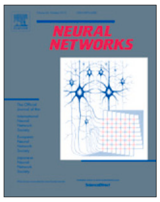
Neural Networks 142 (2021) 661–679

Contents lists available at ScienceDirect

Neural Networks

journal homepage: www.elsevier.com/locate/neunet



Unsupervised learning of disentangled representations in deep restricted kernel machines with orthogonality constraints

Francesco Tonin [*], Panagiotis Patrinos, Johan A.K. Suykens

Department of Electrical Engineering, ESAT-STADIUS, KU Leuven, Kasteelpark Arenberg 10, B-3001 Leuven, Belgium

A R T I C L E   I N F O

Article history:
Received 12 November 2020
Received in revised form 24 May 2021
Accepted 19 July 2021
Available online 26 July 2021

Keywords:
Kernel methods
Unsupervised learning
Manifold learning
Learning disentangled representations

A B S T R A C T

We introduce Constr-DRKM, a deep kernel method for the unsupervised learning of disentangled data representations. We propose augmenting the original deep restricted kernel machine formulation for kernel PCA by orthogonality constraints on the latent variables to promote disentanglement and to make it possible to carry out optimization without first defining a stabilized objective. After discussing a number of algorithms for end-to-end training, we quantitatively evaluate the proposed method's effectiveness in disentangled feature learning. We demonstrate on four benchmark datasets that this approach performs similarly overall to β-VAE on several disentanglement metrics when few training points are available while being less sensitive to randomness and hyperparameter selection than β-VAE. We also present a deterministic initialization of Constr-DRKM's training algorithm that significantly improves the reproducibility of the results. Finally, we empirically evaluate and discuss the role of the number of layers in the proposed methodology, examining the influence of each principal component in every layer and showing that components in lower layers act as local feature detectors capturing the broad trends of the data distribution, while components in deeper layers use the representation learned by previous layers and more accurately reproduce higher-level features.

1. Introduction

The choice of the features on which a machine learning method is trained on is crucial to achieve good performance. Finding representations of the data that make it easier to train classifiers or other predictors is the goal of representation learning (Bengio et al., 2013).

One desirable characteristic of good representations is disentanglement, which means that the learned representation separates the factors of variations in the data (Bengio, 2009; Bengio et al., 2013). In fact, in representation learning it is common to assume a low dimensional multivariate random variable z representing the meaningful factors of variations: a high dimensional observation x is then sampled from the conditional distribution $P(x|z)$. For example, a model trained on faces may learn latent generative factors such as hairstyle, emotion, or the presence of glasses. In this respect, the empirical success of deep learning in supervised learning tasks is often linked to the ability of deep networks to learn meaningful intermediate representations (Vincent et al., 2010; Zeiler & Fergus, 2014).

Regarding the unsupervised learning setting, many benefits of employing disentangled representations have been highlighted. For example, they could be useful in (i) transfer learning, by reusing meaningful representations on new tasks (Bengio et al., 2013), (ii) semi-supervised learning (Schölkopf et al., 2012; Ranzato et al., 2007), (iii) few-shot learning (Ridgeway & Mozer, 2018), (iv) explainability, for instance in the medical domain (Holzinger et al., 2019; Sarhan et al., 2019), and (v) reinforcement learning (Lake et al., 2017).

However, unsupervised learning of disentangled representations is still a key challenge in artificial intelligence research (Bengio et al., 2013; Lake et al., 2017; LeCun et al., 2015). State-of-the-art models are mostly based on the generative adversarial network (GAN) framework, such as InfoGAN (Chen et al., 2016), or on variational autoencoders (VAEs) (Kingma & Welling, 2014), including β-VAEs (Higgins et al., 2017), FactorVAE (Kim & Mnih, 2018) and β-TCVAE (Chen et al., 2018). Concerning the former approaches, it has been observed that training InfoGANs is difficult due to the training instability of the GAN framework (Higgins et al., 2017). Approaches based on VAEs offer training stability (Higgins et al., 2017), but a recent extensive empirical evaluation has found that their disentangling performance was not reliable as it varied widely with hyperparameter selection, random seeds, and among datasets (Locatello et al., 2019).

[*] Corresponding author.
E-mail addresses: francesco.tonin@esat.kuleuven.be (F. Tonin), panos.patrinos@esat.kuleuven.be (P. Patrinos), johan.suykens@esat.kuleuven.be (J.A.K. Suykens).

https://doi.org/10.1016/j.neunet.2021.07.023
0893-6080/© 2021 Elsevier Ltd. All rights reserved.

With respect to unsupervised feature extraction, principal component analysis (PCA) is a standard methodology. More in general, it is well established that kernel methods are stable to train and give reliable performance, which contrasts sharply with the issues outlined in the above paragraph. However, in contrast to deep networks, they are shallow methods, so they cannot take advantage of depth to hierarchically decompose a difficult target function into a composition of simpler functions (Allen-Zhu & Li, 2020). Finally, currently there is no widely accepted technique to promote disentanglement in kernel methods.

This paper introduces Constr-DRKM, a deep kernel method for the unsupervised learning of disentangled representations. Our approach is based on deep restricted kernel machines (deep RKMs) (Suykens, 2017), which provide a framework rooted in well-understood and reliable kernel methods. There has been little investigation on the use of deep RKMs for unsupervised learning. Moreover, while recent evidence suggests that shallow RKMs can achieve good disentangling performance (Pandey et al., 2021), no previous study has investigated the potential for unsupervised learning of disentangled representations in deep RKMs.

We propose employing a deep RKM consisting of multiple kernel PCA layers augmented by orthogonality constraints on the latent variables to perform unsupervised extraction of disentangled features. We explain that the orthogonality constraints have two effects, called *intraorthogonality* and *interorthogonality* effects, which encourage the deep RKM to learn a disentangled representation of the data. As a result of the introduction of the orthogonality constraints, we are able to avoid defining a stabilized version of the deep RKM objective, which is instead needed in the original formulation of deep RKMs (Suykens, 2017). Furthermore, we show how to train the proposed model end-to-end, so that the components of lower layers can use the representations learned by higher layers. To evaluate the proposed methodology, we conduct experiments in the task of disentangled feature learning. We quantitatively show that RKMs can benefit from depth and that a two-layer Constr-DRKM architecture performs similarly to the state of the art on several disentanglement metrics when only a fraction of training points are available while being more reliable in terms of sensitivity to hyperparameter selection than β-VAE.

Our main contributions are as follows.

- We propose a novel deep kernel method called Constr-DRKM by reformulating the deep RKM framework for kernel PCA (Suykens, 2017) into a constrained optimization problem with orthogonality constraints on the latent variables so that disentanglement is encouraged and optimization can be carried out without first defining a stabilized objective.
- We propose an end-to-end training scheme, allowing lower layers to exploit the representation learned by higher layers, by illustrating the use of multiple constrained optimization algorithms, including projected gradient and Cayley Adam (Li et al., 2019). We also introduce a deterministic initialization scheme to improve the reliability of training. Furthermore, we present a denoising procedure for deep RKMs.
- We show how to apply our proposed method to the unsupervised learning of disentangled representations without any prior knowledge on the generative factors, demonstrating empirically that its learned representations perform similarly overall compared to β-VAE in terms of multiple disentanglement metrics on four benchmark datasets when few training points are available. We also evaluate and discuss the influence of hyperparameters on the performance of the proposed method, demonstrating that our approach is less sensitive to hyperparameter choice than β-VAE: Constr-DRKM's disentangling performance tends to remain steady as its hyperparameter γ varies, in contrast to the strong influence of the hyperparameter β on β-VAE's performance. In particular, we show that deterministic initialization of Constr-DRKM's training algorithm considerably improves the reproducibility of the results.
- Finally, we show the benefit of Constr-DRKM over kernel PCA in denoising complex 2D data distributions. In addition, we study the influence of each principal component by visualizing the concept learned by each component in every layer. In this way, we illustrate the role of the different layers: the first layer performs lower-level feature detection and focuses on, for example, edges and corners, while the second layer employs those lower-level features and captures more global features, which in turn allow for more accurate reproduction of the details of the original data distribution.

2. Related work

A conventional method to extract features in an unsupervised manner is PCA (Jolliffe, 1986). As in the representation learning setting introduced in the previous section, the underlying assumption of PCA is that the input high-dimensional observations lie in a lower-dimensional linear subspace. PCA works through a linear transformation that projects the input observations in a new coordinate system, where the direction of maximum variance is called the first principal component, the orthogonal direction with the second-highest variance is called the second principal component, and so on. The dimensionality reduction is achieved by considering only the first s principal components. Kernel PCA is an extension of PCA that introduces nonlinearities by mapping the input observations to a higher dimensional feature space using a kernel function (Schölkopf et al., 1998). Linear PCA is then performed in that space. The conditions under which we can define the mapping φ used in Schölkopf et al. (1998) are given by Mercer's theorem (Mercer, 1909): if k is a positive-definite kernel function, then there exists a feature map φ such that $k(x,z) = \varphi(x)^T\varphi(z)$. The choice of some kernel function can be seen as choosing a suitable Reproducing Kernel Hilbert Space (RKHS); in this setting, Gnecco and Sanguineti (2009) studied suboptimal solutions to kernel PCA to handle large datasets, which constitute a considerable computational challenge to kernel methods. Another common approach to deal with this challenge is to employ the Nyström method (Williams & Seeger, 2001). Alternatively, under the assumption of sparse source signals, Georgiev et al. (2007) formulated conditions under which it is possible to recover a sparse signal using sparse component analysis (SCA) with no independence condition on the source signal, and Gnecco and Sanguineti (2010) examined sparse suboptimal solutions to kernel PCA. Our proposed method builds on multiple layers of kernel PCA and introduces orthogonality constraints on the latent variables to promote disentanglement. In contrast to our method, neither PCA nor kernel PCA exploits depth or deals with disentanglement.

In deep learning, pre-training each layer of a deep network in an unsupervised fashion before fine tuning the entire network is a common technique. In this context, several works have proposed unsupervised methodologies that learn data representations, including Ranzato et al. (2007), Vincent et al. (2008, 2010). For instance, Ranzato et al. (2007) proposed an unsupervised encoder–decoder architecture that employed multiple convolutions, sparsity constraints, and pooling to build a hierarchical representation of the data and boost invariance; in fact, pooling was shown to represent an information bottleneck that can promote invariance (Achille & Soatto, 2018). Another example is Vincent et al. (2010), where invariant representations were obtained using stacked denoising autoencoders. In that work, it was shown that using those representations to train a classifier led to improved accuracy not only for deep networks but also for support vector machines with radial basis function kernel, supporting our hypothesis that kernel methods can benefit from deep representations. Our method shares the idea of exploiting depth with the mentioned works, but it applies that idea in the framework of kernel methods instead of neural networks and it also takes disentanglement of the learned features into account.

One early approach to build disentangled representations was independent component analysis (ICA), which is a method to extract independent components from a signal (Comon, 1994). It is based on the assumption that the observed signal is a linear mixture of unknown latent signals that are non-Gaussian and mutually independent. However, it was shown empirically that ICA has poor disentangling performance compared to the state of the art (Higgins et al., 2017).

A more recent approach to the unsupervised learning of disentangled representations is InfoGAN (Chen et al., 2016). Building upon the GAN framework (Goodfellow et al., 2014), InfoGAN uses a generator G that aims to produce a realistic observation from the concatenation of a noise vector n with a latent representation z. Disentanglement in z is encouraged by adding a regularizer to the typical GAN objective. The regularizer aims to maximize a lower bound of the mutual information between z and the generated observation $G(n, z)$. In contrast to InfoGAN, our method is based on kernel methods, which provide more stable training procedures compared to the GAN framework. Finally, InfoGAN's disentangling performance has been empirically shown to be significantly lower than VAE-based methods (Higgins et al., 2017), which represent the state of the art.

Most VAE-based methods are based on the β-VAE approach (Higgins et al., 2017). In this setting, one assumes that the prior over the latent variables $P(z)$ is Gaussian. Then, the posterior $q_\phi(z|x)$, which is approximated using a deep neural network parametrized by ϕ, is learned by maximizing

$$L(\theta, \phi; x, z, \beta) = \mathbb{E}_{q_{\phi(z|x)}}\left[\log p_\theta(x|z)\right] - \beta D_{\mathrm{KL}}\left(q_\phi(z|x) \parallel p(z)\right), \quad (1)$$

where the likelihood $p_\theta(x|z)$ is approximated by a deep neural network parametrized by θ, β is a hyperparameter controlling the degree of disentanglement and D_{KL} is the Kullback–Leibler (KL) divergence defined as $D_{\mathrm{KL}}(r(x)|s(x)) = \mathbb{E}_r\left[\log r/s\right]$. In Higgins et al. (2017) it is claimed that fixing $\beta > 1$ promotes disentanglement because it imposes a stricter information bottleneck on the latent factors z. Chen et al. (2018) extend this explanation by identifying in (1) a total correlation term that was already known to be related with the disentanglement of a representation (Achille & Soatto, 2018; Ver Steeg & Galstyan, 2015). Similarly to Chen et al. (2018), FactorVAE (Kim & Mnih, 2018) augments the VAE objective (Eq. (1) with $\beta = 1$) with an approximation of the total correlation. However, Locatello et al. (2019) performed a large-scale experimental study analyzing six VAE-based methods, including β-VAE, β-TCVAE and FactorVAE, and concluded that they cannot be used to reliably learn disentangled representations in the unsupervised setting, as their disentangling performance was shown to vary widely with hyperparameter selection, random seeds and among datasets.

Regarding methods based on RKMs (Suykens, 2017), which is the framework our work is built upon, Pandey et al. (2021) has recently proposed a generative model called Gen-RKM, which was experimentally shown to be able to produce disentangled representations. Gen-RKM makes use of a single level of kernel PCA expressed in the RKM framework to derive the latent representations, while the method proposed in this paper introduces a deep architecture made up of multiple kernel PCA objectives. In fact, while Pandey et al. (2021) considered a single feature map consisting of a deep convolutional neural network, in this paper we consider several feature maps over multiple levels. In other words, following the terminology used in Suykens (2017), in the former case deep learning is only performed over layers, while in the latter case depth is given by multiple levels, each associated with a different feature map possibly consisting of multiple layers. For the sake of simplicity, in this paper the term "layer" is used instead of "level" when there is no ambiguity. Moreover, Constr-DRKM introduces new disentanglement constraints on the latent variables, defining in this way a constrained optimization problem that eliminates the need for the stabilization of the loss function used in Pandey et al. (2021) and Suykens (2017) and that can handle explicit and implicit feature maps in the same manner. Before introducing our proposed architecture, it is useful to first briefly illustrate the deep RKM framework.

3. Background: deep restricted kernel machines

This section reviews the framework of deep RKMs, as Constr-DRKM builds upon it. First, the RKM formulation of a single layer of kernel PCA is explained. Then, deep RKMs are introduced by means of an example. This section ends with the description of an open problem in deep RKMs: performing end-to-end training and promoting disentanglement at the same time.

3.1. Restricted kernel machine formulation of kernel PCA

The RKM formulation of kernel PCA gives another expression of the Least-Squares Support Vector Machine (LS-SVM) kernel PCA problem (Suykens et al., 2003) as an energy with visible and hidden units that is similar to the energy of restricted Boltzmann machines (RBMs) (Bengio, 2009; Fischer & Igel, 2014; Hinton et al., 2006; Salakhutdinov, 2015). In this new formulation, contrary to RBMs, both the visible units v_i and the hidden units h_i can be continuous. To derive this formulation, consider a training set of N data points of dimension d, a feature map $\varphi : \mathbb{R}^d \to \mathbb{R}^{d_F}$ and let s be the number of selected principal components. In the LS-SVM setting, the kernel PCA problem can be written as:

$$\begin{aligned} \underset{W, e_i}{\text{minimize}} \quad & J_{\mathrm{kpca}} = \frac{\eta}{2}\operatorname{Tr}(W^T W) - \frac{1}{2\lambda}\sum_{i=1}^{N} e_i^T e_i \\ \text{subject to} \quad & e_i = W^T \varphi(v_i), \quad i = 1, \ldots, N, \end{aligned}$$

where $W \in \mathbb{R}^{d_F \times s}$ is an unknown interconnection matrix, $e_i \in \mathbb{R}^s$ are the error variables and η and λ are hyperparameters. The RKM formulation of kernel PCA (Suykens, 2017) is given by an upper bound of J_{kpca} obtained by introducing the latent representations $h_i \in \mathbb{R}^s$, $i = 1, \ldots, N$ using:

$$\frac{1}{2\lambda} e^T e + \frac{\lambda}{2} h^T h \geq e^T h, \quad \forall e, h \in \mathbb{R}^s.$$

This leads to the following objective:

$$J_t = -\sum_{i=1}^{N} \varphi(v_i)^T W h_i + \frac{\lambda}{2}\sum_{i=1}^{N} h_i^T h_i + \frac{\eta}{2}\operatorname{Tr}\left(W^T W\right), \quad (2)$$

which is the formulation of kernel PCA in the RKM framework. The v_i are called visible units because their states are observed, while the hidden units h_i correspond to feature detectors (Hinton, 2012). In the representation learning literature, h_i is also known as the latent representation or code of v_i; we say that h_i consists of s latent or code variables or of s hidden features or units. Note that the first term of J_t is similar to the energy of an RBM.

As in other energy-based models, the RKM energy for kernel PCA associates a scalar value to each configuration of the variables. As a consequence, given training points $x_i,\ i = 1, \ldots, N$, training means clamping the visible units v_i to the training points x_i and finding an energy function for which the observed configurations of the visible units are given lower energies than unobserved configurations (LeCun et al., 2006). To do so, one characterizes the stationary points of J_t, which results in an eigenvalue problem of the kernel matrix $K \in \mathbb{R}^{N\times N}$, with $K_{ij} = \varphi(x_i)^T\varphi(x_j)$ (Suykens, 2017).

3.2. Deep restricted kernel machines

The theory of deep RKMs was initially proposed in Suykens (2017) to introduce a new perspective on the connection between deep learning and kernel machines. Such deep RKMs are obtained by coupling several RKMs in sequence, resulting in a deep architecture. A possible configuration of a deep RKM that extracts features of some data consists of two kernel PCA layers in sequence. Each kernel PCA layer follows the description given in Section 3.1. The architecture can be summarized in the following manner.

- Layer 1 consists of kernel PCA using as input the observation x_i from the given data. The features extracted by this layer are characterized by its hidden features $h_i^{(1)}$.
- Layer 2 consists of kernel PCA using as input the hidden features $h_i^{(1)}$ from the previous layer. The features extracted by this layer are characterized by its hidden features $h_i^{(2)}$.

Note that this architecture has a similar structure to stacked autoencoders (Bengio, 2009): each layer performs unsupervised learning and the hidden features produced by each layer serve as input to the next layer. The deep RKM is trained by considering an objective function that joins the objectives of each kernel PCA layer. To formalize the training objective, let s_1, s_2 be the number of selected principal components by the first and second layers of kernel PCA, respectively. Then, let $\varphi_1 : \mathbb{R}^d \to \mathbb{R}^{d_{\mathcal{F}_1}}$ be the feature map and of layer 1 and $\varphi_2 : \mathbb{R}^{s_1} \to \mathbb{R}^{d_{\mathcal{F}_2}}$ be the feature map of layer 2. Also, let $\lambda_1, \lambda_2, \eta_1, \eta_2 > 0$ be hyperparameters. The training objective of the above defined deep RKM is:

$$J_{t,\text{deep}} = J_1 + J_2,$$

where J_1 and J_2 are the objectives of a single layer of kernel PCA in the RKM framework as defined in (2) using the suitable input. That means:

$$J_1 = -\sum_{i=1}^{N} \varphi_1(v_i)^T W_1 h_i^{(1)} + \frac{\lambda_1}{2}\sum_{i=1}^{N} h_i^{(1)^T} h_i^{(1)} + \frac{\eta_1}{2}\operatorname{Tr}\left(W_1^T W_1\right)$$

and

$$J_2 = -\sum_{i=1}^{N} \varphi_2(h_i^{(1)})^T W_2 h_i^{(2)} + \frac{\lambda_2}{2}\sum_{i=1}^{N} h_i^{(2)^T} h_i^{(2)} + \frac{\eta_2}{2}\operatorname{Tr}\left(W_2^T W_2\right),$$

where $W_1 \in \mathbb{R}^{d_{\mathcal{F}_1}\times s_1}$ and $W_2 \in \mathbb{R}^{d_{\mathcal{F}_2}\times s_2}$ are the interconnection matrices of layer 1 and layer 2, respectively, and $h_i^{(1)} \in \mathbb{R}^{s_1}$ and $h_i^{(2)} \in \mathbb{R}^{s_2}$ are the hidden features of layer 1 and layer 2, respectively.

Training the above defined deep RKM means finding the interconnection matrices and the hidden features minimizing the energy $J_{t,\text{deep}}$. Since $J_{t,\text{deep}}$ is unbounded below, to make the energy suitable for minimization, Suykens (2017) proposed to instead minimize a stabilized version of $J_{t,\text{deep}}$. Following Pandey et al. (2021), this version is defined as:

$$J_{t,\text{deep}_\text{stab}} = J_{t,\text{deep}} + \frac{c_\text{stab}}{2} J_{t,\text{deep}}^2,$$

where $c_\text{stab} > 0$ is a hyperparameter. It can be shown that $J_{t,\text{deep}}$ and $J_{t,\text{deep}_\text{stab}}$ share the same stationary points (Pandey et al., 2021).

3.3. Effective algorithms for deep RKMs: an open problem

Deriving effective algorithms for training deep RKMs is an open problem. In Suykens (2017) a layer-wise training procedure was proposed for the case of linear kernels. However, previous research has stressed the importance of end-to-end training in deep architectures to produce representations able to efficiently approximate the target function (Allen-Zhu & Li, 2020). The ability of deep learning methods to produce such efficient representations is often linked to hierarchical learning because deep networks can hierarchically decompose a difficult target function into a composition of simpler functions (LeCun et al., 2015). This ability has been recently explained by a mechanism called "backward feature correction" (Allen-Zhu & Li, 2020), which means that layers of lower abstraction can use the representations learned by layers of higher abstraction to improve the quality of their representation. "Backward feature correction" and thus hierarchical learning cannot be achieved using layer-wise training alone.

On the other hand, it is interesting to note that training a deep RKM as the one described in Section 3.2 in a layer-wise manner yields mutually orthogonal hidden features, as they are obtained solving an eigenvalue problem of the symmetric kernel matrix K. This is an advantage when it comes to the disentanglement of the produced representations, as experimentally shown in Pandey et al. (2021) in the single-layer case. However, as it was explained in the previous paragraph, layer-wise training does not take full advantage of the deep architecture, so one would like to perform end-to-end training instead. In the end-to-end training case, one cannot simply solve two eigenvalue problems in sequence, so the mutual orthogonality of the hidden features is lost. Likely, this loss would affect negatively the disentanglement of the learned representations. In this context, the following section proposes Constr-DRKM, which is a method that allows to perform end-to-end training and to promote disentanglement at the same time.

4. The Constr-DRKM method

In the previous section, two key issues in the development of effective training algorithms for deep RKMs were identified: performing end-to-end instead of layer-wise training and promoting disentanglement in the hidden features at the same time. Accordingly, this section aims to propose a method, based on deep RKMs, for the unsupervised learning of latent representations of some given data so that

- it promotes disentanglement in the learned hidden features, and
- it carries out end-to-end training.

To address both aspects, we propose augmenting the original deep RKM formulation of two layers of kernel PCA, as described in Section 3.2, by orthogonality constraints on the latent variables of both layers. The proposed optimization problem is:

$$\begin{aligned}\underset{W_1,W_2,h_i^{(1)},h_i^{(2)}}{\text{minimize}} \quad & J_{t,\text{DRKM}} = J^{(1)}_{t,\text{DRKM}} + J^{(2)}_{t,\text{DRKM}} \\ &= -\sum_{i=1}^{N} \varphi_1(v_i)^T W_1 h_i^{(1)} + \frac{\lambda_1}{2}\sum_{i=1}^{N} h_i^{(1)^T} h_i^{(1)} \\ &\quad + \frac{\eta_1}{2}\operatorname{Tr}\left(W_1^T W_1\right) \\ &\quad - \sum_{i=1}^{N} \varphi_2(h_i^{(1)})^T W_2 h_i^{(2)} + \frac{\lambda_2}{2}\sum_{i=1}^{N} h_i^{(2)^T} h_i^{(2)} \\ &\quad + \frac{\eta_2}{2}\operatorname{Tr}\left(W_2^T W_2\right), \\ \text{subject to} \quad & \begin{bmatrix} H^{(1)^T} \\ H^{(2)^T} \end{bmatrix} \begin{bmatrix} H^{(1)} & H^{(2)} \end{bmatrix} = I_{s_1+s_2},\end{aligned} \tag{3}$$

where $H^{(1)} = [h_1^{(1)}, \dots, h_N^{(1)}]^T \in \mathbb{R}^{N\times s_1}$, $H^{(2)} = [h_1^{(2)}, \dots, h_N^{(2)}]^T \in \mathbb{R}^{N\times s_2}$, $I_{s_1+s_2}$ denotes the $(s_1+s_2)\times(s_1+s_2)$ identity matrix and $\lambda_1, \lambda_2, \eta_1, \eta_2 > 0$ are regularization constants.

The orthogonality constraints have two effects. The first effect, called the *intraorthogonality* effect, enforces the mutual orthogonality of the hidden features learned by the first layer, as well as the mutual orthogonality of the hidden features learned by the second layer. The second effect, called an *interorthogonality* effect, enforces the orthogonality between the hidden features learned by the first layer and the hidden features learned by the second layer so that the two layers are encouraged to learn new features of the data instead of repeating the same features in both layers. Both effects aim to push the deep RKM to learn a more disentangled representation of the data.

By employing end-to-end instead of layer-wise training in deep RKMs, lower layers can improve their representation by exploiting the representation learned by higher layers. The multi-layer architecture aims to encourage disentangled feature learning as well, as it has previously been observed that a two-level hierarchical structure can promote disentanglement (Esmaeili et al., 2019). Note that the formulation shown in (3) can be easily extended to more than two kernel PCA layers. One reason for making use of more layers is that it might improve the invariance of the learned representation, as invariance is promoted by stacking layers (Achille & Soatto, 2018). Another manner of boosting invariance would be to increase the information bottleneck between each layer by selecting fewer principal components (Achille & Soatto, 2018).

After having trained the model on all training data points x_i and having learned the hidden features $h_i^{(1)}$ and $h_i^{(2)}$ of each x_i, one can encode an out-of-sample data point $x^\star$ in the manner proposed in Pandey et al. (2020) extended to the two-layer case. The latent representation of $x^\star$ is computed by projecting it on the latent space using:

$$h^{(1)\star} = \frac{1}{\lambda_1\eta_1}\sum_{i=1}^{N} h_i^{(1)} k_0(x_i, x^\star), \tag{4}$$

$$h^{(2)\star} = \frac{1}{\lambda_2\eta_2}\sum_{i=1}^{N} h_i^{(2)} k_1(h_i^{(1)}, h^{(1)\star}), \tag{5}$$

where $k_0(x,y) = \varphi_1(x)^T\varphi_1(y)$ and $k_1(x,y) = \varphi_2(x)^T\varphi_2(y)$ are the kernel functions for the first and second layer, respectively, obtained by means of the kernel trick. Instead of first defining a feature map φ_i and then deriving the kernel, one can simply choose a positive definite kernel k_{i-1} due to Mercer's theorem (Mercer, 1909), which guarantees the existence of a feature map ϕ such that $k_{i-1}(x,y) = \phi(x)^T\phi(y)$.

4.1. A training algorithm for Constr-DRKM

In the energy-based interpretation of deep RKMs, the training phase of Constr-DRKM consists of finding the interconnection matrices W_1 and W_2 and the hidden units $h_i^{(1)}$ and $h_i^{(2)}$ so that the observed configurations x_i are given lower energies than unobserved configurations. Training of Constr-DRKM can be therefore performed by solving the equality constrained nonlinear optimization problem (3). However, the number of variables of (3) is large and depends on $d_{\mathcal{F}_1}$ and $d_{\mathcal{F}_2}$, which are the dimensionalities of the feature spaces used by φ_1 and φ_2. If, for instance, the Gaussian kernel is used, the dimensionality of the feature space would be infinite and therefore it would not be possible to directly solve (3). To address this issue, we show how to rewrite $J_{t,\text{DRKM}}$ to eliminate the interconnection matrices W_1 and W_2. This can be done considering each layer separately. The following shows how to eliminate the interconnection matrix for the first layer; eliminating W_2 follows the same procedure.

First, recall that, when training RKMs, each visible unit v_i is fixed to the training point x_i. Therefore, in training, the stationary point of $J_{t,\text{DRKM}}$ with respect to W_1 is given by:

$$\left.\frac{\partial J_{t,\text{DRKM}}}{\partial W_1}\right|_{v_i=x_i} = 0 \implies W_1 = \frac{1}{\eta_1}\sum_{i=1}^{N}\varphi(x_i)h_i^{(1)^T}. \tag{6}$$

Then, in order to eliminate the interconnection matrix W_1, two terms in $J^{(1)}_{t,\text{DRKM}}$ have to be rewritten: $\sum_{i=1}^{N}\varphi(v_i)^T W_1 h_i^{(1)^T}$ and $\frac{\eta_1}{2}\operatorname{Tr}\left(W_1^T W_1\right)$. This is accomplished using (6). Using the kernel trick, the latter term can be rewritten as follows:

$$\begin{aligned}\frac{\eta_1}{2}\operatorname{Tr}\left(W_1^T W_1\right) &= \frac{1}{2\eta_1}\operatorname{Tr}\left(\sum_{i=1}^{N}\sum_{j=1}^{N} h_i^{(1)}\varphi_1(x_i)^T\varphi_1(x_j)h_j^{(1)^T}\right) \\ &= \frac{1}{2\eta_1}\operatorname{Tr}\left(\sum_{i=1}^{N}\sum_{j=1}^{N} h_i^{(1)} k_0(x_i,x_j) h_j^{(1)^T}\right) \\ &= \frac{1}{2\eta_1}\operatorname{Tr}\left(H^{(1)^T}K^{(0)}H^{(1)}\right),\end{aligned}$$

where $K^{(0)} \in \mathbb{R}^{N\times N}$ such that $K^{(0)}_{ij} = k_0(x_i, x_j)$.

The former term can be rewritten as follows:

$$\begin{aligned}\sum_{i=1}^{N}\varphi_1(v_i)^T W_1 h_i^{(1)} &= \sum_{i=1}^{N}\varphi_1(v_i)^T\left(\frac{1}{\eta_1}\sum_{j=1}^{N}\varphi_1(x_j)h_j^{(1)^T}\right)h_i^{(1)} \\ &= \frac{1}{\eta_1}\sum_{i=1}^{N}\sum_{j=1}^{N}\varphi_1(v_i)^T\varphi_1(x_j)h_j^{(1)^T}h_i^{(1)} \\ &= \frac{1}{\eta_1}\sum_{i=1}^{N}\sum_{j=1}^{N} k_0(v_i,x_j)h_j^{(1)^T}h_i^{(1)} \quad \text{(kernel trick)} \\ &= \frac{1}{\eta_1}\operatorname{Tr}\left(H^{(1)^T}K_{xv}^{(0)}H^{(1)}\right),\end{aligned}$$

where $K_{xv}^{(0)} \in \mathbb{R}^{N\times N}$ such that $(K_{xv}^{(0)})_{ij} = k_0(v_i, x_j)$. Note that here W_1 was replaced using (6) by its expression in terms of the training points x_i, but the term $\varphi(v_i)$ was not expressed in terms of the training points. Keeping $\varphi(v_i)$ expressed in v_i is useful when, after training, the visible units are taken as unknowns. For instance, in RBMs this is the case in data generation.

Combining the above rewrites, $J^{(1)}_{t,\text{DRKM}}$ becomes:

$$\bar{J}^{(1)}_{t,\text{DRKM}} = -\sum_{i=1}^{N}\varphi_1(v_i)^T W_1 h_i^{(1)} + \frac{\lambda_1}{2}\sum_{i=1}^{N} h_i^{(1)^T} h_i^{(1)} + \frac{\eta_1}{2}\operatorname{Tr}\left(W_1^T W_1\right)$$

$$= -\frac{1}{\eta_1}\sum_{i=1}^{N}\sum_{j=1}^{N}\varphi_1(v_i)^T\varphi_1(x_j)h_j^{(1)^T}h_i^{(1)} + \frac{\lambda_1}{2}\sum_{i=1}^{N}h_i^{(1)^T}h_i^{(1)}$$
$$+ \frac{1}{2\eta_1}\mathrm{Tr}\left(\sum_{i=1}^{N}\sum_{j=1}^{N}h_i^{(1)}\varphi_1(x_i)^T\varphi_1(x_j)h_j^{(1)^T}\right)$$
$$= -\frac{1}{\eta_1}\sum_{i=1}^{N}\sum_{j=1}^{N}k_0(v_i,x_j)h_j^{(1)^T}h_i^{(1)} + \frac{\lambda_1}{2}\sum_{i=1}^{N}h_i^{(1)^T}h_i^{(1)}$$
$$+ \frac{1}{2\eta_1}\mathrm{Tr}\left(\sum_{i=1}^{N}\sum_{j=1}^{N}k_0(x_i,x_j)h_i^{(1)}h_j^{(1)^T}\right)$$
$$= -\frac{1}{\eta_1}\mathrm{Tr}\left(H^{(1)^T}K_{xv}^{(0)}H^{(1)}\right) + \frac{\lambda_1}{2}\mathrm{Tr}\left(H^{(1)}H^{(1)^T}\right)$$
$$+ \frac{1}{2\eta_1}\mathrm{Tr}\left(H^{(1)^T}K^{(0)}H^{(1)}\right).$$

The optimization problem of Constr-DRKM in (3) can finally be rewritten by taking the sum of the expressions for the first and second layers after weight elimination. Given that in training $K_{xv}^{(0)} = K^{(0)}$, as the visible units v_i are clamped to the training points x_i, and because $H^{(1)^T}H^{(1)} = I$ and $H^{(2)^T}H^{(2)} = I$ due to the orthogonality constraints, one can write:

$$\begin{aligned}\underset{h_i^{(1)},h_i^{(2)}}{\text{minimize}}\quad & \bar{J}_{t,\mathrm{DRKM}} = -\frac{1}{2\eta_1}\mathrm{Tr}\left(H^{(1)^T}K^{(0)}H^{(1)}\right) \\ & \qquad - \frac{1}{2\eta_2}\mathrm{Tr}\left(H^{(2)^T}K^{(1)}(H^{(1)})H^{(2)}\right) \\ \text{subject to}\quad & \begin{bmatrix}H^{(1)^T}\\ H^{(2)^T}\end{bmatrix}\begin{bmatrix}H^{(1)} & H^{(2)}\end{bmatrix} = I_{s_1+s_2},\end{aligned} \tag{7}$$

where $K^{(1)} \in \mathbb{R}^{N\times N}$ is the kernel matrix of the second layer defined on the hidden features of the previous layer as $K_{ij}^{(1)}(H^{(1)}) = k_1(h_i^{(1)}, h_j^{(1)})$. The optimization problem (7) has at least one global minimum due to the Weierstrass theorem, as the objective function is continuous and the feasible set is compact, and it may have multiple local minima since it is a non-convex problem. Compared to (3), it does not have the interconnection matrices as variables. This means that not only training is more efficient but also that Constr-DRKM can deal with explicit and implicit feature maps in the same manner, even if their feature space has infinite dimensionality. Furthermore, as a consequence of the introduced constraints and of the elimination of the interconnection matrices, it is not necessary to define a stabilized version of the objective to make it suitable for minimization, as it was instead needed in the original formulation of deep RKMs as explained in Section 3.2 with the additional hyperparameter c_{stab}.

End-to-end training of the proposed deep RKM can be performed by solving the equality constrained nonlinear optimization problem (7). The constraint set of (7) is a Stiefel manifold $\mathrm{St}(s_1+s_2, N)$, so one of the algorithms that has been proposed for optimization on the Stiefel manifold could be employed. For example, one could use Newton's method on the Stiefel manifold developed in Edelman et al. (1998) and Smith (1993, 1994). This algorithm generates points along the geodesic, which is expensive to compute because it uses matrix exponentials. Alternatively, one could avoid computing the geodesic by instead exploiting the Cayley transform to determine the search curve, such as in the algorithms proposed in Wen and Yin (2013), Zhu (2017). However, these methods could also be computationally heavy because the complexity of determining the search curve at each iteration is dominated by a matrix inversion.

A simple algorithm avoiding expensive matrix computations is the quadratic penalty optimization algorithm with warm start.

Algorithm 1 Example algorithm for training a two-layer Constr-DRKM with a quadratic penalty optimization algorithm with warm start. $(h_i^{(1)})_k^s$ denotes the starting point $h_i^{(1)}$ at iteration k. Note that Q also depends on the hyperparameters η_1, η_2 and on the kernel functions k_0 and k_1, which need to be chosen before optimization.

```
1: function TRAIN($(h_i^{(1)})_0^s$, $(h_i^{(2)})_0^s$, $\mu_0 > 0$, $\tau_0 > 0$, $p > 1$)
2:     for $k \leftarrow 0, 1, 2, \ldots$ do
3:         $h_i^{(1)} \leftarrow (h_i^{(1)})_k^s$
4:         $h_i^{(2)} \leftarrow (h_i^{(2)})_k^s$
5:         repeat
6:             Update $\{h_i^{(1)}, h_i^{(2)}\} \leftarrow$ ADAM($Q(h_i^{(1)}, h_i^{(2)}; \mu_k)$)
7:         until $\left\|\nabla Q(h_i^{(1)}, h_i^{(2)}; \mu_k)\right\| \le \tau_k$
8:         $\tau_{k+1} \leftarrow \tau_k/2$
9:         $\mu_{k+1} \leftarrow p * \mu_k$
10:        $(h_i^{(1)})_{k+1}^s \leftarrow h_i^{(1)}$
11:        $(h_i^{(2)})_{k+1}^s \leftarrow h_i^{(2)}$
12:    end for
13:    return $h_i^{(1)}, h_i^{(2)}$
14: end function
```

It works by first defining a function combining the objective of (7) with an additional term penalizing solutions violating the orthogonality constraints. One such function is called the quadratic penalty function $Q(h_i^{(1)}, h_i^{(2)}; \mu)$ and is defined as

$$Q(h_i^{(1)}, h_i^{(2)}; \mu) = \bar{J}_{t,\mathrm{DRKM}} + \frac{\mu}{2}\left\|\begin{bmatrix}H^{(1)^T}\\ H^{(2)^T}\end{bmatrix}\begin{bmatrix}H^{(1)} & H^{(2)}\end{bmatrix} - I_{s_1+s_2}\right\|_F^2,$$

where μ is a penalty parameter penalizing violations of the orthogonality constraints. The constrained optimization problem is then replaced by a sequence of unconstrained ones with increasing μ. In the kth unconstrained problem, the $h_i^{(1)}$ and $h_i^{(2)}$ minimizing $Q(h_i^{(1)}, h_i^{(2)}; \mu_k)$ are sought, with starting point set to the minimizers found in the $(k-1)$-th problem. The unconstrained problems can be then solved by some unconstrained minimization algorithm; e.g., Adam (Kingma & Ba, 2015). On the one hand, if these subproblems were solved to global optimality, every limit point of the sequence generated by Algorithm 1 would be a global minimizer of (7). On the other hand, only computing stationary points approximately for the subproblems prevents us from showing global optimality, so in general the quadratic penalty algorithm employed in Algorithm 1 does not converge to the global solution of (7). Detailed convergence properties of the quadratic penalty method are given in Theorems 17.1 and 17.2 in Nocedal and Wright (2006).

Optimizing (7) can also be addressed with more efficient optimization algorithms. An alternative algorithm is the projected gradient algorithm, whose iterates are specified by

$$H_{k+1} = \mathbf{\Pi}_{\mathrm{St}(s_1+s_2,N)}(H_k - \gamma_k\nabla\bar{J}_{t,\mathrm{DRKM}}(H_k)),$$

where $H_k = [H_k^{(1)}\ H_k^{(2)}]$, $\mathbf{\Pi}_{\mathrm{St}(s_1+s_2,N)}$ is the Euclidean projection onto the Stiefel manifold, and γ_k is the stepsize selected via backtracking. The projection is computed using the compact SVD of H_k. Another suitable optimization algorithm is Cayley Adam (Li et al., 2019), which can exploit the structure of the constraint set of (7). It approximates the Cayley transform iteratively requiring no matrix inversion. For Constr-DRKM training, one could for instance take five iterations for the Cayley transform approximation.

Contrary to the quadratic penalty method, projected gradient gives feasible iterates and does not need an outer optimization loop, which is not needed for Cayley Adam either. The iteration of projected gradient is the most expensive one due to the projection step. From this perspective, quadratic penalty and Cayley Adam are more suited to large-scale problems, even if they may require more iterations than projected gradient. The source code of multiple optimization algorithms for Constr-DRKM is available at https://github.com/taralloc/deeprkm.

In all algorithms, initialization can be done with random values drawn from the standard normal distribution or with deterministic layer-wise kernel PCA initialization, such that the initial hidden features of each layer are computed locally using kernel PCA. The kernels' hyperparameters can be optimized together with the latent variables by adding them as variables in (7). Alternatively, kernels' hyperparameter selection can be carried out by fixing a validation set and selecting the hyperparameter values that perform best on it. When it comes to the scalability of solving (7), the size of both $K^{(0)}$ and $K^{(1)}$ grows quadratically in N. On the other hand, their size does not depend on the dimensionality d of the input space. Regarding the unknown matrices $H^{(1)}$ and $H^{(2)}$, their size does not depend on d, but it depends on N, as well as on the number of selected principal components s_1 and s_2, respectively. In addition, $K^{(1)}$ is computed from $H^{(1)}$, so optimization might suffer from the increased non-linearity.

4.2. Denoising and Constr-DRKM

This section presents a reconstruction procedure that can denoise a test point $x^\star$. As in principal component analysis, denoising is carried out by keeping only the first s principal hidden features because one can assume that noise is concentrated in the components of lower variance. Performing reconstruction is straightforward in PCA because it is just a basis transformation, but the deep architecture and non-linear feature maps of Constr-DRKM pose a greater challenge. In fact, it is possible that a point in the feature space used by φ_1 does not have a pre-image in the input space. Moreover, multiple non-linear mappings have to be taken into account during reconstruction.

We adapt the approach proposed in Mika et al. (1999) and Schölkopf et al. (1999) in the context of kernel PCA to a two-layer Constr-DRKM. Extending the following procedure to more than two layers is straightforward. Call $\mathcal{F}_1$ the feature space used by φ_1 and assume that the mapped data points are centered in $\mathcal{F}_1$. First, $h^{(2)\star}$, the latent representation of $x^\star$ characterized by the second layer, is computed following (5). Similarly, $h^{(1)\star}$ is then computed following (4). Employing this representation, only s_1 components are kept, discarding the components that are noisier. The reconstruction of $x^\star$ from its projections z_k, $k = 1, \dots, s_1$ onto the first s_1 principal components in $\mathcal{F}_1$ is

$$\mathrm{P}_{s_1}\varphi_1(x^\star) = \sum_{k=1}^{s_1} z_k v^k, \tag{8}$$

where $v^k \in \mathcal{F}_1$ is the kth principal component. As explained in Schölkopf et al. (1998), v^k lies in the span of $\varphi_1(x_1)\dots\varphi_1(x_N)$, so (8) can be written as

$$\mathrm{P}_{s_1}\varphi_1(x^\star) = \sum_{k=1}^{s_1} z_k \sum_{i=1}^{N} \alpha_i^k \varphi_1(x_i). \tag{9}$$

The denoised point in the input space is computed by finding a point $\hat{x^\star}$ such that $\varphi_1(\hat{x^\star})$ approximates $\mathrm{P}_{s_1}\varphi_1(x^\star)$. To this aim, we minimize

$$r(\hat{x^\star}) = \left\|\varphi_1(\hat{x^\star}) - \mathrm{P}_{s_1}\varphi_1(x^\star)\right\|^2.$$

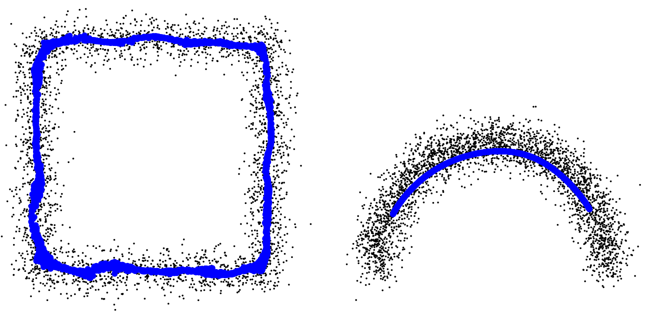

(a) Square. (b) Half circle.

Fig. 1. Denoising a square and a half circle. The noisy dataset is plotted as smaller black points, its denoised version as larger blue points. The number of selected principal components is $s_1 = 2$ for the first layer and $s_2 = 1$ for the second layer. In this experiment, $\sigma_n = 0.1$.

Using (9), the above can be rewritten as

$$\begin{aligned} r(\hat{x^\star}) &= \left\|\varphi_1(\hat{x^\star})\right\|^2 - 2\varphi_1(\hat{x^\star})\cdot \mathrm{P}_{s_1}\varphi_1(x^\star) + \left\|\mathrm{P}_{s_1}\varphi_1(x^\star)\right\|^2 \\ &= k_0(\hat{x^\star},\hat{x^\star}) - 2\sum_{k=1}^{s_1} z_k \sum_{i=1}^{N}\alpha_i^k k_0(\hat{x^\star}, x_i) + \left\|\mathrm{P}_{s_1}\varphi_1(x^\star)\right\|^2. \end{aligned} \tag{10}$$

Given that, by Eq. (17) of Schölkopf et al. (1998),

$$z_k = \sum_{i=1}^{N} \alpha_i^k k_0(x_i, x^\star),$$

that, in the LS-SVM formulation of kernel PCA, $\alpha_i = 1/\lambda_1 e_i$ (Suykens et al., 2002, 2003) and that $e_i = \lambda_1 h_i^{(1)}$ in restricted kernel machines (Suykens, 2017), one can rewrite (10) in terms of the hidden units as

$$r(\hat{x^\star}) = k_0(\hat{x^\star},\hat{x^\star}) - 2\sum_{i=1}^{N}\beta_i k_0(\hat{x^\star}, x_i) + \left\|\mathrm{P}_{s_1}\varphi_1(x^\star)\right\|^2, \tag{11}$$

where

$$\begin{aligned} \beta_i &= \sum_{k=1}^{s_1} z_k\alpha_i^k = \sum_{k=1}^{s_1}\left(\sum_{j=1}^{N}\alpha_j^k k_0(\hat{x^\star}, x_j)\right)\alpha_i^k \\ &= \sum_{j=1}^{N}\sum_{k=1}^{s_1}\alpha_j^k\alpha_i^k k_0(\hat{x^\star}, x_j) = \sum_{j=1}^{N}\sum_{k=1}^{s_1}(h_j^{(1)})_k(h_i^{(1)})_k k_0(\hat{x^\star}, x_j). \end{aligned}$$

In general, standard gradient descent methods can be employed to minimize (11); in this case, note that the last term of (11) is independent of $\hat{x^\star}$. In the case of the RBF kernel, Mika et al. (1999) proposed the following iteration scheme for $\hat{x^\star}$:

$$\hat{x^\star}_{t+1} = \frac{\sum_{i=1}^{N}\beta_i \exp\left(-\left\|\hat{x^\star}_t - x_i\right\|^2/(2\sigma_1^2)\right)x_i}{\sum_{i=1}^{N}\beta_i \exp\left(-\left\|\hat{x^\star}_t - x_i\right\|^2/(2\sigma_1^2)\right)}.$$

In denoising, the starting point is set to the noisy observation $x^\star$.

5. Experimental evaluation

The goal of the experimental evaluation is to test the feasibility of the proposed method for denoising and to assess the advantage of its architecture in the task of unsupervised disentangled feature learning.

5.1. Comparison of optimization algorithms

First, we evaluate the multiple optimization algorithms discussed in Section 4.1 in terms of speed and quality of the found solution. We train a Constr-DRKM model on a subset of $N = 1000$ of the MNIST dataset (LeCun et al., 2010). The model has two RBF layers ($\sigma^2 = 50$) with $s_1 = 10$ and $s_2 = 5$. For the penalty method, we set $\tau_0 = 0.00001$, $p = 8$, $\mu_0 = 1$ and we use L-BFGS to solve the subproblems. The learning rate for Cayley Adam is 5×10^{-5}. The termination condition for Cayley Adam and projected gradient is $\|H_{k+1} - H_k\| / \gamma_k \leq 5 \times 10^{-8}$ where γ_k is the stepsize, and for the outer loop of quadratic penalty is $\left\|H_{k+1}^T H_{k+1}\right\| \leq 10^{-9}$. The results are shown in Table 1. All algorithms achieve a similar final cost and all computed solutions show good feasibility. The lowest average cost is found by Cayley Adam. As expected, projected gradient and Cayley Adam require significantly fewer iterations than quadratic penalty. The iteration of Cayley Adam is about ten times cheaper than the one of projected gradient due to the computationally expensive SVD used in the latter. Even though in this experiment the iteration of Cayley Adam was less costly than the one of quadratic penalty, while performing the experiments in 5.3 we observed that quadratic penalty scaled better in terms of the number of unknown weights and is therefore more suitable for large scale problems. The training procedure described in Algorithm 1 is employed in the experiments in the following subsections.

5.2. Denoising

We applied the denoising procedure explained in Section 4.2 to complex 2D synthetic datasets. Each dataset is generated by selecting 3000 points as a training set and 750 additional points as a validation set. In this set of experiments, the noise n is white Gaussian with zero mean and standard deviation σ_n varying among different values. The number of selected components is $s_1 = 2$ for the first layer and $s_2 = 1$ for the second layer; the hidden units are initialized in a layer-wise manner with kernel PCA. All η and λ are set to 1. The kernel function employed in both layers is the RBF kernel and its bandwidth is selected employing the validation set. Constr-DRKM is first trained on the noisy points to find their latent representations and then each x_i is denoised by computing its pre-image minimizing the expression in (11).

First, a half circle and a square, depicted in Fig. 1, are considered. It can be seen that Constr-DRKM successfully captures the structure of the data distributions for both shapes. Note that denoising was effective even though the chosen overall number of principal components was higher than the dimensionality of the datasets. This would not have been the case with linear PCA, which performs perfect reconstruction, hence does not denoise, when using as many components as the dimensionality of the input data.

Secondly, two additional more complicated datasets were analyzed to study the role of each layer. The result of the influence of each component in every layer can be shown by denoising using only that component. For a dataset with a square and a spiral next to it, three plots, shown in Fig. 2, were produced. The first plot is the result of denoising using only the first component of the first layer: it learns the shapes, but has a few outliers and is still noisy around the center of the spiral. The second plot shows the denoised dataset using only the second component of the first layer: it does not show outliers and better reproduces

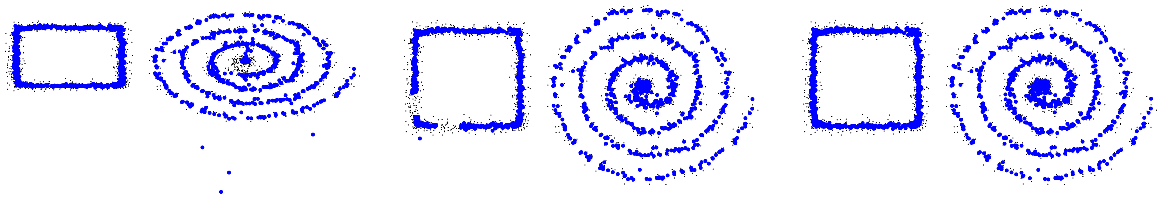
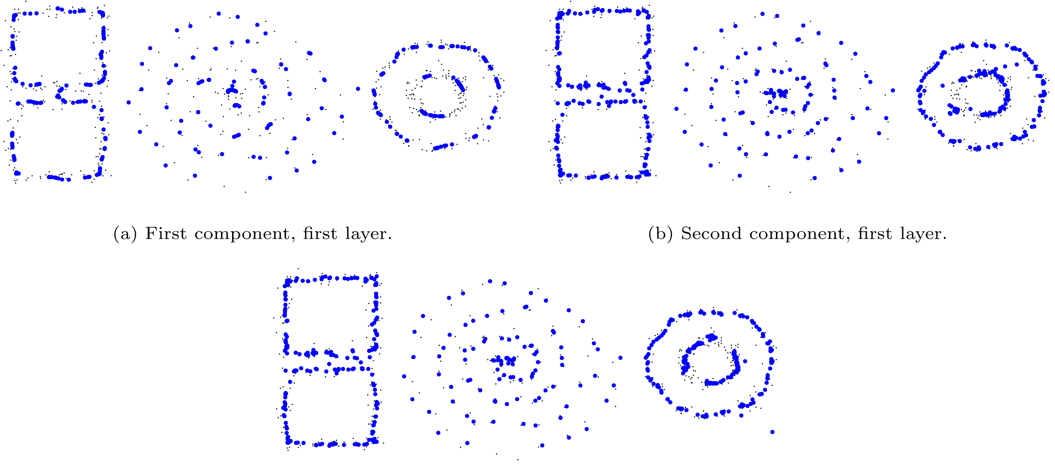

(a) First component, first layer. (b) Second component, first layer. (c) First component, second layer.

Fig. 2. Study of the influence of each component in every layer for a data distribution consisting of a square and a spiral. The number of selected principal components is $s_1 = 2$ for the first layer and $s_2 = 1$ for the second layer.


(a) First component, first layer. (b) Second component, first layer.

(c) First component, second layer.

Fig. 3. Study of the influence of each component in every layer for a data distribution consisting of two squares, a spiral and a ring. The number of selected principal components is $s_1 = 2$ for the first layer and $s_2 = 1$ for the second layer.

Table 1
Comparison of multiple training algorithms for Constr-DRKM on the MNIST dataset. Mean (standard deviation) over five initializations. Lower is better on all metrics.

Algorithm	Final cost	Final feasibility	Nr. iterations	Runtime (s)
Quadratic penalty	−605.8447 (0.00014)	1.37×10^{-11} (10^{-12})	191 993 (1493)	5784 (309)
Projected gradient	−605.8461 (0.00036)	5.51×10^{-12} (10^{-12})	345 (101)	30 (4)
Cayley Adam	−605.8464 (0.00006)	1.73×10^{-12} (10^{-12})	41 005 (7548)	399 (73)

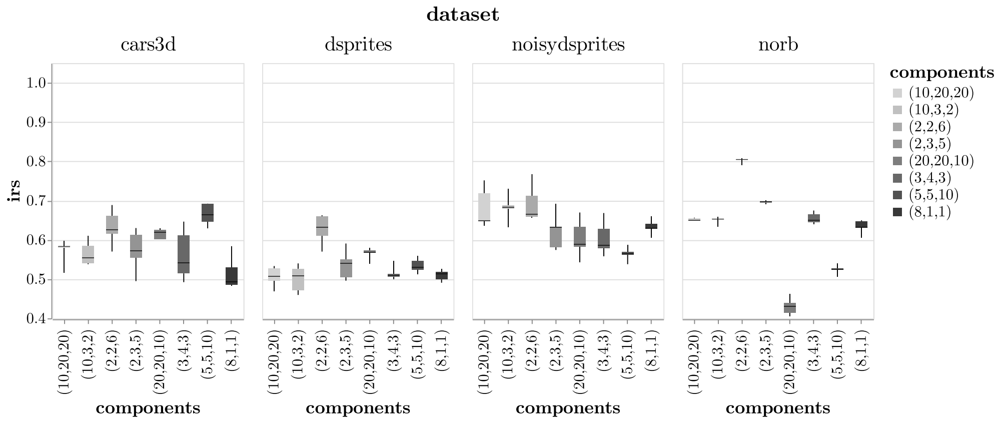


Fig. 4. Boxplot of the IRS score of a three-layer Constr-DRKM architecture according to the number of selected principal components for each dataset. The tuples in the labels are of the form (s_1, s_2, s_3). Lower and upper box boundaries the first and third quartiles, respectively, line inside box median, lower and upper error lines minimum and maximum values, respectively. The results are shown over five random seeds. A higher score indicates better disentangling performance.

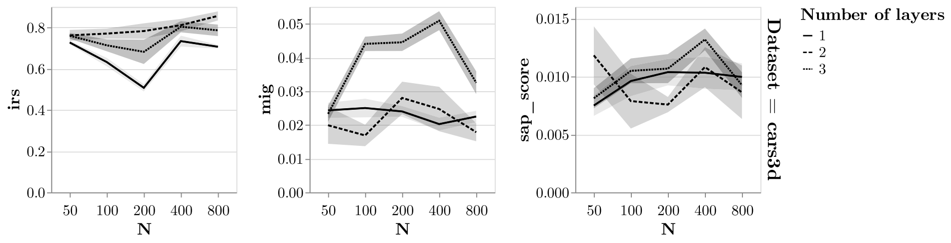


Fig. 5. Mean disentanglement score of different Constr-DRKM architectures according to the number N of training data points and the number n_{layers} of layers for the Cars3D dataset. For the one-layer architecture, $s_1 = 10$, for the two-layer one, $s_1 = 10$ and $s_2 = 5$ and for the three-layer one, $s_1 = 2$, $s_2 = 2$ and $s_3 = 6$. The size of each error band is set to the value of the standard error, extending from the mean. The variance is due to five different random seeds. The higher the curve, the better.

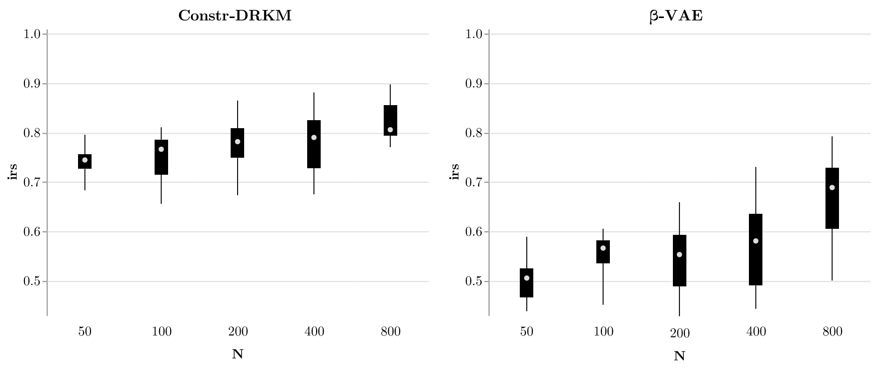
the higher frequency details around the center of the spiral but loses part of the square. The third plot is the result of denoising using only the first component of the second layer: it keeps the higher frequency details and reconstructs the square completely. Overall, the principal component of the first layer captures the broad trend but has outliers, while the second component learns the details of the shapes but loses the general trend in some regions. The component of the second layer, on the other hand, both picks up the general trend and reproduces the details of the shapes.

A similar analysis for a more complicated data distribution, consisting of two squares, a spiral and a ring, is shown in Fig. 3. Using only the first component of the first layer results in some artifacts: two distinct loops of the spiral intersect and two sides of the two squares are joined. On the other hand, the second component reconstructs those shapes correctly but does not well reproduce the inner circle of the ring. This circle is better reproduced by the first component of the second layer. The findings of the previous two experiments suggest that the lower layer in the deep architecture of Constr-DRKM functions as a lower-level feature detector focusing on the broad trends of the data distribution, while the higher layer exploits the representation learned by the lower layer and represents higher-level features which lead to better denoising and more accurate reproduction of the original data distribution. These results are consistent with previous findings in deep Boltzmann machines (Le Roux & Bengio, 2008) and in convolutional neural networks (Zeiler & Fergus, 2014) that lower layers focus on local features, such as edges and corners, and higher layers capture progressively more global and complex patterns with increasing invariance.

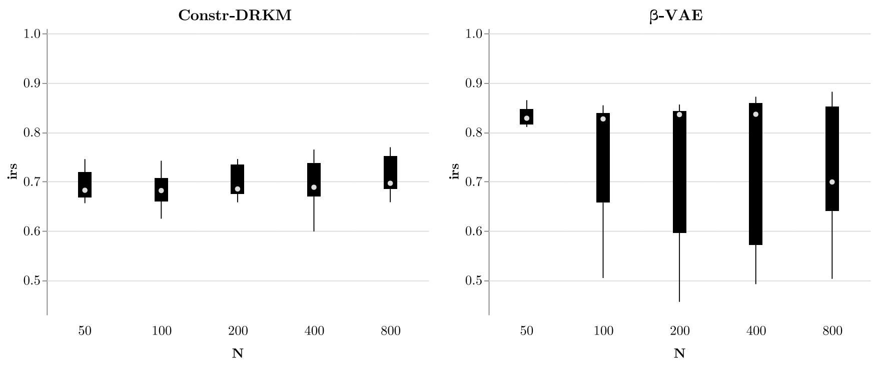
Table 2
Reconstruction error ratios between Constr-DRKM and kernel PCA in denoising complex 2D synthetic data distributions for different noise levels. Ratios larger than 1 mean that the deep architecture resulted in better denoising than the shallow one and the larger than 1 the better. In the deep architecture, the number of selected principal components is $s_1 = 2$ for the first layer and $s_2 = 1$ for the second layer, while the number of principal components used by kernel PCA is 3.

σ_n	Square	Half circle	Dataset of Fig. 2	Dataset of Fig. 3
0.05	1.22	1.36	3.18	2.51
0.1	1.09	1.17	1.70	1.62
0.2	1.06	1.08	1.24	1.21

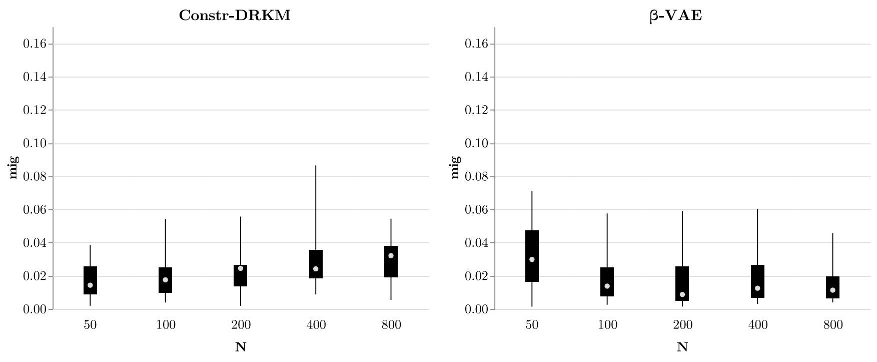
Finally, denoising performance was compared to kernel PCA with the same number of overall components. Table 2 reports the ratio of the reconstruction error, computed for all denoised points, between Constr-DRKM and kernel PCA. In the considered set of experiments, Constr-DRKM outperforms the shallow



(a) dataset = Cars3D, metric = IRS



(b) dataset = SmallNORB, metric = IRS



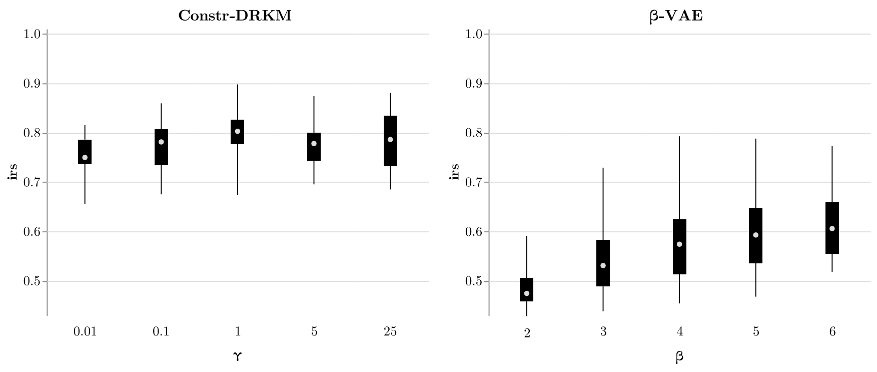
(c) dataset = SmallNORB, metric = MIG

Fig. 6. Boxplots of the disentanglement score of a two-layer Constr-DRKM, with $s_1 = 10$ and $s_2 = 5$, and of a β-VAE model according to the number of training points on the Cars3D and SmallNORB datasets. Disentanglement scores are shown across all choices of the hyperparameter γ for Constr-DRKM and of β for β-VAE and across five random seeds. The boxes show the first and third quartiles in the lower and upper box boundaries, respectively, the circle inside is the median and the lower and upper error lines are the minimum and maximum values, respectively. Higher is better on all metrics.

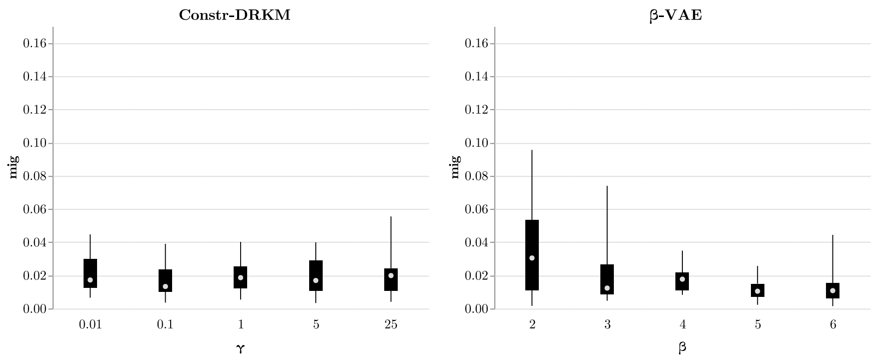
kernel PCA in denoising. More complicated data distributions, such as the ones in Figs. 2 and 3, show greater performance gain compared to the shallow architecture. This was expected because kernel PCA is known to be able to effectively denoise simple data distributions (Mika et al., 1999), but its denoising performance degrades as the datasets become more complex. The superiority of the deep architecture is strongly marked for small σ_n, which, together with the observations on the influence of the second layer made in the previous sets of experiments, seems to indicate that the additional layer has a crucial role in the reconstruction of the finer details of the data distribution, as this effect might become less noticeable with higher noise levels. Overall, our proposed architecture's representational efficiency benefited from depth, since it attained improved denoising of the data distributions in the same number of principal components as the shallow architecture.

5.3. Disentanglement

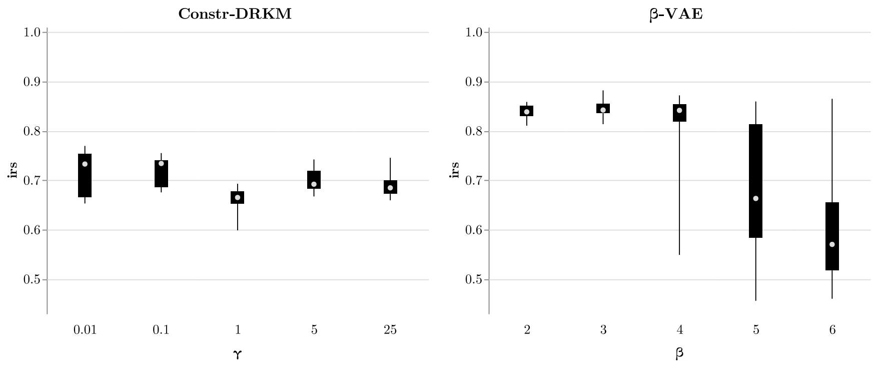
This section aims to quantitatively assess that Constr-DRKM is able to learn a disentangled representation of the factors of variation of the data. In addition, the role of the hyperparameters, of the number of selected principal components, and of the number of layers is studied. We applied our method on the Cars3D dataset (Reed et al., 2015), on the dSprites dataset (Higgins et al., 2017) and on the SmallNORB dataset (LeCun et al., 2004), as well as on a noisy version of dSprites introduced in Locatello et al. (2019) obtained by replacing the background pixels with Gaussian noise with zero mean and unit variance. In all datasets, each data point is generated according to a deterministic function of its ground-truth latent representation. All data points are 64×64 images: Cars3D and noisy dSprites contain RGB images, while dSprites and SmallNORB contain grayscale images. For details of the datasets, see Appendix A.1.



(a) dataset = Cars3D, metric = IRS



(b) dataset = Cars3D, metric = MIG



(c) dataset = SmallNORB, metric = IRS

Fig. 7. Boxplots of the disentanglement score of a two-layer Constr-DRKM, with $s_1 = 10$ and $s_2 = 5$, and of a β-VAE model according to the hyperparameter γ for Constr-DRKM and of β for β-VAE on the cars3D and SmallNORB datasets. Disentanglement score are shown across all N and across five random seeds. The boxes have the same structure as in Fig. 6. Higher is better on all metrics.

In the experiments, the dimension of the learned latent representation is fixed to 10: this choice was also made in the large experimental evaluation of Locatello et al. (2019) and, furthermore, this number is greater than but close to the ground-truth number of factor of variations. Computing the hidden features of some data point in Constr-DRKM translates to selecting 10 principal components. We chose to do so in the deep architecture with $n_{\text{layers}} \geq 1$ of Constr-DRKM by either fixing s_i to 10 for some i such that $1 \leq i \leq n_{\text{layers}}$ or by having $\sum_{i=1}^{n_{\text{layers}}} s_i = 10$ and concatenating all $h^{(i)}$. In all experiments, the learned models are evaluated on a subset of $N_{\text{eval}} = 4000$ data points chosen randomly from the relevant dataset.

Given that there is no single widely accepted measure to quantify disentanglement, we use three metrics that have been proposed in the literature, namely the IRS score (Suter et al., 2019), the mutual information gap (MIG) (Chen et al., 2018) and the SAP score (Kumar et al., 2018). The IRS metric measures robust disentanglement, which means that, if a latent variable is associated with some generative factor G, the inferred value of that latent variable shows little change when G remains the same, regardless of changes in the other generative factors. The MIG metric is computed as the average over all generative factors of the difference between the two latent variables with the highest mutual information with each generative factor. The SAP score is formulated by first building a score matrix S such that S_{ij} is the classification score of predicting the jth generative factor using only the ith latent variable; the final score is the mean of the differences between the top two entries for each column, which corresponds to averaging over the generative factors. For all considered metrics, a higher score indicates better disentangling performance.

In the first set of experiments, the role of the number of selected principal components is investigated. The studied architecture has $n_{\text{layers}} = 3$ and all η are set to 1. The chosen kernel function is the RBF kernel and its bandwidth is added as a variable to the optimization problem; this is the case for all experiments

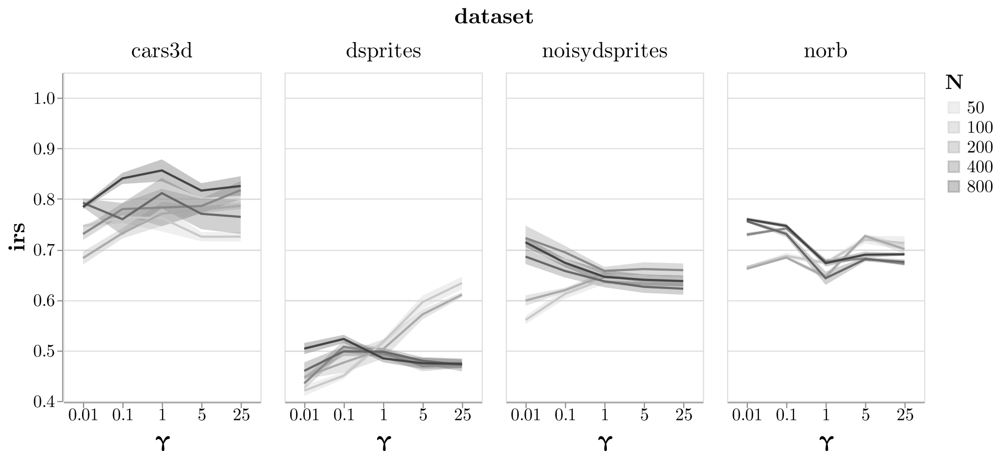


Fig. 8. Line chart of the mean IRS score of a two-layer Constr-DRKM architecture, with $s_1 = 10$ and $s_2 = 5$, according to the hyperparameter γ for each dataset and each number N of training points. The size of each error band is set to the value of the standard error, extending from the mean. The variance is due to five different random seeds. The higher the curve, the better.

Table 3
Comparison of disentangling performance of two Constr-DRKM models with different initialization and β-VAE on the Cars3D dataset with $N = 800$. Mean IRS and MIG scores are reported with their standard deviation, which is due to 5 different random seeds. The employed Constr-DRKM model has two RBF layers ($\sigma^2 = 50$) with $s_1 = 10$ and $s_2 = 5$. The β-VAE model has $\beta = 4$. Higher is better on all metrics.

Type	IRS	MIG
Random initialization	0.843 ± 0.044	0.012 ± 0.010
Layer-wise kernel PCA initialization	0.785 ± 0.000	0.040 ± 0.000
β-VAE	0.752 ± 0.145	0.010 ± 0.011

in this section. For each dataset, the training set is a random sample of $N = 100$ data points. Eight different choices for the number of selected components s_1, s_2 and s_3 are considered. Some representative results are presented in Fig. 4; for additional plots, see Appendix B.

Fig. 4 shows that the number of selected components has an important role in the disentangling performance of Constr-DRKM. For instance, in the SmallNORB dataset a bad choice of s_1, s_2 and s_3 led to considerably worse scores than the other choices. None of the evaluated choices of s_1, s_2 and s_3 consistently resulted in poor performance on all datasets: for instance, (20, 20, 10) was not the best performer on the SmallNORB dataset, but it was the choice with the second best median score on the dSprites dataset. Similarly, no choice of the number of selected components was found to always give a higher disentanglement score than any other choice for the datasets considered in the experiments. However, some choices led to better scores more consistently than others, while also showing smaller variance. For instance, models with $s_1 = 2$, $s_2 = 2$ and $s_3 = 6$ never resulted in significantly poorer performance than the other models and always had modest variance with respect to random seeds in the experiments. Interestingly, in general, randomness affected certain combinations of dataset and disentanglement metric more than others. For instance, in Fig. 4 most models on SmallNORB have small variance, whereas most models have higher variance on the other datasets.

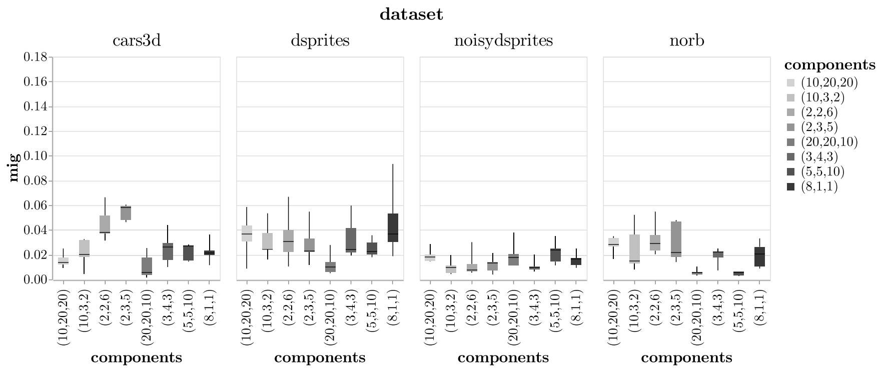
Next, the role of the number n_{layers} of layers on Constr-DRKM's disentangling performance is studied as the number N of training points grows from 50 to 800. In the experiments, n_{layers} is taken from $\{1, 2, 3\}$. The studied architectures are as follows: the one-layer architecture has $s_1 = 10$, the two-layer one has $s_1 = 10$ and $s_2 = 5$ and for the three-layer architecture, $s_1 = 2$, $s_2 = 2$ and $s_3 = 6$. For the two-layer and three-layer architectures, we chose those configurations because they were among the best ones that were empirically evaluated. All experiments are repeated over five random seeds. The results for the Cars3D dataset are shown in Fig. 5. The results indicate that the extent of the influence of n_{layers} on the disentangling performance of Constr-DRKM greatly depends on the disentanglement metric. In particular, on Cars3D all considered n_{layers} had similar SAP score, while varying the number of layers generally greatly affected the MIG and IRS scores. For example, models with 3 layers resulted in approximately double the MIG score on Cars3D compared to the two-layer and one-layer models when $N = 400$. Overall, for the datasets considered in these experiments, adding a third layer did not consistently increase the disentanglement scores, but this may not be the case on more difficult datasets with, for instance, multiple more realistic objects and a complex background. In the following experiments, we fix $n_{\text{layers}} = 2$, as it was in most cases the best or close to the best choice for any considered combination of metric, dataset, and N.

The performance of a two-layer Constr-DRKM is now studied as the number N of training points grows from 50 to 800 and is compared against β-VAE (Higgins et al., 2017). The studied architecture has $s_1 = 10$ and $s_2 = 5$. Regarding the model hyperparameters η_1 and η_2, they have the role of weights in the objective function, so we can consider a single hyperparameter $\gamma = \frac{\eta_1}{\eta_2}$. In the experiments, γ is taken from $\{0.01, 0.1, 1, 5, 25\}$. We repeat the same experiments using a β-VAE model in place of Constr-DRKM where we vary β in $\{2, 3, 4, 5, 6\}$. Some key results are presented in Fig. 6; experiments are repeated five times.

Overall, Constr-DRKM showed good disentangling performance compared to β-VAE across datasets and metrics. On Cars3D, Constr-DRKM outperformed β-VAE in the IRS score; on SmallNORB, Constr-DRKM and β-VAE produced similar MIG scores, but the latter method resulted in better median IRS scores. If we now turn to the analysis of variance, in accordance with Locatello et al. (2019), β-VAE's performance varied greatly with random seed and hyperparameter. For example, on SmallNORB in Fig. 6b the attained score varies from about 0.5 up to almost 0.9 with considerable interquartile range for all N but the smallest. Comparing β-VAE's variance to Constr-DRKM's, it can be seen that our method shows a significantly more limited variance for all N. Even on other datasets, Constr-DRKM shows overall smaller variance on the IRS score than the one shown by β-VAE. This is not always the case for other metrics, as exemplified in Fig. 6c. In general, the observations made in this set of experiments suggest that Constr-DRKM is able to consistently learn a disentangled

Table A.1
Key properties of the datasets used in the experimental evaluation.

Dataset	Input dimensions	# factors of variation	Meaning of the factors of variation and # possible values	Total # data points
Cars3D	3 × 64 × 64	3	– elevation (4 possible values) – azimuth (24 possible values) – object type (183 possible values)	17 568
dSprites	1 × 64 × 64	5	– shape (3 possible values) – scale (6 possible values) – orientation (40 possible values) – position x (32 possible values) – position y (32 possible values)	737 280
Noisy dSprites	3 × 64 × 64	3	– shape (3 possible values) – scale (6 possible values) – orientation (40 possible values) – position x (32 possible values) – position y (32 possible values)	737 280
SmallNORB	1 × 64 × 64	4	– category (5 possible values) – elevation (9 possible values) – azimuth (18 possible values) – lighting condition (6 possible values)	4860



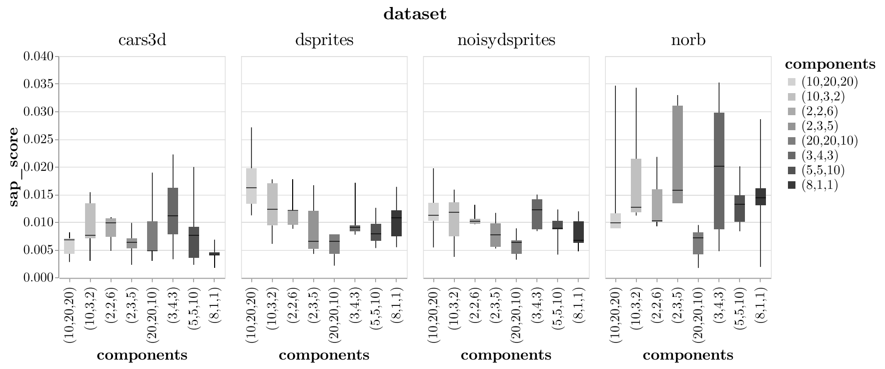


Fig. B.1. Disentanglement score of a three-layer Constr-DRKM architecture according to the number of selected principal components for each dataset for the MIG and SAP metrics. Same structure as Fig. 4.

representation of the input data, while being less affected by randomness and hyperparameter selection than β-VAE.

We now focus on the influence of the hyperparameter γ for Constr-DRKM and β for β-VAE. Fig. 7 plots the disentanglement score attained by both Constr-DRKM and β-VAE as the hyperparameter γ for Constr-DRKM and β for β-VAE varies. The variance is due to different N, which is in the same range as in Fig. 6, and five random seeds. From Fig. 7c it can be observed that γ has little influence on the IRS score on SmallNORB compared to β, which plays an important role in β-VAE's median performance, as setting β to 5 and 6 resulted in reduced performance and greater variance. A similar but less sudden trend can be noted in Fig. 7a on Cars3D: increasing β leads to an increased score, but the median score remains approximately steady when increasing γ. Overall, these results suggest that Constr-DRKM is robust to different weightings of the layers, as γ weights the relative importance of the two layers. In general, therefore, Constr-DRKM is less sensitive to γ than β-VAE is to β in the considered datasets and disentanglement metrics.

The influence of γ is also studied separately for each $N \in \{50, 100, 200, 400, 800\}$ in Fig. 8. It can be noted that most lines are roughly horizontal, meaning that varying γ does not significantly affect the IRS score and that this behavior is shared by all considered number of training points, except for the lowest two N on dSprites, where a higher weight on the first layer led to better performance. These results corroborate the findings of the previous set of experiments: Constr-DRKM's disentangling performance tends to remain steady as its γ hyperparameter varies, contrary to the behavior of β-VAE with respect to its hyperparameter β, which greatly influences its performance.

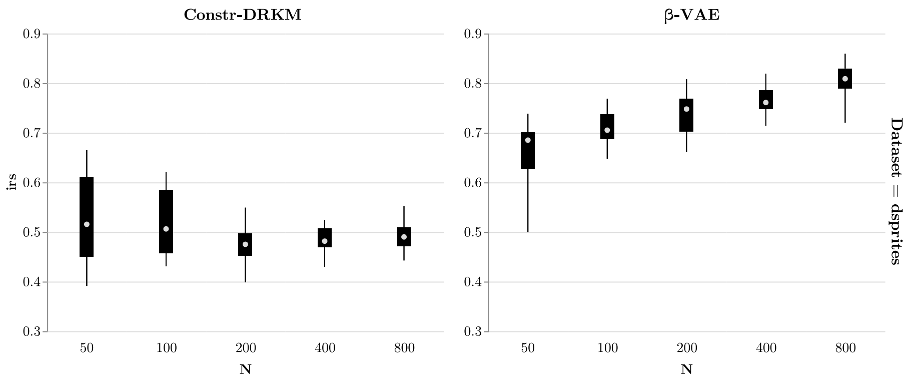


Fig. B.2. IRS score of a two-layer Constr-DRKM, with $s_1 = 10$ and $s_2 = 5$, and of a β-VAE model according to the number of training points. The plot has the same structure as Fig. 6.

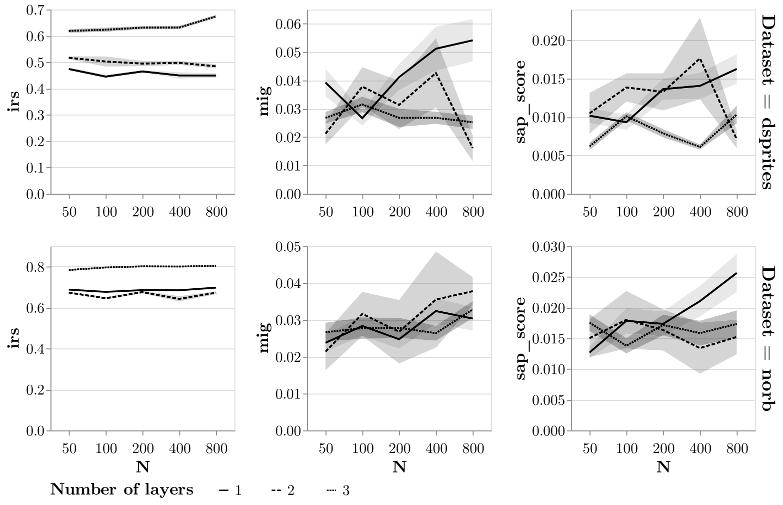


Fig. B.3. Mean disentanglement score of Constr-DRKM architectures for different n_{layers}. Same structure as Fig. 5.

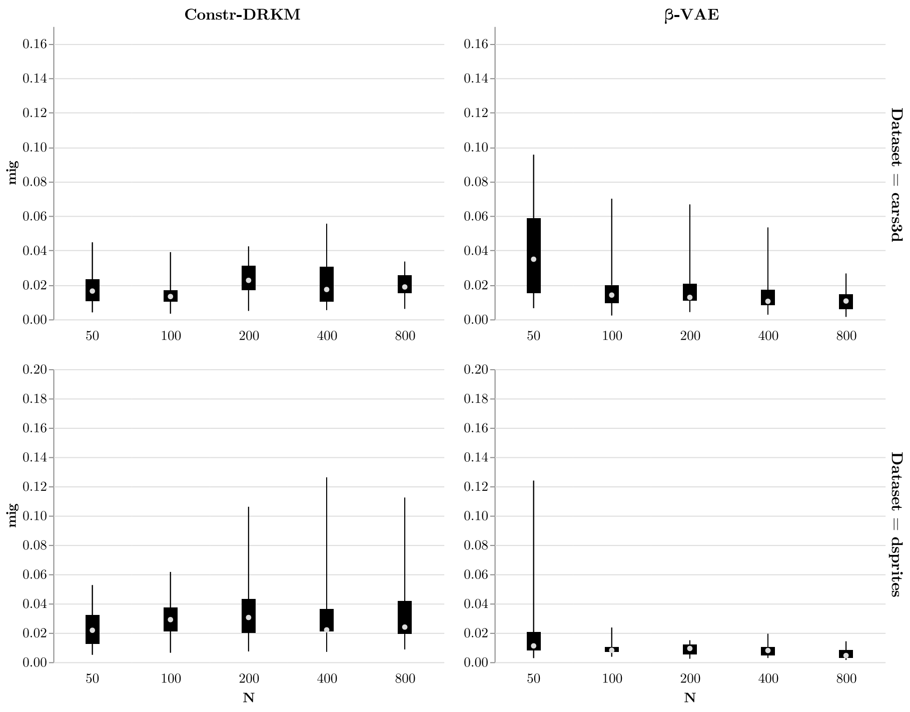


Fig. B.4. MIG score of a two-layer Constr-DRKM and of β-VAE according to N. Same structure as Fig. 6.

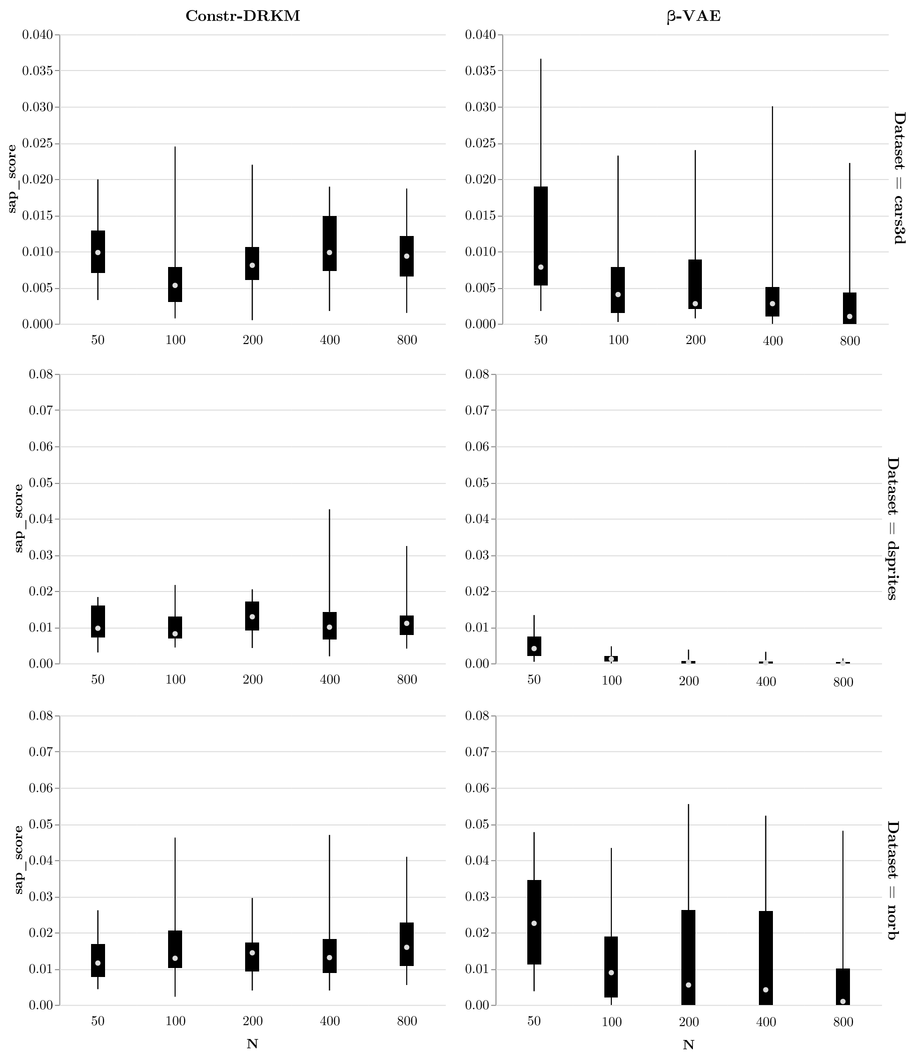


Fig. B.5. SAP score of a two-layer Constr-DRKM and of β-VAE. Same structure as Fig. 6.

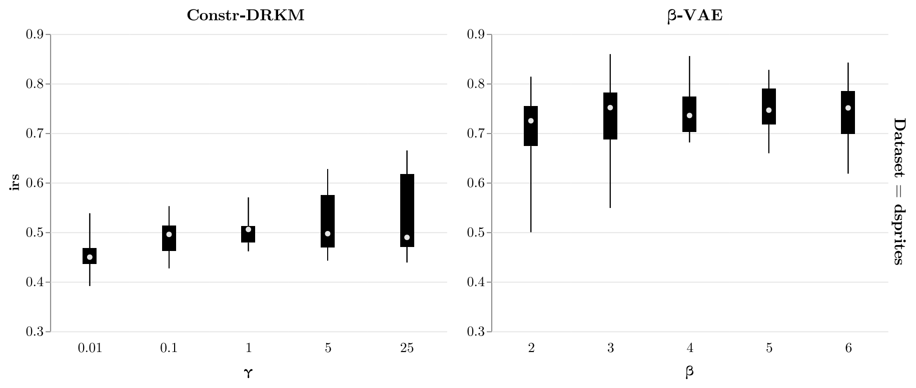


Fig. B.6. IRS score of a two-layer Constr-DRKM and of a β-VAE model according to the hyperparameters on the dSprites dataset. The plot has the same structure as Fig. 7.

Finally, we compare random initialization to layer-wise kernel PCA initialization. Table 3 shows the mean disentanglement scores attained on Cars3D. In this experiment, initializing the hidden units locally using kernel PCA in a layer-wise manner outperforms β-VAE and it provides the additional benefit of increased reliability as the standard deviation is zero because no random seed is employed. In particular, layer-wise kernel PCA initialization on average achieves a slightly lower IRS score compared to random initialization and it significantly outperforms both random initialization and β-VAE on the MIG metric. Overall, when it comes to the variance of the results, Constr-DRKM with layer-wise kernel PCA initialization compares favorably to β-VAE, successfully addressing the core issue of reliability of VAE-based methods brought up by Locatello et al. (2019).

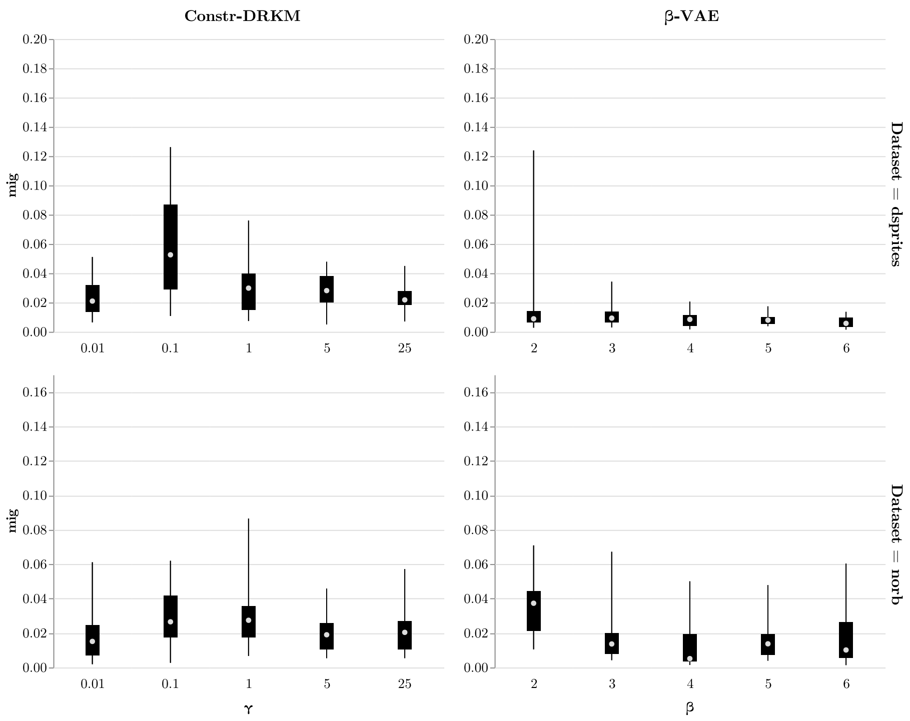


Fig. B.7. MIG score of a two-layer Constr-DRKM and of a β-VAE model according to the hyperparameters. The plot has the same structure as Fig. 7.

6. Conclusion

In this work, we have proposed to reformulate the deep restricted kernel machine framework for kernel PCA (Suykens, 2017) into a constrained optimization problem with orthogonality constraints on the latent variables. We have discussed and empirically evaluated a variety of optimization algorithms to learn the hidden features in an end-to-end manner instead of layer-wise. We have then shown how the proposed method can be applied to denoising and unsupervised disentangled feature learning without any prior knowledge on the generative factors. In the former task, we studied the role of each principal component in every layer showing that components in the first layer perform lower-level feature detection, while components in the second layer employ the representation learned by lower layers and extract more global features, more accurately reproducing the original data distribution. In our experiments in the task of disentangled feature learning, the proposed Constr-DRKM method quantitatively performed similarly overall compared to β-VAE (Higgins et al., 2017) on four benchmark datasets in several disentanglement metrics when few training points are available. In addition, regarding the issue raised in Locatello et al. (2019) that performance of state-of-the-art approaches to disentangled feature learning based on VAEs greatly varies when changing random seed or hyperparameter, Constr-DRKM was less sensitive to randomness and hyperparameter choice compared to β-VAE. In particular, the variance due to Constr-DRKM's hyperparameter γ was smaller than the variance due to the hyperparameter β in β-VAE and it was shown that Constr-DRKM with deterministic layer-wise kernel PCA initialization attained favorable scores without the need of a random seed, considerably improving the reproducibility of the results. Finally, the experimental analysis of the number of layers of Constr-DRKM indicates that adding a layer can increase the disentangling performance, as it was observed that a two-layer architecture is a better choice than a single layer one. Nevertheless, three-layer models did not consistently perform better than two-layer models. This result does not rule out the influence of other factors, as, for example, more challenging datasets may benefit from additional layers. In future work, applying Constr-DRKM to more complicated datasets may be useful to better understand the role of the number of layers. Furthermore, it would be interesting to investigate more advanced constrained optimization algorithms that could boost training efficiency.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

EU: The research leading to these results has received funding from the European Research Council under the European Union's Horizon 2020 research and innovation program/ERC Advanced Grant E-DUALITY (787960). This paper reflects only the authors' views and the Union is not liable for any use that may be made of the contained information. Research Council KUL: Optimization frameworks for deep kernel machines C14/18/068. Flemish Government: FWO: projects: GOA4917N (Deep Restricted Kernel Machines: Methods and Foundations), PhD/Postdoc grant Impulsfonds AI: VR 2019 2203 DOC.0318/1QUATER Kenniscentrum Data en Maatschappij. This research received funding from the Flemish Government (AI Research Program), Belgium. Johan Suykens and Panagiotis Patrinos are affiliated to Leuven.AI — KU Leuven institute for AI, B-3000, Leuven, Belgium. Ford KU Leuven Research Alliance Project KUL0076 (Stability analysis and performance improvement of deep reinforcement learning algorithms). EU H2020 ICT-48 Network TAILOR (Foundations of Trustworthy AI — Integrating Reasoning, Learning and Optimization). This work was supported by the Research Foundation Flanders (FWO) research, Belgium projects G086518N, G086318N, and G0A0920N; Fonds de la Recherche Scientifique — FNRS, Belgium and the Fonds Wetenschappelijk Onderzoek — Vlaanderen, Belgium under EOS

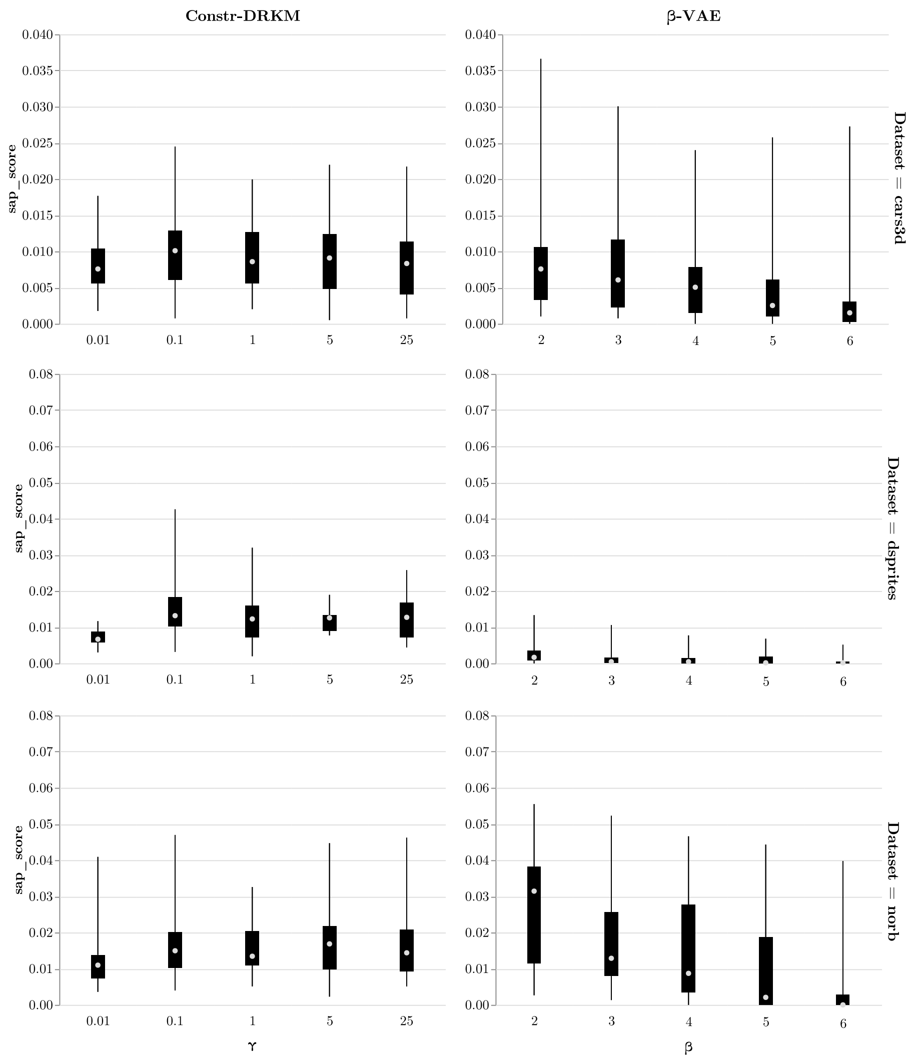


Fig. B.8. SAP score of a two-layer Constr-DRKM and of a β-VAE model according to the hyperparameters. The plot has the same structure as Fig. 7.

Project No. 30468160 (SeLMA). The computational infrastructure and services used in this work were provided by the VSC (Flemish Supercomputer Center), funded by the Research Foundation — Flanders (FWO), Belgium and the Flemish Government, Belgium.

Appendix A. Further details on the experimental evaluation

This section details the setup of the experiments of Section 5.3.

A.1. Datasets

Four datasets are used in the experimental evaluation: the Cars3D dataset (Reed et al., 2015), the dSprites dataset (Higgins et al., 2017), the SmallNORB dataset (LeCun et al., 2004) and a noisy version of dSprites introduced in Locatello et al. (2019). Each data point is generated deterministically from a tuple of factors of variations. The number of the factors of variations varies across datasets. Each factor of variation can take a finite number of values, so the number of training points is fixed and is the number of all possible combinations of the factors of variations.

Table A.1 summarizes the key properties of the datasets considered in the experimental evaluation.

A.2. Hyperparameter selection

The chosen algorithm for the unconstrained optimization problems is Adam (Kingma & Ba, 2015) with learning rate fixed to 10^{-3} and in Algorithm 1 we set $\mu_0 = 1$, $\tau_0 = 0.1$ and $p = 8$. We halt the outer loop of Algorithm 1 after a fixed number of iterations that we choose so that it is larger when the number of variables of the optimization problem is higher. This choice translates to running more outer iterations when the number of training point N is higher, as the number of variables depends on N. In particular, the maximum number of outer iterations was set to 2 for $N = 50$ and $N = 100$, to 4 for $N = 200$ and to 7 for $N = 400$ and $N = 800$. The maximum number of inner iterations was set to 500.

The β-VAE encoder is constructed following the architecture proposed in Higgins et al. (2017). The inputs are images x of dimension $c \times 64 \times 64$, where c is 1 for dSprites and SmallNORB and is 3 for Cars3D and noisy dSprites. We encode x using the following network: `conv` 32 $\to$ `conv` 32 $\to$ `conv` 64 $\to$ `conv` 64 $\to$ `conv` 256 $\to$ FC 256 $\times$ 20, where each `conv` block is a 4×4 convolution with stride 2 except the last block with stride 4. Each `conv` block is followed by `ReLU`.

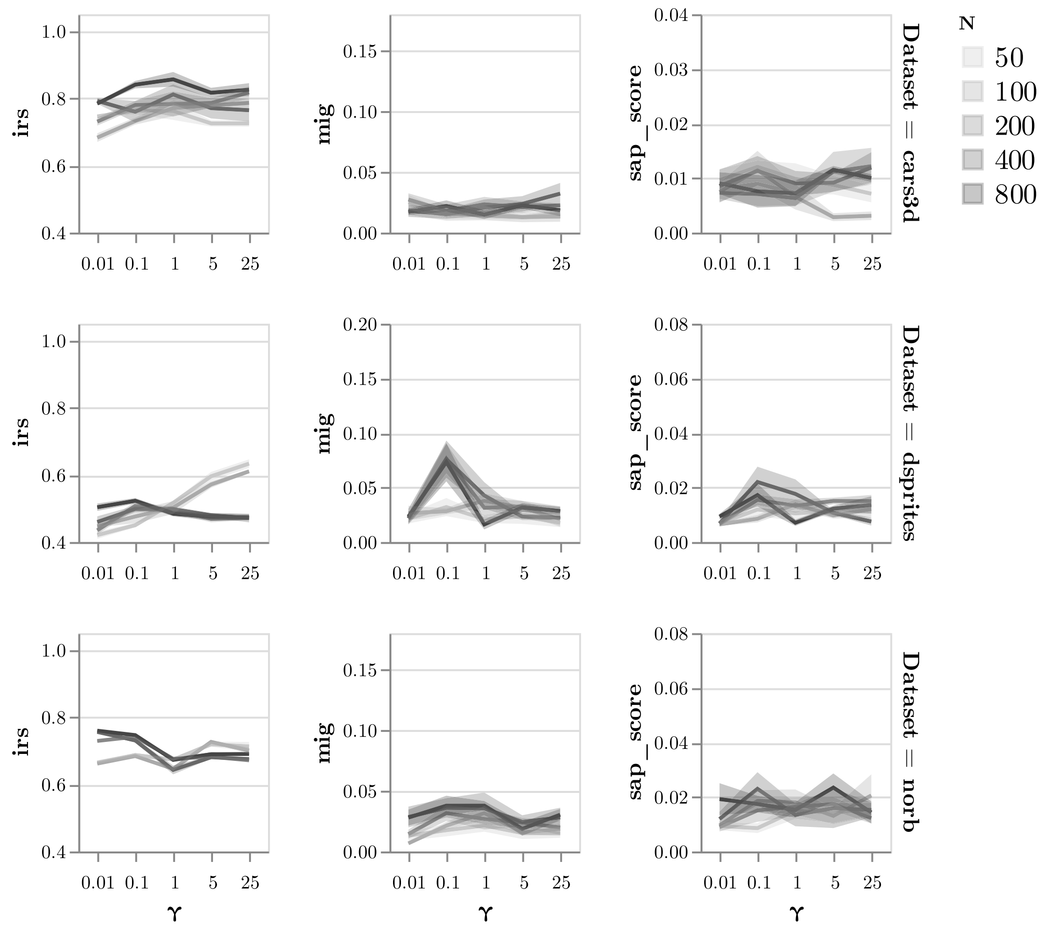


Fig. B.9. Line chart of the mean disentanglement score of a two-layer Constr-DRKM architecture, with $s_1 = 10$ and $s_2 = 5$, according to the hyperparameter γ for each number N of training points. The structure of the plot is the same as Fig. 8.

Appendix B. Additional plots of the experimental results

This section presents additional plots from our experiments discussed in Section 5.3. First, we show plots investigating the influence of the number of selected principal components in a three-layer architecture (Fig. B.1) and the role of the number of layers (Fig. B.3). Then, we compare Constr-DRKM's performance to β-VAE's on multiple datasets and disentanglement metrics in Figs. B.2, B.4 and B.5, focusing on the role of the hyperparameters γ and β in Figs. B.6, B.7, B.8, and B.9.

References

Achille, A., & Soatto, S. (2018). Emergence of invariance and disentanglement in deep representations. *Journal of Machine Learning Research*, *19*(1), 1947–1980.
Allen-Zhu, Z., & Li, Y. (2020). Backward feature correction: how deep learning performs deep learning. ArXiv Preprint.
Bengio, Y. (2009). Learning deep architectures for AI. *Foundations and Trends in Machine Learning*, *2*(1), 1–127.
Bengio, Y., Courville, A., & Vincent, P. (2013). Representation learning: A review and new perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *35*(8), 1798–1828.
Chen, X., Duan, Y., Houthooft, R., Schulman, J., Sutskever, I., & Abbeel, P. (2016). Infogan: interpretable representation learning by information maximizing generative adversarial nets. In *Proceedings of the 30th international conference on neural information processing systems, Vol. 29* (pp. 2172–2180). Curran Associates Inc..
Chen, T. Q., Li, X., Grosse, R. B., & Duvenaud, D. K. (2018). Isolating sources of disentanglement in variational autoencoders. In *Proceedings of the 32nd international conference on neural information processing, Vol. 31* (pp. 2615–2625). Curran Associates Inc..
Comon, P. (1994). Independent component analysis, A new concept? *Signal Processing*, *36*(3), 287–314.
Edelman, A., Arias, T. A., & Smith, S. T. (1998). The geometry of algorithms with orthogonality constraints. *SIAM Journal on Matrix Analysis and Applications*, *20*(2), 303–353.
Esmaeili, B., Wu, H., Jain, S., Bozkurt, A., Siddharth, N., Paige, B., Brooks, D. H., Dy, J., & Meent, J.-W. (2019). Structured disentangled representations. In *Proceedings of the 22nd international conference on artificial intelligence and statistics, Vol. 89* (pp. 2525–2534). PMLR.
Fischer, A., & Igel, C. (2014). Training restricted Boltzmann machines: an introduction. *Pattern Recognition*, *47*(1), 25–39.
Georgiev, P., Theis, F., Cichocki, A., & Bakardjian, H. (2007). Sparse component analysis: A new tool for data mining. In *Data mining in biomedicine* (pp. 91–116). Springer.
Gnecco, G., & Sanguineti, M. (2009). Accuracy of suboptimal solutions to Kernel principal component analysis. *Computational Optimization and Applications*, *42*(2), 265–287.
Gnecco, G., & Sanguineti, M. (2010). Error bounds for suboptimal solutions to kernel principal component analysis. *Optimization Letters*, *4*(2), 197–210.
Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., & Bengio, Y. (2014). Generative adversarial nets. In *Proceedings of the 28th international conference on neural information processing systems, Vol. 27* (pp. 2672–2680). Curran Associates, Inc..
Higgins, I., Matthey, L., Pal, A., Burgess, C., Glorot, X., Botvinick, M., Mohamed, S., & Lerchner, A. (2017). Beta-VAE: learning basic visual concepts with a constrained variational framework. In *The 5th international conference on learning representations*.
Hinton, G. E. (2012). A practical guide to training restricted Boltzmann machines. In G. Montavon, G. B. Orr, & K.-R. Müller (Eds.), *Lecture notes in computer science, Neural networks: tricks of the trade* (2nd ed.). (pp. 599–619). Berlin, Heidelberg: Springer.
Hinton, G. E., Osindero, S., & Teh, Y.-W. (2006). A fast learning algorithm for deep belief nets. *Neural Computation*, *18*(7), 1527–1554.
Holzinger, A., Langs, G., Denk, H., Zatloukal, K., & Müller, H. (2019). Causability and explainability of artificial intelligence in medicine. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, *9*(4), Article e1312.

Jolliffe, I. T. (1986). *Principal components analysis*. Springer.
Kim, H., & Mnih, A. (2018). Disentangling by factorising. In *Proceedings of the 35th international conference on machine learning, Vol. 80* (pp. 2649–2658). PMLR.
Kingma, D. P., & Ba, J. L. (2015). Adam: A method for stochastic optimization. In *The 3rd international conference on learning representations*.
Kingma, D. P., & Welling, M. (2014). Auto-encoding variational bayes. In *The 2nd international conference on learning representations*.
Kumar, A., Sattigeri, P., & Balakrishnan, A. (2018). Variational inference of disentangled latent concepts from unlabeled observations. In *The 4th International conference on learning representations*.
Lake, B. M., Ullman, T. D., Tenenbaum, J. B., & Gershman, S. J. (2017). Building machines that learn and think like people. *Behavioral and Brain Sciences*, *40*, Article e253.
Le Roux, N., & Bengio, Y. (2008). Representational power of restricted Boltzmann machines and deep belief networks. *Neural Computation*, *20*(6), 1631–1649.
LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *Nature*, *521*(7553), 436–444.
LeCun, Y., Chopra, S., Hadsell, R., Ranzato, M., & Huang, F. (2006). A tutorial on energy-based learning. In G. BakIr, T. Hofmann, B. Schölkopf, A. J. Smola, B. Taskar, & S. N. Vishwanathan (Eds.), *Predicting structured data* (pp. 191–246). MIT Press.
LeCun, Y., Cortes, C., & Burges, C. (2010). MNIST Handwritten digit database. http://yann.lecun.com/exdb/mnist.
LeCun, Y., Huang, F. J., & Bottou, L. (2004). Learning methods for generic object recognition with invariance to pose and lighting. In *Proceedings of the 2004 IEEE computer society conference on computer vision and pattern recognition, Vol. 2* (pp. II–97–104).
Li, J., Li, F., & Todorovic, S. (2019). Efficient Riemannian optimization on the Stiefel manifold via the Cayley transform. In *The 8th international conference on learning representations*.
Locatello, F., Bauer, S., Lucic, M., Raetsch, G., Gelly, S., Schölkopf, B., & Bachem, O. (2019). Challenging common assumptions in the unsupervised learning of disentangled representations. In *Proceedings of the 36th international conference on machine learning, Vol. 97* (pp. 4114–4124). PMLR.
Mercer, J. (1909). Functions of positive and negative type, and their connection with the theory of integral equations. *Philosophical Transactions of the Royal Society, Series A*, *209*(441–458), 415–446.
Mika, S., Schölkopf, B., Smola, A., Müller, K.-R., Scholz, M., & Rätsch, G. (1999). Kernel PCA and de-noising in feature spaces. In *Proceedings of the 1998 conference on advances in neural information processing systems, Vol. 11* (pp. 536–542). MIT Press.
Nocedal, J., & Wright, S. (2006). *Numerical Optimization* (2nd ed.). Springer-Verlag.
Pandey, A., Schreurs, J., & Suykens, J. A. K. (2020). Robust generative restricted kernel machines using weighted conjugate feature duality. In *International conference on machine learning, optimization, and data science*.
Pandey, A., Schreurs, J., & Suykens, J. A. K. (2021). Generative restricted kernel machines: A framework for multi-view generation and disentangled feature learning. *Neural Networks*, *135*, 177–191.
Ranzato, M., Huang, F. J., Boureau, Y.-L., & LeCun, Y. (2007). Unsupervised learning of invariant feature hierarchies with applications to object recognition. In *Proceedings of the 2007 IEEE computer society conference on computer vision and pattern recognition* (pp. 1–8).
Reed, S. E., Zhang, Y., Zhang, Y., & Lee, H. (2015). Deep visual analogy-making. In *Proceedings of the 28th international conference on neural information processing systems, Vol. 28* (pp. 1252–1260). Curran Associates, Inc..
Ridgeway, K., & Mozer, M. C. (2018). Learning deep disentangled embeddings with the f-statistic loss. In *Proceedings of the 32nd international conference on neural information processing systems, Vol. 31* (pp. 185–194). Curran Associates, Inc..
Salakhutdinov, R. (2015). Learning deep generative models. *Annual Review of Statistics and its Application*, *2*, 361–385.
Sarhan, M. H., Eslami, A., Navab, N., & Albarqouni, S. (2019). Learning interpretable disentangled representations using adversarial VAEs. In *Domain adaptation and representation transfer and medical image learning with less labels and imperfect data* (pp. 37–44). Springer.
Schölkopf, B., Janzing, D., Peters, J., Sgouritsa, E., Zhang, K., & Mooij, J. (2012). On causal and anticausal learning. In *Proceedings of the 29th international conference on machine learning* (pp. 1255–1262). Madison, WI, USA: Omnipress.
Schölkopf, B., Mika, S., Burges, C. J., Knirsch, P., Muller, K.-R., Ratsch, G., & Smola, A. J. (1999). Input space versus feature space in kernel-based methods. *IEEE Transactions on Neural Networks*, *10*(5), 1000–1017.
Schölkopf, B., Smola, A., & Müller, K.-R. (1998). Nonlinear component analysis as a Kernel eigenvalue problem. *Neural Computation*, *10*(5), 1299–1319.
Smith, S. T. (1993). *Geometric optimization methods for adaptive filtering* (Ph.D. dissertation), USA: Harvard University.
Smith, S. T. (1994). Optimization techniques on Riemannian manifolds. *Fields Institute Communications*, *3*(3), 113–135.
Suter, R., Miladinovic, D., Schölkopf, B., & Bauer, S. (2019). Robustly disentangled causal mechanisms: validating deep representations for interventional robustness. In *Proceedings of the 36th international conference on machine learning, Vol. 97* (pp. 6056–6065). PMLR.
Suykens, J. A. K. (2017). Deep restricted kernel machines using conjugate feature duality. *Neural Computation*, *29*(8), 2123–2163.
Suykens, J. A. K., Van Gestel, T., De Brabanter, J., De Moor, B., & Vandewalle, J. (2002). *Least squares support vector machines*. World Scientific.
Suykens, J. A. K., Van Gestel, T., Vandewalle, J., & De Moor, B. (2003). A support vector machine formulation to PCA analysis and its Kernel version. *IEEE Transactions on Neural Networks*, *14*(2), 447–450.
Ver Steeg, G., & Galstyan, A. (2015). Maximally informative hierarchical representations of high-dimensional data. In *Proceedings of the 18th international conference on artificial intelligence and statistics, Vol. 38* (pp. 1004–1012). PMLR.
Vincent, P., Larochelle, H., Bengio, Y., & Manzagol, P.-A. (2008). Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th international conference on machine learning* (pp. 1096–1103). Madison, WI, USA: Omnipress.
Vincent, P., Larochelle, H., Lajoie, I., Bengio, Y., & 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*(110), 3371–3408.
Wen, Z., & Yin, W. (2013). A feasible method for optimization with orthogonality constraints. *Mathematical Programming*, *142*(1), 397–434.
Williams, C., & Seeger, M. (2001). Using the Nyström Method to speed up Kernel machines. In *Proceedings of the 2000 conference on advances in neural information processing systems, Vol. 13* (pp. 682–688). MIT Press.
Zeiler, M. D., & Fergus, R. (2014). Visualizing and understanding convolutional networks. In *Proceedings of the 13th european conference on computer vision* (pp. 818–833). Springer.
Zhu, X. (2017). A Riemannian conjugate gradient method for optimization on the Stiefel manifold. *Computational Optimization and Applications*, *67*(1) 73–110.