Modifications

Donglai Wei

2010.7.11

1)Change of t,k-term: Rescue Annealed Local-Table move

Previous Formula:
$-logp(x,z|\lambda)$
$=$
(t-term)$\underline{log\frac{\Gamma(m_{..}+\gamma)}{\Gamma(\gamma)}+\sum_{j=1}^{J}\{log\frac{\Gamma(n_{j..}+\alpha)}{\Gamma(\alpha)}-\sum_{t=1}^{m_{j.}}[log(\Gamma(n_{jt.}))+log\alpha]\}}$

$+$(k-term)$\sum_{k=1}^{K}[log(\frac{\Gamma(n_{..k}+W\phi_0)}{\Pi_{w=1}^{W}\Gamma(\phi_0+n_{..k}^{w})})+log(\frac{\Gamma(\phi_0)^W}{\Gamma(W\phi_0)})-\underline{log(\Gamma(m_{.k}))-log\gamma]}$

(underlined part come from Hierarchical Dirichlet Process)

New Formula:
$-logp(x,z|\lambda)$
$=$
(t-term)$\underline{+\sum_{j=1}^{J}\{log\frac{\Gamma(n_{j..}+\alpha)}{\Gamma(\alpha)}-\sum_{t=1}^{m_{j.}}[log(\Gamma(n_{jt.}))+log\alpha]\}}$

$+$(k-term)$\underline{log\frac{\Gamma(m_{..}+\gamma)}{\Gamma(\gamma)}}+\sum_{k=1}^{K}[log(\frac{\Gamma(n_{..k}+W\phi_0)}{\Pi_{w=1}^{W}\Gamma(\phi_0+n_{..k}^{w})})+log(\frac{\Gamma(\phi_0)^W}{\Gamma(W\phi_0)})-\underline{log(\Gamma(m_{.k}))-log\gamma]}$

1) Previously,t-term wants only 1 table per restaurant while k-term wants every word forms a dish, which requires subtle annealing schedule.

By putting the restriction of $\Gamma(m_{..}+\gamma)$ down to k-term, we now can anneal local-table move avoiding creating too many tables.

Annealing Schedule: $[0.2, 0.4, 0.6, 0.8, 1]^p$
Fixing other parameters(p=0.5), Figure 1 is the comparison of the annealing results for different forulae(anneal both local-table and merge-table)

2) Other Strategies:
i) no aneal m-t,aneal l-t: doesn't work
ii) aneal m-t,no aneal l-t: WORKS
iii) no aneal m-t, no aneal l-t: doesn't work

{Tests from now on use the new formula}

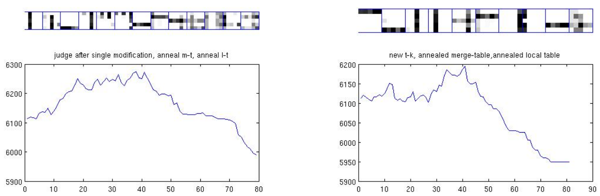


Figure 1: aneal m-t,aneal l-t left: Previous t,k-term, right: new t,k-term

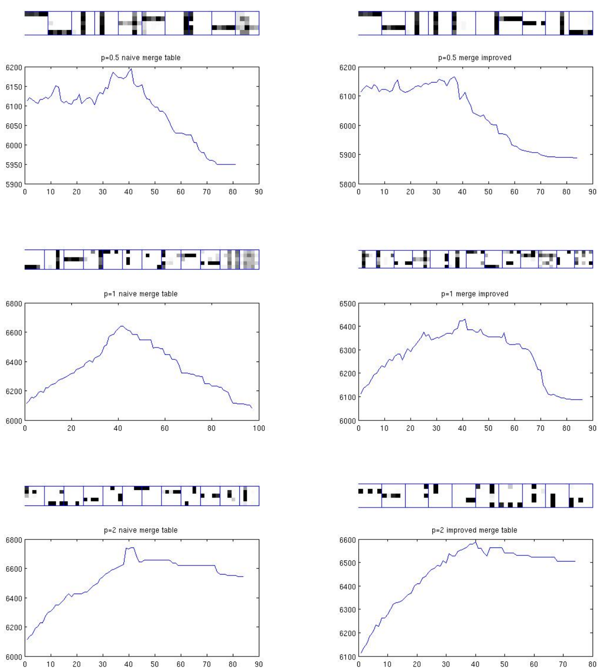


Figure 2: left: old merge-table; right: new merge-table; first row: T=0.5; second row: T=1; third row: T=2;

2) New merge-table

1) Previously, in restaurant j, merge-table only tries find the best table t^* for certain table t to merge while serving k_{jt^*}
2) A better merge-table should also search for the best k for the new merged table.

Fixing other parameters, Figure 2 is the comparison of the merge-table for different Annealing power p$\in [0.5, 1, 2]$.

{Tests from now on use the new merge-table}

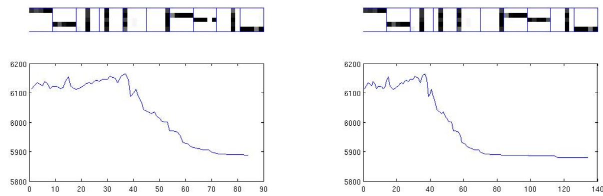


Figure 3: left: Get Stuck; right: Better config by local dish

3) Local Dish

i) Background

So far, for tables, we have:
1) Local-Table-Refinement
2) Local-Search-Dish
3) Merge-Table

But for dishes, we only have:
1) Merge-Dish

ii) Local Maxima

On the left of Figure 3, we expect the third to last dish to be a bar, which shares the word,say w, with the second to last dish.

But, in decompose restaurant, we will never see word w magically go to the third to last dish since the (k-term)sampling likelihood is almost 0, while merge-dish is too cumbersome to help.

It's not a perfect config since word w in ground truth comes from those two dishes. In some restaurants, it will cause small tables serving the second to last dish while the third to last dish is around.

iii) Welcome: Local Dish

Now we only have the building block "table", which forms restaurants and dishes.

There is an another key fundamental part——**Word**
We can do Local-Dish-Refinement by greedily deciding the allocation of one certain word in the dish.(exchage it with another dish)
(Detail is in the pseudo-code description)

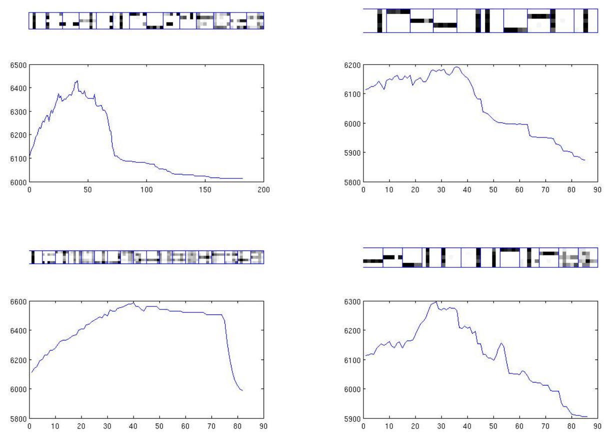


Figure 4: left: Anneal local-table; right: No Anneal local-table first row: p=1;second row: p=2;

4) Remarks

1) I still think it a bad idea to anneal local-table.

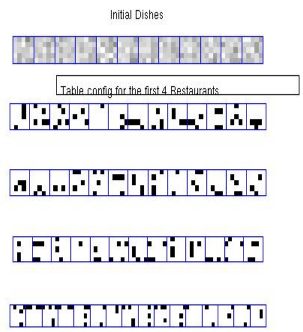
Though annealing local-table works for p=0.5(Temperature Schedule:$[0.2, 0.4, 0.6, 0.8, 1]^p$)
It fails(figure 4,left) for p=1,2 while it would still work(figure 4,right) if we do not anneal local-table. Maybe we can make up other stories for it.

From Above (p<1 is better than others, only anneal merge-table is better than anneal both m-t,l-t), we can see that, we do not need those much annealing to get out of the dominance of t-term.

It is only the "merge-table" that is doing bad without annealing, which is too greedy while the dish config is still vague.

Also, I did not anneal merge-dish and local-dish, which were doing right things without annealing.

If we anneal them, dishes will be less likely to merge or to refine, leading to bad config.



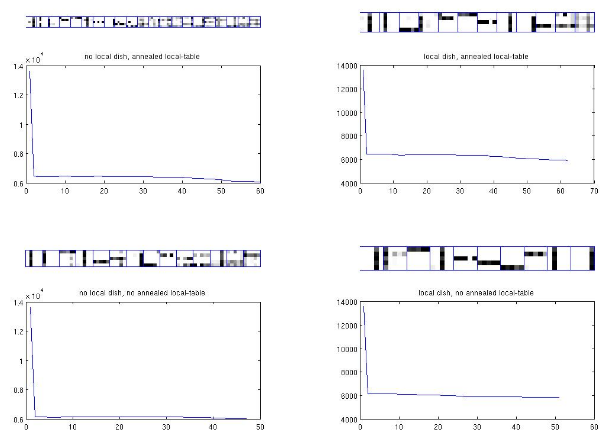


Figure 5: left: No local-dish; right: local-dish first row: annealed local-table;second row: no annealed local-table;

2) Local-Dish move can really make a change:)
Above, it seems local-dish is just for further, minor refinement.

Figure 5 shows that Local-Dish is **"indispensable"**.

I use the same initialization with that from Teh's Gibbs Sampling.(Every Restaurant has 12 tables and there are 12 random flat dish).

i) We still need annealing, since getting t-term better is a lot easier(simply merge-table) than improving k-term.
ii) Again, no anneal local-table is much better
iii) Without Local-Dish, we can still solve it with other annealing schedule. But the point is that with Local-Dish, our alogrithm becomes more robust.

3) By now, we've almost figure out for small(40 5 by5 res) and medium(200 5 by5 res) toy data.
I'm still debugging mex to see what will happen for larger datas.