HIGH-LEVEL APIS IN BIGDL

Module 6

Learning Objectives

You will be able to:

- Understand high-level APIs in BigDL
- Learn to use these APIs

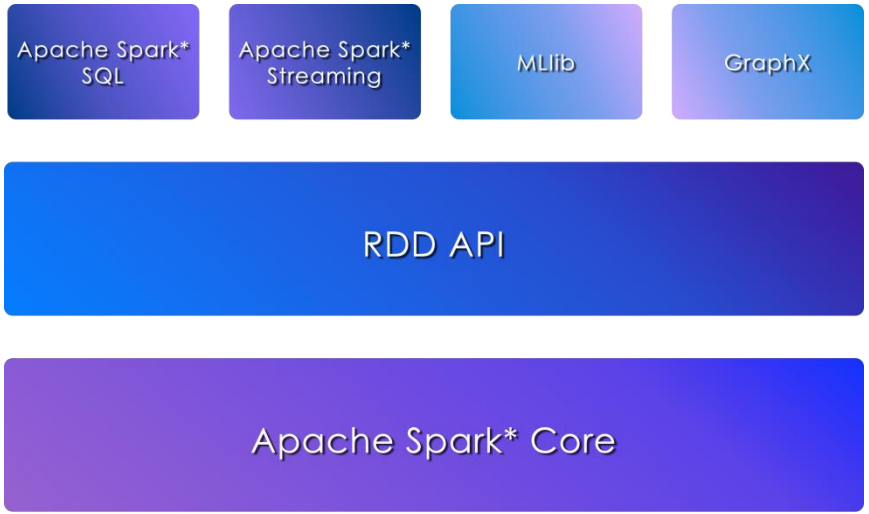
APACHE SPARK* DATAFRAME API

Apache Spark* API Evolution

In Spark* v1.0 primary data abstraction was provided by RDD (Resilient Distributed Dataset) API

RDDs are

- Stable : they have been in use since early days of Spark*
- Supported by many applications



*Other names and brands may be claimed as the property of others.

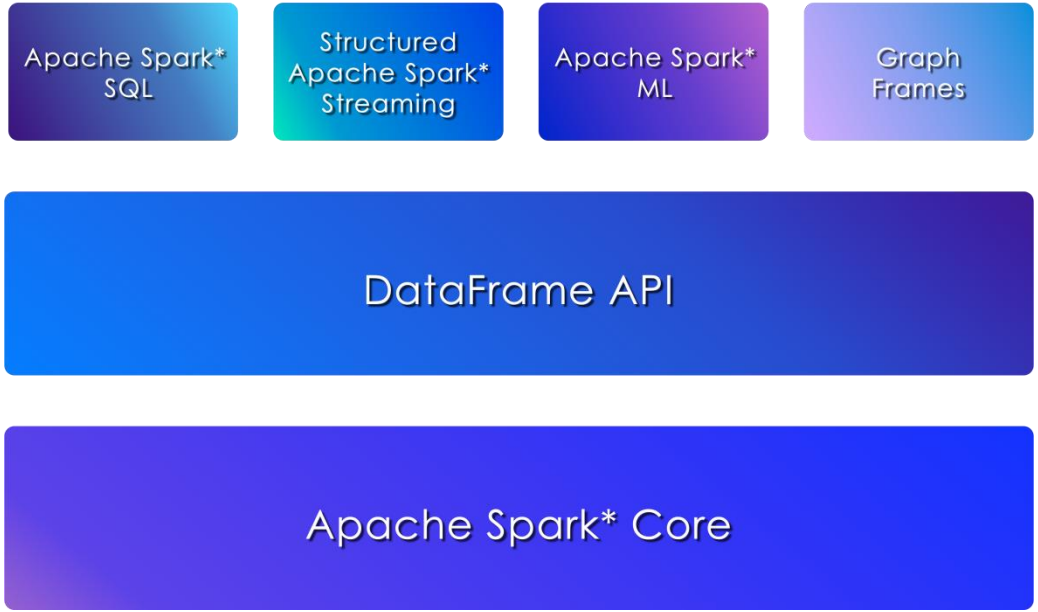
DataFrames in Apache Spark* v2

In Spark* v2, new API called Dataframe and DataSet are introduced

DataFrame Features

- Provide high-level APIs
- Easy to use
- Provides better performance via various optimizations (better memory management, optimizing code on the fly ..etc.)

Other Spark* components (SQL, ML, Streaming) are now migrating to the new API



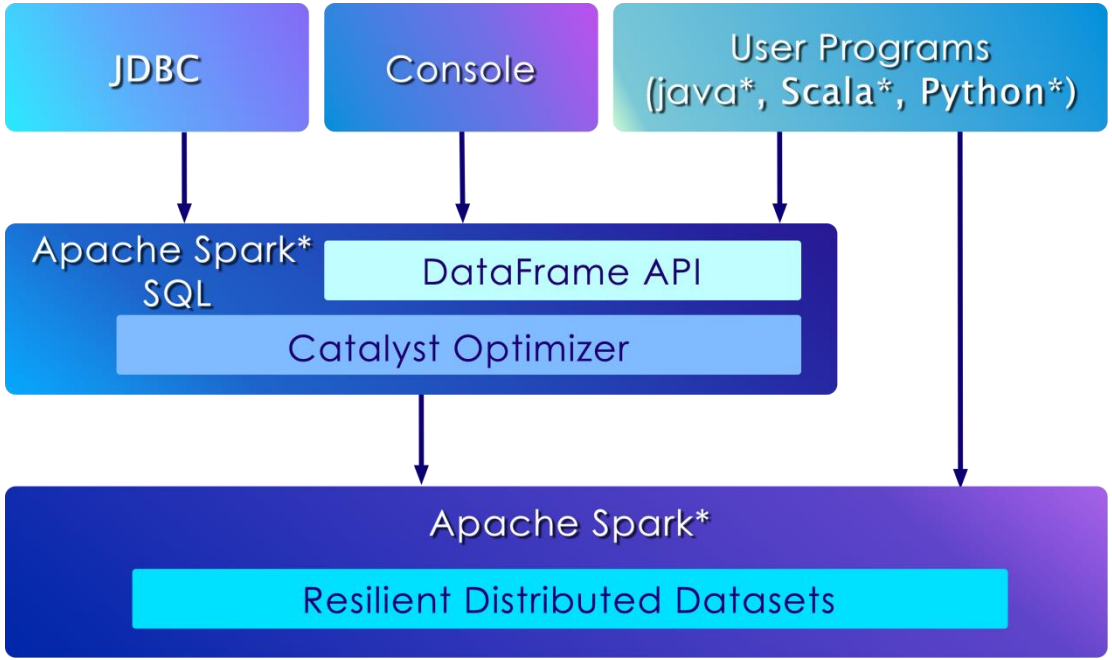
*Other names and brands may be claimed as the property of others.

DataFrame Architecture

Dataframes are part of 'Spark SQL*' package

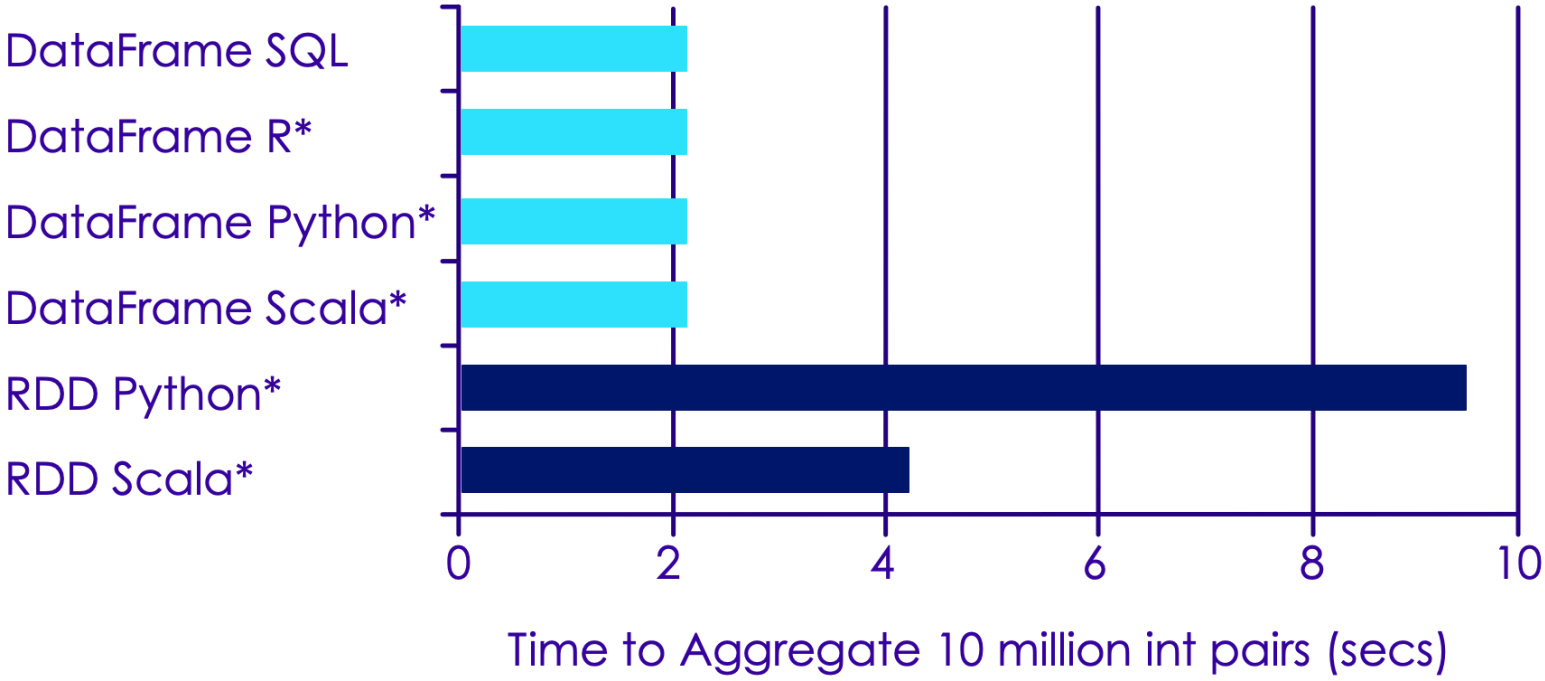
Catalyst Optimizer will generate optimized code for runtime

So typically, programming using DataFrame API yields in better, faster code than programming in RDD API



*Other names and brands may be claimed as the property of others.

DataFrame Performance



Comparing Apache Spark* Data Structures

RDD	DataFrame	Dataset
• Since v1 • Original data API • Gives complete control to developer • Can be a bit low level	• Since Spark* v2 (experimental from 1.3) • Created to give high-level data access • Data has schema, organized as columns • Supports SQL • DataFrame = RDD + Schema • Catalyst query optimizer	• Since Spark* v2 • Effort to unify RDD and DataFrame
Java*, Python*, Scala*	Java*, Python*, Scala*	Java*, Scala*, Python* (partial support)

*Other names and brands may be claimed as the property of others.

ANALYTICS ZOO

Analytics Zoo

Analytics Zoo provides a unified analytics + AI platform

It brings together Apache Spark*, TensorFlow*, Keras* and BigDL

Supports 'pipeline' style programming (popular in Scikit* and Spark)

Data wrangling and analysis using PySpark*

Deep learning model development using TensorFlow* or Keras*

Provides built-in deep learning models (more on this later)

*Other names and brands may be claimed as the property of others.

Analytics Zoo

Standard Spark* jobs

No changes to the Apache Spark* or Apache Hadoop* clusters needed

Iterative

Each iteration of the training runs as a Spark* job

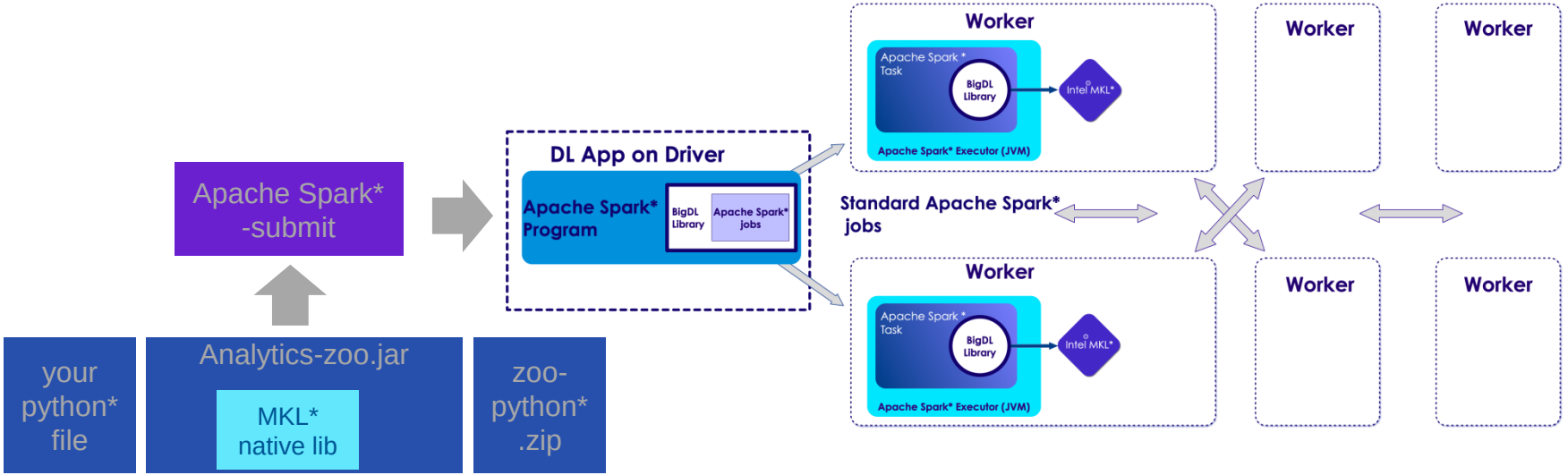
Data parallel

Each Spark* task runs the same model on a subset of the data (batch)

(see next slide for diagram)

*Other names and brands may be claimed as the property of others.

Analytics Zoo & Apache Spark*



*Other names and brands may be claimed as the property of others.

Getting Started with Analytics Zoo – Python*

There are 2 ways of getting Analytics Zoo going with Python*

Option 1 : Using PIP

Option 2 : Manual install

(see next slides for more details)

*Other names and brands may be claimed as the property of others.

Analytics Zoo Install for Python* – Option 1 : Using PIP

```
## 6.1 - Analytics Zoo install with PIP
pip install --upgrade pip
# change the zoo version accordingly
pip install analytics-zoo==0.2.0     # for Python 2.7
pip3 install analytics-zoo==0.2.0    # for Python 3.5 and Python 3.6
```

This is tested with PIP v9.0.1

Pip install only supports local mode

This method also installs PySpark*

For up to date instructions see here *Other names and brands may be claimed as the property of others.

Analytics Zoo Install for Python* – Option 2 : Manual Install (1 of 2)

```
## 6.2 - Analytics Zoo install manually

# Step 1 : Install  JDK 8, preferably Oracle JDK
export JAVA_HOME=/path/to/jdk

# Step 2 : Download and install Spark
# location : https://spark.apache.org/downloads.html
export SPARK_HOME=/path/to/spark

# Step 3 : Download Latest Analytics Zoo
# location : https://analytics-zoo.github.io/master/#release-download/

# Step 4 : Extract Analytics Zoo
export ANALYTICS_ZOO_HOME=/path/to/analytics_zoo
```

*Other names and brands may be claimed as the property of others.

Analytics Zoo Install for Python* – Option 2 : Manual Install (2 of 2)

```
# Step 5 : Run one of the following
# 5.1 for Jupyter
$ANALYTICS_ZOO_HOME/bin/jupyter-with-zoo.sh

# 5.2 pyspark shell
$ANALYTICS_ZOO_HOME/bin/pyspark-with-zoo.sh

# 5.3 spark shell (Scala)
$ANALYTICS_ZOO_HOME/bin/spark-shell-with-zoo.sh
```

*Other names and brands may be claimed as the property of others.

Using Analytics Zoo in Python*

Make sure the following env variables are set

- SPARK_HOME
- ANALYTICS_ZOO_HOME

always first call **init_nncontext()** at the very beginning of the code. This will create a SparkContext with optimized performance configuration and initialize the BigDL engine

```
## 6.3 - Zoo in Python
from zoo.common.nncontext import init_nncontext
sc = init_nncontext("sample app")
```

*Other names and brands may be claimed as the property of others.

ANALYTICS ZOO FEATURES

Analytics Zoo Features (List)

Distributed Tensorflow* and Keras* on Apache Spark*/BigDL

High-level abstractions and APIs

Built-in deep learning models

*Other names and brands may be claimed as the property of others.

Zoo Features: high-level API

Analytics provides DataFrame-based high-level API

Also supports native integration with Apache Spark* ML Pipeline

These are in NNFrames package

NNFrames provides both Python* and Scala* APIs

Compatible with Spark* v1.6 and v2.x

*Other names and brands may be claimed as the property of others.

NNFrames Highlights

Easy-to-use DataFrame (DataSet)-based API for training, prediction and evaluation

Integration with Apache Spark* ML pipeline and compatibility

Can use feature transformers and algorithms in Spark* ML.

Run inference or transfer learning from pre-trained models of Caffe*, Keras*, Tensorflow* or BigDL.

Training of BigDL built-in neural models (e.g., Inception, ResNet, Wide And Deep).

Rich toolset for feature extraction and processing, including image, audio and texts.

NNFrames Useful Classes

Class	Description
NNEstimator	Extends org.apache.spark.ml.Estimator
NNModel	Extends Apache Spark's* ML Transformer
NNClassifier	Used for classification tasks
NNClassifierModel	Specialized NNModel for classification task
NNImageReader	Reads in Image data

*Other names and brands may be claimed as the property of others.

NNFrames: NNImageReader

Load images into DataFrames using NNImageReader
Process loaded data using DataFrame transformations

```
## 6.9 - NNImageReader
from zoo.common.nncontext import init_nncontext
from zoo.pipeline.nnframes import NNImageReader

sc = init_nncontext()

# read images from a path
imageDF = NNImageReader.readImages('/path/to/images', sc)

getName = udf(lambda row: ...)

df = imageDF.withColumn("name", getName(col("image")))
```

zoo.feature.image Package : Feature Engineering of Images

Class	Description
NNImageReader	Reads images into Apache Spark* DataFrame
Image Transformers	Zoo provides many pre-defined image processing transformers built on top of OpenCV™
	ImageBrightness – adjusts Image brightness
	ImageResize – resize image
	ImageMatToTensor - Transform opencv mat to tensor
	for full list see here

*Other names and brands may be claimed as the property of others.

zoo.feature.image Package: Feature Engineering of Images

Process Images using built-in feature engineering operations

```
##  6.10 - Zoo Image Processing
from zoo.pipeline.nnframes import NNImageReader, ChainedPreprocessing
from zoo.feature.image import RowToImageFeature, ImageChannelNormalize,
ImageMatToTensor, ImageFeatureToTensor

transformer = ChainedPreprocessing(
                [RowToImageFeature(),
                ImageChannelNormalize(123.0, 117.0, 104.0),
                ImageMatToTensor(),
                ImageFeatureToTensor()])
```

Zoo : Keras* Style APIs

```
## 6.11 - Zoo : Keras Style APIs
from zoo.pipeline.api.keras.layers import Convolution2D, MaxPooling2D, Dense
from zoo.pipeline.api.keras.models import Sequential

model = Sequential()
        .add(Convolution2D(32, 3, 3, activation='relu',
                input_shape=(1, 28, 28))) ¥
        .add(MaxPooling2D(pool_size=(2, 2))) ¥
        .add(Flatten())
        .add(Dense(10, activation='softmax')))
```

*Other names and brands may be claimed as the property of others.

NNFrames: Native DL Support in Apache Spark* Dataframes

Train Model Using Spark ML Pipelines

Process loaded data using DataFrame transformations

```
## 6.12 - NNFrames : Support for Spark DataFrames
from zoo.pipeline.nnframes import NNEstimator
from bigdl.nn.criterion import CrossEntropyCriterion

estimator = NNEstimator(model, CrossEntropyCriterion(), transformer) ¥
                .setLearningRate(0.003) ¥
                .setBatchSize(40) ¥
                .setMaxEpoch(1) ¥
                .setFeaturesCol("image")
                .setCachingSample(False)

# use fit on DataFrames
nnModel = estimator.fit(df)
```

ANALYTICS ZOO BUILT-IN MODELS

Built-in Models

Model	Description
Object Detection	Identification of Objects within Images
Image Classifier	Classification of Images as one of n classes
Text Classification	Identifying Text as one of n classes
Recommendation	Predict User-item relationship

Object Detection API

Analytics Zoo provides a collection of pre-trained models for Object Detection

These models can be used for out-of-the-box inference

Two typical kind of pre-trained Object Detection models: SSD and Faster RCNN-supported models

- PASCAL VOC model
- COCO model

Object Detector Sample Usage – Python*

```
## 6.4 - Object Detection API
from zoo.common.nncontext import init_nncontext
from zoo.models.image.objectdetection import ImageSet, ObjectDetector

sc = init_nncontext("image detector app")

model = ObjectDetector.load_model(model_path)
image_set = ImageSet.read(img_path, sc)
output = model.predict_image_set(image_set)
```

*Other names and brands may be claimed as the property of others.

Image Classification API

Analytics Zoo provides a collection of pre-trained models for Image Classification

Provided models:

- Alexnet
- Inception-V1
- VGG
- Resnet
- Densenet, Mobilenet, Squeezenet

You can download the models here

Image Classification Usage

```
## 6.5 - Object Detection API
from zoo.common.nncontext import init_nncontext
from zoo.models.image.objectdetection import ImageSet, ImageClassifier

sc = init_nncontext("image detector app")

imc = ImageClassifier.load_model(model_path)
image_set = ImageSet.read(img_path, sc)
output = imc.predict_image_set(image_set)
```

Text Classification API

Analytics Zoo provides pre-defined models having different encoders that can be used for classifying texts.

Features:

- Easy-to-use models, could be fed into NNFrames or BigDL Optimizer for training
- The encoders we support include CNN, LSTM and GRU

Example 1

Text Classifier Sample Code

```
## 6.6 - Text Classification API
## Step 1 : define the model
from zoo.models.image.objectdetection import TextClassifier


text_classifier = TextClassifier(
                    class_num,  # The number of text categories to be classified.
                    Positive int.
                    token_length,  # The size of each word vector. Positive int.
                    sequence_length=500, # The length of a sequence. Positive int.
                    Default is 500.
                    encoder="cnn", # The encoder for input sequences. String. 'cnn' or
                    'lstm' or 'gru' are supported. Default is 'cnn'.
                    encoder_output_dim=256 # The output dimension for the encoder.
                    Positive int. Default is 256.
                    )
```

Text Classifier Sample Code

```
## Step 2 : train the model
optimizer = Optimizer(
    model=text_classifier,
    training_rdd=train_rdd,
    criterion=ClassNLLCriterion(logProbAsInput=False),
    end_trigger=MaxEpoch(20),
    batch_size=128,
    optim_method=Adagrad(learningrate=0.01, learningrate_decay=0.001))

optimizer.set_validation(
    batch_size=128,
    val_rdd=val_rdd,
    trigger=EveryEpoch(),
    val_method=[Top1Accuracy()])
```

Text Classifier Sample Code

```
## Step 3 : Predict
# Predict for probability distributions.
results = text_classifier.predict(rdd)

# Predict for class labels. By default, label starts from 0.
result_classes = text_classifier.predict_classes(rdd)
```

Recommendation API

Analytics Zoo provides two Recommender models, including Wide and Deep (WND) learning model and Neural network-based Collaborative Filtering (NCF) model.

Features:

- Easy-to-use models, could be fed into NNFrames or BigDL Optimizer for training
- Recommenders can handle either explicit or implicit feedback, given corresponding features
- It provides three user-friendly APIs to predict user item pairs, and recommend items (users) for users (items)

Example1

Recommendation API Usage

```
## 6.7 - Recommender API
## define  model
wide_n_deep = WideAndDeep(
                class_num,
                column_info,
                model_type="wide_n_deep",
                hidden_layers=(40, 20, 10))

## train
optimizer = Optimizer(
                model=wide_n_deep,
                training_rdd=train_data,
                criterion=ClassNLLCriterion(),
                optim_method=Adam(learningrate = 0.001, learningrate_decay=0.00005),
                end_trigger=MaxEpoch(10),
                batch_size=batch_size)
optimizer.optimize()
```

Recommendation API Usage

```
## predict
userItemPairPrediction = wide_n_deep.predict_user_item_pair(valPairFeatureRdds)

userRecs = wide_n_deep.recommend_for_user(valPairFeatureRdds, 3)
itemRecs = wide_n_deep.recommend_for_item(valPairFeatureRdds, 3)
```

Summary

Learned about:

- high level Dataframe APIs in Apache Spark*
- Analytics Zoo API
- Handy classes in Analytics Zoo

*Other names and brands may be claimed as the property of others.

Lab 6.1 : Zoo APIs

Overview:
We will get familiar with Analytics Zoo APIs

Run Time
20-30 mins

Instructions
Please follow lab guide

Recommended Resources

- Intel Analytics Zoo : https://github.com/intel-analytics/analytics-zoo

- Intel AI Academy : https://software.intel.com/ai-academy

- https://software.intel.com/en-us/ai-academy/basics