tf.contrib.learn.Head

Contents

Class Head

Properties

logits_dimension

Methods

create_model_fn_ops

Class **Head**

Defined in `tensorflow/contrib/learn/python/learn/estimators/head.py`.

Interface for the head/top of a model.

Given logits (or output of a hidden layer), a Head knows how to compute predictions, loss, default metric and export signature. It is meant to,

1) Simplify writing model_fn and to make model_fn more configurable 2) Support wide range of machine learning models. Since most heads can work with logits, they can support DNN, RNN, Wide, Wide&Deep, Global objectives, Gradient boosted trees and many other types of machine learning models. 2) To allow users to seamlessly switch between 1 to n heads for multi objective learning (See _MultiHead implementation for more details)

Common usage: Here is simplified model_fn to build a multiclass DNN model.

```
def _my_dnn_model_fn(features, labels, mode, params, config=None):
  # Optionally your callers can pass head to model_fn as a param.
  head = tf.contrib.learn.multi_class_head(...)
  input = tf.contrib.layers.input_from_feature_columns(features, ...)
  last_hidden_layer_out = tf.contrib.layers.stack(
      input, tf.contrib.layers.fully_connected, [1000, 500])
  logits = tf.contrib.layers.fully_connected(
      last_hidden_layer_out, head.logits_dimension, activation_fn=None)

  def _train_op_fn(loss):
    return optimizer.minimize(loss)

  return head.create_model_fn_ops(
      features=features,
      labels=labels,
      mode=mode,
      train_op_fn=_train_op_fn,
      logits=logits,
      scope=...)
```

Most heads also support logits_input which is typically the output of the last hidden layer. Some heads (like heads responsible for candidate sampling or hierarchical softmax) intrinsically will not support logits and you have to pass logits_input. Here is a common usage,

```
return head.create_model_fn_ops(
    features=features,
    labels=labels,
    mode=mode,
    train_op_fn=_train_op_fn,
    logits_input=last_hidden_layer_out,
    scope=...)
```

There are cases where computing and applying gradients can not be meaningfully captured with train_op_fn we support (for example, with sync optimizer). In such case, you can take the responsibility on your own. Here is a common use case,

```
model_fn_ops = head.create_model_fn_ops(
    features=features,
    labels=labels,
    mode=mode,
    train_op_fn=tf.contrib.learn.no_op_train_fn,
    logits=logits,
    scope=...)
if mode == tf.contrib.learn.ModeKeys.TRAIN:
  optimizer = ...
  sync = tf.train.SyncReplicasOptimizer(opt=optimizer, ...)
  update_op = tf.contrib.layers.optimize_loss(optimizer=sync,
                                              loss=model_fn_ops.loss, ...)
  hooks = [sync.make_session_run_hook(is_chief)]
  ... upate train_op and hooks in ModelFnOps and return
```

Properties

`logits_dimension`

Size of the last dimension of the logits `Tensor`.

Typically, logits is of shape `[batch_size, logits_dimension]`.

Returns:

The expected size of the `logits` tensor.

Methods

`create_model_fn_ops`

```
create_model_fn_ops(
    features,
    mode,
    labels=None,
    train_op_fn=None,
    logits=None,
    logits_input=None,
    scope=None
)
```

Returns `ModelFnOps` that a model_fn can return.

Please note that, + Exactly one of `logits` and `logits_input` must be provided. + All args must be passed via name.

Args:

- `features`: Input `dict` of `Tensor` objects.
- `mode`: Estimator's `ModeKeys`.
- `labels`: Labels `Tensor`, or `dict` of same.
- `train_op_fn`: Function that takes a scalar loss `Tensor` and returns an op to optimize the model with the loss. This is used in TRAIN mode and must not be None. None is allowed in other modes. If you want to optimize loss yourself you can pass `no_op_train_fn` and then use ModeFnOps.loss to compute and apply gradients.
- `logits`: logits `Tensor` to be used by the head.
- `logits_input`: `Tensor` from which to build logits, often needed when you don't want to compute the logits. Typically this is the activation of the last hidden layer in a DNN. Some heads (like the ones responsible for candidate sampling) intrinsically avoid computing full logits and only accepts logits_input.
- `scope`: Optional scope for `variable_scope`.

Returns:

An instance of `ModelFnOps`.

Raises:

- `ValueError`: If `mode` is not recognized.
- `ValueError`: If neither or both of `logits` and `logits_input` is provided.

Last updated November 2, 2017.