CODE formula

Donglai Wei

2010.5.12

1) General Formula

W:number of unique words
$n_{..k}$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
$-logp(x,z|\lambda)$
$=$
(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)$\sum_{k=1}^{K}[log(\frac{\Gamma(n_{..k}+W\phi_0)}{\Gamma(W\phi_0)}) + log(\Pi_{w=1}^{W}\frac{\Gamma(\phi_0)}{\Gamma(\phi_0+n^{w}_{..k})}) - \underline{log(\Gamma(m_{.k}) - log\gamma]}$

2) Split-Merge Search Scheme

i) Local Search table

Goal: Find the best table for Customer i in Restaurant j with word w and previous table t_{ji}
(t-term is underlined,k-term is not)

(A) Cost of moving the customer out of previous table/dish

- (a) if $n_{jt_{ji}.}$=1(i was the only customer in the table):$m_{j.}$=$m_{j.}$-1
  - i. if $n_{..k_{jt_{ji}}}$=1(i was the only customer in the dish):$\underline{-log(\Sigma_j m_{j.} - 1 + \gamma) + log(\alpha)} + log(\gamma) - log(W)$,
  - ii. else:$\underline{-log(\Sigma_j m_{j.} - 1 + \gamma) + log(\alpha)} + log(m_{.k} - 1) + log(\phi_0 + n^{w}_{..k} - 1) + log(n_{..k} - 1 + W\phi_0))$,
- (b) else:(cannot be the only customer in the dish):$\underline{log(n_{jt_{ji}.} - 1)} + log(m_{.k} - 1) + log(\phi_0 + n^{w}_{..k} - 1) - log(n_{..k} - 1 + W\phi_0))$,

update $m_{j.}, classes\{k_{jt_{ji}}\}$

(B) Cost of moving the customer to table t^{*}

- (a) if $n_{jt^{*}.}$=0(a new table): find the best dish for it

i. if assign the new table to an old dish: $\underline{log(\Sigma_j m_{j.}+\gamma)-log(\alpha)}-log(m_{.k})-log(\phi_0+n^w_{..k})+log(n_{..k}+W\phi_0))$

ii. else: $\underline{log(\Sigma_j m_{j.}+\gamma)-log(\alpha)}-log(\gamma)+log(W)$

(b) else: t^* is not a new table

i. $\underline{-log(n_{jt^*.})}-log(\phi_0+n^w_{..k_{t^*}})+log(n_{..k_{t^*}}+W\phi_0)),$

iii) Local Search dish

Goal: Find the best dish for table t in Restaurant j with previous dish k

(A) Cost of moving the table out of previous dish

(a) if $n_{..k_{jt}}=n_{.jt}$(t was the only table in the dish): -classes$\{k\}.F$

(b) else: $log(m_{.k}-1)+log(\frac{\Gamma(n_{..k}-n_{.jt}+W\phi_0)}{\Gamma(n_{..k}+W\phi_0)}+log(\Pi_{w=1}^W\frac{\Gamma(\phi_0+n^w_{..k})}{\Gamma(\phi_0+n^w_{..k}-n^w_{jt.})})$

(B) Cost of moving the table to dish k^*

(a) if $n_{..k^*}$=0(new dish): $log(W)-log(\gamma)$

(b) else: $-log(m_{.k})+log(\phi_0+n^w_{..k})-log(n_{..k}-1+W\phi_0))$

2) Decompose Search Scheme

Now the variable is z_{ji} and $m_{j.}$
Previously, tables are grouped by Restaurants: $log(\Gamma(n_{jt.}))$;
Here, we group them by dishes: $log(\Gamma(n_{.t^*k}))$;

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

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

=
(z-term)+ $\sum_{k=1}^K[log(\frac{\Gamma(n_{..k}+W\phi_0)}{\Pi_1^W\Gamma(\phi_0+n^w_{..k})\Gamma(m_{.k})\Pi_{t^*=1}^{m_{.k}}\Gamma(n_{.t^*k})})]$

(TK-term)$-Tlog\alpha-Klog\gamma+Klog\frac{\Gamma(\phi_0)^W}{\Gamma(W\phi_0)}+log\Gamma(T+\gamma)$

(constant-term)$-log\Gamma(\gamma)+\sum_{j=1}^J log\frac{\Gamma(n_{j..}+\alpha)}{\Gamma(\alpha)}$

Observation:

1. TK-term:

(i) T$\rightsquigarrow$T-1: $\Delta(TK-term)$=log(α)-log($T-1+\gamma$) <0 a.e.

(ii) K$\rightsquigarrow$K-1: $\Delta(TK-term)$=log(γ)-log($\frac{\Gamma(\phi_0)^W}{\Gamma(W\phi_0)}$) <0 a.e.

1) 2)
1) (Decompose dish) K$\rightsquigarrow$K-1: 1) (Decompose table)T$\rightsquigarrow$T+1:
1)(Decompose table in a restaurant)