Big data: architectures and data analytics

Regression algorithms

Regression algorithms

- Spark MLlib provides also a set of regression algorithms
  - E.g., Linear regression
- A regression algorithm is used to predict the value of a continuous attribute (the target attribute) by applying a model on the predictive attributes
- The model is trained on a set of training data
  - i.e., a set of data for which the value of the target attribute is know

Regression algorithms

- The regression algorithms available in Spark work only on numerical data
  - They work similarly to classification algorithms, but they **predict continuous numerical values** (the target attribute is a continuous numerical attribute)
- The input data must be transformed in a DataFrame having the following attributes:
  - label: double
    - The continuous numerical value to be predicted
  - features: Vector of doubles
    - Predictive features

Linear regression and structured data

Linear regression and structured data

- Linear regression is a popular, effective and efficient regression algorithm
- The following slides show how to instantiate a linear regression algorithm in Spark an apply it on unlabeled data
- The input dataset is a structured dataset with a fixed number of attributes
  - One attribute is the target attribute (the label)
    - We suppose the first column contains the target attribute
  - The others are predictive attributes that are used to predict the value of the target attribute

Linear regression and structured data

- Consider the following example file

  2.0,0.0,1.1,0.1

  5.0,2.0,1.0,-1.0

  5.0,2.0,1.3,1.0

  2.0,0.0,1.2,-0.5
- It contains four records
- Each record has three predictive attributes and the target attribute
  - The first attribute (column) is the target attribute
  - The other attributes (columns) are predictive attributes

Linear regression and structured data: Example

```
package it.polito.bigdata.spark.sparkmllib;

import org.apache.spark.api.java.*;
import org.apache.spark.sql.Dataset;
import org.apache.spark.sql.Row;
import org.apache.spark.sql.SparkSession;
import org.apache.spark.ml.Pipeline;
import org.apache.spark.ml.PipelineModel;
import org.apache.spark.ml.PipelineStage;
import org.apache.spark.ml.linalg.Vector;
import org.apache.spark.ml.linalg.Vectors;
import org.apache.spark.ml.regression.LinearRegression;
import org.apache.spark.ml.feature.LabeledPoint;
```

Linear regression and structured data: Example

```
public class SparkDriver {
    public static void main(String[] args) {
        String inputFileTraining;       String inputFileTest;   String outputPath;
        inputFileTraining=args[0];
        inputFileTest=args[1];
        outputPath=args[2];

        // Create a Spark Session object and set the name of the application
        // We use some Spark SQL transformation in this program
        SparkSession ss = SparkSession.builder().
                                appName("MLlib - logistic regression").getOrCreate();

        // Create a Java Spark Context from the Spark Session
        // When a Spark Session has already been defined this method
        // is used to create the Java Spark Context
        JavaSparkContext sc = new JavaSparkContext(ss.sparkContext());
```

Linear regression and structured data: Example

```
// **************************
//Training step
// **************************

// Read training data from a text file
// Each line has the format: target attribute,
// list of three numerical attribute values
// attribute values.
// E.g.,                    1.0,5.8,0.51.7
JavaRDD<String> trainingData=sc.textFile(inputFileTraining);
```

Linear regression and structured data: Example

```
// Map each input record/data point of the input file to a LabeledPoint
JavaRDD<LabeledPoint> trainingRDD=trainingData.map(record ->
        {
                String[] fields = record.split(",");
                // Fields of 0 contains the id of the target attribute
                double targetLabel = Double.parseDouble(fields[0]);

                //The other three cells of fields contain the (numerical)
                // values of the three predictive attributes
                // Create an array of doubles containing those values
                double[] attributesValues = new double[3];

                attributesValues[0] = Double.parseDouble(fields[1]);
                attributesValues[1] = Double.parseDouble(fields[2]);
                attributesValues[2] = Double.parseDouble(fields[3]);
```

Linear regression and structured data: Example

```
                // Create a dense vector based on the content of
                // attributesValues
                Vector attrValues= Vectors.dense(attributesValues);

                // Return a LabeledPoint based on the content of
                // the current line
                return new LabeledPoint(targetLabel , attrValues);
});
```

Linear regression and structured data: Example

```
// Prepare training data.
// We use LabeledPoint, which is a JavaBean.
// We use Spark SQL to convert RDDs of JavaBeans
// into Dataset<Row>. The columns of the Dataset are label
// and features
Dataset<Row> training =
        ss.createDataFrame(trainingRDD, LabeledPoint.class).cache();
```

Linear regression and structured data: Example

```
// Create a LinearRegression object.
// LinearRegression is an Estimator that is used to
// create a regression model based on linear regression.
LinearRegression lr = new LinearRegression();

// We can set the values of the parameters of the
// Linear Regression algorithm using the setter methods.
// There is one set method for each parameter
// For example, we are setting the number of maximum iterations to 10
// and the regularization parameter. to 0.0.1
lr.setMaxIter(10);
lr.setRegParam(0.01);


// Define the pipeline that is used to create the linear regression
// model on the training data.
// In this case the pipeline contains one single stage/step (the model
// generation step).
Pipeline pipeline = new Pipeline().setStages(new PipelineStage[] {lr});
```

Linear regression and structured data: Example

```
// Execute the pipeline on the training data to build the
// classification model
PipelineModel model = pipeline.fit(training);

// Now, the classification model can be used to predict the class label
// of new unlabeled data
```

Linear regression and structured data: Example

```
// ***************************
// Prediction  step
// ***************************

// Read unlabeled data
// For the unlabeled data only the predictive attributes are available
// The value of the predicted target attribute is not available
// and must be predicted by applying the regression model inferred
// during the previous phase
JavaRDD<String> unlabeledData=sc.textFile(inputFileTest);
```

Linear regression and structured data: Example

```
// Map each unlabeled input record/data point of the input file to
// a LabeledPoint
JavaRDD<LabeledPoint> unlabeledRDD=unlabeledData.map(record ->
{
        String[] fields = record.split(",");

        // The last three cells of fields contain the (numerical) values of the
        // three predictive attributes
        // Create an array of doubles containing those three values
        double[] attributesValues = new double[3];

        attributesValues[0] = Double.parseDouble(fields[1]);
        attributesValues[1] = Double.parseDouble(fields[2]);
        attributesValues[2] = Double.parseDouble(fields[3]);
```

Linear regression and structured data: Example

```
                // Create a dense vector based in the content of attributesValues
                Vector attrValues= Vectors.dense(attributesValues);

                //The class label in unknown.
                //To create a LabeledPoint a label value must be specified
                // also for the unlabeled data. I set it to 0.
                //The specified value  does not impact on the prediction because
                // the label column is not used to perform the prediction
                double targetLabel = 0;

                // Return a new LabeledPoint
                return new LabeledPoint(targetLabel, attrValues);
});

// Create the DataFrame based on the new unlabeled data
Dataset<Row> test =
                ss.createDataFrame(unlabeledRDD, LabeledPoint.class);
```

Linear regression and structured data: Example

```
// Make predictions on test documents using the transform()
// method.
//The transform will only use the 'features' columns
Dataset<Row> predictions = model.transform(test);

//The returned Dataset<Row> has the following schema (attributes)
// - features: vector (values of the attributes)
// - label: double (value of the class label)
// - prediction: double (the predicted target attribute)

//Select only the features (i.e., the value of the attributes) and
// the predicted class for each record
Dataset<Row> predictionsDF=predictions.select("features",
"prediction");
```

Linear regression and structured data: Example

```
        // Save the result in an HDFS file
        JavaRDD<Row> predictionsRDD = predictionsDF.javaRDD();
        predictionsRDD.saveAsTextFile(outputPath);


        // Close the Spark Context object
        sc.close();
    }
}
```

Linear regression

Linear regression and textual data

- The linear regression algorithms can be used also when the input dataset is a collection of documents/texts
- Also in this case the text must be mapped to a set of continuous attributes

Linear regression and parameter setting

- The tuning approach that we used for the classification problem can also be used to optimize the regression problem
- The only difference is given by the used evaluator
  - In this case the difference between the actual value and the predicted one must be computed