Applied Deep Learning

TensorFlow

Oct 6th, 2016

YUN-NUNG (VIVIAN) CHEN

National Taiwan University

WWW.CSIE.NTU.EDU.TW/~YVCHEN/F105-ADL

Slide credit from Stanford class

Deep Learning Toolkit

Torch

Caffee

Theano (Keras, Lasagne)

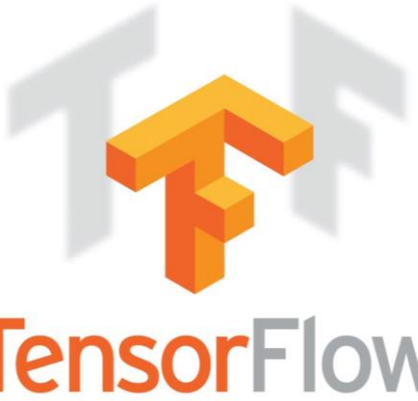
Tensorflow

CNTK

CuDNN

Mxnet

etc.



Tool Design

- Model specification
  - Configuration file
    - caffee, CNTK, etc
  - Programmatic generation
    - Torch, Theano, TensorFlow
- High-level language
  - Lua
    - Torch
  - Python
    - Theano, TensorFlow

Introduction

TensorFlow is an open source software library for machine intelligence developed by Google

- Provides primitives for *defining functions on tensors* and *automatically computing their derivatives*

Prerequisite: Python 2.7/3.3+ & numpy

What is a Tensor?

Definition

- Tensors are multilinear maps from vector spaces to the real numbers → n-dimensional arrays

Example

- Scalar $f : \mathbb{R} \to \mathbb{R}, f(e_1) = c$
- Vector $f : \mathbb{R}^n \to \mathbb{R}, f(e_i) = v_i$
- Matrix $f : \mathbb{R}^n \to \mathbb{R}^m, f(e_i, e_j) = M_{ij}$

Deep learning process is flows of tensors → a sequence of tensor operations

Installation

Installation

Requirements

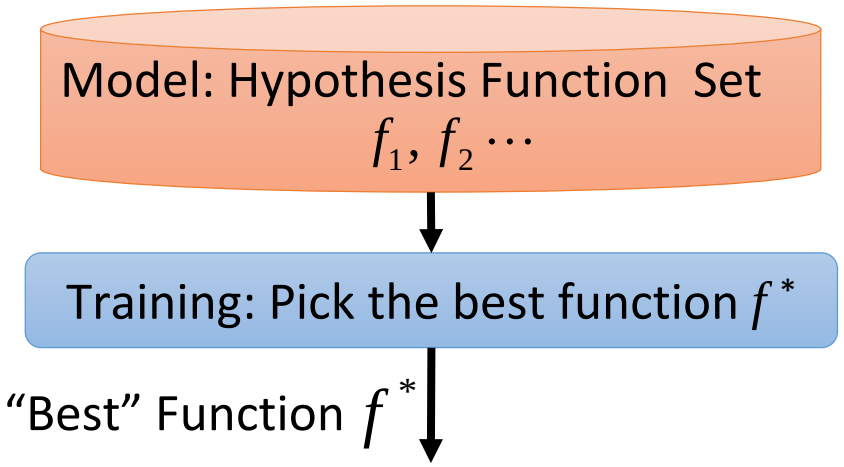
- Python API: Python 2.7 or Python 3.3+
- GPU: Cuda Toolkit >= 7.0 and cuDNN >= v3

Suggested procedure – virtualenv installation

1. Install pip & virtualenv
2. Create a virtualenv environment
3. Activate the virtualenv environment
4. Install tensorflow in the environment
5. Activate the environment every time you want to use TensorFlow

Review

Deep Learning Framework



Q1. What is the model? **Model Architecture**

Q2. What does a "good" function mean? **Loss Function Design**

Q3. How do we pick the "best" function? **Optimization**

After defining the model and loss function, TensorFlow automatically computes the gradient for optimization

Sample Program

```
import tensorflow as tf
import numpy as np

x_data = np.random.rand(100).astype(np.float32)
y_data = x_data * 0.1 + 0.3

W = tf.Variable(tf.random_uniform([1], -1.0, 1.0))
b = tf.Variable(tf.zeros([1]))
y = W * x_data + b

loss = tf.reduce_mean(tf.square(y - y_data))
optimizer = tf.train.GradientDescentOptimizer(0.5)
train = optimizer.minimize(loss)

init = tf.initialize_all_variables()

sess = tf.Session()
sess.run(init)

for step in range(201):
    sess.run(train)
    if step % 20 == 0:
        print(step, sess.run(W), sess.run(b))
```

Import the APIs

Create 100 phony x, y data points in NumPy, y = x * 0.1 + 0.3

Try to find values for W and b that compute y_data = W * x_data + b (W should be 0.1 and b 0.3)

Minimize the mean squared errors.

Initialize the variables.

Launch the graph.

Fit the line.

→ Learns best fit is W: [0.1], b: [0.3]

Basic Usage

Represents computations as graphs

Executes graphs in the context of `Sessions`

Represents data as tensors

Maintains state with `Variables`

Uses feeds and fetches to get data into and out of any operations

TensorFlow programs are usually structured into a *construction phase*, that assembles a graph, and an *execution phase* that uses a session to execute ops in the graph