**P18-第二节-2021-机器学习任务攻略**

Framework of ML

Training data:{(),()}

Trainging: step1:functiong with unknown y=(x)

Step2:define loss form training data L()

Step3:optimization =arigL

Testing data{,} 用于测试模型的泛化能力。

General guide 优化模型

1. 当模型的train\_loss过大，1)可能是我们的模型太小，无法接近实际的函数，为此我们可以让模型更加复杂，增加输入的特征，增加层数来。2)如果是我们的优化函数不好，卡在局部最优，或者无法正确找到loss的最小解。

对于1）我们可以尝试比较不同层的training\_error,在深层比底层具有更强能力的基础上，如果低层比高层损失更小，那表明深层的optim没有做好。

模型不够大，优化不够好。

1. Train\_loss小，test\_loss小，则OK。
2. Train\_loss小，test\_loss大，过拟合。

在训练上表示更好，测试集不行。解决方式是增多数据集，采用复杂函数去拟合少量数据，然而其实是一个简单函数所以在测试集表现不佳。还可以采用data augmengtation 通过数据增强获得“新”数据。给模型一点限制，限制模型可以降低模型神经元，共用参数，正则，dropout.使用stop\_early.

为什么kaggle要讲数据集分为test\_public和test\_private,防止偷鸡和模型真正的泛化能力。我们需要将训练数据切割，还可以采用交叉验证(N-flod Cross validtion)和shuffle做的差不多，在交叉验证上我们选择平均的误差。

1. Train\_loss小，test\_loss大， 表明mismach，测试和训练的分布不同。

**P19-2021-类神经网络训练不起来(一)loss minma和saddle point**

Train\_loss一直不降低，此时的gradient接近于0，可以是因为陷入局部最优(local minima)或者鞍点(saddle point), 梯度为0时是我们需要关注的特殊点。

如何分辨是哪个点呢?在该点我们可以采用泰勒展开逼近，梯度是该点函数一阶的导数，而此时特殊点梯度为0，消除了含有一阶导数的项，只剩近似函数值和二阶导数hessian 矩阵，通过二阶导数正负来判断该点是极大值还是极小值，方法和求函数极大极小值类似，通过二阶导数判断，不过采用矩阵论里的hessian表示。如果在该点附近的点有时高有时候低则表明是鞍点。

还是看hession矩阵来判断三种不同情况，求出二阶偏微分后，构造hession矩阵，然后求解矩阵的特征值，若是正定矩阵则为local\_max，负定则local\_min，若特征值有正有负则表示该点是saddle point。此时gradient=0，我们还可以通过看H的情况来判断我们现在如何更新。Hession矩阵是实对称的矩阵，然后通过数学定义一定可以求特征值(eigen value)和特征向量(eigen vector)，沿着负特征值的特征方向我们可以减少损失，这个方向就是我们可以减少损失的方向，因为上面采用泰勒近似通过二次项比较近似和该点的损失。

如果我们延展维度，可能低维的极小值变成了高维的鞍点，有可能现在的local\_minima在高维上其实是saddle point，这样我们在高维上是否可以继续利用saddle point进行优化减少损失，从低维的的local\_minama只是一种假象，我们从高维度还可以对问题进行优化。

**P20-2021-类神经网络训练不起来怎么办(二) batch 和momentum**

在训练中我们将数据分成相同的小batch来计算损失，进行参数的更新，所有batch进行一次遍历算做一次epoch,采用Shuffle打乱数据，导致每一次epoch所拆分的batch内容都是不同的。 两种极端情况，batch\_size=N和batch\_size=1,一个是看完全部资料进行更新，扫描的慢，但是步伐稳健因为综合即正或负，另一个是每看完一个资料就进行更新,每一个资料都会导致走的方向可能不同，但是速度快。

采用不同的batch\_size对MNIST数据处理，刚开始batch\_size的变大并没有让时间变得很长，因为在GPU中我们并行运算，当batch\_size到达并行的顶峰，此时计算时间会非线性增长。而且每epoch的时间是一个减函数，为此我们选择折中，选择较大的batch\_size花费的时间反而小。而batch\_size越大我们发现acc会降低，因为对于一个bach\_size=N来说我们发现每个epoch都是计算相同数据的损失，那么这个损失函数是确定的，二对于小batch\_size来说，每个batch\_size由于里面的数据不同，我们每次的损失函数不同，那么朝着梯度方向走的方向也会不同。在论文在trian上表现相同的大batch\_size和小batch\_size，在test上发现大batch\_size会出现过拟合。大batch\_size让训练速度变快，但是小batch\_size在Optim和generalization上表现更好，采用大的batch\_size去训练深层网络或者大模型，训练速度很快。、

如何采用momentum让梯度下降中的点可以保持惯性不让鞍点卡住，在进行参数更新时，我们让前一步的移动方向和当前梯度反方向融合来更新参数。 初始的momentum=0，随着第一步g的进行，我们m变为上一步的反方向，对于第二步来说m就是第一步的梯度反方向。Momentum随着参数更新，可以返现momentum在每次更新后将这次梯度考虑进来，为此到我们会考虑过去所有的g.

**P21-自动调整学习率**

在训练中当loss降的很低时，此时的梯度是很低的吗？答案是否定的，此时可能因为学习率过大，导致我们的参数在低谷上方左右横跳。在一个简单的两个参数的error convx中我们尝试10e-7的学习率，难以到达收敛点，学习率增大100倍也难以到极值点，为此我们思考在不同情况或者不同的参数我们需要不同的学习率来对参数更新。将之前的学习率以此来面对陡峭和平缓的不同情况。

采用root mean square,把参数更新变为,i编号某一参数，t认为是步骤，=,对之前所有g就均方差,用在 Adagrad中，自适应调整梯度，在error surface中如果线条越多表示线段越陡，线条越少表示越平。

在前面RMS基础上，提出了RMSProp,，参数更新不变，但是的计算变为改变，增加超参数，,通过超参数来决定当前的梯度的重要性，在梯度很大的地方我们可以调小学习率让步伐变小，在RMSProp+Momentum建造出 Adam。但是会出现震荡情况，learning rate scheduling可以解决震荡问题，进行学习率的衰减，让随着时间而衰减，还有一种wake up 表示学习率先从低到高再到低，在15年 residual network首次提出这种方法，还在最新的transformer中使用，刚开始学习率低收集数据，但数据变多以后，影响参数。

,融合momentum，RMS和learning rate scheduling ，momentum提供步伐方向，是开平方出来的正值，只影响大小，两者不会抵消,设置学习率的方法，融合多种方法来计算梯度.

P22 classification

回归中的y是标量，分类中y作为一个类别，我们首先需要让分类转化为数字，我们常会采用独热编码(ont-hot)，将类别用向量表示，其中属于哪一类，哪一行为1。 采用一层来输入类别，多类别需要设置多个神经元来输出。我们需要在最后使用softmax实现多标签分类，softmax通过计算不同y在整体的概率，因为才有用了,，导致各个y之间差异更大，还顺便将y归一化到0- 1，输出概率，然后用MSE计算各项之间的距离。在多分类常用信息熵 Cross-entropy,来计算信息熵，而且在pytorch中我们将softmax两者绑定，没有softmanx层。在计算loss中，由于Cross entropy让我们预测的标签归一化后差别很大，导致error surface上在large loss的地方也可以看到梯度，而不会被卡住。

**P23宝可梦，数码宝贝分类器**

基于前面的知识，我们继续进行用一个函数来进行分类问题，我们先随意sample五个例子观察两者直观的区别，然后我们采用边缘检测(edge detection)来进行进行描边，发现宝可梦的线条比较简单较少，而数码宝贝一般存在人形导致图形的线条比较多，如果我们设置一个函数用来分辨图片的线条复杂程度，根据这个结果是否超过边界来完成分类。我们设之函数f中只有一个参数H，然后H的复杂程度从1-10000，模型的复杂程度很高表示模型的选择有很多，比如这里的H选择有一万种，接下来我们来设置函数f的loss，我们认为数据还是又特征x和类别y构成，为此我们采用MSE加上阈值h，设置来计算Error rate。继续设置traning examples,如果基于理想情况，我们认为可以收集到所有的宝可梦和数码宝贝信息，基于此我们可以通过暴力方法求出最合适的h来让loss最小。但是实际情况我们难以收集所有资料，我们永远只是一部分的资料，我们收集的资料要满足iid(independently and identically distributed).我们理想状态下,现实情况下我们所用的训练资料很有限，只能实现,我们期望train出来的h可以在理想的h上误差很小。通过绘画直方图来看两个类不同分布，发现和预期一样，宝可梦线条比较少，然后我们找到了最好的的=4824，=0.28，通过采样野生数据后计算得到=4727，=0.27,出现train的loss反而低，是因为训练的数据少，但是我们使用测试，发现=0.28,再次采样后，这次数码宝贝很多，导致我们的=3642，=0.20，但是在上，=0.37，这是因为我们收集的资料不均匀，导致训练模型误差很大，我们希望,我们希望这种误差很小，所以我们希望 ,表示两个数据集分布接近,我们要找一个好的，这种方法和模型无关，和loss也无关，我们也可以用在深度学习中。找一个好的，我们先将某些h导致误差大于，说这些h破坏了一些资料，不断破坏下去，剩下就是好的数据。 根据Hoeffding不定式可以求泛化误差上界p=,如果h变小我们可以得到的好的更多。在这个例子中，如果我们增多数据，我们可以让训练到坏的资料的概率变小，我们可以通过降低模型的复杂程度或者增加训练数据的数据量，比如现在的大模型就需要庞大的数据来支撑,评定数据好坏是，因为用它训练出来的和实际的误差其实很小。

P24,25梯度下降例子

采用帝国时代来表现如何随机生成一个参数然后朝着梯度方向方向前进的例子。

采用我的世界表示采用g+momentum的情况为什么会让走向loss更大的情况。

P26 optimization

还是复习，优化做的就是找到最小的损失函数，on-line一个x一个x训练，对于数据需要马上处理， off-line表示我们要拿到全部的全部的数据，复习前面的SGD，会陷入鞍点，需要配合动量来更新。Adagrad加分母，考虑之前全部的梯度算平均。RMSProp考虑超参数，认为这一步对于参数的更新影响更大。Adam将SGDM和RMSProp考虑到一起。顺便讲解都是14年出来的优化算法，后续的大模型BERT,yolo,ResNet,BIG-gan都是SGD训练出来的。 MEMO用ADam.

Adam快，不稳定，泛化误差大。SGDM稳定，泛化误差小，收敛的值更小。SWATS算法，训练前期用Adam，后期用SGDM。