# 6998: Learning and graphs (connecting the dots)

Tony Jebara

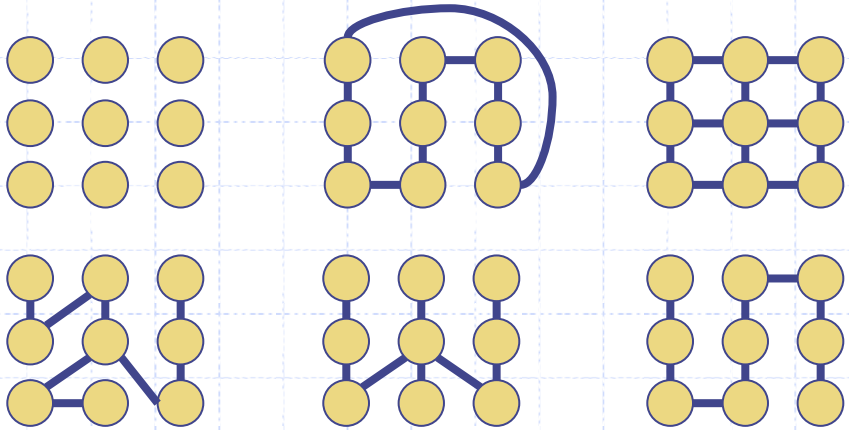
joint with Andy, Bert, Blake, Pannaga, Risi, Stu, Vlad

# Outline

- Why? Classification, Dimensionality Reduction, Clustering,
- Greedy Connections: Nearest Neighbors & kNN
- Less Greedy Connections: Maximum Spanning Trees
- Less Greedy Connections: Matching & b-Matching
- Bayesian Connections: Distributions over Trees
- Bayesian Connections: Distributions over Out-Trees

# Connecting the Dots

- Greedy, Constrained, Bayesian…

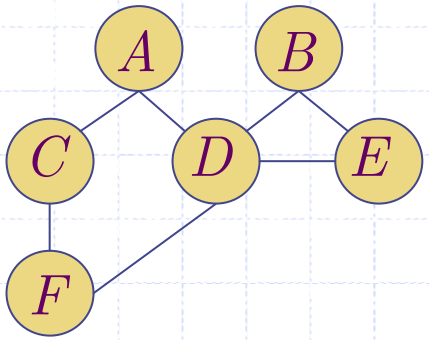


# K-Nearest Neighbors

- Start with unconnected points
- Compute pairs of distances

  $A_{ij} = d(X_i, X_j)$
- Connect each point to its k closest points

  $B_{ij} = A_{ij} <= sort(A_{i*})_k$

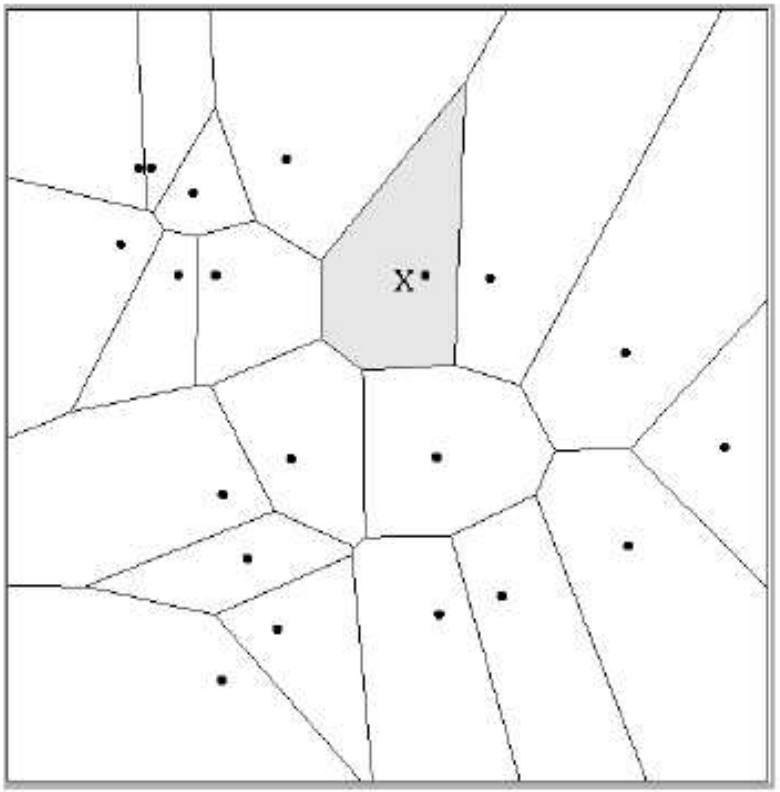



| | A | B | C | D | E | F |
|---|---|---|---|---|---|---|
| A | - | | 1 | 1 | | |
| B | | - | | 1 | 1 | |
| C | 1 | | - | | | 1 |
| D | 1 | 1 | | - | | |
| E | | 1 | | 1 | - | |
| F | | | 1 | 1 | | - |

- Then Symmetrize connectivity matrix: $B_{ij} = \max(B_{ji}, B_{ij})$

# K-Nearest Neighbors Classifier

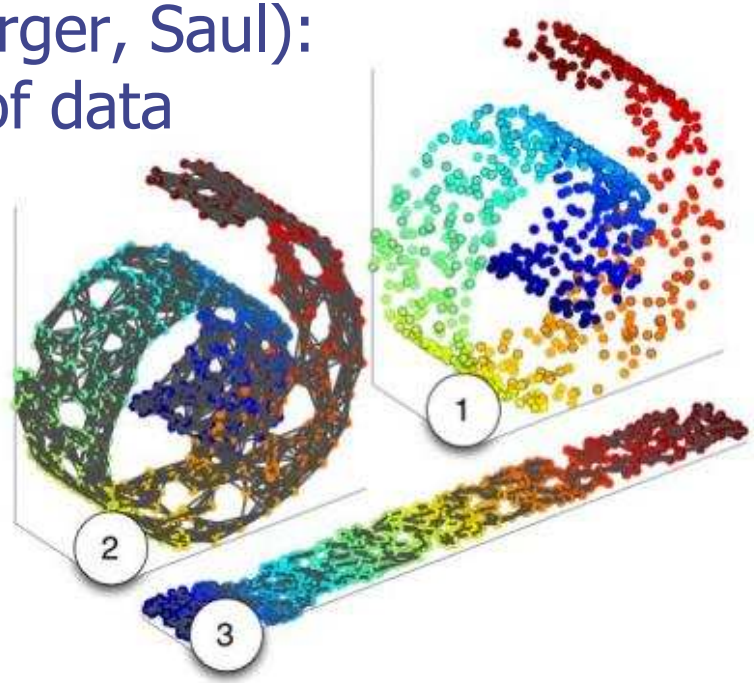
- Find your k nearest neighbors
- Your label is a vote over their labels:



# KNN & Semidefinite Embedding

- Want to visualize high-dimensional $\{x_1,\ldots,x_N\}$ data
- Standard PCA or Kernel PCA (Shoelkopf):
  - -Get matrix A of affinities between pairs $A_{ij}=k(x_i,x_j)$
  - -Get SVD of A & view top projections

- Semidefinite Embedding (Weinberger, Saul):
  - -Get k-nearest neighbors graph of data
  - -Get matrix A
  - -Use max trace (variance) SDP stretch graph A to PD graph K
  - -Get SVD of K & view



# KNN & Semidefinite Embedding

- SDE pulls apart knn connected graph C maximally while preserving pairwise distances between points if $C_{ij}=1$

$$\min_K -\sum_i \lambda_i \quad s.t.\ K \in \kappa$$

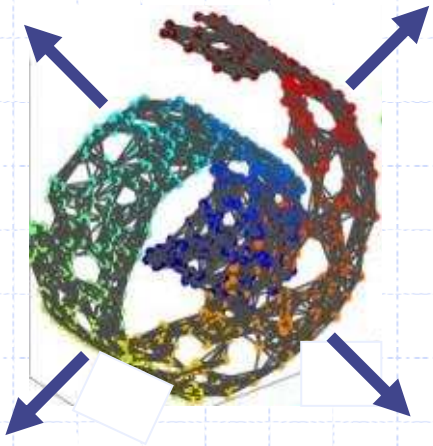
$$\kappa = \left\{ \forall K \in \Re^{N \times N} \right\}$$

$$s.t.\ K \succeq 0$$

$$s.t. \sum_{ij} K_{ij} = 0$$

$$s.t.\ K_{ii} + K_{jj} - K_{ij} - K_{ji} =$$

$$A_{ii} + A_{jj} - A_{ij} - A_{ji} \quad if\ C_{ij} = 1$$



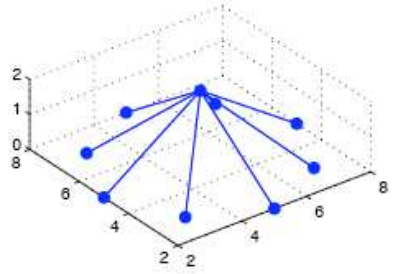
- SDE's stretching of data *improves* the visualization!

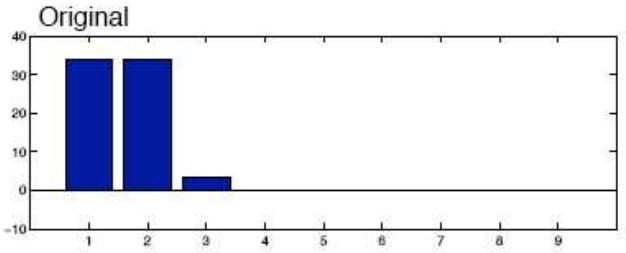
$A$

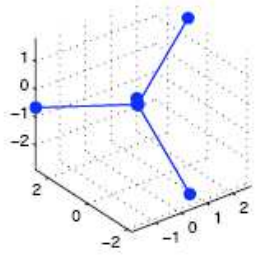
$K$

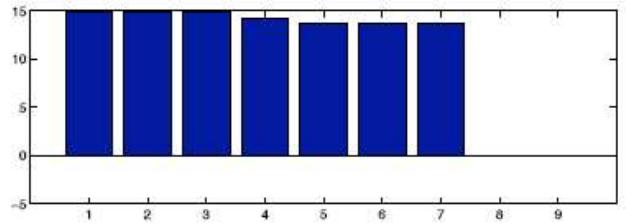
# KNN & Semidefinite Embedding

- Possible problem with SDE's pulling apart behavior.
- SDE stretch can sometimes *worsen* visualization!
- Spokes Experiment:

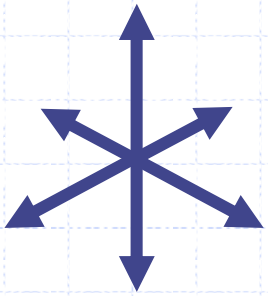




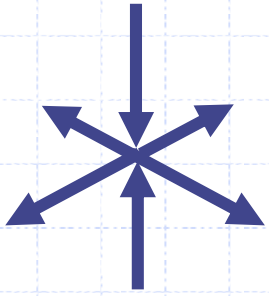






- Want to pull apart only in visualized dimensions
- Squash or iron down remaining ones



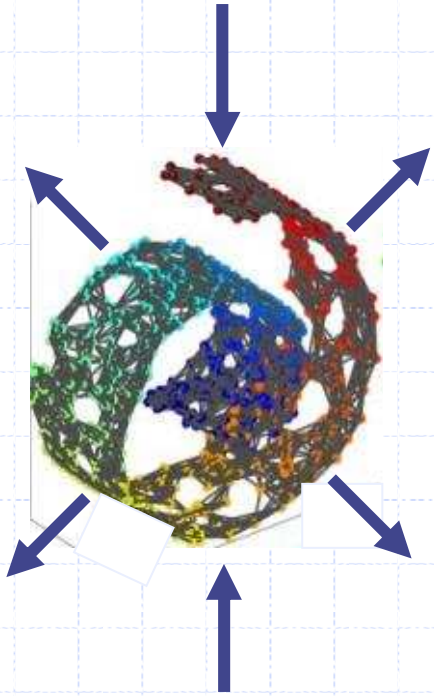
vs.



# Minimum Volume Embedding

- Stretch in d<D top dimensions and squash rest. Not SDP!

$$\min_K -\sum_{i=1}^{d} \lambda_i + \sum_{i=d+1}^{D} \lambda_i \quad s.t.\ K \in \kappa$$



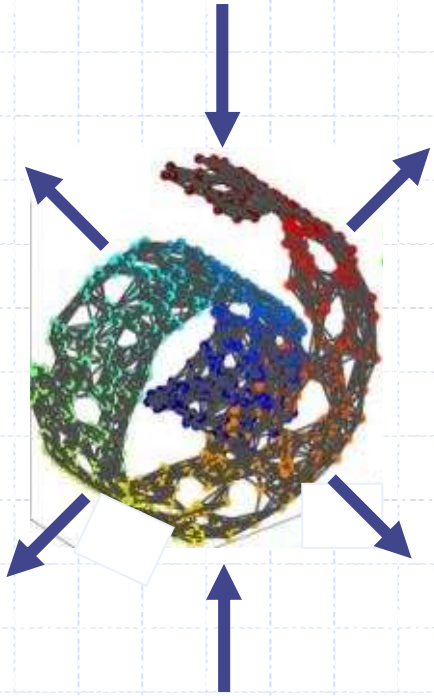
# Minimum Volume Embedding

- Stretch in d<D top dimensions and squash rest. Not SDP!

$$\min_K -\sum_{i=1}^{d} \lambda_i + \sum_{i=d+1}^{D} \lambda_i \quad s.t.\ K \in \kappa$$

$$= \min_K -\sum_{i=1}^{d} \lambda_i + tr \sum_{i=d+1}^{D} \lambda_i$$

$$s.t.\ K \in \kappa, Kv_i = \lambda_i v_i, \lambda_i \geq \lambda_{i+1}, v_i^T v_j = \delta_{ij}$$



# Minimum Volume Embedding

- Stretch in d<D top dimensions and squash rest. Not SDP!

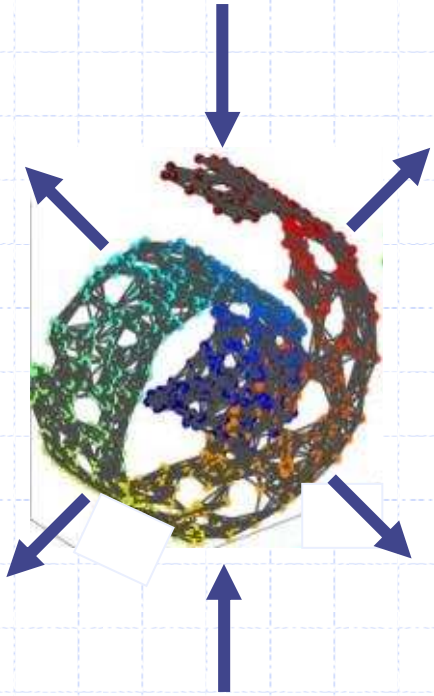
$$\min_K -\sum_{i=1}^{d} \lambda_i + \sum_{i=d+1}^{D} \lambda_i \quad s.t.\ K \in \kappa$$

$$= \min_K -\sum_{i=1}^{d} \lambda_i + tr \sum_{i=d+1}^{D} \lambda_i$$

$$s.t.\ K \in \kappa, Kv_i = \lambda_i v_i, \lambda_i \geq \lambda_{i+1}, v_i^T v_j = \delta_{ij}$$

$$= \min_K -\sum_{i=1}^{d} trKv_i v_i^T + \sum_{i=d+1}^{D} trKv_i v_i^T$$

$$s.t.\ K \in \kappa, Kv_i = \lambda_i v_i, \lambda_i \geq \lambda_{i+1}, v_i^T v_j = \delta_{ij}$$



# Minimum Volume Embedding

- Stretch in d<D top dimensions and squash rest. Not SDP!

$$\min_K -\sum_{i=1}^{d} \lambda_i + \sum_{i=d+1}^{D} \lambda_i \quad s.t.\ K \in \kappa$$

$$= \min_K -\sum_{i=1}^{d} \lambda_i + tr \sum_{i=d+1}^{D} \lambda_i$$

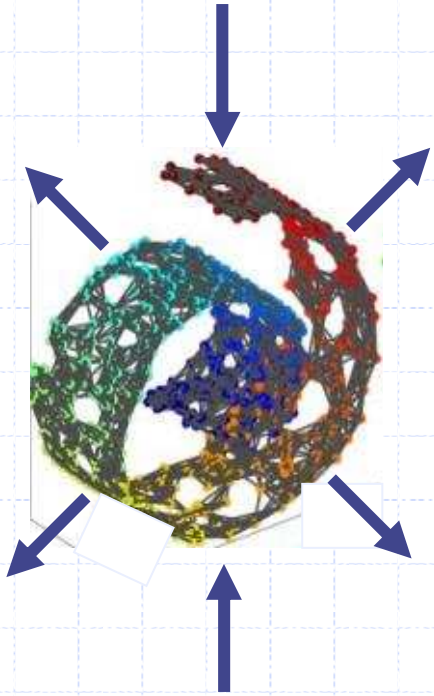
$$s.t.\ K \in \kappa, Kv_i = \lambda_i v_i, \lambda_i \geq \lambda_{i+1}, v_i^T v_j = \delta_{ij}$$

$$= \min_K -\sum_{i=1}^{d} trKv_i v_i^T + \sum_{i=d+1}^{D} trKv_i v_i^T$$

$$s.t.\ K \in \kappa, Kv_i = \lambda_i v_i, \lambda_i \geq \lambda_{i+1}, v_i^T v_j = \delta_{ij}$$

$$= \min_K \min_V -\sum_{i=1}^{d} trKv_i v_i^T + \sum_{i=d+1}^{D} trKv_i v_i^T$$

$$s.t.\ K \in \kappa, \lambda_i \geq \lambda_{i+1}, v_i^T v_j = \delta_{ij}$$



# Minimum Volume Embedding

- Stretch in d<D top dimensions and squash rest. Not SDP!

$$\min_K -\sum_{i=1}^{d}\lambda_i + \sum_{i=d+1}^{D}\lambda_i \quad s.t.\ K \in \kappa$$

$$= \min_K -\sum_{i=1}^{d}\lambda_i + tr\sum_{i=d+1}^{D}\lambda_i$$

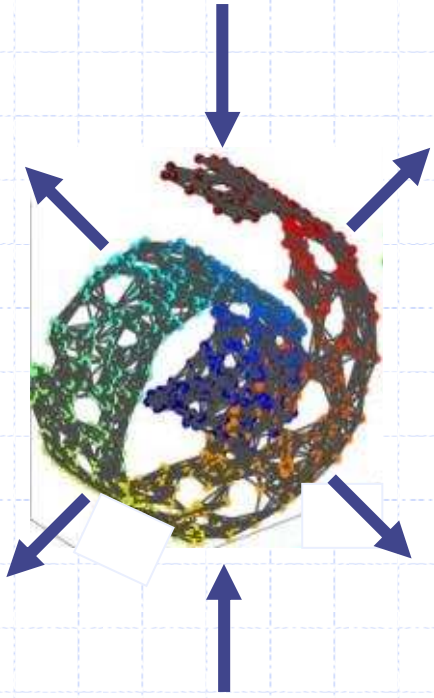
$$s.t.\ K \in \kappa, Kv_i = \lambda_i v_i, \lambda_i \geq \lambda_{i+1}, v_i^T v_j = \delta_{ij}$$

$$= \min_K -\sum_{i=1}^{d} trKv_i v_i^T + \sum_{i=d+1}^{D} trKv_i v_i^T$$

$$s.t.\ K \in \kappa, Kv_i = \lambda_i v_i, \lambda_i \geq \lambda_{i+1}, v_i^T v_j = \delta_{ij}$$

$$= \min_K \min_V -\sum_{i=1}^{d} trKv_i v_i^T + \sum_{i=d+1}^{D} trKv_i v_i^T$$

$$s.t.\ K \in \kappa, \lambda_i \geq \lambda_{i+1}, v_i^T v_j = \delta_{ij}$$



- Variational upper bound on cost à Iterated Monotonic SDP
- Lock V and solve SDP K. Lock K and solve SVD for V.

# Minimum Volume Embedding

- Stretch in d<D top dimensions and squash rest. Not SDP!

$$\min_K -\sum_{i=1}^{d}\lambda_i + \sum_{i=d+1}^{D}\lambda_i \quad s.t.\ K \in \kappa$$

$$= \min_K -\sum_{i=1}^{d}\lambda_i + tr\sum_{i=d+1}^{D}\lambda_i$$

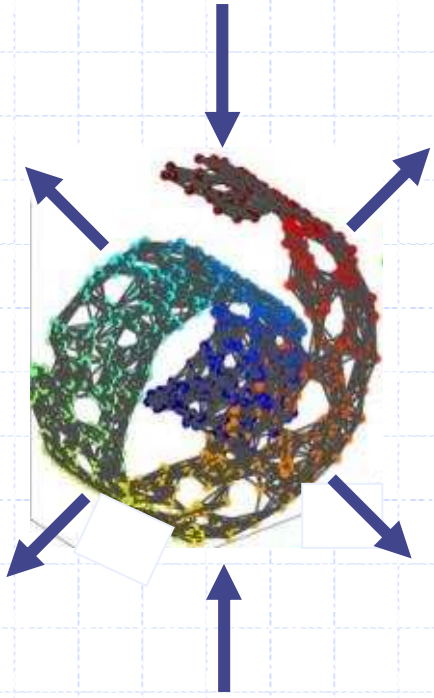
$$s.t.\ K \in \kappa, Kv_i = \lambda_i v_i, \lambda_i \geq \lambda_{i+1}, v_i^T v_j = \delta_{ij}$$

$$= \min_K -\sum_{i=1}^{d} trKv_i v_i^T + \sum_{i=d+1}^{D} trKv_i v_i^T$$

$$s.t.\ K \in \kappa, Kv_i = \lambda_i v_i, \lambda_i \geq \lambda_{i+1}, v_i^T v_j = \delta_{ij}$$

$$= \min_K \min_V -\sum_{i=1}^{d} trKv_i v_i^T + \sum_{i=d+1}^{D} trKv_i v_i^T$$

$$s.t.\ K \in \kappa, \lambda_i \geq \lambda_{i+1}, v_i^T v_j = \delta_{ij}$$



- Variational upper bound on cost à Iterated Monotonic SDP
- Lock V and solve SDP K. Lock K and solve SVD for V.
- Procrustes finds V: sort top eigenvectors of current K!

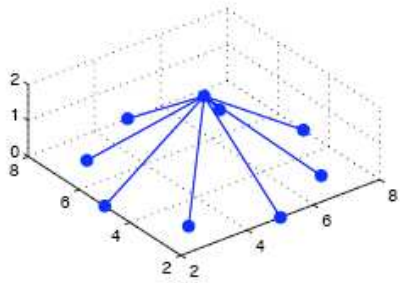
# Minimum Volume Embedding

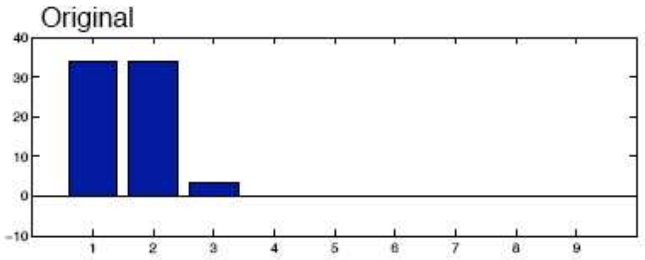
- Overall algorithm:

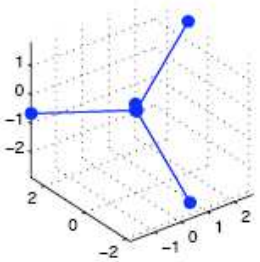
| | |
|---|---|
| Input | $(\vec{x}_i)_{i=1}^N$, kernel $\kappa$, and parameters $d, k$. |
| Step 1 | Form affinity matrix $A \in \Re^{N \times N}$ with pairwise entries $A_{ij} = \kappa(\vec{x}_i, \vec{x}_j)$. |
| Step 2 | Use $A$ to find a binary connectivity matrix $C$ via $k$-nearest neighbors. |
| Step 3 | Initialize $K = A$. |
| Step 4 | Solve for the eigenvectors $\vec{v}_1, \ldots, \vec{v}_N$ and eigenvalues $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_N$ of $K$. |
| Step 5 | Set $B = -\sum_{i=1}^{d} \vec{v}_i \vec{v}_i^T + \sum_{i=d+1}^{N} \vec{v}_i \vec{v}_i^T$. |
| Step 6 | Using SDP find $\hat{K} = \arg\min_{K \in \mathcal{K}} tr(KB)$. |
| Step 7 | If $\|K - \hat{K}\| \geq \epsilon$ set $K = \hat{K}$, go to Step 4. |
| Step 8 | Perform kernel PCA on $K^*$ to get $d$-dimensional output vectors $\vec{y}_1, \ldots, \vec{y}_N$. |

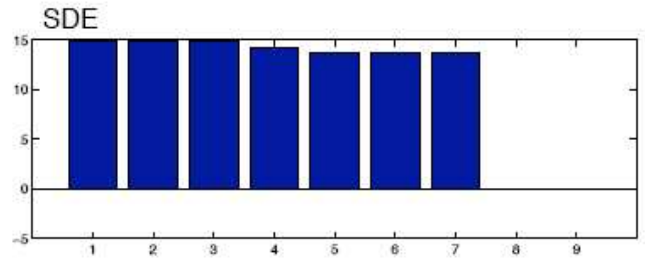
# Minimum Volume Embedding

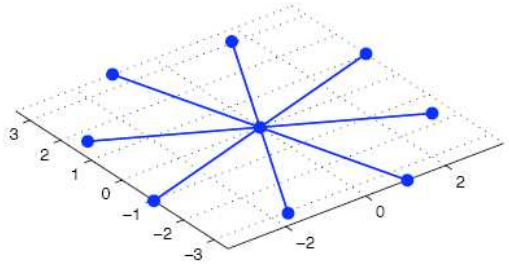
- Spokes experiment visualization and spectra
- Converges in ~5 iterations

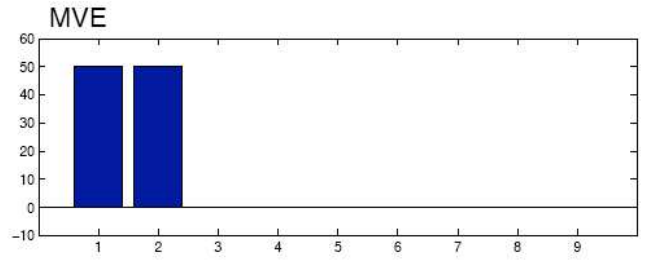








# Minimum Volume Embedding

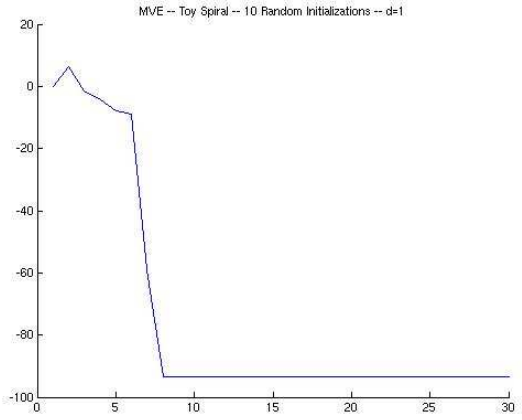
- Swissroll Visualization and Convergence (Connectivity via knn) (d is set to 1)

PCA

SDE

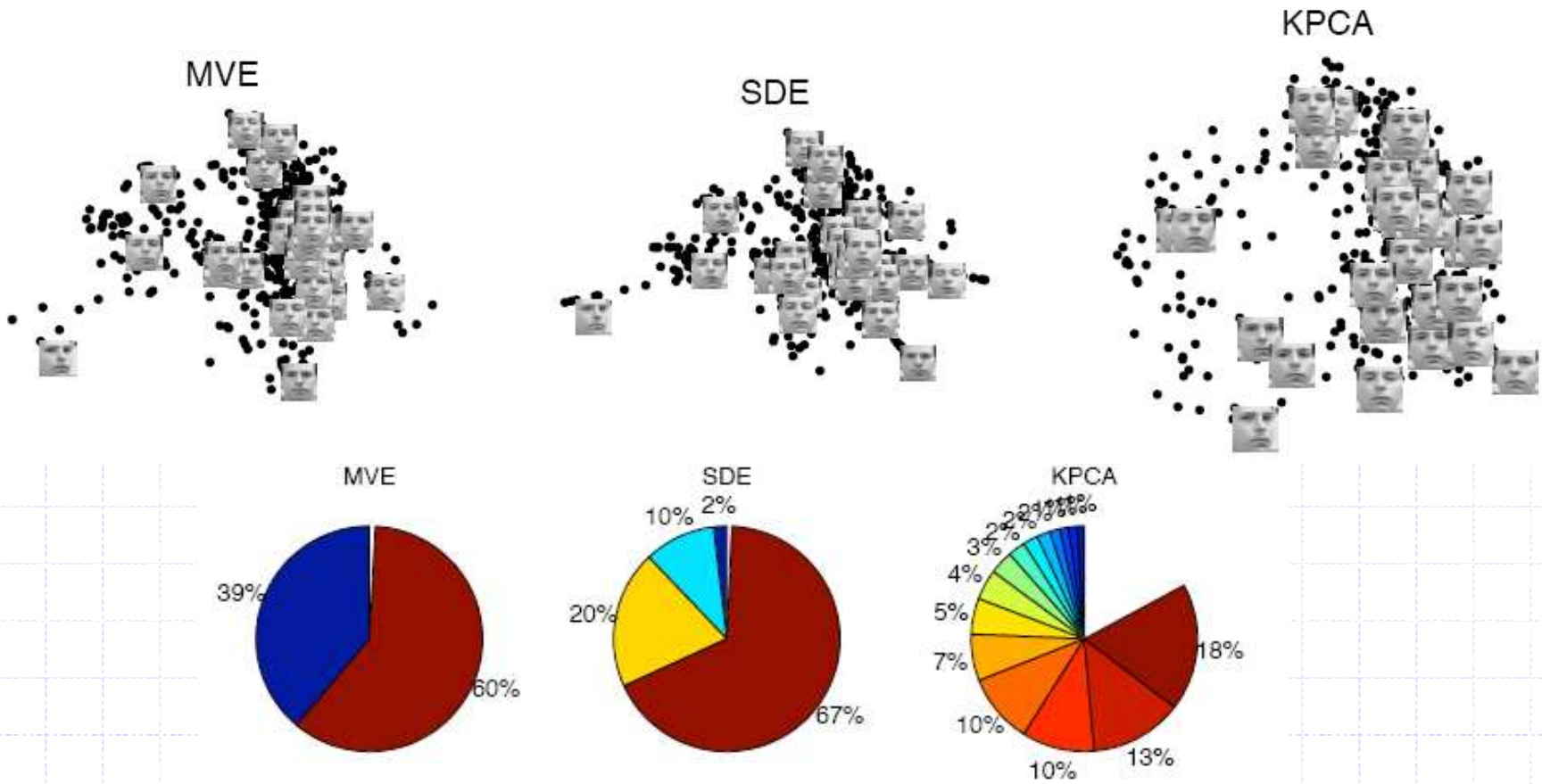
MVE

- Cost convergence the same way after first iteration for any random initialization or K=A…



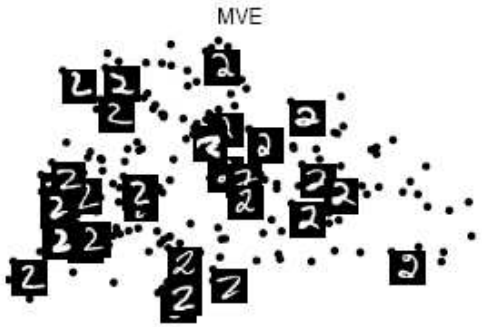
# Minimum Volume Embedding

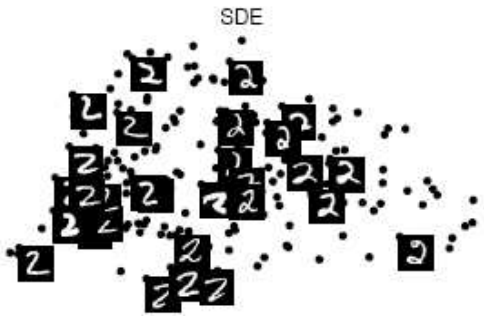
- Face Images Visualization and Spectra
  (Connectivity via knn, d=2)

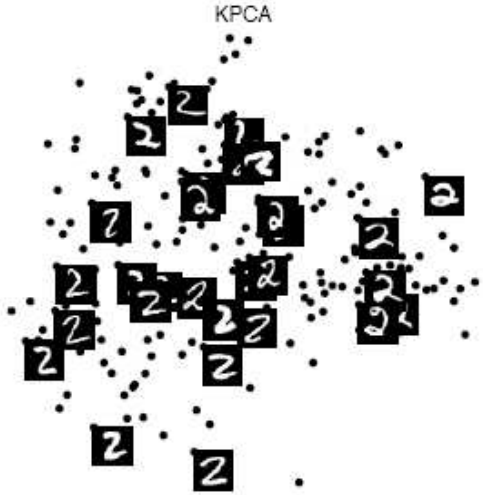


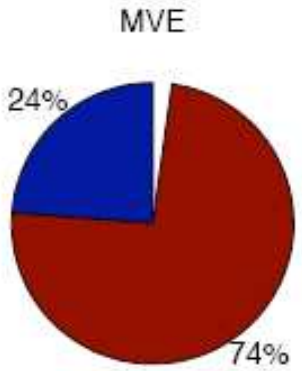
# Minimum Volume Embedding

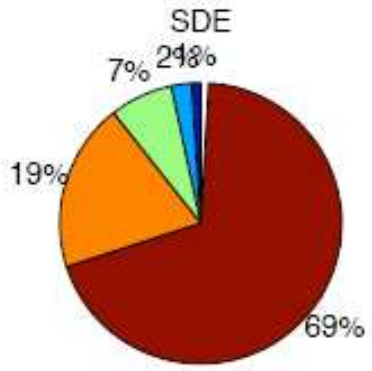
- Digit Images Visualizations and Spectra (Connectivity via knn, d=2)

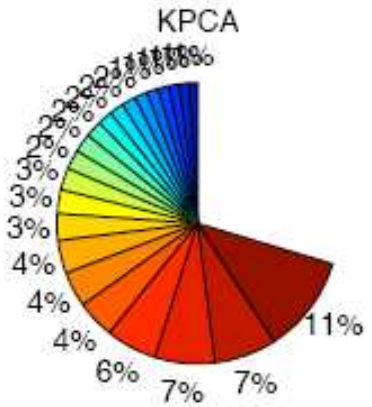


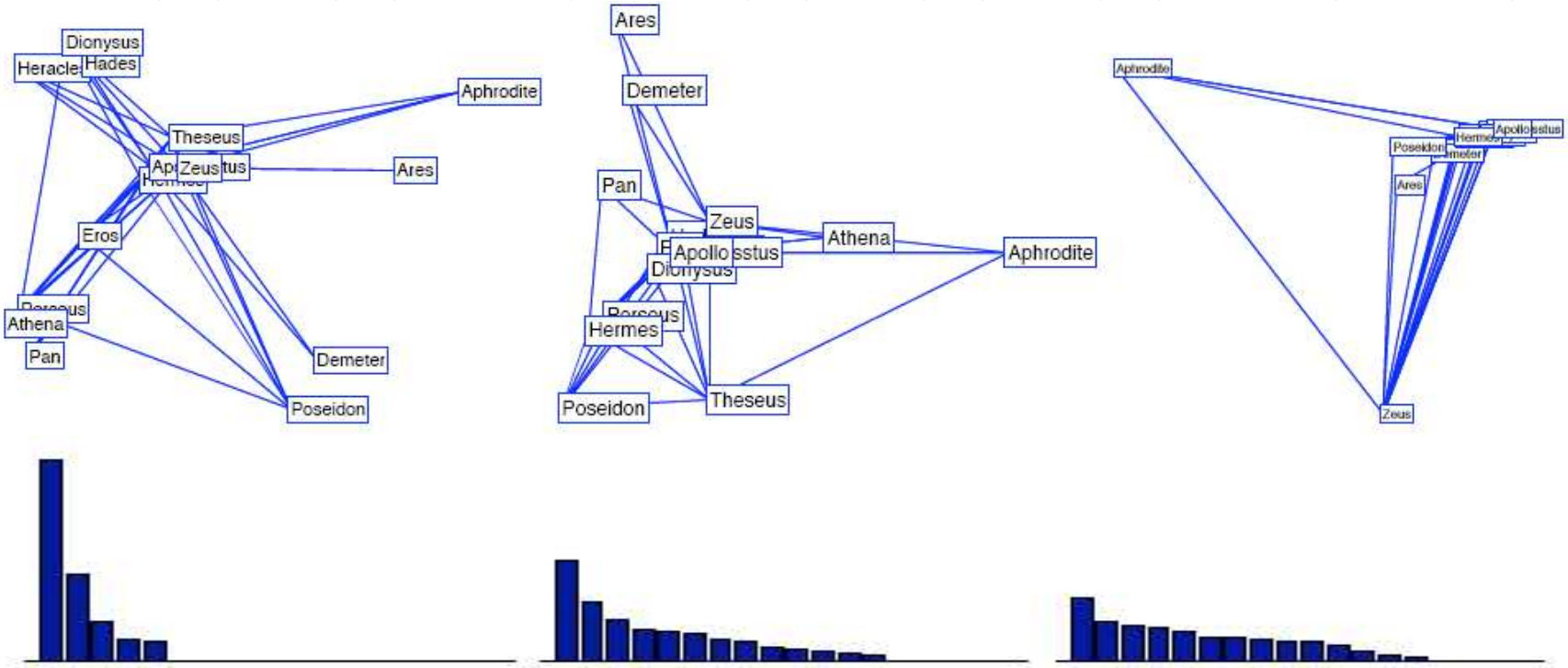



# Minimum Volume Embedding

- Social Network Visualization and Spectra
  (Connectivity determined by friendship links, d=2)



# Minimum Volume Embedding

- Quantitative Performance
- Need to maintain pairwise distances in *available* dimensions

Percentage of eigenvalue energy captured in 2D

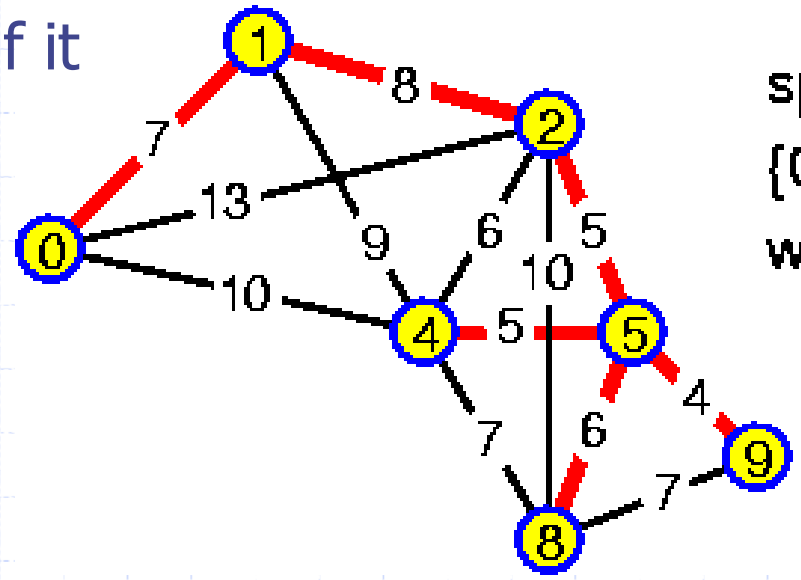
| | MVE | SDE | KPCA |
|---|---|---|---|
| Hubs and Spokes | 100% | 29.9% | 95.0% |
| Spiral (% in 1D) | 99.9% | 99.9% | 45.8% |
| Twos | 97.8% | 88.4% | 18.4% |
| Faces | 99.2% | 83.6% | 31.4% |
| Social Networks | 77.5% | 41.7% | 29.3% |

**Via convexity of max eigenvalue**

- This cost is convex: $\min_K \sum_{i=1}^{1} \lambda_i - \sum_{i=2}^{D} \lambda_i \;\; s.t. \; K \in \kappa$
- MVE cost is not: $\min_K -\sum_{i=1}^{d} \lambda_i + \sum_{i=d+1}^{D} \lambda_i \;\; s.t. \; K \in \kappa$
- But, if d=1, see global convergence behavior...

# Minimum Spanning Tree

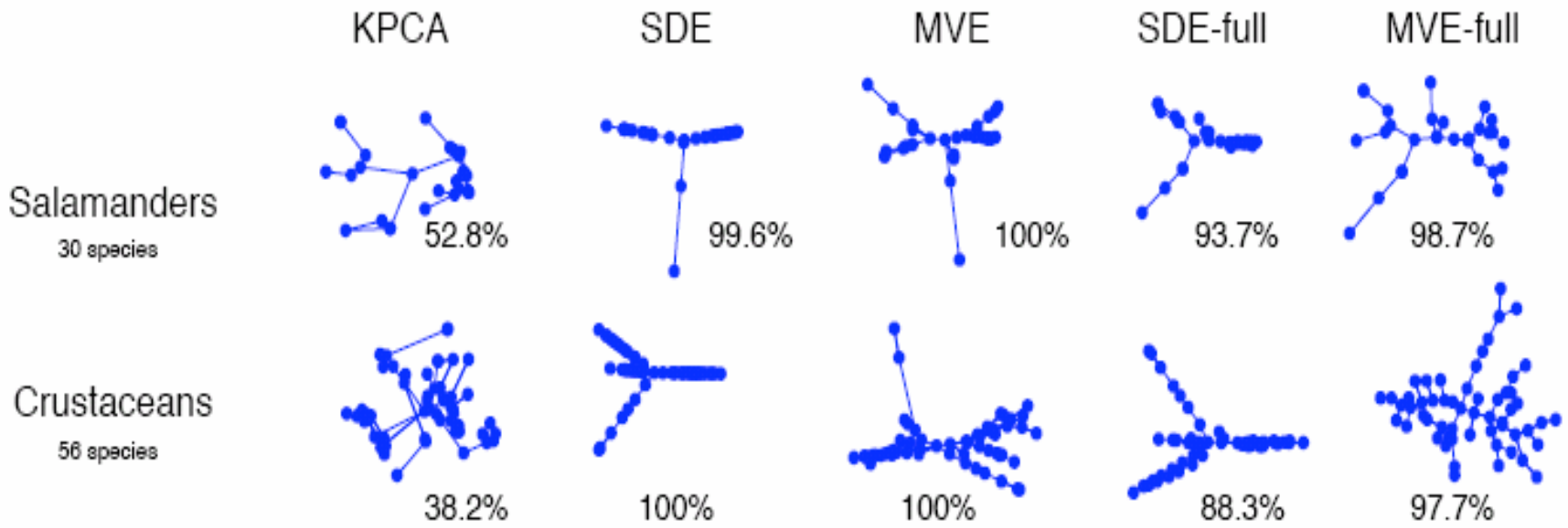
- Start with unconnected points
- Kruskal's Algorithm (Prim too)
- Compute pairs of distances
  $A_{ij} = d(X_i, X_j)$
- Greedily connect the smallest distances in A
- Skip a connection if it forms a loop




spanning
{0,1,2,4,5,8,9}
weight = 35

# Trees for MVE & SDE

- Salamander Phylogenetic Tree Visualization and Spectra (connected with tree structure, d=2)
- SDE & MVE undergo collapsing if connectivity is sparse




$K \in \kappa \;\; as\, usual$

$K \in \kappa \;\; and \;\; K_{ii} + K_{jj} - K_{ij} - K_{ji} \geq A_{ii} + A_{jj} - A_{ij} - A_{ji}$

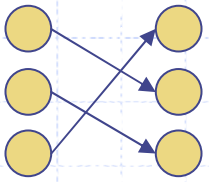
# Matching and B-Matching

- We explore these 2 combinatorial optimization algorithms from graph theory and find ways to use them in learning.
- Maximum Weight Matching = Linear Assignment Problem
  Given weight matrix, find permutation matrix. $O(N^3)$

$$\text{wife} \begin{bmatrix} \$1 & \$6 & \$3 \\ \$4 & \$2 & \$4 \\ \$4 & \$2 & \$5 \end{bmatrix} \rightarrow \begin{matrix} \text{husband} \\ \text{Kuhn-Munkres} \\ \text{Hungarian Algorithm} \end{matrix} \rightarrow \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$\max_P tr\left(P^T A\right)$$

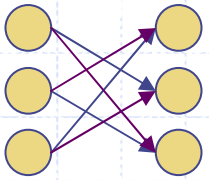
$$\sum_i P_{ij} = \sum_j P_{ij} = 1,\ P_{ij} \in \{0,1\}$$



- B-Matching generalizes to multi-matchings (Mormon). $O(bN^3)$

$$\max_P tr\left(P^T A\right)$$

$$\sum_i P_{ij} = \sum_j P_{ij} = b,\ P_{ij} \in \{0,1\}$$



# Matching and B-Matching

- Balanced versions of nearest neighbor & k-nearest-neighbor

$$A = \begin{bmatrix} 27 & 89 & 6 & 43 & 21 & 79 \\ 25 & 20 & 99 & 23 & 38 & 6 \\ 88 & 30 & 58 & 58 & 78 & 60 \\ 74 & 66 & 42 & 76 & 68 & 5 \\ 14 & 28 & 52 & 53 & 46 & 42 \\ 1 & 47 & 33 & 64 & 57 & 30 \end{bmatrix}$$

$$\max_P tr\left(P^T A\right)$$

$$where\, P_{ij} \in \{0,1\}$$

**Matchings** $O\left(bN^3\right)$

$$\begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

$$\sum_i P_{ij} = \sum_j P_{ij} = 1$$

$$\begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 & 1 \end{bmatrix}$$

$$\sum_i P_{ij} = \sum_j P_{ij} = 3$$

**Neighbors** $O\left(bN^2\right)$

$$\begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix}$$

$$\sum_j P_{ij} = 1$$

$$\begin{bmatrix} 0 & 1 & 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 1 & 0 \end{bmatrix}$$

$$\sum_j P_{ij} = 3$$

# Matching and B-Matching

- Matching can also be solved as a Linear Program:

$$\max_P tr\left(P^T A\right) \quad \sum_i P_{ij} = \sum_j P_{ij} = 1,\ P_{ij} \in [0,1]$$

- Still produces binary matrix even though continuous space.
- B-Matching can be implemented this way as well:

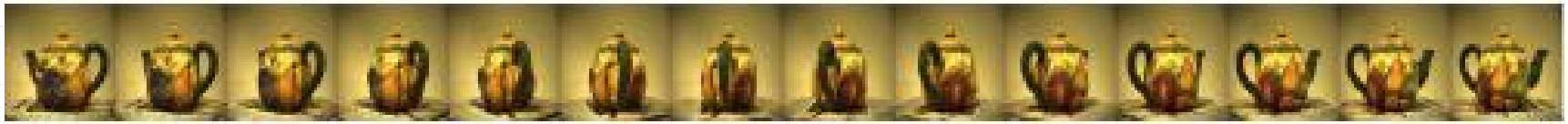
$$\max_P tr\left(P^T A\right) \quad \sum_i P_{ij} = \sum_j P_{ij} = b,\ P_{ij} \in [0,1]$$

- But, need exponentially many additional constraints to prevent non-binary solution. Add these constraints to enforce that there are no odd-length circuits in the matching. This is done efficiently via Edmonds' Blossom method.

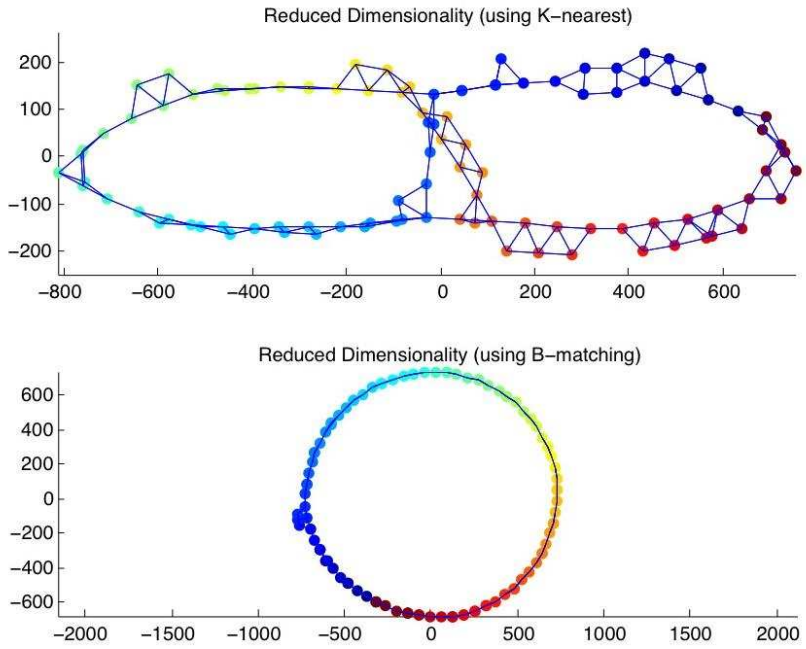
- Available Online: Goblin Algorithm or Bert's Loopy BP

# B-Matched SDE

- Visualization example: images of rotating tea pot.

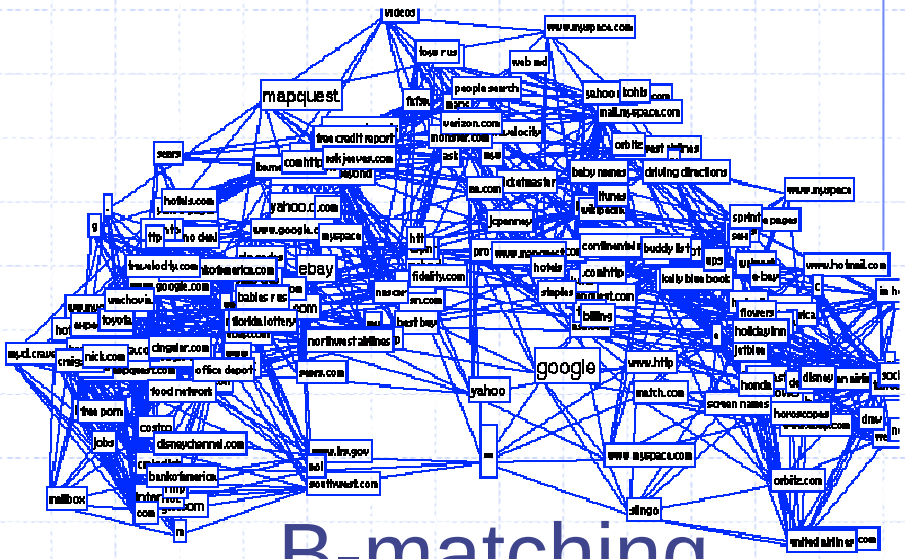


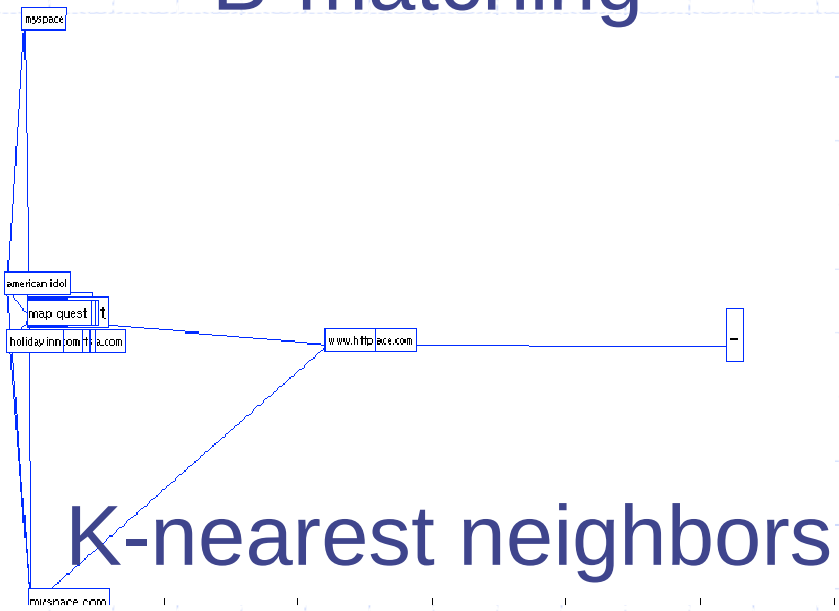
- Get affinity $A_{ij} = \exp(-||X_i - X_j||^2)$ between pairs of images
- Should get ring but noisy images confuse kNN. Greedily connects nearby images without balancing in-degree and out-degree. Get folded over ring even for various values of b (k)
- B-Matching gives clean ring for many values of b.



# B-Matched SDE

- AOL data
- Visualize search terms
- Grow initial connectivity using b-matching (instead of k-nearest neighbors)
- SDE then finds a low dimensional (2D) visualization of the data and connectivity



# Spectral Clustering

- Many clustering algorithms exist in machine learning: kmeans, expectation-maximization, linkage, etc. But: local minima, make parametric cluster assumptions
- More general: converting data to graph, cluster via cut
- Given graph (V,E), weight matrix A, normalized cut is NP $ncut(B) = \frac{\sum_{i \in B, j \in V/B} A_{ij}}{\sum_{i \in B, j \in V} A_{ij}} + \frac{\sum_{i \in V/B, j \in B} A_{ij}}{\sum_{i \in V/B, j \in V} A_{ij}}$
- Spectral Graph Theory: relax discrete optimization over y node indicator vector to continuous:

$$D_{ii} = \sum_j A_{ij} \quad d = \sum_i D_{ii} \quad d_B = \sum_{i \in B} D_{ii} \quad y(i) = \begin{cases} \sqrt{d_{V/B} / d_B d} & if\ i \in B \\ -\sqrt{d_{V/B} / d_B d} & if\ i \notin B \end{cases}$$

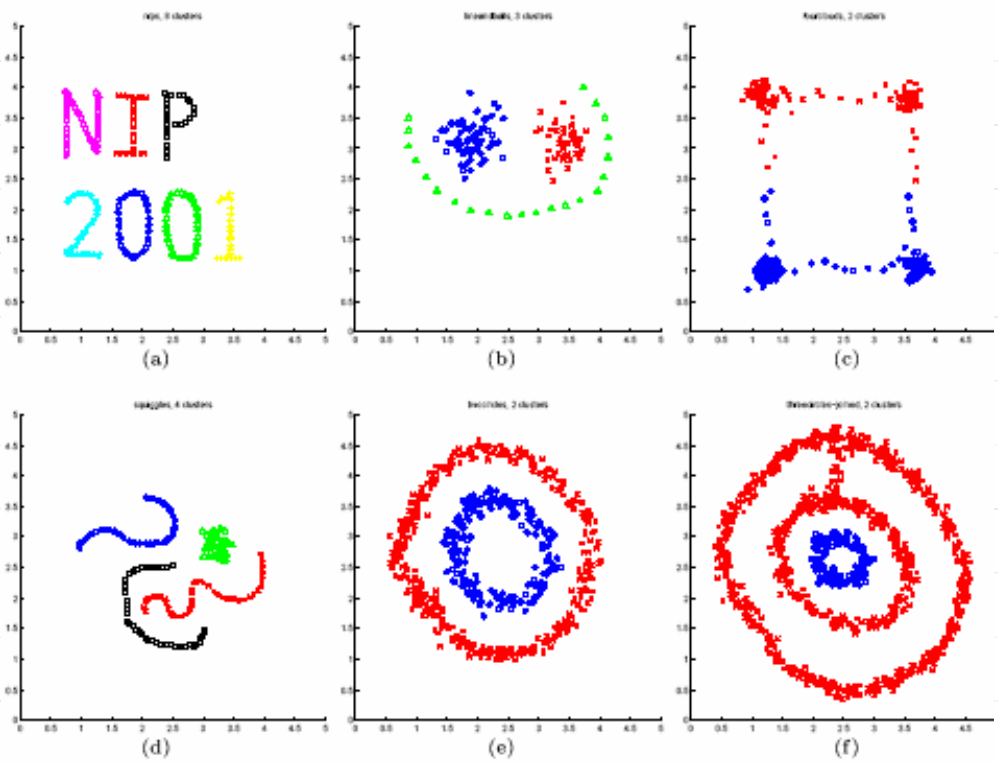
- Relaxed *global* y optimization (Shi & Malik):

$$\min_y y^T (D - A) y \quad such\ that \quad y^T D y = 1 \quad and \quad y^T D e = 0$$

- Solve for y as 2nd smallest eigenvector of: $(D - A) y = \lambda D y$

# Spectral Clustering

- Currently many slight variants of spectral clustering
- Each has varying performance
- One theme is to stabilize the clustering (Ng, Weiss, Jordan) find eigenvectors of normalized Laplacian $L=D^{-1/2}AD^{-1/2}$
- Still imperfect but works if clusters are well separated



# B-Matched Spectral Clustering

- Try to improve spectral clustering using B-Matching
- Assume w.l.og. two clusters of roughly equal size
- If we knew the labeling y = [+1 +1 +1 -1 -1 -1]
  the "ideal" affinity matrix A = $\begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 \end{bmatrix}$
  in other words...

$$A = \tfrac{1}{2}\left(yy^T + 1\right)$$

  and spectral clustering and eigendecomposition is perfect
- The "ideal" affinity is a B-Matching with b=N/2
- Stabilize affinity by finding the closest B-Matching to it:

$$\min_P \|A - P\|^2 \ such\ that \sum_i P_{ij} = \sum_j P_{ij} = \tfrac{N}{2} \ and \ P_{ij} \in \{0,1\}$$

- Then, spectral cluster B-Matching or use it to prune A

$$A^{new} = P \quad or \quad A^{new} = P \circ A$$

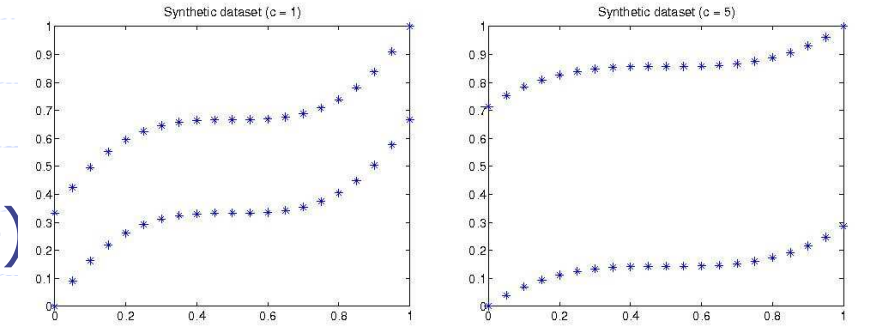
- Also, instead of B-Matching, can do kNN (lazy strawman).

# B-Matched Spectral Clustering

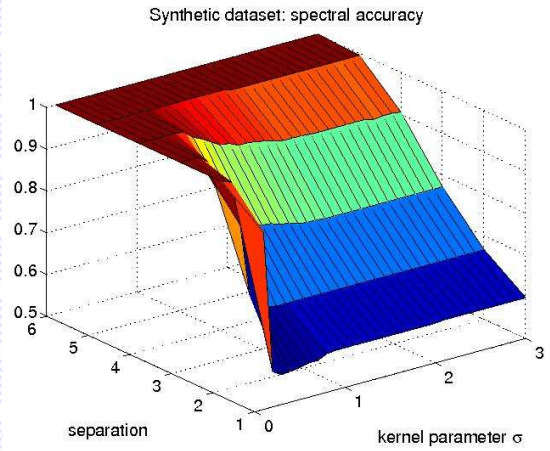
- Synthetic experiment
- Have 2 S-shaped clusters
- Explore different spreads
- Affinity $A_{ij}=\exp(-||X_i-X_j||^2/\sigma^2)$
- Do spectral clustering on

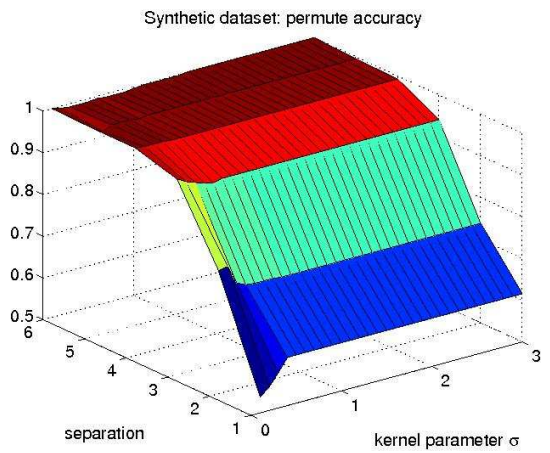
$$A \quad or \quad P \quad or \quad P \circ A$$

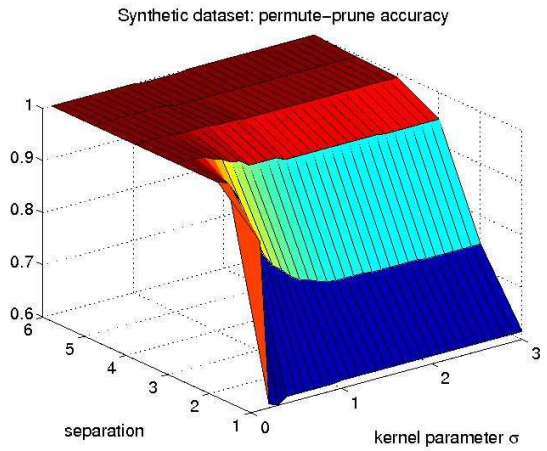
- Evaluate cluster labeling accuracy




(a) Curve separation $c = 1$. (b) Curve separation $c = 5$.

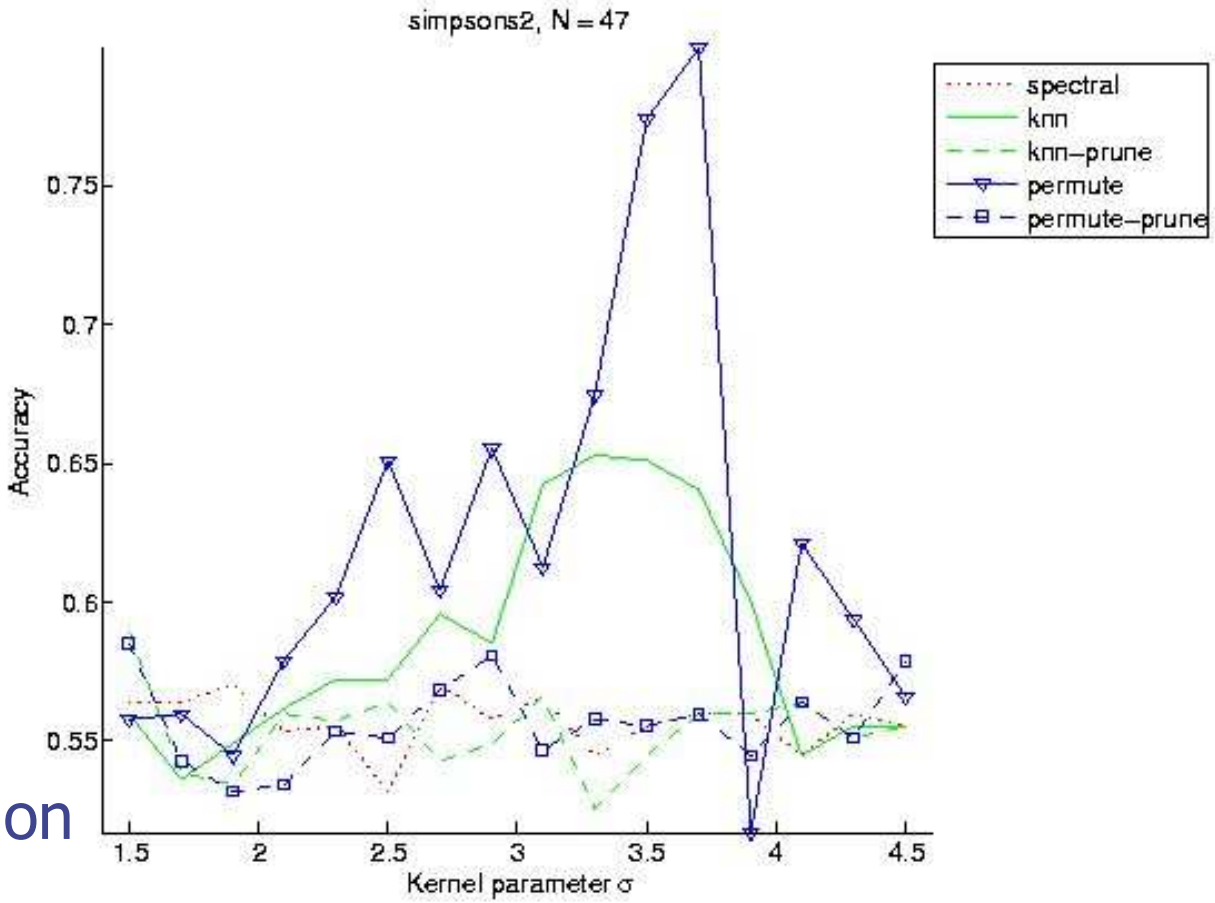


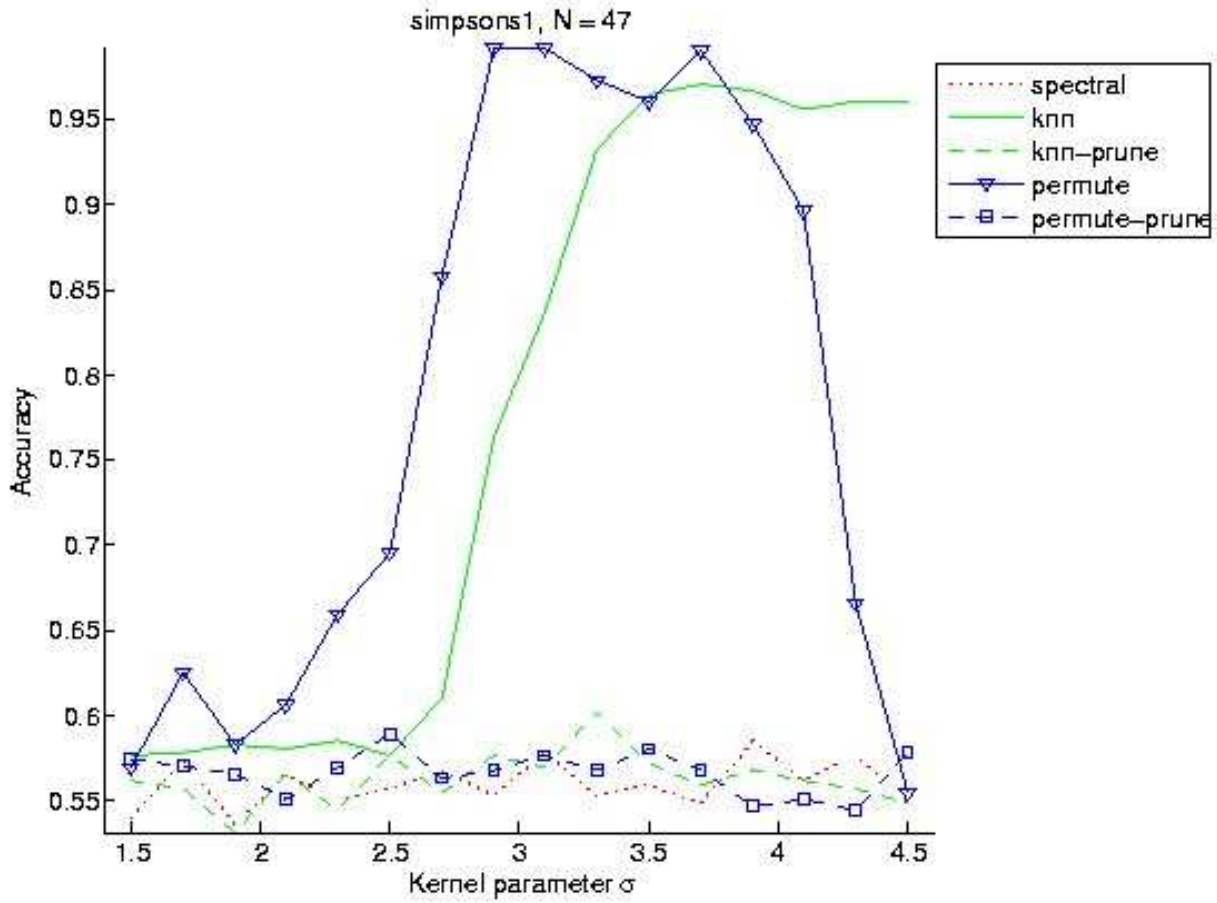
# B-Matched Spectral Clustering

- Clustering images from real video with 2 scenes in it.
- Accuracy is how well we classify both scenes
- Evaluate also with kNN
- Only adjacent frames have high affinity
- BMatching does best since it boosts connection to far frames



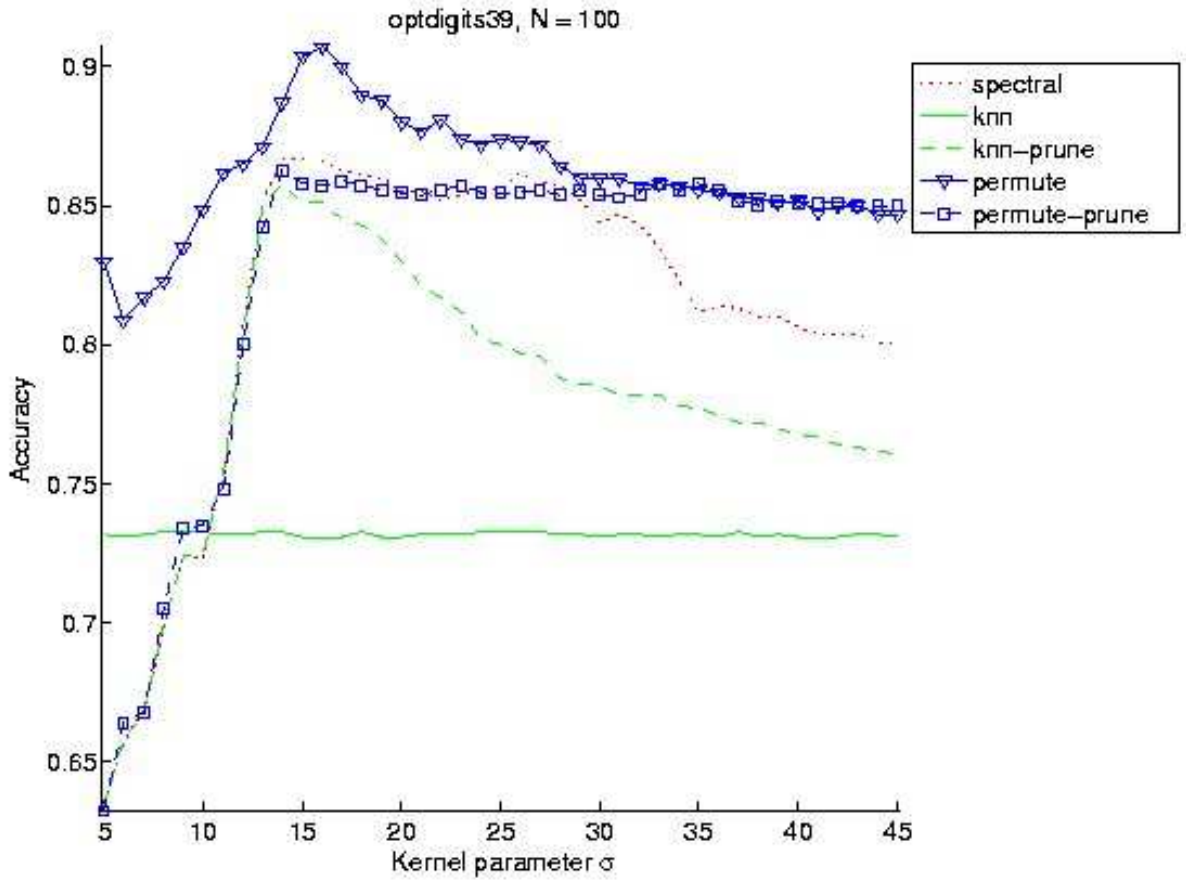
# B-Matched Spectral Clustering

- Clustering images from same video but 2 other scenes



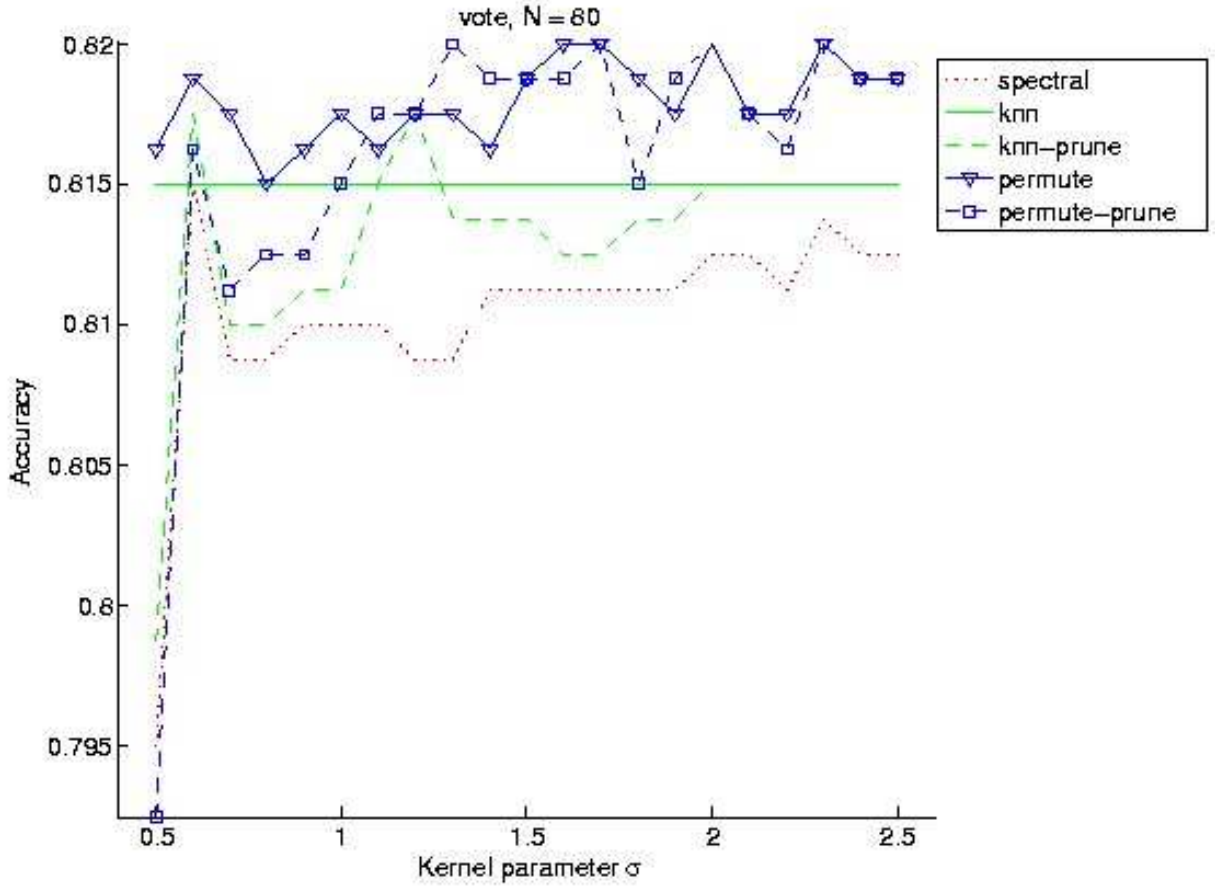
# B-Matched Spectral Clustering

- Unlabeled classification via clustering of UCI Optdigits data



# B-Matched Spectral Clustering

- Unlabeled classification via clustering of UCI Vote dataset

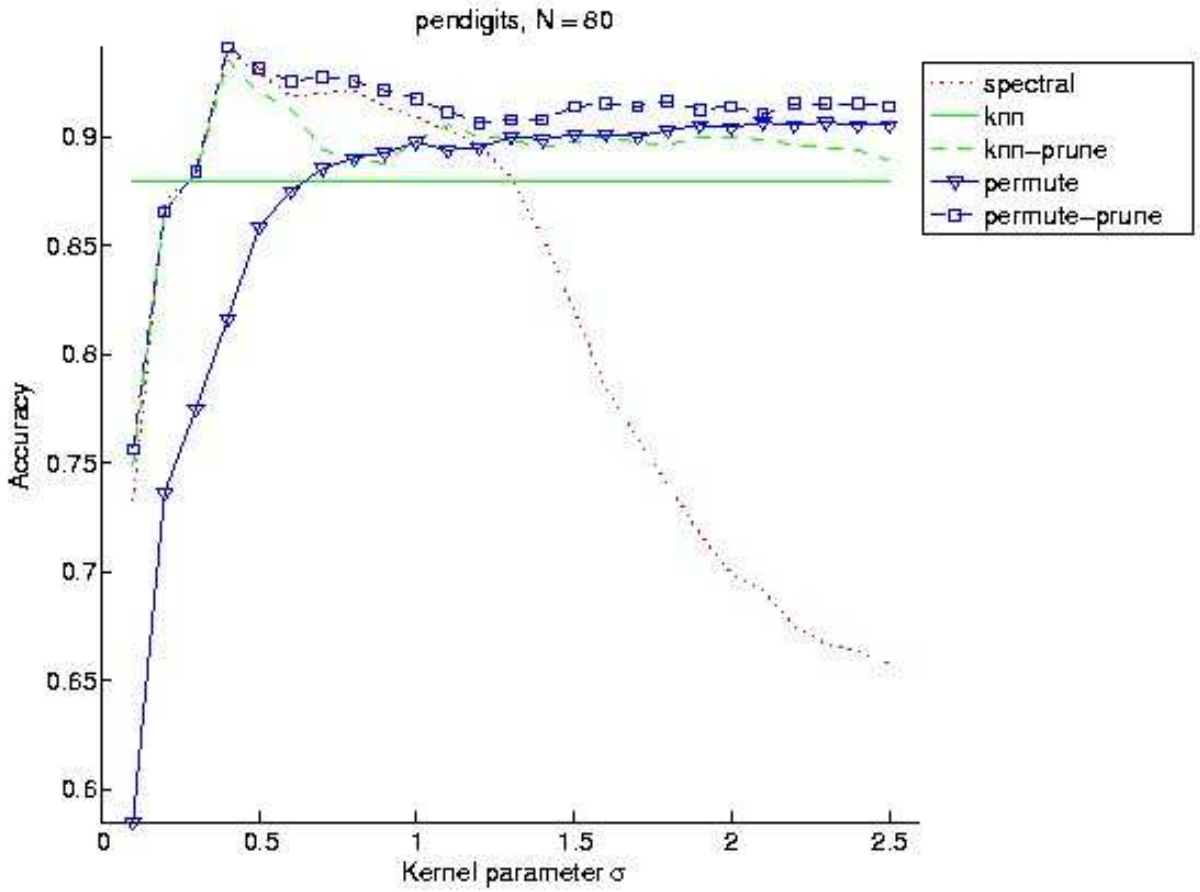


# B-Matched Spectral Clustering

- Classification accuracy via clustering of UCI Pendigits data

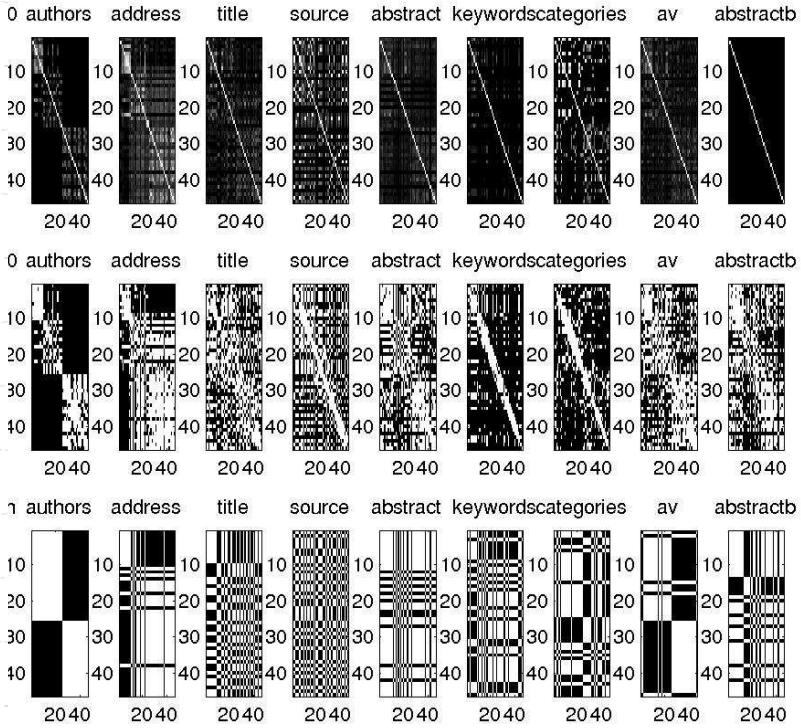
- Here, using the B-Matching to just prune A is better

  kNN always seems a little worse…



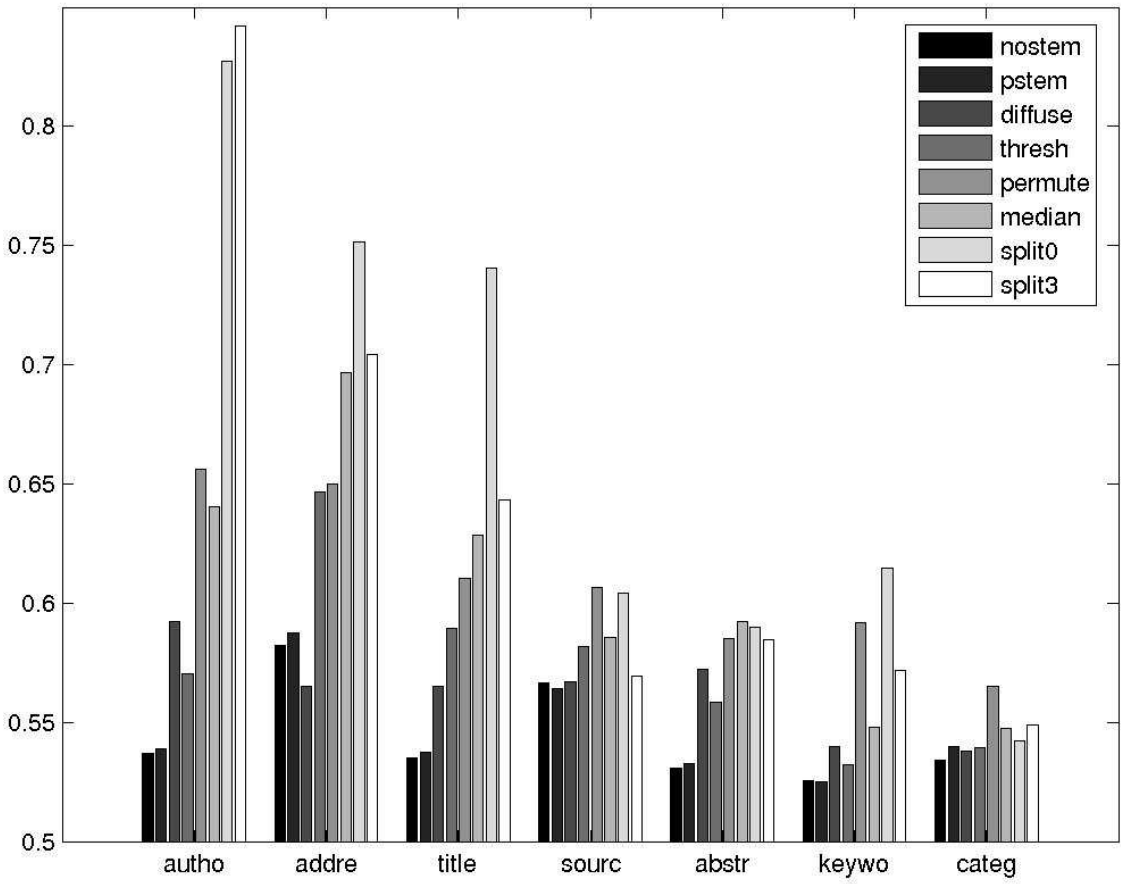
# B-Matched Spectral Clustering

- Applied method to KDD 2005 Challenge. Predict authorship in anonymized BioBase pubs database
- Challenge gives many splits of N~100 documents
- For each split, find NxN matrix saying if documents i & j were authored by same person (1) or different person (0)
- Documents have 8 fields
- Compute affinity A for each field via text frequency kernel
- Find B-Matching P
- Get spectral clustering y and compute ½ ($yy^T$+1)



# B-Matched Spectral Clustering

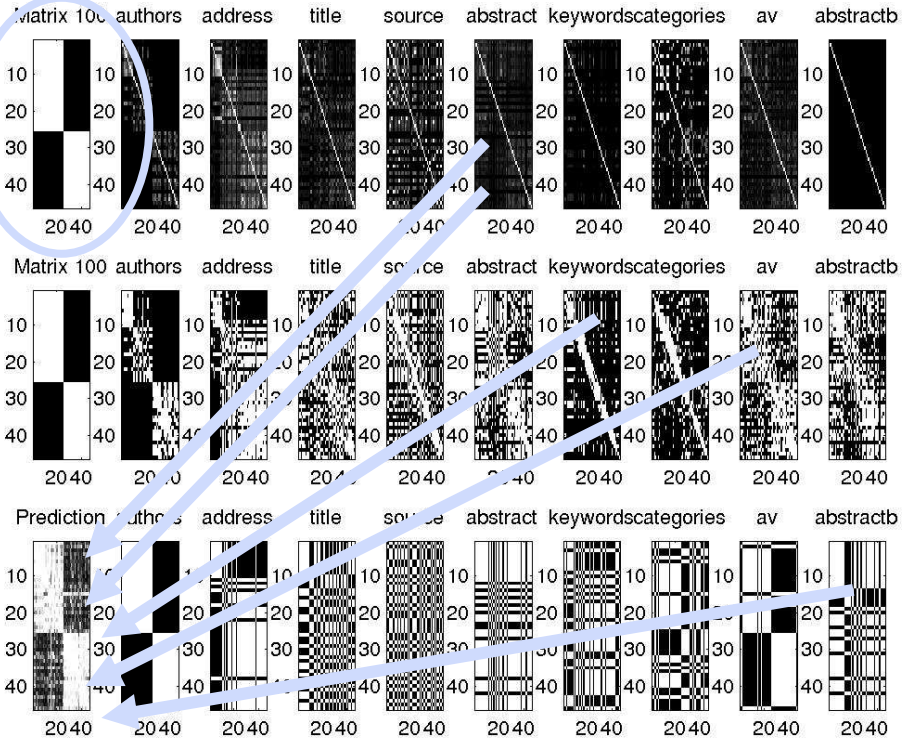
- Accuracy is evaluated using the labeled true same-author & not-same-author matrices
- Explored many processings of the A matrices. Best accuracy was by using the spectral clustered values of the P matrix found via B-Matching



# B-Matched Spectral Clustering

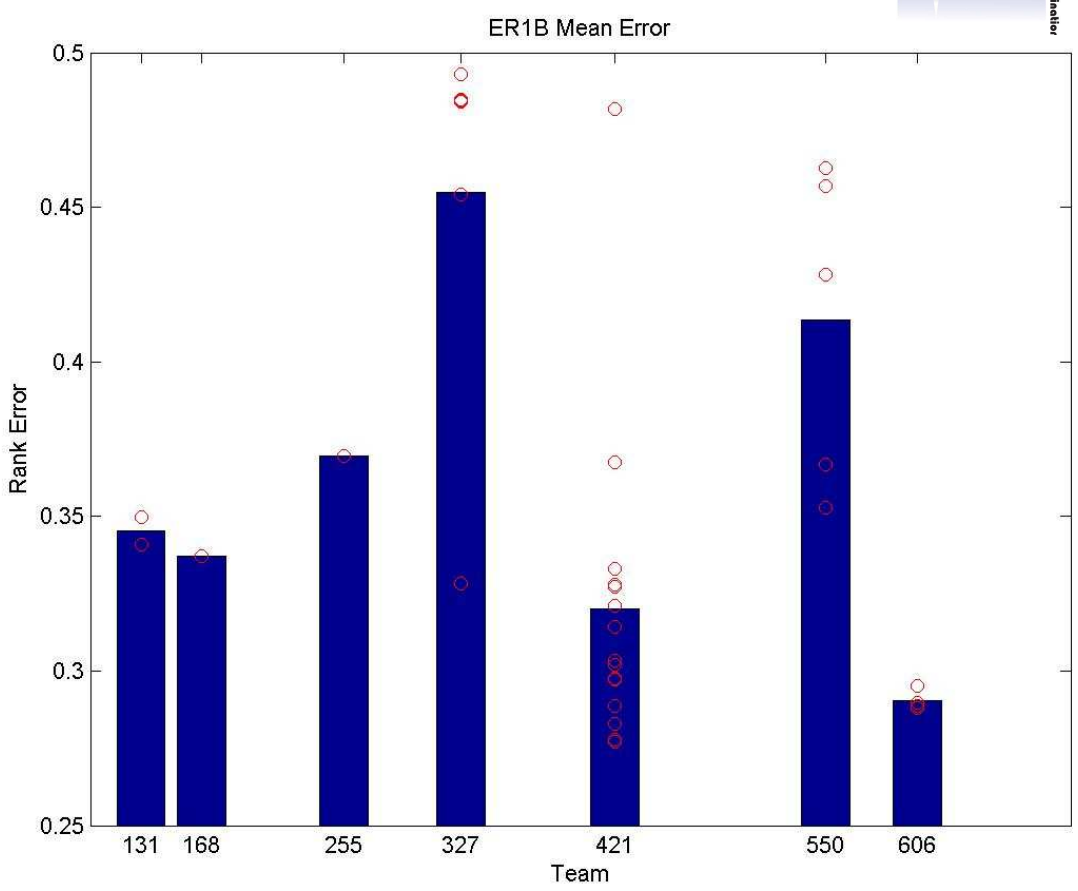
- Merge all the 3x8 matrices into a single hypothesis using an SVM and a quadratic kernel. SVM is trained on labeled data (same author, not same author matrices).

- For each split, we get a single matrix of same-author and not-same-author which was uploaded to KDD Challenge anonymously



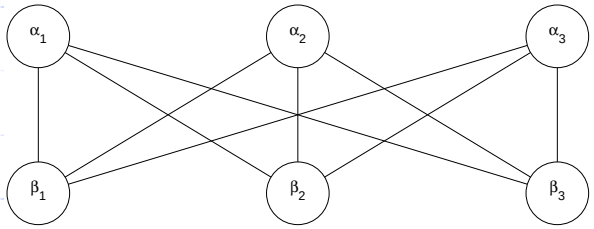
# B-Matched Spectral Clustering

- Total of 7 funded teams attempted this KDD Challenge task and were evaluated by a rank error
- Double-blind submission and evaluation
- Our method had lowest average error

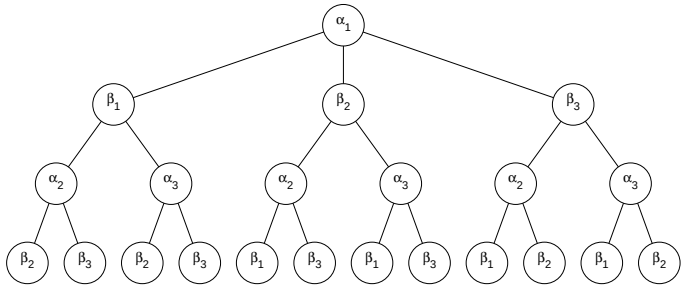


# B-Matching Belief Propagation

- Loopy belief propagation: standard Bayes net inference tool
- Not guaranteed except for Gaussian graphical models.
- Pass max-product messages between cliques until settle
- Can implement matching as loopy BP on bipartite graph.
  $\alpha(i)$ row variable = j à $P_{ij}=1$
  $\beta(j)$ col variable = i à $P_{ij}=1$
  have cliques $\phi$ representing $A_{ij}$ weights
  cliques $\varphi$ for $\alpha$ and $\beta$ agreement



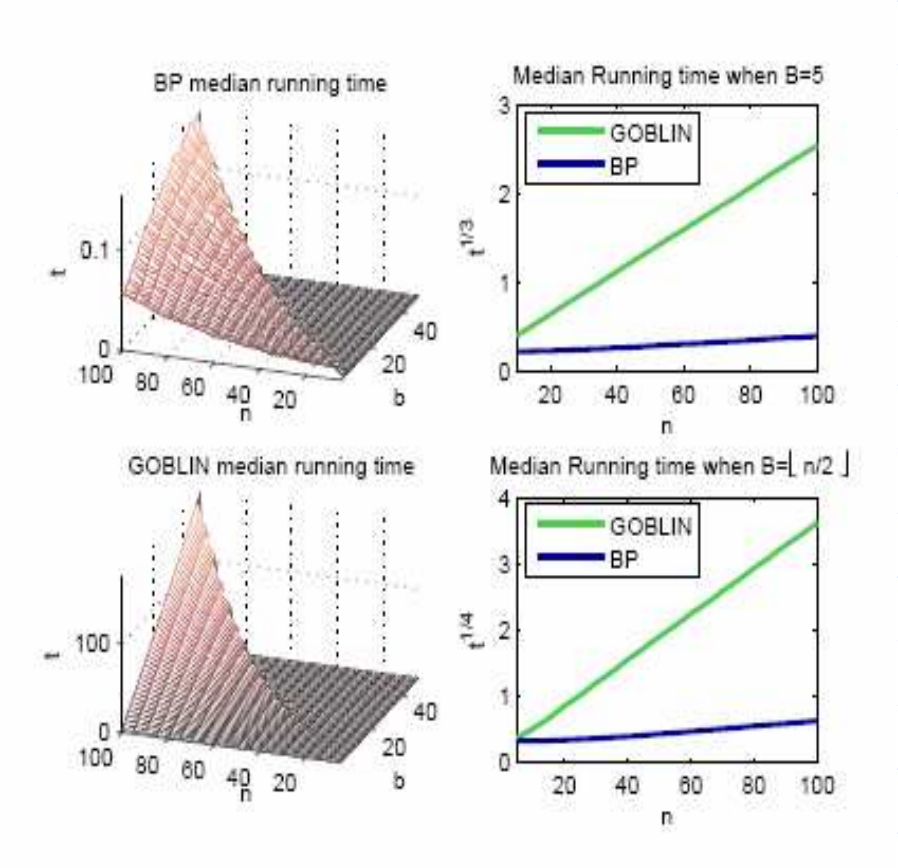

- By unrolling the message passing in time, get a tree can *guarantee* convergence.




- Upper bound on # iterations of message passing < $O(1/\varepsilon)$
  $\varepsilon$ = cost between 1st & 2nd best permutation

# B-Matching Belief Propagation

- Belief propagation drastically outperforms fast classical maximum weight b-matching code.



# B-Matching Classifier

- Belief propagation as a counterpart to kNN for classification

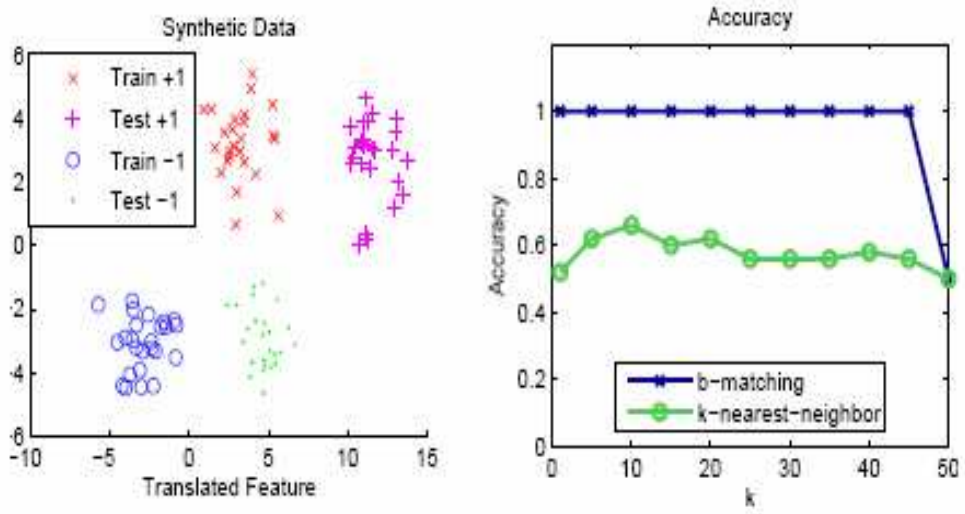



Figure 6: Left: Synthetic classification data where the test data is translated to the right along the $x$-axis. Right: Accuracy for $k$-nearest neighbor versus $b$-matching for various $k$ (or $b$).

# B-Matching Classifier

- Belief propagation as a counterpart to kNN for classification

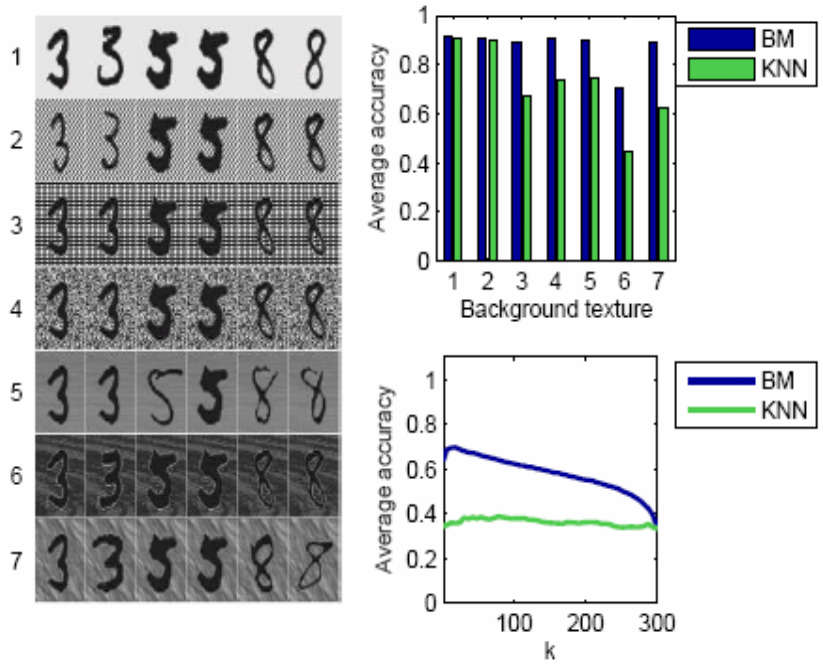


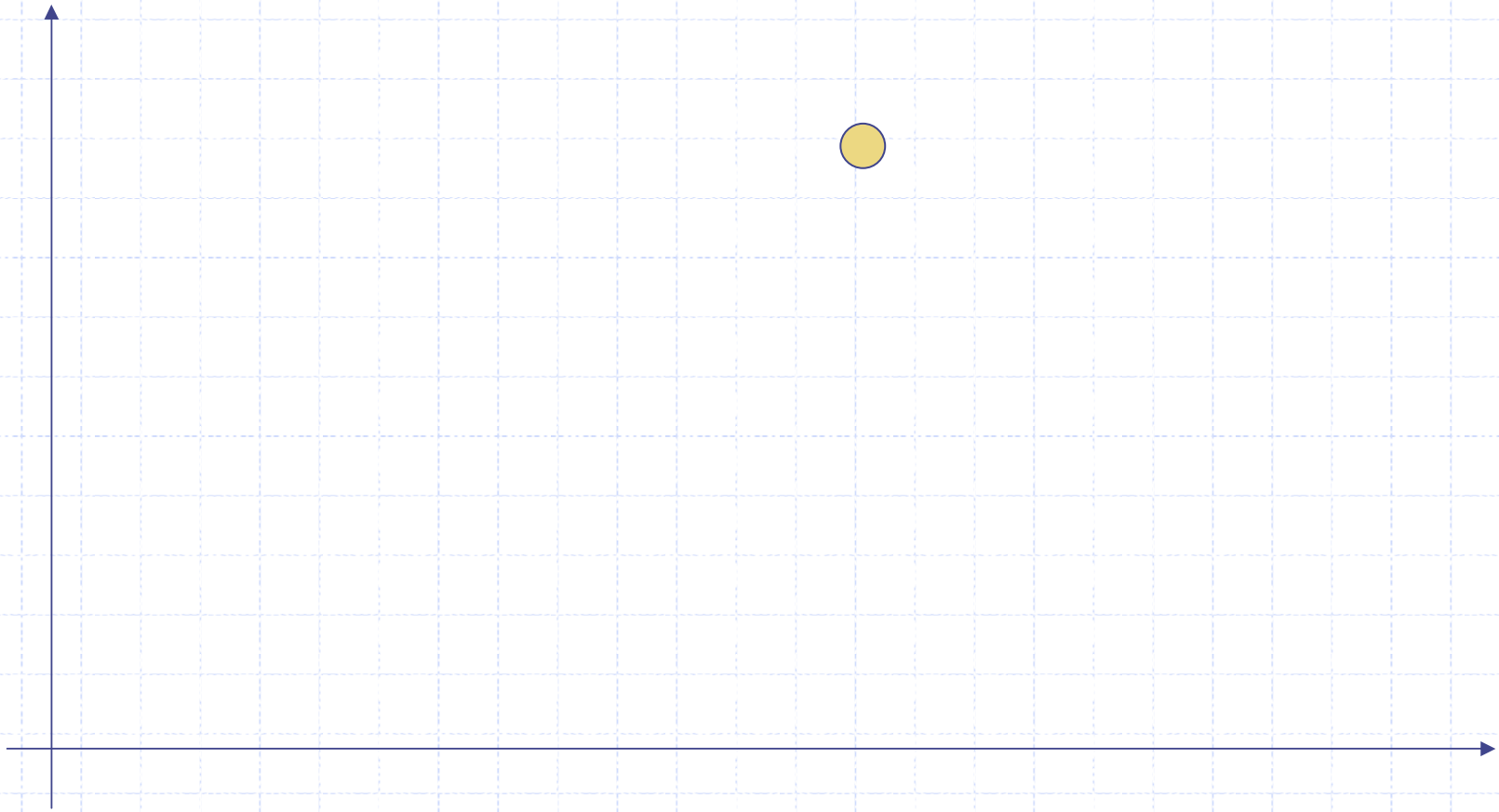

Figure 7: (Left) Examples of digits placed on backgrounds: 1-unaltered, 2-diagonal lines, 3-grid, 4-white noise, 5-brushed metal, 6-wood, 7-marble. (Top right) Average accuracies over 20 random samplings for optimal setting of $b$ or $k$. (Bottom right) Average accuracy on wood background for various settings of $b$ or $k$.

# Outline

- Why? Classification, Dimensionality Reduction, Clustering,
- Greedy Connections: Nearest Neighbors & kNN
- Less Greedy Connections: Maximum Spanning Trees
- Less Greedy Connections: Matching & b-Matching
- Bayesian Connections: Distributions over Trees
- Bayesian Connections: Distributions over Out-Trees

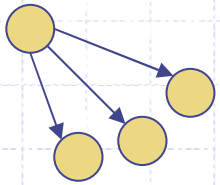
# Motivation: Tree Data (Phylogeny)

- Consider evolution of organism with scalar features (x1,x2)



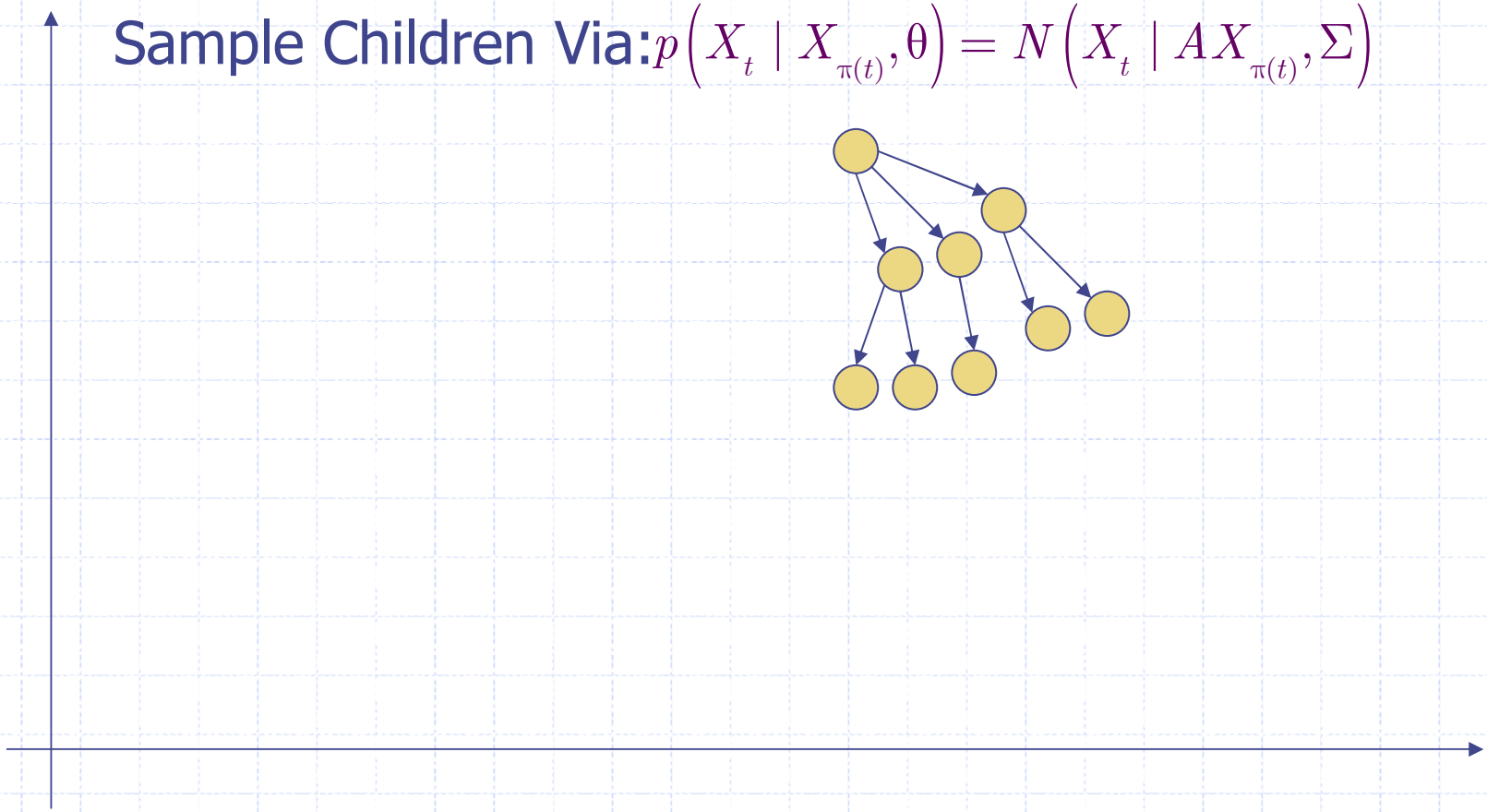
# Motivation: Tree Data (Phylogeny)

- Consider evolution of organism with scalar features (x1,x2)
  - Sample Children Via: $p\left(X_t \mid X_{\pi(t)}, \theta\right) = N\left(X_t \mid AX_{\pi(t)}, \Sigma\right)$



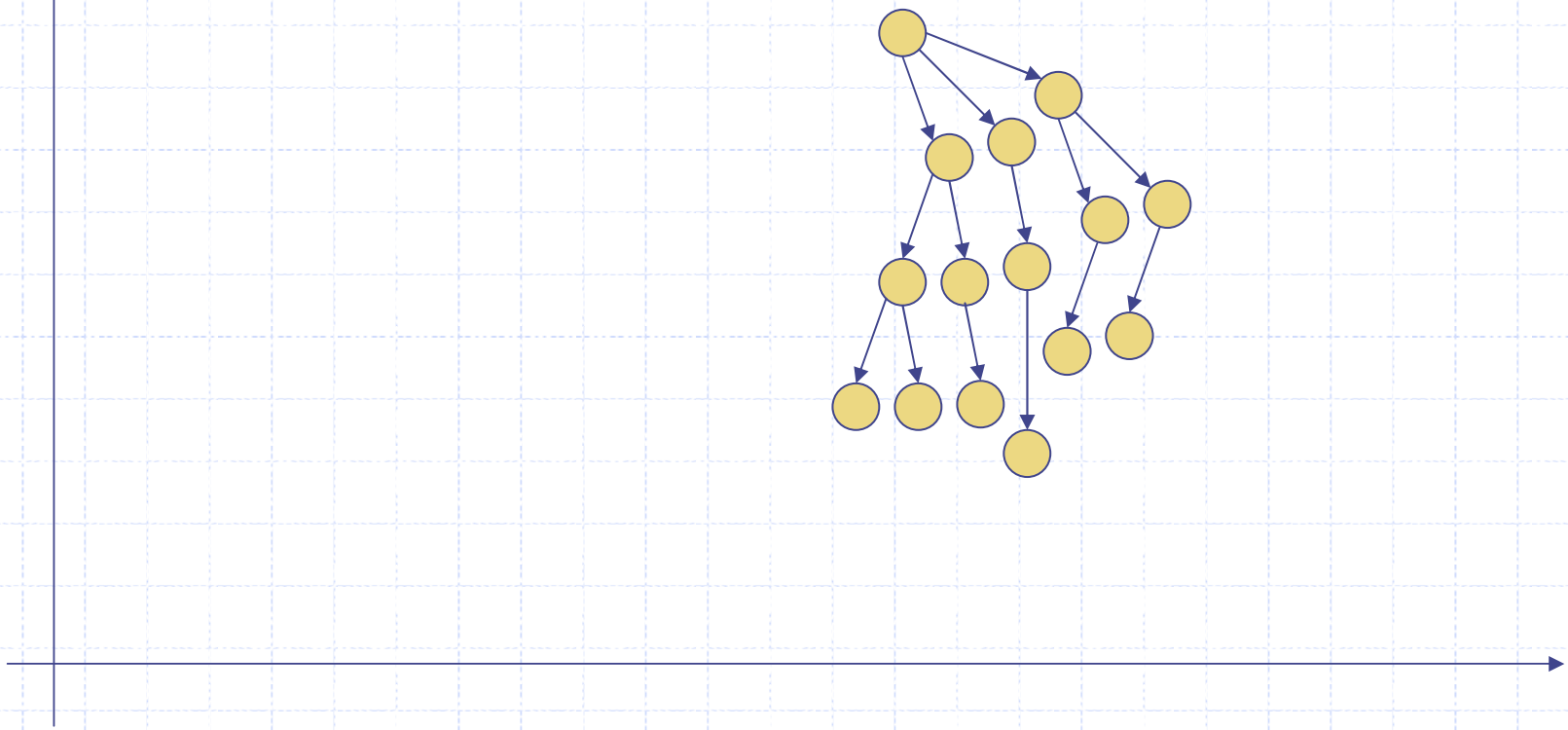
# Motivation: Tree Data (Phylogeny)

- Consider evolution of organism with scalar features (x1,x2)
  - Sample Children Via: $p\left(X_t \mid X_{\pi(t)}, \theta\right) = N\left(X_t \mid AX_{\pi(t)}, \Sigma\right)$



# Motivation: Tree Data (Phylogeny)

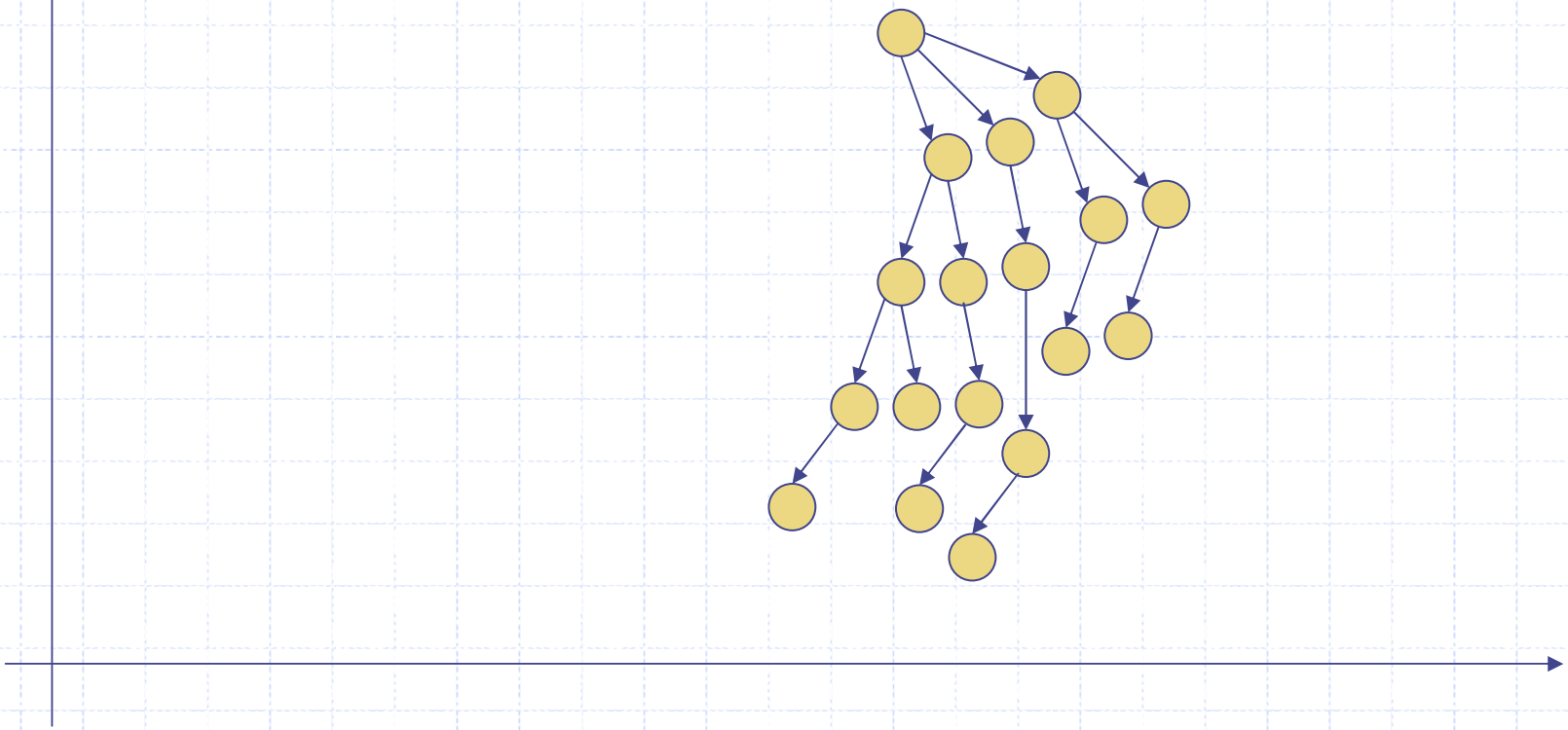
- Consider evolution of organism with scalar features (x1,x2)
  - Sample Children Via: $p\left(X_t \mid X_{\pi(t)}, \theta\right) = N\left(X_t \mid AX_{\pi(t)}, \Sigma\right)$



# Motivation: Tree Data (Phylogeny)

- Consider evolution of organism with scalar features (x1,x2)

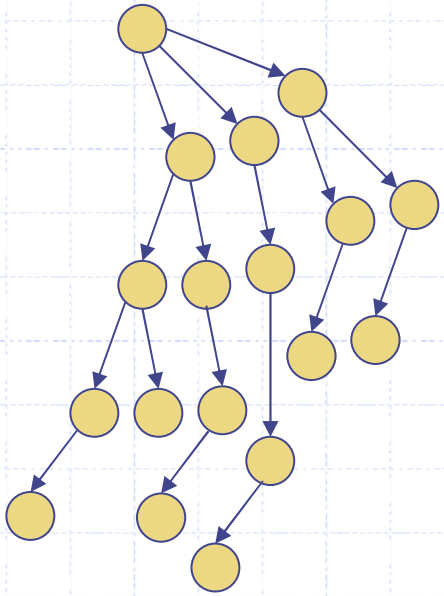
Sample Children Via: $p\left(X_t \mid X_{\pi(t)}, \theta\right) = N\left(X_t \mid AX_{\pi(t)}, \Sigma\right)$



# Motivation: Tree Data (Phylogeny)

- Consider evolution of organism with scalar features (x1,x2)

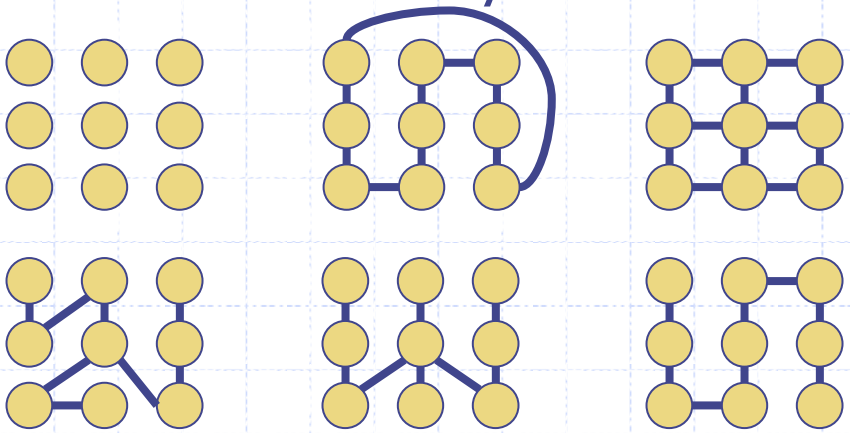
Sample Children Via: $p\left(X_t \mid X_{\pi(t)}, \theta\right) = N\left(X_t \mid AX_{\pi(t)}, \Sigma\right)$



What is likelihood of this dataset if parents and children not ordered: {x1,x2,x3,…,xN}? Not IID…

# Distributions on Graphs/Trees

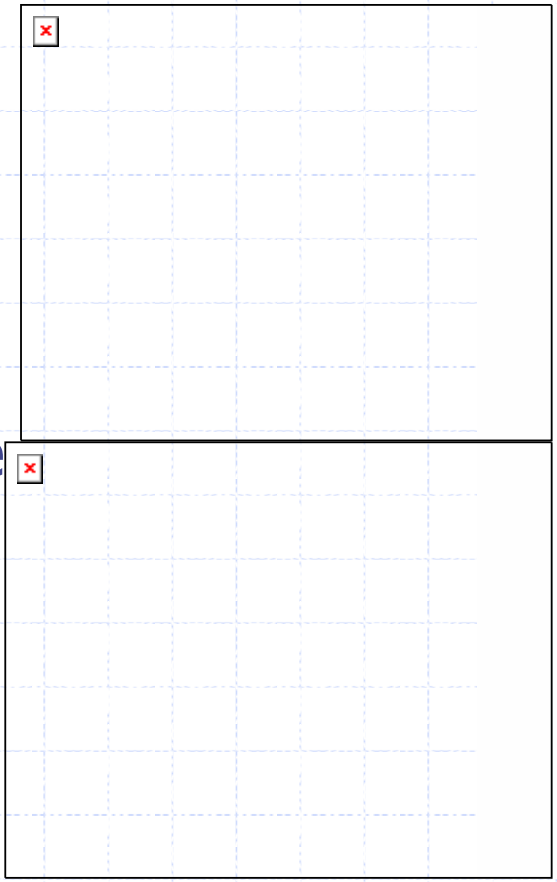
- But, distributions over structure are nasty…
- For n nodes we have $2^{n(n-1)/2}$ undirected graphs
- Cayley's Formula: there are $n^{n-2}$ undirected trees

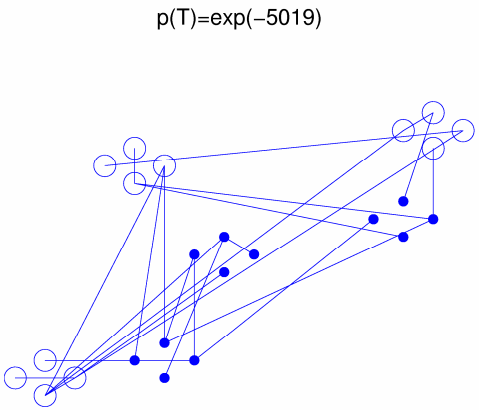


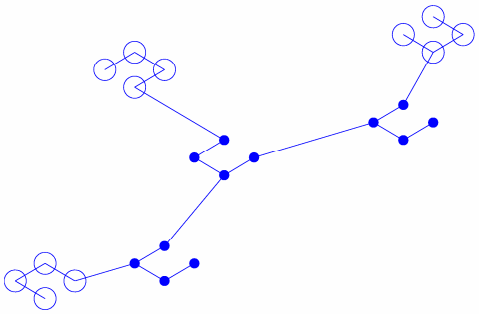
- Graph theory (West) optimizing over trees is $O(n^3)$.
- Exponential family: efficient distribution over tree structures is possible with finite sufficient-statistics.
- Therefore, we will restrict ourselves to distributions over tree structures on cloud of data points…
- Bayesian inference with tree structure as random var.

# Distributions on Trees

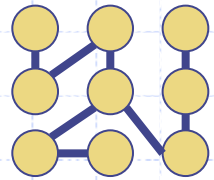
- Consider distribution $p(\mathcal{T})$ over trees connecting datapoints. (unlike standard ML trees over random vars)
- Here, edges imply dependence between $(x_1, x_2, y)$ datapoints.
- In phylogenetic trees edges indicate close or slight DNA mutation.
- Assuming unknown tree structure implies data was sampled in a tree sequence. Each parent generates children that depend on it.





# Distributions on Trees



- Standard learning assumes that points are iid, independent identically distributed.

$$p\left(X_1,\ldots,X_T \mid \Theta\right) = \prod_{t=1}^{T} p\left(X_t \mid \Theta\right)$$

- Often, iid is too restrictive, only need exchangeability i.e. likelihood invariant to permutation of $\{X_1,\ldots,X_T\}$.
- Assuming a tree structure implies datapoints are tdid, tree dependent identically distributed. If we knew the tree structure, we would have:

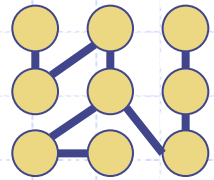
$$p\left(X_1,\ldots,X_T \mid \mathcal{T},\Theta\right) = \prod_{t=1}^{T} p\left(X_t \mid X_{\pi(t)},\Theta\right)$$

- Tree structure is unknown, treat as R.V. and use Bayes:

$$p\left(\mathcal{T} \mid X_1,\ldots,X_T\right) = \frac{\prod_{t=1}^{T} p\left(X_t \mid X_{\pi(t)}\right) p\left(\mathcal{T}\right)}{p\left(X_1,\ldots,X_T\right)}$$

ß n.b. θ omitted

# Distributions on Trees



- Assume some choices for the likelihood and prior:

$$p(\mathcal{T}) = const \qquad p(root \mid \{\}) = const \qquad p\left(X_t \mid X_{\pi(t)}\right) = p\left(X_{\pi(t)} \mid X_t\right)$$

plugging in...

$$p\left(\mathcal{T} \mid X_1,\ldots,X_T\right) = \frac{\prod_{t=1}^{T} p\left(X_t \mid X_{\pi(t)}\right) p(\mathcal{T})}{p\left(X_1,\ldots,X_T\right)} = \frac{1}{Z}\prod_{uv\in\mathcal{T}} \beta_{uv}$$

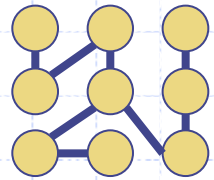
where we have a TxT symmetric matrix of edge weights between all pairs of nodes $\beta_{uv} = p\left(X_u \mid X_v\right) = \beta_{vu}$ and $\beta_{uu} = 0$

- Explicitly, partition function Z needs sum over all $T^{T-2}$ trees!
- Instead use Kirchoff's Matrix Tree Theorem which is $O(T^3)$.

$$Z = \left|Q(\beta)\right| \quad where \qquad Q_{uv}(\beta) = \begin{cases} -\beta_{uv} & u \neq v \in [1,(T-1)] \\ \sum_{w=1}^{T} \beta_{vw} & u = v \in [1,(T-1)] \end{cases}$$

- Z is the determinant of the weighted graph Laplacian Q
- Have valid distribution over tree structures from symmetric conditional/$N(X_u|X_v)$/kernel/rbf/distance between points.

# Distributions on Trees



- Given posterior over tree structures: $p(\mathcal{T} \mid \mathcal{X}) = Z^{-1} \prod_{uv \in \mathcal{T}} \beta_{uv}$
- Can compute its partition function (Matrix Tree Theorem) via the determinant of Q(β)
- Can sample from this distribution (MCMC). Parent à Child.
- Can quickly compute its argmax (Kruskal) just run maximum weight spanning tree on β
- Can compute expectations of functions on edge weights.
- For additive or multiplicative fns. on edges of trees have:

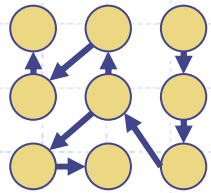
$$f(\mathcal{T}) = \sum_{uv \in \mathcal{T}} f_{uv} \qquad E_{p(\mathcal{T}|\mathcal{X})}\{f(\mathcal{T})\} = tr\left(Q(f \circ \beta) Q^{-1}(\beta)\right)$$

$$g(\mathcal{T}) = \prod_{uv \in \mathcal{T}} g_{uv} \qquad E_{p(\mathcal{T}|\mathcal{X})}\{g(\mathcal{T})\} = \frac{\left|Q(g \circ \beta)\right|}{\left|Q(\beta)\right|}$$

circle indicates Hadamard product

- Thus, can compute interesting quantities such as expected path length between nodes, etc.

# Directed Tree Distributions



- Remove the symmetry assumption: $p\left(X_t \mid X_{\pi(t)}\right) \neq p\left(X_{\pi(t)} \mid X_t\right)$ plugging in...

$$p\left(\mathcal{T} \mid X_1,\ldots,X_T\right) = \frac{\prod_{t=1}^{T} p\left(X_t \mid X_{\pi(t)}\right) p\left(\mathcal{T}\right)}{p\left(X_1,\ldots,X_T\right)} = \frac{1}{Z}\prod_{uv\in\mathcal{T}} \beta_{uv}$$

where we have a TxT *asymmetric* matrix of edge weights between all pairs of nodes $\beta_{uv} = p\left(X_u \mid X_v\right) \neq \beta_{vu}$ and $\beta_{uu} = 0$

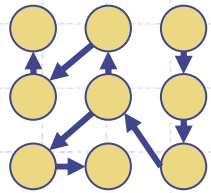
- Explicit partition function needs sum over all directed trees!
- Instead use Tutte's Directed Matrix Tree Theorem in $O(T^4)$.
- Define *asymmetric* in-Laplacian $Q^+$ and out-Laplacian $Q^-$.

$$Q^+_{uv}(\beta) = \begin{cases} -\beta_{uv} & u \neq v \in [1,T] \\ \sum_{w=1}^{T} \beta_{vw} & u = v \in [1,T] \end{cases} \qquad Q^-_{uv}(\beta) = \begin{cases} -\beta_{uv} & u \neq v \in [1,T] \\ \sum_{w=1}^{T} \beta_{wv} & u = v \in [1,T] \end{cases}$$

- Partition function is: $Z = \frac{1}{2T}\sum_{i=1}^{T} \left|m_{ii}\left(Q^+(\beta)\right)\right| + \left|m_{ii}\left(Q^-(\beta)\right)\right|$

Where $m_{ii}()$ is the minor of a matrix with i'th row and column removed.
This gives back standard Matrix Tree Theorem under symmetric $\beta$

# Directed Tree Distributions



- Can consider assymetric conditionals/kernels between pairs of points using the Directed Matrix Tree Theorem.
- For example, can consider a Gaussian model with parent-dependent noise: $p\left(X_t \mid X_{\pi(t)}\right) = N\left(X_t \mid f\left(X_{\pi(t)}\right), F\left(X_{\pi(t)}\right) + \sigma^2 I\right)$

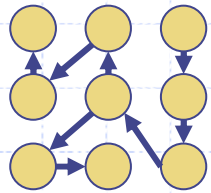
- Also, note that Z is the likelihood of the data:

$$p\left(X_1,\ldots,X_T \mid \Theta\right) = \sum_{\mathcal{T}} \prod_{t=1}^{T} p\left(X_t \mid X_{\pi(t)}, \mathcal{T}, \Theta\right) p(\mathcal{T})$$

$$= \sum_{\mathcal{T}} \prod_{uv \in \mathcal{T}} \beta_{uv}(\Theta) = Z(\Theta)$$

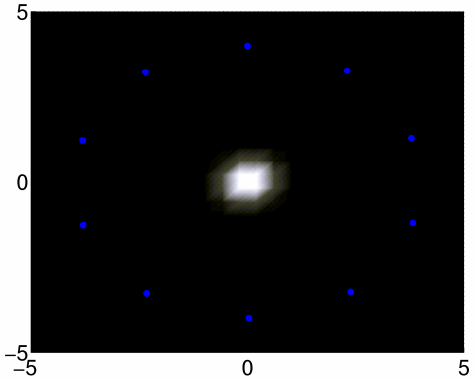
- Where, again $Z(\Theta) = \frac{1}{2T} \sum_{t=1}^{T} \left| m_{ii}\left(Q^{+}\left(\beta(\Theta)\right)\right)\right| + \left| m_{ii}\left(Q^{-}\left(\beta(\Theta)\right)\right)\right|$
- This tdid-likelihood is exchangeable yet not iid...
- This makes it more interesting for parameter estimation
- Theorem: log det of Laplacian is concave in edge weights.

# Learning Tree Distributions



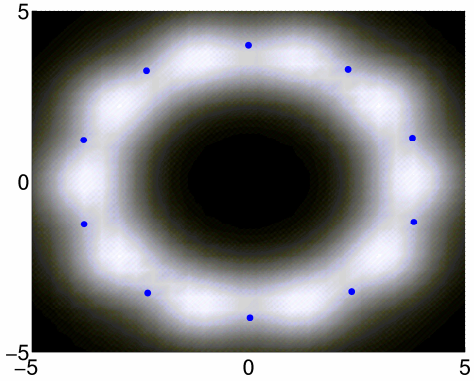
- Can consider maximizing tdid likelihood after integrating over unknown tree structure.
- Max Z(A) over A via gradient ascent by using $\frac{\partial}{\partial A}|A| = |A|A^{-1}$
- Permits tweaking pairwise conditionals/distances/kernels
- Consider unknown A matrix changing conditional or RBF

$$p\left(X_t \mid X_{\pi(t)}, \theta\right) = N\left(X_t \mid AX_{\pi(t)}, I\right)$$



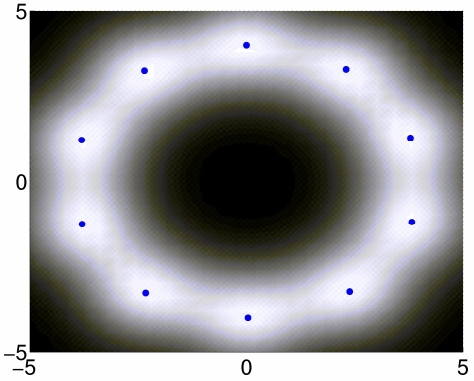

$$p\left(\mathcal{X} \mid \theta\right) = 3e^{-31}$$

$$A = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$$




$$p\left(\mathcal{X} \mid \theta\right) = 7e^{-19}$$

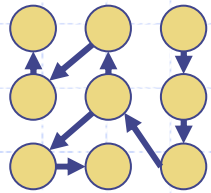
$$A = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$



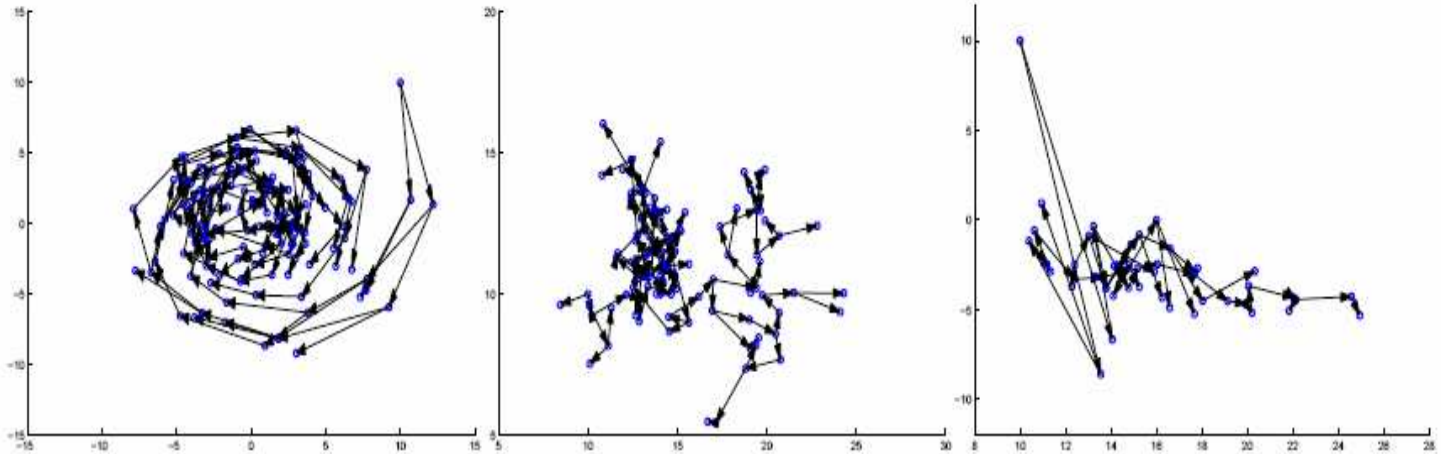

$$p\left(\mathcal{X} \mid \theta\right) = 9e^{-8}$$

$$A = \begin{bmatrix} 0.8 & 0.6 \\ -0.6 & 0.8 \end{bmatrix}$$

# Learning Tree Distributions



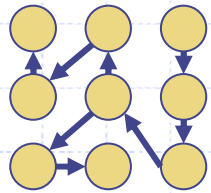
- Can uncover the most likely A evolution matrix



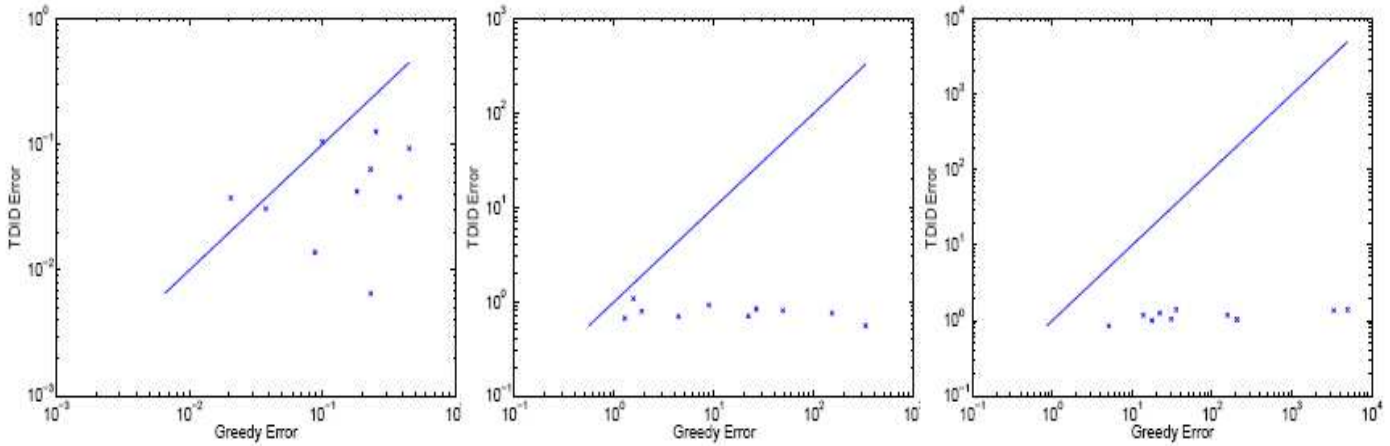
a) $A = \begin{bmatrix} 0.72 & 0.54 \\ -0.54 & 0.72 \end{bmatrix}$ b) $A = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$ c) $A = \begin{bmatrix} 1.1 & 0.3 \\ -0.3 & -0.5 \end{bmatrix}$

Figure 2: Synthesized out-trees with varying $A$ evolution matrix.

# Learning Tree Distributions



- Can uncover the most likely A evolution matrix




a) $D = 1$ b) $D = 2$ c) $D = 3$

Figure 4: Scatter plots of Frobenius norm error rates when reconstructing $A^{true}$ from population data. The vertical axis is the out-tree directed tdid likelihood maximization while horizontal axis is the greedy MWST estimation.