Ciencia de Datos
Tarea 2

Para entregar el 2 de abril de 2018

1. Considera los datos del archivo `gene_expression_2classes.csv`, que contiene la expresión genética de 1000 genes en 40 muestras de tejido, de las cuales las primeras 20 son de pacientes sanos y las 20 restantes de pacientes enfermos de alguna enfermedad.

   a) Compara los métodos de clustering jerárquico y los basados en $k-$means en los datos. ¿Puedes identificar los dos grupos existentes? Prueba usando medidas de disimilaridad euclidena y correlación, así como diferentes tipos de enlace. ¿Cómo cambian los resultados realizando PCA previamente a los datos? ¿Cuántos componentes sugieres usar?

   Realiza un reporte breve de todos estos aspectos, ilustrando de forma adecuada tus hallazgos y conclusiones.

   b) Uno de los médicos quisiera saber qué genes muestran mayores diferencias entre los dos grupos de tejido. ¿Cómo lo verificarias? Implementa tu idea.

2. Implementa kernel $k-$means basandote en el artículo de *Inderjit Dhillon, Yuqiang Guan and Brian Kulis. A Unified view of Kernel k-means, Spectral Clustering and Graph Cuts. UTCS Technical Report, 2005.*

   Escoge (o genera) algunos conjuntos de datos adecuados para verificar la eficiencia y ventajas del método. Comparalo con otros métodos para mostrar en qué casos es mejor su desempeño.

3. Los datos en el archivo `data_fruits_tarea.zip` contienen imágenes preprocesadas de 100×100 pixeles, que corresponden a diferentes tipos de frutas, tomadas en diferentes orientaciones y con diferentes características de forma y maduración. Supón que a una cadena de supermercados le interesa implementar un método automático de reconocimiento del tipo de fruta (y posiblemente su nivel de maduración) a través de las imágenes en color.

   a) Obtén una representación de las imágenes en el espacio `RGB` usando la mediana como medida de resumen de los valores en cada canal ¿Puedes identificar patrones interesantes en esta representación?

   b) Realiza PCA y Kernel PCA con un kernel Gaussiano en los datos que obtuviste. ¿Puedes identificar grupos interesantes o informativos de las imágenes en los primeros componentes principales?

   c) Aplica $K-$means y Kernel $K-$means. Verifica si puedes identificar los diferentes grupos de frutas.

d) Repite los incisos anteriores usando el espacio `HSV` (Hue, Saturation, Value). Para incluir más información sobre cada dimensión, utiliza la información de los tres cuartiles centrales en cada una de ellas, de forma que tengas una representación en un espacio de tamaño $d = 9$. ¿Notas alguna mejoría?

4. Este ejercicio es sobre análisis de sentimientos. En el archivo `movie_reviews.zip` se encuentran las críticas de 500 películas tomadas de `http://www.imdb.com`[1]. En `movie_reviews.csv` se encuentra la información de cada película, incluyendo, entre otra cosas, su nombre, el nombre del archivo de texto donde se encuentra su crítica y el *sentimiento* relacionado con la misma: P (positiva) o N (negativa).

   a) Realiza PCA en la representación de los textos obtenida con *bag of words* usando los n términos más frecuentes (decide el valor de n). Realiza el preproceso que consideres necesario en los textos. ¿Puedes identificar las críticas positivas y negativas en los componentes principales? Si la respuesta es no, explica las razones.

   b) Redefine los datos de entrada de *bag of words* uniendo k documentos de cada categoría, por ejemplo $k = 5$. Repite el inciso anterior. ¿Qué diferencias notaste? Describe tus hallazgos.

5. Este ejercicio es sobre análisis de similaridad en textos. El archivo `train_stock.csv` contiene texto descriptivo de acciones en la bolsa. El texto contenido en las columnas `description_x` y `description_y`, se considera que se refieren a la misma acción si ambos tickers `ticker_x` y `ticker_y` coinciden, aunque las descripciones no sean iguales.

   a) Realiza Kernel PCA con string kernels para analizar la similaridad de los textos `description_x` y `description_y`. Describe los patrones o grupos significativos que logres identificar en los primeros componentes principales. Prueba con diferentes tipos de string kernels y reporta cuál te proporciona el "mejor" resultado.

   b) Considera los textos de `description_x` como tu conjunto de entrenamiento y realiza Kernel PCA como en el inciso anterior. Selecciona algunos datos de la columna `description_y` como datos de "prueba" y verifica qué texto del conjunto de entrenamiento es el más similar a cada uno usando la distancia mínima en el espacio de componentes principales. ¿Coinciden con los del archivo original?

[1] La base de datos fue tomada de `http://www.cs.cornell.edu/people/pabo/movie-review-data/`, donde puedes ver varios detalles de los datos y su preproceso, sin embargo, fueron simplificados y adecuados por mi para que puedas usarlos más facilmente (espero) en este ejercicio.

Ayuda de software e implementación.

Para la implementación de Bag of Words, puedes usar la librería `tm` (text mining) de R. Leé la documentación que la acompaña. Esta librería, entre otras cosas, puede generar un corpus de documentos, realizar preprocesamiento en los mismos y generar una matriz de frecuencia de términos, donde puedes obtener aquellos más frecuentes. Por ejemplo, la siguiente función leé de forma recursiva, todos los textos dentro de un directorio dado (rut.data), los preprocesa y devuelve una matriz de terminos, además de los n términos mas frecuentes de acuerdo a su aparición en los textos:

```
library(tm)
freqterms <- function(rut.data,n){
    corp <- Corpus(DirSource(rut.data,recursive=TRUE),
            readerControl=list(language="en_US"))
    ## si quieres leer los textos almacenados en la lista corp:
    ##inspect(corp[[1]])
    ## preproceso
    corp <- tm_map(corp,stripWhitespace)
    corp <- tm_map(corp,removeNumbers)
    corp <- tm_map(corp,content_transformer(tolower))
    corp <- tm_map(corp,removePunctuation)
    corp <- tm_map(corp,removeWords,stopwords("english"))
    corp <- tm_map(corp,stemDocument)
    ## obtiene matriz de terminos
    tdm <- TermDocumentMatrix(corp,control=list(minDocFreq=100))
    ## remuevo terminos poco repetidos (95% sparsity)
    tdm <- removeSparseTerms(tdm, 0.95)
    ## frecuencia de terminos
    term.freq <- rowSums(as.matrix(tdm))
    sort.freq <- sort.int(term.freq,decreasing=TRUE,index.return=TRUE)
    ## los n mas frecuentes
    nterms.corp <- names(sort.freq$x[1:n])
    return(list(nterms=nterms.corp,tdm=tdm,ndocs=length(corp)))
}
```

Otra función que puede servirte es la siguiente, que crea una matriz de conteos de los términos (palabras) contenidos en `terms`, en los documentos contenidos en una matriz de terminos `tdms`:

```
count.terms <- function(tdms,terms){
    freq.table <- NULL
    for(i in 1:length(terms)){
        aa <- tm_term_score(tdms,terms[i])
```

```
        freq.table <- cbind(freq.table,aa)
    }
    colnames(freq.table) <- terms
    return(freq.table)
}
```

Cuando son muchos documentos, generar la matriz de Gram puede ser muy costosa. Si no son tantos términos, puedes usar PCA directamente en la matriz de frecuencia de términos D.

Si los documentos son cortos, puedes unir varios de ellos para tener una matriz D no tan rala. Trata de tener documentos lo mas balanceados posible. Esta unión puede hacerse directamente en la matriz de términos generada. Preguntame si tienes dudas.

Para String Kernels, puedes usar la librería `kernlab`. Puedes definir diferentes kernels spectrum, sequence, boundrange, constant o exponential con la función `stringdot` mediante

```
kern.spec <- stringdot(type="spectrum",length=len.kern,
                       normalized=TRUE)
```

Leé la ayuda para mas referencias. La matrix de Gram puedes construirla usando la función `kernelMatrix`. Por ejemplo, `kernelMatrix(kern,data.train,data.test)`, devuelve la matriz de Gram que incluye datos de entrenamiento y prueba. La función `kpc` realiza Kernel PCA y `pcv()` calcula los componentes principales.