
dpcluster Documentation

Release 0.1

Teodor Mihai Moldovan

Apr 12, 2017

Contents

1	Usage	3
2	ToDo	5
3	Documentation	7
3.1	<i>dpcluster</i> Package	7
3.1.1	algorithms Module	7
3.1.2	distributions Module	9
3.1.3	test Module	11
4	Indices and tables	13
	Python Module Index	15

dpcluster is a package for grouping together (clustering) vectors. It automatically chooses the number of clusters that fits the data best. Specifically, it models the data as a Dirichlet Process mixture in the exponential family. For a tutorial see “[Dirichlet Process](#)” by [Y.W. Teh \(2010\)](#). Currently the only distribution implemented is the multivariate Gaussian with a Normal-Inverse-Wishart conjugate prior but extensions to other distributions are possible.

Two inference algorithms are implemented:

- Variational inference as described in “[Variational Inference for Dirichlet Process Mixtures](#)” by Blei et al. (2006). This is a batch algorithm that requires storing all data in memory.
- An experimental on-line inference algorithm that requires only $O(\log(n))$ memory where n is the total number of observations.

To install locally run:

```
python setup.py install --user
```


CHAPTER 1

Usage

Here is a simple example to demonstrate clustering a number of random points in the plane:

```
>>> from dpcluster import *
>>> n = 10
>>> data = np.random.normal(size=2*n).reshape(-1, 2)
>>> vdp = VDP(GaussianNIW(2))
>>> vdp.batch_learn(vdp.distr.sufficient_stats(data))
>>> plt.scatter(data[:,0], data[:,1])
>>> vdp.plot_clusters(slc=np.array([0,1]))
>>> plt.show()
```

Running this might produce 2-3 clusters depending on the randomly generated data. The adaptive nature of the Dirichlet Process mixture model becomes apparent when we increase the number of data points from $n = 10$ to $n = 500$. In this case the clustering algorithm will likely explain the data using only one cluster.

CHAPTER 2

ToDo

- Implement more clustering algorithms e.g. based on Gibbs sampling, expectation propagation, stochastic gradient descent.
- Implement more clustering distributions.
- Re-implement algorithms to take advantage of multi-core or GPU computing.

CHAPTER 3

Documentation

dpcluster Package

algorithms Module

```
class dpcluster.algorithms.OnlineVDP(distr, w=0.1, k=25, tol=0.001, max_items=100)
    Experimental online clustering algorithm.
```

Parameters

- **distr** – likelihood-prior distribution pair governing clusters. For now the only option is using a instance of *dpcluster.distributions.GaussianNIW*.
- **w** – non-negative prior weight. The prior has as much influence as w data points.
- **k** – maximum number of clusters.
- **tol** – convergence tolerance.
- **max_items** – maximum queue length.

get_model()

Get current model.

Returns instance of *dpcluster.algorithms.VDP*

put(r, s=0)

Append data.

Parameters **r** – sufficient statistics of data to be appended.

Basic usage example:

```
>>> distr = GaussianNIW(data.shape[2])
>>> x = distr.sufficient_stats(data)
>>> vdp = OnlineVDP(distr)
>>> vdp.put(x)
>>> print vdp.get_model().cluster_parameters()
```

```
class dpcluster.algorithms.Predictor(model, ix, iy)

    distr_fit (*args)
    precomp (*args)
    predict (*args)
    predict_old(z, lgh=(True, True, False), full_var=False)

class dpcluster.algorithms.PredictorKL(model, ix, iy)

    predict (*args)
    predict_old(z, lgh=(True, True, False), full_var=False)

class dpcluster.algorithms.VDP(distr, w=0.1, k=50, tol=1e-05, max_iters=10000)
    Bases: object
```

Variational Dirichlet Process clustering algorithm following “Variational Inference for Dirichlet Process Mixtures” by Blei et al. (2006).

Parameters

- **distr** – likelihood-prior distribution pair governing clusters. For now the only option is using a instance of `dpcluster.distributions.GaussianNIW`.
- **w** – non-negative prior weight. The prior has as much influence as w data points.
- **k** – maximum number of clusters.
- **tol** – convergence tolerance.

`batch_learn(x, verbose=False, sort=True)`

Learn cluster from data. This is a batch algorithm that required all data be loaded in memory.

Parameters

- **x** – sufficient statistics of the data to be clustered. Can be obtained from raw data by calling `dpcluster.distributions.ConjugatePair.sufficient_stats()`
- **verbose** – print progress report
- **sort** – algorithm optimization. Sort clusters at every step.

Basic usage example:

```
>>> distr = GaussianNIW(data.shape[2])
>>> x = distr.sufficient_stats(data)
>>> vdp = VDP(distr)
>>> vdp.batch_learn(x)
>>> print vdp.cluster_parameters()
```

`cluster_parameters()`

Returns Cluster parameters.

`cluster_sizes()`

Returns Data weight assigned to each cluster.

`conditional_expectation(*args)`

`conditional_ll(x, cond)`

Conditional log likelihood.

Parameters

- **x** – sufficient statistics of data.
 - **cond** – slice representing variables to condition on
- conditional_variance** (*x, iy, ix, ret_ll_gr_hs=(True, False, False)*)
ll (*x, ret_ll_gr_hs=(True, False, False)*)
Compute the log likelihoods (ll) of data with respect to the trained model.

Parameters

- **x** – sufficient statistics of the data.
- **ret_ll_gr_hs** – what to return: likelihood, gradient, hessian. Derivatives taken with respect to data, not sufficient statistics.

marginal (*args)
plot_clusters (**kwargs)
Asks each cluster to plot itself. For Gaussian multidimensional clusters pass `slc=np.array([i, j])` as an argument to project clusters on the plane defined by the i'th and j'th coordinate.
pseudo_resp (*args)
pseudo_resp_cache (*args)
resp (*args)
resp_cache (*args)
var_cond_exp (*x, iy, ix, ret_ll_gr_hs=(True, False, False), full_var=False*)

distributions Module

class dpcluster.distributions.**ConjugatePair** (*evidence_distr, prior_distr, prior_param*)
Conjugate prior-evidence pair of distributions in the exponential family. Conjugacy means that the posterior has the same form as the prior with updated parameters.

Parameters

- **evidence_distr** – Evidence distribution. Must be an instance of `ExponentialFamilyDistribution`
- **prior_distr** – Prior distribution. Must be an instance of `ExponentialFamilyDistribution`
- **prior_param** – Prior parameters.

posterior_ll (*x, nu, ret_ll_gr_hs=(True, False, False), usual_x=False*)
Log likelihood (and derivatives) of data under posterior predictive distribution.

Parameters

- **x** – sufficient statistics of data
- **nu** – prior parameters

sufficient_stats (*data*)
sufficient_stats_dim ()

class `dpcluster.distributions.ExponentialFamilyDistribution`

Models a distribution in the exponential family of the form:

$$f(x|\nu) = h(x) \exp(\nu \cdot T(x) - A(\nu))$$

Parameters to be defined in subclasses:

- `h` is the base measure
- `nu` (ν) are the parameters
- `T(x)` are the sufficient statistics of the data
- `A` is the log partition function

ll (`xs, nus, ret_ll_gr_hs=(True, False, False)`)

Log likelihood (and derivatives, optionally) of data under distribution.

Parameters

- `xs` – sufficient statistics of data
- `nus` – parameters of distribution

log_base_measure (`x, ret_ll_gr_hs=(True, False, False)`)

Log of the base measure. To be implemented by subclasses.

Parameters `x` – sufficient statistics of the data.

log_partition (`nu, ret_ll_gr_hs=(True, False, False)`)

Log of the partition function and derivatives with respect to sufficient statistics. To be implemented by subclasses.

Parameters

- `nu` – parameters of the distribution
- `ret_ll_gr_hs` – what to return: log likelihood, gradient, hessian

class `dpcluster.distributions.Gaussian(d)`

Bases: `dpcluster.distributions.ExponentialFamilyDistribution`

Multivariate Gaussian distribution with density:

$$f(x|\mu, \Sigma) = |2\pi\Sigma|^{-1/2} \exp(-((x - \mu)^T \Sigma^{-1} (x - \mu))/2)$$

Natural parameters:

$$\nu = [\Sigma^{-1}\mu, -\Sigma^{-1}/2]$$

Sufficient statistics of data:

$$T(x) = [x, x \cdot x^T]$$

Parameters `d` – dimension.

log_base_measure (`x, ret_ll_gr_hs=(True, True, True)`)

Log base measure.

log_partition (`nus`)

nat2usual (`nus`)

Convert natural parameters to usual parameters

sufficient_stats (`x`)

Sufficient statistics of data. :arg x: data

```
sufficient_stats_dim()
Dimension of sufficient statistics.

usual2nat (mus, Sgs)
Convert usual parameters to natural parameters.

class dpcluster.distributions.GaussianNIW(d)
Bases: dpcluster.distributions.ConjugatePair
Gaussian, Normal-Inverse-Wishart conjugate pair.

The predictive posterior is a multivariate t-distribution.

    Parameters d – dimension

conditional (*args)
conditional_expectation (*args)
conditional_variance (x, nu, iy, ix, ret_ll_gr_hs=(True, True, False), full_var=True)
conditionals_cache (*args)
conditionals_cache_bare (*args)
marginal (nu, slc)
plot (nu, szs, slc, n=100)
posterior_ll (*args)
posterior_ll_cache (*args)
sufficient_stats (data)

class dpcluster.distributions.NIW(d)
Bases: dpcluster.distributions.ExponentialFamilyDistribution
Normal Inverse Wishart distribution defined by:

 $f(\mu, \Sigma | \mu_0, \Psi, k) = \text{Gaussian}(\mu | \mu_0, \Sigma/k) \cdot \text{Inverse-Wishart}(\Sigma | \Psi, \nu - d - 2)$ 
where  $\mu, \mu_0 \in R^d$ ,  $\Sigma, \Psi \in R^{d \times d}$ ,  $k \in R$ ,  $\nu > 2d + 1 \in R$ 

This is an exponential family conjugate prior for the Gaussian.

    Parameters d – dimension

log_base_measure (x, ret_ll_gr_hs=(True, True, True))
log_partition (nu, ret_ll_gr_hs=(True, False, False), no_k_grad=False)
multipsi (a, d)
nat2usual (*args)
sufficient_stats (mus, Sgs)
sufficient_stats_dim ()
usual2nat (mu0, Psi, k, nu)
```

test Module

```
class dpcluster.test.Tests (methodName='runTest')
Bases: unittest.case.TestCase

gen_data (A, mu, n=10)
```

```
setUp()
test_batch_vdp()
test_gaussian()
test_gniw()
test_gniw_conditionals()
test_ll()
test_niw()
test_online_vdp(*args, **kwargs)
test_predictor(*args, **kwargs)
test_presp(*args, **kwargs)
test_resp()
test_vdp_conditionals()

dpcluster.test.grad_check(f, x, eps=0.0001)
```

CHAPTER 4

Indices and tables

- genindex
- modindex
- search

Python Module Index

d

dpcluster.__init__, [7](#)
dpcluster.algorithms, [7](#)
dpcluster.distributions, [9](#)
dpcluster.test, [11](#)

Index

B

batch_learn() (dpcluster.algorithms.VDP method), 8

C

cluster_parameters() (dpcluster.algorithms.VDP method), 8

cluster_sizes() (dpcluster.algorithms.VDP method), 8

conditional() (dpcluster.distributions.GaussianNIW method), 11

conditional_expectation() (dpcluster.algorithms.VDP method), 8

conditional_expectation() (dpcluster.distributions.GaussianNIW method), 11

conditional_ll() (dpcluster.algorithms.VDP method), 8

conditional_variance() (dpcluster.algorithms.VDP method), 9

conditional_variance() (dpcluster.distributions.GaussianNIW method), 11

conditionals_cache() (dpcluster.distributions.GaussianNIW method), 11

conditionals_cache_bare() (dpcluster.distributions.GaussianNIW method), 11

ConjugatePair (class in dpcluster.distributions), 9

D

distr_fit() (dpcluster.algorithms.Predictor method), 8

dpcluster.__init__ (module), 7

dpcluster.algorithms (module), 7

dpcluster.distributions (module), 9

dpcluster.test (module), 11

E

ExponentialFamilyDistribution (class in dpcluster.distributions), 9

G

Gaussian (class in dpcluster.distributions), 10

GaussianNIW (class in dpcluster.distributions), 11

gen_data() (dpcluster.test.Tests method), 11

get_model() (dpcluster.algorithms.OnlineVDP method), 7

grad_check() (in module dpcluster.test), 12

L

ll() (dpcluster.algorithms.VDP method), 9

ll() (dpcluster.distributions.ExponentialFamilyDistribution method), 10

log_base_measure() (dpcluster.distributions.ExponentialFamilyDistribution method), 10

log_base_measure() (dpcluster.distributions.Gaussian method), 10

log_base_measure() (dpcluster.distributions.NIW method), 11

log_partition() (dpcluster.distributions.ExponentialFamilyDistribution method), 10

log_partition() (dpcluster.distributions.Gaussian method), 10

log_partition() (dpcluster.distributions.NIW method), 11

M

marginal() (dpcluster.algorithms.VDP method), 9

marginal() (dpcluster.distributions.GaussianNIW method), 11

multipsi() (dpcluster.distributions.NIW method), 11

N

nat2usual() (dpcluster.distributions.Gaussian method), 10

nat2usual() (dpcluster.distributions.NIW method), 11

NIW (class in dpcluster.distributions), 11

O

OnlineVDP (class in dpcluster.algorithms), 7

P

plot() (dpcluster.distributions.GaussianNIW method), 11
plot_clusters() (dpcluster.algorithms.VDP method), 9
posterior_ll() (dpcluster.distributions.ConjugatePair method), 9
posterior_ll() (dpcluster.distributions.GaussianNIW method), 11
posterior_ll_cache() (dpcluster.distributions.GaussianNIW method), 11
precomp() (dpcluster.algorithms.Predictor method), 8
predict() (dpcluster.algorithms.Predictor method), 8
predict() (dpcluster.algorithms.PredictorKL method), 8
predict_old() (dpcluster.algorithms.Predictor method), 8
predict_old() (dpcluster.algorithms.PredictorKL method), 8
Predictor (class in dpcluster.algorithms), 7
PredictorKL (class in dpcluster.algorithms), 8
pseudo_resp() (dpcluster.algorithms.VDP method), 9
pseudo_resp_cache() (dpcluster.algorithms.VDP method), 9
put() (dpcluster.algorithms.OnlineVDP method), 7

R

resp() (dpcluster.algorithms.VDP method), 9
resp_cache() (dpcluster.algorithms.VDP method), 9

S

setUp() (dpcluster.test.Tests method), 12
sufficient_stats() (dpcluster.distributions.ConjugatePair method), 9
sufficient_stats() (dpcluster.distributions.Gaussian method), 10
sufficient_stats() (dpcluster.distributions.GaussianNIW method), 11
sufficient_stats() (dpcluster.distributions.NIW method), 11
sufficient_stats_dim() (dpcluster.distributions.ConjugatePair method), 9
sufficient_stats_dim() (dpcluster.distributions.Gaussian method), 10
sufficient_stats_dim() (dpcluster.distributions.NIW method), 11

T

test_batch_vdp() (dpcluster.test.Tests method), 12
test_gaussian() (dpcluster.test.Tests method), 12
test_gniw() (dpcluster.test.Tests method), 12
test_gniw_conditionals() (dpcluster.test.Tests method), 12
test_ll() (dpcluster.test.Tests method), 12
test_niw() (dpcluster.test.Tests method), 12
test_online_vdp() (dpcluster.test.Tests method), 12

test_predictor() (dpcluster.test.Tests method), 12
test_presp() (dpcluster.test.Tests method), 12
test_resp() (dpcluster.test.Tests method), 12
test_vdp_conditionals() (dpcluster.test.Tests method), 12
Tests (class in dpcluster.test), 11

U

usual2nat() (dpcluster.distributions.Gaussian method), 11
usual2nat() (dpcluster.distributions.NIW method), 11

V

var_cond_exp() (dpcluster.algorithms.VDP method), 9
VDP (class in dpcluster.algorithms), 8