A Unifying Probabilistic Perspective on Spectral Approaches to Dimensionality Reduction

Neil D. Lawrence

Departments of Neuro- and Computer Science, University of Sheffield, U.K.

Talk at Workshop on Manifold Learning
Hausdorff Research Institute for Mathematics, Bonn

31st May 2011

Outline

Notation

p	data dimensionality	
q	latent dimensionality	
n	number of data points	
$\mathbf{Y}$	*design matrix* containing our data	$n \times p$
$\mathbf{X}$	matrix of latent variables	$n \times q$
$\mathbf{D}$	matrix of interpoint squared distances	$n \times n$
$\mathbf{K}$	similarities/covariance/kernel	$n \times n$
$\mathbf{L}$	Laplacian matrix	$n \times n$

Row vector from matrix $\mathbf{A}$ given by $\mathbf{a}_{i,:}$ column vector $\mathbf{a}_{:,j}$ and element given by $a_{i,j}$.

Outline

Distances and Similarities

- Typical scenario, a data set, **Y** stored in a matrix of dimension $n \times p$.
- Proximity data: a data set in form of distances, **D**, orsimilarities **K**. These matrices are dimension $n \times n$.
  - Similarity matrices have large entries when data points are close.
  - Distance matrices have large entries when points are far apart.

Multidimensional Scaling

- Multidimensional scaling (MDS) algorithms are dimensionality reduction for proximity matrices.
- We can move between similarity and squared distance as follows $d_{i,j} = k_{i,i} - 2k_{i,j} + k_{j,j}$.
  - In MDS this is known as the standard transformation (Mardia et al., 1979).
  - If $k_{i,j} = k(\mathbf{y}_{i,:}, \mathbf{y}_{j,:})$ is a "kernel" this is the "distance in feature space" (Schölkopf and Smola, 2001).
  - If $k_{i,j}$ is an element from a covariance matrix $\mathbf{K}$, it is the *expected squared distance* between two samples with that covariance.

Note: Centering and Squared Distances

- Consider matrix form of squared distance,

$$\mathbf{D} = \text{diag}\left(\mathbf{K}\right)\mathbf{1}^\top - 2\mathbf{K} + \mathbf{1}\text{diag}\left(\mathbf{K}\right)^\top.$$

- A Centering matrix has the form

$$\mathbf{H} = \mathbf{I} - n^{-1}\mathbf{1}\mathbf{1}^\top : \quad \mathbf{H}\mathbf{1} = \mathbf{0}$$

- This implies:

$$-\frac{1}{2}\mathbf{H}\mathbf{D}\mathbf{H} = \mathbf{H}\mathbf{K}\mathbf{H}.$$

- i.e. centered distance matrix is closely related to centred similarity/kernel.

Spectral Dimensionality Reduction in Machine Learning

- Spectral approach to dimensionality reduction.
  1. Convert data to a matrix of dimension $n \times n$.
  2. Visualize data with eigenvectors of matrix.
- Examples:
  - Isomap (Tenenbaum et al., 2000),
  - locally linear embeddings (LLE, Roweis and Saul, 2000),
  - Laplacian eigenmaps (LE, Belkin and Niyogi, 2003) and
  - maximum variance unfolding (MVU, Weinberger et al., 2004).
  - Also kernel PCA (Schölkopf et al., 1998; Ham et al., 2004).

Classical Multidimensional Scaling Perspective

- Classical multidimensional scaling (CMDS)
  1. Compute an $n \times n$ squared distance matrix, $\mathbf{D}$.
  2. Form the centered "similarity matrix" $\mathbf{HKH} = -\frac{1}{2}\mathbf{HDH}$.
  3. Visualize through q principal eigenvectors (as latent matrix $\mathbf{X}$).
- This algorithm matches squared distances computed in $\mathbf{X}$ to those computed in $\mathbf{Y}$ through an L1 error.
- Our Argument:
  - Main innovation in ML work: how to compute the squared distance matrix $\mathbf{D}$.

This Talk

- Introduce probabilistic approach to constructing squared distance matrices.
- Relate isomap, LLE, LE and MVU to the approach.
- Wrap spectral methods in a unifying perspective of *Gaussian random fields* and CMDS.

Outline

- Review
- Maximum Entropy Unfolding
- Relation to Laplacian Eigenmaps
- Relation to Locally Linear Embedding
- Relation to Isomap
- Relation to GP-LVM
- Experiments
- Discussion and Conclusions

Spectral Approaches

- Standard classical MDS gives a *linear* embedding in the Euclidean space implied by $\mathbf{D}$.
- This implies a linear transformation between $\mathbf{X}$ and $\mathbf{Y}$ (if squared distances are computed directly in $\mathbf{Y}$).
- Spectral approaches in machine learning give a *nonlinear* relationship between the data and the distances.
- This is done by not computing $\mathbf{D}$ directly in the space of $\mathbf{Y}$.
- This is very clear for kernel PCA, where $\mathbf{D}$ is computed in a feature space derived from $\mathbf{Y}$.

Spectral Approaches

- Standard classical MDS gives a *linear* embedding in the Euclidean space implied by $\mathbf{D}$.
- This implies a linear transformation between $\mathbf{X}$ and $\mathbf{Y}$ (if squared distances are computed directly in $\mathbf{Y}$).
- Spectral approaches in machine learning give a *nonlinear* relationship between the data and the distances.
- This is done by not computing $\mathbf{D}$ directly in the space of $\mathbf{Y}$.
- This is very clear for kernel PCA, where $\mathbf{D}$ is computed in a feature space derived from $\mathbf{Y}$.

Spectral Approaches

- Standard classical MDS gives a *linear* embedding in the Euclidean space implied by $\mathbf{D}$.
- This implies a linear transformation between $\mathbf{X}$ and $\mathbf{Y}$ (if squared distances are computed directly in $\mathbf{Y}$).
- Spectral approaches in machine learning give a *nonlinear* relationship between the data and the distances.
- This is done by not computing $\mathbf{D}$ directly in the space of $\mathbf{Y}$.
- This is very clear for kernel PCA, where $\mathbf{D}$ is computed in a feature space derived from $\mathbf{Y}$.

Spectral Approaches

- Standard classical MDS gives a *linear* embedding in the Euclidean space implied by $\mathbf{D}$.
- This implies a linear transformation between $\mathbf{X}$ and $\mathbf{Y}$ (if squared distances are computed directly in $\mathbf{Y}$).
- Spectral approaches in machine learning give a *nonlinear* relationship between the data and the distances.
- This is done by not computing $\mathbf{D}$ directly in the space of $\mathbf{Y}$.
- This is very clear for kernel PCA, where $\mathbf{D}$ is computed in a feature space derived from $\mathbf{Y}$.

Spectral Approaches

- Standard classical MDS gives a *linear* embedding in the Euclidean space implied by $\mathbf{D}$.
- This implies a linear transformation between $\mathbf{X}$ and $\mathbf{Y}$ (if squared distances are computed directly in $\mathbf{Y}$).
- Spectral approaches in machine learning give a *nonlinear* relationship between the data and the distances.
- This is done by not computing $\mathbf{D}$ directly in the space of $\mathbf{Y}$.
- This is very clear for kernel PCA, where $\mathbf{D}$ is computed in a feature space derived from $\mathbf{Y}$.

Kernel PCA

- Kernel PCA squared distance is defined through a kernel:

$$d_{i,j} = k(\mathbf{y}_{i,:}, \mathbf{y}_{i,:}) - 2k(\mathbf{y}_{i,:}, \mathbf{y}_{j,:}) - k(\mathbf{y}_{j,:}, \mathbf{y}_{j,:}) \qquad (1)$$

- $k(\cdot, \cdot)$ is a Mercer kernel (Ham et al., 2004).
- Kernel PCA (KPCA) recovers an $\mathbf{x}_{i,:}$ and a mapping from $\mathbf{Y}$ to $\mathbf{X}$ space.
- The mapping is induced through the choice of the *Mercer kernel*.

Kernel PCA

- Kernel PCA squared distance is defined through a kernel:

$$d_{i,j} = k(\mathbf{y}_{i,:}, \mathbf{y}_{i,:}) - 2k(\mathbf{y}_{i,:}, \mathbf{y}_{j,:}) - k(\mathbf{y}_{j,:}, \mathbf{y}_{j,:}) \qquad (1)$$

- $k(\cdot, \cdot)$ is a Mercer kernel (Ham et al., 2004).
- Kernel PCA (KPCA) recovers an $\mathbf{x}_{i,:}$ and a mapping from $\mathbf{Y}$ to $\mathbf{X}$ space.
- The mapping is induced through the choice of the *Mercer kernel*.

Kernel PCA

- Kernel PCA squared distance is defined through a kernel:

$$d_{i,j} = k(\mathbf{y}_{i,:}, \mathbf{y}_{i,:}) - 2k(\mathbf{y}_{i,:}, \mathbf{y}_{j,:}) - k(\mathbf{y}_{j,:}, \mathbf{y}_{j,:}) \qquad (1)$$

- $k(\cdot, \cdot)$ is a Mercer kernel (Ham et al., 2004).
- Kernel PCA (KPCA) recovers an $\mathbf{x}_{i,:}$ and a mapping from $\mathbf{Y}$ to $\mathbf{X}$ space.
- The mapping is induced through the choice of the *Mercer kernel*.

Kernel PCA

- Kernel PCA squared distance is defined through a kernel:

$$d_{i,j} = k(\mathbf{y}_{i,:}, \mathbf{y}_{i,:}) - 2k(\mathbf{y}_{i,:}, \mathbf{y}_{j,:}) - k(\mathbf{y}_{j,:}, \mathbf{y}_{j,:}) \quad (1)$$

- $k(\cdot, \cdot)$ is a Mercer kernel (Ham et al., 2004).
- Kernel PCA (KPCA) recovers an $\mathbf{x}_{i,:}$ and a mapping from $\mathbf{Y}$ to $\mathbf{X}$ space.
- The mapping is induced through the choice of the *Mercer kernel*.

Classical MDS and KPCA

- CMDS procedure performs eigenvalue problem on
$$\mathbf{B} = \mathbf{HKH}.$$
- This matches the KPCA algorithm (Schölkopf et al., 1998)[1].
- **However**, for the commonly used exponentiated quadratic kernel,
$$k(y_{i,:}, y_{j,:}) = \exp(-\gamma \left\|\mathbf{y}_{i,:} - \mathbf{y}_{j,:}\right\|_2^2),$$
KPCA actually *expands* the feature space (Weinberger et al., 2004).

[1] Kernel PCA also has an interpretation as a particular form of *metric* multidimensional scaling, see Williams (2001) for details.

Classical MDS and KPCA

- CMDS procedure performs eigenvalue problem on

$$\mathbf{B} = \mathbf{HKH}.$$

- This matches the KPCA algorithm (Schölkopf et al., 1998)[1].
- **However**, for the commonly used exponentiated quadratic kernel,

$$k(y_{i,:}, y_{j,:}) = \exp(-\gamma \left\| \mathbf{y}_{i,:} - \mathbf{y}_{j,:} \right\|_2^2),$$

KPCA actually *expands* the feature space (Weinberger et al., 2004).

[1] Kernel PCA also has an interpretation as a particular form of *metric* multidimensional scaling, see Williams (2001) for details.

Classical MDS and KPCA

- CMDS procedure performs eigenvalue problem on

$$\mathbf{B} = \mathbf{HKH}.$$

- This matches the KPCA algorithm (Schölkopf et al., 1998)[1].
- **However**, for the commonly used exponentiated quadratic kernel,

$$k(y_{i,:}, y_{j,:}) = \exp(-\gamma \left\| \mathbf{y}_{i,:} - \mathbf{y}_{j,:} \right\|_2^2),$$

KPCA actually *expands* the feature space (Weinberger et al., 2004).

[1] Kernel PCA also has an interpretation as a particular form of *metric* multidimensional scaling, see Williams (2001) for details.

Maximum Variance Unfolding

Learn a "Kernel" for Dimensionality Reduction

- In maximum variance unfolding (MVU) (Weinberger et al., 2004): learn a "kernel matrix" that will allow for dimensionality reduction.
- Preserve only *local* proximity relationships in the data.
  - Take a set of neighbors.
  - Construct a kernel matrix where only distances between neighbors match data distances.

Maximum Variance Unfolding

Learn a "Kernel" for Dimensionality Reduction

- In maximum variance unfolding (MVU) (Weinberger et al., 2004): learn a "kernel matrix" that will allow for dimensionality reduction.
- Preserve only *local* proximity relationships in the data.
  - Take a set of neighbors.
  - Construct a kernel matrix where only distances between neighbors match data distances.

Maximum Variance Unfolding

Learn a "Kernel" for Dimensionality Reduction

- In maximum variance unfolding (MVU) (Weinberger et al., 2004): learn a "kernel matrix" that will allow for dimensionality reduction.
- Preserve only *local* proximity relationships in the data.
  - Take a set of neighbors.
  - Construct a kernel matrix where only distances between neighbors match data distances.

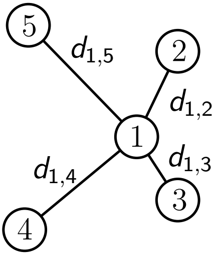
Maximum Variance Unfolding

Learn a "Kernel" for Dimensionality Reduction

- In maximum variance unfolding (MVU) (Weinberger et al., 2004): learn a "kernel matrix" that will allow for dimensionality reduction.
- Preserve only *local* proximity relationships in the data.
  - Take a set of neighbors.
  - Construct a kernel matrix where only distances between neighbors match data distances.

Maximum Variance Unfolding

- Optimize elements of $\mathbf{K}$ by maximizing[2] $\text{tr}(\mathbf{K})$.



- Subject to squared distance constraints between neighbors

$$d_{i,j} = k_{i,i} - 2k_{i,j} + k_{j,j}$$

[2] The trace is the *total variance* of the data in feature space

Maximum Entropy Unfolding

Our Contribution

- Maximize *entropy* instead of variance (Jaynes, 1986): MEU.
- Entropy and variance are closely related.
- Maximum entropy leads to a *probabilistic model*.
- Each spectral approach approximates MEU in some way.

Maximum Entropy Unfolding

Our Contribution

- Maximize *entropy* instead of variance (Jaynes, 1986): MEU.
- Entropy and variance are closely related.
- Maximum entropy leads to a *probabilistic model*.
- Each spectral approach approximates MEU in some way.

Maximum Entropy Unfolding

Our Contribution

- Maximize *entropy* instead of variance (Jaynes, 1986): MEU.
- Entropy and variance are closely related.
- Maximum entropy leads to a *probabilistic model*.
- Each spectral approach approximates MEU in some way.

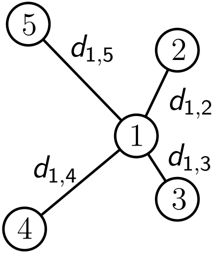
Maximum Entropy Unfolding

Our Contribution

- Maximize *entropy* instead of variance (Jaynes, 1986): MEU.
- Entropy and variance are closely related.
- Maximum entropy leads to a *probabilistic model*.
- Each spectral approach approximates MEU in some way.

Maximum Entropy Unfolding

- Find distribution with maximum entropy subject to constraints on *moments*.

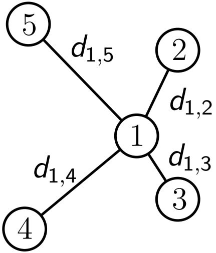


- MEU constraints are on expected distances between neighbors.

$$d_{i,j} = \left\langle \mathbf{y}_{i,:}^{\top}\mathbf{y}_{i,:} \right\rangle - 2\left\langle \mathbf{y}_{i,:}^{\top}\mathbf{y}_{j,:} \right\rangle + \left\langle \mathbf{y}_{j,:}^{\top}\mathbf{y}_{j,:} \right\rangle$$

Maximum Entropy Unfolding

- Find distribution with maximum entropy subject to constraints on *moments*.



- MEU constraints are on expected distances between neighbors.

$$d_{i,j} = k_{i,i} - 2k_{i,j} + k_{j,j}$$

which can be written in terms of the covariance.

Gaussian Random Field

- The maximum entropy probability distribution is a *Gaussian random field*

$$p(\mathbf{Y}) = \prod_{j=1}^{p} \frac{1}{|\mathbf{K}|^{\frac{1}{2}} (2\pi)^{\frac{n}{2}}} \exp\left(-\frac{1}{2}\mathbf{y}_{:,j}^{\top}\mathbf{K}^{-1}\mathbf{y}_{:,j}\right),$$

- Covariance matrix is

$$\mathbf{K} = (\mathbf{L} + \gamma\mathbf{I})^{-1}$$

.

- Where $\mathbf{L}$ is the *Laplacian* matrix associated with the neighborhood graph.
- Off diagonal elements of the Laplacian are Lagrange multipliers from moment constraints.
- On diagonal elements given by negative sum of off-diagonal ($\mathbf{L1} = \mathbf{0}$).

Outline

Relationship to Laplacian Eigenmaps

- Laplacian eigenmaps (Belkin and Niyogi, 2003): graph Laplacian is specified across the data points.
- Laplacian has exactly the same form as our matrix $\mathbf{L}$.
- Parameters of the Laplacian are set either as constant or according to the distance between two points.
- Smallest eigenvectors of this Laplacian are then used for visualizing the data.

Relationship to Laplacian Eigenmaps

- Laplacian eigenmaps (Belkin and Niyogi, 2003): graph Laplacian is specified across the data points.
- Laplacian has exactly the same form as our matrix $\mathbf{L}$.
- Parameters of the Laplacian are set either as constant or according to the distance between two points.
- Smallest eigenvectors of this Laplacian are then used for visualizing the data.

Relationship to Laplacian Eigenmaps

- Laplacian eigenmaps (Belkin and Niyogi, 2003): graph Laplacian is specified across the data points.
- Laplacian has exactly the same form as our matrix **L**.
- Parameters of the Laplacian are set either as constant or according to the distance between two points.
- Smallest eigenvectors of this Laplacian are then used for visualizing the data.

Relationship to Laplacian Eigenmaps

- Laplacian eigenmaps (Belkin and Niyogi, 2003): graph Laplacian is specified across the data points.
- Laplacian has exactly the same form as our matrix $\mathbf{L}$.
- Parameters of the Laplacian are set either as constant or according to the distance between two points.
- Smallest eigenvectors of this Laplacian are then used for visualizing the data.

Smallest Eigenvalues of Laplacian

- Eigendecomposition of the covariance is

$$\mathbf{K} = \mathbf{U}\boldsymbol{\Lambda}\mathbf{U}^\top$$

- Eigendecomposition of the Laplacian is

$$\mathbf{L} = \mathbf{U}\left(\boldsymbol{\Lambda}^{-1} - \gamma\mathbf{I}\right)\mathbf{U}^\top$$

- Principal eigenvalues of $\mathbf{K}$ are smallest eigenvalues of $\mathbf{L}$.
  - (smallest eigenvalue of $\mathbf{L}$ is zero, but this is removed by the centering operation on $\mathbf{K}$, or discarded in LE)

Smallest Eigenvalues of Laplacian

- Eigendecomposition of the covariance is

$$\mathbf{K} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^\top$$

- Eigendecomposition of the Laplacian is

$$\mathbf{L} = \mathbf{U}\left(\mathbf{\Lambda}^{-1} - \gamma\mathbf{I}\right)\mathbf{U}^\top$$

- Principal eigenvalues of $\mathbf{K}$ are smallest eigenvalues of $\mathbf{L}$.
  - (smallest eigenvalue of $\mathbf{L}$ is zero, but this is removed by the centering operation on $\mathbf{K}$, or discarded in LE)

Smallest Eigenvalues of Laplacian

- Eigendecomposition of the covariance is

$$\mathbf{K} = \mathbf{U}\boldsymbol{\Lambda}\mathbf{U}^\top$$

- Eigendecomposition of the Laplacian is

$$\mathbf{L} = \mathbf{U}\left(\boldsymbol{\Lambda}^{-1} - \gamma\mathbf{I}\right)\mathbf{U}^\top$$

- Principal eigenvalues of $\mathbf{K}$ are smallest eigenvalues of $\mathbf{L}$.
  - (smallest eigenvalue of $\mathbf{L}$ is zero, but this is removed by the centering operation on $\mathbf{K}$, or discarded in LE)

Laplacian Eigenmaps

- Set parameters of Laplacian.
- Perform CMDS on the implied matrix $\mathbf{K}$.
  1. No constraints are imposed in Laplacian eigenmaps so distances will not be preserved.
  2. LE gains significant computational advantage by not representing the covariance matrix explicitly.
  3. No matrix inverses required, eigenvalue problem sparse.

Laplacian Eigenmaps

- Set parameters of Laplacian.
- Perform CMDS on the implied matrix $\mathbf{K}$.
  1. No constraints are imposed in Laplacian eigenmaps so distances will not be preserved.
  2. LE gains significant computational advantage by not representing the covariance matrix explicitly.
  3. No matrix inverses required, eigenvalue problem sparse.

Laplacian Eigenmaps

- Set parameters of Laplacian.
- Perform CMDS on the implied matrix $\mathbf{K}$.
  1. No constraints are imposed in Laplacian eigenmaps so distances will not be preserved.
  2. LE gains significant computational advantage by not representing the covariance matrix explicitly.
  3. No matrix inverses required, eigenvalue problem sparse.

Laplacian Eigenmaps

- Set parameters of Laplacian.
- Perform CMDS on the implied matrix **K**.
    1. No constraints are imposed in Laplacian eigenmaps so distances will not be preserved.
    2. LE gains significant computational advantage by not representing the covariance matrix explicitly.
    3. No matrix inverses required, eigenvalue problem sparse.

Laplacian Eigenmaps

- Set parameters of Laplacian.
- Perform CMDS on the implied matrix $\mathbf{K}$.
  1. No constraints are imposed in Laplacian eigenmaps so distances will not be preserved.
  2. LE gains significant computational advantage by not representing the covariance matrix explicitly.
  3. No matrix inverses required, eigenvalue problem sparse.

Outline

Locally Linear Embedding

- The Laplacian should be constrained positive definite.
- This constraint can be imposed by factorizing it as

$$\mathbf{L} = \mathbf{M}\mathbf{M}^\top$$

- To ensure it is a Laplacian, we need to constrain $\mathbf{M}^\top\mathbf{1} = \mathbf{0}$ giving $\mathbf{L}\mathbf{1} = \mathbf{0}$.
  - i.e. $m_{i,i} = -\sum_{j\in\mathcal{N}(i)} m_{j,i}$
  - Set $m_{j,i} = 0$ if $j \notin \mathcal{N}(i)$.

Locally Linear Embedding

- The Laplacian should be constrained positive definite.
- This constraint can be imposed by factorizing it as

$$\mathbf{L} = \mathbf{M}\mathbf{M}^\top$$

- To ensure it is a Laplacian, we need to constrain $\mathbf{M}^\top\mathbf{1} = \mathbf{0}$ giving $\mathbf{L}\mathbf{1} = \mathbf{0}$.
  - i.e. $m_{i,i} = -\sum_{j\in\mathcal{N}(i)} m_{j,i}$
  - Set $m_{j,i} = 0$ if $j \notin \mathcal{N}(i)$.

Locally Linear Embedding

- The Laplacian should be constrained positive definite.
- This constraint can be imposed by factorizing it as

$$\mathbf{L} = \mathbf{M}\mathbf{M}^\top$$

- To ensure it is a Laplacian, we need to constrain $\mathbf{M}^\top \mathbf{1} = \mathbf{0}$ giving $\mathbf{L}\mathbf{1} = \mathbf{0}$.
  - i.e. $m_{i,i} = -\sum_{j \in \mathcal{N}(i)} m_{j,i}$
  - Set $m_{j,i} = 0$ if $j \notin \mathcal{N}(i)$.

Locally Linear Embedding

- The Laplacian should be constrained positive definite.
- This constraint can be imposed by factorizing it as

$$\mathbf{L} = \mathbf{M}\mathbf{M}^\top$$

- To ensure it is a Laplacian, we need to constrain $\mathbf{M}^\top\mathbf{1} = \mathbf{0}$ giving $\mathbf{L}\mathbf{1} = \mathbf{0}$.
  - i.e. $m_{i,i} = -\sum_{j\in\mathcal{N}(i)} m_{j,i}$
  - Set $m_{j,i} = 0$ if $j \notin \mathcal{N}(i)$.

Locally Linear Embedding

- The Laplacian should be constrained positive definite.
- This constraint can be imposed by factorizing it as

$$\mathbf{L} = \mathbf{M}\mathbf{M}^\top$$

- To ensure it is a Laplacian, we need to constrain $\mathbf{M}^\top \mathbf{1} = \mathbf{0}$ giving $\mathbf{L}\mathbf{1} = \mathbf{0}$.
  - i.e. $m_{i,i} = -\sum_{j \in \mathcal{N}(i)} m_{j,i}$
  - Set $m_{j,i} = 0$ if $j \notin \mathcal{N}(i)$.

Locally Linear Embedding

- Locally linear embeddings (Roweis and Saul, 2000) are then a specific case of MEU where
  1. The diagonal sums, $m_{i,i}$, are further constrained to unity.
  2. Model parameters found by maximizing *pseudolikelihood* of the data.

Locally Linear Embedding

- Locally linear embeddings (Roweis and Saul, 2000) are then a specific case of MEU where
  1. The diagonal sums, $m_{i,i}$, are further constrained to unity.
  2. Model parameters found by maximizing *pseudolikelihood* of the data.

Locally Linear Embedding

- Locally linear embeddings (Roweis and Saul, 2000) are then a specific case of MEU where
  1. The diagonal sums, $m_{i,i}$, are further constrained to unity.
  2. Model parameters found by maximizing *pseudolikelihood* of the data.

Point One

- For unit diagonals we have $\mathbf{M} = \mathbf{I} - \mathbf{W}$.
- Here the off diagonal sparsity pattern of $\mathbf{W}$ matches $\mathbf{M}$.
- Thus
$$(\mathbf{I} - \mathbf{W})^\top \mathbf{1} = \mathbf{0}.$$
- LLE proscribes that the smallest eigenvectors of
$$(\mathbf{I} - \mathbf{W})(\mathbf{I} - \mathbf{W})^\top = \mathbf{M}\mathbf{M}^\top = \mathbf{L}$$
(like Laplacian Eigenmaps).
- Equivalent to CMDS on the GRF described by $\mathbf{L}$.

Point One

- For unit diagonals we have $\mathbf{M} = \mathbf{I} - \mathbf{W}$.
- Here the off diagonal sparsity pattern of $\mathbf{W}$ matches $\mathbf{M}$.
- Thus
$$(\mathbf{I} - \mathbf{W})^\top \mathbf{1} = \mathbf{0}.$$
- LLE proscribes that the smallest eigenvectors of
$$(\mathbf{I} - \mathbf{W})(\mathbf{I} - \mathbf{W})^\top = \mathbf{M}\mathbf{M}^\top = \mathbf{L}$$
(like Laplacian Eigenmaps).
- Equivalent to CMDS on the GRF described by $\mathbf{L}$.

Point One

- For unit diagonals we have $\mathbf{M} = \mathbf{I} - \mathbf{W}$.
- Here the off diagonal sparsity pattern of $\mathbf{W}$ matches $\mathbf{M}$.
- Thus

$$(\mathbf{I} - \mathbf{W})^\top \mathbf{1} = \mathbf{0}.$$

- LLE proscribes that the smallest eigenvectors of

$$(\mathbf{I} - \mathbf{W})(\mathbf{I} - \mathbf{W})^\top = \mathbf{M}\mathbf{M}^\top = \mathbf{L}$$

(like Laplacian Eigenmaps).
- Equivalent to CMDS on the GRF described by $\mathbf{L}$.

Point One

- For unit diagonals we have $\mathbf{M} = \mathbf{I} - \mathbf{W}$.
- Here the off diagonal sparsity pattern of $\mathbf{W}$ matches $\mathbf{M}$.
- Thus
$$(\mathbf{I} - \mathbf{W})^\top \mathbf{1} = \mathbf{0}.$$
- LLE proscribes that the smallest eigenvectors of
$$(\mathbf{I} - \mathbf{W})(\mathbf{I} - \mathbf{W})^\top = \mathbf{M}\mathbf{M}^\top = \mathbf{L}$$
(like Laplacian Eigenmaps).
- Equivalent to CMDS on the GRF described by $\mathbf{L}$.

Point One

- For unit diagonals we have $\mathbf{M} = \mathbf{I} - \mathbf{W}$.
- Here the off diagonal sparsity pattern of $\mathbf{W}$ matches $\mathbf{M}$.
- Thus
$$(\mathbf{I} - \mathbf{W})^\top \mathbf{1} = \mathbf{0}.$$

- LLE proscribes that the smallest eigenvectors of
$$(\mathbf{I} - \mathbf{W})(\mathbf{I} - \mathbf{W})^\top = \mathbf{M}\mathbf{M}^\top = \mathbf{L}$$
(like Laplacian Eigenmaps).
- Equivalent to CMDS on the GRF described by $\mathbf{L}$.

Second Point

- Pseudolikelihood approximation (see e.g. Koller and Friedman, 2009, pg 970): product of the conditional densities:

$$p(\mathbf{Y}) \approx \prod_{i=1}^{n} p(\mathbf{y}_{i,:}|\mathbf{Y}_{\backslash i}),$$

  $\mathbf{Y}_{\backslash i}$ represents data other than the ith point.
- True likelihood is proportional to this but requires renormalization.
- In pseudolikelihood normalization is ignored.

Second Point

- Pseudolikelihood approximation (see e.g. Koller and Friedman, 2009, pg 970): product of the conditional densities:

$$p(\mathbf{Y}) \approx \prod_{i=1}^{n} p(\mathbf{y}_{i,:}|\mathbf{Y}_{\backslash i}),$$

  $\mathbf{Y}_{\backslash i}$ represents data other than the ith point.
- True likelihood is proportional to this but requires renormalization.
- In pseudolikelihood normalization is ignored.

Second Point

- Pseudolikelihood approximation (see e.g. Koller and Friedman, 2009, pg 970): product of the conditional densities:

$$p(\mathbf{Y}) \approx \prod_{i=1}^{n} p(\mathbf{y}_{i,:}|\mathbf{Y}_{\backslash i}),$$

  $\mathbf{Y}_{\backslash i}$ represents data other than the ith point.
- True likelihood is proportional to this but requires renormalization.
- In pseudolikelihood normalization is ignored.

Conditionals

- Factors in the GRF are the conditionals,

$$p(\mathbf{y}_{i,:}|\mathbf{Y}_{\backslash i}) = \left(\frac{m_{i,i}^2}{2\pi}\right)^{\frac{p}{2}} \exp\left(-\frac{m_{i,i}^2}{2}\left\|\mathbf{y}_{i,:} - \sum_{j\in\mathcal{N}(i)} \frac{w_{j,i}}{m_{i,i}}\mathbf{y}_{j,:}\right\|_2^2\right).$$

- Maximizing each conditional is equivalent to optimizing LLE objective.
- Constraint that LLE weights sum to one arises naturally because $w_{j,i}/m_{i,i}$ and $m_{i,i} = \sum_{j\in\mathcal{N}(i)} w_{j,i}$.
- In LLE a *further* constraint is imposed $m_{i,i} = 1$.

Conditionals

- Factors in the GRF are the conditionals,

$$p(\mathbf{y}_{i,:}|\mathbf{Y}_{\backslash i}) = \left(\frac{m_{i,i}^2}{2\pi}\right)^{\frac{p}{2}} \exp\left(-\frac{m_{i,i}^2}{2}\left\|\mathbf{y}_{i,:} - \sum_{j\in\mathcal{N}(i)} \frac{w_{j,i}}{m_{i,i}}\mathbf{y}_{j,:}\right\|_2^2\right).$$

- Maximizing each conditional is equivalent to optimizing LLE objective.
- Constraint that LLE weights sum to one arises naturally because $w_{j,i}/m_{i,i}$ and $m_{i,i} = \sum_{j\in\mathcal{N}(i)} w_{j,i}$.
- In LLE a *further* constraint is imposed $m_{i,i} = 1$.

Conditionals

- Factors in the GRF are the conditionals,

$$p(\mathbf{y}_{i,:}|\mathbf{Y}_{\setminus i}) = \left(\frac{m_{i,i}^2}{2\pi}\right)^{\frac{p}{2}} \exp\left(-\frac{m_{i,i}^2}{2}\left\|\mathbf{y}_{i,:} - \sum_{j\in\mathcal{N}(i)} \frac{w_{j,i}}{m_{i,i}}\mathbf{y}_{j,:}\right\|_2^2\right).$$

- Maximizing each conditional is equivalent to optimizing LLE objective.
- Constraint that LLE weights sum to one arises naturally because $w_{j,i}/m_{i,i}$ and $m_{i,i} = \sum_{j\in\mathcal{N}(i)} w_{j,i}$.
- In LLE a *further* constraint is imposed $m_{i,i} = 1$.

Conditionals

- Factors in the GRF are the conditionals,

$$p(\mathbf{y}_{i,:}|\mathbf{Y}_{\backslash i}) = \left(\frac{m_{i,i}^2}{2\pi}\right)^{\frac{p}{2}} \exp\left(-\frac{m_{i,i}^2}{2}\left\|\mathbf{y}_{i,:} - \sum_{j\in\mathcal{N}(i)} \frac{w_{j,i}}{m_{i,i}}\mathbf{y}_{j,:}\right\|_2^2\right).$$

- Maximizing each conditional is equivalent to optimizing LLE objective.
- Constraint that LLE weights sum to one arises naturally because $w_{j,i}/m_{i,i}$ and $m_{i,i} = \sum_{j\in\mathcal{N}(i)} w_{j,i}$.
- In LLE a *further* constraint is imposed $m_{i,i} = 1$.

LLE Approximates MEU

- LLE is an approximation to maximum likelihood.
- Laplacian has factorized form.
- Pseudolikelihood also allows for relatively quick parameter estimation.
  - ignoring the partition function removes the need to invert to recover the covariance matrix.
  - LLE can be applied to larger data sets than MEU or MVU.

Note: The sparsity pattern in the Laplacian for LLE will not match that used in the Laplacian for the other algorithms due to the factorized representation.

LLE Approximates MEU

- LLE is an approximation to maximum likelihood.
- Laplacian has factorized form.
- Pseudolikelihood also allows for relatively quick parameter estimation.
  - ignoring the partition function removes the need to invert to recover the covariance matrix.
  - LLE can be applied to larger data sets than MEU or MVU.

Note: The sparsity pattern in the Laplacian for LLE will not match that used in the Laplacian for the other algorithms due to the factorized representation.

LLE Approximates MEU

- LLE is an approximation to maximum likelihood.
- Laplacian has factorized form.
- Pseudolikelihood also allows for relatively quick parameter estimation.
  - ignoring the partition function removes the need to invert to recover the covariance matrix.
  - LLE can be applied to larger data sets than MEU or MVU.

Note: The sparsity pattern in the Laplacian for LLE will not match that used in the Laplacian for the other algorithms due to the factorized representation.

LLE Approximates MEU

- LLE is an approximation to maximum likelihood.
- Laplacian has factorized form.
- Pseudolikelihood also allows for relatively quick parameter estimation.
  - ignoring the partition function removes the need to invert to recover the covariance matrix.
  - LLE can be applied to larger data sets than MEU or MVU.

Note: The sparsity pattern in the Laplacian for LLE will not match that used in the Laplacian for the other algorithms due to the factorized representation.

LLE Approximates MEU

- LLE is an approximation to maximum likelihood.
- Laplacian has factorized form.
- Pseudolikelihood also allows for relatively quick parameter estimation.
  - ignoring the partition function removes the need to invert to recover the covariance matrix.
  - LLE can be applied to larger data sets than MEU or MVU.

Note: The sparsity pattern in the Laplacian for LLE will not match that used in the Laplacian for the other algorithms due to the factorized representation.

LLE and PCA

- LLE is motivated by considering local linear embeddings of the data.
- Interestingly, as we increase the neighborhood size to $K = n - 1$ we do not recover PCA.
- But PCA is the "optimal" linear embedding!!
- LLE is optimizing a pseudolikelihood: in contrast the MEU algorithm, which LLE approximates, does recover PCA when $K = n - 1$.

LLE and PCA

- LLE is motivated by considering local linear embeddings of the data.
- Interestingly, as we increase the neighborhood size to $K = n - 1$ we do not recover PCA.
- But PCA is the "optimal" linear embedding!!
- LLE is optimizing a pseudolikelihood: in contrast the MEU algorithm, which LLE approximates, does recover PCA when $K = n - 1$.

LLE and PCA

- LLE is motivated by considering local linear embeddings of the data.
- Interestingly, as we increase the neighborhood size to $K = n - 1$ we do not recover PCA.
- But PCA is the "optimal" linear embedding!!
- LLE is optimizing a pseudolikelihood: in contrast the MEU algorithm, which LLE approximates, does recover PCA when $K = n - 1$.

LLE and PCA

- LLE is motivated by considering local linear embeddings of the data.
- Interestingly, as we increase the neighborhood size to $K = n - 1$ we do not recover PCA.
- But PCA is the "optimal" linear embedding!!
- LLE is optimizing a pseudolikelihood: in contrast the MEU algorithm, which LLE approximates, does recover PCA when $K = n - 1$.

Outline

Isomap

- Isomap (Tenenbaum et al., 2000) follows the CMDS framework.
- Sparse graph of distances is created.
- Fill in graph for non-neighbors with a shortest path algorithm.
- Element-wise square the matrix.
- Process this in the usual manner.

Isomap

- Isomap (Tenenbaum et al., 2000) follows the CMDS framework.
- Sparse graph of distances is created.
- Fill in graph for non-neighbors with a shortest path algorithm.
- Element-wise square the matrix.
- Process this in the usual manner.

Isomap

- Isomap (Tenenbaum et al., 2000) follows the CMDS framework.
- Sparse graph of distances is created.
- Fill in graph for non-neighbors with a shortest path algorithm.
- Element-wise square the matrix.
- Process this in the usual manner.

Isomap

- Isomap (Tenenbaum et al., 2000) follows the CMDS framework.
- Sparse graph of distances is created.
- Fill in graph for non-neighbors with a shortest path algorithm.
- Element-wise square the matrix.
- Process this in the usual manner.

Isomap

- Isomap (Tenenbaum et al., 2000) follows the CMDS framework.
- Sparse graph of distances is created.
- Fill in graph for non-neighbors with a shortest path algorithm.
- Element-wise square the matrix.
- Process this in the usual manner.

Compare with MEU

- Both MVU and MEU can be thought of as starting with a sparse graph of (squared) distances.
- Fill in other distances by maximizing the total variance/entropy.
- Interneighbor distances in this graph are preserved just like in isomap.
  1. For isomap the implied covariance can have negative eigenvalues (see (Weinberger et al., 2004)).
  2. Isomap is slower than LLE and LE: requires a dense eigenvalue problem and a shortest path algorithm.

Compare with MEU

- Both MVU and MEU can be thought of as starting with a sparse graph of (squared) distances.
- Fill in other distances by maximizing the total variance/entropy.
- Interneighbor distances in this graph are preserved just like in isomap.
    1. For isomap the implied covariance can have negative eigenvalues (see (Weinberger et al., 2004)).
    2. Isomap is slower than LLE and LE: requires a dense eigenvalue problem and a shortest path algorithm.

Compare with MEU

- Both MVU and MEU can be thought of as starting with a sparse graph of (squared) distances.
- Fill in other distances by maximizing the total variance/entropy.
- Interneighbor distances in this graph are preserved just like in isomap.
  1. For isomap the implied covariance can have negative eigenvalues (see (Weinberger et al., 2004)).
  2. Isomap is slower than LLE and LE: requires a dense eigenvalue problem and a shortest path algorithm.

Compare with MEU

- Both MVU and MEU can be thought of as starting with a sparse graph of (squared) distances.
- Fill in other distances by maximizing the total variance/entropy.
- Interneighbor distances in this graph are preserved just like in isomap.
  1. For isomap the implied covariance can have negative eigenvalues (see (Weinberger et al., 2004)).
  2. Isomap is slower than LLE and LE: requires a dense eigenvalue problem and a shortest path algorithm.

Compare with MEU

- Both MVU and MEU can be thought of as starting with a sparse graph of (squared) distances.
- Fill in other distances by maximizing the total variance/entropy.
- Interneighbor distances in this graph are preserved just like in isomap.
  1. For isomap the implied covariance can have negative eigenvalues (see (Weinberger et al., 2004)).
  2. Isomap is slower than LLE and LE: requires a dense eigenvalue problem and a shortest path algorithm.

Outline

Relationship to GP-LVM

- Gaussian Process latent variable models (Lawrence, 2005) also define Gaussian densities independently over the features.
- GP-LVMs construct a Gaussian process by specifying a covariance function (Mercer kernel) in $\mathbf{X}$.
- A Gauss Markov random field can be specified by a Gaussian process through appropriate covariance functions

$$k(x, x') = \exp(-\|x - x\prime\|_1)$$

- Inverse covariance will be sparse and based on neighborhood.
- In the GP-LVM the neighborhood is learnt by optimization of $\mathbf{X}$.

Relationship to GP-LVM

- Gaussian Process latent variable models (Lawrence, 2005) also define Gaussian densities independently over the features.
- GP-LVMs construct a Gaussian process by specifying a covariance function (Mercer kernel) in $\mathbf{X}$.
- A Gauss Markov random field can be specified by a Gaussian process through appropriate covariance functions

$$k(x, x') = \exp(-\|x - x\prime\|_1)$$

- Inverse covariance will be sparse and based on neighborhood.
- In the GP-LVM the neighborhood is learnt by optimization of $\mathbf{X}$.

Relationship to GP-LVM

- Gaussian Process latent variable models (Lawrence, 2005) also define Gaussian densities independently over the features.
- GP-LVMs construct a Gaussian process by specifying a covariance function (Mercer kernel) in $\mathbf{X}$.
- A Gauss Markov random field can be specified by a Gaussian process through appropriate covariance functions

$$k(x, x') = \exp(-\|x - x\prime\|_1)$$

- Inverse covariance will be sparse and based on neighborhood.
- In the GP-LVM the neighborhood is learnt by optimization of $\mathbf{X}$.

Relationship to GP-LVM

- Gaussian Process latent variable models (Lawrence, 2005) also define Gaussian densities independently over the features.
- GP-LVMs construct a Gaussian process by specifying a covariance function (Mercer kernel) in **X**.
- A Gauss Markov random field can be specified by a Gaussian process through appropriate covariance functions

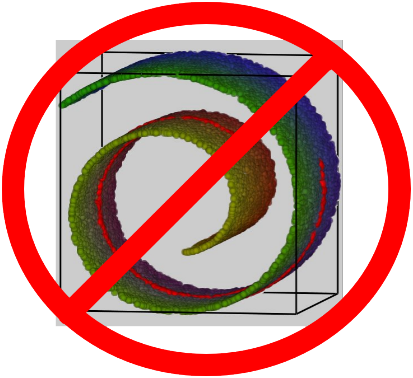
$$k(x, x') = \exp(-\left\|x - x\prime\right\|_1)$$

- Inverse covariance will be sparse and based on neighborhood.
- In the GP-LVM the neighborhood is learnt by optimization of **X**.

Outline

Campaign for Real Data

Say NO to the Swiss Roll



Simple Experiments

- Consider two real data sets.
- We apply each of the spectral methods we have reviewed.
- Apply the MEU framework.
- Follow the suggestion of Harmeling (Harmeling, 2007) and use the GPLVM likelihood (Lawrence, 2005) for embedding quality.
- The higher the likelihood the better the embedding.

Simple Experiments

- Consider two real data sets.
- We apply each of the spectral methods we have reviewed.
- Apply the MEU framework.
- Follow the suggestion of Harmeling (Harmeling, 2007) and use the GPLVM likelihood (Lawrence, 2005) for embedding quality.
- The higher the likelihood the better the embedding.

Simple Experiments

- Consider two real data sets.
- We apply each of the spectral methods we have reviewed.
- Apply the MEU framework.
- Follow the suggestion of Harmeling (Harmeling, 2007) and use the GPLVM likelihood (Lawrence, 2005) for embedding quality.
- The higher the likelihood the better the embedding.

Simple Experiments

- Consider two real data sets.
- We apply each of the spectral methods we have reviewed.
- Apply the MEU framework.
- Follow the suggestion of Harmeling (Harmeling, 2007) and use the GPLVM likelihood (Lawrence, 2005) for embedding quality.
- The higher the likelihood the better the embedding.

Simple Experiments

- Consider two real data sets.
- We apply each of the spectral methods we have reviewed.
- Apply the MEU framework.
- Follow the suggestion of Harmeling (Harmeling, 2007) and use the GPLVM likelihood (Lawrence, 2005) for embedding quality.
- The higher the likelihood the better the embedding.

Motion Capture Data

- Data consists of a 3-dimensional point cloud of the location of 34 points from a subject performing a run.
- 102 dimensional data set containing 55 frames of motion capture.
- Subject begins the motion from stationary and takes approximately three strides of run.
- Should see this structure in the visualization: a starting position followed by a series of loops.
- Data was made available by Ohio State University.
- The two dominant eigenvectors are visualized in following figures.

Motion Capture Data

- Data consists of a 3-dimensional point cloud of the location of 34 points from a subject performing a run.
- 102 dimensional data set containing 55 frames of motion capture.
- Subject begins the motion from stationary and takes approximately three strides of run.
- Should see this structure in the visualization: a starting position followed by a series of loops.
- Data was made available by Ohio State University.
- The two dominant eigenvectors are visualized in following figures.

Motion Capture Data

- Data consists of a 3-dimensional point cloud of the location of 34 points from a subject performing a run.
- 102 dimensional data set containing 55 frames of motion capture.
- Subject begins the motion from stationary and takes approximately three strides of run.
- Should see this structure in the visualization: a starting position followed by a series of loops.
- Data was made available by Ohio State University.
- The two dominant eigenvectors are visualized in following figures.

Motion Capture Data

- Data consists of a 3-dimensional point cloud of the location of 34 points from a subject performing a run.
- 102 dimensional data set containing 55 frames of motion capture.
- Subject begins the motion from stationary and takes approximately three strides of run.
- Should see this structure in the visualization: a starting position followed by a series of loops.
- Data was made available by Ohio State University.
- The two dominant eigenvectors are visualized in following figures.

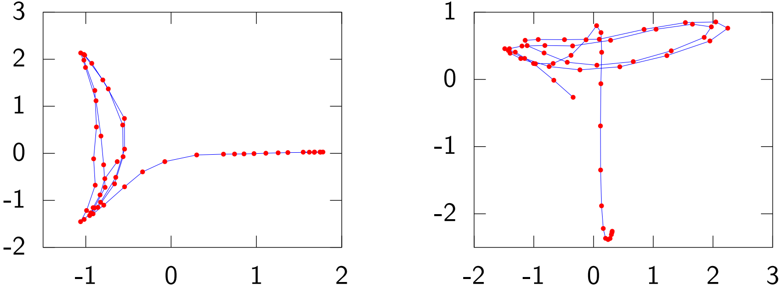
Motion Capture Data

- Data consists of a 3-dimensional point cloud of the location of 34 points from a subject performing a run.
- 102 dimensional data set containing 55 frames of motion capture.
- Subject begins the motion from stationary and takes approximately three strides of run.
- Should see this structure in the visualization: a starting position followed by a series of loops.
- Data was made available by Ohio State University.
- The two dominant eigenvectors are visualized in following figures.

Motion Capture Data

- Data consists of a 3-dimensional point cloud of the location of 34 points from a subject performing a run.
- 102 dimensional data set containing 55 frames of motion capture.
- Subject begins the motion from stationary and takes approximately three strides of run.
- Should see this structure in the visualization: a starting position followed by a series of loops.
- Data was made available by Ohio State University.
- The two dominant eigenvectors are visualized in following figures.

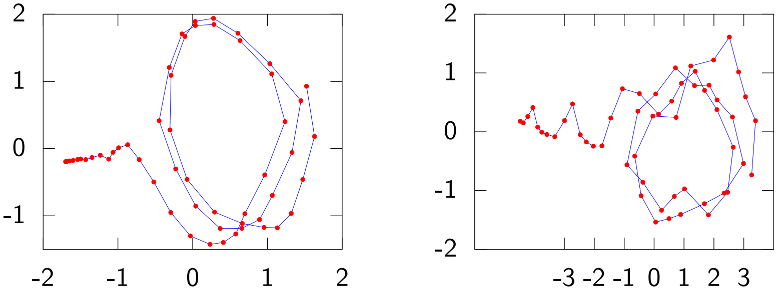
Laplacian Eigenmaps and LLE



(a) Laplacian Eigenmaps (b) Locally Linear Embedding

Figure: Models capture either the cyclic structure or the structure associated with the start of the run or both parts.

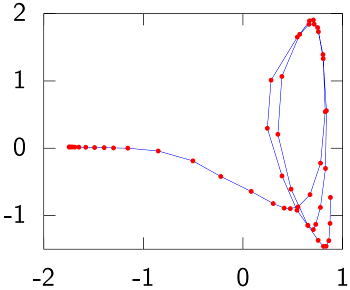
Isomap and MVU



(a) Isomap (b) MVU

Figure: Models capture either the cyclic structure or the structure associated with the start of the run or both parts.

MEU



(a) MEU

Figure: Models capture either the cyclic structure or the structure associated with the start of the run or both parts.

Motion Capture: Model Scores

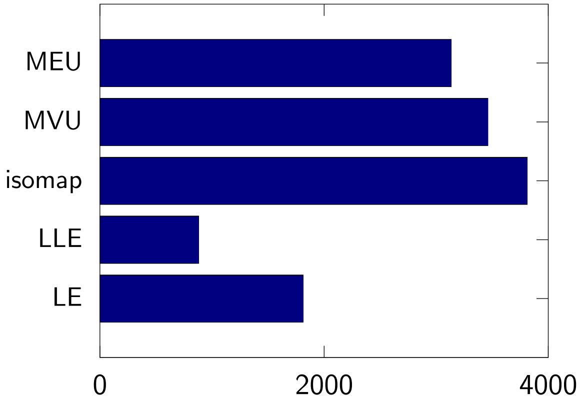


Figure: Model score for the different spectral approaches.

Robot Navigation Example

- Second data set: series of recordings from a robot as it traces a square path in a building.
- It records the strength of WiFi signals (see Ferris et al., 2007, for an application).
- Robot only in two dimensions, the inherent dimensionality of the data should be two.
- Robot completes a single circuit after entry: it is expected to exhibit "loop closure".
- Data consists of 215 frames of measurement of WiFi signal strength of 30 access points.

Robot Navigation Example

- Second data set: series of recordings from a robot as it traces a square path in a building.
- It records the strength of WiFi signals (see Ferris et al., 2007, for an application).
- Robot only in two dimensions, the inherent dimensionality of the data should be two.
- Robot completes a single circuit after entry: it is expected to exhibit "loop closure".
- Data consists of 215 frames of measurement of WiFi signal strength of 30 access points.

Robot Navigation Example

- Second data set: series of recordings from a robot as it traces a square path in a building.
- It records the strength of WiFi signals (see Ferris et al., 2007, for an application).
- Robot only in two dimensions, the inherent dimensionality of the data should be two.
- Robot completes a single circuit after entry: it is expected to exhibit "loop closure".
- Data consists of 215 frames of measurement of WiFi signal strength of 30 access points.

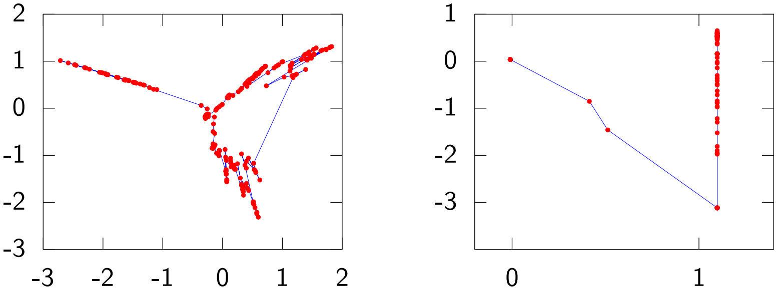
Robot Navigation Example

- Second data set: series of recordings from a robot as it traces a square path in a building.
- It records the strength of WiFi signals (see Ferris et al., 2007, for an application).
- Robot only in two dimensions, the inherent dimensionality of the data should be two.
- Robot completes a single circuit after entry: it is expected to exhibit "loop closure".
- Data consists of 215 frames of measurement of WiFi signal strength of 30 access points.

Robot Navigation Example

- Second data set: series of recordings from a robot as it traces a square path in a building.
- It records the strength of WiFi signals (see Ferris et al., 2007, for an application).
- Robot only in two dimensions, the inherent dimensionality of the data should be two.
- Robot completes a single circuit after entry: it is expected to exhibit “loop closure”.
- Data consists of 215 frames of measurement of WiFi signal strength of 30 access points.

Laplacian Eigenmaps and LLE



(a) Laplacian Eigenmaps (b) Locally Linear Embedding

Figure: Models show loop closure but smooth the trace to different degrees.

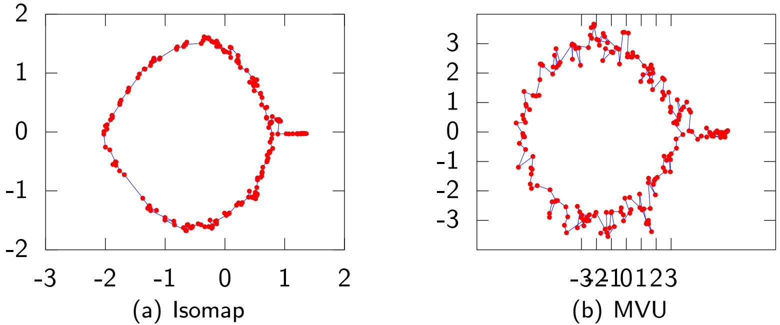
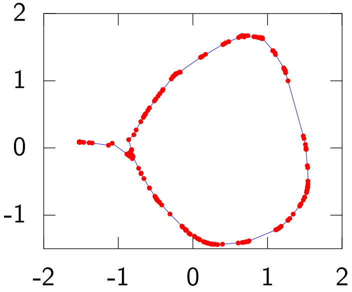


Figure: Models show loop closure but smooth the trace to different degrees.

MEU



(a) MEU

Figure: Models show loop closure but smooth the trace to different degrees.

Robot Navigation: Model Scores

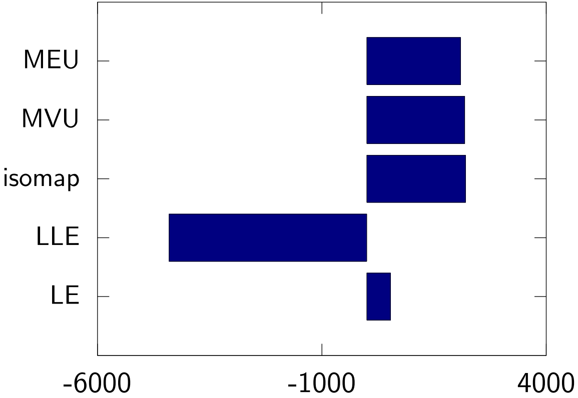


Figure: Model score for the different spectral approaches.

Outline

Discussion

- New perspective on dimensionality reduction algorithms based around maximum entropy.
- Start with MVU and end with GRFs.
- Hope that this perspective on dimensionality reduction will encourage new strands of research at the interface of these areas.

Discussion

- New perspective on dimensionality reduction algorithms based around maximum entropy.
- Start with MVU and end with GRFs.
- Hope that this perspective on dimensionality reduction will encourage new strands of research at the interface of these areas.

Discussion

- New perspective on dimensionality reduction algorithms based around maximum entropy.
- Start with MVU and end with GRFs.
- Hope that this perspective on dimensionality reduction will encourage new strands of research at the interface of these areas.

Stages of Spectral Dimensionality Reduction

- Our perspective shows there are three separate stages used in existing spectral dimensionality algorithms.
  1. A neighborhood between data points is selected. Normally k-nearest neighbors or similar algorithms are used.
  2. Interpoint distances between neighbors are fed to the algorithms which provide a similarity matrix. The way the entries in the similarity matrix are computed is the main difference between the different algorithms.
  3. The relationship between points in the similarity matrix is visualized using the eigenvectors of the similarity matrix.

Stages of Spectral Dimensionality Reduction

- Our perspective shows there are three separate stages used in existing spectral dimensionality algorithms.
  1. A neighborhood between data points is selected. Normally k-nearest neighbors or similar algorithms are used.
  2. Interpoint distances between neighbors are fed to the algorithms which provide a similarity matrix. The way the entries in the similarity matrix are computed is the main difference between the different algorithms.
  3. The relationship between points in the similarity matrix is visualized using the eigenvectors of the similarity matrix.

Stages of Spectral Dimensionality Reduction

- Our perspective shows there are three separate stages used in existing spectral dimensionality algorithms.
  1. A neighborhood between data points is selected. Normally k-nearest neighbors or similar algorithms are used.
  2. Interpoint distances between neighbors are fed to the algorithms which provide a similarity matrix. The way the entries in the similarity matrix are computed is the main difference between the different algorithms.
  3. The relationship between points in the similarity matrix is visualized using the eigenvectors of the similarity matrix.

Stages of Spectral Dimensionality Reduction

- Our perspective shows there are three separate stages used in existing spectral dimensionality algorithms.
  1. A neighborhood between data points is selected. Normally k-nearest neighbors or similar algorithms are used.
  2. Interpoint distances between neighbors are fed to the algorithms which provide a similarity matrix. The way the entries in the similarity matrix are computed is the main difference between the different algorithms.
  3. The relationship between points in the similarity matrix is visualized using the eigenvectors of the similarity matrix.

Stages of Spectral Dimensionality Reduction

- Our perspective shows there are three separate stages used in existing spectral dimensionality algorithms.
  1. A neighborhood between data points is selected. Normally k-nearest neighbors or similar algorithms are used.
  2. Interpoint distances between neighbors are fed to the algorithms which provide a similarity matrix. The way the entries in the similarity matrix are computed is the main difference between the different algorithms.
  3. The relationship between points in the similarity matrix is visualized using the eigenvectors of the similarity matrix.

Our Perspective

- Each step is somewhat orthogonal.
- Neighborhood relations need not come from nearest neighbors: can use structure learning.
- Main difference between approaches is how similarity matrix entries are determined.
- Final step attempts to visualize the similarity using eigenvectors. This is just one possible approach.
- There is an entire field of graph visualization proposing different approaches to visualizing such graphs.

Our Perspective

- Each step is somewhat orthogonal.
- Neighborhood relations need not come from nearest neighbors: can use structure learning.
- Main difference between approaches is how similarity matrix entries are determined.
- Final step attempts to visualize the similarity using eigenvectors. This is just one possible approach.
- There is an entire field of graph visualization proposing different approaches to visualizing such graphs.

Our Perspective

- Each step is somewhat orthogonal.
- Neighborhood relations need not come from nearest neighbors: can use structure learning.
- Main difference between approaches is how similarity matrix entries are determined.
- Final step attempts to visualize the similarity using eigenvectors. This is just one possible approach.
- There is an entire field of graph visualization proposing different approaches to visualizing such graphs.

Our Perspective

- Each step is somewhat orthogonal.
- Neighborhood relations need not come from nearest neighbors: can use structure learning.
- Main difference between approaches is how similarity matrix entries are determined.
- Final step attempts to visualize the similarity using eigenvectors. This is just one possible approach.
- There is an entire field of graph visualization proposing different approaches to visualizing such graphs.

Our Perspective

- Each step is somewhat orthogonal.
- Neighborhood relations need not come from nearest neighbors: can use structure learning.
- Main difference between approaches is how similarity matrix entries are determined.
- Final step attempts to visualize the similarity using eigenvectors. This is just one possible approach.
- There is an entire field of graph visualization proposing different approaches to visualizing such graphs.

Advantages of Existing Approaches

- Conflating the three steps allows faster complete algorithms.
- E.g. mixing 2nd & 3rd allows speed ups by never computing the similarity matrix.
- We still can understand the algorithm from the unifying perspective while exploiting the computational advantages offered by this neat shortcut.

Advantages of Existing Approaches

- Conflating the three steps allows faster complete algorithms.
- E.g. mixing 2nd & 3rd allows speed ups by never computing the similarity matrix.
- We still can understand the algorithm from the unifying perspective while exploiting the computational advantages offered by this neat shortcut.

Advantages of Existing Approaches

- Conflating the three steps allows faster complete algorithms.
- E.g. mixing 2nd & 3rd allows speed ups by never computing the similarity matrix.
- We still can understand the algorithm from the unifying perspective while exploiting the computational advantages offered by this neat shortcut.

Acknowledgements

Conversations with John Kent, Chris Williams, Brenden Lake, Joshua Tenenbaum and John Lafferty have influenced the thinking in this work.

References I

M. Belkin and P. Niyogi. Laplacian eigenmaps for dimensionality reduction and data representation. *Neural Computation*, 15(6):1373–1396, 2003. [DOI].

B. D. Ferris, D. Fox, and N. D. Lawrence. WiFi-SLAM using Gaussian process latent variable models. In M. M. Veloso, editor, *Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI 2007)*, pages 2480–2485, 2007. [PDF].

R. Greiner and D. Schuurmans, editors. *Proceedings of the International Conference in Machine Learning*, volume 21, 2004. Omnipress.

J. Ham, D. D. Lee, S. Mika, and B. Schölkopf. A kernel view of dimensionality reduction of manifolds. In Greiner and Schuurmans (2004). [PDF].

S. Harmeling. Exploring model selection techniques for nonlinear dimensionality reduction. Technical Report EDI-INF-RR-0960, University of Edinburgh,

E. T. Jaynes. Bayesian methods: General background. In J. H. Justice, editor, *Maximum Entropy and Bayesian Methods in Applied Statistics*, pages 1–25. Cambridge University Press, 1986.

D. Koller and N. Friedman. *Probabilistic Graphical Models: Principles and Techniques*. MIT Press, 2009. [Google Books] .

N. D. Lawrence. Probabilistic non-linear principal component analysis with Gaussian process latent variable models. *Journal of Machine Learning Research*, 6:1783–1816, 11 2005.

K. V. Mardia, J. T. Kent, and J. M. Bibby. *Multivariate analysis*. Academic Press, London, 1979. [Google Books] .

S. T. Roweis and L. K. Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290(5500): 2323–2326, 2000. [DOI].

B. Schölkopf, A. Smola, and K.-R. Müller. Nonlinear component analysis as a kernel eigenvalue problem. *Neural Computation*, 10:1299–1319, 1998. [DOI].

B. Schölkopf and A. J. Smola. *Learning with Kernels*. MIT Press, Cambridge, MA, 2001.

J. B. Tenenbaum, V. de Silva, and J. C. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290(5500):2319–2323, 2000. [DOI].

K. Q. Weinberger, F. Sha, and L. K. Saul. Learning a kernel matrix for nonlinear dimensionality reduction. In Greiner and Schuurmans (2004), pages 839–846.

C. K. I. Williams. On a connection between kernel PCA and metric multidimensional scaling. In T. K. Leen, T. G. Dietterich, and V. Tresp, editors, *Advances in Neural Information Processing Systems*, volume 13, pages 675–681, Cambridge, MA, 2001. MIT Press.

Outline

Learning the Neighborhood

Final Experiment: Structure Learning

- Test the ability of L1 regularization of the random field to learn the neighborhood.
- Considered the motion capture data and used the DRILL with a neighborhood size of 20 and full connectivity.
- L1 regularization on the parameters: vary regularization size and seek a maximum under the GPLVM.

Structure Learning from Neighborhood of 20

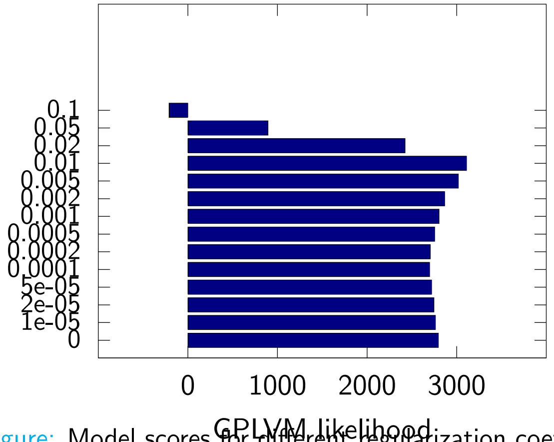


Figure: Model scores for different regularization coefficients.

Structure Learning from Neighborhood of 20

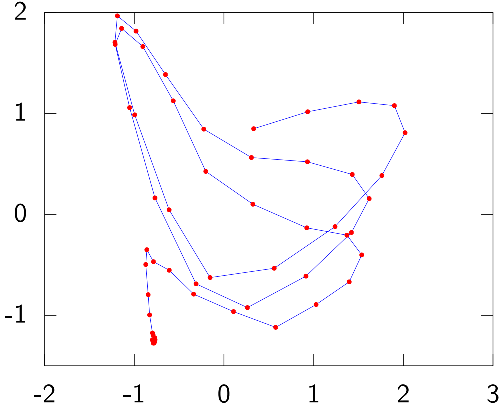


Figure: Visualization associated with highest model score.

Full Structure Learning

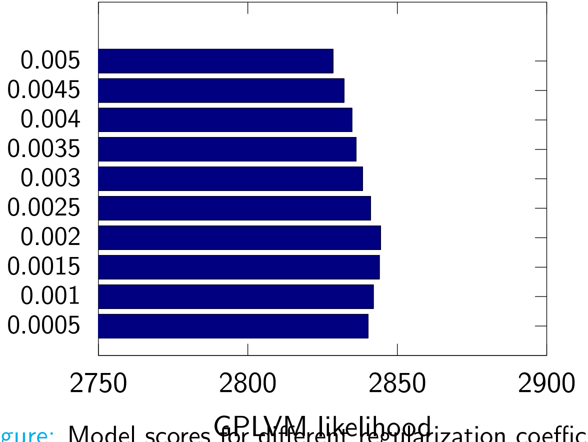


Figure: Model scores for different regularization coefficients.

Full Structure Learning

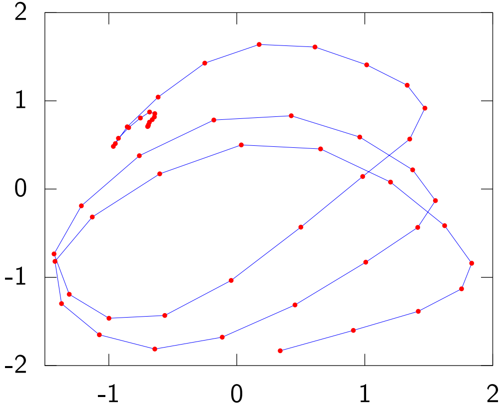


Figure: Visualization associated with highest model score.

Different Neighborhood Scores

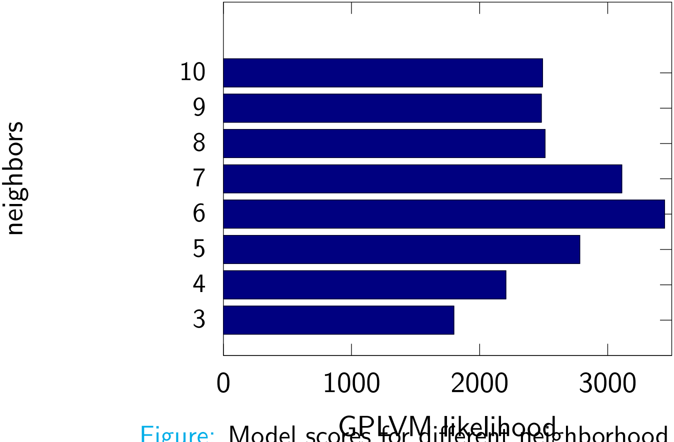


Figure: Model scores for different neighborhood sizes.

Different Neighborhood Scores

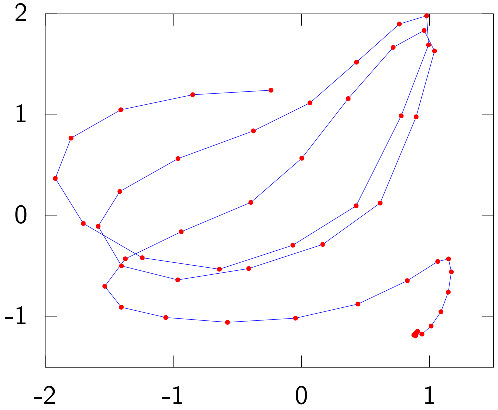


Figure: Visualization associated with highest model score.

Structure Learning from Neighborhood of 6

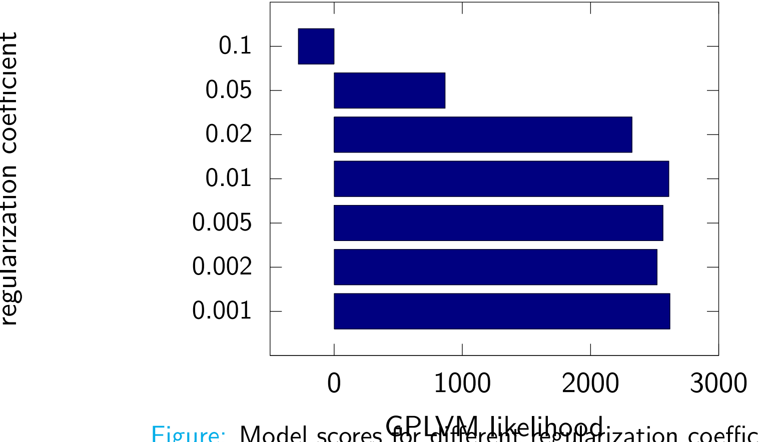


Figure: Model scores for different regularization coefficients.

Structure Learning from Neighborhood of 6

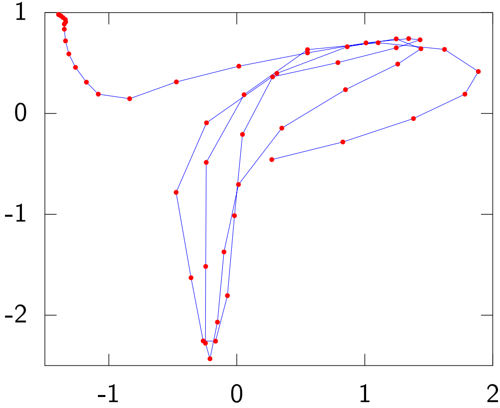


Figure: Visualization associated with highest model score.