MLflow: A Platform for Productionizing Machine Learning

Matei Zaharia
Databricks and Stanford University
Outline

- Challenges using ML in industry
- ML Platforms as an emerging abstraction
- MLflow overview and interesting use cases
About Databricks

Data and ML cloud platform used by >7000 customers
- Millions of VMs and 100,000s of users

Some of our ML customers:

- Shell
- T-Mobile
- H&M
- Regeneron
- Adobe
- HP
- HSBC
- Viacom
- NIH
- Nationwide
ML is Being Adopted for Critical Applications

Nationwide

Price insurance policies based on data and ML
(core business for >90 years!)

Shell

Manage inventory and supply chains
(affects company’s entire cost/revenue)

HSBC

Fraud detection & personalization on 170 PB data
(directly impacts profitability & compliance)
But ML is Different from Traditional Software

<table>
<thead>
<tr>
<th>Traditional Software</th>
<th>Machine Learning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Goal: meet a functional specification</td>
<td>Goal: optimize a metric (e.g. prediction accuracy)</td>
</tr>
<tr>
<td>Quality depends only on application code</td>
<td>Quality depends on input data and tuning parameters</td>
</tr>
<tr>
<td>Pick one software stack</td>
<td>Compare + combine many libraries and algorithms for the same task</td>
</tr>
</tbody>
</table>
Production ML is Even Harder

ML apps must be fed new data to keep working

Design, retraining & inference often done by different people
A Solution is Emerging: ML Platforms

Software to manage the ML development and deployment process, from data to experimentation to production

Examples: Google TFX, Facebook FBLearner, Uber Michelangelo

Typical concerns:

- Data management
- Experiment management
- Model management
- Deployment for inference
- Reproducibility
- Testing & monitoring

All through a consistent interface!
ML Platforms in 2018

Each company largely designing its own platform, with limited scope
- Specific libraries (e.g. TensorFlow for TFX)
- Specific deployment environment (e.g. Kubernetes on AWS)

ML Platform team often becomes a bottleneck in the organization

Can we provide the same benefits with an open platform?
mlflow: An Open Source ML Platform

Based on an **open interface** design philosophy: make it easy to connect arbitrary ML code & tools into the platform

- Simple command-line and REST APIs rather than environment-specific
- Easy to add to existing software

---

**mlflow TRACKING**
Experiment and metric tracking

**mlflow PROJECTS**
Reproducible execution

**mlflow MODELS**
Model packaging and deployment

**mlflow MODEL REGISTRY**
Model management

Built-in connectors
2 million downloads/month on PyPI
260 open source contributors
4x annual growth
1.5M runs/week on Databricks
MLflow Components
MLflow Tracking
Get visibility into experiments and production runs

```python
data = load_text(file)
ngrams = extract_ngrams(data, N=n)
model = train_model(ngrams, learning_rate=lr)
score = compute_accuracy(model)

print("For n=%d, lr=%f: accuracy=%f" % (n, lr, score))
```

What version of my code was this result from?
MLflow Tracking
Get visibility into experiments and production runs

```python
mlflow.keras.autolog()
data = load_text(file)
ngrams = extract_ngrams(data, N=n)
model = train_model(ngrams, learning_rate=lr)
score = compute_accuracy(model)

# Or log custom info if desired
mlflow.log_param("country", "US")
```

Track parameters, metrics, output files & code version
Tracking UI: Inspecting Runs

Language Model

Experiment ID: 0
Artifact Location: /Users/matei/mlflow/mlruns/0

Search Runs: metrics.rmse < 1 and params.model = 'tree'
State: Active
Filter Params: alpha, lr
Filter Metrics: rmse, r2

10 matching runs

<table>
<thead>
<tr>
<th>Date</th>
<th>User</th>
<th>Source</th>
<th>Version</th>
<th>Parameters</th>
<th>Metrics</th>
</tr>
</thead>
<tbody>
<tr>
<td>2018-10-02</td>
<td>matei</td>
<td>lang_model.py</td>
<td>e55d56</td>
<td>input_file: data.txt, lr: 2.0, n: 1</td>
<td>accuracy: 0.77, f1: 0.704</td>
</tr>
<tr>
<td>2018-10-02</td>
<td>matei</td>
<td>lang_model.py</td>
<td>e55d56</td>
<td>input_file: data.txt, lr: 1.0, n: 2</td>
<td>accuracy: 0.254, f1: 0.222</td>
</tr>
<tr>
<td>2018-10-02</td>
<td>matei</td>
<td>lang_model.py</td>
<td>e55d56</td>
<td>input_file: data.txt, lr: 2.0, n: 4</td>
<td>accuracy: 0.835, f1: 0.609</td>
</tr>
<tr>
<td>2018-10-02</td>
<td>matei</td>
<td>lang_model.py</td>
<td>e55d56</td>
<td>input_file: data.txt, lr: 1.0, n: 1</td>
<td>accuracy: 0.663, f1: 0.468</td>
</tr>
<tr>
<td>2018-10-02</td>
<td>matei</td>
<td>lang_model.py</td>
<td>e55d56</td>
<td>input_file: data.txt, lr: 0.2, n: 4</td>
<td>accuracy: 0.034, f1: 0.032</td>
</tr>
<tr>
<td>2018-10-02</td>
<td>matei</td>
<td>lang_model.py</td>
<td>e55d56</td>
<td>input_file: data.txt, lr: 0.1, n: 4</td>
<td>accuracy: 0.177, f1: 0.16</td>
</tr>
</tbody>
</table>
Tracking UI: Comparing Runs

Language Model

Experiment ID: 0  Artifact Location: /Users/matei/mlflow/mlruns/0

Search Runs: metrics.rmse < 1 and params.model = "tree"  State: Active  Search

Filter Params: alpha, lr  Filter Metrics: rmse, r2

10 matching runs  Compare  Delete  Download CSV

<table>
<thead>
<tr>
<th>Date</th>
<th>User</th>
<th>Source</th>
<th>Version</th>
<th>Parameters</th>
<th>Metrics</th>
</tr>
</thead>
<tbody>
<tr>
<td>2018-10-02</td>
<td>matei</td>
<td>lang_model.py</td>
<td>e55d56</td>
<td>input_file</td>
<td>accuracy</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>data.txt</td>
<td>2.0 1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>data.txt</td>
<td>1.0 2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>data.txt</td>
<td>2.0 4</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>data.txt</td>
<td>1.0 1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>data.txt</td>
<td>0.2 4</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>data.txt</td>
<td>0.1 4</td>
</tr>
</tbody>
</table>
MLflow Projects

Package code + dependencies for reusable workflows

my_project/
  └── MLproject
      └── conda.yaml
          entry_points:
              main:
                  parameters:
                      training_data: path
                      lr: {type: float, default: 0.1}
              command: python main.py {training_data} {lr}

$ mlflow run git://<my_project>
mlflow.run(“git://<my_project>”, ...)
Composing Projects

```python
r1 = mlflow.run("ProjectA", params)
if r1 > 0:
    r2 = mlflow.run("ProjectB", ...)
else:
    r2 = mlflow.run("ProjectC", ...)

r3 = mlflow.run("ProjectD", r2)
```

<table>
<thead>
<tr>
<th>Date</th>
<th>User</th>
<th>batch_size</th>
<th>epochs</th>
<th>lr</th>
</tr>
</thead>
<tbody>
<tr>
<td>2018-12-07 12:14:08</td>
<td>matei</td>
<td>512</td>
<td>20</td>
<td>0.05</td>
</tr>
<tr>
<td>2018-12-07 12:14:05</td>
<td>matei</td>
<td>512</td>
<td>40</td>
<td>0.05</td>
</tr>
<tr>
<td>2018-12-07 12:14:05</td>
<td>matei</td>
<td>512</td>
<td>40</td>
<td>0.05</td>
</tr>
<tr>
<td>2018-12-07 12:12:23</td>
<td>matei</td>
<td>512</td>
<td></td>
<td>0.05</td>
</tr>
</tbody>
</table>
MLflow Models

Generic format to package & deploy models from any library

TensorFlow

Python Flavor

ONNX Flavor

Model Logic

Batch & Stream Scoring

Online Serving

Databricks

Packaging Format

Evaluation & Debug Tools

LIME

TCAV

Spark
Example MLflow Model

```
my_model/
├── MLmodel
│   │ run_id: 769915006efd4c4bbd662461
│   │ time_created: 2018-06-28T12:34
│   │ flavors:
│   │     tensorflow:
│   │       saved_model_dir: estimator
│   │       signature_def_key: predict
│   │     python_function:
│   │       loader_module: mlflow.tensorflow
└── estimator/
    ├── saved_model.pb
    │ spark_udf = pyfunc.spark_udf(<run_id>)
```

Usable by tools that understand TensorFlow model format

Usable by any tool that can run Python (Docker, Spark, etc)

```
$ mlflow pyfunc serve -r <run_id>
```

...
MLflow Model Registry

GitHub-like environment for managing and reviewing models

Model Developer

Reviewer, Automated Tools

Application Developer

Batch Scoring

Online Serving
Predicts airline delays (in minutes) using the best Spark RF model from the AutoML Toolkit.
Registered Models > Airline_Delay_SparkML > Version 5

Registered At: 2019-10-11 12:44:44  Creator: clemens@demo.com  
Last Modified: 2019-10-14 12:19:32  Source Run: Run 6151fe768a5e49d39076b07448e60d57

Stage: Staging

Description

Improved the Airline delay model using a GBDT. See run for improved metrics.

Pending Requests

<table>
<thead>
<tr>
<th>Request</th>
<th>Request by</th>
<th>Actions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Transition to</td>
<td><a href="mailto:matei@demo.com">matei@demo.com</a></td>
<td>Approve</td>
</tr>
<tr>
<td>→ Production</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Activities

- clemens@demo.com rejected a stage transition → None 5 minutes ago
- matei@demo.com applied a stage transition None → Staging 4 minutes ago
- matei@demo.com requested a stage transition Staging → Production 4 minutes ago

Tested this offline, looks good to launch!
Tag and Search APIs for Automated CI/CD

**Tags** to track custom metadata for a model version, e.g. test results

**Search API** to automate model management and MLOps actions
How are Companies Using MLflow?
“Expected” Use Cases

- Track experiments during model design
- Track performance of continuous training & deployment pipelines
- Deploy the same model for batch and real-time scoring
- Run pipelines deterministically in different environments
- CI/CD using Model Registry stages and APIs
Example Use Cases

- **T-Mobile**
  Manage the models for ad fraud detection, including monitoring for drift in over 200 metrics.

- **H&M**
  Let data scientists spend 70%-90% of their time on model development instead of tuning and monitoring.

- **ABN AMRO**
  Automated and consistent deployment of 100+ models, from fraud detection, to marketing, to logistics.
Interesting Use Case: Massive # of Models

Company wants to train a separate model for each {facility, customer, chemical processing machine, ...}

- Avoid interference across these entities
- Preserve privacy & compliance

Examples:

- ExxonMobil: Predictive maintenance
- Quby: Energy grid
- Enterprise Software Company: Per-customer models
Interesting Use Case: Massive # of Models

Company wants to train a separate model for each \{facility, customer, chemical processing machine, ...\}:

- Avoid interference across these entities
- Preserve privacy & compliance

**Solution:** “hands-free” ML with large-scale analytics

- Train millions of models in parallel using an AutoML library on each entity
- Query experiment metrics using analytics tools (e.g. MLflow -> Pandas API)
- Run online or batch inference with the models
Interesting Use Case: Experiments Beyond ML

Systems like MLflow can add structure to other experimentation tasks:

- Develop, review & publish visualizations for COVID-19 data using the MLflow Model Registry
- Tune hundreds of Hyperloop engineering design parameters in simulation to optimize efficiency
Interesting Use Case: Reproducibility & Explainability

Government regulators and highly-regulated companies want:

- Documentation of every piece of data and code that went into a result (many have built their own lineage systems for this)
- Explanation of models
- AutoML to demonstrate they employed best practices
Recreate exact configuration of an experiment run
Recently Added Features
Spark and Delta Lake auto-logging: track data sources read and data versions in Delta Lake

```python
with mlflow.start_run(run_name='keras'):
    # log model and datasource
    mlflow.keras.autolog()
    mlflow.spark.autolog()

    df = spark.read.format("delta")
        .option("versionAsOf", 2)
        .load("/delta/clemens_windfarm")
```

To reload data version:

```python
df = spark.read.format("delta")
    .option("versionAsOf", 2)
    .load("/delta/clemens_windfarm")
```
**mlflow Tracking for Model Schemas**

Record what fields are consumed & produced by the model to prevent data mismatches

```python
with mlflow.start_run(run_name='keras'):
    # log model and datasource
    mlflow.keras.autolog()
    mlflow.spark.autolog()

    sig = infer_signature(X_train, y_train)
```

**Schema**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Inputs (7)</strong></td>
<td></td>
</tr>
<tr>
<td>user_purchases_7d</td>
<td>long</td>
</tr>
<tr>
<td>user_purchases_14d</td>
<td>long</td>
</tr>
<tr>
<td>user_location</td>
<td>string</td>
</tr>
<tr>
<td>user_last_login</td>
<td>string</td>
</tr>
<tr>
<td>item_inventory</td>
<td>long</td>
</tr>
<tr>
<td><strong>Outputs (1)</strong></td>
<td></td>
</tr>
<tr>
<td>prediction</td>
<td>double</td>
</tr>
</tbody>
</table>

---

**Note:**
- The code snippet demonstrates how to use `mlflow` for tracking models and their input/output schemas to prevent data mismatches.
- The table shows the inferred signature with various fields and their types.
- This approach ensures that the model's inputs and outputs are consistently tracked across runs.
Tracking for Interpretability

SHAP library feature importances and visualizations

with mlflow.start_run(run_name='keras'):
    # log model and datasource
    mlflow.keras.autolog()
    mlflow.spark.autolog()

    sig = infer_signature(X_train, y_train)

    mlflow.shap.logExplanation(model, X_train[:100])

```
with mlflow.start_run(run_name='keras'):
    # log model and datasource
    mlflow.keras.autolog()
    mlflow.spark.autolog()

    sig = infer_signature(X_train, y_train)

    mlflow.shap.logExplanation(model, X_train[:100])
```
Collaboration between Facebook and Databricks to bring ML platform features to PyTorch

- MLflow autologging for PyTorch Lightning
- TorchScript support for faster packaged models
- Model deployment to TorchServe
Many Open Questions in ML Platforms!

- How to design “feature stores” that can hold fast-changing data about entities to be used in training and inference?

- How to automatically detect performance degradation in models?

- What kind of information should be tracked at inference time?
Conclusion

Machine learning is being applied to critical problems in industry, but requires careful management when the stakes are high.

ML Platforms are an emerging abstraction to help with this.
- And we believe an “open-interface” design is very important for usability.

Many open problems, especially as needs evolve.