SUMMIT

Tel Aviv

Building Machine Learning inference pipelines at scale

Julien Simon
Global Evangelist, AI & Machine Learning
@julsimon

aws SUMMIT Tel Aviv

Problem statement

- Real-life Machine Learning applications require more than predicting with a single model.
- Data may need pre-processing: normalization, feature engineering, dimensionality reduction, etc.
- Predictions may need post-processing: filtering, sorting, combining, etc.

Agenda

Build and deploy ML pipelines with minimal infrastructure drama!

1. Spark (on Amazon EMR)
2. Spark + Amazon SageMaker
3. Amazon SageMaker, aka Inference Pipelines

Building pipelines with Spark

aws SUMMIT Tel Aviv

Apache Spark

https://spark.apache.org/

- Open-source, distributed processing system.
- In-memory caching and optimized execution for fast performance (typically 100x faster than Hadoop).
- Batch processing, streaming analytics, machine learning, graph databases and ad hoc queries.
- API for Java, Scala, Python, R, and SQL.

Apache Spark – *DataFrame*

- Distributed collection of data organized into named columns
- Conceptually equivalent to a table in a relational database
- Wide array of sources: structured files, databases
- Wide array of formats: text, CSV, JSON, Avro, ORC, Parquet

```
{"name": "Jeff"}
{"name": "Boaz",   "age":72}
{"name": "Julien", "age":12}
```

```
df = spark.read.json("people.json")
df.show()
+----+-------+
| age| name  |
+----+-------+
|null| Jeff  |
| 72 | Boaz  |
| 12 | Julien|
+----+-------+
```

MLlib – Machine learning library

https://spark.apache.org/docs/latest/ml-guide.html

- Algorithms: classification, regression, clustering, collaborative filtering.
- Featurization: feature extraction, transformation, dimensionality reduction.
- Tools for constructing, evaluating and tuning pipelines
  - *Transformer* – a transform function that maps a *DataFrame* into a new one
    - Adding a column, changing the rows of a specific column, etc.
    - Predicting the label based on the feature vector
  - *Estimator* – an algorithm that trains on data
    - Consists of a *fit*() function that maps a *DataFrame* into a *Model*

Example: binary classification for text samples

https://github.com/apache/spark/blob/master/examples/src/main/scala/org/apache/spark/examples/ml/PipelineExample.scala

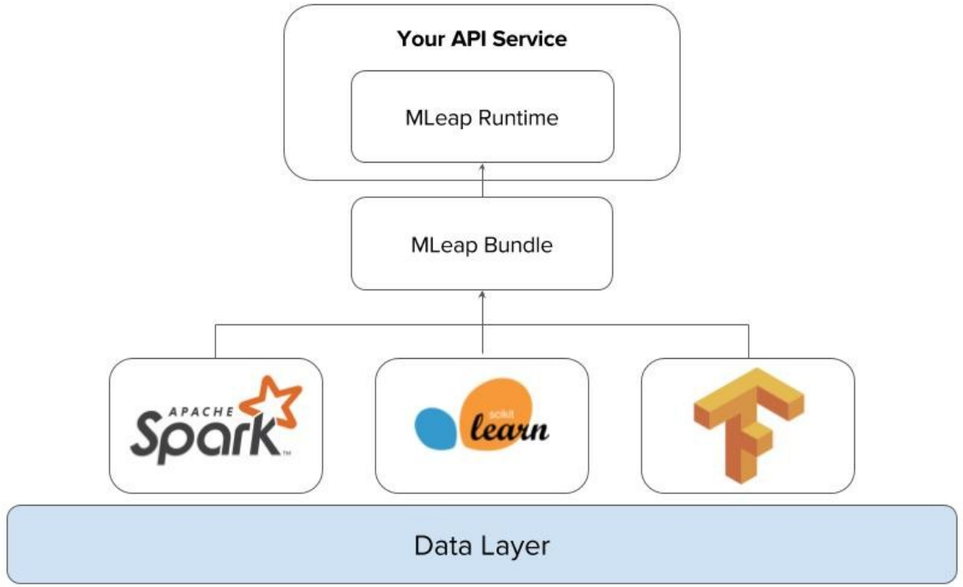
```
// Prepare training documents from a list of (id, text, label) tuples.
val training = <LOAD_TRAINING_DATA>

// Configure an ML pipeline with three stages: tokenizer, hashingTF, and lr.
val tokenizer = new Tokenizer().setInputCol("text”).setOutputCol("words")
val hashingTF = new HashingTF()
   .setNumFeatures(1000).setInputCol(tokenizer.getOutputCol).setOutputCol("features")
val lr = new LogisticRegression().setMaxIter(10).setRegParam(0.001)

val pipeline = new Pipeline().setStages(Array(tokenizer, hashingTF, lr))

// Fit the pipeline to training documents.
val model = pipeline.fit(training)

// Prepare test documents, which are unlabeled (id, text) tuples.
val test = <LOAD_TEST_DATA>

// Make predictions on test documents.
model.transform(test)
```

Demo

Serving SparkML predictions

- Train and predict in the same application (see previous example)
- Or save the model and load it in another Spark application
- Or export the model to PMML and load it elsewhere (Java, R, etc.)
- Or export the model to MLeap
  - http://mleap-docs.combust.ml/
  - Lightweight runtime independent from Spark
  - Interoperability between SparkML, TensorFlow and scikit-learn

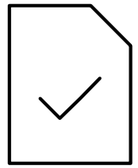
In any case, you need to build and maintain prediction infrastructure :-/



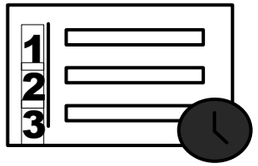
Building pipelines with Spark and Amazon SageMaker

aws SUMMIT Tel Aviv

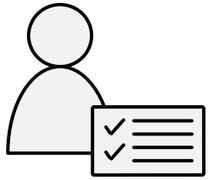
Amazon SageMaker:
Build, Train, and Deploy ML Models at Scale



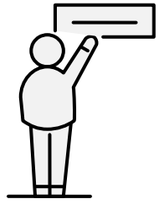
Collect and prepare training data



Choose and optimize your ML algorithm

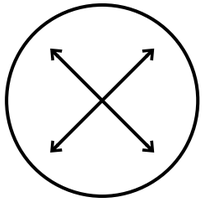


Set up and manage environments for training



Train and Tune ML Models

Deploy models in production



Scale and manage the production environment

intuit

tinder

KIA

Liberty Mutual

CONVOY

SIEMENS

DOW JONES

SONY

aws SUMMIT Tel Aviv

Amazon SageMaker SDK for Spark

https://github.com/aws/sagemaker-spark

- Python and Scala SDK, for Apache Spark 2.1.1 and 2.2.
- Pre-installed on EMR 5.11 and later.

- Train, import, deploy and predict with SageMaker models directly from your Spark application.
  - Standalone,
  - Integration in Spark MLlib pipelines.

- *DataFrames* in, *DataFrames* out:
  automatic data conversion to and from protobuf (crowd goes wild!)

Reason #1 - Decouple ETL and Machine Learning

- Different workloads require different instance types
  - Say, R4 for ETL, P3 for training and C5 for prediction?
  - If you need GPUs for training, running ETL on GPU instances wouldn't be the best option…
- Size and scale them independently
  - Avoid oversizing your Spark cluster.
  - Avoid time-consuming resizing operations on Amazon EMR.
  - Run ETL once, train many models in parallel.

Reason #2 - Run any ML algorithm in any language

- Spark MLlib is great, but you may need something else
- Other ML algorithms
- Deep Learning libraries, like TensorFlow or Apache MXNet
- Your own custom code in any language

Reason #3 - Get the best prediction performance

- Perform ML predictions without using Spark.
  - Save the overhead of the Spark framework
  - Save loading your data in a *DataFrame*
  - Amazon SageMaker can deploy MLeap models

- Improve latency for small-batch predictions.
  - It can be difficult to achieve low-latency predictions with Spark ML models
  - Get real-time predictions with models hosted in Amazon SageMaker
  - Use optimized instances for prediction

Sample use cases for Spark and SageMaker

- Data preparation and feature engineering + training
- Data transformation + batch prediction (model reuse)
- Data cleaning/enrichment with predictions
  - Predict missing values instead of using median.
  - Add new predicted features.
- Deploying a SparkML model at scale

Demos

ETL on Spark, train on SageMaker, predict on Spark
ETL and train on Spark, deploy on SageMaker

Building pipelines with Amazon SageMaker

aws SUMMIT Tel Aviv

Inference Pipelines

- Linear sequence of 2-5 containers that process inference requests
  - Feature engineering with scikit-learn or SparkML (on AWS Glue or Amazon EMR)
  - Predict with built-in or custom containers
  - The pipeline is deployed as a single model
- Useful to preprocess, predict, and post-process
- Available for real-time prediction and batch transform

Getting started

https://ml.aws

https://aws.amazon.com/sagemaker

https://github.com/awslabs/amazon-sagemaker-examples

https://medium.com/@julsimon

Please complete the session survey.

Thank you!

Julien Simon
Global Evangelist, AI & Machine Learning
@julsimon