LDA Intuition

Isabelle Valette

October 22, 2018

How LDA algorithm works?

Let say that we have the following corpus of 3 documents with 2 topics, one about animal and one about pet.

I eat fish and vegetables.
Fish are pets.
My kitten eats fish.

Documents about food and pets

0 - Data Preprocessing

We start with a fictive corpus of 3 documents where we only look at the lemmas we will need in topic modelling from the verbs and the noun. In Italic is the origin sentence the lemma is extracted from.

```
library(dplyr)
library(ggplot2)

rawdocs <- c('eat fish vegetable',   # I eat fish and vegetables
             'fish pet',             # Fish are pets
             'kitten eat fish')       # Kitten east fish


# generate a list of lemmas
docs <- strsplit(rawdocs, split=' ', perl=T)
docs
```

```
## [[1]]
## [1] "eat"       "fish"      "vegetable"
##
## [[2]]
## [1] "fish" "pet"
##
## [[3]]
## [1] "kitten" "eat"    "fish"
```

We can see above a list of lemmas per document and below we plot the vocabulary list.

```
# create a vocabulary list
vocab <- unique(unlist(docs))
vocab
```

```
## [1] "eat"       "fish"      "vegetable" "pet"       "kitten"
```

We are now ready for the parametrisation steps.

1 - Parametrisation

First let's start by setting the parameters for our model: The number of topics, alpha, eta, the number of iterations and the seed for reproducibility.

```
# number of topics
K <- 2
# dirichlet prior.
alpha <- 0.001
# dirichlet prior.
eta <- 0.05
# iterations for collapsed gibbs sampling
iterations <- 20
# set the seed
set.seed(12345)
# replace words with wordIDs
for(i in 1:length(docs)) docs[[i]] <- match(docs[[i]], vocab)
# Visualize the documents with word ids
docs
```

```
## [[1]]
## [1] 1 2 3
##
## [[2]]
## [1] 2 4
##
## [[3]]
## [1] 5 1 2
```

Above, we can see a list of documents where the lemmas have been replaced with their lemma ids.

2 - Initialisation

```
# initialize en empty word-topic count matrix.
wt <- matrix(0, K, length(vocab))
# initialize an empty topic-allocation list ta
ta <- sapply(docs, function(x) rep(0, length(x)))
```

Now let's look at the word topic count matrix. The columns are the lemmas and the rows are the topics.

```
wt
```

```
##      [,1] [,2] [,3] [,4] [,5]
## [1,]    0    0    0    0    0
## [2,]    0    0    0    0    0
```

For each word in each document, we allocated a topic, let's look at this per document, per word topic assignment. In this list, the elements of the lists are the documents. Each documents has a length that corresponds to its number of lemmas. The value of each lemma is the topic assignment.

```
ta
```

```
## [[1]]
## [1] 0 0 0
##
## [[2]]
## [1] 0 0
##
## [[3]]
## [1] 0 0 0
```

3 - Topic Allocation

```
for(d in 1:length(docs)){
    for(w in 1:length(docs[[d]])){
      # randomly assign topic to token w
      ta[[d]][w] <- sample(1:K, 1)
      # topic index
      ti <- ta[[d]][w]
      # wordID for token w
      wi <- docs[[d]][w]
      # update word-topic count matrix
      wt[ti,wi] <- wt[ti,wi]+1
  }
}
ta
```

```
## [[1]]
## [1] 2 2 2
##
## [[2]]
## [1] 2 1
##
## [[3]]
## [1] 1 1 2
```

```
wt
```

```
##      [,1] [,2] [,3] [,4] [,5]
## [1,]    1    0    0    1    1
## [2,]    1    3    1    0    0
```

Firstly we randomly assigned topics to each word in each document. Then we created a word-topic count matrix wt. This wt matrix will enable us to compute the probability of word w under topic t.

4 - Other count matrixes

```
dt <- matrix(0, length(docs), K)
for(d in 1:length(docs)){
    for(t in 1:K){
    dt[d,t] <- sum(ta[[d]]==t)
  }
}
dt
```

```
##      [,1] [,2]
## [1,]    0    3
## [2,]    1    1
## [3,]    2    1
```

Now we have generated a document-topic count matrix where the counts correspond to the number of lemmas assigned to each topic for each document. This will enable use to compute the probablity of topic t in document d.

5 - The magic of the Gibbs sampler

```
for(i in 1:iterations){
  for(d in 1:length(docs)){
    for(w in 1:length(docs[[d]])){

      t0 <- ta[[d]][w] # initial topic assignment to token w
      wid <- docs[[d]][w] # wordID of token w

      dt[d,t0] <- dt[d,t0]-1
      wt[t0,wid] <- wt[t0,wid]-1

      ## COLLAPSED GIBBS SAMPLING MAGIC
      denom_a <- sum(dt[d,]) + K * alpha
      denom_b <- rowSums(wt) + length(vocab) * eta
      p_z <- (wt[,wid] + eta) / denom_b * (dt[d,] + alpha) / denom_a
      t1 <- sample(1:K, 1, prob=p_z/sum(p_z))

      ta[[d]][w] <- t1
      dt[d,t1] <- dt[d,t1]+1
      wt[t1,wid] <- wt[t1,wid]+1

      if(t0!=t1) print(paste0('doc:', d, ' token:' ,w, ' topic:',t0,'=>',t1))
    }
  }
}
```

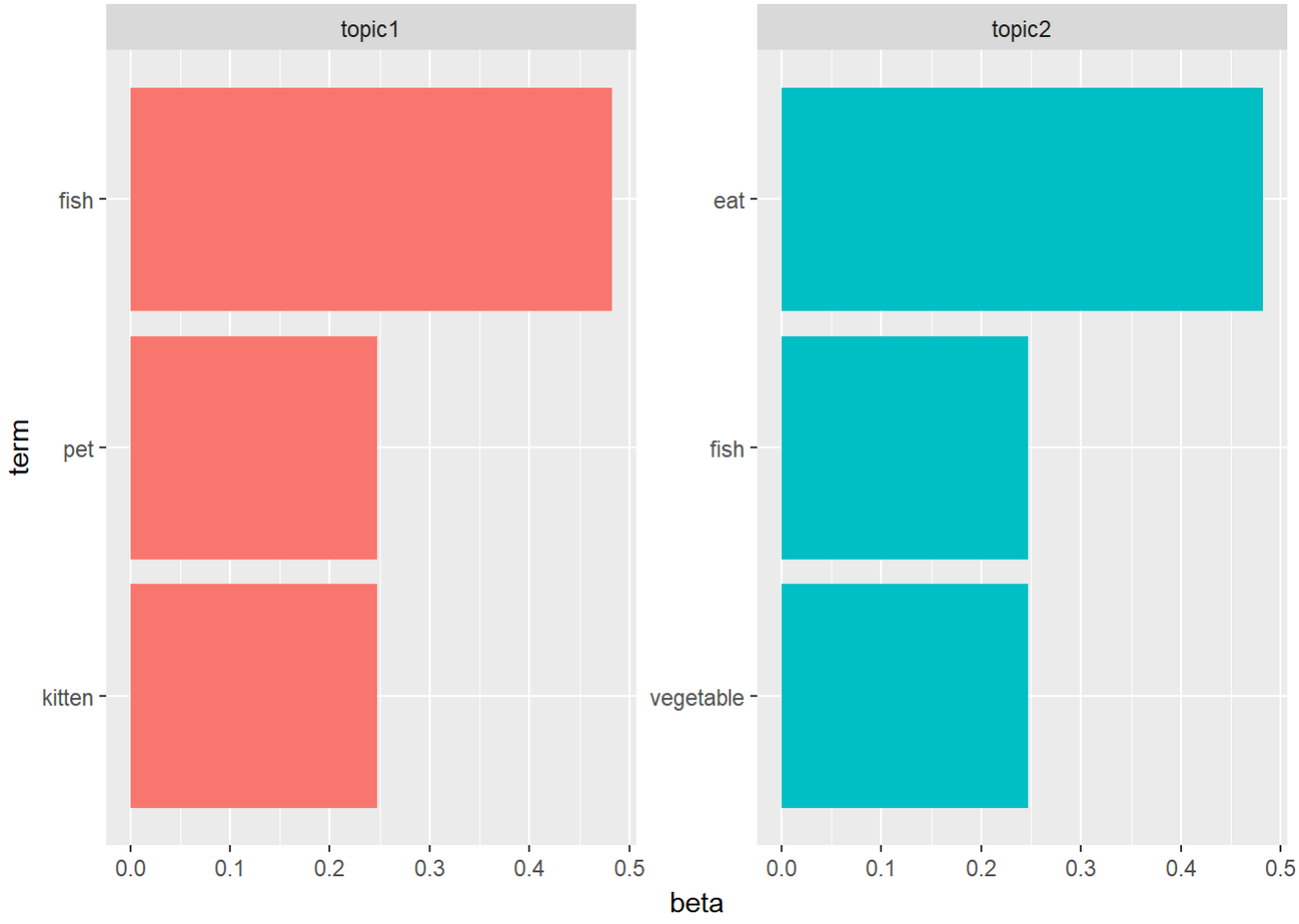
```
## [1] "doc:2 token:1 topic:2=>1"
## [1] "doc:3 token:2 topic:1=>2"
## [1] "doc:3 token:3 topic:2=>1"
## [1] "doc:3 token:2 topic:2=>1"
## [1] "doc:3 token:2 topic:1=>2"
```

6- Visualizing the output of the LDA model

```
# topic probabilities per word
b <- t((wt + eta) / (rowSums(wt+eta)))
beta <- data.frame(term = rep(vocab, 2),
                   topic = c(rep("topic1", 5), rep("topic2", 5)),
                   beta = c(b[,1], b[,2]))

beta %>%
  filter(beta > 0.07) %>%
  mutate(term = reorder(term, beta)) %>%
  ggplot(aes(term, beta, fill = factor(topic))) +
  geom_col(show.legend = FALSE) +
  facet_wrap(~ topic, scales = "free") +
  coord_flip()
```

```
## Warning: package 'bindrcpp' was built under R version 3.4.4
```



```
# compute topic probabilities per document
t <- (dt+alpha) / rowSums(dt+alpha)
theta <- data_frame(document = rep(paste0("doc", 1:3), 2),
                    topic = c(rep("topic1", 3), rep("topic2", 3)),
                    theta = c(t[,1], t[,2]))
theta %>% ggplot(aes(document, theta, fill = factor(topic))) +
  geom_col(show.legend = TRUE)
```