t_sne

Luis Angel Rodriguez Garcia

3/5/2022

Introduction

```
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

We are going to play with the Iris dataset:

```
X <- iris %>% dplyr::select(-Species) %>% as.matrix()
```

Euclidean distance

The way to calculate the Euclidean distance:

$$||\mathbf{x}_i - \mathbf{x}_j||^2 = \mathbf{x}_i^2 + \mathbf{x}_j^2 - 2\mathbf{x}_i\mathbf{x}_j$$

$$\mathbf{x}_k^2 = \mathbf{x}_k'\mathbf{x}_k = x_{k1}^2 + ... + x_{kp}^2$$

```
x_diff <- function(X) {
  sum_x <- apply(X^2, MARGIN=1, FUN=sum)
  sum_x_m <- t(matrix(replicate(150, sum_x), byrow=T, nrow=150))
  cross_times_minus_2 <- -2 * (X %*% t(X))
  D <- t(cross_times_minus_2 + sum_x_m) + sum_x_m
  D <- round(D, digits=4)
}
```

Perplexity

$$Perp_i = 2^{H_i}$$

Where H_i is the Shannon entropy in the point x_i of the conditional probability:

$$
\begin{aligned}
H_i &= -\sum_{j \neq i} p_{j|i} \log(p_{j|i}) \\
&= -\sum_{j \neq i} p_{j|i} \log(p_{j,i}/p_i) \\
&= -\sum_{j \neq i} p_{j|i} (\log(p_{j,i}) - \log(p_i)) \\
&= -\sum_{j \neq i} p_{j|i} (\log(e^{-||x_i - x_j||^2/2\sigma^2}) - \log(\sum_{k \neq i} e^{-||x_i - x_j||^2/2\sigma^2})) \\
&= -\sum_{j \neq i} p_{j|i} ((-||x_i - x_j||^2/2\sigma^2) - \log(\sum_{k \neq i} e^{-||x_i - x_j||^2/2\sigma^2})) \\
&= \sum_{j \neq i} p_{j|i} (\log(S_i) + ||x_i - x_j||^2 \frac{1}{2\sigma^2}) \\
&= \log(S_i) \sum_{j \neq i} p_{j|i} + \frac{1}{2\sigma^2} \sum_{j \neq i} p_{j|i} ||x_i - x_j||^2) \\
&= \log(S_i) + \frac{1}{2\sigma^2} \sum_{j \neq i} p_{j|i} ||x_i - x_j||^2
\end{aligned}
\tag{1}
$$

Where $S_i = \sum_{k \neq i} e^{-||x_i - x_j||^2/2\sigma^2}$ and $\sum_{j \neq i} p_{j|i} = 1$

```
entropy_var <- function(D_i, var=1/2) {
  var_exp = 1/(2*var)
  P_i <- exp(-D_i * var_exp)
  sum_p_i <- sum(P_i)
  H_i <- log(sum_p_i) + (var_exp * sum(D_i * P_i) /sum_p_i)
  P_i <- P_i / sum_p_i
  return(list(entropy=H_i, probs=P_i))
}
```

The goal is to adjust the variability so that the perplexity at each point is the same. We are going to perform a binary search to get the probabilities in such a way that the conditional Gaussian has the same perplexity.

```
binary_optimization <- function(h_diff, var, i, var_min, var_max) {
  if(h_diff > 0) {
    var_min = var[i]
    if(var_max == -Inf || var_max == Inf) {
      var[i] <- var[i] * 2
    } else {
      var[i] <- (var[i] + var_max) / 2
    }
  } else {
    var_max = var[i]
    if(var_min == -Inf || var_min == Inf) {
      var[i] <- var[i] / 2
    } else {
      var[i] <- (var[i] + var_min) / 2
    }
  }
  return(list(varianze=var, min=var_min, max=var_max))
}
```

```r
binary_search_opt <- function(X=matrix(), tolerance=1e-5, perplexity=30) {
  n <- nrow(X)
  p <- ncol(X)

  D <- x_diff(X)

  P <- matrix(0, nrow=150, ncol=150)
  var <- rep(1/2, n)
  log_perp <- log(perplexity)

  for(i in seq_len(n)) {
    var_min <- -Inf
    var_max <- Inf

    index <- c(seq(1,i-1),seq(i+1,n))
    if (i == 1) {
      index <- 2:n
    } else if (i == n) {
      index <- 1:(n-1)
    }

    D_i <- D[i,index]
    res <- entropy_var(D_i, var[i])

    h_star <- res$entropy
    prob_star <- res$probs

    h_diff <- h_star - log_perp
    tries <- 0
    while(abs(h_diff) > tolerance && tries < 50) {
      var_opt <- binary_optimization(h_diff, var, i, var_min, var_max)
      var <- var_opt$varianze
      var_min <- var_opt$min
      var_max <- var_opt$max

      res_loop <- entropy_var(D_i, var[i])
      h_star <- res_loop$entropy
      prob_star <- res_loop$probs

      h_diff <- h_star - log_perp

      if(i==61) {
        print(tries)
        print(sprintf("Index row: %i",i))
        print(sprintf("Var: %f",var[i]))
        print(sprintf("H: %f",h_star))
        print(sprintf("Hdiff: %f",h_diff))
      }

      tries <- tries + 1
    }
    P[i, index] <- prob_star
  }
  print(var)
  print(sprintf("Mean value of sigma: %01.2f", mean(var)))
  return(P)
}
```