ML@B Bootcamp - Basic Models in Tensorflow

Machine Learning at Berkeley

ML@B

Agenda

Basic TensorFlow Structure

Lab Overview

Basic TensorFlow Structure

Step 1: Read in Data

Loading in Datasets isn't terribly challenging and we'll demonstrate with a csv.

Many available general interfaces for loading

Tensorflow has a nice interface for MNIST that provides many convenient methods

Step 2: Define Placeholders

ML@B

Placeholders act as the "input" markers for our TF graph

We need to define placeholders for our features X and the corresponding labels, y.

The first dimension of X and y placeholders are always set to None - this signals to the graph constructor that we want to accept variable batch sizes to our model.

Placeholders not just limited to training data - sometimes we can use them for other values like dropout probability.

Step 3: Define Variables for Each Layer

ML@B

Variables hold the parameters of our model.

Fully-connected and Convolutional layers: both use weight and bias variables.

Weights - typically initialized from a normal random distribution to avoid local minimum and break symmetry in backprop that would occur with zero initialization

Biases - typically initialized as slightly positive (e.g. 0.1)

Step 4: Construct each layer

We limit our focus to fully-connected layers for now

We need to construct the following function for each layer:

$$\sigma(X \cdot W + b)$$

- σ will be the activation function of the layer (relu, sigmoid)
- X is the input from the previous layer (or the input data for the first layer). The size will [batch size, num. input features.]
- W are the weights, and will be of size [num. input features, num. output features]
- b is the bias, and will be of size [1, num. output features]

Step 5: Loss function

Loss fn simplicity:

- Define our objective
- Function that takes in the Y placeholder and compares to the net prediction, $\hat{Y}$
- Eg:
  - Simple : Mean Squared Error $\mathcal{L} = \| Y - \hat{Y} \|_2^2$
  - More advanced: Cross Entropy
    $H_{y'}(y) := -\sum_i (y_i' \log(y_i) + (1 - y_i') \log(1 - y_i))$
  - Super advanced: Adding other quantities to loss functions (eg Style transfer with style loss and content loss)
  - We are provided fns like cross entropy

Step 6: Define Optimizer

ML@B

- Takes in learning rate as parameter
- If loss is failing to converge with each epoch of training or appearing as nan, try decreasing the learning rate
- Set the optimizer to minimize the loss function

Step 7: Create a session

ML@B

- Can use InteractiveSession() for iPython notebooks and Session() for Python scripts
- Make sure to initialize all global variables using: sess.run(tf.global_variables_initializer())

Step 8: Train the Model

ML@B

- For each epoch, go through all batches of training data and run optimizer
- Aggregate loss over all batches in each epoch - print this value at the end of the epoch
- For models that take longer to train, may be a good idea to create a function that runs the optimization process for a given number of epochs
    - Allows you to see if loss is decreasing after certain number of epochs

Step 9: Test Model

ML@B

- Depends on type of problem you are trying to solve
- Can include printing accuracy, plotting a regression line, displaying generated data, etc.

Lab Overview

Basic Models

ML@B

- Linear Regression on Dummy Data
- Logistic Regression on MNIST Data
- Basic Feed Forward Network on MNIST Data

Extra Exercise - Denoising Autoencoder