LAB 2 **Student ID: 32770995**

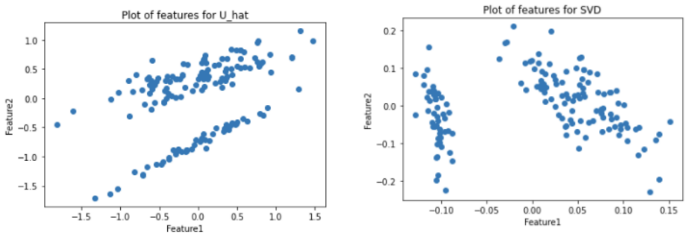
Question 1.1

```
def gd_factorise_ad(A: torch.Tensor, rank: int, num_epochs = 1000, lr = 0.01):

    U = torch.rand(A.shape[0], rank, requires_grad = True)
    V = torch.rand(A.shape[1], rank, requires_grad = True)

    error = 0
    for i in range(num_epochs):

        mse_loss = torch.nn.functional.mse_loss(torch.mm(U, V.t()), A, reduction = 'sum')
        mse_loss.backward()
        U.data = U.data - (lr*U.grad)
        V.data = V.data - (lr*V.grad)
        error = mse_loss

        U.grad.data.zero_()
        V.grad.data.zero_()
    return U, V, error
```

Question 1.2

The reconstruction loss are similar at 15.2288 using gd_factorise_ad function and 15.2289 using svd.

Question 1.3



The plot of datapoints of features in SVD (right) appears to be rotated and scaled from the ones in $\widehat{U}$ (left). Minimizing reconstruction error is the same as maximizing projected variance.

Question 2.1

```
def gd_factorise_ad(A: torch.Tensor, y: torch.Tensor, data_va: torch.Tensor, targets_va: torch.Tensor,rank: int,
                    num_epochs = 100, lr = 0.01):

    W1 = torch.rand(4, 12, requires_grad = True)
    W2 = torch.rand(12, 3, requires_grad = True)
    b1= torch.zeros(1, requires_grad = True)
    b2= torch.zeros(1, requires_grad = True)

    train_acc = 0
    val_acc = 0
    for i in range(num_epochs):

        logits = torch.relu(A @ W1 + b1) @ W2 + b2
        loss = torch.nn.functional.cross_entropy(logits, y, reduction="sum")
        loss.backward()
        W1.data = W1.data - (lr*W1.grad)
        W2.data = W2.data - (lr*W2.grad)
        b1.data = b1.data - (lr*b1.grad)
        b2.data = b2.data - (lr*b2.grad)

        W1.grad.data.zero_()
        W2.grad.data.zero_()
        b1.grad.data.zero_()
        b2.grad.data.zero_()

        if i == (num_epochs - 1):
            preds = torch.argmax(torch.relu(A @ W1 + b1) @ W2 + b2, axis=1)
            train_acc = (y == preds).sum() / preds.shape[0]

            preds_va = torch.argmax(torch.relu(data_va @ W1 + b1) @ W2 + b2, axis=1)
            val_acc = (targets_va == preds_va).sum() / preds_va.shape[0]

    return train_acc, val_acc
```

Question 2.2

	Training Accuracy	Validation Accuracy
0	tensor(0.9400)	tensor(0.9200)
1	tensor(0.9300)	tensor(0.9000)
2	tensor(0.9500)	tensor(0.9200)
3	tensor(0.9500)	tensor(0.9000)
4	tensor(0.6700)	tensor(0.6200)
5	tensor(0.9500)	tensor(0.9000)
6	tensor(0.9500)	tensor(0.9200)
7	tensor(0.9500)	tensor(0.9000)
8	tensor(0.9500)	tensor(0.9200)
9	tensor(0.9400)	tensor(0.8800)

The function was run 10 times. The training and validation accuracy were low for some runs, for example run 4. This may be because the random values the weights were initialized with were not good. The model may be stuck at local minimum and cannot reach the global minimum.