Lab 1 - Playing with gradients and matrices in PyTorch

Abhilash Pulickal Scaria
aps1n21@soton.ac.uk

Introduction

This lab focuses on matrix factorisation using gradient descent and comparing the reconstruction loss with the case when a truncated SVD is used. Matrix completion using gradient based matrix factorisation is also introduced.

1 Implement matrix factorisation using gradient descent

1.1 Implement gradient-based factorisation

```python
from typing import Tuple
import torch

def sgd_factorise(A: torch.Tensor, rank: int,
      num_epochs = 1000,lr = 0.01) ->
      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):
     for r in range(A.shape[0]):
       for c in range(A.shape[1]):
           e = A[r,c] - (U[r] @ torch.t(V[c]))
           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

The $\hat{\boldsymbol{U}}$ and $\hat{\boldsymbol{V}}$ matrix obtained from rank 2 matrix factorisation and the reconstruction loss obtained when comparing with original matrix A is given below.

$$\hat{\boldsymbol{U}} = \begin{bmatrix} 0.6168 & -0.1530 \\ 0.4108 & 1.5961 \\ 1.0798 & 1.1800 \end{bmatrix},$$

$$\hat{\boldsymbol{V}} = \begin{bmatrix} 0.8126 & 1.8290 \\ 0.7836 & -0.2088 \\ 0.8384 & 1.0195 \end{bmatrix},$$

$$\text{Loss} = 0.121970$$

2 Compare your result to truncated SVD

2.1 Compare to the truncated-SVD

The matrix U_t, S_t and V_t obtained by computing SVD is given below

$$\boldsymbol{U}_t = \begin{bmatrix} -0.0801 & -0.7448 & 0.6625 \\ -0.7103 & 0.5090 & 0.4863 \\ -0.6994 & -0.4316 & -0.5697 \end{bmatrix},$$

$$\boldsymbol{S}_t = \begin{bmatrix} 5.3339 & 0.6959 & 0.0000 \end{bmatrix},$$

$$\boldsymbol{V}_t = \begin{bmatrix} -0.8349 & 0.2548 & 0.4879 \\ -0.0851 & -0.9355 & 0.3430 \\ -0.5439 & -0.2448 & -0.8027 \end{bmatrix},$$

$$\text{Loss} = 0.121910$$

The reconstruction error otained in both the cases are almost same. By using gradient descent we found the $\hat{\boldsymbol{U}}$ and $\hat{\boldsymbol{V}}$ which minimised the reconstruction error and this error was similar to that obtained by truncated SVD. The explanation for this is provided by Eckart - Young Theorem. Eckart-Young Theorem states the best k rank approximation for matrix A (which has the lowest reconstruction error(Frobenius norm)) can be obtained using the truncated SVD. Since in this case we did rank 2 factorisation the last singular value of SVD was set to zero.

3 Matrix completion

3.1 Implement masked factorisation

```python
def sgd_factorise_masked(A: torch.Tensor, M:
      torch.Tensor, rank: int, num_epochs=1000,
      lr=0.01) -> 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):
    for r in range(A.shape[0]):
      for c in range(A.shape[1]):
        if M[r,c]:
          e = A[r,c] - (U[r] @ torch.t(V[c]))
          U[r] = U[r] + lr * e * V[c]
          V[c] = V[c] + lr * e * U[r]

  return U, V
```

3.2 Reconstruct a matrix

$$\hat{\boldsymbol{A}} = \hat{\boldsymbol{U}}\hat{\boldsymbol{V}}^T = \begin{bmatrix} 0.3548 & 0.5951 & 0.1529 \\ 2.2076 & 0.0496 & 1.8327 \\ 2.9377 & 0.7770 & 2.2674 \end{bmatrix},$$

$$\text{Loss} = 1.3348774909973145$$

The estimate of completed matrix is given above. Even though the estimated matrix $\hat{\boldsymbol{A}}$ is not exactly same as A, it is still a reasonable estimation. The reconstruction error is also low. From this we can infer that gradient based matrix factorisation is a reasonable method for matrix completion.