Notes on TensorFlow

Sameer Kesava PhD

May 1, 2020

Abstract

Important points on coding in tensorflow and keras with tensorflow backend. Tensorflow objects rather than Python objects preferrable for defining and training in tensorflow.

version 2.1

1 Estimators

- Estimators are like scikit-learn pipelines.
- For training models for distributed environments.

2 Data Formats

- Use a combination of protobuf and tf.recordIO to speed up training.
- Inputs must better be converted to float32 (float64 rarely encountered) since certain layers like dropout or batchnorm will throw out a "Value Error" if the inputs are not of floating point type.

3 Images

- tf.image module for formatting images.
- Resizing can be done using methods such as bilinear, lanczos, etc.
- tf.keras.preprocessing.image.array_to_img()

4 Python Scopes

1. Global: should be explicitly defined.
2. Nonlocal: same as above.
3. local

Functions within function give greater control when also including scopes such as global and nonlocal.

5 Array Broadcasting

- In numpy and tensorflow, the smaller array is "broadcast" across a larger array so that they have compatible shapes.
- **Broadcasting Rule:** In order to broadcast, the size of the trailing axes for both arrays in an operation must either be the same size or one of them must be one.

6 Debugging in TensorFlow

tf.debugging

7 Logits

- The argument "from_logits" that is passed to the loss function should be carefully reviewed. If the last layer has a softmax activation, then from_logits must be set to False.
- If the y labels are negative and not one-hot encoded, using from_logits = False will lead to wrong normalization.
- See **Candidate Sampling** for training data containing large number of classes.

8 Transfer Learning

8.1 Regularization

Adding regularizers to pre-trained models is not straightforward.

1. Load the pre-trained model and add regularizers to the desired layers.
2. Save the model to a json or yaml object.

3. Save the model weights.
4. Delete this model and create a new one using the json object.
5. Load the saved weights into this model.
6. Check for the layers where the regularizers were added, should be created now.

9 Padding

- tf.keras.preprocessing.sequence.pad_sequences
- Post-padding is recommended to be able to use CuDNN implementation of the layers.

10 Masking

- Masking is required to inform the model that some part of the data has padding.
- In models using functional or sequential API, the masks are propagated directly.
- However, in subclassed models, the mask arg. must be explicity passed to the call method. E.g. the compute_mask method of the embedding layer can be passed as an arg. to the call method of the next layer like LSTM.

11 Embedding

- tf.nn.embedding_lookup(): returns the embedded values of the inputs from the embedding tensor.

12 Tokenizer

- For text generation: tf.keras.preprocessing.text.Tokenizer(). Also used in image captioning with visual attention.

13 Dense Layer

- Can take input of more than 2 dimensions.

14 Gradient Calculation

- Variables not part of the gradient can be tracked using tape.watch().
- 1. tf.clip_by_value; tf.clip_by_norm for gradient clipping to address exploding gradient problem.

15 Other Layers

1. Lambda layer: wraps arbitrary expressions such as lambda functions or other functions as a Layer object; helps improve portability of the models.
2. Activation layer: Useful when outputs from multiple inputs are concatenated, and for Batch/Layer normalization layers.

16 Useful functions

It is preferrable to use tf objects, methods and functions as it enables serialization of the code possible and easy visualization in TensorBoard.

1. Check tf.keras.backend for useful functions.
2. tf.map_fn: applies a function to a tensor along an axis.
3. tf.reduce_sum: sum along axes; another important parameter is keepdims.
4. tf.square: squaring the elements.
5. tf.cond(bool, true func, false func): returns only a func.
   - Use lambda expression to return a tensor or other output or the functions must be callable.
   - Objects in true func cannot be accessed by false func.
   - Outputs from both must be of the same type and shape.
   - Functions can return **None**.
6. tf.nn.moments: for mean and variance. Important args: axes and keepdims.
7. tf.reduce_any: logical or along axes; keepdim arg.
8. tf.reduce_mean: mean along axes; keepdim arg.
9. tf.equal: compares elements between 2 tensors. return bool.
10. TensorSpec: for passing tensors with general shapes.

17 Tensor properties version 2.1

- tensor.assign() for assigning a new value or manipulating a tensor variable. For a specific index [m,n], use tensor[m,n].assign(). Direct assignment such as var = var + newvalue will cause issues.
- Do not directly use layer.weights for examining and manipulating, use the .numpy() or .read_value().
- All tf.Variable objects that are not part of training must have their trainable attribute set to False.

18 tf.Graph properties version 2.1

It is very important to understand the graph-based training in tensorflow. The points below are based on my observations, might not be correct. Need to learn more about this.

- Once a graph is created, the variables are added to the graph. These variables cannot be reinitialized later but can be manipulated using the assign method.

19 Keras Layer properties version 2.1

- A layers state is given by its weights/variables. Some layers are **stateless**, i.e. in these layers only data transformation occurs.
- When subclassing layers, parameters such as name, trainable and dtype can be passed to the init method, from where the trainable argument can be controlled.
- A layer's call arguments are inputs, training and mask.

20 Keras Model properties version 2.1

- Keras.Model tracks both its variables and its internal layers.
- All the functions and methods of a model, including the functions corresponding to the inference mode, are evaluated when compiled. This is especially relevant for subclassed models.
- In order to reset some of the Variables in the model or its layers, the layers have to be passed a **stateful** argument during initialization like in RNN.
  1. Directly assigning model.stateful = True will lead to AttributeError as stateful is a readonly property and hence cannot be directly assigned.
  2. Thus, each layer must be passed a stateful argument. Also, it is not a kwarg for a keras layer.

21 Keras Loss methods version 2.1

1. tf.keras.losses.Loss class can be subclassed to define custom loss functions.