Non-parametric Variable Selection on Non-linear Manifolds

Loann David Denis Desboulets

Non-parametric Variable Selection on Non-linear Manifolds

Loann D. DESBOULETS[†]

[†] CNRS, EHESS, Centrale Marseille, AMSE, Aix-Marseille University,
5-9 Boulevard Maurice Bourdet 13001 Marseille, France;
loann.DESBOULETS@univ-amu.fr

November 3, 2020

Abstract

In this paper, I investigate a new non-parametric variable selection framework. To extend the usual non-parametric model, I consider non-linear manifolds which are more flexible. Non-linear manifolds are represented by function compositions, allowing more complex dependences in the data. Based on two manifold approximation techniques, k-nearest neighbours and auto-encoder neural networks, I propose two different procedures to perform non-parametric variable selection. The two methods are complementary, the former being a local estimator, while the latter is a global estimator.

Keywords – Non-parametric, Non-linear Manifolds, Variable Selection, Neural Networks

1. Introduction

Non-parametric models are popular modelling techniques, that allows a great flexibility. When the number of explanatory variables is large, variable selection come in handy. Variable selection is meant to infer a subset of the most pertinent explanatory variables from the non-parametric model

$$y_i = m(\mathbf{x}_i) + \epsilon_i \quad \forall i \in 1, .., n. \tag{1}$$

Where y is the dependent variable, $\mathbf{x}$ is a vector of covariates, ϵ is a noise term and m is an unknown function. Selection is useful in reducing the dimension of the estimator $\hat{m}(\cdot)$, which is a serious issue for high dimensional non-parametric models, known as the curse of dimensionality. There exists plenty of methods to perform non-parametric variable selection, see Mehmood et al. (2012); Desboulets (2018) for recent reviews. However, there might exists more complex correlations, that cannot be modelled directly using functions. And in equation 1, the unknown model $m(\cdot)$ has to be a function. The Swiss Roll data of Roweis and Saul (2000) is an instance of such complex correlations. To get a more general approach, I consider non-linear manifolds which are more flexible than functions. A manifold is often represented using a function composition. This can represent more complex relationships than a single function. Also in the manifold equation, there is also no distinction between explanatory variables and outcome variables. Both are embed into a single matrix $(X_1, X_2, ..., X_p) \equiv \mathbf{X} \in \mathbb{R}^{n \times p}$. This can be useful, in situations where there is a priori no obvious reason to choose a particular variables as being more dependent than the others. It can occurs in many areas of Economics, especially in Finance, for example when regressing some returns on other returns. The manifold $\mathcal{M}$ is written as follow

$$\begin{aligned} \mathcal{M} :=& \{\mathbf{x} \in \mathbb{R}^p | \mathbf{x} = f \circ g(\mathbf{x})\} \\ & g : \mathbb{R}^p \to \mathbb{R}^r, f : \mathbb{R}^r \to \mathbb{R}^p, p > r \end{aligned} \tag{2}$$

The projection function g maps the p-dimensional data onto the lower-dimensional coordinates of the r-manifold. Then, the immersion function f is mapping back to the original space. The data are assumed to be i.i.d. and stationary. The manifold selection problem is to find the variables which have the greatest correlations with all the other variables, expressed in terms of marginal contributions. Obviously, I have to discard the correlation between a variable and itself. Its projection through the manifold cannot be the identity function. The "true" set of these correlated variables is denoted $\mathcal{S}^*$ and defined as follow

$$\mathcal{S}^* := \{j : \exists j' \neq j, \frac{\partial X_{j'}}{\partial X_j} \neq 0\} \Leftrightarrow \{j : \frac{\partial (f \circ g)}{\partial X_j} \neq 1\}$$
$$|\mathcal{S}^*| \leq p \tag{3}$$

The contribution of this paper is to propose two manifold selection operators to infer the set of "active variables" $\mathcal{S}^*$, from equation 3. The first is a locally linear estimator, based on the k-nearest neighbours algorithm named LLMS, for Local Linear Manifold Selection. The second is a global estimator, based on auto-encoders neural networks (Kramer, 1991), named DAMS for Diagonal Auto-encoder Manifold Selection. The two approaches rely on completely different mechanisms but solve almost the same problem. The local estimator does not rely on continuity everywhere, contrary to the global one. This means f, and or g, can be only locally continuous. Moreover, the local approach has low computational complexity but might be sensitive to noisy data. The global one is exactly the opposite. These reasons motivate for having two distinct approaches.

In general, any selection operator relies on a regression estimator, that approximates an associated model. Thus, selection on manifold has to rely on an estimator that approximates the manifold. A large literature is devoted to inference of manifold in the data, referred to as principal manifolds or unsupervised manifold learning. They use non-parametric estimators such as for instance kernels Hastie and Stuetzle (1989); Schölkopf et al. (1998); Mika et al. (1999), or neural networks Kohonen (1998); Vincent et al. (2010), and nearest neighbours Roweis and Saul (2000); Saul and Roweis (2003). A useful introduction to manifold learning can be found in Gorban and Zinovyev (2011).

Obviously these are regression techniques, they are not doing variable selection per se. They only estimate the manifold. The literature on manifold selection is quite recent, but vary across topics. On the one hand, there exists several papers for classification problems and image recognition (Ding-Cheng et al., 2014; Li et al., 2016; Tang et al., 2019; Doquet and Sebag, 2020). Typically, they assume the explanatory variables lie on a manifold. This manifold represents the only source of variability in the data. It summarizes all the information contained in the data, and so it reduces the dimension of the estimator, avoiding de facto the curse of dimensionality. It is a non-parametric equivalent to Principal Component Regression (Jolliffe, 1982). Once extracted, using the manifold instead of the raw data produces a more parsimonious classifier, usually with better performance. The paper of Ding-Cheng et al. (2014) is actually very close to the first of the two methods I propose (LLMS). The main difference is they use a score based on reconstruction error to assess the importance of a variable. They focus on finding which variables is correlated to the binary

outcome, which is not what I do here. I try to find which variables are correlated with all the others, not a single one. On the other hand, there is almost no literature on manifold selection for regression purposes, i.e. only with continuous data. The only method that is close to the one I propose, is the Nonparanormal (hereafter NPN) of Lafferty et al. (2012). In their paper, they assume non-parametric transformations of the variables are distributed as a multivariate Gaussian. Once the transformations are estimated, the selection is performed using the Graphical LASSO Friedman et al. (2008). However, their method is restricted to monotonically increasing transforms. This is a particular kind of manifolds. Hence, their method is not as flexible as the method I propose here.

The manifold model is a general and flexible statistical model. That is desirable, because assuming an inappropriate statistical model might not lead to consistent selection. Intuitively, if a selection method assumes that the data is normally distributed, while it is not, it is unlikely the estimated set of correlated variables is the true one. More formally, let us define a selection operator as $\mathcal{S} = \sigma(\mathbf{X}, P)$. This operator depends on a statistical model, described by a sample $\mathbf{X}$, and a set of parametrized probability distributions $P = \{P_\omega : \omega \in \Omega\}$. For instance, the famous LASSO (Tibshirani, 1996) is based on a linear statistical model $\{Y = \mathbf{X}\beta : \beta \in \mathbb{R}^p\}$. For the sake of simplicity, hereafter the notation for the distribution is shortened to its parameters. The notation for the LASSO is therefore $\sigma_{LASSO}(\mathbf{X}, \beta)$. This LASSO achieves sparsity by solving $min_\beta \left\|Y - \mathbf{X}\beta\right\|_2^2 + \theta \left\|\beta\right\|_1$, where θ is a hyper-parameter that determines the amount of regularisation. Generally speaking, the Manifold Selection (MS) estimator is denoted as $\sigma_{MS}(\mathbf{X}, f \circ g)$. So, using simulations, I show there must be some correspondence between the "true" data generating process and the assumed statistical model, otherwise the selected set is not consistent. Only the two methods I propose infer $\mathcal{S}^*$ correctly when the data is distributed on a manifold.

Without further restrictions, the identification of $\mathcal{S}^*$ is not possible. Also, the function composition is not unique. The condition on the domains of the functions $p > r$ reduces the function space for sure, but is not sufficient to obtain uniqueness. To identify $\mathcal{S}^*$, what is required is the identification of the partial derivatives, not $f \circ g$ itself. Still, if $f \circ g$ is unique, so are its derivatives. The fundamental problem here, is that it is always possible to find a function, as complex as needed, that links two variables. More generally, a smooth curve ($r = 1$) can pass through every points in the data, provided it is non-linear enough. One simple and usual solution for non-parametric models, is to impose a regularity condition on the total variation of $f \circ g$. In the literature, a penalty on the second order derivatives is often used. Indeed, this limits, by definition, the variability of the function. But there

exists other ways to prevent from the aforementioned problem. In the case of the global estimator, based on neural networks, it suffices to have "not too large" intermediate layers. If the activation functions are monotonic, each supplementary neurons in the intermediate layer brings another possible extrema to the fitted function. Limiting the number of neurons therefore limits the variability. For the local estimator, the solution is the local linear models themselves, as a linear function has no variability. Notice that this is true as long as the parameter k is large enough: $k \geq p + 1$. Otherwise there no identification of the linear models.

Another identification problem arises if we consider the variables to be measured with noise, which is often the case empirically.

$$\begin{aligned} \mathcal{M} :=& \{\mathbf{x} \in \mathbb{R}^p | \mathbf{x} = f \circ g(\mathbf{x}^*) + \boldsymbol{\epsilon}\} \\ & g : \mathbb{R}^p \to \mathbb{R}^r, f : \mathbb{R}^r \to \mathbb{R}^p, p > r \end{aligned} \tag{4}$$

Here each variable is contaminated with a specific error term, therefore the dimension of the noise is the same as the dimension of the data. Another possibility is for the coordinates of the manifold to be noisy, but I do not take this one in consideration. In the general case, the conditions that ensures identification of the manifold are not known. A necessary but not sufficient condition is that true values of the data are independent from the noise $\mathbf{x}^* \perp \boldsymbol{\epsilon}$, and the noises are uncorrelated i.e. $\mathbb{V}[\boldsymbol{\epsilon}]$ is diagonal. There is no need to impose the same variance for each noise, nor this variance to be constant over all the distribution of the data.

The rest of the paper is organised as follow. In section 2, I develop the LLMS estimator, based on the k-nearest neighbours algorithm. Its theoretical properties are studied, and three different versions are proposed to improve selection. In section 3, I develop the DAMS estimator, based on neural networks. I study a new network structure to perform selection and extend it to auto-encoder networks. In section 4, I run simulations to compare existing methods, LLMS and DAMS. I show that both proposed methods can select variables consistently.

2. Local Linear Approach via k-Nearest Neighbours

2.1. Motivations for a Locally Linear estimator

The very definition of a manifold is that it is locally a Euclidean space, i.e. in a neighbourhood $\mathbf{N}$ of a given point $\mathbf{x_0}$ defined in a small enough euclidean ball $\mathcal{B}_{x_0}$ of radius ν.

$$\begin{aligned}\mathcal{B}_{\mathbf{x_0}} =&\{\mathbf{X} \in \mathbb{R}^p, \|\mathbf{X} - \mathbf{x_0}\|_2^2 \leq \nu\} \subset \mathbb{R}^p \\ \mathbf{N}_{\mathbf{x_0}} =&\{\mathbf{x} \in \mathcal{B}_{\mathbf{x_0}} | \mathbf{x} = f \circ g(\mathbf{x})\}.\end{aligned} \tag{5}$$

It is also worth noticing that the union of all the possible neighbourhoods is the entire manifold

$$\mathcal{M} = \bigcup_{\forall \mathbf{x_0}} \mathbf{N}_{\mathbf{x_0}}. \tag{6}$$

The data matrix $\mathbf{X}$ is a sample from $\mathcal{M}$, and the neighbourhoods are sub-samples of $\mathbf{X}$. In a neighbourhood of $\mathbf{x_0}$, a linear approximation of the manifold is sufficient as the error term is limited by definition to be a small value: $\mathcal{O}((\mathbf{X} - \mathbf{x_0})^2) \approx \|\mathbf{X} - \mathbf{x_0}\|_2^2 \leq \nu$. Following a first order Taylor expansion of the manifold:

$$f \circ g(\mathbf{x}) = f \circ g(\mathbf{x_0}) + \nabla_{f \circ g}(\mathbf{x_0})(\mathbf{X} - \mathbf{x_0}) + \mathcal{O}((\mathbf{X} - \mathbf{x_0})^2), \tag{7}$$

the definition of a neighbourhood can be changed to

$$\begin{aligned}\mathbf{N}_{\mathbf{x_0}} &= \{\mathbf{x} \in \mathcal{B}_{x_0} | \mathbf{x} \approx f \circ g(\mathbf{x_0}) + \nabla_{f \circ g}(\mathbf{x_0})(\mathbf{X} - \mathbf{x_0})\} \\ \Leftrightarrow \mathbf{N}_{\mathbf{x_0}} &= \{\mathbf{x} \in \mathcal{B}_{x_0} | \mathbf{x} \approx \mu_{\mathbf{x_0}} + \mathbf{x}\beta_{\mathbf{x_0}}\}\end{aligned} \tag{8}$$

Following equations 6 and 8, $\mathbf{X}$ is the union of all the neighbourhoods and the function composition $f \circ g$ is approximated by the union of all the local linear models. So the manifold selection can be rewritten as follow:

$$\sigma(\mathbf{X}, f \circ g) = \sigma(\bigcup_{\forall \mathbf{x_0}} \mathbf{N}_{\mathbf{x_0}}, \bigcup_{\forall \mathbf{x_0}} \mu_{\mathbf{x_0}} + \mathbf{x}\beta_{\mathbf{x_0}}). \tag{9}$$

Now, if I assume separability of the selection operators, I can write the manifold selection as a union of local linear selection:

$$\sigma(\bigcup_{\forall \mathbf{x_0}} \mathbf{N}_{\mathbf{x_0}}, \bigcup_{\forall \mathbf{x_0}} \mu_{\mathbf{x_0}} + \mathbf{x}\beta_{\mathbf{x_0}}) = \bigcup_{\forall \mathbf{x_0}} \sigma(\mathbf{N}_{\mathbf{x_0}}, \mu_{\mathbf{x_0}} + \mathbf{x}\beta_{\mathbf{x_0}}). \tag{10}$$

To illustrate this local separability assumption, let's take a practical example. Assume we have three variables, x, y, z, such that x and y lie on a circle and z is a noise variable. We can write the associated manifold using a function composition. The projection function g maps the two dimensional coordinates x and y onto the one-dimensional circle, and keeps z as is. The immersion function f is basically the inverse of g.

$$\begin{aligned} g\colon \mathbb{R}^3 &\to \mathbb{R}^2 \\ (x, y, z) &\mapsto (t := atan2(x, y), z). \\ f\colon \mathbb{R}^2 &\to \mathbb{R}^3 \\ (t, z) &\mapsto (sin(t), cos(t), z). \end{aligned} \tag{11}$$

with

$$\begin{aligned} atan2\colon \mathbb{R}^2 &\to \mathbb{R}^1 \\ (x, y) &\mapsto atan(y/x)\mathbb{1}_{\{x>0\}} \\ &\quad + \pi\mathbb{1}_{\{x<0 \ \& \ y\geq 0\}} - \pi\mathbb{1}_{\{x<0 \ \& \ y<0\}} \\ &\quad + \frac{\pi}{2}\mathbb{1}_{\{x=0 \ \& \ y>0\}} - \frac{\pi}{2}\mathbb{1}_{\{x=0 \ \& \ y<0\}} \end{aligned} \tag{12}$$

In matrix form, we have:

$$\begin{aligned} X :=& (x, y, z), \\ X =& f \circ g(X). \end{aligned} \tag{13}$$

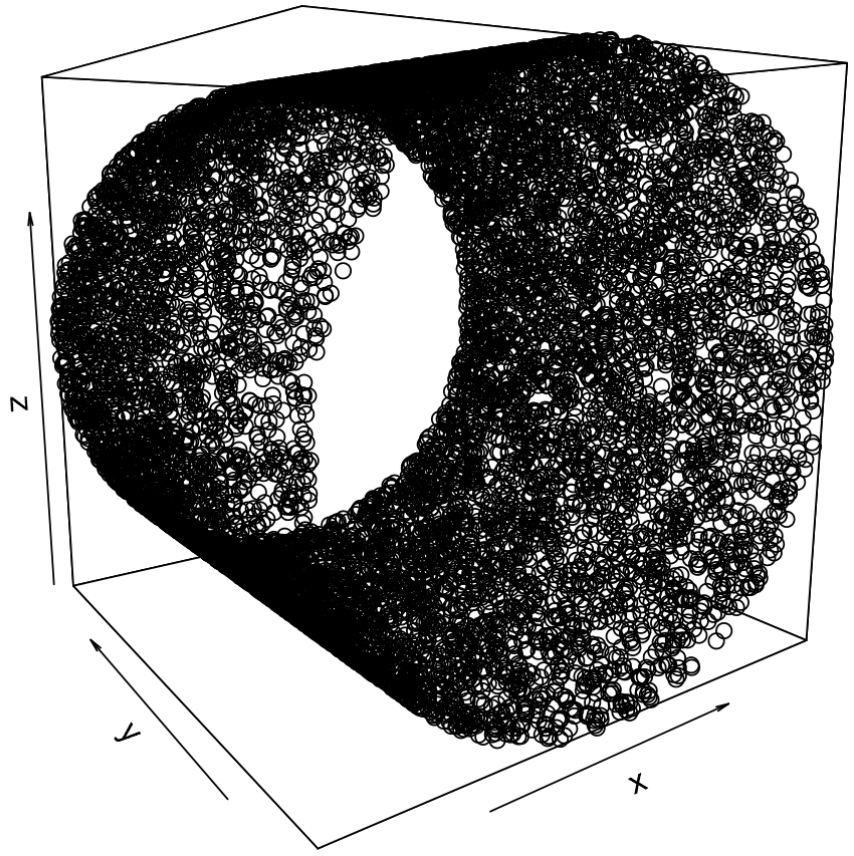
The manifold, which is a cylinder, formed by the three variables x, y, z is depicted in Figure 1. The "true set" of variable $\mathcal{S}^* := \{j : \exists j' \neq j, \frac{\partial X_{j'}}{\partial X_j} \neq 0\}$ requires to compute partial derivatives of each variable w.r.t to all the others. These derivatives are easier to calculate using the implicit form of the manifold equation. Let's define the low-coordinates of the manifold as follow:

$$\begin{aligned} t &\sim \mathcal{U}[-\pi, \pi], \\ \epsilon &\sim \mathcal{U}[-1, 1]. \end{aligned} \tag{14}$$

Then the manifold can be written in implicit form as follow:

$$\begin{aligned} x &= sin(t), \\ y &= cos(t), \\ z &= \epsilon. \end{aligned} \tag{15}$$

Fig. 1. Example of a non-linear manifold formed with three variables x, y, z



These equations can also be written in terms of the observed variables x, y, z only:

$$\begin{aligned} x &= sin(cos^{-1}(y)) = \sqrt{(}1 - y^2), \\ y &= cos(sin^{-1}(x)) = \sqrt{(}1 - x^2), \\ z &= \epsilon \end{aligned} \tag{16}$$

These formulations are equivalent to the formulation in equation 13. Now, we can work out

the partial derivatives:

$$\begin{aligned}
\frac{\partial x}{\partial y} &= -\frac{y}{\sqrt{(1-y^2)}} = \begin{cases} \neq 0 & \text{if } y \neq \{0,-1,+1\} \\ = 0 & \text{if } y = 0 \\ \text{undefined} & \text{if } y = \{-1,+1\} \end{cases} \\
\frac{\partial y}{\partial x} &= -\frac{x}{\sqrt{(1-x^2)}} = \begin{cases} \neq 0 & \text{if } x \neq \{0,-1,+1\} \\ = 0 & \text{if } x = 0 \\ \text{undefined} & \text{if } x = \{-1,+1\} \end{cases} \\
\frac{\partial x}{\partial z} &= \frac{\partial z}{\partial x} = \frac{\partial y}{\partial z} = \frac{\partial z}{\partial y} = 0
\end{aligned} \tag{17}$$

Or in matrix form:

$$\frac{\partial X}{\partial X} = \begin{pmatrix} 1 & -\frac{y}{\sqrt{(1-y^2)}} & 0 \\ -\frac{x}{\sqrt{(1-x^2)}} & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{18}$$

The true set of variable is therefore:

$$\begin{aligned}
\Rightarrow \mathcal{S}^* :=& \ \{j : \exists j' \neq j, \frac{\partial X_{j'}}{\partial X_j} \neq 0\} \\
=& \{1,2\}
\end{aligned} \tag{19}$$

It is worth noticing here, what a selection operator assuming a linear model in the data would do. The plane that best fit the data in Figure 1, is flat in all variables. The correlation matrix of these data is a diagonal matrix, and so are its eigenvectors. So whatever the mechanism that drives the selection, e.g. penalty on the parameters, it would not select anything. This happens because the supports of all three variables are symmetric ($[-1,+1]$). However, with a suitable estimate of the partial derivatives, we can define a selection estimator such that $\sigma(\mathbf{X}, f \circ g) = \mathcal{S}^*$. However, an estimate of the partial derivatives on the whole support is hard to obtain. We can recover the true set, only using local estimates of the derivatives. For the sake of simplicity, let us choose 4 points t_0 and compute the local linear selection at these points. The whole process is illustrated in Figure 2.

$$\begin{aligned}
t_0 &\in \left\{ \frac{\pi}{4}, \frac{-\pi}{4}, \frac{3\pi}{4}, \frac{-3\pi}{4} \right\} \\
\mathbf{x_0} &= (sin(t_0), cos(t_0), z(t_0))
\end{aligned} \tag{20}$$

The local derivatives are obtained by taking the first order Taylor expansion $f(t) = f(t_0) + f'(t_0)(t - t_0)$ of the circle and then differentiating w.r.t to the other variables.

- $t_0 = \frac{\pi}{4}$

$$\begin{aligned} x = sin(t) &\approx +\frac{1}{\sqrt{2}} + \frac{1}{\sqrt{2}}\left(t - \frac{\pi}{4}\right) \\ y = cos(t) &\approx +\frac{1}{\sqrt{2}} - \frac{1}{\sqrt{2}}\left(t - \frac{\pi}{4}\right) \end{aligned} \tag{21}$$

One can see the two approximations have the same coefficients, and only differ by the leading coefficient sign, therefore the derivatives between x and y are trivial. The derivatives involving z do not change, the variable z is not present in the approximations formulas.

$$\begin{aligned} &\frac{\partial x}{\partial y} = -1, \\ &\frac{\partial y}{\partial x} = -1, \\ &\frac{\partial x}{\partial z} = \frac{\partial z}{\partial x} = \frac{\partial y}{\partial z} = \frac{\partial z}{\partial y} = 0. \end{aligned} \tag{22}$$

Thus, the selection set at this point is $\sigma(\mathbf{N}_{\mathbf{x_0}(\frac{\pi}{4})}, \mu_{\mathbf{x_0}(\frac{\pi}{4})} + \mathbf{x}\beta_{\mathbf{x_0}(\frac{\pi}{4})}) = \{1, 2\}$. The same logic is applied to the other points.

- $t_0 = \frac{-\pi}{4}$

$$\begin{aligned} x = sin(t) &\approx +\frac{1}{\sqrt{2}} + \frac{1}{\sqrt{2}}\left(t + \frac{\pi}{4}\right) \\ y = cos(t) &\approx +\frac{1}{\sqrt{2}} + \frac{1}{\sqrt{2}}\left(t + \frac{\pi}{4}\right) \\ &\rightarrow \frac{\partial x}{\partial y} = -1 \\ &\rightarrow \frac{\partial y}{\partial x} = +1 \\ &\rightarrow \frac{\partial x}{\partial z} = \frac{\partial z}{\partial x} = \frac{\partial y}{\partial z} = \frac{\partial z}{\partial y} = 0 \\ &\sigma(\mathbf{N}_{\mathbf{x_0}(\frac{-\pi}{4})}, \mu_{\mathbf{x_0}(\frac{-\pi}{4})} + \mathbf{x}\beta_{\mathbf{x_0}(\frac{-\pi}{4})}) = \{1, 2\} \end{aligned} \tag{23}$$

- $t_0 = \frac{3\pi}{4}$

$$
\begin{aligned}
x = sin(t) &\approx +\frac{1}{\sqrt{2}} - \frac{1}{\sqrt{2}}\left(t - \frac{\pi}{4}\right) \\
y = cos(t) &\approx +\frac{1}{\sqrt{2}} - \frac{1}{\sqrt{2}}\left(t - \frac{\pi}{4}\right) \\
&\rightarrow \frac{\partial x}{\partial y} = +1 \\
&\rightarrow \frac{\partial y}{\partial x} = -1 \\
&\rightarrow \frac{\partial x}{\partial z} = \frac{\partial z}{\partial x} = \frac{\partial y}{\partial z} = \frac{\partial z}{\partial y} = 0 \\
&\sigma(\mathbf{N}_{\mathbf{x_0}(\frac{3\pi}{4})}, \mu_{\mathbf{x_0}(\frac{3\pi}{4})} + \mathbf{x}\beta_{\mathbf{x_0}(\frac{3\pi}{4})}) = \{1, 2\}
\end{aligned}
\tag{24}
$$

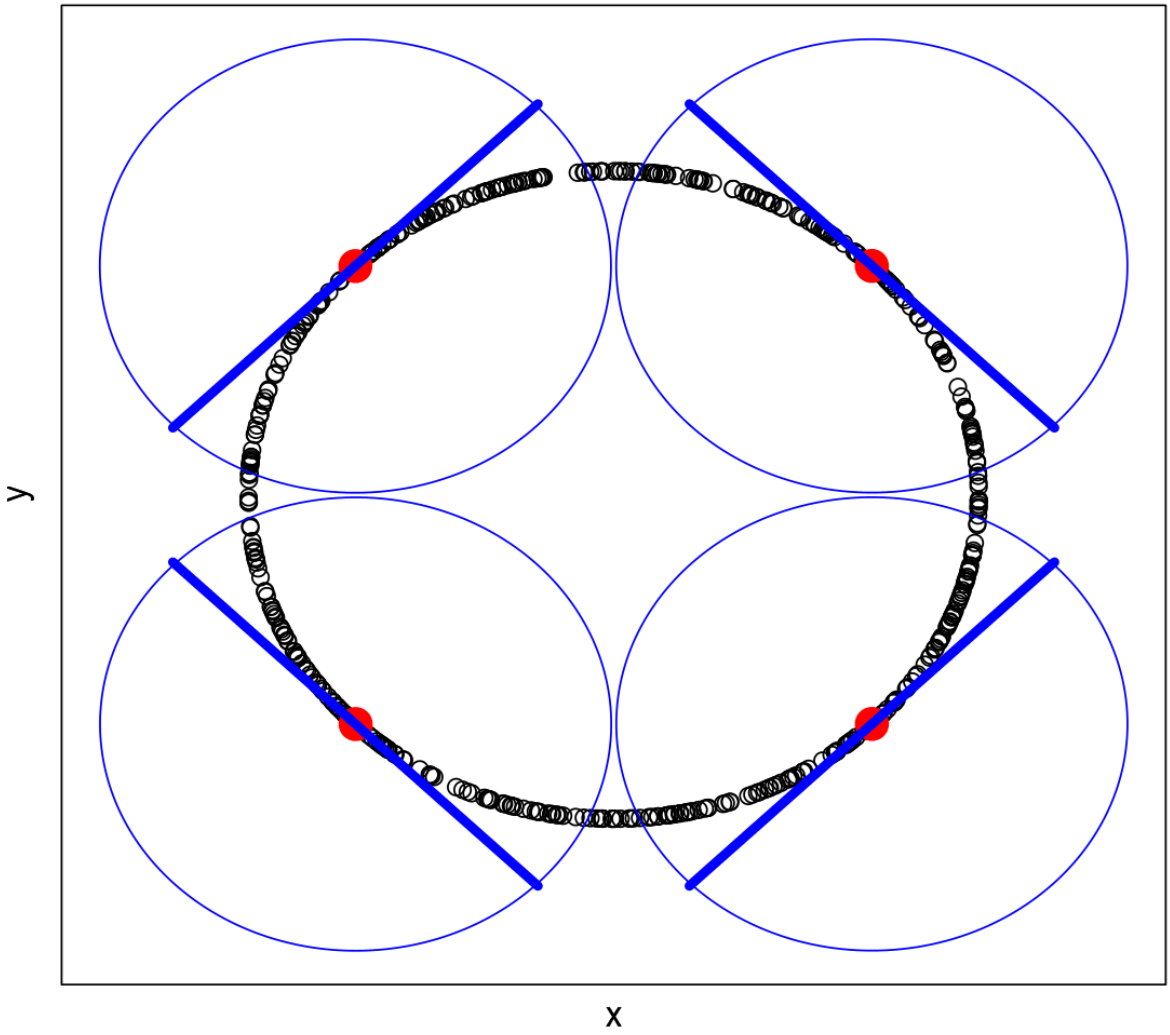
- $t_0 = \frac{-3\pi}{4}$

$$
\begin{aligned}
x = sin(t) &\approx +\frac{1}{\sqrt{2}} - \frac{1}{\sqrt{2}}\left(t + \frac{\pi}{4}\right) \\
y = cos(t) &\approx +\frac{1}{\sqrt{2}} + \frac{1}{\sqrt{2}}\left(t + \frac{\pi}{4}\right) \\
&\rightarrow \frac{\partial x}{\partial y} = +1 \\
&\rightarrow \frac{\partial y}{\partial x} = +1 \\
&\rightarrow \frac{\partial x}{\partial z} = \frac{\partial z}{\partial x} = \frac{\partial y}{\partial z} = \frac{\partial z}{\partial y} = 0 \\
&\sigma(\mathbf{N}_{\mathbf{x_0}(\frac{-3\pi}{4})}, \mu_{\mathbf{x_0}(\frac{-3\pi}{4})} + \mathbf{x}\beta_{\mathbf{x_0}(\frac{-3\pi}{4})}) = \{1, 2\}
\end{aligned}
\tag{25}
$$

In the end, the union of the local selection leads to the same set of selected variables:

$$
\begin{aligned}
\sigma(\mathbf{X}, f \circ g) &= \bigcup_{\forall \mathbf{x_0}} \sigma(\mathbf{N}_{\mathbf{x_0}}, \mu_{\mathbf{x_0}} + \mathbf{x}\beta_{\mathbf{x_0}}) \\
&= \bigcup \{\{1, 2\}, \{1, 2\}, \{1, 2\}, \{1, 2\}\} \\
&= \{1, 2\} \\
&= \mathcal{S}^*.
\end{aligned}
\tag{26}
$$

Fig. 2. Local linear approximation of a non-linear manifold at $t_0 = \left\{\frac{\pi}{4}, \frac{-\pi}{4}, \frac{3\pi}{4}, \frac{-3\pi}{4}\right\}$



Notes: The red data points are the four points $\mathbf{x_0}(t_0)$*, the blue circles represents the neighbourhoods around these points, and the blue lines are the local linear approximates of the circle.*

That separability assumption is only needed for a local approach, and is not required for a global estimator. Doing so, divides the overall selection problem into lots of local selection problems, easier to solve. The parameters from the local linear models can be estimated via eigenvalue decomposition, known also as Principal Components Analysis (PCA) in the literature. This is the linear equivalent of the non-linear manifold model. It is a natural choice as locally the r-manifold is a r-dimensional plane embedded into $\mathbb{R}^p$, and the first r columns of the eigenvectors $\mathbf{V} \in \mathbb{R}^{p \times p}$ project the data onto a r-dimensional plane. Each eigenvector is normalized $\|\mathbf{V}_j\| = 1 \quad \forall j = 1, ..., p$, and they are orthogonal to each other $VV' = Id$.

This is required for identification of the parameters. The low dimensional projection can be mapped back using the transposed of the same vectors. Doing so, produces the best linear approximation in a total least square sense.

$$\mathbf{x} \approx (\mathbf{x} - \mu_{\mathbf{x}})\mathbf{V}_{1:r}\mathbf{V}'_{1:r} + \mu_{\mathbf{x}}. \tag{27}$$

The last step is to define a selection operator based on these parameters. There have been several attempts in the literature to perform variable selection in that case. The Sparse PCA of Zou et al. (2006) uses a composite L1-norm and L2-norm constraint on the eigenvectors. However it would requires the knowledge of r, which is not the case here. The Graphical LASSO of Friedman et al. (2008) is the natural extension of the LASSO selection operator to covariance matrices. It is a serious candidate to perform the local linear selection, but it does not operate on eigenvectors, it operates directly on the covariance matrix. Thus, it does not take the dimension reduction into accounts. Also, even if there have been a very great development of algorithms to solve the graphical LASSO efficiently, the computational burden of many such estimations is large. Finally, the Penalized Matrix Decomposition (PMD) Witten et al. (2009), which uses only L1-norm penalty on the eigenvectors, but also requires the knowledge of r. All the mentioned methods uses norm penalties to achieve sparsity, which have no analytical solutions and therefore are longer to compute.
Generally speaking, selection directly on all the eigenvectors is not viable because they are normalized. This means that elements of $\mathbf{V}_{r+1:p}$ are on the same scale as $\mathbf{V}_{1:r}$, between 0 and 1. And that affects the selection. Remind that r is not known nor estimated. Nevertheless, the eigenvalues $\mathbf{L}$ (in diagonal matrix form) do carry some information about r. They measure the lengths of the principal directions of a plane. Locally the r-manifold is a r-dimensional plane embedded into $\mathbb{R}^p$. Eigenvalues are sorted in decreasing order, so there should be a significant difference between $L_{r,r}$ and $L_{r+1,r+1}$. So a way to account for the uncertainty on r is to use the eigenvectors weighted by the square root of eigenvalues: $\mathbf{VL^{1/2}}$. There have also been propositions in the literature to try to infer r. In which case, I could use that estimate and keep only the corresponding columns of $\mathbf{V}$. Still, the solution with $\mathbf{VL^{1/2}}$ presented before is simpler and faster to compute. Then, to achieve sparsity I will only use a simple threshold on each element of $\mathbf{VL^{1/2}}$. This is solution is possible, instead of using L1 or L2 norms, because both orthogonality and shrinkage are already there from the eigenvalue decomposition. Having normalized the eigenvectors is equivalent to a shrinkage, in the same way as a ridge estimator. Also, the eigenvectors are orthogonal by construction. In a classical regression problem, the non-orthogonality of the parameters is the reason why

the LASSO has no analytical solutions. Thus, I propose a new linear selection operator: the EigenThresholding (ET).

$$\begin{aligned} \mathbf{\Sigma} &= \mathbf{VLV}' \\ V_{ij} &= V_{ij}\mathbb{1}_{|V_{ij}L_{j,j}^{1/2}|>\theta}. \end{aligned} \tag{28}$$

The parameter θ is a penalty that controls the amount of sparsity. To the best of my knowledge, this selection estimator has not been proposed yet. Its properties as a linear selection estimator are examined in the simulations, along with the other linear selection operators.

2.2. Local Linear Manifold Selection

I perform non-parametric variable selection on manifolds using a linear selection operator locally, i.e. on neighbourhoods of the data. Solving the selection at the global level is hard because of the unknown possible non-linearities. However, simple linear selection operators have been well studied and greatly improved over time. The major restriction they suffer is the linearity assumption. To ensure linearity holds, I slice the global non-linear manifold into lots of neighbourhoods. Inside a given neighbourhood, a simple linear selection operator should work very well. Doing so produces a selection $\hat{\mathcal{S}}$ for each neighbourhood. As I want a single selection in the end, I just merge all the selections by averaging them. Therefore, the original problem of performing selection on non-linear manifold is solved.
The standard approach, if I were to follow the literature, would be to use a regression operator to obtain an estimate of the manifold, and then derive a selection $\hat{\mathcal{S}}$. But the crucial point is that I perform selection without ever explicitly estimate the manifold. The full procedure is described in Algorithm 1.

A neighbourhood around a given observation $\mathbf{x}_i$ (a point on the manifold) contains k neighbours, and is defined as:

$$\mathbf{N}_{i,k}(\mathbf{x}_i) = \left\{\mathbf{x} \in \mathbf{X} : \ \|\mathbf{x} - \mathbf{x}_i\| \leq \|\mathbf{x} - \mathbf{x}_i\|_{(k)}\right\} \tag{29}$$

The hyper-parameter k needs to be tuned. In each of these neighbourhoods, I apply a selection operator $\sigma(\mathbf{X}, \Omega)$. In the previous section, I have motivated a choice for a specific estimator. As it was discussed there is not a single estimator suitable for this task. So here I present the general method, using any linear selection operator. Most of them solve a constrained optimisation problem, using a penalty function on the parameters to be estimated.

The constraint ensures sparsity in the parameters, which induces the selection. The penalty function usually involves a penalty parameter θ, it controls the amount of sparsity.
I choose to decompose the correlation matrix for three reasons. First, it standardizes the coefficients associated with each variable. There is no bias towards a candidate with a larger scale. This would have been the case using a covariance matrix. Second, it makes no arbitrary assumption on the dependent variable. All candidates possibly affect each other. This would not be true using the simple linear regression. Third, this implies that all candidates are measured with proportional noise $Var[\epsilon_j] = \upsilon Var[X_j]$. If I were to use a covariance matrix, this would imply constant variance of the noise: $Var[\epsilon_j] = \upsilon$. However, keep in mind that this assumption holds only locally. It can vary across the sample, as we compute it in different neighbourhoods.

The optimization problem in each neighbourhood $\mathbf{N}_{i,k}$ is:

$$\begin{aligned} \mathbf{R}_{\mathbf{N}_{i,k}} &= Cor\left(\mathbf{N}_{i,k}\right) \\ \underset{\tilde{\mathbf{R}}_{\mathbf{N}_{i,k}}}{argmin} &\left\|\mathbf{R}_{\mathbf{N}_{i,k}} - \tilde{\mathbf{R}}_{\mathbf{N}_{i,k}}\right\| + \text{penalty}\left(\tilde{\mathbf{R}}_{\mathbf{N}_{i,k}}, \theta\right) \end{aligned} \tag{30}$$

Then I record the selected variables:

$$\hat{\mathcal{S}}_{\mathbf{N}_{i,k},\theta} = \left\{ j | \tilde{\mathbf{R}}_{\mathbf{N}_{i,k}}(j,s) \neq 0, \forall s \neq j \right\}.$$

The selections $\hat{\mathcal{S}}_{\mathbf{N}_{i,k},\theta}$ are turned into boolean vectors:

$$\Psi_{\mathbf{N}_{i,k},\theta} = \mathbb{1}_{j \in \hat{\mathcal{S}}_{\mathbf{N}_{i,k},\theta}}$$

There is a vector Ψ for each neighbourhood and for each value of θ.

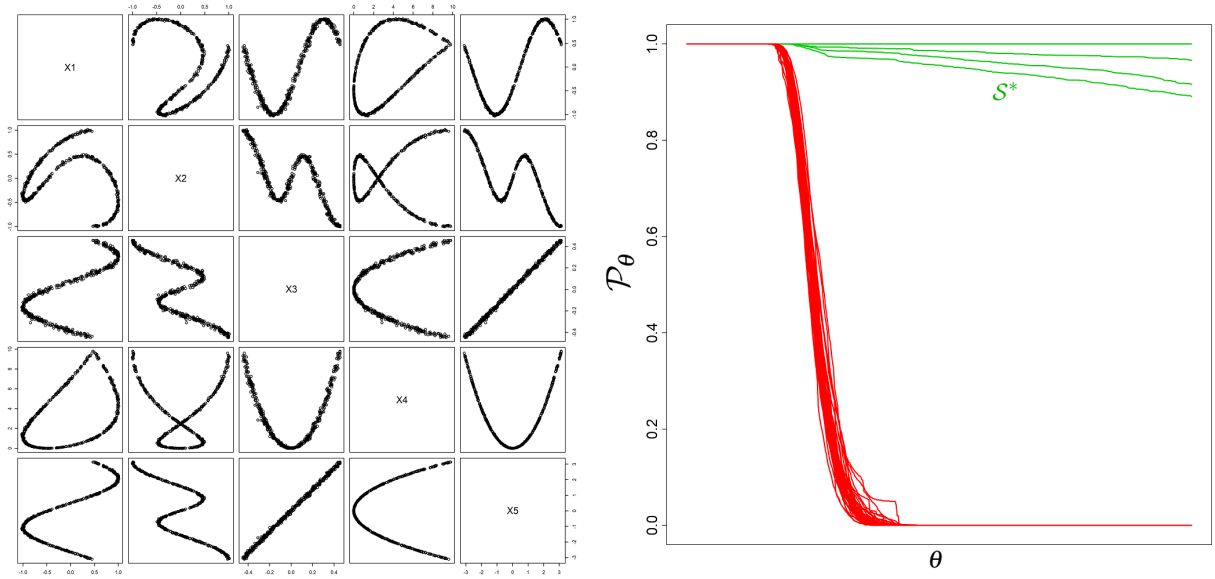
Finally, I average the n boolean selections $\Psi_{\mathbf{N}_{i,k},\theta}$.

$$\mathcal{P}_\theta = \frac{1}{n} \sum_{i=1}^{n} \Psi_{\mathbf{N}_{i,k},\theta}.$$

The vector $\mathcal{P}_\theta$ measures for each candidate variable, the proportion of neighbourhoods in which it was selected. It is simple to see that $\mathcal{P}_\theta \in [0,1]^p$. So I might see it as a "pseudo-probability", for a given variable, to belonging to the manifold.
We can connect the matrix $\mathcal{P}$ to very usual objects known in the variable selection literature

as "selection paths". Usually, one represents the behaviour of the parameter of interest as a function of the penalty parameters θ. Here the same is made but with $\mathcal{P}$, in the same spirit as the "probability paths" of StabSel (Meinshausen and Bühlmann (2010)). The exact definition of θ is narrowed to the selection estimator $\sigma(\mathbf{X}, \Omega)$. For some estimator, there have been procedure proposed to choose θ optimally. This issue is discussed in section 2.4. The algorithm for the LLMS procedure is presented in Algorithm 1.

Fig. 3. Example of non-linear DGP and LLMS selection paths



Notes: The simulation is made with $|\mathcal{S}^*| = 5$, $p = 50$ *and a single dimensional non-linear manifold. Green paths correspond to the variables that belong to the true set* $\mathcal{S}^*$, *the irrelevant ones are coloured in red. The green paths are close to* 1 *and the red ones to* 0, *so it is quite obvious from the chart which variables should be selected.*

Algorithm 1 Local Linear Manifold Selection (LLMS1)

input $\mathbf{X}$, k, Θ, $\sigma(\mathbf{X}, \Omega)$

for $i = 1$ to n **do**

Search for the neighbourhood $\mathbf{N}_{i,k}(X_i)$

for $\theta \in \Theta$ **do**

Estimate the sparse correlation matrix $\tilde{\mathbf{R}}_{\mathbf{N}_{i,k}}$ using $\sigma(\mathbf{N}_{i,k}, \Omega)$

Record active variables in $\hat{\mathcal{S}}_{\mathbf{N}_{i,k},\theta} = \left\{ j | \tilde{\mathbf{R}}_{\mathbf{N}_{i,k}}(j,s) \neq 0, \forall s \neq j \right\}$.

Turn $\hat{\mathcal{S}}$ into boolean vectors $\Psi_{\mathbf{N}_{i,k},\theta} = \mathbb{1}_{j \in \hat{\mathcal{S}}_{\mathbf{N}_{i,k},\theta}}$

end for

end for

Compute the final selection estimator $\mathcal{P}_\theta = \frac{1}{n} \sum_{i=1}^{n} \Psi_{\mathbf{N}_{i,k},\theta}$.

output $\mathcal{P}_\theta$

2.3. Randomization of Distances for Large Candidate Set

There is an issue when constructing neighbourhoods as the candidate set increases. I cannot measure distances onto the manifold directly because $\mathcal{S}^*$ is unknown. I only observe distances in the $\mathbb{R}^p$ space of the candidate variables. So, I can only compute noisy versions of the true distances. It disturbs the ordering of the distances and therefore the neighbourhood is not the same. The local linear assumption might not hold anymore, and as a consequence, it may affect the selection.

A solution to obtain measures of distances closer to the desired ones is to use a modified version of kNN known as Random-kNN (RkNN) (Ho, 1998). Here, I use random subsets $B \subset A \equiv \{j \in \mathbb{N}, j \leq p\}$ of the candidates variable set A for computing distances.

$$
\begin{aligned}
D^{kNN} &= \|\mathbf{x} - \mathbf{x}_i\| \quad \forall \mathbf{x} \in \mathbf{X}_A. \\
N_{i,k}^{kNN} &= \left\{ \mathbf{x} \in \mathbf{X} : \; D^{kNN} \leq D_{(k)}^{kNN} \right\}
\end{aligned}
$$

$$
\begin{aligned}
D^{RkNN} &= \|\mathbf{x} - \mathbf{x}_i\| \quad \forall \mathbf{x} \in \mathbf{X}_B. \\
N_{i,k}^{RkNN} &= \left\{ \mathbf{x} \in \mathbf{X} : \; D^{RkNN} \leq D_{(k)}^{RkNN} \right\}
\end{aligned}
$$

where the subscript (k) indicates the k^{th} order statistics.

Proposition 1. *If $\mathcal{S}^* \in B$ and $m \equiv |B| < |A| = p$ then RkNN yields closer neighbourhoods on $\mathcal{M}$ than kNN.*

$$max\left(\left\|N_{\mathcal{S}^*}^{kNN} - \mathbf{x}_i\right\|\right) \geq max\left(\left\|N_{\mathcal{S}^*}^{RkNN} - \mathbf{x}_i\right\|\right)$$

Proof. Let $\mathbf{X}$ be a matrix of p random variables with n observations, so that $\mathbf{x}_i = f \circ g(\mathbf{x}_i)\ \forall i \in \{1, ..., n\}, \mathbf{x}_i \in \mathbf{X}$. Now let's divide the full set of candidates $\mathbf{X}$ into 3 non-overlapping matrices $\boldsymbol{\mathcal{X}}$, $\boldsymbol{\mathcal{Y}}$ and $\boldsymbol{\mathcal{Z}}$ composed of respectively p_x, p_y and p_z random vectors such $p_x + p_y + p_z = p$. Only $\boldsymbol{\mathcal{X}} \in \mathcal{S}^*$, while $(\boldsymbol{\mathcal{Y}}, \boldsymbol{\mathcal{Z}}) \notin \mathcal{S}^*$). Let $\mathbf{x}_0 \in \boldsymbol{\mathcal{X}}$ be a given random point on the manifold. Wlog, let $\mathbf{x}_1 \in \boldsymbol{\mathcal{X}}$ and $\mathbf{x}_2 \in \boldsymbol{\mathcal{X}}$ be respectively the first and second closest point to $\mathbf{x}_0$:

$$\sum^{p_x}(\mathbf{x}_1 - \mathbf{x}_0)^2 < \sum^{p_x}(\mathbf{x}_2 - \mathbf{x}_0)^2 < \sum^{p_x}(\mathbf{x}_i - \mathbf{x}_0)^2\ \forall i \neq \{0, 1, 2\}, \tag{31}$$

where the power of vector is defined element-wise. W.l.o.g. we can write:

$$\sum^{p_x}(\mathbf{x}_2 - \mathbf{x}_0)^2 = \sum^{p_x}(\mathbf{x}_1 - \mathbf{x}_0)^2 + \epsilon_x$$

with $\epsilon_x > 0$.

Random-kNN consists in computing distances from random subset taken from $\mathbf{X}$, e.g. only $\boldsymbol{\mathcal{X}}$. Now let's define $\mathbf{y} \in \boldsymbol{\mathcal{Y}}$ and $\mathbf{z} \in \boldsymbol{\mathcal{Z}}$, given points which are not on the manifold. Now lets take two sets of indices A and B such that indices of $\mathbf{X}$ that correspond to $(\boldsymbol{\mathcal{X}}, \boldsymbol{\mathcal{Y}})$ are in B and A contains all $\mathbf{X}$, so $B \subset A$. Now let's compare the distances w.r.t to $(\mathbf{x}_0, \mathbf{y}_0, \mathbf{z}_0)$. The condition so the ordering of distances in equation 31 is preserved is

$$\sum^{p_x}(\mathbf{x}_1 - \mathbf{x}_0)^2 + \sum^{p_y}(\mathbf{y}_1 - \mathbf{y}_0)^2 < \sum^{p_x}(\mathbf{x}_2 - \mathbf{x}_0)^2 + \sum^{p_y}(\mathbf{y}_2 - \mathbf{y}_0)^2 + \sum^{p_z}(\mathbf{z}_2 - \mathbf{z}_0)^2$$

Which can be rewritten as

$$\left[\sum^{p_x}(\mathbf{x}_1 - \mathbf{x}_0)^2 - \sum^{p_x}(\mathbf{x}_2 - \mathbf{x}_0)^2\right] < \left[\sum^{p_y}(\mathbf{y}_2 - \mathbf{y}_0)^2 - \sum^{p_y}(\mathbf{y}_1 - \mathbf{y}_0)^2\right] + \sum^{p_z}(\mathbf{z}_2 - \mathbf{z}_0)^2. \tag{32}$$

The first quantity tends to zero as the manifold is more and more sampled ($n \to \infty$).

$$\lim_{n\to\infty}\left[\sum^{p_x}(\mathbf{x}_1 - \mathbf{x}_0)^2 - \sum^{p_x}(\mathbf{x}_2 - \mathbf{x}_0)^2\right] \equiv \epsilon_x \to 0.$$

The distances between two points from $\boldsymbol{\mathcal{Y}}$ depends on its distribution. If we assume each

vector of $\boldsymbol{\mathcal{Y}}$ to be i.i.d. and having finite first and second moments, then their distances are also i.i.d. with finite first and second moment. We can then write

$$\mathbb{E}\left[\sum^{p_y}(\mathbf{y}_2 - \mathbf{y}_0)^2\right] = \mathbb{E}\left[\sum^{p_y}(\mathbf{y}_1 - \mathbf{y}_0)^2\right] = \mu_y$$

Then, it follows that

$$\lim_{p_y \to \infty}\left[\sum^{p_y}(\mathbf{y}_2 - \mathbf{y}_0)^2 - \sum^{p_y}(\mathbf{y}_1 - \mathbf{y}_0)^2\right] \equiv \epsilon_y \to 0.$$

Finally, since the sum of distances among $\boldsymbol{\mathcal{Z}}$ only grows with the number of elements it contains

$$\lim_{p_z \to \infty} \sum^{p_z}(\mathbf{z}_2 - \mathbf{z}_0)^2 \equiv \epsilon_z \to \infty.$$

So we can rewrite equation 32

$$\epsilon_x - \epsilon_y < \epsilon_z.$$

The probability that this condition is true tends to one as the number of irrelevant variables grows

$$\lim_{\substack{p_y \to \infty \\ p_z \to \infty}} \mathbb{P}\left(\epsilon_z > \epsilon_x - \epsilon_y\right) = 1.$$

□

But how can I ensure that $\mathcal{S}^* \in B$ and $m \equiv |B| < |A| = p$? This is an ill-posed problem as it would require the knowledge of $\mathcal{S}^*$.

One solution is to start by sampling the subset B from $\{j \in \mathbb{N}, j \leq p\}$, using a uniform distribution π. It makes no a priori on which variables are more likely to belong to the manifold. Then, I can update that sampling probability π, using selection probabilities from previous steps. As the process goes on, the subset will be drawn from most probably active variables. The ordering of the data may have an impact here, if the selection is biased in the first neighbourhoods. Still, the sampling probability might correct itself after some iterations. One solution to reduce the dependence on initialisation is to proceed by blocks; instead of updating π after each observation i, π could be updated for instance every 10 observations. Finally, we have to ensure that π sums to one, so in each step, it is divided by its sum. This is also useful, as it will never set any of its value to zero exactly. This means that any candidate variable still has a chance to be selected in the subset. Therefore, I propose a second version of the LLMS algorithm.

Algorithm 2 Local Linear Manifold Selection with random subsets (LLMS2)

input $\mathbf{X}$, k, Θ, $\sigma(\mathbf{X}, \Omega)$

Start with equal subset probability $\pi = \frac{1}{p}$

for $i = 1$ to n **do**

Choose a subset of $\{1, ..., p\}$ from probability the probability vector π

Search for the neighbourhood $\mathbf{N}_{i,k}(\mathbf{X}_i)$, computing distances w.r.t to the subset.

for $\theta \in \Theta$ **do**

Estimate the sparse correlation matrix $\tilde{\mathbf{R}}_{\mathbf{N}_{i,k}}$ using $\sigma(\mathbf{N}_{i,k}, \Omega)$

Record active variables in $\hat{\mathcal{S}}_{\mathbf{N}_{i,k},\theta} = \left\{ j | \tilde{\mathbf{R}}_{\mathbf{N}_{i,k}}(j, s) \neq 0, \forall s \neq j \right\}$.

Turn $\hat{\mathcal{S}}$ into boolean vectors $\Psi_{\mathbf{N}_{i,k},\theta} = \mathbb{1}_{j \in \hat{\mathcal{S}}_{\mathbf{N}_{i,k},\theta}}$

end for

Update the subset probability vector $\pi = \pi + \frac{1}{i} \sum_{i'=1}^{i} \Psi$

end for

Compute the final selection estimator $\mathcal{P}_\theta = \frac{1}{n} \sum_{i=1}^{n} \Psi_{\mathbf{N}_{i,k},\theta}$.

output $\mathcal{P}_\theta$

2.4. Optimal Choice for Hyper-parameters

2.4.1. Choice of penalty parameter

The inclusion probability $\mathcal{P}$ depends strongly on the amount of penalty applied θ. Ideally, we would like to find the optimal amount of penalty θ that minimises the difference between $\mathcal{S}^*$ and $\hat{S}$.

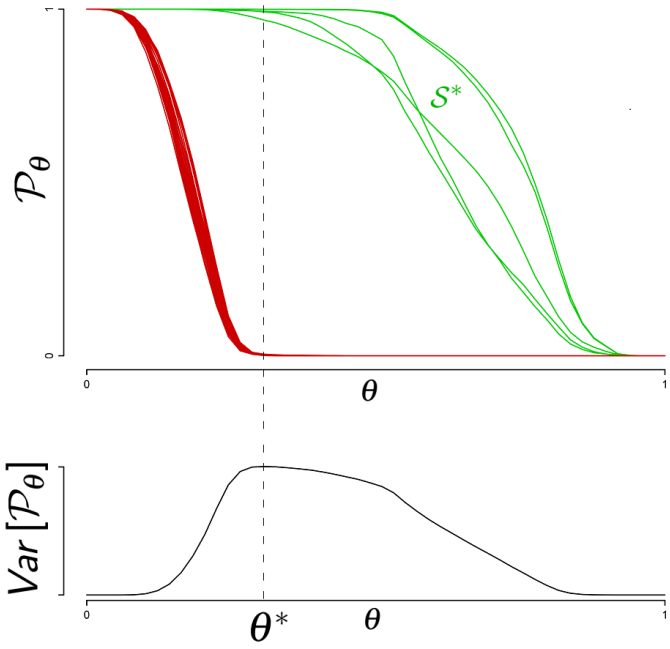
From a simple look at figure 4, this difference is minimized when the discrepancy among the probabilities of inclusion is at its maximum.

This result suggests the following optimal rule:

$$\theta^* = \underset{\theta}{argmax} \left(Var \left[\mathcal{P}_\theta \right] \right) \tag{33}$$

Figure 4 illustrates the selection paths $\mathcal{P}$ as well as the optimal rule for the penalty parameter.

Fig. 4. Choice for optimal penalty



Notes: The simulation is made with $|\mathcal{S}^*| = 5$ *and* $p = 50$.

The optimal choice for the penalty parameter θ^* is to maximize the variance of the probabilities of inclusion. It is displayed as a vertical dashed line, through the two plots. At θ^* the probability of inclusion $\mathcal{P}$ for the truly active variables ($\in \mathcal{S}^*$) are almost one, and almost zero for truly inactive variables ($\notin \mathcal{S}^*$).

Also, when $\theta \to 0$ there is no penalty, and thus all the p candidate variables are selected: $\mathcal{P}_{\theta \to 0} = 1$. When $\theta \to \infty$ there is infinite penalty, and thus no candidate variables are selected: $\mathcal{P}_{\theta \to \infty} = 0$. This implies: $Var\left[\mathcal{P}_{\theta=0}\right] = Var\left[\mathcal{P}_{\theta=\infty}\right] = 0$. However for intermediate amount of penalty $0 < \theta < \infty$ we have $Var\left[\mathcal{P}_{\theta=\infty}\right] \geq 0$. As $\mathcal{P}$ is decreasing in θ there is a unique maximum.

2.4.2. Reducing the sensitivity to the neighbourhood's size parameter via the q-kNN Algorithm

It is known that kNN is sensitive to the hyper-parameter k (Sun and Huang, 2010). Intuitively, the optimal value of k would be the one that produces the largest neighbourhoods, such that the linear assumption holds. It follows that this optimal value is dependent on the shape of the manifold, which is unknown. But even in the case I could measure distances on the $\mathbb{R}^m$ space, these are still not the same as the ones on $\mathcal{M}$. Indeed, take for example

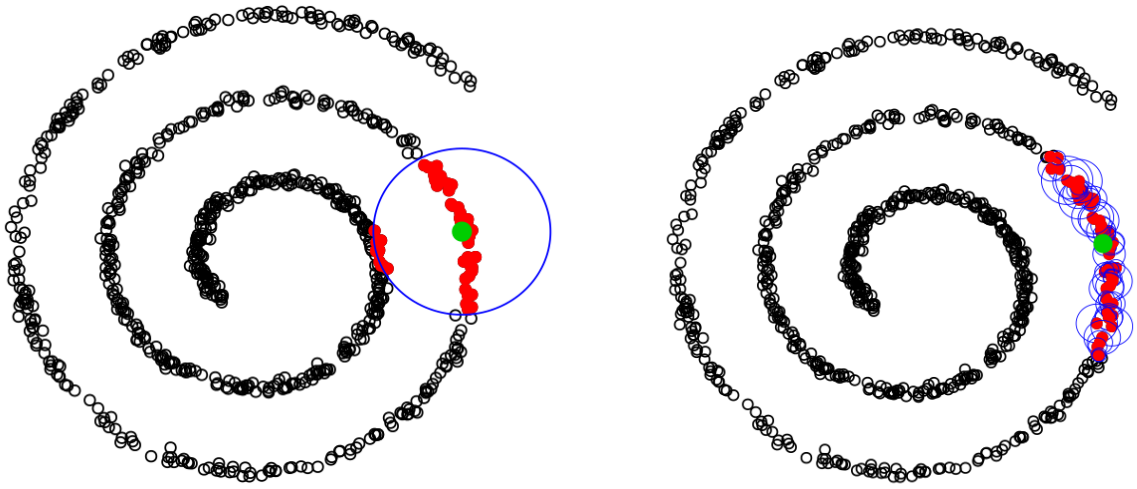
the very usual Euclidean distance. If the manifold is very non-linear, increasing euclidean distances will produce neighbourhoods in which points are very apart on the manifold (see Figure 5(b)). However, small euclidean distances approximate well the distances on the manifold. I can use this property to construct better neighbourhoods. This does not solve the problem of the optimal value of k but still reduces the sensitivity of the algorithm in case the manifold is very non-linear.

So I propose a refined version of kNN, in which neighbourhoods are constructed sequentially from small distances only. The basic idea is to take the $q < k$ nearest points to $\mathbf{x}_i$ and then expanding along with the next q nearest points to the ones previously selected, etc... until the neighbourhood's size is k. This better ensures the local linearity assumption and tends to reduce the sensitivity to k. Therefore I call the new version q-kNN.

$$\begin{aligned}
\mathbf{N}_{i,q}(\mathbf{x}_i) &= \left\{\mathbf{x} \in \mathbf{X} :\ \|\mathbf{x} - \mathbf{x}_i\| \leq \|\mathbf{x} - \mathbf{x}_i\|_{(q)}\right\} \\
\mathbf{N}_{i,>q}(\mathbf{x}_i) &= \left\{\mathbf{x} \in \mathbf{X} :\ \|\mathbf{x} - \mathbf{N}_{i,q}(\mathbf{x}_i)\| \leq \|\mathbf{x} - \mathbf{N}_{i,q}(\mathbf{x}_i)\|_{(q)}\right\} \\
\mathbf{N}_{i,k}(\mathbf{x}_i) &= \left\{\mathbf{x} \in \mathbf{X} :\ \|\mathbf{x} - \mathbf{N}_{i,>q}(\mathbf{x}_i)\| \leq \|\mathbf{x} - \mathbf{N}_{i,>q}(\mathbf{x}_i)\|_{(q)}\right\}
\end{aligned} \tag{34}$$

The first neighbourhood contains q points. Then I compute the q-nearest points of these q points, so the second neighbourhoods as q^2 points. However doubles are removed, so I end up with a second neighbourhood having between $q+1$ and q^2 points. So in the n^{th} step the n^{th} neighbourhood of size at least $q+n$ and at most q^n. With a not too small value of q this sequence quickly converges to a neighbourhood with k points.

Fig. 5. Illustration of the revisited q-kNN algorithm.



(a) kNN (b) q-kNN

Notes: The aim of both algorithms is to find neighbourhood (red dots) around a given data point (green dot): kNN (left) is using simple euclidean distances while the revisited q-kNN (right) is using compounded euclidean distances. The neighbourhoods both contain k points in the end.

The q-kNN comes at the cost of computations. For each neighbourhood, distances w.r.t to all the points have to be evaluated, several times. This can be shortened by computing all pairwise distances once. Then the remaining computations are only orderings, which are usually quite fast. However, if q-kNN is used at the same with time RkNN, it is worse. The pairwise distances have to be evaluated at each step, thus it is very slow.

2.5. Selection Properties

In this part, I study some properties of the selection estimator $\sigma_{LLMS}(\mathbf{X}, f \circ g)$. The usual selection consistency property is known in the literature as the "Oracle Property" (Fan and Li., 2001). The weak version of the Oracle Property is stated as follow:

$$\lim_{n \to \infty} P(\hat{\mathcal{S}} = \mathcal{S}^*) = 1. \tag{35}$$

The strong version requires also the convergence of the estimated model, as established by Fan and Li. (2001). In that case, that would mean convergence of $\hat{f} \circ \hat{g}$. But these are never explicitly estimated. Also, since I am primarily interested in the selected set, I will only focus on the Weak Oracle Property. Nevertheless, the Oracle Property in this form is not easy to work with. So first, I define the boolean version of the set of active variables $\mathcal{S}^*$

denoted Ψ^*. It is a vector of with p elements.

$$\begin{aligned} \mathcal{S}^* :=& \ \{j : \exists j' \neq j, \frac{\partial X_{j'}}{\partial X_j} \neq 0\} \in \mathbb{N} \\ \Psi^* =& \ \mathbb{1}_{j \in \mathcal{S}^*} \in \{0,1\}^p. \end{aligned} \tag{36}$$

Their empirical local counterparts are defined as:

$$\begin{aligned} \hat{\mathcal{S}}_{\mathbf{N_i}} :=& \ \{j : \exists j' \neq j, \frac{\partial X_{j'}}{\partial X_j} \neq 0, \mathbf{X} \in \mathbf{N_i}\} \in \mathbb{N} \\ \hat{\Psi}_{\mathbf{N_i}} =& \ \mathbb{1}_{j \in \hat{\mathcal{S}}_{\mathbf{N_i}}} \in \{0,1\}^p. \end{aligned} \tag{37}$$

The Local Linear Manifold Selection operator, denoted $\mathcal{P}$, is simply the average of the local selection operators after the boolean transformation:

$$\mathcal{P} = \frac{1}{n} \sum_{i=1}^{n} \hat{\Psi}_{\mathbf{N_i}} \in [0,1]^p.$$

Now, the Weak Oracle Property can be restated, so it is easier to work with the LLMS estimator.

$$\begin{aligned} & \lim_{n \to \infty} P(\hat{\mathcal{S}} = \mathcal{S}^*) = 1 \\ \Leftrightarrow & \lim_{n \to \infty} P(\left\| \hat{\Psi} - \Psi^* \right\|_2^2 = 0) = 1 \\ \Leftrightarrow & \lim_{n \to \infty} P(\left\| \hat{\Psi} - \Psi^* \right\|_2^2 \leq \nu) = 1 \ \forall \nu \geq 0. \end{aligned} \tag{38}$$

I will refer to 38 as the Continuous Weak Oracle Property. Then, the LLMS estimator is consistent in selection if:

$$\lim_{n \to \infty} P(\|\mathcal{P} - \Psi^*\|_2^2 \leq \nu) = 1 \ \forall \nu \geq 0. \tag{39}$$

However, since $\mathcal{P}$ is the average of a linear selection operator $\Psi_{\mathbf{N_i}}$, its selection properties have to be inherited from the selection properties of $\Psi_{\mathbf{N_i}}$. The properties of the latter may vary substantially from one estimator to the other. So, I cannot directly deduce the properties of $\mathcal{P}$ in a general way. Nevertheless, the continuous weak oracle is illustrated using simulations in section 4.3. The results seem to indicate that the property holds.

Still, there is one thing of interest, which is to show the behaviour of $\mathcal{P}$ w.r.t to any given local estimator. Therefore, I make three propositions to show that $\mathcal{P}$ is never worse, in terms of selection consistency, than its local linear estimator $\Psi_{\mathbf{N_i}}$. Following these results,

by choosing a consistent local selection estimator ensures the selection consistency of the LLMS estimator.

Proposition 2. *The LLMS estimator $\mathcal{P}$ is closer to the true set Ψ^* than the underlying local linear selection estimator $\Psi_{\mathbf{N_i}}$.*

$$P(\|\mathcal{P} - \Psi^*\|_2^2 \leq \nu) \geq P\left(\frac{1}{n}\sum_{i=1}^{n}\left\|\hat{\Psi}_{\mathbf{N_i}} - \Psi^*\right\|_2^2 \leq \nu\right) \tag{40}$$

Proposition 3. *When the level of type I and type II error α tends to zero, the underlying local linear selection estimator $\Psi_{\mathbf{N_i}}$ is closer to the true set Ψ^* than the LLMS estimator $\mathcal{P}$. When $\alpha = 0$, both are perfectly consistent.*

$$\lim_{\alpha \to 0} P\left(\left\|\hat{\Psi}_{\mathbf{N_i}} - \Psi^*\right\|_2^2 \leq \|\mathcal{P} - \Psi^*\|_2^2\right) \to 1 \tag{41}$$

Proposition 4. *When the number of variables increases, the LLMS estimator $\mathcal{P}$ is closer to the true set Ψ^* than the underlying local linear estimator $\Psi_{\mathbf{N_i}}$.*

$$\lim_{p \to \infty} P\left(\left\|\hat{\Psi}_{\mathbf{N_i}} - \Psi^*\right\|_2^2 \leq \|\mathcal{P} - \Psi^*\|_2^2\right) \to 0 \tag{42}$$

Proof of Proposition 2 is pretty simple, following the triangle inequality. First, I expand each of the norms.

$$\begin{aligned}
\|\mathcal{P} - \Psi^*\|_2^2 &= \left\|\frac{1}{n}\sum_{i=1}^{n}\hat{\Psi}_{\mathbf{N_i}} - \Psi^*\right\|_2^2 \\
\left\|\frac{1}{n}\sum_{i=1}^{n}\hat{\Psi}_{\mathbf{N_i}} - \Psi^*\right\|_2^2 &= \left(\frac{1}{n}\sum_{i=1}^{n}\hat{\Psi}_{\mathbf{N_i}} - \Psi^*\right)^2 \\
&= \left(\frac{1}{n}\sum_{i=1}^{n}\hat{\Psi}_{\mathbf{N_i}}\right)^2 + (\Psi^*)^2 - 2\left(\frac{1}{n}\sum_{i=1}^{n}\hat{\Psi}_{\mathbf{N_i}}\right)\Psi^* \\
\frac{1}{n}\sum_{i=1}^{n}\left\|\hat{\Psi}_{\mathbf{N_i}} - \Psi^*\right\|_2^2 &= \frac{1}{n}\sum_{i=1}^{n}\left(\hat{\Psi}_{\mathbf{N_i}} - \Psi^*\right)^2 \\
&= \left(\frac{1}{n}\sum_{i=1}^{n}\hat{\Psi}_{\mathbf{N_i}}^2\right) + n(\Psi^*)^2 - \frac{1}{n}\sum_{i=1}^{n}2\hat{\Psi}_{\mathbf{N_i}}\Psi^*
\end{aligned} \tag{43}$$

Then, I compare each terms of the expansions. The first terms are easy to compare by

Jensen inequality

$$\left(\frac{1}{n}\sum_{i=1}^{n}\hat{\Psi}^2_{\mathbf{N_i}}\right) \geq \left(\frac{1}{n}\sum_{i=1}^{n}\hat{\Psi}_{\mathbf{N_i}}\right)^2.$$

The term in the middle is bigger if the sample size is greater than one.

$$n(\Psi^*)^2 > (\Psi^*)^2 \quad \forall n > 1.$$

And the last terms in 43 are equal, so that concludes the proof.
Proposition 3 shows the behaviour of the LLMS estimator w.r.t the "consistency" of the underlying linear selection operator. That consistency is represented by a parameter $\alpha \in [0, 1]$, which can be understood as a Type I/II error. To do so, I need to assume that each element of the boolean linear selection operator $\hat{\Psi}_{\mathbf{N_i}}$ is a Bernoulli random variable.

$$\begin{aligned} &\boldsymbol{\pi} \equiv (\pi_1, ..., \pi_p) \in [0, 1]^p, \\ &\pi_j = \begin{cases} 1-\alpha & \text{if } j \in \mathcal{S}^* \\ \alpha & \text{if } j \notin \mathcal{S}^* \end{cases} \\ &\hat{\Psi}_{\mathbf{N_i}} \sim Bern(\boldsymbol{\pi}). \end{aligned} \tag{44}$$

Since the vector Ψ^* is only composed of ones and zeros, then it follows that the difference $(\hat{\Psi}_{\mathbf{N_i}} - \Psi^*)$ is also a vector of Bernoulli r.v. having all the same parameter equal to α.

$$(\hat{\Psi}_{\mathbf{N_i}} - \Psi^*) \sim Bern(\boldsymbol{\alpha}). \tag{45}$$

Also, because the linear selection vector is a boolean vector, taking its square does not change anything, it remains a boolean vector. Therefore its distribution is unchanged.

$$\hat{\Psi}_{\mathbf{N_i}} \in \{0, 1\}^p \Rightarrow (\hat{\Psi}_{\mathbf{N_i}} - \Psi^*)^2 \sim Bern(\alpha). \tag{46}$$

The first norm $\left\|\hat{\Psi}_{\mathbf{N_i}} - \Psi^*\right\|_2^2$ we are interested in, is the sum of the squared difference vector above. Because each element of the difference vector is a Bernoulli r.v., its sum, denoted y, is therefore a Binomial r.v. with parameter p, which is the length of the sum.

$$y \equiv \frac{1}{p}\sum_{j=1}^{p}(\hat{\Psi}_{\mathbf{N_i}} - \Psi^*)^2 \sim Binom(p, \alpha).$$

Now, moving on to the second norm $\|\mathcal{P} - \Psi^*\|_2^2$. By definition, the expected value of the

local linear selection is the vector of parameters $\boldsymbol{\pi}$ in equation 44. It happens to be exactly the definition of the LLMS estimator.

$$E[\hat{\Psi}_{\mathbf{N_i}}] = \boldsymbol{\pi}$$
$$\lim_{n\to\infty} \mathcal{P} \to \boldsymbol{\pi}$$

We can exploit this result to obtain the norm in terms of α and p only:

$$\begin{aligned}\lim_{n\to\infty} \|\mathcal{P} - \Psi^*\|_2^2 &= \|\boldsymbol{\pi} - \Psi^*\|_2^2 \\ &= \|\boldsymbol{\alpha}\|_2^2 \\ &= \sum_{j=1}^{p} \alpha^2 \\ &= \alpha^2 p.\end{aligned}$$

Knowing this, I can write the probability of the first norm, represented by the r.v. y, being greater than the second norm which is equal to $\alpha^2 p$. That probability is easy to obtain as y is binomial.

$$P\left(y \leq \alpha^2 p\right) = \sum_{l=0}^{\lfloor \alpha^2 p \rfloor} \binom{p}{l} \alpha^l (1-\alpha)^{p-l}$$

This probability tends to one as α decreases, and that concludes the proof. However it is worth noticing that when $\alpha = 0$ then both norms are equal to zero.

Proof of Proposition 4, follows directly from the previous proof. Simply by taking the limit of the same probability as the number of candidate variables p increases.

$$\lim_{p\to\infty} P\left(Y \leq \alpha^2 p\right) \to 0.$$

That limits tends to zero, which means that the local linear selection is always worse than LLMS when the number of variable is very large. The LLMS smooths by averaging, and so remains closer to the true set of selected variables.

3. Global Approach via Diagonal Auto-Encoders

Compared to the first section, here I use a completely different approach. Instead of a local estimator, Neural Networks provide a global estimator of the manifold.
There have been plenty of papers on neural networks that implemented variable selection. Many empirical works only uses some heuristics, while the theoretical part is significantly smaller. This literature is mainly driven by computer scientist, so here I also want to highlight the connection with more traditional statistics. Frequently, in regression models for instance, variable/feature is performed through constrained optimization. Parameters associated with a given variable are penalized such that some of them are exactly zeros, thus performing variable selection. These are known as "penalty methods", see Fan and Lv. (2010); Huang et al. (2012); Mehmood et al. (2012); Jović et al. (2015); Desboulets (2018) for complete reviews. Parameter penalization is quite easy to solve in the linear models. Whereas there is often no analytical formulas, simple algorithms can be implemented to get a solution. In non-linear models, the usual practice is to rewrite the problem using basis transformations (Wang and Xia, 2009; Ni et al., 2009) to reduce it back to a linear problem, solved with almost the same aforementioned algorithms. A Neural network can indeed be subject to these kinds of constraint since a network is a combination of multiple basis transformations. These bases are formed with non-linearities, often a sigmoid/tanh function, and parameters matrices. But only imposing sparsity on these matrices does not ensure variable selection, because a given variable is connected to multiple bases. So a sparse neural network is not necessarily a selection operator. One has to impose a sparsity condition on every connection of a variable to the network. Again, there exist different ways to implement this solution.
But before going into the final manifold selection operator, I will introduce a simple concept for sparse neural network, that ensures variable selection in the end.

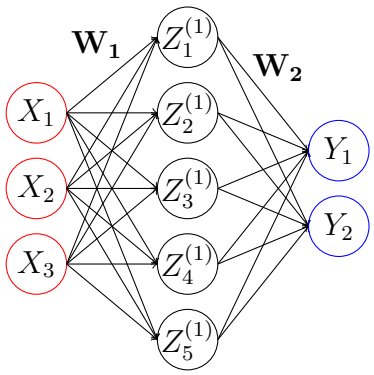
3.1. Restricted Diagonal Layer for Variable Selection

First, I introduce the mechanism I use for variable selection in the neural network framework. To do so, I consider the basic non-parametric model, that distinguishes between inputs and outputs. The objective is still to find the sets of "true" variables, but now there are two such sets: $\mathcal{S}_x^*$ and $\mathcal{S}_y^*$.

$$
\begin{aligned}
&\mathbf{y} = m(\mathbf{x}) \\
&\quad (X_1, X_2, ..., X_{p_x}) \equiv \mathbf{X} \in \mathbb{R}^{n \times p_x} \\
&\quad (Y_1, Y_2, ..., Y_{p_y}) \equiv \mathbf{Y} \in \mathbb{R}^{n \times p_y} \\
&\quad m : \mathbb{R}^{p_x} \to \mathbb{R}^{p_y} \\
\mathcal{S}_x^* :=& \ \{j : \exists j', \frac{\partial Y_{j'}}{\partial X_j} \neq 0\} \\
\mathcal{S}_y^* :=& \ \{j : \exists j', \frac{\partial Y_j}{\partial X_{j'}} \neq 0\} \\
|\mathcal{S}_x^*| \leq& \ p_x \\
|\mathcal{S}_y^*| \leq& \ p_y
\end{aligned}
\tag{47}
$$

The objective is to build a sparse estimator that only include the true sets of variables. And the mapping m is approximated by a neural network. For the choice of the structure of the neural network, I will stick to the proof of Hornik et al. (1990) and use a 3-layer architecture (as illustrated in Figure 6). To summarise, it is the minimal architecture required to approximate any non-linear function. From the viewpoint of these authors, there is no need to make the network deeper, i.e. having more layers.

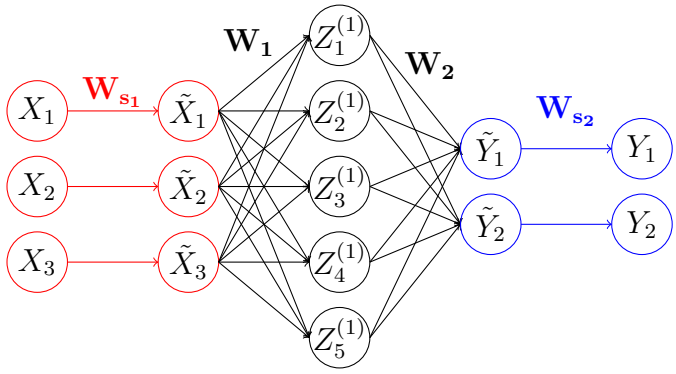
Fig. 6. Basic architecture for a Neural Network



For the sake of simplicity, I omit biases in the representation even though they are included at every layer, but said otherwise. Nevertheless, the simple 3-layer network does not provide a sparse solution. So how do we implement variable selection in neural networks? In the Machine Learning literature, selection often consists in quantifying the predictive importance of a given variable/feature (Breiman (2001)). This can be very computationally consuming. To make the network select variables, I use a different approach based on Neuneier and Zimmermann (1998) and Vecoven (2017). The main idea is to introduce

diagonal matrices that directly connect inputs and outputs to the network, controlling their entry in the model. W.r.t these papers, there are two contributions: (1) selection is performed also on the outputs, not only the inputs, (2) the domain of the diagonal selection parameters is constrained to be between 0 and 1. Figure 7, depicts the addition of the two selection layers at both ends of the network. On top of this, I do not need to enlarge the loss function to impose restrictions on parameters, contrary to Ye and Sun (2018) for instance. Only the structure of the network is modified. Restricting the selection parameters to the $[0, 1]$ domain has advantages in terms of interpretation: they can be seen as "pseudo-probabilities". If a variable seems to be useful to the network then its associated parameter in the input selection layer tends to 1, otherwise, it tends to 0. That directly links to the LLMS estimator presented in the previous part of the paper. Both give a continuous selection vector, whose elements are between 0 and 1.

Fig. 7. Modified architecture for selection in Neural Network



I restrict the domain of $\mathbf{W_{s_1}}$ and $\mathbf{W_{s_2}}$ using the non-linear sigmoid function.

$$\begin{aligned} sigmoid\colon \mathbb{R} &\to [0,1] \\ \boldsymbol{\gamma} &\mapsto \frac{1}{1+e^{-\boldsymbol{\gamma}}}. \end{aligned}$$

$$\begin{aligned} \mathbf{W_{s_1}} &= diag\left[sigmoid(\boldsymbol{\gamma_1})\right], \\ \mathbf{W_{s_2}} &= diag\left[sigmoid(\boldsymbol{\gamma_2})\right]. \end{aligned}$$

For the rest of the network, I use the hyperbolic tangent as the activation function, mainly for the reason that it is symmetric. The output layer is linear so that the network can map any real inputs to any real outputs, without further restrictions. For optimization stability one may still normalize the variables w.r.t their means and standard deviations. No activation function is applied to the selection layers. The full set of equations, in matrix

form, for the modified network structure is:

$$
\begin{aligned}
\mathbf{W_{s_1}} &= diag\left[sigmoid(\boldsymbol{\gamma_1})\right] \\
\tilde{\mathbf{X}} &= \mathbf{X}\mathbf{W_{s_1}} \\
\mathbf{Z^{(0)}} &\equiv \tilde{\mathbf{X}} \\
\mathbf{Z^{(l)}} &= tanh(\mathbf{Z^{(l-1)}}\mathbf{W_l}), \quad \forall l \in 1, \cdots, n_h - 1 \\
\hat{\mathbf{Y}} &= \mathbf{Z^{(n_h-1)}}\mathbf{W_{n_h}} \\
\tilde{\mathbf{Y}} &= \hat{\mathbf{Y}}\mathbf{W_{s_2}} \\
\mathbf{W_{s_2}} &= diag\left[sigmoid(\boldsymbol{\gamma_2})\right] \\
\mathbf{E} &= \mathbf{Y} - \tilde{\mathbf{Y}}
\end{aligned}
$$

Where n_h represents the depth of the network, the number of successive hidden layers. It is a hyper-parameter, that is chosen arbitrarily. Its value is in fact very linked to the neighbourhood size k in LLMS. Both control the degree of non-linearity, and ultimately depends on the shape of the DGP.
The optimization program is:

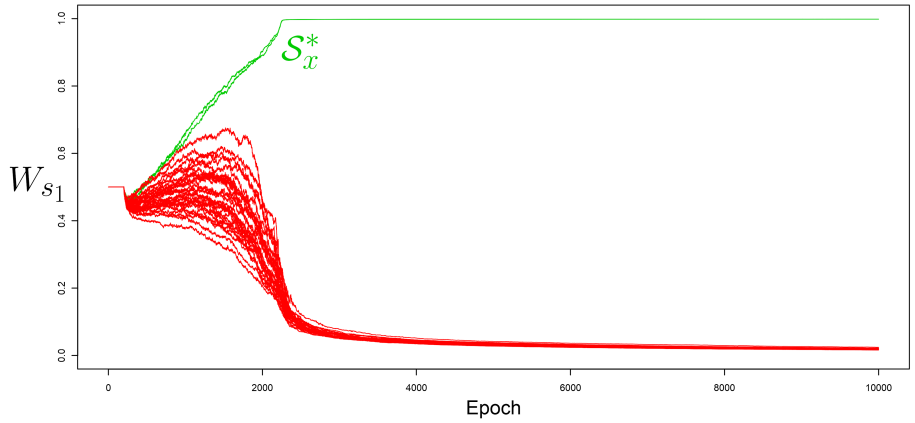
$$\min_{\mathbf{W_1},\ldots,\mathbf{W_{n_h}},\boldsymbol{\gamma_1},\boldsymbol{\gamma_2}} Trace(\mathbf{E}'\mathbf{E}).$$

The estimated selections are represented by selection weights $\mathbf{W_{s_1}}$ and $\mathbf{W_{s_2}}$ obtained at the end of the optimization:

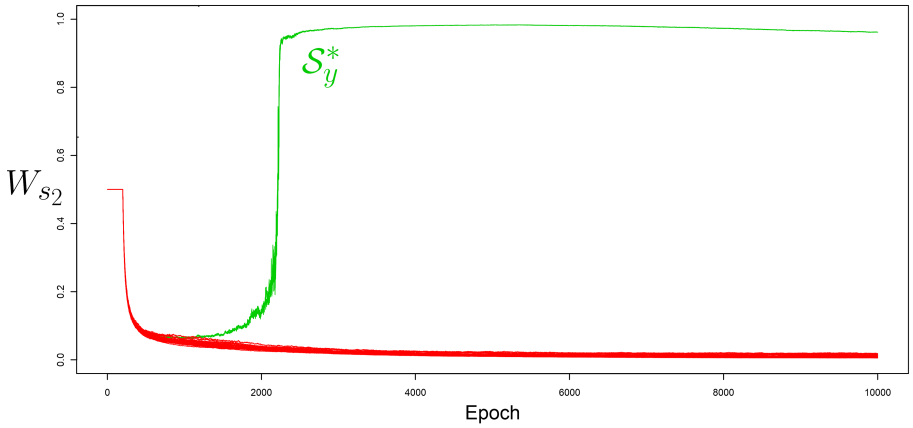
$$
\begin{aligned}
\hat{\mathcal{S}}_x &\to diag(\mathbf{W_{s_1}}), \\
\hat{\mathcal{S}}_y &\to diag(\mathbf{W_{s_2}}).
\end{aligned}
$$

To illustrate the validity of the concept, I run a simulation according to model 47. The selection paths are reported in Figure 8. Green paths correspond to the variables that belong to the true sets $\mathcal{S}_x^*$ and $\mathcal{S}_y^*$, the irrelevant ones are coloured in red. The green paths converge to 1 and the red ones to 0, so it is quite obvious from the chart which variables should be selected. Therefore, the selection seems consistent is this simulation. Nonetheless, I observe the paths for inputs and the paths for outputs do not converge in the same way. On the one hand, the inputs' parameters spread from epoch 1 to epoch 2000, and then are attracted to the boundaries. On the other hand, the outputs' parameters all go down first, and then diverge around epoch 2000.

Fig. 8. Selection paths for inputs and outputs selection



(a) Inputs Layer



(b) Output Layer

3.2. Diagonal Auto-encoders Manifold Selection

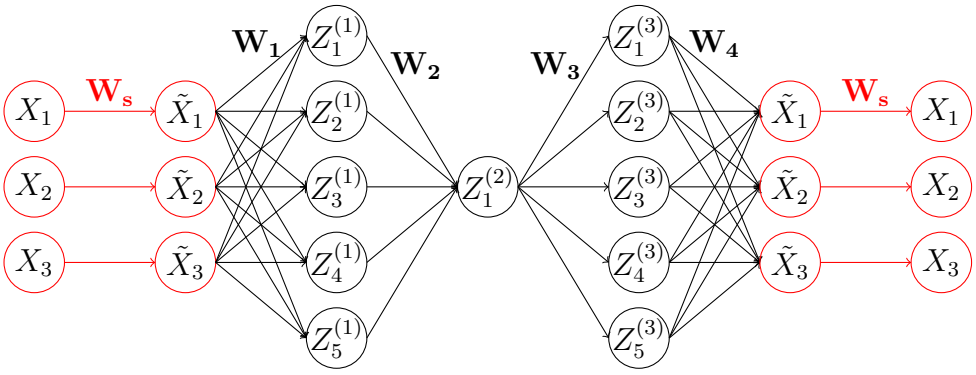
In this section, I extend the model from equation 47 to take into accounts non-linear manifolds. It is a generalisation, and it encompasses the previous section as a special case. Inputs $\mathbf{X}$ and outputs $\mathbf{Y}$ are embedded into a the larger "new inputs" $\mathbf{X}$ of size $p = p_x + p_y$. So the network has the same data at both ends. The model is therefore the same as for LLMS:

$$\begin{aligned}
\mathcal{M} :=& \{\mathbf{x} \in \mathbb{R}^p | \mathbf{x} = f \circ g(\mathbf{x})\} \\
& g : \mathbb{R}^p \to \mathbb{R}^r, f : \mathbb{R}^r \to \mathbb{R}^p, p > r \\
& (X_1, X_2, ..., X_p) \equiv \mathbf{X} \in \mathbb{R}^{n \times p} \\
\mathcal{S}^* :=& \{j : \exists j' \neq j, \frac{\partial X_{j'}}{\partial X_j} \neq 0\} \\
|\mathcal{S}^*| \leq& \ p
\end{aligned} \tag{48}$$

The projection function g maps the input onto a low-dimensional space, that is mapped back to the original space by the immersion function f. The problem turns into the estimation of two separate non-linear functions. Each function can be approximated using the 3-layers neural network. So to estimate both functions, the network is enlarged by combining two 3-layers architectures. At the center, they are connected by a "contraction" layer. This layer reduces the dimension of the input to the dimension of the manifold, as depicted in Figure 9. Its dimension has to be lower than the one of the input, otherwise, the network may learn the identity function. This architecture is not new. It is known as the 5-layers Auto-Encoder, proposed by Kramer (1991).

As I did before, I add the selection layers to this Auto-Encoder, thus obtaining a 7-layers architecture. Since the input flow through a deeper network, I will refer to this selection estimator as DAMS, for Diagonal Auto-Encoder Manifold Selection. Also, because the inputs and outputs are now the same, I can enforce the two selection parameters to be the same. Thanks to the chain rule, the derivative of this shared weight is fairly easy to compute. I only have to sum the two derivatives.

Fig. 9. Modified architecture for Diagonal Auto-Encoder Manifold Selection



The full set of equations, in matrix form, for the DAMS structure is almost the same.

$$
\begin{aligned}
\mathbf{W_s} &= diag\left[sigmoid(\boldsymbol{\gamma})\right] \\
\tilde{\mathbf{X}} &= \mathbf{X}\mathbf{W_s} \\
\mathbf{Z^{(0)}} &\equiv \tilde{\mathbf{X}} \\
\mathbf{Z^{(l)}} &= tanh(\mathbf{Z^{(l-1)}}\mathbf{W_l}), \quad \forall l \in 1, \cdots, n_h - 1 \\
\hat{\mathbf{X}} &= \mathbf{Z^{(n_h-1)}}\mathbf{W_{n_h}} \\
\tilde{\mathbf{X}} &= \hat{\mathbf{X}}\mathbf{W_s} \\
\mathbf{E} &= \mathbf{X} - \tilde{\mathbf{X}}
\end{aligned}
$$

The optimization program is:

$$\min_{\mathbf{W_1},\ldots,\mathbf{W_{n_h}},\boldsymbol{\gamma}} Trace(\mathbf{E}'\mathbf{E}).$$

And the estimated set is simply:

$$\hat{\mathcal{S}} \rightarrow diag(\mathbf{W_s})$$

The DAMS estimator $diag(\mathbf{W_s})$ is an analogue to the LLMS estimator $\mathcal{P}$. Their main difference lies in the differentiability and continuity assumptions of the manifold. DAMS requires both f and g to be continuous everywhere, while LLMS requires continuity only locally, inside the neighbourhoods. Also, for LLMS, the two functions have to be differentiable, at least locally. Differentiability is not needed for DAMS. If the true manifold is not continuous everywhere, DAMS might be inconsistent. The selection parameters depend on the derivatives of the total error. They are not linked to the derivatives of f or g, contrary to LLMS.

Another thing is that the LLMS estimator is dependent on a penalization parameter θ. That is not the case for DAMS, which incorporates that mechanism inside the optimisation

process. The nice thing is that I can still provide the so-called "selection paths" for DAMS exactly as for LLMS. It suffices to plot the evolution of the parameters $diag(\mathbf{W_s})$ throughout the optimisation process. Obviously, the optimal value for $\mathbf{W_s}$ is chosen to be the last of the optimisation process.

3.3. Estimation of the Parameters of the Network

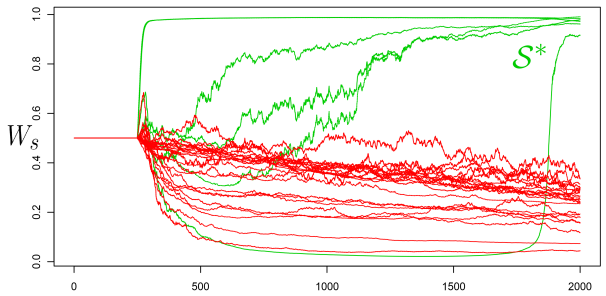
3.3.1. Stochastic First-Order Optimization

The architecture proposed only enlarge the network by 2 linear layers. I do not modify the loss function. Thanks to the chain rule, it makes the computations of the derivatives for each weight as simple as any other layers. Standard backpropagation algorithm can be used without further modifications. Thus, I simply use the gradient descent algorithm, computed on a random batch of the data. This method is known as stochastic gradient descent. It has two main advantages: (1) it avoids local minimas more easily and (2) it reduces the duration of the computations. The size of the batches is a hyper-parameter, usually set to a fraction of the total number of observations.

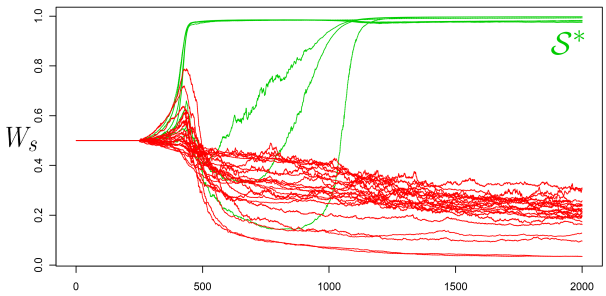
3.3.2. Smoothing selection paths with Ensemble Averaging

There are two reasons why the paths of the parameters might be unstable. First, optimisation of both selection weights and the rest of the network parameters simultaneously. This means that the network has to learn the manifold at the same time it does the selection. A better idea would be to start learning the manifold with all the variables, even though it means a less precise estimator. After that, start the selection. The estimator will then be more precise. So $\mathbf{W_1}, ..., \mathbf{W_{n_h}}$ are being optimised few epochs before $\mathbf{W_s}$. Second, the parameters estimated for a given network depend on weights initialization. A single set of selection paths W_s might not be stable, nor lead to consistent selection. This phenomenon is depicted in Figure 10. The green paths correspond to the variables that are in $\mathcal{S}^*$, while the red correspond to the noisy variables. Indeed, the paths are sometimes chaotic. The solution is to average the paths across many random initialisations. This method is known as "Ensemble Averaging". The network is replicated a certain number of times, to form what is called an "Ensemble". The paths for every replication of the network are then averaged. That produces more stable, but also more consistent selection paths, as it is shown in Figure 11

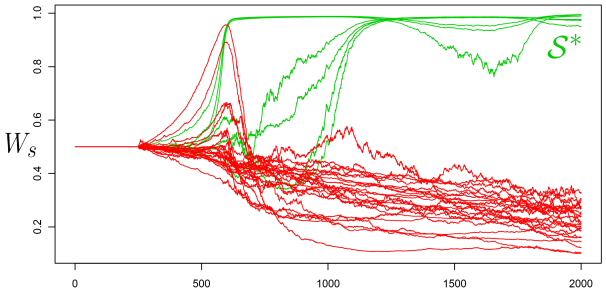
Fig. 10. Examples of DAMS selection paths with a single network



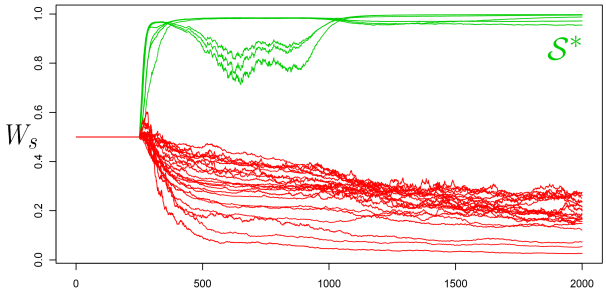
(a) Selection paths for Network n° 1



(b) Selection paths for Network n° 2

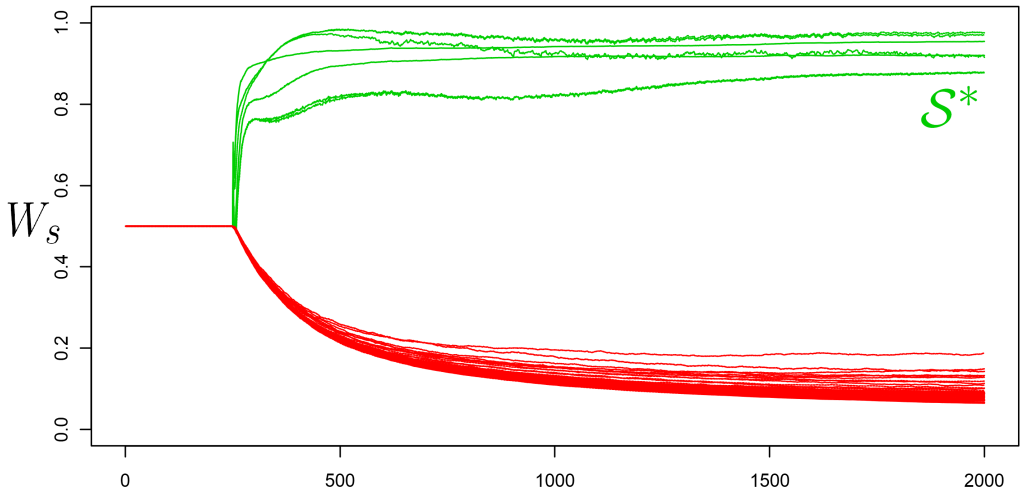


(c) Selection paths for Network n° 3



(d) Selection paths for Network n° 4

Fig. 11. Smooth selection paths for an ensemble average of 10 networks



4. Simulations

4.1. Design

To simulate a manifold, I choose to first simulate r-dimensional coordinates, then project them linearly into $d = |\mathcal{S}^*|$ dimensions, and finally fold them with non-linear functions. This allows to control r and the non-linearities of the manifold. The amount of non-linearity is controlled by linear coefficients $\mathbf{A}$. For each simulated process, $\mathbf{A}$ is randomly chosen, so that the shape of the manifold changes as well. Very small changes in $\mathbf{A}$ might induces very different manifolds in the end. I simulate $X_j = f_{j'}(Z_j A_j)$ with $Z_j \sim U[-2, 2]$, $j' \neq j$, $\mathbf{A} \sim U[-2, 2]$. The non-linear transformations are chosen among the following ones:

- $f_1(\mathbf{x}) = \mathbf{x}$,
- $f_2(\mathbf{x}) = \mathbf{x}^2$,
- $f_3(\mathbf{x}) = sin\left(\frac{\mathbf{x}}{2}\right)$,
- $f_4(\mathbf{x}) = cos\left(\frac{\mathbf{x}}{2}\right)$,
- $f_5(\mathbf{x}) = tanh\,(2\mathbf{x})$,
- $f_6(\mathbf{x}) = exp\left(\frac{(\mathbf{x}-10)^2}{20}\right)$,
- $f_7(\mathbf{x}) = (x+1)\mathbb{1}_{x>\bar{x}} - (x-3)\mathbb{1}_{x\leq\bar{x}}$.

I also set the following parameters:

- $n = 5000$,
- $p = 50$,
- $d = 7$,
- $\upsilon = \{0.01, 0.25\}$

There are two hyper-parameters left. The size of the neighbourhood k for LLMS, and the size of the hidden layers for DAMS. Both control the degree of non-linearities. I choose to perform both algorithms with 3 different values of these parameters. For LLMS, $k = \{5\%, 20\%, 50\%\}$ of the sample size n. And for DAMS, the hidden layer sizes are $\{(7, 1, 7), (7, 3, 7), (7, 5, 7)\}$. The simulation is run 1000 times and results are averaged. The

computations are performed on an Intel(R) Core(TM) i5-3220M CPU 2 × 2.60GHz with 12GB RAM. The estimated selections are compared with existing methods, both parametric and non-parametric. I only compare to selection methods which do not make difference between explanatory variables and outcome variables. These are methods which act on covariance/correlation matrices, for instance, the LASSO is not valid for comparison in these simulations but the Graphical LASSO is.

- Eigen Thresholding (ET)

$$\begin{aligned}\boldsymbol{\Sigma} &= \mathbf{VLV}' \\ p\left(\boldsymbol{\Sigma}, \theta\right) &= V_{ij} \mathbb{1}_{|V_{ij} L_j^{1/2}| > \theta}\end{aligned}$$

- Sparse PCA (sPCA) Zou et al. (2006)

$$\begin{aligned}\boldsymbol{\Sigma} &= (\mathbf{XVV}')'(\mathbf{XVV}') \\ p\left(\boldsymbol{\Sigma}, \theta\right) &= \theta_1 \left\|\mathbf{V}\right\|_1 + \theta_2 \left\|\mathbf{V}\right\|_2^2\end{aligned}$$

- Graphical LASSO (gLASSO) Friedman et al. (2008)

$$\begin{aligned}\boldsymbol{\Sigma} &= Cov(\mathbf{X}) \\ p\left(\boldsymbol{\Sigma}, \theta\right) &= log(det(\boldsymbol{\Sigma}^{-1})) - tr(\boldsymbol{\Sigma}^{-1}) - \theta \left\|\boldsymbol{\Sigma}^{-1}\right\|_1\end{aligned}$$

- Penalized Matrix Decomposition (PMD) Witten et al. (2009)

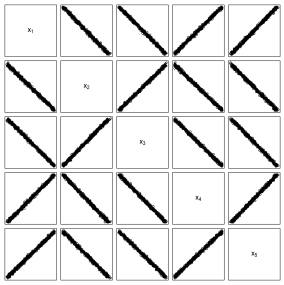
$$\begin{aligned}\boldsymbol{\Sigma} &= (\mathbf{dUV}')'(\mathbf{dUV}') \\ p\left(\boldsymbol{\Sigma}, \theta\right) &= \theta_1 \left\|\mathbf{U}\right\|_1 + \theta_2 \left\|\mathbf{V}\right\|_1\end{aligned}$$

- NonParanormal (NPN) Lafferty et al. (2012)

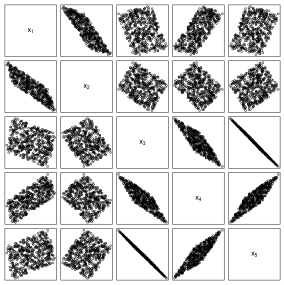
$$\begin{aligned}\mathbf{f_X} &= (f_1(X_1), ..., f_p(X_p)) \\ f_j &= \mathbb{E}(X_j) + Var(X_j)\Phi^{-1}\left(\hat{F}_j(X_j)\right) \quad \forall j \in \{1, ..., p\} \\ \boldsymbol{\Sigma} &= Cov(\mathbf{f_X}) \\ p\left(\boldsymbol{\Sigma}, \theta\right) &= log(det(\boldsymbol{\Sigma}^{-1})) - tr(\boldsymbol{\Sigma}^{-1}) - \theta \left\|\boldsymbol{\Sigma}^{-1}\right\|_1\end{aligned}$$

  where Φ^{-1} is the Inverse Normal CDF and $\hat{F}$ is the empirical CDF.

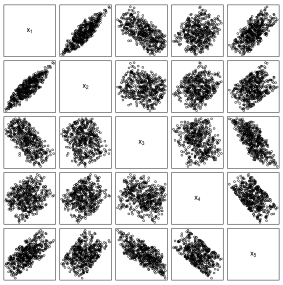
Fig. 12. Simulated DGP with low noise ($\upsilon = 0.01$)



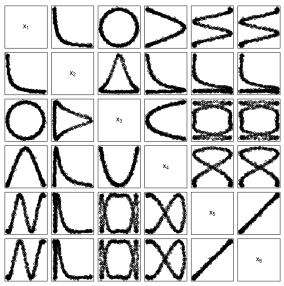
(a) Linear $r = 1$



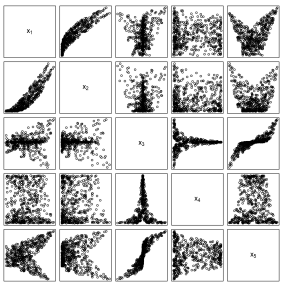
(b) Linear $r = 2$



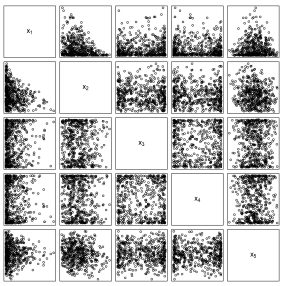
(c) Linear $r = 3$



(d) Non-linear $r = 1$

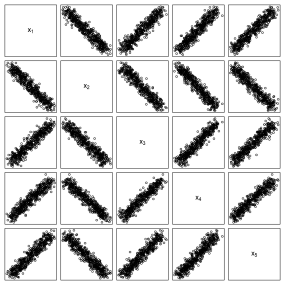


(e) Non-linear $r = 2$

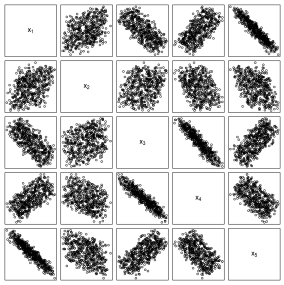


(f) Non-linear $r = 3$

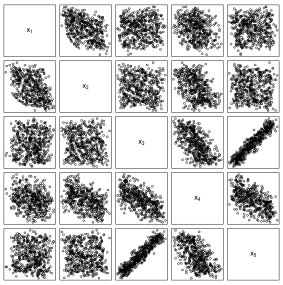
Fig. 13. Simulated DGP with high noise ($\upsilon = 0.25$)



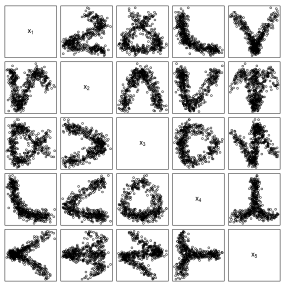
(a) Linear $r = 1$



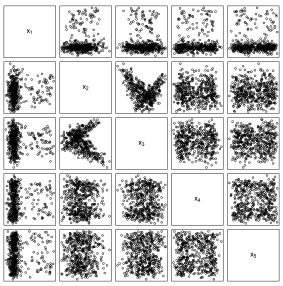
(b) Linear $r = 2$



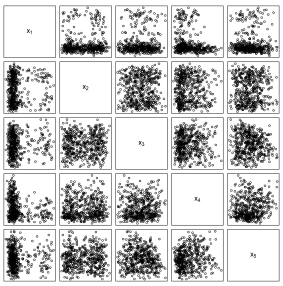
(c) Linear $r = 3$



(d) Non-linear $r = 1$



(e) Non-linear $r = 2$



(f) Non-linear $r = 3$

4.2. Results

In this section, I report the results of the simulations. First, I report average selection paths for DAMS and LLMS in Figure 15 and 14. In both cases, the parameters converge to the true set of variables $\mathcal{S}^*$. The green paths correspond to the variables that are in $\mathcal{S}^*$, while the red correspond to the noisy variables. The paths are smooth, and clearly show which variable should be selected.

Fig. 14. Selection paths - LLMS

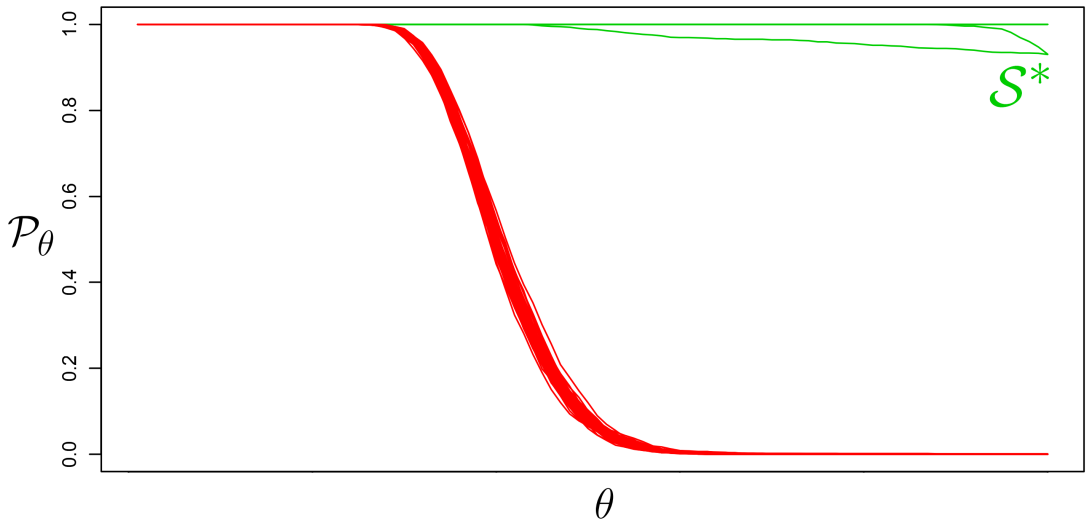
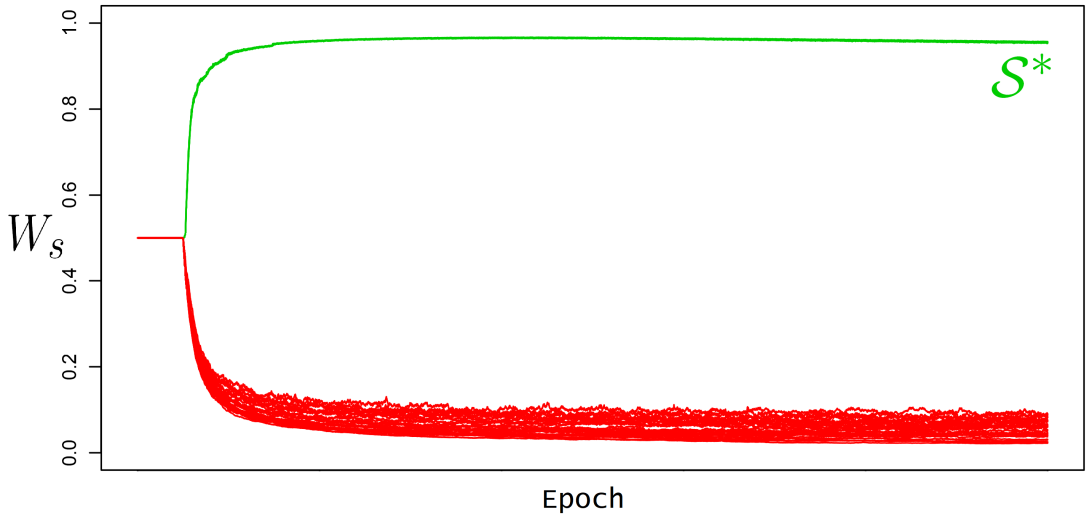


Fig. 15. Selection paths - DAMS



Then, I compare different versions of LLMS and several other algorithms in terms of True Positive Rate (TPR) and False Positive Rate (FPR). The TPR measures the proportion of

selected variables that are truly in the set $\mathcal{S}^*$. The FPR is the opposite, i.e. the proportion of selected variables which are not in $\mathcal{S}$.

$$\text{TPR} = \frac{|\hat{S} \cap \mathcal{S}^*|}{|\mathcal{S}^*|}$$
$$\text{FPR} = \frac{|\hat{S} - \mathcal{S}^*|}{p - |\mathcal{S}^*|}$$

Table 1: Selection rates - Simulation with low noise ($\upsilon = 0.01$)

	Linear		**Non-Linear**						
	$r = \{1,2,3\}$		$r = 1$		$r = 2$		$r = 3$		
	TPR	FPR	TPR	FPR	TPR	FPR	TPR	FPR	Runtime (sec.)
$\text{LLMS1}_{k=5\%}$	100.	-	99.9	-	88.9	-	85.7	-	61.200
$\text{LLMS1}_{k=20\%}$	100.	-	98.9	-	83.5	-	71.4	-	97.300
$\text{LLMS1}_{k=50\%}$	100.	-	96.4	-	59.4	-	51.9	-	291.800
$\text{LLMS2}_{k=5\%}$	100.	-	100.	-	100.	-	100.	-	50.800
$\text{LLMS2}_{k=20\%}$	100.	-	100.	-	98.6	-	98.7	-	60.800
$\text{LLMS2}_{k=50\%}$	100.	-	96.5	-	70.6	-	71.1	-	73.400
$\text{LLMS3}_{k=5\%}$	100.	-	100.	-	90.6	-	86.3	-	179.300
$\text{LLMS3}_{k=20\%}$	100.	-	100.	-	85.2	-	73.4	-	306.500
$\text{LLMS3}_{k=50\%}$	100.	-	96.8	-	62.1	-	50.3	-	1775.300
$\text{DAMS}_{(7,1,7)}$	100.	-	100.	-	100.	-	85.7	-	342.700
$\text{DAMS}_{(7,3,7)}$	100.	-	100.	-	100.	-	100.	- 0.1	354.800
$\text{DAMS}_{(7,5,7)}$	100.	-	100.	-	100.	- 4.6	100.	- 9.8	391.200
ET	100.	-	86.6	-	37.1	-	17.6	-	0.014
sPCA	100.	-	23.2	-	30.3	-	25.8	-	0.625
gLASSO	100.	-	89.3	-	46.7	-	18.7	-	0.036
PMD	100.	- 1.9	89.6	- 7.2	87.5	- 25.6	70.8	- 40.1	2.279
NPN	100.	-	90.3	- 5.7	70.9	- 28.9	71.4	- 76.6	1.058

All values equal to zero are left as blanks.

Results for low variance of noise are reported in Table 1. Each column corresponds to a different DGP. There are six possibles DGP, three linear, three non-linear. Each has a different reduced dimension of r. When that dimension is equal to 1, visual inspection is enough to assess the correlation, as you can see in the graph d) from Figure 12. The correlations become less obvious as r increases. Therefore, the value of r may have an impaction on the selection properties. So in the end, there are four columns in the tables. The first one aggregates the three linear models since the results I obtain do not vary at all with r. Then, each line of the table corresponds to a different algorithm. The three first blocks are for the three versions of LLMS. So there are 9 lines corresponding to LLMS, since there are

three values for k. The next block is for DAMS, there are also 3 lines. The last block is for all other algorithms.

In the first column, you observe TPR (left) and FPR (right) for linear models. Without any surprises, all algorithms have 100% performance. They all select the true set $\mathcal{S}^*$ and have also zero FPR.
Next, in the second column, there are results for the non-linear manifold of dimension $r = 1$. Overall, LLMS performs very well. I observe that LLMS3 is very slightly superior to LLMS1. This result shows that correcting distances with $q - kNN$ improves selection. Another finding is the choice of k does not seem to greatly affect the TPR. That finding is also observed for DAMS, whatever the hidden layer sizes it always selects the true set. Surprisingly, the other algorithms also have decent performances on the non-linear DGP. The linear selection operators (ET, gLASSO and PMD) have a near 90% TPR. Only PMD has a non-zero FPR, but it is quite low (7.2%). The only other non-linear operator which is NPN, has 90.3% TPR and 5.7% FPR. I can conclude that in case of very low dimensional non-linear manifold ($r = 1$), the linear operators have quite good selection properties.
Finally, in the last columns ($r = 2$ and $r = 3$) the performances change radically. The sensitivity to k in LLMS is much greater, whereas DAMS is robust to different layer sizes. This result highlights the difference between a local and a global approach. If k is at 50%, all versions of LLMS fail to select the true set. Also, LLMS1 and LLMS3 have much poorer performance than LLMS2, respectively 85.7% and 86.3% versus 100%. Among the proposed methods, only DAMS with the largest hidden layer $(7, 5, 7)$ has non-zero FPR. This is due to the fact that, in these cases, the contraction layer has a dimension greater than r. Therefore, there are too many dimensions than what is needed to approximate the manifold. These extra dimensions are used by the network and learn the identity function for some noise variables. Symmetrically, the smallest structure $(7, 1, 7)$ has only 85.7% TPR. There are not enough dimensions to approximate the manifold, and it biases the selection afterwards. So, just as k, there is a trade-off for choosing a good value of the hidden layer sizes. Nevertheless, the performance is better than the other algorithms. The linear operators exhibit two patterns. The first one is characterized by a low TPR (around 20%) and a zero FPR. The second pattern is characterized by a relatively high FPR (around 70%) but with also a high FPR (ranging from 40% to 80%). So in the first case, they fail to select all the relevant variables, only some of them. In the second case, they simply select too many variables, including, by chance, some of the relevant ones.

The runtimes show that both LLMS and DAMS are much longer to perform than any competitors. Among the three versions of LLMS, LLMS3 is the longest. The best version in terms of TPR, is LLMS2. It takes around a minute, which is reasonable. The computations depend on the neighbourhood size k, so the runtime can be reduced by reducing k but that may affect selection. DAMS takes, on average, 6 minutes. The computations heavily depend on the batch size and the number of networks in the ensemble. Reducing these two parameters can shorten the runtime but may affect selection.

Table 2: Selection rates - Simulation with high noise ($v = 0.25$)

	Linear		**Non-Linear**						
	$r = \{1, 2, 3\}$		$r = 1$		$r = 2$		$r = 3$		
	TPR	FPR	TPR	FPR	TPR	FPR	TPR	FPR	Runtime (sec.)
$\text{LLMS1}_{k=5\%}$	99.1	-	94.3	-	90.4	-	76.4	-	59.600
$\text{LLMS1}_{k=20\%}$	99.3	-	88.	-	73.7	-	57.1	-	104.100
$\text{LLMS1}_{k=50\%}$	97.7	-	83.4	-	55.8	-	41.9	-	313.300
$\text{LLMS2}_{k=5\%}$	99.7	-	99.7	-	99.8	-	99.6	-	51.000
$\text{LLMS2}_{k=20\%}$	99.7	-	97.3	-	98.1	-	95.7	-	60.000
$\text{LLMS2}_{k=50\%}$	99.1	-	85.4	-	68.5	-	50.6	-	69.100
$\text{LLMS3}_{k=5\%}$	99.4	-	97.1	-	100.	-	80.7	-	147.400
$\text{LLMS3}_{k=20\%}$	99.3	-	90.9	-	75.6	-	58.4	-	300.400
$\text{LLMS3}_{k=50\%}$	98.0	-	84.9	-	59.4	-	41.2	-	1825.300
$\text{DAMS}_{(7,1,7)}$	95.7	-	94.4	-	92.3	-	74.7	-	334.700
$\text{DAMS}_{(7,3,7)}$	100.	-	100.	-	100.	-	97.1	-	355.300
$\text{DAMS}_{(7,5,7)}$	100.	-	100.	-	100.	-	100.	-	371.200
ET	93.7	-	76.6	-	44.3	-	27.5	-	0.016
sPCA	97.6	-	40.6	-	32.3	-	27.1	-	0.635
gLASSO	97.4	-	80.3	-	45.1	-	24.6	-	0.030
PMD	96.7	- 5.8	86.9	- 18.6	85.9	- 28.5	82.9	- 35.5	2.170
NPN	99.9	- 0.1	82.6	-	71.4	- 35.7	72.9	- 73.2	1.093

All values equal to zero are left as blanks.

Results for high variance of noise are reported in Table 2. The structure of the table is the same as the preceding. The only difference w.r.t to Table 1 is the level of the noise. This noise may affect the selection, as it is more difficult to approximate the manifold.

In the first column, for the linear DGPs, I observe almost the same performance for all algorithms, above 95%. DAMS is the only algorithm to have 100% TPR. In columns 2, 3 and 4, the overall performance is poorer than in the low noise case. The relative performance

between each algorithm and the others stays the same, compared to the previous table. Only LLMS2 and DAMS are consistent estimators for all values of r. Still, LLMS2 is sensitive to k. A value between 5% and 20% produces high TPR, ranging from 95.7% to 99.8%. For DAMS, the $(7, 5, 7)$ structure always produces perfect selection.

The results from these simulations are the following. LLMS2 and DAMS are good candidates as selection estimator for data on non-linear manifolds. The basic implementation (LLMS1) is not consistent enough, and it is very sensitive to k. The last version LLMS3 is indeed better than LLMS1, but not as much as LLMS2 is. A combination of LLMS2 and LLMS3 (r-kNN + q-kNN) would certainly produce even better results. But it would need even more computations. And when looking at runtimes of LLMS3, such a combination would be computationally prohibitive.
A linear selection estimator fails to select the right set of variables when the true DGP is non-linear.

4.3. Numerical Oracle Properties

The Oracle Property is the property for a selection operators to uncover the set of true variables $\mathcal{S}^*$. This property has already been discussed for LLMS in section 2.5. Because the two proposed estimators take on continuous values, I have defined that property in terms of a distance w.r.t to the true set, in boolean form, denoted Ψ^*. Therefore, the Weak Continuous Oracle property for LLMS and DAMS is stated as:

$$\lim_{n\to\infty} P\left(\|\mathcal{P}_{\theta^*} - \Psi^*\|_2^2 \leq \nu\right),$$
$$\lim_{n\to\infty} P\left(\|diag(\mathbf{W_s}) - \Psi^*\|_2^2 \leq \nu\right).$$

with ν arbitrarily small. So in this section, I investigate on that property numerically for the two algorithms. For increasing values of n, I simulate a non-linear DGP 1000 times, as I did in the previous section. Then, I compute the frequency at which the observed norm is lower than an arbitrarily small value ν. If the estimator is consistent, as the sample size increases I should observe the norm decreasing as well. In the upper part of the figure, I report the norm along with sample size. It takes on four values $n = \{500, 5'000, 50'000, 500'000\}$. The solid black line represents the median of the different runs for a given sample size. The lower part of the figure represents the frequency at which the norm is lower than a given small value ν. These small values range from 0.01 to 0.0001. There is a coloured line for

each value of ν.

The results of the simulations for LLMS2 are presented in Figure 16. In the upper part, the norm decreases with sample size. In the lower part, the observed frequency tends to 1 as n increases for any value of ν. This finding supports the hypothesis that LLMS is indeed an Oracle estimator. I also note that the convergence is pretty fast. Most of the coloured lines are already closed to 1 at sample size $5'000$.
The results for DAMS are presented in Figure 17. In the upper part, the norm decreases with sample size as well. However, for a small value of n, there are some outliers. The DAMS operator is therefore not stable in selection, in small samples, in these simulations. As n increases, the observed frequency tends also to 1 for any value of ν. Again, this result supports the hypothesis that DAMS is indeed an Oracle estimator. Nevertheless, the convergence is slower than LLMS.

Fig. 16. Continuous Oracle property - LLMS

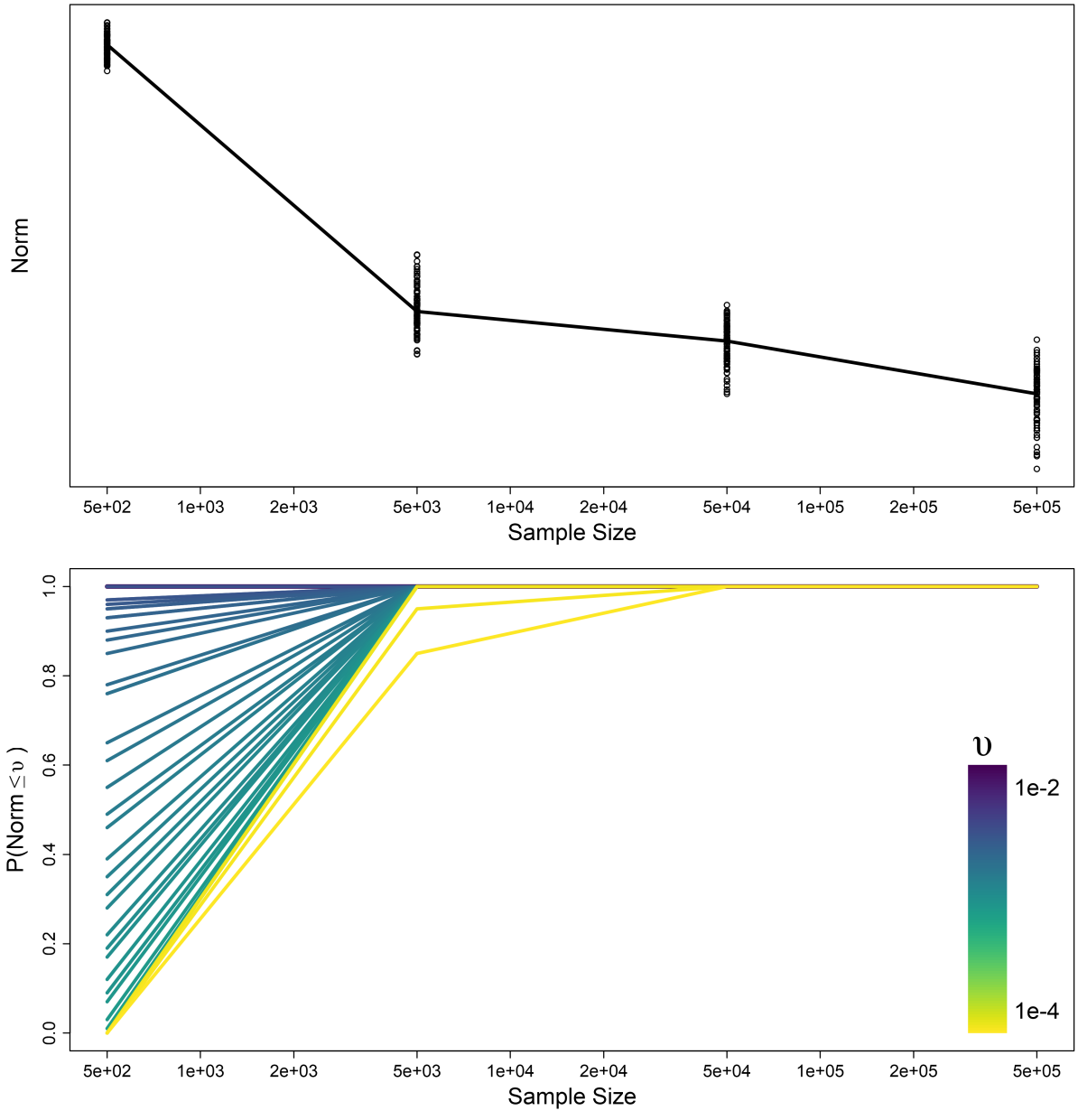
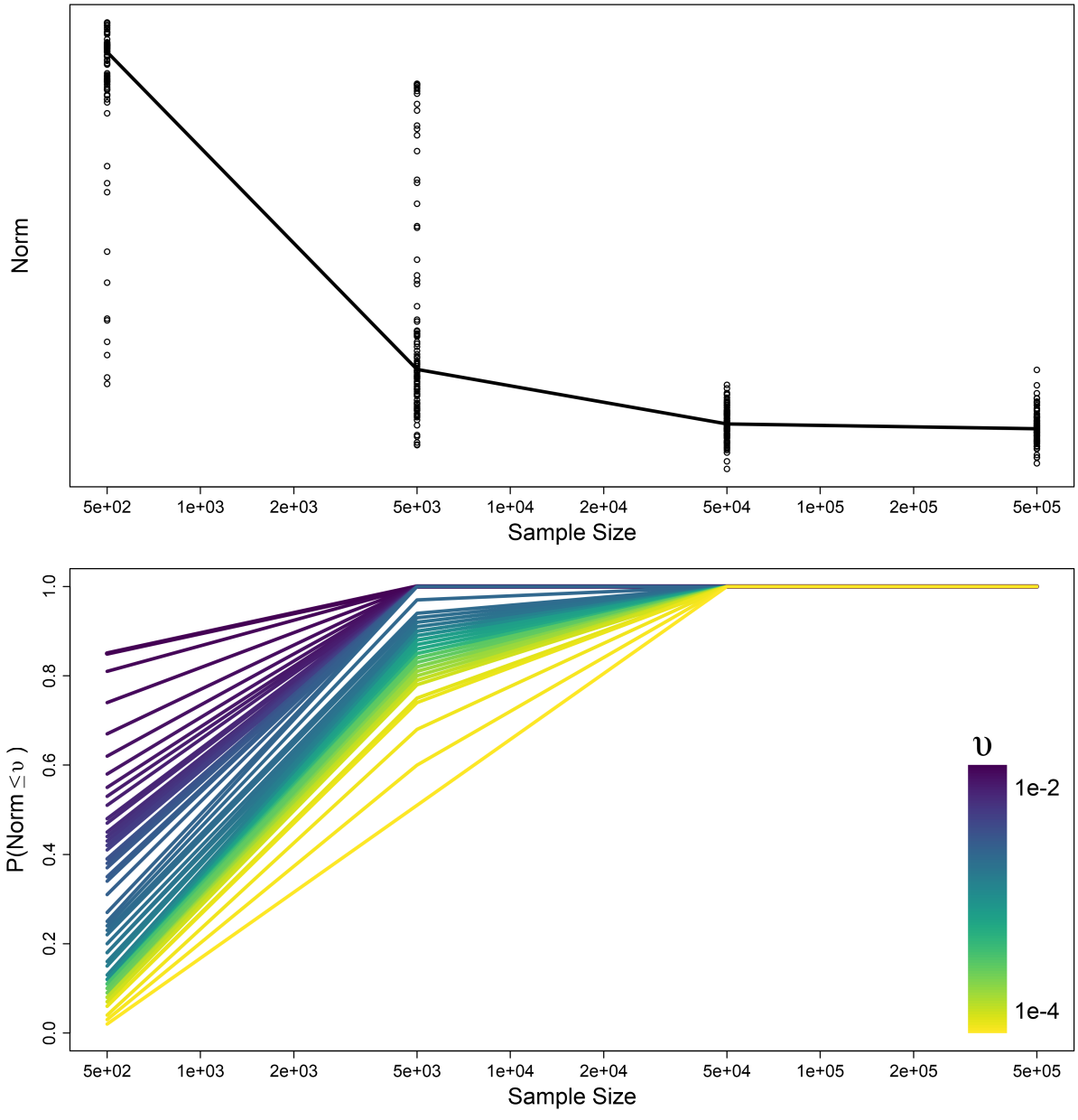


Fig. 17. Continuous Oracle property - DAMS



5. Discussion

In summary, I have proposed two statistical procedures to perform non-parametric variable selection. The first method is the Local Linear Manifold Selection, denoted LLMS. It relies on multiple linear selection estimators. The sample is divided into subregions called neighbourhoods, which have the property to be almost linear. In each of these subregions, a linear selection operator is applied and the result of the selections are averaged. These neighbourhoods are constructed using three different versions of the k-nearest neighbours (kNN) algorithm. The vanilla version, a second one to handle very high dimensional inputs (r-kNN), and a third one to handle very high non-linearities (q-kNN). The results of the simulations indicate very good performance of LLMS, especially the second version. In the second part, I propose a method called DAMS, for Diagonal Auto-encoder Manifold Selection. I first introduce the "diagonal layer", to perform input/output selection in the simple non-parametric model. The diagonal layer network is an enlarged network, with two diagonal matrices of selection parameters, restricted on $[0, 1]$. This layer controls the entry and the exit of the data in the network. Therefore, it acts as a selection operator. Then, I show that Auto-encoder Networks can be used for variable selection on non-linear manifolds. I use the trick known as "Ensemble Averaging", a common method in Machine Learning, to obtain smooth selection paths.
The main findings of the paper are the investigation of their selection properties. For LLMS, selection consistency is theoretically established. As long as the underlying linear selection operator is consistent, LLMS is also consistent. Through simulations, I show that the two methods, LLMS and DAMS, lead to consistent selection. LLMS and DAMS are both consistent as the sample size increases.

Still, there is room for improvement, especially for hyper-parameter selection. For LLMS, I propose an optimal rule for the penalty parameter θ, but not for the neighbourhoods' size k. For instance, the latter could be optimized w.r.t to the manifold reconstruction error. For DAMS, the only hyper-parameters of interest are the layer size, especially the central contraction layer. If it is larger than r, there can be over-identification of f and g. But it is not obvious that selection parameter will be affected. If the contraction layer size is larger or equal to $|\mathcal{S}^*|$, the network can start to learn the identity function, and select more variables. Unfortunately, both r and $|\mathcal{S}_x^*|$ are unknown beforehand. These are issues for future research to explore.

References

Breiman, L. (2001). Random forests. *Machine Learning*, 45:5–32.

Desboulets, L. (2018). A review on variable selection in regression analysis. *Econometrics*, 6(4):45.

Ding-Cheng, F., Feng, C., and Wen-Li, X. (2014). Detecting local manifold structure for unsupervised feature selection. *Acta Automatica Sinica*, 40(10):2253–2261.

Doquet, G. and Sebag, M. (2020). Agnostic feature selection. In *Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2019, Würzburg, Germany, September 16–20, 2019, Proceedings, Part I*, pages 343–358. Springer.

Fan, J. and Li., R. (2001). Variable selection via nonconcave penalized likelihood and its oracle properties. *Journal of the American Statistical Association*, 96:1348–60.

Fan, J. and Lv., J. (2010). A selective overview of variable selection in high dimensional feature space. *Statistica Sinica*, 20:101.

Friedman, J., Hastie, T., and Tibshirani, R. (2008). Sparse inverse covariance estimation with the graphical lasso. *Biostatistics*, 9(3):432–441.

Gorban, A. and Zinovyev, A. (2011). Principal graphs and manifolds. pages 28–59.

Hastie, T. and Stuetzle, W. (1989). Principal curves. *Journal of the American Statistical Association*, 84(406):502–516.

Ho, T. K. (1998). Nearest neighbors in random subspaces. In *Joint IAPR International Workshops on Statistical Techniques in Pattern Recognition (SPR) and Structural and Syntactic Pattern Recognition (SSPR)*, pages 640–648. Springer.

Hornik, K., Stinchcombe, M., and White, H. (1990). Universal approximation of an unknown mapping and its derivatives using multilayer feedforward networks. *Neural networks*, 3(5):551–560.

Huang, J., Breheny, P., and Ma., S. (2012). A selective review of group selection in high-dimensional models. *Statistical Science*, 27.

Jolliffe, I. T. (1982). A note on the use of principal components in regression. *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, 31(3):300–303.

Jović, A., Brkić, K., and Bogunović, N. (2015). A review of feature selection methods with applications. *Paper presented at 38th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO), Opatija, Croatia*, 2015:25–29.

Kohonen, T. (1998). The self-organizing map. *Neurocomputing*, 21(1-3):1–6.

Kramer, M. (1991). Nonlinear principal component analysis using autoassociative neural networks. *AIChE Journal*, 37(2):233–243.

Lafferty, J., Liu, H., and Wasserman, L. (2012). Sparse nonparametric graphical models. *Statistical Science*, 27(4):519–537.

Li, Y., Shi, X., Du, C., Liu, Y., and Wen, Y. (2016). Manifold regularized multi-view feature selection for social image annotation. *Neurocomputing*, 204:135–141.

Mehmood, T., Liland, K. H., Snipen, L., and Sæbø, S. (2012). A review of variable selection methods in partial least squares regression. *Chemometrics and Intelligent Laboratory Systems*, 118:62–69.

Meinshausen, N. and Bühlmann, P. (2010). Stability selection. *Journal of the Royal Statistical Society: Series B*, 72:417–73.

Mika, S., Schölkopf, B., Smola, A., Müller, K.-R., Scholz, M., and Rätsch, G. (1999). Kernel pca and de-noising in feature spaces. pages 536–542.

Neuneier, R. and Zimmermann, H. G. (1998). How to train neural networks. pages 373–423.

Ni, X., Zhang, H. H., and Zhang, D. (2009). Automatic model selection for partially linear models. *Journal of Multivariate Analysis*, 100:2100–11.

Roweis, S. and Saul, L. (2000). Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290(5500):2323–2326.

Saul, L. and Roweis, S. (2003). Think globally, fit locally: unsupervised learning of low dimensional manifolds. *Journal of Machine Learning Research*, 4(Jun):119–155.

Schölkopf, B., Smola, A., and Müller, K.-R. (1998). Nonlinear component analysis as a kernel eigenvalue problem. *Neural Computation*, 10(5):1299–1319.

Sun, S. and Huang, R. (2010). An adaptive k-nearest neighbor algorithm. In *2010 seventh international conference on fuzzy systems and knowledge discovery*, volume 1, pages 91–94. IEEE.

Tang, C., Bian, M., Liu, X., Li, M., Zhou, H., Wang, P., and Yin, H. (2019). Unsupervised feature selection via latent representation learning and manifold regularization. *Neural Networks*, 117:163–178.

Tibshirani, R. (1996). Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B*, 58:267–88.

Vecoven, N. (2017). Feature selection with deep neural networks. Master's thesis.

Vincent, P., Larochelle, H., Lajoie, I., Bengio, Y., and Manzagol, P.-A. (2010). Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *Journal of Machine Learning Research*, 11(Dec):3371–3408.

Wang, H. and Xia, Y. (2009). Shrinkage estimation of the varying coefficient model. *Journal of the American Statistical Association*, 104:747–57.

Witten, D. M., Tibshirani, R., and Hastie, T. (2009). A penalized matrix decomposition, with applications to sparse principal components and canonical correlation analysis. *Biostatistics*, 10(3):515–534.

Ye, M. and Sun, Y. (2018). Variable selection via penalized neural network: a drop-out-one loss approach. In *International Conference on Machine Learning*, pages 5620–5629.

Zou, H., Hastie, T., and Tibshirani, R. (2006). Sparse principal component analysis. *Journal of Computational and Graphical Statistics*, 15(2):265–286.