深度学习补缺补漏篇！准算法工程师总结出的超强面经（含答案）

原创 CV开发者都爱看的 极市平台 2021-07-04 22:05:00 手机阅读

收录于话题

#CV面经

作者丨灯会

来源丨极市平台

编辑丨极市平台

极市导读

作者灯会为21届中部985研究生，七月份将入职某互联网大厂cv算法工程师。在去年灰飞烟灭的算法求职季中，经过几十场不同公司以及不同部门的面试中积累出了CV总复习系列，此为深度学习补缺补漏篇。 >>加入极市CV技术交流群，走在计算机视觉的最前沿

系列文章：

- 深度学习三十问！一位算法工程师经历30+场CV面试后总结的常见问题合集（含答案）
- 深度学习六十问！一位算法工程师经历30+场CV面试后总结的常见问题合集下篇（含答案）
- 一位算法工程师从30+场秋招面试中总结出的超强面经—语义分割篇（含答案）
- 一位算法工程师从30+场秋招面试中总结出的超强面经——目标检测篇（含答案）
- 图像处理知多少？准大厂算法工程师30+场秋招后总结的面经问题详解
- 一位算法工程师从30+场秋招面试中总结出的超强面经—文本检测与GAN篇（含答案）
- 准算法工程师从30+场秋招中总结出的超强面经—C、Python与算法篇篇（含答案）
- 计算机基础与手撕代码篇！准算法工程师总结出的超强面经（含答案）

一、深入理解Batch Normalization批标准化

机器学习领域有个很重要的假设：**IID独立同分布假设**，就是假设训练数据和测试数据是满足相同分布的，这是通过训练数据获得的模型能够在测试集获得好的效果的一个基本保障。那BatchNorm的作用是什么呢？**BatchNorm就是在深度神经网络训练过程中使得每一层神经网络的输入保持相同分布的。**

BN的基本思想其实相当直观：因为深层神经网络在做非线性变换前的**激活输入值**（就是那个x=WU+B，U是输入）**随着网络深度加深或者在训练过程中，其分布逐渐发生偏移或者变动，之所以训练收敛慢，一般是整体分布逐渐往非线性函数的取值区间的上下限两端靠近**（对于Sigmoid函数来说，意味着激活输入值WU+B是大的负值或正值），所以这**导致反向传播时低层神经网络的梯度消失**，这是训练深层神经网络收敛越来越慢的**本质原因**，**而BN就是通过一定的规范化手段，把每层神经网络任意神经元这个输入值的分布强行拉回到均值为0方差为1的标准正态分布**，其实就是把越来越偏的分布强制拉回比较标准的分布，这样使得激活输入值落在非线性函数对输入比较敏感的区域，这样输入的小变化就会导致损失函数较大的变化，意思是**这样让梯度变大，避免梯度消失问题产生，而且梯度变大意味着学习收敛速度快，能大大加快训练速度。**

BN在训练的时候可以根据Mini-Batch里的若干训练实例进行激活数值调整，但是在推理（inference）的过程中，从所有训练实例中获得的统计量来代替Mini-Batch里面m个训练实例获得的均值和方差统计量。

BatchNorm为什么NB呢，关键还是效果好。**①不仅仅极大提升了训练速度，收敛过程大大加快；②还能增加分类效果，一种解释是这是类似于Dropout的一种防止过拟合的正则化表达方式，所以不用Dropout也能达到相当的效果；③另外调参过程也简单多了，对于初始化要求没那么高，而且可以使用大的学习率等。** 总而言之，经过这么简单的变换，带来的好处多得很，这也是为何现在BN这么快流行起来的原因。

二、拉格朗日乘子法和 KKT 条件

为了方便和好记，就把原来的优化问题写成f(x,y)+λh(x,y)的形式，然后分别对λ,x,y求偏导，并且令偏导为0就行了，和之前得到的方程组是一样的。这种方法叫**拉格朗日乘数法**。

KKT条件:

1. **原可行性:g(x*)≤0**
2. **对偶可行性: α≥0**
3. **互补松弛条件:αg(x*)=0**
4. **拉格朗日平稳性: ▽f(x*)=α×▽g(x*)**

这个就是 KKT 条件。 它的含义是这个优化问题的极值点一定满足这组方程组。（不是极值点也可能会满足，但是不会存在某个极值点不满足的情况）它也是原来的优化问题取得极值的必要条件，解出来了极值点之后还是要代入验证的。但是因为约束比较多，情况比较复杂，KKT 条件并不是对于任何情况都是满足的。要满足 KKT条件需要有一些规范性条件（Regularity conditions），就是要求约束条件的质量不能太差，常见的比如：

1. LCQ：如果 h(x)和g(x)都是形如Ax+b的仿射函数，那么极值一定满足 KKT 条件。
2. LICQ：起作用的g(x)函数（即g(x)相当于等式约束的情况）和h(x)函数在极值点处的梯度要线性无关，那么极值一定满足 KKT 条件。
3. Slater条件：如果优化问题是个凸优化问题，且至少存在一个点满足h(x)=0和g(x)=0，极值一定满足 KKT 条件。并且满足强对偶性质（下面会讲）。

三、L0、L1、L2范式

一般来说，监督学习可以看做最小化下面的目标函数：

$$w^* = \arg\min_w \sum_i L(y_i, f(x_i; w)) + \lambda\Omega(w)$$

其中，第一项L(yi,f(xi;w)) 衡量我们的模型（分类或者回归）对第i个样本的预测值f(xi;w)和真实的标签yi之前的误差。因为我们的模型是要拟合我们的训练样本的嘛，所以我们要求这一项最小，也就是要求我们的模型尽量的拟合我们的训练数据。但正如上面说言，我们不仅要保证训练误差最小，我们更希望我们的模型测试误差小，所以我们需要加上第二项，也就是对参数w的规则化函数Ω(w)去约束我们的模型尽量的简单。

正则函数部分：规则化函数Ω(w)也有很多种选择，一般是模型复杂度的单调递增函数，模型越复杂，规则化值就越大。比如，规则化项可以是模型参数向量的范数。然而，不同的选择对参数w的约束不同，取得的效果也不同，但我们在论文中常见的都聚集在：L0范数，L1范数，L2范数，迹范数，Frobenius范数，核范数等。

1、L0范数

L0范数是指向量中非0的元素的个数。如果用L0规则化一个参数矩阵W，就是**希望W中大部分元素是零，实现稀疏。**

L0范数的应用：

1）特征选择：实现特征的自动选择，去除无用特征。*稀疏化可以去掉这些无用特征，将特征对应的权重置为零。*

2）可解释性（interpretability）：例如判断某种病的患病率时，最初有1000个特征，建模后参数经过稀疏化，最终只有5个特征的参数是非零的，那么就可以说影响患病率的主要就是这5个特征。

2、L1范数

L1范数是指向量中各个元素绝对值之和。L1范数也可以实现稀疏，通过将无用特征对应的参数W置为零实现。

L1范数：

$$\|x\|_1 = \sum_{i=1}^{N} |x_i|$$

既然L0可以实现稀疏，为什么不用L0，而要用L1呢？个人理解一是因为L0范数很难优化求解（NP难问题），二是L1范数是L0范数的最优凸近似，而且它比L0范数要容易优化求解。

3、L2范数

L2范数是指向量各元素的平方和然后求平方根。，用在回归模型中也称为岭回归（Ridge regression）。

L2范数：

$$\|\mathbf{x}\|_2 = \sqrt{\sum_{i=1}^{N} x_i^2}$$

L2避免过拟合的原理是：让L2范数的规则项

$$\|\mathbf{x}\|_2$$

尽可能小，可以使得W每个元素都很小，接近于零，但是与L1不同的是，不会等于0；这样得到的模型**抗干扰能力强**，参数很小时，即使样本数据x发生很大的变化，模型预测值y的变化也会很有限。

L2范数的作用：1）学习理论的角度：从学习理论的角度来说，**L2范数可以防止过拟合，提升模型的泛化能力**。2）优化计算的角度：从优化或者数值计算的角度来说，**L2范数有助于处理 condition number不好的情况下矩阵求逆很困难的问题。**

四、L1 Loss& L2 Loss&smooth L1 Loss

1、L1 Loss

L1 我理解成 1 维向量的距离。

使用L1损失函数也被叫做最小化绝对误差(Least Abosulote Error)。这 个名称非常的形象。LAE 就是最小化真实值yi和预测值 $f(xi)$ 之间差值 D_{L1} 的绝对值的和。

$$D_{L1} = \sum_{i=1}^{n} |y_i - f(x_i)|$$

这里的 D_{L1} 其实就是平均绝对误差(MAE),使用L1 损失函数也就是minD_I

公式：

$$L1 = \sum_{i=1}^{n} |y_i - f(x_i)|$$

导数：

$$\frac{\mathrm{d}L_1(x)}{\mathrm{d}x} = \begin{cases} 1 & \text{if } x \geq 0 \\ -1 & \text{otherwise} \end{cases}$$

2、L2 Loss

L2 就是二维空间向量的距离。

公式：

$$L2=\sum_{i=1}^{n}(y_i-f(x_i))^2$$

导数：

$$\frac{\mathrm{d}L_2(x)}{\mathrm{d}x}=2x$$

3、smooth L1 Loss：

公式：

$$\operatorname{smooth}_{L_1}(x)=\begin{cases}0.5x^2 & \text{if } |x|<1\\ |x|-0.5 & \text{otherwise}\end{cases}$$

导数：

$$\frac{\mathrm{d}\operatorname{smooth}_{L_1}}{\mathrm{d}x}=\begin{cases}x & \text{if } |x|<1\\ \pm 1 & \text{otherwise}\end{cases}$$

$$\text{Smooth}\quad L_1=\begin{array}{ll}0.5x^2, & |x|<1\\ |x|-0.5, & x<-1 \text{ or } x>1\end{array}$$

$$\text{Smooth}\quad L_1'=\begin{array}{ll}x, & |x|<1\\ -1, & x<-1\\ 1, & x>1\end{array}$$

比较：

1、L1 loss 在零点不平滑，用的较少；L1loss在零点不平滑，学习慢；2、Smooth L1 Loss 改善了零点不平滑问题。

smooth L1 loss和L1 loss函数的区别在于，L1 loss在0点处导数不唯一，可能影响收敛。smooth L1 loss的解决办法是在0点附近使用平方函数使得它更加平滑。

smooth L1 loss让loss function对于离群点更加鲁棒，即：相比于L2损失函数，其对离群点/异常值（outlier）不敏感，梯度变化相对更小，训练时不容易跑飞。

Smooth L1 Loss 相比L1修改零点不平滑问题，而且在x较大的时候不像L2对异常值敏感，是一个缓慢变化的loss；

3、L2 loss：对离群点比较敏感，如果feature 是 unbounded的话，需要好好调整学习率，防止出现梯度爆炸的情况。

L2loss学习快，因为是平方增长，但是当预测值太大的时候，会在loss中占据主导位置(如真实值为1，预测多次，有一次预测值为100，其余预测为2)。

4、一般来说，L1正则会制造稀疏的特征，大部分无用特征的权重会被置为0。L2正则会让特征的权重不过大，使得特征的权重比较平均。

python numpy实现

```
import numpy as np
#定义L1损失函数
def L1_loss(y_true,y_pre):
    return np.sum(np.abs(y_true-y_pre))
#定义L2损失函数
def L2_loss(y_true,y_pre):
    return np.sum(np.square(y_true-y_pre))
```

```
    return np.sum(np.square(y_true-y_pre))

#假设我们得到了真实值和预测值
y_true = np.array([1,2,3,4,5,6,7,8,9])
y_pre  = np.array([1.2,2.3,3.5,4.3,4.6,5.6,6.1,7.1,8.8])

#定义函数
print('L1 loss is {}'.format(L1_loss(y_true,y_pre)))
"""L1 loss is 4.1000000000000005"""
print('L2 loss is {}'.format(L2_loss(y_true,y_pre)))
"""L2 loss is 2.4500000000000001"""
```

五、Sigmoid与softmax的区别

1、Sigmoid函数

sigmoid常用于**二元分类**，将二元输入映射成0和1。

函数：

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

导数：

$$\mathrm{f}'(\mathrm{z})=\mathrm{f}(\mathrm{z})(1-\mathrm{f}(\mathrm{z}))$$

sigmoid函数推导：

$$\begin{aligned}
f'(z) &= \left(\frac{1}{1+e^{-z}}\right)' \\
&= \frac{e^{-z}}{(1+e^{-z})^2} \\
&= \frac{1+e^{-z}-1}{(1+e^{-z})^2} \\
&= \frac{1}{(1+e^{-z})}\left(1-\frac{1}{(1+e^{-z})}\right) \\
&= f(z)(1-f(z))
\end{aligned}$$

其实logistic函数也就是经常说的sigmoid函数，它的几何形状也就是一条sigmoid曲线（S型曲线）。A logistic function or logistic curve is a common “S” shape (sigmoid curve). 也就是说，sigmoid把一个值映射到0-1之间。

该函数具有如下的特性：当x趋近于负无穷时，y趋近于0；当x趋近于正无穷时，y趋近于1；当x= 0时，y=0.5.

优点： 1.Sigmoid函数的输出映射在(0,1)之间，单调连续，输出范围有限，优化稳定，可以用作输出层。2.求导容易，处处可导，导数为：f′(x)=f(x)(1−f(x))

缺点： 1.由于其软饱和性，容易产生梯度消失，导致训练出现问题。2.其输出并不是以0为中心的。

2、Softmax函数

公式：

$$\sigma(\mathbf{z})_j=\frac{e^{z_j}}{\sum_{k=1}^{K}e^{z_k}}\quad \text{for } j=1,\ldots,K$$

其中 θi和x是列向量,

$$\theta_i^T x$$

可能被换成函数关于x的函数 fi(x)。

参考链接

https://www.cnblogs.com/guoyaohua/p/8724433.html

https://www.cnblogs.com/xinchen1111/p/8804858.html

https://zhuanlan.zhihu.com/p/51912576

https://blog.csdn.net/hejunqing14/article/details/48980321/

极市原创作者激励计划

极市平台深耕CV开发者领域近5年，拥有一大批优质CV开发者受众，覆盖微信、知乎、B站、微博等多个渠道。通过极市平台，您的文章的观点和看法能分享至更多CV开发者，既能体现文章的价值，又能让文章在视觉圈内得到更大程度上的推广。

对于优质内容开发者，极市可推荐至国内优秀出版社合作出书，同时为开发者引荐行业大牛，组织个人分享交流会，推荐名企就业机会，打造个人品牌 IP。

投稿须知：

1.作者保证投稿作品为自己的原创作品。

2.极市平台尊重原作者署名权，并支付相应稿费。文章发布后，版权仍属于原作者。

3.原作者可以将文章发在其他平台的个人账号，但需要在文章顶部标明首发于极市平台

投稿方式：

添加小编微信Fengcall（微信号：fengcall19），备注：**姓名-投稿**

△长按添加极市平台小编

觉得有用麻烦给个在看啦~

收录于话题 #CV面经 18

上一篇

算法岗面经分享 | 字节跳动CV算法实习生面试（含超多知识点干货）

下一篇

计算机基础与手撕代码篇！准算法工程师总结出的超强面经（含答案）

阅读原文