Learning To Traverse Image Manifolds

Piotr Dollár
Computer Science and Engineering
University of California, San Diego
pdollar@cs.ucsd.edu

Vincent Rabaud
Computer Science and Engineering
University of California, San Diego
vrabaud@cs.ucsd.edu

Serge Belongie
Computer Science and Engineering
University of California, San Diego
sjb@cs.ucsd.edu

Abstract

We present a new algorithm, Locally Smooth Manifold Learning (LSML), that learns a warping function from a point on an image manifold to its neighbors. Important characteristics of LSML include the ability to recover the structure of the manifold in sparsely populated regions and beyond the support of the provided data. Applications of our proposed technique include embedding with a natural out-of-sample extension and tasks such as tangent distance estimation, frame rate up-conversion, video compression and motion transfer.

Goal of LSML

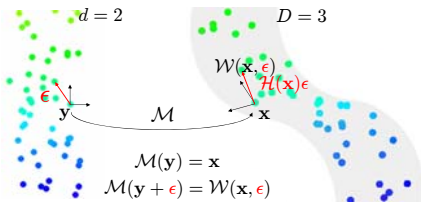
Learn to traverse manifolds

•Learn warping function $\mathcal{W}$

•Takes a point on a manifold and generates neighboring points:

$\mathcal{W}($  $) =$ 

Problem Formulation



$$\mathcal{M}(\mathbf{y}) = \mathbf{x}$$
$$\mathcal{M}(\mathbf{y} + \epsilon) = \mathcal{W}(\mathbf{x}, \epsilon)$$
$$\approx \mathbf{x} + \mathcal{H}(\mathbf{x})\epsilon$$

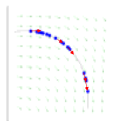
$$\boxed{\mathcal{H} : \mathbb{R}^D \rightarrow \mathbb{R}^{D \times d}}$$

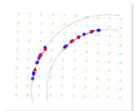
Intuition

• Suppose *training* tangent vectors given

• We could *learn* a regression from coordinates to the *d* tangent vectors

• We could then *test* regression in other places on the manifold

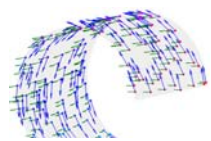




• Can learn beyond support of original data (*generalization*)

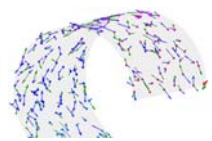
•We can also learn from multiple manifolds, generalize to new manifolds

Simple Case:



Can directly learn regression

Real Data:



Must simultaneously align vectors and learn regression

Error Function

D	dim. of original space
d	dim. of projected space
n	number of data points
$\mathbf{x}^i$	$[D \times 1]$ vector, $i \in [n]$, data point
$\mathcal{N}_i$	indices of neighbors of $\mathbf{x}^i$
$\mathcal{H}_\theta$	$\mathcal{H}_\theta : \mathbb{R}^D \rightarrow \mathbb{R}^{D \times d}$, parameterized by θ
$\boldsymbol{\epsilon}^{ij}$	$[d \times 1]$ vector, free parameters

$$\text{error}_1(\theta) = \min_{\{\boldsymbol{\epsilon}^{ij}\}} \sum_{i=1}^{n} \sum_{j \in \mathcal{N}^i} \left\| \mathcal{H}_\theta(\mathbf{x}^i)\boldsymbol{\epsilon}^{ij} - (\mathbf{x}^j - \mathbf{x}^i) \right\|_2^2$$

$$\Delta^i = \left[(\mathbf{x}^{j_1} - \mathbf{x}^i) \cdots (\mathbf{x}^{j_n} - \mathbf{x}^i) \right]$$
$$\text{SVD}(\Delta^i) = U^i \Sigma^i {V^i}^\top$$

$$\text{error}_2(\theta) = \min_{\{E^i\}} \sum_{i=1}^{n} \left\| \mathcal{H}_\theta(\mathbf{x}^i) E^i - U^i \Sigma^i \right\|_F^2 = \text{error}_1(\theta)$$

The new formulation is equivalent to trying to match a weighted local PCA basis.

Linearization

For practicality, we use a linear parameterization of $\mathcal{H}$:

$$\mathcal{H}_\Theta(\mathbf{x}^i) = \left[\Theta^1 \mathbf{f}^i \cdots \Theta^D \mathbf{f}^i \right]^\top$$

$$\text{error}_{lin}(\theta) = \min_{\{E^i\}} \sum_{i=1}^{n} \sum_{k=1}^{D} \left\| \mathbf{f}^{i\top} \Theta^{k\top} E^i - U_k^i \cdot \Sigma^i \right\|_2^2$$

Radial basis functions:

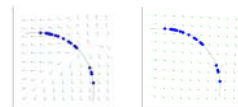
A linear combinations of RBFs can represent an arbitrary function. The number of basis functions controls smoothness.

$$\mathbf{f}_j^i = \exp(\left\| \mathbf{x}^i - \boldsymbol{\mu}^j \right\|_2^2 / 2\sigma^2)$$

Run K-means on the $\mathbf{x}^i$ with f clusters, set μ^j to the cluster centers. Set σ to twice the average distance between a cluster and its neighbor.

Regularization applied to avoid degenerate solutions:

$$\text{error}'_{lin}(\theta) = \text{error}_{lin}(\theta) + \lambda_E \sum_{i=1}^{n} \left\| E^i \right\|_F^2 + \lambda_\theta \sum_{k=1}^{D} \left\| \Theta^k \right\|_F^2$$



Without and with regularization

Minimization Procedure

$\forall k$, initialize Θ^k to a random $d \times f$ matrix.
while $\text{error}_{lin}(\theta)$ still decreases **do**

$\forall i$, solve for the best E^i given the Θ^ks:

$$E^i = \begin{bmatrix} \mathbf{f}^{i\top} \Theta^{1\top} \\ \vdots \\ \mathbf{f}^{i\top} \Theta^{D\top} \end{bmatrix}^+ \begin{bmatrix} U_1^i \cdot \Sigma^i \\ \vdots \\ U_D^i \cdot \Sigma^i \end{bmatrix}$$

$\forall k$, solve for the best Θ^k given the E^i's:

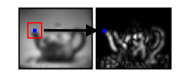
$$\text{vec}\left({\Theta^k}^\top\right) = \begin{bmatrix} {E^1}^\top \otimes \mathbf{f}^{1\top} \\ \vdots \\ {E^n}^\top \otimes \mathbf{f}^{n\top} \end{bmatrix}^+ \begin{bmatrix} \Sigma^i U_k^{1\top} \\ \vdots \\ \Sigma^i U_k^{n\top} \end{bmatrix}$$

end while

Image Case

Per patch approach:
Avoid curse of dimensionality (D=10^6)
Computationally efficient
One image sequence provides lots of training data

Assume each pixel in tangent image $\mathcal{H}_{\cdot k}(\mathbf{x})$ can be computed from a $s \times s$ image patch. Rewrite each image $\mathbf{x}^i \in \mathbb{R}^D$ as a $s^2 \times D$ matrix X^i.
(in practice compute features per patch: $X^i \mapsto F^i$)



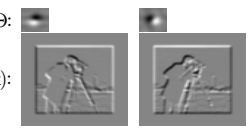
Now, $\mathcal{H}_\theta(\mathbf{x}^i) = (\Theta F^i)^\top$ where Θ is $d \times f$.

$$\text{error}_{img}(\Theta) = \min_{\{E^i\}} \sum_{i=1}^{n} \left\| {F^i}^\top \Theta^\top E^i - U^i \Sigma^i \right\|_F^2$$

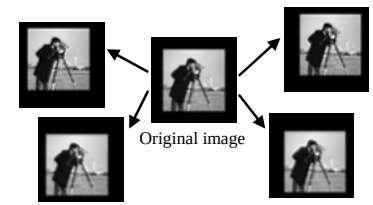
Application: Translation ($F^i = X^i$)

learned Θ:

$\mathcal{H}_\theta(\mathbf{x})$:

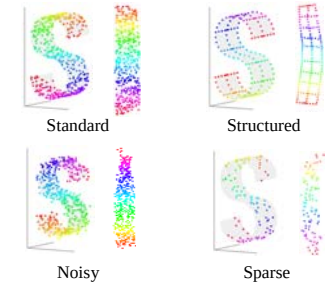


← convolved



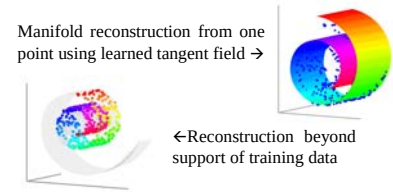
LSML Results

Embedding results on challenging versions of S-curve



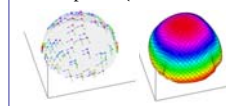
Manifold reconstruction results for Swiss-Roll

Manifold reconstruction from one point using learned tangent field →



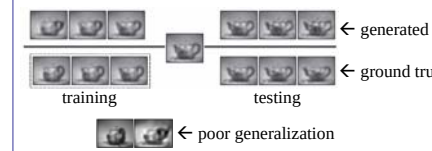
←Reconstruction beyond support of training data

Hemisphere (non isometric manifold)



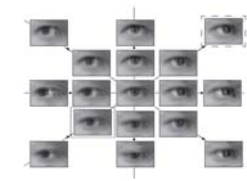
←Tangent field learned correctly; Reconstruction good even where little data

Out of plane rotation



Iris Motion

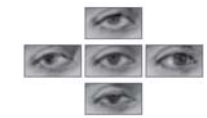
Tangent Images:



Images generated from central image
-inner 8 are within support of training
-outer 8 are outside support of training

Extrapolation:

beyond support of data



novel image