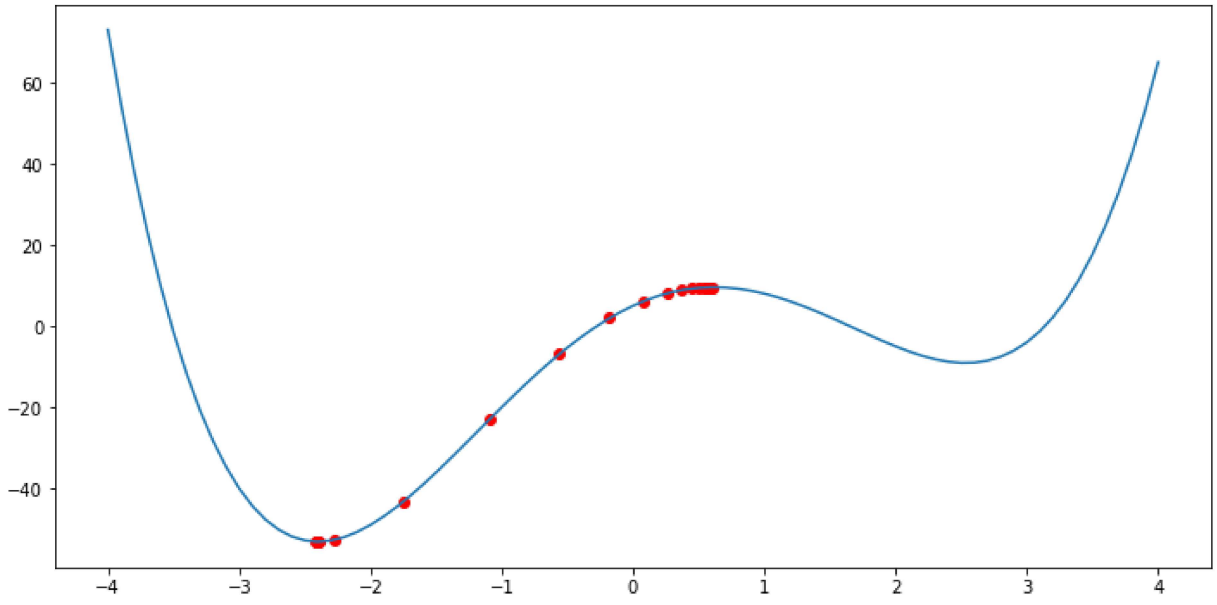
Index number : 190026T

Name : AHAMED M.I.I

In [ ]:

```
#1)

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 = 0.02
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=  -2.4003994283530288 f(x) =  -53.11840483760499
```

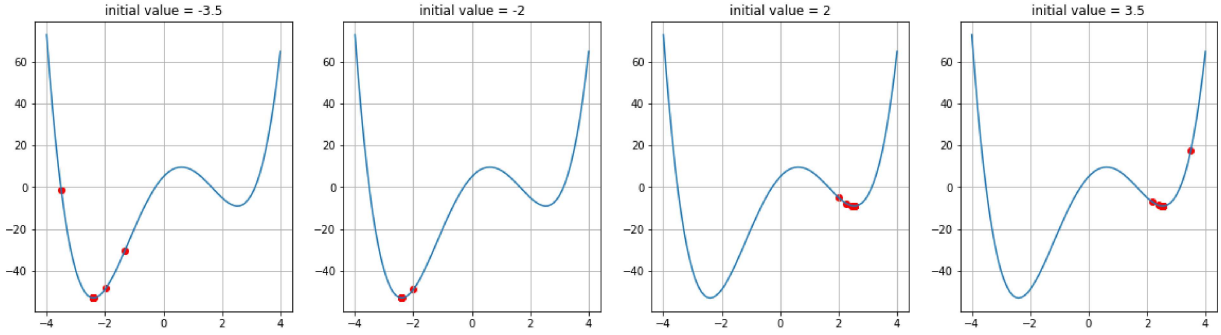


In [ ]:

```
#a) (initial value matters)
alpha = 0.02
X = [-3.5, -2, 2, 3.5]
```

```
fig, ax = plt.subplots(1, 4, figsize=(20, 5))
delta = 0.1
x_ = np.arange(-4, 4+delta, delta)

for j in range(len(X)):
    x = X[j]
    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("initial value = ", X[j])
    print('x= ',x,'f(x) = ',f(x))

    ax[j].set_title("initial value = "+ str(X[j]))
    ax[j].plot(x_,f(x_))
    ax[j].grid(True)
    ax[j].scatter(x_hist, fx_hist, c='r');
```

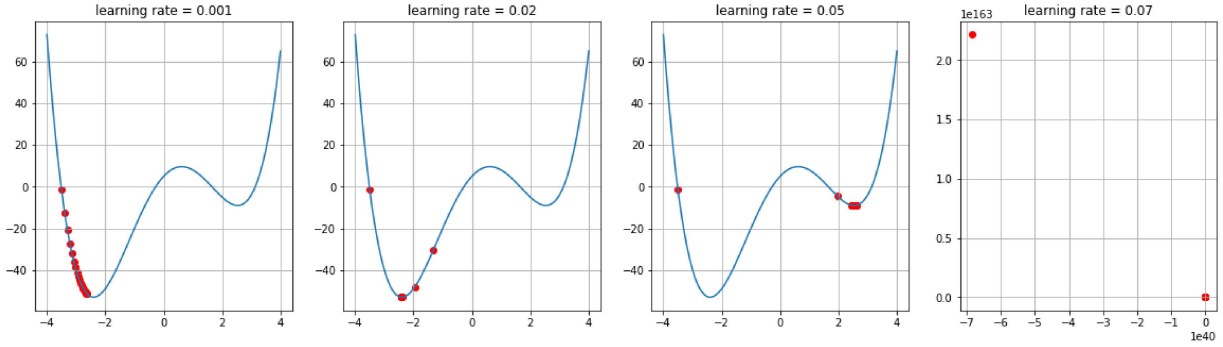
```
initial value =  -3.5
x=  -2.4004031389712734 f(x) =  -53.118404838014925
initial value =  -2
x=  -2.4004031389712566 f(x) =  -53.11840483801494
initial value =  2
x=  2.5338581298324754 f(x) =  -9.083837308516735
initial value =  3.5
x=  2.5338581298317497 f(x) =  -9.083837308516742
```



In [ ]:

```
#a) (learning rate is important)
A = [0.001, 0.02, 0.05, 0.07]


fig, ax = plt.subplots(1, 4, figsize=(20, 5))
delta = 0.1
x_ = np.arange(-4, 4+delta, delta)

for j in range(len(A)):
    x = -3.5
    alpha = A[j]
    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))

    ax[j].set_title("learning rate = "+ str(A[j]))
    ax[j].plot(x_,f(x_))
    ax[j].grid(True)
    ax[j].scatter(x_hist, fx_hist, c='r');
```

```
<ipython-input-18-f5878c5b91e5>:9: RuntimeWarning: overflow encountered in double_sc
alars
  return np.sum([x**i*w[M-i] for i in range(0,M+1)], axis=0)
e:\Softwares\Anaconda\lib\site-packages\numpy\core\fromnumeric.py:87: RuntimeWarnin
g: invalid value encountered in reduce
  return ufunc.reduce(obj, axis, dtype, out, **passkwargs)
<ipython-input-18-f5878c5b91e5>:14: RuntimeWarning: overflow encountered in double_s
calars
  return np.sum([i*x**(i-1)*w[M-i] for i in range(0,M+1)], axis=0)
```



In [ ]:

```
#2)

import ssl
ssl._create_default_https_context = ssl._create_unverified_context
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras.datasets import cifar10, mnist

(x_train, y_train), (x_test, y_test) = cifar10.load_data( )
# ( x_train , y _ t r a i n ) , ( x _ t e s t , y _ t e s 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)]
```

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

In [ ]:

```
# Utility function for displaying
def display(y_train, y_test, y_train_pred, y_test_pred, loss_history, w, showim = Tr
    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,
    print("train_acc = ", train_acc)
```

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

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.
```

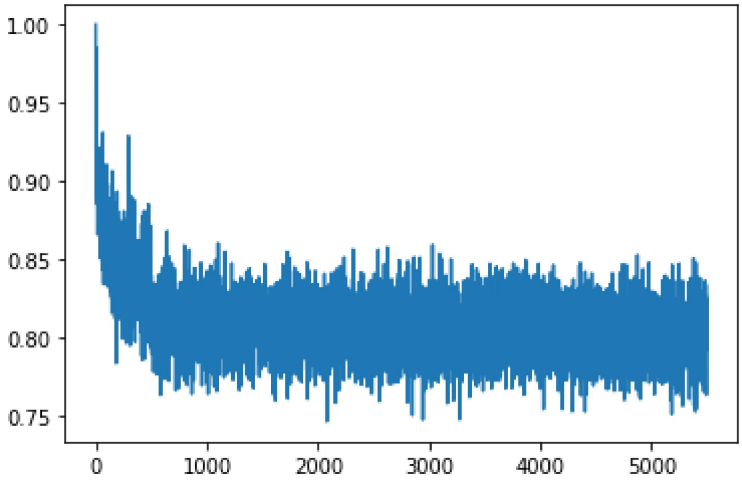
In [ ]:

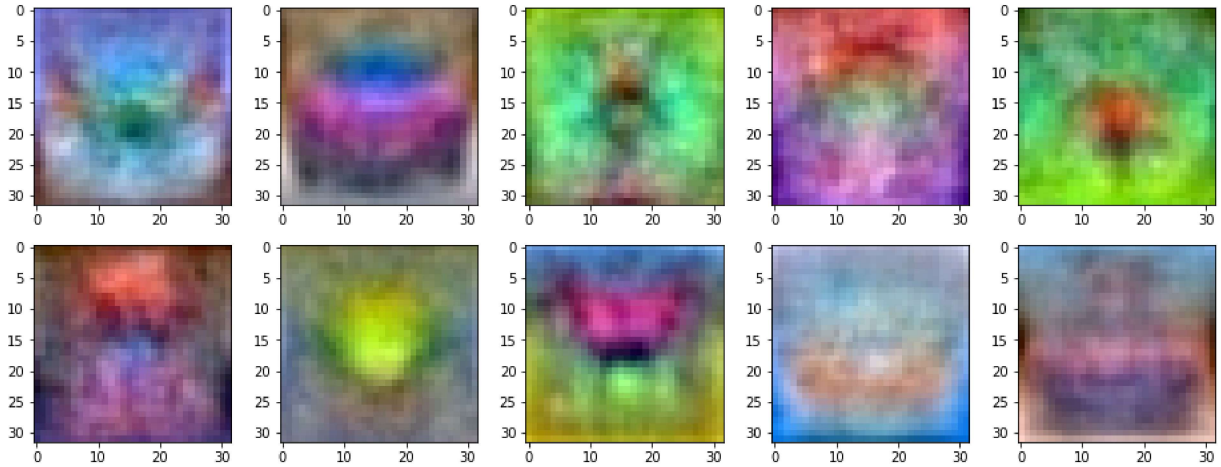
```
std = 1e-5
w = std*np.random.randn(Din, K)
b = np.zeros(K)
lr = 1e-3
lr_decay = 0.1
epochs = 11
batch_size = 100
loss_history = []
rng = np.random.default_rng(seed = 0)

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] #Extract a bath of 100
        y = y_train[batch_indices]

        #Forward pass
        y_pred = x@w + b
        loss = 1./batch_size*np.square(y_pred - y).sum()
        loss_history.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 partial derivative of L with respect to 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 0.813443
Iteration 5 / 11: loss 0.802917
Iteration 10 / 11: loss 0.804666
```

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_history, w, showim = True)
```





```
train_acc =  0.39566
test_acc =  0.3881
```