实验报告

## Kmeans算法

Kmeans简介：Kmeans算法为一般用于在无监督学习（Unsupervised Learning）下的聚类算法，其主要原理是计算每一个点划分至其中心点距离（可以是欧几里得距离、也可以是切比雪夫距离或者是曼哈顿距离）最短的簇，直到满足一定的终止条件为止（如中心点不再发生改变，SSE或者SC系数达到极小值）

二分Kmeans简介：改进型的Kmeans算法，同样在开始指定K值，不同的是，先从K = 1开始进行聚类处理，每次挑选SSE值最大的一个聚类（SSE值最大则说明聚类内部还有聚类的可能性更大）进行二分操作，直到K值达到所指定的值

优点在于：

* + - 算法简洁明了，简单易写；
    - 适用于无监督学习，在分类未明的时候可以很好进行划分；
    - 适用于高维数据，即复数个特征的数据；

缺点在于：

* 最终分类数K不明确，经常要手动介入去设置
* 初值设置不当容易陷入局部最优解
* 处理结果受初值影响较大，初值不一样，其生成结果经常也不一样

算法原理（伪代码）

Kmeans算法

KMEANS(DATA， K)

CLUSTER = INIT(K) //初始化中心点

WHILE NOT Ending(K) //不符合终止条件

CLASS = CLASSIFY(DATA, CLUSTER) //为每一个数据进行分类

UPDATE(CLUSTER, CLASS, DATA) //更新中心点的位置

RETURN CLUSTER //返回中心点集合

二分Kmeans算法

BIKMEANS(DATA， K)

CLUSTER ← INIT(1) //初始化中心点

COUNT = 1 //设定COUNT从1开始

WHILE COUNT NOT EQUAL K //不符合终止条件

CLASS = CLASSIFY(DATA, CLUSTER) //为每一个数据进行分类

INDEX = MAXSSE(DATA, CLASS, CLUSTER)

//返回最大SSE的簇的索引

NEWCOLLECT = KMEANS(DATA[INDEX], 2)

//将该簇的数据进行二分

REPLACE(CLUSTER, INDEX, NEWCOLLECT) //更新中心点

CLASS = CLASSIFY(DATA, CLUSTER) //更新分类

UPDATE(CLUSTER, CLASS, DATA) //更新中心点的位置

RETURN CLUSTER //返回中心点集合

代码具体实现：详见Kmeans中的代码

API：

包Kmeans

类 Kmeans

构造函数 Kmeans(count, dis)

Par1Int count 最终聚类的数量，即K值

Par2Int dis 维度，即每一个数据向量的维度

枚举类 Distance 距离算法

欧几里得距离Euclidean = 0

曼哈顿距离Manhattan = 1

实例化方法 fit(data, maxIter, threhold, distance, debug)

对模型进行训练

Par1np.array data 待训练的数据，维度为N \* dis

Par2int maxIter 最大迭代次数，缺省为10000

Par3int threhold 中心点终止条件，代表threhold个中心点停止作为终止条件，-1则是全部中心点停止，缺省为-1

Par4Kmeans.Distance distance 采用的距离算法，缺省为Kmeans.Distance.Euclidean

Par5bool 是否开启debug模式，缺省为0

实例化方法 classify(data, dis) 对数据进行分类

Par1np.array data 待分类的数据，维度为N \* dis

Par2Kmeans.Distance distance dis 采用的距离算法，缺省为Kmeans.Distance.Euclidean

Return np.array 数据的分类结果，维度为N \* 1

静态方法 SSE(collective, data, dataClass) 计算SSE

Par1np.array collective 中心点集，

Par2np.array data 具体数据

Par3np.array dataClass 每个数据的类型集合

类 BiKmeans

构造函数 Kmeans(count, dis)

Par1Int count 最终聚类的数量，即K值

Par2Int dis 维度，即每一个数据向量的维度

枚举类 Distance 距离算法

欧几里得距离Euclidean = 0

曼哈顿距离Manhattan = 1

实例化方法 fit(data, maxIter, threhold, distance, debug)

对模型进行训练

Par1np.array data 待训练的数据，维度为N \* dis

Par2int maxIter 最大迭代次数，缺省为10000

Par3int threhold 中心点终止条件，代表threhold个中心点停止作为终止条件，-1则是全部中心点停止，缺省为-1

Par4Kmeans.Distance distance 采用的距离算法，缺省为Kmeans.Distance.Euclidean

Par5bool 是否开启debug模式，缺省为0

静态方法 classify(collective, data, dis) 对数据进行分类

Par1np.array collective 中心点的集合，维度为K \* dis

Par2np.array data 待分类的数据，维度为N \* dis

Par3Kmeans.Distance distance dis 采用的距离算法，缺省为Kmeans.Distance.Euclidean

Return np.array 数据的分类结果，维度为N \* 1

实验1：

实验材料：iris数据集

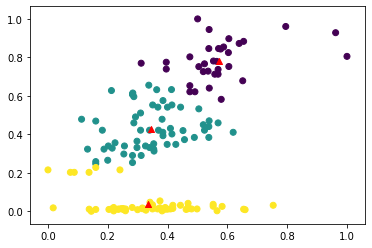
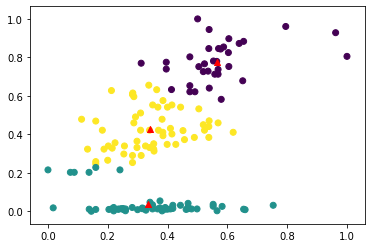
实验对象：Kmeans包，sklearn.cluster.KMeans包

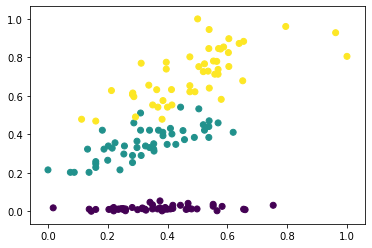
对于复杂数据

iris['sepal\_size'] = iris['sepal\_length'] \* iris['sepal\_width']

iris['petal\_size'] = iris['petal\_length'] \* iris['petal\_width']

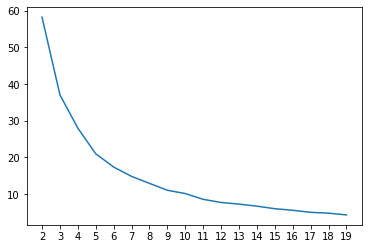
进行聚类处理

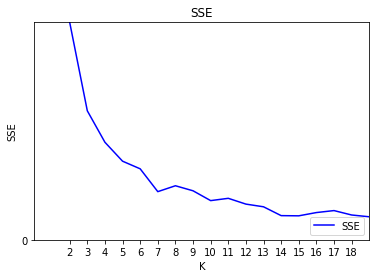
采用SKLearn的Kmeans进行的分类

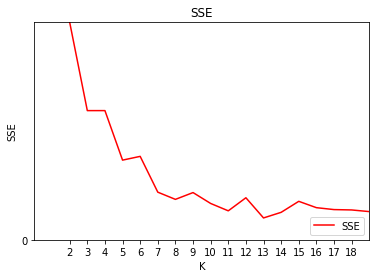
采用土制二分Kmeans进行的分类

数据集的真正分类

可以看出土制的Kmeans算法很好的拟合了SKLearn的Kmeans算法，几乎一致。

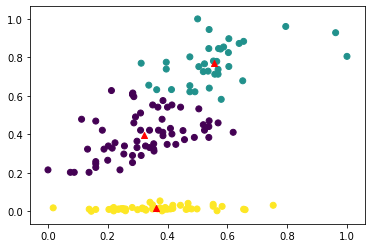
但是我再看一组随着K值而变化的SSE的图表

SKlearn的Kmeans算法产生的SSE图像

采用土制Kmeans算法产生的SSE图像

采用土制二分Kmeans算法产生的SSE图像

很显然，土制Kmeans的SSE波动厉害，线路曲折。尽管手肘法则得以判断标准的K值，但是也说明了在稳定性方面，SKLearn略胜一筹。

并且看出由土制Kmeans法生成的原数据集标签几乎完全拟合的结果也可以发现本算法并不稳定，依赖于初值。

## 反向传播算法（BP）

BP算法简介：BP算法是基于梯度下降（SGD）神经网络的基础上，通过计算代价函数（Cost function）不断调整每个神经元的权重与偏置，使得代价函数最终达到一定的局部最小值，代价函数达到最小值就意味着错误几率最小，即从综合上（正确率、预报率等）最接近真实结果，即适用于预测也适用于分类。

优点在于：

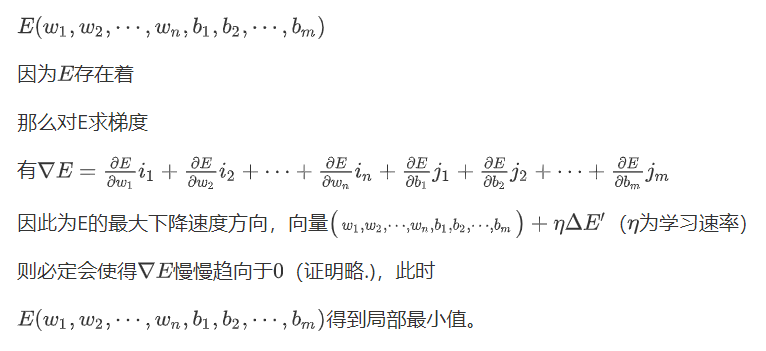
* + - 可用于分类也可用于预测；
    - 通用性好，并且随着隐藏层与输入输出层的调整可以用于高维数据
    - 通过调整代价函数，可以实现识别率、精确率优先

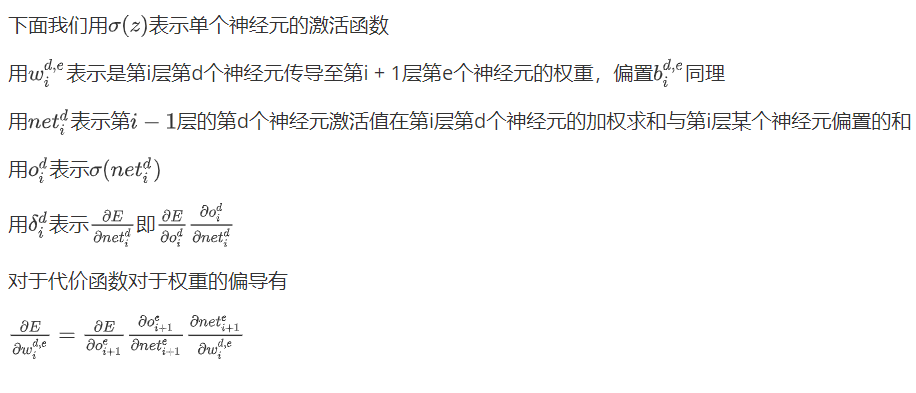
缺点在于：

* 运算复杂，机器学习时间与周期长
* 属于黑箱问题，具体工作机理不明，难以人工调整内部算法
* 自由度高，学习速率、正则化参数不当容易陷入局部最优解
* 对数据训练集的要求高，需要人工对其进行预处理与分类

算法基本原理：

利用数学上的梯度性质，梯度表示某一函数在该点处的方向导数沿着该方向取得最大值，即函数在该点处沿着该方向（此梯度的方向）变化最快，变化率最大（为该梯度的模）。

也就是说设有代价函数

其中对函数求偏导数就需要用到数学中的链式法则（Chain Rule）从最高层一直算起直到最低层，故被称为反向传播。

因此可以得到具体公式（证明略.）

API：

包network

类network

构造函数(layers)

Par1List layers 隐藏层的设计（包括输入输出层）

枚举类 costFun：代价函数

MSE 0（Mean Squared Error）均方误差函数

CEE 1（Cross Entropy Error）交叉熵损失函数

实例化方法feedForward(input) 进行一轮向前传播

Par1np.array input 输入的特征

Return np.array 返回最终的值

实例化方法backProp(input, ans, eta, cost, lamdan)进行反向传播

Par1np.array input 要训练的特征

Par2np.array ans 准确的结果

Par3float eta 学习速率（Learn rate）

Par4network.network.costFun cost损失函数

Par5float lamda / n的值（lamda为正则化参数，n为样本总数）

Return np.array 返回最终的值

实验2：

对mnist集进行分类

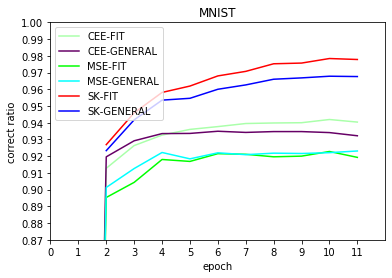
实验材料：MNIST数据集

实验对象：

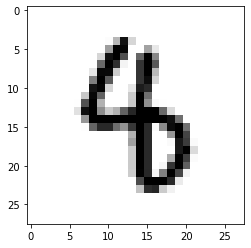
采用MSE的本算法（eta = 1.0, lamdn = 0.0, minibatch = 1）

采用CEE的本算法（eta = 0.1, lamdan = 0.0，minibatch = 1）

SKLearn的相同参数模型

 实验结果：

由图不难看出，本算法的实现是相当好了，尤其是CEE要比MSE的学习速度还要高，仅一轮学习已经达到了0.92的好成绩，最终收敛于0.95左右。但是与SKLearn的标准模型相比，简直是被吊打，SKLearn的拟合度即使是越过了0.96的界限仍然在上升。

另外我们再来看几个有趣的随机样本

这个图显然是旋转的8，但是本模型处于4与8之间的竞合，最终选择了4，但是对于我而言第一眼的确是很像异形的4。因此对模型之改进，仍要加入对旋转图的识别，以及样本存在优化空间。

因此本算法的改进空间：

* 对于旋转的图像进行深度学习
* 增加miniBatch的数据处理方式
* 对8与0之间的识别进行深度学习