

Proyecto PCA a través de SVD

Instituto Tecnológico Autónomo de México
Maestría en Ciencia de Datos

Por

Mirtha Ayala Macías 56596
Luis Federico Puente Peña 103108
David Rivera Flores 160668

Mayo 2018

Contents

1	Introducción	3
2	Objetivo	4
3	Metodología	5
3.1	Compilación y ejecución	5
3.2	Lectura de archivo .CSV	5
3.3	Guardado de memoria y creación de matrices y vectores para implementar SVD	5
3.4	Estandarización	5
3.5	Descomposición de Valores Singulares (SVD)	5
3.6	Análisis de componentes principales (PCA)	5
4	Resultados	6
5	Imágenes del compilado y output del binario.	8
6	Conclusiones	10
7	Referencias	10

1 Introducción

Los métodos de descomposición de valores singulares (SVD por sus siglas en ingles) y análisis de componentes principales (PCA, por sus siglas en inglés) son ampliamente utilizado en estadística para encontrar, ordenar y sintetizar la información.

El método SVD permite realizar una factorización de una matriz, es una generalización de la descomposición de eigenvalores. Tiene aplicaciones para procesar y realizar estadísticas.

El método PCA utiliza para describir un conjunto de datos en términos de nuevas variables (“componentes”) no correlacionados. Los componentes se por la varianza que describen. Lo que hace el método es reducir dimensionalidad, es decir, crea nuevas variables que capturan la información más relevante.

2 Objetivo

El objetivo del proyecto es desarrollar un algoritmo para implementar los métodos de SVD y PCA en el lenguaje de programación C. La implementación consiste en leer un archivo en formato .CSV, construir la matriz, normalizar los datos, aplicar la descomposición SVD y finalmente identificar los componentes principales.

Para esto, se utilizan las herramientas aprendidas en el curso de Métodos Numéricos y Optimación tanto de las librerías estándar de C como de terceros. Por ejemplo, `libstd` (`malloc`, `alloc`), `libstdio` (`printf`), `mat`, `Lapack`, `Blas`, `CSV-parser`, etc.

3 Metodología

3.1 Compilación y ejecución

Para la compilación se utilizó gcc/clang, las bibliotecas blas, lapack, csvparser y el comando make. Make es un programa que automatiza la construcción de programas que permite también configurar algunos flags del compilador para activar warnings y compilar el programa en Mac y Linux.

Para compilar el programa debe ejecutarse el siguiente comando: make con el parámetro build. Para borrar los programas compilados debe ejecutarse el siguientes comando: make con el parámetro clean. Para ejecutar el programa compilado se utiliza: ./svd_pca.

Make invoca al compilador para construir las bibliotecas csvparser y matrix y posteriormente compila el programa svd_pca y lo enlaza con las bibliotecas math, lapack, blas, csvparser y matrix (en el caso de mac usa el framework Accelerate, es decir, incluye lapack y blas).

3.2 Lectura de archivo .CSV

Para almacenar los datos desde el archivo csv. se diseñó una matriz de dos dimensiones que se alojada en memoria al mismo tiempo que los datos del csv. son leídos y convertidos a flotantes (doubles).

3.3 Guardado de memoria y creación de matrices y vectores para implementar SVD

En general para todo el desarrollo de la implentación agregamos funciones para alojar y liberar matrices en memoria e imprimirlas. Todo esto mediante estructuras de datos y punteros hacia doubles.

3.4 Estandarización

La normalización de matriz se hace con funciones de BLAS (daxpy y dgemm). La matriz de 2 dimensiones se utiliza para almacenar los datos de entrada (desde un archivo CSV) y las funciones de blas y lapack usan matrices de 1 dimension (RowMayor Order).

Integramos las funciones para calcular la media y la desviación estándar, con las nuevas funciones para hacer la normalización de la matriz. Esto fue especialmente un reto interesante, porque convertimos la matriz de 2 dimensiones a una matriz/vector de una dimensión para usar las funciones de blas usando el row-major (C) order en lugar de el col-major(Fortran) order, además de combinarlo con una matriz diagonal.

3.5 Descomposición de Valores Singulares (SVD)

En esta etapa se utiliza el método SVD mediante la función dgesvd de las biblioteca LAPACK y el porcentaje de varianza explicada.

Finalmente, usamos las condicionales ifdef y elif junto con las macros de sistema pre-definidas para determinar el archivo de cabecera de blas y la declaración de la función dgesvd_ de LAPACK para compatibilidad entre los sistemas operativos Linux y MacOS.

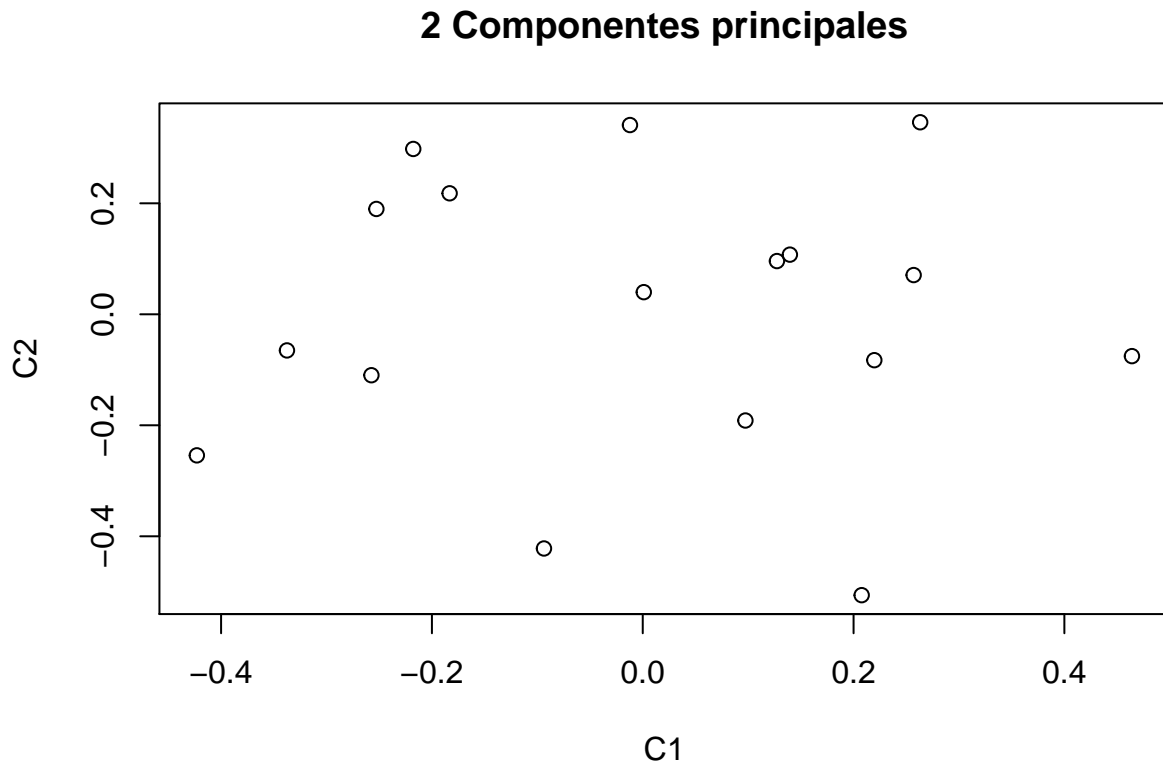
3.6 Análisis de componentes principales (PCA)

Por último, mediante la descomposición SVD se realiza una transformación de las matices para general las variables relacionadas con los componentes principales (PCA).

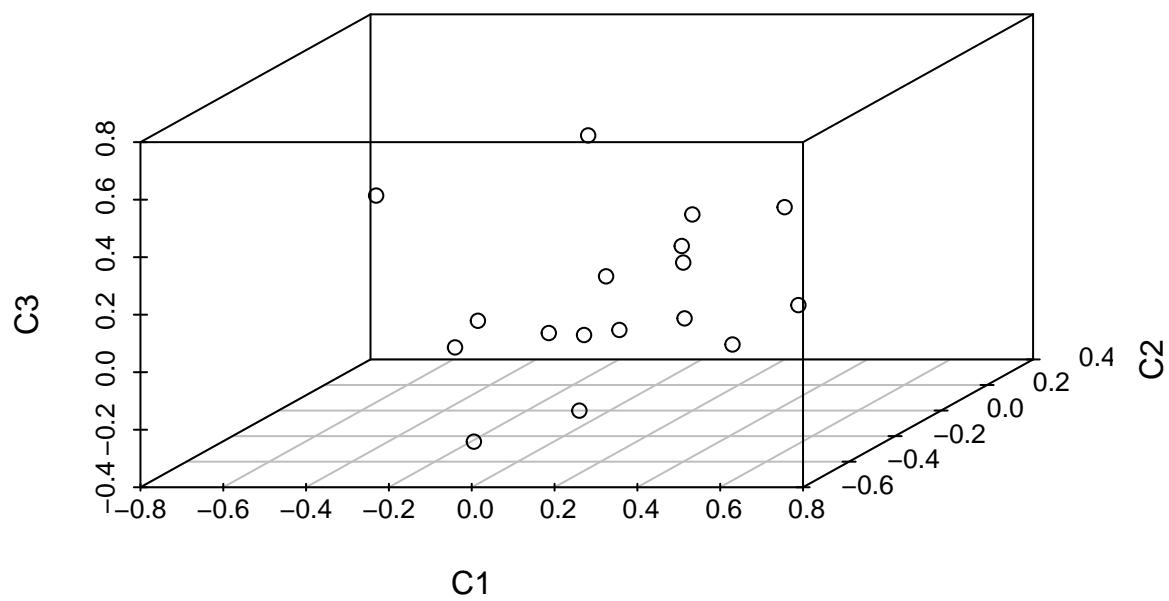
4 Resultados

Mediante todo la SVD es posible obtener los componentes principales del método PCA. Para esta sección se utilizó una base de datos pública de (Kaggle) la cual contiene información del precio de venta de casas en Ames, Iowa en Estados Unidos, así como las características de la casa y de la localidad. A fin de hacer pruebas se realizó la implementación con pocas observaciones. [link](#)

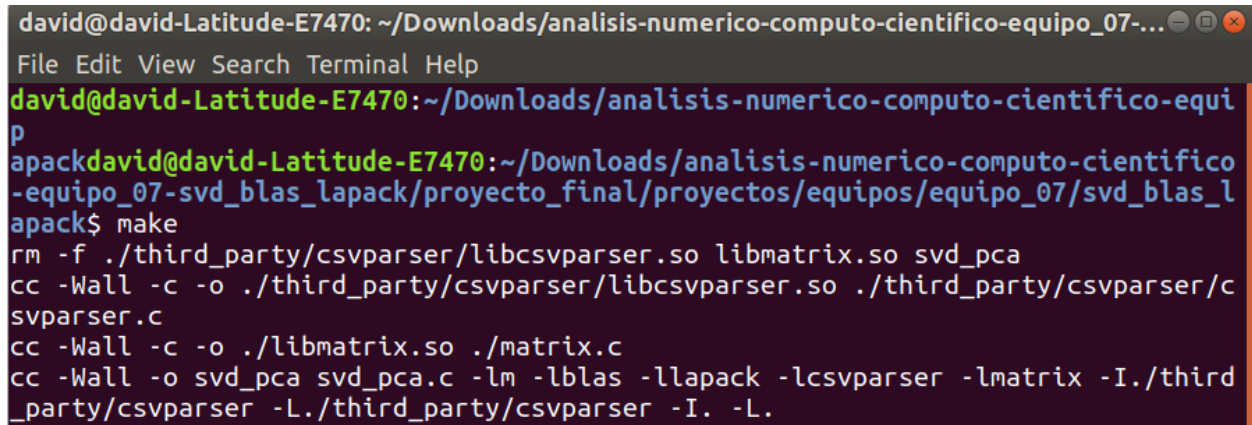
A continuación se grafican los primeros componentes.



Componentes principales



5 Imágenes del compilado y output del binario.



A terminal window titled "david@david-Latitude-E7470: ~/Downloads/analisis-numerico-computo-cientifico-equip..." with a menu bar (File, Edit, View, Search, Terminal, Help). The prompt is "david@david-Latitude-E7470:~/Downloads/analisis-numerico-computo-cientifico-equip". The user enters "p", then "apackdavid@david-Latitude-E7470:~/Downloads/analisis-numerico-computo-cientifico-equip_07-svd_blas_lapack/proyecto_final/proyectos/equipos/equip_07/svd_blas_lapack\$ make". The output shows the removal of old object files and the compilation of three new object files: "libcsvparser.so", "libmatrix.so", and "svd_pca".

```
david@david-Latitude-E7470: ~/Downloads/analisis-numerico-computo-cientifico-equip...
File Edit View Search Terminal Help
david@david-Latitude-E7470:~/Downloads/analisis-numerico-computo-cientifico-equip
p
apackdavid@david-Latitude-E7470:~/Downloads/analisis-numerico-computo-cientifico-equip_07-svd_blas_lapack/proyecto_final/proyectos/equipos/equip_07/svd_blas_lapack$ make
rm -f ./third_party/csvparser/libcsvparser.so libmatrix.so svd_pca
cc -Wall -c -o ./third_party/csvparser/libcsvparser.so ./third_party/csvparser/csvparser.c
cc -Wall -c -o ./libmatrix.so ./matrix.c
cc -Wall -o svd_pca svd_pca.c -lm -lblas -llapack -lcsvparser -lmatrix -I./third_party/csvparser -L./third_party/csvparser -I. -L.
```



```
david@david-Latitude-E7470: ~/Downloads/analisis-numerico-computo-cientifico-equipo_07-...
File Edit View Search Terminal Help
0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000

Raiz de valores singulares
-----
3.1830 2.3658 2.0143 1.8619 1.6330 0.8347 0.6261 0.3196 0.2459 0.1602
0.0821 0.0417 0.0084 0.0066 0.0006 0.0001 0.0000

Vectores propios
-----
0.0505 0.0530 -0.3245 0.1374 -0.2177 0.2979 0.4244 0.2117 -0.0598 0.1304
-0.0526 -0.1642 0.4300 -0.3392 0.0370 -0.3145 -0.2425
0.2206 -0.2113 0.1965 -0.0483 0.4641 -0.0754 0.3408 0.3205 -0.2234 0.0449
-0.1803 0.0441 -0.0781 -0.2376 0.0233 0.4665 -0.2425
0.0531 0.0045 -0.2696 -0.0668 -0.3375 -0.0651 -0.1515 -0.3791 0.1051 0.0237
-0.0223 -0.0961 0.2239 -0.1237 0.2143 0.6661 -0.2425
-0.4404 0.3315 -0.1891 -0.2016 0.2570 0.0707 -0.2017 0.3953 0.4160 -0.1067
0.0341 0.2880 0.0708 -0.0873 0.0714 0.0818 -0.2425
-0.0049 -0.4090 -0.1380 -0.1462 -0.2574 -0.1098 -0.0390 0.1829 -0.2114 -0.6072
0.4073 0.1135 -0.1187 -0.0795 -0.0374 -0.0795 -0.2425
0.1187 -0.0132 -0.2323 0.1250 -0.0937 -0.4220 -0.3206 0.0425 -0.1057 -0.0156
-0.6451 0.1568 -0.0880 -0.1141 -0.2074 -0.2232 -0.2425
0.1616 -0.3357 0.2508 0.1043 0.2632 0.3460 -0.1871 -0.3008 0.4752 -0.2728
-0.1410 -0.1487 0.1028 -0.1648 -0.0911 -0.1600 -0.2425
```

```
david@david-Latitude-E7470: ~/Downloads/analisis-numerico-computo-cientifico-equipo_07-...
File Edit View Search Terminal Help
david@david-Latitude-E7470:~/Downloads/analisis-numerico-computo-cientifico-equi
po_07-svd_blas_lapack/proyecto_final/proyectos/equipos/equipo_07/svd_blas_lapack
$ ./svd_pca
Matriz de entrada
-----
-5.2941 -1918.0000 30.5294 20.9412 70.4706 -282.0588 -213.4706
-236.8824 443.8824 207.0000 13.4706 -94.4118 8.2941
-51.8235 -4.1765 0.2353 18064.7059
9.7059 -768.0000 3.5294 -6.0588 342.4706 -148.0588 192.5294
169.1176 -410.1176 -241.0000 -74.5294 203.5882
-52.7059 -51.8235 -1.1765 -0.7647 -8935.2941
-2.2941 882.0000 28.5294 19.9412 -149.5294 1.9412 -149.4706
-172.8824 455.8824 283.0000 73.4706 -94.4118 -10.7059
-51.8235 2.8235 0.2353 33064.7059
-10.2941 -818.0000 -57.4706 -12.0588 -419.5294
107.9412 -313.4706 -131.8824 345.8824 214.0000
107.4706 -94.4118 -17.7059 220.1765 -4.1765 -1.7647 -
50435.2941
13.7059 3892.0000 27.5294 17.9412 19.4706 57.9412 75.5294 52.1176 642.8824
695.0000 301.4706 97.5882 31.2941 -51.8235 5.8235
0.235359564.7059
14.7059 3747.0000 20.5294 12.9412 96.4706 -368.0588 -273.4706
-296.8824 155.8824 -141.0000 -54.5294 -54.4118
```

6 Conclusiones

En este proyecto se desarrolló un algoritmo para implementar los métodos de SVD y PCA en el lenguaje de programación C mediante las herramientas aprendidas en el curso de Métodos Numéricos y Optimización (malloc, alloc, libstdio -printf-, mat, Lapack, Blas, CSV-parser, etc).

7 Referencias

https://www.tutorialspoint.com/c_standard_library/math_h.htm
<https://sourceforge.net/projects/cccsvparser/>
<https://developer.apple.com/documentation/accelerate?language=objc>
<https://developer.apple.com/documentation/accelerate/blas?language=objc>
https://developer.apple.com/documentation/accelerate/1513298-cblas_daxpy?language=objc
https://developer.apple.com/documentation/accelerate/1513282-cblas_dgemm?language=objc
<https://gcc.gnu.org/onlinedocs/cpp/Ifdef.html>
<https://gcc.gnu.org/onlinedocs/cpp/System-specific-Predefined-Macros.html#System-specific-Predefined-Macros>
https://www.ibm.com/support/knowledgecenter/en/SSFHY8_5.4.0/com.ibm.cluster.essl.v5r4.essl100.doc/am5gr_hgesvd.htm
http://www.netlib.org/lapack/explore-html/d1/d7e/group___double_g_esing_ga84fdf22a62b12ff364621e4713ce02f2.html#ga84fdf22a62b12ff364621e4713ce02f2