Complete Pseudo-Code for ME

Donglai Wei

2010.7.11

0) Notations

Hierarchical Dirichlet Process Model with Dirichlet-Multinomial:

i) Settings

Hyper-parameter:
α, γ: HDP concentration parameter
$\vec{\lambda}$: Prior for Dirichlet Distribution(W=dim($\vec{\lambda}$):number of different words;uniform prior:$\lambda_1, ..., \lambda_W = \lambda_0$)
Hidden Variable:
(M-step)z: Discrete assignment(t_{ji}, k_{jt} correspond to customer,table assignment in Chinese Restaurant Franchise)
(E-step)θ: Multinomial parameter
Observation:
x:$\in$(1,...W)

ii)Formula

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

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

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

(underlined part come from Hierarchical Dirichlet Process)
b) Annealing: Maximize the annealed log Probability:

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

1) ME algorithm:

J Restaurants,K dishes

i) Backbone

(1) Initialization:

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

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

(3) Run for Convergence:

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

ii) Decompose Dish(Negative Temperature:Temperature)

For k=Randperm(K)

(A) Rough Reconfig of Dish k

- (i) For j=Restaurants which have tables serving dish k
- (ii) Decompose Restaurant(j,Temperature,k);
- (iii) END

(B) Further Refinement of Dish k

- (i) LM-Dish(k): *(Local-Search and Merge Move for Dish k)*

(C) Decision

- (i) IF(ΔK + **Temperature** $* \Delta T$ <0):
- (ii) Accept new config
- (iii) END

END

iii) Decompose Restaurants(Restaurant index:j,Negative Temperature:Temperature,The Left Out Dish: k_0)

(A) Rough Reconfig of 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=$\{Nonempty\ dishes\} \setminus k_0$

(iii) WHILE Possible Dish is not empty:

(a) FOR each dish k$\in$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$\setminus k_*$
$t_0 = t_0 \setminus t_{k_*}$

(c) END

(iv) IF there are still customers left in t_0,make it a new table with a new dish K+1

(B) Further Refinement of Restaurant j:

(i) LM-Restaurant(j,Temperature):
(Local-Search and Merge Move for tables in Restaurant j) (%Calculate the change of L between present Restaurant j config and its previous config: ΔL=ΔK+ΔT)

(C) Decision:

(i) IF(ΔK + **Temperature** $* \Delta T$ <0):

(ii) Accept the new configuration

(iii) ELSE:

(iv) Restore Previous Config

(v) END

iv) LM-Restaurant(Restaurant index:j,Negative Temperature:Temperature)

While L 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'(Temperature)**
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 with the best dish k, which increase **L'(Temperature)** most
(if cannot increase **L'(Temperature)***,then leave it alone)*
End

END

v) LM-Dish(Dish index:k)

(A) IF Dish k is not empty:(Local Dish):

Dish List=$\{Nonempty\ dishes\} \setminus k$
FOR w=Randperm(Words Occur in Dish k):

i) Find promising dishes to exchange word w
(find the dishes that appear most often with the tables having word w in dish k)
FOR kk=Dish List
Same Restaurant(kk)=index of restaurants in dish kk, that have tables serving dish k
Promising Dish(kk)=length(Same Restaurant(kk))
END

ii) Find the best approximated Reallocation of word w:
ΔL=0;
FOR kkk=find(Promising Dish==max(Promising Dish)):

a)Naive Reallocation:
For j=Same Restaurant(kkk)
Assign all of words w from the table serving dish k to the table serving dish kkk
END
Calculate the change of L between present config and previous config:l_0 b)Gready Search:
Δl=1
WHILE Δl >0
Δl_k=ΔK(assign one word w from dish kkk back to dish k)
Δl_t=max(ΔT(assign one word w from dish kkk back to dish k in Restaurant j$\in$Same Restaurant(kkk)))
$\Delta l = \Delta l_k + \Delta l_t$
END
$\Delta L = max(\Delta L, l_0 + \Delta l)$
END

iii)Decision:
IF ΔL >0
Accept new config
ELSE
Restore previous config

END

END

(B) IF Dish k is still not empty:(Merge Dish)

Dish List=$\{Nonempty\ dishes\} \setminus k$
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

(C) END

(D) END