Lecture Notes of NOAI Training

Day 1, June 7, 2024

1. 基础库

```python
import torch
import torch.nn as nn
import torch.optim as optim
import numpy as np

import matplotlib.pyplot as plt
import matplotlib
%matplotlib inline
```

2. 随机生成数据

```python
np.random.rand(200,1) # range >0
np.random.randn(200,1) # range 无限制
[:X] # 倒数 x 位
```

3. matplotlib 绘图函数

X：numpy 随机数集合
Y：$y = 2x + 1$ 获得数值集合

```python
def draw_data(X,Y):
    plt.figure(figsize=(8,6))

    plt.scatter(X,Y,color='blue,label='Data'
    ) # 散点图
    plt.xlabel('X')
    plt.ylabel('Y)
    plt.title('Plot')

    plt.legend()
    plt.show()
```

4. 训练线性回归模型

流程：

1. 数据预处理，转变为 tensor
2. 固定随机种子
3. 定义线性回归模型
4. 定义损失函数 MSE 和优化器 SGD
5. 训练模型，共 100 个 epoch
6. 每个 epoch 记录下 loss，weight 和 bias，分别存放在列表中

```
# 定义线性回归模型
def __init__(self):
    super(LinearRegressionModel,self).__init__()
    self.linear=nn.Linear(1,1)

# 训练函数
for epoch in range(100):
    model.train()
    optimizer.zero_grad()

    outputs=model(X_tensor)
    loss=criterion(outputs,Y_tensor)

    loss.backward()
    optimizer.step()

    losses.append(loss.item())
    w_values.append(model.linear.weight.item())
    b_values.append(model.linear.bias.item())
    if epoch%10==0:
        print(f'Epoch {epoch}, Loss: {loss.item()}')

# 实现流程

## 转换张量
X_tensor=torch.tensor(X,dtype=torch.float32)
Y_tensor=#...

## 固定随机种子
set_seed(42)

## 定义损失函数和优化器
criterion=nn.MSELoss()
```

```
optimizer=optim.SGD(model.parameters(),lr=0.1)

## 训练模型并记录损失值和参数变化
losses,w_values,b_values=train(model,optimizer,criterion,X_tensor,Y_tensor)

# 绘制曲线图

def draw_loss(losses):
    plt.figure(figsize=(8,6))
    plt.plot(range(1,len(losses)+1),losses,color='blue',linestyle='-',linewidth=2)
    plt.xlabel('Epoch')
    plt.ylabel('Loss')
    plt.title('Loss Iteration')
    plt.show()
```

$$\text{MSE Expression Format: } \mathrm{MSE}(w,b) = \frac{1}{N}\sum_{i=1}^{N}(y_i - (wx_i + b))^2 \tag{1}$$

```
def draw_loss_contour (X, Y, w_values, b_values, w_true, b_true);
plt. figure(figsize=(10, 6))
w_min, w_max = [0, 3]
b min, b_max = [0, 2.5]
W_range = np.linspace(w_min, w_max, 100)
b
_range = np.linspace(b_min, b max, 100)
W, B = np.meshgrid(w_range, b_range)

# 计算每个网格点的损失
L = np.zeros (W. shape)
for i in range (w. shape[0]) :
for j in range(w. shape[1]) :
w_ij = W[i, j]
b_ij = B[i, j]
Z[i, j] = np.mean((Y - (w_ij* X + b_ij)) ** 2)

# 绘制等高线和参数变化路径
plt. contour (W, B, Z, Levels=50) plt. colorbar()
plt.plot (w_values, b_values, color= red, marker=, linestyle=-, linewidth=2, markersize=s, label= 'Optimized Path')
plt.scatter(w_values[0], b_values[0], color= black, marker=×, s=100, label='Initial Point')
plt.scatter(w_true, b_true, color='black', marker= o, s=100, label='Ref Point')
plt. xlabel('weight')
plt. ylabel('Bias')
plt.title("Plot")
plt.legend()
plt.show()
```