機械学習エンジニアコース
Sprint

－ フレームワーク1_TensorFlow －

DIVE INTO CODE

フレームワークとは何か

Pythonにおけるフレームワークとは、Pythonを使用して機械学習・ディープラーニングやWebアプリケーションを開発する際の枠組みとして機能するソフトウェアのことを指す。
また、プログラムの基本機能をその制御方法も含めて提供する、ライブラリ(汎用性の高い複数のプログラムを一つにまとめたもの)の集合とも言える。

https://valiancesolutions.com/difference-framework-library/

フレームワークの種類

ディープラーニング用
フレームワーク

https://developer.nvidia.com/deep-learning-frameworks

フレームワークの種類

ディープラーニング用
フレームワーク

2019/12
開発を終了する意向を発表

https://preferred.jp/ja/news/pr20191205/

https://developer.nvidia.com/deep-learning-frameworks

フレームワークの使用頻度

職業別にフレームワーク使用頻度を比べてみよう

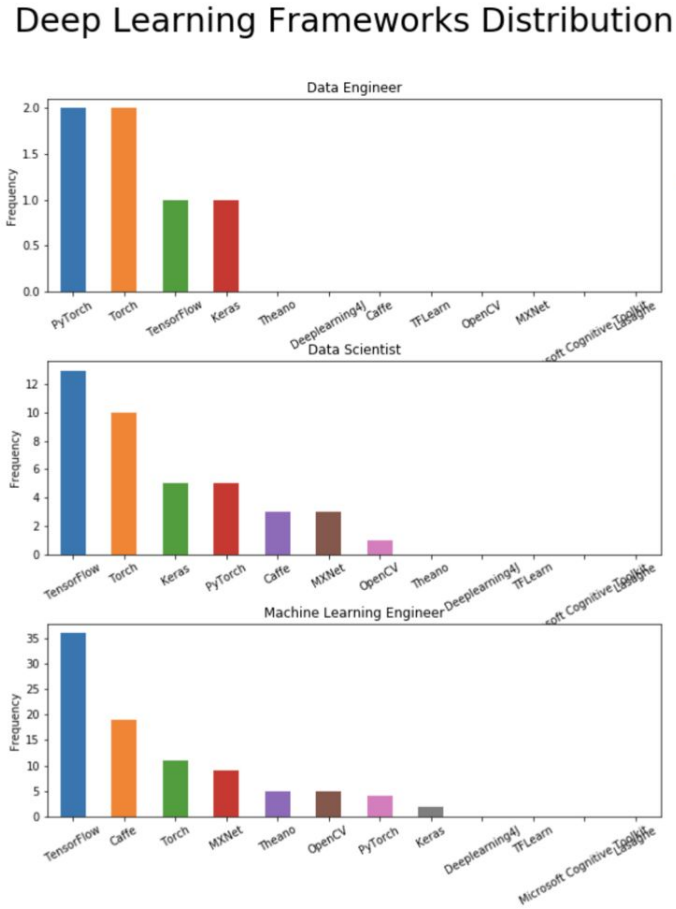
データエンジニア ・データサイエンティスト ・MLエンジニア

※2018年11月記事

https://towardsdatascience.com/what-does-an-ideal-data-scientists-profile-look-like-7d7bd78ff7ab

所感：
データエンジニアはフレームワークをほとんど使ってない?
データサイエンティストには高レベルAPIのKERASが人気?
MLエンジニアにはTensorflowのような低レベルAPIが人気?

※2020年のTensorflowとPytorchについて
https://blog.exxactcorp.com/pytorch-vs-tensorflow-in-2020-what-you-should-know-about-these-frameworks/



縦軸に注意!

Tensorflowとは何か

Tensorflowとは

Google Brainによって開発された、数値計算と大規模な機械学習のためのオープンソフトウェアライブラリである。
Pythonが、プログラムによって一連の手続きを記述し、制御の流れ（制御フロー）を表現する手続き型言語に属するのに対し、Tensorflowは、プログラムによってデータ間の一連の接続（有向グラフ）を記述し、データの移動（データフロー）を表現するデータフロープログラミング言語にあたる。
従来のTensorFlowはDefine-and-Runだったが、ver.2.0よりDefine-by-Runが採用される。

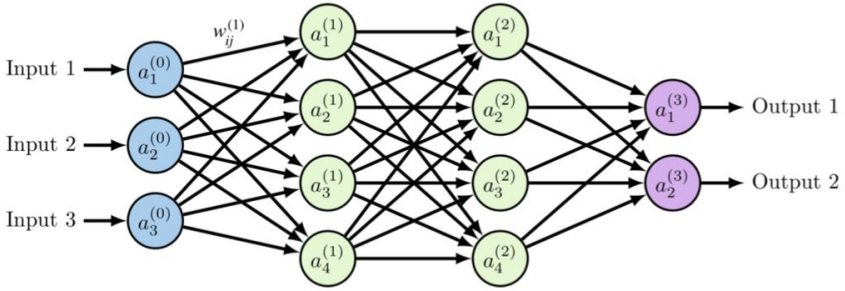
TensorFlowで用いられる知識

計算グラフ（データフローグラフ）

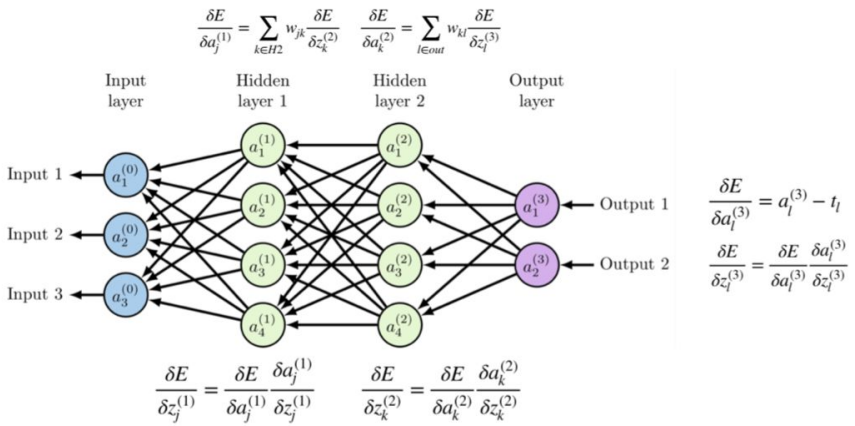
計算の流れを有向グラフ構造により記述したもの

アイデア：
ニューラルネットワークは、一旦計算グラフを構築するならば、それを基にバックプロパゲーションを行い学習することが可能である。
事例：
http://dianne.intec.ugent.be/#gettingstarted

前提：
データが通った計算のルートは記録されるため（裏返していうと、それ以外は記録していない）、その計算グラフに該当するパラメータのみが更新される。



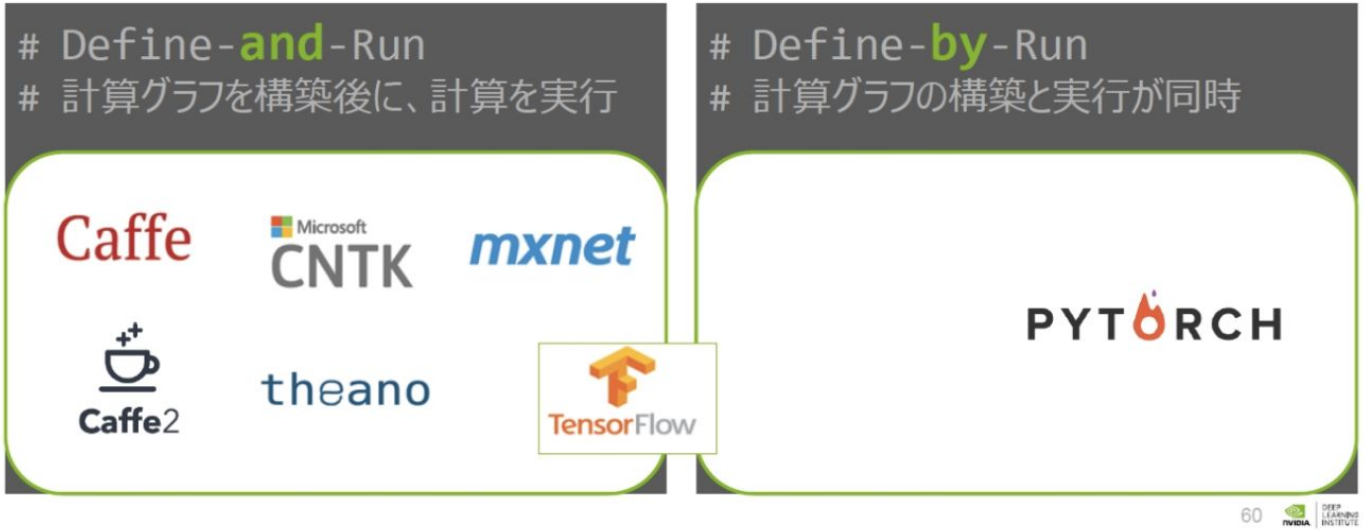
$$\hat{y} = \sigma[W^{(3)}\sigma[W^{(2)}\sigma[W^{(1)}a^{(0)} + b^{(1)}] + b^{(2)}] + b^{(3)}]$$



TensorFlowの構築

計算グラフの構築方法

フレームワークは計算グラフを構築するところが共通するが、構築方法によってフレームワークは２つに分類することができる。



TensorFlowの構築

計算グラフの構築方法

① Define - **by** - Run 方式

Define - **by** - Run

```
1  import numpy as np
2  a = np.ones(10)
3  b = np.ones(10) * 2
4  c = b * a
5  d = c + 1
```

アイデア：

Pythonと同じく一行ずつ実行する。インタプリタ的（逐次解釈しながら実行）と言われる。

事例：

c = b*a を実行した時に計算が遂行され、計算の履歴が生成される。

```
1  class IrisNN(chainer.Chain):
2      def __init__(self):
3          super(IrisNN,self).__init__(
4                  l1 = L.Linear(4,100),
5                  l2 = L.Linear(100,100),
6                  l3 = L.Linear(100,200),
7                  l4 = L.Linear(200,100),
8                  l5 = L.Linear(100,3))
9
10     def __call__(self,x):
11         prob = np.random.randn()
12         h = self.l1(x)
13         h = F.relu(h)
14         if prob > 0:
15             h = F.relu(self.l2(h))
16         else:
17             h = F.relu(self.l3(h))
18             h = F.relu(self.l4(h))
19         h = self.l5(F.dropout(F.relu(h),ratio=0.5))
20         return h
```

データを入力する__call__で初めて計算グラフを構築している

TensorFlowの構築

計算グラフの構築方法

② Define - **and** - Run 方式

アイデア：
計算の手順をグラフの形に変換し（計算グラフの定義）、
その後に**Session**を行ってはじめて計算を遂行する。コンパイラ的（実行可能な機械語に一括翻訳して、その機械語を実行することでプログラムを機能させる）といわれる。
事例：
C = B*Aが実行されても、計算は行っていない。

Define - **and** - Run
(擬似コード)

```
1  A = Variable('A')
2  B = Variable('B')
3  C = B * A
4  D = C + Constant(1)
5  # compiles the function
6  f = compile(D)
7  d = f(A=np.ones(10), B=np.ones(10)*2)
```

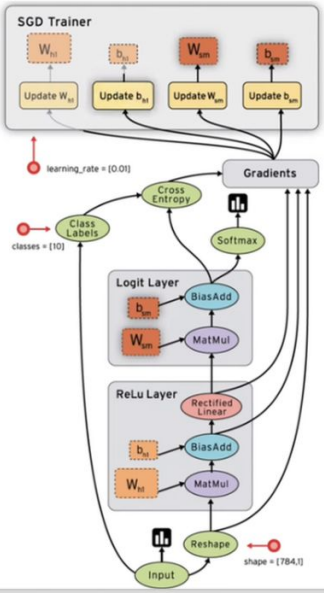
↑ Tensorflowでは、ここでSessionオブジェクトを
定義して計算実行させる

TensorFlowの構築

セッション（Session）

構築した計算グラフ（またはその一部）を実行する
Tensorflowのクラスのひとつ。

アイデア：
sessionは、計算グラフを構成するオペレーションオブジェクトを引数にとりそれを実行する。
前提：
sessionはランタイムシステム（実行モデルの振る舞いを実装したもの）を利用し、デバイスにアクセスする。このとき占有した物理的リソースを解放するために、withブロックを使用するか、またはsess.close()を用いて、このsessionを終了する。
https://www.tensorflow.org/guide/graphs



TensorFlowの構築

セッション（Session）

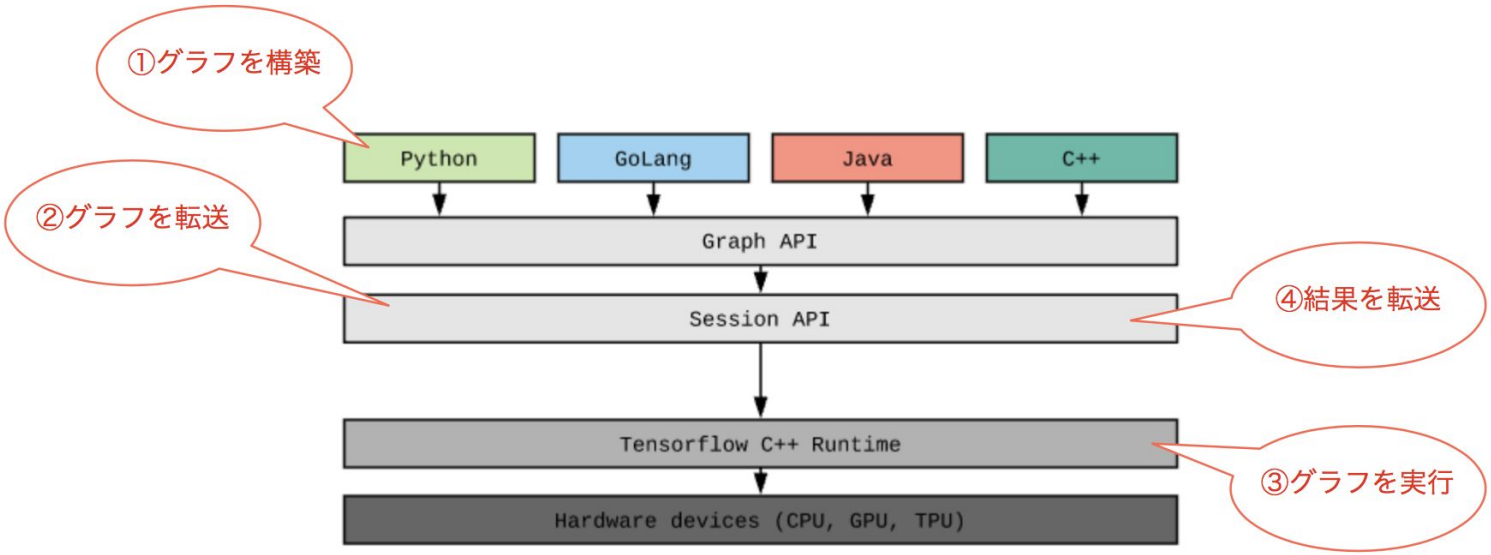
構築した計算グラフ（またはその一部）を実行するTensorflowのクラスのひとつ。

使用法：

TensorFlowの一般的な使い方は、まずpythonなどで計算グラフを構築し、それからtensorflowモジュールでtf.Sessionを実行する、という手順である。

低レイヤ：

このとき、実際に計算を行うにあたりC++ランタイム（コンパイル済みアプリケーション）を利用する。このランタイムへ結合(connection)することをsessionと呼んでいる。



TensorFlowの構築

セッションの事例

TensorFlowの計算グラフの実行は以下のクラスで提供される。

①tf.Session

②tf.InteractiveSession

①tf.Session：

計算グラフとは独立して定義する。

※run()はOperationの中身を取り出してくれる

※eval()はTensorの中身を取り出してくれる

tf.Session

```
a = tf.constant(2.0)
b = tf.constant(1.5)
mul = tf.multiply(a, b)

with tf.Session():
  print(mul.eval())
```

計算グラフ構築後に
中身にアクセス

```
# 3.0
```

TensorFlowの構築

セッションの種類

TensorFlowの計算グラフの実行は以下のクラスで提供される。

①tf.Session

②tf.InteractiveSession

②tf.InteractiveSession：

インタラクティブに操作するためのデフォルトセッション。

計算グラフを構築する前にインスタンスを作成する必要がある。

※ run()はOperationの中身を取り出してくれる

※ eval()はTensorの中身を取り出してくれる

tf.InteractiveSession

```
sess = tf.InteractiveSession()

a = tf.constant(2.0)
b = tf.constant(1.5)
mul = tf.multiply(a, b)
print(mul.eval())

sess.close()
```

計算グラフ構築中に
中身にアクセスできる

これ以降であれば、*.eval()できる

```
# 3.0
# Tensor("Mul_10:0", shape=(), dtype=float32) の中身
```

TensorFlowの構築

Tensorflow2.0の話

tf.Sessionがなくなります。

もともと TensorFlow1.5(2018年1月26日以降)からEagerモードが登場し、sessionなしでも計算できる仕様もありました。TensorFlow2.0からはこのEagerモードがデフォルトになります。

JupiterNotebook
で実行するとき
カーネルを再起動

Eagerモード

```
import tensorflow as tf
import tensorflow.contrib.eager as tfe

tfe.enable_eager_execution()

a = tf.constant(2.0)
b = tf.constant(1.5)
mul = tf.multiply(a, b)

print(mul)
print(mul.numpy())
```

Graphモード

```
import tensorflow as tf

a = tf.constant(2.0)
b = tf.constant(1.5)
mul = tf.multiply(a, b)


with tf.Session() as sess:
    sess.run(mul)
    print(mul)
    print(mul.eval())
```

TensorFlowの構築

Sessionの代わりは？

tf.Sessionの代わりにtf.functionを使うことになります。

https://github.com/tensorflow/community/blob/master/rfcs/20180918-functions-not-sessions-20.md

TensorFlow2.0では、@tf.function（デコレータ）を付与した関数がコンパイルされます。2.0では、EagerモードとGraphモードを混在させることもできます。

TF2.0のEagerモードとGraphモードを解説しているyoutube
https://www.youtube.com/watch?v=WTNH0tcscqo&feature=youtu.be&t=83

EagerExecution（TensorFlow1.x）

```
import tensorflow as tf
import tensorflow.contrib.eager as tfe

tfe.enable_eager_execution()

a = tf.Variable(1.0)
b = tf.Variable(1.0)

def f():
    a.assign(2.0)
    b.assign(3.0)
    return a + b
print(f().numpy())
```

5.0

tf.function（TensorFlow2.0)

```
a = tf.Variable(1.0)
b = tf.Variable(1.0)
@tf.function
def f():
  a.assign(2.0)
  b.assign(3.0)
  return a + b
print(f().numpy())
```

5.0

おまけ

Tensorflow公式サイト

https://www.tensorflow.org/community/roadmap
日本語訳してくれているサイト
http://tensorflow.classcat.com/category/keras/

Tensorflowコミュニティ

https://github.com/tensorflow/community/blob/master/rfcs/20180907-contrib-sunset.md

Tensorflow 2.0 情報

https://developers-jp.googleblog.com/2019/03/what-are-symbolic- and-imperative-apis-in-tensorflow-2-0.html?m=1
tensorflow 2.0 の紹介(日本語訳)
https://qiita.com/halhorn/items/09a64e98a02022e6ccc2

フレームワーク1_TensorFlow
完