Machine Learning in Spark

Shelly Garion

IBM Research -- Haifa
Sparks MLLib

Large Scale Machine Learning on Apache Spark

Why MLLib?

Mahout?

LIBLINEAR?

H2O?

Vowpal Wabbit?

MATLAB?

R?

GraphLab?

scikit-learn?

Weka?
Machine Learning Algorithms

- **Classification**
  - Logistic regression
  - Linear support vector machine (SVM)
  - Naïve Bayes
  - Decision trees and forests

- **Regression**
  - Generalized linear regression (GLM)

- **Recommendation**
  - Alternating least squares (ALS)

- **Clustering**
  - K-means and Streaming K-means
  - Gaussian mixture
  - Power iteration clustering (PIC)
  - Latent Dirichlet allocation (LDA)

- **Dimensionality reduction**
  - Singular value decomposition (SVD)
  - Principal component analysis (PCA)

- **Feature extraction & selection**
  - ...

See: https://spark.apache.org/docs/latest/mllib-guide.html
Performance of MLLib

- It is built on Apache Spark, a fast and general engine for large-scale data processing.
- Run programs up to 100x faster than Hadoop MapReduce in memory, or 10x faster on disk.

Logistic Regression

ALS Results

https://spark.apache.org/
Performance of MLLib

- Speed-up between MLLib versions

Figure 2: (a) Benchmarking results for ALS. (b) MLLib speedup between versions.

Example: K-Means Clustering

Goal:
Segment tweets into clusters by geolocation using Spark MLLib K-means clustering

Example: K-Means Clustering

To run the k-means algorithm in Spark, we need to first read the csv file

```scala
val sc = new SparkContext("local[4]", "kmeans")
// Load and parse the data, we only extract the latitude and longitude of each line
val data = sc.textFile(arg)
val parsedData = data.map {
  line =>
    Vectors.dense(line.split(',').slice(0, 2).map(_.toDouble))
}
```

Then we can run the spark kmeans algorithm:

```scala
val iterationCount = 100
val clusterCount = 10
val model = KMeans.train(parsedData, clusterCount, iterationCount)
```

Example: K-Means Clustering

From the model we can get the cluster centers and group the tweets by cluster:

```scala
val clusterCenters = model.clusterCenters map (_.toArray)
val cost = model.computeCost(parsedData)
println("Cost: " + cost)

val tweetsByGroup = data
  .map {_.split(',').slice(0, 2).map(_.toDouble)}
  .groupBy{rdd => model.predict(Vectors.dense(rdd))}
  .collect()
sc.stop()
```

Example: K-Means Clustering

Spark Ecosystem
Spark SQL & MLLib

// Data can easily be extracted from existing sources, such as Apache Hive.
val trainingTable = sql(""")
    SELECT e.action,
    u.age,
    u.latitude,
    u.longitude
FROM Users u
JOIN Events e
ON u.userId = e.userId"")

// Since `sql` returns an RDD, the results of the above query can be easily used in MLLib.
val training = trainingTable.map { row =>
    val features = Vectors.dense(row(1), row(2), row(3))
    LabeledPoint(row(0), features)
}

val model = SVMWithSGD.train(training) // SVM using Stochastic Gradient Descent

Spark Ecosystem
Spark Streaming & MLLib

// collect tweets using streaming

// train a k-means model
val model: KMeansModel = ...

// apply model to filter tweets
val tweets = TwitterUtils.createStream(ssc, Some(authorizations(0)))
val statuses = tweets.map(_.getText)
val filteredTweets =
    statuses.filter(t => model.predict(featurize(t)) == clusterNumber)

// print tweets within this particular cluster
filteredTweets.print()
Spark Ecosystem
GraphX & MLLib

// assemble link graph
val graph = Graph(pages, links)
val pageRank: RDD[(Long, Double)] = graph.staticPageRank(10).vertices

// load page labels (spam or not) and content features
val labelAndFeatures: RDD[(Long, (Double, Seq((Int, Double))))) = ...
val training: RDD[LabeledPoint] =
  labelAndFeatures.join(pageRank).map {
    case (id, ((label, features), pageRank)) =>
      LabeledPoint(label, Vectors.sparse(features ++ (1000, pageRank))
  }

// train a spam detector using logistic regression
val model = LogisticRegressionWithSGD.train(training)

Xiangrui Meng, MLLib: scalable machine learning on Spark, Spark Workshop April 2014,
http://stanford.edu/~rezab/sparkworkshop/
Machine Learning Pipeline with Spark

- Data pre-processing
- Feature extraction
- Model fitting
- Model training
- Validation
- Model prediction
Machine Learning Pipeline with Spark

Machine Learning Pipeline with Spark

- **ML Dataset:**
  - DataFrame from Spark SQL
    - could have different columns storing text, feature vectors, true labels, and predictions

- **Transformer:**
  - Feature transformers (e.g., OneHotEncoder)
  - Trained ML models (e.g., LogisticRegressionModel)

- **Estimator:**
  - ML algorithms for training models (e.g., LogisticRegression)

- **Evaluator:**
  - Evaluate predictions and compute metrics, useful for tuning algorithm parameters
    - (e.g., BinaryClassificationEvaluator)

- **Pipeline:** chains multiple Transformers and Estimators together to specify an ML workflow
Machine Learning Pipeline with Spark

Learning:

Pipeline (Estimator)

```
Pipeline.fit()
```

Raw text → Words → Feature vectors → Logistic Regression Model

Model:

PipelineModel (Transformer)

```
PipelineModel.transform()
```

Raw text → Words → Feature vectors → Predictions

https://spark.apache.org/docs/latest/ml-guide.html
Example: Alternating Least Squares (ALS)

Collaborative filtering

- Recover a rating matrix from a subset of its entries.

ALS Implementation in MLlib

How to scale to 100,000,000,000 ratings?

Example: Alternating Least Squares (ALS)

Model R as product of user and movie feature matrices A and B of size U×K and M×K

![Diagram of R = A \cdot B^T]

Alternating Least Squares (ALS)

» Start with random A & B
» Optimize user vectors (A) based on movies
» Optimize movie vectors (B) based on users
» Repeat until converged

Example: Alternating Least Squares (ALS)

1. Start with random $A_1$, $B_1$
2. Solve for $A_2$ to minimize $\|R - A_2B_1^T\|$
3. Solve for $B_2$ to minimize $\|R - A_2B_2^T\|$
4. Repeat until convergence

Example: Alternating Least Squares (ALS)

Low-Rank Matrix Factorization:

Iterate:

\[ f[i] = \arg \min_{w \in \mathbb{R}^d} \sum_{j \in \text{Nbrs}(i)} (r_{ij} - w^T f[j])^2 + \lambda \|w\|_2^2 \]

Example: Alternating Least Squares (ALS)

Broadcast everything

- Master loads (small) data file and initializes models.
- Master broadcasts data and initial models.
- At each iteration, updated models are broadcast again.
- Works OK for small data.
- Lots of communication overhead - doesn't scale well.

Example: Alternating Least Squares (ALS)

Data parallel

- Workers load data
- Master broadcasts initial models
- At each iteration, updated models are broadcast again
- Much better scaling
- Works on large datasets
- Works well for smaller models (low K)

Example: Alternating Least Squares (ALS)

Fully parallel

- Workers load data.
- Models are instantiated at workers.
- At each iteration, models are shared via join between workers.
- Much better scalability.
- Works on large datasets

Implementation of ALS in Spark MLLib

ALS on Spark

\[ R = A B^T \]

- broadcast everything
- data parallel
- fully parallel
- block-wise parallel

Cache 2 copies of R in memory, one partitioned by rows and one by columns
Keep A & B partitioned in corresponding way
Operate on blocks to lower communication

Implementation of ALS in Spark MLLib

Communication: All-to-All VS. Communication: Block-to-Block

- users: u1, u2, u3; items: v1, v2, v3, v4
- shuffle size: $O(nnz \cdot k)$ (nnz: number of nonzeros, i.e., rating
- sending the same factor multiple times

- Shuffle size is significantly reduced.
- We cache two copies of ratings — InBlocks for users and InBlocks for items.

Implementation of ALS in Spark MLLib

References