# DUTI算法代码说明

向首兴 2019/5/21

1. 文件架构

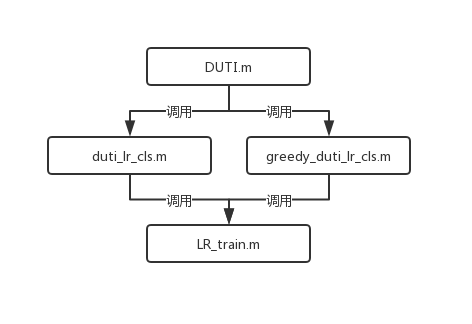


图 1文件调用关系

1. 函数架构及功能
   1. duti\_lr\_cls.m
      1. function [flag\_bugs, delta, ranking] = duti\_lr\_cls(X\_train, y\_train, X\_trust, y\_trust, num\_class, conf,lam, max\_iter)

基于可信点的类标更新数据类标。

1. 参数：

* X\_train：训练数据，n\*d矩阵
* y\_train：训练数据类标，n\*1矩阵
* X\_trust：可信点，m\*d矩阵
* y\_trust：可信点类标，m\*1矩阵
* num\_class：类标数量
* conf：可信点的可信度，m\*1矩阵
* lam：LR学习器的L2正则项的权重
* max\_iter：最大迭代次数

1. 返回值：

* flag\_bugs：修改的数据点的标志，n\*1矩阵(bool)
* delta：修复结果，n\*num\_class矩阵，值在[0, 1]，数据点属于各个类别的概率
* ranking：修改的数据点的优先程度，ranking(i) = (第一次0.5<=delta(i)<=1的轮次) + (1 – 该delta(i))
  + 1. function [cost, grad] = lr\_debug\_obj(delta, X\_train, y\_train, X\_trust, y\_trust, gamma, conf, lam)

根据当前的修复结果，训练LR模型，并结合训练数据原始类标，可信点类标得到当前最小化问题的梯度和损失函数（cost function）值

1. 参数：

* delta：当前的修复结果，n\*num\_class矩阵，值在[0, 1]，数据点属于各个类别的概率
* X\_train：训练数据，n\*d矩阵
* y\_train：训练数据类标，n\*1矩阵
* X\_trust：可信点，m\*d矩阵
* y\_trust：可信点类标，m\*1矩阵
* gamma0：当前的最小化问题中的正则项的权重
* conf：可信点的可信度，m\*1矩阵
* lam：LR学习器的L2正则项的权重

1. 返回值：

* cost：当前的最小化问题的损失函数（cost function）值
* grad：当前迭代中最小化问题的梯度
  + 1. function h = lr\_predict\_prob(X, theta)

返回LR模型预测的类别分布概率。

1. 参数：

* X：数据，n\*d矩阵
* theta：LR学习器的分类面，num\_class\*d矩阵

1. 返回值：

* h：数据点的类别分布概率，n\*num\_class矩阵
  + 1. function [delta, step] = gp\_optimizer(func, delta0)

迭代求解最小化问题。

1. 参数：

* func：最小化问题，返回损失函数值cost和梯度grad（例如函数lr\_debug\_obj）
* delta0：数据点的类别分布概率，n\*num\_class矩阵

1. 返回值：

* delta：数据的类别分布概率，n\*num\_class矩阵
* step：数据的概率分布变化，n\*num\_class矩阵
  + 1. function [step] = gp\_step(grad0, delta0, n, d, lr)

所有数据点在满足线性约束情况下，向梯度方向下降一步。

1. 参数：

* grad0：数据的梯度，n\*num\_class矩阵
* delta0：数据的类别分布概率，n\*num\_class矩阵
* n：数据数量
* d：类别数量num\_class
* lr：学习率

1. 返回值：

* step：数据的概率分布变化，n\*num\_class矩阵
  + 1. function step = row\_direction\_search(grad, delta, d, lr)

每一数据点在满足线性约束情况下，向梯度方向下降一步。

1. 参数：

* grad：数据点的梯度，1\*num\_class矩阵
* delta：数据点的类标概率分布，1\*num\_class矩阵
* d：类别数量num\_class
* lr：学习率

1. 返回值：

* step：数据点的概率分布变化，1\*num\_class矩阵
  1. greedy\_duti\_lr\_cls.m
     1. function [flag\_bugs, delta, ranking] = greedy\_duti\_lr\_cls(X\_train, y\_train, X\_trust, y\_trust, num\_class, conf, lam, max\_iter, max\_depth, search\_grid)

基于可信点的类标更新数据类标。

1. 参数：

* X\_train：训练数据，n\*d矩阵
* y\_train：训练数据类标，n\*1矩阵
* X\_trust：可信点，m\*d矩阵
* y\_trust：可信点类标，m\*1矩阵
* num\_class：类标数量
* conf：可信点的可信度，m\*1矩阵
* lam：LR学习器的L2正则项的权重
* max\_iter：最大迭代次数
* max\_depth：最大搜索深度
* search\_grid：搜索时分组的组数

1. 返回值：

* flag\_bugs：修改的数据点的标志，n\*1矩阵(bool)
* delta：修复结果，n\*num\_class矩阵，值在[0, 1]，数据点属于各个类别的概率
* ranking：修改的数据点的优先程度，ranking(i) = (第一次0.5<=delta(i)<=1的轮次) + (1 – 该delta(i))
  + 1. function [flag\_feasible, step, grad\_score] = row\_direction\_search(grad, delta, d)

每一数据点在满足线性约束情况下，向梯度方向下降最大或者不变化。

1. 参数：

* grad：数据点的梯度，1\*num\_class矩阵
* delta：数据点的类标概率分布，1\*num\_class矩阵
* d：类别数量num\_class

1. 返回值：

* flag\_feasible：是否进行下降（bool）
* step：数据点的概率分布变化，1\*num\_class矩阵
* grad\_score：数据点的概率分布变化的增益（double）
  + 1. function [flag\_feasible, step, grad\_score] = gp\_step(grad0, delta0, n, d)

所有数据点在满足线性约束情况下，向梯度方向下降一步。

1. 参数：

* grad0：数据的梯度，n\*num\_class矩阵
* delta0：数据的类别分布概率，n\*num\_class矩阵
* n：数据数量
* d：类别数量num\_class

1. 返回值：
   * flag\_feasible：是否进行下降，n\*1矩阵（bool）
   * step：数据点的概率分布变化，1\*num\_class矩阵
   * grad\_score：数据概率分布变化的增益，n\*1矩阵（double）
     1. function thresholds = calc\_search\_grid\_thresholds(grad\_scores, search\_grid)

均匀切分或者最大组间距切分数据概率分布变化的增益，返回切分的阈值。

1. 参数：

* grad\_scores：数据概率分布变化的增益，n\*1矩阵（double）
* search\_grid：搜索时分组的组数

1. 返回值：

* thresholds：用来分组的阈值，search\_grid\*1矩阵（double）
  + 1. function [flag\_continue, delta, cost] = line\_search(func, cost\_func, delta0, search\_grid, depth)

根据梯度搜索更好的数据类标概率分布。

1. 参数：

* func：最小化问题，返回损失函数值cost和梯度grad（例如函数lr\_debug\_obj）
* cost\_func：最小化问题的损失函数，返回损失函数值cost（例如函数lr\_ cost \_obj）
* delta0：数据点的类别分布概率，n\*num\_class矩阵
* search\_grid：搜索时分组的组数
* depth：当前的搜索深度

1. 返回值：

* flag\_continue：是否搜索到更好的数据类标概率分布（bool）
* delta：搜索到的数据类标概率分布，n\*num\_class矩阵
* cost：搜索到的数据类标概率分布下的损失函数值
  + 1. function [delta, step] = gp\_optimizer(func, cost\_func, delta0, max\_depth, search\_grid)

迭代求解最小化问题。

1. 参数：

* func：最小化问题，返回损失函数值cost和梯度grad（例如函数lr\_debug\_obj）
* cost\_func：最小化问题的损失函数，返回损失函数值cost（例如函数lr\_ cost \_obj）
* delta0：数据点的类别分布概率，n\*num\_class矩阵
* max\_depth：最大搜索深度（可理解为line\_search的最大次数）
* search\_grid：搜索时分组的组数

1. 返回值：

* delta：数据的类别分布概率，n\*num\_class矩阵
* step：数据的概率分布变化，n\*num\_class矩阵
  + 1. function cost = lr\_cost\_obj(delta, X\_train, y\_train, X\_trust, y\_trust, gamma, conf, lam)

根据当前的修复结果，训练LR模型，并结合训练数据原始类标，可信点类标得到当前最小化问题的损失函数（cost function）值

1. 参数：

* delta：当前的修复结果，n\*num\_class矩阵，值在[0, 1]，数据点属于各个类别的概率
* X\_train：训练数据，n\*d矩阵
* y\_train：训练数据类标，n\*1矩阵
* X\_trust：可信点，m\*d矩阵
* y\_trust：可信点类标，m\*1矩阵
* gamma：当前的最小化问题中的正则项的权重
* conf：可信点的可信度，m\*1矩阵
* lam：LR学习器的L2正则项的权重

1. 返回值：

* cost：当前的最小化问题的损失函数（cost function）值
  + 1. function [cost, grad] = lr\_debug\_obj(delta, X\_train, y\_train, X\_trust, y\_trust, gamma, conf, lam) 与duti\_lr\_cls.m中相同，见上。
    2. function h = lr\_predict\_prob(X, theta)与duti\_lr\_cls.m中相同，见上。
  1. LR\_train.m
     1. function alpha = LR\_train(X, y, lam, alpha0, preprocessed)

训练LR模型分类器。

1. 参数：

* X：数据，n\*d矩阵
* y：训练数据类标分布，n\*num\_class矩阵
* lam：LR学习器的L2正则项的权重
* alpha0：初始分类器，num\_class\*d矩阵
* preprocessed：是否已经对X和y进行了处理（bool），如X需要添加一列0，y应该为1 based

1. 返回值：

* alpha：分类器，num\_class\*d矩阵
  + 1. function [J, grad] = costLR(alpha, X, y, lam)

训练LR模型分类器。

1. 参数：

* X：数据，n\*d矩阵
* y：训练数据类标分布，n\*num\_class矩阵
* lam：LR学习器的L2正则项的权重
* alpha0：初始分类器，num\_class\*d矩阵

1. 返回值：

* alpha：分类器，num\_class\*d矩阵