Advanced Deep Learning 2024 Assignment 2

This is an **individual assignment** and its deadline is **Tuesday, May 14, 2024, 22:00**. You must submit your solution electronically via the Absalon home page.

1 Autoencoders

This assignment is about projecting high-dimensional to low-dimensional spaces to achieve a more compact representation.

In the file, `autoencoders.ipynb`, accompanying this assignment, you will find a Jupyter notebook with a working autoencoder, with a model similar to:

```
class FF(nn.Module):
    def __init__(self,dim1,dim2,dim3):
        super().__init__()
        self.main = nn.Sequential(
            nn.Linear(in_features=dim1, out_features=dim2),
            nn.ReLU(),
            nn.Linear(in_features=dim2, out_features=dim3)
        )

    def forward(self, input):
        return self.main(input)
class Autoencoder(nn.Module):
    def __init__(self, dim1, dim2, dim3):
        super().__init__()
        self.encoder = FF(dim1, dim2, dim3)
        self.decoder = nn.Sequential(
            FF(dim3, dim2, dim1),
            nn.Sigmoid()
        )

    def forward(self, x):
        x = self.encoder(x)
        x = self.decoder(x)
        return x
```

The notebook analyzes the thousands of images of handwritten digits (0–9) in the MNIST dataset illustrated in Figure 1, and when executed, it produces 2 plots similar to those shown in Figure 2. Your task is to:

1. Run `autoencoders.ipynb` at least 3 times and give a qualitative comparison between the resulting plots.

Principle Component Analysis (PCA) is a non-deep-learning method for encoding high-dimensional data in lower dimensions. Given a M samples of N-dimensional vectors, (μ, C) the mean vector and the covariance matrix of the samples, and (λ_i, v_i) the eigenvalues and -vectors of C, a new data-point x can be encoded as the vector-dot-product,

$$\alpha_i = (x - \mu) \cdot v_i, \qquad (1)$$

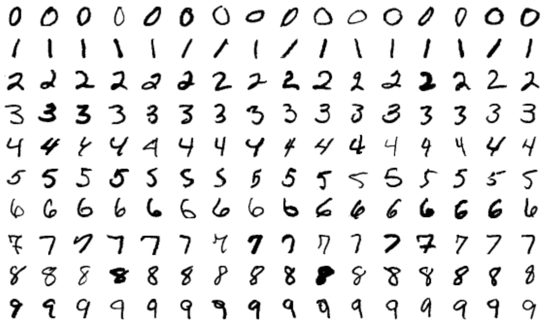

Figure 1: Examples of images in the MNIST dataset [”THE MNIST DATABASE of handwritten digits”. Yann LeCun, Courant Institute, NYU Corinna Cortes, Google Labs, New York Christopher J.C. Burges, Microsoft Research, Redmond].

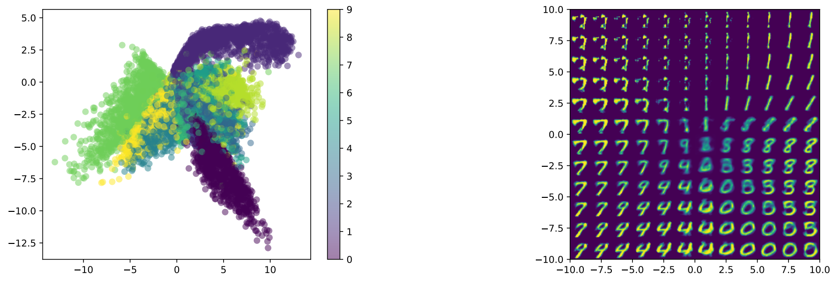

Figure 2: Example of the projection of images from MNIST to the latent space (left), and the resulting images from different latent space values (right).

and new data can be generated by,

$$x = \mu + \sum_{i=0}^{K} \alpha_i \sqrt{\lambda_i} v_i, \tag{2}$$

for some $K < \min(M, N)$ and parameters α_i. Your task is to:

2. Make a baseline comparison of the autoencoder above with Principle Component Analysis (PCA), e.g., using the `decomposition.PCA` module in scikit-learn or your favorite PCA tool. You are to produce a 2-dimensional scatter plot and generate example images similar to `autoencoders.ipynb` based on PCA, and you are to give a qualitative comparison of the two models. Consider also, if you run your PCA program twice, will the result change? Note that in scikit-learn's PCA object, the mean vector μ is called `mean_`, the variance λ_i `explained_variance_[i]`, and the corresponding vector `components_.T[:,i]`

3. For the PCA, the number of eigenvectors to use in the reconstruction is a model choice, as is the dimensionality of the latent space in the autoencoder. Define a range of values for the number of eigenvectors and latent dimensions respectively (dim3), e.g., from 2 to 20, and plot the reconstruction errors as a function of dim3. In your opinion, how does the number of eigenvectors used in the PCA model compare with the number of latent variables in the autoencoder, and is there a simple way to determine the optimal value of dim3 for each of the methods?

Inputs in autoencoders are mapped deterministically to a latent vector. To make the models more robust at decoding latent variables, we use variational autoencoders, wherein inputs are mapped to a probability distribution over latent vectors, and a latent vector is then sampled from that distribution.

In the file, `vae.ipynb`, accompanying this assignment, you will find a Jupyter notebook with a working variational autoencoder, where the autoencoder class is replaced with a VAE class, which has an encoding method that outputs both the mean and the log variance of the latent distribution and samples from this distribution by reparameterization. The loss function is also adjusted to include both the reconstruction loss and the KL divergence term.

4. Run `vae.ipynb` to train a variational autoencoder, sample from the latent space according to a standard normal distribution, and generate new reconstructed images by passing the sampled latent vectors through the decoder. How does the result of the variational autoencoder qualitatively compare with the autoencoder from task 1 above? Next plot the latent representations of the variational autoencoder and qualitatively compare them with those obtained from the autoencoder. Finally, run `plot_generated_samples` to plot the generated samples. Do you see any differences in the generated images when running the function multiple times?