ST340 Assignment 2

Dylan Dijk (1802183), Kum Mew Lee, Aryan Gupta

27/02/2021

Contents

1

A

In the M-step II we want to maximize $f(\boldsymbol{\mu_{1:k}}) = \sum_{i=1}^{n} \sum_{k=1}^{K} \gamma_{ik} \log(p(\boldsymbol{x_i}|\boldsymbol{\mu_k}))$

And we know that $p(\boldsymbol{x_i}|\boldsymbol{\mu_k}) = \prod_{j=1}^{p} \mu_{kj}^{x_{ij}} (1-\mu_{kj})^{(1-x_{ij})}$

Therefore we want to maximize:

$$f(\boldsymbol{\mu_{1:k}}) = \sum_{i=1}^{n} \sum_{k=1}^{K} \gamma_{ik} \sum_{j=1}^{p} [x_{ij} \log(\mu_{kj}) + (1-x_{ij}) \log(1-\mu_{kj})]$$

Now if we fix $k \in \{1, \ldots, K\}$, we have:

$$f(\boldsymbol{\mu_k}) = \sum_{i=1}^{n} \gamma_{ik} \sum_{j=1}^{p} [x_{ij} \log(\mu_{kj}) + (1-x_{ij}) \log(1-\mu_{kj})]$$

Now taking the partial derivative w.r.t a single μ_j:

$$\frac{\partial f}{\partial \mu_j} = \sum_{i=1}^{n} \frac{\gamma_{ik} x_{ij}}{\mu_j} - \frac{\gamma_{ik}(1-x_{ij})}{1-\mu_j}$$

Setting this equal to zero we get:

$$(1-\mu_j) \sum_{i=1}^{n} \gamma_{ik} x_j = \mu_j \sum_{i=1}^{n} \gamma_{ik} (1-x_{ij})$$

After rearranging we have:

$$\mu_j = \frac{\sum_{i=1}^{n} \gamma_{ik} x_{ij}}{\sum_{i=1}^{n} \gamma_{ik}} \qquad \Longrightarrow \qquad \boldsymbol{\mu_k} = \frac{\sum_{i=1}^{n} \gamma_{ik} \boldsymbol{x_i}}{\sum_{i=1}^{n} \gamma_{ik}}$$

Therefore have shown the unique stationary point is obtained by choosing for each k

$$\boldsymbol{\mu_k} = \frac{\sum_{i=1}^{n} \gamma_{ik} \boldsymbol{x_i}}{\sum_{i=1}^{n} \gamma_{ik}}$$

B

i

Here I have just used the function given to us for Lab 4.

```
xs = documents
out_Q1bi = em_mix_bernoulli(xs, K  = 4)
```

ii

The algorithm does not tell us what each cluster represents and we have to infer this ourselves.
Below I have made a table giving the associated words of the largest 8 values of μ in $\boldsymbol{\mu}_k$ for each k.

1	2	3	4
windows	question	team	fact
help	god	games	world
email	fact	players	case
problem	university	baseball	question
system	problem	hockey	course
software	help	season	government
computer	car	win	problem
program	course	league	state

From this table it seems pretty clear that cluster 1 represents articles related to computers and cluster 3 represents articles to sport ('rec' group).
For clusters 2 and 4 it is less clear. Could say cluster 2 represents the articles from the 'science' group as "university" has a large μ and then cluster 4 represents articles from the 'talk' group.
So in summary my guess would be 1 = `comp.*`, 2 = `sci.*`, 3 = `rec`, 4 = `talk.*`

Below I have looked at all possible permutations of the 4 clusters, and calculated the difference between the γ values of each data row and the actual label from the `newsgroups.onehot` dataset. As the γ value represents the probability that each data point belongs to a certain cluster.

```
perm = permutations(n = 4, r = 4, v = 1:4)
gamma_minus_label = vector(length = 24)
for(i in 1:24){
  gamma_minus_label[i] = sum(abs(out_Q1bi$gammas[,perm[i,]] - newsgroups.onehot))
}
```

Comparing my guess for the labeling of the clusters with the order of the `groupnames` vector,

```
groupnames
```

```
> [1] "comp.*" "rec.*"  "sci.*"  "talk.*"
```

the permutation that should give the minimum value should be `1 3 4 2`.

```
algorithm_labels = perm[which.min(gamma_minus_label),]
algorithm_labels
```

```
> [1] 1 3 4 2
```

And the average value of the difference between the actual lables and the gammas for this minimum case is:

```
min(gamma_minus_label)/nrow(documents)
```

```
> [1] 0.8210617
```

iii

So in the function given to us for Lab 4 it uses `.2 + .6*xs[sample(n,K),]` to select the starting μ_k for each k.
This randomly samples K rows from the dataset we are inputting into the algorithm and assigns μ equal to 0.2 if an element of the row is zero and 0.8 if the element is one. Important to note that we shouldn't set a

starting μ equal to zero or one. *The code for how I ran the following 3 cases is in the markdown file*

(A) I am going to rerun the algorithm but now use `.4 + .2*xs[sample(n,K),]`, this now assigns μ equal to 0.4 if an element of the row is zero and 0.6 if the element is one.

Looking again at the value we get for the minimum difference we get between the gammas and the actual labels (divided by number of rows of data)

```
> [1] 0.787949
```

The original function selects equal starting weights to each cluster written as: `rep(log(1/K),K)`. Here the function is using logs to keep the numbers stable.

(B) I will now run the algorithm with starting weights equal to the proportion of each type of article.

Using the same measure:

```
> [1] 0.7703485
```

(C) I will now run the algorithm with starting weights equal to the proportion of each type of article and starting $\mu's$ equal to the proportion of times a word appeared and adding 0.01 so that we have no zero values.

Using the same measure:

```
> [1] 0.8184101
```

Below I have made the same tables as before. The first row has the Table for case A on the left and then for B on the right. Then at the bottom is the table for case C.

1	2	3	4	1	2	3	4
windows	fact	team	question	fact	question	team	windows
email	world	games	god	world	god	games	email
help	case	players	fact	case	fact	players	help
problem	course	baseball	problem	course	problem	baseball	problem
system	question	hockey	university	government	course	hockey	system
software	government	season	course	question	university	season	software
computer	problem	win	christian	problem	christian	win	computer
program	state	league	help	state	world	league	program

1	2	3	4
question	fact	windows	team
god	world	email	games
fact	case	help	players
university	question	problem	baseball
problem	course	system	hockey
help	government	software	season
car	problem	computer	win
course	state	program	league

So in conclusion we can see from the tables that the clusters that seem to represent postings about computers and sport are very stable, but the other two less so.
And looking at the measure I used to determine 'accuracy', the output didn't seem to be too sensitive to changes in starting values. It would of been best if I had ran each case a couple of times, as we do get different output anyways due to the randomness within the functions I used e.g. the functions use `sample`.