|  |  |
| --- | --- |
| Lab 4: Clustering and Dimensionality Reduction | |
| Student ID: 21102358 | Student Name: 류건호 |

Table 1. Clustering accuracies on FashionMNIST dataset

|  |  |  |  |
| --- | --- | --- | --- |
| Input | Initialization | Mean | Std |
| Image | Random initialization | 0.5176 | 0.0405 |
| Image | Kmean++ initialization | 0.5129 | 0.3482 |
| Latent representation | Kmean++ initialization |  |  |

**1. Kmeans (3pt)**

1. Implement kmeans function in src/kmeans.py (Hint, use implemented building blocks in src/kmeans.py
2. Run 01.kmeans.py 5 times and report the mean and standard deviation of accuacy in Table 1

각 Iteration마다 Accuracy의 편차가 상당히 커서 50번의 iteration을 수행하였다.

1. Implement init\_cluster\_advanced function in in src/kmeans.py. This function should initialize cluster centroids based on kmeans++ algorithms
2. Replace init\_clusters function in kmeans function with init\_cluster\_advanced function. Then, run again 01.kmeans.py 5 times and report mean / std statistics in Table 1

각 Iteration마다 Accuracy의 편차가 상당히 커서 50번의 iteration을 수행하였다.

1. Table 1 compares kmeans clustering results depending on initialization algorithm. Which one gives more stable results? What's the reason?

수치상으로는 Kmean++ initialization 기법이 더 안정적이라고 말할 수 있겠으나, 안정성을 크게 향상시켰다고 보기는 어렵다. kmeans++ 알고리즘은 centroid의 거리가 서로 멀어질 수 있도록 centroid를 초기화하는 방식이다. 이는 같은 cluster에 여러 개의 centroid가 선택되는 것을 어느 정도 방지할 수 있지만, cluster를 대표하지 못하는 샘플들을 선택할 확률이 증가하게 된다고 생각한다. 이것이 kmeans+ 알고리즘이 좋은 성능을 내지 못한 이유라고 생각한다.

1. clustering\_accuracy in src/metrics.py is slightly different from classification accuracy. Explain how they are different and why they should be different

kmeans.py 함수에서 prediction 텐서는 cluster만 시켰을 뿐, 임의로 레이블링을 시켰기 때문이다. 따라서 위 코드에서는 matching\_cost라는 confusion matrix를 만들고, 각 레이블에서 가장 많은 예측값을 가진 클러스터를 레이블로 지정하였다. 예를 들어, A클러스터의 신발 예측 수는 300, B클러스터의 신발 예측 수는 1200(최대)... J클러스터의 신발 예측 수는 400개라고 가정한다면, 신발이라고 레이블링 된 클러스터는 B클러스터라고 간주하는 것이다.

**2. Autoencoder (2pt)**

1. You should implement 02\_autoencoder\_train.py to train a denosing autoencoder in src/autoencoders.py. The detailed procedure is available in 02\_autoencoder\_train.py. (This starts from empty code. If you ignore your team project, this may be the longest code that you should implement in this class. But you have already seen most building blocks to write this code in previous labs !)
2. After the pretraining stage, we want to finetune the baseline model for test tasks. Run "transfer\_finetune.py". To perform fine-tuning, you should implement the code of loading pretrained weights. Then report the accuracy of baseline (finetune) model on Table 1
3. Compare two baseline models. Which one gives better performance on test tasks? What do you think is driving the performance improvement?

**3. t-SNE** **(1pt)**

1. Visualization of image space and latent space of FashionMNIST using sklearn.manifold.TSNE

(https://scikit-learn.org/stable/modules/generated/sklearn.manifold.TSNE.html)

1. Compare image space and latent space of FashionMNIST. Which space represent inputs' structure better in tSNE visualization?