BDA6-1

1

Hello everyone, I am Haiying Che, from Institute of Data Science and knowledge Engineering

School of Computer Science, in Beijing Institute of Technology, from this session on, we will discuss something about real application,

Including some platform and big data application and in this session, we will talk about the concepts and mechanism of Spark MLlib.

2

From this chapter ,we will introduce some useful platform， Spark MLlib and TensorFlow.

We will show some case experiments applying the corresponding platform;

and we will also explain 2 kinds of typical big data analysis scenarios, Recommendation System and Social Networking.

3

Apache Spark’s Machine Learning Library (MLlib) is designed for simplicity, scalability, and easy integration with other tools.

With the scalability, language compatibility, and speed of Spark, data scientists can focus on their data problems and models instead of solving the complexities surrounding distributed data (such as infrastructure, configurations, and so on).

Spark MLlib seamlessly integrates with other Spark components such as Spark SQL, Spark Streaming, and DataFrames API.

Machine learning can be applied to a wide variety of data types, such as vectors, text, images, and structured data.

This API adopts the DataFrame from Spark SQL to support a variety of data types.

The library is usable in Java, Scala, and Python as part of Spark applications, so that you can include it in complete workflows.

MLlib allows for preprocessing, munging咀嚼, training of models, and making predictions at scale on data.

You can even use models trained in MLlib to make predictions in Structured Streaming.

4

Big data analytics using spark MLlib include descriptive analysis and predictive analytics based on descriptive analytical data and predictive analytical data.

In the scope of descriptive analysis, it supports statistical descriptions analysis and clustering.

In the scope of predictive analysis, it supports Feature modeling, which includes feature extractor like TF-IDF, Feature transformer like Vector Slicer etc. and feature selector like Chi-square selector. It also supports prediction algorithms like binary classification, multiclass classification and regression.

5

**MLlib** is Apache Spark's scalable machine learning library.

1Ease of use

Usable in Java, Scala, Python, and R.

MLlib fits into [Spark](https://spark.apache.org/)'s APIs and interoperates with [NumPy](http://www.numpy.org/) in Python (as of Spark 0.9) and R libraries (as of Spark 1.5).

You can use any Hadoop data source (e.g. HDFS, HBase, or local files), making it easy to plug into Hadoop workflows.

2 Performance

High-quality algorithms, 100x faster than MapReduce.

Spark excels at iterative computation, enabling MLlib to run fast.

At the same time, we care about algorithmic performance: MLlib contains high-quality algorithms that leverage iteration, and can yield better results than the one yield on MapReduce.

3 Runs everywhere

Spark runs on Hadoop, Apache Mesos, Kubernetes, standalone, or in the cloud, against diverse data sources.

You can run Spark using its [standalone cluster mode](https://spark.apache.org/docs/latest/spark-standalone.html), on [EC2](https://github.com/amplab/spark-ec2), on [Hadoop YARN](https://hadoop.apache.org/docs/current/hadoop-yarn/hadoop-yarn-site/YARN.html), on [Mesos](https://mesos.apache.org/), or on [Kubernetes](https://kubernetes.io/). Access data in [HDFS](https://hadoop.apache.org/docs/stable/hadoop-project-dist/hadoop-hdfs/HdfsUserGuide.html), [Apache Cassandra](https://cassandra.apache.org/), [Apache HBase](https://hbase.apache.org/), [Apache Hive](https://hive.apache.org/), and hundreds of other data sources.

To support Python with Spark, the Apache Spark community released a tool, PySpark. Using PySpark, one can work with RDDs in Python programming language.

6

Built on top of Spark, MLlib is a scalable machine learning library consisting of 4 main components, Algorithms, Featurization, Pipeline and Utilities.

common learning algorithms including classification, regression, clustering, collaborative filtering, dimensionality reduction, and underlying optimization primitives.

and utilities, which include Linear algebra, statistics etc.

And the Featurization include extracting the features and transforming the features.

ML Pipelines provide a uniform set of high-level APIs built on top of DataFrames that help users create and tune practical machine learning pipelines.

the mechanism to finish the ML tasks is to construct the pipeline to finish all the needed steps. Like constructing model, train the model, evaluating the model, tuning the parameters and persistence the model.

7

ML algorithms include:

* Classification: logistic regression, naive Bayes, ...
* Regression: generalized linear regression, survival regression, ...
* Decision trees, random forests, and gradient-boosted trees
* Recommendation: alternating least squares (ALS)
* Clustering: K-means, Gaussian mixtures (GMMs), ...
* Topic modeling: latent Dirichlet allocation (LDA)
* Frequent item sets, association rules, and sequential pattern mining

8

ML workflow utilities include:

* **Feature transformations**: standardization, normalization, hashing, ...
* **ML Pipeline construction**
* **Model evaluation and hyper-parameter tuning**
* **ML persistence**: saving and loading models and Pipelines

Other utilities include:

* **Distributed linear algebra**: SVD, PCA, ...
* **Statistics**: summary statistics, hypothesis testing, ...

9

On the left, it is the normal machine learning pipeline, 1Load/clean data, 2 feature extraction, 3 model training and 4model evaluation and parameter tuning, then repeat the workflow process.

And on the right, it is the Mllib Pipeline Concepts, from 1 load/clean data, 2 transformers, which is corresponding to feature engineering, 3 Estimator, which is corresponding to model training, and 4 Evaluator, which is responsible for model evaluation. Let me explain them one by one

10

A Transformer is an abstraction that includes feature transformers and learned models.

which Transforming data into consumable format,

Take input column, transform it to an output column.

Technically, a Transformer implements a method transform (), which converts one DataFrame into another, generally by appending one or more columns.

For example:

A feature transformer might take a DataFrame, read a column (e.g., text), map it into a new column (e.g., feature vectors), and output a new DataFrame with the mapped column appended.

A learning model might take a DataFrame, read the column containing feature vectors, predict the label for each feature vector, and output a new DataFrame with predicted labels appended as a column.

Examples such as : 1 Normalize the data, 2 Tokenization (which means dividing the sentences into words) and 3 Converting categorical values into numbers.

Like showed in the diagram, transform the data frame 1 into data frame 2.

11

An Estimator abstracts the concept of a learning algorithm or any algorithm that fits or trains on data.

Technically, an Estimator implements a method fit(), which accepts a DataFrame and produces a Model, which is a Transformer.

For example, a learning algorithm such as LogisticRegression is an Estimator, and calling fit() trains a LogisticRegressionModel, which is a Model and hence a Transformer.

12

Evaluator is designed to Evaluate the model performance based certain metrics, like ROC, RMSE.

And Evaluator can Help with automating the model tuning process through Comparing model performance

And Selecting the best model for generating predictions.

The Example here is BinaryClassificationEvaluator with CrossValidator.

Input is the data and output is the best model selected from all the options.

13

In machine learning, it is common to run a sequence of algorithms to process and learn from data.

E.g., a simple text document processing workflow might include several stages:

* Split each document’s text into words.
* Convert each document’s words into a numerical feature vector.
* Learn a prediction model using the feature vectors and labels.

MLlib represents such a workflow as a Pipeline, which consists of a sequence of PipelineStages (Transformers and Estimators) to be run in a specific order.

The pipeline Leverages the uniform API of Transformer & Estimator. It Can be persisted.

These stages are run in order, and the input DataFrame is transformed as it passes through each stage.

**For Transformer stages, the transform () method** is called on the DataFrame.

**For Estimator stages, the fit () method** is called to produce a Transformer (which becomes part of the PipelineModel, or fitted Pipeline), and that **Transformer’s transform () method** is called on the DataFrame.

We illustrate this for the simple text document workflow.

The figure in the middle, the top row represents a Pipeline with three stages.

The first two (Tokenizer and HashingTF) are Transformers (blue), and the third (LogisticRegression) is an Estimator (red).

The bottom row represents data flowing through the pipeline, where cylinders indicate DataFrames.

The Pipeline.fit() method is called on the original DataFrame, which has raw text documents and labels.

The Tokenizer.transform() method splits the raw text documents into words, adding a new column with words to the DataFrame.

The HashingTF.transform() method converts the words column into feature vectors, adding a new column with those vectors to the DataFrame.

Now, since LogisticRegression is an Estimator, the Pipeline first calls LogisticRegression.fit() to produce a LogisticRegressionModel.

If the Pipeline had more Estimators, it would call the LogisticRegressionModel’s transform() method on the DataFrame before passing the DataFrame to the next stage.

A Pipeline is an Estimator.

Thus, after a Pipeline’s fit() method runs, it produces a PipelineModel, which is a Transformer (which means the model is transformer).

This PipelineModel is used at *test time*; the figure at the bottom illustrates this usage.

the PipelineModel has the same number of stages as the original Pipeline, but all Estimators in the original Pipeline have become Transformers.

When the PipelineModel’s transform () method is called on a test dataset, the data are passed through the fitted pipeline in order.

Each stage’s transform () method updates the dataset and passes it to the next stage.

Pipelines and PipelineModels help to ensure that training and test data go through identical feature processing steps.

14

MLlib Estimators and Transformers use a uniform API for specifying parameters.

A Param is a named parameter with self-contained documentation.

A ParamMap is a set of (parameter, value) pairs.

There are two main ways to pass parameters to an algorithm:

Set parameters for an instance. E.g., if lr is an instance of LogisticRegression, one could call lr.setMaxIter(10) (maximum iteration)to make lr.fit() use at most 10 iterations. This API resembles the API used in spark.mllib package.

Pass a ParamMap to fit() or transform(). Any parameters in the ParamMap will override parameters previously specified via setter methods.

Parameters belong to specific instances of Estimators and Transformers. For example, if we have two LogisticRegression instances lr1 and lr2, then we can build a ParamMap with both maxIter parameters specified: ParamMap(lr1.maxIter -> 10, lr2.maxIter -> 20). This is useful if there are two algorithms with the maxIter parameter in a Pipeline.

15

After you build a pipeline in MLlib, it can Automate the model tuning process.

A very important task in ML is model selection, or to say, the use of data to find the best model or parameters for a given task.

This is called parameter tuning. Tuning can be performed on a single Estimators (such as LogisticRegression) or

on the entire Pipeline (which can include multiple algorithms, characterization and other steps).

We need to Build a param grid for grid search-based model selection,

To build parameter grid, we can use ParamGridBuilder tool Class.

ParamGridBuilder() allows to specify different values for a single parameters,

and then compare the entire set of parameters to choose the best options, which define the best model.

CrossValidator divides the data set into several folds, which can be used for independent training and test sets.

For example: when k=5 folds, CrossValidator will generate 5 (training, test) pairs, each of which uses 4/5 data as the training set and 1/5 as the test set.

To evaluate a ParamMap, use Estimator to fit 5 models on 5 different data pairs, and CrossValidator will calculate the average of 5 evaluation metrics.

After selecting the best ParamMap, CrossValidator will finally use the corresponding Estimator and the best ParamMap to refit the entire data set.

16

**ML persistence means Saving and Loading Pipelines**

Often it is worth it to save a model or a pipeline to disk for later use.

**The data scientist creates a model or a pipeline and the data engineer can deploy model at scale and monitor its application.**

In Spark 1.6, a model import/export functionality was added to the Pipeline API.

As of Spark 2.3, the DataFrame-based API in spark.ml and pyspark.ml has complete coverage.

ML persistence works across Scala, Java and Python.

However, R currently uses a modified format, so models saved in R can only be loaded back in R; this should be fixed in the future and is tracked in [SPARK-15572](https://issues.apache.org/jira/browse/SPARK-15572).

**Backwards compatibility for ML persistence**

In general, MLlib maintains backwards compatibility for ML persistence.

I.e., if you save an ML model or Pipeline in one version of Spark, then you should be able to load it back and use it in a future version of Spark.

17

Consider that Spark’s ML Lib is suitable when you’re doing relatively simple ML on a large data set.

ML Lib is not computationally efficient for small data sets, and you’re better off using scikit-learn for small and medium sized data sets (megabytes, up to a few gigabytes).

We have the hands on carried out on sklearn, which include the algorithm and high-level tools.

Scikit-learn (Sklearn) is the most useful and robust library for machine learning in Python.

It provides a selection of efficient tools for machine learning and statistical modeling including classification, regression, clustering and dimensionality reduction via a consistence interface in Python.

18

In this session, we learned concepts and mechanism of spark MLlib.

thank you for your attention, if you have any question, feel free to contact me.