entropy

MDPI

Article

A Factor Analysis Perspective on Linear Regression in the 'More Predictors than Samples' Case

Sebastian Ciobanu * and Liviu Ciortuz *

Faculty of Computer Science, Alexandru Ioan Cuza University of Iaşi, 700506 Iaşi, Romania
* Correspondence: aciobanusebi@gmail.com (S.C.); liviu_ciortuz@yahoo.co.uk (L.C.)

Abstract: Linear regression (LR) is a core model in supervised machine learning performing a regression task. One can fit this model using either an analytic/closed-form formula or an iterative algorithm. Fitting it via the analytic formula becomes a problem when the number of predictors is greater than the number of samples because the closed-form solution contains a matrix inverse that is not defined when having more predictors than samples. The standard approach to solve this issue is using the Moore–Penrose inverse or the L2 regularization. We propose another solution starting from a machine learning model that, this time, is used in unsupervised learning performing a dimensionality reduction task or just a density estimation one—factor analysis (FA)—with one-dimensional latent space. The density estimation task represents our focus since, in this case, it can fit a Gaussian distribution even if the dimensionality of the data is greater than the number of samples; hence, we obtain this advantage when creating the supervised counterpart of factor analysis, which is linked to linear regression. We also create its semisupervised counterpart and then extend it to be usable with missing data. We prove an equivalence to linear regression and create experiments for each extension of the factor analysis model. The resulting algorithms are either a closed-form solution or an expectation–maximization (EM) algorithm. The latter is linked to information theory by optimizing a function containing a Kullback–Leibler (KL) divergence or the entropy of a random variable.

Keywords: more predictors than samples; linear regression; factor analysis; semisupervised regression; missing data

Citation: Ciobanu, S.; Ciortuz, L. A Factor Analysis Perspective on Linear Regression in the 'More Predictors than Samples' Case. *Entropy* **2021**, *23*, 1012. https://doi.org/10.3390/e23081012

Academic Editor: Kevin R. Moon

Received: 5 June 2021
Accepted: 27 July 2021
Published: 3 August 2021

1. Introduction

In machine learning, models can be grouped into two categories: probabilistic and nonprobabilistic. Probabilistic models can be classified as generative and discriminative [1]. Examples of classic generative models are naive Bayes and Gaussian mixture models (GMM). Examples of classic discriminative models are linear regression (LR) and logistic regression. The key difference is whether they model the joint probability of the input and the output—generative models—or they just model the conditional probability of the output given the input—discriminative models. For a classification or a regression task, one may argue that what you need is just a discriminative model, but the generative models have their advantages: they can sometimes handle missing data, can easily generate new data, can be extended to be unsupervised or semisupervised, etc. ([2] p. 268).

As one may notice, there are generative models for unsupervised learning that have counterparts in supervised learning, even though this is not widely discussed in the literature. One such example is the GMM ([2] p. 339) with its counterpart, the Gaussian joint Bayes model ([2] p. 102), also known as quadratic discriminant analysis. Their training/fitting algorithms are similar, as one may notice, for example, in [3,4]:

- for Gaussian joint Bayes:

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

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

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

where x_i is an input observation, (π_j, μ_j, Σ_j) are the parameters of a GMM, observable z_i is the class index corresponding to x_i, j is a class index, and $1_{\{z_i=j\}}$ is the indicator function, which returns 1 if the condition $z_i = j$ is true and 0 otherwise.

- for expectation–maximization (EM) for the GMM, which optimizes a function concerning a Kullback–Leibler (KL) divergence or the entropy of a random variable—check Appendix A for these details—:
  E step:

$$\begin{aligned} w_{ij} &= p(z_i = j | x_i; \pi', \mu', \Sigma') \\ &= \frac{p(x_i | z_i = j; \mu', \Sigma') p(z_i = j; \pi')}{\sum_{l=1}^{K} p(x_i | z_i = l; \mu', \Sigma') p(z_i = l; \pi')} \end{aligned}$$

  M step:

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

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

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

  where x_i is an input observation, (π_j, μ_j, Σ_j) are the parameters of a GMM, unobservable z_i is the cluster index corresponding to x_i, j is a cluster index, and w_{ij} is the probability that x_i belongs to cluster j.

This similarity between GMM and Gaussian joint Bayes is intriguing; hence, we decided to further explore this aspect but starting from other supervised–unsupervised counterparts. As a result, we changed the root model into factor analysis (FA) [5] ([2] p. 381), which is normally used for dimensionality reduction or for density estimation when the dimensionality of the data is greater than the number of samples. Factor analysis is a Gaussian generative model used in unsupervised learning. We aimed at creating its supervised counterpart in order to handle a regression task and then exploit it as much as possible.

After creating the supervised counterpart, we proved a significant property, namely that linear regression is equivalent to (supervised) factor analysis—with one-dimensional latent space—when no constraints are imposed on the covariance matrices.

A linear regression model can be fitted via a closed-form solution or an iterative algorithm. When the number of predictors is greater than the number of samples, there is no closed-form solution. There are other solutions to this problem, as we will see.

We were at the point where we knew that factor analysis was linked to linear regression and that it could be used when the number of samples was lower than the dimensionality of the data—from now on, this is denoted as $D >> n$ or $n << D$. As a result, we shifted our focus from solely exploiting the factor analysis model to highlighting novel linear regression versions applicable in the $D >> n$ regime—linear regression being a widely known and used model—:

- linear regression when $D >> n$,
- semisupervised linear regression when $D >> n$,
- (semisupervised) linear regression when $D >> n$ with missing data.

The structure of this paper is as follows. In Section 2, we include some theoretical background to enhance the readability of this paper. Section 3 contains related work. In Section 4, we include the models we proposed, starting from factor analysis. Section 5 contains experiments using the proposed models. In Section 6, we conclude the paper and show future directions.

We include the full algorithms in the appendices in a pseudocode format, two of them being instances of the expectation–maximization schema.

2. Theoretical Background

We started our analysis from two core models in machine learning: linear regression and factor analysis. We will discuss the aspects of those two models that are relevant to understanding the next sections of this paper.

2.1. Linear Regression

Proposition 1. *Let* $\{(x^{(i)}, y^{(i)}) | x^{(i)} \in \mathbb{R}^{D\times 1}, y^{(i)} \in \mathbb{R}, i \in \{1, \ldots, n\}\}$ *be a data set where* D *is the dimensionality of the input data,* $\{x^{(i)} | i \in \{1, \ldots, n\}\}$ *is the input, and* $\{y^{(i)} | i \in \{1, \ldots, n\}\}$ *is the output. The linear regression model is as follows:*

$$Y^{(i)} = wx^{(i)} + b + \epsilon^{(i)}$$

where $Y^{(i)}$ *is a random variable corresponding to* $y^{(i)}$, $\epsilon^{(i)} \sim \mathcal{N}(0, \sigma^2)$, $\sigma \in \mathbb{R}_+^*$, $w \in \mathbb{R}^{1\times D}$, $b \in \mathbb{R}$. *Then, the parameters* w *and* b *can be estimated via maximum likelihood as follows:*

$$\hat{w}_{LR} = \left(n\bar{y}\bar{x}^\top - YX^\top\right)\left(n\bar{x}\bar{x}^\top - XX^\top\right)^{-1} \tag{1}$$

$$\hat{b}_{LR} = \bar{y} - \hat{w}_{LR}\bar{x}, \tag{2}$$

or, equivalently, as follows:

$$\begin{bmatrix} \hat{w}_{LR}^\top \\ \hat{b}_{LR} \end{bmatrix} = (\tilde{X}\tilde{X}^\top)^{-1}\tilde{X}Y^\top \tag{3}$$

where $\bar{x} = \dfrac{x^{(1)} + \cdots + x^{(n)}}{n}$, $\bar{y} = \dfrac{y^{(1)} + \cdots + y^{(n)}}{n}$,

$$X = \begin{bmatrix} x^{(1)} & \ldots & x^{(n)} \end{bmatrix} \in \mathbb{R}^{D\times n}, \tilde{X} = \begin{bmatrix} \tilde{x}^{(1)} & \ldots & \tilde{x}^{(n)} \end{bmatrix} \in \mathbb{R}^{(D+1)\times n},$$

$$\tilde{x}^{(i)} = \begin{bmatrix} x^{(i)} \\ 1 \end{bmatrix} = \begin{bmatrix} x_1^{(i)} \\ \vdots \\ x_D^{(i)} \\ 1 \end{bmatrix} \in \mathbb{R}^{D+1}, \text{and } Y = \begin{bmatrix} y^{(1)} & \ldots & y^{(n)} \end{bmatrix} \in \mathbb{R}^{1\times n}.$$

A potential problem with Equation (3) is when $\tilde{X}\tilde{X}^\top$ is not invertible. Such case arises when $D + 1 > n$, i.e., there are more predictors than samples. Two standard solutions to this problem are the following:

- Let $A \in \mathbb{R}^{a\times b}$ be a matrix. Then, the Moore–Penrose inverse of A can be defined as $A^+ = \lim_{\alpha\to 0^+}(A^\top A + \alpha I)^{-1}A^\top$, which, algorithmically, is computed via the singular value decomposition of A ([6] Section 2.9). One may notice that the matrix $(\tilde{X}\tilde{X}^\top)^{-1}\tilde{X}$ from Equation (3) is just $(\tilde{X}^\top)^+$ when $\tilde{X}\tilde{X}^\top$ is invertible. When it is not, the solution is to replace the matrix $(\tilde{X}\tilde{X}^\top)^{-1}\tilde{X}$ from Equation (3) with $(\tilde{X}^\top)^+$.

- L2 regularization, which results in ridge regression ([2] p. 225). The matrix $(\tilde{X}\tilde{X}^\top)^{-1}\tilde{X}$ from Equation (3) is replaced with $\left(\tilde{X}\tilde{X}^\top + \begin{bmatrix} \alpha & \dots & 0 & 0 \\ \vdots & \ddots & \vdots & \vdots \\ 0 & \dots & \alpha & 0 \\ 0 & \dots & 0 & 0 \end{bmatrix}\right)^{-1} \tilde{X}$, with $\alpha > 0$; the bigger the α, the more regularization we add to the model, i.e., move away from overfitting. From this point of view, the first solution using the Moore–Penrose inverse can be interpreted as achieving the asymptotically lowest L2 regularization.

2.2. Factor Analysis

The formulas stated in the conclusion of the following Proposition were proved in [5] and are relevant for the factor analysis algorithm—although the matrix Ψ is considered as being diagonal there, the formulas stay the same even if Ψ is not diagonal.

Proposition 2. *Let us consider the following factor analysis model:*
$z \sim \mathcal{N}(0, I)$*-latent variable,* $z \in \mathbb{R}^{d\times 1}$
$x|z \sim \mathcal{N}(\mu + \Lambda z, \Psi)$*,* $x \in \mathbb{R}^{D\times 1}, \mu \in \mathbb{R}^{D\times 1}, \Lambda \in \mathbb{R}^{D\times d}$*, and* $\Psi \in \mathbb{R}^{D\times D}$ *a diagonal matrix. Then:*

$$\begin{bmatrix} x \\ z \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \mu \\ 0 \end{bmatrix}, \begin{bmatrix} \Lambda\Lambda^\top & \Lambda \\ \Lambda^\top & I \end{bmatrix}\right)$$
$$x \sim \mathcal{N}(\mu, \Lambda\Lambda^\top + \Psi)$$
$$z|x \sim \mathcal{N}(\Lambda^\top(\Lambda\Lambda^\top + \Psi)^{-1}(x - \mu), I - \Lambda^\top(\Lambda\Lambda^\top + \Psi)^{-1}\Lambda).$$

A factor analysis model can be fitted via an EM algorithm using the maximum likelihood estimation (MLE) principle, and factor analysis can be used as a density estimation technique when the dimensionality of the data is greater than the number of samples [5].

For the algorithms we developed, we will let $z \sim \mathcal{N}(\mu_z, \Sigma_z)$ 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 $z \sim \mathcal{N}(0, I)$. This generalization leads to the following result.

Proposition 3. *Let us consider the following linear Gaussian system:*
$z \sim \mathcal{N}(\mu_z, \Sigma_z)$*-latent variable,* $z \in \mathbb{R}^{d\times 1}$*,* $\mu_z \in \mathbb{R}^{d\times 1}$*,* $\Sigma_z \in \mathbb{R}^{d\times d}$ *a symmetric and positive definite matrix,*
$x|z \sim \mathcal{N}(\mu + \Lambda z, \Psi)$*,* $x \in \mathbb{R}^{D\times 1}, \mu \in \mathbb{R}^{D\times 1}, \Lambda \in \mathbb{R}^{D\times d}$*, and* $\Psi \in \mathbb{R}^{D\times D}$ *a symmetric and positive definite matrix. (If* Ψ *is a diagonal matrix, then we say that we are in the FA case. If* Ψ *is a scalar matrix,* $\Psi = \eta I$*,* $\eta \in \mathbb{R}_+^*$*, then we are in the probabilistic principal component analysis (PPCA) case [7]. If* Ψ *is any symmetric and positive definite matrix, then we say we are in the unconstrained factor analysis (UncFA) case. If the first two terms are standard (FA and PPCA), the third one is proposed by us—UncFA.) Then:*

$$\begin{bmatrix} x \\ z \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \mu + \Lambda\mu_z \\ \mu_z \end{bmatrix}, \begin{bmatrix} \Lambda\Sigma_z\Lambda^\top + \Psi & \Lambda\Sigma_z \\ (\Lambda\Sigma_z)^\top & \Sigma_z \end{bmatrix}\right)$$
$$x \sim \mathcal{N}(\mu + \Lambda\mu_z, \Lambda\Sigma_z\Lambda^\top + \Psi)$$
$$z|x \sim \mathcal{N}(\mu_z + \Sigma_z\Lambda^\top(\Lambda\Sigma_z\Lambda^\top + \Psi)^{-1}(x - \mu - \Lambda\mu_z), \Sigma_z - \Sigma_z\Lambda^\top(\Lambda\Sigma_z\Lambda^\top + \Psi)^{-1}\Lambda\Sigma_z^\top). \quad (4)$$

The proof can be found in ([8] pp. 9–11).

3. Related Work

Although factor analysis is widely used for dimensionality reduction, its supervised counterpart is, to the best of our knowledge, not present in the literature. What is present is a model called supervised principal component analysis or latent factor regression ([2] 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 principal component analysis 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 system ([2] p. 119). This was already presented in Section 2, and it generalizes the factor analysis generative process by considering that z has a learnable mean and covariance matrix, but it does not go further.

Factor analysis is strongly related to principal component analysis (PCA) [9] because by imposing a certain constraint in factor analysis, we get a model called probabilistic principal component analysis [7] that can be fitted using a closed-form solution, which, in an asymptotic case, is also the solution for PCA. Probabilistic PCA can be kernelized using a model called the Gaussian process latent variable model (GPLVM) [10]. This model also has supervised counterparts [11], but, as in the case of FA, the supervised extension targets dimensionality reduction, and the idea is similar to the one in supervised PCA.

4. Proposed Models

In this section, we propose three models starting from the FA model, each in a new subsection: simple-supervised factor analysis (S2.FA), simple-semisupervised factor analysis (S3.FA), and missing simple-semisupervised factor analysis (MS3.FA). While S2.FA is applicable in the supervised case, regression, S3.FA is meant to be used in a semisupervised context. MS3.FA handles missing input data in a (semi)supervised scenario.

One important remark that will not be restated in this paper is that all the models (FA, S2.FA, S3.FA, MS3.FA) are fitted by maximizing the likelihood (the MLE principle) of the observed data. Another important observation is regarding the names of our proposed algorithms: the algorithms are called "simple-"—S2 = simple-supervised; S3 = simple-semisupervised; MS3 = missing simple-semisupervised—not only because they constitute a simple adaptation of the factor analysis model, but mostly because we created an adaptation of the (simple-)supervised FA model called (simple-)supervised PPCA, and we did not want this model to be confused with the already existing supervised PCA model in the literature. Simple-supervised probabilistic principal component analysis (S2.PPCA) is not discussed in this paper, but it is implemented and usable in the R package that we developed (https://github.com/aciobanusebi/s2fa; accessed on 31 July 2021) along with other undiscussed but related models: Simple-semisupervised unconstrained factor analysis (S3.UncFA), Simple-semisupervised probabilistic principal component analysis (S3.PPCA), Missing simple-semisupervised unconstrained factor analysis (MS3.UncFA), and Missing simple-semisupervised probabilistic principal component analysis (MS3.PPCA).

4.1. The S2.FA Model

The core of this subsection regards the S2.FA model, but in order to link it with LR, we need to also introduce a similar model to S2.FA: S2.UncFA. This link will make S2.FA a good candidate for replacing LR when $D >> n$. These three ideas—S2.FA and S2.UncFA, S2.UncFA-LR link, and replacing LR via S2.FA—will be expanded below.

4.1.1. The S2.FA Model. S2.FA and S2.UncFA

The first model that we propose is called simple-supervised factor analysis. It is a linear Gaussian system with slight changes:

$z \sim \mathcal{N}(\mu_z, \sigma_z^2)$-observed variable, $z \in \mathbb{R}$, $\mu_z \in \mathbb{R}$, $\sigma_z \in \mathbb{R}_+^*$

$x|z \sim \mathcal{N}(\mu + \Lambda z, \Psi)$, $x \in \mathbb{R}^{D\times 1}, \mu \in \mathbb{R}^{D\times 1}, \Lambda \in \mathbb{R}^{D\times 1}$, and $\Psi \in \mathbb{R}^{D\times D}$ a diagonal matrix.

If we do not impose the constraint of Ψ being diagonal, we arrive at the simple-supervised unconstrained factor analysis (S2.UncFA):

$z \sim \mathcal{N}(\mu_z, \sigma_z^2)$-observed variable, $z \in \mathbb{R}$, $\mu_z \in \mathbb{R}$, $\sigma_z \in \mathbb{R}_+^*$

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

In contrast with the factor analysis model, which is fitted via an EM algorithm, S2.UncFA and S2.FA are fitted via analytic formulas (see Propositions 4 and 5).

Proposition 4. *Let* $\{(x^{(i)}, z^{(i)}) | x^{(i)} \in \mathbb{R}^{D\times 1}, z^{(i)} \in \mathbb{R}^{d\times 1}, i \in \{1, \ldots, n\}\}$ *be a data set where* D *is the dimensionality of the input data,* d *is the dimensionality of the output data (although in the context of this paper* $d = 1$*, we decided to expose more general results—*$d \geq 1$*—in order for the reader to gain more insight; this is the reason why we write* Σ_z *and not just* σ_z^2*, or* ${z^{(i)}}^\top$ *and not just* $z^{(i)}$*, or* $\bar{z}^\top$ *and not just* $\bar{z}$*, etc.),* $\{x^{(i)} | i \in \{1, \ldots, n\}\}$ *is the input, and* $\{z^{(i)} | i \in \{1, \ldots, n\}\}$ *is the output. We suppose that the data was generated as follows:*

$z^{(i)} \sim \mathcal{N}(\mu_z, \Sigma_z)$, $z^{(i)} \in \mathbb{R}^{d\times 1}$, $\mu_z \in \mathbb{R}^{d\times 1}$, $\Sigma_z \in \mathbb{R}^{d\times d}$ *a symmetric and positive definite matrix and*

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

Then, the parameters in the S2.UncFA algorithm (training phase) can be estimated via maximum likelihood using the following closed-form formulas:

$$\hat{\mu}_z = \frac{\sum_{i=1}^n z^{(i)}}{n} \tag{5}$$

$$\hat{\Sigma}_z = \frac{\sum_{i=1}^n (z^{(i)} - \hat{\mu}_z)(z^{(i)} - \hat{\mu}_z)^\top}{n} \tag{6}$$

$$\hat{\Lambda} = \left(n\bar{x}\bar{z}^\top - \sum_{i=1}^n x^{(i)} {z^{(i)}}^\top\right)\left(n\bar{z}\bar{z}^\top - \sum_{i=1}^n z^{(i)} {z^{(i)}}^\top\right)^{-1} \tag{7}$$

$$\hat{\mu} = \bar{x} - \hat{\Lambda}\bar{z} \tag{8}$$

$$\hat{\Psi} = \frac{\sum_{i=1}^n (x^{(i)} - \hat{\mu} - \Lambda z^{(i)})(x^{(i)} - \hat{\mu} - \Lambda z^{(i)})^\top}{n} \tag{9}$$

where $\bar{x} = \frac{\sum_{i=1}^n x^{(i)}}{n}$ *and* $\bar{z} = \hat{\mu}_z$*. For the testing/prediction phase, one uses the formula for* $z|x$ *from* (4).

For more elaborate notations and the proof, see [8] pp. 13–17.

Proposition 5. *[We will denote the parameter* $\hat{\Psi}$ *in* (9) *as* $\hat{\Psi}_{S2.UncFA}$*.]*

For the S2.FA algorithm, (9) *is replaced by*

$$\hat{\Psi} = diag\left(\hat{\Psi}_{S2.UncFA}\right) \tag{10}$$

where "diag" takes the diagonal of a matrix and returns the corresponding diagonal matrix.

The proof of Equation (10) is relatively simple, and we skip it for brevity. It can be found in [8] pp. 21–23.

For the step-by-step S2.FA algorithm and also for the matrix form of the algorithm, see Appendix B.

4.1.2. The S2.FA Model. The Link between LR and S2.UncFA

Linear regression and S2.UncFA have the same prediction function after fitting, as we claim and prove below.

Proposition 6. *Let* $\{(x^{(i)}, z^{(i)}) | x^{(i)} \in \mathbb{R}^{D\times 1}, z^{(i)} \in \mathbb{R}^d, i \in \{1, \ldots, n\}\}$ *be a data set where* D *is the dimensionality of the input data,* d *is the dimensionality of the output data (The same observation as earlier: in the context of this paper, only the "*$d = 1$*" case is relevant.),* $\{x^{(i)} | i \in \{1, \ldots, n\}\}$ *is the input, and* $\{z^{(i)} | i \in \{1, \ldots, n\}\}$ *is the output.*

One can fit an S2.UncFA model and obtain—via the relationships (5)–(9)—$\hat{\mu}_z$, $\hat{\Sigma}_z$, $\hat{\Psi}$, $\hat{\Lambda}$, $\hat{\mu}$. Remember that at the test phase (see (4)), the predicted value is

$$predicted_{S2.UncFA}(x^*) = \hat{\mu}_z + \hat{\Sigma}_z\hat{\Lambda}^\top(\hat{\Lambda}\hat{\Sigma}_z\hat{\Lambda}^\top + \hat{\Psi})^{-1}(x^* - \hat{\mu} - \hat{\Lambda}\hat{\mu}_z), \forall x^* \in \mathbb{R}^{D\times 1}.$$

One can fit a linear regression model and obtain $\hat{w}$, $\hat{b}$ from (1) and (2). Remember that at the test phase, the predicted value is

$$predicted_{LR}(x^*) = \hat{w}x^* + \hat{b}, \forall x^* \in \mathbb{R}^{D\times 1}.$$

Then:

$$predicted_{S2.UncFA}(x^*) = predicted_{LR}(x^*), \forall x^* \in \mathbb{R}^{D\times 1}.$$

Proof. Let $X = \begin{bmatrix} x^{(1)} \dots x^{(n)} \end{bmatrix} \in \mathbb{R}^{D\times n}$ and $Z = \begin{bmatrix} z^{(1)} \dots z^{(n)} \end{bmatrix} \in \mathbb{R}^{d\times n}$.

We begin by computing $\hat{\Psi}$.

$$\begin{aligned}
\hat{\Psi} &\overset{(9)}{=} \frac{1}{n}\sum_{i=1}^{n}(x^{(i)} - \hat{\mu} - \hat{\Lambda}z^{(i)})(x^{(i)} - \hat{\mu} - \hat{\Lambda}z^{(i)})^\top \\
&= \frac{1}{n}\sum_{i=1}^{n}(x^{(i)}x^{(i)^\top} - x^{(i)}\hat{\mu}^\top - x^{(i)}z^{(i)^\top}\hat{\Lambda}^\top - \hat{\mu}x^{(i)^\top} + \hat{\mu}\hat{\mu}^\top + \hat{\mu}z^{(i)^\top}\hat{\Lambda}^\top - \\
&\quad - \hat{\Lambda}z^{(i)}x^{(i)^\top} + \hat{\Lambda}z^{(i)}\hat{\mu}^\top + \hat{\Lambda}z^{(i)}z^{(i)^\top}\hat{\Lambda}^\top) \\
&= \frac{1}{n}XX^\top - \bar{x}\hat{\mu}^\top - \frac{1}{n}XZ^\top\hat{\Lambda}^\top - \hat{\mu}\bar{x}^\top + \hat{\mu}\hat{\mu}^\top + \hat{\mu}\bar{z}^\top\hat{\Lambda}^\top - \frac{1}{n}\hat{\Lambda}ZX^\top + \\
&\quad + \hat{\Lambda}\bar{z}\hat{\mu}^\top + \frac{1}{n}\hat{\Lambda}ZZ^\top\hat{\Lambda}^\top \\
&= \frac{1}{n}XX^\top - \frac{1}{n}XZ^\top\hat{\Lambda}^\top - \frac{1}{n}\hat{\Lambda}ZX^\top + \frac{1}{n}\hat{\Lambda}ZZ^\top\hat{\Lambda}^\top - \bar{x}\hat{\mu}^\top - \hat{\mu}\bar{x}^\top + \hat{\mu}\hat{\mu}^\top + \\
&\quad + \hat{\Lambda}\bar{z}\hat{\mu}^\top + \hat{\mu}\bar{z}^\top\hat{\Lambda}^\top.
\end{aligned}$$

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 $\hat{\Psi}$:

$$\begin{aligned}
\hat{\Psi} &= \frac{1}{n}XX^\top - \frac{1}{n}XZ^\top\hat{\Lambda}^\top - \frac{1}{n}\hat{\Lambda}ZX^\top + \frac{1}{n}\hat{\Lambda}ZZ^\top\hat{\Lambda}^\top - \bar{x}\bar{x}^\top + \cancel{\bar{x}\bar{z}^\top\hat{\Lambda}^\top} - \cancel{\bar{x}\bar{x}^\top} + \\
&\quad + \cancel{\hat{\Lambda}\bar{z}\bar{z}^\top} + \cancel{\bar{x}\bar{x}^\top} - \cancel{\bar{x}\bar{z}^\top\hat{\Lambda}^\top} - \cancel{\hat{\Lambda}\bar{z}\bar{z}^\top} + \cancel{\hat{\Lambda}\bar{z}\bar{z}^\top\hat{\Lambda}^\top} + \hat{\Lambda}\bar{z}\bar{x}^\top - \cancel{\hat{\Lambda}\bar{z}\bar{z}^\top\hat{\Lambda}^\top} + \\
&\quad + \bar{x}\bar{z}^\top\hat{\Lambda}^\top - \hat{\Lambda}\bar{z}\bar{z}^\top\hat{\Lambda}^\top \\
&= \frac{1}{n}XX^\top - \frac{1}{n}XZ^\top\hat{\Lambda}^\top - \frac{1}{n}\hat{\Lambda}ZX^\top + \frac{1}{n}\hat{\Lambda}ZZ^\top\hat{\Lambda}^\top - \bar{x}\bar{x}^\top + \hat{\Lambda}\bar{z}\bar{x}^\top + \\
&\quad + \bar{x}\bar{z}^\top\hat{\Lambda}^\top - \hat{\Lambda}\bar{z}\bar{z}^\top\hat{\Lambda}^\top.
\end{aligned} \tag{11}$$

We continue by computing $\hat{\Lambda}\hat{\Sigma}_z\hat{\Lambda}^\top$.

$$\hat{\Lambda}\hat{\Sigma}_z\hat{\Lambda}^\top \overset{(6)}{=} \hat{\Lambda}\left(\frac{1}{n}ZZ^\top - \bar{z}\bar{z}^\top\right)\hat{\Lambda}^\top = \frac{1}{n}\hat{\Lambda}ZZ^\top\hat{\Lambda}^\top - \hat{\Lambda}\bar{z}\bar{z}^\top\hat{\Lambda}^\top. \tag{12}$$

We observe that the above term (see (12)) is also included in $\hat{\Psi}$ (see (11)).

$$\begin{aligned}\hat{\Sigma}_z\hat{\Lambda}^\top &\overset{(7)}{=} \left(\frac{1}{n}ZZ^\top - \bar{z}\bar{z}^\top\right)(n\bar{z}\bar{z}^\top - ZZ^\top)^{-1}(n\bar{z}\bar{x}^\top - ZX^\top)\\ &= \left(\frac{1}{n}ZZ^\top - \bar{z}\bar{z}^\top\right)\frac{-1}{n}\left(\frac{1}{n}ZZ^\top - \bar{z}\bar{z}^\top\right)^{-1}(n\bar{z}\bar{x}^\top - ZX^\top)\\ &= \frac{1}{n}ZX^\top - \bar{z}\bar{x}^\top. \end{aligned} \tag{13}$$

Since $(\hat{\Lambda}\hat{\Sigma}_z\hat{\Lambda}^\top)^\top = \hat{\Lambda}\hat{\Sigma}_z^\top\hat{\Lambda}^\top = \hat{\Lambda}\hat{\Sigma}_z\hat{\Lambda}^\top \Rightarrow \hat{\Lambda}\hat{\Sigma}_z\hat{\Lambda}^\top$ is symmetric.
We have that:

$$\hat{\Lambda}\hat{\Sigma}_z\hat{\Lambda}^\top = \hat{\Lambda}(\hat{\Sigma}_z\hat{\Lambda}^\top) \overset{(13)}{=} \hat{\Lambda}\left(\frac{1}{n}ZX^\top - \bar{z}\bar{x}^\top\right) = \frac{1}{n}\hat{\Lambda}ZX^\top - \hat{\Lambda}\bar{z}\bar{x}^\top. \tag{14}$$

We also have that:

$$\hat{\Lambda}\hat{\Sigma}_z\hat{\Lambda}^\top = (\hat{\Lambda}\hat{\Sigma}_z\hat{\Lambda}^\top)^\top \overset{(13)}{=} \left(\frac{1}{n}\hat{\Lambda}ZX^\top - \hat{\Lambda}\bar{z}\bar{x}^\top\right)^\top = \frac{1}{n}XZ^\top\hat{\Lambda}^\top - \bar{x}\bar{z}^\top\hat{\Lambda}^\top. \tag{15}$$

We continue by computing $\hat{\Lambda}\hat{\Sigma}_z\hat{\Lambda}^\top + \hat{\Psi}$.

As we have already noticed above, $\hat{\Lambda}\hat{\Sigma}_z\hat{\Lambda}^\top$ is also included in $\hat{\Psi}$ (see (11) and (12)). In the computation of $\hat{\Lambda}\hat{\Sigma}_z\hat{\Lambda}^\top + \hat{\Psi}$, we replace $\hat{\Lambda}\hat{\Sigma}_z\hat{\Lambda}^\top$ once with (14) and then with (15). We get:

$$\begin{aligned}\hat{\Lambda}\hat{\Sigma}_z\hat{\Lambda}^\top + \hat{\Psi} &\underset{(14)(15)}{\overset{(11)(12)}{=}} \frac{1}{n}XX^\top - \cancel{\frac{1}{n}XZ^\top\hat{\Lambda}^\top} - \cancel{\frac{1}{n}\hat{\Lambda}ZX^\top} - \bar{x}\bar{x}^\top + \cancel{\hat{\Lambda}\bar{z}\bar{x}^\top} +\\ &\quad + \cancel{\bar{x}\bar{z}^\top\hat{\Lambda}^\top} + \cancel{\frac{1}{n}\hat{\Lambda}ZX^\top} - \cancel{\hat{\Lambda}\bar{z}\bar{x}^\top} + \cancel{\frac{1}{n}XZ^\top\hat{\Lambda}^\top} - \cancel{\bar{x}\bar{z}^\top\hat{\Lambda}^\top}\\ &= \frac{1}{n}XX^\top - \bar{x}\bar{x}^\top. \end{aligned} \tag{16}$$

Observation: The result is exactly the maximum likelihood estimate of the covariance matrix Σ of the input data set if $x \sim \mathcal{N}(\mu, \Sigma)$: $\Sigma_{\text{MLE}} = \frac{1}{n}XX^\top - \bar{x}\bar{x}^\top$. This is natural because according to the relationship (4) we have $x \sim \mathcal{N}(\mu, \Lambda\Sigma_z\Lambda^\top + \Psi)$, and there are enough free parameters in $\Lambda\Sigma_z\Lambda^\top + \Psi$, i.e., $dD + \frac{d^2-d}{2} + d + \frac{D^2-D}{2} + D$ free parameters—dD in Λ, $\frac{d^2-d}{2}$ in Σ_z, $\frac{D^2-D}{2}$ in Ψ, for it to become $\Sigma_{\text{MLE}} = \frac{1}{n}XX^\top - \bar{x}\bar{x}^\top$, since Σ has $\frac{D^2-D}{2} + D$ free parameters.

We return to the initial computation:

$$\begin{aligned}&\textit{predicted}_{\textit{S2.UncFA}}(x^*) =\\ &\overset{(4)}{=} \hat{\mu}_z + \hat{\Sigma}_z\hat{\Lambda}^\top(\hat{\Lambda}\hat{\Sigma}_z\hat{\Lambda}^\top + \hat{\Psi})^{-1}(x^* - \hat{\mu} - \hat{\Lambda}\hat{\mu}_z)\\ &\overset{(5)(13)(16)}{=} \bar{z} + \left(\tfrac{1}{n}ZX^\top - \bar{z}\bar{x}^\top\right)\left(\tfrac{1}{n}XX^\top - \bar{x}\bar{x}^\top\right)^{-1}(x^* - \bar{x} + \cancel{\hat{\Lambda}\bar{z}} - \cancel{\hat{\Lambda}\bar{z}})\\ &= \left(\tfrac{1}{n}ZX^\top - \bar{z}\bar{x}^\top\right)\left(\tfrac{1}{n}XX^\top - \bar{x}\bar{x}^\top\right)^{-1}x^* + \bar{z} - \left(\tfrac{1}{n}ZX^\top - \bar{z}\bar{x}^\top\right)\left(\tfrac{1}{n}XX^\top - \bar{x}\bar{x}^\top\right)^{-1}\bar{x}\\ &= \left(n\bar{z}\bar{x}^\top - ZX^\top\right)\left(n\bar{x}\bar{x}^\top - XX^\top\right)^{-1}x^* + \bar{z} - \left(n\bar{z}\bar{x}^\top - ZX^\top\right)\left(n\bar{x}\bar{x}^\top - XX^\top\right)^{-1}\bar{x}\\ &\overset{(1)}{=} \hat{w}x^* + \bar{z} - \hat{w}\bar{x}\\ &\overset{(2)}{=} \hat{w}x^* + \hat{b}\\ &= \textit{predicted}_{LR}(x^*), \forall x^* \in \mathbb{R}^{D\times 1}. \quad \square \end{aligned}$$

4.1.3. The S2.FA Model. A New Approach for LR When $D >> n$

Since FA can be used to estimate the density of a data set when $D >> n$, and S2.UncFA is equivalent to LR, we consider S2.FA as a new approach to extend LR when $D >> n$ besides the two solutions mentioned in Section 2.

4.2. The S3.FA Model

Factor analysis is a classic generative unsupervised model. Its supervised counterpart is S2.FA as shown in the previous subsection. Those two can be merged into a semisupervised model that we propose here, named simple-semisupervised factor analysis:

$z \sim \mathcal{N}(\mu_z, \sigma_z^2)$-either observed or latent variable, $z \in \mathbb{R}$, $\mu_z \in \mathbb{R}$, $\sigma_z \in \mathbb{R}_+^*$

$x|z \sim \mathcal{N}(\mu + \Lambda z, \Psi)$, $x \in \mathbb{R}^{D\times 1}$, $\mu \in \mathbb{R}^{D\times 1}$, $\Lambda \in \mathbb{R}^{D\times 1}$, and $\Psi \in \mathbb{R}^{D\times D}$ a diagonal matrix.

If we were to speak about Gaussian naive Bayes and the GMM, good hints for combining these two supervised–unsupervised counterparts into a semisupervised model can be found in [12]. We applied those hints for our supervised–unsupervised counterparts—S2.FA and FA—and created an EM algorithm to fit an S3.FA model. For the step-by-step algorithm and the matrix form of the algorithm, see Appendix C. For more elaborate notations and the proof, see [8] pp. 26–34.

4.3. The MS3.FA Model

The algorithm that fits an S3.FA model can be adapted also for the case when not all the components of x are known. We call the resulted model missing simple-semisupervised factor analysis:

$z \sim \mathcal{N}(\mu_z, \sigma_z^2)$-either observed or latent variable, $z \in \mathbb{R}$, $\mu_z \in \mathbb{R}$, $\sigma_z \in \mathbb{R}_+^*$

$x|z \sim \mathcal{N}(\mu + \Lambda z, \Psi)$, $x \in \mathbb{R}^{D\times 1}$, $\mu \in \mathbb{R}^{D\times 1}$, $\Lambda \in \mathbb{R}^{D\times 1}$, and $\Psi \in \mathbb{R}^{D\times D}$ a diagonal matrix; each component of x: $x_1, \ldots, x_D$ is either observed or latent.

The resulting algorithm that fits a MS3.FA model is an EM algorithm. For the step-by-step algorithm, see Appendix D. For more elaborate notations and the proof, see [8] pp. 34–37.

5. Experiments

In this section, we include the experiments we carried out on data with $D >> n$ using the S2.FA, S3.FA, and MS3.FA models, comparing them with other methods. In all the experiments, we computed errors between the real values and the predicted values; the metric we used is mean squared error (MSE):

$$\text{MSE} = \frac{1}{N}\sum_{i=1}^{N}(\text{real}_i - \text{predicted}_i)^2,$$

where N is the number of the unknown elements whose real and predicted values are real_i and predicted_i, respectively; an unknown element represents an output number for S2.FA and S3.FA or an input/output number for MS3.FA. We ran each experiment five times and computed a 95% confidence interval using the t-distribution. Furthermore, in each experiment we used the same three data sets:

- Gas sensor array under flow modulation data set (http://archive.ics.uci.edu/ml/datasets/Gas+sensor+array+under+flow+modulation; accessed on 31 July 2021) [13]: 58 observations, 432 input attributes;
- atp1d—the airline ticket price; 1D refers to the fact that the target price is in the next day—(https://www.openml.org/d/41475; accessed on 31 July 2021) [14]: 337 observations, 411 input attributes; 370 after preprocessing: see below;
- m5spec—corn measured on a NIR spectrometer: mp5 instrument—(http://www.eigenvector.com/data/Corn; accessed on 31 July 2021): 80 observations, 700 input attributes.

All three of these data sets have multiple outputs, but for each data set, we selected only the first output column that appears in the text data file and used it as the output: *ace_conc* for gas sensor array under flow modulation data set, *LBL_ALLminpA_fut_001* for atp1d, the first column in the *propvals* file for m5spec.

We preprocessed each data set simply by dropping the constant columns. Only the second data set has constant columns: from 432 columns, we obtain 370.

5.1. The S2.FA Model: Experiment

The experiment concerning S2.FA covers the comparison of the three solutions presented so far for LR when $D >> n$:

- Moore–Penrose inverse
- ridge regression—L2 regularization
- S2.FA.

Each data set was split into a training part—80%—and a testing part—20%. If the model had hyperparameters, ridge regression, the training part was also split into a new training part—60% of the whole data set—and a validation part—20% of the whole data set—in order to be able to set the hyperparameters (for ridge regression, we used a simple technique: pick $\alpha \in \{10^2, 10^{1.9}, 10^{1.8}, \ldots, 10^{-1.9}, 10^{-2}\}$, which attains the minimum validation error); after setting the hyperparameters, we train a new model on the initial training part—80% of the whole data set—and obtain the final model. All of the MSE errors are reported on the testing part and shown in Table 1.

As one may notice, the best method is different for each data set, so our general advice is to use all the methods on a given data set and pick the best one.

Table 1. Simple-supervised factor analysis (S2.FA) experiment: mean squared error (MSE) 95% confidence intervals on three data sets using three methods for regression when $D >> n$; the best MSE means are marked in bold.

Data Set/Method	Moore–Penrose	Ridge Regression	S2.FA
Gas sensor array under flow modulation	0.0251 ± 0.0254	**0.0062** ± 0.007	0.0452 ± 0.0208
atp1d	94,627.7239 ± 80,183.0076	27,770.9253 ± 42,887.5216	**4724.2957** ± 1616.3341
m5spec	**0.00004** ± 0.00001	0.02676 ± 0.01372	0.37344 ± 0.25025

5.2. The S3.FA Model: Experiment

The experiment concerning S3.FA includes an analysis of algorithms for semisupervised regression when $D >> n$:

- Moore–Penrose inverse—a supervised method
- S2.FA—a supervised method
- S3.FA—a semisupervised method
- label propagation [15]—a semisupervised method: we used the function `sslLabelProp` in the SSL R package [16] with the parameter `alpha` set to 1.

Each data set was split into a training part—80%—and a testing part—20%. We retained from the training part 5%, 10%, 15%, 20%, 25%, . . . , 100% of the output labels. For the supervised methods, we used only the labeled data in the training set, and for the semisupervised methods, we used the full training set when fitting the model. We initialized the S3.FA method with the fitted parameters returned by the S2.FA algorithm. All of the MSE errors are reported on the testing part and shown in Figures 1–3—inspired from [17]—and Table 2. The figures contain all the results from using 5% to 100% of the output labels, but we include less information in the table for brevity.

We notice that on the selected data sets, S3.FA returns poorer results even than S2.FA, which uses only the labeled data. As in the previous experiment, the best method is also data-dependent. The models that show a greater amount of variability compared to the others are S3.FA—in two data sets—and Moore–Penrose—in one data set. Moreover, as expected, the errors tend to decrease as the percentage of labeled training data points increases; the plots do not help us in this regard, but this decrease can be seen numerically in Table 2.

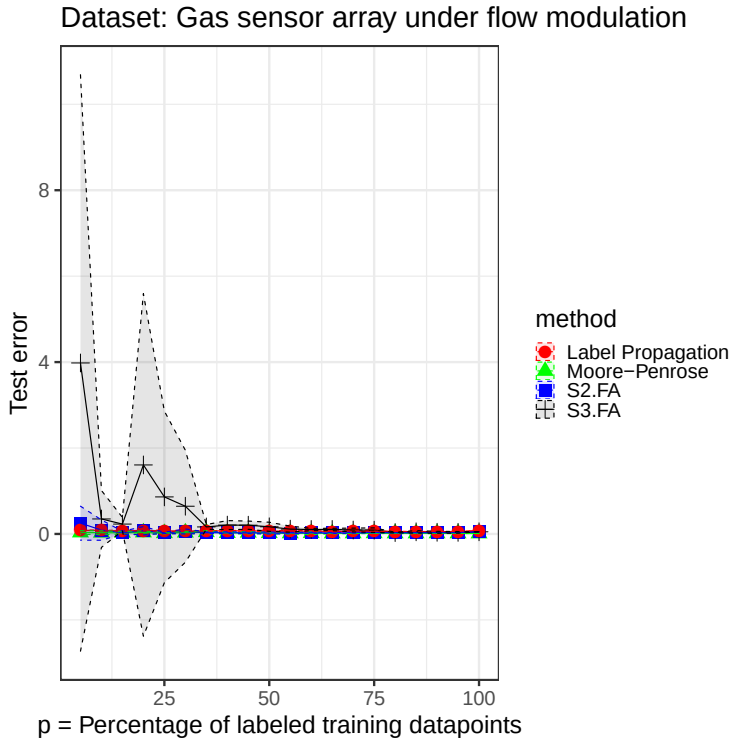

Figure 1. Simple-semisupervised factor analysis (S3.FA) experiment: MSE 95% confidence intervals on the gas sensor array under flow modulation data set using four methods for semisupervised regression when $D >> n$; $p \in \{5\%, 10\%, 15\%, \ldots, 100\%\}$ of the training output labels are retained.

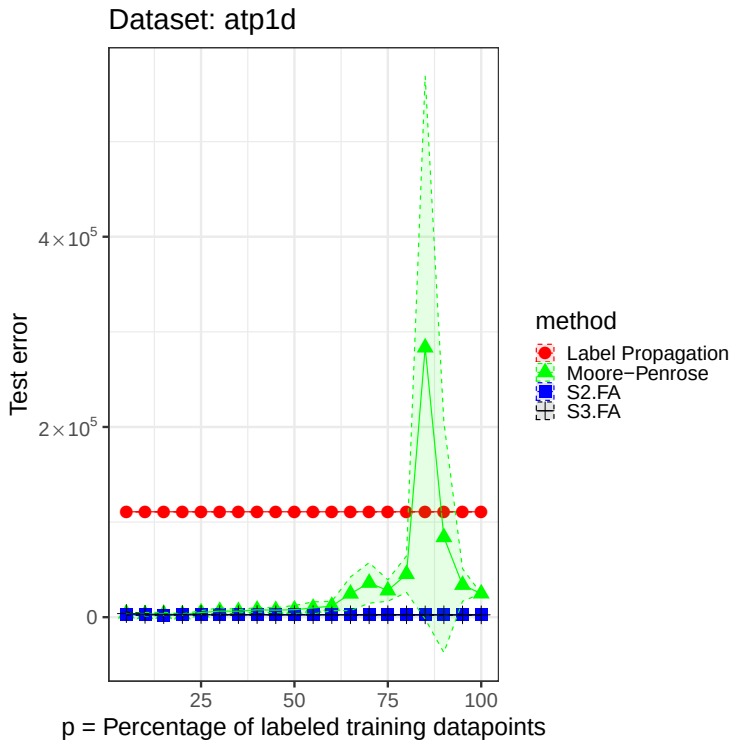

Figure 2. S3.FA experiment: MSE 95% confidence intervals on the *atp1d* data set using four methods for semisupervised regression when $D >> n$; $p \in \{5\%, 10\%, 15\%, \ldots, 100\%\}$ of the training output labels are retained.

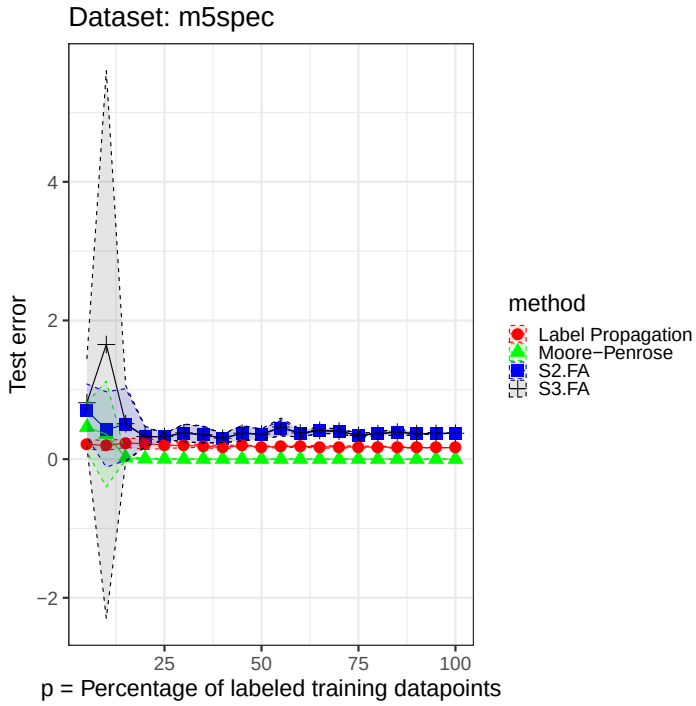

Figure 3. S3.FA experiment: MSE 95% confidence intervals on the *m5spec* data set using four methods for semisupervised regression when $D >> n$; $p \in \{5\%, 10\%, 15\%, \ldots, 100\%\}$ of the training output labels are retained.

Table 2. S3.FA experiment: MSE 95% confidence intervals on three data sets using four methods for semisupervised regression when $D >> n$; $p \in \{5\%, 10\%, 15\%, 30\%, 50\%, 70\%\}$ of the training output labels are retained; the best MSE means are marked in bold.

Data Set	Method	$p = 5\%$	$p = 10\%$	$p = 15\%$
Gas sensor array under flow modulation	Moore–Penrose	**0.034** ± 0.0437	**0.0292** ± 0.0421	**0.0267** ± 0.0326
	S2.FA	0.2511 ± 0.3953	0.0899 ± 0.2367	0.0285 ± 0.0333
	S3.FA	3.9799 ± 6.7087	0.349 ± 0.6626	0.2294 ± 0.1561
	Label Propagation	0.0853 ± 0.0073	0.0825 ± 0.0051	0.0708 ± 0.013
atp1d	Moore–Penrose	3763.2939 ± 749.0966	3079.5042 ± 1842.7676	3428.7567 ± 1722.6437
	S2.FA	**2706.9906** ± 477.3245	**2339.3504** ± 324.8581	**2279.0802** ± 99.5443
	S3.FA	3771.605 ± 1563.6543	2972.1137 ± 639.6724	2633.0816 ± 274.7409
	Label Propagation	110,820.3235 ± 0	110,820.3235 ± 0	110,820.3235 ± 0
m5spec	Moore–Penrose	0.4602 ± 0.354	0.3609 ± 0.7631	**0.0187** ± 0.0221
	S2.FA	0.7003 ± 0.3834	0.4269 ± 0.5441	0.4999 ± 0.5172
	S3.FA	0.8133 ± 0.6366	1.653 ± 3.9497	0.5118 ± 0.5503
	Label Propagation	**0.2175** ± 0.0707	**0.1939** ± 0.0706	0.2343 ± 0.0624
Data Set	**Method**	$p = 30\%$	$p = 50\%$	$p = 70\%$
Gas sensor array under flow modulation	Moore–Penrose	**0.0313** ± 0.0259	**0.0192** ± 0.0145	**0.0132** ± 0.0075
	S2.FA	0.0588 ± 0.0576	0.0233 ± 0.0224	0.0279 ± 0.0116
	S3.FA	0.645 ± 1.3044	0.1666 ± 0.1059	0.0912 ± 0.0443
	Label Propagation	0.0668 ± 0.0086	0.0675 ± 0.018	0.0605 ± 0.0122
atp1d	Moore–Penrose	6401.7587 ± 3602.9182	7907.484 ± 4449.2312	36,041.874 ± 21,300.704
	S2.FA	**2444.9001** ± 404.2393	**2439.162** ± 108.3605	**2399.7948** ± 160.1038
	S3.FA	2760.6602 ± 391.9217	2659.6472 ± 112.9795	2521.9861 ± 222.3765
	Label Propagation	110,820.3235 ± 0	110,820.3235 ± 0	110,820.3235 ± 0
m5spec	Moore–Penrose	**0.00057** ± 0.00085	**0.00009** ± 0.00008	**0.00009** ± 0.00004
	S2.FA	0.37449 ± 0.12508	0.35796 ± 0.07275	0.3995 ± 0.03164
	S3.FA	0.37865 ± 0.12808	0.36181 ± 0.07463	0.40419 ± 0.03415
	Label Propagation	0.19391 ± 0.02754	0.17424 ± 0.00857	0.17752 ± 0.01545

5.3. The MS3.FA Model: Experiment

The experiment concerning MS3.FA includes a comparison of two different types of algorithms for data imputation when $D >> n$:

- Mean imputation: for a given attribute (input column), compute its mean ignoring the missing values, then replace the missing data on that attribute with this computed mean
- MS3.FA.

We also tried two other R packages: mice [18] and Amelia [19], but they could not be applied successfully on our data sets perhaps because they have a peculiarity: $D >> n$.

For each data set, we removed 10%, 20%, 30%, 40%, 50%, and 60% of the input (We could have added missing data also in the output, but we wanted to focus on the missing input data scenario and not on the semisupervised case.) cells and imputed those via the above mentioned algorithms. The results are presented in Figures 4–6 and in Table 3.

From these results, we discover that MS3.FA is better than mean imputation on two data sets, and, as expected, the error increases as the percentage of missing data increases.

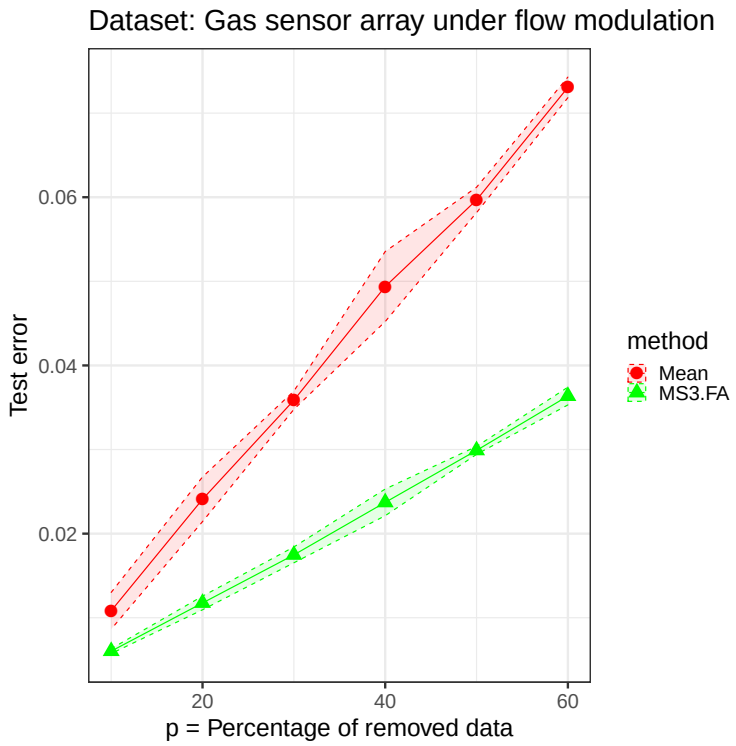

Figure 4. Missing simple-semisupervised factor analysis (MS3.FA) experiment: MSE 95% confidence intervals on the gas sensor array under flow modulation data set using two methods for imputing missing data when $D >> n$; $p \in \{10\%, 20\%, 30\%, 40\%, 50\%, 60\%\}$ of the input data are removed.

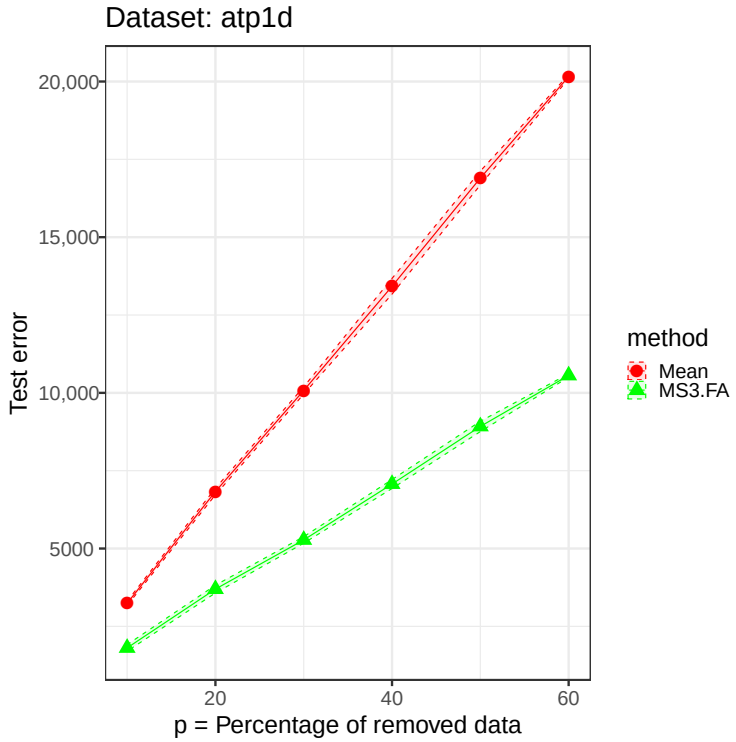

Figure 5. MS3.FA experiment: MSE 95% confidence intervals on the *atp1d* data set using two methods for imputing missing data when $D >> n$; $p \in \{10\%, 20\%, 30\%, 40\%, 50\%, 60\%\}$ of the input data are removed.

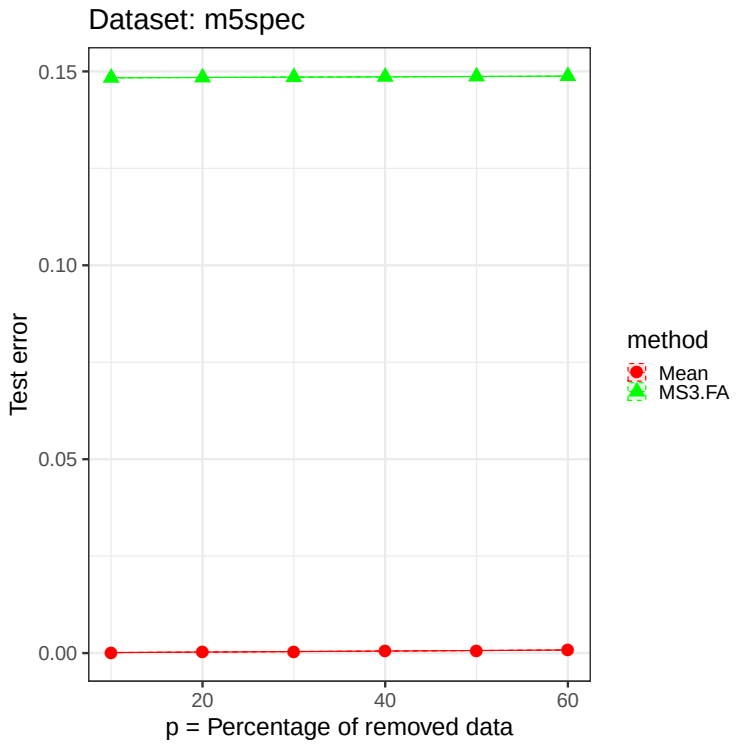

Figure 6. MS3.FA experiment: MSE 95% confidence intervals on the *m5spec* data set using two methods for imputing missing data when $D >> n$; $p \in \{10\%, 20\%, 30\%, 40\%, 50\%, 60\%\}$ of the input data are removed.

Table 3. MS3.FA experiment: MSE 95% confidence intervals on three data sets using two methods for imputing missing data when $D >> n$; $p \in \{10\%, 20\%, 30\%, 40\%, 50\%, 60\%\}$ of the input data are removed; the best MSE means are marked in bold.

Data Set	Method	$p = 10\%$	$p = 20\%$	$p = 30\%$
Gas sensor array under flow modulation	Mean	0.0108 ± 0.0022	0.0241 ± 0.0027	0.0359 ± 0.0011
	MS3.FA	**0.00603** ± 0.00029	**0.01176** ± 0.00081	**0.01747** ± 0.00094
atp1d	Mean	3239.9757 ± 68.2511	6831.0884 ± 105.6119	10,077.1798 ± 117.4693
	MS3.FA	**1807.7748** ± 105.03	**3697.1146** ± 124.2197	**5276.899** ± 104.6252
m5spec	Mean	**0.00013** ± 0.00001	**0.00026** ± 0	**0.00039** ± 0.00001
	MS3.FA	0.14835 ± 0.000002	0.148433 ± 0.000002	0.148518 ± 0.000002
Data Set	**Method**	$p = 40\%$	$p = 50\%$	$p = 60\%$
Gas sensor array under flow modulation	Mean	0.0494 ± 0.0042	0.0597 ± 0.0015	0.0731 ± 0.0012
	MS3.FA	**0.02372** ± 0.00159	**0.0299** ± 0.00047	**0.0364** ± 0.0011
atp1d	Mean	13,423.3625 ± 252.7735	16,899.3166 ± 223.6971	20,156.2493 ± 62.1299
	MS3.FA	**7071.6793** ± 143.3232	**8921.5319** ± 165.5759	**10,558.5723** ± 53.1182
m5spec	Mean	**0.000521** ± 0.000005	**0.000658** ± 0.000008	**0.00080** ± 0.000005
	MS3.FA	0.1486 ± 0.00001	0.14869 ± 0.00001	0.148781 ± 0.000001

6. Conclusions and Future Work

The initial purpose of this paper was to extend an already existing model: factor analysis. We developed its supervised counterpart (S2.FA) and noticed that the unconstrained version (S2.UncFA) is equivalent to linear regression. Because FA is applied in density estimation when the dimensionality of the data is greater than the number of samples, and because of the already mentioned equivalence, the purpose of the paper became to analyze this new method of applying LR when $D >> n$, i.e., via S2.FA. Since FA and S2.FA are generative models and are unsupervised–supervised counterparts, we combined both into a new model S3.FA as an extension of LR to semisupervised learning when $D >> n$. The final extension regards missing data; it is called MS3.FA. We developed an R package (s2fa) with these algorithms; it can be found on GitHub. The experimental parts included several comparisons in the $D >> n$ scenario:

- of S2.FA with other techniques extending LR to the $D >> n$ case,
- of S3.FA with other (semi)supervised regression methods,
- of MS3.FA with another data imputation algorithm.

The bottom line is that we do not necessarily recommend S3.FA for semisupervised regression since our results suggest that it gives poor results, but we encourage the consideration of S2.FA for regression and MS3.FA for missing data imputation as algorithms to be compared with others on a given data set.

As for future work, we could further explore the S2.FA, S3.FA, and MS3.FA algorithms when z is a real vector, not just a real number. Moreover, we can experiment with the PPCA version of the algorithms. Questions regarding the time complexity—empirical or not—can be also addressed; we expect the fitting time to be impractical if the number of columns is large. Because there are models such as mixture of factor analyzers [20] and mixture of linear regression models ([21] Section 14.5.1), another research direction involves mixtures of S2.FAs. Another idea would be to investigate the memory resources required by the algorithms when the data set increases and also to consider scalable systems such as Spark [22] for implementation.

Author Contributions: Conceptualization, S.C. and L.C.; methodology, S.C. and L.C.; software, S.C.; validation, S.C.; formal analysis, S.C.; investigation, S.C.; resources, L.C.; data curation; writing—original draft preparation, S.C.; writing—review and editing, S.C. and L.C.; visualization, S.C. and L.C.; supervision, L.C.; project administration, S.C. and L.C.; funding acquisition. Both authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Publicly available data sets were analyzed in this study. This data can be found here: http://archive.ics.uci.edu/ml/datasets/Gas+sensor+array+under+flow+modulation (accessed on 31 July 2021) for the gas sensor array under flow modulation data set, https://www.openml.org/d/41475 (accessed on 31 July 2021) for atp1d, http://www.eigenvector.com/data/Corn (accessed on 31 July 2021) for m5spec.

Acknowledgments: We thank Cristian Gaţu and Daniel Stamate for attentive proofreading and useful comments of previous versions of this paper.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

GMM	Gaussian mixture model
LR	Linear regression
EM	Expectation–maximization
KL	Kullback–Leibler
FA	Factor analysis
PCA	Principal component analysis
PPCA	Probabilistic principal component analysis
GPLVM	Gaussian process latent variable model
MLE	Maximum likelihood estimation
UncFA	Unconstrained factor analysis
S2.UncFA	Simple-supervised unconstrained factor analysis
S2.FA	Simple-supervised factor analysis
S2.PPCA	Simple-supervised probabilistic principal component analysis
S3.UncFA	Simple-semisupervised unconstrained factor analysis
S3.FA	Simple-semisupervised factor analysis
S3.PPCA	Simple-semisupervised probabilistic principal component analysis
MS3.UncFA	Missing simple-semisupervised unconstrained factor analysis
MS3.FA	Missing simple-semisupervised factor analysis
MS3.PPCA	Missing simple-semisupervised probabilistic principal component analysis
MSE	Mean squared error
ELBO	Evidence lower bound

Appendix A. On the Expectation–Maximization Algorithm

This section provides theoretical details on the EM algorithm [23]. These are relevant in order to establish a link between EM and the information theory field.

A latent variable model assumes that the data we observed—usually, denoted by the random variable X—is not complete: there is also some latent data modeled via random variables—usually, denoted by Z. Often, pairs consisting of an observed point and a latent one constitute the complete dataset, e.g., in a GMM the latent data is the cluster number and such a number is assigned to each point in the observed dataset.

Usually, in a latent variable model the likelihood of the observed data is not tractable—although there are exceptions, like in GMM or factor analysis—and therefore we cannot maximize it directly. Instead we maximize a lower bound for the log-likelihood function called ELBO (evidence lower bound). To simplify the discussion, we consider that we have one single observed datapoint, x. We will also consider the case where Z is continuous; when Z is discrete the $\int$ sign is replaced by the $\sum$ sign.

Let q be any distribution over z, where the z values are the possible values of the random variable Z. The log-likelihood of the observed datapoint x is:

$$\begin{aligned}\log p(x;\theta) &\overset{\text{marginal}}{=} \log \int_z p(x,z;\theta)dz \\ &= \log \int_z \frac{q(z)}{q(z)} p(x,z;\theta)dz \\ &= \log \int_z q(z) \frac{p(x,z;\theta)}{q(z)} dz \\ &= \log \mathbb{E}_q \left[\frac{p(x,z;\theta)}{q(z)} \right] \\ &\overset{\text{Jensen}}{\geq} \underbrace{\mathbb{E}_q \left[\log \frac{p(x,z;\theta)}{q(z)} \right]}_{\text{ELBO}(\theta,q)}.\end{aligned}$$

Furthermore, the following relationships important for the E step of the EM algorithm—see below—can be proven:

$$\begin{aligned}\log p(x;\theta) - \text{ELBO}(\theta,q) &= \text{KL}(q(\cdot)||p(\cdot|x;\theta)) \\ &\Updownarrow \\ \log p(x;\theta) &= \text{ELBO}(\theta,q) + \text{KL}(q(\cdot)||p(\cdot|x;\theta)) \\ &\Updownarrow \\ \text{ELBO}(\theta,q) &= \log p(x;\theta) - \text{KL}(q(\cdot)||p(\cdot|x;\theta)).\end{aligned}$$

Moreover, the following relationships important for the M step of the EM algorithm—see below—can be proven:

$$\begin{aligned}\text{ELBO}(\theta,q) &\overset{\text{def.}}{=} \mathbb{E}_q \left[\log \frac{p(x,z;\theta)}{q(z)} \right] \\ &= \int_z q(z) \log \frac{p(x,z;\theta)}{q(z)} dz \\ &= \int_z q(z) \log p(x,z;\theta) dz - \int_z q(z) \log q(z) dz \\ &= \mathbb{E}_q[\log p(x,z;\theta)] + H(q).\end{aligned}$$

Now, instead of carrying out $\max_\theta \log p(x;\theta)$ we will execute $\max_{\theta,q} \text{ELBO}(\theta,q)$, since ELBO is a lower bound for the log-likelihood of x and hence its maximization will not hurt the process of maximizing $\log p(x;\theta)$.

The ELBO can be maximized in at least two ways:

- via (block) coordinate ascent
  The resulting meta-algorithm is the EM algorithm.
  In fact this is the case for many classic models—EM for GMM [3], EM for factor analysis [5] etc.—.
  EM is an iterative algorithm and an iteration encompasses two steps:
  1. E step:
     $q^{(t)} = \arg\max_q \text{ELBO}(\theta^{(t-1)}, q)$ for $\theta^{(t-1)}$ fixed—from the previous iteration.
     Since $\text{ELBO}(\theta,q) = \log p(x;\theta) - \text{KL}(q(\cdot)||p(\cdot|x;\theta))$, we have:

$$\begin{aligned} q^{(t)} &= \arg\max_q \text{ELBO}(\theta^{(t-1)}, q) \\ &= \arg\max_q (\log p(x;\theta^{(t-1)}) - \text{KL}(q(\cdot)||p(\cdot|x;\theta^{(t-1)}))) \\ &= \arg\min_q \text{KL}(q(\cdot)||p(\cdot|x;\theta^{(t-1)})) \overset{\text{KL property}}{=} p(\cdot|x;\theta^{(t-1)}). \end{aligned}$$

(In this case, we have $\log p(x;\theta^{(t-1)}) = \text{ELBO}(\theta^{(t-1)}; q^{(t)})$.)
So, we obtained the distribution $q^{(t)}$ as a posterior distribution. Although in classic models where conjugate priors are used it is tractable to compute $p(\cdot|x;\theta^{(t-1)})$—this type of inference is called analytical inference—, in other models this is not the case and a solution to this shortcoming is represented by approximate/variational inference.

2. M step:
$\theta^{(t)} = \arg\max_\theta \text{ELBO}(\theta, q^{(t)})$ for $q^{(t)}$ fixed—from the E step.
Since $\text{ELBO}(\theta, q) = \mathbb{E}_q[\log p(x, z;\theta)] + H(q)$, we have:

$$\begin{aligned} \theta^{(t)} &= \arg\max_\theta \text{ELBO}(\theta, q^{(t)}) \\ &= \arg\max_\theta (\mathbb{E}_{q^{(t)}}[\log p(x, z;\theta)] + H(q^{(t)})) \\ &= \arg\max_\theta \mathbb{E}_{q^{(t)}}[\log p(x, z;\theta)]. \end{aligned}$$

So, we obtained a relatively simpler term to maximize. Note that the maximization is further customized using the probabilistic assumptions at hand.

- via gradient ascent: this is the case of Variational Autoencoder [24] which will not be discussed since it is not necessarily relevant to this study.

Appendix B. S2.FA

Algorithm A1 S2.FA—nonmatrix form.

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

Algorithm A2 S2.FA—matrix form.

function TRAIN(X,Z) ▷ $X = \begin{bmatrix} x^{(1)} & \dots & x^{(n)} \end{bmatrix}$

▷ $Z = \begin{bmatrix} z^{(1)} & \dots & z^{(n)} \end{bmatrix}$

$\bar{x} = \frac{\sum_{i=1}^{n} X_{:i}}{n}$

$\hat{\mu}_z = \frac{\sum_{i=1}^{n} Z_{:i}}{n} = \bar{z}$

$\hat{\Sigma}_z = \frac{1}{n}\left(Z - \hat{\mu}_z \underbrace{\begin{bmatrix} 1 & 1 & \dots & 1 \end{bmatrix}}_{\in \mathbb{R}^{1\times n}}\right)\left(Z - \hat{\mu}_z \begin{bmatrix} 1 & 1 & \dots & 1 \end{bmatrix}\right)^\top = \frac{1}{n} Z Z^\top - \hat{\mu}_z \hat{\mu}_z^\top$ ▷ 2 ways to compute

$\hat{\Lambda} = (n\bar{x}\hat{\mu}_z^\top - X Z^\top)(n\hat{\mu}_z\hat{\mu}_z^\top - Z Z^\top)^{-1}$

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

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

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)

Appendix C. S3.FA

The algorithms below are more general: z is not just a real number, as we state in the paper, but a real vector.

Algorithm A3 S3.FA—nonmatrix form.

function LOGLIKELIHOOD($\{(x^{(1)}, z^{(1)}), \ldots, (x^{(a)}, z^{(a)}), x^{(a+1)}, \ldots, x^{(n)}\}$, $(\mu_z, \Sigma_z, \mu, \Lambda, \Psi)$)
  return $\sum_{i=1}^{a}\left(\ln\left(\mathcal{N}(z^{(i)}|\mu_z, \Sigma_z)\right) + \ln\left(\mathcal{N}(x^{(i)}|\mu + \Lambda z^{(i)}, \Psi)\right)\right) + \sum_{i=a+1}^{n} \ln\left(\mathcal{N}(x^{(i)}|\mu + \Lambda\mu_z, \Lambda\Sigma_z\Lambda^\top + \Psi)\right)$
function TRAIN($\{(x^{(1)}, z^{(1)}), \ldots, (x^{(a)}, z^{(a)}), x^{(a+1)}, \ldots, x^{(n)}\}$,nMaxIterations,eps)
  $\bar{x} = \frac{\sum_{i=1}^{n} x^{(i)}}{n}$
  $\theta^{(0)} =$ initializeParameters($\{(x^{(1)}, z^{(1)}), \ldots, (x^{(a)}, z^{(a)}), x^{(a+1)}, \ldots, x^{(n)}\}$)
  $l_{\text{RV\_Do}}^{(0)} =$ LOGLIKELIHOOD($\{(x^{(1)}, z^{(1)}), \ldots, (x^{(a)}, z^{(a)}), x^{(a+1)}, \ldots, x^{(n)}\}$,$\theta^{(0)}$)
  for t = 0:nMaxIterations **do**
    E step: Compute $E[Z^{(i)}]$, $E[Z^{(i)}{Z^{(i)}}^\top], i \in \{a+1, \ldots, n\}$:
    $E_{Z^{(i)}|X^{(i)}=x^{(i)},\theta^{(t)}}[Z^{(i)}] = \mu_z^{(t)} + \Sigma_z^{(t)}{\Lambda^{(t)}}^\top(\Lambda^{(t)}\Sigma_z^{(t)}{\Lambda^{(t)}}^\top + \Psi^{(t)})^{-1}(x^{(i)} - \mu^{(t)} - \Lambda^{(t)}\mu_z^{(t)})$
    $E_{Z^{(i)}|X^{(i)}=x^{(i)},\theta^{(t)}}[Z^{(i)}{Z^{(i)}}^\top] = \Sigma_z^{(t)} + E_{Z^{(i)}|X^{(i)}=x^{(i)},\theta^{(t)}}[Z^{(i)}]E_{Z^{(i)}|X^{(i)}=x^{(i)},\theta^{(t)}}[Z^{(i)}]^\top$
    M Step: Compute $\theta^{(t+1)} = (\mu_z^{(t+1)}, \Sigma_z^{(t+1)}, \mu^{(t+1)}, \Lambda^{(t+1)}, \Psi^{(t+1)})$:
    $\mu_z^{(t+1)} = \frac{\sum_{i=1}^{a} z^{(i)} + \sum_{a+1}^{n} E[Z^{(i)}]}{n}$
    $\Sigma_z^{(t+1)} = \frac{1}{n}(\sum_{i=1}^{a}(z^{(i)} - \mu_z^{(t+1)})(z^{(i)} - \mu_z^{(t+1)})^\top + \sum_{i=a+1}^{n}(E[Z^{(i)}{Z^{(i)}}^\top] - E[Z^{(i)}]{\mu_z^{(t+1)}}^\top - \mu_z^{(t+1)}E[Z^{(i)}]^\top + \mu_z^{(t+1)}{\mu_z^{(t+1)}}^\top))$
    $\Lambda^{(t+1)} = \left(n\bar{x}{\mu_z^{(t+1)}}^\top - \sum_{i=1}^{a} x^{(i)}{z^{(i)}}^\top - \sum_{i=a+1}^{n} x^{(i)}E[Z^{(i)}]^\top\right)$
    $\left(n\mu_z^{(t+1)}{\mu_z^{(t+1)}}^\top - \sum_{i=1}^{a} z^{(i)}{z^{(i)}}^\top - \sum_{i=a+1}^{n} E[Z^{(i)}{Z^{(i)}}^\top]\right)^{-1}$
    $\mu^{(t+1)} = \bar{x} - \Lambda^{(t+1)}\mu_z^{(t+1)}$
    $\Psi^{(t+1)} = \text{diag}(\frac{1}{n}(\sum_{i=1}^{a}(x^{(i)} - \mu^{(t+1)} - \Lambda^{(t+1)}z^{(i)})(x^{(i)} - \mu^{(t+1)} - \Lambda^{(t+1)}z^{(i)})^\top + \sum_{i=a+1}^{n}((x^{(i)} - \mu^{(t+1)})(x^{(i)} - \mu^{(t+1)})^\top - (x^{(i)} - \mu^{(t+1)})E[Z^{(i)}]^\top{\Lambda^{(t+1)}}^\top - \Lambda^{(t+1)}E[Z^{(i)}](x^{(i)} - \mu^{(t+1)})^\top + \Lambda^{(t+1)}E[Z^{(i)}{Z^{(i)}}^\top]{\Lambda^{(t+1)}}^\top)))$
    $l_{\text{RV\_Do}}^{(t+1)} =$ LOGLIKELIHOOD($\{(x^{(1)}, z^{(1)}), \ldots, (x^{(a)}, z^{(a)}), x^{(a+1)}, \ldots, x^{(n)}\}$,$\theta^{(t+1)}$)
    if $\frac{\|\theta^{(t)} - \theta^{(t+1)}\|_2^2}{\|\theta^{(t)}\|_2^2} \leq$eps **or** $\frac{|l_{\text{RV_Do}}^{(t)} - l_{\text{RV_Do}}^{(t+1)}|}{|l_{\text{RV_Do}}^{(t)}|} \leq$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 *S2.FA*
  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)

Algorithm A4 S3.FA—matrix form.

function LOGLIKELIHOOD(X,Z,$(\mu_z, \Sigma_z, \mu, \Lambda, \Psi)$) ▷ $X = \begin{bmatrix} x^{(1)} & \dots & x^{(n)} \end{bmatrix}$

▷ $Z = \begin{bmatrix} z^{(1)} & \dots & z^{(a)} \end{bmatrix}$

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

function TRAIN(X,Z,nMaxIterations,eps) ▷ $X = \begin{bmatrix} x^{(1)} & \dots & x^{(n)} \end{bmatrix}$

▷ $Z = \begin{bmatrix} z^{(1)} & \dots & z^{(a)} \end{bmatrix}$

  $\bar{x} = \frac{\sum_{i=1}^{n} X_{:i}}{n}$

  $\theta^{(0)} = \text{initializeParameters}(X, Z)$

  $l_{\text{RV\_Do}}^{(0)} = \text{LOGLIKELIHOOD}(\text{X,Z,}\theta^{(0)})$

  for t = 0:nMaxIterations **do**

    E step: Compute $E[Z^{(i)}]$, $E[Z^{(i)}{Z^{(i)}}^\top], i \in \{a+1, \dots, n\}$:

    $E\_Z = \mu_z^{(t)} + \Sigma_z^{(t)}{\Lambda^{(t)}}^\top(\Lambda^{(t)}\Sigma_z^{(t)}{\Lambda^{(t)}}^\top + \Psi^{(t)})^{-1}(X_{:,(a+1):n} - \mu^{(t)}\underbrace{\begin{bmatrix}1 & 1 & \dots & 1\end{bmatrix}}_{\in\mathbb{R}^{1\times(n-a)}} - \Lambda^{(t)}\mu_z^{(t)}\underbrace{\begin{bmatrix}1 & 1 & \dots & 1\end{bmatrix}}_{\in\mathbb{R}^{1\times(n-a)}})$

    $E\_Z\_Z\_T = \Sigma_z^{(t)} + E\_ZE\_Z^\top$

    $E\_Z = \begin{bmatrix} Z & E\_Z \end{bmatrix}$

    $E\_Z\_Z\_T = ZZ^\top + E\_Z\_Z\_T$

    M Step: Compute $\theta^{(t+1)} = (\mu_z^{(t+1)}, \Sigma_z^{(t+1)}, \mu^{(t+1)}, \Lambda^{(t+1)}, \Psi^{(t+1)})$:

    $\mu_z^{(t+1)} = \frac{\sum_{i=1}^{n} E\_Z_{:i}}{n}$

    $\Sigma_z^{(t+1)} = \frac{1}{n}E\_Z\_Z\_T - \mu_z^{(t+1)}{\mu_z^{(t+1)}}^\top$

    $\Lambda^{(t+1)} = \left(n\bar{x}{\mu_z^{(t+1)}}^\top - XE\_Z^\top\right)\left(n\mu_z^{(t+1)}{\mu_z^{(t+1)}}^\top - E\_Z\_Z\_T\right)^{-1}$

    $\mu^{(t+1)} = \bar{x} - \Lambda^{(t+1)}\mu_z^{(t+1)}$

    $\Psi^{(t+1)} = \text{diag}(\frac{1}{n}((X - \mu^{(t+1)}\underbrace{\begin{bmatrix}1 & 1 & \dots & 1\end{bmatrix}}_{\in\mathbb{R}^{1\times n}})(X - \mu^{(t+1)}\underbrace{\begin{bmatrix}1 & 1 & \dots & 1\end{bmatrix}}_{\in\mathbb{R}^{1\times n}})^\top - (X - \mu^{(t+1)}\underbrace{\begin{bmatrix}1 & 1 & \dots & 1\end{bmatrix}}_{\in\mathbb{R}^{1\times n}})E\_Z^\top{\Lambda^{(t+1)}}^\top - \Lambda^{(t+1)}E\_Z(X - \mu^{(t+1)}\underbrace{\begin{bmatrix}1 & 1 & \dots & 1\end{bmatrix}}_{\in\mathbb{R}^{1\times n}})^\top + \Lambda^{(t+1)}E\_Z\_Z\_T{\Lambda^{(t+1)}}^\top))$

    $l_{\text{RV\_Do}}^{(t+1)} = \text{LOGLIKELIHOOD}(\text{X,Z,}\theta^{(t+1)})$

    if $\frac{\left\|\theta^{(t)} - \theta^{(t+1)}\right\|_2^2}{\left\|\theta^{(t)}\right\|_2^2} \leq$eps **or** $\frac{|l_{\text{RV_Do}}^{(t)} - l_{\text{RV_Do}}^{(t+1)}|}{|l_{\text{RV_Do}}^{(t)}|} \leq$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 *S2.FA*

  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)

Appendix D. MS3.FA

The algorithms below are more general: z is not just a real number, as we state in the paper, but a real vector.

Algorithm A5 MS3.FA—Other functions.

function CONDNORMALNA(y,(μ,Σ))
  I = $\{i|y_i = NA\}$ ▷ I = Indexes
  OI = $\{i|y_i \neq NA\}$ ▷ OI = Other Indexes
  mean = $\mu_I + \Sigma_{I,OI}\Sigma_{OI}^{-1}(y_{OI} - \mu_{OI})$
  covarianceMatrix = $\Sigma_I - \Sigma_{I,OI}\Sigma_{OI}^{-1}\Sigma_{I,OI}^{\top}$
  $E[Y]_I$ = mean
  $E[Y]_{OI} = y_{OI}$
  $E[YY^{\top}] = E[Y]E[Y]^{\top}$
  $E[YY^{\top}]_{I,I} = E[YY^{\top}]_{I,I}$ + covarianceMatrix
  return ($E[Y]$, $E[YY^{\top}]$)
function FULLNORMAL($\mu_z, \Sigma_z, \mu, \Lambda, \Psi$)
  mean = $\begin{bmatrix} \mu + \Lambda\mu_z \\ \mu_z \end{bmatrix}$
  covarianceMatrix = $\begin{bmatrix} \Lambda\Sigma_z\Lambda^{\top} + \Psi & \Lambda\Sigma_z \\ (\Lambda\Sigma_z)^{\top} & \Sigma_z \end{bmatrix}$
  return (mean,covarianceMatrix)
function LOGLIKELIHOOD($\{(x^{(1)}, z^{(1)}), \ldots, (x^{(n)}, z^{(n)})\}$, ($\mu_z, \Sigma_z, \mu, \Lambda, \Psi$))
  (mean,cov) = FULLNORMAL($\mu_z, \Sigma_z, \mu, \Lambda, \Psi$)
  result = 0
  for i = 1:n **do**
    $y^{(i)} = \begin{bmatrix} x^{(i)} \\ z^{(i)} \end{bmatrix}$
    OI = $\{j|y_j^{(i)} \neq NA\}$ ▷ OI = Other indexes
    result = result + $\ln \mathcal{N}(y_{OI}^{(i)}|\text{mean}_{OI}, \text{cov}_{OI,OI})$
  return result

Algorithm A6 MS3.FA—Train.

function TRAIN($\{(x^{(1)}, z^{(1)}), \ldots, (x^{(n)}, z^{(n)})\}$,nMaxIterations,eps) ▷ data can have NAs
  $\theta^{(0)} = \text{INITIALIZEPARAMETERS}(\{(x^{(1)}, z^{(1)}), \ldots, (x^{(n)}, z^{(n)})\})$
  $l_{\text{RV\_Do}}^{(0)} = \text{LOGLIKELIHOOD}(\{(x^{(1)}, z^{(1)}), \ldots, (x^{(n)}, z^{(n)})\}, \theta^{(0)})$
  for t = 0:nMaxIterations **do**
    E step: Compute the following for all $i \in \{1, \ldots, n\}$:
    $y^{(i)} = \begin{bmatrix} x^{(i)} \\ z^{(i)} \end{bmatrix}$
    fullNormal = $\text{FULLNORMAL}(\mu_z^{(t)}, \Sigma_z^{(t)}, \mu^{(t)}, \Lambda^{(t)}, \Psi^{(t)})$
    $(E[Y^{(i)}], E[Y^{(i)}Y^{(i)\top}]) = \text{CONDNORMALNA}(y^{(i)}, \text{fullNormal})$
    $E[X^{(i)}] = E[Y^{(i)}]_{1:D}$ ▷ D = dim($x^{(i)}$)
    $E[Z^{(i)}] = E[Y^{(i)}]_{(D+1):(D+d)}$ ▷ d = dim($z^{(i)}$)
    $E[X^{(i)}X^{(i)\top}] = E[Y^{(i)}Y^{(i)\top}]_{1:D,1:D}$
    $E[X^{(i)}Z^{(i)\top}] = E[Y^{(i)}Y^{(i)\top}]_{1:D,(D+1):(D+d)}$
    $E[Z^{(i)}X^{(i)\top}] = E[X^{(i)}Z^{(i)\top}]^\top$
    $E[Z^{(i)}Z^{(i)\top}] = E[Y^{(i)}Y^{(i)\top}]_{(D+1):(D+d),(D+1):(D+d)}$
    M Step: Compute $\theta^{(t+1)} = (\mu_z^{(t+1)}, \Sigma_z^{(t+1)}, \mu^{(t+1)}, \Lambda^{(t+1)}, \Psi^{(t+1)})$:
    $\mu_z^{(t+1)} = \frac{\sum_{i=1}^n E[Z^{(i)}]}{n}$
    $\Sigma_z^{(t+1)} = \frac{\sum_{i=1}^n (E[Z^{(i)}Z^{(i)\top}] - E[Z^{(i)}]\mu_z^{(t+1)\top} - \mu_z^{(t+1)}E[Z^{(i)\top}] + \mu_z^{(t+1)}\mu_z^{(t+1)\top})}{n}$
    $\bar{x} = \frac{\sum_{i=1}^n E[X^{(i)}]}{n}$
    $\Lambda^{(t+1)} = \left(n\bar{x}\mu_z^{(t+1)\top} - \sum_{i=1}^n E[X^{(i)}Z^{(i)\top}]\right)\left(n\mu_z^{(t+1)}\mu_z^{(t+1)\top} - \sum_{i=1}^n E[Z^{(i)}Z^{(i)\top}]\right)^{-1}$
    $\mu^{(t+1)} = \bar{x} - \Lambda^{(t+1)}\mu_z^{(t+1)}$
    $\Psi^{(t+1)} = \text{diag}(\frac{1}{n}\sum_{i=1}^n (E[X^{(i)}X^{(i)\top}] - E[X^{(i)}]\mu^{(t+1)\top} - \mu^{(t+1)}E[X^{(i)}]^\top + \mu^{(t+1)}\mu^{(t+1)\top} - E[X^{(i)}Z^{(i)\top}]\Lambda^{(t+1)\top} + \mu^{(t+1)}E[Z^{(i)}]^\top\Lambda^{(t+1)\top} - \Lambda^{(t+1)}E[Z^{(i)}X^{(i)\top}] + \Lambda^{(t+1)}E[Z^{(i)}]^\top\mu^{(t+1)\top} + \Lambda^{(t+1)}E[Z^{(i)}Z^{(i)\top}]\Lambda^{(t+1)\top}))$
    $l_{\text{RV\_Do}}^{(t+1)} = \text{LOGLIKELIHOOD}(\{(x^{(1)}, z^{(1)}), \ldots, (x^{(n)}, z^{(n)})\}, \theta^{(t+1)})$
    if $\frac{\|\theta^{(t)} - \theta^{(t+1)}\|_2^2}{\|\theta^{(t)}\|_2^2} \leq$eps **or** $\frac{|l_{\text{RV_Do}}^{(t)} - l_{\text{RV_Do}}^{(t+1)}|}{|l_{\text{RV_Do}}^{(t)}|} \leq$eps **then**
      break
  return $\theta^{(t)}$

Algorithm A7 MS3.FA—Test and Impute.

function TEST($y^* = (x^*, z^*)$ partially known, ($\hat{\mu}_z$,$\hat{\Sigma}_z$,$\hat{\Lambda}$,$\hat{\mu}$,$\hat{\Psi}$))
  fullNormal = $\text{FULLNORMAL}(\hat{\mu}_z, \hat{\Sigma}_z, \hat{\mu}, \hat{\Lambda}, \hat{\Psi})$
  $(E[Y^*], E[Y^*Y^{*\top}]) = \text{CONDNORMALNA}(y^*, \text{fullNormal})$
  value = $E[Y^*]$
  covarianceMatrix = $E[Y^*Y^{*\top}] - E[Y^*]E[Y^*]^\top$
  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

References

1. Mitchell, T. Generative and Discriminative Classifiers: Naive Bayes and Logistic Regression. (Additional Chapter to *Machine Learning*; McGraw-Hill: New York, NY, USA, 1997.) Published Online. 2017. Available online: https://bit.ly/39Ueb4o (accessed on 31 July 2021).
2. Murphy, K. *Machine Learning: A Probabilistic Perspective*; MIT Press: Cambridge, MA, USA, 2012.
3. Ng, A. Machine Learning Course, Lecture Notes, Mixtures of Gaussians and the EM Algorithm. Available online: http://cs229.stanford.edu/notes2020spring/cs229-notes7b.pdf (accessed on 31 July 2021).
4. Singh, A. *Machine Learning Course, Homework 4, pr 1.1*; CMU: Pittsburgh, PA, USA, 2010; p. 528 in Ciortuz, L.; Munteanu, A.; Bădărău, E. *Machine Learning Exercise Book (In Romanian)*; Alexandru Ioan Cuza University of Iași: Iași, Romania, 2019. Available online: https://bit.ly/320ZuIk (accessed on 31 July 2021).
5. Ng, A. Machine Learning Course, Lecture Notes, Part X. Available online: http://cs229.stanford.edu/notes2020spring/cs229-notes9.pdf (accessed on 31 July 2021).
6. Goodfellow, I.; Bengio, Y.; Courville, A. *Deep Learning*; MIT Press: Cambridge, MA, USA, 2016. Available online: http://www.deeplearningbook.org (accessed on 31 July 2021).
7. Tipping, M.E.; Bishop, C.M. Probabilistic Principal Component Analysis. *J. R. Stat. Soc. Ser. (Stat. Methodol.)* **1999**, *61*, 611–622. Available online: https://bit.ly/2PCxoRr (accessed on 31 July 2021). [CrossRef]
8. Ciobanu, S. Exploiting a New Probabilistic Model: Simple-Supervised Factor Analysis. Master's Thesis, Alexandru Ioan Cuza University of Iași, Iași, Romania, 2019. Available online: https://bit.ly/31UsBx6 (accessed on 31 July 2021).
9. Ng, A. Machine Learning Course, Lecture Notes, Part XI. Available online: http://cs229.stanford.edu/notes2020spring/cs229-notes10.pdf (accessed on 31 July 2021).
10. Lawrence, N.D. Gaussian Process Latent Variable Models for Visualisation of High Dimensional Data. *Adv. Neural Inf. Process. Syst.* **2004**, 329–336. Available online: https://papers.nips.cc/paper/2540-gaussian-process-latent-variable-models-for-visualisation-of-high-dimensional-data.pdf (accessed on 31 July 2021).
11. Gao, X.; Wang, X.; Tao, D.; Li, X. Supervised Gaussian Process Latent Variable Model for Dimensionality Reduction. *IEEE Trans. Syst. Man, Cybern. Part (Cybern.)* **2010**, *41*, 425–434. Available online: https://ieeexplore.ieee.org/document/5545418 (accessed on 31 July 2021).
12. Mitchell, T.; Xing, E.; Singh, A. *Machine Learning Course, Midterm Exam, pr. 5.3*; CMU: Pittsburgh, PA, USA, 2010; p. 565 Ciortuz, L.; Munteanu, A.; Bădărău, E. *Machine Learning Exercise Book (In Romanian)*; Alexandru Ioan Cuza University of Iași: Iași, Romania, 2019. Available online: https://bit.ly/320ZuIk (accessed on 31 July 2021).
13. Ziyatdinov, A.; Fonollosa, J.; Fernández, L.; Gutierrez-Gálvez, A.; Marco, S.; Perera, A. Bioinspired early detection through gas flow modulation in chemo-sensory systems. *Sens. Actuators Chem.* **2015**, *206*, 538–547. [CrossRef]
14. Spyromitros-Xioufis, E.; TSOUMAKAS, G.; WILLIAM, G.; Vlahavas, I. Drawing parallels between multi-label classification and multi-target regression. *arXiv* **2014**, arXiv:1211.6581 v2.
15. Xiaojin, Z.; Zoubin, G. *Learning from Labeled and Unlabeled Data with Label Propagation*; Technical Report CMU-CALD-02–107; Carnegie Mellon University: Pittsburgh, PA, USA, 2002.
16. Wang, J. *SSL: Semi-Supervised Learning*; R Package Version 0.1; 2016. Available online: https://CRAN.R-project.org/package=SSL (accessed on 31 July 2021).
17. Oliver, A.; Odena, A.; Raffel, C.; Cubuk, E.D.; Goodfellow, I.J. Realistic evaluation of deep semi-supervised learning algorithms. *arXiv* **2018**, arXiv:1804.09170.
18. van Buuren, S.; Groothuis-Oudshoorn, K. Mice: Multivariate Imputation by Chained Equations in R. *J. Stat. Softw.* **2011**, *45*, 1–67. Available online: https://www.jstatsoft.org/v45/i03/ (accessed on 31 July 2021). [CrossRef]
19. Honaker, J.; King, G.; Blackwell, M. Amelia II: A Program for Missing Data. *J. Stat. Softw.* **2011**, *45*, 1–47. Available online: http://www.jstatsoft.org/v45/i07/ (accessed on 31 July 2021). [CrossRef]
20. Ghahramani, Z.; Hinton, G.E. *The EM Algorithm for Mixtures of Factor Analyzers*; Technical Report, CRG-TR-96-1; University of Toronto: Toronto, ON, Canada, 1996. Available online: http://mlg.eng.cam.ac.uk/zoubin/papers/tr-96-1.pdf (accessed on 31 July 2021).
21. Bishop, C.M. *Pattern Recognition and Machine Learning*; Springer Science + Business Media: Berlin, Germany, 2006.
22. Zaharia, M.; Xin, R.S.; Wendell, P.; Das, T.; Armbrust, M.; Dave, A.; Meng, X.; Rosen, J.; Venkataraman, S.; Franklin, M.J.; et al. Apache Spark: A Unified Engine for Big Data Processing. *Commun. ACM* **2016**, *59*, 56–65. [CrossRef]
23. Dempster, A.P.; Laird, N.M.; Rubin, D.B. Maximum likelihood from incomplete data via the EM algorithm. *J. R. Stat. Soc. Ser. (Methodol.)* **1977**, *39*, 1–22.
24. Kingma, D.P.; Welling, M. Auto-encoding variational bayes. *arXiv* **2013**, arXiv:1312.6114.