命令式编程(Pytorch)

```
import torch
# 创建训练数据
train_X, train_Y = create_data()
# 创建线性模型
class LinearRegression(torch.nn.Module):
    def __init__(self):
        super().__init__()
        self.linear = torch.nn.Linear(1, 1)
    def forward(self, x):
        out = self.linear(x)
        return out
linear = LinearRegression()
# 代价函数
loss_function = torch.nn.MSELoss()
# 优化器
optimizer = torch.optim.SGD(linear.parameters(), lr=0.01)
# 训练
for epoch in range(500):
    prediction = linear(train_X)
    loss = loss_function(prediction, train_Y)
    optimizer.zero_grad()
    loss.backward()
    optimizer.step()
    print("cost",  loss)
```

声明式编程(TensorFlow1.0)

```
import tensorflow as tf
# 创建训练数据
train_X, train_Y = create_data()
# 定义输入输出占位符
X = tf.placeholder("float")
Y = tf.placeholder("float")
# 创建线性模型
W = tf.Variable(tf.random_uniform([ 1],-1.0,1.0), name="weight")
b = tf.Variable(tf.zeros([ 1]), name="bias")
activation = tf.add(tf.multiply(X, W), b)
# 定义代价函数
cost = tf.reduce_mean(tf.square(activation-Y))
# 定义优化器
optimizer = tf.train.GradientDescentOptimizer(0.01).minimize(cost)
init = tf.global_variables_initializer()
# 训练
with tf.Session() as sess:
    sess.run(init)
    for epoch in range(500):
        for (x, y) in zip(train_X, train_Y):
            sess.run(optimizer, feed_dict={X: x, Y: y})
    print("cost=", sess.run(cost, feed_dict={X: train_X, Y: train_Y}))
```