"""

Interpolation functions

"""

import libtwml

import tensorflow.compat.v1 as tf

import twml

def linear\_interp1(inputs, ref\_inputs, ref\_outputs):

"""

Perform 1D linear interpolation.

Arguments:

inputs:

The query input values.

ref\_inputs:

Reference grid points used for interpolation.

ref\_outputs:

Reference output values used for interpolation.

Returns:

The interpolated outputs for the requested input values.

"""

inputs = tf.convert\_to\_tensor(inputs)

ref\_inputs = tf.convert\_to\_tensor(ref\_inputs)

ref\_outputs = tf.convert\_to\_tensor(ref\_outputs)

ndims = inputs.shape.ndims

ref\_inputs\_ndims = ref\_inputs.shape.ndims

ref\_outputs\_ndims = ref\_inputs.shape.ndims

if (ref\_inputs\_ndims != ndims):

raise ValueError("Dimension mismatch. inputs: %d, ref\_inputs: %d" % (ndims, ref\_inputs\_ndims))

if (ref\_outputs\_ndims != ndims):

raise ValueError("Dimension mismatch. inputs: %d, ref\_outputs: %d" % (ndims, ref\_outputs\_ndims))

if ndims > 2:

raise ValueError("Input dimensions should be < 2D. But got %d." % ndims)

original\_input\_shape = tf.shape(inputs)

# This is needed because isotonic\_calibration expects:

# - inputs of size [num\_samples, num\_classes]

# - ref\_inputs, ref\_outputs of size [num\_classes, num\_bins]

inputs = tf.reshape(inputs, [-1, 1])

ref\_inputs = tf.reshape(ref\_inputs, [1, -1])

ref\_outputs = tf.reshape(ref\_outputs, [1, -1])

# isotonic\_calibration is simply doing linear interpolation.

# This needs to be renamed in the future to make it consistent.

outputs = libtwml.ops.isotonic\_calibration(inputs, ref\_inputs, ref\_outputs)

return tf.reshape(outputs, original\_input\_shape)

def linear\_interp1\_by\_class(inputs, input\_classes, ref\_inputs, ref\_outputs):

"""

Perform 1D linear interpolation.

Arguments:

inputs:

The query input values.

input\_classes:

The class index to use from the reference grid.

ref\_inputs:

Reference 2D grid points used for interpolation.

Each row denotes the grid from a different class.

ref\_outputs:

Reference 2D output values used for interpolation.

Each row denotes the grid from a different class.

Returns:

The interpolated outputs for the requested input values.

"""

inputs = tf.convert\_to\_tensor(inputs)

input\_classes = tf.convert\_to\_tensor(input\_classes)

ref\_inputs = tf.convert\_to\_tensor(ref\_inputs)

ref\_outputs = tf.convert\_to\_tensor(ref\_outputs)

original\_input\_shape = tf.shape(inputs)

# pass through

def in\_func(x):

return x

# indexed function

def cond\_func(i, fn):

idx = input\_classes[i]

x = tf.expand\_dims(fn(), axis=0)

return linear\_interp1(x, ref\_inputs[idx], ref\_outputs[idx])

# Use while loop for now, needs to be replace by a custom C++ op later.

outputs = twml.util.batch\_apply(in\_func, inputs, cond\_func=cond\_func)

return tf.reshape(outputs, original\_input\_shape)