Spark Machine Learning Library

Igor Yakushin
ivy2@uchicago.edu

Introduction

- Spark Machine Learning Library - MLlib - is a distributed machine learning framework on top of Spark
- MLlib has RDD and DataFrame APIs
- RDD API is now in maintenance mode: bugs are fixed, no new features are added
- DataFrame API is catching up, once it does, RDD API will be removed
- Many common machine learning and statistical algorithms have been implemented and are shipped with MLlib which simplify large scale machine learning pipelines, including:
  - summary statistics, correlations, sampling, hypothesis testing
  - classification and regression: support vector machines, logistic regression, linear regression, decision trees, naive Bayes classification
  - cluster analysis methods including k-means
  - dimensionality reduction techniques such as SVD and PCA
  - feature extraction and transformation functions
  - optimization algorithms such as stochastic gradient descent

Logistic Regression

- In this example we have a training set consisting of class label (0 or 1) and 692 numerical features
- The data is sparse and is given in libsvm format in `data/sample_libsvm_data.txt`
- Linear regression model tries to predict the class y based on features x_i as follows:

$$y = \frac{1}{1 + e^{-(\sum_i w_i x_i + b)}}$$

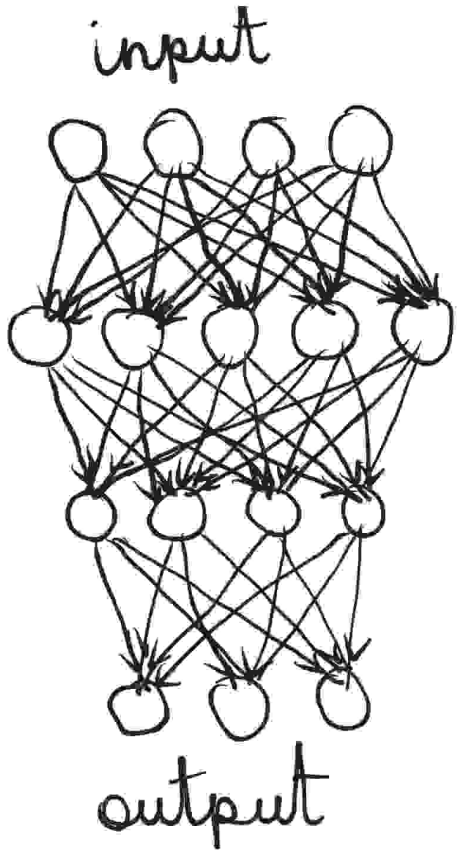
  where w_i and b are parameters that the model needs to learn to minimize error on the given training set.
- After the model is trained, the parameters are printed.
- Since most of them are zero, sparse format is used.

Logistic Regression

```
from pyspark.sql import SparkSession
from pyspark.ml.classification import LogisticRegression

spark = SparkSession.builder.getOrCreate()

# Load training data
training = spark.read.format("libsvm").\
                 load("data/sample_libsvm_data.txt")
lr = LogisticRegression(maxIter=10,
                        regParam=0.3, elasticNetParam=0.8)
# Fit the model
lrModel = lr.fit(training)

# Print the coefficients and intercept for logistic regression
print("Coefficients: " + str(lrModel.coefficients))
print("Intercept: " + str(lrModel.intercept))
```

Perceptron

- The data has
  3 classes and 4 features and is given in libsvm format
  `data/sample_multiclass_classification_data.txt`
- The model is a fully connected neural network with 4 layers.
- 60% of data is used as training set and 40% as validation set.
- 100 epochs is used for training, mini-batch size is 128.
- Prediction accuracy on the validation set is printed: 90%.



Perceptron

```
from pyspark.ml.classification
     import MultilayerPerceptronClassifier as mlp
from pyspark.ml.evaluation
     import MulticlassClassificationEvaluator as mce
spark = SparkSession.builder.getOrCreate()
data = spark.read.format("libsvm")\
    .load("data/sample_multiclass_classification_data.txt")
splits = data.randomSplit([0.6, 0.4], 1234)
train = splits[0]; test = splits[1]
layers = [4, 5, 4, 3]
trainer = mlp(maxIter=100, layers=layers, blockSize=128, seed=1234)
model = trainer.fit(train)
result = model.transform(test)
predictionAndLabels = result.select("prediction", "label")
evaluator = mce(metricName="accuracy")
print(evaluator.evaluate(predictionAndLabels))))
```