



17



目录



收藏



评论



微信



微博



QQ

SDN

首页

博客

学院

下载

GitChat

TinyMind

论坛

问答

商城

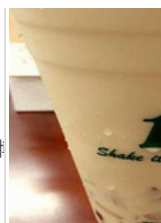
...

搜博主文章



写

woaidapaopao的博客



1点点加盟要



联系我们



关于 招聘 /

©2018 CSDN版权

百度提供支持

经营性网站备案信

网络110报警服务

中国互联网举报中

北京互联网违法和

阅读数

面试笔试整理3：深度学习机器学习面试问题准备（必会）

2017年09月07日 01:13:54

第一部分：深度学习

1、神经网络基础问题

（1）Backpropagation（要能推倒）

后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**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：

$$J(W, b) = \left[\frac{1}{m} \sum_{i=1}^m J(W, b; x^{(i)}, y^{(i)}) \right] + \frac{\lambda}{2} \sum_{l=1}^{n_l-1} \sum_{i=1}^{s_l} \sum_{j=1}^{s_{l+1}} (W_{ji}^{(l)})^2$$

$$= \left[\frac{1}{m} \sum_{i=1}^m \left(\frac{1}{2} \|h_{W,b}(x^{(i)}) - y^{(i)}\|^2 \right) \right] + \frac{\lambda}{2} \sum_{l=1}^{n_l-1} \sum_{i=1}^{s_l} \sum_{j=1}^{s_{l+1}} (W_{ji}^{(l)})^2$$

- 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

$$\delta_i^{(l)} = \left(\sum_{j=1}^{s_{l+1}} W_{ji}^{(l)} \delta_j^{(l+1)} \right) f'(z_i^{(l)})$$

$$\frac{\partial}{\partial W_{ij}^{(l)}} J(W, b; x, y) = a_j^{(l)} \delta_i^{(l+1)}$$

$$\frac{\partial}{\partial b_i^{(l)}} J(W, b; x, y) = \delta_i^{(l+1)}.$$

（2）梯度消失、梯度爆炸

梯度消失：这本质上是由于激活函数的选择导致的，最简单的sigmoid函数为例，在函数的两端梯度求导结果非常小（饱和区），导致后向传播过程中由于多次用到激活函数的导数值使得整体的乘积梯度结果变得越来越小，也就出现了梯度消失的现象。

梯度爆炸：同理，出现在激活函数处在激活区，而且权重W过大的情况下。但是梯度爆炸不如梯度消失出现的机会多。

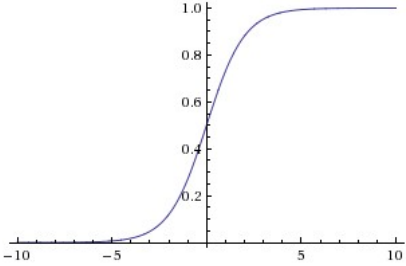
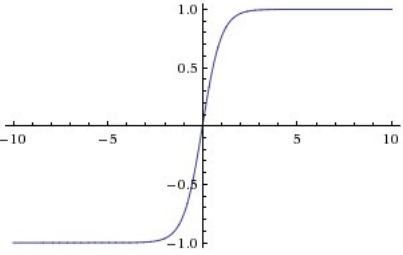
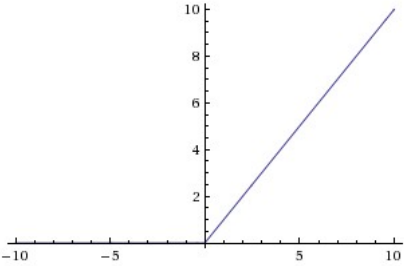

（3）常用的激活函数

加入CSDN，享受更精准的内容推荐，与500万程序员共同成长！

登录

注册

X

	激活函数	公式	缺点	优点
17	目录 收藏 评论 微信 微博 QQ	<div>Sigmoid</div> <div>$\sigma(x) = 1/(1 + e^{-x})$</div> <div></div>	1、会有梯度弥散 2、不是关于原点对称 3、计算exp比较耗时	-
		<div>Tanh</div> <div>$\tanh(x) = 2\sigma(2x) - 1$</div> <div></div>	梯度弥散没解决	1、解决了原点对称问题 2、比sigmoid更快
		<div>ReLU</div> <div>$f(x) = \max(0, x)$</div> <div></div>	梯度弥散没完全解决，在（-）部分相当于神经元死亡而且不会复活	1、解决了部分梯度弥散问题 2、收敛速度更快
		<div>Leaky ReLU</div> <div>$f(x) = 1(x < 0)(\alpha x) + 1(x \geq 0)(x)$</div>	-	解决了神经死亡问题
		<div>Maxout</div> <div>$\max(w_1^T x + b_1, w_2^T x + b_2)$</div> <div></div> <div>Piecewise linear function</div>	参数比较多,本质上是在输出结果上又增加了一层	克服了ReLU的缺点，比较提倡使用

(4) 参数更新方法

方法名称	公式	
Vanilla update	$x += - learning_rate * dx$	

	方法名称	公式	
17	Nesterov Momentum	$x_ahead = x + \mu * v$ $v = \mu * v - learning_rate * dx_ahead$ $x += v$	
目录	Adagrad (自适应的方法，梯度大的方向学习率越来越小,由快到慢)	$cache += dx**2$ $x += - learning_rate * dx / (np.sqrt(cache) + eps)$	
收藏	Adam	$m = \beta_1 * m + (1 - \beta_1) dx$ $v = \beta_2 * v + (1 - \beta_2) (dx**2)$ $x += - learning_rate * m / (np.sqrt(v) + eps)$	
评论			
微信	(5) 解决overfitting的方法		
	dropout , regularization , batch normalizatin ,但是要注意dropout只在训练的时候用，让一部分神经元随机失活。		
微博	Batch normalization是为了让输出都是单位高斯激活，方法是在连接和激活函数之间加入BatchNorm层，计算每个特征的均值和方差进行正则化。		
QQ	2、CNN问题		

(1) 思想

改变全连接为局部连接，这是由于图片的特殊性造成的（ 图像的一部分的统计特性与其他部分是一样的 ），通过局部连接和参数共享大范围的减少参数值。可以通过使用多个filter来提取图片的不同特征（ 多卷积核 ）。

(2) filter尺寸的选择

通常尺寸多为奇数（ 1 , 3 , 5 , 7 ）

(3) 输出尺寸计算公式

输出尺寸=(N - F +padding*2)/stride + 1
步长可以自由选择通过补零的方式来实现连接。

(4) pooling池化的作用

虽然通过 卷积的方式可以大范围的减少输出尺寸（ 特征数 ），但是依然很难计算而且很容易过拟合，所以依然利用图片的静态特性通过池化的方式进一步减少尺寸。

(5) 常用的几个模型，这个最好能记住模型大致的尺寸参数。

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 我称为盗梦空间网络	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

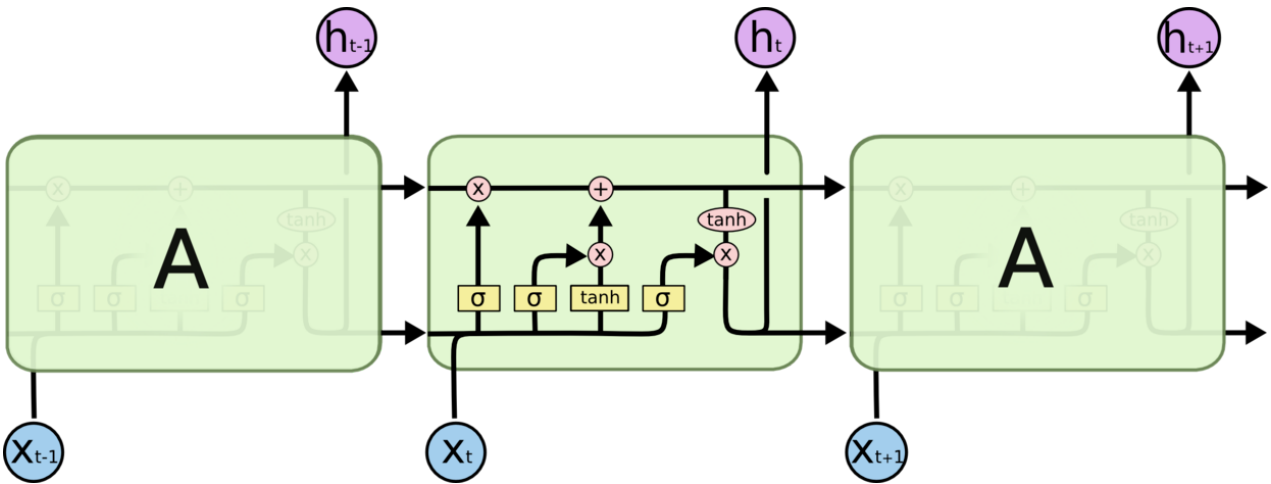
2、RNN

1、RNN原理：

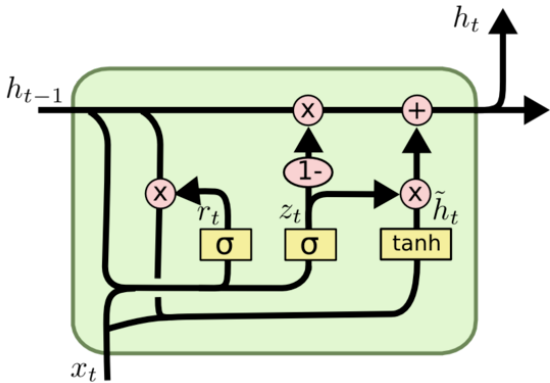
在普通的全连接网络或CNN中，每层神经元的信号只能向上一层传播，样本的处理在各个时刻独立，因此又被成为前向神经网络(Feed-forward+Neural+Networks)。而在RNN中，神经元的输出可以在下一个时间戳直接作用到自身，即第i层神经元在m时刻的输入，除了（ i - 1 ）层神经元在该时刻的输出外，还包括其自身在（ m - 1 ）时刻的输出。所以叫循环神经网络

2、RNN、LSTM、GRU区别

- 17
- 目录
- 收藏
- 评论
- 微信
- 微博
- QQ
- LSTM：因为LSTM有进有出且当前的cell information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。推导forget gate, input gate, cell state, hidden information等因为LSTM有进有出且当前的cell information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。



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

3、LSTM防止梯度弥散和爆炸

LSTM用加和的方式取代了乘积，使得很难出现梯度弥散。但是相应的更大的几率会出现梯度爆炸，但是可以通过给梯度加门限解决这一问题。

4、引出word2vec

这个也就是Word Embedding，是一种高效的从原始语料中学习字词空间向量的预测模型。分为CBOW(Continuous Bag of Words)和Skip-Gram两种形式。其中CBOW是从原始语句推测目标词汇，而Skip-Gram相反。CBOW可以用于小语料库，Skip-Gram用于大语料库。具体的就不是很会了。

3、GAN

1、GAN的思想

GAN结合了生成模型和判别模型，相当于矛与盾的撞击。生成模型负责生成最好的数据骗过判别模型，而判别模型负责识别出哪些是真的哪些是生成模型生成的。但是这些只是在了解了GAN之后才会体到的，但是为什么这样会有效呢？

假设我们有分布 $P_{data}(x)$ ，我们希望能建立一个生成模型来模拟真实的数据分布，假设生成模型为 $P_g(x;\theta)$ ，我们的目的是求解 θ 的值，通常我们都是用最大似然估计。但是现在的问题是由于我们想用NN来模拟 $P_{data}(x)$ ，但是我们很难求解似然函数，因为我们没办法写出生成模型的具体表达形式，于是才有了GAN，也就是用判别模型来代替求解最大似然的过程。

在最理想的状态下，G可以生成足以“以假乱真”的图片 $G(z)$ 。对于D来说，它难以判定G生成的图片究竟是不是真实的，因此 $D(G(z)) = 0.5$ 。这样我们的目的就达成了：我们得到了一个生成式的模型G，它可以用来生成图片。

2、GAN的表达式

$$\min_G \max_D V(D, G) = \mathbb{E}_{x \sim p_{data}(x)} [\log D(x)] + \mathbb{E}_{z \sim p_z(z)} [\log(1 - D(G(z)))].$$

3、GAN的实际计算方法

因为我们不可能有 $P_{data}(x)$ 的分布，所以我们实际中都是用采样的方式来计算差异（也就是积分变求和）。具体实现过程如下：

Algorithm 1 Minibatch stochastic gradient descent training of generative adversarial nets. The number of steps to apply to the discriminator, k , is a hyperparameter. We used $k = 1$, the least expensive option, in our experiments.

for number of training iterations **do**

for k steps **do**

- Sample minibatch of m noise samples $\{z^{(1)}, \dots, z^{(m)}\}$ from noise prior $p_g(z)$.
- Sample minibatch of m examples $\{x^{(1)}, \dots, x^{(m)}\}$ from data generating distribution $p_{data}(x)$.
- Update the discriminator by **ascending** its stochastic gradient:

$$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^m [\log D(x^{(i)}) + \log (1 - D(G(z^{(i)})))]$$

end for

- Sample minibatch of m noise samples $\{z^{(1)}, \dots, z^{(m)}\}$ from noise prior $p_g(z)$.
- Update the generator by **descending** its stochastic gradient:

$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^m \log (1 - D(G(z^{(i)})))$$

end for

The gradient-based updates can use any standard gradient-based learning rule. We used momentum in our experiments.

有几个关键点：判别方程训练 K 次，而生成模型只需要每次迭代训练一次，先最大化（梯度上升）再最小化（梯度下降）。

但是实际计算时 V 的后面一项在 $D(x)$ 很小的情况下由于 \log 函数的原因会导致更新很慢，所以实际中通常将后一项的 $\log(1-D(x))$ 变为 $-\log D(x)$ 。

实际计算的时候还发现不论生成器设计的好多，判别器总是能判断出真假，也就是loss几乎都是0，这可能是因为抽样造成的，生成数据与真实数据的交集过小，无论生成模型多好，判别模型也能分辨出来。解决方法有两个：1、用WGAN 2、引入随时间减少的噪声

4、对GAN有一些改进有引入f-divergence，取代Jensen-Shannon divergence，还有很多，这里主要介绍WGAN

5、WGAN

上面说过了用f-divergence来衡量两个分布的差异，而WGAN的思路是使用Earth Mover distance (挖掘机距离 Wasserstein distance)。

第二部分、机器学习准备

1、决策树相关问题

(1) 各种熵的计算

熵、联合熵、条件熵、交叉熵、KL散度（相对熵）

- 熵用于衡量不确定性，所以均分的时候熵最大
- KL散度用于度量两个分布的不相似性， $KL(p||q)$ 等于交叉熵 $H(p,q)$ -熵 $H(p)$ 。交叉熵可以看成是用 q 编码 p 所需的bit数，减去 p 本身需要的bit数，KL散度相当于用 q 编码 p 需要的额外bits。
- 交互信息Mutual information : $I(x,y) = H(x)-H(x|y) = H(y)-H(y|x)$ 表示观察到 x 后， y 的熵会减少多少。

(2) 常用的树搭建方法：ID3、C4.5、CART

上述几种树分别利用信息增益、信息增益率、Gini指数作为数据分割标准。

- 其中信息增益衡量按照某个特征分割前后熵的减少程度，其实就是上面说的交互信息。

(1) 计算数据集 D 的经验熵 $H(D)$

$$H(D) = - \sum_{k=1}^K \frac{|C_k|}{|D|} \log_2 \frac{|C_k|}{|D|}$$

(2) 计算特征 A 对数据集 D 的经验条件熵 $H(D|A)$

$$H(D|A) = \sum_{i=1}^n \frac{|D_i|}{|D|} H(D_i) = - \sum_{i=1}^n \frac{|D_i|}{|D|} \sum_{k=1}^K \frac{|D_{ik}|}{|D_i|} \log_2 \frac{|D_{ik}|}{|D_i|}$$

(3) 计算信息增益

$$g(D,A) = H(D) - H(D|A)$$

17

目录

收藏

评论

微信

微博

QQ

$$SplitInformation(D, A) = - \sum_{i=1}^n \frac{|D_i|}{|D|} \log \frac{|D_i|}{|D|}$$

$$GainRatio(D, A) = \frac{g(D, A)}{SplitInformation(D, A)}$$

- CART树在分类过程中使用的基尼指数Gini，只能用于切分二叉树，而且和ID3、C4.5树不同，Cart树不会在每一个步骤删除所用特征。

$$Gini(D) = 1 - \sum_{i=0}^n \left(\frac{D_i}{D}\right)^2$$

$$Gini(D | A) = \sum_{i=0}^n \frac{D_i}{D} Gini(D_i)$$

（3）防止过拟合：剪枝

剪枝分为前剪枝和后剪枝，前剪枝本质就是早停止，后剪枝通常是通过衡量剪枝后损失函数变化来决定是否剪枝。后剪枝有：错误率降低剪枝、悲观剪枝、代价复杂度剪枝

（4）前剪枝的几种停止条件

- 节点中样本为同一类
- 特征不足返回多类
- 如果某个分支没有值则返回父节点中的多类
- 样本个数小于阈值返回多类

2、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，局部最优优化问题时可以一定程度上抑制局部最优解的发生。

— 加入CSDN，享受更精准的内容推荐，与500万程序员共同成长！

加入CSDN，享受更精准的内容推荐，与500万程序员共同成长！

登录

注册

17

目录

收藏

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）。

3、SVM相关问题

评论

微信

微博

QQ

(1) 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

(2) RBF核一定是线性可分的吗

不一定，RBF核比较难调参而且容易出现维度灾难，要知道无穷维的概念是从泰勒展开得出的。

(3) 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

(4) SVM的基本思想：

间隔最大化来得到最优分离超平面。方法是将这个问题形式化为一个凸二次规划问题，还可以等价位一个正则化的合页损失最小化问题。SVM又有硬间隔最大化和软间隔SVM两种。这时首先要考虑的是如何定义间隔，这就引出了函数间隔和几何间隔的概念（这里只说思路），我们选择了几何间隔作为距离评定标准（**为什么要这样，怎么求出来的要知道**），我们希望能够最大化与超平面之间的几何间隔x，同时要求所有点都大于这个值，通过一些变化就得到了我们常见的SVM表达式。接着我们发现定义出的x只是由个别几个支持向量决定的。对于原始问题（primal problem）而言，可以利用凸函数的函数包来进行求解，但是发现如果用对偶问题（dual）求解会变得更简单，而且可以引入核函数。而原始问题转为对偶问题需要满足KKT条件（这个条件应该细细思考一下）到这里还都是比较好求解的。因为我们前面说过可以变成软间隔问题，引入了惩罚系数，这样还可以引出hinge损失的等价形式（这样可以用梯度下降的思想求解SVM了）。我个人认为难的地方在于求解参数的SMO算法。

(5) 是否所有的优化问题都可以转化为对偶问题：

这个问题我感觉非常好，有了强对偶和弱对偶的概念。[用知乎大神的解释吧](#)

(6) 处理数据偏斜：

可以对数量多的类使得惩罚系数C越小表示越不重视，相反另数量少的类惩罚系数变大。

4、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、Bootstrap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) 将各个树的未采样子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）。

17	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。
目录	（4）GBDT和Random Forest区别 这个就和上面说的差不多。
收藏	（5）Xgboost 这个工具主要有以下几个特点：
评论	<ul style="list-style-type: none">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。
微信	
微博	
QQ	<h2>5、KNN和Kmean</h2>

- （1）KNN 和Kmean缺点
都属于惰性学习机制，需要大量的计算距离过程，速度慢的可以（但是都有相应的优化方法）。
- （2）KNN
KNN不需要进行训练，只要对于一个陌生的点利用离其最近的K个点的标签判断其结果。KNN相当于多数表决，也就等价于经验最小化。而KNN的优化方式就是用Kd树来实现。
- （3）Kmean
要求自定义K个聚类中心，然后人为的初始化聚类中心，通过不断增加新点变换中心位置得到最终结果。Kmean的缺点可以用Kmean+方法进行一些解决（思想是使得初始聚类中心之间的距离最大化）

6、EM算法、HMM、CRF

- 这三个放在一起不是很恰当，但是有互相有关联，所以就放在一起一起说了。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（maximization）。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。

7、常见基础问题

- （1）数据归一化（或者标准化，注意归一化和标准化不同）的原因
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。

17

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。

目录

(2) 衡量分类器的好坏：
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：

收藏

评论

- 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

微信

- 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

微博

QQ

(3) SVD和PCA

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。

(4) 防止过拟合的方法

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

(5) 数据不平衡问题

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
- 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
- 考虑数据的先验分布

文章标签：

面试

神经网络

深度学习

个人分类：

机器学习

学习数据挖掘进程

深度学习



【CSDN学院】零基础在线学Python全栈，在家学习一样就业拿高薪

找不到满意工作？寻求转型机会？月薪3000至30000，你只差一个技术！八大模块四大企业级项目，覆盖时下热门领域，起薪比肩2年从业者

马上了解 >>

想对作者说点什么？

我来说两句



xiaotao_1

2018-06-19 16:28:31

#1楼

不错