A Non-Convex Method for Polynomial Manifold Learning

Param Mody PARAMMODY@GMAIL.COM
Elina Robeva EROBEVA@MATH.UBC.CA

Abstract

We study fitting low-dimensional polynomial manifolds to data by alternating two simple steps: a closed-form least-squares update of an analytic polynomial decoder and Gauss–Newton updates of the latent codes using the decoder's analytic Jacobian. For intrinsic dimension $k{=}1$, the encoder reduces to enumerating stationary points of a univariate polynomial objective, and for $k{>}1$, each example solves only a small damped $k \times k$ system. The method is scalable because each B-step solves a least squares on the monomial design, and the A-step entails a handful of tiny solves per point. We test our method on various datasets.

1. Introduction

Dimensionality-reduction methods formalise this by trying to find an information–preserving map (encoder) $f : \mathbb{R}^d \to \mathbb{R}^k,\ k \ll d$, together with an approximate inverse (decoder) $g : \mathbb{R}^k \to \mathbb{R}^d$ such that the reconstruction error $\sum_{i=1}^n \|\mathbf{x}_i - g(f(\mathbf{x}_i))\|^2$ is minimised for data $X = \big[\mathbf{x}_1, \dots, \mathbf{x}_n\big]^\top \in \mathbb{R}^{n\times d}$. Principal Component Analysis (PCA) provides an optimal linear solution, but real-world data is usually closer to nonlinear manifolds. We propose a novel algorithm that finds the best fitting polynomial parametrisation of some intrinsic dimension k by doing an alternating least-squares refinement. We assume that the observations are points in the image of a map

$$g : \mathbb{R}^k \to \mathbb{R}^d, \quad g(\boldsymbol{\alpha}) = \begin{bmatrix} g_1(\alpha_1, \alpha_2, \dots, \alpha_k) \\ g_2(\alpha_1, \alpha_2, \dots, \alpha_k) \\ \vdots \\ g_d(\alpha_1, \alpha_2, \dots, \alpha_k) \end{bmatrix},$$

where $g_1, \dots, g_d$ are multivariate polynomials of bounded degree. Our algorithm alternates between finding the best coefficients of $g_1, \dots, g_d$ and finding the best $\boldsymbol{\alpha}^{(j)} \in \mathbb{R}^k$ such that $g(\boldsymbol{\alpha}^{(j)})$ is closest to the jth data point $\mathbf{x}_j$. By iteratively refining both the global polynomial mapping and the individual latent embeddings, the method progressively decreases the reconstruction error. We demonstrate the effectiveness of this approach on several synthetic manifolds. In our experiments, we compare the proposed method against classic and modern alternatives, including nonlinear autoencoders and Principal Polynomial Analysis, to highlight its advantages and disadvantages.

Notation.

1. Latent codes: $A = [\boldsymbol{\alpha}_1^\top, \dots, \boldsymbol{\alpha}_n^\top]^\top \in \mathbb{R}^{n\times k}$.

2. Polynomial feature map: $\Phi_\gamma : \mathbb{R}^k \to \mathbb{R}^m$ is the vector of all monomials of total degree $\leq \gamma$; its Jacobian is $\frac{\partial \Phi_\gamma(\boldsymbol{\alpha})}{\partial \boldsymbol{\alpha}} \in \mathbb{R}^{m \times k}$.
3. Design matrix: $V(A) \in \mathbb{R}^{n \times m}$ with ith row $\Phi_\gamma(\boldsymbol{\alpha}_i)^\top$.
4. The decoder parameters are $W \in \mathbb{R}^{d \times m}$ (the coefficients of the polynomials), the decoder function is $g(\boldsymbol{\alpha}) = W \Phi_\gamma(\boldsymbol{\alpha})$ and reconstructions are $\hat{X} = V(A)W^\top$.

2. Related Work

Principal Polynomial Analysis (PPA) constructs a sequential transform by projecting onto a leading direction, fitting a univariate polynomial in that scalar to predict the residual in the orthogonal complement, and deflating and repeating. The mapping is volume-preserving, admits a closed-form inverse, and has an analytic Jacobian [3]. Our method differs in that we jointly refine all samples via alternating minimisation on a single global objective, rather than stagewise deflation, our B-step is a single overdetermined least-squares fit of an explicit polynomial decoder W, and our A-step uses Gauss–Newton updates in $\mathbb{R}^k$ (for $k > 1$) or stationary-point enumeration (for $k = 1$) with an analytic Jacobian of the monomial map. We do not have the same volume-preservation or closed-form inverse but our method is more robust to non-smooth data.

Kernel PCA [6] is a nonlinear dimensionality reduction method that uses a kernel function to map data into a higher dimensional space and then applies PCA in that space. The idea is that the features become more linearly separable in higher dimensional space and applying PCA in that space helps capture nonlinear relations. It incurs $\mathcal{O}(n^3)$ training unless approximated. In contrast, our decoder is explicit and our method is faster.

Autoencoders. Neural autoencoders [2] are powerful and scalable but non-convex and not interpretable. They also require careful tuning of parameters.

Neighbourhood embeddings. Isomap [7], LLE [5], t-SNE [8] and UMAP [4] provide high-quality embeddings but no analytic inverse. Our goal is not only embeddings but a parametric decoder $g(\boldsymbol{\alpha})$ for reconstruction and analysis.

Principal curves are self-consistent smooth curves passing through the "middle" of the data [1]. They are defined implicitly via projection operators and typically require iterative smoothing and extensions to higher dimensions or to explicit, globally invertible parameterisations are non-trivial.

3. Algorithm

Given centred data $X \in \mathbb{R}^{n \times d}$ and an orthonormal frame $E_0 \in \mathbb{R}^{d \times k}$,

$$A^{(0)} = X E_0 \in \mathbb{R}^{n \times k}, \tag{1}$$

$$W^{(0)} = \arg\min_W \left\| X - V\big(A^{(0)}\big)\, W^\top \right\|_F^2, \tag{2}$$

$$A^{(t)} \leftarrow \mathrm{update}\big(A^{(t-1)}; W^{(t-1)}\big), \tag{3}$$

$$W^{(t)} = \arg\min_W \left\| X - V\big(A^{(t)}\big)\, W^\top \right\|_F^2, \tag{4}$$

$$\hat{X} = V\big(A^{(T)}\big)\, W^{(T)\top}. \tag{5}$$

B-step. Given latent codes $A \in \mathbb{R}^{n\times k}$ and the polynomial feature map $\Phi_\gamma : \mathbb{R}^k \to \mathbb{R}^m$ (all monomials up to total degree γ), the design matrix is $V(A) \in \mathbb{R}^{n\times m}$ with ith row $\Phi_\gamma(\boldsymbol{\alpha}_i)^\top$. The decoder $W \in \mathbb{R}^{d\times m}$ is re-estimated at each outer loop by ordinary least squares:

$$\min_W \; \| X - V(A)\, W^\top \|_F^2,$$

which we solve via a least-squares call (`np.linalg.lstsq`).

A-step for $k = 1$. For a single latent $\alpha \in \mathbb{R}$, let $g(\alpha) = W\,\Phi_\gamma(\alpha)$ and, for sample i, define component polynomials $p_j(\alpha) = g_j(\alpha) - x_{ij}$, $j = 1, \ldots, d$. The scalar objective

$$\ell_i(\alpha) = \sum_{j=1}^{d} p_j(\alpha)^2$$

is a polynomial of degree $\le 2\gamma$. We enumerate candidates as the real roots of the derivative $\ell_i'(\alpha)$ (degree $\le 2\gamma - 1$) together with the previous iterate, and select the α that minimises ℓ_i. Root finding is implemented with standard NumPy polynomial methods.

A-step for $k > 1$. For $\boldsymbol{\alpha} \in \mathbb{R}^k$, the per-sample residual is

$$r(\boldsymbol{\alpha}) \;=\; W\Phi_\gamma(\boldsymbol{\alpha}) - \mathbf{x} \;\in \mathbb{R}^d,$$

with Jacobian

$$J(\boldsymbol{\alpha}) \;=\; \frac{\partial r(\boldsymbol{\alpha})}{\partial \boldsymbol{\alpha}} \;=\; W\,\frac{\partial \Phi_\gamma(\boldsymbol{\alpha})}{\partial \boldsymbol{\alpha}} \;\in \mathbb{R}^{d\times k}.$$

We take a Gauss–Newton step with fixed damping:

$$\Delta\boldsymbol{\alpha} \;=\; \left(J^\top J + 10^{-8} I_k\right)^{-1} J^\top r, \qquad \boldsymbol{\alpha} \leftarrow \boldsymbol{\alpha} - \Delta\boldsymbol{\alpha}.$$

To guarantee per–sample non-increase in loss, we accept the step $\boldsymbol{\alpha} \leftarrow \boldsymbol{\alpha} - \tau\,\Delta\boldsymbol{\alpha}$ with $\tau \in \{1, \frac{1}{2}, \frac{1}{4}, \frac{1}{8}, \frac{1}{16}, \frac{1}{32}\}$ chosen as the first value that satisfies the Armijo condition

$$\ell_i(\boldsymbol{\alpha} - \tau\,\Delta\boldsymbol{\alpha}; W) \;\le\; \ell_i(\boldsymbol{\alpha}; W) \;-\; c\,\tau\,\|J^\top r\|_2^2, \qquad c \in (0,1) \text{ fixed (we use } c = 10^{-4}).$$

If no τ in this set satisfies the condition, we take $\tau = \frac{1}{32}$. This adds at most five extra evaluations of g_W per inner iteration and enforces ℓ_i non–increase at each accepted update.
Gauss–Newton uses the first–order model

$$g(\boldsymbol{\alpha} + \Delta\boldsymbol{\alpha}) \;\approx\; g(\boldsymbol{\alpha}) + J\,\Delta\boldsymbol{\alpha},$$

and chooses $\Delta\boldsymbol{\alpha}$ by fitting the linearised residual:

$$\Delta\boldsymbol{\alpha} \;=\; \arg\min_{u\in\mathbb{R}^k} \; \| r(\boldsymbol{\alpha}) - J\,u \|_2^2 \quad\Longrightarrow\quad (J^\top J)\,\Delta\boldsymbol{\alpha} \;=\; J^\top r(\boldsymbol{\alpha}).$$

This update moves in latent directions that most reduce the error in data space, as measured by the columns of J. The matrix $J^\top J$ weights each latent direction by how strongly g changes along it; with a small damping $\lambda > 0$,

$$(J^\top J + \lambda I)\,\Delta\boldsymbol{\alpha} \;=\; J^\top r(\boldsymbol{\alpha}),$$

which stabilises the step when $J^\top J$ is ill–conditioned and acts like a small trust–region Levenberg–Marquardt step. Because the system is $k \times k$, the solve is inexpensive.

Initialisation We draw several (60, in our case) random orthonormal k-frames E (via QR), set $A = XE$, and run $S = 10$ cheap warm-start alternations (B then A). In each warm pass, the B-step re-fits W by ordinary least squares on $V(A)$; the A-step updates latents per sample (for $k=1$ by stationary-point enumeration, for $k>1$ by one damped Gauss–Newton step with $10^{-8} I_k$). We keep the frame with the lowest reconstruction MSE after these warm passes.

Complexity. Building the design $V(A)$ takes $\mathcal{O}(nm)$ time and $\mathcal{O}(nm)$ memory. A QR-based B-step on $V \in \mathbb{R}^{n\times m}$ with d outputs costs $\mathcal{O}(nm^2 + nmd + m^2 d + m^3)$. The A-step costs $\mathcal{O}(n\, c_A)$, where c_A is the per-sample A-update cost: for $k=1$, enumerating real stationary points via polynomial root finding is $\mathcal{O}(\gamma^3)$ per sample, while for $k>1$ each Gauss–Newton iteration costs $\mathcal{O}(dmk + k^3)$ (forming $J = W\, \partial\Phi/\partial\boldsymbol{\alpha}$ and solving a $k\times k$ system), and we run a small, fixed number of iterations. In usual cases with $n \gg m$ and $d \gg k$, the QR-based B-step dominates the total runtime.

4. Experiments

We evaluate the proposed alternating polynomial fitting method on several synthetic datasets where the ground-truth is known.

4.1. Datasets

We generate $n = 300$ points $\mathbf{x}^{(i)}$ from parametric curves or surfaces, then add Gaussian noise $\boldsymbol{\epsilon} \sim \mathcal{N}(0, 0.02 I_d)$. We used:

- **Cusp**: $\mathbf{x}(t) = (t^2, t^3)$, $t \in [-1, 1]$.
- **Circle**: $\mathbf{x}(t) = (\cos(t), \sin(t))$, $t \in [0, 2\pi]$.
- **Self–intersecting cubic**: $\mathbf{x}(t) = (t^3 - 3t, t^2 - 1)$, $t \in [-2, 2]$.
- **Double–intersection**: $\mathbf{x}(t) = (t(t^2-1)(t^2-9), (t^2-2)^2)$, $t \in [-3.2, 3.2]$.
- **Butterfly surface**: $\mathbf{x}(u, v) = (\sin u, \sin v, \sin u \cos v)$ with points removed where $|\sin u \sin v| < 0.3$ to create a non–trivial topology. More on dataset generation at A.

4.2. Methods Compared

We compare:

1. **PPA** — the standard principal polynomial analysis fit from a PCA initialisation.
2. **Alt(T)** — our alternating refinement for T outer loops.
3. **Autoencoder** — with architecture $d \to h \to k$ in the encoder and $k \to h \to d$ in the decoder, where the hidden width is fixed to $h = 32$ and all hidden activations are $\tanh$. We use Adam with learning rate $\eta = 10^{-2}$ for 800 epochs with full batch-size and no weight decay, dropout, or additional regularizers, and no early stopping.

4.3. Evaluation Metric

For each fit, we compute the mean–squared reconstruction error

$$\mathrm{MSE} = \frac{1}{n}\sum_{i=1}^{n}\left\|\mathbf{x}^{(i)} - \hat{\mathbf{x}}^{(i)}\right\|^2,$$

where $\hat{\mathbf{x}}^{(i)}$ is the model reconstruction from the latent representation. For 1-D curves, we also visualise the reconstructed path in the original space, ordering points by the learned latent coordinate for clarity.

4.4. Results

Table 1 reports MSEs for all datasets: Alt(T) consistently improves over PPA and autoencoders, even with data that does not have a polynomial parametrisation, with the largest gains on highly curved or singular manifolds. We use 10 warm-steps and 60 random trials in the initialisation procedure. We pick $\gamma = 3$ to have a good mix of expressivity and speed.

Table 1: Mean–squared reconstruction error for different methods ($\gamma = 3,\ k \in \{1, 2\}$).

Dataset	PPA	Alt(30)	Autoencoder
Self-intersecting cubic (Figure 1)	1.4225	$\mathbf{3.6787 \times 10^{-4}}$	1.5365×10^{-1}
Double–intersection (Figure 2)	270.09	**3.4212**	52.713
Cusp (Figure 3)	1.8407×10^{-2}	$\mathbf{3.2240 \times 10^{-4}}$	1.5368×10^{-3}
Circle (Figure 4)	4.9766×10^{-1}	$\mathbf{5.0688 \times 10^{-3}}$	5.4529×10^{-2}
Butterfly surface ($k = 2$, Figure 5)	2.3122×10^{-1}	$\mathbf{7.1427 \times 10^{-3}}$	1.8974×10^{-2}

5. Conclusion

We introduced an alternating scheme for learning polynomial parametrisations: a linear least-squares B-step for the decoder and a tiny per-sample A-step for the encoder (root enumeration for k=1, Gauss–Newton/LM for k>1). Our method also has a fully analytic decoder and Jacobian. The main limitations are the growth of monomial features with degree and latent dimension, and the non-convexity from the A-step. Future work includes orthogonal polynomial bases for conditioning, adding a regularisation term to encourage $\boldsymbol{\alpha}$ that are close to each other to also be close in function value, and introducing a blow-up strategy to higher-dimensional singular sets and automating a singularity detection scheme. It would also be interesting to compare the performance of more methods and also compare on datasets with higher k and d.

References

[1] Trevor Hastie and Werner Stuetzle. Principal curves. *Journal of the American Statistical Association*, 84(406):502–516, 1989. doi: 10.1080/01621459.1989.10478797.

[2] G. E. Hinton and R. R. Salakhutdinov. Reducing the Dimensionality of Data with Neural Networks. *Science*, 313(5786):504–507, July 2006. ISSN 0036-8075, 1095-9203. doi: 10.1126/science.1127647. URL https://www.science.org/doi/10.1126/science.1127647.

[3] Valero Laparra, Sandra Jiménez, Devis Tuia, Gustau Camps-Valls, and Jesús Malo. Principal polynomial analysis. *Int. J. Neural Syst.*, 24(7), 2014. doi: 10.1142/S0129065714400073. URL https://doi.org/10.1142/S0129065714400073.

[4] Leland McInnes, John Healy, Nathaniel Saul, and Lukas Großberger. Umap: Uniform manifold approximation and projection, 2018.

[5] Sam T. Roweis and Lawrence K. Saul. Nonlinear Dimensionality Reduction by Locally Linear Embedding. *Science*, 290(5500):2323–2326, December 2000. ISSN 0036-8075, 1095-9203. doi: 10.1126/science.290.5500.2323. URL https://www.science.org/doi/10.1126/science.290.5500.2323.

[6] Bernhard Schölkopf, Alexander J. Smola, and Klaus-Robert Müller. Nonlinear component analysis as a kernel eigenvalue problem. *Neural Comput.*, 10(5):1299–1319, 1998. doi: 10.1162/089976698300017467. URL https://doi.org/10.1162/089976698300017467.

[7] Joshua B. Tenenbaum, Vin de Silva, and John C. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290(5500):2319–2323, 2000. doi: 10.1126/science.290.5500.2319. URL https://www.science.org/doi/abs/10.1126/science.290.5500.2319.

[8] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-SNE. *Journal of Machine Learning Research*, 9:2579–2605, 2008. URL http://www.jmlr.org/papers/v9/vandermaaten08a.html.

Appendix A. Generation

Butterfly surface generation. We sample a uniform tensor grid in parameter space: $u_i, v_j \in [-\pi, \pi]$ with $u_i = \text{linspace}(-\pi, \pi; g)_i$, $v_j = \text{linspace}(-\pi, \pi; g)_j$, form $g \times g$ matrices $(U, V) = \text{meshgrid}(u, v)$, and evaluate the map

$$\mathbf{r}(u, v) = \big(\sin u,\ \sin v,\ \sin u\ \cos v\big).$$

We then remove a band to induce nontrivial topology by keeping only indices with $\big|\sin U \cdot \sin V\big| \geq 0.3$. Finally we flatten the surviving grid points into an $N \times 3$ cloud

$$X = \Big\{ \big(\sin u_i, \sin v_j, \sin u_i \cos v_j\big) \ : \ |\sin u_i \sin v_j| \geq 0.3 \Big\} \ + \ \varepsilon, \quad \varepsilon \sim \mathcal{N}\big(0, \sigma^2 I_3\big)\,,$$

where $N(300)$ is the number of unmasked grid locations and $\sigma^2(0.02)$ is the noise level used in the experiments. We set $g = 25$

Appendix B. Experiments

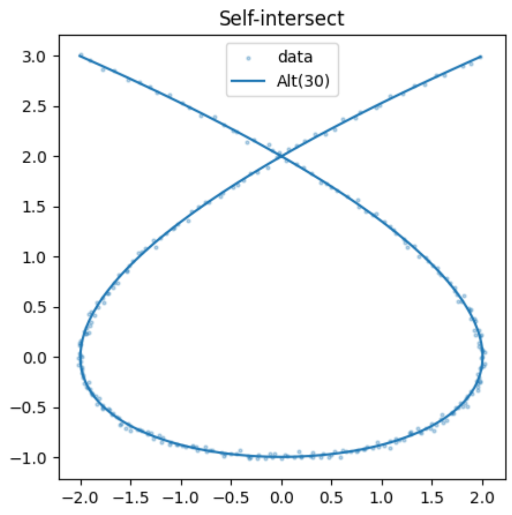

Figure 1: Self-intersect

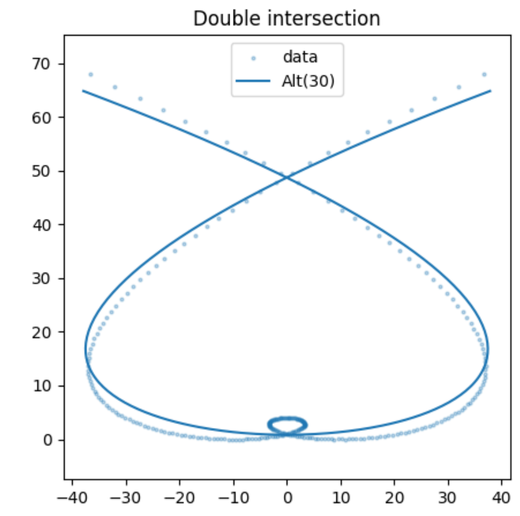

Figure 2: Double intersect

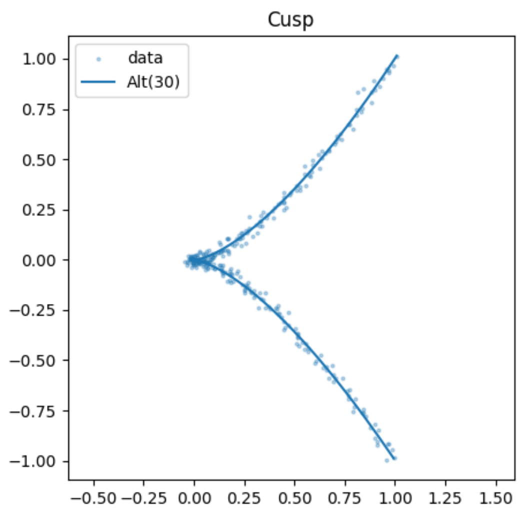

Figure 3: Cusp

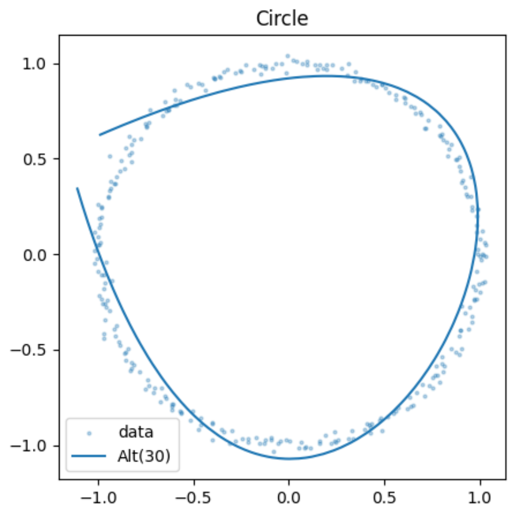

Figure 4: Circle

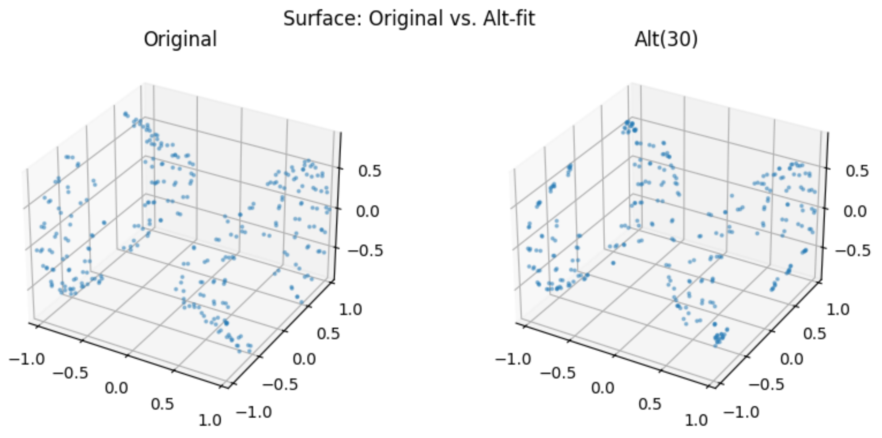

Figure 5: Butterfly