A Brief Introduction to TensorFlow

Dylan Slack

Agenda

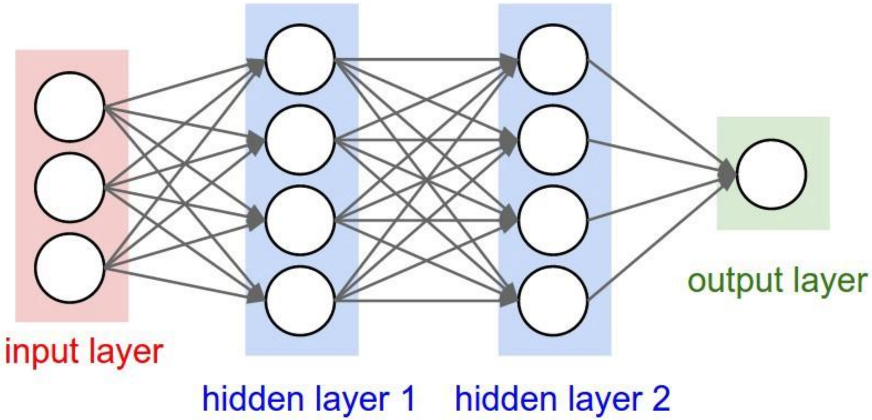
- Introduce the TensorFlow computing paradigm
- Get to some coding examples!
  - Please pull up: https://github.com/dylan-slack/tf_tutorial

Why TensorFlow (TF)?

- It gives us the tools to build, train, and evaluate machine/deep learning models
- It's one of the most popular, widely used tools to do this
- Other tools include, pytorch, caffe, torch, mxnet, etc.

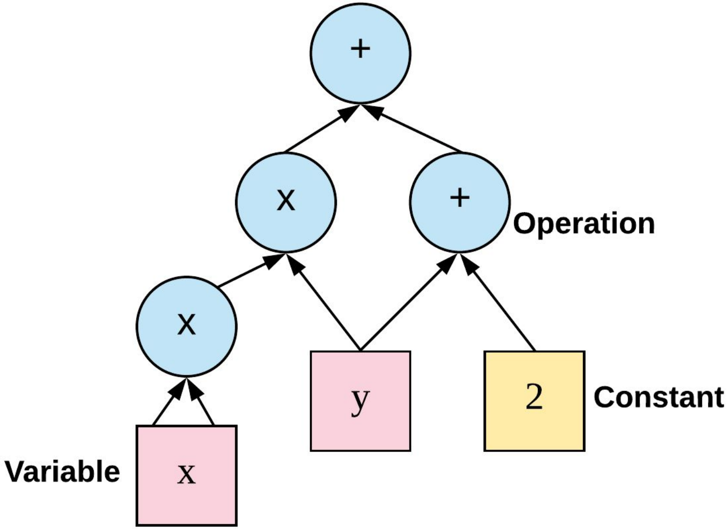
What do we mean by “Deep Learning?”

- Models such as neural networks that use many parameters to approximate complex, non-linear functions



Introducing the TF Computing Paradigm

- TensorFlow requires that you first set up a *graph* that defines your computation



TF Computing

- Once you’ve defined a graph, you can call a *session* to execute operations on the *graph*
- We evaluate *tensors* on the graph
    - *Tensors are* n-dimension arrays
- *Sessions* control the environment where you evaluate tensors

Let’s look at some code

- Check out slide 7 in the github repo

TensorFlow Data Types

- Constants: tf.constant(..)
    - Constants are hardcoded into the defintion of the graph
    - They making loading a graph with many constants a costly operation

TensorFlow Data Types

- Variables: tf.Variable(...)
  - Variables are… variables, they maintain the state of the graph across calls to run
  - Unlike constants they must be initialized

More On Sessions

- Each session has its own copy of variables
- Variables can hold different values across different sessions and that’s ok!

TensorFlow Data Types

- Placeholders: tf.placeholder(...)
  - Assemble the graph without the values needed for computation
  - Include the values through a *feed_dict* at run time

Optimizers

- Optimizers are really where the “magic” starts to happen in TF
- They allow us to compute and apply gradients to our graphs in order to optimize variables
- TF has many different types of optimizers -- here is a good post on why there are so many: https://stackoverflow.com/questions/36162180/gradient-descent-vs-adagrad-vs-momentum-in-tensorflow

Linear Regression Model

Model:

$$y_{pred} = X \cdot \theta + b$$

Loss (Optimization Goal):

$$\mathcal{L}_\theta = \frac{1}{2n} \sum_{i=1}^{n} (y_i - y_{pred_i})^2$$

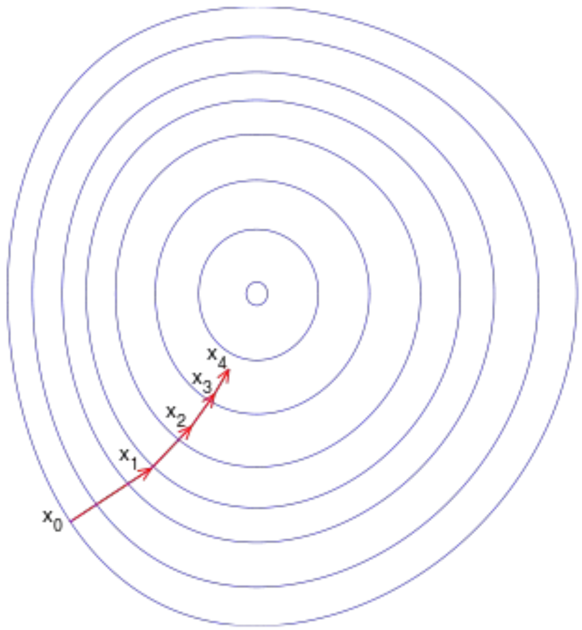
Linear Regression Optimization

- Intuition: We want to make updates to θ such that we minimize the loss
- How? We perform gradient descent

$$\begin{aligned}\theta_{i+1} &= \theta_i - \alpha \nabla_\theta \mathcal{L}_\theta \\ &= \theta_i - \alpha(y^{(i)} - y^{(i)}_{pred})x^{(i)}\end{aligned}$$

Gradient Descent Illustration

- If this is our loss function, our configurations of theta will place us somewhere initially
- We take small steps to minimize our loss by adjusting theta



Connecting this back to TF

- If we define a differentiable loss function, TF can automatically take the gradient and perform the optimization in one call

```
optimizer = tf.train.GradientDescentOptimizer(learning_rate=LEARNING_RATE).\
    minimize(loss) #<< SO easy!!!!
```

Linear Regression Example

- See Part 2 in the notebook
- We predict housing prices using the boston housing dataset

Next Steps

- You can use these same techniques to build and optimize neural networks
- TF is a highly extensive and actively changing piece of software