Batch norm:

tf.nn.batch-normalization()

**TensorFlow**

* In this section we will learn the basic structure of TensorFlow programs.
* Lets see how to implement a minimization function:
  + Example function: J(w) = w^2 - 10w + 25
  + The result should be w = 5 as the function is (w-5)^2 = 0
  + Code v.1:
  + import numpy as np
  + import tensorflow as tf
  + w = tf.Variable(0, dtype=tf.float32) # creating a variable w
  + cost = tf.add(tf.add(w\*\*2, tf.multiply(-10.0, w)), 25.0) # can be written as this - cost = w\*\*2 - 10\*w + 25
  + train = tf.train.GradientDescentOptimizer(0.01).minimize(cost)
  + init = tf.global\_variables\_initializer()
  + session = tf.Session()
  + session.run(init)
  + session.run(w) # Runs the definition of w, if you print this it will print zero
  + session.run(train)
  + print("W after one iteration:", session.run(w))
  + for i in range(1000):
  + session.run(train)

print("W after 1000 iterations:", session.run(w))

* + Code v.2 (we feed the inputs to the algorithm through coefficients):
  + import numpy as np
  + import tensorflow as tf
  + coefficients = np.array([[1.], [-10.], [25.]])
  + x = tf.placeholder(tf.float32, [3, 1])
  + w = tf.Variable(0, dtype=tf.float32) # Creating a variable w
  + cost = x[0][0]\*w\*\*2 + x[1][0]\*w + x[2][0]
  + train = tf.train.GradientDescentOptimizer(0.01).minimize(cost)
  + init = tf.global\_variables\_initializer()
  + session = tf.Session()
  + session.run(init)
  + session.run(w) # Runs the definition of w, if you print this it will print zero
  + session.run(train, feed\_dict={x: coefficients})
  + print("W after one iteration:", session.run(w))
  + for i in range(1000):
  + session.run(train, feed\_dict={x: coefficients})

print("W after 1000 iterations:", session.run(w))

* In TensorFlow you implement only the forward propagation and TensorFlow will do the backpropagation by itself.
* In TensorFlow a placeholder is a variable you can assign a value to later.
* If you are using a mini-batch training you should change the feed\_dict={x: coefficients} to the current mini-batch data.
* Almost all TensorFlow programs use this:
* with tf.Session() as session: # better for cleaning up in case of error/exception
* session.run(init)

session.run(w)

* In deep learning frameworks there are a lot of things that you can do with one line of code like changing the optimizer.***Side notes:***
* Writing and running programs in TensorFlow has the following steps:
  + Create Tensors (variables) that are not yet executed/evaluated.
  + Write operations between those Tensors.
  + Initialize your Tensors.
  + Create a Session.
  + Run the Session. This will run the operations you'd written above.
* Instead of needing to write code to compute the cost function we know, we can use this line in TensorFlow :tf.nn.sigmoid\_cross\_entropy\_with\_logits(logits = ..., labels = ...)
* To initialize weights in NN using TensorFlow use:
* W1 = tf.get\_variable("W1", [25,12288], initializer = tf.contrib.layers.xavier\_initializer(seed = 1))
* b1 = tf.get\_variable("b1", [25,1], initializer = tf.zeros\_initializer())
* For 3-layer NN, it is important to note that the forward propagation stops at Z3. The reason is that in TensorFlow the last linear layer output is given as input to the function computing the loss. Therefore, you don't need A3!
* To reset the graph use tf.reset\_default\_graph()