Deep Learning and Temporal Data Processing

Linear Regression in TensorFlow

Andrea Palazzi

July 9, 2017

University of Modena and Reggio Emilia

Agenda

Linear Regression

Goal

TensorFlow fully unleashes its power when used for deep learning applications. However, this framework can be used to implement any gradient-based learning pipeline.

The goal of this first practice session is implementing a simple **linear regression** model in TensorFlow[1].

Data

For this practice we'll use a small dataset which puts in relation the weight of brain and body for a number of mammal species.

The whole dataset can be found here:
`http://people.sc.fsu.edu/~jburkardt/datasets/regression/x01.txt`

The goal is to fit a simple linear regression model (*i.e.* a line) to this data.

Starter Pack

I already downloaded the data for you. You'll find it in `data/brain_body_weight.txt`.

Also, a function to load the data into python script is available in `lab_utils.py`.

Outline

Here's the outline of what you're expected to do:

1. Load data using `lab_utils/get_brain_body_data`
2. Define appropriate placeholders using `tf.placeholder`
3. Define weight and bias variables using `tf.Variable`
4. Define 1-d linear regression model $y_{pred} = xw + b$
5. Define an appropriate objective function, e.g. $(y_{pred} - y_{true})^2$
6. Create an Optimizer (e.g. `tf.train.AdamOptimizer`)
7. Define a train iteration as one step of loss minimization
8. Loop train iteration until convergence

Template i

```
# Read data
body_weight, brain_weight = # todo
n_samples = len(body_weight)

# Define placeholders (1-d)
x, y = # todo

# Define variables
w, b = # todo
```

Template ii

```
# Linear regression model
y_pred = # todo

# Define objective function
loss = # todo

# Define optimizer
optimizer = # todo

# Define one training iteration
train_step = # todo
```

Template iii

```
with tf.Session() as sess:

    # Initialize all variables
    sess.run(tf.global_variables_initializer())

    for i in range(n_epochs):
        total_loss = 0
        for bo_w, br_w in zip(body_weight, brain_weight):
            _, l = sess.run([train_step, loss],
                             feed_dict={x: bo_w, y: br_w})
            total_loss += l
        print('Epoch {0}: {1}'.format(i, total_loss / n_samples))
```

References

References i

[1] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng.
TensorFlow: Large-scale machine learning on heterogeneous systems, 2015.
Software available from tensorflow.org.