**中山大学数据科学与计算机学院**

**移动信息工程专业-人工智能**

**本科生项目报告**

**（2017-2018学年秋季学期）**

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

课程名称：**Artificial Intelligence**

# Project最终结果展示

2. 最终结果

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 二元分类（F1） | 多元分类  (平均准确率) | 回归（RMSE） | 二元分类排名 | 多元分类排名 | 回归排名 | 总排名 |
| 0.91078697422 | 0.650293150311 | 40.7048743083 | 17 | 2 | 1 | 2 |

1. 个人工作

|  |  |
| --- | --- |
| 时间 | 工作 |
| 15周 | 分别使用LR，PLA，BPNN测试了二元分类。 |
| 使用BPNN测试了回归。 |
| 16周 | 优化回归，并对原数据集进分析，对数据进行取舍，选取训练集和划分验证集。 |
| 修改回归使用的BPNN，将其输出修改为二元分类，更换LOSS Function为逻辑回归的交叉熵 |
| 开始寻找BPNN的优化方式，并加以实践，最终选用Drop和Adam进行优化。 |
| 17周 | 尝试使用PCA的方式进一步处理二元分类的数据，继续优化二元分类的BPNN，并进行调参，并尝试去除噪声。 |
| 开始分析队友提供的多元分类处理过的One-hot、TF、TF-IDF数据，以各种方式尝试降维，通过SPSS分析单词与答案之间的关联性，主要测试了PCA中的偏相关，因子分析，相关性，独立样本检验，卡方检验。最终获得了5\*3种降维数据，并使用BPNN测试不同降维方式的最终效果，测试使用LR代替BPNN。 |
| 回归部分使用Dropout和Adam测试优化效果，并进行参数的调整。 |
| 18周 | 对多元分类数据进行PCA降维，使用Adam优化多元分类的BPNN，尝试多元分类不同维度的降维效果。 |
| 修改回归的数据集，增加属性(离散化连续值)，进行优化调参。 |
| 19周 | 整理代码，继续优化调参，完成实验报告。 |

# 工作流程

1. 算法简介

* **PLA(感知机算法)** 是根据输入的特征向量x对其进行二分分类的线性分类模型，目标是求得一个能够将训练集正实样本和负实样本完全正确分类的超平面。预测函数为：

其中样本为，我们令其中的为阈值，，即为上述预测公式。

在每一次迭代中不断更新W，最终目标是得到一个可以分割全部样本的W，当我们遇到一个错误的样本的时候就更新W，更新公式为，其中为学习率为样本的标签。

* **Logistic回归** 属于广义线性回归模型，它与线性回归模型的形式基本上相同，logistic回归则通过Sigmoid函数将ax+b对应到一个隐状态p，然后根据p决定最终的预测值，预测函数为：

其中的与含义与PLA中相同，逻辑回归使用交叉熵作为代价函数：

逻辑回归使用梯度下降法进行更新，通过对损失函数求导可以得到梯度的方向，之后沿梯度下降的方向迭代多次求解损失函数极小值，更新公式如下：

逻辑回归利用Sigmoid函数将最终的判别结果从映射到，使得可以从概率的角度来进行判决，从而达到软分类的效果。但是最为广义线性回归其依然不能解决线性不可分问题。

* **反向传播神经网络BPNN** 当神经网络仅由一个神经元构成即为类逻辑回归算法。所谓神经网络就是将许多个单一神经元联结在一起，一个神经元的输出就可以是另一个“神经元”的输入。神经网络最左边的一层叫做输入层，最右的一层叫做输出层。中间所有节点组成的一层叫做隐藏层（隐藏层也可以有多层）。一次BPNN的迭代中包括前向传播和反向传播。
  + **前向传播：**每一层的神经元计算出该层各神经元的输出并向下一层传递直到输出层计算出网络的输出结果，前向传播用于计算出网络的输出，不对神经网络的参数进行调整。

我们用表示第层的第单元的输入加权和,表示第层的第单元的输入输出，那么有

其中 表示的是第层的第单元与第层第单元之间的连接参数，为第层的阈值。

* + **反向传播：**用于训练时权值和阈值的调整，训练的结果和实际结果比较会得到一个偏差，并且把偏差一级一级向前传递，使用偏差进行权值和阈值的调整。针对第层的每一个节点 ，计算出其残差 ，该残差表明了该节点对最终输出值的残差产生了多少影响，对于输出层可以直接算出网络产生的激活值与实际值之间的差距，对于隐藏层基于第 层节点的残差的加权平均值计算，反向传播算法可表示为以下几个步骤(求导部分省略)：
    - 对于输出层计算残差：
    - 对于其他各层有计算残差：
    - 计算终需要的偏导数值：
    - 更新权重参数：

上述式中为Loss Function，在回归问题中我们使用方差代价函数，在二元分类中使用交叉熵；为学习率，为权重衰减参数（正则化项系数），用于控制公式中两项的相对重要性；为激活函数的倒数。

* **Adam算法**Adam 是一种可以替代传统随机梯度下降过程的一阶优化算法，它能基于训练数据迭代地更新神经网络权重。Adam 通过计算梯度的一阶矩估计和二阶矩估计而为不同的参数设计独立的自适应性学习率。算法计算了梯度的指数移动均值，超参数 beta1 和 beta2 控制了这些移动均值的衰减率。Adam 算法很容易实现，并且有很高的计算效率和较低的内存需求。根据论文2015 年 ICLR 论文（Adam: A Method for Stochastic Optimization）**[2]**中提出的Adam算法伪代码如下：

|  |
| --- |
| **Algorithm 1:** *Adam* |
| **Input:**  Stepsize  **Input:** Exponential decay rate for the moment estimates  **Input:**  Stochastic objective function with parameters  **Input:**  Initial parameter vector  (Initialize 1st moment vector)  (Initialize 2st moment vector)  (Initialize timestep)  **while**  not converged **do**    (Get gradients)  (Update biased first moment estimate)  (Update biased second raw moment estimate)  (Compute bias-corrected first moment estimate)  (Compute bias-corrected second raw moment estimate)  (Update parameters)  **end while**  **return** |

实现部分：此处展示在matlab中更新w时使用Adam的部分

|  |
| --- |
| %%-----------------------更新w----------------  %隐藏层  **for** i **=** layers**-**1 **:** **-**1 **:** 2  g**=**out**{**i**-**1**}'\***delta**{**i**}** **;** %计算梯度  m**{**i**}** **=** beta1 **\*** m**{**i**}** **+** **(**1 **-** beta1**)** **\*** g**;** %更新m  v**{**i**}** **=** beta2 **\*** v**{**i**}** **+** **(**1 **-** beta2**)** **\*** **(**g **.\*** g**);**%更新v  m\_**=**m**{**i**}/(**1**-**beta1**);** %偏差m更正  v\_**=**v**{**i**}/(**1**-**beta2**);**%偏差v更正  w**{**i**}** **=** w**{**i**}** **+** step**(**i**)** **\*** **(**m\_ **./** **(**sqrt**(**v\_**)+**1e-8**)** **+** lambda**(**i**)** **\*** w**{**i**});**%更新w  theta**{**i**}** **=** theta**{**i**}** **+** step**(**i**)** **/** train\_row **\*** sum**(**delta**{**i**},**1**)** **;**%更新b  **end** |

* **DorpOut算法** 过拟合是神经网络中的一个常见问题，模型只学会在训练集上分类。Dropout通过使其他隐藏单元存在不可靠性来防止共拟合。Dropout的思想是训练整体整个神经网络，并平均整个集合的结果。Dropout以概率p舍弃神经元并让其它神经元以概率q=1-p保留。每个神经元被关闭的概率是相同的。输入时我们使用概率p处理上一层的输入，伪代码如下：

|  |
| --- |
| **Algorithm 2:** *Dropout in train* |
| **Input:**  Dropout rate  **Input:** Output value of before layer  (Initialize Bernoulli Distribution matrix)  (Use dropout deal with output value of before layer)  **return** |

此时我们训练时前向传播公式改为：

在测试时修改前向传播公式为：

此处展示在前向传播中隐藏层中的实现：

|  |
| --- |
| %隐藏层前向传播  **for** i**=**2 **:** layers**-**2  drop **=** gpuArray**.**rand**(**1**,**Node**(**i**+**1**),** 'single'**)** **<** p**;** %生成伯努利分布矩阵  out**{**i**}** **=**1 **./** **(**1 **+** exp**(-**out**{**i **-** 1**}** **\*** w**{**i**}** **.\*** drop **+** theta**{**i**}));**%训练集  out\_val**{**i**}** **=** 1 **./** **(**1 **+** exp**(-**out\_val**{**i **-** 1**}** **\*** w**{**i**}** **\*(**1-p**) +** theta**{**i**}));**%验证集  **end** |

* **卡方检验（chi-square test）** 主要是比较两个样本率(构成比)以及两个分类变量的关联性分析。其根本思想就是在于比较理论频数和实际频数的吻合程度或拟合优度问题。实际值与理论值之间的偏离程度就决定卡方值的大小，卡方值越大，越不符合；卡方值越小，偏差越小，越趋于符合，若两个值完全相等时，卡方值就为0。此处使用卡方检测筛选单词进行降维。

首先引入四格表：（此处将多元分类转化为3个二元分类，此处以LOW为例）

|  |  |  |  |
| --- | --- | --- | --- |
| 组别 | 评价为LOW | 评价不为LOW | 合计 |
| X单词出现 | a | b | a+b |
| X单词不出现 | c | d | c+d |
| 合计 | a+c | b+d | a+b+c+d |

计算理论数（TRC）:; RC是表示第R行C列的理论值，nR为理论数同行的合计数，nC为与理论数同列的合计数，n为总例数。

卡方公式如下：

其中T为TRC公式算出来的理论值，A为真实值，此处使用matlab中卡方检测函数实现，通过卡方检测，获取卡方检测中LOW，MID，HIG卡方值最高的前300个单词，作为降维后的训练集。

* **PCA 主成分分析** PCA的思想是将n维特征映射到k维上（k<n），这k维是全新的正交特征。这k维特征称为主成分，是重新构造出来的k维特征，而不是简单地从n维特征中去除其余n-k维特征。PCA的做法是将数据从原来的坐标系转化到新的坐标系中，根据数据本身的特征来设计新的坐标系。数据变化的主方向也就是协方差矩阵的主特征向量，以此类推之后是数据变化的次方向次特征向量。使用协方差矩阵，我们可以使用新的坐标系来表示原数据，用新的坐标表示的原数据称为旋转后数据，数据的主方向就是旋转后数据的第一维，如果我们要降到第k维，只需要前k个数据变化的主方向。

降维作用是：数据在低维下更容易处理、更容易使用，相关特征，特别是重要特征更能在数据中明确的显示出来，去除数据噪声，降低运算开销。

|  |
| --- |
| **Algorithm 3:** *PCA* |
| **Input:** Data  **Input:** *N* Target dimension  Remove the average  Calculate the covariance matrix  Calculate the eigenvalues and eigenvectors of the covariance matrix  Sort eigenvalues  Reserved N eigenvectors  Transformed Data into new space constructed by the N eigenvectors  **return** newData |

此算法可以再matlab中直接调用PCA函数，也可以手动求协方差提取特征向量排序，然后构成变换矩阵实现降维。这个其实在matlab中直接显示不是很复杂，主要都是矩阵操作，参考文献[1]带有matlab代码。

* **DBSCAN用于去除噪声** 密度聚类是一种基于密度的空间聚类算法，将具有足够密度的区域划分为簇，并在具有噪声的空间数据库中发现任意形状的簇，它将簇定义为密度相连的点的最大集合。DBSCAN 需要两个参数：eps和形成高密度区域所需要的最少点数 (minPts)，它由一个任意未被访问的点开始，然后探索这个点的eps邻域，如果eps邻域里有足够的点，则建立一个新的聚类，否则这个点被标签为噪声点。这个点之后可能被发现在其它点的 eps邻域里，而该eps邻域可能有足够的点，届时这个点会被加入该聚类中。

通过密度聚类的方式我们可以标记出噪声点，同时我们不需要考虑到这个密度聚类最终会聚成多少个类别，我们只需要通过其获得噪声点然后去除即可，那么我们的算法只需要找出所有的中心点和边界点不需要判断类别。

此时算法转化为: 遍历所有数据点，如果其能形成一个新的聚类，将其eps临域中的所有点标记为已访问，遍历完成后筛选出未访问的数据点，此时这些数据点就是噪声点，然后去除这些噪声点即可。

|  |
| --- |
| **Algorithm 4:** *DBSACN to find outliers points* |
| **Input:** Data  **Input:** *eps*  **Input:** *minPts*  *v[]=0* (Flag of point i visited times)  *newData* (Use to save data that remove the outliers points)  **For each** point *P* **in** *Data*  *N*= Find neighbors (*P*)  **If** *N >=* *minPts*  **For each** point *P’* in *P* eps  v*[P’]++*  **end**  **end**  **end**  **For each** point *P* **in** *Data*  **If** *v[P]>0*  Add *P* to *newData*  **end**  **return** newData |

* **尝试的各算法的最佳表现**
  + **二元分类：**

|  |  |  |  |
| --- | --- | --- | --- |
| **算法** | **参数** | **训练集处理与划分** | **测试集F1** |
| PLA | 学习率为0.1  学习率衰减系数为0.9999  迭代10000次 | 随机划分12000个作为验证集，其余为训练集 | 因为验证集效果不佳没有提交 |
| LR | 学习率为0.1  迭代1000次  正则化项系数0.02 | 随机划分12000个作为验证集，其余为训练集 | 0.710827608767 |
| BPNN | 4层BPNN 隐藏层为200，200  学习率 0.01  正则化项系数0.001 | 将type列离散化  随机划分12000个作为验证集，其余为训练集 | 0.7076736741932693 |
| BPNN+  Adam | 4层BPNN  隐藏层节点为200，200  学习率 0.001  正则化项系数0.001  Adam参数使用论文推荐参数 | 将type列离散化  随机划分12000个作为验证集，其余为训练集 | 0.82619308139 |
| BPNN+  Adam+  DropOut | 4层BPNN p=0.4  隐藏层节点为300，300  学习率 0.001  正则化项系数0.005  Adam参数使用论文推荐参数 | 将type列离散化  随机划分12000个作为验证集，其余为训练集 | 0.858904908742 |
| BPNN+  Adam+  DropOut+  DBSACN  去除噪声点 | 4层BPNN p=0.4  隐藏层节点为300，300  学习率 0.001  正则化项系数0.001  Adam参数使用论文推荐参数 | 将type列离散化  随机划分12000个作为验证集，其余为训练集 | 0.857419980601 |

* + **多元分类：**

|  |  |  |  |
| --- | --- | --- | --- |
| **算法** | **参数** | **训练集处理与划分** | **测试集平均准确率** |
| 3个LR | 学习率为0.1  迭代10000次 | 使用卡方检测降维获得3个数据集，每个数据集300列，分别用于3个LR  随机划分10000个作为验证集，其余为训练集 | 0.624964137751 |
| BPNN | 4层BPNN  隐藏层节点为200，200  学习率 0.01  迭代5000次  正则化项系数0.001 | 使用卡方检测降维获得300列数据集卡方值最高的前300个都有的词。  随机划分10000个作为验证集，其余为训练集 | 0.599386574152 |
| BPNN+  Adam | 4层BPNN  隐藏层节点为300，300  学习率 0.001  迭代20000次  正则化项系数0.001  Adam参数使用论文推荐参数 | 使用卡方检测降维获得300列数据集卡方值最高的前300个都有的词。  随机划分10000个作为验证集，其余为训练集 | 0.615212239167 |
| BPNN+  Adam+  PCA | 3层BPNN 迭代150000次  隐藏层节点为30  学习率 0.001  正则化项系数0.001  Adam参数使用论文推荐参数 | 使用PCA降维，获得1789维的数据  随机划分10000个作为验证集，其余为训练集 | 0.650293150311 |

* + **回归：**

|  |  |  |  |
| --- | --- | --- | --- |
| **算法** | **参数** | **训练集处理与划分** | **RMSE** |
| BPNN | 4层BPNN  隐藏层节点为300，300  迭代2000次  step = [0.01, 0.015,0.0200];  lambda=[0.003 0.005 0.009];  Leaky ReLU：alpha=0.25 | 数据集独热编码（年、季节、月、日、星期、小时、天气、温度（2.5/级）），额外标记圣诞节及12的上中和下旬。  使用2011.9-2012.8作为训练集；  同时将2011年部分按照均值比较放大1.4倍使其接近的值 。 | 39.9022559131 |

1. 调参过程

* **二元分类**

二元分类算法使用了Adam算法优化和Dropout，Adam有自适应学习率的功能，所以我只有略微的调整正则化项和Dropout比例防止训练集过拟合，调整过程由我每次迭代输出在训练集和验证集上的F1和准确率作为标准，如果出现过拟合现象就直接加大正则化项或者Dropout的p，防止过拟合，最终参数类似二分法的方式，从不会过拟合到出现严重过拟合的区间选取，这部分因为后期二元分类移交队友，没有继续调参，只是初略调参。

* **多元分类**

多元分类最终使用PCA降维和Adam算法优化，Adam其学习率自适应，不需要人为干预，而且由于多元分类中数据基本不会出现过拟合所以没有调整正则化项的必要，主要调整的参数是PCA降维之后维数，这里我们使用一步步增加维数的方式进行调参。

PCA保留的维度越少就会导致一些细节丢失，但是到了一定维度之后保留PCA降维导致的信息丢失问题也就会缓解，之后在增加维度只会增加自己的计算复杂度，这里我测试的时候因为跑2700维太吃力，所以也没有继续测试，考虑到样本中最长一句话也才1200词多，所以我认为PCA降维的维度没必要再次增加。

* **回归**

回归的调参主要还是对于BPNN学习率和正则化项的调参，以及数据的预处理。

对于回归的参数调整，最有成效还是对于数据集的优化处理，对于参数的调整只能是微调。

1. 尝试原始数据集合并删除索引列和日期列，进行尝试。
2. 尝试在1）的基础上独热编码化时间与天气，发觉验证集的值基本都在一个均值，没有日期差别。
3. 在2）的基础上对比了之前LAB6中的训练集，发现这个训练集少了属性，于是将年、月、日、星期、季节，通过日期列计算出来，然后进行独热编码。
4. 在3）的基础上对照之前LAB6中做法分离各层学习率和正则化项，调整不同层的学习速度。使得训练集的RMES降到40左右，在验证集上降到70左右，测试集RMSE最优为67
5. 在4）的基础上分析了训练集，发现训练集中有每月统计，然后对比了11年和12年的比例，发现比例过于悬殊最多有将近3倍的差值，考虑舍去一些训练集，然后选定了1年（11-9到12-8），此时我们可以使用12年的9-11月这3个月作为验证集，使用在3个验证集同时又低RMSE时选取其作为结果。同时对比11年和12年的前11月总和差距，于是将11年的数据放大1.4倍处理。通过此时调整训练集RMES降至25左右，验证集65左右，测试集62左右。
6. 一开始考虑到12月只有圣诞节所以没有增加节日，在听了Pre之后考虑增加节日，一开始只增加圣诞节当天， 独立使用11年12月作为验证集测试圣诞节效果，发现实际上圣诞节前和圣诞节后拟合效果不好，只有圣诞节当天还行，增加平安夜处理，和对于圣诞节周的标记（tirck）。同时考虑将每个月增加划分成上旬、中旬、下旬共36列，进行再次训练划分完成后，RMSE为43，测试发现划分成上中下旬没有必要花划分那么多，只要有12月下旬在验证集上拟合效果就足够了，所以删除去35列，减少算法运行负担，当然最好还是加上去这是为了rank。
7. 考虑到温度湿度风速也是有略微影响，所以讲温度等分别离散化，然后在独热编码，实际测试中，湿度风速没离散化后没什么影响，只会减速迭代，从其和训练集的结果的相关性检测也可以看出影戏不大，最终保留离散化温度，然后使用正则化项进行微调效果。此时RMSE为39.9。

此处说明一下，这里为什么没有使用Adam优化回归，主要问题就出现在本身离散化之后已经有过拟合的问题，使用Adam之后只会导致过拟合更加严重，回归问题关键不是在数据拟合，而是在于数据预处理，对于拟合没有什么要求。对于预防过拟合使用DropOut测试发现因为DropOut的原因会导致其在原数据集上拟合效果下降，但是在验证集上并没有多少的防止拟合效果，反而会加重算法的运行负担。

1. 数据集分析

* **二元分类**

二元分类数据集没有给出每一列的意义。统计完每一列特征之后我们可以知道第二列是离散值，一共5中类别TypeA-E；倒数第3列和倒数第4列也是离散值取值分别为（0,3）和（0,1），所以此处现将其独热编码，其余列都是连续值。我开始考虑到这个数据集中存在噪声，所以需要去除噪声，此处使用DBSCAN标记噪声点的方式进行噪声的去除，通过去除噪声点对于使用BPNN进行测试，测试集基本没有提升，在观察测试集数据之后发现这个数据集应该是没有太多噪声的，因为其分布和训练集类似，对于二元的其他处理由队友完成。

* **多元分类**

多元分类的训练集有90048个不重复的词(去除标点)，有62252个样本，如果只是进行单纯的转成矩阵表示那会因为数据过大而无法使用，而且其中还有非法词同时也有一些网址等没用的东西，这些就需要过滤掉，不过实验中我们实现的是只提取单词并没有过滤非法词。

对于多元分类的数据集降维是必须的，降维方法主要是提取关键词，或者是用word2vector和PCA这两种改变坐标系的方式，将句子映射到全新坐标系实现降维。对于前者我们可以使用的最简单的一种是相关性检测，也就是计算当前单词与标签的相关性，从相关性检测可以衍生出类似卡方检验等检测相关的变种。对于后者的word2vector，将所有单词使用词向量来表示，然后将整句话构成一个新的向量，但是是考虑到word2vector的最佳用法应该是将句子转换为矩阵在配上CNN，而我没有实现CNN，所以没有考虑使用word2vector降维，我们转而使用PAC降维，PCA降维主要现实使用Matlab中的PCA函数，不过PAC降维也会带来缺点主要就是信息的丢失，所以降维过程中要尝试不同维度的降维。

* **回归**

这个数据集中有缺失值，但是都是在小时那边能够明显补上的，特征列有索引、日期、小时、天气 、温度体感湿度、以及风速。首先索索引列肯定是不需要的，之后对比之前lab6的数据集，我们可以发现少了年份、季度、月等数据，所以此处需要从日期列中分离出来。分离之后转换成独热编码能使模型更好的拟合。

观察每一天的使用情况可以发现如下图此时区分工作日和周末还是很有必要的。

|  |  |
| --- | --- |
|  |  |

在数据集中额外考虑使用日期添加节假日，因为节假日的使用状况和和其星期无关，同时这里额外考虑假节日的放假时长和影响范围，比如圣诞节不止会影响当天，前一天的平安夜也会都到影响，元旦前一天晚上也是类似情况。在针对11年12月的数据进行分析我们能发现在周末的时候温度天气对于使用量也有影响，如果是低温再加之天气不好也会造成使用量下降，最终考虑将温度离散化，并进行独热编码。

通过总体分析我们按月变化如下图

|  |
| --- |
|  |

通过比较每个月的变化量我们可以发现12年的使用量基本在11的1.4-1.5倍左右，如果我们全部训练集会造成预测值出现偏差，所以决定舍去部分训练集，选择一年的区间作为训练集，最终选择11/9-12/8作为训练集，然后12/9-12/11作验证集，这里为了消除11年较低造成的影响，将11年的使用量放大1.4倍在进行训练。

1. 集成学习方法(AdaBoost)

对于AdaBoost我只参与了使用单层决策树+AdaBoost部分，后续因为二元分类移交队友，所以AdaBoost交由队友继续优化，此处概述我参与的部分。

AdaBoost的目标是通过一个弱分类器构建一个强分类器，AdaBoost基本流程是：对训练集中的每个样本，赋予其一个权重W，初值为1/N。首先在训练数据上训练出一个弱分类器并计算该分类器的错误，根据训练结果调整每个样本的权重，以及设置这个分类器的权重，重复上述过程迭代

AdaBoost中的错误率，m表示第m次迭代：

上述式子中，错误率就是被误分类样本的权值之和除总数，此时分类器权重:

上式中我们可以看出时；上式中我们可以看出时，之后调整权重：

* + 如果样本分类正确：
  + 如果样本分类错误：

**单层决策树** 单层决策树使用的是单个特征，然后寻找这个特征中的一个特征，以其作为分界线，将这个特征分成两部分一段为+1，一段为-1，评判标准为哪种划分分类错误率最小。

|  |
| --- |
| **Algorithm 5:** *AdaBoots(单层决策树)* |
| **Input:** *train*  **Input:** *M* (Num of Classifiers)  *w[]=1/N* (Initialize weight)  *classifierList* (Use to save classifier)  *result[]=0* (Use to save result)  **For** *i=1* **to** *M*  *Nowtree* = 寻找最小的单层决策树作为分类器(见上面公式)  [*Nowresult*，*Nowerror*]=predict(*train*, *Nowtree*)  *=0.5\*log((1- Nowerror)/ Nowerror)*  **For** *j=1* **to** size(*train*)  使用上述更新*w*公式更新*w*  *result[j]+= Nowresult[i]*  **end**  *classifierList.add(Nowtree)*  **end**  **return** *result* |

随机取12000个样本做验证集，验证集F1在0.7-0.72左右，分类器增加到一定程度后并不会有更好的效果（C++输出画图麻烦就没有画了，主要还是这东西慢…）此部分交由队友，没有实际提交。

从队友使用的AdaBoost效果来看起单层决策树结果最优为0.75，主要还是单层决策树太弱的问题，队友最终实现多层cart决策树+adaboost效果为接近0.8，相对于我之前实现的BPNN的各种优化来说，AdaBoost效果并没有那么好，而且BPNN的划分原理也和AdaBoost类似，然而还是达不到BPNN的效果，主要问题还是在于此时AdaBoost丢失了效率非常低效运行非常耗时。我认为如果有BPNN可以用为什么还要类似的AdaBoost，尤其是在AdaBoost效果不好的时候，类似LR PLA 从我看到的资料中一般都是推荐用NN没有用AdaBoost的。

# 引用

1. UFLDL Tutorial. http://ufldl.stanford.edu/wiki/index.php/UFLDL\_Tutorial
2. Kingma, Diederik P., Ba, Jimmy.Adam: A Method for Stochastic Optimization, arXiv:1412.6980
3. Srivastava N, Hinton G, Krizhevsky A, et al. Dropout: a simple way to prevent neural networks from overfitting[J]. Journal of Machine Learning Research, 2014, 15(1):1929-1958.
4. Adaboost 算法的原理与推导 http://blog.csdn.net/v\_july\_v/article/details/40718799

# 课程总结

AI这门课主要教是机器学习，最近机器学习也是比较热门，我也比较感兴趣，但是这也是我以前从来没有接触过的领域。

一学期下来收获颇丰，学到的内容主要就是基础的AI知识，绝大多数都是入门的算法，而且算法难度也不是很大。这学期所讲的算法主要分为有监督学习和无监督学习两类。基础的算法比如KNN 、NB、LR、PLA、DT、K-meas，复杂一点的比如PBNN，其实这些算法都差不多，关键在于算法的理解，老师课上讲的原理很细，但是很多时候课堂上听推导是一头雾水，因为老师的推导过程讲的我不是很适应，我还是习惯于下课后看着blog进行推导，blog推导详细很多，而且会附上例子。

对于算法的实现，其实这学期的算法都是比较基础的没有什么难点，主要在于算法的推导与理解，实现这个基本上看个人的代码水平，我觉得实现并不是什么大问题。理论课上只讲了算法的基础内容，具体实验的时候还需要自己对算法进行优化，这个部分也是比较麻烦，优化好一个算法往往比实现一个算法困难，这里吐槽一下我觉得为什么TA们都不太重视算法效率呢，我认为算法的优化要在一定时间内跑出结果最好才是优化，以巨量时间换结果我觉得不是不明智的，不过这个见仁见智了主要还是看在算法用在哪需不需要效率。

对于实验课，首先我觉得实验课的PPT制作一个完整的部分还是应该由一个人完成，不然看了一头雾水，突然跳跃，搞得没有连续性，最后还是要自己去看blog或者跑去看别人的公开课什么的，比如说那个BPNN的PPT就是这样看得我一头雾水最后还是自己去看了斯坦福的网页才知道BPNN的具体实现方式。吐槽一下rank，没有意义的放在那边只会让人心急，我觉得不用rank,直接用个黑箱测评自己测评自己的就好了，我觉得这个也不难实现吧，交个答案上去然后测试就类似oj,还省的每天ta去手动测试，这个rank没什么意义，如果是为了一个获得一个泛化的模型，就不该搞rank，或者说rank不计分，现在rank没有任何意义，rank把目的定死了，rank到最终的结果就是会造成对于测试集的过拟合，就算多个测试集多少也是会存在这个问题。

最后说一下这门课我的期待和实际情况，一开始也有预想到饶阳辉老师的AI理论会偏多，实验课手写算法也略有了解，但是和我期望的还是有所差距，我个人并不喜欢全部理论的东西，我觉得要学一些实际的有用的，个人感觉主要还是实验课的分配不好，我觉得应该加一些TensorFlow、Caffe、Keras这类的使用，理解算法固然重要，但是对于不是深入进行AI科研，我认为在学完原理掌握一些工具也是很重要的。