TensorFlow Basics

Fast Campus
Start Deep Learning with Tensorflow

TensorFlow’s Most Important Classes

- Tensor
- Operation
- Graph
- Session
- Variable
- Optimizer
- Dataset
- Iterator
- Saver

Tensor

- Tensors play a central role in TensorFlow development and serve as the primary objects for storing and manipulating data
- A tensor serves as an N-dimensional array, where N can be zero or more.
  - A tensor’s number of dimensions is called the tensor’s *rank*
  - The size of each dimension is called the tensor’s *shape.*
- TensorFlow provides hundreds of functions for creating, transforming, and processing tensors

Operation

- When the Python interpreter reaches a function that operates on tensors, it doesn't execute the operation immediately. Instead, it creates an instance of the Operation class that represents the operation.
- Every Operation has a property called inputs that contains its input tensors and a property called outputs that contains its output tensors

Graph

- TensorFlow creates a Tensor instance for each tensor in your application and an Operation for each operation involving tensors. It stores these Tensors and Operations in a data structure called a Graph.
- You can access a particular tensor with get_tensor_by_name or access all of the graph's operations by calling get_operations.
- Each Graph stores data in a series of containers called collections. Every collection can be accessed through a particular key, and get_all_collection_keys provides the full list of keys.
    - For example, a graph stores its global variables in the collection whose key is tf.GraphKeys.GLOBAL_VARIABLES.

Session

- After you add tensors and operations to a graph, you can execute the graph's operations by creating and running a session.
- You can create a session by calling tf.Session and then launch the session by calling its run method.
- The first argument of the run method tells the session what processing to perform. If this argument contains tensors, the session will compute the elements of each tensor and return the elements in a NumPy array.
- If this argument contains Operations, the session will perform each operation and return the appropriate result.

Variable

- Variables resemble tensors in many respects. They store values in N-dimensional arrays and can be operated upon using regular TensorFlow operations.
- But during training operations, applications rely on variables to store the state of the model.
  - For example, if an application consists of a neural network, the network's weights and biases will be stored as variables.
- Another difference is that variables require a different set of methods than tensors.
  - For example, after you create a Variable, you need to initialize its value by running a special operation in the session.

Optimizer

- The disparity between an application’s model and the observed data is called loss. A TensorFlow application reduces loss using an optimizer.
- In code, you can create an optimizer by instantiating a subclass of the Optimizer class. Every optimizer has a minimize method that returns an operation that can be executed in a session.

Dataset

- One of the most recent changes to the TensorFlow API is the promotion of the tf.contrib.data package to tf.data. This package provides the all-important Dataset class, which TensorFlow recommends for loading and processing data.
- This class provides many powerful methods for batching and transforming data, and in many cases, you can perform these operations in a multithreaded manner.

Iterator

- The Dataset class provides many powerful capabilities, but it doesn't let you access its data directly. To extract tensors from a dataset, you need to create an instance of the Iterator class.

Saver

- The goal of training is to determine which variables produce the least possible loss.
- Training can take hours or days, so it’s crucial to store the variable’s values during and after training. TensorFlow makes this possible by providing the Saver class.
- After you create a Saver instance, you can call save to store the model's state in numbered checkpoint files.
- You can load the model’s variables from the checkpoint files by calling the restore method.

Training with TensorFlow

1. Construct a mathematical expression for the general model.
2. Declare variables to be updated as training is performed.
3. Obtain an expression for the loss, which is the difference between the model and observation.
4. Create an Optimizer with the loss from Step 3 and call its minimize method.
5. Feeding data into a Session.
6. Execute the session by calling the session’s run method.