```
#r "nuget: TensorFlow.Net"
#r "nuget: TensorFlow.Keras"
#r "nuget: SciSharp.TensorFlow.Redist"
#r "nuget: NumSharp"
```

线性回归

Tensorflow.NET 和Tensorflow 是一致的

```
using System;
using Tensorflow.NumPy;
using static Tensorflow.Binding;
```

```
tf.compat.v1.disable_eager_execution();
```

```
var X = tf.placeholder(tf.float32);
var Y = tf.placeholder(tf.float32);

var W = tf.Variable(-0.06f, name: "weight");
var b = tf.Variable(-0.73f, name: "bias");
```

```
float learning_rate = 0.01f;
int display_step = 50;
```

```
var train_X = np.array(3.3f, 4.4f, 5.5f, 6.71f, 6.93f, 4.168f, 9.779f, 6.182f,
7.59f, 2.167f,
    7.042f, 10.791f, 5.313f, 7.997f, 5.654f, 9.27f, 3.1f);
var train_Y = np.array(1.7f, 2.76f, 2.09f, 3.19f, 1.694f, 1.573f, 3.366f, 2.596f,
2.53f, 1.221f,
    2.827f, 3.465f, 1.65f, 2.904f, 2.42f, 2.94f, 1.3f);
var n_samples = (int)train_X.shape[0];

int display_step = 50;

float learning_rate = 0.01f;

int training_epochs = 1000;
```

```
var pred = tf.add(tf.multiply(X, W), b);
var cost = tf.reduce_sum(tf.pow(pred - Y, 2.0f)) / (2.0f * n_samples);
var optimizer = tf.train.GradientDescentOptimizer(learning_rate).minimize(cost);
var init = tf.global_variables_initializer();
```

```


var pred = tf.add(tf.multiply(X, W), b);

var cost = tf.reduce_sum(tf.pow(pred - Y, 2.0f)) / (2.0f * n_samples);

 var optimizer = tf.train.GradientDescentOptimizer(learning_rate).minimize(cost);

var init = tf.global_variables_initializer();
```

```
var sess = tf.Session();
// Run the initializer
sess.run(init);
```

```
for (int epoch = 0; epoch < training_epochs; epoch++)
{
    foreach (var (x, y) in zip<float>(train_X, train_Y))
        sess.run(optimizer, (X, x), (Y, y));

                // Display logs per epoch step
    if ((epoch + 1) % display_step == 0)
    {
        var c = sess.run(cost, (X, train_X), (Y, train_Y));
        Console.WriteLine($"Epoch: {epoch + 1} cost={c} " + $"W={sess.run(W)} b=
{sess.run(b)}");
    }
}
```

```
var training_cost = sess.run(cost, (X, train_X), (Y, train_Y));
Console.WriteLine($"Training cost={training_cost} W={sess.run(W)} b=
{sess.run(b)}");
```

```
var test_X = np.array(6.83f, 4.668f, 8.9f, 7.91f, 5.7f, 8.7f, 3.1f, 2.1f);
var test_Y = np.array(1.84f, 2.273f, 3.2f, 2.831f, 2.92f, 3.24f, 1.35f, 1.03f);
Console.WriteLine("Testing... (Mean square loss Comparison)");
var testing_cost = sess.run(tf.reduce_sum(tf.pow(pred - Y, 2.0f)) / (2.0f *
test_X.shape[0]),(X, test_X), (Y, test_Y));
```

```
Console.WriteLine($"Testing cost={testing_cost}");
var diff = Math.Abs((float)training_cost - (float)testing_cost);
Console.WriteLine($"Absolute mean square loss difference: {diff}");
```

```
using Tensorflow.NumPy;
using static Tensorflow.Binding;
using static Tensorflow.KerasApi;
```

```
train_X = np.array(3.3f, 4.4f, 5.5f, 6.71f, 6.93f, 4.168f, 9.779f, 6.182f,
    7.59f, 2.167f, 7.042f, 10.791f, 5.313f, 7.997f, 5.654f, 9.27f, 3.1f);

train_Y = np.array(1.7f, 2.76f, 2.09f, 3.19f, 1.694f, 1.573f, 3.366f,
    2.596f, 2.53f, 1.221f, 2.827f, 3.465f, 1.65f, 2.904f, 2.42f, 2.94f, 1.3f);
```

```
tf.enable_eager_execution();
```

```
var layers = keras.layers;
var inputs = keras.Input(shape: 1);
var outputs = layers.Dense(1).Apply(inputs);
var model = keras.Model(inputs, outputs);
```

```
model.summary();
```

```
 model.compile(loss: keras.losses.MeanSquaredError(),
    optimizer: keras.optimizers.SGD(0.005f),
     metrics: new[] { "acc" });
```

```
 model.fit(train_X, train_Y, epochs: 100);
```