Probabilistic Dimensionality Reduction II

Neil D. Lawrence
University of Sheffield

Imperial College
21st October 2015

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	$n \times n$
$\mathbf{H}$	centering matrix	$n \times n$
$\mathbf{B}$	centred similarity/kernel/covariance matrix	$n \times n$
$\mathbf{L}$	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

Mixtures of Gaussians

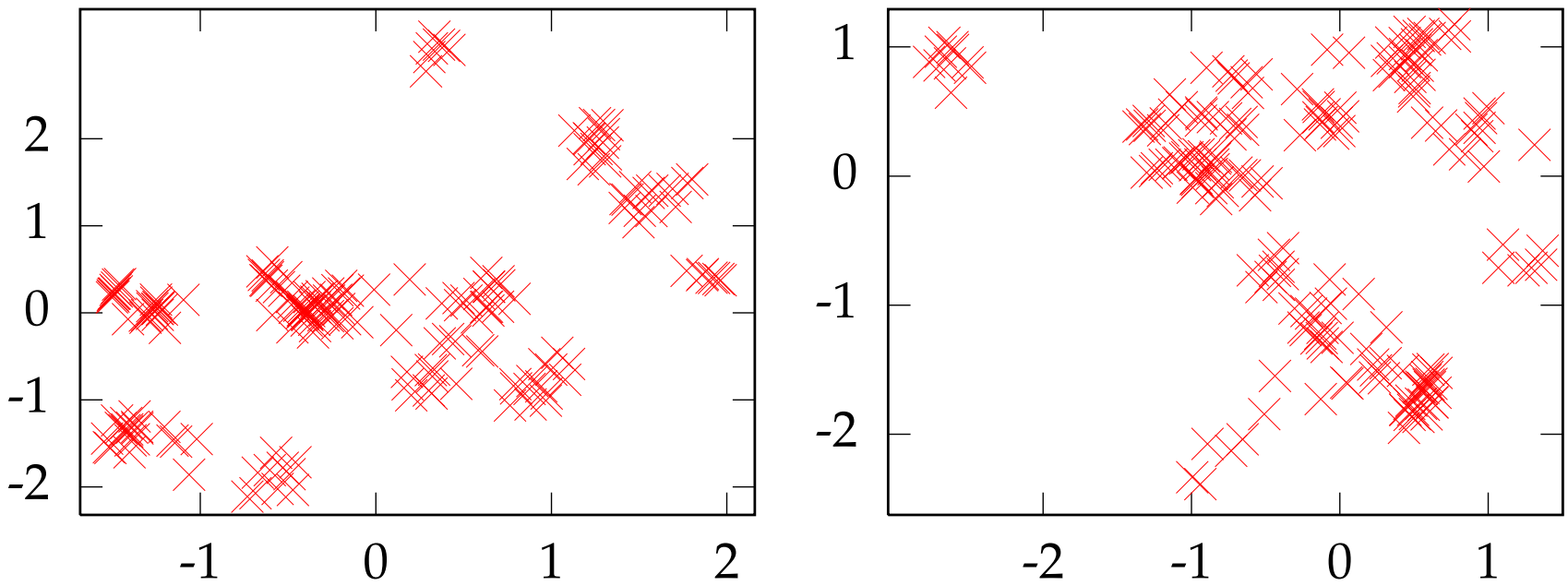


Figure: Two dimensional data sets.

Mixtures of Gaussians

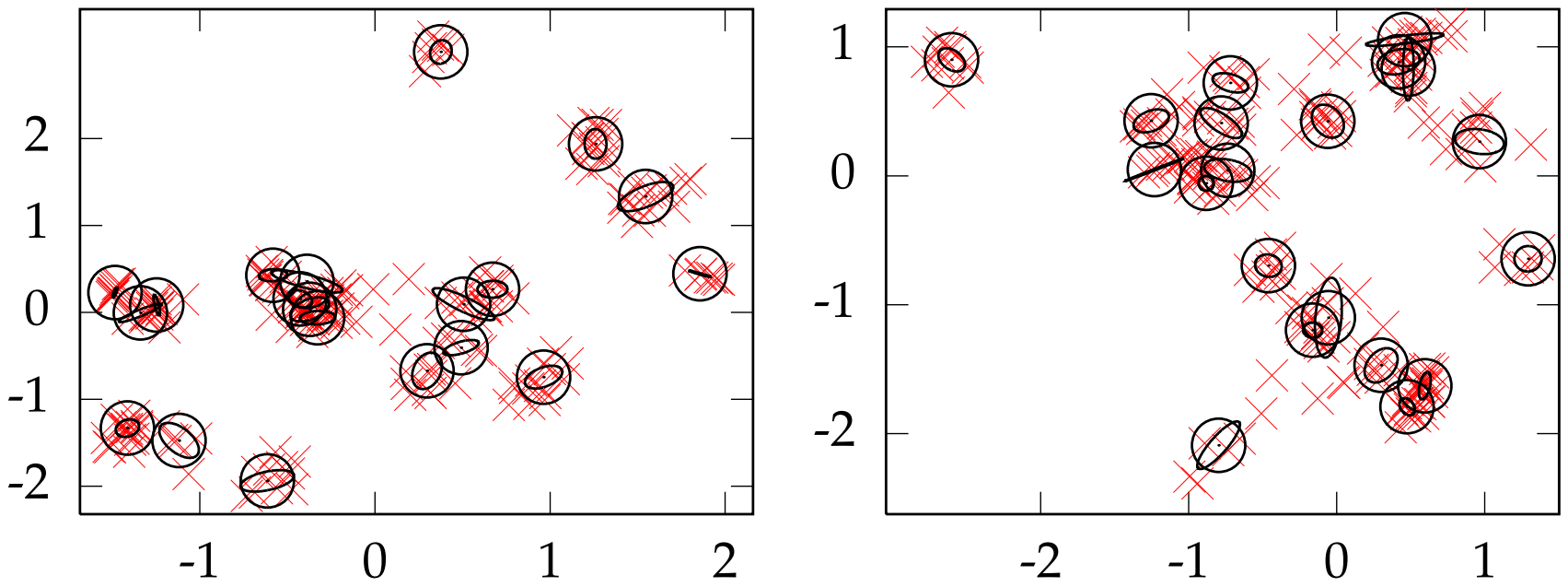


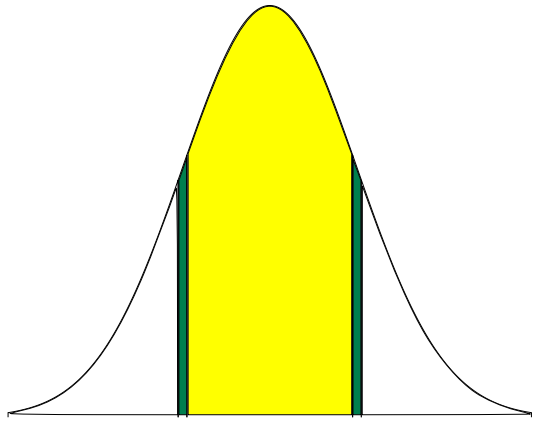
Figure: Complex structure not a problem for mixtures of Gaussians.

Thinking in High Dimensions

- Two dimensional plots of Gaussians can be misleading.
- Our low dimensional intuitions can fail dramatically.
- Two major issues:
  1. In high dimensions all the data moves to a 'shell'. There is nothing near the mean!
  2. Distances between points become constant.
  3. These affects apply to many densities.
- Let's consider a Gaussian "egg".

The Gaussian Egg

- See also Exercise 1.4 in (**?**)

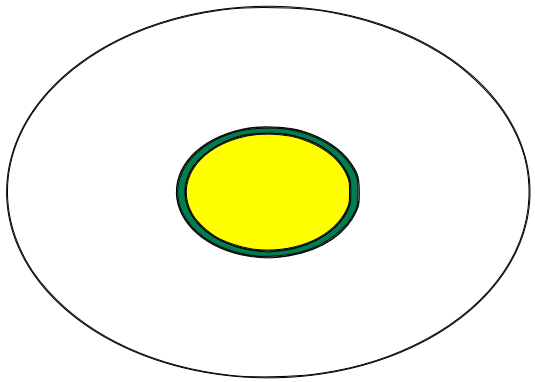


Volumes: 65.8%, 4.8% 29.4%

Figure: One dimensional Gaussian density.

The Gaussian Egg

- See also Exercise 1.4 in (**?**)

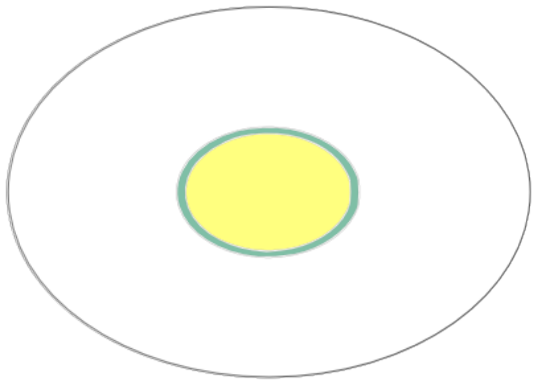


Volumes: 59.4%, 7.4% 33.2%

Figure: Two dimensional Gaussian density.

The Gaussian Egg

- See also Exercise 1.4 in (**?**)



Volumes: 56.1%, 9.2%, 34.7%

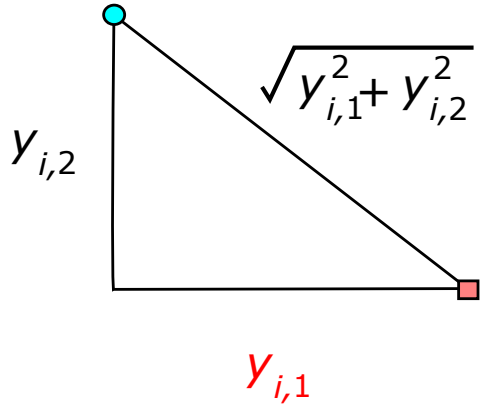
Figure: Three dimensional Gaussian density.

Mathematics

What is the density of probability mass?

$$y_{i,k} \sim \mathcal{N}\left(0, \sigma^2\right)$$

$$\implies y_{i,k}^2 \sim \sigma^2 \chi_1^2$$



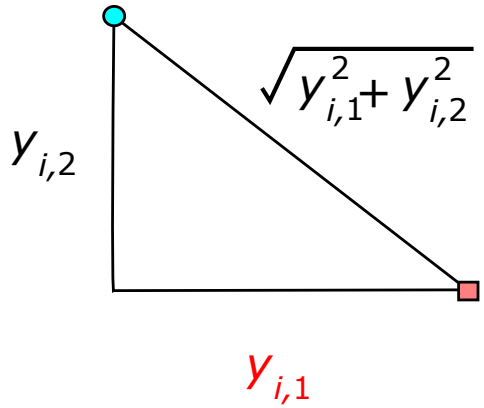
Square of sample from Gaussian is scaled chi-squared density

Mathematics

What is the density of probability mass?

$$y_{i,k} \sim \mathcal{N}\left(0, \sigma^2\right)$$

$$\implies y_{i,k}^2 \sim \mathcal{G}\left(\frac{1}{2}, \frac{1}{2\sigma^2}\right)$$



Chi squared density is a variant of the gamma density with shape parameter $a = \frac{1}{2}$, rate parameter $b = \frac{1}{2\sigma^2}$,

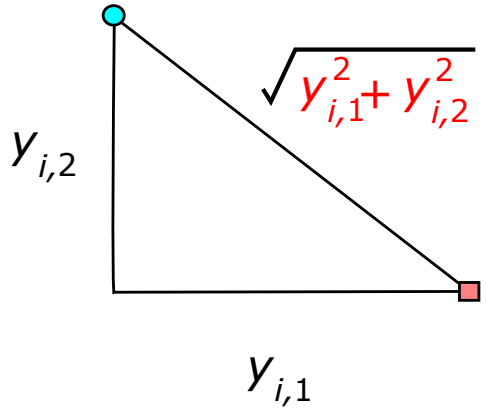
$$\mathcal{G}(x|a,b) = \frac{b^a}{\Gamma(a)} x^{a-1} e^{-bx}.$$

Mathematics

What is the density of probability mass?

$$y_{i,k} \sim \mathcal{N}\left(0, \sigma^2\right)$$

$$\Longrightarrow y_{i,1}^2 + y_{i,2}^2 \sim \mathcal{G}\left(\frac{2}{2}, \frac{1}{2\sigma^2}\right)$$



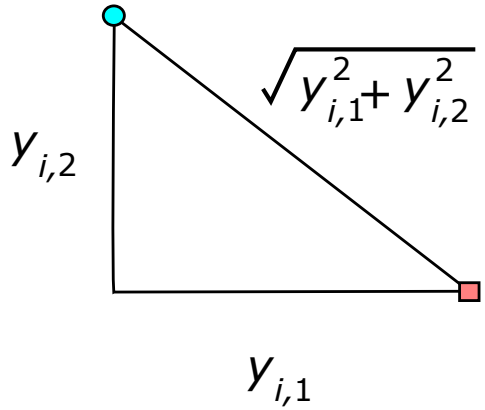
Addition of gamma random variables with the same rate is gamma with sum of shape parameters ($y_{i,k}$s are independent)

Mathematics

What is the density of probability mass?

$$\sum_{k=1}^{p} y_{i,k}^2 \sim \mathcal{G}\left(\frac{p}{2}, \frac{1}{2\sigma^2}\right)$$

$$\Longrightarrow \left\langle \sum_{k=1}^{p} y_{i,k}^2 \right\rangle = p\sigma^2$$



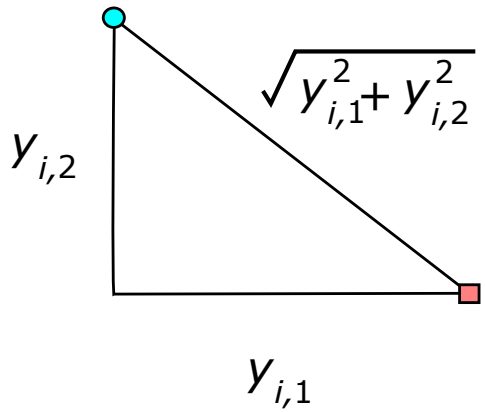
Addition of gamma random variables with the same rate is gamma with sum of shape parameters ($y_{i,k}$s are independent)

Mathematics

What is the density of probability mass?

$$\frac{1}{p}\sum_{k=1}^{p} y_{i,k}^2 \sim \mathcal{G}\left(\frac{p}{2}, \frac{p}{2\sigma^2}\right)$$

$$\Longrightarrow \left\langle \frac{1}{p}\sum_{k=1}^{p} y_{i,k}^2 \right\rangle = \sigma^2$$



Scaling of gamma density scales the rate parameter

Where is the Mass?

- Squared distances are gamma distributed.

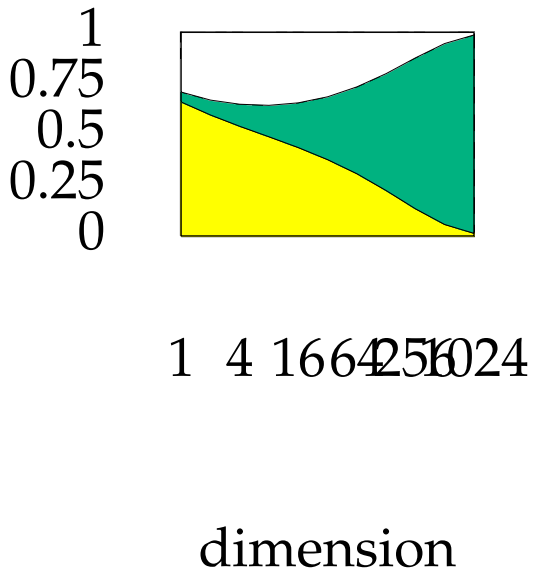
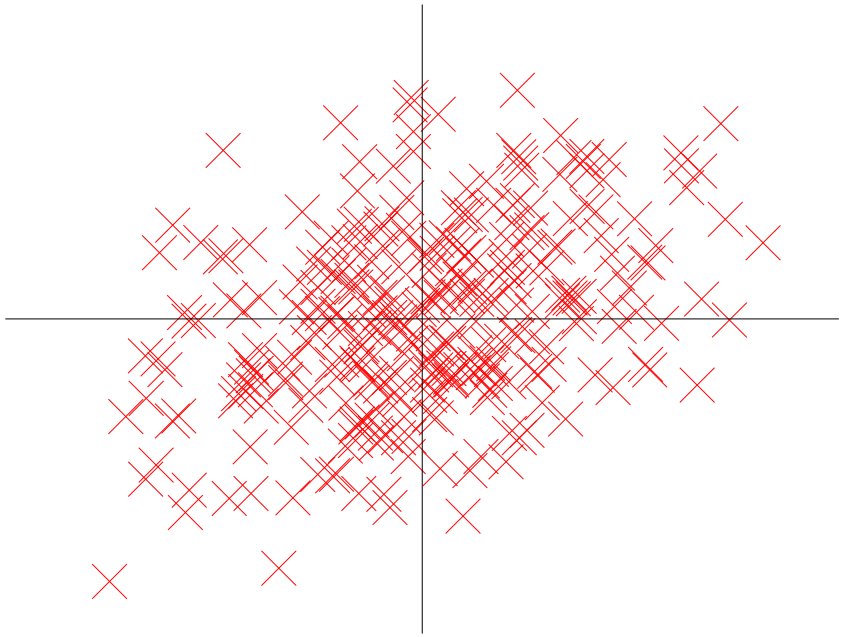


Figure: Plot of probability mass versus dimension. Plot shows the volume of density inside 0.95 of a standard deviation (yellow), between 0.95 and 1.05 standard deviations (green), over 1.05 and

Looking at Gaussian Samples



Interpoint Distances

- The other effect in high dimensions is all points become equidistant.
- Can show this for Gaussians with a similar proof to the above,

$$y_{i,k} \sim \mathcal{N}\left(0, \sigma_k^2\right) \qquad y_{j,k} \sim \mathcal{N}\left(0, \sigma_k^2\right)$$

$$y_{i,k} - y_{j,k} \sim \mathcal{N}\left(0, 2\sigma_k^2\right)$$

$$\left(y_{i,k} - y_{j,k}\right)^2 \sim \mathcal{G}\left(\frac{1}{2}, \frac{1}{4\sigma_k^2}\right)$$

For spherical Gaussian, $\sigma_k^2 = \sigma^2$

$$\sum_{k=1}^{p} \left(y_{i,k} - y_{j,k}\right)^2 \sim \mathcal{G}\left(\frac{p}{2}, \frac{1}{4\sigma^2}\right)$$

$$\frac{1}{p}\sum_{k=1}^{p} \left(y_{i,k} - y_{j,k}\right)^2 \sim \mathcal{G}\left(\frac{p}{2}, \frac{p}{4\sigma^2}\right)$$

Dimension normalized distance between points is drawn

Central Limit Theorem and Non-Gaussian Case

- We can compute the density of squared distance *analytically* for spherical, independent Gaussian data.

Central Limit Theorem and Non-Gaussian Case

- We can compute the density of squared distance *analytically* for spherical, independent Gaussian data.
- More generally, for *independent* data, the *central limit theorem* applies.

Central Limit Theorem and Non-Gaussian Case

- We can compute the density of squared distance *analytically* for spherical, independent Gaussian data.
- More generally, for *independent* data, the *central limit theorem* applies.
  - The mean squared distance in high dimensional space is the mean of the variances.

Central Limit Theorem and Non-Gaussian Case

- We can compute the density of squared distance *analytically* for spherical, independent Gaussian data.
- More generally, for *independent* data, the *central limit theorem* applies.
  - The mean squared distance in high dimensional space is the mean of the variances.
  - The variance about the mean scales as p^{-1}.

Summary

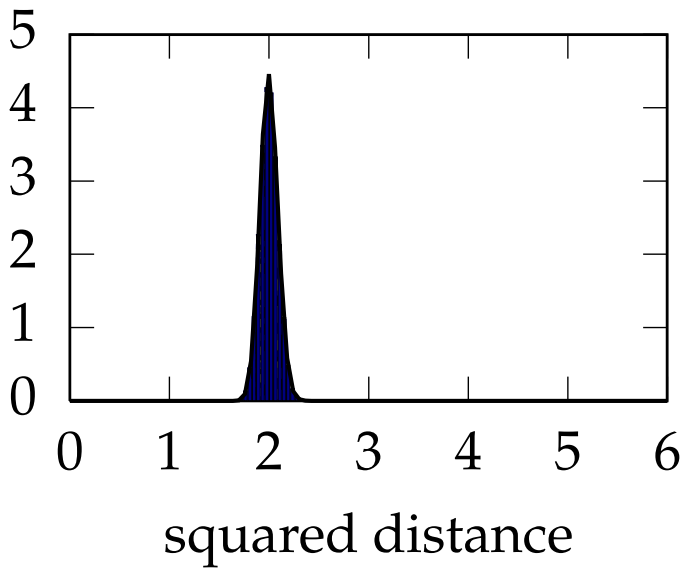
- In high dimensions if individual dimensions are *independent* the distributions behave counter intuitively.
- All data sits at one standard deviation from the mean.
- The densities of squared distances can be analytically calculated for the Gaussian case.
- For non-Gaussian *independent* systems we can invoke the central limit theorem.
- Next we will consider example data sets and see how their interpoint distances are distributed.

Sanity Check

Data sampled from independent Gaussian distribution

- If dimensions are independent, we expect low variance, Gaussian behavior for the distribution of squared distances.

Distance distribution for a Gaussian with $p = 1000$, $n = 1000$

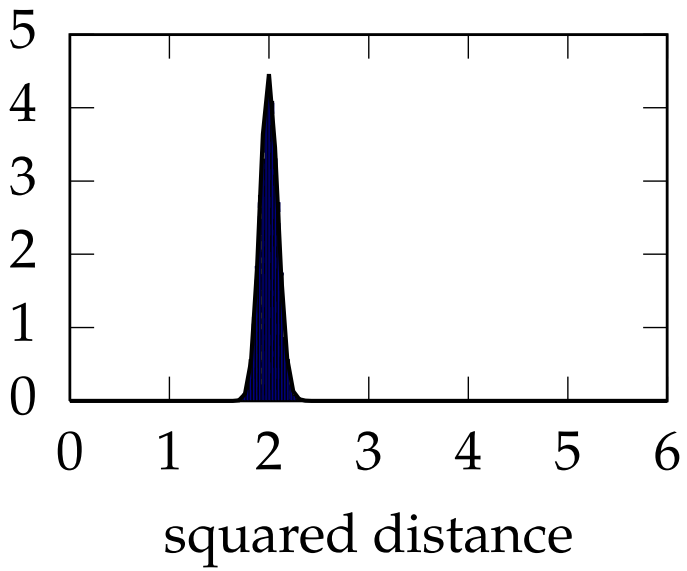


Sanity Check

Same data generation, but fewer data points.

- If dimensions are independent, we expect low variance, Gaussian behaviour for the distribution of squared distances.

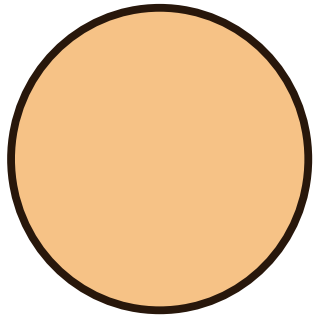
Distance distribution for a Gaussian with $p = 1000$, $n = 100$



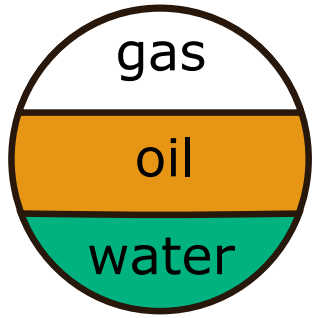
Oil Data

- Simulated measurements from an oil pipeline (Bishop and James, 1993).
- Pipleline contains oil, water and gas.
- Three phases of flow in pipeline—homogeneous, stratified and annular.

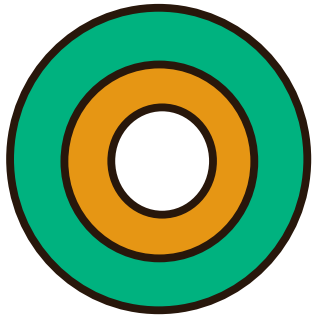
Homogeneous



Stratified



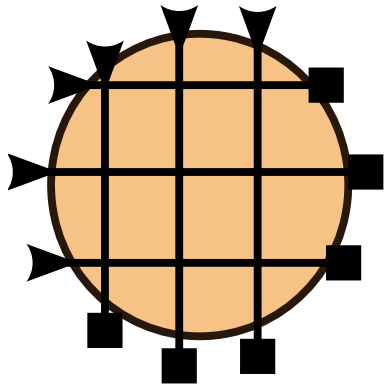
Annular



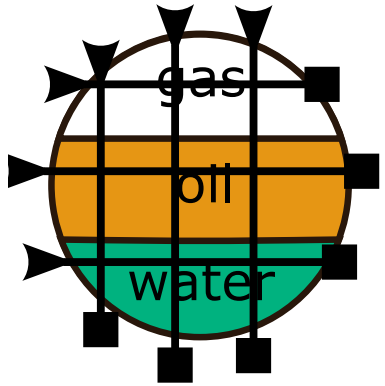
Oil Data

- Simulated measurements from an oil pipeline (Bishop and James, 1993).
- Pipleline contains oil, water and gas.
- Three phases of flow in pipeline—homogeneous, stratified and annular.
- Gamma densitometry sensors arranged in a configuration around pipeline.

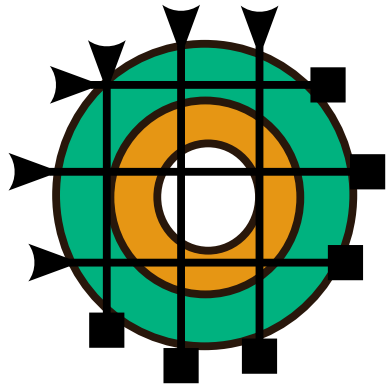
Homogeneous



Stratified



Annular



Oil Data

- 12 simulated measurements of oil flow in a pipe.
- Nature of flow is dependent on relative proportion of oil, water and gas.

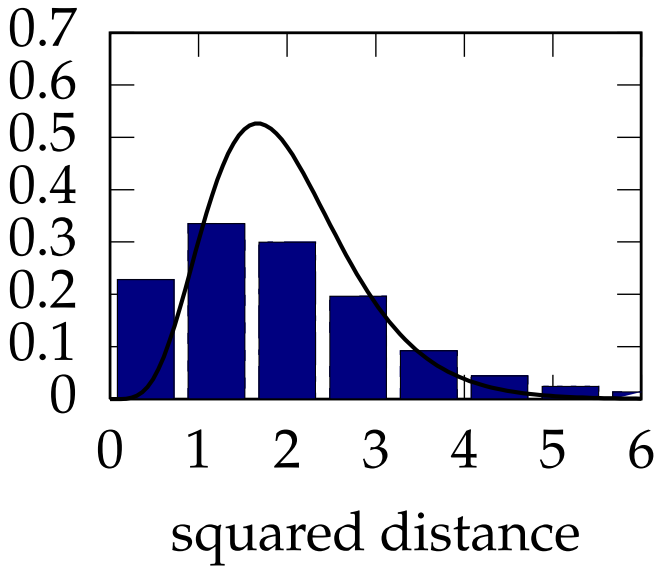
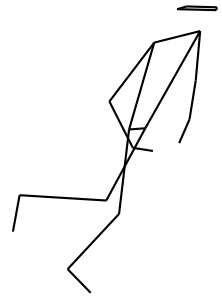


Figure: Interpoint squared distance distribution for oil data with $p = 12$ (variance of squared distances is 1.98 vs predicted 0.667).

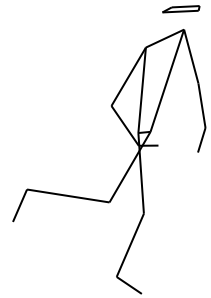
Stick Man Data

- $n = 55$ frames of motion capture.
- xyz locations of 34 points on the body.
- $p = 102$ dimensional data.
- "Run 1" available from `http://accad.osu.edu/research/mocap/mocap_data.htm`.

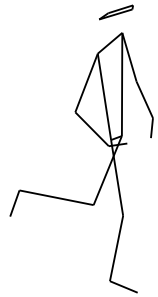
Changing



Angle



of Run



Stick Man

- Motion capture data inter point distance histogram.

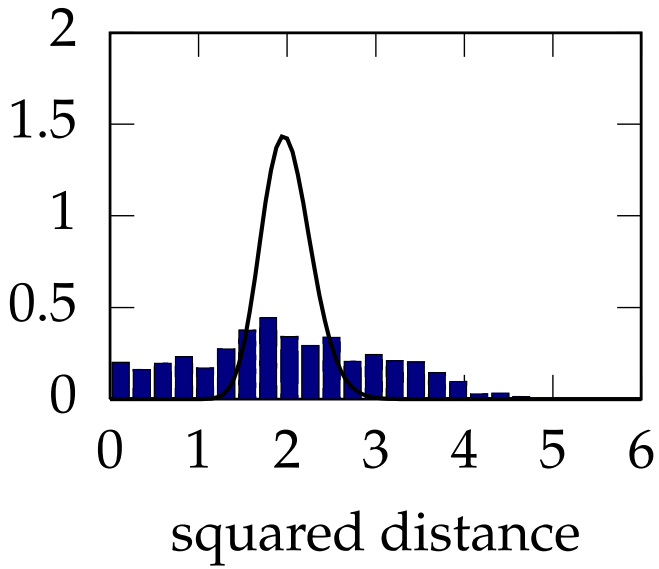
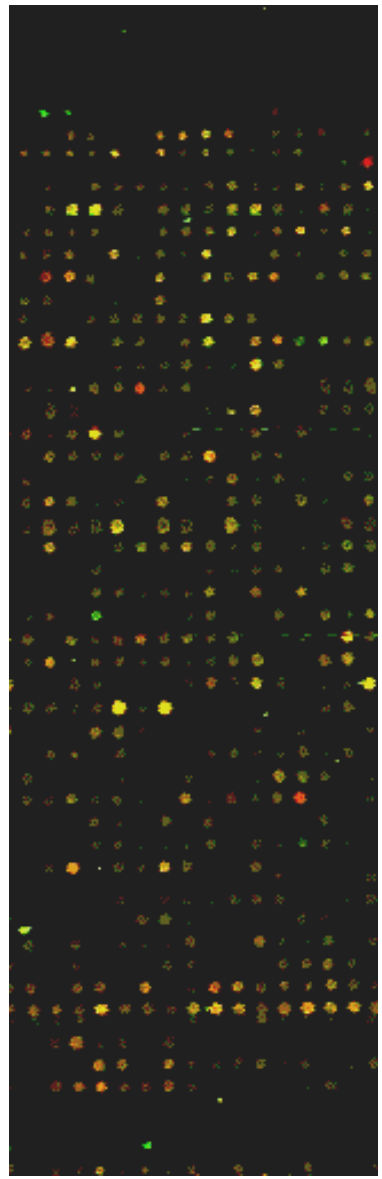


Figure: Interpoint squared distance distribution for stick man data with $p = 102$ (variance of squared distances is 1.09 vs predicted 0.0784).

Microarray Data

- Gene expression measurements reflecting the cell cycle in yeast (?).
- $p = 6,178$ Genes measured for $n = 77$ experiments
- Data available from `http://genome-www.stanford.edu/cellcycle/data/rawdata/individual.htm`.



Microarray Data

- Spellman yeast cell cycle.

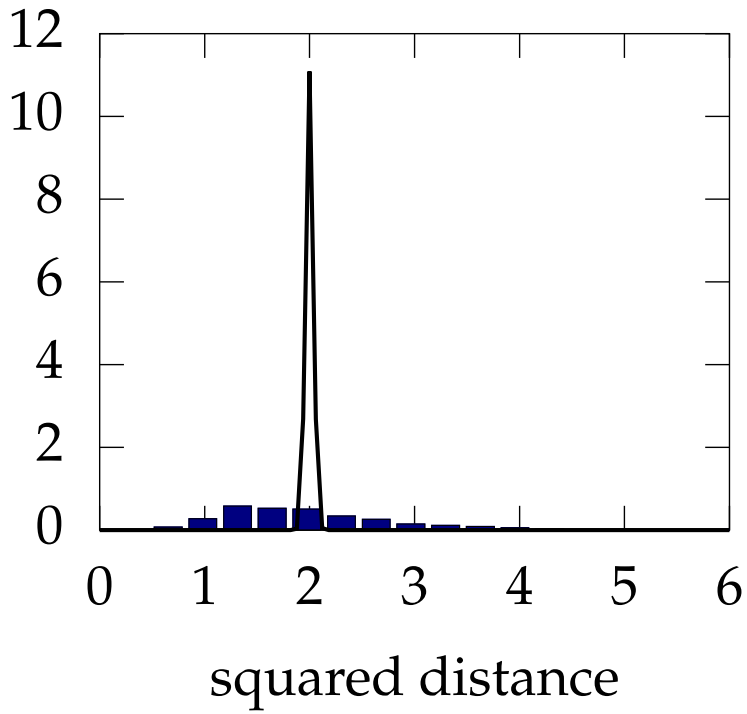


Figure: Interpoint squared distance distribution for Spellman microarray data with $p = 6178$ (variance of squared distances is 0.694 vs predicted 0.00129).

Grid Corpus Vowels

- Grid corpus data modeled for synthesis by Jon Barker.
- 33 context dependent vowel phones from 34 (mixed male/female) subjects.
- Means and variances of synthesis HMM for subjects (**?**).

34 Subjects'

Vowel

Phones

Grid Corpus Vowels

- Grid Corpus: `http://www.dcs.shef.ac.uk/spandh/gridcorpus/`.
- For each context dependent phone: 5 state HMM, one Gaussian component per state. 25 MFCC channels, with deltas and accelerations.

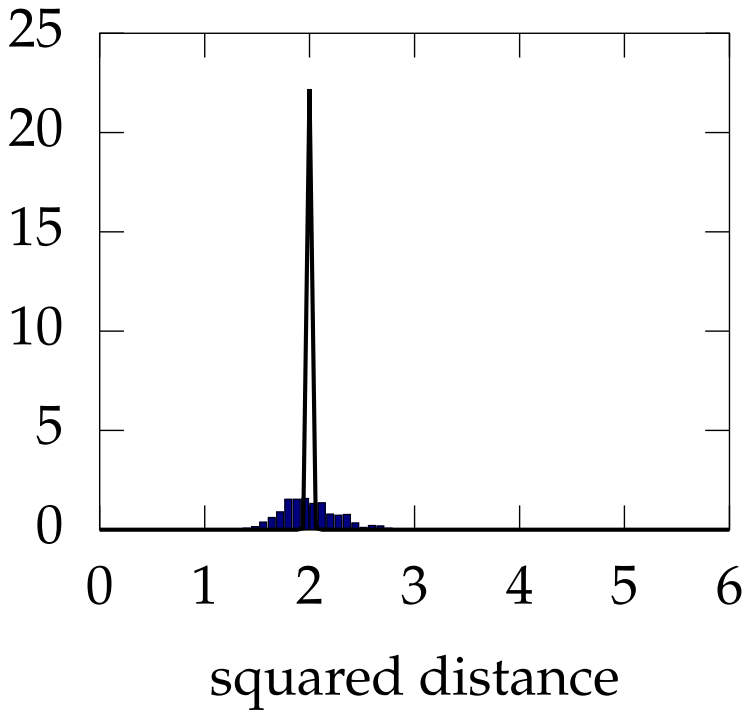


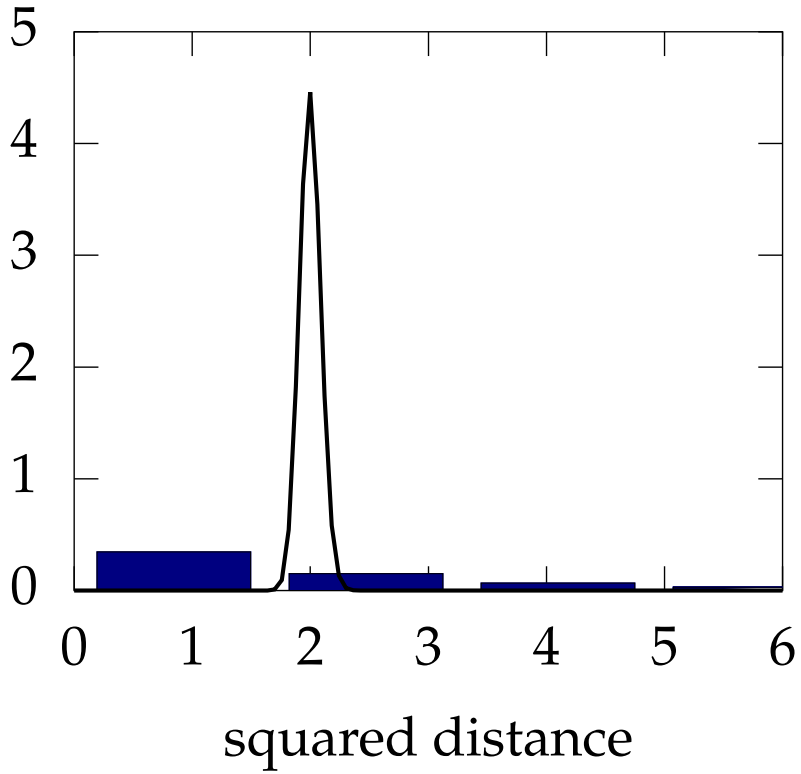
Figure: Interpoint squared distance distribution for Grid corpus vowel data with $n = 24750$ (variance of squared distances is 0.07 vs

Where does practice depart from our theory?

- The situation for real data does not reflect what we expect.
- Real data exhibits greater variances on interpoint distances.
  - Somehow the real data seems to have a smaller effective dimension.
- Let's look at another $p = 1000$.

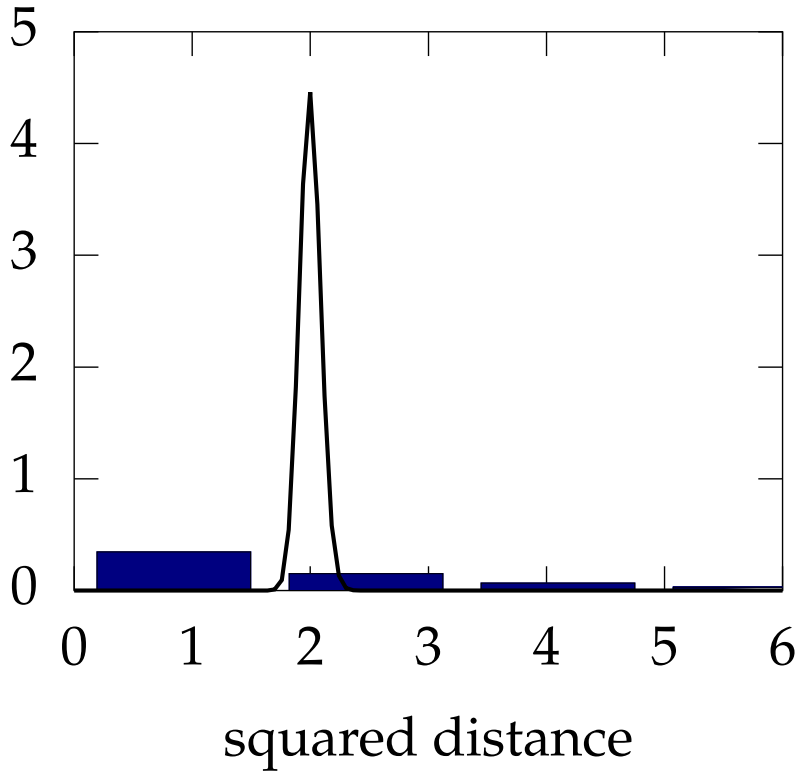
1000-D Gaussian

Distance distribution for a different Gaussian with $p = 1000$



1000-D Gaussian

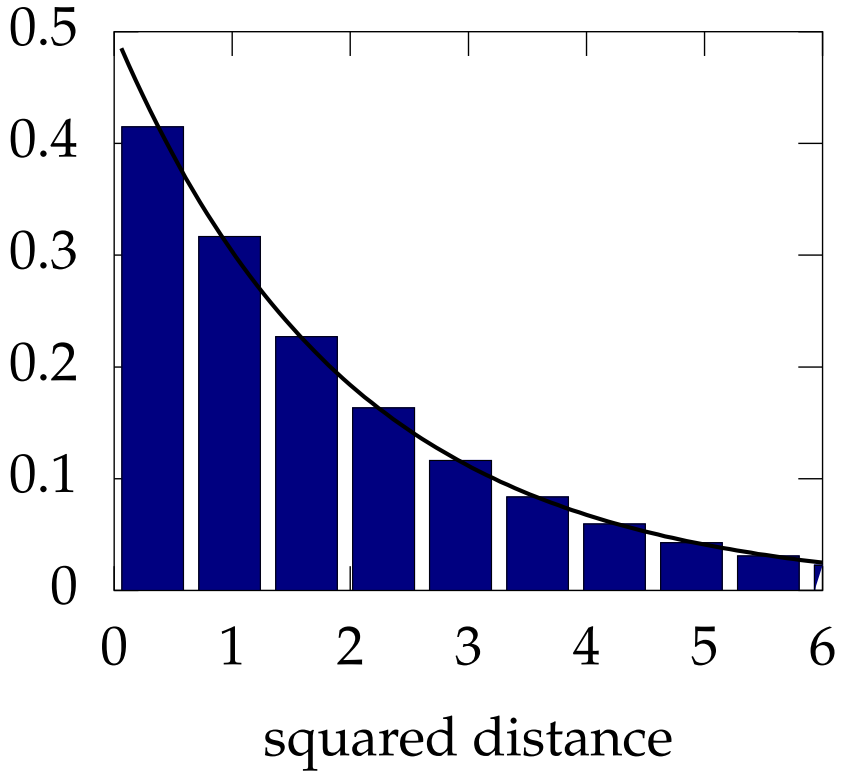
Distance distribution for a different Gaussian with $p = 1000$



1. Gaussian has a specific low rank covariance matrix $\mathbf{C} = \mathbf{W}\mathbf{W}^\top + \sigma^2\mathbf{I}$.
2. Take $\sigma^2 = 1e-2$ and sample $\mathbf{W} \in \Re^{1000\times 2}$ from $\mathcal{N}(0,1)$.

1000-D Gaussian

Distance distribution for a different Gaussian with $p = 1000$



1. Gaussian has a specific low rank covariance matrix $\mathbf{C} = \mathbf{W}\mathbf{W}^\top + \sigma^2\mathbf{I}$.
2. Take $\sigma^2 = 1e-2$ and sample $\mathbf{W} \in \Re^{1000\times 2}$ from $\mathcal{N}(0,1)$.
3. The probability taken assuming dimensionality of 2

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	$n \times n$
$\mathbf{H}$	centering matrix	$n \times n$
$\mathbf{B}$	centred similarity/kernel/covariance matrix	$n \times n$
$\mathbf{L}$	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

Motivation for Non-Linear Dimensionality Reduction

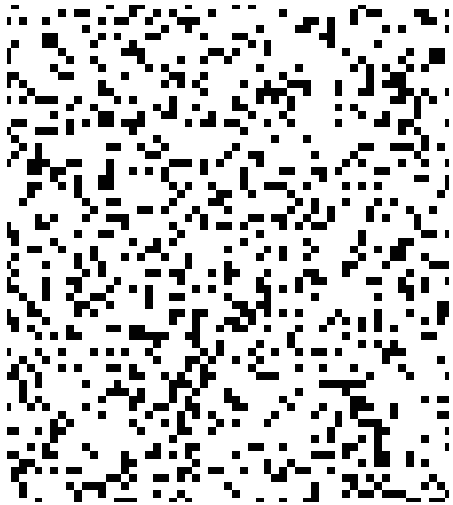
USPS Data Set Handwritten Digit

- 3648 Dimensions
  - 64 rows by 57 columns

Motivation for Non-Linear Dimensionality Reduction

USPS Data Set Handwritten Digit

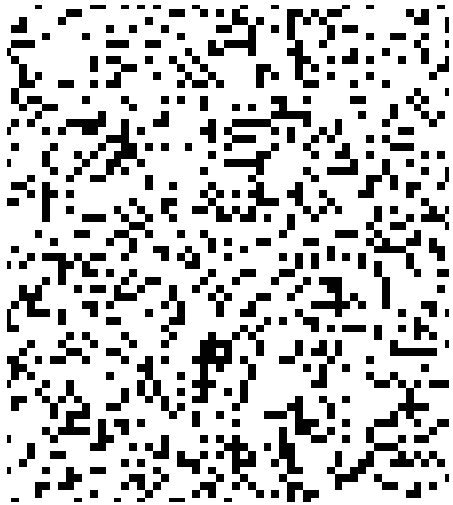
- 3648 Dimensions
  - 64 rows by 57 columns
  - Space contains more than just this digit.



Motivation for Non-Linear Dimensionality Reduction

USPS Data Set Handwritten Digit

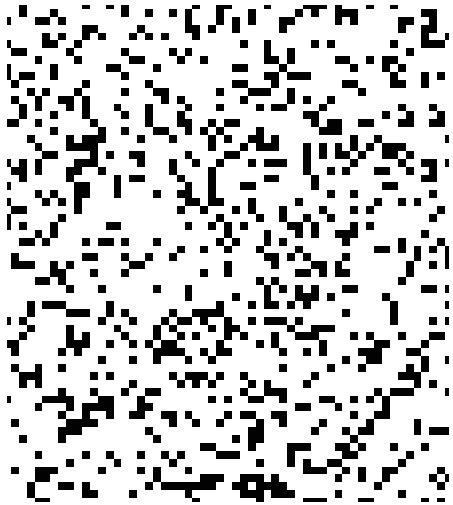
- 3648 Dimensions
  - 64 rows by 57 columns
  - Space contains more than just this digit.
  - Even if we sample every nanosecond from now until the end of the universe, you won't see the original six!



Motivation for Non-Linear Dimensionality Reduction

USPS Data Set Handwritten Digit

- 3648 Dimensions
  - 64 rows by 57 columns
  - Space contains more than just this digit.
  - Even if we sample every nanosecond from now until the end of the universe, you won't see the original six!



Simple Model of Digit

Rotate a 'Prototype'

Simple Model of Digit

Rotate a 'Prototype'

Simple Model of Digit

Rotate a 'Prototype'

Simple Model of Digit

Rotate a 'Prototype'

Simple Model of Digit

Rotate a 'Prototype'

Simple Model of Digit

Rotate a 'Prototype'

Simple Model of Digit

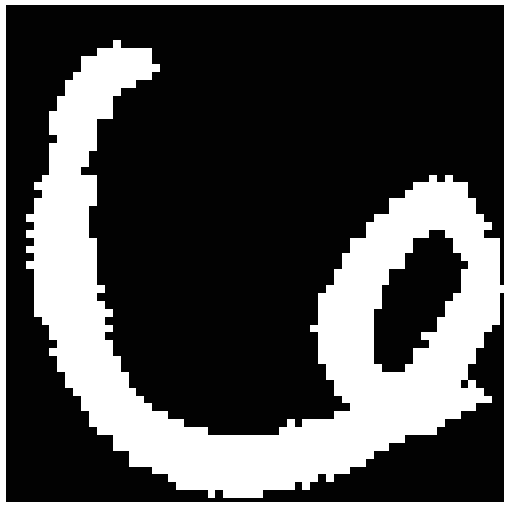
Rotate a 'Prototype'

Simple Model of Digit

Rotate a 'Prototype'

Simple Model of Digit

Rotate a 'Prototype'

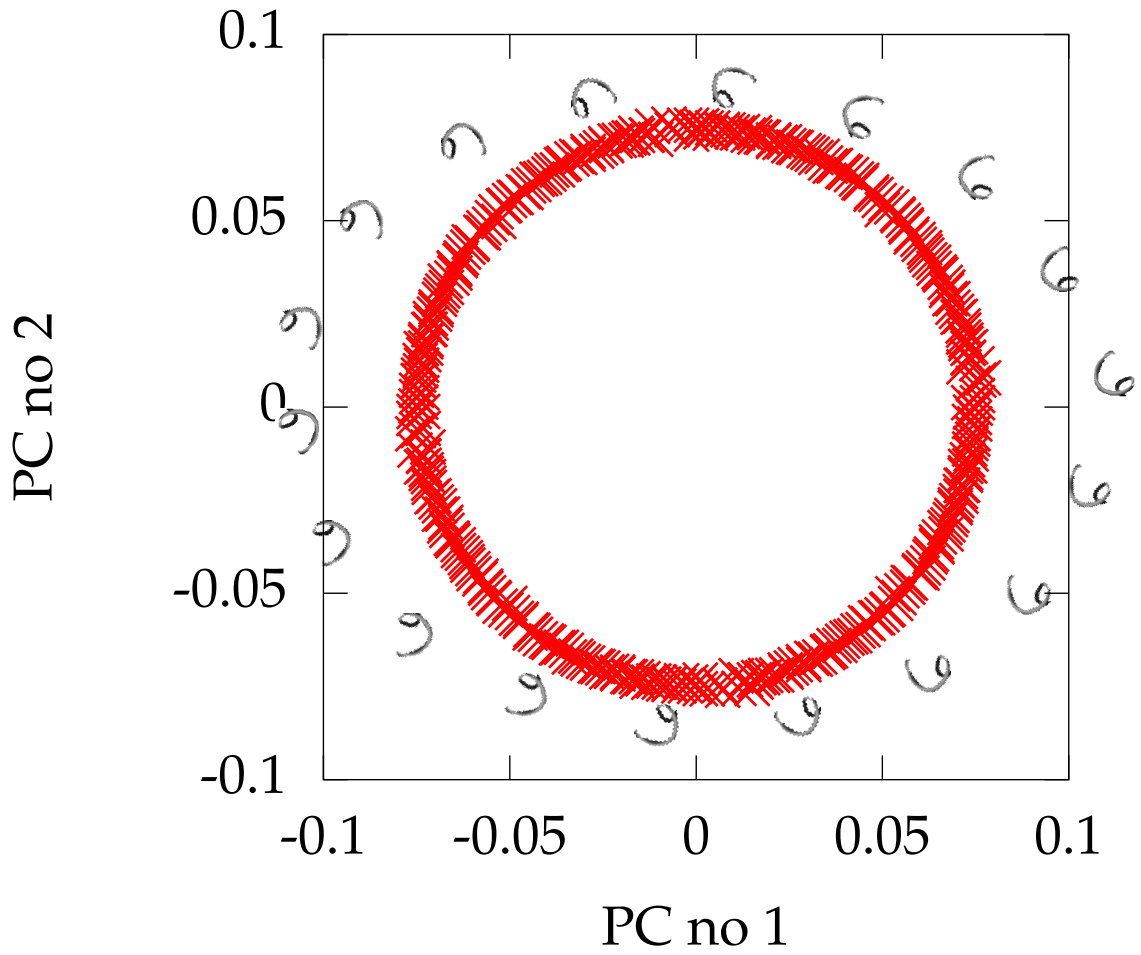


MATLAB Demo

```
demDigitsManifold([1 2], 'all')
```

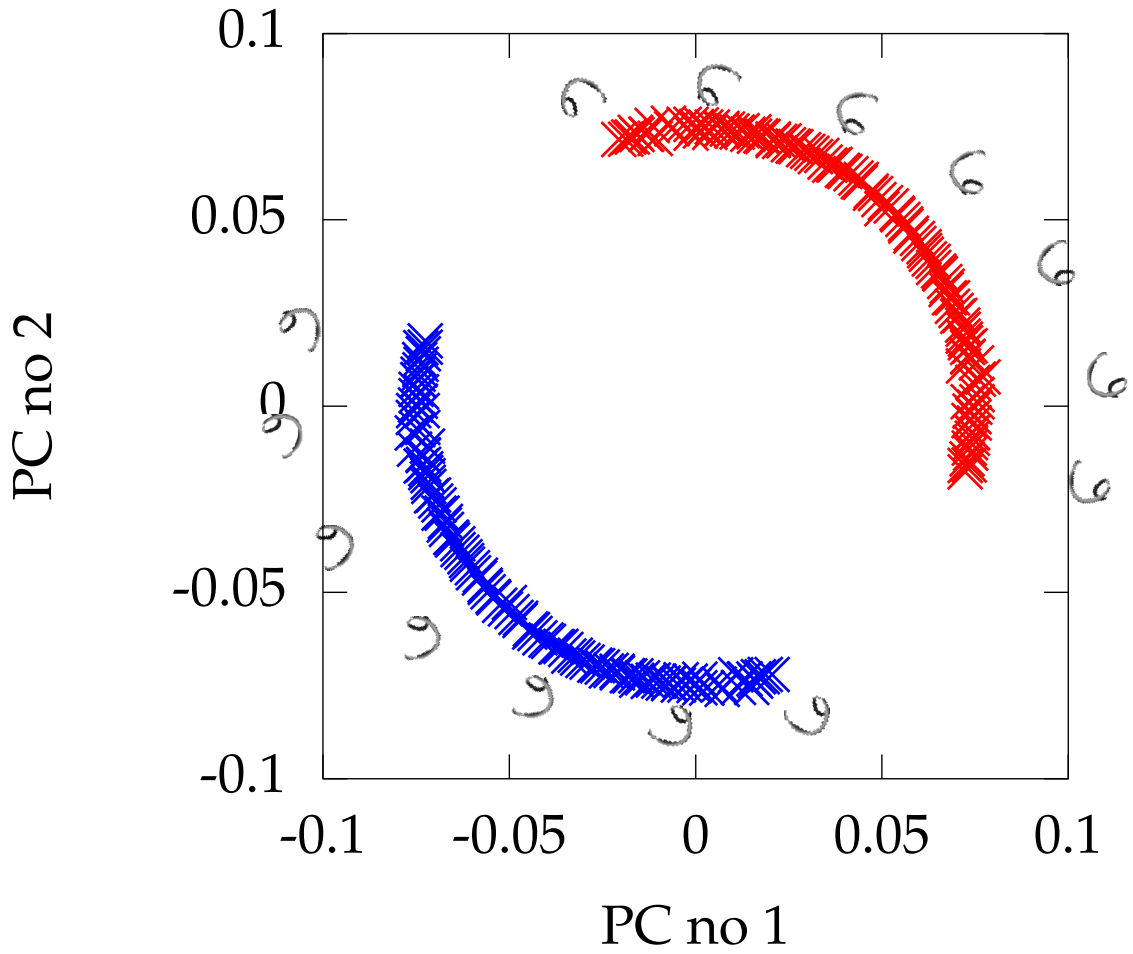
MATLAB Demo

```
demDigitsManifold([1 2], 'all')
```



MATLAB Demo

```
demDigitsManifold([1 2], 'sixnine')
```



Low Dimensional Manifolds

Pure Rotation is too Simple

- In practice the data may undergo several distortions.
  - *e.g.* digits undergo 'thinning', translation and rotation.
- For data with 'structure':
  - we expect fewer distortions than dimensions;
  - we therefore expect the data to live on a lower dimensional manifold.
- Conclusion: deal with high dimensional data by looking for lower dimensional non-linear embedding.

Data Representation

- Classical statistical approach: represent via proximities. (**?**)
- Proximity data: similarities or dissimilarities.
- Example of a dissimilarity matrix: a *squared distance matrix*.

$$d_{i,j} = \left\|\mathbf{y}_{i,:} - \mathbf{y}_{j,:}\right\|^2 = \left(\mathbf{y}_{i,:} - \mathbf{y}_{j,:}\right)^\top \left(\mathbf{y}_{i,:} - \mathbf{y}_{j,:}\right)$$

- For a data set can display as a matrix.

Interpoint Distances for Rotated Sixes

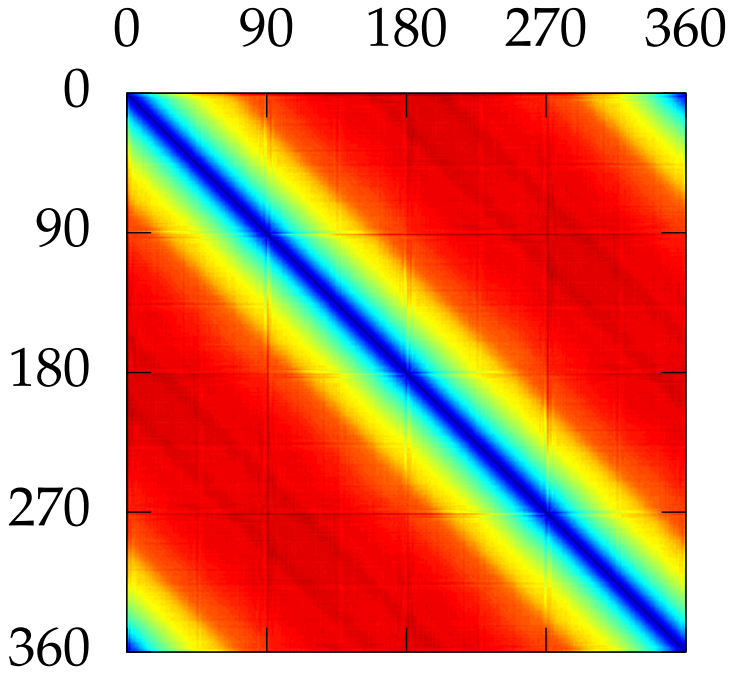


Figure: Interpoint distances for the rotated digits data.

Multidimensional Scaling

- Find a configuration of points, $\mathbf{X}$, such that each

$$\delta_{i,j} = \left\| \mathbf{x}_{i,:} - \mathbf{x}_{j,:} \right\|^2$$

  closely matches the corresponding $d_{i,j}$ in the distance matrix.
- Need an objective function for matching $\mathbf{\Delta} = \left(\delta_{i,j}\right)_{i,j}$ to $\mathbf{D} = \left(d_{i,j}\right)_{i,j}$.

Feature Selection

- An entrywise L_1 norm on difference between squared distances

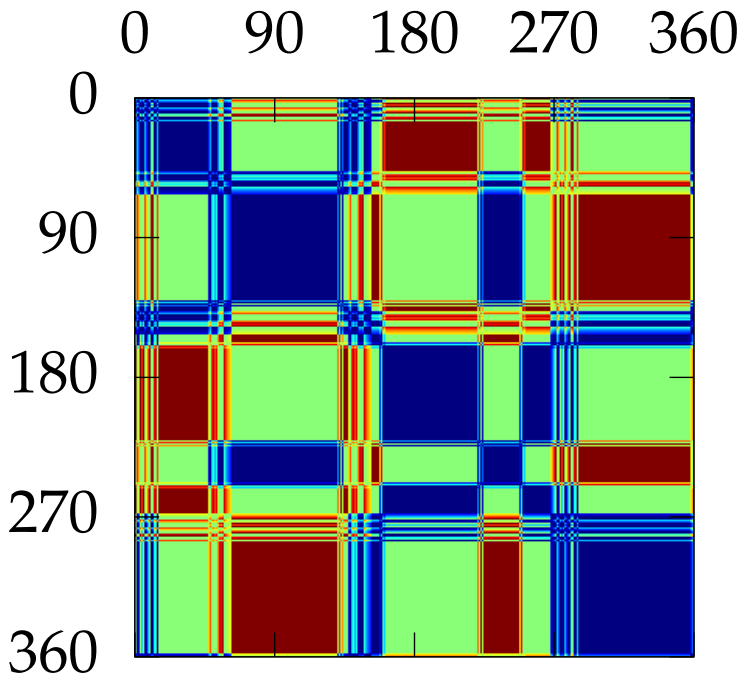
$$E(\mathbf{X}) = \sum_{i=1}^{n} \sum_{j=1}^{n} \left| d_{ij} - \delta_{ij} \right|_1 .$$

- Reduce dimension by selecting features from data set.
- Select for $\mathbf{X}$, in turn, the column from $\mathbf{Y}$ that most reduces this error until we have the desired q.
- To minimise $E(\mathbf{Y})$ we compose $\mathbf{X}$ by extracting the columns of $\mathbf{Y}$ which have the largest variance.

▸ Derive Algorithm

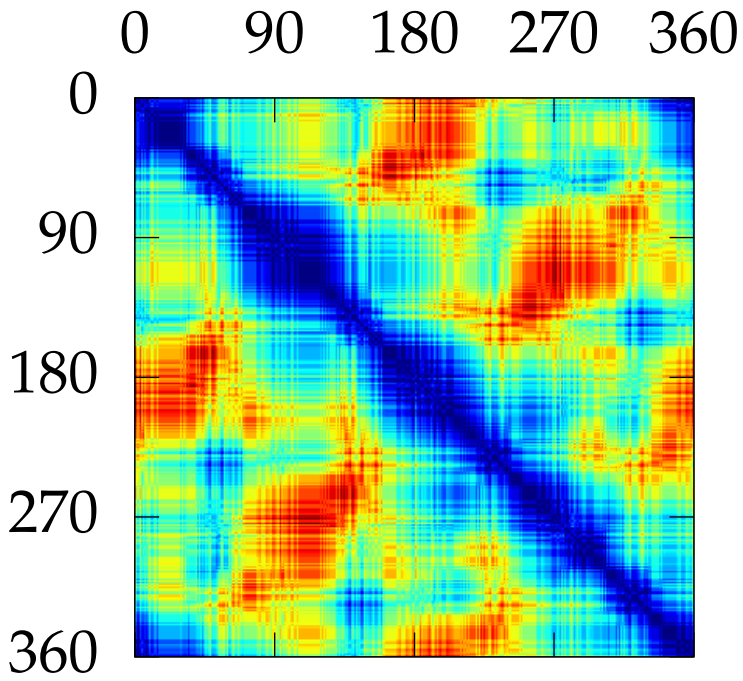
Feature Selection: Motivation

Reconstruction from Latent Space



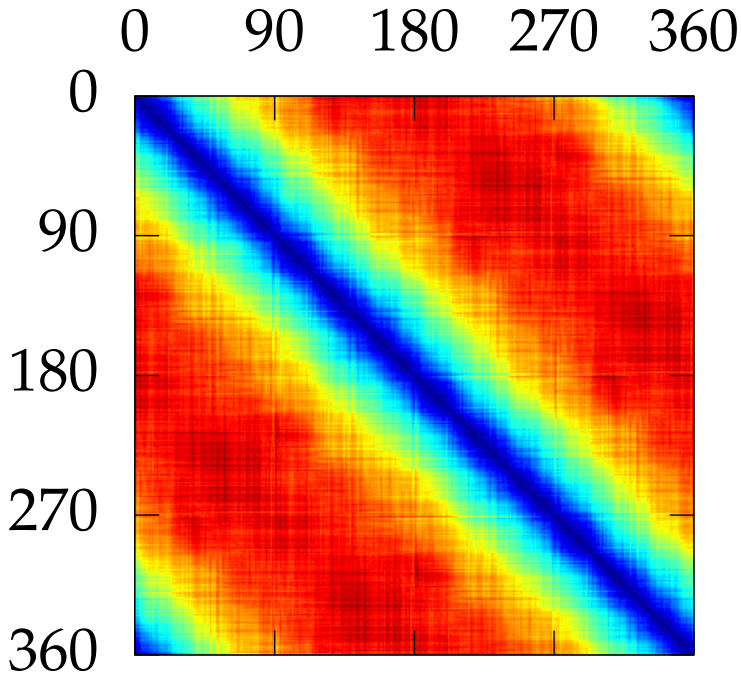
Distances reconstructed with two dimensions. MAE: 0.215.

Reconstruction from Latent Space



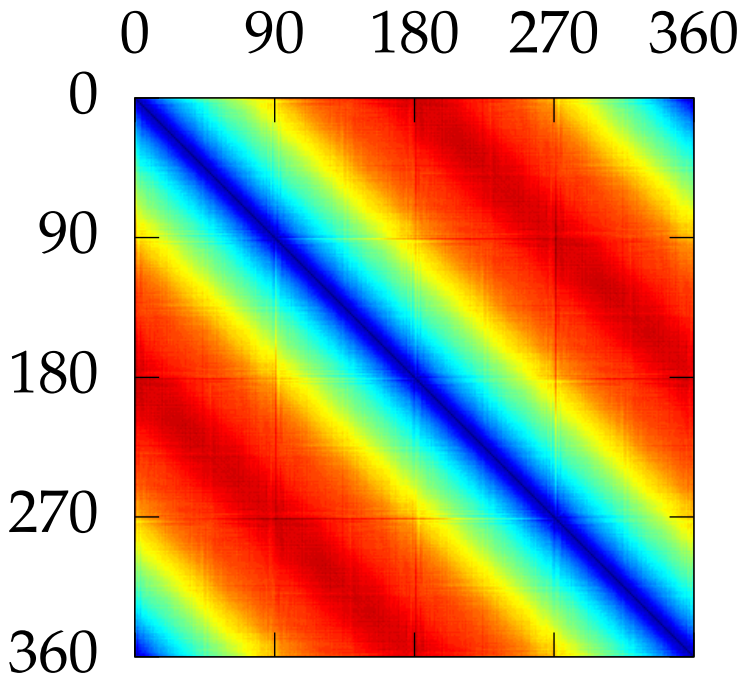
Distances reconstructed with ten dimensions. MAE: 0.214.

Reconstruction from Latent Space



Distances reconstructed with one hundred dimensions. MAE: 0.203.

Reconstruction from Latent Space



Distances reconstructed with 1000 dimensions. MAE: 0.109.

Feature Selection

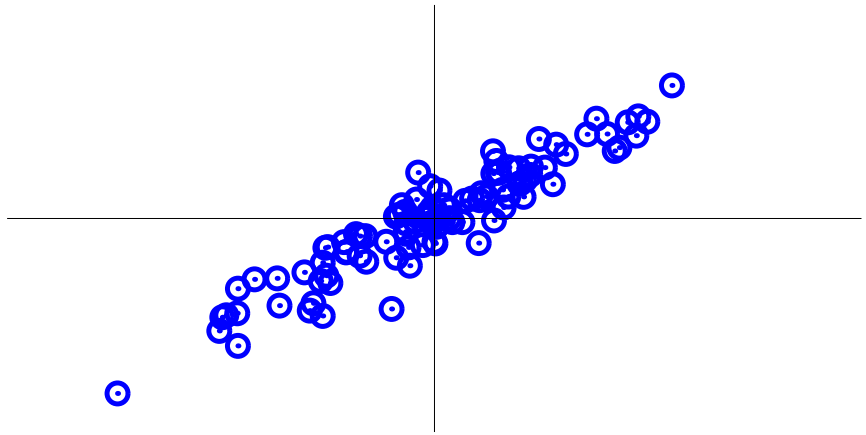


Figure: `demRotationDist`. Feature selection via distance preservation.

Feature Selection

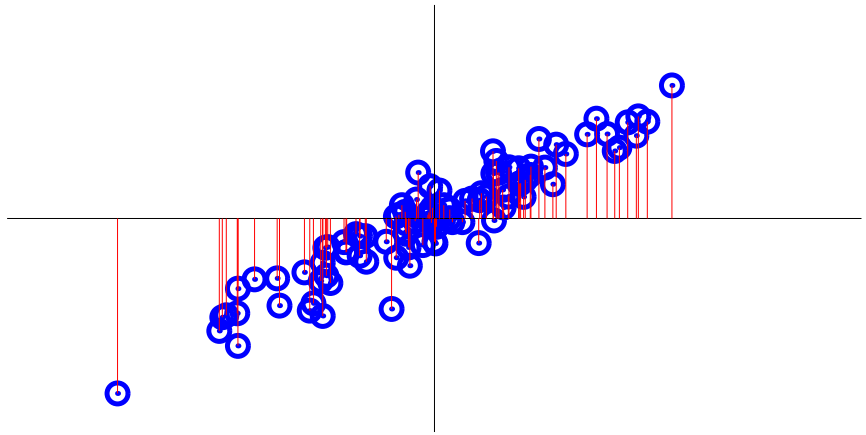


Figure: `demRotationDist`. Feature selection via distance preservation.

Feature Selection

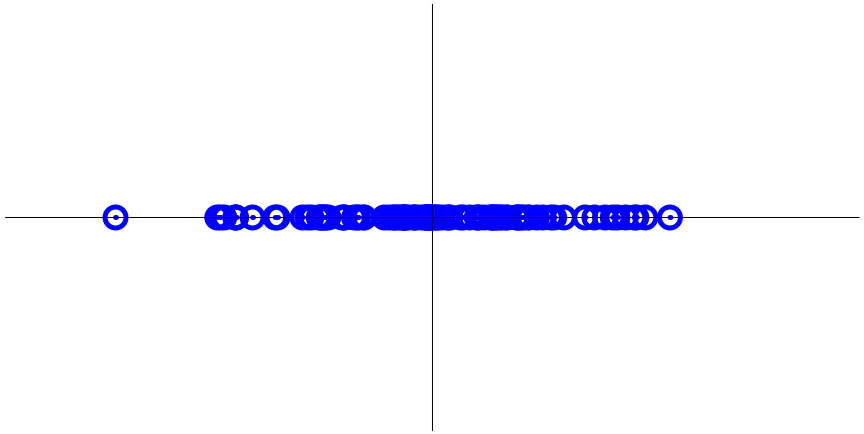


Figure: `demRotationDist`. Feature selection via distance preservation.

Feature Extraction

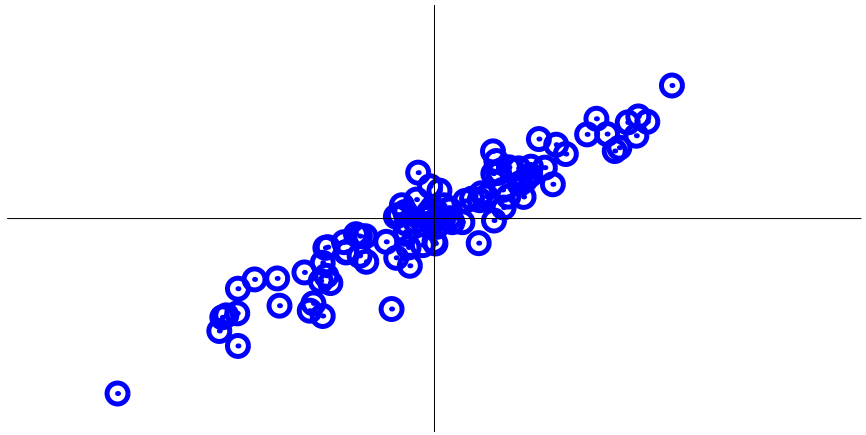


Figure: `demRotationDist`. Rotation preserves interpoint distances.

.

Feature Extraction

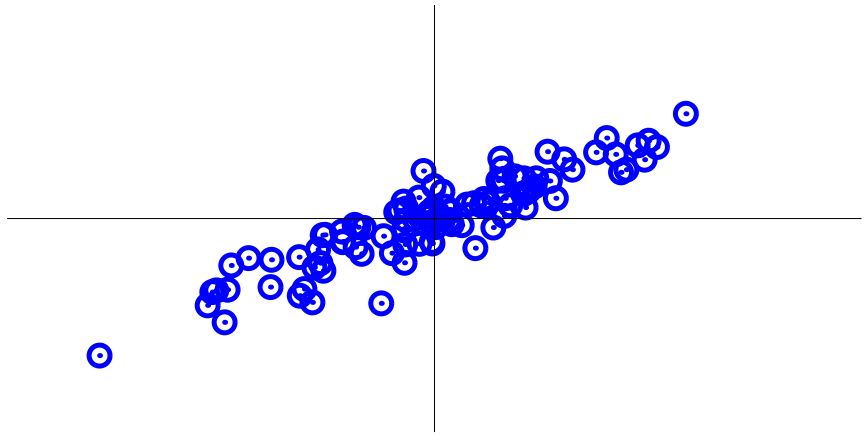


Figure: `demRotationDist`. Rotation preserves interpoint distances.

.

Feature Extraction

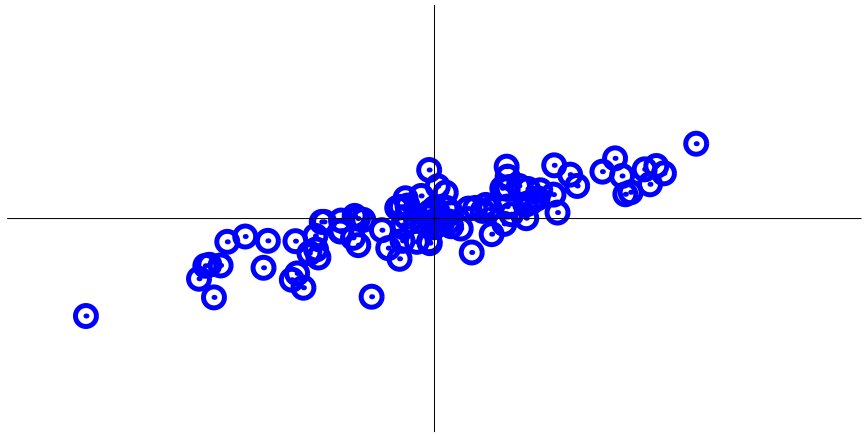


Figure: `demRotationDist`. Rotation preserves interpoint distances.

.

Feature Extraction

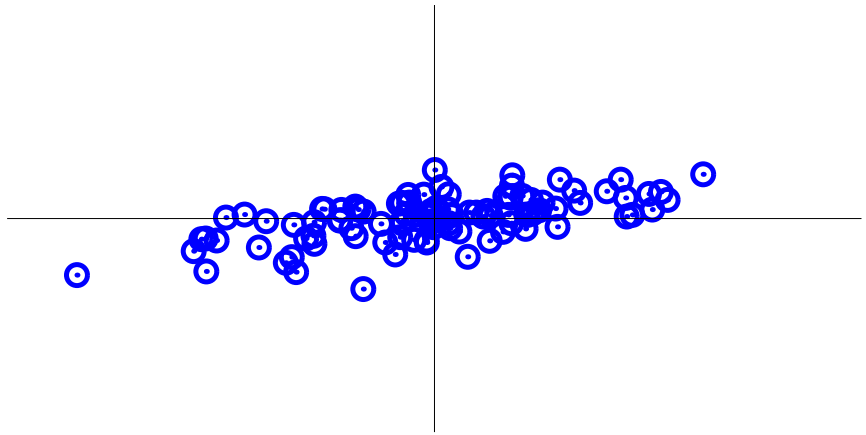


Figure: `demRotationDist`. Rotation preserves interpoint distances.

.

Feature Extraction

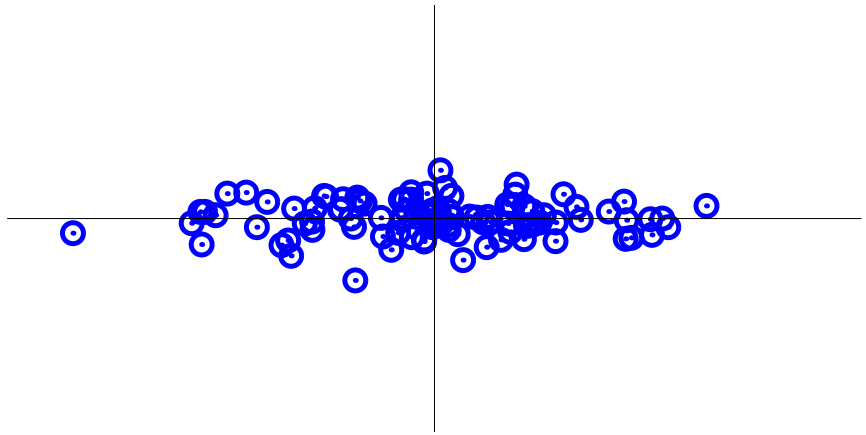


Figure: `demRotationDist`. Rotation preserves interpoint distances.

.

Feature Extraction

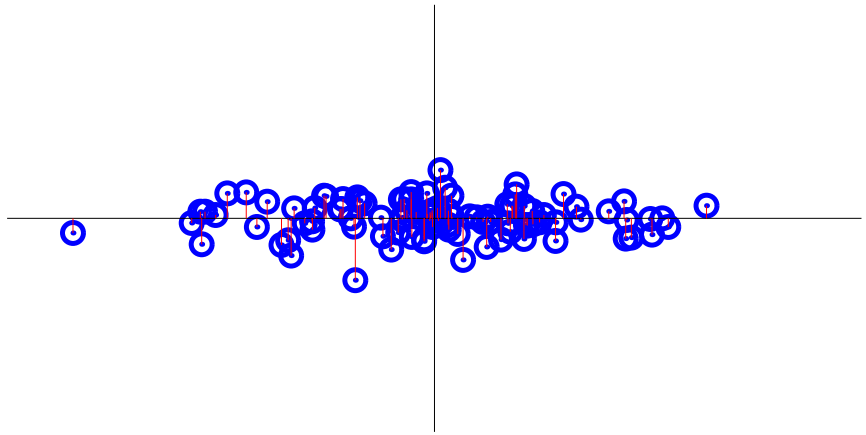


Figure: `demRotationDist`. Rotation preserves interpoint distances. Residuals are much reduced.

Feature Extraction

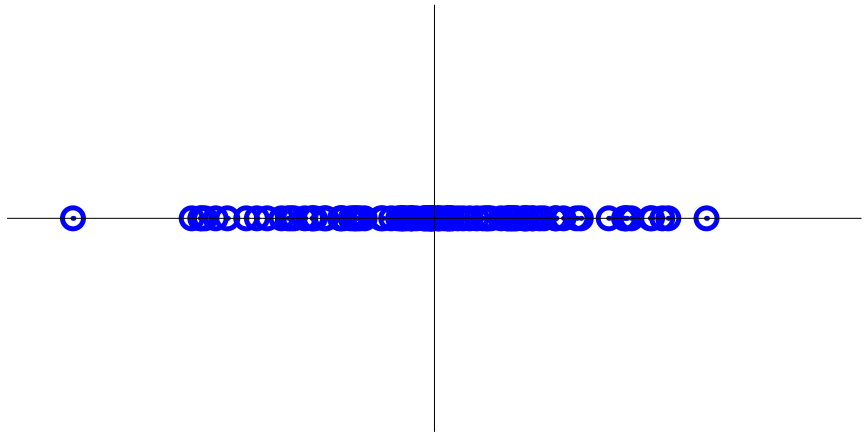


Figure: `demRotationDist`. Rotation preserves interpoint distances. Residuals are much reduced.

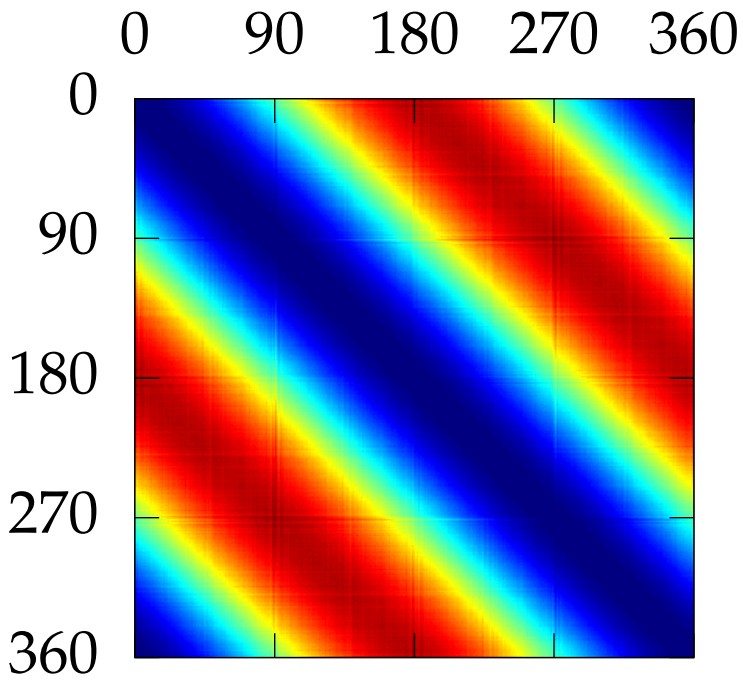
Which Rotation?

- We need the rotation that will minimise residual error.
- We already an algorithm for discarding features/directions.
- Retain features/directions with *maximum variance*.
- Error is then given by the sum of residual variances.

$$E(\mathbf{X}) = \frac{2}{p} \sum_{k=q+1}^{p} \sigma_k^2.$$

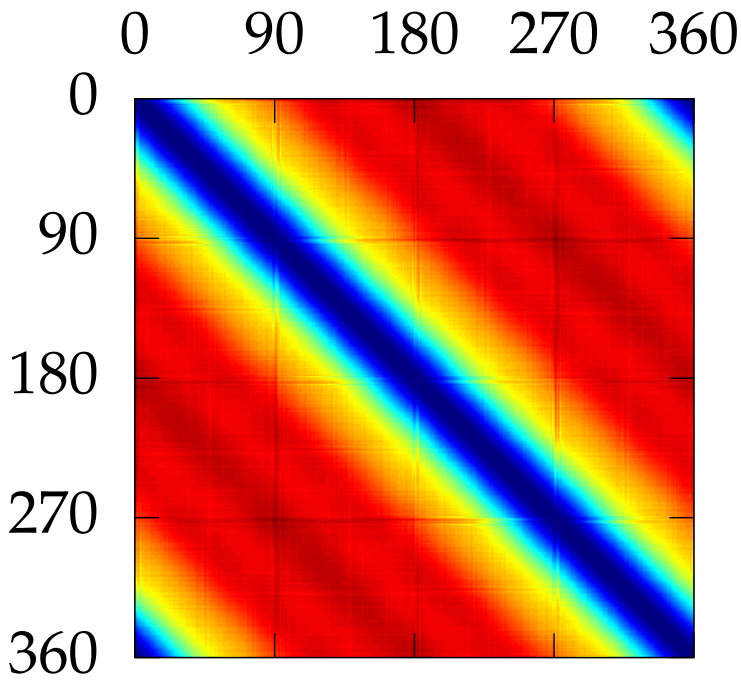
- Rotations of data matrix *do not* effect this analysis.
- Rotate data so that largest variance directions are retained.

Rotation Reconstruction from Latent Space



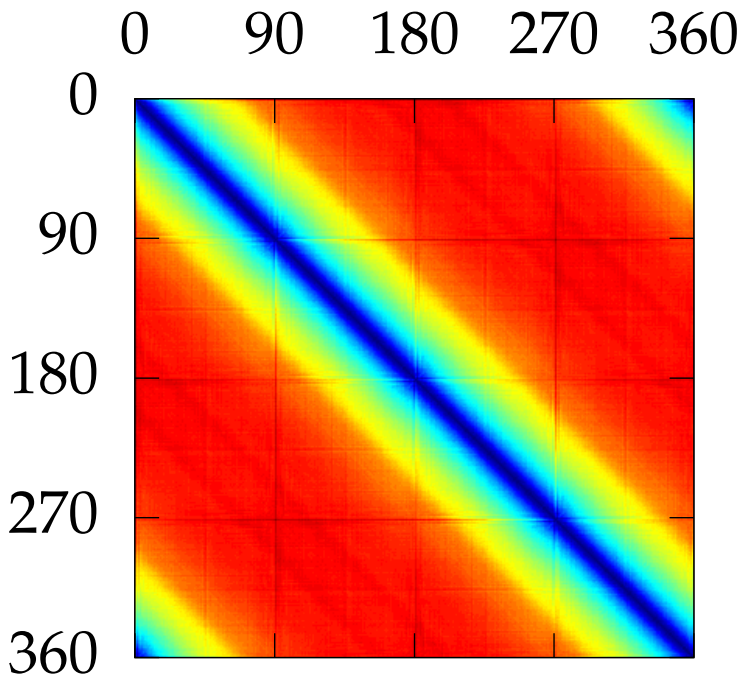
Distances reconstructed with two dimensions. MAE: 3.30×10^{-5}.

Rotation Reconstruction from Latent Space



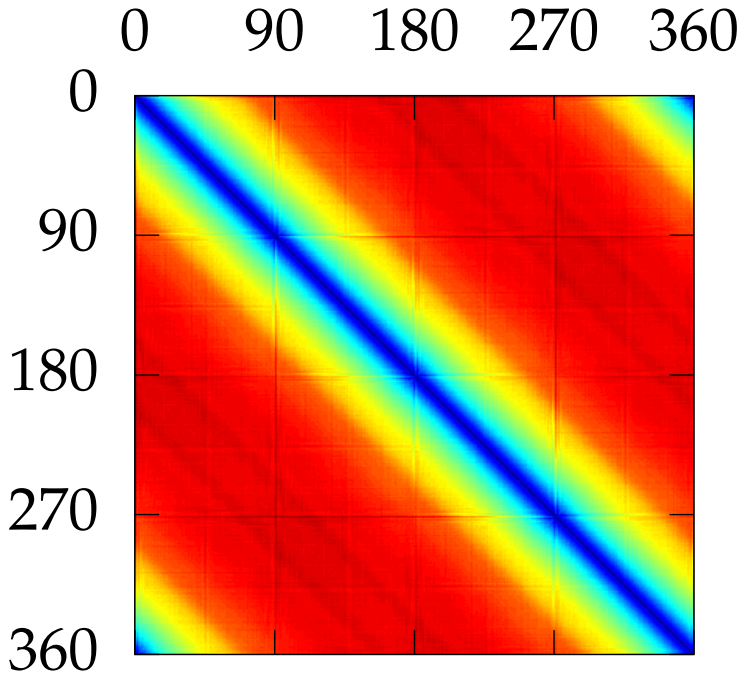
Distances reconstructed with ten dimensions. MAE: 1.52×10^{-5}.

Rotation Reconstruction from Latent Space



Distances reconstructed with one hundred dimensions. MAE: 3.85×10^{-6}.

Rotation Reconstruction from Latent Space



Distances reconstructed with 360 dimensions. MAE: 0000.

Reminder: Principal Component Analysis

- How do we find these directions?
- Find directions in data with maximal variance.
  - That's what PCA does!
- **PCA**: rotate data to extract these directions.
- **PCA**: work on the sample covariance matrix $\mathbf{S} = n^{-1}\hat{\mathbf{Y}}^\top\hat{\mathbf{Y}}$.

▸ Prove PCA

Principal Coordinates Analysis

- The rotation which finds directions of maximum variance is the eigenvectors of the covariance matrix.
- The variance in each direction is given by the eigenvalues.
- **Problem:** working directly with the sample covariance, **S**, may be impossible.
- For example: perhaps we are given distances between data points, but not absolute locations.
  - No access to absolute positions: cannot compute original sample covariance.

▸ Principal Coordinates

Equivalent Eigenvalue Problems

- Principal Coordinate Analysis operates on $\hat{\mathbf{Y}}\hat{\mathbf{Y}}^\top$.
- Two eigenvalue problems are equivalent. One solves for the rotation, the other solves for the location of the rotated points.
- When $p < n$ it is easier to solve for the rotation, $\mathbf{R}_q$. But when $p > n$ we solve for the embedding (principal coordinate analysis).
- In MDS we may not know $\mathbf{Y}$, cannot compute $\hat{\mathbf{Y}}^\top\hat{\mathbf{Y}}$ from distance matrix.
- Can we compute $\hat{\mathbf{Y}}\hat{\mathbf{Y}}^\top$ instead?

Note: Centering and Squared Distances

- Consider matrix form of squared distance,

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

- 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:

$$-\frac{1}{2}\mathbf{H}\mathbf{D}\mathbf{H} = \mathbf{H}\mathbf{Y}\mathbf{Y}^\top\mathbf{H} = \hat{\mathbf{Y}}\hat{\mathbf{Y}}^\top.$$

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

The Covariance Interpretation

- $n^{-1}\hat{\mathbf{Y}}^\top\hat{\mathbf{Y}}$ is the data covariance.
- $\hat{\mathbf{Y}}\hat{\mathbf{Y}}^\top$ is a centred inner product matrix.
  - Also has an interpretation as a covariance matrix (Gaussian processes).
  - It expresses correlation and anti correlation between *data points*.
  - Standard covariance expresses correlation and anti correlation between *data dimensions*.

Distance to Similarity: Gaussian Covariances

- Translate between covariance and distance.
  - Consider a vector sampled from a zero mean Gaussian distribution,
$$\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{K}).$$
  - Expected square distance between two elements of this vector is
$$d_{i,j} = \left\langle \left(z_i - z_j\right)^2 \right\rangle$$
$$d_{i,j} = \left\langle z_i^2 \right\rangle + \left\langle z_j^2 \right\rangle - 2\left\langle z_i z_j \right\rangle$$
    under a zero mean Gaussian with covariance given by $\mathbf{K}$ this is
$$d_{i,j} = k_{i,i} + k_{j,j} - 2k_{i,j}.$$

Standard Transformation

- This transformation is known as the *standard transformation* between a similarity and a distance (Mardia et al., 1979, pg 402).
- If the covariance is of the form $\mathbf{K} = \hat{\mathbf{Y}}\hat{\mathbf{Y}}^\top$ then $k_{i,j} = \hat{\mathbf{y}}_{i,:}^\top \hat{\mathbf{y}}_{j,:}$ and

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

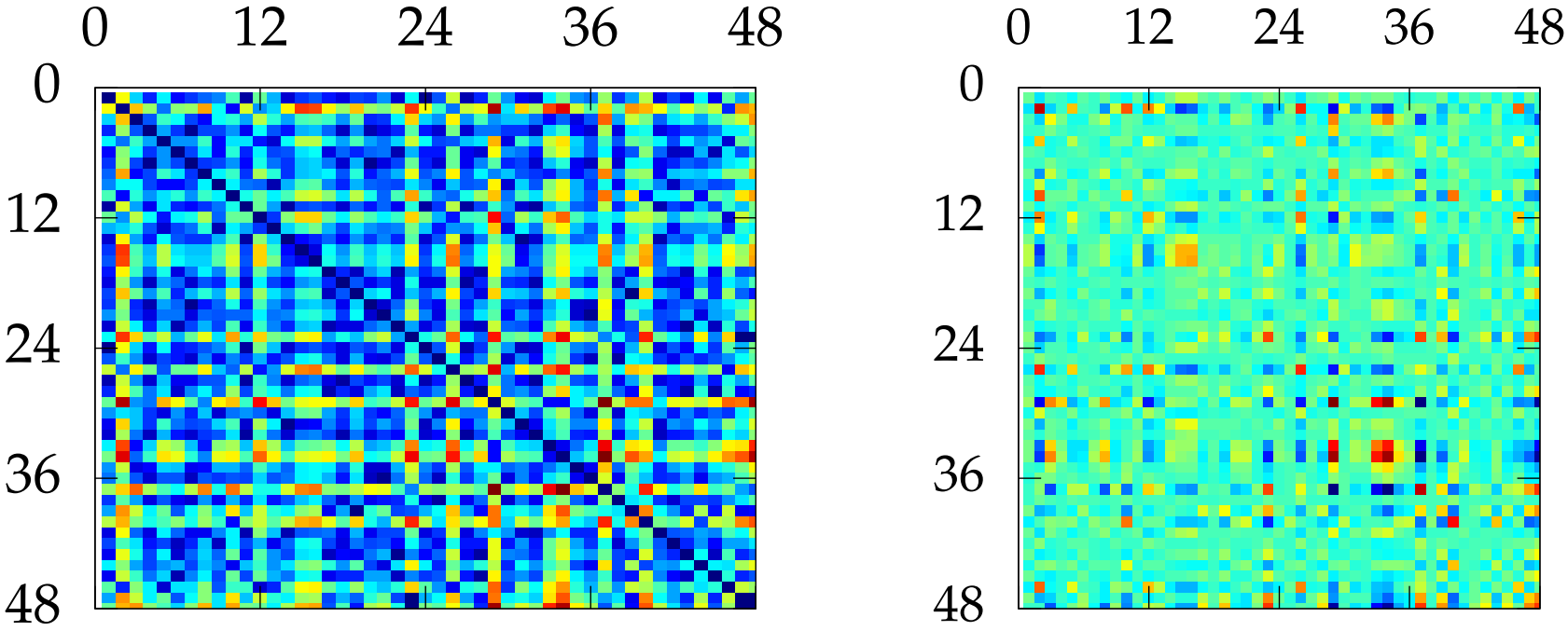
- For other squared distance matrices this gives us an approach to covert to a similarity matrix or kernel matrix so we can perform classical MDS.

Example: Road Distances with Classical MDS

- Classical example: redraw a map from road distances (see e.g. Mardia et al., 1979).
- Here we use distances across Europe.
  - Between each city we have road distance.
  - Enter these in a distance matrix.
  - Convert to a similarity matrix using the covariance interpretation.
  - Perform eigendecomposition.
- See `http://inverseprobability.com/dimred/` for the data we used.

Distance Matrix

Convert distances to similarities using "covariance interpretation".



Left: road distances between European cities.*Right*: Equivalent similarity.

Example: Road Distances with Classical MDS

Beware Negative Eigenvalues

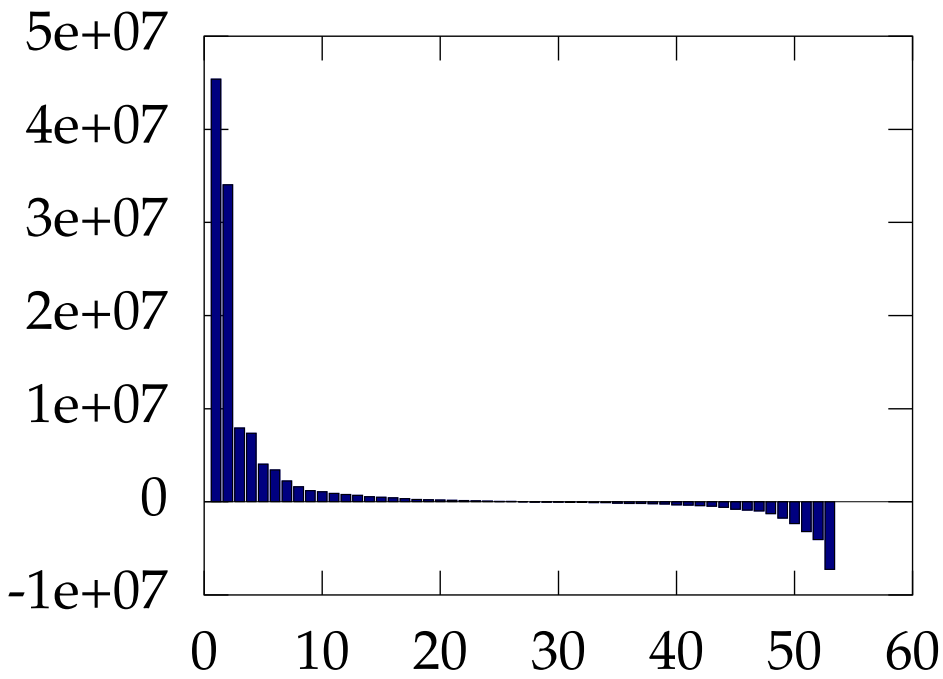
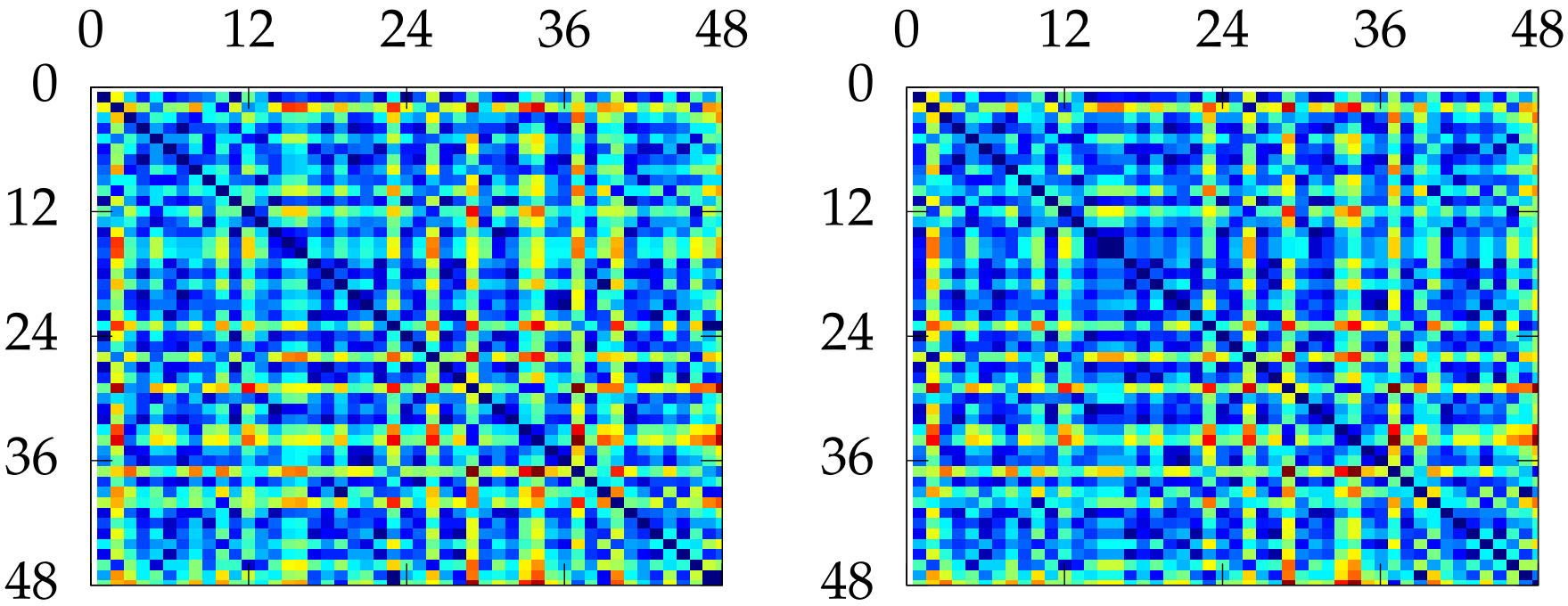


Figure: Eigenvalues of the similarity matrix are negative in this case.

European Cities Distance Matrices



Left: Original Distance matrix. *Right:* Reconstructed distance matrix.

Other Distance Similarity Measures

- Can use similarity/distance of your choice.
- Beware though!
  - The similarity must be positive semi definite for the distance to be Euclidean.
  - For more details see Mardia et al. (1979, Theorem 14.2.2).

Nonlinear Dimensionality Reduction

- How do we get a nonlinear algorithm?
- One idea:
  1. Use linear algorithm (CMDS or Principal Coordinate Analysis).
  2. Make distance matrix *nonlinearly* related to original data.

Nonlinear Dimensionality Reduction

- Let's nonlinearly map data to a new space, and compute distances there.
- Do this using *basis functions*

$$f_i = f(\mathbf{y}_{i,:}) = \sum_{j=1}^{m} w_j \phi_j(\mathbf{y}_{i,:})$$

$$d_{i,j} = (z_i - z_j)^2$$

Exponentiated Quadratic Basis Functions

- Consider these basis functions:

$$\phi_j(\mathbf{y}_{j,:}) = \exp\left(-\frac{1}{\ell^2}\left\|\mathbf{y}_{j,:} - \boldsymbol{\mu}_i\right\|_2^2\right)$$

- take

$$\phi_{i,j} = \phi_j(\mathbf{y}_{i,:})$$

giving basis vector, $\boldsymbol{\phi}_{i,:}$, and design matrix

$$\boldsymbol{\Phi} = [\boldsymbol{\phi}_{1,:} \dots \boldsymbol{\phi}_{n,:}]^\top \in \Re^{n\times m}.$$

Matrix Notation

- In matrix notation we have

$$f(\mathbf{y}_{i,:}) = \mathbf{w}^\top \boldsymbol{\phi}_{i,:} = f_i.$$

- Which parameters $\mathbf{w}$?
- Let's generate random functions: introduce a probability density for $p(\mathbf{w})$.
- Compute *expected squared distance*. Squared distance is:

$$(f_i - f_j)^2 = (\boldsymbol{\phi}_{i,:}^\top \mathbf{w} - \boldsymbol{\phi}_{j,:}^\top \mathbf{w})^2$$

Expected Square Distance

- We can rewrite this as

$$(f_i - f_j)^2 = (\boldsymbol{\phi}_{i,:} - \boldsymbol{\phi}_{j,:})^\top \mathbf{w}\mathbf{w}^\top (\boldsymbol{\phi}_{i,:} - \boldsymbol{\phi}_{j,:}).$$

  Take expectation under $p(\mathbf{w})$

$$\left\langle (f_i - f_j)^2 \right\rangle = (\boldsymbol{\phi}_{i,:} - \boldsymbol{\phi}_{j,:})^\top \left\langle \mathbf{w}\mathbf{w}^\top \right\rangle_{p(\mathbf{w})} (\boldsymbol{\phi}_{i,:} - \boldsymbol{\phi}_{j,:}).$$

- If second moment of $p(\mathbf{w})$ is $\mathbf{I}$,

$$\left\langle \mathbf{w}\mathbf{w}^\top \right\rangle_{p(\mathbf{w})} = \mathbf{I}$$

  then

$$\left\langle (f_i - f_j)^2 \right\rangle = (\boldsymbol{\phi}_{i,:} - \boldsymbol{\phi}_{j,:})^\top (\boldsymbol{\phi}_{i,:} - \boldsymbol{\phi}_{j,:}).$$

- If $\langle \mathbf{w} \rangle_{p(\mathbf{w})} = \mathbf{0}$ then covariance $\text{cov}(\mathbf{w}) = \mathbf{I}$.

Basis Functions

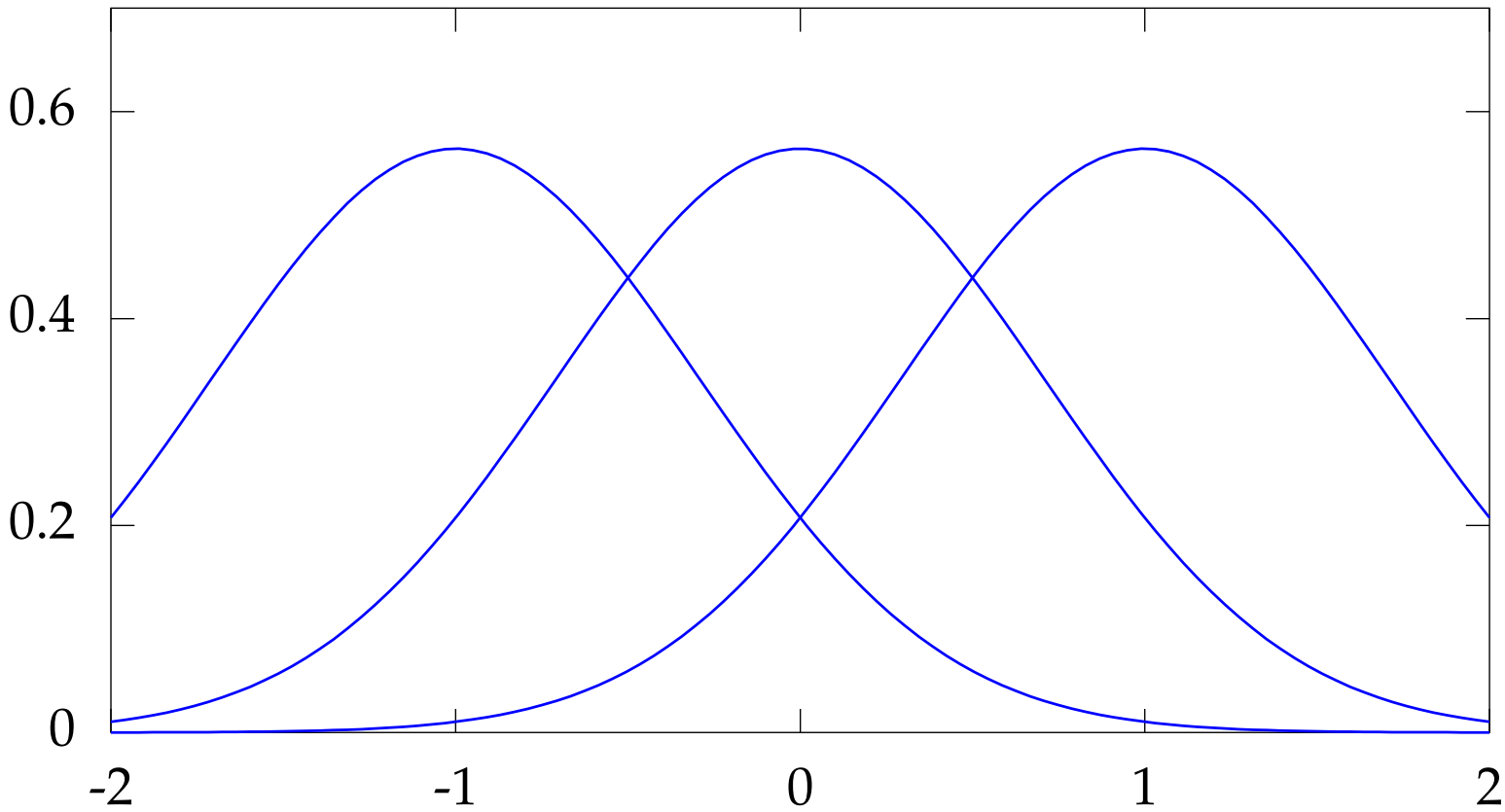


Figure: A small set of exponentiated quadratic basis functions with centers at -1, 0, and 1. The lengthscale of the basis functions is given by $\ell = 1$.

Expected Squared Distances

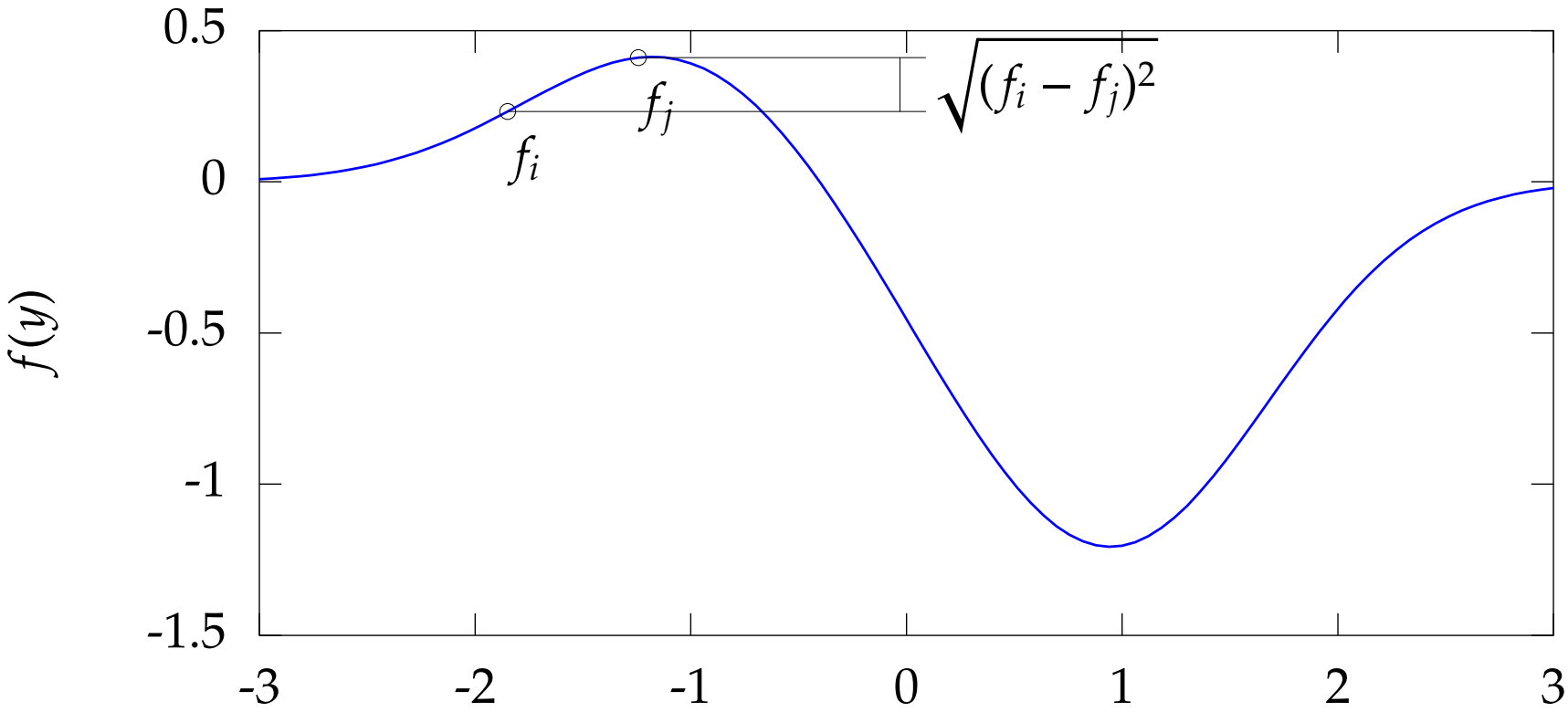


Figure: Distance between two points in the function $f(y)$. A 3 dimensional vector, $\mathbf{w}$, is sampled from a Gaussian with zero mean and unit covariance. This vector is used to weight the different basis functions producing the random function shown.

Expected Squared Distances

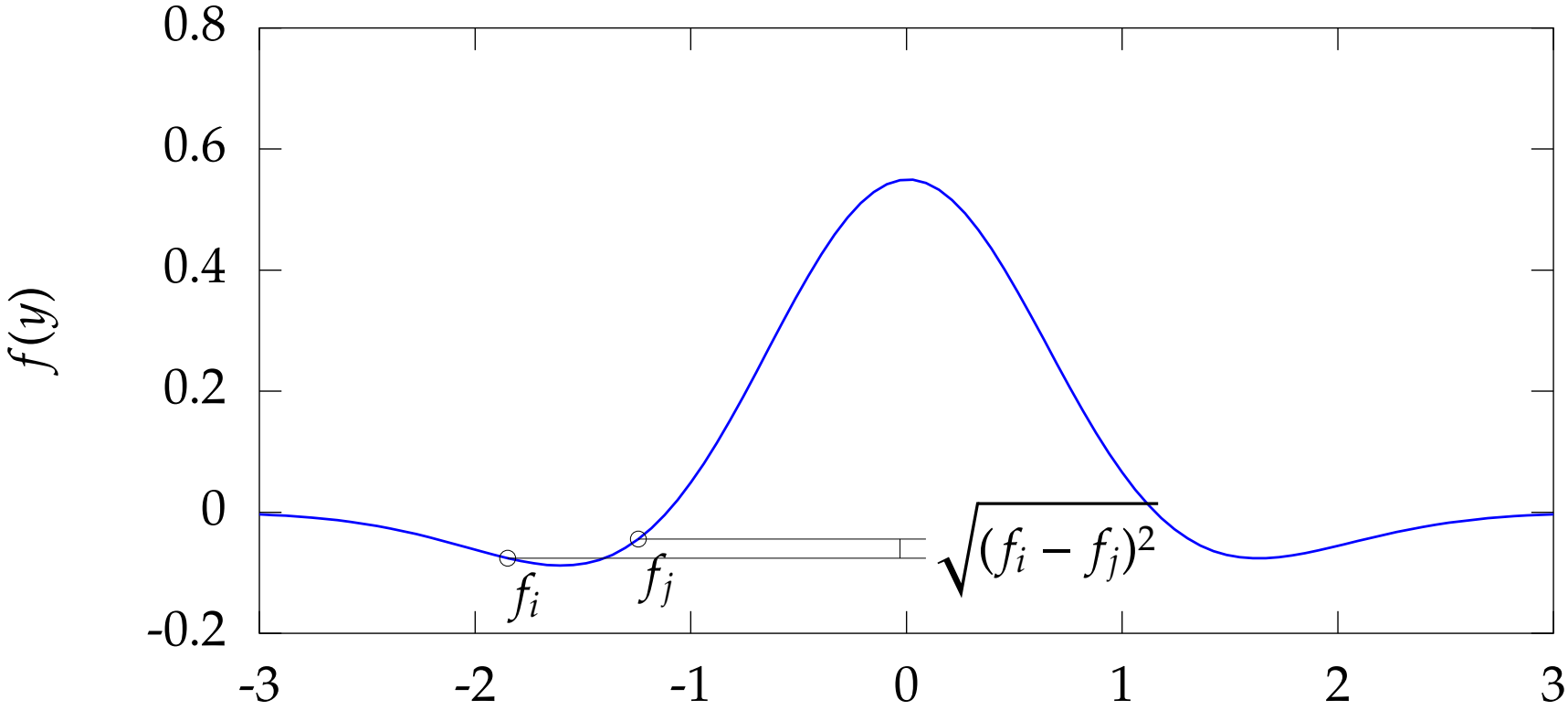


Figure: Distance between two points in the function $f(y)$. A 3 dimensional vector, $\mathbf{w}$, is sampled from a Gaussian with zero mean and unit covariance. This vector is used to weight the different basis functions producing the random function shown.

Expected Squared Distances

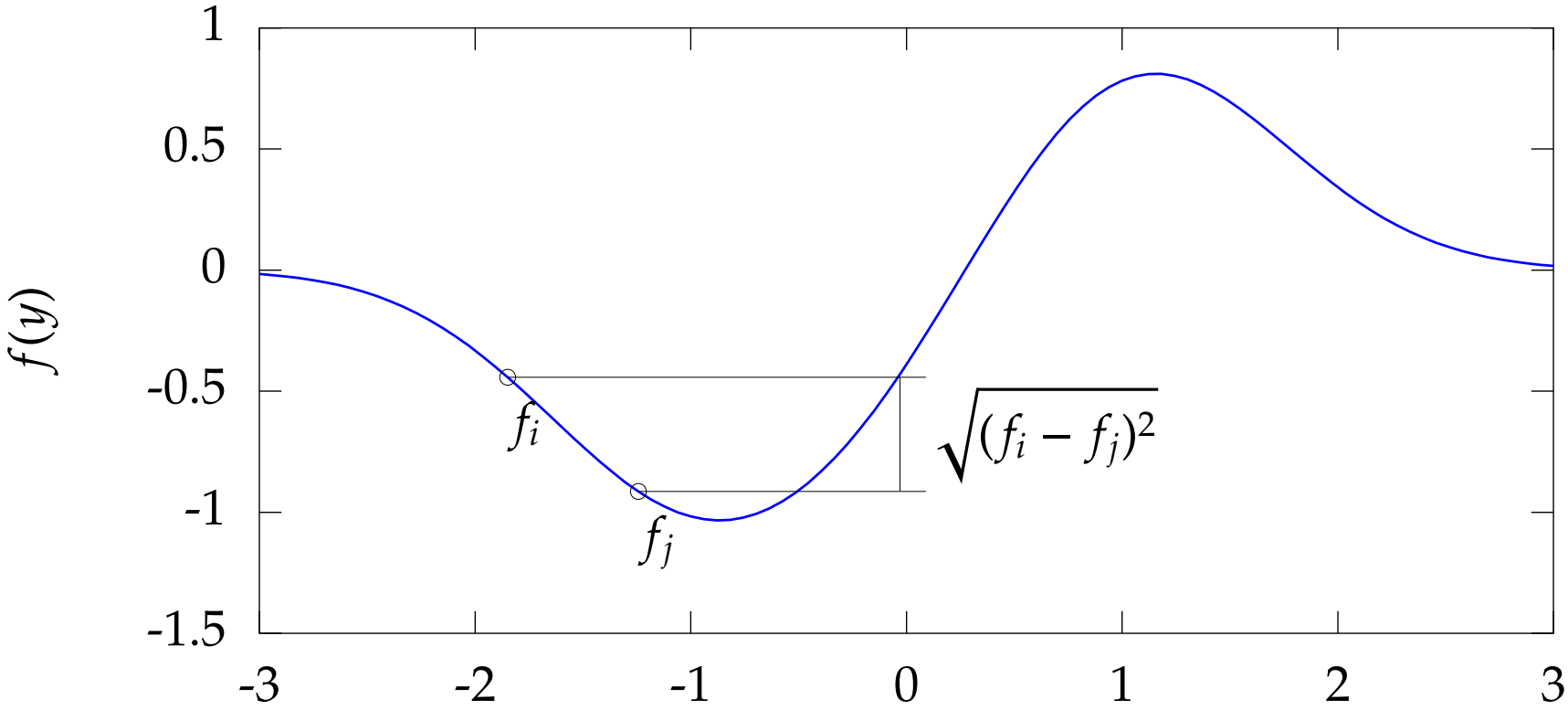


Figure: Distance between two points in the function $f(y)$. A 3 dimensional vector, $\mathbf{w}$, is sampled from a Gaussian with zero mean and unit covariance. This vector is used to weight the different basis functions producing the random function shown.

Expected Squared Distances

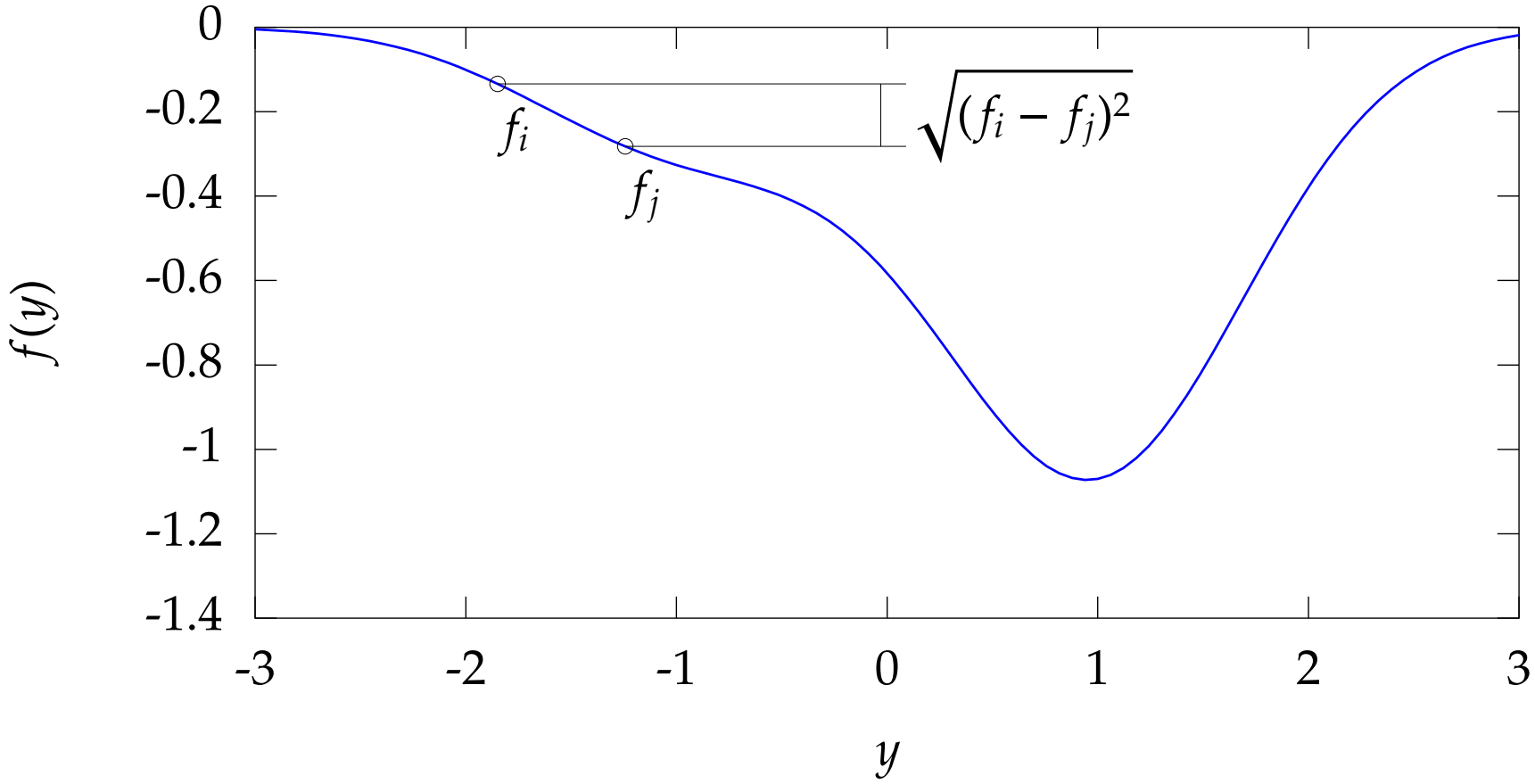


Figure: Distance between two points in the function $f(y)$. A 3 dimensional vector, $\mathbf{w}$, is sampled from a Gaussian with zero mean and unit covariance. This vector is used to weight the different basis functions producing the random function shown.

Number and Location of Basis

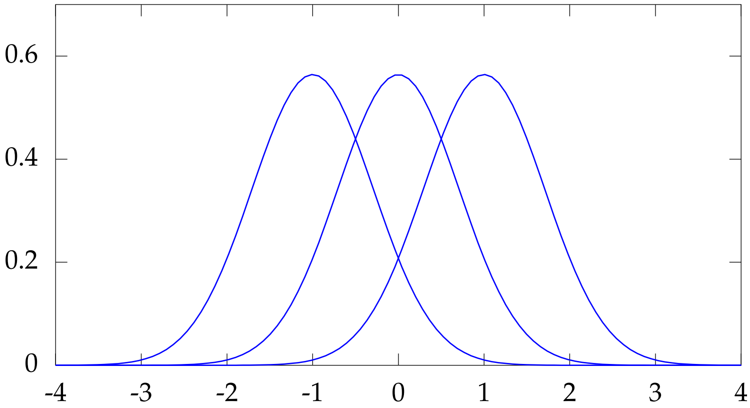


Figure: The exponentiated quadratic basis functions with centers at −1, 0, and 1. As we move away from the centers of the basis functions they decay towards zero.

Problems for Data from Outside Basis

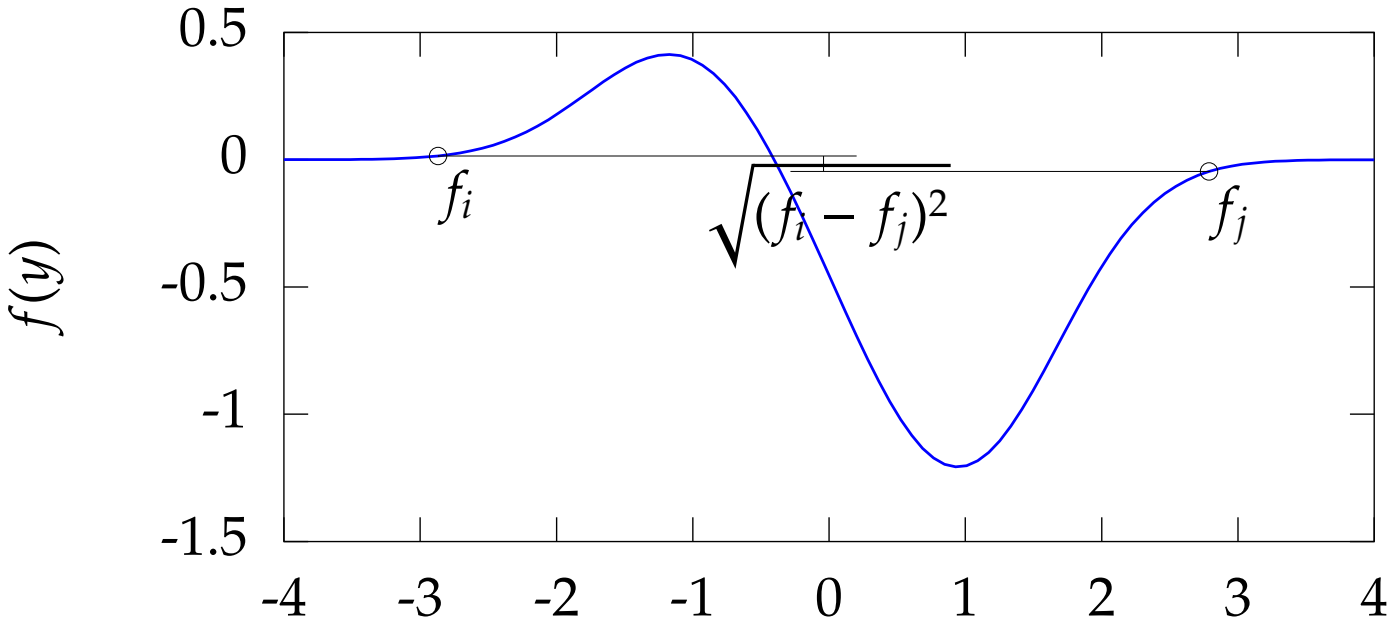


Figure: Distance between two points in the function $f(y)$. Now the locations are far apart in y. However, since they are both in regions where the response from the basis set is small, the distance between the points after mapping through the function, $f(y)$, is small.

Problems for Data from Outside Basis

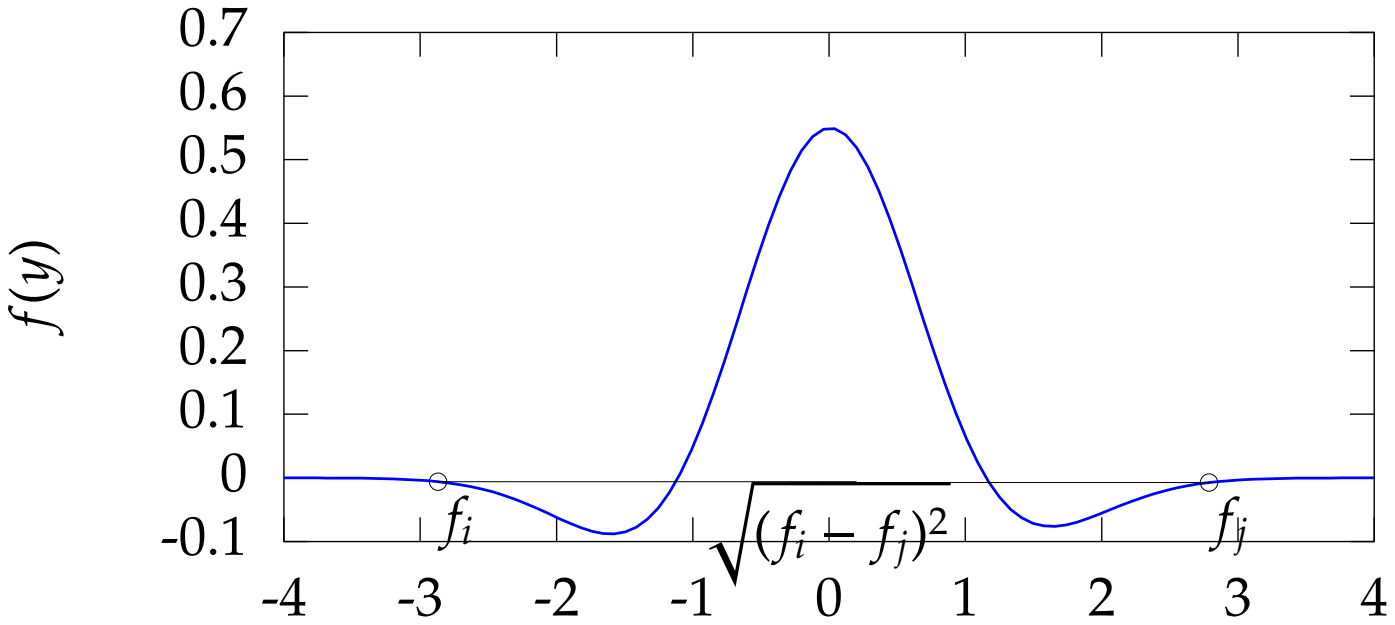


Figure: Distance between two points in the function $f(y)$. Now the locations are far apart in y. However, since they are both in regions where the response from the basis set is small, the distance between the points after mapping through the function, $f(y)$, is small.

Problems for Data from Outside Basis

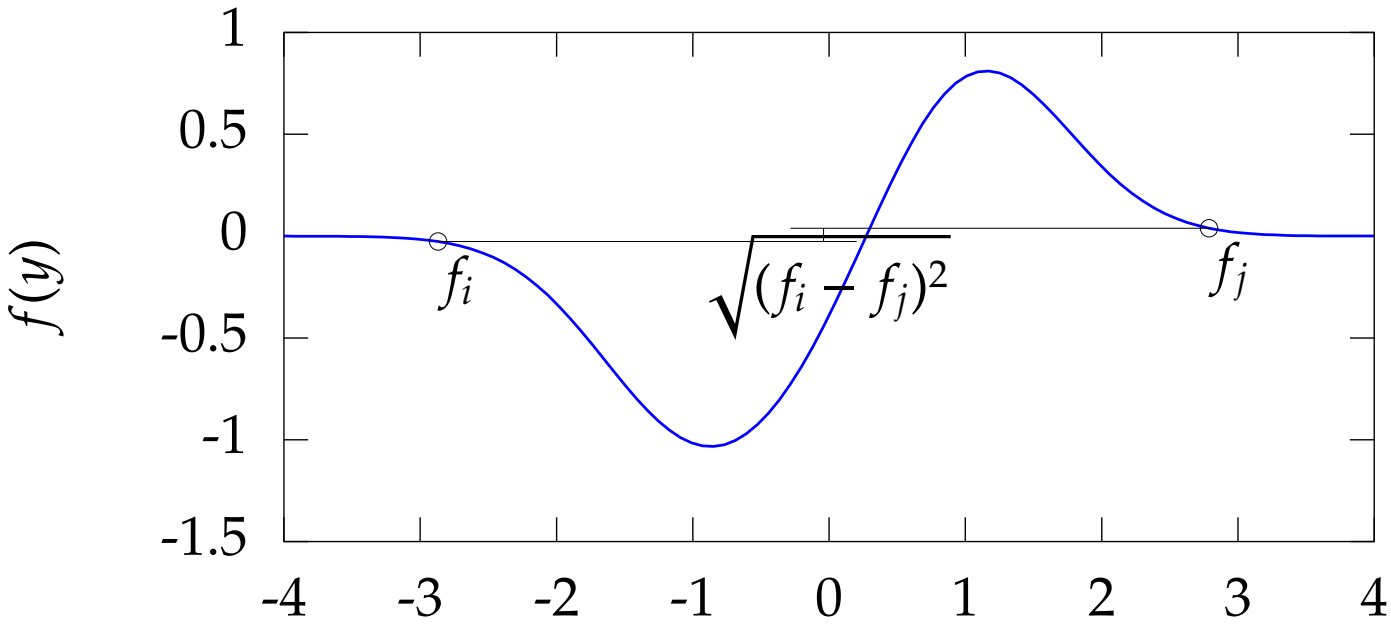


Figure: Distance between two points in the function $f(y)$. Now the locations are far apart in y. However, since they are both in regions where the response from the basis set is small, the distance between the points after mapping through the function, $f(y)$, is small.

Problems for Data from Outside Basis

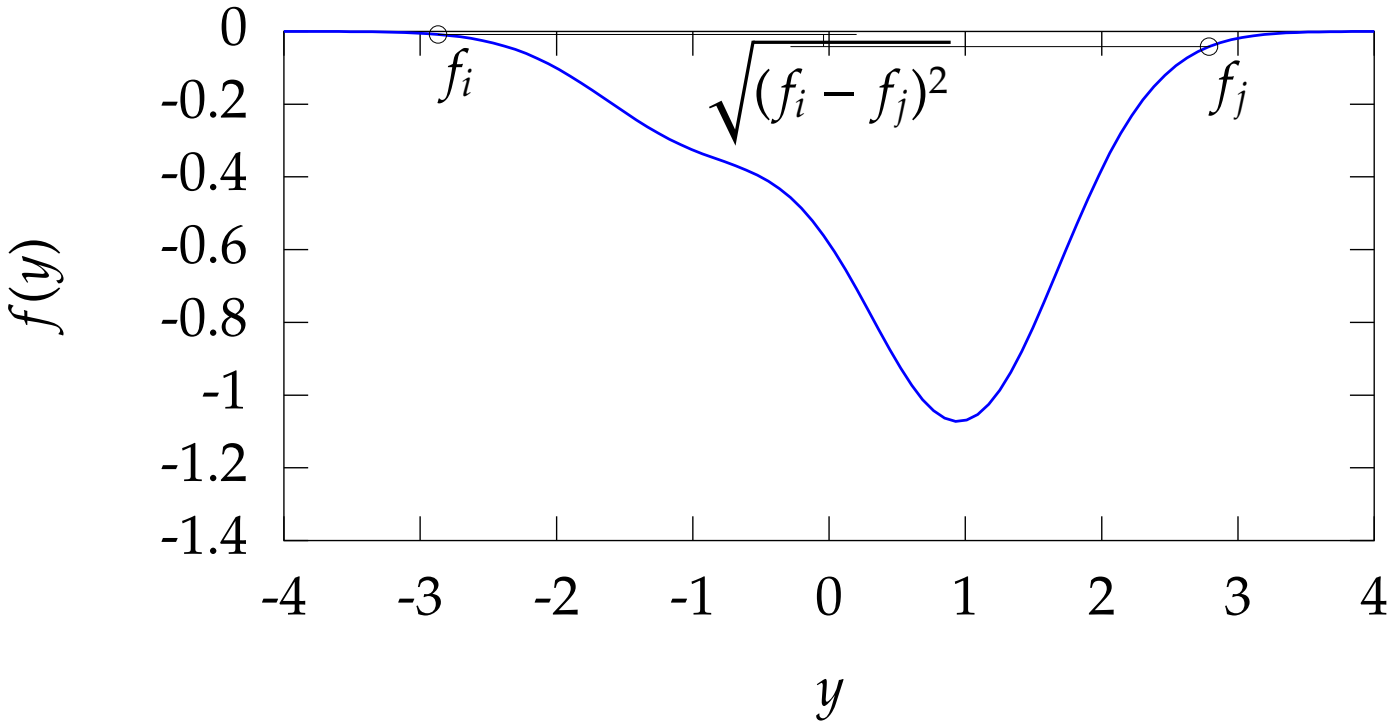


Figure: Distance between two points in the function $f(y)$. Now the locations are far apart in y. However, since they are both in regions where the response from the basis set is small, the distance between the points after mapping through the function, $f(y)$, is small.

Extending the Basis

- Distances are small despite data being far apart.
- Side effect of bad basis function placement.
- For exponentiated quadratic basis function elegant solution: place basis all across the y space.
- This leads to a kernel method.

An Infinite Basis

- We have functions of the form

$$f(y) = \sum_{k=1}^{m} w_k \exp\left(-\frac{(y - a - k\Delta\mu)^2}{\ell^2}\right),$$

  if we set the location parameter of each $\phi_k(y)$ to

$$\mu_k = a + k\Delta\mu.$$

- Distances in feature space are dependent on the inner product between basis vectors.

Infinite Basis Functions

- Decrease $\Delta\mu$ to increase m.
- The inner product between the basis functions becomes

$$k(y, y') = \frac{\alpha}{\sqrt{2\pi\ell^2}} \exp\left(-\frac{(y - y')^2}{2\ell^2}\right).$$

Kernelization

- This procedure for moving from inner products, $\phi(y)^\top \phi(y')$, to covariance functions, $k(y, y')$, is sometimes known as kernelization (Schölkopf and Smola, 2001).
- $k(y, y')$ has the properties of a Mercer kernel.
- This same property allows $k(y, y')$ to be used as a *covariance function*: a function that can generate a covariance matrix.
- The mapping from data to distance is now a Gaussian process (O'Hagan, 1978; Williams; Rasmussen and Williams, 2006).

Random Functions with Infinite Basis

We can sample random functions as before.

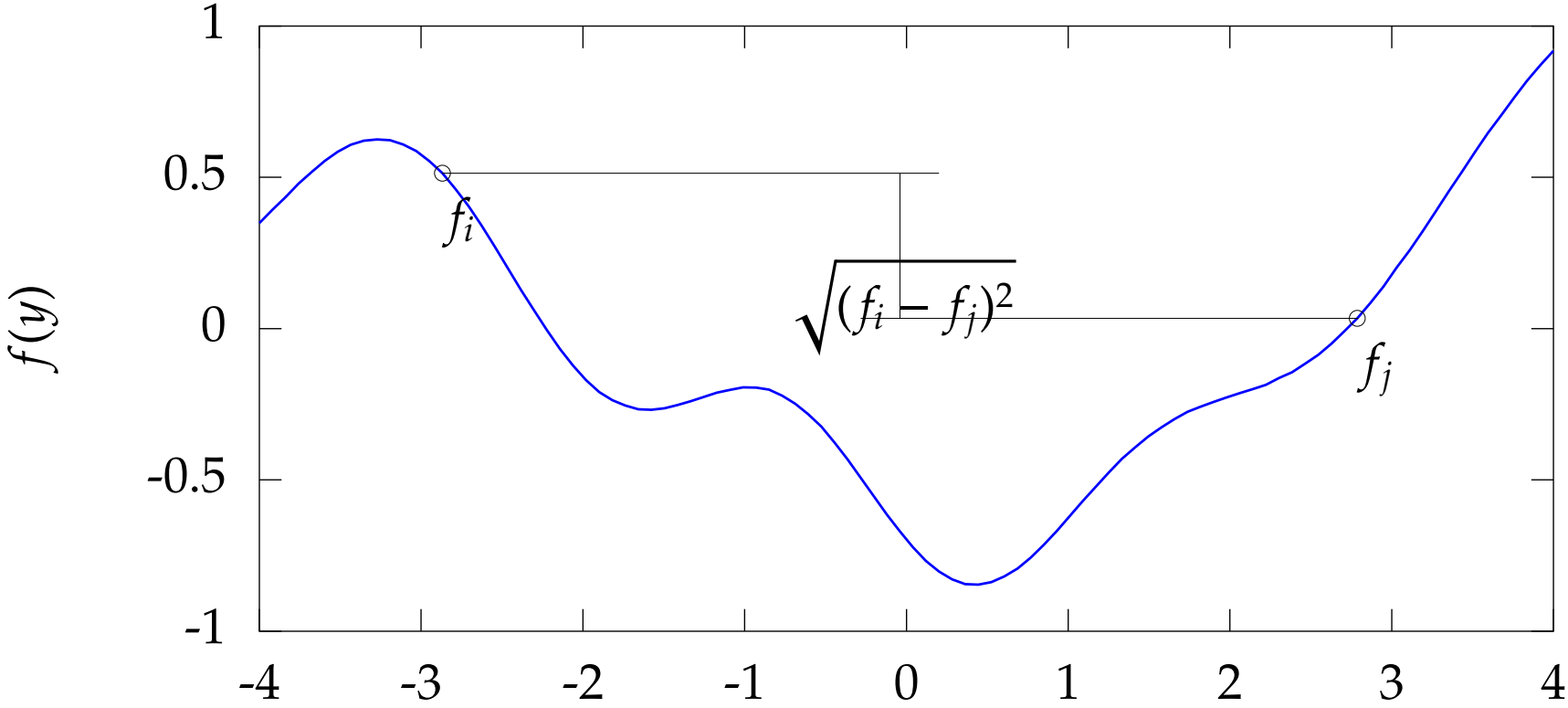


Figure: Distance between two points in the function $f(y)$. The locations are again far apart in y but now we are using an infinite

Random Functions with Infinite Basis

We can sample random functions as before.

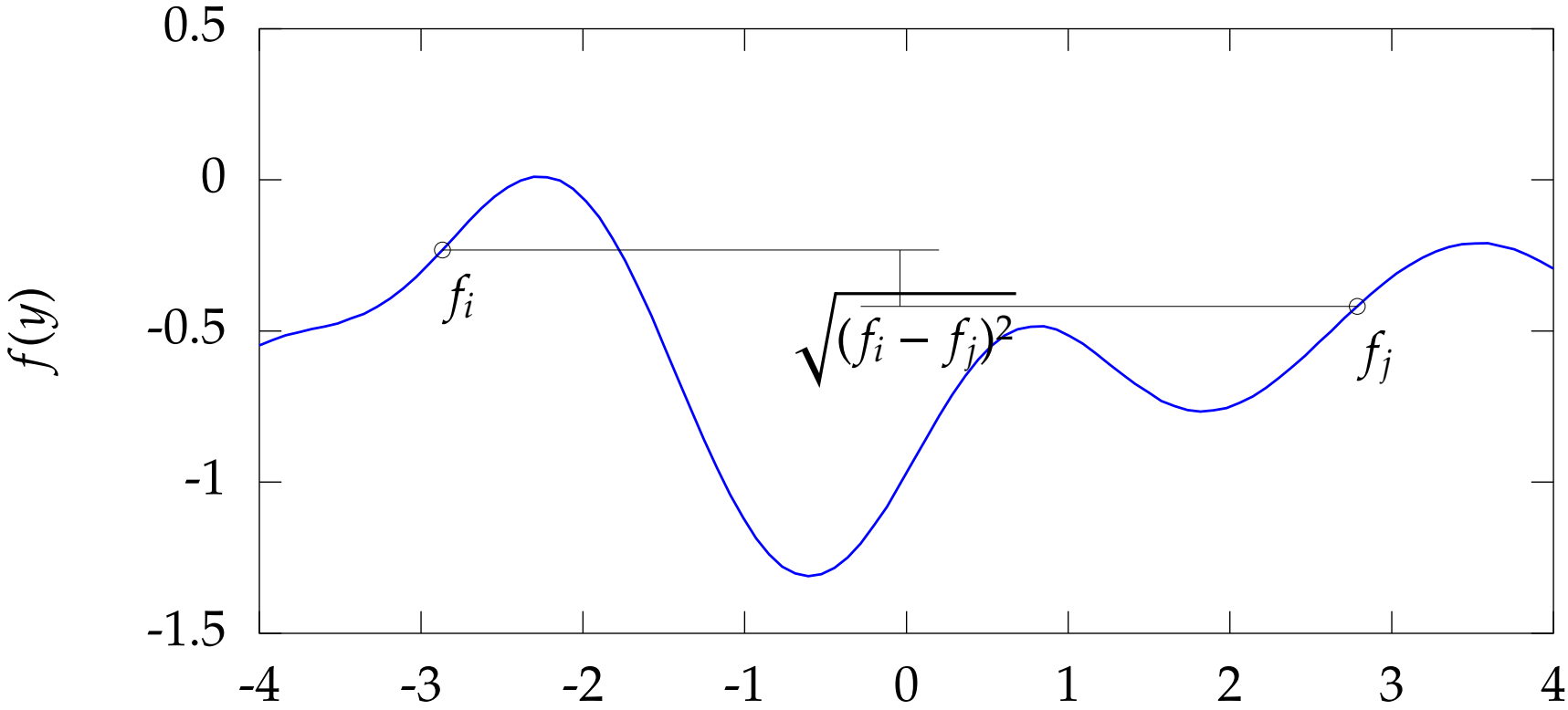


Figure: Distance between two points in the function $f(y)$. The locations are again far apart in y but now we are using an infinite

Random Functions with Infinite Basis

We can sample random functions as before.

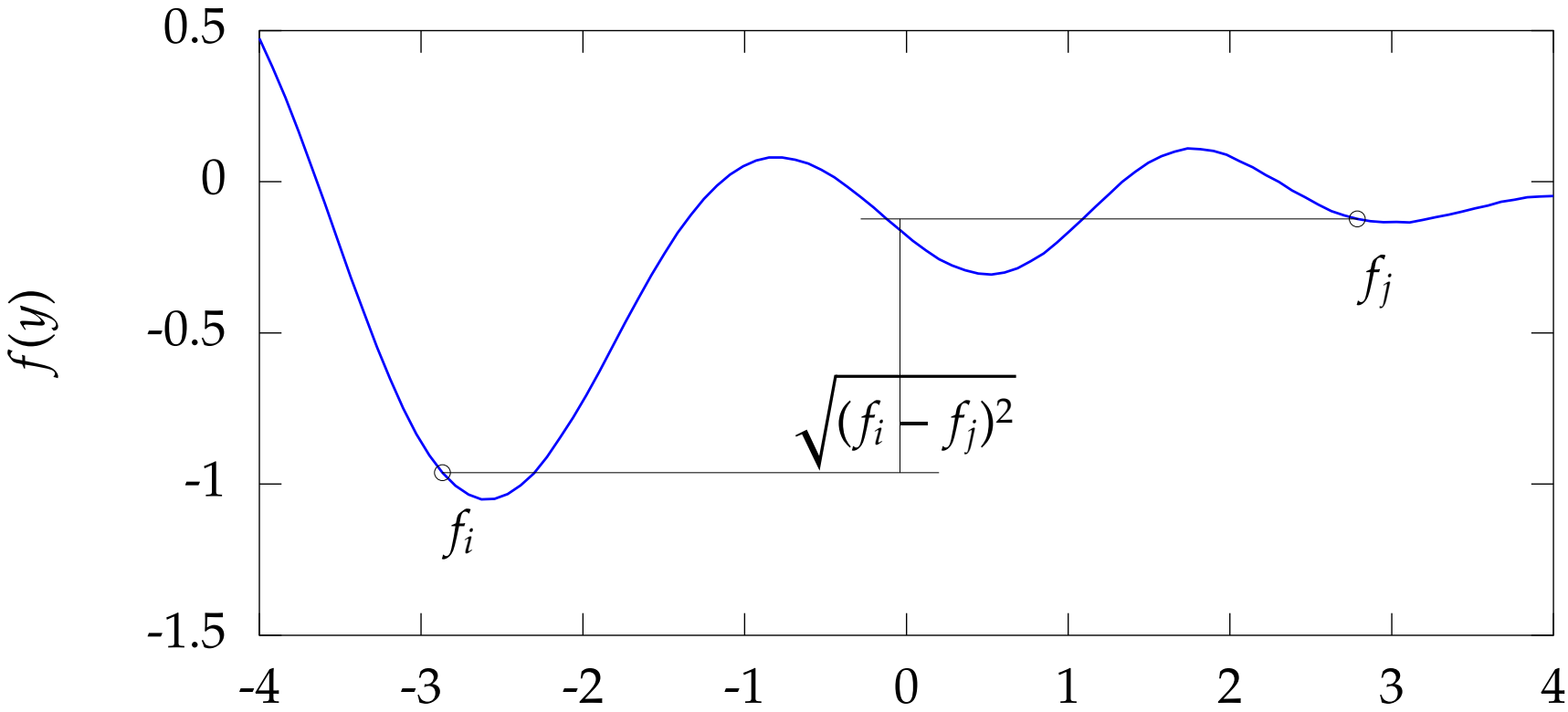


Figure: Distance between two points in the function $f(y)$. The locations are again far apart in y but now we are using an infinite

Random Functions with Infinite Basis

We can sample random functions as before.

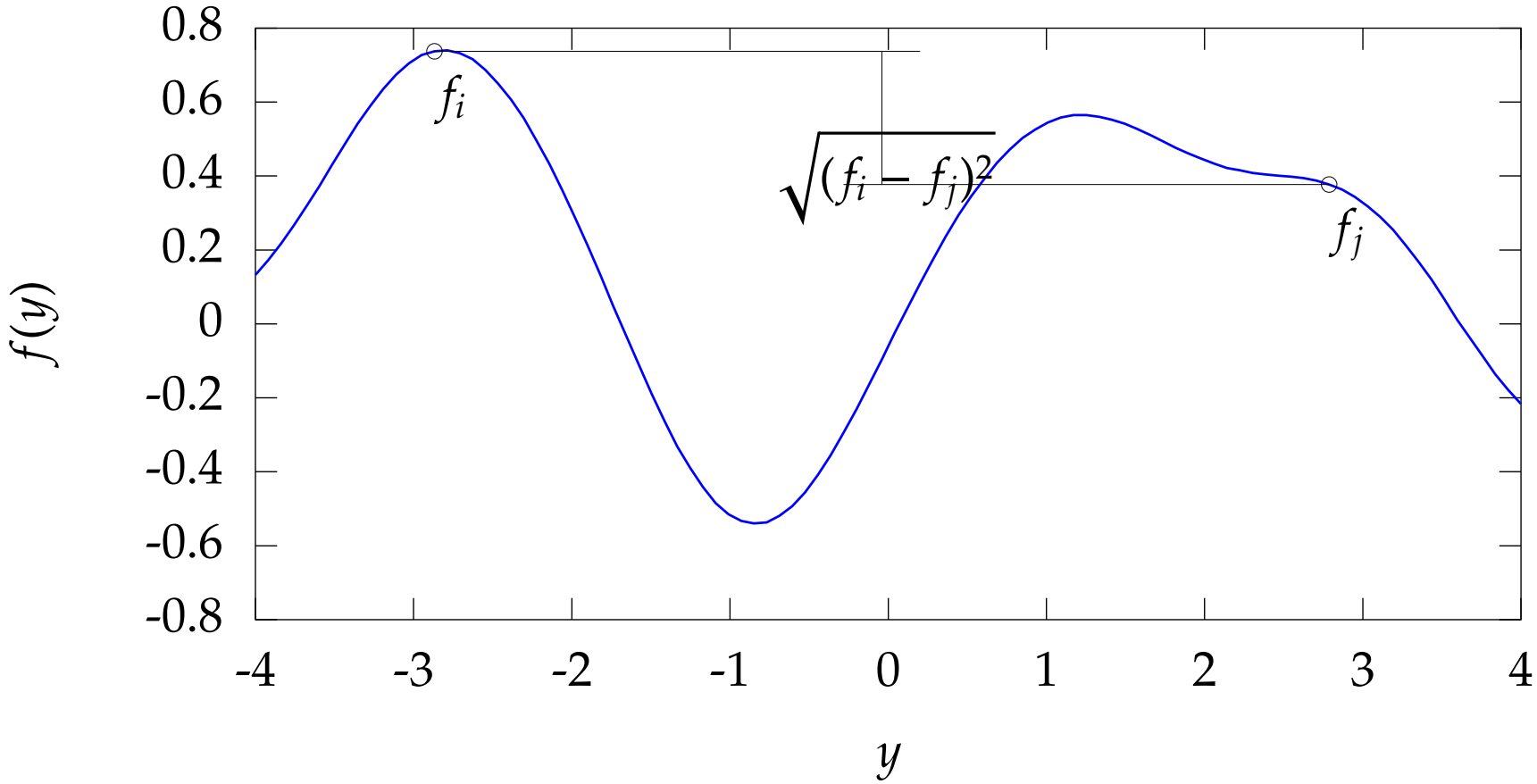


Figure: Distance between two points in the function $f(y)$. The locations are again far apart in y but now we are using an infinite

Similarity and Distance Matrices

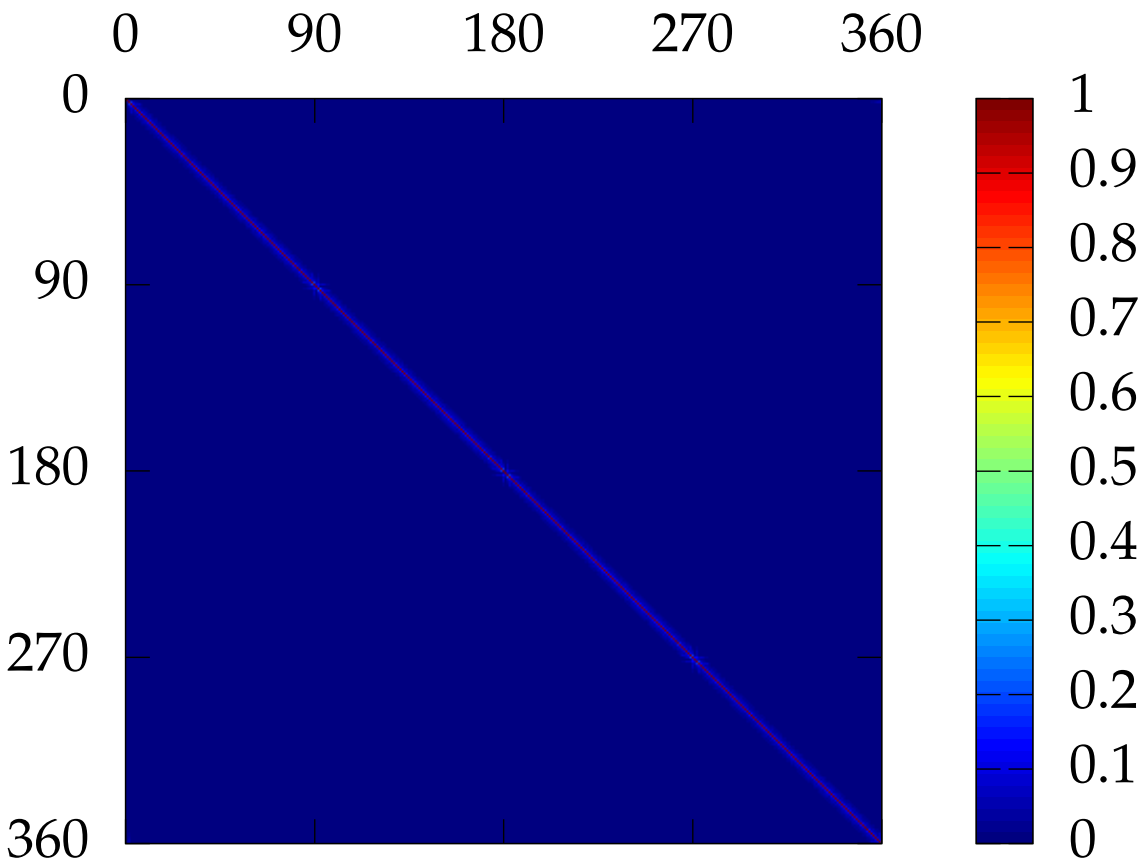


Figure: Similarity matrix for exponentiated quadratic kernel on rotated sixes.

Similarity and Distance Matrices

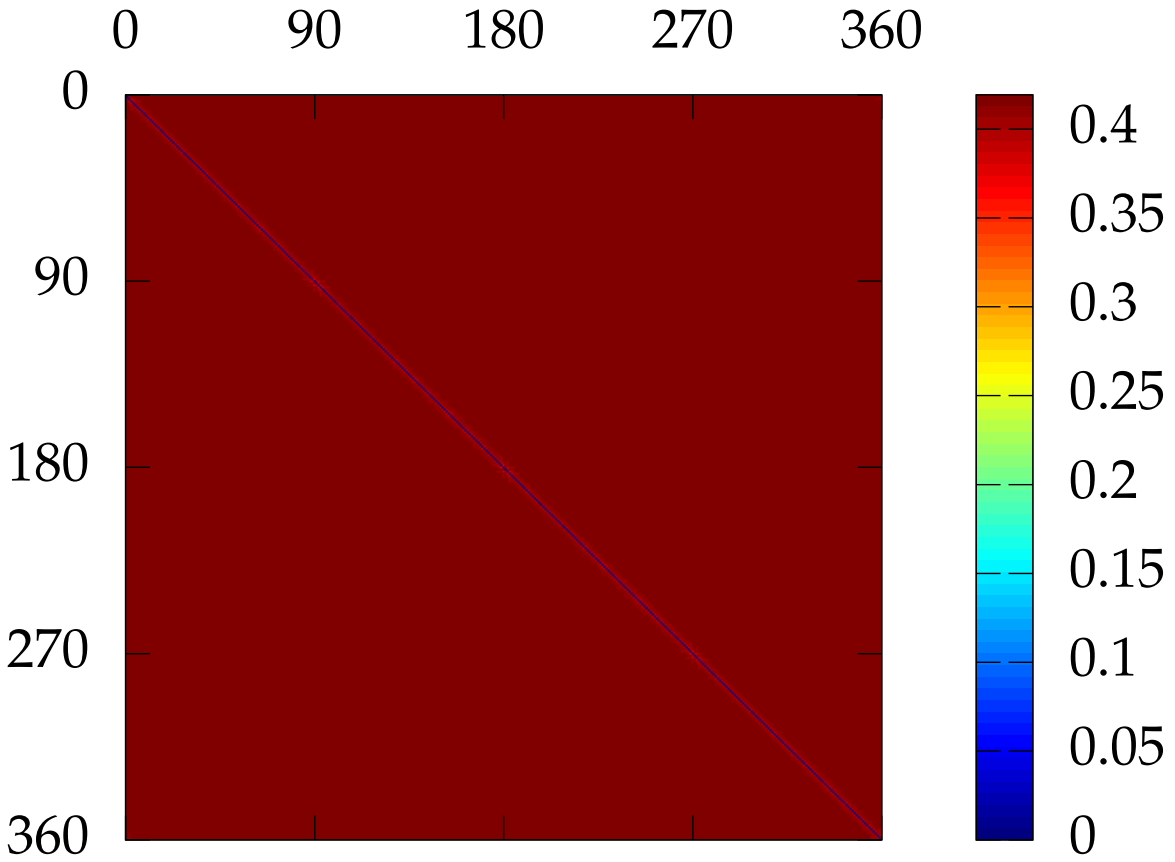


Figure: Implied distance matrix for kernel on rotated sixes. Note that most of the distances are set to $\sqrt{2} \approx 1.41$.

Similarity and Distance Matrices

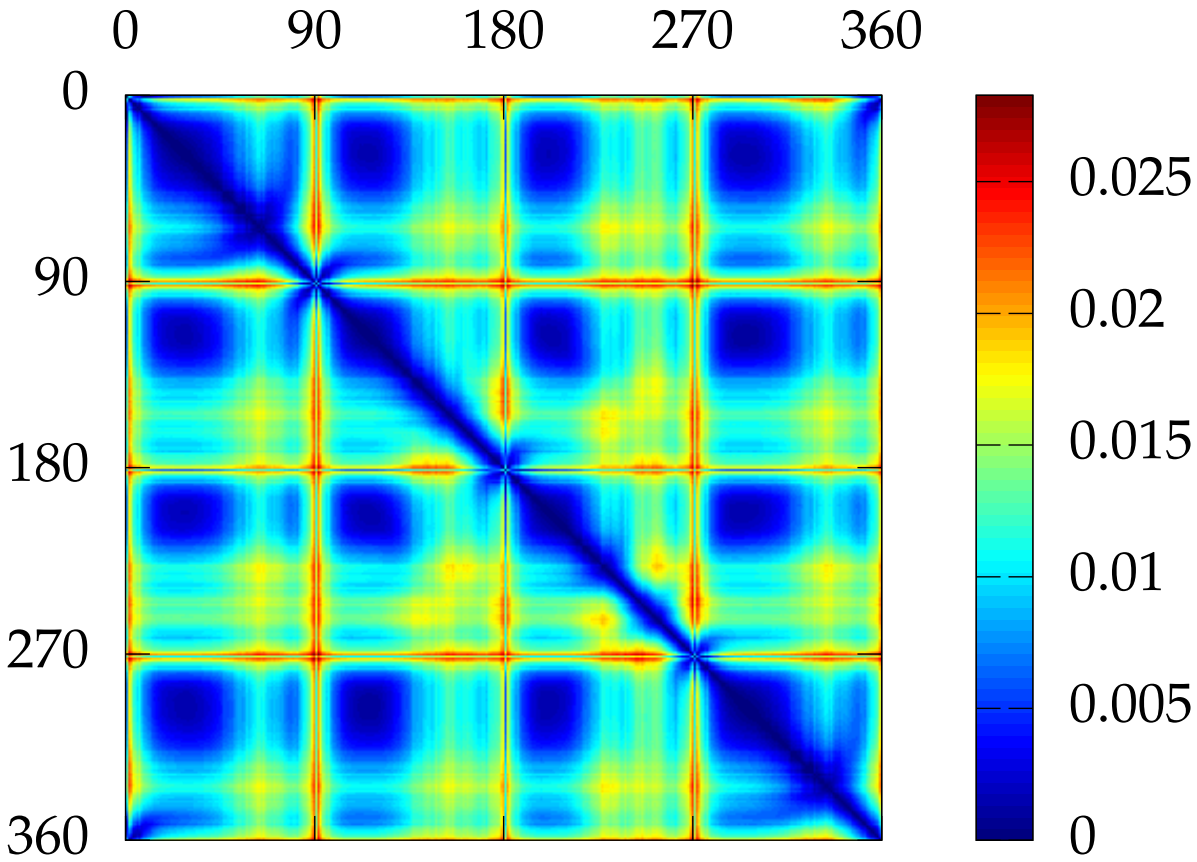
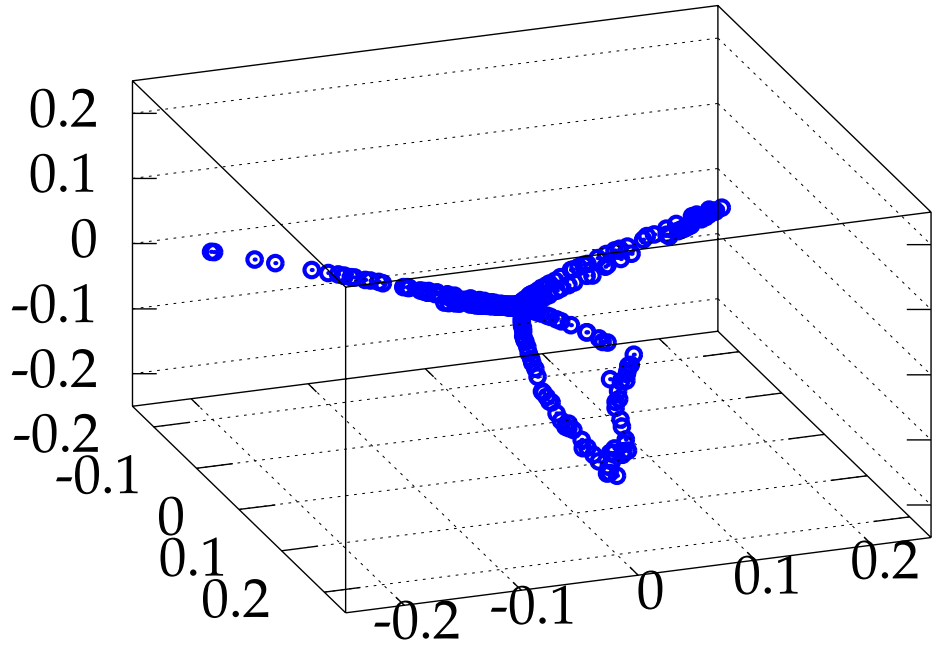
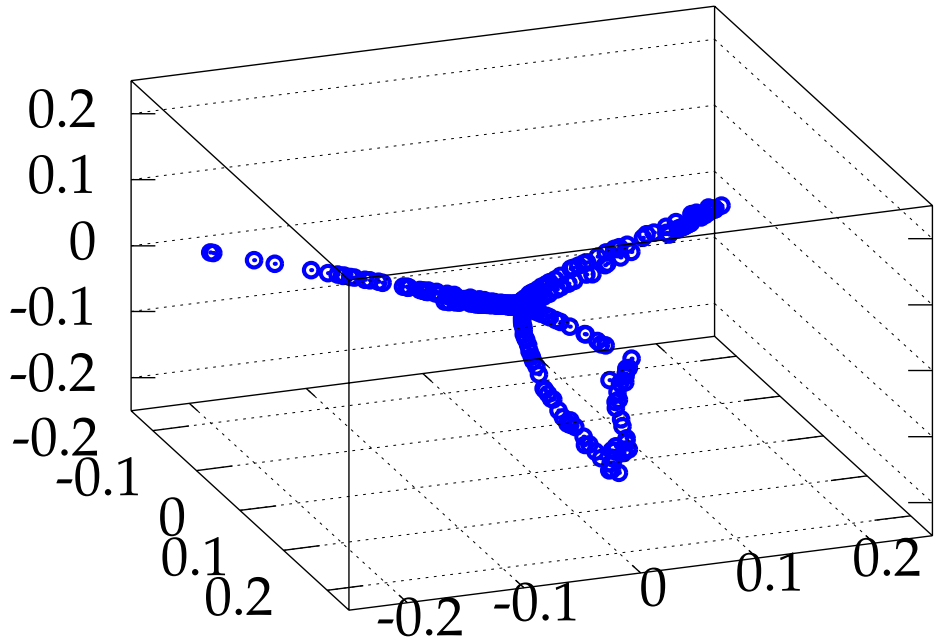


Figure: Implied latent distances for kernel using only $q = 8$ dimensions for latent space.

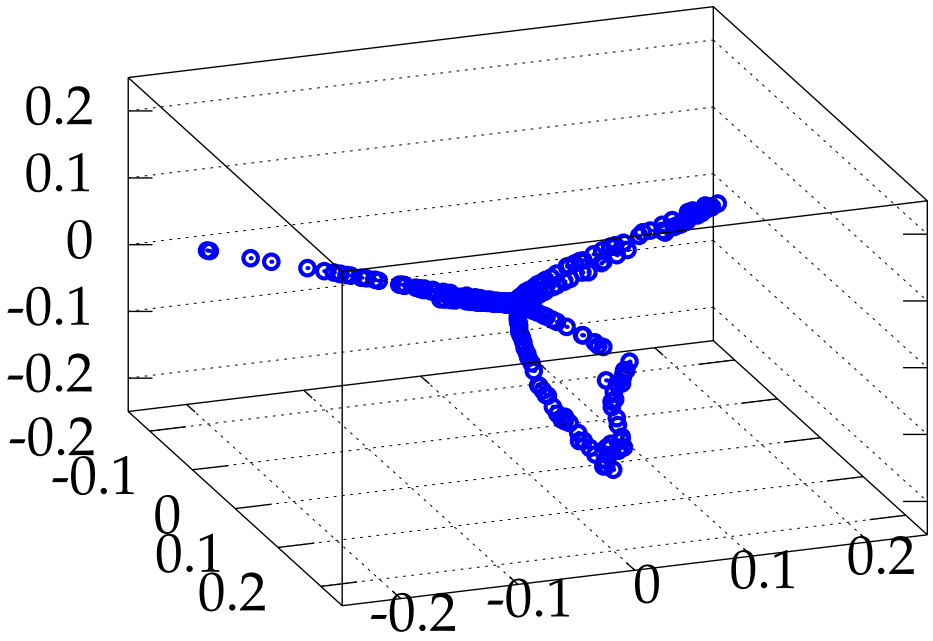
Kernel PCA on Rotated Sixes



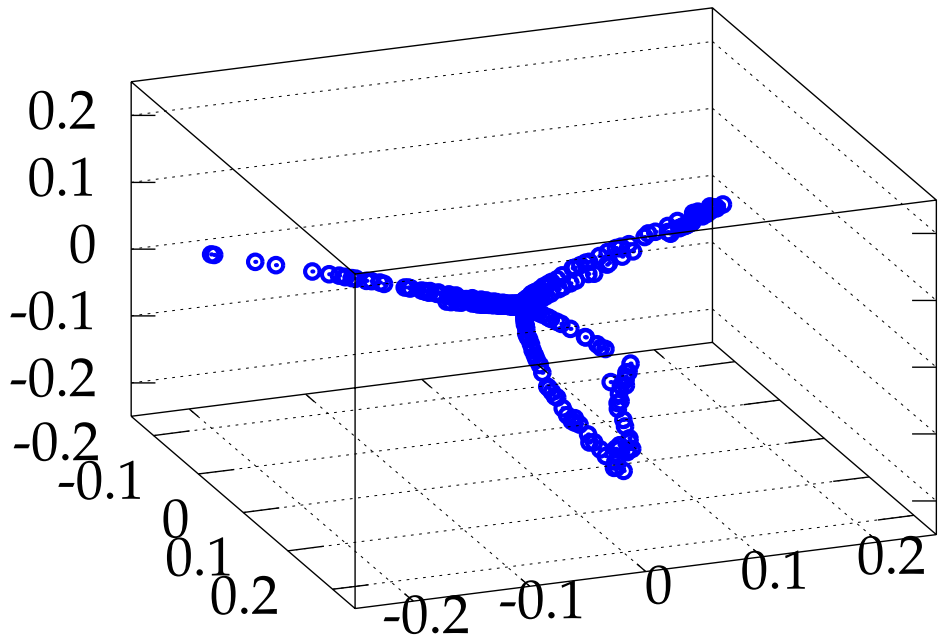
Kernel PCA on Rotated Sixes



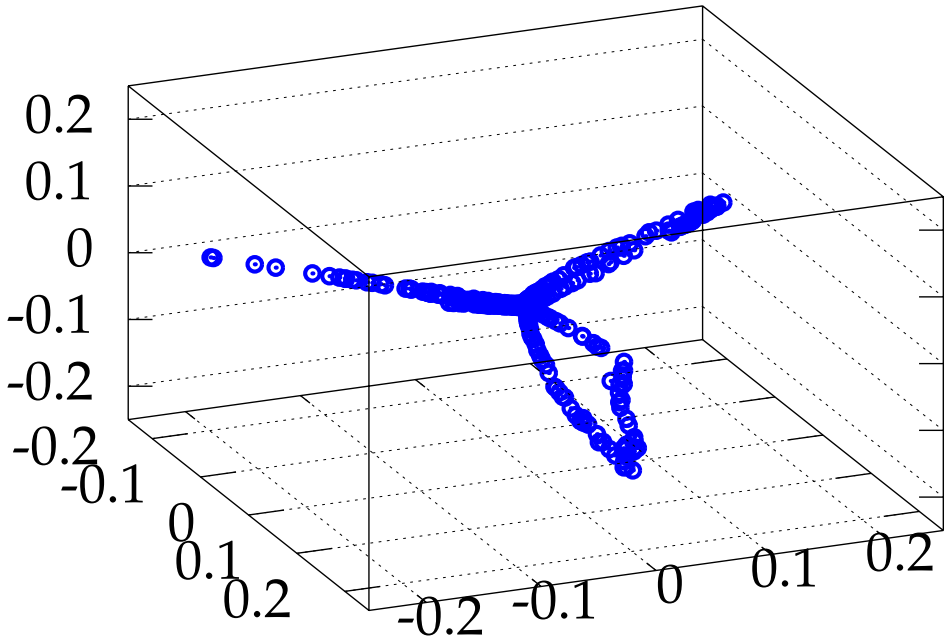
Kernel PCA on Rotated Sixes



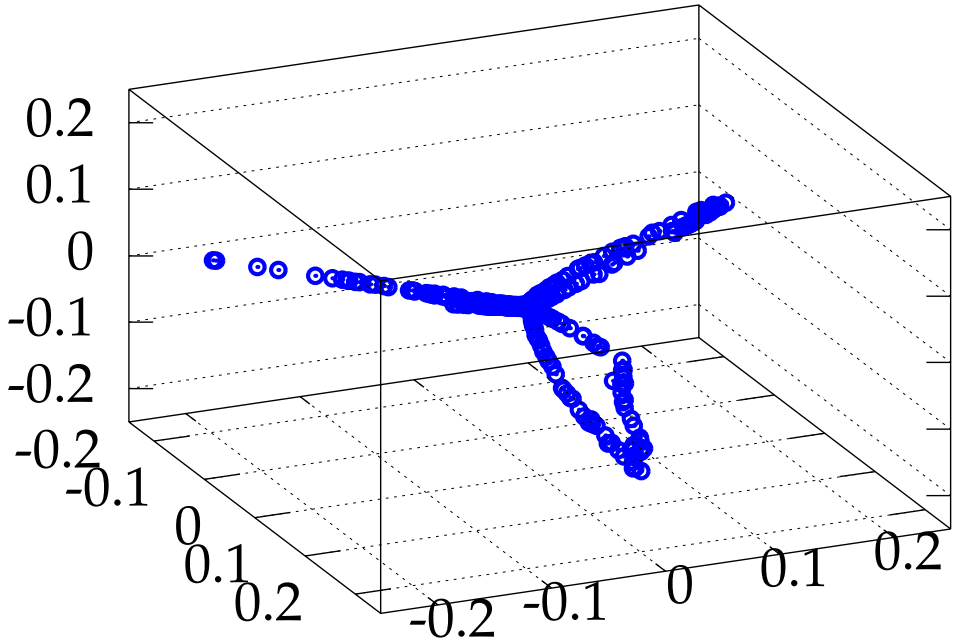
Kernel PCA on Rotated Sixes



Kernel PCA on Rotated Sixes



Kernel PCA on Rotated Sixes



Kernel PCA: A Class of Similarities for Vector Data

- All Mercer kernels are positive semi definite.
- Example, exponentiated quadratic (also known as squared exponential, RBF or Gaussian)

$$k_{i,j} = \exp\left(-\frac{\left\|\mathbf{y}_{i,:} - \mathbf{y}_{j,:}\right\|^2}{2\ell^2}\right).$$

  This leads to a kernel eigenvalue problem.
- This is known as Kernel PCA (Schölkopf et al., 1998).

Implied Distance Matrix

- What is the equivalent distance $\sqrt{d_{i,j}}$?

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

- If point separation is large, $k_{i,j} \to 0$. $k_{i,i} = 1$ and $k_{j,j} = 1$.

$$\sqrt{d_{i,j}} = \sqrt{2}$$

- Kernel with RBF kernel projects along axes PCA can produce poor results.
- Uses many dimensions to keep dissimilar objects a constant amount apart.

Outline

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 $\mathbf{D}$.

Isomap

- MDS finds geometric configuration preserving distances.
- MDS applied to distance along manifold.
- Geodesic Distance ≡ Manifold Distance.
- Cannot compute geodesic distance without knowing manifold.
- Idea: compute distance via shortest path between point-pairs (Tenenbaum et al., 2000).
- Very similar to the road example: data points are cities, graph is roads.

Isomap

- Isomap: define neighbors and compute distances between neighbors.
- Geodesic distance approximated by shortest path through adjacency matrix.

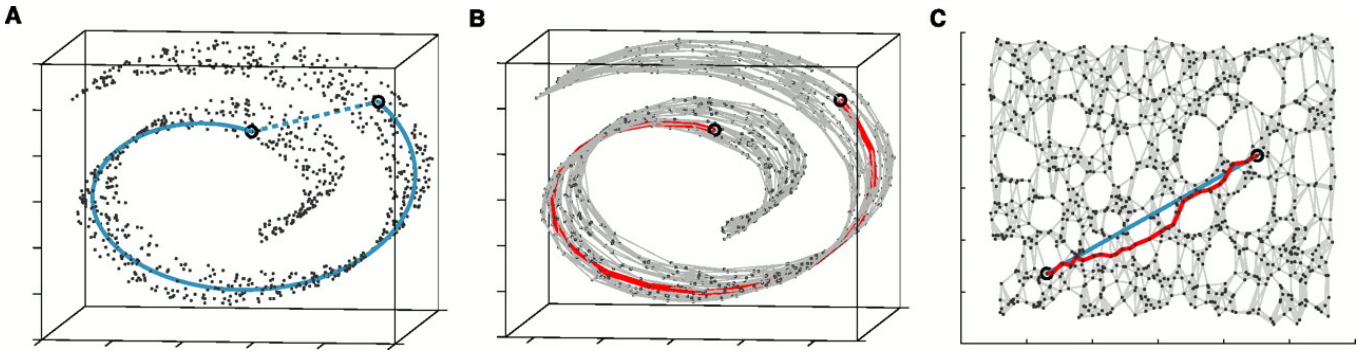


Figure: A: true geodesic distance. B: Approximate distance on graph. C: comparison of true and approximate distances. (Image from Tenenbaum et al., 2000).

Isomap Neighborhood

- Compute nearest k neighbors for each point.
- Construct a graph linking data points through neighbors.

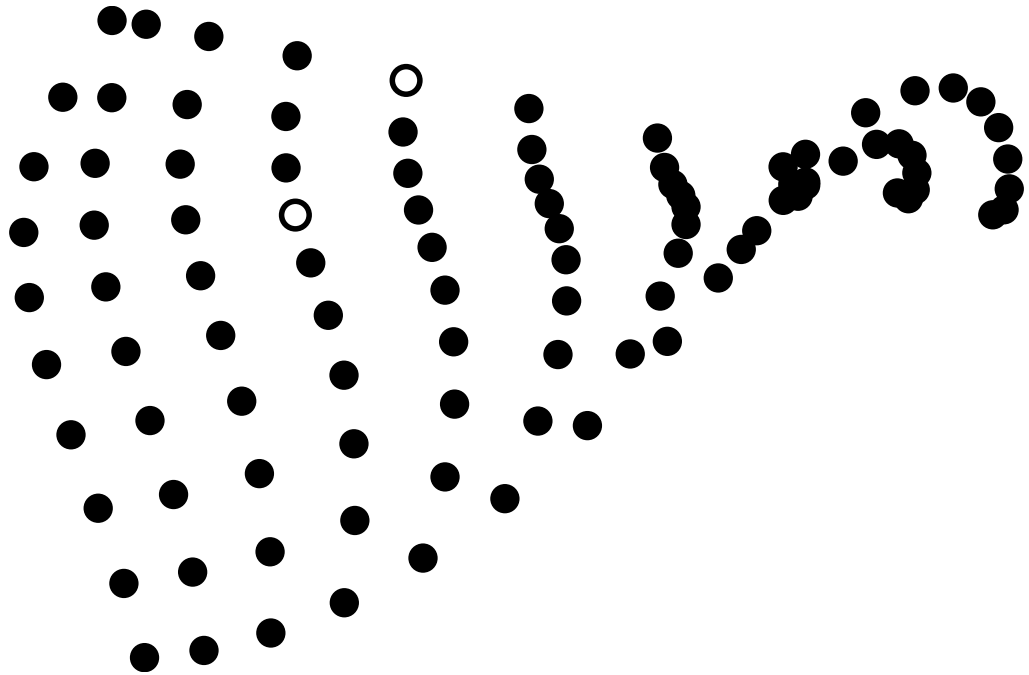


Figure: Distance on graph is a proxy for geodesic distance.

Isomap Neighborhood

- Compute nearest k neighbors for each point.
- Construct a graph linking data points through neighbors.

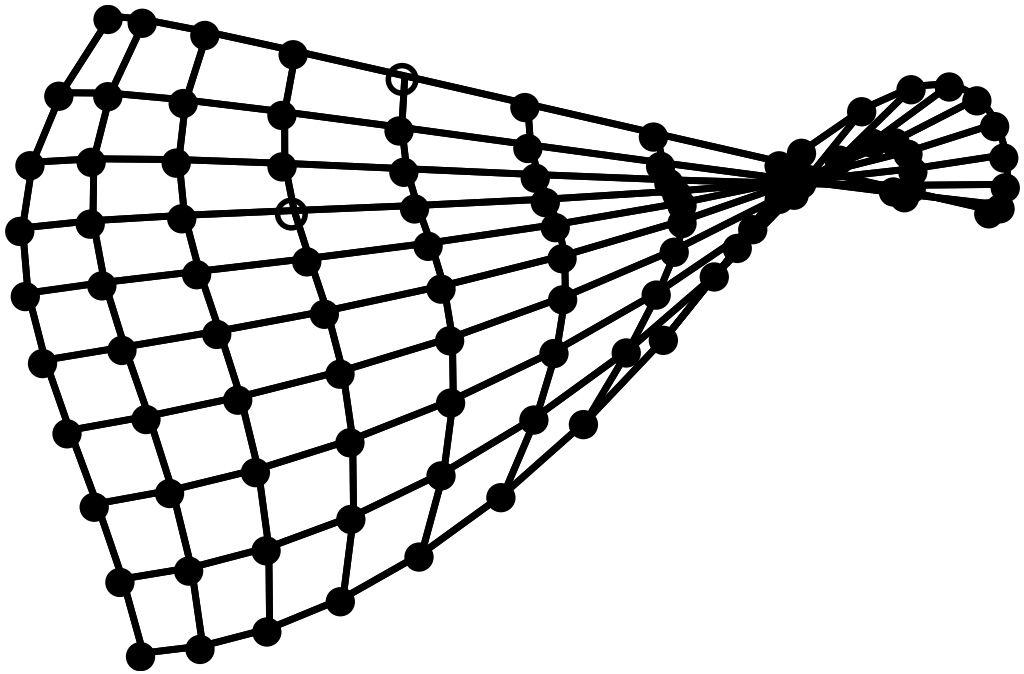


Figure: Distance on graph is a proxy for geodesic distance.

Isomap Neighborhood

- Compute nearest k neighbors for each point.
- Construct a graph linking data points through neighbors.

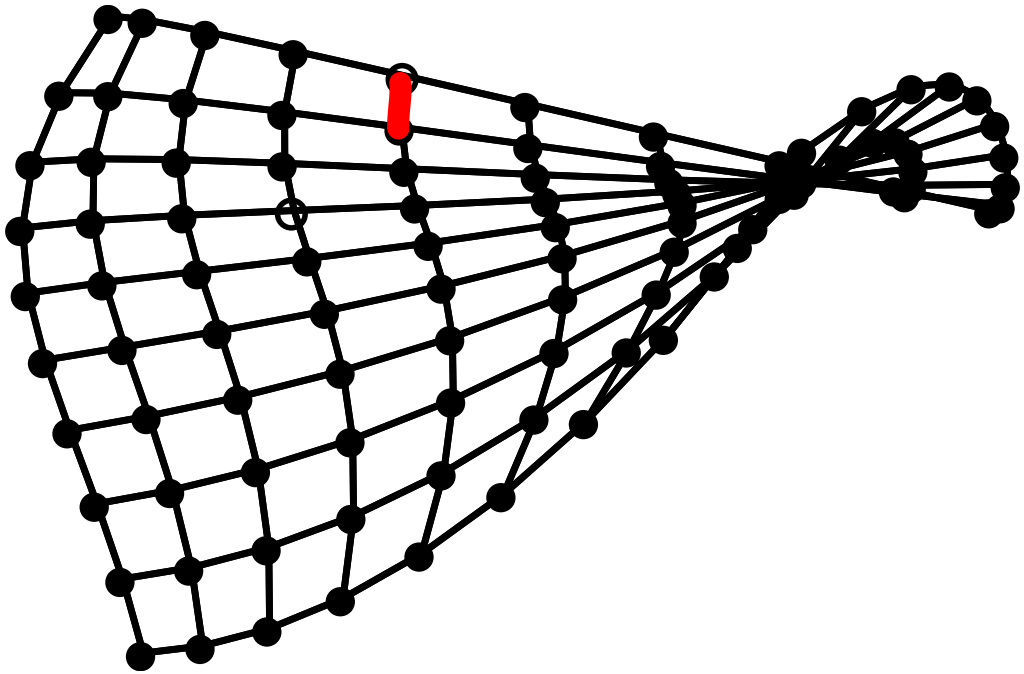


Figure: Distance on graph is a proxy for geodesic distance.

Isomap Neighborhood

- Compute nearest k neighbors for each point.
- Construct a graph linking data points through neighbors.

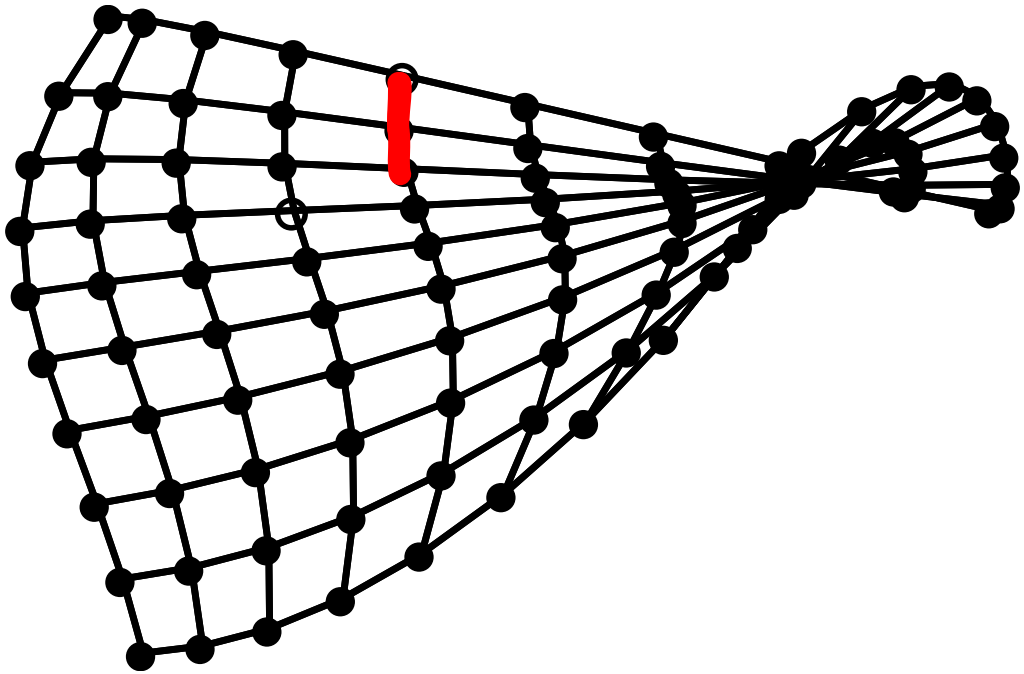


Figure: Distance on graph is a proxy for geodesic distance.

Isomap Neighborhood

- Compute nearest k neighbors for each point.
- Construct a graph linking data points through neighbors.

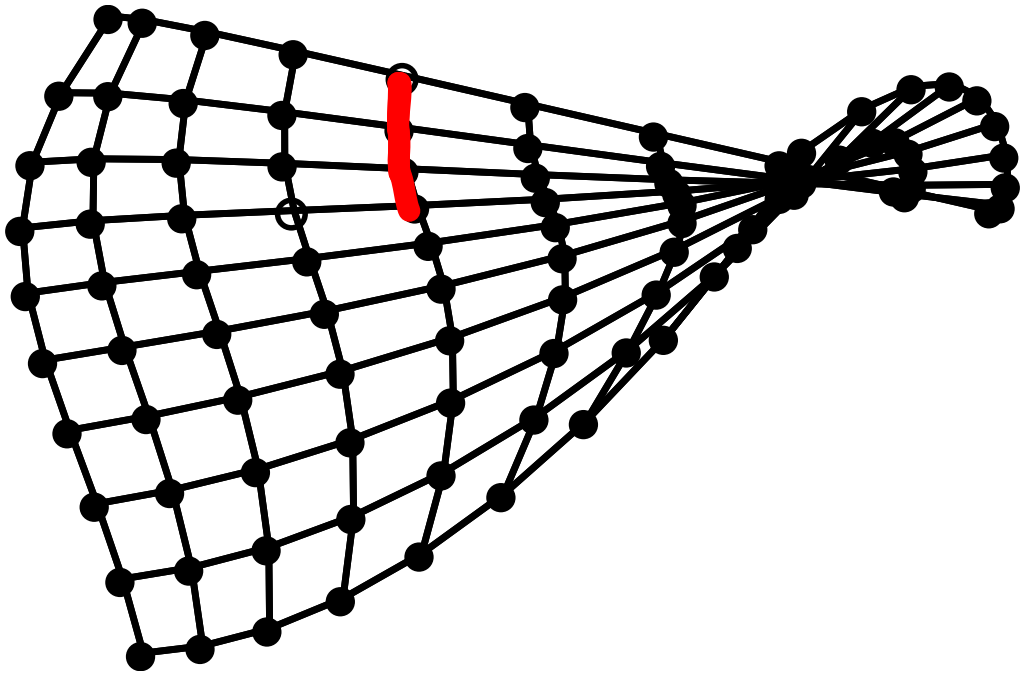


Figure: Distance on graph is a proxy for geodesic distance.

Isomap Neighborhood

- Compute nearest k neighbors for each point.
- Construct a graph linking data points through neighbors.

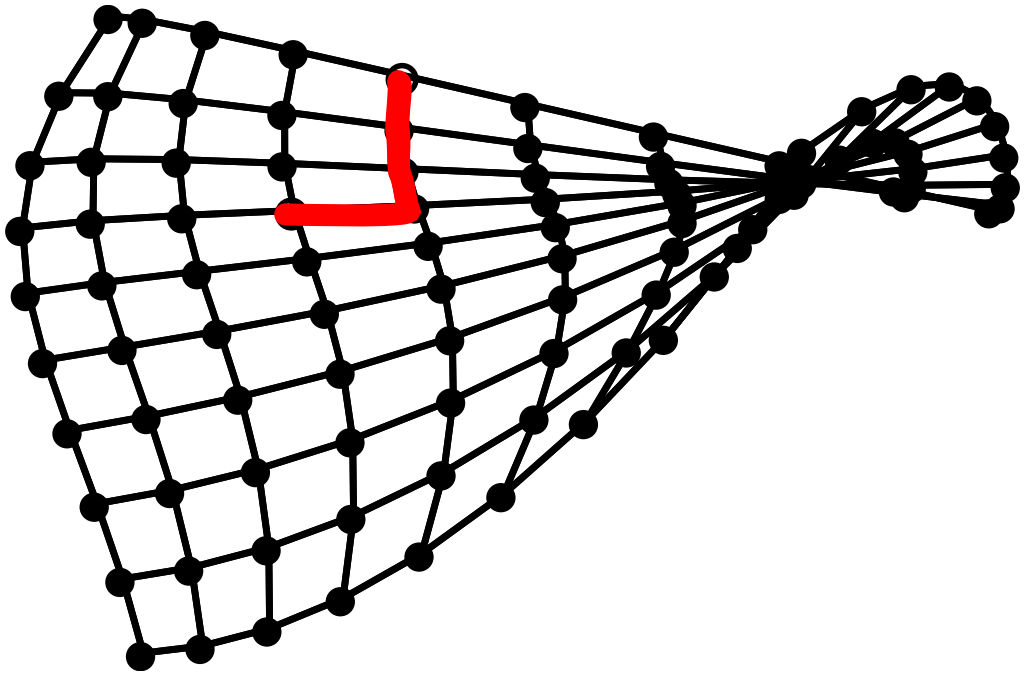


Figure: Distance on graph is a proxy for geodesic distance.

Data Neighborhood

- Need to determine correct number of neighbors.
- Manifold distortions mean neighbors in latent space may not be neighbors in data space.

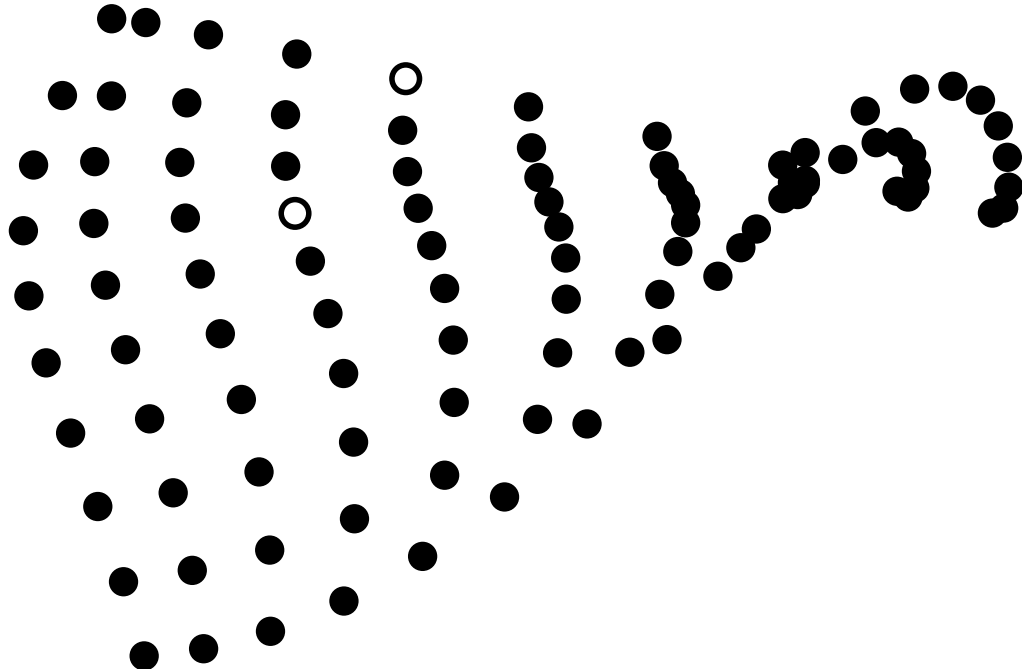


Figure: Quality of approximation depends on quality of graph.

Data Neighborhood

- Need to determine correct number of neighbors.
- Manifold distortions mean neighbors in latent space may not be neighbors in data space.

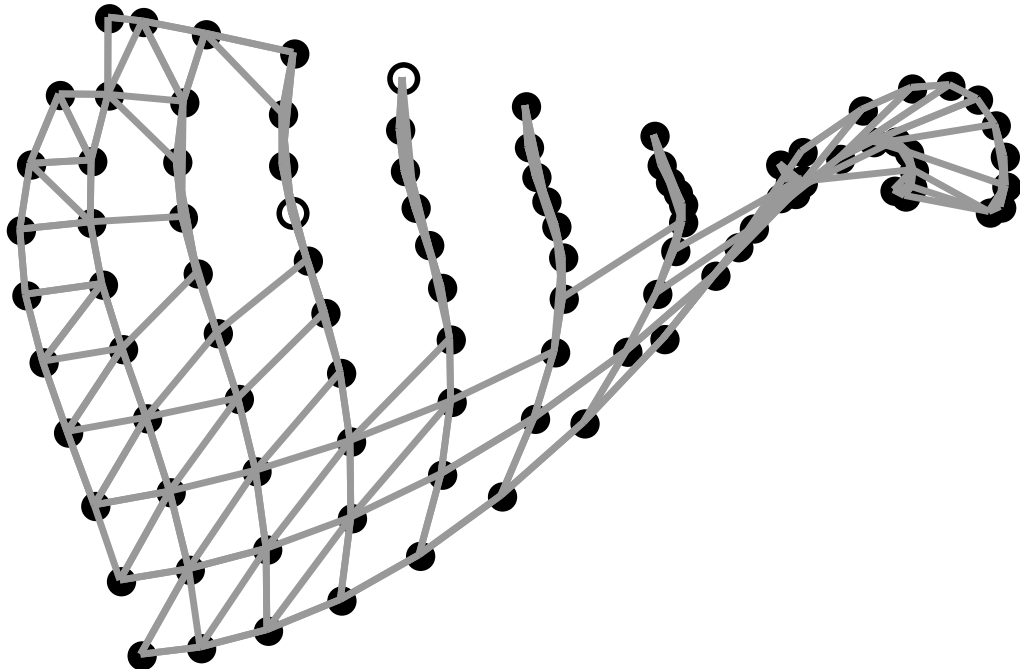


Figure: Quality of approximation depends on quality of graph.

Data Neighborhood

- Need to determine correct number of neighbors.
- Manifold distortions mean neighbors in latent space may not be neighbors in data space.

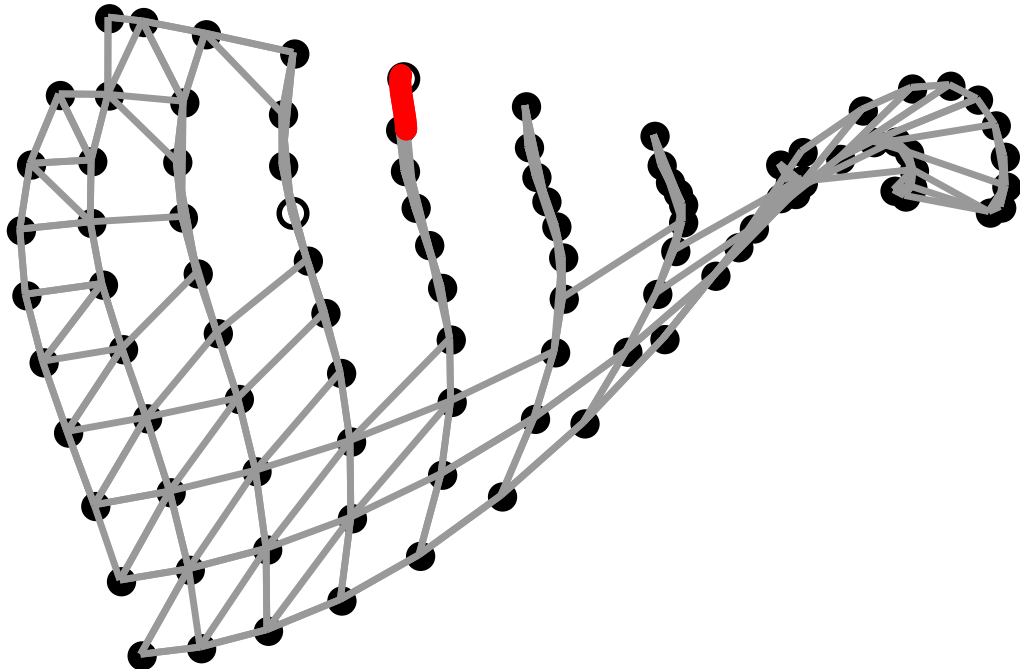


Figure: Quality of approximation depends on quality of graph.

Data Neighborhood

- Need to determine correct number of neighbors.
- Manifold distortions mean neighbors in latent space may not be neighbors in data space.

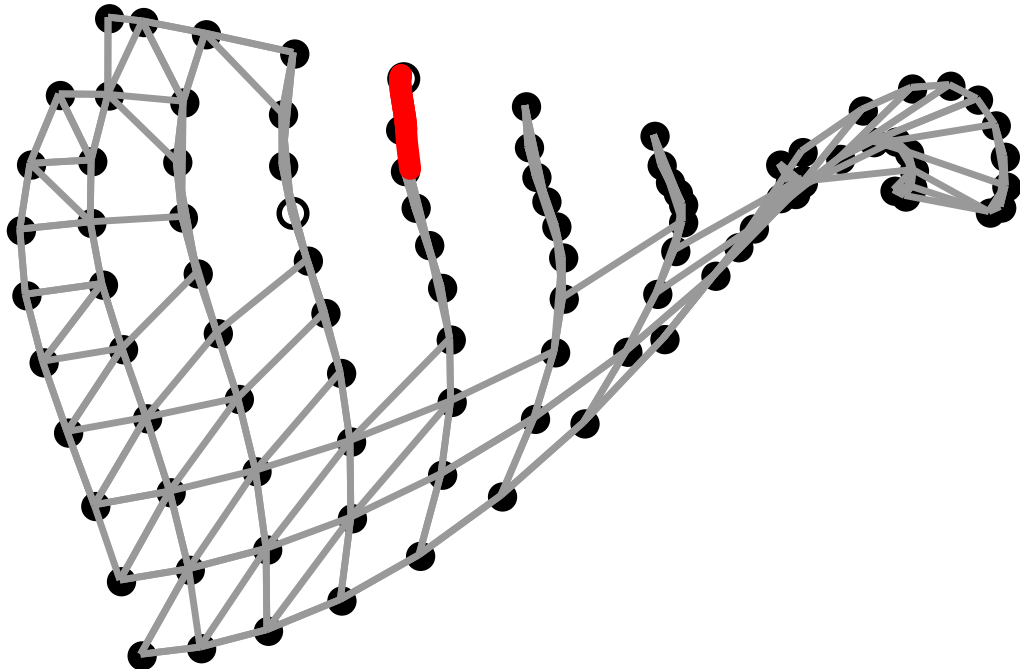


Figure: Quality of approximation depends on quality of graph.

Data Neighborhood

- Need to determine correct number of neighbors.
- Manifold distortions mean neighbors in latent space may not be neighbors in data space.

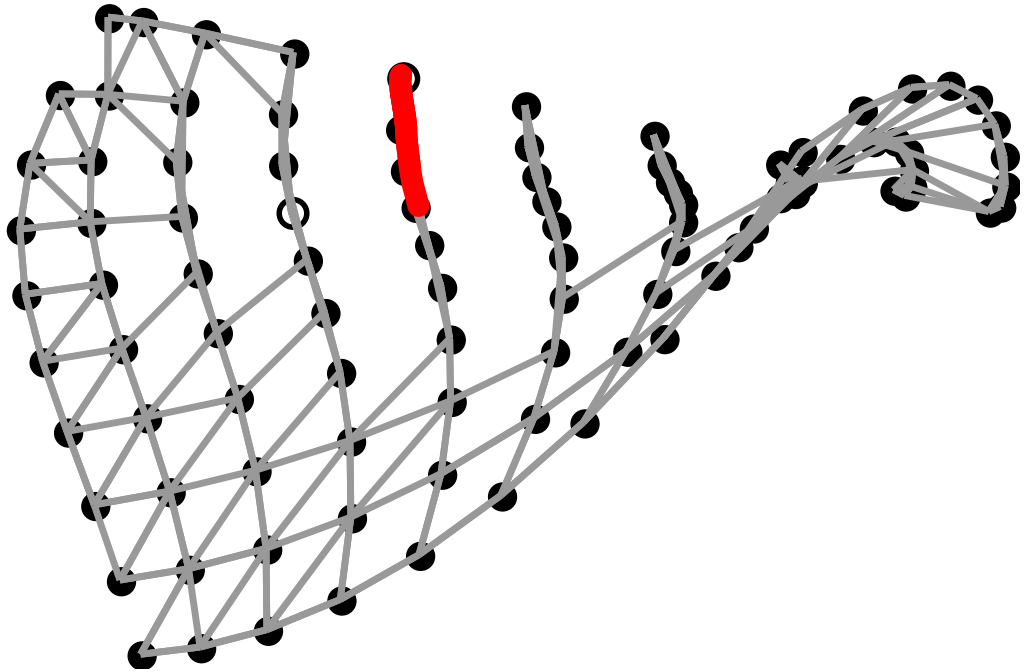


Figure: Quality of approximation depends on quality of graph.

Data Neighborhood

- Need to determine correct number of neighbors.
- Manifold distortions mean neighbors in latent space may not be neighbors in data space.

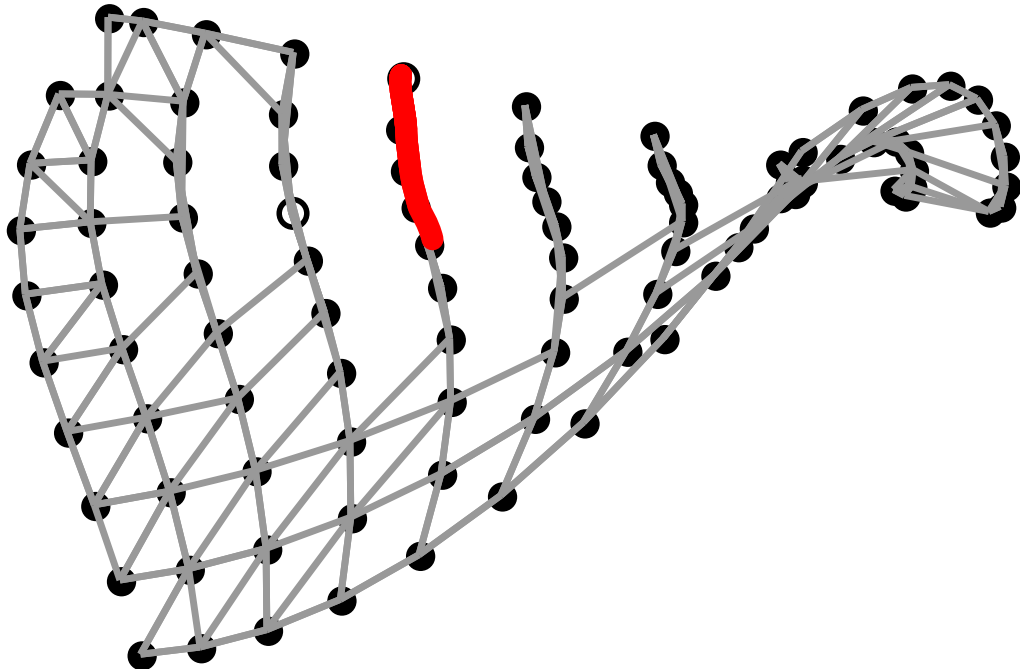


Figure: Quality of approximation depends on quality of graph.

Data Neighborhood

- Need to determine correct number of neighbors.
- Manifold distortions mean neighbors in latent space may not be neighbors in data space.

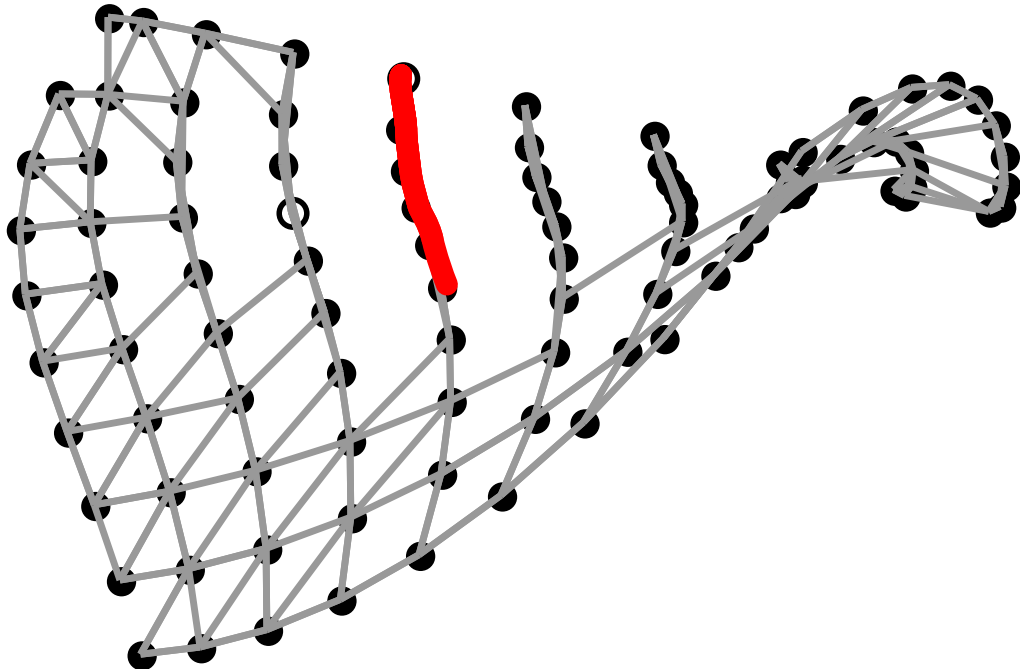


Figure: Quality of approximation depends on quality of graph.

Data Neighborhood

- Need to determine correct number of neighbors.
- Manifold distortions mean neighbors in latent space may not be neighbors in data space.

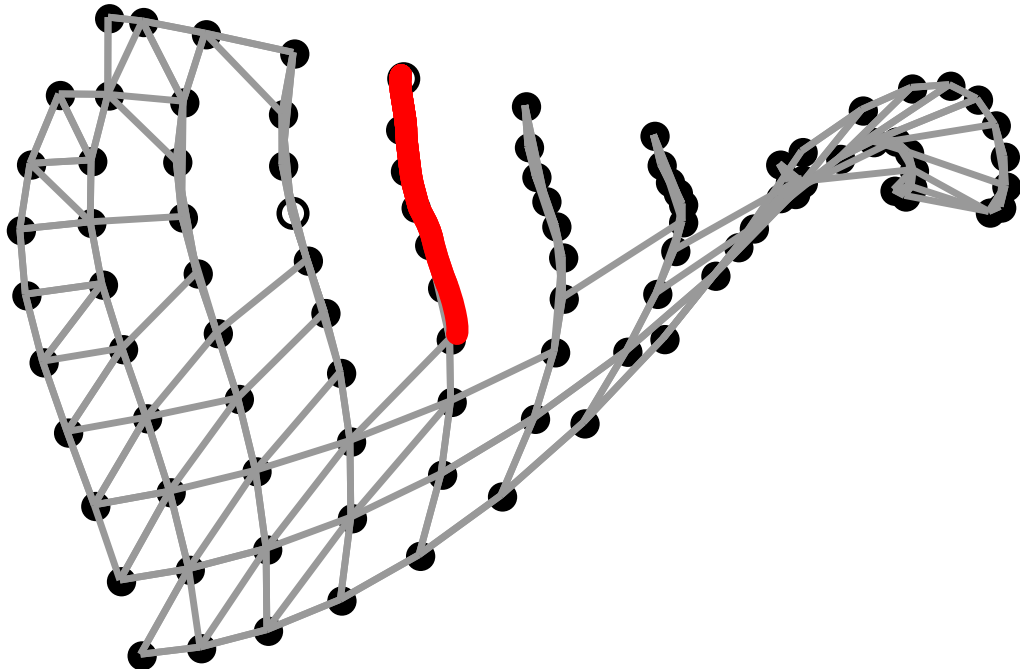


Figure: Quality of approximation depends on quality of graph.

Data Neighborhood

- Need to determine correct number of neighbors.
- Manifold distortions mean neighbors in latent space may not be neighbors in data space.

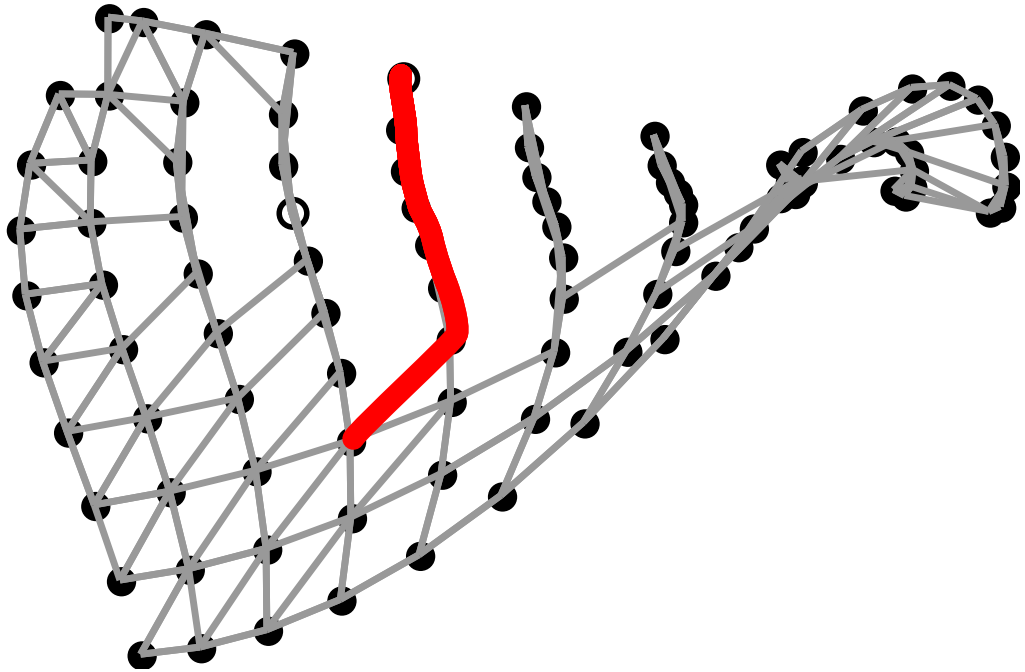


Figure: Quality of approximation depends on quality of graph.

Data Neighborhood

- Need to determine correct number of neighbors.
- Manifold distortions mean neighbors in latent space may not be neighbors in data space.

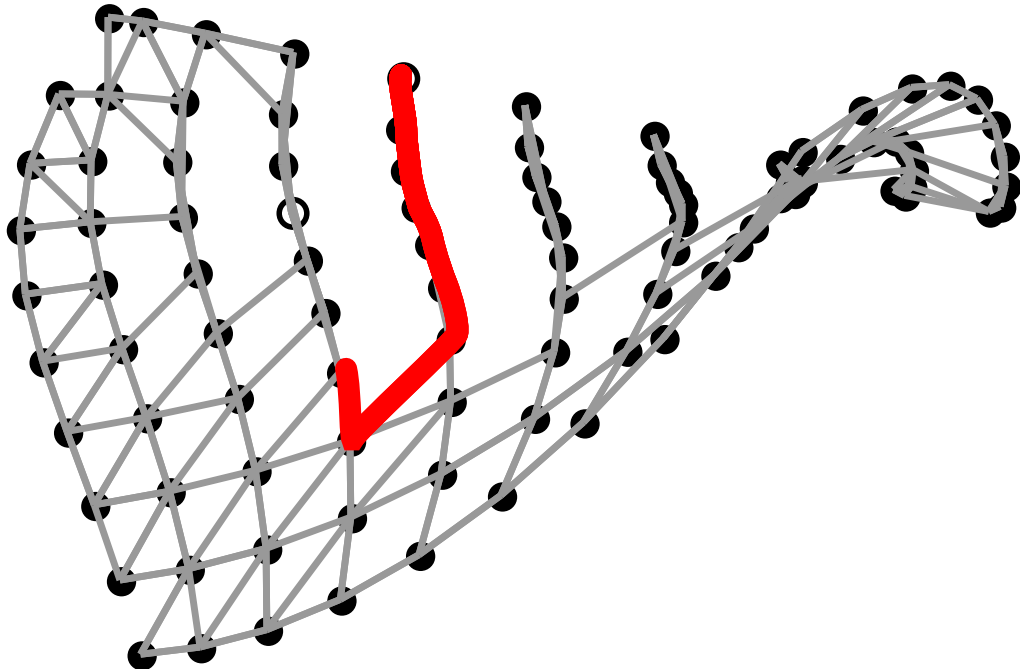


Figure: Quality of approximation depends on quality of graph.

Data Neighborhood

- Need to determine correct number of neighbors.
- Manifold distortions mean neighbors in latent space may not be neighbors in data space.

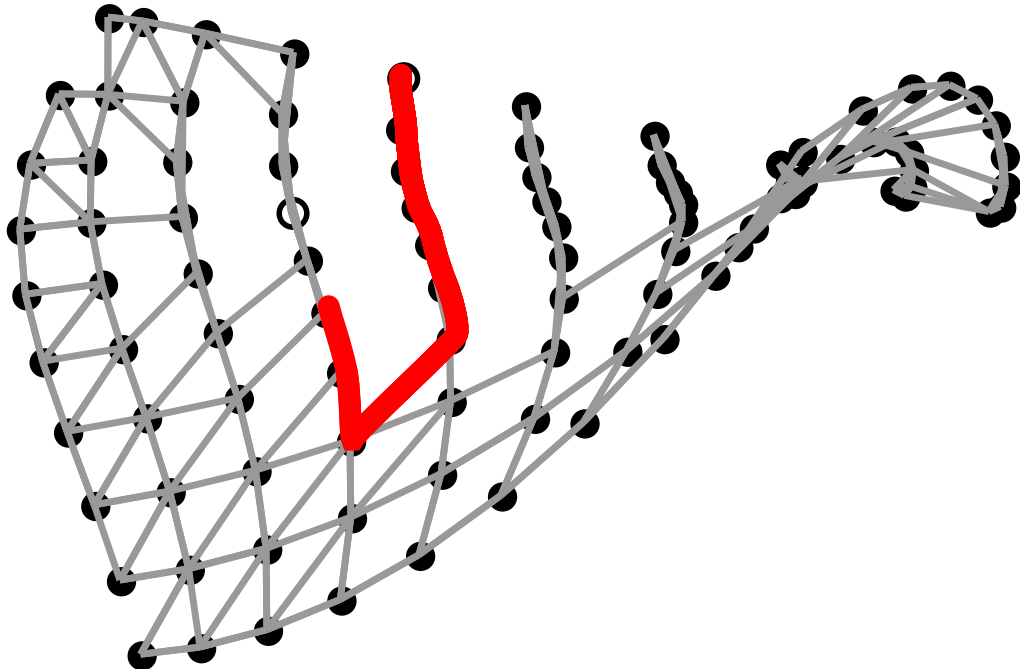


Figure: Quality of approximation depends on quality of graph.

Data Neighborhood

- Need to determine correct number of neighbors.
- Manifold distortions mean neighbors in latent space may not be neighbors in data space.

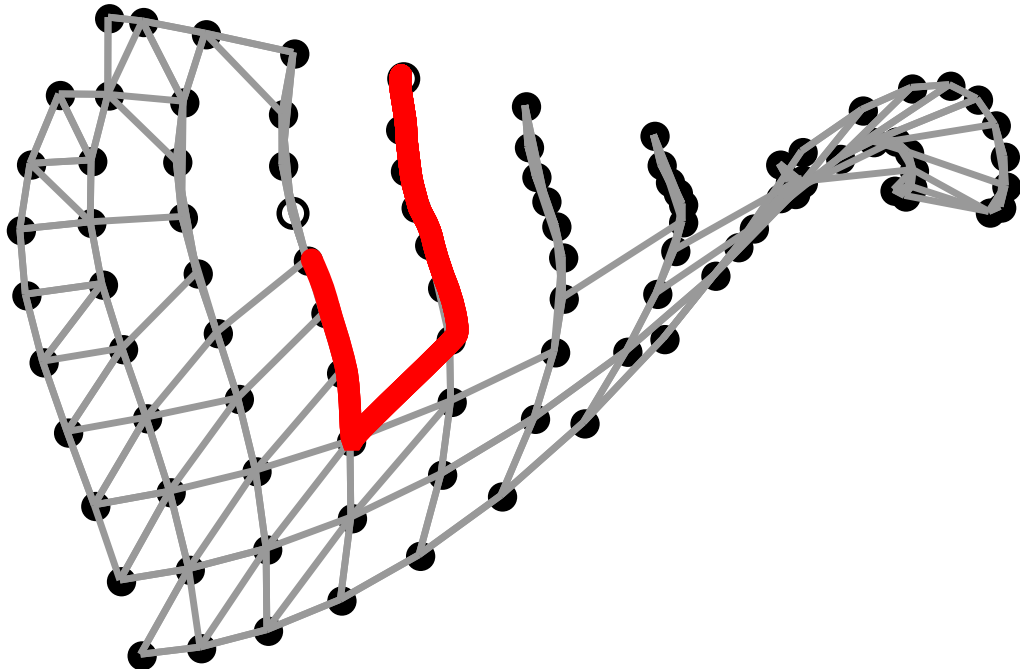


Figure: Quality of approximation depends on quality of graph.

Data Neighborhood

- Need to determine correct number of neighbors.
- Manifold distortions mean neighbors in latent space may not be neighbors in data space.

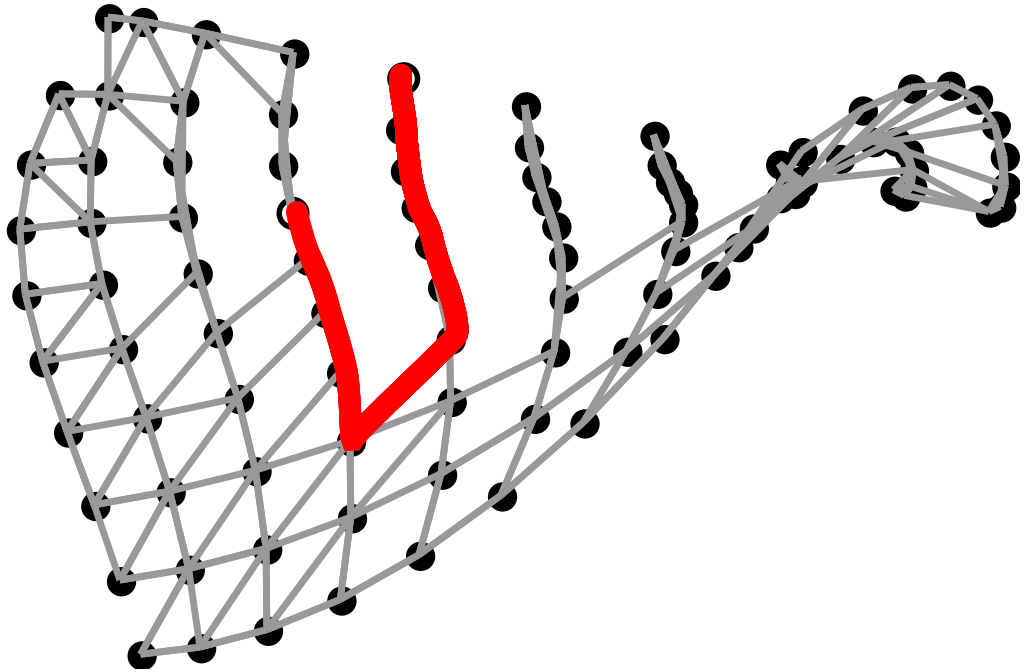


Figure: Quality of approximation depends on quality of graph.

Isomap Algorithm

- Build a neighborhood graph between data points.
- Set each edge in graph to a value given by interpoint distance (not squared!).
- Build a matrix of interpoint distances based on shortest distances in this graph.
- Perform CMDS on this graph.

Isomap on Stick Man

- Two components of stick man data.

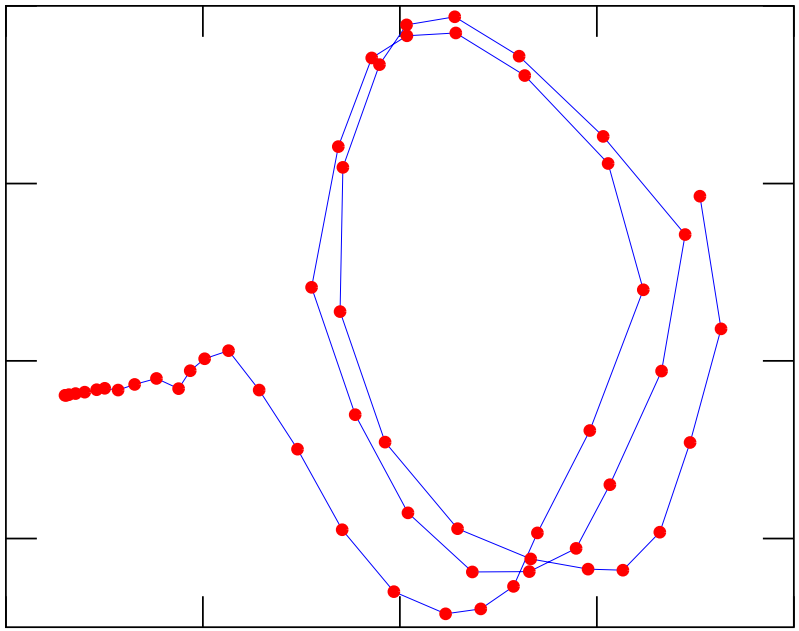


Figure: Stick man data embedded using two dimensions of isomap. `demStickIsomap1`.

Isomap on Oil Data

- Two components of oil data.

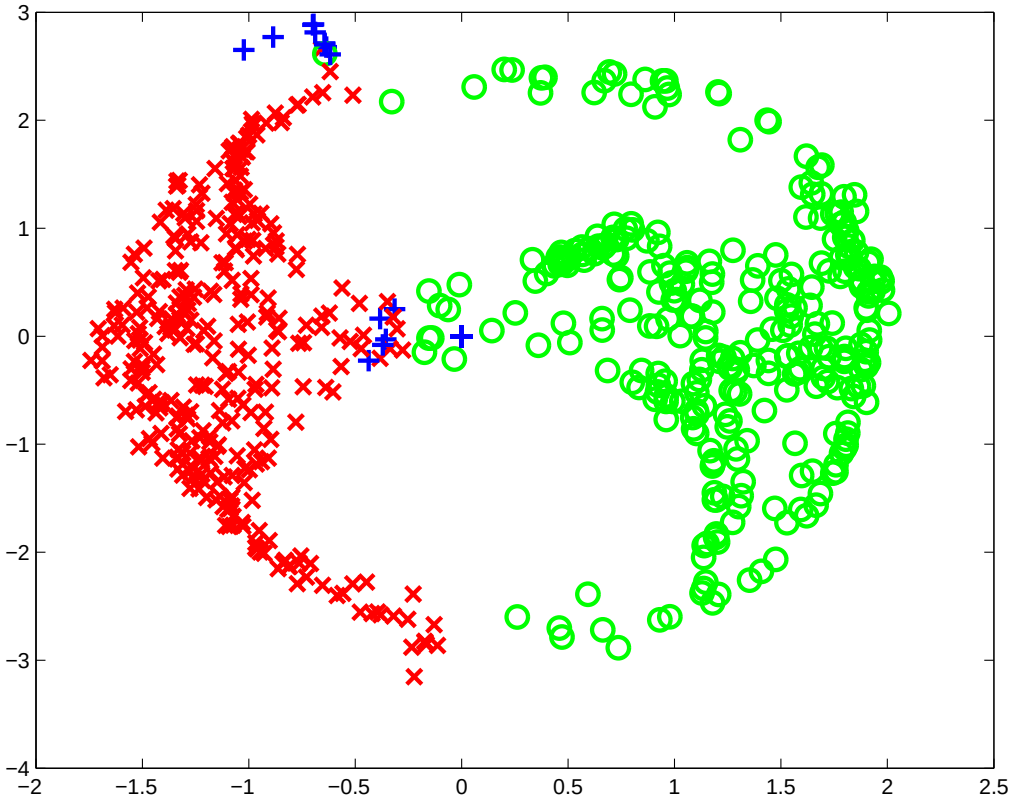


Figure: Oil data embedded using two dimensions of isomap (graph is disconnected). `demOilIsomap1`.

Isomap on Microarray Data

- Two components of Gene Expression data.

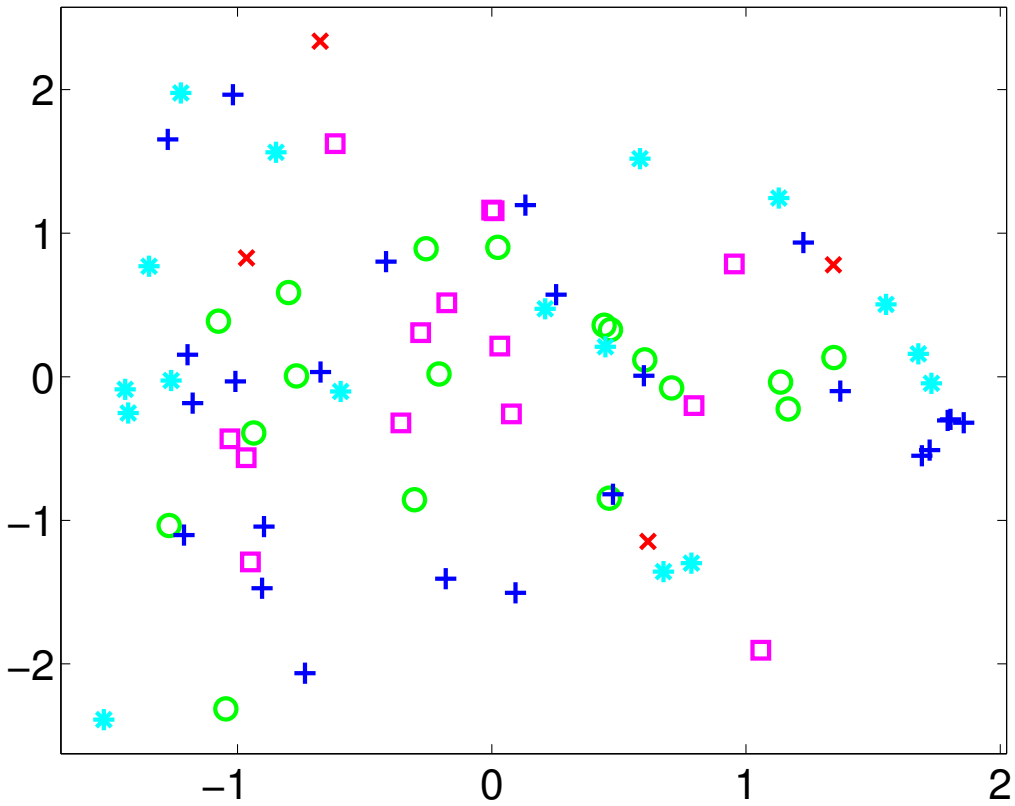


Figure: Gene expression data embedded using two dimensions of isomap. `demSpellmanIsomap1`.

Isomap on Grid Vowels

- Two components of grid vowels data.

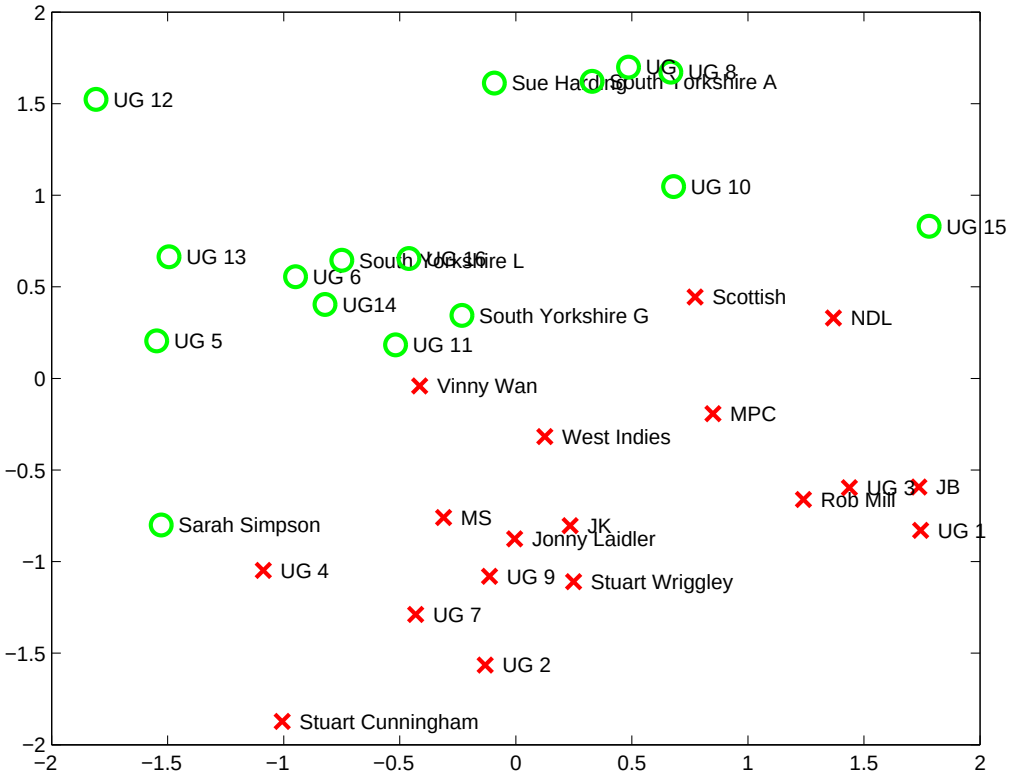


Figure: Grid vowels embedded using two dimensions of isomap. `demGrid_vowelsIsomap1`.

Isomap: Summary

MDS on Shortest Path Approximation of Geodesic Distance

+ Gives good embeddings.
- Can require solution of a very large eigenvalue problem.
- Eigenvalues can be negative (Geodesic distances aren't Euclidean).

Laplacian Eigenmaps I

- Spectral algorithm introduced by Belkin and Niyogi (2003)
- First define neighborhood in the data space.
- Define a sparse adjacency matrix, $\mathbf{A} \in \Re^{n \times n}$, i, jth element, $a_{i,j}$ is non-zero if the ith and jth data points are neighbors.
- A 'good' *one dimensional embedding* is one where the latent points, $\mathbf{X}$ minimize

$$E(\mathbf{X}) = \frac{1}{4}\sum_{i=1}^{n}\sum_{j=1}^{n} a_{i,j}(x_i - x_j)^2,$$

- Which we write as $\delta_{i,j} = \left\|\mathbf{x}_{i,:} - \mathbf{x}_{j,:}\right\|_2^2$, as

$$E(\mathbf{X}) = \frac{1}{4}\sum_{i=1}^{n}\sum_{j=1}^{n} a_{i,j}\delta_{i,j}.$$

Laplacian Eigenmaps II

- Neighboring are non-zero entries adjacency matrix and their inter-point distance in latent space is minimized.
- In matrix form

$$E(\mathbf{X}) = \frac{1}{4}\text{tr}\left(\mathbf{A}\boldsymbol{\Delta}\right).$$

- Rewrite by introducing the *Laplacian*.
  - The degree matrix, $\mathbf{D}$, is diagonal with entries, $d_{i,i} = \sum_j \mathbf{A}_{i,j}$
  - The Laplacian is written

$$\mathbf{L} = \mathbf{D} - \mathbf{A}$$

  - Error function written in terms of $\mathbf{X}$

$$E(\mathbf{X}) = \frac{1}{2}\text{tr}\left(\mathbf{L}\mathbf{X}\mathbf{X}^\top\right)$$

- Objective insensitive to translations.
- Objective minimized by placing all points on top of one another.

Laplacian Eigenmaps III

- Constrain
$$\mathbf{x}_{:,i}^{\top}\mathbf{D}\mathbf{x}_{:,i} = 1.$$
- Objective minimized by the generalized eigenvalue problem,
$$\mathbf{L}\mathbf{u}_i = \lambda_i \mathbf{D}\mathbf{u}_i,$$
- Smallest eigenvalue is zero and is associated with the constant eigenvector, it is discarded.
- Next q smallest eigenvalues are retained for the embedding.
$$\mathbf{x}_{:,i} = \mathbf{u}_{i+1} \quad \text{for} \quad i = 1..q$$

Parameterization in Laplacian Eigenmaps

- **A** either
  1. set to constant values (the "simple-minded approach" Belkin and Niyogi)
  2. or according to distance between two data points,

$$a_{i,j} = \exp\left(-\frac{\left\|\mathbf{y}_{i,:} - \mathbf{y}_{j,:}\right\|_2^2}{2\ell^2}\right),$$

  by analogy between discrete graph Laplacian and the Laplace Beltrami operator (Belkin and Niyogi, 2003).

Laplacian Eigenmaps on Stick Man

- Two components of stick man data.

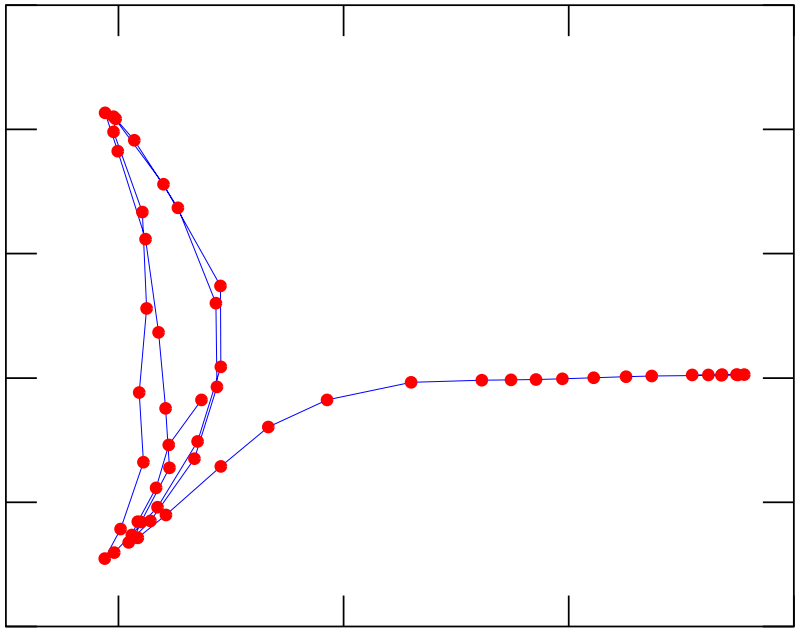


Figure: Stick man data embedded using two dimensions of Laplacian eigenmaps. `demStickLe1`.

Laplacian Eigenmaps on Oil Data

- Two components of oil data.

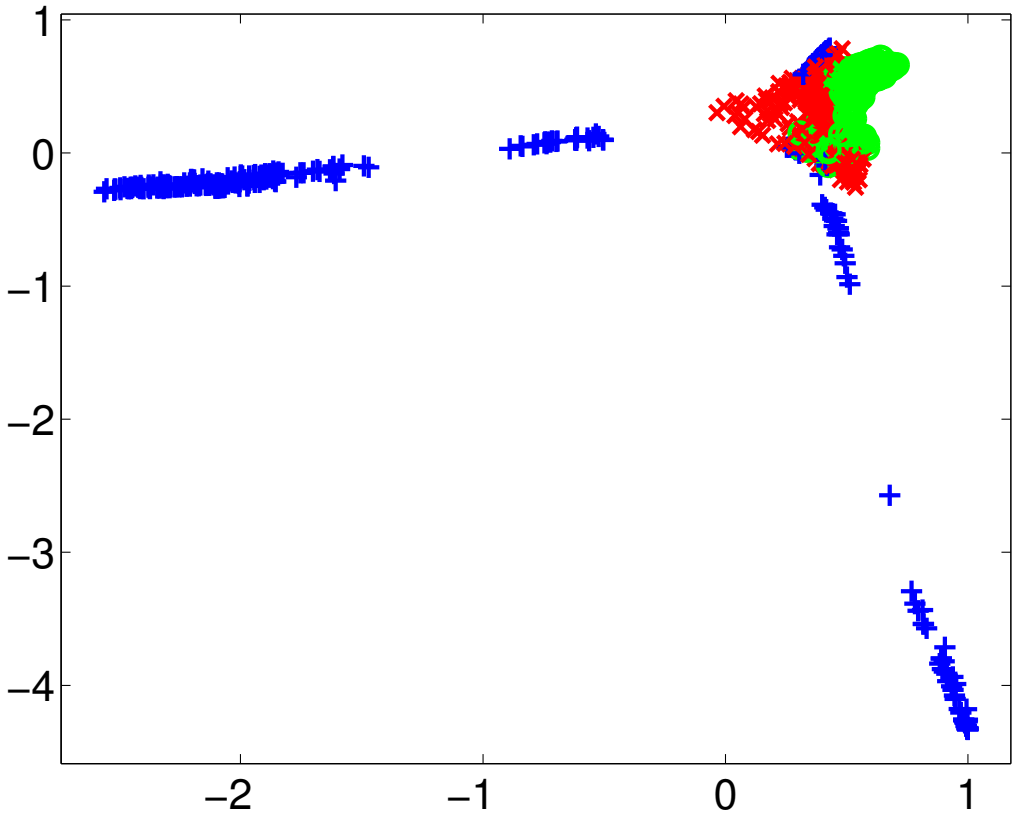


Figure: Oil data embedded using two dimensions of Laplacian eigenmaps (45 neighbors). `demOilLe1`.

Laplacian Eigenmaps on Microarray Data

- Two components of Gene Expression data.

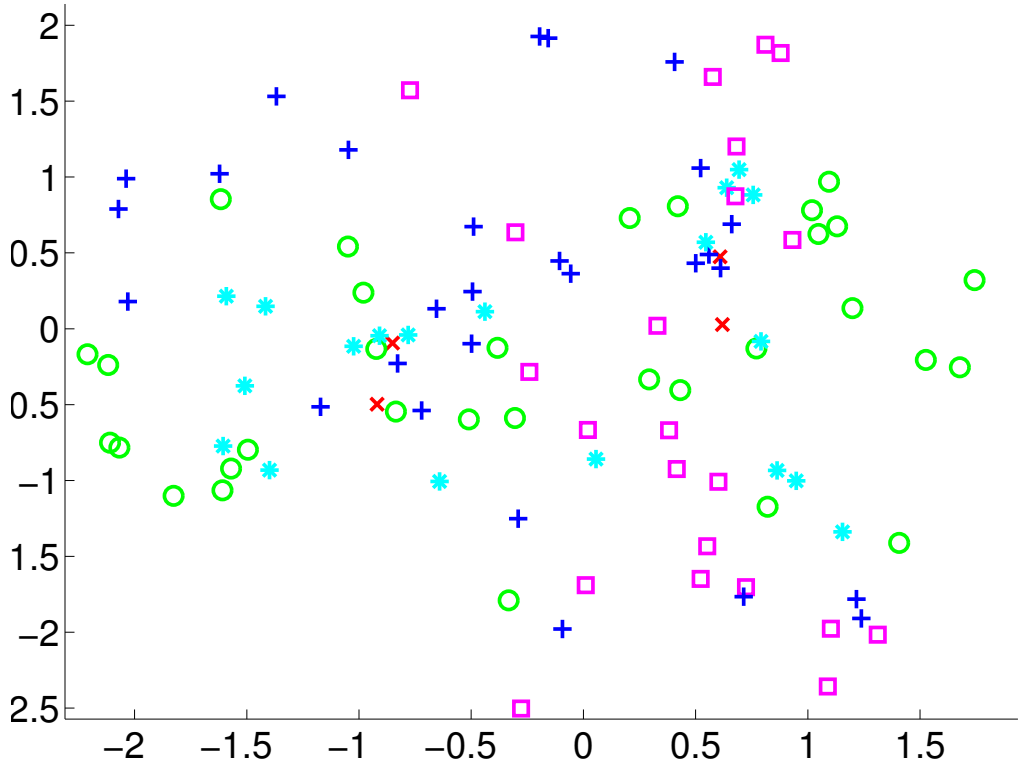


Figure: Gene expression data embedded using two dimensions of Laplacian eigenmaps. `demSpellmanLe1`.

Laplacian Eigenmaps on Grid Vowels

- Two components of grid vowels data.

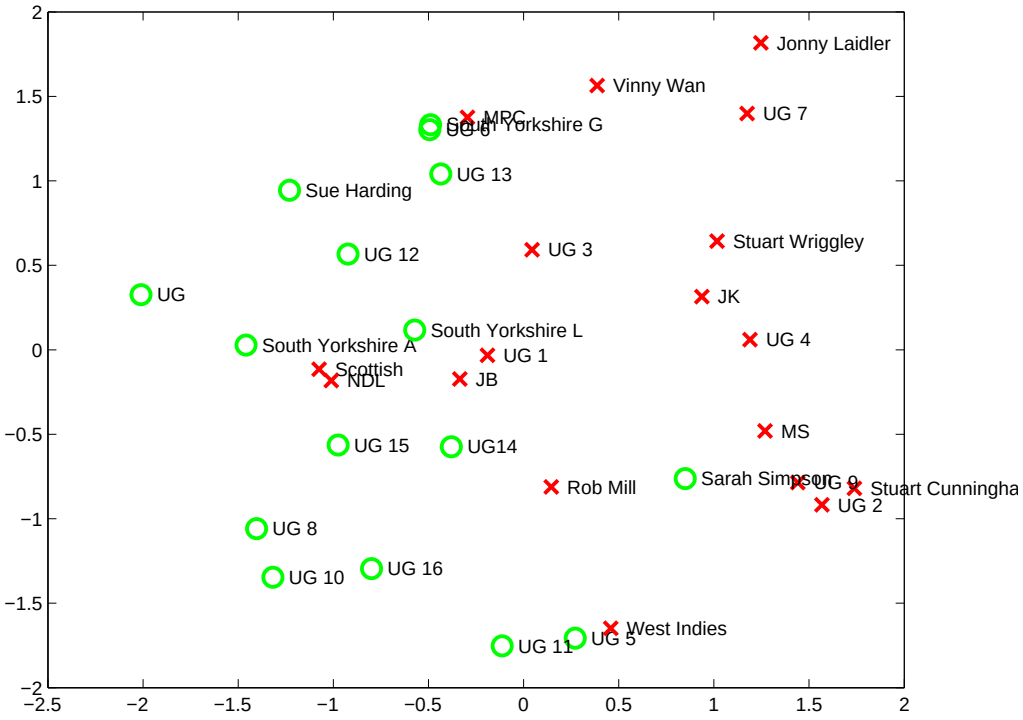


Figure: Grid vowels embedded using two dimensions of Laplacian eigenmaps. `demGrid_vowelsLe1`.

Laplacian Eigenmaps: Summary

Eigenvalue problem on Graph Laplacian

+ Very fast to compute
- Can give poor embeddings for few data points.

Locally Linear Embedding I

- Approximate Non-linear Manifold by small linear patches.
- Assumes distance between data points is small relative to curvature.
- First define a local neighborhood for each data point.
- Find a set of linear regression weights for each data point to be reconstructed by its neighbors.
- For the ith data point, $\mathbf{y}_{i,:}$ and reconstruction weights, $\mathbf{w}_{:,i}$, least squares regression objective is,

$$E(\mathbf{w}_{:,i}) = \frac{1}{2}\left\|\mathbf{y}_{i,:} - \sum_{j\in\mathcal{N}(i)} \mathbf{y}_{j,:} w_{j,i}\right\|_2^2, \tag{1}$$

- Sum over the weights, $\mathbf{w}_{:,j}$ is restricted to neighbors, $\left\{\mathbf{y}_{j,:}\right\}_{j\in\mathcal{N}((()i)}$.

Locally Linear Embedding II

- Objective is invariant to rotation and rescaling of the data.
- The objective is not invariant to translation.
- Use modified objective,

$$E(\mathbf{w}_{:,i}) = \frac{1}{2}\left\|\hat{\mathbf{y}}_{i,:} + \boldsymbol{\mu} - \sum_{j\in\mathcal{N}(i)} \hat{\mathbf{y}}_{j,:} w_{j,i} - \boldsymbol{\mu} \sum_{j\in\mathcal{N}(i)} w_{j,i}\right\|_2^2,$$

  and constrain $\sum_{j\in\mathcal{N}(i)} w_{j,i} = 1$.
- Terms involving $\boldsymbol{\mu}$ cancel and we recover the original objective.
- Constraint $\mathbf{w}_{:,i}^\top \mathbf{1} = 1$ ensures the objective is translation invariant.

Determining the Embedding in LLE I

- For truly low dimensional data, local linear relationships between neighbors should hold for a low dimensional data set we call $\mathbf{X}$.
- To find this dataset minimize the LLE objective.

$$
\begin{aligned}
E(\mathbf{X}) &= \frac{1}{2}\sum_{i=1}^{n} \mathbf{m}_{:,i}^{\top}\mathbf{X}\mathbf{X}^{\top}\mathbf{m}_{:,i} + \text{const} \\
&= \frac{1}{2}\text{tr}\left(\mathbf{M}\mathbf{M}^{\top}\mathbf{X}\mathbf{X}\right) + \text{const} \\
&= \frac{1}{2}\sum_{i=1}^{n} \mathbf{x}_{i,:}^{\top}\mathbf{M}\mathbf{M}^{\top}\mathbf{x}_{i,:} + \text{const}.
\end{aligned}
$$

- Objective function trivially minimized by setting $\mathbf{X} = \mathbf{0}$, so we constrain $\mathbf{X}^{\top}\mathbf{X} = \mathbf{I}$.

Determining the Embedding in LLE II

- This leads to an eigenvalue problem

$$\mathbf{M}\mathbf{M}^\top \mathbf{u}_i = \lambda_i \mathbf{u}_i.$$

  Where smallest $q + 1$ eigenvalues are retained.
- Smallest eigenvector is the constant eigenvector and is associated with an eigenvalue of zero.
- Next q eigenvectors are retained to make up the low dimensional representation

$$\mathbf{x}_{:,i} = \mathbf{u}_{i+1} \quad \text{for} \quad i = 1..q.$$

- This process is extremely similar to Laplacian eigenmaps, despite different motivations.
- In LLE case the constraint on the latent embeddings is not scaled by the degree matrix.

LLE on Stick Man

- Two components of stick man data.

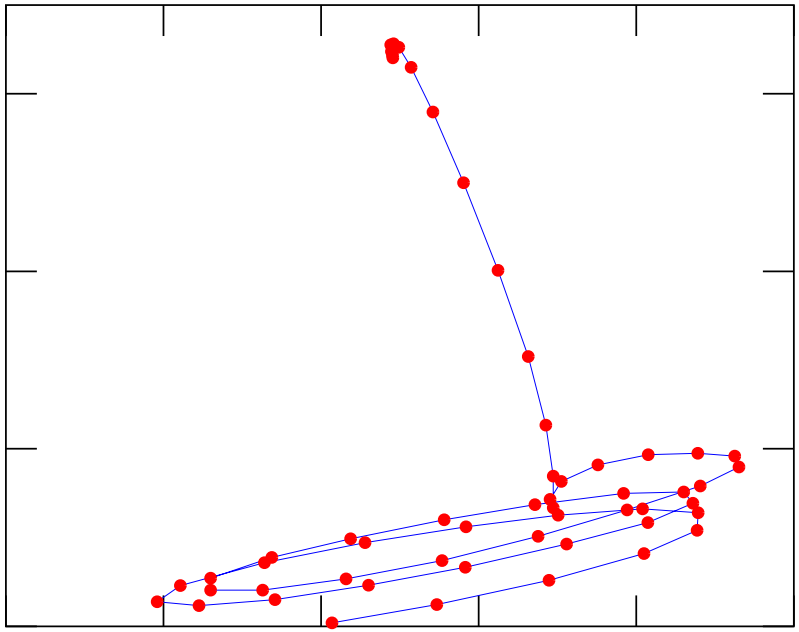


Figure: Stick man data embedded using two dimensions of LLE. `demStickLle1`.

LLE on Oil Data

- Two components of oil data.

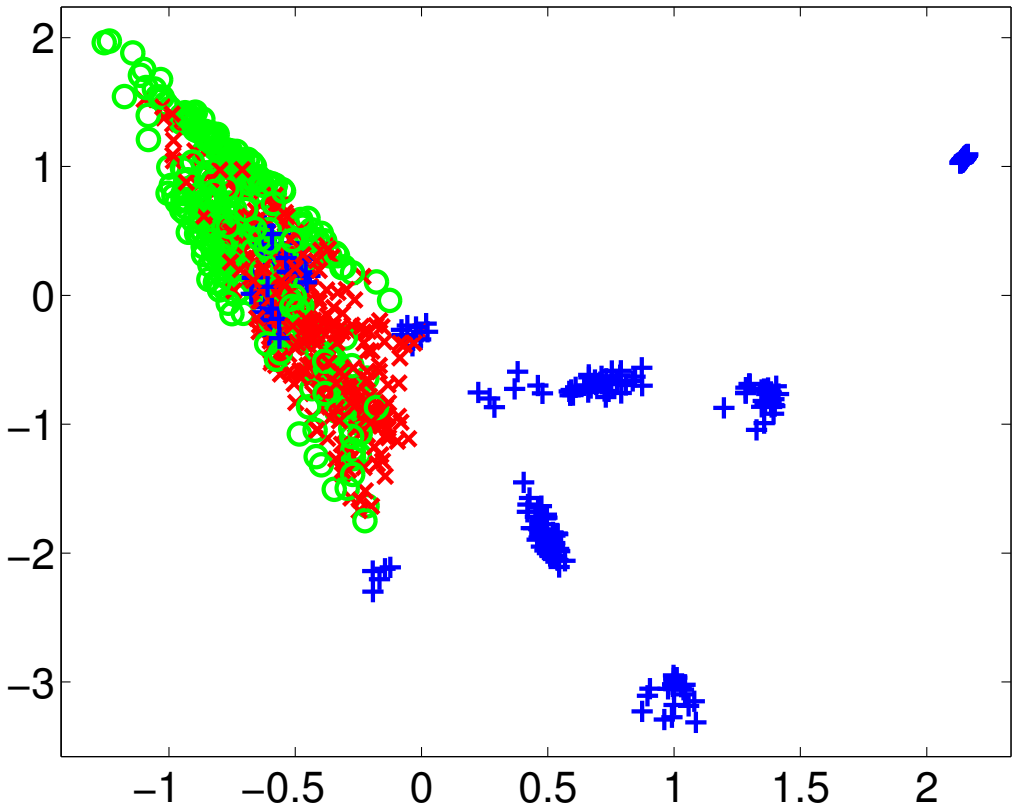


Figure: Oil data embedded using two dimensions of LLE (45 neighbors). `demOilLle1`.

LLE on Microarray Data

- Two components of Gene Expression data.

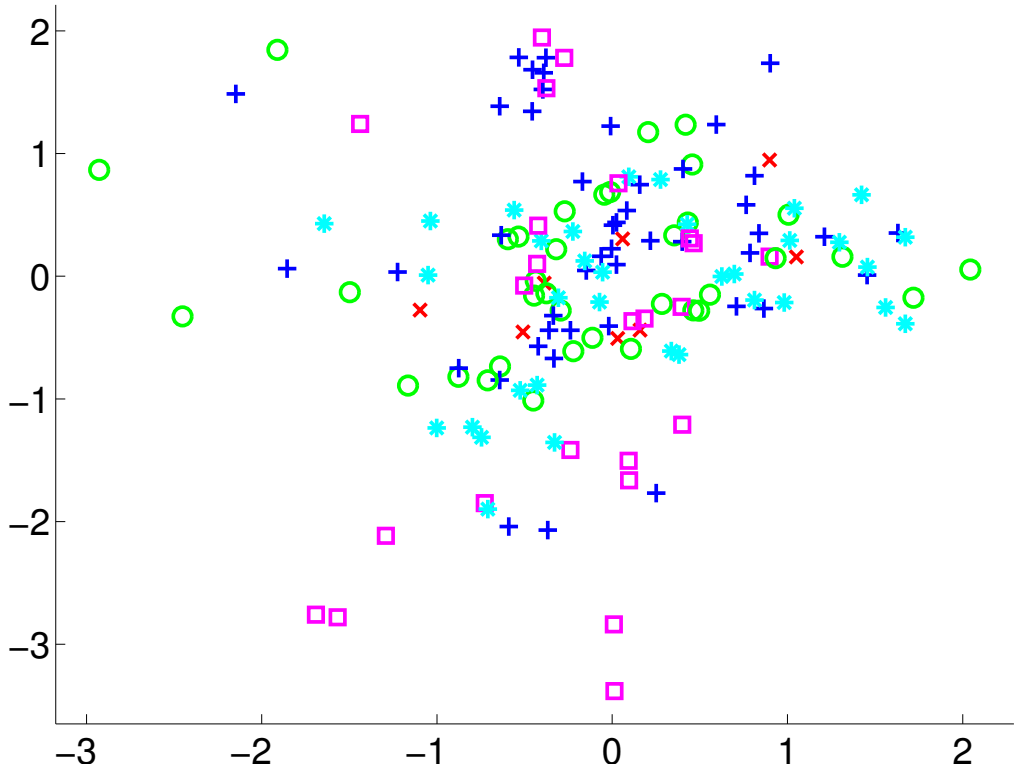


Figure: Gene expression data embedded using two dimensions of LLE. `demSpellmanLle1`.

LLE on Grid Vowels

- Two components of grid vowels data.

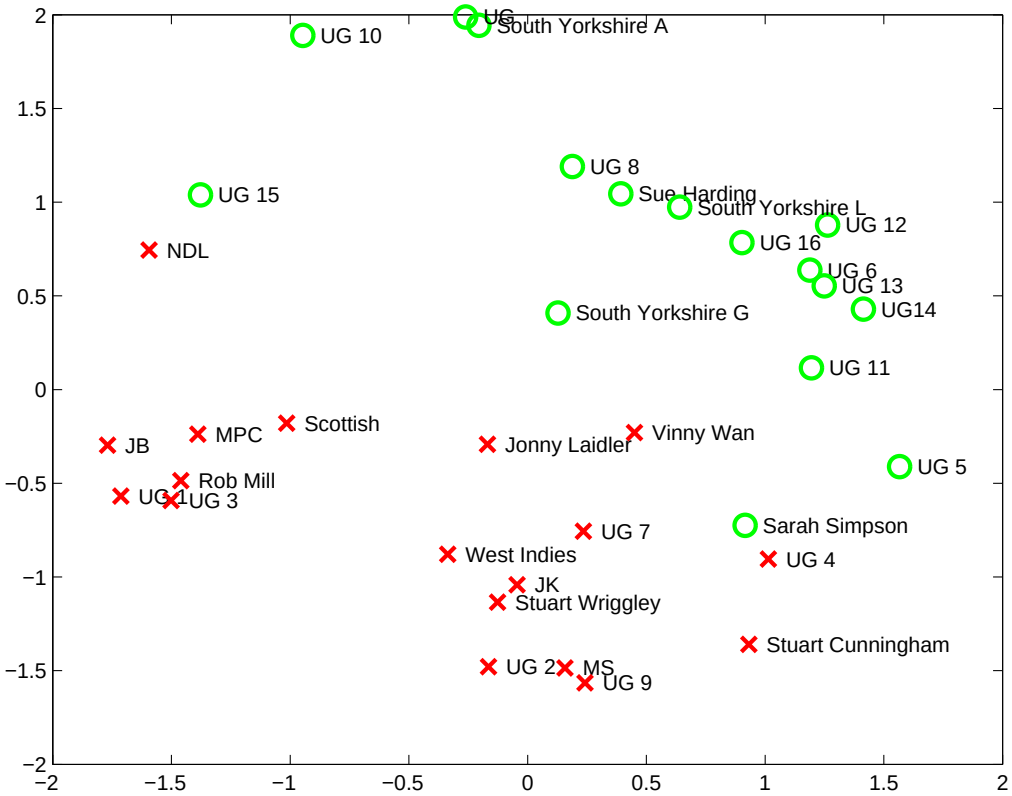


Figure: Grid vowels embedded using two dimensions of LLE. `demGrid_vowelsLle1`.

Locally Linear Embedding: Summary

Model Data with Locally Linear Patches

+ Faster than isomap, slower than LE.
- Can still give poor embeddings for few data points.

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.

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.

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.

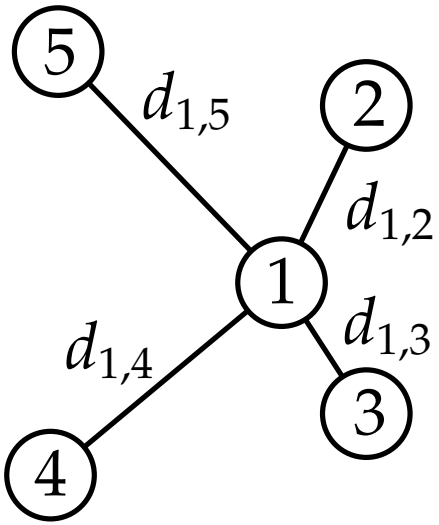
Maximum Variance Unfolding

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[1] $\mathrm{tr}(\mathbf{K})$.



- Subject to squared distance constraints between neighbors

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

MVU on Stick Man

- Two components of stick man data.

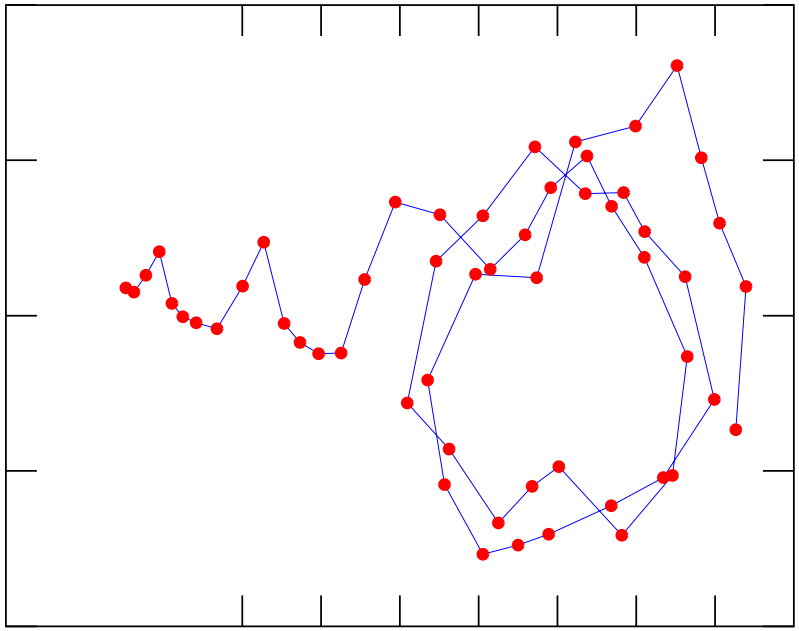


Figure: Stick man data embedded using two dimensions of isomap. `demStickMvu1`.

MVU on Oil Data

- Graph doesn't fully connect until 30 neighbors are used.
- Resulting semi-definite program is too big for SeDuMi on my machine (32GB memory, but it swaps in MATLAB).
- There is approximate version of the algorithm, not applied in this case.

MVU on Grid Vowels

- Two components of grid vowels data.

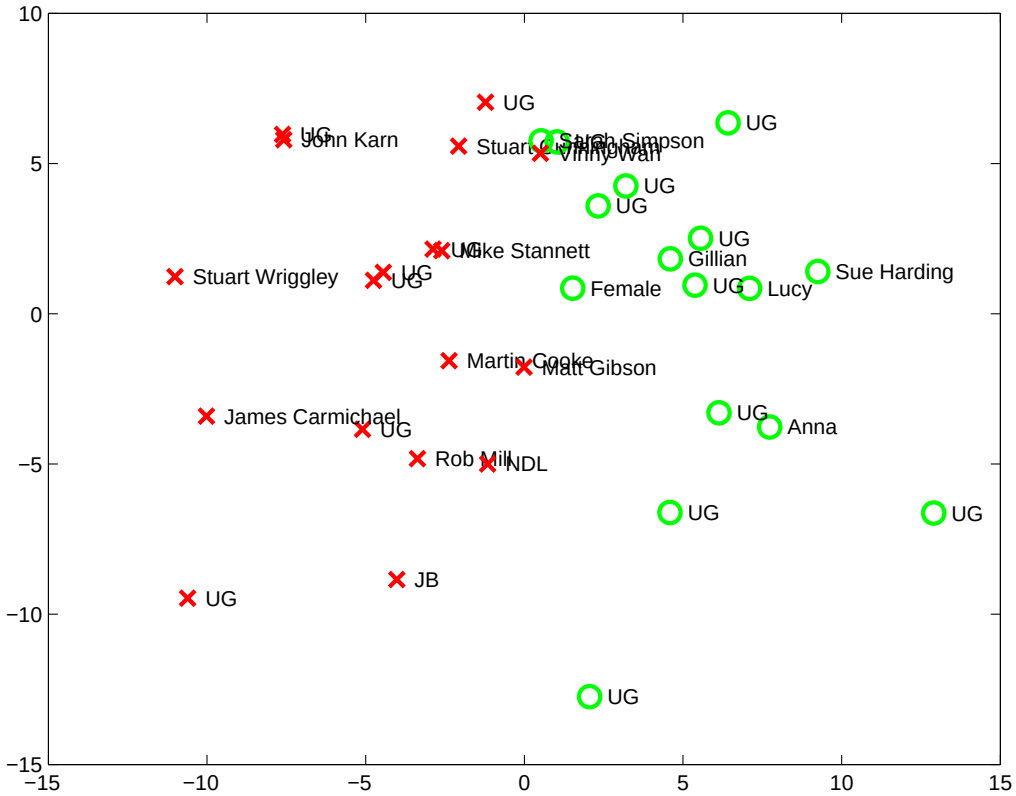


Figure: Grid vowels embedded using two dimensions of isomap. `demGrid_vowelsMvu1`.

Maximum Variance Unfolding: Summary

Chain Neighboring Data together and Maximum Data Variance

+ High quality embeddings with no negative eigenvalues.
- Slower than isomap, LLE and LE.

Outline

Maximum Entropy Unfolding

New Contribution

- Maximize *entropy* instead of variance (Jaynes, 1986): MEU (Lawrence, 2011, 2010).

Maximum Entropy Unfolding

New Contribution

- Maximize *entropy* instead of variance (Jaynes, 1986): MEU (Lawrence, 2011, 2010).
- Entropy and variance are closely related.

Maximum Entropy Unfolding

New Contribution

- Maximize *entropy* instead of variance (Jaynes, 1986): MEU (Lawrence, 2011, 2010).
- Entropy and variance are closely related.
- Maximum entropy leads to a *probabilistic model*.

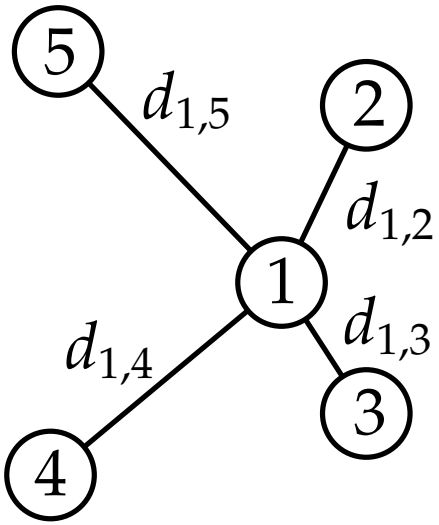
Maximum Entropy Unfolding

New Contribution

- Maximize *entropy* instead of variance (Jaynes, 1986): MEU (Lawrence, 2011, 2010).
- 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*.

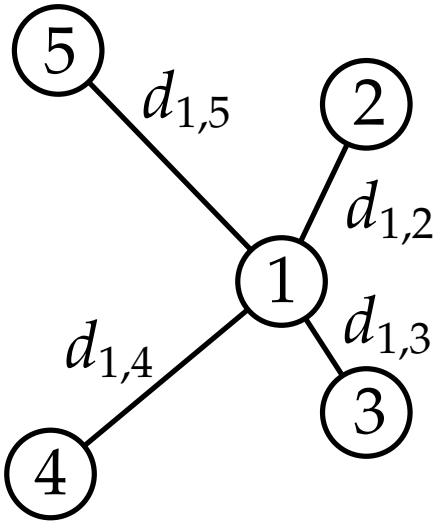


- MEU constraints are on expected distances between neighbors.

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

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.

Maximum Entropy

- Maximum entropy distribution.

$$p(\mathbf{Y}) \propto \exp\left(-\frac{1}{2}\mathrm{tr}\left(\gamma \mathbf{Y}\mathbf{Y}^\top\right)\right)\exp\left(-\frac{1}{2}\sum_i \sum_{j\in\mathcal{N}(i)} \lambda_{i,j} d_{i,j}\right)$$

$\mathcal{N}(i)$ is neighborhood, $\{\lambda_{i,j}\}$, Lagrange multipliers.

Maximum Entropy

- Maximum entropy distribution.

$$p(\mathbf{Y}) \propto \exp\left(-\frac{1}{2}\mathrm{tr}\left(\gamma \mathbf{Y}\mathbf{Y}^\top\right) - \frac{1}{4}\mathrm{tr}\left(\mathbf{\Lambda}\mathbf{D}\right)\right)$$

$\mathcal{N}(i)$ is neighborhood, $\{\lambda_{i,j}\}$, Lagrange multipliers. Lagrange multipliers in sparse matrix $\mathbf{\Lambda}$.

Maximum Entropy

- Maximum entropy distribution.

$$p(\mathbf{Y}) = \frac{\left|\mathbf{L} + \gamma \mathbf{I}\right|^{\frac{1}{2}}}{(2\pi)^{\frac{np}{2}}} \exp\left(-\frac{1}{2}\mathrm{tr}\left((\mathbf{L} + \gamma \mathbf{I})\mathbf{Y}\mathbf{Y}^{\top}\right)\right)$$

$\mathcal{N}(i)$ is neighborhood, $\{\lambda_{i,j}\}$, Lagrange multipliers.
Introduce Laplacian: $\ell_{i,j} = -\lambda_{i,j}$, $\ell_{i,i} = \sum_{j \in \mathcal{N}(i)} \lambda_{i,j}$, $\mathbf{L1} = \mathbf{0}$.

Details: Moving to the Laplacian

- $\mathbf{D}$ has a zero diagonal.
- $\mathrm{tr}\,(\mathbf{LD})$ is unaffected by diagonal of $\mathbf{L}$.
- Constrain $\mathbf{L1} = \mathbf{0}$ giving

$$-\mathrm{tr}\,(\mathbf{\Lambda D}) = \mathrm{tr}\,(\mathbf{LD})$$

Details: Moving to the Laplacian

- $\mathbf{D}$ has a zero diagonal.
- $\mathrm{tr}\,(\mathbf{LD})$ is unaffected by diagonal of $\mathbf{L}$.
- Constrain $\mathbf{L1} = \mathbf{0}$ giving

$$-\mathrm{tr}\,(\mathbf{\Lambda D}) = \mathrm{tr}\,(\mathbf{LD})$$

Details: Moving to the Laplacian

- $\mathbf{D}$ has a zero diagonal.
- $\mathrm{tr}\,(\mathbf{LD})$ is unaffected by diagonal of $\mathbf{L}$.
- Constrain $\mathbf{L1} = \mathbf{0}$ giving

$$-\mathrm{tr}\,(\mathbf{\Lambda D}) = \mathrm{tr}\left(\mathbf{L1}\mathrm{diag}\left(\mathbf{YY}^\top\right)^\top - 2\mathbf{LYY}^\top + \mathrm{diag}\left(\mathbf{YY}^\top\right)\mathbf{1}^\top\mathbf{L}\right)$$

Details: Moving to the Laplacian

- $\mathbf{D}$ has a zero diagonal.
- $\mathrm{tr}\,(\mathbf{LD})$ is unaffected by diagonal of $\mathbf{L}$.
- Constrain $\mathbf{L1} = \mathbf{0}$ giving

$$-\mathrm{tr}\,(\mathbf{\Lambda D}) = \mathrm{tr}\left(\cancel{\mathbf{L1}\mathrm{diag}\left(\mathbf{YY}^\top\right)^\top} - 2\mathbf{LYY}^\top + \cancel{\mathrm{diag}\left(\mathbf{YY}^\top\right)\mathbf{1}^\top\mathbf{L}}\right)$$

Details: Moving to the Laplacian

- $\mathbf{D}$ has a zero diagonal.
- $\mathrm{tr}\,(\mathbf{LD})$ is unaffected by diagonal of $\mathbf{L}$.
- Constrain $\mathbf{L1} = \mathbf{0}$ giving

$$-\mathrm{tr}\,(\mathbf{\Lambda D}) = -2\mathrm{tr}\left(\mathbf{L}\mathbf{Y}\mathbf{Y}^{\top}\right).$$

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 ($\mathbf{L1} = \mathbf{0}$).

Data Feature Independence

- The GRF specifying independence across data *features*.
- Most applications of Gaussian models are applied independently across data *points*.
  - Notable exceptions include Zhu et al. (2003); Lawrence (2004, 2005); Kemp and Tenenbaum (2008).
- Maximum likelihood in this model is equivalent maximizing entropy under distance constraints.

Blessing of Dimensionality

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

Blessing of Dimensionality

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

- Maximum likelihood is consistent: (see e.g. Wasserman, 2003, pg 126)

Blessing of Dimensionality

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

- Maximum likelihood is consistent: (see e.g. Wasserman, 2003, pg 126)
  - As we increase data points parameters become better determined.

Blessing of Dimensionality

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

- Maximum likelihood is consistent: (see e.g. Wasserman, 2003, pg 126)
  - As we increase data points parameters become better determined.
  - **Not** in this model.

Blessing of Dimensionality

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

- Maximum likelihood is consistent: (see e.g. Wasserman, 2003, pg 126)
  - As we increase data points parameters become better determined.
  - **Not** in this model.
  - As we increase data features parameters become better determined.

Blessing of Dimensionality

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

- Maximum likelihood is consistent: (see e.g. Wasserman, 2003, pg 126)
  - As we increase data points parameters become better determined.
  - **Not** in this model.
  - As we increase data features parameters become better determined.
- This turns the large p small n problem on its head.

Blessing of Dimensionality

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

- Maximum likelihood is consistent: (see e.g. Wasserman, 2003, pg 126)
  - As we increase data points parameters become better determined.
  - **Not** in this model.
  - As we increase data features parameters become better determined.
- This turns the large p small n problem on its head.
- There is a "Blessing of Dimensionality" in this model.

Blessing of Dimensionality

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

- Maximum likelihood is consistent: (see e.g. Wasserman, 2003, pg 126)
  - As we increase data points parameters become better determined.
  - **Not** in this model.
  - As we increase data features parameters become better determined.
- This turns the large p small n problem on its head.
- There is a "Blessing of Dimensionality" in this model.

Inverse Covariance

- From the "covariance interpretation" we think of the similarity matrix as a covariance matrix.
  - Each element of the covariance is a function of two data points.
- For LE, LLE and MVU the stiffness matrix is like an *inverse covariance*.
  - This is a *conditional independence* assumption.
  - Describes how points are connected.

Conditional Independence

- A covariance matrix specifies correlation between two variables. If elements are zero those variables are *truly* independent.
  - In a marginal Gaussian those correlations don't change.
- The inverse covariance (precision, or information matrix) specifies conditional independencies.
  - If elements are zero those variables are *conditionally* independent.

Mattress Model

- Points are connected by springs.

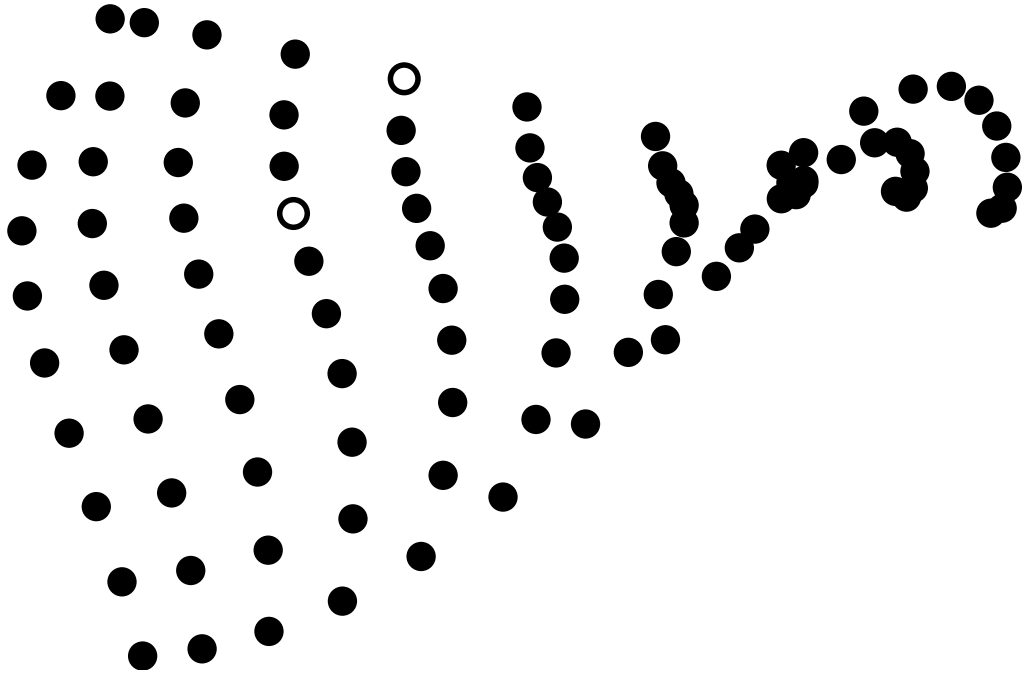


Figure: Physical interpretation of spectral models.

Mattress Model

- Points are connected by springs.

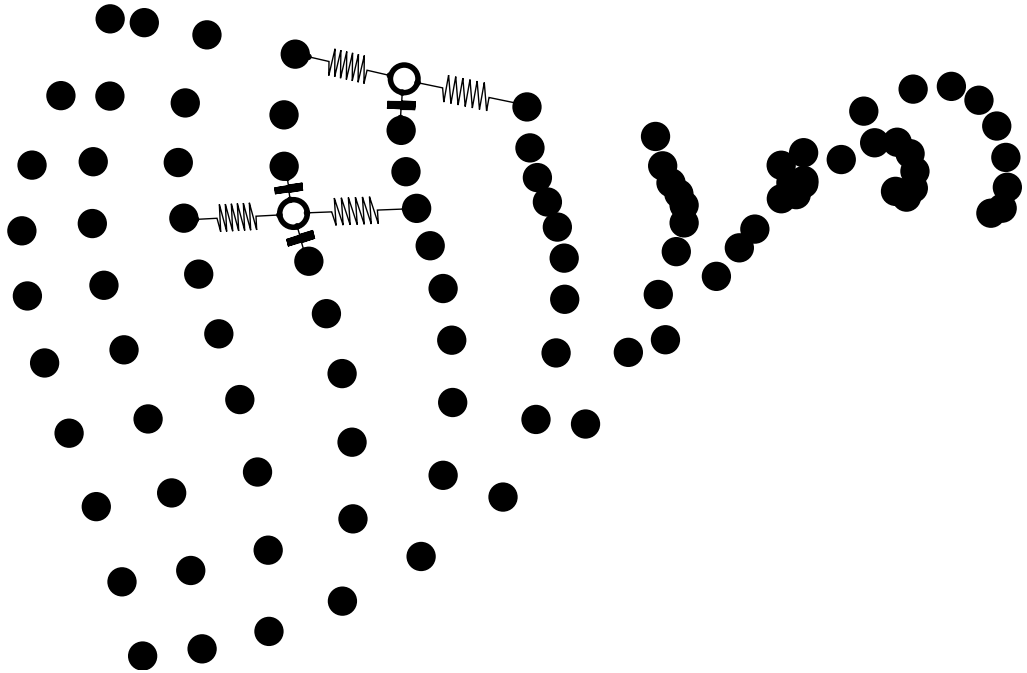
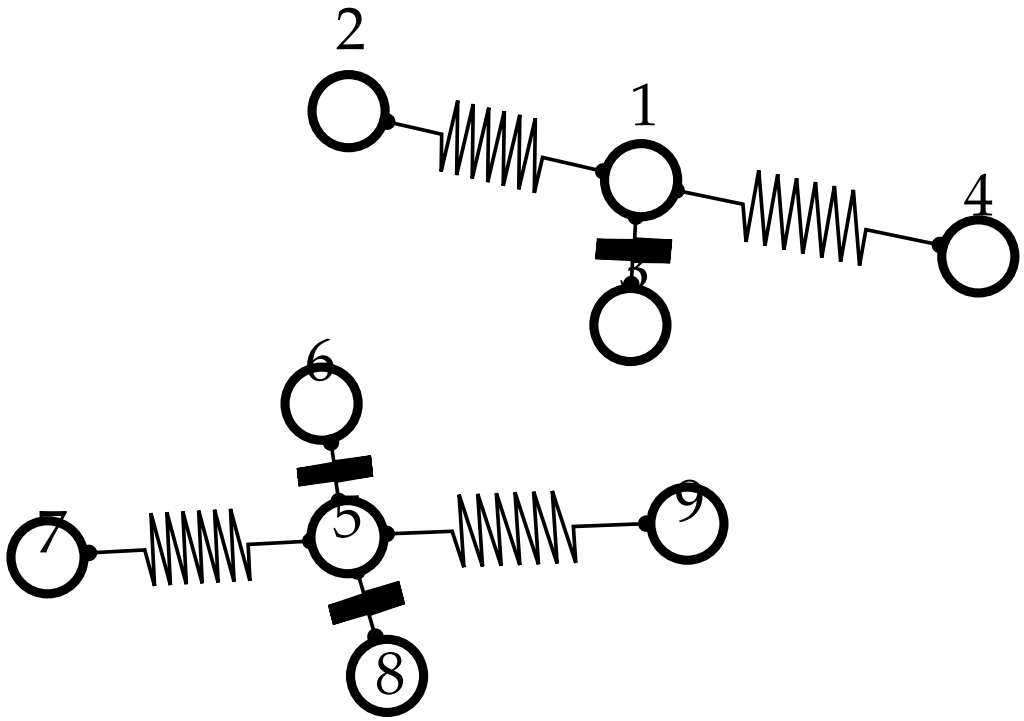


Figure: Physical interpretation of spectral models.

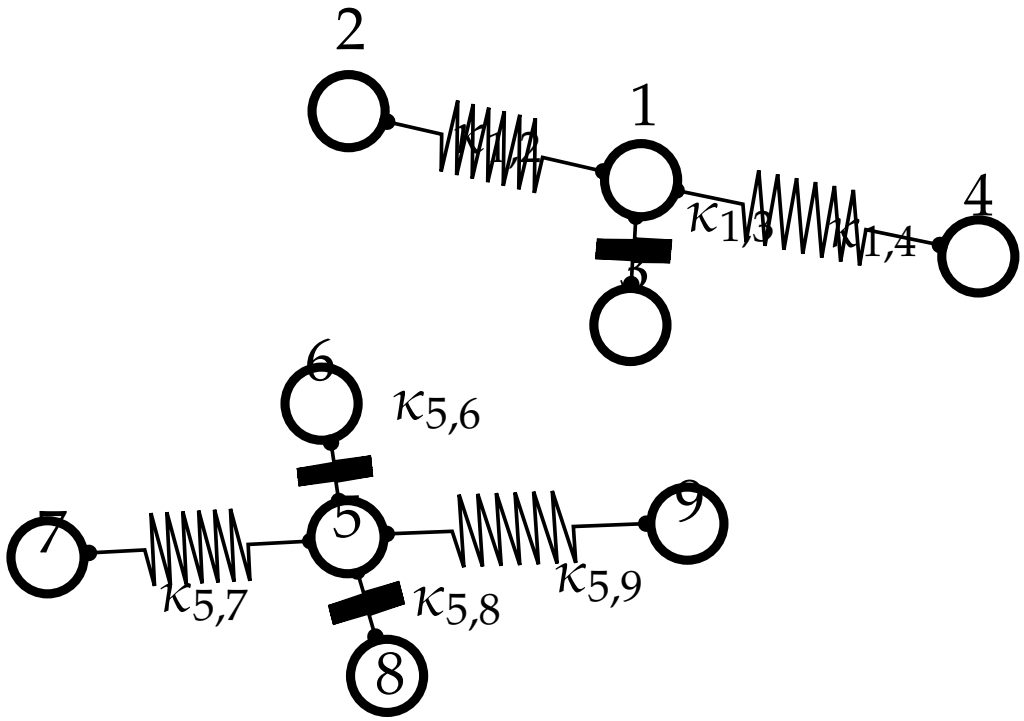
Spring Energy

- Points are connected by springs.



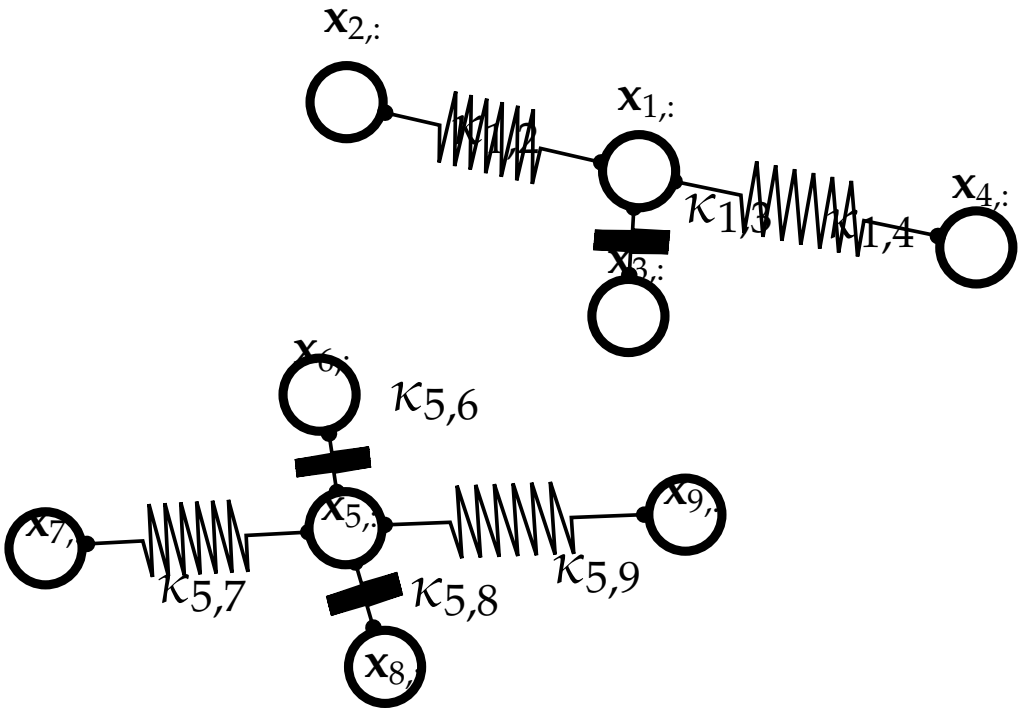
Spring Energy

- Each spring has its own spring constant.



Spring Energy

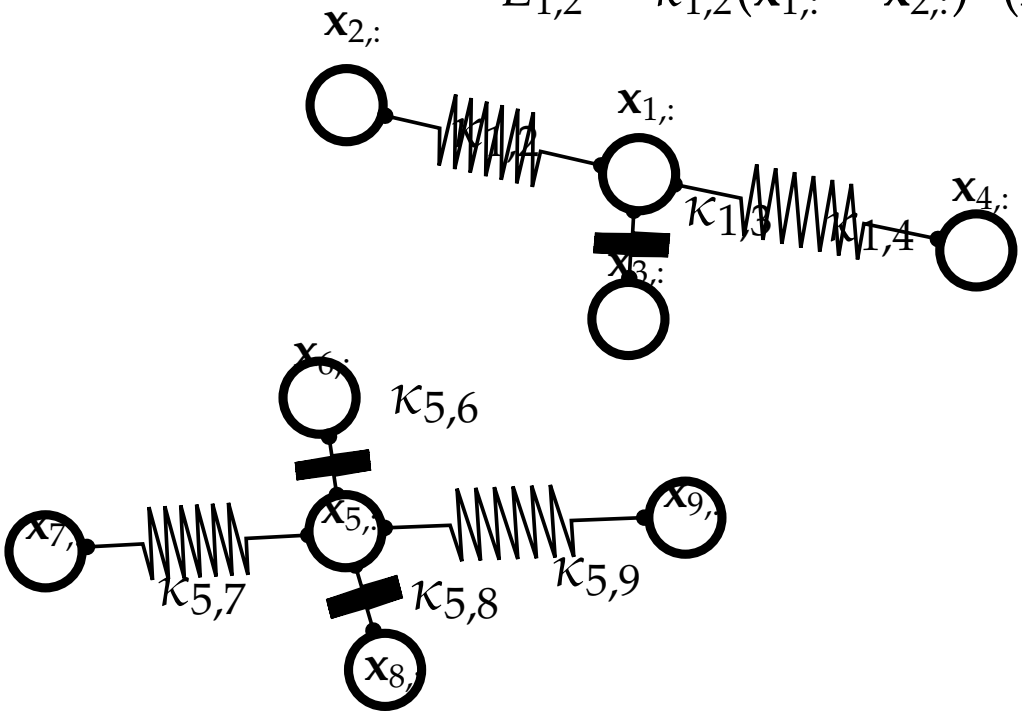
- Place each point at its latent location.



Spring Energy

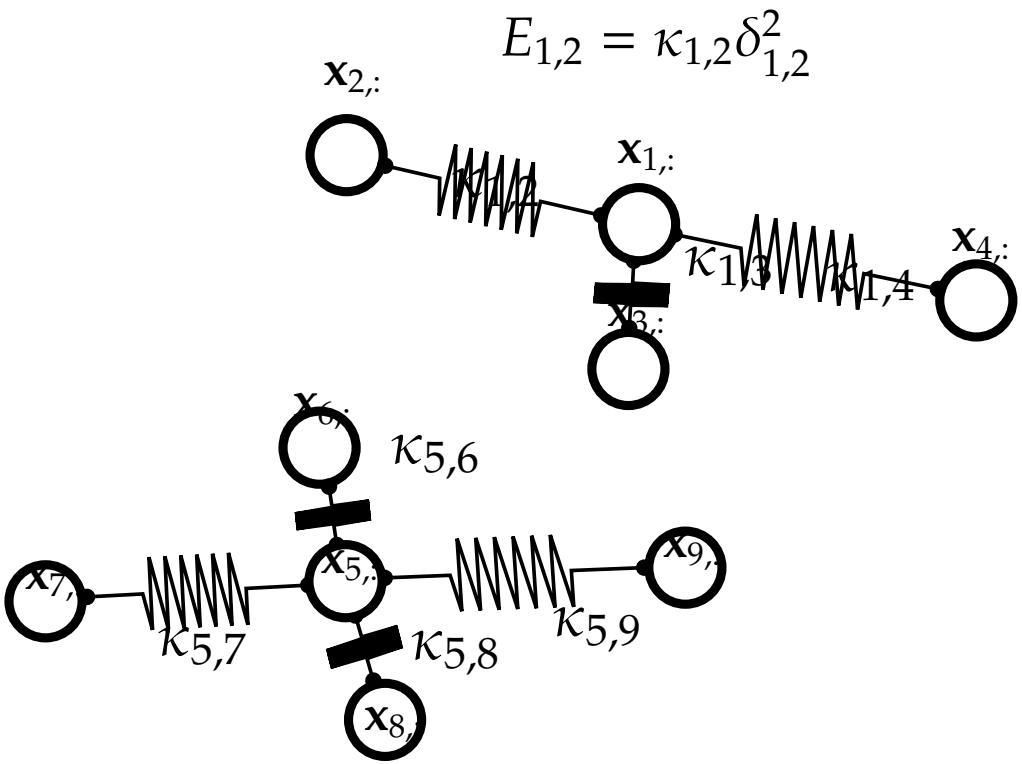
- Potential energy in each string is given by.

$$E_{1,2} = \kappa_{1,2}(\mathbf{x}_{1,:} - \mathbf{x}_{2,:})^\top(\mathbf{x}_{1,:} - \mathbf{x}_{2,:})$$



Spring Energy

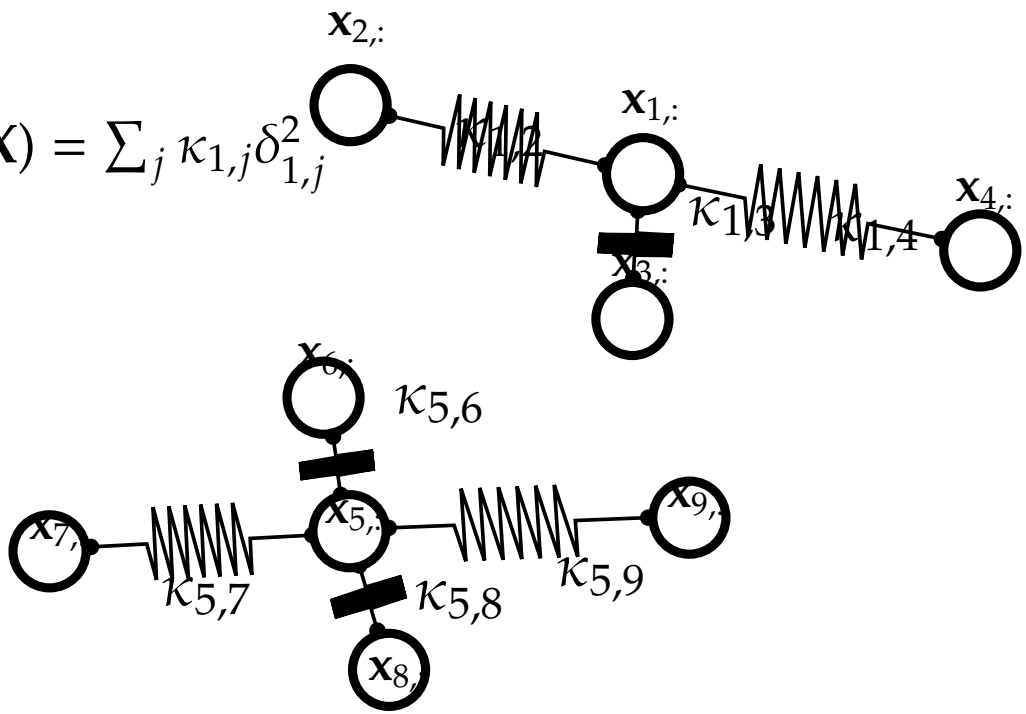
- Which can be expressed as a latent distance.



Spring Energy

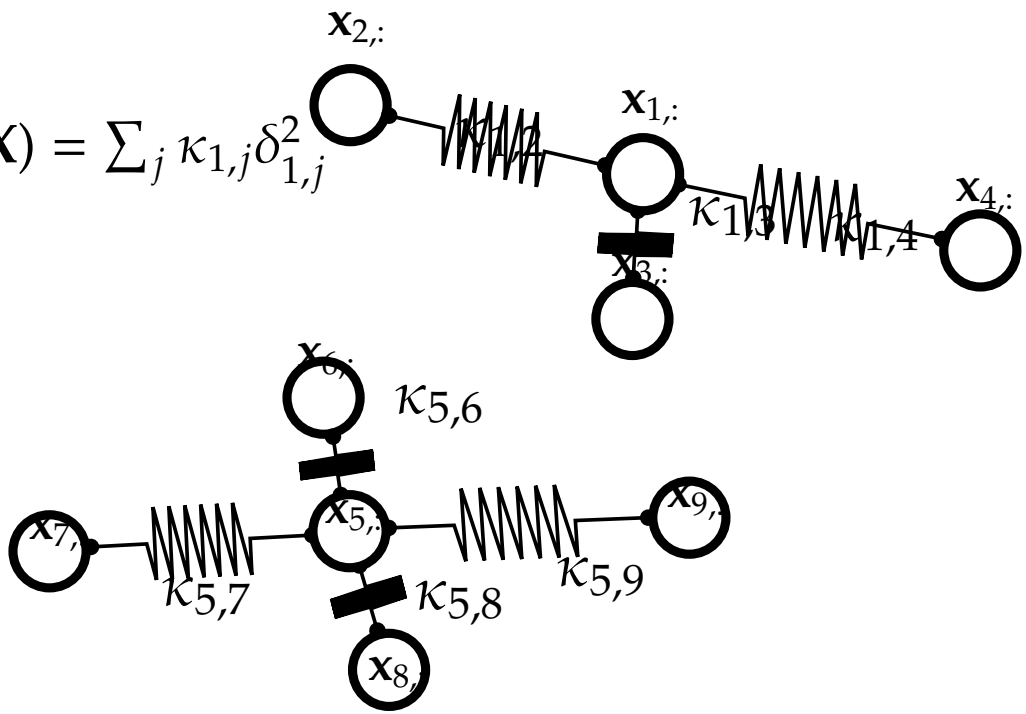
- Energy associated with each point given by sum.

$$E_1(\mathbf{X}) = \sum_j \kappa_{1,j}\delta^2_{1,j}$$



Spring Energy

- Energy associated with system is sum over points.

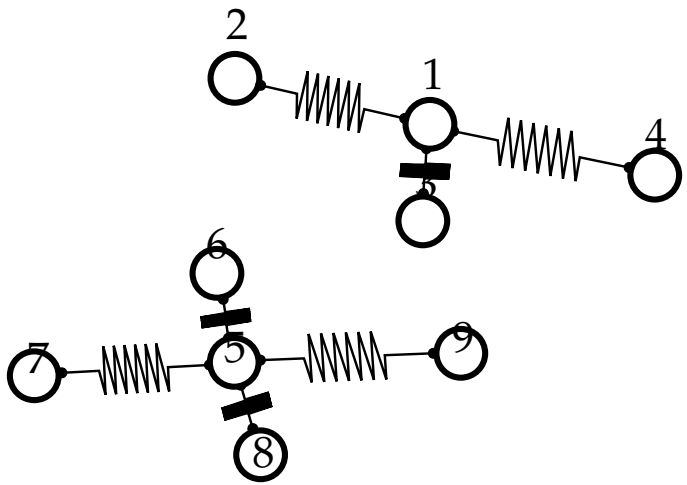


$$E_1(\mathbf{X}) = \sum_j \kappa_{1,j}\delta_{1,j}^2$$

$$E_5(\mathbf{X}) = \sum_j \kappa_{5,j}\delta_{5,j}^2$$

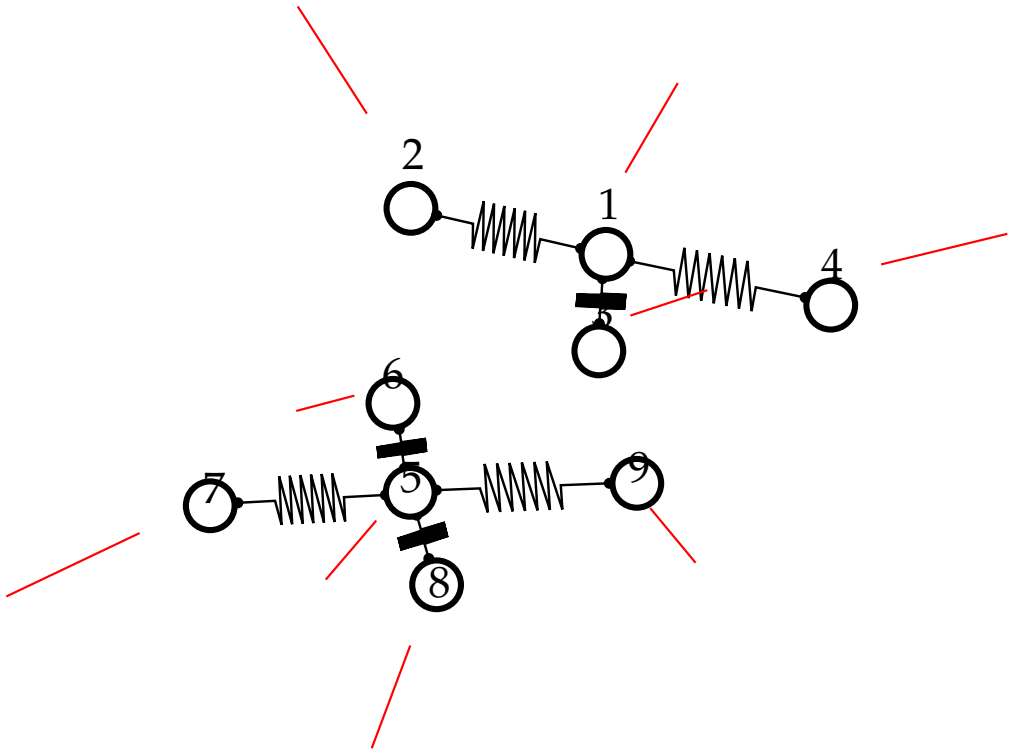
Physical Analogy

- System total energy given by $E(\mathbf{X}, \boldsymbol{\mathcal{K}}) = \sum_{i=1}^{n} \sum_{j=1}^{n} \kappa_{i,j} \delta_{i,j}^2$



Physical Analogy

- System total energy given by $E(\mathbf{X}, \boldsymbol{\mathcal{K}}) = \sum_{i=1}^{n} \sum_{j=1}^{n} \kappa_{i,j} \delta_{i,j}^2$



- Include a force to repel points from the origin

$$E(\mathbf{X}, \boldsymbol{\mathcal{K}}) = -\sum^{n} \mathbf{x}_{i,:}^{\top} \mathbf{x}_{i,:} + \sum^{n} \sum^{n} \kappa_{i,j} \delta_{i,j}^2$$

Energy Minimization

- Minimization with respect to $\mathbf{X}$ gives the following eigenvalue problem

$$\mathbf{L}\mathbf{U} = \mathbf{U}\Gamma\mathbf{\Lambda}^{-2}$$

where $\mathbf{L}$ is the stiffness matrix (which is also a Laplacian matrix) from the graph.

$$\ell_{i,i} = \sum_{j=1}^{n} (\kappa_{i,j} + \kappa_{j,i})$$

$$\ell_{i,j} = -(\kappa_{j,i} + \kappa_{i,j})$$

and

$$\mathbf{X} = \mathbf{U}\mathbf{\Lambda}\mathbf{R}^{\top}$$

and eigenvectors associated with the smallest eigenvalues are retained.[2]

Where Do the Spring Constants Come From?

- Algorithms assume only neighbors in data space are connected by springs (sparse connectivity).
- Different algorithms suggest different values for the springs.
  - Laplacian Eigenmaps prescribe constant spring constants, or values from an RBF on the distances (Belkin and Niyogi, 2003).
  - Locally Linear Embedding considers spring constants that lead to optimal linear reconstruction of data points (Roweis and Saul, 2000).
  - Maximum Variance Unfolding prescribes spring constants that constrain interpoint latent distances to equal those in the data (Weinberger et al., 2004).
  - Maximum Entropy Unfolding fits by maximum likelihood (Lawrence, 2011).

Where Do the Spring Constants Come From?

- Algorithms assume only neighbors in data space are connected by springs (sparse connectivity).
- Different algorithms suggest different values for the springs.
  - Laplacian Eigenmaps prescribe constant spring constants, or values from an RBF on the distances (Belkin and Niyogi, 2003).
  - Locally Linear Embedding considers spring constants that lead to optimal linear reconstruction of data points (Roweis and Saul, 2000).
  - Maximum Variance Unfolding prescribes spring constants that constrain interpoint latent distances to equal those in the data (Weinberger et al., 2004).
  - Maximum Entropy Unfolding fits by maximum likelihood (Lawrence, 2011).

Where Do the Spring Constants Come From?

- Algorithms assume only neighbors in data space are connected by springs (sparse connectivity).
- Different algorithms suggest different values for the springs.
  - Laplacian Eigenmaps prescribe constant spring constants, or values from an RBF on the distances (Belkin and Niyogi, 2003).
  - Locally Linear Embedding considers spring constants that lead to optimal linear reconstruction of data points (Roweis and Saul, 2000).
  - Maximum Variance Unfolding prescribes spring constants that constrain interpoint latent distances to equal those in the data (Weinberger et al., 2004).
  - Maximum Entropy Unfolding fits by maximum likelihood (Lawrence, 2011).

Where Do the Spring Constants Come From?

- Algorithms assume only neighbors in data space are connected by springs (sparse connectivity).
- Different algorithms suggest different values for the springs.
  - Laplacian Eigenmaps prescribe constant spring constants, or values from an RBF on the distances (Belkin and Niyogi, 2003).
  - Locally Linear Embedding considers spring constants that lead to optimal linear reconstruction of data points (Roweis and Saul, 2000).
  - Maximum Variance Unfolding prescribes spring constants that constrain interpoint latent distances to equal those in the data (Weinberger et al., 2004).
  - Maximum Entropy Unfolding fits by maximum likelihood (Lawrence, 2011).

Where Do the Spring Constants Come From?

- Algorithms assume only neighbors in data space are connected by springs (sparse connectivity).
- Different algorithms suggest different values for the springs.
  - Laplacian Eigenmaps prescribe constant spring constants, or values from an RBF on the distances (Belkin and Niyogi, 2003).
  - Locally Linear Embedding considers spring constants that lead to optimal linear reconstruction of data points (Roweis and Saul, 2000).
  - Maximum Variance Unfolding prescribes spring constants that constrain interpoint latent distances to equal those in the data (Weinberger et al., 2004).
  - Maximum Entropy Unfolding fits by maximum likelihood (Lawrence, 2011).

Where Do the Spring Constants Come From?

- Algorithms assume only neighbors in data space are connected by springs (sparse connectivity).
- Different algorithms suggest different values for the springs.
  - Laplacian Eigenmaps prescribe constant spring constants, or values from an RBF on the distances (Belkin and Niyogi, 2003).
  - Locally Linear Embedding considers spring constants that lead to optimal linear reconstruction of data points (Roweis and Saul, 2000).
  - Maximum Variance Unfolding prescribes spring constants that constrain interpoint latent distances to equal those in the data (Weinberger et al., 2004).
  - Maximum Entropy Unfolding fits by maximum likelihood (Lawrence, 2011).

Relationship to Laplacian Eigenmaps

- Laplacian eigenmaps (Belkin and Niyogi, 2003): graph Laplacian is specified across the data points.

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 $\mathbf{L}$.

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.

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

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

Smallest Eigenvalues of Laplacian

- Eigendecomposition of the covariance is

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

- Eigendecomposition of the Laplacian is

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

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)

Laplacian Eigenmaps

- Set parameters of Laplacian.

Laplacian Eigenmaps

- Set parameters of Laplacian.
- Perform CMDS on the implied matrix $\mathbf{K}$.

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.

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.

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.

Locally Linear Embedding

- The Laplacian should be constrained positive definite.

Locally Linear Embedding

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

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

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 $\mathbf{L1} = \mathbf{0}$.

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 $\mathbf{L1} = \mathbf{0}$.
  - i.e. $m_{i,i} = -\sum_{j \in \mathcal{N}(i)} m_{j,i}$

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 $\mathbf{L1} = \mathbf{0}$.
  - 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

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.

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 $\mathbf{M} = \mathbf{I} - \mathbf{W}$.

Point One

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

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

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

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).

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:

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

$\mathbf{Y}_{\setminus i}$ represents data other than the ith point.

Second Point

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

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

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

Second Point

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

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

  $\mathbf{Y}_{\setminus i}$ 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,

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

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.

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 $m_{i,i} = \sum_{j\in\mathcal{N}(i)} w_{j,i}$.

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 $m_{i,i} = 1$.

LLE Approximates MEU

- LLE is an approximation to maximum likelihood.

LLE Approximates MEU

- LLE is an approximation to maximum likelihood.
- Laplacian has factorized form.

LLE Approximates MEU

- LLE is an approximation to maximum likelihood.
- Laplacian has factorized form.
- Pseudolikelihood also allows for relatively quick parameter estimation.

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 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.

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.

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 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 $K = n - 1$.

Acyclic Locally Linear Embedding

- The pseudolikelihood is an approximation.
- Unless neighborhood in $\mathbf{M}$ is forced *acyclic*.
- Then $\mathbf{M}$ is a *Cholesky* factor and pseudolikelihood approximation is *exact*.
- Normalizer of Gaussian model *is*

$$\left(\frac{\left|\mathbf{M}\mathbf{M}^{\top}\right|}{2\pi}\right)^{\frac{p}{2}} = \left(\frac{m_{i,i}^{2}}{2\pi}\right)^{\frac{p}{2}}$$

- This gives a *very fast* approach to fitting MEU.
- We call this acyclic LLE.
- It does include PCA as special case.

Isomap and MEU

- Both MVU and MEU can be thought of as starting with a sparse graph of (squared) distances.

Isomap and 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.

Isomap and 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.

Isomap and 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).

Isomap and 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.

Simple Experiments

- Consider two real data sets.

Simple Experiments

- Consider two real data sets.
- We apply each of the spectral methods we have reviewed.

Simple Experiments

- Consider two real data sets.
- We apply each of the spectral methods we have reviewed.
- Apply the MEU framework.

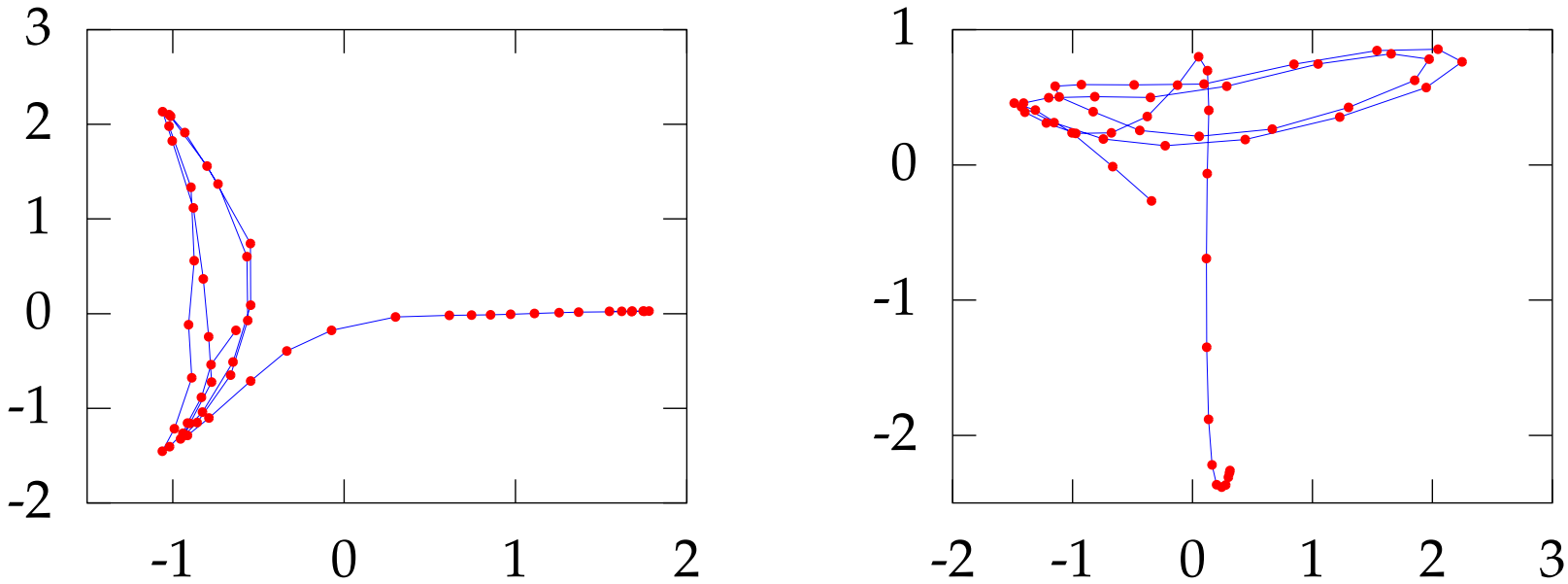
Simple Experiments

- 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.

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.
- First we consider Stick Man Data from before.

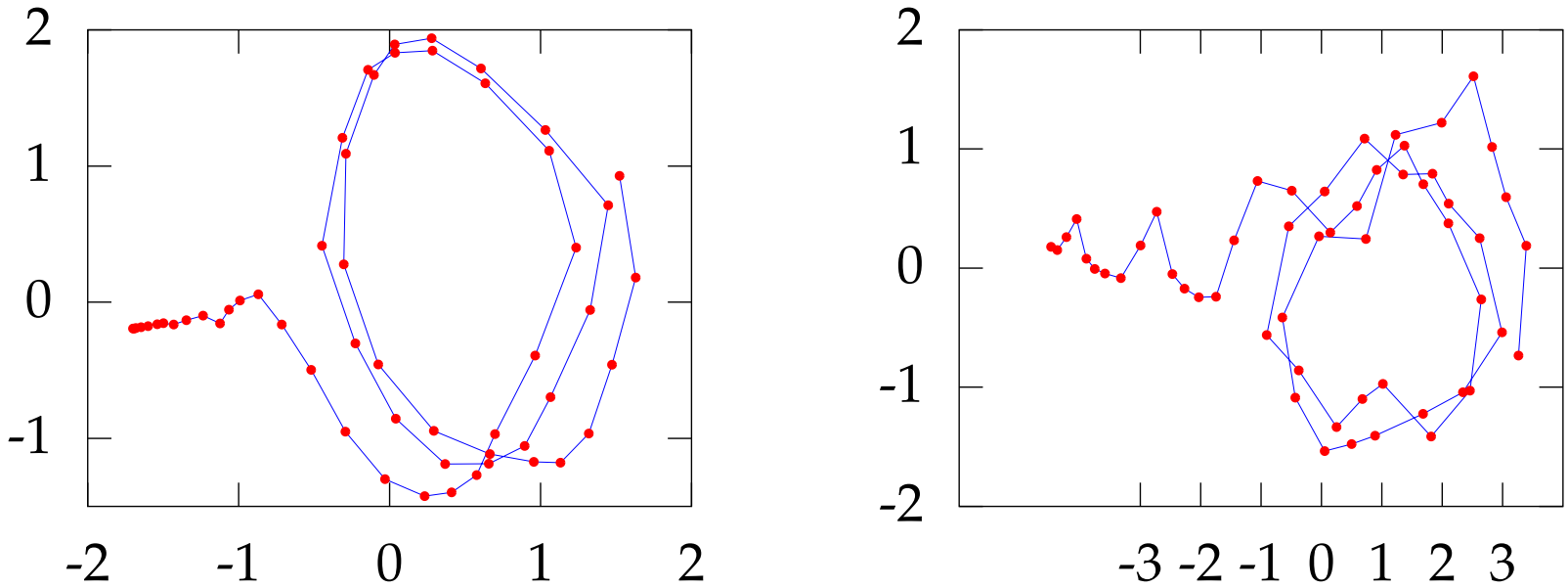
Laplacian Eigenmaps and LLE



(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.

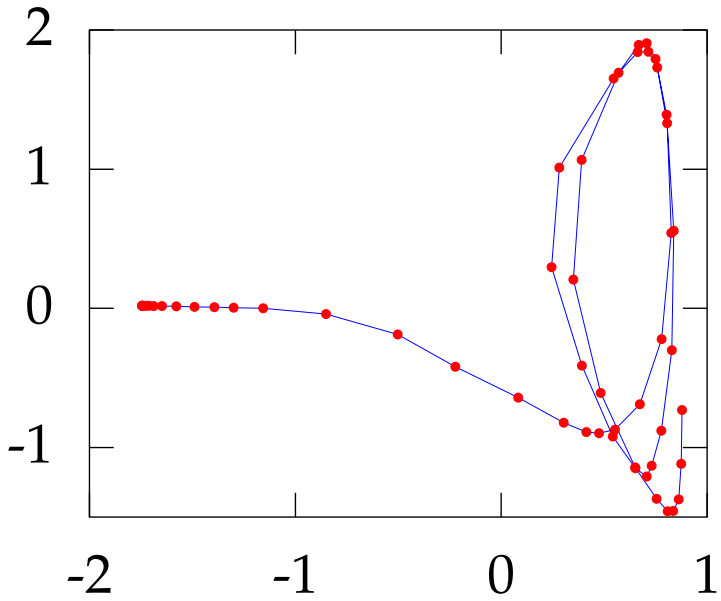
Isomap and MVU



(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

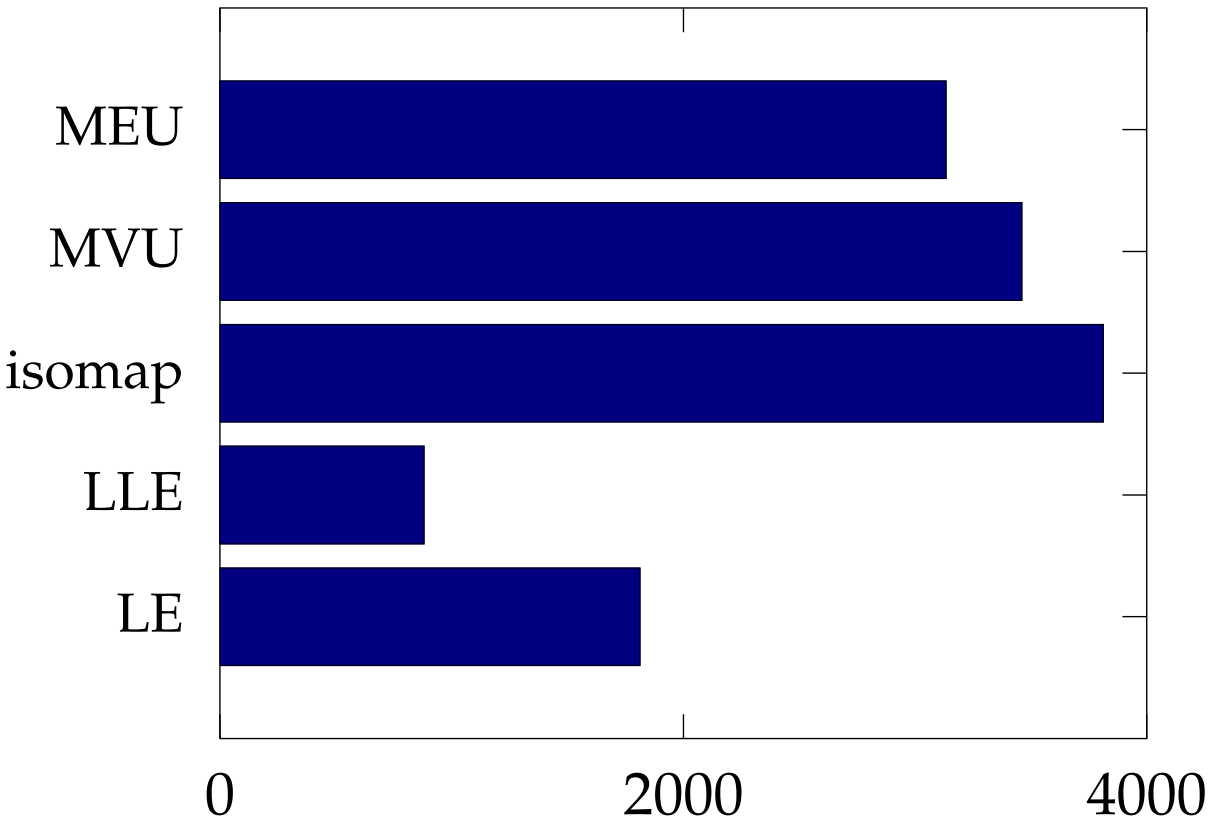


Figure: Model score for the different spectral approaches.

Robot Navigation Example

- Second data set: series of recordings from a robot as it traces a square path in a building.

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 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.

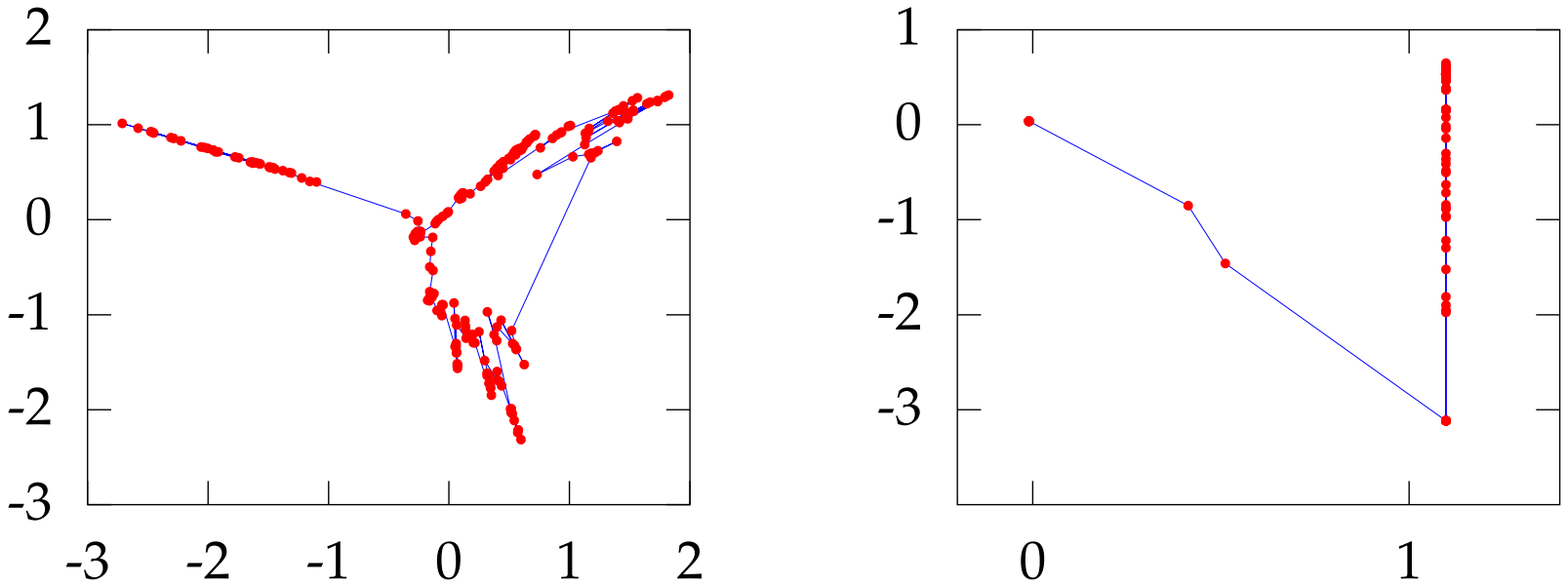
Robot Navigation Example

- 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".

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.

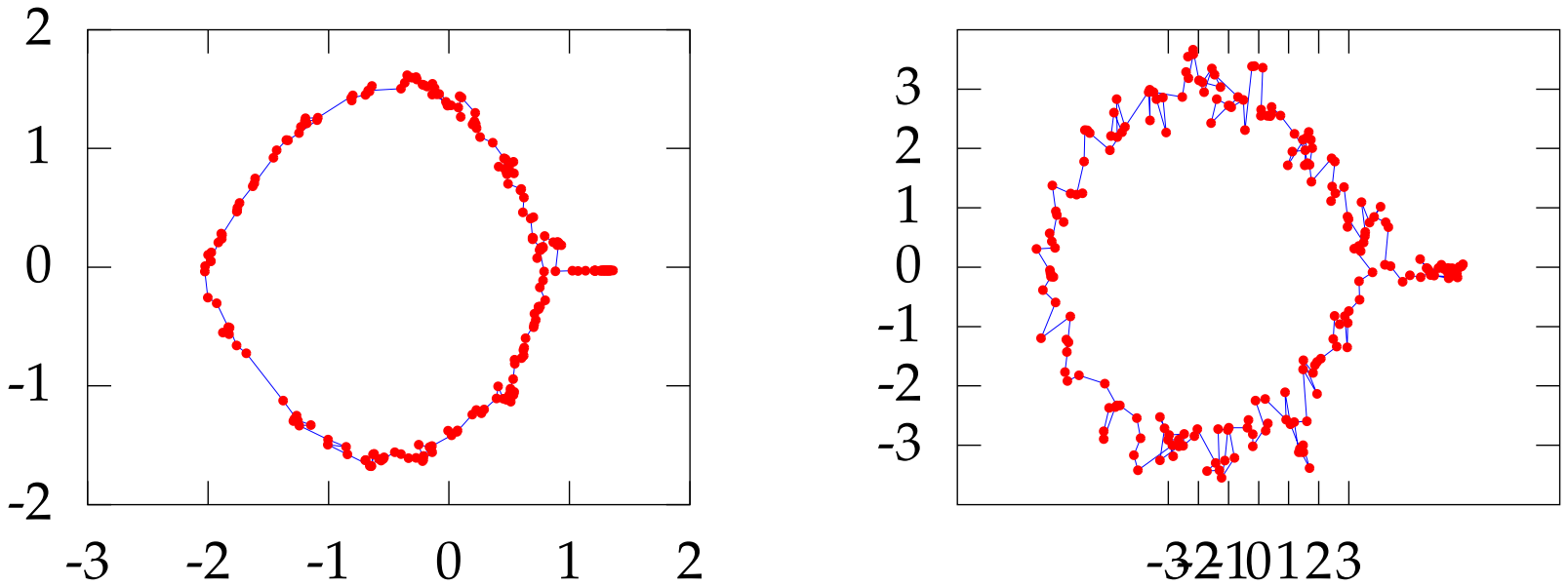
Laplacian Eigenmaps and LLE



(a) Laplacian Eigenmaps (b) Locally Linear Embedding

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

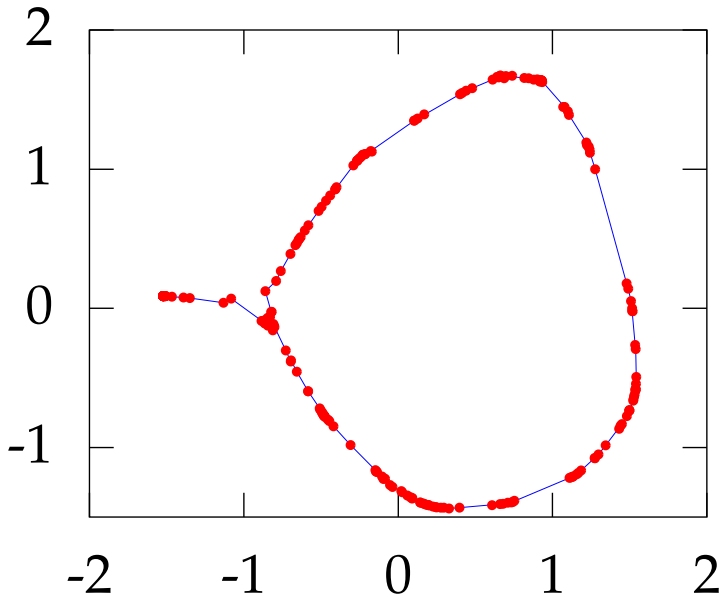
Isomap and MVU



(a) Isomap (b) MVU

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

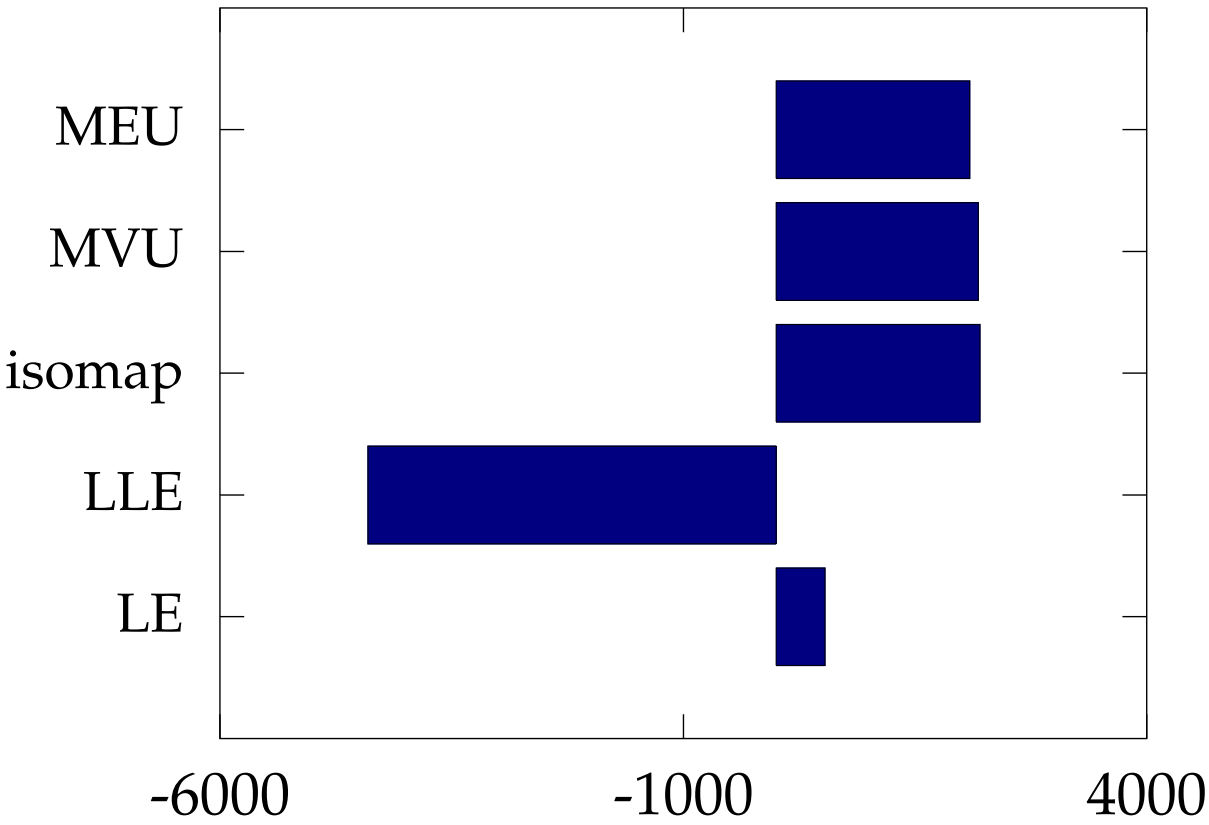


Figure: Model score for the different spectral approaches.

Linear Dimensionality Reduction

Linear Latent Variable Model

- Represent data, $\mathbf{Y}$, with a lower dimensional set of latent variables $\mathbf{X}$.
- Assume a linear relationship of the form

$$\mathbf{y}_{i,:} = \mathbf{W}\mathbf{x}_{i,:} + \boldsymbol{\epsilon}_{i,:},$$

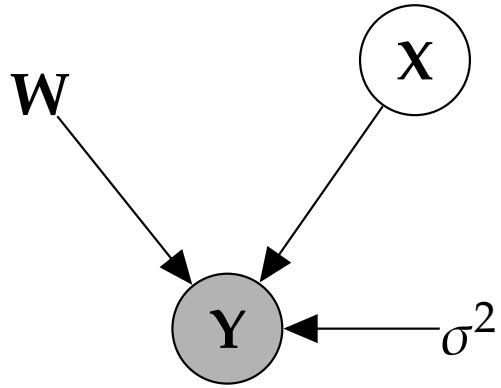
where

$$\boldsymbol{\epsilon}_{i,:} \sim \mathcal{N}\left(\mathbf{0}, \sigma^2\mathbf{I}\right).$$

Linear Latent Variable Model

Probabilistic PCA

- Define *linear-Gaussian relationship* between latent variables and data.

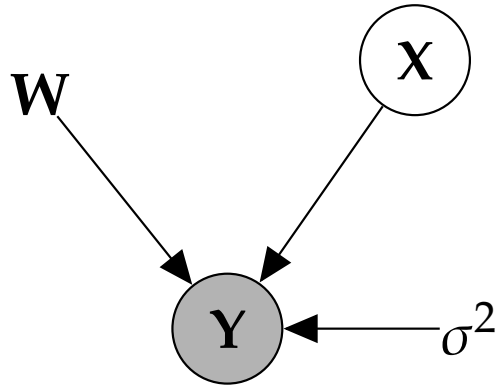


$$p\left(\mathbf{Y}|\mathbf{X},\mathbf{W}\right)=\prod_{i=1}^{n}\mathcal{N}\left(\mathbf{y}_{i,:}|\mathbf{W}\mathbf{x}_{i,:},\sigma^{2}\mathbf{I}\right)$$

Linear Latent Variable Model

Probabilistic PCA

- Define *linear-Gaussian relationship* between latent variables and data.
- **Standard** Latent variable approach:

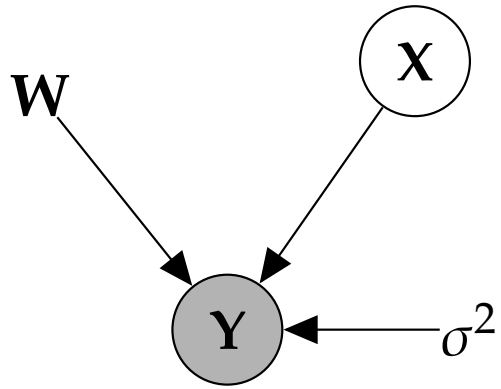


$$p\left(\mathbf{Y}|\mathbf{X},\mathbf{W}\right)=\prod_{i=1}^{n}\mathcal{N}\left(\mathbf{y}_{i,:}|\mathbf{W}\mathbf{x}_{i,:},\sigma^{2}\mathbf{I}\right)$$

Linear Latent Variable Model

Probabilistic PCA

- Define *linear-Gaussian relationship* between latent variables and data.
- **Standard** Latent variable approach:
  - Define Gaussian prior over *latent space*, $\mathbf{X}$.



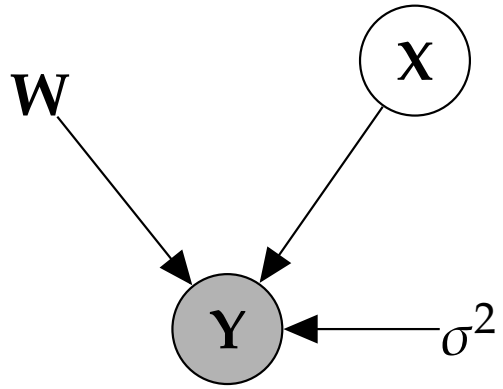
$$p(\mathbf{Y}|\mathbf{X},\mathbf{W}) = \prod_{i=1}^{n} \mathcal{N}\left(\mathbf{y}_{i,:}|\mathbf{W}\mathbf{x}_{i,:}, \sigma^2\mathbf{I}\right)$$

$$p(\mathbf{X}) = \prod_{i=1}^{n} \mathcal{N}\left(\mathbf{x}_{i,:}|\mathbf{0}, \mathbf{I}\right)$$

Linear Latent Variable Model

Probabilistic PCA

- Define *linear-Gaussian relationship* between latent variables and data.
- **Standard** Latent variable approach:
  - Define Gaussian prior over *latent space*, $\mathbf{X}$.
  - Integrate out *latent variables*.



$$p(\mathbf{Y}|\mathbf{X},\mathbf{W}) = \prod_{i=1}^{n} \mathcal{N}\left(\mathbf{y}_{i,:}|\mathbf{W}\mathbf{x}_{i,:}, \sigma^2\mathbf{I}\right)$$

$$p(\mathbf{X}) = \prod_{i=1}^{n} \mathcal{N}\left(\mathbf{x}_{i,:}|\mathbf{0}, \mathbf{I}\right)$$

$$p(\mathbf{Y}|\mathbf{W}) = \prod_{i=1}^{n} \mathcal{N}\left(\mathbf{y}_{i,:}|\mathbf{0}, \mathbf{W}\mathbf{W}^\top + \sigma^2\mathbf{I}\right)$$

Computation of the Marginal Likelihood

$$\mathbf{y}_{i,:} = \mathbf{W}\mathbf{x}_{i,:} + \boldsymbol{\epsilon}_{i,:}, \quad \mathbf{x}_{i,:} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}), \quad \boldsymbol{\epsilon}_{i,:} \sim \mathcal{N}\left(\mathbf{0}, \sigma^2\mathbf{I}\right)$$

Computation of the Marginal Likelihood

$$\mathbf{y}_{i,:} = \mathbf{W}\mathbf{x}_{i,:} + \boldsymbol{\epsilon}_{i,:}, \quad \mathbf{x}_{i,:} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}), \quad \boldsymbol{\epsilon}_{i,:} \sim \mathcal{N}\left(\mathbf{0}, \sigma^2\mathbf{I}\right)$$

$$\mathbf{W}\mathbf{x}_{i,:} \sim \mathcal{N}(\mathbf{0}, \mathbf{W}\mathbf{W}^\top),$$

Computation of the Marginal Likelihood

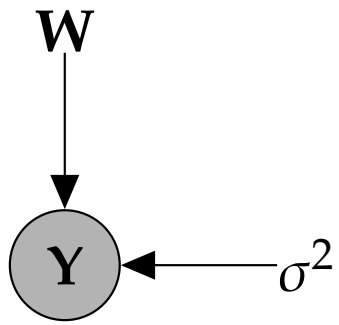
$$\mathbf{y}_{i,:} = \mathbf{W}\mathbf{x}_{i,:} + \boldsymbol{\epsilon}_{i,:}, \quad \mathbf{x}_{i,:} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}), \quad \boldsymbol{\epsilon}_{i,:} \sim \mathcal{N}\left(\mathbf{0}, \sigma^2\mathbf{I}\right)$$

$$\mathbf{W}\mathbf{x}_{i,:} \sim \mathcal{N}(\mathbf{0}, \mathbf{W}\mathbf{W}^\top),$$

$$\mathbf{W}\mathbf{x}_{i,:} + \epsilon_{i,:} \sim \mathcal{N}\left(\mathbf{0}, \mathbf{W}\mathbf{W}^\top + \sigma^2\mathbf{I}\right)$$

Linear Latent Variable Model II

Probabilistic PCA Max. Likelihood Soln (?)



$$p\left(\mathbf{Y}|\mathbf{W}\right) = \prod_{i=1}^{n} \mathcal{N}\left(\mathbf{y}_{i,:}|\mathbf{0}, \mathbf{W}\mathbf{W}^{\top} + \sigma^2\mathbf{I}\right)$$

Linear Latent Variable Model II

Probabilistic PCA Max. Likelihood Soln (?)

$$p\left(\mathbf{Y}|\mathbf{W}\right) = \prod_{i=1}^{n} \mathcal{N}\left(\mathbf{y}_{i,:}|\mathbf{0}, \mathbf{C}\right), \quad \mathbf{C} = \mathbf{W}\mathbf{W}^{\top} + \sigma^2\mathbf{I}$$

Linear Latent Variable Model II

Probabilistic PCA Max. Likelihood Soln (?)

$$p\left(\mathbf{Y}|\mathbf{W}\right) = \prod_{i=1}^{n} \mathcal{N}\left(\mathbf{y}_{i,:}|\mathbf{0}, \mathbf{C}\right), \quad \mathbf{C} = \mathbf{W}\mathbf{W}^{\top} + \sigma^2\mathbf{I}$$

$$\log p\left(\mathbf{Y}|\mathbf{W}\right) = -\frac{n}{2}\log|\mathbf{C}| - \frac{1}{2}\mathrm{tr}\left(\mathbf{C}^{-1}\mathbf{Y}^{\top}\mathbf{Y}\right) + \text{const.}$$

Linear Latent Variable Model II

Probabilistic PCA Max. Likelihood Soln (?)

$$p\left(\mathbf{Y}|\mathbf{W}\right) = \prod_{i=1}^{n} \mathcal{N}\left(\mathbf{y}_{i,:}|\mathbf{0}, \mathbf{C}\right), \quad \mathbf{C} = \mathbf{W}\mathbf{W}^{\top} + \sigma^2\mathbf{I}$$

$$\log p\left(\mathbf{Y}|\mathbf{W}\right) = -\frac{n}{2}\log|\mathbf{C}| - \frac{1}{2}\mathrm{tr}\left(\mathbf{C}^{-1}\mathbf{Y}^{\top}\mathbf{Y}\right) + \text{const.}$$

If $\mathbf{U}_q$ are first q principal eigenvectors of $n^{-1}\mathbf{Y}^{\top}\mathbf{Y}$ and the corresponding eigenvalues are $\mathbf{\Lambda}_q$,

Linear Latent Variable Model II

Probabilistic PCA Max. Likelihood Soln (?)

$$p\left(\mathbf{Y}|\mathbf{W}\right) = \prod_{i=1}^{n} \mathcal{N}\left(\mathbf{y}_{i,:}|\mathbf{0}, \mathbf{C}\right), \quad \mathbf{C} = \mathbf{W}\mathbf{W}^{\top} + \sigma^2\mathbf{I}$$

$$\log p\left(\mathbf{Y}|\mathbf{W}\right) = -\frac{n}{2}\log|\mathbf{C}| - \frac{1}{2}\mathrm{tr}\left(\mathbf{C}^{-1}\mathbf{Y}^{\top}\mathbf{Y}\right) + \text{const.}$$

If $\mathbf{U}_q$ are first q principal eigenvectors of $n^{-1}\mathbf{Y}^{\top}\mathbf{Y}$ and the corresponding eigenvalues are $\mathbf{\Lambda}_q$,

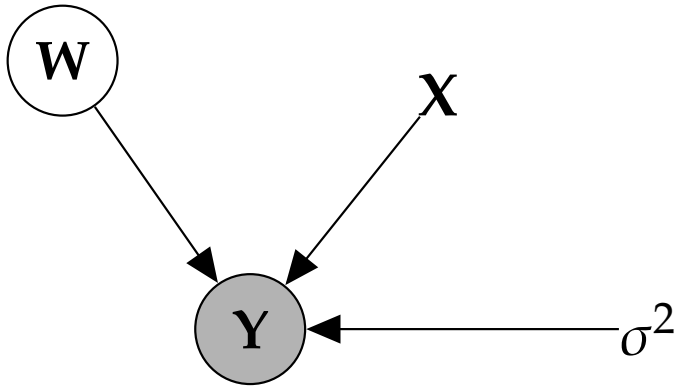
$$\mathbf{W} = \mathbf{U}_q\mathbf{L}\mathbf{R}^{\top}, \quad \mathbf{L} = \left(\mathbf{\Lambda}_q - \sigma^2\mathbf{I}\right)^{\frac{1}{2}}$$

where $\mathbf{R}$ is an arbitrary rotation matrix.

Linear Latent Variable Model III

Dual Probabilistic PCA

- Define *linear-Gaussian relationship* between latent variables and data.

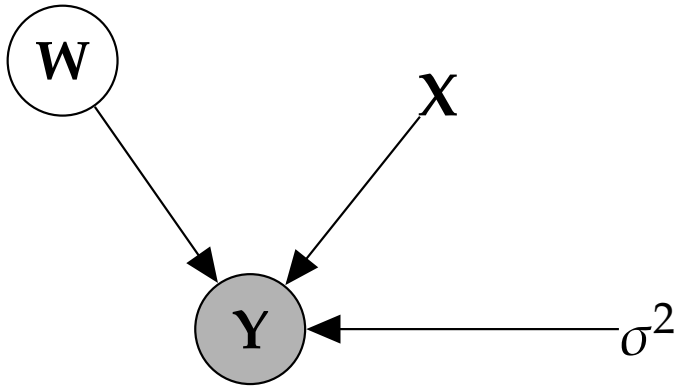


$$p\left(\mathbf{Y}|\mathbf{X},\mathbf{W}\right)=\prod_{i=1}^{n}\mathcal{N}\left(\mathbf{y}_{i,:}|\mathbf{W}\mathbf{x}_{i,:},\sigma^{2}\mathbf{I}\right)$$

Linear Latent Variable Model III

Dual Probabilistic PCA

- Define *linear-Gaussian relationship* between latent variables and data.
- **Novel** Latent variable approach:

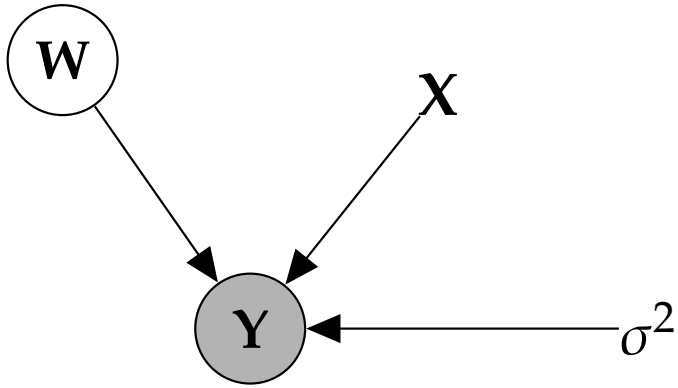


$$p\left(\mathbf{Y}|\mathbf{X},\mathbf{W}\right)=\prod_{i=1}^{n}\mathcal{N}\left(\mathbf{y}_{i,:}|\mathbf{W}\mathbf{x}_{i,:},\sigma^{2}\mathbf{I}\right)$$

Linear Latent Variable Model III

Dual Probabilistic PCA

- Define *linear-Gaussian relationship* between latent variables and data.
- **Novel** Latent variable approach:
  - Define Gaussian prior over *parameters*, $\mathbf{W}$.



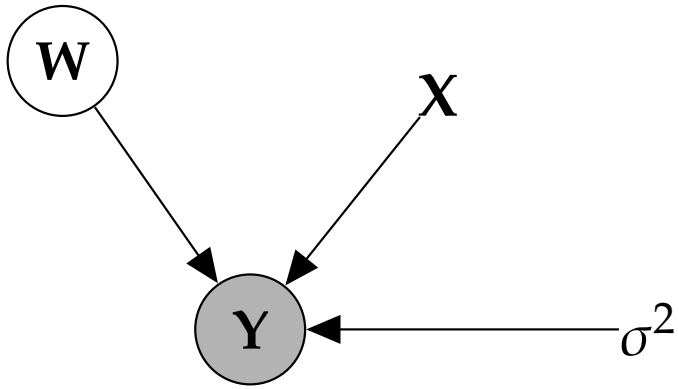
$$p(\mathbf{Y}|\mathbf{X},\mathbf{W}) = \prod_{i=1}^{n} \mathcal{N}\left(\mathbf{y}_{i,:}|\mathbf{W}\mathbf{x}_{i,:}, \sigma^2\mathbf{I}\right)$$

$$p(\mathbf{W}) = \prod_{i=1}^{p} \mathcal{N}\left(\mathbf{w}_{i,:}|\mathbf{0}, \mathbf{I}\right)$$

Linear Latent Variable Model III

Dual Probabilistic PCA

- Define *linear-Gaussian relationship* between latent variables and data.
- **Novel** Latent variable approach:
  - Define Gaussian prior over *parameters*, $\mathbf{W}$.
  - Integrate out *parameters*.



$$p(\mathbf{Y}|\mathbf{X},\mathbf{W}) = \prod_{i=1}^{n} \mathcal{N}\left(\mathbf{y}_{i,:}|\mathbf{W}\mathbf{x}_{i,:}, \sigma^2\mathbf{I}\right)$$

$$p(\mathbf{W}) = \prod_{i=1}^{p} \mathcal{N}\left(\mathbf{w}_{i,:}|\mathbf{0}, \mathbf{I}\right)$$

$$p(\mathbf{Y}|\mathbf{X}) = \prod_{j=1}^{p} \mathcal{N}\left(\mathbf{y}_{:,j}|\mathbf{0}, \mathbf{X}\mathbf{X}^\top + \sigma^2\mathbf{I}\right)$$

Computation of the Marginal Likelihood

$$\mathbf{y}_{:,j} = \mathbf{X}\mathbf{w}_{:,j} + \boldsymbol{\epsilon}_{:,j}, \quad \mathbf{w}_{:,j} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}), \quad \boldsymbol{\epsilon}_{i,:} \sim \mathcal{N}\left(\mathbf{0}, \sigma^2\mathbf{I}\right)$$

Computation of the Marginal Likelihood

$$\mathbf{y}_{:,j} = \mathbf{X}\mathbf{w}_{:,j} + \boldsymbol{\epsilon}_{:,j}, \quad \mathbf{w}_{:,j} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}), \quad \boldsymbol{\epsilon}_{i,:} \sim \mathcal{N}\left(\mathbf{0}, \sigma^2\mathbf{I}\right)$$

$$\mathbf{X}\mathbf{w}_{:,j} \sim \mathcal{N}(\mathbf{0}, \mathbf{X}\mathbf{X}^\top),$$

Computation of the Marginal Likelihood

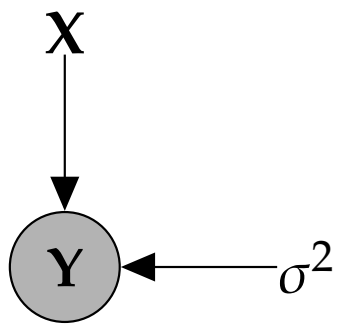
$$\mathbf{y}_{:,j} = \mathbf{X}\mathbf{w}_{:,j} + \boldsymbol{\epsilon}_{:,j}, \quad \mathbf{w}_{:,j} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}), \quad \boldsymbol{\epsilon}_{i,:} \sim \mathcal{N}\left(\mathbf{0}, \sigma^2\mathbf{I}\right)$$

$$\mathbf{X}\mathbf{w}_{:,j} \sim \mathcal{N}(\mathbf{0}, \mathbf{X}\mathbf{X}^\top),$$

$$\mathbf{X}\mathbf{w}_{:,j} + \epsilon_{:,j} \sim \mathcal{N}\left(\mathbf{0}, \mathbf{X}\mathbf{X}^\top + \sigma^2\mathbf{I}\right)$$

Linear Latent Variable Model IV

Dual Probabilistic PCA Max. Likelihood Soln (Lawrence, 2004, 2005)



$$p\left(\mathbf{Y}|\mathbf{X}\right) = \prod_{j=1}^{p} \mathcal{N}\left(\mathbf{y}_{:,j}|\mathbf{0}, \mathbf{X}\mathbf{X}^{\top} + \sigma^2\mathbf{I}\right)$$

Linear Latent Variable Model IV

Dual PPCA Max. Likelihood Soln (Lawrence, 2004, 2005)

$$p\left(\mathbf{Y}|\mathbf{X}\right) = \prod_{j=1}^{p} \mathcal{N}\left(\mathbf{y}_{:,j}|\mathbf{0}, \mathbf{K}\right), \quad \mathbf{K} = \mathbf{X}\mathbf{X}^{\top} + \sigma^2\mathbf{I}$$

Linear Latent Variable Model IV

PPCA Max. Likelihood Soln (?)

$$p\left(\mathbf{Y}|\mathbf{X}\right) = \prod_{j=1}^{p} \mathcal{N}\left(\mathbf{y}_{:,j}|\mathbf{0}, \mathbf{K}\right), \quad \mathbf{K} = \mathbf{X}\mathbf{X}^{\top} + \sigma^2\mathbf{I}$$

$$\log p\left(\mathbf{Y}|\mathbf{X}\right) = -\frac{p}{2}\log|\mathbf{K}| - \frac{1}{2}\mathrm{tr}\left(\mathbf{K}^{-1}\mathbf{Y}\mathbf{Y}^{\top}\right) + \text{const.}$$

Linear Latent Variable Model IV

PPCA Max. Likelihood Soln

$$p\left(\mathbf{Y}|\mathbf{X}\right) = \prod_{j=1}^{p} \mathcal{N}\left(\mathbf{y}_{:,j}|\mathbf{0}, \mathbf{K}\right), \quad \mathbf{K} = \mathbf{X}\mathbf{X}^{\top} + \sigma^2\mathbf{I}$$

$$\log p\left(\mathbf{Y}|\mathbf{X}\right) = -\frac{p}{2}\log|\mathbf{K}| - \frac{1}{2}\mathrm{tr}\left(\mathbf{K}^{-1}\mathbf{Y}\mathbf{Y}^{\top}\right) + \text{const.}$$

If $\mathbf{U}'_q$ are first q principal eigenvectors of $p^{-1}\mathbf{Y}\mathbf{Y}^{\top}$ and the corresponding eigenvalues are $\mathbf{\Lambda}_q$,

Linear Latent Variable Model IV

PPCA Max. Likelihood Soln

$$p\left(\mathbf{Y}|\mathbf{X}\right) = \prod_{j=1}^{p} \mathcal{N}\left(\mathbf{y}_{:,j}|\mathbf{0}, \mathbf{K}\right), \quad \mathbf{K} = \mathbf{X}\mathbf{X}^\top + \sigma^2\mathbf{I}$$

$$\log p\left(\mathbf{Y}|\mathbf{X}\right) = -\frac{p}{2}\log|\mathbf{K}| - \frac{1}{2}\mathrm{tr}\left(\mathbf{K}^{-1}\mathbf{Y}\mathbf{Y}^\top\right) + \text{const.}$$

If $\mathbf{U}'_q$ are first q principal eigenvectors of $p^{-1}\mathbf{Y}\mathbf{Y}^\top$ and the corresponding eigenvalues are $\mathbf{\Lambda}_q$,

$$\mathbf{X} = \mathbf{U}'_q\mathbf{L}\mathbf{R}^\top, \quad \mathbf{L} = \left(\mathbf{\Lambda}_q - \sigma^2\mathbf{I}\right)^{\frac{1}{2}}$$

where $\mathbf{R}$ is an arbitrary rotation matrix.

Linear Latent Variable Model IV

Dual PPCA Max. Likelihood Soln (Lawrence, 2004, 2005)

$$p\left(\mathbf{Y}|\mathbf{X}\right) = \prod_{j=1}^{p} \mathcal{N}\left(\mathbf{y}_{:,j}|\mathbf{0}, \mathbf{K}\right), \quad \mathbf{K} = \mathbf{X}\mathbf{X}^{\top} + \sigma^2\mathbf{I}$$

$$\log p\left(\mathbf{Y}|\mathbf{X}\right) = -\frac{p}{2}\log|\mathbf{K}| - \frac{1}{2}\mathrm{tr}\left(\mathbf{K}^{-1}\mathbf{Y}\mathbf{Y}^{\top}\right) + \text{const.}$$

If $\mathbf{U}'_q$ are first q principal eigenvectors of $p^{-1}\mathbf{Y}\mathbf{Y}^{\top}$ and the corresponding eigenvalues are $\mathbf{\Lambda}_q$,

$$\mathbf{X} = \mathbf{U}'_q\mathbf{L}\mathbf{R}^{\top}, \quad \mathbf{L} = \left(\mathbf{\Lambda}_q - \sigma^2\mathbf{I}\right)^{\frac{1}{2}}$$

where $\mathbf{R}$ is an arbitrary rotation matrix.

Linear Latent Variable Model IV

PPCA Max. Likelihood Soln (?)

$$p\left(\mathbf{Y}|\mathbf{W}\right) = \prod_{i=1}^{n} \mathcal{N}\left(\mathbf{y}_{i,:}|\mathbf{0}, \mathbf{C}\right), \quad \mathbf{C} = \mathbf{W}\mathbf{W}^{\top} + \sigma^2\mathbf{I}$$

$$\log p\left(\mathbf{Y}|\mathbf{W}\right) = -\frac{n}{2}\log|\mathbf{C}| - \frac{1}{2}\mathrm{tr}\left(\mathbf{C}^{-1}\mathbf{Y}^{\top}\mathbf{Y}\right) + \text{const.}$$

If $\mathbf{U}_q$ are first q principal eigenvectors of $n^{-1}\mathbf{Y}^{\top}\mathbf{Y}$ and the corresponding eigenvalues are $\mathbf{\Lambda}_q$,

$$\mathbf{W} = \mathbf{U}_q\mathbf{L}\mathbf{R}^{\top}, \quad \mathbf{L} = \left(\mathbf{\Lambda}_q - \sigma^2\mathbf{I}\right)^{\frac{1}{2}}$$

where $\mathbf{R}$ is an arbitrary rotation matrix.

Equivalence of Formulations

The Eigenvalue Problems are equivalent

- Solution for Probabilistic PCA (solves for the mapping)

$$\mathbf{Y}^\top \mathbf{Y}\mathbf{U}_q = \mathbf{U}_q \mathbf{\Lambda}_q \qquad \mathbf{W} = \mathbf{U}_q \mathbf{L}\mathbf{R}^\top$$

- Solution for Dual Probabilistic PCA (solves for the latent positions)

$$\mathbf{Y}\mathbf{Y}^\top \mathbf{U}'_q = \mathbf{U}'_q \mathbf{\Lambda}_q \qquad \mathbf{X} = \mathbf{U}'_q \mathbf{L}\mathbf{R}^\top$$

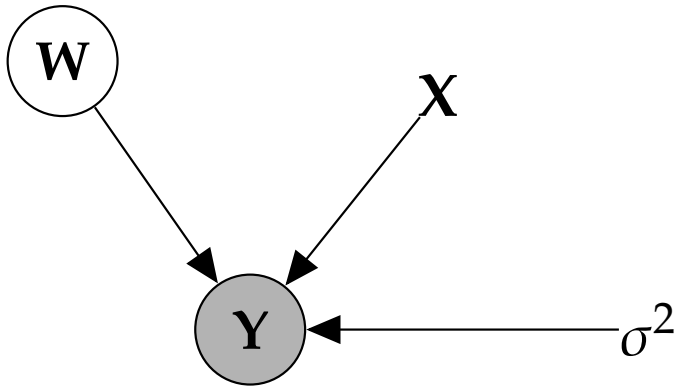
- Equivalence is from

$$\mathbf{U}_q = \mathbf{Y}^\top \mathbf{U}'_q \mathbf{\Lambda}_q^{-\frac{1}{2}}$$

Non-Linear Latent Variable Model

Dual Probabilistic PCA

- Define *linear-Gaussian relationship* between latent variables and data.
- **Novel** Latent variable approach:
  - Define Gaussian prior over *parameteters*, $\mathbf{W}$.
  - Integrate out *parameters*.



$$p(\mathbf{Y}|\mathbf{X},\mathbf{W}) = \prod_{i=1}^{n} \mathcal{N}\left(\mathbf{y}_{i,:}|\mathbf{W}\mathbf{x}_{i,:}, \sigma^2\mathbf{I}\right)$$

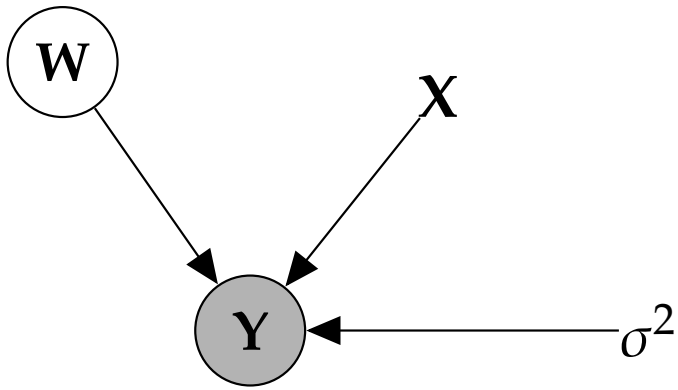
$$p(\mathbf{W}) = \prod_{i=1}^{p} \mathcal{N}\left(\mathbf{w}_{i,:}|\mathbf{0}, \mathbf{I}\right)$$

$$p(\mathbf{Y}|\mathbf{X}) = \prod_{j=1}^{p} \mathcal{N}\left(\mathbf{y}_{:,j}|\mathbf{0}, \mathbf{X}\mathbf{X}^\top + \sigma^2\mathbf{I}\right)$$

Non-Linear Latent Variable Model

Dual Probabilistic PCA

- Inspection of the marginal likelihood shows ...

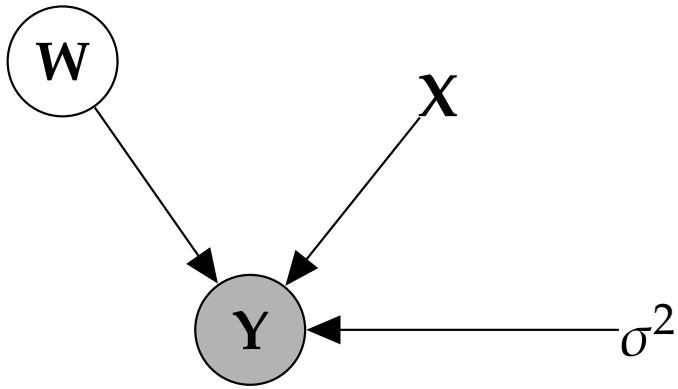


$$p(\mathbf{Y}|\mathbf{X}) = \prod_{j=1}^{p} \mathcal{N}\left(\mathbf{y}_{:,j}|\mathbf{0}, \mathbf{X}\mathbf{X}^\top + \sigma^2\mathbf{I}\right)$$

Non-Linear Latent Variable Model

Dual Probabilistic PCA

- Inspection of the marginal likelihood shows ...
  - The covariance matrix is a covariance function.



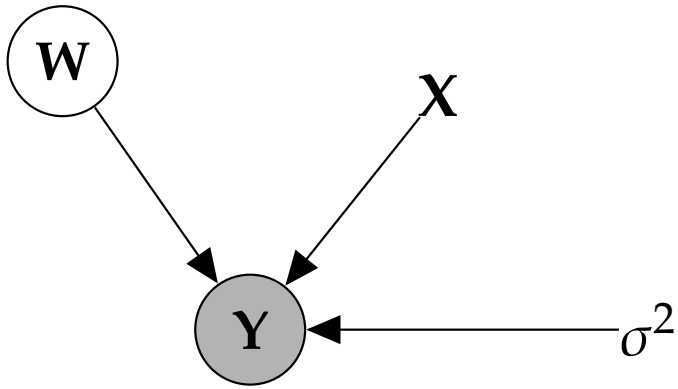
$$p\left(\mathbf{Y}|\mathbf{X}\right) = \prod_{j=1}^{p} \mathcal{N}\left(\mathbf{y}_{:,j}|\mathbf{0}, \mathbf{K}\right)$$

$$\mathbf{K} = \mathbf{X}\mathbf{X}^{\top} + \sigma^2\mathbf{I}$$

Non-Linear Latent Variable Model

Dual Probabilistic PCA

- Inspection of the marginal likelihood shows ...
  - The covariance matrix is a covariance function.
  - We recognise it as the 'linear kernel'.



$$p\left(\mathbf{Y}|\mathbf{X}\right) = \prod_{j=1}^{p} \mathcal{N}\left(\mathbf{y}_{:,j}|\mathbf{0}, \mathbf{K}\right)$$

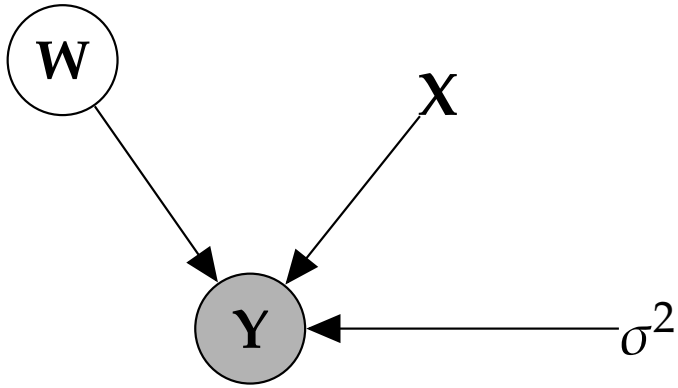
$$\mathbf{K} = \mathbf{X}\mathbf{X}^{\top} + \sigma^2\mathbf{I}$$

This is a product of Gaussian processes with linear kernels.

Non-Linear Latent Variable Model

Dual Probabilistic PCA

- Inspection of the marginal likelihood shows ...
  - The covariance matrix is a covariance function.
  - We recognise it as the 'linear kernel'.
  - We call this the Gaussian Process Latent Variable model (GP-LVM).



$$p(\mathbf{Y}|\mathbf{X}) = \prod_{j=1}^{p} \mathcal{N}\left(\mathbf{y}_{:,j}|\mathbf{0}, \mathbf{K}\right)$$

$\mathbf{K}$ =?

Replace linear kernel with non-linear kernel for non-linear model.

Non-linear Latent Variable Models

Exponentiated Quadratic (EQ) Covariance

- The EQ covariance has the form $k_{i,j} = k\left(\mathbf{x}_{i,:}, \mathbf{x}_{j,:}\right)$, where

$$k\left(\mathbf{x}_{i,:}, \mathbf{x}_{j,:}\right) = \alpha \exp\left(-\frac{\left\|\mathbf{x}_{i,:} - \mathbf{x}_{j,:}\right\|_2^2}{2\ell^2}\right).$$

- No longer possible to optimise wrt $\mathbf{X}$ via an eigenvalue problem.
- Instead find gradients with respect to $\mathbf{X}$, α, ℓ and σ^2 and optimise using conjugate gradients.

Discussion

- New perspective on dimensionality reduction algorithms based around maximum entropy.

Discussion

- New perspective on dimensionality reduction algorithms based around maximum entropy.
- GRFs and CMDS Unify Spectral Approaches in ML.

Stages of Spectral Dimensionality Reduction

- Our perspective shows there are three separate stages used in existing spectral dimensionality algorithms.

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.

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.

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.

Our Perspective

- Each step is somewhat orthogonal.
- Neighborhood relations need not come from nearest neighbors: can use structure learning.

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.

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.

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.

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.

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.

Summary: Spectral Approaches

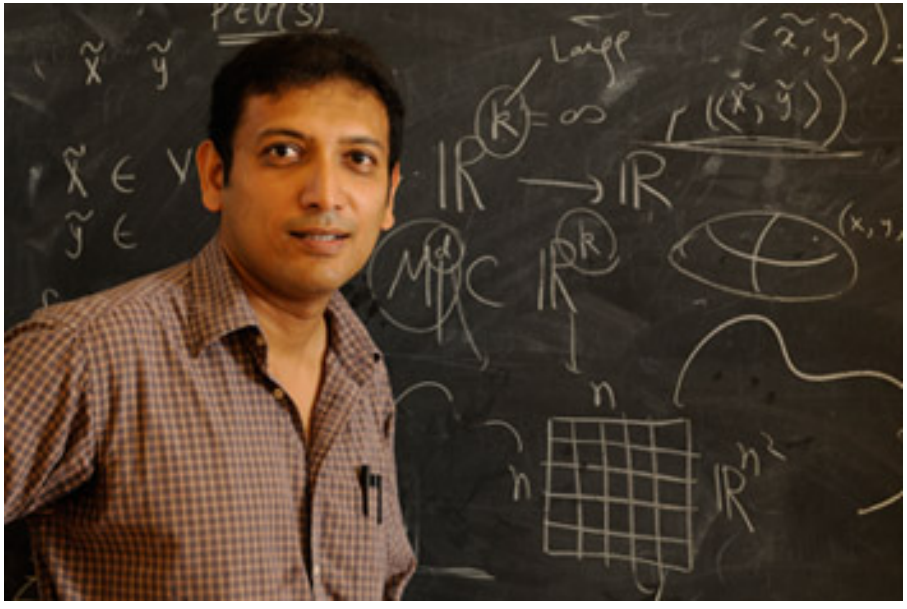
Good

- Unique optimum.

But

- Non trivial for dealing with missing data.
- Difficult to extend (*e.g.* temporal data) in a principled way.

Partha and Sam



References I

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

J. Bilmes, J. Malkin, X. Li, S. Harada, K. Kilanski, K. Kirchhoff, R. Wright, A. Subramanya, J. Landay, P. Dowden, and H. Chizeck. The vocal joystick. In *Proceedings of the IEEE Conference on Acoustics, Speech and Signal Processing*. IEEE, May 2006. To appear.

C. M. Bishop and G. D. James. Analysis of multiphase flows using dual-energy gamma densitometry and neural networks. *Nuclear Instruments and Methods in Physics Research*, A327:580–593, 1993. [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.

C. Kemp and J. B. Tenenbaum. The discovery of structural form. *Proc. Natl. Acad. Sci. USA*, 105(31), 2008.

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

N. D. Lawrence. Gaussian process models for visualisation of high dimensional data. In S. Thrun, L. Saul, and B. Schölkopf, editors, *Advances in Neural Information Processing Systems*, volume 16, pages 329–336, Cambridge, MA, 2004. MIT Press.

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.

N. D. Lawrence. A unifying probabilistic perspective for spectral dimensionality reduction. Technical report, University of Sheffield, [PDF].

References II

N. D. Lawrence. Spectral dimensionality reduction via maximum entropy. In G. Gordon and D. Dunson, editors, *Proceedings of the Fourteenth International Workshop on Artificial Intelligence and Statistics*, volume 15, pages 51–59, Fort Lauderdale, FL, USA, 11-13 April 2011. JMLR W&CP 15. [PDF]. Notable Paper.

N. D. Lawrence and A. J. Moore. Hierarchical Gaussian process latent variable models. In Z. Ghahramani, editor, *Proceedings of the International Conference in Machine Learning*, volume 24, pages 481–488. Omnipress, 2007. [Google Books] . [PDF].

N. D. Lawrence and J. Quiñonero Candela. Local distance preservation in the GP-LVM through back constraints. In W. Cohen and A. Moore, editors, *Proceedings of the International Conference in Machine Learning*, volume 23, pages 513–520. Omnipress, 2006. [Google Books] . [PDF].

D. Lowe and M. E. Tipping. Neuroscale: Novel topographic feature extraction with radial basis function networks. In M. C. Mozer, M. I. Jordan, and T. Petsche, editors, *Advances in Neural Information Processing Systems*, volume 9, pages 543–549, Cambridge, MA, 1997. MIT Press.

D. J. C. MacKay. *Information Theory, Inference and Learning Algorithms*. Cambridge University Press, Cambridge, U.K., 2003. [Google Books] .

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

T. P. Minka. Expectation propagation for approximate Bayesian inference. In J. S. Breese and D. Koller, editors, *Uncertainty in Artificial Intelligence*, volume 17, San Francisco, CA, 2001. Morgan Kauffman.

R. M. Neal. *Bayesian Learning for Neural Networks*. Springer, 1996. Lecture Notes in Statistics 118.

A. O'Hagan. Curve fitting and optimal design for prediction. *Journal of the Royal Statistical Society, B*, 40:1–42, 1978.

C. E. Rasmussen and C. K. I. Williams. *Gaussian Processes for Machine Learning*. MIT Press, Cambridge, MA, 2006. [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].

References III

M. K. Titsias and N. D. Lawrence. Bayesian Gaussian process latent variable model. In Y. W. Teh and D. M. Titterington, editors, *Proceedings of the Thirteenth International Workshop on Artificial Intelligence and Statistics*, volume 9, pages 844–851, Chia Laguna Resort, Sardinia, Italy, 13-16 May 2010. JMLR W&CP 9. [PDF].

J. M. Wang, D. J. Fleet, and A. Hertzmann. Gaussian process dynamical models. In Y. Weiss, B. Schölkopf, and J. C. Platt, editors, *Advances in Neural Information Processing Systems*, volume 18, Cambridge, MA, 2006. MIT Press.

J. M. Wang, D. J. Fleet, and A. Hertzmann. Gaussian process dynamical models for human motion. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(2):283–298, 2008. ISSN 0162-8828. [DOI].

L. A. Wasserman. *All of Statistics*. Springer-Verlag, New York, 2003. [Google Books] .

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. Regression with Gaussian processes. Paper presented at the *Mathematics of Neural Networks and Applications conference*, Oxford, UK, July 1995.

X. Zhu, J. Lafferty, and Z. Ghahramani. Semi-supervised learning: From Gaussian fields to Gaussian processes. Technical Report CMU-CS-03-175, Carnegie Mellon University, [PDF].