In [1]:

```
import numpy as np
import pandas as pd
from sklearn.preprocessing import OneHotEncoder
from scipy.special import softmax
onehot_encoder = OneHotEncoder(sparse=False)
```

In [2]:

```
df = pd.read_csv('/content/drive/MyDrive/Datasets/optdigits.tra',header=None)
df.head()
```

Out[2]:

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | ... | 55 | 56 | 57 | 58 | 59 | 60 | 61 | 62 | 63 | 64 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 0 | 1 | 6 | 15 | 12 | 1 | 0 | 0 | 0 | 7 | ... | 0 | 0 | 0 | 6 | 14 | 7 | 1 | 0 | 0 | 0 |
| **1** | 0 | 0 | 10 | 16 | 6 | 0 | 0 | 0 | 0 | 7 | ... | 0 | 0 | 0 | 10 | 16 | 15 | 3 | 0 | 0 | 0 |
| **2** | 0 | 0 | 8 | 15 | 16 | 13 | 0 | 0 | 0 | 1 | ... | 0 | 0 | 0 | 9 | 14 | 0 | 0 | 0 | 0 | 7 |
| **3** | 0 | 0 | 0 | 3 | 11 | 16 | 0 | 0 | 0 | 0 | ... | 0 | 0 | 0 | 0 | 1 | 15 | 2 | 0 | 0 | 4 |
| **4** | 0 | 0 | 5 | 14 | 4 | 0 | 0 | 0 | 0 | 0 | ... | 0 | 0 | 0 | 4 | 12 | 14 | 7 | 0 | 0 | 6 |

5 rows × 65 columns

In [3]:

```
def loss(X, Y, W):
    """
    Y: onehot encoded
    """
    Z = - X @ W
    N = X.shape[0]
    loss = 1/N * (np.trace(X @ W @ Y.T) + np.sum(np.log(np.sum(np.exp(Z), axis=1))))
    return loss

def gradient(X, Y, W, mu):
    """
    Y: onehot encoded
    """
    Z = - X @ W
    P = softmax(Z, axis=1)
    N = X.shape[0]
    gd = 1/N * (X.T @ (Y - P)) + 2 * mu * W
    return gd

def gradient_descent(X, Y, max_iter=1000, eta=0.1, mu=0.01):
    """
    Very basic gradient descent algorithm with fixed eta and mu
    """
    Y_onehot = onehot_encoder.fit_transform(Y.reshape(-1,1))
    W = np.zeros((X.shape[1], Y_onehot.shape[1]))
    step = 0
    step_lst = []
    loss_lst = []
    acc = []
    W_lst = []

    while step < max_iter:
        step += 1
        W -= eta * gradient(X, Y_onehot, W, mu)
        step_lst.append(step)
        W_lst.append(W)
        loss_lst.append(loss(X, Y_onehot, W))
        acc.append(100*(1-loss(X, Y_onehot, W)))

    df = pd.DataFrame({
        'step': step_lst,
        'loss': loss_lst,
        'acc':acc
    })
    return df, W

class Multiclass:
    def fit(self, X, Y):
        self.loss_steps, self.W = gradient_descent(X, Y)

    def acc_plot(self):
        return self.loss_steps.plot(
            x='step',
            y='acc',
            xlabel='Step',
            ylabel='Accuracy'
        )

    def loss_plot(self):
        return self.loss_steps.plot(
            x='step',
            y='loss',
            xlabel='step',
            ylabel='loss'
        )

    def predict(self, H):
        Z = - H @ self.W
        P = softmax(Z, axis=1)
        return np.argmax(P, axis=1)
```

In [4]:

```
X = (df.iloc[:, 0:64])
x = X.to_numpy()
Y = df.iloc[:, 64]
y = Y.to_numpy()
model = Multiclass()
model.fit(x, y)
```

In [5]:

```
df1 = pd.read_csv("/content/drive/MyDrive/Datasets/optdigits.tes",header=None)
Xt = (df.iloc[:, 0:64])
xt = Xt.to_numpy()
Yt = df.iloc[:, 64]
yt = Yt.to_numpy()
y_pred = model.predict(xt)
```

In [6]:

```
print(model.predict(xt))
print(yt)
```

```
[0 0 7 ... 6 6 7]
[0 0 7 ... 6 6 7]
```

Code inspired from towards data science.

In [8]: