#include "tensorflow/core/framework/op.h"

#include "tensorflow/core/framework/shape\_inference.h"

#include "tensorflow/core/framework/op\_kernel.h"

#include <twml.h>

#include "tensorflow\_utils.h"

using namespace tensorflow;

REGISTER\_OP("IsotonicCalibration")

.Attr("T: {float, double}")

.Input("input: T")

.Input("xs: T")

.Input("ys: T")

.Output("output: T")

.SetShapeFn([](::tensorflow::shape\_inference::InferenceContext\* c) {

// output shape should be the same as input shape.

c->set\_output(0, c->input(0));

return Status::OK();

}).Doc(R"doc(

This operation calibrates probabilities by fitting to a piece-wise non-decreasing function.

Input

input: A tensor containing uncalibrated probabilities.

xs: A tensor containing the boundaries of the bins.

ys: A tensor contianing calibrated values for the corresponding bins.

Expected Sizes:

input: [batch\_size, num\_labels].

xs, ys: [num\_labels, num\_bins].

Expected Types:

input: float or double.

xs, ys: same as input.

Outputs

output: A tensor containing calibrated probabilities with same shape and size as input.

)doc");

template<typename T>

class IsotonicCalibration : public OpKernel {

public:

explicit IsotonicCalibration(OpKernelConstruction\* context)

: OpKernel(context) {}

void Compute(OpKernelContext\* context) override {

const Tensor& input = context->input(0);

const Tensor& xs = context->input(1);

const Tensor& ys = context->input(2);

Tensor\* output = nullptr;

OP\_REQUIRES\_OK(

context,

context->allocate\_output(0, input.shape(), &output));

try {

const twml::Tensor twml\_input = TFTensor\_to\_twml\_tensor(input);

const twml::Tensor twml\_xs = TFTensor\_to\_twml\_tensor(xs);

const twml::Tensor twml\_ys = TFTensor\_to\_twml\_tensor(ys);

twml::Tensor twml\_output = TFTensor\_to\_twml\_tensor(\*output);

twml::linearInterpolation(twml\_output, twml\_input, twml\_xs, twml\_ys);

} catch (const std::exception &e) {

context->CtxFailureWithWarning(errors::InvalidArgument(e.what()));

}

}

};

#define REGISTER(Type) \

\

REGISTER\_KERNEL\_BUILDER( \

Name("IsotonicCalibration") \

.Device(DEVICE\_CPU) \

.TypeConstraint<Type>("T"), \

IsotonicCalibration<Type>); \

REGISTER(float);

REGISTER(double);