Archetypal Analysis as an Autoencoder

Conference Paper · October 2015

CITATION
1

READS
534

4 authors:

Christian Bauckhage
University of Bonn
277 PUBLICATIONS **4,144** CITATIONS
SEE PROFILE

Kristian Kersting
University of Bonn
304 PUBLICATIONS **3,318** CITATIONS
SEE PROFILE

Florian Hoppe
Christian-Albrechts-Universität zu Kiel
8 PUBLICATIONS **96** CITATIONS
SEE PROFILE

Christian Thurau
Game Analytics
65 PUBLICATIONS **1,267** CITATIONS
SEE PROFILE

Archetypal Analysis as an Autoencoder

C. Bauckhage[1], K. Kersting[2], F. Hoppe[3], and C. Thurau[3]

[1] Fraunhofer IAIS, St. Augustin, Germany
[2] TU Dortmund, Dortmund, Germany
[3] Twenty Billion Neurons GmbH, Berlin, Germany

Abstract. We present an efficient approach to archetypal analysis where we use sub-gradient algorithms for optimization over the simplex to determine archetypes and reconstruction coefficients. Runtime evaluations reveal our approach to be notably more efficient than previous techniques. As an practical application, we consider archetypal analysis for autoencoding.

1 Introduction

Archetypal analysis is a matrix factorization method specifically conceived for latent factor analysis [8]. Since it allows for dimensionality reduction and sparse coding alike, it is applicable to feature extraction, clustering, or classification [3].

Contrary to related approaches, latent factors or *archetypes* found through archetypal analysis characterize extremes rather than averages. Archetypes do not rely on implicit density assumptions such as, for example, eigenvectors or cluster centroids which are tailored towards globally or locally Gaussian data. Rather, archetypal analysis introduces a form of symmetry into latent factor modeling: *archetypes are convex combinations of data points and data points are explained as of convex combinations of archetypes.*

It is therefore faithful to the nature of data. For instance, archetypes of non-negative data will be non-negative, too. Also, since archetypes are convex combinations of actual data, they closely resemble certain data points and thus do not require reification when interpreted. These properties are appealing in the sciences [5,16,23,24] as well as in computer vision [2,6,19,21,25,30].

However, computing optimal archetypes is an NP hard problem [1] and even though efficient approximations have become available [3,10,20,22,26], considerable computational costs still hamper the broader use of the method.

In this paper, we address this issue and propose a novel, highly efficient algorithm for archetypal analysis. Applying sub-gradient procedures for quadratic optimization over the simplex we observe clear runtime gains over previous methods so that archetypal analysis becomes applicable to very large data sets. As a corresponding practical application, we present and discuss first experiments on archetypal analysis as an autoencoder for joint feature learning in image analysis.

2 Archetypal Analysis, Properties, and Algorithms

The basic setting for archetypal analysis is as follows: Given an $m \times n$ data matrix $\boldsymbol{X} = [\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n]$ and an integer $k \leq \min\{m, n\}$, determine a column stochastic $n \times k$ matrix $\boldsymbol{B}$ and a column stochastic $k \times n$ matrix $\boldsymbol{A}$ such that $\boldsymbol{X} \approx \boldsymbol{XBA} = \boldsymbol{ZA}$.

The columns $\boldsymbol{z}_j$ of the $m \times k$ matrix $\boldsymbol{Z}$ are called the *archetypes* of the data. Since the column vectors $\boldsymbol{b}_j$ of $\boldsymbol{B}$ are stochastic, the entries of $\boldsymbol{B}$ obey

$$b_{ij} \geq 0 \quad \wedge \quad \sum_{i=1}^{n} b_{ij} = 1 \tag{1}$$

and each archetype $\boldsymbol{z}_j = \boldsymbol{Xb}_j$ is a convex combination of the data vectors in $\boldsymbol{X}$. As the column vectors $\boldsymbol{a}_i$ of $\boldsymbol{A}$ are stochastic, too, the entries of $\boldsymbol{A}$ obey

$$a_{ji} \geq 0 \quad \wedge \quad \sum_{j=1}^{k} a_{ji} = 1 \tag{2}$$

and we realize that archetypal analysis approximates each data vector $\boldsymbol{x}_i \approx \boldsymbol{Za}_i$ as a convex combination of the archetypes in $\boldsymbol{Z}$.

The problem of computing archetypal analysis can be cast as the following constrained quadratic optimization objective

$$\min_{\boldsymbol{A},\boldsymbol{B}} \; E = \left\|\boldsymbol{X} - \boldsymbol{XBA}\right\|^2 \tag{3}$$

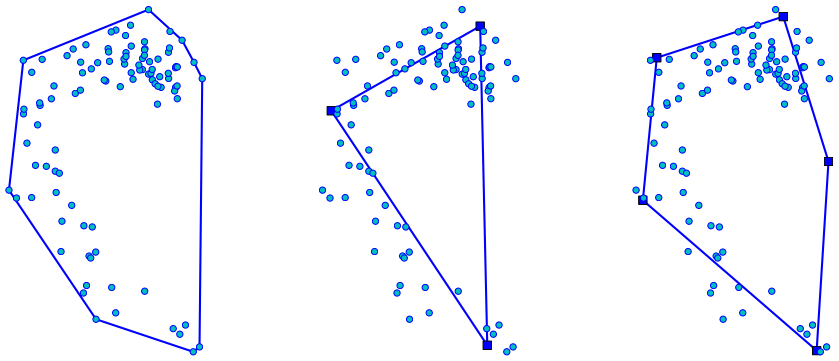
subject to the constraints in (1) and (2)

which is an NP-hard Euclidean sum of square clustering problem [1]. While E is convex in either $\boldsymbol{A}$ or $\boldsymbol{B}$, it is not convex in their product $\boldsymbol{AB}$ and typically has numerous local minima. Known solution strategies therefore randomly initialize both factor matrices and update them iteratively; we shall briefly discuss these algorithms below but first review some of the properties of archetypal analysis.

2.1 Properties of Archetypal Analysis

In [8], Cutler and Breiman prove that, if $k = 1$, the only archetype coincides with the sample mean; for $k > 1$, archetypes necessarily reside on the data convex hull and increasing the number of archetypes improves the approximation of the data convex hull (see Fig. 1).

Once suitable archetypes have been determined, each data point $\boldsymbol{x}_i$ can either be reconstructed exactly or approximated as a convex combination of the $\boldsymbol{z}_j$ (see Fig. 2). As the corresponding coefficient vector $\boldsymbol{a}_i$ is stochastic, it can be interpreted as a distribution over the archetypes and thus be embedded in a simplex spanned by the archetypes (see Fig. 2). This allows for soft clustering or classification since the coefficients a_{ji} correspond to probabilities $p(\boldsymbol{x}_i|\boldsymbol{z}_j)$ which indicate membership to classes or concepts represented by the archetypes $\boldsymbol{z}_k$.



(a) 100 data points $\boldsymbol{x}_i \in \mathbb{R}^2$ and their convex hull (b) $k = 3$ archetypes and respective archetypal hull (c) $k = 5$ archetypes and respective archetypal hull

Fig. 1: Archetypal analysis approximates the convex hull of a set of multivariate data. Increasing the number k of archetypes improves the approximation.

2.2 Traditional Algorithms for Archetypal Analysis

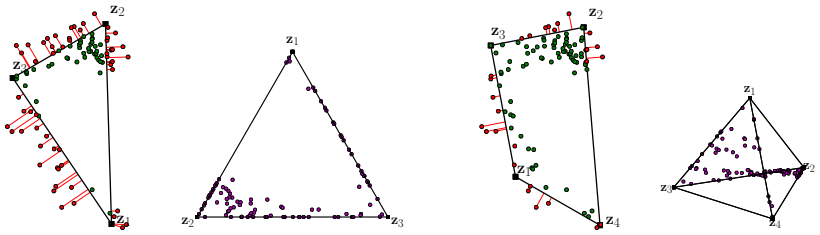
Cutler and Breiman [8] proposed an alternating least squares approach where they randomly initialize $\boldsymbol{B}$ and solve (3) for $\boldsymbol{A}$. Given $\boldsymbol{A}$, they solve (3) for $\boldsymbol{B}$ and repeat. This procedure provably converges towards a local minimum and can be implemented using common solvers for quadratic programming. For better efficiency, Bauckhage and Thurau [3] suggested intelligent initialization strategies and an active set algorithm and thus achieved significant accelerations.

Morup and Hansen [20] proposed an alternating projected gradient approach. Observing that

$$E = \left\|\boldsymbol{X} - \boldsymbol{X}\boldsymbol{B}\boldsymbol{A}\right\|^2 = \operatorname{tr}\left[\boldsymbol{X}^T\boldsymbol{X} - 2\boldsymbol{X}^T\boldsymbol{X}\boldsymbol{B}\boldsymbol{A} + \boldsymbol{A}^T\boldsymbol{B}^T\boldsymbol{X}^T\boldsymbol{X}\boldsymbol{B}\boldsymbol{A}\right] \tag{4}$$

we have $\nabla_{\boldsymbol{A}} E = 2\left[\boldsymbol{Z}^T\boldsymbol{Z}\boldsymbol{A} - \boldsymbol{Z}^T\boldsymbol{X}\right]$ and $\nabla_{\boldsymbol{B}} E = 2\left[\boldsymbol{X}^T\boldsymbol{X}\boldsymbol{B}\boldsymbol{A}\boldsymbol{A}^T - \boldsymbol{X}^T\boldsymbol{X}\boldsymbol{A}^T\right]$ so that archetypal analysis can also be computed by means alternating updates $\boldsymbol{A} \leftarrow \boldsymbol{A} - \eta_A\nabla_{\boldsymbol{A}}$ and $\boldsymbol{B} \leftarrow \boldsymbol{B} - \eta_B\nabla_{\boldsymbol{B}}$ where η_A and η_B are step size parameters. Since the gradient steps may lead out of the constraint sets, updated columns of $\boldsymbol{A}$ and $\boldsymbol{B}$ might not be stochastic and need to be projected back into their feasible regions which are the standard k and standard n simplex, respectively. Morup and Hansen, too, consider intelligent initializations for which they resort to the FASTMAP heuristic [11].

The methods in [3,20] run much faster than the original one [8]. Still, they invoke rather costly quadratic optimization routines or require costly projections of gradients onto a feasible set. Next, we propose an approach to archetypal analysis that avoids such overhead.



(a) visualization of the residual sum of squares (3) and simplicial embedding of coefficient vectors for $k = 3$

(b) visualization of the residual sum of squares (3) and simplicial embedding of coefficient vectors for $k = 4$

Fig. 2: While data inside an archetypal hull can accurately be expressed as convex combinations of archetypes, the constraints in (2) cause data on the outside to be mapped to the nearest point on the hull. For data point $\boldsymbol{x}_i$, the coefficient vector $\boldsymbol{a}_i$ is stochastic and thus resides in a simplex whose vertices correspond to the archetypes $\boldsymbol{z}_j$.

Algorithm 1 greedy Frank-Wolfe procedure to compute matrix $\boldsymbol{A}$ whose columns reside in the simplex Δ^{k-1}

Require: data matrix $\boldsymbol{X}$, matrix of archetypes $\boldsymbol{Z}$, and parameter $t_{\max} \in \mathbb{N}$
  $\boldsymbol{A} \leftarrow [\boldsymbol{e}_1, \boldsymbol{e}_1, \ldots, \boldsymbol{e}_1]$ where $\boldsymbol{e}_1 = [1, 0, \ldots, 0]^T \in \mathbb{R}^k$ // initialize $k \times n$ matrix $\boldsymbol{A}$
  $t \leftarrow 0$
  repeat
    $\boldsymbol{G} = \nabla_{\boldsymbol{A}} E = 2\left[\boldsymbol{Z}^T\boldsymbol{Z}\boldsymbol{A} - \boldsymbol{Z}^T\boldsymbol{X}\right]$ // compute gradient matrix
    for $i \in \{1, \ldots, n\}$ **do** // update columns $\boldsymbol{a}_i$ of $\boldsymbol{A}$
      $j = \operatorname{argmin}_l G_{il}$
      $\boldsymbol{a}_i \leftarrow \boldsymbol{a}_i + 2/(t+2) \cdot (\boldsymbol{e}_j - \boldsymbol{a}_i)$
    $t \leftarrow t + 1$
  until updates "become small" or $t = t_{\max}$

3 Rapid Archetypal Analysis

Our main observation w.r.t. the problem of efficient archetypal analysis is that the columns $\boldsymbol{a}_i$ of $\boldsymbol{A}$ and the columns $\boldsymbol{b}_j$ of $\boldsymbol{B}$ reside in the standard simplices Δ^{k-1} and Δ^{n-1}, respectively. In other words, the columns of either factor matrix are elements of a convex set. Furthermore, if the factor matrices are determined in an alternating manner, that is in a manner where we assume $\boldsymbol{Z} = \boldsymbol{X}\boldsymbol{B}$ to be given in order to update our current estimate of $\boldsymbol{A}$ and then fix $\boldsymbol{A}$ to update our current estimate of $\boldsymbol{B}$, the objective function in (3) becomes convex in either $\boldsymbol{A}$ or $\boldsymbol{B}$. This, however, is to say that both update steps constitute a convex minimization problem over a convex set and can thus be tackled using the efficient Frank-Wolfe procedure [12].

Our idea is thus to refrain from using elaborate quadratic programming but to harness the efficiency of computing gradients $\nabla_{\boldsymbol{A}} E$ and $\nabla_{\boldsymbol{B}} E$ while avoiding

Algorithm 2 greedy Frank-Wolfe procedure to compute matrix $\boldsymbol{B}$ whose columns reside in the simplex Δ^{n-1}

Require: data matrix $\boldsymbol{X}$, matrix of coefficients $\boldsymbol{A}$, and parameter $t_{\max} \in \mathbb{N}$
  $\boldsymbol{B} \leftarrow [\boldsymbol{e}_1, \boldsymbol{e}_1, \ldots, \boldsymbol{e}_1]$ where $\boldsymbol{e}_1 = [1, 0, \ldots, 0]^T \in \mathbb{R}^N$ // initialize $n \times k$ matrix $\boldsymbol{B}$
  $t \leftarrow 0$
  repeat
    $\boldsymbol{G} = \nabla_{\boldsymbol{B}} E = 2\left[\boldsymbol{X}^T\boldsymbol{X}\boldsymbol{B}\boldsymbol{A}\boldsymbol{A}^T - \boldsymbol{X}^T\boldsymbol{X}\boldsymbol{A}^T\right]$ // compute gradient matrix
    for $j \in \{1, \ldots, k\}$ **do** // update columns $\boldsymbol{b}_j$ of $\boldsymbol{B}$
      $i = \operatorname{argmin}_l G_{jl}$
      $\boldsymbol{b}_j \leftarrow \boldsymbol{b}_j + 2/(t+2) \cdot (\boldsymbol{e}_i - \boldsymbol{b}_j)$
    $t \leftarrow t + 1$
  until updates "become small" or $t = t_{\max}$

costly back projections into the feasible set. This can be accomplished if subgradient updates are performed along affine directions $\boldsymbol{e}_j - \boldsymbol{a}_i$ and $\boldsymbol{e}_i - \boldsymbol{b}_j$ within the simplices Δ^{k-1} and Δ^{n-1}, respectively. In a nutshell, this idea leads to the update algorithms 1 and 2 which are variants of a recent algorithm by Clarkson [7] which itself is a variant of the celebrated Frank-Wolfe procedure. A detailed analysis of this algorithm is beyond the scope of this paper but we point out that it quickly achieves ϵ-approximations of the optimal solution that are provably sparse. For a recent excellent survey of projection-free convex optimization, we refer to [17].

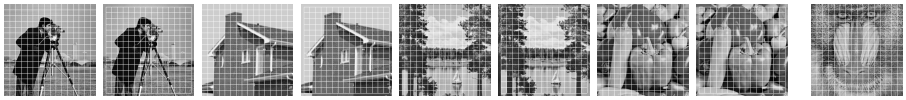
In extensive runtime evaluations (whose details we omit due to lack of space), we examined the behavior of this new algorithm under various choices of the number n of data, the dimensionality m of data, and the number k of archetypes to be determined and found our approach to be two to three orders of magnitude faster than the previous methods in [3,20].

4 Application: Archetypal Analysis as an Autoencoder

In this section, we consider a practical application of archetypal analysis in the context of feature learning. Given that our new algorithm is much faster than previous methods, it now appears practical to apply archetypal analysis not only to sets of images [25] but to considerably larger sets of image patches.

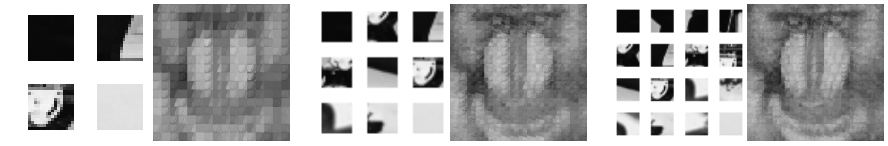
Our application example is motivated by the observation that neural networks are back with a vengeance! Owing to the recent success of deep learning architectures in computer vision, speech recognition, or automatic translation [9,13,14,15,18,27,28,29], research in these fields currently undergoes a paradigm shift. At the heart of many deep learning architectures especially for image analysis is the idea of using autoencoders for feature learning.

Looking at the overall objective of archetypal analysis, namely to find factor matrices such that $\boldsymbol{X} \approx \boldsymbol{X}\boldsymbol{B}\boldsymbol{A}$, we realize that it is indeed an autoencoder that maps $\boldsymbol{X}$ onto itself and the preliminary experiments in this section are intended to fathom the potential of archetypal analysis for joint feature learning.

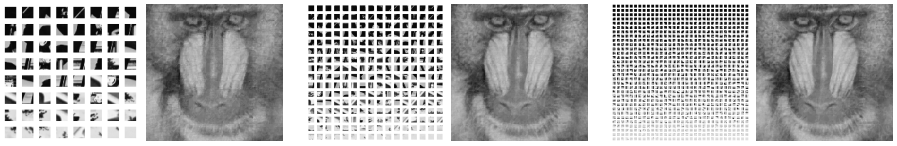


(a) training (b) test

Fig. 3: Image patches (16×16 pixels) for archetypal autoencoding experiments.



(a) $k = 4$ archetypes (b) $k = 9$ archetypes (c) $k = 16$ archetypes



(d) $k = 64$ archetypes (e) $k = 256$ archetypes (f) $k = 1024$ archetypes

Fig. 4: Results of archetypal autoencoding using a growing number of archetypes. Note that the images on the left of each panel visualize archetypal image patches of size 16×16 in each case.

Figure 3(a) shows four images from which we extracted a total of 3844 patches of size 16×16 pixels which we represent in terms of data vectors $\boldsymbol{x}_i \in \mathbb{R}^{256}$. Greedy archetype computation on this data set runs in mere fractions of a second and, for $k \in \{4, 9, 16, 64, 256, 1024\}$ it determines the archetypes shown to the left of each panel in Fig. 4. Interestingly, especially for growing k, archetypes appear to be natural realizations (i.e. actual parts of images) of edge filters and the propensity of autoencoders to learn edge features is sometimes heralded as a special characteristic of deep learning approaches [18,27]. Seen from the point of view of archetypal analysis, however, archetypal image patches are extreme in that they consist of very dark and very bright pixels, a condition typical for edges. This has already been noted in [3] and confirms early observations on neural feature learning [4].

Figure 3(b) shows a test image which we also subdivided into patches of size 16×16 and tried to reconstruct in terms of the archetypes determined from the training images. The corresponding results can be seen on the right of each panel in Fig. 4. As one would expect, for a growing number of archetypes, the reconstructions become better and we note that the reconstruction in Fig. 4(f) is actually an instance of sparse coding where the number of archetypes (1024) far exceeds the dimension of the data (256). Still, our algorithm did compute

this reconstruction in less than a second. Overall, these results suggest that archetypal analysis allows for joint feature learning for image representation. Archetypes were determined on training images independent from the test image, indicate edges like structures, and allow for reasonable reconstructions of the test image. Given the favorable runtime characteristics of the algorithm proposed in this paper, these results therefore point at new directions for feature learning.

5 Conclusion

We addressed efficient archetypal analysis and proposed an approach based on sub-gradient computations inspired by the Frank-Wolfe algorithm.

As a practical application of our novel algorithm, we considered the use of archetypal analysis as an autoencoder for joint image feature learning. Archetypes were determined from a set of training images, were observed to represent edges or contours, and allowed for convincing reconstructions of test images. Running on the CPU of a single computer rather than on graphics hardware, our algorithm processed thousands of images patches in less than a second. Our results thus hint at possible applications of archetypal analysis in machine learning. In ongoing work, we are currently exploring the use of hierarchies of archetypal autoencoders where features learned on a lower level of the hierarchy are combined and form the input for the autoencoder on the next level so as to learn semantic representations of image content that are similar in spirit to those obtained from deep learning architectures.

References

1. Aloise, D., Deshapande, A., Hansen, P., Popat, P.: NP-Hardness of Euclidean Sum-of-Squares Clustering. Machine Learning 75(2), 245–248 (2009)
2. Asbach, M., Mauruschat, D., Plinke, B.: Understanding Multi-spectral Images of Wood Particles with Matrix Factorization. In: OCM. pp. 191–202. KIT Scientific Publishing, Karlsruhe (2013)
3. Bauckhage, C., Thurau, C.: Making Archetypal Analysis Practical. In: Denzler, J., Notni, G. (eds.) DAGM. LNCS, vol. 5748, pp. 272–281. Springer, Heidelberg (2009)
4. Bell, A., Sejnowski, T.: The Indepoendent Components of Natural Images are Edge Filters. Vision Research 37(23), 3327–3338 (1997)
5. Chan, B., Mitchell, D., Cram, L.: Archetypal Analysis of Galaxy Spectra. Monthly Notices of the Royal Astronomical Society 338(3), 790–795 (2003)
6. Cheema, M., Eweiwi, A., Thurau, C., Bauckhage, C.: Action Recognition by Learning Discriminative Key Poses. In: ICCV (ICCV Workshops). pp. 1302–1309. IEEE Press, New York (2011)
7. Clarkson, K.: Coresets, Sparse Greedy Approximation, and the Frank-Wolfe Algorithm. ACM Trans. on Algorithms 6(4), 63:1–63:30 (2010)
8. Cutler, A., Breiman, L.: Archetypal Analysis. Technometrics 36(4), 338–347 (1994)
9. Deng, L., Hinton, G., Kingsbury, B.: New Types of Deep Neural Network Learning for Speech Recognition and Related Applications: An Overview. In: ICASSP. pp. 8599–8603. IEEE Press, New York (2013)

10. Eugster, M., Leisch, F.: Weighted and Robust Archetypal Analysis. Computational Statistics & Data Analysis 55(3), 1215–1225 (2011)
11. Faloutsos, C., Lin, K.I.: FastMap: A Fast Algorithm for Indexing, Data-mining and Visualization of Traditional and Multimedia Datasets. In: Proc. SIGMOD. pp. 163–174. ACM (1995)
12. Frank, M., Wolfe, P.: An Algorithm for Quadratic Programming. Naval Research Logistics Quarterly 3(1–2), 95–110 (1956)
13. Gao, J., X. He, W.Y., Deng, L.: Learning Continuous Phrase Representations for Translation Modeling. In: Proc. ACL (2014)
14. Hannun, A., Case, C., Casper, J., Catanzaro, B., Diamos, G., Elsen, E., Prenger, R., Satheesh, S., Sengupta, S., Coates, A., Ng, A.: Deep Speech: Scaling up End-to-End Speech Recognition. arXiv:1412.5567 [cs.CL] (2014)
15. He, K., Zhang, X., Ren, S., Sun, J.: Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification. arXiv:1502.01852 [cs.CV] (2015)
16. Huggins, P., Pachter, L., Sturmfels, B.: Toward the Human Genotope. Bulletin of Mathematical Biology 69(8), 2723–2735 (2007)
17. Jaggi, M.: Revisiting Frank-Wolfe: Projection-Free Sparse Convex Optimization. J. of Machine Learning Research 28(1), 427–435 (2013)
18. Krizhevsky, A., Sutskever, I., Hinton, G.: Imagenet Classification with Deep Convolutional Neural Networks. In: Proc. NIPS (2012)
19. Marinetti, S., Finesso, L., Marsilio, E.: Matrix factorization methods: application to Thermal NDT/E. In: Proc. Int. Workshop Advances in Signal Processing for Non Destructive Evaluation of Materials (2005)
20. Morup, M., Hansen, L.: Archetypal Analysis for Machine Learning and Data Mining. Neurocomputing 80, 54–63 (2012)
21. Prabhakaran, S., Raman, S., Vogt, J., Roth, V.: Automatic Model Selection in Archetype Analysis. In: Pinz, A., Pock, T., Bischof, H., Leberl, F. (eds.) DAGM. LNCS, vol. 7476, pp. 458–467. Springer, Heidelberg (2012)
22. Seth, S., Eugster, M.: Probabilistic Archetypal Analysis. arXiv:1312.7604v2 [stat.ML] (2014)
23. Stone, E., Cutler, A.: Exploring Archetypal Dynamics of Pattern Formation in Cellular Flames. Physica D 161(3–4), 163–186 (2002)
24. Thogersen, J., Morup, M., Damkiaer, S., Molin, S., Jelsbak, L.: Archetypal Analysis of Diverse Pseudomonas Aeruginosa Transcriptomes Reveals Adaptation in Cystic Fibrosis Airways. BMC Bioinformatics 14(1), 279 (2013)
25. Thurau, C., Bauckhage, C.: Archetypal Images in Large Photo Collections. In: ICSC. pp. 129–136. IEEE Press, New York (2009)
26. Thurau, C., Kersting, K., Wahabzada, M., Bauckhage, C.: Convex Non-negative Matrix Factorization for Massive Datasets. Knowledge and Information Systems 29(2), 457–478 (2011)
27. Tompson, J., Goroshin, R., Jain, A., LeCun, Y., Bregler, C.: Efficient Object Localization Using Convolutional Networks. In: CVPR. IEEE Press, New York (2015)
28. Vinyals, O., Toshev, A., Bengio, S., Erhan, D.: Show and Tell: A Neural Image Caption Generator. arXiv:1411.4555 [cs.CV] (2014)
29. Wiesler, S., Richard, A., Schlüter, R., Ney, H.: Mean-Normalized Stochastic Gradient for Large-Scale Deep Learning. In: ICASSP. pp. 180–184. IEEE Press, New York (2014)
30. Xiong, Y., Liu, W., Zhao, D., Tang, X.: Face Recognition via Archetypal Hull Ranking. In: ICCV. pp. 585–592. IEEE Press, New York (2013)