVM hinge loss; LR L2正则化; adaboost 指数损失
数据敏感度：SVM添加容忍度对outlier不敏感，只关心支持向量，且需要先做归一化; LR对远点敏感
数据量：数据量大就用LR，数据量小且特征少就用SVM非线性核

什么是ill-condition病态问题? 的
训练完的模型，测试样本稍作修改就会得到差别很大的结果，就是病态问题，模型对未知数据的预测能力很差，即泛化误差大。

简述KNN最近邻分类算法的过程?
1. 计算训练样本和测试样本中每个样本点的距离（常见的距离度量有欧式距离，马氏距离等）;
2. 对上面所有的距离值进行排序;
3. 选前k个最小距离的样本;
4. 根据这k个样本的标签进行投票，得到最后的分类类别;

常用的聚类划分方式有哪些? 列举代表算法。
1. 基于划分的聚类:K-means，k-medoids，CLARANS。
2. 基于层次的聚类：AGNES（自底向上），DIANA（自上向下）。
3. 基于密度的聚类：DBSACN，OPTICS，BIRCH(CF-Tree)，CURE。
4. 基于网格的方法：STING，WaveCluster。
5. 基于模型的聚类：EM,SOM，COBWEB。

下面对集成学习模型中的弱学习者描述错误的是?
A.他们经常不会过拟合
B.他们通常带有高偏差，所以其并不能解决复杂学习问题
C.他们通常会过拟合
答案：C，弱学习者是问题的特定部分。所以他们通常不会过拟合，这也就意味着弱学习者通常拥有低方差和高偏差。

下面哪个/些选项对 K 折交叉验证的描述是正确的?
1.增大 K 将导致交叉验证结果时需要更多的时间
2.更大的 K 值相比于小 K 值将对交叉验证结构有更高的信心
3.如果 K=N，那么其称为留一交叉验证，其中 N 为验证集中的样本数量
A. 1 和 2
B. 2 和 3
C. 1 和 3
D. 1、2 和 3
答案（D)：大 K 值意味着对过高估计真实预期误差（训练的折数将更接近于整个验证集样本数）拥有更小的偏差和更多的运行时间（并随着越来越接近极限情况：留一交叉验证）。我们同样在选择 K 值时需要考虑 K 折准确度和方差间的均衡。

最出名的降维算法是 PAC 和 t-SNE。将这两个算法分别应用到数据「X」上，并得到数据集「X_projected_PCA」，「X_projected_tSNE」。下面哪一项对「X_projected_PCA」和「X_projected_tSNE」的描述是正确的?
A. X_projected_PCA 在最近邻空间能得到解释
B. X_projected_tSNE 在最近邻空间能得到解释
C. 两个都在最近邻空间能得到解释
D. 两个都不能在最近邻空间得到解释
答案（B）：t-SNE 算法考虑最近邻点而减少数据维度。所以在使用 t-SNE 之后，所降的维可以在最近邻空间得到解释。但 PCA 不能。

给定三个变量 X，Y，Z。(X, Y)、(Y, Z) 和 (X, Z) 的 Pearson 相关性系数分别为 C1、C2 和 C3。现在 X 的所有值加 2（即 X+2），Y 的全部值减 2（即 Y-2），Z 保持不变。那么运算之后的 (X, Y)、(Y, Z) 和 (X, Z) 相关性系数分别为 D1、D2 和 D3。现在试问 D1、D2、D3 和 C1、C2、C3 之间的关系是什么?
A. D1= C1, D2 < C2, D3 > C3
B. D1 = C1, D2 > C2, D3 > C3

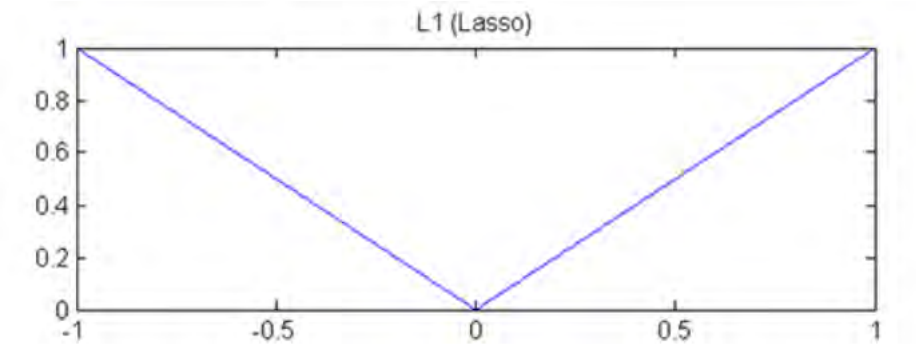
C. D1 = C1, D2 > C2, D3 < C3
D. D1 = C1, D2 < C2, D3 < C3
E. D1 = C1, D2 = C2, D3 = C3
答案（E）：特征之间的相关性系数不会因为特征加或减去一个数而改变。

为了得到和 SVD 一样的投射（projection），你需要在 PCA 中怎样做?
A. 将数据转换成零均值
B. 将数据转换成零中位数
C. 无法做到
答案（A）：当数据有一个 0 均值向量时，PCA 有与 SVD 一样的投射，否则在使用 SVD 之前，你必须将数据均值归 0。

假设我们有一个数据集，在一个深度为 6 的决策树的帮助下，它可以使用 100% 的精确度被训练。现在考虑一下两点，并基于这两点选择正确的选项。
注意：所有其他超参数是相同的，所有其他因子不受影响。
1.深度为 4 时将有高偏差和低方差
2.深度为 4 时将有低偏差和低方差
A. 只有 1
B. 只有 2
C. 1 和 2
D. 没有一个
答案（A）：如果在这样的数据中你拟合深度为 4 的决策树，这意味着其更有可能与数据欠拟合。因此，在欠拟合的情况下，你将获得高偏差和低方差。

在 k-均值算法中，以下哪个选项可用于获得全局最小?
A. 尝试为不同的质心（centroid）初始化运行算法
B. 调整迭代的次数
C. 找到集群的最佳数量
D. 以上所有
答案（D）：所有都可以用来调试以找到全局最小。

你正在使用带有 L1 正则化的 logistic 回归做二分类，其中 C 是正则化参数，w1 和 w2 是 x1 和 x2 的系数。当你把 C 值从 0 增加至非常大的值时，下面哪个选项是正确的?
A. 第一个 w2 成了 0，接着 w1 也成了 0
B. 第一个 w1 成了 0，接着 w2 也成了 0
C. w1 和 w2 同时成了 0



D. 即使在 C 成为大值之后，w1 和 w2 都不能成 0
答案（C）：L1正则化的函数如下图，所以w1和w2可以为0。同时w1和w2是对称的，不会导致一个为0另一个不为0的状态。

假设你使用 log-loss 函数作为评估标准。下面这些选项，哪些是对作为评估标准的 log-loss 的正确解释。
A.如果一个分类器对不正确的分类很自信，log-loss 会严重的批评它。
B.对一个特别的观察而言，分类器为正确的类别分配非常小的概率，然后对 log-loss 的相应分布会非常大。
C.log-loss 越低，模型越好
D.以上都是
答案为（D）

下面哪个选项中哪一项属于确定性算法?
A.PCA
B.K-Means
C. 以上都不是
答案为（A）：确定性算法表明在不同运行中，算法输出并不会改变。如果我们再一次运行算法，PCA 会得出相同的结果，而 k-means 不会。

特征向量的归一化方法有哪些?
线性函数转换，表达式如下:
y=(x-MinValue)/(MaxValue-MinValue)
对数函数转换，表达式如下:
y=log10 (x)
反余切函数转换 ，表达式如下:
y=arctan(x)*2/PI
减去均值，除以方差:
y=(x-means)/ variance

优化算法及其优缺点?
温馨提示：在回答面试官的问题的时候，往往将问题往大的方面去回答，这样不会陷于小的技术上死磕，最后很容易把自己嗑死了。
简言之
1）随机梯度下降
优点：可以一定程度上解决局部最优解的问题
缺点：收敛速度较慢
2）批量梯度下降
优点：容易陷入局部最优解
缺点：收敛速度较快
3）mini_batch梯度下降
综合随机梯度下降和批量梯度下降的优缺点，提取的一个中和的方法。
4）牛顿法
牛顿法在迭代的时候，需要计算Hessian矩阵，当维度较高的时候，计算 Hessian矩阵比较困难。
5）拟牛顿法
拟牛顿法是为了改进牛顿法在迭代过程中，计算Hessian矩阵而提取的算法，它采用的方式是通过逼近Hessian的方式来进行求解。

具体而言
从每个batch的数据来区分
梯度下降：每次使用全部数据集进行训练
优点：得到的是最优解
缺点：运行速度慢，内存可能不够
随机梯度下降：每次使用一个数据进行训练
优点：训练速度快，无内存问题

缺点：容易震荡，可能达不到最优解
Mini-batch梯度下降
优点：训练速度快，无内存问题，震荡较少
缺点：可能达不到最优解
从优化方法上来分：
随机梯度下降（SGD）
缺点
选择合适的learning　　rate比较难
对于所有的参数使用同样的learning rate
容易收敛到局部最优
可能困在saddle point
SGD+Momentum
优点：
积累动量，加速训练
局部极值附近震荡时，由于动量，跳出陷阱
梯度方向发生变化时，动量缓解动荡。
Nesterov Mementum
与Mementum类似，优点：
避免前进太快
提高灵敏度
AdaGrad
优点：
控制学习率，每一个分量有各自不同的学习率
适合稀疏数据
缺点
依赖一个全局学习率
学习率设置太大，其影响过于敏感
后期，调整学习率的分母积累的太大，导致学习率很低，提前结束训练。
RMSProp
优点：
解决了后期提前结束的问题。
缺点：
依然依赖全局学习率
Adam
Adagrad和RMSProp的合体
优点：
结合了Adagrad善于处理稀疏梯度和RMSprop善于处理非平稳目标的优点
为不同的参数计算不同的自适应学习率
也适用于大多非凸优化 - 适用于大数据集和高维空间
牛顿法
牛顿法在迭代的时候，需要计算Hessian矩阵，当维度较高的时候，计算 Hessian矩阵比较困难
拟牛顿法
拟牛顿法是为了改进牛顿法在迭代过程中，计算Hessian矩阵而提取的算法，它采用的方式是通过逼近Hessian的方式来进行求解。

RF与GBDT之间的区别与联系？
1）相同点：都是由多棵树组成，最终的结果都是由多棵树一起决定。
2）不同点：
a 组成随机森林的树可以分类树也可以是回归树，而GBDT只由回归树组成
b 组成随机森林的树可以并行生成，而GBDT是串行生成
c 随机森林的结果是多数表决表决的，而GBDT则是多棵树累加之和
d 随机森林对异常值不敏感，而GBDT对异常值比较敏感
e 随机森林是减少模型的方差，而GBDT是减少模型的偏差
f 随机森林不需要进行特征归一化。而GBDT则需要进行特征归一化

两个变量的 Pearson 相关性系数为零，但这两个变量的值同样可以相关。
A 正确
B 错误
答案为（A）：Pearson相关系数只能衡量线性相关性，但无法衡量非线性关系。如y=x^2，x和y有很强的非线性关系。

下面哪个/些超参数的增加可能会造成随机森林数据过拟合?
A 树的数量
B 树的深度
C 学习速率
答案为（B）：通常情况下，我们增加树的深度有可能会造成模型过拟合。学习速率并不是随机森林的超参数。增加树的数量可能会造成欠拟合。

目标变量在训练集上的 8 个实际值 [0,0,0,1,1,1,1,1]，目标变量的熵是多少?
A. -(5/8 log(5/8) + 3/8 log(3/8))
B. 5/8 log(5/8) + 3/8 log(3/8)
C. 3/8 log(5/8) + 5/8 log(3/8)
D. 5/8 log(3/8) − 3/8 log(5/8)
答案为（A）

下面有关序列模式挖掘算法的描述，错误的是？（C）
A AprioriAll算法和GSP算法都属于Apriori类算法，都要产生大量的候选序列
B FreeSpan算法和PrefixSpan算法不生成大量的候选序列以及不需要反复扫描原数据库
C 在时空的执行效率上，FreeSpan比PrefixSpan更优
D 和AprioriAll相比，GSP的执行效率比较高
@CS青雀，本题解析来源：http://blog.csdn.net/ztf312/article/details/50889238
1. Apriori算法：关联分析原始算法，用于从候选项集中发现频繁项集。两个步骤：进行自连接、进行剪枝。缺点：无时序先后性。
AprioriAll算法：AprioriAll算法与Apriori算法的执行过程是一样的，不同点在于候选集的产生，需要区分最后两个元素的前后。
AprioriSome算法：可以看做是AprioriAll算法的改进
AprioriAll算法和AprioriSome算法的比较：
（1）AprioriAll用 去计算出所有的候选Ck，而AprioriSome会直接用 去计算所有的候选 ，因为 包含 ，所以AprioriSome会产生比较多的候选。
（2）虽然AprioriSome跳跃式计算候选，但因为它所产生的候选比较多，可能在回溯阶段前就占满内存。
（3）如果内存占满了，AprioriSome就会被迫去计算最后一组的候选。
（4）对于较低的支持度，有较长的大序列，AprioriSome算法要好些。
2. GPS算法：类Apriori算法。用于从候选项集中发现具有时序先后性的频繁项集。两个步骤：进行自连接、进行剪枝。缺点：每次计算支持度，都需要扫描全部数据集；对序列模式很长的情况，由于其对应的短的序列模式规模太大，算法很难处理。
3. SPADE算法：改进的GPS算法，规避多次对数据集D进行全表扫描的问题。与GSP算法大体相同，多了一个ID_LIST记录，使得每一次的ID_LIST根据上一次的ID_LIST得到（从而得到支持度）。而ID_LIST的规模是随着剪枝的不断进行而缩小的。所以也就解决了GSP算法多次扫描数据集D问题。
4．FreeSpan算法：即频繁模式投影的序列模式挖掘。核心思想是分治算法。基本思想为：利用频繁项递归地将序列数据库投影到更小的投影数据库集中，在每个投影数据库中生成子序列片断。这一过程对数据和待检验的频繁模式集进行了分割，并且将每一次检验限制在与其相符合的更小的投影数据库中。
优点：减少产生候选序列所需的开销。缺点：可能会产生许多投影数据库，开销很大，会产生很多的
5. PrefixSpan 算法：从FreeSpan中推导演化而来的。收缩速度比FreeSpan还要更快些。

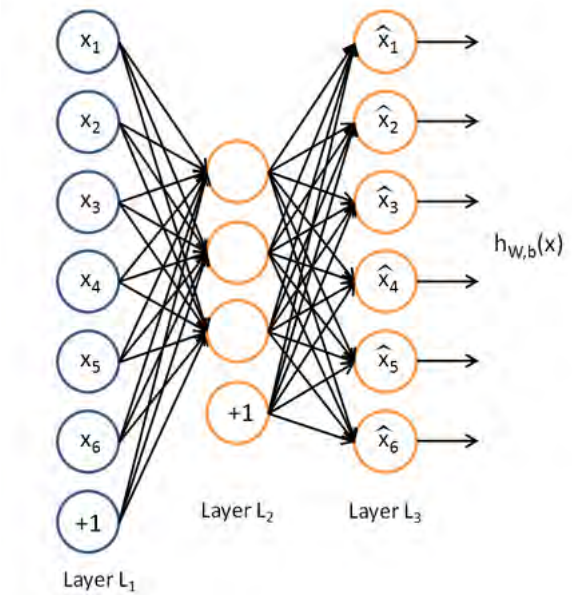
下列哪个不属于常用的文本分类的特征选择算法？（D）
A 卡方检验值
B 互信息
C 信息增益
D 主成分分析

常采用特征选择方法。常见的六种特征选择方法：
1）DF(Document Frequency) 文档频率
DF:统计特征词出现的文档数量，用来衡量某个特征词的重要性
2）MI(Mutual Information) 互信息法
互信息法用于衡量特征词与文档类别直接的信息量。
如果某个特征词的频率很低，那么互信息得分就会很大，因此互信息法倾向"低频"的特征词。
相对的词频很高的词，得分就会变低，如果这词携带了很高的信息量，互信息法就会变得低效。
3）(Information Gain) 信息增益法
通过某个特征词的缺失与存在的两种情况下，语料中前后信息的增加，衡量某个特征词的重要性。
4）CHI(Chi-square) 卡方检验法
利用了统计学中的"假设检验"的基本思想：首先假设特征词与类别直接是不相关的
如果利用CHI分布计算出的检验值偏离阈值越大，那么更有信心否定原假设，接受原假设的备则假设：特征词与类别有着很高的关联度。
5）WLLR(Weighted Log Likelihood Ration)加权对数似然
6）WFO（Weighted Frequency and Odds）加权频率和可能性
http://blog.csdn.net/ztf312/article/details/50890099

类域界面方程法中，不能求线性不可分情况下分类问题近似或精确解的方法是？(D)
A 伪逆法-径向基（R B F）神经网络的训练算法，就是解决线性不可分的情况
B 基于二次准则的H-K算法：最小均方差准则下求得权矢量，二次准则解决非线性问题
C 势函数法 - 非线性
D 感知器算法 - 线性分类算法

机器学习中做特征选择时，可能用到的方法有？ （E）
A、卡方
B、信息增益
C、平均互信息
D、期望交叉熵
E 以上都有

下列方法中，不可以用于特征降维的方法包括（E）
A 主成分分析PCA
B 线性判别分析LDA
C 深度学习SparseAutoEncoder
D 矩阵奇异值分解SVD
E 最小二乘法LeastSquares
特征降维方法主要有：
PCA，LLE，Isomap
SVD和PCA类似，也可以看成一种降维方法
LDA:线性判别分析，可用于降维
AutoEncoder：AutoEncoder的结构与神经网络的隐含层相同，由输入L1,输出 L2组成，中间则是权重连接。Autoencoder通过L2得到输入的重构L3，最小化L3与L1的差别 进行训练得到权重。在这样的权重参数下，得到的L2可以尽可能的保存L1的信息。
Autoencoder的输出L2的维度由输出的神经元个数决定。当输出维度大于L1时，则需要在训练目标函数中加入sparse 惩罚项，避免L2直接复制L1（权重全为1）。所以称为sparseAutoencoder(Andrew Ng提出的)。
结论：SparseAutoencoder大多数情况下都是升维的，所以称之为特征降维的方法不准确。



一般，k-NN最近邻方法在（ A）的情况下效果较好。
A．样本较多但典型性不好 C．样本较少但典型性好
B．样本呈团状分布 D．样本呈链状分布
下列哪些方法可以用来对高维数据进行降维:
A LASSO
B 主成分分析法
C 聚类分析
D 小波分析法
E 线性判别法

F 拉普拉斯特征映射
lasso通过参数缩减达到降维的目的；
pca就不用说了
线性鉴别法即LDA通过找到一个空间使得类内距离最小类间距离最大所以可以看做是降维；
小波分析有一些变换的操作降低其他干扰可以看做是降维
拉普拉斯请看这个http://f.dataguru.cn/thread-287243-1-1.html
以下描述正确的是（D）
A SVM是这样一个分类器，它寻找具有最小边缘的超平面，因此它也经常被称为最小边缘分类器
B 在聚类分析当中，簇内的相似性越大，簇间的差别越大，聚类的效果就越差
C 在决策树中，随着树中结点输变得太大，即使模型的训练误差还在继续降低，但是检验误差开始增大，这是出现了模型拟合不足的原因
D 聚类分析可以看作是一种非监督的分类
以下说法中错误的是（C）
A SVM对噪声（如来自其他分部的噪声样本）具备鲁棒性
B 在adaboost算法中，所有被分错样本的权重更新比例不相同
C boosting和bagging都是组合多个分类器投票的方法，二者都是根据单个分类器的正确率确定其权重
D 给定n个数据点，如果其中一半用于训练，一半用户测试，则训练误差和测试误差之间的差别会随着n的增加而减少的
A 软间隔分类器对噪声是有鲁棒性的。
B 请参考http://blog.csdn.net/v_july_v/article/details/40718799
C boosting是根据分类器正确率确定权重，bagging不是。
D 训练集变大会提高模型鲁棒性。
关于正态分布,下列说法错误的是:
A.正态分布具有集中性和对称性
B.正态分布的均值和方差能够决定正态分布的位置和形态
C.正态分布的偏度为0，峰度为1
D.标准正态分布的均值为0，方差为1
答案 C，标准正态分布即如此。
在以下不同的场景中,使用的分析方法不正确的有
A.根据商家最近一年的经营及服务数据,用聚类算法判断出天猫商家在各自主营类目下所属的商家层级
B.根据商家近几年的成交数据,用聚类算法拟合出用户未来一个月可能的消费金额公式
C.用关联规则算法分析出购买了汽车坐垫的买家,是否适合推荐汽车脚垫
D.根据用户最近购买的商品信息,用决策树算法识别出淘宝买家可能是男还是女
什么是梯度爆炸?
误差梯度是神经网络训练过程中计算的方向和数量，用于以正确的方向和合适的量更新网络权重。
在深层网络或循环神经网络中，误差梯度可在更新中累积，变成非常大的梯度，然后导致网络权重的大幅更新，并因此使网络变得不稳定。在极端情况下，权重的值变得非常大，以至于溢出，导致 NaN 值。
网络层之间的梯度（值大于 1.0）重复相乘导致的指数级增长会产生梯度爆炸。
梯度爆炸会引发什么问题?
在深度多层感知机网络中，梯度爆炸会引起网络不稳定，最好的结果是无法从训练数据中学习，而最坏的结果是出现无法再更新的 NaN 权重值。
梯度爆炸导致学习过程不稳定。—《深度学习》，2016.
在循环神经网络中，梯度爆炸会导致网络不稳定，无法利用训练数据学习，最好的结果是网络无法学习长的输入序列数据。
如何确定是否出现梯度爆炸?
训练过程中出现梯度爆炸会伴随一些细微的信号，如：
模型无法从训练数据中获得更新（如低损失）。
模型不稳定，导致更新过程中的损失出现显著变化。
训练过程中，模型损失变成 NaN。
如果你发现这些问题，那么你需要仔细查看是否出现梯度爆炸问题。
以下是一些稍微明显一点的信号，有助于确认是否出现梯度爆炸问题。
训练过程中模型梯度快速变大。
训练过程中模型权重变成 NaN 值。
训练过程中，每个节点和层的误差梯度值持续超过 1.0。
如何修复梯度爆炸问题?
有很多方法可以解决梯度爆炸问题，本节列举了一些最佳实验方法。
1. 重新设计网络模型
在深度神经网络中，梯度爆炸可以通过重新设计层数更少的网络来解决。
使用更小的批尺寸对网络训练也有好处。
在循环神经网络中，训练过程中在更少的先前时间步上进行更新（沿时间的截断反向传播，truncated Backpropagation through time）可以缓解梯度爆炸问题。
2. 使用 ReLU 激活函数
在深度多层感知机神经网络中，梯度爆炸的发生可能是因为激活函数，如之前很流行的 Sigmoid 和 Tanh 函数。
使用 **ReLU** 激活函数可以减少梯度爆炸。采用 ReLU 激活函数是最适合隐藏层的新实践。
3. 使用长短期记忆网络
在循环神经网络中，梯度爆炸的发生可能是因为某种网络的训练本身就存在不稳定性，如随时间的反向传播本质上将循环网络转换成深度多层感知机神经网络。
使用长短期记忆（LSTM）单元和相关的门类型神经元结构可以减少梯度爆炸问题。
采用 **LSTM** 单元是适合循环神经网络的序列预测的最新最好实践。
4. 使用梯度截断（Gradient Clipping）
在非常深且批尺寸较大的多层感知机网络和输入序列较长的 LSTM 中，仍然有可能出现梯度爆炸。如果梯度爆炸仍然出现，你可以在训练过程中检查和限制梯度的大小。这就是梯度截断。
处理梯度爆炸有一个简单有效的解决方案：如果梯度超过阈值，就截断它们。
——《Neural Network Methods in Natural Language Processing》，2017.
具体来说，检查误差梯度的值是否超过阈值，如果超过，则截断梯度，将梯度设置为阈值。
梯度截断可以一定程度上缓解梯度爆炸问题（梯度截断，即在执行梯度下降步骤之前将梯度设置为阈值）。
——《深度学习》，2016.
在 Keras 深度学习库中，你可以在训练之前设置优化器上的 clipnorm 或 clipvalue 参数，来使用梯度截断。

默认值为 clipnorm=1.0 、clipvalue=0.5。详见：https://keras.io/optimizers/。
5. 使用权重正则化（Weight Regularization）
如果梯度爆炸仍然存在，可以尝试另一种方法，即检查网络权重的大小，并惩罚产生较大权重值的损失函数。该过程被称为权重正则化，通常使用的是 L1 惩罚项（权重绝对值）或 L2 惩罚项（权重平方）。
对循环权重使用 L1 或 L2 惩罚项有助于缓解梯度爆炸。
——On the difficulty of training recurrent neural networks，2013.
在 Keras 深度学习库中，你可以通过在层上设置 kernel_regularizer 参数和使用 L1 或 L2 正则化项进行权重正则化。
LSTM神经网络输入输出究竟是怎样的?
@YJango，本题解析来源：https://www.zhihu.com/question/41949741
2017年1月4日文章 **Recurrent Layers——介绍**

- 第一要明确的是神经网络所处理的单位全部都是：向量

下面就解释为什么你会看到训练数据会是矩阵和张量

- 常规feedforward 输入和输出：矩阵

  输入矩阵形状：(n_samples, dim_input)
  输出矩阵形状：(n_samples, dim_output)

注：真正测试/训练的时候，网络的输入和输出就是向量而已。加入n_samples这个维度是为了可以实现一次训练多个样本，求出平均梯度来更新权重，这个叫做Mini-batch gradient descent。 如果n_samples等于1，那么这种更新方式叫做Stochastic Gradient Descent (SGD)。
Feedforward 的输入输出的本质都是单个向量。

- 常规Recurrent (RNN/LSTM/GRU) 输入和输出：张量

  输入张量形状：(time_steps, n_samples，dim_input)
  输出张量形状：(time_steps, n_samples，dim_output)

注：同样是保留了Mini-batch gradient descent的训练方式，但不同之处在于多了time step这个维度。
Recurrent 的任意时刻的输入的本质还是单个向量，只不过是将不同时刻的向量按顺序输入网络。所以你可能更愿意理解为一串向量 a sequence of vectors，或者是矩阵。

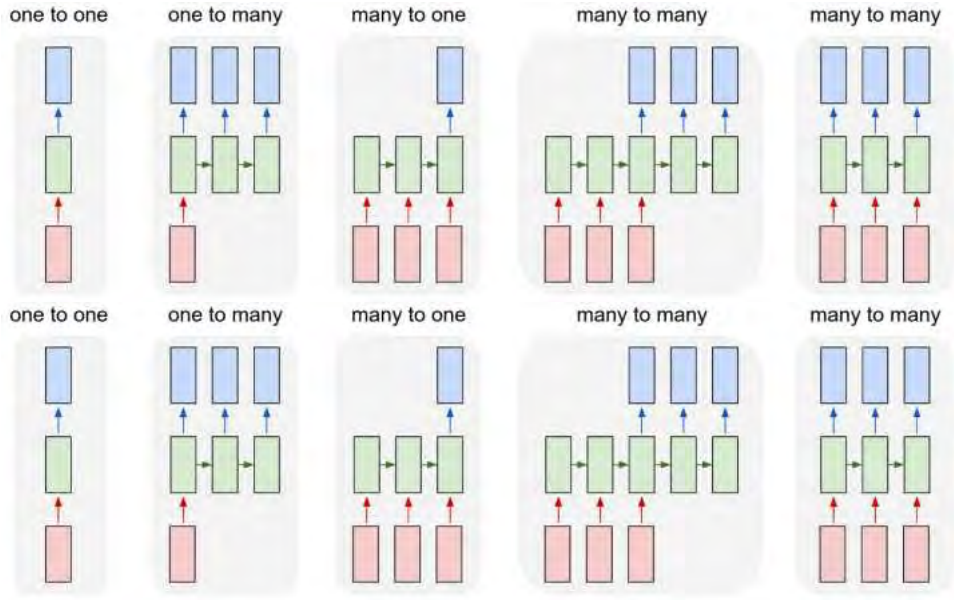
python代码表示预测的话：

```
import numpy as np

#当前所累积的hidden_state,若是最初的vector，则hidden_state全为0
hidden_state=np.zeros((n_samples, dim_input))

#print(inputs.shape)：  (time_steps, n_samples,  dim_input)
outputs = np.zeros((time_steps, n_samples, dim_output))

for i in range(time_steps):
    #输出当前时刻的output，同时更新当前已累积的hidden_state
    outputs[i],hidden_state = RNN.predict(inputs[i],hidden_state)
#print(outputs.shape)：  (time_steps, n_samples, dim_output)
```

但需要注意的是，Recurrent nets的输出也可以是矩阵，而非三维张量，取决于你如何设计。

1. 若想用一串序列去预测另一串序列，那么输入输出都是张量 (例如语音识别 或机器翻译 一个中文句子翻译成英文句子（一个单词算作一个向量），机器翻译还是个特例，因为两个序列的长短可能不同，要用到seq2seq；
2. 若想用一串序列去预测一个值，那么输入是张量，输出是矩阵 （例如，情感分析就是用一串单词组成的句子去预测说话人的心情）

Feedforward 能做的是向量对向量的one-to-one mapping，
Recurrent 将其扩展到了序列对序列 sequence-to-sequence mapping.
但单个向量也可以视为长度为1的序列。所以有下图几种类型：



除了最左侧的one to one是feedforward 能做的，右侧都是Recurrent所扩展的

若还想知道更多

- 可以将Recurrent的横向操作视为累积已发生的事情，并且LSTM的memory cell机制会选择记忆或者忘记所累积的信息来预测某个时刻的输出。
- 以概率的视角理解的话：就是不断的conditioning on已发生的事情，以此不断缩小sample space
- RNN的思想是: current output不仅仅取决于current input，还取决于previous state；可以理解成current output是由current input和previous hidden state两个输入计算而出的。并且每次计算后都会有信息残留于previous hidden state中供下一次计算

以下关于PMF(概率质量函数),PDF(概率密度函数),CDF(累积分布函数)描述错误的是?
A.PDF描述的是连续型随机变量在特定取值区间的概率
B.CDF是PDF在特定区间上的积分
C.PMF描述的是离散型随机变量在特定取值点的概率
D.有一个分布的CDF函数H(x),则H(a)等于P(X<=a)

正确答案：A
解析：
概率质量函数 (probability mass function，PMF)是离散随机变量在各特定取值上的概率。
概率密度函数（p robability density function，PDF ）是对 连续随机变量 定义的，本身不是概率，只有对连续随机变量的取值进行积分后才是概率。
累积分布函数（cumulative distribution function，CDF） 能完整描述一个实数随机变量X的概率分布，是概率密度函数的积分。对于所有实数x 与
pdf相对。线性回归的基本假设有哪些？(ABDE)
A.随机误差项是一个期望值为0的随机变量；
B.对于解释变量的所有观测值，随机误差项有相同的方差；
C.随机误差项彼此相关；
D.解释变量是确定性变量不是随机变量，与随机误差项之间相互独立；
E.随机误差项服从正态分布处理类别型特征时，事先不知道分类变量在测试集中的分布。要将 one-hot encoding（独热码）应用到类别型特征中。那么在训练集中将独热码应用到分类变量可能要面临的困难是什么?
A. 分类变量所有的类别没有全部出现在测试集中
B. 类别的频率分布在训练集和测试集是不同的
C. 训练集和测试集通常会有一样的分布
答案为：A、B ，如果类别在测试集中出现，但没有在训练集中出现，独热码将不能进行类别编码，这是主要困难。如果训练集和测试集的频率分布不相同，我们需要多加小心。假定你在神经网络中的隐藏层中使用激活函数 X。在特定神经元给定任意输入，你会得到输出「-0.0001」。X 可能是以下哪一个激活函数?
A. ReLU
B. tanh
C. SIGMOID
D. 以上都不是
答案为：B，该激活函数可能是 tanh，因为该函数的取值范围是 (-1,1)。
下面哪些对「类型 1（Type-1）」和「类型 2（Type-2）」错误的描述是正确的?
A. 类型 1 通常称之为假正类，类型 2 通常称之为假负类。
B. 类型 2 通常称之为假正类，类型 1 通常称之为假负类。
C. 类型 1 错误通常在其是正确的情况下拒绝假设而出现。
答案为(A)和(C)：在统计学假设测试中，I 类错误即错误地拒绝了正确的假设即假正类错误，II 类错误通常指错误地接受了错误的假设即假负类错误。
在下面的图像中，哪一个是多元共线（multi-collinear）特征?
A. 图 1 中的特征
B. 图 2 中的特征
C. 图 3 中的特征
D. 图 1、2 中的特征
E. 图 2、3 中的特征
F. 图 1、3 中的特征
答案为（D）：在图 1 中，特征之间有高度正相关，图 2 中特征有高度负相关。所以这两个图的特征是多元共线特征。
鉴别了多元共线特征。那么下一步可能的操作是什么?
A. 移除两个共线变量B. 不移除两个变量，而是移除一个
C. 移除相关变量可能会导致信息损失，可以使用带罚项的回归模型（如 ridge 或 lasso regression）。
答案为（B）和（C）：因为移除两个变量会损失一切信息，所以我们只能移除一个特征，或者也可以使用正则化算法（如 L1 和 L2）
给线性回归模型添加一个不重要的特征可能会造成?
A. 增加 R-square
B. 减少 R-square
答案为（A）：在给特征空间添加了一个特征后，不论特征是重要还是不重要，R-square 通常会增加。
假定目标变量的类别非常不平衡，即主要类别占据了训练数据的 99%。现在你的模型在测试集上表现为 99% 的准确度。那么下面哪一项表述是正确的?
A. 准确度并不适合于衡量不平衡类别问题
B. 准确度适合于衡量不平衡类别问题
C. 精确率和召回率适合于衡量不平衡类别问题
D. 精确率和召回率不适合于衡量不平衡类别问题
答案为（A）和（C）
什么是偏差与方差?
泛化误差可以分解成偏差的平方加上方差加上噪声。偏差度量了学习算法的期望预测和真实结果的偏离程度，刻画了学习算法本身的拟合能力，方差度量了同样大小的训练集的变动所导致的学习性能的变化，刻画了数据扰动所造成的影响，噪声表达了当前任务上任何学习算法所能达到的期望泛化误差下界，刻画了问题本身的难度。偏差和方差一般称为bias和variance，一般训练程度越强，偏差越小，方差越大，泛化误差一般在中间有一个最小值，如果偏差较大，方差较小，此时一般称为欠拟合，而偏差较小，方差较大称为过拟合。偏差：$E_D[(\overline{f(x)}-y)^2]$ 方差：$E_D[(f(x,D)-\overline{f(x)})^2]$
解决bias和Variance问题的方法是什么？交叉验证
High bias解决方案:Boosting、复杂模型（非线性模型、增加神经网络中的层）、更多特征
High Variance解决方案：agging、简化模型、降维
采用 EM 算法求解的模型有哪些，为什么不用牛顿法或梯度下降法?
用EM算法求解的模型一般有GMM或者协同过滤，k-means其实也属于EM。EM算法一定会收敛，但是可能收敛到局部最优。由于求和的项数将随着隐变量的数目指数上升，会给梯度计算带来麻烦。

xgboost怎么给特征评分？在训练的过程中，通过Gini指数选择分离点的特征，一个特征被选中的次数越多，那么该特征评分越高。
[python] # feature importance
print(model.feature_importances_)
# plot pyplot.bar(range(len(model.feature_importances_)), model.feature_importances_)
pyplot.show() ==========
# plot feature importance
plot_importance(model)
pyplot.show()
什么是OOB？随机森林中OOB是如何计算的，它有什么优缺点?
bagging方法中Bootstrap每次约有1/3的样本不会出现在Bootstrap所采集的样本集合中，当然也就没有参加决策树的建立，把这1/3的数据称为袋外数据oob（out of bag），它可以用于取代测试集误差估计方法。
袋外数据(oob)误差的计算方法如下：
对于已经生成的随机森林,用袋外数据测试其性能,假设袋外数据总数为O,用这O个袋外数据作为输入,带进之前已经生成的随机森林分类器,分类器会给出O个数据相应的分类,因为这O条数据的类型是已知的,则用正确的分类与随机森林分类器的结果进行比较,统计随机森林分类器分类错误的数目,设为X,则袋外数据误差大小=X/O;这已经经过证明是无偏估计的,所以在随机森林算法中不需要再进行交叉验证或者单独的测试集来获取测试集误差的无偏估计。
假设张三的mp3里有1000首歌，现在希望设计一种随机算法来随机播放。与普通随机模式不同的是，张三希望每首歌被随机到的概率是与一首歌的豆瓣评分（0~10分）成正比的，如朴树的《平凡之路》评分为8.9分，逃跑计划的《夜空中最亮的星》评分为9.5分，则希望听《平凡之路》的概率与《夜空中最亮的星》的概率比为89:95。现在我们已知这1000首歌的豆瓣评分：（1）请设计一种随机算法来满足张三的需求。（2）写代码实现自己的算法。

```
#include <iostream>
#include <time.h>
#include <stdlib.h>
using namespace std;

int findIdx(double songs[],int n,double rnd){
    int left=0;
    int right=n-1;
    int mid;
    while(left<=right){
        mid=(left+right)/2;
        if((songs[mid-1]<=rnd) && (songs[mid]>=rnd))
            return mid;
        if(songs[mid]>rnd)
            right=mid-1;
        else
            left=mid+1;
    }
//    return mid;
}

int randomPlaySong(double sum_scores[],int n){
    double mx=sum_scores[n-1];
    double rnd= rand()*mx/(double)(RAND_MAX);
    return findIdx(sum_scores,n,rnd);
}

int main()
{
    srand(time(0));
    double scores[]={5.5,6.5,4.5,8.5,9.5,7.5,3.5,5.0,8.0,2.0};
    int n=sizeof(scores)/sizeof(scores[0]);
    double sum_scores[n];
    sum_scores[0]=scores[0];

    for(int i=1;i<n;i++)
        sum_scores[i]=sum_scores[i-1]+scores[i];

    cout<<"Calculate the probability of each song: "<<endl;
    int totalScore=sum_scores[n-1];
    for(int i=0;i<n;i++)
        cout<<scores[i]/totalScore<<" ";
    cout<<endl;

    int counts[n];
    for(int i=0;i<n;i++)
        counts[i]=0;

    int i=0;
    int idx;
    int MAX_ITER=100000000;
    while(i<MAX_ITER){
        idx=randomPlaySong(sum_scores,n);
        counts[idx]++;
```

```
        i++;
    }

    cout<<"After simulation, probability of each song: "<<endl;
    for(int i=0;i<n;i++)
        cout<<1.0*counts[i]/MAX_ITER<<" ";
    cout<<endl;

    return 0;
}
```

对于logistic regession问题：prob（t|x）=1/（1+exp（w*x+b））且label y=0或1，请给出loss function和权重w的更新公式及推导。
Logistic regression 的loss function 是log loss, 公式表达为：

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

其中，$h_\theta(x)=P(Y=1|x)$

$$\min L(w,b)=\frac{1}{N}\sum_{i=1}^{N}L(h_\theta(x),\ y)=-\frac{1}{N}\left[\sum_{i=1}^{N}y_i\log(h_\theta(x_i))+(1-y_i)\log(1-h_\theta(x_i))\right]$$

w的更新公式可以由最小化loss function得到，即：
其中大括号里面的部分，等价于逻辑回归模型的对数似然函数，所以也可以用极大似然函数方法求解，根据梯度下降法，其更新公式为：

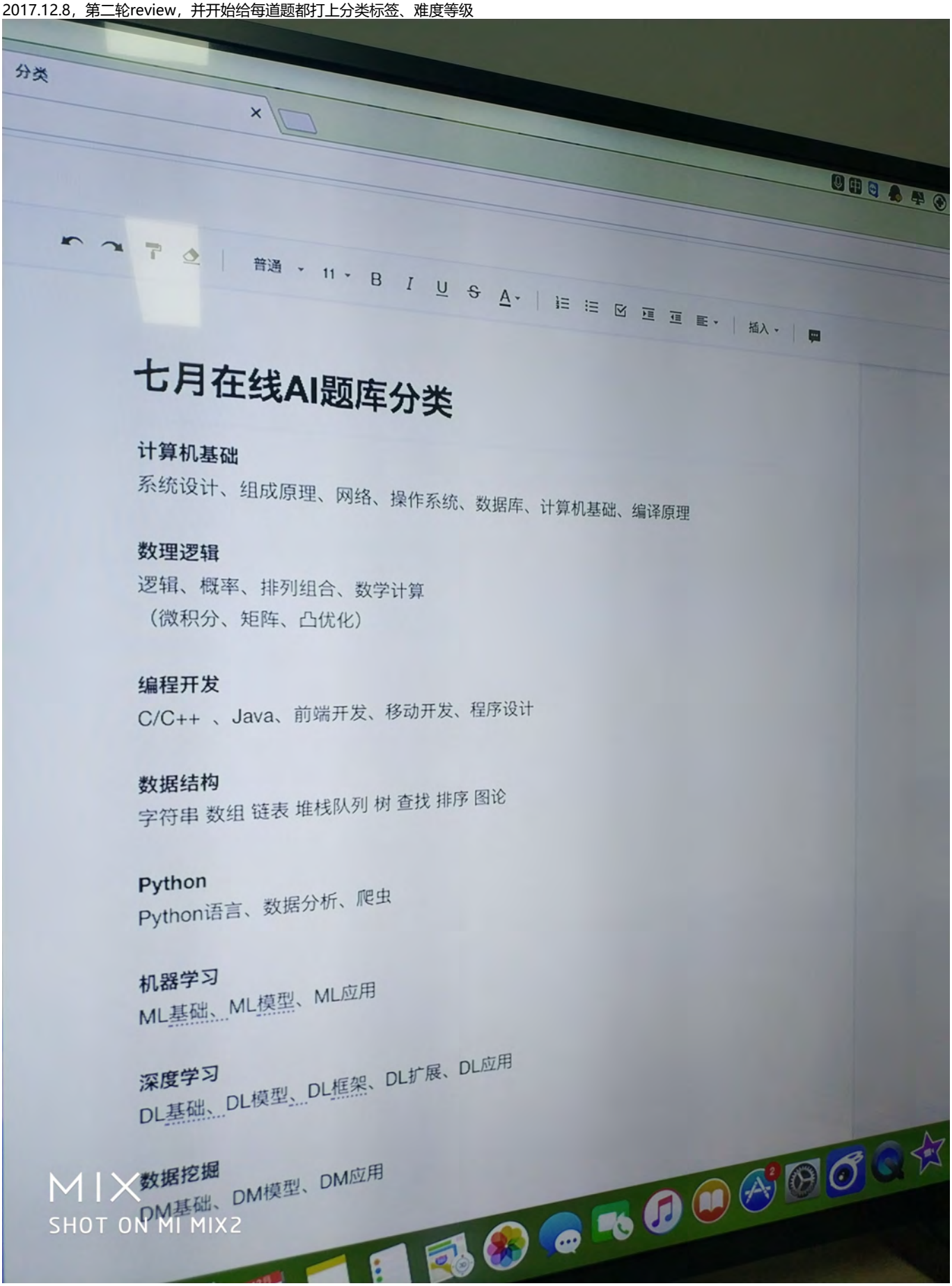
$$w_{j+1}=w_j+\lambda\frac{\partial L}{\partial w}=w_j+\lambda\sum_{i=1}^{N}(y_i-h_\theta(x_i))x_i$$

决策树的父节点和子节点的熵的大小关系是什么？
A. 决策树的父节点更大
B. 子节点的熵更大
C. 两者相等
D. 根据具体情况而定
正确答案：B。在特征选择时，应该给父节点信息增益最大的节点，而信息增益的计算为 IG(Y|X) = H(Y) - H(Y/X)，H(Y/X) 为该特征节点的条件熵，H(Y/X) 越小，即该特征节点的属性对整体的信息表示越“单纯”，IG更大。 则该属性可以更好的分类。H(Y/X) 越大，属性越“紊乱”，IG越小，不适合作为分类属性。
欠拟合和过拟合的原因分别有哪些？如何避免？
欠拟合的原因：模型复杂度过低，不能很好的拟合所有的数据，训练误差大；
避免欠拟合：增加模型复杂度，如采用高阶模型（预测）或者引入更多特征（分类）等。
过拟合的原因：模型复杂度过高，训练数据过少，训练误差小，测试误差大；
避免过拟合：降低模型复杂度，如加上正则惩罚项，如L1，L2，增加训练数据等。
语言模型的参数估计经常使用MLE（最大似然估计）。面临的一个问题是没有出现的项概率为0，这样会导致语言模型的效果不好。为了解决这个问题，需要使用（A）
A. 平滑
B. 去噪
C. 随机插值
D. 增加白噪音

勘误记

- 2017.12.2，七月在线讲师团队开始复审review全部答案和解析，因为这些题要上线七月在线官网和APP，面对几十万人甚至上百万人用，所以我们需要每道题都有答案和解析，且保证答案和解析的精准。分工如下：1~20 AntZ，21~40 褚博士，41~60 梁伟祺，61~80 管博士，81~100 寒小阳，101~120 赵博士，121~140 张雨石，141~160 王赟，161~180 梁伟祺，181~200 AntZ。

- 2017.12.8，第二轮review，并开始给每道题都打上分类标签、难度等级



- 2017.12.9~12.11，第三轮review，并和运营团队开始一道题一道题的录入官网和APP后台系统，且已于双十二当天上线官网和Android APP。
- 2017.12.24，BAT机器学习面试1000题系列，已经整到300多题，加上「七月在线」官网和Android上已有的题，整个AI题库已有数千道。很赞把题库产品化，不断加题。
- 重要说明：自1.8日iOS亦上线题库之后，本文暂停更新和维护，新题都已更新到七月在线APP或七月在线官网题库板块上。

后记

实话说，与整理数据结构/算法类的笔试面试题不同，整理机器学习笔试面试题的难度陡然剧增，因为这类题在网上少之又少，整理一道ML题的难度相当于整理至少10道数据结构/算法题的难度。

但好的是，在整理这个系列的过程中，我们也学到了很多，是一个边整理边学习的过程，很多问题都是在这整理中一点一点明白，包括各类最优化算法、包括RNN等等。在整理的过程中看到一个问题后，会有意无意去深挖，且不断问自己与之相关的问题，就这样通过一个一个问题不断思考，对自己更是一个学习和进步。

且让我们做下去，直到1000题，甚至数千道题的理由只有一个：利于众人、价值长远。

最后，欢迎正在看本文的你，或针对题目的答案留言提出更好意见，或分享你手头上已有的问题（你可以直接在本文评论下留言，也欢迎通过微博私信：@研究者July），共同分享帮助全球更多人，thanks。

July团队、不写日期了，新题请移步七月在线APP或七月在线官网。