单隐层神经网络调试步骤及笔记

@Moplast

2017-02-24

**0 初始化**

**0.1 输入与输出(以 footballshuju.mat为例)**

**0.1.1 输入**

* 3 个特征（维度）

|  |  |  |
| --- | --- | --- |
|  |  |  |
| 1, 2 | 1, 3 | 2, 3 |

观察了一下几个维度的样本点，但是没看出什么明显的所以然来。所以用线性激活函数是不太可能了，应该选用tanh之类的非线性函数。

**0.1.2 输出**

* 2 个值

一开始看输出是2个值，以为是回归问题。但是仔细观察一下，会发现，输出数据是有pattern的。比如在footbalshuju.mat中，第一个输出只有几种值 2,4,8,16,32，第二个输出只有0.2,0.3,0.4,0.5,0.6,0.7,0.8。这样就把回归问题转化成了分类问题。

但这牵扯到一个问题，是把两个输出合并，一共输出5\*7=35个值，还是分别训练两个分类器？35个值的输出似乎过于冗余，而且为了训练35个输出的神经网络，隐层神经元数量会比较多，计算量较大；但是如果分别训练，很有可能会忽略这两个输出之间的联系。这是需要考虑的问题。

暂时先不考虑交叉验证和测试集的问题。等模型出来，再分割样本。

**0.2 选择参数**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 训练 | | | 初始化参数 | |  | |
| 隐层数量 | - | 1 | **正则化参数** | Lambda | | 1 |
| 输入特征数量 | Input\_layer\_size | 3 | **随机初始化参数** | Epsilon\_init | | 0.12 |
| 隐层神经元数量 | Layer\_input\_size | 8 |  | | | |
| 样本输出数量 | Num\_labels | 5 / 7 |
| 最大迭代次数 | Max\_iter | 50 |

**0.3 载入数据**

没什么特别的，只是为了训练不同列的输出，需要手动改一下file\_col……当然也可以修改这部分……

1. **数据预处理**
   1. **输出值 转 分类**

写个函数Numerial2class把数值的输出值 转成 分类的类型。

比如2变成类1, 4为类2,8为类3 …

返回值是转化好的类向量，和映射表(因为是顺序索引， 所以不返回矩阵了，直接向量也可以，当然返回矩阵更直观一些)

* 1. **特征mapping**

暂时没什么想法。

但是有一个优化思路，就是根据这几列的物理或逻辑关系来形成（映射）新的特征值。

输入：1列 信噪比PSNR；2列 码率；5列 测量率残差能量。

输出：3列 量化参数QP；4列 测量率SR。

由于我的重心放在了训练神经网络上，就没有考虑这部分内容。但是我留了类似接口的函数，可以在FeatureMapping() 中补充特征，然后打开mapping\_option，不需要再改动代码了。

**1.3 特征normalization**

每个特征维度不太一样，第一列的值大概都在30左右，第二列则方差挺大，上千的也有，数十的也有，第三列几乎不变。对这种情况需要对特征处理一下。

写了个feature normalization的函数，运用的是最基本的std+mean方法。公式如下：

其中std为列的标准差，mean为列的平均值。

**注意：**特征normalization非常重要，尤其是在迭代次数不那么大的时候。如果不打开normalization\_option，很有可能样本集和测试集的正确率都非常低。

**2 初始化参数和代价函数**

**2.1 初始化参数**

分别初始化输入层到隐层和隐层到输出层的参数。用了随机取参的方法。

**2.2 代价函数**

计算代价函数和梯度。

**这部分实现过程在代码里写的比较详细，这里简单讲，基本略过。其实也就是理论的东西，和你的资料差不多，当然，这也是最重要的实现部分。**

**Step 1:** 前向传播算法计算J

J即代价，用的是交叉熵代价函数。

激活函数为sigmoid。实际上sigmoid非常不好，但是tanh还没有学，在计算lnh的时候会出现复数，暂时还不知道怎么解决。

**Step 2:** 后向传播算法计算gradient

倒推误差，按照公式一步步来……比较懒就略过这部分公式了……

**Step 3:** 加上正则项

按照公式即可……

**3 训练神经网络**

* 用fmincg函数训练神经网络，输入参数为最大迭代次数，自己写的代价函数，随机的初始参数。即可得到训练后的参数和迭代后的代价。

由于没有选用梯度下降法，就不需要学习速率了，貌似fmincg会自动选择最优学习速率。

**4 预测训练集**

跑了一下训练集，发现随便输入的参数跑出来效果还不错。但还没有试过分开数据集测试，很有可能过拟合。接下来就是漫长的调试过程。这里先记录一下初始值和效果。

Initial cost: 3.4左右

Trained cost: 0.83左右（和初始值有关）

Accuracy: 95-100% （和初始值有关）最好的一次居然100% 惊呆了(ﾟДﾟ≡ﾟДﾟ)

Lambda: 1

Max\_iter: 50

Hidden\_layer\_size: 8

**5 预测测试集**

一开始跑测试集，效果都非常差，但是又找不到原因在哪儿。

那么就来尝试一下学习曲线的绘制和交叉验证曲线的绘制，看看到底是过拟合还是欠拟合，以及正则化参数该怎么选取比较好。

**6 调试参数**

**6.1 分开数据集**

由于每个种类的数据集都只有一块，就初步按照0.3,0.4,0.3的比例分配训练集，交叉验证机和测试集了。当然别的比例也可以，改一下函数即可。

* 注意一下，不是简单的直接按照索引前中后分，因为我看到有些数据集不是乱序的，而是按照某个特征相同的放在一起，这样会导致预测正确率降低。所以要打乱然后重组再分数据集。函数里用randperm()来实现打乱。
* 还需要注意的是，打乱后分组所输出的值，一定要包括所有种类的输出，否则训练半天根本就没那个类型的输出，就不会预测到那个空缺的类型。函数里已经考虑到这一点，这就是为什么要用一个while大循环计算三个样本集输出类型的个数，当然，这种概率应该不大。

**6.2 学习曲线**

学习曲线会输出随着训练集样本数量的增加，训练集和验证集在训练出的参数下代价值的变化。是一个很好的过拟合，欠拟合的指示。

用初始化的数据，发现数据一方面过拟合，一方面又欠拟合，我就在特征mapping上想了很久，后来发现只是某个地方输入数据集错了= =

**6.3 Lambda曲线**

Lambda曲线会输出 不同的lambda ，训练集和验证集在训练出的参数下代价值的变化。是很好的lambda值选取建议。一般选取两个最靠近的值即可，不同数据集可能需要选取不同的lambda，不过都差不太多。

**6.4 具体调试**

这个感觉比较靠经验，虽然我应该算什么经验都没有的那种……我具体只调整过几个参数，依次是

|  |  |  |
| --- | --- | --- |
| 参数 | 增大效果 | 减小效果 |
| lambda | 过拟合时使用，可能会让训练集正确率变低，但是可以有效提升测试集正确率 | 欠拟合时使用，可能会让训练集正确率变高。 |
| Max\_iter | 训练集和测试集正确率都不高，且final\_cost也没有比initial\_cost要小多少，就可以尝试增加迭代次数，不过会让训练时间变长。 | - |
| Normalization\_option | 前面说过，实现normalization非常重要，否则要花更多的时间迭代。 | - |
| Hidden\_layer\_size | 太大不好，这个我也不太清楚为什么，一般我都取6-9，效果不会差太多 | 太小不好，我觉得应该要大于输入特征数量。 |

**7 完善方向**

**#1**

之前提到过的，最好还是线性输出，而不是通过分类输出。我现在这样做，总觉得破坏了值的意义。

**#2**

同时输出两个值，因为这两个值可能互相之间有联系。或者说同时将两个值mapping成一个，当然不知道这有什么意义，具体还是要根据数据本身的意义出发。

**#3**

将单隐层扩展到多层。实现是可以实现的，但是一方面需要更多时间，一方面感觉就目前正确率来说，没什么必要。

**#4**

结果正确率和随机初始化值相关性非常大。多次运行程序就可以发现，正确率变化幅度比较大，因为随着初始化值不同，代价收敛到的很有可能只是局部最小值，而不是全局最小值，这就代表了输出的参数不是最好的。不过这个的解决方法，就是多运行几次程序，把正确率高的参数保存下来，以供以后使用。