Tópicos em Avanços Computacionais I

Multilayer Perceptron – DL4J

Joinvile Batista Junior

Configurando via Maven
propriedades do projeto

```
<properties>
   <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
   <java.version>1.8</java.version>
   <logback.version>1.2.3</logback.version>
   <nd4j.version>1.0.0-alpha</nd4j.version>
   <dl4j.version>1.0.0-alpha</dl4j.version>
   <spark.version>2.3.0</spark.version>
</properties>
```

Configurando via Maven
dependências

```
<dependencies>
      <dependency>
         <groupId>ch.qos.logback</groupId>
         <artifactId>logback-classic</artifactId>
         <version>${logback.version}</version>
      </dependency>

      <dependency>
          <groupId>org.apache.spark</groupId>
          <artifactId>spark-core_2.11</artifactId>
          <version>${spark.version}</version>
      </dependency>
```

- repetir dependências do Spark com artifactId
  - spark-sql_2.11 – spark-mllib_2.11

Configurando via Maven
dependências

```
      <dependency>
         <groupId>org.nd4j</groupId>
         <artifactId>nd4j-native</artifactId>
         <version>${nd4j.version}</version>
      </dependency>
      <dependency>
         <groupId>org.deeplearning4j</groupId>
         <artifactId>deeplearning4j-core</artifactId>
         <version>${dl4j.version}</version>
      </dependency>
```

Configurando via Maven
dependências

```
    <dependency>
       <groupId>org.eclipse.collections</groupId>
       <artifactId>eclipse-collections-api</artifactId>
       <version>10.0.0</version>
    </dependency>

    <dependency>
       <groupId>org.eclipse.collections</groupId>
       <artifactId>eclipse-collections</artifactId>
       <version>10.0.0</version>
    </dependency>

</dependencies>
```


Configurando via Maven
plugins

```
  <build>
     <plugins>
      <plugin>
            <groupId>org.apache.maven.plugins</groupId>
            <artifactId>maven-eclipse-plugin</artifactId>
            <version>2.9</version>
            <configuration>
               <downloadSources>true</downloadSources>
               <downloadJavadocs>false</downloadJavadocs>
            </configuration>
         </plugin>
```

Configurando via Maven
plugins

```
<plugin>
   <groupId>org.apache.maven.plugins</groupId>
   <artifactId>maven-compiler-plugin</artifactId>
   <version>3.5.1</version>
   <configuration>
      <source>1.8</source>
      <target>1.8</target>
   </configuration>
</plugin>
```


Configurando via Maven
plugins

```
<plugin>
   <groupId>org.apache.maven.plugins</groupId>
   <artifactId>maven-shade-plugin</artifactId>
   <version>2.4.3</version>
   <configuration>
      <shadeTestJar>true</shadeTestJar>
   </configuration>
</plugin>

   </plugins>
  </build>
</project>
```

Importações
Java básico

java.io
- File
- IOException

java.util
- HashMap
- Map

Importações
Spark

org.apache.spark
- ml
  - Pipeline
  - PipelineStage
  - feature.StringIndexer
- sql
  - Dataset
  - Row
  - SparkSession

Importações
datavec

org.datavec.api
- records.reader
  - RecordReader
  - impl.csv.CSVRecordReader
- split.FileSplit

Importações
ND4J

org.nd4j.linalg
- activations.Activation
- api.ndarray.INDArray
- learning.config.Adam
- lossfunctions.LossFunctions.LossFunction
- dataset
  - DataSet
  - api
    - iterator.DataSetIterator
    - preprocessor.NormalizerMinMaxScaler

Importações
DL4J

org.deeplearning4j
- datasets.datavec.RecordReaderDataSetIterator
- eval
  - Evaluation
  - EvaluationAveraging
- nn
  - api.OptimizationAlgorithm
  - conf
    - MultiLayerConfiguration
    - NeuralNetConfiguration
  - layers
    - DenseLayer
    - OutputLayer
  - multilayer.MultiLayerNetwork
  - weights.WeightInit

Dados da Classe e Main

```
public class App2_MLP_DL4J {

    private static int label_index = 7;
    private static int n_classes = 2;
    private static NormalizerMinMaxScaler pre_processor;
    private static String train_path = "data/train";
    private static String test_path = "data/test";

    public static void main(String[] args)
            throws IOException, InterruptedException {
        SparkSession spark = createSparkSession();
        Dataset<Row> training_data = readTrainingData(spark);
        doETL_AndSave(training_data);
        MultiLayerNetwork model = trainNetwork();
        evaluateNetwork(model);
    }
```

Criando uma Sessão Spark

```
private static SparkSession createSparkSession() {
    return SparkSession
            .builder()
            .master("local[*]")
            .config("spark.sql.warehouse.dir", "spark")
            .appName("App2_MLP_DL4J")
            .getOrCreate();
}
```


Lendo e Visualizando os Dados de Treinamento

```
private static Dataset<Row> readTrainingData(SparkSession spark) {
    Dataset<Row> training_data = spark.sqlContext()
            .read()
            .format("com.databricks.spark.csv")
            .option("header", "true")
            .option("inferSchema", "true")
            .load("data/train.csv");
    training_data.show();
    return training_data;
}
```

ETL

substituindo valores nulos por médios
selecionando relevantes

```
private static void doETL_AndSave(Dataset<Row> training_data) {
    Map<String, Object> mean_values
        = new HashMap<String, Object>();
    mean_values.put("Age", 30);
    mean_values.put("Fare", 32.2);
    Dataset<Row> not_null_training_data
        = training_data.na().fill(mean_values);

    Dataset<Row> relevant_not_null_training_data
        = not_null_training_data.drop
            ("PassengerId", "Name", "Ticket", "Cabin");
    relevant_not_null_training_data.show();
```


ETL

indexadores para colunas não numéricas

```
    StringIndexer sexIndexer = new StringIndexer()
        .setInputCol("Sex")
        .setOutputCol("sexIndex")
        .setHandleInvalid("skip");// skip column having nulls
    StringIndexer embarkedIndexer = new StringIndexer()
        .setInputCol("Embarked")
        .setOutputCol("embarkedIndex")
        .setHandleInvalid("skip");// skip column having nulls

    Pipeline pipeline = new Pipeline()
        .setStages(new PipelineStage[]{sexIndexer, embarkedIndexer});
```

ETL

substituindo as colunas não numéricas

```
Dataset<Row> relevant_not_null_only_numeric_training_data
  = pipeline
       .fit(relevant_not_null_training_data)
       .transform(relevant_not_null_training_data)
       .drop("Sex", "Embarked");
relevant_not_null_only_numeric_training_data.show();

Dataset<Row> relevant_only_numeric_training_data
   = relevant_not_null_only_numeric_training_data
       .select("Pclass", "Age", "SibSp", "Parch", "Fare", "sexIndex",
                "embarkedIndex", "Survived");
relevant_only_numeric_training_data.show();
```


ETL

criando dados de teste a partir dos dados de treinamento

```
Dataset<Row>[] splits
     = relevant_only_numeric_training_data.randomSplit
          (new double[]{0.7, 0.3});
Dataset<Row> reduced_training_data = splits[0];
Dataset<Row> test_data = splits[1];
```

ETL

salvando dados de treinamento e teste após ETL

```
    reduced_training_data
            .coalesce(1)// coalesce(1) writes DF in a single CSV
            .write()
            .format("com.databricks.spark.csv")
            .option("header", "false")
            .option("delimiter", ",")
            .save("data/train");
     test_data
            .coalesce(1)// coalesce(1) writes DF in a single CSV
            .write()
            .format("com.databricks.spark.csv")
            .option("header", "false")
            .option("delimiter", ",")
            .save("data/test");
}
```


Treinamento da Rede

hiperparâmetros e camada de entrada

```
private static MultiLayerNetwork trainNetwork()
            throws IOException, InterruptedException {
      int n_epochs = 1000; // 10000
      int seed = 123;
      int n_inputs = label_index;
      int n_outputs = n_classes;

      DenseLayer input_layer = new DenseLayer.Builder()
            .weightInit(WeightInit.XAVIER)
            .activation(Activation.RELU)
            .nIn(n_inputs)
            .nOut(16)
            .build();
```

Treinamento da Rede
camadas escondidas

```
DenseLayer hidden_layer_1 = new DenseLayer.Builder()
        .weightInit(WeightInit.XAVIER)
        .activation(Activation.RELU)
        .nIn(16).nOut(32)
        .build();

DenseLayer hidden_layer_2 = new DenseLayer.Builder()
        .weightInit(WeightInit.XAVIER)
        .activation(Activation.RELU)
        .nIn(32).nOut(16)
        .build();
```


Treinamento da Rede
camada de saída

```
OutputLayer output_layer
        = new OutputLayer.Builder(LossFunction.XENT)
                // XENT for Binary Classification
        .weightInit(WeightInit.XAVIER)
        .activation(Activation.SOFTMAX)
        .nIn(16).nOut(n_outputs)
        .build();
```

Treinamento da Rede
configurando a rede

```
MultiLayerConfiguration MLPconfiguration
      = new NeuralNetConfiguration.Builder()
      .seed(seed)
      .optimizationAlgo
           (OptimizationAlgorithm.STOCHASTIC_GRADIENT_DESCENT)
      .weightInit(WeightInit.XAVIER)
      .updater(new Adam(0.01)) // 0.0001
      .list()
      .layer(0, input_layer)
      .layer(1, hidden_layer_1)
      .layer(2, hidden_layer_2)
      .layer(3, output_layer)
      .pretrain(false).backprop(true).build();
```


Treinamento da Rede
iterador dos dados de treinamento
normalização dos dados de treinamento

```
int batch_size_training = 128;
DataSetIterator cleared_training_data = readCSVDataset
    (train_path, batch_size_training, label_index, n_classes);

pre_processor = new NormalizerMinMaxScaler();
pre_processor.fit(cleared_training_data);
cleared_training_data.setPreProcessor(pre_processor);
```

Treinamento da Rede

loop de treinamento

```
    MultiLayerNetwork model
      = new MultiLayerNetwork(MLPconfiguration);
    model.init();
    System.out.println("Train model....");
    for (int i = 0; i < n_epochs; i++) {
       model.fit(cleared_training_data);
    }
    return model;
}
```


Avaliação da Rede

iterador dos dados de teste
normalização dos dados de teste
avaliaçao das instâncias dos dados de teste

```
private static void evaluateNetwork(MultiLayerNetwork model)
        throws IOException, InterruptedException {
    int batch_size_test = 128;
    DataSetIterator cleared_test_data = readCSVDataset
        (test_path, batch_size_test, label_index, n_classes);
    cleared_test_data.setPreProcessor(pre_processor);

    Evaluation eval = new Evaluation(2); // for class 1
    while (cleared_test_data.hasNext()) {
       DataSet next = cleared_test_data.next();
       INDArray output = model.output(next.getFeatureMatrix());
       eval.eval(next.getLabels(), output);
    }
```

Avaliação da Rede
visualização das métricas de avaliação

```
    System.out.println(eval.stats());
    System.out.println(" – Example finished –");

    EvaluationAveraging averaging = EvaluationAveraging.Macro;
    double MCC = eval.matthewsCorrelation(averaging);
    System.out.println("Matthews correlation coefficient: " + MCC);
}
```


Iterador de Dados de Entrada

```
private static DataSetIterator readCSVDataset
        (String csvFileClasspath, int batchSize, int label_index,
          int n_classes) throws IOException, InterruptedException {
    RecordReader record_reader = new CSVRecordReader();
    File input = new File(csvFileClasspath);
    if (input.isDirectory()) {
        File[] csvs = input.listFiles(pathname -> {
            String str = pathname.toString();
            return str.endsWith(".csv");
        });
```

Iterador de Dados de Entrada

```
            if (csvs != null && csvs.length > 0) {
            File file = csvs[0];
            File new_file = null;
            if (csvFileClasspath.endsWith("train")) {
                new_file = new File("data/new_train.csv");
            } else {
                new_file = new File("data/new_test.csv");
            }
            file.renameTo(new_file);
            record_reader.initialize(new FileSplit(new_file));
          }
        } else {
            record_reader.initialize(new FileSplit(input));
        }
```


Iterador de Dados de Entrada

```
            DataSetIterator iterator
                = new RecordReaderDataSetIterator
                  (record_reader, batchSize, label_index, n_classes);
            return iterator;
    }
}
```