L10: Scalable Machine Learning with Spark

ANLY 502: Massive Data Fundamentals

Simson Garfinkel & Marck Vaisman

April 3, 2017

Agenda

Administrivia

- A04 graded - issues with solutions
- Academic paper research
- Comments on project proposals - several require resubmits
- No class next two weeks: 4/10 - Passover, 4/17 - Easter Break

Starting a cluster using the AWS Command Line Tool and specifying different parameters

Spark MLlib + ML

Hands-on Examples/Labs using both Pyspark and Scala on Jupyter

Launch a cluster using the AWS CLI + bootstrap

http://www.ec2instances.info/

l10-launch-cluster-class.sh

ahora.txt | l10-launch-cluster-class.sh

```
aws emr create-cluster \
--applications Name=Hadoop Name=Spark Name=Hive Name=Pig Name=Tez Name=Ganglia \
--release-label emr-5.4.0 \
--name "anly502 ml spark master 1-m4.2xlarge, core 3-m4.2xlarge" \
--service-role EMR_DefaultRole \
--instance-groups '[{
            "InstanceCount": 3,
            "InstanceGroupType": "CORE",
            "InstanceType": "m4.2xlarge",
            "Name": "Core instance group - 2"
        }, {
            "InstanceCount": 1,
            "InstanceGroupType": "MASTER",
            "InstanceType": "m4.2xlarge",
            "Name": "Master instance group - 1"
        }]' \
--bootstrap-actions '[{
            "Path": "s3://gu-anly502/bootstrap2-ipython.sh",
            "Name": "ANLY502 ipython Bootstrap"
        },  {
            "Path": "s3://gu-anly502/bootstrap2-jupyter.sh",
            "Name": "ANLY502 Jupyter/Python3 Bootstrap"
        }]' \
--ec2-attributes '{
            "KeyName": "mv-mbp",
            "InstanceProfile": "EMR_EC2_DefaultRole",
            "SubnetId": "subnet-a50def99"
        }'

```

Apache Spark MLlib

Spark MLlib

Spark component that provides the machine learning/data mining algorithms

- Pre-processing techniques
- Classification
- Clustering
- Itemset mining

MLlib API is divided into two packages

- org.apache.spark.mllib (original API - predates DataFrames)
  - *It contains the original APIs built on top of RDDs*
- org.apache.spark.ml (newer API)
  - *It provides higher-level API built on top of DataFrames for constructing ML* ***pipelines***
  - *It is recommended because with DataFrames the API is more versatile and flexible*
  - *It provides the* ***pipeline*** *concept*

Spark MLlib is based on a set of basic local and distributed data types

- Local vector
- Labeled point
- Local matrix
- Distributed matrix

DataFrames for ML are built on top of these basic data types

Local vectors (RDD)

Local **org.apache.spark.mllib.linalg.Vector** objects are used to store vectors of double values

- Both dense and sparse vectors are supported

The MLlib algorithms works on vectors of double

- Non double attributes/values must be mapped to double values

Dense and sparse representations are supported

- E.g., a vector (1.0, 0.0, 3.0) can be represented
  - *in dense format as [1.0, 0.0, 3.0]*
  - *or in sparse format as (3, [0, 2], [1.0, 3.0])*
    - *where 3 is the size of the vector*
    - *[0,2] contains the indexes of the non-zero cells*
    - *[1.0, 3.0] contains the values of the non-zero cells*

Local vectors (RDD)

The following Scala code shows how a vector can be created in Spark

```
import org.apache.spark.mllib.linalg.Vector;

import org.apache.spark.mllib.linalg.Vectors;

// Create a dense vector (1.0, 0.0, 3.0). Vector dv = Vectors.dense(1.0, 0.0, 3.0);

// Create a sparse vector (1.0, 0.0, 3.0) by
// specifying its indices and values corresponding // to non-zero entries


Vector sv = Vectors.sparse(3, new int[] {0, 2},

new double[] {1.0, 3.0});
```

Labeled Points (RDD)

Local **org.apache.spark.mllib.regression.LabeledPoint** objects are local vector associated with a label
The label is a double value

For the classification problem, each class label is associated with an integer value ranging from 0 to C-1, where C is the number of distinct classes

Both dense and sparse vectors associated with a label are supported

In MLlib, labeled points are used by many supervised learning algorithms

The following code shows how a LabeledPoint can be created in Spark

```
import org.apache.spark.mllib.linalg.Vectors;
import org.apache.spark.mllib.regression.LabeledPoint;

// Create a labeled point with a positive label and // a dense feature vector.
LabeledPoint pos = new LabeledPoint(1,

Vectors.dense(1.0, 0.0, 3.0));

// Create a labeled point with a negative label and a sparse feature // vector.
LabeledPoint neg = new LabeledPoint(0, Vectors.sparse(3, new int[] {0, 2}, new double[]
{1.0, 3.0}));
```

Sparse labeled data

Frequently the training data are sparse

- E.g., textual data are sparse. Each document contains only a subset of the possible words
- Hence, sparse vectors are used

MLlib supports reading training examples stored in the LIBSVM format

- It is a commonly used format that represents each document/record as a sparse vector

The LIBSVM format

- Is a text format in which each line represents a labeled sparse feature vector using the following format:
- label index1:value1 index2:value2 ...

where

- label is an integer associated with the class label
- the indexes are one-based (i.e., integer indexes starting from 1) representing the features
- the values are the (double) values of the features
- After loading, the feature indexes are converted to zero-based (i.e., integer indexes starting from 0)

Spark ML

DataFrame

- Spark ML uses DataFrames from Spark SQL as ML datasets, which can hold a variety of data types
- DataFrame could have different columns storing text, feature vectors, (true) labels, and predictions

Transformer

- A Transformer is an algorithm which can transform one DataFrame into another DataFrame
- 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 classification model is a Transformer which can be applied on a DataFrame with features and transforms it into a DataFrame with also predictions

Spark ML

Estimator

- An Estimator is an algorithm which can be applied on a DataFrame to produce a Transformer (a model)
- An Estimator implements a method fit(), which accepts a DataFrame and produces a Model of type Transformer
- An Estimator abstracts the concept of a learning algorithm or any algorithm that fits or trains on an input dataset and returns a model
- A classification algorithm such as Logistic Regression is an Estimator, and calling fit() on it a Logistic Regression Model is built, which is a Model and hence a Transformer

Pipeline

- A Pipeline chains multiple Transformers and Estimators together to specify a Machine learning/ Data Mining workflow
- The output of a transformer/estimator is the input of the next one in the pipeline
  - *a simple text document processing workflow aiming at building a classification model includes several steps*
    - *Split each document into a set of words*
    - *Convert each set of words into a numerical feature vector*
    - *Learn a prediction model using the feature vectors and the associated class labels*

Spark ML

Parameter

- All Transformers and Estimators share a common API for specifying parameters

Summary

- In the new APIs of Spark MLlib the use of the pipeline approach is preferred
- This approach is based on the following steps
  - *The set of Transformers and Estimators that are needed are instantiated*
  - *A pipeline object is created and the sequence of transformers and estimators associated with the pipeline are specified*
  - *The pipeline is executed and model is created*
  - *(optional) The model is applied on new data*

Important!

All the clustering algorithms available in Spark work only with numerical data

- Categorical values must be mapped to integer values (i.e, numerical values)

Logon on AWS console

Before you start the cluster, go to security groups for the ElasticMapReduce-master and open port 8888

Start the cluster using the AWS CLI as shown or manually with both bootstrap actions (s3://gu-anly502/bootstrap2-ipyhton.sh and s3://gu-anly502/bootstrap2-jupyter.sh.) Use m4.2xlarge instances, 1 master and 3 core nodes

Once cluster starts, ssh into master node

```
git clone https://github.com/jhlch/ds-for-telco
git clone https://github.com/pbugnion/s4ds
hadoop fs -mkdir telco
hadoop fs -put churn.all telco/
```