


如何训练一个性能不错的深度神经网络

2018-02-14 大数据挖掘DT数据分析

点击上方蓝字  然后开启 置顶公众号 

向AI转型的程序员都关注了这个号 

大数据挖掘DT数据分析 公众号： **datadw**

本文主要介绍8种实现细节的技巧或tricks：数据增广、图像预处理、网络初始化、训练过程中的技巧、激活函数的选择、不同正则化方法、来自于数据的洞察、集成多个深度网络的方法。

1. 数据增广

在不改变图像类别的情况下，增加数据量，能提高模型的泛化能力。

自然图像的数据增广方式包括很多，如常用的水平翻转（horizontally flipping），一定程度的位移或者裁剪和颜色抖动（color jittering）。此外还可以尝试多种操作的组合，例如同时做旋转和随机尺度变换，此外还可以把每个patch中所有像素在HSV颜色空间中的饱和度和明度提升0.25-4次幂方，乘以0.7-1.4之间的一个因子，再加一个-0.1-0.1之间的值。同样你可以在色调通道（H）对每张图片或patch的所有像素增加一个-0.1~0.1之间的值。

2. 预处理

2.1 最简单的预处理方法

零均值化

标准化

2.1.1 为什么要零均值化

数据有过大的均值可能导致参数的梯度过大，如果有后续的处理，可能要求数据零均值，比如PCA。零均值化并没有消除像素之间的相对差异，人们对图像信息的摄取通常来自于像素之间的相对色差，而不是像素值的高低。

2.1.2 为什么要归一化

归一化是为了让不同维度的数据具有相同的分布。假如二维数据 (X1,X2) 两个维度都服从均值为零的正态分布, 但是X1方差为100, X2方差为1。那么对 (X1,X2) 进行随机采样在二维坐标系中绘制的图像, 应该是狭长的椭圆形。

对这些数据做特征提取会用到以下形式的表达式:

$$S = w1*x1 + w2*x2 + b;$$

那么参数W1, W2的梯度为:

$$dS / dw1 = x1 ; dS / dw2 = x2$$

由于x1与x2在分布规模上的巨大差异, w1与w2的导数也会差异巨大。此时绘制目标函数 (不是S) 的曲面图, 就像一个深邃的峡谷, 沿着峡谷方向变化的是w2, 坡度很小; 在峡谷垂直方向变化的是w1, 坡度非常陡峭, 如图1, 而我们期望的目标函数是图2这样的。

目标函数是非常难以优化的, 因为w1和w2的梯度差异太大, 所以在两个维度上需要不同的迭代方案。但在实际操作中, 为了方便, 我们通常为所有维度设置相同的步长, 随着迭代的进行, 步长的缩减在不同维度也是同步的。这就要求w不同纬度的分布规模大致相同, 而这一切都始于数据的归一化。

2.1.3 实现代码

```
X-=numpy.mean(X,axis=0) # 即x的每一列都减去该列的均值
```

注: 对于灰度图像, 零均值化也可以减去整张图片的均值: `X-=numpy.mean(X)`

```
X/=numpy.std(X,axis=0) # 数据归一化。
```

X是输入数据, (图片数目X图片维度)。另一种标准化 (normalize) 方式是标准化每个维度, 保证每个维度的最大值和最小值是-1和1。这种预处理的方式只在输入的各个特征的尺度或者单位不同时才有意义。以图片像素作为输入为例子, 所有像素值尺度都在0-255这个尺度之间, 所以没必要严格的执行这种预处理操作。在自然图像上进行训练时, 可以不进行归一化操作, 因为理论上图像任一部分的统计性质都应该和其他部分相同, 图像的这种特性被称作平稳性 (stationarity)

本文来自 微信公众号 datadw 【大数据挖掘DT数据分析】

2.2 白化 (Whitening)

白化相当于在零均值化和归一化操作之间插入一个旋转操作，将数据投影到主轴上。一张图片经过白化后，可以认为每个像素之间是统计独立的。然而白化很少在卷积神经网络中使用，可能原因是图像信息本来就是依靠像素之间的相对差异来体现的，白化让像素间去相关，让这种差异变得不确定，损失了信息。

首先将数据零均值化，再计算协方差矩阵（covariance matrix）来观察数据中的相关结构。

```
X-=np.mean(X,axis=0)
cov=np.dot(X.T,X)/X.shape[0] #计算协方差矩阵
```

然后做去相关操作,即通过将原始数据（零均值化后的数据）投影到特征基空间（eigenbasis）。

```
U,S,V=np.linalg.svd(cov)
#计算数据协方差矩阵的
奇异值分解（SVDfactorization）Xrot=np.dot(X,U)
#对数据去相关
```

最后一步变换是白化，即把特征基空间的数据除以每个维度的特征值来标准化尺度。

```
Xwhite=Xrot/np.sqrt(S+1e-5) #除以奇异值的平方根，注意到这里加了个1e-5是
为了防止分母是0的情况。
```

PCA白化的一个缺点是会增加数据中的噪声，因为它把输入数据的所有维度都延伸到相同的大小，这些维度中就包含噪音维度（往往表现为不相关的且方差较小）。这种缺点在实际操作中可以通过把1e-5增大到一个更大的值来引入更强的平滑。

3. 初始化

3.1 不要将参数全部初始化为零

几乎所有的CNN网络都是堆成结构，将参数零初始化会导致流过网络的数据也是对称的（都是零），并且没有办法在不受扰动的情況下打破这种数据对称，从而导致网络无法学习。

参数零初始化时，无论输入是什么，中间神经元的激活值都是相同的（任意一个神经元的激活值 $a=f(WTX)$ ，当权重 W 是零向量时， WTX 也是零向量，因此经过激活函数后激活值都相同），反向传播过程中计算的梯度也是相同，每个权重参数的更新因此也是相同的，网络因此失去了不对称性。

3.2 用小的随机数初始化

初始化参数应该非常接近于零（但不全等于零），来打破网络的对称性。初始参数应该是随机且独立的来保证每个参数更新过程是不同的。给每个参数随机赋予一个接近零的值：

```
W=0.01*numpy.Random.randn (D,H)
```

randn方法生成一个零均值，方差为1的正态分布的随机数，同样也可以换用数值较小的均匀分布来产生初始化参数，但在实践中两种方法最终结果差别不大

3.3 方差归一化

用随机初始化方法来初始化参数会导致输出S的方差随输入数量(X或W向量的维度)增加而变大。

独立随机变量和的方差具有以下性质：

$$\text{Var} (A+B+C) = \text{Var} (A) + \text{Var} (B) + \text{Var} (C)$$

$$S = w_1 * x_1 + w_2 * x_2 + \dots + w_i * x_i + b$$

S是多个随机变量的加权和,假设W各维度之间相互独立，随着数据维度增长，S的方差将会线性积累，由于数据的维度随任务的不同，是不可控的，所以我们希望将S的方差做归一化，这只需要对W做处理就可以了：

```
W=numpy.random.randn(n)/sqrt(n) #n是数据维度
```

上面操作是正确的推导过程：

令 $n * \text{Var}(W) = 1$ ，就得到 $\text{std}(W) = 1 / \sqrt{n}$ 。

注意：现在更多的论文中在实际中都在用ReLU函数，针对ReLU推荐：

```
w=numpy.random.randn(n)*sqrt(2.0/n)
```

4. 训练过程中

4.1 卷积滤波器和池化层大小

输入数据最好是2的整数幂次方，比如32（CIFAR-10中图片尺寸），64，224（ImageNet中常见的尺寸）。此外采用较小尺寸的滤波器（例3x3），小的步长（例1）和0值填充，不仅会减少参数数量，还会提升整个网络的准确率。当用3x3的滤波器，步长为1，填充（pad）为1时，会保持图片或特征图的空间尺寸不变。池化层经常用的池化大小是2x2。

4.2 学习率

使用验证集是获得合适LR（Learning Rate）的有效手段。开始训练时，LR通常设为0.1。在实践中，当你观察到在验证集上的loss或者准确率不在变化时，将LR除以2或5后继续跑。

4.3 在预训练的模型上微调

很多state-of-the-arts deep networks的模型被开源出来，这些预训练的模型泛化能力（generalization abilities）很强，因此可以在这些模型的基础上根据自己的任务微调。微调涉及两个重要的因素：新数据集的大小和两个数据集的相似度。网络顶层特征包含更多dataset-specific特征。

	数据集相似性高	数据集相似性低
数据少	直接提取顶层特征来训练线性分类器	比较困难，尝试用不同层的特征训练一个线性分类器
数据多	用较小的学习率微调更多的层	用较小的学习率微调尽可能多的层

5. 激活函数

激活函数用于在网络中引入非线性。sigmoid 与 tanh 曾经很流行，但现在很少用于视觉模型了，主要原因在于当输入的绝对值较大时，其梯度（导数）接近于零，这时参数几乎不再更新，梯度的反向传播过程将被中断，出现梯度消散的现象。

激活函数示意图，图片来自斯坦福 Stanford CS231n

Sigmoid 激活函数

tanh 激活函数

ReLU 激活函数

ReLU 优点：

实现起来非常简单，加速了计算过程。

加速收敛，没有饱和问题，大大缓解了梯度消散的现象。

ReLU 缺点：

就是它可能会永远“死”掉，假如有一组二维数据 $X(x_1, x_2)$ 分布在 $x_1:[0,1]$, $x_2:[0,1]$ 的区域内，有一组参数 $W(w_1, w_2)$ 对 X 做线性变换，并将结果输入到 ReLU。

$$F = w_1 * x_1 + w_2 * x_2$$

如果 $w_1 = w_2 = -1$ ，那么无论 X 如何取值， F 必然小于等于零。那么 ReLU 函数对 F 的导数将永远为零。这个 ReLU 节点将永远不参与整个模型的学习过程。

为了解决 ReLU 在负区间的导数为零的问题，人们发明了 Leaky ReLU, Parametric ReLU, Randomized ReLU 等变体，他们的中心思想都是为 ReLU 函数在负区间赋予一定的斜率，

从而让其导数不为零（这里设斜率为 α ）。

Leaky ReLU 就是直接给 α 指定一个固定的值，整个模型都用这个斜率：

Parametric ReLU 将 α 作为一个参数，通过从数据中学习获取它的最优值。

Randomized ReLU 的 α 是在规定的区间内随机选取的，在测试阶段是定值。

有学者将当前最优的两类CNN网络结合不同的激活函数在CIFAR-10,CIFAR-100和NDSB数据集上做实验，评价四种激活函数的优劣。实验结果表明Leaky ReLU取较大的 α 准确率更好。Parametric ReLU很容易在小数据集上过拟合（训练集上错误率最低，测试集上不理想），但依然比ReLU好。RReLU效果较好，实验表明它可以克服模型过拟合，这可能由于 α 选择的随机性。在实践中，Parametric ReLU 和 Randomized ReLU 都是可取的。

6. 正则化(Regularizations)

以下是几种常用的方通过控制模型的容量来阻止 神经网络 的过拟合（Overfitting）。

6.1 L2正则化

L2正则化也许是最常用的正则化的形式。它可以通过将模型中所有的参数的平方级作为惩罚项加入到目标函数（objective）中来实现。也就是说，对网络中的每一个权重 w ，我们将其项 $\frac{1}{2}\lambda w^2$ 加入到目标函数中，其中 λ 是正则化的强度参数。在惩罚项公式的前面加上 $\frac{1}{2}$ 是很常见的，这样做的原因是因为优化函数 $\frac{1}{2}\lambda w^2$ 求导的时候不至于前面产生一个常数项因子2，而只是 λw 这样简单的形式。对L2正则化的直观的解释是，L2正则化对尖峰向量的惩罚很强，并且倾向于分散权重的向量。

6.2 L1正则化

L1正则化是另一个相关的常见的正则化方式。这里，对于网络中的每一个权重 w ，我们都会加上一个项 $\lambda|w|$ 到目标函数中。L1正则化有一个非常有趣的属性，那就是它会使得权重向量 w 在优化期间变得稀疏（例如非常接近零向量）。带有L1正则化项结尾的神经网络仅仅使用它的最重要的并且接近常量的噪声的输入的一个稀疏的子集。相比之下，最终的权重向量从L2正则化通常是分散的、小数字。在实践中，如果你不关心明确的特征选择，可以预计L2正则化在L1的性能优越。

6.3 最大范数约束

正规化的另一种形式是实施绝对上限的大小在每个神经元的权向量中，利用投影梯度下降来强制约束。在实践中，这对应于执行参数正常更新，然后执行夹紧约束的 $\text{vec}\{w\}$

每个神经元的权向量满足 $\|\text{vec}\{w\}\|_2 < c$ 。典型的 c 值是3或4的订单。有些人报告改进在使用这种形式的正规化。其吸引人的特性之一是网络不能“爆炸”即使学习速率

6.4 Dropout

Dropout是一个极其有效的、简单的并且是最近才被提出的正则化技术作为以上三种正则化方法（L1、L2、最大范数约束）的补充。在训练期间，dropout能够被理解为在一个全连接的神经网络中的神经网络进行子采样，并且仅仅基于输入数据更新网络采样更新的参数。然而,该指数可能的取样数量,网络并不是独立的,因为他们共享参数。在测试过程中，dropout没有被使用。通过集成指数级的所有子网络解释预测的均值。实践过程中，dropout 的比率为 $p=0.5$ 是一个合理的默认值。但是这个值可以在验证数据上进行微调。

最流行使用的正则化技术Dropout

7. 从数字中观察

7.1 从学习率观察

太高的学习率，loss曲线会很奇怪，很容易会出现参数爆炸现象；低学习率，loss下降很慢；高学习率，一开始loss会下降很快，但很容易跌入局部最小值；好的学习率应该平

滑下降。

7.2 放大loss曲线观察。

图2中横坐标是epoch（网络在整个训练集上完整的跑一遍的时间，所以每个epoch中会有多个mini batches），纵坐标是每个训练batch的分类loss。如果loss曲线表现出线性（下降缓慢）表明学习率太低；如果loss不再下降，表明学习率太高陷入局部最小值；曲线的宽度和batch size有关，如果宽度太宽，说明相邻batch间的变化太大，应该减小batch size。

7.3 从精确率曲线观察。

图3中红色线是训练集上的精确率，绿色验证集上的精确率。当验证集上精确度收敛时，红线和绿线间隔过大很明显训练集上出现了过拟合。当两线间隔很小且准确率都很低时，说明模型学习能力太低，需要增加模型的capacity。

8. 集成

在机器学习中，在训练多个学习器并将它们进行组合来使用是一种前沿的学习方法。众所周知，集成方法通常在得到更高的精确性的时候相比于单个学习器是至关重要的。并且，集成方法已经在现实任务中取得了伟大的成功。在实际应用中，尤其是挑战和竞赛中，几乎所有的第一和第二名获胜者都使用集成。

这里，我们介绍几个在深度学习场景中的集成技巧：

8.1 相同的模型，不同的初始化

使用交叉验证决定最优超参数，然后根据最好的超参数集训练多个方法，但是使用不同的随机初始化。这种方法的危险是模型的多样性仅仅取决于初始化。

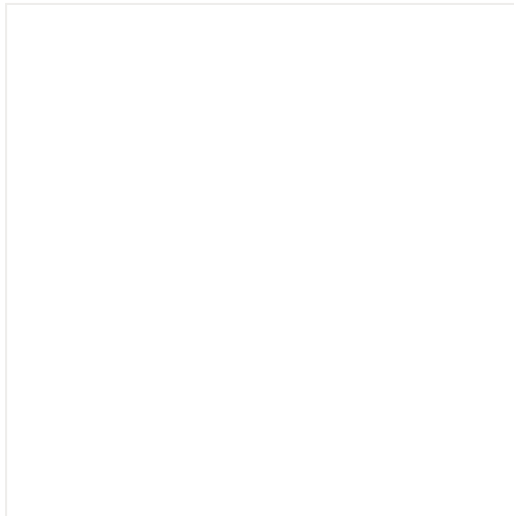
8.2 交叉验证阶段的最优模型的发现

使用交叉验证决定最优超参数，然后选择少量几个效果最好的模型进行集成。这样改善了集成的多样性，但是他也有风险：例如局部最优。在实践中,这可以更容易执行，因为它不需要额外的培训交叉验证后的模型。事实上，你可以直接选择几个最先进的深度模型从 Caffe Model Zoo 执行集成。

8.3 单个模型的不同检查点

如果训练的代价很高，有些人取得了有限的成功在不同的检查点的单一网络随时间(例如在每个阶段)和使用这些形成了一个整体。显然,这受制于某些缺乏多样性，但是在实践中仍然可以工作的很好。这种方法的优点是，非常简便。

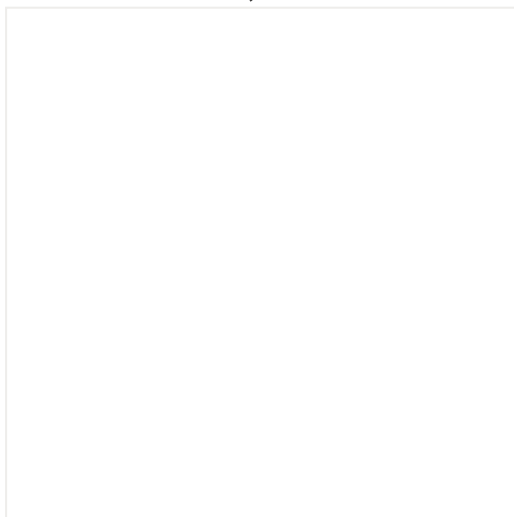
人工智能大数据与深度学习
搜索添加微信公众号: **weic2c**



长按图片，识别二维码，点关注

大数据挖掘DT数据分析
搜索添加微信公众号: **datadw**

教你机器学习，教你数据挖掘



长按图片，识别二维码，点关注

Report