# LAB 1 - INTRODUCING PYTORCH

*Supritha Konaje - 32864477*
ssk1n21@soton.ac.uk

## 1. IMPLEMENT A MATRIX FACTORISATION USING GRADIENT DESCENT

### 1.1. Implement gradient-based factorisation

The PyTorch implementation for gradient based factorisation is given in the following snippet.

```
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

After Standard Gradient Factorisation, we get the value for U and V as follows:

$$\hat{U} = \begin{bmatrix} -0.3615 & 0.6389 \\ 1.5642 & 0.8355 \\ 0.9161 & 1.4575 \end{bmatrix} \qquad \hat{V} = \begin{bmatrix} 1.4534 & 1.1703 \\ -0.3388 & 0.6228 \\ 0.7387 & 0.9807 \end{bmatrix}$$

Using the following code snippet, reconstruction loss is obtained:

```
sgd_loss = torch.nn.functional.mse_loss(
           input = A_sgd , target = A,
           reduction = 'sum')
```

```
Reconstruction Loss = 0.1219
```

## 2. COMPARE YOUR RESULT TO TRUNCATED SVD

Using the in-built PyTorch function, the SVD of matrix A is calculated as:

$$\text{A-SVD} = \begin{bmatrix} 0.2245 & 0.5212 & 0.3592 \\ 3.2530 & -0.0090 & 1.9737 \\ 3.0378 & 0.5983 & 2.1023 \end{bmatrix}$$

The reconstruction loss is given as:

```
Reconstruction Loss = 0.1219
```

The reconstruction loss is almost similar for the algorithm and the in-built function. According to Eckart-Young theorem, the cost function determines the fit between the matrix and the approximating matrix given a condition that the *rank(matrix)* > *rank(approximating matrix)*. Hence, the SVD function is also a good for approximation as the results are similar.

## 3. MATRIX COMPLETION

### 3.1. Implement masked factorisation

The implementation for the masked factorisation is given as:

```
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] == 1:
          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. Reconstruct a matrix

The matrix obtained after reconstruction is:

$$\hat{A} = \begin{bmatrix} 0.3439 & 0.5989 & 0.1657 \\ 2.2581 & 0.0494 & 1.8358 \\ 2.9393 & 0.7159 & 2.2642 \end{bmatrix}$$

The reconstruction loss is:

```
Reconstruction Loss = 1.1962
```

The result that we get is slightly similar to the original matrix. Hence, we can reconstruct a matrix that has some missing elements provided the rank of the matrix is given.