

# Summarizing Distributions of Latent Structures

---

David B. Dahl: Brigham Young University

Peter Müller: University of Texas at Austin

Bayesian Nonparametric Inference: Dependence Structures & Applications

Oaxaca, Mexico

December 6, 2017

# Motivation

- In a typical Bayesian analysis, considerable effort is placed on “**fitting the model**” (e.g., sampling from the posterior) but this is **only half of the inference problem**.
- Meaningful inference also requires **summarizing the posterior distribution** of the parameters of interest.
- Posterior summaries are important for subsequent analyses or in communicating the results to a diverse audience.
- If the parameters of interest live in  $\mathbb{R}^n$ , common posterior summaries are **means** and **medians**.
- Summarizing posterior distributions of parameters with **complicated structure** is more challenging, e.g., the “average” network in the network distribution is not easily defined.
- We consider summarizing distributions of latent structures, e.g., clusterings, feature allocations, and networks.

- We present the **sequentially-allocated latent structure optimization (SALSO)** method to minimize an objective criterion to obtain a *point estimate* based on a collection of randomly-sampled *latent features*.
- SALSO is a *stochastic search* method involving a *series of micro optimizations*.
- The method can be applied to *clusterings, feature allocations, networks*, etc.
- Several objective criterion can be used, including squared error loss, absolute error loss, Binder (1978) loss, or the lower bound of the variation of information loss (Wade & Ghahramani 2017), respectively.

# Setting the Stage

---

## Example: First Clustering in MCMC Output

Clustering in *cluster label* notation:

$$c^{(1)} = (1, 2, 1, 2, 2)$$

Clustering in set *partition* notation:

$$\pi^{(1)} = \{\{1, 3\}, \{2, 4, 5\}\}$$

## Example: First Clustering in MCMC Output

Clustering in ***cluster label*** notation:

$$c^{(1)} = (1, 2, 1, 2, 2)$$

Clustering in set ***partition*** notation:

$$\pi^{(1)} = \{\{1, 3\}, \{2, 4, 5\}\}$$

Clustering as ***pairwise allocation matrix*** (i.e., adjacency matrix):

$$A(c^{(1)}) = \begin{bmatrix} 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 \end{bmatrix}$$

## Example: Second Clustering in MCMC Output

Clustering in ***cluster label*** notation:

$$c^{(2)} = (1, 1, 1, 2, 3)$$

Clustering in set ***partition*** notation:

$$\pi^{(2)} = \{\{1, 2, 3\}, \{4\}, \{5\}\}$$

Clustering as ***pairwise allocation matrix*** (i.e., adjacency matrix):

$$A(c^{(2)}) = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

## Example: Third Clustering in MCMC Output

Clustering in ***cluster label*** notation:

$$c^{(3)} = (1, 1, 2, 1, 2)$$

Clustering in set ***partition*** notation:

$$\pi^{(3)} = \{\{1, 2, 4\}, \{3, 5\}\}$$

Clustering as ***pairwise allocation matrix*** (i.e., adjacency matrix):

$$A(c^{(3)}) = \begin{bmatrix} 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 \end{bmatrix}$$



## Example: Averaging the MCMC Clustering Output

Averaging the vector of cluster labels **does not make sense**.

Averaging the set partitions is **not defined**.

## Example: Averaging the MCMC Clustering Output

Averaging the vector of cluster labels **does not make sense**.

Averaging the set partitions is **not defined**.

Averaging pairwise allocation matrices **does** make sense:

$$\bar{A} = \frac{1}{B} \sum_{b=1}^B A(c^{(b)}) = \begin{bmatrix} 1 & 2/3 & 2/3 & 1/3 & 0 \\ 2/3 & 1 & 1/3 & 2/3 & 1/3 \\ 2/3 & 1/3 & 1 & 0 & 1/3 \\ 1/3 & 2/3 & 0 & 1 & 1/3 \\ 0 & 1/3 & 1/3 & 1/3 & 1 \end{bmatrix}$$

$\bar{A}$  is the estimated **expected pairwise allocation matrix (EPAM)** because it's  $(i, j)$  element estimates  $\mu_{ij} = \Pr(c_i = c_j \mid \text{data})$ .

## More Realistic Example

Suppose MCMC output contains 1,000 posterior samples for the partition of 150 observations (encoded with cluster labels), e.g.:

```
iris.clusterings[1,]
```

```
##      [1] 1 1 1 1 2 2 1 2 1 1 1 1 1 1 2 2 1 1 1 2 1 2
##     [23] 2 1 1 1 1 1 1 1 1 1 2 2 1 1 1 2 1 1 2 3 1 1
##     [45] 1 1 2 1 2 1 4 4 4 4 4 4 4 3 4 4 3 4 3 4 4 4
##     [67] 5 3 3 3 4 4 3 4 4 4 6 4 4 3 7 3 3 4 4 4 4 3
##     [89] 4 4 3 4 3 3 3 4 4 4 3 4 4 4 3 4 3 4 4 4 3 4
##    [111] 4 3 3 3 3 4 4 4 4 3 4 4 4 4 4 4 4 4 3 4 4 4
##    [133] 3 4 3 4 4 4 4 4 4 4 4 4 4 3 3 3 4 4
```

What is the Bayes estimate of the partition from these 1,000 samples?

# Loss Functions and Bayes Estimators

- A Bayes estimator minimizes the posterior expected value of a loss function.
- The 0-1 loss function:

$$L(c, \hat{c}) = \mathbb{I}\{c \neq \hat{c}\}$$

yielding the maximum *a posteriori* (MAP) clustering:

$$\operatorname{argmax}_{\hat{c}} p(\hat{c} \mid \text{data})$$

.

- Equal loss for clusterings that differs by one label and a clustering that differs by many labels.
- Mode may not represent well the “center” of a distribution.

# Loss Functions and Bayes Estimators

- Dahl (2006) suggested a least-squares criterion:

$$\operatorname{argmin}_{\hat{c}} \sum_{i=1}^n \sum_{j=1}^n (A(\hat{c})_{ij} - \mu_{ij})^2$$

- Lau & Green (2007) studied the Binder (1978) loss function in a Bayesian nonparametric context:

$$L(c, \hat{c}) = \sum_{i < j} \mathbf{I}\{c_i = c_j\} \mathbf{I}\{\hat{c}_i \neq \hat{c}_j\} + \mathbf{I}\{c_i \neq c_j\} \mathbf{I}\{\hat{c}_i = \hat{c}_j\}$$

yielding the clustering:

$$\operatorname{argmin}_{\hat{c}} \sum_{i=1}^n \sum_{j=1}^n \mathbf{I}\{\hat{c}_i = \hat{c}_j\} (0.5 - \mu_{ij})$$

- Dahl & Newton (2007) noted that minimizing the posterior expected loss of Binder (1978) is equivalent to the least-squares criterion in Dahl (2006).

## Loss Functions and Bayes Estimators

- Wade & Ghahramani (2017) proposed using the variation of information (VI) of Meilă (2007) as a loss function, yielding the clustering:

$$\operatorname{argmin}_{\hat{c}} \sum_{i=1}^n \left( \log \left( \sum_{j=1}^n \mathbf{I}\{\hat{c}_j = \hat{c}_i\} \right) - 2\mathbb{E} \left( \log \left( \sum_{j=1}^n \mathbf{I}\{\hat{c}_j = \hat{c}_i, c_j = c_i\} \right) \middle| \text{data} \right) \right)$$

which is computationally expensive. Instead, they suggest the clustering that minimizes the **lower bound** of the posterior expected value of the variation of information loss (VI.lb):

$$\operatorname{argmin}_{\hat{c}} \sum_{i=1}^n \left( \log \left( \sum_{j=1}^n \mathbf{I}\{\hat{c}_j = \hat{c}_i\} \right) - 2 \log \left( \sum_{j=1}^n \mathbf{I}\{\hat{c}_j = \hat{c}_i\} \mu_{ij} \right) \right)$$

## Monte Carlo Estimate the Posterior Expected Loss

- For a given  $\hat{c}$ , both the Binder and the lower bound of the VI loss are based on the  $\mu_{ij}$ 's.
- The  $(i, j)$  elements of  $\bar{A}$  are Monte Carlo estimates of the  $\mu_{ij}$ 's, leading to a Monte Carlo estimate of the posterior expected loss.
- But having a means to estimate the posterior expected loss for a given  $\hat{c}$  does not itself give a search algorithm for its minimization.

## Methods for Optimization Given a Loss Function

- Exhaustive search. Infeasible for even moderate  $n$ , e.g.,  $B(15) = 1,382,958,545$ .
- Round the estimated expected pairwise allocation matrix (EPAM). May not lead to a clustering, e.g.:

$$\bar{A} = \begin{bmatrix} 1 & 2/3 & 2/3 & 1/3 & 0 \\ 2/3 & 1 & 1/3 & 2/3 & 1/3 \\ 2/3 & 1/3 & 1 & 0 & 1/3 \\ 1/3 & 2/3 & 0 & 1 & 1/3 \\ 0 & 1/3 & 1/3 & 1/3 & 1 \end{bmatrix} \Rightarrow \begin{bmatrix} 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

- Medvedovic and Sivaganesan (2002) selected a clustering using hierarchal clustering using  $1 - \bar{A}$  as the distance matrix.
- Dahl (2006) selected the clustering in the MCMC output that minimizes the criterion.



# Methods for Optimization Given a Loss Function

- Lau & Green (2007) proposed a heuristic item-swapping algorithm based on binary integer programming to minimize the posterior expected Binder loss.
- Wade & Ghahramani (2017) proposed a greedy search algorithm based on neighborhoods defined by the Hasse diagram, which can be used for Binder or VI.lb loss.
- We propose the **sequentially-allocated latent structure optimization (SALSO)** method to perform a *series of micro optimizations* to stochastically search for the minimizer of the posterior expected value of Binder or VI.lb loss.

# Sequentially-Allocated Latent Structure Optimization

---

# Sequentially-Allocated Latent Structure Optimization

- The SALSO method is applicable for many types of latent structure, including clusterings, feature allocations, & networks.
- The steps to SALSO are:
  1. Starting for an empty structure, build up a full structure by sequentially optimizing the allocation of items.
  2. Improve the full structure by a series of one-at-a-time optimizations.
  3. Do the above steps many times for randomly-selected permutations and choose the structure that minimizes the posterior expected loss.
- The random order in which items are allocated is not necessarily their order in the dataset; the permutation  $\sigma = (\sigma_1, \dots, \sigma_n)$  of  $\{1, \dots, n\}$  gives the sequence in which the  $n$  items are allocated.

## Illustration of SALSO Method

- To illustrate the SALSO method, consider clustering 5 items.
- For simplicity, suppose  $\sigma = (\sigma_1, \dots, \sigma_5) = (1, 2, 3, 4, 5)$ .
- Recall the steps to SALSO are:
  1. Build up a full structure from an empty structure
  2. Improve the full structure
  3. Do it for many random permutations

## Step 1: Build Up a Full Structure

Clustering: ~ ~ ~ ~ ~

## Step 1: Build Up a Full Structure

Clustering: ? ~ ~ ~ ~      Candidates for ? are: 1

## Step 1: Build Up a Full Structure

Clustering: 1 ~ ~ ~ ~

## Step 1: Build Up a Full Structure

Clustering: 1 ? ~ ~ ~      Candidates for ? are: 1, 2



## Step 1: Build Up a Full Structure

Clustering: 1 1 ~ ~ ~

## Step 1: Build Up a Full Structure

Clustering: 1 1 ? ~ ~      Candidates for ? are: 1, 2

## Step 1: Build Up a Full Structure

Clustering: 1 1 2 ~ ~

## Step 1: Build Up a Full Structure

Clustering: 1 1 2 ? ~      Candidates for ? are: 1, 2, 3

## Step 1: Build Up a Full Structure

Clustering: 1 1 2 3 ~

## Step 1: Build Up a Full Structure

Clustering: 1 1 2 3 ?      Candidates for ? are: 1, 2, 3, 4

## Step 1: Build Up a Full Structure

Clustering: 1 1 2 3 3

## Step 2: Improving the Full Structure

Clustering: 1 1 2 3 3



## Step 2: Improving the Full Structure

Clustering: ? 1 2 3 3      Candidates for ? are: 1, 2, 3, 4

## Step 2: Improving the Full Structure

Clustering: 2 1 2 3 3

## Step 2: Improving the Full Structure

Clustering: 2 ? 2 3 3      Candidates for ? are: 1, 2, 3

## Step 2: Improving the Full Structure

Clustering: 2 1 2 3 3

## Step 2: Improving the Full Structure

Clustering: 2 2 ? 3 3      Candidates for ? are: 1, 2, 3

## Step 2: Improving the Full Structure

Clustering: 2 2 2 3 3

## Step 2: Improving the Full Structure

Clustering: 2 2 2 ? 3      Candidates for ? are: 1, 2, 3

## Step 2: Improving the Full Structure

Clustering: 2 2 2 1 3



## Step 2: Improving the Full Structure

Clustering: 2 2 2 1 ?      Candidates for ? are: 1, 2, 3

## Step 2: Improving the Full Structure

Clustering: 2 2 2 1 3

## Step 2: Improving the Full Structure

Clustering: 2 2 2 1 3      Scan completed

## Step 2: Improving the Full Structure

Clustering: 2 2 2 1 3      Put in canonical form

## Step 2: Improving the Full Structure

Clustering: 1 1 1 2 3

## Step 2: Improving the Full Structure

Clustering: 1 1 1 2 3      Any change from start of scan?

## Step 2: Improving the Full Structure

Clustering: 1 1 1 2 3      Yes, so perform another scan

## Step 3: Do It For Many Permutations

- The permutation many lead to a local minimizer.
- Improve the chances of finding the global minimizer by repeating Step 1 and 2 for many random permutations.
  - *This is embarrassingly parallel.*
- Select the structure the minimizes the posterior expected loss among all those good structures obtained by using many random permutations.



# Review of the Steps of the SALSO Method

1. Build up a full structure from an empty structure
2. Improve the full structure
3. Do it for many random permutations

# Software & Constrained Optimization

---

# Software Implementation

SALSO is implemented in the R package “sdols” available on CRAN.

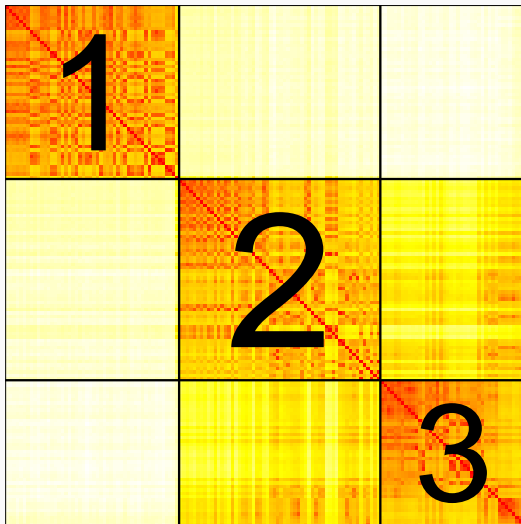
```
library(sdols)
dim(iris.clusterings)

## [1] 1000 150

epam <- expectedPairwiseAllocationMatrix(iris.clusterings)
estimate <- salso(epam, nCandidates=100,
                  budgetInSeconds=20, maxSize=3)
table(estimate)

## estimate
## 1 2 3
## 50 42 58
```

```
plot(confidence(estimate, epam))
```



# Constrained Optimization

- We may want to constrain the optimization.
  - For example, the sake of interpretation, it may be helpful to **limit** the number of clusters or features (yet we still want a small posterior expected loss).

# Constrained Optimization

- We may want to constrain the optimization.
  - For example, the sake of interpretation, it may be helpful to **limit** the number of clusters or features (yet we still want a small posterior expected loss).
- Solution: During micro optimization, never create a structure that violates the constraint.
  - For example, don't consider allocations that create clusters beyond the desired `maxSize`.

# Constrained Optimization

- We may want to constrain the optimization.
  - For example, the sake of interpretation, it may be helpful to **limit** the number of clusters or features (yet we still want a small posterior expected loss).
- Solution: During micro optimization, never create a structure that violates the constraint.
  - For example, don't consider allocations that create clusters beyond the desired `maxSize`.

Clustering: ? 1 2 3 3      Candidates for ? are: 1, 2, 3  
but not: 4

# Empirical Comparison

---



# Comparison Methodology

- Various optimization methods:
  - Hierarchical clustering of Medvedovic and Sivaganesan (2002) using average or complete linkage [mcclust]
  - Draws method of Dahl (2006) [sdols, mcclust]
  - Linear programming method of Lau & Green (2007) [mcclust]
  - Greedy search by Wade & Ghahramani (2017) [mcclust.ext]
  - SALSO method [sdols]
- Loss functions
  - Binder loss (Binder 1978)
  - Lower bound of the variation of information loss (Wade & Ghahramani 2017)
- Datasets: Three sets of MCMC output from a variety of models with 1,000 samples each.

Example 1: Ewens Pitman attraction distribution (Dahl, Day, Tsai 2017) applied to iris data.

Size: 150 observations

Method	Binder	Time
SALSO (12)	3493.16	0.039 seconds
SALSO (100)	3493.16	0.218 seconds
M & S (avg)	3497.01	0.055 seconds
W & G	3500.12	3.8 minutes
M & S (comp)	3512.90	0.055 seconds
L & G	3560.01	2.7 minutes
Draws	3607.48	0.085 seconds

Example 1: Ewens Pitman attraction distribution (Dahl, Day, Tsai 2017) applied to iris data.

Size: 150 observations

Method	VI.lb	Time
M & S (comp)	1.3246	0.026 seconds
M & S (avg)	1.3246	0.026 seconds
SALSO (12)	1.3246	0.033 seconds
SALSO (100)	1.3246	0.197 seconds
W & G	1.3246	7.9 minutes
Draws	1.3320	0.102 seconds
L & G	—	—

Example 2: Gaussian likelihood with a spatial PPM (Page & Quintana 2016) prior.

Size: 600 observations

Method	Binder	Time
SALSO (12)	7509.7	0.843 seconds
SALSO (100)	7509.7	7.033 seconds
L & G	7509.7	8.2 hours
M & W (comp)	8060.7	1.615 seconds
W & G	8365.6	18.8 hours
M & S (avg)	9409.1	1.604 seconds
Draws	10464.5	0.765 seconds

Example 2: Gaussian likelihood with a spatial PPM (Page & Quintana 2016) prior.

Size: 600 observations

Method	VI.lb	Time
M & S (avg)	2.7543	0.700 seconds
SALSO (100)	2.8241	3.94 seconds
M & W (comp)	2.8526	0.715 seconds
SALSO (12)	2.8633	26.61 seconds
Draws	3.7788	1.176 seconds
W & G	4.6411	36.3 hours
L & G	—	—

Example 3: Bivariate Gaussian likelihood with a spatial PPM (Page & Quintana 2016) prior.

Size: 600 observations

Method	Binder	Time
SALSO (12)	46270.74	0.937 seconds
SALSO (100)	46270.74	7.709 seconds
L & G	46271.21	9.8 hours
M & S (avg)	46724.64	1.609 seconds
M & S (comp)	47844.03	1.641 seconds
Draws	53182.66	1.005 seconds
W & G	57761.10	24.1 hours

Example 3: Bivariate Gaussian likelihood with a spatial PPM (Page & Quintana 2016) prior.

Size: 600 observations

Method	VI.lb	Time
SALSO (100)	1.5620	14.182 seconds
SALSO (12)	1.5649	1.552 seconds
M & S (comp)	1.5829	0.673 seconds
M & S (avg)	1.5858	0.733 seconds
Draws	1.9108	0.555 seconds
W & G	7.2197	41.6 hours
L & G	—	—

## **SALSO for Feature Allocation**

---



## SALSO for Feature Allocation

Consider a feature allocation:  $\{\{1, 2, 4, 5, 6\}, \{1, 3, 4, 7\}, \{2, 5\}\}$

... and its associated binary matrix:

Utah	1	0	1
Vermont	0	1	1
Virginia	1	0	0
Washington	1	0	1
West Virginia	0	1	1
Wisconsin	0	0	1
Wyoming	1	0	0

## SALSO for Feature Allocation

Consider a feature allocation:  $\{\{1, 2, 4, 5, 6\}, \{1, 3, 4, 7\}, \{2, 5\}\}$

... and its associated pairwise allocation matrix:

	[,1]	[,2]	[,3]	[,4]	[,5]	[,6]	[,7]
[1,]	2	1	1	2	1	1	1
[2,]	1	2	0	1	2	1	0
[3,]	1	0	1	1	0	0	1
[4,]	2	1	1	2	1	1	1
[5,]	1	2	0	1	2	1	0
[6,]	1	1	0	1	1	1	0
[7,]	1	0	1	1	0	0	1

## SALSO for Feature Allocation

The associated pairwise allocation matrices for the feature allocations in your MCMC output can be averaged:

	[,1]	[,2]	[,3]	[,4]	[,5]	[,6]	[,7]
[1,]	2.04	1.02	1.06	2.02	1.09	1.01	1.00
[2,]	1.02	2.09	0.90	1.08	2.02	1.01	0.96
[3,]	1.06	0.90	1.50	1.14	0.94	0.54	0.32
[4,]	2.02	1.08	1.14	2.21	1.16	1.01	1.01
[5,]	1.09	2.02	0.94	1.16	2.18	1.01	1.02
[6,]	1.01	1.01	0.54	1.01	1.01	1.03	0.54
[7,]	1.00	0.96	0.32	1.01	1.02	0.54	1.54

$$\operatorname{argmin}_{\hat{Z}} \sum_{i=1}^n \sum_{j=1}^n (A(\hat{Z}) - \bar{A})^2$$

## SALSO for Feature Allocation

```
epam <- expectedPairwiseAllocationMatrix(  
  USArrests.featureAllocations)  
est.salso <- salso(epam,structure="featureAllocation")  
est.dlso <- dlso(USArrests.featureAllocations)  
latentStructureFit(est.salso,epam)$squaredError  
  
## [1] 160.2849  
  
latentStructureFit(est.dlso,epam)$squaredError  
  
## [1] 338.8629
```

## Wrapping Up

---

# Way to Improve the Method?

Modified method:

1. Build up a full structure from an empty structure
  - Periodically reallocate items (a la Step 2) in the as-of-yet incomplete structure.
2. Improve the full structure
3. Do it for many random permutations

# Conclusion

- We presented the **sequentially-allocated latent structure optimization (SALSO)** method to minimize an objective criterion to obtain a *point estimate* based on a collection of randomly-sampled *latent features*.
- SALSO is a *stochastic search* method involving a *series of micro optimizations*.
- Status:
  - Well-developed for *clusterings*. Implemented in the “sdols” package on CRAN.
  - Initial version for *feature allocations* in the “sdols” package.
  - Want to apply to other structures, e.g., *networks*.
- Can we pick a representative observation?
- Summarizes other than point estimates?