Spark Machine Learning

大数据机器学习

- 机器学习已成为云计算应用的核心
- 机器学习最近取得的重大突破来自几个方面
  - 大数据
  - 算法进步
  - 更快计算平台（GPU）

四个基本概念：DataFrame

- 以有效的方式保存矢量和其他结构化数据类型
- 与 Pandas DataFrames 类似，共享一些操作
- 它们是分布式对象，是执行图的一部分
- 可将它们转换为 Pandas DataFrame，就可以用 Python 访问它们

四个基本概念：Transformers

- 将一个 DataFrame 转换为另一个 DataFrame 的运算符
- 它们是执行图上的节点，因此在执行整个图之前不会对其进行评估
- 如
  - 将文本文档转换为向量
  - 将 DataFrame 的列从一种形式转换为另一种形式
  - 将 DataFrame 拆分为子集

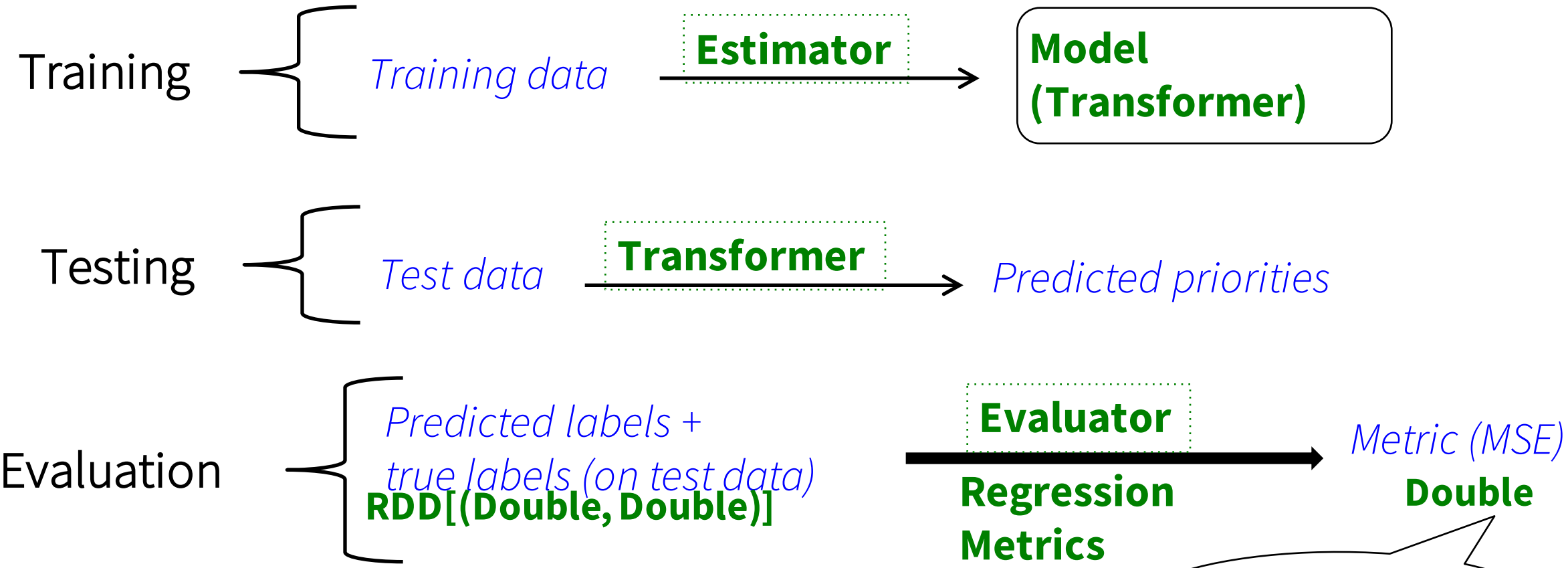
四个基本概念：Estimators

- 封装 ML 和其他算法
- fit()方法将 DataFrame 和参数传递给学习算法以创建模型

Spark MLib

- 步骤 1
  - 输入数据分为两个子集：训练数据与测试数据
  - 在进入计算或学习引擎之前，两者都存储在数据存储器中
- 步骤 2
  - 数据预处理，例如过滤，挖掘，数据聚合，特征提取，模式识别以及某些转换操作
- 步骤 3
  - 使用云计算和存储资源的学习引擎
  - 包括数据清理，模型训练以及在监督下向模型开发的转变。
- 步骤 4
  - 学习模型的构建，适应环境满足预测或分类等学习目标的环境问题
- 步骤 5
  - 通过制定决策或预测进行的训练和测试阶段

+ Evaluation



Model selection

Choose among different models or model hyperparameters

Overview of ML Algorithms

- Prediction — feature vector → label
  - Regression (real value label)
  - Classification (categorical label)
- Feature transformation
- Recommendation
- Clustering
- Other
  - Statistics
  - Linear algebra
  - Optimization

E.g.: given log file, predict priority

`LinearRegression, DecisionTree`
`LogisticRegression, NaiveBayes`

Iterative optimization

Partition data by rows (instances):

- Easy to handle billions of rows
- Hard to scale # features
  - 10^7 for *Regression, SVM, NB
  - 10^3 for DecisionTree

Overview of ML Algorithms

- Prediction
  - Regression
  - Classification
- Feature transformation
- Recommendation
- Clustering
- Other
  - Statistics
  - Linear algebra
  - Optimization

E.g.: convert text to feature vectors

```
Tokenizer, HashingTF, IDF, Word2Vec
```

```
Normalizer, StandardScaler
```

Arguably the *most important* part of machine learning

Per-row transformation

Overview of ML Algorithms

- Prediction
  - Regression
  - Classification
- Feature transformation
- Recommendation
  user → recommended products
- Clustering
- Other
  - Statistics
  - Linear algebra
  - Optimization

E.g.: Recommend movies to users

ALS

Phrased as matrix factorization
Given (users x products) matrix with many missing entries,
Find low-rank factorization.
Fill in missing entries.

Partition data by both users and products.

Scales to millions of users and products.

Overview of ML Algorithms

- Prediction
  - Regression
  - Classification
- Feature transformation
- Recommendation
- Clustering
  feature vectors → clusters (no labels)
- Other
  - Statistics
  - Linear algebra
  - Optimization

E.g.: Given news articles, automatically group articles by topics

KMeans, GaussianMixutre, LDA

Iterative optimization.
Local optima.

Partition data by rows.
Easy to handle billions of rows

Overview of ML Algorithms

- Prediction
  - Regression
  - Classification
- Feature transformation
- Recommendation
- Clustering
- Other (MLLib)
  - Statistics
  - Linear algebra
  - Optimization

E.g.: Is model A significantly better than model B?

```
ChiSqSelector, Statistics,
  MultivariateOnlineSummarizer
```

E.g.: Matrix decomposition

```
DenseMatrix, SparseMatrix,
  EigenValueDecomposition, ...
```

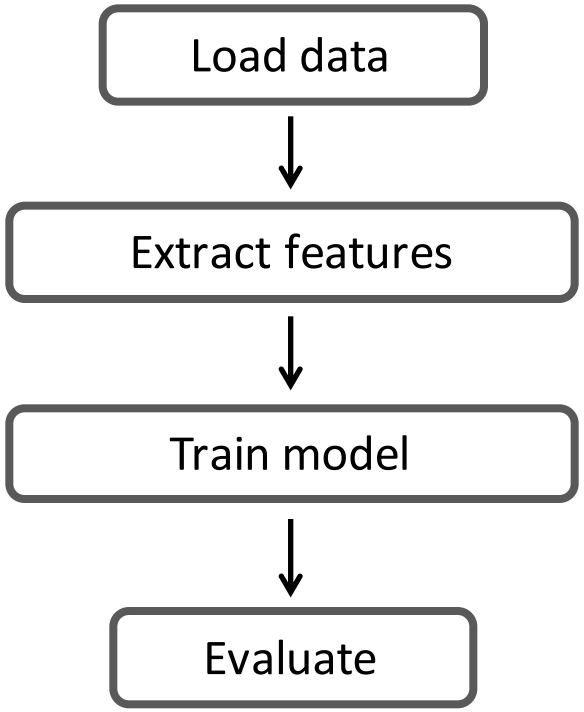
E.g.: Given function f(x), find x to minimize f(x)

```
GradientDescent, LBFGS
```

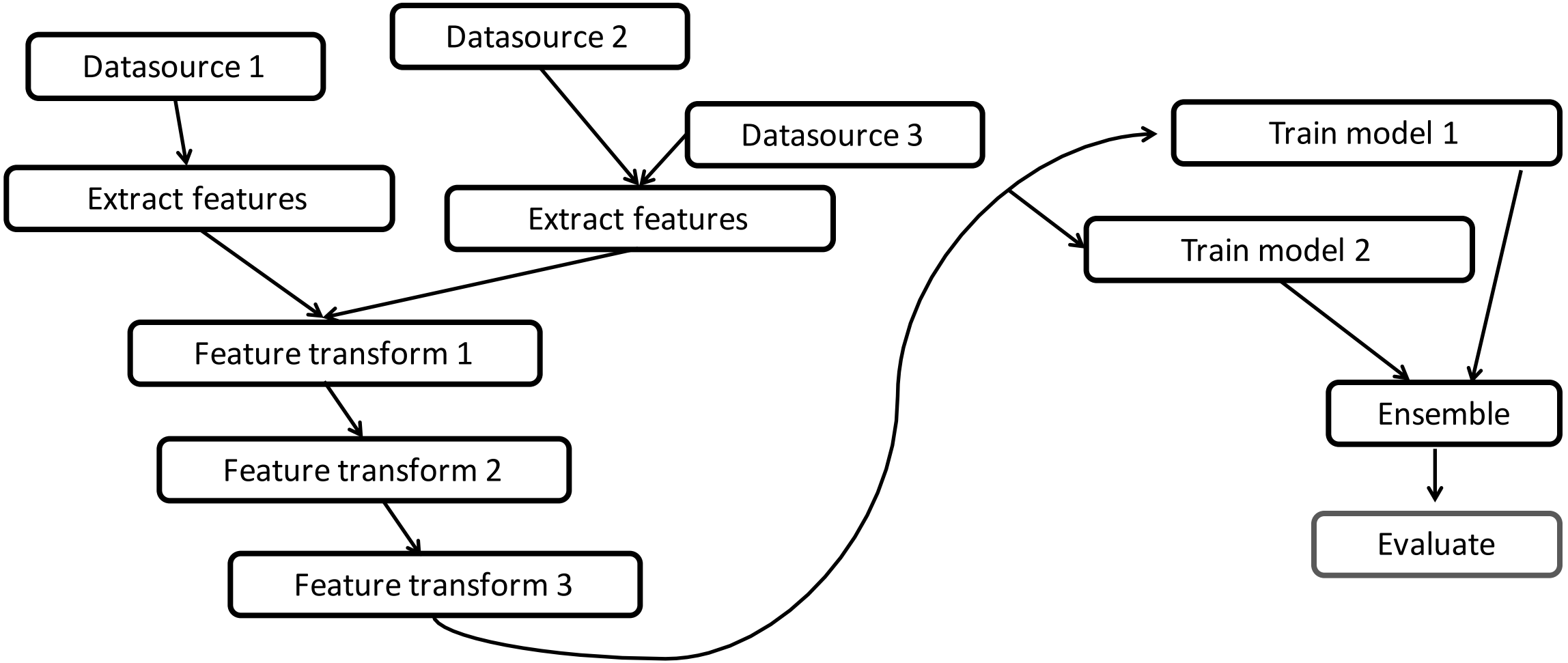
Pipeline

- 通常是线性的，但也可以是有向无环图
- 链接 Transformers 和 Estimators，指定一个 ML 工作流
- 用 fit() 训练完估算器后，Pipeline 就是一个模型，具有 transform() 方法，可对新案例进行预测

ML Pipelines



ML Pipelines



交叉验证

- 机器学习中的一项重要任务是模型选择，或使用数据为给定任务找到最佳模型或参数。这也称为 Tunning
  - Pipeline 可以轻松地一次调整整个 Pipeline，不必分别调整其中的每个元素，简化了模型选择
  - MLlib 支持使用 CrossValidator 类进行模型选择，该类具有一个估计器，一组 ParamMap 和一个评估器

交叉验证

- CrossValidator 首先将数据集划分为一组 folds，它们将被用作单独的训练和测试数据集
  - 如 k = 3 folds，就会生成 3 对（训练，测试）数据集对，每对使用三分之二的数据用于训练，另外三分之一的数据用于测试。
- CrossValidator 遍历 ParamMaps 集。对于每个 ParamMap，它训练给定的估算器并对其进行评估，选择产生最佳评估指标的 ParamMap 作为最佳模型
- 最后，CrossValidator 使用最佳的 ParamMap 和整个数据集来训练最终的估算器

示例

- 创建 DataFrame，包含由矢量表示的标签和多个特征

```
df = sqlContext.createDataFrame
    (data, [“label”, “features”])
```

- 设置算法参数。在这里，我们将 LR 的迭代次数设为 10

```
lr = LogisticRegression(maxIter = 10)
```

示例

- 从数据中训练模型

```
model = lr.fit(df)
```

- 将数据集送入训练好的模型，预测每个点的标签，显示结果

```
model.transform(df).show()
```