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**

教学班级	**m2**	专业（方向）	**移动**
组号	**71**	组名	**阿坤十八式**
学号	**15352446**	姓名	**钟展辉**

一、 Project 最终结果展示

1. 最终结果

GroupID	GroupName	binary	multi	regression	binary_Rank	multi_Rank	regression_Rank	All_Rank
71	阿坤十八式	0.908673469	0.586117471	62.12899588	12	22	6	12

2. 组内分工

张梓坤：二元分类，adaboost

钟镇威：多元分类

钟展辉（本人）：回归

3. 个人工作

Time	Trying algorithm and method	Test_RMSE:
12.20	开始做 AI 期末 project，使用 BPNN 神经网络. 随机梯度下降+数据归一化+tanh 激活函数 训练集 MSE：7000	107
12.22	发现前一天的 MSE 计算方式有误，正确计算 MSE 后训练集 MSE 回升到 15000.改用批梯度下降+独热编码，天气 3 列、小时 24 列、一星期 7 列、隐藏层节点增至 70 个。训	73.5

	练集 MSE：5200	
12.24~ 12.27	暂无优化头绪，尝试调参，隐藏层节点数、迭代次数、学习率等，更换激活函数 RELU 等，将 12 个月份加入独热编码，尝试正则化、不划分验证集用整个数据集训练、增多隐藏层节点数至 300 等。训练集 MSE：3900	78~83
12.28~ 1.2	暂无优化思路，尝试 KNN，K 取 10 时 训练集 MSE：7500	217
1.3	课上看完小组展示后发觉圣诞节的重要性，观察对比 2011 年 12 月份训练集真实值与我的预测值对比，发现圣诞节期间预测值都高出一倍左右。尝试对圣诞节期间预测值进行参数调整。	63.5
1.4	调整参数和迭代次数，开始写报告	62.1
1.7	发现几天没做回归，排名都从第 6 跌倒 20 了，深感再做下去也比不上他们有精力，于是，直接进入期末复习	
1.11	群上提示数据集是华盛顿的，于是调整圣诞节放假时间，并把年份、是否处于圣诞假日期间加入独热编码	58.6

二、 工作流程

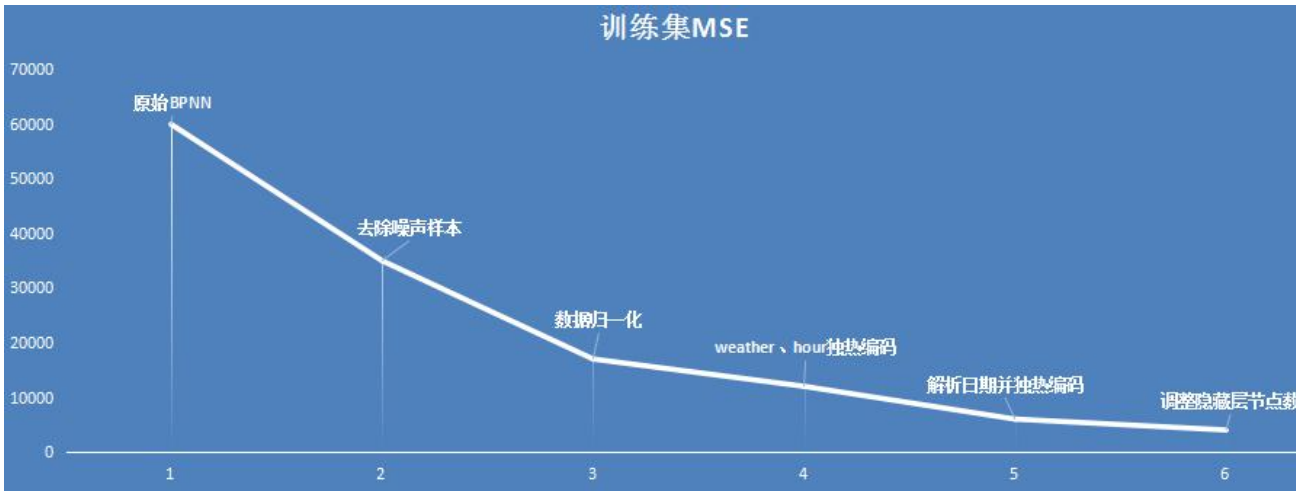
1. 算法简介

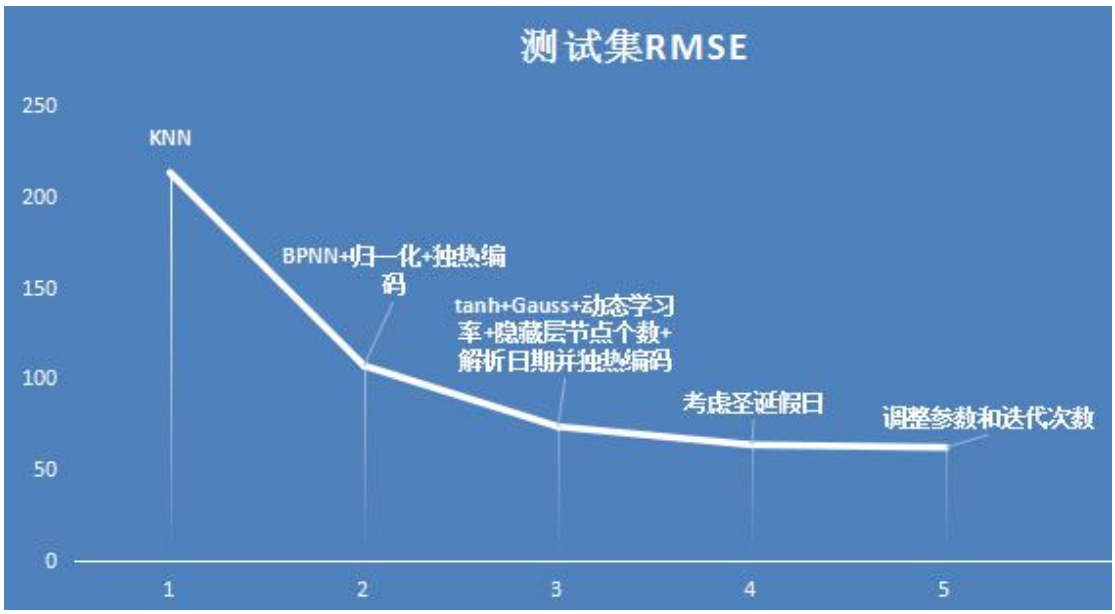
算法	参数	最优 TEST_RMSE
KNN	K=10	217

神经网络	层数：1 个隐藏层 节点数：70 迭代次数：2000 左右	62.1

2. 调参过程

算法优化是基础，用于保证 MSE 平稳下降或者加快学习速度，而数据预处理是降低 MSE 的主要手段：





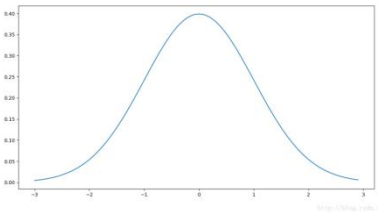
算法优化

上网找到很多算法都是用于分类的 ,用于回归预测的算法很少而且其中神经网络 BPNN 是其中最成熟也是比较容易实现的，而 SVR 之类的新算法不仅没有接触过 , 而且网上相关的文章博客也非常少 , 实现起来不仅会很艰难而且效果估计并不会很好，于是最后决定采用 BPNN 神经网络来做这个实验。

这次使用的 BPNN 算法主体框架继承自上一次实验，在此基础上作改进。改进方法如下：

1、高斯分布初始化。

采用均值 0,方差为 1 的标准正太分布，如下：

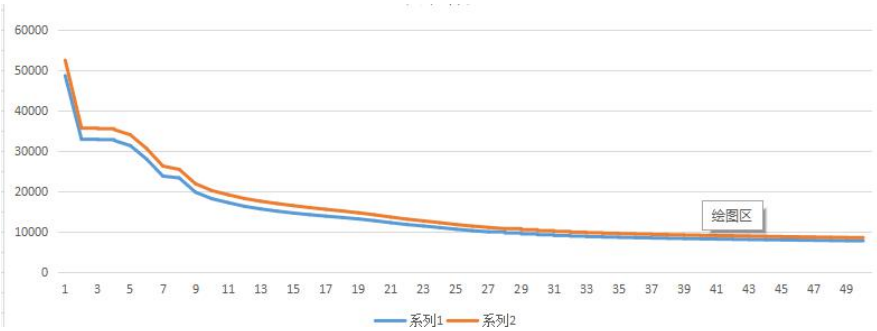


使用高斯分布初始化而不是随机初始化的优点是什么呢？

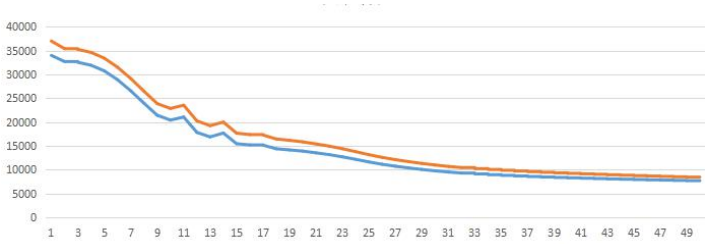
以 sigmoid 函数（logistic neurons）为例，当 x 的绝对值变大时，函数值越来越平滑，趋于饱和，不合适的权重初始化有可能会使得隐藏层的输入的方差过大, 从而在经过 sigmoid 这种非线性层时离中心较远(导数接近 0),因此过早地出现梯度消失.高斯分布初始化能避免这种情况。

（随机初始化偶尔会出现训练集 MSE 卡在某一个区间比较久然后才继续下降，高斯分布初始化比较平稳的下降。）

以下 MSE 曲线每迭代 10 次取值记录，共迭代 500 次：



使用高斯随机分布后的训练集 MSE 下降曲线：



高斯分布初始化前期不会有梯度消失现象。

生成高斯分布随机数的代码

```
int gauss_x = 0;
void UNIFORM(double *p)
{
    int i, a;
    double f;
    for (i = 0; i<2; i++, gauss_x = gauss_x + 689)
    {
        a = rand() + gauss_x;  //加上689是因为系统
        a = a%1000;
        f = (double)a;
        f = f / 1000.0;
        *p = f;
        p++;
    }
}
```

```
#define PI 3.14159
double Gauss(){
    int i, j;
    double A, B, C, E, D, r;
    double uni[2];
    double *p;
    srand((unsigned)time(0));  //随机数种
    E=0;D=1;
    UNIFORM(&uni[0]);  //调用UNIFORM函数产
    A = sqrt((-2)*log(uni[0]));
    B = 2 * PI*uni[1];
    C = A*cos(B);
    r = E + C*D;     //E,D分别是期望和方差
    return r;
}
```

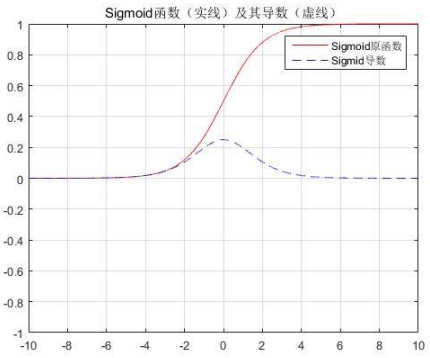
2、tanh 激活函数。

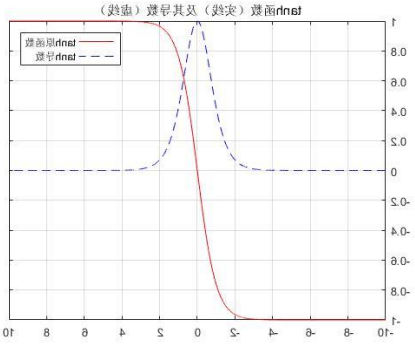
tanh 激活函数：

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

tanh 导函数：

$$f^{'}(x)=1-[f(x)]^{2}$$





Tanh 是 Sigmoid 的变形 ,而由于 tanh 是 0 均值的。因此 tanh 效果 会比 sigmoid 更好。

展示的时候，TA 问到了为什么 tanh 会比 sigmoid 好这一点，当时我没有理清思路，没有讲清楚。这里补充：

跟 sigmoid 一样，tanh 也具有软饱和性，即 x 一旦输入落入饱和区，f'(x) 就会变得接近于 0，导致了梯度也变得非常小，难以训练。但 tanh 网络优点是收敛速度要比 sigmoid 快。因为 tanh 的输出均值比 sigmoid 更接近 0，且在相同的区间内梯度下降幅度比 sigmoid 大，从而降低所需的迭代次数。

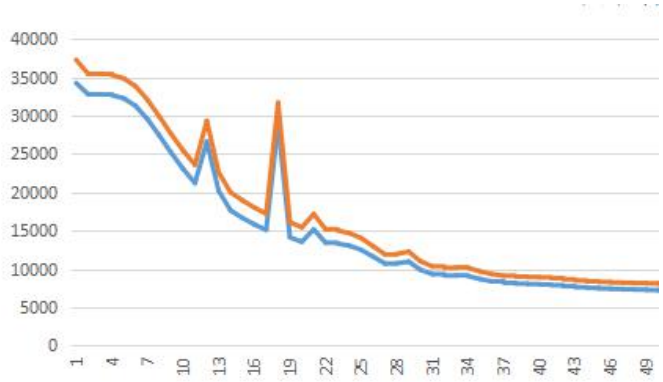
3、动态学习率。

加快前期迭代速度，同时让后期迭代梯度平稳下降。

alpha= =0.005+ 1 /(1+ 0.1 *(cnt+1));

4、批梯度下降。

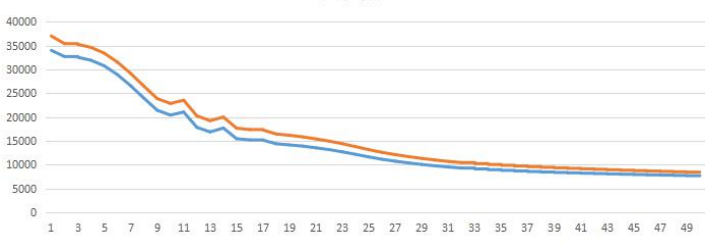
随机梯度下降不稳定,会有明显震荡且每次更新都朝着局部最优而不是整体最优改进；



（图为随机梯度下降）

mini-batch 每次迭代随机选择一定数目的样本进行训练,我觉得随机选择不确定性太大，而且不能很好的利用所有样本，因此最终选择使用批梯度下降。

批梯度下降能保证在每次的迭代中，权值向量都朝着整体更优的方向更新。



（图为批梯度下降）

批梯度下降更稳定一点。

5、隐藏层节点数。

尝试增加隐藏层数量,但是吃力不讨好,还不如调整隐藏层节点数量。一般来讲应设计神经网络应优先考虑 3 层网络（即有 1 个隐层）。靠增加隐层节点数来获得较低的误差,其训练效果要比增加隐层数更容易实现。目前试过隐藏层数为 17、21、25、70、120、300 等,发现隐藏层节点过少,拟合程度较低,隐藏层节点过多，不仅梯度下降速度慢，还会有过拟合情况。最后隐藏层节点数选择 70 个。

6、尝试 L2 正则化

正则化：上面的处理方法中，会出现过拟合的情况，即使训练集 MSE 缓慢的下降到 3000 以下，验证集 MSE 已经提升到 7000 左右，上交 ftp 跑出来的结果 80 左右。因此可以使用 L2 正则化方法防止过拟合.

L2 正则化就是在代价函数后面再加上一个正则化项：

$$C = C_0 + \frac{\lambda}{2n}\sum_w w^2,$$

C0 代表原始的代价函数，后面那一项就是 L2 正则化项：全部参数 w 的平方的和，除以训练集的样本大小 n。

λ就是正则项系数，权衡正则项与 C0 项的比重。另外另一个系数 1/2，主要是为了后面求导的结果方便.

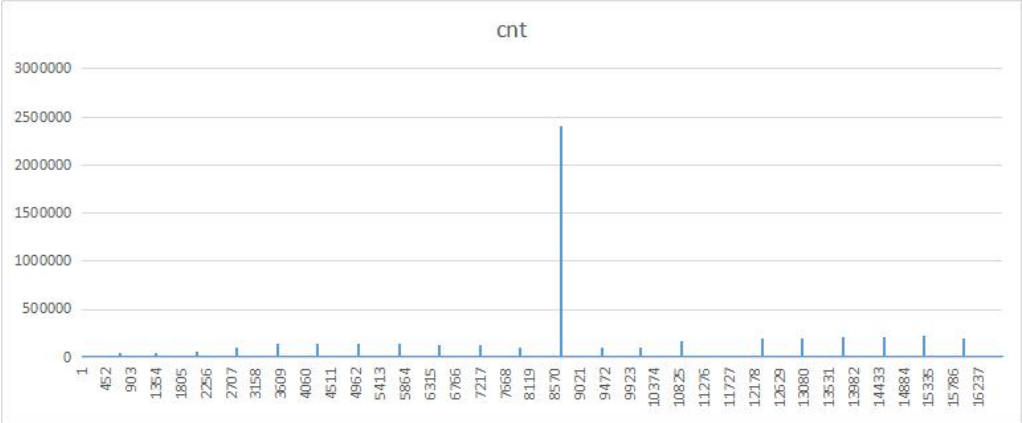
但是由于 L2 正则化调出好参数λ很费时间，太小了没有效果，太大了学习不了，还没有调出一个好的参数。

数据预处理

算法的改进后只能作为一个基础，训练集 MSE 大概能从 60,000 降到 10000 左右，要想再降低 MSE 只能从数据预处理下手，做完数据预处理训练集 MSE 能降到 3000 多，测试集跑的 RMSE 是 60 左右。

1、去除极端值。

使用生成图表看到有十几个样本 cnt 高达 30000。这些噪声点影响极大，神经网络无法学习。MSE 一直在 66000 左右。去除掉极端值后，能下降到 35000 左右。



——

2、归一化。

不同评价指标往往具有不同的量纲和量纲单位，这样的情况会影响到数据分析的结果，为了消除指标之间的量纲影响，需要进行数据标准化处理，以解决数据指标之间的可比性。原始数据经过数据标准化处理后，各指标处于同一数量级，适合进行综合对比评价。归一化后训练集 MSE 能下降到 17000。

——

3、独热编码（weather、hour）。

离散变量，没有数值大小上的意义。比如说 weather 天气主要被分成 3 个类型，特征值分别为 1、2、3，单这三种天气是没有大小关系的，比如不能认为天气一比天气二大这样，但是我们使用的算法却往往默认数据数据是连续的，并且是有序的，这样就不能很好的适配离散特征无序和随机分配的特性。

使用独热编码把 weather 分成 3 列，把 hour 分成 24 列，解析日期分出一个星期 7 列等。

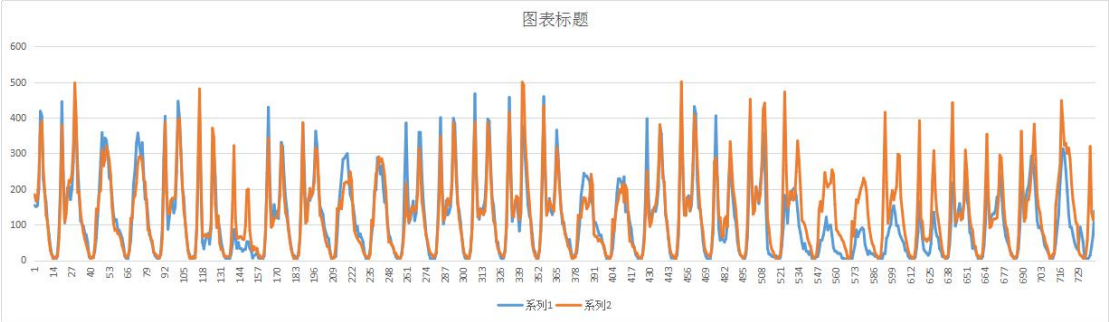
独热编码处理后 MSE 能下降到 6000，如果把月份也独热编码加进特征里，则 MSE 仍能继续下降但是出现过拟合情况。

——

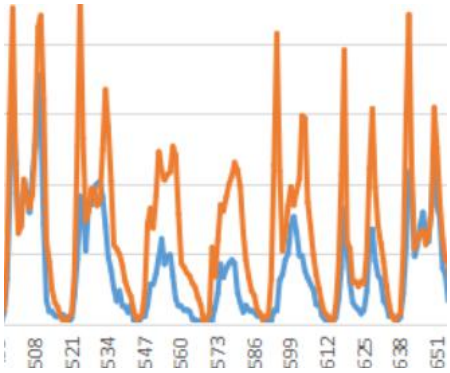
4、 做完以上改进和数据处理后，我的测试集预测结果上交 ftp 后最高有 RMSE=73.5，然后后面的改进尝试比如使用 RELU 激活函数、使用 dropout 正则化等都没有再能提升，后来看了课堂上其他小组的展示后，被他们所说的圣诞节假日所提示，发现这确实也是一个可以改进的地方：

由于测试集是 12 月的数据，而 12 月有比较特殊的圣诞节假日，因此截取出训练集 11 年 12 月份的 cnt 数据与我预测的训练集结果对比：

即训练集 7917~8657 这一区间：



发现在这 740 个数据里面，[521,625]这区间我的预测值都过高：



由于不知道数据集是哪个国家的，因此难以度量圣诞假日长度，一个合理的猜测是圣诞节日 23 号~26 号这期间放假，因此上班人数减少，使用公共自行车的人数也大幅减少，于是我加上判断语句，当判断到当前日期处于圣诞假日期间时，就给它的预测值乘以 0.4~0.7 之间的参数以作调整。最后结果是 RMSE 能下降到 63 左右。

后来 TA 在群上说数据集是华盛顿的，上网一查放假时间大概是 22 号到次年 6

号，于是我再调整一遍，加入年份独热编码，然后 12 月 22 号到 30 号也加入独热编码（在这范围内为 1，否则为 0），最后 RMSE 还能降到 58 左右。

3. 数据集分析

二元分类数据集：因为特征列没有说明因此难以度量各自的重要性，将第一列属性升序排序后发现样本是从 TYPEA 到 TYPEE 的 ,然后 TYPEA 标签几乎全为 0，TYPEE 标签直接就全为 0，其他类型的则 1、0 各占一半左右。

type	label:0	label:1	pro:0	pro:1
type_A	1198	18	0.985197	0.0148026
type_B	3212	2450	0.567291	0.432709
type_C	6484	15890	0.289801	0.710199
type_D	13351	5049	0.725598	0.274402
type_E	348	0	1	0

属性列 K 和 L 的取值为离散值，只包含 0/1 和 0/3 两个离散值。其他属性列取值为连续值。

多元分类数据集：数据量十分庞大，样本数就有 6 万多个，而且每个样本中单词数也很多，如果用 one-hot 矩阵方式存储既耗时间也占用很大空间，因此使用 word2vector 或 document2vector 更好。

回归数据集：预测公共自行车数量，虽然湿度、温度、天气等特征也会有影响，但考虑到人们一般把公共自行车用作上下班工具，因此日期、小时等时间因素影响更大，把时间解析成特征进行训练能取得很好的效果。然后年份也有一定影响，11 年的自行车平均数量少于 12 年，此外测试集是在 12 月份，圣诞假日期间（约在 12 月 22 日到次年 6 日）骑车人数特别少也需要注意。

4. 集成学习方法(AdaBoost)

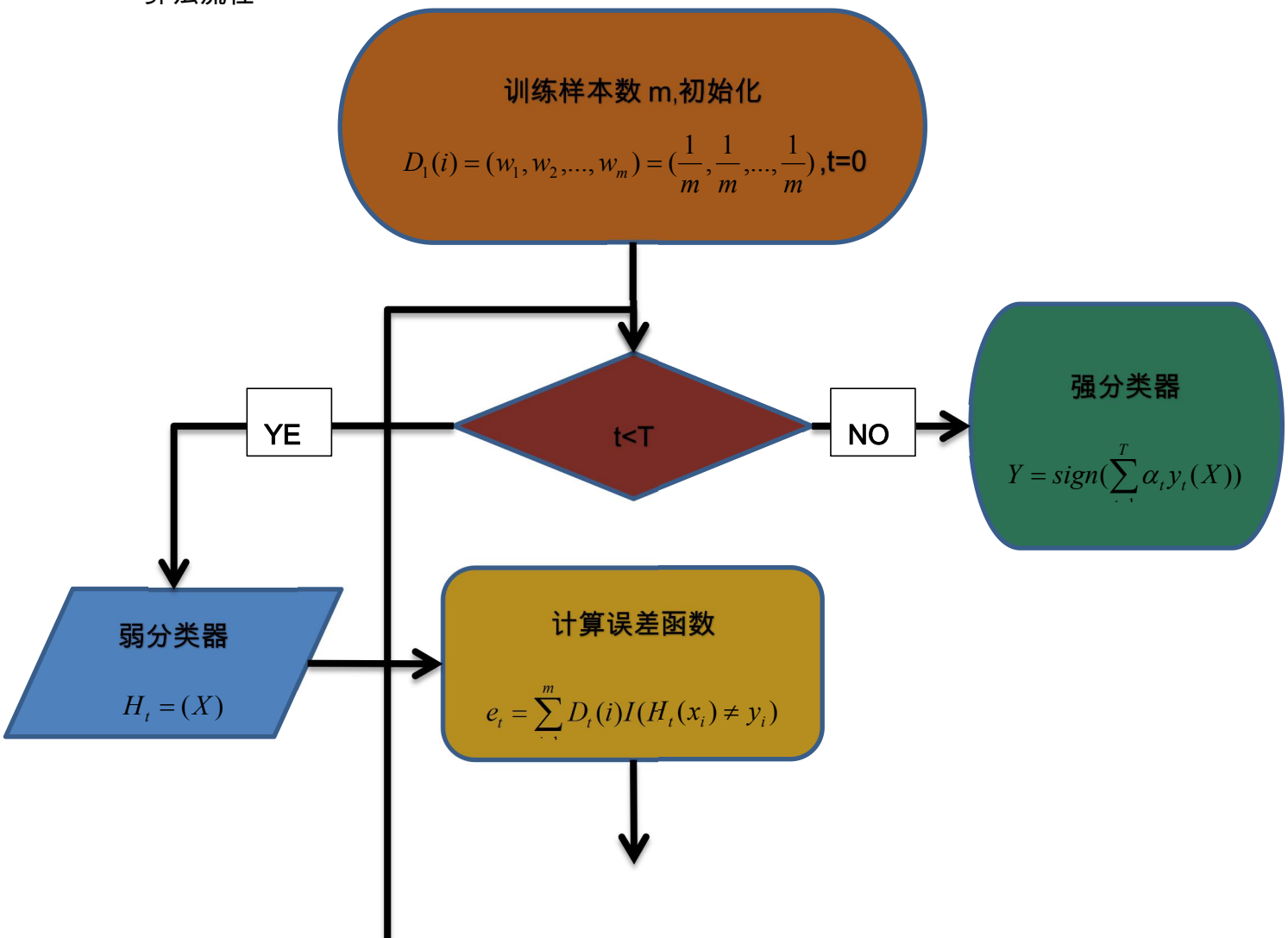
AdaBoost 原理：

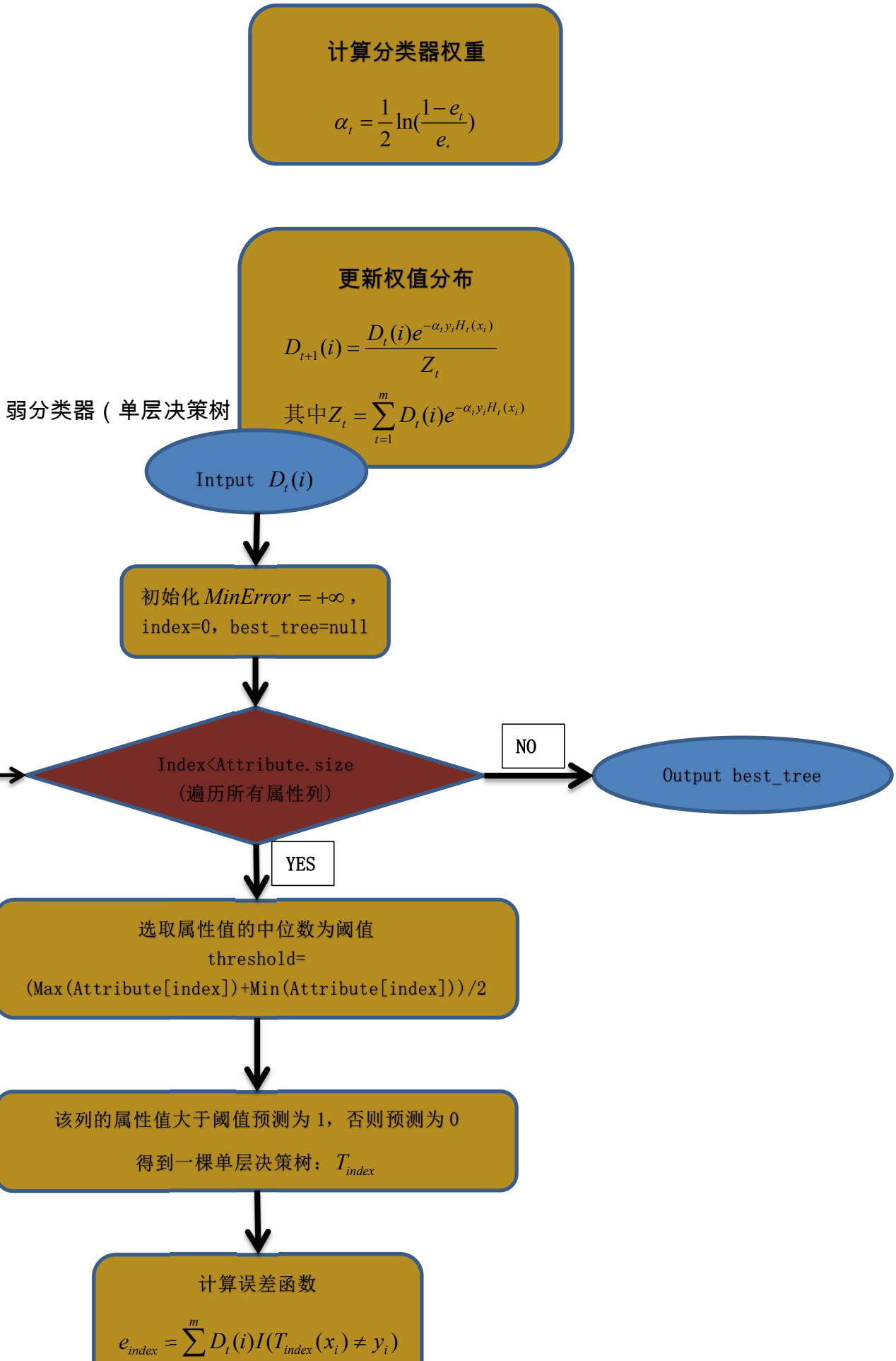
（1）初始化训练数据的权值分布。如果有 N 个样本，则每一个训练样本最开始时都被赋予相同的权重：1/N。

（2）训练弱分类器。具体训练过程中，训练完这一轮，在构造下一个训练集时，降低准确分类的样本的权重、提高误分类的样本的权重。然后，权重更新过的样本集被用于训练下一个分类器，整个训练过程如此迭代地进行下去。

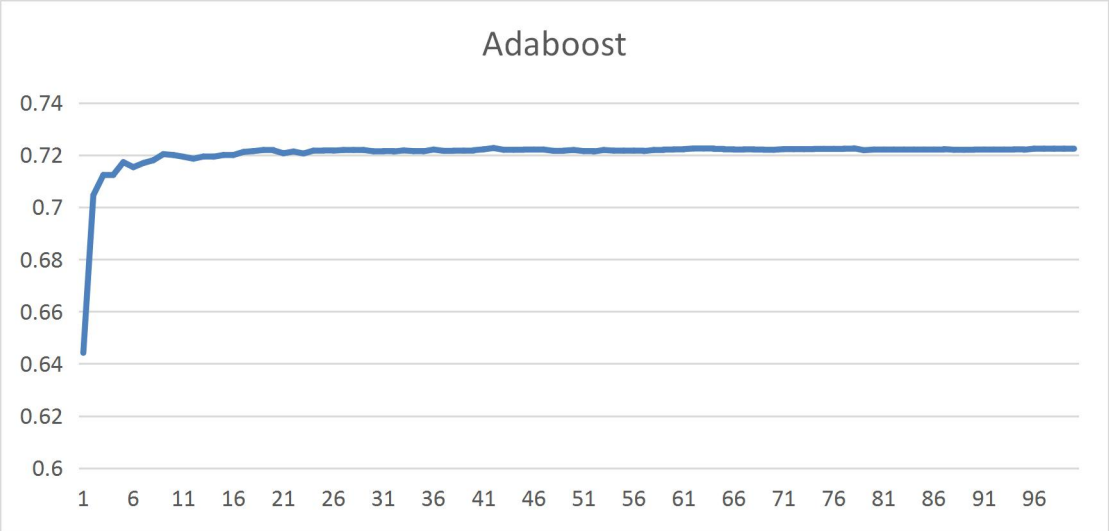
（3）训练完毕后，将所有弱分类器组合成强分类器。各个弱分类器的训练过程结束后，加大分类误差率小的弱分类器的权重，降低分类误差率大的弱分类器的权重，目的是使误差率低的弱分类器在最终分类器中占的权重较大，否则较小。　　我们小组使用 adaboost+决策

算法流程




计算分类器权重
$\alpha_t = \frac{1}{2}\ln(\frac{1-e_t}{e_t})$
更新权值分布
$D_{t+1}(i) = \frac{D_t(i)e^{-\alpha_t y_i H_t(x_i)}}{Z_t}$
其中$Z_t = \sum_{t=1}^{m} D_t(i)e^{-\alpha_t y_i H_t(x_i)}$
弱分类器（单层决策树
Intput $D_t(i)$
初始化 $MinError = +\infty$，
index=0，best_tree=null
Index<Attribute.size
（遍历所有属性列）
NO
Output best_tree
YES
选取属性值的中位数为阈值
threshold=
(Max(Attribute[index])+Min(Attribute[index]))/2
该列的属性值大于阈值预测为 1，否则预测为 0
得到一棵单层决策树：T_{index}
计算误差函数
$e_{index} = \sum_{m} D_t(i)I(T_{index}(x_i) \neq y_i)$

迭代次数（弱分类器数目）与准确率的关系图



注：横坐标值代表 1-1000 的范围。

分析：弱分类器的准确率为 0.64（迭代 1 次），经过 100 次迭代后 Adaboost 的准确率达到 0.72，之后没有再提高，这是由于之后的弱分类器在“最终投票”阶段的权重几乎为 0，不对预测结果有贡献。这说明这种弱模型在经过 100 次学习后已经无法学习到新的规则。

三、 引用

无

四、 课程总结

人工智能实验课算是这学期里所有实验课中我花了最多时间在上面的课了，不仅要研究实验 ppt，请教同学明白算法原理，还要写代码小心调试，在基本正确后还要调试参数、尝试优化方法提高准确率，验收之前还得自己推导一遍公式（像逻辑回归那一次）或者理解各种名词的含义（如朴素贝叶斯、先验概率等），以免被 TA 问的哑口无言，验收完后还得把白天课后时间和晚上空闲时间来写实验

报告，在网上搜博客看懂原理后再自己表述出来或者画图推导公式（那个评分 F 的同学是个教训），然后又自己无中生有想个简单的小数据集出来再自己动笔验证+代码读取运行验证。

虽然花了很多时间有时调参也调的很烦躁，但一个学期下来看到自己做过逻辑回归、决策树、神经网络等算法，还是觉得很有收获。我们的这门人工智能感觉更偏向于机器学习，简单说就是用各种数学模型和方法学习数据集的规律，然后用以预测将来的可能的数据，像实验课这样跟着课程步骤走，学到了挺多算法和思路、基础概念，但是到了底层算法原理方面如果想进一步深入学习还是得学数学啊。