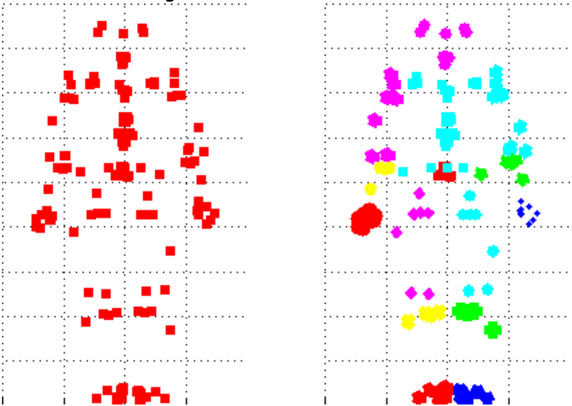
Flexible Priors for Exemplar-based Clustering

Daniel Tarlow[1], Richard Zemel[1], and Brendan Frey[2]

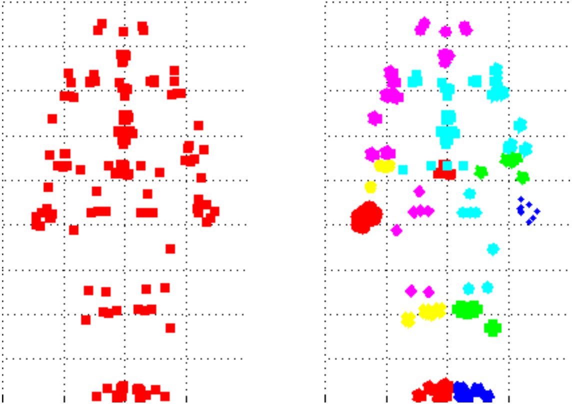
[1] University of Toronto, Department of Computer Science

[2] University of Toronto, Probabilistic and Statistical Inference Group

Why cluster data?



Why cluster data?



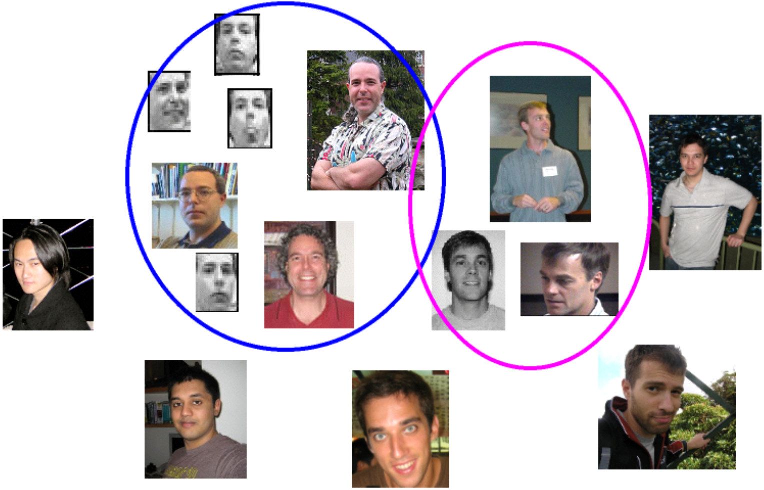
- **We believe there is underlying structure in data that we want to recover.**

Motivating Example

Suppose we want to cluster faces in a personal photo collection.

Motivating Example

Suppose we want to cluster faces in a personal photo collection.



Motivating Example

Suppose we want to cluster faces in a personal photo collection.

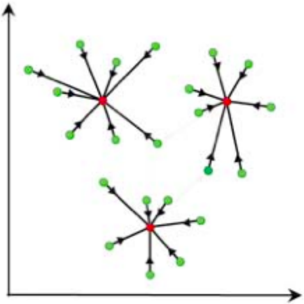
Goal of this work

- General and natural ways of expressing a clustering problem.
  - More complex, possibly non-metric similarity measures.
  - Natural way to express prior knowledge about size of clusters.

Building Blocks

Exemplar-based clustering

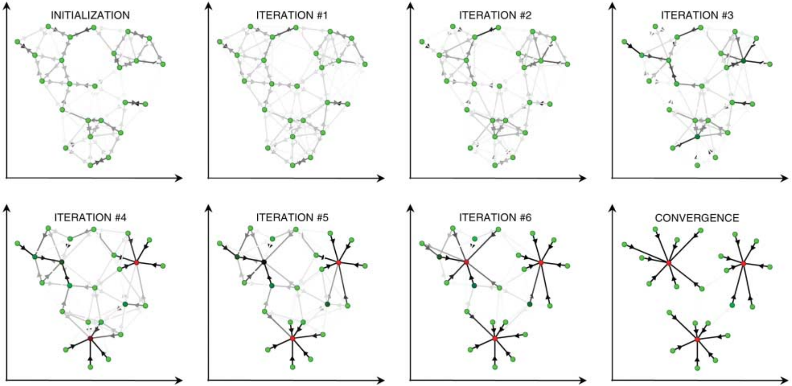
- Can use general similarity measures (e.g., [Dueck & Frey, 2008]).
- No continuous parameter estimation, only combinatorics.



Exemplar-based Clustering (cont'd)

Combinatorics can be dealt with efficiently.

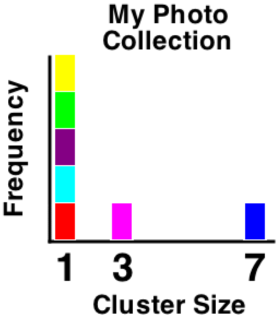
E.g., Max-product (Affinity Propagation [Frey & Dueck, 2007])

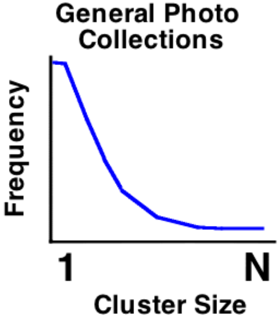


Model selection is awkward.

Flexible priors over cluster sizes

- Express knowledge about cluster size distributions without biasing problem [Welling, 2006].
- Includes Dirichlet Process and Pitman-Yor Process priors.
- Can be used in place of ad-hoc model selection parameters (e.g., specifying k).





Comparison

	Automatically Choose k	Flexible Priors	Non-metric Likelihoods
k-means			
Affinity Propagation	✓		✓
Dirichlet Process Mixture Models	✓	✓	
This Work	✓	✓	✓

Exemplar-based Dirichlet Process Mixture Model

Generative Model (Notation)

Notation

- N data points.
- $C = \{c_1, \ldots, c_N\}$
  Assignment of points to partitions. Points with the same value of c_i belong to the same partition.
  E.g. $c_1 = 7, c_2 = 7, c_3 = 9, c_4 = 9, c_5 = 7 \rightarrow$ [1,2,5], [3,4]
- $E = \{e_1, \ldots, e_N\}$
  Binary variables indicating whether each point is an exemplar.
- $X = \{x_1, \ldots, x_N\}$
  Parameter vectors describing each point.

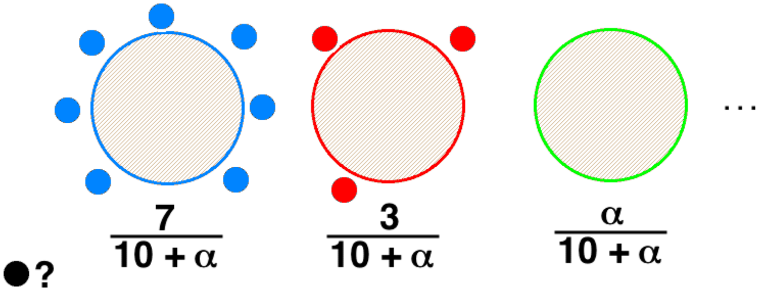
Generative Model

Generative Model:

1. Draw a partition from a Chinese Restaurant Process prior.
2. Choose an exemplar for each non-empty cluster.
3. Draw parameters for each exemplar.
4. Draw parameters for each non-exemplar.

Generative Model (1)

Draw a partition, C, from a Chinese Restaurant Process prior.

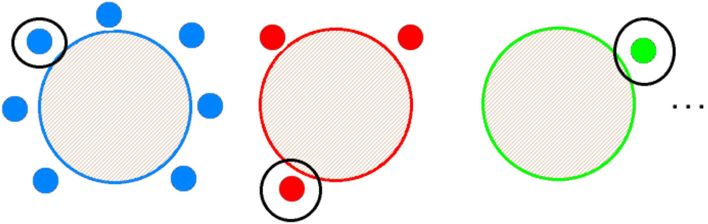


$$P(C;\alpha) = \frac{\Gamma(\alpha)}{\Gamma(N+\alpha)}\alpha^K \prod_{k=1}^{K} \Gamma(N_k)$$

where $N_k = \sum_{i=1}^{N}[c_i = k]$

Generative Model (2)

Choose an exemplar for each non-empty cluster.



$$P(E \mid C) = \prod_{k=1}^{K} \frac{1}{N_k} oneOfN(E_k)$$

where $E_k = \{e_i \mid c_i = k\}$.

Generative Model (3,4)

Draw parameters, X_e, for each exemplar from a base distribution G_0:

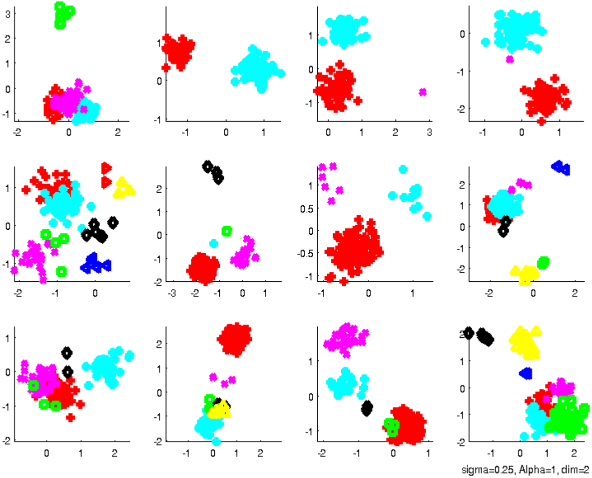
$$P(X_e; G_0) = \prod_{i=1}^{N} P(x_i; G_0)^{[e_i=1]}$$

Draw the parameters for each remaining point from a distribution parameterized by its exemplar:

$$P(X_p \mid X_e, C, E) = \prod_{i=1}^{N} \prod_{j=1, j \neq i}^{N} P(x_i \mid x_j)^{[c_i=c_j \wedge e_i=0 \wedge e_j=1]}$$

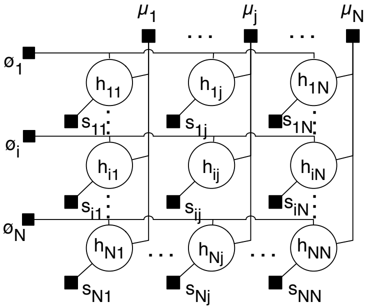
* in reality, don't need to sample or compute normalization constants.

Draws from 2D Gaussian model



Max-product (max-sum) Inference

Factor Graph



$$\phi_i(h_{i1}, \ldots, h_{iN}) = [\sum_{j=1}^{N} h_{ij} = 1]$$

$$\mu_j(h_{1j}, \ldots, h_{Nj}) = \begin{cases} 1 & \text{if } N_j = 0 \\ \frac{\Gamma(N_j)}{N_j} \cdot [h_{jj} = 1] & \text{otherwise} \end{cases}$$

$$s_{ij}(h_{ij}) = \begin{cases} P'(x_i \mid x_j)^{h_{ij}} & \text{if } i \neq j \\ (\alpha \cdot P'(x_j; G_0))^{h_{jj}} & \text{if } i = j \end{cases}$$

Representation

Subtle representational differences from [Blei & Jordan, 2005] and [Kurihara, Teh, & Welling, 2007] variational approximations to the Dirichlet Process:

- Labels always lie in $\{1, \ldots, N\}$, but not necessarily contiguous.
- Infinite. No truncation or finite approximation.
- No ordering of clusters.

Max-sum Belief Propagation in Factor Graphs

Variable, X, to factor, f_i, messages:

$$\tilde{m}_{x \to f_i}(x) = \sum_{f' \in n(x) \setminus f_i} m_{f' \to x}(x)$$

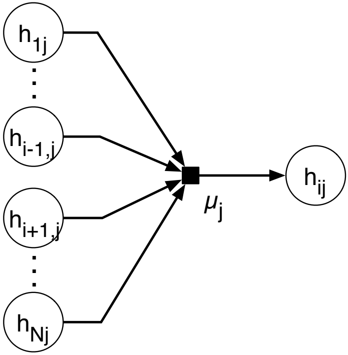
Factor, f, to variables, $X = \mathit{neighbors}(f)$, messages:

$$\tilde{m}_{f \to x}(x) = \max_{X \setminus x} \left[\log f(X) + \sum_{x' \in X \setminus x} m_{x' \to f}(x') \right]$$

Normalize messages:

$$\begin{aligned} m_{A \to B}(1) &= \tilde{m}_{A \to B}(1) - \tilde{m}_{A \to B}(0) \\ m_{A \to B}(0) &= 0 \end{aligned}$$

Factor Graph



Non-trivial calculation: outgoing messages from the μ factors.

$$\tilde{m}_{\mu_j \to h_{ij}}(h_{ij}) = \max_{h_{-ij}} \left[\log \mu_j(h_{1j}, \ldots, h_{Nj}) + \sum_{i' : i' \neq i}^{N} m_{h_{i'j} \to \mu_j}(h_{i'j}) \right]$$

Key Computation

The heart of the computations:

$$\begin{aligned} & \max_{h_{-ij}} \left[\sum_{i'} m_{h_{i'j} \to \mu_j}(h_{i'j}) + \log g(\sum_{i'} h_{i'j}) \right] \\ = \quad & \max_{h_{-ij}} \left[\sum_{i'} h_{i'j} \cdot m_{h_{i'j} \to \mu_j}(1) + \log g(\sum_{i'} h_{i'j}) \right] \end{aligned}$$

Key Computation

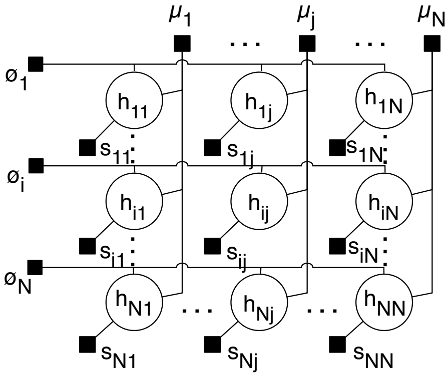
Given N_j, terms decouple:

$$m_{\mu_j \to h_{ij}}(h_{ij}; N_j) \quad = \quad \log g(N_j) + \max_{h_{-ij}} \sum_{i'} h_{i'j} \cdot m_{h_{i'j} \to \mu_j}(1)$$
$$\textbf{s.t. } \textstyle\sum_{i'} h_{i'j} = N_j$$

Simple algorithm:

1. Sort $m_{h_{i'j} \to \mu_j}(1)$ values in descending order
2. for $N_j = 1, \ldots, N$
   - Set the first N_j h_{ij}'s to be 1 and the remainder to be 0.
3. Take the value corresponding to the N_j that produced the largest value.

Remaining Computations



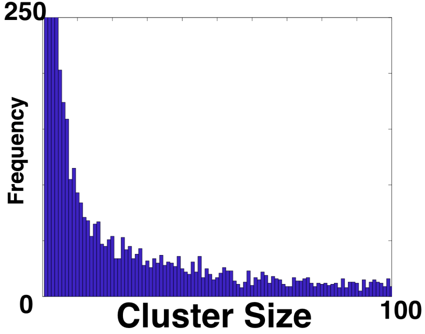
The rest of the messages are standard max-sum updates.

Experiments

Experiments

Generate 1000 synthetic data sets of 100 points each from generative model.

Synthetic data true cluster size distribution:



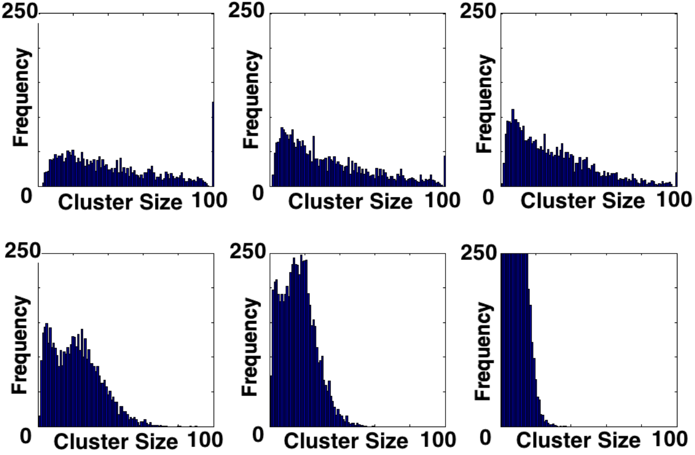
Algorithms

Algorithms

- Affinity Propagation (AP): plus preference parameter, d.
- Dirichlet Process Affinity Propagation (DPAP)
- Iterated Conditional Modes: Initialize to one large group (ICM-1) or N separate groups (ICM-N).

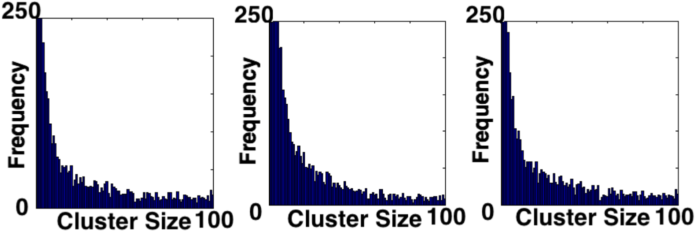
Synthetic Experiments

Affinity Propagation results for various settings of preference:



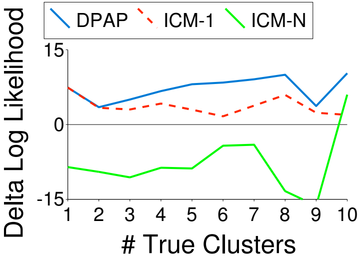
Synthetic Experiments (cont.)

ICM-1, ICM-N, and DPAP results:

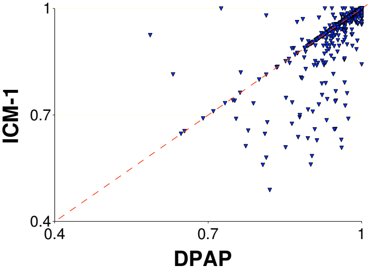


Synthetic Experiments (cont.)

Log likelihood relative to true labels log likelihood.



Rand Index against true labels



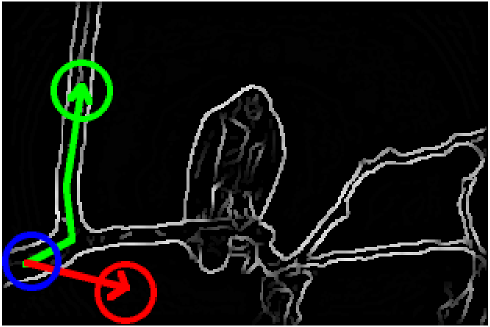
Real Experiments: Image Segmentation

Discretize into superpixels

Real Experiments: Image Segmentation (cont.)

Pairwise superpixel similarity measure combines two components:

1. Shortest path between superpixels in edge distance graph



2. Distance between mean superpixel colors

Algorithms

Give the same similarity matrix to each algorithm, vary tunable parameter(s).

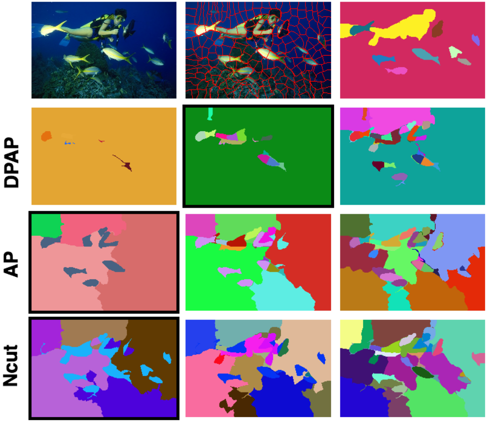
- Affinity Propagation (AP)
- Dirichlet Process Affinity Propagation (DPAP)
- Normalized Cuts (Ncut)

Real Experiments: Image Segmentation (cont.)

Results: Image Segmentation

Real Experiments: Image Segmentation (cont.)

Results: Image Segmentation



Conclusions and Future Work

Demonstrates the implicit prior of two similarity-based clustering algorithms and provides a method for making the prior explicit.

Not practical for large data sets in current state: $O(N^3)$ per iteration

- Some ideas on how to improve speed.

Image segmentation application is only proof of concept. Could improve by:

- Learning generative model for superpixel base distribution.
- Better similarity measure.

In general, it should be helpful to learn application-specific prior distributions over cluster sizes (E.g., Dirichlet Process doesn't appear right for image segmentation).

Thank you

Acknowledgments: Yee Whye Teh