UNIVERSITY OF PATRAS
DEPARTMENT OF COMPUTER AND
INFORMATION TECHNOLOGY ENGINEERS
LABORATORY OF SIGNAL PROCESSING AND
TELECOMMUNICATIONS

Topics Vision Computer

Analysis of CINISTOS-PCA Autoencoders-AE And Variational Autoencoders-VAE

ΔTeacher. Professor Emmanuel Z. Emanuel E. Psarakis
Assistant: Aristides Bifis, Panagiotis Katsos

Patras December 2021

Elementary theory

Analysis of main components

Principal component analysis is process of calculating principal components and using them to build a fiscaling from the data, sometimes using only some of the larger principal components and ignoring the others.

Principal Component Analysis (PCA) is a technique used in applications such as:

- the reduction of the dimensionality of the data,
- the extraction of characteristics and consequently the analysis of data,
- the visualisation of data, and
- the creation of forecasting models.

PCA is also known as the Karhunen-Loeve transformation. PCA can be defined equivalently in the following two ways:

- as the orthogonal projection of the data onto a linear space of smaller dimension than that of the data, known as principal subspace, in such a way as to maximise the dispersion of the data projections in the dimensionality reduction of the data. If the PCA is defined in this way, the first principal co- mponent is the direction that maximises the variance of the projected data. More generally, the m-th **principal component** can be viewed as a direction orthogonal to the previous $m-1$ **principal components** that maximizes the variance of the projected data.
- as the linear projection which minimises the average cost of supply, defined as the mean square distance between

to the data and their corresponding projections. Using this equivalent definition we define a sequence of unit vectors, where:

- the mth vector, is the direction of a line which best describes the data, and
- are orthogonal to the remaining m-1 vectors. This line is defined by minimising its mean square distance from the points. These directions form an orthogonal axis in which the different atomic dimensions of the data are linearly uncorrelated.

The above two different, but equivalent, definitions PCA are discussed below.

Reduction of data dimensionality

As mentioned above, PCA can be thought of as orthonormal linear transformation of the data, in a new coordinate system, such that the largest variance, from some φnumerical projection of the data, falls in the first coordinate and is called the first principal element, the second largest variance the second principal element, and so on.

For this purpose, let's say :

$$X = \{\mathbf{x}_m\}_{m=1}^{M}$$

is our data set, with a population number M and $\mathbf{x}_m$ a continuous euclidean variable in the space R^N . The data set can be placed in a matrix X of dimension $N \times M$.

Our goal now is to project the data in a dimensional space L, with $L < N$, while at the same time we want to maximize the dispersion of the projected data.

We consider that:

- L is given by[1],
- the data are zero-mean[2], i.e:

$$\bar{\mathbf{x}} = \frac{1}{M} \sum_{m=1}^{M} \mathbf{x}_m = \mathbf{0}$$

and

- the register of data covariates, i.e:

$$S = XX^T$$

is known.

To begin, suppose we want to solve the problem for $L = 1$. **Prove** that the desired solution follows from the solution of the following constrained optimization problem:

$$\max_{v1} \mathbf{v}_1^t \, \mathbf{S}\mathbf{v}_1$$
$$\text{subject} : \| v(1) \|_2 = 1$$

where $\| z \|_2$, the l_2 level of the z vector.

Prove also that the solution to the above problem is the eigenvector $\mathbf{v}^\star$ corresponding to the maximum eigenvalue (let λ_1) of the non S[3].

Suppose now that we want to solve the problem for $L = 2$. In this case **prove** that the desired solution follows from the solution of the following constrained optimization problem:

$$\max_{v2} \mathbf{v}_2^t \, \mathbf{S}\mathbf{v}_2$$
$$\text{subject} : \| v(2) \|_2 = 1$$
$$< \mathbf{v}_1^\star , \mathbf{v}_2 > = 0$$

[1] If we are not given L, how could we estimate it? Justify your answer.

[2] If not, what can we do?

[3] Observe that, by definition, the register S is a non-negatively definite register.

where $z|||_2$, the l_2 level of the vector z and $< ., . >$ is the operator of the inner product.

Prove also that the solution of the above problem is the eigenvector $\mathbf{v}^*$ corresponding to the second largest eigenvalue (let λ_2) of the matrix S.

Using the principle of induction, **formulate** the final ($L = L$ optimization problem and explain your solutions in detail.

Let us now consider PCA as the linear projection of our data, which minimizes the average projection cost, defined as the mean square distance between the data and their respective projections.

Principal component analysis is directly related to another matrix factorisation technique, namely the special value distribution (SVD) **analysis of** the X matrix, i.e:

$$X = V \Sigma U^T ,$$

where:

- Σ is a diagonal $N \times$ M matrix of nonnegative numbers σ_k, $k = 1, 2, - - - , max\{N, M\}$ which are called eigenvalues of the matrix X,
- V is an $N \times N$ orthonormal matrix, whose $n-$ th column is the $n-$ th leftmost eigenvector of the matrix X,
- U is an $M \times M$ orthonormal matrix, whose $m-$ th column is the $m-$ th rightmost eigenvector of the matrix X.

Taking into account the above, the codispersant register $= XX^T$ can be written as:

$$\begin{aligned} S &= XX^T = V \Sigma \Sigma^T V^T \\ &= V \Sigma_0 V^T \end{aligned}$$

where Σ_0 is an $N \times N$ quadratic diagonal matrix of the eigenvalues of X.

Our problem can now be formulated as the following optimisation problem:

$$\min_{A} \mathrm{S}\left|\left|-A\right|\right|_{F}^{2}$$
$$\text{subject : rank\{A\}=L}$$

where $\mathrm{C}\left|\left|\left|\right|\right|_{F}$, the Frobenius level of the C[4] register.

Prove that the solution to the above minimization problem is the following:

$$A^{*}=V_{L}\Sigma_{L}V_{L}^{T}$$

where:

- V_L are the first L columns of the V register, and
- Σ_L an $L\times L$ diagonal matrix containing the first L values of the diagonal of the matrix Σ_0.

ΔProcedure

1. Experiment using the MNIST[5] dataset. Specifically, for at least two (2) digits of your choice, calculate:

   (a') the middle digit

   (ϐ') the register of co-dispersants

   (c') the first eight (8) principal components; and

   (d') the digit reconstructions for $L= 1, 8, 16, 64$ and 256 respectively. Plot and annotate your results appropriately.

---

[4] **Prove** that the Frobenius level of a register is equal to *the* l_2 level of the vector resulting from the rows or columns.

[5]The MNIST dataset includes images of handwritten digits with Size 28×28, in the form of vectors of dimension 784. The training data set shall comprise 60 000 vectors and the control data set shall comprise 10 000 vectors.

(e') Plot the error histograms and the reconstruction error for each of the digits you have chosen.

2. On the MNIST training dataset given to you, first apply PCA and calculate the register V_L with $L = 128$.

(a') Reconstruct, with this truncated register, the now compressed training data.

(β') Using this calculated register, produce the compressed control data and present characteristic image pairs (original and compressed image).

PCA Based on nuclei

Suppose now that we have a nonlinear transformation $k(\mathbf{x}) : \mathrm{R}^N \rightarrow \mathrm{R}^L$ and therefore every given $\mathbf{x}_m$ is mapped (projected) to a point $k(\mathbf{x}_m)$.

We can now apply the PCA technique to the space of ha- ϱ characteristics. It is obvious that if:

$$k(X) = [k(\mathbf{x}_1)\ k(\mathbf{x}_2)\ -\ -\ -\ -\ k(\mathbf{x}_M)] \quad (1)$$

is the transformed $L \times M$ register X we can define the register:

$$S_k = k(X)k(X)^T \ .$$

and apply the classical PCA.

List the fundamental differences from the classical technique. We may have assumed that the expected value of our transformed data is zero. What happens when this assumption does not hold; experiment using the MNIST dataset and the non-linear transformation:

$$k(\mathbf{x}) = e^{-\frac{||\times \mathbf{2}||}{0.1_{\dot{2}}}} \quad (2)$$

ΔProcedure

1. For at least two (2) digits of your choice, calculate: (a') the

   average digit

   (ϐ') the register of co-dispersants

   (c') the first eight (8) principal components; and

   (d') the digit reconstructions for $L = 1, 8, 16, 64$ and 256 respectively. Plot and annotate your results appropriately.

   (e') Plot the error histograms and the reconstruction error for each of the digits you have chosen.

AUTOENCODERS

An autoencoder is a neural network that is trained to 'copy' its input to the output. Internally, it consists of a number of hidden layers that are intended to 'efficiently' describe the input data. The network consists of two (2) sections in which they are implemented:

- an encoding function $\mathbf{z} = E(\mathbf{x}; \vartheta)$ where ϑ are the encoder parameters, with $\mathbf{x} \in R^{(N)}$, $\mathbf{z} \in R^{(L)}$ and
- a decoding function $\hat{x} = D(\mathbf{z}; \phi)$ where ϕ are the parameters terms of the decoder, and $\hat{x}$ of the input $\in R^N$ the reconstruction data $\mathbf{x}$.

The 'copying' of the input to the output can be presented as a meaningless process. However, we are not interested in the simple transfer of input data to the output of the network, but in its efficient description. One way to achieve our goal is to limit the size of the output of the encoder relative to that of its input. Such a network is called an undercomplete network

autoencoder. Learning such a 'compressed' representation of the data forces the network to learn the most important features of the training data. The learning process can be described in a simple way by solving the following minimization problem:

$$\min_{\theta,\phi} L(\vartheta, \phi)$$

using the classical backpropagation algorithm

Typical cost functions, which we will use, are:

- the Mean Square Error (MSE) defined as:

$$L(\vartheta, \phi) = \mathrm{E}_{\sim f_X}\left[\frac{1}{N}\| X - D(E(X\ ; \vartheta); \phi)\|_2^2\right] \text{ and}^6$$

- the binary entropy which is defined by the following relation:

$$L(\vartheta, \phi) = -\mathrm{E}_{\sim f_X}\left[X\ \log(D(E(X\ ; \vartheta); \phi)) + (1 - X\ )\log(1 - \mathrm{D}(E(X\ ; \vartheta); \phi))\right].$$

As is easy to see, and we will verify this experimentally, when the network is linear and the cost function is the mean square error, then the undercomplete autoencoder essentially converges to the same solution as PCA. However, autoencoders with non-linear activation functions are generalizations of PCA.

ΔProcedure

In all embodiments, the data must first be centered in order to have a zero mean value. You are advised not to use bias in your network layers. The choice of optimizer, as well as the values of the hyperparameters, is left to your discretion.

1. Construct a full-connected AE, one level at the encoder & decoder respectively and train it on the MNIST dataset:

[6] Where the calculation of the expected value is required, it will be replaced by the arithmetic average over the data belonging to a batch, based on the law of large numbers.

(a') for 40 epochs, with

(β') latent size$= 128$ and

(c') batch size = 250.

Use the binary entropy as a reconstruction cost function.

In every season:

- compare the encoder values with the register V_L of the previous question
- present a plot of the similarity between the two, using any metric you wish.
- what specificity must the particular network have in order for the AE to approach the PCA? Please record your explanation in detail.

2. Design a non-linear, fully connected AE of three surfaces:

   - in the encoder and
   - to the decoder respectively

   and train it on the MNIST training dataset as in Question 1.

   Calculate the number of parameters of your network (based on the size of the intermediate layers you have chosen for your architecture).

3. Train a network of similar architecture to the previous query, but using exactly half the parameters, following the PCA reconstruction logic.

   (a') What should be the relationship between the decoder and the in this case?

(6′) If you use leaky relu activation functions with a slope of 0.2 in the encoder, what activation functions should you use in the decoder and with what slope? Explain your answer.

4. Repeat the above procedure, using pseudo-adaptors of the encoder's edges for the decoder's edges.

5. Provide summary examples of reconstruction and a table with the values of the mean square error of reconstruction of the control data for the three different AEs you have trained.

Probable PCA

We can formulate the probabilistic PCA by introducing a hidden or latent vector t.m. Z whose dimension is identical to the dimension of the **main subspace**, i.e. L, and whose probability density function (spp) is assumed to be normal Gaussian (this is essentially identical to the prior probability density function of t.m. Z), i.e:

$$f_Z(\mathbf{z}) = N\ (\mathbf{z}; \mathbf{0}, \mathbf{I})$$

and respectively or the conditional split of the m2 . X underlying the observations, conditional on the value of the latent variable Z, is also **an isotropic** σ^2 Gaussian scatter plot of the following form:

$$f_{X|Z}\ (\mathbf{x} \mid z) = N\ (\mathbf{x}; W\mathbf{z} + \mu, \sigma^2\mathbf{I})$$

where the expected value of the m2 X is a linear function of τ.μ Z which is multiplied by the $N\times$ M matrix W and summed by the N -dimensional vector μ.

We can also see probabilistic PCA as a regenerative mechanism for data. In particular, an N -dimensional observation X can defined by the following linear transformation of the *L -dimensional* latent variable Z**:**

$$X = WZ + \mu + W$$

where W *is* an isotropic σ^2 Gaussian scatterer of zero mean t.m. This generation mechanism is shown in the following figure.

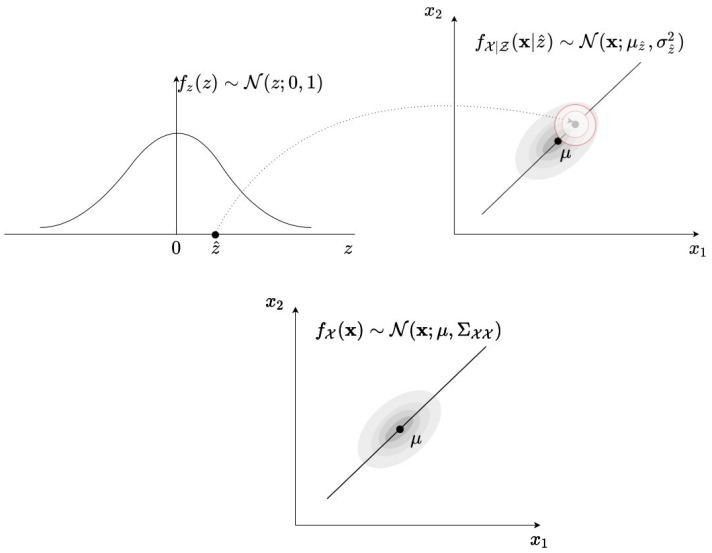

Figure 1: Representation of the latent space in the data space

It is obvious (but nevertheless prove it) that the following relations hold:

$$\begin{aligned} E\{X\} &= E\{WZ + \mu + W\} = \mu \\ \Sigma_{XX} &= E\{(WZ + W)(WZ + W)^T\} \\ &= W E\{ZZ^T\} W^T + E\{WW^t\} \\ &= WW^T + \sigma^2 \mathbf{I}. \end{aligned} \tag{3}$$

and therefore the SPP of the m2. X will be:

$$f_X(\mathbf{x}) = N(\mathbf{x}; \mu, \Sigma_{XX}).$$

Suppose now that we want to calculate the following conditional spp:

$$f_{Z|X}(\mathbf{z} \mid \mathrm{x}) \tag{4}$$

which is known as a posterior split.

If we know the spp $f_{\mathbf{X}}(\mathbf{x})$, then by Bayes' theorem we easily find that:

$$f_{Z|X}(\mathbf{z}|\,\mathrm{x}=\frac{f_{X|Z}(\mathbf{x}|\,\mathrm{z})f_Z(\mathbf{z})}{f_X(\mathbf{x})},$$

where $f_{X|Z}(\mathbf{x}|\,\mathrm{z})$ is the likelihood **function**

Prove that the above conditional SPP, given that relation (4), is given by the following relation:

$$f_{Z|X}(\mathbf{z}|\,\mathrm{x})=N(\mathbf{z};\Sigma_1^{-1}W^T(\mathbf{x}-\mu),\sigma^2\Sigma_1^{-1}), \tag{5}$$

where $\Sigma_1=W(T)W+\sigma^2 I$. It is noteworthy that the posterior value depends on $\mathbf{x}$ while the posterior covariance register does not.

KULLBACK-LEIBLER DIVERGENCE

The Kullback-Leibler Divergence-KLD is a well-known similarity measure[7] which is used to measure the similarity of two distributions. The KLD for $p_X(\mathbf{x})$ and $q_X(\mathbf{x})$ is defined as follows:

$$D_{KL}(p(\mathbf{x})||q(\mathbf{x}))=\mathrm{E}_{\sim p_X}\left[\log\frac{p_X(\mathbf{x})}{q_X(\mathbf{x})}\right]=\int p(\mathbf{x})\log\left[\frac{p_X(\mathbf{x})}{q_X(\mathbf{x})}\right]dx$$

Notice that:

$$D_{KL}(p_X(\mathbf{x})||q(X)(\mathbf{x}))\geq 0 \tag{6}$$

with equality holding when $p_X(\mathbf{x})=q_X(\mathbf{x})$ (**prove it**).

Suppose that you want to check whether the square meters. X, Y with joint spp $f_{XY}(\mathbf{x},\mathbf{y})$ are statistically independent. Suitably define the KLD that will give a solution to your problem. Relate your answer to the concept of mutual information.

Let us now assume that $f_X(\mathbf{x})$ is unknown and therefore not we can calculate the conditional spp $f_{Z|X}(\mathbf{z}|\,\mathrm{x})$:

- or using Bayes' theorem,

[7] KLD not symmetric, i.e., $D_{KL}(p_X(\mathbf{x})||q(X)(\mathbf{x}))\neq D_{KL}(q_X(\mathbf{x})||p(X)(\mathbf{x}))$. This is why it is referred to divergence rather than metric.

- or, alternatively, through the process of **marginalisation** (marginalization):

$$f_X(\mathbf{x}) = \int f_{X|Z}(\mathbf{x}|z) f_Z(\mathbf{z}) dz,$$

because the calculation is not easy.

In this case, we resort to techniques to approximate the unknown spline from a parametric spline family (usually Gaussian) the which let us denote *by* $\hat{f}_{Z|X}(\mathbf{z}|x;\vartheta)$, where ϑ are the parameters of spπ and for the quality of our approximation we will use KLD, i.e:

$$D_{KL}(\hat{f}_{Z|X}(\mathbf{z}|x;\vartheta)||f_{(Z|X)}(\mathbf{z}|x)). \quad (7)$$

For a given image $\mathbf{x}_m$ and applying Ba- yes's theorem, we have:

$$\begin{aligned} D_{KL}(\hat{f}_{Z|X}(\mathbf{z}|x_{(m)};\vartheta) \int ||f_{Z|X}(\mathbf{z}|x_{(m)}))) &= \\ \int \hat{f}_{Z|X}(\mathbf{z}|x_m;\vartheta) \log \frac{\hat{f}_{Z|X}(\mathbf{z}|\mathbf{x}_m;\vartheta)}{f_{Z|X}(\mathbf{z}|x_{(m)})} dz & \\ = \int \hat{f}_{Z|X}(\mathbf{z}|x_m;\vartheta) \log \frac{\hat{f}_{Z|X}(\mathbf{z}|\mathbf{x}_m;\vartheta) f_X(\mathbf{x}_m)}{f_{X|Z}(\mathbf{x}_m|\mathbf{z}) f_Z(\mathbf{z})} dz & \\ = \int \hat{f}_{Z|X}(\mathbf{z}|x_m;\vartheta) \log \frac{\hat{f}_{Z|X}(\mathbf{z}|\mathbf{x}_m;\vartheta)}{f_Z(\mathbf{z})} dz & \\ + \int \hat{f}_{Z|X}(\mathbf{z}|x_m;\vartheta) \log \frac{f_X(\mathbf{x}_m)()}{f_{X|Z}(\mathbf{x}_m|\mathbf{z})} dz & \\ = D_{KL}(\hat{f}_{Z|X}(\mathbf{z}|x_{(m)};\vartheta)||f_{(Z)}(\mathbf{z})) + \log f_X(\mathbf{x}_{(m)}) & \\ - E_{\sim \hat{f}_{Z|X}(\mathbf{z}|\mathbf{x}_m;\theta)}(\log f_{X|Z}(\mathbf{x}_m|z)) & \quad (8) \end{aligned}$$

So, taking into account Relation (6) we easily arrive at the following inequality:

$$\begin{aligned} \log f_X(\mathbf{x}_m) \geq & -D_{KL}(\hat{f}_{Z|X}(\mathbf{z}|x_{(m)};\vartheta)||f_{(Z)}(\mathbf{z})) \\ & + E_{\sim \hat{f}_{Z|X}(\mathbf{z}|\mathbf{x}_m;\theta)}(\log f_{X|Z}(\mathbf{x}_m|z)). \end{aligned} \quad (9)$$

The right-hand member of the above inequality is known as the Evidence Lower Bound-ELBO, i.e:

$$\begin{aligned} G_m(\vartheta) = & -D_{KL}(\hat{f}_{Z|X}(\mathbf{z}|x_{(m)};\vartheta)|| \\ & f_{(Z)}(\mathbf{z})) + E_{\sim \hat{f}_{Z|X}(\mathbf{z}|x_m;\theta)}(\log f_{X|Z}(\mathbf{x}_m|z)) \end{aligned} \quad (10)$$

and maximizing it leads to maximizing the likelihood of our data. Therefore, by solving the following optimization problem:

$$\max_{\theta} G_m(\vartheta)$$

the optimal solution to our original problem is obtained.

Variational AUTOENCODERS

Variational Autoencoders are a special form of Auto-encoders. From an architectural point of view, the two types of networks have many similarities in the sense that both are created with fully connected or convolutional layers. The difference between the two architectures is that, unlike autoencoders, VAEs have a special type of decoder. In particular, the decoder functions as a generative network. To this end, the VAE encoder is trained so that, in addition to the smallest dimension, the space of hidden latent variables also has certain properties that ensure the regenerative character of the decoder.

Most of the time when we sample from the latent variable space of an AE and feed these samples to the decoder output, the resulting output is not qualitative. VAEs are trained so that the latent variable space is smooth, in the sense that every point in this space produces a valid output through the encoder. An interesting property of VAEs is that one can produce the latent representation of two images and, by selecting the points on the line joining these two representations, interpolate between these two images by inserting each of the points on the line in turn into the decoder.

As in the case of probabilistic PCA, we will resort to techniques to approximate the unknown SPP from a parametric family spp (usually Gaussian) which we denote by $\hat{j}_{Z|X}(\mathbf{z}|\ x;\vartheta)$, where ϑ are the parameters of the SPP and for the quality of our approach

we will use the KLD, that is:

$$D_{KL}(\hat{f}_{Z|X}(\mathbf{z}|x;\vartheta) \,||\, f_{Z|X}(\mathbf{z}|x)). \tag{11}$$

We should also emphasize at this point that in case of VAE and the likelihood function $f_{X|Z}(\mathbf{x}|z)$ ϑ will be approximated, by the decoder, by a parametric family of spp which let us denote *by* $\hat{f}_{X|Z}(\mathbf{x}|z;\phi)$, where ϕ are the parameters of the SPP. For a particular image $\mathbf{x}_m$ and applying Bayes' theorem, we have:

$$\begin{aligned} D_{KL}(\hat{f}_{Z|X}(\mathbf{z}|x_{(m)};\vartheta) \,||\, f_{Z|X}(\mathbf{z}|x_{(m)})) &= \int \hat{f}_{Z|X}(\mathbf{z}|x;\vartheta) \log\left[\frac{\hat{f}_{Z|X}(\mathbf{z}|x_{(m)};\vartheta)}{f_{Z|X}(\mathbf{z}|x_{(m)})}\right] dz \\ &= \int \hat{f}_{Z|X}(\mathbf{z}|\mathbf{x};\vartheta) \log\left[\frac{\hat{f}_{Z|X}(\mathbf{z}|x_{(m)};\vartheta) f_X(\mathbf{x}_m)}{f_{X|Z}(\mathbf{x}_m|\mathbf{z};\phi) f_Z(\mathbf{z})}\right] dz \\ &= \int \hat{f}_{Z|X}(\mathbf{z}|\mathbf{x};\vartheta) \log\left[\frac{\hat{f}_{Z|X}(\mathbf{z}|x_{(m)};\vartheta)}{f_Z(\mathbf{z})}\right] dz \\ &+ \int \hat{f}_{Z|X}(\mathbf{z}|\mathbf{x};\vartheta) \log\left[\frac{f_X(\mathbf{x}_m)}{f_{X|Z}(\mathbf{x}_m|\mathbf{z};\phi)}\right] dz \\ &= D_{KL}(\hat{f}_{Z|X}(\mathbf{z}|x_{(m)};\vartheta) \,||\, f_{(Z)}(\mathbf{z})) + \log f_X(\mathbf{x}_{(m)}) \\ &- E_{\sim \hat{f}_{Z|X}(\mathbf{z}|\mathbf{x}_m;\theta)}(\log f_{X|Z}(\mathbf{x}_m|z;\phi)) \end{aligned} \tag{12}$$

Therefore, taking into account Relation (6), we easily arrive at the following inequality:

$$\begin{aligned} \log f_X(\mathbf{x}_m) \geq & - D_{KL}(\hat{f}_{Z|X}(\mathbf{z}|x_{(m)};\vartheta) \,||\, f_{(Z)}(\mathbf{z})) \\ & + E_{\sim \hat{f}_{Z|X}(\mathbf{z}|\mathbf{x}_m;\theta)}(\log f_{X|Z}(\mathbf{x}_m|z;\phi)) \end{aligned} \tag{13}$$

The right-hand member of the above inequality is also in this case the (Evidence Lower Bound-ELBO), i.e:

$$G_m(\vartheta,\phi) = - D_{KL}(\hat{f}_{Z|X}(\mathbf{z}|x_{(m)};\vartheta) \,||\, f_{(Z)}(\mathbf{z})) + E_{\sim \hat{f}_{Z|X}(\mathbf{z}|x_m;\theta)}(\log \hat{f}_{X|Z}(\mathbf{x}_m|z;\phi)) \tag{14}$$

and its maximization leads to the maximization of the likelihood of our data. Therefore, by solving the following problem

optimisation:

$$\max_{\theta,\phi} G_m(\vartheta, \phi)$$

the optimal solution to our original problem is obtained.

It is obvious that if we define the following function:

$$L_m(\vartheta, \phi) = -G_m(\vartheta, \phi) \quad (15)$$

this is a cost function and is the function used to train the network using the regression algorithm. It is obvious that:

- the first term expresses the distance of the posterior distribution of the latent variables from the normal $N(\mathbf{z}; \mathbf{0}, I)$, while
- the second condition relates to the quality of the reconstruction.

The decoder samples Z from $\hat{f}_{Z|X}(\mathbf{z} | x; \vartheta)$. Knowing by stating that a variable that follows a normal distribution with mean $\mu_{\mathbf{x}_m}(\vartheta)$ and standard deviation $\sigma_{\mathbf{x}_m}(\vartheta)$ can be sampled as:

$$Z = \mu_{\mathbf{x}_m}(\vartheta) + \sigma_{\mathbf{x}_m}(\vartheta) \odot E$$

where E is sampled from a standard multivariate normal distribution of appropriate dimensions. This is known as the 'reparame- trization trick', which is used to transfer the stochasticity to the E term so that the backpropagation algorithm can compute the derivatives at this point.

The KLD term of the cost function shall be calculated by means of a closed formula. The first distribution is assumed to be multivariate normal, with mean $\mu_{\mathbf{x}_m}(\vartheta)$ and covariance matrix $\sigma_{\mathbf{x}_m}(\vartheta)I$ or $N(\mu_{\mathbf{x}_m}(\vartheta), \sigma_{\mathbf{x}_m}(\vartheta)I)$, while the second is the standard multivariate normal distribution $N(0, I)$. In this case the KLD term becomes:

$$-D_{KL}(\hat{f}_{Z|X}(\mathbf{z} | x_m; \vartheta) || f(\mathbf{z})) = -\frac{L}{2} + \frac{1}{2}\sum_{l=1}^{L} \sigma^2_{\mathbf{x}_m}(l; \vartheta) + \frac{1}{2}\sum_{l=1}^{L} \mu^2_{x_m}(l; \vartheta) - \ln(\sigma^2_{x_m}(l; \vartheta)) \quad (16)$$

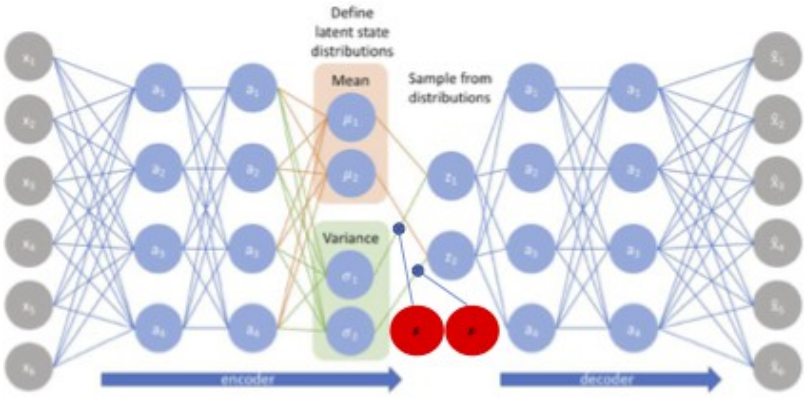

Figure 2: Example of a VAE architecture

ΔProcedure

1. Design a three-level VAE[8]:

   - in the encoder and
   - to the decoder respectively and

   train it:

   - for 100 seasons, with
   - latent size= 2 and
   - batch size= 250.

   For the reconstruction term in the cost function, use the binary entropy.

   (a') Create a batch of noise from N ($\mathbf{z}$; $\mathbf{0}$, I) which is given at each epoch, after the training and control phase, to the decoder. Show how the resulting reconstruction evolves for this batch (reconstruction figures for epochs 1, 50 and 100).

[8] The choice of optimizer as well as the values of the hyperparameters is left to your discretion.

(β′) At the end of the training, create the latent representation of all the control data. Print on a common scatter plot these representations, colouring those corresponding to the same digits with a common colour. Present the plot and record your observations.

Bibliography

[1] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[2] Diederik P. Kingma and Max Welling. an introduction to variational autoencoders. *coRR*, abs/1906.02691, 2019.

[3] Yunfei Teng and Anna Choromanska. invertible autoencoder for domain adaptation.*Computation*, 7(2), 2019.

[4] Philip R. Merrifield. Book reviews : Modern factor analysis by harry h. harman. chicago: university of chicago press, 1960. *educational and Psychological Measurement*, 21(4):1043-1047, 1961.