Spectral Curvature Clustering for Hybrid Linear Modeling

Guangliang Chen

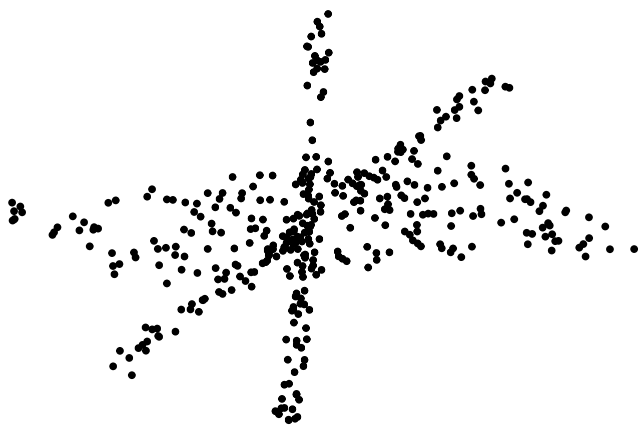
Applied Math Seminar

Duke University

September 28, 2009

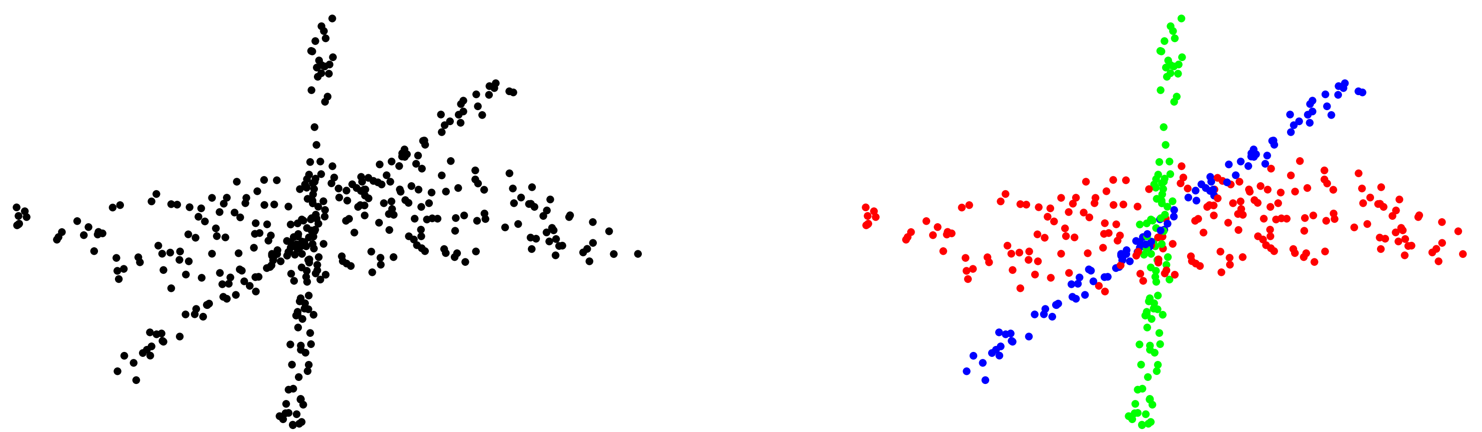
Hybrid Linear Modeling

- **Given**: $X = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N\} \subset \mathbb{R}^D$ sampled from K Borel probability measures supported around affine subspaces of dimensions $d_1, \ldots, d_K$



Hybrid Linear Modeling

- **Given**: $X = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N\} \subset \mathbb{R}^D$ sampled from K Borel probability measures supported around affine subspaces of dimensions $d_1, \ldots, d_K$



- **Goals**:
  - Determine K and $d_1, \ldots, d_K$ (if unknown)
  - Cluster data into subsets and model each subspace

Some Applications

- **Motion Segmentation**
  - Given trajectory vectors of pre-selected feature points along the image frames in a video sequence, cluster the trajectories according to the motions
- **Face Image Clustering**
  - Classify frontal images of several human subjects under different angles and illumination conditions
- **Temporal Video Segmentation**
  - Partition a long video sequence into multiple short segments containing different scenes

Outline of the Talk

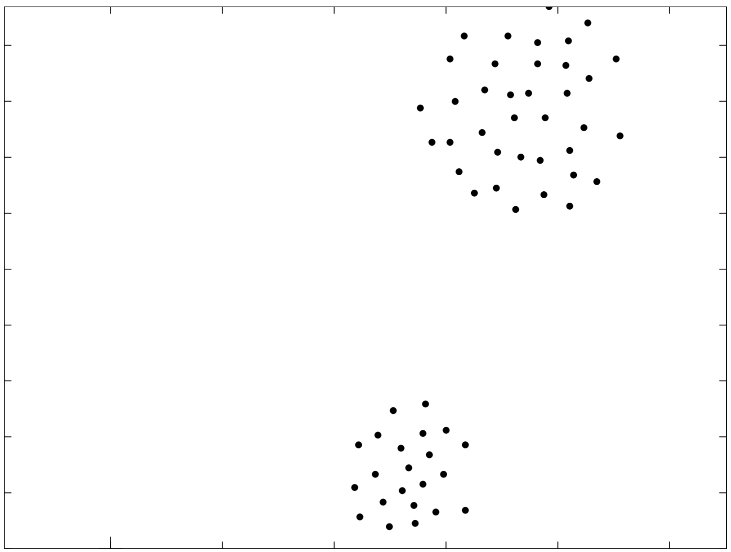
- Hybrid linear modeling via SCC
  - The SCC algorithm
  - Theoretical analysis
  - Numerical techniques
- Extension to multi-manifold modeling through
  - Kernelization
  - Localization
- Application to motion segmentation

Two Assumptions

- K and d_k are known
  - We want to focus on clustering and modeling
- d_k are all equal to d
  - Otherwise set $d = \max d_k$ and treat all subspaces as being d dimensional

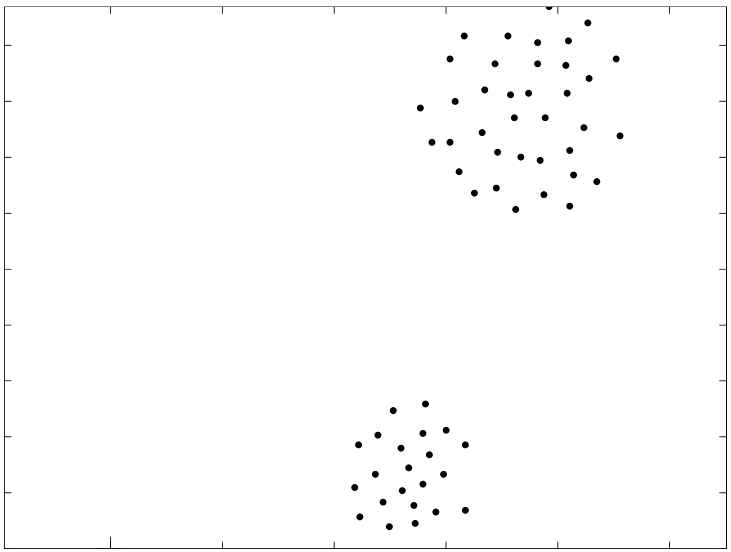
$d = 0$: Point Clouds

- **Example**



$d = 0$: Point Clouds

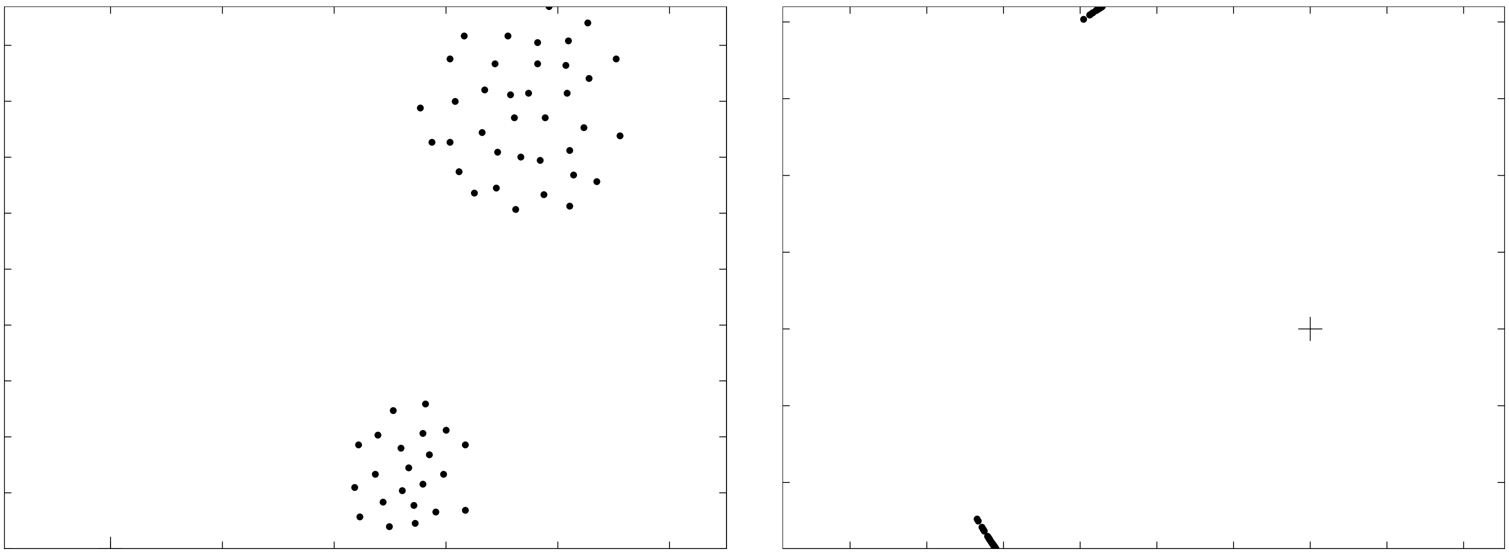
- **Example**



- **Spectral Clustering** (*Ng-Jordan-Weiss, NIPS 01'*)
  - Construct pairwise weights: $\mathbf{W}_{ij} = e^{-\|\mathbf{x}_i - \mathbf{x}_j\|^2/\sigma}$
  - Compute $\mathbf{W}$'s top K e.v.'s: $\mathbf{U} = [\mathbf{u}_1 \dots \mathbf{u}_K] \in \mathbb{R}^{N \times K}$ and map data to the **row** vectors of $\mathbf{U}$
  - Cluster data in the $\mathbf{U}$ space by Kmeans

$d = 0$: Point Clouds

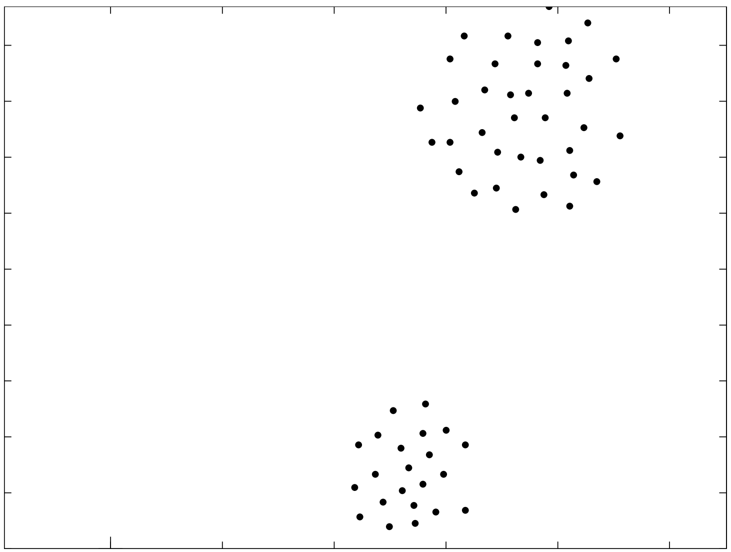
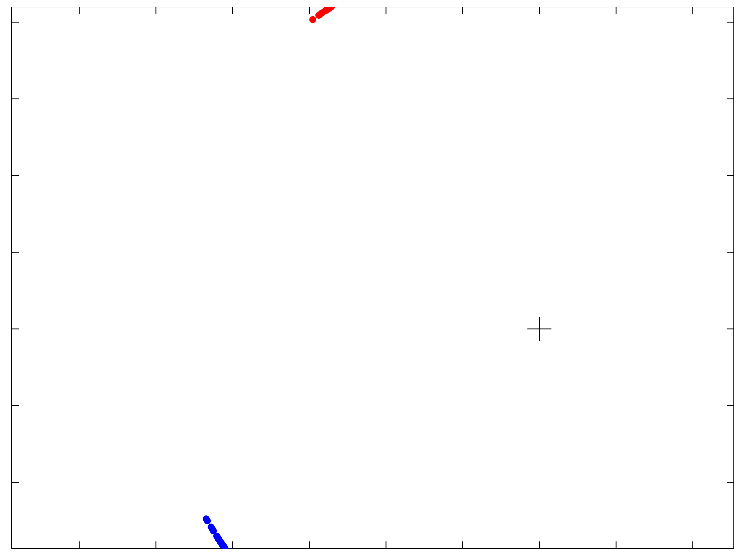
- **Example**



- **Spectral Clustering** (*Ng-Jordan-Weiss, NIPS 01'*)
  - Construct pairwise weights: $\mathbf{W}_{ij} = e^{-\|\mathbf{x}_i - \mathbf{x}_j\|^2/\sigma}$
  - Compute $\mathbf{W}$'s top K e.v.'s: $\mathbf{U} = [\mathbf{u}_1 \ldots \mathbf{u}_K] \in \mathbb{R}^{N \times K}$ and map data to the **row** vectors of $\mathbf{U}$
  - Cluster data in the $\mathbf{U}$ space by Kmeans

$d = 0$: Point Clouds

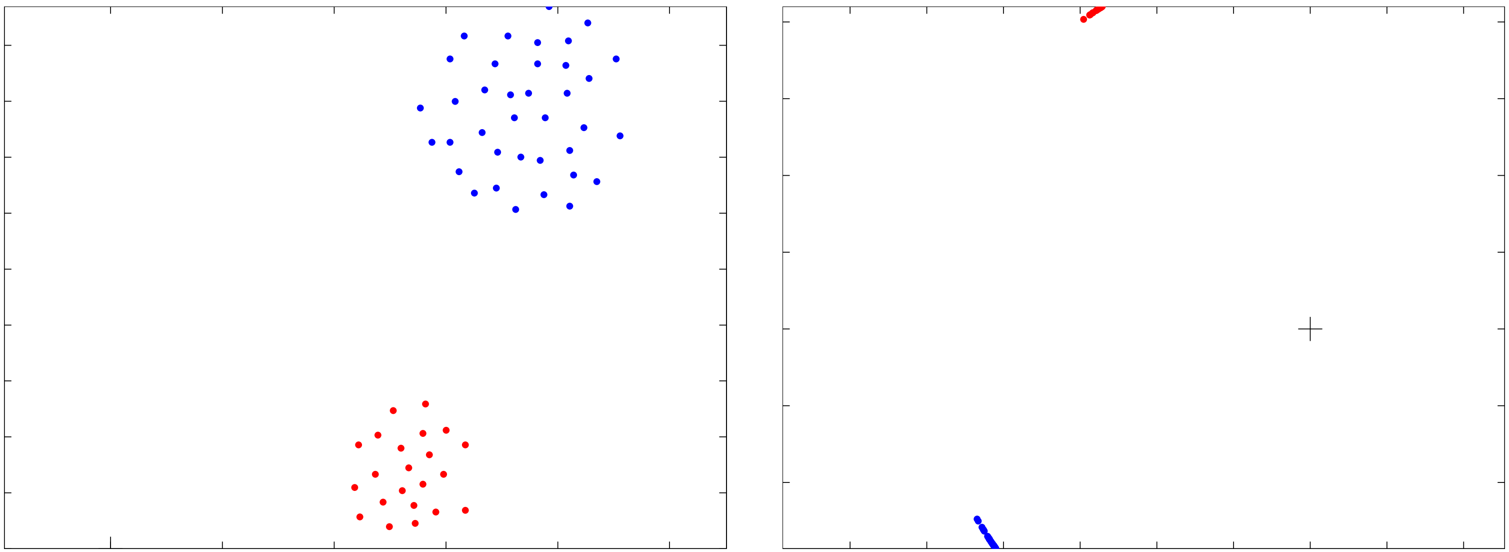
- **Example**

- **Spectral Clustering** (*Ng-Jordan-Weiss, NIPS 01'*)
  - Construct pairwise weights: $\mathbf{W}_{ij} = e^{-\|\mathbf{x}_i - \mathbf{x}_j\|^2/\sigma}$
  - Compute $\mathbf{W}$'s top K e.v.'s: $\mathbf{U} = [\mathbf{u}_1 \ldots \mathbf{u}_K] \in \mathbb{R}^{N\times K}$ and map data to the **row** vectors of $\mathbf{U}$
  - Cluster data in the $\mathbf{U}$ space by Kmeans

$d = 0$: Point Clouds

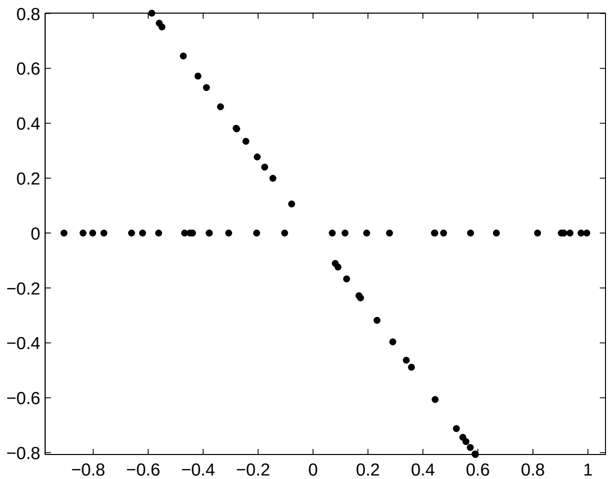
- **Example**



- **Spectral Clustering** (*Ng-Jordan-Weiss, NIPS 01'*)
  - Construct pairwise weights: $\mathbf{W}_{ij} = e^{-\|\mathbf{x}_i - \mathbf{x}_j\|^2/\sigma}$
  - Compute $\mathbf{W}$'s top K e.v.'s: $\mathbf{U} = [\mathbf{u}_1 \dots \mathbf{u}_K] \in \mathbb{R}^{N \times K}$ and map data to the **row** vectors of $\mathbf{U}$
  - Cluster data in the $\mathbf{U}$ space by Kmeans

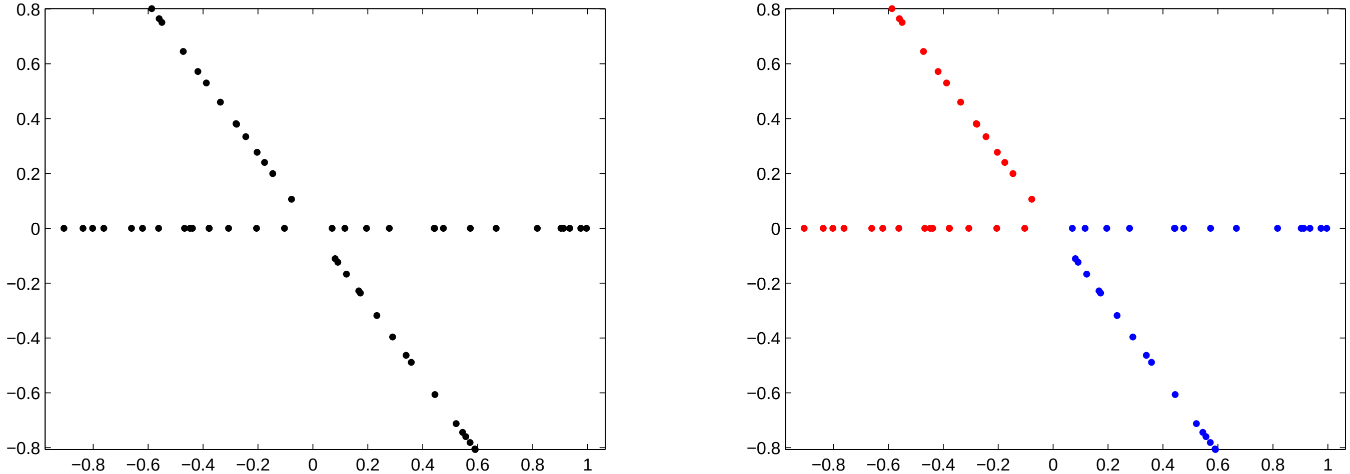
When $d \geq 1$

Consider the 2 lines clustering problem ($d = 1$):



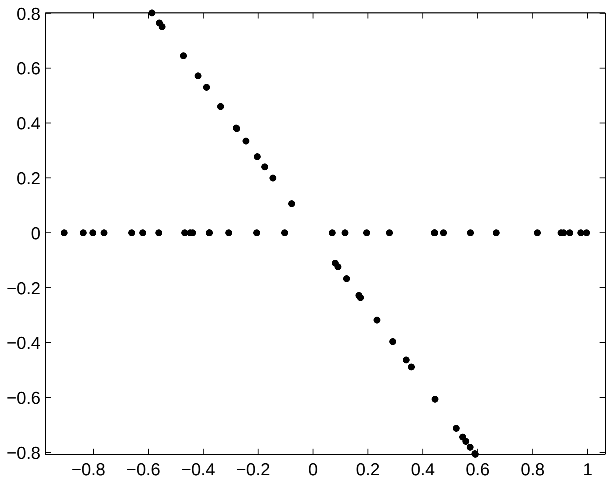
When $d \geq 1$

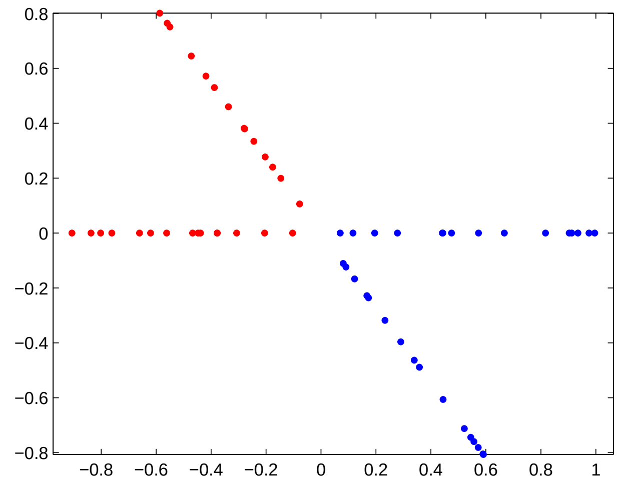
Clusters found by spectral clustering:



When $d \geq 1$

Clusters found by spectral clustering:





Conclusions: cannot compute weights using only

- distance
- 2 points

Multi-way Clustering

- **Idea** (for d-planes clustering, $d \geq 0$):
  - Assign an affinity measure to any $d+2$ points, using e.g., volume, LS error
  - *Process* the resulting $(d+2)$-way affinity tensor to cluster data

Multi-way Clustering

- **Idea** (for d-planes clustering, $d \geq 0$):
  - Assign an affinity measure to any $d+2$ points, using e.g., volume, LS error
  - *Process* the resulting $(d+2)$-way affinity tensor to cluster data
- **Important questions**:
  - What are good multiwise affinities?
  - How to process affinity tensors both theoretically and practically (N^{d+2} affinities!)?
  - How to rigorously justify such an algorithm?

Multi-way Clustering

- **Idea** (for d-planes clustering, $d \geq 0$):
  - Assign an affinity measure to any $d+2$ points, using e.g., volume, LS error
  - *Process* the resulting $(d+2)$-way affinity tensor to cluster data
- **Important questions**:
  - What are good multiwise affinities?
  - How to process affinity tensors both theoretically and practically (N^{d+2} affinities!)?
  - How to rigorously justify such an algorithm?
- **Previous work**:
  Govindu (CVPR 05'), Agarwal et al. (CVPR 05', ICML 06'), Shashua et al. (ECCV 06')

Polar Curvature

- **Definition**: For any $Z = \{\mathbf{z}_1, \ldots, \mathbf{z}_{d+2}\} \subset \mathbb{R}^D$, and the $(d+1)$-simplex $\mathcal{S}$, the *polar curvature* of Z is

$$c_{\mathrm{p}}^2(Z) := \mathrm{diam}(Z)^2 \cdot \sum \mathrm{psin}_{\mathbf{z}_i}(Z)^2,$$

where $\mathrm{psin}_{\mathbf{z}_i}$ is the polar sine at $\mathbf{z}_i, 1 \leq i \leq d+2$:

$$\mathrm{psin}_{\mathbf{z}_i}(Z) := \frac{(d+1)! \cdot \mathrm{V}_{d+1}(\mathcal{S})}{\prod_{j \neq i} \|\mathbf{z}_j - \mathbf{z}_i\|}.$$

- **Two special cases**:
  - $d = 0$: $\mathrm{psin}_{\mathbf{z}_i}(Z) \equiv 1, c_{\mathrm{p}}(Z) = \|\mathbf{z}_1 - \mathbf{z}_2\|$
  - $d = 1$: $\mathrm{psin}_{\mathbf{z}_i}(Z) = \sin_{\mathbf{z}_i}(Z)$

Polar Curvature - cont'd

- **Main property** (*Lerman & Whitehouse, 2008*):

$$\int c_{\mathrm{p}}^2(Z)\, d\mu^{d+2}(Z) \approx \text{d-dim (squared) LS error of } \mu$$

It generalizes the following identity ($d = 0$):

$$\int \|\mathbf{x}-\mathbf{y}\|^2\, d\mu(\mathbf{x})d\mu(\mathbf{y}) = 2 \cdot \int \|\mathbf{x}-\bar{\mathbf{x}}\|^2\, d\mu(\mathbf{x})$$

Polar Curvature - cont'd

- **Main property** (*Lerman & Whitehouse, 2008*):

$$\int c_{\mathrm{p}}^2(Z)\, d\mu^{d+2}(Z) \approx \text{d-dim (squared) LS error of } \mu$$

It generalizes the following identity ($d = 0$):

$$\int \|\mathbf{x}-\mathbf{y}\|^2\, d\mu(\mathbf{x})d\mu(\mathbf{y}) = 2 \cdot \int \|\mathbf{x}-\bar{\mathbf{x}}\|^2\, d\mu(\mathbf{x})$$

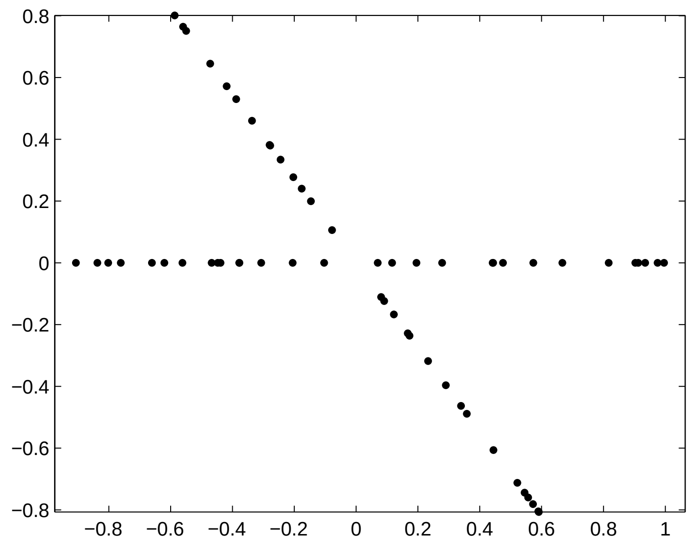
- **Other possible curvatures** (with same property):
  - c_{LS}: d-dim least squares error of Z
  - c_h: minimum height from any vertex to its opposite face in the $(d+1)$-simplex Z

The Polar Tensor

- Affinity tensor $\mathcal{A}_{\mathsf{p}} \in \mathbb{R}^{N\times\cdots\times N}$ (of order $d+2$):

$$\mathcal{A}_{\mathsf{p}}(i_1, \ldots, i_{d+2}) = e^{-c_{\mathsf{p}}^2(\mathbf{x}_{i_1},\ldots,\mathbf{x}_{i_{d+2}})/\sigma}$$

- For clean subspaces and $\sigma \to 0$:
  - $\mathcal{A}_{\mathsf{p}} \approx 1$ within an underlying cluster
  - $\mathcal{A}_{\mathsf{p}} \approx 0$ between clusters

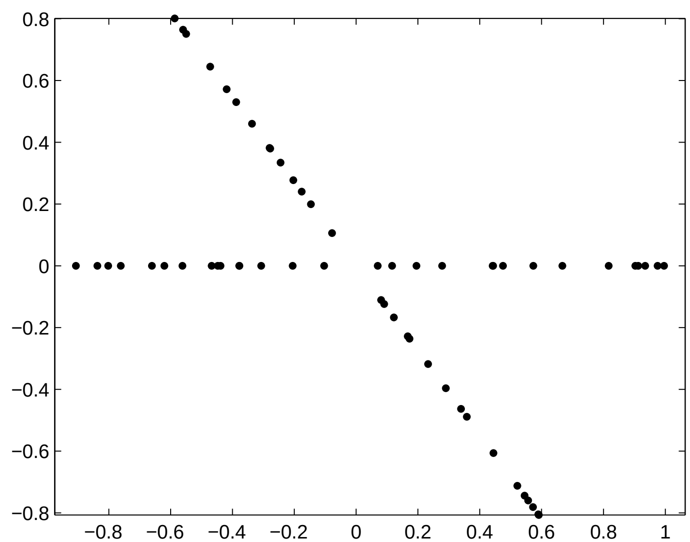


From Affinities to Weights

- (Govindu 05') Construct pairwise weights from affinities:

$$\mathbf{W}_{ik} = \sum_{\forall j_1, \ldots, j_{d+1}} \mathcal{A}_{\mathsf{p}}(i, j_1, \ldots, j_{d+1}) \cdot \mathcal{A}_{\mathsf{p}}(k, j_1, \ldots, j_{d+1})$$

- Within-cluster weights: large;
  between-cluster weights: small



Theoretical SCC (TSCC)

- Compute affinity tensor $\mathcal{A}_{\mathrm{p}}$
- Form weight matrix $\mathbf{W}$ from $\mathcal{A}_{\mathrm{p}}$
- Apply spectral clustering
  - Extract top K eigenvectors of $\mathbf{W}$: $\mathbf{U} = [\mathbf{u}_1 \dots \mathbf{u}_K]$
  - Apply Kmeans to the row vectors of $\mathbf{U}$

Another Interpretation of TSCC

- Define an affinity matrix by unfolding the tensor $\mathcal{A}_\mathsf{p}$:

$$\mathbf{A}(i,:) = \{\mathcal{A}_\mathsf{p}(i, j_1, \ldots, j_{d+1}) \mid \forall j_1, \ldots, j_{d+1}\} \in \mathbb{R}^{N^{d+1}},$$

  containing closeness information between point i and all $(d+1)$-tuples of points (spanning d-planes)

- Apply SVD (reduce dimension) + Kmeans (cluster data)

  (*Note that* $\mathbf{W} = \mathbf{A} \cdot \mathbf{A}'$)

Another Interpretation of TSCC

- Define an affinity matrix by unfolding the tensor $\mathcal{A}_\mathsf{p}$:

$$\mathbf{A}(i,:) = \{\mathcal{A}_\mathsf{p}(i, j_1, \ldots, j_{d+1}) \mid \forall j_1, \ldots, j_{d+1}\} \in \mathbb{R}^{N^{d+1}},$$

  containing closeness information between point i and all $(d+1)$-tuples of points (spanning d-planes)

- Apply SVD (reduce dimension) + Kmeans (cluster data)

  (*Note that* $\mathbf{W} = \mathbf{A} \cdot \mathbf{A}'$)

- **Important observation**:
  Enough to have some representative $(d+1)$-tuples from each cluster, thus possible to reduce N^{d+1} to $O(K)$!

Two-step Justification

- Step 1: we assume an ideal tensor:

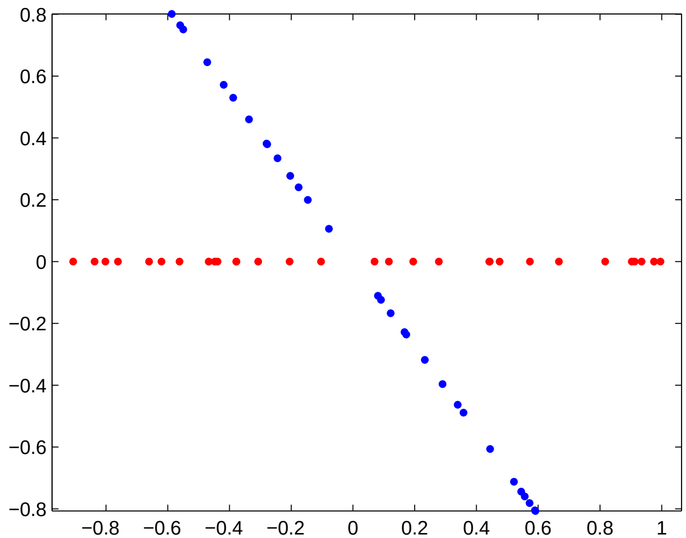
  $\widetilde{\mathcal{A}}(i_1, \ldots, i_{d+2}) = 1$ within-cluster and 0 between-clusters,

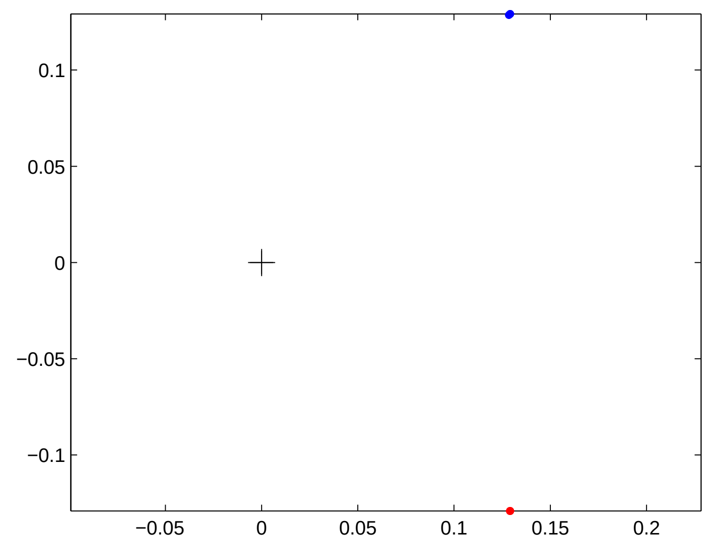
  and show that SCC works perfectly with $\widetilde{\mathcal{A}}$

  (*can be closely approximated for clean data* + $\sigma \to 0$)
- Step 2: we examine more general tensors by using operator perturbation theory

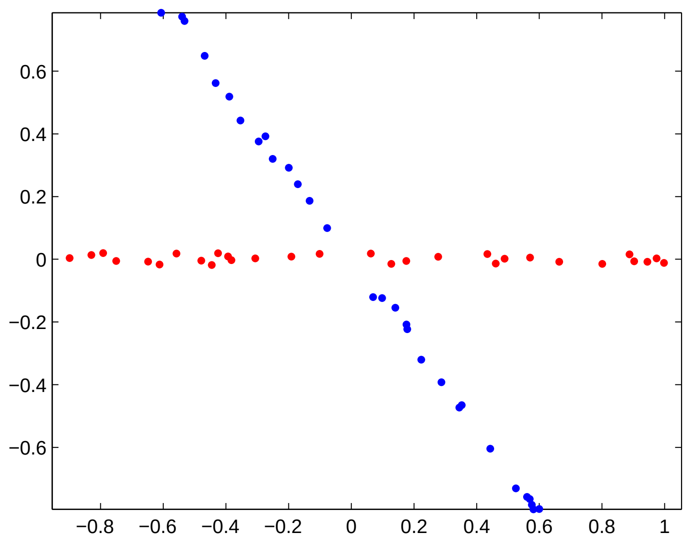
Step 1 - The Ideal Case

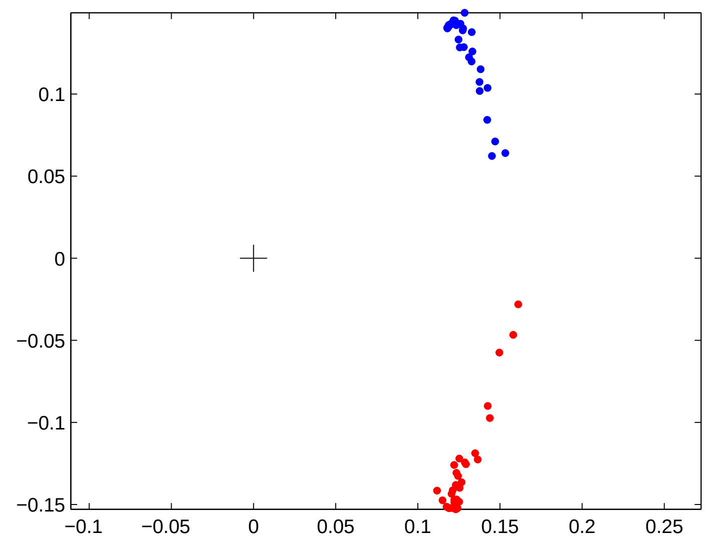
- The matrix $\mathbf{W}$ is block-diagonal, each block corresponding to an underlying cluster
- The rows of $\mathbf{U}$ are exactly K orthonormal vectors, each representing a true cluster





Step 2 - More General Cases





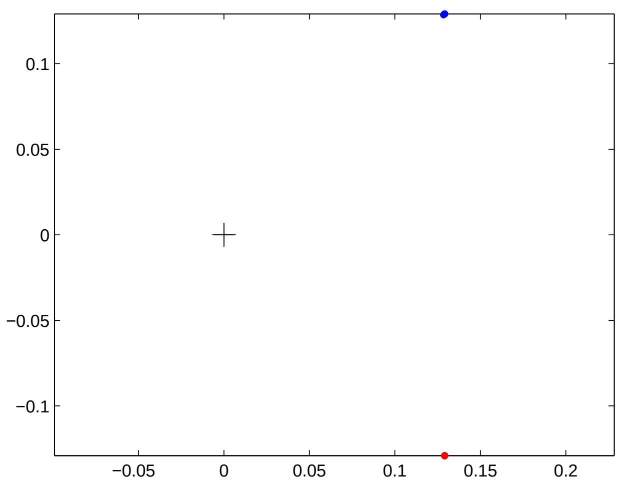
Step 2 - Goodness of Clustering

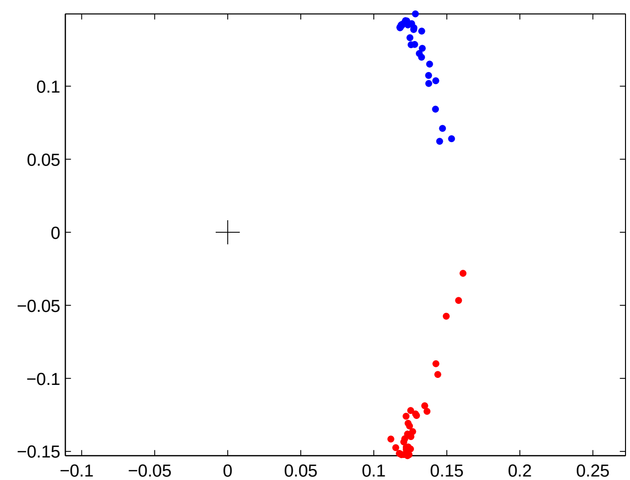
Total variance of true clusters in the $\mathbf{U}$ space:

$$\mathrm{TV}(\mathbf{U}) := \sum_{k} \sum_{i \in \mathrm{I}_k} \| \mathbf{u}^{(i)} - \mathbf{c}^{(k)} \|_2^2$$

in which

- $\mathbf{u}^{(i)}$: i-th row of $\mathbf{U}$
- $\mathbf{c}^{(k)}$: center of underlying cluster I_k





Step 2 - Perturbation Analysis

Let $\mathcal{A}$ be a general affinity tensor, and define

$$\mathcal{E} := \mathcal{A} - \widetilde{\mathcal{A}},$$

then TSCC (with $\mathcal{A}$) achieves that

$$\mathrm{TV}(\mathbf{U}) \lesssim N^{-(d+2)} \|\mathcal{E}\|_F^2$$

Step 2 - Probabilistic Analysis

Let μ_k: underlying measure of the k-th cluster, and

$$\alpha := \frac{1}{\sigma^2} \sum_k c_{\mathrm{p}}^2(\mu_k) + c_{\mathrm{inc'd}}(\mu_1, \ldots, \mu_K; \sigma),$$

in which

- $c_{\mathrm{p}}^2(\mu_k) = \int c_{\mathrm{p}}^2(Z)\, d\mu_k^{d+2}(Z)$: flatness measure of μ_k
- $c_{\mathrm{inc'd}}$: separation measure between all μ_k

Then using the polar tensor $\mathcal{A}_{\mathrm{p}}$, TSCC achieves that

$$\mathrm{TV}(\mathbf{U}) \lesssim \alpha \quad \text{with high probability}$$

Numerical Challenges

- Complexity is high:
  - Cannot store/compute $\mathcal{A}_{\mathsf{p}}$ (N^{d+2} elements)
  - Even harder to compute $\mathbf{W}$ ($O(N^{d+3})$ time)

$$\mathbf{W}_{ik} = \sum_{\forall j_1,\ldots,j_{d+1}} \mathcal{A}_{\mathsf{p}}(i, j_1, \ldots, j_{d+1}) \cdot \mathcal{A}_{\mathsf{p}}(k, j_1, \ldots, j_{d+1})$$

Numerical Challenges

- Complexity is high:
  - Cannot store/compute $\mathcal{A}_{\mathsf{p}}$ (N^{d+2} elements)
  - Even harder to compute $\mathbf{W}$ ($O(N^{d+3})$ time)

$$\mathbf{W}_{ik} = \sum_{\forall j_1,\ldots,j_{d+1}} \mathcal{A}_{\mathsf{p}}(i, j_1, \ldots, j_{d+1}) \cdot \mathcal{A}_{\mathsf{p}}(k, j_1, \ldots, j_{d+1})$$

- $\mathcal{A}_{\mathsf{p}}$ contains a sensitive parameter σ (which should be data-dependent); not clear how to efficiently select its optimal value

Problem with Uniform Sampling

- **Idea**: (Govindu 05') estimate $\mathbf{W}$ by randomly sampling a constant c number of $(d+1)$-tuples of points:

$$\mathbf{W}_{ik} \approx \sum_{t=1}^{c} \mathcal{A}_{\mathsf{p}}(i, j_1^{(t)}, \ldots, j_{d+1}^{(t)}) \cdot \mathcal{A}_{\mathsf{p}}(k, j_1^{(t)}, \ldots, j_{d+1}^{(t)})$$

Problem with Uniform Sampling

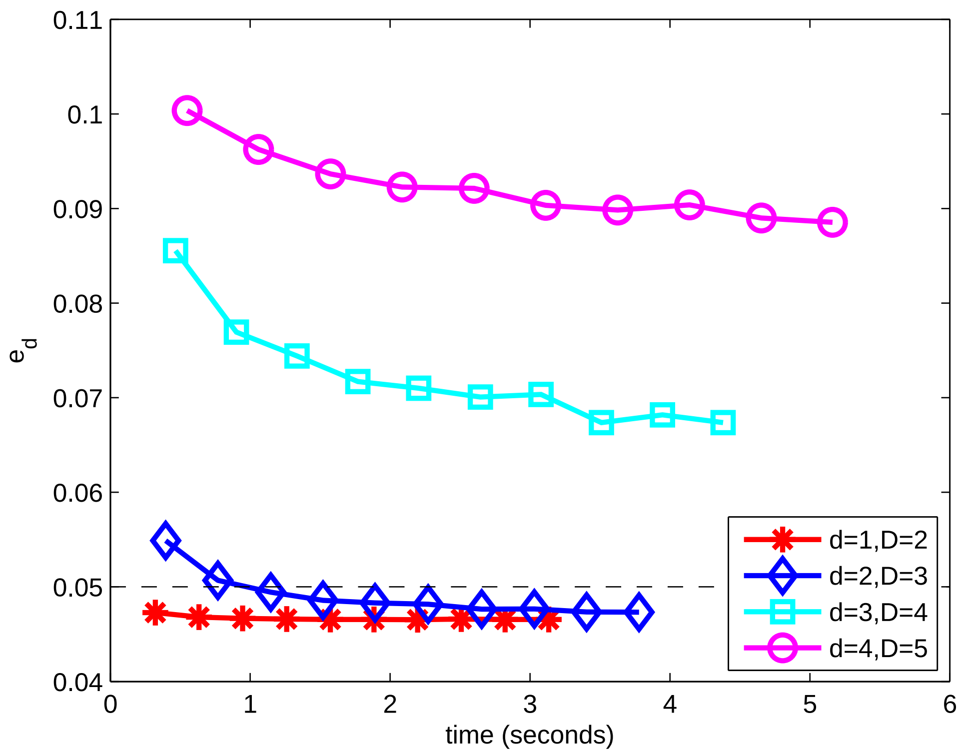
- **Idea**: (Govindu 05') estimate $\mathbf{W}$ by randomly sampling a constant c number of $(d+1)$-tuples of points:

$$\mathbf{W}_{ik} \approx \sum_{t=1}^{c} \mathcal{A}_{\mathsf{p}}(i, j_1^{(t)}, \ldots, j_{d+1}^{(t)}) \cdot \mathcal{A}_{\mathsf{p}}(k, j_1^{(t)}, \ldots, j_{d+1}^{(t)})$$

- **Performance**: Does not work for large d

Problem with Uniform Sampling

$K = 3$ d-dim linear subspaces in $\mathbb{R}^D$, $N = 100K$.
Use $c = 1 \cdot N, \ldots, 10 \cdot N$ independently.
Plot of error (averaged over 500 experiments) against time



Fixing Uniform Sampling

- Why would uniform sampling fail?

$$\mathbf{W}_{ik} \approx \sum_{t=1}^{c} \mathcal{A}_{\mathsf{p}}(i, j_1^{(t)}, \ldots, j_{d+1}^{(t)}) \cdot \mathcal{A}_{\mathsf{p}}(k, j_1^{(t)}, \ldots, j_{d+1}^{(t)})$$

Fixing Uniform Sampling

- Why would uniform sampling fail?

$$\mathbf{W}_{ik} \approx \sum_{t=1}^{c} \mathcal{A}_{\mathsf{p}}(i, j_1^{(t)}, \ldots, j_{d+1}^{(t)}) \cdot \mathcal{A}_{\mathsf{p}}(k, j_1^{(t)}, \ldots, j_{d+1}^{(t)})$$

  - Only tuples of points $(j_1^{(t)}, \ldots, j_{d+1}^{(t)})$ sampled from same underlying clusters matter
(those mixed are useless and even harmful!)

Fixing Uniform Sampling

- Why would uniform sampling fail?

$$\mathbf{W}_{ik} \approx \sum_{t=1}^{c} \mathcal{A}_{\mathsf{p}}(i, j_1^{(t)}, \ldots, j_{d+1}^{(t)}) \cdot \mathcal{A}_{\mathsf{p}}(k, j_1^{(t)}, \ldots, j_{d+1}^{(t)})$$

  - Only tuples of points $(j_1^{(t)}, \ldots, j_{d+1}^{(t)})$ sampled from same underlying clusters matter
(those mixed are useless and even harmful!)
  - However, the probability of sampling a “good” tuple is $1/K^{d+1}$ (small when K, d large)

Fixing Uniform Sampling

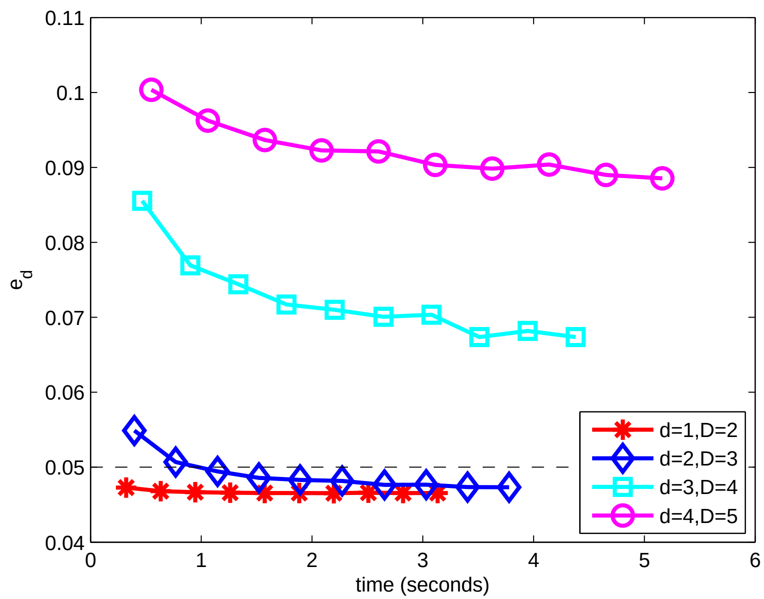
- Why would uniform sampling fail?
  - Only tuples of points $(j_1^{(t)}, \ldots, j_{d+1}^{(t)})$ sampled from same underlying clusters matter
    (those mixed are useless and even harmful!)
  - However, the probability of sampling a “good” tuple is $1/K^{d+1}$ (small when K, d large)
- One way to fix this is to sample *iteratively*:
  Fix $c = 100 \cdot K$.
  - 0th iteration: estimate clusters by uniform sampling
  - Subsequent iterations: sample tuples from same clusters obtained in the preceding iteration

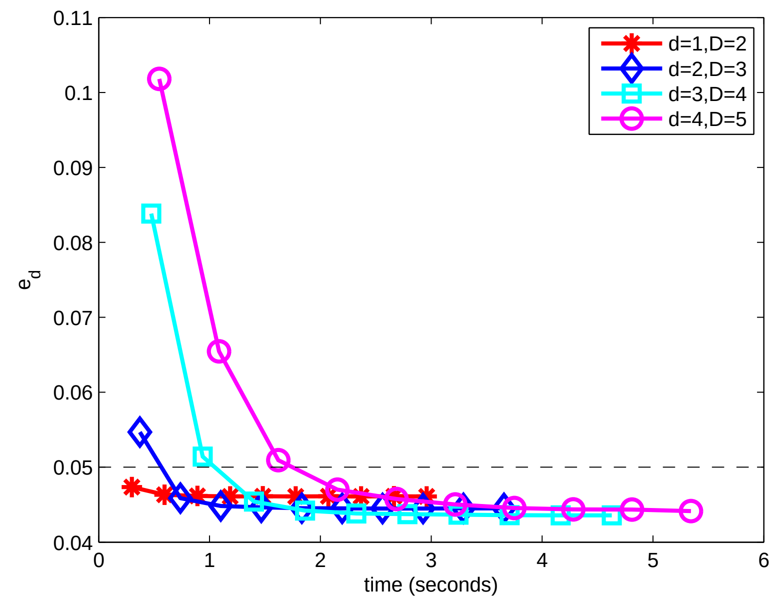
Fixing Uniform Sampling

- Why would uniform sampling fail?
  - Only tuples of points $(j_1^{(t)}, \ldots, j_{d+1}^{(t)})$ sampled from same underlying clusters matter
    (those mixed are useless and even harmful!)
  - However, the probability of sampling a “good” tuple is $1/K^{d+1}$ (small when K, d large)
- One way to fix this is to sample *iteratively*:
  Fix $c = 100 \cdot K$.
  - 0th iteration: estimate clusters by uniform sampling
  - Subsequent iterations: sample tuples from same clusters obtained in the preceding iteration
- Other ways: sample from local regions

Uniform vs Iterative

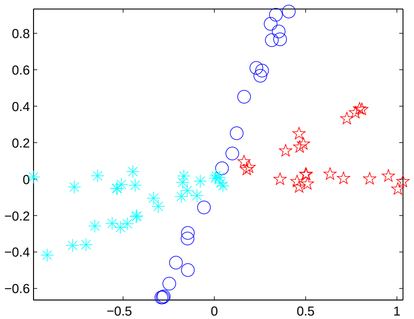
- Uniform Sampling: $c = 1 \cdot N, \ldots, 10 \cdot N$, respectively
- Iterative Sampling: $c = N$ fixed in each iteration



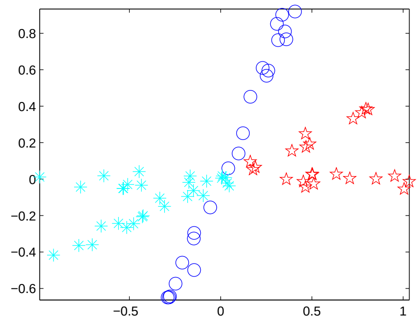


The Parameter σ

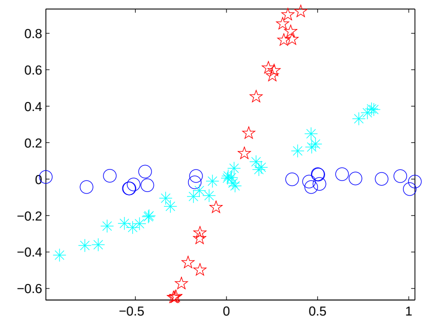
Common practice is to try several manually selected values, which is inefficient and often fails:



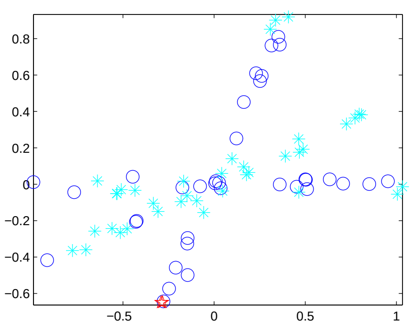
(a) $\sigma = 1$



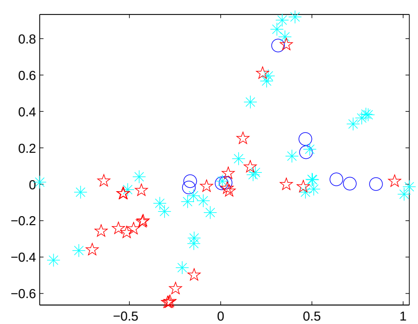
(b) $\sigma = 0.5$



(c) $\sigma = 0.0573$



(d) $\sigma = 0.01$



(e) $\sigma = 0.001$

The Parameter σ

We automatically infer it from data.

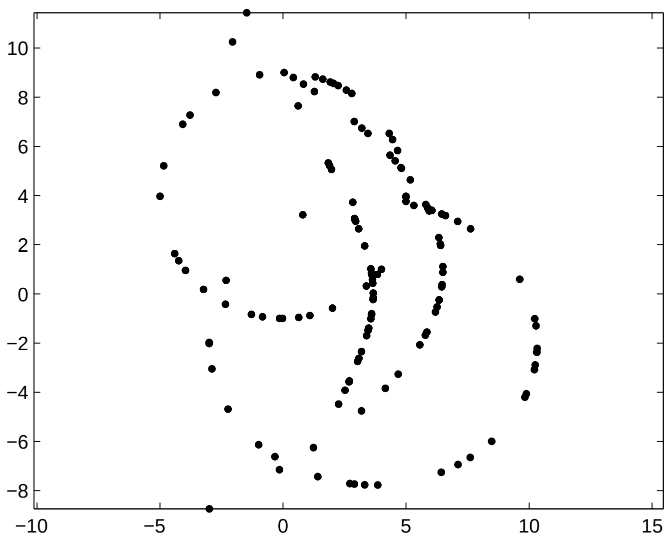
- Minimality of $\alpha := \frac{1}{\sigma^2}\sum_k c_{\mathrm{p}}^2(\mu_k) + c_{\mathrm{inc'd}}(\mu_1, \ldots, \mu_K, \sigma)$ implies that optimal σ, σ_{opt}, should have upper and lower bounds
- If we divide all the computed curvatures into two groups:
  - (small) curvatures of within-cluster points
  - (large) curvatures of between-cluster points,

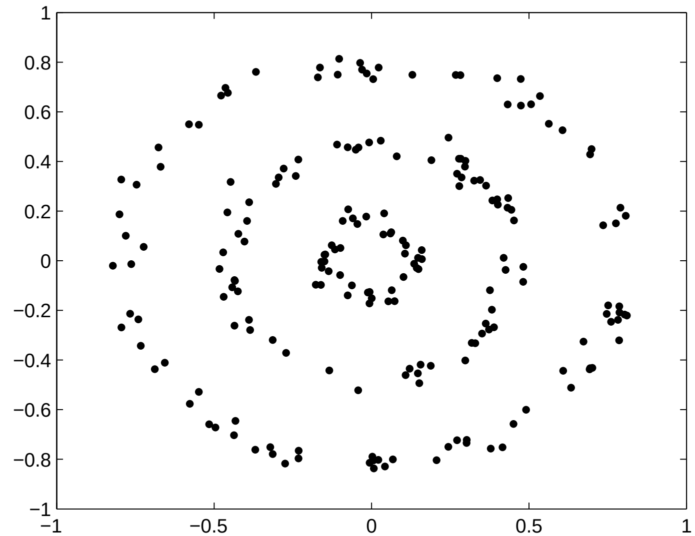
  then σ_{opt} is the maximum of the small curvatures.
- Claim:

$$\sigma_{\text{opt}} \in \left[\mathbf{c}\left(N \cdot c/K^{d+1}\right), \mathbf{c}\left(N \cdot c/K\right)\right],$$

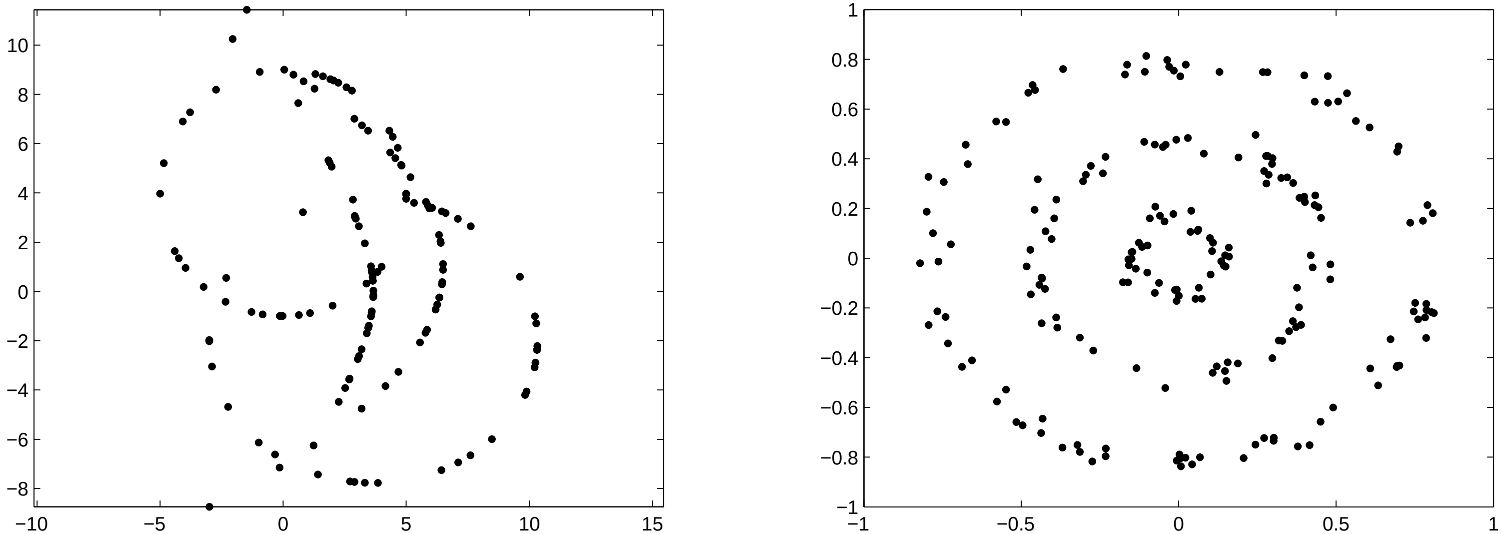
  where $\mathbf{c}$: vector of all $N \cdot c$ curvatures, sorted in nondecreasing order

From Linear to Nonlinear





From Linear to Nonlinear

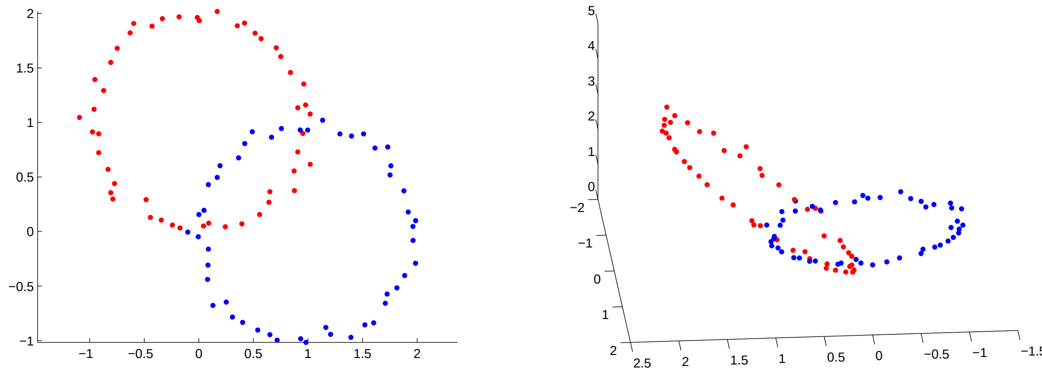


Two natural ways of extending SCC for manifold clustering:

- **Kernelize** SCC: linearize data in a feature space and apply SCC there
- **Localize** SCC: apply SCC for near neighbors to compute pairwise weights

Kernel SCC (KSCC)

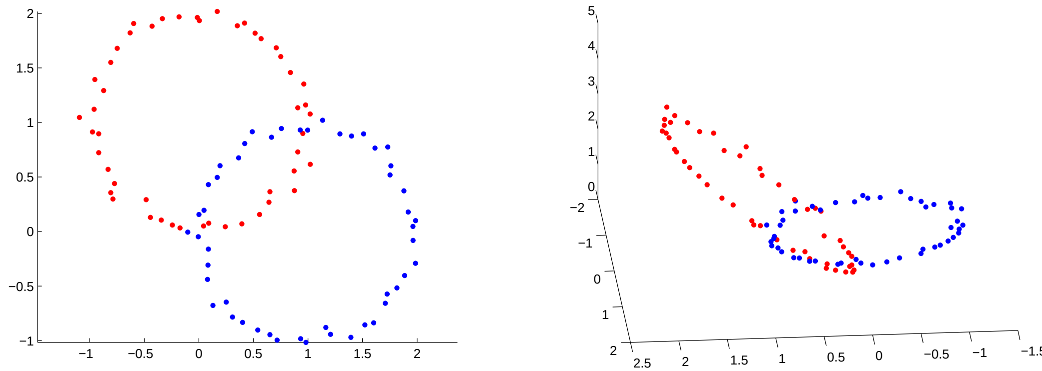
- The idea is to find a feature map Φ to map data to linear manifolds and then apply SCC in the feature space



$$(\Phi(\mathbf{x}) = \Phi(x_1, x_2) = (x_1, x_2, x_1^2 + x_2^2))$$

Kernel SCC (KSCC)

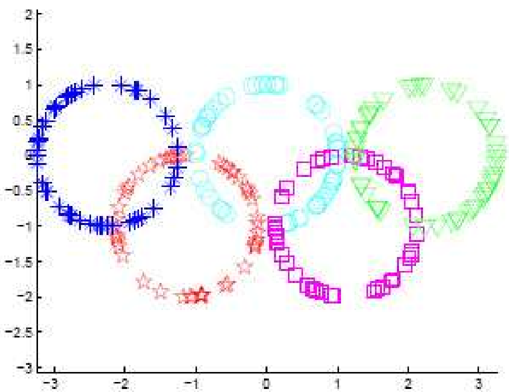
- The idea is to find a feature map Φ to map data to linear manifolds and then apply SCC in the feature space



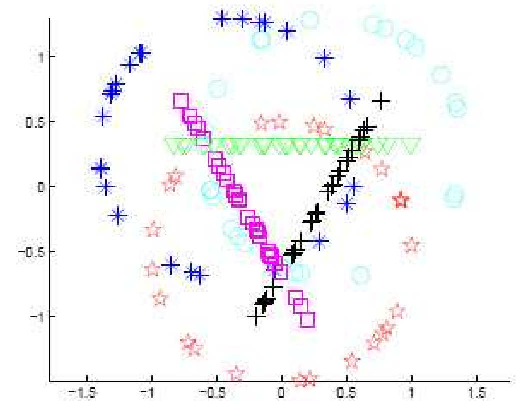
$(\Phi(\mathbf{x}) = \Phi(x_1, x_2) = (x_1, x_2, x_1^2 + x_2^2))$

- Since $c_p(\mathbf{x}_1, \ldots, \mathbf{x}_{d+2})$=diameter*volume/edgeLengths and hence SCC depends only on dot products, we only need to specify the kernel function $k(\mathbf{x}, \mathbf{y}) = <\Phi(\mathbf{x}), \Phi(\mathbf{y})>$ and use it to replace dot product in SCC (e.g., $k(\mathbf{x}, \mathbf{y}) = <\mathbf{x}, \mathbf{y}> + \|\mathbf{x}\|_2^2 \cdot \|\mathbf{y}\|_2^2$)

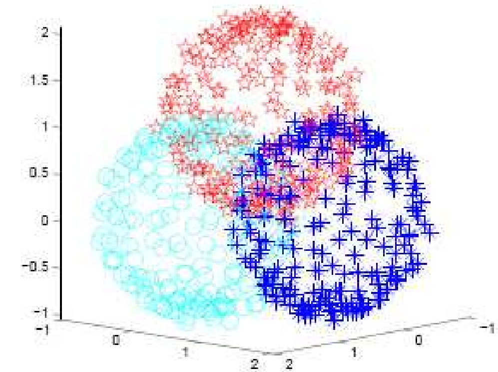
KSCC: Some Examples



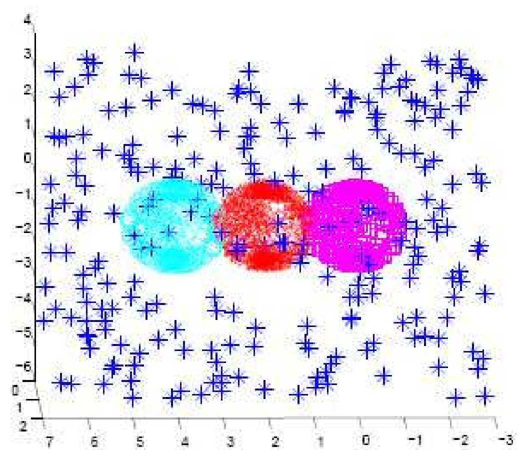
(a) five circles



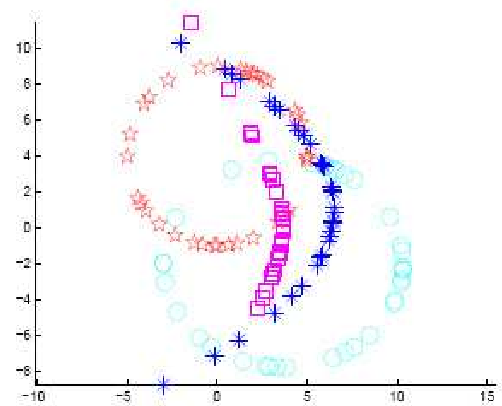
(b) three lines and three circles



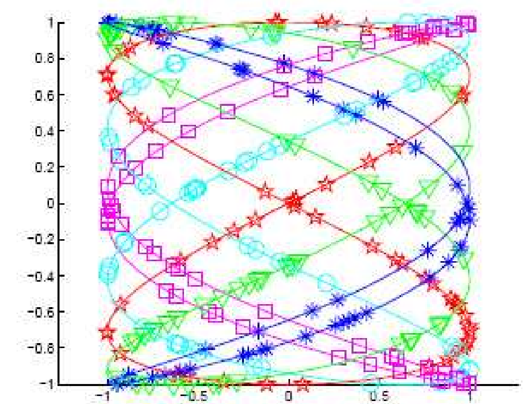
(c) three (noisy) spheres



(d) three adjacent unit spheres and a plane through their centers



(e) four 1D conic sections



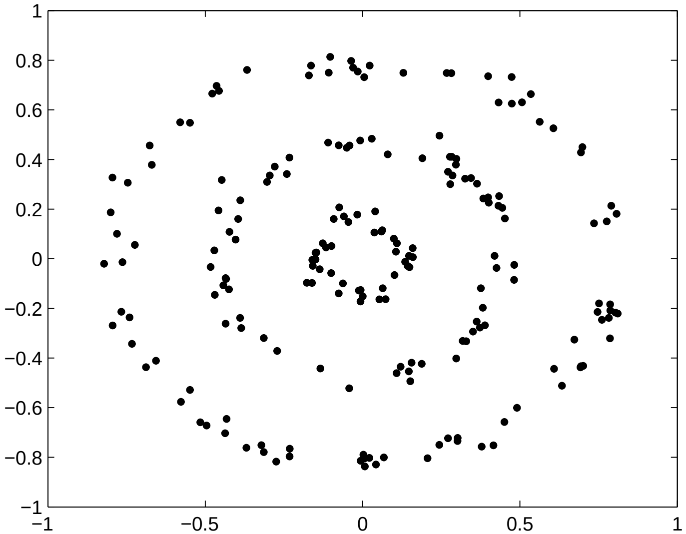
(f) five Lissajous curves

Local SCC

Idea: Fix an integer $m \geq d+2$. Compute pairwise weights only using and for nearest neighbors

$$\mathbf{W}_{ik} = \sum_{j_1,\ldots,j_{m-1} \in \mathcal{N}(i)} \mathcal{A}_\mathsf{p}(i, j_1, \ldots, j_{m-1}) \cdot \mathcal{A}_\mathsf{p}(k, j_1, \ldots, j_{m-1})$$

for $k \in \mathcal{N}(i)$, and 0 otherwise

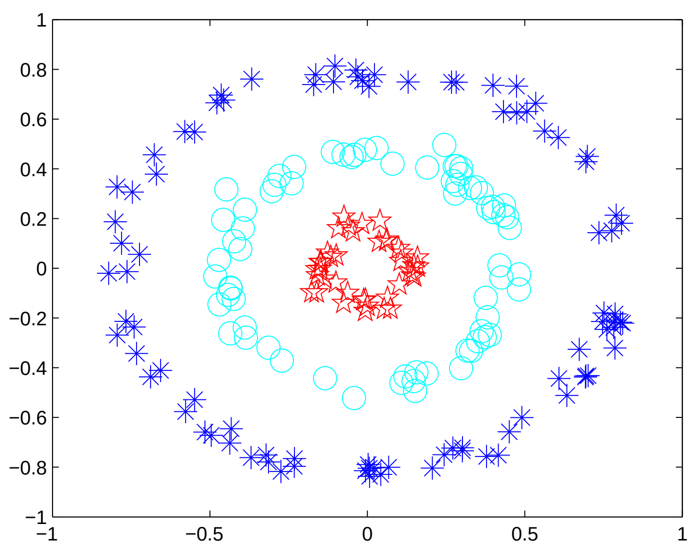


Local SCC

Idea: Fix an integer $m \geq d + 2$. Compute pairwise weights only using and for nearest neighbors

$$\mathbf{W}_{ik} = \sum_{j_1,\ldots,j_{m-1} \in \mathcal{N}(i)} \mathcal{A}_{\mathsf{p}}(i, j_1, \ldots, j_{m-1}) \cdot \mathcal{A}_{\mathsf{p}}(k, j_1, \ldots, j_{m-1})$$

for $k \in \mathcal{N}(i)$, and 0 otherwise

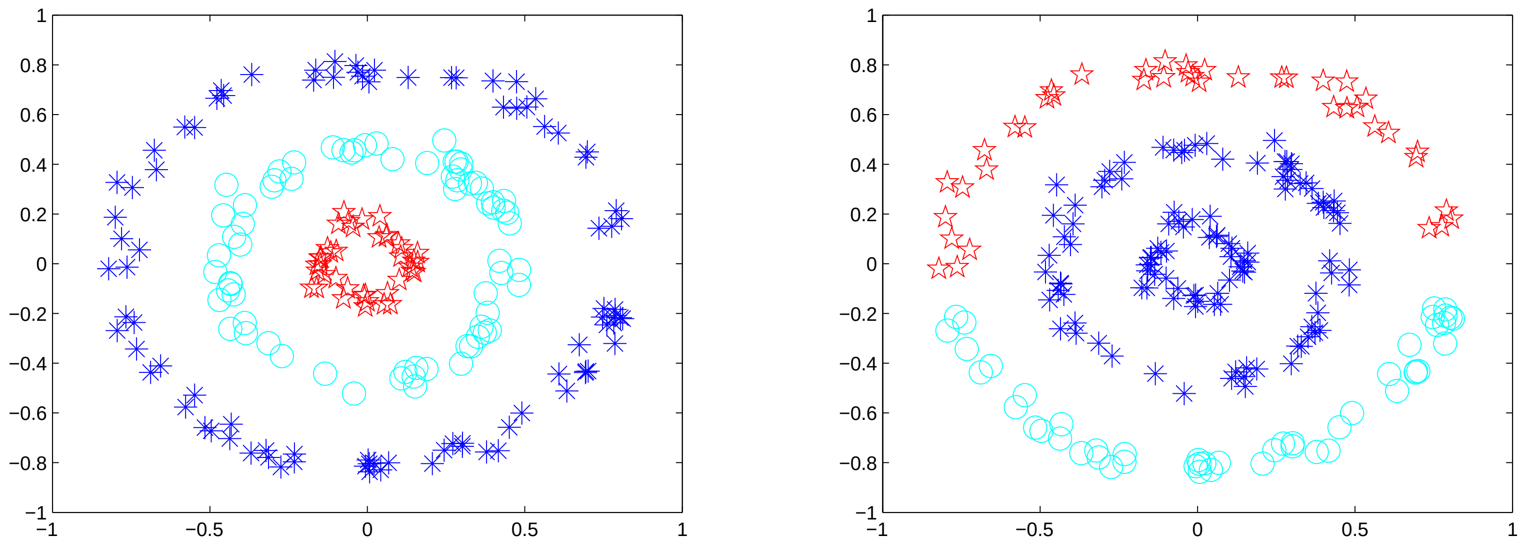


Local SCC

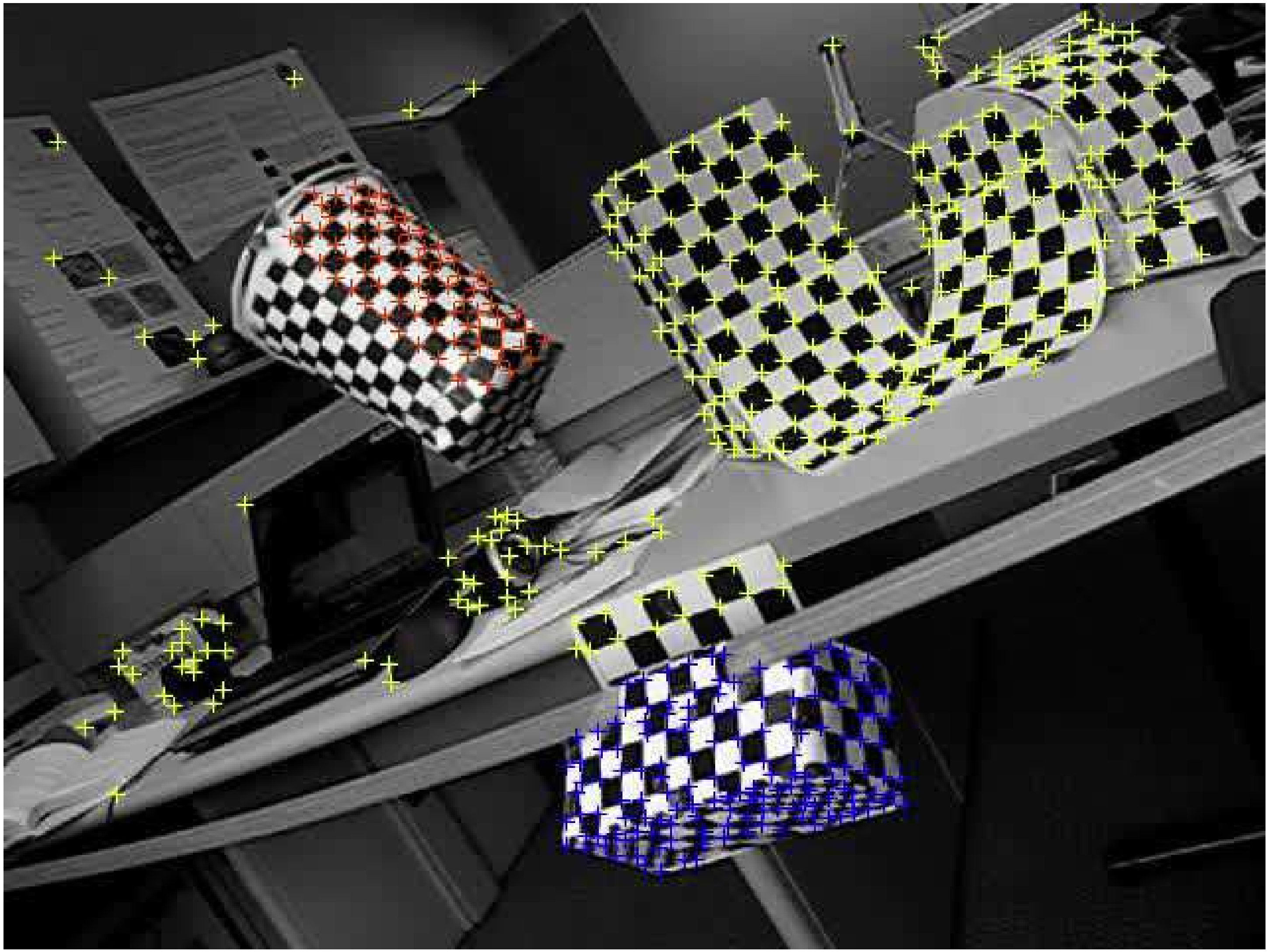
Idea: Fix an integer $m \geq d + 2$. Compute pairwise weights only using and for nearest neighbors

$$\mathbf{W}_{ik} = \sum_{j_1,\ldots,j_{m-1} \in \mathcal{N}(i)} \mathcal{A}_{\mathsf{p}}(i, j_1, \ldots, j_{m-1}) \cdot \mathcal{A}_{\mathsf{p}}(k, j_1, \ldots, j_{m-1})$$

for $k \in \mathcal{N}(i)$, and 0 otherwise



Application: Motion Segmentation



Application: Motion Segmentation

- **Problem**: cluster (pre-collected) trajectory vectors

$$\mathbf{z}^{(i)} = (x_1^{(i)}, y_1^{(i)}, x_2^{(i)}, x_2^{(i)}, \ldots, x_F^{(i)}, y_F^{(i)})', 1 \leq i \leq N$$

  of feature points tracked on different moving objects

Application: Motion Segmentation

- **Problem**: cluster (pre-collected) trajectory vectors

$$\mathbf{z}^{(i)} = (x_1^{(i)}, y_1^{(i)}, x_2^{(i)}, x_2^{(i)}, \ldots, x_F^{(i)}, y_F^{(i)})', 1 \leq i \leq N$$

of feature points tracked on different moving objects

- Under the affine camera model, i.e.,

$$(x_f^{(i)}, y_f^{(i)})' = \left(\mathbf{A}_f\right)_{2\times 3} \mathbf{r}_{3\times 1}^{(i)} + \left(\mathbf{b}_f\right)_{2\times 1},$$

we have for trajectories on k-th moving object

$$[\mathbf{z}^{(1)}, \ldots, \mathbf{z}^{(N_k)}]_{2F\times N_k} = \begin{bmatrix} \mathbf{A}_1 & \mathbf{b}_1 \\ \vdots & \vdots \\ \mathbf{A}_F & \mathbf{b}_F \end{bmatrix}_{2F\times 4} \begin{bmatrix} \mathbf{r}^{(1)} & \ldots & \mathbf{r}^{(N_k)} \\ 1 & \ldots & 1 \end{bmatrix}_{4\times N_k}$$

Application: Motion Segmentation

- **Problem**: cluster (pre-collected) trajectory vectors

$$\mathbf{z}^{(i)} = (x_1^{(i)}, y_1^{(i)}, x_2^{(i)}, x_2^{(i)}, \ldots, x_F^{(i)}, y_F^{(i)})', 1 \leq i \leq N$$

  of feature points tracked on different moving objects
- **Fact**: Trajectories associated with same moving object live on a distinct 3D affine subspace
- **Tool**: hybrid linear modeling via SCC

Performance on a Benchmark

Hopkins155 Database of 155 video sequences: 120 two motions ($N = 266, F = 30$), 35 three motions ($N = 398, F = 29$)

classfication errors	two motions		three motions	
	mean	median	mean	median
RANSAC	5.56%	1.18%	22.94%	22.03%
GPCA	4.59%	0.38%	28.66%	28.26%
LSA 5	6.73%	1.99%	29.28%	31.63%
LSA $4K$	3.45%	0.59%	9.73%	2.33%
MSL	4.14%	0.00%	8.23%	1.76%
SCC $2F$	1.40%	0.10%	5.77%	2.21%
SCC 5	2.10%	0.26%	4.94%	1.70%

Summary & Open Questions

- Presented SCC + kernelization & localization
- SCC
  - Automatic inference of K and d_k
  - Further improvement for mixed dimensions
  - Theoretical investigation of iterative sampling
  - New initialization (e.g., by multiscale SVD)
- Kernel SCC
  - Optimal kernel selection
- Local SCC
  - Automatic tuning of the parameters (e.g., neighborhood size)

Acknowledgements

- **Collaborators**:
  - Ery Arias-Castro, UCSD (on localization of scc)
  - Stefan Atev, U of MN (on kernelization of scc)
  - Gilad Lerman (PhD advisor), U of MN (on all three)
- **Contact**: *glchen@math.duke.edu*
- **SCC website** (with papers, matlab codes, data, etc.): *http://www.math.duke.edu/~glchen/scc.html*

Thank you for coming to the talk!