Optimización convexa
Tarea 4: Programación por metas

Acoyani Garrido Sandoval

26 de Febrero de 2026

1. Introducción

La **programación por metas** es un método especial de la programación lineal cuyo objetivo es encontrar la forma más óptima de reconciliar metas contradictorias o que de alguna forma compiten entre sí. Este método es especialmente atractivo para la vida real, pues con frecuencia ocurren situaciones donde las restricciones de un escenario son imposibles de cumplir en su totalidad.

La idea principal detrás de este método consiste en **definir metas para los objetivos**, y posteriormente **minimizar la desviación respecto a esas metas.** Esto se hace convirtiendo nuestras restricciones en igualdades, y añadiendo 2 variables de desviación a nuestra función objetivo. Como ejemplo, si tenemos una restricción como la siguiente:

$$25x + 20y \leq 5000$$

Entonces convertimos nuestra desigualdad en igualdad y añadimos 2 variables adicionales, d^- y d^+:

$$25x + 20y + d^- - d^+ = 5000$$

donde d^- y d^+ representan significativamente un **déficit** (qué tanto nuestra solución estará por debajo de la meta) y un **exceso** (qué tanto nuestra solución estará por encima de la meta), y su función matemática es absorber la diferencia entre lo que deseamos y lo que pudimos encontrar en la vida real. Dichas variables tienen 2 restricciones:

- **Ambas variables deben ser positivas**, ya que no hay una sola variable que indique déficit o exceso, sino una para cada rubro.
- **Sólo una puede ser mayor que 0 a la vez**, ya que es imposible tener déficit y exceso al mismo tiempo.

Esta conversión de desigualdad a igualdad con exceso y déficit la hacemos sobre la restricción que podamos identificar en nuestro problema como **restricción suave,** por motivo de que hay un cierto grado de libertad dependiente de la persona a la cual aplique el problema para tener déficit o superávit en el cumplimiento de nuestra meta. Como deseamos cumplir esa meta lo mejor posible, **a partir de esta meta será nuestra función objetivo,** de la siguiente forma, siguiendo el ejemplo de la ecuación anterior:

$$25x + 20y + \rho d^- - \tau d^+ = 5000$$

donde ρ y τ son **factores de penalización** para nuestro déficit y superávit que se determinan al momento de resolver el problema con el fin de establecer qué tan grave es el déficit y el superávit: entre más alto sea ese parámetro, la ejecución de la solución tenderá fuertemente a evitar incurrir en déficit o superávit.

Y entonces, el problema consistirá en **minimizar lo que queremos evitar en nuestro problema**. Si nuestra meta suave es evitar incurrir en exceso, **minimizamos el exceso** d^+. Si nuestra meta suave es evitar el déficit, minimizamos el déficit d^-. Y si nuestra meta suave es ajustarnos lo más posible a un valor fijo, **minimizamos ambas variables.**

Entonces, suponiendo que nuestro problema consiste en evitar exceder un presupuesto establecido, **nuestra función objetivo** es:

$$min\ Z = 25x + 20y + \rho d^+$$

donde ya no figura la cantidad de 5000 ni el déficit d^- por motivo de que dichas cantidades son parte de la meta, no de nuestro problema, el cual es minimizar el exceso en el cumplimiento de la meta.

2. Escenario

Travel Tech es una agencia de viajes corporativos. Un día, la empresa recibe una petición para un viaje grande: el retiro anual de una startup tecnológica en la Ciudad de México, para lo cual será necesario alojar a 50 empleados durante un fin de semana. El cliente manifiesta que no desea hoteles convencionales, y en su lugar preferiría Airbnb's.

Sin embargo, algunos de sus requisitos resultan ser conflictivos, por lo que hay que plantear una propuesta que equilibre lo mejor posible sus demandas:

1. **Logística:** Prefieren pocas casas grandes, con el fin de minimizar la cantidad de reservaciones.
2. **Presupuesto:** Hay un tope de gastos, pero hay una cierta flexibilidad para excederlo.

3. **Calidad:** Sólo aceptan lugares bien calificados.

Para ello, vamos a construir un modelo de optimización que seleccione la combinación ideal de inmuebles, cumpliendo los requisitos físicos (restricciones duras) y minimizando las desviaciones de las metas financieras (restricciones suaves).

3. Conjunto de datos

La empresa Airbnb pone a disposición del público un conjunto de datos en su portal Inside Airbnb (Airbnb, 2026) que muestra datos trimestrales al día de hoy actualizados al dís 27 de Septiembre de 2025. Dicho conjunto de datos consta de una gran cantidad de inmuebles que han sido listados en la plataforma, y contiene datos tales como la colonia donde se ubica, la calificación del inmueble, el cupo máximo, o el precio en pesos; con lo que el conjunto de datos tiene información suficiente para poder tomar una decisión.

Sin embargo, antes de poder usarlo, es necesario limpiar el conjunto de datos: hay que convertir el texto que muestra el precio del inmueble y convertirlo a punto flotante, y tumbar aquellas columnas que no sean relevantes. Esto se logra mediante el siguiente código:

```
import pandas
import pulp


columnas_detalles_calificacion = \
[
    "review_scores_accuracy",
    "review_scores_cleanliness",
    "review_scores_checkin",
    "review_scores_communication",
    "review_scores_location",
    "review_scores_value"
]


def cargar_limpiar_datos(path):
    dataframe = pandas.read_csv(path, on_bad_lines="skip")

    dataframe["price"] = dataframe["price"].str.replace("$", "", regex=False)
    dataframe["price"] = dataframe["price"].str.replace(",", "", regex=False)
    dataframe["price"] = dataframe["price"].astype(float)

    columnas_relevantes = \
    [
        "price",
        "accommodates",
        "review_scores_rating"
    ]
    dataframe = dataframe.dropna(subset=columnas_relevantes)

    dataframe = dataframe.reset_index(drop=True)
    return dataframe
```

4. Planteamiento matemático

Una vez limpios los datos, el siguiente paso es plantear el modelo en la librería de Python llamada PULP, la cual es un resolvedor de problemas de programación lineal, para lo cual requerimos definir matemáticamente nuestro problema:

4.1. Restricciones

Restricciones suaves

- **Presupuesto:** El enunciado del problema establece que el cliente está dispuesto a asumir un cierto exceso en el presupuesto. Por lo que la restricción presupuestaria necesariamente **es una restricción suave.**

Restricciones duras

- **Capacidad:** Los inmuebles elegidos deben tener cupo para al menos 50 empleados.
- **Cantidad de reservaciones**: Los inmuebles elegidos deben sumar no más de 12.
- **Calidad:** Los inmuebles elegidos deben tener al menos 4.5 de calificación.

Todas ellas son restricciones duras que deben cumplirse obligatoriamente para poder entregar el paquete de reservaciones al cliente. Si no hay suficiente capacidad, no todos podrán hospedarse; si hay más de 12 reservaciones, será excesivamente difícil gestionarlas todas; y si la calificación es menor de 4.5, la experiencia para todos será decepcionante.

4.2. Variables

Nuestro problema se trata de **programación por metas,** por lo que nuestra función objetivo se construye a partir de la restricción suave, que es: el presupuesto por noche debe ser de preferencia no más de $45,000 pesos (el presupuesto objetivo total, que es de $70,000 pesos, entre 2 noches que van a durar los empleados en la Ciudad de México).

Entonces, las variables que usaremos para el problema serán:

Conjuntos e índices

I El conjunto de todos los inmuebles candidatos.

i El índice de cada inmueble candidato.

Variables de decisión

x_i Para cada inmueble candidato, ¿fue elegido? 1 significa que sí fue elegido, 0 significa que no se eligió. El índice de todas las x es uno por cada inmueble; si nuestro conjunto de datos es de 6000 inmuebles, será $\{x_1, x_2, ..., x_{6000}\}$.

Parámetros Se definen aquí los atributos de los inmuebles, con el fin de definir posteriormente las restricciones con base en ellos.

c_i Costo por noche de cada inmueble.

k_i Kapacidad de cada inmueble.

q_i Quality de cada inmueble.

Variables de desviación Se definen por ser el problema de programación por metas.

d^+ Exceso presupuestario.

d^- Subejercicio presupuestario.

Variables de penalización El factor de penalización es un multiplicador constante que se añade a las variables de desviación, y tiene como propósito definir qué tan agresivamente debe nuestro método minimizar la variable de desviación. Un factor mayor que 1 hará que el método "vea.^{ex}cesos más grandes, y por lo tanto, intente minimizar el resultado final de forma más agresiva; un factor menor que 1 hará que el método "vea.^{ex}cesos menores, lo cual lo hará más permisivo al minimizar el exceso.

ρ Penalización del exceso presupuestario.

4.3. Restricciones duras

Como lo hemos hecho en la programación lineal simple, nuestras restricciones duras aquí son desigualdades, pues éstas únicamente pueden cumplirse o incumplirse sin ningún margen intermedio:

- **Capacidad:** Los inmuebles elegidos deben tener cupo para al menos 50 empleados.
$$\sum_{i=1}^{|I|} (k_i x_i) \geq 50$$
- **Cantidad de reservaciones**: Los inmuebles elegidos deben sumar no más de 12.
$$\sum_{i=1}^{|I|} (x_i) \leq 12$$
- **Calidad:** Los inmuebles elegidos deben tener al menos 4.5 de calificación.
$$q_i \geq 4.5$$

4.4. Restricción suave y función objetivo

Restricción suave Conforme al enunciado del problema, nuestra restricción suave es que el presupuesto de preferencia debe ser no más de $35,000 pesos por noche, pero hay espacio para exceder el presupuesto. Por lo tanto, nuestra restricción suave es:

$$\sum_{i=1}^{|I|} (c_i x_i) + d^- - d^+ = 35,000$$

Función objetivo Queremos retendemos reducir el exceso d^+ todo lo que sea posible. Por lo tanto, nuestro problema es de **minimizar el exceso** y por lo tanto nuestra función objetivo es:

$$min\ Z = \sum_{i=1}^{|I|} (c_i x_i) + d^- - d^+$$

4.5. Planteamiento del problema en PULP

Para resolver el problema con la librería PULP de Python, comenzaremos con una serie de pruebas de cordura:

```
def resolver_problema(df, **kwargs):
    meta_presupuesto = kwargs["presupuesto"]
    max_propiedades = kwargs["propiedades"]
    min_capacidad = kwargs["capacidad"]
    min_calificacion = kwargs["calificacion"]
    rho = kwargs["penalizacion_exceso_presupuesto"]
    nombre = kwargs["nombre"]

    candidatos = df[ df["review_scores_rating"] >= min_calificacion ].copy()

    if nombre == "Exigente":
        for una_columna in columnas_detalles_calificacion:
            candidatos = candidatos[ candidatos[una_columna] >= min_calificacion ]

    candidatos = candidatos.reset_index(drop=True)

    if len(candidatos) == 0:
        print("Nos quedamos sin inmuebles, no podemos proceder")
        return None
    elif candidatos["accommodates"].sum() < min_capacidad:
        print("Nos quedamos sin suficientes inmuebles para alojar a todos")
        return None
```

En esta función, luego de desempaquetar nuestros parámetros recibidos, aplicamos inmediatamente nuestra **restricción de calidad,** la cual podemos implementar como un simple filtro de PANDAS por ser una restricción que impera única y exclusivamente sobre la calificación de nuestros inmuebles. Adicionalmente, nuestro problema contempla un escenario adicional que es el *escenario exigente,* en el cual nuestra restricción de calificación aplica a todos los componentes de la calificación y no sólo al promedio final; por lo que, si estamos en dicho escenario, aplicamos ese filtrado adicional.

Posteriormente, hacemos unas pruebas rápidas de cordura: revisamos si todavía quedan inmuebles después de filtrarlos, y si la capacidad de todo el universo de inmuebles tiene todavía cupo para todos. Dichas pruebas de cordura nos permitirán determinar *a priori* si nuestro problema es inviable con los datos que tenemos.

Una vez implementado este pre-procesamiento, podemos pasar a plantear el problema, comenzando con nuestras variables:

```
I = range(len(candidatos))
x = pulp.LpVariable.dicts("x", I, cat="Binary")
c = candidatos["price"].values
k = candidatos["accommodates"].values
q = candidatos["review_scores_rating"].values
exceso = pulp.LpVariable("dmas", lowBound=0, cat="Continuous")
deficit = pulp.LpVariable("dmenos", lowBound=0, cat="Continuous")
rho = 1
```

A continuación, definimos nuestra función objetivo.

```
modelo = pulp.LpProblem("Tarea 4 - Escenario {nombre}", pulp.LpMinimize)
modelo += ( pulp.lpSum( c[i]*x[i] for i in I ) + (rho * exceso), "Funcion objetivo" )
```

Y luego definimos nuestras restricciones, incluyendo nnuestra restricción suave.

```
modelo += ( pulp.lpSum( k[i]*x[i] for i in I) >= min_capacidad, "Capacidad minima" )
modelo += ( pulp.lpSum( x[i] for i in I) >= max_propiedades, "Limite de reservaciones" )
modelo += ( pulp.lpSum( c[i]*x[i] for i in I) + deficit - exceso == meta_presupuesto,
            "Presupuesto")
```

Una vez definido lo anterior, podemos proceder a resolver nuestro modelo.

```
modelo.solve(pulp.PULP_CBC_CMD(msg=0))
```

A continuación, revisamos que el modelo haya sido resuelto, y en su caso, extraemos los resultados.

```
status = pulp.LpStatus[modelo.status]
print(f"Problema resuelto, status: {status}")
if status != "Optimal":
    print("No se encontro solucion optima")
    return None
```

5. Solución del problema

6. Resultados encontrados en el conjunto de datos

7. Análisis de sensibilidad

7.1. Rangos de factibilidad

7.2. Rangos de optimalidad

7.3. Precios sombra

8. Diferentes escenarios

8.1. Escenario austero

8.2. Escenario de influencer

8.3. Escenario exigente

8.4. Escenario familiar

8.5. Escenario de cercanía

Código Python

```
#!/usr/bin/env python3

# Adjuntar aquí el código de Python usado para resolver el ejercicio
```

Referencias

Airbnb (2026). Inside airbnb: Obtenga los datos. https://insideairbnb.com/get-the-data/. Consultado el 26 de Febrero de 2026.