TensorFlow skill testing project: Classification in TensorFlow

Objective: Create and run your own machine learning algorithms in TensorFlow on a real-world dataset.

Architecture: Your code will consist of three modules. This document contains the specification for the second module.

Module 2: graph_constructor.py

You have complete freedom to design this module any way that makes sense to you. Your goal will be to build three different computational graphs. The first will be a simple logistic regression model, the second will be a two-layer neural net, and the third will be a *k*-nearest-neighbour classifier. These graphs will be used to train and test your algorithm using real world data. This module, as well as **data_preprocessing.py**, are essentially helper functions for **learn.py**, which you'll build out next.

Common mistakes

There are a few things to keep an eye on when you start working with TensorFlow. Here are some things you should watch out for as you build this module:

1. Training ops: Training ops are functions like

   tf.train.GradientDescentOptimizer(learning_rate).minimize(loss_function)

   In order to make your algorithm learn, you'll have to *explicitly run these* in a TensorFlow session. In other words, if your training op is called **train_op()**, you'll need a line like

   train_op_val = session.run(train_op, feed_dict=feed_dict)

   in your code.

2. Cross-entropy: the function

   tf.nn.softmax_cross_entropy_with_logits(logits=logits, labels=labels)

takes **logits** as one of its arguments. **logits** is supposed to be the raw output of your model *prior to normalization* by the **softmax()** function. Be sure not to take the **softmax()** of your logits *prior* to feeding them into the cross-entropy function, because **softmax_cross_entropy_with_logits()** implements the **softmax()** itself internally.