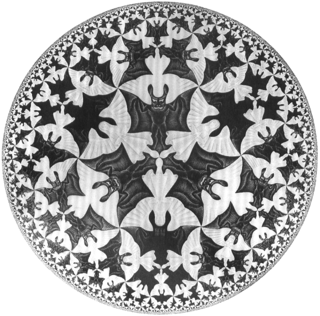
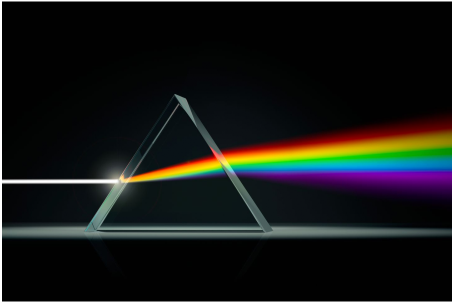
Manifold Learning and Spectral Methods



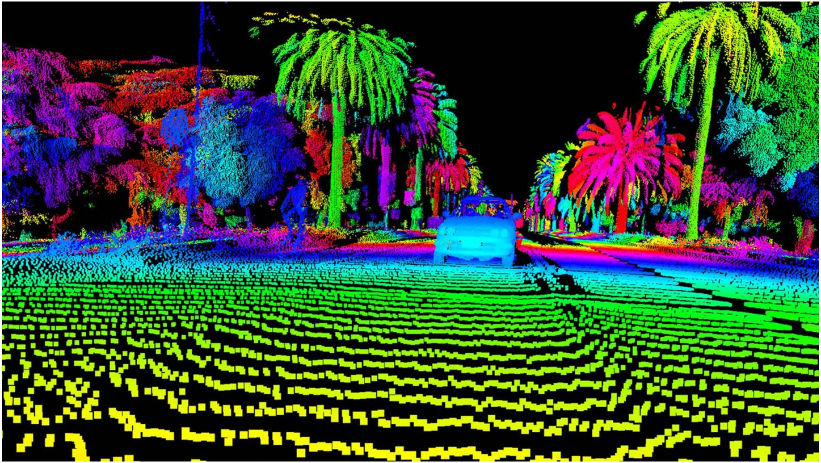


David Pfau

DeepMind

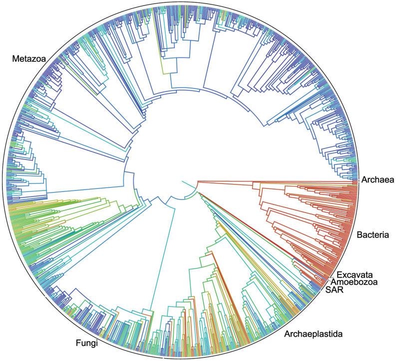
Buenos Aires MLSS, 28 June 2018

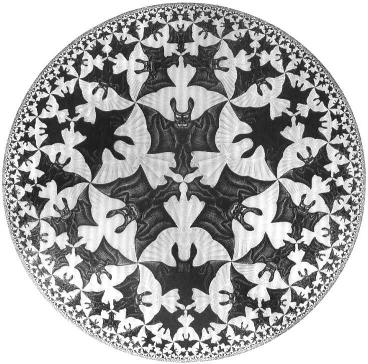
What this tutorial is about



Designing models for graph- or manifold-structured input

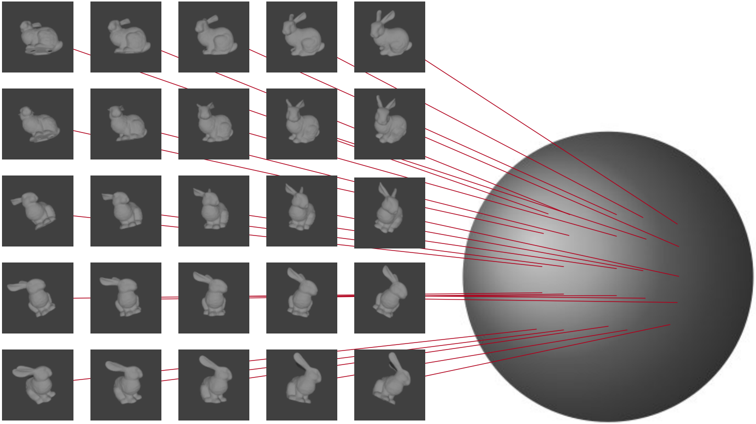
What this tutorial is about





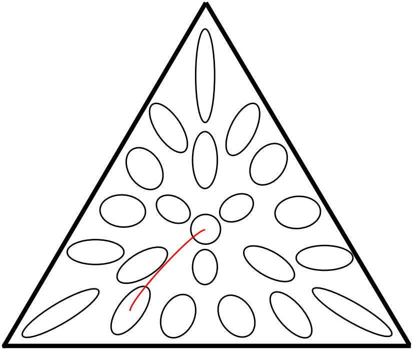
Designing models for data with latent graph- or manifold-structure

What this tutorial is about



Discovering latent manifold structure in data

What this tutorial is *not* about



$$\mathcal{F}(\theta) = \mathbb{E}_x \left[\nabla_\theta \mathrm{log} p(x|\theta) \nabla_\theta \mathrm{log} p(x|\theta)^T \right]$$

$$\theta_{t+1} \leftarrow \theta_t + \mathcal{F}^{-1} \nabla_\theta \mathrm{log} p(x_t|\theta_t)$$

Information geometry - the manifold structure of parameter space

Spectral Learning

Any learning algorithm where the usual tools (SGD, EM, MCMC...) are replaced with eigendecomposition or SVD

Spectral Learning

Any learning algorithm where the usual tools (SGD, EM, MCMC...) are replaced with eigendecomposition or SVD

Solves a nonconvex optimization exactly!

$$\mathbf{A} = \mathbf{\Phi}\mathbf{\Lambda}\mathbf{\Phi}^T$$

$$\min_{\substack{\mathbf{\Phi} \\ \mathbf{\Phi}^T\mathbf{\Phi}=\mathbf{I}}} \mathrm{Tr}(\mathbf{\Phi}^T\mathbf{A}\mathbf{\Phi})$$

Spectral Learning

Any learning algorithm where the usual tools (SGD, EM, MCMC...) are replaced with eigendecomposition or SVD

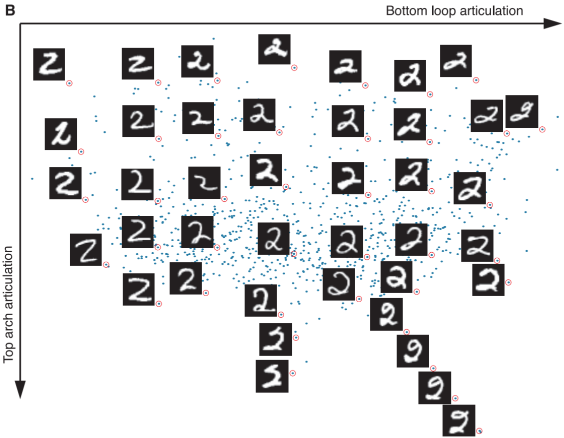
Solves a nonconvex optimization exactly!

$$\mathbf{A} = \mathbf{\Phi}\mathbf{\Lambda}\mathbf{\Phi}^T$$

$$\min_{\substack{\mathbf{\Phi} \\ \mathbf{\Phi}^T\mathbf{\Phi}=\mathbf{I}}} \mathrm{Tr}(\mathbf{\Phi}^T\mathbf{A}\mathbf{\Phi})$$

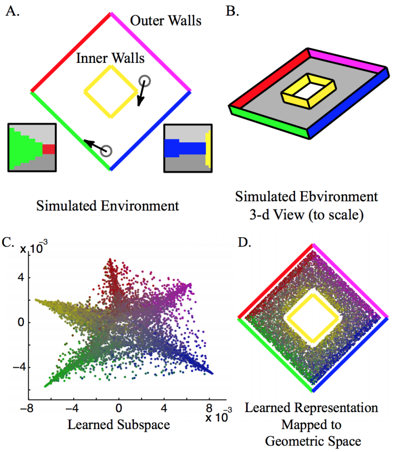
Functions on $\mathbb{R}^n$ can be decomposed with Fourier basis. Spectral analysis **generalizes Fourier analysis to manifolds and graphs**

What is this good for?



Tenenbaum, De Silva and Langford 2000

What is this good for?



Boots, Siddiqi and Gordon 2011

Outline

Theory

Graphs

Manifolds and Differential Geometry

Spectral Theory

Outline

Theory

Graphs

Manifolds and Differential Geometry

Spectral Theory

The Geometry of Data

Classic Manifold Learning

Embedding Hierarchies in Hyperbolic Space

Analyzing the Geometry of Deep Generative Models

Outline

Theory

- Graphs
- Manifolds and Differential Geometry
- Spectral Theory

The Geometry of Data

- Classic Manifold Learning
- Embedding Hierarchies in Hyperbolic Space
- Analyzing the Geometry of Deep Generative Models

Spectral Deep Learning

- Convolutions on Graphs and Manifolds
- Spectral Graph Convolutional Neural Networks
- Inference in Spectral Learning with Deep Networks

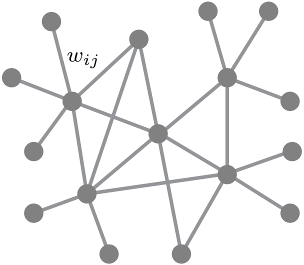
Part I

Theory

Graph Theory (in three slides)

Graphs

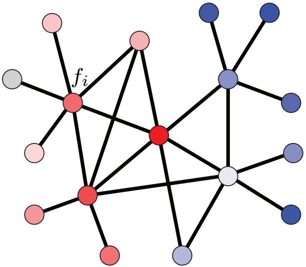
Weighted undirected graph $\mathcal{G}$ with vertices $\mathcal{V} = \{1, \ldots, n\}$, edges $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ and edge weights $w_{ij} \geq 0$ for $(i, j) \in \mathcal{E}$



Graphs

Weighted undirected graph $\mathcal{G}$ with vertices $\mathcal{V} = \{1, \ldots, n\}$, edges $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ and edge weights $w_{ij} \geq 0$ for $(i, j) \in \mathcal{E}$

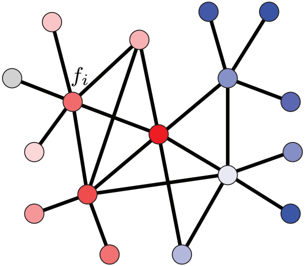
Functions over the vertices
$L^2(\mathcal{V}) = \{f : \mathcal{V} \to \mathbb{R}\}$



Graphs

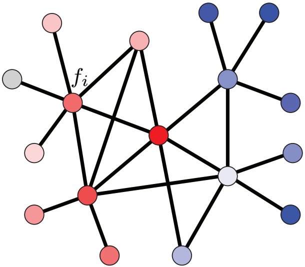
Weighted undirected graph $\mathcal{G}$ with vertices $\mathcal{V} = \{1, \ldots, n\}$, edges $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ and edge weights $w_{ij} \geq 0$ for $(i, j) \in \mathcal{E}$

Functions over the vertices $L^2(\mathcal{V}) = \{f : \mathcal{V} \to \mathbb{R}\}$ represented as vectors $\mathbf{f} = (f_1, \ldots, f_n)$



Graphs

Weighted undirected graph $\mathcal{G}$ with vertices $\mathcal{V} = \{1, \ldots, n\}$, edges $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ and edge weights $w_{ij} \geq 0$ for $(i, j) \in \mathcal{E}$



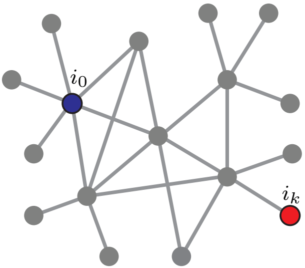
Functions over the vertices
$L^2(\mathcal{V}) = \{f : \mathcal{V} \to \mathbb{R}\}$ represented as vectors $\mathbf{f} = (f_1, \ldots, f_n)$

Inner product

$$\langle f, g \rangle_{L^2(\mathcal{V})} = \sum_{i \in \mathcal{V}} f_i \, g_i = \mathbf{f}^\top \mathbf{g}$$

Shortest paths on graphs

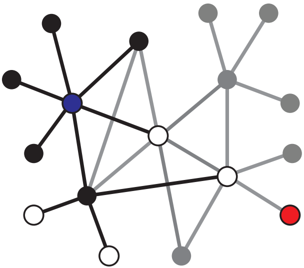
Dijkstra's Algorithm: Find sequence $i_0, \ldots, i_k \in \mathcal{V}$ from source i_0 to sink i_k that minimizes $\sum_t w_{i_t i_{t+1}}$



Shortest paths on graphs

Dijkstra's Algorithm: Find sequence $i_0, \ldots, i_k \in \mathcal{V}$ from source i_0 to sink i_k that minimizes $\sum_t w_{i_t i_{t+1}}$

Create queue of active nodes and shortest-path tree, starting from source

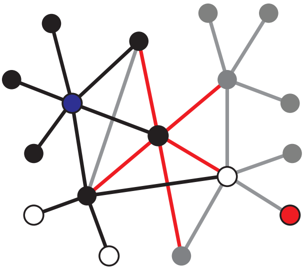


Shortest paths on graphs

Dijkstra's Algorithm: Find sequence $i_0, \ldots, i_k \in \mathcal{V}$ from source i_0 to sink i_k that minimizes $\sum_t w_{i_t i_{t+1}}$

Create queue of active nodes and shortest-path tree, starting from source

Remove active node i closest to source, compute distance to neighbors $\mathrm{dist}[j] = \mathrm{dist}[i] + w_{ij}$.



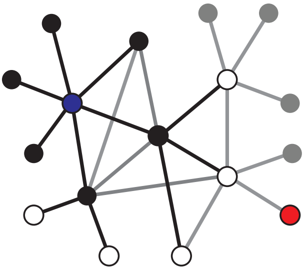
Shortest paths on graphs

Dijkstra's Algorithm: Find sequence $i_0, \ldots, i_k \in \mathcal{V}$ from source i_0 to sink i_k that minimizes $\sum_t w_{i_t i_{t+1}}$

Create queue of active nodes and shortest-path tree, starting from source

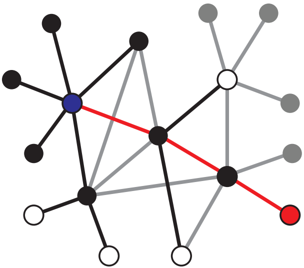
Remove active node i closest to source, compute distance to neighbors $\mathrm{dist}[j] = \mathrm{dist}[i] + w_{ij}$.

If $\mathrm{dist}[j]$ is less than before, set i as parent node of j in shortest path tree.



Shortest paths on graphs

Dijkstra's Algorithm: Find sequence $i_0, \ldots, i_k \in \mathcal{V}$ from source i_0 to sink i_k that minimizes $\sum_t w_{i_t i_{t+1}}$



Create queue of active nodes and shortest-path tree, starting from source

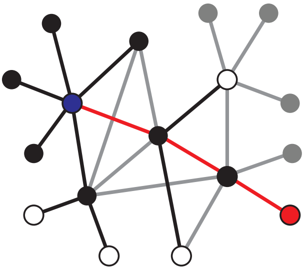
Remove active node i closest to source, compute distance to neighbors $\mathrm{dist}[j] = \mathrm{dist}[i] + w_{ij}$.

If $\mathrm{dist}[j]$ is less than before, set i as parent node of j in shortest path tree.

Stop when sink is reached. Path to the root of the tree is the shortest path.

Shortest paths on graphs

Dijkstra's Algorithm: Find sequence $i_0, \ldots, i_k \in \mathcal{V}$ from source i_0 to sink i_k that minimizes $\sum_t w_{i_t i_{t+1}}$



Create queue of active nodes and shortest-path tree, starting from source

Remove active node i closest to source, compute distance to neighbors $\mathrm{dist}[j] = \mathrm{dist}[i] + w_{ij}$.

If $\mathrm{dist}[j]$ is less than before, set i as parent node of j in shortest path tree.

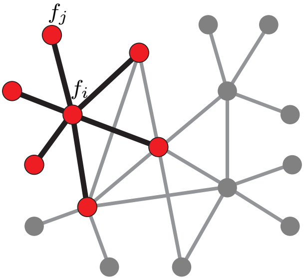
Stop when sink is reached. Path to the root of the tree is the shortest path.

Complexity: $\mathcal{O}(|\mathcal{E}| + |\mathcal{V}| \log |\mathcal{V}|)$

Graph Laplacian

Unnormalized Laplacian $\Delta : L^2(\mathcal{V}) \to L^2(\mathcal{V})$

$$(\Delta f)_i = \sum_{j:(i,j)\in\mathcal{E}} w_{ij}(f_i - f_j)$$

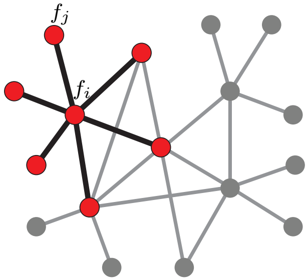


Graph Laplacian

Unnormalized Laplacian $\Delta : L^2(\mathcal{V}) \to L^2(\mathcal{V})$

$$
\begin{aligned}
(\Delta f)_i &= \sum_{j:(i,j)\in\mathcal{E}} w_{ij}(f_i - f_j) \\
&= f_i \sum_{j:(i,j)\in\mathcal{E}} w_{ij} - \sum_{j:(i,j)\in\mathcal{E}} w_{ij} f_j
\end{aligned}
$$

(up to scale) difference between f and its local average



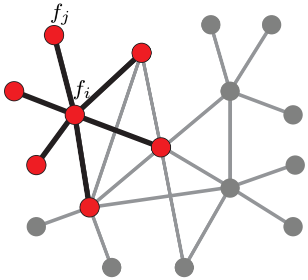
Graph Laplacian

Unnormalized Laplacian $\Delta : L^2(\mathcal{V}) \to L^2(\mathcal{V})$

$$(\Delta f)_i = \sum_{j:(i,j)\in\mathcal{E}} w_{ij}(f_i - f_j)$$

(up to scale) difference between f and its local average

Represented as a positive semi-definite $n \times n$ matrix $\mathbf{\Delta} = \mathbf{D} - \mathbf{W}$ where $\mathbf{W} = (w_{ij})$ and $\mathbf{D} = \mathrm{diag}(\sum_{j\neq i} w_{ij})$



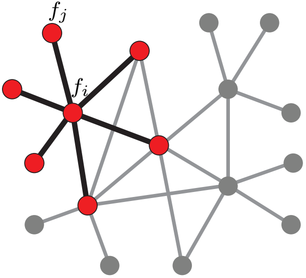
Graph Laplacian

Unnormalized Laplacian $\Delta : L^2(\mathcal{V}) \to L^2(\mathcal{V})$

$$(\Delta f)_i = \sum_{j:(i,j)\in\mathcal{E}} w_{ij}(f_i - f_j)$$

(up to scale) difference between f and its local average

Represented as a positive semi-definite $n \times n$ matrix $\mathbf{\Delta} = \mathbf{D} - \mathbf{W}$ where $\mathbf{W} = (w_{ij})$ and $\mathbf{D} = \mathrm{diag}(\sum_{j\neq i} w_{ij})$



Dirichlet energy of f

$$\|f\|^2_{\mathcal{G}} = \frac{1}{2}\sum_{ij=1}^{n} w_{ij}(f_i - f_j)^2 = \mathbf{f}^\top \mathbf{\Delta} \mathbf{f}$$

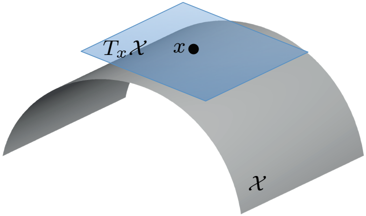
measures the smoothness of f (how fast it changes locally)

Manifolds and Differential Geometry

Riemannian manifolds

Manifold $\mathcal{X}$ = locally flat space (no non-differentiable corners)

Tangent plane $T_x\mathcal{X}$ = local Euclidean representation of manifold $\mathcal{X}$ around x



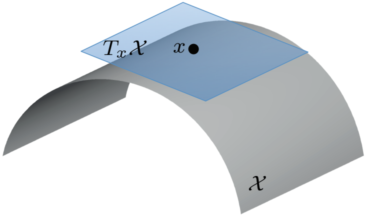
Riemannian manifolds

Manifold $\mathcal{X}$ = locally flat space (no non-differentiable corners)

Tangent plane $T_x\mathcal{X}$ = local Euclidean representation of manifold $\mathcal{X}$ around x

Riemannian metric describes the local intrinsic structure at x

$$\langle \cdot, \cdot \rangle_{T_x\mathcal{X}} : T_x\mathcal{X} \times T_x\mathcal{X} \to \mathbb{R}$$



Riemannian manifolds

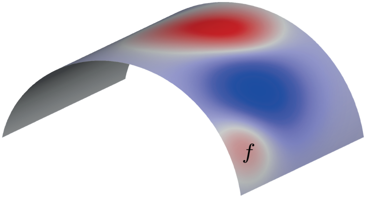
Manifold $\mathcal{X}$ = locally flat space (no non-differentiable corners)

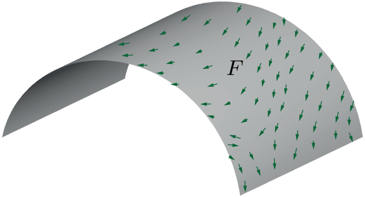
Tangent plane $T_x\mathcal{X}$ = local Euclidean representation of manifold $\mathcal{X}$ around x

Riemannian metric describes the local intrinsic structure at x

$$\langle \cdot, \cdot \rangle_{T_x\mathcal{X}} : T_x\mathcal{X} \times T_x\mathcal{X} \to \mathbb{R}$$

Scalar fields $f : \mathcal{X} \to \mathbb{R}$ and vector fields $F : \mathcal{X} \to T\mathcal{X}$





Riemannian manifolds

Manifold $\mathcal{X}$ = locally flat space (no non-differentiable corners)

Tangent plane $T_x\mathcal{X}$ = local Euclidean representation of manifold $\mathcal{X}$ around x

Riemannian metric describes the local intrinsic structure at x

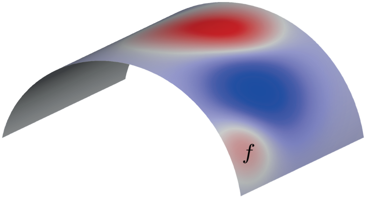
$$\langle \cdot, \cdot \rangle_{T_x\mathcal{X}} : T_x\mathcal{X} \times T_x\mathcal{X} \to \mathbb{R}$$

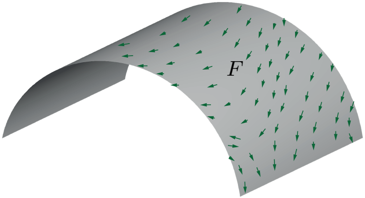
Scalar fields $f : \mathcal{X} \to \mathbb{R}$ and vector fields $F : \mathcal{X} \to T\mathcal{X}$

Inner products

$$\langle f, g \rangle_{L^2(\mathcal{X})} = \int_{\mathcal{X}} f(x) g(x) dx$$

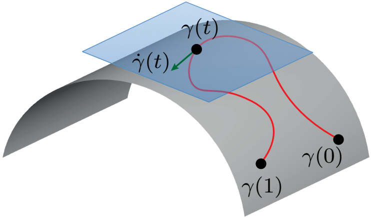
$$\langle F, G \rangle_{L^2(T\mathcal{X})} = \int_{\mathcal{X}} \langle F(x), G(x) \rangle_{T_x\mathcal{X}} dx$$





Shortest paths on manifolds

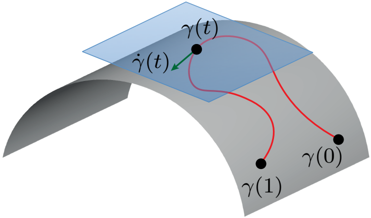
Curve $\gamma(\cdot) : [0, 1] \to \mathcal{X}$ = smooth path along manifold



Shortest paths on manifolds

Curve $\gamma(\cdot) : [0, 1] \to \mathcal{X}$ = smooth path along manifold

Velocity $\dot{\gamma}(\cdot) : [0, 1] \to T_x\mathcal{X}$ = local direction of change along the curve



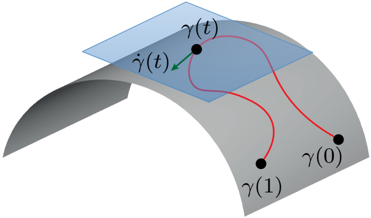
Shortest paths on manifolds

Curve $\gamma(\cdot) : [0,1] \to \mathcal{X}$ = smooth path along manifold

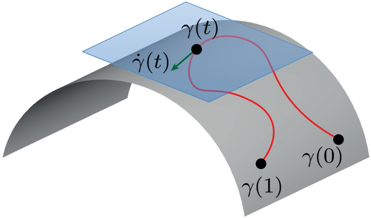
Velocity $\dot{\gamma}(\cdot) : [0,1] \to T_x\mathcal{X}$ = local direction of change along the curve

Curve energy integrates length at constant velocity

$$\mathcal{S}[\gamma] = \int_0^1 dt \left\langle \dot{\gamma(t)}, \dot{\gamma(t)} \right\rangle_{T_{\gamma(t)}\mathcal{X}}$$



Shortest paths on manifolds



Curve $\gamma(\cdot) : [0, 1] \to \mathcal{X}$ = smooth path along manifold

Velocity $\dot{\gamma}(\cdot) : [0, 1] \to T_x\mathcal{X}$ = local direction of change along the curve

Curve energy integrates length at constant velocity

$$\mathcal{S}[\gamma] = \int_0^1 dt \left\langle \dot{\gamma(t)}, \dot{\gamma(t)} \right\rangle_{T_{\gamma(t)}\mathcal{X}}$$

Geodesic shortest curve between points

$$\gamma^* = \min_{\substack{\gamma \\ \gamma(0)=x_0 \\ \gamma(1)=x_1}} \mathcal{S}[\gamma]$$

Shortest paths on manifolds

Curve $\gamma(\cdot) : [0,1] \to \mathcal{X}$ = smooth path along manifold

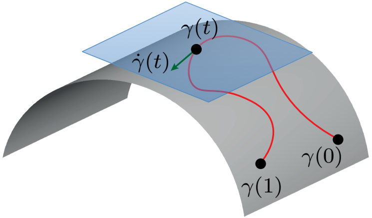
Velocity $\dot{\gamma}(\cdot) : [0,1] \to T_x\mathcal{X}$ = local direction of change along the curve

Curve energy integrates length at constant velocity

$$\mathcal{S}[\gamma] = \int_0^1 dt \left\langle \dot{\gamma(t)}, \dot{\gamma(t)} \right\rangle_{T_{\gamma(t)}\mathcal{X}}$$

Geodesic shortest curve between points

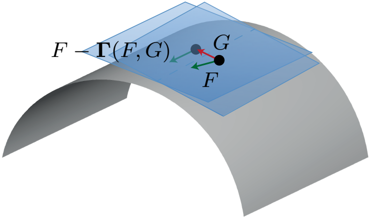
$$\gamma^* = \min_{\substack{\gamma \\ \gamma(0)=x_0 \\ \gamma(1)=x_1}} \mathcal{S}[\gamma]$$



Geodesic is a path such that the velocities are locally parallel

Parallel Transport

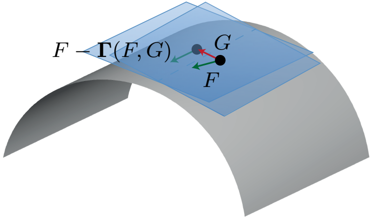
Connection $\mathbf{\Gamma}_x(F, G)$: infinitesimal change to vector $F \in T_x\mathcal{X}$ that keeps it locally parallel when moved in the direction $G \in T_x\mathcal{X}$.

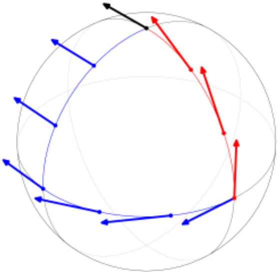


Parallel Transport

Connection $\mathbf{\Gamma}_x(F, G)$: infinitesimal change to vector $F \in T_x\mathcal{X}$ that keeps it locally parallel when moved in the direction $G \in T_x\mathcal{X}$.

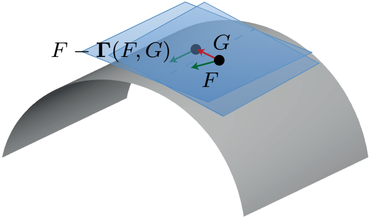
Parallel transport Sequence of vectors $F(t) \in T_{\gamma(t)}\mathcal{X}$ along curve γ that are all locally parallel.





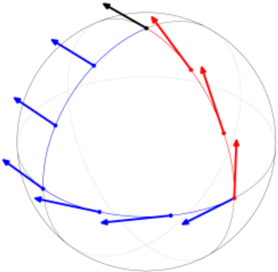
Parallel Transport

Connection $\mathbf{\Gamma}_x(F, G)$: infinitesimal change to vector $F \in T_x\mathcal{X}$ that keeps it locally parallel when moved in the direction $G \in T_x\mathcal{X}$.



Parallel transport Sequence of vectors $F(t) \in T_{\gamma(t)}\mathcal{X}$ along curve γ that are all locally parallel.

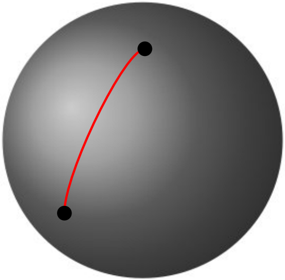
$$\dot{F}(t) + \mathbf{\Gamma}_{\gamma(t)}(F(t), \dot{\gamma}(t)) = 0$$



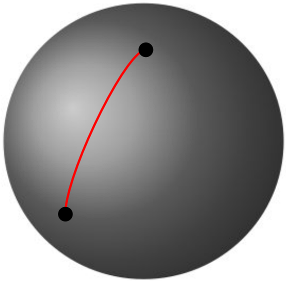
Geodesic equation

Parallel transport:

$$\dot{F}(t) + \boldsymbol{\Gamma}_{\gamma(t)}(F(t), \dot{\gamma}(t)) = 0$$



Geodesic equation



Parallel transport:

$$\dot{F}(t) + \mathbf{\Gamma}_{\gamma(t)}(F(t), \dot{\gamma}(t)) = 0$$

The geodesic equation:

$$\ddot{\gamma}(t) + \mathbf{\Gamma}_{\gamma(t)}(\dot{\gamma}(t), \dot{\gamma}(t)) = 0$$

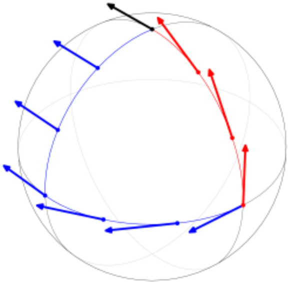
Geodesic equation

Parallel transport:

$$\dot{F}(t) + \mathbf{\Gamma}_{\gamma(t)}(F(t), \dot{\gamma}(t)) = 0$$

The geodesic equation:

$$\ddot{\gamma}(t) + \mathbf{\Gamma}_{\gamma(t)}(\dot{\gamma}(t), \dot{\gamma}(t)) = 0$$



A geodesic is a curve such that if you parallel transport its velocity at a point along itself, the vector always remains tangent to the curve

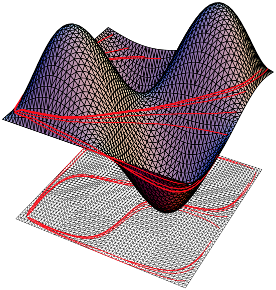
Geodesic equation

Parallel transport:

$$\dot{F}(t) + \mathbf{\Gamma}_{\gamma(t)}(F(t), \dot{\gamma}(t)) = 0$$

The geodesic equation:

$$\ddot{\gamma}(t) + \mathbf{\Gamma}_{\gamma(t)}(\dot{\gamma}(t), \dot{\gamma}(t)) = 0$$



A geodesic is a curve such that if you parallel transport its velocity at a point along itself, the vector always remains tangent to the curve

Solve numerically on meshes with fast marching method

Kimmel and Sethian 1998

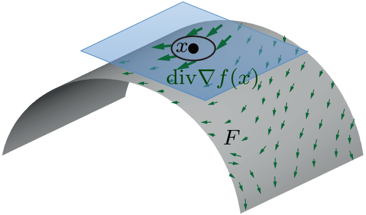
Manifold Laplacian

Laplacian $\Delta : L^2(\mathcal{X}) \to L^2(\mathcal{X})$

$$\Delta f(x) = -\mathrm{div}\,\nabla f(x)$$

where gradient $\nabla : L^2(\mathcal{X}) \to L^2(T\mathcal{X})$
and divergence $\mathrm{div} : L^2(T\mathcal{X}) \to L^2(\mathcal{X})$
are adjoint operators

$$\langle F, \nabla f \rangle_{L^2(T\mathcal{X})} = \langle -\mathrm{div} F, f \rangle_{L^2(\mathcal{X})}$$



Manifold Laplacian

Laplacian $\Delta : L^2(\mathcal{X}) \to L^2(\mathcal{X})$

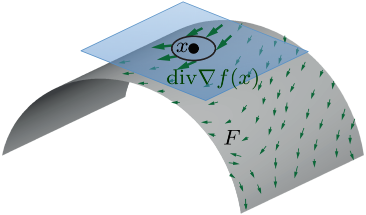
$$\Delta f(x) = -\mathrm{div}\, \nabla f(x)$$

where gradient $\nabla : L^2(\mathcal{X}) \to L^2(T\mathcal{X})$
and divergence $\mathrm{div} : L^2(T\mathcal{X}) \to L^2(\mathcal{X})$
are adjoint operators

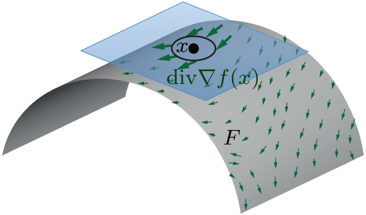
$$\langle F, \nabla f \rangle_{L^2(T\mathcal{X})} = \langle -\mathrm{div} F, f \rangle_{L^2(\mathcal{X})}$$

Laplacian is self-adjoint

$$\langle \Delta f, f \rangle_{L^2(\mathcal{X})} = \langle f, \Delta f \rangle_{L^2(\mathcal{X})}$$



Manifold Laplacian



Laplacian $\Delta : L^2(\mathcal{X}) \to L^2(\mathcal{X})$

$$\Delta f(x) = -\mathrm{div}\, \nabla f(x)$$

where gradient $\nabla : L^2(\mathcal{X}) \to L^2(T\mathcal{X})$
and divergence $\mathrm{div} : L^2(T\mathcal{X}) \to L^2(\mathcal{X})$
are adjoint operators

$$\langle F, \nabla f \rangle_{L^2(T\mathcal{X})} = \langle -\mathrm{div} F, f \rangle_{L^2(\mathcal{X})}$$

Laplacian is self-adjoint

$$\langle \Delta f, f \rangle_{L^2(\mathcal{X})} = \langle f, \Delta f \rangle_{L^2(\mathcal{X})}$$

Continuous limit of graph Laplacian
under some conditions

Dirichlet energy of f

$$\langle \nabla f, \nabla f \rangle_{L^2(T\mathcal{X})} = \int_{\mathcal{X}} f(x) \Delta f(x) dx$$

measures the smoothness of f (how fast it changes locally)

Spectral Theory for Graphs and Manifolds

Orthogonal bases on graphs

Find the smoothest orthogonal basis $\{\phi_1, \ldots, \phi_n\} \subseteq L^2(\mathcal{V})$

$$
\begin{aligned}
&\min_{\phi_1} \; E_{\mathrm{Dir}}(\psi_1) \quad \text{s.t.} \quad \|\phi_1\| = 1 \\
&\min_{\phi_k} \; E_{\mathrm{Dir}}(\psi_k) \quad \text{s.t.} \quad \|\phi_k\| = 1, \quad k = 2, 3, \ldots n \\
&\qquad\qquad\qquad\qquad\qquad\;\; \phi_k \perp \mathrm{span}\{\phi_1, \ldots, \phi_{k-1}\}
\end{aligned}
$$

Orthogonal bases on graphs

Find the smoothest orthogonal basis $\{\phi_1, \ldots, \phi_n\} \subseteq L^2(\mathcal{V})$

$$\min_{\mathbf{\Phi} \in \mathbb{R}^{n \times n}} \mathrm{trace}(\mathbf{\Phi}^\top \mathbf{\Delta} \mathbf{\Phi}) \quad \text{s.t.} \quad \mathbf{\Phi}^\top \mathbf{\Phi} = \mathbf{I}$$

Orthogonal bases on graphs

Find the smoothest orthogonal basis $\{\phi_1, \ldots, \phi_n\} \subseteq L^2(\mathcal{V})$

$$\min_{\mathbf{\Phi} \in \mathbb{R}^{n \times n}} \mathrm{trace}(\mathbf{\Phi}^\top \mathbf{\Delta} \mathbf{\Phi}) \quad \text{s.t.} \quad \mathbf{\Phi}^\top \mathbf{\Phi} = \mathbf{I}$$

Solution: $\mathbf{\Phi} =$ Laplacian eigenvectors

Laplacian eigenvectors and eigenvalues

Eigendecomposition of a graph Laplacian

$$\mathbf{\Delta} = \mathbf{\Phi}\mathbf{\Lambda}\mathbf{\Phi}^\top$$

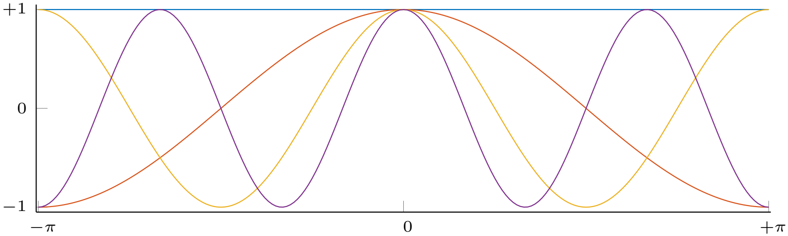
where $\mathbf{\Phi} = (\boldsymbol{\phi}_1, \ldots, \boldsymbol{\phi}_n)$ are orthogonal eigenvectors $(\mathbf{\Phi}^\top\mathbf{\Phi} = \mathbf{I})$ and $\mathbf{\Lambda} = \mathrm{diag}(\lambda_1, \ldots, \lambda_n)$ the corresponding non-negative eigenvalues

Laplacian eigenvectors and eigenvalues

Eigendecomposition of a graph Laplacian

$$\mathbf{\Delta} = \mathbf{\Phi}\mathbf{\Lambda}\mathbf{\Phi}^\top$$

where $\mathbf{\Phi} = (\boldsymbol{\phi}_1, \ldots, \boldsymbol{\phi}_n)$ are orthogonal eigenvectors $(\mathbf{\Phi}^\top\mathbf{\Phi} = \mathbf{I})$ and $\mathbf{\Lambda} = \mathrm{diag}(\lambda_1, \ldots, \lambda_n)$ the corresponding non-negative eigenvalues



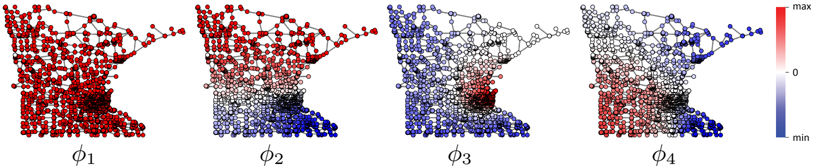
First eigenfunctions of 1D Euclidean Laplacian

Laplacian eigenvectors and eigenvalues

Eigendecomposition of a graph Laplacian

$$\mathbf{\Delta} = \mathbf{\Phi} \mathbf{\Lambda} \mathbf{\Phi}^\top$$

where $\mathbf{\Phi} = (\phi_1, \ldots, \phi_n)$ are orthogonal eigenvectors ($\mathbf{\Phi}^\top \mathbf{\Phi} = \mathbf{I}$) and $\mathbf{\Lambda} = \mathrm{diag}(\lambda_1, \ldots, \lambda_n)$ the corresponding non-negative eigenvalues



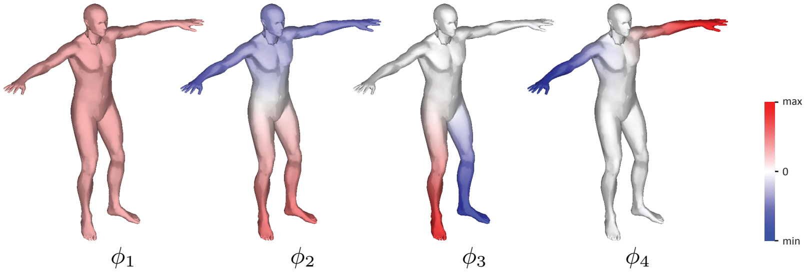
First eigenfunctions of a graph Laplacian

Laplacian eigenvectors and eigenvalues

Eigendecomposition of a graph Laplacian

$$\mathbf{\Delta} = \mathbf{\Phi}\mathbf{\Lambda}\mathbf{\Phi}^\top$$

where $\mathbf{\Phi} = (\phi_1, \ldots, \phi_n)$ are orthogonal eigenvectors ($\mathbf{\Phi}^\top\mathbf{\Phi} = \mathbf{I}$) and $\mathbf{\Lambda} = \mathrm{diag}(\lambda_1, \ldots, \lambda_n)$ the corresponding non-negative eigenvalues



First eigenfunctions of a manifold Laplacian

Fourier analysis on Euclidean spaces

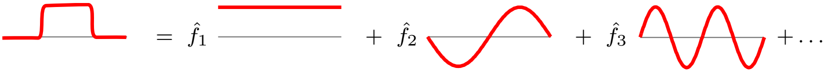
A function $f : [-\pi, \pi] \to \mathbb{R}$ can be written as a Fourier series

$$f(x) = \sum_{k \geq 0} \frac{1}{2\pi} \int_{-\pi}^{\pi} f(x') e^{-ikx'} dx' e^{ikx}$$

Fourier analysis on Euclidean spaces

A function $f : [-\pi, \pi] \to \mathbb{R}$ can be written as a Fourier series

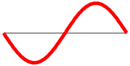
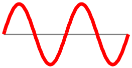
$$f(x) = \sum_{k \geq 0} \langle f, e^{ikx} \rangle_{L^2([-\pi,\pi])} \, e^{ikx}$$



Fourier analysis on Euclidean spaces

A function $f : [-\pi, \pi] \to \mathbb{R}$ can be written as a Fourier series

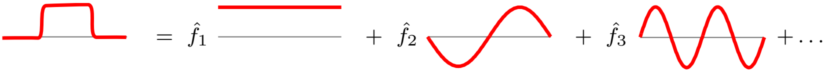
$$f(x) = \sum_{k \geq 0} \underbrace{\langle f, e^{ikx} \rangle_{L^2([-\pi,\pi])}}_{\hat{f}_k \text{ Fourier coefficient}} e^{ikx}$$

 $= \hat{f}_1$  $+ \; \hat{f}_2$  $+ \; \hat{f}_3$  $+ \ldots$

Fourier analysis on Euclidean spaces

A function $f : [-\pi, \pi] \to \mathbb{R}$ can be written as a Fourier series

$$f(x) = \sum_{k \geq 0} \underbrace{\langle f, e^{ikx} \rangle_{L^2([-\pi,\pi])}}_{\hat{f}_k \text{ Fourier coefficient}} e^{ikx}$$



Fourier basis = Laplacian eigenfunctions: $-\frac{d^2}{dx^2} e^{ikx} = k^2 e^{ikx}$

Fourier analysis on graphs and manifolds

A function $f : \mathcal{V} \to \mathbb{R}$ can be written as Fourier series

$$f = \sum_{k=1}^{n} \underbrace{\langle f, \phi_k \rangle_{L^2(\mathcal{V})}}_{\hat{f}_k} \phi_k$$

Fourier basis = Laplacian eigenfunctions: $\Delta \phi_k = \lambda_k \phi_k$

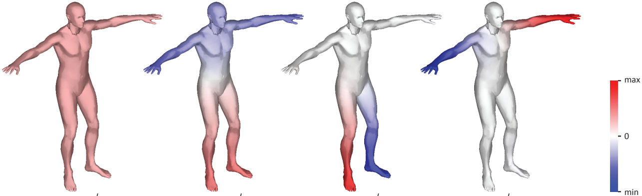
Fourier analysis on graphs and manifolds

A function $f : \mathcal{V} \to \mathbb{R}$ can be written as Fourier series

$$f = \sum_{k=1}^{n} \underbrace{\langle f, \phi_k \rangle_{L^2(\mathcal{V})}}_{\hat{f}_k} \ \phi_k$$

Fourier basis = Laplacian eigenfunctions: $\Delta \phi_k = \lambda_k \phi_k$

λ_k = frequency



ϕ_1 ϕ_2 ϕ_3 ϕ_4

First Fourier basis elements of a manifold.

Summary

Manifolds and graphs are natural extensions of vector spaces

Summary

Manifolds and graphs are natural extensions of vector spaces

Lines can be generalized to shortest paths and geodesics

Summary

Manifolds and graphs are natural extensions of vector spaces

Lines can be generalized to shortest paths and geodesics

The Laplacian operator defines smoothness of a function

Summary

Manifolds and graphs are natural extensions of vector spaces

Lines can be generalized to shortest paths and geodesics

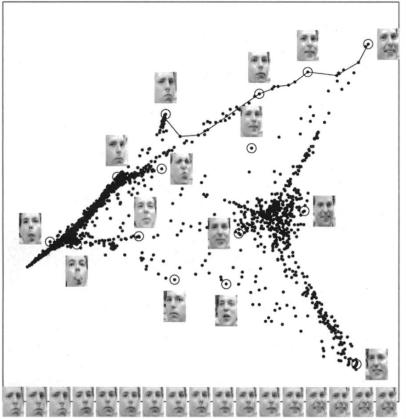
The Laplacian operator defines smoothness of a function

Spectral decompositions generalize Fourier analysis

The Geometry of Data

Classic Manifold Learning: A Time Before Deep Learning

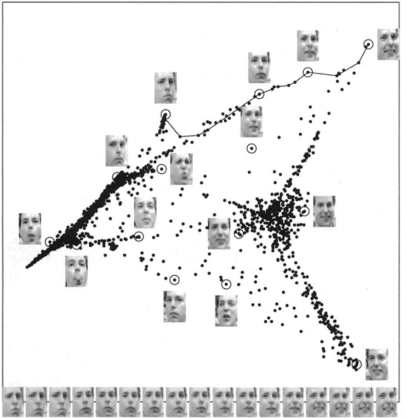
Manifold Learning



Learn nonlinear embedding of data into low dimensional space that preserves distances locally

Roweis and Saul 2000

Manifold Learning

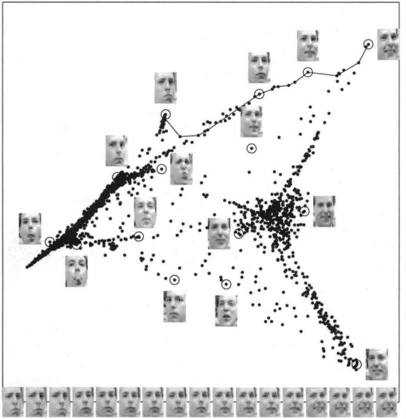


Learn nonlinear embedding of data into low dimensional space that preserves distances locally

The data itself does not have to be manifold structured

Roweis and Saul 2000

Manifold Learning



Learn nonlinear embedding of data into low dimensional space that preserves distances locally

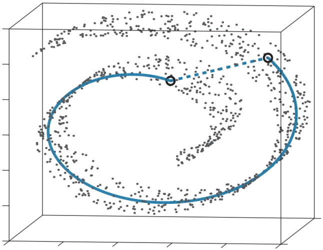
The data itself does not have to be manifold structured

The dataset has some latent manifold structure

Roweis and Saul 2000

IsoMap

Compute nearest neighbors graph of dataset

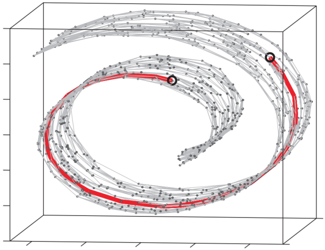


Tenenbaum, de Silva and Langford 2000

IsoMap

Compute nearest neighbors graph of dataset

Construct matrix $\mathbf{D} \in \mathbb{R}^{n \times n}$ of geodesic distances between pairs of data by Dijkstra's algorithm



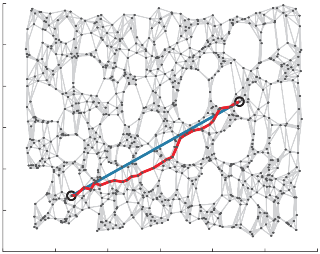
Tenenbaum, de Silva and Langford 2000

IsoMap

Compute nearest neighbors graph of dataset

Construct matrix $\mathbf{D} \in \mathbb{R}^{n \times n}$ of geodesic distances between pairs of data by Dijkstra's algorithm

Embed data in low dimensional space with multidimensional scaling (MDS):



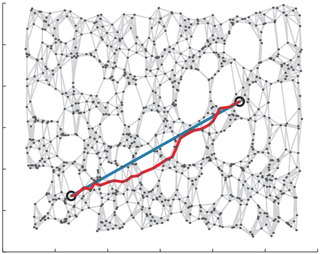
Tenenbaum, de Silva and Langford 2000

IsoMap

Compute nearest neighbors graph of dataset

Construct matrix $\mathbf{D} \in \mathbb{R}^{n \times n}$ of geodesic distances between pairs of data by Dijkstra's algorithm

Embed data in low dimensional space with multidimensional scaling (MDS):



$$\mathbf{B} = -\frac{1}{2}\left(\mathbf{I} - \frac{1}{n}\mathbf{1}\mathbf{1}^T\right)\mathbf{D}\left(\mathbf{I} - \frac{1}{n}\mathbf{1}\mathbf{1}^T\right)$$

Tenenbaum, de Silva and Langford 2000

IsoMap

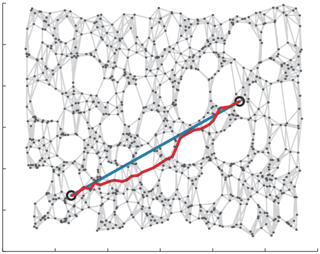
Compute nearest neighbors graph of dataset

Construct matrix $\mathbf{D} \in \mathbb{R}^{n \times n}$ of geodesic distances between pairs of data by Dijkstra's algorithm

Embed data in low dimensional space with multidimensional scaling (MDS):

$$\mathbf{B} = -\frac{1}{2}\left(\mathbf{I} - \frac{1}{n}\mathbf{1}\mathbf{1}^T\right)\mathbf{D}\left(\mathbf{I} - \frac{1}{n}\mathbf{1}\mathbf{1}^T\right)$$

$\lambda_1, \ldots, \lambda_k$, $\phi_1, \ldots, \phi_k$ top eigenvalues and eigenvectors of $\mathbf{B}$



Tenenbaum, de Silva and Langford 2000

IsoMap

Compute nearest neighbors graph of dataset

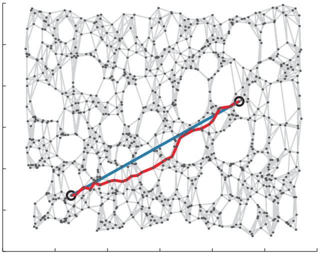
Construct matrix $\mathbf{D} \in \mathbb{R}^{n \times n}$ of geodesic distances between pairs of data by Dijkstra's algorithm

Embed data in low dimensional space with multidimensional scaling (MDS):

$$\mathbf{B} = -\frac{1}{2}\left(\mathbf{I} - \frac{1}{n}\mathbf{1}\mathbf{1}^T\right)\mathbf{D}\left(\mathbf{I} - \frac{1}{n}\mathbf{1}\mathbf{1}^T\right)$$

$\lambda_1, \ldots, \lambda_k$, $\phi_1, \ldots, \phi_k$ top eigenvalues and eigenvectors of $\mathbf{B}$

Embedding: $\left(\sqrt{\lambda_1}\phi_1, \ldots, \sqrt{\lambda_k}\phi_k\right)$



Tenenbaum, de Silva and Langford 2000

IsoMap

Compute nearest neighbors graph of dataset

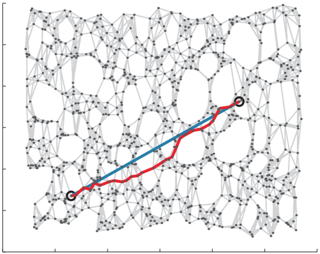
Construct matrix $\mathbf{D} \in \mathbb{R}^{n \times n}$ of geodesic distances between pairs of data by Dijkstra's algorithm

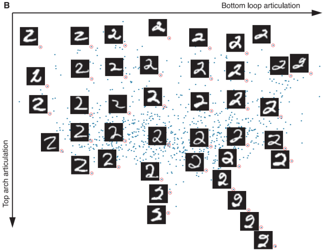
Embed data in low dimensional space with multidimensional scaling (MDS):

$$\mathbf{B} = -\frac{1}{2}\left(\mathbf{I} - \frac{1}{n}\mathbf{1}\mathbf{1}^T\right)\mathbf{D}\left(\mathbf{I} - \frac{1}{n}\mathbf{1}\mathbf{1}^T\right)$$

$\lambda_1, \ldots, \lambda_k$, $\phi_1, \ldots, \phi_k$ top eigenvalues and eigenvectors of $\mathbf{B}$

Embedding: $\left(\sqrt{\lambda_1}\phi_1, \ldots, \sqrt{\lambda_k}\phi_k\right)$





Tenenbaum, de Silva and Langford 2000

IsoMap

Compute nearest neighbors graph of dataset

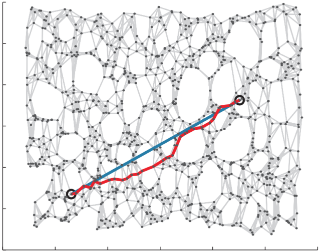
Construct matrix $\mathbf{D} \in \mathbb{R}^{n \times n}$ of geodesic distances between pairs of data by Dijkstra's algorithm

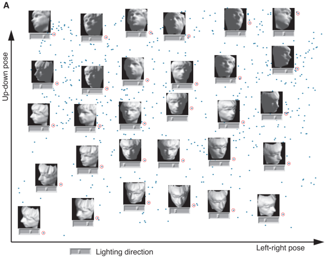
Embed data in low dimensional space with multidimensional scaling (MDS):

$$\mathbf{B} = -\frac{1}{2}\left(\mathbf{I} - \frac{1}{n}\mathbf{1}\mathbf{1}^T\right)\mathbf{D}\left(\mathbf{I} - \frac{1}{n}\mathbf{1}\mathbf{1}^T\right)$$

$\lambda_1, \ldots, \lambda_k$, $\phi_1, \ldots, \phi_k$ top eigenvalues and eigenvectors of $\mathbf{B}$

Embedding: $\left(\sqrt{\lambda_1}\phi_1, \ldots, \sqrt{\lambda_k}\phi_k\right)$

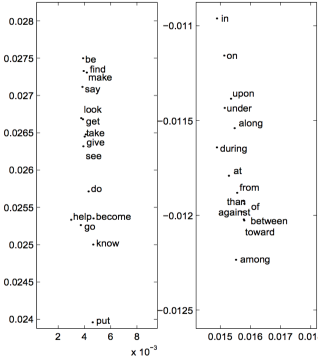




Tenenbaum, de Silva and Langford 2000

Laplacian eigenmaps

Compute weighted nearest neighbors graph of dataset

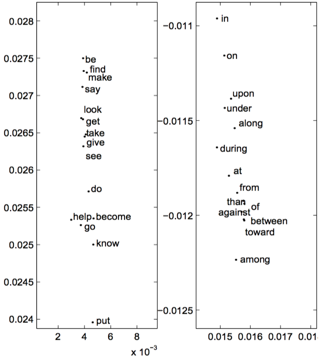


Belkin and Niyogi 2002

Laplacian eigenmaps

Compute weighted nearest neighbors graph of dataset

Construct graph Laplacian matrix $\mathbf{\Delta} = \mathbf{D} - \mathbf{W}$



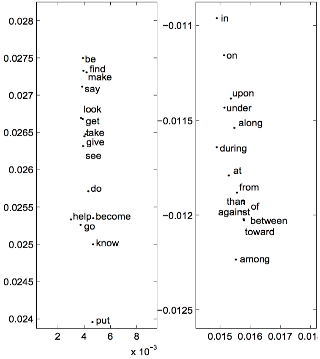
Belkin and Niyogi 2002

Laplacian eigenmaps

Compute weighted nearest neighbors graph of dataset

Construct graph Laplacian matrix
$\mathbf{\Delta} = \mathbf{D} - \mathbf{W}$

$\lambda_1, \ldots, \lambda_k$, $\phi_1, \ldots, \phi_k$ bottom eigenvalues and eigenvectors of $\mathbf{\Delta}$



Belkin and Niyogi 2002

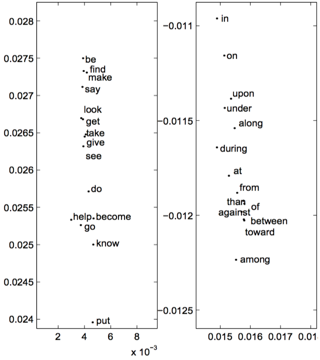
Laplacian eigenmaps

Compute weighted nearest neighbors graph of dataset

Construct graph Laplacian matrix $\mathbf{\Delta} = \mathbf{D} - \mathbf{W}$

$\lambda_1, \ldots, \lambda_k$, $\phi_1, \ldots, \phi_k$ bottom eigenvalues and eigenvectors of $\mathbf{\Delta}$

Embedding: $(\phi_2, \ldots, \phi_k)$



Belkin and Niyogi 2002

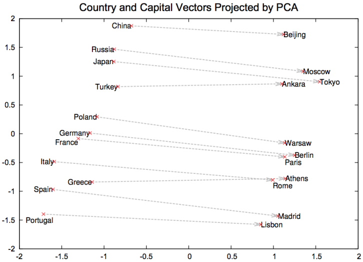
Laplacian eigenmaps

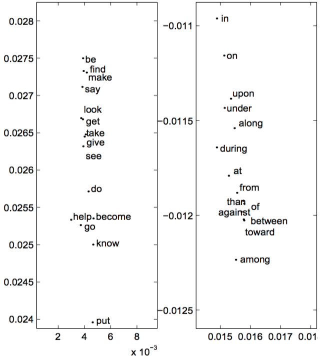
Compute weighted nearest neighbors graph of dataset

Construct graph Laplacian matrix $\mathbf{\Delta} = \mathbf{D} - \mathbf{W}$

$\lambda_1, \ldots, \lambda_k,\ \phi_1, \ldots, \phi_k$ bottom eigenvalues and eigenvectors of $\mathbf{\Delta}$

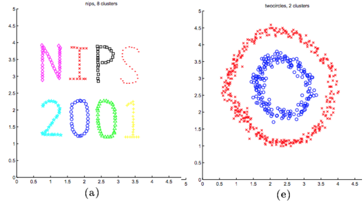
Embedding: $(\phi_2, \ldots, \phi_k)$



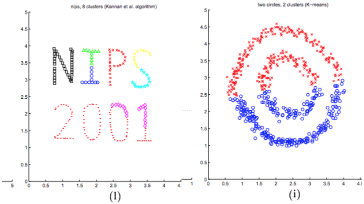


Spectral clustering

Same embedding as Laplacian eigenmaps, but use embedding vectors for clustering instead of visualization



Spectral Clustering



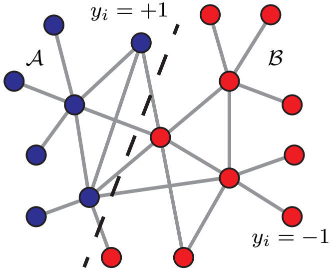
Other

Ng, Jordan and Weiss 2002

Normalized cuts

Minimum (normalized) cut problem:
partition vertices into sets $\mathcal{A}, \mathcal{B} \subset \mathcal{V}$

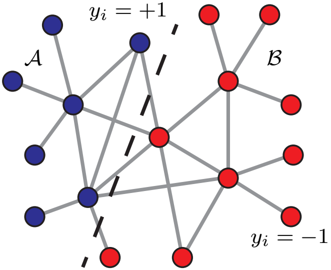
$$\mathrm{NCut}(\mathbf{y}) = \frac{\sum_{i \in \mathcal{A}, j \in \mathcal{B}} w_{ij}}{\sum_{i \in \mathcal{A}, j \in \mathcal{V}} w_{ij}} + \frac{\sum_{i \in \mathcal{B}, j \in \mathcal{A}} w_{ij}}{\sum_{i \in \mathcal{B}, j \in \mathcal{V}} w_{ij}}$$



Shi and Malik 2000

Normalized cuts

Minimum (normalized) cut problem:
partition vertices into sets $\mathcal{A}, \mathcal{B} \subset \mathcal{V}$

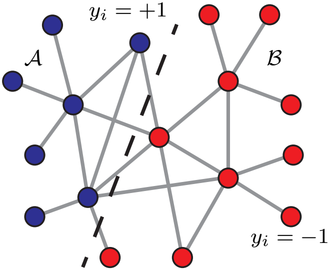


$$\begin{aligned} \mathrm{NCut}(\mathbf{y}) &= \frac{\sum_{i \in \mathcal{A}, j \in \mathcal{B}} w_{ij}}{\sum_{i \in \mathcal{A}, j \in \mathcal{V}} w_{ij}} + \frac{\sum_{i \in \mathcal{B}, j \in \mathcal{A}} w_{ij}}{\sum_{i \in \mathcal{B}, j \in \mathcal{V}} w_{ij}} \\ &= \frac{\mathbf{y}^T(\mathbf{D} - \mathbf{W})\mathbf{y}}{\mathbf{y}^T\mathbf{y}} \end{aligned}$$

Shi and Malik 2000

Normalized cuts

Minimum (normalized) cut problem:
partition vertices into sets $\mathcal{A}, \mathcal{B} \subset \mathcal{V}$

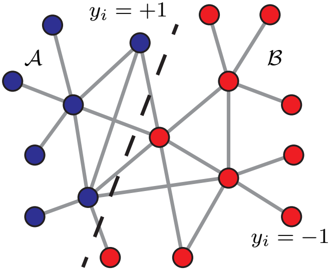


$$\begin{aligned}\mathrm{NCut}(\mathbf{y}) &= \frac{\sum_{i\in\mathcal{A},j\in\mathcal{B}} w_{ij}}{\sum_{i\in\mathcal{A},j\in\mathcal{V}} w_{ij}} + \frac{\sum_{i\in\mathcal{B},j\in\mathcal{A}} w_{ij}}{\sum_{i\in\mathcal{B},j\in\mathcal{V}} w_{ij}} \\ &= \frac{\mathbf{y}^T\boldsymbol{\Delta}\mathbf{y}}{\mathbf{y}^T\mathbf{y}}\end{aligned}$$

Shi and Malik 2000

Normalized cuts

Minimum (normalized) cut problem: partition vertices into sets $\mathcal{A}, \mathcal{B} \subset \mathcal{V}$



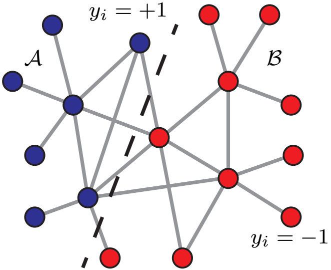
$$
\begin{aligned}
\mathrm{NCut}(\mathbf{y}) &= \frac{\sum_{i\in\mathcal{A},j\in\mathcal{B}} w_{ij}}{\sum_{i\in\mathcal{A},j\in\mathcal{V}} w_{ij}} + \frac{\sum_{i\in\mathcal{B},j\in\mathcal{A}} w_{ij}}{\sum_{i\in\mathcal{B},j\in\mathcal{V}} w_{ij}} \\
&= \frac{\mathbf{y}^T \boldsymbol{\Delta} \mathbf{y}}{\mathbf{y}^T \mathbf{y}}
\end{aligned}
$$

Relax optimization from binary vectors to continuous vectors. Solution is bottom eigenfunctions of graph Laplacian.

Shi and Malik 2000

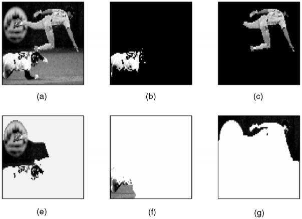
Normalized cuts

Minimum (normalized) cut problem: partition vertices into sets $\mathcal{A}, \mathcal{B} \subset \mathcal{V}$



$$\begin{aligned}\mathrm{NCut}(\mathbf{y}) &= \frac{\sum_{i\in\mathcal{A},j\in\mathcal{B}} w_{ij}}{\sum_{i\in\mathcal{A},j\in\mathcal{V}} w_{ij}} + \frac{\sum_{i\in\mathcal{B},j\in\mathcal{A}} w_{ij}}{\sum_{i\in\mathcal{B},j\in\mathcal{V}} w_{ij}} \\ &= \frac{\mathbf{y}^T\boldsymbol{\Delta}\mathbf{y}}{\mathbf{y}^T\mathbf{y}}\end{aligned}$$

Relax optimization from binary vectors to continuous vectors. Solution is bottom eigenfunctions of graph Laplacian.



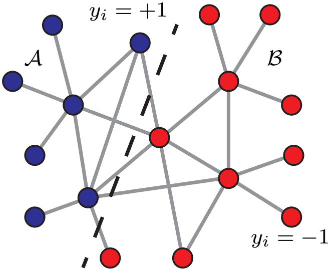
Used in image segmentation where each vertex is a pixel

Shi and Malik 2000

Normalized cuts

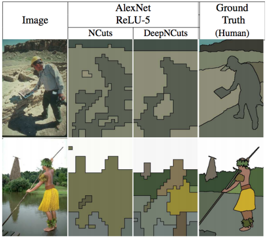
Minimum (normalized) cut problem:
partition vertices into sets $\mathcal{A}, \mathcal{B} \subset \mathcal{V}$

$$
\begin{aligned}
\mathrm{NCut}(\mathbf{y}) &= \frac{\sum_{i\in\mathcal{A},j\in\mathcal{B}} w_{ij}}{\sum_{i\in\mathcal{A},j\in\mathcal{V}} w_{ij}} + \frac{\sum_{i\in\mathcal{B},j\in\mathcal{A}} w_{ij}}{\sum_{i\in\mathcal{B},j\in\mathcal{V}} w_{ij}} \\
&= \frac{\mathbf{y}^T \boldsymbol{\Delta} \mathbf{y}}{\mathbf{y}^T \mathbf{y}}
\end{aligned}
$$



Relax optimization from binary vectors to continuous vectors. Solution is bottom eigenfunctions of graph Laplacian.

Can be incorporated as layer inside a deep network by backpropagating through eigendecomposition



Shi and Malik 2000; Ionescu, Vantzos and Sminchisescu 2015

Inference on held-out data

A common pattern:

Inference on held-out data

A common pattern:

Construct Gram matrix from kernel $k(\mathbf{x}, \mathbf{x}')$

$$\mathbf{M} = \begin{bmatrix} k(\mathbf{x}_1, \mathbf{x}_1) & k(\mathbf{x}_1, \mathbf{x}_2) & \dots & k(\mathbf{x}_1, \mathbf{x}_n) \\ k(\mathbf{x}_2, \mathbf{x}_1) & k(\mathbf{x}_2, \mathbf{x}_2) & \dots & k(\mathbf{x}_2, \mathbf{x}_n) \\ \vdots & \vdots & \ddots & \vdots \\ k(\mathbf{x}_n, \mathbf{x}_1) & k(\mathbf{x}_n, \mathbf{x}_2) & \dots & k(\mathbf{x}_n, \mathbf{x}_n) \end{bmatrix}$$

Inference on held-out data

A common pattern:

Construct Gram matrix from kernel $k(\mathbf{x}, \mathbf{x}')$

$$\mathbf{M} = \begin{bmatrix} k(\mathbf{x}_1, \mathbf{x}_1) & k(\mathbf{x}_1, \mathbf{x}_2) & \dots & k(\mathbf{x}_1, \mathbf{x}_n) \\ k(\mathbf{x}_2, \mathbf{x}_1) & k(\mathbf{x}_2, \mathbf{x}_2) & \dots & k(\mathbf{x}_2, \mathbf{x}_n) \\ \vdots & \vdots & \ddots & \vdots \\ k(\mathbf{x}_n, \mathbf{x}_1) & k(\mathbf{x}_n, \mathbf{x}_2) & \dots & k(\mathbf{x}_n, \mathbf{x}_n) \end{bmatrix}$$

Use top/bottom eigenvectors of Gram matrix as embedding

Inference on held-out data

A common pattern:

Construct Gram matrix from kernel $k(\mathbf{x}, \mathbf{x}')$

$$\mathbf{M} = \begin{bmatrix} k(\mathbf{x}_1, \mathbf{x}_1) & k(\mathbf{x}_1, \mathbf{x}_2) & \dots & k(\mathbf{x}_1, \mathbf{x}_n) \\ k(\mathbf{x}_2, \mathbf{x}_1) & k(\mathbf{x}_2, \mathbf{x}_2) & \dots & k(\mathbf{x}_2, \mathbf{x}_n) \\ \vdots & \vdots & \ddots & \vdots \\ k(\mathbf{x}_n, \mathbf{x}_1) & k(\mathbf{x}_n, \mathbf{x}_2) & \dots & k(\mathbf{x}_n, \mathbf{x}_n) \end{bmatrix}$$

Use top/bottom eigenvectors of Gram matrix as embedding

Works for fixed dataset $\mathbf{x}_1, \dots, \mathbf{x}_n$ -
but what is the embedding vector for a new data point $\mathbf{x}'$?

Inference on held-out data

Assume data drawn $\mathbf{x}_1, \ldots, \mathbf{x}_n \sim p(\mathbf{x})$. Gram matrix is approximation to linear operator:

$$\frac{1}{n}\mathbf{M}\begin{pmatrix} f(\mathbf{x}_1) \\ f(\mathbf{x}_2) \\ \vdots \\ f(\mathbf{x}_n) \end{pmatrix}_i = \frac{1}{n}\sum_j k(\mathbf{x}_i, \mathbf{x}_j) f(\mathbf{x}_j) \approx \mathbb{E}_{p(\mathbf{x})}[k(\mathbf{x}_i, \mathbf{x}) f(\mathbf{x})]$$

$$= \mathcal{K}_p[f](\mathbf{x}_i)$$

Bengio, Paiement, Vincent et al 2004

Inference on held-out data

Assume data drawn $\mathbf{x}_1, \ldots, \mathbf{x}_n \sim p(\mathbf{x})$. Gram matrix is approximation to linear operator:

$$\begin{aligned}\frac{1}{n}\mathbf{M}\begin{pmatrix} f(\mathbf{x}_1) \\ f(\mathbf{x}_2) \\ \vdots \\ f(\mathbf{x}_n) \end{pmatrix}_i = \frac{1}{n}\sum_j k(\mathbf{x}_i, \mathbf{x}_j) f(\mathbf{x}_j) \quad &\approx \quad \mathbb{E}_{p(\mathbf{x})}[k(\mathbf{x}_i, \mathbf{x}) f(\mathbf{x})] \\ &= \quad \mathcal{K}_p[f](\mathbf{x}_i)\end{aligned}$$

Eigenvectors ϕ_k of $\mathbf{M}$ are approximation to eigenfunctions $\phi_k(\cdot)$ of linear operator $\mathcal{K}_p$

Bengio, Paiement, Vincent et al 2004

Inference on held-out data

Assume data drawn $\mathbf{x}_1, \ldots, \mathbf{x}_n \sim p(\mathbf{x})$. Gram matrix is approximation to linear operator:

$$\frac{1}{n}\mathbf{M}\begin{pmatrix} f(\mathbf{x}_1) \\ f(\mathbf{x}_2) \\ \vdots \\ f(\mathbf{x}_n) \end{pmatrix}_i = \frac{1}{n}\sum_j k(\mathbf{x}_i, \mathbf{x}_j) f(\mathbf{x}_j) \quad \approx \quad \mathbb{E}_{p(\mathbf{x})}[k(\mathbf{x}_i, \mathbf{x}) f(\mathbf{x})]$$

$$= \quad \mathcal{K}_p[f](\mathbf{x}_i)$$

From eigenvalues $\lambda_1, \ldots, \lambda_k$ and eigenvectors $\phi_1, \ldots, \phi_k$ of $\mathbf{M}$, can approximate eigenfunction of $\mathcal{K}_p$ with Nyström method:

$$\phi_k(\mathbf{x}') \propto \sum_i \phi_{ki} k(\mathbf{x}_i, \mathbf{x}')$$

Bengio, Paiement, Vincent et al 2004

Trade-offs of manifold learning

☺ Exactly solvable by eigendecomposition

Trade-offs of manifold learning

☺ Exactly solvable by eigendecomposition

☺ Data efficient (works with $\mathcal{O}(1000)$ data points)

Trade-offs of manifold learning

- ☺ Exactly solvable by eigendecomposition
- ☺ Data efficient (works with $\mathcal{O}(1000)$ data points)
- 😐 Unsupervised learning without a generative model

Trade-offs of manifold learning

- ☺ Exactly solvable by eigendecomposition
- ☺ Data efficient (works with $\mathcal{O}(1000)$ data points)
- 😐 Unsupervised learning without a generative model
- ☹ Learning scales as $\mathcal{O}(n \log n)$ with size of training data

Trade-offs of manifold learning

- ☺ Exactly solvable by eigendecomposition
- ☺ Data efficient (works with $\mathcal{O}(1000)$ data points)
- 😐 Unsupervised learning without a generative model
- ☹ Learning scales as $\mathcal{O}(n \log n)$ with size of training data
- ☹ Inference scales as $\mathcal{O}(n)$ with size of training data

Trade-offs of manifold learning

- ☺ Exactly solvable by eigendecomposition
- ☺ Data efficient (works with $\mathcal{O}(1000)$ data points)
- 😐 Unsupervised learning without a generative model
- ☹ Learning scales as $\mathcal{O}(n \log n)$ with size of training data
- ☹ Inference scales as $\mathcal{O}(n)$ with size of training data
- ☹ Performance degrades for noisy or clustered data

Trade-offs of manifold learning

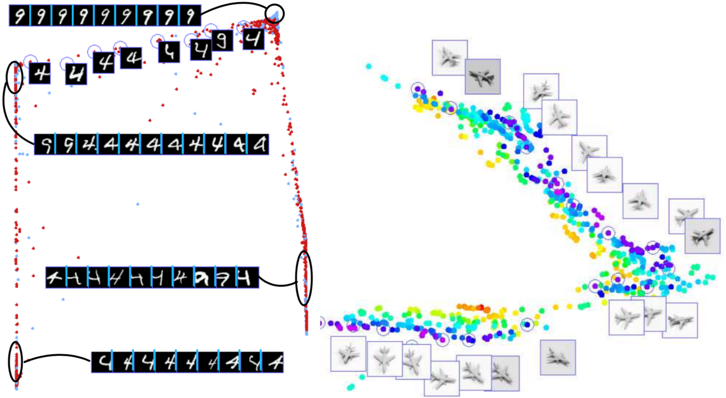
- ☺ Exactly solvable by eigendecomposition
- ☺ Data efficient (works with $\mathcal{O}(1000)$ data points)
- 😐 Unsupervised learning without a generative model
- ☹ Learning scales as $\mathcal{O}(n \log n)$ with size of training data
- ☹ Inference scales as $\mathcal{O}(n)$ with size of training data
- ☹ Performance degrades for noisy or clustered data
- ☹ Embeddings collapse on more complex data

Trade-offs of manifold learning

- ☺ Exactly solvable by eigendecomposition
- ☺ Data efficient (works with $\mathcal{O}(1000)$ data points)
- 😐 Unsupervised learning without a generative model
- ☹ Learning scales as $\mathcal{O}(n \log n)$ with size of training data
- ☹ Inference scales as $\mathcal{O}(n)$ with size of training data
- ☹ Performance degrades for noisy or clustered data
- ☹ Embeddings collapse on more complex data

Other methods have become popular for low-dimensional visualization - especially t-SNE (Maaten and Hinton 2008)

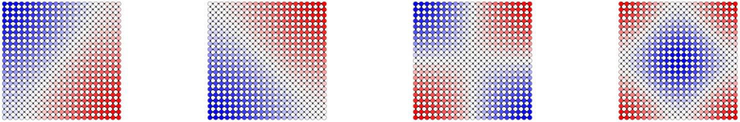
Collapsed embeddings



Hadsell, Chopra and LeCun 2006

Improving spectral embeddings

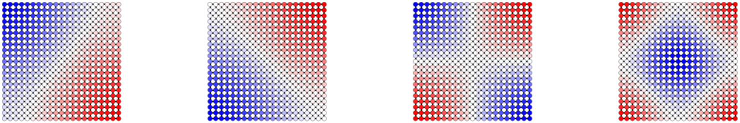
Bottom eigenfunctions of graph Laplacian of evenly-scaled grid are natural coordinates



Pfau and Burgess 2018

Improving spectral embeddings

Bottom eigenfunctions of graph Laplacian of evenly-scaled grid are natural coordinates

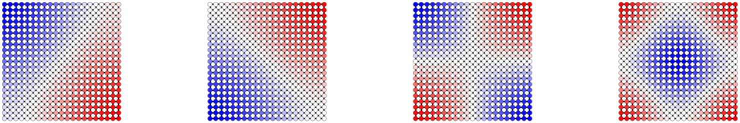


Bottom eigenfunctions of unevely-scaled grid are not

Pfau and Burgess 2018

Improving spectral embeddings

Bottom eigenfunctions of graph Laplacian of evenly-scaled grid are natural coordinates



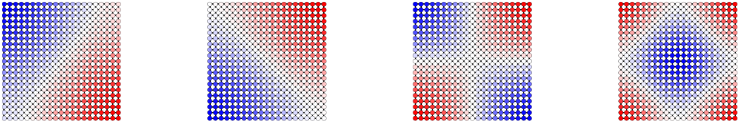
Bottom eigenfunctions of unevely-scaled grid are not

Eigenfunctions must be orthogonal, but can still be predictable, e.g. $\sin(2x)$ and $\sin(x)$ and $\cos(x)$.

Pfau and Burgess 2018

Improving spectral embeddings

Bottom eigenfunctions of graph Laplacian of evenly-scaled grid are natural coordinates



Bottom eigenfunctions of unevely-scaled grid are not

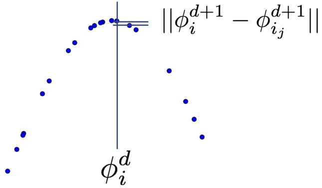
Eigenfunctions must be orthogonal, but can still be predictable, e.g. $\sin(2x)$ and $\sin(x)$ and $\cos(x)$.

Instead of using lowest eigenfunctions as embedding, use lowest eigenfunctions that are unpredictable from lower eigenfunctions

Pfau and Burgess 2018

Minimally redundant Laplacian eigenmaps

After adding eigenfunctions $\phi^1, \ldots, \phi^d$ to embedding, evaluate ϕ^{d+1} as candidate.

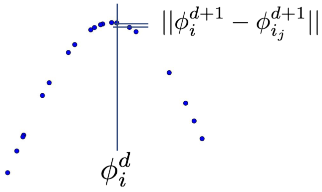


Pfau and Burgess 2018

Minimally redundant Laplacian eigenmaps

After adding eigenfunctions $\phi^1, \ldots, \phi^d$ to embedding, evaluate ϕ^{d+1} as candidate.

If value of ϕ^{d+1} at point i can be predicted from nearest neighbors of i in $\phi^1, \ldots, \phi^d$, eigenfunction is too predictable



Pfau and Burgess 2018

Minimally redundant Laplacian eigenmaps

NORB: Model dataset for studying invariance in object recognition

Pfau and Burgess 2018; LeCun, Huang and Bottou 2005

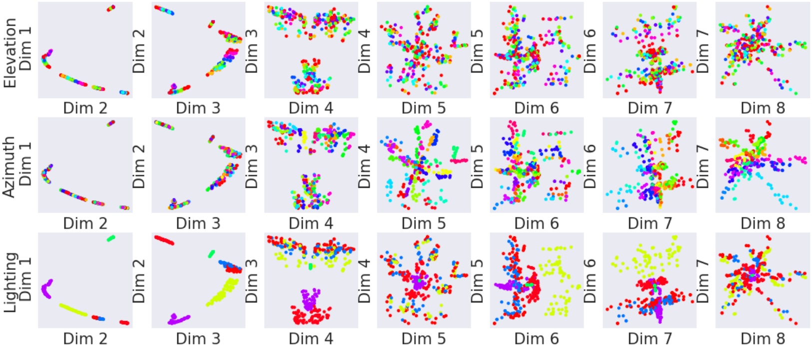
Minimally redundant Laplacian eigenmaps

NORB: Model dataset for studying invariance in object recognition

Consider a single object under different lighting and rotation

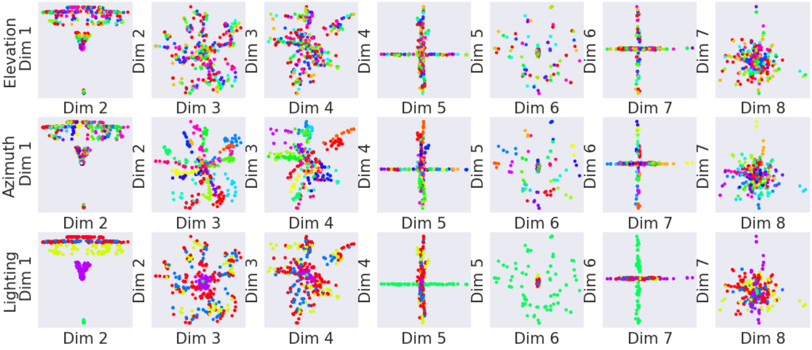
Pfau and Burgess 2018; LeCun, Huang and Bottou 2005

Minimally redundant Laplacian eigenmaps



Pfau and Burgess 2018

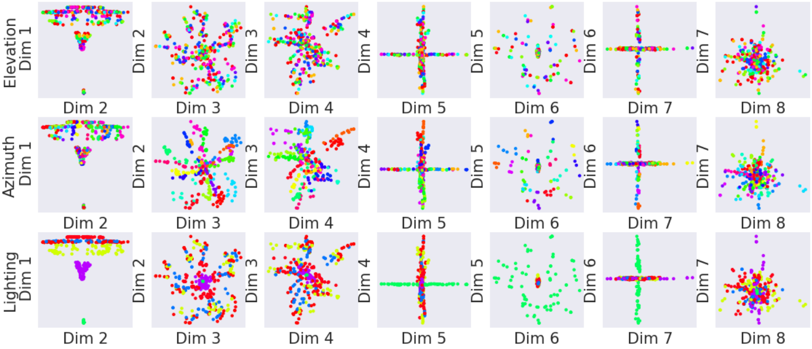
Minimally redundant Laplacian eigenmaps



With filtering by redundancy, all variation captured by 6 eigenfunctions

Pfau and Burgess 2018

Minimally redundant Laplacian eigenmaps

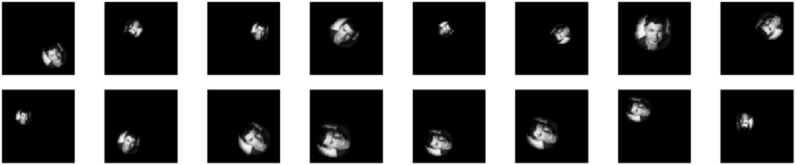


With filtering by redundancy, all variation captured by 6 eigenfunctions

Works with less than 1000 points!

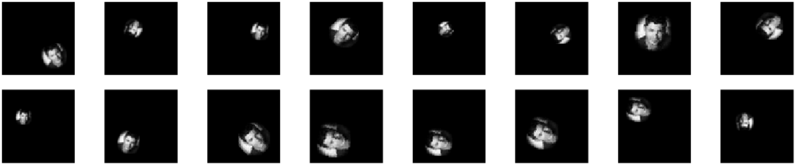
Pfau and Burgess 2018

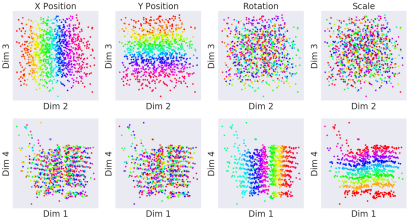
Minimally redundant Laplacian eigenmaps



Pfau and Burgess 2018

Minimally redundant Laplacian eigenmaps



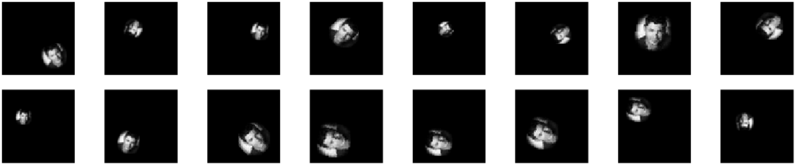


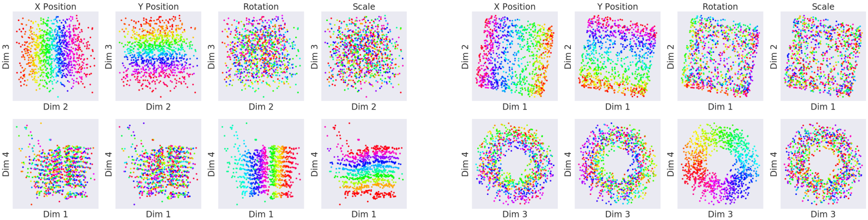
β-VAE

Disentangling VAEs learn the dimension but not the topology

Pfau and Burgess 2018

Minimally redundant Laplacian eigenmaps





β-VAE Laplacian eigenmaps

Disentangling VAEs learn the dimension but not the topology

Laplacian eigenmaps learns the topology but misses some dimensions

Pfau and Burgess 2018

Trade-offs of manifold learning

- ☺ Exactly solvable by eigendecomposition
- ☺ Data efficient (works with $\mathcal{O}(1000)$ data points)
- 😐 Unsupervised learning without a generative model
- ☹ Learning scales as $\mathcal{O}(n \log n)$ with size of training data
- ☹ Inference scales as $\mathcal{O}(n)$ with size of training data
- ☹ Performance degrades for noisy or clustered data
- ☹ Embeddings collapse on more complex data

Trade-offs of manifold learning

- ☺ Exactly solvable by eigendecomposition
- ☺ Data efficient (works with $\mathcal{O}(1000)$ data points)
- 😐 Unsupervised learning without a generative model
- ☹ Learning scales as $\mathcal{O}(n \log n)$ with size of training data
- ☹ Inference scales as $\mathcal{O}(n)$ with size of training data
- ☹ Performance degrades for noisy or clustered data
- ☺ Collapsed embeddings can be fixed by choice of eigenvector

Trade-offs of manifold learning

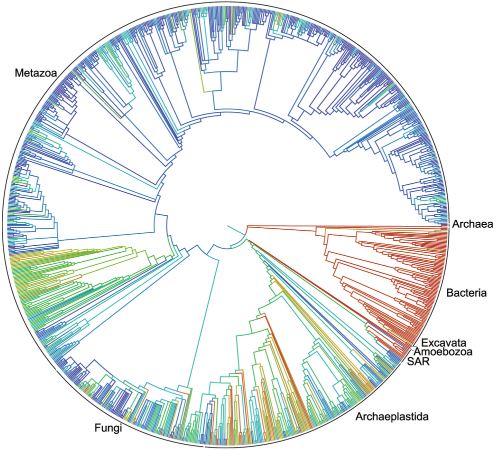
- ☺ Exactly solvable by eigendecomposition
- ☺ Data efficient (works with $\mathcal{O}(1000)$ data points)
- 😐 Unsupervised learning without a generative model
- ☹ Learning scales as $\mathcal{O}(n \log n)$ with size of training data
- ☹ Inference scales as $\mathcal{O}(n)$ with size of training data
- ☹ Performance degrades for noisy or clustered data
- ☺ Collapsed embeddings can be fixed by choice of eigenvector
- ☺ Can discover topology of data without prior assumptions

Embedding Hierarchies in Hyperbolic Spaces

Hierarchical data

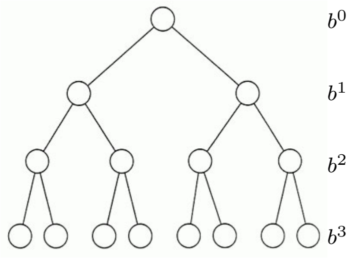
Some data can be embedded uniformly in flat space

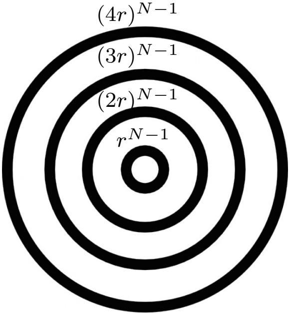
Hierarchical data



What about data with hierarchical structure?

Hierarchical data

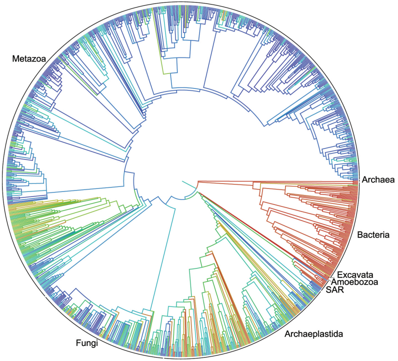


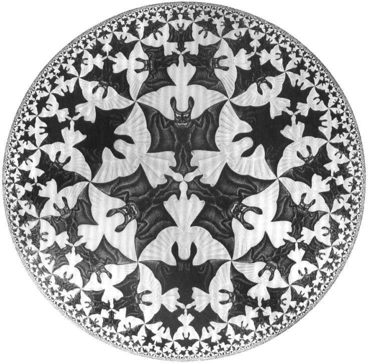


Tree with branching factor b has b^ℓ nodes at layer ℓ - exponential growth

Area of sphere in $\mathbb{R}^N$ with radius r grows as r^{N-1} - polynomial growth

Hierarchical data

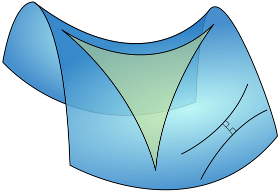




Idea: embed nodes in hierarchy in hyperbolic space instead of flat space

Hyperbolic geometry

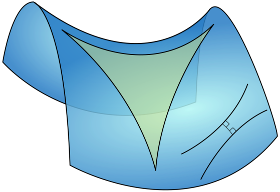
Hyperbolic space: manifold with constant negative curvature



Hyperbolic geometry

Hyperbolic space: manifold with constant negative curvature

Angles of triangle add to less than 180 degrees

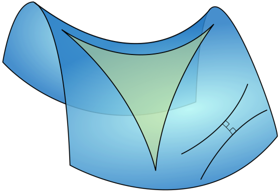


Hyperbolic geometry

Hyperbolic space: manifold with constant negative curvature

Angles of triangle add to less than 180 degrees

Surface area of spheres grows exponentially!



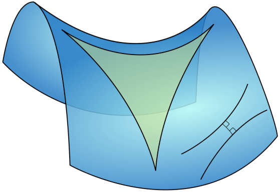
Hyperbolic geometry

Hyperbolic space: manifold with constant negative curvature

Angles of triangle add to less than 180 degrees

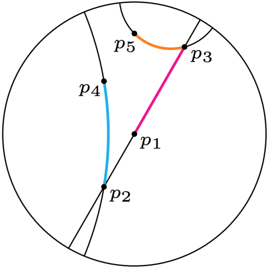
Surface area of spheres grows exponentially!

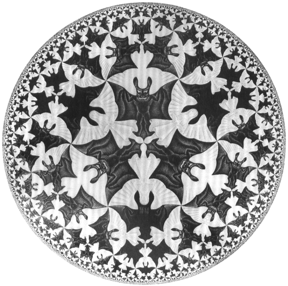
Many possible models of hyperbolic space



The Poincaré ball

Maps hyperbolic space to open ball $\mathbb{B}^N$



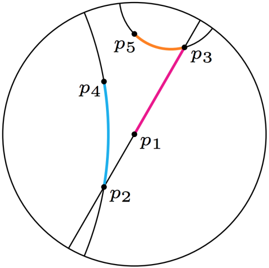


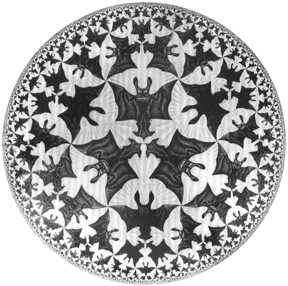
Nickel and Kiela 2017

The Poincaré ball

Maps hyperbolic space to open ball $\mathbb{B}^N$

Metric: $\langle \mathbf{u}, \mathbf{v} \rangle_{T_{\mathbf{x}}\mathcal{P}} = \left(\frac{2}{1-\mathbf{x}^T\mathbf{x}} \right)^2 \mathbf{u}^T \mathbf{v}$





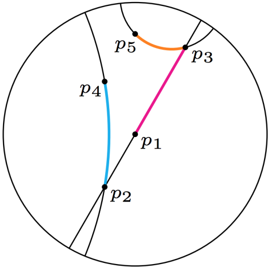
Nickel and Kiela 2017

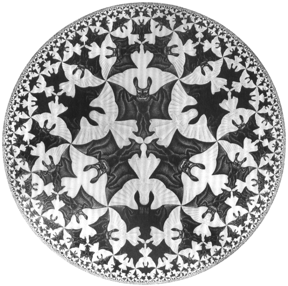
The Poincaré ball

Maps hyperbolic space to open ball $\mathbb{B}^N$

Metric: $\langle \mathbf{u}, \mathbf{v} \rangle_{T_{\mathbf{x}}\mathcal{P}} = \left(\frac{2}{1-\mathbf{x}^T\mathbf{x}} \right)^2 \mathbf{u}^T \mathbf{v}$

Geodesics: Circles that are orthogonal to boundary of ball





Nickel and Kiela 2017

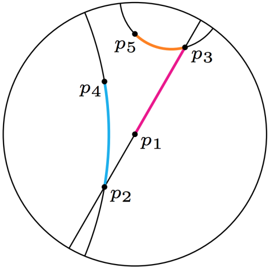
The Poincaré ball

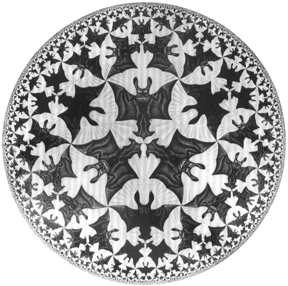
Maps hyperbolic space to open ball $\mathbb{B}^N$

Metric: $\langle \mathbf{u}, \mathbf{v} \rangle_{T_{\mathbf{x}}\mathcal{P}} = \left(\frac{2}{1-\mathbf{x}^T\mathbf{x}} \right)^2 \mathbf{u}^T \mathbf{v}$

Geodesics: Circles that are orthogonal to boundary of ball

Distances:
$d(\mathbf{u}, \mathbf{v}) = \operatorname{arcosh}\left(1 + 2\frac{||\mathbf{u}-\mathbf{v}||^2}{(1-||\mathbf{u}||^2)(1-||\mathbf{v}||^2)}\right)$





Nickel and Kiela 2017

Poincaré embeddings for learning hierarchical representation

Given edges $\mathcal{E}$ from a graph, find an embedding $\mathbf{u}_i$ for vertex i that minimizes:

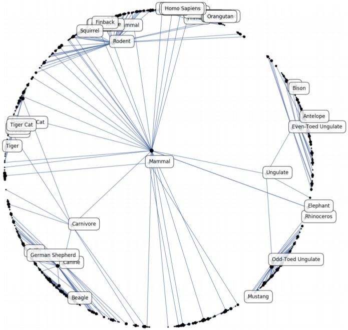
$$\sum_{i,j \in \mathcal{E}} \log \frac{e^{-d(\mathbf{u}_i, \mathbf{u}_j)}}{\sum_{j' \text{ st } ij' \notin \mathcal{E}} e^{-d(\mathbf{u}_i, \mathbf{u}_{j'})}}$$

Nickel and Kiela 2017

Poincaré embeddings for learning hierarchical representation

Given edges $\mathcal{E}$ from a graph, find an embedding $\mathbf{u}_i$ for vertex i that minimizes:

$$\sum_{i,j \in \mathcal{E}} \log \frac{e^{-d(\mathbf{u}_i, \mathbf{u}_j)}}{\sum_{j' \,\mathrm{st}\, ij' \notin \mathcal{E}} e^{-d(\mathbf{u}_i, \mathbf{u}_{j'})}}$$



Nickel and Kiela 2017

Poincaré embeddings for learning hierarchical representation

Given edges $\mathcal{E}$ from a graph, find an embedding $\mathbf{u}_i$ for vertex i that minimizes:

$$\sum_{i,j \in \mathcal{E}} \log \frac{e^{-d(\mathbf{u}_i, \mathbf{u}_j)}}{\sum_{j' \text{ st } ij' \notin \mathcal{E}} e^{-d(\mathbf{u}_i, \mathbf{u}_{j'})}}$$

			Dimensionality					
			5	10	20	50	100	200
WordNet Reconstruction	**Euclidean**	Rank	3542.3	2286.9	1685.9	1281.7	1187.3	1157.3
		MAP	0.024	0.059	0.087	0.140	0.162	0.168
	Translational	Rank	205.9	179.4	95.3	92.8	92.7	91.0
		MAP	0.517	0.503	0.563	0.566	0.562	0.565
	Poincaré	Rank	4.9	4.02	3.84	3.98	3.9	**3.83**
		MAP	0.823	0.851	0.855	0.86	0.857	**0.87**
WordNet Link Pred.	**Euclidean**	Rank	3311.1	2199.5	952.3	351.4	190.7	81.5
		MAP	0.024	0.059	0.176	0.286	0.428	0.490
	Translational	Rank	65.7	56.6	52.1	47.2	43.2	40.4
		MAP	0.545	0.554	0.554	0.56	0.562	0.559
	Poincaré	Rank	5.7	**4.3**	4.9	4.6	4.6	4.6
		MAP	0.825	0.852	0.861	**0.863**	0.856	0.855

State of the art results on link reconstruction and link prediction on WordNet noun dataset

Nickel and Kiela 2017

Poincaré embeddings for learning hierarchical representation

Given edges $\mathcal{E}$ from a graph, find an embedding $\mathbf{u}_i$ for vertex i that minimizes:

$$\sum_{i,j \in \mathcal{E}} \log \frac{e^{-d(\mathbf{u}_i, \mathbf{u}_j)}}{\sum_{j' \text{ st } ij' \notin \mathcal{E}} e^{-d(\mathbf{u}_i, \mathbf{u}_{j'})}}$$

Table 3: Spearman's ρ for Lexical Entailment on HYPERLEX.

	FR	**SLQS-Sim**	**WN-Basic**	**WN-WuP**	**WN-LCh**	**Vis-ID**	**Euclidean**	**Poincaré**
ρ	0.283	0.229	0.240	0.214	0.214	0.253	0.389	0.512

State of the art results on graded lexical entailment on HYPERLEX

Nickel and Kiela 2017

Trade-offs of Poincaré model

☺ Geodesics are simple

Trade-offs of Poincaré model

☺ Geodesics are simple

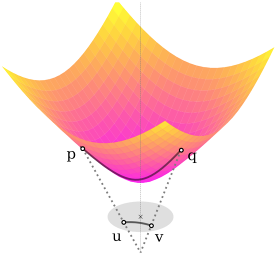
☺ Metric is diagonal

Trade-offs of Poincaré model

- ☺ Geodesics are simple
- ☺ Metric is diagonal
- ☹ Gradients of metric are unstable near boundary

The Lorentz model

Maps N-dim hyperbolic space to surface in $\mathbb{R}^{N+1}$

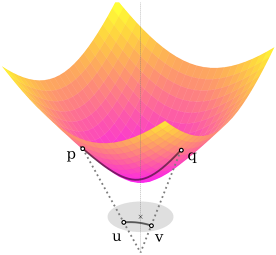


Nickel and Kiela 2018

The Lorentz model

Maps N-dim hyperbolic space to surface in $\mathbb{R}^{N+1}$

$\langle \mathbf{u}, \mathbf{v} \rangle_{\mathcal{L}} = -u_0 v_0 + \sum_{i=1}^{N+1} u_i v_i$



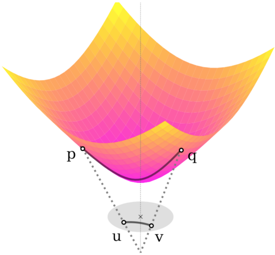
Nickel and Kiela 2018

The Lorentz model

Maps N-dim hyperbolic space to surface in $\mathbb{R}^{N+1}$

$\langle \mathbf{u}, \mathbf{v} \rangle_{\mathcal{L}} = -u_0 v_0 + \sum_{i=1}^{N+1} u_i v_i$

$\mathcal{H}^N = \{\mathbf{x} \in \mathbb{R}^{N+1} : \langle \mathbf{x}, \mathbf{x} \rangle_{\mathcal{L}} = -1, x_0 > 0\}$



Nickel and Kiela 2018

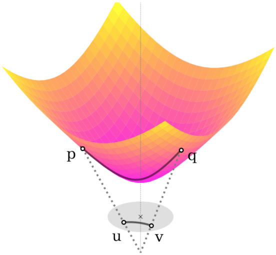
The Lorentz model

Maps N-dim hyperbolic space to surface in $\mathbb{R}^{N+1}$

$\langle \mathbf{u}, \mathbf{v} \rangle_{\mathcal{L}} = -u_0 v_0 + \sum_{i=1}^{N+1} u_i v_i$

$\mathcal{H}^N = \{\mathbf{x} \in \mathbb{R}^{N+1} : \langle \mathbf{x}, \mathbf{x} \rangle_{\mathcal{L}} = -1, x_0 > 0\}$

Metric: $\langle \mathbf{u}, \mathbf{v} \rangle_{T_{\mathbf{x}}\mathcal{L}} = \langle \mathbf{u}, \mathbf{v} \rangle_{\mathcal{L}}$



Nickel and Kiela 2018

The Lorentz model

Maps N-dim hyperbolic space to surface in $\mathbb{R}^{N+1}$

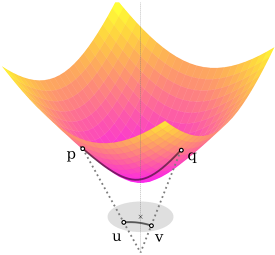
$\langle \mathbf{u}, \mathbf{v} \rangle_{\mathcal{L}} = -u_0 v_0 + \sum_{i=1}^{N+1} u_i v_i$

$\mathcal{H}^N = \{\mathbf{x} \in \mathbb{R}^{N+1} : \langle \mathbf{x}, \mathbf{x} \rangle_{\mathcal{L}} = -1, x_0 > 0\}$

Metric: $\langle \mathbf{u}, \mathbf{v} \rangle_{T_{\mathbf{x}}\mathcal{L}} = \langle \mathbf{u}, \mathbf{v} \rangle_{\mathcal{L}}$

Geodesics: starting at $\mathbf{x}$ and going in direction $\mathbf{v}$, $||\mathbf{v}||_{\mathcal{L}} = 1$

$$\exp_{\mathbf{x}}(t\mathbf{v}) = \cosh(t)\mathbf{x} + \sinh(t)\mathbf{v}$$



Nickel and Kiela 2018

The Lorentz model

Maps N-dim hyperbolic space to surface in $\mathbb{R}^{N+1}$

$\langle \mathbf{u}, \mathbf{v} \rangle_{\mathcal{L}} = -u_0 v_0 + \sum_{i=1}^{N+1} u_i v_i$

$\mathcal{H}^N = \{\mathbf{x} \in \mathbb{R}^{N+1} : \langle \mathbf{x}, \mathbf{x} \rangle_{\mathcal{L}} = -1, x_0 > 0\}$

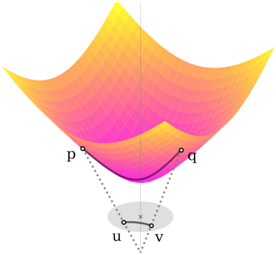
Metric: $\langle \mathbf{u}, \mathbf{v} \rangle_{T_{\mathbf{x}}\mathcal{L}} = \langle \mathbf{u}, \mathbf{v} \rangle_{\mathcal{L}}$

Geodesics: starting at $\mathbf{x}$ and going in direction $\mathbf{v}$, $||\mathbf{v}||_{\mathcal{L}} = 1$

$$\exp_{\mathbf{x}}(t\mathbf{v}) = \cosh(t)\mathbf{x} + \sinh(t)\mathbf{v}$$

Projection: maps vectors onto tangent space:

$$\mathrm{proj}_{\mathbf{x}}(\mathbf{u}) = \mathbf{u} + \langle \mathbf{x}, \mathbf{u} \rangle_{\mathcal{L}}$$



Nickel and Kiela 2018

Riemannian SGD

Scale Euclidean gradient by inverse metric:

$$\mathbf{h} = \mathbf{M}_{\mathbf{x}}^{-1} \nabla f(\mathbf{x})$$

Bonnabel 2013

Riemannian SGD

Scale Euclidean gradient by inverse metric:

$$\mathbf{h} = \mathbf{M}_{\mathbf{x}}^{-1} \nabla f(\mathbf{x})$$

Project gradient onto tangent space:

$$\mathrm{grad} f(\mathbf{x}) = \mathrm{proj}(\mathbf{h})$$

Bonnabel 2013

Riemannian SGD

Scale Euclidean gradient by inverse metric:

$$\mathbf{h} = \mathbf{M}_{\mathbf{x}}^{-1} \nabla f(\mathbf{x})$$

Project gradient onto tangent space:

$$\mathrm{grad} f(\mathbf{x}) = \mathrm{proj}(\mathbf{h})$$

Follow geodesic along Riemannian gradient:

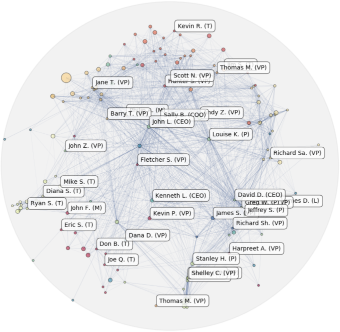
$$\mathbf{x} \leftarrow \exp_{\mathbf{x}}(-\eta \mathrm{grad} f(\mathbf{x}))$$

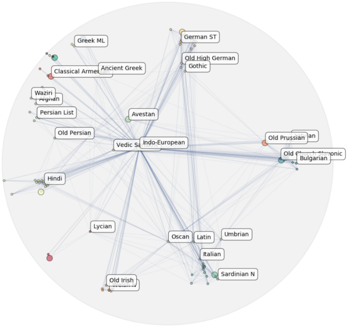
Bonnabel 2013

Learning continuous hierarchies in the Lorentz model

		WordNet Nouns			WordNet Verbs			EuroVoc			ACM			MeSH		
		2	5	10	2	5	10	2	5	10	2	5	10	2	5	10
MR	Poincaré	90.7	4.9	4.02	10.71	1.39	1.35	2.83	1.25	1.23	4.14	1.8	1.71	61.11	14.05	12.8
	Lorentz	22.8	3.18	2.95	3.64	1.26	1.23	1.63	1.24	1.17	3.05	1.67	1.63	38.99	14.13	12.42
	Δ%	74.8	35.1	36.2	66.0	9.6	8.9	42.4	6.1	3.4	26.3	7.2	4.8	36.2	-0.5	2.9
MAP	Poincaré	11.8	82.8	86.5	36.5	91.0	91.2	64.3	94.0	94.4	69.3	94.1	94.8	19.5	76.3	79.4
	Lorentz	30.5	92.3	92.8	57.9	93.5	93.3	87.1	95.8	96.5	82.9	96.6	97.0	34.8	77.7	79.9
	Δ%	61.3	10.3	6.8	58.6	2.7	2.3	35.6	1.6	2.0	19.6	2.7	2.3	43.9	1.8	0.6
ρ	Poincaré	13.8	57.2	58.5	11.0	54.1	55.1	37.5	57.5	61.4	59.8	63.5	62.9	42.2	69.9	74.9
	Lorentz	41.0	58.9	59.5	47.9	55.5	56.6	54.5	61.7	67.5	65.9	65.9	65.9	64.5	71.4	76.3

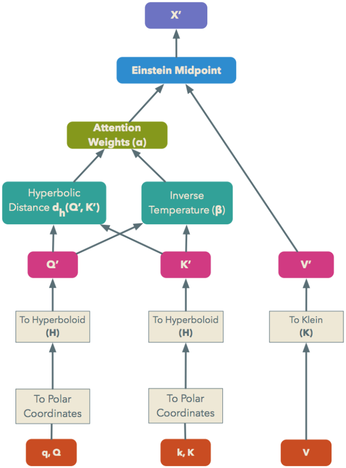
Significant improvement over Poincaré, especially in low dimensions





Nickel and Kiela 2018

Hyperbolic Attention Networks



Imposes hyperbolic geometry on activations of deep network with attention

Gulcehre, Denil, Malinowski et al 2018

Analyzing the Geometry of Deep Generative Models

What is the shape of latent space?

Higgins, Matthey, Pal et al 2017

What is the shape of latent space?

Are straight lines in latent space really straight?

Higgins, Matthey, Pal et al 2017

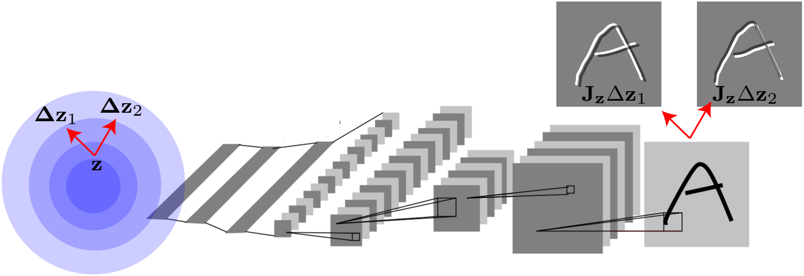
What is the shape of latent space?

Are straight lines in latent space really straight?

What is the right notion of distance in latent space?

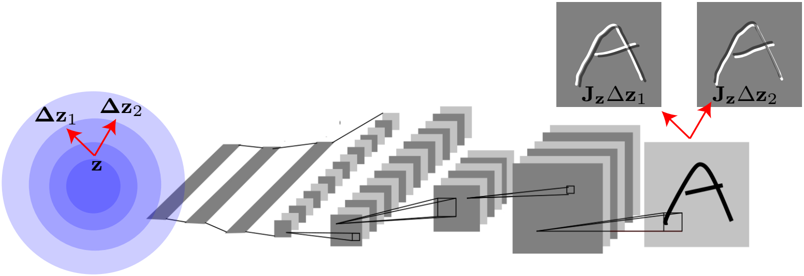
Higgins, Matthey, Pal et al 2017

Latent space metric



Deep generative model with decoder $f(\mathbf{z})$

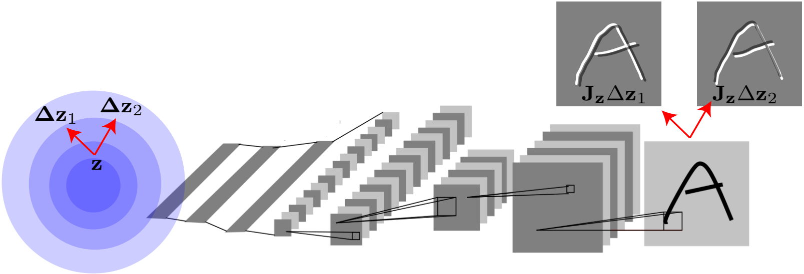
Latent space metric



Deep generative model with decoder $f(\mathbf{z})$

Manifold $\mathcal{X}$ generated by f has tangent space $T_{\mathbf{z}}\mathcal{X} = \mathrm{span}(\mathbf{J_z})$ where $\mathbf{J_z}$ is Jacobian of f at $\mathbf{z}$

Latent space metric



Deep generative model with decoder $f(\mathbf{z})$

Manifold $\mathcal{X}$ generated by f has tangent space $T_\mathbf{z}\mathcal{X} = \mathrm{span}(\mathbf{J_z})$ where $\mathbf{J_z}$ is Jacobian of f at $\mathbf{z}$

Use ℓ_2 metric in observation space as metric in latent space:

$$\langle \Delta\mathbf{z}_1, \Delta\mathbf{z}_2 \rangle_{T_\mathbf{z}\mathcal{X}} = \Delta\mathbf{z}_1^T \mathbf{J}_\mathbf{z}^T \mathbf{J_z} \Delta\mathbf{z}_2 = \Delta\mathbf{z}_1^T \mathbf{M_z} \Delta\mathbf{z}_2$$

Arvanitidis, Hansen and Hauberg 2018

Geodesic equation

How do we derive the geodesic equation from the introduction:

$$\ddot{\gamma} = -\mathbf{\Gamma}_{\gamma(t)} \left(\dot{\gamma}(t), \dot{\gamma}(t)\right)$$

Geodesic equation

How do we derive the geodesic equation from the introduction:

$$\ddot{\gamma} = -\mathbf{\Gamma}_{\gamma(t)}\left(\dot{\gamma}(t), \dot{\gamma}(t)\right)$$

$$\gamma^* = \min_{\gamma} \mathcal{S}[\gamma] \quad = \quad \min_{\gamma} \int_0^1 dt L(\gamma, \dot{\gamma}, t)$$

Geodesic equation

How do we derive the geodesic equation from the introduction:

$$\ddot{\gamma} = -\mathbf{\Gamma}_{\gamma(t)}\left(\dot{\gamma}(t), \dot{\gamma}(t)\right)$$

$$\begin{aligned}
\gamma^* = \min_{\gamma} \mathcal{S}[\gamma] &= \min_{\gamma} \int_0^1 dt L(\gamma, \dot{\gamma}, t) \\
&= \min_{\gamma} \int_0^1 dt \left\langle \dot{\gamma(t)}, \dot{\gamma(t)} \right\rangle_{T_{\gamma(t)}\mathcal{X}}
\end{aligned}$$

Geodesic equation

How do we derive the geodesic equation from the introduction:

$$\ddot{\gamma} = -\mathbf{\Gamma}_{\gamma(t)}\left(\dot{\gamma}(t), \dot{\gamma}(t)\right)$$

$$\begin{aligned}
\gamma^* = \min_{\gamma} \mathcal{S}[\gamma] &= \min_{\gamma} \int_0^1 dt L(\gamma, \dot{\gamma}, t) \\
&= \min_{\gamma} \int_0^1 dt \left\langle \dot{\gamma(t)}, \dot{\gamma(t)} \right\rangle_{T_{\gamma(t)}\mathcal{X}} \\
&= \min_{\gamma} \int_0^1 dt \sum_{ij} \mathbf{M}^{ij}_{\gamma(t)} \dot{\gamma}_i(t) \dot{\gamma}_j(t)
\end{aligned}$$

Geodesic equation

How do we derive the geodesic equation from the introduction:

$$\ddot{\gamma} = -\mathbf{\Gamma}_{\gamma(t)}\left(\dot{\gamma}(t), \dot{\gamma}(t)\right)$$

$$\begin{aligned}
\gamma^* = \min_\gamma \mathcal{S}[\gamma] &= \min_\gamma \int_0^1 dt L(\gamma, \dot{\gamma}, t) \\
&= \min_\gamma \int_0^1 dt \left\langle \dot{\gamma(t)}, \dot{\gamma(t)} \right\rangle_{T_{\gamma(t)}\mathcal{X}} \\
&= \min_\gamma \int_0^1 dt \sum_{ij} \mathbf{M}^{ij}_{\gamma(t)} \dot{\gamma}_i(t) \dot{\gamma}_j(t)
\end{aligned}$$

Euler-Lagrange equation:

$$\frac{\partial L}{\partial \gamma^*} = \frac{d}{dt} \frac{\partial L}{\partial \dot{\gamma}^*}$$

Geodesic equation

How do we derive the geodesic equation from the introduction:

$$\ddot{\gamma} = -\mathbf{\Gamma}_{\gamma(t)}\left(\dot{\gamma}(t), \dot{\gamma}(t)\right)$$

$$\begin{aligned}
\gamma^* = \min_\gamma \mathcal{S}[\gamma] &= \min_\gamma \int_0^1 dt L(\gamma, \dot{\gamma}, t) \\
&= \min_\gamma \int_0^1 dt \left\langle \dot{\gamma(t)}, \dot{\gamma(t)} \right\rangle_{T_{\gamma(t)}\mathcal{X}} \\
&= \min_\gamma \int_0^1 dt \sum_{ij} \mathbf{M}^{ij}_{\gamma(t)} \dot{\gamma}_i(t) \dot{\gamma}_j(t)
\end{aligned}$$

Euler-Lagrange equation:

$$\frac{\partial L}{\partial \gamma^*} = \frac{d}{dt} \frac{\partial L}{\partial \dot{\gamma}^*}$$

Exercise: Derive the left and right side of the Euler-Lagrange equation for $L(\gamma, \dot{\gamma}, t) = \sum_{ij} \mathbf{M}^{ij}_{\gamma(t)} \dot{\gamma}_i(t) \dot{\gamma}_j(t)$

Geodesic equation

$$\frac{\partial L}{\partial \gamma_i} = \sum_{jk} \frac{\partial \mathbf{M}^{jk}_{\gamma(t)}}{\partial \gamma_i} \dot{\gamma}_k(t) \dot{\gamma}_j(t)$$

Geodesic equation

$$\frac{\partial L}{\partial \gamma_i} = \sum_{jk} \frac{\partial \mathbf{M}^{jk}_{\gamma(t)}}{\partial \gamma_i} \dot{\gamma}_k(t) \dot{\gamma}_j(t)$$

$$\begin{aligned}
\frac{d}{dt}\frac{\partial L}{\partial \dot{\gamma}_i} = \frac{d}{dt}\left(2\sum_j \mathbf{M}^{ij}_{\gamma(t)} \dot{\gamma}_j(t)\right) \quad &= \quad 2\sum_j \left[\frac{\partial \mathbf{M}^{ij}_{\gamma(t)}}{\partial t} \dot{\gamma}_j(t) + \mathbf{M}^{ij}_{\gamma(t)} \ddot{\gamma}_j(t)\right] \\
&= \quad 2\sum_j \left[\sum_k \frac{\partial \mathbf{M}^{ij}_{\gamma(t)}}{\partial \gamma_k} \dot{\gamma}_k(t) \dot{\gamma}_j(t) + \mathbf{M}^{ij}_{\gamma(t)} \ddot{\gamma}_j(t)\right]
\end{aligned}$$

Geodesic equation

$$\frac{\partial L}{\partial \gamma_i} = \sum_{jk} \frac{\partial \mathbf{M}^{jk}_{\gamma(t)}}{\partial \gamma_i} \dot{\gamma}_k(t) \dot{\gamma}_j(t)$$

$$\begin{aligned}
\frac{d}{dt} \frac{\partial L}{\partial \dot{\gamma}_i} = \frac{d}{dt} \left(2 \sum_j \mathbf{M}^{ij}_{\gamma(t)} \dot{\gamma}_j(t) \right) \quad &= \quad 2 \sum_j \left[\frac{\partial \mathbf{M}^{ij}_{\gamma(t)}}{\partial t} \dot{\gamma}_j(t) + \mathbf{M}^{ij}_{\gamma(t)} \ddot{\gamma}_j(t) \right] \\
&= \quad 2 \sum_j \left[\sum_k \frac{\partial \mathbf{M}^{ij}_{\gamma(t)}}{\partial \gamma_k} \dot{\gamma}_k(t) \dot{\gamma}_j(t) + \mathbf{M}^{ij}_{\gamma(t)} \ddot{\gamma}_j(t) \right]
\end{aligned}$$

$$\boxed{\ddot{\gamma}_j(t) = -\tfrac{1}{2} \sum_i \left(\mathbf{M}^{-1}_{\gamma(t)} \right)^{ij} \sum_k \left[\left(2 \tfrac{\partial \mathbf{M}^{ij}_{\gamma(t)}}{\partial \gamma_k} - \tfrac{\partial \mathbf{M}^{jk}_{\gamma(t)}}{\partial \gamma_i} \right) \dot{\gamma}_k(t) \dot{\gamma}_j(t) \right]}$$

Geodesic equation

$$\frac{\partial L}{\partial \gamma_i} = \sum_{jk} \frac{\partial \mathbf{M}^{jk}_{\gamma(t)}}{\partial \gamma_i} \dot{\gamma}_k(t) \dot{\gamma}_j(t)$$

$$\begin{aligned}
\frac{d}{dt}\frac{\partial L}{\partial \dot{\gamma}_i} = \frac{d}{dt}\left(2\sum_j \mathbf{M}^{ij}_{\gamma(t)} \dot{\gamma}_j(t)\right) \quad &= \quad 2\sum_j \left[\frac{\partial \mathbf{M}^{ij}_{\gamma(t)}}{\partial t} \dot{\gamma}_j(t) + \mathbf{M}^{ij}_{\gamma(t)} \ddot{\gamma}_j(t)\right] \\
&= \quad 2\sum_j \left[\sum_k \frac{\partial \mathbf{M}^{ij}_{\gamma(t)}}{\partial \gamma_k} \dot{\gamma}_k(t) \dot{\gamma}_j(t) + \mathbf{M}^{ij}_{\gamma(t)} \ddot{\gamma}_j(t)\right]
\end{aligned}$$

$$\boxed{\begin{gathered}
\ddot{\gamma}_j(t) = -\textstyle\sum_{ik} \mathbf{\Gamma}^{ik}_j(\gamma(t)) \dot{\gamma}_k(t) \dot{\gamma}_j(t) \\
\mathbf{\Gamma}^{ik}_j(x) = \frac{1}{2}\left(\mathbf{M}^{-1}_x\right)^{ij}\left(2\frac{\partial \mathbf{M}^{ij}_x}{\partial x_k} - \frac{\partial \mathbf{M}^{jk}_x}{\partial x_i}\right)
\end{gathered}}$$

Geodesic equation

$$\frac{\partial L}{\partial \gamma_i} = \sum_{jk} \frac{\partial \mathbf{M}^{jk}_{\gamma(t)}}{\partial \gamma_i} \dot{\gamma}_k(t) \dot{\gamma}_j(t)$$

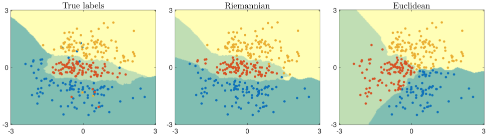
$$\begin{aligned} \frac{d}{dt}\frac{\partial L}{\partial \dot{\gamma}_i} = \frac{d}{dt}\left(2\sum_j \mathbf{M}^{ij}_{\gamma(t)} \dot{\gamma}_j(t)\right) &= 2\sum_j \left[\frac{\partial \mathbf{M}^{ij}_{\gamma(t)}}{\partial t}\dot{\gamma}_j(t) + \mathbf{M}^{ij}_{\gamma(t)}\ddot{\gamma}_j(t)\right] \\ &= 2\sum_j \left[\sum_k \frac{\partial \mathbf{M}^{ij}_{\gamma(t)}}{\partial \gamma_k}\dot{\gamma}_k(t)\dot{\gamma}_j(t) + \mathbf{M}^{ij}_{\gamma(t)}\ddot{\gamma}_j(t)\right] \end{aligned}$$

$$\ddot{\gamma}_j(t) = -\textstyle\sum_{ik} \mathbf{\Gamma}^{ik}_j(\gamma(t))\dot{\gamma}_k(t)\dot{\gamma}_j(t)$$

$$\mathbf{\Gamma}^{ik}_j(x) = \tfrac{1}{2}\left(\mathbf{M}^{-1}_x\right)^{ij}\left(2\tfrac{\partial \mathbf{M}^{ij}_x}{\partial x_k} - \tfrac{\partial \mathbf{M}^{jk}_x}{\partial x_i}\right)$$

$\mathbf{\Gamma}^{ik}_j(x)$ are the parameters of the Levi-Civita connection: unique connection induced by the metric

The geodesics of deep generative models



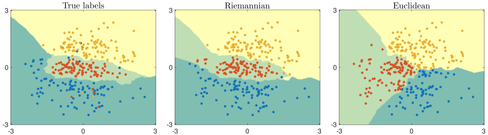
Digits	Linear	Riemannian
$\{0, 1, 2\}$	$77.57(\pm 0.87)\%$	$\mathbf{94.28}(\pm\mathbf{1.14})\%$
$\{3, 4, 7\}$	$77.80(\pm 0.91)\%$	$\mathbf{89.54}(\pm\mathbf{1.61})\%$
$\{5, 6, 9\}$	$64.93(\pm 0.96)\%$	$\mathbf{81.13}(\pm\mathbf{2.52})\%$

Table 1: The F-measure results for k-means.

Classification from latent representations is better

Arvanitidis, Hansen and Hauberg 2018

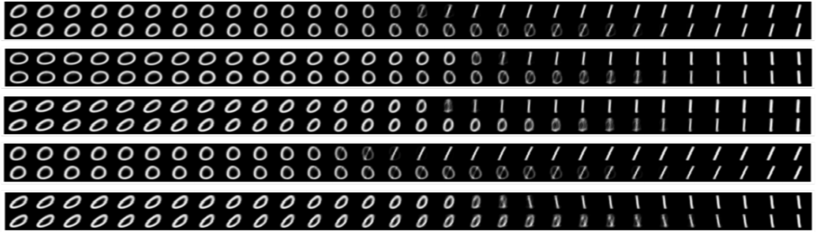
The geodesics of deep generative models



Digits	Linear	Riemannian
$\{0,1,2\}$	$77.57(\pm 0.87)\%$	$\mathbf{94.28}(\pm \mathbf{1.14})\%$
$\{3,4,7\}$	$77.80(\pm 0.91)\%$	$\mathbf{89.54}(\pm \mathbf{1.61})\%$
$\{5,6,9\}$	$64.93(\pm 0.96)\%$	$\mathbf{81.13}(\pm \mathbf{2.52})\%$

Table 1: The F-measure results for k-means.

Classification from latent representations is better



Transitions along geodesics are smoother

Arvanitidis, Hansen and Hauberg 2018

Summary

Differential geometry and spectral theory are a powerful suite of tools for thinking about the geometry of data

Summary

Differential geometry and spectral theory are a powerful suite of tools for thinking about the geometry of data

Straight lines generalize to geodesics, which are defined from the local metric structure

Summary

Differential geometry and spectral theory are a powerful suite of tools for thinking about the geometry of data

Straight lines generalize to geodesics, which are defined from the local metric structure

The spectrum of the Laplacian generalizes Fourier analysis to graphs and manifolds

Summary

Differential geometry and spectral theory are a powerful suite of tools for thinking about the geometry of data

Straight lines generalize to geodesics, which are defined from the local metric structure

The spectrum of the Laplacian generalizes Fourier analysis to graphs and manifolds

Classic manifold learning uses spectral decompositions to map data manifolds in high-dim space to flat space

Summary

Differential geometry and spectral theory are a powerful suite of tools for thinking about the geometry of data

Straight lines generalize to geodesics, which are defined from the local metric structure

The spectrum of the Laplacian generalizes Fourier analysis to graphs and manifolds

Classic manifold learning uses spectral decompositions to map data manifolds in high-dim space to flat space

Hierarchical data is naturally mapped to hyperbolic space

Summary

Differential geometry and spectral theory are a powerful suite of tools for thinking about the geometry of data

Straight lines generalize to geodesics, which are defined from the local metric structure

The spectrum of the Laplacian generalizes Fourier analysis to graphs and manifolds

Classic manifold learning uses spectral decompositions to map data manifolds in high-dim space to flat space

Hierarchical data is naturally mapped to hyperbolic space

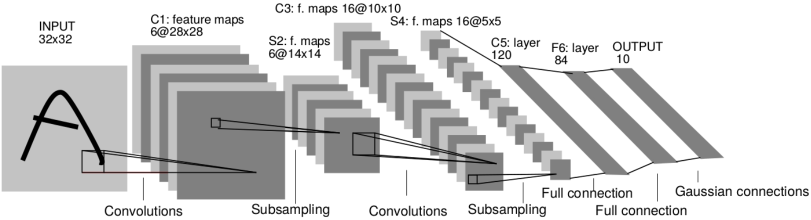
The latent space of deep generative models is better understood as being curved

Part III

Spectral Deep Learning

Convolutions on Graphs and Manifolds

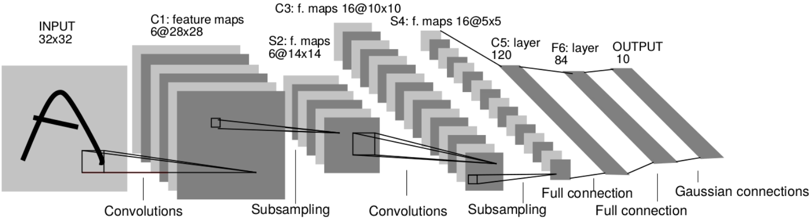
Key properties of CNNs



☺ Convolutional **(Translation invariance)**

LeCun et al. 1989

Key properties of CNNs

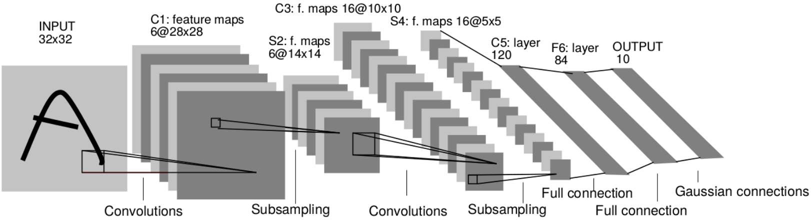


☺ Convolutional **(Translation invariance)**

☺ Scale Separation **(Compositionality)**

LeCun et al. 1989

Key properties of CNNs



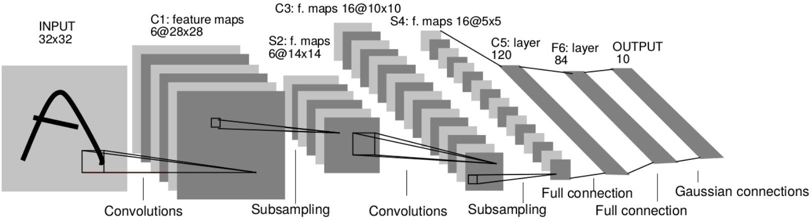
☺ Convolutional **(Translation invariance)**

☺ Scale Separation **(Compositionality)**

☺ Filters localized in space **(Deformation Stability)**

LeCun et al. 1989

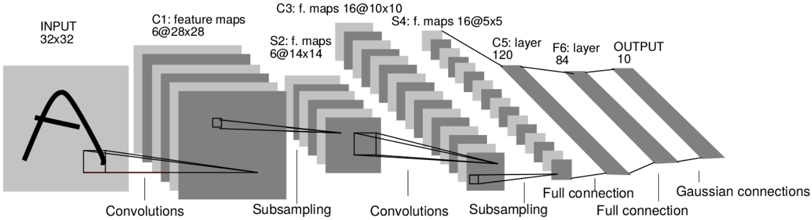
Key properties of CNNs



- ☺ Convolutional **(Translation invariance)**
- ☺ Scale Separation **(Compositionality)**
- ☺ Filters localized in space **(Deformation Stability)**
- ☺ $\mathcal{O}(1)$ parameters per filter (independent of input image size n)

LeCun et al. 1989

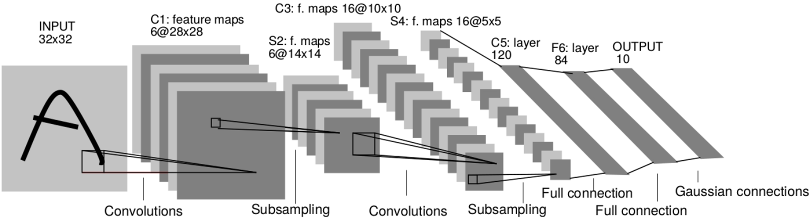
Key properties of CNNs



- ☺ Convolutional **(Translation invariance)**
- ☺ Scale Separation **(Compositionality)**
- ☺ Filters localized in space **(Deformation Stability)**
- ☺ $\mathcal{O}(1)$ parameters per filter (independent of input image size n)
- ☺ $\mathcal{O}(n)$ complexity per layer (filtering done in the spatial domain)

LeCun et al. 1989

Key properties of CNNs



- ☺ Convolutional **(Translation invariance)**
- ☺ Scale Separation **(Compositionality)**
- ☺ Filters localized in space **(Deformation Stability)**
- ☺ $\mathcal{O}(1)$ parameters per filter (independent of input image size n)
- ☺ $\mathcal{O}(n)$ complexity per layer (filtering done in the spatial domain)
- ☺ $\mathcal{O}(\log n)$ layers in classification tasks

LeCun et al. 1989

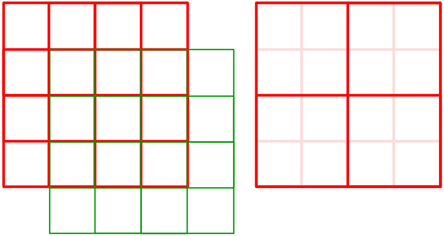
CNNs and Euclidean Geometry

CNNs are defined over Euclidean domains or Grids Ω. Two fundamental properties:

Translation Invariance (yielding convolutions).

Multiscale structure (yielding downsampling).

Inductive bias that exploits stationarity and deformation stability of many tasks.



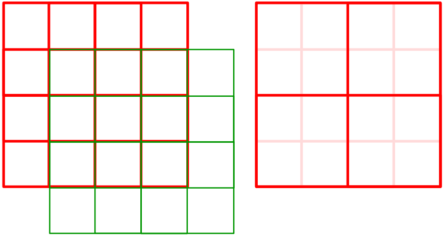
CNNs and Euclidean Geometry

CNNs are defined over Euclidean domains or Grids Ω. Two fundamental properties:

Translation Invariance (yielding convolutions).

Multiscale structure (yielding downsampling).

Inductive bias that exploits stationarity and deformation stability of many tasks.



Roadmap: extend CNNs to non-Euclidean geometries by replacing filtering and pooling by appropriate operators

Convolution: Euclidean space

Given two functions $f, g : [-\pi, \pi] \to \mathbb{R}$ their convolution is a function

$$(f \star g)(x) = \int_{-\pi}^{\pi} f(x')g(x - x')dx'$$

Convolution: Euclidean space

Given two functions $f, g : [-\pi, \pi] \to \mathbb{R}$ their convolution is a function

$$(f \star g)(x) = \int_{-\pi}^{\pi} f(x')g(x - x')dx'$$

Shift-invariance: $f(x - x_0) \star g(x) = (f \star g)(x - x_0)$

Convolution: Euclidean space

Given two functions $f, g : [-\pi, \pi] \to \mathbb{R}$ their convolution is a function

$$(f \star g)(x) = \int_{-\pi}^{\pi} f(x')g(x - x')dx'$$

Shift-invariance: $f(x - x_0) \star g(x) = (f \star g)(x - x_0)$

Convolution theorem: Fourier transform diagonalizes the convolution operator

Convolution: Euclidean space

Given two functions $f, g : [-\pi, \pi] \to \mathbb{R}$ their convolution is a function

$$(f \star g)(x) = \int_{-\pi}^{\pi} f(x')g(x - x')dx'$$

Shift-invariance: $f(x - x_0) \star g(x) = (f \star g)(x - x_0)$

Convolution theorem: Fourier transform diagonalizes the convolution operator
$\Rightarrow$ convolution can be computed in the Fourier domain as

$$\widehat{(f \star g)} = \hat{f} \cdot \hat{g}$$

Convolution: Euclidean space

Given two functions $f, g : [-\pi, \pi] \to \mathbb{R}$ their convolution is a function

$$(f \star g)(x) = \int_{-\pi}^{\pi} f(x')g(x - x')dx'$$

Shift-invariance: $f(x - x_0) \star g(x) = (f \star g)(x - x_0)$

Convolution theorem: Fourier transform diagonalizes the convolution operator
$\Rightarrow$ convolution can be computed in the Fourier domain as

$$\widehat{(f \star g)} = \hat{f} \cdot \hat{g}$$

Efficient computation using FFT

Convolution Theorem

Convolution of two vectors $\mathbf{f} = (f_1, \ldots, f_n)^\top$ and $\mathbf{g} = (g_1, \ldots, g_n)^\top$

$$\mathbf{f} \star \mathbf{g} \quad = \quad \begin{bmatrix} g_1 & g_2 & \cdots & \cdots & g_n \\ g_n & g_1 & g_2 & \cdots & g_{n-1} \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ g_3 & g_4 & \cdots & g_1 & g_2 \\ g_2 & g_3 & \cdots & \cdots & g_1 \end{bmatrix} \begin{bmatrix} f_1 \\ \vdots \\ f_n \end{bmatrix}$$

Convolution Theorem

Convolution of two vectors $\mathbf{f} = (f_1, \ldots, f_n)^\top$ and $\mathbf{g} = (g_1, \ldots, g_n)^\top$

$$\mathbf{f} \star \mathbf{g} \quad = \quad \underbrace{\begin{bmatrix} g_1 & g_2 & \cdots & \cdots & g_n \\ g_n & g_1 & g_2 & \cdots & g_{n-1} \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ g_3 & g_4 & \cdots & g_1 & g_2 \\ g_2 & g_3 & \cdots & \cdots & g_1 \end{bmatrix}}_{\text{circulant matrix}} \begin{bmatrix} f_1 \\ \vdots \\ f_n \end{bmatrix}$$

Convolution Theorem

Convolution of two vectors $\mathbf{f} = (f_1, \ldots, f_n)^\top$ and $\mathbf{g} = (g_1, \ldots, g_n)^\top$

$$\mathbf{f} \star \mathbf{g} \quad = \quad \underbrace{\begin{bmatrix} g_1 & g_2 & \cdots & \cdots & g_n \\ g_n & g_1 & g_2 & \cdots & g_{n-1} \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ g_3 & g_4 & \cdots & g_1 & g_2 \\ g_2 & g_3 & \cdots & \cdots & g_1 \end{bmatrix}}_{\text{diagonalized by Fourier basis}} \begin{bmatrix} f_1 \\ \vdots \\ f_n \end{bmatrix}$$

Convolution Theorem

Convolution of two vectors $\mathbf{f} = (f_1, \ldots, f_n)^\top$ and $\mathbf{g} = (g_1, \ldots, g_n)^\top$

$$\mathbf{f} \star \mathbf{g} = \begin{bmatrix} g_1 & g_2 & \cdots & \cdots & g_n \\ g_n & g_1 & g_2 & \cdots & g_{n-1} \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ g_3 & g_4 & \cdots & g_1 & g_2 \\ g_2 & g_3 & \cdots & \cdots & g_1 \end{bmatrix} \begin{bmatrix} f_1 \\ \vdots \\ f_n \end{bmatrix}$$

$$= \mathbf{\Phi} \begin{bmatrix} \hat{g}_1 & & \\ & \ddots & \\ & & \hat{g}_n \end{bmatrix} \mathbf{\Phi}^\top \mathbf{f}$$

Convolution Theorem

Convolution of two vectors $\mathbf{f} = (f_1, \ldots, f_n)^\top$ and $\mathbf{g} = (g_1, \ldots, g_n)^\top$

$$\begin{aligned}
\mathbf{f} \star \mathbf{g} \quad &= \quad \begin{bmatrix} g_1 & g_2 & \cdots & \cdots & g_n \\ g_n & g_1 & g_2 & \cdots & g_{n-1} \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ g_3 & g_4 & \cdots & g_1 & g_2 \\ g_2 & g_3 & \cdots & \cdots & g_1 \end{bmatrix} \begin{bmatrix} f_1 \\ \vdots \\ f_n \end{bmatrix} \\
&= \quad \mathbf{\Phi} \begin{bmatrix} \hat{g}_1 & & \\ & \ddots & \\ & & \hat{g}_n \end{bmatrix} \begin{bmatrix} \hat{f}_1 \\ \vdots \\ \hat{f}_n \end{bmatrix}
\end{aligned}$$

Convolution Theorem

Convolution of two vectors $\mathbf{f} = (f_1, \ldots, f_n)^\top$ and $\mathbf{g} = (g_1, \ldots, g_n)^\top$

$$\mathbf{f} \star \mathbf{g} = \begin{bmatrix} g_1 & g_2 & \cdots & \cdots & g_n \\ g_n & g_1 & g_2 & \cdots & g_{n-1} \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ g_3 & g_4 & \cdots & g_1 & g_2 \\ g_2 & g_3 & \cdots & \cdots & g_1 \end{bmatrix} \begin{bmatrix} f_1 \\ \vdots \\ f_n \end{bmatrix}$$

$$= \boldsymbol{\Phi} \begin{bmatrix} \hat{f}_1 \cdot \hat{g}_1 \\ \vdots \\ \hat{f}_n \cdot \hat{g}_n \end{bmatrix}$$

Spectral convolution

Spectral convolution of $\mathbf{f}, \mathbf{g} \in L^2(\mathcal{V})$ can be defined by analogy

$$\mathbf{f} \star \mathbf{g} \quad = \quad \sum_{k \geq 1} \langle \mathbf{f}, \boldsymbol{\phi}_k \rangle_{L^2(\mathcal{V})} \langle \mathbf{g}, \boldsymbol{\phi}_k \rangle_{L^2(\mathcal{V})} \, \boldsymbol{\phi}_k$$

Spectral convolution

Spectral convolution of $\mathbf{f}, \mathbf{g} \in L^2(\mathcal{V})$ can be defined by analogy

$$\mathbf{f} \star \mathbf{g} \quad = \quad \sum_{k \geq 1} \underbrace{\langle \mathbf{f}, \boldsymbol{\phi}_k \rangle_{L^2(\mathcal{V})} \langle \mathbf{g}, \boldsymbol{\phi}_k \rangle_{L^2(\mathcal{V})}}_{\text{product in the Fourier domain}} \boldsymbol{\phi}_k$$

Spectral convolution

Spectral convolution of $\mathbf{f}, \mathbf{g} \in L^2(\mathcal{V})$ can be defined by analogy

$$\mathbf{f} \star \mathbf{g} \quad = \quad \underbrace{\sum_{k \geq 1} \underbrace{\langle \mathbf{f}, \boldsymbol{\phi}_k \rangle_{L^2(\mathcal{V})} \langle \mathbf{g}, \boldsymbol{\phi}_k \rangle_{L^2(\mathcal{V})}}_{\text{product in the Fourier domain}} \boldsymbol{\phi}_k}_{\text{inverse Fourier transform}}$$

Spectral convolution

Spectral convolution of $\mathbf{f}, \mathbf{g} \in L^2(\mathcal{V})$ can be defined by analogy

$$\mathbf{f} \star \mathbf{g} \quad = \quad \sum_{k \geq 1} \langle \mathbf{f}, \boldsymbol{\phi}_k \rangle_{L^2(\mathcal{V})} \langle \mathbf{g}, \boldsymbol{\phi}_k \rangle_{L^2(\mathcal{V})} \, \boldsymbol{\phi}_k$$

In matrix-vector notation

$$\mathbf{f} \star \mathbf{g} = \boldsymbol{\Phi} \, (\boldsymbol{\Phi}^\top \mathbf{g}) \circ (\boldsymbol{\Phi}^\top \mathbf{f})$$

Spectral convolution

Spectral convolution of $\mathbf{f}, \mathbf{g} \in L^2(\mathcal{V})$ can be defined by analogy

$$\mathbf{f} \star \mathbf{g} \quad = \quad \sum_{k \geq 1} \langle \mathbf{f}, \boldsymbol{\phi}_k \rangle_{L^2(\mathcal{V})} \langle \mathbf{g}, \boldsymbol{\phi}_k \rangle_{L^2(\mathcal{V})} \, \boldsymbol{\phi}_k$$

In matrix-vector notation

$$\mathbf{f} \star \mathbf{g} = \boldsymbol{\Phi} \, \mathrm{diag}(\hat{g}_1, \ldots, \hat{g}_n) \boldsymbol{\Phi}^\top \mathbf{f}$$

Spectral convolution

Spectral convolution of $\mathbf{f}, \mathbf{g} \in L^2(\mathcal{V})$ can be defined by analogy

$$\mathbf{f} \star \mathbf{g} \quad = \quad \sum_{k \geq 1} \langle \mathbf{f}, \boldsymbol{\phi}_k \rangle_{L^2(\mathcal{V})} \langle \mathbf{g}, \boldsymbol{\phi}_k \rangle_{L^2(\mathcal{V})} \, \boldsymbol{\phi}_k$$

In matrix-vector notation

$$\mathbf{f} \star \mathbf{g} = \underbrace{\boldsymbol{\Phi} \, \mathrm{diag}(\hat{g}_1, \ldots, \hat{g}_n) \boldsymbol{\Phi}^\top}_{\mathbf{G}} \mathbf{f}$$

Spectral convolution

Spectral convolution of $\mathbf{f}, \mathbf{g} \in L^2(\mathcal{V})$ can be defined by analogy

$$\mathbf{f} \star \mathbf{g} \quad = \quad \sum_{k \geq 1} \langle \mathbf{f}, \boldsymbol{\phi}_k \rangle_{L^2(\mathcal{V})} \langle \mathbf{g}, \boldsymbol{\phi}_k \rangle_{L^2(\mathcal{V})} \, \boldsymbol{\phi}_k$$

In matrix-vector notation

$$\mathbf{f} \star \mathbf{g} = \underbrace{\boldsymbol{\Phi} \, \mathrm{diag}(\hat{g}_1, \ldots, \hat{g}_n) \boldsymbol{\Phi}^\top}_{\mathbf{G}} \mathbf{f}$$

Not shift-invariant! ($\mathbf{G}$ has no circulant structure)

Spectral convolution

Spectral convolution of $\mathbf{f}, \mathbf{g} \in L^2(\mathcal{V})$ can be defined by analogy

$$\mathbf{f} \star \mathbf{g} \quad = \quad \sum_{k \geq 1} \langle \mathbf{f}, \boldsymbol{\phi}_k \rangle_{L^2(\mathcal{V})} \langle \mathbf{g}, \boldsymbol{\phi}_k \rangle_{L^2(\mathcal{V})} \, \boldsymbol{\phi}_k$$

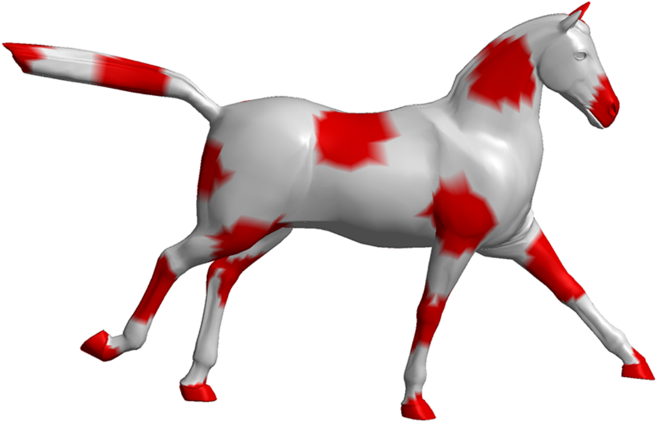
In matrix-vector notation

$$\mathbf{f} \star \mathbf{g} = \underbrace{\boldsymbol{\Phi} \, \mathrm{diag}(\hat{g}_1, \ldots, \hat{g}_n) \boldsymbol{\Phi}^\top}_{\mathbf{G}} \mathbf{f}$$

Not shift-invariant! ($\mathbf{G}$ has no circulant structure)

Filter coefficients depend on basis $\boldsymbol{\phi}_1, \ldots, \boldsymbol{\phi}_n$

Effect of spectral convolution



Function $\mathbf{f}$

Effect of spectral convolution

'Edge detecting' spectral filter $\mathbf{\Phi G \Phi^\top f}$

Effect of spectral convolution

Same spectral filter, different basis $\mathbf{\Psi} \mathbf{G} \mathbf{\Psi}^\top \mathbf{f}$

Effect of spectral convolution

High-frequency Laplacian eigenvector ϕ_{50}

Spectral Graph Convolutional Neural Networks

Spectral graph CNN

Convolution expressed in the spectral domain

$$\mathbf{g} = \mathbf{\Phi} \mathbf{W} \mathbf{\Phi}^\top \mathbf{f}$$

where $\mathbf{W}$ is $n \times n$ diagonal matrix of learnable spectral filter coefficients

Bruna, Zaremba, Szlam, LeCun 2014

Spectral graph CNN

Convolution expressed in the spectral domain

$$\mathbf{g} = \mathbf{\Phi} \mathbf{W} \mathbf{\Phi}^\top \mathbf{f}$$

where $\mathbf{W}$ is $n \times n$ diagonal matrix of learnable spectral filter coefficients

☹ Filters are basis-dependent ⇒ does not generalize across graphs!

Bruna, Zaremba, Szlam, LeCun 2014

Spectral graph CNN

Convolution expressed in the spectral domain

$$\mathbf{g} = \mathbf{\Phi} \mathbf{W} \mathbf{\Phi}^\top \mathbf{f}$$

where $\mathbf{W}$ is $n \times n$ diagonal matrix of learnable spectral filter coefficients

☹ Filters are basis-dependent ⇒ does not generalize across graphs!

☹ $\mathcal{O}(n)$ parameters per layer

Bruna, Zaremba, Szlam, LeCun 2014

Spectral graph CNN

Convolution expressed in the spectral domain

$$\mathbf{g} = \mathbf{\Phi} \mathbf{W} \mathbf{\Phi}^\top \mathbf{f}$$

where $\mathbf{W}$ is $n \times n$ diagonal matrix of learnable spectral filter coefficients

- ☹ Filters are basis-dependent $\Rightarrow$ does not generalize across graphs!
- ☹ $\mathcal{O}(n)$ parameters per layer
- ☹ $\mathcal{O}(n^2)$ computation of forward and inverse Fourier transforms $\mathbf{\Phi}^\top, \mathbf{\Phi}$ (no FFT on graphs)

Bruna, Zaremba, Szlam, LeCun 2014

Spectral graph CNN

Convolution expressed in the spectral domain

$$\mathbf{g} = \mathbf{\Phi} \mathbf{W} \mathbf{\Phi}^\top \mathbf{f}$$

where $\mathbf{W}$ is $n \times n$ diagonal matrix of learnable spectral filter coefficients

- ☹ Filters are basis-dependent $\Rightarrow$ does not generalize across graphs!
- ☹ $\mathcal{O}(n)$ parameters per layer
- ☹ $\mathcal{O}(n^2)$ computation of forward and inverse Fourier transforms $\mathbf{\Phi}^\top, \mathbf{\Phi}$ (no FFT on graphs)
- ☹ No guarantee of spatial localization of filters

Bruna, Zaremba, Szlam, LeCun 2014

Localization and Smoothness

Vanishing moments: In the Euclidean setting

$$\int_{-\infty}^{+\infty} |x|^{2k} |f(x)|^2 dx = \int_{-\infty}^{+\infty} \left| \frac{\partial^k \hat{f}(\omega)}{\partial \omega^k} \right|^2 d\omega$$

Bruna, Zaremba, Szlam, LeCun 2014; Henaff, Bruna, LeCun 2015

Localization and Smoothness

Vanishing moments: In the Euclidean setting

$$\int_{-\infty}^{+\infty} |x|^{2k} |f(x)|^2 dx = \int_{-\infty}^{+\infty} \left| \frac{\partial^k \hat{f}(\omega)}{\partial \omega^k} \right|^2 d\omega$$

Localization in space = smoothness in frequency domain

Bruna, Zaremba, Szlam, LeCun 2014; Henaff, Bruna, LeCun 2015

Localization and Smoothness

Vanishing moments: In the Euclidean setting

$$\int_{-\infty}^{+\infty} |x|^{2k} |f(x)|^2 dx = \int_{-\infty}^{+\infty} \left| \frac{\partial^k \hat{f}(\omega)}{\partial \omega^k} \right|^2 d\omega$$

Localization in space = smoothness in frequency domain

Parametrize the filter using a smooth spectral transfer function $\tau(\lambda)$

Bruna, Zaremba, Szlam, LeCun 2014; Henaff, Bruna, LeCun 2015

Localization and Smoothness

Vanishing moments: In the Euclidean setting

$$\int_{-\infty}^{+\infty} |x|^{2k} |f(x)|^2 dx = \int_{-\infty}^{+\infty} \left| \frac{\partial^k \hat{f}(\omega)}{\partial \omega^k} \right|^2 d\omega$$

Localization in space = smoothness in frequency domain

Parametrize the filter using a smooth spectral transfer function $\tau(\lambda)$

Application of the filter

$$\tau(\mathbf{\Delta})\mathbf{f} = \mathbf{\Phi}\tau(\mathbf{\Lambda})\mathbf{\Phi}^\top \mathbf{f}$$

Bruna, Zaremba, Szlam, LeCun 2014; Henaff, Bruna, LeCun 2015

Localization and Smoothness

Vanishing moments: In the Euclidean setting

$$\int_{-\infty}^{+\infty} |x|^{2k}|f(x)|^2 dx = \int_{-\infty}^{+\infty} \left| \frac{\partial^k \hat{f}(\omega)}{\partial \omega^k} \right|^2 d\omega$$

Localization in space = smoothness in frequency domain

Parametrize the filter using a smooth spectral transfer function $\tau(\lambda)$

Application of the filter

$$\tau(\mathbf{\Delta})\mathbf{f} = \mathbf{\Phi} \begin{pmatrix} \tau(\lambda_1) & & \\ & \ddots & \\ & & \tau(\lambda_n) \end{pmatrix} \mathbf{\Phi}^\top \mathbf{f}$$

Bruna, Zaremba, Szlam, LeCun 2014; Henaff, Bruna, LeCun 2015

Localization and Smoothness

Vanishing moments: In the Euclidean setting

$$\int_{-\infty}^{+\infty} |x|^{2k}|f(x)|^2 dx = \int_{-\infty}^{+\infty} \left| \frac{\partial^k \hat{f}(\omega)}{\partial \omega^k} \right|^2 d\omega$$

Localization in space = smoothness in frequency domain

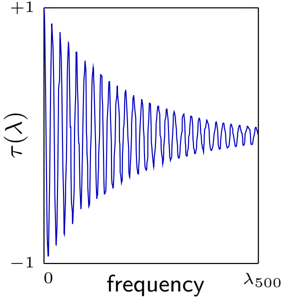
Parametrize the filter using a smooth spectral transfer function $\tau(\lambda)$

Application of the parametric filter with learnable parameters $\boldsymbol{\alpha}$

$$\tau_{\boldsymbol{\alpha}}(\boldsymbol{\Delta})\mathbf{f} = \boldsymbol{\Phi} \begin{pmatrix} \tau_{\boldsymbol{\alpha}}(\lambda_1) & & \\ & \ddots & \\ & & \tau_{\boldsymbol{\alpha}}(\lambda_n) \end{pmatrix} \boldsymbol{\Phi}^\top \mathbf{f}$$

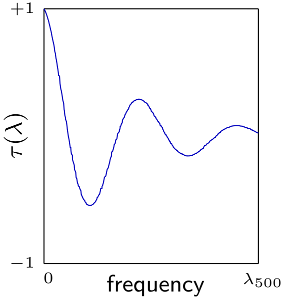
Bruna, Zaremba, Szlam, LeCun 2014; Henaff, Bruna, LeCun 2015

Localization and smoothness



Non-smooth spectral filter (delocalized in space)

Localization and smoothness



Smooth spectral filter (localized in space)

Spectral graph CNN with polynomial filters (ChebNet)

Represent spectral transfer function as a polynomial or order r

$$\tau_{\boldsymbol{\alpha}}(\lambda) = \sum_{j=0}^{r} \alpha_j \lambda^j$$

where $\boldsymbol{\alpha} = (\alpha_0, \ldots, \alpha_r)^\top$ is the vector of filter parameters

Defferrard, Bresson, Vandergheynst 2016; Kipf, Welling 2016

Spectral graph CNN with polynomial filters (ChebNet)

Represent spectral transfer function as a polynomial or order r

$$\tau_{\boldsymbol{\alpha}}(\lambda) = \sum_{j=0}^{r} \alpha_j \lambda^j$$

where $\boldsymbol{\alpha} = (\alpha_0, \ldots, \alpha_r)^\top$ is the vector of filter parameters

☺ $\mathcal{O}(1)$ parameters per layer

Defferrard, Bresson, Vandergheynst 2016; Kipf, Welling 2016

Spectral graph CNN with polynomial filters (ChebNet)

Represent spectral transfer function as a polynomial or order r

$$\tau_{\boldsymbol{\alpha}}(\lambda) = \sum_{j=0}^{r} \alpha_j \lambda^j$$

where $\boldsymbol{\alpha} = (\alpha_0, \ldots, \alpha_r)^\top$ is the vector of filter parameters

- ☺ $\mathcal{O}(1)$ parameters per layer
- ☺ Filters have guaranteed r-hops support

Defferrard, Bresson, Vandergheynst 2016; Kipf, Welling 2016

Spectral graph CNN with polynomial filters (ChebNet)

Represent spectral transfer function as a polynomial or order r

$$\tau_{\boldsymbol{\alpha}}(\lambda) = \sum_{j=0}^{r} \alpha_j \lambda^j$$

where $\boldsymbol{\alpha} = (\alpha_0, \ldots, \alpha_r)^\top$ is the vector of filter parameters

- ☺ $\mathcal{O}(1)$ parameters per layer
- ☺ Filters have guaranteed r-hops support
- ☺ No explicit computation of $\mathbf{\Phi}^\top, \mathbf{\Phi} \Rightarrow \mathcal{O}(nr)$ computational complexity

Defferrard, Bresson, Vandergheynst 2016; Kipf, Welling 2016

Example: citation networks

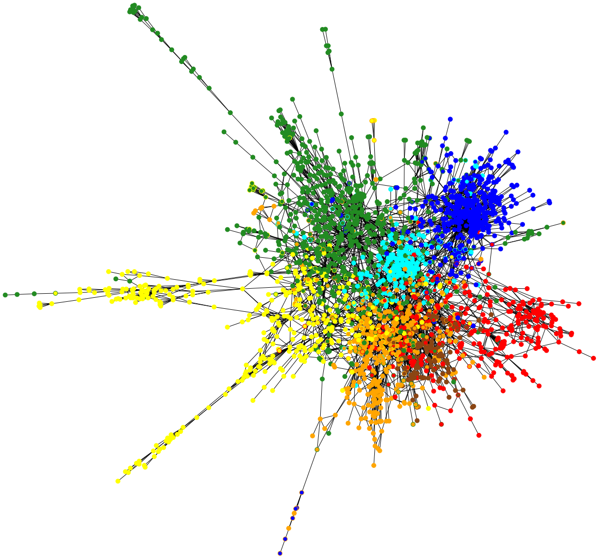


Figure: Monti, Boscaini, Masci, Rodolà, Svoboda, Bronstein 2017

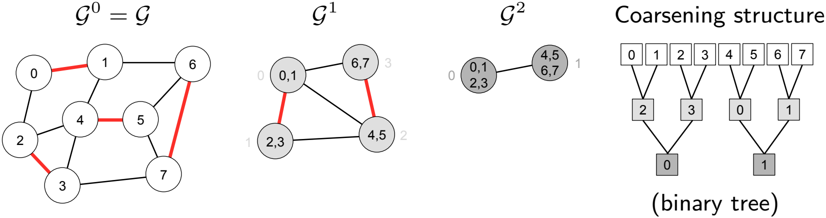
Example: citation networks

Method	**Cora**[1]	**PubMed**[2]
Manifold Regularization[3]	59.5%	70.7%
Semidefinite Embedding[4]	59.0%	71.1%
Label Propagation[5]	68.0%	63.0%
DeepWalk[6]	67.2%	65.3%
Planetoid[7]	75.7%	77.2%
Spectral graph CNN[8]	**81.6%**	**78.7%**

Classification accuracy of different methods on citation network datasets

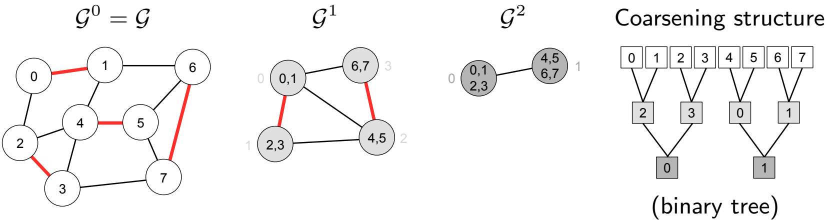
Data: [1,2]Sen et al. 2008; methods: [3]Belkin et al. 2006; [4]Weston et al. 2012; [5]Zhu et al. 2003; [6]Perozzi et al. 2014; [7]Yang et al. 2016; [8]Kipf, Welling 2016 (simplification of ChebNet)

Graph pooling



Produce a sequence of coarsened graphs

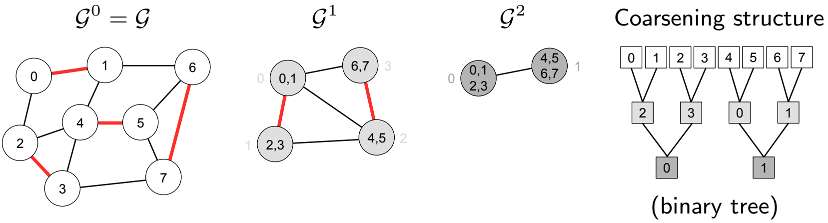
Graph pooling



Produce a sequence of coarsened graphs

Max or average pooling of collapsed vertices

Graph pooling



Produce a sequence of coarsened graphs

Max or average pooling of collapsed vertices

Binary tree arrangement of node indices

Limitations of spectral graph CNNs

- ☹ Poor generalization across domains with different shapes unless kernels are localized

Limitations of spectral graph CNNs

- ☹ Poor generalization across domains with different shapes unless kernels are localized (can be remedied to some extent with spectral transformer networks[1])

[1] Yi, Su, Guo, Guibas 2017

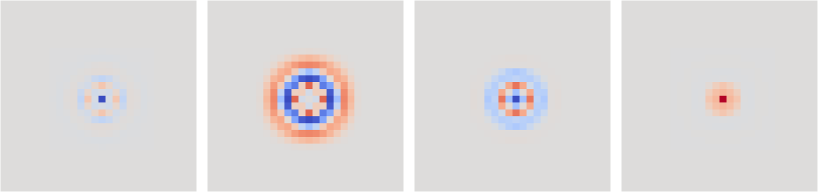
Limitations of spectral graph CNNs

- ☹ Poor generalization across domains with different shapes unless kernels are localized (can be remedied to some extent with spectral transformer networks[1])

- ☹ Spectral kernels are isotropic due to rotation invariance of the Laplacian

[1] Yi, Su, Guo, Guibas 2017

Only rotationally-symmetric kernels!



Example of Chebyshev filters (order $r = 7$) on Euclidean grid

Anisotropic kernels on manifolds

Scale t Orientation θ Elongation α

Examples of anisotropic heat kernels on a manifold

Boscaini, Masci, Rodolà, Bronstein, Cremers 2016

Limitations of spectral graph CNNs

- ☹ Poor generalization across domains with different shapes unless kernels are localized (can be remedied to some extent with spectral transformer networks[1])

- ☹ Spectral kernels are isotropic due to rotation invariance of the Laplacian (on manifolds, anisotropic Laplacians can be constructed[2])

[1] Yi, Su, Guo, Guibas 2017; [2] Boscaini, Masci, Rodolà, Bronstein, Cremers 2016

Limitations of spectral graph CNNs

- ☹ Poor generalization across domains with different shapes unless kernels are localized (can be remedied to some extent with spectral transformer networks[1])
- ☹ Spectral kernels are isotropic due to rotation invariance of the Laplacian (on manifolds, anisotropic Laplacians can be constructed[2])
- ☹ Only undirected graphs, as symmetry of the Laplacian matrix is assumed

[1] Yi, Su, Guo, Guibas 2017; [2] Boscaini, Masci, Rodolà, Bronstein, Cremers 2016

Different formulations of non-Euclidean CNNs

Spectral domain

Spatial domain

Different formulations of non-Euclidean CNNs

Spectral domain

Spatial domain

Geometric Deep Learning Tutorial: NIPS 2017

https://vimeo.com/248497329

Summary

The convolution theorem allows us to generalize convolution operators from grids to graphs

Summary

The convolution theorem allows us to generalize convolution operators from grids to graphs

Naive spectral graph CNNs are unstable to deformation

Summary

The convolution theorem allows us to generalize convolution operators from grids to graphs

Naive spectral graph CNNs are unstable to deformation

Forcing the filter spectrum to be smooth stabilizes and localizes filters

Summary

The convolution theorem allows us to generalize convolution operators from grids to graphs

Naive spectral graph CNNs are unstable to deformation

Forcing the filter spectrum to be smooth stabilizes and localizes filters

Pooling operation can be replaced with graph pooling

Inference in Spectral Learning with Deep Networks

Manifold learning (again)

Basic pattern:

Construct Gram matrix from kernel $k(\mathbf{x}, \mathbf{x}')$

$$\mathbf{M} = \begin{bmatrix} k(\mathbf{x}_1, \mathbf{x}_1) & k(\mathbf{x}_1, \mathbf{x}_2) & \dots & k(\mathbf{x}_1, \mathbf{x}_n) \\ k(\mathbf{x}_2, \mathbf{x}_1) & k(\mathbf{x}_2, \mathbf{x}_2) & \dots & k(\mathbf{x}_2, \mathbf{x}_n) \\ \vdots & \vdots & \ddots & \vdots \\ k(\mathbf{x}_n, \mathbf{x}_1) & k(\mathbf{x}_n, \mathbf{x}_2) & \dots & k(\mathbf{x}_n, \mathbf{x}_n) \end{bmatrix}$$

Use top/bottom eigenvectors of Gram matrix as embedding

Use Nyström approximation for inference on held-out data:
$\phi_k(\mathbf{x}') \propto \sum_i \phi_{ki} k(\mathbf{x}_i, \mathbf{x}')$

Manifold learning (again)

Basic pattern:

Construct Gram matrix from kernel $k(\mathbf{x}, \mathbf{x}')$

$$\mathbf{M} = \begin{bmatrix} k(\mathbf{x}_1, \mathbf{x}_1) & k(\mathbf{x}_1, \mathbf{x}_2) & \dots & k(\mathbf{x}_1, \mathbf{x}_n) \\ k(\mathbf{x}_2, \mathbf{x}_1) & k(\mathbf{x}_2, \mathbf{x}_2) & \dots & k(\mathbf{x}_2, \mathbf{x}_n) \\ \vdots & \vdots & \ddots & \vdots \\ k(\mathbf{x}_n, \mathbf{x}_1) & k(\mathbf{x}_n, \mathbf{x}_2) & \dots & k(\mathbf{x}_n, \mathbf{x}_n) \end{bmatrix}$$

Use top/bottom eigenvectors of Gram matrix as embedding

Use Nyström approximation for inference on held-out data:
$\phi_k(\mathbf{x}') \propto \sum_i \phi_{ki} k(\mathbf{x}_i, \mathbf{x}')$

Why not just learn parameterized $\phi_k(\mathbf{x})$ directly?

Deep learn all the things

A successful strategy: take a branch of machine learning and fit the model using a deep network

Deep learn all the things

A successful strategy: take a branch of machine learning and fit the model using a deep network

Variational Bayes: Variational autoencoders

Deep learn all the things

A successful strategy: take a branch of machine learning and fit the model using a deep network

Variational Bayes: Variational autoencoders

Reinforcement Learning: Deep RL (DQN, A3C, PPO, TRPO, DDPG...)

Deep learn all the things

A successful strategy: take a branch of machine learning and fit the model using a deep network

Variational Bayes: Variational autoencoders

Reinforcement Learning: Deep RL (DQN, A3C, PPO, TRPO, DDPG...)

Spectral Learning: ????

Deep learn all the things

A successful strategy: take a branch of machine learning and fit the model using a deep network

Variational Bayes: Variational autoencoders

Reinforcement Learning: Deep RL (DQN, A3C, PPO, TRPO, DDPG...)

Spectral Learning: **Spectral Inference Networks**

Pfau, Petersen, Agarwal, Barrett and Stachenfeld 2018

Trade-offs of manifold learning

- ☺ Exactly solvable by eigendecomposition
- ☺ Data efficient (works with $\mathcal{O}(1000)$ data points)
- 😐 Unsupervised learning without a generative model
- ☹ Learning scales as $\mathcal{O}(n \log n)$ with size of training data
- ☹ Inference scales as $\mathcal{O}(n)$ with size of training data
- ☹ Performance degrades for noisy or clustered data
- ☺ Collapsed embeddings can be fixed by choice of eigenvector
- ☺ Can discover topology of data without prior assumptions

Trade-offs of manifold learning

☺ **Exactly solvable by eigendecomposition**

☺ **Data efficient (works with $\mathcal{O}(1000)$ data points)**

😐 Unsupervised learning without a generative model

☹ **Learning scales as $\mathcal{O}(n \log n)$ with size of training data**

☹ **Inference scales as $\mathcal{O}(n)$ with size of training data**

☹ Performance degrades for noisy or clustered data

☺ Collapsed embeddings can be fixed by choice of eigenvector

☺ Can discover topology of data without prior assumptions

Spectral inference networks trade the first two for the second two

Eigendecomposition as optimization

Rayleigh quotient has top eigenvector as argmax:

$$\max_{\boldsymbol{\phi}} \frac{\boldsymbol{\phi}^T \mathbf{A} \boldsymbol{\phi}}{\boldsymbol{\phi}^T \boldsymbol{\phi}}$$

Pfau, Petersen, Agarwal, Barrett and Stachenfeld 2018

Eigendecomposition as optimization

Rayleigh quotient has top eigenvector as argmax:

$$\max_{\boldsymbol{\phi}} \frac{\boldsymbol{\phi}^T \mathbf{A} \boldsymbol{\phi}}{\boldsymbol{\phi}^T \boldsymbol{\phi}}$$

Generalize to multiple eigenvectors (up to rotation):

$$\max_{\boldsymbol{\Phi}} \mathrm{Tr}\left((\boldsymbol{\Phi}^T \boldsymbol{\Phi})^{-1} \boldsymbol{\Phi}^T \mathbf{A} \boldsymbol{\Phi} \right)$$

Pfau, Petersen, Agarwal, Barrett and Stachenfeld 2018

Eigendecomposition as optimization

Rayleigh quotient has top eigenvector as argmax:

$$\max_{\boldsymbol{\phi}} \frac{\boldsymbol{\phi}^T \mathbf{A} \boldsymbol{\phi}}{\boldsymbol{\phi}^T \boldsymbol{\phi}}$$

Generalize to multiple eigenvectors (up to rotation):

$$\max_{\mathbf{\Phi}} \mathrm{Tr}\left((\mathbf{\Phi}^T \mathbf{\Phi})^{-1} \mathbf{\Phi}^T \mathbf{A} \mathbf{\Phi}\right)$$

$$\max_{\mathbf{\Phi}} \mathrm{Tr}\left(\left(\sum_i \boldsymbol{\phi}_i^T \boldsymbol{\phi}_i\right)^{-1} \sum_{ij} A_{ij} \boldsymbol{\phi}_i^T \boldsymbol{\phi}_j\right)$$

Pfau, Petersen, Agarwal, Barrett and Stachenfeld 2018

Eigendecomposition as optimization

Rayleigh quotient has top eigenvector as argmax:

$$\max_{\boldsymbol{\phi}} \frac{\boldsymbol{\phi}^T \mathbf{A} \boldsymbol{\phi}}{\boldsymbol{\phi}^T \boldsymbol{\phi}}$$

Generalize to multiple eigenvectors (up to rotation):

$$\max_{\boldsymbol{\Phi}} \operatorname{Tr}\left((\boldsymbol{\Phi}^T \boldsymbol{\Phi})^{-1} \boldsymbol{\Phi}^T \mathbf{A} \boldsymbol{\Phi} \right)$$

$$\max_{\boldsymbol{\phi}} \operatorname{Tr}\left(\mathbb{E}_{\mathbf{x}} \left[\boldsymbol{\phi}(\mathbf{x}) \boldsymbol{\phi}(\mathbf{x})^T \right]^{-1} \mathbb{E}_{\mathbf{x},\mathbf{x}'} \left[k(\mathbf{x}, \mathbf{x}') \boldsymbol{\phi}(\mathbf{x}) \boldsymbol{\phi}(\mathbf{x}')^T \right] \right)$$

Replace A_{ij} with $k(\mathbf{x}, \mathbf{x}')$ and sums with expectations

Pfau, Petersen, Agarwal, Barrett and Stachenfeld 2018

Eigendecomposition as optimization

Most machine learning:

$$\max_{\theta} \mathbb{E}_{\mathbf{x}}[f_{\theta}(\mathbf{x})]$$

Empirical gradient in unbiased

Pfau, Petersen, Agarwal, Barrett and Stachenfeld 2018

Eigendecomposition as optimization

Most machine learning:

$$\max_{\theta} \mathbb{E}_{\mathbf{x}}[f_\theta(\mathbf{x})]$$

Empirical gradient in unbiased

Spectral inference networks:

$$\max_{\theta} \mathrm{Tr}\left(\mathbb{E}_{\mathbf{x}}\left[\boldsymbol{\phi}_\theta(\mathbf{x})\boldsymbol{\phi}_\theta(\mathbf{x})^T\right]^{-1}\mathbb{E}_{\mathbf{x},\mathbf{x}'}\left[k(\mathbf{x},\mathbf{x}')\boldsymbol{\phi}_\theta(\mathbf{x})\boldsymbol{\phi}_\theta(\mathbf{x}')^T\right]\right)$$

Empirical gradient is biased

Pfau, Petersen, Agarwal, Barrett and Stachenfeld 2018

Eigendecomposition as optimization

Most machine learning:

$$\max_{\theta} \mathbb{E}_{\mathbf{x}}[f_\theta(\mathbf{x})]$$

Empirical gradient in unbiased

Spectral inference networks:

$$\max_{\theta} \mathrm{Tr}\left(\mathbb{E}_{\mathbf{x}}\left[\boldsymbol{\phi}_\theta(\mathbf{x})\boldsymbol{\phi}_\theta(\mathbf{x})^T\right]^{-1}\mathbb{E}_{\mathbf{x},\mathbf{x}'}\left[k(\mathbf{x},\mathbf{x}')\boldsymbol{\phi}_\theta(\mathbf{x})\boldsymbol{\phi}_\theta(\mathbf{x}')^T\right]\right)$$

Empirical gradient is biased

Solution: use moving average of gradient of $\phi_\theta \phi_\theta^T$ term

Pfau, Petersen, Agarwal, Barrett and Stachenfeld 2018

Spectral inference network gradients

Objective is of the form $\mathrm{Tr}(\mathbf{\Sigma}^{-1}\mathbf{\Pi})$

Pfau, Petersen, Agarwal, Barrett and Stachenfeld 2018

Spectral inference network gradients

Objective is of the form $\mathrm{Tr}(\mathbf{\Sigma}^{-1}\mathbf{\Pi})$

Gradient is of the form $\mathrm{Tr}\left(\mathbf{\Sigma}^{-1}\nabla_\theta\mathbf{\Pi}\right) - \mathrm{Tr}\left(\mathbf{\Sigma}^{-1}\mathbf{\Pi}\mathbf{\Sigma}^{-1}\nabla_\theta\mathbf{\Sigma}\right)$

Pfau, Petersen, Agarwal, Barrett and Stachenfeld 2018

Spectral inference network gradients

Objective is of the form $\mathrm{Tr}(\mathbf{\Sigma}^{-1}\mathbf{\Pi})$

Gradient is of the form $\mathrm{Tr}\left(\mathbf{\Sigma}^{-1}\nabla_\theta\mathbf{\Pi}\right) - \mathrm{Tr}\left(\mathbf{\Sigma}^{-1}\mathbf{\Pi}\mathbf{\Sigma}^{-1}\nabla_\theta\mathbf{\Sigma}\right)$

To break symmetry between eigenfunctions, use gradient

$$\mathrm{Tr}\left(\mathbf{L}^{-T}\mathrm{diag}(\mathbf{L})^{-1}\nabla_\theta\mathbf{\Pi}\right) - \mathrm{Tr}\left(\mathbf{L}^{-T}\mathrm{triu}\left(\mathbf{\Lambda}\mathrm{diag}(\mathbf{L})^{-1}\right)\nabla_\theta\mathbf{\Sigma}\right)$$

where $\mathbf{L}$ is Cholesky decomposition of $\mathbf{\Pi}$ and $\mathbf{\Lambda} = \mathbf{L}^{-1}\mathbf{\Pi}\mathbf{L}^{-T}$

Pfau, Petersen, Agarwal, Barrett and Stachenfeld 2018

Spectral inference network gradients

Objective is of the form $\mathrm{Tr}(\mathbf{\Sigma}^{-1}\mathbf{\Pi})$

Gradient is of the form $\mathrm{Tr}\left(\mathbf{\Sigma}^{-1}\nabla_\theta\mathbf{\Pi}\right) - \mathrm{Tr}\left(\mathbf{\Sigma}^{-1}\mathbf{\Pi}\mathbf{\Sigma}^{-1}\nabla_\theta\mathbf{\Sigma}\right)$

To break symmetry between eigenfunctions, use gradient

$$\mathrm{Tr}\left(\mathbf{L}^{-T}\mathrm{diag}(\mathbf{L})^{-1}\nabla_\theta\mathbf{\Pi}\right) - \mathrm{Tr}\left(\mathbf{L}^{-T}\mathrm{triu}\left(\mathbf{\Lambda}\mathrm{diag}(\mathbf{L})^{-1}\right)\nabla_\theta\mathbf{\Sigma}\right)$$

where $\mathbf{L}$ is Cholesky decomposition of $\mathbf{\Pi}$ and $\mathbf{\Lambda} = \mathbf{L}^{-1}\mathbf{\Pi}\mathbf{L}^{-T}$

To reduce bias in the gradient, use moving average for $\mathbf{\Sigma}$ and $\nabla_\theta\mathbf{\Sigma}$

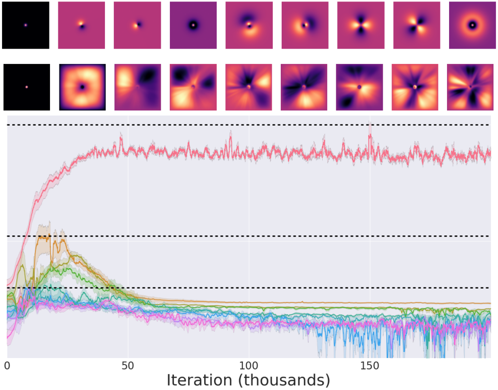
Pfau, Petersen, Agarwal, Barrett and Stachenfeld 2018

Spectral inference networks

Sanity check: the Schrödinger equation

$$E\psi(\mathbf{x}) = -\frac{\hbar}{2m}\nabla^2\psi(\mathbf{x}) - \frac{\psi(\mathbf{x})}{|\mathbf{x}|}$$

Without bias correction:



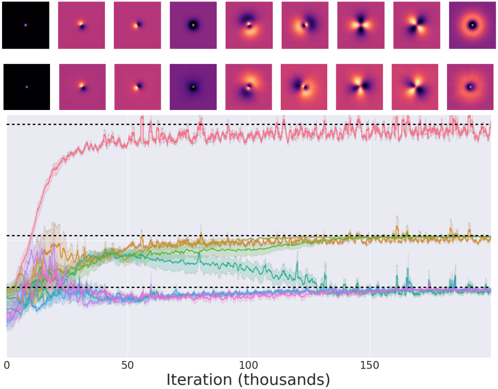
Pfau, Petersen, Agarwal, Barrett and Stachenfeld 2018

Spectral inference networks

Sanity check: the Schrödinger equation

$$E\psi(\mathbf{x}) = -\frac{\hbar}{2m}\nabla^2\psi(\mathbf{x}) - \frac{\psi(\mathbf{x})}{|\mathbf{x}|}$$

With bias correction:



Pfau, Petersen, Agarwal, Barrett and Stachenfeld 2018

Approximating Laplacian eigenmaps

Objective:

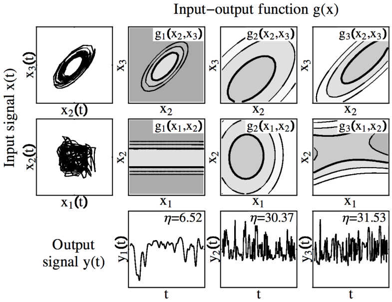
$$k(\mathbf{x}, \mathbf{x}')\boldsymbol{\phi}(\mathbf{x})\boldsymbol{\phi}(\mathbf{x})^T = (\boldsymbol{\phi}(\mathbf{x}) - \boldsymbol{\phi}(\mathbf{x}'))(\boldsymbol{\phi}(\mathbf{x}) - \boldsymbol{\phi}(\mathbf{x}'))^T$$

Approximating Laplacian eigenmaps

Objective:

$$k(\mathbf{x},\mathbf{x}')\boldsymbol{\phi}(\mathbf{x})\boldsymbol{\phi}(\mathbf{x})^T = (\boldsymbol{\phi}(\mathbf{x}) - \boldsymbol{\phi}(\mathbf{x}'))(\boldsymbol{\phi}(\mathbf{x}) - \boldsymbol{\phi}(\mathbf{x}'))^T$$

If $\mathbf{x}$, $\mathbf{x}'$ are sequential video frames, equivalent to Slow Feature Analysis



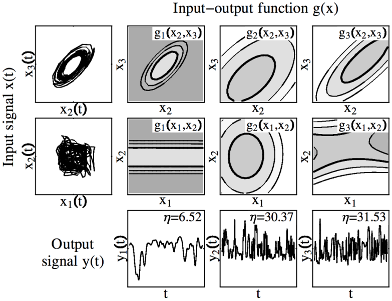
Wiskott and Sejnowski 2002; Sprekeler 2011

Approximating Laplacian eigenmaps

Objective:

$$k(\mathbf{x}, \mathbf{x}')\boldsymbol{\phi}(\mathbf{x})\boldsymbol{\phi}(\mathbf{x})^T = (\boldsymbol{\phi}(\mathbf{x}) - \boldsymbol{\phi}(\mathbf{x}'))(\boldsymbol{\phi}(\mathbf{x}) - \boldsymbol{\phi}(\mathbf{x}'))^T$$

If $\mathbf{x}$, $\mathbf{x}'$ are sequential video frames, equivalent to Slow Feature Analysis



SFA learned layer-by-layer rather than end-to-end

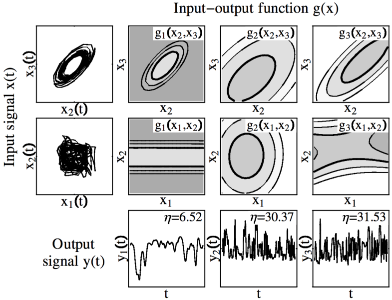
Wiskott and Sejnowski 2002; Sprekeler 2011

Approximating Laplacian eigenmaps

Objective:

$$k(\mathbf{x}, \mathbf{x}')\boldsymbol{\phi}(\mathbf{x})\boldsymbol{\phi}(\mathbf{x})^T = (\boldsymbol{\phi}(\mathbf{x}) - \boldsymbol{\phi}(\mathbf{x}'))(\boldsymbol{\phi}(\mathbf{x}) - \boldsymbol{\phi}(\mathbf{x}'))^T$$

If $\mathbf{x}$, $\mathbf{x}'$ are sequential video frames, equivalent to Slow Feature Analysis



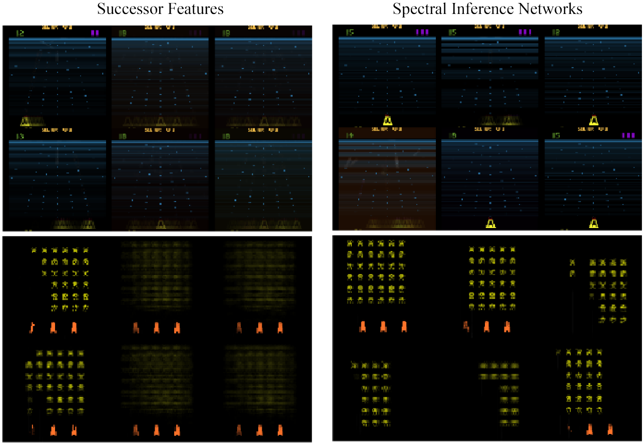
SFA learned layer-by-layer rather than end-to-end

SFA learned feature-by-feature rather than fully online

Wiskott and Sejnowski 2002; Sprekeler 2011

Spectral inference networks on Atari

More interpretable features compared to other approaches when trained on random policies on Atari games



Pfau, Petersen, Agarwal, Barrett and Stachenfeld 2018

Summary

Eigenfunctions can be learned by stochastic gradient descent with function approximation

Summary

Eigenfunctions can be learned by stochastic gradient descent with function approximation

Slow feature analysis is a special case of Spectral Inference Networks

Summary

Eigenfunctions can be learned by stochastic gradient descent with function approximation

Slow feature analysis is a special case of Spectral Inference Networks

While less efficient than standard spectral algorithms, it is far more scalable

Future challenges in manifold and spectral learning

Can we encode more structural assumptions into the choice of kernel?

Future challenges in manifold and spectral learning

Can we encode more structural assumptions into the choice of kernel?

Can we combine the speed and efficiency of nonparametric methods with the scalability of parametric methods?

Future challenges in manifold and spectral learning

Can we encode more structural assumptions into the choice of kernel?

Can we combine the speed and efficiency of nonparametric methods with the scalability of parametric methods?

Can we use curvature in observation space as a learning signal rather than a post-hoc analysis tool?

Future challenges in manifold and spectral learning

Can we encode more structural assumptions into the choice of kernel?

Can we combine the speed and efficiency of nonparametric methods with the scalability of parametric methods?

Can we use curvature in observation space as a learning signal rather than a post-hoc analysis tool?

Can we use the learned representations for challenging downstream tasks?

Future challenges in manifold and spectral learning

Can we encode more structural assumptions into the choice of kernel?

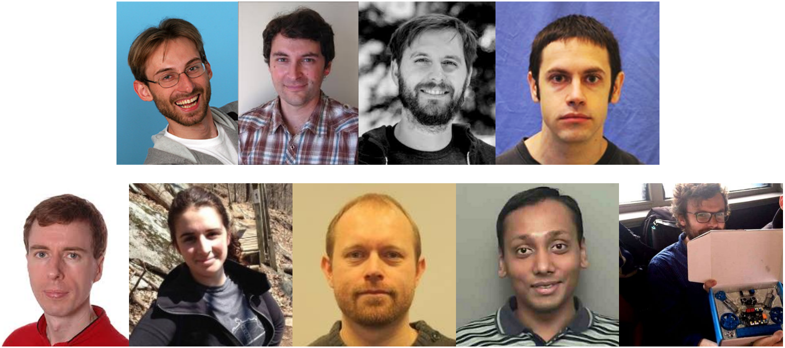
Can we combine the speed and efficiency of nonparametric methods with the scalability of parametric methods?

Can we use curvature in observation space as a learning signal rather than a post-hoc analysis tool?

Can we use the learned representations for challenging downstream tasks?

Can we better connect spectral learning with probabilistic models?

Acknowledgements



Michael Bronstein, Xavier Bresson, Joan Bruna, Arthur Szlam, David Barrett, Kim Stachenfeld, Stig Petersen, Ashish Agarwal, Chris Burgess

Get in touch: `pfau@google.com`, `@pfau` on Twitter

Bibliography

M. P. Do Carmo, "Differential Geometry of Curves and Manifolds", **Classic textbook on differential geometry**

R. Kimmel and J. A Sethian, "Computing Geodesic Paths on Manifolds", PNAS 95(15): 8431-8435, 1998.

S. Roweis and L. K. Saul, "Nonlinear Dimensionality Reduction by Locally Linear Embedding", Science 290(5500): 2323-2326, 2000.

J. B. Tenenbaum, V. De Silva, J. C. Langford, "A Global Geometric Framework for Nonlinear Dimensionality Reduction", Science 290(5500): 2319-2323, 2000.

M. Belkin and P. Niyogi, "Laplacian Eigenmaps for Dimensionality Reduction and Data Representation", Neural Computation 15(6): 1373-1396, 2002.

T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado and J. Dean, "Distributed Representation of Words and Phrases and Their Compositionality", Adv. NIPS: 3111-3119, 2013. **Deep learning of high-dimensional word embeddings with skip-grams**

Bibliography

A. Y. Ng, M. I. Jordan, Y. Weiss, "On Spectral Clustering: Analysis and An Algorithm", Adv. NIPS: 849-856, 2002.

J. Shi and J. Malik, "Normalized Cuts and Image Segmentation", IEEE TPAMI 22(8): 888-905, 2000.

C. Ionescu, O. Vantzos, C. Sminchisescu, "Matrix Backpropagation for Deep Networks with Structured Data", Proc. ICCV: 2965-2973, 2015. **Use normalized cuts as layer inside deep network**

Y. Bengio, J.-F. Paiement, P. Vincent, O. Delalleau, N. Le Roux, M. Ouimet, "Out-of-Sample Extensions for LLE, IsoMap, MDS, Eigenmaps and Spectral Clustering", Adv. NIPS: 177-184, 2004.

L. van der Maaten and G. Hinton, "Visualizing Data using t-SNE", JMLR 9(11): 2579-2605, 2008.

R. Hadsell, S. Chopra and Y. LeCun, "Dimensionality Reduction by Learning an Invariant Mapping", CVPR: 1735-1742, 2006.

Bibliography

D. Pfau and C. P. Burgess, "Minimally Redundant Laplacian Eigenmaps", ICLR Workshops, 2018.

M. Nickel and D. Kiela, "Poincaré Embeddings for Learning Hierarchical Representations", Adv. NIPS: 6341-6350, 2017.

M. Nickel and D. Kiela, "Learning Continuous Hierarchies in the Lorentz Model of Hyperbolic Geometry", ICML 2018.

S. Bonnabel, "Stochastic Gradient Descent on Riemannian Manifolds", IEEE Trans. Automatic Control 58(9): 2217-2229, 2013.

C. Gulcehre, M. Denil, M. Malinowski, A. Razavi, R. Pascanu, K. M. Hermann, P. Battaglia, V. Bapst, D. Raposo, A. Santoro, N. de Freitas, "Hyperbolic Attention Networks", arXiv:1805.09786, 2018.

I. Higgins, L. Matthey, A. Pal, C. P. Burgess, X. Glorot, M. Botvinick, S. Mohamed, A. Lerchner, "β-VAE: Learning Basic Visual Concepts with a Constrained Variational Framework", Proc. ICLR, 2017.

Bibliography

G. Arvanitidis, L. K. Hansen, S. Hauberg, “Latent Space Oddity: on the Curvature of Deep Generative Models”, ICLR 2018.

B. Boots, S. M. Siddiqi, G. J. Gordon, “Closing the Learning-Planning Loop with Predictive State Representations”, IJRR 30(7): 954-966, 2011. **Use spectral learning for planning and control**

M. M. Bronstein, J. Bruna, Y. LeCun, A. Szlam, P. Vandergheynst, “Geometric deep learning: going beyond Euclidean data”, arXiv:1611.08097, 2016. **First review paper of geometric deep learning**

J. Bruna, W. Zaremba, A. Szlam, Y. LeCun, “Spectral networks and locally connected networks on graphs”, *Proc. ICML* 2014. **First Spectral CNN on graphs**

M. Henaff, J. Bruna, and Y. LeCun, “Deep convolutional networks on graph-structured data”, arXiv:1506.05163, 2015. **Spectral CNN with smooth multipliers**

Bibliography

M. Defferrard, X. Bresson, P. Vandergheynst, "Convolutional neural networks on graphs with fast localized spectral filtering", *Proc. NIPS* 2016. **Spectral CNN with Chebychev polynomial filters (ChebNet)**

T. N. Kipf, M. Welling, "Semi-supervised classification with graph convolutional networks", arXiv:1609.02907, 2016. **Graph convolutional network (GCN) framework, a simplification of ChebNet**

F. Monti, D. Boscaini, J. Masci, E. Rodolà, J. Svoboda, M. M. Bronstein, "Geometric deep learning on graphs and manifolds using mixture model CNNs", *Proc. CVPR* 2017. **Mixture Model network (MoNet) framework**

D. Boscaini, J. Masci, E. Rodolà, M. M. Bronstein, D. Cremers, "Anisotropic diffusion descriptors", *Computer Graphics Forum* 35(2):431–441, 2016. **Anisotropic heat kernels**

L. Yi, H. Su, X. Guo, L. Guibas, "SyncSpecCNN: Synchronized Spectral CNN for 3D Shape Segmentation", Proc. CVPR 2017. **Spectral transformer networks**

Bibliography

Y. LeCun, B. Boser, J. S. Denker, R. E. Howard, W. Hubbard, L. D. Jackel, "Backpropagation applied to handwritten ZIP code recognition", *Neural Computation* 1(4):541–551, 1989. **Classical Euclidean CNN**

D. Pfau, S. Petersen, A. Agarwal, D. Barrett and K. Stachenfeld, "Spectral Inference Networks: Unifying Spectral Methods with Deep Learning", arXiv: 1806.02215, 2018.

L. Wiskott and T. J. Sejnowski, "Slow Feature Analysis: Unsupervised Learning of Invariances", Neural Computation 14(4): 715-770, 2002.

H. Sprekeler, "On the Relation of Slow Feature Analysis and Laplacian Eigenmaps", Neural Computation 23(12): 3287-3302, 2011. **Shows equivalence of SFA and Laplacian eigenmaps**