Machine Learning Homework 1

Gausse Mael DONGMO KENFACK (董沬高斯)
Student ID: 2024403346

December 2024

Contents

1 Minimum Error Formulation of PCA **1**

2 Deep Generative Model **3**
2.1 Data and Architecture 3
2.2 Bernoulli VAE 3
2.3 Gaussian VAE 5
2.4 Comparison 5

1 Minimum Error Formulation of PCA

Principal Component Analysis (PCA) is a dimensionality reduction technique that works by finding a new set of variables, smaller than the original set of variables, retaining most of the sample's information. It can be understood through two formulations: **maximizing variance** or **minimizing error**. This section focuses on the error minimization perspective.

We have a set of data points $\{\mathbf{x}_n\}$, $n = 1, \ldots, N$, and each $\mathbf{x}_n \in \mathbf{R}$. We also have a set of complete orthonormal basis μ_i, $i = 1, \ldots, p$, where $\mu_i \mu_j = \delta_{ij}$. We want to approximate $\mathbf{x}_n$ by

$$\tilde{\mathbf{x}}_n = \sum_{i=1}^{d} z_{ni}\mu_i + \sum_{i=d+1}^{p} b_i \mu_i,$$

and the objective function is the approximation error

$$J = \frac{1}{N}\sum_{n=1}^{N} \|\mathbf{x}_n - \tilde{\mathbf{x}}_n\|^2.$$

To solve the problem and find the optimal solutions of z_{ni} and b_i which minimize J, we just need to compute the derivative with respect to those variables and set it to 0.

- First let's expand J

$$\begin{aligned}
J &= \frac{1}{N}\sum_{n=1}^{N} \|\mathbf{x}_n - \tilde{\mathbf{x}}_n\|^2 \\
&= \frac{1}{N}\sum_{n=1}^{N} \|\mathbf{x}_n\|^2 - 2\mathbf{x}_n^\top \tilde{\mathbf{x}}_n + \|\tilde{\mathbf{x}}_n\|^2 \\
&= \frac{1}{N}\sum_{n=1}^{N} \|\mathbf{x}_n\|^2 - 2\mathbf{x}_n^\top \left(\sum_{i=1}^{d} z_{ni}\mu_i + \sum_{i=d+1}^{p} b_i\mu_i\right) + \left\|\sum_{i=1}^{d} z_{ni}\mu_i + \sum_{i=d+1}^{p} b_i\mu_i\right\|^2 \\
&= \frac{1}{N}\sum_{n=1}^{N} \left(\|\mathbf{x}_n\|^2 - 2\sum_{i=1}^{d} z_{ni}(\mathbf{x}_n^\top\mu_i) - 2\sum_{i=d+1}^{p} b_i(\mathbf{x}_n^\top\mu_i) + \sum_{i=1}^{d} z_{ni}^2 + \sum_{i=d+1}^{p} b_i^2\right)
\end{aligned}$$

  due to the orthogonality of the μ_i

- Derivative of J with respect to z_{ni}

$$\frac{\partial J}{\partial z_{ni}} = \frac{2}{N}\sum_{n=1}^{N}\left(-\sum_{i=1}^{d}(\mathbf{x}_n^\top\mu_i) + \sum_{i=1}^{d} z_{ni}\right)$$

  setting $\frac{\partial J}{\partial z_{ni}} = 0$ we have

$$\begin{aligned}
\frac{2}{N}\sum_{n=1}^{N}\left(-\sum_{i=1}^{d}(\mathbf{x}_n^\top\mu_i) + \sum_{i=1}^{d} z_{ni}\right) &= 0 \\
\implies \sum_{n=1}^{N}\sum_{i=1}^{d} z_{ni} &= \sum_{n=1}^{N}\sum_{i=1}^{d}(\mathbf{x}_n^\top\mu_i) \\
\implies z_{ni} = \mathbf{x}_n^\top\mu_i, i = 1,\ldots,d, n = 1,\ldots,N
\end{aligned}$$

- Derivative of J with respect to b_i

$$\frac{\partial J}{\partial b_i} = \frac{2}{N}\sum_{n=1}^{N}\left(\sum_{i=d+1}^{p} b_i - \sum_{i=d+1}^{p}(\mathbf{x}_n^\top\mu_i)\right)$$

setting $\frac{\partial J}{\partial b_i} = 0$ we have

$$\begin{aligned}
&\frac{2}{N}\sum_{n=1}^{N}\left(\sum_{i=d+1}^{p} b_i - \sum_{i=d+1}^{p} (\mathbf{x}_n^\top \mu_i)\right) = 0 \\
&\implies \sum_{n=1}^{N}\sum_{i=d+1}^{p} b_i = \sum_{n=1}^{N}\sum_{i=d+1}^{p} (\mathbf{x}_n^\top \mu_i) \\
&\implies N\sum_{i=d+1}^{p} b_i = \sum_{i=d+1}^{p}\sum_{n=1}^{N} (\mathbf{x}_n^\top \mu_i) \\
&\implies \sum_{i=d+1}^{p} b_i = \sum_{i=d+1}^{p} \frac{\mu_i}{N}\sum_{n=1}^{N} \mathbf{x}_n^\top \\
&\implies \sum_{i=d+1}^{p} b_i = \sum_{i=d+1}^{p} (\bar{\mathbf{x}}_n^\top \mu_i) \\
&\implies b_i = \bar{\mathbf{x}}_n^\top \mu_i, i = d+1, \ldots, p.
\end{aligned}$$

Those are the optimal solutions.

2 Deep Generative Model

In generative AI, Variational Auto-encoders (VAEs) are powerful tools for visualizing high-dimensional data in a meaningful, lower-dimensional latent space. In this section, we explore two types of VAEs, each employing a different decoder: one with a **Bernoulli decoder** and the other with a **Gaussian decoder**.

For this problem, the latent variable $\mathbf{z}$ is fixed to a dimension of 40. We will construct and train both VAEs on the MNIST dataset, then compare the quality of their generated outputs to better understand the impact of the decoder choice.

2.1 Data and Architecture

To implement and evaluate the Variational Autoencoders (VAEs), we started by preparing the MNIST dataset. For the Bernoulli VAE, pixel values were binarized by thresholding to either 0 or 1, aligning with the Bernoulli decoder's probabilistic assumptions. For the Gaussian VAE, the pixel values were normalized to the $[0, 1]$ range to ensure compatibility with the Gaussian decoder.

The encoder architecture was shared between both VAEs. It consisted of an input layer that accepted flattened MNIST images (28×28=784 dimensions), followed by a fully connected hidden layer with 128 neurons and tanh activation.

Both VAEs were trained using the ELBO loss, which consists of the reconstruction loss and a KL divergence term. We used the Adam optimizer and trained the models for 30 epochs.

2.2 Bernoulli VAE

The Bernoulli VAE used a decoder that predicted probabilities for each pixel with a sigmoid activation function and the reconstruction loss was computed using binary cross-entropy.

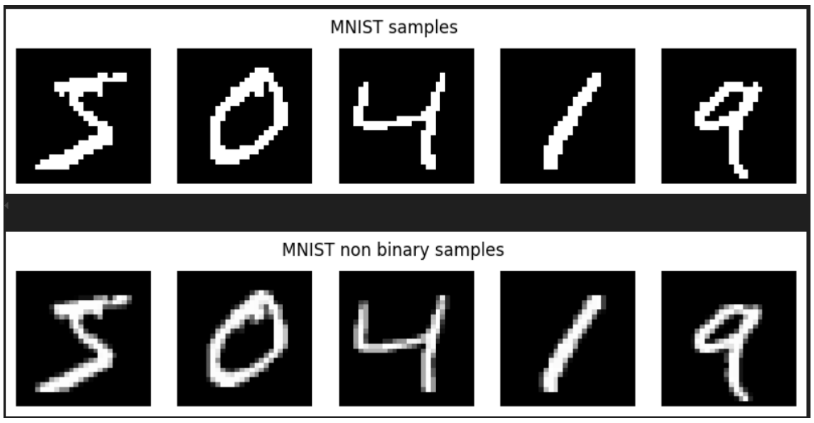

Figure 1: Example of Data of the MNIST dataset

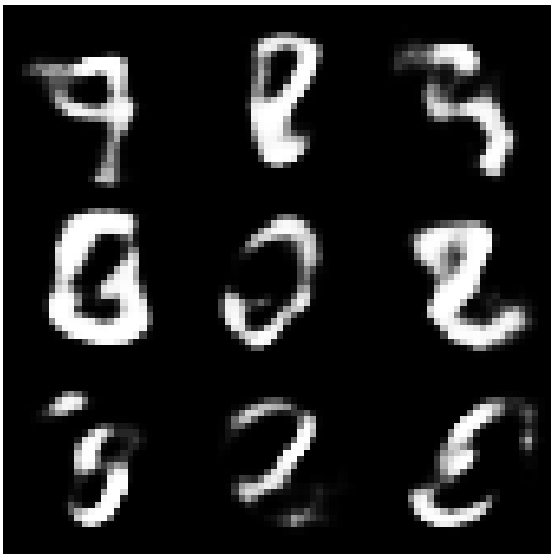

Figure 2: Sample generated by the Bernoulli decoder after only 30 epochs

2.3 Gaussian VAE

The Gaussian VAE, on the other hand, employed a decoder with a linear activation function that predicted mean values for the pixel intensities. We trained differents models dependending of the value of σ. Here are the results

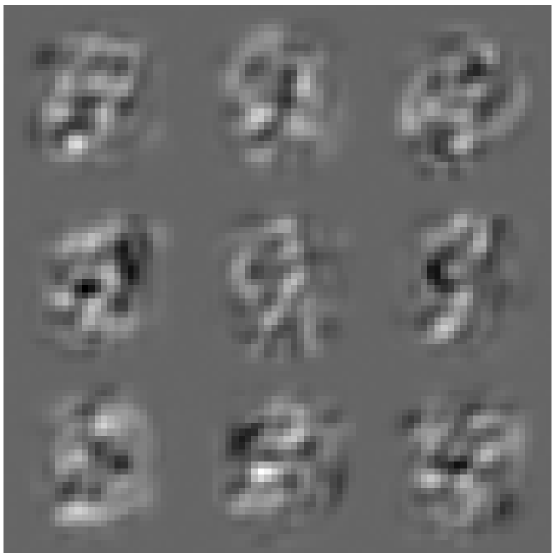

Figure 3: Sample generated by the Gaussian decoder after 30 epochs ($\sigma = 0.01$)

2.4 Comparison

The key differences between the two encoders can be seen the following table.

Feature	**Bernoulli Decoder**	**Gaussian Decoder**
Assumption on $\mathbf{x}$	Binary data (0 or 1)	Continuous data ([0, 1] pixel values)
Modeled distribution	Bernoulli($\mu_\theta(\mathbf{z})$)	$\mathcal{N}(\mu_\theta(\mathbf{z}), \sigma^2 I)$
Output activation	Sigmoid	None (linear output)
Reconstruction loss	Binary cross-entropy	Mean squared error
Output type	Probability (0 to 1)	Continuous value
Best suited for	Binarized data	Normalized or continuous data

We generated some images with different inputs to see the outputs and comment it. In the following images the Bernoulli encoder is in the middle, the Gaussian at the end, and the first image is the input image.

The Gaussian decoder handles continuous data well and captures the overall structure of the digits. The samples show some level of diversity, indicating that the latent space is being used effectively.

The Bernoulli decoder typically excels at generating clear, sharp images for datasets where the pixel values are binary or can be approximated as such. This makes it particularly effective for datasets like binarized MNIST.

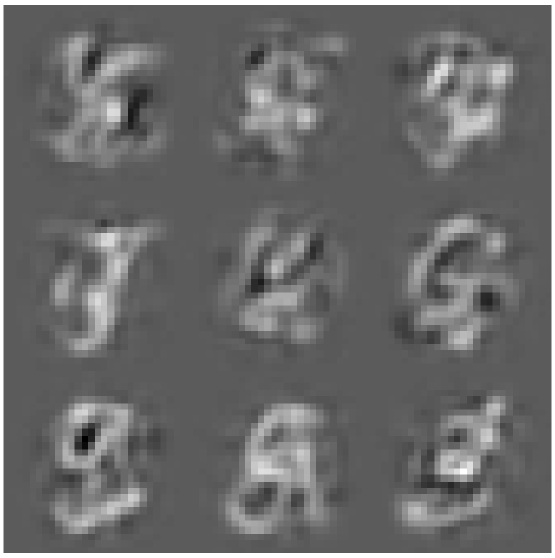

Figure 4: Sample generated by the Gaussian decoder after 30 epochs ($\sigma = 0.1$)

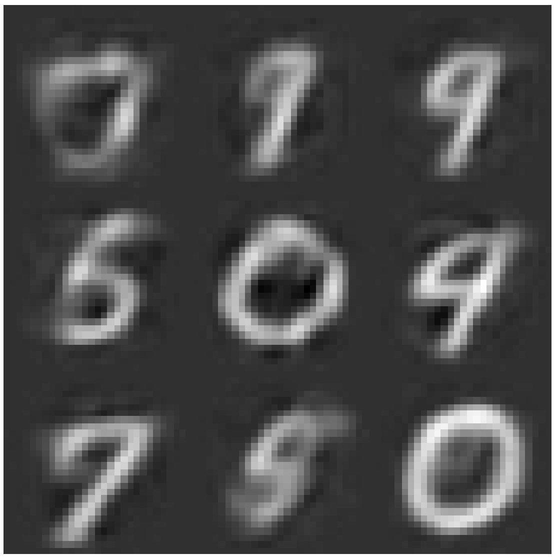

Figure 5: Sample generated by the Gaussian decoder after 30 epochs ($\sigma = 1$)

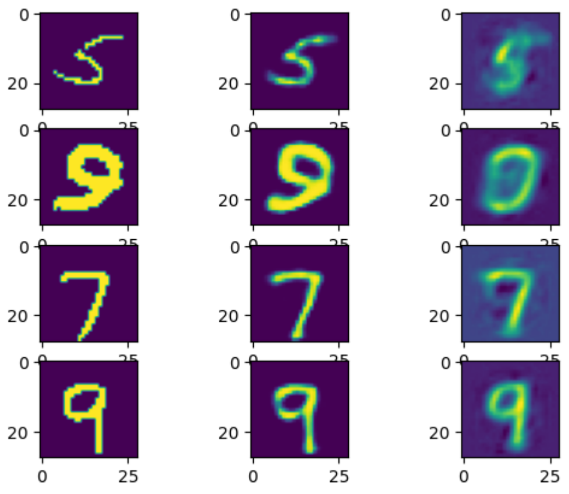

Figure 6: Generating samples from binary inputs

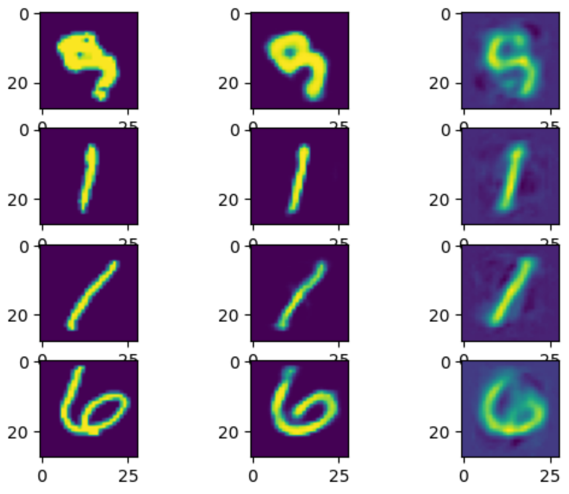

Figure 7: Generating samples from non-binary inputs

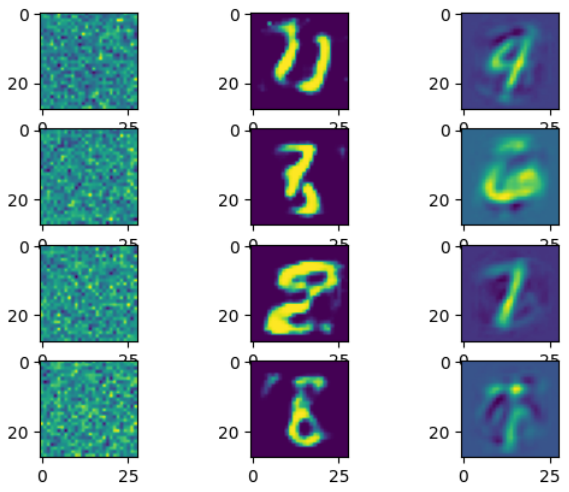

Figure 8: Generating samples from random inputs

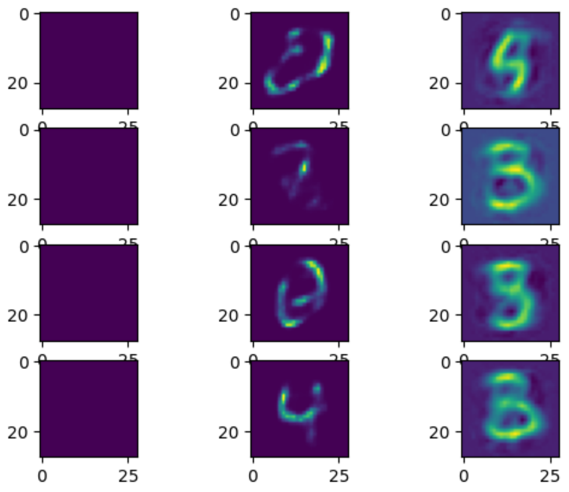

Figure 9: Generating samples from blank inputs