案例3：使用SRN对sinx做自回归

2023 年 4 月 26 日

[1]:
```
import sys
sys.path.append(r"D:\Rhitta_GPU")
from math import sqrt
import cupy as cp
import numpy as np
import pandas as pd
import rhitta.nn as nn
import matplotlib.pyplot as plt
```

第一步：载入数据集

[2]:
```
data = pd.read_csv("../data/dataset/sinx.csv", header=0, index_col=0)
print(data.head())
```

```
         y0        y1        y2        y3  label=y4
0 -0.048592 -0.065253  0.180059 -0.043315  0.140453
1 -0.065253  0.180059 -0.043315  0.158397 -0.182901
2  0.180059 -0.043315  0.158397  0.261649 -0.027983
3 -0.043315  0.158397  0.261649  0.078123  0.091882
4  0.158397  0.261649  0.078123  0.242422  0.256120
```

划分训练集、测试集　由于这里就是简单测试SRN是否work，就简单全部变成训练集算了

[3]:
```
time_series = data[["y0", "y1", "y2", "y3"]].values
labels = data["label=y4"].values
time_series = cp.array(time_series)
labels=cp.array(labels)
time_series.shape,labels.shape
```

[3]: ((296, 4), (296,))

第二步：选择models库里面的SRN模型，并初始化

[4]:
```
simple_rnn = nn.SRN(input_size=1, hidden_size=3, time_dimension=4)
linear = nn.Linear(input_size=3,output_size=1)
```

第三步：构造计算图

[5]:
```
# 初始隐藏状态节点，输入时间序列节点列表，标签节点
h_0 = nn.to_tensor(size=(1, 3))
inputs = [nn.to_tensor(size=(1, 1)) for i in range(4)]
label = nn.to_tensor(size=(1, 1))

# 将上述节点丢进来构建计算图
h_out = simple_rnn(inputs, h_0)
output = linear(h_out)
loss = nn.MSELoss(output, label)  # 把y和刚刚的输出节点丢进来，构造完整的计算图
```

第四步：初始化优化器

[6]:
```
learning_rate = 0.01
optimizer = nn.Adam(nn.default_graph, loss, learning_rate=learning_rate)
```

第五步：开始训练、评估

[7]:
```
batch_size = 16
epochs = 30

for epoch in range(epochs):
    count = 0
    N= 296

    # 填坑并训练
    for i in range(N):
```

```
        # 输入时间序列
        for j in range(4):
            inputs[j].set_value(time_series[i, j])
        # 输入隐藏状态
        h_0.set_value(np.zeros((1, 3)))
        # 输入标签
        label.set_value(labels[i])
        # 前向反向传播
        optimizer.one_step()
        # 更新计数器
        count += 1
        # 计数器达到batch_size就更新模型参数
        if count >= batch_size:
            optimizer.update()
            count = 0

    # 每个epoch后评估模型的平均平方损失
    acc_loss = 0
    for i in range(N):
        for j in range(4):
            inputs[j].set_value(time_series[i, j])
        h_0.set_value(np.zeros((1, 3)))
        label.set_value(labels[i])
        loss.forward()
        acc_loss += loss.value
    average_loss = acc_loss / N
    print("epoch:{} , average_loss:{}".format(epoch , sqrt(average_loss)))
```

```
epoch:0 , average_loss:1.848456907654333
epoch:1 , average_loss:1.532382814092584
epoch:2 , average_loss:1.224935022501346
epoch:3 , average_loss:0.9555258012701698
epoch:4 , average_loss:0.7349725079110184
epoch:5 , average_loss:0.5815401817449617
epoch:6 , average_loss:0.48313097998638005
epoch:7 , average_loss:0.4127816977039678
epoch:8 , average_loss:0.3518978831883651
```

```
epoch:9 , average_loss:0.29871293330094134
epoch:10 , average_loss:0.25596649843962255
epoch:11 , average_loss:0.2343254013871879
epoch:12 , average_loss:0.24760496252392442
epoch:13 , average_loss:0.2736686780050771
epoch:14 , average_loss:0.2189930023807349
epoch:15 , average_loss:0.21663926471835257
epoch:16 , average_loss:0.23311660058903214
epoch:17 , average_loss:0.22554695479804504
epoch:18 , average_loss:0.22800300962466666
epoch:19 , average_loss:0.26354167868038875
epoch:20 , average_loss:0.22004842988348444
epoch:21 , average_loss:0.1963979983808989
epoch:22 , average_loss:0.19922809406042816
epoch:23 , average_loss:0.21472084753487053
epoch:24 , average_loss:0.18796350501833187
epoch:25 , average_loss:0.23775927395632673
epoch:26 , average_loss:0.2878547690245738
epoch:27 , average_loss:0.22563456285478725
epoch:28 , average_loss:0.19956148136308322
epoch:29 , average_loss:0.18531323546638445
```

绘制sinx,sinx+noise,predict的曲线

注意被预测的点是y4,y5,. . . .,y300

[8]:
```python
plt.figure(figsize=(10,5))
plt.subplot(111)

# 获取原始x轴坐标
x = np.linspace(0, 6.28, 300)
x = x[4:]

# 真实sinx曲线
y_real=np.sin(x)
plt.plot(x,y_real,"o--", label='sin(x)')
```

```
# 带噪音的sinx曲线
plt.plot(x,cp.asnumpy(labels),"b--", label='sin(x)+noise')

# sinx的预测曲线
y_predict=[]
for i in range(N):
    for j in range(4):
        inputs[j].set_value(time_series[i, j])
    h_0.set_value(np.zeros((1, 3)))
    label.set_value(labels[i])
    output.forward()
    y_predict.append(cp.asnumpy(output.value)[0][0])
plt.plot(x,y_predict,"g-", label='predict')

plt.legend()
```

[8]: <matplotlib.legend.Legend at 0x1d11a3f3730>