中科院计算培训中心

Part 3 大数据挖掘分析工具

Mahout和MLlib

- 杨文川

1）Yarn中的Mahout介绍
2）Spark中的Mahout/MLib介绍
3）推荐系统及其Mahout实现方法
4）信息聚类及其MLlib实现方法
5）分类技术在Mahout/MLib中的实现方法

Hadoop大数据挖掘工具Mahout

- Mahout 是 Apache Software Foundation (ASF) 开发的一个开源项目
  - 目标是创建一些可伸缩的数据挖掘算法，供开发人员在 Apache 在许可下免费使用。
  - Mahout 包含许多实现，包括集群、分类、CF 和进化程序。
  - 此外，通过使用 Apache Hadoop 库，Mahout 可以有效地扩展到云中。

背景知识

- Mahout的意思是大象的饲养者及驱赶者。
  - Mahout 这个名称来源于Hadoop徽标上的大象
  - Mahout利用Hadoop来实现可伸缩性和容错性。

Mahout 的历史

- Mahout 项目是由 Apache Lucene（开源搜索）社区中，对数据挖掘感兴趣的一些成员发起的
  - 希望建立一个可靠、文档翔实、可伸缩的项目，在其中实现一些常见的，用于集群和分类的数据挖掘算法。
  - 此后在发展中，又并入了更多广泛的数据挖掘方法

Mahout的特性

- 虽然在开源领域中较晚出现，但 Mahout 已经提供了大量功能
- 主要特性包括：
  - 支持 MapReduce 的集群实现包括 K-Means、模糊 K-Means、Canopy、Dirichlet 和 Mean-Shift。
  - Distributed Naive Bayes 和 Complementary Naive Bayes 分类实现。
  - 针对进化编程的分布式适用性功能。
  - Matrix 和矢量库。
  - 上述算法的示例。

使用 Mahout 实现集群算法

- Mahout 支持一些集群算法实现（都是使用 MapReduce 编写的），它们都有一组各自的目标和标准
- 以聚类为例，其提供了：
  - Canopy：一种快速集群算法，通常用于为其他集群算法创建初始种子。
  - K-Means（以及 模糊 K-Means）：根据项目与之前迭代的质心（或中心）之间的距离将项目添加到 k 集群中。
  - Mean-Shift：无需任何关于集群数量的 推理知识的算法，它可以生成任意形状的集群。
  - Dirichlet：借助基于多种概率模型的集群，它不需要提前执行特定的集群视图。

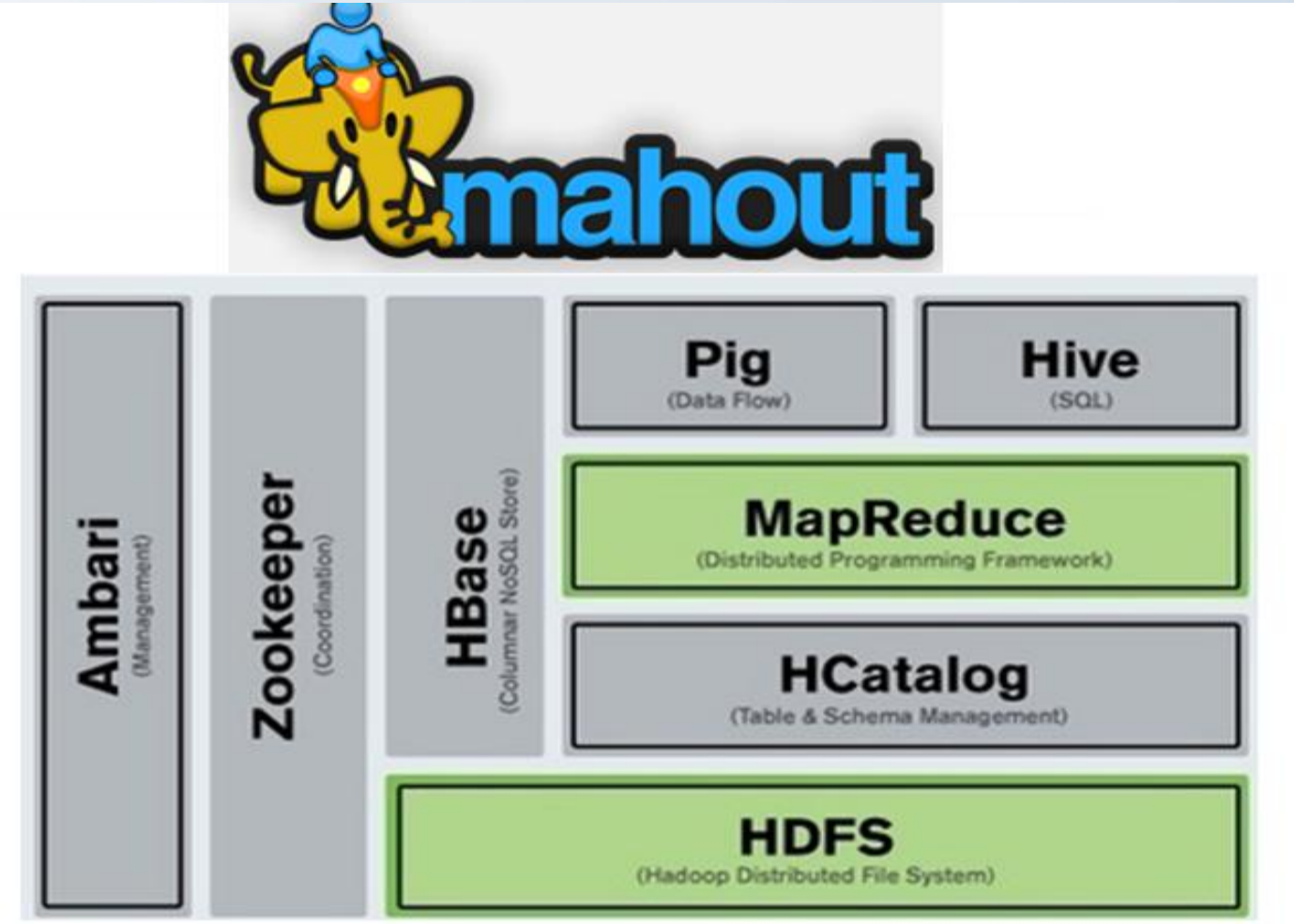
使用 Mahout 创建数据集群

- 具体的步骤包括：
  - 1.准备输入。如果创建文本集群，需要将文本转换成数值表示。
  - 2.使用 Mahout 中可用的 Hadoop 就绪的驱动程序运行所选集群算法。
  - 3.计算结果。
  - 4.如果有必要，执行迭代。

Mahout的发展

- Apache Mahout 为集群、分类和 CF(协同过滤)提供了许多重要的功能，但它还存在很大的发展空间。
  - MapReduce 的随机决策实现，它提供了分类、关联规则、用于识别文档主题的 Latent Dirichlet Allocation
  - 以及许多使用 HBase ，和其他辅助存储选项的类别选项。

Mahout与Hadoop家族其他主要成员关系



Mahout的基础

- Mahout提供了分布式的挖掘环境，具体讲：
  1 基于AFS Hadoop集群
  2 采用DFS分布式文件系统
  3 利用MapReduce 计算模型
  4 实现了一批开源的挖掘方法

Mahout 核心挖掘算法

- Mahout孵化了相当多的技术和算法，很多都是在开发和实验阶段。
- 有3个核心主题：
  - 协同过滤/推荐系统、聚类和分类。

Spark大数据挖掘工具MLlib

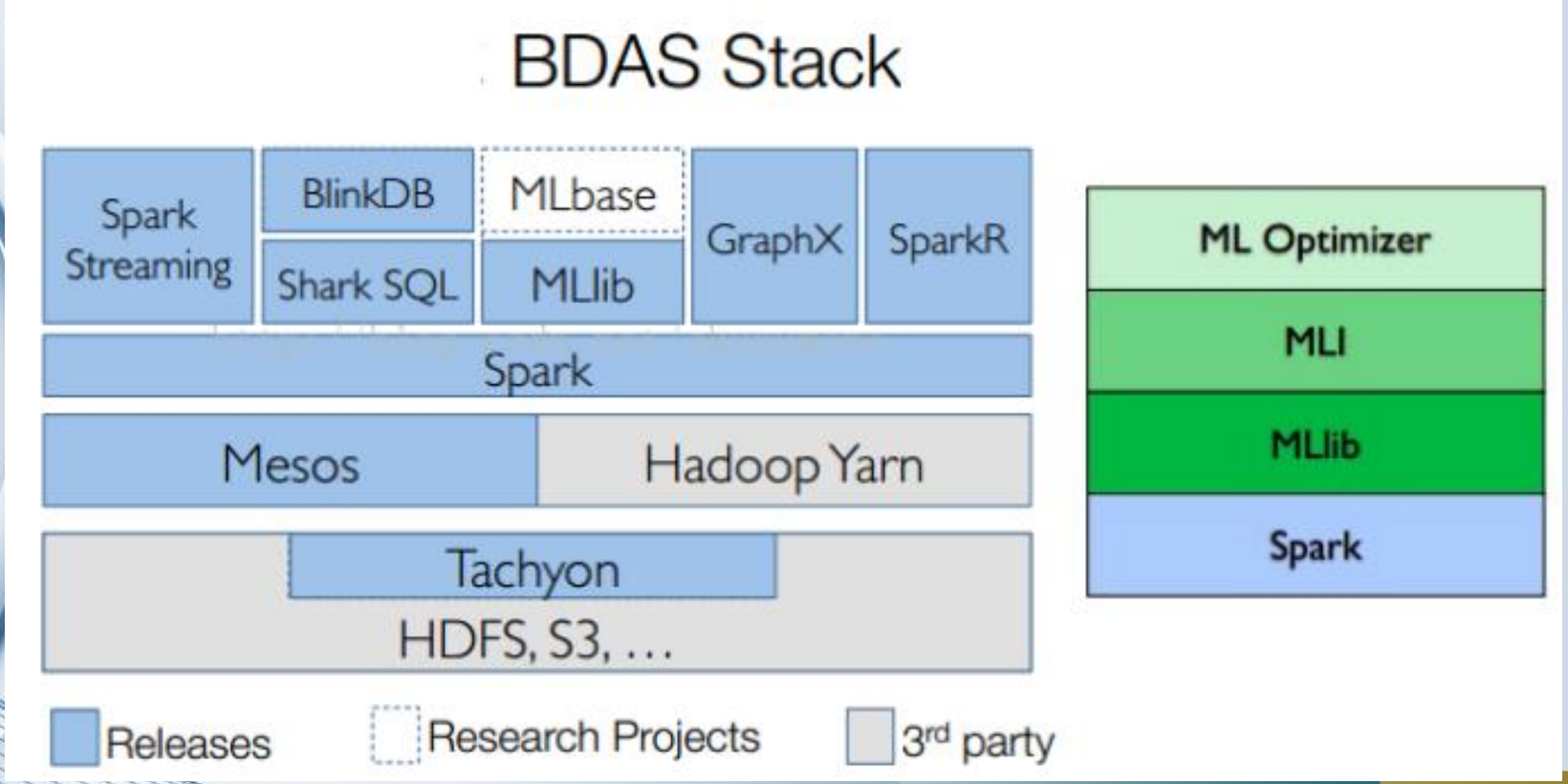
- MLlib是构建在Spark上的分布式数据挖掘工具，利用Spark的内存计算，和适合迭代型计算的优势，使性能大幅度提升。
- 同时Spark算子丰富的表现力，让大规模数据挖掘的算法开发不再复杂
- MLlib作为Spark其中一部分，目前已经完全包含入Spark中。

Spark MLlib

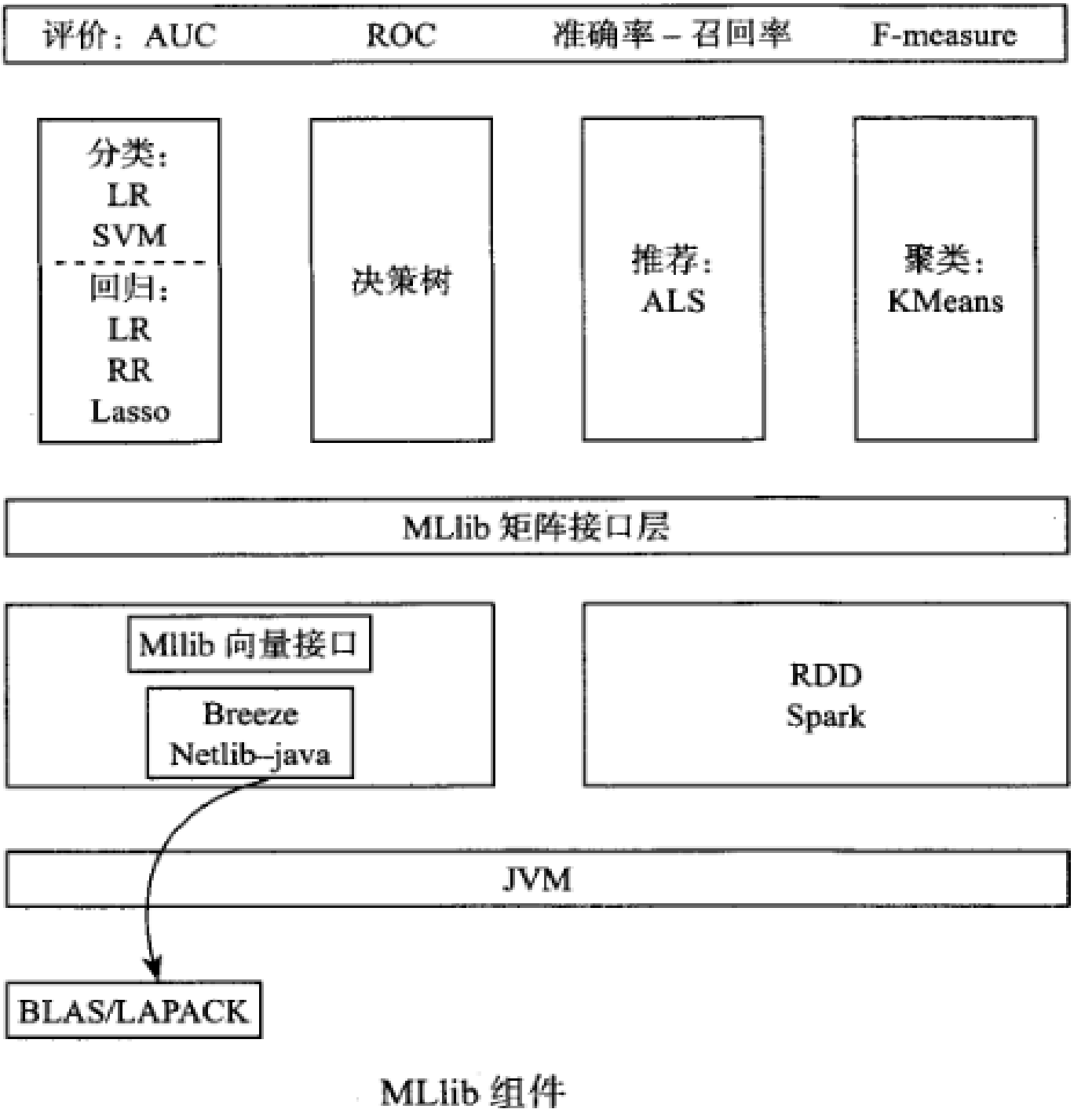
- 数据挖掘框架(Spark MLlib)
- MLlib是Spark对常用的数据挖掘算法的实现库，同时包括相关的测试和数据生成器：
- MLlib目前支持多种常见的数据挖掘问题：
  - 二元分类、回归、聚类以及协同过滤，同时也包括一个底层的梯度下降优化基础算法。
- 下面介绍MLlib中所支持的功能，后面将给出相应调用MLlib的例子。

MLlib介绍

- Spark数据挖掘有三大组件 ML Optimizer, MLI, Mlib。目前 ML Optimizer, MLI还不够完善。



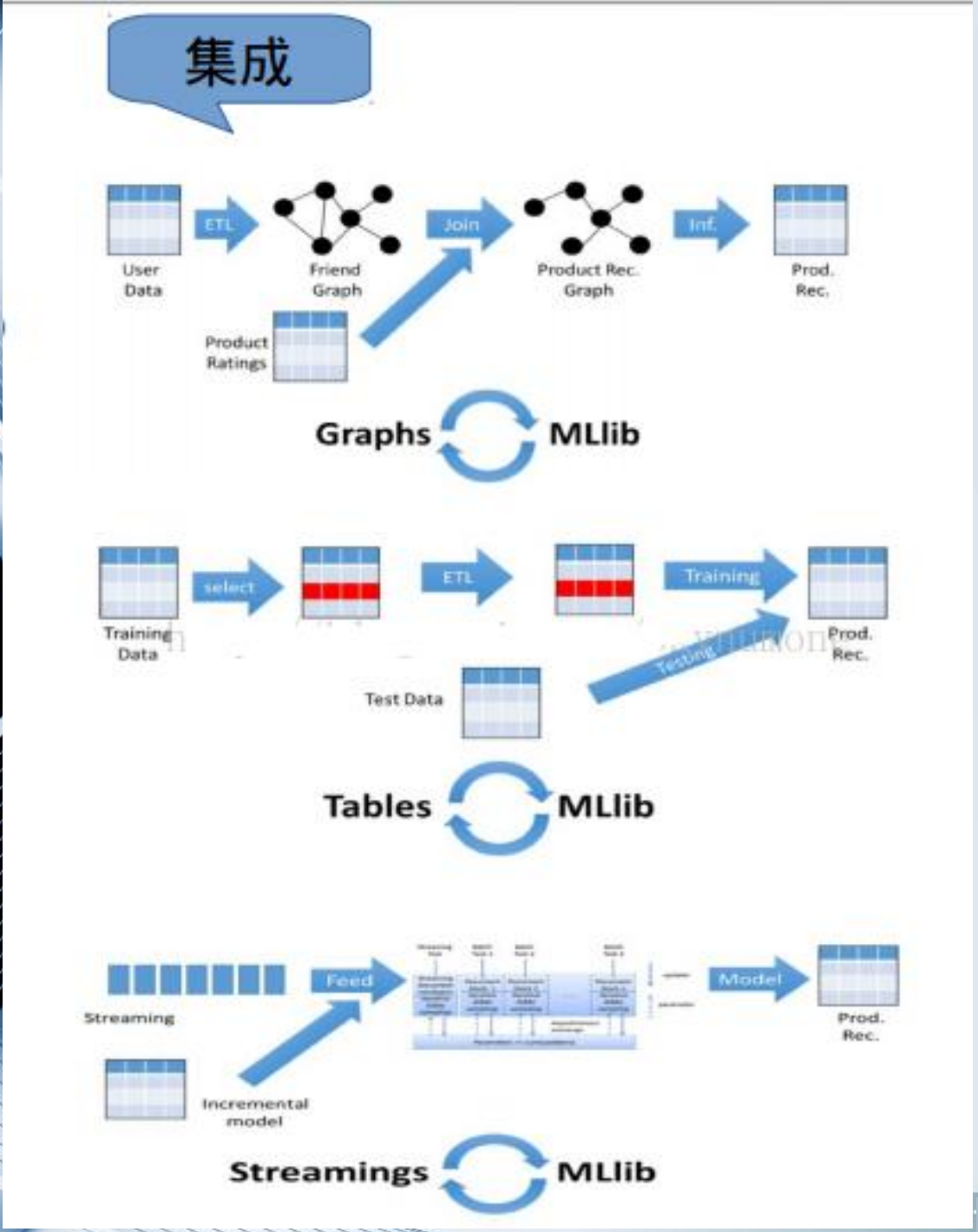
MLlib底层组件图



MLlib 组件

依赖

- MLlib将会调用jblas线性代数库，这个库本身依赖于原生的Fortran程序。
- 如果节点中没有这些库，需安装gfortran runtime library。
  - 如果程序没有办法自动检测到这些，MLlib将会抛出链接错误的异常。
  - 如果想用Python调用MLlib，需安装NumPy1.7或者更新的版本。



MLlib集成

- 可以与Spark其他组件进行集成

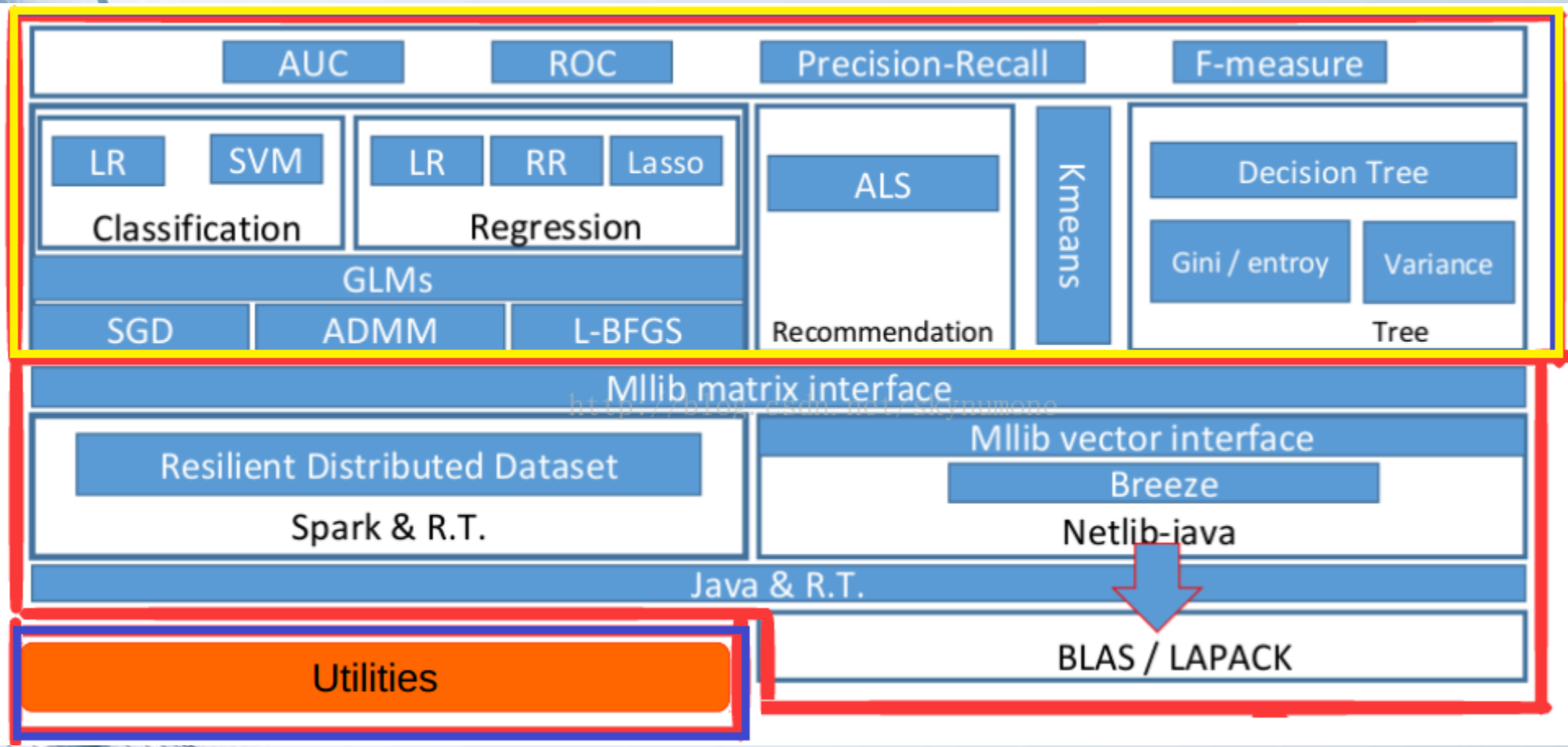
MLlib操作步骤

- MLlib操作步骤
- 第一步：读数据，进行向量化或者矩阵化
- 第二步：设置参数
- 第三步：进行模型训练
- 第四步：对数据进行预测
- 第五步：进行评估

MLlib 构成

- MLlib主要有三部分组成：
- 第一部分：实用程序部分(使用算法)
- 第二部分：基础部分(线性代数的运行库)
- 第三部分：算法部分(算法库)

MLlib 构成图

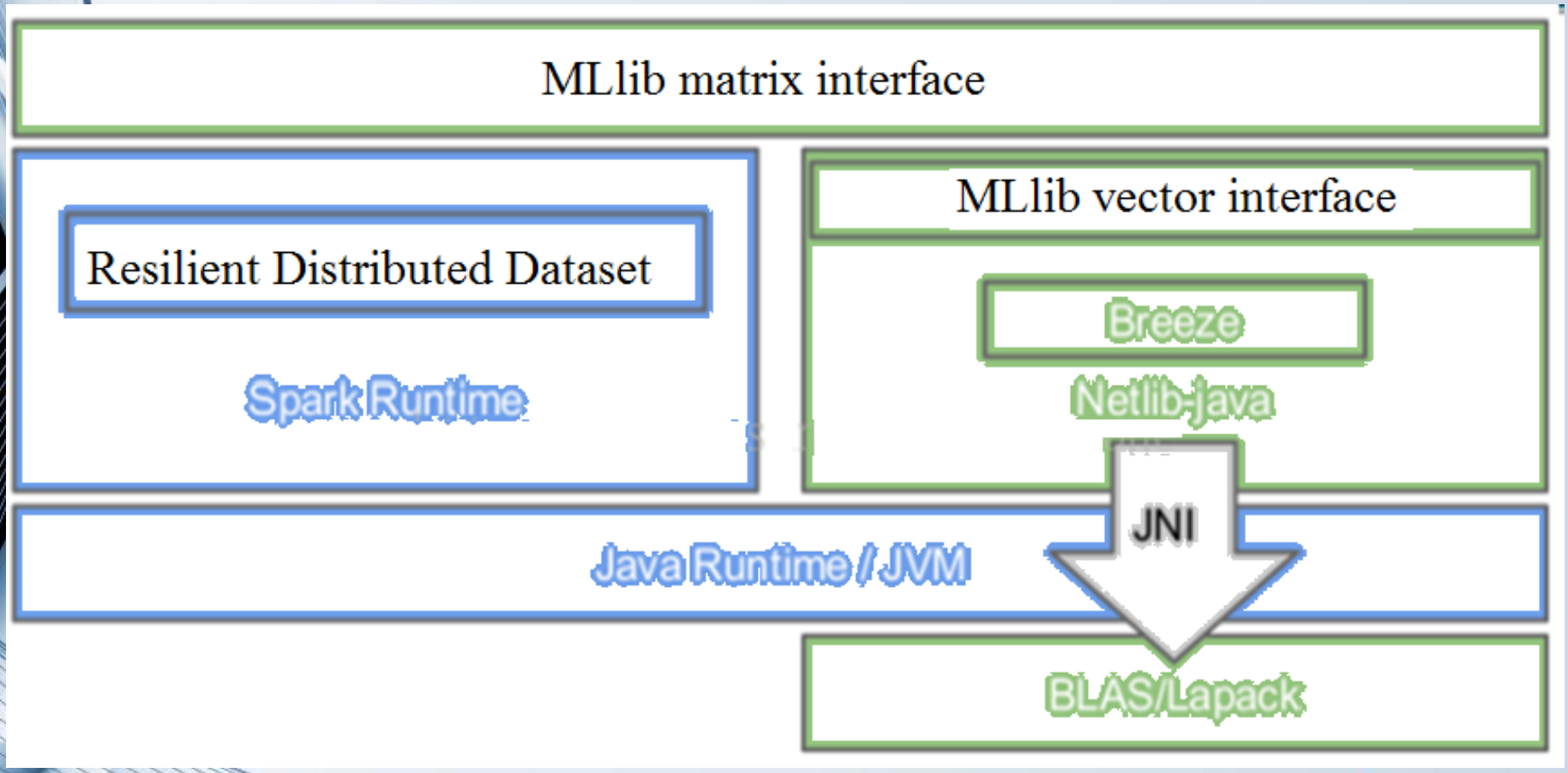

AUC
ROC
Precision-Recall
F-measure
LR
SVM
Classification
LR
RR
Lasso
Regression
GLMs
SGD
ADMM
L-BFGS
ALS
Recommendation
Kmeans
Decision Tree
Gini / entroy
Variance
Tree
Mllib matrix interface
Resilient Distributed Dataset
Spark & R.T.
Mllib vector interface
Breeze
Netlib-java
Java & R.T.
Utilities
BLAS / LAPACK

实用程序部分

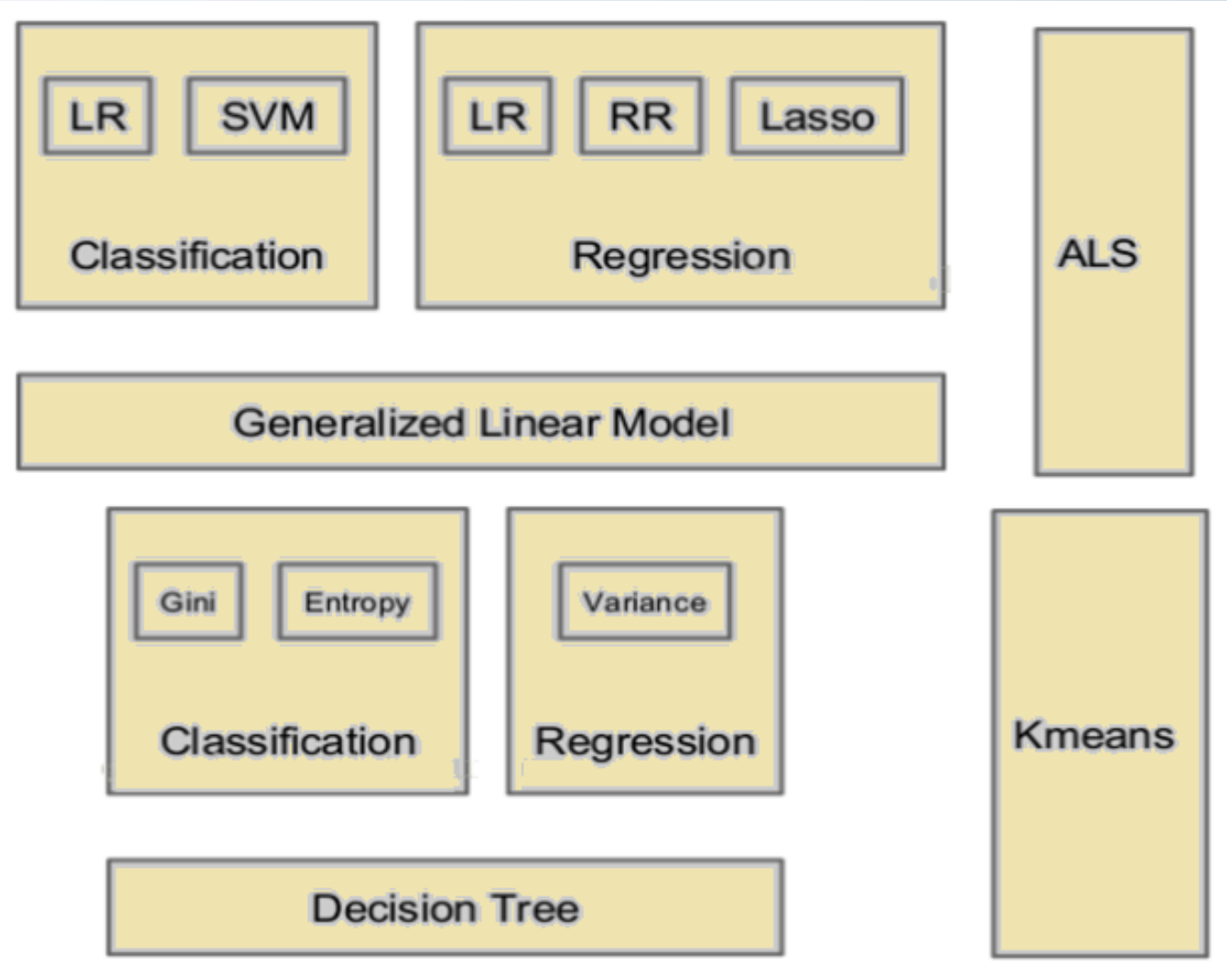
- Data validator: Binary label validator
- Label parser
  - Bin-class parser
  - Multi-class parser
- Several data generators
  - SVM generator
  - Matrix Factorization generator
  - Logistic Regression generator
  - Linear Regression generator
  - Kmeans generator
- Several dataset loaders:
  - labeledData
  - libSVMData
  - wholeTextFilesReader

```
val data: RDD[LabeledPoint] = MLUtils.loadLabeledData(sc, path)

val data: RDD[LabeledPoint] = MLUtils.loadLibSVMData(sc, path)

val data: RDD[LabeledPoint] =
  MLUtils.loadLibSVMData(sc, path , labelParser)

val data: RDD[LabeledPoint] =
  MLUtils.loadLibSVMData(sc, path, labelParser, numFeatures)

val data: RDD[LabeledPoint] =
  MLUtils.loadLibSVMData(sc, path , labelParser , numFeatures , minSplits)

val data: RDD[(String, String)] = sc.wholeTextFiles(dirPath , minSplits)
```

基础部分


MLlib matrix interface
Resilient Distributed Dataset
Spark Runtime
MLlib vector interface
Breeze
Netlib-java
JNI
Java Runtime / JVM
BLAS/Lapack

算法部分


LR
SVM
Classification
LR
RR
Lasso
Regression
ALS
Generalized Linear Model
Gini
Entropy
Classification
Variance
Regression
Kmeans
Decision Tree

MLlib的数据存储

- MLlib支持存储在本地的向量和矩阵，也提供分布式的矩阵(底层实现是一个或多个RDD)。
  - 目前版本中，本地的向量和矩阵数据模型，提供公共服务接口。

Vector类的两种实现

- 本地向量的基本类是Vector类，官方提供了Vector类的两种实现：
  - 稠密向量(dense vector)和稀疏向量(sparse vector)。
- 官方推荐使用Vectors类中，提供的工厂模式的方法，创建本地向量。

两种向量存储模式例子

- 稀疏向量只列出非0元素的 数值 及其 索引值。
- 稀疏向量不但能够节省空间，还能够提升计算性能

$$\text{dense}: \underbrace{1.}\ \underbrace{0.}\ \underbrace{0.}\ \underbrace{0.}\ \underbrace{0.}\ \underbrace{0.}\ \underbrace{3.}$$

$$\text{sparse}: \begin{cases} \text{size}: 7 \\ \text{indices}: \underbrace{0}\ \underbrace{6} \\ \text{values}: \underbrace{1.}\ \underbrace{3.} \end{cases}$$

Training set:

- 12 million examples
- 500 features
- sparsity: 10%

	dense	sparse
storage	47GB	7GB
time	240s	58s

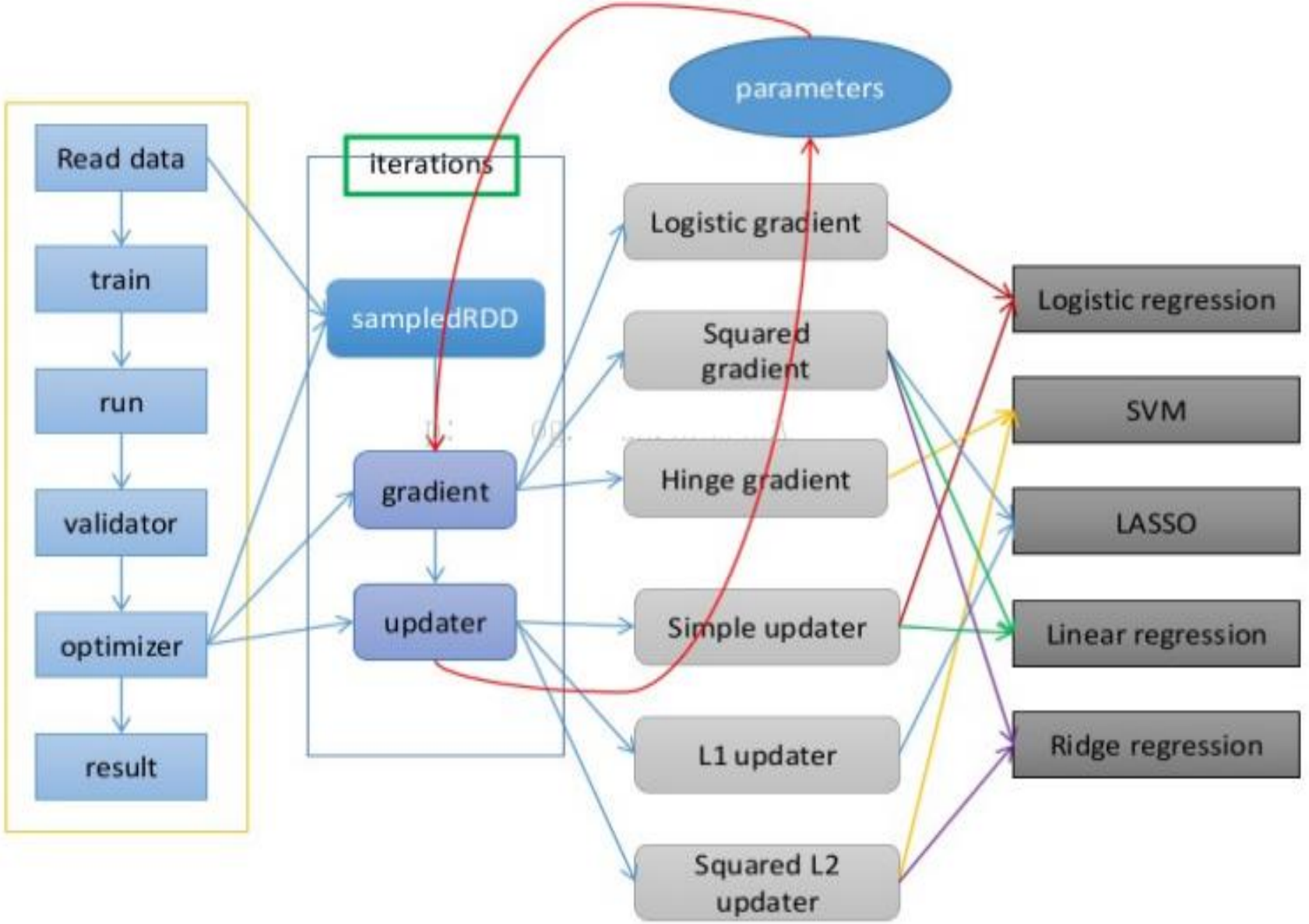
40GB savings in storage, 4x speedup in computation

稀疏数据

- 一般用稀疏数据训练模型，下面其数据格式。
  - MLlib支持读取LIBSVM格式(一种文本格式)的训练数据，这种数据默认被LIBSVM和LIBLINEAR使用。
  - 文件的每一行，代表一个被标记的稀疏特征向量，格式请参考：
    - label index1:value1　index2:value2…
  - 默认序号索引是从1开始，并且是升序的，加载后，特征的序号被转换为从0开始。
  - 可使用MLUtils.loadLibSVMFile方法读取存储为LIBSVM格式的训练数据

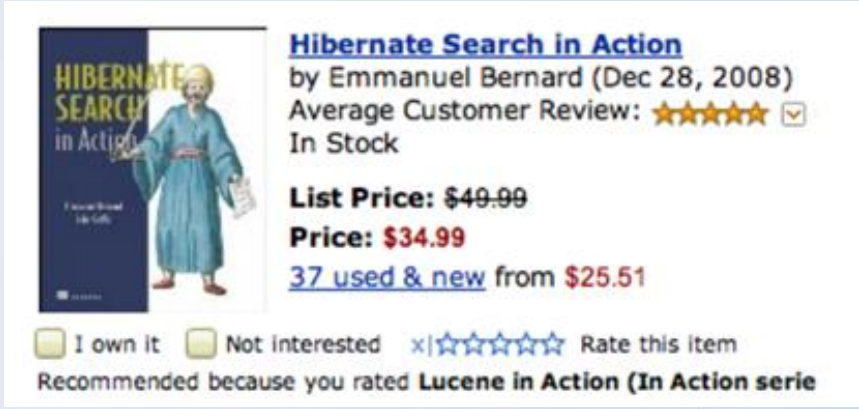
How Spark

MLlib 运行结构



推荐系统

- 推荐系统是目前使用的系统中最普及的
  - 相关的服务或网页，包括基于历史行为推荐书、电影、文档。
  - 尝试推论出用户偏好，并标记出用户不知晓的、感兴趣的item

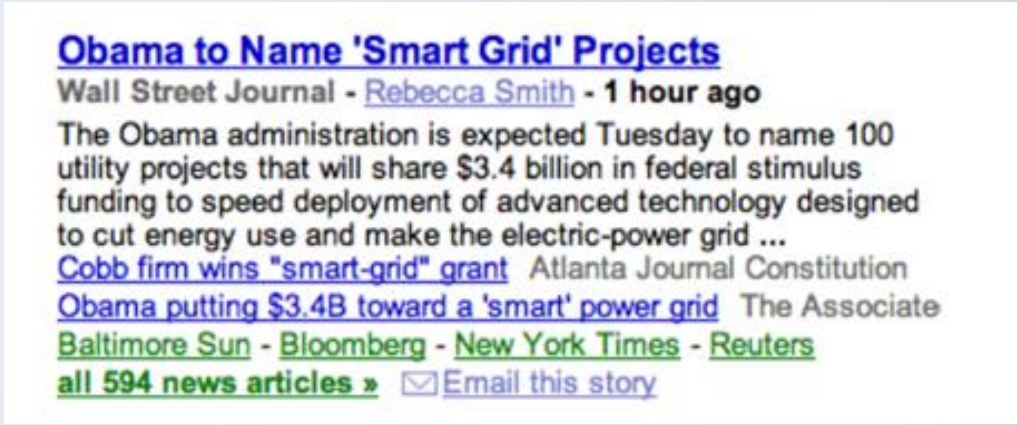


应用实例

- Amazon.com是最出名的使用推荐系统商务网站。基于交易和网页活性，Amazon推荐给用户可能感兴趣的书籍和其他item。
- Netflix类似于推荐用户感兴趣的DVDs，并且为研究者提供百万大奖去提升推荐质量。
- 约会网站像L Ь ́mseti将一部分用户推荐给其他用户。
- 社交网络网站像Facebook，用推荐技术的变形来为用户识别最可能建立联系的朋友

聚类

- 聚类技术尝试将大量拥有相同相似度的事物，聚集到不同的类中。
  - 聚类有助于在海量的、很难弄懂的事物集合中，发现结构，甚至层次。
  - 可以使用聚类，根据网站日志发现用户的经常使用模式



应用实例

- Google News可根据具备逻辑性的故事，使用新闻文章的Topic聚集新闻，而不是文章的列表。
  - 搜索引擎(像Clusty)基于相同的方法，聚集搜索结果。
- 使用聚类技术，基于消费者属性，收入、位置、购买习惯，可将不用用户分到不用的类中

分类

- 分类技术用于决定一个事物，是不是属于一种类型、类目，或者该事物是不是含有某些属性。
  - 分类有助于判断一个新进入事物，是否匹配先前发现的模式， 也常用于分类行为或者模式。
  - 分类也可用来检测可疑的网络活动或欺诈。也可根据用户发的信息，判定表示失望或者满意

应用实例

- Yahoo!：Mail决定接收的信息是不是垃圾邮件，基于先前邮件和用户的垃圾邮件报告，以及邮件的特性。一些信息被分类为垃圾邮件

- Picasa (http://picasa.google.com/)和其他的照片管理应用可以判断一张照片中是否含有人脸。
- 光学字符识别软件：通过将小区域作为独立字符来分类，将扫描文本的若干小区域归类到独立的字符上

谢　谢