Outline: ME algorithm

Donglai Wei

2010.9.4

1) Formula

Hyper-parameter Φ:
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)

Free Parameter:
$n^w_{j.k}$number of occurence of word w in dish k in restaurant j

Counting Statistics:
J Restaurants,K dishes
$n_{jtk} = \sum_{w=1}^{W} \delta(k_{jt} - k) n^w_{j.k}$number of customers in table t Restaurant j serving dish k *(δ :Dirac delta function)*
$n_{j.k} = \sum_{w=1}^{W} n^w_{j.k}$number of customers serving dish k
$n^w_{..k} = \sum_{j=1}^{J} n^w_{j.k}$number of occurence of word w in dish k
$n_{j..} = \sum_{k=1}^{K} n_{j.k}$number of customers in in Restaurant j
$m_{jk} = 1 - \delta(\sum_{w=1}^{W} n^w_{j.k})$number of tables in Restaurant j serving dish k
$m_{.k} = \sum_{j=1}^{J} m_{jk}$number of tables in dish k
$m_{..} = \sum_{k=1}^{K} m_{.k}$number of tables in total

(Marginalize θ and Search over $n^w_{j.k}, \forall j, k, w$)
Goal:Maximize log Probability: L=$log(p(\vec{x}, \vec{z}|\Phi)$

a) Original Formula

$$\begin{aligned}
L &= log(p(\vec{x},\vec{z}|\Phi)) = log \int_\theta p(\vec{x},\vec{z},\theta|\Phi)\, d\theta = log \int_\theta p(\vec{x},\theta|\vec{z},\Phi)p(\vec{z}|\Phi)\, d\theta \\
&= log(p(\vec{z}|\Phi)) + log \int_\theta p(\vec{x},\theta|\vec{z},\Phi)\, d\theta \\
&= \textit{(HDP stochastic process term)} log\{\frac{\Gamma(\gamma)}{\Gamma(m_{..}+\gamma)}\Pi_{k=1}^{K}[\Gamma(m_{.k})]\gamma^K\} + \sum_{j=1}^{J} log\{[\frac{\Gamma(\alpha)}{\Gamma(n_{j..}+\alpha)}\Pi_{k=1}^{K}(\Gamma(n_{j.k}))]\alpha^{m_{..}}\} \\
&+ \textit{(Likelihood term)} \sum_{k=1}^{K}\{log(\frac{\Gamma(W\lambda_0)}{\Gamma(n_{..k}+W\lambda_0)}) + \sum_{w=1}^{W} log(\frac{\Gamma(\lambda_0+n^w_{..k})}{\Gamma(\lambda_0)})\}
\end{aligned}$$

b) z-m View(Decompose Dish)

The objection function can be roughly divided for each dish k.
In terms of J-W-K coordinate, we are trying to find the best config for each J-W plane to maximize L.
The sub objection function for each k(J-W plane) is composed of the counts of customers:
1)in J direction for every w
2)in W direction for every j
3)on the whole plane.

$$\begin{aligned}
L &= logp(\vec{x},\vec{z}|\Phi) \\
&= \textit{(z-term)} \sum_{k=1}^{K}[log(\frac{\Pi_{w=1}^{W}\Gamma(\lambda_0+n^w_{..k})\ \Pi_{j=1}^{J}\alpha\Gamma(n_{jkt_k})}{\Gamma(n_{..k}+W\lambda_0)}) + log(\gamma\frac{\Gamma(W\lambda_0)}{\Gamma(\lambda_0)^W})] \\
&+ \textit{(m-term)} log(\frac{\Pi_{k=1}^{K}\Gamma(m_{.k})}{\Gamma(m_{..}+\gamma)}) \\
&+ \textit{(constant)} \sum_{j=1}^{J}[log\frac{\Gamma(\alpha)}{\Gamma(n_{j..}+\alpha)}] + log(\Gamma(\gamma))
\end{aligned}$$

In order to improve the config of dish k, we implement **Decompose Dish** move:
1) Delete dish k and then reconfig the involved restaurants and words.
2) If new dishes are proposed to increase L, try to merge them to old dishes.

c)t-w-k View(Decompose Restaurant/Word)

In order to imporve the config of involved restaurants and words, we need to rewrite the formula as follows.

$$\begin{aligned}
L &= logp(x,z|\lambda)\\
&= \textit{(t-term)}\sum_{j=1}^{J}\{\sum_{k=1}^{K}[log(\Gamma(n_{j.k}))+log\alpha]\}\\
&+ \textit{(w-term)}\sum_{w=1}^{W}\{\sum_{k=1}^{K}[log(\Gamma(\lambda_0+n^{w}_{..k}))-log(\Gamma(\lambda_0))]\}\\
&+ \textit{(k-term)}log\frac{1}{\Gamma(m_{..}+\gamma)}+\sum_{k=1}^{K}[log(\frac{\Gamma(W\lambda_0)}{\Gamma(n_{..k}+W\lambda_0)})+log(\Gamma(m_{.k})+log\gamma]\\
&+ \textit{(constant)}\sum_{j=1}^{J}[log\frac{\Gamma(\alpha)}{\Gamma(n_{j..}+\alpha)}]+log(\Gamma(\gamma))
\end{aligned}$$

(define $log(\Gamma(n_{j.k}))=0$*,if* $n_{j.k}=0$*)*
Since we are going to initialize with "Every restaurant has only one table and its own dish" which already gives the best t-term, we should anneal t-term during the heuristic search.

Thus the object function during annealing is :
L'=Temperature*t-term+(k-term+w-term)
which is the same as previous $t_{ji}-k_{jt}$ view.
Symmetrically, if we initialize with "Every word is a dish" which gives the best w-term, we should anneal w-term instead.

2) ME algorithm:

J Restaurants,K dishes

i) Backbone

$\%a)Initialization:$
```
    Every restaurant has only one table and its own dish
```
$\%b)Annealing:(n:number\ \ of\ \ iterations;\ \ p:annealing\ \ power)$
```
    For iter=1:n
        Temperature=(iter/n)^p
        Decompose Dish(Temperature)
    End
```

$\%c)Run\ \ for\ \ Convergence:$
```
    While L doesn't increase any more:
        For j=randperm(J):
            Decompose Restaurants(j,1,0)
        End
        For w=randperm(W):
            Decompose Word(w,1,0)
```

```
        End
        Decompose Dish(1)
    End
```

ii) Decompose Dish(Negative Temperature:Temperature)

```
For k=Randperm(K)

  %a)Reconfig without Dish k
    For j=Restaurants which have tables serving dish k
        Decompose Restaurant(j,Temperature,k);
    End
    For w=words which appear in dish k
        Decompose Word(w,Temperature,k);
    End

  %b)Merge new proposed dishes
    For k=new proposed dishes
        Merge Dish(Temperature,k);
    End

  %c)Decision
    If(Δ k-term + Δ w-term + Temperature * Δ t-term <0):
        Accept new config
    End

End
```

iii) Decompose Restaurants(Restaurant index:j,Negative Temperature:T,The Decomposed Dish: k_0)

%a)Rough Reconfig Restaurant j

Make Restaurant j into one table t_0 where customers following uniform distribution

$\%(P(t_{\mathtt{ji}} = t_0) = \frac{1}{W})$

Possible Dish=$\{Nonempty \quad dishes\} \setminus \ k_0$

While Possible Dish is not empty:

For each dish $k \in$Possible Dish:

For each customer i in t_0:

sample $t_{\mathtt{ji}} \in \{t_0, t_{\mathtt{k}}\} \sim \{\frac{1}{W}, \frac{n_{\cdot\cdot\mathtt{k}}{}^{\mathtt{w}}+\phi}{n_{\cdot\cdot\mathtt{k}}+W\phi}\}$

End

Propose to form table $t_{\mathtt{k}}$ with customers whose $t_{\mathtt{ji}} = t_{\mathtt{k}}$

Calculate the change $d_{\mathtt{k}}$ for k-term and w-term:

End

%Sample a proposal $t_{\mathtt{k}*}$ according to the weight and make the new table:

Sample a proposal $\{t_{\mathtt{k}_1}, ..., t_{\mathtt{k}_\mathtt{K}}\} \sim \ e^{\mathtt{r}_{\mathtt{proposal}}\{\mathtt{d}_{\mathtt{k}_1}, \ldots, \mathtt{d}_{\mathtt{k}_\mathtt{K}}\}}$

$\% r_{\mathtt{proposal}} > 0$, the more decrease of $d_{\mathtt{k}}$, the less propable to form table $t_{\mathtt{k}}$

Possible Dish=Possible Dish$\setminus \ k_*$

$t_0 = t_0 \ \setminus \ t_{\mathtt{k}_*}$

End

```
If there are still customers left in t_0:
   make it a new table with a new dish K+1
End

%b)Further Refinement of Restaurant j(Local-Search and Merge Move for tables in Restaurant j)

LM-Restaurant(j,T)

%c)Decision

Calculate the change of L between present Restaurant j config and its previous config:
```

$\Delta\ L' = \Delta\ k - term + \ \mathbf{T}\Delta\ t - term + \Delta\ w - term$

```
If(Δ L' <0):
    Accept the new configuration
Else:
    Restore Previous Config
End
```

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

```
While L doesn't increase any more:

     %a)Local Search Table:
        For t_1=Randperm(m_j.):
             For t_2=Randperm(m_j. \ 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'(T)
                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 with the best dish k, which increase L'(T) most
            %if cannot increase L'(T),then leave it alone
        End

End
```

v) Decompose Word(Word index:j,Negative Temperature:T,The Decomposed Dish: k_0)

The only difference from "Decompose Restaurant" is:

In DR, sample according to the w-term to approximate t-term:$t_{ji} \in \{t_0, t_k\} \sim \{\frac{1}{W}, \frac{n^w_{..k}+\phi}{n_{..k}+W\phi}\}$
In DW, sample according to the t-term to approximate w-term:$t_{ji} \in \{t_0, t_k\} \sim \{log(\alpha), log(n_{jt.})\}$

vi) LM-Word(Dish index:k)

Similar to LM-Restaurant:

In DR, tables are the projection from dishes to the Restaurant j
In DW, tables are the projection from dishes to the Word w

vii) Merge-Dish(Dish index:k,Negative Temperature:T)

```
Dish List={Nonempty dishes} \ k

While Dish List is not empty:
    Randomly pick a dish k∈ Dish List
    Dish List=Dish List\k
    Merge dish k to the dish∈ Dish List which increase L'(T) mostly
    %if cannot increase L'(T), then leave it alone
 End
```