

09

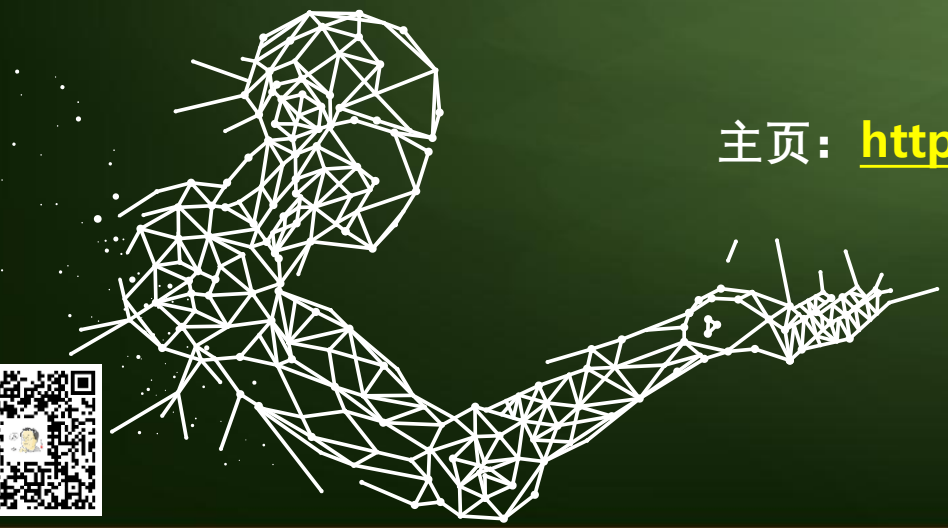


第9章 Spark Streaming

授课教师：[李阳源]

ywlee1212@gmail.com

主页：https://sites.google.com/view/dalguri



SL-01:提纲



9.01 基于大数据的机器学习
9.02 机器学习库MLlib概述
9.03 基本的数据类型
9.04 基本的统计分析工具
9.05 机器学习流水线
9.06 特征提取、转换和选择及局部敏感哈希
9.07 分类算法
9.08 聚类算法
9.09 频繁模式挖掘算法
9.10 协同过滤算法
9.11 模型选择

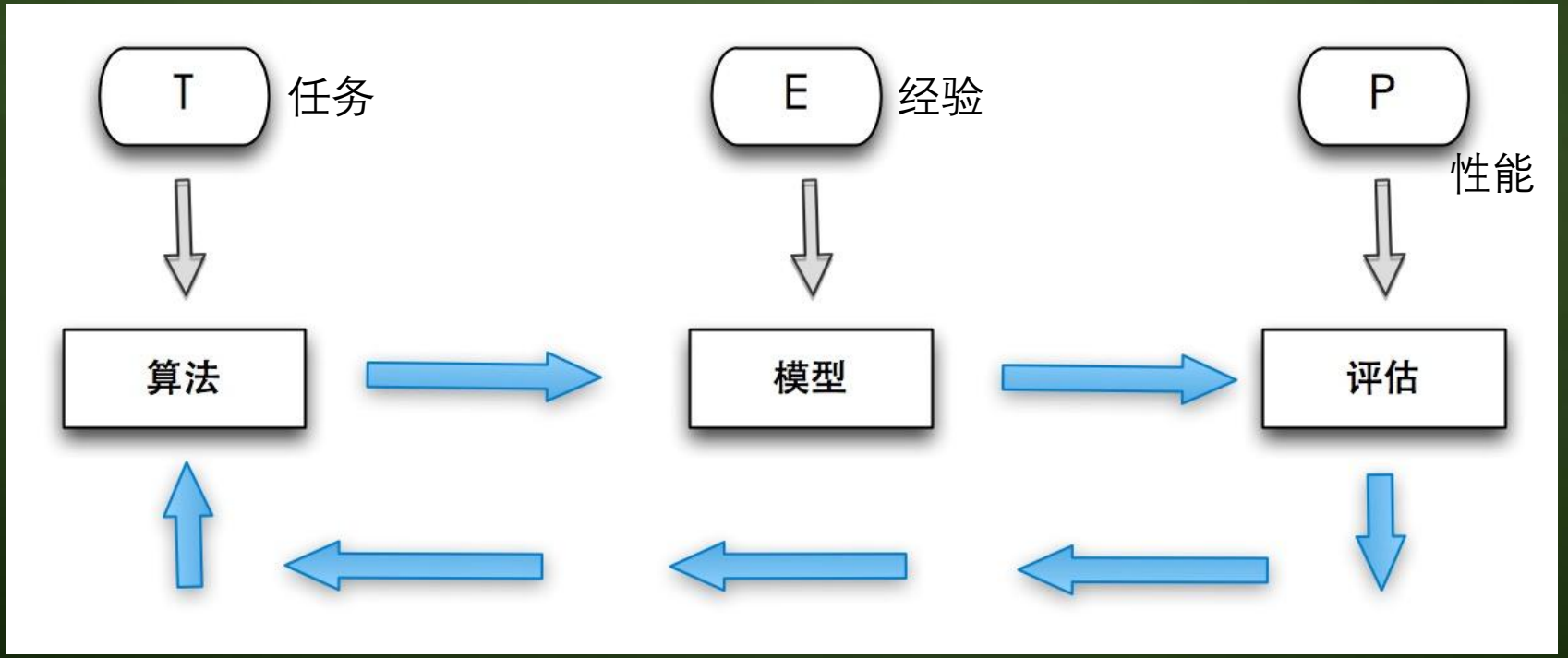




9.1 基于大数据的机器学习



机器学习可以看做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。**机器学习利用数据或以往的经验，以此优化计算机程序的性能标准。**



机器学习强调三个关键词：算法、经验、性能

9.1 基于大数据的机器学习



- 传统的机器学习算法，由于技术和单机存储的限制，只能在少量数据上使用，依赖于数据抽样
- 大数据技术的出现，可以支持在全量数据上进行机器学习
- 机器学习算法涉及大量**迭代计算**
- 基于磁盘的MapReduce不适合进行大量迭代计算
- 基于内存的Spark比较适合进行大量迭代计算

9.1 基于大数据的机器学习



- 相比而言，Spark立足于内存计算，适用于迭代计算，能很好地与机器学习算法相匹配，这也是近年来Spark平台流行的重要原因，业界的很多业务纷纷从Hadoop平台转向Spark平台

- Spark提供了一个基于海量数据的机器学习库，它提供了常用机器学习算法的分布式实现，对普通开发者而言，只需要有Spark编程基础，并且了解机器学习算法的基本原理和方法中相关参数的含义，就可以轻松地通过调用相应的API来实现基于海量数据的机器学习过程

9.2 机器学习库MLlib概述



- MLlib是Spark的机器学习（Machine Learning）库，旨在简化机器学习的工程实践工作
- MLlib由一些通用的学习算法和工具组成，包括**分类、回归、聚类、协同过滤、降维**等，同时还包括底层的优化原语和高层的流水线（Pipeline）API，具体如下：

  - **算法工具**：常用的学习算法，如分类、回归、聚类和协同过滤；
  - **特征化工具**：特征提取、转化、降维和选择工具；
  - **流水线(Pipeline)**：用于构建、评估和调整机器学习工作流的工具；
  - **持久性**：保存和加载算法、模型和管道；
  - **实用工具**：线性代数、统计、数据处理等工具。

9.2 机器学习库MLlib概述(1)



目前MLlib支持的主要机器学习算法

类型	算法
基本统计 (Basic Statistics)	Summary Statistics , Correlations , Stratified Sampling , Hypothesis Testing , Random Data Generation
分类和回归 (Classification and Regression)	Support Vector Machines (SVM) , Logistic Regression , Linear Regression , Naive Bayes , Decision Trees , Random Forest , Gradient-Boosted Trees
协同过滤 (Collaborative Filtering)	Alternating Least Squares (ALS)
聚类 (Clustering)	K-Means , Gaussian Mixture Model , Latent Dirichlet Allocation （ LDA ） , Bisecting K-Means
降维 (Dimensionality Reduction)	Singular Value Decomposition (SVD) , Principal Component Analysis (PCA)
特征提取和转换 (Feature Extraction and Transformation)	Term Frequency-Inverse Document Frequency (TF-IDF) , Word2Vec , StandardScaler , Normalizer

9.2 机器学习库MLlib概述



Spark 机器学习库从1.2 版本以后被分为两个包：A

- **spark.mllib** 包含基于RDD的原始算法API。Spark MLlib 历史比较长，在1.0 以前的版本即已经包含了，提供的算法实现都是基于原始的 RDD
- **spark.ml** 则提供了基于DataFrames 高层次的API，可以用来构建机器学习工作流（PipeLine）。ML Pipeline 弥补了原始 MLlib 库的不足，向用户提供了一个基于 DataFrame 的机器学习工作流式 API 套件

9.3 基本的数据类型



9.3.1 本地向量(Local Vector)
9.3.2 标注点(Labeled Point)
9.3.3 本地矩阵(Local Matrix)
9.3.4 数据源(Data Source)

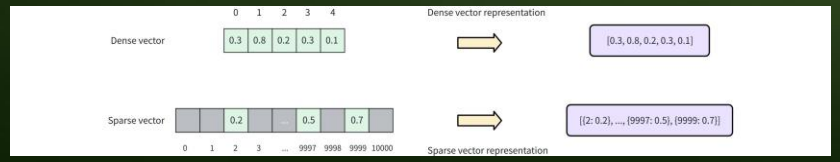
9.3.1 本地向量



- 稠密向量（Dense Vector）
  - 使用**双精度浮点型**数组来表示每一维的元素
- 稀疏向量（Sparse Vector）
  - 稀疏向量则是基于一个**整型索引数组**和一个**双精度浮点型**的值数组。

- 向量(1.0, 0.0, 3.0)
  - 稠密向量表示形式是[1.0,0.0,3.0]
  - 而稀疏向量形式则是(3, [0,2], [1.0, 3.0])

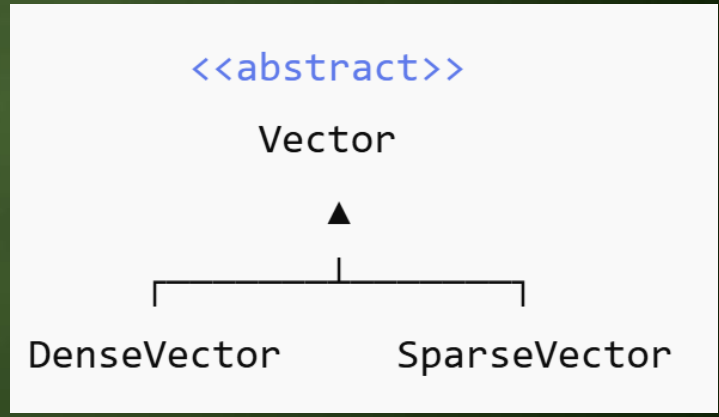
稀疏向量的格式是：(向量长度, [非零元素的索引], [非零元素的值])

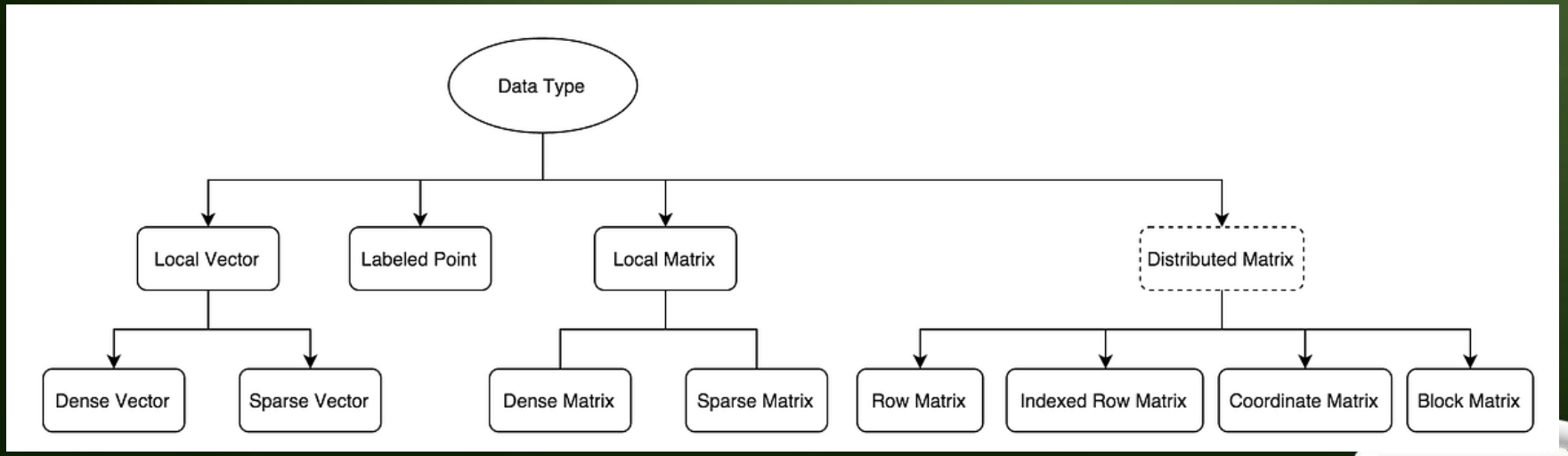


9.3.1 本地向量



所有本地向量都以org.apache.spark.ml.linalg.Vector为基类，DenseVector和SparseVector分别是它的两个继承类，故推荐使用Vectors工具类下定义的工厂方法(factory method)来创建本地向量。





9.3.1 本地向量

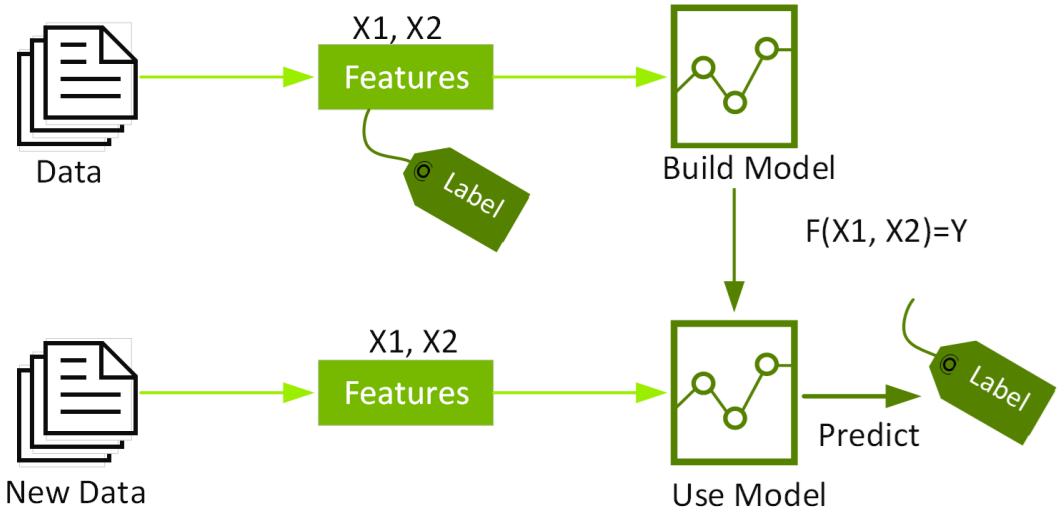


```
>>> from pyspark.ml.linalg import Vectors
# 创建一个稠密向量
>>> dv = Vectors.dense([2.0, 0.0, 8.0])
# 创建一个稀疏向量
# 方法的第2个参数（数组）指定了非0元素的索引，而第3个参数（数组）则给出了非零元素的值
>>> sv1 = Vectors.sparse(3, [0, 2], [2.0, 8.0])
# 另一种创建稀疏向量的方法
# 方法的第2个参数是一个序列，其中每个元素都是一个非零值的元组：(index,elem)
>>> sv2 = Vectors.sparse(3, [(0, 2.0), (2, 8.0)])
>>> print("稠密向量 dv:", dv)
稠密向量 dv: [2.0,0.0,8.0]
>>> print("稀疏向量 sv1:", sv1)
稀疏向量 sv1: (3,[0,2],[2.0,8.0])
>>> print("稀疏向量 sv2:", sv2)
稀疏向量 sv2: (3,[0,2],[2.0,8.0])
```

9.3.2 标注点



- 标注点（Labeled Point）是一种**带有标签**（Label/Response）的**本地向量**，通常用在监督学习算法中，它可以是稠密或者是稀疏的。
- 由于标签是用双精度浮点型来存储的，因此，标注点类型在回归（Regression）和分类（Classification）问题上均可使用。
- 例如，对于二分类问题，则正样本的标签为1，负样本的标签为0；对于多类别的分类问题来说，标签则应是一个以0开始的索引序列：0, 1, 2 …



Labeled Point = (Label, Vector)

每一个标注点表示 **一个训练样本**
由两部分组成：
Label（标签，y）：正确答案
Features（特征，x）：输入特征向量

9.3.2 标注点



标注点的实现类是pyspark.mllib.regression.LabeledPoint，位于pyspark.mllib.regression包下。
标注点的创建方法如下：

```
>>> from pyspark.mllib.regression import LabeledPoint #引入必要的包
>>> from pyspark.mllib.linalg import Vectors
>>> from pyspark.sql.session import SparkSession

# 创建一个标签为1.0（分类中可视为正样本）的稠密向量标注点
>>> pos = LabeledPoint(1.0, Vectors.dense([2.0, 0.0, 8.0]))

# 创建一个标签为0.0（分类中可视为负样本）的稀疏向量标注点
>>> neg = LabeledPoint(0.0, Vectors.sparse(3, [0, 2], [2.0, 8.0]))

>>> print("Positive Example:", pos)
Positive Example: (1.0,[2.0,0.0,8.0])
>>> print("Negative Example:", neg)
Negative Example: (0.0,(3,[0,2],[2.0,8.0]))
```

9.3.2 标注点



在实际的机器学习问题中，稀疏向量数据是非常常见的。MLlib提供了读取LIBSVM格式数据的支持，该格式被广泛用于LIBSVM、LIBLINEAR等机器学习库。
在该格式下，每一个带标签的样本点由以下格式表示：

label index1:value1 index2:value2 index3:value3 …

其中，label是该样本点的标签值，

一系列index:value则代表了该样本向量中所有非零元素的索引和元素值。

需要特别注意的是，index是以1开始并递增的。

9.3.2 标注点



下面读取一个LIBSVM格式文件生成向量：

```
# 从文件加载LIBSVM格式的数据
>>> path = "file:///usr/local/spark/data/mllib/sample_libsvm_data.txt"
# 创建一个 SparkSession（如果尚未创建）
>>> spark = SparkSession.builder.appName("example").getOrCreate()
>>> examples = spark.read.format("libsvm").load(path)
```

9.3.2 标注点



下面继续查看加载进来的标注点的值。

```
>>> examples.head()
Row(label=0.0, features=SparseVector(692, {127: 51.0, 128: 159.0, 129: 253.0, 130: 159.0, 131: 50.0, 15
4: 48.0, 155: 238.0, 156: 252.0, 157: 252.0, 158: 252.0, 159: 237.0, 181: 54.0, 182: 227.0, 183: 253.0, 18
4: 252.0, 185: 239.0, 186: 233.0, 187: 252.0, 188: 57.0, 189: 6.0, 207: 10.0, 208: 60.0, 209: 224.0, 210: 2
52.0, 211: 253.0, 212: 252.0, 213: 202.0, 214: 84.0, 215: 252.0, 216: 253.0, 217: 122.0, 235: 163.0, 236:
252.0, 237: 252.0, 238: 252.0, 239: 253.0, 240: 252.0, 241: 252.0, 242: 96.0, 243: 189.0, 244: 253.0, 245:
167.0, 262: 51.0, 263: 238.0, 264: 253.0, 265: 253.0, 266: 190.0, 267: 114.0, 268: 253.0, 269: 228.0, 270:
47.0, 271: 79.0, 272: 255.0, 273: 168.0, 289: 48.0, 290: 238.0, 291: 252.0, 292: 252.0, 293: 179.0, 294: 1
2.0, 295: 75.0, 296: 121.0, 297: 21.0, 300: 253.0, 301: 243.0, 302: 50.0, 316: 38.0, 317: 165.0, 318: 253.0,
319: 233.0, 320: 208.0, 321: 84.0, 328: 253.0, 329: 252.0, 330: 165.0, 343: 7.0, 344: 178.0, 345: 252.0, 3
46: 240.0, 347: 71.0, 348: 19.0, 349: 28.0, 356: 253.0, 357: 252.0, 358: 195.0, 371: 57.0, 372: 252.0, 373:
252.0, 374: 63.0, 384: 253.0, 385: 252.0, 386: 195.0, 399: 198.0, 400: 253.0, 401: 190.0, 412: 255.0, 413:
253.0, 414: 196.0, 426: 76.0, 427: 246.0, 428: 252.0, 429: 112.0, 440: 253.0, 441: 252.0, 442: 148.0, 454:
85.0, 455: 252.0, 456: 230.0, 457: 25.0, 466: 7.0, 467: 135.0, 468: 253.0, 469: 186.0, 470: 12.0, 482: 85.0,
483: 252.0, 484: 223.0, 493: 7.0, 494: 131.0, 495: 252.0, 496: 225.0, 497: 71.0, 510: 85.0, 511: 252.0, 51
2: 145.0, 520: 48.0, 521: 165.0, 522: 252.0, 523: 173.0, 538: 86.0, 539: 253.0, 540: 225.0, 547: 114.0, 54
8: 238.0, 549: 253.0, 550: 162.0, 566: 85.0, 567: 252.0, 568: 249.0, 569: 146.0, 570: 48.0, 571: 29.0, 572:
85.0, 573: 178.0, 574: 225.0, 575: 253.0, 576: 223.0, 577: 167.0, 578: 56.0, 594: 85.0, 595: 252.0, 596: 2
52.0, 597: 252.0, 598: 229.0, 599: 215.0, 600: 252.0, 601: 252.0, 602: 252.0, 603: 196.0, 604: 130.0, 622:
28.0, 623: 199.0, 624: 252.0, 625: 252.0, 626: 253.0, 627: 252.0, 628: 252.0, 629: 233.0, 630: 145.0, 651:
25.0, 652: 128.0, 653: 252.0, 654: 253.0, 655: 252.0, 656: 141.0, 657: 37.0}))
```

9.3.3 本地矩阵(local matrix)



本地矩阵具有**整型**的**行、列索引值**和**双精度浮点型**的**元素值**，它存储在单机上。MLlib支持稠密矩阵DenseMatrix和稀疏矩阵SparseMatrix两种本地矩阵。

- 稠密矩阵将所有元素的值存储在一个列优先（Column-major）的双精度型数组中。
- 稀疏矩阵则将非零元素以列优先的CSC（Compressed Sparse Column）模式进行存储。

本地矩阵的基类是pyspark.ml.linalg.Matrix，DenseMatrix和SparseMatrix均是它的继承类。

和本地向量类似，pyspark.ml包也为本地矩阵提供了相应的工具类Matrices，调用工厂方法即可创建实例。

9.3.3 本地矩阵



下面创建一个稠密矩阵：

```
>>> from pyspark.ml.linalg import Matrices

# 创建一个3行2列的稠密矩阵[ [1.0, 2.0], [3.0, 4.0], [5.0, 6.0] ]
# 注意，这里的数组参数是列优先的，即按照列的方式从数组中提取元素
>>> dm = Matrices.dense(3, 2, [1.0, 3.0, 5.0, 2.0, 4.0, 6.0])

>>> print(dm)
DenseMatrix([[1., 2.],
             [3., 4.],
             [5., 6.]])
```

9.3.3 本地矩阵



下面继续创建一个稀疏矩阵。

```
# 创建一个3行2列的稀疏矩阵[ [9.0,0.0], [0.0,8.0], [0.0,6.0]]
# 第1个数组参数表示列指针，即每一列元素的开始索引
# 第2个数组参数表示行索引，即对应的元素属于哪一行
# 第3个数组参数即按列优先排列的所有非零元素，通过列指针和行索引即可判断每个元素所在的位置
>>> sm = Matrices.sparse(3, 2, [0, 1, 3], [0, 2, 1], [9.0, 6.0, 8.0])

>>> print(sm)
3 X 2 CSCMatrix
(0,0) 9.0
(2,1) 6.0
(1,1) 8.0
```

9.3.4 数据源



数据源是指正在使用的数据的来源地。

比如Parquet, CSV, JSON和JDBC等。

在这一节中我们将介绍MLlib中一些特定的数据源，

包含**图像数据源**与**LIBSVM数据源**。

9.3.4 数据源



（1）图像数据源（Image Data Source）。

用于从一个目录中加载图像文件，它可以通过Java库中的ImageIO将压缩图像（jpeg、png等）加载为原始图像表示。

加载的DataFrame有一个StructType列"image"，其包含以图像模式存储的图像数据。

"image"列的有如下属性：

（a）origin: StringType (表示图像的文件路径)
（b）height: IntegerType (图像的高度)
（c）width: IntegerType (图像的宽度)
（d）nChannels: IntegerType (图像的通道)
（e）mode: IntegerType (同OpenCV兼容的图像模式)
（f）data: BinaryType (按OpenCV兼容的顺序排列图像字节：在大多数情况下是按行排列的BGR)-→cf. RGB

9.3.4 数据源



```
# 从文件加载图像数据
# 这里假设您的图像数据位于 "file:///usr/local/spark/data/mllib/images/origin/kittens
" 路径下
>>> from pyspark.sql import SparkSession
# 创建一个 SparkSession（如果尚未创建）
>>> spark = SparkSession.builder.appName("example").getOrCreate()
>>> df = spark.read.format("image") \
    .option("dropInvalid", True) \
    .load("file:///usr/local/spark/data/mllib/images/origin/kittens")
```

9.3.4 数据源



第2步：输出image列的origin 、width和height属性值。

```
# 选择DataFrame中的图像属性
>>> selected_df = df.select("image.origin", "image.width", "image.height")

# 显示图像属性，不截断显示
>>> selected_df.show(truncate=False)
+------------------------------------------------------------------------+-----+------
|origin                                                                  |width|height|
+------------------------------------------------------------------------+-----+------+
|file:///spark/data/mllib/images/origin/kittens/54893.jpg                |300  |311   |
|file:///spark/data/mllib/images/origin/kittens/DP802813.jpg             |199  |313   |
|file:///spark/data/mllib/images/origin/kittens/29.5.a_b_EGDP022204.jpg |300  |200
|
|file:///spark/data/mllib/images/origin/kittens/DP153539.jpg             |300  |296   |
+------------------------------------------------------------------------+-----+------+
```

9.3.4 数据源



（2）MLlib库提供LibSVMDatasource类来将libsvm格式的数据加载为dataframe。

Libsvm是由国立台湾大学以C++语言开发的一个开源机器学习库，主要提供有关支持向量机的算法。该库的数据格式是每一行代表一个稀疏的特征向量，格式为

<label> <index1>:<value1> <index2>:<value2>，…。

- 其中<label>是数据样本的类别，<index>是索引值，<value>是对应位置的特征的属性值。

注意到特征向量是稀疏的，所以其他索引值对应的属性值默认为0。转换后的DataFrame有两列：

- 标签，以Double类型存储；
- 特征，以Vector类型存储。

9.3.4 数据源



第1步：读取Spark自带的libsvm格式的输入数据

```
>>> df = spark.read.format("libsvm") \
    .option("numFeatures", "780") \
    .load("file:///usr/local/spark/data/mllib/sample_libsvm_data.txt")
```

LibSVMDataSource的option参数含义如下表

参数	含义
numFeatures	特征的数目。如果该参数未指定或参数值非正数，方法能够自动确定特征的数量，但会带来额外的开销。
vectorType	特征向量的类型，取值为”sparse”或“dense”，默认值为“sparse”，即稀疏向量。

9.3.4 数据源



第2步：输出结果。

```
>>> df.show(10)
+-----+--------------------+
|label|            features|
+-----+--------------------+
|  0.0|(780,[127,128,129...|
|  1.0|(780,[158,159,160...|
|  1.0|(780,[124,125,126...|
|  1.0|(780,[152,153,154...|
|  1.0|(780,[151,152,153...|
|  0.0|(780,[129,130,131...|
|  1.0|(780,[158,159,160...|
|  1.0|(780,[99,100,101,...|
|  0.0|(780,[154,155,156...|
|  0.0|(780,[127,128,129...|
+-----+--------------------+
only showing top 10 rows
```