From PCA To VAE: A Comprehensive Survey

Project Category: General Machine Learning
Project Mentor: Jeff Z. HaoChen

Name: Hongyue Li
SUNet ID: lhy
Department of Computer Science
Stanford University
`lhy@stanford.edu`

1 Introduction

Dimensionality reduction is a fundamental tool in machine learning , aimed at transforming high-dimensional data into a lower-dimensional representation while preserving the essential information. Principal Component Analysis (PCA) has been widely used for this purpose[1]. However, PCA is limited to linear transformations and has no generative power. In order to learn non-linear latent features and to generate samples, people have developed more complex models such as Autoencoders, factor analysis, and Variational Autoencoders(VAE). In this paper, we compare the performance of these models, and show that VAE exhibits strong performance over MNIST dataset.

Although Variational Autoencoders is a powerful architecture capable of representation learning and generative modeling, the standard VAE often experiences the disentaglement problem[2] as the prior over the latent variables is commonly an isotropic Gaussian. We then show that by choosing a mixture of gaussian prior this problem can be effectively mitigated.

2 Related Work

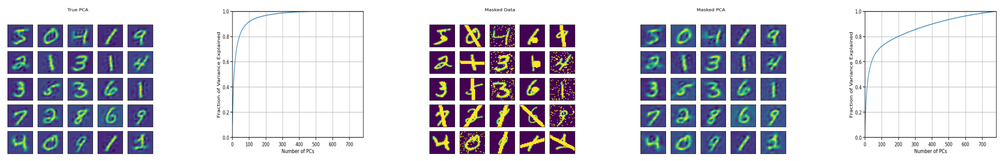
Principle Component Analysis, or PCA, is a technique that is widerly used for applications such as dimensionality reduction, lossy data compression, feature extraction, and data visualization[1]. PCA projects data onto a lower dimensional subspace that maximizes the variance of the projected data and minimizes the average projection cost. Since then, people have extended PCA in various forms, such as probabilistic PCA, kernel PCA, factor analysis, etc[3].

Kingma and Welling[4] developed the Variational Autoencoder model in 2013 that extends the EM algorithms to more complex models parameterized by neural networks. VAE and PCA have been shown to have close connections. Rolinek et al. 2019[5] explained the effectiveness of VAE as VAE pursue the directions learned by PCA in the following sense: "The diagonal approximation in the encoder together with the inherent stochasticity force local orthogonality of the decoder. The local behavior of promoting both reconstruction and orthogonality matches closely how the PCA embedding is chosen."

3 Methods

3.1 Principal Component Analysis (PCA)

We performed PCA on the true MNIST dataset, and we also created a masked dataset where we randomly masked out pixels in each image. As we can see below, even with masked data, PCA with 50 principle components can capture enough characteristics to reconstruct the digits.



(a) (b) (c) (d) (e)

Figure 1: (a)PCA reconstructed digits using true data, dim=50. (b) Fraction of variance explained for true PCA. (c) Examples of masked digits. (d) PCA reconstructed digits using masked images, dim=50. (e) Fraction of variance explained using masked digits.

3.2 Autoencoder (AE)

We implemented an Autoencoder with latent dimension 3. Both the encoder and decoder have only one hidden layer of dimension 256. All layers are fully connected with Relu activations between them. We train for 20 epochs with Adam optimizer to minimize the reconstruction MSE loss.

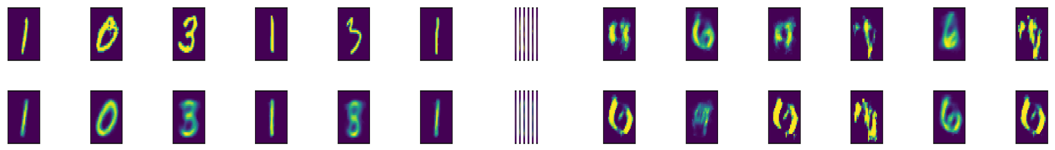


Figure 2: Left: The top row are the original images, while the bottom row are the reconstructed images with only 3 latent dimensions. Right: Generated images of the decoder from random latent variables.

As we can see, 3 latent dimensions are enough to capture most traits of the digits, but Autoencoder has no ability to generate random images.

3.3 Factor Analysis

In order to learn the latent structure and have generative power, we first consider the Factor Analysis Model. We assume the variations in styles of the digits come from a gaussian random variable plus a small gaussian noise. We choose 4 latent dimensions and perform the EM algorithm with 100 iterations. As we can see, although factor analysis model has some generative power, more than half of the generated images are still unrecognizable.

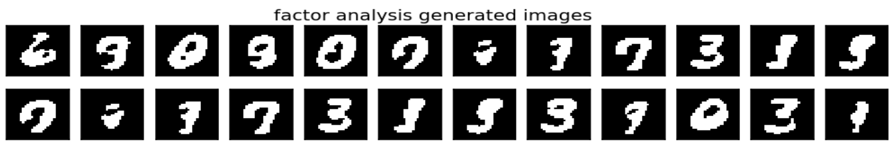
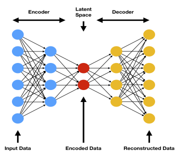
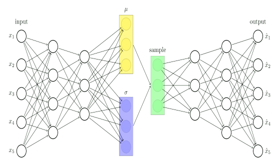


Figure 3: digits generated by factor analysis model

3.4 Variational Autoencoder (VAE)



(a) Autoencoder



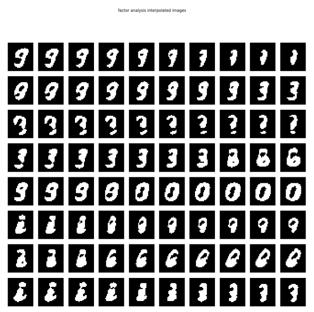
(b) VAE

Figure 4: comparison between the structures of Autoencoder and Variational Autoencoder

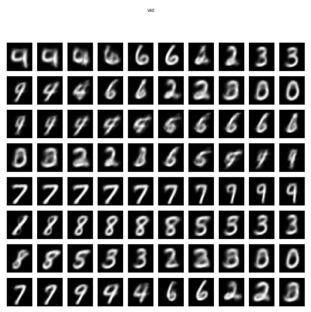
For better generated images, We implemented a Variational Autoencoder. We used a binarized version of the dataset in which each pixel value is either 0 or 1. We represent a 28×28 image $\mathbf{x}$ as a flattened 784 dimensional vector of binary values, i.e. $\mathbf{x} \in \{0, 1\}^{784}$.

To train our VAE, our goal is to maximize the Evidence Lower Bound (ELBO). As the exact posterior $p(\mathbf{z} \mid \mathbf{x})$ is unknown, we used an approximate, amortized posterior $q_{\boldsymbol{\phi}}(\mathbf{z} \mid \mathbf{x}) = \mathcal{N}(\mathbf{z} \mid \mu_{\boldsymbol{\phi}}(\mathbf{x}), \text{diag}(\sigma^2_{\boldsymbol{\phi}}(\mathbf{x})))$. We let our encoder be $E_{\boldsymbol{\phi}}(\mathbf{x}) := \left(\mu_{\boldsymbol{\phi}}(\mathbf{x}), \log \sigma^2_{\boldsymbol{\phi}}(\mathbf{x})\right)$. We parametrize $E_{\boldsymbol{\phi}}$ as a neural network with two layers of hidden units and ReLU activations. We used 512 hidden units in the first layer and 256 in the second. Then we let $\mu_{\boldsymbol{\phi}}$ and $\log \sigma^2_{\boldsymbol{\phi}}$ be affine functions of the hidden layer activations.

We specify our generative model as: $\mathbf{z} \sim \mathcal{N}(0, I), \quad \mathbf{x} \mid \mathbf{z} \sim \text{Bernoulli}(\sigma(D_{\boldsymbol{\theta}}(\mathbf{z}))$ Here, $D_{\boldsymbol{\theta}} : \mathbb{R}^2 \to \mathbb{R}^{784}$ is the decoder. We parametrized $D_{\boldsymbol{\theta}}$ as a fully connected neural network with two hidden layers and ReLU activations. For symmetry, we used 256 units in the first hidden layer and 512 in the second.



(a) Factor Analysis



(b) VAE

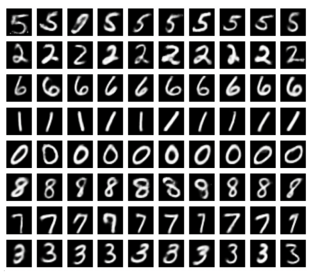

(c) GMVAE

Figure 5: comparison between the interpolated images generated by factor analysis, VAE, and GMVAE

3.5 Additional Experiments

However, since different digits have very different shapes, it is not ideal to simply assume all digits are generated from the same $z \sim \mathcal{N}(0, I)$. This choice of prior causes each dimension of the multivariate Gaussian to be pushed towards learning a separate continuous factor of variation from the data, which can result in learned representations that are structured and disentangled[6]. For example, we should better assume the digit '2' and digit '6' come from different latent distributions. Ideally, we want a mixture of Gaussian priors with 10 components, one for each digit. In subsequent sections we show that a better choice of prior can indeed result in better performance.

3.5.1 Factor Analysis For Each Digit

If we first cluster all digits and then do factor analysis, one for each digit, we get much better generated digits.

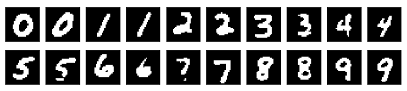


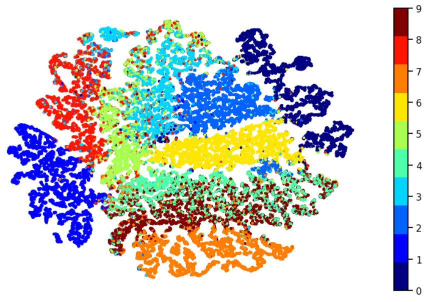
Figure 6: Generated images from factor analysis for each digit.

3.5.2 Gaussian Mixture Variational Autoencoders(GMVAE)

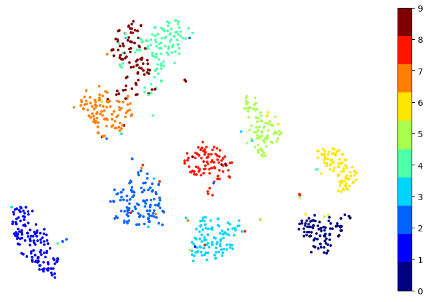
Based on our experiment with factor analysis, we believe there is a need to separate different digits to different parts in our latent distributions. This is usually done by imposing a mixture of Gaussian prior. I first tried to adjust the VAE model myself. However, I encountered difficulty in the inference task as reparametrization trick does not work for discrete variables unless we use Monte Carlo sampling extensively. Luckily, Dilokthanakul et al. [6] adjusted the architecture of the standard VAE and made this work. As we can see in Figure 5, GMVAE yields the best performance when compared with Factor Analysis and standard VAE.

$$\begin{aligned} \omega &\sim \mathcal{N}(0, I) \\ \pi &\sim \text{Dirichlet}(\alpha) \\ z &\sim \text{Categorical}(\pi) \\ x|z,\omega &\sim \mathcal{N}(\mu_z(x;\theta), \text{diag}(\sigma_z^2(x;\theta))) \\ y|x &\sim \text{Bernoulli}(\sigma(D_\theta(x))) \end{aligned} \quad \Big| \quad \begin{aligned} \omega &\sim \mathcal{N}(0, I) \\ z &\sim \text{Multinomial}(\pi) \\ x|z,\omega &\sim \prod_{k=1}^{K} \mathcal{N}(\mu_{z_k(\omega;\beta),\text{diag}(\sigma^2_{z_k}(\omega;\beta))})^{z_k} \\ y|x &\sim \mathcal{N}(\mu(x;\theta), \text{diag}(\sigma^2(x;\theta))) \end{aligned}$$

My proposed probabilistic model is shown on the left hand side, while the model proposed by Dilokthanakul et al. 2017 is shown on the right.



(a) t-SNE plot for VAE



(b) t-SNE plot for GMVAE

Figure 7: With a Gaussian Mixture prior, the latent embeddings among different digits are much more far apart, successfully disentangled the latent embeddings.

4 Conclusion

$$\begin{tikzcd} \text{PCA} \arrow[r, "Randomness"] \arrow[d, "Non-Linearity"'] & \text{Factor Analysis} \arrow[d, "Non-Linearity"] \\ \text{AE} \arrow[r, "Randomness"] & \text{Variational AE} \end{tikzcd}$$

We started from PCA, and showed that with 50 principle directions the model can reconstruct digits effectively. Then we found out that with Autoencoders, 3 latent dimensions is enough to capture most information of the digits, a huge improvement from PCA due to its ability to capture non-linear information.

However, both models lack generative ability, so we experimented with factor analysis and VAEs. Although factor analysis generate image poorly when we apply it directly on all digits, it can generate digits satisfactorily when we apply factor analysis on separate digits. This simple experiment reveals the disentanglement problem, which is not so apparent in the VAE case due to its deep network and ability to learn a separate continuous factor of variation from the data. In order to address this issue, it is natural to impose a mixture of Gaussian prior over the model. However, my initial attempt is not successful due to the technical difficulty that reparametrization only works for certain distributions, and does not work for categorical distribution. Finally, we improved the structure of our VAE by applying the Gaussian Mixture VAE(GMVAE) model developed in [6], which successfully addressed this problem.

5 Appendices

5.1 EM algorithm for Factor Analysis

We derive the EM algorithm for Factor Analysis here, keeping all important steps while omitting tedious calculations.

Setup: Observations $x \in \mathbb{R}^d$, latent variables $z \in \mathbb{R}^k$, $x = \Lambda z + \epsilon$, where $z \sim N(0, I), \Lambda \in \mathbb{R}^{d \times k}$ is the factor loading matrix, $\epsilon \sim N(0, \Phi)$ is the noise, here Φ is a diagonal matrix.

Necessary Probability Distributions:

Joint distribution: $p(x, z|\theta) = \mathcal{N}(\binom{0}{0}, \begin{pmatrix} I & \Lambda^T \\ \Lambda & \Lambda\Lambda^T + \Phi \end{pmatrix})$

Posterior distribution: $p(z|x, \theta) = \mathcal{N}(\Lambda^T(\Lambda\Lambda^T + \Phi)^{-1}x, I - \Lambda^T(\Lambda\Lambda^T + \Phi)^{-1}\Lambda)$

To reach equality in Jensen's inequality, we let $q(z) = p(z|x, \theta)$.

E-Step: ELBO $= \mathbb{E}_{z\sim q}[\log p(x, z|\theta)] = \sum_{i=1}^N \mathbb{E}_{z_i\sim q_i}[\log p(x_i|z_i, \theta) + \log p(z_i)]$

M-step: Taking gradient of ELBO with respect to Λ and set it to zero,

$0 = \nabla_\Lambda(\text{ELBO}) = \nabla_\Lambda \sum_{i=1}^N \mathbb{E}_{z_i\sim q_i}[\log p(x_i|z_i, \theta)] = \sum_{i=1}^N \mathbb{E}_{z_i\sim q_i}[\nabla_\Lambda \log p(x_i|z_i, \theta)]$

$0 = \sum_{i=1}^N \mathbb{E}_{z_i\sim q_i}[\nabla_\Lambda(x_i - \Lambda z_i)^T\Phi^{-1}(x_i - \Lambda z_i)] = \sum_{i=1}^N 2\mathbb{E}_{z_i\sim q_i}[\Phi^{-1}(x_i z_i^T - \Lambda z_i z_i^T)]$

So the update for the factor loading matrix is $\Lambda^* = (\sum_i x_i \mathbb{E}_{q_i}[z_i]^T)(\sum_i \mathbb{E}_{q_i}[z_i z_i]^T)^{-1}$.

Similarly, taking gradient with respect to Φ^{-1} and set it to zero,

we get $0 = \frac{N}{2}\Phi - \frac{1}{2}\sum_i(x_i x_i^T - 2\Lambda\mathbb{E}[z_i]x_i^T + \Lambda\mathbb{E}[z_i z_i^T]\Lambda^T)$, so $\Phi^* = \frac{1}{N}\sum_i(x_i x_i^T - \Lambda^*\mathbb{E}[z_i]x_i^T)$. To satisfy the constraint, we set non-diagonal elements of Φ^* to zero.

Remark 5.1. *Notice the similarity between factor analysis and linear regression.*

	Linear Regression	*Factor Analysis*
Model	$y = X\beta + \epsilon$	$x = \Lambda z + \epsilon$
Update Rule	$\beta^* = (X^T X)^{-1}X^T y$	$\Lambda^* = (\sum_i x_i \mathbb{E}_{q_i}[z_i]^T)(\sum_i \mathbb{E}_{q_i}[z_i z_i]^T)^{-1}$

Factor Analysis seems like a version of "probabilistic linear regression".

5.2 Remarks on VAE

The structure of VAE can be found in CS229 lecture notes. Here I just want to make two remarks.

$$(1): \text{ELBO}(x; q) = \log p(x) - D_{KL}(q||p_{z|x})$$

This implies that ELBO is a lower bound of log likelihood and the gap is exactly the KL divergence between the true posterior and the variational distribution q.

$$(2): \text{ELBO}(x; q) = \mathbb{E}_{z\sim q}\log p(x|z) - D_{KL}(q||p_z)$$

The first term $\mathbb{E}_{z\sim q}\log p(x|z)$ is the reconstruction error and the second term $-D_{KL}(q||p_z)$ is the regularization term. Note the similarity between VAE and PCA: $\mathbb{E}_{z\sim q}\log p(x|z)$ is similar to minimizing the reconstruction error $|y - \hat{y}|^2$ in PCA, while $-D_{KL}(q||p_z)$ is similar to the objective of maximizing the variance among $\hat{y}$ in PCA. The two errors correspond exactly to the two formulations of PCA.

6 Contribution

I worked on all parts of this project.

The code for PCA, autoencoder, Factor Analysis, VAE is available here: `https://github.com/ordinarylhy/CS229Project`.

The code for Gaussian Mixture VAE is written by Jhosimar George Arias Figueroa and is available here: `https://github.com/jariasf/GMVAE`.

I am grateful to receive the help from Jeff HaoChen during his office hours.

References

[1] Ian Jolliffe. Principal component analysis. Springer Verlag, New York, 2002.

[2] Emile Mathieu, Tom Rainforth, N. Siddharth, and Yee Whye Teh. Disentangling disentanglement in variational autoencoders, 2019.

[3] W. Wu, D.L. Massart, and S. de Jong. The kernel pca algorithms for wide data. part i: Theory and algorithms, 1997.

[4] Diederik P Kingma and Max Welling. Auto-encoding variational bayes, 2022.

[5] Michal Rolinek, Dominik Zietlow, and Georg Martius. Variational autoencoders pursue pca directions (by accident), 2019.

[6] Nat Dilokthanakul, Pedro A. M. Mediano, Marta Garnelo, Matthew C. H. Lee, Hugh Salimbeni, Kai Arulkumaran, and Murray Shanahan. Deep unsupervised clustering with gaussian mixture variational autoencoders, 2017.