Lab 1 Report

Jack Hilton-Jones (jhj1g23@soton.ac.uk)

I. Matrix Factorisation Implementation Using Gradient Descent

The code below computes the reconstruction loss of a rank-2 factorisation of the matrix A. The output reconstruction loss $\|\mathbf{A} - \hat{\mathbf{U}}\hat{\mathbf{V}}^T\|_F^2$ is 0.12194 (4 d.p).

```
def sgd_factorise(A:torch.Tensor, rank: int, num_epochs=1000, lr=0.01) -> Tuple[torch.Tensor, torch.Tensor]:
  m,n = A.shape
  U_hat = torch.rand(m, rank)
  V_hat = torch.rand(n, rank)

  for epoch in range(num_epochs):
    for r in range(m):
      for c in range(n):
        error = A[r,c] - torch.dot(U_hat[r, :], V_hat[c, :])
        U_hat[r, :] += lr*error*V_hat[c, :]
        V_hat[c, :] += lr* error* U_hat[r, :]
  return U_hat, V_hat

A = torch.tensor([[0.3374, 0.6005, 0.1735], [3.3359, 0.0492, 1.8374], [2.9407, 0.5301, 2.2620]])
U_hat, V_hat = sgd_factorise(A, rank=2)
A_reconstructed = torch.mm(U_hat, V_hat.t())
reconstruction_loss = torch.nn.functional.mse_loss(A, A_reconstructed, reduction ='sum').item()
reconstruction_loss
```

II. Comparison of Matrix factorisation to truncated SVD

The SVD of the matrix A is computed using `torch.svd`. The SVD of the matrix is computed by setting the last singular value to zero, which will ignore the component contributing the least. The error is 0.12191(4 d.p). The Eckart-Young theorem states that the lowest least-squares error for rank-k approximation is given by the first k terms of its singular value decompositions. This truncates S by retaining only the largest k singular values and their corresponding singular vectors. The error is lower for the truncated SVD as the noise is reduced. This is because the smaller singular values are associated with noise, whereas truncating the SVD highlights the more significant data, therefore reducing the loss.

```
U,S,V = torch.svd(A)
S_truncated = S.clone()
S_truncated[-1] = 0
A_reconstructed_svd = torch.mm(torch.mm(U, torch.diag(S_truncated)), V.t())
reconstruction_loss_svd = torch.nn.functional.mse_loss(A, A_reconstructed_svd, reduction = 'sum'). item()
reconstruction_loss_svd
```

III. Matrix Completion

The code implemented takes two matrices of the same shape. The mask is used to compute updates using only the valid values.

$$\begin{bmatrix} 0.3402 & 0.5995 & 0.1706 \\ 1.9416 & 0.0493 & 1.8367 \\ 2.9402 & 0.7020 & 2.6207 \end{bmatrix}$$

Using matrix factorisation, an approximation for the missing data from the original matrix A is given. The approximate matrix is very close to the original matrix in the given values, however in the missing values the accuracy decreases significantly. Adjusting the learning rate and increasing the number of epochs could help tune the model to increase the accuracy. This tells us that the with the right parameters, the model can be tuned to predict missing data.

IV. Movie Recommendation

Implementing the code below, the predicted rating for the film 'A Beautiful Mind' by the 5th user is 3.4667(4 d.p) and therefore it is expected that they would rate this film a 3. The sum of squared errors is 3896975.25, with this being computed while only considering the valid ratings by multiplying the error by the mask matrix. This is a measure of how well the factorised matrix approximates the original matrix.

```
A = torch.load('ratings.pt')
titles_df = pd.read_csv('titles.csv')

def sgd_factorise_masked(A: torch.Tensor, M: torch.Tensor, rank: int, num_epochs: int=1000, lr=1e-5) -> Tuple[torch.Tensor, torch.Tensor]:
    U = torch.rand(A.shape[0], rank)
    V = torch.rand(A.shape[1], rank)

    for epoch in range(num_epochs):
      err = (A - U @ V.t())* M
      U += lr * err @ V
      V += lr * err.t() @ U

    return U, V

M = (A > 0).float()
movie_index = titles_df[titles_df['name'] == 'A Beautiful Mind'].index[0]
U_hat, V_hat = sgd_factorise_masked(A, M, rank=5)
predicted_rating = torch.mm(U_hat, V_hat.t())[movie_index, 4].item()
sse = torch.sum(((A - torch.mm(U_hat, V_hat.t())) * M) ** 2).item()
predicted_rating, sse
```