Universitatea “Alexandru Ioan Cuza” din Iaşi
Facultatea de Informatică

LUCRARE DE DISERTAŢIE

Exploiting a new probabilistic model: S2FA
Simple-Supervised Factor Analysis

propusă de

Student: Sebastian-Adrian Ciobanu
Coordonator ştiinţific: Conf. Dr. Liviu Ciortuz

Sesiunea: iulie
2019

Universitatea “Alexandru Ioan Cuza” din Iaşi
Facultatea de Informatică

Exploiting a new probabilistic model: S2FA

Simple-Supervised Factor Analysis

Student: Sebastian-Adrian Ciobanu
Coordonator ştiinţific: Conf. Dr. Liviu Ciortuz

Sesiunea: iulie
2019

Contents

Chapter 1

Introduction

1.1 Motivation

In **machine learning**, models can be grouped into two categories: **probabilistic** and **non-probabilistic**. **Probabilistic** models can be themselves classified as **generative** and **discriminative** [1]. Examples of classic generative models are *Naive Bayes* and *Gaussian Mixture Model.* Examples of classic discriminative models are *Linear Regression* and *Logistic Regression.* The key difference is whether they model the joint probability of the input and the output (generative) or they just model the conditional probability of the output given the input (discriminative). For a classification or a regression task one may argue that what you need is just a discriminative model, but the generative ones have their **advantages**: can sometimes handle missing data, can easily generate new data, can be extended to be unsupervised or semi-supervised etc. [5, p.268]

As one may notice, there are generative models for **unsupervised** learning which have counterparts in **supervised** learning, even though this is not very widely discussed. One such example is *Gaussian Mixture Model* and *Gaussian Joint Bayes.* Their training/fitting algorithms are very similar, as one may notice, for example in [2, 3] (for brevity, we present only the formulas and omit the notations which are intuitive):

for *Gaussian Joint Bayes*:

$$\pi_j = \frac{1}{n}\sum_{i=1}^{n} 1_{\{z_i=j\}}$$

$$\mu_j = \frac{\sum_{i=1}^{n} 1_{\{z_i=j\}} x_i}{\sum_{i=1}^{n} 1_{\{z_i=j\}}}$$

$$\Sigma_j = \frac{\sum_{i=1}^{n} 1_{\{z_i=j\}} (x_i - \mu_j)(x_i - \mu_j)^\top}{\sum_{i=1}^{n} 1_{\{z_i=j\}}}$$

for *EM/GMM* (for more information on *EM* = *Expectation Maximization* [5, p.349]):

E step

$$w_{ij} \stackrel{\text{not.}}{=} p(z_i = j | x_i; \pi', \mu', \Sigma') = \frac{p(x_i | z_i = j; \mu', \Sigma') p(z_i = j; \pi')}{\sum_{l=1}^{K} p(x_i | z_i = l; \mu', \Sigma') p(z_i = l; \pi')}$$

M step

$$\pi_j = \frac{1}{n}\sum_{i=1}^{n} w_{ij}$$

$$\mu_j = \frac{\sum_{i=1}^{n} w_{ij} x_i}{\sum_{i=1}^{n} w_{ij}}$$

$$\Sigma_j = \frac{\sum_{i=1}^{n} w_{ij} (x_i - \mu_j)(x_i - \mu_j)^\top}{\sum_{i=1}^{n} w_{ij}}$$

This similarity intrigued us and wanted to exploit such a generative model in all the possible ways in order to get some insight and to discover new relationships between models or to create new ones. Because *Gaussian Mixture Model* and *Gaussian Joint Bayes* are widely used we thought that this exploitation may be already done by someone else. As a result, we changed the root model into ***Factor Analysis (FA)*** [4], which is normally used for *dimensionality reduction*. It is a Gaussian generative model used in unsupervised learning. We proposed ourselves to create the supervised counterpart in order to handle a regression task and then exploit it as much as possible.

1.2 Previous work

Although *Factor Analysis* is widely used for dimensionality reduction, its **supervised counterpart** is not present in the literature. What is present is a model called *Supervised PCA* or *Latent factor regression* [5, p.405]. The idea is that not only the input (for a regression task) is generated by a latent variable (as one applies *Factor Analysis* to replace the input in the problem with a low dimensional embedding), but also the output. The key idea is that the purpose of *Supervised PCA* is still dimensionality reduction and not at all regression, which is where we want to push the *Factor Analysis* model.

There is also a term called *Linear Gaussian Systems* [5, p.119] which expresses some properties of the generative process involved in *Factor Analysis*, but does not involve any of the ideas we want to develop.

Factor Analysis is strongly related to *Principal Component Analysis* (PCA) [6], because by imposing a constraint in Factor Analysis, we get a model called *Probabilistic Principal Component Analysis* [7] that can be fitted using a closed-form solution which is also the solution for *PCA*. *Probabilistic PCA* can be kernelized using a model called *Gaussian Process Latent Variable Model* (*GPLVM*) [8]. This model also has supervised counterparts [9], but, as in the case of *FA*, the supervised extension targets dimensionality reduction and the idea is similar to the one in *Supervised PCA*.

1.3 Contributions

We have seen so far that there is little visible work in the literature regarding moving from *FA* to regression. We managed to do the following:

- create the supervised version of *Factor Analysis* which can be used for regression: we called the model **S2FA = Simple-Supervised Factor Analysis**; an important note is that we start with the parameters in *FA* unconstrained (**S2UncFA = Simple-Supervised Unconstrained Factor Analysis)** then go to *S2FA* and then constrain them further to get a **S2PPCA (= Simple-Supervised Principal Component Analysis)** version

- compare the models with *Linear regression* (*LR*): one important result is that *S2UncFA* is (strongly) **equivalent** to *Linear Regression*

- develop some **semi-supervised** algorithms that combine *Factor Analysis* and *regression via Factor Analysis*: we called the models **S3UncFA = Simple Semi-Supervised Unconstrained Factor Analysis**, **S3FA** and **S3PPCA**

- develop a model that encapsulates the supervised and unsupervised versions and also handles **missing data in input or output**: we called the models **MS3UncFA = Missing Simple Semi-Supervised Unconstrained Factor Analysis**, **MS3FA**, **MS3PPCA**.

- discuss **other extensions** we thought of

- create an **R package** with *FA*, *PPCA*, *S2UncFA*, *S2FA*, *S2PPCA*, *S3UncFA*, *S3FA*, *S3PPCA*, *MS3UncFA*, *MS3FA*, *MS3PPCA*

- create **mini-experiments** with the R package in order to highlight how the new models can be used and they remark themselves

These contributions are discussed in detail in this paper in the following sections.

Chapter 2

The algorithms

2.1 Prerequisites

2.1.1 On notations

- we usually work with columns vectors
- $x^{(i)}$ represents the i$^{\text{th}}$ vector from a (training) dataset
- x_j represents the j$^{\text{th}}$ component of the vector x
- $A_{j:}$ represents the j$^{\text{th}}$ row of the matrix A
- $A_{:j}$ represents the j$^{\text{th}}$ column of the matrix A
- $\theta^{(t)}$ represents the parameters at iteration t
- NA comes from *Not Available* and denotes a missing value

2.1.2 On matrices

- One is accustomed to multiply matrices in the following way:

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{bmatrix} \begin{bmatrix} 7 & 8 \\ 9 & 10 \end{bmatrix} = \begin{bmatrix} \begin{bmatrix} 1 & 2 \end{bmatrix} \begin{bmatrix} 7 \\ 9 \end{bmatrix} & \begin{bmatrix} 1 & 2 \end{bmatrix} \begin{bmatrix} 8 \\ 10 \end{bmatrix} \\ \begin{bmatrix} 3 & 4 \end{bmatrix} \begin{bmatrix} 7 \\ 9 \end{bmatrix} & \begin{bmatrix} 3 & 4 \end{bmatrix} \begin{bmatrix} 8 \\ 10 \end{bmatrix} \\ \begin{bmatrix} 5 & 6 \end{bmatrix} \begin{bmatrix} 7 \\ 9 \end{bmatrix} & \begin{bmatrix} 5 & 6 \end{bmatrix} \begin{bmatrix} 8 \\ 10 \end{bmatrix} \end{bmatrix} = \begin{bmatrix} 25 & 28 \\ 57 & 64 \\ 89 & 100 \end{bmatrix}$$

But the same operation can be treated as a sum:

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{bmatrix} \begin{bmatrix} 7 & 8 \\ 9 & 10 \end{bmatrix} = \begin{bmatrix} 1 \\ 3 \\ 5 \end{bmatrix} \begin{bmatrix} 7 & 8 \end{bmatrix} + \begin{bmatrix} 2 \\ 4 \\ 6 \end{bmatrix} \begin{bmatrix} 9 & 10 \end{bmatrix} = \begin{bmatrix} 1 \cdot 7 & 1 \cdot 8 \\ 3 \cdot 7 & 3 \cdot 8 \\ 5 \cdot 7 & 5 \cdot 8 \end{bmatrix} + \begin{bmatrix} 2 \cdot 9 & 2 \cdot 10 \\ 4 \cdot 9 & 4 \cdot 10 \\ 6 \cdot 9 & 6 \cdot 10 \end{bmatrix} = \begin{bmatrix} 25 & 28 \\ 57 & 64 \\ 89 & 100 \end{bmatrix}$$

In a general case we can write:

$$AB^{\top} = \sum_{i=1}^{n} a^{(i)} {b^{(i)}}^{\top}, A \in \mathbb{R}^{m \times n}, B \in \mathbb{R}^{s \times n} \tag{2.1}$$

where $A = \begin{bmatrix} a^{(1)} & a^{(2)} & \dots & a^{(n)} \end{bmatrix}, a^{(i)} \in \mathbb{R}^{m\times 1}, \forall i \in \{1, \dots, n\}$

and $B = \begin{bmatrix} b^{(1)} & b^{(2)} & \dots & b^{(n)} \end{bmatrix}, b^{(i)} \in \mathbb{R}^{s\times 1}, \forall i \in \{1, \dots, n\}$

- Another observation which will be useful is:

$$\begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix} \begin{bmatrix} 1 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 2 & 2 \\ 3 & 3 \end{bmatrix}$$

In general,

$$a \begin{bmatrix} 1 & 1 & \dots & 1 \end{bmatrix} = \begin{bmatrix} a & a & \dots & a \end{bmatrix}, \text{ where } a \in \mathbb{R}^{n\times 1}. \tag{2.2}$$

- Another important observation is:

$$\begin{bmatrix} a & b \end{bmatrix} \begin{bmatrix} c & 0 \\ 0 & d \end{bmatrix} \begin{bmatrix} e \\ f \end{bmatrix} = ace + bdf$$

In general:

$$xMy^\top = \sum_{i=1}^{d} x_i M_{ii} y_i, \text{ where } M \in \mathbb{R}^{d\times d} \text{ - diagonal matrix, } x \in \mathbb{R}^{d\times 1}, y \in \mathbb{R}^{d\times 1} \tag{2.3}$$

- One last comment is the following:

Let $A = \begin{bmatrix} a_{11} & \dots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{n1} & \dots & a_{nn} \end{bmatrix}$.

Then $\text{diag}(A) = \begin{bmatrix} a_{11} & 0 & \dots & 0 \\ 0 & a_{22} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & a_{nn} \end{bmatrix}$ and $\text{Tr}(A) = a_{11} + \dots + a_{nn}$

2.1.3 On Naive/Joint Bayes and EM/GMM

Although the derivation of the formulas used to fit a **Naive/Joint Bayes** model is often omitted, in our context it is important to highlight it and also its similarity to a part of the *EM* algorithm.

As discussed in [5, sect.3.5.1.1], one must write the log-likelihood of the (observed) data. To be more clear, we make the following **notations** irrespective of the model/problem/algorithm (intuitively, X is the input and Z is the output):

$(X, Z)|\theta$ - any random variable (RV) with parameters θ

$(X^{(i)}, Z^{(i)})|\theta \sim (X, Z)|\theta$, for $i \in \{1, \dots, n\}$

$(X^{(1)}, Z^{(1)}), \dots, (X^{(n)}, Z^{(n)})$ - usually independent, given the parameters

RV_D $= ((X^{(1)}, Z^{(1)}), \dots, (X^{(n)}, Z^{(n)}))$ - random variable for the data

D $= ((x^{(1)}, z^{(1)}), \dots, (x^{(n)}, z^{(n)}))$ - the data

In this context, the **log-likelihood of the data** is given by:

$$l_{\text{RV\_D}}(\theta) = \ln p_{\text{RV\_D}|\theta}(D|\theta) \tag{2.4}$$

and we maximize it and obtain θ_{MLE}.

In the **EM/GMM** algorithm, one tries to maximize the log-likelihood of the observed data in a number of iterations and at each iteration a variant of the log-likelihood of the complete data is maximized (for details of a concrete *EM* algorithm see the section on *S3UncFA*). That variant of the log-likelihood of the complete data is almost the same as the log-likelihood of the (observed) data in *Gaussian Joint Bayes*.

To be more clear, we also make the following **notations** irrespective of the model (intuitively, X is the input and Z is the output):

$(X, Z)|\theta$ - any random variable (RV) with parameters θ
$(X^{(i)}, Z^{(i)})|\theta \sim (X, Z)|\theta$, for $i \in \{1, \ldots, n\}$
$(X^{(1)}, Z^{(1)}), \ldots, (X^{(n)}, Z^{(n)})$ - usually independent, given the parameters
RV_Do $= (X^{(1)}, \ldots, X^{(n)})$ - random variable for the observed data
RV_Dc $= ((X^{(1)}, Z^{(1)}), \ldots, (X^{(n)}, Z^{(n)}))$ - random variable for the complete data
$Z^{(1)}, \ldots, Z^{(n)}$ - latent variables
Do $= (x^{(1)}, \ldots, x^{(n)})$ - the observed data
Dc $= ((x^{(1)}, Z^{(1)}), \ldots, (x^{(n)}, Z^{(n)}))$ - the complete data

In this context, the **log-likelihood of the observed data** is given by:

$l_{\text{RV\_Do}}(\theta) = \ln p_{\text{RV\_Do}|\theta}(Do|\theta)$

and the **log-likelihood of the complete data** is given by

$$l_{\text{RV\_Dc}}(\theta) = \ln p_{\text{RV\_Dc}|\theta}(Dc|\theta) \tag{2.5}$$

One can spot a huge similarity between the formulas 2.4 and 2.5. The only difference is that in 2.5, there are latent (unknown) variables involved.

Hence, there could be established some **links** between *Gaussian Joint Bayes* (*GJB*) and *EM/GMM*:

- they both have the **objective** to (try to) **maximize the log-likelihood of the observed data**
- *Gaussian Joint Bayes* is the **supervised counterpart** of *EM/GMM* or the other way round: *EM/GMM* is the unsupervised counterpart of *Gaussian Joint Bayes*
- **at each iteration of *EM/GMM* a variant of the log-likelihood of the complete data is maximized and this function is almost the same as the log-likelihood of the (observed) data in *GJB***; as a result, the M step executes a slightly modified (a generalized) variant of *GJB*; also, informally said, if there were no latent variables, *EM/GMM* would converge in a single iteration and the fitted parameters would be the same as those returned by *GJB* (in fact, this is about **semi-supervised** *EM* and a concrete example of such an algorithm can be found in the section on *S3UncFA*)

The inspiration to create the supervised counterpart of *Factor Analysis* and other extensions also came from these links and the existence of *EM/FA*.

2.1.4 On (general) Factor Analysis

The following formulas are derived in [4] and are relevant for the *FA* algorithm (although Ψ is considered diagonal there, the formulas stay the same even if Ψ is not diagonal).

$z \sim \mathcal{N}(0, I)$, $z \in \mathbb{R}^{d\times 1}$

$x|z \sim \mathcal{N}(\mu + \Lambda z, \Psi)$, $x \in \mathbb{R}^{D\times 1}, \mu \in \mathbb{R}^{D\times 1}, \Lambda \in \mathbb{R}^{D\times d}, \Psi \in \mathbb{R}^{D\times D}$ - symmetric and positive definite matrix

$$\begin{bmatrix} x \\ z \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \mu \\ 0 \end{bmatrix}, \begin{bmatrix} \Lambda\Lambda^\top & \Lambda \\ \Lambda^\top & I \end{bmatrix}\right)$$

$x \sim \mathcal{N}(\mu, \Lambda\Lambda^\top + \Psi)$

$z|x \sim \mathcal{N}(\Lambda^\top(\Lambda\Lambda^\top + \Psi)^{-1}(x - \mu), I - \Lambda^\top(\Lambda\Lambda^\top + \Psi)^{-1}\Lambda)$

For the algorithms we developed, we need that $z \sim \mathcal{N}(\mu_z, \Sigma_z)$ and not $z \sim \mathcal{N}(0, I)$, because z becomes observed data and we want to learn its parameters and not impose something unrealistic like $z \sim \mathcal{N}(0, I)$, although we could have left $z \sim \mathcal{N}(0, I)$ and standardize the output. For more details on this alternative ($z \sim \mathcal{N}(0, I)$) see the section on the **standardization-destandardization** variant of *S2UncFA*.

Although those formulas are stated and derived in [5, sect.4.4.1,4.4.3], their form is different than the form in [4] and so we derived them here. We prefer the latter form because, as stated in [10, sec.3.1.2,4] in the formulas in [5, sect.4.4.1,4.4.3] we would need to apply Woodbury Matrix Inversion to obtain a better solution and another way to obtain that better solution is to proceed as in [4], which we do below.

Let us consider the following model (which is a *Linear Gaussian System*):

$z \sim \mathcal{N}(\mu_z, \Sigma_z)$, $z \in \mathbb{R}^{d\times 1}$, $\mu_z \in \mathbb{R}^{d\times 1}$, $\Sigma_z \in \mathbb{R}^{d\times d}$

$x|z \sim \mathcal{N}(\mu + \Lambda z, \Psi)$, $x \in \mathbb{R}^{D\times 1}, \mu \in \mathbb{R}^{D\times 1}, \Lambda \in \mathbb{R}^{D\times d}$, $\Psi \in \mathbb{R}^{D\times D}$ - symmetric and positive definite matrix

The following model

$z \sim \mathcal{N}(\mu_z, \Sigma_z)$, $z \in \mathbb{R}^{d\times 1}$, $\mu_z \in \mathbb{R}^{d\times 1}$, $\Sigma_z \in \mathbb{R}^{d\times d}$

$\epsilon \sim \mathcal{N}(0, \Psi)$, $\epsilon \in \mathbb{R}^{D\times 1}$, $\Psi \in \mathbb{R}^{D\times D}$ - symmetric and positive definite matrix

$x = \mu + \Lambda z + \epsilon$, $x \in \mathbb{R}^{D\times 1}, \mu \in \mathbb{R}^{D\times 1}, \Lambda \in \mathbb{R}^{D\times d}$

z, ϵ - independent

is equivalent to the earlier model because: if z is known, then z is a constant in $x = \mu + \Lambda z + \epsilon$ and ϵ, being independent of z, will have the same distribution: $\epsilon|z \sim \epsilon \sim \mathcal{N}(0, \Psi)$. So, we just add a constant to a normal distribution and, so, only its expected value is changed: $x|z \sim \mathcal{N}(\mu + \Lambda z, \Psi)$.

We have: $\begin{bmatrix} x \\ z \end{bmatrix} = \begin{bmatrix} \mu \\ 0 \end{bmatrix} + \begin{bmatrix} I & \Lambda \\ 0 & I \end{bmatrix} \begin{bmatrix} \epsilon \\ z \end{bmatrix}$.

Because ϵ and z are normally distributed and independent, $\begin{bmatrix} \epsilon \\ z \end{bmatrix}$ is normally distributed.

From [11, (355)] we obtain that $\begin{bmatrix} x \\ z \end{bmatrix}$ is normally distributed. Because of that, we only need to compute $E[x]$, $E[z]$, $E[xx^\top]$, $E[zz^\top]$, $Cov[x, x]$, $Cov[z, z]$, $Cov[x, z]$, $Cov[z, x]$.

$E[x] = E[\mu + \Lambda z + \epsilon] \overset{\text{lin. of E}}{=} \mu + \Lambda E[z] + E[\epsilon] = \mu + \Lambda\mu_z + 0 = \mu + \Lambda\mu_z$

$E[z] = \mu_z$

$$
\begin{aligned}
Cov[x,x] &\overset{\text{def.}}{=} E[(x-E[x])(x-E[x])^\top] \\
&= E[(\cancel{\mu}+\Lambda z+\epsilon-\cancel{\mu}-\Lambda\mu_z)(\cancel{\mu}+\Lambda z+\epsilon-\cancel{\mu}-\Lambda\mu_z)^\top] \\
&= E[(\Lambda z+\epsilon-\Lambda\mu_z)(z^\top\Lambda^\top+\epsilon^\top-\mu_z^\top\Lambda^\top)], \\
&\qquad \text{since } (A+B)^\top = A^\top+B^\top \text{ and } (AB)^\top = B^\top A^\top \\
&= E[\Lambda zz^\top\Lambda^\top+\Lambda z\epsilon^\top-\Lambda z\mu_z^\top\Lambda^\top+\epsilon z^\top\Lambda^\top+\epsilon\epsilon^\top-\epsilon\mu_z^\top\Lambda^\top- \\
&\qquad -\Lambda\mu_z z^\top\Lambda^\top-\Lambda\mu_z\epsilon^\top+\Lambda\mu_z\mu_z^\top\Lambda^\top] \\
&\overset{\text{lin. of E}}{=} \Lambda E[zz^\top]\Lambda^\top+\Lambda E[z\epsilon^\top]-\Lambda E[z]\mu_z^\top\Lambda^\top+E[\epsilon z^\top]\Lambda^\top+E[\epsilon\epsilon^\top]-E[\epsilon]\mu_z^\top\Lambda^\top- \\
&\qquad -\Lambda\mu_z E[z^\top]\Lambda^\top-\Lambda\mu_z E[\epsilon^\top]+\Lambda\mu_z\mu_z^\top\Lambda^\top
\end{aligned}
$$

It is known that: $Cov[z,z] = E[zz^\top]-E[z]E[z]^\top \Rightarrow E[zz^\top] = Cov[z,z]+E[z]E[z]^\top$.
$E[zz^\top] = Cov[z,z]+E[z]E[z]^\top = \Sigma_z+\mu_z\mu_z^\top$.
$E[z\epsilon^\top] \overset{\text{indep.}}{=} E[z]E[\epsilon^\top] \overset{E[X^\top]=E[X]^\top}{=} E[z]E[\epsilon]^\top = E[z]\cdot 0^\top = 0$
$E[\epsilon\epsilon^\top] = \Psi$
$E[\epsilon^\top] \overset{E[X^\top]=E[X]^\top}{=} E[\epsilon]^\top = 0^\top = 0$
We resume the computation of $Cov[x,x]$.

$$
\begin{aligned}
Cov[x,x] &= \Lambda(\Sigma_z+\mu_z\mu_z^\top)\Lambda^\top-\Lambda\mu_z\mu_z^\top\Lambda^\top+\Psi-\cancel{\Lambda\mu_z\mu_z^\top\Lambda^\top}+\cancel{\Lambda\mu_z\mu_z^\top\Lambda^\top} \\
&= \Lambda\Sigma_z\Lambda^\top+\cancel{\Lambda\mu_z\mu_z^\top\Lambda^\top}-\cancel{\Lambda\mu_z\mu_z^\top\Lambda^\top}+\Psi \\
&= \Lambda\Sigma_z\Lambda^\top+\Psi
\end{aligned}
$$

$$
\begin{aligned}
Cov[x,z] &\overset{\text{def.}}{=} E[(x-E[x])(z-E[z])^\top] \\
&= E[(\cancel{\mu}+\Lambda z+\epsilon-\cancel{\mu}-\Lambda\mu_z)(z-\mu_z)^\top] \\
&= E[(\Lambda z+\epsilon-\Lambda\mu_z)(z^\top-\mu_z^\top)], \text{ since } (A+B)^\top = A^\top+B^\top \\
&= E[\Lambda zz^\top-\Lambda z\mu_z^\top+\epsilon z^\top-\epsilon\mu_z^\top-\Lambda\mu_z z^\top+\Lambda\mu_z\mu_z^\top] \\
&= \Lambda E[zz^\top]-\Lambda E[z]\mu_z^\top+E[\epsilon z^\top]-E[\epsilon]\mu_z^\top-\Lambda\mu_z E[z]^\top+\Lambda\mu_z\mu_z^\top
\end{aligned}
$$

$E[\epsilon z^\top] \overset{\text{indep.}}{=} E[\epsilon]E[z^\top] = 0\cdot E[z^\top] = 0$
We resume the computation of $Cov[x,z]$:

$$
\begin{aligned}
Cov[x,z] &= \Lambda(\Sigma_z+\mu_z\mu_z^\top)-\Lambda\mu_z\mu_z^\top-\cancel{\Lambda\mu_z\mu_z^\top}+\cancel{\Lambda\mu_z\mu_z^\top} \\
&= \Lambda\Sigma_z+\cancel{\Lambda\mu_z\mu_z^\top}-\cancel{\Lambda\mu_z\mu_z^\top} \\
&= \Lambda\Sigma_z
\end{aligned}
$$

$$Cov[z,z] = \Sigma_z$$

$$
\begin{aligned}
Cov[z,x] &= E[(z-E[z])(x-E[x])^\top] \\
&= E[((x-E[x])(z-E[z])^\top)^\top] \\
&= E[(x-E[x])(z-E[z])^\top]^\top \\
&= Cov[x,z]^\top
\end{aligned}
$$

Putting all together, we have that:

$$\begin{bmatrix} x \\ z \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \mu + \Lambda\mu_z \\ \mu_z \end{bmatrix}, \begin{bmatrix} \Lambda\Sigma_z\Lambda^\top + \Psi & \Lambda\Sigma_z \\ (\Lambda\Sigma_z)^\top & \Sigma_z \end{bmatrix}\right)$$

From [11, (349-351)] we obtain:

$x \sim \mathcal{N}(\mu + \Lambda\mu_z, \Lambda\Sigma_z\Lambda^\top + \Psi)$

From [11, (352)] we obtain:

$z|x \sim \mathcal{N}(\mu_z + \Sigma_z\Lambda^\top(\Lambda\Sigma_z\Lambda^\top + \Psi)^{-1}(x - \mu - \Lambda\mu_z), \Sigma_z - \Sigma_z\Lambda^\top(\Lambda\Sigma_z\Lambda^\top + \Psi)^{-1}\Lambda\Sigma_z^\top)$

Putting all together, we have:

$z \sim \mathcal{N}(\mu_z, \Sigma_z),\ z \in \mathbb{R}^{d\times 1},\ \mu_z \in \mathbb{R}^{d\times 1},\ \Sigma_z \in \mathbb{R}^{d\times d}$

$x|z \sim \mathcal{N}(\mu + \Lambda z, \Psi),\ x \in \mathbb{R}^{D\times 1}, \mu \in \mathbb{R}^{D\times 1}, \Lambda \in \mathbb{R}^{D\times d},\ \Psi \in \mathbb{R}^{D\times D}$ - symmetric and positive definite matrix

$$\begin{bmatrix} x \\ z \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \mu + \Lambda\mu_z \\ \mu_z \end{bmatrix}, \begin{bmatrix} \Lambda\Sigma_z\Lambda^\top + \Psi & \Lambda\Sigma_z \\ (\Lambda\Sigma_z)^\top & \Sigma_z \end{bmatrix}\right)$$

$x \sim \mathcal{N}(\mu + \Lambda\mu_z, \Lambda\Sigma_z\Lambda^\top + \Psi)$

$z|x \sim \mathcal{N}(\mu_z + \Sigma_z\Lambda^\top(\Lambda\Sigma_z\Lambda^\top + \Psi)^{-1}(x - \mu - \Lambda\mu_z), \Sigma_z - \Sigma_z\Lambda^\top(\Lambda\Sigma_z\Lambda^\top + \Psi)^{-1}\Lambda\Sigma_z^\top)$

(2.6)

If $z \in \mathbb{R}^{1\times 1}, x \in \mathbb{R}^{1\times 1}$, we immediately derive the following results:

$z \sim \mathcal{N}(c, d^2),\ z \in \mathbb{R}^{1\times 1},\ c \in \mathbb{R}^{1\times 1},\ d \in \mathbb{R}^{1\times 1}, d > 0$

$x|z \sim \mathcal{N}(a + bz, \sigma^2),\ x \in \mathbb{R}^{1\times 1}, a \in \mathbb{R}^{1\times 1}, b \in \mathbb{R}^{1\times 1},\ \sigma \in \mathbb{R}^{1\times 1}, \sigma > 0$

$$\begin{bmatrix} x \\ z \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} a + bc \\ c \end{bmatrix}, \begin{bmatrix} b^2d^2 + \sigma^2 & bd^2 \\ bd^2 & d^2 \end{bmatrix}\right)$$

$x \sim \mathcal{N}(a + bc, b^2d^2 + \sigma^2)$

$z|x \sim \mathcal{N}(c + \frac{bd^2}{b^2d^2+\sigma^2}(x - a - bc), d^2 - \frac{b^2d^4}{b^2d^2+\sigma^2})$

(2.7)

2.2 S2UncFA - Simple-Supervised Unconstrained Factor Analysis

2.2.1 S2UncFA for unidimensional input and unidimensional output

Consider the following **model** where X is the input attribute and Z is the output attribute in a regression problem.

$Z \sim \mathcal{N}(c, d^2),\ Z \in \mathbb{R}, c \in \mathbb{R}, d \in \mathbb{R}, d > 0$

$X|Z \sim \mathcal{N}(a + bz, \sigma^2),\ X \in \mathbb{R}, a \in \mathbb{R}, b \in \mathbb{R}, \sigma \in \mathbb{R}, \sigma > 0$

$\theta = (c, d, a, b, \sigma)$

$(X^{(i)}, Z^{(i)})|\theta \sim (X, Z)|\theta,\ i \in \{1, \ldots, n\}$

$(X^{(1)}, Z^{(1)}), \ldots, (X^{(n)}, Z^{(n)})$ - independent, given the parameters θ

$\mathrm{RV\_D} = ((X^{(1)}, Z^{(1)}), \ldots, (X^{(n)}, Z^{(n)}))$ - random variable for the training data

$\mathrm{D} = ((x^{(1)}, z^{(1)}), \ldots, (x^{(n)}, z^{(n)}))$ - the training data

We can write down the **log-likelihood of the data**:

$$\begin{aligned} l_{\mathrm{RV\_D}}(\theta) &= \ln p_{\mathrm{RV\_D}|\theta}(D|\theta) \\ &= \ln p_{\mathrm{RV\_D}|\theta}((x^{(1)}, z^{(1)}), \ldots, (x^{(n)}, z^{(n)})|\theta) \end{aligned}$$

$$
\begin{aligned}
&\overset{\text{indep.}}{=} \ln\left(\prod_{i=1}^{n} p_{X^{(i)},Z^{(i)}|\theta}(x^{(i)}, z^{(i)}|\theta)\right), \text{ since } E_{(X,Y)}[X+Y] = E_X[X] + E_Y[Y] \\
&= \sum_{i=1}^{n} \ln p_{X^{(i)},Z^{(i)}|\theta}(x^{(i)}, z^{(i)}|\theta) \\
&\overset{\text{mult.rule}}{=} \sum_{i=1}^{n} \ln(p_{Z^{(i)}|\theta}(z^{(i)}|\theta) p_{X^{(i)}|Z^{(i)},\theta}(x^{(i)}|z^{(i)}, \theta)) \\
&= \sum_{i=1}^{n} \Big(\ln(p_{Z^{(i)}|\theta}(z^{(i)}|\theta)) + \ln p_{X^{(i)}|Z^{(i)},\theta}(x^{(i)}|z^{(i)}, \theta)\Big) \\
&= \sum_{i=1}^{n} \left(\ln\left(\frac{1}{\sqrt{2\pi}d} e^{-\frac{1}{2}\left(\frac{z^{(i)}-c}{d}\right)^2}\right) + \ln\left(\frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{1}{2}\left(\frac{x^{(i)}-a-bz^{(i)}}{\sigma}\right)^2}\right)\right) \\
&= \sum_{i=1}^{n} \left(-\ln\sqrt{2\pi} - \ln d - \frac{1}{2}\left(\frac{z^{(i)}-c}{d}\right)^2 - \ln\sqrt{2\pi} - \ln\sigma - \frac{1}{2}\left(\frac{x^{(i)}-a-bz^{(i)}}{\sigma}\right)^2\right) \\
&= -2n\ln\sqrt{2\pi} - n\ln d - n\ln\sigma - \frac{1}{2d^2}\sum_{i=1}^{n}(z^{(i)}-c)^2 - \frac{1}{2\sigma^2}\sum_{i=1}^{n}(x^{(i)}-a-bz^{(i)})^2
\end{aligned}
$$

$$\frac{\partial l_{\text{RV\_D}}}{\partial c} = -\frac{1}{2d^2}\sum_{i=1}^{n} 2(z^{(i)}-c)(-1)$$

$$\frac{\partial l_{\text{RV\_D}}}{\partial c} = 0 \Rightarrow \sum_{i=1}^{n}(z^{(i)}-c) = 0 \Rightarrow \sum_{i=1}^{n} z^{(i)} - nc = 0 \Rightarrow \boxed{\hat{c} = \frac{\sum_{i=1}^{n} z^{(i)}}{n}}$$

$$\frac{\partial l_{\text{RV\_D}}}{\partial d} = -\frac{n}{d} - \frac{1}{2}\sum_{i=1}^{n}(z^{(i)}-c)^2(-2)d^{-3} = -\frac{n}{d} + \frac{\sum_{i=1}^{n}(z^{(i)}-c)^2}{d^3}$$

$$\frac{\partial l_{\text{RV\_D}}}{\partial d} = 0 \Rightarrow \frac{\sum_{i=1}^{n}(z^{(i)}-c)^2}{d^3} = \frac{n}{d} \Rightarrow \boxed{\hat{d}^2 = \frac{\sum_{i=1}^{n}(z^{(i)}-\hat{c})^2}{n}}$$

$$\frac{\partial l_{\text{RV\_D}}}{\partial a} = -\frac{1}{2\sigma^2}\sum_{i=1}^{n} 2(x^{(i)}-a-bz^{(i)})(-1)$$

$$\frac{\partial l_{\text{RV\_D}}}{\partial a} = 0 \Rightarrow \sum_{i=1}^{n}(x^{(i)}-a-bz^{(i)}) = 0 \Rightarrow \sum_{i=1}^{n} x^{(i)} - na - b\sum_{i=1}^{n} z^{(i)} = 0$$

Observation:

$$a = \frac{\sum_{i=1}^{n} x^{(i)} - b\sum_{i=1}^{n} z^{(i)}}{n} = \bar{x} - b\bar{z}, \text{ where } \bar{x} = \frac{\sum_{i=1}^{n} x^{(i)}}{n} \text{ and } \bar{z} = \frac{\sum_{i=1}^{n} z^{(i)}}{n}$$

$$\frac{\partial l_{\text{RV\_D}}}{\partial b} = -\frac{1}{2\sigma^2}\sum_{i=1}^{n} 2(x^{(i)}-a-bz^{(i)})z^{(i)}$$

$$\frac{\partial l_{\text{RV\_D}}}{\partial b} = 0 \Rightarrow \sum_{i=1}^{n}(x^{(i)}-a-bz^{(i)})z^{(i)} = 0 \Rightarrow \sum_{i=1}^{n} x^{(i)}z^{(i)} - a\sum_{i=1}^{n} z^{(i)} - b\sum_{i=1}^{n} {z^{(i)}}^2 = 0$$

We have the following system:

$$\begin{cases} na + \left(\sum_{i=1}^n z^{(i)}\right) b = \sum_{i=1}^n x^{(i)} \\ \left(\sum_{i=1}^n z^{(i)}\right) a + \left(\sum_{i=1}^n z^{(i)2}\right) b = \sum_{i=1}^n x^{(i)} z^{(i)} \end{cases}$$

where a and b are unknown.

We use Cramer's rule to solve it.

$$\Delta = \begin{vmatrix} n & \sum_{i=1}^n z^{(i)} \\ \sum_{i=1}^n z^{(i)} & \sum_{i=1}^n z^{(i)2} \end{vmatrix} = n\sum_{i=1}^n z^{(i)2} - \left(\sum_{i=1}^n z^{(i)}\right)^2$$

$$\Delta_a = \begin{vmatrix} \sum_{i=1}^n x^{(i)} & \sum_{i=1}^n z^{(i)} \\ \sum_{i=1}^n x^{(i)} z^{(i)} & \sum_{i=1}^n z^{(i)2} \end{vmatrix} = \left(\sum_{i=1}^n x^{(i)}\right)\left(\sum_{i=1}^n z^{(i)2}\right) - \left(\sum_{i=1}^n x^{(i)} z^{(i)}\right)\left(\sum_{i=1}^n z^{(i)}\right)$$

$$\Delta_b = \begin{vmatrix} n & \sum_{i=1}^n x^{(i)} \\ \sum_{i=1}^n z^{(i)} & \sum_{i=1}^n x^{(i)} z^{(i)} \end{vmatrix} = n\sum_{i=1}^n x^{(i)} z^{(i)} - \left(\sum_{i=1}^n x^{(i)}\right)\left(\sum_{i=1}^n z^{(i)}\right)$$

$$\boxed{\hat{a} = \frac{\Delta_a}{\Delta} = \frac{\left(\sum_{i=1}^n x^{(i)}\right)\left(\sum_{i=1}^n z^{(i)2}\right) - \left(\sum_{i=1}^n x^{(i)} z^{(i)}\right)\left(\sum_{i=1}^n z^{(i)}\right)}{n\sum_{i=1}^n z^{(i)2} - \left(\sum_{i=1}^n z^{(i)}\right)^2}}$$

$$\boxed{\hat{b} = \frac{\Delta_b}{\Delta} = \frac{n\sum_{i=1}^n x^{(i)} z^{(i)} - \left(\sum_{i=1}^n x^{(i)}\right)\left(\sum_{i=1}^n z^{(i)}\right)}{n\sum_{i=1}^n z^{(i)2} - \left(\sum_{i=1}^n z^{(i)}\right)^2}}$$

$$\frac{\partial l_{\text{RV\_D}}}{\partial \sigma} = -\frac{n}{\sigma} - \frac{1}{2}\sum_{i=1}^n (x^{(i)} - a - bz^{(i)})^2(-2)\sigma^{-3} = -\frac{n}{\sigma} + \frac{\sum_{i=1}^n (x^{(i)} - a - bz^{(i)})^2}{\sigma^3}$$

$$\frac{\partial l_{\text{RV\_D}}}{\partial \sigma} = 0 \Rightarrow \boxed{\hat{\sigma}^2 = \frac{\sum_{i=1}^n (x^{(i)} - \hat{a} - \hat{b}z^{(i)})^2}{n}}$$

Observation: We omitted and will omit in the other proofs computing **second derivatives**.

As we saw, there are closed-form solutions for each parameter in this Maximum Likelihood framework. After learning the parameters, given a new instance x^*, we compute from 2.7 the following:

$$\boxed{Z^*|x* \sim \mathcal{N}\left(\hat{c} + \frac{\hat{b}\hat{d}^2}{\hat{b}^2\hat{d}^2 + \hat{\sigma}^2}(x^* - \hat{a} - \hat{b}\hat{c}), \hat{d}^2 - \frac{\hat{b}^2\hat{d}^4}{\hat{b}^2\hat{d}^2 + \hat{\sigma}^2}\right)}$$

and will **predict** Z^* by the **expected value** (mean) of $Z^*|x^*$. We can also return the **variance** of $Z^*|x^*$ to express a **degree of confidence**.

2.2.2 S2UncFA

Consider the following **model** where X represents the input attributes and Z represents the output attributes in a regression problem. In this setup, **we do no constrain Ψ to be diagonal**.

$Z \sim \mathcal{N}(\mu_z, \Sigma_z)$, $Z \in \mathbb{R}^{d\times 1}$, $\mu_z \in \mathbb{R}^{d\times 1}$, $\Sigma_z \in \mathbb{R}^{d\times d}$

Algorithm 1 S2UncFA for 1D input and 1D output

function $\textsc{Train}(\{(x^{(i)}, z^{(i)}) | i \in \{1, \dots, n\}\})$

  $\hat{c} = \frac{\sum_{i=1}^{n} z^{(i)}}{n}$

  $\hat{d}^2 = \frac{\sum_{i=1}^{n} (z^{(i)} - \hat{c})^2}{n}$

  $\hat{a} = \frac{\left(\sum_{i=1}^{n} x^{(i)}\right)\left(\sum_{i=1}^{n} {z^{(i)}}^2\right) - \left(\sum_{i=1}^{n} x^{(i)} z^{(i)}\right)\left(\sum_{i=1}^{n} z^{(i)}\right)}{n \sum_{i=1}^{n} {z^{(i)}}^2 - \left(\sum_{i=1}^{n} z^{(i)}\right)^2}$

  $\hat{b} = \frac{n \sum_{i=1}^{n} x^{(i)} z^{(i)} - \left(\sum_{i=1}^{n} x^{(i)}\right)\left(\sum_{i=1}^{n} z^{(i)}\right)}{n \sum_{i=1}^{n} {z^{(i)}}^2 - \left(\sum_{i=1}^{n} z^{(i)}\right)^2}$

  $\hat{\sigma}^2 = \frac{\sum_{i=1}^{n} (x^{(i)} - \hat{a} - \hat{b} z^{(i)})^2}{n}$

  return $(\hat{c}, \hat{d}^2, \hat{a}, \hat{b}, \hat{\sigma}^2)$

function $\textsc{Test}(x^*, (\hat{c}, \hat{d}^2, \hat{a}, \hat{b}, \hat{\sigma}^2))$

  value $= \hat{c} + \frac{\hat{b}\hat{d}^2}{\hat{b}^2 \hat{d}^2 + \hat{\sigma}^2}(x^* - \hat{a} - \hat{b}\hat{c})$

  variance $= \hat{d}^2 - \frac{\hat{b}^2 \hat{d}^4}{\hat{b}^2 \hat{d}^2 + \hat{\sigma}^2}$

  return (value, variance)

$X|Z \sim \mathcal{N}(\mu + \Lambda Z, \Psi)$, $X \in \mathbb{R}^{D \times 1}, \mu \in \mathbb{R}^{D \times 1}, \Lambda \in \mathbb{R}^{D \times d}$, $\Psi \in \mathbb{R}^{D \times D}$ - symmetric and positive definite matrix

$\theta = (\mu_z, \Sigma_z, \mu, \Lambda, \Psi)$

$(X^{(i)}, Z^{(i)})|\theta \sim (X, Z)|\theta$, $i \in \{1, \dots, n\}$

$(X^{(1)}, Z^{(1)}), \dots, (X^{(n)}, Z^{(n)})$ - independent, given the parameters θ

RV_D $= ((X^{(1)}, Z^{(1)}), \dots, (X^{(n)}, Z^{(n)}))$ - random variable for the training data

D $= ((x^{(1)}, z^{(1)}), \dots, (x^{(n)}, z^{(n)}))$ - the training data

We can write down the **log-likelihood of the data**:

$$
\begin{aligned}
l_{\text{RV\_D}}(\theta) &= \ln p_{\text{RV\_D}|\theta}(D|\theta) \\
&= \ln p_{\text{RV\_D}|\theta}((x^{(1)}, z^{(1)}), \dots, (x^{(n)}, z^{(n)})|\theta) \\
&\overset{\text{indep.}}{=} \ln \left(\prod_{i=1}^{n} p_{X^{(i)}, Z^{(i)}|\theta}(x^{(i)}, z^{(i)}|\theta) \right), \text{ since } E_{(X,Y)}[X+Y] = E_X[X] + E_Y[Y] \\
&= \sum_{i=1}^{n} \ln p_{X^{(i)}, Z^{(i)}|\theta}(x^{(i)}, z^{(i)}|\theta) \\
&\overset{\text{mult.rule}}{=} \sum_{i=1}^{n} \ln (p_{Z^{(i)}|\theta}(z^{(i)}|\theta) p_{X^{(i)}|Z^{(i)},\theta}(x^{(i)}|z^{(i)}, \theta)) \\
&= \sum_{i=1}^{n} \Big(\ln(p_{Z^{(i)}|\theta}(z^{(i)}|\theta)) + \ln(p_{X^{(i)}|Z^{(i)},\theta}(x^{(i)}|z^{(i)}, \theta)) \Big) \\
&= \sum_{i=1}^{n} \ln \left(\frac{1}{\sqrt{(2\pi)^d \det(\Sigma_z)}} e^{-\frac{1}{2}(z^{(i)} - \mu_z)^\top \Sigma_z^{-1} (z^{(i)} - \mu_z)} \right) + \\
&\qquad + \sum_{i=1}^{n} \ln \left(\frac{1}{\sqrt{(2\pi)^D \det(\Psi)}} e^{-\frac{1}{2}(x^{(i)} - \mu - \Lambda z^{(i)})^\top \Psi^{-1} (x^{(i)} - \mu - \Lambda z^{(i)})} \right) \\
&= \sum_{i=1}^{n} \left(-\frac{d}{2} \ln(2\pi) - \frac{1}{2} \ln(\det(\Sigma_z)) - \frac{1}{2}(z^{(i)} - \mu_z)^\top \Sigma_z^{-1} (z^{(i)} - \mu_z) \right) + \\
&\qquad + \sum_{i=1}^{n} \left(-\frac{D}{2} \ln(2\pi) - \frac{1}{2} \ln(\det(\Psi)) - \frac{1}{2}(x^{(i)} - \mu - \Lambda z^{(i)})^\top \Psi^{-1} (x^{(i)} - \mu - \Lambda z^{(i)}) \right)
\end{aligned}
$$

$$= -\frac{nd}{2}\ln(2\pi) - \frac{n}{2}\ln(\det(\Sigma_z)) - \frac{1}{2}\sum_{i=1}^{n}(z^{(i)}-\mu_z)^\top \Sigma_z^{-1}(z^{(i)}-\mu_z) -$$
$$-\frac{nD}{2}\ln(2\pi) - \frac{n}{2}\ln(\det(\Psi)) - \frac{1}{2}\sum_{i=1}^{n}(x^{(i)}-\mu-\Lambda z^{(i)})^\top \Psi^{-1}(x^{(i)}-\mu-\Lambda z^{(i)})$$

Observation: We will use the known fact that if a matrix A is symmetric, then A^{-1} is also symmetric.

$$\frac{\partial l_{\text{RV\_D}}}{\partial \mu_z} \overset{[11,(84)]}{=} -\frac{1}{2}\sum_{i=1}^{n}(-2(-I)^\top \Sigma_z^{-1}(z^{(i)}-\mu_z)) = -\sum_{i=1}^{n}\Sigma_z^{-1}(z^{(i)}-\mu_z)$$

$$\frac{\partial l_{\text{RV\_D}}}{\partial \mu_z} = 0 \overset{\Sigma_z\cdot|}{\Rightarrow} \sum_{i=1}^{n}(z^{(i)}-\mu_z) = 0 \Rightarrow \sum_{i=1}^{n} z^{(i)} - n\mu_z = 0 \Rightarrow \boxed{\hat{\mu}_z = \frac{\sum_{i=1}^{n} z^{(i)}}{n}}$$

$$\frac{\partial l_{\text{RV\_D}}}{\partial \Sigma_z} \overset{[11,(61)][12,(4b)]}{=} -\frac{n}{2}\Sigma_z^{-\top} - \frac{1}{2}\sum_{i=1}^{n}(-\Sigma_z^{-\top}(z^{(i)}-\mu_z)(z^{(i)}-\mu_z)^\top \Sigma_z^{-\top}) =$$
$$= -\frac{n}{2}\Sigma_z^{-1} + \frac{1}{2}\sum_{i=1}^{n}\Sigma_z^{-1}(z^{(i)}-\mu_z)(z^{(i)}-\mu_z)^\top \Sigma_z^{-1}$$

$$\frac{\partial l_{\text{RV\_D}}}{\partial \Sigma_z} = 0 \overset{\Sigma_z\cdot|\ldots|\cdot\Sigma_z}{\Rightarrow} -\frac{n}{2}\Sigma_z + \frac{1}{2}\sum_{i=1}^{n}(z^{(i)}-\mu_z)(z^{(i)}-\mu_z)^\top = 0 \Rightarrow$$

$\Rightarrow \boxed{\hat{\Sigma}_z = \frac{\sum_{i=1}^{n}(z^{(i)}-\hat{\mu}_z)(z^{(i)}-\hat{\mu}_z)^\top}{n}}$ which is symmetric and positive definite if rank($\begin{bmatrix} z^{(1)}-\hat{\mu}_z & \ldots & z^{(n)}-\hat{\mu}_z \end{bmatrix}$) = d (which usually holds in practice if $n \geq d$),

i.e. the **constraint is satisfied**

$$\frac{\partial l_{\text{RV\_D}}}{\partial \mu} \overset{[11,(84)]}{=} -\frac{1}{2}\sum_{i=1}^{n}(-2(-I)^\top \Psi^{-1}(x^{(i)}-\mu-\Lambda z^{(i)})) = -\sum_{i=1}^{n}\Psi^{-1}(x^{(i)}-\mu-\Lambda z^{(i)})$$

$$\frac{\partial l_{\text{RV\_D}}}{\partial \mu} = 0 \Rightarrow \sum_{i=1}^{n}(x^{(i)}-\mu-\Lambda z^{(i)}) = 0 \Rightarrow \mu = \frac{\sum_{i=1}^{n} x^{(i)} - \Lambda\sum_{i=1}^{n} z^{(i)}}{n} = \bar{x} - \Lambda\bar{z}$$

where $\bar{x} = \frac{\sum_{i=1}^{n} x^{(i)}}{n}$ and $\bar{z} = \frac{\sum_{i=1}^{n} z^{(i)}}{n}$

$$\frac{\partial l_{\text{RV\_D}}}{\partial \Lambda} \overset{[11,(88)]}{=} -\frac{1}{2}\sum_{i=1}^{n}(-2\Psi^{-1}(x^{(i)}-\mu-\Lambda z^{(i)}){z^{(i)}}^\top) = -\sum_{i=1}^{n}\Psi^{-1}(x^{(i)}-\mu-\Lambda z^{(i)}){z^{(i)}}^\top$$

$$\frac{\partial l_{\text{RV\_D}}}{\partial \Lambda} = 0 \overset{\Psi\cdot|}{\Rightarrow} \sum_{i=1}^{n}(x^{(i)}-\mu-\Lambda z^{(i)}){z^{(i)}}^\top = 0 \overset{\mu=\bar{x}-\Lambda\bar{z}}{\Rightarrow} \sum_{i=1}^{n}(x^{(i)}-\bar{x}+\Lambda\bar{z}-\Lambda z^{(i)}){z^{(i)}}^\top = 0 \Rightarrow$$
$$\Rightarrow \sum_{i=1}^{n} x^{(i)}{z^{(i)}}^\top - \bar{x}\sum_{i=1}^{n}{z^{(i)}}^\top + \sum_{i=1}^{n}\Lambda\bar{z}{z^{(i)}}^\top - \Lambda\sum_{i=1}^{n} z^{(i)}{z^{(i)}}^\top = 0 \Rightarrow$$

$$\Rightarrow \sum_{i=1}^{n} x^{(i)} {z^{(i)}}^\top - n\bar{x}\bar{z}^\top + n\Lambda\bar{z}\bar{z}^\top - \Lambda\sum_{i=1}^{n} z^{(i)} {z^{(i)}}^\top = 0 \Rightarrow$$

$$\Rightarrow \Lambda\left(n\bar{z}\bar{z}^\top - \sum_{i=1}^{n} z^{(i)} {z^{(i)}}^\top\right) = n\bar{x}\bar{z}^\top - \sum_{i=1}^{n} x^{(i)} {z^{(i)}}^\top \Rightarrow$$

$$\Rightarrow \boxed{\hat{\Lambda} = \left(n\bar{x}\bar{z}^\top - \sum_{i=1}^{n} x^{(i)} {z^{(i)}}^\top\right)\left(n\bar{z}\bar{z}^\top - \sum_{i=1}^{n} z^{(i)} {z^{(i)}}^\top\right)^{-1}}$$

$$\boxed{\hat{\mu} = \bar{x} - \hat{\Lambda}\bar{z}}$$

$$\frac{\partial l_{\text{RV\_D}}}{\partial \Psi} \overset{[11,(61)][12,(4b)]}{=} -\frac{n}{2}\Psi^{-\top} - \frac{1}{2}\sum_{i=1}^{n}(-\Psi^{-\top}(x^{(i)} - \mu - \Lambda z^{(i)})(x^{(i)} - \mu - \Lambda z^{(i)})^\top \Psi^{-\top}) =$$

$$= -\frac{n}{2}\Psi^{-1} + \frac{1}{2}\sum_{i=1}^{n}\Psi^{-1}(x^{(i)} - \mu - \Lambda z^{(i)})(x^{(i)} - \mu - \Lambda z^{(i)})^\top \Psi^{-1}$$

$$\frac{\partial l_{\text{RV\_D}}}{\partial \Psi} = 0 \overset{\Psi\cdot|\ldots|\cdot\Psi}{\Rightarrow} -\frac{n}{2}\Psi + \frac{1}{2}\sum_{i=1}^{n}(x^{(i)} - \mu - \Lambda z^{(i)})(x^{(i)} - \mu - \Lambda z^{(i)})^\top = 0 \Rightarrow$$

$\Rightarrow \boxed{\hat{\Psi} = \frac{\sum_{i=1}^{n}(x^{(i)} - \mu - \Lambda z^{(i)})(x^{(i)} - \mu - \Lambda z^{(i)})^\top}{n}}$ which is symmetric and positive definite if rank$\left(\begin{bmatrix} x^{(1)} - \hat{\mu} - \hat{\Lambda}z^{(1)} & \ldots & x^{(n)} - \hat{\mu} - \hat{\Lambda}z^{(n)} \end{bmatrix}\right)$ = D (which usually holds in practice if $n \geq D$), i.e. the **constraint is satisfied**

As we saw, there are closed-form solutions for each parameter in this Maximum Likelihood framework.

After learning the parameters, given a new instance x^*, we compute from 2.6 the following:
$Z^*|x^* \sim \mathcal{N}(\hat{\mu}_z + \hat{\Sigma}_z\hat{\Lambda}^\top(\hat{\Lambda}\hat{\Sigma}_z\hat{\Lambda}^\top + \hat{\Psi})^{-1}(x^* - \hat{\mu} - \hat{\Lambda}\hat{\mu}_z), \hat{\Sigma}_z - \hat{\Sigma}_z\hat{\Lambda}^\top(\hat{\Lambda}\hat{\Sigma}_z\hat{\Lambda}^\top + \hat{\Psi})^{-1}\hat{\Lambda}\hat{\Sigma}_z^\top)$
and will **predict** Z^* by the **expected value** (mean) of $Z^*|x^*$. We can also return the **covariance matrix** of $Z^*|x^*$ to express a **degree of confidence**.

Algorithm 2 S2UncFA

```
function TRAIN({(x^(i), z^(i)) | i ∈ {1, ..., n}})
    μ̂_z = (Σ_{i=1}^n z^(i)) / n
    Σ̂_z = (Σ_{i=1}^n (z^(i) − μ̂_z)(z^(i) − μ̂_z)^T) / n
    Λ̂ = (n x̄ z̄^T − Σ_{i=1}^n x^(i) z^(i)^T)(n z̄ z̄^T − Σ_{i=1}^n z^(i) z^(i)^T)^(−1)
    μ̂ = x̄ − Λ̂ z̄
    Ψ̂ = (Σ_{i=1}^n (x^(i) − μ − Λz^(i))(x^(i) − μ − Λz^(i))^T) / n
    return (μ̂_z, Σ̂_z, Λ̂, μ̂, Ψ̂)
function TEST(x*, (μ̂_z, Σ̂_z, Λ̂, μ̂, Ψ̂))
    value = μ̂_z + Σ̂_z Λ̂^T (Λ̂ Σ̂_z Λ̂^T + Ψ̂)^(−1) (x* − μ̂ − Λ̂ μ̂_z)
    covarianceMatrix = Σ̂_z − Σ̂_z Λ̂^T (Λ̂ Σ̂_z Λ̂^T + Ψ̂)^(−1) Λ̂ Σ̂_z^T
    return (value, covarianceMatrix)
```

Because of 2.1 and 2.2, one can notice that some formulas in the train phase can be vectorised (written in **matrix form**). One possibility is as follows:

$$\hat{\mu}_z = \frac{\sum_{i=1}^{n} z^{(i)}}{n}$$

$$\hat{\Sigma}_z = \frac{1}{n}\left(Z_m - \hat{\mu}_z \underbrace{\begin{bmatrix}1 & 1 & \dots & 1\end{bmatrix}}_{\in\mathbb{R}^{1\times n}}\right)\left(Z_m - \hat{\mu}_z \begin{bmatrix}1 & 1 & \dots & 1\end{bmatrix}\right)^{\top}$$

$$\hat{\Lambda} = (n\bar{x}\bar{z}^{\top} - X_m Z_m^{\top})(n\bar{z}\bar{z}^{\top} - Z_m Z_m^{\top})^{-1}$$

$$\hat{\mu} = \bar{x} - \hat{\Lambda}\bar{z}$$

$$\hat{\Psi} = \frac{1}{n}\left(X_m - \hat{\mu} \underbrace{\begin{bmatrix}1 & 1 & \dots & 1\end{bmatrix}}_{\in\mathbb{R}^{1\times n}} - \hat{\Lambda} Z_m\right)\left(X_m - \hat{\mu} \begin{bmatrix}1 & 1 & \dots & 1\end{bmatrix} - \hat{\Lambda} Z_m\right)^{\top}$$

where $Z_m = \begin{bmatrix}z^{(1)} & \dots & z^{(n)}\end{bmatrix}$ and $X_m = \begin{bmatrix}x^{(1)} & \dots & x^{(n)}\end{bmatrix}$

Observation: Another formula for $\hat{\Sigma}_z$ is the following:

$$\hat{\Sigma}_z = \frac{\sum_{i=1}^{n}(z^{(i)} - \hat{\mu}_z)(z^{(i)} - \hat{\mu}_z)^{\top}}{n} = \frac{\sum_{i=1}^{n}(z^{(i)}z^{(i)\top} - z^{(i)}\hat{\mu}_z^{\top} - \hat{\mu}_z z^{(i)\top} + \hat{\mu}_z\hat{\mu}_z^{\top})}{n} =$$

$$= \frac{\sum_{i=1}^{n} z^{(i)}z^{(i)} - n\hat{\mu}_z\hat{\mu}_z^{\top} - \cancel{n\hat{\mu}_z\hat{\mu}_z^{\top}} + \cancel{n\hat{\mu}_z\hat{\mu}_z^{\top}}}{n} = \frac{\sum_{i=1}^{n} z^{(i)}z^{(i)\top}}{n} - \hat{\mu}_z\hat{\mu}_z^{\top} =$$

$$= \frac{1}{n} Z_m Z_m^{\top} - \hat{\mu}_z\hat{\mu}_z^{\top}$$

One can also notice that a relatively **simpler derivation** for $\hat{\mu}$ and $\hat{\Lambda}$ could have been possible if we had denoted from the very beginning the following:

$T = \begin{bmatrix}Z \\ 1\end{bmatrix}, T^{(i)} = \begin{bmatrix}Z^{(i)} \\ 1\end{bmatrix}, t^{(i)} = \begin{bmatrix}z^{(i)} \\ 1\end{bmatrix}$

$\beta = \begin{bmatrix}\Lambda & \mu\end{bmatrix}$ and in this manner:

$Z \sim \mathcal{N}(\mu_z, \Sigma_z)$

$X|Z \sim \mathcal{N}(\beta T, \Psi)$

In the end, the log-likelihood would have been:

$$l_{\text{RV\_D}}(\theta) = -\frac{nd}{2}\ln(2\pi) - \frac{n}{2}\ln(\det(\Sigma_z)) - \frac{1}{2}\sum_{i=1}^{n}(z^{(i)} - \mu_z)^{\top}\Sigma_z^{-1}(z^{(i)} - \mu_z) -$$
$$- \frac{nD}{2}\ln(2\pi) - \frac{n}{2}\ln(\det(\Psi)) - \frac{1}{2}\sum_{i=1}^{n}(x^{(i)} - \beta t^{(i)})^{\top}\Psi^{-1}(x^{(i)} - \beta t^{(i)})$$

In general, the derivatives would have been the same, except for β:

$$\frac{\partial l_{\text{RV\_D}}}{\partial \beta} \overset{[11,(88)]}{=} -\frac{1}{2}\sum_{i=1}^{n}(-2\Psi^{-1}(x^{(i)} - \beta t^{(i)})t^{(i)\top}) = \sum_{i=1}^{n}\Psi^{-1}(x^{(i)} - \beta t^{(i)})t^{(i)\top}$$

$$\frac{\partial l_{\text{RV\_D}}}{\partial \beta} = 0 \overset{\Psi\cdot|}{\Rightarrow} \sum_{i=1}^{n}(x^{(i)} - \beta t^{(i)})t^{(i)\top} \Rightarrow \sum_{i=1}^{n}(x^{(i)}t^{(i)\top} - \beta t^{(i)}t^{(i)\top}) = 0 \Rightarrow$$

$$\Rightarrow \sum_{i=1}^{n} x^{(i)}t^{(i)\top} - \beta\sum_{i=1}^{n} t^{(i)}t^{(i)\top} = 0 \Rightarrow \boxed{\hat{\beta} = \left(\sum_{i=1}^{n} x^{(i)}t^{(i)\top}\right)\left(\sum_{i=1}^{n} t^{(i)}t^{(i)\top}\right)^{-1}}$$

In this way, $\hat{\beta}$ can be computed on the spot (μ, Λ have been computed by solving a system of equations with the substitution method).

There is also a matrix form for $\hat{\beta}$:

$\hat{\beta} = X_m T_m^\top (T_m T_m^\top)^{-1}$, where $X_m = \begin{bmatrix} x^{(1)} & \dots & x^{(n)} \end{bmatrix}$ and $T_m = \begin{bmatrix} t^{(1)} & \dots & t^{(n)} \end{bmatrix}$.

So, the training phase in matrix form would become:

$$\hat{\mu}_z = \frac{\sum_{i=1}^n z^{(i)}}{n}$$

$$\hat{\Sigma}_z = \frac{1}{n} \left(Z_m - \mu_z \underbrace{\begin{bmatrix} 1 & 1 & \dots & 1 \end{bmatrix}}_{\in \mathbb{R}^{1\times n}} \right) \left(Z_m - \mu_z \begin{bmatrix} 1 & 1 & \dots & 1 \end{bmatrix} \right)^\top$$

$$\hat{\beta} = X_m T_m^\top (T_m T_m^\top)^{-1}$$

$$\hat{\Psi} = \frac{1}{n} (X_m - \hat{\beta} T_m)(X_m - \hat{\beta} T_m)^\top$$

where $Z_m = \begin{bmatrix} z^{(1)} & \dots & z^{(n)} \end{bmatrix}$, $T_m = \begin{bmatrix} t^{(1)} & \dots & t^{(n)} \end{bmatrix}$ and $X_m = \begin{bmatrix} x^{(1)} & \dots & x^{(n)} \end{bmatrix}$.

In order to retrieve $\hat{\mu}$ and $\hat{\Lambda}$, we can apply the following:

$$\hat{\Lambda} = \hat{\beta} \underbrace{\begin{bmatrix} I_d \\ 0 \end{bmatrix}}_{\in \mathbb{R}^{(d+1)\times d}}$$

$$\hat{\mu} = \hat{\beta} \underbrace{\begin{bmatrix} 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix}}_{\in \mathbb{R}^{(d+1)\times 1}}$$

or just select the first d columns of $\hat{\beta}$ for $\hat{\Lambda}$ and the last one for $\hat{\mu}$.

2.2.3 Weak equivalence to Linear Regression

There are a few reasons why *Linear Regression* (*LR*) comes to mind in our context:

- one with experience in *LR* would have recognized a similarity in the model definition when $x \in \mathbb{R}^{1\times 1}$:

  S2UncFA:

  $z \sim \mathcal{N}(\mu_z, \Sigma_z)$

  $x|z \sim \mathcal{N}(a^\top z + b, \sigma^2)$

  versus

  LR:

  $$y|x \sim \mathcal{N}(\underbrace{a^\top x + b}_{\text{or } \beta \begin{bmatrix} x \\ 1 \end{bmatrix}}, \sigma^2)$$

  As a difference, the input in our model is the output in *LR* and our output is *LR*'s input.

- it is known that in the case of *LR* when the input is uni-dimensional, we can use the following formulas [13]:

$$\hat{a} = \frac{n(\sum_{i=1}^{n} x_i y_i) - (\sum_{i=1}^{n} x_i)(\sum_{i=1}^{n} y_i)}{n(\sum_{i=1}^{n} x_i^2) - (\sum_{i=1}^{n} x_i)^2}$$

$$\hat{b} = \frac{(\sum_{i=1}^{n} x_i^2)(\sum_{i=1}^{n} y_i) - (\sum_{i=1}^{n} x_i)(\sum_{i=1}^{n} x_i y_i)}{n(\sum_{i=1}^{n} x_i^2) - (\sum_{i=1}^{n} x_i)^2}$$

  In *S2UncFA* for 1-dimensional input and output we obtained the same formulas, except for the fact that the input and output are swapped.

- it is known that *LR* has a closed-form solution in matrix form (normal equations) [13]:

$$y|x \sim \mathcal{N}(\beta \begin{bmatrix} x \\ 1 \end{bmatrix}, \sigma^2)$$

$$\hat{\beta}^\top = (XX^\top)^{-1} X y$$

  where $x = \begin{bmatrix} x^{(1)} & x^{(2)} & \dots & x^{(n)} \end{bmatrix}$, $x^{(i)} \in \mathbb{R}^{D \times 1}$, $y \in \mathbb{R}^{n \times 1}$

  It results immediately that $\hat{\beta} = y^\top X^\top (XX^\top)^{-1}$ which is exactly our formula for $\hat{\beta}$ in *S2UncFA*, where $X := T_m$ and $y := X_m^\top$, in the case when the input in *S2UncFA* is unidimensional.

As a conclusion, when the **input in *S2UncFA* is 1-dimensional**, the training phase in *S2UncFA* when computing $\hat{\beta}$ (i.e. $\hat{\mu}$, $\hat{\Lambda}$) is **equivalent to fitting a *LR*** model via normal equations with the **input and output swapped**.

But what happens when the input in *S2UncFA* is multi-dimensional?

As noted in [14], there is a natural extension for *LR* to **multi-output regression**. The conclusion there is that the task is equivalent to D independent single-output *LR* tasks (if the output has D dimensions). This result can be noticed if we look at the objective function which has to be minimized:

$J(\beta) = \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{D} ((\beta t^{(i)})_j - y_j^{(i)})^2$.

The term $\beta t_j^{(i)}$ uses only the j$^{\text{th}}$ row of β: $(\beta t^{(i)})_j = \beta_{j:} t^{(i)}$ so $J(\beta)$ can be divided into D independent sums, i.e. which do not share parameters:

$J(\beta) = \frac{1}{2} \sum_{i=1}^{n} (\beta_{1:} t^{(i)} - y_1^{(i)})^2 + \dots + \frac{1}{2} \sum_{i=1}^{n} (\beta_{D:} t^{(i)} - y_D^{(i)})^2$

As a result, $J(\beta)$ can be optimized by optimizing each of those D sums. Furthermore, it can be easily noticed that a sum corresponds to the objective function of a single-output *LR* model.

The closed-form solution is very similar to the one for single-output *LR*:

$\beta^\top = (XX^\top)^{-1} XY$,

where $X = \begin{bmatrix} x^{(1)} & x^{(2)} & \dots & x^{(n)} \end{bmatrix}$, $x^{(i)} \in \mathbb{R}^{d \times 1}, i \in \{1, \dots, n\}$, $\mathrm{Y} = \begin{bmatrix} y^{(1)^\top} \\ y^{(2)^\top} \\ \vdots \\ y^{(n)^\top} \end{bmatrix}$, $y^{(i)} \in \mathbb{R}^{D \times 1}$

It results immediately that: $\beta = Y^\top X^\top (XX^\top)^{-1}$ which **is exactly our formula for $\hat{\beta}$ in *S2UncFA*** irrespective to the number of input dimensions (in *S2UncFA*), where $X := T_m$, $Y := X_m^\top$.

2.2.4 Strong equivalence to Linear Regression

We have seen that there is a weak equivalence between *S2UncFA* and *LR*. There is also another equivalence between those two which we call ***strong equivalence***. We discuss it below.

For a new instance x^*, we predict:

$\hat{\mu}_z + \hat{\Sigma}_z\hat{\Lambda}^\top(\hat{\Lambda}\hat{\Sigma}_z\hat{\Lambda}^\top + \hat{\Psi})^{-1}(x^* - \hat{\mu} - \hat{\Lambda}\hat{\mu}_z)$

We will show that this hyperplane (when x^* varies) is the same as the one learnt by a (multi-output) *LR* model with the input X_m and the output Z_m.

For the moment, we omit the index $_m$ in X_m, Z_m for brevity.

$$\begin{aligned}
\hat{\Psi} &= \frac{1}{n}\sum_{i=1}^{n}(x^{(i)} - \hat{\mu} - \hat{\Lambda}z^{(i)})(x^{(i)} - \hat{\mu} - \hat{\Lambda}z^{(i)})^\top \\
&= \frac{1}{n}\sum_{i=1}^{n}(x^{(i)}x^{(i)\top} - x^{(i)}\hat{\mu}^\top - x^{(i)}z^{(i)\top}\hat{\Lambda}^\top - \hat{\mu}x^{(i)\top} + \hat{\mu}\hat{\mu}^\top + \hat{\mu}z^{(i)\top}\hat{\Lambda}^\top - \\
&\qquad - \hat{\Lambda}z^{(i)}x^{(i)\top} + \hat{\Lambda}z^{(i)}\hat{\mu}^\top + \hat{\Lambda}z^{(i)}z^{(i)\top}\hat{\Lambda}^\top) \\
&= \frac{1}{n}XX^\top - \bar{x}\hat{\mu}^\top - \frac{1}{n}XZ^\top\hat{\Lambda}^\top - \hat{\mu}\bar{x}^\top + \hat{\mu}\hat{\mu}^\top + \hat{\mu}\bar{z}^\top\hat{\Lambda}^\top - \frac{1}{n}\hat{\Lambda}ZX^\top + \hat{\Lambda}\hat{z}\hat{\mu}^\top + \frac{1}{n}\hat{\Lambda}ZZ^\top\hat{\Lambda}^\top \\
&= \frac{1}{n}XX^\top - \frac{1}{n}XZ^\top\hat{\Lambda}^\top - \frac{1}{n}\hat{\Lambda}ZX^\top + \frac{1}{n}\hat{\Lambda}ZZ^\top\hat{\Lambda}^\top - \bar{x}\hat{\mu}^\top - \hat{\mu}\bar{x}^\top + \hat{\mu}\hat{\mu}^\top + \hat{\Lambda}\hat{z}\hat{\mu}^\top + \hat{\mu}\bar{z}^\top\hat{\Lambda}^\top
\end{aligned}$$

We substitute $\hat{\mu}$ with $\bar{x} - \hat{\Lambda}\bar{z}$.

$\bar{x}\hat{\mu}^\top = \bar{x}(\bar{x} - \hat{\Lambda}\bar{z})^\top = \bar{x}\bar{x}^\top - \bar{x}\bar{z}^\top\hat{\Lambda}^\top$

$\hat{\mu}\bar{x}^\top = (\bar{x} - \hat{\Lambda}\bar{z})\bar{x}^\top = \bar{x}\bar{x}^\top - \hat{\Lambda}\bar{z}\bar{x}^\top$

$\hat{\mu}\hat{\mu}^\top = (\bar{x} - \hat{\Lambda}\bar{z})(\bar{x} - \hat{\Lambda}\bar{z})^\top = \bar{x}\bar{x}^\top - \bar{x}\bar{z}^\top\hat{\Lambda}^\top - \hat{\Lambda}\bar{z}\bar{x}^\top + \hat{\Lambda}\bar{z}\bar{z}^\top\hat{\Lambda}^\top$

$\hat{\Lambda}\bar{z}\hat{\mu}^\top = \hat{\Lambda}\bar{z}(\bar{x} - \hat{\Lambda}\bar{z})^\top = \hat{\Lambda}\bar{z}\bar{x}^\top - \hat{\Lambda}\bar{z}\bar{z}^\top\hat{\Lambda}^\top$

$\hat{\mu}\bar{z}^\top\hat{\Lambda}^\top = (\bar{x} - \hat{\Lambda}\bar{z})\bar{z}^\top\hat{\Lambda}^\top = \bar{x}\bar{z}^\top\hat{\Lambda}^\top - \hat{\Lambda}\bar{z}\bar{z}^\top\hat{\Lambda}^\top$

We return to compute $\hat{\Psi}$:

$$\begin{aligned}
\hat{\Psi} &= \frac{1}{n}XX^\top - \frac{1}{n}XZ^\top\hat{\Lambda}^\top - \frac{1}{n}\hat{\Lambda}ZX^\top + \frac{1}{n}\hat{\Lambda}ZZ^\top\hat{\Lambda}^\top - \bar{x}\bar{x}^\top + \cancel{\bar{x}\bar{z}^\top\hat{\Lambda}^\top} - \cancel{\bar{x}\bar{x}^\top} + \cancel{\hat{\Lambda}\bar{z}\bar{z}^\top} + \cancel{\bar{x}\bar{x}^\top} - \\
&\qquad - \cancel{\bar{x}\bar{z}^\top\hat{\Lambda}^\top} - \cancel{\hat{\Lambda}\bar{z}\bar{z}^\top} + \cancel{\hat{\Lambda}\bar{z}\bar{z}^\top\hat{\Lambda}^\top} + \hat{\Lambda}\bar{z}\bar{x}^\top - \cancel{\hat{\Lambda}\bar{z}\bar{z}^\top\hat{\Lambda}^\top} + \bar{x}\bar{z}^\top\hat{\Lambda}^\top - \hat{\Lambda}\bar{z}\bar{z}^\top\hat{\Lambda}^\top \\
&= \frac{1}{n}XX^\top - \frac{1}{n}XZ^\top\hat{\Lambda}^\top - \frac{1}{n}\hat{\Lambda}ZX^\top + \frac{1}{n}\hat{\Lambda}ZZ^\top\hat{\Lambda}^\top - \bar{x}\bar{x}^\top + \hat{\Lambda}\bar{z}\bar{x}^\top + \bar{x}\bar{z}^\top\hat{\Lambda}^\top - \hat{\Lambda}\bar{z}\bar{z}^\top\hat{\Lambda}^\top
\end{aligned}$$

$$\hat{\Lambda}\hat{\Sigma}_z\hat{\Lambda}^\top = \hat{\Lambda}\left(\frac{1}{n}ZZ^\top - \bar{z}\bar{z}^\top\right)\hat{\Lambda}^\top = \frac{1}{n}\hat{\Lambda}ZZ^\top\hat{\Lambda}^\top - \hat{\Lambda}\bar{z}\bar{z}^\top\hat{\Lambda}^\top$$

We observe that the above term is also included in $\hat{\Psi}$.

$$\begin{aligned}
\hat{\Sigma}_z\hat{\Lambda}^\top &= \left(\frac{1}{n}ZZ^\top - \bar{z}\bar{z}^\top\right)(n\bar{z}\bar{z}^\top - ZZ^\top)^{-1}(n\bar{z}\bar{x}^\top - ZX^\top) \\
&= \left(\frac{1}{n}ZZ^\top - \bar{z}\bar{z}^\top\right)\left(-\frac{1}{n}\right)\left(\frac{1}{n}ZZ^\top - \bar{z}\bar{z}^\top\right)^{-1}(n\bar{z}\bar{x}^\top - ZX^\top) \\
&= \frac{1}{n}ZX^\top - \bar{z}\bar{x}^\top
\end{aligned}$$

$(\hat{\Lambda}\hat{\Sigma}_z\hat{\Lambda}^\top)^\top = \hat{\Lambda}\hat{\Sigma}_z^\top\hat{\Lambda}^\top = \hat{\Lambda}\hat{\Sigma}_z\hat{\Lambda}^\top \Rightarrow \hat{\Lambda}\hat{\Sigma}_z\hat{\Lambda}^\top$ - symmetric.

We have that:

$$\hat{\Lambda}\hat{\Sigma}_z\hat{\Lambda}^\top = \hat{\Lambda}(\hat{\Sigma}_z\hat{\Lambda}^\top) = \hat{\Lambda}\left(\frac{1}{n}ZX^\top - \bar{z}\bar{x}^\top\right) = \frac{1}{n}\hat{\Lambda}ZX^\top - \hat{\Lambda}\bar{z}\bar{x}^\top \tag{2.8}$$

We also have that:

$$\hat{\Lambda}\hat{\Sigma}_z\hat{\Lambda}^\top = (\hat{\Lambda}\hat{\Sigma}_z\hat{\Lambda}^\top)^\top = \left(\frac{1}{n}\hat{\Lambda}ZX^\top - \hat{\Lambda}\bar{z}\bar{x}^\top\right)^\top = \frac{1}{n}XZ^\top\hat{\Lambda}^\top - \bar{x}\bar{z}^\top\hat{\Lambda}^\top \tag{2.9}$$

We observed that $\hat{\Lambda}\hat{\Sigma}_z\hat{\Lambda}^\top$ is included twice in $\hat{\Lambda}\hat{\Sigma}_z\hat{\Lambda}^\top + \hat{\Psi}$. We replaced it once with 2.8 and then with 2.9.

We get:

$$\begin{aligned}
\hat{\Lambda}\hat{\Sigma}_z\hat{\Lambda}^\top + \hat{\Psi} &= \frac{1}{n}XX^\top - \cancel{\frac{1}{n}XZ^\top\hat{\Lambda}^\top} - \cancel{\frac{1}{n}\hat{\Lambda}ZX^\top} - \bar{x}\bar{x}^\top + \cancel{\hat{\Lambda}\bar{z}\bar{x}^\top} + \\
&\quad + \cancel{\bar{x}\bar{z}^\top\hat{\Lambda}^\top} + \cancel{\frac{1}{n}\hat{\Lambda}ZX^\top} - \cancel{\hat{\Lambda}\bar{z}\bar{x}^\top} + \cancel{\frac{1}{n}XZ^\top\hat{\Lambda}^\top} - \cancel{\bar{x}\bar{z}^\top\hat{\Lambda}^\top} \\
&= \frac{1}{n}XX^\top - \bar{x}\bar{x}^\top
\end{aligned}$$

Observation: The result is exactly the sample covariance matrix for the training set. This is natural because $x \sim \mathcal{N}(\mu, \Lambda\Sigma_z\Lambda^\top + \Psi)$ and there are enough free parameters for $\Lambda\Sigma_z\Lambda^\top + \Psi$ to become $\frac{1}{n}XX^\top - \bar{x}\bar{x}^\top$ (which is the Maximum Likelihood estimator for Σ, where $x \sim \mathcal{N}(\mu, \Sigma)$).

We return to the initial computation:

$$\begin{aligned}
&\hat{\mu}_z + \hat{\Sigma}_z\hat{\Lambda}^\top(\hat{\Lambda}\hat{\Sigma}_z\hat{\Lambda}^\top + \hat{\Psi})^{-1}(x^* - \hat{\mu} - \hat{\Lambda}\hat{\mu}_z) = \\
&\qquad = \bar{z} + \left(\frac{1}{n}Z\bar{x} - \bar{z}\bar{x}^\top\right)\left(\frac{1}{n}XX^\top - \bar{x}\bar{x}^\top\right)^{-1}(x^* - \bar{x} + \cancel{\hat{\Lambda}\bar{z}} - \cancel{\hat{\Lambda}\bar{z}}) \\
&\qquad = \underbrace{\left(\frac{1}{n}Z\bar{x} - \bar{z}\bar{x}^\top\right)\left(\frac{1}{n}XX^\top - \bar{x}\bar{x}^\top\right)^{-1}}_{\text{the estimator for the intercept term in (multi-output) } LR \text{ with input } X_m \text{ and output } Z_m} x^* + \\
&\qquad + \underbrace{\bar{z} - \left(\frac{1}{n}Z\bar{x} - \bar{z}\bar{x}^\top\right)\left(\frac{1}{n}XX^\top - \bar{x}\bar{x}^\top\right)^{-1}\bar{x}}_{\text{the estimator for the bias term in (multi-output) } LR \text{ with input } X_m \text{ and output } Z_m}
\end{aligned}$$

Hence, ***S2UncFA* and (multi-output) *LR* are (strongly) equivalent (with respect to the predicted values)**.

Observation: It is important to note that we now have **a new interpretation for *LR***: one can swap input and output, fit a multi-output *LR*, and apply the Bayes' rule to get the same result as in *LR*.

Observation: In the Machine Learning literature [15, 16], there is a result that says that *Naive/Joint Bayes* and *Logistic Regression* have the same form of the decision boundary (i.e. they are linear classifiers), but, in the end, their fitting algorithms do not return the same parameters. This is not the case in our context: *S2UncFA* and *LR* are not equivalent only because they are both linear, but because their fitting algorithms return the same parameters.

2.3 S2FA - Simple-Supervised Factor Analysis

One important detail is that when we derived the *S2UncFA* algorithm, the matrix Ψ was any symmetric and positive definite matrix, although in *Factor Analysis* (*FA*), Ψ is constrained to be diagonal. We derive an algorithm similar to *S2UncFA*, but this time we impose Ψ to be **diagonal**.

As one may notice, the derivation is almost the same except for $\frac{\partial l_{\text{RV\_D}}}{\partial \Psi}$ which we compute below.

Let $\Psi = \begin{bmatrix} a_1^2 & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & a_D^2 \end{bmatrix}$. Then $\Psi^{-1} = \begin{bmatrix} \frac{1}{a_1^2} & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & \frac{1}{a_D^2} \end{bmatrix}$.

$l_{\text{RV\_D}}$ becomes:

$$
\begin{aligned}
l_{\text{RV\_D}}(\theta) = &-\frac{nd}{2}\ln(2\pi) - \frac{n}{2}\ln(\det(\Sigma_z)) - \frac{1}{2}\sum_{i=1}^{n}(z^{(i)} - \mu_z)^\top \Sigma_z^{-1}(z^{(i)} - \mu_z) - \\
&- \frac{nD}{2}\ln(2\pi) - \frac{n}{2}\ln(\det(\Psi)) - \frac{1}{2}\sum_{i=1}^{n}(x^{(i)} - \mu - \Lambda z^{(i)})^\top \Psi^{-1}(x^{(i)} - \mu - \Lambda z^{(i)}) \\
\overset{2.3}{=} &-\frac{nd}{2}\ln(2\pi) - \frac{n}{2}\ln(\det(\Sigma_z)) - \frac{1}{2}\sum_{i=1}^{n}(z^{(i)} - \mu_z)^\top \Sigma_z^{-1}(z^{(i)} - \mu_z) - \\
&- \frac{nD}{2}\ln(2\pi) - \frac{n}{2}\ln\left(\prod_{i=1}^{D} a_i^2\right) - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{D}(x^{(i)} - \mu - \Lambda z^{(i)})_j^2 \frac{1}{a_j^2} \\
= &-\frac{nd}{2}\ln(2\pi) - \frac{n}{2}\ln(\det(\Sigma_z)) - \frac{1}{2}\sum_{i=1}^{n}(z^{(i)} - \mu_z)^\top \Sigma_z^{-1}(z^{(i)} - \mu_z) - \\
&- \frac{n}{2}\sum_{i=1}^{D} 2\ln a_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{D}(x_j^{(i)} - \mu_j - \Lambda_{j:}z^{(i)})^2 \frac{1}{a_j^2}
\end{aligned}
$$

$$
\frac{\partial l_{\text{RV\_D}}}{\partial a_l} = -\frac{n}{a_l} - \frac{1}{2}\sum_{i=1}^{n}(x_l^{(i)} - \mu - \Lambda_{l:}z^{(i)})^2(-2)\frac{1}{a_l^3} = -\frac{n}{a_l} + \frac{\sum_{i=1}^{n}(x_l^{(i)} - \mu_l - \Lambda_{l:}z^{(i)})^2}{a_l^3}
$$

$$
\frac{\partial l_{\text{RV\_D}}}{\partial a_l} = 0 \Rightarrow \boxed{\hat{a}_l^2 = \frac{\sum_{i=1}^{n}(x^{(i)} - \hat{\mu}_l - \hat{\Lambda}_{l:}z^{(i)})^2}{n}, \forall l = \{1, \dots, D\}}
$$

and we have: $\boxed{\hat{\Psi} = \begin{bmatrix} \hat{a}_1^2 & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & \hat{a}_D^2 \end{bmatrix}}$.

Observation: $\hat{\Psi}$ can also be computed as follows:

$$
\boxed{\hat{\Psi} = \operatorname{diag}\left(\frac{\sum_{i=1}^{n}(x^{(i)} - \hat{\mu} - \hat{\Lambda}z^{(i)})(x^{(i)} - \hat{\mu} - \hat{\Lambda}z^{(i)})^\top}{n}\right)},
$$

where the term in $\operatorname{diag}(\cdot)$ is exactly $\hat{\Psi}$ in *S2UncFA*.

One variant in **matrix form** of the training phase is as follows:

$$
\hat{\mu}_z = \frac{\sum_{i=1}^{n} z^{(i)}}{n}
$$

$$
\hat{\Sigma}_z = \frac{1}{n}\left(Z_m - \hat{\mu}_z \underbrace{\begin{bmatrix} 1 & 1 & \dots & 1 \end{bmatrix}}_{\in \mathbb{R}^{1\times n}}\right)\left(Z_m - \hat{\mu}_z \begin{bmatrix} 1 & 1 & \dots & 1 \end{bmatrix}\right)^\top
$$

Algorithm 3 S2FA

function TRAIN($\{(x^{(i)}, z^{(i)}) | i \in \{1, \dots, n\}\}$)
  $\hat{\mu}_z = \frac{\sum_{i=1}^{n} z^{(i)}}{n}$
  $\hat{\Sigma}_z = \frac{\sum_{i=1}^{n} (z^{(i)} - \hat{\mu}_z)(z^{(i)} - \hat{\mu}_z)^\top}{n}$
  $\hat{\Lambda} = \left(n\bar{x}\bar{z}^\top - \sum_{i=1}^{n} x^{(i)} {z^{(i)}}^\top\right) \left(n\bar{z}\bar{z}^\top - \sum_{i=1}^{n} z^{(i)} {z^{(i)}}^\top\right)^{-1}$
  $\hat{\mu} = \bar{x} - \hat{\Lambda}\bar{z}$
  $\hat{\Psi} = \operatorname{diag}\left(\frac{\sum_{i=1}^{n} (x^{(i)} - \hat{\mu} - \hat{\Lambda} z^{(i)})(x^{(i)} - \hat{\mu} - \hat{\Lambda} z^{(i)})^\top}{n}\right)$
  return ($\hat{\mu}_z$,$\hat{\Sigma}_z$,$\hat{\Lambda}$,$\hat{\mu}$,$\hat{\Psi}$)

function TEST(x^*,($\hat{\mu}_z$,$\hat{\Sigma}_z$,$\hat{\Lambda}$,$\hat{\mu}$,$\hat{\Psi}$))
  value $= \hat{\mu}_z + \hat{\Sigma}_z \hat{\Lambda}^\top (\hat{\Lambda}\hat{\Sigma}_z\hat{\Lambda}^\top + \hat{\Psi})^{-1}(x^* - \hat{\mu} - \hat{\Lambda}\hat{\mu}_z)$
  covarianceMatrix $= \hat{\Sigma}_z - \hat{\Sigma}_z \hat{\Lambda}^\top (\hat{\Lambda}\hat{\Sigma}_z\hat{\Lambda}^\top + \hat{\Psi})^{-1} \hat{\Lambda} \hat{\Sigma}_z^\top$
  return (value, covarianceMatrix)

$$\hat{\Lambda} = (n\bar{x}\bar{z}^\top - X_m Z_m^\top)(n\bar{z}\bar{z}^\top - Z_m Z_m^\top)^{-1}$$
$$\hat{\mu} = \bar{x} - \hat{\Lambda}\bar{z}$$
$$\hat{\Psi} = \operatorname{diag}\left(\frac{1}{n}\left(X_m - \hat{\mu}\underbrace{\begin{bmatrix}1 & 1 & \dots & 1\end{bmatrix}}_{\in \mathbb{R}^{1\times n}} - \hat{\Lambda} Z_m\right)\left(X_m - \hat{\mu}\begin{bmatrix}1 & 1 & \dots & 1\end{bmatrix} - \hat{\Lambda} Z_m\right)^\top\right)$$
$$\text{where } Z_m = \begin{bmatrix} z^{(1)} & \dots & z^{(n)} \end{bmatrix} \text{ and } X_m = \begin{bmatrix} x^{(1)} & \dots & x^{(n)} \end{bmatrix}$$

Observation: The **weak** equivalence to *LR* holds.

Observation: If the **input** is **1-dimensional**, there is no difference between *S2FA* and *S2UncFA*. Hence, the **strong** equivalence to *LR* holds.

Observation: Empirically, we noticed that **multi-output (with d dimensions) *S2FA* is different than d independent single-output *S2FA* tasks unlike the case of *LR* (which is equivalent to *S2UncFA*)**.

2.4 S2PPCA - Simple-Supervised Probabilistic Principal Component Analysis

In *S2FA* we constrained Ψ to be diagonal. Remember that in *FA*, if Ψ is of the form $\eta^2 I$, then *FA* becomes *PPCA*. We derive an algorithm similar to *S2UncFA*, but this time we **impose Ψ to be $\eta^2 I$**.

As discussed for *S2FA*, the derivation is almost the same except for $\frac{\partial l_{\text{RV\_D}}}{\partial \Psi}$ which we compute below.

Let $\Psi = \eta^2 I$, $\eta \in \mathbb{R}, \eta > 0$. Then $\Psi^{-1} = \frac{1}{\eta^2} I$, $\det(\Psi) = \eta^{2D}$.

$l_{\text{RV\_D}}$ becomes:

$$\begin{aligned} l_{\text{RV\_D}}(\theta) = &-\frac{nd}{2}\ln(2\pi) - \frac{n}{2}\ln(\det(\Sigma_z)) - \frac{1}{2}\sum_{i=1}^{n}(z^{(i)} - \mu_z)^\top \Sigma_z^{-1}(z^{(i)} - \mu_z) - \\ &- \frac{nD}{2}\ln(2\pi) - \frac{n}{2}\ln(\det(\Psi)) - \frac{1}{2}\sum_{i=1}^{n}(x^{(i)} - \mu - \Lambda z^{(i)})^\top \Psi^{-1}(x^{(i)} - \mu - \Lambda z^{(i)}) \end{aligned}$$

$$
\begin{aligned}
&= -\frac{nd}{2}\ln(2\pi) - \frac{n}{2}\ln(\det(\Sigma_z)) - \frac{1}{2}\sum_{i=1}^{n}(z^{(i)} - \mu_z)^\top \Sigma_z^{-1}(z^{(i)} - \mu_z) - \\
&\quad - \frac{nD}{2}\ln(2\pi) - \frac{n}{2}\ln\left(\eta^{2D}\right) - \frac{1}{2\eta^2}\sum_{i=1}^{n}(x^{(i)} - \mu - \Lambda z^{(i)})^\top (x^{(i)} - \mu - \Lambda z^{(i)}) \\
&= -\frac{nd}{2}\ln(2\pi) - \frac{n}{2}\ln(\det(\Sigma_z)) - \frac{1}{2}\sum_{i=1}^{n}(z^{(i)} - \mu_z)^\top \Sigma_z^{-1}(z^{(i)} - \mu_z) - \\
&\quad - \frac{nD}{2}\ln(2\pi) - nD\ln(\eta) - \frac{1}{2\eta^2}\sum_{i=1}^{n}(x^{(i)} - \mu - \Lambda z^{(i)})^\top (x^{(i)} - \mu - \Lambda z^{(i)})
\end{aligned}
$$

$$
\frac{\partial l_{\mathrm{RV\_D}}}{\partial \eta} = -\frac{nD}{\eta} - \frac{1}{2}(-2)\frac{1}{\eta^3}\sum_{i=1}^{n}(x^{(i)} - \mu - \Lambda z^{(i)})^\top (x^{(i)} - \mu - \Lambda z^{(i)})
$$

$$
\frac{\partial l_{\mathrm{RV\_D}}}{\partial \eta} = 0 \Rightarrow \boxed{\hat{\eta}^2 = \frac{\sum_{i=1}^{n}(x^{(i)} - \hat{\mu} - \hat{\Lambda} z^{(i)})^\top (x^{(i)} - \hat{\mu} - \hat{\Lambda} z^{(i)})}{n}}
$$

Observation: $\hat{\eta}^2$ can also be computed as follows:

$$
\boxed{\hat{\eta}^2 = \frac{1}{D}\mathrm{Tr}\left(\frac{\sum_{i=1}^{n}(x^{(i)} - \hat{\mu} - \hat{\Lambda} z^{(i)})(x^{(i)} - \hat{\mu} - \hat{\Lambda} z^{(i)})^\top}{n}\right)},
$$

where the term in $\mathrm{Tr}(\cdot)$ is exactly $\hat{\Psi}$ in *S2UncFA*.

Algorithm 4 S2PPCA

function TRAIN($\{(x^{(i)}, z^{(i)}) | i \in \{1, \ldots, n\}\}$)
  $\hat{\mu}_z = \frac{\sum_{i=1}^{n} z^{(i)}}{n}$
  $\hat{\Sigma}_z = \frac{\sum_{i=1}^{n}(z^{(i)} - \hat{\mu}_z)(z^{(i)} - \hat{\mu}_z)^\top}{n}$
  $\hat{\Lambda} = \left(n\bar{x}\bar{z}^\top - \sum_{i=1}^{n} x^{(i)} z^{(i)\top}\right)\left(n\bar{z}\bar{z}^\top - \sum_{i=1}^{n} z^{(i)} z^{(i)\top}\right)^{-1}$
  $\hat{\mu} = \bar{x} - \hat{\Lambda}\bar{z}$
  $\hat{\eta}^2 = \frac{1}{D}\mathrm{Tr}\left(\frac{\sum_{i=1}^{n}(x^{(i)} - \hat{\mu} - \hat{\Lambda} z^{(i)})(x^{(i)} - \hat{\mu} - \hat{\Lambda} z^{(i)})^\top}{n}\right)$
  return $(\hat{\mu}_z, \hat{\Sigma}_z, \hat{\Lambda}, \hat{\mu}, \hat{\eta}^2)$
function TEST($x^*, (\hat{\mu}_z, \hat{\Sigma}_z, \hat{\Lambda}, \hat{\mu}, \hat{\eta}^2)$)
  value $= \hat{\mu}_z + \hat{\Sigma}_z \hat{\Lambda}^\top (\hat{\Lambda}\hat{\Sigma}_z\hat{\Lambda}^\top + \hat{\eta}^2 I)^{-1}(x^* - \hat{\mu} - \hat{\Lambda}\hat{\mu}_z)$
  covarianceMatrix $= \hat{\Sigma}_z - \hat{\Sigma}_z \hat{\Lambda}^\top (\hat{\Lambda}\hat{\Sigma}_z\hat{\Lambda}^\top + \hat{\eta}^2 I)^{-1}\hat{\Lambda}\hat{\Sigma}_z^\top$
  return (value, covarianceMatrix)

One variant in **matrix form** of the training phase is as follows:

$$
\hat{\mu}_z = \frac{\sum_{i=1}^{n} z^{(i)}}{n}
$$

$$
\hat{\Sigma}_z = \frac{1}{n}\left(Z_m - \hat{\mu}_z \underbrace{\begin{bmatrix} 1 & 1 & \ldots & 1 \end{bmatrix}}_{\in \mathbb{R}^{1 \times n}}\right)\left(Z_m - \hat{\mu}_z \begin{bmatrix} 1 & 1 & \ldots & 1 \end{bmatrix}\right)^\top
$$

$$
\hat{\Lambda} = (n\bar{x}\bar{z}^\top - X_m Z_m^\top)(n\bar{z}\bar{z}^\top - Z_m Z_m^\top)^{-1}
$$

$\hat{\mu} = \bar{x} - \hat{\Lambda}\bar{z}$

$$\hat{\eta}^2 = \frac{1}{D}\mathrm{Tr}\left(\frac{1}{n}\left(X_m - \hat{\mu}\underbrace{\begin{bmatrix}1 & 1 & \dots & 1\end{bmatrix}}_{\in\mathbb{R}^{1\times n}} - \hat{\Lambda}Z_m\right)\left(X_m - \hat{\mu}\begin{bmatrix}1 & 1 & \dots & 1\end{bmatrix} - \hat{\Lambda}Z_m\right)^\top\right)$$

where $Z_m = \begin{bmatrix}z^{(1)} & \dots & z^{(n)}\end{bmatrix}$ and $X_m = \begin{bmatrix}x^{(1)} & \dots & x^{(n)}\end{bmatrix}$

Observation: The **weak** equivalence to *LR* holds.

Observation: If the **input** is **1-dimensional**, there is no difference between *S2PPCA* and *S2UncFA*. Hence, the **strong** equivalence to *LR* holds.

Observation: Empirically, we noticed that **multi-output (with d dimensions) *S2PPCA* is different than d independent single-output *S2PPCA* tasks unlike the case of *LR* (which is equivalent to *S2UncFA*)**.

2.5 Standardization-Destandardization Version

In order to **debug** or to check our implementation of the mentioned algorithms, we also implemented another variant for each of them. In such a variant the idea is as follows:

- we **standardize** the output to have mean 0 and covariance matrix I
- we **adapt** the formulas from [4] for the supervised case (which are simpler because $z \sim \mathcal{N}(0, I)$)
- after the test phase, we **destandardize** the predicted values (we also update the returned covariance matrix)

The relevant part here is the standardization-destandardization which we briefly discuss below.

Standardization:

It is known that if $X \sim \mathcal{N}(\mu, \sigma^2)$ we can obtain a new standardized random variable via $Z = \frac{X-\mu}{\sigma} \sim \mathcal{N}(0, I)$.

A similar result holds for random vectors: if $X \sim \mathcal{N}(\mu, \Sigma)$ we can obtain a new standardized random vector via $Z = B^{-1}(X - \mu) \sim \mathcal{N}(0, I)$, where $\Sigma = BB^\top$. B can be computed via Cholesky factorization of Σ or eigenvalue decomposition of Σ [17].

The same process can be applied to data.

$D_X = \{x^{(i)} | i \in \{1, \dots, n\}\}$

$X \sim \mathcal{N}(\mu, \Sigma)$

$x^{(i)} \sim X$, $\forall i \in \{1, \dots, n\}$

$x^{(1)}, \dots, x^{(n)}$ - independent

From this information we can compute the Maximum Likelihood estimators $\mu_{\mathrm{X;MLE}}$, $\Sigma_{\mathrm{X;MLE}}$, $B_{\mathrm{X;MLE}}$ where $\Sigma_{\mathrm{X;MLE}} = B_{\mathrm{X;MLE}}B_{\mathrm{X;MLE}}^\top$.

It can be shown that if we make a new dataset D_Z as follows:

$D_Z = \{z^{(i)} = B_{\mathrm{X;MLE}}^{-1}(x^{(i)} - \mu_{\mathrm{X;MLE}}) | i \in \{1, \dots, n\}\}$ and we compute $\mu_{\mathrm{Z;MLE}}$, $\Sigma_{\mathrm{Z;MLE}}$ we obtain $\mu_{\mathrm{Z;MLE}} = 0$ and $\Sigma_{\mathrm{Z;MLE}} = I$.

This is exactly what we did: we created the dataset D_Z (from D_X, where D_X = the output dataset in the context of *S2UncFA*, *S2FA*, *S2PPCA* $= \{z^{(i)}|i \in \{1, \ldots, n\}\} \neq D_Z$) and provided it to the algorithm.

Destandardization:

In the test phase, we know that, given a new instance x^*, we compute the mean and covariance matrix of the random variable $Z^*|x*$. Let us denote this random variable by Y. So, we get $Y \sim \mathcal{N}(\mu_Y, \Sigma_Y)$. We destandardize it below.

The new random variable we are interested in is: $B_{\text{X;MLE}}Y + \mu_{\text{X;MLE}}$ - it was obtained from the following equation: $Y = B_{\text{X;MLE}}^{-1}(\text{new} - \mu_{\text{X;MLE}})$.

$E[B_{\text{X;MLE}}Y + \mu_{x;MLE}] \overset{\text{lin. of E}}{=} B_{\text{X;MLE}}E[Y] + \mu_{\text{X;MLE}} = B_{\text{X;MLE}}\mu_Y + \mu_{\text{X;MLE}}$.

$Var[B_{\text{X;MLE}} + \mu_{\text{X;MLE}}] \overset{Var[BX]=BVar[X]B^\top}{=} B_{\text{X;MLE}}\Sigma_Y B_{\text{X;MLE}}^\top$

In the end, we will return $B_{\text{X;MLE}}\mu_Y + \mu_{\text{X;MLE}}$ as the predicted value and $B_{\text{X;MLE}}\Sigma_Y B_{\text{X;MLE}}^T$ as the covariance matrix.

Observation: It was observed empirically that those variants are **equivalent** to the initial algorithms: *S2UncFA, S2FA, S2PPCA*. This will not be the case when we talk about the *S3* algorithms.

2.6 S3UncFA - Simple-Semi-Supervised Unconstrained Factor Analysis

We have discussed so far 3 new algorithms that can be used for regression, a **supervised** task. Recall that we started from *FA*, an **unsupervised** task. We will cover in the next pages the **semi-supervised** case where we develop *EM* algorithms that can learn the parameters when the input data is available, but the output data is available or not. Good hints for the case of *Gaussian Naive Bayes* and *Gaussian Mixture Model* are given in [18].

Consider the following **model** where X represents the input attributes and Z, the output attributes in a regression problem. In this setup, we **do not constrain Ψ to be diagonal**.

$Z \sim \mathcal{N}(\mu_z, \Sigma_z)$, $Z \in \mathbb{R}^{d\times 1}$, $\mu_z \in \mathbb{R}^{d\times 1}$, $\Sigma_z \in \mathbb{R}^{d\times d}$

$X|Z \sim \mathcal{N}(\mu + \Lambda Z, \Psi)$, $X \in \mathbb{R}^{D\times 1}, \mu \in \mathbb{R}^{D\times 1}, \Lambda \in \mathbb{R}^{D\times d}$, $\Psi \in \mathbb{R}^{D\times D}$ - symmetric and positive definite matrix

$\theta = (\mu_z, \Sigma_z, \mu, \Lambda, \Psi)$

$(X^{(i)}, Z^{(i)})|\theta \sim (X, Z)|\theta$, $i \in \{1, \ldots, n\}$

$(X^{(1)}, Z^{(1)}), \ldots, (X^{(n)}, Z^{(n)})$ - independent, given the parameters θ

$Z^{(a+1)}, \ldots, Z^{(n)}$ - latent variables

RV_Do $= ((X^{(1)}, Z^{(1)}), \ldots, (X^{(a)}, Z^{(a)}), X^{(a+1)}, \ldots, X^{(n)})$ - random variable for the observed data

RV_Dc $= ((X^{(1)}, Z^{(1)}), \ldots, (X^{(a)}, Z^{(a)}), (X^{(a+1)}, Z^{(a+1)}), \ldots, (X^{(n)}, Z^{(n)}))$ - random variable for the complete data

RV_Dl $= (Z^{(a+1)}, \ldots, Z^{(n)})$ - random variable for the latent data

Do $= ((x^{(1)}, z^{(1)}), \ldots, (x^{(a)}, z^{(a)}), x^{(a+1)}, \ldots, x^{(n)})$ - the observed data

Dc $= ((x^{(1)}, z^{(1)}), \ldots, (x^{(a)}, z^{(a)}), (x^{(a+1)}, Z^{(a+1)}), \ldots, (x^{(n)}, Z^{(n)}))$ - the complete data

Dl $= (Z^{(a+1)}, \ldots, Z^{(n)})$ - the latent data

Before we start, we make the following observation: if X is a random variable with p_X its probability density/mass function, then $p_X(X)$ is also a random variable. For example: if $X \sim \mathcal{N}(\mu, \sigma^2)$ with $p_X(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}$ its probability density function, then $p_X(X) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{1}{2}\left(\frac{X-\mu}{\sigma}\right)^2}$ is also a random variable.

E step:

We can write down the **log-likelihood of the complete data**:

$$
\begin{aligned}
l_{\text{RV\_Dc}}(\theta) &= \ln p_{\text{RV\_Dc}|\theta}(Dc|\theta) \\
&= \ln p_{\text{RV\_Dc}|\theta}((x^{(1)}, z^{(1)}), \ldots, (x^{(a)}, z^{(a)}), (x^{(a+1)}, Z^{(a+1)}), \ldots, (x^{(n)}, Z^{(n)})|\theta) \\
&\overset{\text{indep.}}{=} \ln\left(\left(\prod_{i=1}^{a} p_{X^{(i)},Z^{(i)}|\theta}(x^{(i)}, z^{(i)}|\theta)\right)\left(\prod_{i=a+1}^{n} p_{X^{(i)},Z^{(i)}|\theta}(x^{(i)}, Z^{(i)}|\theta)\right)\right), \\
&\qquad\qquad\qquad \text{since } E_{(X,Y)}[X+Y] = E_X[X] + E_Y[Y] \\
&= \sum_{i=1}^{a} \ln p_{X^{(i)},Z^{(i)}|\theta}(x^{(i)}, z^{(i)}|\theta) + \sum_{i=a+1}^{n} \ln p_{X^{(i)},Z^{(i)}|\theta}(x^{(i)}, Z^{(i)}|\theta) \\
&\overset{\text{mult.rule}}{=} \sum_{i=1}^{a} \ln(p_{Z^{(i)}|\theta}(z^{(i)}|\theta) p_{X^{(i)}|Z^{(i)},\theta}(x^{(i)}|z^{(i)}, \theta)) + \\
&\qquad\qquad + \sum_{i=a+1}^{n} \ln(p_{Z^{(i)}|\theta}(Z^{(i)}|\theta) p_{X^{(i)}|Z^{(i)},\theta}(x^{(i)}|Z^{(i)}, \theta)) \\
&= \sum_{i=1}^{a} \Big(\ln(p_{Z^{(i)}|\theta}(z^{(i)}|\theta)) + \ln(p_{X^{(i)}|Z^{(i)},\theta}(x^{(i)}|z^{(i)}, \theta))\Big) + \\
&\qquad\qquad + \sum_{i=a+1}^{n} \Big(\ln(p_{Z^{(i)}|\theta}(Z^{(i)}|\theta)) + \ln(p_{X^{(i)}|Z^{(i)},\theta}(x^{(i)}|Z^{(i)}, \theta))\Big) \\
&= \sum_{i=1}^{a} \ln\left(\frac{1}{\sqrt{(2\pi)^d \det(\Sigma_z)}} e^{-\frac{1}{2}(z^{(i)}-\mu_z)^\top \Sigma_z^{-1}(z^{(i)}-\mu_z)}\right) + \\
&\qquad + \sum_{i=1}^{a} \ln\left(\frac{1}{\sqrt{(2\pi)^D \det(\Psi)}} e^{-\frac{1}{2}(x^{(i)}-\mu-\Lambda z^{(i)})^\top \Psi^{-1}(x^{(i)}-\mu-\Lambda z^{(i)})}\right) + \\
&+ \sum_{i=a+1}^{n} \ln\left(\frac{1}{\sqrt{(2\pi)^d \det(\Sigma_z)}} e^{-\frac{1}{2}(Z^{(i)}-\mu_z)^\top \Sigma_z^{-1}(Z^{(i)}-\mu_z)}\right) + \\
&\qquad + \sum_{i=a+1}^{n} \ln\left(\frac{1}{\sqrt{(2\pi)^D \det(\Psi)}} e^{-\frac{1}{2}(x^{(i)}-\mu-\Lambda Z^{(i)})^\top \Psi^{-1}(x^{(i)}-\mu-\Lambda Z^{(i)})}\right) \\
&= \sum_{i=1}^{a} \left(-\frac{d}{2}\ln(2\pi) - \frac{1}{2}\ln(\det(\Sigma_z)) - \frac{1}{2}(z^{(i)} - \mu_z)^\top \Sigma_z^{-1}(z^{(i)} - \mu_z)\right) + \\
&\qquad + \sum_{i=1}^{a} \left(-\frac{D}{2}\ln(2\pi) - \frac{1}{2}\ln(\det(\Psi)) - \frac{1}{2}(x^{(i)} - \mu - \Lambda z^{(i)})^\top \Psi^{-1}(x^{(i)} - \mu - \Lambda z^{(i)})\right) + \\
&+ \sum_{i=a+1}^{n} \left(-\frac{d}{2}\ln(2\pi) - \frac{1}{2}\ln(\det(\Sigma_z)) - \frac{1}{2}(Z^{(i)} - \mu_z)^\top \Sigma_z^{-1}(Z^{(i)} - \mu_z)\right) + \\
&\qquad + \sum_{i=a+1}^{n} \left(-\frac{D}{2}\ln(2\pi) - \frac{1}{2}\ln(\det(\Psi)) - \frac{1}{2}(x^{(i)} - \mu - \Lambda Z^{(i)})^\top \Psi^{-1}(x^{(i)} - \mu - \Lambda Z^{(i)})\right) \\
&= -\frac{nd}{2}\ln(2\pi) - \frac{n}{2}\ln(\det(\Sigma_z)) - \frac{1}{2}\sum_{i=1}^{a}(z^{(i)} - \mu_z)^\top \Sigma_z^{-1}(z^{(i)} - \mu_z) -
\end{aligned}
$$

$$-\frac{nD}{2}\ln(2\pi)-\frac{n}{2}\ln(\det(\Psi))-\frac{1}{2}\sum_{i=1}^{a}(x^{(i)}-\mu-\Lambda z^{(i)})^\top\Psi^{-1}(x^{(i)}-\mu-\Lambda z^{(i)})-$$
$$-\frac{1}{2}\sum_{i=a+1}^{n}(Z^{(i)}-\mu_z)^\top\Sigma_z^{-1}(Z^{(i)}-\mu_z)-\frac{1}{2}\sum_{i=a+1}^{n}(x^{(i)}-\mu-\Lambda Z^{(i)})^\top\Psi^{-1}(x^{(i)}-\mu-\Lambda Z^{(i)})$$

Before we proceed we make the following observations: (2.10)

- $E_{(X,Y)}[X+Y]=E_X[X]+E_Y[Y]$
- Let $(X_1,Y_1)\sim(X,Y),(X_2,Y_2)\sim(X,Y)$, $(X_1,Y_1),(X_2,Y_2)$ - independent.

  $p((x_1,y_1),(x_2,y_2))\overset{\text{indep.}}{=}p(x_1,y_1)p(x_2,y_2)$

  $$p(x_1,x_2)\overset{\text{marginal}}{=}\sum_{y_1,y_2}p(x_1,y_1,x_2,y_2)=\overset{\text{indep.}}{=}\sum_{y_1,y_2}p(x_1,y_1)p(x_2,y_2)=$$
  $$=\left(\sum_{y_1}p(x_1,y_1)\right)\left(\sum_{y_2}p(x_2,y_2)\right)\overset{\text{marginal}}{=}p(x_1)p(x_2)$$

  Note that by replacing the sum sign by the integral sign, the above formula holds also for continuous random variables.

  Almost the same proof for:

  $$p(y_1,y_2)=p(y_1)p(y_2). \tag{2.11}$$

  $$p(x_1,x_2|y_1,y_2)\overset{\text{cond.pb.}}{=}\frac{p(x_1,x_2,y_1,y_2)}{p(y_1,y_2)}\overset{\text{indep.2.11}}{=}\frac{p(x_1,y_1)p(x_2,y_2)}{p(y_1)p(y_2)}=$$
  $$=\frac{p(x_1,y_1)}{p(y_1)}\frac{p(x_2,y_2)}{p(y_2)}\overset{\text{cond.pb.}}{=}p(x_1|y_1)p(x_2|y_2)$$

  Almost the same proof for:

  $p(y_1,y_2|x_1,x_2)=p(y_1|x_1)p(y_2|x_2)$

  Note that by replacing the sum sign by the integral sign, the above formula holds also for continuous random variables.

  $$p(x_1,y_1,x_2)\overset{\text{marginal}}{=}\sum_{y_2}p(x_1,y_1,x_2,y_2)\overset{\text{indep.}}{=}\sum_{y_2}p(x_1,y_1)p(x_2,y_2)=$$
  $$=p(x_1,y_1)\sum_{y_2}p(x_2,y_2)=p(x_1,y_1)p(x_2)$$

  Note that by replacing the sum sign by the integral sign, the above formula holds also for continuous random variables.

  This whole observation can be extended to $(X_1,Y_1),\dots,(X_n,Y_n)$.

- In [4], they swap the gradient sign and the expectation sign without saying why this is correct. We will proceed differently, by following some ideas from [19].

  $$(Z^{(i)}-\mu_z)^\top\Sigma_z^{-1}(Z^{(i)}-\mu_z)=\underbrace{{Z^{(i)}}^\top\Sigma_z^{-1}Z^{(i)}}_{\in\mathbb{R}}-{Z^{(i)}}^\top\Sigma_z^{-1}\mu_z-\mu_z^\top\Sigma_z^{-1}Z^{(i)}+\mu_z^\top\Sigma_z^{-1}\mu_z$$

$$Z^{(i)^\top}\Sigma_z^{-1}Z^{(i)} \overset{\in\mathbb{R}}{=} \mathrm{Tr}\left(Z^{(i)^\top}\Sigma_z^{-1}Z^{(i)}\right) \overset{[11,(16)]}{=} \mathrm{Tr}\left(\Sigma_z^{-1}Z^{(i)}Z^{(i)^\top}\right)$$

$$\begin{aligned}(x^{(i)}-\mu-\Lambda Z^{(i)})^\top\Psi^{-1}(x^{(i)}-\mu-\Lambda Z^{(i)}) &= ((x^{(i)}-\mu)^\top-(\Lambda Z^{(i)})^\top)\Psi^{-1}(x^{(i)}-\mu-\Lambda Z^{(i)}) = \\ &= (x^{(i)}-\mu)^\top\Psi^{-1}(x^{(i)}-\mu)-(x^{(i)}-\mu)^\top\Psi^{-1}\Lambda Z^{(i)}-(\Lambda Z^{(i)})^\top\Psi^{-1}(x^{(i)}-\mu)+ \\ &+ \underbrace{(\Lambda Z^{(i)})^\top\Psi^{-1}\Lambda Z^{(i)}}_{\in\mathbb{R}}\end{aligned}$$

$$\begin{aligned}(\Lambda Z^{(i)})^\top\Psi^{-1}\Lambda Z^{(i)} \overset{\in\mathbb{R}}{=} \mathrm{Tr}((\Lambda Z^{(i)})^\top\Psi^{-1}\Lambda Z^{(i)}) &\overset{[11,(16)]}{=} \mathrm{Tr}(\Psi^{-1}\Lambda Z^{(i)}(\Lambda Z^{(i)})^\top) = \\ &= \mathrm{Tr}(\Psi^{-1}\Lambda Z^{(i)}Z^{(i)^\top}\Lambda^\top)\end{aligned}$$

- From [20], we have the following (A - is a matrix):
  $$\mathrm{Tr}(E[A]) = \mathrm{Tr}\left(\begin{bmatrix}E[a_{11}] & \dots & E[a_{1n}]\\ \vdots & \ddots & \vdots \\ E[a_{n1}] & \dots & E[a_{nn}]\end{bmatrix}\right) = \textstyle\sum_{i=1}^n E[a_{ii}] \overset{\text{lin. of. E}}{=} E[\sum_{i=1}^n a_{ii}] = E[\mathrm{Tr}(A)]$$
  Hence, we can swap the expectation sign and the trace sign.

We return to the *EM* algorithm:

$$\begin{aligned}Q(\theta|\theta^{(t)}) &= E_{\mathrm{RV\_Dl|RV\_Do=Do},\theta^{(t)}}[l_{\mathrm{RV\_Dc}}(\theta)] \\ &\overset{\text{lin. of E;2.10}}{=} -\frac{n(d+D)}{2}\ln(2\pi)-\frac{n}{2}\ln(\det(\Sigma_z))-\frac{n}{2}\ln(\det(\Psi))- \\ &-\frac{1}{2}\sum_{i=1}^{a}(z^{(i)}-\mu_z)^\top\Sigma_z^{-1}(z^{(i)}-\mu_z)-\frac{1}{2}\sum_{i=a+1}^{n}(\mathrm{Tr}(\Sigma_z^{-1}E_{Z^{(i)}|X^{(i)}=x^{(i)},\theta^{(t)}}[Z^{(i)}Z^{(i)^\top}])- \\ &-E_{Z^{(i)}|X^{(i)}=x^{(i)},\theta^{(t)}}[Z^{(i)^\top}]\Sigma_z^{-1}\mu_z-\mu_z^\top\Sigma_z^{-1}E_{Z^{(i)}|X^{(i)}=x^{(i)},\theta^{(t)}}[Z^{(i)}]+\mu_z^\top\Sigma_Z^{-1}\mu_z)- \\ &-\frac{1}{2}\sum_{i=1}^{a}(x^{(i)}-\mu-\Lambda z^{(i)})^\top\Psi^{-1}(x^{(i)}-\mu-\Lambda z^{(i)})- \\ &-\frac{1}{2}\sum_{i=a+1}^{n}((x^{(i)}-\mu)^\top\Psi^{-1}(x^{(i)}-\mu)-(x^{(i)}-\mu)^\top\Lambda E_{Z^{(i)}|X^{(i)}=x^{(i)},\theta^{(t)}}[Z^{(i)}]- \\ &-E_{Z^{(i)}|X^{(i)}=x^{(i)},\theta^{(t)}}[Z^{(i)^\top}]\Lambda^\top\Psi^{-1}(x^{(i)}-\mu)+\mathrm{Tr}(\Psi^{-1}\Lambda E_{Z^{(i)}|X^{(i)}=x^{(i)},\theta^{(t)}}[Z^{(i)}Z^{(i)^\top}])\Lambda^\top)\end{aligned}$$

M step:

We will omit the E's subscript for brevity.

$$\begin{aligned}\frac{\partial Q}{\partial\mu_z} &\overset{[11,(69,84,85)];\Sigma_z-sym.\Rightarrow\Sigma_z^{-1}-sym.}{=} -\frac{1}{2}\sum_{i=1}^{a}(-2(-I)^\top\Sigma_z^{-1}(z^{(i)}-\mu_z))- \\ &-\frac{1}{2}\sum_{i=a+1}^{n}(-(E[Z^{(i)^\top}]\Sigma_z^{-1})^\top-\Sigma_z^{-1}E[Z^{(i)}]+2\Sigma_z^{-1}\mu_z) \overset{\Sigma_z-sym.}{=} \\ &= \sum_{i=1}^{a}\Sigma_z^{-1}(z^{(i)}-\mu_z)-\frac{1}{2}\sum_{i=a+1}^{n}(-\Sigma_z^{-1}E[Z^{(i)}]-\Sigma_z^{-1}E[Z^{(i)}]+2\Sigma_z^{-1}\mu_z) \\ &= \sum_{i=1}^{a}\Sigma_z^{-1}(z^{(i)}-\mu_z)-\sum_{i=a+1}^{n}(\Sigma_z^{-1}\mu_z-\Sigma_z^{-1}E[Z^{(i)}])\end{aligned}$$

$$\frac{\partial Q}{\partial \mu_z} = 0 \Rightarrow \sum_{i=1}^{a}(z^{(i)} - \mu_z) - \sum_{i=a+1}^{n}(\mu_z - E[Z^{(i)}]) = 0 \Rightarrow$$

$$\Rightarrow \boxed{\hat{\mu}_z = \frac{\sum_{i=1}^{a} z^{(i)} + \sum_{a+1}^{n} E[Z^{(i)}]}{n}} \tag{2.12}$$

$$\frac{\partial Q}{\partial \Sigma_z} \overset{[11,(61),(124)][12,(4b)]}{=} -\frac{n}{2}\Sigma_z^{-\top} - \frac{1}{2}\sum_{i=1}^{a}(-\Sigma_z^{-\top}(z^{(i)} - \mu_z)(z^{(i)} - \mu_z)^\top \Sigma_z^{-\top}) -$$
$$- \frac{1}{2}\sum_{i=a+1}^{n}(-\Sigma_z^{-\top} I^\top (E[Z^{(i)}Z^{(i)\top}])^\top \Sigma_z^{-\top} - (-\Sigma_z^{-\top} E[Z^{(i)}\top]^\top \mu_z^\top \Sigma_z^{-\top}) -$$
$$- (-\Sigma_z^{-\top}(\mu_z^\top)^\top E[Z^{(i)}]^\top \Sigma_z^{-\top}) + (-\Sigma_z^{-\top}(\mu_z^\top)^\top \mu_z^\top \Sigma_z^{-\top}))$$
$$\overset{\Sigma_z^{-\top} = \Sigma_z^{-1}}{=} -\frac{n}{2}\Sigma_z^{-1} + \frac{1}{2}\sum_{i=1}^{a}\Sigma_z^{-1}(z^{(i)} - \mu_z)(z^{(i)} - \mu_z)^\top \Sigma_z^{-1} - \frac{1}{2}\sum_{i=a+1}^{n}(-\Sigma_z^{-1}E[Z^{(i)}Z^{(i)\top}]\Sigma_z^{-1} +$$
$$+ \Sigma_z^{-1}E[Z^{(i)}]\mu_z^\top \Sigma_z^{-1} + \Sigma_z^{-1}\mu_z E[Z^{(i)}]\Sigma_z^{-1} - \Sigma_z^{-1}\mu_z\mu_z^\top \Sigma_z^{-1})$$

$$\frac{\partial Q}{\partial \Sigma_z} = 0 \overset{\Sigma_z \cdot | \ldots | \Sigma_z}{\Rightarrow} -\frac{n}{2}\Sigma_z + \frac{1}{2}\sum_{i=1}^{a}(z^{(i)} - \mu_z)(z^{(i)} - \mu_z)^\top - \frac{1}{2}\sum_{i=a+1}^{n}(-E[Z^{(i)}Z^{(i)\top}] +$$
$$+ E[Z^{(i)}]\mu_z^\top + \mu_z E[Z^{(i)}]^\top - \mu_z\mu_z^\top) = 0 \Rightarrow$$

$$\Rightarrow \boxed{\hat{\Sigma}_z = \frac{\sum_{i=1}^{a}(z^{(i)} - \hat{\mu}_z)(z^{(i)} - \hat{\mu}_z)^\top + \sum_{i=a+1}^{n}(E[Z^{(i)}Z^{(i)\top}] - E[Z^{(i)}]\hat{\mu}_z^\top - \hat{\mu}_z E[Z^{(i)}]^\top + \hat{\mu}_z\hat{\mu}_z^\top)}{n}} \tag{2.13}$$

$$\frac{\partial Q}{\partial \mu} \overset{[11,(69,86)]}{=} -\frac{1}{2}\sum_{i=1}^{a} -2I^\top \Psi^{-1}(x^{(i)} - \mu - \Lambda z^{(i)}) - \frac{1}{2}\sum_{i=a+1}^{n}(-2I^\top \Psi^{-1}(x^{(i)} - \mu) +$$
$$+ \Psi^{-1}\Lambda E[Z^{(i)}] + \Psi^{-1}\Lambda E[Z^{(i)}]) = \sum_{i=1}^{a}\Psi^{-1}(x^{(i)} - \mu - \Lambda z^{(i)}) +$$
$$+ \sum_{i=a+1}^{n}\Psi^{-1}(x^{(i)} - \mu - \Lambda E[Z^{(i)}])$$

$$\frac{\partial Q}{\partial \mu} = 0 \Rightarrow \mu = \frac{\sum_{i=1}^{a}(x^{(i)} - \Lambda z^{(i)}) + \sum_{i=a+1}^{n}(x^{(i)} - \Lambda E[Z^{(i)}])}{n}$$

$$\frac{\partial Q}{\partial \Lambda} \overset{[11,(88,118,70,71)]}{=} -\frac{1}{2}\sum_{i=1}^{a}(-2\Psi^{-1}(x^{(i)} - \mu - \Lambda z^{(i)})z^{(i)\top}) -$$
$$- \frac{1}{2}\sum_{i=a+1}^{n}(\Psi^{-\top} I^\top \Lambda (E[Z^{(i)}Z^{(i)\top}])^\top + I\Psi^{-1}\Lambda E[Z^{(i)}Z^{(i)\top}] -$$
$$- ((x^{(i)} - \mu)^\top \Psi^{-1})^\top E[Z^{(i)}]^\top - \Psi^{-1}(x^{(i)} - \mu)E[Z^{(i)}]^\top)$$
$$= \sum_{i=1}^{a}\Psi^{-1}(x^{(i)} - \mu - \Lambda z^{(i)})z^{(i)\top} + \sum_{i=a+1}^{n}(\Psi^{-1}(x^{(i)} - \mu)E[Z^{(i)}]^\top - \Psi^{-1}\Lambda E[Z^{(i)}Z^{(i)\top}])$$

$$\frac{\partial Q}{\partial \Lambda} = 0 \Rightarrow \sum_{i=1}^{a}(x^{(i)} - \mu - \Lambda z^{(i)}){z^{(i)}}^\top + \sum_{i=a+1}^{n}((x^{(i)} - \mu)E[{Z^{(i)}}^\top] - \Lambda E[Z^{(i)}{Z^{(i)}}^\top]) = 0 \Rightarrow$$

$$\Lambda = \left(\sum_{i=1}^{a}(x^{(i)} - \mu){z^{(i)}}^\top + \sum_{i=a+1}^{n}(x^{(i)} - \mu)E[{Z^{(i)}}^\top]\right)\left(\sum_{i=1}^{a} z^{(i)}{z^{(i)}}^\top + \sum_{i=a+1}^{n} E[Z^{(i)}{Z^{(i)}}^\top]\right)^{-1}$$

We omit the next steps of finding $\hat{\Lambda}$, $\hat{\mu}$ for brevity. These steps are very similar to those discussed in the section deriving *S2UncFA*.

$$\boxed{\hat{\Lambda} = \left(n\bar{x}\hat{\mu}_z^\top - \sum_{i=1}^{a} x^{(i)}{z^{(i)}}^\top - \sum_{i=a+1}^{n} x^{(i)}E[Z^{(i)}]^\top\right)\left(n\hat{\mu}_z\hat{\mu}_z^\top - \sum_{i=1}^{a} z^{(i)}{z^{(i)}}^\top - \sum_{i=a+1}^{n} E[Z^{(i)}{Z^{(i)}}^\top]\right)^{-1}} \tag{2.14}$$

$$\boxed{\hat{\mu} = \bar{x} - \hat{\Lambda}\hat{\mu}_z} \tag{2.15}$$

$$\frac{\partial Q}{\partial \Psi} \overset{[11,(61,124,57)]}{=} -\frac{n}{2}\Psi^{-\top} - \frac{1}{2}\sum_{i=1}^{a}(-\Psi^{-1}(x^{(i)} - \mu - \Lambda z^{(i)})(x^{(i)} - \mu - \Lambda z^{(i)})^\top \Psi^{-\top}) -$$
$$- \frac{1}{2}\sum_{i=a+1}^{n}(-\Psi^{-\top}(x^{(i)} - \mu)(x^{(i)} - \mu)^\top\Psi^{-\top} - (-\Psi^{-\top}(x^{(i)} - \mu)E[Z^{(i)}]^\top\Lambda^\top\Psi^{-\top}) -$$
$$- (-\Psi^{-\top}(E[{Z^{(i)}}^\top]\Lambda^\top)^\top(x^{(i)} - \mu)^\top\Psi^{-\top}) + (-\Psi^{-\top}I^\top(\Lambda E[Z^{(i)}{Z^{(i)}}^\top]\Lambda^\top)^\top\Psi^{-1}))$$

$$\frac{\partial Q}{\partial \Psi} = 0 \overset{\Psi^{-\top}=\Psi^{-1}}{\Rightarrow} -n\Psi + \sum_{i=1}^{a}(x^{(i)} - \mu - \Lambda z^{(i)})(x^{(i)} - \mu - \Lambda z^{(i)})^\top + \sum_{i=a+1}^{n}((x^{(i)} - \mu)(x^{(i)} - \mu)^\top -$$
$$- (x^{(i)} - \mu)E[Z^{(i)}]^\top\Lambda^\top - \Lambda E[Z^{(i)}](x^{(i)} - \mu)^\top + \Lambda E[Z^{(i)}{Z^{(i)}}^\top]\Lambda^\top) \Rightarrow$$

$$\boxed{\begin{aligned}\Rightarrow \hat{\Psi} = \frac{1}{n}(&\sum_{i=1}^{a}(x^{(i)} - \hat{\mu} - \hat{\Lambda}z^{(i)})(x^{(i)} - \hat{\mu} - \hat{\Lambda}z^{(i)})^\top + \sum_{i=a+1}^{n}((x^{(i)} - \hat{\mu})(x^{(i)} - \hat{\mu})^\top - \\ &-(x^{(i)} - \hat{\mu})E[Z^{(i)}]^\top\hat{\Lambda}^\top - \hat{\Lambda}E[Z^{(i)}](x^{(i)} - \hat{\mu})^\top + \hat{\Lambda}E[Z^{(i)}{Z^{(i)}}^\top]\hat{\Lambda}^\top))\end{aligned}} \tag{2.16}$$

which is symmetric and usually (in practice) positive definite $\Rightarrow$ **constraint is satisfied**

These were the **E and M steps**. The *EM* algorithm increases at each iteration the **log-likelihood** of the **observed** data:

$$\begin{aligned}
l_{\text{RV_Do}} &= \ln\left(p_{\text{RV_Do}|\theta}\left(Do|\theta\right)\right) \\
&= \ln\left(p_{\text{RV_Do}|\theta}\left((x^{(1)}, z^{(1)}), \ldots, (x^{(a)}, z^{(a)}), x^{(a+1)}, \ldots, x^{(n)}|\theta\right)\right) \\
&\overset{\text{indep.;2.10}}{=} \ln\left(\prod_{i=1}^{a} p_{X^{(i)},Z^{(i)}|\theta}(x^{(i)}, z^{(i)}|\theta) \prod_{i=a+1}^{n} p_{X^{(i)}|\theta}(x^{(i)}|\theta)\right) \\
&= \sum_{i=1}^{a}\ln\left(p_{X^{(i)},Z^{(i)}|\theta}(x^{(i)}, z^{(i)}|\theta)\right) + \sum_{i=a+1}^{n}\ln\left(p_{X^{(i)}|\theta}(x^{(i)}|\theta)\right) \\
&\overset{\text{mult.rule}}{=} \sum_{i=1}^{a}\ln\left(p_{Z^{(i)}|\theta}(z^{(i)}|\theta)p_{X^{(i)}|Z^{(i)},\theta}(x^{(i)}|z^{(i)}, \theta)\right) + \sum_{i=a+1}^{n}\ln\left(p_{X^{(i)}|\theta}(x^{(i)}|\theta)\right)
\end{aligned}$$

$$= \sum_{i=1}^{a} \left(\ln \left(\mathcal{N}(z^{(i)}|\mu_z, \Sigma_z) \mathcal{N}(x^{(i)}|\mu + \Lambda z^{(i)}, \Psi) \right) \right) + $$
$$+ \sum_{i=a+1}^{n} \ln \left(\mathcal{N}(x^{(i)}|\mu + \Lambda\mu_z, \Lambda\Sigma_z\Lambda^\top + \Psi) \right)$$

$$= \boxed{\sum_{i=1}^{a} \left(\ln \left(\mathcal{N}(z^{(i)}|\mu_z, \Sigma_z) \right) + \ln \left(\mathcal{N}(x^{(i)}|\mu + \Lambda z^{(i)}, \Psi) \right) \right) + \sum_{i=a+1}^{n} \ln \left(\mathcal{N}(x^{(i)}|\mu + \Lambda\mu_z, \Lambda\Sigma_z\Lambda^\top + \Psi) \right)} \tag{2.17}$$

We compute $E_{Z^{(i)}|X^{(i)}=x^{(i)},\theta^{(t)}}[Z^{(i)}]$ and $E_{Z^{(i)}|X^{(i)}=x^{(i)},\theta^{(t)}}[Z^{(i)}{Z^{(i)}}^\top]$ as follows using the ideas in 2.6:

- $$\boxed{E_{Z^{(i)}|X^{(i)}=x^{(i)},\theta^{(t)}}[Z^{(i)}] = \mu_z^{(t)} + \Sigma_z^{(t)}{\Lambda^{(t)}}^\top(\Lambda^{(t)}\Sigma_z^{(t)}{\Lambda^{(t)}}^\top + \Psi^{(t)})^{-1}(x^{(i)} - \mu^{(t)} - \Lambda^{(t)}\mu_z^{(t)})} \tag{2.18}$$

- It is known the fact that $\mathrm{Cov}[X,Y] = E[XY^\top] - E[X]E[Y^\top] \Rightarrow E[XY^\top] = Cov[X,Y] +$ $+ E[X]E[Y^\top]$ Hence:
$$\boxed{E_{Z^{(i)}|X^{(i)}=x^{(i)},\theta^{(t)}}[Z^{(i)}{Z^{(i)}}^\top] = \Sigma_z^{(t)} + E_{Z^{(i)}|X^{(i)}=x^{(i)},\theta^{(t)}}[Z^{(i)}]} \tag{2.19}$$

Observation: The algorithm can be rewritten in **matrix form** using the same principles used when developing *S2UncFA* in matrix form.

Observation: We implemented the **standardization-destandardization** version of the algorithm, i.e. when $z \sim \mathcal{N}(0, I)$. Theoretically, we noticed that although for *FA* $\hat{\mu} = \frac{\sum_{i=1}^{n} x^{(i)}}{n}$ and this is found by solving the equation $\frac{\partial l_{\text{RV\_Dc}}}{\partial \mu} = 0$ (notice it is not $l_{\text{RV\_Do}}$!), in our case that cannot be trivially proved. Empirically, the two variants of the algorithm differ and the standardization-destandardization variant still increases the log-likelihood of the observed data at each iteration.

As we said, we cannot prove that $\hat{\mu} = \frac{\sum_{i=1}^{n} x^{(i)}}{n}$. We discuss this below.

Let $\{z^{(i)}|i \in \{1,\ldots,a\}\}$ be standardized and $Z^{(i)} \sim \mathcal{N}(0, I), i \in \{a+1,\ldots,n\}$.

We write down the log-likelihood of the observed data:

$$l_{RV\_Do}(\theta) = \ln \left(p_{\text{RV\_Do}|\theta}(Do|\theta) \right)$$
$$= \sum_{i=1}^{a} \left(-\frac{d}{2}\ln(2\pi) - \frac{1}{2}{z^{(i)}}^\top z^{(i)} \right) +$$
$$+ \sum_{i=1}^{a} \left(-\frac{D}{2}\ln(2\pi) - \frac{1}{2}\ln\left(\det(\Psi)\right) - \frac{1}{2}(x^{(i)} - \mu - \Lambda z^{(i)})^\top\Psi^{-1}(x^{(i)} - \mu - \Lambda z^{(i)}) \right) +$$
$$+ \sum_{i=a+1}^{n} \left(-\frac{D}{2}\ln(2\pi) - \frac{1}{2}\ln(\det(\Lambda\Lambda^\top + \Psi)) - \frac{1}{2}(x^{(i)} - \mu)^\top(\Lambda\Lambda^\top + \Psi)^{-1}(x^{(i)} - \mu) \right)$$

$$\frac{\partial l_{\text{RV\_Do}}}{\partial \mu} \overset{[11,(86)]}{=} \sum_{i=1}^{a} \left(-\frac{1}{2}\left(-2\Psi^{-1}(x^{(i)} - \mu - \Lambda z^{(i)}) \right) \right) + \sum_{i=a+1}^{n} \left(-\frac{1}{2}\left(-2(\Lambda\Lambda^\top + \Psi)^{-1}(x^{(i)} - \mu) \right) \right)$$

$$= \sum_{i=1}^{a} \Psi^{-1}(x^{(i)} - \mu - \Lambda z^{(i)}) + \sum_{i=a+1}^{n} (\Lambda\Lambda^\top + \Psi)^{-1}(x^{(i)} - \mu)$$

$$= \sum_{i=1}^{a} \Psi^{-1}x^{(i)} - a\Psi^{-1}\mu - a\Psi^{-1}\Lambda\bar{z} + \sum_{i=a+1}^{n} (\Lambda\Lambda^\top + \Psi)^{-1}x^{(i)} - (n-a)(\Lambda\Lambda^\top + \Psi)^{-1}\mu$$

$$\overset{\bar{z}=0}{=} \sum_{i=1}^{a} \Psi^{-1}x^{(i)} - a\Psi^{-1}\mu + \sum_{i=a+1}^{n} (\Lambda\Lambda^\top + \Psi)^{-1}x^{(i)} - (n-a)(\Lambda\Lambda^\top + \Psi)^{-1}\mu$$

$$\frac{\partial l_{\text{RV\_Do}}}{\partial \mu} = 0 \Rightarrow \hat{\mu} = \left(a\hat{\Psi}^{-1} + (n-a)(\hat{\Lambda}\hat{\Lambda}^\top + \hat{\Psi})^{-1}\right)^{-1} \left(\sum_{i=1}^{a} \hat{\Psi}^{-1}x^{(i)} + \sum_{i=a+1}^{n} (\hat{\Lambda}\hat{\Lambda}^\top + \hat{\Psi})^{-1}x^{(i)}\right)$$

So, we do not obtain that $\hat{\mu} = \frac{\sum_{i=1}^{n} x^{(i)}}{n}$.

Algorithm 5 S3UncFA

function TRAIN($\{(x^{(1)}, z^{(1)}), \ldots, (x^{(a)}, z^{(a)}), x^{(a+1)}, \ldots, x^{(n)}\}$,nMaxIterations,eps)
  $\theta^{(0)}$ = initializeParameters($\{(x^{(1)}, z^{(1)}), \ldots, (x^{(a)}, z^{(a)}), x^{(a+1)}, \ldots, x^{(n)}\}$)
  $l^{(0)}_{\text{RV\_Do}} = l_{\text{RV\_Do}}(\theta^{(0)})$ according to 2.17
  for t = 0:nMaxIterations **do**
    E step: Compute $E[Z^{(i)}]$, $E[Z^{(i)}Z^{(i)^\top}], i \in \{a+1, \ldots, n\}$ according to 2.18 and 2.19
    M Step: Compute $\theta^{(t+1)} = (\mu_z^{(t+1)}, \Sigma_z^{(t+1)}, \mu^{(t+1)}, \Lambda^{(t+1)}, \Psi^{(t+1)})$ according to 2.12, 2.13, 2.14, 2.15, 2.16
    $l^{(t+1)}_{\text{RV\_Do}} = l_{\text{RV\_Do}}(\theta^{(t+1)})$ according to 2.17
    if $\frac{\|\theta^{(t)} - \theta^{(t+1)}\|_2^2}{\|\theta^{(t)}\|_2^2} \leq$eps **or** $\frac{|l^{(t)}_{\text{RV_Do}} - l^{(t+1)}_{\text{RV_Do}}|}{|l^{(t)}_{\text{RV_Do}}|} \leq$eps **then**
      break
  return $\theta^{(t)}$
function TEST(x^*,($\hat{\mu}_z$,$\hat{\Sigma}_z$,$\hat{\Lambda}$,$\hat{\mu}$,$\hat{\Psi}$)) ▷ The same as in *S2UncFA*
  value = $\hat{\mu}_z + \hat{\Sigma}_z\hat{\Lambda}^\top(\hat{\Lambda}\hat{\Sigma}_z\hat{\Lambda}^\top + \hat{\Psi})^{-1}(x^* - \hat{\mu} - \hat{\Lambda}\hat{\mu}_z)$
  covarianceMatrix = $\hat{\Sigma}_z - \hat{\Sigma}_z\hat{\Lambda}^\top(\hat{\Lambda}\hat{\Sigma}_z\hat{\Lambda}^\top + \hat{\Psi})^{-1}\hat{\Lambda}\hat{\Sigma}_z^\top$
  return (value, covarianceMatrix)

2.7 S3FA - Simple-Semi-Supervised Factor Analysis

In the last section we derived *S3UncFA*. In this algorithm, Ψ must be a symmetric and positive definite matrix. If we impose Ψ to be **diagonal**, then we would be in the *FA* context and so the modified algorithm is called *S3FA*, which we briefly discuss below.

As one may notice, the derivation is almost the same except for $\frac{\partial l_{\text{RV\_Dc}}}{\partial \Psi}$. This derivative is very similar to the one in *S2FA*, so we only present the final result:

$$\boxed{\begin{aligned}\Rightarrow \hat{\Psi} = \text{diag}(\frac{1}{n}(&\sum_{i=1}^{a}(x^{(i)} - \hat{\mu} - \hat{\Lambda}z^{(i)})(x^{(i)} - \hat{\mu} - \hat{\Lambda}z^{(i)})^\top + \sum_{i=a+1}^{n}((x^{(i)} - \hat{\mu})(x^{(i)} - \hat{\mu})^\top - \\ &-(x^{(i)} - \hat{\mu})E[Z^{(i)}]^\top\Lambda^\top - \hat{\Lambda}E[Z^{(i)}](x^{(i)} - \hat{\mu})^\top + \hat{\Lambda}E[Z^{(i)}Z^{(i)^\top}]\hat{\Lambda}^\top)))\end{aligned}}$$

where the term in diag($\cdot$) is exactly $\hat{\Psi}$ in *S3UncFA*.

2.8 S3PPCA - Simple-Semi-Supervised Principal Component Analysis

A similar approach applies when Ψ **must be of the form** $\eta^2 I$, $\eta \in \mathbb{R}$, $\eta > 0$. The resulted algorithm is called *S3PPCA* and we briefly discuss it below.

As one may notice, the derivation is almost the same except for $\frac{\partial l_{\mathrm{RV\_Dc}}}{\partial \Psi}$. This derivative is very similar to the one in *S2PPCA*, so we only present the final result:

$$\boxed{\begin{aligned}\Rightarrow \hat{\eta}^2 = \frac{1}{D}\mathrm{Tr}(\frac{1}{n}(\sum_{i=1}^{a}(x^{(i)} - \hat{\mu} - \hat{\Lambda}z^{(i)})(x^{(i)} - \hat{\mu} - \hat{\Lambda}z^{(i)})^\top + \sum_{i=a+1}^{n}((x^{(i)} - \hat{\mu})(x^{(i)} - \hat{\mu})^\top - \\ -(x^{(i)} - \hat{\mu})E[Z^{(i)}]^\top\Lambda^\top - \hat{\Lambda}E[Z^{(i)}](x^{(i)} - \hat{\mu})^\top + \hat{\Lambda}E[Z^{(i)}{Z^{(i)}}^\top]\hat{\Lambda}^\top)))\end{aligned}}$$

where the term in $\mathrm{Tr}(\cdot)$ is exactly $\hat{\Psi}$ in *S3UncFA*.

2.9 MS3UncFA - Missing Simple-Semi-Supervised Unconstrained Factor Analysis

The *S3UncFA*, *S3FA*, *S3PPCA* algorithms were developed in order to handle the semi-supervised case of *S2UncFA*, *S2FA*, *S2PPCA*, i.e. for some instances the output part exists, for other instances the output part does not exist (i.e. is **missing data**). We propose a generalization of the algorithm which has its **roots in an algorithm that computes the Maximum Likelihood estimators for the parameters of a multivariate normal distribution (with missing data)**. The new algorithm handles **missing data in input**. It also handles **partially observed output**, i.e. for an instance the output part can exist, not exist, or exist only on some components (= a vector with missing values).

The ***first*** *step* in generalising is as follows: the formulas in the M step of *S3UncFA*, *S3FA*, *S3PPCA* can be rewritten only in terms of $E[Z^{(i)}]$ and $E[Z^{(i)}{Z^{(i)}}^\top]$ if we denote $E[Z^{(i)}] = z^{(i)}$, if $i \in \{1, \ldots, a\}$ and $E[Z^{(i)}{Z^{(i)}}^\top] = z^{(i)}{z^{(i)}}^\top$, if $i \in \{1, \ldots, a\}$.

The ***second*** *step* in generalising is as follows: because we admit missing values on input, we will also have to compute $E[X^{(i)}]$, $E[X^{(i)}{X^{(i)}}^\top]$, $E[X^{(i)}{Z^{(i)}}^\top]$.

$$E[X^{(i)}] = E\left[\begin{bmatrix} X_1^{(i)} \\ \vdots \\ X_D^{(i)} \end{bmatrix}\right] = \begin{bmatrix} E[X_1^{(i)}] \\ \vdots \\ E[X_D^{(i)}] \end{bmatrix}, \text{ where } E[X_j^{(i)}] = \begin{cases} x_j^{(i)} & \text{if } x_j^{(i)} \exists \text{ in the dataset} \\ \text{different} & \text{otherwise} \end{cases}$$

$$E[X^{(i)}{X^{(i)}}^\top] = E\left[\begin{bmatrix} X_1^{(i)}X_1^{(i)} & \ldots & X_1^{(i)}X_D^{(i)} \\ \vdots & \ddots & \vdots \\ X_D^{(i)}X_1^{(i)} & \ldots & X_D^{(i)}X_D^{(i)} \end{bmatrix}\right] = \begin{bmatrix} E[X_1^{(i)}X_1^{(i)}] & \ldots & E[X_1^{(i)}X_D^{(i)}] \\ \vdots & \ddots & \vdots \\ E[X_D^{(i)}X_1^{(i)}] & \ldots & E[X_D^{(i)}X_D^{(i)}] \end{bmatrix},$$

$$\text{where } E[X_j^{(i)}X_k^{(i)}] = \begin{cases} x_j^{(i)}x_k^{(i)} & \text{if } x_j^{(i)}, x_k^{(i)} \exists \text{ in the dataset} \\ x_j^{(i)}E[X_k^{(i)}] & \text{if only } x_j^{(i)} \exists \text{ in the dataset} \\ E[X_j^{(i)}]x_k^{(i)} & \text{if only } x_k^{(i)} \exists \text{ in the dataset} \\ \text{different} & \text{otherwise} \end{cases}$$

$$E[X^{(i)}Z^{(i)\top}] = E\left[\begin{bmatrix} X_1^{(i)}Z_1^{(i)} & \dots & X_1^{(i)}Z_d^{(i)} \\ \vdots & \ddots & \vdots \\ X_D^{(i)}Z_1^{(i)} & \dots & X_D^{(i)}Z_d^{(i)} \end{bmatrix}\right] = \begin{bmatrix} E[X_1^{(i)}Z_1^{(i)}] & \dots & E[X_1^{(i)}Z_d^{(i)}] \\ \vdots & \ddots & \vdots \\ E[X_D^{(i)}Z_1^{(i)}] & \dots & E[X_D^{(i)}Z_d^{(i)}] \end{bmatrix},$$

$$\text{where } E[X_j^{(i)}Z_k^{(i)}] = \begin{cases} x_j^{(i)}z_k^{(i)} & \text{if } x_j^{(i)}, z_k^{(i)} \exists \text{ in the dataset} \\ x_j^{(i)}E[Z_k^{(i)}] & \text{if only } x_j^{(i)} \exists \text{ in the dataset} \\ E[X_j^{(i)}]z_k^{(i)} & \text{if only } z_k^{(i)} \exists \text{ in the dataset} \\ \text{different} & \text{otherwise} \end{cases}$$

Also:

$$E[Z^{(i)}] = E\left[\begin{bmatrix} Z_1^{(i)} \\ \vdots \\ Z_d^{(i)} \end{bmatrix}\right] = \begin{bmatrix} E[Z_1^{(i)}] \\ \vdots \\ E[Z_d^{(i)}] \end{bmatrix}, \text{ where } E[Z_j^{(i)}] = \begin{cases} z_j^{(i)} & \text{if } x_j^{(i)} \exists \text{ in the dataset} \\ \text{different} & \text{otherwise} \end{cases}$$

$$E[Z^{(i)}Z^{(i)\top}] = E\left[\begin{bmatrix} Z_1^{(i)}Z_1^{(i)} & \dots & Z_1^{(i)}Z_d^{(i)} \\ \vdots & \ddots & \vdots \\ Z_d^{(i)}Z_1^{(i)} & \dots & Z_d^{(i)}Z_d^{(i)} \end{bmatrix}\right] = \begin{bmatrix} E[Z_1^{(i)}Z_1^{(i)}] & \dots & E[Z_1^{(i)}Z_d^{(i)}] \\ \vdots & \ddots & \vdots \\ E[Z_d^{(i)}Z_1^{(i)}] & \dots & E[Z_d^{(i)}Z_d^{(i)}] \end{bmatrix},$$

$$\text{where } E[Z_j^{(i)}Z_k^{(i)}] = \begin{cases} z_j^{(i)}z_k^{(i)} & \text{if } z_j^{(i)}, z_k^{(i)} \exists \text{ in the dataset} \\ z_j^{(i)}E[Z_k^{(i)}] & \text{if only } z_j^{(i)} \exists \text{ in the dataset} \\ E[Z_j^{(i)}]z_k^{(i)} & \text{if only } z_k^{(i)} \exists \text{ in the dataset} \\ \text{different} & \text{otherwise} \end{cases}$$

In order to understand the ***different*** part above, we do not formalize it further, but give the following example:

$D = 3, d = 2$

$$x^{(i)} = \begin{bmatrix} 1 \\ 2 \\ NA \end{bmatrix}, z^{(i)} = \begin{bmatrix} 4 \\ NA \end{bmatrix}.$$

$$\text{Let } y^{(i)} = \begin{bmatrix} x^{(i)} \\ z^{(i)} \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \\ NA \\ 4 \\ NA \end{bmatrix}.$$

We know from 2.6 that:

$$Y^{(i)} = \begin{bmatrix} X^{(i)} \\ Z^{(i)} \end{bmatrix} \sim \mathcal{N}\left(\underbrace{\begin{bmatrix} \mu^{(t)} + \Lambda^{(t)}\mu_z^{(t)} \\ \mu_z^{(t)} \end{bmatrix}}_{\text{not. } \mu_Y^{(t)}}, \underbrace{\begin{bmatrix} \Lambda^{(t)}\Sigma_z^{(t)}\Lambda^{(t)\top} & \Lambda^{(t)}\Sigma_z^{(t)} \\ (\Lambda^{(t)}\Sigma_z^{(t)})^\top & \Sigma_z^{(t)} \end{bmatrix}}_{\text{not. } \Sigma_Y^{(t)}} \right)$$

Obviously, we can compute $E[X^{(i)}]$, $E[Z^{(i)}]$, $E[X^{(i)}X^{(i)\top}]$, $E[Z^{(i)}Z^{(i)\top}]$, $E[X^{(i)}Z^{(i)\top}]$ by computing $E[Y^{(i)}]$, $E[Y^{(i)}Y^{(i)\top}]$.

$$E[Y^{(i)}] = \begin{bmatrix} 1 \\ 2 \\ E[Y_3^{(i)}] \\ 4 \\ E[Y_5^{(i)}] \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \\ E_{Y_3^{(i)}|Y_1^{(i)}=1, Y_2^{(i)}=2, Y_4^{(i)}=4, \theta^{(t)}}[Y_3^{(i)}] \\ 4 \\ E_{Y_5^{(i)}|Y_1^{(i)}=1, Y_2^{(i)}=2, Y_4^{(i)}=4, \theta^{(t)}}[Y_5^{(i)}] \end{bmatrix}$$

$$E[Y^{(i)}{Y^{(i)}}^\top] = \begin{bmatrix} 1 & 2 & \square & 4 & \square \\ 2 & 4 & \square & 8 & \square \\ \square & \square & \square & \square & \square \\ 4 & 8 & \square & 16 & \square \\ \square & \square & \square & \square & \square \end{bmatrix}$$

$E[Y_1^{(i)}Y_3^{(i)}] = \mathrm{Cov}[Y_1^{(i)}, Y_3^{(i)}] + E[Y_1^{(i)}]E[Y_3^{(i)}] \overset{Y_1^{(i)}=\text{constant RV}=1}{=} 0 + 1 \cdot E[Y_3^{(i)}] = E[Y_3^{(i)}]$

Similar for $E[Y_1^{(i)}Y_5^{(i)}]$, $E[Y_2^{(i)}Y_3^{(i)}]$, $E[Y_2^{(i)}Y_5^{(i)}]$, $E[Y_4^{(i)}Y_3^{(i)}]$, $E[Y_4^{(i)}Y_5^{(i)}]$, $E[Y_3^{(i)}Y_1^{(i)}]$, $E[Y_3^{(i)}Y_2^{(i)}]$, $E[Y_3^{(i)}Y_4^{(i)}]$, $E[Y_5^{(i)}Y_1^{(i)}]$, $E[Y_5^{(i)}Y_2^{(i)}]$, $E[Y_5^{(i)}Y_4^{(i)}]$.

$$\begin{aligned} E[Y_3^{(i)}Y_5^{(i)}] &= \mathrm{Cov}[Y_3^{(i)}, Y_5^{(i)}] + E[Y_3^{(i)}]E[Y_5^{(i)}] \\ &= \mathrm{Cov}_{Y_3^{(i)}, Y_5^{(i)} | Y_1^{(i)}=1, Y_2^{(i)}=2, Y_4^{(i)}=4, \theta^{(t)}}[Y_3^{(i)}, Y_5^{(i)}] + E[Y_3^{(i)}]E[Y_5^{(i)}] \end{aligned}$$

Similar for $E[Y_3^{(i)}Y_3^{(i)}]$, $E[Y_5^{(i)}Y_3^{(i)}]$, $E[Y_5^{(i)}Y_5^{(i)}]$.

We did not explicitly say how one could compute the following:
$E_{Y_3^{(i)} | Y_1^{(i)}=1, Y_2^{(i)}=2, Y_4^{(i)}=4, \theta^{(t)}}[Y_3^{(i)}]$,
$E_{Y_5^{(i)} | Y_1^{(i)}=1, Y_2^{(i)}=2, Y_4^{(i)}=4, \theta^{(t)}}[Y_5^{(i)}]$,
$\mathrm{Cov}_{Y_3^{(i)}, Y_5^{(i)} | Y_1^{(i)}=1, Y_2^{(i)}=2, Y_4^{(i)}=4, \theta^{(t)}}[Y_3^{(i)}, Y_5^{(i)}]$,
$\mathrm{Cov}_{Y_3^{(i)}, Y_3^{(i)} | Y_1^{(i)}=1, Y_2^{(i)}=2, Y_4^{(i)}=4, \theta^{(t)}}[Y_3^{(i)}, Y_3^{(i)}]$,
$\mathrm{Cov}_{Y_5^{(i)}, Y_3^{(i)} | Y_1^{(i)}=1, Y_2^{(i)}=2, Y_4^{(i)}=4, \theta^{(t)}}[Y_5^{(i)}, Y_3^{(i)}]$,
$\mathrm{Cov}_{Y_5^{(i)}, Y_5^{(i)} | Y_1^{(i)}=1, Y_2^{(i)}=2, Y_4^{(i)}=4, \theta^{(t)}}[Y_5^{(i)}, Y_5^{(i)}]$.

These can be computed using the fact that $Y^{(i)} \sim \mathcal{N}(\mu_Y^{(t)}, \Sigma_Y^{(t)})$ and the formulas [11, (352-354)] which refer to the fact that the conditional of the joint Gaussian is Gaussian.

In the end, the update formulas at the M step are (we unfolded the brackets when needed):

$$\boxed{\hat{\mu}_z = \frac{\sum_{i=1}^n E[Z^{(i)}]}{n}}$$

$$\boxed{\hat{\Sigma}_z = \frac{\sum_{i=1}^n (E[Z^{(i)}{Z^{(i)}}^\top] - E[Z^{(i)}]\hat{\mu}_z^\top - \hat{\mu}_z E[{Z^{(i)}}^\top] + \hat{\mu}_z\hat{\mu}_z^\top)}{n}}$$

$$\bar{x} = \frac{\sum_{i=1}^n E[X^{(i)}]}{n}$$

$$\boxed{\hat{\Lambda} = \left(n\bar{x}\hat{\mu}_z^\top - \sum_{i=1}^n E[X^{(i)}{Z^{(i)}}^\top]\right)\left(n\hat{\mu}_z\hat{\mu}_z^\top - \sum_{i=1}^n E[Z^{(i)}{Z^{(i)}}^\top]\right)^{-1}}$$

$$\boxed{\hat{\mu} = \bar{x} - \hat{\Lambda}\hat{\mu}_z}$$

$$\boxed{\begin{aligned} \hat{\Psi} = \frac{1}{n}\sum_{i=1}^n (&E[X^{(i)}{X^{(i)}}^\top] - E[X^{(i)}]\hat{\mu}^\top - \hat{\mu}E[X^{(i)}]^\top + \hat{\mu}\hat{\mu}^\top - E[X^{(i)}{Z^{(i)}}^\top]\hat{\Lambda}^\top + \\ &+ \hat{\mu}E[Z^{(i)}]^\top\hat{\Lambda}^\top - \hat{\Lambda}E[Z^{(i)}{X^{(i)}}^\top] + \hat{\Lambda}E[Z^{(i)}]^\top\hat{\mu}^\top + \hat{\Lambda}E[Z^{(i)}{Z^{(i)}}^\top]\hat{\Lambda}^\top) \end{aligned}}$$

The resulting algorithm **(training phase of *MS3UncFA*)** is the same as the training algorithm in *S3UncFA*, except that the E step and the M step are modified as described above. We replace the test phase by the following and add an imputation phase:

Algorithm 6 MS3UncFA - test and impute

function TEST($y^* = (x^*, z^*)$ partially known with MI the indexes such that if $i \in MI$, then $y_i^* = NA$, $(\hat{\mu}_z,\hat{\Sigma}_z,\hat{\Lambda},\hat{\mu},\hat{\Psi})$)
    Compute the conditional $E[Y^*_{MI}]$ and the conditional $\text{Cov}[Y^*_{MI}, Y^*_{MI}]$ in the same manner as in the training E step. (Y^*_{MI} = is a vector containing $Y_i^*, i \in MI$)
    Compute the conditional $E[Y^*]$ and $\text{Cov}[Y^*, Y^*]$. ▷ simple if step 2 is done
    value = $E[Y^*]$
    covarianceMatrix = $\text{Cov}[Y^*, Y^*]$
    return (value, covarianceMatrix)
function IMPUTE($y^* = (x^*, z^*)$ partially known, $(\hat{\mu}_z,\hat{\Sigma}_z,\hat{\Lambda},\hat{\mu},\hat{\Psi})$)
    (value, covarianceMatrix) = Test(y^*,$(\hat{\mu}_z,\hat{\Sigma}_z,\hat{\Lambda},\hat{\mu},\hat{\Psi})$)
    return value

Observation: The algorithm can be rewritten in **matrix form** using the same principles used when developing *S2UncFA* in matrix form.

2.10 MS3FA - Missing Simple-Semi-Supervised Factor Analysis

As in the case of *FA*, *S2FA* and *S3FA* we impose Ψ to be **diagonal** and we develop immediately a new algorithm: *MS3FA*.

As one may notice, the derivation is almost the same as in *MS3UncFA*, but the formula for $\hat{\Psi}$ differs:

$$\boxed{\begin{aligned}\hat{\Psi} = \text{diag}(\frac{1}{n}\sum_{i=1}^{n}(&E[X^{(i)}{X^{(i)}}^\top] - E[X^{(i)}]\hat{\mu}^\top - \hat{\mu}E[X^{(i)}]^\top + \hat{\mu}\hat{\mu}^\top - E[X^{(i)}{Z^{(i)}}^\top]\hat{\Lambda}^\top + \\ &+ \hat{\mu}E[Z^{(i)}]^\top\hat{\Lambda}^\top - \hat{\Lambda}E[Z^{(i)}{X^{(i)}}^\top] + \hat{\Lambda}E[Z^{(i)}]^\top\hat{\mu}^\top + \hat{\Lambda}E[Z^{(i)}{Z^{(i)}}^\top]\hat{\Lambda}^\top))\end{aligned}}$$

where the term in $\text{diag}(\cdot)$ is exactly $\hat{\Psi}$ in *MS3UncFA*.

2.11 MS3PPCA - Missing Simple-Semi-Supervised Principal Component Analysis

Almost the same approach applies when Ψ **must be of the form** $\eta^2 I, \eta \in \mathbb{R}, \eta > 0$. In this case, we call the resulted algorithm *MS3UncPPCA*, but the formula for $\hat{\Psi} = \hat{\eta}^2 I$ differs:

$$\boxed{\begin{aligned}\hat{\Psi} = \frac{1}{D}\text{Tr}(\frac{1}{n}\sum_{i=1}^{n}(&E[X^{(i)}{X^{(i)}}^\top] - E[X^{(i)}]\hat{\mu}^\top - \hat{\mu}E[X^{(i)}]^\top + \hat{\mu}\hat{\mu}^\top - E[X^{(i)}{Z^{(i)}}^\top]\hat{\Lambda}^\top + \\ &+ \hat{\mu}E[Z^{(i)}]^\top\hat{\Lambda}^\top - \hat{\Lambda}E[Z^{(i)}{X^{(i)}}^\top] + \hat{\Lambda}E[Z^{(i)}]^\top\hat{\mu}^\top + \hat{\Lambda}E[Z^{(i)}{Z^{(i)}}^\top]\hat{\Lambda}^\top))\end{aligned}}$$

where the term in $\text{Tr}(\cdot)$ is exactly $\hat{\Psi}$ in *MS3UncFA*.

2.12 Handling missing values in Linear Regression and in S2FA

In [21], it is presented the standard way in which *Naive Bayes* can handle missing data in input: *simply ignore attribute in instance where its value is missing.* We will adapt this idea to *S2FA*.

We will also interpret this in terms of the **log-likelihood of the (observed) data**. We write it down:

$$\begin{aligned}
l_{\text{RV_D}}(\theta) &= \ln(p_{\text{RV_D}|\theta}(D|\theta)) \\
&= \ln\left(p_{\text{RV_D}|\theta}\left(\{x_j^{(i)}|i \in \{1,\ldots,n\}, j \in \{1,\ldots,D\}, x_j^{(i)} \neq NA\} \cup \{z^{(i)}|i \in \{1,\ldots,n\}\}\right)\right)
\end{aligned}$$

Because Ψ is diagonal, we have that the attribute components of $x^{(i)}$ are independent, given $z^{(i)}$.

We return to the computation of $l_{\text{RV\_D}}(\theta)$:

$$\begin{aligned}
l_{\text{RV_D}}(\theta) &\overset{indep.;mult.rule}{=} \sum_{i=1}^{n} \ln(p_{Z^{(i)}|\theta}(z^{(i)}|\theta)) + \sum_{\substack{i=\overline{1,n} \\ j=\overline{1,D} \\ x_j^{(i)} \neq NA}} \ln \mathcal{N}(x_j^{(i)}|\mu_j + \Lambda_{j:}z^{(i)}, a_j^2) \\
&= -\frac{nd}{2}\ln(2\pi) - \frac{n}{2}\ln(\det(\Sigma_z)) - \frac{1}{2}\sum_{i=1}^{n}(z^{(i)} - \mu_z)^\top \Sigma_z^{-1}(z^{(i)} - \mu_z) + \\
&\quad + \sum_{\substack{i=\overline{1,n} \\ j=\overline{1,D} \\ x_j^{(i)} \neq NA}} \left(-\frac{1}{2}\ln(2\pi) - \ln a_j - \frac{1}{2}\left(\frac{x_j^{(i)} - \mu_j - \Lambda_{j:}z^{(i)}}{a_j}\right)^2\right)
\end{aligned}$$

One can notice that $\hat{\mu}_z$ and $\hat{\Sigma}_z$ remain the same as in *S2FA*.

$$\begin{aligned}
\frac{\partial l_{\text{RV_D}}}{\partial \mu_l} &= \sum_{\substack{i=\overline{1,n} \\ x_l^{(i)} \neq NA}} \left(-\frac{1}{2a_l^2}2(x_l^{(i)} - \mu_l - \Lambda_{l:}z^{(i)})(-1)\right) \\
&= \sum_{\substack{i=\overline{1,n} \\ x_l^{(i)} \neq NA}} \left(\frac{1}{a_l^2}(x_l^{(i)} - \mu_l - \Lambda_{l:}z^{(i)})\right)
\end{aligned}$$

$$\frac{\partial l_{\text{RV\_D}}}{\partial \mu_l} = 0 \Rightarrow \mu_l = \frac{\sum_{i=\overline{1,n}, x_l^{(i)} \neq NA}(x_l^{(i)} - \Lambda_{l:}z^{(i)})}{|\{i|i = \overline{1,n}, x_l^{(i)} \neq NA\}|}, \forall l = \overline{1,D}$$

Since $x_j^{(i)} - \mu_j - \Lambda_{j:}z^{(i)} \in \mathbb{R}$, we have:
$(x_j^{(i)} - \mu_j - \Lambda_{j:}z^{(i)})^2 = (x_j^{(i)} - \mu_j - \Lambda_{j:}z^{(i)})^\top I(x_j^{(i)} - \mu_j - \Lambda_{j:}z^{(i)})$.

$$\begin{aligned}
\frac{\partial l_{\text{RV_D}}}{\partial \Lambda_{l:}} &\overset{[11,(88)]}{=} \sum_{\substack{i=\overline{1,n} \\ x_l^{(i)} \neq NA}} \left(-\frac{1}{2a_l^2}(-2I(x_l^{(i)} - \mu_l - \Lambda_{l:}z^{(i)}){z^{(i)}}^\top)\right) \\
&= \sum_{\substack{i=\overline{1,n} \\ x_l^{(i)} \neq NA}} \left(\frac{1}{a_l^2}(x_l^{(i)} - \mu_l - \Lambda_{l:}z^{(i)}){z^{(i)}}^\top\right)
\end{aligned}$$

$$\frac{\partial l_{\text{RV\_D}}}{\partial \Lambda_{l:}} = 0 \Rightarrow \sum_{\substack{i=\overline{1,n} \\ x_l^{(i)} \neq NA}} \left(\frac{1}{a_l^2}(x_l^{(i)} - \mu_l - \Lambda_{l:}z^{(i)}){z^{(i)}}^\top\right) = 0$$

Solving the system in μ_l and $\Lambda_{l:}$ in a similar manner as we did in *S2UncFA* we obtain:

$$\boxed{\begin{aligned}\hat{\Lambda}_{l:} = &\left(n_l \frac{\sum_{i=\overline{1,n}}^{x_l^{(i)} \neq NA} x_l^{(i)}}{n_l} \left(\frac{\sum_{i=\overline{1,n}}^{x_l^{(i)} \neq NA} z^{(i)}}{n_l} \right)^\top - \sum_{i=\overline{1,n}}^{x_l^{(i)} \neq NA} x_l^{(i)} z^{(i)\top} \right) \\ &\left(n_l \left(\frac{\sum_{i=\overline{1,n}}^{x_l^{(i)} \neq NA} z^{(i)}}{n_l} \right) \left(\frac{\sum_{i=\overline{1,n}}^{x_l^{(i)} \neq NA} z^{(i)}}{n_l} \right)^\top - \sum_{i=\overline{1,n}}^{x_l^{(i)} \neq NA} z^{(i)} z^{(i)\top} \right)^{-1}\end{aligned}}$$

$$\boxed{\hat{\mu}_l = \frac{\sum_{i=\overline{1,n}}^{x_l^{(i)} \neq NA} x_l^{(i)}}{n_l} - \hat{\Lambda}_{l:} \frac{\sum_{i=\overline{1,n}}^{x_l^{(i)} \neq NA} z^{(i)}}{n_l}}$$

where $n_l = |\{i | i = \overline{1,n}, x_l^{(i)} \neq NA\}|$.

$$\frac{l_{\text{RV\_D}}}{a_l} = \sum_{\substack{i=\overline{1,n} \\ x_l^{(i)} \neq NA}} \left(-\frac{1}{a_l} - \frac{1}{2}(x_l^{(i)} - \mu_l - \Lambda_{l:} z^{(i)})^2 (-2) \frac{1}{a_l^3} \right)$$

$$\frac{l_{\text{RV\_D}}}{a_l} = 0 \Rightarrow \boxed{\hat{a}_l^2 = \frac{\sum_{i=\overline{1,n}}^{x_l^{(i)} \neq NA} (x_l^{(i)} - \hat{\mu}_l - \hat{\Lambda}_{l:} z^{(i)})^2}{|\{i | i = \overline{1,n}, x_l^{(i)} \neq NA\}|}}$$

As a result, we found **closed-form solutions** for the parameters. This is not usually the case, as we saw in *MS3UncFA*, *MS3FA*, *MS3PPCA*. Actually, **when missing data is only in input, the above algorithm is equivalent to *MS3FA*** (we just maximize $l_{\text{RV\_Do}}$ in different ways: using closed-form solutions or iteratively using the *EM* algorithm).

If we adapt the above algorithm to the ***PPCA*** case (i.e.: $\Psi = \eta^2 I, \eta \in \mathbb{R}, \eta > 0$):

$$\boxed{\hat{\eta}^2 = \frac{\sum_{i=\overline{1,n}, j=\overline{1,D}, x_j^{(i)} \neq NA} (x_j^{(i)} - \hat{\mu}_j - \hat{\Lambda}_{j:} z^{(i)})^2}{|\{(i,j) | i = \overline{1,n}, j = \overline{1,D}, x_j^{(i)} \neq NA\}|}}$$

we obtain an **algorithm equivalent to *MS3PPCA* when missing data is only in input**.

After the training phase, if we were to **impute** missing data in input we would return the mean of $X^*|z*$, if z^* were provided, or the mean of X^* (i.e. $\hat{\mu} + \hat{\Lambda}\hat{\mu}_z$), otherwise.

We briefly **compare how the above model handles missing data and how *Linear regression* (*LR*) can handle it**. If Ψ is diagonal, we can learn the parameters with closed-form solutions without the need of the *EM* algorithm. In the *LR* model, one cannot immediately handle missing data. First, we need a distribution over the input X. Second, we need that the distribution $X|Y$ have diagonal covariance matrix, but, after applying the Bayes' rule, we do not usually obtain a diagonal covariance matrix. As a result, we must apply the *EM* algorithm to handle missing data which is more difficult than the solution above. Hence, ***S2FA* and *S2PPCA* can handle naturally missing data in input unlike the *LR* model which requires the use of the *EM* algorithm**.

Chapter 3

Other extensions

As we have seen so far, the initial algorithm *S2UncFA* has been extended as follows:

- into *S2FA*, *S2PPCA* by modifying the **structure of** Ψ
- into *S3UncFA*, *S3FA*, *S3PPCA* by considering the **semi-supervised** scenario
- into *MS3UncFA*, *MS3FA*, *MS3PPCA* by considering the possibility of having **missing data in input and output**

We thought of other extensions of this algorithm, but some of them are **easier to apply** than we saw in the extensions above and some of them **cannot be applied**. We will discuss them below.

When we noticed the **strong equivalence** between the *Linear Regression* (*LR*) model and *S2UncFA*, we thought that there could also be a relationship between *LR* and *S2FA*/*S2PPCA*. As a result, we started to integrate some extensions of the *LR* model in *S2FA*/*S2PPCA* and hoped that we ended up with some new insights.

3.1 Weighted extension

The first extension of the *Linear Regression* model we thought of was *Weighted Linear Regression.* In this model, each instance has its own noise term, i.e. each noise term has its own variance (weight). The **disadvantage** here is that you cannot learn the weights from data and so, because we do not have this feature in *S2FA* and *S2PPCA* we **did not proceed with this idea**.

3.2 Ridge extension

The second extension of the *Linear Regression* model we thought of was *Ridge Regression* (L2 regularization), as a possibility to be equivalent to *S2PPCA* because they both have a similar parameter: λ_{ridge} and η. This idea has been immediately dismissed because λ_{ridge} is a **hyper-parameter** which is empirically set and η is a **parameter** which is learnt. Despite this, **we still introduced this feature in the training phase**, because, what we actually do there is fitting a *Linear Regression* [recall the weak equivalence to the *Linear Regression*] and introducing L2 regularization in this case is very simple: just replace the formula for $\hat{\Lambda}$ in *S2UncFA* with: $\hat{\Lambda} = \left(n\bar{x}\bar{z}^\top - \sum_{i=1}^n x^{(i)}z^{(i)^\top}\right)\left(n\bar{z}\bar{z}^\top - \sum_{i=1}^n z^{(i)}z^{(i)^\top} - \lambda_{\text{ridge}}I\right)^{-1}$

Notice that we used regularization only **on the intercept term (not on bias)**. We can introduce this feature in all of the algorithms developed so far. Also, notice that now the inverse is well defined (anytime when λ_{ridge} is not 0) and so this can be used as a solution when the matrix whose inverse we want to compute is singular.

3.3 Kernel extension

3.3.1 Linear-Regression-based

A third extension, not necessarily linked to *Linear Regression*, is about kernelization. One observation is that *Linear Regression* cannot be kernelized (but *Ridge Regression* can). So a first step is to see if we could kernelize our *S2UncFA*. Obviously, this is not the case because *S2UncFA* is (strongly) equivalent to ***Linear Regression*** **(and we cannot kernelize it)**. When moving to *S2FA*, *S2PPCA*, we have a problem: at the testing phase, the terms in the formula cannot be reduced, the **expression is quite large** and perhaps one cannot arrive at something simpler, which can be kernelized. As a result, the kernelization in this manner is not possible (yet). An important observation is that if we introduce **Ridge regularization** in the training phase, we also end up in the testing phase with **large expressions** whose terms cannot be reduced even in the case of *S2UncFA*.

3.3.2 GPLVM-based

Another idea to kernelize our models is to use *GPLVM* (*Gaussian Process Latent Variable Model*), which is used when one wants to kernelize *PPCA*. As one may notice if we kernelized the input or the output the Λ term would be of infinite dimensionality when using an infinite-dimensional kernel. *GPLVM* handles this problem. We expose the workflow of *GPLVM* which is relevant for us: it starts from *PPCA* (the number of parameters in $\Psi = \eta^2 I$ is not infinite even if the kernel is infinite-dimensional); it starts from the unsupervised case (i.e. *PPCA*); it starts from the log-likelihood of observed data (i.e. just the input, X); it removes Λ by integrating it out (so now we do not have parameters with infinite dimensionality); the output (Z) is now interpreted as parameters for the objective function; the objective function can be kernelized in Z; it uses the gradient method to optimize the objective function, by computing the gradients with respect to Z, so the idea of the *EM* algorithm disappears; **the output of the algorithm is represented by the value of Z where the function is optimized and not (!) the parameters (Λ, Ψ etc.); the supervised adaptation does not make sense because the objective function would become a constant**. Studying this workflow (the last 2 observations are critical) **we cannot kernelize *S2FA* or *S2PPCA* in this manner**, because we need the parameters in the test phase, and because we are in the supervised case.

3.4 Discrete data extension

A fourth extension is described next.

All the algorithms we developed use only **continuous** random variables via **normal** distribution. We propose models that can handle **discrete** data via discrete random variables and discuss the **problems that arise**.

In order to introduce discrete random variables we replace somewhere the normal distribution with the **Bernoulli** distribution (with or without the ***sigmoid*** function), as this is done when moving from the *Linear Regression* model to the *Logistic Regression* model.

We can posit a Bernoulli distribution on:

- **output**: it would not make sense because we would end up with a mixture model, which is not what we want (we want to remain in the *Factor Analysis* framework)
- **output after applying the Bayes' rule** (in the testing phase): it is actually impossible to do this because $Z^*|x^*$ is normally distributed and you cannot say it is a Bernoulli random variable. What you can do is to apply a heuristic: compute sigmoid(predictedValue) and predict 1, if it is > 0.5 and 0, otherwise
- **input**, which we discuss below.

Recall that sigmoid(z) $= \sigma(z) = \frac{1}{1+e^{-z}}$. Let us assume that there are D input attributes and all of them are binary.

Let $X_j^{(i)}|z^{(i)} \sim$ Bernoulli$(\sigma(\mu_j + \Lambda_{j:}z^{(i)}))$, $\forall j = \overline{1, D}$, $X_j^{(i)}, X_k^{(i)}$ - independent, given the output, $\forall i \neq k$.

Let us assume that we are in the supervised case.

$$l_{\text{RV\_D}}(\theta) = \sum_{i=1}^{n} \ln \mathcal{N}(z^{(i)}|\mu_z, \Sigma_z) + \sum_{i=1}^{n}\sum_{j=1}^{D} \left((1 - x_j^{(i)}) \ln(1 - \sigma(\mu_j + \Lambda_{j:}z^{(i)})) + x_j^{(i)} \ln(\sigma(\mu_j + \Lambda_{j:}z^{(i)}))\right)$$

$$\ln(\sigma(\mu_j + \Lambda_{j:}z^{(i)})) = -\ln(1 + e^{-(\mu_j+\Lambda_{j:}z^{(i)})})$$

$$\begin{aligned}
\ln(1 - \sigma(\mu_j + \Lambda_{j:}z^{(i)})) &= \ln\left(1 - \frac{1}{1 + e^{-(\mu_j+\Lambda_{j:}z^{(i)})}}\right) \\
&= \ln \frac{e^{-(\mu_j+\Lambda_{j:}z^{(i)})}}{1 + e^{-(\mu_j+\Lambda_{j:}z^{(i)})}} \\
&= -(\mu_j + \Lambda_{j:}z^{(i)}) - \ln\left(1 + e^{-(\mu_j+\Lambda_{j:}z^{(i)})}\right)
\end{aligned}$$

We return to the computation of $l_{\text{RV\_D}}$.

$$\begin{aligned}
l_{\text{RV_D}}(\theta) = \sum_{i=1}^{n} &\ln \mathcal{N}(z^{(i)}|\mu_z, \Sigma_z) - \\
&- \sum_{i=1}^{n}\sum_{j=1}^{D} \left((1 - x_j^{(i)}) \left(\mu_j + \Lambda_{j:}z^{(i)} + \ln\left(1 + e^{-(\mu_j+\Lambda_{j:}z^{(i)})}\right)\right) + x_j^{(i)} \ln\left(1 + e^{-(\mu_j+\Lambda_{j:}z^{(i)})}\right)\right)
\end{aligned}$$

As one may notice, the parameters can be learnt using, for example, the **gradient ascent** method. We actually perform D independent *Logistic Regression* tasks. At the testing phase, we encounter a problem: we cannot compute $Z|x*$:

$$p(z|x^*) \overset{\text{Bayes' rule}}{=} \frac{p(x^*|z)p(z)}{\int_z p(x^*|z)p(z)dz}$$

The integral does not have a closed-form expression as it did when we worked only with the normal distribution. There are approaches that handle this problem. They involve **Variational Inference** [5, chap.21]. A concrete solution is in [22] which is referred to as *PCA for categorical data* [5, ex.21.9], or *Variational EM for binary FA with sigmoid link* [5, 12.4]. It gives a solution for the unsupervised case.

Speaking of the **unsupervised** case, there are other **problems** which are found **earlier** in the derivation of the algorithm. When we try to apply the expectation operator over $l_{\text{RV\_D}}(\theta)$ we notice that we cannot distribute it to $Z^{(i)}$, because ln and e are not linear operators, so we **cannot swap the expectation operator with $\boldsymbol{ln}$ or $\boldsymbol{e}$**. The same applies to the **semi-supervised** case.

Observation: If we had wanted to work with **discrete** data, but **not only binary data**, we could have used **Categorical** random variables and the ***softmax*** function instead of Bernoulli and *sigmoid*.

We **conclude** that there is **no obvious link between *S2FA/S2PPCA* and *Linear Regression*, we do not find a reason to adapt the model to the Weighted case, we can add a Ridge regularization in the training phase, we cannot kernelize the model and we can use discrete data, but this adaptation is not straightforward**.

Chapter 4

The R Package: s2fa

We created an R Package that includes the algorithms we have discussed so far and also those for *PPCA* and *FA*. The package can be found at: `https://github.com/aciobanusebi/s2fa`. The exported functions are:

- **faInit**: an initialization procedure for *EM/FA* or *EM/PPCA*
- **faFit**: fit the parameters via *EM/FA* or *EM/PPCA*
- **faPredict**: predict new values, given the parameters of a trained *FA/PPCA* model
- **faPlot**: plot the hyperplane learnt by *FA/PPCA* (works only if the input is 1- or 2-dimensional and the output is 1-dimensional)
- **s2faFit**: fit the parameters via the algorithm *S2UncFA*/*S2FA*/*S2PPCA*
- **s2faPredict**: predict new values, given the parameters of a trained *S2UncFA*/*S2FA*/*S2PPCA* model
- **s2faPlot**: plot the hyperplane learnt by *S2UncFA*/*S2FA*/*S2PPCA* (works only if the input is 1- or 2-dimensional and the output is 1-dimensional)
- **s3faInit**: an initialization procedure for the algorithm *S3UncFA*/*S3FA*/*S3PPCA*
- **s3faFit**: fit the parameters via the algorithm *S3UncFA*/*S3FA*/*S3PPCA*
- **s3faPredict**: predict new values, given the parameters of a trained *S3UncFA*/*S3FA*/*S3PPCA* model
- **s3faPlot**: plot the hyperplane learnt by *S3UncFA*/*S3FA*/*S3PPCA* (works only if the input is 1- or 2-dimensional and the output is 1-dimensional)
- **mS3faInit**: an initialization procedure for the algorithm *MS3UncFA*/*MS3FA*/*MS3PPCA*
- **mS3faFit**: fit the parameters via the algorithm *MS3UncFA*/*MS3FA*/*MS3PPCA*
- **mS3faPredict**: predict new values, given the parameters of a trained *MS3UncFA*/*MS3FA*/*MS3PPCA* model
- **mS3faImpute**: impute missing data via *MS3UncFA*/*MS3FA*/*MS3PPCA*

For more details about their use and examples, please consult the man pages: after installing the package write in the console "?s2fa::[function_name]", where [function_name] is one of the names above.

We discuss some **aspects regarding the implementation**:

- the **matrix versions/forms** of the algorithms for *FA*, *PPCA*, *S2UncFA*, *S2FA*, *S2PPCA*, *S3UncFA*, *S3FA*, *S3PPCA* were used (i.e. we used vectorised code) because they are faster than the non-matrix counterparts

- the **non-matrix versions/forms** of the algorithms for *MS3UncFA*, *MS3FA*, *MS3PPCA* were used

- in the algorithms we had to compute **two inverses**:
  - when updating/computing $\hat{\Lambda}$ there is an inverse: we noticed that matrix is **symmetric and positive definite** and so, we computed the inverse using the **Cholesky decomposition**: 'chol2inv(chol(·))', due to time efficiency and numerical stability
  - $(\hat{\Lambda}\hat{\Sigma}_z\hat{\Lambda} + \hat{\Psi})^{-1}$: if we talk about *S2UncFA/S2FA/S2PPCA* (i.e. supervised case) and λ_{ridge} is set to **0**, then we solve the system via the **QR decomposition** (i.e. using 'qr.solve(·)'; we do this because of the weak equivalence to *Linear Regression*, which is usually fitted via QR due to its numerical stability), otherwise we simply use the 'solve(·)' method

- in the *FA* cases of the algorithms we **did not store** $\hat{\Psi}$, but only the diagonal; when adding $\hat{\Psi}$ to a matrix **we updated only its diagonal**; **we computed** $\hat{\Psi}$ **with the formula of the type 'diag(·)'** because, in general, this is faster than using a for loop

- in the *PPCA* cases of the algorithms we **did not store** $\hat{\Psi}$, but only $\hat{\eta}$; when adding $\hat{\Psi}$ to a matrix **we updated only its diagonal**; **we computed** $\hat{\Psi}$ **with the formula of the type '**$\frac{1}{D} \cdot Tr(\cdot)$**'** because, in general, this is faster than using a for loop

- we have an option such that our *EM* algorithms run in the '**turboEM**' framework [23, 24] which provides **acceleration schemes** for the *EM* algorithm (and not only). Those acceleration schemes target fixed-point algorithms. Briefly, a **fixed-point algorithm** is an iterative algorithm that tries to find the parameters of a function (f) such that $f(parameters) == parameters$. Firstly, the parameters are initialized. Then, at each iteration the old parameters are updated as new parameters by applying the function f on the old parameters: $parameters := f(parameters)$. One may notice that we could frame any *EM* algorithm like this.

Chapter 5

Some Experiments

As one may notice, the resulted algorithms may be used in a variety of Machine Learning tasks/contexts: **single-output regression, multi-output regression, single-output semi-supervised regression, multi-output semi-supervised regression, learning with missing data, imputing continuous missing data (i.e. matrix completion, image inpainting etc.), generative algorithms, *EM* algorithms** etc. Each task can be applied in **different fields** (data science, image processing etc.) with different datasets. So, there would be a lot of experiments to make. We do not claim that the experiments below are complete. In fact, they are far from that. This is the reason why we entitled this section "**Some Experiments**".

5.1 Plots of the learnt hyperplanes

Data: House data; **Source**: 'Long-Kogen Realty, Chicago, USA'; 13 input features, 1 output feature (price); 26 instances

Note 1: On each plot some points are plotted: in *FA/PPCA* - input and the output column which is not shown to the algorithm; in *S2UncFA/S2FA/S2PPCA* - input and output; in *S3UncFA*, *S3FA*, *S3PPCA* - supervised input and supervised output

Note 2: On each plot, the resulted hyperplane (in red) is plotted and also the one resulted from fitting a *Linear Regression* on the plotted points (in blue).

Note 3: In the *FA/PPCA* plots with uni-dimensional input, the resulted hyperplane is not visible; it is below the *Linear Regression* hyperplane, because z in *FA/PPCA* is around 0 (since $z \sim \mathcal{N}(0, 1)$). This is visible in the plots with 2-dimensional input, where z in *FA/PPCA* is around $\begin{bmatrix} 0 \\ 0 \end{bmatrix}$ (since $z \sim \mathcal{N}(0, I)$).

Note 4: If the input is uni-dimensional, then the plots in a series are all the same.

Note 5: In the *S2UncFA/S2FA/S2PPCA* plots with uni-dimensional input, the resulted hyperplane and the *Linear Regression* hyperplane overlap.

Note 6: On this dataset, in the *S3UncFA/S3FA/S3PPCA* plots with uni-dimensional input, the resulted hyperplane and the *Linear Regression* hyperplane overlap.

Note 7: In the *S2UncFA* plot, the resulted hyperplane and the *Linear Regression* hyperplane overlap.

See Figures 1 - 11.

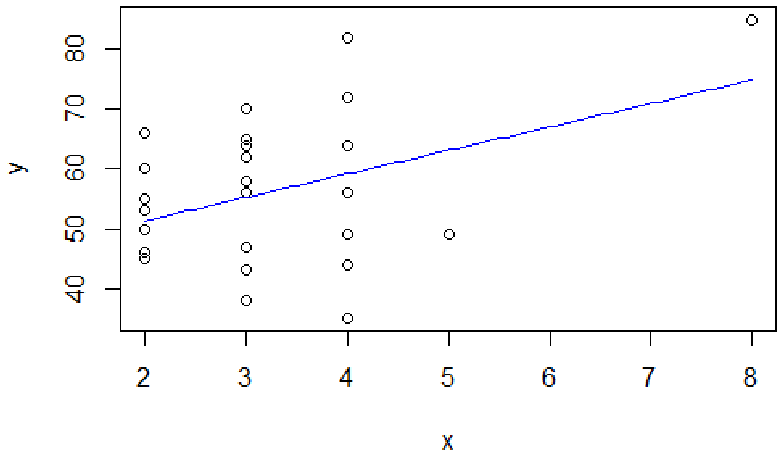

Figure 5.1: Hyperplanes. FA, PPCA - 1D: the second column - input; dimension of latent space = 1

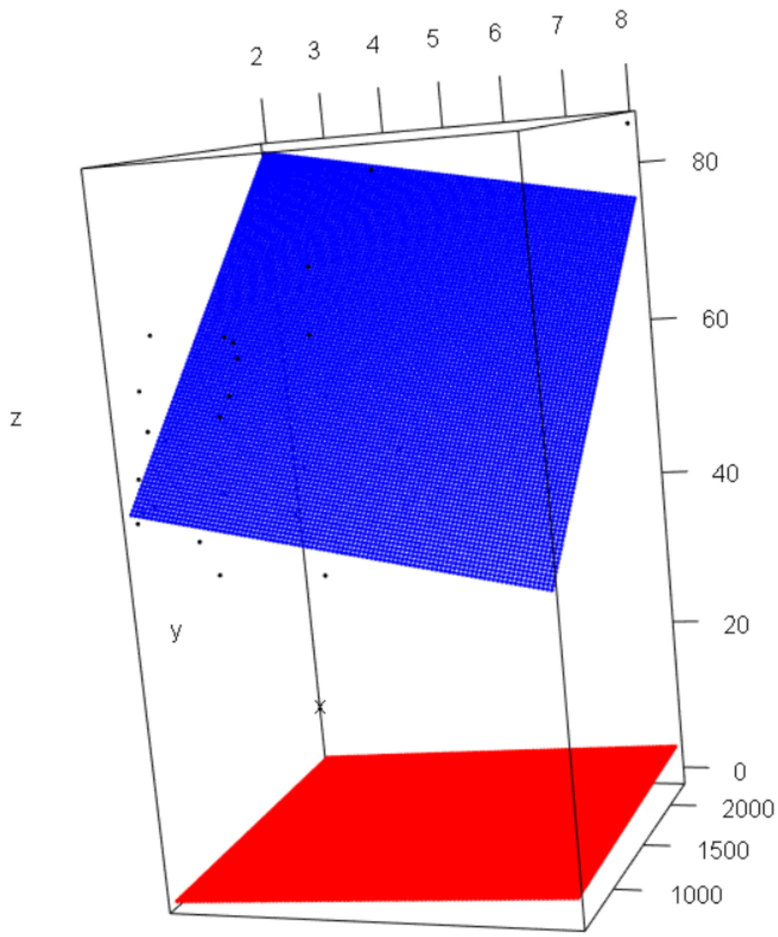

Figure 5.2: Hyperplanes. FA - 2D: the second and third columns - input; dimension of latent space = 1

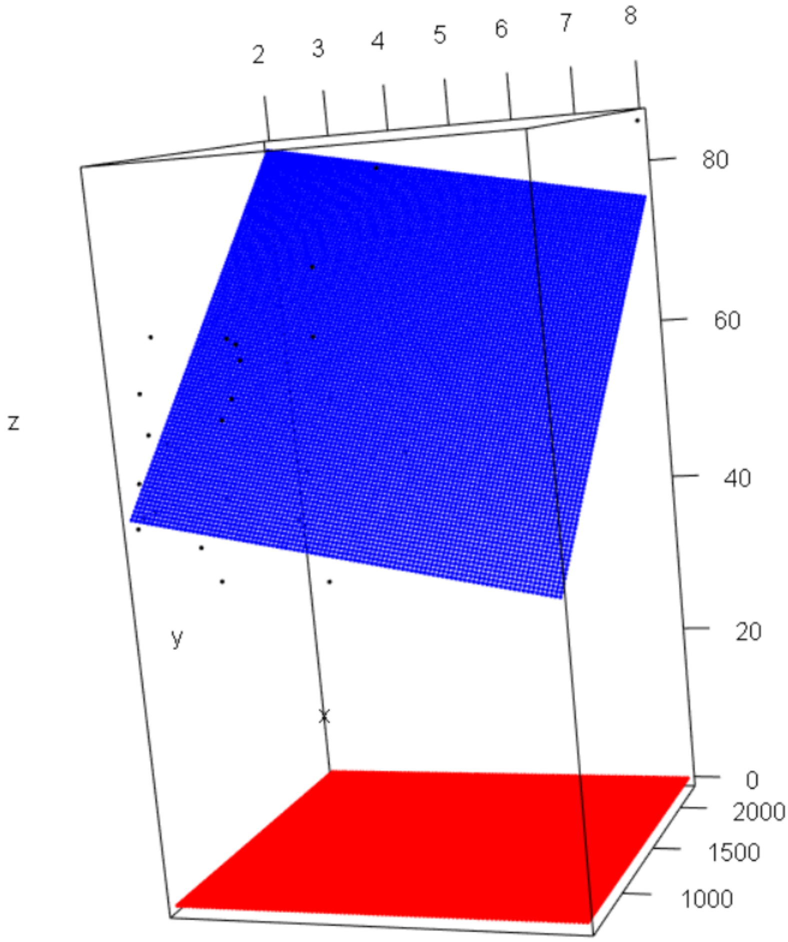

Figure 5.3:
Hyperplanes. PPCA - 2D: the second and third columns - input; dimension of latent space = 1

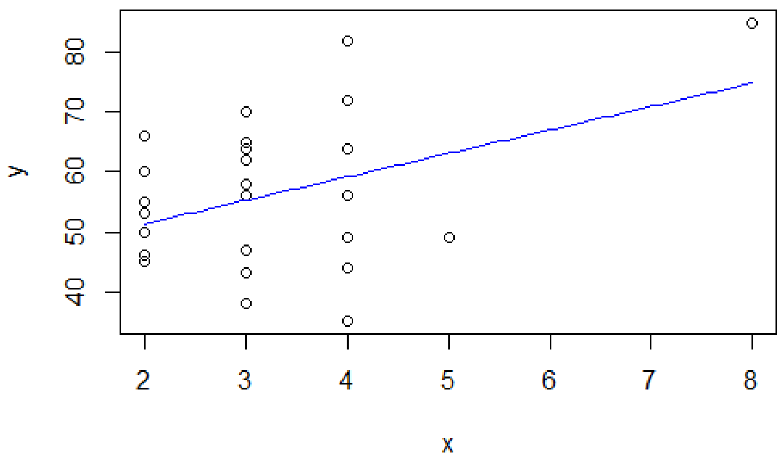

Figure 5.4:
Hyperplanes. S2UncFA, S2FA, S2PPCA - 1D: the second column - input; the first column - output

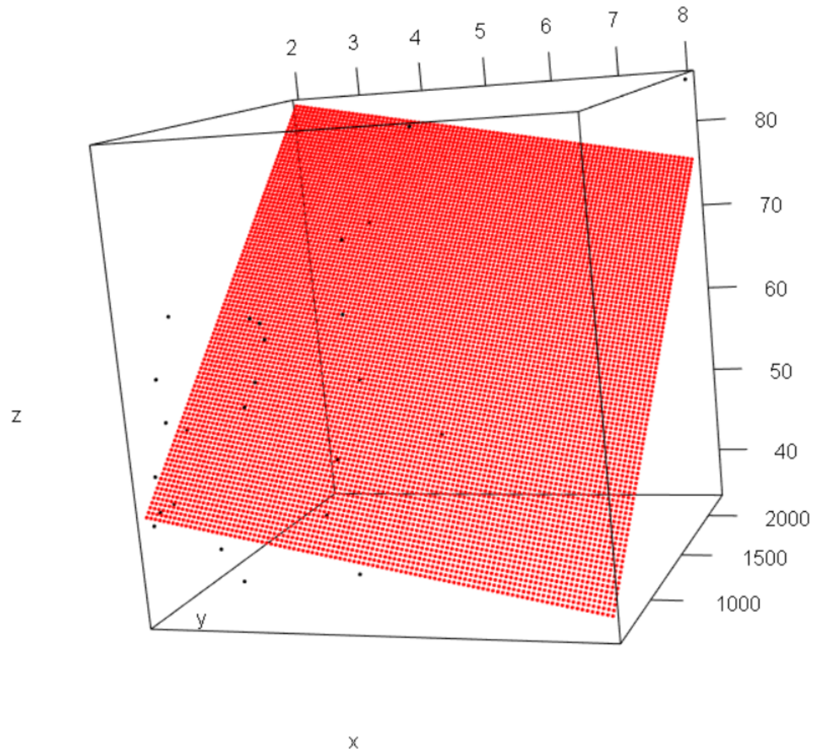

Figure 5.5: Hyperplanes. S2UncFA 2D: the second and third columns - input; the first column - output

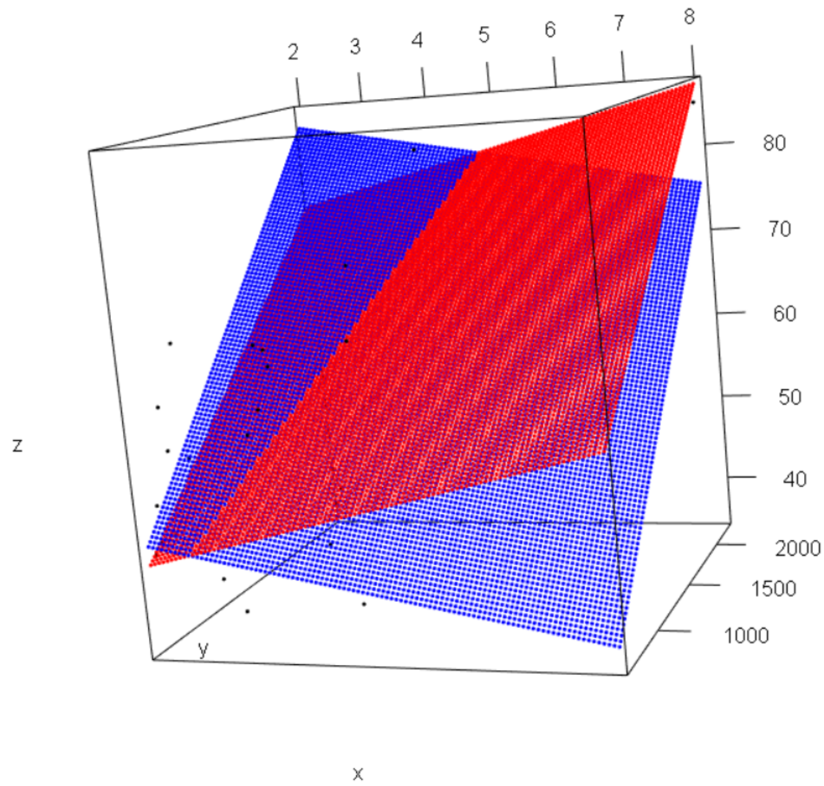

Figure 5.6: Hyperplanes. S2FA - 2D: the second and third columns - input; the first column - output

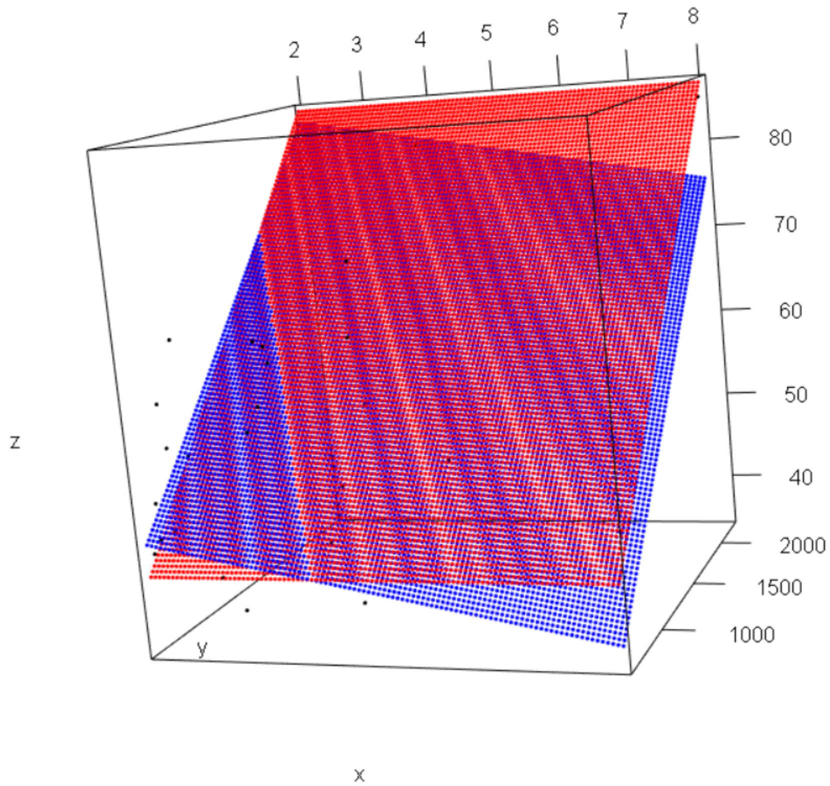

Figure 5.7: Hyperplanes. S2PPCA - 2D: the second and third columns - input; the first column - output

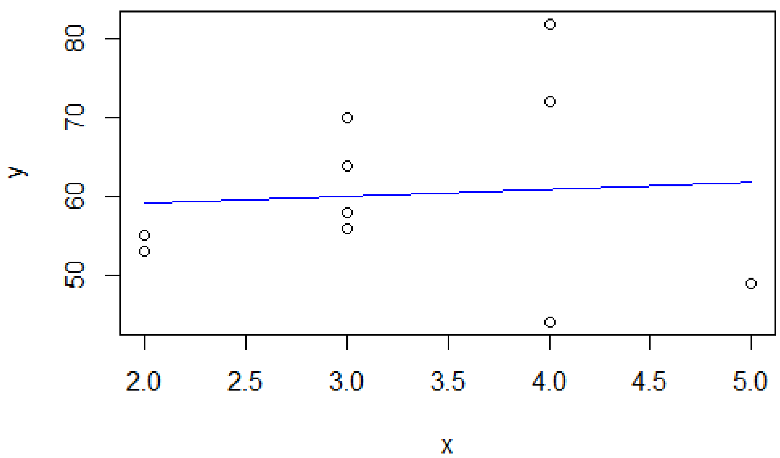

Figure 5.8: Hyperplanes. S3UncFA, S3FA, S3PPCA - 1D: only the first 20 rows; the first column - output; the second column - input; first ten rows - supervised; rows 11-20 - unsupervised

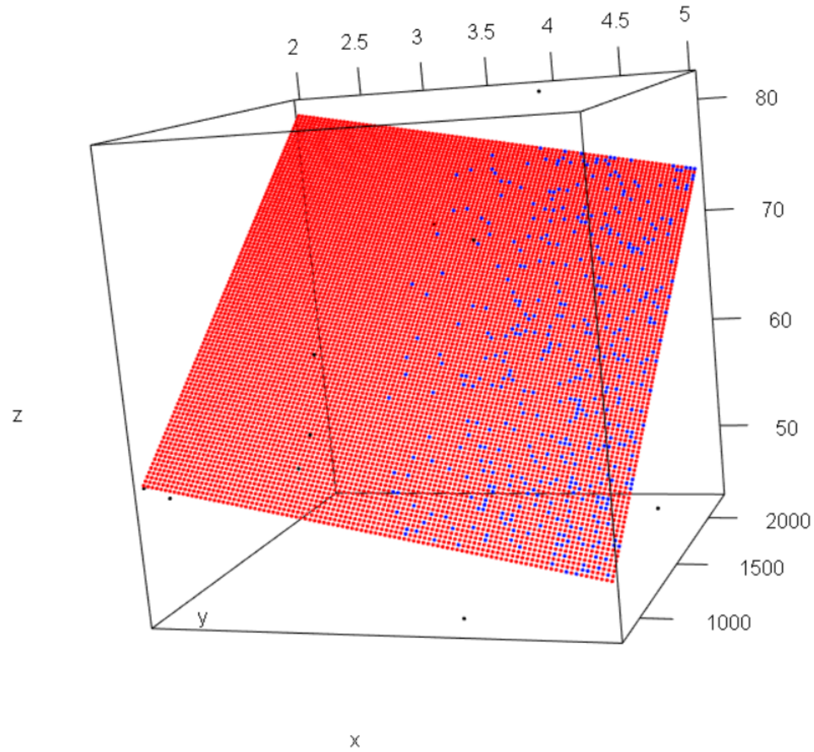

Figure 5.9: Hyperplanes. S3UncFA - 2D: only the first 20 rows; the first column - output; the second and third columns - input; first ten rows - supervised; rows 11-20 - unsupervised

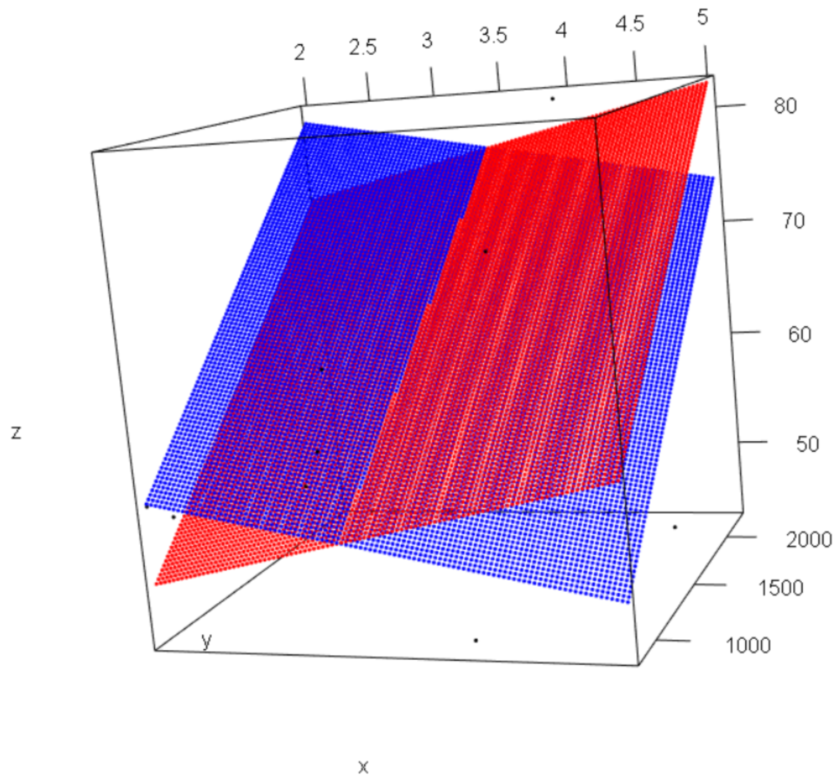

Figure 5.10: Hyperplanes. S3FA - 2D: only the first 20 rows; the first column - output; the second and third columns - input; first ten rows - supervised; rows 11-20 - unsupervised

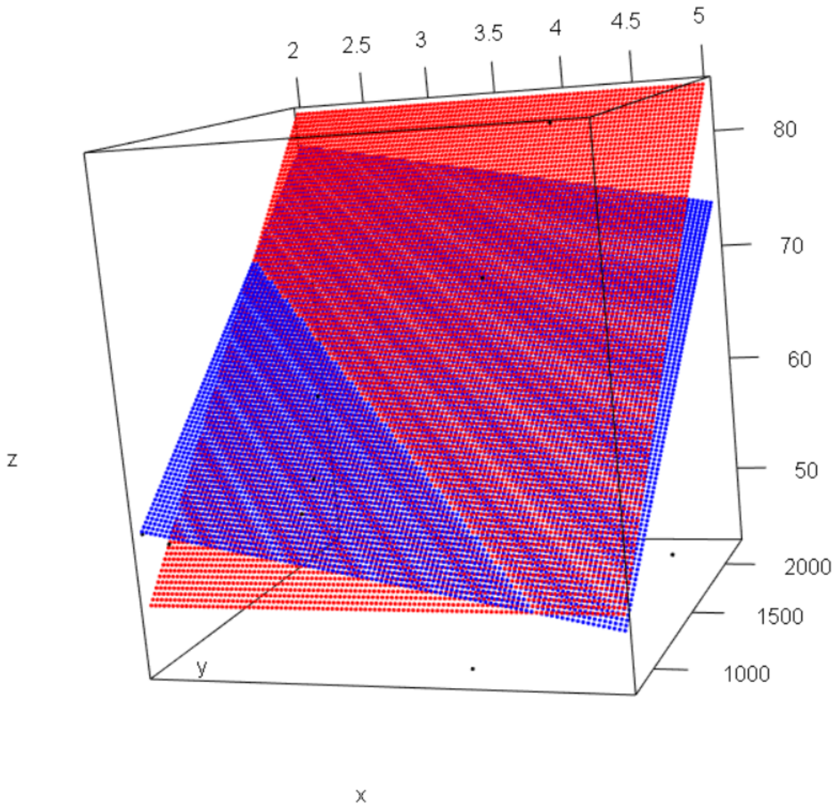

Figure 5.11: Hyperplanes. S3PPCA - 2D: only the first 20 rows; the first column - output; the second and third columns - input; first ten rows - supervised; rows 11-20 - unsupervised

5.2 Plots of the log-likelihood function for the EM algorithms

Data: House data; Source: 'Long-Kogen Realty, Chicago, USA'; 13 input features, 1 output feature (price); 26 instances

Note: It can be observed that in each plot, the log-likelihood function increases, as the number of iterations increases, which is exactly the behaviour we expect

See Figures 12 - 19.

5.3 Synthetic dataset

Note: We created a dataset and trained the models on this dataset in order to see if the original parameters are retrieved.

Note 2: In the case of *S3* algorithms, half of the instances were considered supervised and half, unsupervised.

Note 3: In the case of *MS3* algorithms, NAs were introduced at random with 0.1 probability, i.e. for each cell in the training matrix, there is a 0.1 chance for it to become NA.

Note 4: As we will see, *FA* and *PPCA* do not provide reasonable parameters when the output is not standardized.

See Figures 20 - 22.

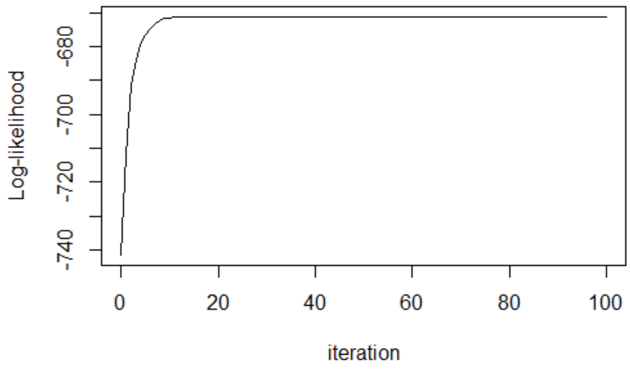

Figure 5.12: Log-likelihood. FA:all the columns without the first one-input;dimension of latent space=1

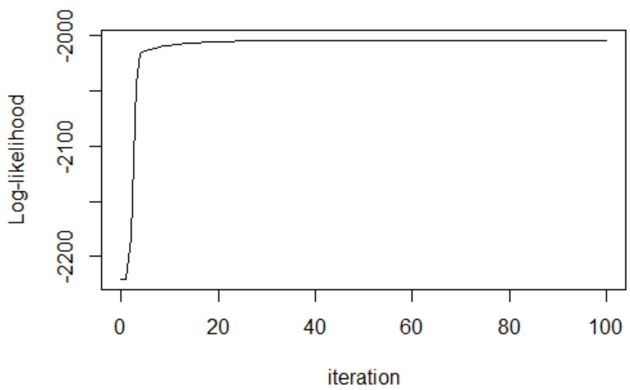

Figure 5.13: Log-likelihood. PPCA: all the columns without the first one - input; dimension of latent space = 1

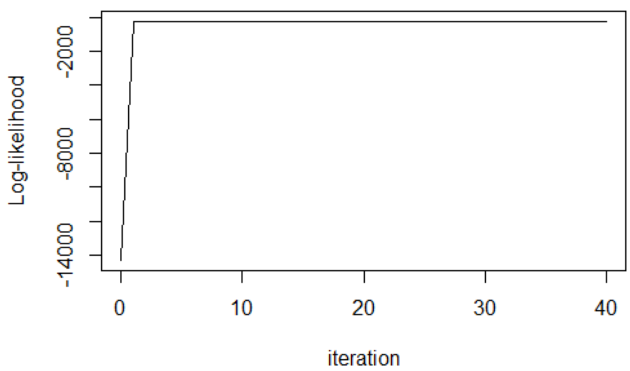

Figure 5.14: Log-likelihood. S3UncFA: only the first 20 rows; the first column - output; the second and third columns - input; first ten rows - supervised; rows 11-20 - unsupervised

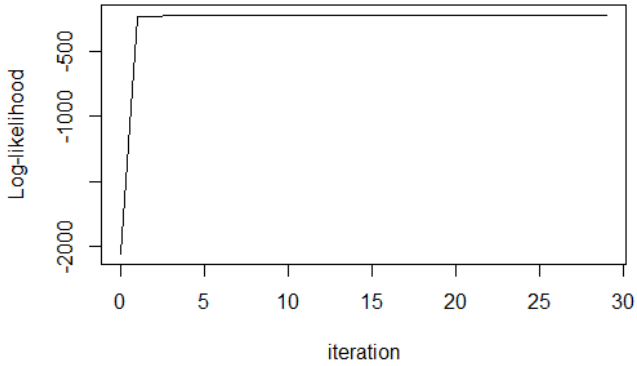

Figure 5.15: Log-likelihood. S3FA: only the first 20 rows; the first column - output; the second and third columns - input; first ten rows - supervised; rows 11-20 - unsupervised

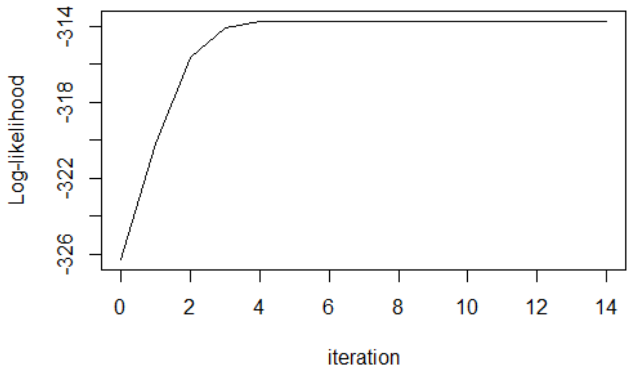

Figure 5.16: Log-likelihood. S3PPCA: only the first 20 rows; the first column - output; the second and third columns - input; first ten rows - supervised; rows 11-20 - unsupervised

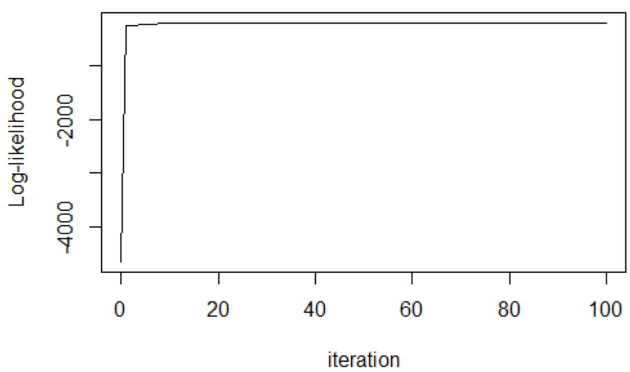

Figure 5.17: Log-likelihood. MS3UncFA: only the first 20 rows; first column - output; the second and third columns - input; NAs on rows 1-10, on columns 2-3 and on rows 11-20, on column 1

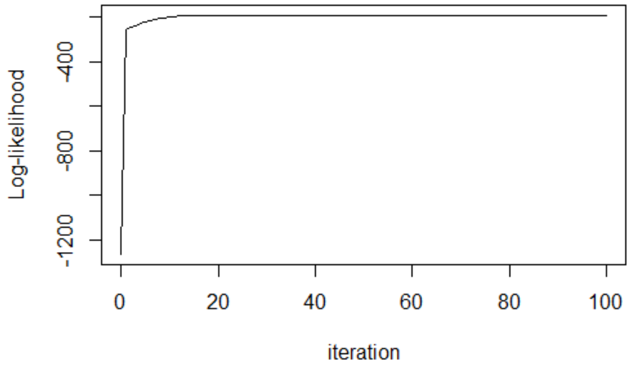

Figure 5.18: Log-likelihood. MS3FA: only the first 20 rows; first column - output; the second and third columns - input; NAs on rows 1-10, on columns 2-3 and on rows 11-20, on column 1

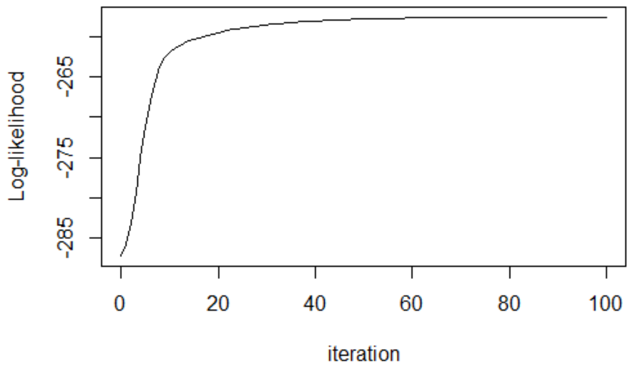

Figure 5.19: Log-likelihood. MS3PPCA: only the first 20 rows; first column - output; the second and third columns - input; NAs on rows 1-10, on columns 2-3 and on rows 11-20, on column 1

	Real	UncFA (No sense)	S2UncFA	S3UncFA	MS3UncFA
mu_z_t	0 0	-	-0.01698442 0.09240896	-0.01112323 0.0876206	-0.01755008 0.09365257
sigma_z_t	1 0 0 1	-	0.94201830 0.09317562 0.09317562 0.88439545	0.94707575 0.09747946 0.09747946 0.87222129	0.94392589 0.09366921 0.09366921 0.88623445
mu_t	0.1130546 0.4173151 0.931297	-	0.2905105 0.5091869 0.7305429	0.1665314 0.5363376 0.7030081	0.2531930 0.6981052 0.7073677
lambda_t	61.97006 44.98000 48.59601 45.48474 61.82364 51.47675	-	61.71871 45.28328 48.17748 45.37308 62.15855 51.27872	61.74634 45.33749 48.26275 45.14258 62.22332 51.18338	61.66480 45.01595 48.21913 45.34420 62.18195 51.10132
psi_t	3.42175496 -3.25167111 -0.05563868 -3.25167111 7.02402871 -0.04794944 -0.05563868 -0.04794944 5.97557407	-	3.745374 -3.4351139 1.5632549 -3.435114 7.1646712 -0.9560841 1.563255 -0.9560841 6.0887514	2.9369178 -2.7752778 0.7810976 -2.7752778 6.6938954 -0.5960124 0.7810976 -0.5960124 5.8742916	3.594599 -3.413740 1.468377 -3.413740 6.854124 -1.191240 1.468377 -1.191240 6.615590

Figure 5.20: Synthetic dataset. Unc: 100 training instances, 3-dimensional input, 2-dimensional output

	Real	FA	S2FA	S3FA	MS3FA
mu_z_t	0	-	0.01866815	0.0167579	0.0197953
sigma_z_t	1	-	0.9940638	0.9904724	0.9906906
mu_t	0.5978012 0.6504156	1.567697 1.597662	0.6341849 0.8980653	0.7282556 0.9685072	0.6203953 0.9059031
lambda_t	50.02886 37.33689	42.99078 32.22177	50.00562 37.47543	50.09231 37.54379	50.06370 37.51553
diag(psi_t)	3.014497 2.438143	3.614494 2.03046	3.219408 2.106209	3.605281 2.070043	3.148793 2.027744

Figure 5.21: Synthetic dataset. FA: 100 training instances, 2-dimensional input, 1-dimensional output

	Real	PPCA	S2PPCA	S3PPCA	MS3PPCA
mu_z_t	0.5894521	-	0.5471244	0.5316495	0.0197953
sigma_z_t	4.191633	-	4.307797	4.501333	0.9906906
mu_t	0.1844163 0.8506565	3.22599 3.036063	0.1967427 1.337151	0.3565684 1.426059	0.6203953 0.9059031
lambda_t	5.54558 3.140015	11.50368 6.408855	5.536671 3.105167	5.397206 3.028318	50.06370 37.51553
diag(psi_t)	6.467466 6.467466	6.240042 6.240042	6.188948 6.188948	6.782395 6.782395	3.148793 2.027744

Figure 5.22: Synthetic dataset. PPCA: 100 training instances, 2-dimensional input, 1-dimensional output

5.4 Single-output regression

Data: Abalone dataset; 4177 instances; 8 input variables; 1 output variable (Rings - age of abalone). The first column was converted to numeric as follows: F - 1, I - 2, M - 3

Note: We applied all the algorithms on the first 1000, 3000 and 4000 rows of this dataset. We tested on the rest of rows and computed Mean Squared Error (MSE), Mean Absolute Error (MAE), Median Absolute Error (MdAE), correlation coefficient between the vector of predicted values and the vector of real values.

Note 2: For the *S2* algorithms, the input and output were the first 1000/3000/4000 rows. For the *S3* algorithms, the supervised input and output were the first 1000/3000/4000 rows and the unsupervised input was the input attributes on rest of rows. For the *MS3* algorithms, the input and output were the whole dataset, but NAs were introduced as follows: on input on the first 1000/3000/4000 rows with 0.1 probability and on output on the rest of rows.

Note 3: The maximum number of iterations for the *EM* algorithms was set to 100.

Note 4: Intuitively, the *S3* versions should work better that the *S2* versions. Moreover, the *MS3* versions should work worse than the *S3* versions. From the results, we encounter this on the unconstrained case (first three columns), but not on the others.

Note 5: From the results, we see that *PPCA* versions perform better than *FA* versions.

See Figures 23 - 25.

	S2UncFA	S3UncFA	MS3UncFA	S2FA	S3FA	MS3FA	S2PPCA	S3PPCA	MS3PPCA	Lin. Regr.
MSE	5.559	5.555	5.854	25.816	32.890	32.868	15.194	91.031	134.134	5.559
MAE	1.855	1.854	1.911	4.058	4.556	4.550	3.123	8.413	10.004	1.855
MdAE	1.570	1.567	1.630	3.439	3.790	3.777	2.560	8.764	12.187	1.570
Cor	0.698	0.698	0.696	0.586	0.575	0.574	0.544	0.071	0.020	0.698

Figure 5.23: Single-output regression: 1000 rows

	S2UncFA	S3UncFA	MS3UncFA	S2FA	S3FA	MS3FA	S2PPCA	S3PPCA	MS3PPCA	Lin. Regr.
MSE	4.533	4.533	4.557	11.876	13.603	13.482	7.715	9.718	11.097	4.533
MAE	1.610	1.610	1.615	2.767	2.984	2.969	2.072	2.288	2.440	1.610
MdAE	1.268	1.268	1.275	2.406	2.622	2.608	1.531	1.534	1.650	1.268
Cor	0.704	0.704	0.703	0.578	0.574	0.574	0.535	0.489	0.452	0.704

Figure 5.24: Single-output regression: 3000 rows

	S2UncFA	S3UncFA	MS3UncFA	S2FA	S3FA	MS3FA	S2PPCA	S3PPCA	MS3PPCA	Lin. Regr.
MSE	2.208	2.208	2.223	10.182	10.441	10.399	5.081	5.171	5.187	2.208
MAE	1.172	1.172	1.175	2.524	2.557	2.552	1.707	1.720	1.722	1.172
MdAE	0.993	0.993	1.001	2.124	2.151	2.143	1.324	1.348	1.312	0.993
Cor	0.711	0.711	0.709	0.745	0.745	0.745	0.713	0.712	0.713	0.711

Figure 5.25: Single-output regression: 4000 rows

5.5 Impute missing data

Data: Boston House Price Dataset; 506 observations; 13 input variables; 1 output variable (MEDV - house price)

Note: We introduced NAs into the dataset at random with 0.1 probability. We wanted to see how the imputation algorithm works. We compared it against the imputation given by the mean of the corresponding column. We computed the MSE between the real values and the imputed ones.

Note 2: We were also curious if the input/output columns matter. As a result, we gave the 13 columns as input and the other column as output (13_1 case) and then we gave the first 7 columns as input and the other 7 columns as output (7_7 case).

Note 3: From the results, we see that the mean imputation is the worst among the others. In the unconstrained case, the input/output columns do not matter (the result is the same), but for the *FA* and *PPCA* versions, we obtained better results in the 7_7 case.

See Figure 26.

5.6 Data augmentation

Data: Boston House Price Dataset; 506 observations; 13 input variables; 1 output variable (MEDV - house price)

Note: We wanted to see if data augmentation can help a regressor. We chose a regression tree model ('rpart' package in r). We trained a regression tree on the dataset (only on the first 400 rows), learnt the parameters of our *S2* model, generated 100 new instances via the known *S2* generation procedure, added those to the training dataset and trained a regression tree on this augmented dataset. We computed the MSE, MAE, MdAE and correlation coefficient between the real values and the predicted ones in each case. We executed this procedure 5 times. *S2PPCA* does not generally give improvements, but the other two seem to help the regressor

Mean	MS3UncFA 13_1 case	MS3FA 13_1 case	MS3PPCA 13_1 case	MS3UncFA 7_7 case	MS3FA 7_7 case	MS3PPCA 7_7 case
247.83	79.579	233.074	246.726	79.579	83.078	92.016

Figure 5.26: Impute missing data

	None	S2UncFA	S2FA	S2PPCA
MSE	54.6032964912411	18.9899257631364 31.8378296477406 21.3312055759326 26.1855496650328 30.8527350903843	23.4357395906758 19.7168454456672 19.6978291677237 42.1246407911356 29.8385461398302	59.6056759920788 29.2674606974383 54.2180323061209 55.442101380417 46.1089478496146
MAE	5.10758484121894	3.4072616033526 4.3560238568827 3.43847852311047 3.81795446632395 3.97358282751157	3.56575826397014 3.38390650015026 3.40535929394287 5.13308341362212 4.25015201621889	5.60982706798336 4.26750744074158 5.53370310266137 5.51352112908655 5.12140261831075
MdAE	3.72780448717949	2.91929849933377 4.09071963816178 2.66932063622748 3.53323869759357 2.74798439340666	2.60419248641536 2.73645881377479 2.71624283343786 4.24704337050196 3.82212794784922	4.03473804109253 3.27361322555969 3.95293421088071 3.83565838047459 3.84794743362689
Cor	0.570245829704148	0.634686153181339 0.496262781418265 0.584592242957179 0.653144050889019 0.574731592497647	0.59827507396356 0.573740756343142 0.612232987869697 0.612635499921113 0.592321538715421	0.604759443196241 0.573012553785038 0.544968300938752 0.544014211487599 0.571269994950697

Figure 5.27: Data augmentation

Unit: seconds	min	mean	median	max	neval
S3FA – fit	4.548421	4.683793	4.641505	4.823499	10
MS3FA – fit	9.851952	10.253672	10.168517	11.123803	10

Figure 5.28: Time: Matrix vs non-matrix form

many times.

See Figure 27.

5.7 Time comparisons

5.7.1 Matrix vs non-matrix form

Note: The *MS3* algorithms are a generalization of the others. We implemented the other ones in matrix form and the *MS3* algorithms in non-matrix form. An advantage of using vectorised code is better speed. We generated 1000 instances with 20-dimensional input and 10-dimensional output with a diagonal Ψ. We ran the *S3FA* and *MS3FA* algorithms (we handled them to do exactly the same task) 10 times. One can notice that *S3FA* is twice faster than *MS3FA*.

See Figure 28.

5.7.2 turboEM

Data: Boston House Price Dataset; 506 observations; 13 input variables; 1 output variable (MEDV - house price); used only the first 300 instances

Note: We compared the execution times of the *S3* versions of the algorithms using *turboEM*. We noticed that when the stopping criterion is linked to the parameters, then simple *EM* is the fastest. We also noticed that when the stopping criterion is linked to the log-likelihood, then *SquarEM* is the fastest.

See Figure 29.

S3_	stop	method	value.objfn	itr	fpeval	objfeval	convergence	elapsed.time
UncFA	objfn	em	18959.10	100	100	101	FALSE	9.57
UncFA	objfn	squarem	18959.81	9	17	11	TRUE	1.07
UncFA	objfn	pem	18953.42	100	210	733	FALSE	69.92
FA	objfn	em	20211.32	100	100	101	FALSE	9.06
FA	objfn	squarem	20211.32	13	25	14	TRUE	1.25
FA	objfn	pem	20211.32	6	22	42	TRUE	3.63
PPCA	objfn	em	33247.17	100	100	101	FALSE	8.53
PPCA	objfn	squarem	33247.14	10	18	12	TRUE	1.02
PPCA	objfn	pem	33247.14	8	26	49	TRUE	4.08
UncFA	param	em	18959.10	100	100	1	FALSE	0.20
UncFA	param	squarem	18953.82	100	199	126	FALSE	12.75
UncFA	param	pem	18953.42	100	210	740	FALSE	74.93
FA	param	em	20211.32	100	100	1	FALSE	0.19
FA	param	squarem	20211.32	15	29	16	TRUE	1.43
FA	param	pem	20211.32	6	22	42	TRUE	3.58
PPCA	param	em	33247.17	100	100	1	FALSE	0.17
PPCA	param	squarem	33247.14	13	24	15	TRUE	1.47
PPCA	param	pem	33247.14	7	24	46	TRUE	3.86

Figure 5.29: Time: turboEM

Chapter 6

Conclusion and future work

The **purpose** of this work was to create a **new probabilistic generative model: *S2FA* (Simple-Supervised Factor Analysis)**, the **supervised counterpart** of the *Factor Analysis* Model and to extend it as much as possible. We proceeded as follows: we created *S2UncFA*, *S2FA* and *S2PPCA* versions, we derived in detail the *EM* algorithms for the **semi-supervised** versions (*S3UncFA*, *S3FA*, *S3PPCA*), we developed models that can learn the parameters with **missing data** and **impute missing data in input or output** (*MS3UncFA*, *MS3FA*, *MS3PPCA*). We observed some relationships with the *Linear Regression* Model: *S2UncFA* is **strongly** equivalent to *Linear Regression*, *S2FA* and *S2PPCA* are **weakly** equivalent to it. We developed an **R package** (s2fa) in order to test the algorithms and to let them be publicly accessible. Some implementation details were noted in the corresponding section. Furthermore, **other extensions** and why they work or not were discussed: Weighted, Ridge, Kernel, Discrete data. In the **experimental** part, we found that on some datasets our algorithms can be used for regression, to impute missing data, to augment a dataset and that some of our algorithms have an advantage (good speed), because they were implemented in matrix form.

As **future work**, we thought of the following: compute the **second derivatives** (Hessian) in order to have the proofs completed, integrate the functions in the R package not only with *turboEM* package, but also with ***FixedPoint*** [25] which completes the list of fixed-point acceleration schemes, continue the idea of handling discrete input data via **Bernoulli** distribution and sigmoid function, **compare the imputation** of *MS3* algorithms with the imputation given by learning the parameters of a multivariate normal distribution with missing data, apply imputation in **image inpainting**. Because there are concepts like mixture of factor analyzers [19] and of mixture of Linear Regression models [26, sec.14.5.1], another direction involves **mixtures of *S2FAs*** (and others). A recent paper discusses a model called ***Deep*** *Gaussian Mixture Models* [27] and so one should try also to adapt it to *S2FA*. As *Linear Regression* and *Logistic Regression* can be considered the seed of neural networks, the same may apply to *S2FA*, but this must be more studied. One last idea would be to see how much **memory** the algorithms need and to think if the algorithms can be implemented in a **big data** framework like *Spark* [28] in order to not be constrained by memory limits.

Bibliography

[0] Liviu Ciortuz, Alina Munteanu, Elena Bădărău - Machine Learning exercise book - 2018

[1] Tom Mitchell - Generative and Discriminative Classifiers: Naive Bayes and Logistic Regression - draft chapter intended for inclusion in the upcoming second edition of the textbook *Machine Learning*, T.M. Mitchell - 2017. Available: https://www.cs.cmu.edu/~tom/mlbook/NBayesLogReg.pdf

[2] Andrew Ng, Stanford University, ML course, 2009 fall, lecture notes, parts VIII and IX

[3] CMU, Machine Learning course, 2010 fall, Aarti Singh, HW4, pr 1.1. In [0, pag. 492].

[4] Andrew Ng, Stanford University, ML course, lecture notes, part X. Available: http://cs229.stanford.edu/notes/cs229-notes9.pdf

[5] Kevin Murphy - Machine Learning: A Probabilistic Perspective - MIT press - 2012

[6] Andrew Ng, Stanford University, ML course, lecture notes, part XI. Available: http://cs229.stanford.edu/notes/cs229-notes10.pdf

[7] Michael Tipping, Christopher Bishop - Probabilistic Principal Components Analysis - Journal of the Royal Statistical Society: Series B (Statistical Methodology) 61.3 (1999): 611-622. Available: http://www.robots.ox.ac.uk/~cvrg/hilary2006/ppca.pdf

[8] Neil Lawrence - Gaussian Process Latent Variable Models for Visualisation of High Dimensional Data - Advances in neural information processing systems - 2004. Available: https://papers.nips.cc/paper/2540-gaussian-process-latent-variable-models-for-visualisation-of-high-dimensional-data.pdf

[9] Xinbo Gao, Xiumei Wang, Dacheng Tao, Xuelong Li - Supervised Gaussian Process Latent Variable Model for Dimensionality Reduction - IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics) 41.2 (2010): 425-434. Available: https://ieeexplore.ieee.org/document/5545418

[10] F. Serhan Daniş, Ali Taylan Cemgil - Factor Analysis: A Tutorial - 2015. Available: https://www.researchgate.net/publication/301565604_Factor_Analysis_A_Tutorial

[11] Kaare Brandt Petersen, Michael Syskind Pedersen - Matrix Cookbook - Technical University of Denmark - 2012. Available: https://www.math.uwaterloo.ca/~hwolkowi/matrixcookbook.pdf

[12] Sam Roweis - Matrix Identities, 1999 - Note available on the internet (May 20, 2004) at https://cs.nyu.edu/~roweis/notes/matrixid.pdf

[13] CMU, Machine Learning course, 2011 fall, Eric Xing, HW1, pr. 2.1. In [0, pag. 192].

[14] Stanford, Machine Learning course, 2007 fall, Andrew Ng, HW1, pr. 3. In [0, pag. 202].

[15] CMU, Machine Learning course, 2009 spring, Ziv Bar-Joseph, HW2, pr. 2. In [0, pag. 361].

[16] CMU, Machine Learning course, 2011 spring, Tom Mitchell, HW2, pr. 2.2. In [0, pag. 365].

[17] Chuong B. Do - The Multivariate Gaussian Distribution - Stanford, CA, USA - 2008. Available: http://cs229.stanford.edu/section/gaussians.pdf

[18] CMU, Machine Learning course, 2010 spring, T. Mitchell, E. Xing, A. Singh, midterm, pr. 5.3. In [0, pag. 530].

[19] Zoubin Ghahramani, Geoffrey E. Hinton - The EM Algorithm for Mixtures of Factor Analyzers - Vol. 60. Technical Report CRG-TR-96-1, University of Toronto - 1998. Available: http://mlg.eng.cam.ac.uk/zoubin/papers/tr-96-1.pdf

[20] StackExchange - Second Answer on 'Trace trick for expectations of quadratic forms'. Available: https://math.stackexchange.com/questions/2228398/trace-trick-for-expectations-of-quadratic-forms

[21] David Sontag, Introduction to Bayesian methods, Lecture 10, New York University. Available: http://people.csail.mit.edu/dsontag/courses/ml14/slides/lecture10.pdf

[22] Michael Tipping - Probabilistic Visualisation of High-dimensional Binary Data - Advances in neural information processing systems - 1999. https://pdfs.semanticscholar.org/8046/686197e65c94a31c573b14d6c52c19239a18.pdf

[23] Jennifer F. Bobb and Ravi Varadhan - turboEM: A Suite of Convergence Acceleration Schemes for EM and MM algorithms. Available: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.461.8280&rep=rep1&type=pdf

[24] Package 'turboEM' - 2018. Available: https://cran.r-project.org/web/packages/turboEM/turboEM.pdf

[25] Stuart Baumann, Margaryta Klymak - FixedPoint: A suite of acceleration algorithms with applications - 2019. Available: https://cran.r-project.org/web/packages/FixedPoint/vignettes/FixedPoint.pdf

[26] Christopher Bishop - Pattern Recognition and Machine Learning - Springer - 2006

[27] Cinzia Viroli, Geoffrey J. McLachlan - Deep Gaussian Mixture Models - Statistics and Computing 29.1 (2019): 43-51. Available: https://link.springer.com/article/10.1007/s11222-017-9793-z

[28] https://spark.apache.org/