



SynapseML

非公式日本語解説(私見)

Microsoft
Cloud Solution Architect (Data&Analytics)
Keisuke Takahashi

Agenda

1. SynapseML とは
2. SynapseML の全体像
3. SynapseML の利用例
 - synapse.ml.opencv による画像変換
 - synapse.ml.featureize によるデータ加工
 - 回帰 (Linear Regressor • LightGBM • VW)
 - Cognitive Services を利用した英日テキスト翻訳
 - Geospatial Services を利用した Reverse geocoding
 - ONNX モデル (LightGBM) の利用
 - パイプラインの作成と実行
4. まとめ

Agenda

1. SynapseML とは
2. SynapseML の全体像
3. SynapseML の利用例
 - synapse.ml.opencv による画像変換
 - synapse.ml.featureize によるデータ加工
 - 回帰 (Linear Regressor · LightGBM · VW)
 - Cognitive Services を利用した英日テキスト翻訳
 - Geospatial Services を利用した Reverse geocoding
 - ONNX モデル (LightGBM) の利用
 - パイプラインの作成と実行
4. まとめ

What is SynapseML? (Referring to the official docs)

The screenshot shows the official SynapseML website. At the top, there's a navigation bar with links for "SynapseML", "Docs", "Blog", and "Videos". On the right side of the bar are dropdown menus for "0.9.5", "English", "Developer Docs", and search functions. Below the navigation bar is a sidebar with links to "Introduction", "Getting Started", "Features", "Transformers", "Estimators", and "Reference". To the right of the sidebar is a large central area. It features a blue starburst logo with the text "Version: 0.9.5" above it. Below the logo is the word "SynapseML" in a large, bold, white font. A detailed description follows: "SynapseML is an ecosystem of tools aimed towards expanding the distributed computing framework [Apache Spark](#) in several new directions. SynapseML adds many deep learning and data science tools to the Spark ecosystem, including seamless integration of Spark Machine Learning pipelines with [Microsoft Cognitive Toolkit \(CNTK\)](#), [LightGBM](#) and [OpenCV](#). These tools enable powerful and highly-scalable predictive and analytical models for a variety of datasources." Another paragraph describes how SynapseML brings networking capabilities to the Spark Ecosystem, allowing users to embed web services into their models. At the bottom of the main content area, it says "SynapseML requires Scala 2.12, Spark 3.0+, and Python 3.6+. See the API documentation [for Scala](#) and [for PySpark](#)". A prominent "Get Started" button is located at the bottom of the page.

0.9.5 ▾ A ⓘ English ▾ Developer Docs ▾

SynapseML Docs Blog Videos

Version: 0.9.5

Introduction

Getting Started >

Features >

Transformers >

Estimators >

Reference >

Examples

Explore our Features

Papers

SynapseML

SynapseML is an ecosystem of tools aimed towards expanding the distributed computing framework [Apache Spark](#) in several new directions. SynapseML adds many deep learning and data science tools to the Spark ecosystem, including seamless integration of Spark Machine Learning pipelines with [Microsoft Cognitive Toolkit \(CNTK\)](#), [LightGBM](#) and [OpenCV](#). These tools enable powerful and highly-scalable predictive and analytical models for a variety of datasources.

SynapseML also brings new networking capabilities to the Spark Ecosystem. With the HTTP on Spark project, users can embed **any** web service into their SparkML models. In this vein, SynapseML provides easy to use SparkML transformers for a wide variety of [Microsoft Cognitive Services](#). For production grade deployment, the Spark Serving project enables high throughput, sub-millisecond latency web services, backed by your Spark cluster.

SynapseML requires Scala 2.12, Spark 3.0+, and Python 3.6+. See the API documentation [for Scala](#) and [for PySpark](#).

Get Started

SynapseML とは (公式Docsより)

シナプスムラ ドキュメント ブログ ビデオ

0.9.5 ▾ Aあ 英語 ▾ 開発者ドキュメント ▾

検索

バージョン: 0.9.5

例
機能を確認する
書類

紹介

はじめ >
顔立ち >
トランスフォーマー >
推定 >
参考 >


シナプスムラ

SynapseML は、分散コンピューティング フレームワーク [Apache Spark](#) をいくつかの新しい方向に拡大することを目的としたツールのエコシステムです。SynapseML は、Spark の機械学習バイプラインとマイクロソフトコグニティブ ツールキット (CNTK) 、[LightGBM](#)、および [OpenCV](#) をシームレスに統合するなど、多くのディープラーニングとデータ サイエンス ツールを Spark エコシステムに追加します。これらのツールは、さまざまなデータソースに対して、強力で拡張性の高い予測モデルおよび分析モデルを実現します。

SynapseML は、スパークエコシステムにも新しいネットワーク機能をもたらします。HTTP on Spark プロジェクトを使用すると、ユーザーは任意の Web サービスを SparkML モデルに埋め込むことができます。この脈では、SynapseML は、さまざまなマイクロソフトのコグニティブ サービスに使用できる SparkML トランスフォーマーを提供します。運用グレードのデプロイでは、Spark サービス プロジェクトにより、Spark クラスターに裏打ちされた高スループットのサブミリ秒待機時間 Web サービスが有効になります。

シナプスMLは、スカラ2.12、スパーク3.0+、およびPython 3.6+を必要とします。[スカラ](#)と[PySpark](#)のAPIドキュメントを参照してください。

What is SynapseML? (Referring to the official blog)

The screenshot shows a dark-themed web page for the SynapseML blog. At the top, there's a navigation bar with links for 'SynapseML', 'Docs', 'Blog' (which is highlighted in blue), and 'Videos'. On the right side of the nav bar are links for '0.9.5', 'English', 'Developer Docs', and several icons for search and refresh.

Recent posts

- [Overview](#)
- [Publication - Large-Scale Intelligent Microservices](#)
- [MMLSpark: empowering AI for Good with Mark Hamilton](#)
- [Dear Spark developers: Welcome to Azure Cognitive Services](#)
- [Publication - MMLSpark: Unifying Machine Learning Ecosystems at Massive Scales](#)

Overview

February 2, 2022 · One min read

Synapse Machine Learning expands the distributed computing framework [Apache Spark](#) in several new directions. SynapseML adds several machine learning frameworks to the SparkML Ecosystem, including [LightGBM](#), [Vowpal Wabbit](#), [OpenCV](#), [Isolation Forest](#), and the [Microsoft Cognitive Toolkit \(CNTK\)](#). These tools allow users to craft powerful and highly-scalable models that span multiple ML ecosystems.

SynapseML also brings new networking capabilities to the Spark ecosystem. With the [HTTP on Spark](#) project, users can embed any web service into their SparkML models and use their Spark clusters for massive networking workflows. In this vein, SynapseML provides easy to use SparkML transformers for a wide variety of Microsoft Cognitive Services. Finally, the [Spark Serving](#) project enables high throughput, sub-millisecond latency web services, backed by your Spark cluster.

Visit the [SynapseML Github repository](#) to learn more.

[Older Post](#)
[Publication - Large-Scale Intelligent Microservices » »](#)

SynapseML とは (公式ブログより)

The screenshot shows a web browser displaying a blog post from the SynapseML website. The page has a dark background with white text. At the top, there is a navigation bar with links for 'シナプスマル' (SynapseML), 'ドキュメント' (Documentation), 'ブログ' (Blog), and 'ビデオ' (Videos). On the right side of the header are buttons for version '0.9.5', language 'Aあ 英語', developer documentation, and user settings. Below the header, there is a sidebar titled '最近の投稿' (Recent Posts) containing links to other blog posts. The main content area features a large heading '概要' (Overview) and a sub-headline '2022年2月2日 · 1分読み取り'. The main text discusses the expansion of SynapseML's machine learning capabilities by adding various ML frameworks like LightGBM, OpenCV, and CNTK to the SparkML ecosystem. It also mentions the addition of a new network function called 'HTTP on Spark' and the integration of Web services with Spark clusters. A note at the bottom encourages reading the SynapseML GitHub repository for more details. At the bottom right, there is a box with a link to another blog post and a '古い投稿' (Older posts) button.

シナプスマル ドキュメント ブログ ビデオ

0.9.5 ▾ Aあ 英語 ▾ 開発者ドキュメント ▾

概要

2022年2月2日 · 1分読み取り

シナプスマルは、いくつかの新しい方向に分散コンピューティングフレームワークApache Sparkを拡張します。SynapseMLは、LightGBM、母音ワビット、OpenCV、隔離フォレスト、マイクロソフトコグニティブツールキット(CNTK)など、SparkMLエコシステムに複数の機械学習フレームワークを追加します。これらのツールを使用すると、複数のMLエコシステムにまたがる強力で拡張性の高いモデルを作成できます。

SynapseMLはSparkエコシステムにも新しいネットワーク機能をもたらします。HTTP on Sparkプロジェクトでは、ユーザーは任意のWebサービスをSparkMLモデルに組み込み、Sparkクラスターを使用して大規模なネットワーキングワークフローを実行できます。この静脈では、SynapseMLは、さまざまなマイクロソフトのコグニティブサービスに使用できるSparkMLトランسفォーマーを提供します。最後に、Sparkサービスプロジェクトは、Sparkクラスターに裏打ちされた、高スループットのサブミリ秒待機時間のWebサービスを有効にします。

詳細については、[SynapseML Github リポジトリ](#)をご覧ください。

古い投稿
出版物 - 大規模インテリジェントマイクロサービス » »

What is SynapseML? (Referring to GitHub)

The screenshot shows a GitHub repository page for "Synapse Machine Learning". At the top, there's a header with a file icon, "275 lines (187 sloc) | 14.5 KB", and a row of buttons: "Raw", "Blame", "Copy", "Edit", and "Delete". Below the header is a large blue logo consisting of three stylized, overlapping triangles pointing upwards. The main title "Synapse Machine Learning" is centered above a horizontal line. Below the line are several status indicators: "Azure Pipelines succeeded", "codecov 85%", "chat on gitter", "release notes", "api docs scala", "api docs python", "academic paper", and "version 0.9.5 master version 0.9.5-17-9af38da0-SNAPSHOT". A detailed description follows: "SynapseML (previously MMLSpark) is an open source library to simplify the creation of scalable machine learning pipelines. SynapseML builds on [Apache Spark](#) and SparkML to enable new kinds of machine learning, analytics, and model deployment workflows. SynapseML adds many deep learning and data science tools to the Spark ecosystem, including seamless integration of Spark Machine Learning pipelines with the [Open Neural Network Exchange \(ONNX\)](#), [LightGBM](#), [The Cognitive Services](#), [Vowpal Wabbit](#), and [OpenCV](#). These tools enable powerful and highly-scalable predictive and analytical models for a variety of datasources." Another paragraph below states: "SynapseML also brings new networking capabilities to the Spark Ecosystem. With the HTTP on Spark project, users can embed **any** web service into their SparkML models. For production grade deployment, the Spark Serving project enables high throughput, sub-millisecond latency web services, backed by your Spark cluster." On the far left edge of the browser window, there's a vertical sidebar with the text "Octotree >".

Octotree >

275 lines (187 sloc) | 14.5 KB

Azure Pipelines succeeded codecov 85% chat on gitter

release notes api docs scala api docs python academic paper

version 0.9.5 master version 0.9.5-17-9af38da0-SNAPSHOT

SynapseML (previously MMLSpark) is an open source library to simplify the creation of scalable machine learning pipelines. SynapseML builds on [Apache Spark](#) and SparkML to enable new kinds of machine learning, analytics, and model deployment workflows. SynapseML adds many deep learning and data science tools to the Spark ecosystem, including seamless integration of Spark Machine Learning pipelines with the [Open Neural Network Exchange \(ONNX\)](#), [LightGBM](#), [The Cognitive Services](#), [Vowpal Wabbit](#), and [OpenCV](#). These tools enable powerful and highly-scalable predictive and analytical models for a variety of datasources.

SynapseML also brings new networking capabilities to the Spark Ecosystem. With the HTTP on Spark project, users can embed **any** web service into their SparkML models. For production grade deployment, the Spark Serving project enables high throughput, sub-millisecond latency web services, backed by your Spark cluster.

SynapseML とは (GitHub より)

The screenshot shows the GitHub repository page for SynapseML. At the top, there's a header with a file icon, the number 275, the text '行 (187 スロック) 14.5 KB', and a download button. Below the header is a toolbar with various icons for file operations. The main content area features the SynapseML logo (a blue geometric shape resembling a stylized flower or star), the title 'シナプス機械学習' (Synapse Machine Learning), and several status indicators: 'Azure Pipelines succeeded', 'codecov 85%', 'chat on gitter', 'release notes', 'api docs scala', 'api docs python', 'academic paper', 'version 0.9.5', and 'master version 0.9.5-17-9af38da0-SNAPSHOT'. Below these indicators is a detailed description of the project's purpose and capabilities, mentioning its integration with Apache Spark, various machine learning models, and its use in a Spark ecosystem. The bottom of the page includes a sidebar with navigation icons.

SynapseML (以前は MMLSpark) は、スケーラブルな機械学習パイプラインの作成を簡素化するオープンソースライブラリです。SynapseML は、新しい種類の機械学習、分析、およびモデル展開ワークフローを可能にするために、Apache Spark と SparkML をベースに構築します。SynapseML は、Spark 機械学習パイプラインとオープンニューラル ネットワーク エクスチェンジ (ONNX)、LightGBM、コグニティブ サービス、Vowpal Wabbit、および OpenCV をシームレスに統合するなど、多くのディープラーニングとデータサイエンスツールを Spark エコシステムに追加します。これらのツールは、さまざまなデータソースに対して、強力で拡張性の高い予測モデルおよび分析モデルを実現します。

SynapseML は、スパークエコシステムにも新しいネットワーク機能をもたらします。HTTP on Spark プロジェクトを使用すると、ユーザーは任意の Web サービスを SparkML モデルに埋め込むことができます。運用グレードのデプロイでは、Spark サービス プロジェクトにより、Spark クラスターに裏打ちされた高スループットのサブミリ秒待機時間 Web サービスが有効になります。

つまり SynapseML とは

- 旧名称 MMLSpark
- Apache Spark を拡張するツールのエコシステム
 - 機械学習フレームワークを追加
 - 複数のMLシステムをまたがるモデルの作成が可能
- パイプラインを簡素化可能
- Webサービスとの統合が可能
 - 任意のWebサービスを SparkML モデルに埋め込むことができる
- オープンソース ライブラリ
- Spark 3.2+, Scala 2.12, Python 3.6+ が必要 (GitHubより)

つまり SynapseML とは

- ・旧名称 MMLSpark
- ・Apache Spark を拡張するツールのエコシステム
 - ・機械学習フレームワークを追加
 - ・複数のMLシステムをまたがるモデルの作成が可能
- ・パイプラインを簡素化可能
- ・Webサービスとの統合が可能
 - ・任意のWebサービスを SparkML モデルに埋め込むことができる
- ・オープンソース ライブラリ
- ・Spark 3.2+, Scala 2.12, Python 3.6+ が必要 (GitHubより)

Apache Spark を拡張するツールのエコシステム

- 機械学習フレームワーク（および関連するツールやアルゴリズム）の追加
 - Vowpal Wabbit
 - LightGBM
 - OpenCV
 - CNTK
 - Isolation Forest
- 複数のMLシステムをまたがるモデルの作成が可能
 - ONNX

パイプラインを簡素化可能

The screenshot shows a dark-themed web page for the **SynapseML** documentation. On the left, a sidebar lists various categories: Introduction, Getting Started, Features (with Cognitive Services, Geospatial Services, Responsible AI, ONNX, LightGBM, Vowpal Wabbit, About, and Vowpal Wabbit - Overview), Spark Serving, OpenCV, Classification, Regression, Other, and Transformers. The **Vowpal Wabbit** section is currently selected. The main content area displays a code snippet for transforming data using the Vowpal Wabbit Featurizer:

```
data.printSchema()  
  
Add pipeline to add featurizer, convert all feature columns into vector.  
  
from synapse.ml.vw import VowpalWabbitFeaturizer, VowpalWabbitContextualBandit, VectorZipper  
from pyspark.ml import Pipeline  
pipeline = Pipeline(stages=[  
    VowpalWabbitFeaturizer(inputCols=['GUser_id'], outputCol='GUser_id_feature'),  
    VowpalWabbitFeaturizer(inputCols=['GUser_major'], outputCol='GUser_major_feature'),  
    VowpalWabbitFeaturizer(inputCols=['GUser_hobby'], outputCol='GUser_hobby_feature'),  
    VowpalWabbitFeaturizer(inputCols=['GUser_favorite_character'], outputCol='GUser_favorite_chara  
    VowpalWabbitFeaturizer(inputCols=['TAction_0_topic'], outputCol='TAction_0_topic_feature'),  
    VowpalWabbitFeaturizer(inputCols=['TAction_1_topic'], outputCol='TAction_1_topic_feature'),  
    VowpalWabbitFeaturizer(inputCols=['TAction_2_topic'], outputCol='TAction_2_topic_feature'),  
    VowpalWabbitFeaturizer(inputCols=['TAction_3_topic'], outputCol='TAction_3_topic_feature'),  
    VowpalWabbitFeaturizer(inputCols=['TAction_4_topic'], outputCol='TAction_4_topic_feature'),  
    VectorZipper(inputCols=['TAction_0_topic_feature', 'TAction_1_topic_feature', 'TAction_2_topic  
])  
tranformation_pipeline = pipeline.fit(data)  
transformed_data = tranformation_pipeline.transform(data)  
  
display(transformed_data)
```

Below the code, a note states: "Build VowpalWabbit Contextual Bandit model and compute performance statistics."

The top right of the page includes a version dropdown (0.9.5), language dropdown (English), developer documentation link, and search bar.

On the right side, there is a sidebar with links to other Vowpal Wabbit examples:

- Advantages of VowpalWabbit
- Limitations of VowpalWabbit on Spark
- VowpalWabbit Usage:
- Heart Disease Detection with VowalWabbit Classifier
- Adult Census with VowpalWabbitClassifier
- Boston house price prediction with VowpalWabbitRegressor - Quantile Regression
- Quantile Regression for Drug Discovery with VowpalWabbitRegressor
- VW Contextual Bandit

Webサービスとの統合が可能

The screenshot shows a dark-themed web page for SynapseML. On the left is a sidebar with navigation links: Introduction, Getting Started, Features, Cognitive Services, Geospatial Services, Responsible AI, ONNX, LightGBM, Vowpal Wabbit, About, Vowpal Wabbit - Overview, Spark Serving (which is expanded to show About, SparkServing - Deploying a Classifier, OpenCV, Classification, and Regression), and a few other collapsed sections like Features and Cognitive Services.

The main content area has a title "Deploying a Deep Network with the CNTKModel". Below it is a code block:

```
import synapse.ml
from synapse.ml.cntk import CNTKModel
import pyspark
from pyspark.sql.functions import udf, col

df = spark.readStream.server() \
    .address("localhost", 8888, "my_api") \
    .load() \
    .parseRequest(<Insert your models input schema here>)

# See notebook examples for how to create and save several
# examples of CNTK models
network = CNTKModel.load("file:///path/to/my_cntkmodel.mml")

transformed_df = network.transform(df).makeReply(<Whatever column you wish to send back>)

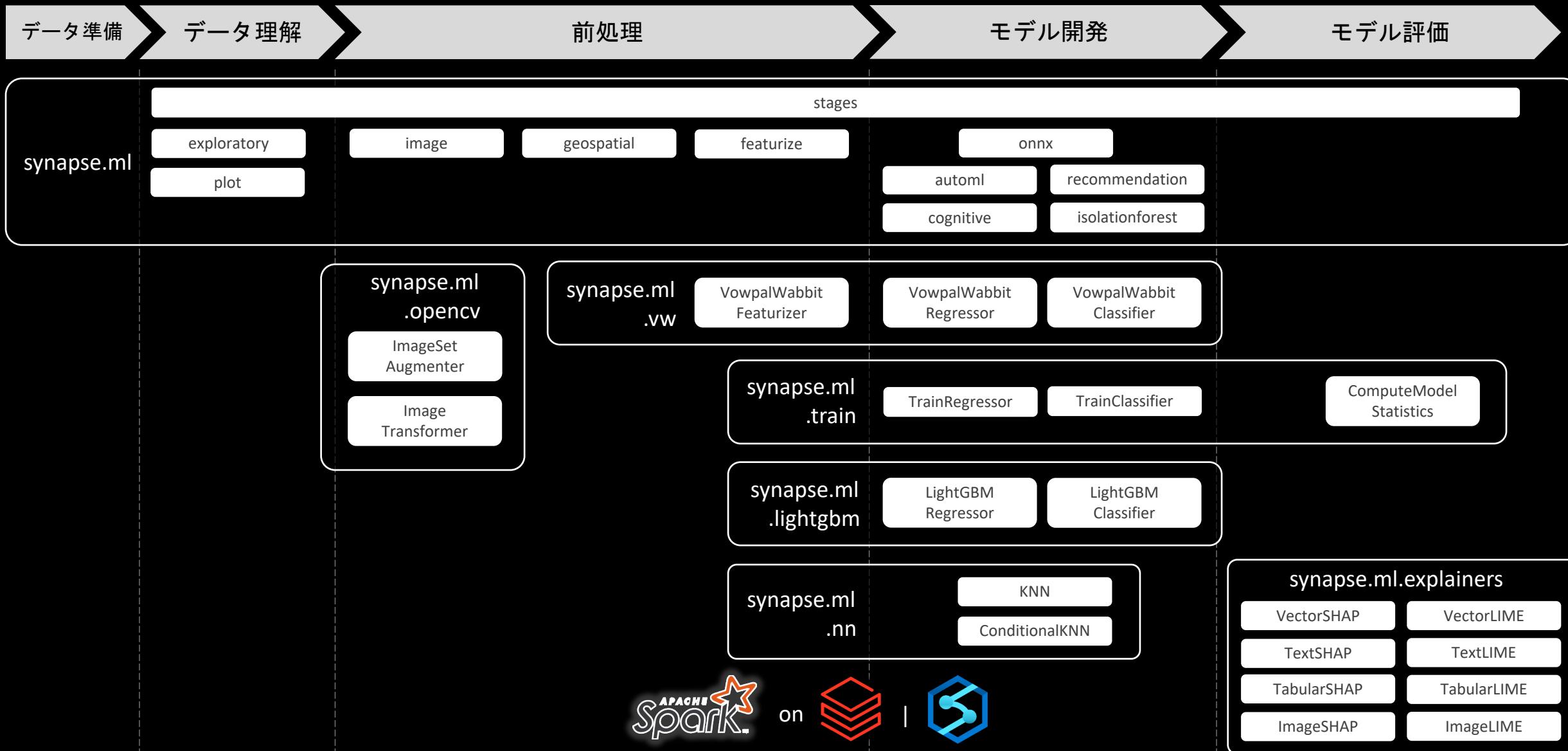
server = transformed_df \
    .writeStream \
    .server() \
    .replyTo("my_api") \
    .queryName("my_query") \
    .option("checkpointLocation", "file:///path/to/checkpoints") \
    .start()
```

To the right of the main content are several sections: "An Engine for Deploying Spark Jobs as Distributed Web Services", "Usage" (with links to Jupyter Notebook Examples and Spark Serving Hello World), "Deploying a Deep Network with the CNTKModel" (which is currently highlighted), "Architecture" (with links to Head Node Load Balanced, Fully Distributed (Custom Load Balancer), and Sub-Millisecond Latency with Continuous Processing), and "Parameters".

Agenda

1. SynapseML とは
2. SynapseML の全体像
3. SynapseML の利用例
 - synapse.ml.opencv による画像変換
 - synapse.ml.featureize によるデータ加工
 - 回帰 (Linear Regressor · LightGBM · VW)
 - Cognitive Services を利用した英日テキスト翻訳
 - Geospatial Services を利用した Reverse geocoding
 - ONNX モデル (LightGBM) の利用
 - パイプラインの作成と実行
4. まとめ

SynapseML の全体像



Agenda

1. SynapseML とは
2. SynapseML の全体像
3. SynapseML の利用例
 - synapse.ml.opencv による画像変換
 - synapse.ml.featureize によるデータ加工
 - 回帰 (Linear Regressor • LightGBM • VW)
 - Cognitive Services を利用した英日テキスト翻訳
 - Geospatial Services を利用した Reverse geocoding
 - ONNX モデル (LightGBM) の利用
 - パイプラインの作成と実行
4. まとめ

synapse.ml.opencv による画像変換

シナリオ

1. Spark に画像データを読み込む
2. 画像を小さくリサイズしてクロップ
3. 画像を多次元配列に変換
4. 多次元配列を Spark の Row オブジェクトに変換
5. Spark の UDF として上記変換を実行

```
from synapse.mlopencv import ImageTransformer

imageDir = "wasbs://publicwasb@mmlspark.blob.core.windows.net/sampleImages"
images = spark.read.image().load(imageDir).cache()

tr = (ImageTransformer()
      .setOutputCol("transformed")
      .resize(size=(200, 200))
      .crop(0, 0, height = 180, width = 180) )

small = tr.transform(images).select("transformed")
```

synapse.ml.featureize によるデータ加工

シナリオ

1. Spark にテーブルデータを読み込む
2. データを学習用とテスト用に分ける
3. 各データの型変換を行う
4. モデルを作成する
5. モデルをテストする

```
flightDelay =  
spark.read.parquet("wasbs://publicwasb@mmlspark.blob.core.windows.net/On_Time_Performance_2012_9.parquet")  
  
from synapse.ml.feature import DataConversion  
flightDelay = DataConversion(cols=["Quarter", "Month", "DayofMonth", "DayOfWeek",  
                                     "OriginAirportID", "DestAirportID",  
                                     "CRSDepTime", "CRSArrTime"],  
                           convertTo="double") \  
.transform(flightDelay)  
  
train, test = flightDelay.randomSplit([0.75, 0.25])
```

synapse.ml.featureaize によるデータ加工

シナリオ

1. Spark にテーブルデータを読み込む
2. データを学習用とテスト用に分ける
3. 各データの型変換を行う
4. モデルを作成する
5. モデルをテストする

```
from synapse.ml.train import TrainRegressor, TrainedRegressorModel
from pyspark.ml.regression import

trainCat = DataConversion(cols=["Carrier", "DeptTimeBlk", "ArrTimeBlk"],
                           convertTo="toCategorical").transform(train)
testCat = DataConversion(cols=["Carrier", "DeptTimeBlk", "ArrTimeBlk"],
                           convertTo="toCategorical").transform(test)
lr = LinearRegression().setRegParam(0.1).setElasticNetParam(0.3)
model = TrainRegressor(model=lr, labelCol="ArrDelay").fit(trainCat)

scoredData = model.transform(testCat)
```

回帰(共通)

シナリオ

1. データを用意し、Spark の Dataframe に読み込む
2. データを学習用とテスト用に分ける

```
from sklearn.datasets import load_boston  
  
boston = load_boston()  
  
feature_cols = ['f' + str(i) for i in range(boston.data.shape[1])]  
header = ['target'] + feature_cols  
df = spark.createDataFrame(pd.DataFrame(data=np.column_stack((boston.target, boston.data))),  
columns=header)).repartition(1)  
  
train_data, test_data = df.randomSplit([0.75, 0.25], seed=42)
```

回帰 (Linear Regressor)

シナリオ

3. 用意したデータを Linear Regressor 用のフォーマットに変換する
4. 線形回帰用のクラスをインスタンス化する
5. 学習を行う
6. テストを行う

```
from pyspark.ml.feature import VectorAssembler
from pyspark.ml.regression import

featurizer = VectorAssembler(inputCols=feature_cols, outputCol='features')
lr_train_data = featurizer.transform(train_data)['target', 'features']
lr_test_data = featurizer.transform(test_data)['target', 'features']

lr = LinearRegression(labelCol='target')

lr_model = lr.fit(lr_train_data)
lr_predictions = lr_model.transform(lr_test_data)
```

回帰 (LightGBM)

シナリオ

3. LightGBM 用の回帰クラスをインスタンス化する
4. 学習を行う
5. テストを行う

```
from synapse.ml.lightgbm import LightGBMRegressor

lgr = LightGBMRegressor(
    objective='quantile',
    alpha=0.2,
    learningRate=0.3,
    numLeaves=31,
    labelCol='target',
    numIterations=100)

repartitioned_data = lr_train_data.repartition(1).cache()
lg_model = lgr.fit(repartitioned_data)
lg_predictions = lg_model.transform(lr_test_data)
```

回帰 (Vowpal Wabbit)

シナリオ

3. 用意したデータを Vowpal Wabbit 用のフォーマットに変換する
4. vw 用の回帰クラスをインスタンス化する
5. 学習を行う
6. テストを行う

```
from synapse.ml.vw import VowpalWabbitRegressor, VowpalWabbitFeaturizer

vw_featurizer = VowpalWabbitFeaturizer(inputCols=feature_cols, outputCol='features')

vw_train_data = vw_featurizer.transform(train_data)['target', 'features']
vw_train_data = vw_train_data.repartition(1).cache().repartition(1)
vw_test_data = vw_featurizer.transform(test_data)['target', 'features']

args = --holdout_off --loss_function quantile -l 7 -q :: --power_t 0.3
vwr = VowpalWabbitRegressor(labelCol='target', args=args, numPasses=100)

vw_model = vwr.fit(vw_train_data)
vw_predictions = vw_model.transform(vw_test_data)
```

Cognitive Services を利用した英日テキスト翻訳

シナリオ

1. APIキーを用意する
2. 入力データを Spark Dataframe 形式で用意する
3. Translate クラスのインスタンスを作成する
4. Translate オブジェクトのメソッドをコールして翻訳の実行と返却データの整形を行う

```
from synapse.ml.cognitive import Translate
from pyspark.sql.functions import col, flatten

translator_key = os.environ["TRANSLATOR_KEY"]

df = spark.createDataFrame(["Hello, what is your name?", "Bye"], ["text",])

translate = (Translate()
    .setSubscriptionKey(translator_key)
    .setLocation("eastus")
    .setTextCol("text")
    .setToLanguage(["ja"])
    .setOutputCol("translation"))
```

Cognitive Services を利用した英日テキスト翻訳

シナリオ

1. APIキーを用意する
2. 入力データを Spark Dataframe 形式で用意する
3. Translate クラスのインスタンスを作成する
4. Translate オブジェクトのメソッドをコールして翻訳の実行と返却データの整形を行う

```
display(translate
  .transform(df)
  .withColumn("translation", flatten(col("translation.translations")))
  .withColumn("translation", col("translation.text"))
  .select("translation"))
```

Geospatial Services を利用した Reverse geocoding

シナリオ

1. APIキーを用意する
2. 入力データを Spark Dataframe 形式で用意する
3. ReverseAddressGeocoder クラスのインスタンスを作成する
4. ReverseAddressGeocoder オブジェクトのメソッドをコールして住所取得の実行と返却データの整形を行う

```
from pyspark.sql.functions import col
from pyspark.sql.types import StructType,StructField, DoubleType
from synapse.ml.cognitive import *
from synapse.ml.geospatial import *

azureMapsKey = os.environ["AZURE_MAPS_KEY"]

df = spark.createDataFrame([
    48.858561  2.294911,
    47.639765  -122.127896,
    47.621028  -122.348170,
    47.734012  -122.102737
]), StructType([StructField("lat", DoubleType()), StructField("lon", DoubleType())]))
```

Geospatial Services を利用した Reverse geocoding

シナリオ

1. APIキーを用意する
2. 入力データを Spark Dataframe 形式で用意する
3. ReverseAddressGeocoder クラスのインスタンスを作成する
4. ReverseAddressGeocoder オブジェクトのメソッドをコールして住所取得の実行と返却データの整形を行う

```
rev_geocoder = (ReverseAddressGeocoder()
    .setSubscriptionKey(azureMapsKey)
    .setLatitudeCol("lat")
    .setLongitudeCol("lon")
    .setOutputCol("output"))

display(rev_geocoder.transform(FixedMiniBatchTransformer().setBatchSize(10).transform(df)).select(col("*"),
    col("output.response.addresses").getItem(0).getField("address").getField("freeformAddress")
    .alias("In Polygon"),
    col("output.response.addresses").getItem(0).getField("address").getField("country").alias
    ("Intersecting Polygons")
    ).drop("output"))
```

ONNX モデル (LightGBM) の利用

シナリオ

1. モデルとそのペイロードを用意する (ペイロードは onnxmltools.convert.convert_lightgbm を利用して取得)
2. ONNXModel クラスのインスタンスを作成し、ペイロードをセットする
3. モデルに入力する Dataframe のカラム名と、モデルから出力する Dataframe のカラム名を、それぞれセットする
4. Spark Dataframe を使用して推論を実行する

```
from synapse.ml.onnx import ONNXModel

onnx_ml = ONNXModel().setModelPayload(model_payload_ml)

onnx_ml = (
    onnx_ml
        .setDeviceType("CPU")
        .setFeedDict({"input": "features"})
        .setFetchDict({"probability": "probabilities", "prediction": "label"})
        .setMiniBatchSize(5000)
)
onnx_ml.transform(testDf)
```

パイプラインの作成と実行

シナリオ

1. 予め学習用・テスト用・バリデーション用のデータを用意しておく
2. パイプラインで実行したいインスタンスを作成する
3. パイプラインを作成する
4. パイプラインを使ってモデルの学習を行う
5. モデルを使って各種推論を行う

```
from pyspark.ml import Pipeline
from pyspark.ml.feature import Tokenizer, Word2Vec

tokenizer = Tokenizer(inputCol="text", outputCol="words")
partitions = train.rdd.getNumPartitions()
word2vec = Word2Vec(maxIter=4, seed=42, inputCol="words", outputCol="features",
                    numPartitions=partitions)

textFeaturizer = Pipeline(stages = [tokenizer, word2vec]).fit(train)

ptrain = textFeaturizer.transform(train).select(["label", "features"])
ptest = textFeaturizer.transform(test).select(["label", "features"])
pvalidation = textFeaturizer.transform(validation).select(["label", "features"])
```

Agenda

1. SynapseML とは
2. SynapseML の全体像
3. SynapseML の利用例
 - synapse.ml.opencv による画像変換
 - synapse.ml.featureize によるデータ加工
 - 回帰 (Linear Regressor · LightGBM · VW)
 - Cognitive Services を利用した英日テキスト翻訳
 - Geospatial Services を利用した Reverse geocoding
 - ONNX モデル (LightGBM) の利用
 - パイプラインの作成と実行
4. まとめ

まとめ

- SynapseMLについて、個人的な調査結果をご紹介しました。
- Azure Synapse Analytics または Azure Databricks 上で SynapseML を使用することで、Spark ML よりも広い範囲の機械学習プロセスを、单一のプラットフォーム・単一のライブラリで実現することができます。
- また、Azure Cognitive Servicesへのアクセシビリティも提供されているため、AI開発をより加速させることができます。
- SynapseMLはオープンソースソフトウェアです。今回深掘りしていない機能については、ソースやドキュメントを探索してみてください。

A large black silhouette of a hand reaches out from the bottom left towards a crowd of stylized human figures. The figures are colored in various shades of blue, purple, and orange. One figure in the center is highlighted in white, standing out from the crowd. The background is a dark, textured surface.

Thank You!