Assignment 3

# Part 1 – No changes

## Output

A picture containing calendar

Description automatically generated

## Code

# DL7K.py CS5173/6073 cheng 2023  
# iris linear classifier as softmax regression  
# autograd and Linear are used  
# CrossEntropyLoss and Adam used   
# all samples in minibatch training  
# two classes merged into one and virginica is left  
# final loss and misclassification number printed  
# Usage: python DL7K.py  
  
import numpy as np  
import torch  
import random  
  
d = 4  
X = torch.tensor(np.genfromtxt('iris.data', delimiter=",")[:, :d], dtype=torch.float32)  
m = len(X)  
  
q = 2  
y = torch.zeros(m, dtype=torch.long)  
for i in range(100, 150):  
 y[i] = 1  
  
class LinearRegression(torch.nn.Module):  
 def \_\_init\_\_(self):  
 super(LinearRegression, self).\_\_init\_\_()  
 self.linear = torch.nn.Linear(d, q)  
  
 def forward(self, x):  
 o = self.linear(x)  
 return o  
  
model = LinearRegression()  
loss\_fun = torch.nn.CrossEntropyLoss()  
optimizer = torch.optim.Adam(model.parameters())  
  
batch\_size = 32  
rounds = 10000  
indices = list(range(m))  
for i in range(rounds):  
 random.shuffle(indices)  
 batch\_indices = torch.tensor(indices[:batch\_size])  
 Xbatch = X[batch\_indices]  
 ybatch = y[batch\_indices]  
 o = model(Xbatch)  
 loss = loss\_fun(o, ybatch)  
 optimizer.zero\_grad()  
 loss.backward()  
 optimizer.step()  
  
print(loss.item())  
for name, param in model.named\_parameters():  
 print(name, param)  
o = model(X)  
ypred = torch.argmax(o, dim=1)  
print('ypred =', ypred)  
misclassified = torch.sum((ypred != y))  
print('misclassified =', misclassified.item())

# Part 2

## Output

A picture containing calendar

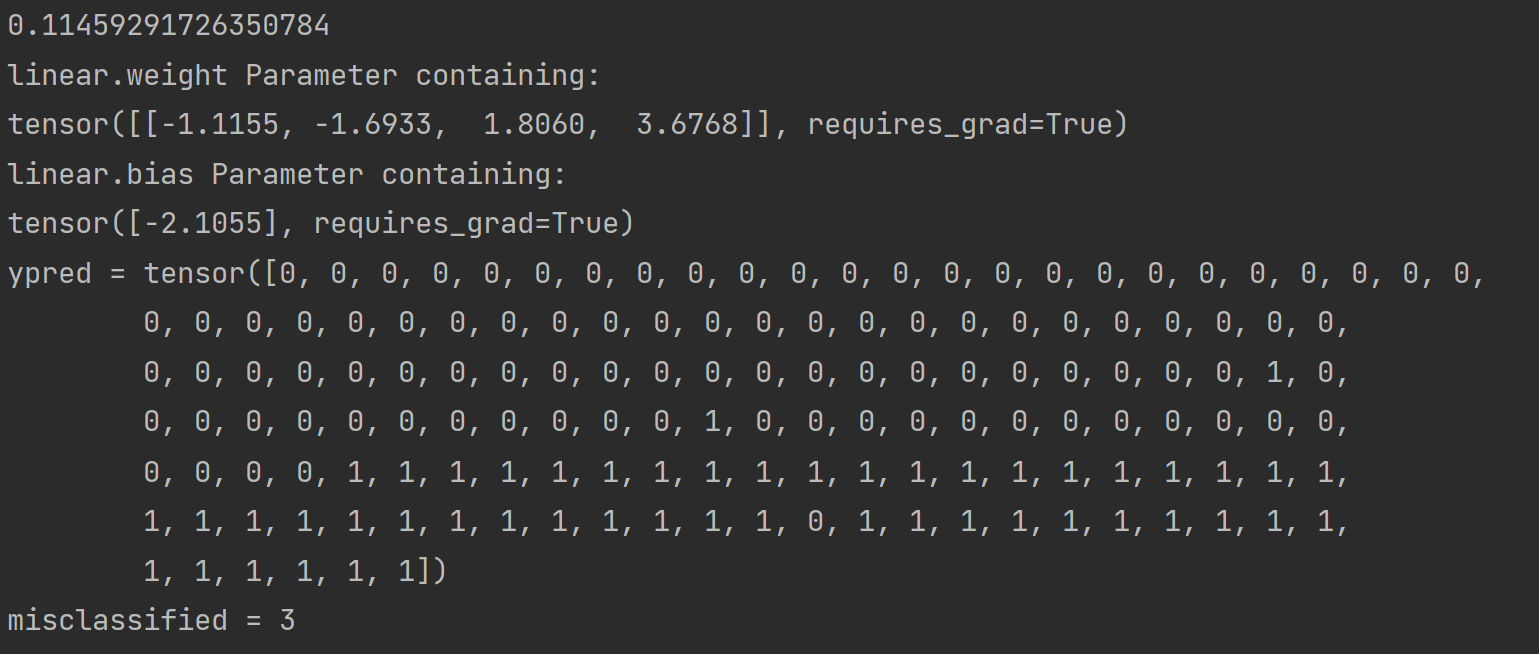
Description automatically generated

## Code

# DL7K.py CS5173/6073 cheng 2023  
# iris linear classifier as softmax regression  
# autograd and Linear are used  
# CrossEntropyLoss and Adam used   
# all samples in minibatch training  
# two classes merged into one and virginica is left  
# final loss and misclassification number printed  
# Usage: python DL7K.py  
  
import numpy as np  
import torch  
import random  
  
d = 4  
X = torch.tensor(np.genfromtxt('iris.data', delimiter=",")[:, :d], dtype=torch.float32)  
m = len(X)  
  
q = 1  
y = torch.zeros(m, dtype=torch.long)  
for i in range(100, 150):  
 y[i] = 1  
  
class LinearRegression(torch.nn.Module):  
 def \_\_init\_\_(self):  
 super(LinearRegression, self).\_\_init\_\_()  
 self.linear = torch.nn.Linear(d, q)  
  
 def forward(self, x):  
 o = self.linear(x)  
 o = torch.cat((torch.zeros(len(x), 1), o), 1)  
 return o  
  
model = LinearRegression()  
loss\_fun = torch.nn.CrossEntropyLoss()  
optimizer = torch.optim.Adam(model.parameters())  
  
batch\_size = 32  
rounds = 10000  
indices = list(range(m))  
for i in range(rounds):  
 random.shuffle(indices)  
 batch\_indices = torch.tensor(indices[:batch\_size])  
 Xbatch = X[batch\_indices]  
 ybatch = y[batch\_indices]  
 o = model(Xbatch)  
 loss = loss\_fun(o, ybatch)  
 optimizer.zero\_grad()  
 loss.backward()  
 optimizer.step()  
  
print(loss.item())  
for name, param in model.named\_parameters():  
 print(name, param)  
o = model(X)  
ypred = torch.argmax(o, dim=1)  
print('ypred =', ypred)  
misclassified = torch.sum((ypred != y))  
print('misclassified =', misclassified.item())

# Part 3

## Output



# Comments

The above code/output shows that 2 different sigmoid functions can be used by using torch.zeros and torch.ones to create the bias terms. As you can see, when using torch.ones instead of torch.zeros, the output tesnor is not all 1s.