tensorflow-crash (course)

Insight Data Science
December 7, 2017
New York City

www.github.com/adrian-soto/tensorflow-crash/

TensorFlow

From Wikipedia, the free encyclopedia

TensorFlow is an open-source software library for dataflow programming across a range of tasks. It is a symbolic math library, and also used for machine learning applications such as neural networks.[3]

TODAY: run a couple of basic examples that can serve as a starting point for your future projects!

We will be working off of code in the Github repo

www.github.com/adrian-soto/tensorflow-crash/

Please clone it into your machine now!

Companies using TensorFlow

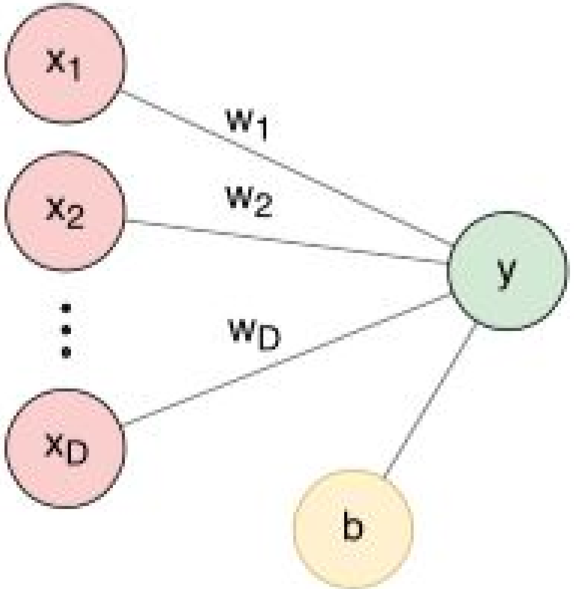
airbnb
NVIDIA.
UBER
SAP
kakao
DeepMind
Dropbox
ebay
Google
intel
Coca-Cola
mi
ZTE
QUALCOMM

So let’s install Tensorflow!

```
1. Create new conda environment
  $ conda create -n tensorflow python= 2.7 (or 3.6)

2. Activate environment
  $ source activate tensorflow

3. Ensure iPython is installed IN THE ENVIRONMENT
  $ conda install ipython
  $ pip install jupyter (if want to use notebooks)

4. Install tensorflow
  $ pip install --ignore-installed --upgrade
https://storage.googleapis.com/tensorflow/mac/cpu/tensorflow-1.4.0-py2-none-any.whl

5. Test Tensorflow
  $ ipython
  $ # in iPython
  $ import tensorflow as tf
  $ hello = tf.constant('Hello, Tensorflow!')
  $ sess = tf.Session()
  $ print(sess.run(hello))
If output is
  Hello, Tensorflow!
the installation and execution via iPython succeeded :)
```

… or, even easier,run the `install_tf.sh` script in the repo

What is a neuron?



Activation:

$$a = \sum_i w_i x_i + b$$

Activity:

$$y = f(a)$$

$f(a)$ typically non-linear

- $f(a) = 1/(1 + e^{-a})$ (logistic sigmoid)
- $f(a) = \tanh(a)$ (hyperbolic sigmoid)
- $f(a) = \max(0, a)$ (rectified linear unit or ReLu)

How to train a neuron (1): online learning

For each data point in training set:

1. Compute activation $$a = \sum_i w_i x_i$$
2. Evaluate activity $$y(a) = \frac{1}{1 + e^{-a}}$$
3. Calculate error $$e = t - y$$
4. Adjust weights in the direction that reduces error $$\Delta w_i = \eta e x_i \qquad \eta\text{: learning rate}$$

MacKay, Information Theory, Inference and Learning algorithms, pages 475-476

How to train a neuron (2): batch learning

Batch learning for the single neuron classifier

For each input/target pair $(\mathbf{x}^{(n)}, t^{(n)})$ $(n = 1, \ldots, N)$, compute $y^{(n)} = y(\mathbf{x}^{(n)}; \mathbf{w})$, where

$$y(\mathbf{x}; \mathbf{w}) = \frac{1}{1 + \exp(-\sum_i w_i x_i)}, \tag{39.18}$$

define $e^{(n)} = t^{(n)} - y^{(n)}$, and compute for each weight w_i

$$g_i^{(n)} = -e^{(n)} x_i^{(n)}. \tag{39.19}$$

Then let

$$\Delta w_i = -\eta \sum_n g_i^{(n)}. \tag{39.20}$$

MacKay, Information Theory, Inference and Learning algorithms, page 476

How to train a neuron (3): minibatch learning

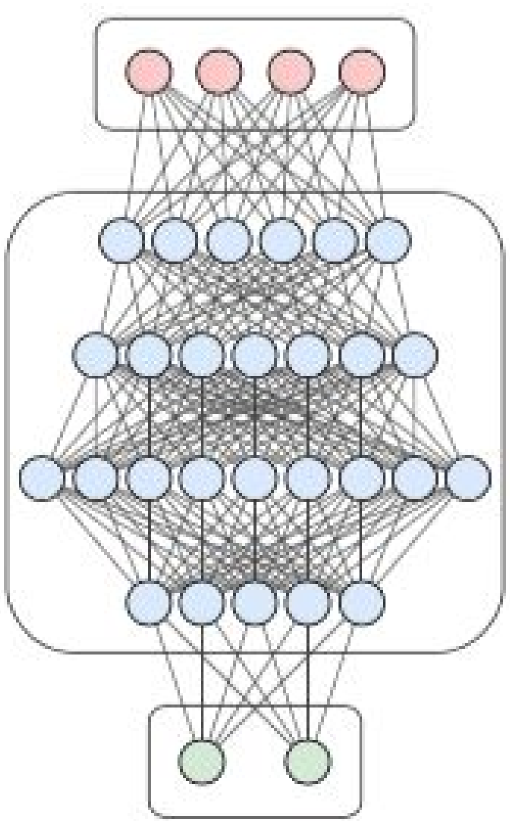
Intermediate option to online learning and batch learning

1. Divide data set into *minibatches*

2. Loop over a prefixed number of training *epochs*

    i. Evaluate activity, calculate error within the batch
    ii. Update weights accordingly for that batch
    iii. Move to next batch

An example of a deep neural network



input layer: $\vec{x}$

1st hidden: $\vec{z}^{(1)} = a(w^{(1)}\vec{x} + \vec{b}^{(1)})$

2nd hidden: $\vec{z}^{(2)} = a(w^{(2)}\vec{z}^{(1)} + \vec{b}^{(2)})$

3rd hidden: $\vec{z}^{(3)} = a(w^{(3)}\vec{z}^{(2)} + \vec{b}^{(3)})$

4th hidden: $\vec{z}^{(4)} = a(w^{(4)}\vec{z}^{(3)} + \vec{b}^{(4)})$

output layer: $\vec{y} = w^{(\text{out})}\vec{z}^{(4)} + \vec{b}^{(\text{out})}$

One of my students, Robert, asked:

> Maybe I'm missing something fundamental, but supervised neural networks seem equivalent to fitting a pre-defined function to some given data, then extrapolating – what's the difference?

I agree with Robert. The supervised neural networks we have studied so far are simply parameterized nonlinear functions which can be fitted to data. Hopefully you will agree with another comment that Robert made:

> Unsupervised networks seem much more interesting than their supervised counterparts. I'm amazed that it works!

MacKay, Information Theory, Inference and Learning algorithms, page 484

Unfortunately, today we'll be focusing on supervised learning…
but tomorrow starts your exploration!