Annealed ME

Donglai Wei

2010.6.24

0) Notations

Hierarchical Dirichlet Process Model with Dirichlet-Multinomial:

i)Formula

n_{jtk}number of customers in table t, restaurant j, with dish k
m_{jk}number of tables in restaurant j, with dish k
$n_{..k}$number of customers in dish k
$n^w_{..k}$number of occurence of word w in dish k
$n_{j..}$number of customers in in Restaurant j
$n_{jt.}$number of customers in table t in Restaurant j
$m_{..}$number of tables in total
$m_{.k}$number of tables in dish k
J Restaurants,K dishes

a) Goal:(Marginalize θ and Search over z:)
Maximize log Probability $L = logp(x, z|\lambda)$
$=$
(t-term)$\underline{log\frac{\Gamma(\gamma)}{\Gamma(m_{..}+\gamma)} + \sum_{j=1}^{J}\{log\frac{\Gamma(\alpha)}{\Gamma(n_{j..}+\alpha)} + \sum_{t=1}^{m_{j.}}[log(\Gamma(n_{jt.})) + log\alpha]\}}$

$+$(k-term)$\sum_{k=1}^{K}[log(\frac{\Pi_{w=1}^{W}\Gamma(\lambda_0+n^w_{..k})}{\Gamma(n_{..k}+W\lambda_0)}) + log(\frac{\Gamma(W\lambda_0)}{\Gamma(\lambda_0)^W}) + \underline{log(\Gamma(m_{.k})) + log\gamma]}$

(underlined part come from Hierarchical Dirichlet Process)

b) Annealing: Maximize the annealed log Probability:

L'(Temperature)=Temperature*(t-term)+(k-term)

1) ME algorithm:

i) Unified Backbone

(1) Annealing:(n:number of iterations; p:annealing power)

   (i) For iter=1:n
   (ii) Temperature=$(\frac{iter}{n})^p$
   (iii) Decompose Restaurants(Temperature)
   (iv) Merge Dish(Temperature)
   (v) End

(2) Running for Convergence:

   (i) While L doesn't increase any more
   (ii) Decompose Restaurants(1)
   (iii) Merge Dish(1)
   (iv) End

ii) Decompose Restaurants(Temperature)

For j=Randperm(J)

(A) Rough reconfiguration for Restaurant j:

   (i) Make Restaurant j into one table t_0 where customers following uniform distribution:
   (%Thus the Probability $P(t_{ji} = t_0)$=$\frac{1}{W}$)
   (ii) Possible Dish={$Nonempty\ dishes$}
   (iii) While Possible Dish is not empty:
      (a) For each dish k∈Possible Dish, propose to form a new table t_k out of t_0 with dish k and calculate the change for these two dishes Δk:
      (%For each customer i in t_0, sample $t_{ji} \in \{t_0, t_k\} \sim \{\frac{1}{W}, \frac{n^w_{..k}+\phi}{n_{..k}+W\phi}\}$)
      (%Propose to form table t_k with customers whose $t_{ji} = t_k$) Sample a proposal t_{k*} according to the weight and make the new table:
      (%Sample a proposal $\{t_{k_1}, ..., t_{k_K}\} \sim e^{r_{proposal}\{\Delta k_1, ...\Delta k_K\}}$)
      (%$r_{proposal} > 0$, the more decrease of Δk, the less propable to form table t_k)
      (b) Possible Dish=Possible Dish$\backslash k_*$
   (iv) If there are still customers left in t_0,make it a new table with a new dish

(B) Refined reconfiguration for Restaurant j:TKM(j,**Temperature**):
(%Divide the change of L between present Restaurant j config and its previous config into t-term,k-term change: ΔL=ΔK+ΔT)

(C) Decision:
If(ΔK + **Temperature** $* \Delta T$ <0):
Accept the new configuration
else:
Restore Previous Config

End

iii) Merge Dish(Temperature)

Dish List={$Nonempty\ dishes$}

While Dish List is not empty:
1) Randomly pick a dish k$\in$ Dish List
2) Dish List=Dish List\k
3) Merge dish k to the dish$\in$ Dish List which increase **L'(Temperature)** mostly
(if cannot increase **L'(Temperature)**, then leave it alone)
End

iv) TKM(Restaurant index,Temperature)

While **L'(Temperature)** doesn't increase any more:

(A) Local Search Table:

For t_1=Randperm($m_{j.}$):
For t_2=Randperm($m_{j.} \setminus t_1$):

While local move can be made:
Greedily move one customer at a time from t_1 to t_2 if the move increases L
End

End
End

(B) Local Search Dish:

For t_1=Randperm($m_{j.}$):
Assign t_1 to the dish which increase L most(allow it to have new dish)
End

(C) Merge table:

For t_1=Randperm($m_{j.}$):
Merge table t_1 to the table in j which increase **L'(Temperature)** most
(if cannot increase **L'(Temperature)**,then leave it alone)
End

End

2) Anealing

Things to play with:

1. Parameters to tune:
   (a) Annealing Scheme:n:number of iterations; p:annealing power
   (b) $r_{proposal}$:constant or annealed? (we may want it peaky in the beginning and allow more variability later on)
2. Functions to Anneal:("Merge table" is annealed,"Local-Search-Dish" doesn't change t-term, thus no anneal needed)
   1) Anneal "Local-table" and "Merge dish"?

P.S. The tests on done on 5 by 5 matrix(10 bars) with 40 restaurants, because the problem becomes easier with more restaurants.

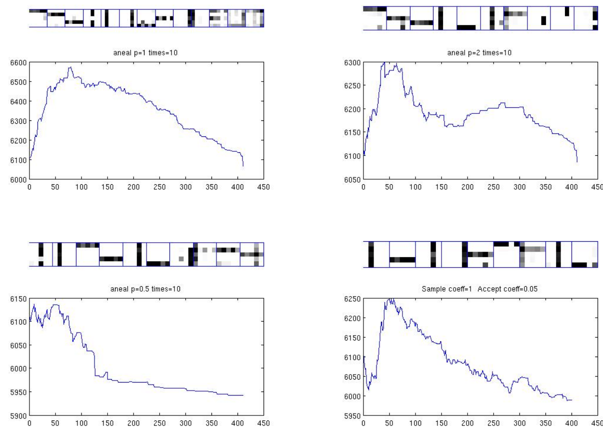

Figure 1: Down-Right is previous no-annealing result,the annealed ones(constant $r_{proposal}$) donot look good

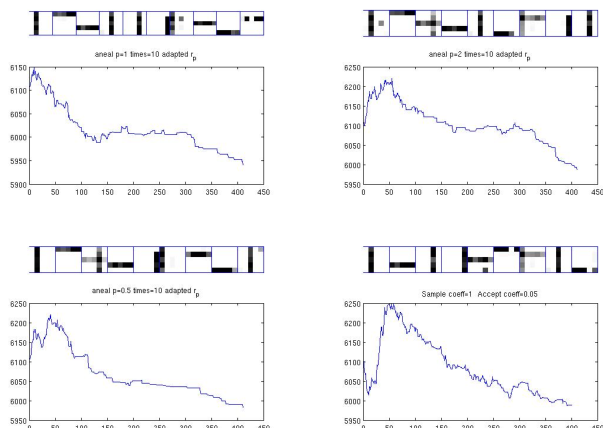

Figure 2: Using annealed $r_{proposal}$, p=1 is even better than the no-annealed one(down-right) by finding right number bars

i)Tuning parameters:

1) Annealing Scheme:n=10; p$\in$\{0.5,1,2\};

2) Annealed $r_{proposal}$;

Simply, I set $r_{proposal}$=$\frac{1}{T}$, where we want harder proposal assigment in the beginning and softer one later on.

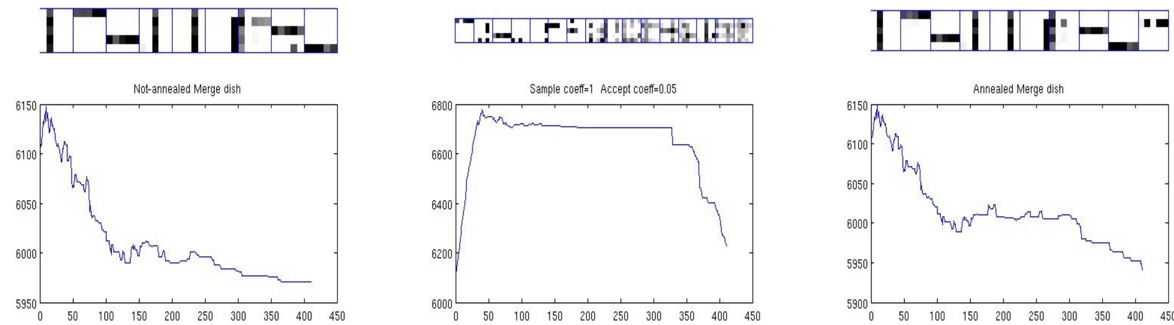

Figure 3: (left)no anneal local-table,merge-dish;(mid)anneal local-table only(right)anneal merge-dish only

ii)Annealed Local-table and Merge Dish:

From Figure 3, we can see that:

1) annealing merge dish(right) can be also useful to prevent mixture of bars;
2) annealing local table(left) is harmful to prevent tables from communicating with each other;