Tricks for Dirichlet-Multinomial in HDP

Donglai Wei

2010.5.12

0)Problem

The biggest problem for Dirichlet-Multinomial is that the Energy formula for tables itself does not contain strong clues about the clustering of different customers at all.

Given the number of customers in every table, the formula does not say anything about the distribution of different customers.
Also, the formula favors small number of tables, which can lead wrong direction during the search.
(t-term)$\underline{log\frac{\Gamma(T+\gamma)}{\Gamma(\gamma)} + \sum_{j=1}^{J}\{log\frac{\Gamma(n_{j..}+\alpha)}{\Gamma(\alpha)} - \sum_{t=1}^{m_{j.}}[log(\Gamma(n_{jt.}) + log\alpha]\}}$

The hope lies in the "communicating power" among the dishes during **Initialization** and **Searching**. (initialize and search the table config with the auxilary of the config of other **restaurants or dishes**)

BAD RESULTS

Hyperparameters:$\alpha = 0.5, \gamma = 1.5, \phi_0 = 0.2$
Though the probability is close to the ground truth, the config is nothing but junk....

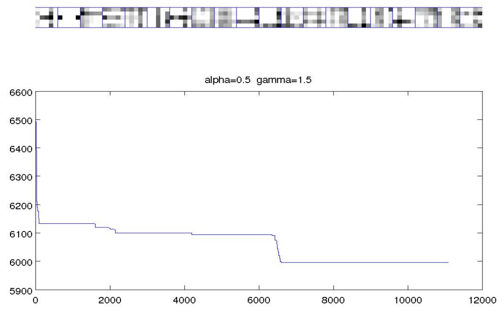


Figure 1: ME result: Simply every restaurant has only one table

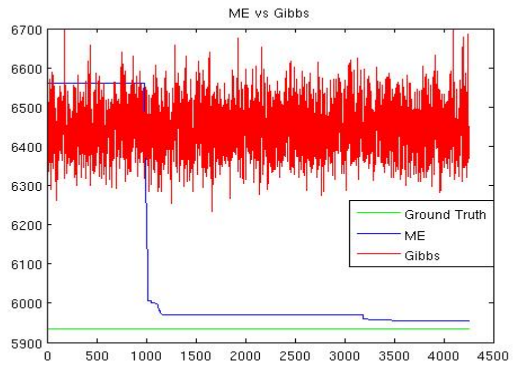


Figure 2: -log probability comparison

2) Initialization?

i)Extreme Case

Bottom-up Initialization: All customers are assigned to different tables with different dishes.
Top-down Initialization: Every Restaurant has only one table with the same dish.

We know they are of no good since they don't capture potential mixture components.

The bad result above uses top-down Initialization.

ii)Auxilary Restaurants

Hoping that some restaurant may have only one mixture component,we have the following initializaiton:

For each of the J restaurants independently:

1. Make J-1 tables serving J-1 dishes, where each dish is informed by all the data from one of the other restaurants
2. Search over assignments t_{ji} of customers to these tables (keep k_{jt} fixed and don't allow the option of making a new table&dish not shared with any other restaurant)
3. Store the at most J-1 tables with at least one customer (hopefully this will be a number bigger than one but much less than J-1)

But the problem is we are still assigning the customer too greedily, which does not capture the distribution of different words:
Actually, **it is equivalent to simply assigning customers with the same type(word) to the same table.**
The difference among the k-term happens at $log\frac{1}{\Gamma(\phi_0+n^{w_0}_{..k})}$ which is concave(bigger $n^{w_0}_{..k}$,bigger decrease)
Thus assign the first customer to the auxilary restaurant with the highest count of word w_0 will mostly decrease the Free Energy(Negative log Probability).

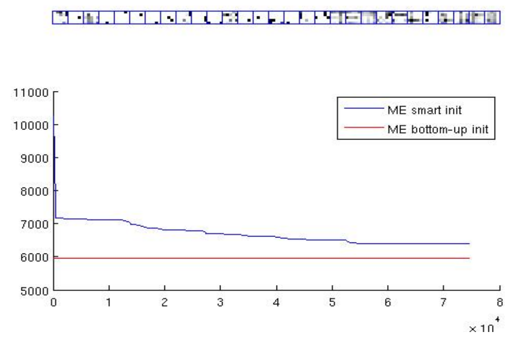


Figure 3: Auxilary Restaurants Initialization ends up with smaller dishes

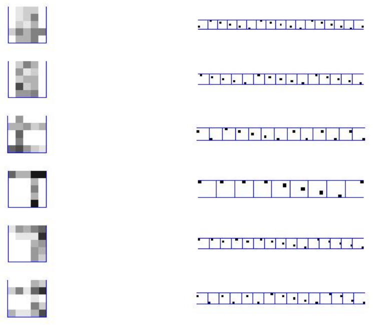


Figure 4: Initialization for the first six restaurants(left)with table config(right)

iii)Gibbs Initialization

One step back, let's see if initializaiton really matters.
Start with an "almost done" Gibbs sampling(500 runs) result, below is the comparison of table configs for the first five restaurants.

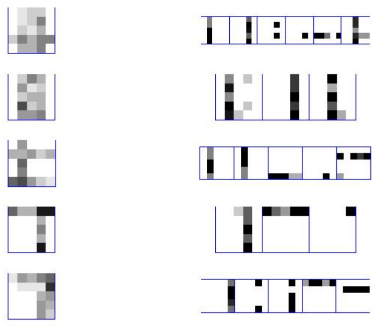


Figure 5: Crude Gibbs Sampling results

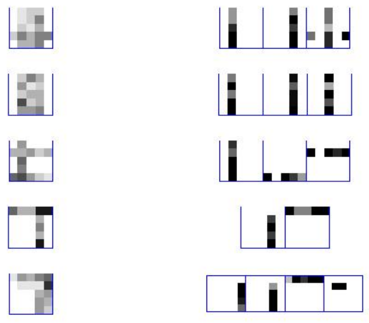
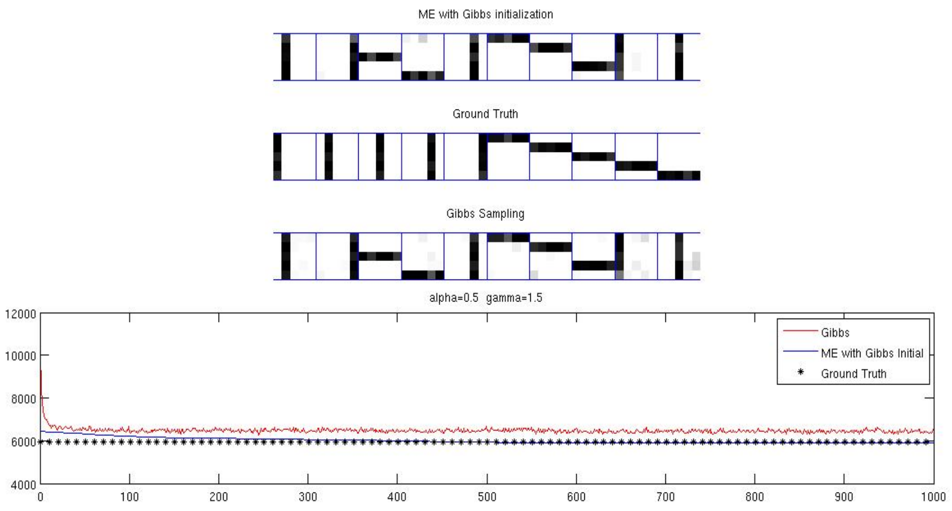


Figure 6: ME results with Gibbs Initialization



1. With rough idea of how the dishes look like, ME algorithm works.
2. Noticably, ME algorithm can group small tables correctly and efficiently by minimizing Free Energy, while it may take longer time for Gibbs sampling.

iv)Finer Auxilary Restaurants

Intuition:(improve the rough Auxilary Restaurants)

1. (Inspired by Gibbs Initialization) We only need to divide each restaurant into **small reliable** tables and ME can do the rest.

2. (Inspired by Auxilary Restaurants)Assigning cutomers one by one may be too greedy, we may try to group them first according to auxilary restaurants.
   (In other words, it may be too utopia that some restaurant is made of one bar, but it could probably happen that the intersection of 2 restaurants may be part of the bar)

So, the idea is: e.g. Given other 39 mixtures of bars, split a mixture of bars into reasonable parts(unoverlapped) and assign customers to them.

```
Psuedocode:How to find the parts of a Restaurant(R)

Preposessing: find the overlapped regions with other Restaurants.
While there are still words in R unassigned to parts
    1. (a) Top-down: define the part as the biggest overlapped region
       (b) (or)Bottom-up: define the part as the smallest overlapped region containing (R-biggest overlapped region)
    2. assign words falling into the new part to it.
    3. R=R-part;Overlapped Region=Overlapped Region-part
end
```

(I choose the bottom-up approach)Below, we can see that the heuristic initializaiton works to some degree(Fig 7), but the result is still not good.
From Fig 8, we can see that the problem lies in **splitting tables**.
From Fig 9, we can see that the correct dishes are found and we only need to **split and merge the noise**.
Since we can not expect too much from the initializaiton, we have to improve our searching methods.

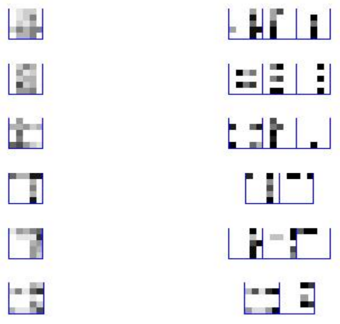


Figure 7: Finer Auxilary Restaurants(FAR) Initialization

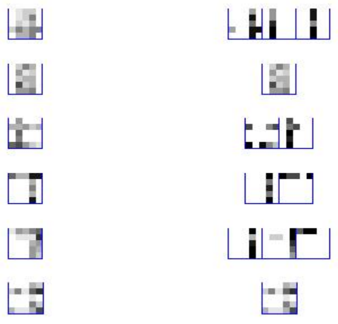


Figure 8: ME results with (FAR) Initialization

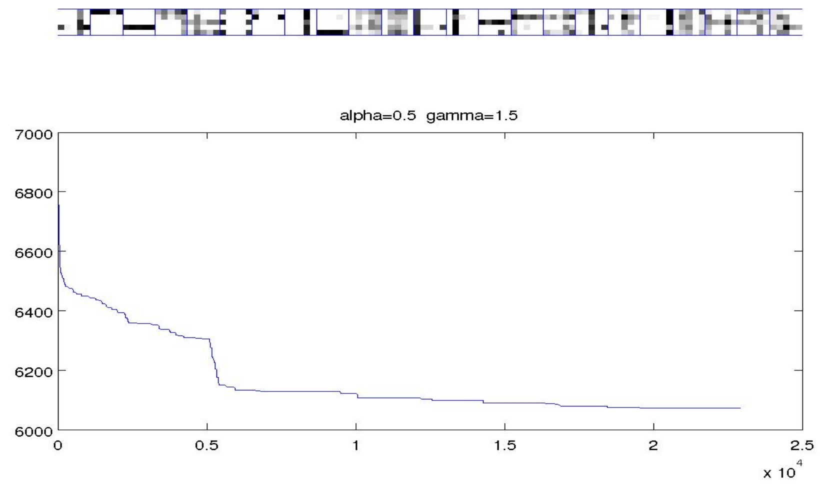


Figure 9: ME results with (FAR) Initialization

3) Searching Scheme?

0) A Little Modification of Spliting dishes

In order to encourage splitting the noisy dishes, we allow merging new dishes to other dishes during the split.

i) Spliting tables with Auxilary Dishes

From Fig8, we can see that previous algorithm to split tables doesn't do a good job.
Preivously:

1. Use 2-means++ to split a table into 2 new tables with new dishes
2. Iteratively local search the table for customers and local search dishes for the 2 new tables

For Normal-Inverse-Wishart-Gaussian case, two-means++ works well because the new tables will penalize for the "inconsistency" of the data themselves.
For Dirichlet-Multinomial case, however, we know that the tables only care about the number of customers instead of their types(words).
So the first step in the alogrithm breaks down by producing super big and super small tables...

Inspired by Initialization with auxilary restaurants, we can **split tables with auxilary dishes**!!!(same pseudo-code)

Below, we can see that split with auxilary dishes does a little better to split the tables.
But still, we have noisy dishes in Fig12.

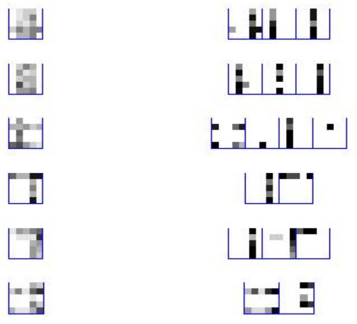


Figure 10: split with auxilary dishes with (FAR) Initialization

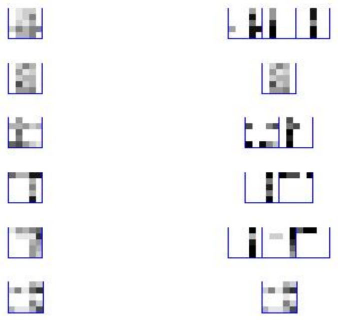


Figure 11: normal split with (FAR) Initialization

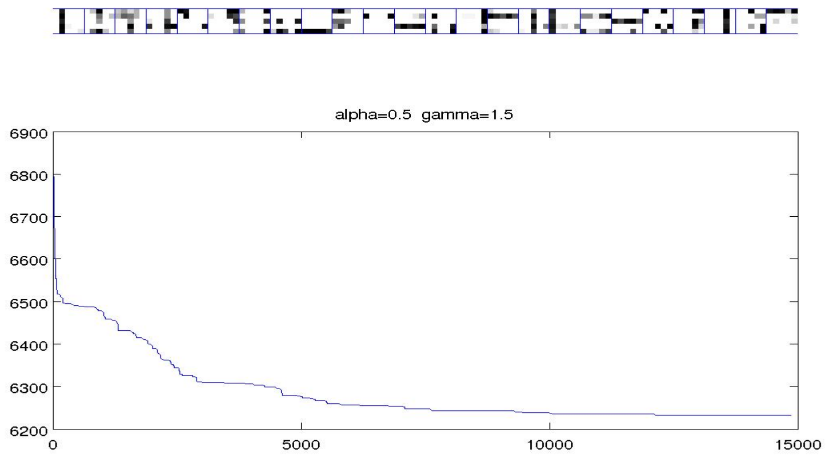


Figure 12: ME results with (FAR) Initialization

ii) Spliting tables with damped "t-term"

In Fig10, it seems natural to split the first table in the third restaurant according to the "right dishes". But the problem is assigning the splitted tables to correct dishes does not decrease enough Free energy. For most of the time, splitting tables increase a little more free energy.(Creating the potential well)

Since we may benifit from smaller tables, we can split tables with damped "t-terms".
Instead of judging the split move by "Δt-term+Δk-term<0", we can use "$\alpha\Delta$t-term+Δk-term<0" where $\alpha \in (0,1)$

Well,IT WORKS!!!!! Below are two different runs with $\alpha = 0.7$:

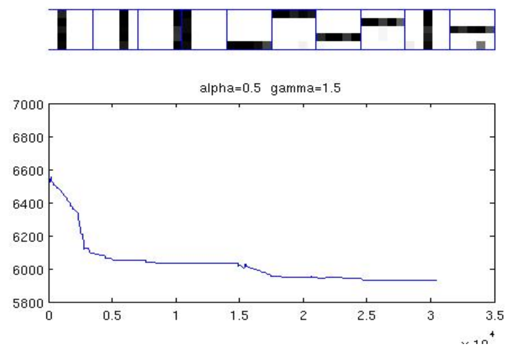


Figure 13: damp+auxilary dishes+(FAR) Initialization

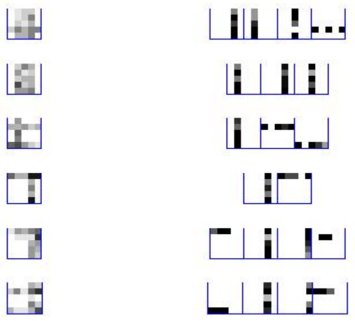


Figure 14: normal split with (FAR) Initialization

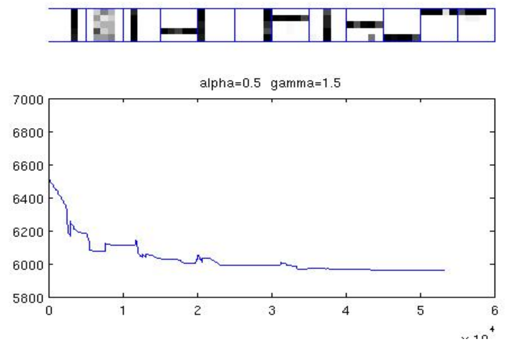


Figure 15: damp+auxilary dishes+(FAR) Initialization

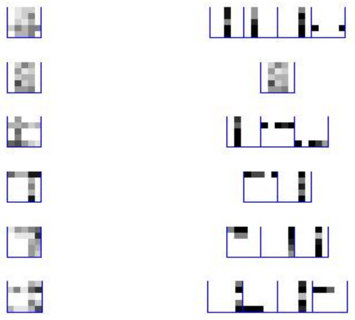


Figure 16: normal split with (FAR) Initialization

iv) Extras

Previously, I was a little inconfident about the "damped t-term" and only ran it with fewer iterations, which "as expected" doesn't provide good result.

So I came up with two extra methods...

1. **"Spliting tables with Re-initializaiton"**
   The problem is that we have found the true dishes(10) and the problem is to get rid of the noisy ones(10).
   Simply, instead of struggling with splitting tables and dishes, we can start it over again!!

   Remember,we achieve this only by initializing with naive parts. So we can reinitialize with current dishes!!!

   As expected, true dishes have bigger chance to be selected

2. **"The Grand Split"** As we mentioned before, we can do a "grand" split of tables instead of do it locally.

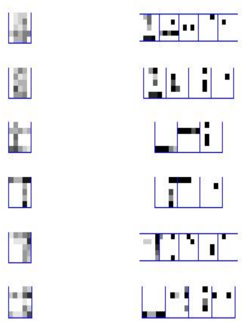


Figure 17: first six restaurants after Reinitialization

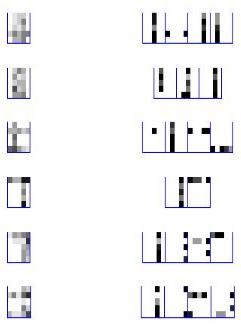


Figure 18: first six restaurants after iterations

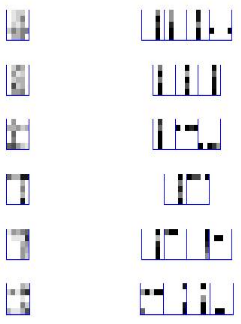


Figure 19: first six restaurants after iterations(another run)

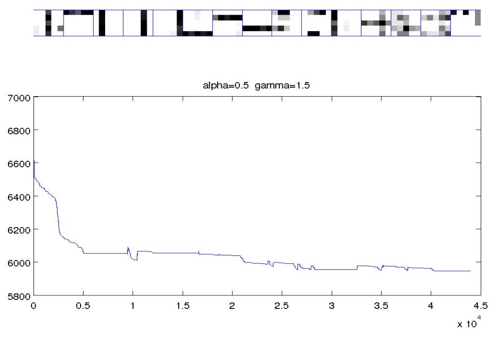


Figure 20: damp+auxilary dishes+(FAR) Initialization

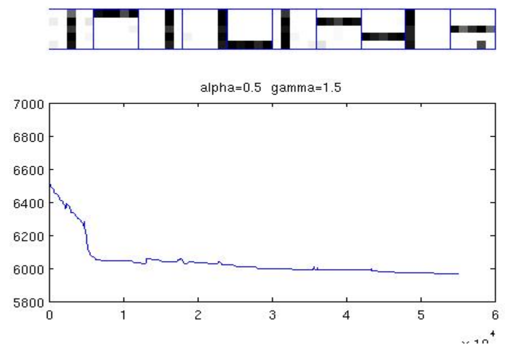


Figure 21: normal split with (FAR) Initialization

Previously, we judge the split table move one restaurant after another independently, now we split them all and find better config for them.

But it doesn't work significantly well...