Deep Learning: Lab 2

Johnny Joyce (jvjj1u19@soton.ac.uk)

1 Implement matrix factorisation using gradient descent (take 2)

Q1.1 and Q1.2: gradient-based factorisation

Gradient based factorisation was implemented using PyTorch's automatic differentiation. For the given iris dataset, this achieved a rank-2 factorisation with a mean square error **15.2289**.

By computing the singular value decomposition, we can construct an optimal rank-2 reconstruction by deleting all but the first two singular values (this follows from the Eckart-Young theorem, as discussed in Q2.1 of lab 1). The mean square error of this reconstruction was **15.2288**, so we can see that the solution produced by our implemented algorithm was approximately a mere 0.0001 away from optimal.

Q1.3: Comparison to PCA

Figure 1 shows scatter plots of the projection onto the first two principal axes though SVD and though $\widehat{\mathbf{U}}$.

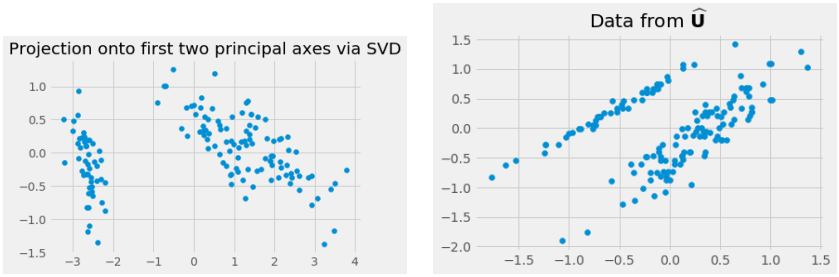

Figure 1: Scatter plots of the given data.

2 A simple MLP

Q2.1 and Q2.2: implementing and testing

The MLP was implemented and the results were recorded. Figure 2 shows a plot of the error on both the training and testing set across each epoch.

Figure 2: Plot of the training/validation error across each epoch.

To check whether a validation instance is correctly classified or not, we can calculate the logits and check whether the argmax is equal to the instance's label. The first time the validation accuracy was calculated, a value of 78% was obtained. However, upon re-shuffling the training and validation data and re-training, validation accuracies of 40-55% would frequently appear. In these situations, we may need to permute the logits such that each logit corresponds to the most frequently predicted respective value in the training data (similarly to swapping "true" and "false" in a binary classifier).