Proyecto 2

Curso Propedéutico 2017
Maestría en Ciencias de Datos 2017

Instructor: Mauricio Benjamín García Tec

El proyecto consta de dos partes en las que se evaluará:

1. Conocimiento teórico de Álgebra Lineal
2. Aplicaciones de la Descomposición en Valores Singulares (SVD) a sistemas de ecuaciones y aproximación de matrices (reducción/comprensión de información) usando `Python` y la librería `numpy`.

La **fecha de entrega** es el **domingo 2 de julio a las 11:59pm** al correo electrónico mencionado. Si no contestan todas las preguntas del proyecto, entreguen lo más que hayan completado.

El **formato de entrega** de ambas partes será en un **cuaderno de Jupyter** titulado `Tarea2.pynb` que pondrán en el repositorio de Github de la clase en la carpeta `Alumnos/<su nombre>/`. Deberán generar un pull request.

Los cuadernos de Jupyter permiten escribir matemáticas usando MathJax (una versión de **Latex** para HTML). Idealmente sus respuestas deberán usar **MathJax**, de no ser posible, usen caractéres normales.

Parte 1: Teoría de Álgebra Lineal y Optimización

En esta sección se verificará que se tiene un conocimiento básico de Álgebra Lineal.

Deberán contestar brevemente las siguientes preguntas, intentando resaltar el aspecto intuitivo y no definiciones abstractas:

1. ¿Por qué una matriz equivale a una transformación lineal entre espacios vectoriales?
2. ¿Cuál es el efecto de transformación lineal de una matriz diagonal y el de una matriz ortogonal?
3. ¿Qué es la descomposición en valores singulares de una matriz?
4. ¿Qué es diagonalizar una matriz y que representan los eigenvectores?
5. ¿ Intuitivamente qué son los eigenvectores?
6. ¿Cómo interpretas la descomposición en valores singulares como una composición de tres tipos de transformaciones lineales simples?
7. ¿Qué relación hay entre la descomposición en valores singulares y la diagonalización?
8. ¿Cómo se usa la descomposición en valores singulares para dar un aproximación de rango menor a una matriz?
9. Describe el método de minimización por descenso gradiente
10. Menciona 4 ejemplo de problemas de optimización (dos con restricciones y dos sin restricciones) que te parecan interesantes como Científico de Datos

Parte 2: Aplicaciones en Python

Van a mostrar tres aplicaciones de la SVD y optimizacíon. Recuerden que la multiplicación matricial en numpy es con la función `numpy.multiply`.

1. Se mostrará una aplicación de la SVD a la compresión de imágenes y reducción de ruido. Podrán usar la función `numpy.linalg.svd`, pueden consultar la ayuda en este link.

Una imagen puede verse como un arreglo de `m x n` entradas donde cada entrada representa un pixel. El caso más sencillo es el caso de imágenes en blanco y negro, donde cada entrada toma un valor en el intervalo `(0,1)` que representa una escala de gris desde negro hasta blanco. Una imagen en este caso es simplemente una matriz numérica con coeficientes en `(0,1)`. Este proyecto es más sencillo cuando se trabaja exclusivamente en blanco y negro. Hace el proyecto con imágenes a color es totalmente optativo y no es requerido.
Deberán hacer un script de `Python` que realicen las siguientes tareas:
    - Recibir el *path* de un archivo de imagen `png` y convertirlo en una matriz numérica que represente a la versión en blanco y negro de la imagen. Ayuda aquí.
    - Realizar y verificar la descomposición `svd`.
    - Usar la descomposición para dar una aproximación de grado `k` de la imagen.
    - Para alguna imagen de su elección, elegir distintos valores de aproximación a la imagen original.
    - Contestar, ¿qué tiene que ver este proyecto con compresión de imágenes?
2. Ahora veremos la aplicación a pseudoinversa y sistemas de ecuaciones
    - Programar una función que dada cualquier matriz devuelva la pseudainversa usando la descomposición SVD. Hacer otra función que resuelva cualquier sistema de ecuaciones de la forma Ax=b usando esta pseudoinversa.
    - Jugar con el sistema Ax=b donde A=[[1,1],[0,0]] y b puede tomar distintos valores. (a) Observar que pasa si b esta en la imagen de A (contestar cuál es la imagen) y si no está (ej. b = [1,1]). (b) Contestar, ¿la solución resultante es única? Si hay más de una solución, investigar que carateriza a la solución devuelta. (c) Repetir cambiando A=[[1,1],[0,1e-32]], ¿En este caso la solucíon es única? ¿Cambia el valor devuelto de x en cada posible valor de b del punto anterior?
3. En este ejercicio usarán la paquetería `pandas` para trabajar con datos link a introducción a pandas, programarán un ajuste de mínimos cuadrados.
    - Deben programar un script que lea el archivo study_vs_sat.csv y lo almacene como un data frame de pandas.
    - Pleantear como un problema de optimización que intente hacer una aproximación de la forma `sat_score ~ alpha + beta*study_hours` minimizando la suma de los errores de predicción al cuadrado, pueden consultar este link
      ¿Cuál es el gradiente de la función que se quiere optimizar (hint: las variables que queremos optimizar son alpha y beta)?
    - Programar una función que reciba valores de alpha, beta y el vector sat_score y devuelva un vector array de numpy de predicciones `alpha + beta*study_hours_i`, con un valor por cada individuo
    - Definan un numpy array X de dos columnas, la primera con unos en todas sus entradas y la segunda con la variable study_hours. Observen que `X*[alpha,beta]` nos devuelve `alpha + beta*study_hours_i` en cada entrada y que entonces el problema se vuelve `sat_score ~ X*[alpha,beta]`
    - Calculen la pseudoinversa X^+ de X y computen `(X^+)*sat_score` para obtener alpha y beta soluciones.
    - Comparen la solución anterior con la de la fórmula directa de solución exacta `(alpha,beta)=(X^t*X)^(-1)*X^t*study_hours`.
    - **(Avanzado)** Usen la libreria `matplotlib` par visualizar las predicciones con alpha y beta solución contra los valores reales de sat_score.
    - **(Muy avanzado)** Programen el método de descenso gradiente para obtener alpha y beta por vía de un método numérico de optimización. Experimenten con distintos learning rates (tamaños de paso)