Lab 1–Playing with gradients and matrices in PyTorch
By: Abhash Shrestha (35464984) as14n23@soton.ac.uk

1.1 Gradient-based factorisation

```
def sgd_factorize(A, rank, num_epochs=1000, lr=0.01):

  U = torch.rand(A.shape[0], rank)
  V = torch.rand(A.shape[1], rank)
  for epoch in range(num_epochs):
    for i in range(A.shape[0]):
      for j in range(A.shape[1]):
        e = A[i, j] - torch.dot(U[i], V[j])
      U[i] = U[i] + lr * e * V[j]
      V[j] = V[j] + lr * e * U[i]
  return U, V
```

1.2 Reconstruction error

```
Matrix (U):
      ([[-0.1688,  0.5531],
       [ 1.8053,  0.6097],
       [ 1.3245,  1.1926]])

Matrix (V):
    ([[ 1.5151,  0.8581],
      [-0.2859,  0.8232],
      [ 0.7832,  0.9041]])

RECONSTRUCTED Matrix (V):
    ([[ 0.2188,  0.5036,  0.3678],
      [ 3.2582, -0.0142,  1.9650],
      [ 3.0300,  0.6031,  2.1155]])

ORIGINAL Matrix (V):
    ([[0.3374, 0.6005, 0.1735],
      [3.3359, 0.0492, 1.8374],
      [2.9407, 0.5301, 2.2620]])

Reconstruction Loss: tensor(0.1223)

loss = mse_loss(UV_hat, A, reduction='sum')
```

2.1 Compare to the truncated-SVD

```
U_svd, Sigma_trunc, Vt = torch.svd(A)
   Sigma_trunc[-1] = 0
   Sigma_trunc = torch.diag(Sigma_trunc)
   A_recon = U_svd @ Sigma_trunc @ Vt.t()
   loss =  torch.nn.functional.mse_loss(A_recon, A,
reduction='sum')

Reconstruction Loss: tensor(0.1291)
```

For truncated SVD, the matrix is obtained using truncated SVD of input matrix A. As compared to SGD based matrix factorization, where they are obtained by using multiplication of matrices U and V " UV_hat = torch.mm(U, V.t()) ".

Similarly, the loss is calculated differently for each of them. In truncated SVD, loss is the squared L2 norm of difference between original matrix A and the reconstructed matrix.
torch.norm(A - A_tilde) ** 2

In SGD, it is calculated using mean squared error loss function : loss = mse_loss(UV_hat, A, reduction='sum')

There is almost no difference in the reconstruction loss as the one obtained using truncated SVD with rank 5. This shows that SDG based approach is as good for re-constructing the original matrix compared to truncated SVD.

3.1 Masked factorisation

```
def sgd_factorize_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] == 1:
               e = A[r, c] - torch.dot(U[r], V[c])
               U[r] += lr * e * V[c]
               V[c] += lr * e * U[r]
   return U, V
```

The Estimated complete matrix was:
[0.3406, 0.5973, 0.1706] ,
[2.2198, 0.0489, 1.8369],
[2.9400, 0.9660, 2.2629]

This method managed to achieve a reasonably good result.
The MSE between the original and completed matrices was 0.6253363490104675.
The the value is low so it indicates that the completed matrix closely approximates the original matrix. A low MSE means that there are no substantial differences and the approximation is accurate.

If there were a high MSE, it could be due to various factors such as suboptimal parameter choices, or limitations of the matrix completion algorithm.

Original Matrix (A)	Completed Matrix (A_completed)	Difference
[0.3374, 0.6005, 0.1735]	[0.3406, 0.5973, 0.1706]	Very small
[None, 0.0492, 1.8374]	[2.2198, 0.0489, 1.8369]	Large for 1st entry, small for 3rd
[2.9407, 0.0, 2.2620]	[2.9400, 0.9660, 2.2629]	Small for 1st and 3rd entries, large for 2nd entry

The MSE loss was higher for SGD based factorisation than GD based one.

4.1 Movie rating

The Predicted rating for the movie 'A Beautiful Mind': 3.432534694671631, the Actual Original rating for User 5 on 'A Beautiful Mind' is 0.0.
Sum of squared errors over valid values was 66477632.0

```
def sgd_factorize_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)
```