Assignment 2

# Figure 1

Chart, line chart, histogram

Description automatically generated

# Figure 2 – Adam Algorithm Results

Chart, line chart

Description automatically generated

# Figure 3 - Losses

A picture containing shape

Description automatically generated

# Code

# DL5G.py CS5173/6073 cheng 2023  
# autoregression on hospitalization  
# making a new module with Linear a submodule  
# using MSELoss  
# Goodfellow Algorithm 8.7 Adam  
# with random sample of training data  
# Usage: python DL5G.py  
  
import numpy as np  
import random  
import torch  
import matplotlib.pyplot as plt  
  
x = torch.tensor(np.genfromtxt('hamiltonCountyHospitalization.txt'), dtype=torch.float32)  
T = len(x)  
num\_train = T // 2  
tau = 4  
batch\_size = 32  
  
features = [x[i: T-tau+i] for i in range(tau)]  
X = torch.stack(features, 1)  
y = x[tau:].reshape((-1, 1))  
Xtrain = X[:num\_train]  
ytrain = y[:num\_train]  
  
  
class LinearRegression(torch.nn.Module):  
 def \_\_init\_\_(self, in\_features=tau):  
 super(LinearRegression, self).\_\_init\_\_()  
 self.linear = torch.nn.Linear(in\_features, 1)  
  
 def forward(self, x):  
 return self.linear(x)  
  
  
model = LinearRegression()  
y2 = model(X)  
plt.plot(y)  
plt.plot(y2.detach().numpy())  
plt.show()  
  
for param in model.parameters():  
 param.s = torch.zeros\_like(param)  
 param.r = torch.zeros\_like(param)  
rho1 = 0.9  
rho2 = 0.999  
delta = 1e-8  
  
loss\_fun = torch.nn.MSELoss()  
eta = 0.001   
rho1t = rho1  
rho2t = rho2  
rounds = 1000  
losses = np.zeros(rounds)  
indices = list(range(num\_train))  
for i in range(rounds):  
 random.shuffle(indices)  
 batch\_indices = torch.tensor(indices[:batch\_size])  
 y\_pred = model(X[batch\_indices])  
 loss = loss\_fun(y\_pred, y[batch\_indices]) / batch\_size  
 losses[i] = loss.item()  
 model.zero\_grad()  
 loss.backward()  
 for param in model.parameters():  
 param.s = rho1 \* param.s + (1 - rho1) \* param.grad # one line in Algorithm 8.7  
 param.r = rho2 \* param.r + (1 - rho2) \* param.grad.square()  
 param.data -= eta \* (param.s/(1 - rho1t))/(np.sqrt(param.r/(1 - rho2t)) + delta) # the next four lines in Algorithm 8.7  
 rho1t \*= rho1 # so rho1t is rho1^t, to be used above  
 rho2t \*= rho2 # and rho2t is rho2^t  
  
y2 = model(X)  
plt.plot(y)  
plt.plot(y2.detach().numpy())  
plt.show()  
  
plt.plot(losses)  
plt.show()