Neural Network-Assisted Nonlinear Multiview Analysis: Identifiability and Algorithm

Qi Lyu and Xiao Fu
School of Electrical Engineering and Computer Science
Oregon State University
Email: (lyuqi,xiao.fu)@oregonstate.edu

Abstract

Multiview analysis aims at extracting common information from data samples that are acquired in different domains, e.g., image, text, and audio. Classic multiview analysis, e.g., canonical correlation analysis (CCA), tries to match different views in a certain latent domain via linear transformations. More recently, powerful nonlinear learning tools such as kernel methods and neural networks are employed for enhancing multiview analysis. However, unlike linear CCA whose theoretical aspects are clearly understood, nonlinear CCA formulations are largely intuition-driven. In particular, it is unclear under what conditions the shared components can be identified through nonlinear transformations—while identifiability plays an essential role in many applications. In this work, we revisit nonlinear multiview analysis and address both the theoretical and computational aspects. We motivate our work from a nonlinear multiview mixture model viewpoint that is a natural extension of classic generative models for linear CCA. From there, we derive a new nonlinear multiview analysis formulation. We show that solving the formulated criterion leads to identification of the latent shared components' range space, under reasonable conditions. Our derivation and formulation also offer new insights and interpretations to existing deep neural network-based CCA formulations. On the computation side, we propose an effective algorithm with simple and scalable update rules. A series of simulations corroborate our theoretical analysis.

1 Introduction

Multiview analysis has been an indispensable tool in statistical signal processing, machine learning, and data analytics. A view can be understood as measurements of data entities (e.g., a cat) in a certain domain (e.g., text, image, and audio). Most data entities naturally appear in different domains with different forms. Multiview analysis aims at extracting essential and common information from different views to represent the entities. Compared with single-view analysis tools like principal component analysis (PCA), multiview analysis tools such as canonical correlation analysis (CCA) [1, 2] have an array of benefits, e.g., more robust to colored but view-specific noise and interferences [3–6].

Classic CCA has been extensively studied in the literature, ever since its proposal in statistics in the 1930s [1, 2, 7–9]. In a nutshell, classic CCA seeks linear transformations for the views. The transformations are supposed to ‘project’ the views to a domain where the views share similar representations. Interestingly, the formulated optimization problem, although being nonconvex, can be converted into a generalized eigendecomposition problem and solved efficiently [8]. In recent
years, many attempts have been made towards scaling up classic CCA to handle big datasets; see [10,11]. Beyond two views, a series of generalized CCA (GCCA) formulations exist [12]—whose scalable versions have also been considered [13–16].

Linear transformation-based CCA/GCCA algorithms are elegant in computation. However, from a modeling viewpoint, restricting the transformations to be linear makes the “modeling power” limited. For decades, a lot effort has been invested to extending the CCA/GCCA ideas to the nonlinear regime—via incorporating a variety of nonlinear transformations. For example, kernel CCA has been popular since the 2000s [17–20]. More recently, together with the success of deep learning, deep neural networks are also used to enhance multiview analysis in unsupervised representation learning [21, 22]. Compared to kernel methods, deep neural networks are considered more flexible and more scalable.

Nonlinear multiview analysis tools like kernel CCA and deep CCA have demonstrated effectiveness in many real-world applications, e.g., image representation learning [21] and speech processing [23]. This is encouraging—which shows that incorporating nonlinearity in data processing is indeed well-motivated. On the other hand, it is still largely unclear why these methods work or how to improve existing schemes with theoretical supports. In fact, unlike linear CCA, whose generative models, parameter identifiability issues, and computational aspects are fairly well understood, nonlinear CCA formulations are largely intuition-driven.

In this work, we revisit the nonlinear multiview analysis problem, and offer both theoretical understanding and theory-backed implementation. Our work is motivated by a recent work in linear CCA [4], where different views are modeled as mixtures of shared components and view-specific components (which are interferences). This model is similar to the one considered in machine learning under the context of probabilistic CCA [3]. Under this model, interesting interpretation for the effectiveness of CCA can be obtained. Specifically, the work in [4] shows that the classic linear CCA can extract the shared components' range space—no matter how strong are the interference terms. The model is fairly simple and succinct, yet the insight is significant: It explains the robustness of linear multiview analysis and gives clear scenarios under which multiview analysis is preferred over the single-view ones.

Building upon the intriguing perspectives in [4], we take a step further and consider the following problem: If the acquired views are nonlinearly distorted mixtures of shard and view-specific components, is it still possible to extract the same shared information as in the linear case? This problem is very well-motivated, since a large variety of acquired real data are subject to unknown nonlinear distortions, due to various reasons such as limited sensor dynamic range, modeling error, and non-additive noise.

However, taking into consideration of unknown nonlinear distortions makes the problem of interest quite challenging—both in theory and practice. Nonlinearity removal from mixture models was considered under two cases, which are both single-view analysis problems: 1) nonlinear independent component analysis (nICA) [24–28], where the components of interest are statistically independent random processes, and 2) nonlinear mixture learning (NML) [29], where the sought latent components reside in a confined manifold, i.e., the probability simplex. In both cases, the assumptions on the components of interest are leveraged on to come up with identification criteria. However, both statistical independence and the probability-simplex type structure are considered special assumptions, which may not hold in general. In addition, both nICA and NML assume ‘clean’ mixtures without interference present. With multiple views available, can we circumvent these assumptions? More interestingly, is fending against non-interesting interfering components
still feasible under nonlinear settings?

**Contributions** Bearing these questions in mind, we address both the analytical and computational aspects of nonlinear multiview analysis. Specifically, our contributions are as follows:

- **Model-Based Formulation and Analysis.** We propose a multiview nonlinear mixture model that is a natural extension of the mixture model-based linear multiview analysis [3, 4]. To be specific, we model each view as a nonlinear mixture of shared and view-specific components, where the nonlinear distortions are unknown continuous invertible functions. We propose an identification criterion for extracting the shared information across views. We show that solving the formulated problem can indeed identify the range space spanned by the shared components—regardless how strong the interference terms are. This is surprising, since the identifiability result is consistent with that of the linear case, despite of working under a much more challenging scenario.

- **Neural Network-Based Algorithm Design.** Based on our formulation, we propose a neural network based implementation. The formulated optimization surrogate is delicately designed to realize the identification criterion in practice. In particular, possible trivial solutions revealed in the analysis are circumvented via judicious construction of the optimization objective and constraints. Based on the formulation, we propose a simple and easy-to-implement algorithm. The proposed implementation is compatible with virtually all popular neural network architectures (e.g., convolutive neural networks (CNN) and fully connected neural networks [30]) and is scalable for handling big data.

- **Extensive Simulations.** We test our method in a number of simulations under different scenarios to validate the identifiability theory and to showcase the effectiveness of the implementation.

**Notation** We largely follow establish conventions in signal processing. To be specific, we use \(x, x, X\) to represent a scalar, vector, and matrix, respectively. \(\| \cdot \|\) denotes the Euclidean norm, i.e., \(\|x\|_2\) and \(\|X\|_F\), respectively; \(\top\) and \(\dagger\) denote transpose and pseudoinverse, respectively; \(f'\) and \(f''\) denote first-order and second-order derivative of \(f\), respectively; \(\text{Tr}(\cdot)\) denotes the trace operator.

**2 Background**

In this section, we briefly introduce some preliminaries pertaining to our work.

**2.1 Mixture Models**

Mixture models have proven very useful in signal processing and machine learning. The simplest linear mixture model (LMM) can be written as [31–33]:

\[
y_\ell = A s_\ell, \quad \ell = 0, 1, 2, \ldots, (1)
\]

where \(y_\ell \in \mathbb{R}^M\) denotes the \(\ell\)th observed signal, \(A \in \mathbb{R}^{M \times N}\) the mixing system, and \(s_\ell \in \mathbb{R}^K\) the latent signals (or, the ‘sources’).

If both the mixing system and the sources are unknown, estimating \(A\) and \(s_\ell\) simultaneously poses a very hard problem—which is known as the blind source separation (BSS) problem [31]. The
BSS problem is ill-posed, since in general the model is not identifiable; i.e., even if there is no noise, one could have an infinite number of solutions that satisfy (1). To be specific, for any solution \( A, s_\ell \) satisfying \( y_\ell = As_\ell \), one can let \( \tilde{A} = AQ \) and \( \tilde{s}_\ell = Q^{-1}s_\ell \) for an arbitrary nonsingular \( Q \in \mathbb{R}^{K \times K} \) such that
\[
y_\ell = \tilde{A}\tilde{s}_\ell.
\]

Nonetheless, identifiability can be established via exploiting properties of \( s_\ell \). For example, the seminal work of independent component analysis (ICA) [34] shows identifiability of \( s_\ell \) can be established (up to scaling and permutation ambiguities) leveraging statistical independence between the elements of \( s_\ell \)—if \( s_{k,\ell} \) for every \( k \) is a zero-mean random process observed at time \( \ell \). Later on, identifiability of \( A \) and/or \( s_\ell \) were established through exploiting other properties of the latent components, e.g., nonnegativity [35, 36], convex geometry [37, 38], quasi-stationarity [39–42], boundedness [43], and sparsity [44], just to name a few.

Beyond the LMM, nonlinear mixture models (NMMs) have also attracted much attention—since nonlinearity naturally happens in practice for numerous reasons. For example, starting from the 1990s, a line of work named nonlinear ICA [24, 26–28] considered the following model:
\[
y_\ell = g(s_\ell), \quad \ell = 0, 1, 2, \ldots, (2)
\]
where \( g(\cdot) : \mathbb{R}^K \rightarrow \mathbb{R}^M \) is a nonlinear mixing system. While it was shown that (2) is in general not identifiable [45], the so-called post-nonlinear (PNL) model can be identified, under the premise that \( s_{k,\ell} \) and \( s_{j,\ell} \) are statistically independent random processes [26, 27, 46]. To be specific, consider the model
\[
y_\ell = g(As_\ell), \quad \ell = 0, 1, 2, \ldots, (3)
\]
with \( g = [g_1, \ldots, g_M]^\top \) and \( g_m(\cdot) : \mathbb{R} \rightarrow \mathbb{R} \) being a scalar nonlinear function. The PNL model, although less general relative to (2), is still very meaningful—which finds applications where different nonlinear effects are introduced by multiple sensors, e.g., in brain signal processing [47–49] and hyperspectral imaging [50, 51]. Under the PNL model, \( s_\ell \) can be identified if the sources are statistically independent [26, 27]. Recently, the work in [29] proved that even if the sources are dependent, some other properties, e.g., nonnegativity and stochasticity (i.e., \( 1^\top s_\ell = 1 \)), can be exploited to identify \( s_\ell \) under the post-nonlinear model.

### 2.2 Multiview Data and Mixture Learning

In practice, data may be present in different feature spaces—leading to the so-called multiview data. Multiview data has been frequently connected to mixture models, since it is believed that multiple views of the same data sample have certain shared but latent components. In addition, it is commonly believed that using multiple views may have some advantages over using a single view for mixture model learning problems, e.g., in combating noise and strong interferences—since more information is available.

One way of modeling multiview data is using the following linear model [3]:
\[
y^{(q)}_\ell \approx A^{(q)}s_\ell, (4)
\]
where \( q \) is the index of views. From the above, one can see that \( s_\ell \) can be used to model the latent shared components of different views, while \( A^{(q)} \) is the basis of the subspace where the \( q \)th view is observed. In other words, the different appearances of the views are caused by the differences of
the observation subspaces, while the latent representations of the views are identical. In a recent work [4], the above model in (4) is further developed to incorporate view-specific components, where we have:

\[ y_{\ell}^{(q)} = A^{(q)} s_{\ell}^{(q)}, \quad s_{\ell}^{(q)} = \Pi s_{\ell}^{\top}, \quad c_{\ell}^{(q)} \]

where \( A^{(q)} \in \mathbb{R}^{M_q \times (K+R_q)} \), \( \Pi \in \mathbb{R}^{(K+R_q) \times (K+R_q)} \), \( s_{\ell} \in \mathbb{R}^K \) collects the shared components and \( c_{\ell}^{(q)} \in \mathbb{R}^{R_q} \) the view-specific components. This model is plausible, since it gives more flexibility for modeling the cross-view differences. More importantly, it explains the enhanced robustness of multiview analysis against the single view ones, e.g., PCA, for extracting essential information about the data. More specifically, consider the following linear CCA formulation [2,13]:

\[
\begin{align*}
\text{minimize} & \sum_{\ell=1}^{N} \left\| B_1 y_{\ell}^{(1)} - B_2 y_{\ell}^{(2)} \right\|_2^2 \\
\text{subject to} & \mathbb{E} \left[ B^{(q)} y_{\ell}^{(q)} (y_{\ell}^{(q)})^{\top} (B^{(q)})^{\top} \right] = I, \quad q = 1, 2
\end{align*}
\]

where \( B^{(q)} \in \mathbb{R}^{R_q \times M_q} \), in which we have

\[ 1 \leq R \leq K, \]

and (6b) is employed to avoid degenerate solutions. The work in [4] shows that, when noise is absent, solving the above can identify \( \hat{x}_{\ell} = \Theta s_{\ell} \), where \( \Theta \in \mathbb{R}^{K \times K} \) is a nonsingular matrix—no matter how strong the view-specific components are. Note that \( \hat{x}_{\ell} = \Theta s_{\ell} \) is again an LMM, and many techniques introduced in the previous section can be used to identify \( s_{\ell} \)—with the non-interesting components \( c_{\ell}^{(q)} \) eliminated.

### 2.3 Nonlinear Multiview Learning

As in the single view case, nonlinearity is natural to be considered in the multiview case. For example, nonlinear learning tools such as the kernel method and deep learning have been combined with CCA, where interesting results are observed [19–22]. For example, the idea of deep CCA is to employ deep neural networks, instead of \( B_{\ell}^{(q)} \)'s as in the classic CCA, to perform data transformation; e.g., the deep CCA formulation in [22] is to maximize the correlation between \( f_{\ell}^{(q)}(y_{\ell}^{(q)}) \) across \( q \), where \( f_{q}(\cdot) \) is a deep neural network. Nevertheless, these works mostly consider practical implementations and statistical learning-flavored analysis (e.g., finite sample v.s. population case analysis as in [19]). It is unclear if nonlinearly distorted multiview mixtures can be identified. This is the starting point of our work.
3 Proposed Approach

3.1 Signal Model

In this section, we propose a nonlinear multiview learning method that aims at learning shared information from the following model:

\[ y^q_\ell = g^q(A^q s^q_\ell), \quad (7a) \]
\[ s^q_\ell = \Pi[s^\top_\ell, (c^q_\ell)]^\top, \quad \ell = 1, 2, \ldots, \quad (7b) \]

where the shared components are zero-mean random processes and uncorrelated, i.e.,

\[ E[s_{i,\ell}] = 0, \quad E[s_\ell s_\ell^\top] = \Sigma_s : \text{diagonal,} \]

the function \( g^q = [g_1^q, \ldots, g_{Mq}^q]^\top \) and

\[ g_m^q(\cdot) : \mathbb{R} \rightarrow \mathbb{R} \]

represents the view specific nonlinear distortion. Under the nonlinear setting, we also assume that the latent variables are defined over continuous open sets \( S \) and \( C_q \), i.e.,

\[ s_\ell \in S, \quad c^q_\ell \in C_q, \quad q = 1, 2. \quad (8) \]

Note that this model is a natural extension of the multiview linear mixture models as in [3, 4], especially the one in [4].

3.2 A Function Learning-Based Formulation

Our goal is to learn

\[ x_\ell = \Theta s_\ell, \quad \ell = 0, 1, 2, \ldots, \]

from the nonlinearly distorted views. If this is viable, then, many other methods can be applied to pin down \( s_\ell \), e.g., a number of blind source separation (BSS) algorithms [40, 41, 52]. Note that we do not impose strong structural assumptions on \( s_\ell \), e.g., statistical independence between \( s_{r,\ell} \) and \( s_{j,\ell} \) as in [26, 27] and the stochasticity assumption (i.e., \( 1^\top s_\ell = 1, \quad s_\ell \geq 0 \)) as in [29]—existing nonlinear ICA and mixture learning techniques cannot be applied to the problem of interest.

Our idea is to seek \( f^{(q)} \) and \( B^{(q)} \) such that the following criterion is satisfied:

\[
\begin{align*}
\text{find } & B^{(1)}, B^{(2)}, f^{(1)}, f^{(2)} \\
\text{s.t. } & B^{(1)} f^{(1)} (y^{(1)}_\ell) = B^{(2)} f^{(2)} (y^{(2)}_\ell), \quad \forall y_\ell \\
& f^{(q)} : \text{invertible, } q = 1, 2, \\
& E \left[ B^{(q)} f^{(q)} (y^{(q)}_\ell) f^{(q)} (y^{(q)}_\ell)^\top B^{(q)} \right] = I. 
\end{align*}
\]

(9a)

(9b)

(9c)

(9d)
Ideally, we wish to obtain

\[ B^{(q)} = [\Theta, 0] \Pi^T(A^{(q)})^\dagger \in \mathbb{R}^{K \times M_q}, \quad (10a) \]

\[ f^{(q)}(\cdot) = \left[ f_1^{(q)}(\cdot), \ldots, f_{M_q}^{(q)}(\cdot) \right]^T : \mathbb{R}^{M_q} \to \mathbb{R}^{M_q}, \quad (10b) \]

\[ f_m^{(q)}(\cdot) = (g_m^{(q)}(\cdot))^{-1}, \quad m = 1, \ldots, M_q, \quad (10c) \]

where \( \Theta \in \mathbb{R}^{K \times K} \) is nonsingular. The above will extract the shared row subspace. Also note that the solution in (10) is a legitimate solution to Problem (9). The key question is: Is the solution in (10) the only solution? That is, does the formulation in (9) have identifiability for the shared row subspace?

### 3.3 Nonlinearity Removal

To see how we approach the identifiability problem, let us start with a simple illustrative case where \( A^{(q)} \in \mathbb{R}^{2 \times 2} \) and there is only one shared component. This way, the generative model is simplified as follows:

\[
y^{(1)}_t = \begin{bmatrix} g_1^{(1)} \left( A_{11}^{(1)} s_t + A_{12}^{(1)} c_t^{(1)} \right), & g_2^{(1)} \left( A_{21}^{(1)} s_t + A_{22}^{(1)} c_t^{(1)} \right) \end{bmatrix}^T
\]

\[
y^{(2)}_t = \begin{bmatrix} g_1^{(2)} \left( A_{11}^{(2)} s_t + A_{12}^{(2)} c_t^{(2)} \right), & g_2^{(2)} \left( A_{21}^{(2)} s_t + A_{22}^{(2)} c_t^{(2)} \right) \end{bmatrix}^T.
\]

For this simplified case, the solution \( B^{(q)} \) becomes a row vector \( (b^{(q)})^\top \). One can show that the following holds:

**Theorem 1 (The 2 \times 2 Case)** Assume that the elements of \( A^{(q)} \in \mathbb{R}^{2 \times 2} \) are drawn from any jointly continuous distribution for \( q = 1, 2 \), and that \( g^{(q)} \) and the composition \( f^{(q)} \circ g^{(q)} \) are both twice differentiable.. Also assume that (9b), i.e.,

\[ (b^{(1)})^\top f^{(1)}(A^{(1)} s_t^{(1)}) = (b^{(2)})^\top f^{(2)}(A^{(2)} s_t^{(2)}) \]

holds for all \( s_t \in \mathcal{S} \) and \( c_t^{(q)} \in \mathcal{C} \). Suppose that \((b^{(q)}, f^{(q)}) \) for \( q = 1, 2 \) satisfies (9b) and (9c) with \( \|b^{(q)}\|_0 = 2 \). Then, we have the following holds:

\[ h_i(x) = f_i^{(q)} \circ g_i^{(q)}(x) = \alpha_i^{(q)} x + d_i^{(q)}, \quad \alpha_i^{(q)} \neq 0; \]

i.e., the composition is an affine function with probability one.

**Proof** Let us denote \( h_q = f_q \circ g_q \) for \( q = 1, 2 \) as the composition of the learned \( f_q \) and \( g_q \) in the generative model. Hence, from Eq. (9), we have

\[
b_1^{(1)} h_1^{(1)} \left( A_{11}^{(1)} s + A_{12}^{(1)} c^{(1)} \right) + b_2^{(1)} h_2^{(1)} \left( A_{21}^{(1)} s + A_{22}^{(1)} c^{(1)} \right) \\
= b_1^{(2)} h_1^{(2)} \left( A_{11}^{(2)} s + A_{12}^{(2)} c^{(2)} \right) + b_2^{(2)} h_2^{(2)} \left( A_{21}^{(2)} s + A_{22}^{(2)} c^{(2)} \right)
\]

where we have omitted the subscript “\( \ell \)” for notational simplicity.
Since we have assumed that the above holds for all possible \( y \), it holds for all values of \( c^{(q)} \) and \( s \) in a continuous space. Denote \( \phi(c^{(1)}, c^{(2)}, s) \) as

\[
\phi(c^{(1)}, c^{(2)}, s) = b_1^{(1)} h_1^{(1)} \left( A_{11}^{(1)} s + A_{12}^{(1)} c^{(1)} \right) + b_2^{(1)} h_2^{(1)} \left( A_{21}^{(1)} s + A_{22}^{(1)} c^{(1)} \right) - b_1^{(2)} h_1^{(2)} \left( A_{11}^{(2)} s + A_{12}^{(2)} c^{(2)} \right) - b_2^{(2)} h_2^{(2)} \left( A_{21}^{(2)} s + A_{22}^{(2)} c^{(2)} \right).
\]

To proceed, taking derivative w.r.t \( s \) and \( c^{(1)} \) respectively. Then, we have

\[
\frac{\partial \phi(c^{(1)}, c^{(2)}, s)}{\partial s} = 0 \Rightarrow b_1^{(1)} A_{11}^{(1)} (h_1^{(1)})' + b_2^{(1)} A_{21}^{(1)} (h_2^{(1)})' = b_1^{(2)} A_{11}^{(2)} (h_1^{(2)})' + b_2^{(2)} A_{21}^{(2)} (h_2^{(2)})'.
\]

\[
\frac{\partial \phi(c^{(1)}, c^{(2)}, s)}{\partial c^{(1)}} = 0 \Rightarrow b_1^{(1)} A_{12}^{(1)} (h_1^{(1)})' + b_2^{(1)} A_{22}^{(1)} (h_2^{(1)})' = 0.
\]

By further taking the second derivatives, we have

\[
\frac{\partial^2 \phi(c^{(1)}, c^{(2)}, s)}{\partial (c^{(1)})^2} = 0
\]

\[
\Rightarrow b_1^{(1)} (A_{12}^{(1)})^2 (h_1^{(1)})'' + b_2^{(1)} (A_{22}^{(1)})^2 (h_2^{(1)})'' = 0,
\]

\[
\frac{\partial^2 \phi(c^{(1)}, c^{(2)}, s)}{\partial c^{(1)} \partial s} = 0
\]

\[
\Rightarrow b_1^{(1)} A_{11}^{(1)} A_{12}^{(1)} (h_1^{(1)})'' + b_2^{(1)} A_{21}^{(1)} A_{22}^{(1)} (h_2^{(1)})'' = 0.
\]

From the above, one can establish the following system of linear equations:

\[
\begin{bmatrix}
b_1^{(1)} A_{11}^{(1)} & b_1^{(1)} A_{12}^{(1)} \\
1 & (A_{12}^{(1)})^2
\end{bmatrix}
\begin{bmatrix}
(h_1^{(1)})'' \\
(h_2^{(1)})''
\end{bmatrix} = 0
\]

In order to prove that \( h_1^{(1)} \)'s are affine functions, we need to show that \( (h_1^{(1)})'' = 0 \). In fact, it is sufficient to show that the left matrix \( H \) has full rank. Since we can re-express \( H \) in the following form

\[
H = \begin{bmatrix}
A_{11}^{(1)} & A_{12}^{(1)} \\
A_{21}^{(1)} & A_{22}^{(1)}
\end{bmatrix}^\top \begin{bmatrix}
b_1^{(1)} A_{21}^{(1)} \\
0
\end{bmatrix}.
\]

It is readily seen that \( \text{rank}(H) = 2 \). Indeed, for the left matrix, it is full rank almost surely since \( A \) is drawn from an absolutely continuous distribution. By the same reason, neither \( A_{21}^{(1)} \) nor \( A_{22}^{(1)} \) is zero with probability one. Since \( \|b^{(1)}\|_0 = 2 \), \( H \) has to be full rank. Therefore, the product of these two matrices is still full rank which leads to the fact that \( (h_1^{(1)})'' = 0 \). By symmetry, one can readily show that \( (h_2^{(2)})'' = 0 \).

We also need to show that \( \alpha_i^{(q)} \neq 0 \). This is trivial since (9c) holds. That is, \( h_i(x) = f_i^{(q)} \circ g_i^{(g)}(x) \) is a composition of two invertible function—thereby being invertible itself. This excludes the possibility of \( \alpha_i^{(q)} \) being zero—which would have made \( h_i(x) \) non-invertible.
Theorem 1 shows that even if we have different view-specific components for multiview data, we can always identify the shared components under mild assumptions. This property is quite appealing since no strong assumptions like statistical independence, non-negativity or simplex structure is required for \( s_\ell \)—as opposed to existing frameworks [24–27,29]. In addition, there is no constraint on the energy of each component—i.e., even if the energy of the shared component is significantly smaller compared to that of the view-specific component, the proposed criterion can still recover the shared subspace. This property is the same as that of the classic linear CCA [4].

In the proof of Theorem 1, we have implicitly assume that there at least exists a solution \( b^{(q)} \) to Problem (9) such that \( \|b^{(q)}\|_0 = M_q \). In fact, this holds for more general case where \( B^{(q)} \) is a \( K \times M_q \) matrix. We have the following proposition:

**Proposition 1** Assume that \( A^{(q)} \in \mathbb{R}^{M_q \times (K+R_q)} \) is drawn from a joint absolutely continuous distribution, where \( M_q \geq K + R_q \). Then, there always exists a solution \( B^{(q)} \in \mathbb{R}^{R \times M_q} \) of Problem (9) that satisfies

\[
\|B^{(q)}\|_0 = RM_q, \ 1 \leq R \leq K,
\]

with probability one.

The proof is relegated to Appendix 8.

With the insights gained from the 2 \( \times \) 2 case and Proposition 1, we are ready to show the subspace identifiability of the general case. We have the following theorem:

**Theorem 2 (Nonlinearity Removal)** Consider the same nonlinear mixing model as in (7). Assume that

\[
M_q \geq K + R_q, \ q = 1, 2,
\]

and that the mixing matrices \( A^{(q)} \) are drawn from certain absolutely continuous distributions. Suppose that \( (B^{(q)}, f^{(q)}) \) for \( q = 1, 2 \) satisfies (9b) and (9c) with \( 1^T B^{(q)} \|_0 = M_q \). Then, we have the following holds almost surely:

\[
h_{i}^{(q)}(x) = f_{i}^{(q)} \circ g_{i}^{(q)}(x) = \alpha_{i}^{(q)} x + d_{i}^{(q)}, \ \alpha_{i}^{(q)} \neq 0, \ \forall i,
\]

i.e., the composition is an affine function with probability one.

The proof is relegated to Appendix 7.

To summarize, we have the following proposition:

**Corollary 1 (Subspace Identifiability)** Under the generative model (7), assume that a feasible solution of Problem (9), denoted by \( B^{(q)}_* \) and \( f^{(q)}_* \) for \( q = 1, 2 \), can be found, where \( \|B^{(q)}_*\|_0 = K M_q \). Then, we have

\[
B^{(q)}_* f^{(q)}_*(y^{(q)}_{\ell}) = \Theta s_\ell, \ q = 1, 2,
\]

for a certain nonsingular \( \Theta \in \mathbb{R}^{K \times K} \).

The above can be shown by combining Theorem 2 and the identifiability theorem of linear CCA [4].
### 3.4 Related Works

We should mention there are two works that are close to our intention. A classic work in [20] utilizes kernel CCA to solve the ICA problem under the linear mixture model. The method splits \( y_\ell = As_\ell \) to \( y_\ell^{(1)} = A^{(1)}s_\ell \) and \( y_\ell^{(2)} = A^{(2)}s_\ell \), where \( A = [A^{(1)}, A^{(2)}] \), and then applies kernel CCA to much the transformed \( y_\ell^{(1)} \) and \( y_\ell^{(2)} \)—through which \( s_\ell \) identification can be achieved. The model resorts to nonlinear CCA, while the model is not nonlinearly distorted. Another work in [46] considers the model in (3), i.e., \( y_\ell = g(As_\ell) \). Algorithms were derived for removing nonlinearity, again, through splitting the received signals two too parts, i.e., \( y_\ell^{(1)} \) and \( y_\ell^{(2)} \), and maximizing the correlation between transformed \( y_\ell^{(1)} \) and \( y_\ell^{(2)} \). This is effectively multiview matching as we did. Nevertheless, the work does not consider view-specific components, and has strong Gaussianity assumption on \( x_\ell \), where \( x_\ell = As_\ell \), for establishing identifiability of \( g(\cdot) \).

### 4 Practical Implementation

The identifiability theorems indicate that the nonlinear functions \( g^{(q)}(\cdot) \) in the data generation process for \( q = 1, 2 \) may be removed up to affine transformations—if one can find a solution to Problem (9). Problem (9) is nevertheless more of a conceptual formulation that requires practical approximations under finite samples and workable parametrization for \( f^{(q)} \) and \( g^{(q)} \). In this section, we cast (9) into a numerical optimization-friendly form and propose an algorithm for tackling the reformulated problem.

#### 4.1 Parametrization for Nonlinear Functions

To tackle Problem (9), we first parameterize \( f^{(q)} \) and \( g^{(q)} \) as two neural networks. Since we seek for continuous functions, neural networks are good candidates since they are the so-called “universal function approximators”. Note that one-hidden-layer networks can already represent all continuous functions over bounded domains to \( \epsilon \)-accuracy with a finite number of neurons [53]. Nevertheless, we keep our parameterization flexible to incorporate multiple layers—which have proven effective in practice [30]. In addition, many established network configurations and architectures can be incorporated into our framework. This is particularly of interest since multiview data oftentimes are with quite diverse forms, e.g., image and text. Using some established neural network paradigms for different types of data (e.g., CNN for image data [30]) may enhance performance.

In our case, since we consider \( g_m^{(q)}(\cdot) : \mathbb{R} \to \mathbb{R} \) as a nonlinear function that is independent with others, we parameterize it as follows:

\[
g_m^{(q)}(x) = (w_L^{(q)})^\top \times 
\sigma \left( W_L^{(q)} \ldots \sigma \left( W_2^{(q)} \sigma \left( w_1^{(q)} x + \gamma_1^{(q)} \right) + \gamma_2 \right) \ldots + \gamma_{L-1}^{(q)} \right),
\]

where \( w_1^{(q)} \in \mathbb{R}^{N_1} \) and \( w_L^{(q)} \in \mathbb{R}^{N_L} \) are the network weights of the input and output layers, respectively, \( W_l^{(q)} \in \mathbb{R}^{N_l \times N_{l-1}} \) denotes the network weights from layer \( l - 1 \) to layer \( l \), \( \gamma_l^{(q)} \in \mathbb{R}^{N_l} \) is the bias term of layer \( l \), and \( \sigma(x) \) is the so-called activation function. One typical activation function is the sigmoid function, i.e.,

\[
\sigma(x) = \frac{1}{1 + e^{-x}}.
\]
Figure 1: The proposed neural network based implementation.

\[ \text{sigmoid}(x) = \frac{1}{1+e^{-x}} \]

\[ \text{tanh}(x) = \frac{e^{2x} - 1}{e^{2x} + 1} \]

\[ \text{relu}(x) = \begin{cases} 0, & x < 0 \\ x, & x \geq 0 \end{cases} \]

Figure 2: Some frequently used activation functions.

There exist many other choices, e.g., the \text{tanh} function, and the so-called \textit{rectified linear unit} \text{ReLU} function \cite{30}; see Fig. 2 for some examples. Note that different configurations of \( W^{(q)} \) and \( \sigma(\cdot) \) lead to different types of neural networks; see \cite{30}. For conciseness, we use \( \theta_f^{(q)} \) and \( \theta_g^{(q)} \) to denote the network parameters in the neural networks representing \( f^{(q)} \) of \( g^{(q)} \) for \( q = 1, 2 \), respectively.

### 4.2 Reformulation

With neural network-based parametrization for \( f^{(q)} \) and \( g^{(q)} \), we consider the following optimization problem:

\[
\begin{align*}
\text{minimize} & \quad \sum_{q=1}^{2} \sum_{\ell=1}^{N} \left\| \mathbf{u}_\ell - B^{(q)} f^{(q)} (y^{(q)}_\ell) \right\|_2^2 \\
& \quad + \lambda \sum_{q=1}^{2} \sum_{\ell=1}^{N} \left\| y^{(q)}_\ell - g^{(q)} (f^{(q)} (y^{(q)}_\ell)) \right\|_2^2 \\
\text{subject to} & \quad \frac{1}{N} \left[ \sum_{\ell=1}^{N} \mathbf{u}_\ell \mathbf{u}_\ell^\top \right] = \mathbf{I}, \\
& \quad \frac{1}{N} \sum_{\ell=1}^{N} \mathbf{u}_\ell = \mathbf{0},
\end{align*}
\]

\((11)\)
where \( U = [u_1, \ldots, u_N] \in \mathbb{R}^{M \times N} \) represent the extracted shared components, where \( N \) is the number of samples, \( y^{(q)}_\ell \) is the \( \ell \)th input data for the \( q \)th view, \( f^{(q)}(\cdot) = [f^{(q)}_1(\cdot), \ldots, f^{(q)}_{Mq}(\cdot)]^\top \) is element-wise non-linear mapping that we aim at learning for nonlinearity removal, and \( g^{(q)}(\cdot) = [g^{(q)}_1(\cdot), \ldots, g^{(q)}_{Mq}(\cdot)]^\top \) is another nonlinear function whose presence will be explained shortly.

To explain the formulation, the first term in the cost function is a translation of (9b) via introducing a slack variable \( U \). The second term is to enforce \( f^{(q)} \) to be invertible, at least for the available data samples \( y^{(q)}_\ell \) for \( \ell = 1, \ldots, N \). Note that if \( f^{(q)} \) is invertible, then there exists \( g^{(q)} \) such that the second term is zero—but the converse is not necessarily true. Nonetheless, using such a data reconstruction to prevent \( f^{(q)} \) from being an irreversible function in general works, especially when the number of samples is large. This idea is known as the autoencoder [30], which is considered a nonlinear counterpart of PCA. The term \( UU^\top = I \) corresponds to (9d). An illustration of the proposed neural network based implementation is shown in Fig. 1.

We should highlight the constraint

\[
(1/N) \sum_{\ell=1}^N u_\ell = 0,
\]

i.e., the extracted matrix \( U \) has a zero column mean—which was not listed in the conceptual formulation in (9). Adding this constraint effectively means that we aims at extracting \( \Theta \tilde{S} \) where

\[
\tilde{S} = \left( I - (1/N)11^\top \right) S,
\]

i.e., the row-range space of the column mean removed \( S \). This is reasonable since, in many cases (e.g., speech processing and brain signal processing), the interesting and informative part of \( s_\ell \) is the variation over \( \ell \), rather than the mean. Hence, assuming \( s_\ell \) to have zero mean across different dimensions is reasonable. In addition, adding this constraint is in fact quite vital for avoiding numerical problems. To see this subtle point, recall that according to Theorems 1-2, we have

\[
f^{(q)}(y^{(q)}_\ell) = DA^{(q)} s^{(q)}_\ell + d^{(q)},
\]

where \( D^{(q)} = \text{Diag}(\alpha^{(q)}_1, \ldots, \alpha^{(q)}_{Mq}) \). If we directly match \( f^{(1)}(y^{(1)}_\ell) \) with \( f^{(2)}(y^{(2)}_\ell) \), i.e., enforcing

\[
B^{(1)} f^{(1)}(y^{(1)}_\ell) = B^{(1)} f^{(2)}(y^{(2)}_\ell) = u_\ell,
\]

then, one trivial solution is to simply making \( D \approx 0 \) and \( d^{(1)} = d^{(2)} \), with \( B^{(1)} = B^{(2)} \)—i.e., the constant \( d^{(q)} \) can easily dominate. Hence, we enforce \( u_\ell \) to be zero mean, which will automatically take out \( d^{(q)} \).

**Remark 1** Note that we do not constrain \( \|1^\top B^{(q)}\|_0 = M_q \) in our working formulation since it seems that \( B^{(q)} \) with zero column sums rarely happens (it never happened in our extensive experiments). Hence, incorporating such a hard constraint just for ‘safety’ in theory may not be worthy—a lot more complex algorithms may be required for handling this constraint.

### 4.3 Proposed Algorithm

We propose a block coordinate descent (BCD)-based algorithm.
4.3.1 The \((f,g,B)\)-Subproblem

We first consider the problem of updating the neural networks when fixing \(U\). This is an unconstrained optimization problem and can be handled by gradient descent. Denote the loss function as

\[ L(\theta, U) = \sum_{\ell=1}^{N} L_{\ell}(\theta, U), \]

where \(\theta\) collects \(\theta_{h}^{(q)}, \theta_{g}^{(q)}\) and \(B^{(q)}\) for \(q = 1, 2\), and

\[
L_{\ell}(\theta, U) = \sum_{q=1}^{2} \| u_{\ell} - B^{(q)} f^{(q)}(y_{\ell}^{(q)}) \|^2_2 \\
+ \lambda \sum_{q=1}^{2} \| y_{\ell}^{(q)} - g^{(q)}(f^{(q)}(y_{\ell}^{(q)})) \|^2_2.
\]

At iteration \(t\), the update rule is simply

\[
\theta_{t+1} \leftarrow \theta_{t} - \gamma_{t} \nabla_{\theta} L(\theta_{t}, U_{t}).
\]

(12)

where \(\gamma_{t}\) is the step size chosen for the \(t\)th update.

Note that computing \(\nabla_{\theta} L(\theta_{t}, U_{t})\) is normally not easy, since computing gradient of neural networks per se is a resource-consuming process—the gradient normally requires backpropagation (BP) based algorithms to compute in a sample-by-sample manner if multiple layers are involved. Hence, instead of using the full gradient to update \(\theta\), one can also use stochastic gradient, i.e.,

\[
\theta_{t+1} \leftarrow \theta_{t} - \gamma_{t} \sum_{\ell \in I_{t}} \nabla_{\theta} L_{\ell}(\theta_{t}, U_{t}),
\]

(13)

where \(I_{t}\) is an index set randomly sampled at iteration \(t\).

4.3.2 \(U\)-Subproblem

To update \(U\), we solve the following subproblem

\[
\min_{U} \sum_{q=1}^{2} \sum_{\ell=1}^{N} \| u_{\ell} - B^{(q)} f^{(q)}(y_{\ell}^{(q)}) \|^2_2 \\
\text{s.t.} \quad UU^{\top} = I, \quad \frac{U1}{N} = 0.
\]

(14)

This problem is seemingly difficult since it has two constraints, in which one is nonconvex. However, this problem turns out to have a semi-algebraic solution. To see this, we show the following lemma:

**Lemma 1** Consider the following optimization problem

\[
\min_{U} \|U - Z\|_F^2 \\
\text{subject to} \quad UU^{\top} = I, \quad \frac{U1}{N} = 0.
\]

(15a)

(15b)
An optimal solution is

\[ U_\star = PQ^\top \]

where \( P \) and \( Q \) are left and right singular vectors of \( ZW \) respectively, with \( W = I - \frac{1}{N}11^\top \).

**Proof:** The proof is relegated to Appendix 9.

One can see that the \( U \)-subproblem can be re-expressed as

\[
\min_U \left\| U - \frac{1}{2} \sum_{q=1}^2 B_{t+1}^{(q)}F_{t+1}^{(q)} \right\|_F^2
\]

s.t. \( UU^\top = I, \quad U1_N = 0 \),

where \( F^{(q)} = [f^{(q)}(y_1^{(q)}), \ldots, f^{(q)}(y_N^{(q)})] \). The above can be shown by expanding (14):

\[
\sum_{q=1}^2 \sum_{\ell=1}^N \left\| u_\ell - B_{t+1}^{(q)}f_{t+1}^{(q)}(y_\ell^{(q)}) \right\|_2^2
\]

\[ = \sum_{q=1}^2 \left\| U - B_{t+1}^{(q)}F_{t+1}^{(q)} \right\|_F^2
\]

\[ = \sum_{q=1}^2 \left\| U \right\|_F^2 - 2 \text{Tr}(UU^\top B_{t+1}^{(q)}F_{t+1}^{(q)}) + \left\| B_{t+1}^{(q)}F_{t+1}^{(q)} \right\|_F^2.
\]

Since the first term and the last term are constants, we have the following equivalent problem:

\[
\min_U \sum_{q=1}^2 -\text{Tr}(UU^\top B_{t+1}^{(q)}F_{t+1}^{(q)})
\]

s.t. \( UU^\top = I, \quad U1_N = 0 \),

which is exactly equivalent to (16). Then, a solution can be obtained via applying Lemma 1.

The overall algorithm is summarized in Algorithm 1. One can see that the algorithm does not have computationally heavy updates. In practice, one can update \( \theta \) multiple times until switching to the next block—which often improve the speed for convergence according to our empirical observations.

**Remark 2** We should mention that although our technical part was developed under \( q = 1, 2 \), the theorems and algorithm naturally hold when one has \( Q \geq 3 \) views. Another remark is that many off-the-shelf tricks for speeding up training neural networks, e.g., adaptive step size and momentum [54,55], can also be employed to determine \( \gamma_{t,s} \), which can normally speed up convergence.

### 5 Numerical Results

In this section, we apply the proposed algorithm to a number of synthetic-data experiments to showcase the effectiveness and validate our analysis.
Algorithm 1: Nonlinear Multiview Component Analysis (NMCA)

Data: Data matrices $Y^{(q)} \in \mathbb{R}^{M_q \times N}$, estimated dimension of the latent space and network structures.

Result: $U, B^{(q)}$ and $\theta$

1. Initialize $U_0 \leftarrow PQ^\top$ where $PDQ^\top = \sum_{q=1}^{Q} F_0^{(q)}$, $F_0^{(q)} = B_0^{(q)} f_0^{(q)} (Y^{(q)})$ with zero row mean;
2. $t \leftarrow 1$;
3. while stopping criterion is not reached do
   4. $\theta_{s,0} \leftarrow \theta_t$;
   5. $s \leftarrow 1$;
   6. while stopping criterion is not reached do
      7. $\theta_{t,s+1} \leftarrow \theta_{t,s} - \gamma_{t,s} \sum_{\ell \in I_{t,s}} \nabla_{\theta_{t,s}} L_{\ell}(\theta_{t,s}, U_t)$;
      8. $s \leftarrow s + 1$;
   9. end
10. $\theta_{t+1} \leftarrow \theta_{t,s}$;
11. $U_{t+1} \leftarrow PQ^\top$ where $PDQ^\top = (\sum_{q=1}^{Q} F_{t+1}^{(q)}) W$; $t \leftarrow t + 1$;
12. end

5.1 Setup

Baselines In this section, we employ a number of baselines to benchmark the proposed approach:

1. Principal Component Analysis (PCA): PCA is widely used for feature extraction and dimensionality reduction. The first principal component has the largest possible variance. We will show that PCA is not able to recover the shared components when the noise dominates.

2. Canonical Correlation Analysis (CCA) [2]: CCA finds linear transformations of the two views that exhibit maximum correlation with each other.

3. Kernel CCA (KCCA) [56]: KCCA matches the two-views in a high dimension feature space via certain nonlinear kernel functions. The baseline from [56] is a scalable version of KCCA.


5. Deep Canonically Correlated AutoEncoders (DCCAE) [21]: DCCAE utilizes autoencoder on top of DCCA, to avoid trivial solutions.

Performance Metric: Per Corollary 1, the nonlinear multiview analysis recovers the range space of $S^\top$, where $S = [s_1, \ldots, s_N]$. Therefore, we employ the the following subspace distance measure

$$\text{dist}(S,U) = \|P_s \perp Q_u^\top\|_2$$

as the performance metric, where $S = \text{range}(S^\top)$ and $U = \text{range}(U^\top)$, $P_s \perp$ is defined as [7]:

$$P_s \perp = I - S^\top (SS^\top)^{-1} S.$$
and $Q_u$ is the orthogonal basis of $U$. Note that $\|X\|_2$ here denotes the matrix 2-norm, i.e., the largest singular value of $X$. The metric is bounded within $[0, 1]$. If $U \approx \Theta S$, then $\|P_s^\perp Q_u^\top\|$ should be close to zero.

Algorithm 1 is implemented as follows. The $\theta$ update is implemented with multiple updates in each iteration (i.e., 100 times for each fixed $U_t$). The batch size $|I_t|$ is 1,000. The Adam algorithm [55] is employed for updating $\theta$. The initial step size is set to be $10^{-3}$ in all the experiments. The algorithm is stopped after 5,000 epochs in total.

5.2 Simulations

In the first experiment, we construct the views as follows. The shared component matrix $s_\ell \in \mathbb{R}^2$ is randomly sampled point from a parabola ($y = x^2, x \in [-1, 1]$). Besides, the sample mean is subtracted. The shared components $s_\ell \in \mathbb{R}^2$ for $\ell = 1, \ldots, N$ are shown in Fig. 3. View-specific components $c_q(\ell) \in \mathbb{R}^1$ for $q = 1, 2$ are set to be zero-mean unit-variance i.i.d. Gaussian components with different means (0.5 and -0.8, respectively) and variances (1 and 1.5, respectively). Hence, we have $M_1 = 3$ and $M_2 = 3$. The sample size for each view is $N = 1,000$. The elements of the mixing matrices $A^{(1)} \in \mathbb{R}^{3 \times 3}, A^{(2)} \in \mathbb{R}^{3 \times 3}$ follow zero-mean unit-variance i.i.d. Gaussian distribution. The nonlinear functions employed are as follows: $g_1^{(1)}(x) = 3 \text{sigmoid}(x) + 0.1x$, $g_2^{(1)}(x) = 5 \text{sigmoid}(x) + 0.2x$, $g_3^{(1)}(x) = \exp(x)$, $g_1^{(2)}(x) = 5 \text{tanh}(x) + 0.2x$, $g_2^{(2)}(x) = 2 \text{tanh}(x) + 0.1x$, $g_3^{(2)}(x) = x^3 + x$. Fig. 4 visualizes the observed views (via t-SNE [57]). One can see the views are quite severely distorted. In the experiment, our method uses a dimension-wise neural network structure in order to learn the inverse of the nonlinear mapping. Every network has one-hidden-layer with 256 neurons. We adopt the ReLU activation function to act as a neuron.

The learned $\hat{x}_\ell = B^{(q)}f(y^{(q)}_\ell)$ by the proposed approach and the outputs of the baselines are visualized in Fig. 5. One can see that the proposed approach yields a scaled and permuted version of the parabola in Fig. 3—which is as expected. The baselines are not as promising: DCCA and DCCAE also work reasonably—but the scatter plots are much noisier. The linear mapping based methods PCA and CCA clearly failed.

Fig. 6 shows the learned $\hat{f}_m^{(q)} \circ g_m^{(q)}$ for $m = 1, 2, 3$ and $q = 1, 2$. One can see that in this simulation, all the composition functions are visually affine.

Table 1 shows the averaged $\text{dist}$ from 10 random trials under the same settings as those for in Figs. 5-6. Several observations are in order: First, the proposed approach admits a $\text{dist}$ value that is almost zero, validating our subspace identifiability claim. It is much lower than those of the baselines, perhaps because our method exploits model information. Second, subtracting the mean

Figure 3: The shared 2-dimensional components.
Figure 4: Visualization of view 1 (left) and view 2 (right).

Figure 5: Example of the recovered shared component. From top-left to bottom-right: proposed, proposed without zero-mean constraint, DCCA, DCCAE, KCCA, CCA and PCA

<table>
<thead>
<tr>
<th>method</th>
<th>Proposed</th>
<th>Proposed w/o zero-mean</th>
<th>DCCA</th>
</tr>
</thead>
<tbody>
<tr>
<td>dist</td>
<td>0.024(3.55e^{-3})</td>
<td>0.996(5.62e^{-4})</td>
<td>0.339(3.73e^{-2})</td>
</tr>
<tr>
<td>method</td>
<td>DCCAE</td>
<td>KCCA</td>
<td>CCA</td>
</tr>
<tr>
<td>dist</td>
<td>0.281(1.83e^{-2})</td>
<td>0.532(3.90e^{-2})</td>
<td>0.971(4.60e^{-3})</td>
</tr>
<tr>
<td>method</td>
<td>PCA</td>
<td></td>
<td></td>
</tr>
<tr>
<td>dist</td>
<td>0.998(1.67e^{-3})</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
this constraint is much worse. In our formulation [cf. the constraint (1)]

Figure 6: The nonlinear distortion and the learned composite functions of two views. Left: view 1; Right: view 2. From top to bottom: dimension 1 to 3.

Table 2: The \( \text{dist} \) value [mean (standard deviation)] of the approaches against number of samples.

<table>
<thead>
<tr>
<th># samples</th>
<th>500</th>
<th>1000</th>
<th>2000</th>
<th>5000</th>
<th>10000</th>
</tr>
</thead>
<tbody>
<tr>
<td>proposed</td>
<td>0.023(3.72e(^{-3}))</td>
<td>0.024(3.55e(^{-3}))</td>
<td>0.010(7.29e(^{-4}))</td>
<td>2.98(^{-4})(2.26e(^{-4}))</td>
<td>2.13e(^{-3})(1.21e(^{-3}))</td>
</tr>
<tr>
<td>DCCA</td>
<td>0.514(6.32e(^{-2}))</td>
<td>0.339(3.73e(^{-2}))</td>
<td>0.388(1.78e(^{-2}))</td>
<td>0.293(6.21e(^{-2}))</td>
<td>0.252(7.35e(^{-3}))</td>
</tr>
<tr>
<td>DCCAE</td>
<td>0.336(6.12e(^{-2}))</td>
<td>0.281(1.83e(^{-2}))</td>
<td>0.331(5.73e(^{-2}))</td>
<td>0.368(4.82e(^{-2}))</td>
<td>0.253(3.08e(^{-2}))</td>
</tr>
<tr>
<td>KCCA</td>
<td>0.570(2.70e(^{-2}))</td>
<td>0.578(6.88e(^{-3}))</td>
<td>0.483(1.65e(^{-2}))</td>
<td>0.493(1.49e(^{-2}))</td>
<td>0.488(8.68e(^{-3}))</td>
</tr>
<tr>
<td>CCA</td>
<td>0.977(6.84e(^{-3}))</td>
<td>0.971(4.60e(^{-3}))</td>
<td>0.977(1.95e(^{-3}))</td>
<td>0.978(4.01e(^{-4}))</td>
<td>0.977(1.26e(^{-3}))</td>
</tr>
<tr>
<td>PCA</td>
<td>0.977(2.71e(^{-2}))</td>
<td>0.998(1.67e(^{-3}))</td>
<td>0.991(8.26e(^{-3}))</td>
<td>0.995(5.53e(^{-4}))</td>
<td>0.990(8.62e(^{-3}))</td>
</tr>
</tbody>
</table>

in our formulation [cf. the constraint \((1/N)\sum_{\ell=1}^{N} u_\ell = 0\)] is indeed crucial, since the result without this constraint is much worse.

Table 2 shows the performance of the algorithms when the number of samples \(N\) changes. Note that the nonlinearity removal theorems are derived under the population case, i.e., the functional equations in (9b) hold for all \(s_\ell \in \mathcal{S}\) and \(c_{q(\ell)} \in \mathcal{C}_q\). This is not attainable in practice, where we only have a finite number of samples. Hence, it is of interest to observe how the performance scales with \(N\). From Table 2 one can see that the \(\text{dist}\) value of the proposed approach clearly decreases when the number of samples increases from 500 to 10,000. On the other hand, using 500 samples, the proposed approach already exhibits good performance.

To observe the impact of the view-specific interference, we define the **Shared Component to Interference Ratio (SCIR)**

\[
\text{SCIR} = 10 \log_{10} \left( \frac{\|S\|_{\mathcal{F}}^2/\mathcal{K}}{\frac{1}{Q} \sum_{q=1}^{Q} ||C(q)||_{\mathcal{F}}^2/R_q} \right) \text{ dB}
\]
Table 3: The dist value [mean (standard deviation)] of the approaches SCIR.

<table>
<thead>
<tr>
<th>SCIR</th>
<th>-10 dB</th>
<th>-20 dB</th>
</tr>
</thead>
<tbody>
<tr>
<td>proposed</td>
<td>0.024 (3.55e⁻³)</td>
<td>0.022 (1.26e⁻³)</td>
</tr>
<tr>
<td>DCCA</td>
<td>0.339 (3.73e⁻²)</td>
<td>0.324 (8.58e⁻³)</td>
</tr>
<tr>
<td>DCCAE</td>
<td>0.281 (1.83e⁻²)</td>
<td>0.339 (1.70e⁻²)</td>
</tr>
<tr>
<td>KCCA</td>
<td>0.578 (6.88e⁻³)</td>
<td>0.351 (1.73e⁻²)</td>
</tr>
<tr>
<td>CCA</td>
<td>0.971 (4.60e⁻³)</td>
<td>0.985 (5.21e⁻³)</td>
</tr>
<tr>
<td>PCA</td>
<td>0.998 (1.67e⁻³)</td>
<td>0.995 (4.00e⁻³)</td>
</tr>
</tbody>
</table>

Table 4: Different structures of neural network

<table>
<thead>
<tr>
<th># params of each dim</th>
<th>structure</th>
<th>distance</th>
</tr>
</thead>
<tbody>
<tr>
<td>≈ 512</td>
<td>[1,256,1]</td>
<td>2.42e⁻² (3.55e⁻³)</td>
</tr>
<tr>
<td></td>
<td>[1,16,16,16,1]</td>
<td>2.94e⁻² (3.19e⁻³)</td>
</tr>
<tr>
<td>≈ 2048</td>
<td>[1,1024,1]</td>
<td>2.73e⁻² (2.15e⁻³)</td>
</tr>
<tr>
<td></td>
<td>[1,32,32,32,1]</td>
<td>2.06e⁻² (5.57e⁻³)</td>
</tr>
<tr>
<td>≈ 8192</td>
<td>[1,4096,1]</td>
<td>4.78e⁻² (3.41e⁻³)</td>
</tr>
<tr>
<td></td>
<td>[1,64,64,64,1]</td>
<td>1.54e⁻² (3.04e⁻³)</td>
</tr>
</tbody>
</table>

Table 3 shows the result under different SCIRs. One can see that even if the ratio is -20 dB, the performance of the proposed approach is still very good—the average subspace distance metric is 0.022, whereas the best-performing competitor (i.e., DCCA) can only reach an average value of 0.324. This observation is consistent with our analysis, as well as the interference-robust property of linear CCA [4].

Table 4 shows the dist value against different network structures. Observing this is of interest, since in machine learning tuning the structure of networks to fit a task can sometimes be tedious. The proposed framework is unsupervised, which means that there may not even be training samples for us to tune the network structure. Fortunately, the observation here is that the result is not heavily affected by the choice of neural network structures—although using more parameters and more layers could slightly improve performance.

In Fig. 7, we test that to what extent can the nonlinearity be solved by the proposed model. Specifically, we set the nonlinear function \( f(x) = \alpha x^3 + x \), then we try different values of \( \alpha = 0.01, 0.1, 1, 10 \).

to see how well is the nonlinear function is removed. In this simulation, the shared components are zero-mean i.i.d. Gaussian variables and \( K = 3 \), while the view-specific components for both views follow the same distribution with \( R_1 = R_2 = 2 \). One can see that when \( \alpha \) is increasing, the performance of the proposed approach indeed deteriorates, given the same computational resources, as expected. Nonetheless, even when \( \alpha = 0.5 \), one can see that the nonlinearity imposed on the views is largely removed.

In Table 5, we present the results of the algorithms under different \( K \), i.e., the dimension of the shared components. We fix the observation dimension to be 5 and the \( K \) ranges from 1 to 4. One can see that different \( K \) settings essentially do not affect the results—which is consistent with
Figure 7: The nonlinear distortion and the learned composite functions of one example dimension. From left to right: \( \alpha \) is 0.01, 0.05, 0.1, 0.5, 1 and the corresponding recovered subspace metric is 0.011, 0.014, 0.019, 0.158, 0.345

<table>
<thead>
<tr>
<th>source dim</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>metric</td>
<td>0.032(1.00e^{-3})</td>
<td>0.016(2.56e^{-3})</td>
<td>0.015(2.62e^{-3})</td>
<td>0.015(3.34e^{-3})</td>
</tr>
</tbody>
</table>

CCA identifiability analyses in [4] and this work.

6 Conclusion

In this work, we revisited the multiview analysis problem under a nonlinear setting. We showed that under a class of unknown nonlinear distortion, the appealing identifiability guarantees for the classic linear multiview analysis still holds—if the identification criterion is properly formulated. We also proposed a neural network-based implementation for the proposed criterion, as well as an inexact BCD-based optimization algorithm. A suite of numerical results were presented to corroborate our analysis.

References


7 Proof of Theorem 2

We show the case that $M_1 = K + R_1$ and $M_2 = K + R_2$, i.e., the mixing matrices $A^{(1)}$ and $A^{(2)}$ are square nonsingular matrices. The case where $M_q > K + R_q$ can then be shown via a straightforward approach: The $M_q$ observations output by $g^{(q)}(\cdot)$ can be divided to (possibly overlapped) groups, each of which has $K + R_q$ channels.

It suffices to show that the nonlinearity can be removed when $R = 1$, i.e.,

$$B^{(q)} = (b^{(q)})^T$$

is a row vector. In this case, we consider only extracting one shared component. From Eq. (9), we have the following equation

$$
\sum_{m=1}^{M_1} b_m^{(1)} h_m^{(1)} \left( \sum_{k=1}^{K} A_{m,k}^{(1)} s_k + \sum_{r=1}^{R_1} A_{m,K+r}^{(1)} c_r^{(1)} \right)
= \sum_{j=1}^{M_2} b_j^{(2)} h_j^{(2)} \left( \sum_{l=1}^{K} A_{j,l}^{(2)} s_l + \sum_{s=1}^{R_2} A_{j,K+s}^{(2)} c_s^{(2)} \right).
$$

Denote $\phi(c^{(1)}, c^{(2)}, s)$ as

$$
\phi(c^{(1)}, c^{(2)}, s) =
\sum_{m=1}^{M_1} b_m^{(1)} h_m^{(1)} \left( \sum_{k=1}^{K} A_{m,k}^{(1)} s_k + \sum_{k=1}^{K} A_{m,K+c_k^{(1)}}^{(1)} \right) - \sum_{j=1}^{M_2} b_j^{(2)} h_j^{(2)} \left( \sum_{l=1}^{K} A_{j,l}^{(2)} s_l + \sum_{l=1}^{K} A_{j,K+c_l^{(2)}}^{(2)} \right).
$$
Let us first consider $h^{(1)}$, by taking derivatives
\[
\frac{\partial^2 \phi(c^{(1)}, c^{(2)}, s)}{\partial c_1^{(1)} \partial s_k} = 0, \quad k = 1, \ldots, K
\]
which leads to $K$ linear equations w.r.t. $(h_m^{(1)})''$ for $m = 1, \ldots, M_1$. In addition
\[
\frac{\partial^2 \phi(c^{(1)}, c^{(2)}, s)}{\partial c_1^{(1)} \partial c_r^{(1)}} = 0, \quad r = 1, \ldots, R_1,
\]
yields another $R_1$ linear equations. In total, we have collected $K + R_1$ linear equations as follows:
\[
\sum_{m=1}^{M_1} b_m^{(1)} A_m^{(1)} A_{m,K+1} (h_m^{(1)})'' = 0, \quad k = 1, \ldots, K + R_1.
\]

It can be rewritten as the following linear system:
\[
(A^{(1)})^\top \begin{bmatrix}
  b_1^{(1)} A_{1,K+1} \\
  \vdots \\
  b_{M_1}^{(1)} A_{M_1,K+1}
\end{bmatrix} \begin{bmatrix}
  (h_1^{(1)})'' \\
  \vdots \\
  (h_{M_1}^{(1)})''
\end{bmatrix} = 0,
\]
which holds for all $s_\ell \in S$ and $c_\ell^{(q)} \in C_q$, where $A^{(1)} \in \mathbb{R}^{M_1 \times M_1}$ is the mixing matrix, which is full rank since it is drawn from an absolutely continuous distribution.

Now what remains is to show that the diagonal matrix in the middle is full rank which is equivalently that all the diagonal elements are non-zero. Since $A^{(1)}$ is assumed to be continuously random variable, the probability that any element of it being zero is zero.

Therefore, we only need to show that $b_1^{(1)}, \ldots, b_{M_1}^{(1)}$ are not zeros. By the assumption that \(\|B^{(q)}\|_0 = RM_q\), none of $b^{(1)}$ will be zero. As a result, the matrix in the middle is full rank. Consequently, $(h^{(1)})''$ has to be a all-zero vector. In addition, Proposition 1 guarantees that there exists such a solution $b^{(1)}$. By applying the same proof to the second view, one can show that $h^{(2)}$ is also an affine mapping.

**8 Proof of Proposition 1**

To show this, it suffices to construct a solution such that $1^\top B^{(q)}$ has no zero elements. Let $f^{(q)} \circ g^{(q)} = I$. Then, one solution of $B^{(q)}$ is
\[
B^{(q)} = \Pi^\top [\Theta, 0] (A^{(q)})^\dagger.
\]

The above solution is nothing but selecting out some columns of $(A^{(q)})^\dagger$. Recall that we have assume that $A^{(q)}$ is drawn from a certain joint absolutely continuous distribution, which means
that \((A^{(q)})^\dagger\) also follows a certain absolutely continuous distribution [58]—thereby the elements of \(B^{(q)}\). Hence, the following holds:

\[
\text{Prob}\left( B^{(q)}(i, j) = 0 \right) = 0,
\]

with probability one.

9 Proof of Lemma 1

First, notice that the rows of \(U\) live in the null space of \(1\). Hence, we can re-express \(U\) by

\[
U = VW,
\]

where \(W = I - \frac{1}{N}11^\top\). Hence, Problem (15) can be written as the following equivalent problem:

\[
\begin{align*}
\min_V & \|VW - Z\|_F^2 \\
\text{s.t.} & \quad VWV^\top V^\top = I.
\end{align*}
\]

By expanding the above, we have:

\[
\begin{align*}
\min_V & \|VW\|_F^2 - 2 \text{Tr}(VWZ^\top) + \|Z\|_F^2 \\
\text{s.t.} & \quad VWV^\top V^\top = I.
\end{align*}
\]

The last term \(\|Z\|_F^2\) is a constant, it can be replaced with \(\|ZW\|_F^2\). The middle term also can be written as

\[
-2 \text{Tr}(VWW^\top Z^\top)
\]

since \(W\) is an orthogonal complement projector with \(W = W^\top\) and \(W = W^2\). Thus, we have the reformulated problem:

\[
\begin{align*}
\min_V & \|VW\|_F^2 - 2 \text{Tr}(VWW^\top Z^\top) + \|ZW\|_F^2 \\
\text{s.t.} & \quad VWV^\top V^\top = I.
\end{align*}
\]

It is of the following form:

\[
\begin{align*}
\min_U & \|U - ZW\|_F^2 \\
\text{s.t.} & \quad UU^\top = I.
\end{align*}
\]

Using \(U = VW\) back, we have:

\[
\begin{align*}
\min_U & \|U - ZW\|_F^2 \\
\text{s.t.} & \quad UU^\top = I.
\end{align*}
\]

Expanding the objective again and discarding constant terms, we reach the following equivalent optimization problem:

\[
\begin{align*}
\max_U & \quad \text{Tr}(U^\top ZW) \\
\text{s.t.} & \quad UU^\top = I.
\end{align*}
\]

An optimal solution can be obtained via invoking the Procrustes projection [36, 41]; i.e., a solution is simply taking SVD of \(ZW = PDQ^\top\) and let \(U = PQ^\top\).