Proyecto 1

Profesor: Cristóbal Rojas
Ayudante: Gustavo Cornejo

Introducción

En este proyecto pondrán a prueba lo aprendido sobre vectores, su manipulación y usos. En particular, desarrollarán la clase `Vec` que permite trabajar con vectores generalizados. La primera parte de este proyecto consiste en utilizar esta clase para elaborar un sistema recomendador de películas, mientras que en la segunda parte del proyecto se utilizará la librería `numpy` para resolver un problema en el contexto de sistemas de ecuaciones lineales. Este proyecto podrán realizarlo en grupo, con un **máximo de 3 integrantes por grupo**. El plazo para la entrega del proyecto es hasta el **8 de Noviembre a las 23:59 hrs.**.

Clase `Vec` (15 pts.)

Su primera tarea será crear la clase `Vec` en un programa de python, la cual utilizán para manipular vectores en formato de diccionarios de python. Los vectores de esta clase tendrán únicamente dos atributos:

- Dominio(`D`): representa el conjunto de labels o coordenadas que puede tener un vector.
- coordenadas(`f`): valores asociados a cada coordenada (label) del vector.

Usted deberá crear ambos atributos de la siguiente manera: el dominio de cada vector debe ser una lista, mientras que el atributo de coordenadas debe ser un diccionario donde las `keys` sean los labels de las coordenadas cuyo valor es distinto de 0, y los `values` correspondan al valor en dicha coordenada. Un ejemplo de esto sería el siguiente:

Suponga que quiere definir el vector $v = (0, 1)$ en $\mathbb{R}^2$, entonces, en este caso los atributos serían de la siguiente forma:

```
D=['x','y']  ;  f={'y':1}
```

y el vector se instanciaría como

```
v = Vec(['x','y'],{'y':1}).
```

Por otro lado para $v = (0, 1, 1)$ en $\mathbb{R}^3$ tendríamos

```
D=['x','y','z']  ;  f={'y':1,'z':1}
```

y la instanciación

```
v = Vec(['x','y','z'],{'y':1,'z':1}).
```

Luego de esto deberá implementar las siguientes funciones (ya sea como métodos de la clase Vec o como funciones fuera de la clase):

- `getitem(v, k)`: entrega el valor de la coordenada k del vector v
- `setitem(v, k, val)`: cambia el valor de la coordenada k del vector v a `val`
- `equal(u, v)`: compara los vectores u y v, y entrega $True$ si son iguales y $False$ si no.
- `add(u, v)`: entrega la suma de los vectores u y v.
- `dot(u, v)`: entrega el resultado del producto punto entre u y v.
- `scalar_mul(v, alpha)`: entrega un nuevo vector u, que corresponde a $u = \alpha v$

- `neg(v)`: entrega un nuevo vector que corresponde al inverso aditivo de v.

A continuación, se presentan algunas consideraciones que es necesario tener en mente:

- Para cada método se debe asegurar que las operaciones preserven la condición **sparse** del vector resultante. Por ejemplo, si se ejecuta la operación `w = add(u,v)` para

  `u=Vec(['x','y','z'],{'y':1})` y `v=Vec(['x','y','z'],{'z':1})`,

  el resultado deberá ser

  `w= Vec(['x','y','z'],{'y':1,'z':1})` y no `w=Vec(['x','y','z'],{'x':0,'y':1,'z'=1})`.

  Además, se debe respetar el dominio y la definición de cada operación, es decir, si se ejecuta la linea `print(getitem(u,'R'))` para `u=Vec(['x','y','z'],{'y':1})`, el programa deberá avisar de alguna forma (por ejemplo, a través de un print) que esta operación no puede realizarse ya que `'R'` no es un label válido de u (no está en su dominio). De igual forma deben considerarse las condiciones necesarias para todos los demás métodos (no se pueden sumar vectores con dominios distintos, etc...).

- Es importante que te asegures que tus métodos funcionen correctamente, pues los usarás en la segunda parte del proyecto.

Sistema Recomendador de Películas (25 pts.)

Contexto

Existe una infinidad de aplicaciones que se basan en vectores. Ejemplo de esto son prácticamente todos los algoritmos de *Machine Learning*, que mezclan tópicos como Álgebra Lineal y Estadística, y el área de las Ciencias de Datos que investiga sobre Sistemas Recomendadores como los que se encuentran presentes en las plataformas *Amazon*, *Facebook*, *Netflix*, etc. *Amazon* por ejemplo, ofrece productos en base a las preferencias del usuario, las cuales pueden ser extraídas en forma de *rating* explícito (calificación que un usuario haya asignado a un producto), o en forma de *rating* implícito, el cual se basa en el comportamiento de un usuario (por ejemplo, la cantidad de tiempo que visita la página de un determinado producto).

Dentro del área de Sistemas Recomendadores, se utilizan diversas técnicas para predecir las preferencias/gustos de los usuarios. Dentro de las más simples e intuitivas se encuentran los métodos basados en la idea de usar los gustos de usuarios que son *vecinos* del usuario al que se le quieren hacer recomendaciones, en el sentido que sus gustos son *similares*. Esta técnica es ampliamente utilizada en contextos de predicción (no solo en Sistemas Recomendadores), y suele ser denominada KNN, en referencia al término *k-Nearest Neighbors* (o los k vecinos más cercanos).

Pero, ¿cómo se puede saber qué tan similares son los gustos de dos usuarios? Existen múltiples medidas de similaridad, basadas en diferentes nociones de *distancia*. Para este proyecto utilizaremos una medida llamada **similaridad de coseno**, la cual en términos prácticos mide qué tan "alineados" están dos vectores dados. Formalmente, si u y v son los vectores de gustos de dos usuarios, entonces definimos la *similaridad* entre u y v como

$$\text{sim}(u, v) = \frac{u \cdot v}{\left(\sqrt{u \cdot u}\right)\left(\sqrt{v \cdot v}\right)} \tag{1}$$

donde $u \cdot v$ es el producto punto entre u y v. Es importante notar que, como en este caso los vectores no contienen componentes negativas, el restultado de esta medida es siempre positivo. Además, el factor en el denominador hace que el valor de esta medida sea como máximo 1 (cuando $u = v$). La idea es que mientras

más alineados esten los vectores u y v, más cercano a 1 será el valor de $\text{sim}(u, v)$, lo que interpretamos como una mayor similaridad entre u y v.

Ahora, consideremos el contexto de una plataforma que ofrece *streaming* de un total de n películas a m usuarios, y supongamos que dicha plataforma cuenta con datos de la forma (`user id`, `movie id`, `rating`), donde `rating` es un número entre 1 y 5 (como las estrellas de *Netflix*). Luego, para cada usuario i (con $i \in \{1, \ldots, m\}$), podemos definir su vector de gustos como $u_i = (\texttt{rating movie}_1, \ldots, \texttt{rating movie}_n)$. De esta forma, se estima que usuarios que tienen una similaridad alta entre sus vectores de gustos, tienen preferencias similares.

MovieLens

MovieLens consiste en un recomendador de películas que se remonta a la década de los '90. Su funcionamiento se basa en *ratings* explícitos, es decir, en recomendar películas en base a los *ratings* que han entregado los usuarios. Para este proyecto, recibirás una pequeña muestra de la base de datos, conformada por 943 usuarios, 1.682 películas y 80.000 *ratings*.

Los archivos de la base de datos son los siguientes:

- `ratings.csv`: Es un archivo separado por comas, que contiene datos de la forma (`user_id`, `movie_id`, `rating`). El primer elemento corresponde al identificador único de un usuario, el segundo al identificador único de una película, y el tercero al `rating` entregado por el usuario a dicha película.
- `movies.csv`: Es un archivo separado por comas, que contiene pares de la forma (`movie_id`, `movie_name`). Al igual que en el archivo anterior, el primer elemento corresponde al identificador único de la película, mientras que el segundo corresponde al nombre de dicha película.
- Recuerda que para leer un archivo `file.csv`, separar el contenido de sus filas y ponerlo todo por ejemplo en una lista llamada `archivo`, puedes usar:

```
archivo=[i.strip().split(‘,’) for i in list(open(‘file.csv’))]
```

Por desarrollar

En base a lo anterior, se te pide desarrollar los siguientes puntos utilizando la clase `Vec` programada previamente:

(a) Para cada usuario, crea su *vector de gustos*, utilizando la clase `Vec`. Para esto, debes definir correctamente el dominio (*ids* de las películas), y la función `movie_id` $\to$ `rating` (utilizando un diccionario). Si un usuario no ha calificado alguna película, debes asumir que el *rating* que asigna a dicha película es 0 (nota que esto se hace automáticamente gracias a la clase `Vec`). Guarda los vectores en un diccionario llamado `users`, utilizando la sintaxis `users[user_id] = user_vec` de modo que `users[n]` sea el *vector de gustos* del usuario con id n. **(5 pts.)**

(b) De manera equivalente, para cada película crea su vector de la clase `Vec` con los ratings que todos los usuarios le dieron a esta película. Debes definir correctamente el dominio (*ids* de los usuarios), y la función `user_id` $\to$ `rating`. Nuevamente, si un usuario no ha calificado una película, debes asumir que el *rating* es 0. Guarda los vectores en un diccionario llamado `movies`, utilizando la sintaxis `movies[movie_id] = movies_vec`, de modo que `movies[movie_id]` sea el vector de la película con este *id*. **(5 pts.)**

(c) Define una función `vecinos(users, user_id, k)` que recibe un diccionario `users` (donde `users[i]` es el vector de gustos del usuario i), el identificador `user_id` de un usuario en el diccionario `users` y

un entero positivo k. La función debe, en base a la noción de similaridad de coseno (1), encontrar los k usuarios del diccionario `users` más similares a `user_id` (pero diferentes de este!), y retornarlos en una lista en orden decreciente de similaridad utilizando el formato `[(user_i, similatity_i), ..., (user_k, similarity_k)]` (Nota que es una lista de tuplas). Se asumirá que $k \in \{0, \ldots, |\texttt{users}| - 1\}$ y que el *id* del usuario es válido. **(7 pts.)**

HINT: Recuerda que para ordenar un diccionario `dict` de forma decreciente según sus valores, puedes usar

```
sorted(dict.items(), key = lambda x:x[1], reverse=True)
```

que entrega una lista `[(key,value),...]` en el order deseado.

(d) Programe un método `recomendación(user_id)` que entrege como resultado una o varias peliculas que recomiende para el usuario entregado en el argumento de la función. Para esto se recomienda usar la función `vecinos(users,user_id,k)` y considerar las opiniones de sus vecinos más cercanos, lo cual se puede hacer, por ejemplo, tomando la media un promedio entre las recomendaciones de los vecinos o asignando un más importancia a los vecinos más cercanos o tomando directamente una recomendación por vecino. Debe justificar la elección que hizo para su programa, comentando por qué eligio el sistema usado, la cantidad de vecinos utilizada y la cantidad de peliculas entregadas por su función. **(4 pts.)**

(e) Reflexione: ¿cuál es la ventaja de enfrentar este problema utilizando la clase que programó en la primera parte de la tarea? **(4 pts.)**

HINT: Incluya el concepto de *sparsity* en su respuesta.

Resolución de sistemas lineales (25 pts.)

El objetivo de este ejercicio es que puedan familiarizarse con herramientas ya existentes para la resolución de ciertos problemas de algebra lineal, en particular herramientas como las librerias numpy y pandas.

Descripción del problema.

Deberán ayudar a una empresa a calcular correctamente precios de producción de nuevos productos que planean ingresar al mercado. Para esto, la empresa le entrega los archivos siguientes:

- Costos_prod.csv : contiene los costos de producción de 50 productos que la empresa ya está vediendo
- Recursos.csv : contiene la cantidad necesaria de cada una de las materias primas para la elaboración de cada uno de los productos en Costos_prod.csv.
- Recursos_new.csv : contiene la cantidad necesaria de cada una de las materias primas para la elaboración de los productos nuevos de la empresa.

Con esta información, deberán determinar el costo de cada uno de los nuevos productos. Para resolver el problema, deberán seguir los pasos siguientes:

a) Cargar los datos de Costos_prod.csv y Recursos.csv a un dataFrame de la librería pandas e imprimir estos dataframes en la pantalla, con el fin de entender cómo se organizan los datos (para la impresión en pantalla se recomienda usar el comando dataFrame.head()).

b) Crear una matriz A utilizando la librería numpy, donde la entrada i, j de la matriz corresponda a la cantidad necesaria de la materia prima j para producir el producto i.

c) Crear un vector de costos de producción con la librería numpy.

d) Obtener el precio de cada materia prima resolviendo la ecuación $Ax = b$, donde A corresponde a la matriz creada en el item b), b corresponde al vector creado en el item c), y x corresponderá al vector de precios buscado (para esto se recomienda usar el método solve() de la librería numpy.linalg, que pueden revisar en: https://numpy.org/doc/stable/reference/generated/numpy.linalg.solve.html)

De esta forma, una vez conseguidos los precios, deberá entregar una lista con los precios para los productos nuevos, de los cuales podrá encontrar información en el archivo Recursos_new.csv. Para lo anterior debe realizar lo siguiente:

e) Cargar los datos de Recursos_new.csv y crear una matriz de numpy tal como hizo en el inciso b), es decir, una matriz donde la entrada i, j de la matriz corresponda a la cantidad necesaria de la materia prima j para producir el producto i.

f) Calcular e imprimir en pantalla el producto matriz vector $A \cdot x$ donde A es la matriz que armó en el inciso e), y x es el vector de costos que obtuvo en d). Tenga cuidado de ordenar bien el vector de costos si es necesario.

Finalmente reflexione sobre las siguientes preguntas:

g) ¿Podría obtenerse la información de los precios que obtuvo en d) solo con la información de 40 productos antiguos?

h) ¿Sería útil de algún modo que los archivos Recursos.csv y Costos_prod.csv tuviesen la información de más productos antiguos? ¿Por qué?

Consideraciones adicionales

1. Debe entregar su respuesta de cada ejercicio en un jupyter notebook donde cada celda esté ejecutada. Además debe poder ser ejecutado sin problemas.
2. Debe dejar claro donde debe modificarse el path en cada caso para que se carguen archivos.
3. Se evaluará el orden del notebook (penalización máxima de **5 pts.** por falta de orden).
4. Si bien no se prohibe el uso de librerías externas, si debe comentarse dentro del mismo código el uso que se les da a estas librerías.
5. La fecha de entrega del proyecto es el **8 de Noviembre hasta las 23:59 hrs**.