1. 对比scikit-learn和Spark中的机器学习算法。

见文档《scikit-learn和Spark中的机器学习算法对比》。

2. 测试scikit-learn中线性SVM的运算结果和Spark中是否相同。

答：scikit-learn中有2种方式运行线性SVM分类，即SCV with linear kernel和LinearSVC，

两者运行结果（决策边界）不同，原因在于：

A. 损失函数不同：LinearSVC minimizes the squared hinge loss while SVC minimizes the regular hinge loss.

B. 多分类时，分类方式不同：LinearSVC uses the One-vs-All (also known as One-vs-Rest) multiclass reduction while SVC uses the One-vs-One multiclass reduction。

但是这两种方式计算损失函数是用hessian矩阵，对hessian矩阵处理的策略是定义一个内存处理类Cache类，预先认为分配一定的内存，存储计算好的*Qij*，其序号的检索采用双向链表的方式，加快检索速度。

而Spark中的SVM算法最小化损失函数使用SGD优化算法：

|  |  |  |  |
| --- | --- | --- | --- |
| 参数 | Spark | scikit-learn(LinearSVC) | scikit-learn(SVC kernel=linear) |
| 迭代次数 | maxIteration: 100 | max\_iter: 1000 | max\_iter: -1 |
| 步长 | stepSize: 1.0 |  |  |
| 收敛精度 | threshould: 0.0 | tol: 1e-4 | tol: 1e-3 |
| 正则化系数 | regParam： 0.01 | C: 1.0 | C: 1.0 |
| 损失函数 | regular hinge loss | loss: squared\_hinge | loss: regular hinge loss |
| 随机样本比例 | miniBatchFraction: 1.0 |  |  |
| 输入label | [0, 1] | [-1, 1] | [-1, 1] |
| 正则化函数 | SquaredL2Updater | penalty: l2 |  |
|  |  | dual: True(samples比feature多是最好为false) |  |
| 分类数 | 2 | multi\_class: ovr | decision\_function\_shape: None |
| 截距 | addIntercept: Fasle | fit\_interceptL: True(数据已经居中的时候不用添加) |  |
|  |  | verbose: False | verbose: False |
|  |  | random\_state: None(shuffle时用) | random\_state: None |
|  |  |  | shrinking: True |

在scikit-learn种，SGDClassifier的参数loss设置为hinge时，即是线性支持向量机，因此这里对比Spark的SVMwithSGD和SGDClassifier（loss=hinge）：

|  |  |  |
| --- | --- | --- |
| 参数 | Spark | SGDClassifier（loss=hinge） |
| 迭代次数 | maxIteration: 100 | n\_iter: 5 |
| 步长 | stepSize: 1.0 | learning\_rate：   optimal-eta = 1.0 / (alpha \* (t + t0)) |
| 收敛精度 | threshould: 0.0 |  |
| 正则化系数 | regParam： 0.01 | alpha: 0.0001 |
| 损失函数 | regular hinge loss | regular hinge loss |
| 随机样本比例 | miniBatchFraction: 1.0 |  |
| 输入label | [0, 1] | [-1, 1] |
| 正则化函数 | SquaredL2Updater | penalty: l2 |
| 分类数 | 2 |  |
| 截距 | addIntercept: Fasle | fit\_intercept: True(if false, data should be centered) |
|  |  | shuffle: True |
|  |  | random\_state: None |
|  |  | n\_jobs: 只有多分类的时候用 |
|  |  | power\_t：0.5（缩放学习率的指数） |

正在研究：

SGDClassifier如何得到模型的参数（每一维的系数，迭代次数，每次迭代的损失，是否收敛等）。

3. 测试scikit-learn中线性SVM单机多核是否线性。

答：Spark种的SVMwithSGD只能做二分类，而SGDClassifier（loss=hinge）中的n\_jobs只有在多分类的时候使用，n\_jobs：The number of CPUs to use to do the OVA (One Versus All, for multi-class problems) computation。因此测试线性与否时只能针对多分类，同时并非与Spark相同的数据并行。

1. ALS单机版的测试

libFM是一种用于分解机器的软件实现，其具有随机梯度下降（SGD）和交替最小二乘法（ALS）优化以及使用马尔科夫链蒙特卡罗（MCMC）的贝叶斯推理。，pyFM、pywFM和fastFM实现了python版的libFM。

参考：

[1] [Factorization Machines with libFM, in ACM Trans. Intell. Syst. Technol., 3(3), May. 2012](http://www.csie.ntu.edu.tw/~b97053/paper/Factorization Machines with libFM.pdf)

[2] <https://github.com/jfloff/pywFM>

[3] <https://github.com/coreylynch/pyFM>

[4] https://github.com/ibayer/fastFM/