Yang Liu
3323643

Lab Exercise 1

COMP6248
yl58n21@soton.ac.uk

1.1 Implement gradient-based factorisation

```
def sgd_factorise(A:torch.Tensor, rank:int, num_epochs=1000,
lr=0.01) -> Tuple[torch.Tensor, torch.Tensor]:
    m,n = A.shape
    U = torch.rand(m, rank)
    V = torch.rand(n, rank)
    for epoch in range(num_epochs):
        for r in range(m):
            for c in range(n):
                e = A[r, c] - U[r] @ V[c].t()
                U[r] = U[r] + lr * e * V[c]
                V[c] = V[c] + lr * e * U[r]
    return U, V
```

1.2 Factorise and compute reconstruction error

$$\hat{U} = \begin{bmatrix} -0.3409 & 0.5592 \\ 1.3151 & 0.9444 \\ 0.7344 & 1.4513 \end{bmatrix} \quad \hat{V} = \begin{bmatrix} 1.5187 & 1.3255 \\ -0.4731 & 0.6499 \\ 0.7272 & 1.0813 \end{bmatrix} \quad \textbf{loss} = \| \mathbf{A} - \hat{\mathbf{U}}\hat{\mathbf{V}}^{\mathbf{T}} \|_{\mathbf{F}}^{\mathbf{2}} = \mathbf{0.1221}$$

2.1 Compare to the truncated-SVD loss $= \| \mathbf{A} - \mathbf{U_t}\mathbf{\Sigma_t}\mathbf{V_t^T} \|_{\mathbf{F}}^{\mathbf{2}} = \mathbf{0.1219}$ The loss is 0.1219 which is slightly better than the result from my algorithm above. This could be explained by Eckart-Young theorem which states that the least squares approximation in s dimensions of a matrix A can be found by replacing the smallest m-s roots of Σ with zeroes and remultiplying $U\Sigma V^T$ which is exactly what we did.

3.1 Implement masked factorisation

```
def sgd_factorise_masked(A:torch.Tensor, M:torch.Tensor,rank:int,
num_epochs=1000,lr=0.01) -> Tuple[torch.Tensor, torch.Tensor]:
    m, n = A.shape
    U = torch.rand(m, rank)
    V = torch.rand(n, rank)
    for epoch in range(num_epochs):
        for r in range(m):
            for c in range(n):
                if M[r, c] != 0:
                    e = A[r, c] - U[r] @ V[c].t()
                    U[r] = U[r] + lr * e * V[c]
                    V[c] = V[c] + lr * e * U[r]
    return U, V
```

3.2 Implement masked factorisation

$$\hat{A} = \begin{bmatrix} 0.3416 & 0.5996 & 0.1683 \\ \mathbf{2.2856} & 0.0493 & 1.8366 \\ 2.9399 & \mathbf{0.6160} & 2.2632 \end{bmatrix} \quad A = \begin{bmatrix} 0.3374 & 0.6005 & 0.1735 \\ \mathbf{3.3359} & 0.0492 & 1.8374 \\ 2.9407 & \mathbf{0.5301} & 2.2620 \end{bmatrix} \quad loss = \| A - \hat{A} \|_2 = 1.1106$$

$\hat{A}$ is the completed matrix which is very close to the original matrix A in terms of the valid values and somehow captures the partial information of the missing values through the process of factorisation and reconstruction which probably means that some information of the missing values are carried by the other parts of the matrix.