Lab 2–PyTorch Autograd

By: Abhash Shrestha (35464984) as14n23@soton.ac.uk

1.1 Implement gradient-based factorisation using PyTorch's AD

The GD based factorisation was applied using the following snippet:

```
def gd_factorise_ad(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, requires_grad=True)
   V_hat = torch.rand(n, rank, requires_grad=True)

   for epoch in range(num_epochs):
      A_pred = torch.matmul(U_hat, V_hat.t())
      loss = F.mse_loss(A_pred, A, reduction='sum')
      loss.backward()
      with torch.no_grad():
         U_hat -= lr * U_hat.grad
         V_hat -= lr * V_hat.grad
         U_hat.grad.zero_()
        V_hat.grad.zero_()
return U_hat, V_hat, A_pred
```

1.2 Factorise and compute reconstruction error

The Reconstruction Loss with rank 2 factorization using default learning rates and epochs was 15.228955268859863.

And the reconstruction loss using Truncated Singular Value Decomposition was 15.228833198547363.

Both methods have almost a similar loss indicating that both are performing similarly in terms of approximating a matrix.

Using rank 4 gives the following errors: Reconstruction Loss using gd_factorise_ad: 0.04392794892191887

Reconstruction Loss using truncated SVD: 3.979467450010432e-11

These errors are extremely low.
The data we are using has 4 features as well and attempting to represent it in a lower dimensional space (rank 1,2 etc) results in loss of information and a higher reconstruction loss as opposed to when we use rank 4 and get very low loss.

1.3 Compare against PCA

Data Projected using U_hat (Gradient Descent) is more like a linear line with a positive slope indicating a low variance along one of the axes. where as the other plot is scattered.
Relevant diagrams in figure 1 and 2.
The plot for SVD(fig 1) is quite spread out, which means that it successfully captured the variance in the data. This is what PCA aims to do as well and the plot suggests that it successfully identified the direction of maximum variance in the original data.
On the other hand, the second plot(fig 2) for GD is quite tight. This is because GD tries to minimise the reconstruction loss as it prioritizes representing the data with minimal deviation from the chosen subspace.

2.1 Implement the MLP

Manual Initialisation and update of W1, W2, b1, b2 was done as such:

Initialisation:

```
W1 = torch.randn(4, 12, requires_grad=True)
b1 = torch.zeros(12, requires_grad=True)
W2 = torch.randn(12, 3, requires_grad=True)
b2 = torch.zeros(3, requires_grad=True)
```

Update:

```
W1 -= lr * W1.grad
b1 -= lr * b1.grad
W2 -= lr * W2.grad
b2 -= lr * b2.grad
```

2.2 Test the MLP

The training and validation accuracies remain constant as 76.99% and 68.00% which suggests that the model may not be learning effectively from the training data, it can also mean that the model is stuck at a local minima. Perhaps the default parameters used were not optimal. I tried to use a learning rate of 0.1 as opposed to 0.01 and saw an increase in accuracy.
The Training Accuracy was 97.00% and the validation accuracy was 83.99%.

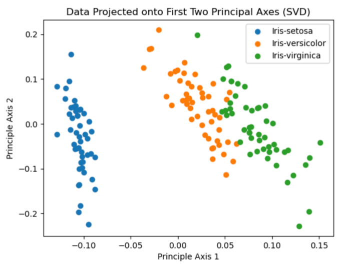

fig 1: SVD projected

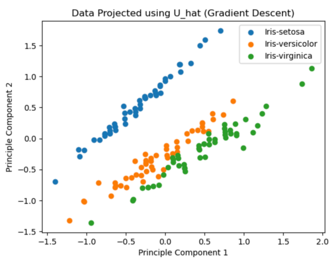

fig 2: GD projected