Name : Devindi De Silva

Index number : 190128H

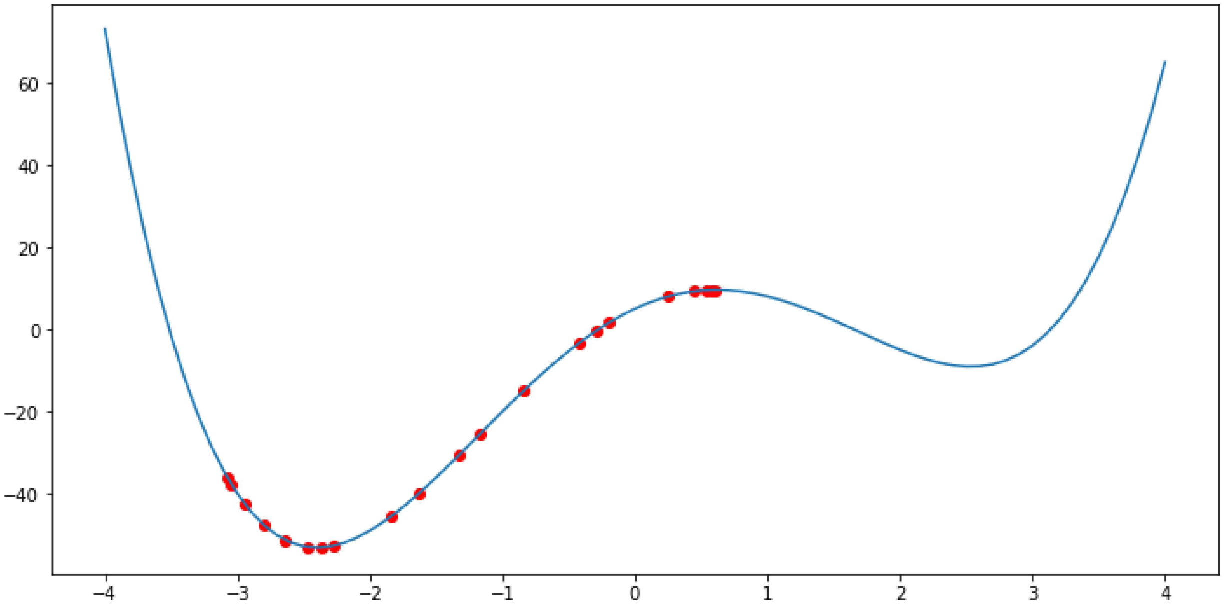
In [ ]:

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

def f(x):
    w= np.array([1,-1,-12,15,5])
    m = np.size(w)- 1
    return np.sum([x**i*w[m-i] for i in range(0, m+1)], axis=0)

def g(x):
    w= np.array([1,-1,-12,15,5])
    m = np.size(w)- 1
    return np.sum([i*x**(i-1)*w[m-i] for i in range(0, m+1)], axis=0)
alpha=0.05
x=0.6
x_hist = np.array(x)
fx_hist = np.array(f(x))
for i in range(20):
    x=x-alpha*g(x)
    x_hist = np.append(x_hist,x)
    fx_hist = np.append(fx_hist, f(x))


print('x = ',x,'f(x) = ',f(x))
fig = plt.figure(figsize = (12,6))
ax=plt.subplot(1,1,1)
delta =0.1
x_ = np.arange(-4,4+delta,delta)
ax.plot(x_,f(x_))
ax.scatter(x_hist,fx_hist,c='r')
```

```
x =  -0.29497479850285213 f(x) =  -0.43550699945570187
```

Out[ ]:

```
<matplotlib.collections.PathCollection at 0x1f731617550>
```



In [ ]:

```
from scipy.optimize import fsolve
from scipy.optimize import minimize
x0 = 0.6
root = fsolve(g, x0)
print(root)

minimum = minimize(f,x0)
print(minimum)
```

```
[0.61654501]
      fun: -53.1184048380149
 hess_inv: array([[0.01680084]])
      jac: array([-2.38418579e-06])
  message: 'Optimization terminated successfully.'
     nfev: 20
      nit: 3
     njev: 10
   status: 0
  success: True
        x: array([-2.40040317])
```

In [ ]:

```
import numpy as np
import tensorflow as tf
from tensorflow import keras
import matplotlib . pyplot as plt
from tensorflow . keras . datasets import cifar10 , mnist

( x_train , y_train ) , ( x_test , y_test ) = cifar10 . load_data ( )
# ( x_train , y_train ) , ( x_test , y_tes t ) = mnist . load_data ( )
print ( " x_train => " , x_train . shape )
Ntr = x_train . shape [ 0 ]
Nte = x_test . shape [ 0 ]
Din = 3072 # CIFAR10
# Din = 784 # MINIST
x_train = x_train [ range (Ntr ) , : ]
x_test = x_test [ range (Nte ) , : ]
y_train = y_train [ range (Ntr ) ]
y_test = y_test [ range (Nte )]

k = len(np.unique(y_train))

y_train = tf.keras.utils.to_categorical(y_train, num_classes = k)
y_test =tf.keras.utils.to_categorical(y_test, num_classes = k)

x_train = np.reshape(x_train, (Ntr, Din))
x_test = np.reshape(x_test, (Nte, Din))
x_train = x_train.astype(np.float32)
x_test = x_test.astype(np.float32)

x_train /= 255.
x_test /= 255.
```

```
 x_train =>  (50000, 32, 32, 3)
```

In [ ]:

```
# Utility function for diaplaying
def display(y_train, y_test, y_train_pred, y_test_pred, loss_history, w, showim = True
    plt.plot(loss_history)


     # For diapaying the weights matrix w as an image. 32*32*3 assumption is there
    if showim:
        f, axarr = plt.subplots(2, 5)
        f.set_size_inches(16, 6)
        for i in range(10):
            img = w[:, i].reshape(32, 32, 3)# CIFAR10
            # img = w1[:, i].reshape(28, 28)# MNIST
            img = (img - np.amin(img))/(np.amax(img) - np.amin(img))
            axarr[i//5, i%5].imshow(img)
        plt.show()

    train_acc = np.mean(np.abs(np.argmax(y_train, axis=1) == np.argmax(y_train_pred, a
    print("train_acc = ", train_acc)

    test_acc = np.mean(np.abs(np.argmax(y_test, axis=1) == np.argmax(y_test_pred, axis
    print("test_acc = ", test_acc)
```

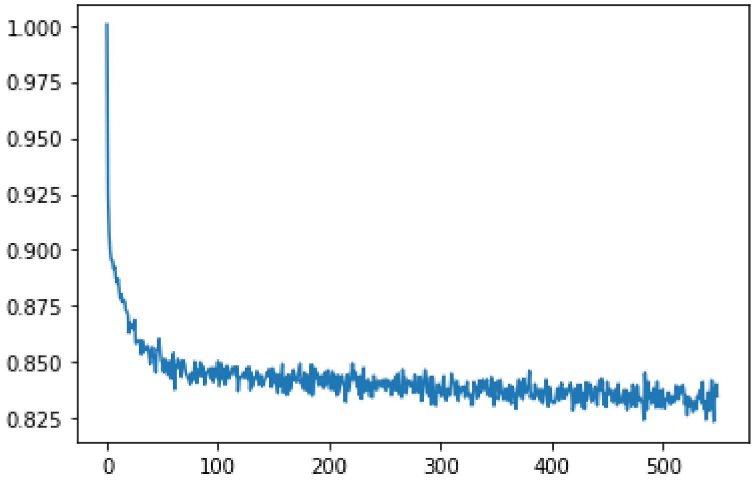
In [ ]:

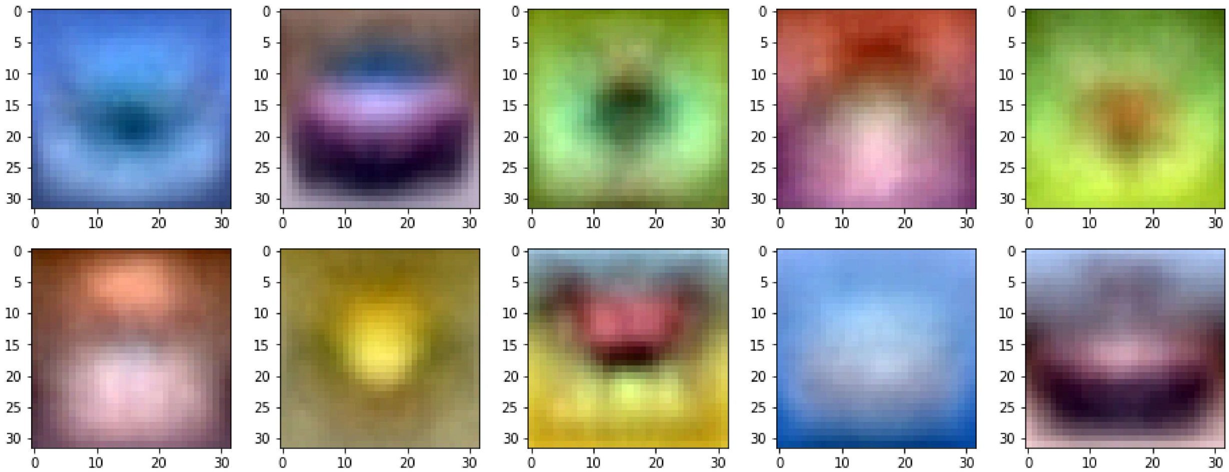
```
std = 1e-5
w = std*np.random.randn(Din, k)
b = np.zeros(k)
lr = 1e-3
lr_decay = 0.1
epochs = 11
batch_size = 1000
loss_hist = []
rng = np.random.default_rng(seed = 0)

for e in range(epochs):
    indices = np.arange(Ntr)
    rng.shuffle(indices)

    for batch in range(Ntr//batch_size):
        batch_indices = indices[batch*batch_size:(batch+1)*batch_size]
        x = x_train[batch_indices]
        y = y_train[batch_indices]

        #forward pass
        y_pred = x@w + b
        loss = 1./batch_size*np.square(y_pred-y).sum()
        loss_hist.append(loss)

        #backward pass
        dy_pred = 1./batch_size* (2.0*(y_pred - y))
        dw = x.T @ dy_pred
        db = dy_pred.sum(axis = 0)*1
        w = w - lr*dw #dw is daba L/daba w
        b = b - lr*db

    if e % 5 == 0:
        print("Iteration %d / %d: loss %f"%(e, epochs,loss))
    if e % 10 == 0:
        lr *= lr_decay
```

```
Iteration 0 / 11: loss 0.850462
Iteration 5 / 11: loss 0.836772
Iteration 10 / 11: loss 0.834906
```

In [ ]:

```
y_train_pred = x_train.dot(w) + b
y_test_pred = x_test.dot(w) + b
display(y_train, y_test, y_train_pred, y_test_pred, loss_hist, w, showim = True)
```





```
train_acc =  0.33562
test_acc =  0.3355
```