Mathematical Underpinnings - Variational Deep Autoencoder

Michał Gromadzki

May 2024

1 Introduction

Clustering is a key technique in data analysis, helping us find patterns and group similar items in datasets. Traditional methods like k-means often fall short when dealing with complex or high-dimensional data. This is where Variational Autoencoders (VAEs) and their extension, Variational Deep Embedding (VaDE), come into play.

In this project, I implement and compare the performance of both, VAEs and VaDEs, on a well-known MNIST dataset.

2 Implementation

Both models have been implemented as classes in Python using Tensorflow API. To help with the implementation particular layers of the models' such as Encoder and Decoder have been implemented as separate classes. All classes have been implemented in models.py file.

2.1 Helper layers

The Sampling layer is designed to implement the reparameterization trick. This layer takes the mean (z_mean) and log-variance (z_log_var) of a latent distribution as inputs. It generates a sample from this distribution by adding a random noise scaled by the standard deviation - exp($0.5 \cdot z\_log\_var$).

The Encoder layer compresses the input data into a lower-dimensional latent representation. It consists of several dense layers with ReLU activations, progressively transforming the input features into a compact latent space. This latent representation captures the most salient features of the input data in a reduced form.

The Decoder layer reconstructs the original input data from its latent representation. This layer is composed of multiple dense layers with ReLU activations, which gradually transform the latent vector back into the original input dimensions. The final dense layer uses a sigmoid activation function to ensure the output values are within a 0 to 1 range. The decoder effectively reverses the encoding process, allowing the autoencoder to generate reconstructions of the input data.

The Autoencoder class integrates the encoder and decoder into a singular model. It first uses the encoder to transform the input data into a latent representation. This latent vector is then passed to the decoder, reconstructing the input data.

2.2 VAE

The Variational Autoencoder (VAE) model consists of three primary components: an encoder, a decoder, and a sampling mechanism. The encoder compresses input data of dimension input_dim into a lower-dimensional latent space of dimension latent_dim, producing two outputs: z_mean and z_log_var, which represent the parameters of the latent variable's normal distribution. The sampling component uses the reparameterization trick to generate latent variables

z from these parameters. The decoder then reconstructs the input data from the sampled latent variables. Figure 1 depicts the architecture of the model.

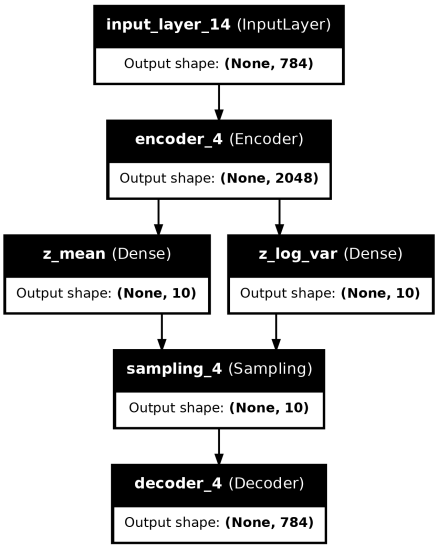


Figure 1: Architecture of VAE

2.3 VaDE

The VaDE (Variational Deep Autoencoder) model consists of the encoder, decoder, sampling mechanism, and learnable clustering parameters. The forward pass works in the same way as in the VAE. However, during back-propagation, the clustering parameters are also trained. This facilitates clustering within the latent space, where pi_ represents the cluster priors, mu_c denotes the cluster means, and log_var_c indicates the cluster variances. Figure 2 presents the architecture of the model.

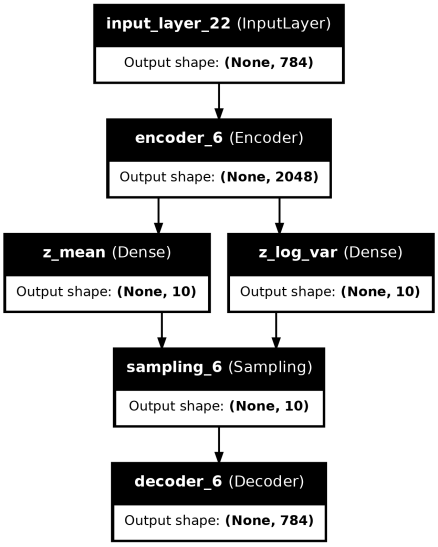


Figure 2: Architecture of VaDE

3 Training

Both models were trained on the MNIST dataset with a batch size of 128 for 100 epochs. The VAE was optimized using the ADAM optimizer with a learning rate of 1^{-3} while the VaDE used the ADAM optimizer with a learning rate of 1^{-4}. Both models have been trained with ELBO loss, note that the implementation differs for each model.

VaDE was trained in two steps. The first step involved training an autoencoder and using its weights as a starting point. Then run a Gaussian Mixture Model on its latent space to get the starting values of the mixture components in VaDE. The second step involved training all of the model's parameters for 100 epochs.

The training runes for VAE and VaDE have been implemented in vae.py and vade.py respectively.

4 Results

All results have been computed in analysis.ipynb notebook.

4.1 Metrics

Firstly I evaluate the performance of VaDE by comparing the clustering with the true classes. I used clustering accuracy and v_measure and got the scores of 0.8167 and 0.7648 respectively.

Secondly, I compare clusters identified in the latent spaces of VaDE and VAE using silhouette_score and calinski_harabasz_score. Values of the metrics are presented in Table 1.

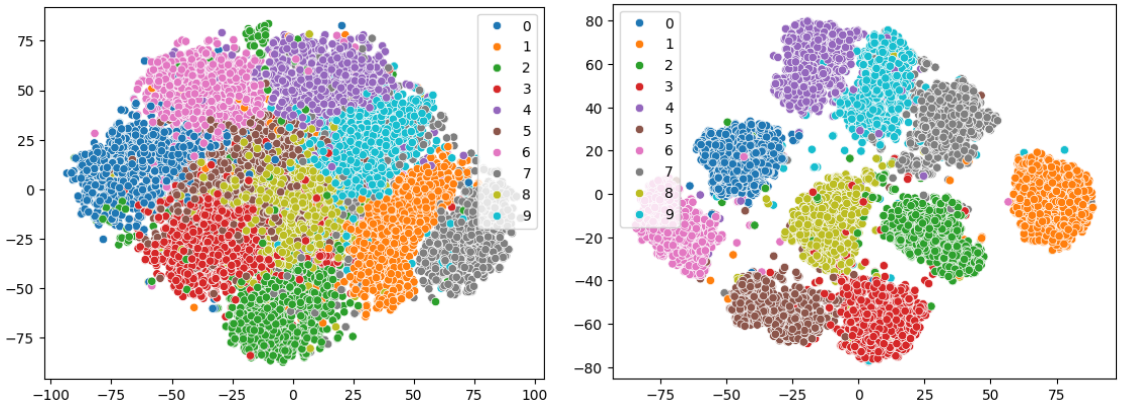
	VAE	VaDE
silhouette_score	0.089	0.189
calinski_harabasz_score	4662.5	7623.4

Table 1: Metrics for VAE and VaDE

Note that all the metrics computed for VaDE are comparable to the ones found in the literature.

4.2 Visualizations

Figure 3 presents the latent spaces of VAE and VaDE.

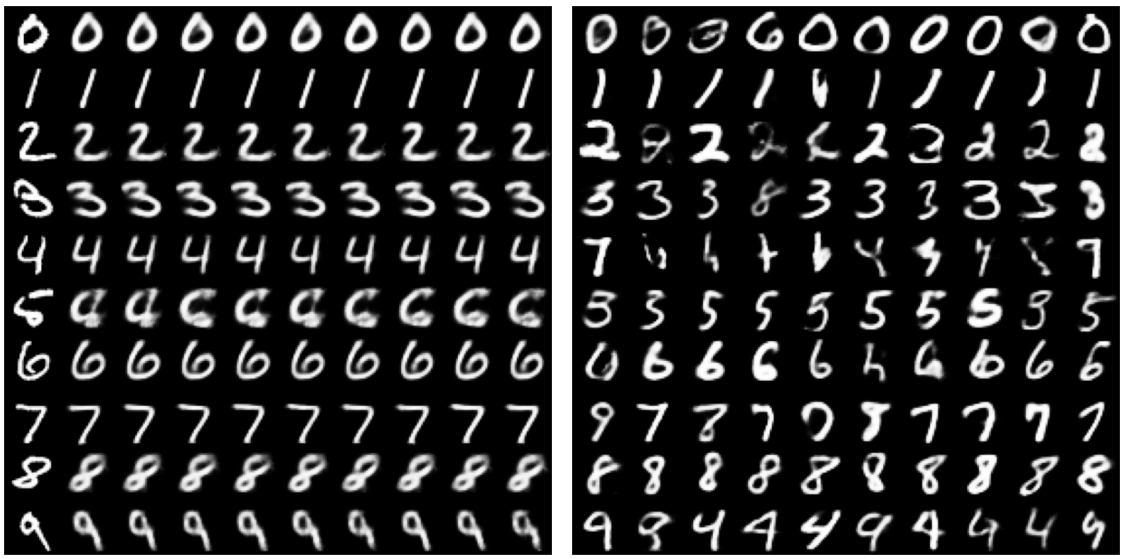


(a) Latent space of VAE

(b) Latent space of VaDE

Figure 3: Latent spaces of VAE and VaDE

Figure 4 depicts observations generated by VAE and VaDE.



(a) Observations generated by VAE

(b) Observations generated by VaDE

Figure 4: Observations generated by VAE and VaDE

The results here seem a bit surprising since VaDE is thought to be generally the better model. However, I have run the code from https://github.com/kozlovskia/Variational_Clustering and got similar results. The image with observations generated by the code from the repository is saved in other/repo_samples.png. This phenomenon may be caused by the incorrect implementation of the loss function. However, I am positive that the loss function is implemented in the same way as in the mentioned repository.

5 Conclusion

In conclusion, this project demonstrates the effectiveness of Variational Autoencoders (VAEs) and their extension, Variational Deep Embedding (VaDE), in handling complex and high-dimensional data for clustering tasks. The experimental results on the MNIST dataset highlight the superior clustering performance of VaDE, as evidenced by higher silhouette and Calinski-Harabasz scores. The visualizations further illustrate the improved cluster separation. On the other hand, VAE seems to outperform VaDE in generating new observations.