Introduction to TensorFlow

Alejandro Solano - EuroPython 2017
cat
input

\[
\text{cat} \quad \quad + \quad \quad \sin \exp \log \text{cat}
\]
Deep Learning
What is TensorFlow?

- TensorFlow is an open-source library for Deep Learning.
- Developed by the Google Brain team and released in November 2015.
- Version 1.0.0 was launched in February 2017.
Installation
Install TensorFlow (Linux and Mac OS)

- Download Anaconda
- Create an environment with all must-have libraries.

```
$ conda create -n tensorflow python=3.5
$ source activate tensorflow
$ conda install pandas matplotlib jupyter notebook scipy scikit
$ pip install tensorflow
```
Install TensorFlow (Windows)

- Download Anaconda
- Create an environment with all must-have libraries.

$ conda create -n tensorflow python=3.5
$ activate tensorflow
$ conda install pandas matplotlib jupyter notebook scipy scikit
$ pip install tensorflow
Concepts
cat
The diagram illustrates a model prediction process. The input is fed into the model, which predicts whether the target is a cat or not. If the prediction is not a cat, it is labeled as 'non-cat'. The TensorFlow logo indicates the use of TensorFlow for this process.
non-cat

MODEL

cat

COST

TensorFlow
non-cat

MODEL

OPTIMIZER

cost

tensorFlow

Graph

cat
Graph

MODEL

OPTIMIZER

COST
Graph

MODEL

OPTIMIZER

COST

placeholder

placeholder

TensorFlow
Graph

- **Placeholders**: gates where we introduce example
- **Model**: makes predictions. Set of **variables** and operations
- **Cost function**: function that computes the model error
- **Optimizer**: algorithm that optimizes the variables so the cost would be zero
Session: Graph + Data

inputs → MODEL → COST → OPTIMIZER → targets
Graph, Data and Session

- **Graph**: Layout of the prediction and learning process. It does not include data.
- **Data**: Examples that will train the neural network. It consists on two kinds: inputs and targets.
- **Session**: Where everything takes place. Here is where we feed the graph with data.
Session: Graph + Data

MODEL

OPTIMIZER

COST

TensorFlow
Session: Graph + Data
Session: Graph + Data
Session: Graph + Data

MODEL

COST

train

non-cat

100

OPTIMIZER

TensorFlow
Hello world!
Hello world!: Sum of two integers

import tensorflow as tf
Hello world!: Sum of two integers
Hello world!: Sum of two integers

##### GRAPH ######

```python
a = tf.placeholder(tf.int32)
b = tf.placeholder(tf.int32)
sum_graph = tf.add(a, b)
```

##### DATA ######

```python
num1 = 3
num2 = 8
```
Hello world!: Sum of two integers

```python
### SESSION ###
with tf.Session() as sess:
    sum_outcome = sess.run(sum_graph, feed_dict={
        a: num1,
        b: num2
    })
```
In [5]:

```python
with tf.Session() as sess:
    sum_output = sess.run(sum_graph, feed_dict={
        'a': num1,
        'b': num2
    })

print("The sum of {} and {} is {}".format(num1, num2, sum_output))

The sum of 3 and 8 is 11
```
Regression
TensorFlow for Regression: learning how to sum

- Mission: learn how to sum using 10,000 examples.

\[ x_1 + x_2 = y \]
Mission: learn how to sum using 10,000 examples.

\[ x_1 \ ? \ x_2 = y \]
TensorFlow for Regression: learning how to sum

- Mission: learn how to sum using 10,000 examples.
TensorFlow for Regression: learning how to sum

- Mission: learn how to sum using 10,000 examples.

\[ x_1 + x_2 = y \]

250 m
TensorFlow for Regression: learning how to sum

- Mission: learn how to sum using 10,000 examples.
TensorFlow for Regression: learning how to sum

- Mission: learn how to sum using 10,000 examples.

  \[ x_1 + x_2 = y \]

- We assume the relationship between \( x \) and \( y \) is a linear function.

  \[ x \cdot W + b = y \]
TensorFlow for Regression: learning how to sum

- Mission: learn how to sum using 10,000 examples.

\[ x_1 + x_2 = y \]

- We assume the relationship between \( x \) and \( y \) is a linear function.

\[ x \cdot W + b = y \]

variables to be learned
Neural Network

\[ x \cdot W_1 + b_1 = y \]
Neural Network

\[ x \cdot W_1 + b_1 = y \]

\[ (x \cdot W_1 + b_1) \cdot W_2 + b_2 = y \]
Neural Network

\[ x \cdot W_1 + b_1 = y \]

\[ (x \cdot W_1 + b_1) \cdot W_2 + b_2 = y \]

\[ (((x \cdot W_1 + b_1) \cdot W_2 + b_2) \cdot W_3 + b_3) = y \]
Neural Network

\[ x \cdot W_1 + b_1 = y \]

\[ \sigma(x \cdot W_1 + b_1) \cdot W_2 + b_2 = y \]

\[ \tanh(\sigma(x \cdot W_1 + b_1) \cdot W_2 + b_2) \cdot W_3 + b_3 = y \]
# PLACEHOLDERS

```python
x = tf.placeholder(tf.float32, [None, 2])
y = tf.placeholder(tf.float32, [None, 1])
```
# PLACEHOLDERS

```python
x = tf.placeholder(tf.float32, [None, 2])
y = tf.placeholder(tf.float32, [None, 1])
```

(we don’t know how many examples we’ll have, but we do know that each one of them has 2 numbers as input and 1 as target)
TensorFlow for Regression: learning how to sum

# MODEL
W = tf.Variable(tf.truncated_normal([2, 1], stddev=0.05))
b = tf.Variable(tf.random_normal([1]))

output = tf.add(tf.matmul(x, W), b)
TensorFlow for Regression: learning how to sum
Cost (loss) function

\[ x \cdot W + b = y \]

prediction

target
Cost (loss) function

\[ y - (x \cdot W + b) \]
Cost (loss) function

$$\left[ y - (x \cdot W + b) \right]^2$$
Cost (loss) function

\[ \sum [ y_i - (x_i \cdot W + b) ]^2 \]
TensorFlow for Regression: learning how to sum

\[ \text{cost} = \text{tf.reduce_sum}(\text{tf.square}(\text{output} - y)) \]
Gradient Descent

cost function

cost = f(w1, w2, b)
Gradient Descent

cost function

cost = f(w1, w2, b)
Gradient Descent

cost function

cost = f(w1, w2, b)
Gradient Descent

cost function

cost = f(w1, w2, b)
TensorFlow for Regression: learning how to sum

```python
optimizer = tf.train.GradientDescentOptimizer(Learning_rate=0.00001)

optimizer = optimizer.minimize(cost)
```
TensorFlow for Regression: learning how to sum
Data split

data
Data split

Data

train data  test data
TensorFlow for Regression: learning how to sum

```python
from helper import get_data, split_data

# DATA
inputs, targets = get_data(max_int=10, size=10000)

# split train and test data
train_inputs, test_inputs, train_targets, test_targets = split_data(inputs, targets)
```
TensorFlow for Regression: learning how to sum
with tf.Session() as sess:
    sess.run(tf.global_variables_initializer())

    for epoch in range(epochs):
        sess.run(optimizer, feed_dict={
            x: train_inputs,
            y: train_targets
        })
Testing Accuracy: 0.9568796753883362

The sum of 5 plus 7 is 11.991448402404785

The weights are: [[ 0.88418758]  
[ 0.8903569 ]]  
and the bias is: [ 1.33801162]
Classification
MODEL

\[
[0.175, 0.825]
\]
TensorFlow for Classification

- Mission: learn if the sum of two numbers is higher than 10.

\[
\begin{align*}
\text{if } (x_1 + x_2 > 10) & \text{ then } y = [0; 1] \\
\text{else } y &= [1; 0]
\end{align*}
\]
TensorFlow for Classification

- Mission: learn if the sum of two numbers is higher than 10.

\[ x_1 \ ?? \ x_2 = y \]
TensorFlow for Classification

- Mission: learn if the sum of two numbers is higher than 10
  \[ x_1 + x_2 = y \]

- More complexity: we add a new layer
Neural Networks: intuition

First layers extract the more basic features, while the next ones will work from this information.

computes the sum

classifies the sum
Neural Networks: intuition

First layers extract the more basic features, while the next ones will work from this information.
Testing Accuracy: 1.0

Final test (5 + 3), (7 + 6), (10 + 10):
[[ 8.93008471e-01  1.06991500e-01]
 [ 1.24206737e-01  8.75793278e-01]
 [ 4.71629581e-04  9.99528408e-01]]

Hidden layer weights and bias:
[[ -1.64704931]
 [ -1.64078069]]
[ 0.19933932]
Output layer weights and bias:
[[ 1.21974468 -1.26005733]]
[-0.2669307]
To know more...

Deep learning

- Neural Networks and Deep Learning - Michael Nielsen
- Stanford’s CS231n - Andrej Karpathy

Tensorflow

- Tensorflow Tutorials - Hvass Laboratories
- Deep Learning Foundations Nanodegree - Udacity
To start to know more...

Basics

- Intro to Data Science - Udacity
- Intro to Machine Learning - Udacity