- **Rishabh Patil**
- **SAP : 60009200056**
- **Div : K/K2**

EVALUATE AND ANALYSIS PREDICTION USING APPROPIATE OPTIMIZERS

In [ ]:

```
import numpy as np
import pandas as pd
from sklearn import datasets
import matplotlib.pyplot as plt
```

CONSTANTS

In [ ]:

```
n_epochs = 300
epochs = list(range(n_epochs))
c = 1
```

FUNCTIONS

In [ ]:

```
def  activation(x,w,b):
    y_in = np.dot(w,x) + b
    y_hat = 1/(1+ np.exp(-(y_in)))
    return y_hat
```

In [ ]:

```
def delta_w(x, y, y_hat):
    dw = c * (-2)* (y-y_hat) * (y_hat*(1-y_hat)) * x
    return dw
```

In [ ]:

```
def delta_b(y, y_hat):
    db = c * (-2) * (y-y_hat) * (y_hat*(1-y_hat))
    return db
```

INPUT

In [ ]:

```
X = [0.5, 2.5]
Y = [0.2, 0.9]
```

BATCH GRADIENT DESCENT

In [ ]:

```
def gradient_descent(Xin, Yin):
    ws_gd = []
    w, b = -2, -2
    print('OLD WEIGHTS AND BIAS:',w,b)
    n = len(Xin)
    for epoch in range(n_epochs):
        dw, db = 0,0
        for x,y in zip(Xin,Yin):
            y_hat = activation(x,w,b)
```

```
            # print(y_hat)
            if y_hat != y:
                dw += delta_w(x, y, y_hat)
                db += delta_b(y, y_hat)

        w -= dw/n
        b -= db/n
        ws_gd.append(w)

    # print(y_hat)
    print('NEW WEIGHTS AND BIAS:',round(w,3),round(b,3),'\n\n')
    return ws_gd
```

In [ ]:

```
ws_gd = gradient_descent(X,Y)
```

```
OLD WEIGHTS AND BIAS: -2 -2
NEW WEIGHTS AND BIAS: 1.733 -2.198
```

MINI BATCH GRADIENT DESCENT

In [ ]:

```
def miniBatchGD(Xin,Yin):
    ws_mbgd = []
    w,b = -2, -2
    batchSize = 1
    print('OLD WEIGHTS AND BIAS:',w,b)

    for epoch in epochs:
        dw, db, counter = 0, 0, 0
        for x,y in zip(Xin,Yin):
            y_hat = activation(x,w,b)
            # print(y_hat)
            dw += delta_w(x, y, y_hat)
            db += delta_b(y, y_hat)
            counter += 1
            if counter % batchSize == 0:
                w -= (c * dw)
                b -= (c * db)

        ws_mbgd.append(w)

    print('NEW WEIGHTS AND BIAS:',round(w,3),round(b,3),'\n\n')
    return ws_mbgd
```

In [ ]:

```
ws_mbgd = miniBatchGD(X,Y)
```

```
OLD WEIGHTS AND BIAS: -2 -2
NEW WEIGHTS AND BIAS: 1.792 -2.282
```

SDG WITH MOMENTUM

In [ ]:

```
def sgdMomentum(Xin, Yin):
    ws_sgdm = []
    w,b = -2, -2
    beta = 0.9
    print('OLD WEIGHTS AND BIAS:',w,b)

    for epoch in epochs:
        dw,db,vw,vb = 0,0,0,0
```

```
        for x,y in zip(Xin,Yin):
            y_hat = activation(x,w,b)
            # print(y_hat)
            dw += delta_w(x, y, y_hat)
            db += delta_b(y, y_hat)
        vw = beta * vw + (1-beta) * dw
        vb = beta * vb + (1-beta) * db

        w -= c * vw
        b -= c * vb
        ws_sgdm.append(w)

    print('NEW WEIGHTS AND BIAS:',round(w,3),round(b,3),'\n\n')
    return ws_sgdm
```

In [ ]:

```
ws_sgdm = sgdMomentum(X,Y)
plt.show()
```

```
OLD WEIGHTS AND BIAS: -2 -2
NEW WEIGHTS AND BIAS: -1.413 -1.372
```

NAG

In [ ]:

```
def NAG(Xin, Yin):
    ws_nag = []
    beta = 0.9
    w,b = -2, -2
    print('OLD WEIGHTS AND BIAS:',w,b)

    for epoch in epochs:
        dw, db, vw, vb = 0,0,0,0
        for x,y in zip(Xin,Yin):
            vw += beta * vw
            vb += beta * vb
            wt = w - vw
            bt = b - vb
            y_hat = activation(x,w,b)
            # print(y_hat)
            dw += delta_w(x, y, y_hat)
            db += delta_b(y, y_hat)
            w -= c * (beta * vw + (1-beta) * dw)
            b -= c * (beta * vb + (1-beta) * db)
        ws_nag.append(w)

    print('NEW WEIGHTS AND BIAS:',round(w,3),round(b,3),'\n\n')
    return ws_nag
```

In [ ]:

```
ws_nag = NAG(X,Y)
```

```
OLD WEIGHTS AND BIAS: -2 -2
NEW WEIGHTS AND BIAS: 0.529 -0.649
```

ADAGRAD

In [ ]:

```
def adagrad(Xin, Yin):
    eps = 2
    beta = 0.9
    w,b = -2, -2
```

```
    ws_adagrad = []
    print('OLD WEIGHTS AND BIAS:', w,b)

    for epoch in epochs:
        dw, db = 0, 0
        for x, y in zip(Xin, Yin):
            y_hat = activation(x,w,b)
            dw += delta_w(x, y, y_hat)
            db += delta_b(y, y_hat)

        w -= (c/(np.sqrt(dw**2 + eps)))*dw
        b -= (c/(np.sqrt(db**2 + eps)))*db

        ws_adagrad.append(w)

    print('NEW WEIGHTS AND BIAS:',round(w,3),round(b,3),'\n\n')
    return ws_adagrad
```

In [ ]:

```
ws_adagrad = adagrad(X, Y)
```

```
OLD WEIGHTS AND BIAS: -2 -2
NEW WEIGHTS AND BIAS: 1.776 -2.26
```

ADADELTA / RMSPROP

In [ ]:

```
def adadelta(Xin, Yin):
    eps = 2
    beta = 0.9
    w,b = -2, -2

    ws_adaDelta = []
    print('OLD WEIGHTS AND BIAS:', w,b)

    for epoch in epochs:
        dw,db = 0,0
        vw, vb = 0,0
        for x, y in zip(Xin, Yin):
            y_hat = activation(x,w,b)
            dw += delta_w(x, y, y_hat)
            db += delta_b(y, y_hat)

        vw = beta * vw + (1-beta) * dw
        vb = beta * vb + (1-beta) * db

        w = w - (c/(np.sqrt(vw + eps)))*dw**2
        b = b - (c/(np.sqrt(vb + eps)))*db**2

        ws_adaDelta.append(w)

    print('NEW WEIGHTS AND BIAS:',round(w,3),round(b,3),'\n\n')
    return ws_adaDelta
```

In [ ]:

```
ws_adaDelta = adadelta(X, Y)
```

```
OLD WEIGHTS AND BIAS: -2 -2
NEW WEIGHTS AND BIAS: -2.024 -2.048
```

ADAM

In [ ]:

```
def Adam(Xin, Yin):
    eps = 2
    beta1, beta2 = 0.45, 0.85
    w,b = -2, -2

    ws_adam = []
    print('OLD WEIGHTS AND BIAS:', w,b)

    for epoch in epochs:
        dw,db = 0,0
        vw, vb = 0,0
        v_ww,v_bb=0,0

        for x, y in zip(Xin, Yin):
            y_hat = activation(x,w,b)
            dw += delta_w(x, y, y_hat)
            db += delta_b(y, y_hat)

        vw = beta1 * vw + (1-beta1) * dw
        vb = beta1 * vb + (1-beta1) * db

        v_ww = beta2 * v_ww + (1-beta2) * dw**2
        v_bb = beta2 * v_bb + (1-beta2) * db**2

        v_wwh = v_ww/(1-beta2)**epoch
        v_bbh = v_bb/(1-beta2)**epoch

        v_wh = vw/(1-beta1)**epoch
        v_bh = vb/(1-beta1)**epoch

        w = w - (c*v_wh/(np.sqrt(v_wwh+eps)))*dw
        b = b - (c*v_bh/(np.sqrt(v_bbh+eps)))*db

        ws_adam.append(w)

    print('NEW WEIGHTS AND BIAS:',round(w,3),round(b,3),'\n\n')
    return ws_adam
```

In [ ]:

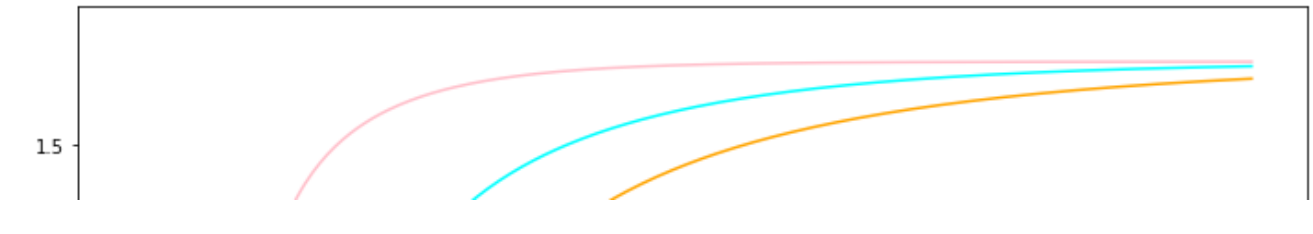
```
ws_adam = Adam(X, Y)
```

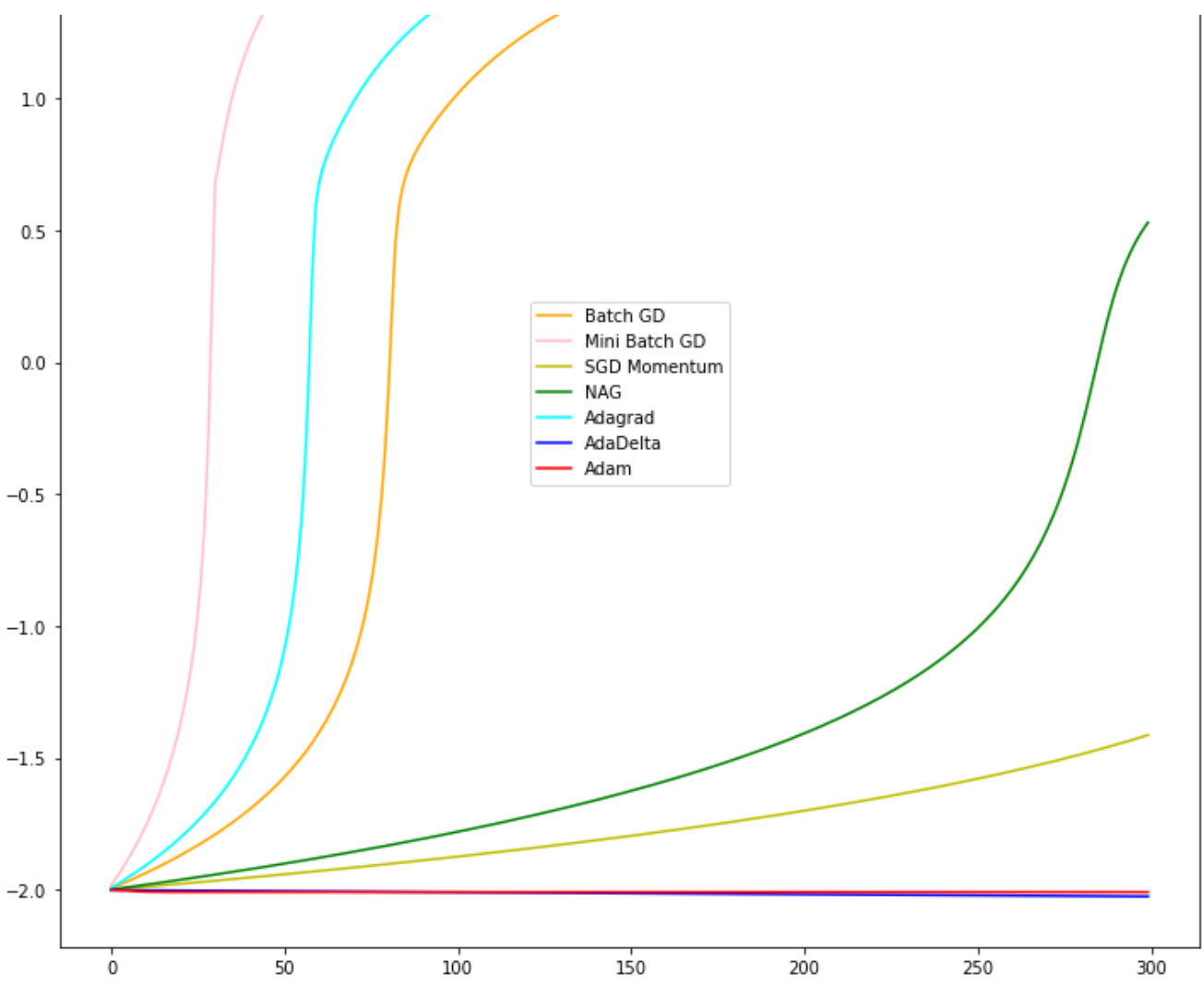
```
OLD WEIGHTS AND BIAS: -2 -2
NEW WEIGHTS AND BIAS: -2.008 -2.012
```

ANALYSIS OF ALL GRADIENT DESCENT

In [ ]:

```
plt.figure(figsize=(12,12))
plt.plot(epochs, ws_gd, color='orange')
plt.plot(epochs, ws_mbgd, color='pink')
plt.plot(epochs, ws_sgdm, color='y')
plt.plot(epochs, ws_nag, color='g')
plt.plot(epochs, ws_adagrad, color='cyan')
plt.plot(epochs, ws_adaDelta, color='b')
plt.plot(epochs, ws_adam, color='r')
plt.legend(["Batch GD", "Mini Batch GD", "SGD Momentum", "NAG", "Adagrad", "AdaDelta", "
Adam"], loc ="best")
plt.show()
```




Batch GD
Mini Batch GD
SGD Momentum
NAG
Adagrad
AdaDelta
Adam
1.0
0.5
0.0
−0.5
−1.0
−1.5
−2.0
0
50
100
150
200
250
300