The Journal of Open Source Software

# ExpFamilyPCA.jl: A Julia Package for Exponential Family Principal Component Analysis

**Logan Mondal Bhamidipaty** [1], **Mykel J. Kochenderfer** [1], **and Trevor Hastie** [1]

**1** Stanford University

**DOI:** 10.21105/joss.07403

**Software**

- Review
- Repository
- Archive

**Editor:** Oskar Laverny

**Reviewers:**

- @ManuelStapper
- @gdalle
- @dufourc1

**Submitted:** 12 October 2024
**Published:** 10 January 2025

## Summary

Principal component analysis (PCA) (Hotelling, 1933; Jolliffe, 2002; Pearson, 1901) is popular for compressing, denoising, and interpreting high-dimensional data, but it underperforms on binary, count, and compositional data because the objective assumes data is normally distributed. Exponential family PCA (EPCA) (Collins et al., 2001) generalizes PCA to accommodate data from any exponential family distribution, making it more suitable for fields where these data types are common, such as geochemistry, marketing, genomics, political science, and machine learning (Greenacre, 2021; Hastie et al., 2009).

`ExpFamilyPCA.jl` is a library for EPCA written in Julia, a dynamic language for scientific computing (Bezanson et al., 2017). It is the first EPCA package in Julia and the first in any language to support EPCA for multiple distributions.

## Statement of Need

EPCA is used in reinforcement learning (Roy et al., 2005), sample debiasing (R. Huang & Lee, 2023), and compositional analysis (Gan & Valdez, 2024). Wider adoption, however, remains limited due to the lack of implementations. The only other EPCA package is written in MATLAB and supports just one distribution (Chambrier, 2016). This is surprising, as other Bregman-based optimization techniques have been successful in areas like mass spectrometry (Nozaki & Nakamoto, 2017), ultrasound denoising (J. Huang & Yang, 2013), topological data analysis (Edelsbrunner & Wagner, 2019), and robust clustering (Banerjee et al., 2005). These successes suggest that EPCA holds untapped potential in signal processing and machine learning.

The absence of a general EPCA library likely stems from the limited interoperability between fast symbolic differentiation and optimization libraries in popular languages like Python and C. Julia, by contrast, uses multiple dispatch which promotes high levels of generic code reuse (Karpinski, 2019). Multiple dispatch allows `ExpFamilyPCA.jl` to integrate fast symbolic differentiation (Gowda et al., 2022), optimization (Mogensen & Riseth, 2018), and numerically stable computation (Mächler, 2015) without requiring costly API conversions.[1] As a result, `ExpFamilyPCA.jl` delivers speed, stability, and flexibility, with built-in support for most common distributions (§ Supported Distributions) and flexible constructors for custom distributions (§ Custom Distributions).

[1]Symbolic differentiation is essential for flexibly specifying the EPCA objective (see documentation). While algorithmic differentiation is faster in general, symbolic differentiation is performed only once to generate a closed form for the optimizer (e.g., `Optim.jl` (Mogensen & Riseth, 2018)), making it more efficient here. *LogExpFunctions.jl* (2024) (which implements ideas from Mächler (2015)) mitigates overflow and underflow in exponential and logarithmic operations.

## Principal Component Analysis

### Geometric Interpretation

Given a data matrix $X \in \mathbb{R}^{n \times d}$ with $n$ observations and $d$ features, PCA seeks the closest low-rank approximation $\Theta \in \mathbb{R}^{n \times d}$ by minimizing the reconstruction error

$$\begin{aligned} \underset{\Theta}{\text{minimize}} \quad & \frac{1}{2}\|X - \Theta\|_F^2 \\ \text{subject to} \quad & \operatorname{rank}(\Theta) = k \end{aligned}$$

where $\|\cdot\|_F$ denotes the Frobenius norm. The optimal $\Theta$ is a $k$-dimensional linear subspace that can be written as the product of the projected observations $A \in \mathbb{R}^{n \times k}$ and the basis $V \in \mathbb{R}^{k \times d}$:

$$X \approx \Theta = AV.$$

This suggests that each observation $x_i \in \operatorname{rows}(X)$ can be well-approximated by a linear combination of $k$ basis vectors (the rows of $V$):

$$x_i \approx \theta_i = a_i V$$

for $i = 1, \ldots, n$.

### Probabilistic Interpretation

The PCA objective is equivalent to maximum likelihood estimation for a Gaussian model. Under this lens, each observation $x_i$ is a noisy realization of a $d$-dimensional Gaussian at $\theta_i \in \operatorname{rows}(\Theta)$:

$$x_i \sim \mathcal{N}(\theta_i, I).$$

To recover the latent structure $\Theta$, PCA solves

$$\begin{aligned} \underset{\Theta}{\text{maximize}} \quad & \sum_{i=1}^{n} \log \mathcal{L}(x_i; \theta_i) \\ \text{subject to} \quad & \operatorname{rank}(\Theta) = k \end{aligned}$$

where $\mathcal{L}$ is the likelihood function.

## Exponential Family PCA

### Exponential Family

Following Forster & Warmuth (2002), we define the exponential family as the set of distributions with densities of the form

$$p_\theta(x) = \exp(\theta \cdot x - G(\theta))$$

where $\theta$ is the natural parameter and $G$ is the log-partition function.

### Link Function

The link function $g(\theta)$ connects the natural parameter $\theta$ to the mean parameter $\mu$ of an exponential family distribution. It is defined as the gradient of the log-partition function $G(\theta)$:

$$\mu = g(\theta) = \nabla G(\theta).$$

The link function serves a role analogous to that in generalized linear models (GLMs) (McCullagh & Nelder, 1989). In GLMs, the link function connects the linear predictor to the mean of the distribution, enabling flexibility in modeling various data types. Similarly, in EPCA, the link function maps the low-dimensional latent variables to the expectation parameters of the exponential family, thereby generalizing the linear assumptions of traditional PCA to accommodate diverse distributions (see appendix).

### Bregman Divergences

EPCA extends the probabilistic interpretation of PCA using a measure of statistical difference called the Bregman divergence (Bregman, 1967; Efron, 2004). The Bregman divergence $B_F$ for a strictly convex, continuously differentiable function $F$ is

$$B_F(p\|q) = F(p) - F(q) - \langle \nabla F(q), p - q \rangle.$$

This can be interpreted as the difference between $F(p)$ and its linear approximation about $q$. When $F$ is the convex conjugate of the log-partition function of an exponential family distribution, minimizing the Bregman divergence corresponds to maximizing the associated log-likelihood (Azoury & Warmuth, 2001; Forster & Warmuth, 2002) (see documentation).

### Loss Function

EPCA generalizes the PCA objective as a Bregman divergence between the data $X$ and the expectation parameters $g(\Theta)$:

$$\begin{aligned} \underset{\Theta}{\text{minimize}} \quad & B_F(X\|g(\Theta)) \\ \text{subject to} \quad & \mathrm{rank}\left(\Theta\right) = k \end{aligned}$$

where

- $g(\theta)$ is the **link function** and the gradient of $G$,
- $G(\theta)$ is a strictly convex, continuously differentiable function (usually the **log-partition** of an exponential family distribution),
- and $F(\mu)$ is the **convex conjugate** of $G$ defined by

$$F(\mu) = \max_{\theta} \langle \mu, \theta \rangle - G(\theta).$$

This suggests that data from the exponential family is well-approximated by expectation parameters

$$x_i \approx g(\theta_i) = g(a_i V).$$

### Regularization

Following Collins et al. (2001), we introduce a regularization term to ensure the optimum converges

$$\begin{aligned} \underset{\Theta}{\text{minimize}} \quad & B_F(X \| g(\Theta)) + \epsilon B_F(\mu_0 \| g(\Theta)) \\ \text{subject to} \quad & \operatorname{rank}(\Theta) = k \end{aligned}$$

where $\epsilon > 0$ and $\mu_0 \in \mathrm{range}(g)$.[2]

### Example: Poisson EPCA

The Poisson EPCA objective is the generalized Kullback-Leibler (KL) divergence (see appendix), making Poisson EPCA ideal for compressing discrete distribution data.

This is useful in applications like belief compression in reinforcement learning (Roy et al., 2005), where high-dimensional belief states can be effectively reduced with minimal information loss. Below we recreate similar figures[3] to Roy & Gordon (2002) and Roy et al. (2005) and observe that Poisson EPCA almost perfectly reconstructs a $41$-dimensional belief distribution using just $5$ basis components. For a larger environment with $200$ states, PCA struggles even with $10$ basis components.

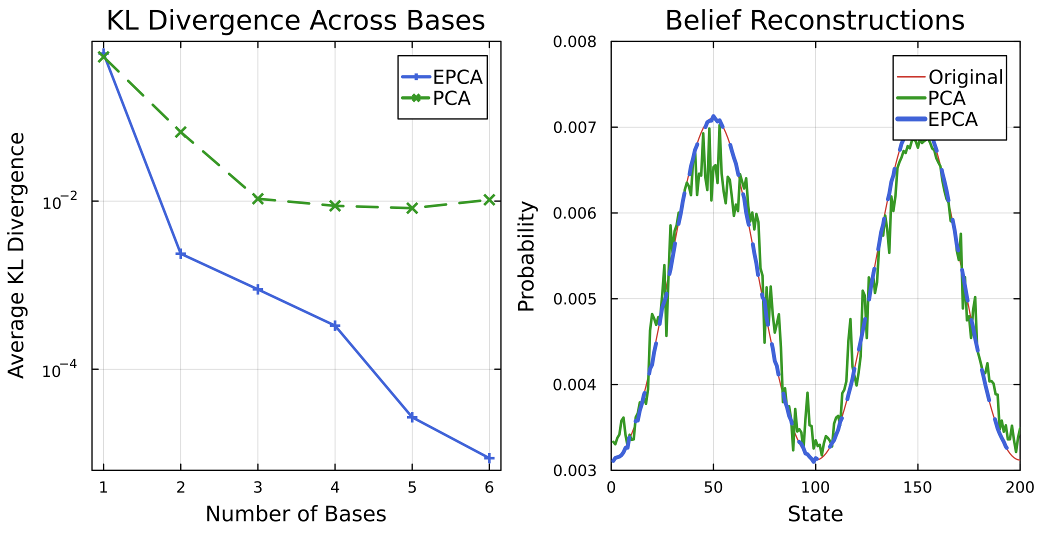

**Figure 1:** Left - KL Divergence for Poisson EPCA versus PCA. Right - Reconstructions from the models.

## API

### Supported Distributions

`ExpFamilyPCA.jl` includes efficient EPCA implementations for several exponential family distributions.

| Julia | Description |
|---|---|
| `BernoulliEPCA` | For binary data |

[2]In practice, we allow $\epsilon \geq 0$, because special cases of EPCA like traditional PCA are well-known to converge without regularization. Similarly, we pick $\mu_0$ to simplify terms in the objective.

[3]See Figure 3(a) in Roy & Gordon (2002) and Figure 12(c) in Roy et al. (2005).

| Julia | Description |
|---|---|
| `BinomialEPCA` | For count data with a fixed number of trials |
| `ContinuousBernoulliEPCA` | For modeling probabilities between $0$ and $1$ |
| `GammaEPCA` | For positive continuous data |
| `GaussianEPCA` | Standard PCA for real-valued data |
| `NegativeBinomialEPCA` | For over-dispersed count data |
| `ParetoEPCA` | For modeling heavy-tailed distributions |
| `PoissonEPCA` | For count and discrete distribution data |
| `WeibullEPCA` | For modeling life data and survival analysis |

### Custom Distributions

When working with custom distributions, certain specifications are often more convenient and computationally efficient than others. For example, inducing the gamma EPCA objective from the log-partition $G(\theta) = -\log(-\theta)$ and its derivative $g(\theta) = -1/\theta$ is much simpler than implementing the full the Itakura-Saito distance (Itakura & Saito, 1968) (see appendix):

$$D(P(\omega), \hat{P}(\omega)) = \frac{1}{2\pi} \int_{-\pi}^{\pi} \left[ \frac{P(\omega)}{\hat{P}(\omega)} - \log \frac{P(\omega)}{\hat{P}\omega} - 1 \right] d\omega.$$

In `ExpFamilyPCA.jl`, we would write:

```julia
G(θ) = -log(-θ)
g(θ) = -1 / θ
gamma_epca = EPCA(indim, outdim, G, g, Val((:G, :g)); options = NegativeDomain())
```

A lengthier discussion of the `EPCA` constructors and math is provided in the documentation.

### Usage

Each `EPCA` object supports a three-method interface: `fit!`, `compress`, and `decompress`. `fit!` trains the model and returns the compressed training data; `compress` returns compressed input; and `decompress` reconstructs the original data from the compressed representation.

```julia
X = sample_from_gamma(n1, indim)  # matrix of gamma-distributed data
Y = sample_from_gamma(n2, indim)

X_compressed = fit!(gamma_epca, X)
Y_compressed = compress(gamma_epca, Y)
Y_reconstructed = decompress(gamma_epca, Y_compressed)
```

## Acknowledgments

We thank Ryan Tibshirani, Arec Jamgochian, Robert Moss, and Dylan Asmar for their help and guidance.

## References

Azoury, K. S., & Warmuth, M. K. (2001). Relative loss bounds for on-line density estimation with the exponential family of distributions. *Machine Learning*, *43*, 211–246. https://doi.org/10.1023/A:1010896012157

Banerjee, A., Merugu, S., Dhillon, I. S., & Ghosh, J. (2005). Clustering with Bregman divergences. *Journal of Machine Learning Research*, *6*(58), 1705–1749. http://jmlr.org/papers/v6/banerjee05b.html

Bezanson, J., Edelman, A., Karpinski, S., & Shah, V. B. (2017). Julia: A fresh approach to numerical computing. *SIAM Review*, *59*(1), 65–98. https://doi.org/10.1137/141000671

Bregman, L. M. (1967). The relaxation method of finding the common point of convex sets and its application to the solution of problems in convex programming. *USSR Computational Mathematics and Mathematical Physics*, *7*(3), 200–217. https://doi.org/10.1016/0041-5553(67)90040-7

Chambrier, G. de. (2016). *E-PCA*. https://github.com/gpldecha/e-pca

Collins, M., Dasgupta, S., & Schapire, R. E. (2001). A generalization of principal components analysis to the exponential family. *Advances in Neural Information Processing Systems*, *14*. https://doi.org/10.7551/mitpress/1120.003.0084

Edelsbrunner, H., & Wagner, H. (2019). Topological data analysis with Bregman divergences. *Journal of Computational Geometry*, Vol. 9 No. 2 (2018): Special Issue of Selected Papers from SoCG 2017. https://doi.org/10.20382/JOCG.V9I2A6

Efron, B. (2004). The estimation of prediction error. *Journal of the American Statistical Association*, *99*(467), 619–632. https://doi.org/10.1198/016214504000000692

Forster, J., & Warmuth, M. K. (2002). Relative expected instantaneous loss bounds. *Journal of Computer and System Sciences*, *64*(1), 76–102. https://doi.org/10.1006/jcss.2001.1798

Gan, G., & Valdez, E. A. (2024). Compositional Data Regression in Insurance with Exponential Family PCA. *Variance*, *17*(1).

Gowda, S., Ma, Y., Cheli, A., Gwóźzdź, M., Shah, V. B., Edelman, A., & Rackauckas, C. (2022). High-performance symbolic-numerics via multiple dispatch. *Association for Computing Machinery Communications in Computer Algebra*, *55*(3), 92–96. https://doi.org/10.1145/3511528.3511535

Greenacre, M. (2021). Compositional data analysis. *Annual Review of Statistics and Its Application*, *8*(1), 271–299.

Hastie, T., Tibshirani, R., Friedman, J. H., & Friedman, J. H. (2009). *The Elements of Statistical Learning: Data Mining, Inference, and Prediction* (Vol. 2). Springer. https://doi.org/10.1007/978-0-387-84858-7

Hotelling, H. (1933). Analysis of a complex of statistical variables into principal components. *Journal of Educational Psychology*, *24*, 498–520. https://doi.org/10.1037/h0071325

Huang, J., & Yang, X. (2013). Fast reduction of speckle noise in real ultrasound images. *Signal Processing*, *93*(4), 684–694. https://doi.org/10.1016/j.sigpro.2012.09.005

Huang, R., & Lee, Y. (2023). *Debiasing sample loadings and scores in exponential family PCA for sparse count data*. https://arxiv.org/abs/2312.13430

Itakura, F., & Saito, S. (1968). Analysis synthesis telephony based on the maximum likelihood method. *Proceedings of the 6th International Congress on Acoustics*, C-17-C-20.

Jolliffe, I. T. (2002). *Principal component analysis for special types of data*. Springer. https://doi.org/10.1007/0-387-22440-8_13

Karpinski, S. (2019). *The unreasonable effectiveness of multiple dispatch*. Conference Talk at JuliaCon 2019, available at https://www.youtube.com/watch?v=kc9HwsxE1OY.

*LogExpFunctions.jl*. (2024). GitHub. https://github.com/JuliaStats/LogExpFunctions.jl

Mächler, M. (2015). *Accurately computing* $\log(1-\exp(-|a|))$ *assessed by the 'Rmpfr' package*.

https://doi.org/10.13140/RG.2.2.11834.70084

McCullagh, P., & Nelder, J. A. (1989). *Generalized Linear Models* (2nd ed.). Chapman & Hall/CRC. https://doi.org/10.1201/9780203753736

Mogensen, P. K., & Riseth, A. N. (2018). Optim: A mathematical optimization package for Julia. *Journal of Open Source Software*, *3*(24), 615. https://doi.org/10.21105/joss.00615

Nozaki, Y., & Nakamoto, T. (2017). Itakura-Saito distance based autoencoder for dimensionality reduction of mass spectra. *Chemometrics and Intelligent Laboratory Systems*, *167*, 63–68. https://doi.org/10.1016/j.chemolab.2017.05.002

Pearson, K. (1901). On lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, *2*(11), 559–572. https://doi.org/10.1080/14786440109462720

Roy, N., & Gordon, G. (2002). Exponential family PCA for belief compression in POMDPs. In S. Becker, S. Thrun, & K. Obermayer (Eds.), *Advances in Neural Information Processing Systems* (Vol. 15). MIT Press. https://proceedings.neurips.cc/paper_files/paper/2002/file/a11f9e533f28593768ebf87075ab34f2-Paper.pdf

Roy, N., Gordon, G., & Thrun, S. (2005). Finding approximate POMDP solutions through belief compression. *Journal of Artificial Intelligence Research*, *23*, 1–40. https://doi.org/10.1613/jair.1496