Complete Pseudo-Code for ME

Donglai Wei

2010.6.22

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

$n_{..k}$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

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)

1) ME algorithm:

i) Backbone

(1) Find bars:

- (i) While the number of dishes doesn't change any more
- (ii) Decompose Restaurants(Hard Proposal Assignment,Merge Dish,Accept all)
- (iii) End

(2) Find Higher log Probability L:

- (i) While L doesn't increase any more
- (ii) Decompose Restaurants(Soft Proposal Assignment,Merge Table,Accept/Reject)
- (iii) End

ii) Decompose Restaurants(Proposal option,TKM option,Decision option)

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 ΔL_k:
    (%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 $t_{ji} = t_k$)
  - (b) If(**Proposal option**==Hard Assignment Proposal):
    Find the proposal t_{k*} which has the biggest ΔP and make the new table:

    elseif(**Proposal option**==Soft Assignment Proposal):
    Sample a proposal t_{k*} according to the weight and make the new table:
    (%Sample a proposal $\{t_{k_1}, ..., t_{k_K}\} \sim e^{r_{proposal}\{\Delta L_{k_1}, ... \Delta L_{k_K}\}}$)
    (%$r_{proposal} > 0$, the more decrease of ΔL_k, the less propable to form table t_k)
  - (c) 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,**TKM option**):
(%Calculate the change of L between present Restaurant j config and its previous config: ΔL)

(C) Decision:
If(**Decision option**==Accept):
Accept the new configuration

elseif(**Decision option**==Accept/Reject):
Accept the new config with Probability min$\{e^{r_{accept}\Delta L}, 1\}$
(%$r_{accept} > 0$, if L increase,always accept; otherwise more decrease,more likely to reject)

End

ii) TKM(Restaurant index,TKM option)

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
End

End

(B) Local Search Dish:

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

(ii) Merge tables with the same dish(since we don't have merge table move during "Find Bars")

if(**TKM option**==Merge table):

(C) Merge table:
While no more changes of table assignment and dish assignment can increase P:
b=rand([0,1])

Switch (ceil(b*2)):
case 1: Randomly pick a table in restaurant j, merge it to the table in j which increase L most(if cannot increase L,leave the table alone)
case 2: Randomly pick a table in restaurant j, assign it the dish which increase L most(allow it to have new dish)
End

End

elseif(**TKM option**==Merge Dish)

(D) Merge dishes: Dish List=$\{k : \exists t \ s.t. \ k_{jt} = k\}$

While no more changes of dish assignment can increase L:
Randomly pick a dish, merge it to the dish which increase L mostly(if cannot increase L,leave the dish alone)
End