bnpy Documentation

Release 0.1.01

Mike Hughes

2016 December 09

BNPy (or bnpy) is Bayesian Nonparametric clustering for Python.

Our goal is to make it easy for Python programmers to train state-of-the-art clustering models on large datasets. We focus on nonparametric models based on the Dirichlet process, especially extensions that handle hierarchical and sequential datasets. Traditional parametric counterparts (like finite mixture models) are also supported.

Training a model with **bnpy** requires the user to specify the dataset, the model, and the algorithm to use. Flexible keyword options allow advanced users lots of control, but smart defaults make it simple for beginners. **bnpy**‘s modular implementation makes it possible to try many variants of models and algorithms, to find the best fit for the data at hand.

CHAPTER 1

Example Gallery

You can find many examples of **bnpy** in action in our curated Example Gallery.

These same demos are also directly available as Python scrips inside the project Github repository.

CHAPTER 2

Quick Start

You can use **bnpy** to train a model in two ways: (1) from a command line/terminal, or (2) from within a Python script (of course). Both options require specifying a dataset, an allocation model, an observation model (likelihood), and an algorithm. Optional keyword arguments with reasonable defaults allow control of specific model hyperparameters, algorithm parameters, etc.

Below, we show how to call bnpy to train a 8 component Gaussian mixture model on a default toy dataset stored in a .csv file on disk. In both cases, log information is printed to stdout, and all learned model parameters are saved to disk.

2.1 Training from a terminal

```
python -m bnpy.Run /path/to/my_dataset.csv FiniteMixtureModel Gauss EM --K 8 --output_path
```

2.2 Training via Python

```
import bnpy
bnpy.run('/path/to/dataset.csv',
         'FiniteMixtureModel', 'Gauss', 'EM',
         K=8, output_path='/tmp/my_dataset/results/')
```

2.3 Featured algorithms

Train a Dirichlet-process Gaussian mixture model (DP-GMM) via full-dataset variational coordinate ascent. This algorithm is often called “VB” for variational Bayes.

```
python -m bnpy.Run /path/to/dataset.csv DPMixtureModel Gauss VB --K 8
```

Train DP-GMM via scalable incremental or “memoized” variational coordinate ascent, with birth and merge moves, with data divided into 10 batches.

```
python -m bnpy.Run /path/to/dataset.csv DPMixtureModel Gauss memoVB --K 8 --nBatch 10 --mo
```

Train HDP-HMM model to capture sequential structure in the dataset

2.4 Getting Help

```
# print help message for required arguments
python -m bnpy.Run --help
```

```
# print help message for specific keyword options for Gaussian mixture models
python -m bnpy.Run /path/to/dataset.csv FiniteMixtureModel Gauss EM --kwhelp
```

CHAPTER 3

Supported allocation models

The following are possible *allocation* models, which is **bnpy**-terminology for a generative model which assigns clusters to structured datasets.

- **Mixture models**
  - *FiniteMixtureModel* : fixed number of clusters
  - *DPMixtureModel* : infinite number of clusters, via the Dirichlet process
- **Topic models (aka admixtures models)**
  - *FiniteTopicModel* : fixed number of topics. This is Latent Dirichlet allocation.
  - *HDPTopicModel* : infinite number of topics, via the hierarchical Dirichlet process
- **Hidden Markov models (HMMs)**
  - *FiniteHMM* : Markov sequence model with a fixture number of states
  - *HDPHMM* : Markov sequence models with an infinite number of states
- **COMING SOON**
  - relational models (like the IRM, MMSB, etc.)
  - grammar models

CHAPTER 4

Supported observations models

Any of the above allocation models can be combined with one of these *observation* models, which describe how to produce data assigned to a specific cluster.

- **Real-valued vector observations (1-dim, 2-dim, ... D-dim)**
  - *Gauss* : Full-covariance Gaussian
  - *DiagGauss* : Diagonal-covariance Gaussian
  - *ZeroMeanGauss* : Zero-mean, full-covariance
  - *AutoRegGauss* : first-order auto-regressive Gaussian
- **Binary vector observations (1-dim, 2-dim, ... D-dim)**
  - *Bern* : Bernoulli
- **Discrete, bag-of-words data (each observation is one of V symbols)**
  - *Mult* : Multinomial

CHAPTER 5

Supported algorithms

- **Variational methods**
  - *EM* : Expectation-maximization
  - *VB* : variational Bayes
  - *soVB* : stochastic variational (online)
  - *moVB* : memoized variational (online)
- **COMING SOON**
  - Gibbs sampling

5.1 Installation

5.1.1 Requirements

bnpy requires Python 2.7+ and the following packages:

- numpy >= 1.11
- scipy >= 0.18
- pandas >= 0.18
- Cython >= 0.25
- joblib >= 0.10
- memory_profiler >= 0.41
- munkres >= 1.0
- numexpr >= 2.6
- psutil >= 5.0
- scikit_learn >= 0.18

For interactivity and visualization, we also recommend:

- ipython >= 5.1
- matplotlib >= 1.5

5.1.2 Easy installation of bnpy

First, make sure you have a working local install of the Anaconda python distribution, which makes managing common Python packages within userspace a breeze.

Then, you can just clone the latest stable version of bnpy via:

```
git clone https://github.com/bnpy/bnpy.git
```

And then install from the cloned source via:

```
cd bnpy/
pip install -e .
```

Verifying correct installation

Within a terminal, you can first verify basic installation with:

```
python -m bnpy.Run --help
```

You can further train a very simple model:

```
python -m bnpy.Run \
        DATASET_PATH/faithful/faithful.csv \
        FiniteMixtureModel Gauss VB --nLap 1 --K 3
```

To further verify matplotlib installation, enter:

```
from matplotlib import pylab
pylab.plot([1,2,3])
pylab.show()
```

5.1.3 Advanced Installation

Some of bnpy's advanced features require compiling custom C++ source code for fast algorithms. These aren't needed for basic usage, but do come in handy.

Installing with Eigen C++ libraries

The Eigen C++ Matrix template library (>=3.0) is used for:

- fast local step updates for hidden Markov models
- fast local step updates for L-sparse mixtures

If you want these features, go download and install Eigen from http://www.eigen.tuxfamily.org.

To install bnpy with Eigen support, you need to set the following environment variable:

```
export EIGENPATH=/path/to/eigen/
```

You can verify the right location by verifying the following directory exists:

```
ls $EIGENPATH/Eigen/
```

If the $EIGENPATH env variable is set when you perform **pip install**, the required C++ libraries should be built and useful automatically.

Installing with Boost C++ math libraries

The Boost C++ math library (>= 1.52) is used for the following features:

- fast local step updates for L-sparse topic models

If you want these features, go download and install boost from http://www.boost.org.

To install bnpy with Boost C++ support, you need to set the following environment variable:

```
export BOOSTMATHPATH=/path/to/boost/include/
```

You can verify the right location by verifying the following directory exists:

```
ls $BOOSTMATHPATH/math/
```

If the $BOOSTMATHPATH env variable is set when you perform **pip install**, the required C++ libraries should be built and useful automatically.

5.1.4 Common errors with matplotlib

If you try the above and get errors about not having “wx” or “wxpython” or “qt” installed, you need to configure your Matplotlib_backend.

I recommend setting your matplotlibrc file to have *backend: TkAgg* for Linux, and *backend: MacOSX* for Mac.

5.2 Example Gallery

5.2.1 Asterisk toy dataset

Training mixture models and DP mixture models on well-separated 2D Gaussian blobs.

Variational coordinate descent for Mixture of Gaussians

How to do Variational Bayes (VB) coordinate descent for GMM.

Here, we train a finite mixture of Gaussians with full covariances.

We'll consider a mixture model with a symmetric Dirichlet prior:

$$\pi \sim \text{Dir}(1/K, 1/K, \dots 1/K)$$

as well as a standard conjugate prior on the mean and covariances, such that

$$\mathbb{E}[\mu_k] = 0$$
$$\mathbb{E}[\Sigma_k] = 0.1 I_D$$

We will initialize the approximate variational posterior using K=10 randomly chosen examples ('randexamples' procedure), and then perform coordinate descent updates (alternating local step and global step) until convergence.

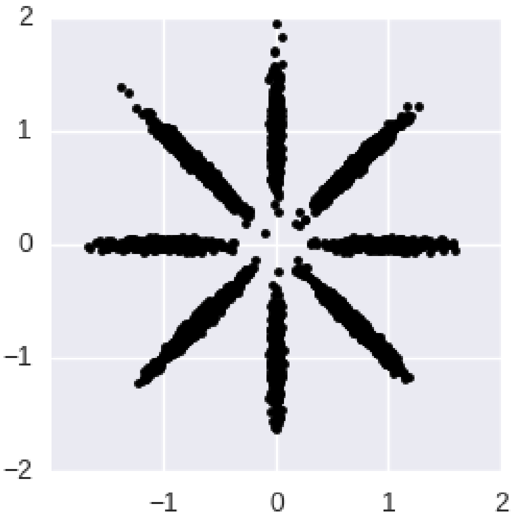
```
import bnpy
import numpy as np
import os

from matplotlib import pylab
import seaborn as sns
# sphinx_gallery_thumbnail_number = 3

FIG_SIZE = (3, 3)
pylab.rcParams['figure.figsize'] = FIG_SIZE
```

Read bnpy's built-in "AsteriskK8" dataset from file.

```
dataset_path = os.path.join(bnpy.DATASET_PATH, 'AsteriskK8')
dataset = bnpy.data.XData.read_npz(
    os.path.join(dataset_path, 'x_dataset.npz'))
```

Make a simple plot of the raw data

```
pylab.plot(dataset.X[:, 0], dataset.X[:, 1], 'k.')
pylab.gca().set_xlim([-2, 2])
pylab.gca().set_ylim([-2, 2])
pylab.tight_layout()
```



Training the model

Let’s do one single run of the VB algorithm.

Using 10 clusters and the ‘randexamples’ initializatio procedure.

```
trained_model, info_dict = bnpy.run(
    dataset, 'FiniteMixtureModel', 'Gauss', 'VB',
    output_path='/tmp/AsteriskK8/helloworld-K=10/',
    nLap=100,
    sF=0.1, ECovMat='eye',
    K=10,
    initname='randexamples')
```

Out:

```
Dataset Summary:
X Data
  num examples: 5000
  num dims: 2
Allocation Model:  Finite mixture model. Dir prior param 1.00
Obs. Data  Model:  Gaussian with full covariance.
Obs. Data  Prior:  Gauss-Wishart on mean and covar of each cluster
  E[  mean[k] ] =
   [ 0.  0.]
  E[ covar[k] ] =
  [[ 0.1  0. ]
   [ 0.   0.1]]
Initialization:
  initname = randexamples
  K = 10 (number of clusters)
  seed = 1607680
  elapsed_time: 0.0 sec
Learn Alg: VB | task  1/1 | alg. seed: 1607680 | data order seed: 8541952
```

```
task_output_path: /tmp/AsteriskK8/helloworld-K=10/1
        1/100 after      0 sec. |    132.9 MiB | K   10 | loss  6.582634775e-01 |
        2/100 after      0 sec. |    132.9 MiB | K   10 | loss  4.350235353e-01 | Ndiff   6
        3/100 after      0 sec. |    132.9 MiB | K   10 | loss  3.454096950e-01 | Ndiff  19
        4/100 after      0 sec. |    132.9 MiB | K   10 | loss  3.049230819e-01 | Ndiff  17
        5/100 after      0 sec. |    132.9 MiB | K   10 | loss  2.732439109e-01 | Ndiff  10
        6/100 after      0 sec. |    132.9 MiB | K   10 | loss  2.326372999e-01 | Ndiff   4
        7/100 after      0 sec. |    132.9 MiB | K   10 | loss  2.100254570e-01 | Ndiff
        8/100 after      0 sec. |    132.9 MiB | K   10 | loss  2.097453779e-01 | Ndiff   1
        9/100 after      0 sec. |    132.9 MiB | K   10 | loss  2.094741199e-01 | Ndiff   1
       10/100 after      0 sec. |    132.9 MiB | K   10 | loss  2.091535256e-01 | Ndiff   1
       11/100 after      0 sec. |    132.9 MiB | K   10 | loss  2.087740779e-01 | Ndiff   1
       12/100 after      0 sec. |    132.9 MiB | K   10 | loss  2.083262051e-01 | Ndiff   1
       13/100 after      0 sec. |    132.9 MiB | K   10 | loss  2.078022512e-01 | Ndiff   1
       14/100 after      0 sec. |    132.9 MiB | K   10 | loss  2.072001819e-01 | Ndiff   1
       15/100 after      0 sec. |    132.9 MiB | K   10 | loss  2.065288318e-01 | Ndiff   1
       16/100 after      0 sec. |    132.9 MiB | K   10 | loss  2.058118708e-01 | Ndiff   1
       17/100 after      0 sec. |    132.9 MiB | K   10 | loss  2.050844132e-01 | Ndiff   1
       18/100 after      0 sec. |    132.9 MiB | K   10 | loss  2.043795824e-01 | Ndiff   1
       19/100 after      0 sec. |    132.9 MiB | K   10 | loss  2.037146182e-01 | Ndiff   1
       20/100 after      0 sec. |    132.9 MiB | K   10 | loss  2.030678434e-01 | Ndiff   1
       21/100 after      0 sec. |    132.9 MiB | K   10 | loss  2.022621299e-01 | Ndiff   1
       22/100 after      0 sec. |    132.9 MiB | K   10 | loss  2.008648784e-01 | Ndiff   1
       23/100 after      0 sec. |    132.9 MiB | K   10 | loss  2.002168334e-01 | Ndiff
       24/100 after      0 sec. |    132.9 MiB | K   10 | loss  1.995420809e-01 | Ndiff
       25/100 after      1 sec. |    132.9 MiB | K   10 | loss  1.988594991e-01 | Ndiff
       26/100 after      1 sec. |    132.9 MiB | K   10 | loss  1.982035545e-01 | Ndiff
       27/100 after      1 sec. |    132.9 MiB | K   10 | loss  1.974254707e-01 | Ndiff   1
       28/100 after      1 sec. |    132.9 MiB | K   10 | loss  1.966834121e-01 | Ndiff   1
       29/100 after      1 sec. |    132.9 MiB | K   10 | loss  1.964267444e-01 | Ndiff   1
       30/100 after      1 sec. |    132.9 MiB | K   10 | loss  1.961250819e-01 | Ndiff   1
       31/100 after      1 sec. |    132.9 MiB | K   10 | loss  1.957730227e-01 | Ndiff   1
       32/100 after      1 sec. |    132.9 MiB | K   10 | loss  1.953666781e-01 | Ndiff   1
       33/100 after      1 sec. |    132.9 MiB | K   10 | loss  1.949058031e-01 | Ndiff   1
       34/100 after      1 sec. |    132.9 MiB | K   10 | loss  1.943977009e-01 | Ndiff
       35/100 after      1 sec. |    132.9 MiB | K   10 | loss  1.938623040e-01 | Ndiff
       36/100 after      1 sec. |    132.9 MiB | K   10 | loss  1.933345511e-01 | Ndiff
       37/100 after      1 sec. |    132.9 MiB | K   10 | loss  1.928570776e-01 | Ndiff
       38/100 after      1 sec. |    132.9 MiB | K   10 | loss  1.924667495e-01 | Ndiff
       39/100 after      1 sec. |    132.9 MiB | K   10 | loss  1.921899371e-01 | Ndiff
       40/100 after      1 sec. |    132.9 MiB | K   10 | loss  1.919922352e-01 | Ndiff
       41/100 after      1 sec. |    132.9 MiB | K   10 | loss  1.916267958e-01 | Ndiff
       42/100 after      1 sec. |    132.9 MiB | K   10 | loss  1.909787395e-01 | Ndiff
... done. converged.
```

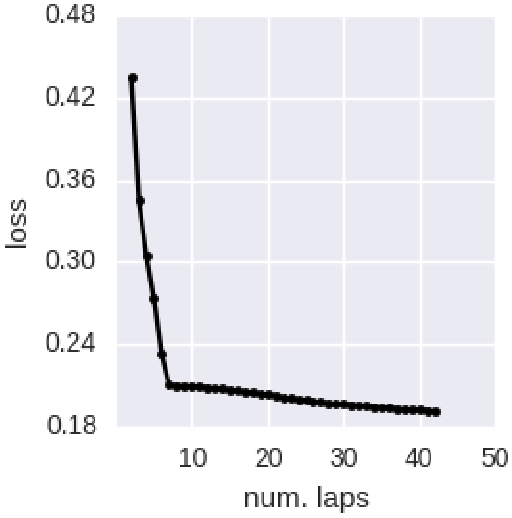
Loss function trace plot

We can plot the value of the loss function over iterations, starting after the first full pass over the dataset (first lap).

As expected, we see monotonic decrease in the loss function’s score after every subsequent iteration.

Remember that the VB algorithm for GMMs is *guaranteed* to decrease this loss function after every step.

```
pylab.plot(info_dict['lap_history'][1:], info_dict['loss_history'][1:], 'k.-')
pylab.xlabel('num. laps')
pylab.ylabel('loss')
pylab.tight_layout()
```



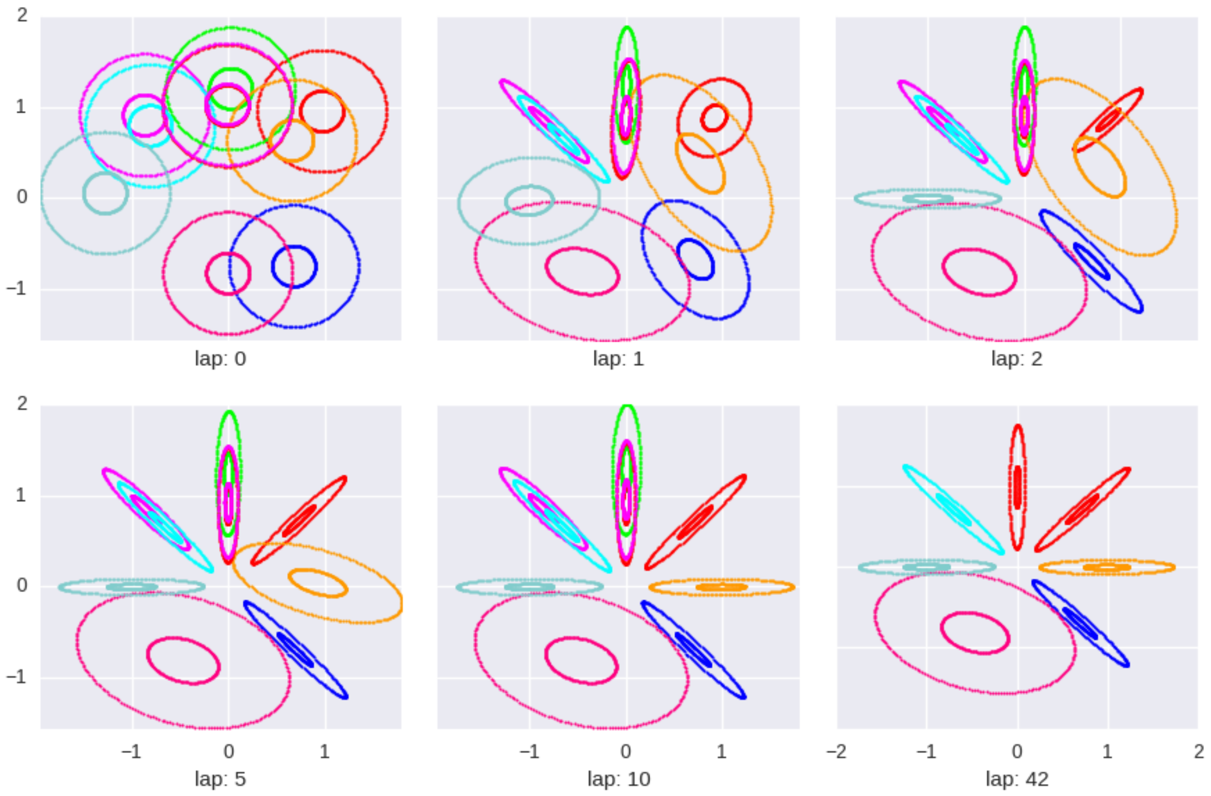
Visualization of learned clusters

Here's a short function to show the learned clusters over time.

```
def show_clusters_over_time(
        task_output_path=None,
        query_laps=[0, 1, 2, 5, 10, None],
        nrows=2):
    ''' Read model snapshots from provided folder and make visualizations

    Post Condition
    --------------
    New matplotlib plot with some nice pictures.
    '''
    ncols = int(np.ceil(len(query_laps) // float(nrows)))
    fig_handle, ax_handle_list = pylab.subplots(
        figsize=(FIG_SIZE[0] * ncols, FIG_SIZE[1] * nrows),
        nrows=nrows, ncols=ncols, sharex=True, sharey=True)
    for plot_id, lap_val in enumerate(query_laps):
        cur_model, lap_val = bnpy.load_model_at_lap(task_output_path, lap_val)
        # Plot the current model
        cur_ax_handle = ax_handle_list.flatten()[plot_id]
        bnpy.viz.PlotComps.plotCompsFromHModel(
            cur_model, Data=dataset, ax_handle=cur_ax_handle)
        cur_ax_handle.set_xticks([-2, -1, 0, 1, 2])
        cur_ax_handle.set_yticks([-2, -1, 0, 1, 2])
        cur_ax_handle.set_xlabel("lap: %d" % lap_val)
    pylab.tight_layout()
```

Show the estimated clusters over time

```
show_clusters_over_time(info_dict['task_output_path'])
```



Out:

```
SKIPPED 3 comps with size below 0.00
```

Total running time of the script: (0 minutes 1.969 seconds)

Download Python source code: plot-02-demo=vb_single_run-model=dp_mix+gauss.py

Download Jupyter notebook: plot-02-demo=vb_single_run-model=dp_mix+gauss.ipynb

Generated by Sphinx-Gallery

Initialization for Mixtures of Gaussians

How to initialize Gaussian observation models.

We demonstrate a few possible initialization procedures for Gaussian observation models (which include Gauss, DiagGauss, ZeroMeanGauss).

Initialization depends on two key user-specified procedures:

1. Specifying hyperparameters for the conjugate prior
2. Specifying how many clusters are created

```
import bnpy
import numpy as np
import os

from matplotlib import pylab
import seaborn as sns
# sphinx_gallery_thumbnail_number = 2

FIG_SIZE = (3, 3)
SMALL_FIG_SIZE = (2, 2)
```

Out:

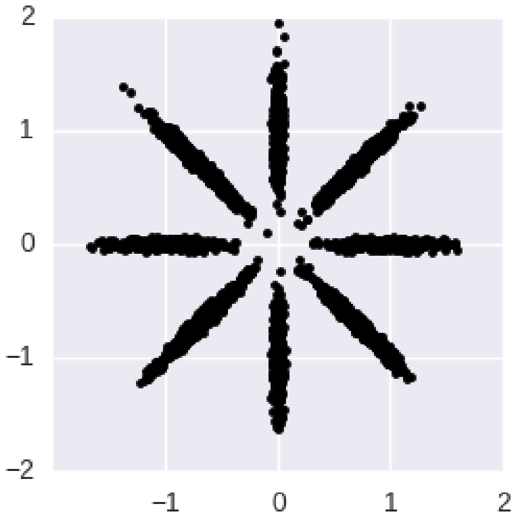
```
/home/docs/checkouts/readthedocs.org/user_builds/bnpy/envs/latest/local/lib/python2.7/site-
```

Read bnpy's built-in "AsteriskK8" dataset from file.

```
dataset_path = os.path.join(bnpy.DATASET_PATH, 'AsteriskK8')
dataset = bnpy.data.XData.read_npz(
    os.path.join(dataset_path, 'x_dataset.npz'))
```

Make a simple plot of the raw data

```
pylab.figure(figsize=FIG_SIZE)
pylab.plot(dataset.X[:, 0], dataset.X[:, 1], 'k.')
pylab.gca().set_xlim([-2, 2])
pylab.gca().set_ylim([-2, 2])
pylab.tight_layout()
```



Utility function for displaying many random initializations side by side.

```
def show_many_random_initial_models(
        obsPriorArgsDict,
        initArgsDict,
        nrows=1, ncols=6):
    ''' Create plot of many different random initializations
    '''
    fig_handle, ax_handle_list = pylab.subplots(
        figsize=(SMALL_FIG_SIZE[0] * ncols, SMALL_FIG_SIZE[1] * nrows),
        nrows=nrows, ncols=ncols, sharex=True, sharey=True)
    for trial_id in range(nrows * ncols):
        cur_model = bnpy.make_initialized_model(
            dataset,
            allocModelName='FiniteMixtureModel',
            obsModelName='Gauss',
            algName='VB',
            allocPriorArgsDict=dict(gamma=10.0),
            obsPriorArgsDict=obsPriorArgsDict,
            initArgsDict=initArgsDict,
            seed=int(trial_id),
            )
        # Plot the current model
        cur_ax_handle = ax_handle_list.flatten()[trial_id]
        bnpy.viz.PlotComps.plotCompsFromHModel(
            cur_model, Data=dataset, ax_handle=cur_ax_handle)
        cur_ax_handle.set_xticks([-2, -1, 0, 1, 2])
        cur_ax_handle.set_yticks([-2, -1, 0, 1, 2])
    pylab.tight_layout()
```

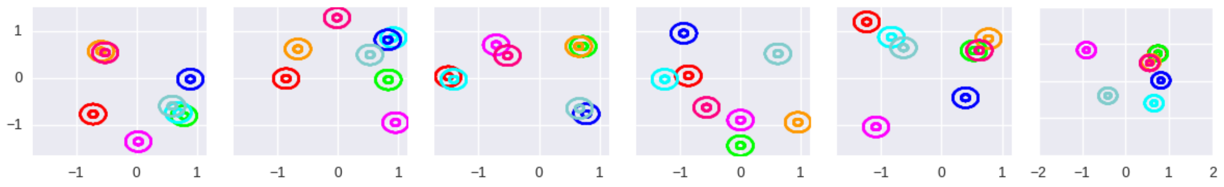
initname: 'randexamples'

This procedure selects K examples uniformly at random. Each cluster is then initialized from one selected example, using a standard global step update.

Example 1: Initialize with 8 clusters, with prior biased towards small covariances

$$\mathbb{E}_{\text{prior}}[\Sigma_k] = 0.01 I_D$$

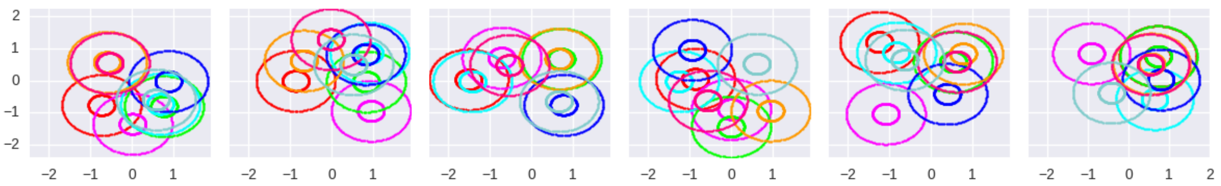
```
show_many_random_initial_models(
    dict(sF=0.01, ECovMat='eye'),
    dict(initname='randexamples', K=8))
```



Example 2: Initialize with 8 clusters, with prior biased towards moderate covariances

$$\mathbb{E}_{\text{prior}}[\Sigma_k] = 0.2 I_D$$

```
show_many_random_initial_models(
    dict(sF=0.2, ECovMat='eye'),
    dict(initname='randexamples', K=8))
```



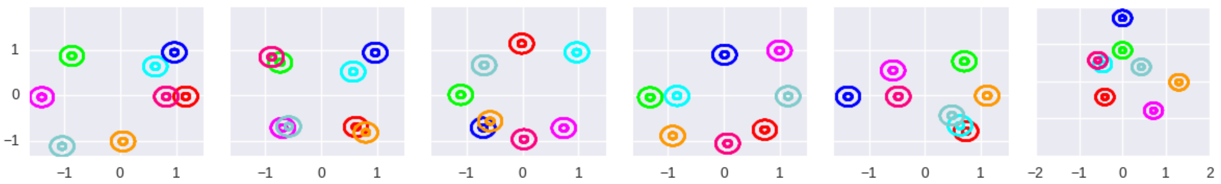
initname: 'bregmankmeans'

This procedure selects K examples using a distance-biased procedure. First, one example is chosen uniformly at random. Next, each successive example is chosen with probability proportional to the distance from the nearest example in the chosen set.

We measure distance using the appropriate Bregman divergence.

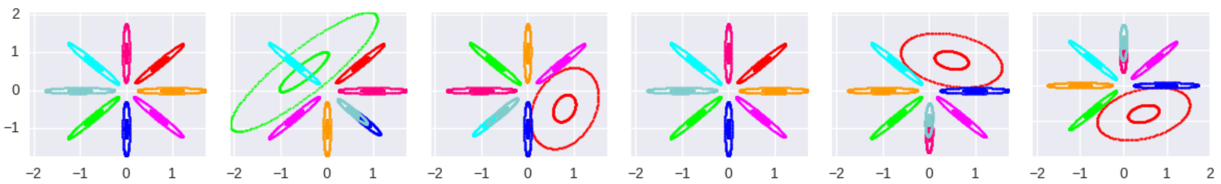
Example 1: Initialize with 8 clusters, with prior biased towards small covariances

```
show_many_random_initial_models(
    dict(sF=0.01, ECovMat='eye'),
    dict(initname='bregmankmeans', K=8, init_NiterForBregmanKMeans=0))
```



Example 2: Initialize as above, then allow the k-means algorithm to run for 10 iterations to "refine" the initial clustering.

```
show_many_random_initial_models(
    dict(sF=0.01, ECovMat='eye'),
    dict(initname='bregmankmeans', K=8, init_NiterForBregmanKMeans=10))
```



Total running time of the script: (0 minutes 4.934 seconds)

Download Python source code: `plot-01-demo=init_methods-model=mix+gauss.py`

Download Jupyter notebook: `plot-01-demo=init_methods-model=mix+gauss.ipynb`

Generated by Sphinx-Gallery

Variational with birth and merge proposals for DP mixtures of Gaussians

How to train a DP mixture model.

We'll show that despite diverse, poor quality initializations, our proposal moves that insert new clusters (birth) and remove redundant clusters (merge) can consistently recover the same ideal posterior with 8 clusters.

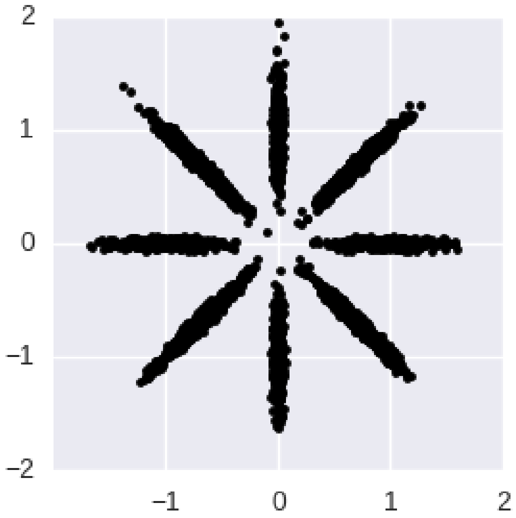
```
import bnpy
import numpy as np
import os

from matplotlib import pylab
import seaborn as sns
# sphinx_gallery_thumbnail_number = 2

FIG_SIZE = (3, 3)
pylab.rcParams['figure.figsize'] = FIG_SIZE
```

Read dataset from file.

```
dataset_path = os.path.join(bnpy.DATASET_PATH, 'AsteriskK8')
dataset = bnpy.data.XData.read_npz(
    os.path.join(dataset_path, 'x_dataset.npz'))
```

Make a simple plot of the raw data

```
pylab.plot(dataset.X[:, 0], dataset.X[:, 1], 'k.')
pylab.gca().set_xlim([-2, 2])
pylab.gca().set_ylim([-2, 2])
pylab.tight_layout()
```



Setup: Function for visualization

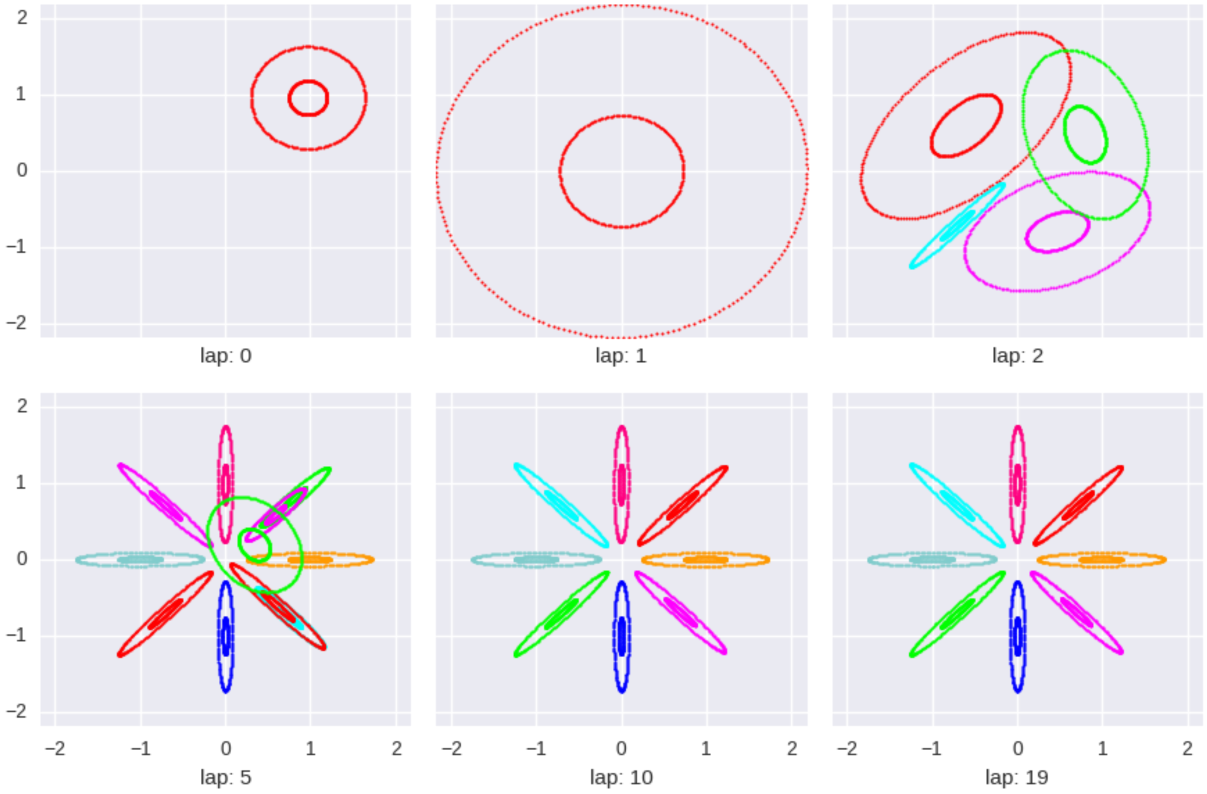
Here's a short function to show the learned clusters over time.

```
def show_clusters_over_time(
        task_output_path=None,
        query_laps=[0, 1, 2, 5, 10, None],
        nrows=2):
    ''' Read model snapshots from provided folder and make visualizations

    Post Condition
    --------------
    New matplotlib plot with some nice pictures.
    '''
    ncols = int(np.ceil(len(query_laps) // float(nrows)))
    fig_handle, ax_handle_list = pylab.subplots(
        figsize=(FIG_SIZE[0] * ncols, FIG_SIZE[1] * nrows),
        nrows=nrows, ncols=ncols, sharex=True, sharey=True)
    for plot_id, lap_val in enumerate(query_laps):
        cur_model, lap_val = bnpy.load_model_at_lap(task_output_path, lap_val)
        # Plot the current model
        cur_ax_handle = ax_handle_list.flatten()[plot_id]
        bnpy.viz.PlotComps.plotCompsFromHModel(
            cur_model, Data=dataset, ax_handle=cur_ax_handle)
        cur_ax_handle.set_xticks([-2, -1, 0, 1, 2])
        cur_ax_handle.set_yticks([-2, -1, 0, 1, 2])
        cur_ax_handle.set_xlabel("lap: %d" % lap_val)
    pylab.tight_layout()
```

Training from K=1 cluster

Using 1 initial cluster, with birth and merge proposal moves.

```
K1_trained_model, K1_info_dict = bnpy.run(
    dataset, 'DPMixtureModel', 'Gauss', 'memoVB',
    output_path='/tmp/AsteriskK8/trymoves-K=1/',
    nLap=100, nTask=1, nBatch=1,
    sF=0.1, ECovMat='eye',
    K=1, initname='randexamples',
    moves='birth,merge,shuffle',
    m_startLap=5, b_startLap=2, b_Kfresh=4)

show_clusters_over_time(K1_info_dict['task_output_path'])
```



Out:

```
Dataset Summary:
X Data
  total size: 5000 units
  batch size: 5000 units
  num. batches: 1
Allocation Model:  DP mixture with K=0. Concentration gamma0= 1.00
Obs. Data  Model:  Gaussian with full covariance.
Obs. Data  Prior:  Gauss-Wishart on mean and covar of each cluster
  E[  mean[k] ] =
   [ 0.  0.]
  E[ covar[k] ] =
  [[ 0.1  0. ]
   [ 0.   0.1]]
Initialization:
  initname = randexamples
  K = 1 (number of clusters)
  seed = 1607680
  elapsed_time: 0.0 sec
Learn Alg: memoVB | task  1/1 | alg. seed: 1607680 | data order seed: 8541952
task_output_path: /tmp/AsteriskK8/trymoves-K=1/1
BIRTH @ lap 1.00: Disabled. Waiting for lap >= 2 (--b_startLap).
MERGE @ lap 1.00: Disabled. Cannot plan merge on first lap. Need valid SS that represent wh
    1.000/100 after      0 sec. |    131.5 MiB | K    1 | loss  1.105578508e+00 |
MERGE @ lap 2.00: Disabled. Waiting for lap >= 5 (--m_startLap).
BIRTH @ lap 2.00 : Added 4 states. 1/1 succeeded. 0/1 failed eval phase. 0/1 failed build p
```

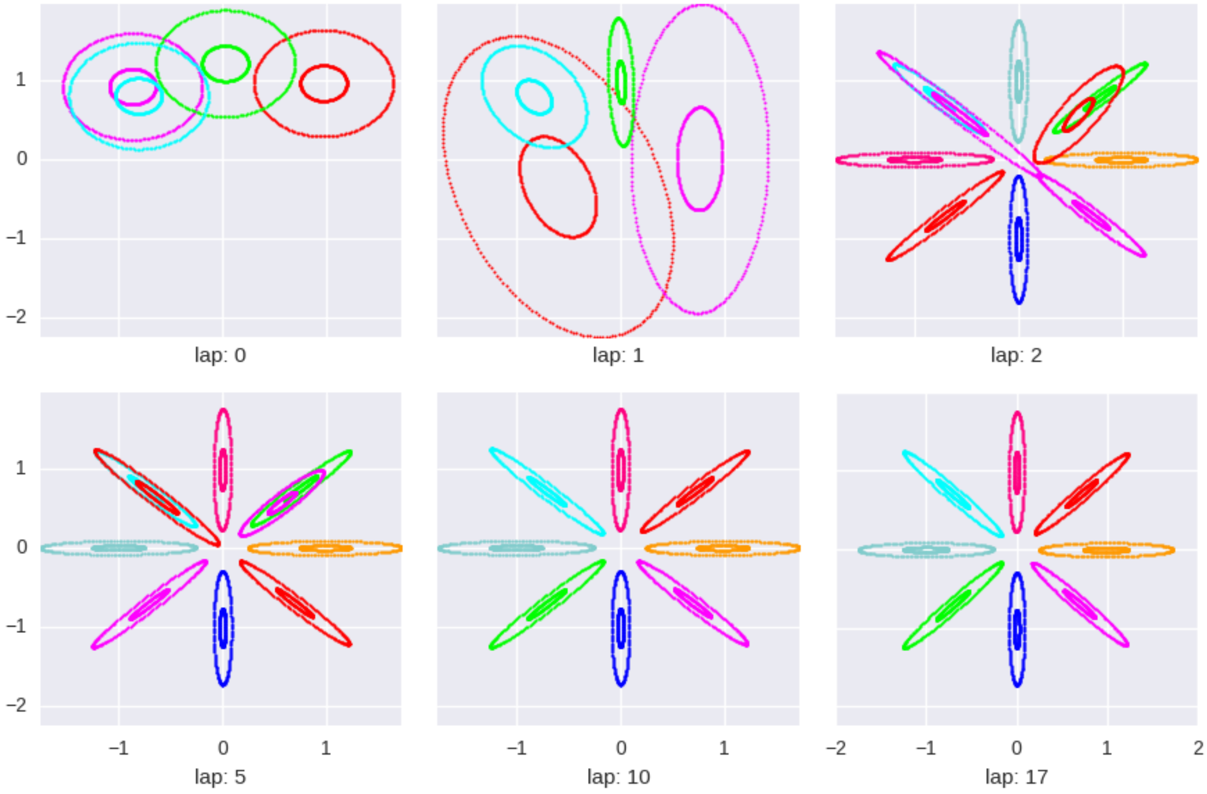
```
    2.000/100 after      0 sec. |    131.6 MiB | K    5 | loss  7.390275336e-01 |
MERGE @ lap 3.00: Disabled. Waiting for lap >= 5 (--m_startLap).
BIRTH @ lap 3.00 : Added 12 states. 3/4 succeeded. 1/4 failed eval phase. 0/4 failed build
    3.000/100 after      1 sec. |    131.6 MiB | K   17 | loss  2.819339824e-02 |
MERGE @ lap 4.00: Disabled. Waiting for lap >= 5 (--m_startLap).
BIRTH @ lap 4.00 : Added 0 states. 0/12 succeeded. 10/12 failed eval phase. 2/12 failed bui
    4.000/100 after      2 sec. |    131.6 MiB | K   17 | loss -1.240759022e-02 | Ndiff   8
BIRTH @ lap 5.000 : None attempted. 0 past failures. 0 too small. 17 too busy.
MERGE @ lap 5.00 : 4/30 accepted. Ndiff 98.67. 10 skipped.
    5.000/100 after      2 sec. |    131.6 MiB | K   13 | loss -2.742336056e-02 | Ndiff   8
BIRTH @ lap 6.000 : None attempted. 0 past failures. 0 too small. 13 too busy.
MERGE @ lap 6.00 : 3/23 accepted. Ndiff 767.62. 5 skipped.
    6.000/100 after      3 sec. |    131.6 MiB | K   10 | loss -4.626686932e-02 | Ndiff   8
BIRTH @ lap 7.000 : None attempted. 0 past failures. 0 too small. 10 too busy.
MERGE @ lap 7.00 : 0/18 accepted. Ndiff 0.00. 0 skipped.
    7.000/100 after      3 sec. |    131.6 MiB | K   10 | loss -4.686532682e-02 | Ndiff
BIRTH @ lap 8.00 : Added 0 states. 0/2 succeeded. 2/2 failed eval phase. 0/2 failed build p
MERGE @ lap 8.00 : 2/2 accepted. Ndiff 0.09. 5 skipped.
    8.000/100 after      3 sec. |    131.6 MiB | K    8 | loss -4.769229049e-02 | Ndiff
BIRTH @ lap 9.00 : Added 0 states. 0/1 succeeded. 1/1 failed eval phase. 0/1 failed build p
MERGE @ lap 9.00 : 0/9 accepted. Ndiff 0.00. 0 skipped.
    9.000/100 after      3 sec. |    131.6 MiB | K    8 | loss -4.787962424e-02 | Ndiff
BIRTH @ lap 10.000 : None attempted. 0 past failures. 0 too small. 8 too busy.
MERGE @ lap 10.00 : 0/10 accepted. Ndiff 0.00. 0 skipped.
   10.000/100 after      4 sec. |    131.6 MiB | K    8 | loss -4.787962433e-02 | Ndiff
BIRTH @ lap 11.00 : Added 0 states. 0/1 succeeded. 1/1 failed eval phase. 0/1 failed build
MERGE @ lap 11.00 : 0/5 accepted. Ndiff 0.00. 0 skipped.
   11.000/100 after      4 sec. |    131.6 MiB | K    8 | loss -4.787962433e-02 | Ndiff
BIRTH @ lap 12.00 : Added 0 states. 0/1 succeeded. 1/1 failed eval phase. 0/1 failed build
MERGE @ lap 12.00 : 0/4 accepted. Ndiff 0.00. 0 skipped.
   12.000/100 after      4 sec. |    131.6 MiB | K    8 | loss -4.787962433e-02 | Ndiff
MERGE @ lap 13.00: No promising candidates, so no attempts.
BIRTH @ lap 13.000 : None attempted. 8 past failures. 0 too small. 0 too busy.
   13.000/100 after      4 sec. |    131.6 MiB | K    8 | loss -4.787962433e-02 | Ndiff
BIRTH @ lap 14.000 : None attempted. 2 past failures. 0 too small. 6 too busy.
MERGE @ lap 14.00 : 0/9 accepted. Ndiff 0.00. 0 skipped.
   14.000/100 after      4 sec. |    131.6 MiB | K    8 | loss -4.787962433e-02 | Ndiff
BIRTH @ lap 15.000 : None attempted. 0 past failures. 0 too small. 8 too busy.
MERGE @ lap 15.00 : 0/10 accepted. Ndiff 0.00. 0 skipped.
   15.000/100 after      4 sec. |    131.6 MiB | K    8 | loss -4.787962433e-02 | Ndiff
BIRTH @ lap 16.000 : None attempted. 1 past failures. 0 too small. 7 too busy.
MERGE @ lap 16.00 : 0/5 accepted. Ndiff 0.00. 0 skipped.
   16.000/100 after      5 sec. |    131.6 MiB | K    8 | loss -4.787962433e-02 | Ndiff
BIRTH @ lap 17.000 : None attempted. 3 past failures. 0 too small. 5 too busy.
MERGE @ lap 17.00 : 0/4 accepted. Ndiff 0.00. 0 skipped.
   17.000/100 after      5 sec. |    131.6 MiB | K    8 | loss -4.787962433e-02 | Ndiff
MERGE @ lap 18.00: No promising candidates, so no attempts.
BIRTH @ lap 18.000 : None attempted. 8 past failures. 0 too small. 0 too busy.
   18.000/100 after      5 sec. |    131.6 MiB | K    8 | loss -4.787962433e-02 | Ndiff
BIRTH @ lap 19.000 : None attempted. 2 past failures. 0 too small. 6 too busy.
MERGE @ lap 19.00 : 0/9 accepted. Ndiff 0.00. 0 skipped.
   19.000/100 after      5 sec. |    131.6 MiB | K    8 | loss -4.787962433e-02 | Ndiff
... done. converged.
```

```
SKIPPED 1 comps with size below 0.00
SKIPPED 2 comps with size below 0.00
```

Training from K=4 cluster

Now using 4 initial clusters, with birth and merge proposal moves.

```
K4_trained_model, K4_info_dict = bnpy.run(
    dataset, 'DPMixtureModel', 'Gauss', 'memoVB',
    output_path='/tmp/AsteriskK8/trymoves-K=4/',
    nLap=100, nTask=1, nBatch=1,
    sF=0.1, ECovMat='eye',
    K=4, initname='randexamples',
    moves='birth,merge,shuffle',
    m_startLap=5, b_startLap=2, b_Kfresh=4)

show_clusters_over_time(K4_info_dict['task_output_path'])
```



Out:

```
Dataset Summary:
X Data
  total size: 5000 units
  batch size: 5000 units
  num. batches: 1
```

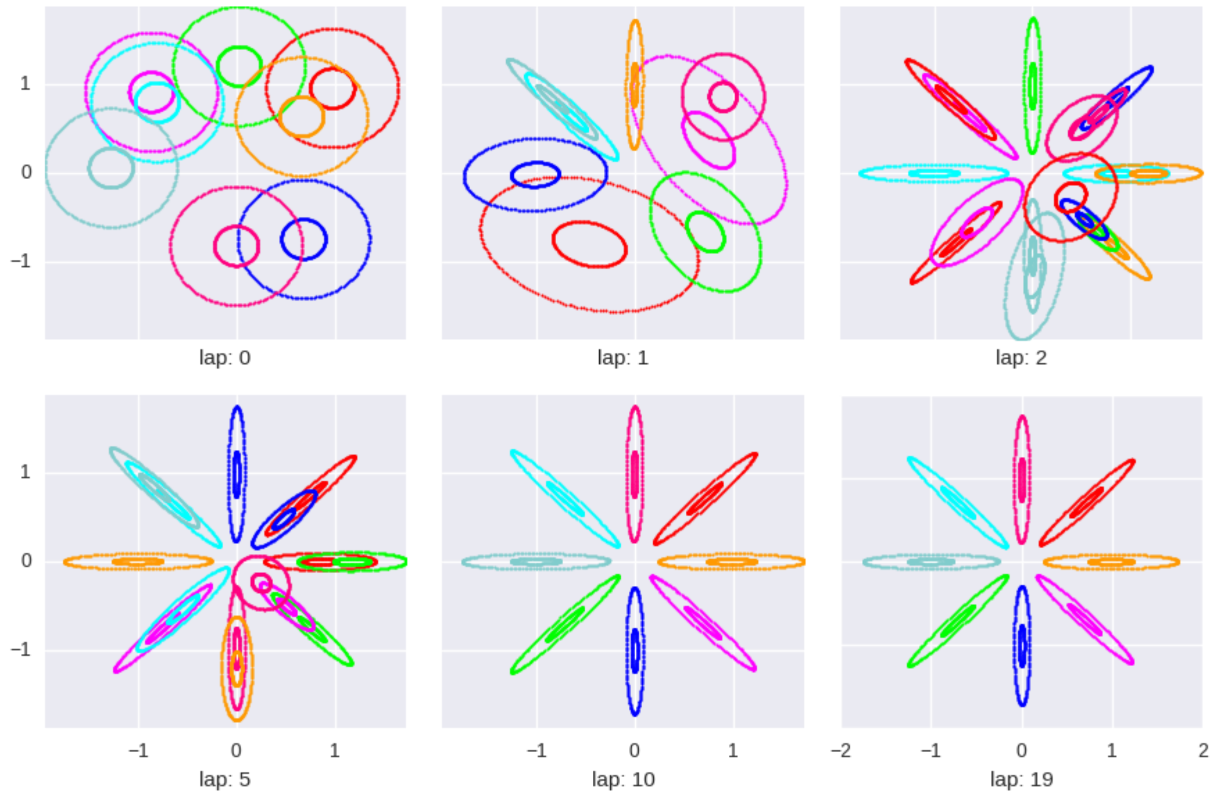
```
Allocation Model:  DP mixture with K=0. Concentration gamma0= 1.00
Obs. Data  Model:  Gaussian with full covariance.
Obs. Data  Prior:  Gauss-Wishart on mean and covar of each cluster
  E[  mean[k] ] =
   [ 0.  0.]
  E[ covar[k] ] =
  [[ 0.1  0. ]
   [ 0.   0.1]]
Initialization:
  initname = randexamples
  K = 4 (number of clusters)
  seed = 1607680
  elapsed_time: 0.0 sec
Learn Alg: memoVB | task  1/1 | alg. seed: 1607680 | data order seed: 8541952
task_output_path: /tmp/AsteriskK8/trymoves-K=4/1
BIRTH @ lap 1.00: Disabled. Waiting for lap >= 2 (--b_startLap).
MERGE @ lap 1.00: Disabled. Cannot plan merge on first lap. Need valid SS that represent wh
    1.000/100 after      0 sec. |    130.2 MiB | K    4 | loss  9.001549009e-01 |
MERGE @ lap 2.00: Disabled. Waiting for lap >= 5 (--m_startLap).
BIRTH @ lap 2.00 : Added 8 states. 2/4 succeeded. 2/4 failed eval phase. 0/4 failed build p
    2.000/100 after      0 sec. |    130.2 MiB | K   12 | loss  6.597743621e-03 |
MERGE @ lap 3.00: Disabled. Waiting for lap >= 5 (--m_startLap).
BIRTH @ lap 3.00 : Added 0 states. 0/9 succeeded. 9/9 failed eval phase. 0/9 failed build p
    3.000/100 after      1 sec. |    130.2 MiB | K   12 | loss -2.949554282e-02 | Ndiff   8
MERGE @ lap 4.00: Disabled. Waiting for lap >= 5 (--m_startLap).
BIRTH @ lap 4.00 : Added 0 states. 0/3 succeeded. 3/3 failed eval phase. 0/3 failed build p
    4.000/100 after      2 sec. |    130.2 MiB | K   12 | loss -3.255430235e-02 | Ndiff   1
BIRTH @ lap 5.000 : None attempted. 0 past failures. 0 too small. 12 too busy.
MERGE @ lap 5.00 : 2/25 accepted. Ndiff 0.00. 5 skipped.
    5.000/100 after      2 sec. |    130.2 MiB | K   10 | loss -3.311213517e-02 | Ndiff   1
BIRTH @ lap 6.000 : None attempted. 1 past failures. 0 too small. 9 too busy.
MERGE @ lap 6.00 : 2/17 accepted. Ndiff 417.46. 3 skipped.
    6.000/100 after      2 sec. |    130.2 MiB | K    8 | loss -4.787665915e-02 | Ndiff   1
BIRTH @ lap 7.000 : None attempted. 0 past failures. 0 too small. 8 too busy.
MERGE @ lap 7.00 : 0/9 accepted. Ndiff 0.00. 0 skipped.
    7.000/100 after      3 sec. |    130.2 MiB | K    8 | loss -4.787962431e-02 | Ndiff
BIRTH @ lap 8.00 : Added 0 states. 0/1 succeeded. 1/1 failed eval phase. 0/1 failed build p
MERGE @ lap 8.00 : 0/2 accepted. Ndiff 0.00. 0 skipped.
    8.000/100 after      3 sec. |    130.2 MiB | K    8 | loss -4.787962433e-02 | Ndiff
MERGE @ lap 9.00: No promising candidates, so no attempts.
BIRTH @ lap 9.00 : Added 0 states. 0/1 succeeded. 1/1 failed eval phase. 0/1 failed build p
    9.000/100 after      3 sec. |    130.2 MiB | K    8 | loss -4.787962433e-02 | Ndiff
BIRTH @ lap 10.000 : None attempted. 1 past failures. 0 too small. 7 too busy.
MERGE @ lap 10.00 : 0/9 accepted. Ndiff 0.00. 0 skipped.
   10.000/100 after      3 sec. |    130.2 MiB | K    8 | loss -4.787962433e-02 | Ndiff
BIRTH @ lap 11.000 : None attempted. 2 past failures. 0 too small. 6 too busy.
MERGE @ lap 11.00 : 0/8 accepted. Ndiff 0.00. 0 skipped.
   11.000/100 after      3 sec. |    130.2 MiB | K    8 | loss -4.787962433e-02 | Ndiff
BIRTH @ lap 12.000 : None attempted. 0 past failures. 0 too small. 8 too busy.
MERGE @ lap 12.00 : 0/9 accepted. Ndiff 0.00. 0 skipped.
   12.000/100 after      4 sec. |    130.2 MiB | K    8 | loss -4.787962433e-02 | Ndiff
BIRTH @ lap 13.000 : None attempted. 5 past failures. 0 too small. 3 too busy.
MERGE @ lap 13.00 : 0/2 accepted. Ndiff 0.00. 0 skipped.
```

```
   13.000/100 after      4 sec. |    130.2 MiB | K    8 | loss -4.787962433e-02 | Ndiff
MERGE @ lap 14.00: No promising candidates, so no attempts.
BIRTH @ lap 14.000 : None attempted. 8 past failures. 0 too small. 0 too busy.
   14.000/100 after      4 sec. |    130.2 MiB | K    8 | loss -4.787962433e-02 | Ndiff
BIRTH @ lap 15.000 : None attempted. 1 past failures. 0 too small. 7 too busy.
MERGE @ lap 15.00 : 0/9 accepted. Ndiff 0.00. 0 skipped.
   15.000/100 after      4 sec. |    130.2 MiB | K    8 | loss -4.787962433e-02 | Ndiff
BIRTH @ lap 16.000 : None attempted. 2 past failures. 0 too small. 6 too busy.
MERGE @ lap 16.00 : 0/8 accepted. Ndiff 0.00. 0 skipped.
   16.000/100 after      4 sec. |    130.2 MiB | K    8 | loss -4.787962433e-02 | Ndiff
BIRTH @ lap 17.000 : None attempted. 0 past failures. 0 too small. 8 too busy.
MERGE @ lap 17.00 : 0/9 accepted. Ndiff 0.00. 0 skipped.
   17.000/100 after      4 sec. |    130.2 MiB | K    8 | loss -4.787962433e-02 | Ndiff
... done. converged.
SKIPPED 2 comps with size below 0.00
```

Training from K=8 cluster

Now using 8 initial clusters

```
K8_trained_model, K8_info_dict = bnpy.run(
    dataset, 'DPMixtureModel', 'Gauss', 'memoVB',
    output_path='/tmp/AsteriskK8/trymoves-K=8/',
    nLap=100, nTask=1, nBatch=1,
    sF=0.1, ECovMat='eye',
    K=8, initname='randexamples',
    moves='birth,merge,shuffle',
    m_startLap=5, b_startLap=2, b_Kfresh=4)

show_clusters_over_time(K8_info_dict['task_output_path'])
```



Out:

```
Dataset Summary:
X Data
  total size: 5000 units
  batch size: 5000 units
  num. batches: 1
Allocation Model:  DP mixture with K=0. Concentration gamma0= 1.00
Obs. Data  Model:  Gaussian with full covariance.
Obs. Data  Prior:  Gauss-Wishart on mean and covar of each cluster
  E[  mean[k] ] =
   [ 0.   0.]
  E[ covar[k] ] =
  [[ 0.1   0. ]
   [ 0.    0.1]]
Initialization:
  initname = randexamples
  K = 8 (number of clusters)
  seed = 1607680
  elapsed_time: 0.0 sec
Learn Alg: memoVB | task  1/1 | alg. seed: 1607680 | data order seed: 8541952
task_output_path: /tmp/AsteriskK8/trymoves-K=8/1
BIRTH @ lap 1.00: Disabled. Waiting for lap >= 2 (--b_startLap).
MERGE @ lap 1.00: Disabled. Cannot plan merge on first lap. Need valid SS that represent wh
    1.000/100 after      0 sec. |    132.3 MiB | K    8 | loss  6.479391366e-01 |
MERGE @ lap 2.00: Disabled. Waiting for lap >= 5 (--m_startLap).
BIRTH @ lap 2.00 : Added 12 states. 3/8 succeeded. 5/8 failed eval phase. 0/8 failed build
```

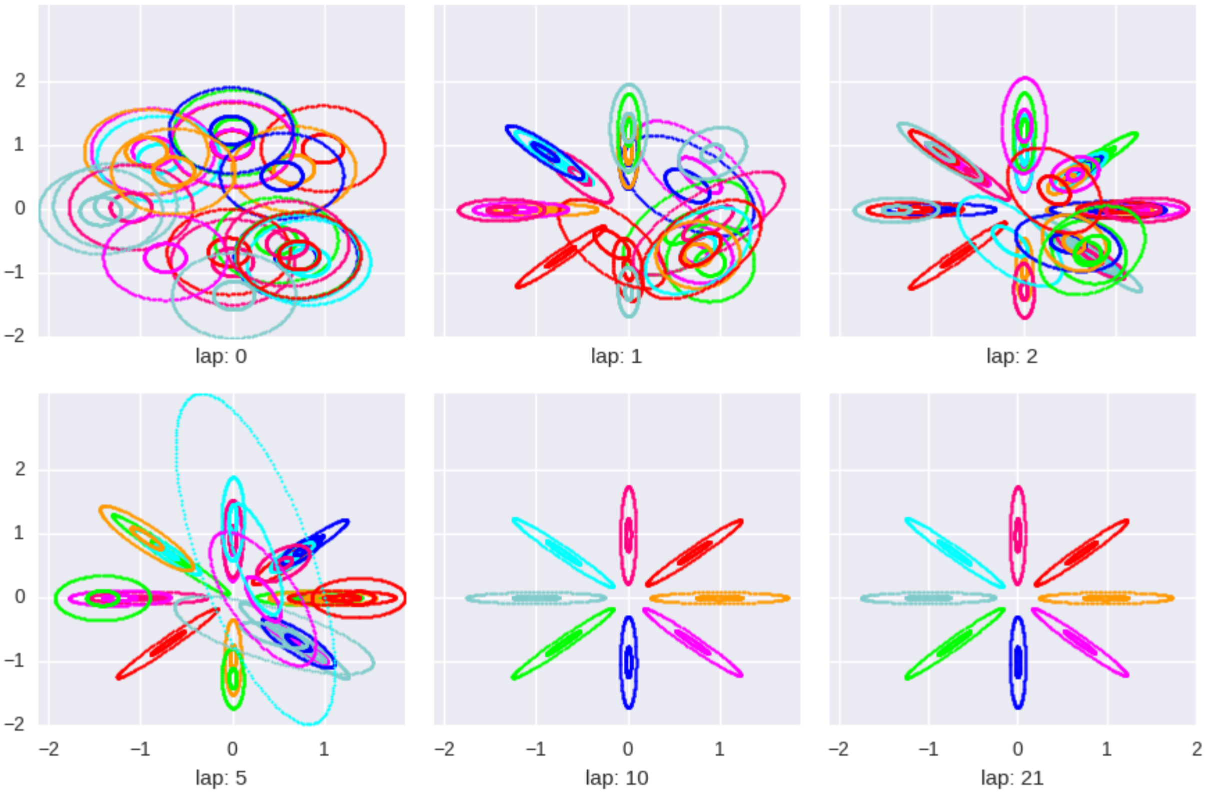
```
    2.000/100 after      1 sec. |    132.4 MiB | K   20 | loss  7.345925210e-02 |
MERGE @ lap 3.00: Disabled. Waiting for lap >= 5 (--m_startLap).
BIRTH @ lap 3.00 : Added 0 states. 0/13 succeeded. 8/13 failed eval phase. 5/13 failed buil
    3.000/100 after      2 sec. |    132.4 MiB | K   20 | loss  1.384405895e-02 | Ndiff   7
MERGE @ lap 4.00: Disabled. Waiting for lap >= 5 (--m_startLap).
BIRTH @ lap 4.00 : Added 0 states. 0/12 succeeded. 8/12 failed eval phase. 4/12 failed buil
    4.000/100 after      3 sec. |    132.4 MiB | K   20 | loss  1.057016337e-02 | Ndiff   1
BIRTH @ lap 5.000 : None attempted. 0 past failures. 0 too small. 20 too busy.
MERGE @ lap 5.00 : 4/41 accepted. Ndiff 635.65. 6 skipped.
    5.000/100 after      4 sec. |    132.4 MiB | K   16 | loss -3.619883330e-03 | Ndiff   1
BIRTH @ lap 6.000 : None attempted. 0 past failures. 0 too small. 16 too busy.
MERGE @ lap 6.00 : 4/26 accepted. Ndiff 459.69. 10 skipped.
    6.000/100 after      5 sec. |    132.4 MiB | K   12 | loss -2.198388303e-02 | Ndiff   1
BIRTH @ lap 7.000 : None attempted. 0 past failures. 0 too small. 12 too busy.
MERGE @ lap 7.00 : 3/21 accepted. Ndiff 366.43. 7 skipped.
    7.000/100 after      5 sec. |    132.4 MiB | K    9 | loss -4.222694604e-02 | Ndiff   1
BIRTH @ lap 8.000 : None attempted. 1 past failures. 0 too small. 8 too busy.
MERGE @ lap 8.00 : 1/16 accepted. Ndiff 75.14. 1 skipped.
    8.000/100 after      5 sec. |    132.4 MiB | K    8 | loss -4.787951351e-02 | Ndiff   1
BIRTH @ lap 9.00 : Added 0 states. 0/1 succeeded. 1/1 failed eval phase. 0/1 failed build p
MERGE @ lap 9.00 : 0/7 accepted. Ndiff 0.00. 0 skipped.
    9.000/100 after      6 sec. |    132.4 MiB | K    8 | loss -4.787962433e-02 | Ndiff
BIRTH @ lap 10.00 : Added 0 states. 0/4 succeeded. 4/4 failed eval phase. 0/4 failed build
MERGE @ lap 10.00 : 0/3 accepted. Ndiff 0.00. 0 skipped.
   10.000/100 after      6 sec. |    132.4 MiB | K    8 | loss -4.787962433e-02 | Ndiff
BIRTH @ lap 11.00 : Added 0 states. 0/1 succeeded. 1/1 failed eval phase. 0/1 failed build
MERGE @ lap 11.00 : 0/2 accepted. Ndiff 0.00. 0 skipped.
   11.000/100 after      6 sec. |    132.4 MiB | K    8 | loss -4.787962433e-02 | Ndiff
BIRTH @ lap 12.000 : None attempted. 3 past failures. 0 too small. 5 too busy.
MERGE @ lap 12.00 : 0/5 accepted. Ndiff 0.00. 0 skipped.
   12.000/100 after      6 sec. |    132.4 MiB | K    8 | loss -4.787962433e-02 | Ndiff
BIRTH @ lap 13.000 : None attempted. 1 past failures. 0 too small. 7 too busy.
MERGE @ lap 13.00 : 0/11 accepted. Ndiff 0.00. 0 skipped.
   13.000/100 after      6 sec. |    132.4 MiB | K    8 | loss -4.787962433e-02 | Ndiff
BIRTH @ lap 14.000 : None attempted. 1 past failures. 0 too small. 7 too busy.
MERGE @ lap 14.00 : 0/7 accepted. Ndiff 0.00. 0 skipped.
   14.000/100 after      7 sec. |    132.4 MiB | K    8 | loss -4.787962433e-02 | Ndiff
BIRTH @ lap 15.000 : None attempted. 5 past failures. 0 too small. 3 too busy.
MERGE @ lap 15.00 : 0/3 accepted. Ndiff 0.00. 0 skipped.
   15.000/100 after      7 sec. |    132.4 MiB | K    8 | loss -4.787962433e-02 | Ndiff
BIRTH @ lap 16.000 : None attempted. 5 past failures. 0 too small. 3 too busy.
MERGE @ lap 16.00 : 0/2 accepted. Ndiff 0.00. 0 skipped.
   16.000/100 after      7 sec. |    132.4 MiB | K    8 | loss -4.787962433e-02 | Ndiff
BIRTH @ lap 17.000 : None attempted. 3 past failures. 0 too small. 5 too busy.
MERGE @ lap 17.00 : 0/5 accepted. Ndiff 0.00. 0 skipped.
   17.000/100 after      7 sec. |    132.4 MiB | K    8 | loss -4.787962433e-02 | Ndiff
BIRTH @ lap 18.000 : None attempted. 1 past failures. 0 too small. 7 too busy.
MERGE @ lap 18.00 : 0/11 accepted. Ndiff 0.00. 0 skipped.
   18.000/100 after      7 sec. |    132.4 MiB | K    8 | loss -4.787962433e-02 | Ndiff
BIRTH @ lap 19.000 : None attempted. 1 past failures. 0 too small. 7 too busy.
MERGE @ lap 19.00 : 0/7 accepted. Ndiff 0.00. 0 skipped.
   19.000/100 after      7 sec. |    132.4 MiB | K    8 | loss -4.787962433e-02 | Ndiff
... done. converged.
```

```
SKIPPED 3 comps with size below 0.00
SKIPPED 1 comps with size below 0.00
```

Training from K=25 cluster

Now using 25 initial clusters

```
K25_trained_model, K25_info_dict = bnpy.run(
    dataset, 'DPMixtureModel', 'Gauss', 'memoVB',
    output_path='/tmp/AsteriskK8/trymoves-K=25/',
    nLap=100, nTask=1, nBatch=1,
    sF=0.1, ECovMat='eye',
    K=25, initname='randexamples',
    moves='birth,merge,shuffle',
    m_startLap=5, b_startLap=2, b_Kfresh=4)

show_clusters_over_time(K25_info_dict['task_output_path'])
```



Out:

```
Dataset Summary:
X Data
  total size: 5000 units
  batch size: 5000 units
  num. batches: 1
```

```
Allocation Model:  DP mixture with K=0. Concentration gamma0= 1.00
Obs. Data  Model:  Gaussian with full covariance.
Obs. Data  Prior:  Gauss-Wishart on mean and covar of each cluster
  E[  mean[k] ] =
   [ 0.  0.]
  E[ covar[k] ] =
  [[ 0.1  0. ]
   [ 0.   0.1]]
Initialization:
  initname = randexamples
  K = 25 (number of clusters)
  seed = 1607680
  elapsed_time: 0.0 sec
Learn Alg: memoVB | task  1/1 | alg. seed: 1607680 | data order seed: 8541952
task_output_path: /tmp/AsteriskK8/trymoves-K=25/1
BIRTH @ lap 1.00: Disabled. Waiting for lap >= 2 (--b_startLap).
MERGE @ lap 1.00: Disabled. Cannot plan merge on first lap. Need valid SS that represent wh
    1.000/100 after      0 sec. |    132.2 MiB | K   25 | loss  4.392317798e-01 |
MERGE @ lap 2.00: Disabled. Waiting for lap >= 5 (--m_startLap).
BIRTH @ lap 2.00 : Added 14 states. 4/18 succeeded. 11/18 failed eval phase. 3/18 failed bu
    2.000/100 after      2 sec. |    132.2 MiB | K   39 | loss  1.219944497e-01 |
MERGE @ lap 3.00: Disabled. Waiting for lap >= 5 (--m_startLap).
BIRTH @ lap 3.00 : Added 0 states. 0/15 succeeded. 3/15 failed eval phase. 12/15 failed bui
    3.000/100 after      3 sec. |    135.2 MiB | K   39 | loss  8.910722771e-02 | Ndiff   5
MERGE @ lap 4.00: Disabled. Waiting for lap >= 5 (--m_startLap).
BIRTH @ lap 4.00 : Added 0 states. 0/15 succeeded. 9/15 failed eval phase. 6/15 failed buil
    4.000/100 after      5 sec. |    135.2 MiB | K   39 | loss  8.017398425e-02 | Ndiff   3
BIRTH @ lap 5.000 : None attempted. 0 past failures. 0 too small. 39 too busy.
MERGE @ lap 5.00 : 8/63 accepted. Ndiff 338.97. 32 skipped.
    5.000/100 after      7 sec. |    135.2 MiB | K   31 | loss  5.747324378e-02 | Ndiff   3
BIRTH @ lap 6.000 : None attempted. 0 past failures. 0 too small. 31 too busy.
MERGE @ lap 6.00 : 10/39 accepted. Ndiff 1032.92. 38 skipped.
    6.000/100 after      8 sec. |    135.2 MiB | K   21 | loss  1.309182046e-02 | Ndiff   3
BIRTH @ lap 7.000 : None attempted. 0 past failures. 0 too small. 21 too busy.
MERGE @ lap 7.00 : 5/35 accepted. Ndiff 323.06. 14 skipped.
    7.000/100 after      9 sec. |    135.2 MiB | K   16 | loss -8.888342290e-03 | Ndiff   3
BIRTH @ lap 8.000 : None attempted. 0 past failures. 0 too small. 16 too busy.
MERGE @ lap 8.00 : 4/24 accepted. Ndiff 443.50. 12 skipped.
    8.000/100 after      9 sec. |    135.2 MiB | K   12 | loss -3.261743488e-02 | Ndiff   3
BIRTH @ lap 9.000 : None attempted. 0 past failures. 0 too small. 12 too busy.
MERGE @ lap 9.00 : 3/22 accepted. Ndiff 185.39. 7 skipped.
    9.000/100 after     10 sec. |    135.2 MiB | K    9 | loss -4.523238984e-02 | Ndiff   3
BIRTH @ lap 10.00 : Added 0 states. 0/1 succeeded. 1/1 failed eval phase. 0/1 failed build
MERGE @ lap 10.00 : 1/13 accepted. Ndiff 1.72. 1 skipped.
   10.000/100 after     10 sec. |    135.2 MiB | K    8 | loss -4.770137591e-02 | Ndiff   3
BIRTH @ lap 11.00 : Added 0 states. 0/3 succeeded. 3/3 failed eval phase. 0/3 failed build
MERGE @ lap 11.00 : 0/4 accepted. Ndiff 0.00. 0 skipped.
   11.000/100 after     10 sec. |    135.2 MiB | K    8 | loss -4.787962433e-02 | Ndiff
MERGE @ lap 12.00: No promising candidates, so no attempts.
BIRTH @ lap 12.00 : Added 0 states. 0/3 succeeded. 3/3 failed eval phase. 0/3 failed build
   12.000/100 after     11 sec. |    135.2 MiB | K    8 | loss -4.787962433e-02 | Ndiff
BIRTH @ lap 13.000 : None attempted. 3 past failures. 0 too small. 5 too busy.
MERGE @ lap 13.00 : 0/4 accepted. Ndiff 0.00. 0 skipped.
```

```
   13.000/100 after     11 sec. |    135.2 MiB | K    8 | loss -4.787962433e-02 | Ndiff
BIRTH @ lap 14.000 : None attempted. 2 past failures. 0 too small. 6 too busy.
MERGE @ lap 14.00 : 0/11 accepted. Ndiff 0.00. 0 skipped.
   14.000/100 after     11 sec. |    135.2 MiB | K    8 | loss -4.787962433e-02 | Ndiff
BIRTH @ lap 15.000 : None attempted. 2 past failures. 0 too small. 6 too busy.
MERGE @ lap 15.00 : 0/9 accepted. Ndiff 0.00. 0 skipped.
   15.000/100 after     11 sec. |    135.2 MiB | K    8 | loss -4.787962433e-02 | Ndiff
BIRTH @ lap 16.000 : None attempted. 4 past failures. 0 too small. 4 too busy.
MERGE @ lap 16.00 : 0/4 accepted. Ndiff 0.00. 0 skipped.
   16.000/100 after     11 sec. |    135.2 MiB | K    8 | loss -4.787962433e-02 | Ndiff
MERGE @ lap 17.00: No promising candidates, so no attempts.
BIRTH @ lap 17.000 : None attempted. 8 past failures. 0 too small. 0 too busy.
   17.000/100 after     11 sec. |    135.2 MiB | K    8 | loss -4.787962433e-02 | Ndiff
BIRTH @ lap 18.000 : None attempted. 3 past failures. 0 too small. 5 too busy.
MERGE @ lap 18.00 : 0/4 accepted. Ndiff 0.00. 0 skipped.
   18.000/100 after     11 sec. |    135.2 MiB | K    8 | loss -4.787962433e-02 | Ndiff
BIRTH @ lap 19.000 : None attempted. 2 past failures. 0 too small. 6 too busy.
MERGE @ lap 19.00 : 0/11 accepted. Ndiff 0.00. 0 skipped.
   19.000/100 after     12 sec. |    135.2 MiB | K    8 | loss -4.787962433e-02 | Ndiff
BIRTH @ lap 20.000 : None attempted. 2 past failures. 0 too small. 6 too busy.
MERGE @ lap 20.00 : 0/9 accepted. Ndiff 0.00. 0 skipped.
   20.000/100 after     12 sec. |    135.2 MiB | K    8 | loss -4.787962433e-02 | Ndiff
BIRTH @ lap 21.000 : None attempted. 4 past failures. 0 too small. 4 too busy.
MERGE @ lap 21.00 : 0/4 accepted. Ndiff 0.00. 0 skipped.
   21.000/100 after     12 sec. |    135.2 MiB | K    8 | loss -4.787962433e-02 | Ndiff
... done. converged.
SKIPPED 4 comps with size below 0.00
SKIPPED 3 comps with size below 0.00
```

Total running time of the script: (0 minutes 32.509 seconds)

`Download Python source code: plot-03-demo=vb+proposals-model=dp_mix+gauss.py`

`Download Jupyter notebook: plot-03-demo=vb+proposals-model=dp_mix+gauss.ipynb`

Generated by Sphinx-Gallery

5.2.2 Standard Normal dataset

Variational training of DP mixture models on thousands of points from a single 1D Gaussian with mean 0 and variance 1.

Variational with merge and delete proposals for DP mixtures of Gaussians

How delete moves can be more effective than merges.

In this example, we show how merge moves alone may not be enough to reliably escape local optima. Instead, we show that more flexible delete moves can escape from situations where merges alone fail.

```python
import bnpy
import numpy as np
import os
```

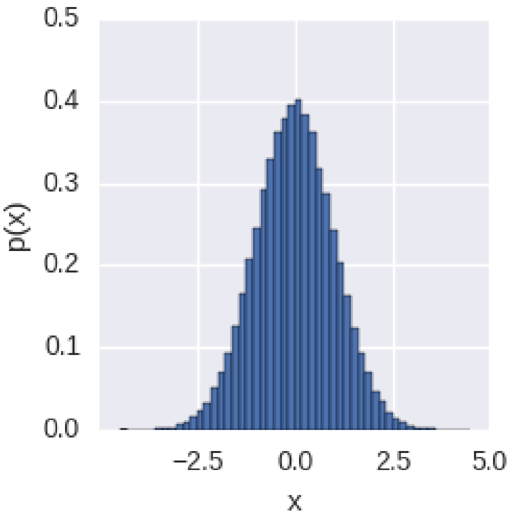
```
from matplotlib import pylab
import seaborn as sns

FIG_SIZE = (3, 3)
pylab.rcParams['figure.figsize'] = FIG_SIZE
```

Create toy dataset of many points drawn from standard normal

```
prng = np.random.RandomState(42)
X = prng.randn(100000, 1)
dataset = bnpy.data.XData(X, name='StandardNormalK1')
```

Make a simple plot of the raw data

```
pylab.hist(dataset.X[:, 0], 50, normed=1)
pylab.xlabel('x')
pylab.ylabel('p(x)')
pylab.tight_layout()
```



Setup: Determine specific settings of the proposals

```
merge_kwargs = dict(
    m_startLap=10,
    m_pair_ranking_procedure='total_size',
    m_pair_ranking_direction='descending',
    )

delete_kwargs = dict(
    d_startLap=10,
    d_nRefineSteps=50,
    )
```

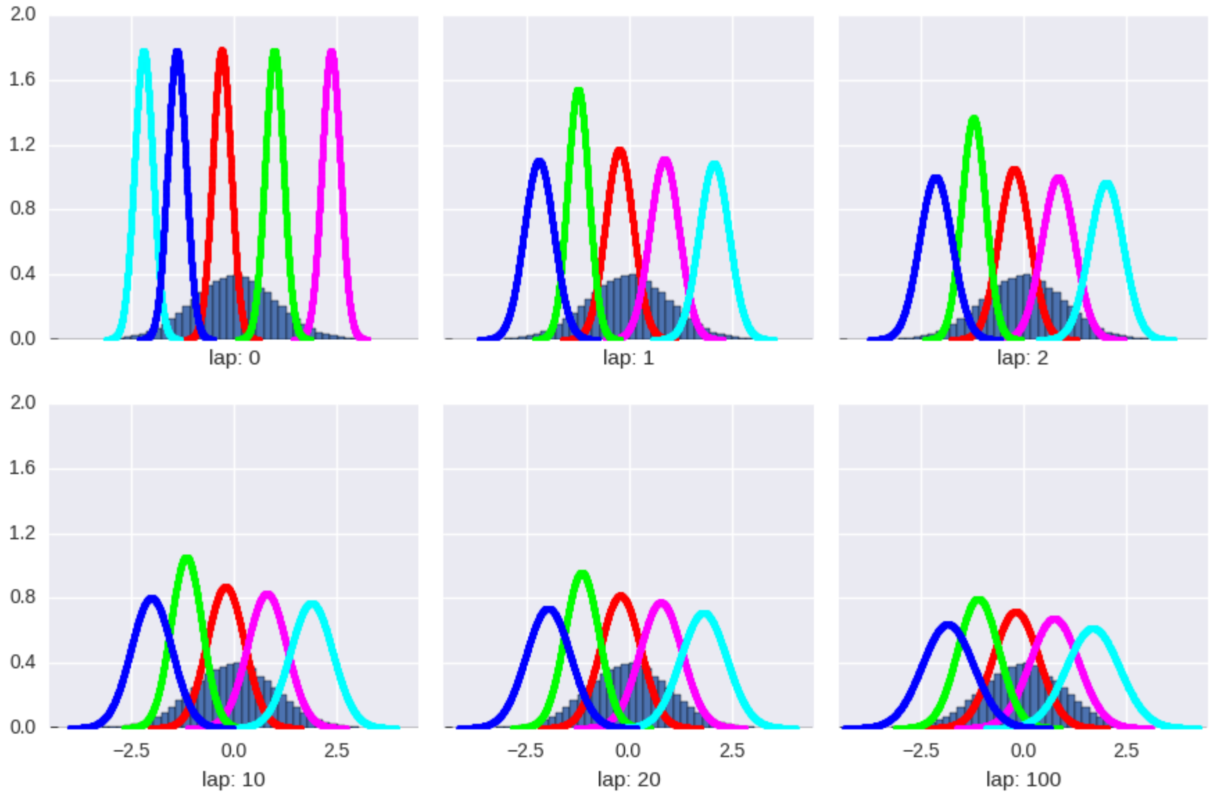
Setup: Helper function to display the learned clusters

```
def show_clusters_over_time(
        task_output_path=None,
        query_laps=[0, 1, 2, 10, 20, None],
        nrows=2):
    '''
    '''
    ncols = int(np.ceil(len(query_laps) // float(nrows)))
    fig_handle, ax_handle_list = pylab.subplots(
        figsize=(FIG_SIZE[0] * ncols, FIG_SIZE[1] * nrows),
        nrows=nrows, ncols=ncols, sharex=True, sharey=True)
    for plot_id, lap_val in enumerate(query_laps):
        cur_model, lap_val = bnpy.load_model_at_lap(task_output_path, lap_val)
        cur_ax_handle = ax_handle_list.flatten()[plot_id]
        bnpy.viz.PlotComps.plotCompsFromHModel(
            cur_model, dataset=dataset, ax_handle=cur_ax_handle)
        cur_ax_handle.set_xlim([-4.5, 4.5])
        cur_ax_handle.set_xlabel("lap: %d" % lap_val)
    pylab.tight_layout()
```

Run with *merge* moves only, from K=5 initial clusters

Unfortunately, no pairwise merge is accepted. The model is stuck using 5 clusters when one cluster would do.

```
gamma = 5.0
sF = 0.1
K = 5

m_trained_model, m_info_dict = bnpy.run(
    dataset, 'DPMixtureModel', 'Gauss', 'memoVB',
    output_path=('/tmp/StandardNormalK1/' +
        'trymoves-K=%d-gamma=%s-ECovMat=%s*eye-moves=merge,shuffle/' % (
            K, gamma, sF)),
    nLap=100, nTask=1, nBatch=1,
    gamma0=gamma, sF=sF, ECovMat='eye',
    K=K, initname='randexamplesbydist',
    moves='merge,shuffle',
    **dict(**merge_kwargs))

show_clusters_over_time(m_info_dict['task_output_path'])
```



Out:

```
Dataset Summary:
X Data
  total size: 100000 units
  batch size: 100000 units
  num. batches: 1
Allocation Model:  DP mixture with K=0. Concentration gamma0= 5.00
Obs. Data  Model:  Gaussian with full covariance.
Obs. Data  Prior:  Gauss-Wishart on mean and covar of each cluster
  E[  mean[k] ] =
   [ 0.]
  E[ covar[k] ] =
  [[ 0.1]]
Initialization:
  initname = randexamplesbydist
  K = 5 (number of clusters)
  seed = 1607680
  elapsed_time: 0.0 sec
Learn Alg: memoVB | task  1/1 | alg. seed: 1607680 | data order seed: 8541952
task_output_path: /tmp/StandardNormalK1/trymoves-K=5-gamma=5.0-ECovMat=0.1*eye-moves=merge/
MERGE @ lap 1.00: Disabled. Cannot plan merge on first lap. Need valid SS that represent wh
    1.000/100 after      0 sec. |    143.8 MiB | K    5 | loss  1.547776525e+00 |
MERGE @ lap 2.00: Disabled. Waiting for lap >= 10 (--m_startLap).
    2.000/100 after      0 sec. |    143.8 MiB | K    5 | loss  1.446186110e+00 | Ndiff  25
MERGE @ lap 3.00: Disabled. Waiting for lap >= 10 (--m_startLap).
    3.000/100 after      0 sec. |    143.8 MiB | K    5 | loss  1.434177601e+00 | Ndiff  12
```

```
MERGE @ lap 4.00: Disabled. Waiting for lap >= 10 (--m_startLap).
    4.000/100 after      0 sec. |    143.8 MiB | K    5 | loss  1.429642157e+00 | Ndiff
MERGE @ lap 5.00: Disabled. Waiting for lap >= 10 (--m_startLap).
    5.000/100 after      0 sec. |    143.8 MiB | K    5 | loss  1.427329685e+00 | Ndiff
MERGE @ lap 6.00: Disabled. Waiting for lap >= 10 (--m_startLap).
    6.000/100 after      1 sec. |    143.8 MiB | K    5 | loss  1.425951747e+00 | Ndiff
MERGE @ lap 7.00: Disabled. Waiting for lap >= 10 (--m_startLap).
    7.000/100 after      1 sec. |    143.8 MiB | K    5 | loss  1.425047267e+00 | Ndiff
MERGE @ lap 8.00: Disabled. Waiting for lap >= 10 (--m_startLap).
    8.000/100 after      1 sec. |    143.8 MiB | K    5 | loss  1.424412975e+00 | Ndiff
MERGE @ lap 9.00: Disabled. Waiting for lap >= 10 (--m_startLap).
    9.000/100 after      1 sec. |    143.8 MiB | K    5 | loss  1.423946233e+00 | Ndiff
MERGE @ lap 10.00 : 0/10 accepted. Ndiff 0.00. 0 skipped.
   10.000/100 after      1 sec. |    143.8 MiB | K    5 | loss  1.423589980e+00 | Ndiff
MERGE @ lap 11.00: No promising candidates, so no attempts.
   11.000/100 after      1 sec. |    143.8 MiB | K    5 | loss  1.423310120e+00 | Ndiff
MERGE @ lap 12.00: No promising candidates, so no attempts.
   12.000/100 after      1 sec. |    143.8 MiB | K    5 | loss  1.423085107e+00 | Ndiff
MERGE @ lap 13.00: No promising candidates, so no attempts.
   13.000/100 after      1 sec. |    143.8 MiB | K    5 | loss  1.422900698e+00 | Ndiff
MERGE @ lap 14.00: No promising candidates, so no attempts.
   14.000/100 after      1 sec. |    143.8 MiB | K    5 | loss  1.422747120e+00 | Ndiff
MERGE @ lap 15.00 : 0/10 accepted. Ndiff 0.00. 0 skipped.
   15.000/100 after      2 sec. |    143.8 MiB | K    5 | loss  1.422617464e+00 | Ndiff
MERGE @ lap 16.00: No promising candidates, so no attempts.
   16.000/100 after      2 sec. |    143.8 MiB | K    5 | loss  1.422506710e+00 | Ndiff
MERGE @ lap 17.00: No promising candidates, so no attempts.
   17.000/100 after      2 sec. |    143.8 MiB | K    5 | loss  1.422411133e+00 | Ndiff
MERGE @ lap 18.00: No promising candidates, so no attempts.
   18.000/100 after      2 sec. |    143.8 MiB | K    5 | loss  1.422327908e+00 | Ndiff
MERGE @ lap 19.00: No promising candidates, so no attempts.
   19.000/100 after      2 sec. |    143.8 MiB | K    5 | loss  1.422254864e+00 | Ndiff
MERGE @ lap 20.00 : 0/10 accepted. Ndiff 0.00. 0 skipped.
   20.000/100 after      2 sec. |    143.8 MiB | K    5 | loss  1.422190300e+00 | Ndiff
MERGE @ lap 21.00: No promising candidates, so no attempts.
   21.000/100 after      2 sec. |    143.8 MiB | K    5 | loss  1.422132869e+00 | Ndiff
MERGE @ lap 22.00: No promising candidates, so no attempts.
   22.000/100 after      2 sec. |    143.8 MiB | K    5 | loss  1.422081490e+00 | Ndiff
MERGE @ lap 23.00: No promising candidates, so no attempts.
   23.000/100 after      2 sec. |    143.8 MiB | K    5 | loss  1.422035287e+00 | Ndiff
MERGE @ lap 24.00: No promising candidates, so no attempts.
   24.000/100 after      3 sec. |    143.8 MiB | K    5 | loss  1.421993542e+00 | Ndiff
MERGE @ lap 25.00 : 0/10 accepted. Ndiff 0.00. 0 skipped.
   25.000/100 after      3 sec. |    143.8 MiB | K    5 | loss  1.421955661e+00 | Ndiff
MERGE @ lap 26.00: No promising candidates, so no attempts.
   26.000/100 after      3 sec. |    143.8 MiB | K    5 | loss  1.421921151e+00 | Ndiff
MERGE @ lap 27.00: No promising candidates, so no attempts.
   27.000/100 after      3 sec. |    143.8 MiB | K    5 | loss  1.421889597e+00 | Ndiff
MERGE @ lap 28.00: No promising candidates, so no attempts.
   28.000/100 after      3 sec. |    143.8 MiB | K    5 | loss  1.421860649e+00 | Ndiff
MERGE @ lap 29.00: No promising candidates, so no attempts.
   29.000/100 after      3 sec. |    143.8 MiB | K    5 | loss  1.421834007e+00 | Ndiff
MERGE @ lap 30.00 : 0/10 accepted. Ndiff 0.00. 0 skipped.
```

```
   30.000/100 after      3 sec. |    143.8 MiB | K    5 | loss  1.421809418e+00 | Ndiff
MERGE @ lap 31.00: No promising candidates, so no attempts.
   31.000/100 after      3 sec. |    143.8 MiB | K    5 | loss  1.421786661e+00 | Ndiff
MERGE @ lap 32.00: No promising candidates, so no attempts.
   32.000/100 after      4 sec. |    143.8 MiB | K    5 | loss  1.421765547e+00 | Ndiff
MERGE @ lap 33.00: No promising candidates, so no attempts.
   33.000/100 after      4 sec. |    143.8 MiB | K    5 | loss  1.421745910e+00 | Ndiff
MERGE @ lap 34.00: No promising candidates, so no attempts.
   34.000/100 after      4 sec. |    143.8 MiB | K    5 | loss  1.421727607e+00 | Ndiff
MERGE @ lap 35.00 : 0/10 accepted. Ndiff 0.00. 0 skipped.
   35.000/100 after      4 sec. |    143.8 MiB | K    5 | loss  1.421710511e+00 | Ndiff
MERGE @ lap 36.00: No promising candidates, so no attempts.
   36.000/100 after      4 sec. |    143.8 MiB | K    5 | loss  1.421694512e+00 | Ndiff
MERGE @ lap 37.00: No promising candidates, so no attempts.
   37.000/100 after      4 sec. |    143.8 MiB | K    5 | loss  1.421679511e+00 | Ndiff
MERGE @ lap 38.00: No promising candidates, so no attempts.
   38.000/100 after      4 sec. |    143.8 MiB | K    5 | loss  1.421665422e+00 | Ndiff
MERGE @ lap 39.00: No promising candidates, so no attempts.
   39.000/100 after      4 sec. |    143.8 MiB | K    5 | loss  1.421652166e+00 | Ndiff
MERGE @ lap 40.00 : 0/10 accepted. Ndiff 0.00. 0 skipped.
   40.000/100 after      4 sec. |    143.8 MiB | K    5 | loss  1.421639676e+00 | Ndiff
MERGE @ lap 41.00: No promising candidates, so no attempts.
   41.000/100 after      5 sec. |    143.8 MiB | K    5 | loss  1.421627889e+00 | Ndiff
MERGE @ lap 42.00: No promising candidates, so no attempts.
   42.000/100 after      5 sec. |    143.8 MiB | K    5 | loss  1.421616749e+00 | Ndiff
MERGE @ lap 43.00: No promising candidates, so no attempts.
   43.000/100 after      5 sec. |    143.8 MiB | K    5 | loss  1.421606208e+00 | Ndiff
MERGE @ lap 44.00: No promising candidates, so no attempts.
   44.000/100 after      5 sec. |    143.8 MiB | K    5 | loss  1.421596220e+00 | Ndiff
MERGE @ lap 45.00 : 0/10 accepted. Ndiff 0.00. 0 skipped.
   45.000/100 after      5 sec. |    143.8 MiB | K    5 | loss  1.421586745e+00 | Ndiff
MERGE @ lap 46.00: No promising candidates, so no attempts.
   46.000/100 after      5 sec. |    143.8 MiB | K    5 | loss  1.421577745e+00 | Ndiff
MERGE @ lap 47.00: No promising candidates, so no attempts.
   47.000/100 after      5 sec. |    143.8 MiB | K    5 | loss  1.421569187e+00 | Ndiff
MERGE @ lap 48.00: No promising candidates, so no attempts.
   48.000/100 after      5 sec. |    143.8 MiB | K    5 | loss  1.421561041e+00 | Ndiff
MERGE @ lap 49.00: No promising candidates, so no attempts.
   49.000/100 after      5 sec. |    143.8 MiB | K    5 | loss  1.421553280e+00 | Ndiff
MERGE @ lap 50.00 : 0/10 accepted. Ndiff 0.00. 0 skipped.
   50.000/100 after      6 sec. |    143.8 MiB | K    5 | loss  1.421545876e+00 | Ndiff
MERGE @ lap 51.00: No promising candidates, so no attempts.
   51.000/100 after      6 sec. |    143.8 MiB | K    5 | loss  1.421538809e+00 | Ndiff
MERGE @ lap 52.00: No promising candidates, so no attempts.
   52.000/100 after      6 sec. |    143.8 MiB | K    5 | loss  1.421532055e+00 | Ndiff
MERGE @ lap 53.00: No promising candidates, so no attempts.
   53.000/100 after      6 sec. |    143.8 MiB | K    5 | loss  1.421525596e+00 | Ndiff
MERGE @ lap 54.00: No promising candidates, so no attempts.
   54.000/100 after      6 sec. |    143.8 MiB | K    5 | loss  1.421519413e+00 | Ndiff
MERGE @ lap 55.00 : 0/10 accepted. Ndiff 0.00. 0 skipped.
   55.000/100 after      6 sec. |    143.8 MiB | K    5 | loss  1.421513490e+00 | Ndiff
MERGE @ lap 56.00: No promising candidates, so no attempts.
   56.000/100 after      6 sec. |    143.8 MiB | K    5 | loss  1.421507811e+00 | Ndiff
```

```
MERGE @ lap 57.00: No promising candidates, so no attempts.
   57.000/100 after      6 sec. |    143.8 MiB | K    5 | loss  1.421502363e+00 | Ndiff
MERGE @ lap 58.00: No promising candidates, so no attempts.
   58.000/100 after      6 sec. |    143.8 MiB | K    5 | loss  1.421497133e+00 | Ndiff
MERGE @ lap 59.00: No promising candidates, so no attempts.
   59.000/100 after      7 sec. |    143.8 MiB | K    5 | loss  1.421492107e+00 | Ndiff
MERGE @ lap 60.00 : 0/10 accepted. Ndiff 0.00. 0 skipped.
   60.000/100 after      7 sec. |    143.8 MiB | K    5 | loss  1.421487275e+00 | Ndiff
MERGE @ lap 61.00: No promising candidates, so no attempts.
   61.000/100 after      7 sec. |    143.8 MiB | K    5 | loss  1.421482626e+00 | Ndiff
MERGE @ lap 62.00: No promising candidates, so no attempts.
   62.000/100 after      7 sec. |    143.8 MiB | K    5 | loss  1.421478151e+00 | Ndiff
MERGE @ lap 63.00: No promising candidates, so no attempts.
   63.000/100 after      7 sec. |    143.8 MiB | K    5 | loss  1.421473840e+00 | Ndiff
MERGE @ lap 64.00: No promising candidates, so no attempts.
   64.000/100 after      7 sec. |    143.8 MiB | K    5 | loss  1.421469686e+00 | Ndiff
MERGE @ lap 65.00 : 0/10 accepted. Ndiff 0.00. 0 skipped.
   65.000/100 after      7 sec. |    143.8 MiB | K    5 | loss  1.421465679e+00 | Ndiff
MERGE @ lap 66.00: No promising candidates, so no attempts.
   66.000/100 after      7 sec. |    143.8 MiB | K    5 | loss  1.421461813e+00 | Ndiff
MERGE @ lap 67.00: No promising candidates, so no attempts.
   67.000/100 after      8 sec. |    143.8 MiB | K    5 | loss  1.421458080e+00 | Ndiff
MERGE @ lap 68.00: No promising candidates, so no attempts.
   68.000/100 after      8 sec. |    143.8 MiB | K    5 | loss  1.421454475e+00 | Ndiff
MERGE @ lap 69.00: No promising candidates, so no attempts.
   69.000/100 after      8 sec. |    143.8 MiB | K    5 | loss  1.421450991e+00 | Ndiff
MERGE @ lap 70.00 : 0/10 accepted. Ndiff 0.00. 0 skipped.
   70.000/100 after      8 sec. |    143.8 MiB | K    5 | loss  1.421447622e+00 | Ndiff
MERGE @ lap 71.00: No promising candidates, so no attempts.
   71.000/100 after      8 sec. |    143.8 MiB | K    5 | loss  1.421444363e+00 | Ndiff
MERGE @ lap 72.00: No promising candidates, so no attempts.
   72.000/100 after      8 sec. |    143.8 MiB | K    5 | loss  1.421441210e+00 | Ndiff
MERGE @ lap 73.00: No promising candidates, so no attempts.
   73.000/100 after      8 sec. |    143.8 MiB | K    5 | loss  1.421438156e+00 | Ndiff
MERGE @ lap 74.00: No promising candidates, so no attempts.
   74.000/100 after      8 sec. |    143.8 MiB | K    5 | loss  1.421435198e+00 | Ndiff
MERGE @ lap 75.00 : 0/10 accepted. Ndiff 0.00. 0 skipped.
   75.000/100 after      9 sec. |    143.8 MiB | K    5 | loss  1.421432332e+00 | Ndiff
MERGE @ lap 76.00: No promising candidates, so no attempts.
   76.000/100 after      9 sec. |    143.8 MiB | K    5 | loss  1.421429553e+00 | Ndiff
MERGE @ lap 77.00: No promising candidates, so no attempts.
   77.000/100 after      9 sec. |    143.8 MiB | K    5 | loss  1.421426858e+00 | Ndiff
MERGE @ lap 78.00: No promising candidates, so no attempts.
   78.000/100 after      9 sec. |    143.8 MiB | K    5 | loss  1.421424243e+00 | Ndiff
MERGE @ lap 79.00: No promising candidates, so no attempts.
   79.000/100 after      9 sec. |    143.8 MiB | K    5 | loss  1.421421705e+00 | Ndiff
MERGE @ lap 80.00 : 0/10 accepted. Ndiff 0.00. 0 skipped.
   80.000/100 after      9 sec. |    143.8 MiB | K    5 | loss  1.421419241e+00 | Ndiff
MERGE @ lap 81.00: No promising candidates, so no attempts.
   81.000/100 after      9 sec. |    143.8 MiB | K    5 | loss  1.421416847e+00 | Ndiff
MERGE @ lap 82.00: No promising candidates, so no attempts.
   82.000/100 after      9 sec. |    143.8 MiB | K    5 | loss  1.421414521e+00 | Ndiff
MERGE @ lap 83.00: No promising candidates, so no attempts.
```

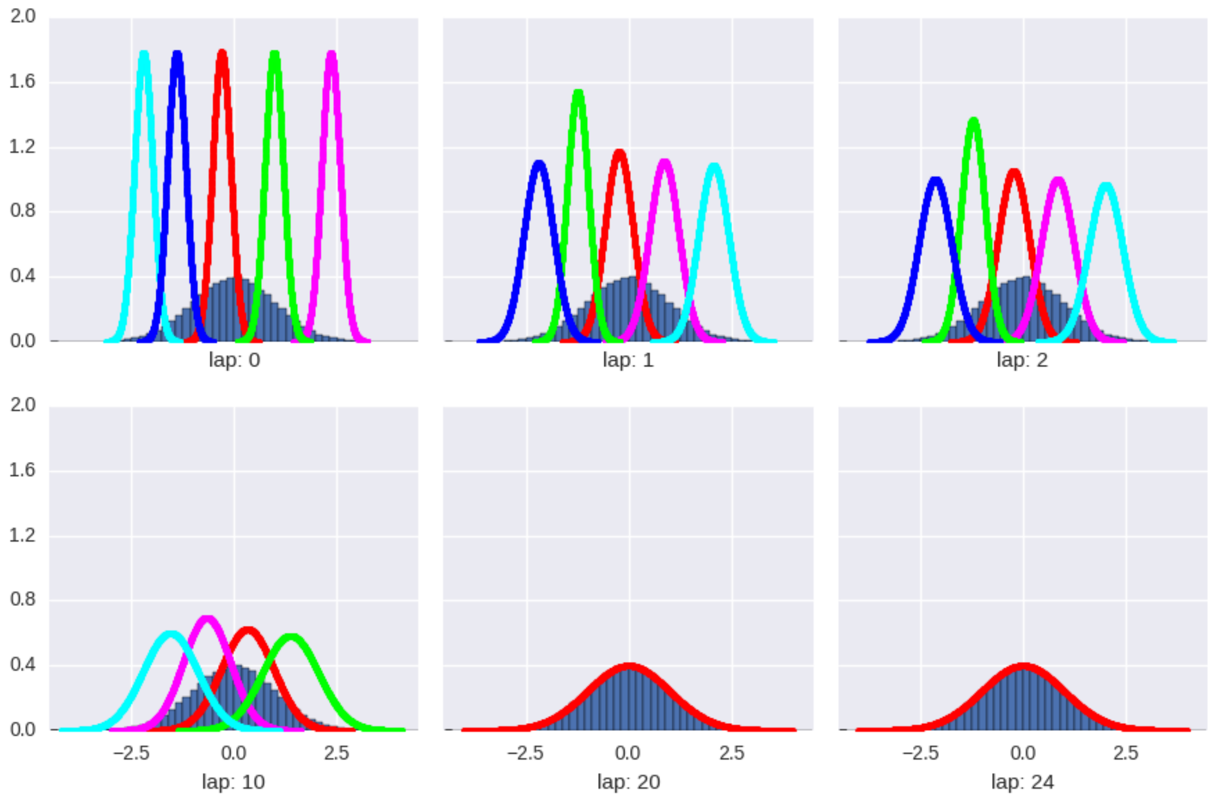
```
   83.000/100 after      9 sec. |    143.8 MiB | K    5 | loss  1.421412261e+00 | Ndiff
MERGE @ lap 84.00: No promising candidates, so no attempts.
   84.000/100 after      9 sec. |    143.8 MiB | K    5 | loss  1.421410063e+00 | Ndiff
MERGE @ lap 85.00 : 0/10 accepted. Ndiff 0.00. 0 skipped.
   85.000/100 after     10 sec. |    143.8 MiB | K    5 | loss  1.421407925e+00 | Ndiff
MERGE @ lap 86.00: No promising candidates, so no attempts.
   86.000/100 after     10 sec. |    143.8 MiB | K    5 | loss  1.421405845e+00 | Ndiff
MERGE @ lap 87.00: No promising candidates, so no attempts.
   87.000/100 after     10 sec. |    143.8 MiB | K    5 | loss  1.421403820e+00 | Ndiff
MERGE @ lap 88.00: No promising candidates, so no attempts.
   88.000/100 after     10 sec. |    143.8 MiB | K    5 | loss  1.421401849e+00 | Ndiff
MERGE @ lap 89.00: No promising candidates, so no attempts.
   89.000/100 after     10 sec. |    143.8 MiB | K    5 | loss  1.421399930e+00 | Ndiff
MERGE @ lap 90.00 : 0/10 accepted. Ndiff 0.00. 0 skipped.
   90.000/100 after     10 sec. |    143.8 MiB | K    5 | loss  1.421398060e+00 | Ndiff
MERGE @ lap 91.00: No promising candidates, so no attempts.
   91.000/100 after     10 sec. |    143.8 MiB | K    5 | loss  1.421396238e+00 | Ndiff
MERGE @ lap 92.00: No promising candidates, so no attempts.
   92.000/100 after     10 sec. |    143.8 MiB | K    5 | loss  1.421394462e+00 | Ndiff
MERGE @ lap 93.00: No promising candidates, so no attempts.
   93.000/100 after     11 sec. |    143.8 MiB | K    5 | loss  1.421392731e+00 | Ndiff
MERGE @ lap 94.00: No promising candidates, so no attempts.
   94.000/100 after     11 sec. |    143.8 MiB | K    5 | loss  1.421391043e+00 | Ndiff
MERGE @ lap 95.00 : 0/10 accepted. Ndiff 0.00. 0 skipped.
   95.000/100 after     11 sec. |    143.8 MiB | K    5 | loss  1.421389396e+00 | Ndiff
MERGE @ lap 96.00: No promising candidates, so no attempts.
   96.000/100 after     11 sec. |    143.8 MiB | K    5 | loss  1.421387789e+00 | Ndiff
MERGE @ lap 97.00: No promising candidates, so no attempts.
   97.000/100 after     11 sec. |    143.8 MiB | K    5 | loss  1.421386221e+00 | Ndiff
MERGE @ lap 98.00: No promising candidates, so no attempts.
   98.000/100 after     11 sec. |    143.8 MiB | K    5 | loss  1.421384690e+00 | Ndiff
MERGE @ lap 99.00: No promising candidates, so no attempts.
   99.000/100 after     11 sec. |    143.8 MiB | K    5 | loss  1.421383195e+00 | Ndiff
MERGE @ lap 100.00 : 0/10 accepted. Ndiff 0.00. 0 skipped.
  100.000/100 after     11 sec. |    143.8 MiB | K    5 | loss  1.421381735e+00 | Ndiff
... done. not converged. max laps thru data exceeded.
```

Run with *delete* moves, from K=5 initial clusters

More flexible delete moves *are* accepted.

```
d_trained_model, d_info_dict = bnpy.run(
    dataset, 'DPMixtureModel', 'Gauss', 'memoVB',
    output_path=('/tmp/StandardNormalK1/' +
        'trymoves-K=%d-gamma=%s-ECovMat=%s*eye-moves=delete,shuffle/' % (
            K, gamma, sF)),
    nLap=100, nTask=1, nBatch=1,
    gamma0=gamma, sF=sF, ECovMat='eye',
    K=K, initname='randexamplesbydist',
    moves='delete,shuffle',
    **dict(delete_kwargs))
```

```
show_clusters_over_time(d_info_dict['task_output_path'])
```



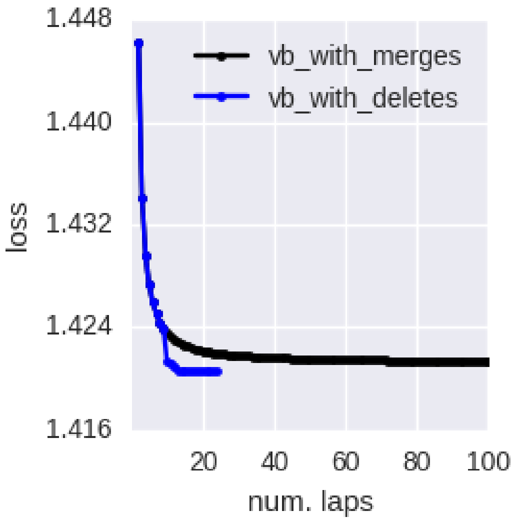
Out:

```
Dataset Summary:
X Data
  total size: 100000 units
  batch size: 100000 units
  num. batches: 1
Allocation Model:  DP mixture with K=0. Concentration gamma0= 5.00
Obs. Data  Model:  Gaussian with full covariance.
Obs. Data  Prior:  Gauss-Wishart on mean and covar of each cluster
  E[  mean[k] ] =
   [ 0.]
  E[ covar[k] ] =
  [[ 0.1]]
Initialization:
  initname = randexamplesbydist
  K = 5 (number of clusters)
  seed = 1607680
  elapsed_time: 0.0 sec
Learn Alg: memoVB | task  1/1 | alg. seed: 1607680 | data order seed: 8541952
task_output_path: /tmp/StandardNormalK1/trymoves-K=5-gamma=5.0-ECovMat=0.1*eye-moves=delete
DELETE @ lap 1.00: Disabled. Cannot delete before first complete lap, because SS that repre
    1.000/100 after      0 sec. |    151.4 MiB | K    5 | loss  1.547776525e+00 |
DELETE @ lap 2.00: Disabled. Waiting for lap >= 10 (--d_startLap).
    2.000/100 after      0 sec. |    151.4 MiB | K    5 | loss  1.446186110e+00 | Ndiff   25
```

```
DELETE @ lap 3.00: Disabled. Waiting for lap >= 10 (--d_startLap).
    3.000/100 after      0 sec. |    151.4 MiB | K    5 | loss  1.434177601e+00 | Ndiff  12
DELETE @ lap 4.00: Disabled. Waiting for lap >= 10 (--d_startLap).
    4.000/100 after      0 sec. |    151.4 MiB | K    5 | loss  1.429642157e+00 | Ndiff   8
DELETE @ lap 5.00: Disabled. Waiting for lap >= 10 (--d_startLap).
    5.000/100 after      0 sec. |    151.4 MiB | K    5 | loss  1.427329685e+00 | Ndiff   5
DELETE @ lap 6.00: Disabled. Waiting for lap >= 10 (--d_startLap).
    6.000/100 after      1 sec. |    151.4 MiB | K    5 | loss  1.425951747e+00 | Ndiff   4
DELETE @ lap 7.00: Disabled. Waiting for lap >= 10 (--d_startLap).
    7.000/100 after      1 sec. |    151.4 MiB | K    5 | loss  1.425047267e+00 | Ndiff   3
DELETE @ lap 8.00: Disabled. Waiting for lap >= 10 (--d_startLap).
    8.000/100 after      1 sec. |    151.4 MiB | K    5 | loss  1.424412975e+00 | Ndiff   2
DELETE @ lap 9.00: Disabled. Waiting for lap >= 10 (--d_startLap).
    9.000/100 after      1 sec. |    151.4 MiB | K    5 | loss  1.423946233e+00 | Ndiff   2
DELETE @ lap 10.00: 1/1 accepted. Ndiff 43526.74.
   10.000/100 after      3 sec. |    151.4 MiB | K    4 | loss  1.421389609e+00 | Ndiff   2
DELETE @ lap 11.00: 1/1 accepted. Ndiff 49884.65.
   11.000/100 after      5 sec. |    151.4 MiB | K    3 | loss  1.421157450e+00 | Ndiff   2
DELETE @ lap 12.00: 1/1 accepted. Ndiff 43419.31.
   12.000/100 after      6 sec. |    156.6 MiB | K    2 | loss  1.420888889e+00 | Ndiff   2
DELETE @ lap 13.00: 1/1 accepted. Ndiff 42527.97.
   13.000/100 after      6 sec. |    156.6 MiB | K    1 | loss  1.420559635e+00 | Ndiff   2
DELETE @ lap 14.00: Ineligible. Did not find >= 2 UIDs in entire model.
   14.000/100 after      6 sec. |    156.6 MiB | K    1 | loss  1.420559635e+00 | Ndiff
DELETE @ lap 15.00: Ineligible. Did not find >= 2 UIDs in entire model.
   15.000/100 after      6 sec. |    156.6 MiB | K    1 | loss  1.420559635e+00 | Ndiff
DELETE @ lap 16.00: Ineligible. Did not find >= 2 UIDs in entire model.
   16.000/100 after      6 sec. |    156.6 MiB | K    1 | loss  1.420559635e+00 | Ndiff
DELETE @ lap 17.00: Ineligible. Did not find >= 2 UIDs in entire model.
   17.000/100 after      6 sec. |    156.6 MiB | K    1 | loss  1.420559635e+00 | Ndiff
DELETE @ lap 18.00: Ineligible. Did not find >= 2 UIDs in entire model.
   18.000/100 after      6 sec. |    156.6 MiB | K    1 | loss  1.420559635e+00 | Ndiff
DELETE @ lap 19.00: Ineligible. Did not find >= 2 UIDs in entire model.
   19.000/100 after      6 sec. |    156.6 MiB | K    1 | loss  1.420559635e+00 | Ndiff
DELETE @ lap 20.00: Ineligible. Did not find >= 2 UIDs in entire model.
   20.000/100 after      6 sec. |    156.6 MiB | K    1 | loss  1.420559635e+00 | Ndiff
DELETE @ lap 21.00: Ineligible. Did not find >= 2 UIDs in entire model.
   21.000/100 after      6 sec. |    156.6 MiB | K    1 | loss  1.420559635e+00 | Ndiff
DELETE @ lap 22.00: Ineligible. Did not find >= 2 UIDs in entire model.
   22.000/100 after      6 sec. |    156.6 MiB | K    1 | loss  1.420559635e+00 | Ndiff
DELETE @ lap 23.00: Ineligible. Did not find >= 2 UIDs in entire model.
   23.000/100 after      6 sec. |    156.6 MiB | K    1 | loss  1.420559635e+00 | Ndiff
DELETE @ lap 24.00: Ineligible. Did not find >= 2 UIDs in entire model.
   24.000/100 after      6 sec. |    156.6 MiB | K    1 | loss  1.420559635e+00 | Ndiff
... done. converged.
```

Loss function trace plot

```
pylab.plot(
    m_info_dict['lap_history'][1:],
    m_info_dict['loss_history'][1:], 'k.-',
    label='vb_with_merges')
pylab.plot(
    d_info_dict['lap_history'][1:],
    d_info_dict['loss_history'][1:], 'b.-',
    label='vb_with_deletes')
pylab.legend(loc='upper right')
pylab.xlabel('num. laps')
pylab.ylabel('loss')
pylab.tight_layout()
```



Total running time of the script: (0 minutes 20.276 seconds)

`Download Python source code: plot-01-demo=deletes-model=dp_mix+gauss.py`

`Download Jupyter notebook: plot-01-demo=deletes-model=dp_mix+gauss.ipynb`

Generated by Sphinx-Gallery

5.2.3 Old faithful dataset

Variational training of DP mixture models on classic dataset on eruption times from Old Faithful geyser in Yellowstone National Park.

Variational with merge and delete proposals for DP mixtures of Gaussians

How delete moves can be more effective than merges.

In this example, we show how merge moves alone may not be enough to reliably escape local optima. Instead, we show that more flexible delete moves can escape from situations where merges alone fail.

```
import bnpy
import numpy as np
import os
```

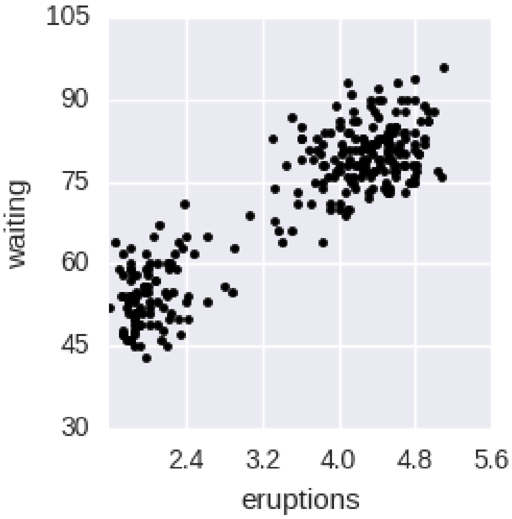
```
from matplotlib import pylab
import seaborn as sns

FIG_SIZE = (3, 3)
pylab.rcParams['figure.figsize'] = FIG_SIZE
```

Load dataset from file

```
dataset_path = os.path.join(bnpy.DATASET_PATH, 'faithful')
dataset = bnpy.data.XData.read_csv(
    os.path.join(dataset_path, 'faithful.csv'))
```

Make a simple plot of the raw data

```
pylab.plot(dataset.X[:, 0], dataset.X[:, 1], 'k.')
pylab.xlabel(dataset.column_names[0])
pylab.ylabel(dataset.column_names[1])
pylab.tight_layout()
data_ax_h = pylab.gca()
```



Setup: Determine specific settings of the proposals

```
merge_kwargs = dict(
    m_startLap=10,
    m_pair_ranking_procedure='total_size',
    m_pair_ranking_direction='descending',
    )

delete_kwargs = dict(
    d_startLap=20,
    d_nRefineSteps=50,
    )
```

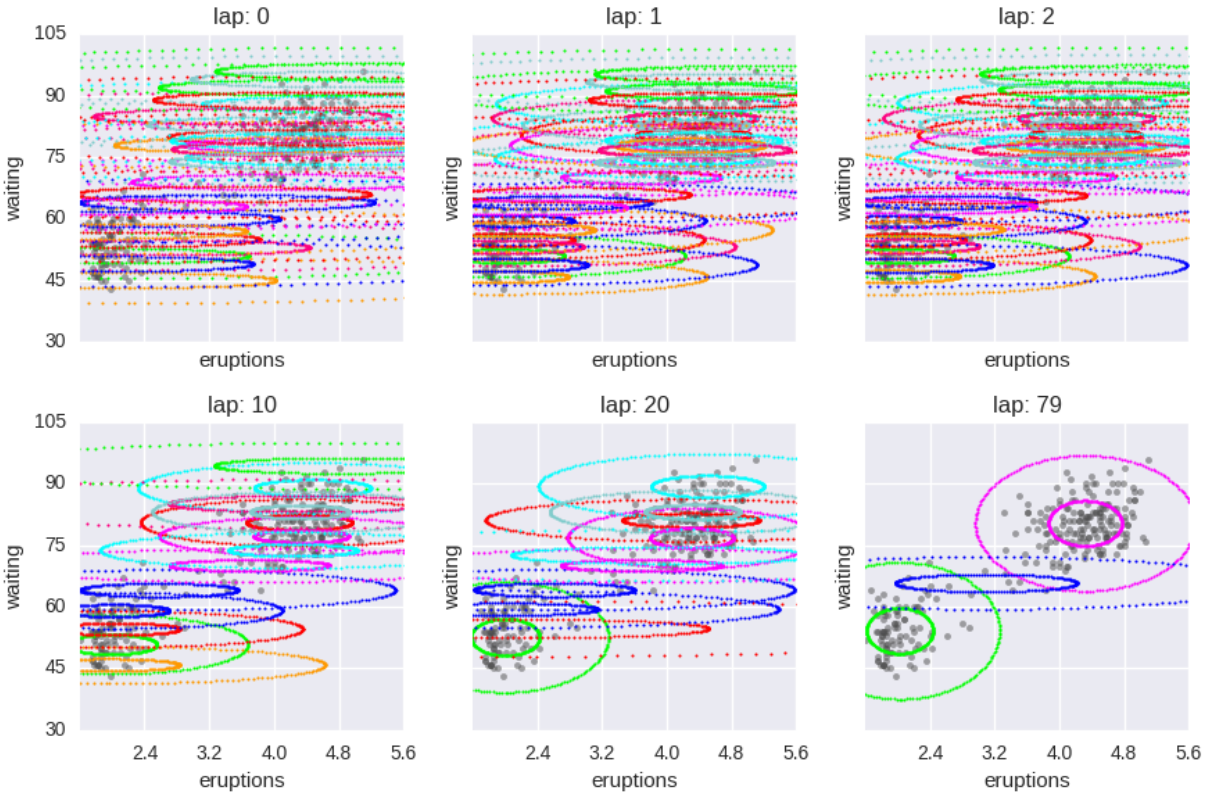
Setup: Helper function to display the learned clusters

```
def show_clusters_over_time(
        task_output_path=None,
        query_laps=[0, 1, 2, 10, 20, None],
        nrows=2):
    '''
    '''
    ncols = int(np.ceil(len(query_laps) // float(nrows)))
    fig_handle, ax_handle_list = pylab.subplots(
        figsize=(FIG_SIZE[0] * ncols, FIG_SIZE[1] * nrows),
        nrows=nrows, ncols=ncols, sharex=True, sharey=True)
    for plot_id, lap_val in enumerate(query_laps):
        cur_model, lap_val = bnpy.load_model_at_lap(task_output_path, lap_val)
        cur_ax_handle = ax_handle_list.flatten()[plot_id]
        bnpy.viz.PlotComps.plotCompsFromHModel(
            cur_model, dataset=dataset, ax_handle=cur_ax_handle)
        cur_ax_handle.set_title("lap: %d" % lap_val)
        cur_ax_handle.set_xlabel(dataset.column_names[0])
        cur_ax_handle.set_ylabel(dataset.column_names[1])
        cur_ax_handle.set_xlim(data_ax_h.get_xlim())
        cur_ax_handle.set_ylim(data_ax_h.get_ylim())
    pylab.tight_layout()
```

DiagGauss observation model, without moves

Start with too many clusters (K=25)

```
gamma = 5.0
sF = 5.0
K = 25

diag1_trained_model, diag1_info_dict = bnpy.run(
    dataset, 'DPMixtureModel', 'DiagGauss', 'memoVB',
    output_path=('/tmp/faithful/' +
        'trymoves-K=%d-gamma=%s-lik=DiagGauss-ECovMat=%s*eye-moves=none/' % (
            K, gamma, sF)),
    nLap=1000, nTask=1, nBatch=1, convergeThr=0.0001,
    gamma0=gamma, sF=sF, ECovMat='eye',
    K=K, initname='randexamplesbydist',
    )
show_clusters_over_time(diag1_info_dict['task_output_path'])
```



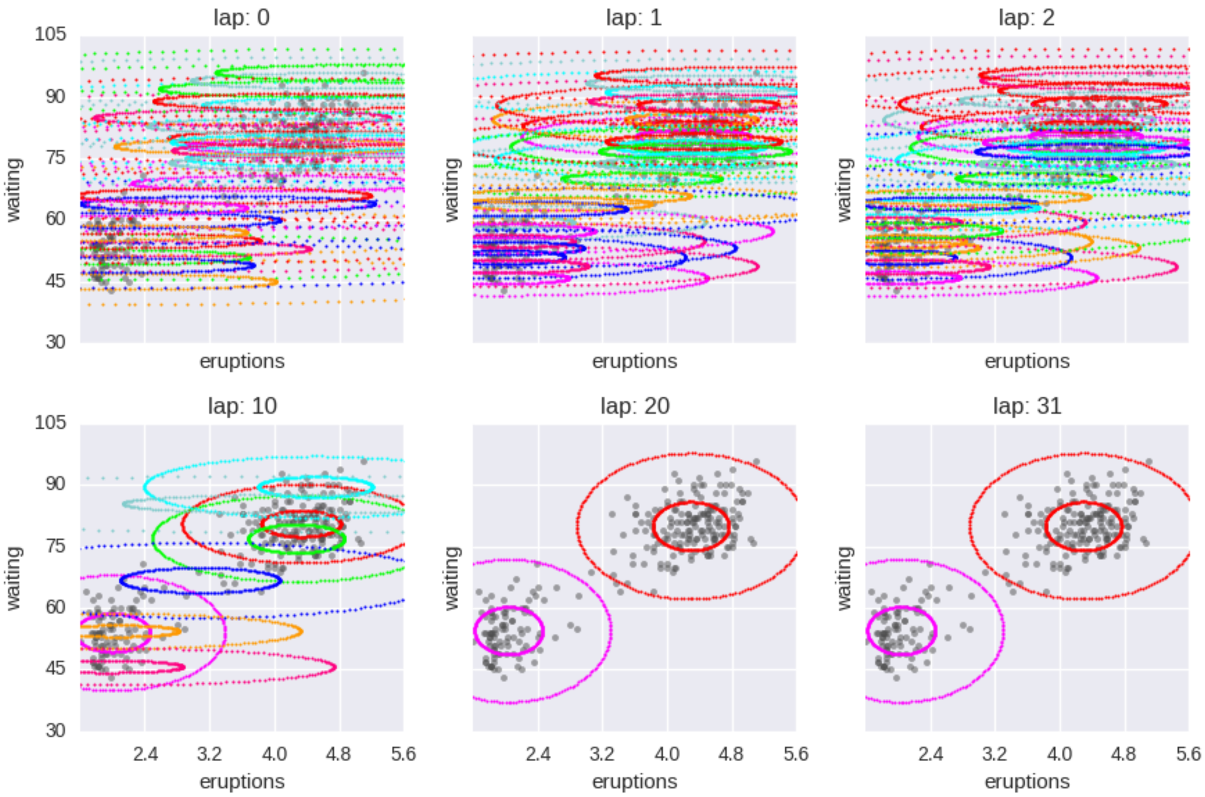
Out:

```
Dataset Summary:
X Data
  total size: 272 units
  batch size: 272 units
  num. batches: 1
Allocation Model:  DP mixture with K=0. Concentration gamma0= 5.00
Obs. Data  Model:  Gaussian with diagonal covariance.
Obs. Data  Prior:  independent Gauss-Wishart prior on each dimension
  Wishart params
    nu = 4
  beta = [ 10  10]
  Expectations
  E[  mean[k]] =
  [ 0  0]
  E[ covar[k]] =
  [[ 5.  0.]
   [ 0.  5.]]
Initialization:
  initname = randexamplesbydist
  K = 25 (number of clusters)
  seed = 1607680
  elapsed_time: 0.0 sec
Learn Alg: memoVB | task  1/1 | alg. seed: 1607680 | data order seed: 8541952
task_output_path: /tmp/faithful/trymoves-K=25-gamma=5.0-lik=DiagGauss-ECovMat=5.0*eye-moves
    1.000/1000 after      0 sec. |    153.9 MiB | K   25 | loss  3.062238769e+00 |
    2.000/1000 after      0 sec. |    153.9 MiB | K   25 | loss  3.029485836e+00 | Ndiff
    3.000/1000 after      0 sec. |    153.9 MiB | K   25 | loss  2.988929671e+00 | Ndiff
    4.000/1000 after      0 sec. |    153.9 MiB | K   25 | loss  2.934114089e+00 | Ndiff
    5.000/1000 after      0 sec. |    153.9 MiB | K   25 | loss  2.864320498e+00 | Ndiff
    6.000/1000 after      0 sec. |    153.9 MiB | K   25 | loss  2.796068518e+00 | Ndiff
    7.000/1000 after      0 sec. |    153.9 MiB | K   25 | loss  2.755200911e+00 | Ndiff
    8.000/1000 after      0 sec. |    153.9 MiB | K   25 | loss  2.724352077e+00 | Ndiff
    9.000/1000 after      0 sec. |    153.9 MiB | K   25 | loss  2.714818175e+00 | Ndiff
   10.000/1000 after      0 sec. |    153.9 MiB | K   25 | loss  2.710015995e+00 | Ndiff
   11.000/1000 after      0 sec. |    153.9 MiB | K   25 | loss  2.704686401e+00 | Ndiff
   12.000/1000 after      0 sec. |    153.9 MiB | K   25 | loss  2.698284253e+00 | Ndiff
   13.000/1000 after      0 sec. |    153.9 MiB | K   25 | loss  2.684190801e+00 | Ndiff
   14.000/1000 after      0 sec. |    153.9 MiB | K   25 | loss  2.673906693e+00 | Ndiff
   15.000/1000 after      0 sec. |    153.9 MiB | K   25 | loss  2.665323886e+00 | Ndiff
   16.000/1000 after      0 sec. |    153.9 MiB | K   25 | loss  2.654124178e+00 | Ndiff
   17.000/1000 after      0 sec. |    153.9 MiB | K   25 | loss  2.640063487e+00 | Ndiff
   18.000/1000 after      0 sec. |    153.9 MiB | K   25 | loss  2.617231862e+00 | Ndiff
   19.000/1000 after      0 sec. |    153.9 MiB | K   25 | loss  2.586775359e+00 | Ndiff
   20.000/1000 after      0 sec. |    153.9 MiB | K   25 | loss  2.569671629e+00 | Ndiff
   21.000/1000 after      0 sec. |    153.9 MiB | K   25 | loss  2.545014854e+00 | Ndiff
   22.000/1000 after      0 sec. |    153.9 MiB | K   25 | loss  2.525424540e+00 | Ndiff
   23.000/1000 after      0 sec. |    153.9 MiB | K   25 | loss  2.509251223e+00 | Ndiff
   24.000/1000 after      0 sec. |    153.9 MiB | K   25 | loss  2.493302635e+00 | Ndiff
   25.000/1000 after      0 sec. |    153.9 MiB | K   25 | loss  2.488984785e+00 | Ndiff
   26.000/1000 after      0 sec. |    153.9 MiB | K   25 | loss  2.484950693e+00 | Ndiff
   27.000/1000 after      0 sec. |    153.9 MiB | K   25 | loss  2.480608059e+00 | Ndiff
   28.000/1000 after      0 sec. |    153.9 MiB | K   25 | loss  2.467505075e+00 | Ndiff
   29.000/1000 after      0 sec. |    153.9 MiB | K   25 | loss  2.465130737e+00 | Ndiff
   30.000/1000 after      0 sec. |    153.9 MiB | K   25 | loss  2.463555210e+00 | Ndiff
   31.000/1000 after      0 sec. |    153.9 MiB | K   25 | loss  2.461714070e+00 | Ndiff
   32.000/1000 after      0 sec. |    153.9 MiB | K   25 | loss  2.459545432e+00 | Ndiff
   33.000/1000 after      0 sec. |    153.9 MiB | K   25 | loss  2.456958951e+00 | Ndiff
   34.000/1000 after      0 sec. |    153.9 MiB | K   25 | loss  2.453823064e+00 | Ndiff
   35.000/1000 after      0 sec. |    153.9 MiB | K   25 | loss  2.449961599e+00 | Ndiff
   36.000/1000 after      0 sec. |    153.9 MiB | K   25 | loss  2.445166898e+00 | Ndiff
   37.000/1000 after      0 sec. |    153.9 MiB | K   25 | loss  2.439248319e+00 | Ndiff
   38.000/1000 after      0 sec. |    153.9 MiB | K   25 | loss  2.432174168e+00 | Ndiff
   39.000/1000 after      0 sec. |    153.9 MiB | K   25 | loss  2.424387029e+00 | Ndiff
   40.000/1000 after      0 sec. |    153.9 MiB | K   25 | loss  2.416911819e+00 | Ndiff
   41.000/1000 after      0 sec. |    153.9 MiB | K   25 | loss  2.409264356e+00 | Ndiff
   42.000/1000 after      0 sec. |    153.9 MiB | K   25 | loss  2.392900067e+00 | Ndiff
   43.000/1000 after      0 sec. |    153.9 MiB | K   25 | loss  2.379189152e+00 | Ndiff
   44.000/1000 after      0 sec. |    153.9 MiB | K   25 | loss  2.375964695e+00 | Ndiff
   45.000/1000 after      0 sec. |    153.9 MiB | K   25 | loss  2.373574138e+00 | Ndiff
   46.000/1000 after      1 sec. |    153.9 MiB | K   25 | loss  2.371010870e+00 | Ndiff
   47.000/1000 after      1 sec. |    153.9 MiB | K   25 | loss  2.368474850e+00 | Ndiff
   48.000/1000 after      1 sec. |    153.9 MiB | K   25 | loss  2.366360106e+00 | Ndiff
   49.000/1000 after      1 sec. |    153.9 MiB | K   25 | loss  2.364898037e+00 | Ndiff
   50.000/1000 after      1 sec. |    153.9 MiB | K   25 | loss  2.363200338e+00 | Ndiff
   51.000/1000 after      1 sec. |    153.9 MiB | K   25 | loss  2.354385944e+00 | Ndiff
   52.000/1000 after      1 sec. |    153.9 MiB | K   25 | loss  2.350192787e+00 | Ndiff
   53.000/1000 after      1 sec. |    153.9 MiB | K   25 | loss  2.350192415e+00 | Ndiff
   54.000/1000 after      1 sec. |    153.9 MiB | K   25 | loss  2.350192256e+00 | Ndiff
   55.000/1000 after      1 sec. |    153.9 MiB | K   25 | loss  2.350192185e+00 | Ndiff
   56.000/1000 after      1 sec. |    153.9 MiB | K   25 | loss  2.350192152e+00 | Ndiff
   57.000/1000 after      1 sec. |    153.9 MiB | K   25 | loss  2.350192136e+00 | Ndiff
   58.000/1000 after      1 sec. |    153.9 MiB | K   25 | loss  2.350192129e+00 | Ndiff
   59.000/1000 after      1 sec. |    153.9 MiB | K   25 | loss  2.350192124e+00 | Ndiff
   60.000/1000 after      1 sec. |    153.9 MiB | K   25 | loss  2.350192122e+00 | Ndiff
   61.000/1000 after      1 sec. |    153.9 MiB | K   25 | loss  2.350192120e+00 | Ndiff
   62.000/1000 after      1 sec. |    153.9 MiB | K   25 | loss  2.350192119e+00 | Ndiff
   63.000/1000 after      1 sec. |    153.9 MiB | K   25 | loss  2.350192119e+00 | Ndiff
   64.000/1000 after      1 sec. |    153.9 MiB | K   25 | loss  2.350192118e+00 | Ndiff
   65.000/1000 after      1 sec. |    153.9 MiB | K   25 | loss  2.350192118e+00 | Ndiff
   66.000/1000 after      1 sec. |    153.9 MiB | K   25 | loss  2.350192117e+00 | Ndiff
   67.000/1000 after      1 sec. |    153.9 MiB | K   25 | loss  2.350192117e+00 | Ndiff
   68.000/1000 after      1 sec. |    153.9 MiB | K   25 | loss  2.350192117e+00 | Ndiff
   69.000/1000 after      1 sec. |    153.9 MiB | K   25 | loss  2.350192117e+00 | Ndiff
   70.000/1000 after      1 sec. |    153.9 MiB | K   25 | loss  2.350192117e+00 | Ndiff
   71.000/1000 after      1 sec. |    153.9 MiB | K   25 | loss  2.350192116e+00 | Ndiff
   72.000/1000 after      1 sec. |    153.9 MiB | K   25 | loss  2.350192116e+00 | Ndiff
   73.000/1000 after      1 sec. |    153.9 MiB | K   25 | loss  2.350192116e+00 | Ndiff
   74.000/1000 after      1 sec. |    153.9 MiB | K   25 | loss  2.350192116e+00 | Ndiff
   75.000/1000 after      1 sec. |    153.9 MiB | K   25 | loss  2.350192116e+00 | Ndiff
   76.000/1000 after      1 sec. |    153.9 MiB | K   25 | loss  2.350192116e+00 | Ndiff
   77.000/1000 after      1 sec. |    153.9 MiB | K   25 | loss  2.350192116e+00 | Ndiff
   78.000/1000 after      1 sec. |    153.9 MiB | K   25 | loss  2.350192116e+00 | Ndiff
   79.000/1000 after      1 sec. |    153.9 MiB | K   25 | loss  2.350192116e+00 | Ndiff
... done. converged.
SKIPPED 2 comps with size below 0.00
```

DiagGauss observation model

Start with too many clusters (K=25) Use merges and deletes to reduce to a better set.

```
gamma = 5.0
sF = 5.0
K = 25

diag_trained_model, diag_info_dict = bnpy.run(
    dataset, 'DPMixtureModel', 'DiagGauss', 'memoVB',
    output_path=('/tmp/faithful/' +
        'trymoves-K=%d-gamma=%s-lik=DiagGauss-ECovMat=%s*eye-moves=merge,delete,shuffle/'
            K, gamma, sF)),
    nLap=100, nTask=1, nBatch=1,
    gamma0=gamma, sF=sF, ECovMat='eye',
    K=K, initname='randexamplesbydist',
    moves='merge,delete,shuffle',
    **dict(delete_kwargs.items() + merge_kwargs.items()))

show_clusters_over_time(diag_info_dict['task_output_path'])
```



Out:

```
Dataset Summary:
X Data
  total size: 272 units
  batch size: 272 units
  num. batches: 1
Allocation Model:  DP mixture with K=0. Concentration gamma0= 5.00
Obs. Data  Model:  Gaussian with diagonal covariance.
Obs. Data  Prior:  independent Gauss-Wishart prior on each dimension
  Wishart params
    nu = 4
  beta = [ 10  10]
  Expectations
  E[  mean[k]] =
  [ 0  0]
  E[ covar[k]] =
  [[ 5.  0.]
   [ 0.  5.]]
Initialization:
  initname = randexamplesbydist
  K = 25 (number of clusters)
  seed = 1607680
  elapsed_time: 0.0 sec
Learn Alg: memoVB | task  1/1 | alg. seed: 1607680 | data order seed: 8541952
task_output_path: /tmp/faithful/trymoves-K=25-gamma=5.0-lik=DiagGauss-ECovMat=5.0*eye-moves
MERGE @ lap 1.00: Disabled. Cannot plan merge on first lap. Need valid SS that represent wh
```

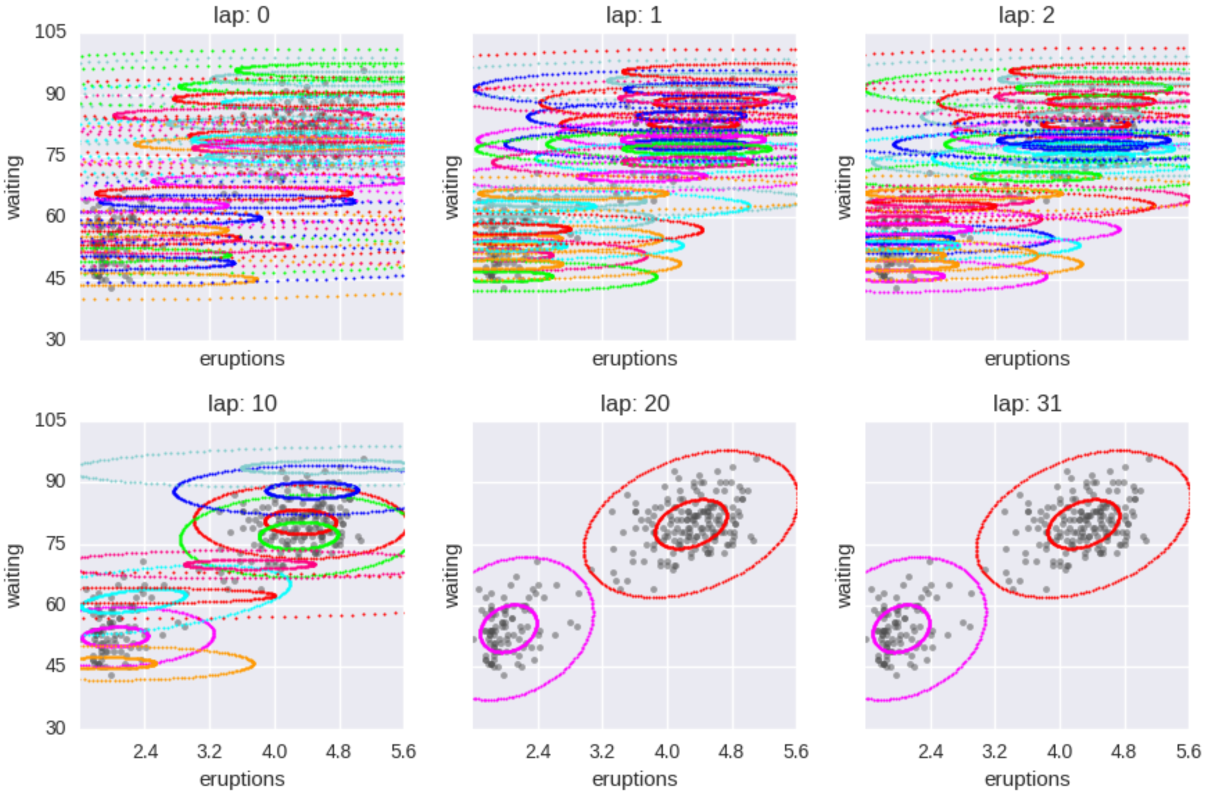
```
DELETE @ lap 1.00: Disabled. Cannot delete before first complete lap, because SS that repre
    1.000/100 after      0 sec. |    153.9 MiB | K   25 | loss  3.051961291e+00 |
MERGE @ lap 2.00: Disabled. Waiting for lap >= 10 (--m_startLap).
DELETE @ lap 2.00: Disabled. Waiting for lap >= 20 (--d_startLap).
    2.000/100 after      0 sec. |    153.9 MiB | K   25 | loss  3.022209522e+00 | Ndiff
MERGE @ lap 3.00: Disabled. Waiting for lap >= 10 (--m_startLap).
DELETE @ lap 3.00: Disabled. Waiting for lap >= 20 (--d_startLap).
    3.000/100 after      0 sec. |    153.9 MiB | K   25 | loss  2.984471596e+00 | Ndiff
MERGE @ lap 4.00: Disabled. Waiting for lap >= 10 (--m_startLap).
DELETE @ lap 4.00: Disabled. Waiting for lap >= 20 (--d_startLap).
    4.000/100 after      0 sec. |    153.9 MiB | K   25 | loss  2.926643058e+00 | Ndiff
MERGE @ lap 5.00: Disabled. Waiting for lap >= 10 (--m_startLap).
DELETE @ lap 5.00: Disabled. Waiting for lap >= 20 (--d_startLap).
    5.000/100 after      0 sec. |    153.9 MiB | K   25 | loss  2.862469660e+00 | Ndiff
MERGE @ lap 6.00: Disabled. Waiting for lap >= 10 (--m_startLap).
DELETE @ lap 6.00: Disabled. Waiting for lap >= 20 (--d_startLap).
    6.000/100 after      0 sec. |    153.9 MiB | K   25 | loss  2.779705996e+00 | Ndiff
MERGE @ lap 7.00: Disabled. Waiting for lap >= 10 (--m_startLap).
DELETE @ lap 7.00: Disabled. Waiting for lap >= 20 (--d_startLap).
    7.000/100 after      0 sec. |    153.9 MiB | K   25 | loss  2.734025386e+00 | Ndiff
MERGE @ lap 8.00: Disabled. Waiting for lap >= 10 (--m_startLap).
DELETE @ lap 8.00: Disabled. Waiting for lap >= 20 (--d_startLap).
    8.000/100 after      0 sec. |    153.9 MiB | K   25 | loss  2.705925412e+00 | Ndiff
MERGE @ lap 9.00: Disabled. Waiting for lap >= 10 (--m_startLap).
DELETE @ lap 9.00: Disabled. Waiting for lap >= 20 (--d_startLap).
    9.000/100 after      0 sec. |    153.9 MiB | K   25 | loss  2.695114165e+00 | Ndiff
DELETE @ lap 10.00: Disabled. Waiting for lap >= 20 (--d_startLap).
MERGE @ lap 10.00 : 11/21 accepted. Ndiff 96.55. 39 skipped.
   10.000/100 after      0 sec. |    153.9 MiB | K   14 | loss  2.588387948e+00 | Ndiff
DELETE @ lap 11.00: Disabled. Waiting for lap >= 20 (--d_startLap).
MERGE @ lap 11.00 : 6/10 accepted. Ndiff 57.31. 23 skipped.
   11.000/100 after      0 sec. |    153.9 MiB | K    8 | loss  2.418907387e+00 | Ndiff
DELETE @ lap 12.00: Disabled. Waiting for lap >= 20 (--d_startLap).
MERGE @ lap 12.00 : 4/5 accepted. Ndiff 24.39. 14 skipped.
   12.000/100 after      0 sec. |    153.9 MiB | K    4 | loss  2.344906523e+00 | Ndiff
DELETE @ lap 13.00: Disabled. Waiting for lap >= 20 (--d_startLap).
MERGE @ lap 13.00 : 1/4 accepted. Ndiff 0.00. 2 skipped.
   13.000/100 after      0 sec. |    153.9 MiB | K    3 | loss  2.342439735e+00 | Ndiff
MERGE @ lap 14.00: No promising candidates, so no attempts.
DELETE @ lap 14.00: Disabled. Waiting for lap >= 20 (--d_startLap).
   14.000/100 after      0 sec. |    153.9 MiB | K    3 | loss  2.341617749e+00 | Ndiff
DELETE @ lap 15.00: Disabled. Waiting for lap >= 20 (--d_startLap).
MERGE @ lap 15.00 : 0/2 accepted. Ndiff 0.00. 0 skipped.
   15.000/100 after      0 sec. |    153.9 MiB | K    3 | loss  2.341288596e+00 | Ndiff
DELETE @ lap 16.00: Disabled. Waiting for lap >= 20 (--d_startLap).
MERGE @ lap 16.00 : 0/1 accepted. Ndiff 0.00. 0 skipped.
   16.000/100 after      0 sec. |    153.9 MiB | K    3 | loss  2.341147153e+00 | Ndiff
DELETE @ lap 17.00: Disabled. Waiting for lap >= 20 (--d_startLap).
MERGE @ lap 17.00 : 0/2 accepted. Ndiff 0.00. 0 skipped.
   17.000/100 after      0 sec. |    153.9 MiB | K    3 | loss  2.341080073e+00 | Ndiff
MERGE @ lap 18.00: No promising candidates, so no attempts.
DELETE @ lap 18.00: Disabled. Waiting for lap >= 20 (--d_startLap).
   18.000/100 after      1 sec. |    153.9 MiB | K    3 | loss  2.341044964e+00 | Ndiff
```

```
DELETE @ lap 19.00: Disabled. Waiting for lap >= 20 (--d_startLap).
MERGE @ lap 19.00 : 0/2 accepted. Ndiff 0.00. 0 skipped.
   19.000/100 after      1 sec. |    153.9 MiB | K    3 | loss  2.341025070e+00 | Ndiff
MERGE @ lap 20.00: No promising candidates, so no attempts.
DELETE @ lap 20.00: 1/1 accepted. Ndiff 169.12.
   20.000/100 after      1 sec. |    153.9 MiB | K    2 | loss  2.330925760e+00 | Ndiff
DELETE @ lap 21.00: 0/1 accepted. Ndiff 0.00.
MERGE @ lap 21.00 : 0/1 accepted. Ndiff 0.00. 0 skipped.
   21.000/100 after      1 sec. |    153.9 MiB | K    2 | loss  2.330925543e+00 | Ndiff
MERGE @ lap 22.00: No promising candidates, so no attempts.
DELETE @ lap 22.00: 0/1 accepted. Ndiff 0.00.
   22.000/100 after      1 sec. |    153.9 MiB | K    2 | loss  2.330925529e+00 | Ndiff
MERGE @ lap 23.00: No promising candidates, so no attempts.
DELETE @ lap 23.00: Empty plan. 0 UIDs eligible as delete target. 0 too busy with other mov
   23.000/100 after      1 sec. |    153.9 MiB | K    2 | loss  2.330925528e+00 | Ndiff
MERGE @ lap 24.00: No promising candidates, so no attempts.
DELETE @ lap 24.00: Empty plan. 0 UIDs eligible as delete target. 0 too busy with other mov
   24.000/100 after      1 sec. |    153.9 MiB | K    2 | loss  2.330925528e+00 | Ndiff
MERGE @ lap 25.00: No promising candidates, so no attempts.
DELETE @ lap 25.00: Empty plan. 0 UIDs eligible as delete target. 0 too busy with other mov
   25.000/100 after      1 sec. |    153.9 MiB | K    2 | loss  2.330925528e+00 | Ndiff
DELETE @ lap 26.00: Empty plan. 0 UIDs eligible as delete target. 0 too busy with other mov
MERGE @ lap 26.00 : 0/1 accepted. Ndiff 0.00. 0 skipped.
   26.000/100 after      1 sec. |    153.9 MiB | K    2 | loss  2.330925528e+00 | Ndiff
MERGE @ lap 27.00: No promising candidates, so no attempts.
DELETE @ lap 27.00: 0/1 accepted. Ndiff 0.00.
   27.000/100 after      1 sec. |    153.9 MiB | K    2 | loss  2.330925528e+00 | Ndiff
MERGE @ lap 28.00: No promising candidates, so no attempts.
DELETE @ lap 28.00: 0/1 accepted. Ndiff 0.00.
   28.000/100 after      1 sec. |    153.9 MiB | K    2 | loss  2.330925528e+00 | Ndiff
MERGE @ lap 29.00: No promising candidates, so no attempts.
DELETE @ lap 29.00: Empty plan. 0 UIDs eligible as delete target. 0 too busy with other mov
   29.000/100 after      1 sec. |    153.9 MiB | K    2 | loss  2.330925528e+00 | Ndiff
MERGE @ lap 30.00: No promising candidates, so no attempts.
DELETE @ lap 30.00: Empty plan. 0 UIDs eligible as delete target. 0 too busy with other mov
   30.000/100 after      1 sec. |    153.9 MiB | K    2 | loss  2.330925528e+00 | Ndiff
DELETE @ lap 31.00: Empty plan. 0 UIDs eligible as delete target. 0 too busy with other mov
MERGE @ lap 31.00 : 0/1 accepted. Ndiff 0.00. 0 skipped.
   31.000/100 after      1 sec. |    153.9 MiB | K    2 | loss  2.330925528e+00 | Ndiff
... done. converged.
```

Gauss observation model

Start with too many clusters (K=25) Use merges and deletes to reduce to a better set.

```
full_trained_model, full_info_dict = bnpy.run(
    dataset, 'DPMixtureModel', 'Gauss', 'memoVB',
    output_path=('/tmp/faithful/' +
        'trymoves-K=%d-gamma=%s-lik-Gauss-ECovMat=%s*eye-moves=merge,delete,shuffle/' % (
            K, gamma, sF)),
    nLap=100, nTask=1, nBatch=1,
    gamma0=gamma, sF=sF, ECovMat='eye',
    K=K, initname='randexamplesbydist',
    moves='merge,delete,shuffle',
    **dict(delete_kwargs.items() + merge_kwargs.items()))

show_clusters_over_time(full_info_dict['task_output_path'])
```



Out:

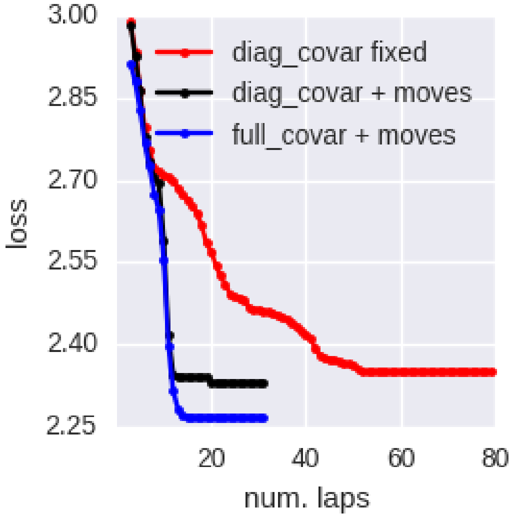
```
Dataset Summary:
X Data
  total size: 272 units
  batch size: 272 units
  num. batches: 1
Allocation Model:  DP mixture with K=0. Concentration gamma0= 5.00
Obs. Data  Model:  Gaussian with full covariance.
Obs. Data  Prior:  Gauss-Wishart on mean and covar of each cluster
  E[  mean[k] ] =
   [ 0.  0.]
  E[ covar[k] ] =
  [[ 5.  0.]
   [ 0.  5.]]
Initialization:
  initname = randexamplesbydist
  K = 25 (number of clusters)
  seed = 1607680
  elapsed_time: 0.0 sec
Learn Alg: memoVB | task  1/1 | alg. seed: 1607680 | data order seed: 8541952
```

```
task_output_path: /tmp/faithful/trymoves-K=25-gamma=5.0-lik-Gauss-ECovMat=5.0*eye-moves=mer
MERGE @ lap 1.00: Disabled. Cannot plan merge on first lap. Need valid SS that represent wh
DELETE @ lap 1.00: Disabled. Cannot delete before first complete lap, because SS that repre
    1.000/100 after      0 sec. |    155.5 MiB | K   25 | loss  2.959097158e+00 |
MERGE @ lap 2.00: Disabled. Waiting for lap >= 10 (--m_startLap).
DELETE @ lap 2.00: Disabled. Waiting for lap >= 20 (--d_startLap).
    2.000/100 after      0 sec. |    155.5 MiB | K   25 | loss  2.937002302e+00 | Ndiff
MERGE @ lap 3.00: Disabled. Waiting for lap >= 10 (--m_startLap).
DELETE @ lap 3.00: Disabled. Waiting for lap >= 20 (--d_startLap).
    3.000/100 after      0 sec. |    155.5 MiB | K   25 | loss  2.912622033e+00 | Ndiff
MERGE @ lap 4.00: Disabled. Waiting for lap >= 10 (--m_startLap).
DELETE @ lap 4.00: Disabled. Waiting for lap >= 20 (--d_startLap).
    4.000/100 after      0 sec. |    155.5 MiB | K   25 | loss  2.880654959e+00 | Ndiff
MERGE @ lap 5.00: Disabled. Waiting for lap >= 10 (--m_startLap).
DELETE @ lap 5.00: Disabled. Waiting for lap >= 20 (--d_startLap).
    5.000/100 after      0 sec. |    155.5 MiB | K   25 | loss  2.829532053e+00 | Ndiff
MERGE @ lap 6.00: Disabled. Waiting for lap >= 10 (--m_startLap).
DELETE @ lap 6.00: Disabled. Waiting for lap >= 20 (--d_startLap).
    6.000/100 after      0 sec. |    155.5 MiB | K   25 | loss  2.769025659e+00 | Ndiff
MERGE @ lap 7.00: Disabled. Waiting for lap >= 10 (--m_startLap).
DELETE @ lap 7.00: Disabled. Waiting for lap >= 20 (--d_startLap).
    7.000/100 after      0 sec. |    155.5 MiB | K   25 | loss  2.725273863e+00 | Ndiff
MERGE @ lap 8.00: Disabled. Waiting for lap >= 10 (--m_startLap).
DELETE @ lap 8.00: Disabled. Waiting for lap >= 20 (--d_startLap).
    8.000/100 after      0 sec. |    155.5 MiB | K   25 | loss  2.675946971e+00 | Ndiff
MERGE @ lap 9.00: Disabled. Waiting for lap >= 10 (--m_startLap).
DELETE @ lap 9.00: Disabled. Waiting for lap >= 20 (--d_startLap).
    9.000/100 after      0 sec. |    155.5 MiB | K   25 | loss  2.647374823e+00 | Ndiff
DELETE @ lap 10.00: Disabled. Waiting for lap >= 20 (--d_startLap).
MERGE @ lap 10.00 : 11/18 accepted. Ndiff 107.08. 41 skipped.
   10.000/100 after      1 sec. |    155.5 MiB | K   14 | loss  2.556490112e+00 | Ndiff
DELETE @ lap 11.00: Disabled. Waiting for lap >= 20 (--d_startLap).
MERGE @ lap 11.00 : 6/7 accepted. Ndiff 88.56. 27 skipped.
   11.000/100 after      1 sec. |    155.6 MiB | K    8 | loss  2.396890018e+00 | Ndiff
DELETE @ lap 12.00: Disabled. Waiting for lap >= 20 (--d_startLap).
MERGE @ lap 12.00 : 4/5 accepted. Ndiff 35.75. 11 skipped.
   12.000/100 after      1 sec. |    155.6 MiB | K    4 | loss  2.315013331e+00 | Ndiff
DELETE @ lap 13.00: Disabled. Waiting for lap >= 20 (--d_startLap).
MERGE @ lap 13.00 : 2/3 accepted. Ndiff 6.24. 3 skipped.
   13.000/100 after      1 sec. |    155.6 MiB | K    2 | loss  2.282002142e+00 | Ndiff
MERGE @ lap 14.00: No promising candidates, so no attempts.
DELETE @ lap 14.00: Disabled. Waiting for lap >= 20 (--d_startLap).
   14.000/100 after      1 sec. |    155.6 MiB | K    2 | loss  2.268874488e+00 | Ndiff
DELETE @ lap 15.00: Disabled. Waiting for lap >= 20 (--d_startLap).
MERGE @ lap 15.00 : 0/1 accepted. Ndiff 0.00. 0 skipped.
   15.000/100 after      1 sec. |    155.6 MiB | K    2 | loss  2.268112275e+00 | Ndiff
MERGE @ lap 16.00: No promising candidates, so no attempts.
DELETE @ lap 16.00: Disabled. Waiting for lap >= 20 (--d_startLap).
   16.000/100 after      1 sec. |    155.6 MiB | K    2 | loss  2.268077838e+00 | Ndiff
MERGE @ lap 17.00: No promising candidates, so no attempts.
DELETE @ lap 17.00: Disabled. Waiting for lap >= 20 (--d_startLap).
   17.000/100 after      1 sec. |    155.6 MiB | K    2 | loss  2.268076838e+00 | Ndiff
MERGE @ lap 18.00: No promising candidates, so no attempts.
```

```
DELETE @ lap 18.00: Disabled. Waiting for lap >= 20 (--d_startLap).
   18.000/100 after      1 sec. |    155.6 MiB | K    2 | loss  2.268076812e+00 | Ndiff
MERGE @ lap 19.00: No promising candidates, so no attempts.
DELETE @ lap 19.00: Disabled. Waiting for lap >= 20 (--d_startLap).
   19.000/100 after      1 sec. |    155.6 MiB | K    2 | loss  2.268076811e+00 | Ndiff
DELETE @ lap 20.00: 0/1 accepted. Ndiff 0.00.
MERGE @ lap 20.00 : 0/1 accepted. Ndiff 0.00. 0 skipped.
   20.000/100 after      1 sec. |    155.6 MiB | K    2 | loss  2.268076811e+00 | Ndiff
MERGE @ lap 21.00: No promising candidates, so no attempts.
DELETE @ lap 21.00: 0/1 accepted. Ndiff 0.00.
   21.000/100 after      1 sec. |    155.6 MiB | K    2 | loss  2.268076811e+00 | Ndiff
MERGE @ lap 22.00: No promising candidates, so no attempts.
DELETE @ lap 22.00: Empty plan. 0 UIDs eligible as delete target. 0 too busy with other mo
   22.000/100 after      1 sec. |    155.6 MiB | K    2 | loss  2.268076811e+00 | Ndiff
MERGE @ lap 23.00: No promising candidates, so no attempts.
DELETE @ lap 23.00: Empty plan. 0 UIDs eligible as delete target. 0 too busy with other mo
   23.000/100 after      1 sec. |    155.6 MiB | K    2 | loss  2.268076811e+00 | Ndiff
MERGE @ lap 24.00: No promising candidates, so no attempts.
DELETE @ lap 24.00: Empty plan. 0 UIDs eligible as delete target. 0 too busy with other mo
   24.000/100 after      1 sec. |    155.6 MiB | K    2 | loss  2.268076811e+00 | Ndiff
DELETE @ lap 25.00: Empty plan. 0 UIDs eligible as delete target. 0 too busy with other mo
MERGE @ lap 25.00 : 0/1 accepted. Ndiff 0.00. 0 skipped.
   25.000/100 after      1 sec. |    155.6 MiB | K    2 | loss  2.268076811e+00 | Ndiff
MERGE @ lap 26.00: No promising candidates, so no attempts.
DELETE @ lap 26.00: 0/1 accepted. Ndiff 0.00.
   26.000/100 after      1 sec. |    155.6 MiB | K    2 | loss  2.268076811e+00 | Ndiff
MERGE @ lap 27.00: No promising candidates, so no attempts.
DELETE @ lap 27.00: 0/1 accepted. Ndiff 0.00.
   27.000/100 after      1 sec. |    155.6 MiB | K    2 | loss  2.268076811e+00 | Ndiff
MERGE @ lap 28.00: No promising candidates, so no attempts.
DELETE @ lap 28.00: Empty plan. 0 UIDs eligible as delete target. 0 too busy with other mo
   28.000/100 after      1 sec. |    155.6 MiB | K    2 | loss  2.268076811e+00 | Ndiff
MERGE @ lap 29.00: No promising candidates, so no attempts.
DELETE @ lap 29.00: Empty plan. 0 UIDs eligible as delete target. 0 too busy with other mo
   29.000/100 after      1 sec. |    155.6 MiB | K    2 | loss  2.268076811e+00 | Ndiff
DELETE @ lap 30.00: Empty plan. 0 UIDs eligible as delete target. 0 too busy with other mo
MERGE @ lap 30.00 : 0/1 accepted. Ndiff 0.00. 0 skipped.
   30.000/100 after      1 sec. |    155.6 MiB | K    2 | loss  2.268076811e+00 | Ndiff
MERGE @ lap 31.00: No promising candidates, so no attempts.
DELETE @ lap 31.00: Empty plan. 0 UIDs eligible as delete target. 0 too busy with other mo
   31.000/100 after      1 sec. |    155.6 MiB | K    2 | loss  2.268076811e+00 | Ndiff
... done. converged.
```

Loss function trace plot

```
pylab.figure()
pylab.plot(
    diag1_info_dict['lap_history'][2:],
    diag1_info_dict['loss_history'][2:], 'r.-',
    label='diag_covar fixed')
pylab.plot(
    diag_info_dict['lap_history'][2:],
    diag_info_dict['loss_history'][2:], 'k.-',
    label='diag_covar + moves')
pylab.plot(
    full_info_dict['lap_history'][2:],
    full_info_dict['loss_history'][2:], 'b.-',
    label='full_covar + moves')
pylab.legend(loc='upper right')
pylab.xlabel('num. laps')
pylab.ylabel('loss')
pylab.tight_layout()
```



Total running time of the script: (0 minutes 5.577 seconds)

Download Python source code: plot-02-demo=merge_and_delete-model=dp_mix+gauss.py

Download Jupyter notebook: plot-02-demo=merge_and_delete-model=dp_mix+gauss.ipynb

Generated by Sphinx-Gallery

Variational training for Mixtures of Gaussians

Showcase of different models and algorithms applied to same dataset.

In this example, we show how bnpy makes it easy to apply different models and algorithms to the same dataset.

```
import bnpy
import numpy as np
import os

from matplotlib import pylab
import seaborn as sns

SMALL_FIG_SIZE = (2.5, 2.5)
```

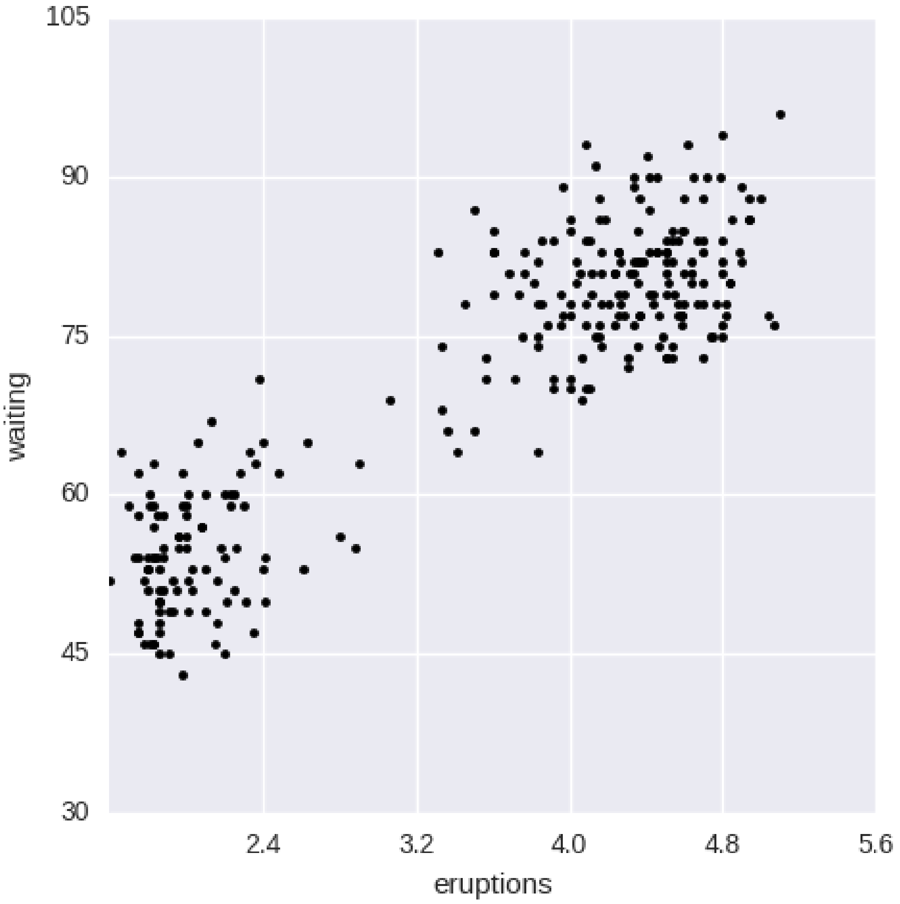
```
FIG_SIZE = (5, 5)
pylab.rcParams['figure.figsize'] = FIG_SIZE
```

Load dataset from file

```
dataset_path = os.path.join(bnpy.DATASET_PATH, 'faithful')
dataset = bnpy.data.XData.read_csv(
    os.path.join(dataset_path, 'faithful.csv'))
```

Make a simple plot of the raw data

```
pylab.plot(dataset.X[:, 0], dataset.X[:, 1], 'k.')
pylab.xlabel(dataset.column_names[0])
pylab.ylabel(dataset.column_names[1])
pylab.tight_layout()
data_ax_h = pylab.gca()
```



Setup: Helper function to display the learned clusters

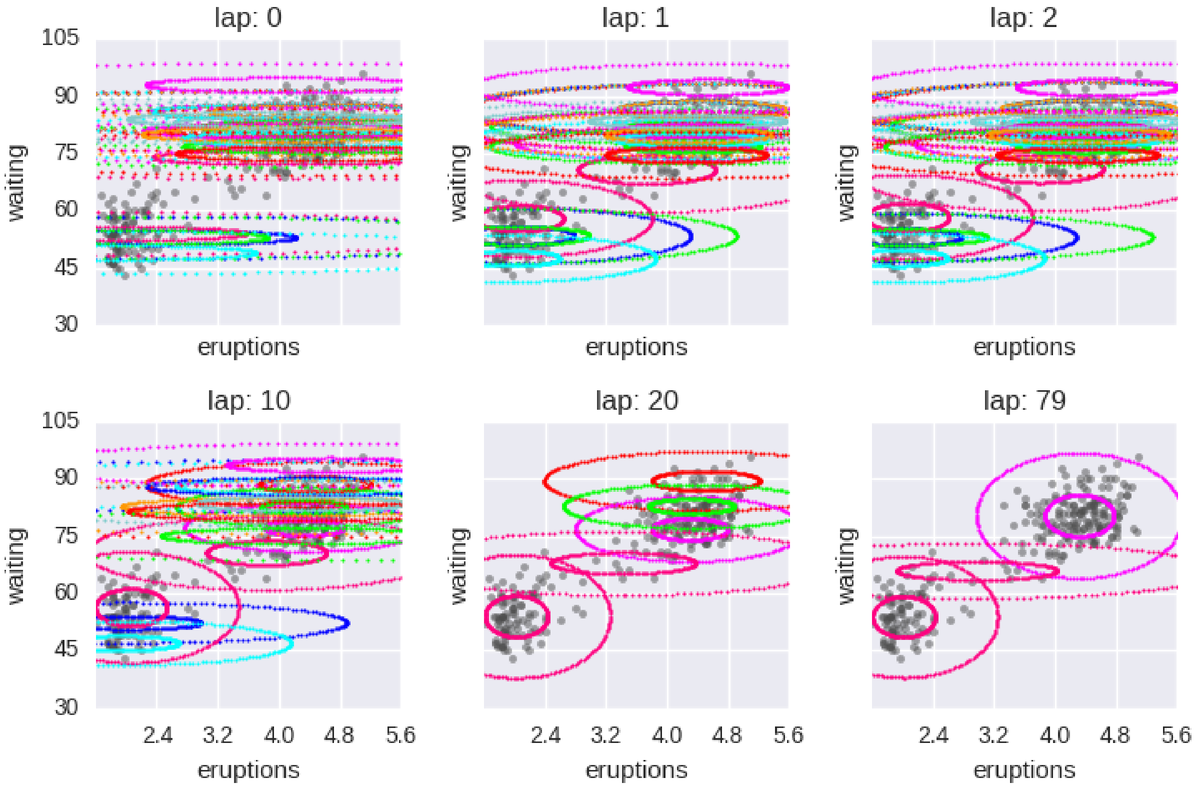
```
def show_clusters_over_time(
        task_output_path=None,
        query_laps=[0, 1, 2, 10, 20, None],
        nrows=2):
    ''' Show 2D elliptical contours overlaid on raw data.
    '''
    ncols = int(np.ceil(len(query_laps) // float(nrows)))
    fig_handle, ax_handle_list = pylab.subplots(
        figsize=(SMALL_FIG_SIZE[0] * ncols, SMALL_FIG_SIZE[1] * nrows),
        nrows=nrows, ncols=ncols, sharex=True, sharey=True)
    for plot_id, lap_val in enumerate(query_laps):
        cur_model, lap_val = bnpy.load_model_at_lap(task_output_path, lap_val)
        cur_ax_handle = ax_handle_list.flatten()[plot_id]
        bnpy.viz.PlotComps.plotCompsFromHModel(
            cur_model, dataset=dataset, ax_handle=cur_ax_handle)
        cur_ax_handle.set_title("lap: %d" % lap_val)
        cur_ax_handle.set_xlabel(dataset.column_names[0])
        cur_ax_handle.set_ylabel(dataset.column_names[1])
        cur_ax_handle.set_xlim(data_ax_h.get_xlim())
        cur_ax_handle.set_ylim(data_ax_h.get_ylim())
    pylab.tight_layout()
```

DiagGauss observation model

Assume diagonal covariances.

Start with too many clusters (K=20)

```
gamma = 5.0
sF = 5.0
K = 20

diag_trained_model, diag_info_dict = bnpy.run(
    dataset, 'DPMixtureModel', 'DiagGauss', 'memoVB',
    output_path='/tmp/faithful/showcase-K=20-lik=DiagGauss-ECovMat=5*eye/',
    nLap=1000, nTask=1, nBatch=1, convergeThr=0.0001,
    gamma0=gamma, sF=sF, ECovMat='eye',
    K=K, initname='randexamples',
    )
show_clusters_over_time(diag_info_dict['task_output_path'])
```



Out:

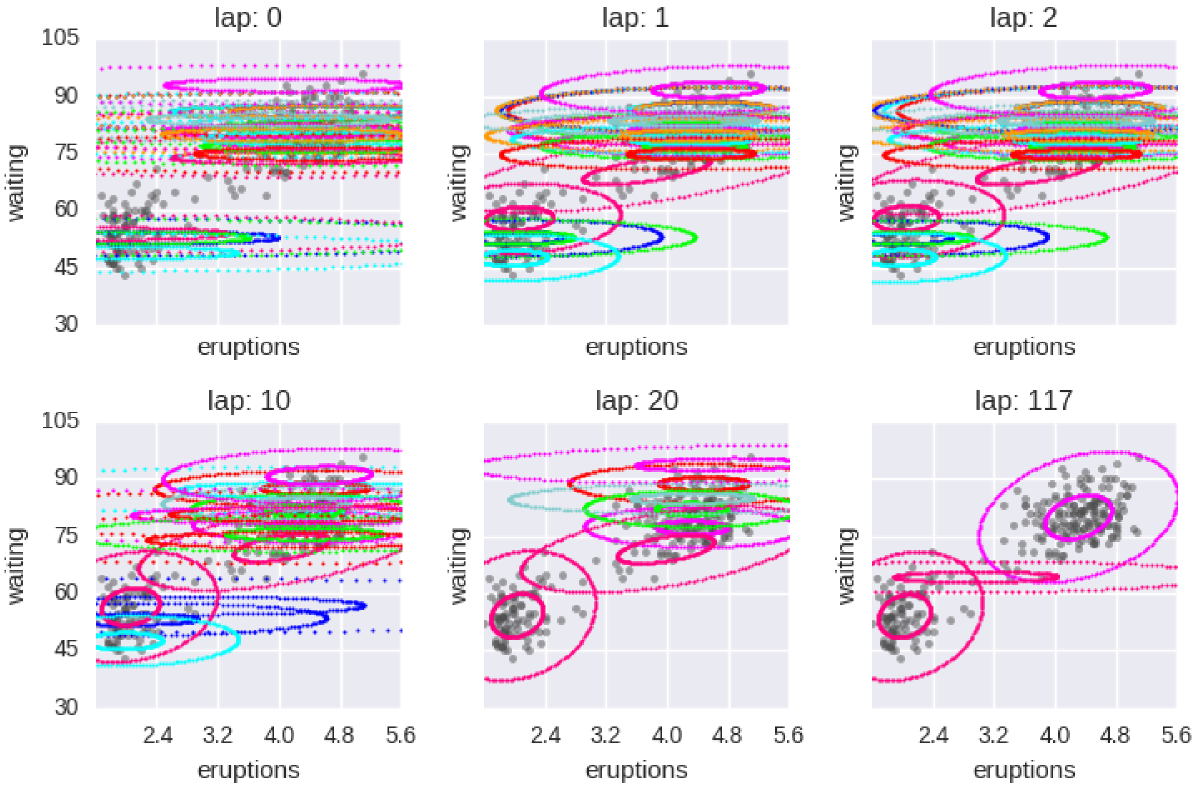
```
Dataset Summary:
X Data
  total size: 272 units
  batch size: 272 units
  num. batches: 1
Allocation Model:  DP mixture with K=0. Concentration gamma0= 5.00
Obs. Data  Model:  Gaussian with diagonal covariance.
Obs. Data  Prior:  independent Gauss-Wishart prior on each dimension
  Wishart params
    nu = 4
  beta = [ 10  10]
  Expectations
  E[  mean[k]] =
  [ 0  0]
  E[ covar[k]] =
  [[ 5.  0.]
   [ 0.  5.]]
Initialization:
  initname = randexamples
  K = 20 (number of clusters)
  seed = 1607680
  elapsed_time: 0.0 sec
Learn Alg: memoVB | task  1/1 | alg. seed: 1607680 | data order seed: 8541952
task_output_path: /tmp/faithful/showcase-K=20-lik=DiagGauss-ECovMat=5*eye/1
    1.000/1000 after      0 sec. |    158.6 MiB | K   20 | loss  3.002088161e+00 |
    2.000/1000 after      0 sec. |    158.6 MiB | K   20 | loss  2.966309748e+00 | Ndiff
    3.000/1000 after      0 sec. |    158.6 MiB | K   20 | loss  2.949498401e+00 | Ndiff
    4.000/1000 after      0 sec. |    158.6 MiB | K   20 | loss  2.931792100e+00 | Ndiff
    5.000/1000 after      0 sec. |    158.6 MiB | K   20 | loss  2.910174417e+00 | Ndiff
    6.000/1000 after      0 sec. |    158.6 MiB | K   20 | loss  2.881973764e+00 | Ndiff
    7.000/1000 after      0 sec. |    158.6 MiB | K   20 | loss  2.833491488e+00 | Ndiff
    8.000/1000 after      0 sec. |    158.6 MiB | K   20 | loss  2.793876249e+00 | Ndiff
    9.000/1000 after      0 sec. |    158.6 MiB | K   20 | loss  2.752249036e+00 | Ndiff
   10.000/1000 after      0 sec. |    158.6 MiB | K   20 | loss  2.698456106e+00 | Ndiff
   11.000/1000 after      0 sec. |    158.6 MiB | K   20 | loss  2.651622886e+00 | Ndiff
   12.000/1000 after      0 sec. |    158.6 MiB | K   20 | loss  2.624108159e+00 | Ndiff
   13.000/1000 after      0 sec. |    158.6 MiB | K   20 | loss  2.587572334e+00 | Ndiff
   14.000/1000 after      0 sec. |    158.6 MiB | K   20 | loss  2.560219008e+00 | Ndiff
   15.000/1000 after      0 sec. |    158.6 MiB | K   20 | loss  2.544887510e+00 | Ndiff
   16.000/1000 after      0 sec. |    158.6 MiB | K   20 | loss  2.527562170e+00 | Ndiff
   17.000/1000 after      0 sec. |    158.6 MiB | K   20 | loss  2.513204290e+00 | Ndiff
   18.000/1000 after      0 sec. |    158.6 MiB | K   20 | loss  2.507574530e+00 | Ndiff
   19.000/1000 after      0 sec. |    158.6 MiB | K   20 | loss  2.497639081e+00 | Ndiff
   20.000/1000 after      0 sec. |    158.6 MiB | K   20 | loss  2.497002974e+00 | Ndiff
   21.000/1000 after      0 sec. |    158.6 MiB | K   20 | loss  2.496368366e+00 | Ndiff
   22.000/1000 after      0 sec. |    158.6 MiB | K   20 | loss  2.495732569e+00 | Ndiff
   23.000/1000 after      0 sec. |    158.6 MiB | K   20 | loss  2.495103986e+00 | Ndiff
   24.000/1000 after      0 sec. |    158.6 MiB | K   20 | loss  2.494497370e+00 | Ndiff
   25.000/1000 after      0 sec. |    158.6 MiB | K   20 | loss  2.493928089e+00 | Ndiff
   26.000/1000 after      0 sec. |    158.6 MiB | K   20 | loss  2.493406054e+00 | Ndiff
   27.000/1000 after      0 sec. |    158.6 MiB | K   20 | loss  2.492931650e+00 | Ndiff
   28.000/1000 after      0 sec. |    158.6 MiB | K   20 | loss  2.492495409e+00 | Ndiff
   29.000/1000 after      0 sec. |    158.6 MiB | K   20 | loss  2.492081031e+00 | Ndiff
   30.000/1000 after      0 sec. |    158.6 MiB | K   20 | loss  2.491669505e+00 | Ndiff
   31.000/1000 after      0 sec. |    158.6 MiB | K   20 | loss  2.491242078e+00 | Ndiff
   32.000/1000 after      0 sec. |    158.6 MiB | K   20 | loss  2.490781261e+00 | Ndiff
   33.000/1000 after      0 sec. |    158.6 MiB | K   20 | loss  2.490270354e+00 | Ndiff
   34.000/1000 after      0 sec. |    158.6 MiB | K   20 | loss  2.489692235e+00 | Ndiff
   35.000/1000 after      0 sec. |    158.6 MiB | K   20 | loss  2.489027945e+00 | Ndiff
   36.000/1000 after      0 sec. |    158.6 MiB | K   20 | loss  2.488255286e+00 | Ndiff
   37.000/1000 after      0 sec. |    158.6 MiB | K   20 | loss  2.487347489e+00 | Ndiff
   38.000/1000 after      0 sec. |    158.6 MiB | K   20 | loss  2.486272015e+00 | Ndiff
   39.000/1000 after      0 sec. |    158.6 MiB | K   20 | loss  2.484989607e+00 | Ndiff
   40.000/1000 after      0 sec. |    158.6 MiB | K   20 | loss  2.483453885e+00 | Ndiff
   41.000/1000 after      0 sec. |    158.6 MiB | K   20 | loss  2.481612017e+00 | Ndiff
   42.000/1000 after      0 sec. |    158.6 MiB | K   20 | loss  2.479407351e+00 | Ndiff
   43.000/1000 after      0 sec. |    158.6 MiB | K   20 | loss  2.476785615e+00 | Ndiff
   44.000/1000 after      0 sec. |    158.6 MiB | K   20 | loss  2.473707547e+00 | Ndiff
   45.000/1000 after      0 sec. |    158.6 MiB | K   20 | loss  2.470173027e+00 | Ndiff
   46.000/1000 after      0 sec. |    158.6 MiB | K   20 | loss  2.466261998e+00 | Ndiff
   47.000/1000 after      0 sec. |    158.6 MiB | K   20 | loss  2.462175093e+00 | Ndiff
   48.000/1000 after      0 sec. |    158.6 MiB | K   20 | loss  2.458145085e+00 | Ndiff
   49.000/1000 after      0 sec. |    158.6 MiB | K   20 | loss  2.453753531e+00 | Ndiff
   50.000/1000 after      1 sec. |    158.6 MiB | K   20 | loss  2.441832238e+00 | Ndiff
   51.000/1000 after      1 sec. |    158.6 MiB | K   20 | loss  2.437414963e+00 | Ndiff
   52.000/1000 after      1 sec. |    158.6 MiB | K   20 | loss  2.435586138e+00 | Ndiff
   53.000/1000 after      1 sec. |    158.6 MiB | K   20 | loss  2.433555668e+00 | Ndiff
   54.000/1000 after      1 sec. |    158.6 MiB | K   20 | loss  2.431430213e+00 | Ndiff
   55.000/1000 after      1 sec. |    158.6 MiB | K   20 | loss  2.429483776e+00 | Ndiff
   56.000/1000 after      1 sec. |    158.6 MiB | K   20 | loss  2.428084819e+00 | Ndiff
   57.000/1000 after      1 sec. |    158.6 MiB | K   20 | loss  2.427098172e+00 | Ndiff
   58.000/1000 after      1 sec. |    158.6 MiB | K   20 | loss  2.425535651e+00 | Ndiff
   59.000/1000 after      1 sec. |    158.6 MiB | K   20 | loss  2.415976963e+00 | Ndiff
   60.000/1000 after      1 sec. |    158.6 MiB | K   20 | loss  2.412692266e+00 | Ndiff
   61.000/1000 after      1 sec. |    158.6 MiB | K   20 | loss  2.412692093e+00 | Ndiff
   62.000/1000 after      1 sec. |    158.6 MiB | K   20 | loss  2.412692005e+00 | Ndiff
   63.000/1000 after      1 sec. |    158.6 MiB | K   20 | loss  2.412691958e+00 | Ndiff
   64.000/1000 after      1 sec. |    158.6 MiB | K   20 | loss  2.412691932e+00 | Ndiff
   65.000/1000 after      1 sec. |    158.6 MiB | K   20 | loss  2.412691918e+00 | Ndiff
   66.000/1000 after      1 sec. |    158.6 MiB | K   20 | loss  2.412691910e+00 | Ndiff
   67.000/1000 after      1 sec. |    158.6 MiB | K   20 | loss  2.412691906e+00 | Ndiff
   68.000/1000 after      1 sec. |    158.6 MiB | K   20 | loss  2.412691904e+00 | Ndiff
   69.000/1000 after      1 sec. |    158.6 MiB | K   20 | loss  2.412691903e+00 | Ndiff
   70.000/1000 after      1 sec. |    158.6 MiB | K   20 | loss  2.412691902e+00 | Ndiff
   71.000/1000 after      1 sec. |    158.6 MiB | K   20 | loss  2.412691901e+00 | Ndiff
   72.000/1000 after      1 sec. |    158.6 MiB | K   20 | loss  2.412691901e+00 | Ndiff
   73.000/1000 after      1 sec. |    158.6 MiB | K   20 | loss  2.412691901e+00 | Ndiff
   74.000/1000 after      1 sec. |    158.6 MiB | K   20 | loss  2.412691901e+00 | Ndiff
   75.000/1000 after      1 sec. |    158.6 MiB | K   20 | loss  2.412691901e+00 | Ndiff
   76.000/1000 after      1 sec. |    158.6 MiB | K   20 | loss  2.412691901e+00 | Ndiff
   77.000/1000 after      1 sec. |    158.6 MiB | K   20 | loss  2.412691901e+00 | Ndiff
   78.000/1000 after      1 sec. |    158.6 MiB | K   20 | loss  2.412691901e+00 | Ndiff
   79.000/1000 after      1 sec. |    158.6 MiB | K   20 | loss  2.412691901e+00 | Ndiff
... done. converged.
```

Gauss observations + VB

Assume full covariances.

Start with too many clusters (K=20)

```
full_trained_model, full_info_dict = bnpy.run(
    dataset, 'DPMixtureModel', 'Gauss', 'VB',
    output_path='/tmp/faithful/showcase-K=20-lik=Gauss-ECovMat=5*eye/',
    nLap=1000, nTask=1, nBatch=1, convergeThr=0.0001,
    gamma0=gamma, sF=sF, ECovMat='eye',
    K=K, initname='randexamples',
    )
show_clusters_over_time(full_info_dict['task_output_path'])
```



Out:

```
WARNING: Found unrecognized keyword args. These are ignored.
  --nBatch
Dataset Summary:
X Data
  num examples: 272
  num dims: 2
Allocation Model:  DP mixture with K=0. Concentration gamma0= 5.00
Obs. Data  Model:  Gaussian with full covariance.
Obs. Data  Prior:  Gauss-Wishart on mean and covar of each cluster
  E[  mean[k] ] =
   [ 0.  0.]
  E[ covar[k] ] =
  [[ 5.  0.]
   [ 0.  5.]]
Initialization:
  initname = randexamples
  K = 20 (number of clusters)
  seed = 1607680
  elapsed_time: 0.0 sec
Learn Alg: VB | task  1/1 | alg. seed: 1607680 | data order seed: 8541952
task_output_path: /tmp/faithful/showcase-K=20-lik=Gauss-ECovMat=5*eye/1
        1/1000 after      0 sec. |    154.6 MiB | K   20 | loss  2.904406498e+00 |
        2/1000 after      0 sec. |    154.6 MiB | K   20 | loss  2.876066302e+00 | Ndiff
        3/1000 after      0 sec. |    154.6 MiB | K   20 | loss  2.864944306e+00 | Ndiff
        4/1000 after      0 sec. |    154.6 MiB | K   20 | loss  2.852026479e+00 | Ndiff
```

```
    5/1000 after      0 sec. |    154.6 MiB | K   20 | loss  2.836506635e+00 | Ndiff
    6/1000 after      0 sec. |    154.6 MiB | K   20 | loss  2.817804673e+00 | Ndiff
    7/1000 after      0 sec. |    154.6 MiB | K   20 | loss  2.787080350e+00 | Ndiff
    8/1000 after      0 sec. |    154.6 MiB | K   20 | loss  2.743017730e+00 | Ndiff
    9/1000 after      0 sec. |    154.6 MiB | K   20 | loss  2.714906517e+00 | Ndiff
   10/1000 after      0 sec. |    154.6 MiB | K   20 | loss  2.670842807e+00 | Ndiff
   11/1000 after      0 sec. |    154.6 MiB | K   20 | loss  2.640944068e+00 | Ndiff
   12/1000 after      0 sec. |    154.6 MiB | K   20 | loss  2.600612644e+00 | Ndiff
   13/1000 after      0 sec. |    154.6 MiB | K   20 | loss  2.582832638e+00 | Ndiff
   14/1000 after      0 sec. |    154.6 MiB | K   20 | loss  2.569005251e+00 | Ndiff
   15/1000 after      0 sec. |    154.6 MiB | K   20 | loss  2.544279312e+00 | Ndiff
   16/1000 after      0 sec. |    154.6 MiB | K   20 | loss  2.523247072e+00 | Ndiff
   17/1000 after      0 sec. |    154.6 MiB | K   20 | loss  2.511328327e+00 | Ndiff
   18/1000 after      0 sec. |    154.6 MiB | K   20 | loss  2.503251361e+00 | Ndiff
   19/1000 after      0 sec. |    154.6 MiB | K   20 | loss  2.491852524e+00 | Ndiff
   20/1000 after      0 sec. |    154.6 MiB | K   20 | loss  2.490247584e+00 | Ndiff
   21/1000 after      0 sec. |    154.6 MiB | K   20 | loss  2.488483904e+00 | Ndiff
   22/1000 after      0 sec. |    154.6 MiB | K   20 | loss  2.486427687e+00 | Ndiff
   23/1000 after      0 sec. |    154.6 MiB | K   20 | loss  2.484005616e+00 | Ndiff
   24/1000 after      0 sec. |    154.6 MiB | K   20 | loss  2.481221463e+00 | Ndiff
   25/1000 after      0 sec. |    154.6 MiB | K   20 | loss  2.478177971e+00 | Ndiff
   26/1000 after      0 sec. |    154.6 MiB | K   20 | loss  2.475094563e+00 | Ndiff
   27/1000 after      0 sec. |    154.6 MiB | K   20 | loss  2.471974138e+00 | Ndiff
   28/1000 after      0 sec. |    154.6 MiB | K   20 | loss  2.467024647e+00 | Ndiff
   29/1000 after      0 sec. |    154.6 MiB | K   20 | loss  2.453713427e+00 | Ndiff
   30/1000 after      0 sec. |    154.6 MiB | K   20 | loss  2.445147652e+00 | Ndiff
   31/1000 after      0 sec. |    154.6 MiB | K   20 | loss  2.439161802e+00 | Ndiff
   32/1000 after      1 sec. |    154.6 MiB | K   20 | loss  2.438783471e+00 | Ndiff
   33/1000 after      1 sec. |    154.6 MiB | K   20 | loss  2.438429230e+00 | Ndiff
   34/1000 after      1 sec. |    154.6 MiB | K   20 | loss  2.438098475e+00 | Ndiff
   35/1000 after      1 sec. |    154.6 MiB | K   20 | loss  2.437790556e+00 | Ndiff
   36/1000 after      1 sec. |    154.6 MiB | K   20 | loss  2.437503774e+00 | Ndiff
   37/1000 after      1 sec. |    154.6 MiB | K   20 | loss  2.437235314e+00 | Ndiff
   38/1000 after      1 sec. |    154.6 MiB | K   20 | loss  2.436981439e+00 | Ndiff
   39/1000 after      1 sec. |    154.6 MiB | K   20 | loss  2.436737744e+00 | Ndiff
   40/1000 after      1 sec. |    154.6 MiB | K   20 | loss  2.436499373e+00 | Ndiff
   41/1000 after      1 sec. |    154.6 MiB | K   20 | loss  2.436261156e+00 | Ndiff
   42/1000 after      1 sec. |    154.6 MiB | K   20 | loss  2.436017666e+00 | Ndiff
   43/1000 after      1 sec. |    154.6 MiB | K   20 | loss  2.435763198e+00 | Ndiff
   44/1000 after      1 sec. |    154.6 MiB | K   20 | loss  2.435491737e+00 | Ndiff
   45/1000 after      1 sec. |    154.6 MiB | K   20 | loss  2.435196931e+00 | Ndiff
   46/1000 after      1 sec. |    154.6 MiB | K   20 | loss  2.434872198e+00 | Ndiff
   47/1000 after      1 sec. |    154.6 MiB | K   20 | loss  2.434511031e+00 | Ndiff
   48/1000 after      1 sec. |    154.6 MiB | K   20 | loss  2.434107639e+00 | Ndiff
   49/1000 after      1 sec. |    154.6 MiB | K   20 | loss  2.433657974e+00 | Ndiff
   50/1000 after      1 sec. |    154.6 MiB | K   20 | loss  2.433161085e+00 | Ndiff
   51/1000 after      1 sec. |    154.6 MiB | K   20 | loss  2.432620547e+00 | Ndiff
   52/1000 after      1 sec. |    154.6 MiB | K   20 | loss  2.432045498e+00 | Ndiff
   53/1000 after      1 sec. |    154.6 MiB | K   20 | loss  2.431450777e+00 | Ndiff
   54/1000 after      1 sec. |    154.6 MiB | K   20 | loss  2.430855677e+00 | Ndiff
   55/1000 after      1 sec. |    154.6 MiB | K   20 | loss  2.430280760e+00 | Ndiff
   56/1000 after      1 sec. |    154.6 MiB | K   20 | loss  2.429742357e+00 | Ndiff
   57/1000 after      1 sec. |    154.6 MiB | K   20 | loss  2.429246005e+00 | Ndiff
```

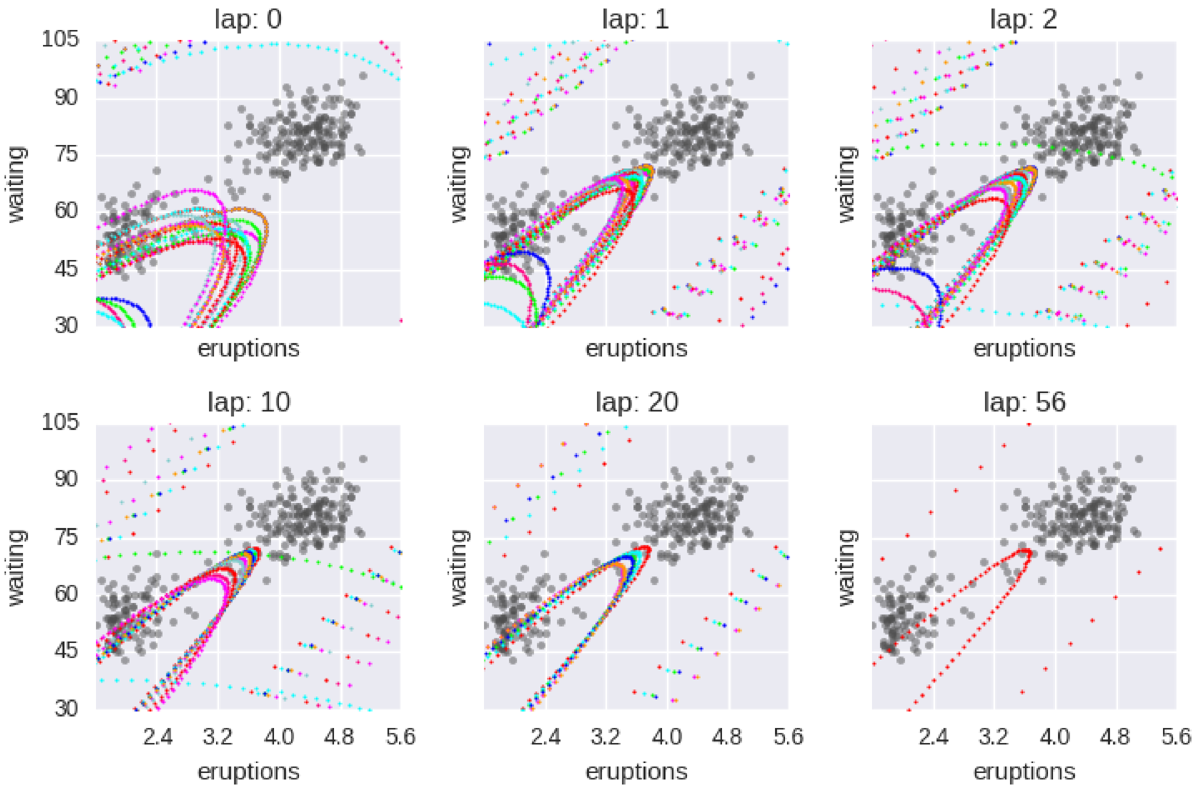
```
       58/1000 after      1 sec. |    154.6 MiB | K   20 | loss  2.428783131e+00 | Ndiff
       59/1000 after      1 sec. |    154.6 MiB | K   20 | loss  2.428335044e+00 | Ndiff
       60/1000 after      1 sec. |    154.6 MiB | K   20 | loss  2.427882144e+00 | Ndiff
       61/1000 after      1 sec. |    154.6 MiB | K   20 | loss  2.427412198e+00 | Ndiff
       62/1000 after      1 sec. |    154.6 MiB | K   20 | loss  2.426923365e+00 | Ndiff
       63/1000 after      1 sec. |    154.6 MiB | K   20 | loss  2.426419184e+00 | Ndiff
       64/1000 after      1 sec. |    154.6 MiB | K   20 | loss  2.425898789e+00 | Ndiff
       65/1000 after      1 sec. |    154.6 MiB | K   20 | loss  2.425353548e+00 | Ndiff
       66/1000 after      1 sec. |    154.6 MiB | K   20 | loss  2.424771068e+00 | Ndiff
       67/1000 after      1 sec. |    154.6 MiB | K   20 | loss  2.424138280e+00 | Ndiff
       68/1000 after      1 sec. |    154.6 MiB | K   20 | loss  2.423441574e+00 | Ndiff
       69/1000 after      1 sec. |    154.6 MiB | K   20 | loss  2.422666132e+00 | Ndiff
       70/1000 after      1 sec. |    154.6 MiB | K   20 | loss  2.421796837e+00 | Ndiff
       71/1000 after      1 sec. |    154.6 MiB | K   20 | loss  2.420824917e+00 | Ndiff
       72/1000 after      1 sec. |    154.6 MiB | K   20 | loss  2.419763730e+00 | Ndiff
       73/1000 after      1 sec. |    154.6 MiB | K   20 | loss  2.418655856e+00 | Ndiff
       74/1000 after      1 sec. |    154.6 MiB | K   20 | loss  2.417536411e+00 | Ndiff
       75/1000 after      1 sec. |    154.6 MiB | K   20 | loss  2.416396660e+00 | Ndiff
       76/1000 after      1 sec. |    154.6 MiB | K   20 | loss  2.415209641e+00 | Ndiff
       77/1000 after      1 sec. |    154.6 MiB | K   20 | loss  2.413957301e+00 | Ndiff
       78/1000 after      1 sec. |    154.6 MiB | K   20 | loss  2.412627524e+00 | Ndiff
       79/1000 after      1 sec. |    154.6 MiB | K   20 | loss  2.411204929e+00 | Ndiff
       80/1000 after      1 sec. |    154.6 MiB | K   20 | loss  2.409665366e+00 | Ndiff
       81/1000 after      1 sec. |    154.6 MiB | K   20 | loss  2.407972796e+00 | Ndiff
       82/1000 after      1 sec. |    154.6 MiB | K   20 | loss  2.406075452e+00 | Ndiff
       83/1000 after      1 sec. |    154.6 MiB | K   20 | loss  2.403897086e+00 | Ndiff
       84/1000 after      1 sec. |    154.6 MiB | K   20 | loss  2.401317119e+00 | Ndiff
       85/1000 after      1 sec. |    154.6 MiB | K   20 | loss  2.398127190e+00 | Ndiff
       86/1000 after      1 sec. |    154.6 MiB | K   20 | loss  2.393870554e+00 | Ndiff
       87/1000 after      1 sec. |    154.6 MiB | K   20 | loss  2.385927886e+00 | Ndiff
       88/1000 after      1 sec. |    154.6 MiB | K   20 | loss  2.373557979e+00 | Ndiff
       89/1000 after      1 sec. |    154.6 MiB | K   20 | loss  2.370431956e+00 | Ndiff
       90/1000 after      1 sec. |    154.6 MiB | K   20 | loss  2.361734581e+00 | Ndiff
       91/1000 after      1 sec. |    154.6 MiB | K   20 | loss  2.356420337e+00 | Ndiff
       92/1000 after      1 sec. |    154.6 MiB | K   20 | loss  2.356419191e+00 | Ndiff
       93/1000 after      1 sec. |    154.6 MiB | K   20 | loss  2.356418521e+00 | Ndiff
       94/1000 after      1 sec. |    154.6 MiB | K   20 | loss  2.356418120e+00 | Ndiff
       95/1000 after      1 sec. |    154.6 MiB | K   20 | loss  2.356417878e+00 | Ndiff
       96/1000 after      2 sec. |    154.6 MiB | K   20 | loss  2.356417731e+00 | Ndiff
       97/1000 after      2 sec. |    154.6 MiB | K   20 | loss  2.356417640e+00 | Ndiff
       98/1000 after      2 sec. |    154.6 MiB | K   20 | loss  2.356417584e+00 | Ndiff
       99/1000 after      2 sec. |    154.6 MiB | K   20 | loss  2.356417550e+00 | Ndiff
      100/1000 after      2 sec. |    154.6 MiB | K   20 | loss  2.356417528e+00 | Ndiff
      101/1000 after      2 sec. |    154.6 MiB | K   20 | loss  2.356417515e+00 | Ndiff
      102/1000 after      2 sec. |    154.6 MiB | K   20 | loss  2.356417507e+00 | Ndiff
      103/1000 after      2 sec. |    154.6 MiB | K   20 | loss  2.356417502e+00 | Ndiff
      104/1000 after      2 sec. |    154.6 MiB | K   20 | loss  2.356417499e+00 | Ndiff
      105/1000 after      2 sec. |    154.6 MiB | K   20 | loss  2.356417497e+00 | Ndiff
      106/1000 after      2 sec. |    154.6 MiB | K   20 | loss  2.356417495e+00 | Ndiff
      107/1000 after      2 sec. |    154.6 MiB | K   20 | loss  2.356417495e+00 | Ndiff
      108/1000 after      2 sec. |    154.6 MiB | K   20 | loss  2.356417494e+00 | Ndiff
      109/1000 after      2 sec. |    154.6 MiB | K   20 | loss  2.356417494e+00 | Ndiff
      110/1000 after      2 sec. |    154.6 MiB | K   20 | loss  2.356417494e+00 | Ndiff
      111/1000 after      2 sec. |    154.6 MiB | K   20 | loss  2.356417493e+00 | Ndiff
      112/1000 after      2 sec. |    154.6 MiB | K   20 | loss  2.356417493e+00 | Ndiff
      113/1000 after      2 sec. |    154.6 MiB | K   20 | loss  2.356417493e+00 | Ndiff
      114/1000 after      2 sec. |    154.6 MiB | K   20 | loss  2.356417493e+00 | Ndiff
      115/1000 after      2 sec. |    154.6 MiB | K   20 | loss  2.356417493e+00 | Ndiff
      116/1000 after      2 sec. |    154.6 MiB | K   20 | loss  2.356417493e+00 | Ndiff
      117/1000 after      2 sec. |    154.6 MiB | K   20 | loss  2.356417493e+00 | Ndiff
... done. converged.
```

ZeroMeanGauss observations + VB

Assume full covariances and fix all means to zero.

Start with too many clusters (K=20)

```
zm_trained_model, zm_info_dict = bnpy.run(
    dataset, 'DPMixtureModel', 'ZeroMeanGauss', 'VB',
    output_path='/tmp/faithful/showcase-K=20-lik=ZeroMeanGauss-ECovMat=5*eye/',
    nLap=1000, nTask=1, nBatch=1, convergeThr=0.0001,
    gamma0=gamma, sF=sF, ECovMat='eye',
    K=K, initname='randexamples',
    )
show_clusters_over_time(zm_info_dict['task_output_path'])
```



Out:

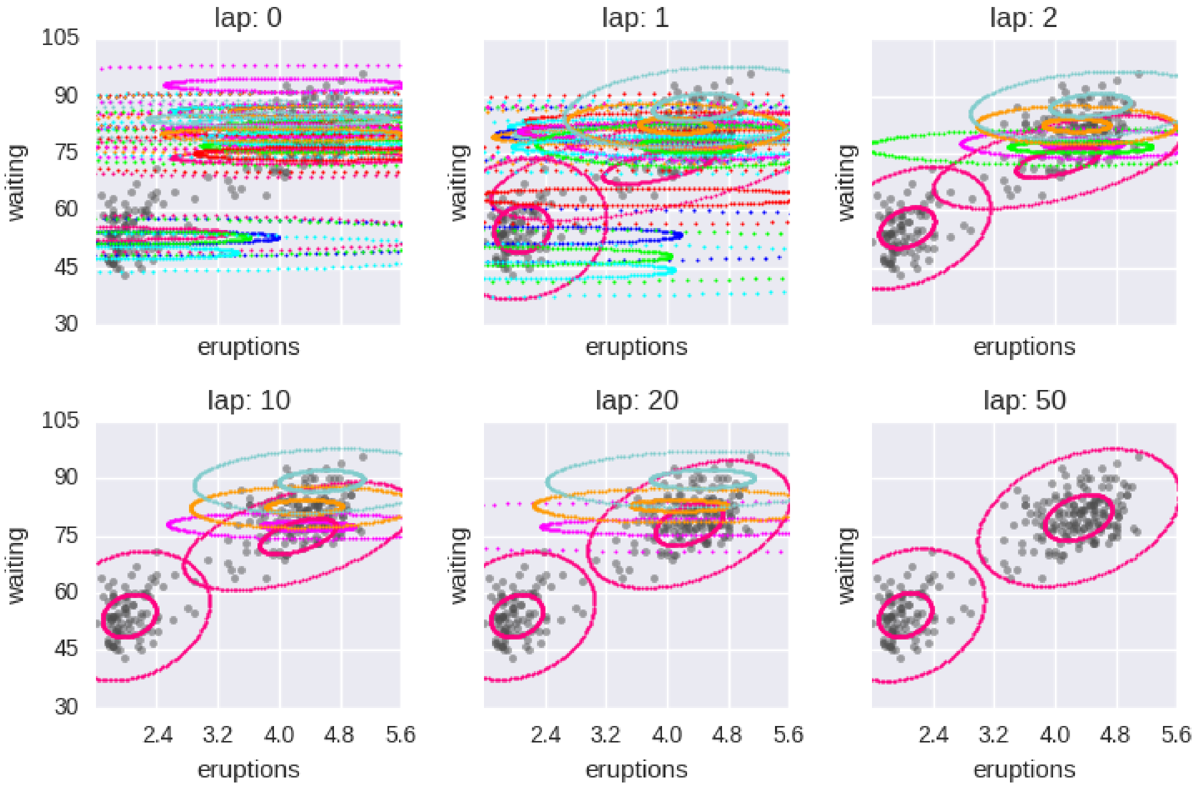
```
WARNING: Found unrecognized keyword args. These are ignored.
  --nBatch
Dataset Summary:
X Data
  num examples: 272
  num dims: 2
Allocation Model:  DP mixture with K=0. Concentration gamma0= 5.00
Obs. Data  Model:  Gaussian with fixed zero means, full covariance.
Obs. Data  Prior:  Wishart on prec matrix Lam
  E[ CovMat[k] ] =
  [[ 5.  0.]
   [ 0.  5.]]
Initialization:
  initname = randexamples
  K = 20 (number of clusters)
  seed = 1607680
  elapsed_time: 0.0 sec
Learn Alg: VB | task  1/1 | alg. seed: 1607680 | data order seed: 8541952
task_output_path: /tmp/faithful/showcase-K=20-lik=ZeroMeanGauss-ECovMat=5*eye/1
        1/1000 after      0 sec. |    154.6 MiB | K   20 | loss  4.019419551e+00 |
        2/1000 after      0 sec. |    154.6 MiB | K   20 | loss  3.989063967e+00 | Ndiff
        3/1000 after      0 sec. |    154.6 MiB | K   20 | loss  3.941416771e+00 | Ndiff
        4/1000 after      0 sec. |    154.6 MiB | K   20 | loss  3.899222162e+00 | Ndiff
        5/1000 after      0 sec. |    154.6 MiB | K   20 | loss  3.876371765e+00 | Ndiff
        6/1000 after      0 sec. |    154.6 MiB | K   20 | loss  3.842038547e+00 | Ndiff
        7/1000 after      0 sec. |    154.6 MiB | K   20 | loss  3.808536791e+00 | Ndiff
        8/1000 after      0 sec. |    154.6 MiB | K   20 | loss  3.779782054e+00 | Ndiff
        9/1000 after      0 sec. |    154.6 MiB | K   20 | loss  3.755287922e+00 | Ndiff
       10/1000 after      0 sec. |    154.6 MiB | K   20 | loss  3.733837701e+00 | Ndiff
       11/1000 after      0 sec. |    154.6 MiB | K   20 | loss  3.700117498e+00 | Ndiff
       12/1000 after      0 sec. |    154.6 MiB | K   20 | loss  3.694082693e+00 | Ndiff
       13/1000 after      0 sec. |    154.6 MiB | K   20 | loss  3.686765019e+00 | Ndiff
       14/1000 after      0 sec. |    154.6 MiB | K   20 | loss  3.675117147e+00 | Ndiff
       15/1000 after      0 sec. |    154.6 MiB | K   20 | loss  3.652297952e+00 | Ndiff
       16/1000 after      0 sec. |    154.6 MiB | K   20 | loss  3.639249110e+00 | Ndiff
       17/1000 after      0 sec. |    154.6 MiB | K   20 | loss  3.603588941e+00 | Ndiff
       18/1000 after      0 sec. |    154.6 MiB | K   20 | loss  3.598749399e+00 | Ndiff
       19/1000 after      0 sec. |    154.6 MiB | K   20 | loss  3.592583511e+00 | Ndiff
       20/1000 after      0 sec. |    154.6 MiB | K   20 | loss  3.574936974e+00 | Ndiff
       21/1000 after      0 sec. |    154.6 MiB | K   20 | loss  3.572459120e+00 | Ndiff
       22/1000 after      0 sec. |    154.6 MiB | K   20 | loss  3.570977707e+00 | Ndiff
       23/1000 after      0 sec. |    154.6 MiB | K   20 | loss  3.569306514e+00 | Ndiff
       24/1000 after      0 sec. |    154.6 MiB | K   20 | loss  3.567412054e+00 | Ndiff
       25/1000 after      0 sec. |    154.6 MiB | K   20 | loss  3.565249330e+00 | Ndiff
       26/1000 after      0 sec. |    154.6 MiB | K   20 | loss  3.562749369e+00 | Ndiff
       27/1000 after      0 sec. |    154.6 MiB | K   20 | loss  3.559772804e+00 | Ndiff
       28/1000 after      0 sec. |    154.6 MiB | K   20 | loss  3.555894232e+00 | Ndiff
       29/1000 after      0 sec. |    154.6 MiB | K   20 | loss  3.549078207e+00 | Ndiff
       30/1000 after      0 sec. |    154.6 MiB | K   20 | loss  3.531303622e+00 | Ndiff
       31/1000 after      0 sec. |    154.6 MiB | K   20 | loss  3.529382170e+00 | Ndiff
       32/1000 after      0 sec. |    154.6 MiB | K   20 | loss  3.527237440e+00 | Ndiff
       33/1000 after      0 sec. |    154.6 MiB | K   20 | loss  3.524655304e+00 | Ndiff
       34/1000 after      0 sec. |    154.6 MiB | K   20 | loss  3.521314832e+00 | Ndiff
       35/1000 after      0 sec. |    154.6 MiB | K   20 | loss  3.516111933e+00 | Ndiff
       36/1000 after      0 sec. |    154.6 MiB | K   20 | loss  3.502294322e+00 | Ndiff
       37/1000 after      0 sec. |    154.6 MiB | K   20 | loss  3.491586191e+00 | Ndiff
       38/1000 after      0 sec. |    154.6 MiB | K   20 | loss  3.485039030e+00 | Ndiff
       39/1000 after      0 sec. |    154.6 MiB | K   20 | loss  3.468792370e+00 | Ndiff
       40/1000 after      0 sec. |    154.6 MiB | K   20 | loss  3.467909258e+00 | Ndiff
       41/1000 after      1 sec. |    154.6 MiB | K   20 | loss  3.466944463e+00 | Ndiff
       42/1000 after      1 sec. |    154.6 MiB | K   20 | loss  3.465874350e+00 | Ndiff
       43/1000 after      1 sec. |    154.6 MiB | K   20 | loss  3.464679105e+00 | Ndiff
       44/1000 after      1 sec. |    154.6 MiB | K   20 | loss  3.463333221e+00 | Ndiff
       45/1000 after      1 sec. |    154.6 MiB | K   20 | loss  3.461803293e+00 | Ndiff
       46/1000 after      1 sec. |    154.6 MiB | K   20 | loss  3.460044727e+00 | Ndiff
       47/1000 after      1 sec. |    154.6 MiB | K   20 | loss  3.457996608e+00 | Ndiff
       48/1000 after      1 sec. |    154.6 MiB | K   20 | loss  3.455573178e+00 | Ndiff
       49/1000 after      1 sec. |    154.6 MiB | K   20 | loss  3.452647608e+00 | Ndiff
       50/1000 after      1 sec. |    154.6 MiB | K   20 | loss  3.449011332e+00 | Ndiff
       51/1000 after      1 sec. |    154.6 MiB | K   20 | loss  3.444220039e+00 | Ndiff
       52/1000 after      1 sec. |    154.6 MiB | K   20 | loss  3.436689553e+00 | Ndiff
       53/1000 after      1 sec. |    154.6 MiB | K   20 | loss  3.416217624e+00 | Ndiff
       54/1000 after      1 sec. |    154.6 MiB | K   20 | loss  3.397676918e+00 | Ndiff
       55/1000 after      1 sec. |    154.6 MiB | K   20 | loss  3.395369454e+00 | Ndiff
       56/1000 after      1 sec. |    154.6 MiB | K   20 | loss  3.395369454e+00 | Ndiff
... done. converged.
```

Gauss observations + stochastic VB

Assume full covariances and fix all means to zero.

Start with too many clusters (K=20)

```
stoch_trained_model, stoch_info_dict = bnpy.run(
    dataset, 'DPMixtureModel', 'Gauss', 'soVB',
    output_path=\
        '/tmp/faithful/showcase-K=20-lik=Gauss-ECovMat=5*eye-alg=soVB/',
    nLap=50, nTask=1, nBatch=50,
    rhoexp=0.51, rhodelay=1.0,
    gamma0=gamma, sF=sF, ECovMat='eye',
    K=K, initname='randexamples',
    )
show_clusters_over_time(stoch_info_dict['task_output_path'])
```

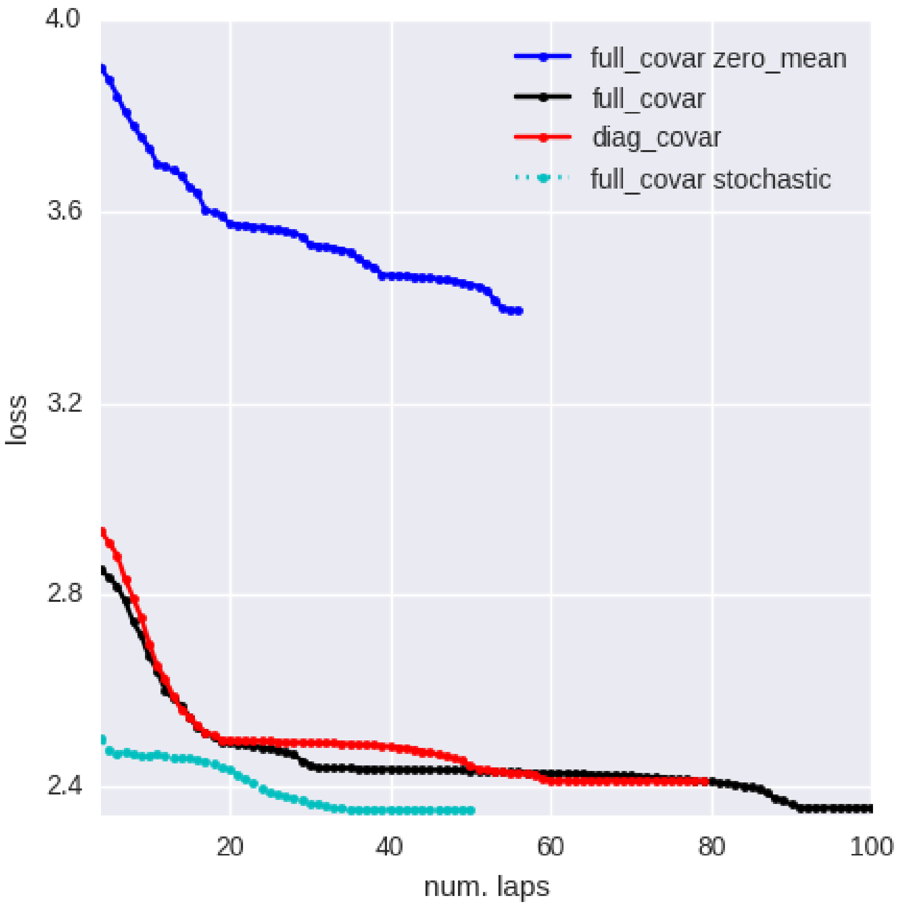


Out:

```
Dataset Summary:
X Data
  total size: 272 units
  batch size: 6 units
  num. batches: 50
Allocation Model:  DP mixture with K=0. Concentration gamma0= 5.00
Obs. Data  Model:  Gaussian with full covariance.
Obs. Data  Prior:  Gauss-Wishart on mean and covar of each cluster
  E[  mean[k] ] =
   [ 0.  0.]
  E[ covar[k] ] =
  [[ 5.  0.]
   [ 0.  5.]]
Initialization:
  initname = randexamples
  K = 20 (number of clusters)
  seed = 1607680
  elapsed_time: 0.0 sec
Learn Alg: soVB | task  1/1 | alg. seed: 1607680 | data order seed: 8541952
task_output_path: /tmp/faithful/showcase-K=20-lik=Gauss-ECovMat=5*eye-alg=soVB/1
    0.020/50 after      0 sec. |    154.8 MiB | K   20 | loss  3.020105590e+01 |  lrate 0.7
    0.040/50 after      0 sec. |    154.8 MiB | K   20 | loss  1.693550049e+01 |  lrate 0.5
    0.060/50 after      0 sec. |    154.8 MiB | K   20 | loss  1.182159020e+01 |  lrate 0.4
    1.000/50 after      1 sec. |    154.8 MiB | K   20 | loss  1.314590658e+01 |  lrate 0.1
    2.000/50 after      1 sec. |    154.8 MiB | K   20 | loss  2.522804913e+00 |  lrate 0.0
    3.000/50 after      2 sec. |    154.8 MiB | K   20 | loss  2.507513303e+00 |  lrate 0.0
    4.000/50 after      3 sec. |    154.8 MiB | K   20 | loss  2.498366509e+00 |  lrate 0.0
    5.000/50 after      4 sec. |    154.8 MiB | K   20 | loss  2.475120548e+00 |  lrate 0.0
    6.000/50 after      4 sec. |    154.8 MiB | K   20 | loss  2.469641662e+00 |  lrate 0.0
    7.000/50 after      5 sec. |    154.8 MiB | K   20 | loss  2.470932762e+00 |  lrate 0.0
    8.000/50 after      6 sec. |    154.8 MiB | K   20 | loss  2.466158362e+00 |  lrate 0.0
    9.000/50 after      6 sec. |    154.8 MiB | K   20 | loss  2.465517619e+00 |  lrate 0.0
   10.000/50 after      7 sec. |    154.8 MiB | K   20 | loss  2.465435382e+00 |  lrate 0.0
   11.000/50 after      8 sec. |    154.8 MiB | K   20 | loss  2.467696036e+00 |  lrate 0.0
   12.000/50 after      8 sec. |    154.8 MiB | K   20 | loss  2.461949856e+00 |  lrate 0.0
   13.000/50 after      9 sec. |    154.8 MiB | K   20 | loss  2.461043110e+00 |  lrate 0.0
   14.000/50 after     10 sec. |    154.8 MiB | K   20 | loss  2.461264382e+00 |  lrate 0.0
   15.000/50 after     11 sec. |    154.8 MiB | K   20 | loss  2.458851451e+00 |  lrate 0.0
   16.000/50 after     11 sec. |    154.8 MiB | K   20 | loss  2.454655643e+00 |  lrate 0.0
   17.000/50 after     12 sec. |    154.8 MiB | K   20 | loss  2.451234188e+00 |  lrate 0.0
   18.000/50 after     13 sec. |    154.8 MiB | K   20 | loss  2.446281746e+00 |  lrate 0.0
   19.000/50 after     13 sec. |    154.8 MiB | K   20 | loss  2.440793273e+00 |  lrate 0.0
   20.000/50 after     14 sec. |    154.8 MiB | K   20 | loss  2.434143198e+00 |  lrate 0.0
   21.000/50 after     15 sec. |    154.8 MiB | K   20 | loss  2.423545849e+00 |  lrate 0.0
   22.000/50 after     15 sec. |    154.8 MiB | K   20 | loss  2.414924985e+00 |  lrate 0.0
   23.000/50 after     16 sec. |    154.8 MiB | K   20 | loss  2.405741175e+00 |  lrate 0.0
   24.000/50 after     17 sec. |    154.8 MiB | K   20 | loss  2.396273286e+00 |  lrate 0.0
   25.000/50 after     17 sec. |    154.8 MiB | K   20 | loss  2.388169784e+00 |  lrate 0.0
   26.000/50 after     18 sec. |    154.8 MiB | K   20 | loss  2.383086462e+00 |  lrate 0.0
   27.000/50 after     19 sec. |    154.8 MiB | K   20 | loss  2.378566216e+00 |  lrate 0.0
   28.000/50 after     20 sec. |    154.8 MiB | K   20 | loss  2.374468977e+00 |  lrate 0.0
   29.000/50 after     20 sec. |    154.8 MiB | K   20 | loss  2.369573743e+00 |  lrate 0.0
   30.000/50 after     21 sec. |    154.8 MiB | K   20 | loss  2.365030091e+00 |  lrate 0.0
   31.000/50 after     22 sec. |    154.8 MiB | K   20 | loss  2.361840492e+00 |  lrate 0.0
   32.000/50 after     22 sec. |    154.8 MiB | K   20 | loss  2.358180297e+00 |  lrate 0.0
   33.000/50 after     23 sec. |    154.8 MiB | K   20 | loss  2.355370718e+00 |  lrate 0.0
   34.000/50 after     24 sec. |    154.8 MiB | K   20 | loss  2.354001069e+00 |  lrate 0.0
   35.000/50 after     24 sec. |    154.8 MiB | K   20 | loss  2.352655966e+00 |  lrate 0.0
   36.000/50 after     25 sec. |    154.8 MiB | K   20 | loss  2.351706756e+00 |  lrate 0.0
   37.000/50 after     26 sec. |    154.8 MiB | K   20 | loss  2.351346242e+00 |  lrate 0.0
   38.000/50 after     27 sec. |    154.8 MiB | K   20 | loss  2.351470554e+00 |  lrate 0.0
   39.000/50 after     27 sec. |    154.8 MiB | K   20 | loss  2.351530696e+00 |  lrate 0.0
   40.000/50 after     28 sec. |    154.8 MiB | K   20 | loss  2.351708300e+00 |  lrate 0.0
   41.000/50 after     29 sec. |    154.8 MiB | K   20 | loss  2.351489520e+00 |  lrate 0.0
   42.000/50 after     29 sec. |    154.8 MiB | K   20 | loss  2.351625569e+00 |  lrate 0.0
   43.000/50 after     30 sec. |    154.8 MiB | K   20 | loss  2.351357552e+00 |  lrate 0.0
   44.000/50 after     31 sec. |    154.8 MiB | K   20 | loss  2.351536177e+00 |  lrate 0.0
   45.000/50 after     32 sec. |    154.8 MiB | K   20 | loss  2.351543540e+00 |  lrate 0.0
   46.000/50 after     32 sec. |    154.8 MiB | K   20 | loss  2.351346627e+00 |  lrate 0.0
   47.000/50 after     33 sec. |    154.8 MiB | K   20 | loss  2.351293738e+00 |  lrate 0.0
   48.000/50 after     34 sec. |    154.8 MiB | K   20 | loss  2.351507196e+00 |  lrate 0.0
   49.000/50 after     34 sec. |    154.8 MiB | K   20 | loss  2.351496294e+00 |  lrate 0.0
   50.000/50 after     35 sec. |    154.8 MiB | K   20 | loss  2.351523756e+00 |  lrate 0.0
... active. not converged.
```

Compare loss function traces for all methods

```
pylab.figure()

pylab.plot(
    zm_info_dict['lap_history'],
    zm_info_dict['loss_history'], 'b.-',
    label='full_covar zero_mean')
pylab.plot(
    full_info_dict['lap_history'],
    full_info_dict['loss_history'], 'k.-',
    label='full_covar')
pylab.plot(
    diag_info_dict['lap_history'],
    diag_info_dict['loss_history'], 'r.-',
    label='diag_covar')
pylab.plot(
    stoch_info_dict['lap_history'],
    stoch_info_dict['loss_history'], 'c.:',
    label='full_covar stochastic')
pylab.legend(loc='upper right')
pylab.xlabel('num. laps')
pylab.ylabel('loss')
pylab.xlim([4, 100]) # avoid early iterations
pylab.ylim([2.34, 4.0]) # handpicked
pylab.draw()
pylab.tight_layout()
```



Total running time of the script: (0 minutes 42.505 seconds)

`Download Python source code: plot-01-demo=vb_algs-model=mix_gauss.py`

`Download Jupyter notebook: plot-01-demo=vb_algs-model=mix_gauss.ipynb`

Generated by Sphinx-Gallery

5.2.4 Toy bars dataset: one bar per doc

Variational training of mixture models and topic models with Multinomial likelihoods on simple toy "bars" dataset.

01: Standard variational training for mixture model

How to train a mixture of multinomials.

```
import bnpy
import numpy as np
import os
```

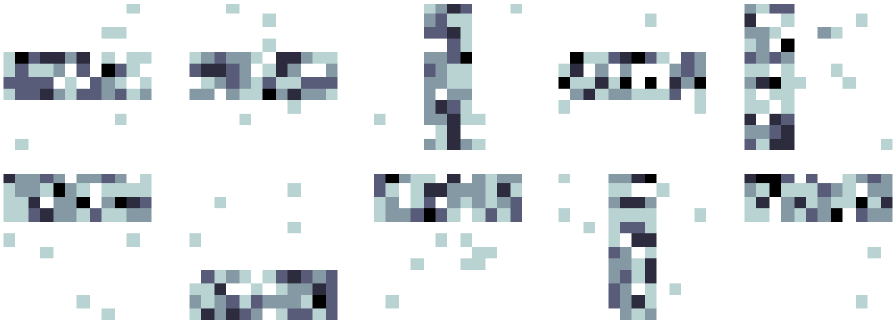
```
from matplotlib import pylab
import seaborn as sns

FIG_SIZE = (3, 3)
SMALL_FIG_SIZE = (1,1)
pylab.rcParams['figure.figsize'] = FIG_SIZE
```

Read toy “bars” dataset from file.

```
dataset_path = os.path.join(bnpy.DATASET_PATH, 'bars_one_per_doc')
dataset = bnpy.data.BagOfWordsData.read_npz(
    os.path.join(dataset_path, 'dataset.npz'))
```

Make a simple plot of the raw data

```
X_csr_DV = dataset.getSparseDocTypeCountMatrix()
bnpy.viz.BarsViz.show_square_images(
    X_csr_DV[:10].toarray(), vmin=0, vmax=5)
#pylab.colorbar()
#pylab.clabel('word count')
pylab.tight_layout()
```



Let’s do one single run of the VB algorithm.

Using 10 clusters and the ‘randexamples’ initializatio procedure.

```
trained_model, info_dict = bnpy.run(
    dataset, 'FiniteMixtureModel', 'Mult', 'VB',
    output_path='/tmp/bars_one_per_doc/helloworld-K=10/',
    nLap=1000, convergeThr=0.0001,
    K=10, initname='randomlikewang',
    gamma0=50.0, lam=0.1)
```

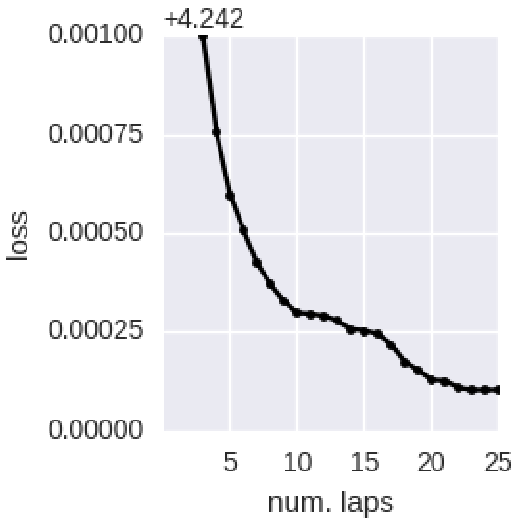
Out:

```
WARNING: Found unrecognized keyword args. These are ignored.
  --gamma0
Dataset Summary:
BagOfWordsData
  size: 2000 units (documents)
  vocab size: 144
  min     5%    50%    95%    max
```

```
   38    42    46    51    57  nUniqueTokensPerDoc
  100   100   100   100   100  nTotalTokensPerDoc
Hist of word_count across tokens
      1       2       3    <10    <100  >=100
   0.38    0.29    0.19    0.14       0       0
Hist of unique docs per word type
     <1     <10    <100   <0.10   <0.20   <0.50 >=0.50
      0       0       0       0       0    >.99       0
Allocation Model:  Finite mixture model. Dir prior param 1.00
Obs. Data  Model:  Multinomial over finite vocabulary.
Obs. Data  Prior:  Dirichlet over finite vocabulary
  lam = [ 0.1  0.1] ...
Initialization:
  initname = randomlikewang
  K = 10 (number of clusters)
  seed = 1607680
  elapsed_time: 0.0 sec
Learn Alg: VB | task  1/1 | alg. seed: 1607680 | data order seed: 8541952
task_output_path: /tmp/bars_one_per_doc/helloworld-K=10/1
        1/1000 after      0 sec. |    151.2 MiB | K   10 | loss  4.711444785e+00 |
        2/1000 after      0 sec. |    151.2 MiB | K   10 | loss  4.244123622e+00 | Ndiff  2
        3/1000 after      0 sec. |    151.2 MiB | K   10 | loss  4.242999256e+00 | Ndiff
        4/1000 after      0 sec. |    151.2 MiB | K   10 | loss  4.242755538e+00 | Ndiff
        5/1000 after      0 sec. |    151.2 MiB | K   10 | loss  4.242599251e+00 | Ndiff
        6/1000 after      0 sec. |    151.2 MiB | K   10 | loss  4.242508901e+00 | Ndiff
        7/1000 after      0 sec. |    151.2 MiB | K   10 | loss  4.242425981e+00 | Ndiff
        8/1000 after      0 sec. |    151.2 MiB | K   10 | loss  4.242371327e+00 | Ndiff
        9/1000 after      0 sec. |    151.2 MiB | K   10 | loss  4.242328843e+00 | Ndiff
       10/1000 after      0 sec. |    151.2 MiB | K   10 | loss  4.242298419e+00 | Ndiff
       11/1000 after      0 sec. |    151.2 MiB | K   10 | loss  4.242296009e+00 | Ndiff
       12/1000 after      0 sec. |    151.2 MiB | K   10 | loss  4.242290695e+00 | Ndiff
       13/1000 after      0 sec. |    151.2 MiB | K   10 | loss  4.242278821e+00 | Ndiff
       14/1000 after      0 sec. |    151.2 MiB | K   10 | loss  4.242257185e+00 | Ndiff
       15/1000 after      0 sec. |    151.2 MiB | K   10 | loss  4.242254419e+00 | Ndiff
       16/1000 after      0 sec. |    151.2 MiB | K   10 | loss  4.242244922e+00 | Ndiff
       17/1000 after      0 sec. |    151.2 MiB | K   10 | loss  4.242219919e+00 | Ndiff
       18/1000 after      0 sec. |    151.2 MiB | K   10 | loss  4.242175380e+00 | Ndiff
       19/1000 after      0 sec. |    151.2 MiB | K   10 | loss  4.242154460e+00 | Ndiff
       20/1000 after      0 sec. |    151.2 MiB | K   10 | loss  4.242128894e+00 | Ndiff
       21/1000 after      0 sec. |    151.2 MiB | K   10 | loss  4.242126836e+00 | Ndiff
       22/1000 after      0 sec. |    151.2 MiB | K   10 | loss  4.242109815e+00 | Ndiff
       23/1000 after      0 sec. |    151.2 MiB | K   10 | loss  4.242103791e+00 | Ndiff
       24/1000 after      0 sec. |    151.2 MiB | K   10 | loss  4.242103790e+00 | Ndiff
       25/1000 after      0 sec. |    151.2 MiB | K   10 | loss  4.242103790e+00 | Ndiff
... done. converged.
```

First, we can plot the loss function over time We'll skip the first few iterations, since performance is quite bad.

```
pylab.figure(figsize=FIG_SIZE)
pylab.plot(info_dict['lap_history'][2:], info_dict['loss_history'][2:], 'k.-')
pylab.xlabel('num. laps')
pylab.ylabel('loss')
pylab.tight_layout()
```

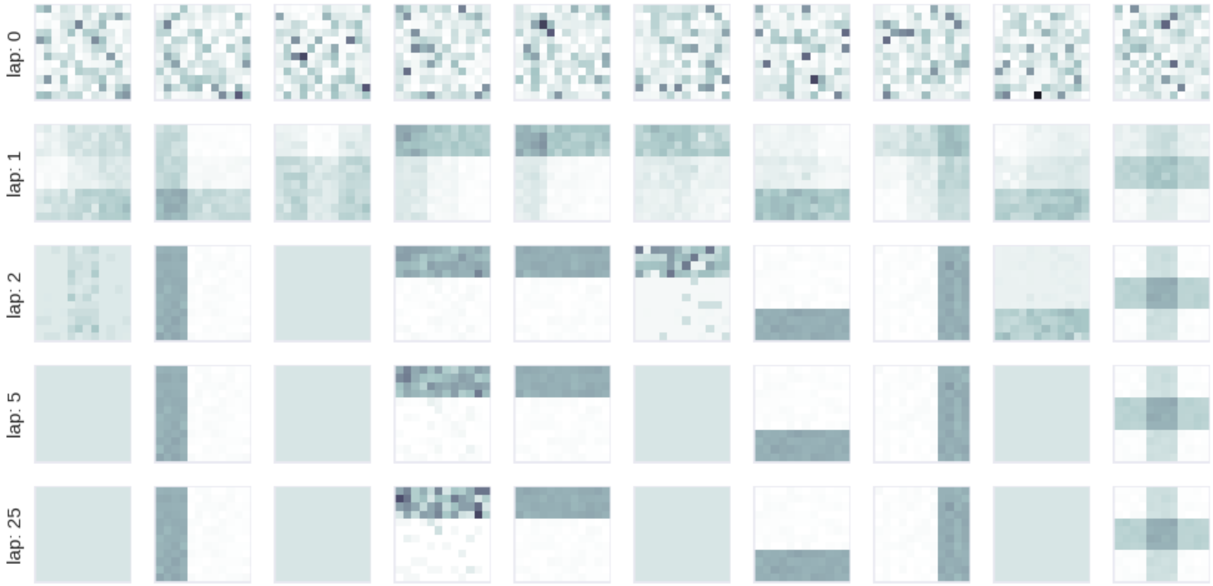


Setup: Useful function to display learned bar structure over time.

```
def show_bars_over_time(
        task_output_path=None,
        query_laps=[0, 1, 2, 5, None],
        ncols=10):
    '''
    '''
    nrows = len(query_laps)
    fig_handle, ax_handles_RC = pylab.subplots(
        figsize=(SMALL_FIG_SIZE[0] * ncols, SMALL_FIG_SIZE[1] * nrows),
        nrows=nrows, ncols=ncols, sharex=True, sharey=True)
    for row_id, lap_val in enumerate(query_laps):
        cur_model, lap_val = bnpy.load_model_at_lap(task_output_path, lap_val)
        cur_topics_KV = cur_model.obsModel.getTopics()
        # Plot the current model
        cur_ax_list = ax_handles_RC[row_id].flatten().tolist()
        bnpy.viz.BarsViz.show_square_images(
            cur_topics_KV,
            vmin=0.0, vmax=0.06,
            ax_list=cur_ax_list)
        cur_ax_list[0].set_ylabel("lap: %d" % lap_val)
    pylab.tight_layout()
```

Show the clusters over time

```
show_bars_over_time(info_dict['task_output_path'])
```



Total running time of the script: (0 minutes 5.443 seconds)

`Download Python source code: plot-01-demo=vb_single_run-model=mix+mult.py`

`Download Jupyter notebook: plot-01-demo=vb_single_run-model=mix+mult.ipynb`

Generated by Sphinx-Gallery

05: Training for HDP Topic Model with birth and merge proposals

```
import bnpy
import numpy as np
import os
import sys

from matplotlib import pylab
import seaborn as sns

FIG_SIZE = (3, 3)
SMALL_FIG_SIZE = (1,1)
pylab.rcParams['figure.figsize'] = FIG_SIZE
```

Read dataset from file.

```
dataset_path = os.path.join(bnpy.DATASET_PATH, 'bars_one_per_doc')
dataset = bnpy.data.BagOfWordsData.read_npz(
    os.path.join(dataset_path, 'dataset.npz'))
```

Set the local step algorithmic keyword args

```
local_step_kwargs = dict(
    # Perform at most this many iterations at each document
    nCoordAscentItersLP=100,
    # Stop local iters early when max change in doc-topic counts < this thr
    convThrLP=0.01,
    restartLP=0,
    doMemoizeLocalParams=0,
    )

merge_kwargs = dict(
    m_startLap=5,
    m_pair_ranking_procedure='total_size',
    m_pair_ranking_direction='descending',
    )

birth_kwargs = dict(
    b_startLap=2,
    b_stopLap=6,
    b_Kfresh=3,
    b_nRefineSteps=5,
    )
```

Setup: Helper function to plot bars at each stage of training

```
def show_bars_over_time(
        task_output_path=None,
        query_laps=[0, 1, 2, 5, None],
        ncols=10):
    '''
    '''
    nrows = len(query_laps)
    fig_handle, ax_handles_RC = pylab.subplots(
        figsize=(SMALL_FIG_SIZE[0] * ncols, SMALL_FIG_SIZE[1] * nrows),
        nrows=nrows, ncols=ncols, sharex=True, sharey=True)
    for row_id, lap_val in enumerate(query_laps):
        cur_model, lap_val = bnpy.load_model_at_lap(task_output_path, lap_val)
        cur_topics_KV = cur_model.obsModel.getTopics()
        # Plot the current model
        cur_ax_list = ax_handles_RC[row_id].flatten().tolist()
        bnpy.viz.BarsViz.show_square_images(
            cur_topics_KV,
            vmin=0.0, vmax=0.06,
            ax_list=cur_ax_list)
        cur_ax_list[0].set_ylabel("lap: %d" % lap_val)
    pylab.tight_layout()
```

Training from K=3 with births

Initialization: 3 topics, using random initial guess

```
trained_model, info_dict = bnpy.run(
    dataset, 'HDPTopicModel', 'Mult', 'memoVB',
    output_path='/tmp/bars_one_per_doc/trymoves-model=hdp+mult-K=3-moves=birth,merge,shuffl
    nLap=20, convergeThr=0.001, nBatch=1,
    K=3, initname='randomlikewang',
    alpha=0.5, lam=0.1,
    moves='birth,merge,shuffle',
    **dict(merge_kwargs.items() +
           local_step_kwargs.items() +
           birth_kwargs.items()))

show_bars_over_time(info_dict['task_output_path'])
```

Total running time of the script: (0 minutes 0.000 seconds)

`Download Python source code: run-05-demo=topic_model_vb+births-model=hdp_topic+mul`

`Download Jupyter notebook: run-05-demo=topic_model_vb+births-model=hdp_topic+mult.`

Generated by Sphinx-Gallery

04: Training HDP Topic Model with merge proposals

```
import bnpy
import numpy as np
import os
import sys

from matplotlib import pylab
import seaborn as sns

FIG_SIZE = (3, 3)
SMALL_FIG_SIZE = (1,1)
pylab.rcParams['figure.figsize'] = FIG_SIZE
```

Read dataset from file.

```
dataset_path = os.path.join(bnpy.DATASET_PATH, 'bars_one_per_doc')
dataset = bnpy.data.BagOfWordsData.read_npz(
    os.path.join(dataset_path, 'dataset.npz'))
```

Set the local step algorithmic keyword args

```
local_step_kwargs = dict(
    # Perform at most this many iterations at each document
    nCoordAscentItersLP=100,
    # Stop local iters early when max change in doc-topic counts < this thr
    convThrLP=0.001,
    restartLP=0,
    doMemoizeLocalParams=0,
    )

merge_kwargs = dict(
    m_startLap=5,
    m_pair_ranking_procedure='total_size',
    m_pair_ranking_direction='descending',
    )
```

Run the VB+proposals algorithm with only merges and re-shuffling.

Initialization: 10 topics, using randomlikewang

```
trained_model, info_dict = bnpy.run(
    dataset, 'HDPTopicModel', 'Mult', 'memoVB',
    output_path=
        '/tmp/bars_one_per_doc/' +
        'trymoves-model=hdp+mult-K=10-moves=merge,shuffle/',
    nLap=50, convergeThr=0.001, nBatch=1,
    K=10, initname='randomlikewang',
    alpha=0.5, lam=0.1,
    moves='merge,shuffle',
    **dict(merge_kwargs.items() + local_step_kwargs.items()))
```

```
def show_bars_over_time(
        task_output_path=None,
        query_laps=[0, 1, 2, 5, None],
        ncols=10):
    ''' Show square-image visualization of estimated topics over time.

    Post Condition
    --------------
    New matplotlib figure with visualization (one row per lap).
    '''
    nrows = len(query_laps)
    fig_handle, ax_handles_RC = pylab.subplots(
        figsize=(SMALL_FIG_SIZE[0] * ncols, SMALL_FIG_SIZE[1] * nrows),
        nrows=nrows, ncols=ncols, sharex=True, sharey=True)
    for row_id, lap_val in enumerate(query_laps):
        cur_model, lap_val = bnpy.load_model_at_lap(task_output_path, lap_val)
        cur_topics_KV = cur_model.obsModel.getTopics()
        # Plot the current model
        cur_ax_list = ax_handles_RC[row_id].flatten().tolist()
        bnpy.viz.BarsViz.show_square_images(
            cur_topics_KV,
            vmin=0.0, vmax=0.06,
            ax_list=cur_ax_list)
        cur_ax_list[0].set_ylabel("lap: %d" % lap_val)
    pylab.tight_layout()
```

Examine the bars over time

```
show_bars_over_time(info_dict['task_output_path'])
```

Total running time of the script: (0 minutes 0.000 seconds)

`Download Python source code: run-04-demo=topic_model_vb+merges-model=hdp_topic+mul`

`Download Jupyter notebook: run-04-demo=topic_model_vb+merges-model=hdp_topic+mult.`

Generated by Sphinx-Gallery

02: Training DP mixture model with birth and merge proposals

How to train a DP mixture of multinomials.

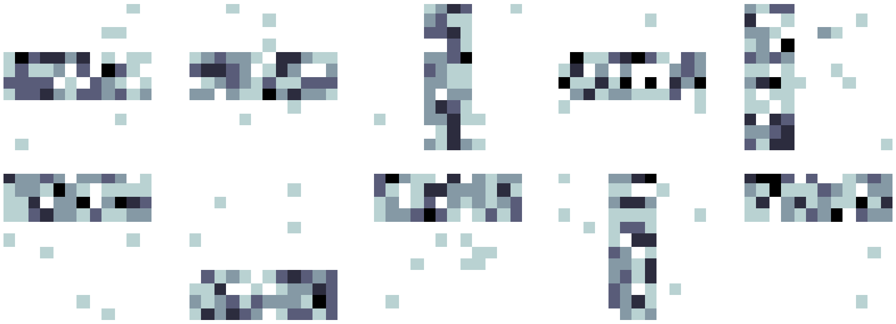
```
import bnpy
import numpy as np
import os

from matplotlib import pylab
import seaborn as sns

FIG_SIZE = (3, 3)
SMALL_FIG_SIZE = (1.5, 1.5)
pylab.rcParams['figure.figsize'] = FIG_SIZE
```

Read dataset from file.

```
dataset_path = os.path.join(bnpy.DATASET_PATH, 'bars_one_per_doc')
dataset = bnpy.data.BagOfWordsData.read_npz(
    os.path.join(dataset_path, 'dataset.npz'))
```

Make a simple plot of the raw data

```
X_csr_DV = dataset.getSparseDocTypeCountMatrix()
bnpy.viz.BarsViz.show_square_images(
    X_csr_DV[:10].toarray(), vmin=0, vmax=5)
pylab.tight_layout()
```



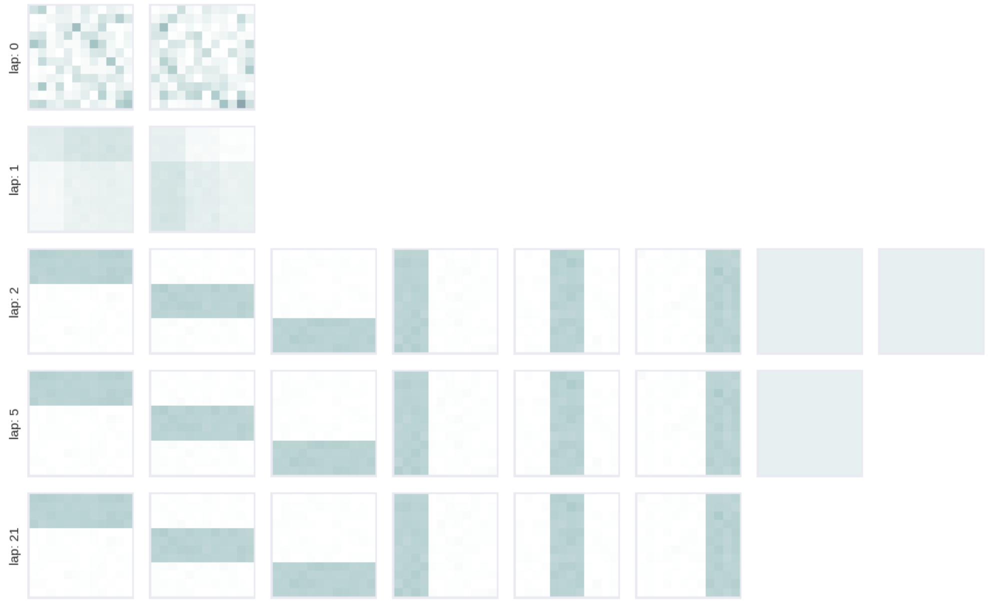
Setup: Function to show bars from start to end of training run

```
def show_bars_over_time(
        task_output_path=None,
        query_laps=[0, 1, 2, 5, None],
        ncols=10):
    '''
    '''
    nrows = len(query_laps)
    fig_handle, ax_handles_RC = pylab.subplots(
        figsize=(SMALL_FIG_SIZE[0] * ncols, SMALL_FIG_SIZE[1] * nrows),
        nrows=nrows, ncols=ncols, sharex=True, sharey=True)
    for row_id, lap_val in enumerate(query_laps):
        cur_model, lap_val = bnpy.load_model_at_lap(task_output_path, lap_val)
        cur_topics_KV = cur_model.obsModel.getTopics()
        # Plot the current model
        cur_ax_list = ax_handles_RC[row_id].flatten().tolist()
        bnpy.viz.BarsViz.show_square_images(
            cur_topics_KV,
            vmin=0.0, vmax=0.1,
            ax_list=cur_ax_list)
        cur_ax_list[0].set_ylabel("lap: %d" % lap_val)
    pylab.tight_layout()
```

From K=2 initial clusters

Using random initialization

```
initname = 'randomlikewang'
K = 2

K2_trained_model, K2_info_dict = bnpy.run(
    dataset, 'DPMixtureModel', 'Mult', 'memoVB',
    output_path='/tmp/bars_one_per_doc/trymoves-K=%d-initname=%s/' % (
        K, initname),
    nTask=1, nLap=50, convergeThr=0.0001, nBatch=1,
    K=K, initname=initname,
    gamma0=50.0, lam=0.1,
    moves='birth,merge,shuffle,delete',
    b_startLap=2,
    m_startLap=5,
    d_startLap=10)

show_bars_over_time(K2_info_dict['task_output_path'])
```



Out:

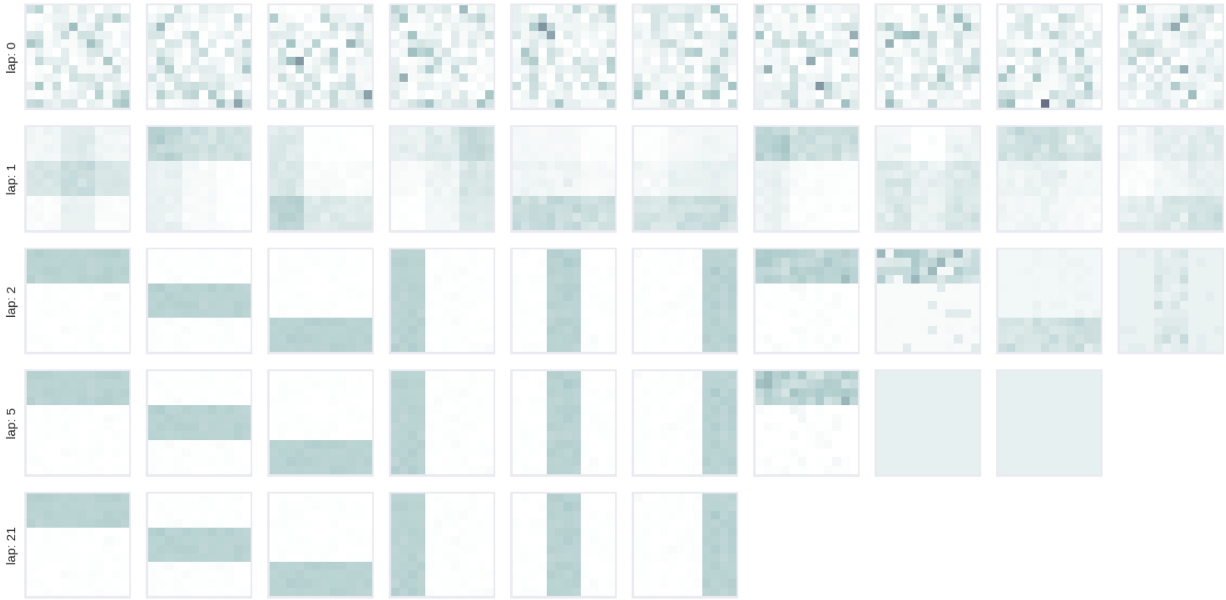
```
Dataset Summary:
BagOfWordsData
  total size: 2000 units
  batch size: 2000 units
  num. batches: 1
Allocation Model:  DP mixture with K=0. Concentration gamma0= 50.00
Obs. Data  Model:  Multinomial over finite vocabulary.
Obs. Data  Prior:  Dirichlet over finite vocabulary
  lam = [ 0.1  0.1] ...
Initialization:
  initname = randomlikewang
  K = 2 (number of clusters)
  seed = 1607680
  elapsed_time: 0.0 sec
Learn Alg: memoVB | task  1/1 | alg. seed: 1607680 | data order seed: 8541952
task_output_path: /tmp/bars_one_per_doc/trymoves-K=2-initname=randomlikewang/1
BIRTH @ lap 1.00: Disabled. Waiting for lap >= 2 (--b_startLap).
MERGE @ lap 1.00: Disabled. Cannot plan merge on first lap. Need valid SS that represent wh
DELETE @ lap 1.00: Disabled. Cannot delete before first complete lap, because SS that repre
    1.000/50 after      0 sec. |    166.4 MiB | K    2 | loss  4.852914927e+00 |
MERGE @ lap 2.00: Disabled. Waiting for lap >= 5 (--m_startLap).
DELETE @ lap 2.00: Disabled. Waiting for lap >= 10 (--d_startLap).
BIRTH @ lap 2.00 : Added 6 states. 2/2 succeeded. 0/2 failed eval phase. 0/2 failed build p
    2.000/50 after      0 sec. |    166.4 MiB | K    8 | loss  4.125791748e+00 |
MERGE @ lap 3.00: Disabled. Waiting for lap >= 5 (--m_startLap).
DELETE @ lap 3.00: Disabled. Waiting for lap >= 10 (--d_startLap).
BIRTH @ lap 3.00 : Added 0 states. 0/6 succeeded. 0/6 failed eval phase. 6/6 failed build p
    3.000/50 after      1 sec. |    166.4 MiB | K    8 | loss  4.125791567e+00 | Ndiff    0
MERGE @ lap 4.00: Disabled. Waiting for lap >= 5 (--m_startLap).
DELETE @ lap 4.00: Disabled. Waiting for lap >= 10 (--d_startLap).
BIRTH @ lap 4.000 : None attempted. 6 past failures. 2 too small. 0 too busy.
    4.000/50 after      1 sec. |    166.4 MiB | K    8 | loss  4.125791567e+00 | Ndiff    0
DELETE @ lap 5.00: Disabled. Waiting for lap >= 10 (--d_startLap).
BIRTH @ lap 5.000 : None attempted. 0 past failures. 0 too small. 8 too busy.
MERGE @ lap 5.00 : 1/16 accepted. Ndiff 0.00. 0 skipped.
    5.000/50 after      1 sec. |    166.4 MiB | K    7 | loss  4.125791567e+00 | Ndiff    0
DELETE @ lap 6.00: Disabled. Waiting for lap >= 10 (--d_startLap).
BIRTH @ lap 6.000 : None attempted. 1 past failures. 0 too small. 6 too busy.
MERGE @ lap 6.00 : 1/5 accepted. Ndiff 0.00. 0 skipped.
    6.000/50 after      1 sec. |    166.4 MiB | K    6 | loss  4.125791567e+00 | Ndiff    0
DELETE @ lap 7.00: Disabled. Waiting for lap >= 10 (--d_startLap).
BIRTH @ lap 7.000 : None attempted. 0 past failures. 0 too small. 6 too busy.
MERGE @ lap 7.00 : 0/5 accepted. Ndiff 0.00. 0 skipped.
    7.000/50 after      1 sec. |    166.4 MiB | K    6 | loss  4.125791567e+00 | Ndiff    0
MERGE @ lap 8.00: No promising candidates, so no attempts.
DELETE @ lap 8.00: Disabled. Waiting for lap >= 10 (--d_startLap).
BIRTH @ lap 8.00 : Added 0 states. 0/1 succeeded. 0/1 failed eval phase. 1/1 failed build p
    8.000/50 after      2 sec. |    166.4 MiB | K    6 | loss  4.125791567e+00 | Ndiff    0
MERGE @ lap 9.00: No promising candidates, so no attempts.
DELETE @ lap 9.00: Disabled. Waiting for lap >= 10 (--d_startLap).
BIRTH @ lap 9.000 : None attempted. 6 past failures. 0 too small. 0 too busy.
    9.000/50 after      2 sec. |    166.4 MiB | K    6 | loss  4.125791567e+00 | Ndiff    0
BIRTH @ lap 10.000 : None attempted. 0 past failures. 0 too small. 6 too busy.
```

```
DELETE @ lap 10.00: 0/1 accepted. Ndiff 0.00.
MERGE @ lap 10.00 : 0/10 accepted. Ndiff 0.00. 0 skipped.
   10.000/50 after      2 sec. |    166.4 MiB | K    6 | loss  4.125791567e+00 | Ndiff
MERGE @ lap 11.00: No promising candidates, so no attempts.
BIRTH @ lap 11.000 : None attempted. 0 past failures. 0 too small. 6 too busy.
DELETE @ lap 11.00: 0/1 accepted. Ndiff 0.00.
   11.000/50 after      2 sec. |    166.4 MiB | K    6 | loss  4.125791567e+00 | Ndiff
BIRTH @ lap 12.000 : None attempted. 0 past failures. 0 too small. 6 too busy.
DELETE @ lap 12.00: 0/1 accepted. Ndiff 0.00.
MERGE @ lap 12.00 : 0/5 accepted. Ndiff 0.00. 0 skipped.
   12.000/50 after      2 sec. |    166.4 MiB | K    6 | loss  4.125791567e+00 | Ndiff
MERGE @ lap 13.00: No promising candidates, so no attempts.
BIRTH @ lap 13.000 : None attempted. 0 past failures. 0 too small. 6 too busy.
DELETE @ lap 13.00: 0/1 accepted. Ndiff 0.00.
   13.000/50 after      2 sec. |    166.4 MiB | K    6 | loss  4.125791567e+00 | Ndiff
MERGE @ lap 14.00: No promising candidates, so no attempts.
BIRTH @ lap 14.000 : None attempted. 0 past failures. 0 too small. 6 too busy.
DELETE @ lap 14.00: 0/1 accepted. Ndiff 0.00.
   14.000/50 after      2 sec. |    166.4 MiB | K    6 | loss  4.125791567e+00 | Ndiff
BIRTH @ lap 15.000 : None attempted. 0 past failures. 0 too small. 6 too busy.
DELETE @ lap 15.00: 0/1 accepted. Ndiff 0.00.
MERGE @ lap 15.00 : 0/10 accepted. Ndiff 0.00. 0 skipped.
   15.000/50 after      2 sec. |    166.4 MiB | K    6 | loss  4.125791567e+00 | Ndiff
MERGE @ lap 16.00: No promising candidates, so no attempts.
BIRTH @ lap 16.000 : None attempted. 0 past failures. 0 too small. 6 too busy.
DELETE @ lap 16.00: 0/1 accepted. Ndiff 0.00.
   16.000/50 after      2 sec. |    166.4 MiB | K    6 | loss  4.125791567e+00 | Ndiff
BIRTH @ lap 17.000 : None attempted. 0 past failures. 0 too small. 6 too busy.
DELETE @ lap 17.00: 0/1 accepted. Ndiff 0.00.
MERGE @ lap 17.00 : 0/5 accepted. Ndiff 0.00. 0 skipped.
   17.000/50 after      3 sec. |    166.4 MiB | K    6 | loss  4.125791567e+00 | Ndiff
MERGE @ lap 18.00: No promising candidates, so no attempts.
BIRTH @ lap 18.000 : None attempted. 0 past failures. 0 too small. 6 too busy.
DELETE @ lap 18.00: 0/1 accepted. Ndiff 0.00.
   18.000/50 after      3 sec. |    166.4 MiB | K    6 | loss  4.125791567e+00 | Ndiff
MERGE @ lap 19.00: No promising candidates, so no attempts.
BIRTH @ lap 19.000 : None attempted. 0 past failures. 0 too small. 6 too busy.
DELETE @ lap 19.00: 0/1 accepted. Ndiff 0.00.
   19.000/50 after      3 sec. |    166.4 MiB | K    6 | loss  4.125791567e+00 | Ndiff
BIRTH @ lap 20.000 : None attempted. 0 past failures. 0 too small. 6 too busy.
DELETE @ lap 20.00: 0/1 accepted. Ndiff 0.00.
MERGE @ lap 20.00 : 0/10 accepted. Ndiff 0.00. 0 skipped.
   20.000/50 after      3 sec. |    166.4 MiB | K    6 | loss  4.125791567e+00 | Ndiff
MERGE @ lap 21.00: No promising candidates, so no attempts.
BIRTH @ lap 21.000 : None attempted. 0 past failures. 0 too small. 6 too busy.
DELETE @ lap 21.00: 0/1 accepted. Ndiff 0.00.
   21.000/50 after      3 sec. |    166.4 MiB | K    6 | loss  4.125791567e+00 | Ndiff
... done. converged.
```

K=10 initial clusters

Using random initialization

```
initname = 'randomlikewang'
K = 10

K10_trained_model, K10_info_dict = bnpy.run(
    dataset, 'DPMixtureModel', 'Mult', 'memoVB',
    output_path='/tmp/bars_one_per_doc/trymoves-K=%d-initname=%s/' % (
        K, initname),
    nTask=1, nLap=50, convergeThr=0.0001, nBatch=1,
    K=K, initname=initname,
    gamma0=50.0, lam=0.1,
    moves='birth,merge,shuffle,delete',
    b_startLap=2,
    m_startLap=5,
    d_startLap=10)

show_bars_over_time(K10_info_dict['task_output_path'])
```



Out:

```
Dataset Summary:
BagOfWordsData
  total size: 2000 units
  batch size: 2000 units
  num. batches: 1
Allocation Model:  DP mixture with K=0. Concentration gamma0= 50.00
Obs. Data  Model:  Multinomial over finite vocabulary.
Obs. Data  Prior:  Dirichlet over finite vocabulary
  lam = [ 0.1  0.1] ...
Initialization:
  initname = randomlikewang
  K = 10 (number of clusters)
  seed = 1607680
  elapsed_time: 0.0 sec
```

```
Learn Alg: memoVB | task  1/1 | alg. seed: 1607680 | data order seed: 8541952
task_output_path: /tmp/bars_one_per_doc/trymoves-K=10-initname=randomlikewang/1
BIRTH @ lap 1.00: Disabled. Waiting for lap >= 2 (--b_startLap).
MERGE @ lap 1.00: Disabled. Cannot plan merge on first lap. Need valid SS that represent wh
DELETE @ lap 1.00: Disabled. Cannot delete before first complete lap, because SS that repre
    1.000/50 after      0 sec. |    173.4 MiB | K   10 | loss  4.712365207e+00 |
MERGE @ lap 2.00: Disabled. Waiting for lap >= 5 (--m_startLap).
DELETE @ lap 2.00: Disabled. Waiting for lap >= 10 (--d_startLap).
BIRTH @ lap 2.00 : Added 2 states. 1/6 succeeded. 0/6 failed eval phase. 5/6 failed build p
    2.000/50 after      1 sec. |    174.0 MiB | K   12 | loss  4.128419662e+00 |
MERGE @ lap 3.00: Disabled. Waiting for lap >= 5 (--m_startLap).
DELETE @ lap 3.00: Disabled. Waiting for lap >= 10 (--d_startLap).
BIRTH @ lap 3.00 : Added 0 states. 0/4 succeeded. 0/4 failed eval phase. 4/4 failed build p
    3.000/50 after      1 sec. |    174.0 MiB | K   12 | loss  4.127318928e+00 | Ndiff   17
MERGE @ lap 4.00: Disabled. Waiting for lap >= 5 (--m_startLap).
DELETE @ lap 4.00: Disabled. Waiting for lap >= 10 (--d_startLap).
BIRTH @ lap 4.00 : Added 0 states. 0/2 succeeded. 0/2 failed eval phase. 2/2 failed build p
    4.000/50 after      1 sec. |    174.0 MiB | K   12 | loss  4.127075221e+00 | Ndiff   11
DELETE @ lap 5.00: Disabled. Waiting for lap >= 10 (--d_startLap).
BIRTH @ lap 5.000 : None attempted. 0 past failures. 0 too small. 12 too busy.
MERGE @ lap 5.00 : 3/18 accepted. Ndiff 0.00. 12 skipped.
    5.000/50 after      2 sec. |    174.0 MiB | K    9 | loss  4.126920998e+00 | Ndiff   11
DELETE @ lap 6.00: Disabled. Waiting for lap >= 10 (--d_startLap).
BIRTH @ lap 6.000 : None attempted. 1 past failures. 0 too small. 8 too busy.
MERGE @ lap 6.00 : 3/5 accepted. Ndiff 21.29. 10 skipped.
    6.000/50 after      2 sec. |    174.0 MiB | K    6 | loss  4.125791567e+00 | Ndiff   11
DELETE @ lap 7.00: Disabled. Waiting for lap >= 10 (--d_startLap).
BIRTH @ lap 7.000 : None attempted. 0 past failures. 0 too small. 6 too busy.
MERGE @ lap 7.00 : 0/9 accepted. Ndiff 0.00. 0 skipped.
    7.000/50 after      2 sec. |    174.0 MiB | K    6 | loss  4.125791567e+00 | Ndiff    0
MERGE @ lap 8.00: No promising candidates, so no attempts.
DELETE @ lap 8.00: Disabled. Waiting for lap >= 10 (--d_startLap).
BIRTH @ lap 8.00 : Added 0 states. 0/2 succeeded. 0/2 failed eval phase. 2/2 failed build p
    8.000/50 after      2 sec. |    174.0 MiB | K    6 | loss  4.125791567e+00 | Ndiff    0
MERGE @ lap 9.00: No promising candidates, so no attempts.
DELETE @ lap 9.00: Disabled. Waiting for lap >= 10 (--d_startLap).
BIRTH @ lap 9.000 : None attempted. 6 past failures. 0 too small. 0 too busy.
    9.000/50 after      2 sec. |    174.0 MiB | K    6 | loss  4.125791567e+00 | Ndiff    0
BIRTH @ lap 10.000 : None attempted. 0 past failures. 0 too small. 6 too busy.
DELETE @ lap 10.00: 0/1 accepted. Ndiff 0.00.
MERGE @ lap 10.00 : 0/6 accepted. Ndiff 0.00. 0 skipped.
   10.000/50 after      2 sec. |    174.0 MiB | K    6 | loss  4.125791567e+00 | Ndiff    0
MERGE @ lap 11.00: No promising candidates, so no attempts.
BIRTH @ lap 11.000 : None attempted. 0 past failures. 0 too small. 6 too busy.
DELETE @ lap 11.00: 0/1 accepted. Ndiff 0.00.
   11.000/50 after      2 sec. |    174.0 MiB | K    6 | loss  4.125791567e+00 | Ndiff    0
BIRTH @ lap 12.000 : None attempted. 0 past failures. 0 too small. 6 too busy.
DELETE @ lap 12.00: 0/1 accepted. Ndiff 0.00.
MERGE @ lap 12.00 : 0/9 accepted. Ndiff 0.00. 0 skipped.
   12.000/50 after      3 sec. |    174.0 MiB | K    6 | loss  4.125791567e+00 | Ndiff    0
MERGE @ lap 13.00: No promising candidates, so no attempts.
BIRTH @ lap 13.000 : None attempted. 0 past failures. 0 too small. 6 too busy.
DELETE @ lap 13.00: 0/1 accepted. Ndiff 0.00.
```

```
   13.000/50 after      3 sec. |    174.0 MiB | K    6 | loss  4.125791567e+00 | Ndiff
MERGE @ lap 14.00: No promising candidates, so no attempts.
BIRTH @ lap 14.000 : None attempted. 0 past failures. 0 too small. 6 too busy.
DELETE @ lap 14.00: 0/1 accepted. Ndiff 0.00.
   14.000/50 after      3 sec. |    174.0 MiB | K    6 | loss  4.125791567e+00 | Ndiff
BIRTH @ lap 15.000 : None attempted. 0 past failures. 0 too small. 6 too busy.
DELETE @ lap 15.00: 0/1 accepted. Ndiff 0.00.
MERGE @ lap 15.00 : 0/6 accepted. Ndiff 0.00. 0 skipped.
   15.000/50 after      3 sec. |    174.0 MiB | K    6 | loss  4.125791567e+00 | Ndiff
MERGE @ lap 16.00: No promising candidates, so no attempts.
BIRTH @ lap 16.000 : None attempted. 0 past failures. 0 too small. 6 too busy.
DELETE @ lap 16.00: 0/1 accepted. Ndiff 0.00.
   16.000/50 after      3 sec. |    174.0 MiB | K    6 | loss  4.125791567e+00 | Ndiff
BIRTH @ lap 17.000 : None attempted. 0 past failures. 0 too small. 6 too busy.
DELETE @ lap 17.00: 0/1 accepted. Ndiff 0.00.
MERGE @ lap 17.00 : 0/9 accepted. Ndiff 0.00. 0 skipped.
   17.000/50 after      3 sec. |    174.0 MiB | K    6 | loss  4.125791567e+00 | Ndiff
MERGE @ lap 18.00: No promising candidates, so no attempts.
BIRTH @ lap 18.000 : None attempted. 0 past failures. 0 too small. 6 too busy.
DELETE @ lap 18.00: 0/1 accepted. Ndiff 0.00.
   18.000/50 after      3 sec. |    174.0 MiB | K    6 | loss  4.125791567e+00 | Ndiff
MERGE @ lap 19.00: No promising candidates, so no attempts.
BIRTH @ lap 19.000 : None attempted. 0 past failures. 0 too small. 6 too busy.
DELETE @ lap 19.00: 0/1 accepted. Ndiff 0.00.
   19.000/50 after      3 sec. |    174.0 MiB | K    6 | loss  4.125791567e+00 | Ndiff
BIRTH @ lap 20.000 : None attempted. 0 past failures. 0 too small. 6 too busy.
DELETE @ lap 20.00: 0/1 accepted. Ndiff 0.00.
MERGE @ lap 20.00 : 0/6 accepted. Ndiff 0.00. 0 skipped.
   20.000/50 after      3 sec. |    174.0 MiB | K    6 | loss  4.125791567e+00 | Ndiff
MERGE @ lap 21.00: No promising candidates, so no attempts.
BIRTH @ lap 21.000 : None attempted. 0 past failures. 0 too small. 6 too busy.
DELETE @ lap 21.00: 0/1 accepted. Ndiff 0.00.
   21.000/50 after      3 sec. |    174.0 MiB | K    6 | loss  4.125791567e+00 | Ndiff
... done. converged.
```

Total running time of the script: (0 minutes 15.776 seconds)

Download Python source code: plot-02-demo=vb+proposals-model=dp_mix+mult.py

Download Jupyter notebook: plot-02-demo=vb+proposals-model=dp_mix+mult.ipynb

Generated by Sphinx-Gallery

03: Standard variational training for topic model

```
import bnpy
import numpy as np
import os

from matplotlib import pylab
import seaborn as sns

FIG_SIZE = (3, 3)
```

```
SMALL_FIG_SIZE = (1,1)
pylab.rcParams['figure.figsize'] = FIG_SIZE
```

Read dataset from file.

```
dataset_path = os.path.join(bnpy.DATASET_PATH, 'bars_one_per_doc')
dataset = bnpy.data.BagOfWordsData.read_npz(
    os.path.join(dataset_path, 'dataset.npz'))
```

Make a simple plot of the raw data

```
X_csr_DV = dataset.getSparseDocTypeCountMatrix()
bnpy.viz.BarsViz.show_square_images(
    X_csr_DV[:10].toarray(), vmin=0, vmax=5)
pylab.tight_layout()
```

Setup: Function to show bars from start to end of training run

```
def show_bars_over_time(
        task_output_path=None,
        query_laps=[0, 1, 2, 5, None],
        ncols=10):
    '''
    '''
    nrows = len(query_laps)
    fig_handle, ax_handles_RC = pylab.subplots(
        figsize=(SMALL_FIG_SIZE[0] * ncols, SMALL_FIG_SIZE[1] * nrows),
        nrows=nrows, ncols=ncols, sharex=True, sharey=True)
    for row_id, lap_val in enumerate(query_laps):
        cur_model, lap_val = bnpy.load_model_at_lap(task_output_path, lap_val)
        cur_topics_KV = cur_model.obsModel.getTopics()
        # Plot the current model
        cur_ax_list = ax_handles_RC[row_id].flatten().tolist()
        bnpy.viz.BarsViz.show_square_images(
            cur_topics_KV,
            vmin=0.0, vmax=0.06,
            ax_list=cur_ax_list)
        cur_ax_list[0].set_ylabel("lap: %d" % lap_val)
    pylab.tight_layout()
```

Train LDA topic model

Using 10 clusters and the 'randexamples' initialization procedure.

```
local_step_kwargs = dict(
    # perform at most this many iterations at each document
    nCoordAscentItersLP=100,
    # stop local iters early when max change in doc-topic counts < this thr
    convThrLP=0.001,
    )

trained_model, info_dict = bnpy.run(
    dataset, 'FiniteTopicModel', 'Mult', 'VB',
    output_path='/tmp/bars_one_per_doc/helloworld-model=topic+mult-K=10/',
    nLap=100, convergeThr=0.01,
    K=10, initname='randomlikewang',
    alpha=0.5, lam=0.1,
    **local_step_kwargs)
```

First, we can plot the loss function over time We'll skip the first few iterations, since performance is quite bad.

```
pylab.figure(figsize=FIG_SIZE)
pylab.plot(info_dict['lap_history'][1:], info_dict['loss_history'][1:], 'k.-')
pylab.xlabel('num. laps')
pylab.ylabel('loss')
pylab.tight_layout()
```

Show the clusters over time

```
show_bars_over_time(info_dict['task_output_path'])
```

Train LDA topic model with restarts

Using 10 clusters and the 'randexamples' initialization procedure.

```
r_local_step_kwargs = dict(
    # perform at most this many iterations at each document
    nCoordAscentItersLP=100,
    # stop local iters early when max change in doc-topic counts < this thr
    convThrLP=0.001,
    # perform restart proposals at each document
    restartLP=1,
    restartNumItersLP=5,
    restartNumTrialsLP=5,
    )

r_trained_model, r_info_dict = bnpy.run(
    dataset, 'FiniteTopicModel', 'Mult', 'VB',
    output_path='/tmp/bars_one_per_doc/helloworld-model=topic+mult-K=10-localstep=restarts
    nLap=100, convergeThr=0.01,
    K=10, initname='randomlikewang',
    alpha=0.5, lam=0.1,
    **r_local_step_kwargs)

show_bars_over_time(r_info_dict['task_output_path'])
```

Total running time of the script: (0 minutes 0.000 seconds)

`Download Python source code: run-03-demo=topic_model_vb_single_run-model=hdp_topic`

`Download Jupyter notebook: run-03-demo=topic_model_vb_single_run-model=hdp_topic+m`

Generated by Sphinx-Gallery

5.2.5 Toy bars dataset: many bars per doc

Variational training of mixture models and topic models with Multinomial likelihoods on toy “bars” dataset with multiple bars per document.

01: Standard variational training for mixture model

How to train a mixture of multinomials.

```
import bnpy
import numpy as np
import os

from matplotlib import pylab
import seaborn as sns

FIG_SIZE = (3, 3)
SMALL_FIG_SIZE = (1,1)
pylab.rcParams['figure.figsize'] = FIG_SIZE
```

Read toy “bars” dataset from file.

```
dataset_path = os.path.join(bnpy.DATASET_PATH, 'bars_many_per_doc')
dataset = bnpy.data.BagOfWordsData.read_npz(
    os.path.join(dataset_path, 'dataset.npz'))
```

Make a simple plot of the raw data

```
X_csr_DV = dataset.getSparseDocTypeCountMatrix()
bnpy.viz.BarsViz.show_square_images(
    X_csr_DV[:10].toarray(), vmin=0, vmax=5)
#pylab.colorbar()
#pylab.clabel('word count')
pylab.tight_layout()
```

Let’s do one single run of the VB algorithm.

Using 10 clusters and the ‘randexamples’ initializatio procedure.

```
trained_model, info_dict = bnpy.run(
    dataset, 'FiniteMixtureModel', 'Mult', 'VB',
    output_path='/tmp/bars_many_per_doc/helloworld-K=10/',
    nLap=1000, convergeThr=0.0001,
    K=10, initname='randomlikewang',
    gamma0=50.0, lam=0.1)
```

First, we can plot the loss function over time We’ll skip the first few iterations, since performance is quite bad.

```
pylab.figure(figsize=FIG_SIZE)
pylab.plot(info_dict['lap_history'][2:], info_dict['loss_history'][2:], 'k.-')
pylab.xlabel('num. laps')
pylab.ylabel('loss')
pylab.tight_layout()
```

Setup: Useful function to display learned bar structure over time.

```
def show_bars_over_time(
        task_output_path=None,
        query_laps=[0, 1, 2, 5, None],
        ncols=10):
    '''
    '''
    nrows = len(query_laps)
    fig_handle, ax_handles_RC = pylab.subplots(
        figsize=(SMALL_FIG_SIZE[0] * ncols, SMALL_FIG_SIZE[1] * nrows),
        nrows=nrows, ncols=ncols, sharex=True, sharey=True)
    for row_id, lap_val in enumerate(query_laps):
        cur_model, lap_val = bnpy.load_model_at_lap(task_output_path, lap_val)
        cur_topics_KV = cur_model.obsModel.getTopics()
        # Plot the current model
        cur_ax_list = ax_handles_RC[row_id].flatten().tolist()
        bnpy.viz.BarsViz.show_square_images(
            cur_topics_KV,
            vmin=0.0, vmax=0.06,
            ax_list=cur_ax_list)
        cur_ax_list[0].set_ylabel("lap: %d" % lap_val)
    pylab.tight_layout()
```

Show the clusters over time

```
show_bars_over_time(info_dict['task_output_path'])
```

Total running time of the script: (0 minutes 0.000 seconds)

`Download Python source code: run-01-demo=vb_single_run-model=mix+mult.py`

`Download Jupyter notebook: run-01-demo=vb_single_run-model=mix+mult.ipynb`

Generated by Sphinx-Gallery

5.2.6 we8there bag-of-words text dataset

Text from some restaurant reviews (Taddy 2012). Variational training of mixture models and topic models with Multinomial likelihoods.

VB coordinate descent for Mixture of Multinomials

```
import bnpy
import numpy as np
import os

from matplotlib import pylab
import seaborn as sns
```

```
FIG_SIZE = (3, 3)
SMALL_FIG_SIZE = (1,1)
pylab.rcParams['figure.figsize'] = FIG_SIZE
```

Read text dataset from file

```
dataset_path = os.path.join(bnpy.DATASET_PATH, 'we8there', 'raw')
dataset = bnpy.data.BagOfWordsData.read_npz(
    os.path.join(dataset_path, 'dataset.npz'),
    vocabfile=os.path.join(dataset_path, 'x_csc_colnames.txt'))

# Filter out documents with less than 20 words
doc_ids = np.flatnonzero(
    dataset.getDocTypeCountMatrix().sum(axis=1) >= 20)
dataset = dataset.make_subset(docMask=doc_ids, doTrackFullSize=False)
```

Make a simple plot of the raw data

```
bnpy.viz.PrintTopics.plotCompsFromWordCounts(
    dataset.getDocTypeCountMatrix()[:10],
    dataset.vocabList,
    prefix='doc',
    Ktop=10)
```

Train with birth and merge proposals

Take the best of 1 initializations

Ideally, we’d run this longer, but this is convenient for rapid inspection.

```
merge_kwargs = dict(
    m_startLap=5,
    m_pair_ranking_procedure='elbo',
    m_pair_ranking_direction='descending',
    m_pair_ranking_do_exclude_by_thr=1,
    m_pair_ranking_exclusion_thr=-0.0005,
    )

trained_model, info_dict = bnpy.run(
    dataset, 'DPMixtureModel', 'Mult', 'memoVB',
    output_path='/tmp/we8there/helloworld-model=dp_mix+mult-K=30/',
    nLap=15, convergeThr=0.0001, nTask=1, nBatch=1,
    K=30, initname='bregmankmeans+lam1+iter1',
    gamma0=50.0, lam=0.1,
    moves='birth,merge,shuffle',
    b_startLap=2, b_Kfresh=5, b_stopLap=10,
    **merge_kwargs)

bnpy.viz.PrintTopics.plotCompsFromHModel(
    trained_model,
    vocabList=dataset.vocabList,
    Ktop=10)
```

Total running time of the script: (0 minutes 0.000 seconds)

`Download Python source code: run-02-demo=mix_vb+proposals-model=dp_mix+mult.py`

`Download Jupyter notebook: run-02-demo=mix_vb+proposals-model=dp_mix+mult.ipynb`

Generated by Sphinx-Gallery

VB coordinate descent for Mixture of Multinomials

```
import bnpy
import numpy as np
import os

from matplotlib import pylab
import seaborn as sns

FIG_SIZE = (3, 3)
SMALL_FIG_SIZE = (1,1)
pylab.rcParams['figure.figsize'] = FIG_SIZE
```

Read text dataset from file

```
dataset_path = os.path.join(bnpy.DATASET_PATH, 'we8there', 'raw')
dataset = bnpy.data.BagOfWordsData.read_npz(
    os.path.join(dataset_path, 'dataset.npz'),
    vocabfile=os.path.join(dataset_path, 'x_csc_colnames.txt'))

# Filter out documents with less than 20 words
doc_ids = np.flatnonzero(
    dataset.getDocTypeCountMatrix().sum(axis=1) >= 20)
dataset = dataset.make_subset(docMask=doc_ids, doTrackFullSize=False)
```

Make a simple plot of the raw data

```
bnpy.viz.PrintTopics.plotCompsFromWordCounts(
    dataset.getDocTypeCountMatrix()[:10],
    dataset.vocabList,
    prefix='doc',
    Ktop=10)
```

Train with K=1 cluster

This is a simple baseline.

```
trained_model, info_dict = bnpy.run(
    dataset, 'DPMixtureModel', 'Mult', 'VB',
    output_path='/tmp/we8there/helloworld-model=dp_mix+mult-K=1/',
    nLap=1000, convergeThr=0.0001, nTask=1,
    K=1, initname='bregmankmeans+lam1+iter1',
    gamma0=50.0, lam=0.1)
```

```
bnpy.viz.PrintTopics.plotCompsFromHModel(
    trained_model,
    vocabList=dataset.vocabList,
    Ktop=10)
```

Train with K=3 clusters

Take the best of 10 initializations

```
trained_model, info_dict = bnpy.run(
    dataset, 'DPMixtureModel', 'Mult', 'VB',
    output_path='/tmp/we8there/helloworld-model=dp_mix+mult-K=3/',
    nLap=1000, convergeThr=0.0001, nTask=10,
    K=3, initname='bregmankmeans+lam1+iter1',
    gamma0=50.0, lam=0.1)

bnpy.viz.PrintTopics.plotCompsFromHModel(
    trained_model,
    vocabList=dataset.vocabList,
    Ktop=10)
```

Train with K=10 clusters

Take the best of 10 initializations

```
trained_model, info_dict = bnpy.run(
    dataset, 'DPMixtureModel', 'Mult', 'VB',
    output_path='/tmp/we8there/helloworld-model=dp_mix+mult-K=10/',
    nLap=1000, convergeThr=0.0001, nTask=10,
    K=10, initname='bregmankmeans+lam1+iter1',
    gamma0=50.0, lam=0.1)

bnpy.viz.PrintTopics.plotCompsFromHModel(
    trained_model,
    vocabList=dataset.vocabList,
    Ktop=10)
```

Train with K=30 clusters

Take the best of 10 initializations

```
trained_model, info_dict = bnpy.run(
    dataset, 'DPMixtureModel', 'Mult', 'VB',
    output_path='/tmp/we8there/helloworld-model=dp_mix+mult-K=30/',
    nLap=1000, convergeThr=0.0001, nTask=10,
    K=30, initname='bregmankmeans+lam1+iter1',
    gamma0=50.0, lam=0.1)

bnpy.viz.PrintTopics.plotCompsFromHModel(
    trained_model,
    vocabList=dataset.vocabList,
    Ktop=10)
```

Total running time of the script: (0 minutes 0.000 seconds)

Download Python source code: `run-01-demo=mix_vb_single_run-model=mix+mult.py`

Download Jupyter notebook: `run-01-demo=mix_vb_single_run-model=mix+mult.ipynb`

Generated by Sphinx-Gallery

Standard variational training for topic model

```
import bnpy
import numpy as np
import os

from matplotlib import pylab
import seaborn as sns

FIG_SIZE = (2, 2)
SMALL_FIG_SIZE = (1.5, 1.5)
```

Read text dataset from file

```
dataset_path = os.path.join(bnpy.DATASET_PATH, 'we8there', 'raw')
dataset = bnpy.data.BagOfWordsData.read_npz(
    os.path.join(dataset_path, 'dataset.npz'),
    vocabfile=os.path.join(dataset_path, 'x_csc_colnames.txt'))

# Filter out documents with less than 20 words
doc_ids = np.flatnonzero(
    dataset.getDocTypeCountMatrix().sum(axis=1) >= 20)
dataset = dataset.make_subset(docMask=doc_ids, doTrackFullSize=False)
```

Train LDA topic model

Using 10 clusters and a random initialization procedure.

```
local_step_kwargs = dict(
    # perform at most this many iterations at each document
    nCoordAscentItersLP=100,
    # stop local iters early when max change in doc-topic counts < this thr
    convThrLP=0.001,
    )

trained_model, info_dict = bnpy.run(
    dataset, 'FiniteTopicModel', 'Mult', 'VB',
    output_path='/tmp/we8there/helloworld-model=topic+mult-K=10/',
    nLap=10, convergeThr=0.01,
    K=10, initname='randomlikewang',
    alpha=0.5, lam=0.1,
    **local_step_kwargs)
```

First, we can plot the loss function over time We'll skip the first few iterations, since performance is quite bad.

```
pylab.figure(figsize=FIG_SIZE)
pylab.plot(info_dict['lap_history'][1:], info_dict['loss_history'][1:], 'k.-')
pylab.xlabel('num. laps')
pylab.ylabel('loss')
pylab.tight_layout()
```

Setup: Helper function to plot bars at each stage of training

```
def show_top_words_over_time(
        task_output_path=None,
        vocabList=None,
        query_laps=[0, 1, 2, 5, None],
        ncols=10):
    '''
    '''
    nrows = len(query_laps)
    fig_handle, ax_handles_RC = pylab.subplots(
        figsize=(SMALL_FIG_SIZE[0] * ncols, SMALL_FIG_SIZE[1] * nrows),
        nrows=nrows, ncols=ncols, sharex=True, sharey=True)
    for row_id, lap_val in enumerate(query_laps):
        cur_model, lap_val = bnpy.load_model_at_lap(task_output_path, lap_val)
        # Plot the current model
        cur_ax_list = ax_handles_RC[row_id].flatten().tolist()
        bnpy.viz.PrintTopics.plotCompsFromHModel(
            cur_model,
            vocabList=vocabList,
            fontsize=9,
            Ktop=7,
            ax_list=cur_ax_list)
        cur_ax_list[0].set_ylabel("lap: %d" % lap_val)
    pylab.subplots_adjust(
        wspace=0.04, hspace=0.1,
        left=0.01, right=0.99, top=0.99, bottom=0.1)
    pylab.tight_layout()
```

Show the topics over time

```
show_top_words_over_time(
    info_dict['task_output_path'], vocabList=dataset.vocabList)
```

Total running time of the script: (0 minutes 0.000 seconds)

`Download Python source code: run-03-demo=topic_vb_single_run-model=hdp_topic+mult.`

`Download Jupyter notebook: run-03-demo=topic_vb_single_run-model=hdp_topic+mult.ip`

Generated by Sphinx-Gallery

Birth and merge variational training for topic model

```
import bnpy
import numpy as np
import os

from matplotlib import pylab
import seaborn as sns

FIG_SIZE = (2, 2)
SMALL_FIG_SIZE = (1.5, 1.5)
```

Read text dataset from file

```
dataset_path = os.path.join(bnpy.DATASET_PATH, 'we8there', 'raw')
dataset = bnpy.data.BagOfWordsData.read_npz(
    os.path.join(dataset_path, 'dataset.npz'),
    vocabfile=os.path.join(dataset_path, 'x_csc_colnames.txt'))

# Filter out documents with less than 20 words
doc_ids = np.flatnonzero(
    dataset.getDocTypeCountMatrix().sum(axis=1) >= 20)
dataset = dataset.make_subset(docMask=doc_ids, doTrackFullSize=False)
```

Train LDA topic model

Using 10 clusters and a random initialization procedure.

```
local_step_kwargs = dict(
    # perform at most this many iterations at each document
    nCoordAscentItersLP=100,
    # stop local iters early when max change in doc-topic counts < this thr
    convThrLP=0.001,
    )
merge_kwargs = dict(
    m_startLap=5,
    )
birth_kwargs = dict(
    b_startLap=2,
    b_stopLap=20,
    b_Kfresh=5)

trained_model, info_dict = bnpy.run(
    dataset, 'HDPTopicModel', 'Mult', 'memoVB',
    output_path='/tmp/we8there/trymoves-model=hdp_topic+mult-K=5/',
    nLap=20, convergeThr=0.01, nBatch=1,
    K=5, initname='randomlikewang',
    gamma=50.0, alpha=0.5, lam=0.1,
    moves='birth,merge,shuffle',
    **dict(local_step_kwargs.items() +
        merge_kwargs.items() +
        birth_kwargs.items()))
```

Setup: Helper function to plot topics at each stage of training

```
def show_top_words_over_time(
        task_output_path=None,
        vocabList=None,
        query_laps=[0, 1, 2, 5, None],
        ncols=10):
    '''
    '''
    nrows = len(query_laps)
    fig_handle, ax_handles_RC = pylab.subplots(
        figsize=(SMALL_FIG_SIZE[0] * ncols, SMALL_FIG_SIZE[1] * nrows),
        nrows=nrows, ncols=ncols, sharex=True, sharey=True)
    for row_id, lap_val in enumerate(query_laps):
        cur_model, lap_val = bnpy.load_model_at_lap(task_output_path, lap_val)
        # Plot the current model
        cur_ax_list = ax_handles_RC[row_id].flatten().tolist()
        bnpy.viz.PrintTopics.plotCompsFromHModel(
            cur_model,
            vocabList=vocabList,
            fontsize=9,
            Ktop=7,
            ax_list=cur_ax_list)
        cur_ax_list[0].set_ylabel("lap: %d" % lap_val)
    pylab.subplots_adjust(
        wspace=0.04, hspace=0.1,
        left=0.01, right=0.99, top=0.99, bottom=0.1)
    pylab.tight_layout()
```

Show the topics over time

```
show_top_words_over_time(
    info_dict['task_output_path'], vocabList=dataset.vocabList)
```

Total running time of the script: (0 minutes 0.000 seconds)

`Download Python source code: run-04-demo=topic_vb+proposals-model=hdp_topic+mult.p`

`Download Jupyter notebook: run-04-demo=topic_vb+proposals-model=hdp_topic+mult.ipy`

Generated by Sphinx-Gallery

5.2.7 Small wikipedia bag-of-words text dataset

Text from a few thousand wikipedia articles. Variational training of mixture models and topic models with Multinomial likelihoods.

Birth and merge variational training for topic model

```
import bnpy
import numpy as np
import os
```

```
from matplotlib import pylab
import seaborn as sns

FIG_SIZE = (2, 2)
SMALL_FIG_SIZE = (1.5, 1.5)
```

Read text dataset from file

```
dataset_path = os.path.join(bnpy.DATASET_PATH, 'wiki')
dataset = bnpy.data.BagOfWordsData.LoadFromFile_ldac(
    os.path.join(dataset_path, 'train.ldac'),
    vocabfile=os.path.join(dataset_path, 'vocab.txt'))

# Filter out the first 1000 documents with less than 50 words
doc_ids = np.flatnonzero(np.logical_and(
    dataset.getDocTypeCountMatrix().sum(axis=1) >= 50,
    dataset.getDocTypeCountMatrix().sum(axis=1) < 500))[:1000]
dataset = dataset.make_subset(docMask=doc_ids, doTrackFullSize=False)
```

Train LDA topic model

Using 10 clusters and a random initialization procedure.

```
local_step_kwargs = dict(
    # perform at most this many iterations at each document
    nCoordAscentItersLP=100,
    # stop local iters early when max change in doc-topic counts < this thr
    convThrLP=0.01,
    )
merge_kwargs = dict(
    m_startLap=5,
    )
birth_kwargs = dict(
    b_startLap=4,
    b_stopLap=10,
    b_Kfresh=5)

trained_model, info_dict = bnpy.run(
    dataset, 'HDPTopicModel', 'Mult', 'memoVB',
    output_path='/tmp/wiki/trymoves-model=hdp_topic+mult-K=5/',
    nLap=20, convergeThr=0.01, nBatch=5,
    K=5, initname='randomlikewang',
    gamma=50.0, alpha=0.5, lam=0.1,
    moves='birth,merge,shuffle',
    **dict(local_step_kwargs.items() +
        merge_kwargs.items() +
        birth_kwargs.items()))
```

Setup: Helper function to plot topics at each stage of training

```
def show_top_words_over_time(
        task_output_path=None,
        vocabList=None,
        query_laps=[0, 1, 2, 5, None],
        ncols=10):
    '''
    '''
    nrows = len(query_laps)
    fig_handle, ax_handles_RC = pylab.subplots(
        figsize=(SMALL_FIG_SIZE[0] * ncols, SMALL_FIG_SIZE[1] * nrows),
        nrows=nrows, ncols=ncols, sharex=True, sharey=True)
    for row_id, lap_val in enumerate(query_laps):
        cur_model, lap_val = bnpy.load_model_at_lap(task_output_path, lap_val)
        # Plot the current model
        cur_ax_list = ax_handles_RC[row_id].flatten().tolist()
        bnpy.viz.PrintTopics.plotCompsFromHModel(
            cur_model,
            vocabList=vocabList,
            fontsize=9,
            Ktop=7,
            ax_list=cur_ax_list)
        cur_ax_list[0].set_ylabel("lap: %d" % lap_val)
    pylab.subplots_adjust(
        wspace=0.04, hspace=0.1,
        left=0.01, right=0.99, top=0.99, bottom=0.1)
    pylab.tight_layout()
```

Show the topics over time

```
show_top_words_over_time(
    info_dict['task_output_path'], vocabList=dataset.vocabList)
```

Total running time of the script: (0 minutes 0.000 seconds)

`Download Python source code: run-01-demo=topic_vb+proposals-model=hdp_topic+mult.p`

`Download Jupyter notebook: run-01-demo=topic_vb+proposals-model=hdp_topic+mult.ipy`

Generated by Sphinx-Gallery

5.2.8 Small dataset of 6 motion-capture sequences

Variational training of mixture models and HMM models with various Gaussian observation models.

Comparing models for sequential data

How to train mixtures and HMMs with various observation models on the same dataset.

```
import bnpy
import numpy as np
import os
```

```
from matplotlib import pylab
import seaborn as sns

SMALL_FIG_SIZE = (2.5, 2.5)
FIG_SIZE = (5, 5)
pylab.rcParams['figure.figsize'] = FIG_SIZE
```

Load dataset from file

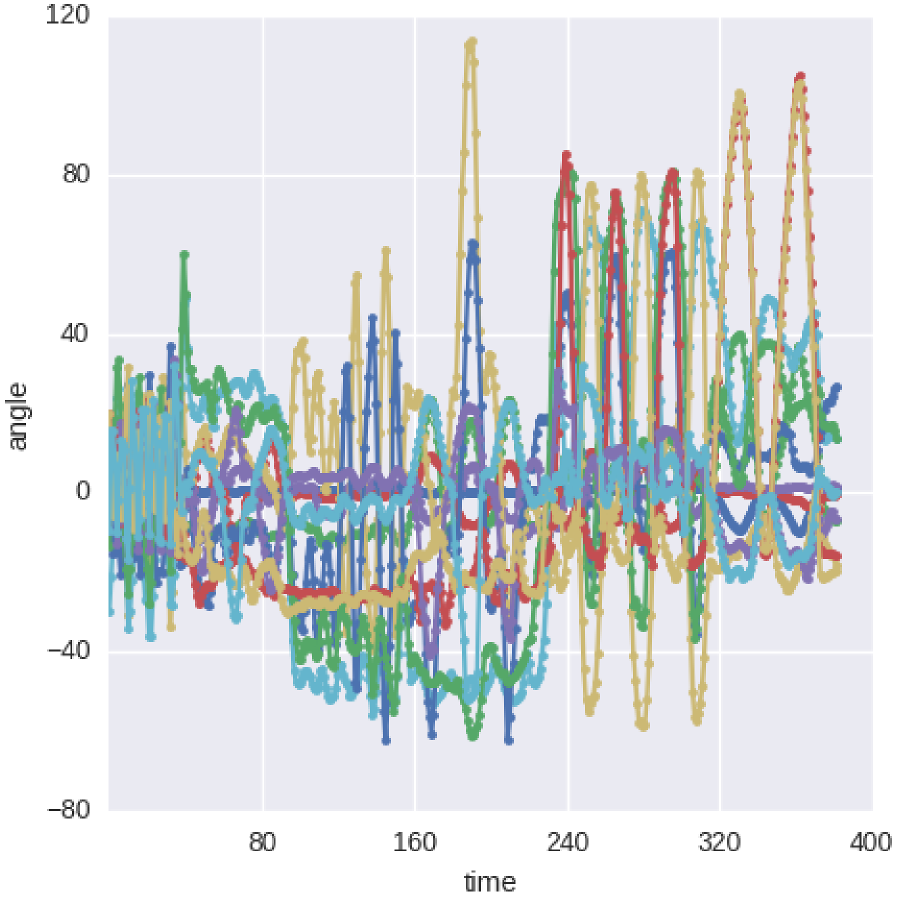
```
dataset_path = os.path.join(bnpy.DATASET_PATH, 'mocap6')
dataset = bnpy.data.GroupXData.read_npz(
    os.path.join(dataset_path, 'dataset.npz'))
```

Setup: Function to make a simple plot of the raw data

```
def show_single_sequence(seq_id):
    start = dataset.doc_range[seq_id]
    stop = dataset.doc_range[seq_id + 1]
    for dim in xrange(12):
        X_seq = dataset.X[start:stop]
        pylab.plot(X_seq[:, dim], '.-')
    pylab.xlabel('time')
    pylab.ylabel('angle')
    pylab.tight_layout()
```

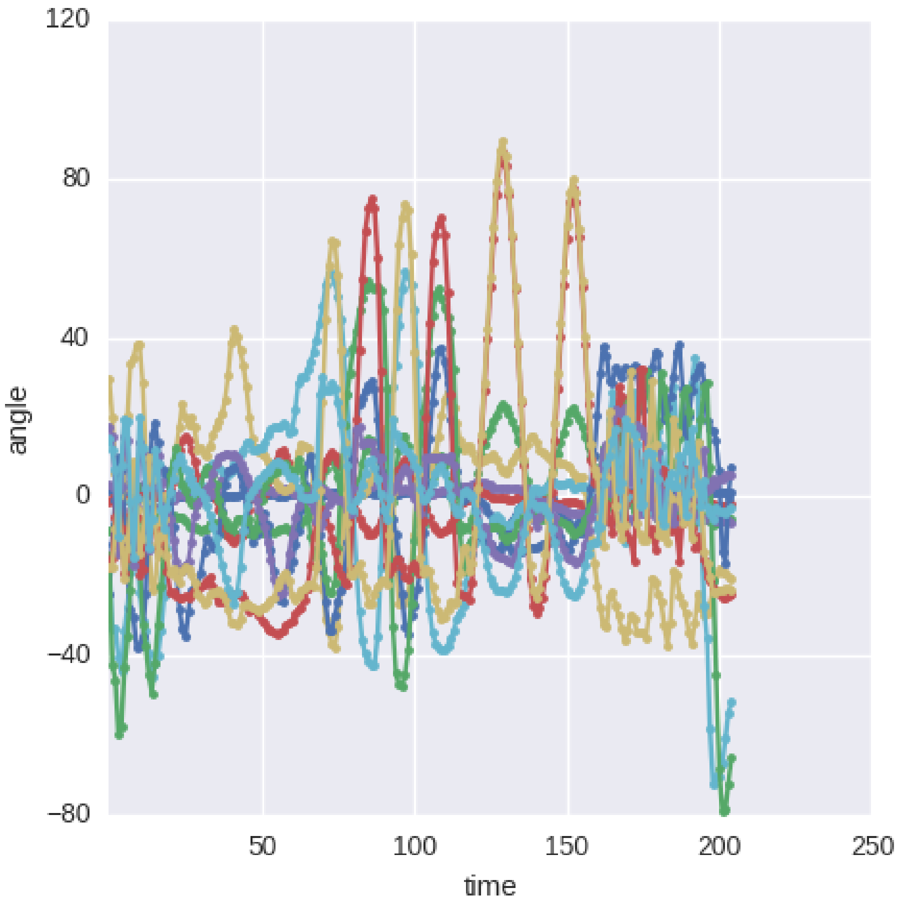
Visualization of the first sequence

```
show_single_sequence(0)
```



Visualization of the second sequence

```
show_single_sequence(1)
```



Setup: hyperparameters

```
alpha = 0.5
gamma = 5.0
sF = 1.0
K = 20
```

DP mixture with *DiagGauss* observation model

```
mixdiag_trained_model, mixdiag_info_dict = bnpy.run(
    dataset, 'DPMixtureModel', 'DiagGauss', 'memoVB',
    output_path='/tmp/mocap6/showcase-K=20-model=DP+DiagGauss-ECovMat=1*eye/',
    nLap=50, nTask=1, nBatch=1, convergeThr=0.0001,
    alpha=alpha, gamma=gamma, sF=sF, ECovMat='eye',
    K=K, initname='randexamples',
    )
```

Out:

```
WARNING: Found unrecognized keyword args. These are ignored.
  --alpha
Dataset Summary:
GroupXData
  total size: 6 units
  batch size: 6 units
  num. batches: 1
Allocation Model:  DP mixture with K=0. Concentration gamma0= 5.00
Obs. Data  Model:  Gaussian with diagonal covariance.
Obs. Data  Prior:  independent Gauss-Wishart prior on each dimension
  Wishart params
    nu = 14  ...
  beta = [ 12  12]  ...
  Expectations
  E[  mean[k]] =
  [ 0  0] ...
  E[ covar[k]] =
  [[ 1.  0.]
   [ 0.  1.]] ...
Initialization:
  initname = randexamples
  K = 20 (number of clusters)
  seed = 1607680
  elapsed_time: 0.0 sec
Learn Alg: memoVB | task  1/1 | alg. seed: 1607680 | data order seed: 8541952
task_output_path: /tmp/mocap6/showcase-K=20-model=DP+DiagGauss-ECovMat=1*eye/1
    1.000/50 after      0 sec. |    178.8 MiB | K   20 | loss  3.845424399e+00 |
    2.000/50 after      0 sec. |    178.8 MiB | K   20 | loss  3.756537185e+00 | Ndiff   26
    3.000/50 after      0 sec. |    178.8 MiB | K   20 | loss  3.723105766e+00 | Ndiff   29
    4.000/50 after      0 sec. |    178.8 MiB | K   20 | loss  3.703275957e+00 | Ndiff   19
    5.000/50 after      0 sec. |    178.8 MiB | K   20 | loss  3.689932307e+00 | Ndiff   22
    6.000/50 after      0 sec. |    178.8 MiB | K   20 | loss  3.681432839e+00 | Ndiff   24
    7.000/50 after      0 sec. |    178.8 MiB | K   20 | loss  3.676243541e+00 | Ndiff   20
    8.000/50 after      0 sec. |    178.8 MiB | K   20 | loss  3.672159855e+00 | Ndiff   16
    9.000/50 after      0 sec. |    178.8 MiB | K   20 | loss  3.669045283e+00 | Ndiff   13
   10.000/50 after      0 sec. |    178.8 MiB | K   20 | loss  3.665208654e+00 | Ndiff   13
   11.000/50 after      0 sec. |    178.8 MiB | K   20 | loss  3.661952478e+00 | Ndiff   16
   12.000/50 after      0 sec. |    178.8 MiB | K   20 | loss  3.658115620e+00 | Ndiff   17
   13.000/50 after      0 sec. |    178.8 MiB | K   20 | loss  3.655058982e+00 | Ndiff   12
   14.000/50 after      0 sec. |    178.8 MiB | K   20 | loss  3.653853641e+00 | Ndiff    5
   15.000/50 after      0 sec. |    178.8 MiB | K   20 | loss  3.652570589e+00 | Ndiff    5
   16.000/50 after      0 sec. |    178.8 MiB | K   20 | loss  3.651988134e+00 | Ndiff    5
   17.000/50 after      0 sec. |    178.8 MiB | K   20 | loss  3.651566377e+00 | Ndiff    5
   18.000/50 after      0 sec. |    178.8 MiB | K   20 | loss  3.651133556e+00 | Ndiff    6
   19.000/50 after      0 sec. |    178.8 MiB | K   20 | loss  3.650800669e+00 | Ndiff    6
   20.000/50 after      0 sec. |    178.8 MiB | K   20 | loss  3.650523019e+00 | Ndiff    6
   21.000/50 after      0 sec. |    178.8 MiB | K   20 | loss  3.650284547e+00 | Ndiff    6
   22.000/50 after      0 sec. |    178.8 MiB | K   20 | loss  3.650075236e+00 | Ndiff    5
   23.000/50 after      0 sec. |    178.8 MiB | K   20 | loss  3.649874747e+00 | Ndiff    4
   24.000/50 after      0 sec. |    178.8 MiB | K   20 | loss  3.649684370e+00 | Ndiff    3
   25.000/50 after      0 sec. |    178.8 MiB | K   20 | loss  3.649503048e+00 | Ndiff    2
   26.000/50 after      0 sec. |    178.8 MiB | K   20 | loss  3.649287038e+00 | Ndiff    2
   27.000/50 after      0 sec. |    178.8 MiB | K   20 | loss  3.648992642e+00 | Ndiff    1
   28.000/50 after      1 sec. |    178.8 MiB | K   20 | loss  3.648274600e+00 | Ndiff    1
   29.000/50 after      1 sec. |    178.8 MiB | K   20 | loss  3.647520307e+00 | Ndiff    1
   30.000/50 after      1 sec. |    178.8 MiB | K   20 | loss  3.647350990e+00 | Ndiff    1
   31.000/50 after      1 sec. |    178.8 MiB | K   20 | loss  3.647242457e+00 | Ndiff    1
   32.000/50 after      1 sec. |    178.8 MiB | K   20 | loss  3.647171899e+00 | Ndiff    1
   33.000/50 after      1 sec. |    178.8 MiB | K   20 | loss  3.647125709e+00 | Ndiff    2
   34.000/50 after      1 sec. |    178.8 MiB | K   20 | loss  3.647094615e+00 | Ndiff    2
   35.000/50 after      1 sec. |    178.8 MiB | K   20 | loss  3.647072676e+00 | Ndiff    1
   36.000/50 after      1 sec. |    178.8 MiB | K   20 | loss  3.647056795e+00 | Ndiff    1
   37.000/50 after      1 sec. |    178.8 MiB | K   20 | loss  3.647045269e+00 | Ndiff    1
   38.000/50 after      1 sec. |    178.8 MiB | K   20 | loss  3.647036962e+00 | Ndiff    1
   39.000/50 after      1 sec. |    178.8 MiB | K   20 | loss  3.647031023e+00 | Ndiff    1
   40.000/50 after      1 sec. |    178.8 MiB | K   20 | loss  3.647026801e+00 | Ndiff    1
   41.000/50 after      1 sec. |    178.8 MiB | K   20 | loss  3.647023808e+00 | Ndiff    0
   42.000/50 after      1 sec. |    178.8 MiB | K   20 | loss  3.647021689e+00 | Ndiff    0
   43.000/50 after      1 sec. |    178.8 MiB | K   20 | loss  3.647020187e+00 | Ndiff    0
   44.000/50 after      1 sec. |    178.8 MiB | K   20 | loss  3.647019120e+00 | Ndiff    0
   45.000/50 after      1 sec. |    178.8 MiB | K   20 | loss  3.647018361e+00 | Ndiff    0
   46.000/50 after      1 sec. |    178.8 MiB | K   20 | loss  3.647017819e+00 | Ndiff    0
   47.000/50 after      1 sec. |    178.8 MiB | K   20 | loss  3.647017432e+00 | Ndiff    0
   48.000/50 after      1 sec. |    178.8 MiB | K   20 | loss  3.647017154e+00 | Ndiff    0
   49.000/50 after      1 sec. |    178.8 MiB | K   20 | loss  3.647016955e+00 | Ndiff    0
   50.000/50 after      1 sec. |    178.8 MiB | K   20 | loss  3.647016811e+00 | Ndiff    0
... done. not converged. max laps thru data exceeded.
```

HDP-HMM with *DiagGauss* observation model

Assume diagonal covariances.

Start with too many clusters (K=20)

```
hmmdiag_trained_model, hmmdiag_info_dict = bnpy.run(
    dataset, 'HDPHMM', 'DiagGauss', 'memoVB',
    output_path='/tmp/mocap6/showcase-K=20-model=HDPHMM+DiagGauss-ECovMat=1*eye/',
    nLap=50, nTask=1, nBatch=1, convergeThr=0.0001,
    alpha=alpha, gamma=gamma, sF=sF, ECovMat='eye',
    K=K, initname='randexamples',
    )
```

Out:

```
WARNING: Found unrecognized keyword args. These are ignored.
  --alpha
Dataset Summary:
GroupXData
  total size: 6 units
  batch size: 6 units
  num. batches: 1
Allocation Model:  None
Obs. Data  Model:  Gaussian with diagonal covariance.
Obs. Data  Prior:  independent Gauss-Wishart prior on each dimension
  Wishart params
```

```
    nu = 14  ...
  beta = [ 12  12]  ...
  Expectations
  E[  mean[k]] =
  [ 0  0] ...
  E[ covar[k]] =
  [[ 1.  0.]
   [ 0.  1.]] ...
Initialization:
  initname = randexamples
  K = 20 (number of clusters)
  seed = 1607680
  elapsed_time: 0.0 sec
Learn Alg: memoVB | task  1/1 | alg. seed: 1607680 | data order seed: 8541952
task_output_path: /tmp/mocap6/showcase-K=20-model=HDPHMM+DiagGauss-ECovMat=1*eye/1
    1.000/50 after      0 sec. |    180.1 MiB | K   20 | loss  3.717124121e+00 |
    2.000/50 after      0 sec. |    179.0 MiB | K   20 | loss  3.612329446e+00 | Ndiff   35
    3.000/50 after      1 sec. |    179.0 MiB | K   20 | loss  3.580489903e+00 | Ndiff   22
    4.000/50 after      1 sec. |    179.0 MiB | K   20 | loss  3.565890757e+00 | Ndiff   19
    5.000/50 after      1 sec. |    179.0 MiB | K   20 | loss  3.552817867e+00 | Ndiff   18
    6.000/50 after      1 sec. |    179.0 MiB | K   20 | loss  3.547626693e+00 | Ndiff    9
    7.000/50 after      1 sec. |    179.0 MiB | K   20 | loss  3.545715575e+00 | Ndiff    3
    8.000/50 after      2 sec. |    179.0 MiB | K   20 | loss  3.542710111e+00 | Ndiff    8
    9.000/50 after      2 sec. |    179.0 MiB | K   20 | loss  3.534107178e+00 | Ndiff   14
   10.000/50 after      2 sec. |    179.0 MiB | K   20 | loss  3.530991441e+00 | Ndiff    9
   11.000/50 after      2 sec. |    179.0 MiB | K   20 | loss  3.527520848e+00 | Ndiff    9
   12.000/50 after      2 sec. |    179.0 MiB | K   20 | loss  3.525056000e+00 | Ndiff   11
   13.000/50 after      3 sec. |    179.0 MiB | K   20 | loss  3.523781079e+00 | Ndiff    4
   14.000/50 after      3 sec. |    179.0 MiB | K   20 | loss  3.521694790e+00 | Ndiff    8
   15.000/50 after      3 sec. |    179.0 MiB | K   20 | loss  3.518502897e+00 | Ndiff    4
   16.000/50 after      3 sec. |    179.0 MiB | K   20 | loss  3.517515182e+00 | Ndiff    4
   17.000/50 after      3 sec. |    179.0 MiB | K   20 | loss  3.517412339e+00 | Ndiff    0
   18.000/50 after      3 sec. |    179.0 MiB | K   20 | loss  3.517396038e+00 | Ndiff    0
   19.000/50 after      4 sec. |    179.0 MiB | K   20 | loss  3.517386711e+00 | Ndiff    0
   20.000/50 after      4 sec. |    179.0 MiB | K   20 | loss  3.517380000e+00 | Ndiff    0
   21.000/50 after      4 sec. |    179.0 MiB | K   20 | loss  3.517375098e+00 | Ndiff    0
   22.000/50 after      4 sec. |    179.0 MiB | K   20 | loss  3.517371095e+00 | Ndiff    0
   23.000/50 after      4 sec. |    179.0 MiB | K   20 | loss  3.517367640e+00 | Ndiff    0
   24.000/50 after      5 sec. |    179.0 MiB | K   20 | loss  3.517364400e+00 | Ndiff    0
   25.000/50 after      5 sec. |    179.0 MiB | K   20 | loss  3.517361220e+00 | Ndiff    0
   26.000/50 after      5 sec. |    179.0 MiB | K   20 | loss  3.517357862e+00 | Ndiff    0
   27.000/50 after      5 sec. |    179.0 MiB | K   20 | loss  3.517353576e+00 | Ndiff    0
   28.000/50 after      5 sec. |    179.0 MiB | K   20 | loss  3.517347351e+00 | Ndiff    0
   29.000/50 after      5 sec. |    179.0 MiB | K   20 | loss  3.517338850e+00 | Ndiff    0
   30.000/50 after      6 sec. |    179.0 MiB | K   20 | loss  3.517329011e+00 | Ndiff    0
   31.000/50 after      6 sec. |    179.0 MiB | K   20 | loss  3.517319213e+00 | Ndiff    0
   32.000/50 after      6 sec. |    179.0 MiB | K   20 | loss  3.517310197e+00 | Ndiff    0
   33.000/50 after      6 sec. |    179.0 MiB | K   20 | loss  3.517301719e+00 | Ndiff    0
   34.000/50 after      6 sec. |    179.0 MiB | K   20 | loss  3.517293335e+00 | Ndiff    0
   35.000/50 after      6 sec. |    179.0 MiB | K   20 | loss  3.517285024e+00 | Ndiff    0
   36.000/50 after      7 sec. |    179.0 MiB | K   20 | loss  3.517277155e+00 | Ndiff    0
   37.000/50 after      7 sec. |    179.0 MiB | K   20 | loss  3.517270251e+00 | Ndiff    0
   38.000/50 after      7 sec. |    179.0 MiB | K   20 | loss  3.517264770e+00 | Ndiff    0
   39.000/50 after      7 sec. |    179.0 MiB | K   20 | loss   3.517260855e+00 | Ndiff
   40.000/50 after      7 sec. |    179.0 MiB | K   20 | loss   3.517258241e+00 | Ndiff
   41.000/50 after      7 sec. |    179.0 MiB | K   20 | loss   3.517256507e+00 | Ndiff
   42.000/50 after      7 sec. |    179.0 MiB | K   20 | loss   3.517255300e+00 | Ndiff
   43.000/50 after      8 sec. |    179.0 MiB | K   20 | loss   3.517254394e+00 | Ndiff
   44.000/50 after      8 sec. |    179.0 MiB | K   20 | loss   3.517253647e+00 | Ndiff
   45.000/50 after      8 sec. |    179.0 MiB | K   20 | loss   3.517252964e+00 | Ndiff
   46.000/50 after      8 sec. |    179.0 MiB | K   20 | loss   3.517252267e+00 | Ndiff
   47.000/50 after      8 sec. |    179.0 MiB | K   20 | loss   3.517251461e+00 | Ndiff
   48.000/50 after      8 sec. |    179.0 MiB | K   20 | loss   3.517250379e+00 | Ndiff
   49.000/50 after      9 sec. |    179.0 MiB | K   20 | loss   3.517248618e+00 | Ndiff
   50.000/50 after      9 sec. |    179.0 MiB | K   20 | loss   3.517244801e+00 | Ndiff
... done. not converged. max laps thru data exceeded.
```

HDP-HMM with *Gauss* observation model

Assume full covariances.

Start with too many clusters (K=20)

```
hmmfull_trained_model, hmmfull_info_dict = bnpy.run(
    dataset, 'HDPHMM', 'Gauss', 'memoVB',
    output_path='/tmp/mocap6/showcase-K=20-model=HDPHMM+Gauss-ECovMat=1*eye/',
    nLap=50, nTask=1, nBatch=1, convergeThr=0.0001,
    alpha=alpha, gamma=gamma, sF=sF, ECovMat='eye',
    K=K, initname='randexamples',
    )
```

Out:

```
WARNING: Found unrecognized keyword args. These are ignored.
  --alpha
Dataset Summary:
GroupXData
  total size: 6 units
  batch size: 6 units
  num. batches: 1
Allocation Model:  None
Obs. Data  Model:  Gaussian with full covariance.
Obs. Data  Prior:  Gauss-Wishart on mean and covar of each cluster
  E[  mean[k] ] =
   [ 0.  0.]  ...
  E[ covar[k] ] =
  [[ 1.  0.]
   [ 0.  1.]] ...
Initialization:
  initname = randexamples
  K = 20 (number of clusters)
  seed = 1607680
  elapsed_time: 0.0 sec
Learn Alg: memoVB | task  1/1 | alg. seed: 1607680 | data order seed: 8541952
task_output_path: /tmp/mocap6/showcase-K=20-model=HDPHMM+Gauss-ECovMat=1*eye/1
```

```
    1.000/50 after      0 sec. |    179.0 MiB | K   20 | loss  3.453792348e+00 |
    2.000/50 after      0 sec. |    179.0 MiB | K   20 | loss  3.332556042e+00 | Ndiff   27
    3.000/50 after      1 sec. |    179.0 MiB | K   20 | loss  3.280863302e+00 | Ndiff   33
    4.000/50 after      1 sec. |    179.0 MiB | K   20 | loss  3.247802919e+00 | Ndiff   27
    5.000/50 after      1 sec. |    179.0 MiB | K   20 | loss  3.232386544e+00 | Ndiff   16
    6.000/50 after      1 sec. |    179.0 MiB | K   20 | loss  3.214482068e+00 | Ndiff   24
    7.000/50 after      2 sec. |    179.0 MiB | K   20 | loss  3.202981898e+00 | Ndiff   20
    8.000/50 after      2 sec. |    179.0 MiB | K   20 | loss  3.198209231e+00 | Ndiff    9
    9.000/50 after      2 sec. |    179.0 MiB | K   20 | loss  3.196602056e+00 | Ndiff    4
   10.000/50 after      2 sec. |    179.0 MiB | K   20 | loss  3.194049899e+00 | Ndiff    3
   11.000/50 after      2 sec. |    179.0 MiB | K   20 | loss  3.192384953e+00 | Ndiff    3
   12.000/50 after      3 sec. |    179.0 MiB | K   20 | loss  3.190378614e+00 | Ndiff    3
   13.000/50 after      3 sec. |    179.0 MiB | K   20 | loss  3.188801039e+00 | Ndiff    4
   14.000/50 after      3 sec. |    179.0 MiB | K   20 | loss  3.187980225e+00 | Ndiff    2
   15.000/50 after      3 sec. |    179.0 MiB | K   20 | loss  3.187267197e+00 | Ndiff    1
   16.000/50 after      3 sec. |    179.0 MiB | K   20 | loss  3.187104503e+00 | Ndiff    0
   17.000/50 after      4 sec. |    179.0 MiB | K   20 | loss  3.186910658e+00 | Ndiff    0
   18.000/50 after      4 sec. |    179.0 MiB | K   20 | loss  3.186639962e+00 | Ndiff    1
   19.000/50 after      4 sec. |    179.0 MiB | K   20 | loss  3.186450784e+00 | Ndiff    0
   20.000/50 after      4 sec. |    179.0 MiB | K   20 | loss  3.185973787e+00 | Ndiff    1
   21.000/50 after      4 sec. |    179.0 MiB | K   20 | loss  3.185779780e+00 | Ndiff    0
   22.000/50 after      5 sec. |    179.0 MiB | K   20 | loss  3.185453273e+00 | Ndiff    0
   23.000/50 after      5 sec. |    179.0 MiB | K   20 | loss  3.182363213e+00 | Ndiff    2
   24.000/50 after      5 sec. |    179.0 MiB | K   20 | loss  3.182200890e+00 | Ndiff    1
   25.000/50 after      5 sec. |    179.0 MiB | K   20 | loss  3.182084494e+00 | Ndiff    1
   26.000/50 after      5 sec. |    179.0 MiB | K   20 | loss  3.182015005e+00 | Ndiff    0
   27.000/50 after      6 sec. |    179.0 MiB | K   20 | loss  3.181989580e+00 | Ndiff    0
   28.000/50 after      6 sec. |    179.0 MiB | K   20 | loss  3.181974836e+00 | Ndiff    0
   29.000/50 after      6 sec. |    179.0 MiB | K   20 | loss  3.181947074e+00 | Ndiff    0
   30.000/50 after      6 sec. |    179.0 MiB | K   20 | loss  3.181838450e+00 | Ndiff    0
   31.000/50 after      6 sec. |    179.0 MiB | K   20 | loss  3.181817711e+00 | Ndiff    0
   32.000/50 after      7 sec. |    179.0 MiB | K   20 | loss  3.181814701e+00 | Ndiff    0
   33.000/50 after      7 sec. |    179.0 MiB | K   20 | loss  3.181813740e+00 | Ndiff    0
   34.000/50 after      7 sec. |    179.0 MiB | K   20 | loss  3.181813509e+00 | Ndiff    0
   35.000/50 after      7 sec. |    179.0 MiB | K   20 | loss  3.181813438e+00 | Ndiff    0
   36.000/50 after      7 sec. |    179.0 MiB | K   20 | loss  3.181813408e+00 | Ndiff    0
   37.000/50 after      8 sec. |    179.0 MiB | K   20 | loss  3.181813395e+00 | Ndiff    0
   38.000/50 after      8 sec. |    179.0 MiB | K   20 | loss  3.181813388e+00 | Ndiff    0
   39.000/50 after      8 sec. |    179.0 MiB | K   20 | loss  3.181813385e+00 | Ndiff    0
   40.000/50 after      8 sec. |    179.0 MiB | K   20 | loss  3.181813383e+00 | Ndiff    0
   41.000/50 after      8 sec. |    179.0 MiB | K   20 | loss  3.181813382e+00 | Ndiff    0
   42.000/50 after      9 sec. |    179.0 MiB | K   20 | loss  3.181813382e+00 | Ndiff    0
   43.000/50 after      9 sec. |    179.0 MiB | K   20 | loss  3.181813381e+00 | Ndiff    0
   44.000/50 after      9 sec. |    179.0 MiB | K   20 | loss  3.181813381e+00 | Ndiff    0
   45.000/50 after      9 sec. |    179.0 MiB | K   20 | loss  3.181813381e+00 | Ndiff    0
   46.000/50 after      9 sec. |    179.0 MiB | K   20 | loss  3.181813381e+00 | Ndiff    0
   47.000/50 after     10 sec. |    179.0 MiB | K   20 | loss  3.181813381e+00 | Ndiff    0
   48.000/50 after     10 sec. |    179.0 MiB | K   20 | loss  3.181813381e+00 | Ndiff    0
   49.000/50 after     10 sec. |    179.0 MiB | K   20 | loss  3.181813381e+00 | Ndiff    0
   50.000/50 after     10 sec. |    179.0 MiB | K   20 | loss  3.181813381e+00 | Ndiff    0
... done. not converged. max laps thru data exceeded.
```

HDP-HMM with *AutoRegGauss* observation model

Assume full covariances.

Start with too many clusters (K=20)

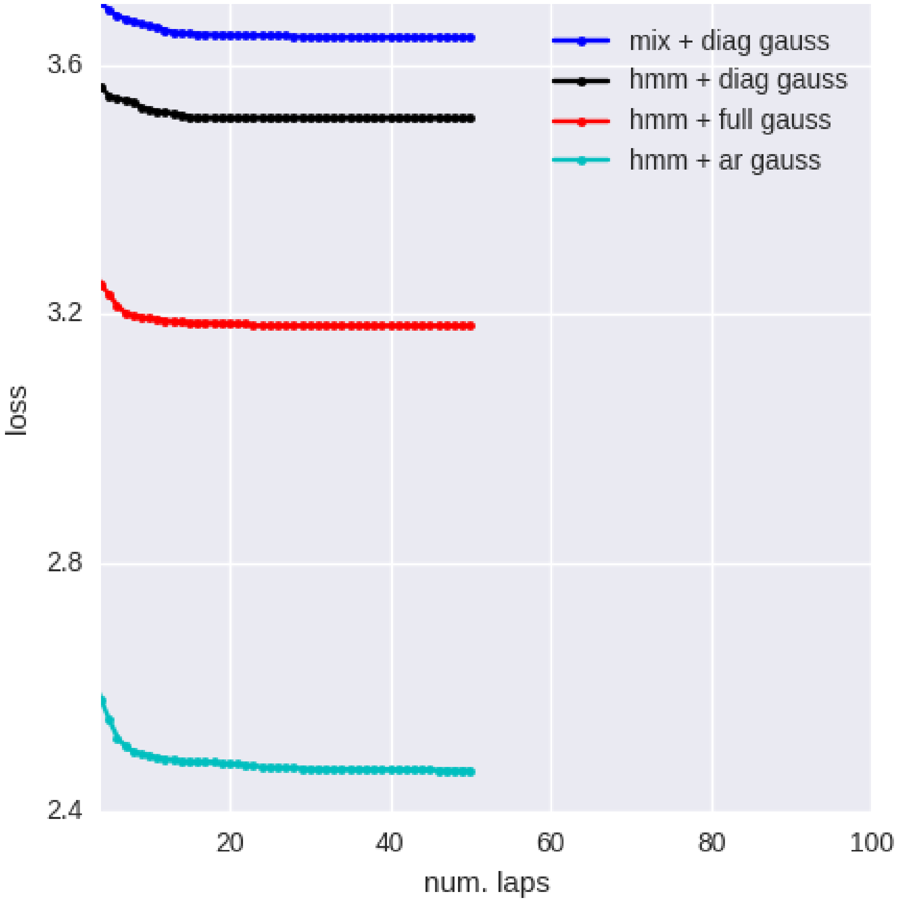
```
hmmar_trained_model, hmmar_info_dict = bnpy.run(
    dataset, 'HDPHMM', 'AutoRegGauss', 'memoVB',
    output_path='/tmp/mocap6/showcase-K=20-model=HDPHMM+AutoRegGauss-ECovMat=1*eye/',
    nLap=50, nTask=1, nBatch=1, convergeThr=0.0001,
    alpha=alpha, gamma=gamma, sF=sF, ECovMat='eye',
    K=K, initname='randexamples',
    )
```

Out:

```
WARNING: Found unrecognized keyword args. These are ignored.
  --alpha
Dataset Summary:
GroupXData
  total size: 6 units
  batch size: 6 units
  num. batches: 1
Allocation Model:  None
Obs. Data  Model:  Auto-Regressive Gaussian with full covariance.
Obs. Data  Prior:  MatrixNormal-Wishart on each mean/prec matrix pair: A, Lam
  E[ A ] =
  [[ 1.  0.]
   [ 0.  1.]] ...
  E[ Sigma ] =
  [[ 1.  0.]
   [ 0.  1.]] ...
Initialization:
  initname = randexamples
  K = 20 (number of clusters)
  seed = 1607680
  elapsed_time: 0.0 sec
Learn Alg: memoVB | task  1/1 | alg. seed: 1607680 | data order seed: 8541952
task_output_path: /tmp/mocap6/showcase-K=20-model=HDPHMM+AutoRegGauss-ECovMat=1*eye/1
    1.000/50 after      0 sec. |    179.0 MiB | K   20 | loss  2.908127656e+00 |
    2.000/50 after      1 sec. |    179.0 MiB | K   20 | loss  2.712797722e+00 | Ndiff   53
    3.000/50 after      1 sec. |    179.0 MiB | K   20 | loss  2.623844667e+00 | Ndiff   51
    4.000/50 after      1 sec. |    179.0 MiB | K   20 | loss  2.579060900e+00 | Ndiff   47
    5.000/50 after      1 sec. |    179.0 MiB | K   20 | loss  2.548616789e+00 | Ndiff   37
    6.000/50 after      2 sec. |    179.0 MiB | K   20 | loss  2.518588127e+00 | Ndiff   28
    7.000/50 after      2 sec. |    179.0 MiB | K   20 | loss  2.504804864e+00 | Ndiff   10
    8.000/50 after      2 sec. |    179.0 MiB | K   20 | loss  2.496420689e+00 | Ndiff    3
    9.000/50 after      2 sec. |    179.0 MiB | K   20 | loss  2.492445130e+00 | Ndiff    1
   10.000/50 after      3 sec. |    179.0 MiB | K   20 | loss  2.490165194e+00 | Ndiff    1
   11.000/50 after      3 sec. |    179.0 MiB | K   20 | loss  2.485730707e+00 | Ndiff    4
   12.000/50 after      3 sec. |    179.0 MiB | K   20 | loss  2.484287434e+00 | Ndiff    2
   13.000/50 after      3 sec. |    179.0 MiB | K   20 | loss  2.482746506e+00 | Ndiff    2
   14.000/50 after      4 sec. |    179.0 MiB | K   20 | loss  2.480880306e+00 | Ndiff    2
   15.000/50 after      4 sec. |    179.0 MiB | K   20 | loss  2.480467067e+00 | Ndiff    1
   16.000/50 after      4 sec. |    179.0 MiB | K   20 | loss  2.479991235e+00 | Ndiff
   17.000/50 after      5 sec. |    179.0 MiB | K   20 | loss  2.479792583e+00 | Ndiff
   18.000/50 after      5 sec. |    179.0 MiB | K   20 | loss  2.478520054e+00 | Ndiff
   19.000/50 after      5 sec. |    179.0 MiB | K   20 | loss  2.477365552e+00 | Ndiff
   20.000/50 after      5 sec. |    179.0 MiB | K   20 | loss  2.476525301e+00 | Ndiff
   21.000/50 after      6 sec. |    179.0 MiB | K   20 | loss  2.476119048e+00 | Ndiff
   22.000/50 after      6 sec. |    179.0 MiB | K   20 | loss  2.474839041e+00 | Ndiff
   23.000/50 after      6 sec. |    179.0 MiB | K   20 | loss  2.473186959e+00 | Ndiff
   24.000/50 after      6 sec. |    179.0 MiB | K   20 | loss  2.471452324e+00 | Ndiff
   25.000/50 after      7 sec. |    179.0 MiB | K   20 | loss  2.471162626e+00 | Ndiff
   26.000/50 after      7 sec. |    179.0 MiB | K   20 | loss  2.470983405e+00 | Ndiff
   27.000/50 after      7 sec. |    179.0 MiB | K   20 | loss  2.470774342e+00 | Ndiff
   28.000/50 after      7 sec. |    179.0 MiB | K   20 | loss  2.469880859e+00 | Ndiff
   29.000/50 after      8 sec. |    179.0 MiB | K   20 | loss  2.468970839e+00 | Ndiff
   30.000/50 after      8 sec. |    179.0 MiB | K   20 | loss  2.467972189e+00 | Ndiff
   31.000/50 after      8 sec. |    179.0 MiB | K   20 | loss  2.467661612e+00 | Ndiff
   32.000/50 after      9 sec. |    179.0 MiB | K   20 | loss  2.467483397e+00 | Ndiff
   33.000/50 after      9 sec. |    179.0 MiB | K   20 | loss  2.467450274e+00 | Ndiff
   34.000/50 after      9 sec. |    179.0 MiB | K   20 | loss  2.467407757e+00 | Ndiff
   35.000/50 after      9 sec. |    179.0 MiB | K   20 | loss  2.467292085e+00 | Ndiff
   36.000/50 after     10 sec. |    179.0 MiB | K   20 | loss  2.467236037e+00 | Ndiff
   37.000/50 after     10 sec. |    179.0 MiB | K   20 | loss  2.467197165e+00 | Ndiff
   38.000/50 after     10 sec. |    179.0 MiB | K   20 | loss  2.467112796e+00 | Ndiff
   39.000/50 after     10 sec. |    179.0 MiB | K   20 | loss  2.466989563e+00 | Ndiff
   40.000/50 after     11 sec. |    179.0 MiB | K   20 | loss  2.466727224e+00 | Ndiff
   41.000/50 after     11 sec. |    179.0 MiB | K   20 | loss  2.466587267e+00 | Ndiff
   42.000/50 after     11 sec. |    179.0 MiB | K   20 | loss  2.466255511e+00 | Ndiff
   43.000/50 after     11 sec. |    179.0 MiB | K   20 | loss  2.466170740e+00 | Ndiff
   44.000/50 after     12 sec. |    179.0 MiB | K   20 | loss  2.466133408e+00 | Ndiff
   45.000/50 after     12 sec. |    179.0 MiB | K   20 | loss  2.466086219e+00 | Ndiff
   46.000/50 after     12 sec. |    179.0 MiB | K   20 | loss  2.466016271e+00 | Ndiff
   47.000/50 after     12 sec. |    179.0 MiB | K   20 | loss  2.465964032e+00 | Ndiff
   48.000/50 after     13 sec. |    179.0 MiB | K   20 | loss  2.465914037e+00 | Ndiff
   49.000/50 after     13 sec. |    179.0 MiB | K   20 | loss  2.465798065e+00 | Ndiff
   50.000/50 after     13 sec. |    179.0 MiB | K   20 | loss  2.465630615e+00 | Ndiff
... done. not converged. max laps thru data exceeded.
```

Compare loss function traces for all methods

```
pylab.figure()

pylab.plot(
    mixdiag_info_dict['lap_history'],
    mixdiag_info_dict['loss_history'], 'b.-',
    label='mix + diag gauss')
pylab.plot(
    hmmdiag_info_dict['lap_history'],
    hmmdiag_info_dict['loss_history'], 'k.-',
    label='hmm + diag gauss')
pylab.plot(
    hmmfull_info_dict['lap_history'],
    hmmfull_info_dict['loss_history'], 'r.-',
    label='hmm + full gauss')
pylab.plot(
    hmmar_info_dict['lap_history'],
    hmmar_info_dict['loss_history'], 'c.-',
    label='hmm + ar gauss')
pylab.legend(loc='upper right')
pylab.xlabel('num. laps')
pylab.ylabel('loss')
pylab.xlim([4, 100]) # avoid early iterations
pylab.ylim([2.4, 3.7]) # handpicked
pylab.draw()
pylab.tight_layout()
```



Total running time of the script: (0 minutes 33.511 seconds)

`Download Python source code: plot-01-demo=many_models_same_data.py`

`Download Jupyter notebook: plot-01-demo=many_models_same_data.ipynb`

Generated by Sphinx-Gallery

`Download all examples in Python source code: examples_python.zip`

Download all examples in Jupyter notebooks: `examples_jupyter.zip`

Generated by Sphinx-Gallery

5.3 Allocation Model Guide

5.3.1 What is an allocation model?

Within **bnpy**, every hierarchical model we support has two pieces: an allocation model and an observation model. We use the label "allocation model" to describe the generative process that allocates cluster assignments to individual data points.

TODO ILLUSTRATION

In this document, we give a high-level overview of how we define an allocation model and how variational inference works. We also define the essential variational inference API functions that any concrete allocation model (an instance of the abstract `AllocModel` class) should support.

5.3.2 Quick Links

Here are some quick links to documentation for each of the possible allocation models supported by bnpy.

- Mixture models
- Topic models
- Hidden Markov models

5.3.3 Generative model

An allocation model defines a probabilistic generative process for assigning (aka allocating) clusters to data atoms. There are two types of variables involved: cluster probability vectors $pi_j$, and discrete assignments $z_n$ at each data aton indexed by $n$. Each allocation model defines a joint distribution

$$\log p(\pi, z) = \log p(\pi) + \log p(z|\pi)$$

First, we generate a set of global cluster probabilities $pi_0$.

$$\pi_0 \sim \mathrm{Dir}_K(\frac{\alpha_0}{K})$$

Depending on the model, we may next generate several more cluster probability vectors $pi_j$.

Second, we draw cluster assignment variables $z_n$ at each data atom $n$.

$$z_n \sim \mathrm{Cat}(\pi_{j1}, \ldots \pi_{jK})$$

Example: Mixture model

For example, consider a simple finite mixture model with K clusters. The complete allocation model would be:

$$\pi_0 \sim \text{Dir}_K(\alpha_0 \frac{1}{K})$$
$$z_n \sim \text{Cat}(\pi_{01}, \dots \pi_{0K})$$

To extend this to a Dirichlet process mixture model, we simply use a stick-breaking distribution instead:

$$\pi_0 \sim \text{Stick}(\alpha_0)$$
$$z_n \sim \text{Cat}(\pi_{01}, \dots \pi_{0K}, \dots)$$

5.3.4 Variational Inference

Variational inference for allocation models tries to optimize an approximate posterior:

$$\log q(\pi, z) = \log q(\pi|\theta) + \log q(z|r)$$

The optimization objective is to make this approximate posterior as close to the true posterior as possible. Remember that this objective incorporates terms from the observation model as well. The optimization finds values for the free parameters – pseudo-counts theta and assignments r – that make the objective function as large as possible.

$$= \mathcal{L}_{\text{alloc}}(r, theta) + (r, ...)$$

Expanding the allocation model terms, we have

$$\mathcal{L}_{\text{alloc}}(r, theta) = +\mathcal{L}_{\text{entropy}}$$
$$= \mathbb{E}_q[\log p(z) + \frac{\log p(\pi)}{\log q(\pi)}]$$
$$\mathcal{L}_{\text{entropy}} = -\mathbb{E}_q[\log q(z)]$$
$$\log p(z|\alpha) \geq$$

Every variational algorithm proceeds by iteratively improving this objective function by cycling through four concrete steps:

- Local step: optimize the local assignments r and any local theta values.
- Summary step: compute summary statistics from the local parameters.
- Global step:
- Objective function evaluation step

5.3.5 Variational API

Within **bnpy**, each possible allocation model is a subclass of the general-purpose abstract base class: `AllocModel`. Each `AllocModel` instance has both state and behaviors. The *state* represents two key values: the hyperparameters that define the prior and the global variational parameters that define the approximate posterior. The *behaviors* are the four fundamental steps of inference, as well as some auxiliary functions.

Attributes

For any generative model in our framework, the hyperparameters of an allocation model are just the set of concentration parameters $alpha_j$ that parameterize the generative story for each pi_j probability vector. Thus, each allocation model will hold one or more *alpha* values as attributes.

Each `AllocModel` subclass will have model-specific global parameters, which are represented as instance attributes. For example, a `FiniteMixtureModel` has a vector of Dirichlet pseudo-counts called *theta*, while a `DPMixtureModel` instance has a vector of Beta pseudo-counts called *eta*.

Each of the four conceptual steps of the variational inference – local step, summary step, global step, and objective step – is associated with a single instance-level function of an AllocModel object. The general abstract interface for using these functions is documented below. Each subclass will provide an actual implementation of these functions.

Local step

The local step, specified by calc_local_params, finds local parameters for the dataset.

class `bnpy.allocmodel.`**`AllocModel`**(*inferType*)

`calc_local_params`(*Data*, *LP*)

Compute local parameters for each data item and component.

This is the E-step of EM algorithm.

Returned LP contains optimal values of local parameters specific to the provided dataset. Updated values computed using current global parameter attributes.

Possible keyword arguments control model-specific computations.

Parameters

- **`Data`** (`DataObj`) – Dataset to compute local parameters for.
- **`LP`** (*`dict`*) – Must contain cond. likelihoods in field 'E_log_soft_ev', a 2D array that is N x K provided by the observation model.

Returns LP (*dict*) – Contains updated fields for all K clusters in current model. * 'resp' : N x K 2D array, soft assignments for each data atom.

Summary step

The summary step, specified by get_global_suff_stats, summarizes a dataset Data and its associated local parameters LP. It produces a bag of sufficient statistics SS.

class `bnpy.allocmodel.`**`AllocModel`**(*inferType*)

`get_global_suff_stats`(*Data*, *SS*, *LP*, ***kwargs*)

Compute low-dim summaries for provided local params.

Returned sufficient statistics are deterministic given Data, LP.

Possible keyword arguments control model-specific computations.

Parameters

- **Data** (DataObj) – Dataset to be summarized.
- **SS** (SuffStatBag) – If present, all summaries will be added to this bag. If None, new bag will be created and returned.
- **LP** (*dict*) – Holds valid local params for K' clusters and all atoms in Data.

Returns **SS** (SuffStatBag) – Updated fields for each of K' clusters represented in LP

Global step

The global step, performed by update_global_params,

class bnpy.allocmodel.**AllocModel**(*inferType*)

get_global_suff_stats(*Data*, *SS*, *LP*, ***kwargs*)
Compute low-dim summaries for provided local params.

Returned sufficient statistics are deterministic given Data, LP.

Possible keyword arguments control model-specific computations.

Parameters

- **Data** (DataObj) – Dataset to be summarized.
- **SS** (SuffStatBag) – If present, all summaries will be added to this bag. If None, new bag will be created and returned.
- **LP** (*dict*) – Holds valid local params for K' clusters and all atoms in Data.

Returns **SS** (SuffStatBag) – Updated fields for each of K' clusters represented in LP

Objective evaluation step

During inference, we need to verify that each step is working as expected. Thus, we need to be able to compute the scalar value of the objective given any current set of global parameters (stored in self) and local parameters (summarized in SS).

class bnpy.allocmodel.**AllocModel**(*inferType*)

calc_evidence(*Data*, *SS*, *LP*, *todict=0*, ***kwargs*)
Calculate ELBO objective function value for provided state.

Parameters

- **Data** (*optional,*) – If not provided, relies exclusively on summaries in SS
- **SS** (SuffStatBag) – Contains valid summaries for desired dataset.
- **LP** (*optional, dict*) – If not provided, relies exclusively on summaries in SS If provided, used in place of summaries in SS when possible.

Keyword Arguments **todict** (*boolean*) – If True, return a dict with different ELBO terms

under named keys like 'Ldata' and 'Lentropy'

If False [default], return scalar value equal to sum of terms.

Returns **L** (*float*) – Represents sum of all terms in optimization objective. Will be a dict if todict option is True.

Mixture Models

bnpy supports two kinds of mixture models: *FiniteMixtureModel* and *DPMixtureModel*.

FiniteMixtureModel The finite mixture has the following generative representation as an allocation model. There is a single top-level vector of cluster probabilities π_0. Each data atom's assignment is drawn i.i.d. according to the probabilities in this vector.

$$\begin{aligned}[\pi_{01}, \pi_{02}, \dots \pi_{0K}] &\sim \mathrm{Dir}_K(\frac{\alpha_0}{K}) \\ \text{for } n &\in 1, \dots N : \\ z_n &\sim \mathrm{Cat}_K(\pi_{01}, \pi_{0K})\end{aligned}$$

Here, $\alpha_0 > 0$ is the uniform concentration parameter. TODO interpret.

DPMixtureModel The Dirichlet Process (DP) mixture has the following generative representation as an allocation model. It modifies the finite mixture by using the StickBreaking process to K active weights and a remainder weight, all inside pi_0.

$$\begin{aligned}[\pi_{01}, \pi_{02}, \dots \pi_{0K}, \pi_{0,>K}] &\sim \mathrm{StickBreaking}_K(\pi_0) \\ \text{for } n &\in 1, \dots N : \\ z_n &\sim \mathrm{Cat}_K(\pi_{01}, \pi_{0K})\end{aligned}$$

If we take the limit as K grows to infinity, these two generative models are equivalent.

Using mixtures with other bnpy modules As usual, to train a hierarchical model whose allocation is done by FiniteMixtureModel,

Supported DataObj Types Mixture models can apply to almost all data formats available in bnpy. Any data suitable for topic models or sequence models can also be fit with a basic mixture model.

The only formats that do not apply are those based on GraphData, which require the subclass of mixture models (TBD).

Supported Learning Algorithms Currently, the practical differences are:

- *FiniteMixtureModel* supports EM, VB, soVB, moVB
- *DPMixtureModel* supports VB, soVB, and moVB.
-
    - with birth/merge/delete moves for moVB

EM (MAP) inference for the DPMixtureModel is possible, but just not implemented yet.

Common tasks with mixtures

Accessing learned cluster assignments Given a dataset of interest Data (a `DataObj`), and an hmodel (an instance of `HModel`) properly initialized with K active clusters, we simply perform a local step.

```
>>> LP = hmodel.calc_local_params(Data)
>>> resp = LP['resp']
```

Here, resp is a 2D array of size N x K. Each entry resp[n, k] gives the probability that data atom n is assigned to cluster k under the posterior. Thus, each entry resp[n,k] must be a value within the interval [0,1]. The sum of every row must equal one.

```
>>> assert resp[n, k] >= 0.0
>>> assert resp[n, k] <= 1.0
>>> assert np.allclose(np.sum(resp[n,:]), 1.0)
```

To convert to hard assignments

```
>>> Z = resp.argmax(axis=1)
```

Here, Z is a 1D array of size N, where entry Z[n] is an integer in the set {0, 1, 2, ... K-1, K}.

Accessing learned cluster probabilities

```
>>> pi0 = hmodel.allocModel.get_active_cluster_probs()
>>> assert pi0.ndim == 1
>>> assert pi0.size == hmodel.allocModel.K
```

Global update summaries For a global update, mixture models require only one sufficient statistic: an expected count value for each cluster k. This value gives the expected number of data atoms assigned to k throughout the dataset.

- **Count N_k** Expected assignments to state k across all data items.

```
>>> LP = hmodel.calc_local_params(Data)
>>> SS = hmodel.get_global_suff_stats(Data, LP)
>>> Nvec = SS.N # or SS.getCountVec()
>>> assert Nvec.size == hmodel.allocModel.K
[ ... TODO ... ]
```

ELBO summaries To compute the ELBO, mixture models require only one non-linear summary statistic: the entropy of the learned assignment parameters *resp*.

$$\begin{aligned} &= +\mathcal{L}_{\text{alloc}} - E[\log q(z)] \\ -E[\log q(z)] &= \sum_{k=1}^{K} H_k \\ H_k &= -\sum_{n=1}^{N} r_{nk} \log r_{nk} \end{aligned}$$

You can compute this by enabling the correct keyword flag when calling the summary step function.

```
>>> LP = hmodel.calc_local_params(Data)
>>> SS = hmodel.get_global_suff_stats(Data, LP, doPrecompEntropy=1)
>>> Hresp =  SS.getELBOTerm('Hresp')
>>> assert Hresp.ndim == 1
>>> assert Hresp.size == SS.K
[ ... TODO ... ]
```

Topic Models

Supported Data Formats Topic models can be applied to any dataset that has group structure.

Supported Learning Algorithms

- *FiniteTopicModel* supports VB, soVB, moVB
- *HDPTopicModel* supports VB, soVB, and moVB. * with birth/merge/delete moves for moVB

Possible Implementations

- FiniteTopicModel: stuff here
- HDPTopicModel: more stuff here

There are two types of mixture model supported. Both define the model in terms of a global parameter vector β, where β_k gives the probability of topic k, and local assignments z, where z_n indicates which state {1, 2, 3, ... K} is assigned to data item n.

The *FiniteMixtureModel* has a generative process:

$$\begin{aligned} [\beta_1, \beta_2, \ldots \beta_K] &\sim \text{Dir}(\gamma, \gamma, \ldots \gamma) \\ z_n &\sim \text{Discrete}(\beta) \end{aligned}$$

while the *DPMixtureModel* has generative process:

$$\begin{aligned} [\beta_1, \beta_2, \ldots \beta_K \ldots] &\sim \text{StickBreaking}(\gamma_0) \\ z_n &\sim \text{Discrete}(\beta) \end{aligned}$$

If we let K grow to infinity, these two models converge if $\gamma = \gamma_0/K$.

TOC

5.4 Observation Models

All observation models define a *likelihood* for producing data x_n from some cluster-specific density with parameter ϕ_k:

$$p(x|\phi, z) = \prod_{n=1}^{N} p(x_n|\phi_k)^{\delta_k(z_n)}$$

Supported Bayesian methods require specifying a (conjugate) prior:

$$p(\phi) = \prod_{k=1}^{K} p(\phi_k)$$

5.4.1 Variational methods for observation models

The links below describe the mathematical and computational details for performing standard variational optimization for supported observation models:

Zero-mean full-covariance Gaussian observation model: Variational Methods

TODO update this page

Generative model

The diagonal Gaussian observation model generates each data vector x_n of length D from a multivariate Gaussian with mean $\mu_k \in \mathbb{R}^D$ and a diagonal covariance matrix:

$$\begin{matrix} x_{n1} \\ x_{n2} \\ \vdots \\ x_{nD} \end{matrix} \sim \mathcal{N} \left(\begin{matrix} \mu_{k1} \\ \mu_{k2} \\ \vdots \\ \mu_{kD} \end{matrix}, \begin{matrix} \lambda_{k1}^{-1} & & & \\ & \lambda_{k2}^{-1} & & \\ & & \ddots & \\ & & & \lambda_{kD}^{-1} \end{matrix} \right)$$

Global Random Variables The global random variables are the cluster-specific means and precisions (inverse variances).

For each cluster *k*, we have the following global random variables:

$$\begin{aligned} \mu_{k1}, \mu_{k2}, \ldots \mu_{kD} \qquad & \mu_{kd} \in \mathbb{R} \\ \lambda_{k1}, \lambda_{k2}, \ldots \lambda_{kD} \qquad & \lambda_{kd} \in (0, +\infty) \end{aligned}$$

Local Random Variables Each dataset observation at index *n* has its own cluster assignment:

$$z_n \in \{1, 2, \dots K\}$$

The generative model and approximate posterior for z_n is determined by an allocation model. For all computations needed by our current observation model, we'll assume either a point estimate or an approximate posterior for z_n is known.

Normal Wishart prior

Each dimension *d* has a mean μ_{kd} and variance λ_{kd} which have a joint univariate Normal-Wishart prior with scalar hyperparameters $\bar{\nu}, \bar{\beta}_d$ for the Wishart prior and then $\bar{m}_d, \bar{\kappa}$ for the Normal prior:

$$\lambda_{kd} \sim \mathcal{W}_1(\bar{\nu}, \bar{\beta}_d)$$
$$\mu_{kd} \sim \mathcal{N}_1(\bar{m}_d, \bar{\kappa}^{-1} \lambda_{kd}^{-1})$$

These are represented by the following numpy array attributes of the `Prior` parameter bag:

- **`nu`** [float] degrees of freedom
- **`beta`** [1D array, size D] scale parameters that set mean of lambda
- **`m`** [1D array, size D] mean of the parameter mu
- **`kappa`** [float] scalar precision of mu

Several keyword arguments can be used to determine the values of the prior hyperparameters when calling bnpy.run

- **`--nu`** [float] Sets value of $\bar{\nu}$. Defaults to D + 2.
- **`--kappa`** [float] Sets value of $\bar{\kappa}$. Defaults to ???.
- **`--ECovMat`** [str] Determines the expected value of data covariance under the prior. Possible values include 'eye' and 'diagcovdata'. TODO
- **`--sF`** [float] These two options set the value of $\bar{\beta}$. TODO.
- TODO set m??

Approximate posterior

We assume the following factorized approximate posterior family for variational optimization:

$$q(z, \mu, \lambda) = \prod_{n=1}^{N} q(z_n) \cdot \prod_{k=1}^{K} (\mu_k, \lambda_k)$$

The specific forms of the global and local factors are given below.

Posterior for local assignments For each observation vector at index *n*, we assume an independent approximate posterior over the assigned cluster indicator $z_n \in \{1, 2, \dots K\}$.

$$\begin{aligned} q(z) &= \prod_{n=1}^{N} q(z_n | \hat{r}_n) \\ &= \prod_{n=1}^{N} \text{Discrete}(z_n | \hat{r}_{n1}, \hat{r}_{n2}, \dots \hat{r}_{nK}) \end{aligned}$$

Thus, for this observation model the only local variational parameter is the assignment responsibility array $\hat{r} = \{\{\hat{r}_{nk}\}_{k=1}^{K}\}_{n=1}^{N}$.

Inside the *LP* dict, this is represented by the *resp* numpy array:

- **resp** [2D array, size N x K] Parameters of approximate posterior q(z) over cluster assignments. resp[n,k] = probability observation n is assigned to component k.

Remember, all computations required by our observation model assume that the `resp` array is given. The actual values of `resp` are updated by an allocation model.

Posterior for global parameters The goal of variational optimization is to find the best approximate posterior distribution for the mean and precision parameters of each cluster *k*:

$$\begin{aligned} q(\mu, \lambda) &= \prod_{k=1}^{K} \prod_{d=1}^{D} q(\mu_{kd}, \lambda_{kd}) \\ &= \prod_{k=1}^{K} \prod_{d=1}^{D} \mathcal{W}_1(\lambda_{kd} | \hat{\nu}_k, \hat{\beta}_{kd}) \mathcal{N}_1(\mu_{kd} | \hat{m}_{kd}, \hat{\kappa}_k^{-1} \lambda_{kd}^{-1}) \end{aligned}$$

This approximate posterior is represented by the *Post* attribute of the *DiagGaussObsModel*. This is a ParamBag with the following attributes:

- **K** [int] number of active clusters
- **nu** [1D array, size K] Defines $\hat{\nu}_k$ for each cluster
- **beta** [2D array, size K x D] Defines $\hat{\beta}_{kd}$ for each cluster and dimension
- **m** [2D array, size K x D] Defines $\hat{m}_{kd}$ for each cluster and dimension
- **kappa** [2D array, size K] Defines $\hat{\kappa}_k$ for each cluster

Objective function Variational optimization will find the approximate posterior parameters that maximize the following objective function, given a fixed observed dataset $x = \{x_1, \dots x_N\}$ and fixed prior hyparparameters $\bar{\nu}, \bar{\beta}, \bar{m}, \bar{\kappa}$.

$$\begin{aligned}
\mathcal{L}^{\text{DiagGauss}}(\hat{\nu}, \hat{\beta}, \hat{m}, \hat{\kappa}) = & \sum_{k=1}^{K}\sum_{d=1}^{D} c^{\text{NW}}_{1,1}(\hat{\nu}_k, \hat{\beta}_{kd}, \hat{m}_{kd}, \hat{\kappa})_k - c^{\text{NW}}_{1,1}(\bar{\nu}, \bar{\beta}_d, \bar{m}_d, \bar{\kappa}) \\
& + \frac{1}{2}\sum_{k=1}^{K}\sum_{d=1}^{D} (N_k(\hat{r}) + \bar{\nu} - \hat{\nu}_k)\, \mathbb{E}_q[\log \lambda_{kd}] \\
& - \frac{1}{2}\sum_{k=1}^{K}\sum_{d=1}^{D} (N_k(\hat{r}) + \bar{\kappa} - \hat{\kappa}_k)\, \mathbb{E}_q[\lambda_{kd}] \\
& + \sum_{k=1}^{K}\sum_{d=1}^{D} (S^{x}_{kd}(x, \hat{r}) + \bar{\kappa}\bar{m}_d - \hat{\kappa}_k \hat{m}_{kd})\, \mathbb{E}_q[\lambda_{kd}\mu_{kd}] \\
& - \frac{1}{2}\sum_{k=1}^{K}\sum_{d=1}^{D} \left(S^{x^2}_{kd}(x, \hat{r}) + \bar{\beta}_d + \bar{\kappa}\bar{m}_d^2 - \hat{\beta}_{kd} - \hat{\kappa}_k \hat{m}_{kd}^2\right) \mathbb{E}_q[\lambda_{kd}\mu^2_{kd}]
\end{aligned}$$

This objective function is computed by calling the Python function `calc_evidence`.

Sufficient statistics The sufficient statistics of this observation model are functions of the local parameters $\hat{r}$ and the observed data x.

$$\begin{aligned}
N_k(\hat{r}) &= \sum_{n=1}^{N} \hat{r}_{nk} \\
S^{x}_{kd}(x, \hat{r}) &= \sum_{n=1}^{N} \hat{r}_{nk} x^2_{nd} \\
S^{x^2}_{kd}(x, \hat{r}) &= \sum_{n=1}^{N} \hat{r}_{nk} x^2_{nd}
\end{aligned}$$

These fields are stored within the sufficient statistics parameter bag `SS` as the following fields:

- **`SS.N`** [1D array, size K] SS.N[k] = N_k
- **`SS.x`** [2D array, size K x D] SS.x[k,d] = $S^{x}_{kd}(x, \hat{r})$
- **`SS.xx`** [2D array, size K x D] SS.xx[k,d] = $S^{x^2}_{kd}(x, \hat{r})$

Cumulant function The cumulant function of the univariate Normal-Wishart is evaluated for each dimension d separately. The function takes 4 scalar input arguments and produces a scalar output.

$$c^{\text{NW}}_{1,1}(\nu, \beta_d, m_d, \kappa) = -\frac{1}{2}\log 2\pi + \frac{1}{2}\log \kappa + \frac{\nu}{2}\log\frac{\beta_d}{2} - \log\Gamma\left(\frac{\nu}{2}\right)$$

Coordinate Ascent Updates

Local step update As with all observation models, the local step computes the *expected* log conditional probability of assigning each observation to each cluster:

$$\mathbb{E}[\log p(x_n|\mu_k, \lambda_k)] = -\frac{D}{2}\log 2\pi + \frac{1}{2}\sum_{d=1}^{D}\mathbb{E}[\log \lambda_{kd}] - \frac{1}{2}\sum_{d=1}^{D}\mathbb{E}[\lambda_{kd}(x_{nd} - \mu_{kd})^2]$$

where the elementary expectations required are:

$$\begin{aligned}\mathbb{E}[\log \lambda_{kd}] &= \psi\left(\frac{\hat{\nu}_k}{2}\right) - \log \frac{\hat{\beta}_{kd}}{2} \\ \mathbb{E}_q\left[\lambda_{kd}(x_{nd} - \mu_{kd})^2\right] &= \frac{1}{\hat{\kappa}_k} + \frac{\hat{\nu}_k}{\hat{\beta}_{kd}}(x_{nd} - \hat{m}_{kd})^2\end{aligned}$$

In our implementation, this is done via the function `calc_local_params`, which computes the following arrays and places them inside the local parameter dict `LP`.

- **`E_log_soft_ev`** [2D array, N x K] log probability of assigning each observation n to each cluster k

Global step update The global step update produces an updated approximate posterior over the global random variables. Concretely, this means updated values for each field of the `Post` ParamBag attribute of the DiagGaussObsModel.

$$\begin{aligned}\hat{\nu}_k &\leftarrow N_k(\hat{r}) + \bar{\nu} \\ \hat{\kappa}_k &\leftarrow N_k(\hat{r}) + \bar{\kappa} \\ \hat{m}_{kd} &\leftarrow \frac{1}{\hat{\kappa}_k}\left(S_k^x(x, \hat{r}) + \bar{\kappa}\bar{m}_d\right) \\ \hat{\beta}_{kd} &\leftarrow S_{kd}^{x^2}(x, \hat{r}) + \bar{\beta}_d + \bar{\kappa}\bar{m}_d^2 - \hat{\kappa}_k\hat{m}_{kd}^2\end{aligned}$$

Our implementation performs this update when calling the function `update_global_params`.

Initialization

Initialization creates valid values of the parameters which define the approximate posterior over the global random variables. Concretely, this means it creates a valid setting of the `Post` attribute of the DiagGaussObsModel object.

TODO

Diagonal-covariance Gaussian observation model: Variational Bayesian Methods

Generative model

The diagonal Gaussian observation model generates each data vector x_n of length D from a multivariate Gaussian with mean $\mu_k \in \mathbb{R}^D$ and a diagonal covariance matrix:

$$\begin{matrix} x_{n1} \\ x_{n2} \\ \vdots \\ x_{nD} \end{matrix} \sim \mathcal{N} \left(\begin{matrix} \mu_{k1} \\ \mu_{k2} \\ \vdots \\ \mu_{kD} \end{matrix}, \begin{matrix} \lambda_{k1}^{-1} & & & \\ & \lambda_{k2}^{-1} & & \\ & & \ddots & \\ & & & \lambda_{kD}^{-1} \end{matrix} \right)$$

Global Random Variables The global random variables are the cluster-specific means and precisions (inverse variances).

For each cluster *k*, we have the following global random variables:

$$\begin{matrix} \mu_{k1}, \mu_{k2}, \dots \mu_{kD} & \qquad \mu_{kd} \in \mathbb{R} \\ \lambda_{k1}, \lambda_{k2}, \dots \lambda_{kD} & \qquad \lambda_{kd} \in (0, +\infty) \end{matrix}$$

Local Random Variables Each dataset observation at index *n* has its own cluster assignment:

$$z_n \in \{1, 2, \dots K\}$$

The generative model and approximate posterior for z_n is determined by an allocation model. For all computations needed by our current observation model, we'll assume either a point estimate or an approximate posterior for z_n is known.

Normal Wishart prior

Each dimension *d* has a mean μ_{kd} and variance λ_{kd} which have a joint univariate Normal-Wishart prior with scalar hyperparameters $\bar{\nu}, \bar{\beta}_d$ for the Wishart prior and then $\bar{m}_d, \bar{\kappa}$ for the Normal prior:

$$\begin{aligned} \lambda_{kd} &\sim \mathcal{W}_1(\bar{\nu}, \bar{\beta}_d) \\ \mu_{kd} &\sim \mathcal{N}_1(\bar{m}_d, \bar{\kappa}^{-1}\lambda_{kd}^{-1}) \end{aligned}$$

These are represented by the following numpy array attributes of the `Prior` parameter bag:

- **`nu`** [float] degrees of freedom
- **`beta`** [1D array, size D] scale parameters that set mean of lambda
- **`m`** [1D array, size D] mean of the parameter mu
- **`kappa`** [float] scalar precision of mu

Several keyword arguments can be used to determine the values of the prior hyperparameters when calling bnpy.run

- **`--nu`** [float] Sets value of $\bar{\nu}$. Defaults to D + 2.

- **--kappa** [float] Sets value of $\bar{\kappa}$. Defaults to ???.
- **--ECovMat** [str] Determines the expected value of data covariance under the prior. Possible values include 'eye' and 'diagcovdata'. TODO
- **--sF** [float] These two options set the value of $\bar{\beta}$. TODO.
- TODO set m??

Approximate posterior

We assume the following factorized approximate posterior family for variational optimization:

$$q(z, \mu, \lambda) = \prod_{n=1}^{N} q(z_n) \cdot \prod_{k=1}^{K} (\mu_k, \lambda_k)$$

The specific forms of the global and local factors are given below.

Posterior for local assignments For each observation vector at index *n*, we assume an independent approximate posterior over the assigned cluster indicator $z_n \in \{1, 2, \dots K\}$.

$$\begin{aligned} q(z) &= \prod_{n=1}^{N} q(z_n|\hat{r}_n) \\ &= \prod_{n=1}^{N} \text{Discrete}(z_n|\hat{r}_{n1}, \hat{r}_{n2}, \dots \hat{r}_{nK}) \end{aligned}$$

Thus, for this observation model the only local variational parameter is the assignment responsibility array $\hat{r} = \{\{\hat{r}_{nk}\}_{k=1}^{K}\}_{n=1}^{N}$.

Inside the *LP* dict, this is represented by the *resp* numpy array:

- **resp** [2D array, size N x K] Parameters of approximate posterior q(z) over cluster assignments. resp[n,k] = probability observation n is assigned to component k.

Remember, all computations required by our observation model assume that the `resp` array is given. The actual values of `resp` are updated by an allocation model.

Posterior for global parameters The goal of variational optimization is to find the best approximate posterior distribution for the mean and precision parameters of each cluster *k*:

$$\begin{aligned} q(\mu, \lambda) &= \prod_{k=1}^{K} \prod_{d=1}^{D} q(\mu_{kd}, \lambda_{kd}) \\ &= \prod_{k=1}^{K} \prod_{d=1}^{D} \mathcal{W}_1(\lambda_{kd}|\hat{\nu}_k, \hat{\beta}_{kd}) \mathcal{N}_1(\mu_{kd}|\hat{m}_{kd}, \hat{\kappa}_k^{-1}\lambda_{kd}^{-1}) \end{aligned}$$

This approximate posterior is represented by the *Post* attribute of the *DiagGaussObsModel*. This is a ParamBag with the following attributes:

- **K** [int] number of active clusters
- **nu** [1D array, size K] Defines $\hat{\nu}_k$ for each cluster
- **beta** [2D array, size K x D] Defines $\hat{\beta}_{kd}$ for each cluster and dimension
- **m** [2D array, size K x D] Defines $\hat{m}_{kd}$ for each cluster and dimension
- **kappa** [2D array, size K] Defines $\hat{\kappa}_k$ for each cluster

Objective function Variational optimization will find the approximate posterior parameters that maximize the following objective function, given a fixed observed dataset $x = \{x_1, \dots x_N\}$ and fixed prior hyparparameters $\bar{\nu}, \bar{\beta}, \bar{m}, \bar{\kappa}$.

$$\begin{aligned}\mathcal{L}^{\text{DiagGauss}}(\hat{\nu}, \hat{\beta}, \hat{m}, \hat{\kappa}) &= -\frac{ND}{2} \log 2\pi \\ &+ \sum_{k=1}^{K} \sum_{d=1}^{D} c^{\text{NW}}_{1,1}(\hat{\nu}_k, \hat{\beta}_{kd}, \hat{m}_{kd}, \hat{\kappa})_k - c^{\text{NW}}_{1,1}(\bar{\nu}, \bar{\beta}_d, \bar{m}_d, \bar{\kappa}) \\ &+ \frac{1}{2} \sum_{k=1}^{K} \sum_{d=1}^{D} (N_k(\hat{r}) + \bar{\nu} - \hat{\nu}_k)\, \mathbb{E}_q[\log \lambda_{kd}] \\ &- \frac{1}{2} \sum_{k=1}^{K} \sum_{d=1}^{D} (N_k(\hat{r}) + \bar{\kappa} - \hat{\kappa}_k)\, \mathbb{E}_q[\lambda_{kd}] \\ &+ \sum_{k=1}^{K} \sum_{d=1}^{D} (S^{x}_{kd}(x, \hat{r}) + \bar{\kappa}\bar{m}_d - \hat{\kappa}_k \hat{m}_{kd})\, \mathbb{E}_q[\lambda_{kd}\mu_{kd}] \\ &- \frac{1}{2} \sum_{k=1}^{K} \sum_{d=1}^{D} \left(S^{x^2}_{kd}(x, \hat{r}) + \bar{\beta}_d + \bar{\kappa}\bar{m}_d^2 - \hat{\beta}_{kd} - \hat{\kappa}_k \hat{m}_{kd}^2\right) \mathbb{E}_q[\lambda_{kd}\mu_{kd}^2]\end{aligned}$$

This objective function is computed by calling the Python function `calc_evidence`.

Sufficient statistics The sufficient statistics of this observation model are functions of the local parameters $\hat{r}$ and the observed data x.

$$\begin{aligned}N_k(\hat{r}) &= \sum_{n=1}^{N} \hat{r}_{nk} \\ S^{x}_{kd}(x, \hat{r}) &= \sum_{n=1}^{N} \hat{r}_{nk} x_{nd}^2 \\ S^{x^2}_{kd}(x, \hat{r}) &= \sum_{n=1}^{N} \hat{r}_{nk} x_{nd}^2\end{aligned}$$

These fields are stored within the sufficient statistics parameter bag `SS` as the following fields:

- **SS.N** [1D array, size K] SS.N[k] = N_k
- **SS.x** [2D array, size K x D] SS.x[k,d] = $S^{x}_{kd}(x, \hat{r})$
- **SS.xx** [2D array, size K x D] SS.xx[k,d] = $S^{x^2}_{kd}(x, \hat{r})$

Cumulant function The cumulant function of the univariate Normal-Wishart is evaluated for each dimension d separately. The function takes 4 scalar input arguments and produces a scalar output.

$$c^{\mathrm{NW}}_{1,1}(\nu, \beta_d, m_d, \kappa) = \frac{1}{2}\log 2\pi - \frac{1}{2}\log\kappa - \frac{\nu}{2}\log\frac{\beta_d}{2} + \log\Gamma\left(\frac{\nu}{2}\right)$$

Coordinate Ascent Updates

Local step update As with all observation models, the local step computes the *expected* log conditional probability of assigning each observation to each cluster:

$$\mathbb{E}[\log p(x_n|\mu_k, \lambda_k)] = -\frac{D}{2}\log 2\pi + \frac{1}{2}\sum_{d=1}^{D}\mathbb{E}[\log\lambda_{kd}] - \frac{1}{2}\sum_{d=1}^{D}\mathbb{E}[\lambda_{kd}(x_{nd} - \mu_{kd})^2]$$

where the elementary expectations required are:

$$\begin{aligned}\mathbb{E}[\log\lambda_{kd}] &= \psi\left(\frac{\hat{\nu}_k}{2}\right) - \log\frac{\hat{\beta}_{kd}}{2} \\ \mathbb{E}_q\left[\lambda_{kd}(x_{nd} - \mu_{kd})^2\right] &= \frac{1}{\hat{\kappa}_k} + \frac{\hat{\nu}_k}{\hat{\beta}_{kd}}(x_{nd} - \hat{m}_{kd})^2\end{aligned}$$

In our implementation, this is done via the function `calc_local_params`, which computes the following arrays and places them inside the local parameter dict `LP`.

- **`E_log_soft_ev`** [2D array, N x K] log probability of assigning each observation n to each cluster k

Global step update The global step update produces an updated approximate posterior over the global random variables. Concretely, this means updated values for each field of the `Post` ParamBag attribute of the DiagGaussObsModel.

$$\begin{aligned}\hat{\nu}_k &\leftarrow N_k(\hat{r}) + \bar{\nu} \\ \hat{\kappa}_k &\leftarrow N_k(\hat{r}) + \bar{\kappa} \\ \hat{m}_{kd} &\leftarrow \frac{1}{\hat{\kappa}_k}\left(S^x_k(x, \hat{r}) + \bar{\kappa}\bar{m}_d\right) \\ \hat{\beta}_{kd} &\leftarrow S^{x^2}_{kd}(x, \hat{r}) + \bar{\beta}_d + \bar{\kappa}\bar{m}^2_d - \hat{\kappa}_k\hat{m}^2_{kd}\end{aligned}$$

Our implementation performs this update when calling the function `update_global_params`.

Initialization

Initialization creates valid values of the parameters which define the approximate posterior over the global random variables. Concretely, this means it creates a valid setting of the `Post` attribute of the DiagGaussObsModel object.

TODO

Full-covariance Gaussian observation model: Variational Bayesian Methods

TODO revise this!

Generative model

The diagonal Gaussian observation model generates each data vector x_n of length D from a multivariate Gaussian with mean $\mu_k \in \mathbb{R}^D$ and a diagonal covariance matrix:

$$\begin{matrix} x_{n1} \\ x_{n2} \\ \vdots \\ x_{nD} \end{matrix} \sim \mathcal{N}\left(\begin{matrix} \mu_{k1} \\ \mu_{k2} \\ \vdots \\ \mu_{kD} \end{matrix}, \begin{matrix} \lambda_{k1}^{-1} & & & \\ & \lambda_{k2}^{-1} & & \\ & & \ddots & \\ & & & \lambda_{kD}^{-1} \end{matrix} \right)$$

Global Random Variables The global random variables are the cluster-specific means and precisions (inverse variances).

For each cluster *k*, we have the following global random variables:

$$\begin{matrix} \mu_{k1}, \mu_{k2}, \ldots \mu_{kD} & \mu_{kd} \in \mathbb{R} \\ \lambda_{k1}, \lambda_{k2}, \ldots \lambda_{kD} & \lambda_{kd} \in (0, +\infty) \end{matrix}$$

Local Random Variables Each dataset observation at index *n* has its own cluster assignment:

$$z_n \in \{1, 2, \ldots K\}$$

The generative model and approximate posterior for z_n is determined by an allocation model. For all computations needed by our current observation model, we'll assume either a point estimate or an approximate posterior for z_n is known.

Normal Wishart prior

Each dimension *d* has a mean μ_{kd} and variance λ_{kd} which have a joint univariate Normal-Wishart prior with scalar hyperparameters $\bar{\nu}, \bar{\beta}_d$ for the Wishart prior and then $\bar{m}_d, \bar{\kappa}$ for the Normal prior:

$$\lambda_{kd} \sim \mathcal{W}_1(\bar{\nu}, \bar{\beta}_d)$$
$$\mu_{kd} \sim \mathcal{N}_1(\bar{m}_d, \bar{\kappa}^{-1}\lambda_{kd}^{-1})$$

These are represented by the following numpy array attributes of the `Prior` parameter bag:

- **nu** [float] degrees of freedom
- **beta** [1D array, size D] scale parameters that set mean of lambda
- **m** [1D array, size D] mean of the parameter mu
- **kappa** [float] scalar precision of mu

Several keyword arguments can be used to determine the values of the prior hyperparameters when calling bnpy.run

- **`--nu`** [float] Sets value of $\bar{\nu}$. Defaults to D + 2.
- **`--kappa`** [float] Sets value of $\bar{\kappa}$. Defaults to ???.
- **`--ECovMat`** [str] Determines the expected value of data covariance under the prior. Possible values include 'eye' and 'diagcovdata'. TODO
- **`--sF`** [float] These two options set the value of $\bar{\beta}$. TODO.
- TODO set m??

Approximate posterior

We assume the following factorized approximate posterior family for variational optimization:

$$q(z, \mu, \lambda) = \prod_{n=1}^{N} q(z_n) \cdot \prod_{k=1}^{K} (\mu_k, \lambda_k)$$

The specific forms of the global and local factors are given below.

Posterior for local assignments For each observation vector at index *n*, we assume an independent approximate posterior over the assigned cluster indicator $z_n \in \{1, 2, \dots K\}$.

$$\begin{aligned} q(z) &= \prod_{n=1}^{N} q(z_n | \hat{r}_n) \\ &= \prod_{n=1}^{N} \text{Discrete}(z_n | \hat{r}_{n1}, \hat{r}_{n2}, \dots \hat{r}_{nK}) \end{aligned}$$

Thus, for this observation model the only local variational parameter is the assignment responsibility array $\hat{r} = \{\{\hat{r}_{nk}\}_{k=1}^{K}\}_{n=1}^{N}$.

Inside the *LP* dict, this is represented by the *resp* numpy array:

- **`resp`** [2D array, size N x K] Parameters of approximate posterior q(z) over cluster assignments. resp[n,k] = probability observation n is assigned to component k.

Remember, all computations required by our observation model assume that the `resp` array is given. The actual values of `resp` are updated by an allocation model.

Posterior for global parameters The goal of variational optimization is to find the best approximate posterior distribution for the mean and precision parameters of each cluster *k*:

$$\begin{aligned} q(\mu, \lambda) &= \prod_{k=1}^{K} \prod_{d=1}^{D} q(\mu_{kd}, \lambda_{kd}) \\ &= \prod_{k=1}^{K} \prod_{d=1}^{D} \mathcal{W}_1(\lambda_{kd} | \hat{\nu}_k, \hat{\beta}_{kd}) \mathcal{N}_1(\mu_{kd} | \hat{m}_{kd}, \hat{\kappa}_k^{-1} \lambda_{kd}^{-1}) \end{aligned}$$

This approximate posterior is represented by the *Post* attribute of the *DiagGaussObsModel*. This is a ParamBag with the following attributes:

- **K** [int] number of active clusters
- **nu** [1D array, size K] Defines $\hat{\nu}_k$ for each cluster
- **beta** [2D array, size K x D] Defines $\hat{\beta}_{kd}$ for each cluster and dimension
- **m** [2D array, size K x D] Defines $\hat{m}_{kd}$ for each cluster and dimension
- **kappa** [2D array, size K] Defines $\hat{\kappa}_k$ for each cluster

Objective function Variational optimization will find the approximate posterior parameters that maximize the following objective function, given a fixed observed dataset $x = \{x_1, \dots x_N\}$ and fixed prior hyparparameters $\bar{\nu}, \bar{\beta}, \bar{m}, \bar{\kappa}$.

$$\begin{aligned}\mathcal{L}^{\text{DiagGauss}}(\hat{\nu}, \hat{\beta}, \hat{m}, \hat{\kappa}) &= \sum_{k=1}^{K}\sum_{d=1}^{D} c^{\text{NW}}_{1,1}(\hat{\nu}_k, \hat{\beta}_{kd}, \hat{m}_{kd}, \hat{\kappa})_k - c^{\text{NW}}_{1,1}(\bar{\nu}, \bar{\beta}_d, \bar{m}_d, \bar{\kappa}) \\ &+ \frac{1}{2}\sum_{k=1}^{K}\sum_{d=1}^{D} (N_k(\hat{r}) + \bar{\nu} - \hat{\nu}_k)\, \mathbb{E}_q[\log \lambda_{kd}] \\ &- \frac{1}{2}\sum_{k=1}^{K}\sum_{d=1}^{D} (N_k(\hat{r}) + \bar{\kappa} - \hat{\kappa}_k)\, \mathbb{E}_q[\lambda_{kd}] \\ &+ \sum_{k=1}^{K}\sum_{d=1}^{D} (S^{x}_{kd}(x, \hat{r}) + \bar{\kappa}\bar{m}_d - \hat{\kappa}_k\hat{m}_{kd})\, \mathbb{E}_q[\lambda_{kd}\mu_{kd}] \\ &- \frac{1}{2}\sum_{k=1}^{K}\sum_{d=1}^{D} \left(S^{x^2}_{kd}(x, \hat{r}) + \bar{\beta}_d + \bar{\kappa}\bar{m}_d^2 - \hat{\beta}_{kd} - \hat{\kappa}_k\hat{m}_{kd}^2\right) \mathbb{E}_q[\lambda_{kd}\mu_{kd}^2]\end{aligned}$$

This objective function is computed by calling the Python function `calc_evidence`.

Sufficient statistics The sufficient statistics of this observation model are functions of the local parameters $\hat{r}$ and the observed data x.

$$\begin{aligned}N_k(\hat{r}) &= \sum_{n=1}^{N} \hat{r}_{nk} \\ S^{x}_{kd}(x, \hat{r}) &= \sum_{n=1}^{N} \hat{r}_{nk} x_{nd}^2 \\ S^{x^2}_{kd}(x, \hat{r}) &= \sum_{n=1}^{N} \hat{r}_{nk} x_{nd}^2\end{aligned}$$

These fields are stored within the sufficient statistics parameter bag `SS` as the following fields:

- **SS.N** [1D array, size K] SS.N[k] = N_k
- **SS.x** [2D array, size K x D] SS.x[k,d] = $S^{x}_{kd}(x, \hat{r})$
- **SS.xx** [2D array, size K x D] SS.xx[k,d] = $S^{x^2}_{kd}(x, \hat{r})$

Cumulant function The cumulant function of the univariate Normal-Wishart is evaluated for each dimension d separately. The function takes 4 scalar input arguments and produces a scalar output.

$$c^{\mathrm{NW}}_{1,1}(\nu, \beta_d, m_d, \kappa) = -\frac{1}{2}\log 2\pi + \frac{1}{2}\log\kappa + \frac{\nu}{2}\log\frac{\beta_d}{2} - \log\Gamma\left(\frac{\nu}{2}\right)$$

Coordinate Ascent Updates

Local step update As with all observation models, the local step computes the *expected* log conditional probability of assigning each observation to each cluster:

$$\mathbb{E}[\log p(x_n|\mu_k, \lambda_k)] = -\frac{D}{2}\log 2\pi + \frac{1}{2}\sum_{d=1}^{D}\mathbb{E}[\log\lambda_{kd}] - \frac{1}{2}\sum_{d=1}^{D}\mathbb{E}[\lambda_{kd}(x_{nd} - \mu_{kd})^2]$$

where the elementary expectations required are:

$$\begin{aligned}\mathbb{E}[\log\lambda_{kd}] &= \psi\left(\frac{\hat\nu_k}{2}\right) - \log\frac{\hat\beta_{kd}}{2}\\ \mathbb{E}_q\left[\lambda_{kd}(x_{nd} - \mu_{kd})^2\right] &= \frac{1}{\hat\kappa_k} + \frac{\hat\nu_k}{\hat\beta_{kd}}(x_{nd} - \hat m_{kd})^2\end{aligned}$$

In our implementation, this is done via the function `calc_local_params`, which computes the following arrays and places them inside the local parameter dict `LP`.

- **`E_log_soft_ev`** [2D array, N x K] log probability of assigning each observation n to each cluster k

Global step update The global step update produces an updated approximate posterior over the global random variables. Concretely, this means updated values for each field of the `Post` ParamBag attribute of the DiagGaussObsModel.

$$\begin{aligned}\hat\nu_k &\leftarrow N_k(\hat r) + \bar\nu\\ \hat\kappa_k &\leftarrow N_k(\hat r) + \bar\kappa\\ \hat m_{kd} &\leftarrow \frac{1}{\hat\kappa_k}\left(S^x_k(x, \hat r) + \bar\kappa\bar m_d\right)\\ \hat\beta_{kd} &\leftarrow S^{x^2}_{kd}(x, \hat r) + \bar\beta_d + \bar\kappa\bar m_d^2 - \hat\kappa_k\hat m_{kd}^2\end{aligned}$$

Our implementation performs this update when calling the function `update_global_params`.

Initialization

Initialization creates valid values of the parameters which define the approximate posterior over the global random variables. Concretely, this means it creates a valid setting of the `Post` attribute of the DiagGaussObsModel object.

TODO

Gaussian Regression observation model: Variational Bayesian Methods

Generative model

The Gaussian regression observation model explains a observed collection of input/ouptut data pairs $\{x_n, y_n\}_{n=1}^N$. Each input observation x_n is a vector of length D, while each output observation y_n is a scalar.

In this document, we assume that the observed input data x are fixed and focus on a generative model for the output data y which depends on x. Various generative models, such as the diagonal-covariance Gaussian, are possible for the observed data x. These can be straight-forwardly combined with the model here to produce a joint model for both x and y.

Each cluster k produces the output values according to the following standard Bayesian linear regression model:

$$y_n \sim \mathcal{N}\left(b_k + \sum_{d=1}^{D} w_{kd} x_{nd}, \delta_k^{-1} \right)$$

Here, the cluster-specific parameters are a weight vector w_k, an intercept weight b_k, and a precision scalar δ_k. These are the global random variables of this observation model.

Alternatively, if we define an *expanded* input data vector $\tilde{x}_n = [x_{n1} x_{n2} \dots x_{nD} \ 1]$, we can write the generative model more simply as:

$$y_n \sim \mathcal{N}\left(\sum_{d=1}^{D+1} w_{kd} \tilde{x}_{nd}, \delta_k^{-1} \right)$$

Global Random Variables The global random variables are the cluster-specific weights and precisions. For each cluster k, we have

$$\begin{aligned} w_k &\in \mathbb{R}^D, \qquad w_k = [w_{k1}, w_{k2}, \dots w_{kD} w_{kD+1}] \\ \delta_k &\in (0, +\infty) \end{aligned}$$

For convenience, let E denote the size of this expanded representation, where $E = D + 1$.

Local Random Variables Each dataset observation at index n has its own cluster assignment:

$$z_n \in \{1, 2, \dots K\}$$

The generative model and approximate posterior for z_n is determined by an allocation model. For all computations needed by our current observation model, we'll assume either a point estimate or an approximate posterior for z_n is known.

Normal-Wishart prior

We assume that the weights w_k and the precision δ_k have a joint Normal-Wishart prior with hyperparameters:

- $\bar{\nu}$ [positive scalar] count parameter of the Wishart prior on precision

- $\bar{\tau}$ [positive scalar] location parameter of the Wishart prior on precision
- $\bar{w}$ [vector of size E] mean value of the Normal prior on cluster weights
- $\bar{P}$ [positive definite E x E matrix] precision matrix for the Normal prior on cluster weights

Mathematically, we have:

$$\delta_k \sim \mathcal{W}_1(\bar{\nu}, \bar{\tau})$$
$$w_k \sim \mathcal{N}_E(\bar{w}, \delta_k^{-1} \bar{P}^{-1})$$

Under this prior, here are some useful expectations for the precision random variable:

$$\mathbb{E}_{\text{prior}}[\delta_k] = \frac{\bar{\nu}}{\bar{\tau}}$$
$$\mathbb{E}_{\text{prior}}[\delta_k^{-1}] = \frac{\bar{\tau}}{\bar{\nu} - 2}$$
$$\text{Var}_{\text{prior}}[\delta_k] = \frac{\bar{\nu}}{\bar{\tau}^2}$$

Likewise, here are some useful prior expectations for the weight vector random variable:

$$\mathbb{E}_{\text{prior}}[w_k] = \bar{w}$$
$$\text{Cov}_{\text{prior}}[w_k] = \frac{\bar{\tau}}{\bar{\nu} - 2} \bar{P}^{-1}$$

And some useful joint expectations:

$$\mathbb{E}_{\text{prior}}[\delta_k w_k] = \frac{\bar{\nu}}{\bar{\tau}} \bar{w}$$
$$\mathbb{E}_{\text{prior}}[\delta_k w_k w_k^T] = \bar{P}^{-1} + \frac{\bar{\nu}}{\bar{\tau}} \bar{w} \bar{w}^T$$

In our Python implementation of the `GaussRegressYFromFixedXObsModel` class, these quantities are represented by the following numpy array attributes of the `Prior` parameter bag:

- **pnu** [float] value of $\bar{\nu}$
- **ptau** [float] value of $\bar{\tau}$
- **w_E** [1D array, size E] value of $\bar{w}$
- **P_EE** [2D array, size E x E] value of $\bar{P}$

Several keyword arguments can be used to determine the values of the prior hyperparameters when calling bnpy.run

- **--pnu** [float] Sets value of $\bar{\nu}$. Defaults to 1.
- **--ptau** [float] Sets value of $\bar{\tau}$. Defaults to 1.
- **--w_E** [float or 1D array] Sets value of the vector $\bar{w}$. If float is provided, the whole vector is filled with that value. Defaults to 0.
- **--P_diag_val** [float or 1D array] Sets $\bar{P}$ to diagonal matrix with specified values. Defaults to 1e-6.

Approximate posterior

We assume the following factorized approximate posterior family for variational optimization:

$$q(z, w, \delta) = \prod_{n=1}^{N} q(z_n) \cdot \prod_{k=1}^{K} (w_k, \delta_k)$$

The specific forms of the global and local factors are given below.

Posterior for local assignments For each observation vector at index *n*, we assume an independent approximate posterior over the assigned cluster indicator $z_n \in \{1, 2, \dots K\}$.

$$\begin{aligned} q(z) &= \prod_{n=1}^{N} q(z_n | \hat{r}_n) \\ &= \prod_{n=1}^{N} \text{Discrete}(z_n | \hat{r}_{n1}, \hat{r}_{n2}, \dots \hat{r}_{nK}) \end{aligned}$$

Thus, for this observation model the only local variational parameter is the assignment responsibility array $\hat{r} = \{\{\hat{r}_{nk}\}_{k=1}^{K}\}_{n=1}^{N}$.

Inside the *LP* dict, this is represented by the *resp* numpy array:

- **resp** [2D array, size N x K] Parameters of approximate posterior q(z) over cluster assignments. resp[n,k] = probability observation n is assigned to component k.

Remember, all computations required by our observation model assume that the `resp` array is given. The actual values of `resp` are updated by an allocation model.

Posterior for global parameters The goal of variational optimization is to find the best approximate posterior distribution for the mean and precision parameters of each cluster *k*:

$$q(w, \delta) = \prod_{k=1}^{K} \mathcal{W}_1(\delta_k | \hat{\nu}_k, \hat{\tau}_k) \mathcal{N}_E(w_k | \hat{w}_k, \delta_k^{-1} \hat{P}_k^{-1})$$

Within our Python implementation in the class `GaussRegressYFromFixedXObsModel`, this approximate posterior is represented within the *Post* attribute. This attribute is a ParamBag object containing the following numpy arrays:

- **K** [int] number of active clusters
- **pnu_K** [1D array, size K] Defines $\hat{\nu}_k$ for each cluster
- **ptau_K** [1D array, size K] Defines $\hat{\tau}_k$ for each cluster
- **w_KE** [2D array, size K x E] Defines $\hat{w}_{ke}$ for each cluster and expanded dimension
- **P_KEE** [2D array, size K x E x E] Defines precision matrix $\hat{P}_k$ for each cluster

Objective function Variational optimization will find the approximate posterior parameters that maximize the following objective function, given a fixed observed dataset $x = \{x_1, \dots x_N\}$ and fixed prior hyparparameters $\bar{\nu}, \bar{\tau}, \bar{w}, \bar{P}$.

$$\begin{aligned}
\mathcal{L}^{\text{Gaussian Regression}}(y, x, \hat{\nu}, \hat{\tau}, \hat{w}, \hat{P}) = &-\frac{N}{2}\log 2\pi \\
&+ \sum_{k=1}^{K}\sum_{d=1}^{D} c_{1,E}^{\text{NW}}(\hat{\nu}_k, \hat{\tau}_k, \hat{w}_k, \hat{P}_k) - c_{1,E}^{\text{NW}}(\bar{\nu}, \bar{\tau}, \bar{w}, \bar{P}) \\
&- \frac{1}{2}\sum_{k=1}^{K} (N_k(\hat{r}) + \bar{\nu} - \hat{\nu}_k)\, \mathbb{E}_q[\log \delta_k] \\
&- \frac{1}{2}\sum_{k=1}^{K} \left(S_k^{yy}(y, \hat{r}) + \bar{\tau} + \bar{w}\bar{P}\bar{w} - \hat{\tau}_k - \hat{w}_k\hat{P}_k\hat{w}_k\right) \mathbb{E}_q[\delta_k] \\
&+ \sum_{k=1}^{K} \left(S_k^{yx}(x, y, \hat{r}) + \bar{P}\bar{w} - \hat{P}_k\hat{w}_k\right)^T \mathbb{E}_q[\delta_k w_k] \\
&- \frac{1}{2}\sum_{k=1}^{K} \text{trace}\left(\left(S_k^{xx^T}(x, \hat{r}) + \bar{P} - \hat{P}_k\right) \mathbb{E}_q[\delta_k w_k w_k^T]\right)
\end{aligned}$$

This objective function is computed by calling the Python function `calc_evidence`.

We can directly interpret this function as a lower bound on the marginal evidence:

$$\log p(y|x, \bar{\nu}, \bar{\tau}, \bar{w}, \bar{P}) \geq \mathcal{L}^{\text{Gaussian Regression}}(y, x, \hat{\nu}, \hat{\tau}, \hat{w}, \hat{P})$$

Sufficient statistics The sufficient statistics of this observation model are functions of the local parameters $\hat{r}$, the observed input data x, and the observed output data y.

$$\begin{aligned}
N_k(\hat{r}) &= \sum_{n=1}^{N} \hat{r}_{nk} \\
S_k^{y^2}(y, \hat{r}) &= \sum_{n=1}^{N} \hat{r}_{nk} y_n^2 \\
S_k^{yx}(x, y, \hat{r}) &= \sum_{n=1}^{N} \hat{r}_{nk} y_n x_n \\
S_k^{xx^T}(x, \hat{r}) &= \sum_{n=1}^{N} \hat{r}_{nk} x_n x_n^T
\end{aligned}$$

These fields are stored within the sufficient statistics parameter bag `SS` as the following fields:

- **`SS.N`** [1D array, size K] SS.N[k] = N_k
- **`SS.yy_K`** [1D array, size K] SS.yy[k] = $S_k^{y^2}(y, \hat{r})$
- **`SS.yx`** [2D array, size K x E] SS.yx[k] = $S_k^{yx}(x, y, \hat{r})$
- **`SS.xxT`** [3D array, size K x E x E] SS.xxT[k] = $S_k^{xx^T}(x, \hat{r})$

Cumulant function The cumulant function of the Normal-Wishart produces a scalar output from 4 input arguments:

$$c^{\text{NW}}_{1,E}(\nu, \tau, w, P) = \frac{E}{2} \log 2\pi - \frac{1}{2} \log |P| - \frac{\nu}{2} \log \frac{\tau}{2} + \log \Gamma \left(\frac{\nu}{2} \right)$$

where $\Gamma(\cdot)$ is the gamma function, and $\log |P|$ is the log determinant of the E x E matrix P.

Coordinate Ascent Updates

Local step update As with all observation models, the local step computes the *expected* log conditional probability of assigning each observation to each cluster:

$$\mathbb{E}_q[\log p(y_n|x_n, w_k, \delta_k)] = -\frac{1}{2} \log 2\pi + \frac{1}{2}\mathbb{E}[\log \delta_k] - \frac{1}{2}\mathbb{E}[\delta_k(y_n - w_k^T \tilde{x}_n)^2]$$

where the elementary expectations required are:

$$\mathbb{E}_q[\log \delta_k] = -\log \frac{\hat{\tau}_k}{2} + \psi \left(\frac{\hat{\nu}_k}{2} \right)$$
$$\mathbb{E}_q \left[\delta_k \left(y_n - w_k^T \tilde{x}_n \right)^2 \right] = \tilde{x}_n^T \hat{P}_k^{-1} \tilde{x}_n + \frac{\hat{\nu}_k}{\hat{\tau}_k} (y_n - \bar{w}_k^T \tilde{x}_n)^2$$

The above operations can be efficiently computed via smart vectorized calculations on modern cpus.

In our implementation, this is done via the function `calc_local_params`, which computes the following arrays and places them inside the local parameter dict `LP`.

- **`E_log_soft_ev`** [2D array, N x K] log probability of assigning each observation n to each cluster k

Global step update The global step update produces an updated approximate posterior over the global random variables. Concretely, this means updated values for each of the four parameters which define each cluster-specific Normal-Wishart:

$$\begin{aligned}
\hat{\nu}_k &\leftarrow N_k(\hat{r}) + \bar{\nu} \\
\hat{P}_k &\leftarrow \bar{P}_k + S_k^{xx^T}(x, \hat{r}) \\
\hat{w}_k &\leftarrow \hat{P}_k^{-1} \left(\bar{P}\bar{w} + S_k^{yx}(x, y, \hat{r}) \right) \\
\hat{\tau}_k &\leftarrow \bar{\tau} + S_k^{y^2}(y, \hat{r}) + \bar{w}^T \bar{P} \bar{w} - \hat{w}_k^T \hat{P}_k \hat{w}_k
\end{aligned}$$

Our implementation performs this update when calling the function `update_global_params`.

Initialization

Initialization creates valid values of the parameters which define the approximate posterior over the global random variables. Concretely, this means it creates a valid setting of the `Post` attribute of the `GaussRegressYFromFixedXObsModel` object.

TODO

Index