Rubyで機械学習が出来る未来を目指すRed Data Toolsの現状と今後について

by Yusaku Hatanaka
@RubyWorld Conference 2017

self.intoroduction

```
{
   name: "Yusaku Hatanaka",
   twitter: "@hatappi",
   github: "hatappi",
   hatena: "hatappi1225",
   company: "Speee, Inc."
   languages: %w(ruby go python),
   icon: " "
}
```

UZOU

UZOUはコンテンツやユーザーを分析した上で最適な広告を配信し、
ユーザーのまだ見ぬ体験を提供する、ネイティブアド配信プラットフォームです。

媒体様向け

広告主様・広告代理店様向け

Agenda

- Rubyにおける機械学習の現状について
- Red Data Toolsについて
- 現状の取り組み
- 未来について
- まとめ

Ruby

機械学習したい！

機械学習に限らず
データ分析において
データ連携は必要となる

データ連携というと

JSONやCSV ??

JSONやCSV

ファイル間でのやりとりだけでなくAPIでのリクエスト・レスポンスなどで使用される

```python
import pandas as pd
import pyarrow as pa
from time import time

df = pd.DataFrame({"a": ["a" * i for i in list(range(5000))],
                   "b": ["b" * i for i in list(range(5000))],
                   "c": ["c" * i for i in list(range(5000))]})

start_time = time()
df.to_csv("/dev/shm/test.csv", header=False)
print("write : {0}s".format(time() - start_time))
```

```
write : 0.8009328842163086s
```

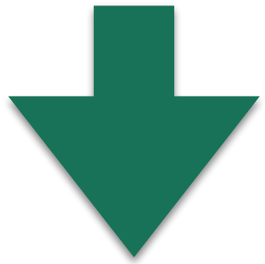
```ruby
require "time"
require "csv"

start_time = Time.now
CSV.open("/dev/shm/test.csv", headers: false) do |row|
  puts "batch size : #{row.count}"
end
puts "read : #{Time.now - start_time}s"
 File.delete "/dev/shm/test.csv"
```

```
batch size : 5000
read : 0.12274529s
```

Pythonで出力したCSVをRubyで読み込む例

CSVやJSONで
大きなデータを処理

特定の行で集計処理をしたい場合でも
一度すべてを読み込む必要があるので
大きなデータを処理するには限界がある



機械学習では数百万の大きなデータセットを使うことも
あるので大きなデータを扱えるようになりたい

Apache Parquet

Apache Parquetとは

- カラム型のデータフォーマット
- 全部を読み込まずともデータの一部を取り出せる
  - 例えば特定の列を取り出して集計する
- サイズが小さい
  - カラムでデータを格納するため同じ型のデータを圧縮することが出来る

大きなデータを処理して保存することに向いている

Apache Parquet

```python
import pandas as pd
import pyarrow as pa
import pyarrow.parquet as pq
from time import time

df = pd.DataFrame({"a": ["a" * i for i in list(range(5000))],
                   "b": ["b" * i for i in list(range(5000))],
                   "c": ["c" * i for i in list(range(5000))]})
start_time = time()
table = pa.Table.from_pandas(df)
pq.write_table(table, "/dev/shm/pandas.parquet")
print("write : {0}s".format(time() - start_time))
```

```
write : 0.1390378475189209s
```

```ruby
require 'arrow'
require 'parquet'
require "time"

path = '/dev/shm/pandas.parquet'
start_time = Time.now
reader = Parquet::ArrowFileReader.new(path)
table = reader.read_table
puts table.n_rows
puts "read : #{Time.now - start_time}s"
 File.delete path
```

```
5000
read : 0.042164239s
```

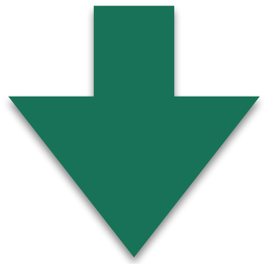
Python上のデータをParquet形式でファイル出力したものを
Rubyで読み込む

大量のデータを

扱えるようになった🎉

複数のシステム間でのデータを連携

Parquetはデータを圧縮するためデータ処理する過程で複数システム間でデータ連携を行う際はそれを元に戻すオーバーヘッドが生じる



シームレスにシステム間でデータ連携したい!

Apache Arrow

Apache Arrowとは

- オンメモリでカラム型データを扱うためのフォーマットとアルゴリズム
- シリアライズ・デシリアライズコストが**ほぼ0**
  - データを渡す側も受け取る側もCPUを使わない
- ゼロコピー
  - データを受け取る側はメモリコピーに時間を使わなくてもよい

データ交換にむいている

Apache Arrow

```python
import pandas as pd
import pyarrow as pa
from time import time

df = pd.DataFrame({"a": ["a" * i for i in list(range(5000))],
                   "b": ["b" * i for i in list(range(5000))],
                   "c": ["c" * i for i in list(range(5000))]})
start_time = time()
record_batch = pa.RecordBatch.from_pandas(df)
with pa.OSFile("/dev/shm/pandas.arrow", "wb") as sink:
    schema = record_batch.schema
    writer = pa.RecordBatchFileWriter(sink, schema)
    writer.write_batch(record_batch)
    writer.close()
print("write : {0}s".format(time() - start_time))
```

```
write : 0.07749533653259277s
```

```ruby
require "arrow"
require "time"
require "csv"

Input = Arrow::MemoryMappedInputStream

start_time = Time.now
Input.open("/dev/shm/pandas.arrow") do |input|
  reader = Arrow::RecordBatchFileReader.new(input)
  puts "batch size : #{reader.get_record_batch(0).count}"
end
puts "read : #{Time.now - start_time}s"
 File.delete "/dev/shm/pandas.arrow"
```

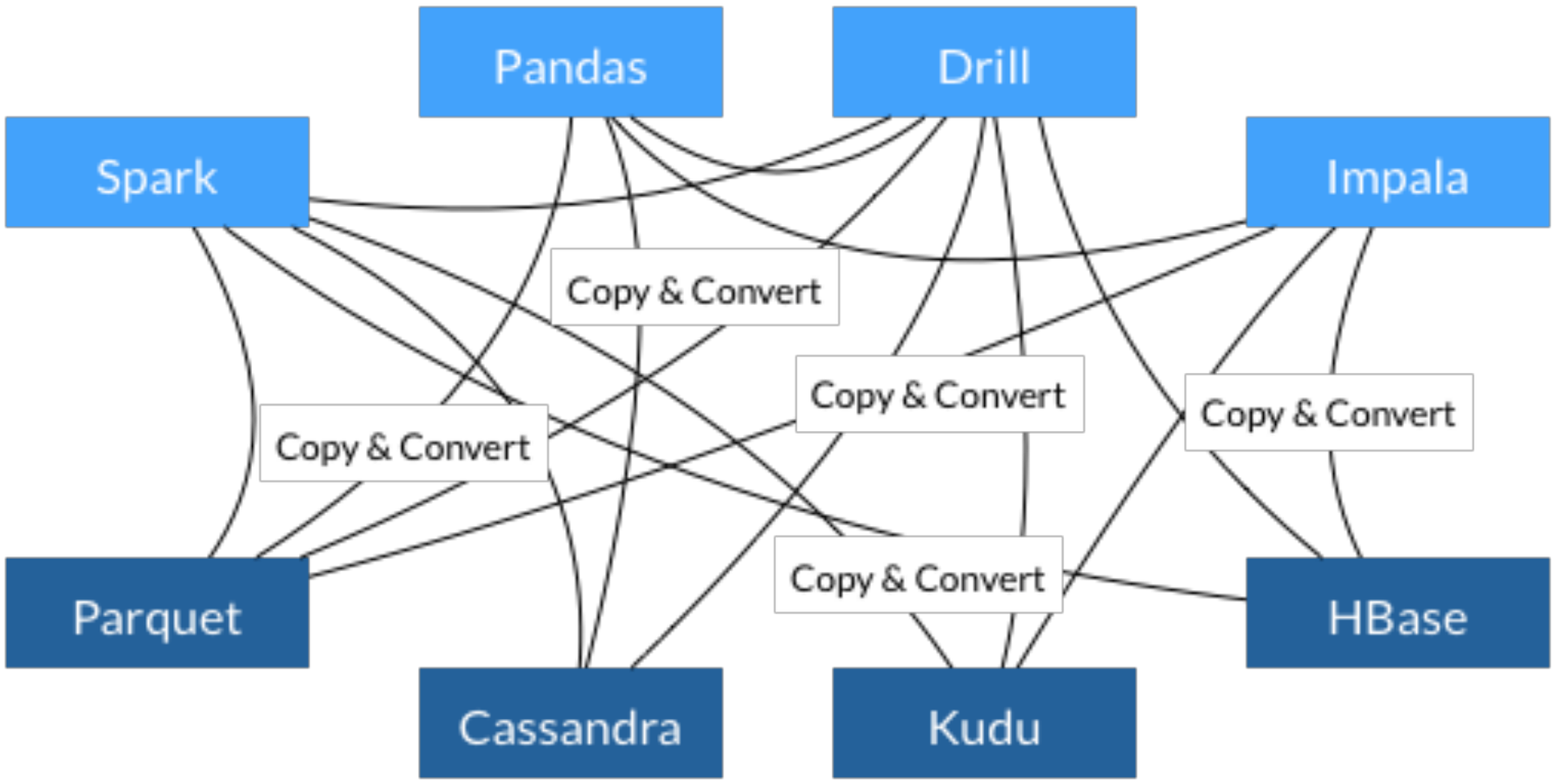
```
batch size : 5000
read : 0.007414542s
```

Python上のデータをArrow形式で出力しRubyで読み込む

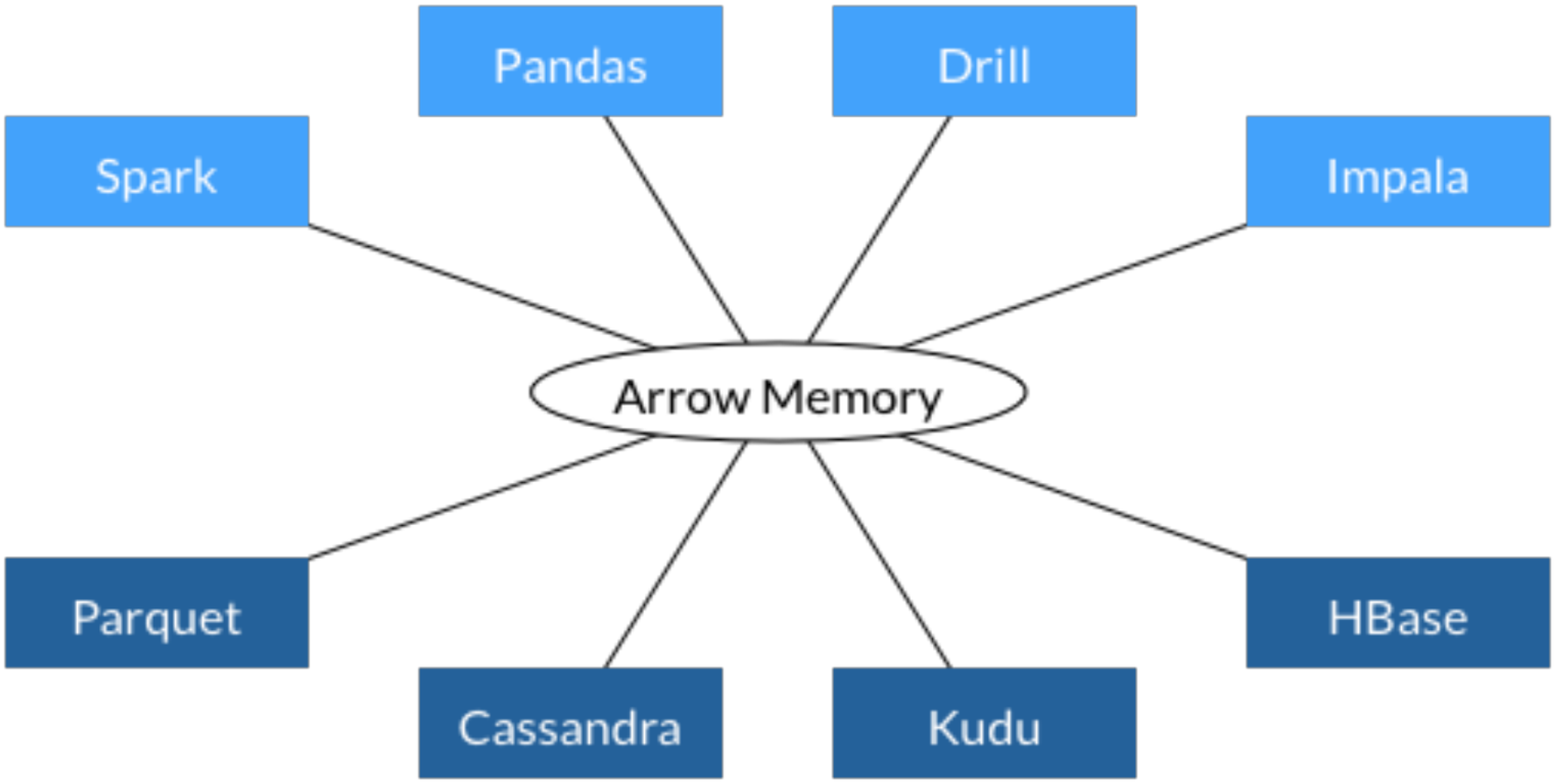
5000レコードに対する読み書き比較まとめ

	Pythonでの書き込み (s)	Rubyでの読み込み (s)
CSV	0.8009	0.1227
Apache Parquet	0.1390	0.0422
Apache Arrow	0.0775	0.0074

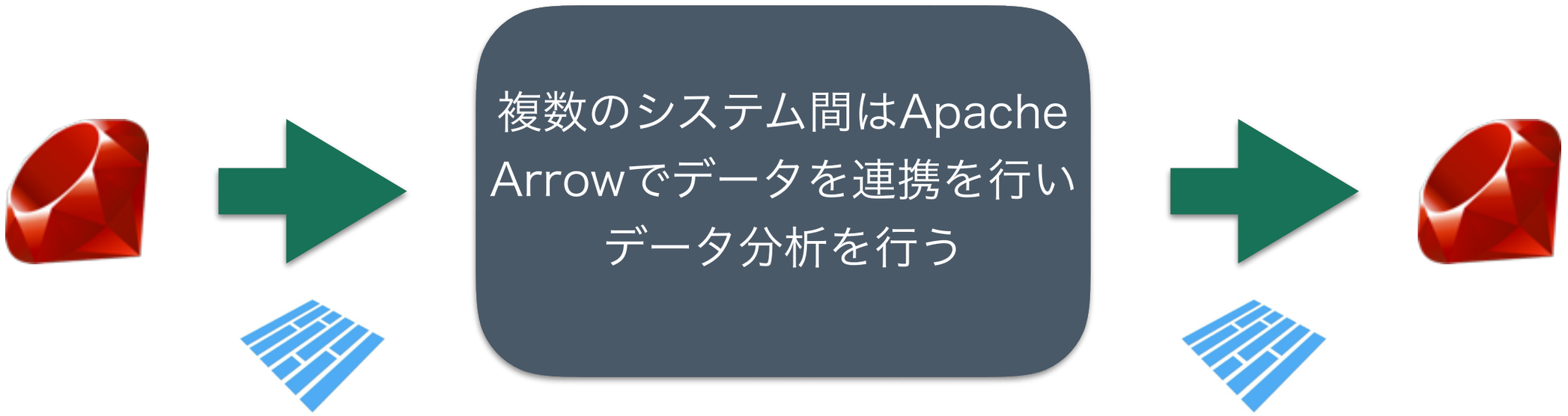
Apache Arrowがない時



Apache Arrowがある時



RubyでParquetを扱えると



Parquetを扱うことでデータの連携は出来た

しかしRubyでデータ分析がしたい！！には届かない

RubyでArrowを扱えると

Rubyでデータ処理の一部を担うことが出来る!!

RubyでArrowを扱えると

Apache Arrowに対応することで必要な部分から
Rubyを使ったデータ分析をはじめることが出来る

例

- Rubyで集めたデータをApache Arrowに対応しているPandasやSparkに連携し分析した結果をRubyで受け取ってRailsを使ったwebアプリで可視化

- 後にPandasやSpark部分を徐々にRubyへ移行することも出来る

RED
DATA
TOOLS

Agenda

- Rubyにおける機械学習の現状について
- Red Data Toolsについて
- 現状の取り組み
- 未来について
- まとめ

Red Data Tools

- 株式会社クリアコードの須藤さんが2017年2月にプロジェクトを設立

- **Ruby用のデータ処理ツールを提供すること**を目的としたプロジェクト

- 多くの言語が共通して使用できるApache Arrowを使用することで**Rubyコミュニティーを超えて協力する**

Agenda

- Rubyにおける機械学習の現状について
- Red Data Toolsについて
- 現状の取り組み
- 未来について
- まとめ

現状の取り組み

- Red Arrow
- 既存gemのArrow化
- 新しいツールを提供

現状の取り組み

- Red Arrow
- 既存gemのArrow化
- 新しいツールを提供

Red Arrow

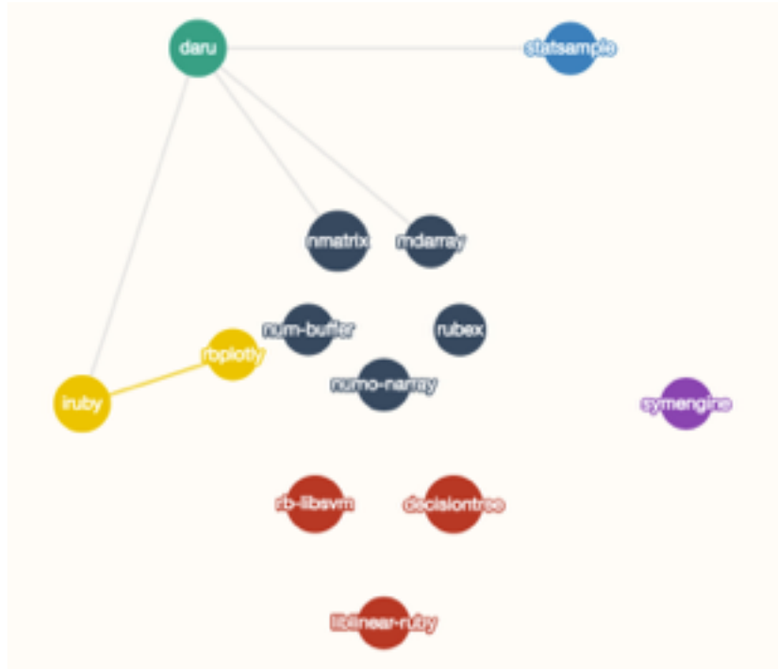
- Apache ArrowのRubyバインディング
- GitHub: red-data-tools/red-arrow

現状の取り組み

- Red Arrow
- 既存gemのArrow化
- 新しいツールを提供

SciRuby・Ruby Numo

- 科学技術計算、データ可視化用途のGem群を開発しているプロジェクト
- Pythonでいうpandasにあたるdaruやnumpyにあたるnumo-narryなどがある
- Gem間が独立していて連携が難しい

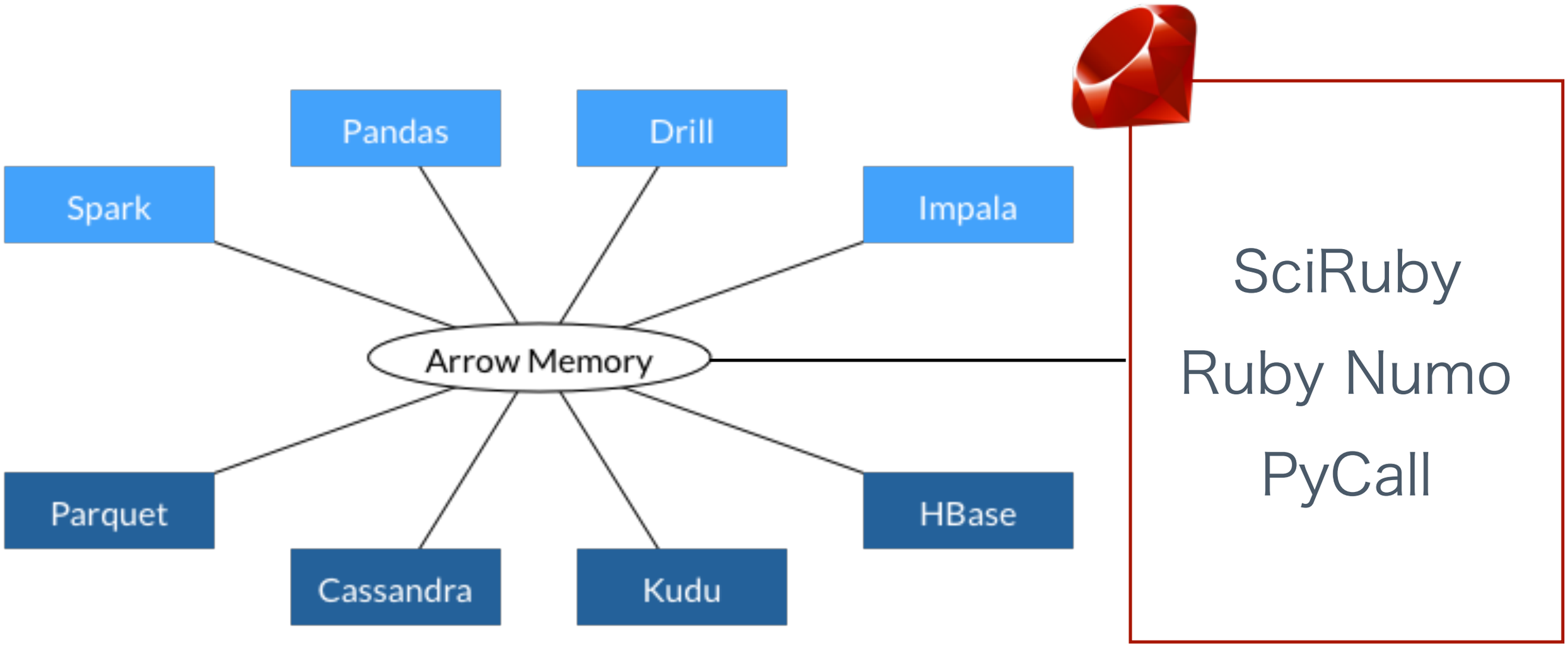


PyCall

- @mrknさんが作成されているRubyとPythonのブリッジライブラリ
- Pythonで作成された既存の資産を使うことが出来るのでRubyでPythonのオブジェクトを使用することが出来る

```ruby
require 'pycall/import'
include PyCall::Import

pyimport 'numpy', as: :np

x = np.matrix.([[1,2], [3, 4]])
# => matrix([[1, 2],
#            [3, 4]])
y = np.matrix.([[5,6], [7, 8]])
# => matrix([[5, 6],
#            [7, 8]])

z = x * y
# => matrix([[19, 22],
#            [43, 50]])
z.max.()
# => 50
```

既存gemのArrow化



Apache Arrowに対応させることで
既存のgemを使用してデータ分析をはじめられる

現状の取り組み

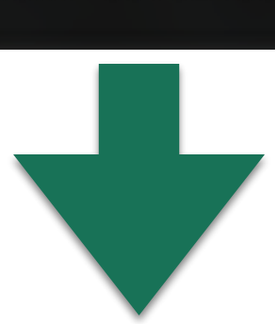
- Red Arrow
- 既存gemのArrow化
- 新しいツールを提供

Red Chainer

- ChainerをRubyへポーティングしたもの
- Chainerのクラスやパラメータの持ち方を参考に Rubyで書くことで**0**から作るのではなく既存の資産を活かして作ることが出来る
- 内部の配列にはApache Arrowに対応したnumo-narryさせることでApache Arrow形式に変換することも可能

MNIST

```
epoch       main/loss   validation/main/loss  main/accuracy  validation/main/accuracy  elapsed_time
1           0.333724                          0.9069                                   33.3207
2           0.143013                          0.957867                                 66.4229
3           0.100396                          0.970133                                 101.706
4           0.0778566                         0.975917                                 138.321
5           0.0612491                         0.9807                                   170.414
6           0.0503291                         0.983667                                 201.796
     total [##################################.....................]  61.67%
this epoch [#########.........................................]  16.67%
      3700 iter, 6 epoch / 10 epochs
    17.878 iters/sec. Estimated time to finish: 00:01:08.
```



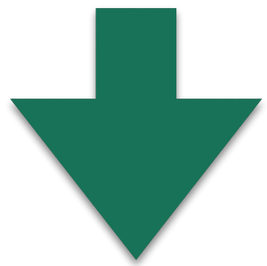
```
epoch       main/loss   validation/main/loss  main/accuracy  validation/main/accuracy  elapsed_time
1           0.333724                          0.9069                                   33.3207
2           0.143013                          0.957867                                 66.4229
3           0.100396                          0.970133                                 101.706
4           0.0778566                         0.975917                                 138.321
5           0.0612491                         0.9807                                   170.414
6           0.0503291                         0.983667                                 201.796
7           0.0408361                         0.9876                                   234.164
8           0.034511                          0.9894                                   266.771
9           0.0289062                         0.991133                                 300.145
10          0.0255364                         0.991817                                 334.758
```

Agenda

- Rubyにおける機械学習の現状について
- Red Data Toolsについて
- 現状の取り組み
- 未来について
- まとめ

Red Data Toolsの今後

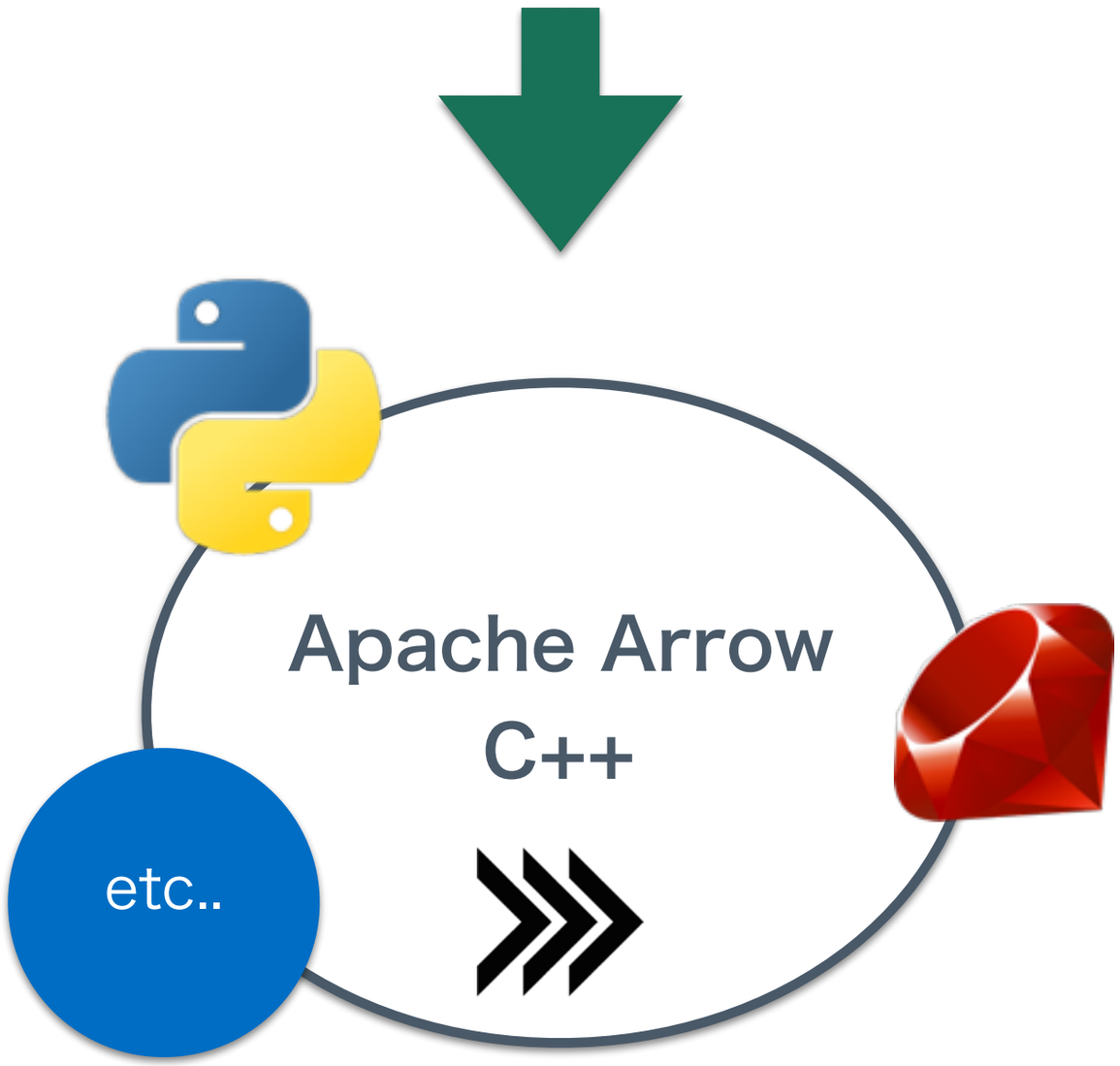
- 引き続き既存のgemのAapche Arrowへの対応は行う
- Red Chainerのような新しいツールの作成



Ruby間でデータ分析が出来るようになりたい

Red Data Toolsの今後

Apache Arrow本体への開発への参加



Agenda

- Rubyにおける機械学習の現状について
- Red Data Toolsについて
- 現状の取り組み
- 未来について
- まとめ

まとめ

- Apache Arrowに対応することで必要な部分からRubyでデータ分析をはじめることが出来る
- 少しづつでもRubyでデータ分析が出来る未来へ！

告知

開発イベントやってます

- 毎月1回の定期開催
- 場所は弊社で行ってます @東京 - 六本木
- データ分析これからの人でもどなたでもOKです！

Gitter

- 日本語　：https://gitter.im/red-data-tools/ja
- English：https://gitter.im/red-data-tools/en