STA663
Infinite Latent Feature Models and the Indian Buffet Process

Radhika Anand

April 29, 2015

1 Outline

1.1 Definition of the Indian Buffet Process

In the Indian buffet process, N customers enter a restaurant one after another. Each customer encounters a buffet consisting of infinitely many dishes arranged in a line. The first customer starts at the left of the buffet and takes a serving from each dish, stopping after a Poisson(α) number of dishes. The ith customer moves along the buffet, sampling dishes in proportion to their popularity, taking dish k with probability mk/i , where mk is the number of previous customers who have sampled that dish. Having reached the end of all previous sampled dishes, the ith customer then tries a Poisson(α/i) number of new dishes.

1.2 Steps

1) Start by defining a probability distribution over equivalence classes of binary matrices with a finite number of rows and an unbounded number of columns. This distribution is suitable for use as a prior in probabilistic models that represent objects using a potentially infinite array of features

2) Next we see that the Indian Buffet Process is a simple generative process that results in the same distribution (as described in previous step) over equivalence classes because IBP has finite number of objects (people) and infinite number of features (dishes)

3) We then define a Gibbs Sampler for models using this concept of IBP

4) Further, to illustrate how IBP can be used as a prior in models for unsupervised learning, we derive and test a linear-Gaussian latent feature model in which the features are binary

5) We start with a finite realization and then take the infinite limit

2 Algorithm

1) We use Gamma prior for α

$$\alpha \sim Gamma(1, 1)$$

2) Prior on Z is obtained by IBP (after taking the infinite limit) as:

$$P(z_{ik} = 1|\mathbf{z}_{-i,k}) = \frac{m_{-i,k}}{N}$$

where $z_{-i,k}$ is the set of assignments of other objects, not including i, for feature k, and $m_{-i,k}$ is the number of objects possessing feature k, not including i.

3) Likelihood is given by

$$P(X|Z,\sigma_X,A) = \frac{1}{(2\pi\sigma_x^2)^{ND/2}} exp(-\frac{1}{2\sigma_X^2} tr((X-ZA)^T(X-ZA))) \tag{1}$$

where Z is the binary feature matrix and A is the weight matrix. σ_X and σ_A are the respective noise terms. Next, we integrate out A to get $P(X|Z,\sigma_X,\sigma_A)$.

4) Using this likelihood and the prior given by IBP, full conditional posterior for Z can be calculated as:

$$P(z_{ik}|X,Z_{-(i,k)},\sigma_X,\sigma_A) \propto P(X|Z,\sigma_X,\sigma_A) * P(z_{ik}=1|\mathbf{z}_{-i,k})$$

5) Now, to sample the number of new features for observation i, we use a truncated distribution, computing probabilities for a range of values $K_1^{(i)}$ up to an upper bound. The prior on number of features is given by $Poisson(\frac{\alpha}{N})$. Using this prior and the likelihood, we sample the number of new features.

6) Conditional posterior for α is given by:

$$P(\alpha|Z) \sim Gamma(1+K_+, 1+\sum_{i=1}^{N} H_i)$$

where H_N is the Nth harmonic number given by $H_N = \sum_{j=1}^{N} 1/j$

7) We now run the Gibbs Sampler using these full conditionals

8) To update σ_X and σ_A, we use MH algorithm as follows:

$$\epsilon \sim Uniform(-.05, .05) \tag{2}$$
$$\sigma_X^* = \sigma_X + \epsilon \tag{3}$$
$$\sigma_A^* = \sigma_X + \epsilon \tag{4}$$
$$\tag{5}$$

data.txt

```
2184313 function calls in 9.094 seconds

   Ordered by: internal time

   ncalls  tottime  percall  cumtime  percall filename:linenofunction
   286528    3.351    0.000    3.351    0.000 {numpy.core._dotblas.dot}
    35816    1.212    0.000    6.131    0.000 <ipython-input-88-d1ee8a415007>:2likelihood
        1    0.988    0.988    9.094    9.094 <ipython-input-19-5b64a3c9505b>:3sampler
    35816    0.495    0.000    1.086    0.000 linalg.py:455inv
   107448    0.412    0.000    0.758    0.000 twodim_base.py:190eye
   112552    0.355    0.000    0.355    0.000 {numpy.core.multiarray.zeros}
    35816    0.333    0.000    0.778    0.000 linalg.py:1679det
    35816    0.314    0.000    0.314    0.000 {method 'trace' of 'numpy.ndarray' objects}
    21332    0.255    0.000    0.255    0.000 {sum}
```

```
 71632    0.189    0.000    0.329    0.000 linalg.py:139_commonType
112782    0.120    0.000    0.120    0.000 {numpy.core.multiarray.array}
 71632    0.092    0.000    0.149    0.000 linalg.py:209_assertNdSquareness
107448    0.090    0.000    0.191    0.000 numeric.py:394asarray
 82361    0.090    0.000    0.090    0.000 {max}
 35816    0.081    0.000    0.081    0.000 {method 'astype' of 'numpy.generic' objects}
 10980    0.077    0.000    0.077    0.000 {method 'uniform' of 'mtrand.RandomState' objects}
 71632    0.075    0.000    0.090    0.000 linalg.py:198_assertRankAtLeast2
214896    0.070    0.000    0.070    0.000 {issubclass}
 35816    0.068    0.000    0.068    0.000 {method 'astype' of 'numpy.ndarray' objects}
 35816    0.060    0.000    0.060    0.000 linalg.py:101get_linalg_error_extobj
143264    0.057    0.000    0.090    0.000 linalg.py:111isComplexType
 35816    0.057    0.000    0.451    0.000 fromnumeric.py:1225trace
 35816    0.053    0.000    0.120    0.000 linalg.py:106_makearray
 71632    0.038    0.000    0.054    0.000 linalg.py:124_realType
 35816    0.035    0.000    0.035    0.000 linalg.py:219_assertNoEmpty2d
 71732    0.017    0.000    0.017    0.000 {min}
 71681    0.016    0.000    0.016    0.000 {method 'get' of 'dict' objects}
 72024    0.015    0.000    0.015    0.000 {len}
 25000    0.013    0.000    0.013    0.000 {math.factorial}
 15201    0.013    0.000    0.013    0.000 {range}
 35865    0.012    0.000    0.012    0.000 {getattr}
 35816    0.007    0.000    0.007    0.000 {method '__array_prepare__' of 'numpy.ndarray' objects}
     1    0.006    0.006    0.016    0.016 <ipython-input-15-51bada944323>:2sampleIBP
   343    0.002    0.000    0.002    0.000 {method 'send' of 'zmq.backend.cython.socket.Socket' objects}
   196    0.002    0.000    0.005    0.000 encoder.py:212iterencode
   300    0.002    0.000    0.028    0.000 iostream.py:207write
  1274    0.001    0.000    0.002    0.000 {method 'sub' of '_sre.SRE_Pattern' objects}
    49    0.001    0.000    0.002    0.000 uuid.py:101__init__
  1274    0.001    0.000    0.003    0.000 encoder.py:33encode_basestring
    49    0.001    0.000    0.001    0.000 {zmq.backend.cython._poll.zmq_poll}
    49    0.001    0.000    0.020    0.000 session.py:589send
   196    0.001    0.000    0.007    0.000 __init__.py:193dumps
    49    0.001    0.000    0.001    0.000 {posix.urandom}
    49    0.001    0.000    0.024    0.000 iostream.py:151flush
    49    0.001    0.000    0.011    0.000 session.py:530serialize
   196    0.001    0.000    0.006    0.000 encoder.py:186encode
   196    0.001    0.000    0.008    0.000 jsonapi.py:31dumps
    49    0.001    0.000    0.003    0.000 uuid.py:579uuid4
   300    0.001    0.000    0.001    0.000 {_codecs.utf_8_decode}
  1231    0.001    0.000    0.001    0.000 {isinstance}
   637    0.000    0.000    0.000    0.000 traitlets.py:395__get__
    49    0.000    0.000    0.002    0.000 session.py:515sign
   300    0.000    0.000    0.001    0.000 {method 'decode' of 'str' objects}
   398    0.000    0.000    0.000    0.000 iostream.py:93_is_master_process
    49    0.000    0.000    0.001    0.000 attrsettr.py:35__getattr__
    49    0.000    0.000    0.000    0.000 session.py:200extract_header
    49    0.000    0.000    0.002    0.000 socket.py:250send_multipart
    49    0.000    0.000    0.000    0.000 {built-in method now}
    49    0.000    0.000    0.005    0.000 session.py:496msg
   196    0.000    0.000    0.000    0.000 encoder.py:101__init__
   196    0.000    0.000    0.008    0.000 session.py:84<lambda>
   245    0.000    0.000    0.000    0.000 {method 'update' of '_hashlib.HASH' objects}
   196    0.000    0.000    0.000    0.000 {method 'join' of 'str' objects}
    49    0.000    0.000    0.000    0.000 {map}
   349    0.000    0.000    0.001    0.000 iostream.py:102_check_mp_mode
    49    0.000    0.000    0.001    0.000 {hasattr}
    49    0.000    0.000    0.002    0.000 iostream.py:123_flush_from_subprocesses
    49    0.000    0.000    0.000    0.000 hmac.py:88copy
    49    0.000    0.000    0.005    0.000 session.py:493msg_header
   147    0.000    0.000    0.000    0.000 {method 'encode' of 'unicode' objects}
```

```
300    0.000    0.000    0.001    0.000 utf_8.py:15decode
300    0.000    0.000    0.000    0.000 {method 'write' of '_io.StringIO' objects}
 49    0.000    0.000    0.003    0.000 session.py:441msg_id
 49    0.000    0.000    0.000    0.000 uuid.py:197__str__
 49    0.000    0.000    0.000    0.000 iostream.py:96_is_master_thread
100    0.000    0.000    0.000    0.000 {numpy.core.multiarray.copyto}
 49    0.000    0.000    0.000    0.000 iostream.py:238_flush_buffer
 49    0.000    0.000    0.001    0.000 poll.py:77poll
 49    0.000    0.000    0.001    0.000 session.py:195msg_header
100    0.000    0.000    0.000    0.000 {numpy.core.multiarray.empty}
 49    0.000    0.000    0.000    0.000 iostream.py:247_new_buffer
100    0.000    0.000    0.000    0.000 numeric.py:141ones
147    0.000    0.000    0.000    0.000 {method 'copy' of '_hashlib.HASH' objects}
 49    0.000    0.000    0.000    0.000 {method 'isoformat' of 'datetime.datetime' objects}
196    0.000    0.000    0.000    0.000 hmac.py:83update
 49    0.000    0.000    0.000    0.000 threading.py:1152currentThread
 49    0.000    0.000    0.000    0.000 encoder.py:37replace
 49    0.000    0.000    0.000    0.000 jsonutil.py:102date_default
 49    0.000    0.000    0.000    0.000 {locals}
 49    0.000    0.000    0.000    0.000 hmac.py:100_current
 49    0.000    0.000    0.000    0.000 {method 'close' of '_io.StringIO' objects}
 49    0.000    0.000    0.000    0.000 {method 'digest' of '_hashlib.HASH' objects}
300    0.000    0.000    0.000    0.000 {time.time}
447    0.000    0.000    0.000    0.000 {posix.getpid}
100    0.000    0.000    0.000    0.000 {method 'poisson' of 'mtrand.RandomState' objects}
 49    0.000    0.000    0.000    0.000 hmac.py:119hexdigest
 50    0.000    0.000    0.000    0.000 {method 'gamma' of 'mtrand.RandomState' objects}
 49    0.000    0.000    0.000    0.000 {method 'hexdigest' of '_hashlib.HASH' objects}
 98    0.000    0.000    0.000    0.000 {method 'extend' of 'list' objects}
 49    0.000    0.000    0.000    0.000 threading.py:983ident
 49    0.000    0.000    0.000    0.000 {method 'upper' of 'str' objects}
 49    0.000    0.000    0.000    0.000 {method 'getvalue' of '_io.StringIO' objects}
 49    0.000    0.000    0.000    0.000 {method 'count' of 'list' objects}
 49    0.000    0.000    0.000    0.000 {method 'group' of '_sre.SRE_Match' objects}
 49    0.000    0.000    0.000    0.000 hmac.py:30__init__
147    0.000    0.000    0.000    0.000 {method 'append' of 'list' objects}
  1    0.000    0.000    9.094    9.094 <string>:1<module>
 49    0.000    0.000    0.000    0.000 {thread.get_ident}
 49    0.000    0.000    0.000    0.000 {method 'copy' of 'dict' objects}
 49    0.000    0.000    0.000    0.000 py3compat.py:11no_code
  1    0.000    0.000    0.000    0.000 {method 'disable' of '_lsprof.Profiler' objects}
```

Table 1: Times for likelihood function

	Time (in secs)
Old Likelihood	0.161613
New Likelihood	0.147949

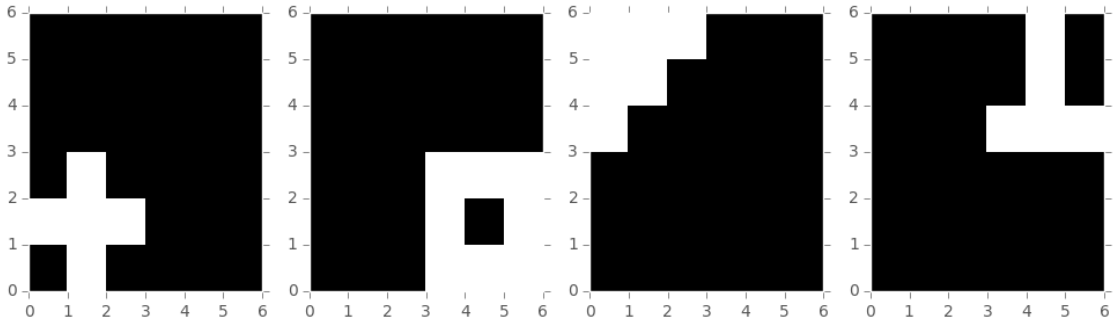

Figure 1: Main features used to simulate data

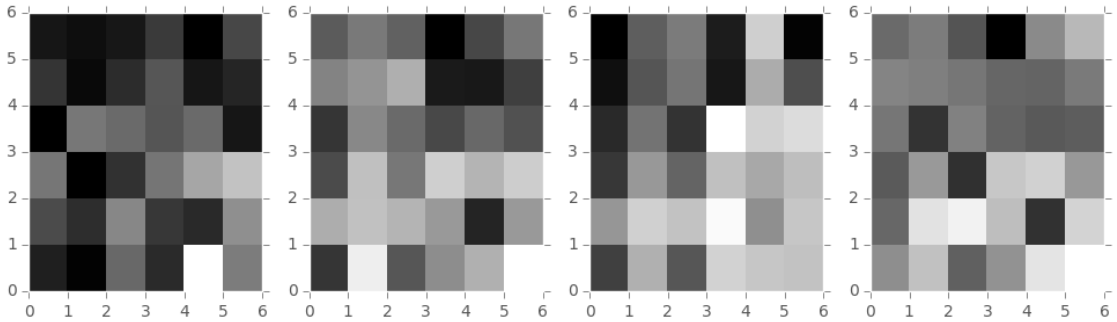

Figure 2: Simulated data

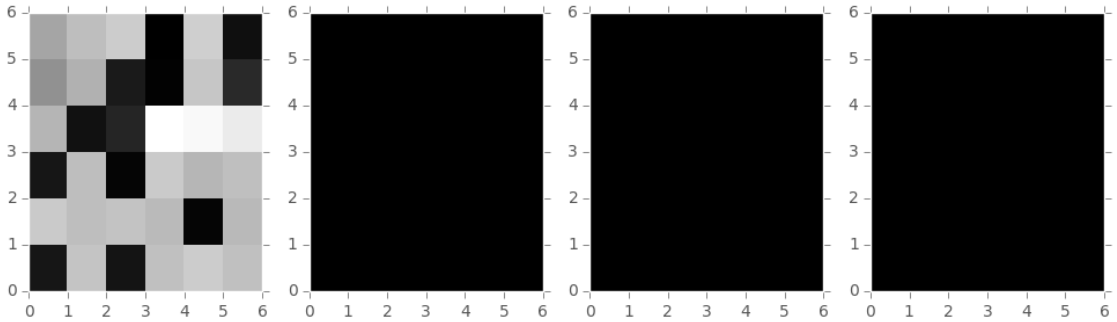

Figure 3: Features detected by code

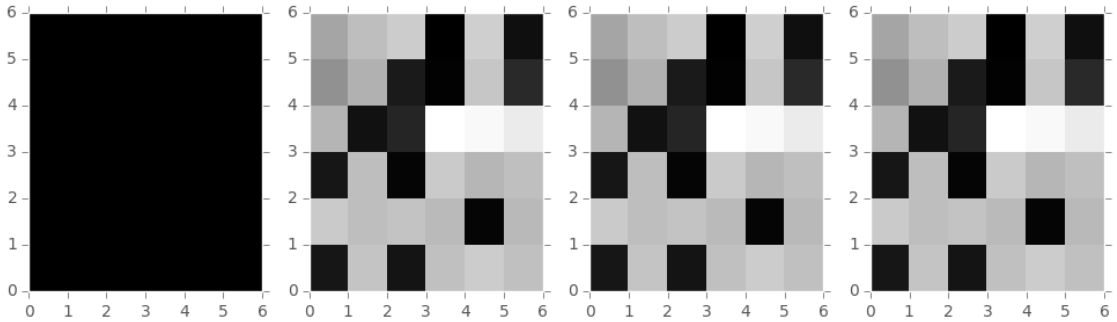

Figure 4: Features possesed by each image in the data

Table 2: Times

	Time (in secs)
Naive	2.280513
Optimized	1.659680
Cython	2.088348

Table 3: Presence of original features

	F1	F2	F3	F4
1st image	0	1	0	0
2nd image	1	1	0	0
3rd image	1	1	0	1
4th image	1	1	0	0

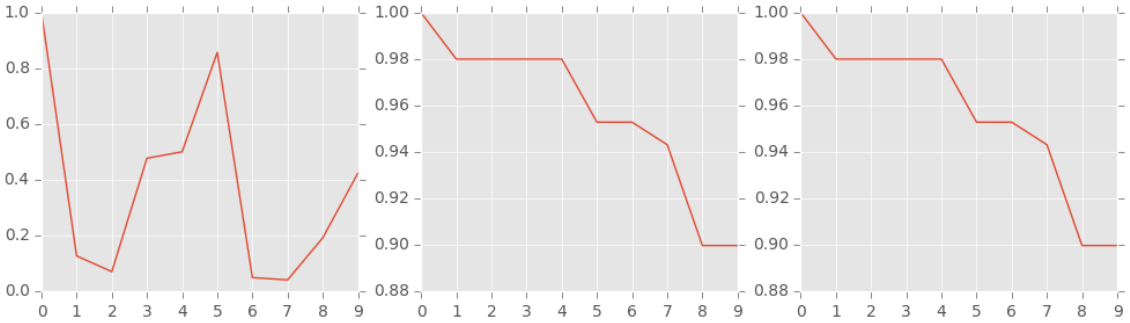

Figure 5: Trace plots for alpha, original sigma X and original sigma alpha