Written languages recognition for song lyrics

Unsupervised learning

LE Son Tung, NGUYEN Thi Ha Giang

November 2020

The objective of this project is to automatically classify song lyrics into English and French. For knowing which song is English or French, we will use Naive Bayes and Markovian classifiers. We also study how to differentiate the language between phrases of a random text using Viterbi and Baum-Welch algorithms.

1 Text base

1.1 Construction and treatment of texts

We first construct a raw texts file including 60 files of different song lyrics, including 30 English song (files' names begin with *EN*) and 30 French song (files' names begin with *FR*).

```
## [1] "EN_All I want.txt"             "EN_As long as you love me.txt"
## [3] "EN_Bad day.txt"                "EN_Beautiful in white.txt"
```

```
## [1] "FR_Aline.txt"            "FR_Andalouse.txt"
## [3] "FR_Balance ton quoi.txt" "FR_Beau malheur.txt"
```

We wrote a function named `normalise_text` to clean texts by

- changing all letters to lower case,
- replacing characters with accents in French by the equivalent Latin alphabets (for example replacing $\grave{u}$, $\hat{u}$ and $\ddot{u}$ by u),
- removing all other symbols and numbers.

After loading texts into `R`, we applied this function to obtain the normalized texts and labeled each text x_i with $y_i = 1$ if it is in English and $y_i = -1$ if it is in French. We obtain a vector of text's label y with length of 60.

1.2 Frequency of characters

We write a function `unigram` to calculate the log-frequency (add-1 smoothing) of each symbol j for each text x_i.

We then obtain the matrix `X_unigram` of size 60x27 (There are 26 letters and the space).

```
##            a           b           c          d          e
## 1 0.04236640 0.019893541 0.010761057 0.04680099 0.05998800
## 2 0.06054968 0.005082603 0.008877674 0.02763995 0.09124867
## 3 0.07819557 0.013368406 0.011634803 0.05686580 0.06895386
```

1.3 Histogram of log-frequency

We plot the histogram of each symbols in each class and obtain overlap graphs, with red representing EN and blue is FR. These graphs shows obvious differences in the distribution of log-frequency between English and

French.

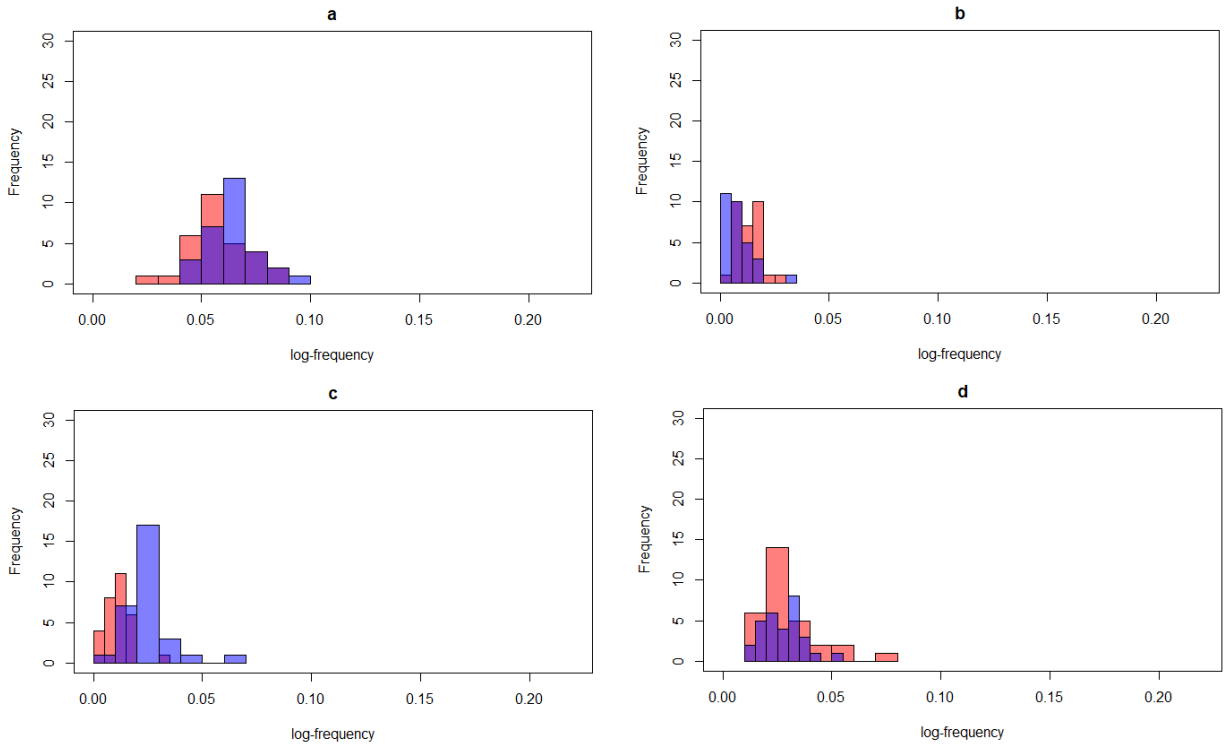

Figure 1: Histogram of some characters

2 Naive Bayes classification

Assume that

$$f_k(x|\theta_k) = N(x; \mu_k, \sigma_k^2 I_p)$$

and x are mutually independent conditional on θ_k.

2.1 Parameters estimation

First, we write the function `bayes_naif` to return the mean and variance of each symbol for each class.

```
para=bayes_naif(X_unigram,y)
head(para$m) #mean
```

```
##            1          -1
## a 0.05769055 0.064516342
## b 0.01309183 0.007768996
## c 0.01171411 0.024957164
## d 0.02996906 0.027164627
## e 0.08994582 0.125586941
## f 0.01504497 0.007973830
```

```
head(para$sigma) #variance
```

```
##               1           -1
## a 1.995363e-04 1.150970e-04
```

```
## b 3.495114e-05 3.810515e-05
## c 3.761471e-05 1.153860e-04
## d 1.988267e-04 8.331062e-05
## e 2.115553e-04 2.512199e-04
## f 2.872392e-05 2.569677e-05
```

2.2 Classification

Then we write another function `prediction_bayes` for our classification, using the parameter obtained above. By applying this function, we will have the predicted class for each observation.

```
y_pred_bayes_naif = prediction_bayes(para,X_unigram)
```

```
y_pred_bayes_naif$prediction
```

```
##  [1]  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1
## [26]  1  1  1  1  1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
## [51] -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
```

2.3 Performance validation

For cross validation, we will try two ways to split the train-test sets: stratified and unstratified k-fold, with $k = 5$. We compare between the predicted values and the observing values

```
pred_stratified_bayes_naif == y
```

```
##  [1] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
## [16] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
## [31] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
## [46] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
```

```
pred_unstratified_bayes_naif == y
```

```
##  [1] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
## [16] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
## [31] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
## [46] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
```

The Bayes Naives Classificatioin works well in this case.

3 Markovian classification

3.1 Estimation of parameters

For each lyrics, we construct a bigram matrix 27x27 `X_bigram`, where X_{ij} is the frequency of transition from character i to character j (the character space " " in also included in this matrix).

Suppose that

$$f_k(x|\theta_k) = MC(x; \pi; A)$$

Hence, we have to estimate two parameters: the probability of initial state π and matrix of transition A for each class.

In this project, we set the parameter π equals to the probability of transition from the space " " to a symbol, which is the last row of the matrix of transition A. (This is justified, as the first character of a text is always a first character of a word, which is usually after the character space " ".)

Notice that, when we construct the bigram matrix, we added a character space " " at the beginning of each text. In this case the density of the Markov chain for a text calculated from bigram matrix is given as:

$$f_k(x|\theta_k) = \prod_{i,j} (A_{ij}^{(k)})^{X_{ij}}$$

From this we have the log-likelihood of all texts:

$$L = \sum_{m=1}^{n} \sum_{i,j} \sum_{k} X_{m,ij} \times \log(A_{ij}^{(k)}) \times 1_{(y_m=k)}$$

With $\sum_j A_{ij}^{(k)} = 1$, we get MLE result:

$$A_{ij}^{(k)} = \frac{\sum_{m=1}^{n} X_{m,ij} \times 1_{(y_m=k)}}{\sum_{m=1}^{n} \sum_j X_{m,ij} \times 1_{(y_m=k)}}$$

3.2 Classification

For each text, we calculate the posterior density for k classes and classify basing on them. But for Markovian classification, we calculate logarit of posterior density, because values of density in this case is very small maybe cause crashing in R.

```
y_pred_markovien = prediction_markovien(para,X_bigram)
head(y_pred_markovien$log_posterior)
```

```
##              1        -1
## [1,] -1427.570 -1943.676
## [2,] -3052.215 -4401.218
## [3,] -3618.514 -4986.036
## [4,] -2733.356 -3885.539
## [5,] -2127.459 -2901.073
## [6,] -1076.824 -1564.385
```

```
y_pred_markovien$prediction
```

```
##  [1]  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1
## [26]  1  1  1  1  1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
## [51] -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
```

3.3 Performance evaluation

Same with the Naive Bayes Classification, we try cross-validation in two way and obtain the following results

```
pred_stratified_markovien == y
```

```
##  [1] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
## [16] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
## [31] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
## [46] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
```

```
pred_unstratified_markovien == y
```

```
##  [1] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
## [16] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
## [31] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
## [46] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
```

4 Viterbi Algorithm

4.1 Construction of random text

We construct the text combined by 30 random phrases by the way below:

- We create the matrix of transition *EN-FR*: $A = \begin{pmatrix} 0.8 & 0.2 \\ 0.2 & 0.8 \end{pmatrix}$ and the initial probability $\pi = (0.5, 0.5)$.

```
#Transition matrix between EN-FR
A_text = matrix(c(0.8,0.2,0.2,0.8),2,2)
#Initial probability
pi_text = c(1/2,1/2)
```

- Simulate a chain `y_viterbi` based on those parameters.
- For each state in `y_viterbi`, we select randomly a English text if this state is 1 and a French text if this state is -1. And we take the random phrase from this text.

4.2 Viterbi for detection passages in French and English of the fabricated text

For implementing Viterbi algorithm, we set the emission law of distribution for each phrase to be the Markov chain in the previous part. The density of each phrase is calculted the same way as in the Markovian classification.

```
Viterbi(Pi,A,B,text_viterbi,c(1,-1))==y_viterbi
```

```
##  [1] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
## [16] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
```

5 Baum-Welch Algorithm

5.1 Baum-Welch Algorithm for two hidden states

We have

$$P(x_{1:n}z_{1:n}|\theta) = \Big[p(z_1)\prod_{t=2}^{n} p(z_t|z_{t-1})\Big]\Big[\prod_{t=1}^{n} p(x_t|z_t)\Big].$$

Hence,

$$\log P(x_{1:n}z_{1:n}|\theta) = \sum_k 1_{(z_1=k)} \log \pi_k + \sum_{t=2}^{n}\sum_{j,k} 1_{(z_{t-1}=j, z_t=k)} \log A_{j,k} + \sum_{t=1}^{n}\sum_k 1(z_t=k)\sum_{m,l} X_{m,l}^{(t)} \log B_{m,l}^{(k)}.$$

Since $Q(\theta, \theta^{\text{old}}) = E_{\theta^{\text{old}}, x_{1:n}}\big[\log P(x_{1:n}z_{1:n}|\theta)\big]$, we obtain

$$Q(\theta, \theta^{\text{old}}) = \sum_k P(z_1 = k|\theta^{\text{old}}; x_{1:n}) \log \pi_k + \sum_{t=2}^{n}\sum_{j,k} P(z_{t-1}=j, z_t=k|\theta^{\text{old}}; x_{1:n}) \log A_{j,k} + \sum_{t=1}^{n}\sum_k\sum_{m,l} P(z_{t-1} = j, z_t = k|\theta^{\text{old}}; x_{1:n}) X_{m,l}^{(t)} \log B_{m,l}^{(k)}.$$

In EM method, denote $P(z_t = k|\theta; x_{1:n}) = \gamma_t(k)$ and $P(z_{t-1} = i, z_t = j|\theta; x_{1:n}) = \epsilon_{t-1,t}(i,j)$. Then, for each iteration, we update

$$\pi_k = \gamma_1(k)$$

$$A_{ij} = \frac{\sum_{t=2}^{n} \epsilon_{t-1,t}(i,j)}{\sum_k \sum_{t=2}^{n} \epsilon_{t-1,t}(i,j)}$$

$$B_{m,l}^{k} = \frac{\sum_{t=1}^{n} \gamma_t(k) \times X_{m,l}^{t}}{\sum_l \sum_{t=1}^{n} \gamma_t(k) \times X_{m,l}^{t}}$$

5.2 Using Viterbi parameters

Here are the likelihood for each iteration

```
para=markovien(EM_text(text_original, 1, 1), y)
Pi = pi_text
A = A_text
B = para$A

X = EM_text(text_viterbi, 1/100, 1)
pred = EM(X, Pi, A, B, c(1,-1))$prediction
```

```
## [1] -3088.484
## [1] -2839.234
## [1] -2839.602
## [1] -2839.602
```

and comparison of EM clustering and true values

```
pred == y_viterbi
```

```
##  [1] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
## [16] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
```

5.3 Using random parameters

Similarly, we have the likelihood for each iteration and the comparison

```
para=markovien(EM_text(text_original, 1, 1), y)
Pi = pi_text
A = A_text
B = para$A

B[[1]][,] =runif(27*27)
B[[2]][,] =runif(27*27)
B[[1]]=t(apply(B[[1]],1,function(x)x/sum(x)))
B[[2]]=t(apply(B[[2]],1,function(x)x/sum(x)))

X = EM_text(text_viterbi, 1/100, 1)
pred = EM(X, Pi, A, B, c(1,-1))$prediction
```

```
## [1] -4277.291
## [1] -2908.141
## [1] -2899.904
## [1] -2897.906
## [1] -2896.571
## [1] -2896.569
```

```
pred == y_viterbi
```

```
##  [1]  TRUE  TRUE FALSE  TRUE  TRUE FALSE FALSE FALSE FALSE  TRUE  TRUE  TRUE
## [13]  TRUE  TRUE  TRUE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [25]  TRUE FALSE  TRUE FALSE  TRUE  TRUE
```

5.4 Comments

With the used parameters for Viterbi, EM gives us a good clustering. With random parameters, the result of EM method is not good because: (1) length of phrases constituting the text are quite short and (2) the EM

method converges to a local optimal.

If we set threshold of changes in likelihood for early stopping too small, EM will converge to a point that is over-fitting for a short random text and it cannot well detect the changes between *EN-FR*, even we start with estimated parameters in Markovian classification.

Another problem is that the computation power of `R` might not be sufficient for EM method. When some values too close to 0, `R` recognizes them as 0 and doesn't execute the division. To deal with this problem, we try re-scaling the conditional likelihood of future evidence matrix β in EM function. However, this cannot truly solve the problem. Another easy way is reducing scale of matrix bigram (dividing all transition's frequencies by scale value: 5, 10, 100...). This is also not an absolute solution, but it still gives us desirable results.

6 References

Murphy, Kevin P., *Machine Learning: A Probabilistic Perspective*, 2012