import numpy as np

import matplotlib.pyplot as plt

import torch

import torch.nn as nn

# 生成随机漫步数据

N = 1000

T = 1.0

dt = T/N

W = np.zeros((N+1,))

for i in range(1, N+1):

dW = np.sqrt(dt) \* np.random.normal()

W[i] = W[i-1] + dW

# 定义模型类

class BrownianMotionModel(nn.Module):

def \_\_init\_\_(self, input\_size, hidden\_size, output\_size):

super(BrownianMotionModel, self).\_\_init\_\_()

self.hidden\_size = hidden\_size

self.rnn = nn.RNN(input\_size, hidden\_size, batch\_first=True)

self.fc = nn.Linear(hidden\_size, output\_size)

def forward(self, x, hidden):

out, hidden = self.rnn(x, hidden)

out = self.fc(out)

return out, hidden

# 定义训练函数

def train\_model(model, criterion, optimizer, x\_train, y\_train, num\_epochs):

for epoch in range(num\_epochs):

hidden = torch.zeros(1, 1, model.hidden\_size)

loss = 0

for i in range(len(x\_train)):

x = torch.Tensor(x\_train[i]).unsqueeze(0).unsqueeze(2)

y\_true = torch.Tensor(y\_train[i]).unsqueeze(0).unsqueeze(2)

y\_pred, hidden = model(x, hidden)

loss += criterion(y\_pred, y\_true)

optimizer.zero\_grad()

loss.backward()

optimizer.step()

if epoch % 10 == 0:

print('Epoch [{}/{}], Loss: {:.4f}'.format(epoch+1, num\_epochs, loss.item()))

# 训练模型

input\_size = 1

hidden\_size = 10

output\_size = 1

lr = 0.01

num\_epochs = 1000

x\_train = W[:-1].reshape(-1, 1)

y\_train = W[1:].reshape(-1, 1)

model = BrownianMotionModel(input\_size, hidden\_size, output\_size)

criterion = torch.nn.MSELoss()

optimizer = torch.optim.Adam(model.parameters(), lr=lr)

train\_model(model, criterion, optimizer, x\_train, y\_train, num\_epochs)

# 生成新的随机漫步数据并可视化

x\_test = x\_train[-1].reshape(-1, 1)

y\_test = []

hidden = torch.zeros(1, 1, model.hidden\_size)

t = np.linspace(0, T, N+1)

for i in range(N - len(x\_train) + 1):

x = torch.Tensor(x\_test[i]).unsqueeze(0).unsqueeze(2)

y\_pred, hidden = model(x, hidden)

y\_test.append(y\_pred.detach().numpy()[0][0])

t\_test = np.linspace(T, 2\*T, N - len(x\_train) + 1)

y\_test = np.array(y\_test)

W\_test = np.concatenate([W, y\_test.cumsum()])

plt.plot(np.concatenate([t, t\_test]), W\_test)

plt.xlabel('Time')

plt.ylabel('Position')

plt.title('Brownian Motion')

plt.show()

效果截图：







