蒋豆芽

深度学习——正则化 已购

来自 【机器学习面试题汇总与解析（蒋豆芽面试题总结）】 | 70 浏览 | 0 回复 | 2021-04-23

蒋豆芽

+关注

机器学习面试题汇总与解析——正则化

1. **解决模型训练过拟合有哪些思路？** ☆☆☆☆☆
2. **如何判断过拟合？** ☆☆☆☆☆
3. **正则化l_1（lasso）和l_2（ridge）的区别？** ☆☆☆☆☆
4. **L1有什么缺点？** ☆☆☆☆☆
5. **L1正则为什么可以达到模型的稀疏性** ☆☆☆☆☆
6. **说说BN（Batch Normolization）的原理** ☆☆☆☆☆
7. **知道BN吗？公式写一下，有什么作用与优势？BN的计算过程。** ☆☆☆☆☆
8. **BN训练和测试有什么不同？** ☆☆☆☆☆
9. **介绍一下BN和LN？有什么差异？LN是在哪个维度上进行归一化？** ☆☆☆☆☆
10. **要同时使用BN和dropout该如何使用？** ☆☆☆☆☆
11. **BN的gama labada意义** ☆☆☆☆☆
12. **数据增强的方法** ☆☆☆☆☆
13. **两个正则化的参数分布** ☆☆☆☆☆
14. **在预测的时候，是使用dropout训练出的权重还是要乘以keep-prib呢，为什么？** ☆☆☆☆☆
15. **为什么Lasso可以筛选变量?** ☆☆☆☆☆
16. **L1正则化为什么能缓解过拟合** ☆☆☆☆☆
17. **BN+CONV融合公式及作用** ☆☆☆☆☆

==========

- 本专栏适合于**Python已经入门**的学生或人士，有一定的编程基础。
- 本专栏适合于**算法工程师、机器学习、图像处理求职**的学生或人士。
- 本专栏针对面试题答案进行了**优化，尽量做到好记、言简意赅。这才是一份面试题总结的正确打开方式。这样才方便背诵**
- 如专栏内容有错漏，欢迎在评论区指出或私聊我更改，一起学习，共同进步。

蒋豆芽

==========

1. **解决模型训练过拟合有哪些思路？** ☆☆☆☆☆

   参考回答

   1. l_1**正则化**，

      对于待正则的网络层参数 $\vec{w}$，l_1正则化为：

      $$l_1 = \lambda||\vec{w}||_1 = \sum_i |w_i| \tag{.}$$

      l_1不仅可以**约束参数量**，还可以使**参数更稀疏**。因为对目标函数经过优化后，一部分参数会变为0，另一部分参数为非零实值。**非零实值说明这部分参数是最重要的特征**。这样一看，l_1还可以用来**挑选特征**。

   2. l_2**正则化**

      l_1正则化我们理解后，l_2正则化也就很简单了，就是乘方加和：

      $$l_2 = \frac{1}{2}\lambda||\vec{w}||_2^2 = \sum_i |w_i|^2 \tag{.}$$

      l_2正则化会使部分特征**趋近于0**，也就达到正则化的目的了。

      此外，l_1正则化和l_2正则化也可以联合使用，例如：

      $$\lambda||\vec{w}||_1 + \lambda||\vec{w}||_2^2 \tag{.}$$

      这种形式也被称为“**Elastic网络正则化**”。

   3. **随机失活（dropout）**

      随机失活（dropout）是神经网络中相当常用的网络正则化方法。原理就在于，即使一张图像缺少少量部分，但我们人眼依然能分辨出目标。那一张图像可以有不同的地方缺失，这样就如同样本变多了，我们知道训练样本越大，网络越不容易拟合。

      随机失活（dropout）在实际使用过程中，就有一个失活率ρ，当然这个失活率不能太大。常用的0.2-0.3

   4. **验证集的使用**

      我们可以将数据集划分为**训练集**、**验证集**、**测试集**，在训练过程中使用训练集，在每一个epoch结束后使用**验证集**验证模型效果，画出训练曲线，这样就可以判断是否过拟合了。当发现网络有点过拟合了，当然就是“**早停**”了，可以直接停止训练了。

   5. **扩充数据集**

      数据集越大，网络泛化性能越好，所以努力扩充数据集，通过平移、翻转、旋转、放缩、随机截取、加噪声、色彩抖动等等方式。

蒋豆芽

7. **Bagging和Boosting**（模型融合）

   Bagging算法使用**Bootstrap方法**从原始样本集中随机抽取N个样本但**不放回**。共提取K个轮次，得到K个独立的训练集，元素可以重复。分类问题以结果中的多个值投票作为最终结果，回归问题以平均值作为最终结果。结果采用投票法，避免了决策树的过拟合问题。

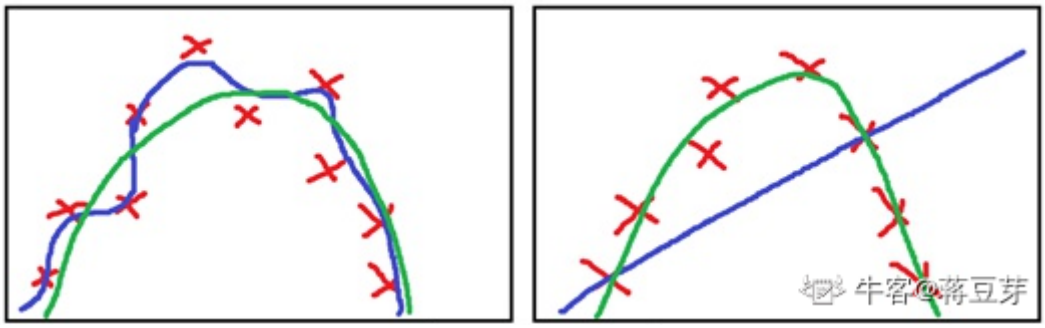
   Boosting是为每个训练样本设置一个权重，在下一轮分类中，误分类的样本权重较大，即每轮样本相同，但样本权重不同；对于分类器来说，分类误差小的样本权重较大，反之则小。

   采用模型融合的方式也可以避免过拟合。

答案解析

过拟合与欠拟合

讲正则化前，需要先讲过拟合与欠拟合。**神经网络在完成学习后，对应就是一个函数，网络学习的过程，就是这个函数拟合数据分布的过程**：



如图中，绿色就是拟合刚好的函数。而左边的蓝色就是过拟合，拟合得太好了，针对以后的新数据可能泛化性能就差；而右边的蓝色就是欠拟合，没有很好的拟合现有的数据分布。

过拟合与欠拟合的评判标准

过拟合：训练集效果很好，测试集效果较差

欠拟合：训练集效果差，测试集效果也差。

正则化的作用

正则化应用于过拟合的场景，正则化可以减轻过拟合现象，使得网络更好地拟合数据。

什么是正则化

我们从损失函数入手，前面的文章我们讲过**均方误差损失**函数，神经网络学习的过程就是最小化我们的损失函数，如下：

蒋豆芽

这个$f(x)$可以是二次曲线、三次曲线，甚至n次曲线，但是不管它是几次曲线，从高等数学的**泰勒展开原理**我们可以得知，$f(x)$可以表示为多项式的趋近：$f(x) = w_0x_0 + w_1x_1 + ... + w_nx_n$。

好，那么直观来看，$w_0, w_1 ... w_n$都是特征，网络为什么出现过拟合？**就是因为特征分得太细、太多。**我们举个例子，我们可以根据“头发长短”、“繁殖器官”这两个特征基本就可以很好的区分“男”、“女”，泛化性能更好，虽然会将“女装大佬”分错，但是“女装大佬”毕竟是少数。而如果我们特征分得更细化，“胡子长短”、“喉结大小”、“指甲长短”，这样虽然可以区分出“女装大佬”，但是这就过拟合了，网络很可能会将“小喉结的男人”都区分成女人了，错误率就提高了，反而**过拟合**了。

所以一个网络过拟合，可能就是特征过多，**那我们减少特征不就可以减轻过拟合了吗？**即减少w的个数。**问题就转化为求0范数（向量中非0元素的个数）了。**减少w的个数就是最小化$|\vec{w}|_0$。我们将这一项加入目标函数中：

$$arg\,min\ \frac{1}{N}\sum_{i=1}^{N}(y_i - f(x_i))^2 + |\vec{w}|_0 \tag{.}$$

所以最终的目标就是既要**损失函数最小**，又要**控制特征的数量**，那么就是两者求和最小，要在这两者的约束下进行网络优化，这就是我们的最终目标了。

1. l_1**正则化**，

   对于待正则的网络层参数 $\vec{w}$，l_1正则化为：

   $$l_1 = \lambda||\vec{w}||_1 = \sum_i |w_i| \tag{.}$$

   其中，λ控制正则项大小，较大的λ取值将较大程度约束模型复杂度；反之亦然。需注意，l_1不仅可以约束参数量，还可以使参数更稀疏。因为对目标函数经过优化后，**一部分参数会变为0（一部分特征被去除了，减轻模型过拟合）**，另一部分参数为非零实值。**非零实值说明这部分参数是最重要的特征。**这样一看，l_1还可以用来**挑选特征。**

2. l_2**正则化**

   l_1正则化我们理解后，l_2正则化也就很简单了，就是乘方加和：

   $$l_2 = \frac{1}{2}\lambda||\vec{w}||_2^2 = \sum_i |w_i|^2 \tag{.}$$

   l_2正则化会使部分特征趋近于0，也就达到正则化的目的了。

   此外，l_1正则化和l_2正则化也可以联合使用，例如：

   $$\lambda||\vec{w}||_1 + \lambda||\vec{w}||_2^2 \tag{.}$$

   这种形式也被称为“Elastic网络正则化”。

3. **随机失活（dropout）**

同样本变多了，我们知道训练样本越大，网络越不容易拟合。

对网络来说，每一层有很多个网络节点，那我们让一部分节点失活，在该epoch中不参与网络训练，这样就可以避免网络过拟合。

随机失活（dropout）在实际使用过程中，就有一个失活率ρ，当然这个失活率不能太大。常用的0.2-0.3

4. **验证集的使用**

   我们可以将数据集划分为**训练集**、**验证集**、**测试集**，在训练过程中使用训练集，在每一个epoch结束后使用验证集验证模型效果，画出训练曲线，这样就可以判断是否过拟合了。当发现网络有点过拟合了，当然就是“**早停**”了，可以直接停止训练了。

5. **扩充数据集**

   这个方法不用多说，数据集越大，网络泛化性能越好，所以努力扩充数据集，通过平移、翻转、旋转、放缩、随机截取、加噪声、色彩抖动等等方式。

6. **BN（Batch Normolization）**

   BN（Batch Normolization）是Google提出的用于解决深度网络**梯度消失**和**梯度爆炸**的问题，可以起到**一定的正则化作用**。我们来说一下它的原理：

   批规范化，即在模型每次随机梯度下降训练时，通过mini-batch来对每一层卷积的输出做**规范化操作**，使得结果（各个维度）的**均值为0**，**方差为1**。

   BN操作共分为四步。输入为x_i，第一步计算均值：

$$\mu_\beta = \frac{1}{m}\sum_{i=1}^{m} x_i \tag{.}$$

   第二步计算数据方差：

$$\sigma_\beta^2 = \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_\beta)^2 \tag{.}$$

   第三步进行规范化：

$$x_i^* = \frac{x_i - \mu_\beta}{\sqrt{\sigma_\beta^2 + \epsilon}} \tag{.}$$

   第四步尺度变换和偏移：

$$y_i = \gamma \cdot x_i^* + \beta = BN_{\gamma,\beta}(x_i) \tag{.}$$

   m表示mini-batch中的数据个数，可以看出，BN实际就是对网络的每一层都进行**白化操作**。**白化操作是线性的，最后的“尺度变换和偏移”操作是为了让BN能够在线性和非线性之间做一个权衡**，而这个偏移的参数γ和 β 是神经网络在训练时学出来的（即当$\gamma = \sqrt{Var(x_i)} = \sigma_\beta$和$\beta = E(x_i) = \mu_\beta$时），从而保证整个网络的容量（capacity）。

参数（γ、β）有什么作用

对网络某一层**A**的输出数据做**归一化**，然后送入网络下一层**B**，这样是会**影响到本层网络A所学习到的特征的。**比如网络中间某一层学习到特征数据本身就分布在S型激活函数的两侧，如果强制把它给归一化处理、标准差也限制在了1，把数据变换成分布于s函数的中间部分，这样就相当于这一层网络所学习到的特征分布**被搞坏**了。于是**BN**最后的“**尺度变换和偏移**”操作，引入了可学习参数γ、β，这就是算法**关键之处。**引入了这个可学习重构参数γ、β，**让我们的网络可以学习恢复出原始网络所要学习的特征分布。**

7. **Bagging和Boosting**（模型融合）

   Bootstrap是一种抽样方法，即随机抽取数据并将其放回。如一次抽取一个样本，然后放回样本集中，下次可能再抽取这个样本。接着将每轮未抽取的数据合并形成**袋外数据集**（Out of Bag, OOB），用于模型中的测试集。

   Bagging算法使用Bootstrap方法从原始样本集中随机抽取N个样本但不放回。共提取K个轮次，得到K个独立的训练集，元素可以重复。分类问题以结果中的多个值**投票**作为最终结果，回归问题以**平均值**作为最终结果。结果采用投票法，避免了决策树的过拟合问题。

   Boosting是为每个训练样本设置一个权重，在下一轮分类中，误分类的样本权重较大，即每轮样本相同，但样本权重不同；对于分类器来说，分类误差小的样本权重较大，反之则小。

   采用模型融合的方式也可以避免过拟合。

类似的问题还有：

2. **如何判断过拟合？** ☆☆☆☆☆

   参考回答

   过拟合：训练集效果很好，测试集效果较差

   欠拟合：训练集效果差，测试集效果也差。

   答案解析

   无。

3. **正则化l_1（lasso）和l_2（ridge）的区别？** ☆☆☆☆☆

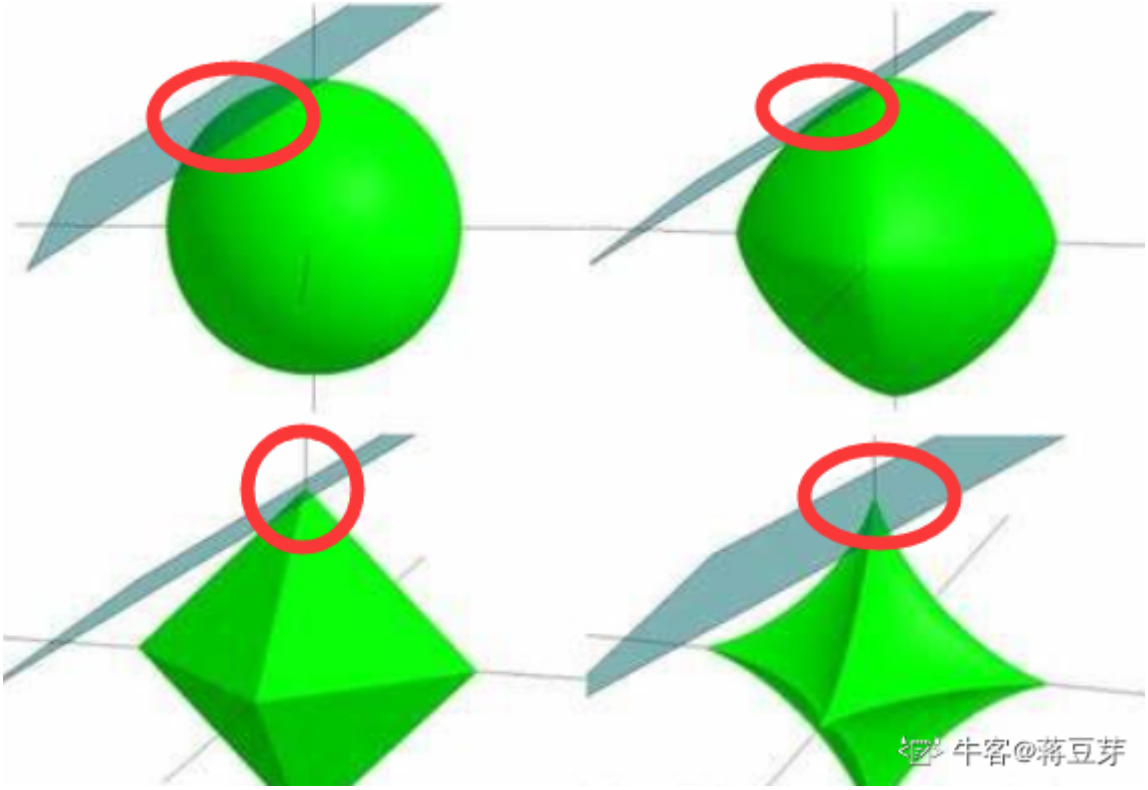
   参考回答

   1. L1是模型各个参数的绝对值之和；L2是模型各个参数的平方和的开方值。
   2. L1会趋向于产生少量的特征，而其他的特征都是0；L2会选择更多的特征，这些特征都会接近于0。

蒋豆芽

L1正则项解空间为菱形。最优解必定是有且仅有一个交点。除非目标函数具有特殊的形状，否则和菱形的唯一交点大概率出现在坐标轴上，这样就会导致某一维的权重为0 ，产生**稀疏权重矩阵**。而对于**L2**圆形的解空间，总存在一个切点，切点通常不会位于坐标轴上，因此每一维的参数都不会是0，当最小化||w||时，就会使每一项趋近于0。

直接观察其图像，下面四幅分别为p=2，p=1.5，p=1，p=0.7



4. **L1有什么缺点?** ☆☆☆☆☆

参考回答

L1正则要算绝对值，算绝对值比较麻烦；直接平方要比算绝对值来得简单，这一点上**L2正则**计算更加简便**（优化时求导方便）**。

答案解析

无。

5. **L1正则为什么可以达到模型的稀疏性** ☆☆☆☆☆

参考回答

L1是模型各个参数的绝对值之和$|\vec{w}|_0$，那么对目标函数经过优化后，一部分参数会变为0，另一部分参数为非零实值。这样我们就得到了**稀疏特征**了。

和菱形的唯一交点大概率出现在坐标轴上，这样就会导致某一维的权重为0 ，产生**稀疏权重矩阵**。

6. **说说BN（Batch Normolization）的原理**☆☆☆☆☆

   参考回答

   BN（Batch Normolization）是Google提出的用于解决深度网络**梯度消失**和**梯度爆炸**的问题，可以起到一定的**正则化作用**。我们来说一下它的原理：

   批规范化，即在模型每次随机梯度下降训练时，通过mini-batch来对每一层卷积的输出做规范化操作，使得结果（各个维度）的**均值为0**，**方差为1**。

   BN操作共分为四步。输入为x_i，第一步计算均值：

   第二步计算数据方差：

   第三步进行规范化：

   第四步尺度变换和偏移：

   答案解析

   无。

7. **知道BN吗？公式写一下，有什么作用与优势？BN的计算过程。** ☆☆☆☆☆

   参考回答

   答案参考上面。

   答案解析

   无。

8. **BN训练和测试有什么不同？** ☆☆☆☆☆

   参考回答

   训练时，均值和方差针对一个**Batch**。

   测试时，均值和方差针对**整个数据集**而言。因此，在训练过程中除了正常的前向传播和反向求导之外，我们还要记录**每一个Batch的均值和方差**，以便训练完成之后按照下式计算**整体的均值和方差**：

蒋豆芽

$$\mathrm{Var}[x] \leftarrow \frac{m}{m-1}\mathbb{E}_B[\sigma_B^2]$$

上面简单理解就是：对于**均值**来说直接计算所有batch μ_β值的平均值；然后对于**标准偏差**采用每个batch σ_β的**无偏估计（无偏估计是用样本统计量来估计总体参数时的一种无偏推断）**。最后测试阶段，BN的使用公式就是：

$$y = \frac{\gamma}{\sqrt{\mathrm{Var}[x]+\epsilon}} \cdot x + \left(\beta - \frac{\gamma\,\mathrm{E}[x]}{\sqrt{\mathrm{Var}[x]+\epsilon}}\right)$$

答案解析

无。

9. **介绍一下BN和LN？有什么差异？LN是在哪个维度上进行归一化？** ☆☆☆☆☆

参考回答

LN：Layer Normalization，LN是“横”着来的，对一个样本，**不同的神经元**间做**归一化**。

BN：Batch Normalization，BN是“竖”着来的，**各个维度**做**归一化**，所以与**batch size**有关系。

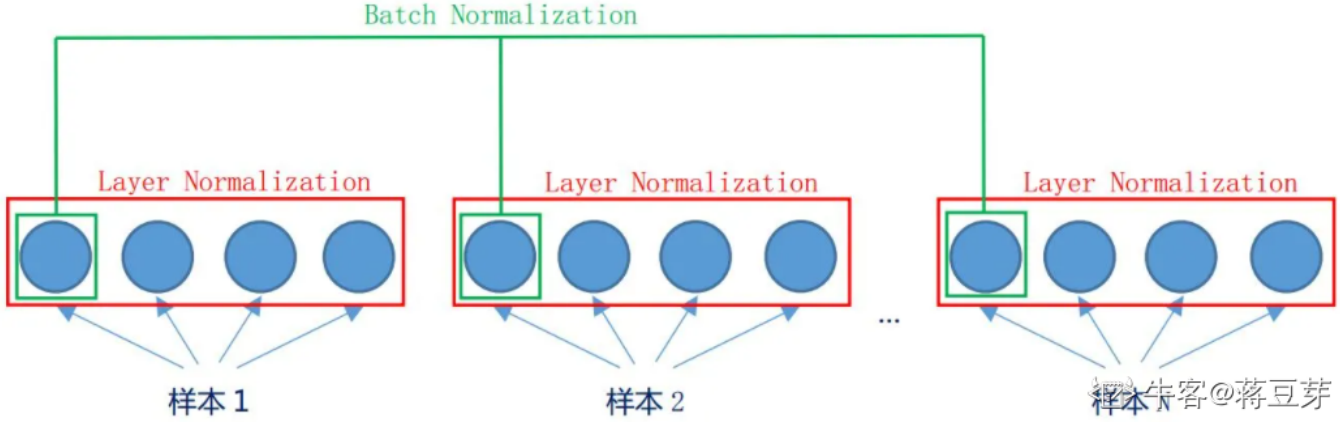
二者提出的目的都是为了加快模型收敛，减少训练时间。

答案解析

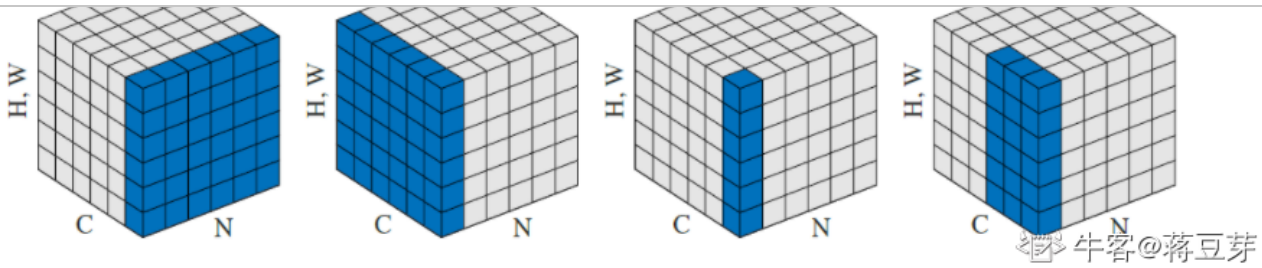
LN中同层神经元输入拥有相同的均值和方差，不同的输入样本有不同的均值和方差；

BN中则针对不同神经元输入计算均值和方差，同一个batch中的输入拥有相同的均值和方差。

所以，LN不依赖于batch的大小和输入sequence的深度，因此可以用于**batchsize为1**和RNN中对边长的输入sequence的normalize操作。



蒋豆芽



10. **要同时使用BN和dropout该如何使用？** ☆☆☆☆☆

参考回答

同时使用BN和dropout时，可能存在**方差偏移**的问题

针对**方差偏移**，论文给出了两种解决方案：

1. 拒绝方差偏移，只**在所有BN层的后面采用dropout层**（现在大部分开源的模型，都在网络的中间加了BN，我们也就只能在softmax的前一层加加dropout了，效果还行，至少不会比不加dropout差。还有另外一种方法是模型训练完后，固定参数，以测试模式对训练数据求BN的均值和方差，再对测试数据进行归一化，论文证明这种方法优于baseline）
2. dropout原文提出了一种高斯dropout，论文再进一步对高斯dropout进行扩展，提出了一个**均匀分布Dropout**，这样做带来了一个好处就是这个形式的Dropout（又称为“Uout”）对方差的偏移的敏感度降低了

答案解析

1. **dropout**

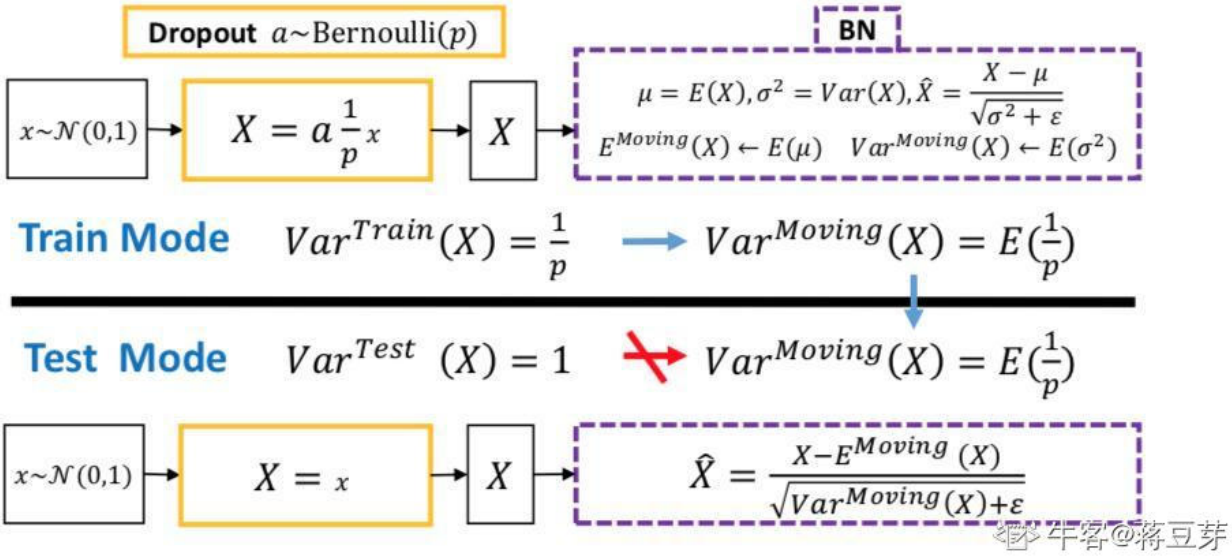
   dropout在训练时，以一定的概率**p**来drop掉相应的神经网络节点，以(1-p)的概率来保留相应的神经网络节点，这相当于每一次训练时模型的网络结构都不一样，也可以理解为训练时添加了不同的数据，所以能够有效减少**过拟合**。

   问题呢，是出在测试时，因为训练的时候以**概率p** drop了一些节点，比如dropout设置为0.5，隐藏层共有6个节点，那训练的时候有3个节点的值被丢弃，而测试的时候这6个节点都被保留下来，这就导致了**训练**和**测试**的时候以该层节点为输入的下一层的神经网络节点获取的**期望**会有量级上的差异。为了解决这个问题，在训练时对当前dropout层的输出数据**除以（1-p）**，之后再输入到下一层的神经元节点，以作为失活神经元的补偿，以使得在训练时和测试时每一层的输入有大致相同的期望。

2. **Batch Normalization**

   BN就是在深度神经网络训练时通过对每一个batch的数据采用**均值**和**方差**进行**归一化**，使得每一层神经网络的输入保持相同的分布，这样能够加快训练的速度。此外，因为在训练时，为每一次迭代求全局的均值和方差是不现实的，因此借鉴moment的方式对均值和方差进行更新，使得每一层归一化的均值和方差都不一样，也相当于引入了噪声，能够增加模型的鲁棒性，有效减少过拟合。

蒋豆芽



首先，先明确dropout和BN结合使用使模型性能下降的连接方式，用通俗的话讲，就是先在网络的内部使用dropout，随后再跟上一个BN层，而且这个BN层还不止一个。那么问题出在哪呢？原因有二。首先，如上图所示，因为训练时采用了dropout，虽然通过除以(1-p)的方式来使得训练和测试时，每个神经元输入的期望大致相同，但是他们的**方差却不一样。**第二，BN是采用训练时得到的均值和方差对数据进行归一化的，现在**dropout层的方差不一样**，一步错步步错，**最终导致输出不准确，影响最后的性能。**

11. **BN的gama labada意义** ☆☆☆☆☆

参考回答

对网络某一层**A**的输出数据做**归一化**，然后送入网络下一层**B**，这样是会**影响到本层网络A所学习到的特征的。**于是**BN**最后的“**尺度变换和偏移**”操作，引入了可学习参数γ、β，这就是算法**关键之处。**引入了这个可学习重构参数γ、β，**让我们的网络可以学习恢复出原始网络所要学习的特征分布。**

答案解析

无。

12. **数据增强的方法** ☆☆☆☆☆

参考回答

数据集越大，网络泛化性能越好，所以努力扩充数据集，通过平移、翻转、旋转、放缩、随机截取、加噪声、色彩抖动等等方式。

答案解析

蒋豆芽

13. **两个正则化的参数分布** ☆ ☆ ☆ ☆ ☆

参考回答

L1正则化假设参数分布为**Laplace分布**；L2正则化假设参数分布为**正态分布**

答案解析

无。

14. **在预测的时候，是使用dropout训练出的权重还是要乘以keep-prib呢，为什么?** ☆ ☆ ☆ ☆ ☆

参考回答

要乘以keep-prib。

因为神经元预测的时候就没办法随机丢弃，一种”补偿“的方案就是每个神经元的权重都乘以一个**p**，这样在“总体上”使得**测试数据**和**训练数据**是大致一样的。保证**测试**的时候把这个神经元的权重乘以**p**可以得到**同样的期望。**

答案解析

因为我们训练的时候会随机的丢弃一些神经元，但是预测的时候就没办法随机丢弃了。如果丢弃一些神经元，这会带来结果不稳定的问题，也就是给定一个测试数据，有时候输出a有时候输出b，结果不稳定，这是实际系统不能接受的，用户可能认为模型预测不准。那么一种”补偿“的方案就是每个神经元的权重都乘以一个p，这样在“总体上”使得测试数据和训练数据是大致一样的。比如一个神经元的输出是**x**，那么在训练的时候它有**p**的概率参与训练，**(1-p)**的概率丢弃，那么它输出的期望是$p \times x + (1-p) \times 0 = p \times x$。因此**测试**的时候把这个神经元的权重乘以**p**可以得到**同样的期望。**

15. **为什么Lasso可以筛选变量?** ☆ ☆ ☆ ☆ ☆

参考回答

略。

答案解析

无。

16. **L1正则化为什么能缓解过拟合** ☆ ☆ ☆ ☆ ☆

参考回答

蒋豆芽

答案解析

无。

17. **BN+CONV融合公式及作用** ☆☆☆☆☆

参考回答

网络完成训练后，在inference阶段，为了加速运算，通常将卷积层和BN层进行融合：

卷积层：

$$z = w * x + b$$

BN层：

$$y = \frac{x - E[x]}{\sqrt{var[x]}} \cdot \gamma + \beta$$

融合两层： 将Conv层的公式带入到BN层的公式

$$\begin{aligned} y &= \frac{w * x + b - E[x]}{\sqrt{var[x]}} \cdot \gamma + \beta \\ &= \frac{w * x}{\sqrt{var[x]}} \cdot \gamma + \left(\frac{b - E[x]}{\sqrt{var[x]}} \cdot \gamma + \beta \right) \end{aligned}$$

融合后相当于：

$$w_{new} = \frac{w * x}{\sqrt{var}} \cdot \gamma$$

$$b_{new} = \frac{b - mean}{\sqrt{var}} \cdot \gamma + \beta$$

inference阶段，**E[x]为滑动均值**，**Var[x]为滑动方差**

将BN层融合到卷积层中，相当于对卷积核进行一定的修改，没有增加卷积的计算量，同时整个BN层的计算量都省去了。

答案解析

无。

蒋豆芽

收藏 赞

相关专栏

机器学习面试题汇总与解析（蒋豆芽面试题总结）

27篇文章 | 90订阅

已订阅

0条评论 默认排序

没有回复

请留下你的观点吧~

发布