σ-PCA: a unified neural model for linear and nonlinear principal component analysis

Anonymous authors
Paper under double-blind review

Abstract

Linear principal component analysis (PCA), nonlinear PCA, and linear independent component analysis (ICA) – those are three methods with single-layer autoencoder formulations for learning linear transformations with certain characteristics from data. Linear PCA learns orthogonal transformations that orient axes to maximise variance, but it suffers from a subspace rotational indeterminacy: it fails to find a unique rotation for axes that share the same variance. Both nonlinear PCA and linear ICA reduce the subspace indeterminacy from rotational to permutational by maximising statistical independence under the assumption of unit variance. The main difference between them is that nonlinear PCA only learns rotations while linear ICA learns not just rotations but any linear transformation with unit variance. The relationship between all three can be understood by the singular value decomposition of the linear ICA transformation into a sequence of rotation, scale, rotation. Linear PCA learns the first rotation; nonlinear PCA learns the second. The scale is simply the inverse of the standard deviations. The problem is that, in contrast to linear PCA, conventional nonlinear PCA cannot be used directly on the data to learn the first rotation, the first being special as it reduces dimensionality and orders by variances. In this paper, we have identified the cause, and as a solution we propose σ-PCA: a unified neural model for linear and nonlinear PCA as single-layer autoencoders. With our formulation, nonlinear PCA can learn not just the second, but also the first rotation – by maximising both variance and statistical independence. And so, like linear PCA, nonlinear PCA can now learn a semi-orthogonal transformation that reduces dimensionality and orders by variances, but, unlike linear PCA, it does not suffer from rotational indeterminacy.

1 Introduction

Principal component analysis (PCA) (Pearson, 1901; Hotelling, 1933) needs no introduction. It is classical, ubiquitous, perennial. It is an unsupervised learning method that can be arrived at from three paradigms of representation learning (Bengio et al., 2013): neural, manifold, and probabilistic. This paper is about learning **linear** transformations from data using the first paradigm: the *neural*, in the form of a single-layer autoencoder – a model which encodes the data into a new representation and then decodes that representation back into a reconstruction of the original data.

From the data, PCA learns a linear *(semi-)orthogonal* transformation $\mathbf{W}$ that transforms a given data point $\mathbf{x}$ into a new representation $\mathbf{y} = \mathbf{x}\mathbf{W}$, a representation with reduced dimensionality and minimal loss of information (Diamantaras & Kung, 1996; Jolliffe & Cadima, 2016). This representation has components that are uncorrelated, i.e. have no linear associations, or, in other words, the covariance between two distinct components is zero. When the variances of the components are clearly distinct, PCA has a unique solution, up to sign indeterminacies, that consists in a set of principal eigenvectors (or axes) ordered by their variances. This solution can usually be obtained from the singular value decomposition (SVD) of the data matrix (or the eigendecomposition of the data covariance matrix). Such a solution not only maximises the amount of variance but also minimises the error of reconstruction. And it is the latter – in the form of the squared difference between the data and its reconstruction – that is also the loss function of a single-layer linear autoencoder. However, minimising such a loss will result in a solution that lies in the PCA *subspace* (Baldi &

Hornik, 1989; Bourlard & Kamp, 1988; Oja, 1992b), i.e. a linear combination of the principal eigenvectors, rather than the actual principal eigenvectors. And so, a linear autoencoder performs PCA – but not in its expected axis-aligned form.

What happens when we insert a nonlinearity in the single-layer autoencoder? By natural extension, it must be a form of nonlinear PCA (Xu, 1993; Karhunen & Joutsensalo, 1994; 1995; Karhunen et al., 1998). However, to yield an interesting output, it was observed that the input data needs to be *whitened*, i.e. rotated and scaled into uncorrelated components of *unit* variance. And after the input is whitened, applying nonlinear PCA results in a similar output (Karhunen & Joutsensalo, 1994; 1995; Karhunen et al., 1998; Hyvärinen & Oja, 2000) to linear independent component analysis (ICA) (Jutten & Herault, 1991; Comon, 1994; Bell & Sejnowski, 1997; Hyvarinen, 1999). [1]

Linear ICA, too with a long history in unsupervised learning (Hyvärinen & Oja, 2000), yields, as the name implies, not just uncorrelated but independent components. Linear ICA seeks to find a linear transformation $\mathbf{B}$ – not necessarily orthogonal – such that $\mathbf{y} = \mathbf{x}\mathbf{B}$ has its components as statistically independent as possible. It is based on the idea that $\mathbf{x}$ is an observed linear mixing of the hidden sources $\mathbf{y}$, and it aims to recover these sources solely based on the assumptions that they are non-Gaussian and independent. Without any other assumptions, what is gained in independence is lost in precision: unlike PCA, we cannot determine the variances of the components nor their order (Hyvarinen, 1999; Hyvärinen & Oja, 2000; Hyvärinen, 2015); they are impossible to recover when we have no prior knowledge about how the data was generated. For if $\mathbf{B}$ is a transformation that recovers the sources, then so is $\mathbf{B}\mathbf{\Lambda}$, with $\mathbf{\Lambda}$ an arbitrary diagonal scaling matrix. And so, the best that can be done, without any priors, is to assume the sources have *unit* variance. Linear ICA guarantees identifiability when its assumptions are met (Comon, 1994), meaning that it can recover the true sources up to trivial sign, scale, and permutational indeterminacies.

We thus have three methods for learning linear transformations from data: linear PCA can learn rotations that orient axes into directions that maximise variance, but it suffers from *subspace rotational indeterminacy*: it cannot separate components that have the same variance, i.e. it cannot identify the true principal eigenvectors in subspaces that share the same eigenvalue. Nonlinear PCA (Karhunen & Joutsensalo, 1994) can learn rotations that orient axes into directions that maximise statistical independence, reducing the subspace indeterminacy from rotational to a trivial permutational, but it cannot be applied directly to data without preprocessing – whitening. Linear ICA learns a linear transformation – not just a rotation – that points the axes into directions that maximise statistical independence under the assumption of unit variance. The relationship between all three can be best understood by the SVD of the linear ICA transformation into a sequence of rotation, scale, rotation – stated formally, $\mathbf{W}\mathbf{\Sigma}^{-1}\mathbf{V}$, with $\mathbf{W}$ and $\mathbf{V}$ orthogonal, and $\mathbf{\Sigma}$ diagonal. In one formulation of linear ICA, linear PCA learns the first rotation, $\mathbf{W}$, and nonlinear PCA learns the second, $\mathbf{V}$. The scale, $\mathbf{\Sigma}^{-1}$, is simply the inverse of the standard deviations.

The problem is that, in contrast to linear PCA, conventional nonlinear PCA cannot be used directly on the data to learn the first rotation, the first being special as it can reduce dimensionality and order by variances. The process of first applying the linear PCA rotation and then dividing by the standard deviations is the whitening preprocessing step, the prerequisite for nonlinear PCA. And so, nonlinear PCA, rather than being on an equal footing, is dependent on linear PCA. Conventional nonlinear PCA, by the mere introduction of a nonlinear function, loses the ability for dimensionality reduction and ordering by variances – a jarring disparity between both linear and nonlinear PCA.

In this paper, we have identified the reason why conventional nonlinear PCA has been unable to recover the first rotation, $\mathbf{W}$: it has been missing, in the reconstruction loss, $\mathbf{\Sigma}$. This means that the nonlinear PCA model we want to learn is not $\mathbf{W}$ but $\mathbf{W}\mathbf{\Sigma}^{-1}$. The reason why $\mathbf{\Sigma}$ is needed is that it standardises the components to unit variance before applying the nonlinearity – while still allowing us to compute the variances. Another key observation for nonlinear PCA to work for dimensionality reduction is that it should put an emphasis not on the decoder contribution, but on the encoder contribution – in contrast to conventional linear and nonlinear PCA. In light of the introduction of $\mathbf{\Sigma}$, we call this model σ-PCA. This is so as to distinguish

[1] In the literature, for ICA, the terms linear and nonlinear refer to the transformation, while, for PCA, they refer to the function. Both linear ICA and nonlinear PCA use nonlinear functions, but their resulting transformations are still linear.

it from the conventional PCA model, which merely is a special case with $\mathbf{\Sigma} = \mathbf{I}$. [2] From this model we can derive both linear and nonlinear PCA. With our formulation, nonlinear PCA can now learn not just the second, but also the first rotation, i.e. it can also be applied directly to the data without whitening. And so, like linear PCA, it can learn a semi-orthogonal transformation that reduces dimensionality and orders by variances, but, unlike linear PCA, it does not suffer from rotational indeterminacy. Our main contribution is carving out a place for nonlinear PCA as a method in its own right.

2 Background

Here we look at two things: (1) indeterminacies that arise when learning linear transformations from data, and (2) the conventional neural models for linear and nonlinear PCA as single-layer autoencoders.

2.1 Learning linear transformations from data

Given a set of observations, the goal is to learn a linear transformation that uncovers the true independent sources that generated the observations. It is as if the observations are the result of the sources becoming linearly entangled (Bengio et al., 2013) and what we want to do is disentangle the observations and *identify* the sources (Hyvärinen & Oja, 2000). [3]

Let $\mathbf{Y}_0 \in \mathbb{R}^{n\times k}$ be the data matrix of the true independent sources, with $\mathbf{y}_0 \in \mathbb{R}^{1\times k}$ a row vector representing a source sample, and $\mathbf{X} \in \mathbb{R}^{n\times p}$ be the observations data matrix, with $\mathbf{x} \in \mathbb{R}^{1\times p}$ a row vector representing an observation sample. As the observations can have a higher dimension than the sources, we have $p \geq k$. Without loss of generality, we assume the sources are centred. The observations were generated by transforming the sources with a true transformation $\mathbf{B}_0^{-1} \in \mathbb{R}^{k\times p}$ [4] to obtain the observations $\mathbf{X} = \mathbf{Y}_0\mathbf{B}_0^{-1}$.

To learn from data is to ask the following: without knowing what the true transformation was, to what extent can we recover the sources $\mathbf{Y}_0$ from the observations $\mathbf{X}$? or in other words, to what extent can we recover the left invertible transformation $\mathbf{B}_0$? The problem is then to identify $\mathbf{B}$ such that

$$\mathbf{y} = \mathbf{x}\mathbf{B}, \tag{1}$$

with $\mathbf{y}$ as close as possible to $\mathbf{y}_0$. In the ideal case, we want $\mathbf{B} = \mathbf{B}_0$, allowing us to recover the true sources as $\mathbf{y} = \mathbf{x}\mathbf{B}_0 = \mathbf{y}_0\mathbf{B}_0^{-1}\mathbf{B}_0 = \mathbf{y}_0$. Alas, without prior knowledge of $\mathbf{B}_0$, it is impossible to recover exactly due to noise and due to potentially insufficient data. In addition, the transformation $\mathbf{B}$ can only be recovered up to $\mathbf{B}_0\mathbf{Q}$, where $\mathbf{Q} \in \mathbb{R}^{k\times k}$ has orthogonal columns. This is because $\mathbf{Q}$ cancels out in $\mathbf{B}^{-1}\mathbf{B}$, e.g. $\mathbf{B}^{-1}\mathbf{B} = \mathbf{B}^{-1}\mathbf{Q}^{-1}\mathbf{Q}\mathbf{B} = \mathbf{I}$. The indeterminacy matrix $\mathbf{Q}$ can take the form of a combination of certain special indeterminacies:

$$\text{Scale} \quad \mathbf{\Lambda} = \operatorname{diag}([\lambda_1, ..., \lambda_k]) \in \mathbb{R}_+^{k\times k} \tag{2}$$

$$\text{Sign} \quad \mathbf{I}_\pm = \operatorname{diag}([\pm 1, ..., \pm 1]) \in \mathbb{R}^{k\times k} \tag{3}$$

$$\text{Permutation} \quad \mathbf{P} \in \mathbb{R}^{k\times k} \text{ a permutation matrix} \tag{4}$$

$$\text{Rotation} \quad \mathbf{R}_+ \in \mathbb{R}^{k\times k} \text{ an orthogonal matrix, with } \det(\mathbf{R}_+) = 1. \tag{5}$$

The sign, permutation, and rotation matrices are all special orthogonal matrices that can be used to express any orthogonal[5] indeterminacy matrix $\mathbf{R}$ as

$$\mathbf{R} = \mathbf{I}_\pm\mathbf{P}\mathbf{R}_+\mathbf{P}'\mathbf{I}'_\pm. \tag{6}$$

[2] Arguably, σ-PCA should be the general PCA model, for it is common to divide by the standard deviations when performing the PCA transformation. To avoid scattering σ- every time we mention PCA, we shall simply refer to it as PCA and to the model with $\mathbf{\Sigma} = \mathbf{I}$ as the conventional PCA model.

[3] To disentangle means to recover the sources but not necessarily their distributions, while to identify means to recover not just the sources but also their distributions. Identifiability implies disentanglement but disentanglement does not imply identifiability.

[4] Let $\mathbf{B} \in \mathbb{R}^{p\times k}$; if $k < p$, then $\mathbf{B}^{-1} \in \mathbb{R}^{k\times p}$ is a left inverse of $\mathbf{B}$; if $k > p$, then it is a right inverse.

[5] A reflection is an orthogonal transformation with $\det(\mathbf{R}) = -1$, and it can be obtained as a combination of $\mathbf{I}_\pm\mathbf{P}\mathbf{R}_+\mathbf{P}'\mathbf{I}'_\pm$.

An indeterminacy can be over the entire space or only a subspace, i.e. affecting only a subset of the columns. If it affects a subspace, then it takes the form of a block orthogonal matrix, and we shall denote it with the subscript s, e.g. a subspace orthogonal indeterminacy would be $\mathbf{R}_s$. The worst of the four indeterminacies is the rotational indeterminacy. This is because it does a linear combinations of the sources, i.e. entangles the sources, making them impossible to identify. The other three are trivial in comparison.

We say that a linear transformation is **identifiable** if for any two transformations $\mathbf{A}$ and $\mathbf{B}$ satisfying $\mathbf{y} = \mathbf{x}\mathbf{A}$ and $\mathbf{y} = \mathbf{x}\mathbf{B}$, then $\mathbf{A} = \mathbf{B}\mathbf{\Lambda}\mathbf{P}\mathbf{I}_\pm$ – in other words, there is no rotational indeterminacy, because $\mathbf{I}_\pm\mathbf{P}\mathbf{R}_+\mathbf{P}'\mathbf{I}'_\pm$ reduces to $\mathbf{P}''\mathbf{I}''_\pm$.

2.2 Singular value decomposition (SVD)

Let $\mathbf{B} \in \mathbb{R}^{p\times k}$ represent a linear transformation, then its reduced SVD (Trefethen & Bau III, 1997) is

$$\mathbf{B} = \mathbf{U}\mathbf{S}\mathbf{V}^T, \tag{7}$$

where $\mathbf{U} \in \mathbb{R}^{p\times k}$ a semi-orthogonal matrix with each column a left eigenvector, $\mathbf{V} \in \mathbb{R}^{k\times k}$ an orthogonal matrix with each column a right eigenvector, and $\mathbf{S} \in \mathbb{R}^{k\times k}$ a diagonal matrix of ordered eigenvalues.

Sign indeterminacy If all the eigenvectors have distinct eigenvalues, then this decomposition is, up to sign indeterminacies, unique. This is because we can write

$$\mathbf{B} = \mathbf{U}\mathbf{S}\mathbf{V}^T = \mathbf{U}\mathbf{I}_\pm\mathbf{S}\mathbf{I}_\pm\mathbf{V}^T = \mathbf{U}'\mathbf{S}\mathbf{V}'^T. \tag{8}$$

Subspace rotational indeterminacy If multiple eigenvectors share the same eigenvalue, then, in addition to the sign indeterminacy, there are subspace rotational and permutational indeterminacies. This means that any decomposition with an arbitrary rotation or reflection of those eigenvectors would remain valid.[6] Let $\mathbf{R}_s \in \mathbb{R}^{k\times k}$ be an arbitrary block orthogonal matrix where each block affects the subset of eigenvectors that share the same eigenvalue, then it can commute with $\mathbf{S}$, i.e. $\mathbf{S}\mathbf{R}_s = \mathbf{R}_s\mathbf{S}$ (see Appendix I), so we can write

$$\mathbf{U}\mathbf{S}\mathbf{V}^T = \mathbf{U}\mathbf{I}_\pm\mathbf{R}_s\mathbf{S}\mathbf{R}_s^T\mathbf{I}_\pm^T\mathbf{V}^T = \mathbf{U}'\mathbf{S}\mathbf{V}'^T. \tag{9}$$

2.3 Linear PCA via SVD

Linear PCA can be obtained by the SVD of the data matrix. Let $\mathbf{X} \in \mathbb{R}^{n\times p}$ be the data matrix, where each row vector $\mathbf{x}$ is a data sample, then we can decompose $\mathbf{X}$ into

$$\begin{aligned}
\mathbf{X} &= \mathbf{U}\mathbf{S}\mathbf{V}^T && (10)\\
&= \mathbf{U}\mathbf{I}_\pm\mathbf{R}_s\mathbf{S}\mathbf{R}_s^T\mathbf{I}_\pm^T\mathbf{V}^T && (11)\\
\mathbf{X} &= \sqrt{n}\mathbf{Y}\mathbf{S}\mathbf{W}^T = \mathbf{Y}\mathbf{\Sigma}\mathbf{W}^T && (12)\\
\mathbf{Y} &= \mathbf{X}\mathbf{W}\mathbf{\Sigma}^{-1}, && (13)
\end{aligned}$$

where $\mathbf{Y} = \frac{1}{\sqrt{n}}\mathbf{U}\mathbf{I}_\pm\mathbf{R}_s$ and $\mathbf{W} = \mathbf{V}\mathbf{I}_\pm\mathbf{R}_s$ and $\mathbf{\Sigma} = \frac{1}{\sqrt{n}}\mathbf{S}$.

The columns of $\mathbf{W}$ are the principal eigenvectors, the columns of $\mathbf{Y}$ are the principal components, and $\mathbf{\Sigma}$ is an ordered diagonal matrix of the standard deviations.

Linear PCA has both a sign and subspace rotational indeterminacies. Because of the presence of $\mathbf{R}_s$, there is a subspace rotational indeterminacy, which that linear PCA cannot separate, cannot identify, components that share the same variance.

[6] In theory, this is the case and the rotation is random, but in practice, the eigenvalues are not exactly equal so an SVD solver could still numerically reproduce the same fixed rotational indeterminacy (Hyvärinen et al., 2009).

2.4 Degrees of identifiability

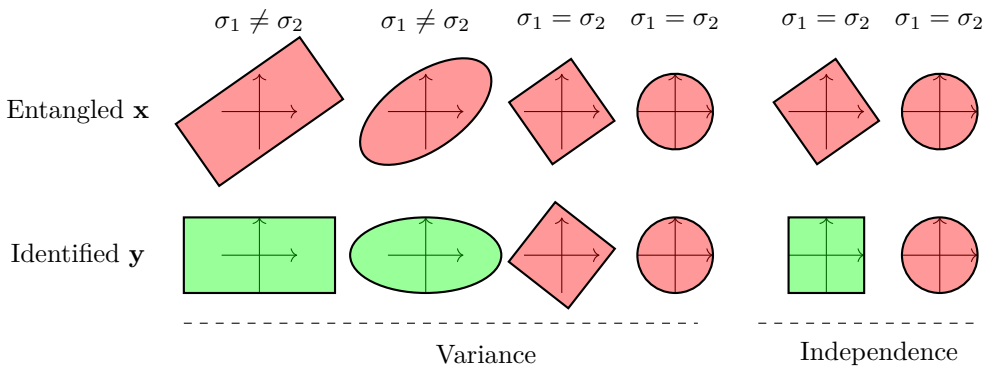


Figure 1: Identifiability of rotations using variance or independence maximisation. The aim is to find a rotation that transforms $\mathbf{x}$ into $\mathbf{y}$, aligning the axes. Red indicates unaligned; green, aligned. For the rectangle and the ellipse, representing distinct variances, we can figure out a rotation, except that we do not know if we should flip up or down, left or right – a sign indeterminacy. For the square and the circle, representing equal variance, we cannot figure out a rotation only from the variance. For the square, however, we can use independence to figure out a rotation, except that we do not know if we should flip and/or permute the vertical and horizontal axes – sign and permutational indeterminacies. For the circle, nothing can be done. There are no favoured directions; even if we apply a rotation, it would remain the same – a rotational indeterminacy. A Gaussian is like a circle.

From the SVD, we know that any linear transformation can be decomposed into a sequence of rotation, scale, rotation. A rotation – an orthogonal transformation – is thus a building block. The first rotation, in particular, can be used for reducing dimensionality, so it can be semi-orthogonal. There are two ways for finding rotations from data: maximising variance and maximising independence. The former fails for components with equal variance, and the latter, although succeeding for components with equal variance, fails for components with Gaussian distributions. The latter fails because of two things: a unit-variance Gaussian has spherical symmetry and maximising statistical independence is akin to maximising non-Gaussianity.

Non-Gaussianity is an essential assumption of linear ICA (Comon, 1994) to guarantee identifiability. With a linear transformation, identifiability theory guarantees that independent sources with unit variance can be recovered if at least all but one of the components have non-Gaussian distributions (Comon, 1994; Hyvärinen & Oja, 2000). Figure 1 illustrates the identifiability of rotations using variance or independence maximisation. Although maximising independence cannot identify sources with Gaussian distributions, maximising variance can still identify the sources for one particular case: when the true transformation is orthogonal and the variances are all distinct (the ellipse in Fig. 1; see Appendix D.3).

Let us now consider how and to what degree we can identify the linear transformation based on whether it is (semi-)orthogonal or non-orthogonal. Although we will be expressing what is identified in terms of the true transformations, in reality, what we end up recovering, due to noise and absence of infinite data, are but approximations of the true transformations.

2.4.1 Semi-orthogonal transformation

Suppose that $\mathbf{B}_0$ is semi-orthogonal, i.e. $\mathbf{B}_0^{-1} = \mathbf{B}_0^T$ with $\mathbf{B}_0^T\mathbf{B}_0 = \mathbf{I}$. Let $\mathbf{E}_0$ denote the matrix of independent orthonormal eigenvectors ordered by their variances, so that true transformation is $\mathbf{B}_0 = \mathbf{E}_0$.

Linear PCA can recover up to sign, and subspace rotational and permutational indeterminacies:

$$\mathbf{B} = \mathbf{E}_0\mathbf{I}_{\pm}\mathbf{R}_s. \tag{14}$$

If all the variances are clearly distinct, then linear PCA can recover $\mathbf{B} = \mathbf{E}_0\mathbf{I}_\pm$, but if all the variances are the same, then $\mathbf{B} = \mathbf{E}_0\mathbf{R}$.

If $\mathbf{B}_0$ had orthogonal but non-orthonormal columns, then there is also a scale indeterminacy, arising because we can introduce a diagonal matrix $\mathbf{\Lambda}$ such that $\mathbf{B}^{-1}\mathbf{B} = \mathbf{B}^T\mathbf{\Lambda}^{-1}\mathbf{\Lambda}\mathbf{B} = \mathbf{B}'^{-1}\mathbf{B}'$.

Although conventional nonlinear PCA can resolve the rotational indeterminacy (Karhunen & Joutsensalo, 1994; 1995; Karhunen et al., 1998), it currently can only works after linearly transforming the inputs into decorrelated components with unit variances – after a preprocessing step called whitening. This can be better understood when we consider the non-orthogonal transformation.

2.4.2 Non-orthogonal transformation

Suppose that $\mathbf{B}_0$ is non-orthogonal. With a non-orthogonal matrix, the columns do not necessarily have unit norm, and so, without any prior information, there is a scale indeterminacy, arising because we can introduce a diagonal matrix $\mathbf{\Lambda}$ such that $\mathbf{B}^{-1}\mathbf{B} = \mathbf{B}^{-1}\mathbf{\Lambda}^{-1}\mathbf{\Lambda}\mathbf{B} = \mathbf{B}'^{-1}\mathbf{B}'$ – there is no way to tell what $\mathbf{\Lambda}$ is. Let us suppose that the true transformation is $\mathbf{B}_0 = \mathbf{U}_0\mathbf{\Lambda}_0$, where $\mathbf{\Lambda}_0$ represents the standard deviations of the independent components, and $\mathbf{U}_0$ represents the linear transformation that results in unit variance components.

Linear ICA allows us to recover $\mathbf{U}_0$ up to sign and permutational indeterminacies. It exploits the fact that once the data has been whitened – rotated and scaled into uncorrelated components of *unit* variance – then any other subsequent rotation will maintain unit variance. What this means is that if $\mathbf{A}$ is a whitening matrix, then $\mathbb{E}((\mathbf{x}\mathbf{A})^T\mathbf{x}\mathbf{A}) = \mathbf{I}$, and if we attempt to apply another rotation $\mathbf{V}$, then

$$\mathbb{E}(\mathbf{V}^T\mathbf{A}^T\mathbf{x}^T\mathbf{x}\mathbf{A}\mathbf{V}) = \mathbf{V}^T\mathbb{E}((\mathbf{x}\mathbf{A})^T\mathbf{x}\mathbf{A})\mathbf{V} = \mathbf{V}^T\mathbf{V} = \mathbf{I}. \tag{15}$$

Therefore $\mathbf{U}$ can be recovered in two stages by first finding a whitening transformation $\mathbf{A}$ and then finding a rotation $\mathbf{V}$ that maximises independence, resulting in $\mathbf{U} = \mathbf{A}\mathbf{V}$. We already have a method for finding a whitening matrix: linear PCA. Linear PCA recovers $\mathbf{A} = \mathbf{E}_0\mathbf{I}_\pm\mathbf{R}_s$. What is left now is to find $\mathbf{V}$. For finding $\mathbf{V}$, FastICA (Hyvarinen, 1999) uses a fast fixed-point algorithm. An alternative method for finding $\mathbf{V}$ is conventional nonlinear PCA (Karhunen & Joutsensalo, 1994). If $\mathbf{U}_0 = \mathbf{E}_0\mathbf{\Sigma}^{-1}\mathbf{V}_0$, then

$$\mathbf{V} = \mathbf{R}_s^T\mathbf{I}_\pm\mathbf{V}_0\mathbf{I}'_\pm\mathbf{P}, \tag{16}$$

resulting in

$$\mathbf{U} = \mathbf{A}\mathbf{V} = \mathbf{E}_0\mathbf{I}_\pm\mathbf{R}_s\mathbf{\Sigma}^{-1}\mathbf{R}_s^T\mathbf{I}_\pm\mathbf{V}_0\mathbf{I}'_\pm\mathbf{P} \tag{17}$$

$$\mathbf{B} = \mathbf{U}\mathbf{\Lambda} = \mathbf{E}_0\mathbf{I}_\pm\mathbf{R}_s\mathbf{\Sigma}^{-1}\mathbf{R}_s^T\mathbf{I}_\pm\mathbf{V}_0\mathbf{I}'_\pm\mathbf{P}\mathbf{\Lambda}. \tag{18}$$

The scale indeterminacy can only be resolved if prior knowledge is available about the norm of the columns of $\mathbf{B}_0$ or $\mathbf{B}_0^{-1}$.

2.5 Conventional neural linear PCA

A single-layer tied linear autoencoder has a solution that lies in the PCA subspace (Baldi & Hornik, 1989; Bourlard & Kamp, 1988; Oja, 1992b). To see why, let $\mathbf{X} \in \mathbb{R}^{n\times p}$ be the *centred* data matrix, with $\mathbf{x}$ a row vector of $\mathbf{X}$, and $\mathbf{W} \in \mathbb{R}^{p\times k}$, with $k \leq p$, then we can write the autoencoder reconstruction loss as

$$\mathcal{L}(\mathbf{W}) = \mathbb{E}(||\mathbf{x} - \mathbf{x}\mathbf{W}\mathbf{W}^T||_2^2) \tag{19}$$

under the orthonormality constraint $\mathbf{W}^T\mathbf{W} = \mathbf{I}$. From this we see that the autoencoder transforms the data with $\mathbf{W}$ (the encoder) into a set of components $\mathbf{y} = \mathbf{x}\mathbf{W}$, and then it reconstructs the original data with $\mathbf{W}^T$ (the decoder) to obtain the reconstruction $\hat{\mathbf{x}} = \mathbf{y}\mathbf{W}^T$. If $k < p$ then it performs dimensionality reduction. Under the orthonormality constraint, the contribution to the gradient originating from the encoder is zero (see Appendix A.2.1). Given that, the linear PCA loss can be equally expressed in a form that results in the subspace learning algorithm (Oja, 1989) weight update:

$$\mathcal{L}(\mathbf{W}) = \mathbb{E}(||\mathbf{x} - \mathbf{x}[\mathbf{W}]_{sg}\mathbf{W}^T||_2^2), \tag{20}$$

where $[\quad]_{sg}$ is the stop gradient operator, an operator that marks the value of the weight as a constant – i.e. the weight is not treated as a variable that contributes to the gradient.

Let $\mathbf{E}_0 \in \mathbb{R}^{p\times k}$ be the matrix whose columns are ordered orthonormal independent eigenvectors, then the minimum of Eq. 19 or 20 is obtained by *any* orthonormal basis of the PCA subspace spanned by the columns of $\mathbf{E}_0$. That is, the optimal $\mathbf{W}$ has the form

$$\mathbf{E}_0\mathbf{I}_{\pm}\mathbf{R}_s\mathbf{R}, \tag{21}$$

where $\mathbf{R} \in \mathbb{R}^{k\times k}$ is an orthogonal indeterminacy over the entire space, resulting in a combined orthogonal indeterminacy $\mathbf{R}_s\mathbf{R}$. The loss is not affected by the presence of $\mathbf{R}$ and $\mathbf{R}_s$, given that $\mathbf{E}_0\mathbf{I}_{\pm}\mathbf{R}_s\mathbf{R}\mathbf{R}^T\mathbf{R}_s^T\mathbf{I}_{\pm}^T\mathbf{E}_0^T = \mathbf{E}_0\mathbf{E}_0^T$. If $\mathbf{R} = \mathbf{I}$, then the linear PCA solution $\mathbf{E}_0\mathbf{I}_{\pm}\mathbf{R}_s$ can be recovered. The reason why $\mathbf{R}$ appears is because the loss is symmetric: no component is favoured over the other. To eliminate $\mathbf{R}$, we simply need to break the symmetry (see Section 3.6 and Appendix A.3).

2.6 Conventional neural nonlinear PCA

A straightforward extension (Xu, 1993; Karhunen & Joutsensalo, 1994; 1995; Karhunen et al., 1998) from linear to nonlinear PCA is to simply introduce in Eq. 19 a nonlinearity h, such as tanh. The reconstruction loss, under the constraint $\mathbf{W}^T\mathbf{W} = \mathbf{I}$, becomes

$$\mathcal{L}(\mathbf{W}) = \mathbb{E}(||\mathbf{x} - h(\mathbf{x}\mathbf{W})\mathbf{W}^T||_2^2). \tag{22}$$

This loss tends to work only when the input is already whitened. This means that, when the input has unit variance, this loss can recover a square orthogonal matrix that maximises statistical independence. It can be shown that it maximises statistical independence as exact relationships can be established between the nonlinear PCA loss and several ICA methods (Karhunen et al., 1998). In particular, these relationships become more apparent when noting that, under the constraint $\mathbf{W}^T\mathbf{W} = \mathbf{I}$, Eq. 22 can also be written as

$$\mathcal{L}(\mathbf{W}) = \mathbb{E}(||\mathbf{y} - h(\mathbf{y})||_2^2), \tag{23}$$

which is similar to the Bussgang criterion used in blind source separation (Lambert, 1996; Haykin, 1996).

Unlike linear PCA, conventional nonlinear PCA cannot recover a useful semi-orthogonal matrix that reduces dimensionality. With the current formulation, the mere introduction of a nonlinear function creates a jarring disparity between both linear and nonlinear PCA. Ideally, we would like to be able to apply nonlinear PCA directly on the input to reduce dimensionality while maximising both variance and statistical independence, allowing us to order by variances and to eliminate the subspace rotational indeterminacy.

3 Unified neural model for linear and nonlinear PCA

Instead of generalising directly from the linear reconstruction loss from Eq. 19, let us take a step back and consider the loss with $\mathbf{B} \in \mathbb{R}^{p\times k}$, a left invertible transformation:

$$\mathbb{E}(||\mathbf{x} - h(\mathbf{x}\mathbf{B})\mathbf{B}^{-1}||_2^2). \tag{24}$$

The conventional nonlinear PCA loss sets $\mathbf{B} = \mathbf{W}$, with $\mathbf{W}$ orthogonal. But if we consider that (1) the conventional nonlinear PCA loss requires unit variance input, and (2) the linear PCA transformation, whether from the SVD decomposition or from the whitening transformation, takes the form $\mathbf{W}\mathbf{\Sigma}^{-1}$, then it seems natural that $\mathbf{B}$ should not be just $\mathbf{W}$ but $\mathbf{W}\mathbf{\Sigma}^{-1}$, with $\mathbf{\Sigma}$ a diagonal matrix of the standard deviations. The transformation $\mathbf{W}\mathbf{\Sigma}^{-1}$ standardises the components to unit variance before applying the nonlinearity h. We thus propose a *general* form of the unified linear and nonlinear PCA model: σ-PCA. Its reconstruction loss is

$$\mathcal{L}(\mathbf{W}, \mathbf{\Sigma}) = \mathbb{E}(||\mathbf{x} - h(\mathbf{x}\mathbf{W}\mathbf{\Sigma}^{-1})\mathbf{\Sigma}\mathbf{W}^T||_2^2), \tag{25}$$

with $\mathbf{W}^T\mathbf{W} = \mathbf{I}$. Formulating it this way allows us to unify linear and nonlinear PCA. Indeed, if h is linear, then we recover the linear PCA loss: $\mathbb{E}(||\mathbf{x} - \mathbf{x}\mathbf{W}\mathbf{\Sigma}^{-1}\mathbf{\Sigma}\mathbf{W}^T||_2^2) = \mathbb{E}(||\mathbf{x} - \mathbf{x}\mathbf{W}\mathbf{W}^T||_2^2)$.

From this *general* form of the loss, we will see that we can derive *specific* losses for linear PCA, nonlinear PCA, and linear ICA in equations 42, 43, and 44, respectively.

3.1 Nonlinear PCA

To derive the new nonlinear PCA loss from the general form in Eq. 25, we need to make one modification: we need to omit, from the weight update, the decoder contribution. To understand why, we need to have a closer look at the individual contributions of both the encoder and the decoder to the update of $\mathbf{W}$. Given that gradient contributions are additive, we will be computing each contribution separately then adding them up. This is as if we untie the weights of the encoder ($\mathbf{W}_e$, $\mathbf{\Sigma}_e$) and the decoder ($\mathbf{W}_d$, $\mathbf{\Sigma}_d$), compute their respective gradients, and then tie them back up, summing the contributions. Without loss of generality, we will look at the stochastic gradient descent update (i.e. batch size of 1, omitting the expectation), and multiply by $\frac{1}{2}$ for convenience, so as to compute the gradients of

$$\frac{1}{2}||\mathbf{x} - h(\mathbf{x}\mathbf{W}_e\mathbf{\Sigma}_e^{-1})\mathbf{\Sigma}_d\mathbf{W}_d^T||_2^2. \tag{26}$$

Let $\mathbf{y} = \mathbf{x}\mathbf{W}_e$, $\mathbf{z} = \mathbf{y}\mathbf{\Sigma}_e^{-1}$, $\hat{\mathbf{x}} = h(\mathbf{z})\mathbf{\Sigma}_d\mathbf{W}_d^T$, and $\hat{\mathbf{y}} = \hat{\mathbf{x}}\mathbf{W}_e$. By computing the gradients, we get

$$\frac{\partial \mathcal{L}}{\partial \mathbf{W}_e} = \mathbf{x}^T(\hat{\mathbf{x}} - \mathbf{x})\mathbf{W}_d\mathbf{\Sigma}_d \odot h'(\mathbf{z})\mathbf{\Sigma}_e^{-1} \tag{27}$$

$$\frac{\partial \mathcal{L}}{\partial \mathbf{W}_d} = (\hat{\mathbf{x}} - \mathbf{x})^T h(\mathbf{z})\mathbf{\Sigma}_d \tag{28}$$

Now that we have traced the origin of each contribution, we can easily compute the gradient when the weights are tied, i.e. $\mathbf{W} = \mathbf{W}_e = \mathbf{W}_d$ and $\mathbf{\Sigma} = \mathbf{\Sigma}_e = \mathbf{\Sigma}_d$, but we can still inspect each contribution separately:

$$\frac{\partial \mathcal{L}}{\partial \mathbf{W}_e} = \mathbf{x}^T(\hat{\mathbf{y}} - \mathbf{y}) \odot h'(\mathbf{z}) \tag{29}$$

$$\frac{\partial \mathcal{L}}{\partial \mathbf{W}_d} = (\hat{\mathbf{x}} - \mathbf{x})^T h(\mathbf{z})\mathbf{\Sigma} \tag{30}$$

$$\frac{\partial \mathcal{L}}{\partial \mathbf{W}} = \frac{\partial \mathcal{L}}{\partial \mathbf{W}_e} + \frac{\partial \mathcal{L}}{\partial \mathbf{W}_d}. \tag{31}$$

From this, we can see the following: the contribution of the decoder, $\frac{\partial \mathcal{L}}{\partial \mathbf{W}_d}$, will be zero when $\hat{\mathbf{x}} \approx \mathbf{x}$, and that of the encoder, $\frac{\partial \mathcal{L}}{\partial \mathbf{W}_e}$, when $\hat{\mathbf{y}} \approx \mathbf{y}$. Thus, unsurprisingly, the decoder puts the emphasis on the input reconstruction, while the encoder puts the emphasis on the latent reconstruction.

In the *linear* case, if $\mathbf{W}^T\mathbf{W} = \mathbf{I}$, the contribution of the encoder is zero. This is because we have $\frac{\partial \mathcal{L}}{\partial \mathbf{W}_e} = \mathbf{x}^T\mathbf{y}(\mathbf{W}^T\mathbf{W} - \mathbf{I}) = \mathbf{0}$, and so it has no effect and can be removed. We can remove it from the gradient by using the stop gradient operator $[\quad]_{sg}$:

$$\mathcal{L}(\mathbf{W}) = ||\mathbf{x} - \mathbf{x}[\mathbf{W}]_{sg}\mathbf{W}^T||_2^2. \tag{32}$$

This results in

$$\frac{\partial \mathcal{L}}{\partial \mathbf{W}} = -\mathbf{x}^T\mathbf{y} + \mathbf{W}\mathbf{y}^T\mathbf{y}, \tag{33}$$

which is simply the subspace learning algorithm (Oja, 1989).

In the conventional extension from linear to nonlinear PCA, the contribution of the encoder was also removed (Karhunen & Joutsensalo, 1994), as it seemed appropriate to simply generalise the subspace learning algorithm (Oja, 1989; Karhunen & Joutsensalo, 1994). But we find that it should be the other way around for the nonlinear case: what matters is not the contribution of the decoder, but the contribution of the encoder.

Indeed, in the *nonlinear* case, $\frac{\partial \mathcal{L}}{\partial \mathbf{W}_e}$ is no longer zero. The problem is that it is overpowered by the decoder contribution, in particular when performing dimensionality reduction (see Appendix F.6). Arguably it would be desirable that the latent of the reconstruction is as close as possible to the latent of the input, i.e. put more emphasis on latent reconstruction. This means that for the nonlinear case we want to remove $\frac{\partial \mathcal{L}}{\partial \mathbf{W}_d}$ from the update and only keep $\frac{\partial \mathcal{L}}{\partial \mathbf{W}_e}$ – which is opposite of the linear case. We thus arrive at what the loss should be in the nonlinear case, along with its corresponding gradient:

$$\mathcal{L}_{\text{NLPCA}}(\mathbf{W}; \mathbf{\Sigma}) = \boxed{\mathbb{E}(||\mathbf{x} - h(\mathbf{x}\mathbf{W}\mathbf{\Sigma}^{-1})\mathbf{\Sigma}[\mathbf{W}^T]_{sg}||_2^2)} \tag{34}$$

$$\frac{\partial \mathcal{L}_{\text{NLPCA}}}{\partial \mathbf{W}} = \mathbb{E}((\mathbf{x}^T h(\mathbf{y}\mathbf{\Sigma}^{-1})\mathbf{\Sigma}\mathbf{W}^T\mathbf{W} - \mathbf{x}^T\mathbf{y}) \odot h'(\mathbf{y}\mathbf{\Sigma}^{-1})). \tag{35}$$

Although this loss naturally enforces orthogonality of the columns of $\mathbf{W}$ by the presence in the gradient of the term $\mathbf{W}^T\mathbf{W}$, it does not maintain their unit norm because $\mathbf{W}$ and $\mathbf{\Sigma}$ can both be scaled by the same amount and $\mathbf{W}\mathbf{\Sigma}^{-1}$ would remain the same. We thus need to use a constraint, or a regulariser (see Appendix C for methods). The constraint, for instance, can be a projective constraint, i.e. normalising each column $\mathbf{w}_i$ of $\mathbf{W}$ to unit norm after each update:

$$\mathbf{w}_i \leftarrow \frac{\mathbf{w}_i}{||\mathbf{w}_i||_2}. \tag{36}$$

Though $\mathbf{\Sigma}$ could be trainable, without any constraints, it could increase without limit. For instance, if $h = \tanh$, we have $\lim_{\sigma\to\infty} \sigma \tanh(z/\sigma) = z$. One option is to add an L_2 regulariser, but $\mathbf{\Sigma}$ need not be trainable, as we already have a natural choice for it: the estimated standard deviation of $\mathbf{y}$. Indeed we can set $\mathbf{\Sigma} = [\text{diag}(\sqrt{\text{var}(\mathbf{y})})]_{sg}$ (see Appendix F.2). We discuss what h needs to be in Section 3.3.

Relation to linear ICA In the gradient update of nonlinear PCA (Eq. 35), we can clearly see two terms, one corresponding to maximising variance and the other to maximising independence – both under the constraint $\mathbf{W}^T\mathbf{W} = \mathbf{I}$. The variance maximising term, $-\mathbf{x}^T\mathbf{y}$, can be obtained in isolation using the loss $-||\mathbf{y}||_2^2$, which is the negative of the variance if taken in expectation. The independence maximising term, $\mathbf{x}^T h(\mathbf{y}\mathbf{\Sigma}^{-1})\mathbf{\Sigma}\mathbf{W}^T\mathbf{W}$, can be seen as a generalisation of the linear ICA update to non-unit variance input; indeed, assuming a square $\mathbf{W}$, $\mathbf{W}^T\mathbf{W} = \mathbf{I}$, and unit variance, the term reduces to $\mathbf{x}^T h(\mathbf{y})$ – which is the standard form of a linear ICA update for pre-whitened input (Hyvärinen & Oja, 1998). Both variance and independence terms can be seen as Hebbian update rules, the former being linear, the latter nonlinear.

As linear ICA is based on maximising non-Gaussianity, a general formulation (Comon, 1994; Hyvärinen & Oja, 1998) of the loss is

$$\mathcal{J}(\mathbf{W}) = |\mathbb{E}(F(\mathbf{x}\mathbf{W})) - \mathbb{E}(F(\mathbf{n}))|, \tag{37}$$

under the constraint $\mathbf{W}^T\mathbf{W} = \mathbf{I}$, where $\mathbf{n} \sim \mathcal{N}(\mathbf{0}, \mathbf{1})$ and F is a smooth nonlinear function. This, however, assumes that the inputs are pre-whitened. For nonlinear PCA, we can make a generalisation of this loss that takes the variances into account:

$$J(\mathbf{W}) = |\mathbb{E}(F(\mathbf{x}\mathbf{W}\mathbf{\Sigma}^{-1})\mathbf{\Sigma}) - \mathbb{E}(F(\mathbf{n})\mathbf{\Sigma}))|. \tag{38}$$

And so, instead of a comparing with a Gaussian with unit variance, we are now comparing with a Gaussian with different variances. As the term involving the Gaussian is constant (Hyvärinen & Oja, 1998), the linear ICA problem reduces to finding all the extrema points of the first term $\mathbb{E}(F(\mathbf{x}\mathbf{W}\mathbf{\Sigma}^{-1})\mathbf{\Sigma})$. Now, in a similar way that conventional nonlinear PCA (Karhunen et al., 1998) can be put in relation to linear ICA, we can put nonlinear PCA in relation to linear ICA by noting that if $\mathbf{W}^T\mathbf{W} = \mathbf{I}$ is enforced, then the nonlinear PCA loss can be expressed as

$$\mathbb{E}(||h(\mathbf{y}\mathbf{\Sigma}^{-1})\mathbf{\Sigma} - [\mathbf{y}]_{sg}||_2^2). \tag{39}$$

If we chose $F(z) = (h(z) - [z]_{sg})^2$, then the nonlinear PCA loss becomes equivalent to the loss in Eq. 38. Given that, it can be shown that the loss will lead the columns of $\mathbf{W}$ to converge to the independent eigenvectors if the independent components z_i satisfy $\mathbb{E}(z_i F'(z_i) - F''(z_i)) \neq 0$, because if the condition is satisfied then the independent eigenvectors are stable stationary points of the loss. This guarantees *local convergence* to the extrema points – the proof of this is given in Hyvärinen & Oja (1998). Global convergence, however, cannot be easily shown, but numerical simulations tend to show that the linear ICA loss tends to converge globally (Hyvärinen & Oja, 1998; Karhunen et al., 1998).

Nonlinear PCA can be put in relation to other methods related to linear ICA, such as blind deconvolution and L_1 sparsity constraints. Both connections are discussed in Appendix F.3 and F.4.

3.2 Relationship between linear PCA, nonlinear PCA, and linear ICA

Linear PCA, nonlinear PCA, and linear ICA seek to find a matrix $\mathbf{B} \in \mathbb{R}^{p \times k}$ such that $\mathbf{y} = \mathbf{x}\mathbf{B}$. Linear PCA puts an emphasis on maximising variance, linear ICA on maximising statistical independence, and nonlinear PCA on both. All three assume that the resulting components of $\mathbf{y}$ have unit variance. This $\mathbf{B}$ is found by being decomposed into $\mathbf{W}\mathbf{\Sigma}^{-1}\mathbf{V}$, with $\mathbf{W} \in \mathbb{R}^{p \times k}$ semi-orthogonal, $\mathbf{\Sigma} \in \mathbb{R}^{k \times k}$ diagonal, and $\mathbf{V} \in \mathbb{R}^{k \times k}$ orthogonal.

Orthogonality is a defining aspect of PCA, so in the case of both linear and nonlinear PCA, we have $\mathbf{V} = \mathbf{I}$, i.e. the restriction is for $\mathbf{B}$ to have orthogonal columns; there is no such restriction with ICA. Simply stated,

$$\text{(Non)linear PCA} \quad \mathbf{W}\mathbf{\Sigma}^{-1} \tag{40}$$

$$\text{Linear ICA} \quad \mathbf{W}\mathbf{\Sigma}^{-1}\mathbf{V}. \tag{41}$$

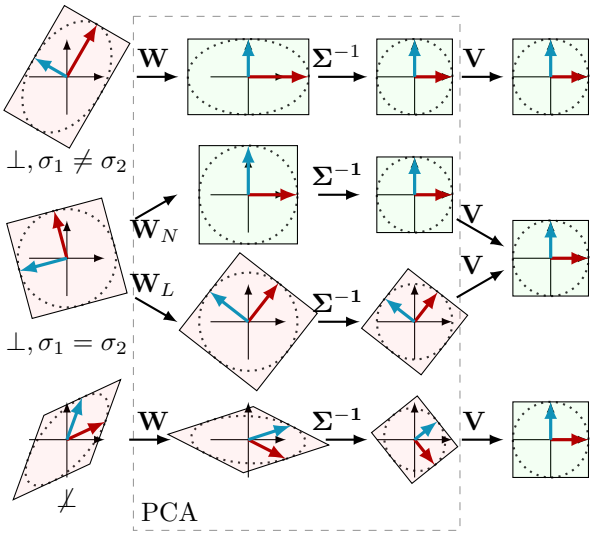


Figure 2: Linear PCA, nonlinear PCA, and linear ICA. Green indicates alignment of axes.

As part of the linear ICA transformation, linear PCA can only find the first rotation, $\mathbf{W}$, while nonlinear PCA can find both rotations, $\mathbf{W}$ and $\mathbf{V}$. Figure 2 summarises the differences between all three methods based on the orthogonality of the overall transformation and the variances of the independent components.

The reconstruction losses of linear PCA (LPCA), nonlinear PCA (NLPCA), and linear ICA (LICA) follow the same pattern of $\mathbb{E}(||\mathbf{x} - h(\mathbf{xB})\mathbf{B}^{-1}||_2^2$, but with a few subtle differences, which we can write as

$$\boxed{\begin{aligned} \mathcal{L}_{\text{LPCA}}(\mathbf{W}) &= \mathbb{E}(||\mathbf{x} - \mathbf{xWW}^T||_2^2), && (42)\\ \mathcal{L}_{\text{NLPCA}}(\mathbf{W};\mathbf{\Sigma}) &= \mathbb{E}(||\mathbf{x} - h(\mathbf{xW\Sigma}^{-1})\mathbf{\Sigma}[\mathbf{W}^T]_{sg}||_2^2), && (43)\\ \mathcal{L}_{\text{LICA}}(\mathbf{W},\mathbf{V};\mathbf{\Sigma}) &= \mathbb{E}(||\mathbf{x}[\mathbf{W\Sigma}^{-1}]_{sg} - h(\mathbf{x}[\mathbf{W\Sigma}^{-1}]_{sg}\mathbf{V})\mathbf{V}^T||_2^2) + \mathcal{L}_{\text{NLPCA}}(\mathbf{W};\mathbf{\Sigma}). && (44)\end{aligned}}$$

The value of $\mathbf{\Sigma}$ is set as $\text{diag}(\boldsymbol{\sigma})$, where $\boldsymbol{\sigma}^2 = \text{var}(\mathbf{xW})$ (see Section 3.4). The nonlinear PCA loss, in particular, requires a unit norm constraint. Unlike the nonlinear PCA and linear ICA losses, the linear PCA loss in Eq. 42 does not allow us to obtain a solution without a random rotational indeterminacy (see Section 2.5), but there are a multiple ways to obtain one by breaking the symmetry. We propose one in Section 3.6 and list many existing methods in Appendix A.3.

Linear PCA emphasises input reconstruction, while nonlinear PCA emphasises latent reconstruction. We can see this clearly from their losses and their respective weight updates (assuming $\mathbf{W}^T\mathbf{W} = \mathbf{I}$) [7]:

$$\begin{aligned} \text{Linear PCA} \quad & \mathbb{E}(||\mathbf{x} - \mathbf{x}[\mathbf{W}]_{sg}\mathbf{W}^T||_2^2), && (45)\\ \text{Nonlinear PCA} \quad & \mathbb{E}(||\mathbf{x} - h(\mathbf{xW\Sigma}^{-1})\mathbf{\Sigma}[\mathbf{W}^T]_{sg}||_2^2) && (46)\end{aligned}$$

$$\begin{aligned} \text{Linear PCA} \quad \Delta\mathbf{W} &\propto \mathbf{x}^T\mathbf{y} - \mathbf{W}\mathbf{y}^T\mathbf{y} &&= \boxed{(\mathbf{x} - \hat{\mathbf{x}})^T\mathbf{y}}, && (47)\\ \text{Nonlinear PCA} \quad \Delta\mathbf{W} &\propto (\mathbf{x}^T\mathbf{y} - \mathbf{x}^T\hat{\mathbf{x}}\mathbf{W}) \odot h'(\mathbf{y}) &&= \boxed{\mathbf{x}^T(\mathbf{y} - \hat{\mathbf{y}})} \odot h'(\mathbf{y}). && (48)\end{aligned}$$

Suppose that $\mathbf{\Sigma}$ is ordered, let $\mathbf{W}_\text{L}$ be the axis-aligned linear PCA solution, $\mathbf{W}_\text{N}$ the nonlinear PCA solution, then we have

$$\begin{aligned} \mathbf{W}_L &= \mathbf{E}_0\mathbf{I}_\pm\mathbf{R}_s && (49)\\ \mathbf{W}_N &= \mathbf{E}_0\mathbf{I}_\pm\mathbf{P}_s && (50)\end{aligned}$$

Linear ICA can be performed in two ways: (1) linear PCA followed by nonlinear PCA:

$$\mathbf{W}_L\mathbf{\Sigma}^{-1}\mathbf{V} = \mathbf{E}_0\mathbf{I}_\pm\mathbf{R}_s\mathbf{\Sigma}^{-1}\mathbf{R}_s^T\mathbf{I}_\pm\mathbf{V}_0\mathbf{I}'_\pm\mathbf{P} = \mathbf{E}_0\mathbf{\Sigma}^{-1}\mathbf{V}_0\mathbf{I}'_\pm\mathbf{P}, \tag{51}$$

and (2) nonlinear PCA followed by nonlinear PCA (with unit-variance), akin to a 2-layer model:

$$\mathbf{W}_N\mathbf{\Sigma}^{-1}\mathbf{V} = \mathbf{E}_0\mathbf{I}_\pm\mathbf{P}_s\mathbf{\Sigma}^{-1}\mathbf{P}_s^T\mathbf{I}_\pm\mathbf{V}_0\mathbf{I}'_\pm\mathbf{P} = \mathbf{E}_0\mathbf{\Sigma}^{-1}\mathbf{V}_0\mathbf{I}'_\pm\mathbf{P}. \tag{52}$$

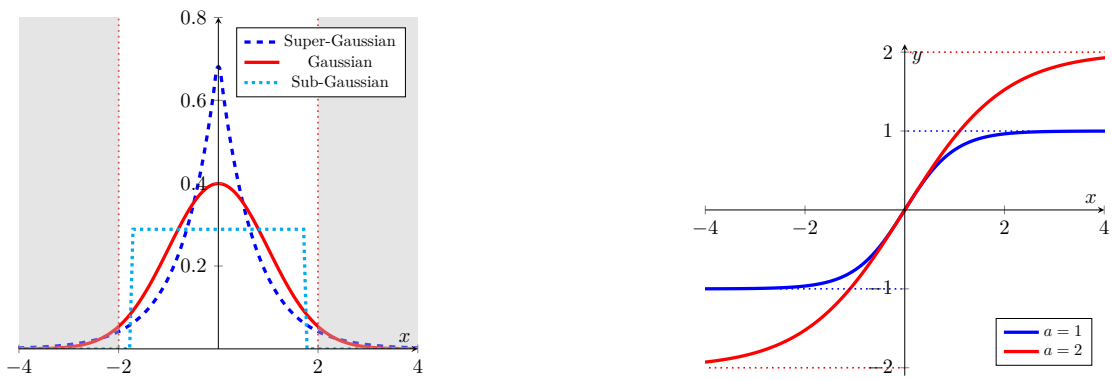
The nonlinear PCA solution $\mathbf{W}_N$ exists within the linear ICA solution space. Therefore, in theory, the nonlinear PCA solution for the first rotation can be recovered using linear ICA (as linear PCA followed by conventional nonlinear PCA). If $\mathbf{V}_0 = \mathbf{I}$, then when we apply linear ICA, we obtain

[7] In the linear case, under the constraint $\mathbf{W}^T\mathbf{W} = \mathbf{I}$, the encoder contribution disappears, making the updates obtained from $||\mathbf{x} - \mathbf{x}[\mathbf{W}]_{sg}\mathbf{W}^T||$ and $||\mathbf{x} - \mathbf{xWW}^T||$ equivalent.

$$\mathbf{W}_L\mathbf{\Sigma}^{-1}\mathbf{V} = \mathbf{E}_0\mathbf{I}_{\pm}\mathbf{R}_s\mathbf{\Sigma}^{-1}\mathbf{R}_s^T\mathbf{I}'_{\pm}\mathbf{P} = \mathbf{E}_0\mathbf{I}_{\pm}\mathbf{\Sigma}^{-1}\mathbf{P}. \tag{53}$$

To reduce the whole space permutation indeterminacy, we can simple renormalise $\mathbf{E}_0\mathbf{I}_{\pm}\mathbf{\Sigma}^{-1}\mathbf{P}$ to unit norm columns and reorder by the estimated variances to obtain $\mathbf{W}_N$.

3.3 Choice of nonlinearity



(a) Probability density functions of three distributions

(b) $a\tanh(x/a)$

Figure 3: Three distributions with unit variance (a): uniform distribution (sub-Gaussian), Gaussian distribution, and Laplace distribution (super-Gaussian). Shaded in grey is any $|x| \geq 2\sigma$. When using tanh without any scale adjustment, we can see that for the super-Gaussian distribution (or for any heavy-tailed distribution), values beyond 2σ might have their reconstruction impaired because of the squashing by tanh (b). A remedy, in this case, would be to use $a\tanh(x/a)$ with $a \geq 1$. For a sub-Gaussian distribution, it is more suited to use $a \leq 1$ as the values are within 2σ .

The choice of nonlinearity generally depends on whether the distribution is sub- or super-Gaussian (Hyvärinen & Oja, 2000; Bingham et al., 2015), but a typical one is tanh. When $\mathbf{\Sigma}$ is estimated from the data, it is beneficial to use $a\tanh(z/a)$ with $a > 0$ instead of $\tanh(z)$. The choice of a will generally depend on whether the distribution is sub- or super-Gaussian, with $a \leq 1$ for a sub-Gaussian and $a \geq 1$ for a super-Gaussian. The reason why a super-Gaussian requires $a \geq 1$ is that it is more likely to be tail-heavy, so, given that z is standardised, there will be more values greater 2σ, which would result in tanh squashing them close 1 (see Fig. 3), making it unlikely to satisfy $\mathbb{E}(a\tanh(z/a)) \approx \mathbb{E}(z)$ for the stationary point (see Appendix F.1). A trainable $\mathbf{\Sigma}$ is more likely to work with a super-Gaussian than a sub-Gaussian. There are also other options, such as setting a to be trainable, using $\max(-a, \min(z, a))$ as an approximation of tanh, and using an asymmetric function such as $\tanh(z)/\sqrt{\mathrm{var}(tanh(z))}$ – we briefly explored these during experiments in Appendix G.

3.4 Mean and variance of data

Mean centring is an important part of PCA (Diamantaras & Kung, 1996; Hyvärinen & Oja, 2000; Jolliffe & Cadima, 2016). So far, we have assumed that the data $\mathbf{x}$ is centred; if it is not, then it can easily be centred as a pre-processing step or during training. If the latter, we can explicitly write the loss function in Eq. 34 as

$$\mathcal{L}(\mathbf{W}) = \mathbb{E}(||\mathbf{x} - \boldsymbol{\mu}_x - (h(\frac{(\mathbf{x} - \boldsymbol{\mu}_x)\mathbf{W}}{\boldsymbol{\sigma}})\boldsymbol{\sigma})[\mathbf{W}^T]_{sg}||_2^2), \tag{54}$$

where $\boldsymbol{\mu}_x = \mathbb{E}(\mathbf{x})$ and $\boldsymbol{\sigma}^2 = \text{var}(\mathbf{xW})$. The mean and variance can be estimated on a batch of data, or, as is done with batch normalisation (Ioffe & Szegedy, 2015), estimated using exponential moving averages (Appendix J). See Appendix F.8 for a non-centred variant.

3.5 Ordering of the components

After minimising the nonlinear PCA loss, we can order the components based on $\mathbf{\Sigma}$. It is also possible to automatically induce the order based on index position. One such method is to introduce the projective deflation-like term $P(\mathbf{W}) = I - \triangleleft(\mathbf{W}^T[\mathbf{W}]_{sg})$ into Eq. 34 as follows

$$\mathcal{L}(\mathbf{W}) = \mathbb{E}(||\mathbf{x} - h(\mathbf{xW}P(\mathbf{W})\mathbf{\Sigma}^{-1})\mathbf{\Sigma}[\mathbf{W}^T]_{sg}||_2^2), \tag{55}$$

where $\triangleleft$ is lower triangular without the diagonal. See Appendix F.7 for details and other methods.

3.6 An asymmetric linear PCA loss

In Eq. 47 and 48, we see that $\mathbf{y}^T\mathbf{y}$ is symmetric, while $\mathbf{x}^T\hat{\mathbf{x}}$ is not. This is why we obtain axis-aligned components with the latter, but not the former. To obtain axis-aligned components with linear PCA, we simply need to break the symmetry. The generalised Hebbian algorithm (Sanger, 1989) breaks the symmetry by replacing $\mathbf{y}^T\mathbf{y}$ with its lower triangular $\triangleleft(\mathbf{y}^T\mathbf{y})$. Many other methods exist (see Appendix A.3). Here, we propose two unit-norm-preserving methods for breaking the symmetry. The first consists in simply introducing $\mathbf{\Lambda} = \text{diag}(\lambda_1, ..., \lambda_k)$, a linearly-spaced diagonal matrix such that $1 \geq \lambda_1 > ... > \lambda_k > 0$ to obtain

$$\Delta\mathbf{W} \propto \mathbf{x}^T\mathbf{y} - \mathbf{W}(\mathbf{y}\mathbf{\Lambda}^{\frac{1}{2}})^T\mathbf{y}\mathbf{\Lambda}^{-\frac{1}{2}}. \tag{56}$$

This is simply a unit-norm-preserving variant of the weighted subspace algorithm (Oja, 1992a; Oja et al., 1992). The second that we propose as a loss function is

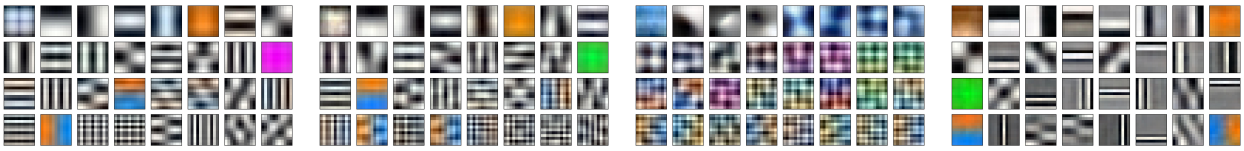
$$\mathcal{L}(\mathbf{W}) = \mathbb{E}(||\mathbf{x}[\mathbf{W}]_{sg}\mathbf{W}^T - \mathbf{x}||_2^2 + ||\mathbf{W}[\hat{\mathbf{\Sigma}}]_{sg}\mathbf{\Lambda}^{\frac{1}{2}}||_2^2 - ||\mathbf{W}[\hat{\mathbf{\Sigma}}]_{sg}||_2^2 - ||\mathbf{xW}\mathbf{\Lambda}^{\frac{1}{2}}||_2^2 + ||\mathbf{xW}||_2^2), \tag{57}$$

where $\hat{\mathbf{\Sigma}}$ is a diagonal matrix of the estimated batch standard deviations. This loss combines reconstruction, weighted regularisation, and weighted variance maximisation. See Appendix A.3.2 for derivation.

4 Experiments

We applied the neural PCA models on image patches and time signals (see Appendix G for additional experiments); we optimised using gradient descent (see Appendix H for training details).

Image patches We extracted 11×11px overlapping patches, with zero padding, from a random subset of 500 images from CIFAR-10 (Krizhevsky et al., 2009), resulting in a total of 512K patches. We estimated $\mathbf{\Sigma}$ from the data and we used $a\tanh(z/a)$ with $a = 4$, though $a \in [3, 14]$ also worked well (see Appendix Fig. 11). We used $a > 1$ as a lot of natural images tend to have components super-Gaussian distributions with similar variances (Hyvärinen et al., 2009). We applied linear PCA, nonlinear PCA, and FastICA (Hyvarinen, 1999), and summarised the results in Fig. 4. We see that the PCA filters (Fig. 4a) appear like blurred combinations of the filters found by nonlinear PCA (Fig. 4d). This stems from the rotational indeterminacy of linear PCA for components with similar variances. We see that $a = 1$ does not work (Fig. 4d) in comparison to $a = 4$(Fig. 4d). In fact we can have a smooth transition of disentanglement, passing through linear PCA filters up to nonlinear PCA filters (see Appendix Fig. 11). We also see that conventional nonlinear PCA (Fig. 4g) failed to recover any meaningful filters. This is equally the case when the contribution of the decoder is not removed (Fig. 4f), highlighting that it is in fact detrimental.

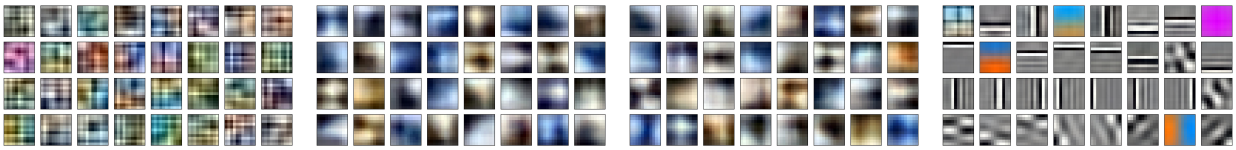


(a) Linear PCA using SVD

(b) Linear PCA using unit-norm-preserving loss (Eq. 57)

(c) Nonlinear PCA with $h(x) = \tanh(x)$ (Eq. 54)

(d) Nonlinear PCA with $h(x) = 4\tanh(x/4)$ (Eq. 54)



(e) Linear PCA using symmetric loss (Eq. 25)

(f) Nonlinear PCA (Eq. 25) without stop gradient on the decoder

(g) Conventional nonlinear PCA (Eq. 22) without whitening

(h) FastICA with filters ordered by norm (inversely proportional to the standard deviations)

Figure 4: A set of 32 11x11px filters obtained on patches from the CIFAR-10 dataset. We obtained similar filters with the proposed unit-norm-preserving linear PCA loss (b) as the ones obtained via SVD (a). The filters obtained by nonlinear PCA (c,d) seem to have further separated the mixed filters from linear PCA. In particular, this is obvious by looking at the vertical and horizontal line filters at different positions that have been unmixed with nonlinear PCA. We see that setting $a = 1$ is not enough to separate (c), requiring a large a such as $a = 4$ (d). We obtained meaningless filters in the PCA subspace with the symmetric linear PCA loss (e). We also obtained meaningless filters when the contribution of the decoder was included (f). This is similarly the case with conventional nonlinear PCA without whitening (g). FastICA (h) relaxes the orthogonality assumption of the overall transformation, so we see filters that are not necessarily orthogonal; however, there is some overlap with nonlinear PCA filters.

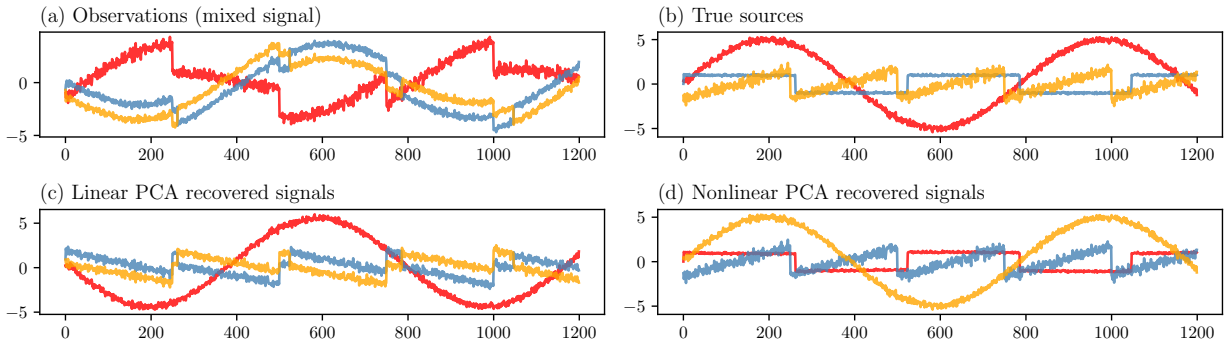


Figure 5: Three signals (sinusoidal, square, and sawtooth) that were mixed with an *orthogonal* mixing matrix. Linear PCA separated the sinusoidal signal as it had a distinct variance, but did not separate the square and the sawtooth signals as they had the same variance. Nonlinear PCA separated the signals and recovered their variances.

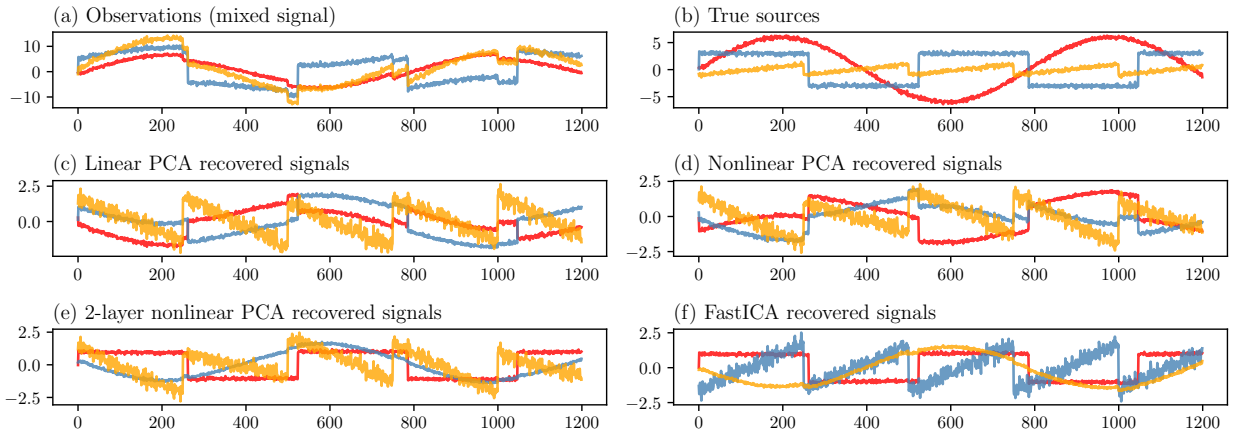


Figure 6: Three signals (sinusoidal, square, and sawtooth) that were mixed with a *non-orthogonal* mixing matrix. As expected, both linear (c) and nonlinear PCA (d) were not able to recover the signals. Only FastICA (f) and the 2-layer nonlinear PCA model (e) recovered the signals, but without their variances.

Time signals We adapted a classic linear ICA example of signal separation (Comon, 1994) from scikit-learn (Pedregosa et al., 2011). First, we considered an orthogonal transformation. We generated three noisy signals: sinusoidal, square, and sawtooth. The last two had the same variance, and we mixed all three using a random *orthogonal* mixing matrix. We estimated the variance from data and used $a \tanh(z/a)$ with $a = 0.8$. Figure 5 shows the result of applying linear and nonlinear PCA. As expected, linear PCA separated the sinusoidal signal, but not the square and sawtooth signals because they had the same variance. Nonlinear PCA recovered the three signals and their variances, up to sign indeterminacies. Second, we considered a non-orthogonal transformation. We mixed the three signals, this time with distinct variances, using a *non-orthogonal* mixing matrix. We summarised the results in Fig. 6. As expected, both linear and nonlinear PCA did not recover the signals – they simple recovered the closest orthogonal transformation in terms of reconstruction loss. Both FastICA (linear PCA followed by the fast fixed-point algorithm) and the two-layer nonlinear PCA model (Eq. 44) recovered the signals, but without their variances due to the scale indeterminacy.

5 Related work

Hyvärinen (2015) previously proposed a unified model of linear PCA and linear ICA as a theoretical framework from the probabilistic paradigm, where the variances of the components are modelled as separate parameters but are eventually integrated out. Our key idea is the same: we model the variances as separate parameters, except that we do not integrate them out, and we flesh out the key role they play, from the neural paradigm, for nonlinear PCA.

Neural PCA was started with the seminal work of Oja (1982) which showed that the first principal component can be extracted with a Hebbian (Hebb, 1949) learning rule. This spurred a lot of interest in PCA neural networks in the 80s and 90s, resulting in different variants and extensions for the extraction of multiple components (Foldiak, 1989; Oja, 1989; Rubner & Schulten, 1990; Kung & Diamantaras, 1990; Oja, 1992b; Bourlard & Kamp, 1988; Baldi & Hornik, 1989; Oja, 1992a; 1995) (see Diamantaras & Kung (1996) for an overview), some of which extracted the axis-aligned solution while others the subspace solution. A linear autoencoder with a mean squared error reconstruction loss was shown to extract the subspace solution (Bourlard & Kamp, 1988; Baldi & Hornik, 1989).

Unlike in the linear case, the term nonlinear PCA has been applied more broadly (Hyvärinen & Oja, 2000) to single-layer (Xu, 1993; Karhunen & Joutsensalo, 1994; Oja, 1995), multi-layer (Kramer, 1991; Oja, 1991; Scholz & Vigário, 2002), and kernel-based (Schölkopf et al., 1998) variants. As a single-layer autoencoder, the main emphasis has been its close connection to linear ICA, especially with whitened data (Karhunen et al., 1997; 1998; Hyvärinen & Oja, 2000). Linear ICA is a more powerful extension to PCA in that it seeks to find components that are non-Gaussian and statistically independent, with the overall transformation not necessarily orthogonal. Many algorithms have been proposed (see Hyvärinen & Oja (2000); Bingham et al. (2015) for an exhaustive overview), with the most popular being FastICA (Hyvarinen, 1999). Reconstruction ICA (RICA) (Le et al., 2011) is an autoencoder formulation which combines a linear reconstruction loss with L_1 sparsity; although it was proposed as a method for learning overcomplete ICA features, we show that RICA, in fact, is performing nonlinear PCA, except that it does not preserve unit norm (see Appendix F.4).

6 Discussion and conclusion

We have proposed σ-PCA, a unified neural model for linear and nonlinear PCA. This model allows nonlinear PCA to be on equal footing with linear PCA: it can learn a semi-orthogonal transformation that reduces dimensionality and orders by variances. But, unlike linear PCA, nonlinear PCA does not suffer from subspace rotational indeterminacy: it can identify non-Gaussian components that have the same variance.

Previously, it was only possible to apply conventional nonlinear PCA after whitening the input. This meant that when we consider the linear ICA transformation as a sequence of rotation, scale, rotation, then conventional nonlinear PCA could only be applied to learn the second rotation on unit variance input, but was not able to learn the first rotation. With our model, nonlinear PCA can now be applied to learn not just the second but also the first rotation. Although the nonlinear PCA solution implicitly exists in the linear ICA solution space, it is not necessarily explicitly recovered, because linear ICA seeks to find an overall transformation of the form $\mathbf{W}\mathbf{\Sigma}^{-1}\mathbf{V}$ that maximises independence. It is thus over-parametrised when the aim is to find an orthogonal transformation of the form $\mathbf{W}\mathbf{\Sigma}^{-1}$, and so linear ICA would be more susceptible to over-fitting, resulting in an overall non-orthogonal transformation.

An interesting aspect that emerged is that there is an elegant mirroring between linear and nonlinear PCA: the former puts an emphasis on input reconstruction, while the latter on latent reconstruction; the former relies on the decoder contribution, while the latter relies on the encoder contribution. With the σ-PCA model, we can have a smooth transition between both, rather than a harsh dichotomy. The advantage of nonlinear PCA is that it can be applied wherever linear PCA tends to be applied to learn orthogonal transformations – with the added benefit of identifiability.

References

Pierre Baldi and Kurt Hornik. Neural networks and principal component analysis: Learning from examples without local minima. *Neural networks*, 2(1):53–58, 1989.

David Barber. *Bayesian reasoning and machine learning.* Cambridge University Press, 2012.

Anthony J Bell and Terrence J Sejnowski. The "independent components" of natural scenes are edge filters. *Vision research*, 37(23):3327–3338, 1997.

Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1798–1828, 2013.

Ella Bingham, Samuel Kaski, Jorma Laaksonen, and Jouko Lampinen. *Advances in independent component analysis and learning machines.* Academic Press, 2015.

Hervé Bourlard and Yves Kamp. Auto-association by multilayer perceptrons and singular value decomposition. *Biological cybernetics*, 59(4):291–294, 1988.

J-F Cardoso and Beate H Laheld. Equivariant adaptive source separation. *IEEE Transactions on signal processing*, 44(12):3017–3030, 1996.

Pierre Comon. Independent component analysis, a new concept? *Signal processing*, 36(3):287–314, 1994.

Konstantinos I Diamantaras and Sun Yuan Kung. *Principal component neural networks: theory and applications.* John Wiley & Sons, Inc., 1996.

Yuguang Fang, Kenneth A Loparo, and Xiangbo Feng. Inequalities for the trace of matrix product. *IEEE Transactions on Automatic Control*, 39(12):2489–2490, 1994.

Peter Foldiak. Adaptive network for optimal linear feature extraction. In *Proceedings of IEEE/INNS Int. Joint. Conf. Neural Networks*, volume 1, pp. 401–405, 1989.

Harry Horace Harman. *Modern factor analysis.* University of Chicago press, 1976.

Simon Haykin. *Adaptive Filter Theory (3rd Ed.).* Prentice-Hall, Inc., USA, 1996.

Donald Olding Hebb. The organization of behaviour. *A neurophysiological theory*, 1949.

Harold Hotelling. Analysis of a complex of statistical variables into principal components. *Journal of educational psychology*, 24(6):417, 1933.

A. Hyvärinen. Survey on independent component analysis. *Neural Computing Surveys*, 2:94–128, 1999.

Aapo Hyvarinen. Fast and robust fixed-point algorithms for independent component analysis. *IEEE transactions on Neural Networks*, 10(3):626–634, 1999.

Aapo Hyvärinen. A unified probabilistic model for independent and principal component analysis. In *Advances in Independent Component Analysis and Learning Machines*, pp. 75–82. Elsevier, 2015.

Aapo Hyvärinen and Erkki Oja. Independent component analysis by general nonlinear hebbian-like learning rules. *signal processing*, 64(3):301–313, 1998.

Aapo Hyvärinen and Erkki Oja. Independent component analysis: algorithms and applications. *Neural networks*, 13(4-5):411–430, 2000.

Aapo Hyvärinen, Jarmo Hurri, and Patrick O Hoyer. *Natural image statistics: A probabilistic approach to early computational vision.*, volume 39. Springer Science & Business Media, 2009.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning*, pp. 448–456, 2015.

Ian T Jolliffe and Jorge Cadima. Principal component analysis: a review and recent developments. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 374(2065):20150202, 2016.

Christian Jutten and Jeanny Herault. Blind separation of sources, part i: An adaptive algorithm based on neuromimetic architecture. *Signal processing*, 24(1):1–10, 1991.

Juha Karhunen and Jyrki Joutsensalo. Representation and separation of signals using nonlinear pca type learning. *Neural networks*, 7(1):113–127, 1994.

Juha Karhunen and Jyrki Joutsensalo. Generalizations of principal component analysis, optimization problems, and neural networks. *Neural Networks*, 8(4):549–562, 1995.

Juha Karhunen, Erkki Oja, Liuyue Wang, Ricardo Vigario, and Jyrki Joutsensalo. A class of neural networks for independent component analysis. *IEEE Transactions on neural networks*, 8(3):486–504, 1997.

Juha Karhunen, Petteri Pajunen, and Erkki Oja. The nonlinear pca criterion in blind source separation: Relations with other approaches. *Neurocomputing*, 22(1-3):5–20, 1998.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

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

Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

Sun-Yuan Kung and Kostas I Diamantaras. A neural network learning algorithm for adaptive principal component extraction (apex). In *International Conference on Acoustics, Speech, and Signal Processing*, pp. 861–864. IEEE, 1990.

Russell Howard Lambert. *Multichannel blind deconvolution: FIR matrix algebra and separation of multipath mixtures*. University of Southern California, 1996.

Quoc Le, Alexandre Karpenko, Jiquan Ngiam, and Andrew Ng. Ica with reconstruction cost for efficient overcomplete feature learning. *Advances in neural information processing systems*, 24, 2011.

E. Oja. *Subspace Methods of Pattern Recognition*. Electronic & electrical engineering research studies. Research Studies Press, 1983.

E Oja. Principal component analysis by homogeneous neural networks, part i: The weighted subspace criterion. *IEICE Trans. on Inf. and Syst.*, 75(3):366–375, 1992a.

Erkki Oja. Simplified neuron model as a principal component analyzer. *Journal of mathematical biology*, 15 (3):267–273, 1982.

Erkki Oja. Neural networks, principal components, and subspaces. *International journal of neural systems*, 1 (01):61–68, 1989.

Erkki Oja. Data compression, feature extraction, and autoassociation in feedforward neural networks. *Artificial neural networks*, 1991.

Erkki Oja. Principal components, minor components, and linear neural networks. *Neural networks*, 5(6): 927–935, 1992b.

Erkki Oja. Pca, ica, and nonlinear hebbian learning. In *Proc. Int. Conf. on Artificial Neural Networks (ICANN'95)*, pp. 89–94, 1995.

Erkki Oja, Hidemitsu Ogawa, and Jaroonsakdi Wangviwattana. Principal component analysis by homogeneous neural networks, part ii: Analysis and extensions of the learning algorithms. *IEICE Transactions on Information and Systems*, 75(3):376–382, 1992.

Bruno A Olshausen and David J Field. Emergence of simple-cell receptive field properties by learning a sparse code for natural images. *Nature*, 381(6583):607–609, 1996.

Karl Pearson. Liii. on lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin philosophical magazine and journal of science*, 2(11):559–572, 1901.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

Oren Rippel, Michael Gelbart, and Ryan Adams. Learning ordered representations with nested dropout. In *International Conference on Machine Learning*, pp. 1746–1754. PMLR, 2014.

Jeanette Rubner and Klaus Schulten. Development of feature detectors by self-organization: A network model. *Biological Cybernetics*, 62(3):193–199, 1990.

Tim Salimans and Durk P Kingma. Weight normalization: A simple reparameterization to accelerate training of deep neural networks. *Advances in neural information processing systems*, 29, 2016.

Terence D Sanger. Optimal unsupervised learning in a single-layer linear feedforward neural network. *Neural networks*, 2(6):459–473, 1989.

Erhard Schmidt. Zur theorie der linearen und nichtlinearen integralgleichungen. *Mathematische Annalen*, 63 (4):433–476, 1907.

Bernhard Schölkopf, Alexander Smola, and Klaus-Robert Müller. Nonlinear component analysis as a kernel eigenvalue problem. *Neural computation*, 10(5):1299–1319, 1998.

Matthias Scholz and Ricardo Vigário. Nonlinear pca: a new hierarchical approach. In *Esann*, pp. 439–444, 2002.

Michael E Tipping and Christopher M Bishop. Probabilistic principal component analysis. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 61(3):611–622, 1999.

Lloyd N Trefethen and David Bau III. *Numerical Linear Algebra*, volume 50. SIAM, 1997.

Liuyue Wang and Juha Karhunen. A unified neural bigradient algorithm for robust pca and mca. *International journal of neural systems*, 7(01):53–67, 1996.

Liuyue Wang, Juha Karhunen, and Erkki Oja. A bigradient optimization approach for robust pca, mca, and source separation. In *Proceedings of ICNN'95-International Conference on Neural Networks*, volume 4, pp. 1684–1689. IEEE, 1995.

R.J. Williams and San Diego. Institute for Cognitive Science University of California. *Feature Discovery Through Error-correction Learning*. ICS report. Institute for Cognitive Science, University of California, San Diego, 1985.

Lei Xu. Least mean square error reconstruction principle for self-organizing neural-nets. *Neural networks*, 6 (5):627–648, 1993.

Appendix

A Linear PCA

In this section we go through different methods related to PCA and highlight their loss functions and weight updates where relevant. There are a variety of other neural methods for PCA; see (Diamantaras & Kung, 1996) for an in-depth review.

A.1 PCA as eigendecomposition of covariance matrix or SVD of data matrix

Let $\mathbf{X} \in \mathbb{R}^{n \times p}$ be the *centred* data matrix and let $\mathbf{C} = \frac{1}{n}\mathbf{X}^T\mathbf{X}$ be the symmetric covariance matrix, then we can write the SVD of $\mathbf{X}$ as

$$\mathbf{X} = \mathbf{U}\mathbf{S}\mathbf{V}^T, \tag{58}$$

where $\mathbf{U} \in \mathbb{R}^{n \times p}$ is a semi-orthogonal matrix, $\mathbf{S} \in \mathbb{R}^{p \times p}$ is a diagonal matrix of singular values, and $\mathbf{V} \in \mathbb{R}^{p \times p}$ is an orthogonal matrix; and we can write the eigendecomposition of $\mathbf{C}$ as

$$\mathbf{C} = \mathbf{E}\mathbf{\Lambda}\mathbf{E}^T, \tag{59}$$

where $\mathbf{E} \in \mathbb{R}^{p \times p}$ is an orthogonal matrix of eigenvectors corresponding to the principal axes, and $\mathbf{\Lambda} \in \mathbb{R}^{p \times p}$ is a diagonal matrix corresponding to the variances.

We can relate both by noting that we have

$$\mathbf{C} = \frac{1}{n}\mathbf{X}^T\mathbf{X} = \frac{1}{n}\mathbf{V}\mathbf{S}^T\mathbf{U}^T\mathbf{U}\mathbf{S}\mathbf{V}^T \tag{60}$$

$$\mathbf{E}\mathbf{\Lambda}\mathbf{E}^T = \mathbf{V}\frac{1}{n}\mathbf{S}^2\mathbf{V}^T. \tag{61}$$

From this we can see that $\mathbf{E} = \mathbf{V}$ and $\mathbf{\Lambda} = \frac{1}{n}\mathbf{S}^2 = \text{diag}(\boldsymbol{\sigma}^2)$.

A.2 PCA subspace solutions

A.2.1 Linear autoencoder

Let $\mathbf{X}$ be the centred data matrix and $\mathbf{x}$ a row of $\mathbf{X}$, then we can write the reconstruction loss of a linear autoencoder as

$$\mathcal{L}(\mathbf{W}_e, \mathbf{W}_d) = \frac{1}{2}\mathbb{E}(||\mathbf{x}\mathbf{W}_e\mathbf{W}_d^T - \mathbf{x}||_2^2), \tag{62}$$

where $\mathbf{W}_e \in \mathbb{R}^{p \times k}$ is the encoder and $\mathbf{W}_d^T\mathbb{R}^{p \times k}$ is the decoder. Without loss of generality, we omit the expectation, the loss yields the following gradients:

$$\frac{\partial \mathcal{L}}{\partial \mathbf{W}_e} = \mathbf{x}^T(\mathbf{x}\mathbf{W}_e\mathbf{W}_d^T - \mathbf{x})\mathbf{W}_d \tag{63}$$

$$\frac{\partial \mathcal{L}}{\partial \mathbf{W}_d} = (\mathbf{x}\mathbf{W}_e\mathbf{W}_d^T - \mathbf{x})^T\mathbf{x}\mathbf{W}_e. \tag{64}$$

Let us consider the tied weights case, i.e. $\mathbf{W}_e = \mathbf{W}_d = \mathbf{W}$, but keep the contributions separate. Let $\mathbf{y} = \mathbf{x}\mathbf{W}$, we can write

$$\frac{\partial \mathcal{L}}{\partial \mathbf{W}_e} = \mathbf{x}^T(\mathbf{x}\mathbf{W}\mathbf{W}^T - \mathbf{x})\mathbf{W} \tag{65}$$
$$= \mathbf{x}^T\mathbf{x}\mathbf{W}(\mathbf{W}^T\mathbf{W} - \mathbf{I}) \tag{66}$$
$$= \mathbf{x}^T\mathbf{y}(\mathbf{W}^T\mathbf{W} - \mathbf{I}) \tag{67}$$
$$\frac{\partial \mathcal{L}}{\partial \mathbf{W}_d} = (\mathbf{x}\mathbf{W}\mathbf{W}^T - \mathbf{x})^T\mathbf{x}\mathbf{W} \tag{68}$$
$$= (\mathbf{x}\mathbf{W}\mathbf{W}^T)^T\mathbf{x}\mathbf{W} - \mathbf{x}^T\mathbf{x}\mathbf{W} \tag{69}$$
$$= \mathbf{W}\mathbf{W}^T\mathbf{x}^T\mathbf{x}\mathbf{W} - \mathbf{x}^T\mathbf{x}\mathbf{W} \tag{70}$$
$$= (\mathbf{W}\mathbf{W}^T - \mathbf{I})\mathbf{x}^T\mathbf{x}\mathbf{W} \tag{71}$$
$$= (\mathbf{W}\mathbf{W}^T - \mathbf{I})\mathbf{x}^T\mathbf{y} \tag{72}$$

We can note that the term $-\mathbf{x}^T\mathbf{y}$ appears in both the contributions of the encoder and the decoder, and this term is a variance maximisation term. Indeed, given that the data is centred, we can write the variance maximisation loss as

$$\mathcal{J}(\mathbf{W}) = -\frac{1}{2}\mathbb{E}(||\mathbf{x}\mathbf{W}||_2^2) = -\frac{1}{2}\mathbb{E}(||\mathbf{y}||_2^2), \tag{73}$$

which has the following gradient (omitting the expectation)

$$\frac{\partial \mathcal{J}}{\partial \mathbf{W}} = -\mathbf{x}^T\mathbf{x}\mathbf{W} = -\mathbf{x}^T\mathbf{y}. \tag{74}$$

Thus, the encoder multiplies the variance maximisation term with $(\mathbf{W}^T\mathbf{W} - \mathbf{I})$ and the decoder with $(\mathbf{W}\mathbf{W}^T - \mathbf{I})$. If $\mathbf{W}$ is constrained to be orthogonal, i.e. $\mathbf{W}^T\mathbf{W} = \mathbf{I}$, then the contribution from the encoder is zero, and the contribution from the decoder is dominate. We can also note that the variance maximisation of the latent results in a Hebbian update rule $\Delta\mathbf{W} \propto \mathbf{x}^T\mathbf{y}$ (Oja, 1982).

Let $\hat{\mathbf{x}} = \mathbf{y}\mathbf{W}^T$ and $\hat{\mathbf{y}} = \mathbf{x}\mathbf{W}$, then we can rewrite the contributions as

$$\frac{\partial \mathcal{L}}{\partial \mathbf{W}_e} = \mathbf{x}^T(\hat{\mathbf{y}} - \mathbf{y}) \tag{75}$$
$$\frac{\partial \mathcal{L}}{\partial \mathbf{W}_d} = (\hat{\mathbf{x}} - \mathbf{x})^T\mathbf{y}, \tag{76}$$

highlighting that the encoder puts an emphasis on latent reconstruction while the decoder puts an emphasis on input reconstruction. Rewritten as weight updates, we have

$$\Delta\mathbf{W}_e \propto \mathbf{x}^T\mathbf{y} - \mathbf{x}^T\mathbf{x}\mathbf{W} \tag{77}$$
$$\Delta\mathbf{W}_d \propto \mathbf{x}^T\mathbf{y} - \mathbf{W}\mathbf{y}^T\mathbf{y}. \tag{78}$$

A.2.2 Subspace learning algorithm

The subspace learning algorithm (Oja, 1983; Williams & University of California, 1985; Oja, 1989; Hyvärinen & Oja, 2000) was proposed as a generalisation of Oja's single component neural learning algorithm (Oja, 1982). The weight update has the following form:

$$\Delta \mathbf{W} = \mathbf{x}^T\mathbf{x}\mathbf{W} - \mathbf{W}(\mathbf{W}^T\mathbf{x}^T\mathbf{x}\mathbf{W}) \tag{79}$$
$$= \mathbf{x}^T\mathbf{y} - \mathbf{W}\mathbf{y}^T\mathbf{y}. \tag{80}$$

We see that this update is exactly the same as Eq. 78: it is simply the decoder contribution from a linear reconstruction loss. As we have seen from Eq. 66, the contribution of the encoder is zero if $\mathbf{W}$ is semi-orthogonal, so in the subspace learning algorithm the encoder contribution is simply omitted. The subspace learning algorithm can be obtained as a reconstruction loss with the stop gradient operator placed on the encoder:

$$\mathcal{L}(\mathbf{W}) = \frac{1}{2}||\mathbf{x}[\mathbf{W}]_{sg}\mathbf{W}^T - \mathbf{x}||_2^2. \tag{81}$$

A.3 Finding axis-aligned principal vectors

Here we look at the main idea for obtaining axis-aligned solutions: symmetry breaking. We saw in the previous section that the weight updates in the linear case are symmetric – no particular component is favoured over the other. Therefore, any method that breaks the symmetry will tend to lead to the axis-aligned PCA solution.

A.3.1 Weighted subspace algorithm

One straightforward way to break the symmetry is to simply weigh each component differently. This has been shown to converge (Oja, 1992a; Oja et al., 1992; Xu, 1993; Hyvärinen & Oja, 2000) to the PCA eigenvectors.

The weighted subspace algorithm does exactly that, and it modifies the update rule of the subspace learning algorithm into

$$\Delta \mathbf{W} = \mathbf{x}^T\mathbf{y} - \mathbf{W}\mathbf{y}^T\mathbf{y}\mathbf{\Lambda}^{-1}, \tag{82}$$
$$\text{or } \Delta \mathbf{W} = \mathbf{x}^T\mathbf{y}\mathbf{\Lambda} - \mathbf{W}\mathbf{y}^T\mathbf{y}. \tag{83}$$

where $\mathbf{\Lambda} = \text{diag}(\lambda_1, ..., \lambda_k)$ such that $\lambda_1 > ... > \lambda_k > 0$.

The second form can be written as a loss function:

$$\mathcal{L}(\mathbf{W}) = \frac{1}{2}\mathbb{E}(||\mathbf{x}[\mathbf{W}]_{sg}\mathbf{W}^T - \mathbf{x}||_2^2 - ||\mathbf{x}\mathbf{W}\mathbf{\Lambda}^{\frac{1}{2}}||_2^2 + ||\mathbf{x}\mathbf{W}||_2^2). \tag{84}$$

A.3.2 Weighted subspace algorithm with unit norm

Although the above updates (Eq. 82 and 83) do converge to the eigenvectors, they no longer maintain unit norm columns. To see why, we can look at the stationary point, where we know that $\Delta\mathbf{W}$ should be $\mathbf{0}$ and that the components of $\mathbf{y}$ should be uncorrelated, i.e. $\mathbb{E}(\mathbf{y}^T\mathbf{y}) = \hat{\mathbf{\Sigma}}$ is diagonal. We can write:

$$\Delta \mathbf{W} = \mathbf{0} \tag{85}$$
$$\mathbb{E}(\mathbf{x}^T\mathbf{y} - \mathbf{W}\mathbf{y}^T\mathbf{y}\mathbf{\Lambda}^{-1}) = \mathbf{0} \tag{86}$$
$$\mathbb{E}(\mathbf{W}^T\mathbf{x}^T\mathbf{y} - \mathbf{W}^T\mathbf{W}\mathbf{y}^T\mathbf{y}\mathbf{\Lambda}^{-1}) = \mathbf{0} \tag{87}$$
$$\mathbb{E}(\mathbf{y}^T\mathbf{y}) - \mathbf{W}^T\mathbf{W}\mathbb{E}(\mathbf{y}^T\mathbf{y})\mathbf{\Lambda}^{-1} = \mathbf{0} \tag{88}$$
$$\hat{\mathbf{\Sigma}}^2 - \mathbf{W}^T\mathbf{W}\hat{\mathbf{\Sigma}}^2\mathbf{\Lambda}^{-1} = \mathbf{0} \tag{89}$$
$$\mathbf{W}^T\mathbf{W}\mathbf{\Lambda}^{-1}\hat{\mathbf{\Sigma}}^2 = \hat{\mathbf{\Sigma}}^2 \tag{90}$$
$$\mathbf{W}^T\mathbf{W} = \mathbf{\Lambda}. \tag{91}$$

As we know $\mathbf{W}$ has orthogonal columns, $\mathbf{W}^T\mathbf{W}$ must be diagonal, and so the norm of each ith column becomes equal to $\sqrt{\lambda_i}$. If we want unit norm columns, a straightforward remedy is to normalise at the end of training. A closer look allows us to see that the main contributor to the norm is the diagonal part of $\mathbf{y}^T\mathbf{y}\mathbf{\Lambda}^{-1}$ – this suggests to us two options that could maintain unit norm columns.

As a first option, we can simply counteract the effect of $\mathbf{\Lambda}^{-1}$ on the diagonal by multiplying by its inverse on the other side of $\mathbf{y}^T\mathbf{y}$ to obtain

$$\Delta\mathbf{W} = \mathbf{x}^T\mathbf{y} - \mathbf{W}(\mathbf{y}\mathbf{\Lambda})^T\mathbf{y}\mathbf{\Lambda}^{-1}, \tag{92}$$

$$\text{or } \Delta\mathbf{W} = \mathbf{x}^T\mathbf{y}\mathbf{\Lambda} - \mathbf{W}(\mathbf{y}\mathbf{\Lambda})^T\mathbf{y}, \tag{93}$$

for which we now have

$$\Delta\mathbf{W} = \mathbf{0} \tag{94}$$

$$\hat{\mathbf{\Sigma}}^2 - \mathbf{W}^T\mathbf{W}\mathbf{\Lambda}\hat{\mathbf{\Sigma}}^2\mathbf{\Lambda}^{-1} = \mathbf{0} \tag{95}$$

$$\mathbf{W}^T\mathbf{W}\mathbf{\Lambda}\mathbf{\Lambda}^{-1}\hat{\mathbf{\Sigma}}^2 = \hat{\mathbf{\Sigma}}^2 \tag{96}$$

$$\mathbf{W}^T\mathbf{W} = \mathbf{I}. \tag{97}$$

As a second option, we can remove the diagonal part of $\mathbf{y}^T\mathbf{y}\mathbf{\Lambda}^{-1}$. This consists in adding

$$-\mathbf{W}(\text{diag}(\text{diag}^{-1}(\mathbf{y}^T\mathbf{y})) - \text{diag}(\text{diag}^{-1}(\mathbf{y}^T\mathbf{y}\mathbf{\Lambda}^{-1}))) \tag{98}$$

to Eq. 82, or

$$-\mathbf{W}(\text{diag}(\text{diag}^{-1}(\mathbf{y}^T\mathbf{y}\mathbf{\Lambda})) - \text{diag}(\text{diag}^{-1}(\mathbf{y}^T\mathbf{y}))) \tag{99}$$

to Eq. 83. The latter can be derived from the gradient of

$$\frac{1}{2}(||\mathbf{W}[\hat{\mathbf{\Sigma}}]_{sg}\mathbf{\Lambda}^{\frac{1}{2}}||_2^2 - ||\mathbf{W}[\hat{\mathbf{\Sigma}}]_{sg}||_2^2). \tag{100}$$

We thus arrive at a total loss that maintains unit norm columns:

$$\mathcal{L}(\mathbf{W}) = \frac{1}{2}\mathbb{E}(||\mathbf{x}[\mathbf{W}]_{sg}\mathbf{W}^T - \mathbf{x}||_2^2 + ||\mathbf{W}[\hat{\mathbf{\Sigma}}]_{sg}\mathbf{\Lambda}^{\frac{1}{2}}||_2^2 - ||\mathbf{W}[\hat{\mathbf{\Sigma}}]_{sg}||_2^2 - ||\mathbf{x}\mathbf{W}\mathbf{\Lambda}^{\frac{1}{2}}||_2^2 + ||\mathbf{x}\mathbf{W}||_2^2) \tag{101}$$

We see that this combines reconstruction, weighted regularisation, and weighted variance maximisation. We can also note from this that λ_i should be ≤ 1 to avoid the variance maximisation term overpowering the other terms.

A.3.3 Generalised Hebbian Algorithm (GHA)

The generalised Hebbian algorithm (GHA) (Sanger, 1989) learns multiple PCA components by combining Oja's update rule (Oja, 1982) with a Gram-Schmidt-like orthogonalisation term. It breaks the symmetry in the subspace learning algorithm by taking the lower triangular part of $\mathbf{y}^T\mathbf{y}$, allowing the weights to converge to the true PCA eigenvectors.

$$\mathbf{y} = \mathbf{x}\mathbf{W} \tag{102}$$

$$\Delta\mathbf{W} \propto \mathbf{x}^T\mathbf{y} - \mathbf{W}\triangle(\mathbf{y}^T\mathbf{y}) \tag{103}$$

One way to write this as a loss function is to use variance maximisation and the stop gradient operator to get the following:

$$\mathcal{L}(\mathbf{W}) = -\frac{1}{2}||\mathbf{x}\mathbf{W}||_2^2 + \mathbf{1}\mathbf{W} \odot [\mathbf{W}\triangle(\mathbf{y}^T\mathbf{y})]_{sg}\mathbf{1}^T. \tag{104}$$

A.3.4 Autoencoder with GHA update

We can also consider combining the orthogonalisation term of GHA with the autoencoder reconstruction to get

$$\mathcal{L}(\mathbf{W}) = ||\mathbf{x} - \mathbf{x}\mathbf{W}\mathbf{W}^T||_2^2 + \mathbf{1}\mathbf{W} \odot [\mathbf{W}\triangleq(\mathbf{y}^T\mathbf{y})]_{sg}\mathbf{1}^T, \tag{105}$$

where $\triangleq$ is the operation that takes the lower triangular without the diagonal; otherwise, the term $-\mathbf{W}\text{diag}(\text{diag}^{-1}(\mathbf{y}^T\mathbf{y}))$ would be doubled unless we also include another $\mathbf{x}^T\mathbf{y}$ to counteract it. The double term might still work in practice, but it becomes detrimental when combined with a projective unit norm constraint. If $\mathbf{W}$ is orthogonal, then this is similar to combining the subspace update rule (Eq. 80) with the orthogonalisation term of GHA to result in

$$\Delta\mathbf{W} = \mathbf{x}^T\mathbf{y} - \mathbf{W}\mathbf{y}^T\mathbf{y} - \mathbf{W}\triangleq(\mathbf{y}^T\mathbf{y}), \tag{106}$$

which can be obtained from the following loss

$$\mathcal{L}(\mathbf{W}) = ||\mathbf{x} - \mathbf{x}[\mathbf{W}]_{sg}\mathbf{W}^T||_2^2 + \mathbf{1}\mathbf{W} \odot [\mathbf{W}\triangleq(\mathbf{y}^T\mathbf{y})]_{sg}\mathbf{1}^T. \tag{107}$$

Nonetheless, keeping the encoder contribution can still help maintain the orthogonality of $\mathbf{W}$, as the term $\mathbf{W}^T\mathbf{W} - \mathbf{I}$ is similar to the one derived from a symmetric orthogonalisation regulariser (see Appendix B).

Another variant we can consider is to make triangular the full gradient of the linear autoencoder update, which includes not just the contribution of the decoder but also that of the encoder, i.e. by taking

$$\frac{\partial\mathcal{L}}{\partial\mathbf{W}} = \mathbf{x}^T\mathbf{y}(\mathbf{W}^T\mathbf{W} - \mathbf{I}) - \mathbf{x}^T\mathbf{y} + \mathbf{W}\mathbf{y}^T\mathbf{y} \tag{108}$$

and changing it into

$$\frac{\partial\mathcal{L}}{\partial\mathbf{W}} = \mathbf{x}^T\mathbf{y}(\triangle(\mathbf{W}^T\mathbf{W}) - \mathbf{I}) - \mathbf{x}^T\mathbf{y} + \mathbf{W}\triangle(\mathbf{y}^T\mathbf{y}). \tag{109}$$

Though the contribution from the encoder is negligible when $\mathbf{W}$ is close to orthogonal, the term $\triangle\mathbf{W}^T\mathbf{W} - \mathbf{I}$ also acts as an approximation to Gram-Schmidt orthogonalisation (see Appendix B), and so when $\mathbf{W}$ deviates from being orthogonal, it can help bring it back more quickly.

A.3.5 Nested dropout

Another way to break the symmetry and enforce an ordering is nested dropout (Rippel et al., 2014), a procedure for randomly removing nested sets of components. It can be shown (Rippel et al., 2014) that it leads to an exact equivalence with PCA in the case of a single-layer linear autoencoder. An idea similar in spirit was the previously proposed hierarchical PCA method (Scholz & Vigário, 2002) for ordering of the principal components; however, it was limited in number of dimensions and was not stochastic. Nested dropout works by assigning a prior distribution $p(.)$ over the component indices $1...k$, and then using it to sample an index $\mathcal{L}$ and drop the units $\mathcal{L}+1,...,k$. A typical choice of distribution is a geometric distribution with $p(j) = \rho^{j-1}(1-\rho)$, $0 < \rho < 1$.

Let $\mathbf{m}_{1|j} \in \{0,1\}^k$ be a vector such that $m_i = \begin{cases} 1 & \text{if } i < j \\ 0 & \text{otherwise} \end{cases}$, then the reconstruction loss becomes

$$\mathcal{L}(\mathbf{W}) = \frac{1}{2}\mathbb{E}(||(\mathbf{x}\mathbf{W} \odot \mathbf{m}_{1|j})\mathbf{W}^T - \mathbf{x}||_2^2), \tag{110}$$

where the mask $\mathbf{m}_{1|j}$ is randomly sampled during each update step. One issue with nested dropout is that the higher the index the smaller the gradient update. This means that training has to run much longer for them to converge. One suggested remedy for this (Rippel et al., 2014) is to perform gradient sweeping, where once an earlier index has converged, it can be frozen and the nested dropout index can be incremented.

A non-stochastic version, which is an extension of the hierarchical PCA method (Scholz & Vigário, 2002), sums all the reconstruction loss to result in

$$\mathcal{L}(\mathbf{W}) = \frac{1}{2}\sum_j \mathbb{E}(||\mathbf{x}\mathbf{W}_{1|j}\mathbf{W}_{1|j}^T - \mathbf{x}||_2^2), \tag{111}$$

where $\mathbf{W}_{1|j}$ is the truncated matrix that contains the first j columns of $\mathbf{W}$. However, this can get more computationally demanding when there is a large number of components compared to the stochastic version.

A.3.6 Weighted variance maximisation

We can formulate a variant of the weighted subspace algorithm more explicitly as a reconstruction loss combined with a weighted variance maximisation term:

$$\mathcal{L}(\mathbf{W}) = \frac{1}{2}\mathbb{E}(||\mathbf{x} - \mathbf{x}\mathbf{W}\mathbf{W}^T||_2^2 - \alpha||\mathbf{x}\mathbf{W}\mathbf{\Lambda}^{\frac{1}{2}}||_2^2), \tag{112}$$

where $\mathbf{\Lambda} = \text{diag}(\lambda_1, ..., \lambda_k)$ consists of linearly spaced values between 0 and 1, and $\alpha \in \{-1, 1\}$. We can also pick $\mathbf{\Lambda}$ to be proportional to the variances, but the term should not be trainable – we can do that with the stop gradient operator: $\mathbf{\Lambda} = [\frac{\mathbb{E}(\mathbf{y}^2)}{\max_i(\mathbb{E}(y_i^2))}]_{sg}$.

The associated gradient of the loss is

$$\frac{\partial \mathcal{L}}{\partial \mathbf{W}} = -\mathbf{x}^T\mathbf{y} + \mathbf{W}\mathbf{y}^T\mathbf{y} + \mathbf{x}^T\mathbf{y}(\mathbf{W}^T\mathbf{W} - \mathbf{I}) - \alpha\mathbf{x}^T\mathbf{y}\mathbf{\Lambda} \tag{113}$$

$$= -\mathbf{x}^T\mathbf{y}(\mathbf{I} + \alpha\mathbf{\Lambda}) + \mathbf{W}\mathbf{y}^T\mathbf{y} + \mathbf{x}^T\mathbf{y}(\mathbf{W}^T\mathbf{W} - \mathbf{I}). \tag{114}$$

Following a similar analysis as in Appendix A.3.2, it can be shown that at the stationary point we have $\mathbf{W}^T\mathbf{W} = \mathbf{I} + \frac{1}{2}\alpha\mathbf{\Lambda}$.

Alternative, we can use a stochastic weighting by applying nested dropout on the variance regulariser:

$$\mathcal{L}(\mathbf{W}) = \mathbb{E}(||\mathbf{x} - \mathbf{x}\mathbf{W}\mathbf{W}^T||_2^2 - \alpha||\mathbf{x}\mathbf{W} \odot \mathbf{m}_{1|j}||_2^2), \tag{115}$$

with a randomly sampled mask per data sample.

B Enforcing orthogonality

There are a few ways to enforce orthogonality either via a regulariser or a projective constraint, and this can be symmetric or asymmetric.

B.1 Symmetric

A symmetric orthogonalisation scheme can be used when there are no favoured components.

Regulariser A straightforward regulariser, which also enforces unit norm, is

$$J(\mathbf{W}) = \alpha||\mathbf{I} - \mathbf{W}^T\mathbf{W}||_F^2, \tag{116}$$

and it has the gradient

$$\frac{\partial J}{\partial \mathbf{W}} = 4\alpha\mathbf{W}(\mathbf{W}^T\mathbf{W} - \mathbf{I}). \tag{117}$$

Combining the above with PCA or ICA updates has been previously referred to as the bigradient rule (Wang et al., 1995; Wang & Karhunen, 1996; Karhunen et al., 1997), where α is at most $\frac{1}{8}$, for which the justification can be derived from its corresponding projective constraint.

Projective constraint An iterative algorithm, as used in FastICA (Hyvarinen, 1999), is the following:

1. $\mathbf{w}_i \leftarrow \dfrac{\mathbf{w}_i}{||\mathbf{w}_i||}$
2. Repeat until convergence:
   $\mathbf{W} \leftarrow \dfrac{3}{2}\mathbf{W} - \dfrac{1}{2}\mathbf{W}\mathbf{W}^T\mathbf{W}.$

We can note that

$$\frac{3}{2}\mathbf{W} - \frac{1}{2}\mathbf{W}\mathbf{W}^T\mathbf{W} = \mathbf{W} - \frac{1}{2}\mathbf{W}(\mathbf{W}^T\mathbf{W} - \mathbf{I}) \tag{118}$$

$$= \mathbf{W} - \frac{1}{8\alpha}\frac{\partial J}{\partial \mathbf{W}}, \tag{119}$$

which allows to to see that we can set $\alpha = \frac{1}{8}$ in Eq. 117 to derive the projective constraint from the stochastic gradient descent update.

The iterative algorithm can be shown to converge (Hyvarinen, 1999; Hyvärinen & Oja, 2000) by analysing the evolution of the eigenvalues of $\mathbf{W}^T\mathbf{W}$. The initial normalisation guarantees that, prior to any iterations, the eigenvalues of $\mathbf{W}^T\mathbf{W}$ are ≤ 1. Let $\mathbf{E}^T\mathbf{\Lambda}\mathbf{E}$ be the eigendecomposition of $\mathbf{W}^T\mathbf{W}$, and let us consider the general case

$$\hat{\mathbf{W}} = \mathbf{W} - \beta\mathbf{W}(\mathbf{W}^T\mathbf{W} - \mathbf{I}) \tag{120}$$

$$= (1 + \beta)\mathbf{W} - \beta\mathbf{W}\mathbf{W}^T\mathbf{W}. \tag{121}$$

After one iteration we have

$$\hat{\mathbf{W}}^T\hat{\mathbf{W}} = (1+\beta)^2\mathbf{W}^T\mathbf{W} - 2(1+\beta)\beta(\mathbf{W}^T\mathbf{W})^2 + \beta^2(\mathbf{W}^T\mathbf{W})^3 \tag{122}$$
$$= (1+\beta)^2\mathbf{E}^T\mathbf{\Lambda}\mathbf{E} - 2(1+\beta)\beta(\mathbf{E}^T\mathbf{\Lambda}\mathbf{E})^2 + \beta^2(\mathbf{E}^T\mathbf{\Lambda}\mathbf{E})^3 \tag{123}$$
$$= (1+\beta)^2\mathbf{E}^T\mathbf{\Lambda}\mathbf{E} - 2(1+\beta)\beta\mathbf{E}^T\mathbf{\Lambda}^2\mathbf{E} + \beta^2\mathbf{E}^T\mathbf{\Lambda}^3\mathbf{E} \tag{124}$$
$$= \mathbf{E}^T((1+\beta)^2\mathbf{\Lambda} - 2(1+\beta)\beta\mathbf{\Lambda}^2 + \beta^2\mathbf{\Lambda}^3)\mathbf{E}. \tag{125}$$

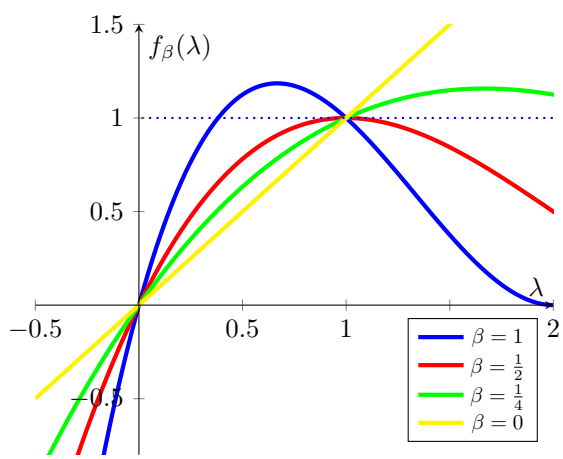
We now have the eigenvalues after one iteration as a function of the eigenvalues of the previous iteration:

$$f_\beta(\lambda) = (1+\beta)^2\lambda - 2(1+\beta)\beta\lambda^2 + \beta^2\lambda^3 \tag{126}$$
$$f'_\beta(\lambda) = (1+\beta)^2 - 4(1+\beta)\beta\lambda + 3\beta^2\lambda^2. \tag{127}$$

We know that the eigenvalues are in the interval $]0, 1]$. For $\lambda = 1$ we have

$$f_\beta(1) = 1 \tag{128}$$
$$f'_\beta(1) = (1+\beta)^2 - 4(1+\beta)\beta + 3\beta^2 = 1 - 2\beta. \tag{129}$$

We see that there is a stationary point at $\beta = \frac{1}{2}$. Noting that $f_0(\lambda) = \lambda$, we can plot f_β:



We see that if $\lambda \in]0, 1]$ and $\beta \in]0, \frac{1}{2}]$, then the eigenvalues will always be in $]0, 1]$; and given that $f(\lambda) > \lambda$, all the eigenvalues will eventually converge to 1.

Thus, for $\beta = \frac{1}{2}$ we get the fastest convergence where λ does not exceed $]0, 1]$. This means that when using stochastic gradient descent with a learning rate of 1 and with a symmetric regulariser (Eq. 116), an optimal value for α is $\frac{1}{8}$. This analysis does not consider potential other interactions from the use of additional losses that could affect $\mathbf{W}$ or the use of an adaptive optimiser like Adam Kingma & Ba (2014).

B.2 Asymmetric

Regulariser From the symmetric regulariser we can derive an asymmetric version that performs a Gram-Schmidt-like orthogonalisation. We simply need to use the lower (or upper) triangular part of $\mathbf{W}^T\mathbf{W}$ while using a stop gradient on either $\mathbf{W}$ or $\mathbf{W}^T$. This means that the loss, which also maintains unit norm, is

$$J(\mathbf{W}) = \frac{1}{2}||\triangle(\mathbf{W}^T[\mathbf{W}]_{sg}) - \mathbf{I}||_F^2 \tag{130}$$

or without the diagonal (to remove the unit norm regulariser):

$$J(\mathbf{W}) = \frac{1}{2}||\text{⊴}(\mathbf{W}^T[\mathbf{W}]_{sg})||_F^2, \tag{131}$$

where ⊴ refers to the lower triangular matrix without the diagonal. From this, we have

$$\frac{\partial J}{\partial \mathbf{W}} = \mathbf{W}\text{⊴}(\mathbf{W}^T\mathbf{W} - \mathbf{I}), \tag{132}$$

or

$$\frac{\partial J}{\partial \mathbf{W}} = \mathbf{W}\text{⊴}(\mathbf{W}^T\mathbf{W}). \tag{133}$$

The use of such an update has been previously referred to as the hierarchic version of the bigradient algorithm (Wang et al., 1995).

Recall that in the weight update of GHA (Appendix A.3.3) we have the term $\mathbf{W}\text{⊴}(\mathbf{y}^T\mathbf{y})$; the asymmetric update can benefit from being at a similar scale in order to be effective, especially when the variances are large. To do this, we can weight the loss by the non-trainable standard deviations to obtain

$$J(\mathbf{W}) = \frac{1}{2}||\text{⊴}(\mathbf{W}^T[\mathbf{W}\hat{\mathbf{\Sigma}}]_{sg})||_F^2. \tag{134}$$

Gram-Schmidt projective constraint After each gradient update step, we can apply the Gram-Schmidt orthogonalisation procedure (Schmidt, 1907).

B.3 Encoder contribution of linear reconstruction

As noted in Appendix A.2.1, in a linear autoencoder the contribution of the encoder to the gradient is zero when $\mathbf{W}$ is orthogonal, and so if used in isolation it can serve as an implicit orthogonal regulariser:

$$J(\mathbf{W}) = \mathbb{E}(||\mathbf{x} - \mathbf{x}\mathbf{W}[\mathbf{W}^T]_{sg}||_2^2), \tag{135}$$

having

$$\frac{\partial J}{\partial \mathbf{W}} = \mathbf{x}^T\mathbf{y}(\mathbf{W}^T\mathbf{W} - \mathbf{I}) \tag{136}$$

as gradient. We see that it simply replaces in the symmetric regulariser the first $\mathbf{W}$ with $\mathbf{x}^\mathbf{T}\mathbf{y}$.

C Enforcing unit norm

If not already taken care of by the orthogonality constraint, there are a few ways to enforce unit norm.

C.1 Projective constraint

The columns $\mathbf{w}_i^T$ of $\mathbf{W}$ can be normalised to unit norm after each update step:

$$\mathbf{w}_i^T \leftarrow \frac{\mathbf{w}_i^T}{||\mathbf{w}_i^T||}. \tag{137}$$

C.2 Regularisation

We can enforce unit norm by adding a regulariser on the norm of the column vectors $\mathbf{w}_i^T$. This results in the loss

$$J(\mathbf{w}_i^T) = \frac{1}{2}(\mathbf{1} - ||\mathbf{w}_i^T||_2)^2 \tag{138}$$

which has the following gradient

$$\frac{\partial J}{\partial \mathbf{w}_i^T} = (1 - \frac{1}{||\mathbf{w}_i^T||})\mathbf{w}_i^T \tag{139}$$

C.3 Differentiable weight normalisation

A third way to do this is to use differentiable weight normalisation (Salimans & Kingma, 2016), except without the scale parameter. This consists in replacing the column vectors with

$$\hat{\mathbf{w}}_i^T = \frac{\mathbf{w}_i^T}{||\mathbf{w}_i^T||}. \tag{140}$$

This has the following Jacobian

$$\frac{\partial \hat{\mathbf{w}}_i^T}{\partial \mathbf{w}_i^T} = \frac{1}{||\mathbf{w}_i^T||}(\mathbf{I} - \frac{\mathbf{w}_i^T \mathbf{w}_i}{||\mathbf{w}_i^T||^2}), \tag{141}$$

which results for a given loss $\mathcal{L}$ in

$$\frac{\partial \mathcal{L}}{\partial \mathbf{w}_i^T} = \frac{\partial \hat{\mathbf{w}}_i^T}{\partial \mathbf{w}_i^T} \frac{\partial \mathcal{L}}{\partial \hat{\mathbf{w}}_i^T}. \tag{142}$$

D Linear ICA

D.1 Overview

The goal of ICA (Jutten & Herault, 1991; Comon, 1994; Bell & Sejnowski, 1997; Hyvarinen, 1999) is to linearly transform the data into a set of components that are as statistically independent as possible. That is, if $\mathbf{x} \in \mathbb{R}^{1\times k}$ is a row vector, the goal is to find an unmixing matrix $\mathbf{B} \in \mathbb{R}^{k\times k}$ such that

$$\mathbf{y} = \mathbf{x}\mathbf{B} \tag{143}$$

has its components $y_1, ..., y_k$ as independent as possible.

In an alternative equivalent formulation, we assume that the observed $x_1, ..., x_k$ were generated by mixing k independent sources using a mixing matrix $\mathbf{A}$, i.e. we have $\mathbf{x} = \mathbf{s}\mathbf{A}$ with $\mathbf{A} = \mathbf{B}^{-1}$ and $\mathbf{s} = \mathbf{y}$, and the goal is to recover the mixing matrix.

To be able to estimate $\mathbf{B}$, at least $k-1$ of the independent components *must* have non-Gaussian distributions. Otherwise, if the independent components have Gaussian distributions, then the model is *not* identifiable.

Without any other assumptions about how the data was mixed, ICA has two ambiguities: it is not possible to determine the variances nor the order of the independent sources.

There are a few different algorithms for performing ICA (Hyvärinen & Oja, 2000; Bingham et al., 2015), the most popular being FastICA (Hyvarinen, 1999). In the ICA model, the mixing matrix can be arbitrary, meaning that it is not necessarily orthogonal; however, to go about finding it, ICA algorithms generally decompose it into a sequence of transformations that include orthogonal matrices, for there is a guarantee that any matrix has an SVD that encodes the sequence: rotation (orthogonal), scaling (diagonal), rotation (orthogonal). FastICA uses PCA as a preprocessing step to whiten the data and reduce dimensionality. The whitening operation performs the initial rotation and scaling. What remains after that is to find the last rotation that maximises non-Gaussianity. If $\mathbf{USV^T}$ is the SVD of the covariance matrix, then the whitening transformation is $\mathbf{VS^{-1}}$, and the resulting overall ICA transformation has the following form

$$\mathbf{B} = \mathbf{VS}^{-1}\mathbf{W}, \tag{144}$$

with $\mathbf{V}$ and $\mathbf{W}$ orthogonal matrices and $\mathbf{S}$ diagonal.

Ordering of components As ICA makes no assumption about the transformation, all the ICs are assumed to have unit variance. And so there is no order implied. Without prior knowledge about how the mixing of the sources occurred, it is impossible to resolve the ICA ambiguity. However, if we have reason to assume that the unmixing matrix is close to orthogonal, or simply has unit norm columns (akin to relaxing the orthogonality constraint in PCA while maintaining unit norm directions) then we can in fact order the components by their variances. In this case, the norm of the columns of the unmixing matrix obtained by FastICA is inversely proportional to the standard deviations of the components. On the other hand, it is also possible to assume that it is the mixing matrix, rather than the unmixing matrix, that has unit norm columns (Hyvärinen, 1999). It is also possible to base the ordering on the measure of non-Gaussianity (Hyvärinen, 1999).

D.2 Equivariant adaptive separation via independence

Equivariant adaptive separation via independence (EASI) (Cardoso & Laheld, 1996) is a serial updating algorithm for source separation. It combines both a whitening term

$$\mathbf{W}(\mathbf{y}^T\mathbf{y} - \mathbf{I}) \tag{145}$$

and a skew-symmetric term

$$\mathbf{W}(\mathbf{y}^T h(\mathbf{y}) - h(\mathbf{y})^T\mathbf{y}) \tag{146}$$

to obtain the following global relative gradient update rule

$$\Delta\mathbf{W} = -\eta\mathbf{W}(\mathbf{y}^T\mathbf{y} - \mathbf{I} + \mathbf{y}^T h(\mathbf{y}) - h(\mathbf{y})^T\mathbf{y}), \tag{147}$$

with η the learning rate. The skew-symmetric term originates from skew-symmetrising $\mathbf{W}h(\mathbf{y})^T\mathbf{y}$ in order to roughly preserve orthogonality with each update. To see why, suppose we have $\mathbf{W}^T\mathbf{W} = \mathbf{I}$, and we modify it into $\mathbf{W} + \mathbf{W}\boldsymbol{\mathcal{E}}$, then we can expand it as

$$(\mathbf{W} + \mathbf{W}\boldsymbol{\mathcal{E}})^T(\mathbf{W} + \mathbf{W}\boldsymbol{\mathcal{E}}) = \mathbf{I} + \boldsymbol{\mathcal{E}}^T + \boldsymbol{\mathcal{E}} + \boldsymbol{\mathcal{E}}^T\boldsymbol{\mathcal{E}}. \tag{148}$$

If we want $\mathbf{W}+\mathbf{W}\boldsymbol{\mathcal{E}}$ to remain orthogonal up to first-order, we must also have $(\mathbf{W}+\mathbf{W}\boldsymbol{\mathcal{E}})^T(\mathbf{W}+\mathbf{W}\boldsymbol{\mathcal{E}}) = \mathbf{I}+o(\boldsymbol{\mathcal{E}})$. This implies that $\boldsymbol{\mathcal{E}}$ must be skew-symmetric with $\boldsymbol{\mathcal{E}}^T = -\boldsymbol{\mathcal{E}}$.

D.3 Non-identifiability of a Gaussian

When the criterion used depends on the variance and/or the independence of components, the only case where it is possible to identify a Gaussian distribution is when the transformation is orthogonal and all the

variances are clearly distinct. This means that linear PCA can identify the sources if they have distinct variances and they were transformed by a rotation. It is impossible to identify in any other case.

Given $\mathbf{y} \sim \mathcal{N}(\mathbf{0}, \mathbf{\Sigma}^2)$ and $\mathbf{x} = \mathbf{y}\mathbf{A}$, with $\mathbf{\Sigma}^2 = \text{diag}(\boldsymbol{\sigma}^2)$, then, from the affine property of the multivariate Gaussian, we have

$$\mathbf{x} \sim \mathcal{N}(\mathbf{0}, \mathbf{A}^T\mathbf{\Sigma}^2\mathbf{A}). \tag{149}$$

If $\mathbf{A} = \mathbf{B}^{-1}$ is orthogonal, then if all the variances are distinct then it is possible to identify up to sign indeterminacy. But if components have equal variances, then they are not possible to identify.

If $\mathbf{A} = \mathbf{B}^{-1}$ is non-orthogonal, then given the symmetry of $\mathbf{A}^T\mathbf{\Sigma}^2\mathbf{A}$, the spectral theorem allows us to write $\mathbf{A}^T\mathbf{\Sigma}^2\mathbf{A} = \mathbf{Q}^T\mathbf{\Lambda}\mathbf{Q}$, where $\mathbf{Q}$ is orthogonal and $\mathbf{\Lambda}$ is diagonal. Therefore there is no way to recover the non-orthogonal $\mathbf{A}$ even if the sources have distinct variances because it could have been equally transformed by the orthogonal $\mathbf{Q}$.

E Probabilistic PCA and Factor Analysis

Probabilistic PCA (pPCA) (Tipping & Bishop, 1999) and Factor Analysis (FA) (Harman, 1976) are two other methods for learning linear transformations. The former is related to linear PCA whereas the latter – in a loose sense – to linear ICA. The primary difference is that, unlike the standard linear PCA and ICA models, the pPCA and FA models include an additional term for modelling the noise.

Without loss of generality, we will assume that the inputs have zero mean. Let $\mathbf{x} \in \mathbb{R}^{1\times p}$ be an input sample and $\mathbf{y} \in \mathbb{R}^{1\times k}$ its corresponding latent. The probabilistic PCA model is of the form

$$\mathbf{x} = \mathbf{y}\mathbf{A} + \lambda\boldsymbol{\epsilon}, \tag{150}$$

where $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ and λ is a scalar. The covariance of the input can be expressed as $\mathbf{C} = \frac{1}{n}\mathbb{E}(\mathbf{x}^T\mathbf{x}) = \mathbf{A}^T\mathbf{A} + \lambda^2\mathbf{I}$. The pPCA solution takes the form

$$\mathbf{A} = \mathbf{\Sigma}\mathbf{U}^T, \tag{151}$$

where $\mathbf{\Sigma} = (\max(\mathbf{0}, \mathbf{S} - \lambda^2\mathbf{I}))^{1/2}$, $\mathbf{S}$ is a diagonal matrix of the eigenvalues of $\mathbf{C}$, and $\mathbf{U}$ is an orthogonal matrix of the eigenvectors of $\mathbf{C}$. We can express the latents as a function of the input using

$$\mathbf{B} = \mathbf{A}^{-1} = \mathbf{U}\mathbf{\Sigma}^{-1}. \tag{152}$$

The solution is thus no different than that of linear PCA, except that the standard deviations are truncated by that of the noise.

FA further generalises probabilistic PCA by modelling the scale of each component of the noise independently, and so its data model is

$$\mathbf{x} = \mathbf{y}\mathbf{A} + \boldsymbol{\epsilon}\mathbf{\Lambda}, \tag{153}$$

where $\mathbf{\Lambda}$ is a diagonal matrix. FA goes about finding both $\mathbf{\Lambda}$ and $\mathbf{A}$ in an iterative fashion (Barber, 2012). This originates from the fact that if we start from an initial guess of $\mathbf{\Lambda}$, estimated from the variances of the input data, then we can reduce the FA model to that of pPCA by dividing by $\mathbf{\Lambda}$ to obtain

$$\mathbf{x}' = \mathbf{x}\mathbf{\Lambda}^{-1} = \mathbf{y}\mathbf{A}\mathbf{\Lambda}^{-1} + \boldsymbol{\epsilon}. \tag{154}$$

This can now be solved in the same way as pPCA. This processes is repeated a number of times until the estimate of the loglikelihood of the data, under the Gaussian assumption, no longer changes. This means that the solution obtained ends up in being of the form

$$\mathbf{A} = \mathbf{\Sigma}\mathbf{U}^T\mathbf{\Lambda}, \tag{155}$$

where $\mathbf{\Sigma} = (\max(\mathbf{0}, \mathbf{S} - \mathbf{I}))^{1/2}$, and $\mathbf{S}$ and $\mathbf{U}$ correspond, respectively, to the eigenvalues and eigenvectors of $\mathbf{\Lambda}^{-1}\mathbf{C}\mathbf{\Lambda}^{-1}$. This means that, unlike PCA, the matrix $\mathbf{A}$ does not necessarily have orthogonal rows because $\mathbf{A}\mathbf{A}^T = \mathbf{\Sigma}\mathbf{V}^T\mathbf{\Lambda}^2\mathbf{V}\mathbf{\Sigma}$. Some FA methods proceed with doing an additional rotation by multiplying with another orthogonal matrix $\mathbf{V} \in \mathbb{R}^{k\times k}$ to obtain

$$\mathbf{A} = \mathbf{V}^T\mathbf{\Sigma}\mathbf{U}^T\mathbf{\Lambda}. \tag{156}$$

This means that the overall transformation of from the inputs to the latents is

$$\mathbf{B} = \mathbf{\Lambda}^{-1}\mathbf{U}\mathbf{\Sigma}^{-1}\mathbf{V}. \tag{157}$$

We this see that, similarly to linear ICA, FA estimates two orthogonal transformations, and so the resulting overall transformation is not necessarily orthogonal. Where they differ is in the presence of an additional scaling of the inputs, and notably, in the process of finding the second rotation, linear ICA maximises independence, whereas FA maximises variance (Hyvärinen & Oja, 2000).

From the point of view of learning a linear transformation $\mathbf{B}$ such that $\mathbf{y} = \mathbf{x}\mathbf{B}$, we can summarise the models learnt by the different methods in Tab. 1.

	$\mathbf{B}$	Criterion
Linear PCA	$\mathbf{W}\mathbf{\Sigma}^{-1}$	variance
Probabilistic PCA	$\mathbf{W}\mathbf{\Sigma}^{-1}$	variance
Nonlinear PCA	$\mathbf{W}\mathbf{\Sigma}^{-1}$	variance and independence
Linear ICA	$\mathbf{W}\mathbf{\Sigma}^{-1}\mathbf{V}$	variance ($\mathbf{W}$); independence ($\mathbf{V}$)
Factor Analysis	$\mathbf{\Lambda}^{-1}\mathbf{W}\mathbf{\Sigma}^{-1}\mathbf{V}$	variance

Table 1: Linear transformation models learnt by the different methods. Both $\mathbf{W}$ and $\mathbf{V}$ are orthogonal, whereas both $\mathbf{\Lambda}$ and $\mathbf{\Sigma}$ are diagonal. The optimal values found for each of the variables are not necessarily the same. We can note that FA has the most parameters.

F Nonlinear PCA

F.1 Stationary point

F.1.1 Modified derivative

Lets consider the case where $h'(x) = 1$. This approximation does not seem to affect the learned filters when $\mathbf{\Sigma}$ is estimated from the data (see Appendix G.4). We can write our gradient as

$$\frac{\partial \mathcal{L}}{\partial \mathbf{W}} \approx \mathbf{x}^T(h(\mathbf{y}\mathbf{\Sigma}^{-1})\mathbf{\Sigma}\mathbf{W}^T\mathbf{W} - \mathbf{y}) = \mathbf{x}^T(\hat{\mathbf{y}} - \mathbf{y}). \tag{158}$$

We can take a look at what happens at the stationary point. Taking Eq. 158, and assuming that $\mathbf{W}^T\mathbf{W} = \mathbf{I}$, we have

$$\Delta \mathbf{W} = \mathbf{x}^T\mathbf{y} - \mathbf{x}^T h(\mathbf{y}\mathbf{\Sigma}^{-1})\mathbf{\Sigma}\mathbf{W}^T\mathbf{W} \tag{159}$$
$$\mathbf{0} = \mathbb{E}(\mathbf{W}^T\mathbf{x}^T\mathbf{y} - \mathbf{W}^T\mathbf{x}^T h(\mathbf{y}\mathbf{\Sigma}^{-1})\mathbf{\Sigma}\mathbf{W}^T\mathbf{W}) \tag{160}$$
$$\mathbf{0} = \mathbf{\Sigma}^2 - \mathbb{E}(\mathbf{y}^T h(\mathbf{y}\mathbf{\Sigma}^{-1}))\mathbf{\Sigma}\mathbf{W}^T\mathbf{W} \tag{161}$$
$$\mathbf{\Sigma}^2 = \mathbb{E}(\mathbf{y}^T h(\mathbf{y}\mathbf{\Sigma}^{-1}))\mathbf{\Sigma}\mathbf{I} \tag{162}$$
$$\mathbf{\Sigma} = \mathbb{E}(\mathbf{y}^T h(\mathbf{y}\mathbf{\Sigma}^{-1})). \tag{163}$$

Given that $\mathbb{E}(\mathbf{y}^T h(\mathbf{y}\mathbf{\Sigma}^{-1}))$ is expected to be diagonal, we end up with $\sigma_i = \mathbb{E}(y_i h(y_i/\sigma_i))$. To be able to satisfy the above, we would need h to adapt such that $\sqrt{\text{var}(h(\mathbf{y}\mathbf{\Sigma}^{-1}))} \approx \mathbf{1}$. We can do this by using a function that adjusts the scale, such as $h_a(x) = a\tanh(x/a)$. For instance, if $x \sim \mathcal{N}(0,1)$, we have $\sqrt{\text{var}(tanh(x))} \approx 0.62$ and $\sqrt{\text{var}(3\tanh(\frac{x}{3}))} \approx 0.90$. So by setting a value of at least $a = 3$, we get closer back to unit variance.

Alternatively, we can consider an asymmetric nonlinear function such as $a\tanh(x)$, where $a = 1/\sqrt{\text{var}(tanh(x))}$. If the standardised x is roughly standard normal, then $a \approx 1.6$.

We also have another option instead of scaling: we can compensate by adding an additional loss term that adds to the encoder contribution as follows:

$$\mathcal{L}(\mathbf{W}) = \mathbb{E}(||\mathbf{x} - h(\mathbf{x}\mathbf{W}\mathbf{\Sigma}^{-1})\mathbf{\Sigma}[\mathbf{W}^T]_{sg}||_2^2) \tag{164}$$
$$+ \alpha\mathbb{E}(||\mathbf{x} - \mathbf{x}\mathbf{W}[\mathbf{W}^T]_{sg}||_2^2) \tag{165}$$
$$h(x) = \tanh(x) \tag{166}$$
$$h'(x) = 1 \tag{167}$$

This results in

$$\Delta \mathbf{W} = \alpha(\mathbf{x}^T\mathbf{y} - \mathbf{x}^T\mathbf{y}\mathbf{W}^T\mathbf{W}) + \mathbf{x}^T\mathbf{y} - \mathbf{x}^T h(\mathbf{y}\mathbf{\Sigma}^{-1})\mathbf{\Sigma}\mathbf{W}^T\mathbf{W} \tag{168}$$
$$\mathbf{W}^T\mathbf{W} = (\alpha\mathbf{\Sigma}^2 + \mathbf{\Sigma}^2)(\alpha\mathbf{\Sigma}^2 + \mathbb{E}(\mathbf{y}^T h(\mathbf{y}\mathbf{\Sigma}^{-1}))\mathbf{\Sigma})^{-1} \tag{169}$$
$$= (\mathbf{I} + \alpha^{-1}\mathbf{I})(\mathbf{I} + \alpha^{-1}\mathbb{E}(\mathbf{y}^T h(\mathbf{y}\mathbf{\Sigma}^{-1}))\mathbf{\Sigma}^{-1})^{-1} \tag{170}$$
$$\xrightarrow[\alpha\to\infty]{} \mathbf{I} \tag{171}$$

And so if we set α large enough, we can compensate for the scale. In practice $\alpha \geq 2$ is enough to induce an effect.

F.1.2 Unmodified derivative

If the derivative of h is not modified, and assuming $h_a = a\tanh(x/a)$, $h_a'(x) = 1 - h_a^2(x)/a^2$, the stationary point yields

$$\mathbf{\Sigma}^2 = \mathbb{E}(\mathbf{y}^T h_a(\mathbf{y}\mathbf{\Sigma}^{-1})\mathbf{\Sigma}) - \frac{1}{a}\mathbb{E}((\mathbf{y}^T\mathbf{y} - \mathbf{y}^T h_a(\mathbf{y}\mathbf{\Sigma}^{-1})\mathbf{\Sigma})h_a^2(\mathbf{y}\mathbf{\Sigma}^{-1})). \tag{172}$$

We can see that the larger a is, the more the equality will be satisfied. The choice of a will generally depend on the type of distribution, which can be seen more clearly in Fig. 3.

F.2 Gradient of trainable $\boldsymbol{\Sigma}$

From the loss in Eq. 26, we have

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{\Sigma}_e} = -\text{diag}(\text{diag}^{-1}(\mathbf{W}_e^T \mathbf{x}^T (\hat{\mathbf{x}} - \mathbf{x}) \mathbf{W}_d \boldsymbol{\Sigma}_d \odot h'(\mathbf{z}) \boldsymbol{\Sigma}_e^{-2})) \tag{173}$$

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{\Sigma}_d} = \text{diag}((\hat{\mathbf{x}} - \mathbf{x}) \mathbf{W}_d \odot h(\mathbf{z})) \tag{174}$$

where diag creates a diagonal matrix from a vector and diag^{-1} takes the diagonal part of the matrix as vector. If we tie the weights but keep the contributions separate, we obtain

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{\Sigma}_e} = -\text{diag}(\text{diag}^{-1}(\mathbf{y}^T (\hat{\mathbf{y}} - \mathbf{y}) \odot h'(\mathbf{y} \boldsymbol{\Sigma}^{-1}) \boldsymbol{\Sigma}^{-1})) \tag{175}$$

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{\Sigma}_d} = \text{diag}((\hat{\mathbf{y}} - \mathbf{y}) \odot h(\mathbf{y} \boldsymbol{\Sigma}^{-1})) \tag{176}$$

Given $\boldsymbol{\Sigma} = \text{diag}(\boldsymbol{\sigma})$, we can write

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{\sigma}_e} = -\mathbf{y}(\hat{\mathbf{y}} - \mathbf{y}) h'(\mathbf{y} \boldsymbol{\sigma}^{-1}) \boldsymbol{\sigma}^{-1} \tag{177}$$

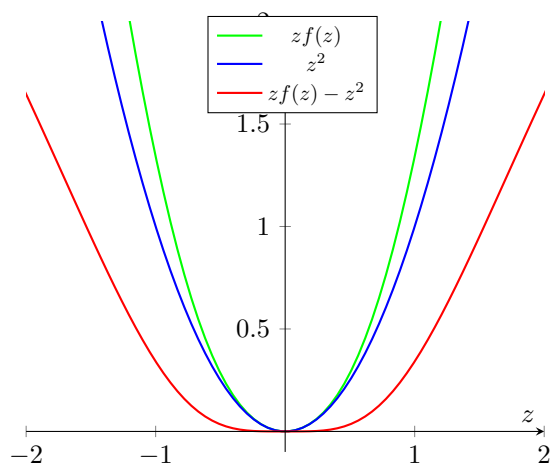
$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{\sigma}_d} = (\hat{\mathbf{y}} - \mathbf{y}) h(\mathbf{y} \boldsymbol{\sigma}^{-1}) \tag{178}$$

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{\sigma}} = -\mathbf{y}(\hat{\mathbf{y}} - \mathbf{y}) h'(\mathbf{y} \boldsymbol{\sigma}^{-1}) \boldsymbol{\sigma}^{-1} + (\hat{\mathbf{y}} - \mathbf{y}) h(\mathbf{y} \boldsymbol{\sigma}^{-1}) \tag{179}$$

Let us consider $h = \tanh$, which results in

$$\begin{aligned}
\frac{\partial \mathcal{L}}{\partial \boldsymbol{\sigma}} &= -\mathbf{y}(\hat{\mathbf{y}} - \mathbf{y})(1 - h^2(\mathbf{y} \boldsymbol{\sigma}^{-1})) \boldsymbol{\sigma}^{-1} + (\hat{\mathbf{y}} - \mathbf{y}) h(\mathbf{y} \boldsymbol{\sigma}^{-1}) && (180)\\
&= (\hat{\mathbf{y}} - \mathbf{y})(-\mathbf{y} \boldsymbol{\sigma}^{-1} + \mathbf{y} \boldsymbol{\sigma}^{-1} h^2(\mathbf{y} \boldsymbol{\sigma}^{-1}) + h(\mathbf{y} \boldsymbol{\sigma}^{-1})) && (181)\\
&= (\hat{\mathbf{y}} - \mathbf{y})(-\mathbf{y} \boldsymbol{\sigma}^{-1} + f(\mathbf{y} \boldsymbol{\sigma}^{-1})) && (182)\\
&= \boldsymbol{\sigma}(\mathbf{y}^2 \boldsymbol{\sigma}^{-2} - \mathbf{y} \boldsymbol{\sigma}^{-1} f(\mathbf{y} \boldsymbol{\sigma}^{-1}) - \hat{\mathbf{y}} \mathbf{y} \boldsymbol{\sigma}^{-2} + \hat{\mathbf{y}} \boldsymbol{\sigma}^{-1} f(\mathbf{y} \boldsymbol{\sigma}^{-1})) && (183)\\
&= \boldsymbol{\sigma}(\mathbf{z}^2 - \mathbf{z} f(\mathbf{z}) - \hat{\mathbf{z}} \mathbf{z} + \hat{\mathbf{z}} f(\mathbf{z})), && (184)
\end{aligned}$$

where $f(z) = z h^2(z) + h(z)$, $\mathbf{z} = \mathbf{y} \boldsymbol{\sigma}^{-1}$, and $\hat{\mathbf{z}} = \hat{\mathbf{y}} \boldsymbol{\sigma}^{-1}$. We can plot $f(z)z$, z^2 and $f(z)z - z^2$



Given that $\mathbf{y}$ is centred, we have $\mathbb{E}(\mathbf{y}^2) = \text{var}(\mathbf{y})$, and so if we take the expectation for the gradient, we will obtain

$$\mathbb{E}(\mathbf{z}^2\boldsymbol{\sigma}) = \mathbb{E}(\mathbf{y}^2\boldsymbol{\sigma}^{-1}) = \text{var}(\mathbf{y})\boldsymbol{\sigma}^{-1}. \tag{185}$$

For $|z| < 1$, we also have $zf(z) \approx z^2$. And so we can see that $\frac{\partial \mathcal{L}}{\partial \boldsymbol{\sigma}}$ takes the form of terms that are roughly $\propto \text{var}(\mathbf{y})\boldsymbol{\sigma}^{-1}$, but there is no guarantee that $\boldsymbol{\Sigma}$ would stop growing unless the reconstruction differences are zero.

F.3 Reconstruction loss in latent space and relationship to blind deconvolution

When performing the reconstruction in latent space, we need to add something that maintains the orthogonality of $\mathbf{W}$. If we multiply both sides of the difference in Eq. 34 by $[\mathbf{W}]_{sg}$ and include a symmetric orthogonal regulariser, we obtain

$$\mathcal{L}(\mathbf{W}, \boldsymbol{\Sigma}) = \mathbb{E}(||\mathbf{x}[\mathbf{W}]_{sg} - h(\frac{\mathbf{xW}}{\boldsymbol{\sigma}})\boldsymbol{\sigma}[\mathbf{W}^T\mathbf{W}]_{sg}||_2^2) + \beta||\mathbf{I} - \mathbf{W}^T\mathbf{W}||_F^2. \tag{186}$$

Although we can assume that $[\mathbf{W}^T\mathbf{W}]_{sg} = \mathbf{I}$ is maintained by an orthogonal regulariser, and might be tempted to simply remove it, we find that keeping it helps make it more stable, especially during the initial period of training where $\mathbf{W}$ might be transitioning via a non-orthogonal matrix. Given that $[\mathbf{W}^T\mathbf{W}]_{sg}$ is square, we can also use its lower triangular form $\text{◺}[\mathbf{W}^T\mathbf{W}]_{sg}$ to obtain an automatic ordering:

$$\mathcal{L}(\mathbf{W}, \boldsymbol{\Sigma}) = \mathbb{E}(||\mathbf{x}[\mathbf{W}]_{sg} - h(\frac{\mathbf{xW}}{\boldsymbol{\sigma}})\boldsymbol{\sigma}\text{◺}[\mathbf{W}^T\mathbf{W}]_{sg}||_2^2) + \beta||\mathbf{I} - \mathbf{W}^T\mathbf{W}||_F^2. \tag{187}$$

If indeed we do assume that $[\mathbf{W}^T\mathbf{W}]_{sg} = \mathbf{I}$, we find that it generally requires a much stronger orthogonal regulariser $\beta > 1$, with an asymmetric regulariser being better than the symmetric, and it works better when $\boldsymbol{\Sigma}$ is trainable rather than estimated from data.

$$\mathcal{L}(\mathbf{W}, \boldsymbol{\Sigma}) = \mathbb{E}(||\mathbf{x}[\mathbf{W}]_{sg} - h(\frac{\mathbf{xW}}{\boldsymbol{\sigma}})\boldsymbol{\sigma}||_2^2) + \beta||\mathbf{I} - \mathbf{W}^T\mathbf{W}||_F^2. \tag{188}$$

Instead of $\beta||\mathbf{I} - \mathbf{W}^T\mathbf{W}||_F^2$ we can also use the asymmetric version $\beta||\text{◺}\mathbf{W}^T[\mathbf{W}]_{sg}||_F^2$ (see Appendix B.2).

Alternatively, we can use the linear reconstruction loss $\mathbb{E}(||\mathbf{xW}[\mathbf{W}^T]_{sg} - \mathbf{x}||_2^2)$ for maintaining the orthogonality:

$$\mathcal{L}(\mathbf{W}, \boldsymbol{\Sigma}) = \mathbb{E}(||\mathbf{x}[\mathbf{W}]_{sg} - h(\mathbf{xW}\boldsymbol{\Sigma})\boldsymbol{\Sigma}||_2^2) + \mathbb{E}(||\mathbf{xW}[\mathbf{W}^T]_{sg} - \mathbf{x}||_2^2). \tag{189}$$

Putting this in context of blind deconvolution (Lambert, 1996; Haykin, 1996), we can note the first term in Eq. 186,

$$\mathbb{E}(||h(\frac{\mathbf{y}}{\boldsymbol{\sigma}})\boldsymbol{\sigma} - [\mathbf{y}]_{sg}||_2^2), \tag{190}$$

can be seen as a modified form of the Bussgang criterion

$$\mathbb{E}(||h(\mathbf{y}) - \mathbf{y}||_2^2. \tag{191}$$

F.4 Connection with L_1 sparsity

ICA is closely related to sparse coding (Olshausen & Field, 1996; Bell & Sejnowski, 1997). This is because maximising sparsity can be seen as a method for maximising non-Gaussianity (Hyvärinen & Oja, 2000)– which is a particular ICA method. Reconstruction ICA (RICA) (Le et al., 2011) is a method that combines L_1 sparsity with a linear reconstruction loss, and it was initially proposed for learning overcomplete sparse features, in contrast to conventional ICA which does not model overcompleteness. RICA is, in fact, simply performing nonlinear PCA: it induces a latent reconstruction term in the weight update, and, by tying the weights of the encoder and decoder, it enforces $\mathbf{W}$ to have orthogonal columns. RICA is thus a special case of the more general ICA model which can learn an arbitrary transformation rather than just an orthogonal transformation. The RICA loss is

$$\mathcal{L}(\mathbf{W}) = ||\mathbf{x} - \mathbf{x}\mathbf{W}\mathbf{W}^T||_2^2 + \beta \sum |\mathbf{x}\mathbf{w_j^T}|, \tag{192}$$

where $\mathbf{w}_j^T$ is a column of $\mathbf{W}$. This works equally well if we use the subspace variant, i.e.:

$$\mathcal{L}(\mathbf{W}) = ||\mathbf{x} - \mathbf{x}[\mathbf{W}]_{sg}\mathbf{W}^T||_2^2 + \beta \sum |\mathbf{x}\mathbf{w_j^T}|, \tag{193}$$

If we compute the gradient contributions from Eq. 193, we obtain

$$\frac{\partial \mathcal{L}}{\partial \mathbf{W}_e} = \beta \mathbf{x}^T \text{sign}(\mathbf{y}) \tag{194}$$

$$\frac{\partial \mathcal{L}}{\partial \mathbf{W}_d} = (\hat{\mathbf{x}} - \mathbf{x})^T \mathbf{y}, \tag{195}$$

which results in the combined gradient

$$\frac{\partial \mathcal{L}}{\partial \mathbf{W}} = (\hat{\mathbf{x}} - \mathbf{x})^T \mathbf{y} + \beta \mathbf{x}^T \text{sign}(\mathbf{y}) \tag{196}$$

$$= \hat{\mathbf{x}}^T \mathbf{y} + \mathbf{x}^T(\beta \text{sign}(\mathbf{y}) - \mathbf{y}). \tag{197}$$

The main relevant part in Eq. 197 is

$$\mathbf{x}^T(\beta \text{sign}(\mathbf{y}) - \mathbf{y}), \tag{198}$$

and this is similar to the latent reconstruction term in the nonlinear PCA gradient, except we now have a dependency on the value of β. No suggestion for the best value of β is given by the authors (Le et al., 2011); however, previous works on sparse coding (Olshausen & Field, 1996) have set β to be proportional to the standard deviation of the input. We can perhaps gain some insight why by looking more closely at the influence on Eq. 197 of the norm of the input. For that we extract $||\mathbf{x}||_2$ to get

$$\beta \text{sign}(\mathbf{y}) - \mathbf{y} = ||\mathbf{x}||_2(\beta \frac{\text{sign}(\mathbf{y})}{||\mathbf{x}||_2} - \frac{\mathbf{x}\mathbf{W}}{||\mathbf{x}||_2}). \tag{199}$$

We now see that if $||\mathbf{x}||_2$ increases, then the norm of $\frac{\text{sign}(\mathbf{y})}{||\mathbf{x}||_2}$ will decrease, while that of $\frac{\mathbf{x}\mathbf{W}}{||\mathbf{x}||_2}$ will remain constant. Indeed we have

$$\frac{||\text{sign}(\mathbf{y})||_2}{||\mathbf{x}||_2} \leq \frac{\sqrt{k}}{||\mathbf{x}||_2} \xrightarrow[||\mathbf{x}||_2 \to \infty]{} 0 \tag{200}$$

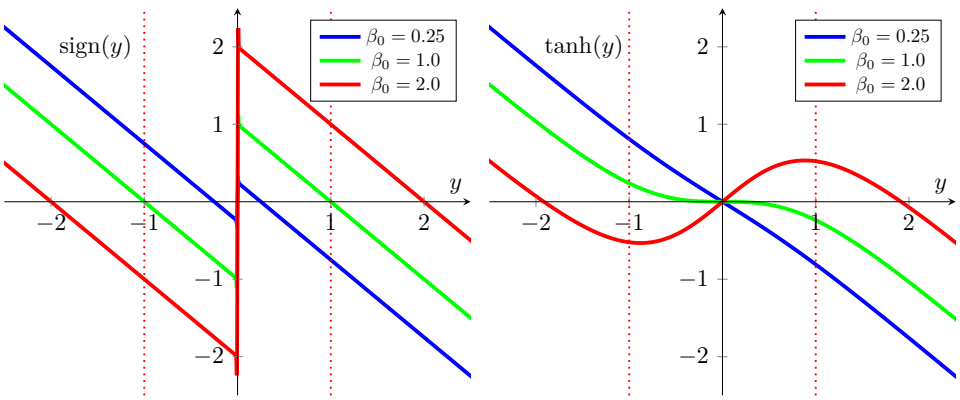
and

$$\frac{||\mathbf{xW}||_2}{||\mathbf{x}||_2} \approx 1, \tag{201}$$

because, from the reconstruction term, we have

$$||\mathbf{xW}||_2^2 = \mathbf{xWW}^T\mathbf{x}^T = \hat{\mathbf{x}}\mathbf{x}^T \approx \mathbf{xx}^T = ||\mathbf{x}||_2^2. \tag{202}$$

And so to make the relative difference $\beta\text{sign}(\mathbf{y}) - \mathbf{y}$ invariant to the norm of the input, we can set β to be proportional to it, or more simply proportional to the standard deviation of the input as it is already centred, i.e. $\beta = \beta_0\mathbb{E}(||\mathbf{x}||_2)$. As to what the value of β_0 should be, we can plot the functions $f_{\beta_0}(y) = \beta_0\text{sign}(y) - y$ and $f_{\beta_0}(y) = \beta_0 \tanh(y) - y$:



For tanh, if $\beta_0 \leq 1$, then it has a single inflection point, while if $\beta_0 > 1$, then it has two inflection points. A reasonable range for β_0 appears to be $]0.0, 1.0]$. This range is in line with (Olshausen & Field, 1996) where β_0 was set to 0.14.

The additional term in the gradient does not maintain unit norm exactly; this is because, if we follow a similar analysis as in Appendix A.3.2, we have for $\Delta\mathbf{W} = \mathbf{0}$

$$\mathbf{W}^T\mathbf{W} = \mathbf{I} - \beta\mathbb{E}(\mathbf{y}^T\text{sign}(\mathbf{y}))\mathbf{\Sigma}^{-2}, \tag{203}$$

if using the subspace RICA loss, or

$$\mathbf{W}^T\mathbf{W} = \mathbf{I} - \frac{\beta}{2}\mathbb{E}(\mathbf{y}^T\text{sign}(\mathbf{y}))\mathbf{\Sigma}^{-2}, \tag{204}$$

if using the original RICA loss. Therefore, the norm of the columns is less than one.

Another thing we can note is that $\lim_{a\to\infty} \tanh(ax) = \text{sign}(x)$, so in Eq. 197 we could potentially replace sign with tanh to get

$$\frac{\partial\mathcal{L}}{\partial\mathbf{W}} = \hat{\mathbf{x}}^T\mathbf{y} + \mathbf{x}^T(\beta\tanh(\mathbf{y}) - \mathbf{y}). \tag{205}$$

We know that tanh is the derivative of log cosh, which is none other than the function used by FastICA for the negentropy approximation (Hyvarinen, 1999). This can be expressed as the following loss function:

$$\mathcal{L}(\mathbf{W}) = ||\mathbf{x} - \mathbf{x}[\mathbf{W}]_{sg}\mathbf{W}^T||_2^2 + \beta\sum\log\cosh(\mathbf{xw_j^T}). \tag{206}$$

F.5 Skew-symmetric update

Similar to the derivation of the EASI algorithm (see Appendix D.2), we can combine the gradient obtained from a linear reconstruction loss with the term $\mathbf{W}\mathbf{y}^T h(\mathbf{y})$. However, to avoid a similar problem as in the previous section of the gradient update not maintaining unit norm columns, we can simply remove the diagonal part of $\mathbf{y}^T h(\mathbf{y})$, resulting in the weight update

$$\Delta\mathbf{W} \propto \mathbf{x}^T\mathbf{y} - \mathbf{W}\mathbf{y}^T\mathbf{y} - \beta\mathbf{W}(\mathbf{y}^T h(\mathbf{y}) - \text{diag}(h(\mathbf{y}) \odot \mathbf{y})). \tag{207}$$

Or, similar to EASI, we can use it in its skew-symmetric form:

$$\Delta\mathbf{W} \propto \mathbf{x}^T\mathbf{y} - \mathbf{W}\mathbf{y}^T\mathbf{y} - \beta\mathbf{W}(\mathbf{y}^T h(\mathbf{y}) - h(\mathbf{y})^T\mathbf{y})), \tag{208}$$

which we can also write as the loss function

$$\mathcal{L}(\mathbf{W}) = \frac{1}{2}||\mathbf{x} - \mathbf{x}[\mathbf{W}]_{sg}\mathbf{W}^T||_2^2 + \beta\mathbf{1}\mathbf{W} \odot [\mathbf{W}(\mathbf{y}^T h(\mathbf{y}) - h(\mathbf{y})^T\mathbf{y})))]_{sg}\mathbf{1}^T. \tag{209}$$

As justified by the previous section, we can set $\beta = \mathbb{E}(||\mathbf{x}||_2)$ or we can simply use $\mathbf{\Sigma}$ to compensate:

$$\Delta\mathbf{W} \propto \mathbf{x}^T\mathbf{y} - \mathbf{W}\mathbf{y}^T\mathbf{y} - \mathbf{W}(\mathbf{y}^T h(\mathbf{y}\mathbf{\Sigma}^{-1}) - \text{diag}(\mathbf{y} \odot h(\mathbf{y}\mathbf{\Sigma}^{-1}))\mathbf{\Sigma}. \tag{210}$$

In all of these variants, we can have $h = \text{sign}$ or $h = \tanh$.

F.6 Modified decoder contribution

The gradient updates of the encoder and decoder (as derived in 3.1) are

$$\frac{\partial\mathcal{L}}{\partial\mathbf{W}_e} = \mathbf{x}^T(\hat{\mathbf{y}} - \mathbf{y}) \odot h'(\mathbf{z}) \tag{211}$$

$$\frac{\partial\mathcal{L}}{\partial\mathbf{W}_d} = (\hat{\mathbf{x}} - \mathbf{x})^T h(\mathbf{z})\mathbf{\Sigma}. \tag{212}$$

The main issue is that, when both are used in the tied case, the decoder contribution overpowers the encoder contribution. This is evident in the linear case, where, when $\mathbf{W}$ is semi-orthogonal, the encoder contribution is zero. Indeed, let $\mathbf{W} \in \mathbb{R}^{p\times k}$ with $p > k$, recall that in the linear case (Appendix A.2.1) we have

$$\frac{\partial\mathcal{L}}{\partial\mathbf{W}_e} = \mathbf{x}^T\mathbf{y}(\mathbf{W}^T\mathbf{W} - \mathbf{I}) \tag{213}$$

$$\frac{\partial\mathcal{L}}{\partial\mathbf{W}_d} = (\mathbf{W}\mathbf{W}^T - \mathbf{I})\mathbf{x}^T\mathbf{y} \tag{214}$$

If we consider their Frobineus norms, we can write

$$||\frac{\partial\mathcal{L}}{\partial\mathbf{W}_e}||_F = 0 \tag{215}$$

$$\lambda_{\min}(\mathbf{x}^T\mathbf{y})\sqrt{p-k} \leq ||\frac{\partial\mathcal{L}}{\partial\mathbf{W}_d}||_F, \tag{216}$$

where the lower bound on the decoder contribution can be derived from the fact that for $\mathbf{A} \in \mathbb{R}^{p\times k}$ and $\mathbf{B} \in \mathbb{R}^{k\times n}$, we have (Fang et al., 1994)

$$\lambda_{\min}(\mathbf{A})\|\mathbf{B}\|_F \leq \|\mathbf{AB}\|_F \leq \lambda_{\max}(\mathbf{A})\|\mathbf{B}\|_F, \tag{217}$$

where $\lambda_{\min}(\mathbf{A})$ and $\lambda_{\max}(\mathbf{A})$ refer to the smallest and largest eigenvalues of $\mathbf{A}$, respectively, and

$$||\mathbf{WW}^T - \mathbf{I}||_F^2 = tr((\mathbf{WW}^T - \mathbf{I})^T(\mathbf{WW}^T - \mathbf{I})) \tag{218}$$

$$= tr(\mathbf{WW}^T\mathbf{WW}^T - 2\mathbf{WW}^T + \mathbf{I}) \tag{219}$$

$$= tr(\mathbf{I} - \mathbf{WW}^T) = p - k. \tag{220}$$

In the nonlinear case, the gradient updates are approximately close to the linear update (especially if $h(z) = a\tanh(z/a)$ with $a > 3$), except that the encoder contribution is no longer zero, and it is still overpowered by the decoder contribution. And so if omit the latter, we can better see the effect of the former.

An alternative would be to modify the relative scale so that the decoder contribution does not overpower the encoder contribution. The variance of the components plays a role in this, given that this is not a problem in the case of conventional nonlinear PCA with whitened input, where all the components have unit variance. As a This suggests three options: (1) scale the encoder contribution by $\mathbf{\Sigma}$; (2) drop $\mathbf{\Sigma}$ from the decoder contribution and, optionally, $h'(\mathbf{z})$ from the encoder contribution; (3) scale down the decoder contribution by a constant, which can simply be the inverse of the spectral, Frobineus, or nuclear norm of $\mathbf{\Sigma}$.

For the third option we can write it as a loss:

$$\mathcal{L}(\mathbf{W}) = \mathbb{E}(||\mathbf{x} - h(\mathbf{xW\Sigma}^{-1})\mathbf{\Sigma}[\mathbf{W}^T]_{sg}||_2^2 + \frac{1}{||\mathbf{\Sigma}||_2}||\mathbf{x} - [h(\mathbf{xW\Sigma}^{-1})\mathbf{\Sigma}]_{sg}\mathbf{W}^T||_2^2). \tag{221}$$

F.7 Ordering the components based on index position

Here we look at a few ways for ordering the components automatically based on index position.

F.7.1 Regulariser

One way to do this is via a Gram-Schmidt-like regulariser (Wang et al., 1995) by adding

$$J(\mathbf{W}) = ||\text{⊴}(\mathbf{W}^T[\mathbf{W}]_{sg})||_F^2, \tag{222}$$

where ⊴ refers to the lower triangular matrix without the diagonal element.

F.7.2 Triangular weight update

In the GHA (see Appendix A.3.3), we can order the components by index position by taking the lower (or upper) triangular part of the term $\mathbf{Wy}^T\mathbf{y}$, a term which originates from the linear decoder contribution. Similarly to the GHA, we can also include a triangular term taken from the nonlinear decoder contribution, which is

$$\mathbf{W}(h(\mathbf{z})\mathbf{\Sigma})^T h(\mathbf{z})\mathbf{\Sigma}, \tag{223}$$

where $\mathbf{z} = \mathbf{y\Sigma}^{-1}$. However, as we have seen in Section F.6, we need to scale this term appropriately lest it overpowers the encoder contribution. We can consider variations where we drop $\mathbf{\Sigma}$ from either left or right, or make the approximation of $y \approx h(\mathbf{z})\mathbf{\Sigma}$:

$$\mathbf{W}\triangleleft\ ((h(\mathbf{z})\mathbf{\Sigma})^T h(\mathbf{z})) \tag{224}$$
$$\text{or } \mathbf{W}\triangleleft\ (h(\mathbf{z})^T h(\mathbf{z})\mathbf{\Sigma}) \tag{225}$$
$$\text{or } \mathbf{W}\triangleleft\ (h(\mathbf{z})^T \mathbf{y}) \tag{226}$$
$$\text{or } \mathbf{W}\triangleleft\ (\mathbf{y}^T h(\mathbf{z})) \tag{227}$$
$$\text{or } \mathbf{W}\triangleleft\ (\mathbf{z}^T \mathbf{y}) \tag{228}$$
$$\text{or } \mathbf{W}\triangleleft\ (\mathbf{y}^T \mathbf{z}) \tag{229}$$

If $\mathbf{\Sigma}$ is non-trainable and $h = \tanh$, then $\mathbf{\Sigma} = a\hat{\mathbf{\Sigma}}$ where $a \geq 3$ and $\hat{\mathbf{\Sigma}}$ is the estimated standard deviation of $\mathbf{y}$.

F.7.3 Weighted latent reconstruction

Another option is similar to the weighted subspace algorithm (see A.3.2) where we insert a linearly-spaced $\mathbf{\Lambda}$ into the weight update derived in Eq. 158:

$$\Delta\mathbf{W} \propto \mathbf{x}^T\mathbf{y}\mathbf{\Lambda} - \mathbf{x}^T h(\mathbf{y}\mathbf{\Sigma}^{-1})\mathbf{\Sigma}\mathbf{\Lambda}\mathbf{W}^T\mathbf{W}, \tag{230}$$

where $\mathbf{\Lambda} = \text{diag}(\lambda_1, ..., \lambda_k)$ such that $1 \geq \lambda_1 > ... > \lambda_k > 0$.

As loss function we can have:

$$\mathcal{L}(\mathbf{W}) = \mathbb{E}(||[\mathbf{y}]_{sg}\mathbf{\Lambda} - h(\mathbf{y}\mathbf{\Sigma}^{-1})\mathbf{\Sigma}\mathbf{\Lambda}[\mathbf{W}^T\mathbf{W}]_{sg}||_2^2). \tag{231}$$

F.7.4 Embedded projective deflation

Let us consider the loss

$$\mathcal{L}(\mathbf{W}_e, \mathbf{W}_d, \mathbf{\Sigma}_e, \mathbf{\Sigma}_d) = \frac{1}{2}||\mathbf{x} - h(\mathbf{x}\mathbf{W}_{e1}P(\mathbf{W}_e)\mathbf{\Sigma}_e^{-1})\mathbf{\Sigma}_d\mathbf{W}_d^T||_2^2, \tag{232}$$

where

$$P(\mathbf{W}_e) = \alpha\mathbf{I} + \beta\mathbf{W}_{e2}^T\mathbf{W}_{e3}, \tag{233}$$

and $\mathbf{W}_e = \mathbf{W}_{e1} = \mathbf{W}_{e2} = \mathbf{W}_{e3}$ for elucidating the contribution of each part to the gradient. Taking the gradient of the loss with respect to the weights, we have

$$\frac{\partial\mathcal{L}}{\partial\mathbf{W}_{e1}} = \mathbf{x}^T(((\hat{\mathbf{x}} - \mathbf{x})\mathbf{W}_d) \odot h'(\mathbf{y}\mathbf{\Sigma}^{-1}))P(\mathbf{W}_e) \tag{234}$$
$$\frac{\partial\mathcal{L}}{\partial\mathbf{W}_{e2}} = \beta\mathbf{W}_e((\hat{\mathbf{x}} - \mathbf{x})\mathbf{W}_d \odot h'(\mathbf{y}\mathbf{\Sigma}^{-1}))^T\mathbf{y} \tag{235}$$
$$\frac{\partial\mathcal{L}}{\partial\mathbf{W}_{e3}} = \beta\mathbf{W}_e\mathbf{y}^T((\hat{\mathbf{x}} - \mathbf{x})\mathbf{W}_d \odot h'(\mathbf{y}\mathbf{\Sigma}^{-1})), \tag{236}$$

Setting $\mathbf{W}_e = \mathbf{W}_d = \mathbf{W}$ and $\mathbf{\Sigma}_e = \mathbf{\Sigma}_d = \mathbf{\Sigma}$, we obtain

$$\frac{\partial \mathcal{L}}{\partial \mathbf{W}_{e1}} = \mathbf{x}^T((\hat{\mathbf{y}} - \mathbf{y}) \odot h'(\mathbf{y}\mathbf{\Sigma}^{-1}))P(\mathbf{W}) \tag{237}$$

$$\frac{\partial \mathcal{L}}{\partial \mathbf{W}_{e2}} = \beta\mathbf{W}((\hat{\mathbf{y}} - \mathbf{y}) \odot h'(\mathbf{y}\mathbf{\Sigma}^{-1}))^T\mathbf{y} \tag{238}$$

$$\frac{\partial \mathcal{L}}{\partial \mathbf{W}_{e3}} = \beta\mathbf{W}\mathbf{y}^T((\hat{\mathbf{y}} - \mathbf{y}) \odot h'(\mathbf{y}\mathbf{\Sigma}^{-1})) \tag{239}$$

Let $\mathbf{y}_\delta = (\hat{\mathbf{y}} - \mathbf{y}) \odot h'(\mathbf{y}\mathbf{\Sigma}^{-1})$, then we can write

$$\frac{\partial \mathcal{L}}{\partial \mathbf{W}_{e1}} = \mathbf{x}^T\mathbf{y}_\delta P(\mathbf{W}) \tag{240}$$

$$\frac{\partial \mathcal{L}}{\partial \mathbf{W}_{e2}} = \beta\mathbf{W}\mathbf{y}_\delta^T\mathbf{y} \tag{241}$$

$$\frac{\partial \mathcal{L}}{\partial \mathbf{W}_{e3}} = \beta\mathbf{W}\mathbf{y}^T\mathbf{y}_\delta, \tag{242}$$

which results in the overall gradient

$$\frac{\partial \mathcal{L}}{\partial \mathbf{W}_{e}} = \mathbf{x}^T\mathbf{y}_\delta P(\mathbf{W}) + \beta\mathbf{W}(\mathbf{y}_\delta^T\mathbf{y} + \mathbf{y}^T\mathbf{y}_\delta). \tag{243}$$

If we set $P(\mathbf{W}) = \mathbf{I} - \triangle\mathbf{W}^T[\mathbf{W}]_{sg}$, we obtain

$$\frac{\partial \mathcal{L}}{\partial \mathbf{W}_{e}} = \mathbf{x}^T\mathbf{y}_\delta P(\mathbf{W}) - \mathbf{W}\overline{\triangle}\,(\mathbf{y}_\delta^T\mathbf{y}). \tag{244}$$

This introduces an asymmetric term $\mathbf{W}\overline{\triangle}\,(\mathbf{y}_\delta^T\mathbf{y})$, similar to GHA, resulting in an ordering of components by index position.

F.7.5 Nested dropout

Similarly to the linear case (Appendix A.3.5), let $\mathbf{m}_{1|j} \in \{0,1\}^k$ be a vector such that $m_i = \begin{cases} 1 & \text{if } i < j \\ 0 & \text{otherwise} \end{cases}$, then we can order components using the loss

$$\mathcal{L}(\mathbf{W}) = \mathbb{E}(||\mathbf{x} - h(\mathbf{x}\mathbf{W}\mathbf{\Sigma}^{-1}) \odot \mathbf{m}_{1|j}\mathbf{\Sigma}[\mathbf{W}^T]_{sg}||_2^2 + ||\mathbf{I} - \mathbf{W}^T\mathbf{W}||_F^2. \tag{245}$$

One thing to note about nested dropout is that the larger the index, the less frequently the component receives a gradient update, so training has to run for longer. It can also benefit from the addition of an orthogonal regulariser to provide a gradient signal to the components receiving less frequent updates from the reconstruction.

F.8 Non-centred nonlinear PCA

When using a zero-centred function like tanh, its input should also be zero-centred, so, to obtain a non-centred version of nonlinear PCA, we can simply subtract the mean, and then add it back after passing through the nonlinear function:

$$\mathcal{L}(\mathbf{W}, \boldsymbol{\sigma}) = \mathbb{E}(||\mathbf{x} - (h(\frac{\mathbf{x}\mathbf{W} - \bar{\boldsymbol{\mu}}_y}{\boldsymbol{\sigma}})\boldsymbol{\sigma} + \bar{\boldsymbol{\mu}}_y)[\mathbf{W}^T]_{sg}||_2^2). \tag{246}$$

We are simply standardising (or normalising to zero mean and unit variance) the components before the nonlinearity and then undoing the standardisation after the nonlinearity. This could also be seen as applying a batch normalisation layer (Ioffe & Szegedy, 2015) and then undoing it after the nonlinearity.

We can find an upper bound of the loss by writing

$$\mathcal{L}(\mathbf{W}, \boldsymbol{\sigma}) = \mathbb{E}(||\mathbf{x} - (h(\frac{\mathbf{x}\mathbf{W} - \bar{\boldsymbol{\mu}}_\mathbf{y}}{\boldsymbol{\sigma}})\boldsymbol{\sigma} + \bar{\boldsymbol{\mu}}_\mathbf{y})[\mathbf{W}^T]_{sg}||_2^2) \tag{247}$$

$$= \mathbb{E}(||\mathbf{x} - \bar{\boldsymbol{\mu}}_\mathbf{x} + \bar{\boldsymbol{\mu}}_\mathbf{x} - (h(\frac{\mathbf{x}\mathbf{W} - \bar{\boldsymbol{\mu}}_\mathbf{y}}{\boldsymbol{\sigma}})\boldsymbol{\sigma} + \bar{\boldsymbol{\mu}}_\mathbf{y})[\mathbf{W}^T]_{sg}||_2^2) \tag{248}$$

$$\leq \mathbb{E}(||\mathbf{x} - \bar{\boldsymbol{\mu}}_\mathbf{x} - (h(\frac{\mathbf{x}\mathbf{W} - \bar{\boldsymbol{\mu}}_\mathbf{y}}{\boldsymbol{\sigma}})\boldsymbol{\sigma})[\mathbf{W}^T]_{sg}||_2^2) + \mathbb{E}(||\bar{\boldsymbol{\mu}}_\mathbf{x} - \bar{\boldsymbol{\mu}}_\mathbf{y}[\mathbf{W}^T]_{sg}||_2^2). \tag{249}$$

The upper bound consists of the centred nonlinear PCA loss and a linear mean reconstruction loss. Due to the stop gradient operator, the latter has no effect when $\mathbf{W}$ is orthogonal (see Appendix A.2.1). We can instead change it to

$$\mathbb{E}(||\bar{\boldsymbol{\mu}}_\mathbf{x} - \bar{\boldsymbol{\mu}}_\mathbf{x}\mathbf{W}\mathbf{W}^T||_2^2), \tag{250}$$

resulting into the following non-centred loss:

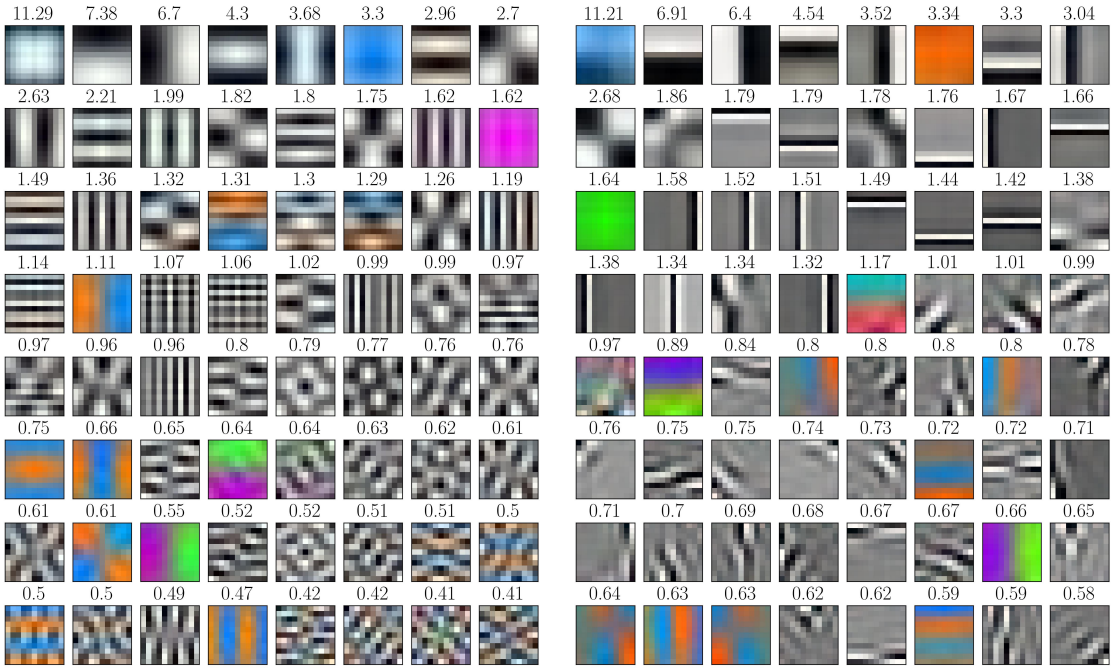
$$\mathcal{L}(\mathbf{W}, \boldsymbol{\sigma}) = \mathbb{E}(||\mathbf{x} - \bar{\boldsymbol{\mu}}_\mathbf{x} - (h(\frac{\mathbf{x}\mathbf{W} - \bar{\boldsymbol{\mu}}_\mathbf{y}}{\boldsymbol{\sigma}})\boldsymbol{\sigma})[\mathbf{W}^T]_{sg}||_2^2) + \mathbb{E}(||\bar{\boldsymbol{\mu}}_\mathbf{x} - \bar{\boldsymbol{\mu}}_\mathbf{x}\mathbf{W}\mathbf{W}^T||_2^2). \tag{251}$$

G Additional Experiments

G.1 Nonlinear PCA

G.1.1 Resolving linear PCA rotational indeterminacy

Here we attempt to recover the rotational indeterminacy $\mathbf{R}$ from $\mathbf{W}_L = \mathbf{W}_N\mathbf{R}$. For that we use gradient descent to recover $\mathbf{R}$ using the loss $||\mathbf{W}_L - \mathbf{W}_N\mathbf{R}||_2^2 + ||\mathbf{R}^T\mathbf{R} - \mathbf{I}||$. Figure 7 displays the first 64 linear and nonlinear PCA filters, and Fig. 8 shows the matches for a given linear PCA filter where the corresponding value in $|R_{ij}| > 0.2$.



(a) Linear PCA

(b) Nonlinear PCA

Figure 7: PCA filters ordered by their standard deviations

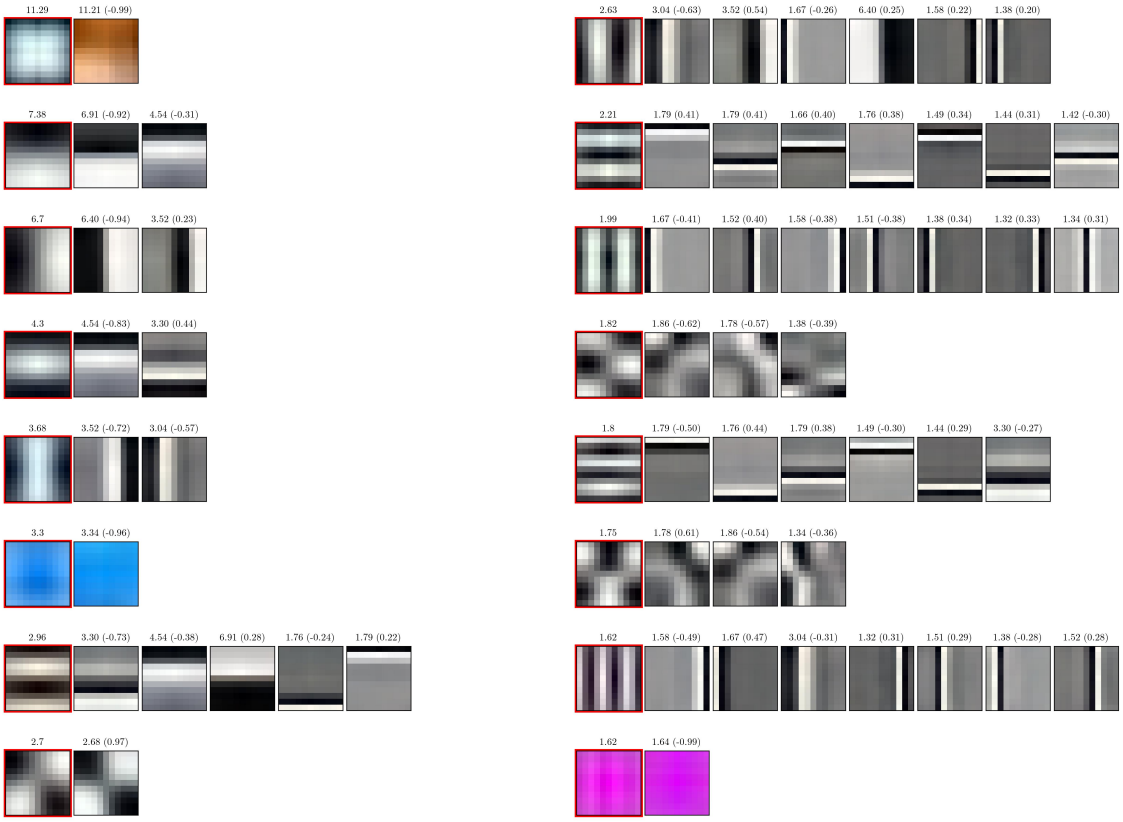
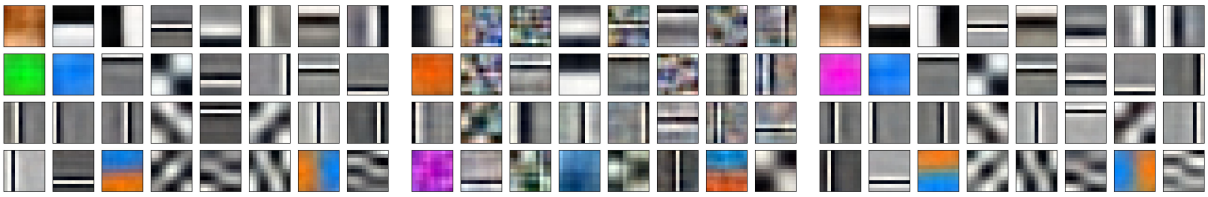


Figure 8: The filters with the red bounding box are the top 16 linear PCA filters. The filters adjacent to each one of the filters in red are the nonlinear PCA filters that could be used as linear combination to generate the linear PCA filters. We can note that the filters generally have similar variances, but there are a few outliers which are undoubtedly due to the unconstrained fitting of the rotational indeterminacy matrix $\mathbf{R}$.

G.1.2 Trainable Σ



(a) Epoch 3 - without regularisation (b) Epoch 10 - without regularisation (c) Epoch 10 - with $1e^{-3}$ L_2 regularisation

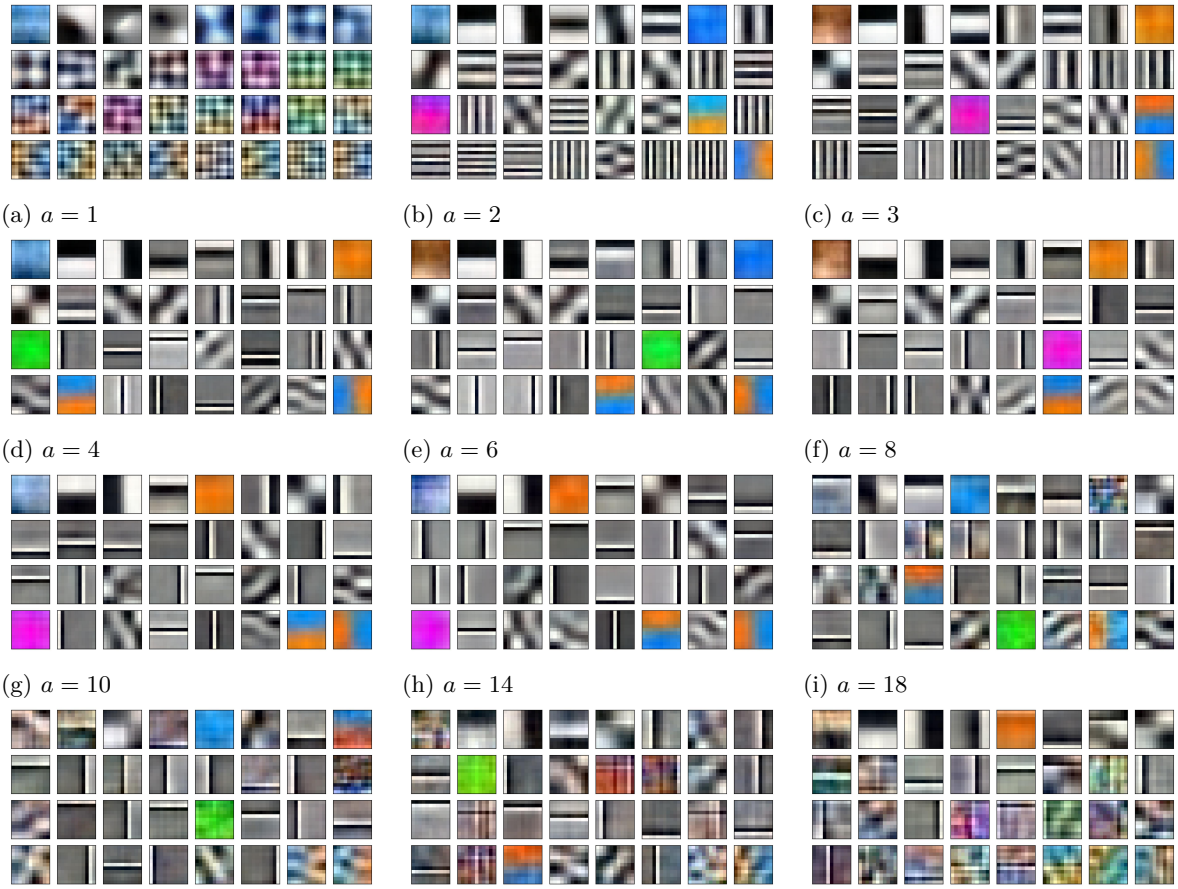
Figure 9: Without any regularisation, σ will tend to keep increasing and result in the degeneration of the filters. This does no occur when using the estimated standard deviations for σ.

G.1.3 Scaling of the input

(a) 0.01$\mathbf{x}$ (b) 100$\mathbf{x}$ (c) 1000$\mathbf{x}$

Figure 10: When using nonlinear PCA with estimated standard deviations, it automatically adapts to the scale of the input.

G.1.4 Scale of tanh



(a) $a = 1$ (b) $a = 2$ (c) $a = 3$

(d) $a = 4$ (e) $a = 6$ (f) $a = 8$

(g) $a = 10$ (h) $a = 14$ (i) $a = 18$

(j) $a = 22$ (k) $a = 26$ (l) $a = 30$

Figure 11: Showing the effect of varying the scalar a in $a \tanh(x/a)$ on the obtained filters. We see that we start getting better defined filters from $a = 2$, but quite a few are still entangled/superimposed. We get better separation from at least $a = 3$. The filters start degenerating $a > 14$.

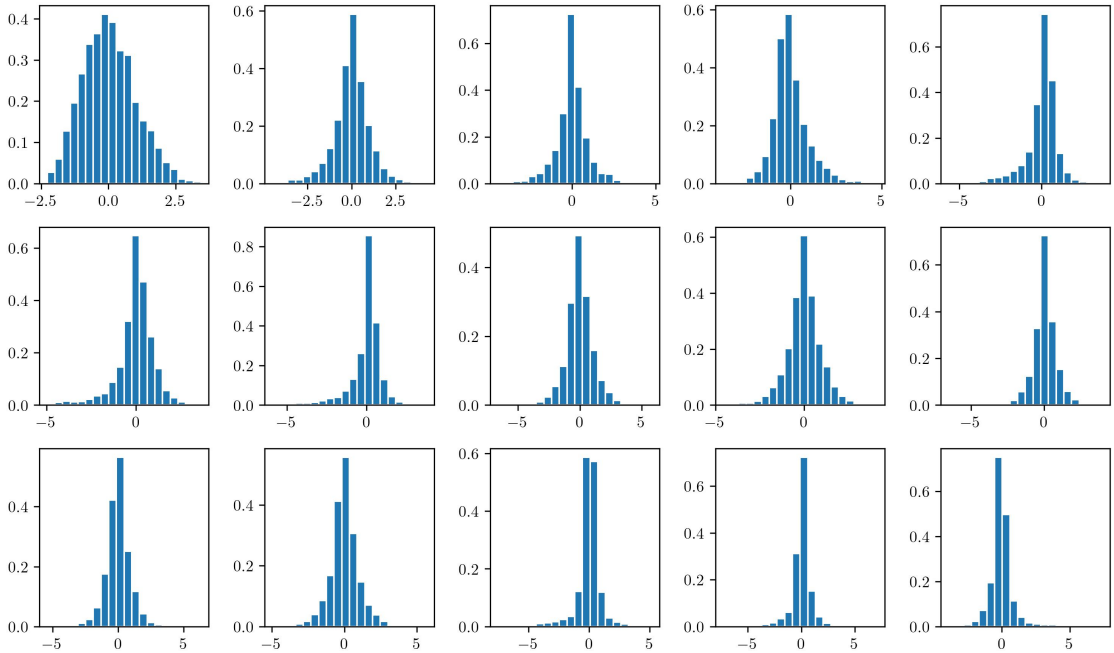


Figure 12: The distributions of the first 15 filters ordered by their variances. We see that they mostly tend to be super-Gaussian distributions, which is why a larger value of a in $a \tanh(x/a)$ was needed.

G.1.5 Latent space reconstruction

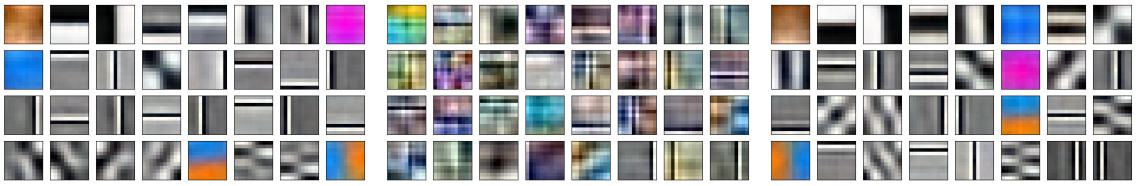
Here we look at the reconstruction loss in latent space and look the filters obtained using the losses described in Section F.3, which we repeat here for convenience:

$$\mathcal{L}(\mathbf{W}, \mathbf{\Sigma}) = \mathbb{E}(||\mathbf{x}[\mathbf{W}]_{sg} - h(\frac{\mathbf{xW}}{\boldsymbol{\sigma}})\boldsymbol{\sigma}[\mathbf{W}^T\mathbf{W}]_{sg}||_2^2) + \beta||\mathbf{I} - \mathbf{W}^T\mathbf{W}||_F^2 \tag{186}$$

$$\mathcal{L}(\mathbf{W}, \mathbf{\Sigma}) = \mathbb{E}(||\mathbf{x}[\mathbf{W}]_{sg} - h(\frac{\mathbf{xW}}{\boldsymbol{\sigma}})\boldsymbol{\sigma}\triangle[\mathbf{W}^T\mathbf{W}]_{sg}||_2^2) + \beta||\mathbf{I} - \mathbf{W}^T\mathbf{W}||_F^2 \tag{187}$$

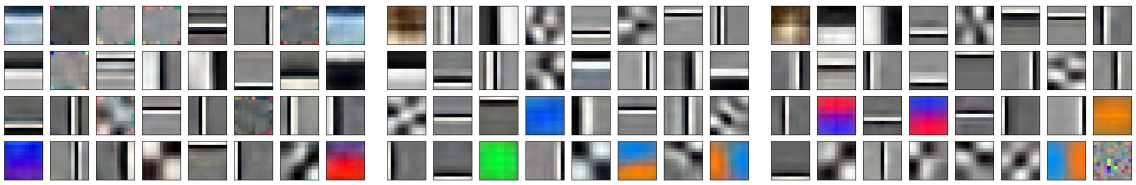
$$\mathcal{L}(\mathbf{W}, \mathbf{\Sigma}) = \mathbb{E}(||\mathbf{x}[\mathbf{W}]_{sg} - h(\frac{\mathbf{xW}}{\boldsymbol{\sigma}})\boldsymbol{\sigma}||_2^2) + \beta||\mathbf{I} - \mathbf{W}^T\mathbf{W}||_F^2. \tag{188}$$

$$\mathcal{L}(\mathbf{W}, \mathbf{\Sigma}) = \mathbb{E}(||\mathbf{x}[\mathbf{W}]_{sg} - h(\mathbf{xW\Sigma})\mathbf{\Sigma}||_2^2) + \mathbb{E}(||\mathbf{xW}[\mathbf{W}^T]_{sg} - \mathbf{x}||_2^2). \tag{189}$$



(a) $\beta = 1$ (b) $\beta = 0$ (c) $\beta = 1, \triangle[\mathbf{W}^T\mathbf{W}]_{sg}$

Figure 13: Using Eq. 186, we see that we do not require a strong gain on the orthogonal regulariser.



(a) $\beta = 1$ (b) $\beta = 10$ (c) $\beta = 100$

Figure 14: Using Eq. 188, we see that it benefits from having a larger gain on the orthogonal regulariser.

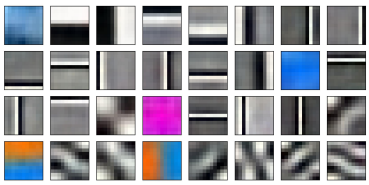
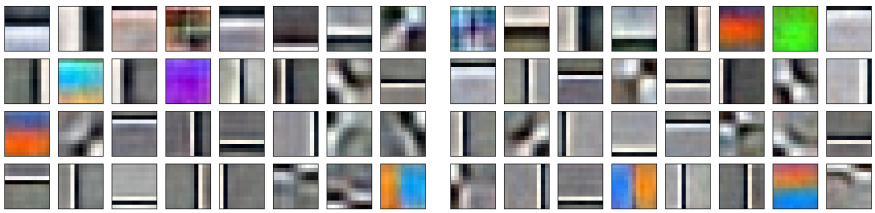


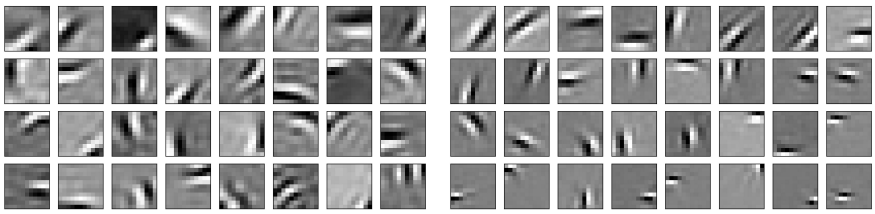
Figure 15: Using Eq. 189, we see that adding the encoder contribution from the linear reconstruction maintains the orthogonality of the components.

G.1.6 Asymmetric nonlinear function with modified derivative



(a) CIFAR-10
$h(z) = 1.6\tanh(z)$
$h'(z) = 1$

(b) CIFAR-10
$h(z) = \tanh(z)/\sqrt{\mathrm{var}(\tanh(z))}$
$h'(z) = 1$

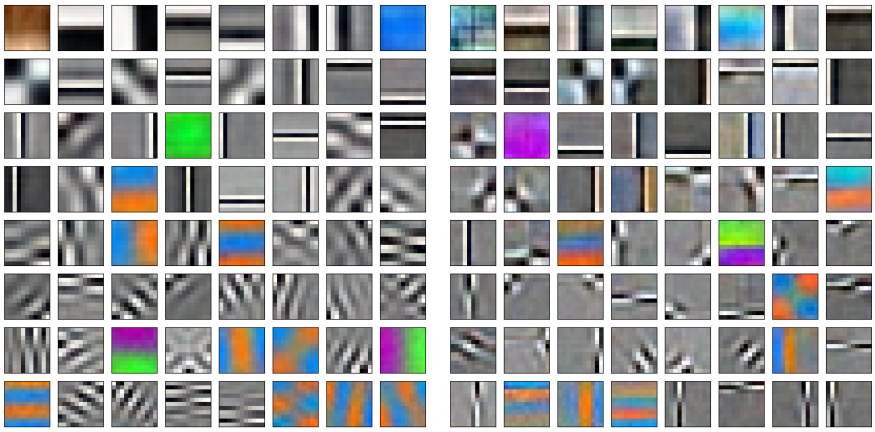


(c) MNIST
$h(z) = 1.6\tanh(z)$
$h'(z) = 1$

(d) MNIST
$h(z) = \tanh(z)/\sqrt{\mathrm{var}(\tanh(z))}$
$h'(z) = 1$

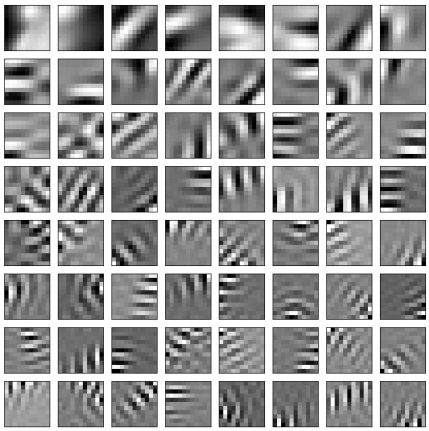
Figure 16: Filters obtained on CIFAR-10 and MNIST with an asymmetric activation function where a is either a constant or adaptive.

G.1.7 First 64 filters

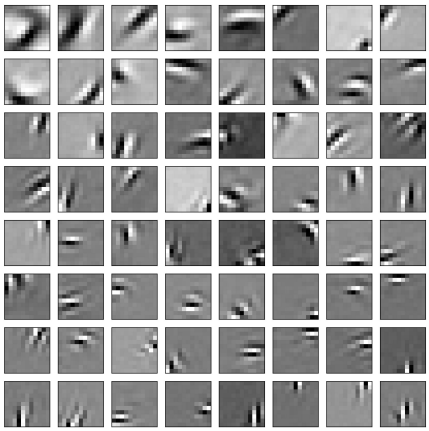


(a) CIFAR-10 - symmetric

(b) CIFAR-10 - asymmetric



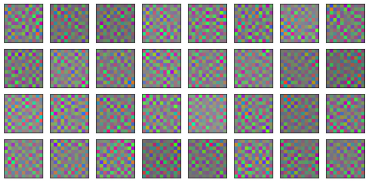
(c) MNIST - symmetric



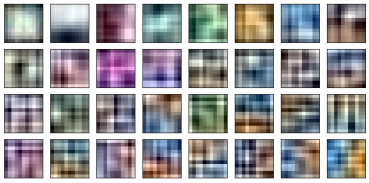
(d) MNIST - asymmetric

Figure 17: The first 64 filters, sorted by variance, obtained on CIFAR-10 and MNIST with either a symmetric activation function ($h(z) = 4\tanh(z/4)$) or an adapative asymmetric activation function ($h(z) = \tanh(z)/\sqrt{\mathrm{var}(\tanh(z))}, h'(z) = 1$). We can note that the obtained filters seem to be more localised in the asymmetric case compared to the symmetric case.

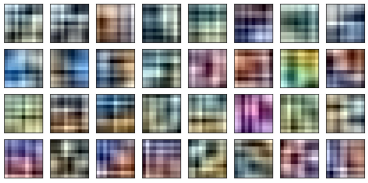
G.2 L_1 sparsity - RICA



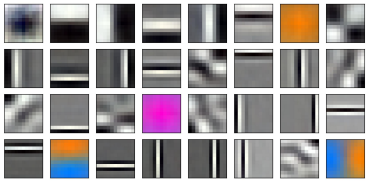
(a) $0.01\mathbf{x}$, $\beta = 1$



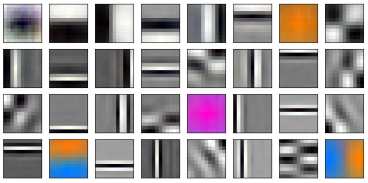
(b) $100\mathbf{x}$, $\beta = 1$



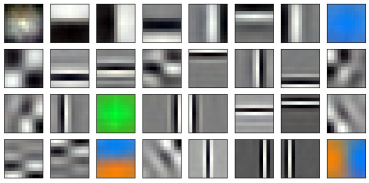
(c) $1000\mathbf{x}$, $\beta = 1$



(d) $0.01\mathbf{x}$, $\beta = \mathbb{E}(||\mathbf{x}||_2)$

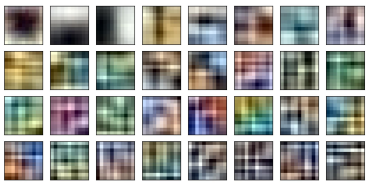


(e) $100\mathbf{x}$, $\beta = \mathbb{E}(||\mathbf{x}||_2)$

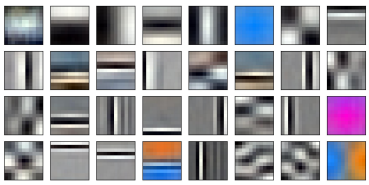


(f) $1000\mathbf{x}$, $\beta = \mathbb{E}(||\mathbf{x}||_2)$

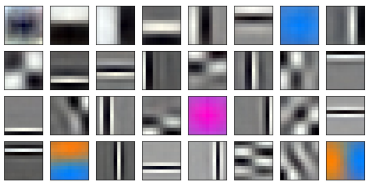
Figure 18: When using RICA F.4, the strength of the sparsity regulariser needs to be adjusted to adapt to the scale of the input. Here we show the effect of the input scale on the obtained filters between adaptive and nonadaptive β. (a-c) have $\beta = 1$, while (d-f) have $\beta = \mathbb{E}(||\mathbf{x}||_2)$. We see that making β proportional to $\mathbb{E}(||\mathbf{x}||_2)$ makes it invariant to the scale of the input.



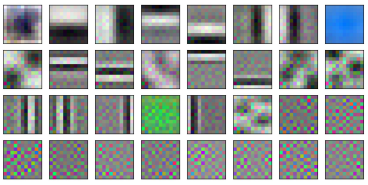
(a) $\beta = 0.01\mathbb{E}(||\mathbf{x}||_2)$



(b) $\beta = 0.1\mathbb{E}(||\mathbf{x}||_2)$



(c) $\beta = 1.0\mathbb{E}(||\mathbf{x}||_2)$



(d) $\beta = 2.0\mathbb{E}(||\mathbf{x}||_2)$

Figure 19: Filters obtained on CIFAR-10 using RICA F.4 with varying L_1 sparsity regularisation intensity. Unit weight normalisation was used (see Appendix C).

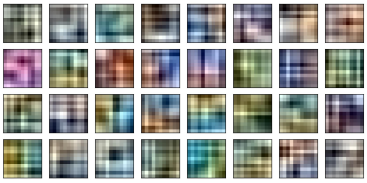
G.3 Linear PCA

Here we summarise variations of linear PCA methods.

Method	Filters
1: PCA via SVD (A.1)	

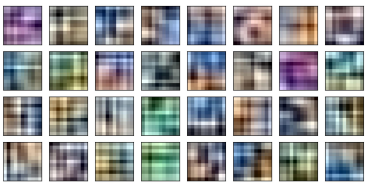
2: Linear autoencoder (A.2.1)

$$\mathcal{L}(\mathbf{W}) = \mathbb{E}(||\mathbf{x} - \mathbf{x}\mathbf{W}\mathbf{W}^T||_2^2)$$



3: Subspace learning algorithm (A.2.2)

$$\Delta\mathbf{W} \propto \mathbf{x}^T\mathbf{y} - \mathbf{W}\mathbf{y}^T\mathbf{y}$$



4: Weighted subspace learning algorithm (WSLA) - variant 1 (A.3.1)

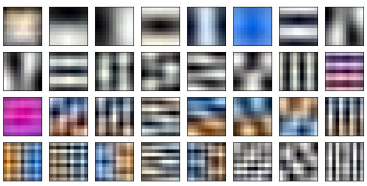
$$\Delta\mathbf{W} \propto \mathbf{x}^T\mathbf{y} - \mathbf{W}\mathbf{y}^T\mathbf{y}\mathbf{\Lambda}^{-1}$$
$$1 \geq \lambda_1 > ... > \lambda_k > 0$$

5: WSLA - variant 2 (A.3.1)

$$\Delta\mathbf{W} \propto \mathbf{x}^T\mathbf{y}\mathbf{\Lambda} - \mathbf{W}\mathbf{y}^T\mathbf{y}$$
$$1 \geq \lambda_1 > ... > \lambda_k > 0$$

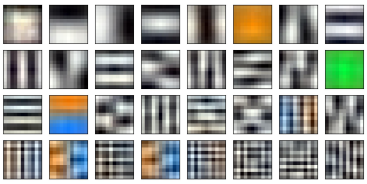
6: WSLA - variant 3 (A.3.1)

$$\Delta\mathbf{W} \propto \mathbf{x}^T\mathbf{y} - \mathbf{W}(\mathbf{y}\mathbf{\Lambda}^{\frac{1}{2}})^T\mathbf{y}\mathbf{\Lambda}^{-\frac{1}{2}}$$
$$1 \geq \lambda_1 > ... > \lambda_k > 0$$



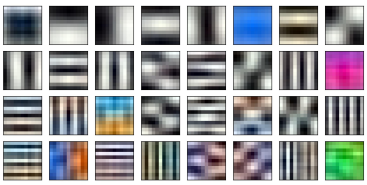
7: Weighted subspace algorithm with unit norm (A.3.2)

$$\mathcal{L}(\mathbf{W}) = \mathbb{E}(||\mathbf{x}[\mathbf{W}]_{sg}\mathbf{W}^T - \mathbf{x}||_2^2$$
$$+ ||\mathbf{W}\hat{\mathbf{\Sigma}}\mathbf{\Lambda}^{\frac{1}{2}}||_2^2 - ||\mathbf{W}\hat{\mathbf{\Sigma}}||_2^2$$
$$- ||\mathbf{x}\mathbf{W}\mathbf{\Lambda}^{\frac{1}{2}}||_2^2 + ||\mathbf{x}\mathbf{W}||_2^2)$$



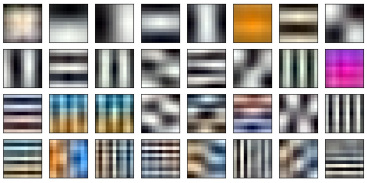
8: Generalised hebbian algorithm (GHA) (A.3.3)

$$\mathbf{y} = \mathbf{x}\mathbf{W}$$
$$\mathbf{\Delta W} \propto \mathbf{x}^T\mathbf{y} - \mathbf{W}\triangle(\mathbf{y}^T\mathbf{y})$$



9: GHA with encoder contribution (A.3.4)

$$\begin{aligned}\Delta\mathbf{W} \propto \quad & \mathbf{x}^T\mathbf{y} - \mathbf{W}\triangle(\mathbf{y}^T\mathbf{y}) \\ & - \mathbf{x}^T\mathbf{y}(\triangle(\mathbf{W}^T\mathbf{W}) - \mathbf{I})\end{aligned}$$

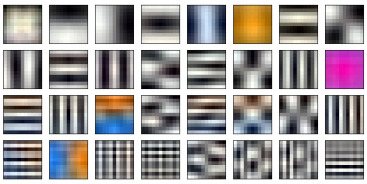


10: GHA + subspace learning algorithm (A.3.4)

$$\begin{aligned}\Delta\mathbf{W} \propto \quad & \mathbf{x}^T\mathbf{y} - \mathbf{W}\triangle(\mathbf{y}^T\mathbf{y}) \\ & - \mathbf{W}(\mathbf{y}^T\mathbf{y} - \mathrm{diag}(\mathbf{y}^2))\end{aligned}$$

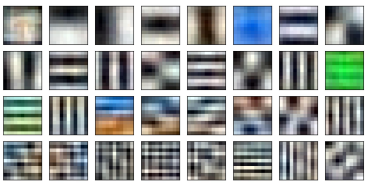
11: Reconstruction + GHA (A.3.4)

$$\begin{aligned}\mathcal{L}(\mathbf{W}) = \mathbb{E}(&||\mathbf{x} - \mathbf{x}\mathbf{W}\mathbf{W}^T||_2^2 \\ & + \mathbf{1}\mathbf{W} \odot [\mathbf{W}\triangle(\mathbf{y}^T\mathbf{y})]_{sg}\mathbf{1}^T)\end{aligned}$$



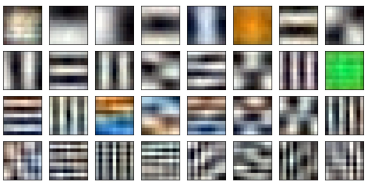
12: Nested dropout (A.3.5)

$$\mathcal{L}(\mathbf{W}) = \mathbb{E}(||(\mathbf{x}\mathbf{W} \odot \mathbf{m}_{1|j})\mathbf{W}^T - \mathbf{x}||_2^2)$$



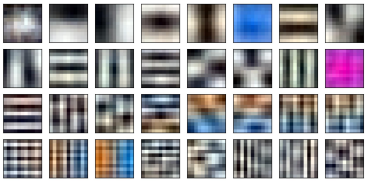
13: Variance maximiser with weighted regulariser (A.3.6)

$$J(\mathbf{W}) = \mathbb{E}(||\mathbf{x} - \mathbf{x}\mathbf{W}\mathbf{W}^T||_2^2 - ||\mathbf{x}\mathbf{W}\mathbf{\Lambda}^{\frac{1}{2}}||_2^2)$$
$$1 \geq \lambda_1 > ... > \lambda_k > 0$$



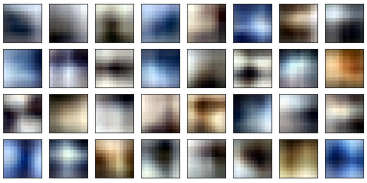
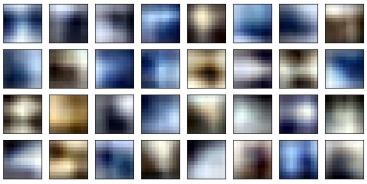
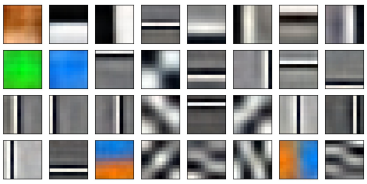
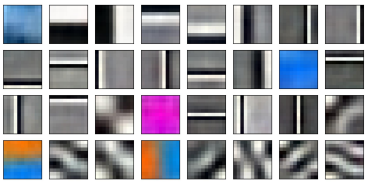
14: Variance maximiser regulariser with nested dropout (A.3.6)

$$\mathcal{L}(\mathbf{W}) = \mathbb{E}(||\mathbf{x} - \mathbf{x}\mathbf{W}\mathbf{W}^T||_2^2 - ||\mathbf{x}\mathbf{W} \cdot \mathbf{m}_{1|j}||_2^2)$$



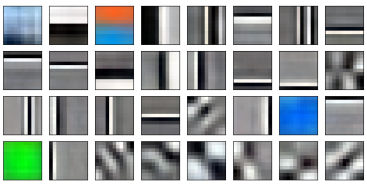
G.4 Nonlinear PCA

Here we summarise variations of nonlinear PCA methods.

Method	Filters
1: Conventional without whitening (2.6) $\mathcal{L}(\mathbf{W}) = \mathbb{E}(\|\|\mathbf{x} - h(\mathbf{x}\mathbf{W})\mathbf{W}^T\|\|_2^2)$	
2: Differentiable $\boldsymbol{\sigma}$ - without stop gradient (3) $\mathcal{L}(\mathbf{W}, \mathbf{\Sigma}) = \mathbb{E}(\|\|\mathbf{x} - h(\mathbf{x}\mathbf{W}\mathbf{\Sigma}^{-1})\mathbf{\Sigma}\mathbf{W}^T\|\|_2^2)$	
3: Differentiable $\boldsymbol{\sigma}$ (Eq. 34) $\mathcal{L}(\mathbf{W}, \mathbf{\Sigma}) = \mathbb{E}(\|\|\mathbf{x} - h(\mathbf{x}\mathbf{W}\mathbf{\Sigma}^{-1})\mathbf{\Sigma}[\mathbf{W}^T]_{sg}\|\|_2^2)$	
4: Differentiable $\boldsymbol{\sigma}$ - latent reconstruction with reconstruction orthogonal regulariser (F.3) $\mathcal{L}(\mathbf{W}, \mathbf{\Sigma}) = \mathbb{E}(\|\|\mathbf{x}[\mathbf{W}]_{sg} - h(\mathbf{x}\mathbf{W}\mathbf{\Sigma})\mathbf{\Sigma}\|\|_2^2) + \mathbb{E}(\|\|\mathbf{x}\mathbf{W}[\mathbf{W}^T]_{sg} - \mathbf{x}\|\|_2^2)$	

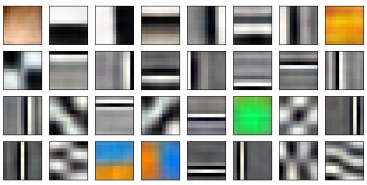
5: Differentiable $\boldsymbol{\sigma}$ - latent reconstruction with symmetric orthogonal regulariser (F.3)

$$\mathcal{L}(\mathbf{W}, \mathbf{\Sigma}) = \mathbb{E}(||\mathbf{x}[\mathbf{W}]_{sg} - h(\mathbf{x}\mathbf{W}\mathbf{\Sigma})\mathbf{\Sigma}||_2^2) + \beta||\mathbf{I} - \mathbf{W}^T\mathbf{W}||_F^2$$
$$\beta = 10$$



6: Differentiable $\boldsymbol{\sigma}$ - latent reconstruction with asymmetric orthogonal regulariser (F.3)

$$\mathcal{L}(\mathbf{W}, \mathbf{\Sigma}) = \mathbb{E}(||\mathbf{x}[\mathbf{W}]_{sg} - h(\mathbf{x}\mathbf{W}\mathbf{\Sigma})\mathbf{\Sigma}||_2^2) + \beta||(\triangle(\mathbf{W}^T[\mathbf{W}]_{sg}))||_F^2$$
$$\beta = 10$$

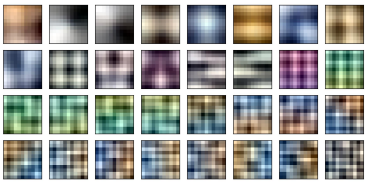


7: Differentiable $\boldsymbol{\sigma}$ - latent reconstruction with $[\mathbf{W}^T\mathbf{W}]_{sg}$ and symmetric orthogonal regulariser (F.3)

$$\mathcal{L}(\mathbf{W}, \mathbf{\Sigma}) = \mathbb{E}(||\mathbf{x}[\mathbf{W}]_{sg} - h(\frac{\mathbf{x}\mathbf{W}}{\boldsymbol{\sigma}})\boldsymbol{\sigma}[\mathbf{W}^T\mathbf{W}]_{sg}||_2^2) + ||\mathbf{I} - \mathbf{W}^T\mathbf{W}||_F^2$$

8: EMA $\boldsymbol{\sigma}$ - tanh (3)

$$\mathcal{L}(\mathbf{W}) = \mathbb{E}(||\mathbf{x} - h(\mathbf{x}\mathbf{W}\mathbf{\Sigma}^{-1})\mathbf{\Sigma}[\mathbf{W}^T]_{sg}||_2^2)$$
$$h(x) = \tanh(x)$$

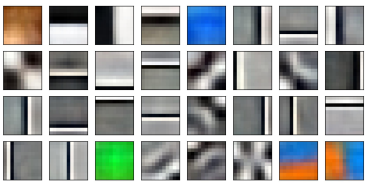


9: EMA $\boldsymbol{\sigma}$ - scaled tanh (3.3)

$$\mathcal{L}(\mathbf{W}) = \mathbb{E}(||\mathbf{x} - h(\mathbf{x}\mathbf{W}\mathbf{\Sigma}^{-1})\mathbf{\Sigma}[\mathbf{W}^T]_{sg}||_2^2)$$
$$h(x) = 4\tanh(\frac{x}{4})$$

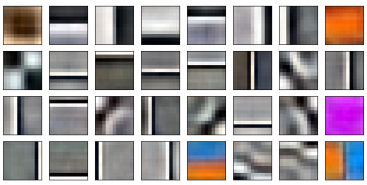
10: EMA $\boldsymbol{\sigma}$ - scaled tanh with modified derivative (Eq. 158)

$$\mathcal{L}(\mathbf{W}) = \mathbb{E}(||\mathbf{x} - h(\mathbf{x}\mathbf{W}\mathbf{\Sigma}^{-1})\mathbf{\Sigma}[\mathbf{W}^T]_{sg}||_2^2)$$
$$h(x) = 4\tanh(\frac{x}{4})$$
$$h'(x) = 1$$



11: EMA $\boldsymbol{\sigma}$ - with hard tanh approximation (3.3)

$$\mathcal{L}(\mathbf{W}) = \mathbb{E}(||\mathbf{x} - h(\mathbf{x}\mathbf{W}\boldsymbol{\Sigma}^{-1})\boldsymbol{\Sigma}[\mathbf{W}^T]_{sg}||_2^2)$$
$$h(x) = max(-2, min(2, 0))$$
$$h'(x) = 1$$



12: Asymmetric activation - adaptive (F.1.1)

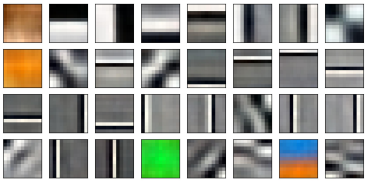
$$\mathcal{L}(\mathbf{W}) = \mathbb{E}(||\mathbf{x} - h(\mathbf{x}\mathbf{W}\boldsymbol{\Sigma}^{-1})\boldsymbol{\Sigma}[\mathbf{W}^T]_{sg}||_2^2)$$
$$h(\mathbf{x}) = \mathbf{a}\tanh(\mathbf{x})$$
$$h'(x) = 1$$
$$\mathbf{a} = 1/[\sqrt{\text{var}(h(\mathbf{z}))}]_{sg}$$

13: Asymmetric activation - constant (F.1.1)

$$\mathcal{L}(\mathbf{W}) = \mathbb{E}(||\mathbf{x} - h(\mathbf{x}\mathbf{W}\boldsymbol{\Sigma}^{-1})\boldsymbol{\Sigma}[\mathbf{W}^T]_{sg}||_2^2)$$
$$h(\mathbf{x}) = 1.6\tanh(\mathbf{x})$$
$$h'(x) = 1$$

14: EMA $\boldsymbol{\sigma}$ - latent reconstruction with symmetric orthogonal regulariser - scaled tanh (F.3)

$$\mathcal{L}(\mathbf{W}, \boldsymbol{\Sigma}) = \mathbb{E}(||\mathbf{x}[\mathbf{W}]_{sg} - h(\mathbf{x}\mathbf{W}\boldsymbol{\Sigma})\boldsymbol{\Sigma}[\mathbf{W}^T\mathbf{W}]_{sg}||_2^2)$$
$$+ ||\mathbf{I} - \mathbf{W}^T\mathbf{W}||_F^2$$
$$h(x) = 4\tanh(\frac{x}{4})$$

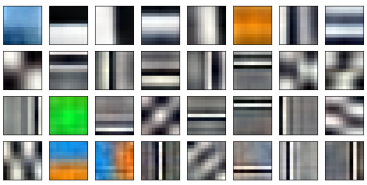


15: EMA $\boldsymbol{\sigma}$ - differentiable scaled tanh (3.3)

$$\mathcal{L}(\mathbf{W}, a) = \mathbb{E}(||\mathbf{x} - h_a(\mathbf{x}\mathbf{W}\boldsymbol{\Sigma}^{-1})\boldsymbol{\Sigma}[\mathbf{W}^T]_{sg}||_2^2)$$
$$h_a(x) = a\tanh(\frac{x}{a})$$

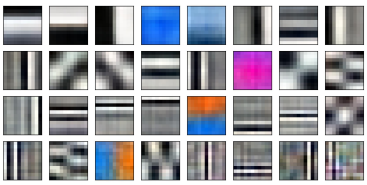
16: EMA $\boldsymbol{\sigma}$ - baked-in GS. No need to order the components as it is done automatically. (3.5)

$$\mathcal{L}(\mathbf{W}) = \mathbb{E}(||\mathbf{x} - h(\mathbf{x}\mathbf{W}P(\mathbf{W})\boldsymbol{\Sigma}^{-1})\boldsymbol{\Sigma}[\mathbf{W}^T]_{sg}||_2^2)$$
$$P(\mathbf{W}) = (I - \triangle(\mathbf{W}^T[\mathbf{W}]_{sg})$$



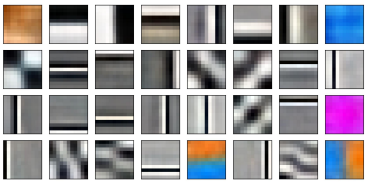
17: EMA $\boldsymbol{\sigma}$ - scaled tanh with nested dropout (F.7.5)

$$\mathcal{L}(\mathbf{W}) = \mathbb{E}(||\mathbf{x} - h(\mathbf{x}\mathbf{W}\boldsymbol{\Sigma}^{-1}) \odot \mathbf{m}_{1|j}\boldsymbol{\Sigma}[\mathbf{W}^T]_{sg}||_2^2) + ||\mathbf{W}^T\mathbf{W} - \mathbf{I}||_F^2$$
$$h(x) = 4\tanh(\frac{x}{4})$$



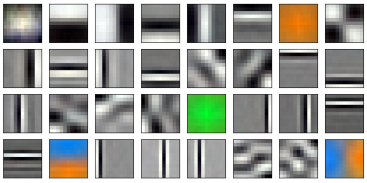
18: EMA $\boldsymbol{\sigma}$ - modified tanh derivative - with linear reconstruction (F.1.1)

$$\mathcal{L}(\mathbf{W}) = \mathbb{E}(||\mathbf{x} - h(\mathbf{x}\mathbf{W}\boldsymbol{\Sigma}^{-1})\boldsymbol{\Sigma}[\mathbf{W}^T]_{sg}||_2^2) + \alpha\mathbb{E}(||\mathbf{x} - \mathbf{x}\mathbf{W}[\mathbf{W}^T]_{sg}||_2^2)$$
$$h(x) = \tanh(x)$$
$$h'(x) = 1$$
$$\alpha \geq 2$$



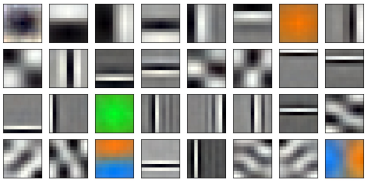
19: RICA (F.4)

$$\mathcal{L}(\mathbf{W}) = \mathbb{E}(||\mathbf{x} - \mathbf{x}\mathbf{W}\mathbf{W}^T||_2^2 + \beta\sum|\mathbf{x}\mathbf{w_j^T}|)$$
$$\beta = \mathbb{E}(||\mathbf{x}||_2)$$



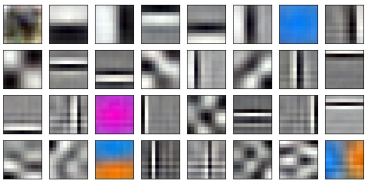
20: Reconstruction with log cosh regulariser (F.4)

$$\mathcal{L}(\mathbf{W}) = \mathbb{E}(||\mathbf{x} - \mathbf{x}\mathbf{W}\mathbf{W}^T||_2^2 + \beta\sum\log\cosh(\mathbf{x}\mathbf{w_j^T}))$$



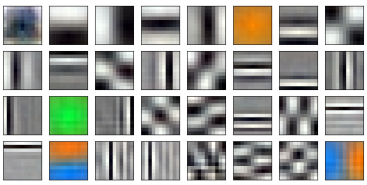
21: Reconstruction skew symmetric $\mathbf{y}^T h(\mathbf{y})$ (F.5)

$$\mathcal{L}(\mathbf{W}) = \mathbb{E}(||\mathbf{x} - \mathbf{x}\mathbf{W}\mathbf{W}^T||_2^2 + \beta\mathbf{1}\mathbf{W} \odot [\mathbf{W}(\mathbf{y}^T h(\mathbf{y}) - h(\mathbf{y})^T\mathbf{y}))]_{sg}\mathbf{1}^T)$$
$$h = \tanh$$
$$\beta = \mathbb{E}(||\mathbf{x}||_2)$$

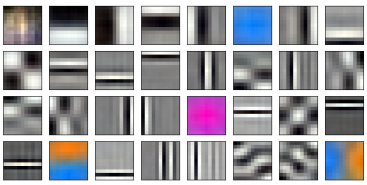


22: Reconstruction with symmetric $\mathbf{y}^T h(\mathbf{y})$ without diagonal (F.5)

$$\mathcal{L}(\mathbf{W}) = \mathbb{E}(||\mathbf{x} - \mathbf{x}\mathbf{W}\mathbf{W}^T||_2^2 + \beta\mathbf{1}\mathbf{W} \odot [\mathbf{W}(\mathbf{y}^T h(\mathbf{y}) - \mathrm{diag}(h(\mathbf{y}) \odot \mathbf{y})))]_{sg}\mathbf{1}^T)$$
$$h = \tanh$$
$$\beta = \mathbb{E}(||\mathbf{x}||_2)$$

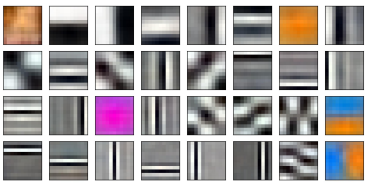


23: Reconstruction with symmetric $\mathbf{y}^T h(\mathbf{y}/\boldsymbol{\sigma})\boldsymbol{\sigma}$ without diagonal (F.5)



$$\begin{aligned}\mathcal{L}(\mathbf{W}) &= \mathbb{E}(||\mathbf{x} - \mathbf{x}\mathbf{W}\mathbf{W}^T||_2^2 \\ &\quad + \mathbf{1}\mathbf{W} \odot [\mathbf{W}(\mathbf{y}^T h(\mathbf{y}\boldsymbol{\Sigma}^{-1})\boldsymbol{\Sigma} \\ &\quad - \operatorname{diag}(h(\mathbf{y}\boldsymbol{\Sigma}^{-1})\boldsymbol{\Sigma} \odot \mathbf{y})))]_{sg}\mathbf{1}^T) \\ h &= \operatorname{sign}\end{aligned}$$

24: EMA $\boldsymbol{\sigma}$ - scaled tanh with triangular update (F.7.2)



$$\begin{aligned}\mathcal{L}(\mathbf{W}) &= \mathbb{E}(||\mathbf{x} - h(\mathbf{x}\mathbf{W}\boldsymbol{\Sigma}^{-1})\boldsymbol{\Sigma}[\mathbf{W}^T]_{sg}||_2^2 \\ &\quad + \mathbf{1}\mathbf{W} \odot [\mathbf{W} \triangledown (\mathbf{y}^T\mathbf{y}\boldsymbol{\Sigma}^{-1})]_{sg}\mathbf{1}^T) \\ h(x) &= 6\tanh(\frac{x}{6})\end{aligned}$$

G.5 Time Series

G.5.1 Orthogonal - same variance

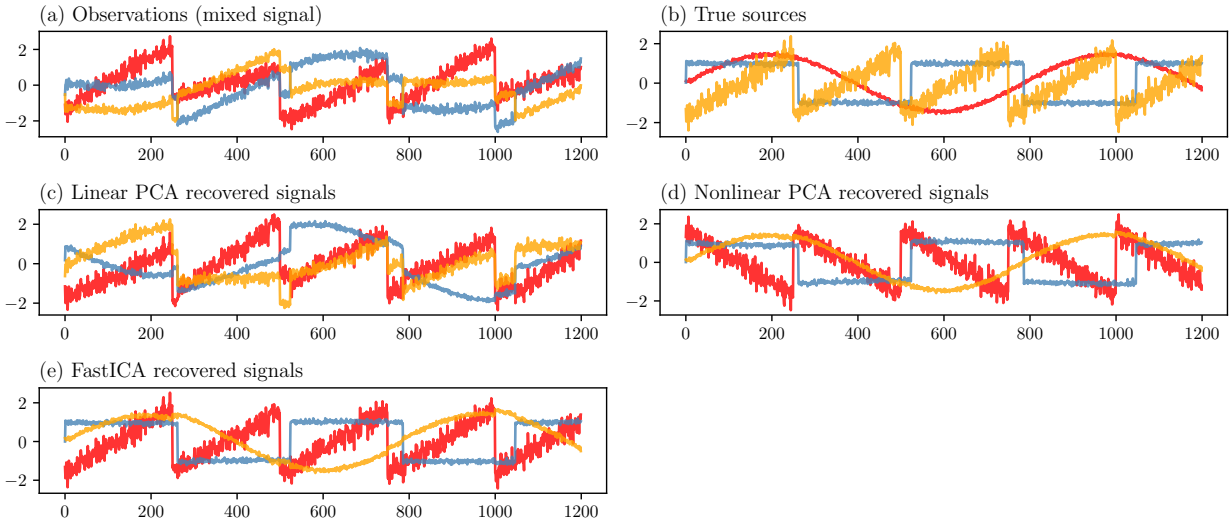


Figure 20: Three signals (sinusoidal, square, and sawtooth) that were mixed with an *orthogonal* mixing matrix. Linear PCA was unable to separate the signals as they all had the same variance.

G.5.2 Orthogonal - all distinct variances

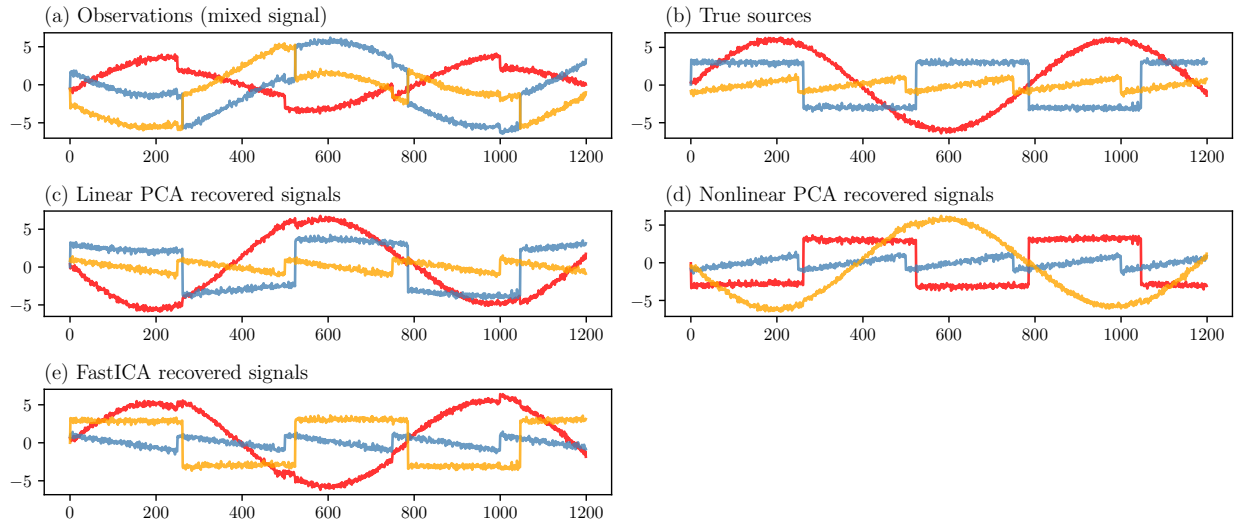
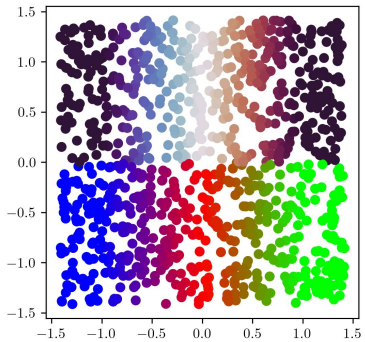


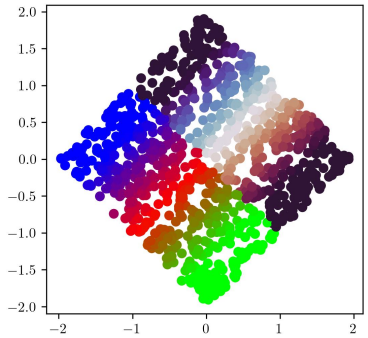
Figure 21: Three signals (sinusoidal, square, and sawtooth) with distinct variances that were mixed with an *orthogonal* mixing matrix. All three managed to separate the signals.

G.6 Sub- and Super- Gaussian 2D points

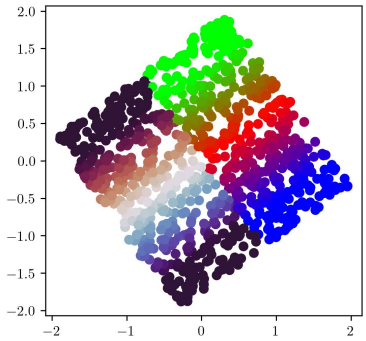
Here we look at the effect of applying linear PCA, nonlinear PCA, and linear ICA on 2D points ($n = 1000$) sampled from either a uniform distribution (sub-Gaussian) or a Laplace distribution (super-Gaussian). Figures 22 and 23 summarise the results.



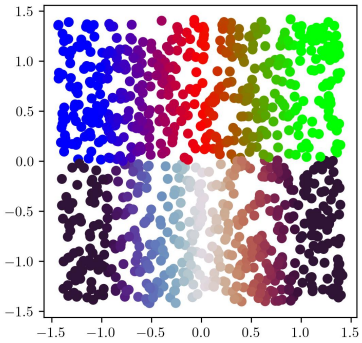
(a) Original



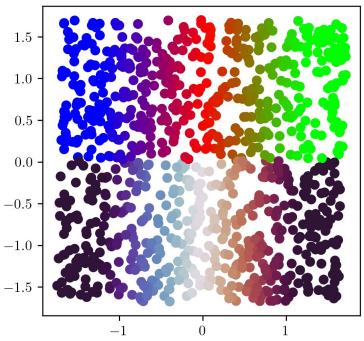
(b) Rotated by $\pi/4$



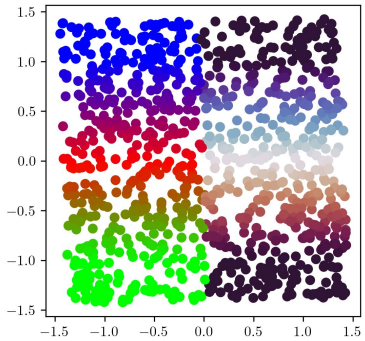
(c) Linear PCA



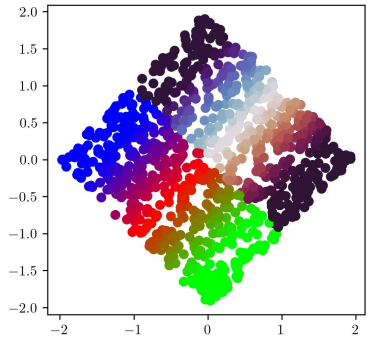
(d) Nonlinear PCA
$h(z) = \tanh(z)$



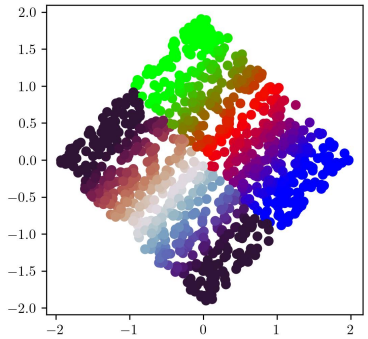
(e) FastICA



(f) Nonlinear PCA
$h(z) = 0.5\tanh(z/0.5)$

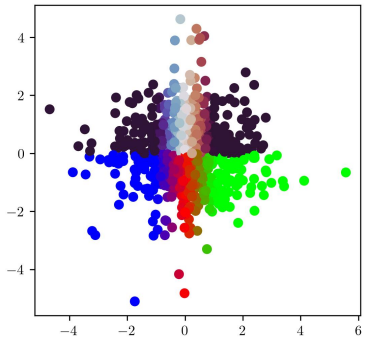


(g) Nonlinear PCA
$h(z) = 2\tanh(z/2)$

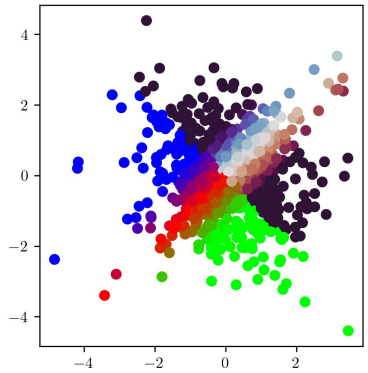


(h) Nonlinear PCA
$h(z) = 3\tanh(z/3)$

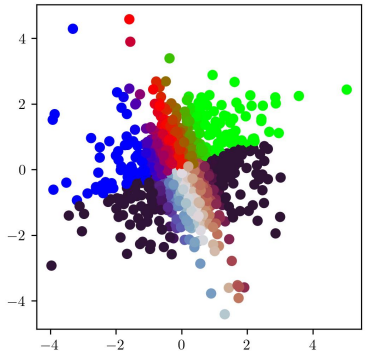
Figure 22: Uniform distribution (sub-Gaussian) with equal variance. The original data (a) was rotated by $\pi/4$ (b), then we attempted to recover the original data from the rotated data. We see that linear PCA (c) failed to recover the original data. Both nonlinear PCA (d) and FastICA (e) managed to recover the original data (up to sign and permutational indeterminacies, given the equal variance). We also see that $h(z) = a\tanh(z/a)$ with $a \leq 1$ worked best for the uniform distribution with nonlinear PCA.



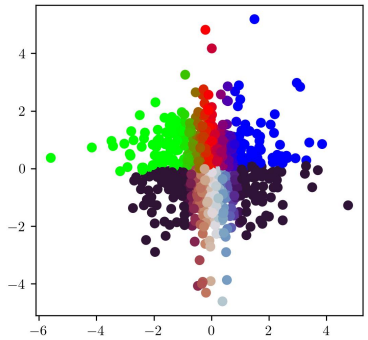
(a) Original



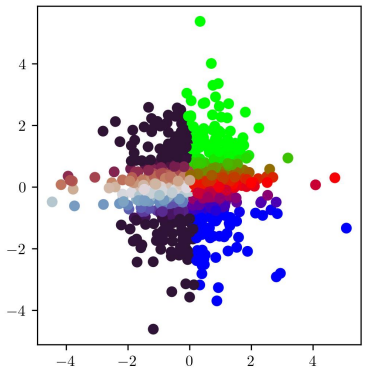
(b) Rotated by $\pi/4$



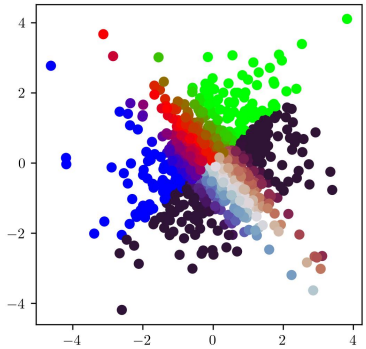
(c) Linear PCA



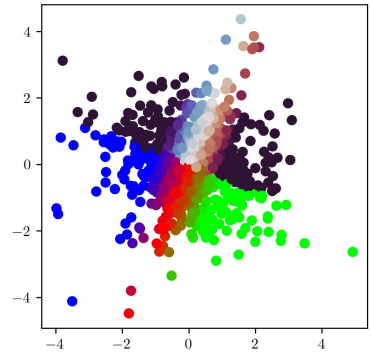
(d) Nonlinear PCA
$h(z) = 3\tanh(z/3)$



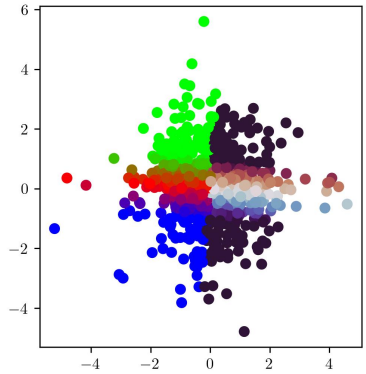
(e) FastICA



(f) Nonlinear PCA
$h(z) = \tanh(z)$



(g) Nonlinear PCA
$h(z) = 2\tanh(z/2)$



(h) Nonlinear PCA
$h(z) = 4\tanh(z/4)$

Figure 23: Laplace distribution (super-Gaussian) with equal variance. The original data (a) was rotated by $\pi/4$ (b), then we attempted to recover the original data from the rotated data. We see that linear PCA (c) failed to recover the original data. Both nonlinear PCA (d) and FastICA (e) managed to recover the original data (up to sign and permutational indeterminacies, given the equal variance). We also see that $h(z) = a\tanh(z/a)$ with $a \geq 3$ work best for the Laplace distribution with nonlinear PCA.

H Training details

patches We used the Adam optimiser (Kingma & Ba, 2014) with a learning rate either of 0.01 or 0.001, $\beta_1 = 0.9$ and $\beta_2 = 0.999$. We used a batch size of 128. For the patches, each epoch consisted of 4K iterations, and, unless stated otherwise, we trained for a total of three epochs. With the majority of the nonlinear PCA methods, we used a projective unit norm constraint (see Appendix C).

Time signals We used the vanilla SGD optimiser with a learning rate of 0.01 and momentum 0.9. We used a batch size of 100 and trained for a total of 200 epochs and picked the weights with the lowest reconstruction loss. When using the Adam optimiser, we noted that it seemed better to also use differentiable weight normalisation in addition to the projective unit norm constraint.

2D points We used the vanilla SGD optimiser with a learning rate of 0.01 and momentum 0.9. We used a batch size of 100 and trained for a total of 100 epochs.

I Block rotation matrix RS = SR

$$\mathbf{SR} = \begin{pmatrix} s_1\mathbf{I}_{2\times 2} & \mathbf{0} \\ \mathbf{0} & s_2 \end{pmatrix} \begin{pmatrix} \mathbf{R}_{2\times 2} & \mathbf{0} \\ \mathbf{0} & 1 \end{pmatrix} = \begin{pmatrix} s_1\mathbf{R}_{2\times 2} & \mathbf{0} \\ \mathbf{0} & s_2 \end{pmatrix} = \begin{pmatrix} \mathbf{R}_{2\times 2} & \mathbf{0} \\ \mathbf{0} & 1 \end{pmatrix} \begin{pmatrix} s_1\mathbf{I}_{2\times 2} & \mathbf{0} \\ \mathbf{0} & s_2 \end{pmatrix} = \mathbf{RS} \tag{252}$$

J Exponential moving average (EMA)

$$\bar{\boldsymbol{\mu}} = \frac{1}{b}\sum \mathbf{y}_i \tag{253}$$

$$\bar{\boldsymbol{\sigma}}^2 = \frac{1}{b}\sum (\mathbf{y}_i - \bar{\boldsymbol{\mu}})^2 \tag{254}$$

$$\hat{\boldsymbol{\mu}} = \alpha\hat{\boldsymbol{\mu}} + (1-\alpha)[\bar{\boldsymbol{\mu}}]_{sg} \tag{255}$$

$$\hat{\boldsymbol{\sigma}}^2 = \alpha\hat{\boldsymbol{\sigma}}^2 + (1-\alpha)[\bar{\boldsymbol{\sigma}}^2]_{sg} \tag{256}$$

$$\text{Batch} \quad n_{\mathrm{B}}(\mathbf{y}) = \frac{\mathbf{y} - \bar{\boldsymbol{\mu}}}{\sqrt{\bar{\boldsymbol{\sigma}}^2 + \epsilon}} \tag{257}$$

$$\text{EMA} \quad n_{\mathrm{E}}(\mathbf{y}) = \frac{\mathbf{y} - \hat{\boldsymbol{\mu}}}{\sqrt{\hat{\boldsymbol{\sigma}}^2 + \epsilon}} \tag{258}$$

where $[\quad]_{sg}$ is the stop gradient operator, α is the momentum, and b is the batch size.