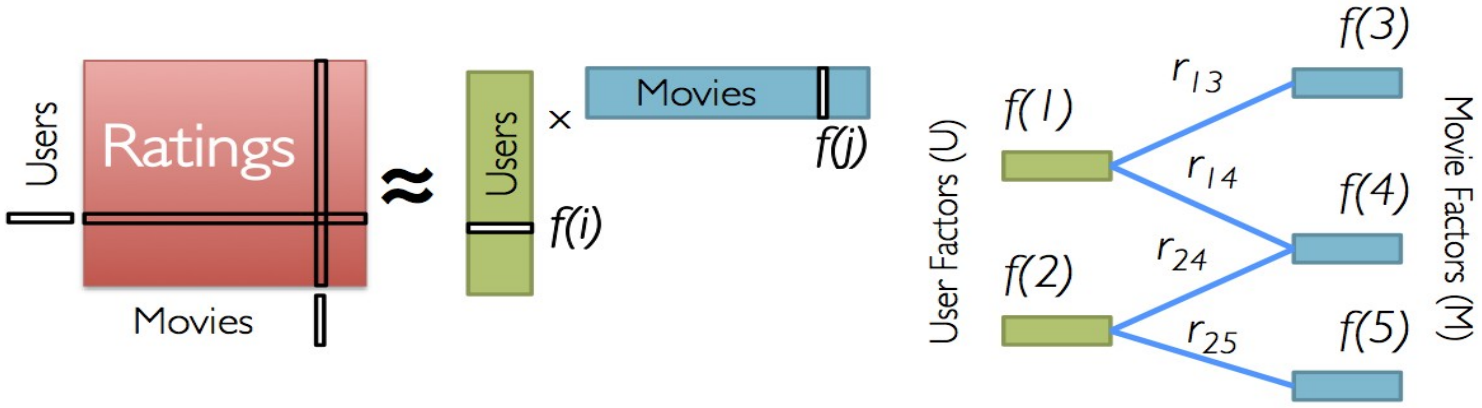
Low-Rank Matrix Factorization:



Iterate:

$$f[i] = \arg\min_{w \in \mathbb{R}^d} \sum_{j \in \mathrm{Nbrs}(i)} \left(r_{ij} - w^T f[j]\right)^2 + \lambda ||w||_2^2$$

第六课： Spark 机器学习入门

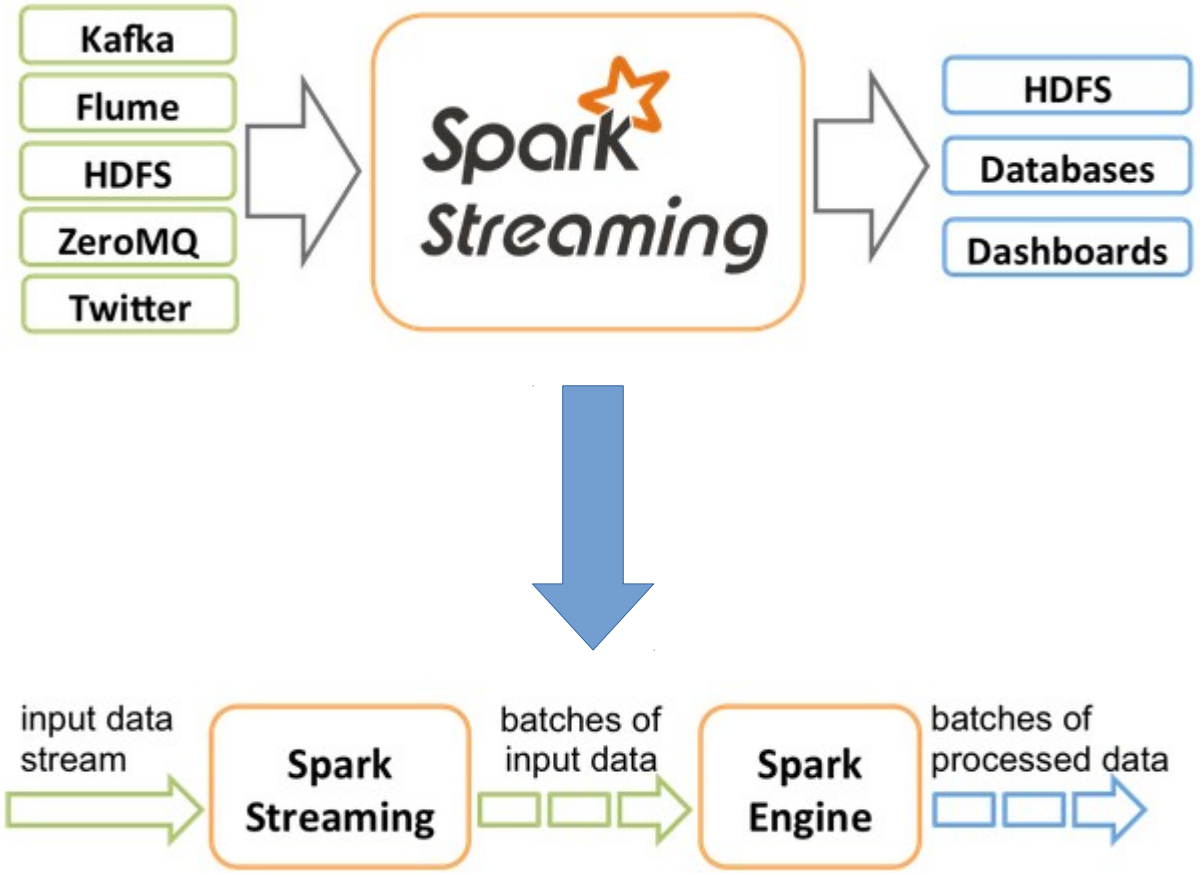
法律声明

【声明】本视频和幻灯片为炼数成金网络课程的教学资料，所有资料只能在课程内使用，不得在课程以外范围散播，违者将可能被追究法律和经济责任。

课程详情访问炼数成金培训网站

http://edu.dataguru.cn

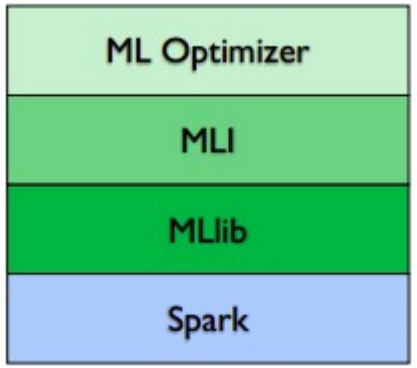
上周回顾



- SparkStreaming
  - 无状态操作
  - 状态操作
  - windows 操作

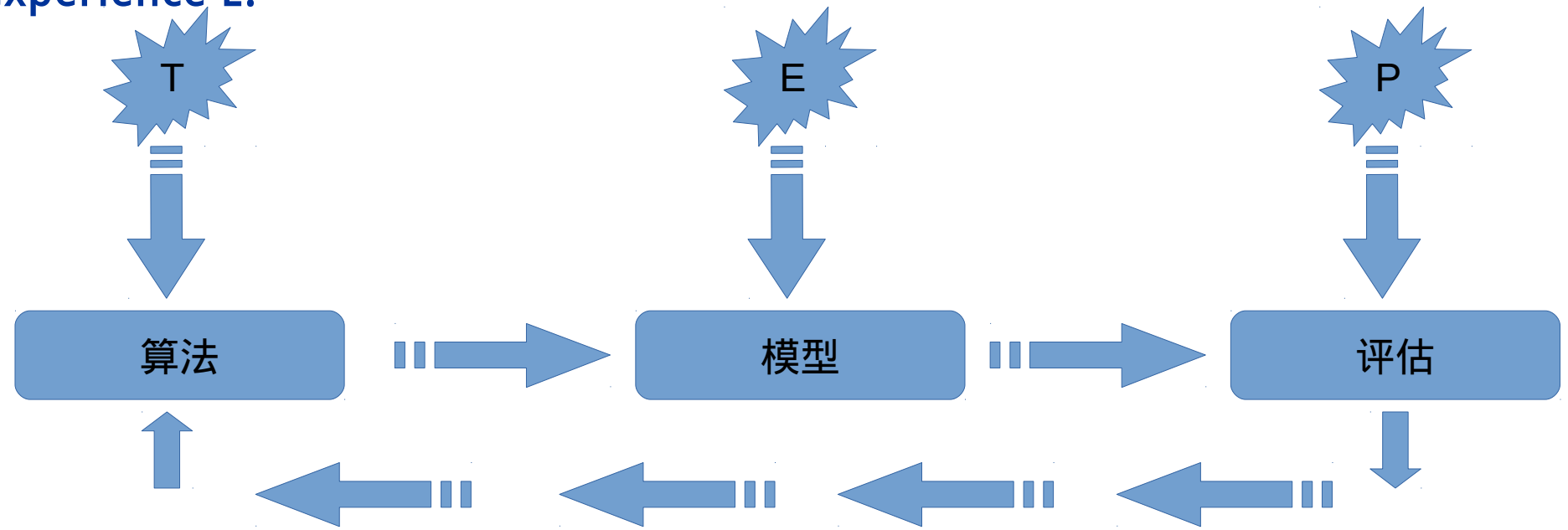
本周内容

- 机器学习
  - 定义
  - 分类
  - 常用算法
- Mllib
  - 什么是 MLlib
  - MLlib 构成
  - MLlib 的运行架构
- 实例演示
  - K-Means 算法介绍和实例
  - 协同过滤 算法介绍和实例



什么是机器学习

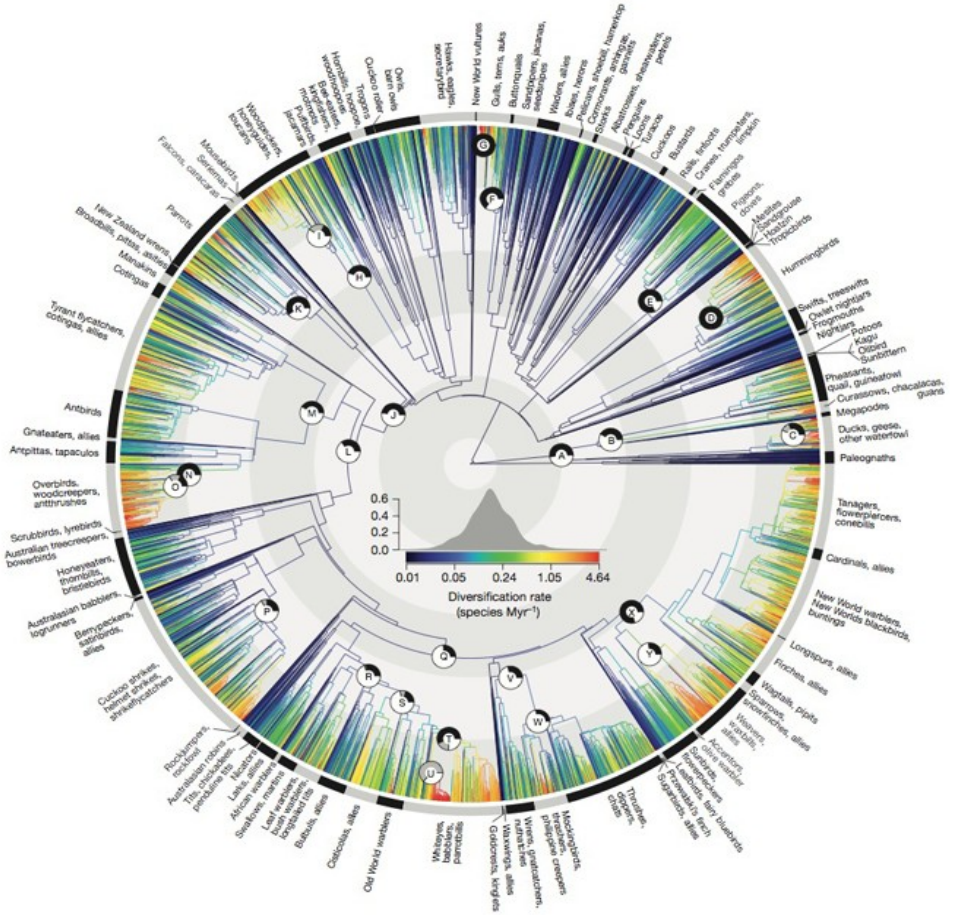
- 机器学习是一门人工智能的科学，该领域的主要研究对象是人工智能，特别是如何在经验学习中改善具体算法的性能。
- 机器学习是对能通过经验自动改进的计算机算法的研究。
- 机器学习是用数据或以往的经验，以此优化计算机程序的性能标准。
- A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E.



什么是机器学习

- 机器学习应用十分广泛
  - 数据挖掘
  - 计算机视觉
  - 自然语言处理
  - 生物特征识别
  - 搜索引擎
  - 医学诊断
  - 检测信用卡欺诈
  - 证券市场分析
  - DNA 序列测序
  - 语音和手写识别
  - 战略游戏和机器人运用

2012 最佳科技图表：
http://shijue.me/show_idea/50d129c48ddf872e41000252



机器学习分类

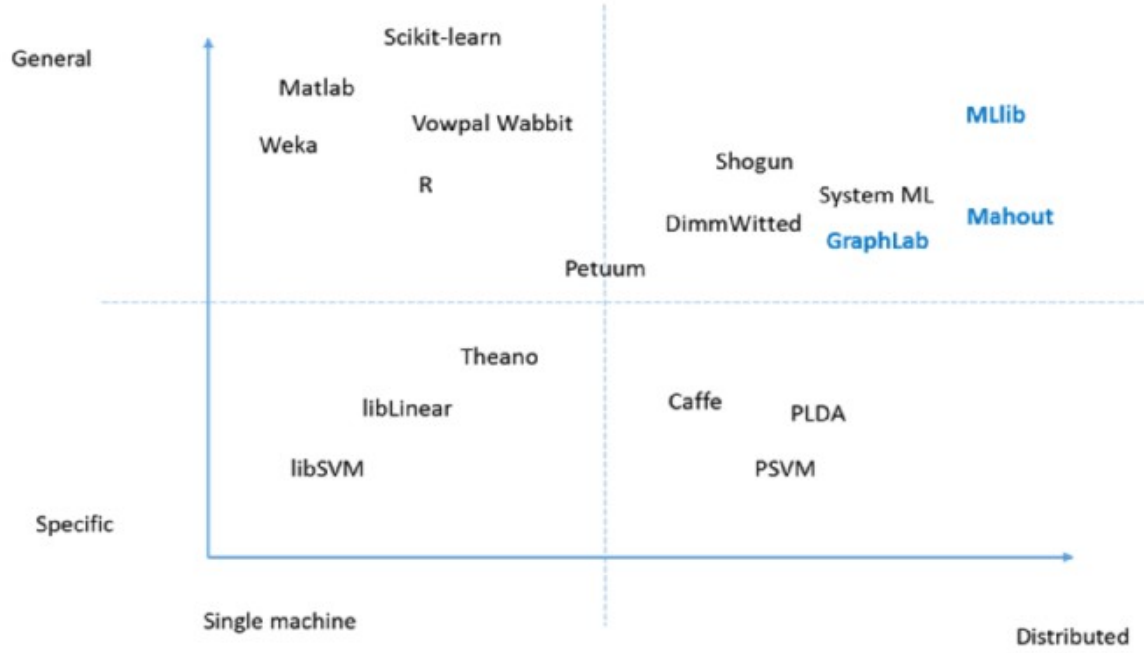
- 监督学习从给定的训练数据集中学习出一个函数（模型），当新的数据到来时，可以根据这个函数（模型）预测结果。监督学习的训练集要求是包括输入和输出，也可以说是特征和目标。训练集中的目标是由人标注（标量）的。常见的监督学习算法包括回归分析和统计分类。
  - 二元分类是 ML 要解决的基本问题，将测试数据分成两个类。如垃圾邮件的判别、房贷是否允许等等问题的判断。
  - 多元分类是二元分类的逻辑延伸。例如，在因特网的流分类的情况下，根据问题的网页可以被归类为体育，新闻，技术，或成人 / 色情，依此类推。
- 无监督学习与监督学习相比，训练集没有人为标注的结果。常见的无监督学习算法有聚类。
- 半监督学习介于监督学习与无监督学习之间。
- 增强学习通过观察来学习做成如何的动作。每个动作都会对环境有所影响，学习对象根据观察到的周围环境的反馈来做出判断。

机器学习常用算法

具体的机器学习算法有：

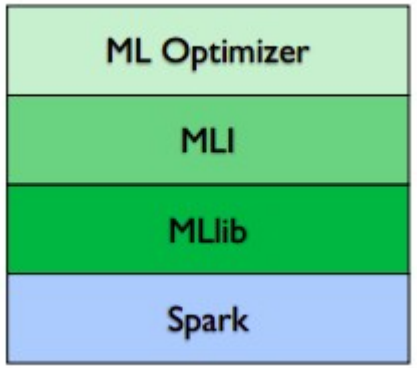
- 构造条件概率：回归分析和统计分类
  - 人工神经网络
  - 决策树（Decision tree）
  - 高斯过程回归
  - 线性判别分析
  - 最近邻居法
  - 感知器
  - 径向基函数核
  - 支持向量机
- 通过再生模型构造概率密度函数（Probability density function）：
  - 最大期望算法（Expectation-maximization algorithm）
  - graphical model：包括贝叶斯网和Markov随机场
  - Generative Topographic Mapping
- 近似推断技术：
  - 马尔可夫链（Markov chain）蒙特卡罗方法
  - 变分法
- 最优化（Optimization）：大多数以上方法，直接或者间接使用最优化算法。

一个数据集可以采用多种算法来进行机器学习



本周内容

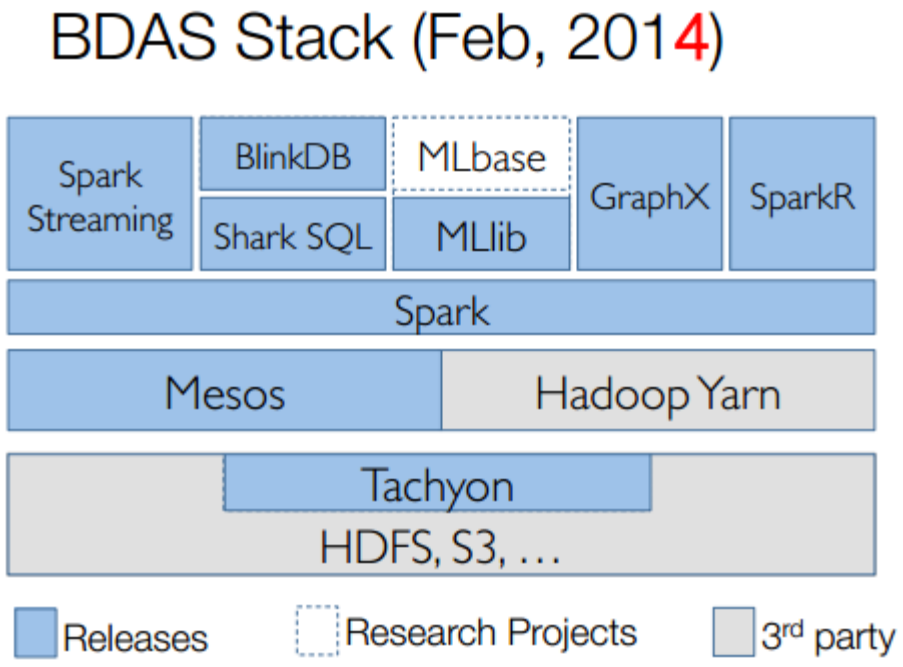
- 机器学习
  - 定义
  - 分类
  - 常用算法
- Mllib
  - 什么是 MLlib
  - MLlib 构成
  - MLlib 的运行架构
- 实例演示
  - K-Means 算法介绍和实例
  - 协同过滤 算法介绍和实例

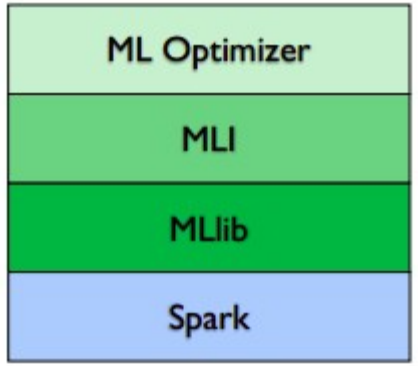


什么是 MLlib

What is Spark MLlib

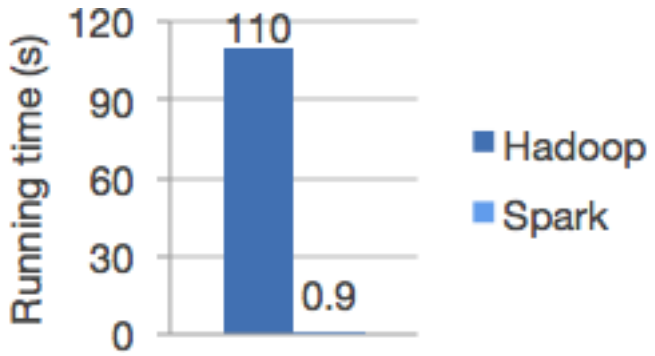
- A fast engine for parallel machine learning.



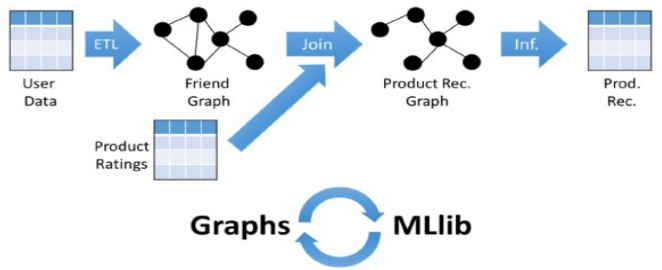


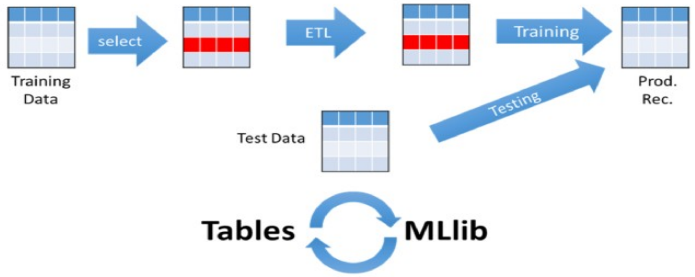
为什么 MLlib

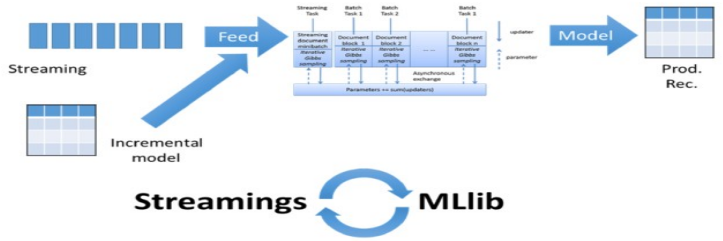
性能



集成



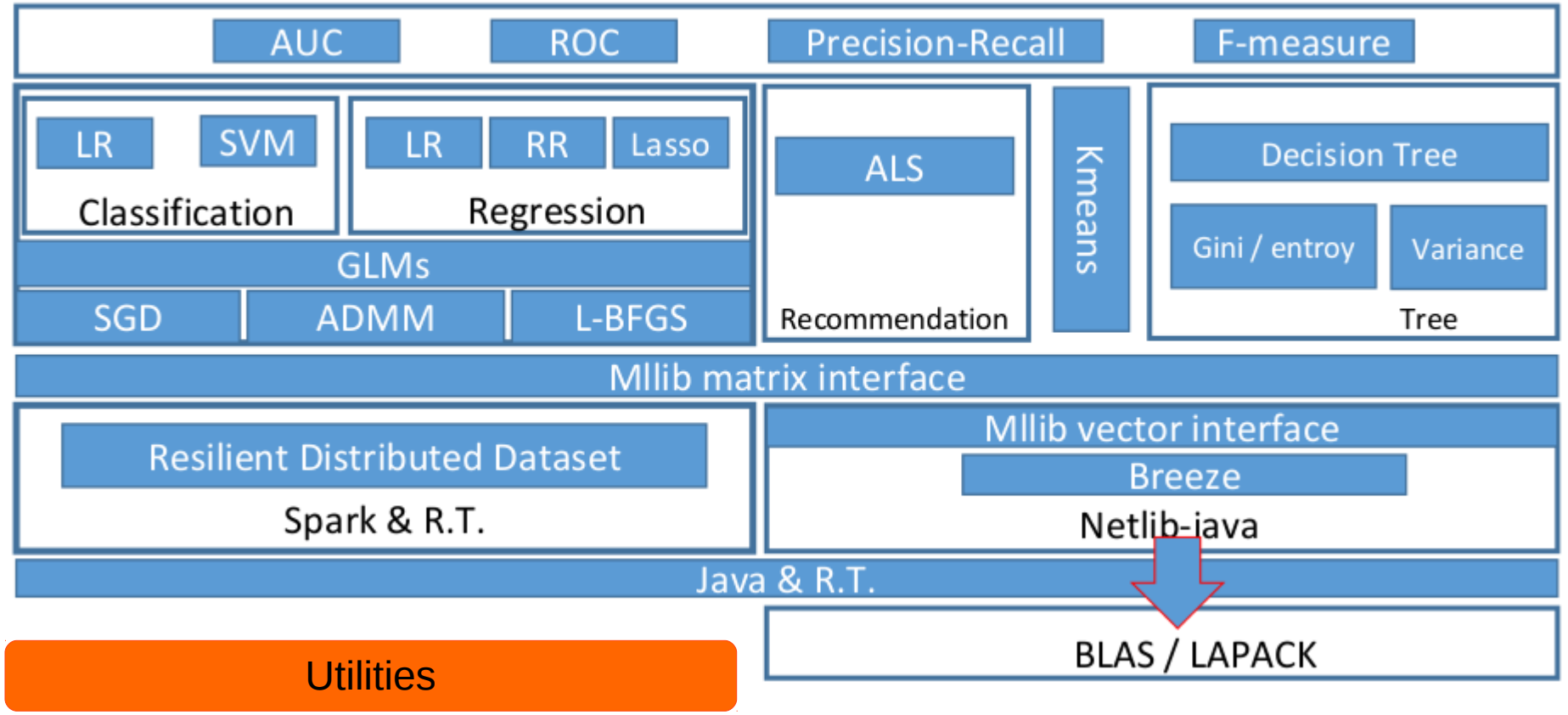




易用

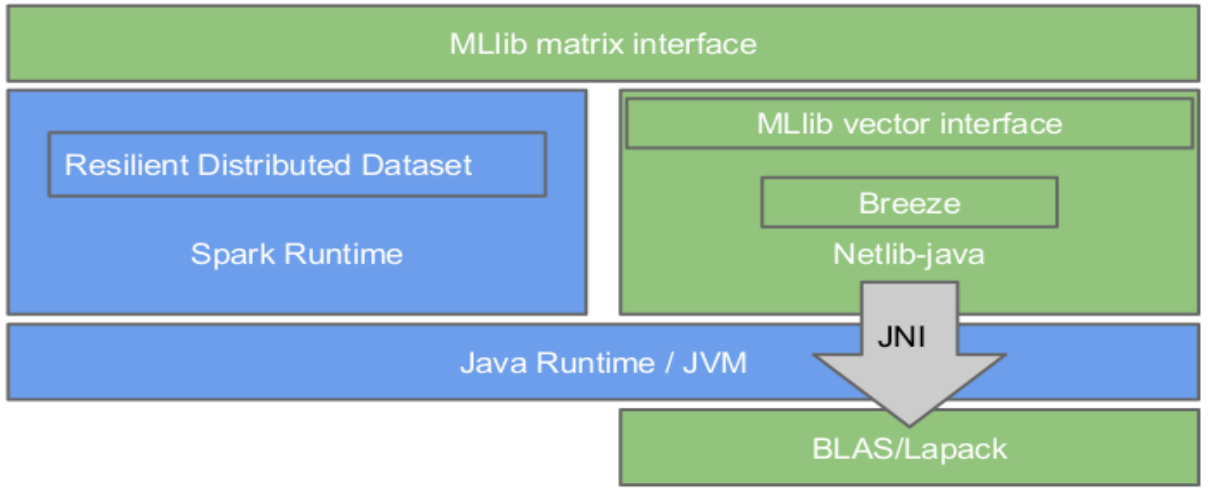
```scala
// Load and parse the data
val data = sc.textFile("mllib/data/ridge-data/lpsa.data")
val parsedData = data.map { line =>
 val parts = line.split(',')
 LabeledPoint(parts(0).toDouble, Vectors.dense(parts(1).split(' ').map(_.toDouble)))
}

// Building the model
val numIterations = 100
val model = LinearRegressionWithSGD.train(parsedData, numIterations)

// Evaluate model on training examples and compute training error
val valuesAndPreds = parsedData.map { point =>
 val prediction = model.predict(point.features)
 (point.label, prediction)
}
val MSE = valuesAndPreds.map{case(v, p) => math.pow((v - p), 2)}.mean()
println("training Mean Squared Error = " + MSE)
```

MLlib 构成


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.
BLAS / LAPACK
Utilities

MLlib 构成

- 基础部分
  - Local vector
    - DenseVector
    - SparseVector
  - Labeled point
  - Local matrix
  - Distributed matrix
    - RowMatrix
      - Multivariate summary statistics
    - IndexedRowMatrix
    - CoordinateMatrix



MLlib 构成

Vector

```scala
import org.apache.spark.mllib.linalg.{Vector, Vectors}

// Create a dense vector (1.0, 0.0, 3.0).
val dv: Vector = Vectors.dense(1.0, 0.0, 3.0)
// Create a sparse vector (1.0, 0.0, 3.0) by specifying its indices and values corresponding to nonzero entries.
val sv1: Vector = Vectors.sparse(3, Array(0, 2), Array(1.0, 3.0))
// Create a sparse vector (1.0, 0.0, 3.0) by specifying its nonzero entries.
val sv2: Vector = Vectors.sparse(3, Seq((0, 1.0), (2, 3.0)))
```

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

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

```scala
import org.apache.spark.mllib.linalg.Vectors
import org.apache.spark.mllib.regression.LabeledPoint

// Create a labeled point with a positive label and a dense feature vector.
val pos = LabeledPoint(1.0, Vectors.dense(1.0, 0.0, 3.0))

// Create a labeled point with a negative label and a sparse feature vector.
val neg = LabeledPoint(0.0, Vectors.sparse(3, Array(0, 2), Array(1.0, 3.0)))
```

Double

Training set:

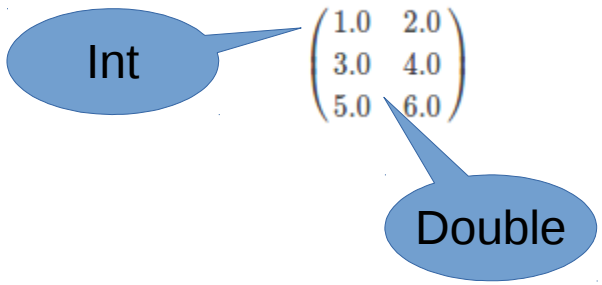
- 12 million examples
- 500 features
- sparsity: 10%

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

40GB savings in storage, 4x speedup in computation

MLlib 构成

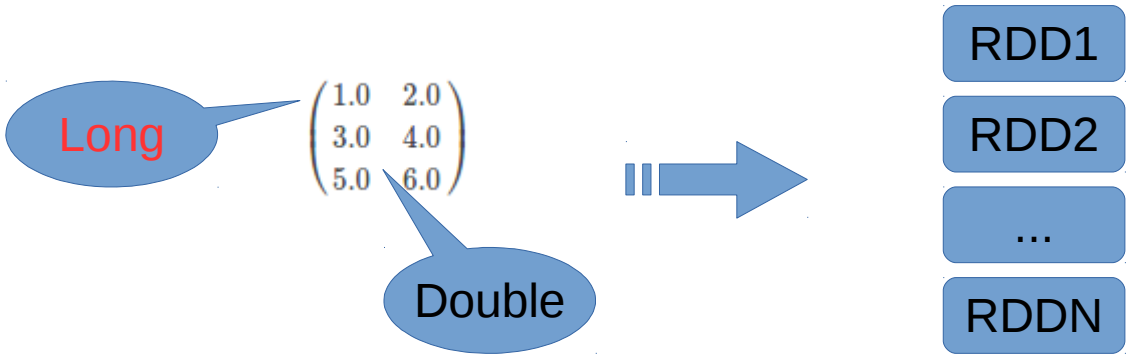
Local Matrix




```
import org.apache.spark.mllib.linalg.{Matrix, Matrices}

// Create a dense matrix ((1.0, 2.0), (3.0, 4.0), (5.0, 6.0))
val dm: Matrix = Matrices.dense(3, 2, Array(1.0, 3.0, 5.0, 2.0, 4.0, 6.0))
```

MLlib 构成

- Distributed Matrix



MLlib 构成

- RowMatrix

```scala
import org.apache.spark.mllib.linalg.Vector
import org.apache.spark.mllib.linalg.distributed.RowMatrix

val rows: RDD[Vector] = ... // an RDD of local vectors
// Create a RowMatrix from an RDD[Vector].
val mat: RowMatrix = new RowMatrix(rows)

// Get its size.
val m = mat.numRows()
val n = mat.numCols()
```

```scala
import org.apache.spark.mllib.linalg.Matrix
import org.apache.spark.mllib.linalg.distributed.RowMatrix
import org.apache.spark.mllib.stat.MultivariateStatisticalSummary

val mat: RowMatrix = ... // a RowMatrix

// Compute column summary statistics.
val summary: MultivariateStatisticalSummary = mat.computeColumnSummaryStatistics()
println(summary.mean) // a dense vector containing the mean value for each column
println(summary.variance) // column-wise variance
println(summary.numNonzeros) // number of nonzeros in each column

// Compute the covariance matrix.
val cov: Matrix = mat.computeCovariance()
```

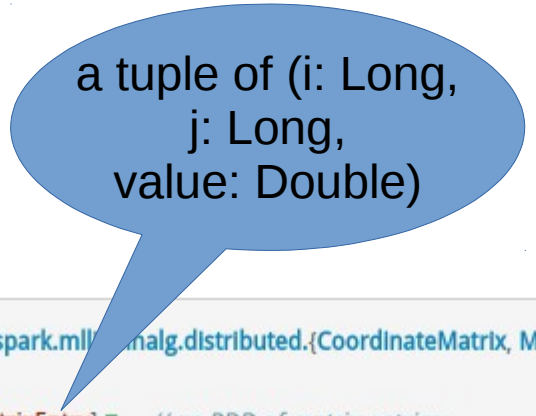
MLlib 构成

- IndexedRowMatrix

```scala
import org.apache.spark.mllib.linalg.distributed.{IndexedRow, IndexedRowMatrix, RowMatrix}

val rows: RDD[IndexedRow] = ... // an RDD of indexed rows
// Create an IndexedRowMatrix from an RDD[IndexedRow].
val mat: IndexedRowMatrix = new IndexedRowMatrix(rows)

// Get its size.
val m = mat.numRows()
val n = mat.numCols()

// Drop its row indices.
val rowMat: RowMatrix = mat.toRowMatrix()
```

MLlib 构成

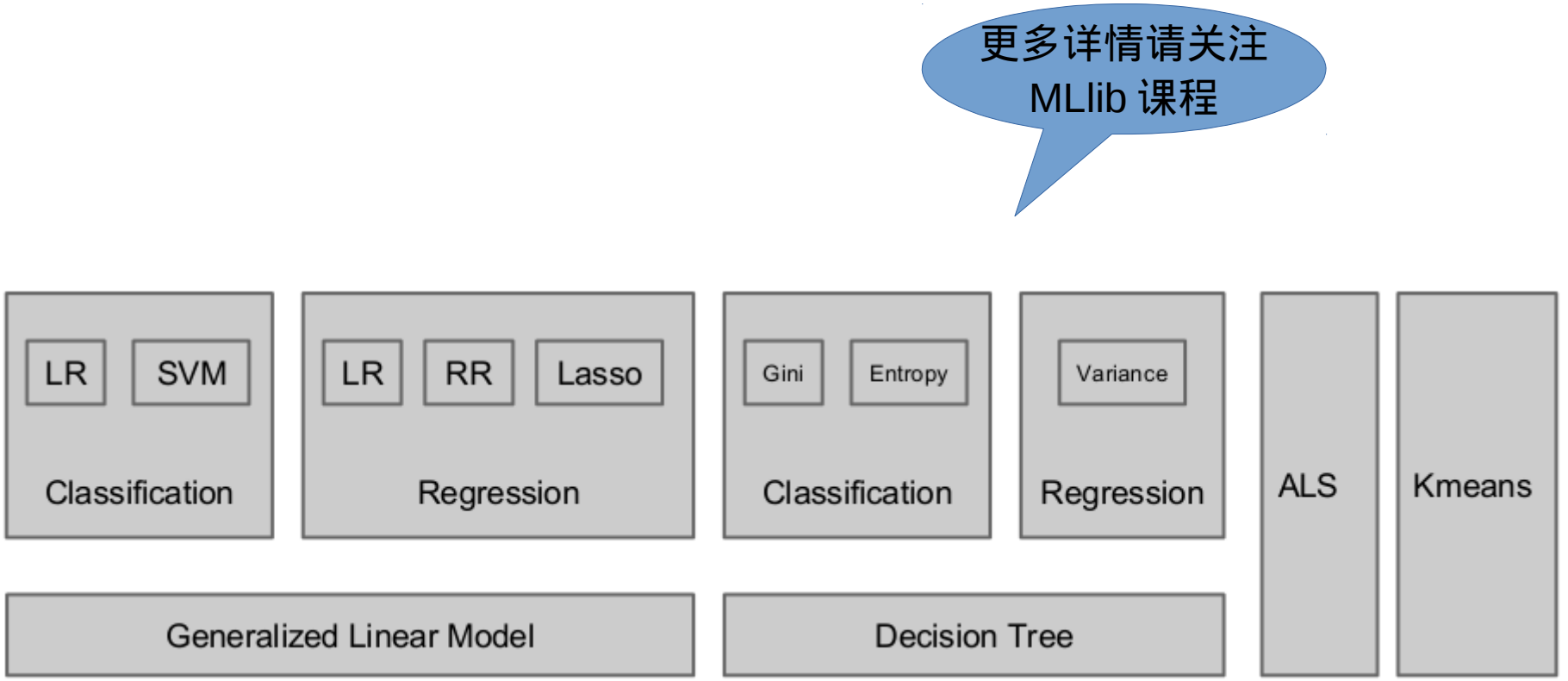
CoordinateMatrix




```
Import org.apache.spark.mllib.linalg.distributed.{CoordinateMatrix, MatrixEntry}

val entries: RDD[MatrixEntry] = ... // an RDD of matrix entries
// Create a CoordinateMatrix from an RDD[MatrixEntry].
val mat: CoordinateMatrix = new CoordinateMatrix(entries)

// Get its size.
val m = mat.numRows()
val n = mat.numCols()

// Convert it to an IndexRowMatrix whose rows are sparse vectors.
val indexedRowMatrix = mat.toIndexedRowMatrix()
```

MLlib 构成

- 算法部分



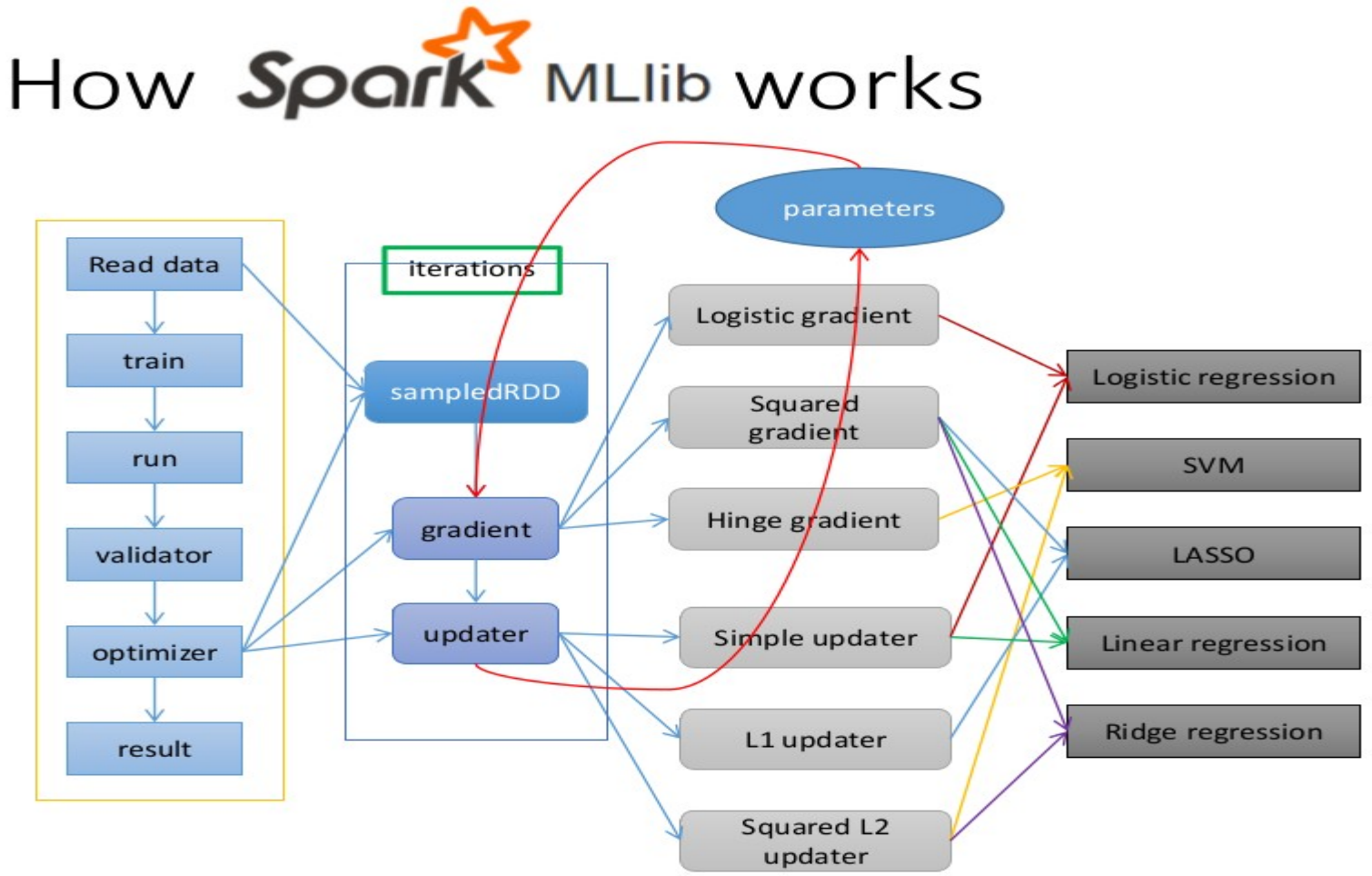
MLlib 构成

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)
```

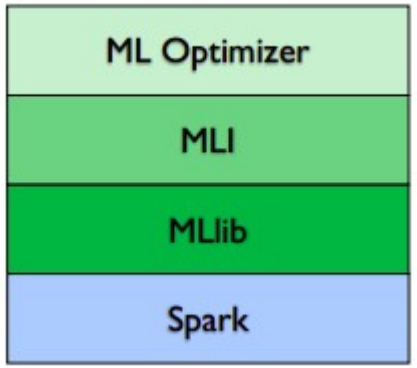
MLlib 运行结构


How Spark MLlib works
Read data
train
run
validator
optimizer
result
iterations
sampledRDD
gradient
updater
parameters
Logistic gradient
Squared gradient
Hinge gradient
Simple updater
L1 updater
Squared L2 updater
Logistic regression
SVM
LASSO
Linear regression
Ridge regression

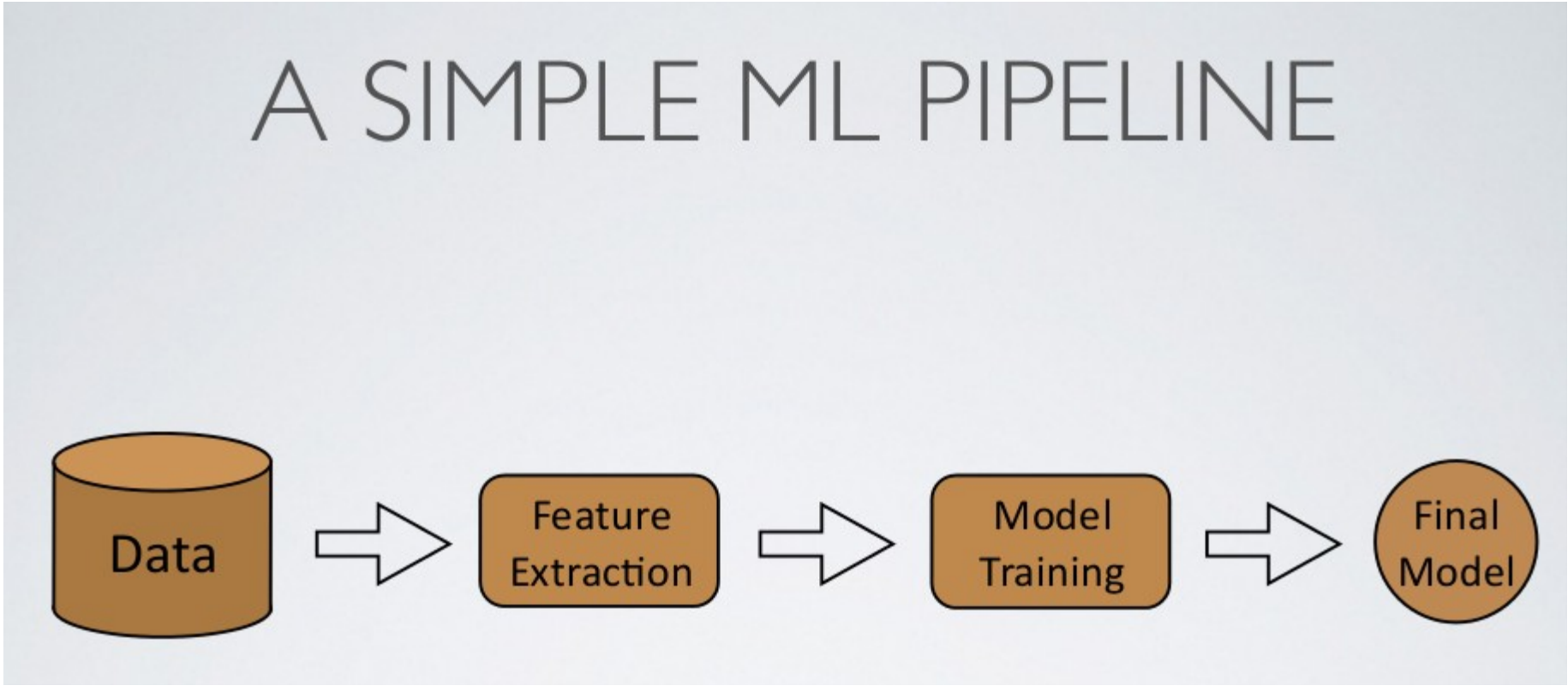
本周内容

- 机器学习
  - 定义
  - 分类
  - 常用算法
- Mllib
  - 什么是 MLlib
  - MLlib 构成
  - MLlib 的运行架构

- 实例演示
  - K-Means 算法介绍和实例
  - 协同过滤 算法介绍和实例

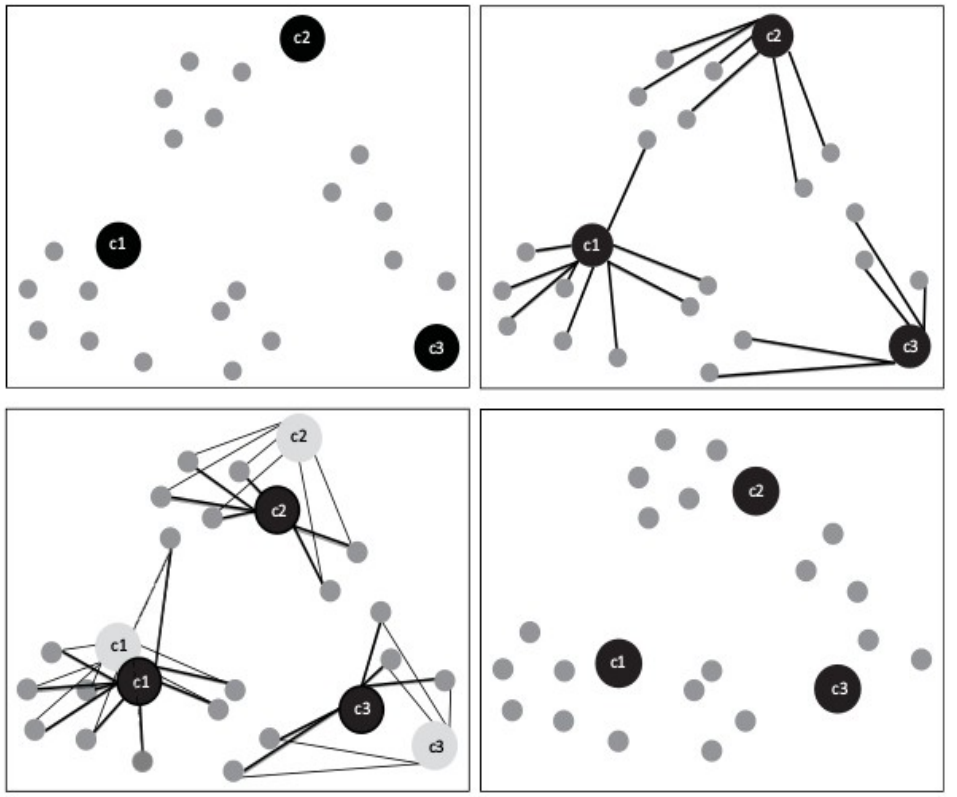


实例演示


A SIMPLE ML PIPELINE
Data
Feature Extraction
Model Training
Final Model

K-Means

- K-Means 是聚类的一个算法，是一个无监督学习，目标是将一部分实体根据某种意义上的相似度和另一部分实体聚在一起。聚类通常被用于探索性的分析。
- 算法：
  - 1 选择 K 个点作为初始中心
  - 2 将每个点指派到最近的中心，形成 K 个簇（聚类）
  - 3 重新计算每个簇的中心
  - 4 重复 2-3 直至中心不发生变化
- 距离
  - 绝对值距离
  - 欧氏距离
  - 闵可夫斯基距离
  - 切比雪夫距离
  - 马氏距离

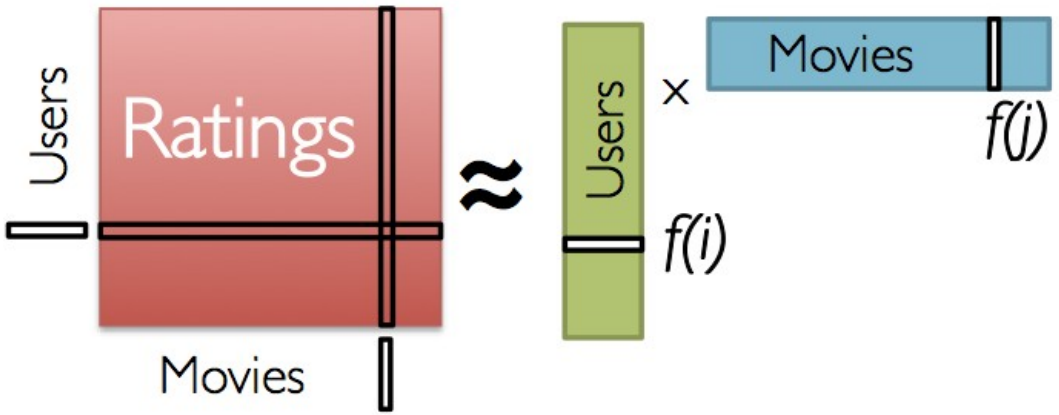


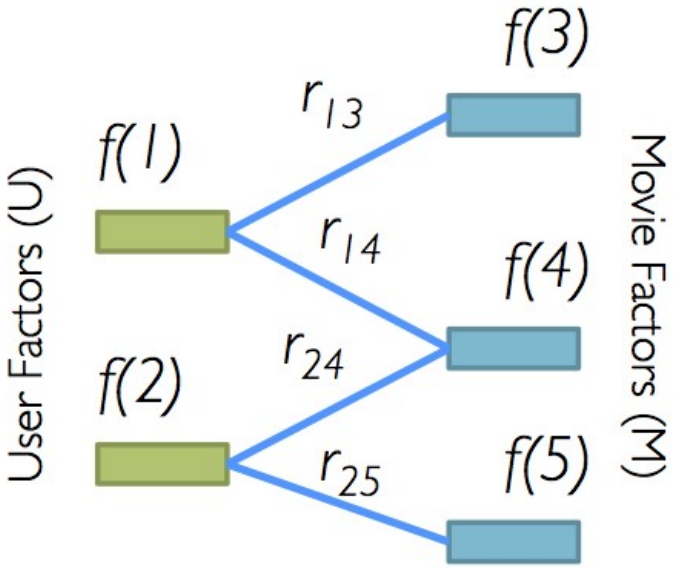
K-Means

- MLlib K-Means 的实现中包含一个 k-means++ 方法的并行化变体 kmeans||。
- MLlib 里面的实现有如下的参数：
  - k 是所需的类簇的个数。
  - maxIterations 是最大的迭代次数。
  - initializationMode 这个参数决定了是用随机初始化还是通过 k-means|| 进行初始化。
  - runs 是跑 k-means 算法的次数（k-mean 算法不能保证能找出最优解，如果在给定的数据集上运行多次，算法将会返回最佳的结果）。
  - initializiationSteps 决定了 k-means|| 算法的步数。
  - epsilon 决定了判断 k-means 是否收敛的距离阀值。

协同过滤

Low-Rank Matrix Factorization:





Iterate:

$$f[i] = \arg\min_{w \in \mathbb{R}^d} \sum_{j \in \mathrm{Nbrs}(i)} \left(r_{ij} - w^T f[j]\right)^2 + \lambda ||w||_2^2$$

协同过滤

- 协同过滤常被应用于推荐系统。这些技术旨在补充用户 - 商品关联矩阵中所缺失的部分。
- MLlib 当前支持基于模型的协同过滤，其中用户和商品通过一小组隐性因子进行表达，并且这些因子也用于预测缺失的元素。 MLLib 使用交替最小二乘法 (ALS) 来学习这些隐性因子。

- 在 MLlib 中的实现有如下的参数：
  - numBlocks 是用于并行化计算的分块个数（设置为 -1 为自动配置）。
  - rank 是模型中隐性因子的个数。
  - iterations 是迭代的次数。
  - lambda 是 ALS 的正则化参数。
  - implicitPrefs 决定了是用显性反馈 ALS 的版本还是用适用隐性反馈数据集的版本。
  - alpha 是一个针对于隐性反馈 ALS 版本的参数，这个参数决定了偏好行为强度的基准

课程回顾

- 机器学习
- MLlib
- 下周 GraphX
  - GraphX 原理
  - GranpX 实例

Thanks

FAQ 时间