Teradata Data Science Day 2018

Kyloで構築するDatalakeと分析環境

~ データラングリング ~

テラデータ・コンサルティング本部
シニア・データエンジニア　伊藤 真央

講演者紹介

シニア・データエンジニア

伊藤 真央

- 学士・修士 (島根大学・ウィスコンシン大学マディソン校)
- 米国のStartupを皮切りに、Smart Home, ERP, Web Marketingなど多様な業界でのクラウドシステム・機械学習・分散処理に関するプロジェクトに関わる
- 現在は、主にDataLake関連のプロジェクトに従事

本日のアジェンダ

- Kyloとは？
- Kylo と 分析環境
- 分析デモ ~ モーションキャプチャデータ BVH
- 分析デモ ~ アーキテクチャ
- 分析デモ ~ 機械学習 - Clustering
- 分析デモ ~ データラングリング
- 分析デモ ~ 機械学習 適用ステップ
- 分析デモ ~ 結論
- まとめ

Kyloについて

Kyloの特徴についてまとめると…

- DataLake構築ツールのOSS
- ビッグデータの要素技術ではない
  SparkやHadoop等のツールを置き換えるものではない
- NiFiの拡張版
- テンプレート管理ツール

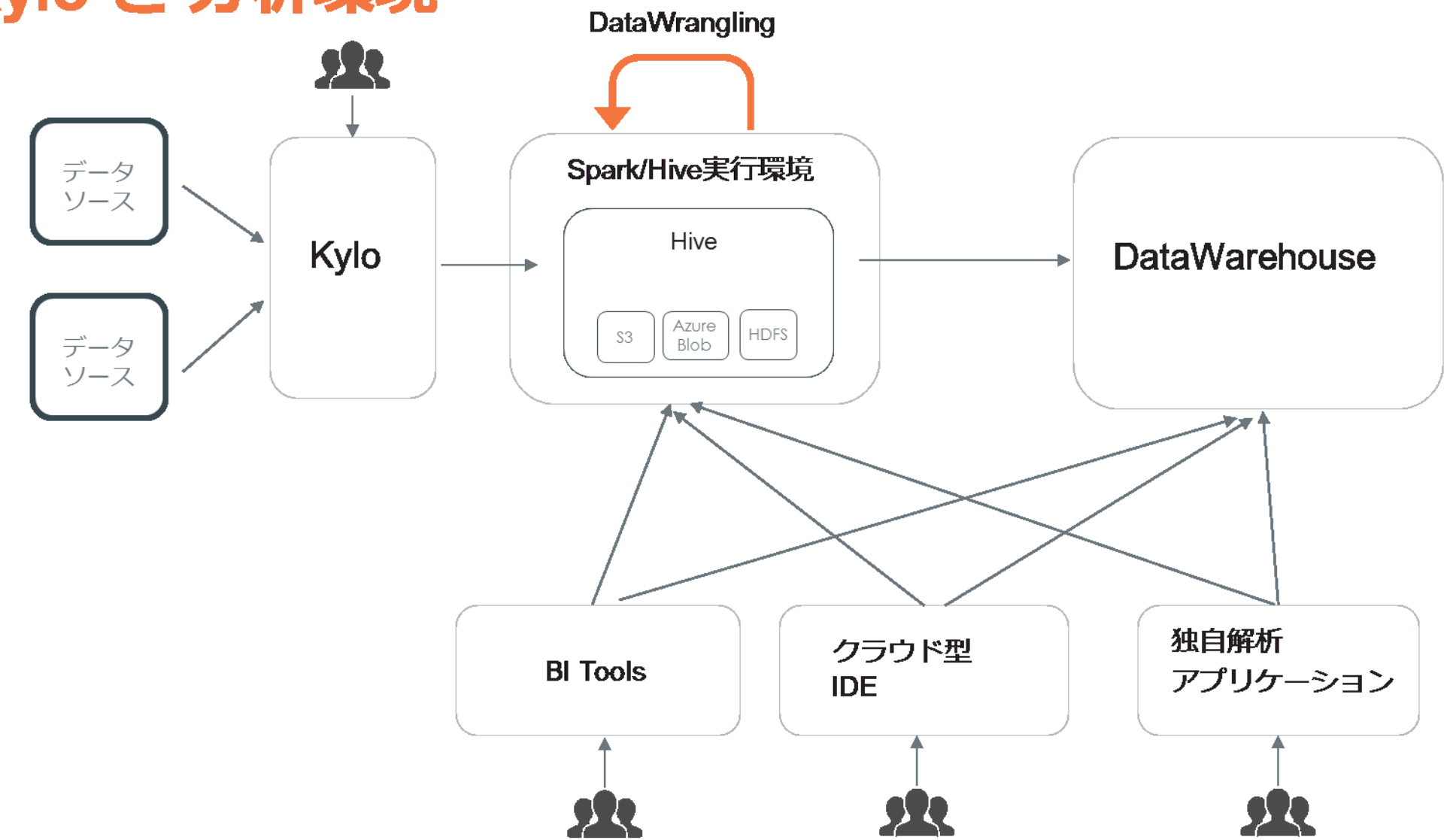
Kylo 概観

いくつかのOSSは別で代用できる：例えば、

elasticserach -> Solr

MySQL -> PostgreSql

Kylo と 分析環境



分析デモ ~ 人間の動作解析 via モーションキャプチャデータ

モーションキャプチャデータ (BVH File) from カーネギーメロン大学：

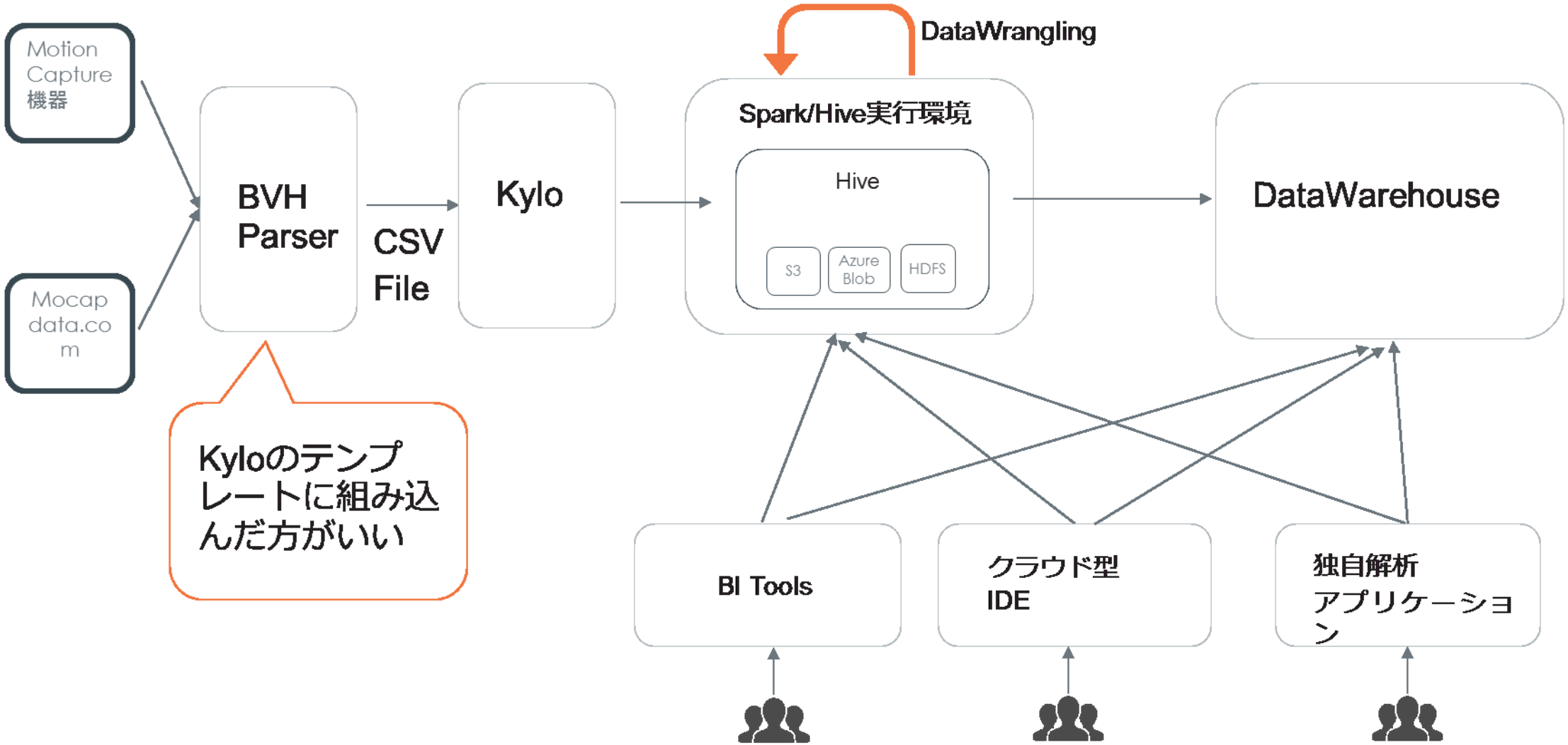
(1) 元のデータ (BVH Format未対応)
http://mocap.cs.cmu.edu/

(2) CMUのデータを、BVHファイルに変換して、ホスティングしてくれているサイト
https://sites.google.com/a/cgspeed.com/cgspeed/motion-capture/cmu-bvh-conversion :

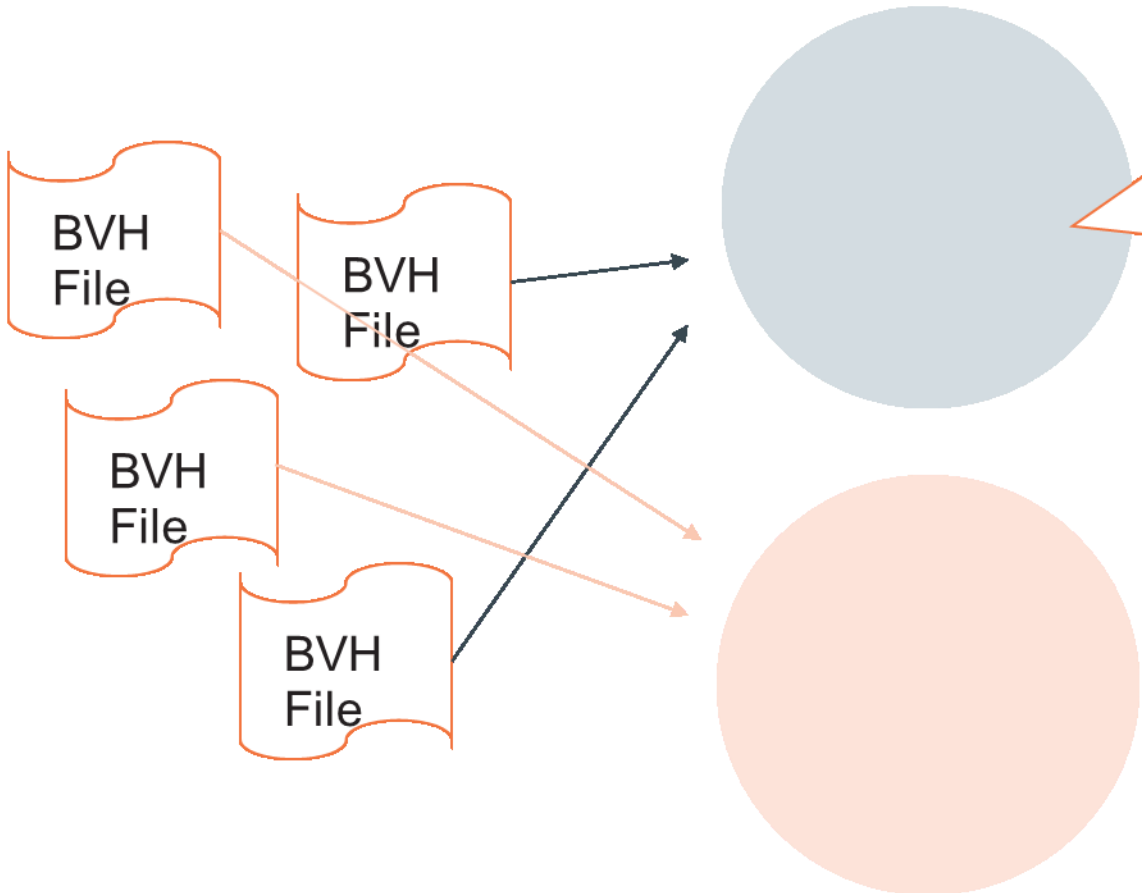
分析デモ ~ BVH File – モーションキャプチャデータ

実際のBVHファイルのフォーマットをお見せします

分析デモ ~ アーキテクチャ



分析デモ ~ 機械学習 – クラスタリング



それぞれのグループが、同じ動作のファイルのみを含んでいるか確認。

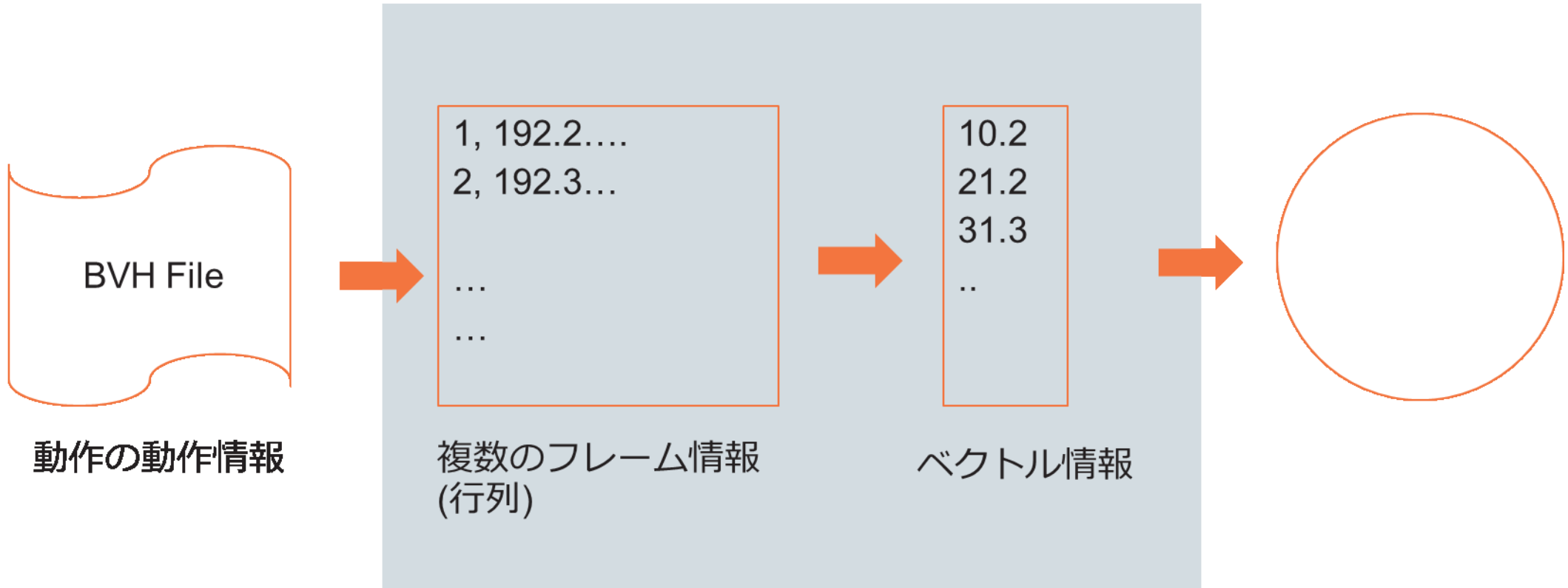
例えば，“歩く”という動作は、全て同じグループに属しているか

説明の簡単化のために、以下のデータに着目：

1. 2つの動作だけ対象（”歩く”、”走る”）

2. hip_zposition だけで、クラスタリング
   (お尻の縦の位置情報)

分析デモ～データラングリング



それでは、デモを始めます

分析デモ ~ データラングリング

1. Frame間の差を計算するために、一つ前のコラムを作成
lag(hip_zposition,1).over(partitionBy(category,sub_category,bvh_file_index).orderBy(frame_no)).as(“hip_zposition_before”)

2. Filter Rowsで、最初のフレームを削除
filter(!isnull(hip_zposition_before))

3. 差を求める
subtract(hip_zposition,hip_zposition_before).as(“hip_zposition_difference”)

4. Frameをaggし、ベクトル情報に変換
groupBy(category,sub_category,bvh_file_index).agg(avg(hip_zposition_difference).as(“hip_zposition_difference_average”))

5. データの標準化
vectorAssembler(["hip_zposition_difference_average"],"hip_position_vector")
StandardScaler().setInputCol("hip_position_vector").setOutputCol("hip_position_standarlized").setWithMean(true).setWithStd(true).run(select(hip_position_vector))

分析デモ ~機械学習 適用ステップ

Kyloで機械学習を行う場合は、以下のステップを行う

1. Python等を使って、機械学習のモデルを学習させる
  (1) Pythonなどで、Data Wrangling済みのHive Tableを読み込む
  (2) Spark Mlibなどで、使いたいモデルを, Pipelineで構築し、学習させる
  (3) 学習済みのPipelineを saveメソッドで保存
  * HDFSに、学習済みのモデルが保存される
  * Piplineの情報しか Kylo側で読み込めないので、注意

2. KyloのData Wranglerで、transform(modelpath)による機械学習適用
  * modelpathは、(3) で保存したパスを指定。

=> Data Wrangling UIでは、”学習”フェーズは、不可。予測フェーズのみ可能

分析デモ ~ 結論

今回の結果では、非常に綺麗に、”歩く”と”走る”という２つのグループに、クラスタリングすることができた。しかしながら、以下の点から、データ数が増えると、うまく識別できない動作も出てくる：

1. ”お尻”のポジションでの分析
 => ”走る“と”歩く”では、当然、お尻の上限の差は違いがあると思われる。
しかし、人間によっては、あまりお尻のポジションが変わらずに、歩いたり走ったりする可能性もあるため、そのようなケースが出てしまった場合、間違った結果になる

まとめ

分析という観点で、Kyloをまとめると…

1. KyloはデータのData Wranglingによる前処理が主な役割
   => シンプルな集計的な分析なら、Kyloで対応可

2. 機械学習などの複雑な処理は、他のツールを使用する

Thank you.

データラングリング – その他

1. データ正規化
2. データ標準化
3. Condition
4. Pivot

データラングリング – データ正規化

$$\text{Normalize(hip\_zposition)} = \frac{\text{hip\_zposition} - \min(\text{hip\_zposition})}{\max(\text{hip\_zposition}) - \min(\text{hip\_zposition})}$$

データラングリング – データ正規化（MinMaxScalerなし)

1. 正規化
(1) max と min値を持つData Frameを作成
groupBy(category,sub_category,bvh_file_index).agg(max(hip_zposition).as(“hip_zposition_max”), min(hip_zposition).as(“hip_zposition_min”))
(2) 結果を、Hive Tableとして保存
(3) Join オリジナルテーブル + (2)のテーブル
　　Canvasでは、一つしかJoin Keyを指定できないので、Edit SQL で追加する
(4) MaxとMinで標準化
(subtract(hip_zposition,hip_zposition_min)/subtract(hip_zposition_max,hip_zposition_min)).as("hip_zposition_normalized")

Catalog機能で、結果を確認
 SELECT * FROM `default`.`hip_zposition_normalized` where hip_zposition_normalized > 0 LIMIT 20

データラングリング – データ正規化（MinMaxScaler使用)

1. 正規化
(1) Spark Mlibが扱える型へ変換

vectorAssembler([“hip_xposition”, “hip_yposition”],“hip_position_vector”)

(2) MinMaxScalerで、正規化
StandardScaler().setInputCol("hip_position_vector").setOutputCol("hip_xposition_standarlized").setWithMean(true).setWithStd(true).run(select(hip_position_vector))

データラングリング – データ標準化

$$\text{standarlize(hip\_zposition)} = \frac{\text{hip\_zposition} - \mu}{\sigma}$$

データラングリング – データ標準化 (StandardScaler使用)

1. 標準化
(1) Spark Mlibが扱える型へ変換

vectorAssembler([“hip_xposition”, “hip_yposition”],“hip_position_vector”)

(2) StandardScalerで、標準化

StandardScaler().setInputCol("hip_position_vector").setOutputCol("hip_position_standarlized"
).setWithMean(true).setWithStd(true).run(select(hip_position_vector))

データラングリング – Condition

when(isnull(hip_zposition_before), 0).otherwise(hip_zposition_before)

新しい列を作り
Hip_zposition_beforeの値が, null (isnull)の場合、0を記入。
それ以外であれば、hip_zposition_beforeの値を入力。

データラングリング – Pivot

groupBy(sub_category).pivot("bvh_file_index").agg(avg(hip_xposition))

注意ポイント：pivot関数は引数がstring型なので、""で囲むのを忘れないようにする