Additive Component Analysis

Calvin Murdock
Machine Learning Department
Carnegie Mellon University
cmurdock@cs.cmu.edu

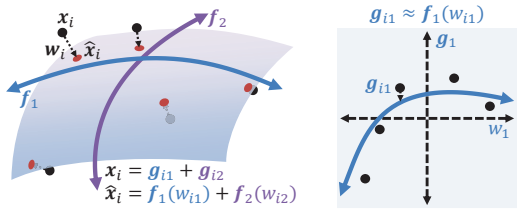
Fernando De la Torre
The Robotics Institute
Carnegie Mellon University
ftorre@cs.cmu.edu

Abstract

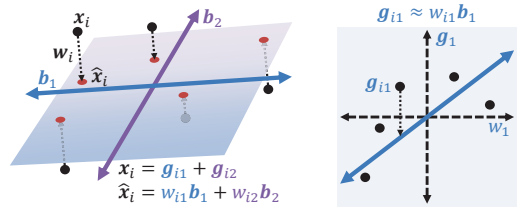
Principal component analysis (PCA) is one of the most versatile tools for unsupervised learning with applications ranging from dimensionality reduction to exploratory data analysis and visualization. While much effort has been devoted to encouraging meaningful representations through regularization (e.g. non-negativity or sparsity), underlying linearity assumptions can limit their effectiveness. To address this issue, we propose Additive Component Analysis (ACA), a novel nonlinear extension of PCA. Inspired by multivariate nonparametric regression with additive models, ACA fits a smooth manifold to data by learning an explicit mapping from a low-dimensional latent space to the input space, which trivially enables applications like denoising. Furthermore, ACA can be used as a drop-in replacement in many algorithms that use linear component analysis methods as a subroutine via the local tangent space of the learned manifold. Unlike many other nonlinear dimensionality reduction techniques, ACA can be efficiently applied to large datasets since it does not require computing pairwise similarities or storing training data during testing. Multiple ACA layers can also be composed and learned jointly with essentially the same procedure for improved representational power, demonstrating the encouraging potential of nonparametric deep learning. We evaluate ACA on a variety of datasets, showing improved robustness, reconstruction performance, and interpretability.

1. Introduction

Identifying the underlying structure of data is one of the most important tasks in machine learning. As technological advancements facilitate the construction of datasets with increasing size and dimensionality, data analysis is becoming more challenging due to computational constraints and the curse of dimensionality. In the field of computer vision especially, data often consist of thousands if not millions of features, resulting in drastically increased training data requirements. However, real-world data often concentrate near manifolds with lower intrinsic dimensionality [27]. For example, while the typical image resolution of digital photographs is large, the space of natural images occupies an extremely small volume in comparison to that of all possible pixel instantiations.



(a) Additive Component Analysis (ACA)



(b) Principal Component Analysis (PCA)

Figure 1: An overview of ACA (a) in comparison to PCA (b) on the task of fitting a two-dimensional surface to three-dimensional data. Both methods minimize the sum of squared distances between the data and their orthogonal projections. However, while PCA learns a linear subspace spanned by basis vectors $\boldsymbol{b}_j$, ACA learns a nonlinear manifold defined by the summation of points along smooth curves $\boldsymbol{f}_j$, resulting in reduced reconstruction error. The key to our approach is the decomposition of each data point $\boldsymbol{x}_i$ into a sum of target components $\boldsymbol{g}_{ij}$, which allows the basis functions to be learned through simple, univariate regression.

Because the geometric nature of data is typically unknown, a variety of properties have been proposed for encouraging the extraction of meaningful low-dimensional representations. Component analysis methods are founded on the implicit assumption that useful representations are those that can accurately reconstruct input data. However, to enable effective generalization and interpretability, modeling assumptions or regularization often must be employed. Principal Component Analysis (PCA) fits a low dimensional subspace to data by finding directions of maximal

variance. While the underlying linearity assumption holds remarkably well for some data (e.g. aligned images of Lambertian objects such as faces [3]), even small perturbations (e.g. image translations) can introduce nonlinearities that bias the results. Kernel PCA handles nonlinear interactions by performing PCA in a higher-dimensional reproducing kernel Hilbert space, but is not optimized to effectively reconstruct the input data. Alternatively, manifold learning has acheived much success under the assumption that meaningful representations should preserve the local geometry of input data. However, these methods are often computationally expensive, difficult to interpret, and sensitive to noise.

To address these issues, we propose Additive Component Analysis (ACA), a novel method for nonlinear component analysis that explicitly optimizes reconstruction error. The motivating hypothesis underlying our approach is that reconstructed input data should vary slowly with respect to lower-dimensional representations, relaxing the strict linearity assumption of PCA. This is closely related to the slowness principle of Slow Feature Analysis (SFA) [38], in which invariant, high-level visual representations of sequential data are assumed to change slowly in time. However, our approach is more general, allowing for its application to unordered data and the automatic discovery of latent dimensions of variation other than time.

Our approach can be interpreted as an unsupervised additive model [10] constructed to predict training data from latent input variables, effectively fitting a smooth manifold to data with complexity controlled by an intuitive roughness penalty. An overview is shown in Fig. 1, along with comparisons to PCA. Our contributions can be summarized as follows: (1) ACA learns a memory-efficient explicit mapping from a low-dimensional latent space to the original input space by minimizing reconstruction error. This differs substantially from most other methods for nonlinear component analysis and manifold learning, resulting in improved robustness to noise while circumventing typical complications such as the pre-image problem [18] and out-of-sample inference [7]. (2) Efficient learning is accomplished with an alternating optimization algorithm that does not require the computation or storage of pairwise similarity matrices, thus enabling its efficient application to large datasets. (3) To enable increased representational power via nonparametric deep learning, we extend our optimization procedure to jointly learn multiple ACA layers. (4) Finally, we demonstrate the effectiveness of our method by showing improved performance on a variety of datasets.

1.1. Background and Related Work

Nonlinear dimensionality reduction has been an active area of research in recent years. In this section, we provide a brief overview of some previous methods along with some background on nonparametric statistics.

Nonlinear Component Analysis: Numerous attempts have been made to model nonlinearities within a component analysis framework. The most prominent example is kernel PCA [33], which applies an implicit nonlinear function to the input data and performs PCA in this feature space using the kernel trick. While out-of-sample inference is enabled via a representer theorem, there is no clear back-projection from the latent space to the original input space due to the pre-image problem [18]. Furthermore, the computational requirements of kernel PCA prevent its use on large datasets. Similarly, Gaussian Process Latent Variable Models [19] provide a general probabilistic interpretation of nonlinear PCA, but still suffer from many of the same issues due to the kernelized covariance function. Recently, approximate kernel methods have been proposed to improve computational efficiency. In [30], data are explicitly mapped to a randomized feature space in which inner products approximate kernel function evaluations. Using this idea, random nonlinear features have led to scalable algorithms for nonlinear PCA [23]. However, these approaches all first transform the input data, preventing their effective application to data reconstruction and denoising.

Manifold Learning: Methods for manifold learning find low-dimensional data representations by minimizing local geometric distortions, e.g. [34, 4]. Most often formalized as eigendecompositions, these algorithms do not learn explicit mappings to the latent space and thus cannot support back-projection or out-of-sample extensions directly [7]. Furthermore, these techniques tend to be topologically unstable, relying on unintuitive hyperparameters (e.g. neighborhood size) that require careful tuning in order to avoid degenerate behavior like short circuiting [1].

Autoencoders and Deep Neural Networks: Alternatively, autoencoders attempt to reconstruct data by learning explicit nonlinear mappings to and from latent representations. While shown to be equivalent to PCA in the linear case [2], nonlinear activation functions and stacking can enable a rich class of nonlinear representations [36]. In fact, some deep learning models can be interpreted as learning data manifolds [5] or lower-dimensional distributions [16]. While these methods employ explicit nonlinear mappings for reconstructing the original data, it is not yet clear how different regularization techniques and model architectures affect the space of learnable nonlinear functions [39], so they tend to require significant engineering effort and still often result in overfitting and poor interpretability.

Nonparametric Statistics: Unlike parametric methods that have a fixed complexity, nonparametric methods can adapt to the data, allowing for the representation of a wide range of nonlinearities. However, they are ineffective in high-dimensional settings due to the large amount of training data required to effectively characterize full data distributions [37]. To address this issue, additive models consider

a smaller class of nonparametric functions that decompose into sums of univariate functions considering each input dimension independently [10] via smoothing splines, piecewise polynomial functions with roughness penalties that encourage functions with small second derivatives [37]. Other nonparametric methods have also generalized the notion of principal components as geometric objects passing through the center of data [15, 29], but they cannot generally be used for dimensionality reduction. The method that is most similar to ACA is [11], which also learns explicit nonparametric functions to minimize a least-squares objective, but requires good initializations and is intractable for large datasets.

2. Additive Component Analysis

In this section, we formalize learning for ACA as an optimization problem and describe an approach for solving it. Given a dataset of vectors $\boldsymbol{x}_i \in \mathbb{R}^d$ for $i = 1, \ldots, n$, we aim to infer lower-dimensional latent representations $\boldsymbol{w}_i \in \mathbb{R}^m$ that can be used to optimally reconstruct the corresponding data. Recall that PCA accomplishes this by minimizing the error of approximating data as linear combinations of learned basis vectors $\boldsymbol{b}_j$ for $j = 1, \ldots, m$ with lower-dimensional latent representations given as the corresponding coefficients $\boldsymbol{w}_i$. We generalize this idea by approximating data as the sum of learned nonlinear basis functions $\boldsymbol{f}_j$ evaluated at some latent variables w_{ij}, resulting in approximations given by the additive model $\boldsymbol{f}(\boldsymbol{w}_i) = \sum_j \boldsymbol{f}_j(w_{ij})$.

With a least-squares reconstruction objective, our optimization problem can be formalized as follows in Eq. 1. Here, we aim to learn both the basis functions $\boldsymbol{f}_j$ and latent variables $\boldsymbol{w}_i$. We constrain the basis functions to belong to the set of cubic smoothing splines $\mathcal{F}$ with a roughness parameter ρ that balances approximation accuracy and smoothness. In addition, in order to compress the latent space and constrain the domains of the basis functions, we enforce that the latent representations belong to a closed set $\mathcal{W}$, implemented using a small amount of ℓ_2 regularization with a fixed hyperparameter of $\lambda = 0.01$.

$$\underset{\substack{\boldsymbol{f}_j \in \mathcal{F}, \\ \boldsymbol{w}_i \in \mathcal{W}}}{\arg\min} \sum_{i=1}^{n} \left\| \boldsymbol{x}_i - \sum_{j=1}^{m} \boldsymbol{f}_j(w_{ij}) \right\|_2^2 \tag{1}$$

This objective essentially minimizes the error in approximating data by projecting them onto an m-dimensional nonlinear manifold. Optimization for this problem presents an interesting challenge. A common approach for similar constrained component analysis problems (e.g. non-negative matrix factorization [8], dictionary learning [17], etc.) is alternating minimization. With one set of variables fixed, the resulting problem is usually much simpler. In our case, however, this is not so. With the latent representations $\boldsymbol{w}_i$ fixed, the optimization problem reduces to that of a supervised additive model, which must be solved using an iterative backfitting algorithm, often requiring many iterations to converge [10]. Instead, we derive an equivalent formulation of our problem that allows for alternating minimization with simple, closed-form updates.

2.1. Equivalent Problem Formulation

To enable simpler optimization, we introduce additional auxiliary variables by decomposing $\boldsymbol{x}_i$ into a sum of target components $\boldsymbol{g}_{ij}$, which we enforce with an affine equality constraint. Our optimization problem can then be equivalently written as follows:

$$\underset{\substack{\boldsymbol{f}_j \in \mathcal{F}, \\ \boldsymbol{w}_i \in \mathcal{W}, \boldsymbol{g}_{ij}}}{\arg\min} \sum_{i=1}^{n} \left\| \sum_{j=1}^{m} \left(\boldsymbol{g}_{ij} - \boldsymbol{f}_j(w_{ij})\right) \right\|_2^2 \text{s.t.} \sum_{j=1}^{m} \boldsymbol{g}_{ij} = \boldsymbol{x}_i \tag{2}$$

When expanded, the squared norm introduces additional cross terms in the form of $(\boldsymbol{g}_{ij} - \boldsymbol{f}_j(w_{ij}))^\intercal (\boldsymbol{g}_{ik} - \boldsymbol{f}_k(w_{ik}))$ for $k \neq j$. Interestingly, we can simply ignore these cross terms without affecting the solution, resulting in Eq. 3:

$$\underset{\substack{\boldsymbol{f}_j \in \mathcal{F}, \\ \boldsymbol{w}_i \in \mathcal{W}, \boldsymbol{g}_{ij}}}{\arg\min} \sum_{i=1}^{n} \sum_{j=1}^{m} \left\| \boldsymbol{g}_{ij} - \boldsymbol{f}_j(w_{ij}) \right\|_2^2 \text{s.t.} \sum_{j=1}^{m} \boldsymbol{g}_{ij} = \boldsymbol{x}_i \tag{3}$$

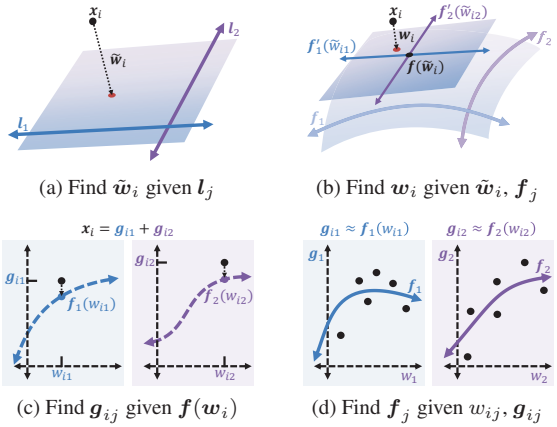
To see why this problem is equivalent to Eq. 1, consider solving only for the auxiliary target components $\boldsymbol{g}_{ij}$ with the basis functions $\boldsymbol{f}_j$ and latent representations $\boldsymbol{w}_i$ fixed. This decomposes into independent subproblems for each data instance $i = 1, \ldots, n$. We can then concatenate the variables so that $\mathbf{G}_i = [\boldsymbol{g}_{i1}, \ldots, \boldsymbol{g}_{im}]$, $\mathbf{F}_i = [\boldsymbol{f}_1(w_{i1}), \ldots, \boldsymbol{f}_m(w_{im})]$, and $\mathbf{A} = \mathbf{1}_m^\intercal \otimes \mathbf{I}_d$, where $\otimes$ denotes the Kronecker product. The target components in $\mathbf{G}_i$ can then be found by solving the optimization problem in Eq. 4. Note that this is simply the projection of the evaluated basis functions $\boldsymbol{f}_j(w_{ij})$ in $\mathbf{F}_i$ onto the affine subspace defined by the equality constraint. Thus, its solution is given in closed form, where $\mathbf{A}^+\boldsymbol{x}_i$ is a point on the affine subspace and the columns of $\mathbf{N}$ form an orthonormal basis for the nullspace of $\mathbf{A}$.

$$\begin{aligned} &\underset{\mathbf{G}_i}{\arg\min} \|\mathbf{G}_i - \mathbf{F}_i\|_F^2 \quad \text{s.t. } \mathbf{A}\text{vec}(\mathbf{G}_i) = \boldsymbol{x}_i \\ &\text{vec}(\mathbf{G}_i) = \mathbf{A}^+\boldsymbol{x}_i + \mathbf{N}\mathbf{N}^\intercal \left(\text{vec}(\mathbf{F}_i) - \mathbf{A}^+\boldsymbol{x}_i\right) \end{aligned} \tag{4}$$

Here, $\mathbf{A}^+\boldsymbol{x}_i = \frac{1}{m}\boldsymbol{x}_i \otimes \mathbf{1}$ and $\mathbf{N}\mathbf{N}^\intercal = \left(\mathbf{I} - \frac{1}{m}\mathbf{1}\mathbf{1}^\intercal\right) \otimes \mathbf{I}$. After some simplification, the target components $\boldsymbol{g}_{ij}$ are given by Eq. 5. Intuitively, this can be interpreted as distributing the current approximation error equally among the target components $\boldsymbol{g}_{ij}$ so that they sum to $\boldsymbol{x}_i$.

$$\boldsymbol{g}_{ij} = \boldsymbol{f}_j(w_{ij}) + \frac{1}{m}\left(\boldsymbol{x}_i - \sum_{j=1}^{m} \boldsymbol{f}_j(w_{ij})\right) \tag{5}$$

Plugging this back into our problem in Eq. 3 gives our original objective function in Eq. 1 rescaled by m^{-1}. Thus, both problems have exactly the same solutions.



(a) Find $\tilde{\boldsymbol{w}}_i$ given $\boldsymbol{l}_j$ (b) Find $\boldsymbol{w}_i$ given $\tilde{\boldsymbol{w}}_i$, $\boldsymbol{f}_j$

(c) Find $\boldsymbol{g}_{ij}$ given $\boldsymbol{f}(\boldsymbol{w}_i)$ (d) Find $\boldsymbol{f}_j$ given w_{ij}, $\boldsymbol{g}_{ij}$

Figure 2: A visualization of one iteration of our alternating optimization procedure. (a) First, approximate latent variables $\tilde{\boldsymbol{w}}_i$ are found by projecting each data point $\boldsymbol{x}_i$ onto the affine subspace defined by linear basis function approximations $\boldsymbol{l}_j$, initialized using PCA. (b) Then, $\boldsymbol{w}_i$ is updated by projecting $\boldsymbol{x}_i$ onto the tangent space at the point $\boldsymbol{f}(\boldsymbol{w}_i)$. This step is repeated multiple times with smaller step sizes for increased accuracy, resulting in an approximate orthogonal projection of $\boldsymbol{x}_i$ onto the manifold. (c) The target components $\boldsymbol{g}_{ij}$ are then found by equally redistributing the reconstruction error between them. (d) Finally, the basis functions $\boldsymbol{f}_j$ (along with their linear approximations $\boldsymbol{l}_j$) are found using simple univariate regression.

2.2. Alternating Optimization

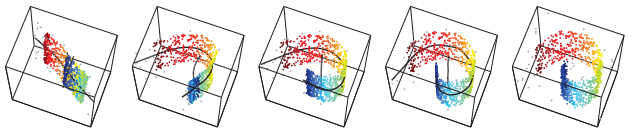
Unlike our original problem formulation in Eq. 1, the one in Eq. 3 naturally lends itself to an efficient alternating minimization algorithm. After initializing with PCA, we fix the basis functions $\boldsymbol{f}_j$ and jointly solve for the latent representations $\boldsymbol{w}_i$ and target components $\boldsymbol{g}_{ij}$. Then, with these variables fixed, we solve for the basis functions $\boldsymbol{f}_j$, repeating this process until convergence. Despite the nonconvexity of our problem, this alternating optimization procedure has been shown to converge consistently to good solutions, as demonstrated empirically in the experiments discussed in Sec. 4. An overview is shown in Fig. 2 and an example of its progression on a synthetic dataset is shown in Fig. 3.

1.) Latent Variables: With $\boldsymbol{f}_j$ fixed, we solve Eq. 6 for each data instance $i = 1, \ldots, n$:

$$\underset{\boldsymbol{w}_i \in \mathcal{W}, \boldsymbol{g}_{ij}}{\arg\min} \sum_{j=1}^{m} \left\| \boldsymbol{g}_{ij} - \boldsymbol{f}_j(w_{ij}) \right\|_2^2 \text{ s.t.} \sum_{j=1}^{m} \boldsymbol{g}_{ij} = \boldsymbol{x}_i \quad (6)$$

This is simply the projection of $\boldsymbol{x}_i$ onto the learned manifold. Solving this directly is difficult due to the nonlinear basis functions $\boldsymbol{f}_j$. However, since we enforce that they be smooth with small second derivatives, they can be effectively approximated by first-order Taylor expansions centered around some approximate solutions $\tilde{w}_{ij}$, which define the tangent space of the manifold at the point $\boldsymbol{f}(\boldsymbol{w}_i)$:

$$\boldsymbol{f}_j(w_{ij}) \approx \boldsymbol{u}_{ij} + w_{ij} \boldsymbol{f}'_j(\tilde{w}_{ij}), \boldsymbol{u}_{ij} = \boldsymbol{f}_j(\tilde{w}_{ij}) - \tilde{w}_{ij} \boldsymbol{f}'_j(\tilde{w}_{ij}) \quad (7)$$



Initialization Iteration 20 Iteration 40 Iteration 56 **Original Data**

Figure 3: An example of our optimization procedure applied to a noisy synthetic dataset. The original data points are shown on the right. On the left are their denoised projections using the learned basis functions (shown in black) throughout optimization. Starting from a linear subspace at initialization, the basis functions adapt to the nonlinear structure of the data, resulting in a near perfect reconstruction of the true underlying manifold.

The initial $\tilde{w}_{ij}$ can be found by projecting $\boldsymbol{x}_i$ onto the affine subspace defined by linear approximations $\boldsymbol{l}_j$ to the spline functions $\boldsymbol{f}_j$. After plugging in this approximation, the resulting problem, shown below in Eq. 8, is a strictly-convex quadratic program. Furthermore, its unique minimizers are given by simple, closed-form expressions.

$$\underset{\boldsymbol{w}_i \in \mathcal{W}, \boldsymbol{g}_{ij}}{\arg\min} \sum_{j=1}^{m} \left\| \boldsymbol{g}_{ij} - \boldsymbol{u}_{ij} - w_{ij} \boldsymbol{f}'_j(\tilde{w}_{ij}) \right\|_2^2 \text{ s.t.} \sum_{j=1}^{m} \boldsymbol{g}_{ij} = \boldsymbol{x}_i \quad (8)$$

Due to the equivalence between Eqs. 1 and 3, we can substitute in the solution for $\boldsymbol{g}_{ij}$ into Eq. 8 and solve for $\boldsymbol{w}_i$ directly as follows, where $\mathbf{D}_i = [\boldsymbol{f}'_1(\tilde{w}_{ij}), \ldots, \boldsymbol{f}'_m(\tilde{w}_{ij})]$ and $\boldsymbol{u}_i = \sum_j \boldsymbol{u}_{ij}$:

$$\underset{\boldsymbol{w}_i \in \mathcal{W}}{\arg\min} \left\| \boldsymbol{x}_i - \boldsymbol{u}_i - \mathbf{D}_i \boldsymbol{w}_i \right\|_2^2 \quad (9)$$

This approximation can be improved by repeatedly updating $\boldsymbol{w}_i$ with decreasing step sizes. Afterward, the target components $\boldsymbol{g}_{ij}$ are found using the closed form solution in Eq. 5. Note that inference can later be performed on test data using this same procedure.

2.) Basis Functions: With $\boldsymbol{w}_i$ and $\boldsymbol{g}_{ij}$ fixed, we then solve Eq. 10 for each $j = 1, \ldots, m$:

$$\underset{\boldsymbol{f}_j \in \mathcal{F}}{\arg\min} \sum_{i=1}^{n} \left\| \boldsymbol{g}_{ij} - \boldsymbol{f}_j(w_{ij}) \right\|_2^2 \quad (10)$$

This is standard univariate regression in which the latent variables w_{ij} are mapped to the target components $\boldsymbol{g}_{ij}$. We restrict the basis functions $\boldsymbol{f}_j$ to be roughness-penalized smoothing splines [13] due to their generality and efficient computation. Thus, the set $\mathcal{F}$ can be defined as:

$$\mathcal{F} = \left\{ \boldsymbol{f} : \mathbb{R} \to \mathbb{R}^d \,\middle|\, \forall q \in [1, d]: \int (f''_q(x))^2 dx \leq \gamma \right\} \quad (11)$$

These constraints are implemented with a roughness penalty balancing approximation accuracy and complexity with a roughness hyperparameter ρ. The solution to the problem in Eq. 10 is a cubic spline with knots at each of the training

points, which can be expressed as a linear combination of spline basis functions b_{tj} for $t = 1, \ldots, n_b$ [13]. Specifically, we model the target components $\boldsymbol{g}_{ij}$ as $\boldsymbol{f}_j(w_{ij}) = \sum_t \boldsymbol{c}_{tj} b_{tj}(w_{ij})$ with coefficient vectors $\boldsymbol{c}_{tj} \in \mathbb{R}^d$. In our implementation, we use B-spline basis functions because they have bounded support resulting in sparse, banded matrices and linear-time inverse computations [14]. Furthermore, their evaluation and derivatives can be efficiently computed using a simple recursive formula [37]. Eq. 10 can then be reformulated as a simple ℓ_2-regularized least-squares problem with the closed form solution given in Eq. 12 below for $i = 1, \ldots, n$ and $s, t = 1, \ldots, n_b$, where $\mathbf{C}_j = [\boldsymbol{c}_{1j}, \ldots, \boldsymbol{c}_{n_b j}]$.

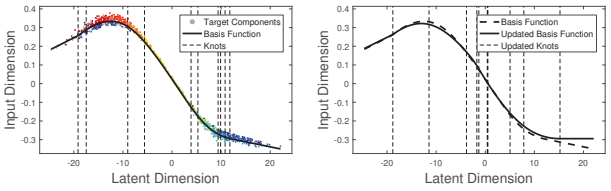
$$\mathbf{B}_j(i,t) = b_{tj}(w_{ij}), \quad \boldsymbol{\Omega}_j(s,t) = \int b''_{sj}(x) b''_{tj}(x) dx$$
$$\mathbf{C}_j = \left(\rho \mathbf{B}_j^\intercal \mathbf{B}_j + (1-\rho)\boldsymbol{\Omega}_j\right)^{-1} \mathbf{B}_j^\intercal \left[\boldsymbol{g}_{1j} \cdots \boldsymbol{g}_{nj}\right]^\intercal \quad (12)$$

With knots at each training instance, the number of basis functions would grow linearly with the number of training examples, which would become computationally intractable for large datasets. However, with a small enough ρ, the solution can be well-approximated with knots placed at a randomly selected subset of $n_b = 20$ training instances so that $n_b \ll n$. This allows the basis functions to be defined using a relatively small number of parameters.

2.3. Approximate Stochastic Optimization

While the optimization procedure described in the previous section is memory efficient due to the separability of the cost function into smaller subproblems, solving for all latent variables at each iteration can be computationally expensive (and possibly redundant) for extremely large datasets. Ideally, we would instead prefer to take a stochastic approach that considers only a random subset of the data at each iteration. The basis functions could then be updated with a certain step size by taking a weighted average with the parameters from the previous iteration. However, since the knot locations of the spline functions change at each iteration, their corresponding parameters are not comparable.

To overcome this issue, we propose an approach that approximates the spline functions from the previous iteration using the knots from the current iteration so that their parameters can be averaged. Specifically, we use Schoenberg's variation diminishing spline approximation [25, 24], a simple and efficient method for function approximation that does not require solving a linear system of equations as with the roughness-penalized spline approximation described in Sec. 2.2. To understand this method, first recall that spline functions can be interpreted geometrically as smoothed versions of their control polygons, which are piecewise-linear functions with vertices located at specific control points. For a cubic spline $f(w) = \sum_t c_t b_t(w)$ with a knot vector $\boldsymbol{\tau}$,



(a) Original Basis Function (b) Updated Approximation

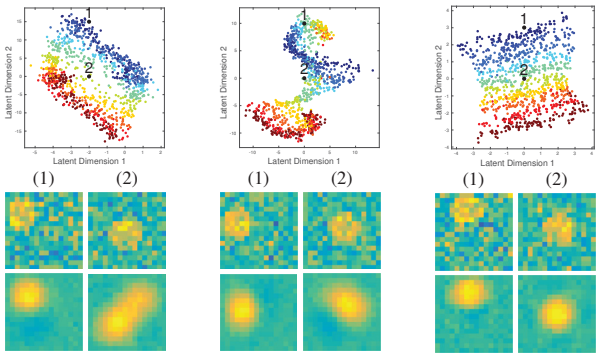
Figure 4: An example of the variation diminishing spline approximation used in our stochastic optimization technique for parameter averaging of basis functions with different knot locations. The original basis function (a) was fitted to target components with knot locations denoted by dotted vertical lines while the updated basis function (b) was approximated with different knot locations without requiring expensive least-squares fitting.

these control points have coordinates (τ_t^*, c_t) where $\tau_t^* = \frac{1}{3}(\tau_{t+1} + \tau_{t+2} + \tau_{t+3})$ are the knot averages of $\boldsymbol{\tau}$. Similarly, for any function f, it's variation diminishing cubic spline approximation is given by $(Vf)(w) = \sum_t f(\tau_t^*) b_t(w)$ where the coefficients are given directly as function evaluations at the knot averages. Thus, before updating the basis function parameters from the previous iteration, we take a variation diminishing spline approximation of their control polygons evaluated at the new knot averages from the current iteration, essentially resulting in a linear interpolation of the control points. While this is only a rough approximation as shown in Fig. 4, it leads to effective learning with significantly reduced training time, which we demonstrate experimentally in Sec. 4.

3. Composition of Additive Models

Despite their generality, additive models can only represent a relatively small set of possible multivariate functions. Thus, the space of manifolds that can be learned with ACA is also limited. Consider, for example, a dataset of noisy images containing translated circles like those in Fig. 5b. Its intrinsic dimensionality equals two because there are only two independent dimensions of variation: horizontal and vertical location. However, the underlying nonlinearities cannot be effectively modeled with ACA, resulting in poor latent separability and reconstruction performance. This is demonstrated in the top of Fig. 5.

This fundamental limitation of component analysis is a result of the restricted additive interactions allowed between latent variables. To address this, we propose a deep extension of our approach that stacks multiple ACA layers together, increasing representational power by composing ℓ additive models $\boldsymbol{f}^k$ constructed as the sum of basis functions $\boldsymbol{f}_j^k$ for $j = 1, \ldots, m_k$ where $m_{k-1} < m_k < d$ so that $\boldsymbol{f} = \boldsymbol{f}^\ell \circ \boldsymbol{f}^{\ell-1} \circ \cdots \circ \boldsymbol{f}^1$. Similar approaches have seen much success within the area of deep learning, partially due to the observation that increasing depth can allow for comparable expressivity with exponentially fewer parameters [6]. In-



(a) One Layer ($m = 2$) (b) Two Layers (Greedy) ($\boldsymbol{m} = [2, 4]$) (c) Two Layers (Joint) ($\boldsymbol{m} = [2, 4]$)

Figure 5: A demonstration of the increased representational power provided by the composition of additive models, comparing ACA with one layer (a), with two layers trained greedily layer by layer (b), and with two layers trained jointly (c). The learned low-dimensional latent space (top) indicates two numbered example points. The corresponding original images (middle) are compared alongside the denoised reconstructions (bottom). Deep ACA results in better performance and more interpretable representations by jointly learning richer interactions between latent variables.

deed, in Fig. 5c, we show that deep ACA successfully models translation, resulting in reduced reconstruction error and an interpretable two-dimensional representation in which latent variables correspond to different spatial dimensions.

While function composition makes optimization more difficult, we use the an approach similar to the Method of Auxiliary Coordinates (MAC) [12] to learn the parameters of all layers jointly using essentially the same procedure described in Sec. 2.2. This leads to a more interpretable latent space in comparison to a greedy approach that learns the parameters of each layer independently, as shown in Fig. 5.

We now aim to infer a set of latent variables $\boldsymbol{w}_i^k \in \mathbb{R}^{m_k}$ for $k = 1, \ldots, \ell$, where $\boldsymbol{w}_i^1$ is our low-dimensional representation and the others are constrained to be intermediate layer outputs, i.e. $\boldsymbol{w}_i^{k+1} = \boldsymbol{f}^k(\boldsymbol{w}_i^k)$ for $k = 1, \ldots, \ell - 1$. For simpler notation, we fix $\boldsymbol{w}_i^{\ell+1} = \boldsymbol{x}_i$ and denote $\boldsymbol{f}^{k\uparrow} = \boldsymbol{f}^\ell \circ \boldsymbol{f}^{\ell-1} \circ \cdots \circ \boldsymbol{f}^k$, giving the optimization problem in Eq. 13. Our optimization procedure can then proceed as described in Sec. 2.2.

$$\underset{\substack{\boldsymbol{f}_j^k \in \mathcal{F}, \\ \boldsymbol{w}_i \in \mathcal{W}}}{\arg\min} \sum_{i=1}^{n} \sum_{k=1}^{\ell} \left\| \boldsymbol{x}_i - \boldsymbol{f}^{k\uparrow}(\boldsymbol{w}_i^k) \right\|_2^2 \text{ s.t. } \boldsymbol{w}_i^{k+1} = \boldsymbol{f}^k(\boldsymbol{w}_i^k) \quad (13)$$

To enable effective learning of the intermediate layers, we ignore the equality constraint when solving for the latent variables so that each $\boldsymbol{f}^{k\uparrow}(\boldsymbol{w}_i^k)$ optimally reconstructs $\boldsymbol{x}_i$. (Note that this bears some similarity to deeply-supervised deep neural networks, in which intermediate loss functions encourage the discriminability of hidden layers [21].) In other words, deep ACA can be interpreted as learning a sequence of manifolds with decreasing dimensionality so that $\boldsymbol{w}_i^k$ can be found by orthogonally projecting $\boldsymbol{x}_i$ onto the m_k-dimensional manifold defined by $\boldsymbol{f}^{k\uparrow}$.

As before, we first approximate the latent variables by fixing the basis functions and iteratively projecting $\boldsymbol{x}_i$ onto the resulting manfiold's tangent space, which is constructed as the first-order Taylor expansion of $\boldsymbol{f}^{k\uparrow}(\boldsymbol{w}_i^k)$ around $\tilde{\boldsymbol{w}}_i^k$:

$$\mathbf{D}_i^k = [\boldsymbol{f}_1^{k\prime}(\tilde{w}_{ij}^k), \ldots, \boldsymbol{f}_{m_k}^{k\prime}(\tilde{w}_{ij}^k)], \; \mathbf{D}_i^{k\uparrow} = \mathbf{D}_i^\ell \mathbf{D}_i^{\ell-1} \cdots \mathbf{D}_i^k$$
$$\boldsymbol{f}^{k\uparrow}(\boldsymbol{w}_i^k) \approx \mathbf{D}_i^{k\uparrow} \boldsymbol{w}_i^k + \boldsymbol{u}_i^k, \; \boldsymbol{u}_i^k = \boldsymbol{f}^{k\uparrow}(\tilde{\boldsymbol{w}}_i^k) - \mathbf{D}_i^{k\uparrow} \tilde{\boldsymbol{w}}_i^k \quad (14)$$

Analogous to Eq. 9, the result can be solved in closed-form.

We again decompose each set of latent variables into target components so that $\boldsymbol{w}_{ij}^{k+1} = \sum_{j=1}^{m_k} \boldsymbol{g}_{ij}^k$. After reintroducing the equality constraint and following a derivation similar to that in Sec. 2.1, they can then be given as:

$$\boldsymbol{g}_{ij}^k = \boldsymbol{f}_j(w_{ij}^k) + \frac{1}{m_k} \left(\boldsymbol{w}_i^{k+1} - \sum_{j=1}^{m_k} \boldsymbol{f}_j(w_{ij}^k) \right) \quad (15)$$

Finally, we fit the basis functions $\boldsymbol{f}^k(w_{ij}^k)$ to the target components $\boldsymbol{g}_{ij}^{k+1}$ using standard regression as in Sec. 2.2.

4. Experimental Results

In this section, we evaluate the effectiveness of our method through qualitative and quantitative analyses on a variety of synthetic and real datasets. This is intended to demonstrate the wide applicability of ACA and to encourage its use as a simple alternative to PCA. Specifically, we demonstrate robustness to noise, improved denoising and reconstruction performance, and more interpretable representations with better separation of semantic categories, including large-scale experiments on the MNIST dataset.

Because ACA explicitly optimizes reconstruction accuracy, it is naturally very robust to noise unlike most approaches to manifold learning that require estimation of a neighborhood graph using pairwise distances. To demonstrate this, we constructed a synthetic dataset consisting of 1000 points sampled along a curved two-dimensional manifold embedded in three-dimensions. We then added various amounts of Gaussian noise along with 50 uniformly random outliers. A visualization of this data can be seen in Fig. 3 and qualitative comparisons are shown in Fig. 6 with a variety of nonlinear dimensionality reduction techniques. ACA consistently results in superior low-dimensional representations even in the presence of extreme noise. Importantly, unlike the other compared nonlinear methods, ACA trivially supports reconstruction of the underlying manifold for visualization of denoised data.

In image data, "noise" can take a variety of forms, including sensor noise, cast shadows, misalignment, occlusions, etc. Due to its ability to model complex nonlinear structure, ACA results in perceptually more accurate image

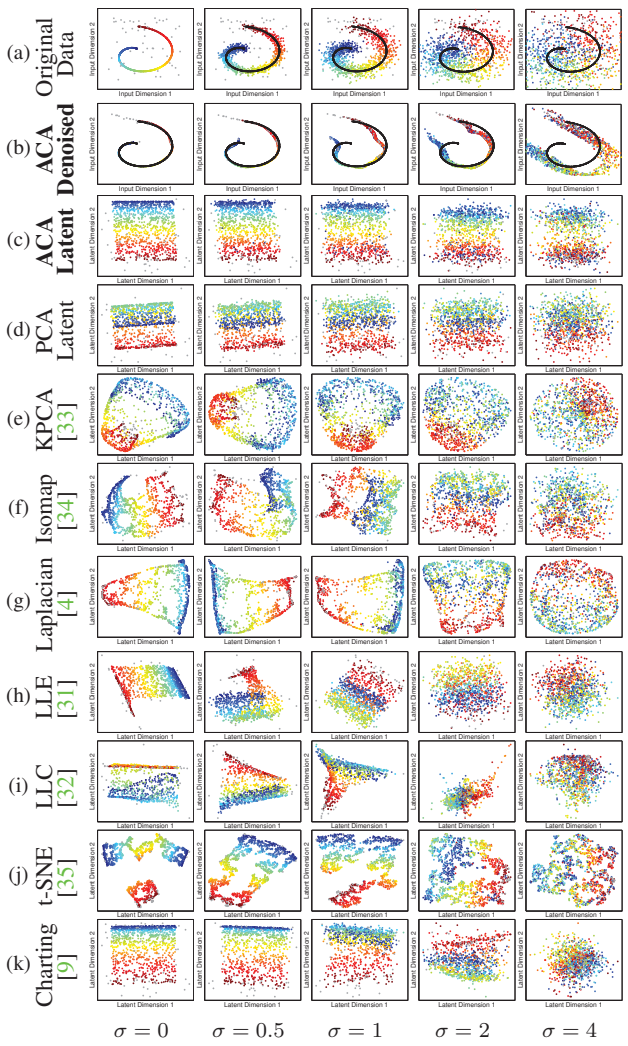
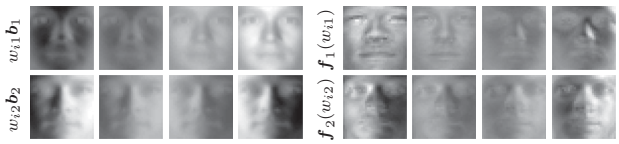


Figure 6: Qualitative comparisons showing our method's superior robustness to noise. In each column, different amounts of Gaussian noise are added to points along the manifold from Fig. 3 (shown in color) along with uniformly random outliers (shown in grey). Top views of the original noisy data (a) are compared to the denoised reconstructions achieved by ACA (b). The corresponding low-dimensional latent spaces (c) are also compared to those of PCA (d) and a variety of other nonlinear dimensionality reduction techniques (e-k). Even with large amounts of noise, ACA recovers the underlying structure of the data very well resulting in consistent low-dimensional representations.

reconstructions that are invariant to many of these sources. This is demonstrated in Fig. 7 on the Extended Yale Face Database B [22], which contains partially aligned images of faces under different lighting conditions. Dimensionality reduction was performed on ZCA whitened images using ACA and PCA with 4 and 20 components respectively, giving similar average mean-squared reconstruction error. Example components are visualized in Fig. 8. Also compared was Kernel PCA with 4 components and a Gaussian kernel

Figure 7: A demonstration of the invariance and complex denoising capabilities of ACA. Given images of faces under a variety of different lighting conditions (a), dimensionality reduction was performed using ACA (b) and KPCA (c) with 4 components and PCA (d) with 20 components. Because it is able to learn rich nonlinearities, ACA achieves more perceptually plausible denoised images that retain more detail with fewer components.



(a) PCA Components (b) ACA Components

Figure 8: A visualization of $m = 2$ example components learned by (a) PCA and (b) ACA evaluated at different values of the latent variables w_{ij} (increasing from left to right) from the experiment in Fig. 7. The PCA basis vectors $\boldsymbol{b}_j$ can only be rescaled by a scalar multiple while the nonlinear basis functions $\boldsymbol{f}_j$ of ACA allow for richer variations that better model individuals' appearance.

with parameter $\sigma^2 = 2$. Since KPCA does not directly enable back-projection, approximate pre-images were found using fixed-point iterations [26]. The resulting de-whitened image reconstructions for ACA are perceptually more plausible, resulting in better shadow removal while preserving more details for improved identity preservation.

In addition to enabling accurate data reconstruction, ACA can also encode complex invariances due to the underlying smoothness constraints. This results in low-dimensional representations that are useful for high-level tasks such as exploratory data analysis and clustering. This is demonstrated in Fig. 9, which shows how the parameter ρ affects class separability and data reconstruction performance on unseen testing data. In this experiment, approximately half of the 1440 processed images from the COIL-20 dataset [28] were used as training data for a two layer deep ACA model with $\boldsymbol{m} = [2, 4]$ and a varying roughness parameter. The learned models were then applied to the remaining testing images and the corresponding two-dimensional representations plotted alongside example image reconstructions. Increasing ρ allows for rougher basis functions with higher complexity, giving better separability of categories (shown as different colors) in the latent space

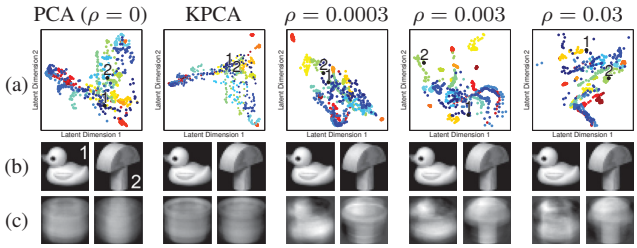
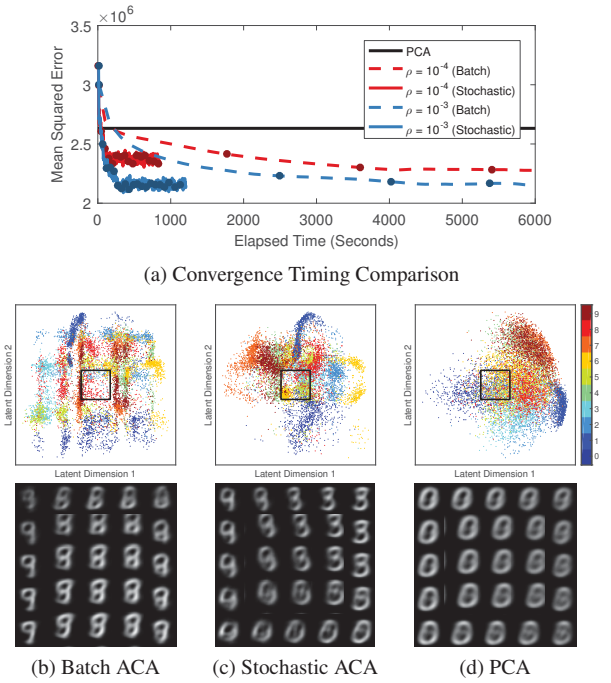


Figure 9: A visualization of how the roughness parameter ρ affects the performance of ACA on novel testing data. Two-dimensional latent embeddings (a) of images from the COIL-20 dataset are shown in columns for PCA, KPCA, and ACA with different values of ρ. Two example images (b) are also shown, along with their reconstructions (c). Increasing ρ can improve image reconstruction performance and the separability of object classes (shown as different colors in the latent space), but can also reduce performance due to overfitting.

in comparison to PCA and KPCA with a Gaussian kernel ($\sigma^2 = 7$). However, it can also lead to overfitting and poor reconstruction accuracy of test images.

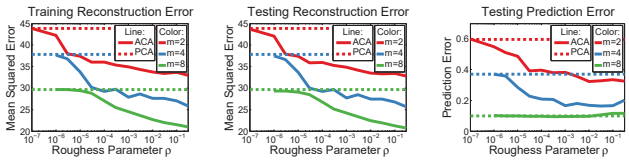
Finally, we demonstrate large-scale results on the MNIST dataset [20] containing 60k training images and 10k testing images, which is prohibitive for many nonlinear dimensionality reduction implementations. In Fig. 10, training error is shown against elapsed time using both batch optimization and the approximate stochastic technique from Sec. 2.3 with a batch size of 1000. Stochastic optimization leads to much faster convergence in less than 10 minutes with an unoptimized Matlab implementation. Also shown are the resulting two-dimensional latent representations and example reconstructions of the testing images. While batch optimization leads to slightly lower training error, the over-separated latent space indicates overfitting in comparison to stochastic optimization. Note that while some techniques designed specifically for low-dimensional visualization (e.g. t-SNE [35]) may result in better class separation, they cannot reconstruct the input or be applied to new data, leading to limited applicability. In Figure 11, quantitative results are also shown demonstrating reconstruction performance and nearest-neighbor classification performance.



(a) Convergence Timing Comparison

(b) Batch ACA (c) Stochastic ACA (d) PCA

Figure 10: The effect of our approximate stochastic optimization scheme applied to the MNIST dataset. Reconstruction error (a) is plotted throughout training for both stochastic and batch optimization with solid dots shown every 20 iterations. The resulting two-dimensional test data embeddings for $\rho = 10^{-3}$ (b,c) are compared against to those of PCA (d) alongside grids of reconstructed images from the regions indicated by black squares.



(a) (b) (c)

Figure 11: Results on the MNIST datset, showing reconstruction error of training images (a), reconstruction error of testing images (b), and testing nearest-neighbor classification error (c). Performance is compared between PCA and ACA for a variety of roughness parameters ρ and numbers of components m.

5. Conclusion

Additive Component Analysis combines the simplicity and broad applicability of linear component analysis with the nonlinear representational power of manifold learning. It produces robust and interpretable latent representations given by memory-efficient models optimized for data reconstruction. This results in significantly improved performance, especially in the presence of noise, enabling the detailed analysis and visualization of large, real-world datasets. Furthermore, the composition of multiple ACA layers can overcome the modeling limitations of additive components, with parameters that can be learned jointly with the same memory-efficient optimization procedure. We believe that this demonstrates the encouraging potential for nonparametric deep learning using compositions of additive models as an alternative to standard linear transformations with fixed nonlinear activation functions. This could potentially lead to adaptive representational power with far fewer parameters, reduced overfitting due to the underlying smoothness assumption, and superior robustness.

Acknowledgmenets: This research was supported in part by the National Science Foundation under grants RI-1617953 and IIS-1418523. The content is solely the responsibility of the authors and does not necessarily represent the official views of the National Science Foundation.

References

[1] M. Balasubramanian and E. L. Schwartz. The Isomap algorithm and topological stability. *Science*, 295(5552):7–7, 2002. 2

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

[3] R. Basri and D. W. Jacobs. Lambertian reflectance and linear subspaces. *Pattern Analysis and Machine Intelligence (PAMI)*, 25(2):218–233, 2003. 2

[4] M. Belkin and P. Niyogi. Laplacian eigenmaps for dimensionality reduction and data representation. *Neural computation*, 15(6):1373–1396, 2003. 2, 7

[5] Y. Bengio, A. Courville, and P. Vincent. Representation learning: A review and new perspectives. *Pattern Analysis and Machine Intelligence (PAMI)*, 35(8):1798–1828, 2013. 2

[6] Y. Bengio and O. Delalleau. On the expressive power of deep architectures. In *Algorithmic Learning Theory*. Springer, 2011. 5

[7] Y. Bengio, J.-F. Paiement, P. Vincent, O. Delalleau, N. Le Roux, and M. Ouimet. Out-of-sample extensions for LLE, Isomap, MDS, Eigenmaps, and spectral clustering. In *Advances in Neural Information Processing Systems (NIPS)*, 2004. 2

[8] M. W. Berry, M. Browne, A. N. Langville, V. P. Pauca, and R. J. Plemmons. Algorithms and applications for approximate nonnegative matrix factorization. *Computational Statistics & Data Analysis*, 52(1):155–173, 2007. 3

[9] M. Brand. Charting a manifold. In *Advances in Neural Information Processing Systems (NIPS)*, 2002. 7

[10] A. Buja, T. Hastie, and R. Tibshirani. Linear smoothers and additive models. *The Annals of Statistics*, pages 453–510, 1989. 2, 3

[11] M. A. Carreira-Perpiñán and Z. Lu. Dimensionality reduction by unsupervised regression. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2008. 3

[12] M. A. Carreira-Perpiñán and W. Wang. Distributed optimization of deeply nested systems. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2014. 6

[13] P. Craven and G. Wahba. Smoothing noisy data with spline functions. *Numerische Mathematik*, 31(4):377–403, 1978. 4, 5

[14] C. De Boor. *A practical guide to splines*, volume 27. Springer-Verlag New York, 1978. 5

[15] T. Hastie and W. Stuetzle. Principal curves. *Journal of the American Statistical Association*, 84(406):502–516, 1989. 3

[16] D. P. Kingma and M. Welling. Auto-encoding variational bayes. *International Conference on Learning Representations (ICLR)*, 2014. 2

[17] K. Kreutz-Delgado, J. F. Murray, B. D. Rao, K. Engan, T. S. Lee, and T. J. Sejnowski. Dictionary learning algorithms for sparse representation. *Neural Computation*, 15(2):349–396, 2003. 3

[18] J. T. Kwok and I. W. Tsang. The pre-image problem in kernel methods. In *International Conference on Machine Learning (ICML)*, pages 408–415, 2003. 2

[19] N. Lawrence. Probabilistic non-linear principal component analysis with gaussian process latent variable models. *Journal of Machine Learning Research*, 6:1783–1816, 2005. 2

[20] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. 8

[21] C.-Y. Lee, S. Xie, P. Gallagher, Z. Zhang, and Z. Tu. Deeply-supervised nets. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2015. 6

[22] K.-C. Lee, J. Ho, and D. J. Kriegman. Acquiring linear subspaces for face recognition under variable lighting. *Pattern Analysis and Machine Intelligence (PAMI)*, 27(5):684–698, 2005. 7

[23] D. Lopez-Paz, S. Sra, A. Smola, Z. Ghahramani, and B. Schölkopf. Randomized nonlinear component analysis. In *International Conference on Machine Learning (ICML)*, 2014. 2

[24] T. Lyche and K. Mørken. *Spline methods*. Department of Mathematics, University of Oslo, 2008. 5

[25] M. Marsden and I. J. Schoenberg. *On Variation Diminishing Spline Approximation Methods*, pages 247–268. Birkhäuser Boston, Boston, MA, 1988. 5

[26] S. Mika, B. Schölkopf, A. J. Smola, K.-R. Müller, M. Scholz, and G. Rätsch. Kernel pca and de-noising in feature spaces. In *Advances in Neural Information Processing Systems (NIPS)*, 1999. 7

[27] H. Narayanan and S. Mitter. Sample complexity of testing the manifold hypothesis. In *Advances in Neural Information Processing Systems (NIPS)*, 2010. 1

[28] S. A. Nene, S. K. Nayar, H. Murase, et al. Columbia object image library (COIL-20). Technical report, Technical Report CUCS-005-96, 1996. 7

[29] U. Ozertem and D. Erdogmus. Locally defined principal curves and surfaces. *Journal of Machine Learning Research (JMLR)*, 12:1249–1286, 2011. 3

[30] A. Rahimi and B. Recht. Random features for large-scale kernel machines. In *Advances in Neural Information Processing Systems (NIPS)*, 2007. 2

[31] S. T. Roweis and L. K. Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290(5500):2323–2326, 2000. 7

[32] S. T. Roweis, L. K. Saul, and G. E. Hinton. Global coordination of local linear models. *Advances in Neural Information Processing Systems (NIPS)*, 2002. 7

[33] B. Schölkopf, A. Smola, and K.-R. Müller. Nonlinear component analysis as a kernel eigenvalue problem. *Neural computation*, 10(5):1299–1319, 1998. 2, 7

[34] J. B. Tenenbaum, V. De Silva, and J. C. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290(5500):2319–2323, 2000. 2, 7

[35] L. Van der Maaten and G. Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(2579-2605):85, 2008. 7, 8

[36] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, and P.-A. Manzagol. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *Journal of Machine Learning Research (JMLR)*, 11:3371–3408, 2010. 2

[37] L. Wasserman. *All of nonparametric statistics*. Springer Science & Business Media, 2006. 2, 3, 5

[38] L. Wiskott and T. J. Sejnowski. Slow feature analysis: Unsupervised learning of invariances. *Neural computation*, 14(4):715–770, 2002. 2

[39] C. Zhang, S. Bengio, M. Hardt, B. Recht, and O. Vinyals. Understanding deep learning requires rethinking generalization. 2017. 2