# **背景知识**

## **实际场景**

**二分类问题**：逻辑回归可以解决

建模案例：

1、对数据进行二元分类

2、输出sigmoid值，预测某件事情的概率

**广义线性模型**：可以用于保险预测，根据保障责任计算预期的总索赔（总索赔就是纯保费）

总索赔额集中在0附近

案例：

1、可以用tweedie分布进行拟合，参数在1-2之间

2、可以通过两步建模间接近似：索赔次数、平均索赔金额

索赔次数：泊松分布、负二项分布

平均索赔次数：伽马分布、逆高斯分布

## **广义线性模型定义**

线性回归回顾：响应变量Y的值有两部分，

一个是系统组件：线性预测器η=βTx，可以拟合

一个是误差组件：白噪声是高斯随机变量, ε~N(0, 1)

Y~N(xTβ, 1)

GLM允许误差项的概率分布为指数分布族：伯努利分布（逻辑回归），泊松分布，gamma分布，复合泊松Gamma分布，Tweedie分布…

广义线性模型有三个组件：

1、系统组件：线性预测器η=βTx，β为参数。η范围是R

2、随机组件：指数分布族，参数为θ

拟合变量Y的期望μ(0-1之间)和自然参数θ之间一一对应，这个对应关系为ψ

3、连接函数g使得η=g(μ)，逻辑回归中为sigmoid

广义线性回归中，连接函数和ψ相等时，这个连接函数为标准连接函数

y – θ <-> μ <-> η <-> β, 求导：l/θ \* θ/μ \* μ/η \* η/β

一些连接函数参考：

(tweedie, gamma回归等都可以使用log作为连接函数)

|  |  |  |  |
| --- | --- | --- | --- |
| 名称 | identity | logit | log |
| 连接函数 | η=μ | η=ln{μ/(1-μ)} | η=ln(μ) |
| 激活函数 | μ=η | μ = exp(η) / (1 + exp(η)) | μ=exp(η) |
| μ的范围 | R | 0~1 | >0 |

# **SF模型-密态SSLR/SSGLM**

## **参数估计**

一阶优化器：SGD参数估计法

优点：计算和通信量都比较小

二阶优化器：迭代重加权最小二乘法IRLS

优点：初始化准确、收敛快

缺点：计算、通信复杂度高

sf可以用两种优化器叠加，前几轮用二阶初始化并且简单迭代，后面用一阶更快收敛

## **密态计算加法和乘法：基于秘密分享**

SS求和案例：

（3个参与方，数据为15、25、10）

15的这一方生成2个数3, 7并用15减去这两个数得到5，有了3、7、5三个分片

然后把3、5发给另外两个参与方，并且收到两个数值6和9

计算6+9+7 -> 22，把剩下两方的计算结果值加起来，得到最终的50

SS求积案例：

A方把a拆为a1和a0，发出去a1，B方发出b0

随机生成u、v并且计算z=uv，同样对这三个数进行拆分

A算e0 = a0 - u0，f0 = b0 – v0，B计算e1和f1

交换得到e = e0 + e1和f

A计算c0=f\*a0 + e\*b0 + z0, B计算c1 = -ef + f\*a1 + e\*b1 + z1

双方得到c = c0 + c1 = ab

有了加法和乘法，就可以进行参数估计

# **应用实现：理论到sf应用**

## **SSGLM参数**

ss\_glm\_train\_comp.str\_attr(

name="label\_dist\_type",

desc…

is\_list=False,

is\_optional=False,

allowed\_values=[“Bernoulli”, “Poisson”, “Gamma”, “Tweedie”]

)

link\_type参数：

Logit: 逻辑回归，Bernoulli分布

Log: 上面Bernoulli以外的分布都可以用

Reciprocal

Identity

Tweedie需要tweedie\_power这个参数，可以选0-1之间的参数建立几个模型，选效果最好的

optimizer选SGD时可以设置iter\_start\_irls，这是个int参数，可以选1-3

offset\_col参数为偏置值

dist\_scale描述数据的方差，一般默认为1，可以调

weight\_col是数据的加权

## **SSLR使用**

1、生成spu，准备归一化好的数据

sf.init(…)

alice, bob = sf.PYU(…),…

vdf\_x0=sf.data.vertical.read\_csv({bob:bob\_x0\_file})

vdf\_x1=sf.data.vertical.read\_csv({bob:bob\_x1\_file})

vdf\_y0=sf.data.vertical.read\_csv({alice:alice\_y0\_file})

x=sf.data.mix.MixDataFrame(partitions=[vdf\_x0, vdf\_x1])

y=sf.data.mix.MixDataFrame(partitions=[vdf\_y0])

2、模型训练

from secretflow.ml.linear import FlLogisticRegressionMix

model = FlLogisticRegressionMix()

model.fit(

x,

y,

batch\_size=64,

epochs=3,

learning\_rate=0.1,

aggregators=[aggregator0, aggregator1, aggregator2],

heus=[heu0, heu1, heu2]

)

3、模型评估

import numpy as np

from sklearn.metrics import roc\_auc\_score

y\_pred=np.concatenate(sf.reveal(model.predict(x)))

auc=roc\_ auc\_score(label.values, y\_pred)

acc=np.mean((y.pred>0.5)==label.values)

print(auc, acc)

## **SSGLM使用**

可以选用IRLS和SGD两种训练方式。

model = SSGLM(cluster.spu)

model.fit\_irls()

model.fit\_sgd()

# **实践**

明文、密文分别训练open\_ mtpl2数据集，完整代码见homework 1