Machine learning lifecycle management with Acumos AI platform across multiple environments

17th July, 2019

Senior Researcher, Hitachi, Ltd.
Soichi Takashige

Engineer, Hitachi, Ltd.
Yusuke Furuyama
Today’s Agenda

1. Background: Machine Learning
2. Machine Learning MLOps Cycle
3. Acumos AI Platform
4. Our Assumed Use-case for on-prem ML Lifecycle
5. Evaluations on cutting-edge Acumos AI (Boreas release)
6. Confirmed Scenario
7. Technical Problems
8. Design Considerations
9. Conclusion
Our Profile

Soichi Takashige

• Senior researcher at Hitachi, Ltd.
• Development and management of platforms, including hypervisors, networks and cloud (OpenStack etc.)

Yusuke Furuyama

• Solutions engineer at Hitachi, Ltd.
• Leveraging OSS related to AI and bigdata for system integration.
1. Background: Machine Learning

- Machine learning technology is a core technology for the state of the art automation and control systems.
  - Various algorithms have been developed (ex. Deep Learning)
  - Lots of benefits from AI.
  - In Hitachi, the number of AI project is increasing

- It is widely used not only in the Web or IT area, but also spread into OT area (such as industry, city development, etc.)

https://www.hitachi.co.jp/products/it/bigdata/case/index.html
2. Machine Learning MLOps Cycle

- Machine learning model is developed, evaluated, and operated in DevOps manner (so called “MLOps”) in nature because it requires quick iterations of trial and error.

- Furthermore, we need MANY open source tools to develop models.

- Process are iterative, with many tools to operate. Too complex to handle workflow.

- We need supporting software to reduce painful iteration tasks of MLOps!
3-1. Acumos AI Platform

- OSS AI platform hosted by The Linux Foundation
- Makes it easy to build, share, deploy AI apps
  - Package tool kits (TensorFlow, scikit-learn) and models with a common API
  - Provide marketplace for sharing AI models internally within company and publicly
  - Container based easy deployment to both public cloud and private environment

3-2. Why Acumos AI Platform?

Official docs say Acumos AI seems to be useful to meet Enterprise and OT needs.

- **Acumos can run inside on-premise environment.**
  - AWS, Azure and GCP are great! But requires users to put data into public cloud.
  - Many OT users cannot ship their data, on-premise solution is necessary.

- **Acumos supports multiple machine learning libraries, scikit-learn, TensorFlow, ...**
  - Supporting only TensorFlow, or only scikit-learn is not enough.
  - We want general and standard solution to support multiple library.

- **Acumos can handle multi-tenant with authentication and access control.**
  - MLflow, and other tools can handle “single-user” mode for now.

- **Acumos doesn’t required any infrastructure skills for data scientists.**
  - In Kubeflow, data scientists should define CRD to run simple TensorFlow training job.
4-1. Our Assumed Use-case for on-prem ML Lifecycle

- We defined the generalized machine learning model lifecycle inside on-premise environment from several cases.
- We evaluated how our tasks would be supported by Acumos AI Platform.

**Characteristics**

A) Development and deployment environment is separated into different platform.
B) Development environment want to share expensive GPUs.
C) Registered model may be deployed and managed in several production environments

**1) Modeling**
- Data scientists
- Portal
- Workspace
- Data Lake
- Shared GPU

**2) Packaging**
- Model Repository
- Ship model
- Multiple Production Environments

**3) Deployment**
- Evaluation Env.
- Remote on-premise production
- Edge computing devices

**4) Monitoring**
- Log
- Log
- Log

A) Development and deployment environment is separated into different platform.
B) Development environment want to share expensive GPUs.
C) Registered model may be deployed and managed in several production environments.
4-2. Requirements for MLOps in on-premise env.

- We want to reduce the cost of painful iteration task of machine learning lifecycle.

**Pain points**
- Tons of interactive data manipulation with many different tools!
- Poor GPU / CPU resource at hand
- Tracking all training history to create model

**Requirements for MLOps**
- Prepare jupyter, and integrated SDKs for backend for every data scientists.
- Efficient resource sharing with training job scheduler
- Data and experiment management

**Pain points**
- Update inference pipelines and APIs according to change to the model.
- No way to detect and analyze quality problem on model.

**Requirements for MLOps**
- Serving models with automated pipeline and API generation
- Monitoring and tracking the behavior of the deployed model.
5-1. Managed workspace (e.g. Jupyter), and integrated SDK

- Acumos manages set of training resources and editors (like jupyter notebook) for every scientists by grouping them as “Project.”
- Data scientists don’t have to construct and maintain their environment.

1) Modeling

- Do modeling
- New in Boreas release

Portal

“Machine Learning Workbench”
In Design Studio

Launch and Connect

Projects

Projects

Jupyter notebook

Jupyter notebook

Launch and Connect
5-2. Efficient resource sharing with training job scheduler

- We’d like to run model training jobs on shared GPU resource pool.
- Acumos team will support NiFi to define and execute training pipelines in the Projects environment (*1). Currently we can’t test that functionality yet.

1) Modeling

Define jobs to run “model training” code.

Projects

NiFi environment

Shared resource pool

Spark / Kubernetes with GPUs.

Not yet?

(*1) [https://docs.acumos.org/en/latest/release-notes/boreas/release-boreas.html#id7](https://docs.acumos.org/en/latest/release-notes/boreas/release-boreas.html#id7)
5-3. Data and experiment management

- Tracking model training history is crucial in machine learning development.

- Currently, there seems no way to manage and track the parameters, data, and result of experiments using Acumos AI ML workbench.

- We’d like to have experiment management per “Project,” like jupyter notebook.
5-4-1. Serving models with automated pipeline and API generation

- Model is wrapped by executable platform binary, pipeline, and API endpoints to form microservice.

2) Packaging

Creating model wrapping codes, then creating zip archive.

On-boarding zip as “model”.

Downloading dockerized model solution deployment.
5-4-2. Serving models with automated pipeline and API generation

- Downloaded solution deployment package can be exported to other clusters.
- Deployment script automatically setup required environment on the target platforms.
  - (On-premise environment should be able to connect the development environment via network connection)

3) Deployment

- Exporting
- $ unzip solution.zip
- $ sudo deploy.sh
- Deploy on cluster.
- Loading Container
- Production Engineers
- Consuming application
- API
- Inference sub-system with model
- On-premise Kubernetes cluster

Shared dev. environment
5-5. Monitoring and tracking the behavior of the deployed model.

- Model performance is recorded in log, and collected by Beats and ELK stack. (*1)
- We can analyze the model performance from collected set of data.

4) Monitoring

On-premise Kubernetes cluster

Persistent Volume
- Log files

ELK stack
- Log files
- Log files
- Log files

Not yet confirmed

Inference sub-system with model

Logging

Consuming application

API
6. Confirmed Scenario

Requirements for MLOps

Prepare jupyter, and integrated SDKs for backend for every data scientists.

Efficient resource sharing with training job scheduler

Data and experiment management

Serving models with automated pipeline and API generation

Monitoring and tracking the performance and quality of the model

<table>
<thead>
<tr>
<th>Requirement</th>
<th>Available</th>
<th>Planned</th>
<th>Not available</th>
<th>Not yet evaluated</th>
</tr>
</thead>
<tbody>
<tr>
<td>Prepare jupyter, and integrated SDKs for backend for every data scientists.</td>
<td>✔️ Available</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Efficient resource sharing with training job scheduler</td>
<td>✔️ Available</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Data and experiment management</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Serving models with automated pipeline and API generation</td>
<td>✔️ Available</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Monitoring and tracking the performance and quality of the model</td>
<td>✔️ Available</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Confirmed Scenario

Users can prepare their own “Projects”, “Notebooks” and “Pipelines” from Machine Learning WorkBench.

Planned to support NiFi, but currently we can’t evaluate this upcoming new feature.

Currently no version management is planned.

Serving models with automated pipeline and API generation

Monitoring and tracking the performance and quality of the model
We tried cutting edge version of the release (system-integration HEAD+pull request 4374 /patchset 26 and 41 based). It will need some more time to stabilize the environment. It may be good to wait until upcoming patchset is committed to the tree. *(updated: patches are merged into the master branch on 12, Jul.)*

**At Installation:**
- acumos_k8s_prep.sh: experienced network corruption during k8s setup code.
- Components are installed via Helm charts, manifests, and shell scripts. Understanding all of them are headache.
- reinstallation or continue installation after fixing minor problem is not so easy task.
  - installer automatically erase all installed components beforehand.

**Runtime restrictions:**
- Some quick fixes are required to run on multi-node configurations.
  - /etc/hosts are rewritten manually. Some of them are not consistent.

**Architecture:**
- System configurations changed so often, because currently rearchitecting for Openshift, generic k8s, and docker environments.
  - example: API management layer, Kong(Athena) → Ingres(Boreas, for Kubernetes)
- It takes a lot of time to upgrade and validate the functionality of the deployments of different architecture. We’d better to watch and learn the difference of architectures for every releases and updates for a while.

We reported these problems to Acumos AI community and JIRA tickets were filed.
8-1. Design Considerations: Good points in design

• Concept and API design are well-designed, and may have the rooms for future extension.

Generating solution deployment package as a “zip with install scripts.” are convenient, due to its portability and extensibility to support many environments.

Concept of “Managing Projects and Resources” in Acumos AI is very convenient for data scientists to archive all related resources at once.
8-2. Design considerations: Expectations for extensibilities

We’d like to have some more flexibility to add features easily to Acumos AI Platform.

1) “Project extension” to add capability to manage extra components in “Project”
   - It is happy to have simple extension to add managed resource to the “Project”
   - Example resources are:
     - Experiment management service: (e.g. MLflow)
     - Data sets to track appropriate data sets for training machine learning models.

2) Easy way to specify whether jupyter requires “GPUs” or not.
   - We’d like to add some hints or guideline to specify the requirements for infrastructure. (e.g. “GPU” required. etc.)
   - Kubeflow can specify the full requirements by specifying manifest, but it feels “too much.” for daily use.

3) Additional “solution deployment” export plugin for non-k8s, and edges.
   - We’d like to standardize the workflow to on-board and export solution deployment package even if target environment is non-k8s.
   - Extra package generator other than “k8s” platform is a good option.
9. Conclusion

- We have evaluated the functionality of Acumos AI platform whether it satisfies the basic requirements for machine learning lifecycle within the on-premise environment.
- We concluded the capabilities and designed of Acumos AI platform as follows:

1) Acumos AI provides “MLOps” capability for data scientists and engineers to support lifecycle of machine learning models.

2) Acumos AI basically fits well with the requirements to develop and run machine learning models on on-premise environment.
   - Dedicated jupyter notebooks with tenant management called “Project”
   - Managing metadata of registered models with versions
   - Generating microservices from registered models for several deployment targets

3) Acumos AI also announces the support for several new functionalities in upcoming release or update in new future. It may be better to wait to evaluate until new update is published.
   - Data pipeline creation with workflow scheduler (Apache NiFi)
   - Logging data which are provided from generated microservices.

4) Acumos AI has clear and well-designed concept and APIs. Seems one of the promising platforms in AI area.
Trademarks

• The Linux Foundation® is registered trademark of The Linux Foundation
• Amazon Web Services, the “Powered by AWS” logo, and AWS are trademarks of Amazon.com, Inc. or its affiliates in the United States and/or other countries.
• Azure is either a registered trademark or a trademark of Microsoft Corporation in the United States and/or other countries.
• GCP is a trademark of Google Inc.
• Docker is a trademark or a registered trademark of Docker, Inc. in the United States and/or other countries.
• Kubernetes is a trademark or registered trademark of The Linux Foundation in the United States and/or other countries.
• Other brand names and product names used in this material are trademarks, registered trademarks, or trade names of their respective holders.