**TEXT MINING FOR SOCIAL SCIENCES**
**PROBLEM SET 3**

**Maria Fernández, Tim Kreienkamp, Joan Verdú**

*2) Consider the following multinomial mixture model for describing documents. First, each document d is allocated a latent topic $z_d \in \{0,\ldots,K\}$, where $Pr[z_d = k] = \rho_k$.*
*Then, each word in document d is generated by taking independent draws from $\beta_k$, a probability vector with V elements (V is the number of distinct terms in the corpus). Let $x_{dv}$ be the count in document d of term v.*

*(a) In this model, what are the parameters, what are the latent variables, and what is the observed data?*

Parameters are:
$\rho_k$: Probabilities of a document to belong to a topic (K+1 parameters)
$\beta_k$: Vector (dimensions V) of probabilities of distinc terms, for each topic k. At the end there are (K+1)·V
Adding all, we have (K+1)(V+1) parameters

Latent variables are the topics of each document ($z_d$), assuming that each document belongs just to one topic (simple model topic).

Observed data are documents (d, $n_d$ in total) and words (w, V unique ones) of each document

*(b) Write down the complete data log-likelihood function. Recall that this expresses the log of the joint probability of the latent variables and the observed data.*

The likelihood is a factorization of probabilities of each document. We know the topic of each document is a summation, so the probability of the words of a document is the product of probabilities of each term to the power of counts, weighted by the probability of belonging to the assigned topic (z). That is

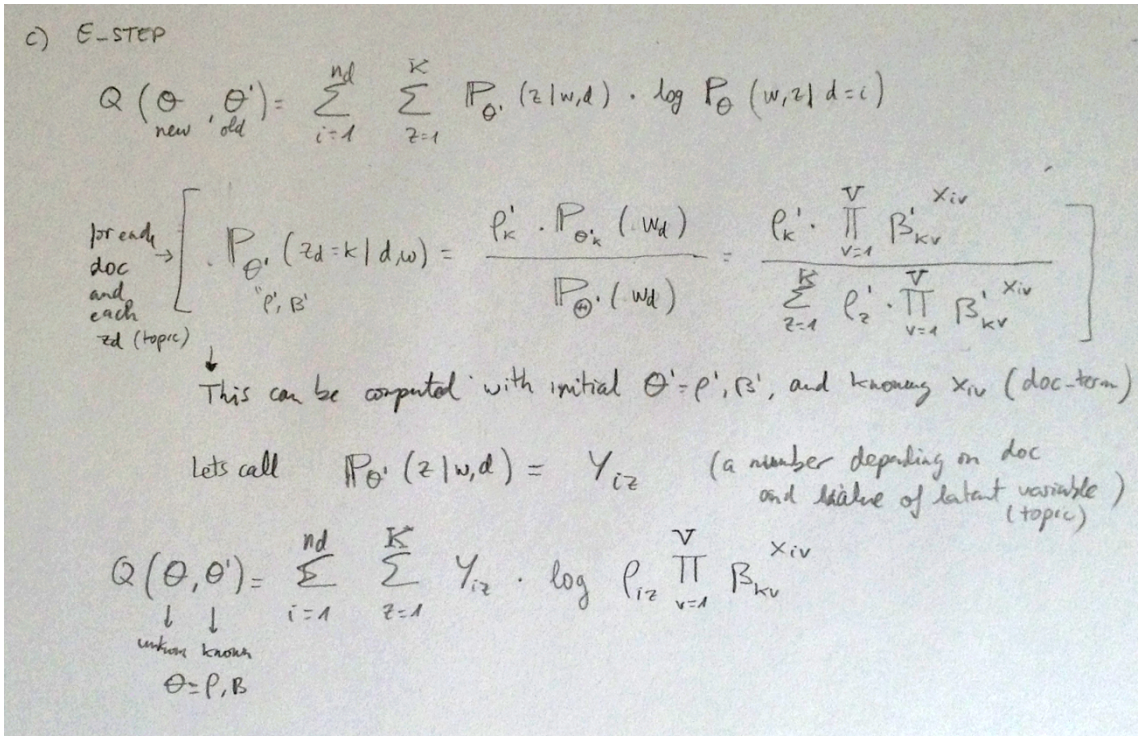
$$L(d, w, z; \rho, \beta) = \prod_{d=1}^{n_d} \rho_{d,z_i} \prod_{j=1}^{V} \beta_{z,j}^{x_{dv}}$$

Then the log likelihood is just (transforming log of product to sums of log):

$$\log L = \sum_{d=1}^{n_d} log\, \rho_{d,z_i} \prod_{j=1}^{V} \beta_{z,j}^{x_{dv}} = \sum_{d=1}^{n_d} log\, \rho_{d,z_i} + \sum_{d=1}^{n_d} \sum_{j=1}^{V} x_{dv} log \beta_{z,j}$$

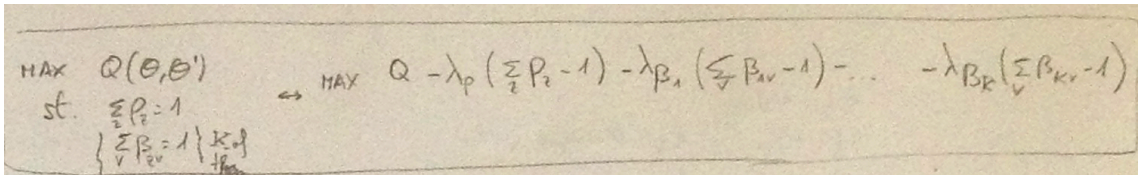
Where $n_d$ are the number of documents; $\rho_{d,z_i}$ is the probability of document i of belonging to topic zi; then $\beta_{k,j}$ is the probability of term j when topic is z; and finally recall $x_{dv}$ is the count of term v in document d (so x is function of w and d).

*(c) Compute the expected value of the above log-likelihood function given fixed values for the parameters. Recall this is the E-step in the EM algorithm. Denote this function Q, and note that it will depend on the parameter values.*



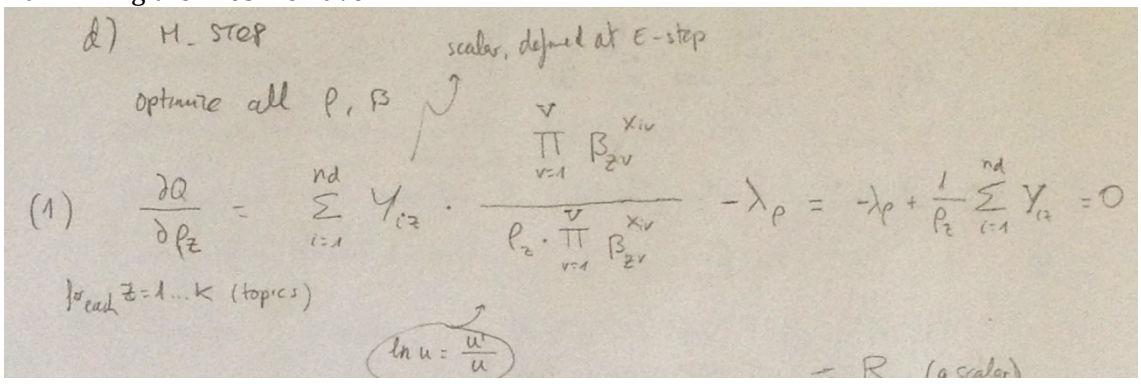

*(d) Maximize Q with respect to the parameter values. Show ALL steps. (Hint: this will involve the setting up of a constrained maximization problem).*

The setup is

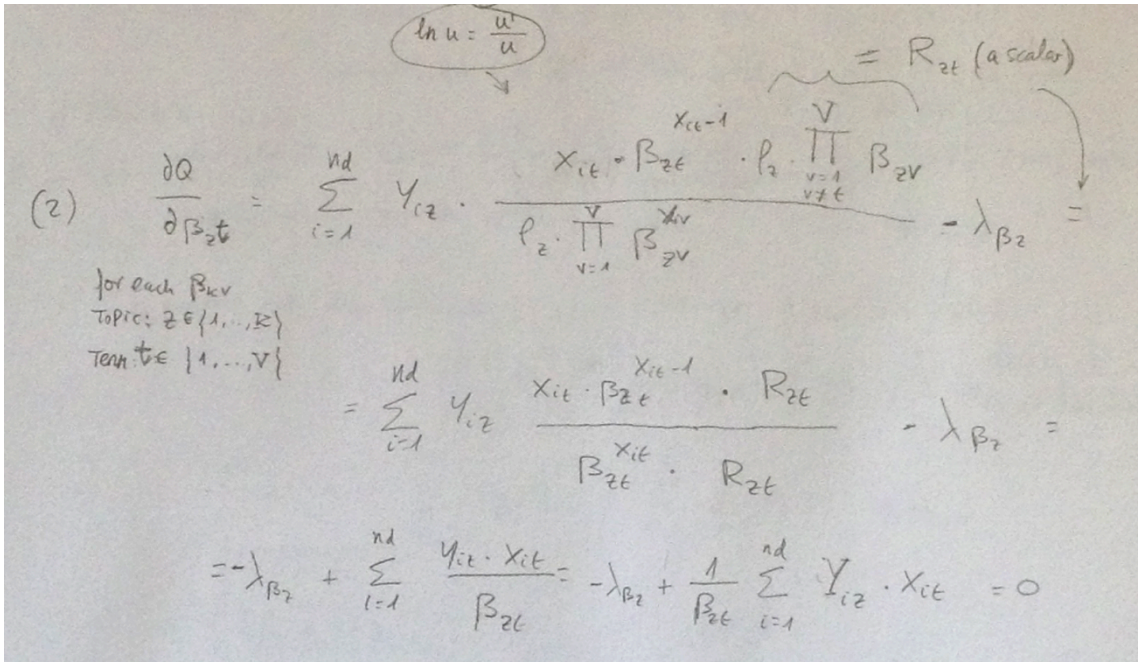



**where we also have to consider all rhos>0 and all betas>0**, since they are probabilities.

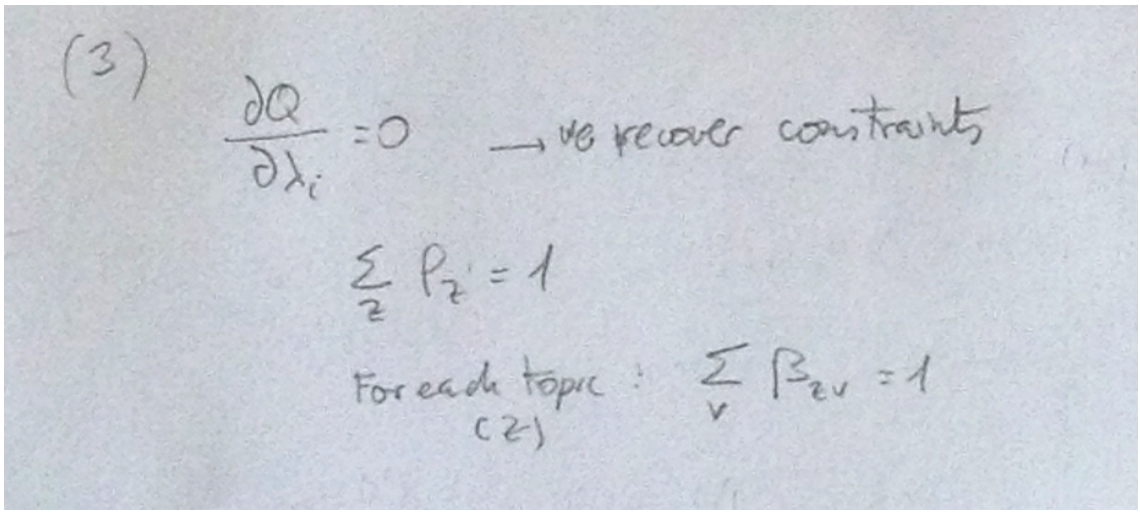
Maximizing the rhos we have:



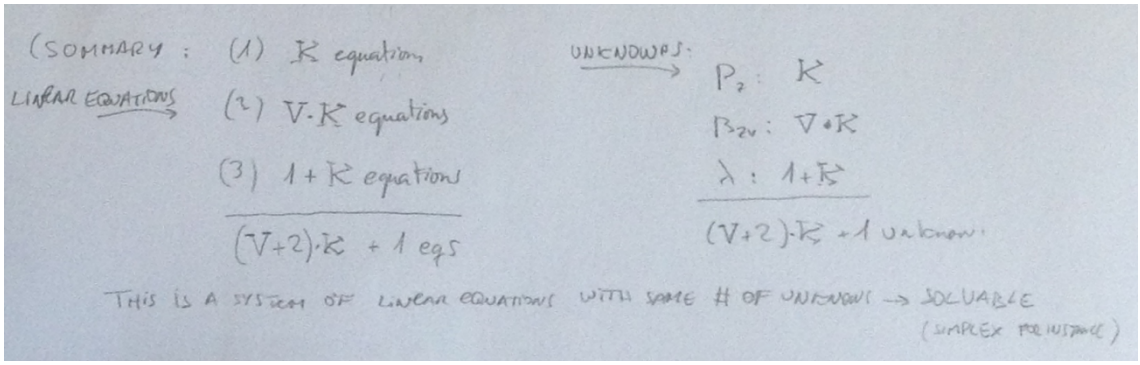

Max betas:




And max lambdas:




So we have a solvable system of linear equations.



*(e) Use your answers to the above to write pseudo-code for implementing the EM algorithm for multinomial mixture models.*

Basically to find a local optimum for the parameters rho and beta we have to do iteratively steps E and M of the EM algorithm until the variation of log L gets under a predefined threshold.

```
# Load/calculate data
Docs= load_docs()
Words= load_words()
Topics=load_topics()
Z= unique(topics)
D= unique(Docs)
W= unique(words)
K=length(Z)
Nd=length(D)
V=length(W)
X = count(docs,words) #matrix of Nd x V
# Initialize parameters and
Rho= random(K)
Beta= matrix(random(K*V),rows=K,cols=V)
Diff= 1
Delta= 0.001
P= matrix(NA,rows=Nd,cols=K)
# Iterate EM algorithm
Max_Iter=1000
Iter=0
While (abs(diff)>delta AND iter<Max_Iter) do
        #E step
        S=0
        For (d in Docs) :
                Denominator= for (z in Z) :
                        S=S+rho(z)*factorial(beta(row=z),Xdv)
                For (z in Z) :
                        P[d,z]=[rho(z)*factorial(beta(row=z),Xdv)]/ denominator
        # we only need this to go for optimization, the other part of Q is fixed
        #M step
        Rho_old=rho
        Beta_old=beta
        #solve linear equations to update   rho, beta values
        Rho,Beta= solve (P,Rho_old,Beta_old,Xdv,)
        #Compute gain
        Diff=1-[LogL(rho,beta,xdv)/LogL(rho_old,beta_old_xdv)]
        Iter += 1
```

Since this is dependent on initial values of parameters, to be closer to the global optimum, we should repeat the procedure several times, and get the optimal solution than yields maximum value of log L.

If we consider previous code as function calculate_EM that yields estimated rho, beta coefficients and optimum logL, we have this pseudocode (considering data includes docs,words and topics):

```
# Repeat EM and get best result
Nsampling=100
RhoF,BetaF,LogLF=calculate_EM(data)
For (I in 1:Nsampling) :
        Rho,beta,logL=calculate_EM(data)
        If logL>logLF :
                rhoF=rho
                betaF=beta
                logLF=logL
```