NN Practicing Notes

---

1 A Simple Regression Function (一個簡單的迴歸函式)

Thank you for your reading!

This part is a simple machine-learning project, Utilizing the knowledge of backpropagation in neural networks to enable the implementation of regression functions.

Hiri`Jiang

Dalian.China

2026.2.3

```python
import numpy as np
import matplotlib.pyplot as plt

#preparing for inputdata & truedata
input_data = np.arange(0,np.pi*2,0.1)
correct_data = np.sin(input_data)
input_data = ((input_data)/(np.pi))-1
n_data = len(correct_data)

#installing size of diffirent layer
n_in = 1
n_mid = 3
n_out = 1

#installing parameters
wb_width =  0.01
eta = 0.1
epoch = 2001
interval = 500

#MID layer
class mid_layer:
    def __init__ (self,n_upper,n):
        self.w = wb_width*np.random.randn(n_upper,n)
        self.b = wb_width*np.random.randn(n)

    def forward(self,x):
        self.x = x
        self.u = np.dot(self.x,self.w)+self.b
        self.y = 1/(1+np.exp(-self.u)) #here we need the activation function

    def backward(self,grad_y):
        self.grad_y = grad_y
        self.delta = self.grad_y*(1-self.y)*self.y
        self.grad_w = np.dot(self.x.T,self.delta)
        self.grad_b = np.sum(self.delta,axis=0)
        self.grad_x = np.dot(self.delta,self.w.T)

    def update(self,eta):
        self.w -= eta*self.grad_w
        self.b -= eta*self.grad_b

#OUT layer
class out_layer:
    def __init__(self,n_upper,n):
        self.w = wb_width*np.random.randn(n_upper,n)
        self.b = wb_width*np.random.randn(n)
    def forward(self,x):
        self.x = x
```

```
        self.u = np.dot(self.x,self.w)+self.b
        self.y = self.u #here we need the activation function

    def backward(self,t):
        self.t = t
        self.delta = self.y-self.t
        self.grad_w = np.dot(self.x.T,self.delta)
        self.grad_b = np.sum(self.delta,axis=0)
        self.grad_x = np.dot(self.delta,self.w.T)

    def update(self,eta):
        self.w -= eta*self.grad_w
        self.b -= eta*self.grad_b

#initializeing for different layers
middle_layer=mid_layer(n_in,n_mid)
output_layer=out_layer(n_mid,n_out)

#Learning project
for i in range(epoch):
    index_random = np.arange(n_data)
    np.random.shuffle(index_random)
    total_error = 0
    plot_x = []
    plot_y = []

    for idx in index_random:
        x = input_data[idx:idx+1]
        t = correct_data[idx:idx+1]

        middle_layer.forward(x.reshape(1,1))
        output_layer.forward(middle_layer.y)
        output_layer.backward(t.reshape(1,1))
        middle_layer.backward(output_layer.grad_x)
        middle_layer.update(eta)
        output_layer.update(eta)

        if i % interval == 0:
            y = output_layer.y.reshape(-1)
            total_error += 1/2*np.sum(np.square(y-t))
            plot_x.append(x)
            plot_y.append(y)

    if i % interval == 0:
        plt.plot(input_data,correct_data,linestyle="dashed")
        plt.scatter(plot_x,plot_y,marker='3')
        plt.show()

        print("Epoch:"+str(i)+"/"+str(epoch),"Error:"+str(total_error/n_data))
```

That is all.

Email： hiri@mail.dlut.edu.cn

Or yyjiang66@gmail.com