Online Dictionary Learning for Approximate Archetypal Analysis

Jieru Mei, Chunyu Wang, and Wenjun Zeng

Microsoft Research Asia, Beijing, China
meijieru@gmail.com, {chnuwa, wezeng}@microsoft.com

Abstract. Archetypal analysis is an unsupervised learning approach which represents data by convex combinations of a set of archetypes. The archetypes generally correspond to the extremal points in the dataset and are learned by requiring them to be convex combinations of the training data. In spite of its nice property of interpretability, the method is slow. We propose a variant of archetypal analysis which scales gracefully to large datasets. The core idea is to decouple the binding between data and archetypes and require them to be unit normalized. Geometrically, the method learns a convex hull inside the unit sphere and represents the data by their projections on the closest surfaces of the convex hull. By minimizing the representation error, the method pushes the convex hull surfaces close to the regions of the sphere where the data reside. The vertices of the convex hull are the learned archetypes. We apply the method to human faces and poses to validate its effectiveness in the context of reconstructions and classifications.

Keywords: Archetypal analysis · Convex hull · Sparsity

1 Introduction

Unsupervised basis learning is a class of methods [8] which aims to discover the underlying low-dimensional structures of the data. They may serve different purposes depending on the specific tasks. For example, methods such as [36, 23, 32] are usually used for dimension reductions. Methods such as sparse coding [21, 17], non-negative matrix factorization [14] and the clustering methods [37, 5] are usually used as data modeling tools.

Archetypal Analysis (AA) [6, 4] is an unsupervised learning method which represents data by convex combinations of the archetypes. One distinctive property of AA is that the archetypes are convex combinations of the training data which binds the data and archetypes. There is a geometric interpretation for the method. See Fig. 1. First, convex combinations of the archetypes form a principal convex hull which we call *P-hull*. To minimize the representation error on the dataset, the P-hull should be large enough to cover as many training data as possible. Second, since each archetype is required to be chosen from the convex hull (*D-hull*) formed by convex combinations of training data, the P-hull should be within the D-hull. So archetypal analysis seeks for the largest P-hull

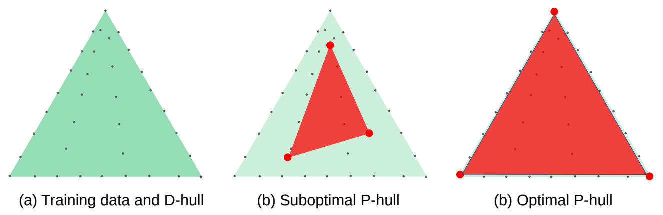

Fig. 1: Illustrations of D-hulls and P-hulls. (a) The convex combinations of training data form a convex hull called D-hull. (b) The red dots denote the archetypes. Each archetype is chosen from convex combinations of training data. So it is within the D-hull. Archetypal analysis represents data by convex combinations of archetypes which forms another hull called P-hull (c) To minimize the representation error, the P-hull should be large enough to represent all the data.

whose vertices are within the D-hull. In general cases, the learned archetypes approximately correspond to the extremal points in the dataset.

In spite of the nice interpretability, the method has not received sufficient attention because it is slow especially when the dataset is large. To scale to large training datasets, sparse coding uses online dictionary learning [20] to iteratively update the bases, where each iteration is based on a mini-batch of data. However, this learning scheme cannot be applied to AA because of the explicit binding between the data and the archetypes. More specifically, updating the archetypes needs to recompute the coefficients with respect to *all* the training data which makes mini-batch learning meaningless.

In this work, we propose a fast variant of archetypal analysis called Decoupled Archetypal Analysis (DAA). The core idea is to decouple the binding between the data and the archetypes, thus making mini-batch based dictionary learning possible. The method first projects the data onto the unit sphere without severely distorting the data structures. Then it learns a set of archetypes on the sphere. Instead of requiring the archetypes to be within the D-hull as in the original AA method, we require them to be within the unit sphere which is a relaxation of the original requirement.

As shown in Fig. 2(b), the method learns a convex hull (triangle in the figure) *within* the sphere and represents data using their surfaces. By minimizing the representation error, DAA drives the surfaces to be close to the data. Thus the convex hull vertices are pushed towards the extremal points. The union of the surfaces resembles the P-hull in AA.

The decoupling between data and archetypes enables us to use an online algorithm (similar to [20]) to optimize the problem which scales gracefully to large datasets. For example, we learn 400 archetypes from 300K human pose data using only several minutes. We present extensive experiments on realistic data sources including human faces and poses. In particular, we obtain decent performance on 3D human pose estimation and pose-based action recognition which are comparable to the state-of-the-arts.

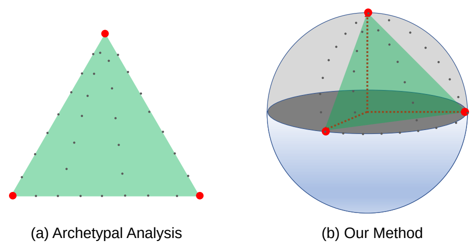

Fig. 2: Comparisons of (a) archetypal analysis and (b) our method. The gray and red dots represent the training data and the learned archetypes, respectively. Our method first embeds the two-dimensional data to a 3D unit sphere and learns a convex hull whose surfaces are close to the data.

2 Related Work

We briefly discuss the related work on unsupervised learning including sparse coding, non-negative matrix factorization and manifold learning.

2.1 Sparse Coding and Non-negative Matrix Factorization

It is natural to compare our approach to sparse coding [33, 25, 9, 24] which activates only a small number of bases when representing a datum. Sparse coding enforces that the l_1-norm of the coefficients is smaller than a threshold λ. In contrast, our method requires that the l_1-norm of the coefficients equals one. In addition, the representation is also sparse which we will discuss in more details later. Another related work is non-negative matrix factorization, which assumes the data are non-negative and learns two non-negative and rank-reduced factor matrices such that their product approximates the input data. Some variants of sparse coding such as [13] incorporate these two methods and require non-negative coefficients in addition to sparsity in order to learn additive object parts features. Our approach differs from the above approaches in that we require the data and bases to be unit normalized and enforce joint non-negative and sum-to-one constraints.

2.2 Manifold Learning

Principal Component Analysis (PCA) is a common method for representing high-dimensional data in terms of bases. It is well suited for data which lie in a linear space or equivalently on a low-dimensional linear manifold. But it is inappropriate for data which lie in non-linear spaces. The limitations of PCA motivate researchers to develop approaches for detecting low-dimensional manifold structures in high-dimensional data [31, 27, 12, 1, 7, 39]. Typically these methods use projection methods that preserve local properties of the data (*e.g.*, geodesic distance). These methods are global, in that a single low dimensional structure is used in the representation of the data and they typically assume that the manifold is connected. Although our approach also represents data by bases (archetypes), it does not explicitly learn the low-dimensional data manifolds as the above approaches. Instead it learns a set of representative bases which correspond to the extremal data to best represent the dataset.

3 Revisit Archetypal Analysis

We consider a dataset $\mathbf{X} = \{\mathbf{x}_1, \cdots, \mathbf{x}_n\}$ having n training data where $\mathbf{x}_i \in \mathbb{R}^m$ is a datum. Archetypal analysis represents a datum $\mathbf{x}$ by convex combinations of archetypes $\mathbf{Z} = \{\mathbf{z}_1, \cdots, \mathbf{z}_k\}$: $\mathbf{x} = \mathbf{Z}\boldsymbol{\alpha}$ where $\boldsymbol{\alpha}$ resides in the simplex Δ_k:

$$\Delta_k \triangleq \left\{\boldsymbol{\alpha} \in \mathbb{R}^k \text{ s.t. } \boldsymbol{\alpha} \succeq 0 \text{ and } |\boldsymbol{\alpha}|_1 = 1\right\}. \tag{1}$$

Meanwhile each archetype $\mathbf{z}$ is required to be chosen from the convex combinations of the training data: $\mathbf{z} = \mathbf{X}\boldsymbol{\beta}$ where $\boldsymbol{\beta} \in \Delta_n$. The archetypes are learned by minimizing the representation error on the whole training dataset:

$$\begin{aligned} &\min_{\mathbf{A},\mathbf{B}} \|\mathbf{X} - \mathbf{XBA}\|_{\mathrm{F}}^2, \quad \mathbf{A} = [\boldsymbol{\alpha}_1, \cdots, \boldsymbol{\alpha}_n], \quad \mathbf{B} = [\boldsymbol{\beta}_1, \cdots, \boldsymbol{\beta}_k] \\ &\boldsymbol{\alpha}_i \in \Delta_k, \quad \text{for} \quad i = 1 \ldots n, \quad \boldsymbol{\beta}_j \in \Delta_n, \quad \text{for} \quad j = 1 \ldots k, \end{aligned} \tag{2}$$

where $\mathbf{Z} = \mathbf{XB}$ denotes the learned k archetypes.

It would help to give a geometric interpretation for AA which inspires our approach. There are two convex hulls which are constructed from the dataset (D-hull) and the archetypes (P-hull), respectively. See Fig. 1 for illustrations. Loosely speaking, the P-hull is within the D-hull and is optimized to have the largest overlap with the D-hull. Ideally, the P-hull is the same as the D-hull.

4 Decoupled Archetypal Analysis

We learn archetypes without requiring them to be chosen from convex combinations of data. However, if there is no constraint on the archetypes, the problem becomes ill-posed and has infinite numbers of solutions. We solve the problem by normalizing the data to lie on the unit sphere and the archetypes are within the unit sphere which can be regarded as a relaxation to the constraints in AA.

4.1 Project Data onto the Unit Sphere

We normalize each datum $\mathbf{x}_i$ such that it lies on the unit sphere: $\|\mathbf{x}_i\|_2 = 1, \ \forall i$. This is usually accomplished by directly normalizing the data: $\mathbf{x}_i \mapsto \frac{\mathbf{x}_i}{\|\mathbf{x}_i\|_2}$, or by centering and then normalizing: $\mathbf{x}_i \mapsto \frac{\mathbf{x}_i - \bar{\mathbf{x}}}{\|\mathbf{x}_i - \bar{\mathbf{x}}\|_2}$ where $\bar{\mathbf{x}}$ is the data mean $\frac{1}{n}\sum_{i=1}^n \mathbf{x}_i$. For many signals, little information is lost in normalization. Indeed the normalization is a common first step in many applications. We directly normalize the data in the following experiments in this paper unless stated elsewhere.

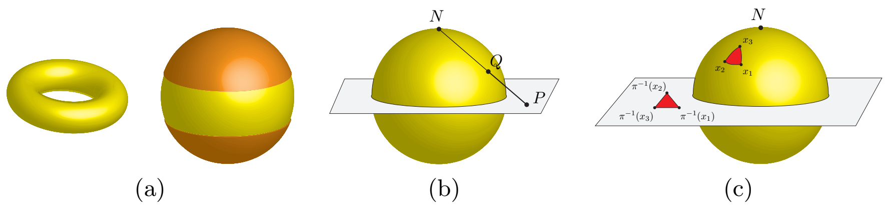

Fig. 3: (a) The torus is flattened into a band when normalized. (b) Stereographic projection from the equatorial plane into the sphere maps P to Q. (c) A triangle in the plane and the corresponding curved triangle on the sphere.

However, some structures may collapse under direct normalization. Consider, for example, the usual two-dimensional torus. Fig. 3(a) shows the torus and its normalization in the sphere. The normalization flattens the torus into a band around the equator and the inner cavity is lost. To avoid data collision, we use stereographic projection to map data in $\mathbb{R}^d$ into the d dimensional sphere $\mathbb{S}^d$ in $\mathbb{R}^{d+1}$. Fig. 3(b) shows stereographic projection from the equatorial plane onto the sphere, through the north pole N. The point P is mapped to the point Q, which is the intersection of the line through N, P and the unit sphere. It is worth noting that this is an invertible conformal map because it preserves angles between curves. The mapping is: $P \mapsto Q = N + \frac{2}{1+\|P\|^2}(P - N)$, and the inverse mapping is $Q \mapsto P = N + \frac{1}{1-Q_{d+1}}(Q - N)$, where Q_{d+1} is the $(d+1)$-th coordinate of Q. In some situations, the entire data analysis can be carried out in the normalized space rather than in the original space. But for some situations, *e.g.*, for visualization purposes, it would help to map the learned bases back to the original space. Fig. 3(c) shows an example in which the learned bases are back-projected to the original space.

4.2 Learning the Archetypes

We now describe how to learn a set of archetypes $\mathbf{D} = [\mathbf{d}_1, \cdots, \mathbf{d}_k]$ from the normalized data. We first give the formulation and then explain why the formulation can learn the archetypes. The following is the proposed formulation:

$$\begin{aligned} &\min_{\mathbf{D},\mathbf{C}} \|\mathbf{X} - \mathbf{D}\mathbf{C}\|_{\mathrm{F}}^2, \quad \mathbf{C} = [\mathbf{c}_1, \cdots, \mathbf{c}_n], \\ &\mathbf{c}_i \in \Delta_k, \quad \|\mathbf{x}_i\|_2 = 1, \quad \mathbf{D} \in \mathcal{C}, \end{aligned} \tag{3}$$

where $\mathcal{C} \triangleq \{\mathbf{D} \in \mathbb{R}^{m\times k} \text{ s.t. } \forall j = 1 \ldots k,\ \|\mathbf{d}_j\|_2^2 \le 1.\}$

Since the learned archetypes lie inside the unit sphere, their convex combinations will form a convex hull *inside* the unit sphere. See Fig. 4 for an illustration. Since the data are also normalized to lie on the unit sphere, they are guaranteed to lie *outside* the convex hull. So in the reconstruction stage, the data will be projected onto the *surfaces* of the convex hull so as to minimize the representation error. In the learning stage, to minimize the overall representation error on

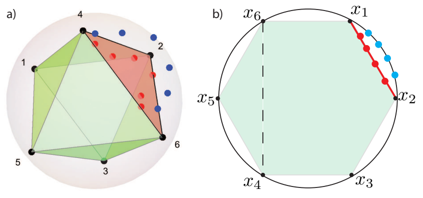

Fig. 4: Illustration of data reconstructions in 3D and 2D. The black points are the learned archetypes and the blue/cyan points are the data.

the whole dataset, the algorithm will push the surfaces as close to the data as possible. The union of the set of convex hull surfaces (which data are projected on) resembles the P-hull as defined in the original archetypal analysis. The convex hull vertices are the learned archetypes. In summary, directly optimizing the problem (3) will learn archetypes which approximately correspond to the extremal points in the dataset.

4.3 Represent Data by Archetypes

After learning the archetypes, we represent a datum x by convex combinations of the archetypes: $\hat{\mathbf{x}} = \mathbf{D}\mathbf{c}^*$ where

$$\mathbf{c}^* \triangleq \arg\min_{\mathbf{c}\in\Delta_k} \|\mathbf{x} - \mathbf{D}\mathbf{c}\|_2^2 \tag{4}$$

We would like to note that by solving the equation (4), the datum is projected to the surface of the convex hull. In ideal cases, the difference between $\mathbf{x}$ and the reconstructed datum $\hat{\mathbf{x}}$ is small.

One interesting property of our method is that only the archetypes on the same surface can be activated at the same time (co-activated). For example, in Fig. 4(b), the archetypes $\mathbf{x}_4$ and $\mathbf{x}_6$ will never be co-activated to represent a datum. If they are co-activated, the datum would be projected into the interior of the convex hull, which is impossible. This is because projecting the datum to the interior will first cross a surface and projecting the datum to that surface achieves smaller error. So the archetypes learned using our method are automatically clustered in groups which provides a tight representation of the datasets. This is a desirable property for reconstruction and classification tasks which need strong regularization capabilities.

4.4 Geometric Comparison with Related Work

To highlight their differences to our approach, we provide geometric interpretations for sparse coding and non-negative matrix factorization. Sparse coding represents a datum by a small number of bases by constraining the l_1-norm of the coefficient vector $\|\boldsymbol{\alpha}\|_1$ to be smaller than a constant λ, which could be interpreted geometrically. We augment each basis $\mathbf{d}$ by its negative $-\mathbf{d}$ and require non-negative coefficients as shown in Fig. 5(a). Sparse coding projects a datum to a scaled convex hull (cyan in Fig. 5(a)) formed by the bases where the scale is $\|\boldsymbol{\alpha}\|_1$ which varies for different data. The variation of the scale makes it non-trivial to interpret the properties of the bases. In contrast, the scale of the convex hull for our approach is fixed to be one.

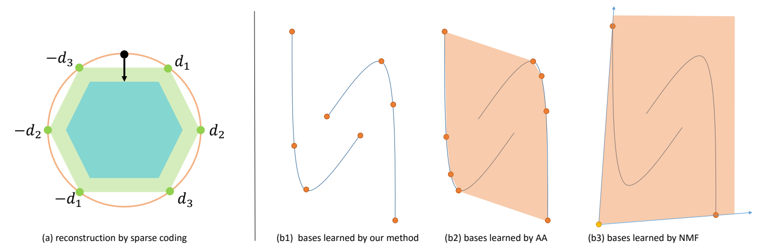
$-d_3$
d_1
$-d_2$
d_2
$-d_1$
d_3
(a) reconstruction by sparse coding
(b1) bases learned by our method
(b2) bases learned by AA
(b3) bases learned by NMF

Fig. 5: Geometric interpretation of sparse coding and comparison of different methods on a naive dataset. The orange points represent the learned bases and the points within the orange area could be reconstructed without any error.

We also compare different methods on a naive dataset in Fig. 5(b). Geometrically, data lie in a convex polyhedral cone and NMF aims to find the edges of the cone. Similar to archetypal analysis, this method also tends to learn a large convex hull for non-convex datasets but does not explore the local structures of the manifold. Instead, our method tends to use a mixture of small surfaces to approximate the local non-convex regions.

5 Optimization

We sketch the optimization algorithm in this section. Although the problem (3) is not convex with respect to $\mathbf{D}$ and $\mathbf{C}$, it is convex with respect to one of them with the other fixed. We propose an online learning algorithm similar to [20]. At each iteration, only a mini-batch of samples are used to update the dictionary of archetypes, endowing us with the ability to handle large-scale datasets. Similarly as [20], the sufficient statistics with respect to coefficients $\mathbf{c}_i$ are aggregated, which are used for dictionary updates.

5.1 Algorithm Outline

The outline is shown in Algorithm 1. At each iteration t, we deal with one sample $\mathbf{x}_t$ (the extension to multiple samples is straightforward). Given the sample for current iteration, we first compute the coefficient by solving the equation (5) based on the dictionary $\mathbf{D}_{t-1}$ computed in the last iteration. Then the sufficient statistics of $\mathbf{c}_i$, namely $\mathbf{A}$ and $\mathbf{B}$, are updated. Finally, the dictionary is updated

Algorithm 1 Online dictionary learning

```
Require: Data $\mathbf{X}$ in $\mathbb{R}^{m\times n}$, $k$ (number of archetypes), $T$ (number of iterations)
 1: $\mathbf{A}_0 \leftarrow 0, \mathbf{B}_0 \leftarrow 0$ (reset past information).
 2: Initialize $\mathbf{D}$ in $\mathbb{R}^{m\times k}$ with random columns from $\mathbf{X}$.
 3: for $t = 1 \rightarrow T$ do
 4:     Sample $\mathbf{x}_t$ from $\mathbf{X}$.
 5:     Compute coefficient $\mathbf{c}_t$ for $\mathbf{x}_t$ by solving equation (5)
 6:     $\mathbf{A}_t \leftarrow \mathbf{A}_{t-1} + \mathbf{c}_t\mathbf{c}_t^T$.
 7:     $\mathbf{B}_t \leftarrow \mathbf{B}_{t-1} + \mathbf{x}_t\mathbf{c}_t^T$.
 8:     Compute dictionary $\mathbf{D}_t$ using equation (6)
 9: end for
10: return $\mathbf{D}_T$
```

based on historical observed data, by minimizing $\hat{f}_t(\mathbf{D}) \triangleq \frac{1}{t}\sum_{i=1}^{t} ||\mathbf{x}_i - \mathbf{D}\mathbf{c}_i||_2^2$, where the coefficients $\mathbf{c}_i$ are obtained at iteration i of the algorithm. It acts as an approximation to $\hat{\mathbf{c}}_i(\mathbf{D}_{t-1})$ which denotes recomputing the coefficients for all the data in previous iterations based on the update-to-date dictionary $\mathbf{D}_{t-1}$. We will describe the details for computing the coefficients and updating the dictionary in the following subsections.

5.2 Compute Coefficients

For calculating the coefficient $\mathbf{c}_t$ given current dictionary $\mathbf{D}_{t-1}$, we need to solve the following problem:

$$\mathbf{c}_t \triangleq \arg\min_{\mathbf{c}\in\Delta_k} ||\mathbf{x}_t - \mathbf{D}_{t-1}\mathbf{c}||_2^2. \tag{5}$$

which is a least-squares optimization problem with simplicial constraint. Several approaches have been used for this kind of problem. Culter et al. [6] proposed a penalty approach. Also, fast iterative shrinkage-thresholding algorithm (FISTA) could be used. Following [4], we use an active-set algorithm, which benefits from the sparsity of the coefficient $\mathbf{c}$. It is worth noting that the coefficients obtained by the equation (5) are sparse because of the interaction of curved geometry of the sphere and the linear surfaces, which encourages using low dimensional surfaces to represent the data. The maximum number of activated archetypes is determined by the number of vertices of the surface.

5.3 Update Dictionary

We update the dictionary using the sufficient statistics $\mathbf{A}$ and $\mathbf{B}$:

$$\begin{aligned}\mathbf{D}_t &\triangleq \arg\min_{\mathbf{D}\in\mathcal{C}} \frac{1}{t}\sum_{i=1}^{t} \|\mathbf{x}_i - \mathbf{D}\mathbf{c}_i\|_2^2 \\ &= \arg\min_{\mathbf{D}\in\mathcal{C}} \frac{1}{t}(\mathrm{Tr}(\mathbf{D}^T\mathbf{D}\mathbf{A}_t) - 2\,\mathrm{Tr}(\mathbf{D}^T\mathbf{B}_t)).\end{aligned} \tag{6}$$

Since this formula is the same as the one in [20], we utilize the same dictionary update scheme. Block-coordinate descent approach with warm restart is used, and the main advantage of it is parameter free.

We sequentially update the column $\mathbf{d}_j$, while fixing the others. Firstly, the algorithm calculates a new position $\mathbf{u}_j = \mathbf{d}_j + (1/\mathbf{A}[j,j])(\mathbf{b}_j - \mathbf{D}\mathbf{a}_j)$, where $\mathbf{b}_j$ and $\mathbf{a}_j$ are j-th column of $\mathbf{A}$ and $\mathbf{B}$ respectively. Then we project $\mathbf{u}_j$ into the unit sphere by $\mathbf{d}'_j = \mathbf{u}_j / \max(\|\mathbf{u}_j\|_2, 1)$ to satisfy the constraint $\|\mathbf{d}_j\| \leq 1$. The convergence of this convex optimization problem is guaranteed.

6 Experiments

We evaluate the proposed method from three aspects. First, we evaluate the learning speed on large-scale datasets and compare with the original archetypal analysis method [4]. Second, we experimentally demonstrate that the convex combinations of the learned archetypes accurately approximate the D-hull. In particular, the archetypes correspond to the extremal points in the dataset. Third, we give applications of using the archetypes for data modeling in the context of reconstruction and classification tasks[1].

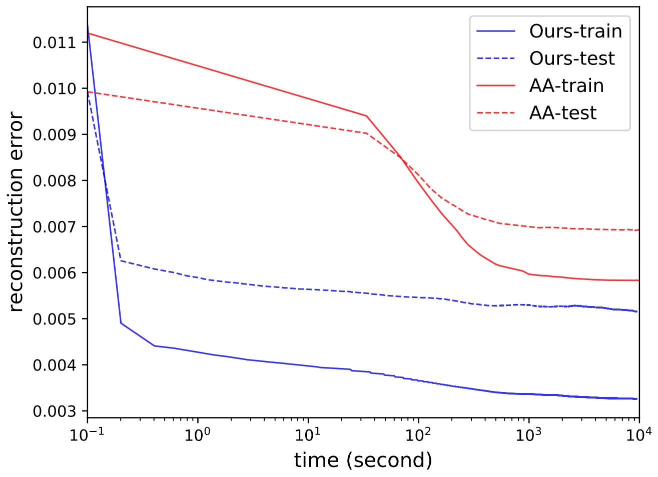

Fig. 6: Comparison of the learning speed for our approach and [4] on the H3.6M dataset. The results of our method and [4] are represented by blue and red lines, respectively. The solid line denotes the results on the training set and the dotted line denotes the results on the testing set. For [4], the points are plotted for each iteration. For our method, the points are plotted for every ten batches.

[1] We give more experiment results in the supplementary files

6.1 Speed

We implement our algorithm based on Eigen [11]. For the original AA method [4], we use the fastest implementation included in the SPAMS toolbox [20]. Both approaches use Intel MKL and multi-threading. The experiments are done on a single Intel CPU of i7-5930K.

We experiment on the H3.6M dataset [16] which has about three million human poses. The dimension of a pose is 48. Following the protocol in [22], we sample about 300K training poses and 3000 testing poses. We learn 400 archetypes from the dataset. The results are shown in Fig. 6.

First, the reconstruction error of our approach on the training dataset drops to a lower level than [4]. This is expected as the archetypes in our approach have larger flexibility because they can be anywhere in the sphere. This differs from [4] where the archetypes can only be in the D-hull. Second, our approach uses much less time than [4] to reach similar levels of reconstruction errors. In particular, when the first iteration (which has to traverse *the whole* dataset once) of archetypal analysis is finished, our method has updated the dictionary for about 1700 times (around 3 epochs) and obtained a small reconstruction error. The gap will become even larger when the number of training data increases.

6.2 Interpretability

One distinctive property of archetypal analysis [6, 4] is that some of the learned archetypes correspond to the extremal points in the dataset. In this case, the convex combinations of the archetypes will form a convex hull which tightly surrounds the data hull. In this section, we examine the archetypes learned on two data sources: human faces and poses.

Face Images It is known that the imaging of an object in variable illumination conditions can be accurately approximated by convex combinations of a set of images captured at extreme illumination conditions [10]. Hence this is a good data source to visually evaluate whether the proposed approach can identify the extremal points as the AA method.

We experiment on the cropped Yale face dataset [10]. We normalize each image to have unit l_2-norm for our method. The archetypal analysis directly uses the original images. The results are shown in Fig. 7. For visualization purposes, for each archetype, we find the nearest neighbor in the face dataset and highlight that face image in a particular color. The face images with blue rectangles denote the archetypes learned by our method. We also visualize the archetypes learned by AA in red rectangles. We can see that for both methods, the archetypes mostly correspond to the extremal illumination conditions, and the convex hull of the archetypes largely overlapped with the convex hull of the images, which justifies the effectiveness of our method.

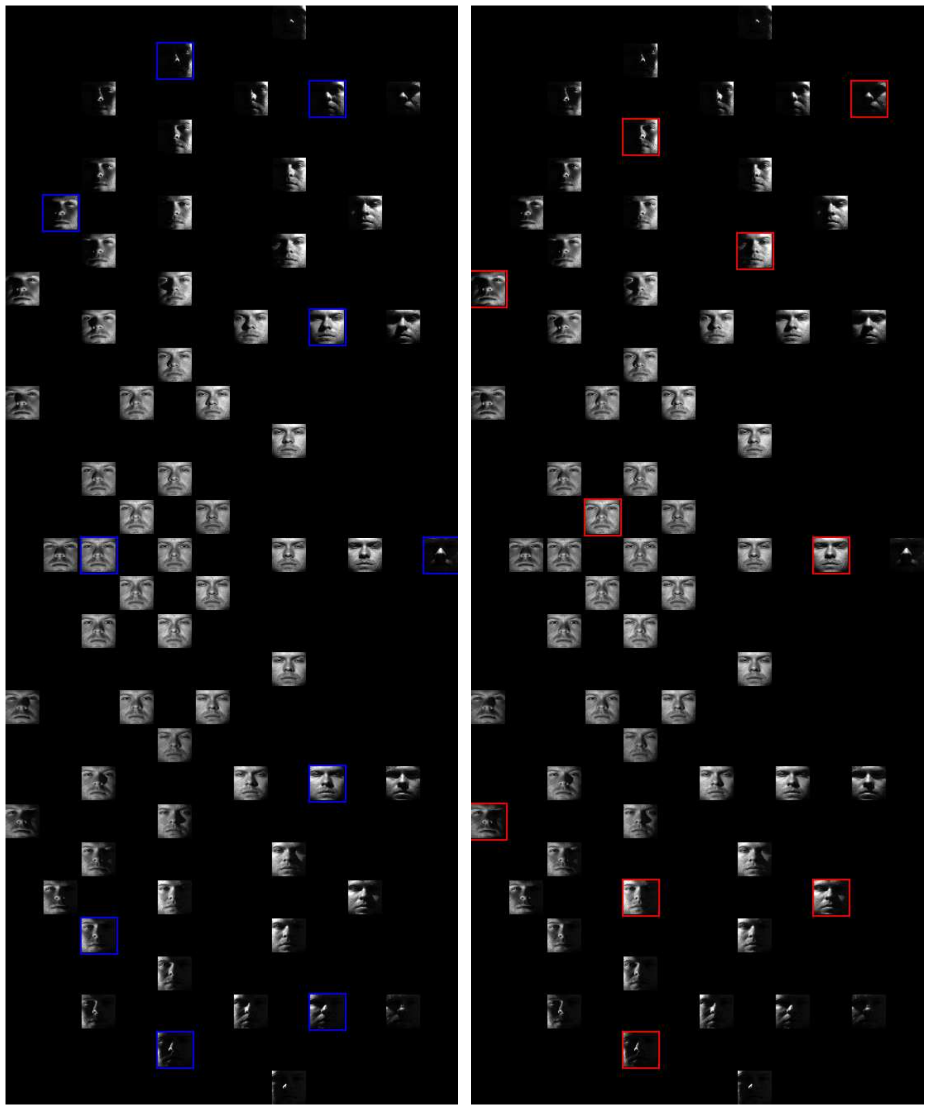

Fig. 7: This figure shows 64 images from the Yale face dataset which correspond to the same person under different lighting directions. We arrange the images according to the two axis-es of the lighting directions. The nearest neighbors of archetypes identified by our method and archetypal analysis are annotated with blue and red rectangles, respectively.

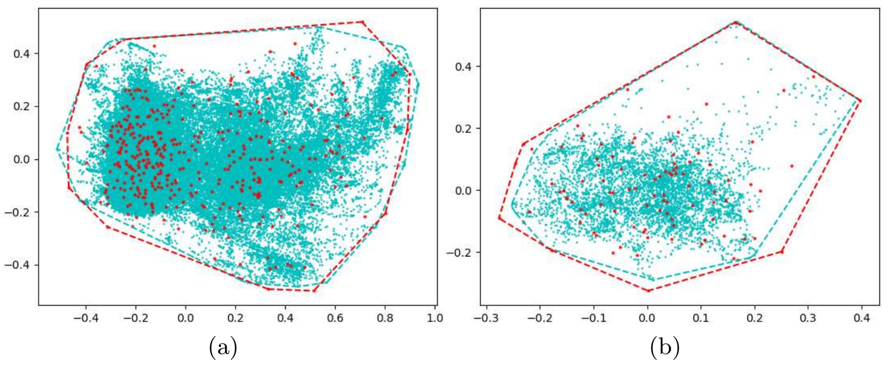

Fig. 8: Visualization of the learned archetypes on human poses. We use [36] to project the data into 2D. The data and the learned archetypes are represented by cyan and red points respectively. The convex hull of the data/archetypes is plotted using the dotted line. (a) the archetypes learned on all poses. (b) the archetypes learned on the poses of a particular action.

Pose We also experiment on the large 3D human pose dataset H3.6M. We visualize the poses and the learned archetypes by projecting them to two-dimensional space using PCA [36]. The results are shown in Fig. 8. We can see that some of the learned archetypes correspond to the extremal points in the dataset. In addition, the convex combinations of the archetypes form a convex hull which tightly surrounds the training dataset. The left and right figures show the results learned on all poses and on poses of a particular action respectively.

6.3 Applications

In this section, we use the learned archetypes as a data modeling tool for 3D human pose estimation and pose-based action recognition where strong regularization capabilities would help.

3D Pose Estimation A 2D pose is represented by d joint locations in 2D $\mathbf{x} \in \mathbb{R}^{2d}$. Similarly, a 3D pose is denoted by $\mathbf{y} \in \mathbb{R}^{3d}$. The 2D and 3D pose are related by the camera parameters $\mathbf{M} : \mathbf{x} = \mathbf{My}$ [26].

The task of 3D human pose estimation is to estimate the 3D pose $\mathbf{y}$ and the camera parameters $\mathbf{M}$ from a 2D pose $\mathbf{x}$. It is usually achieved by minimizing $\|\mathbf{x} - \mathbf{My}\|_2$ which is an ill-posed problem. The authors of [26] propose to reduce the ambiguities by representing a 3D pose by a set of PCA bases $\mathbf{B}$ and seeking for the optimal coefficient for a testing 2D pose: $\mathbf{M}^*, \boldsymbol{\beta}^* = \arg\min_{\mathbf{M},\boldsymbol{\beta}} \|\mathbf{x} - \mathbf{MB}\boldsymbol{\beta}\|_2$. The estimated 3D pose is $\hat{\mathbf{y}} = \mathbf{B}\boldsymbol{\beta}^*$.

We replace the PCA bases in [26] with the archetypes learned by our approach. In particular, we learn 400 archetypes on the training set. We also compare with a related baseline which uses the bases learned by sparse coding [20]. The sparse bases are learned with the regularization factor being set as $\lambda = 1.2/\sqrt{3d}$ as suggested in [20] where $3d$ is the data dimension.

We experiment on the H3.6M dataset where the groundtruth 2D poses are assumed known. To reveal the properties of different bases (or archetypes), we calculate three metrics: (1) the projection error in 2D which is computed as $\|\mathbf{x} - \mathbf{M}\mathbf{B}\boldsymbol{\beta}^*\|$ when groundtruth camera parameters are known; the unit is pixel; (2) the mean per joint position error (MPJPE) in 3D when groundtruth camera parameters are known; the unit is mm; (3) MPJPE in 3D when camera parameters are jointly estimated.

Table 1: Pose estimation results on the H3.6M dataset. Results using estimated 2D pose are listed in parentheses for comparison with state-of-the-arts.

Method	projection error	MPJPE	
		known camera	estimated camera
Chen et al. [3]	-	-	- (114.18)
Tome et al. [34]	-	-	- (88.39)
Zhou et al. [40]	-	-	- (79.90)
Sun et al. [30]	-	-	- (59.10)
sparse coding	0.37	231.82	252.86 (276.75)
archetypal analysis	4.13	52.56	75.93 (126.88)
our method	3.68	50.71	76.91 (106.52)

The results are shown in Table 1. First, we can see that the projection error of sparse coding is much smaller than archetypal analysis and our method. However, the 3D pose estimation error is much larger. This is because the bases learned by sparse coding have weaker regularization properties. In other words, sparse coding doesn't provide a tight/bounded representation for the data manifold as our method. Second, the result of our method is similar to archetypal analysis. Third, the final error 106.52mm is achieved when the 2D poses are estimated by [2] and the camera parameters are jointly estimated using our method. The result is comparable to some state-of-the-arts which rely on deep networks.

Pose-based Action Recognition We also evaluate our approach on a classification task of pose-based action recognition. We adopt a simple nearest neighbor based classification framework. For a pose sequence, following [35], we extract a set of snippets with each being a short sequence of poses, to represent the original sequence. Then for each action class, we learn a set of archetypes based on the snippets. In the testing stage, for a pose sequence, we reconstruct its snippets using the archetypes of each class and the class which achieves the minimum average reconstruction error over all the snippets of the sequence is the predicted action class.

We experiment on the NTU RGB+D dataset [28], which is a large benchmark consisting of four million poses. The number of archetypes is set to 1024 by cross-validation. For sparse coding, the regularization factor is set to 0.1 by cross-validation. Following previous works, we report the results using two different train/test split methods: Cross-Subject (CS) and Cross-View (CV).

Table 2: Action recognition accuracy (%) on the NTU RGB+D dataset with Cross-Subject and Cross-View settings.

Method	CS	CV
LieNet [15]	61.37	66.95
PA-LSTM [28]	62.93	70.27
ST-LSTM (Tree Traversal) + Trust Gate [19]	69.20	77.70
STA-LSTM [29]	73.40	81.20
Ensemble TS-LSTM [18]	74.60	81.25
VA-LSTM [38]	**79.40**	**87.60**
archetypal analysis	-	-
sparse coding	64.95	77.61
our method	68.07	80.93

The experimental results are shown in Table 2. It is not feasible to learn archetypes using [4] on such a large dataset so we don't list its results. We observe our method outperforms sparse coding by 3.12% and 3.32% in accuracy for CS and CV settings respectively, as our method learns a tighter representation than sparse coding. We also compare with the state-of-the-art methods which take advantage of deep neural networks such as [28, 15, 19, 29, 18, 38]. The recognition accuracy of our method is lower than VA-LSTM. The reason may be that our method doesn't model the motions as LSTM. However, considering that our method is simple and not tuned for the action recognition task, the results are already promising.

7 Conclusion

We present a fast variant of archetypal analysis, which not only maintains the favorable interpretability, but also scales gracefully to large datasets. It decouples the deep binding between data and archetypes, thus enables an online optimization algorithm. This formulation learns tight representations which is a favorable property for many tasks. Experiments on pose estimation and pose-based action recognition reveal its power.

References

1. Belkin, M., Niyogi, P.: Laplacian eigenmaps for dimensionality reduction and data representation. Neural Computation **15**(6), 1373–1396 (2003)

2. Cao, Z., Simon, T., Wei, S., Sheikh, Y.: Realtime multi-person 2d pose estimation using part affinity fields. In: CVPR. pp. 1302–1310 (2017)
3. Chen, C., Ramanan, D.: 3d human pose estimation = 2d pose estimation + matching. In: CVPR. pp. 5759–5767 (2017)
4. Chen, Y., Mairal, J., Harchaoui, Z.: Fast and robust archetypal analysis for representation learning. In: CVPR. pp. 1478–1485 (2014)
5. Coates, A., Ng, A.Y.: Learning feature representations with k-means. In: Neural Networks: Tricks of the Trade - Second Edition, pp. 561–580. Springer (2012)
6. Cutler, A., Breiman, L.: Archetypal analysis. Technometrics **36**(4), 338–347 (1994)
7. Donoho, D.L., Grimes, C.: Hessian eigenmaps: Locally linear embedding techniques for high-dimensional data. Proceedings of the National Academy of Sciences **100**(10), 5591–5596 (2003)
8. Duda, R.O., Hart, P.E., Stork, D.G.: Pattern classification. Wiley, New York (1973)
9. Efron, B., Hastie, T., Johnstone, I., Tibshirani, R., et al.: Least angle regression. The Annals of statistics **32**(2), 407–499 (2004)
10. Georghiades, A.S., Belhumeur, P.N., Kriegman, D.J.: From few to many: Illumination cone models for face recognition under variable lighting and pose. TPAMI **23**(6), 643–660 (2001)
11. Guennebaud, G., Jacob, B., et al.: Eigen v3. http://eigen.tuxfamily.org (2010)
12. Ham, J., Lee, D.D., Mika, S., Schölkopf, B.: A kernel view of the dimensionality reduction of manifolds. In: ICML (2004)
13. Hoyer, P.O.: Non-negative sparse coding. In: Proceedings of the 12th IEEE Workshop on Neural Networks for Signal Processing, NNSP 2002, Martigny, Valais, Switzerland, September 4-6, 2002. pp. 557–565 (2002)
14. Hoyer, P.O.: Non-negative matrix factorization with sparseness constraints. Journal of Machine Learning Research **5**, 1457–1469 (2004)
15. Huang, Z., Wan, C., Probst, T., Gool, L.V.: Deep learning on lie groups for skeleton-based action recognition. In: CVPR. pp. 1243–1252 (2017)
16. Ionescu, C., Papava, D., Olaru, V., Sminchisescu, C.: Human3.6m: Large scale datasets and predictive methods for 3d human sensing in natural environments. TPAMI **36**(7), 1325–1339 (2014)
17. Lee, H., Battle, A., Raina, R., Ng, A.Y.: Efficient sparse coding algorithms. In: NIPS. pp. 801–808 (2006)
18. Lee, I., Kim, D., Kang, S., Lee, S.: Ensemble deep learning for skeleton-based action recognition using temporal sliding LSTM networks. In: ICCV. pp. 1012–1020 (2017)
19. Liu, J., Shahroudy, A., Xu, D., Wang, G.: Spatio-temporal LSTM with trust gates for 3d human action recognition. In: ECCV. pp. 816–833 (2016)
20. Mairal, J., Bach, F.R., Ponce, J., Sapiro, G.: Online dictionary learning for sparse coding. In: ICML. pp. 689–696 (2009)
21. Mallat, S.: A wavelet tour of signal processing. Academic press (1999)
22. Martinez, J., Hossain, R., Romero, J., Little, J.J.: A simple yet effective baseline for 3d human pose estimation. In: ICCV. pp. 2659–2668 (2017)
23. Mika, S., Ratsch, G., Weston, J., Scholkopf, B., Mullers, K.R.: Fisher discriminant analysis with kernels. In: Neural networks for signal processing IX, 1999. Proceedings of the 1999 IEEE signal processing society workshop. pp. 41–48. IEEE (1999)
24. Olshausen, B.A., et al.: Emergence of simple-cell receptive field properties by learning a sparse code for natural images. Nature **381**(6583), 607–609 (1996)
25. Osborne, M.R., Presnell, B., Turlach, B.A.: On the lasso and its dual. Journal of Computational and Graphical statistics **9**(2), 319–337 (2000)

26. Ramakrishna, V., Kanade, T., Sheikh, Y.: Reconstructing 3d human pose from 2d image landmarks. In: ECCV. pp. 573–586 (2012)
27. Saul, L.K., Roweis, S.T.: Think globally, fit locally: unsupervised learning of low dimensional manifolds. Journal of machine learning research **4**(Jun), 119–155 (2003)
28. Shahroudy, A., Liu, J., Ng, T., Wang, G.: NTU RGB+D: A large scale dataset for 3d human activity analysis. In: CVPR. pp. 1010–1019 (2016)
29. Song, S., Lan, C., Xing, J., Zeng, W., Liu, J.: An end-to-end spatio-temporal attention model for human action recognition from skeleton data. In: AAAI. pp. 4263–4270 (2017)
30. Sun, X., Shang, J., Liang, S., Wei, Y.: Compositional human pose regression. In: ICCV. pp. 2621–2630 (2017)
31. Tenenbaum, J.B., De Silva, V., Langford, J.C.: A global geometric framework for nonlinear dimensionality reduction. science **290**(5500), 2319–2323 (2000)
32. Thompson, B.: Canonical correlation analysis. Encyclopedia of statistics in behavioral science (2005)
33. Tibshirani, R.: Regression shrinkage and selection via the lasso. Journal of the Royal Statistical Society. Series B (Methodological) pp. 267–288 (1996)
34. Tomè, D., Russell, C., Agapito, L.: Lifting from the deep: Convolutional 3d pose estimation from a single image. In: CVPR. pp. 5689–5698 (2017)
35. Wang, C., Wang, Y., Yuille, A.L.: Mining 3d key-pose-motifs for action recognition. In: CVPR. pp. 2639–2647 (2016)
36. Wold, S., Esbensen, K., Geladi, P.: Principal component analysis. Chemometrics and intelligent laboratory systems **2**(1-3), 37–52 (1987)
37. Yu, G., Sapiro, G., Mallat, S.: Solving inverse problems with piecewise linear estimators: From gaussian mixture models to structured sparsity. IEEE Trans. Image Processing **21**(5), 2481–2499 (2012)
38. Zhang, P., Lan, C., Xing, J., Zeng, W., Xue, J., Zheng, N.: View adaptive recurrent neural networks for high performance human action recognition from skeleton data. In: ICCV. pp. 2136–2145 (2017)
39. Zhang, Z., Zha, H.: Principal manifolds and nonlinear dimensionality reduction via tangent space alignment. SIAM J. Scientific Computing **26**(1), 313–338 (2004)
40. Zhou, X., Zhu, M., Pavlakos, G., Leonardos, S., Derpanis, K.G., Daniilidis, K.: Monocap: Monocular human motion capture using a CNN coupled with a geometric prior. CoRR **abs/1701.02354** (2017)