Deep Learning - Lab 1 Exercise

Han Zhang hz5g21@soton.ac.uk

1 Exercise 1

Exercise 1.1:Implement gradient-based factorisation.

```
from typing import Tuple
def sgd_factorise(A: torch.Tensor, rank:
    int, num_epochs: int = 1000, lr: float =
    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] += lr*e*V[c]
        V[c] += lr*e*U[r]
  return U, V
```

Exercise 1.2:Factorise and compute reconstruction error.

$$\hat{U} = \begin{bmatrix} 0.6182 & -0.1694 \\ 0.4796 & 1.5415 \\ 1.1527 & 1.1063 \end{bmatrix} \hat{V} = \begin{bmatrix} 0.8617 & -1.8438 \\ 0.7562 & -0.2437 \\ 0.8646 & 1.0077 \end{bmatrix}$$

The reconstruction loss of $||A - \hat{U}\hat{V}||$ is about 0.1221.

2 Exercise 2

Exercise 2.1:Compare to the truncated-SVD.

```
def svd_factorise(A: torch.Tensor, rank:
    int) -> Tuple[torch.Tensor,
    torch.Tensor, torch.Tensor]:
  U, S, V = torch.svd(A)
  S[-1] = 0
  return (U, torch.diag(S), V)
```

The reconstruction loss of the SVD approach is about 0.1219, which is smaller than the previous result 0.1221. This means the performance of SVD factorisation is better than that of gradient-based factorisation. These results could be explained by the Eckart-Young theorem. That said, the SVD factorisation is optimum for low-rank approximation.

3 Exercise 3

Exercise 3.1:Implement masked factorisation.

```
def sgd_factorise_masked(A: torch.Tensor, M:
    torch.Tensor, rank: int, num_epochs: int
    = 1000, lr: float = 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] += lr * e * V[c]
          V[c] += lr * e * U[r]
  return U, V
```

Exercise 3.2:Reconstruct a matrix.

$$\hat{U} = \begin{bmatrix} 0.6728 & -0.0480 \\ 0.2790 & 1.2861 \\ 0.7915 & 1.4682 \end{bmatrix} \hat{V} = \begin{bmatrix} 0.6240 & 1.6661 \\ 0.8810 & -0.1529 \\ 0.3502 & 1.3524 \end{bmatrix}$$

$$\hat{A} = \begin{bmatrix} 0.3397 & 0.6000 & 0.1706 \\ 2.3169 & 0.0492 & 1.8370 \\ 2.9402 & 0.4729 & 2.2628 \end{bmatrix}$$

$$\mathrm{A} = \begin{bmatrix} 0.3374 & 0.6005 & 0.1735 \\ 3.3359 & 0.0492 & 1.8374 \\ 2.9407 & 0.5301 & 2.2620 \end{bmatrix}$$

The reconstruction error is about 1.0416, which is very close to the original matrix A. All the reconstructed values in $\hat{A}$ are close to their original values in A expect the a_{10}. This means that the information describing their values was carried in the matrix's other values. The reduced rank approximation captures some of this information, which is projected back to construct the masked values. a_{10} may contain more independent information compared to others as it contributes to a major part of the reconstruction error.