آموزش زبان برنامه نویسی پایتون
بخش هفتم
تنسورفلو، ویرایش۱

مرجع اصلی

Krishna Rungta, *TensorFlow in 1 Day*, 2018.

https://www.guru99.com/tensor-tensorflow.html

محمدجواد فدائی‌اسلام

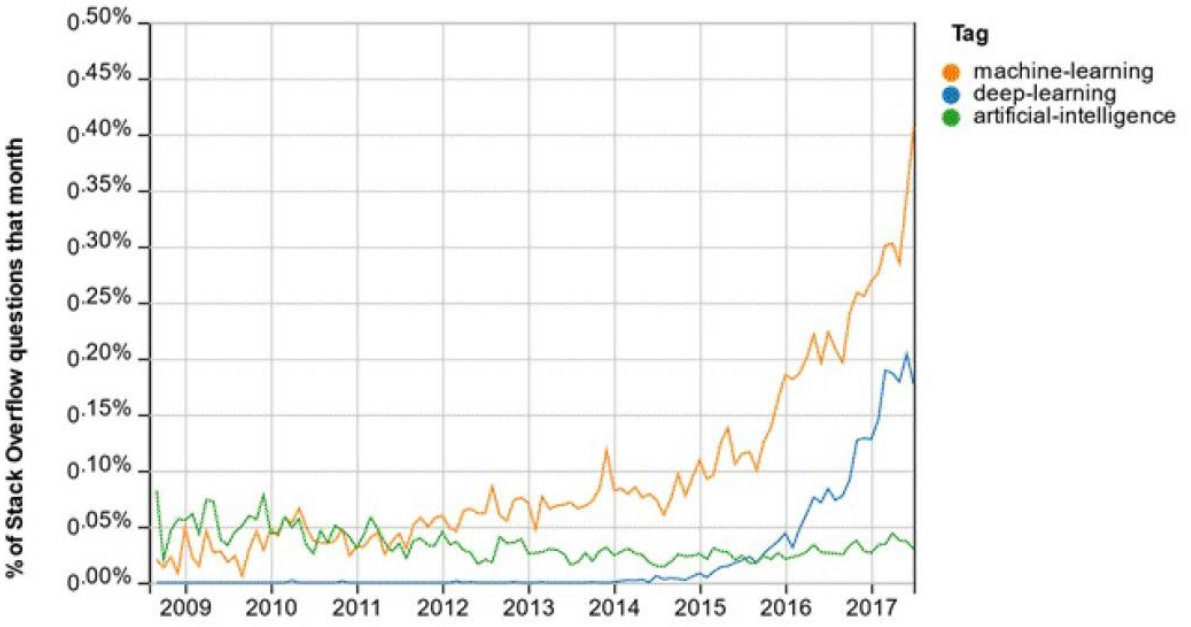
۲۱ مرداد ۱۴۰۰

باسمه تعالی

فهرست مطالب

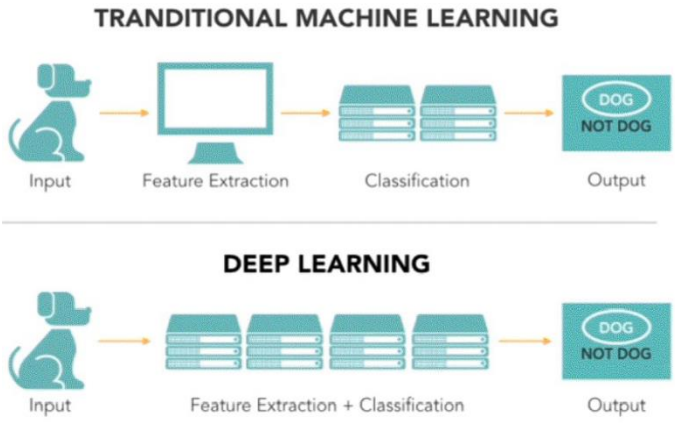
۱- یادگیری عمیق[۱]

یادگیری عمیق زیرمجموعه یادگیری ماشین است که سعی دارد شبکه عصبی موجود در مغز را تقلید کند. از آن جهت که در آن، لایه‌های متعدد برهم نهاده شده است، عمیق نام گرفته است. نمودار زیر پرس و جوهای مطرح شده درباره آن را در stackoverflow نشان می‌دهد که به نوعی بیان‌کننده اهمیت آن است.



شکل ۱- پرس و جوهای ماهانه درباره یادگیری ماشین، یادگیری عمیق و هوش مصنوعی در stackoverflow

در روش‌های کلاسیک یادگیری مرحله‌ای به عنوان استخراج ویژگی وجود دارد که در آن ویژگی‌های موثر به صورت دستی استخراج می‌شود. اما در یادگیری عمیق فرآیند استخراج ویژگی به صورت خودکار انجام می‌شود.



شکل ۲- مراحل یادگیری در یادگیری عمیق و یادگیری کلاسیک

۱ - deep learning

	Machine Learning	Deep Learning
Data Dependencies	Excellent performances on a small/medium dataset	Excellent performance on a big dataset
Hardware dependencies	Work on a low-end machine.	Requires powerful machine, preferably with GPU: DL performs a significant amount of matrix multiplication
Feature engineering	Need to understand the features that represent the data	No need to understand the best feature that represents the data
Execution time	From few minutes to hours	Up to weeks. Neural Network needs to compute a significant number of weights
Interpretability	Some algorithms are easy to interpret (logistic, decision tree), some are almost impossible (SVM, XGBoost)	Difficult to impossible

شکل ۳- مقایسه یادگیری عمیق و یادگیری کلاسیک

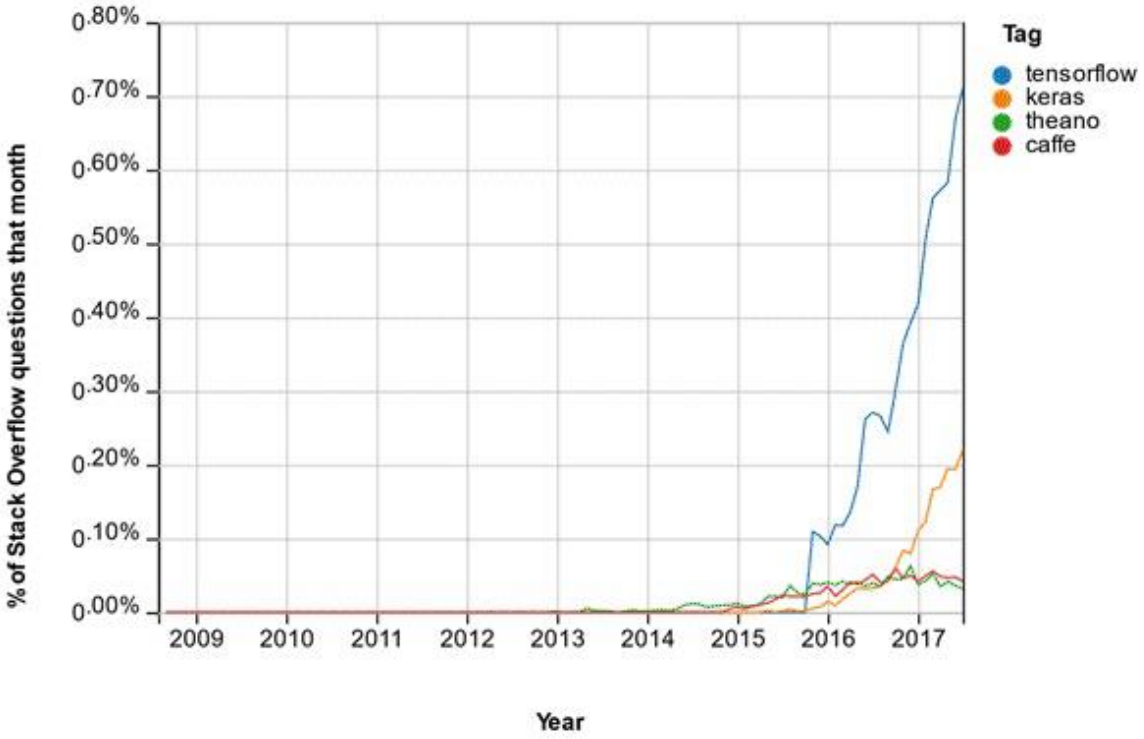
۲ - تنسورفلو[۲]

در حال حاضر تنسورفلو مشهورترین کتابخانه یادگیری عمیق در جهان و محصول شرکت گوگل است. این کتابخانه به نحوی ساخته شده است که قابلیت اجرا بر روی چند پردازنده، پردازنده گرافیکی و سیستم عامل موبایل دارد و برای آن چندین مبدل[۳]به زبان‌های دیگر مثل Python، C++ یا Java وجود دارد. تنسورفلو در سال ۲۰۱۵ ابداع شد، اما نسخه پایدار آن در سال ۲۰۱۷ ارایه شد. آن یک نرم‌افزار متن‌باز است. در نمودار زیر (شکل ۴) میزان محبوبیت چارچوب‌های برنامه‌نویسی مشابه با آن از لحاظ میزان پرس و جو نشان داده شده است.

ورودی این نرم‌افزار، آرایه‌های چند بعدی است که تنسور[٤]نام دارند. این تنسورها از میان لیستی از عملگرها عبور می‌کنند و از سوی دیگر خارج می‌شوند. در یک تنسور، یک نوع داده وجود دارد و اندازه آن هم (ابعاد) تعیین شده است. تنسور به عنوان داده ورودی، وارد شبکه می‌شود و یا حاصل یک محاسبه است. تمام عملیات‌ها که بر روی تنسور انجام می‌شود از داخل یک گراف هدایت می‌شود.

---

۲ - TensorFlow
۳ - wrapper
٤ - Tensor



شکل ۴- میزان محبوبیت تنسورفلو و چارچوب های مشابه

گراف مجموعه عملیاتی که پشت سرهم انجام می‌شود را نشان می‌دهد. هر عمل با یک گره[۵]نشان داده می‌شود. لبه‌ها در نقش تنسورها هستند. گراف، طرح کلی عملیات بر روی داده‌ها را نشان می‌دهد، اما هنوز مقداری محاسبه نشده است.

۳- برنامه ساده تنسورفلو

```
##Define the variable
X_1 = tf.constant([1.0, 2.0],tf.float32)
X_2 = tf.constant([3.0, 4.0],tf.float32)
##Define the computation
multiply = tf.multiply(X_1, X_2)
##Execute the operation
sess = tf.Session()
result = sess.run(multiply)
print(result)
sess.close()
```

نمای ۵- برنامه ساده TensorFlow

برنامه بالا از سه قسمت ایجاد تنسور، تغریف عملگر و اجرا تشکیل شده است.

۵ - op node

```
[3. 8.]
>>>
```
نمای ۶- خروجی برنامه

۴- تعریف تنسور

یک تنسور از سه جز تشکیل شده است:

۱- نام یکتا، ۲- بُعد[۶] و ۳- نوع داده[۷]

برای انتقال اطلاعات باید از تنسور استفاده نمود. همان‌طور که در ذیل آمده است، چهار نوع اصلی تنسور در تنسورفلو وجود دارد که درباره برخی از آنها توضیحاتی ارایه می‌شود.

- tf.Variable
- tf.constant
- tf.placeholder
- tf.SparseTensor

۵- تعریف تنسور ثابت

در شکل (۷) آرگومان‌های سه‌گانه تنسوری از نوع constant توضیح داده شده است. همان‌طور که مطرح شد، این تنها تعریف داده است و در این مرحله، عملیاتی اجرا نمی‌شود.

```
tf.constant(value, dtype, name = "")

arguments
        - 'value':  Value of n dimension to define the tensor. Optional
        - 'dtype':  Define the type of data, for example:
                -  'tf.string': String variable
                -  'tf.float32': Float variable
                -  'tf.int16': Integer variable
        -'name':  Name of the tensor. Optional. By default, 'Const_1:0'
```
نمای ۷- اجزای تنسور

در نمای ۸ تنسور از نوع constant با نوع داده صحیحی ۱۶ بیتی و نام my_constant تعریف شده است.

```
r2 = tf.constant( [[1, 4], [2, 5], [3, 6]], tf.int16, name = "my_constant")
print(r2)
```
نمای ۸- تعریف و نمایش تنسور

۶ - dimension or shape
۷ - data type or dtype

خروجی نمای (۸) در ذیل نمایش داده شده است. دستور print نوع تنسور را نمایش می‌دهد و ارتباطی با مقدار موجود در آن ندارد.

```
Tensor("my_constant:0", shape=(3, 2), dtype=int16)
```
نمای ۹- خروجی کد نمای ۸

- دستور ی برای ایجاد یک تنسور که دارای مقدار اولیه ۰ یا ۱ است.

```
tf.zeros()
tf.ones()
```

آرگومان دو دستور بالا باید ابعاد تنسور باشد. مثال:

```
a = tf.zeros(10)
b = tf.ones([10, 10])
```

۶- نوع داده[۸] در تنسور

نوع داده تمام مقادیر یک تنسور یکسان است. اگر نوع داده تنسور ذکر نشود تنسور به صورت خودکار مشابه‌ترین توع داده به مقادیر را بر می‌گزیند.

- نمایش نوع داده تنسور با نام m_shape

```
print(m_shape.dtype)
```

- تغییر نوع داده تنسور a از اعشاری به صحیح

```
a = tf.cast(a, dtype=tf.int32)
```

۷- عملگرهای ساده

عملگرهای ساده در تنسورفلو طراحی شده اند که در ذیل نام آنها آمده است. مثالی از کاربرد آنها در شکل (۱۰) آمده است.

```
tf.add(a, b)
tf.substract(a, b)
tf.multiply(a, b)
tf.div(a, b)
tf.pow(a, b)
tf.exp(a)
tf.sqrt(a)
```

۸ - types of data

```
# Add
tensor_a = tf.constant([[1,2]], dtype = tf.int32)
tensor_b = tf.constant([[3, 4]], dtype = tf.int32)
tensor_add = tf.add(tensor_a, tensor_b)
```

نمای ۱۰- تعریف یک عملگر

باید توجه نمود که عملیاتی انجام نمی‌شود، صرفا عملگری تعریف می‌شود. ضرب و تقسیم به صورت درایه به درایه انجام می‌شود و ابعاد ماتریس خروجی با ورودی یکی است.

۸- تعریف تنسور متغیر

برای ایجاد چنین متغیری باید از تابع tf.get_variable() استفاده نمود. در شکل (۱۱) نحوه تعریف آن آمده است.

```
tf.get_variable(name = "", values, dtype, initializer)

argument
        - 'name = """': Name of the variable
        - 'values': Dimension of the tensor
        - 'dtype': Type of data. Optional
        - 'initializer': How to initialize the tensor. Optional
        If initializer is specified, there is no need to include the 'values' as the
        shape of 'initializer' is used.
```

نمای ۱۱ – تعریف تنسور متغیر

- تعریف تنسور متغیر با ابعاد 2×1

```
var = tf.get_variable("var", [1, 2])
```

- تعریف تنسور متغیر با مقدار اولیه از یک تنسور ثابت

```
tensor_const = tf.constant([[10, 20], [30, 40]])
var2 = tf.get_variable("var2", dtype=tf.int32, initializer=tensor_const)
```

۹- تعریف تنسور Placeholder

ساختار تعریف آن به صورت زیر است:

```
tf.placeholder(dtype,shape=None,name=None )

    arguments:
    - 'dtype': Type of data
    - 'shape': dimension of the placeholder. Optional. By default,
    shape of the data
    - 'name': Name of the placeholder. Optional
```

نمای ۱۲– تعریف تنسور PlaceHolder

این تنسور با هدف تغذیه اطلاعات درون تنسورها استفاده می‌شود. تنها در یک نشست که توضیح آن خواهد آمد، اطلاعات در آن تزریق می‌شود و برای آن باید از تابع feed_dict استفاده نمود.

Session -۱۰

تنسورفلو تقریبا با سه جزء اصلی کار می‌کند: ۱- Tensor، ۲- Graph، ۳- Session

در session تمام عملیاتی که در گراف تعریف شده است اجرا می‌شود. گراف و session از هم مستقل هستند و می‌توان session را اجرا نمود تا مقادیری به دست آورد و آن را در موارد لزوم به کار برد. در الگوریتم (۵) یک برنامه ساده به تنسورفلو نوشته شده است. از تابع eval() هم برای اجرای یک نشست می‌توان استفاده کرد. عملکرد این تابع مشابه با run() است.

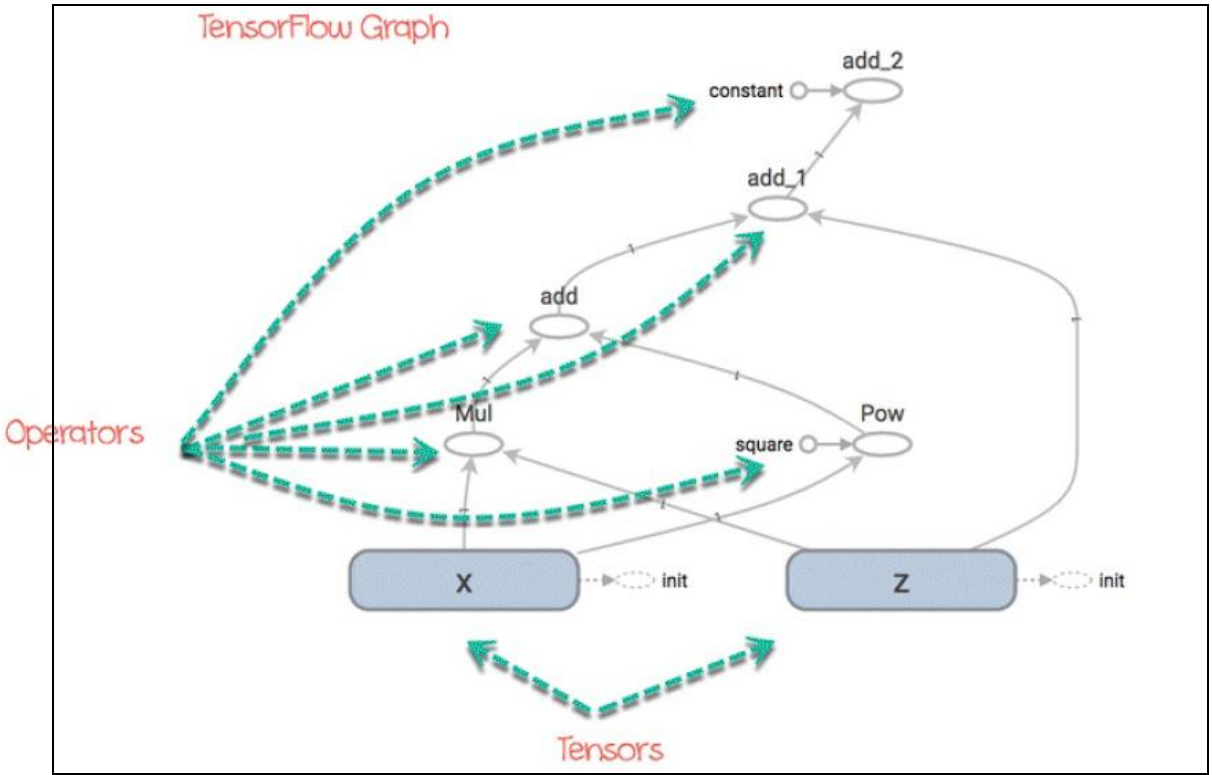
Graph -۱۱

گراف از گره و یال تشکیل شده است. گره نماینده یک عملیات است و یال نماینده تنسور یا داده. در حقیقت گراف یک جریان اطلاعات[۹]را نشان می‌دهد. فرض کنید تابع زیر بخواهد محاسبه شود.

$$f(x.z) = xz + x^2 + z + 5$$

تنسورفلو جریانی مشابه با گراف زیر جهت محاسبه آن ایجاد می‌کند.

۹- dataflow



شکل ۱۳ – گراف جریان داده ایجاد شده توسط TensorFlow

کد متناظر با رابطه بالا در نمای (۱۴) آمده است.

```
# ---------------------------------Graph
# Initialize a variable called x with a constant value of 5
x = tf.get_variable("x", dtype=tf.int32, initializer=tf.constant([5]))

# Initialize a variable called z with a constant value of 6
z = tf.get_variable("z", dtype=tf.int32, initializer=tf.constant([6]))

# Initialize a constant tensor called c with a constant value of 5
c = tf.constant([5], name = "constant")

# Initialize a constant tensor called square with a constant value of 2
square = tf.constant([2], name = "square")

# Construct the operator
f = tf.multiply(x, z) + tf.pow(x, square) + z + c
# ---------------------------------Session
init = tf.global_variables_initializer() # prepare to initialize all variables
with tf.Session() as sess:
        init.run()   # Initialize x and y
        result = f.eval()
        print(result)
```

شکل ۱۴ – گراف جریان داده برای رابطه بالا و محاسبه آن

۱۲- پیاده‌سازی شبکه کانولوشنی