50.021 -AI

Alex

Week 04: Tensorflow: Tensors,Variables,Placeholders

[The following notes are compiled from various sources such as textbooks, lecture materials, Web resources and are shared for academic purposes only, intended for use by students registered for a specific course. In the interest of brevity, every source is not cited. The compiler of these notes gratefully acknowledges all such sources.]

The goal of this lesson is to explain some common elements that you will see in tensorflow codes ... and why they are used.

- placeholder
- Variable
- variable_scope, name_scope – for encapsulation. `https://www.tensorflow.org/api_docs/python/tf/variable_scope` `https://www.tensorflow.org/api_docs/python/tf/Graph#name_scope`

We have seen Tensors and Operators in Monday's lecture. An operator takes tensors as inputs, and produces an output tensor. A tensor is just a unit of data. It can have a datatype, a rank (the number of dimensions, 0 for a scalar, 1 for a vector, 2 for a matrix, n for higher order tensors), and a shape.

In terms of the graph: the operators are the nodes, and tensors are the edges.

The philosophy is: you create a graph in python, then you create a session, initialize all global variables, and do a `sess.run([op1,op2], feed_dict=fd)`. The latter runs the graph. When you run the graph, there is a compilation step, and the actual computation is run in $C++$.

Placeholder

A place holder is a tensor, with one special property: if you do not feed it with a feed dictionary, you will get an error message. You can feed values not only into placeholders, but only placeholders will generate an error if unfed. Thus they are a good choice for feeding in data into the graph.

Variables

A variable is a tensor with two special properties:

- its value is saved when one write a checkpoint using `tf.train.Saver`
- it is a mutable tensor

We explain both.

Saving values: when training a neural network, we have to compute activations of layers, gradients and updates of weights. When we want to use a trained neural network, then we need to save and load the weights. So one needs to save the weights during training. Variables are those tensors, for which the current values are stored when writing a checkpoint using `tf.train.Saver`. Thats why variables are used in tensorflow to store weights and other kind of trainable/learnable parameters of a network. This is their main usage in tensorflow.

Mutable tensor: you can define a variable `v`, and use `v.assign(val)` inside a graph, or `sess.run(v.assign(val))` within python to assign value *val* to a variable. The ability to assign a value by *tf.assign* is not the formal definition of mutability, but its most practical consequence.

Variables can be simply put into existence by `v=tf.get_variable(...)` without depending on anything in the graph, and they can be assigned some value. All other tensors except placeholders are the result of some graph computation. For that reason variables are often used for step counters and other data that can change during training.

- note 1: some tensorflow operations require a mutable tensor - that is usually a variable.
- note 2: you can use variables with feed dicts, but they will not complain if you forget to feed them.
- note 3: if using `v=tf.get_variable(name='v', ...)` to create a variable, then it comes with a safeguard (that can be disabled by `reuse_variable()`) against accessing the same variable twice by two calls of `v=tf.get_variable(name='v',...)` that refer to the same `name`. For that reason `v=tf.get_variable(name='v', ...)` should be preferred over `v=tf.Variable(...)`

  `variable_scope('scopename')` (see lesson) is a way to encapsulate variables with the same name. `variable_scope('scopename')` allows to reuse a function that creates a tensor flow operation and during that creation also creates a variable with a fixed name –

which without this scope would cause an error due to above safeguard.

Note: `name_scope('scopename')` does NOT work with `v=tf.get_variable(...)`, but only with `tf.Variable`. in terms of style it is encouraged to use `v=tf.get_variable(...)`

variable_scope

The tensorflow tutorial says wisely: Variable scope allows to create new variables and to share already created ones while providing checks to not create or share by accident.

`https://www.tensorflow.org/programmers_guide/variables`