1 Gradient Descent with Autograd

1.1 Implement gradient-based factorisation using PyTorch's AD
Due to length of code, we kindly ask the reader to visit the following Github notebook and find 'gd_factorise_ad' method for full implementation.

1.2 Factorise and compute reconstruction error on real data
With learning late = 0.01 and 2000 epochs, reconstruction loss = 0.0254. Performing truncated SVD on the same data gives reconstruction loss = 1.0201.

1.3 Compare against PCA
SVD is a numerical method of computing the PCA. The reconstruction error is If mean-centred datapoint is $\boldsymbol{X}$ and its approximations are $\boldsymbol{X}$ then reconstruction error is $||\boldsymbol{X} - \tilde{\boldsymbol{X}}||_F^2$. With Eckhart-Young Theorem, we know that an optimal choice of $\tilde{\boldsymbol{X}}$ is $\boldsymbol{X}_k$ found via truncated SVD.

(a) From $\boldsymbol{U}$ using truncated SVD (b) From $\hat{\boldsymbol{U}}$ using factorisation

Figure 1: Projection of data onto principle directions.

2 A simple MLP

2.1 Implement the MLP
We kindly ask the reader to visit the following Github notebook and find 'mlp_train' method for full implementation.

2.2 Test the MLP

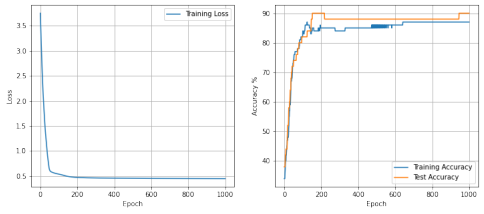

Figure 2: Left plot showing training loss. Right plot showing training accuracy in blue and validation accuracy in orange.