Universidad de la República

Facultad de Ingeniería

Fundamentos de Optimización

Puntos de Fekete

Autores:
Federico Bello
Julieta Umpierrez

28 de septiembre de 2025

UNIVERSIDAD
DE LA REPÚBLICA
URUGUAY

Índice

1. Introducción al problema

El problema detrás de los algoritmos que se presentan a continuación esta en el marco de la resolución del Problema de Fekete que consiste en encontrar la distribución de N puntos en una esfera de manera de minimizar algún tipo de energía. En este caso se toma la energía logarítmica. Como se explica en [1] este problema tiene gran relevancia en varias áreas que van desde posición satelital y mapeos terrestres a geometría computacional.

2. Parte 1

Para denotar el gradiente de E respecto a x_k se utilizará la siguiente notación: $\nabla_{x_k} E$. El calculo se detalla a continuación:

$$\nabla_{x_k} E = \frac{\partial E}{\partial x_k} = -\frac{\partial}{\partial x_k} \sum_{i=1}^{N} \sum_{j=1, j\neq i}^{N} log\left(||x_i - x_j||_2\right) \overbrace{=}^{(1)} -2 \sum_{j=1, j\neq k}^{N} \frac{\partial log\left(||x_k - x_j||_2\right)}{\partial x_k}$$

$$\overbrace{=}^{(2)} -2 \sum_{j=1, j\neq k}^{N} \frac{1}{||x_k - x_j||_2} \frac{\partial ||x_k - x_j||_2}{\partial x_k} \overbrace{=}^{(3)} -2 \sum_{j=1, j\neq k}^{N} \frac{x_k - x_j}{||x_k - x_j||_2^2}$$

Donde (1) es tomando en cuenta que para un x_k este se va a dar tanto en x_i como en x_j no simultáneamente, por lo que sobrevive ese termino dos veces, de ahí el 2 multiplicando. (2) es aplicando regla de la cadena y derivando el logaritmo y (3) es aplicando $\frac{\partial}{\partial \mathbf{x}} ||\mathbf{x} - \mathbf{a}|_2 = \frac{\mathbf{x}-\mathbf{a}}{||\mathbf{x}-\mathbf{a}||_2}$ extraída de [2].

La implementación de este gradiente se encuentra en el *notebook* en la función *grad_i*. En la función *grad* se tiene la implementación del gradiente respecto a toda la configuración de puntos, la cual consiste en simplemente calcular el gradiente respecto a cada punto por separado.

3. Parte 3

Se comenzó implementando un descenso por gradiente proyectado (PGD), según el algoritmo 1. El gradiente se calcula siguiendo el resultado anterior por lo tanto, si la matriz $\mathbf{X}$ de puntos es de tamaño Nx3, donde N es la cantidad de puntos, el gradiente tendrá el mismo tamaño, en el cual en la i-ésima fila esta el gradiente respecto al i-ésimo punto. Mientras tanto, para proyectar a la esfera unidad alcanza con dividir cada punto por su respectiva norma.

Algorithm 1 Descenso por Gradiente Proyectado

Require: α el valor del paso y tol la tolerancia

Sea $\mathbf{X_0}$ la matriz inicial de puntos, $p(\mathbf{X})$ la proyección de los puntos a la superficie de interés y $g(\mathbf{X})$ el gradiente para cada punto de la función a optimizar.

$\mathbf{X} \leftarrow \mathbf{X_0}$

do

$\mathbf{X^{ant}} \leftarrow \mathbf{X}$

$\mathbf{X} \leftarrow \mathbf{X} - \alpha g(\mathbf{x})$

$\mathbf{X} \leftarrow p(\mathbf{X})$

while $||\mathbf{X} - \mathbf{X^{ant}}||_2 > tol$

return X

4. Parte 2

Para esta segunda parte se siguió el algoritmo especificado en 2. La idea del mismo se basa en PGD, solo que en lugar de calcular el gradiente para todos los puntos en cada iteración se utiliza el gradiente respecto a un solo punto. La idea es simple: elegir un punto cualquiera y encontrar la ubicación óptima para ese punto y esa distribución, luego, repetir para cada punto. A esta repetición se le llamara *época* y al método se le denominara PGDP. Se termina el algoritmo una vez que todos los puntos son óptimos. Por lo tanto, la condición de parada en cada caso sale natural de la sección anterior, donde la iteración por punto termina una vez que el punto se mueve *poco* (por lo que el gradiente es cercano a 0) y la iteración global una vez que todos los puntos en una determinada época se movieron *poco*.

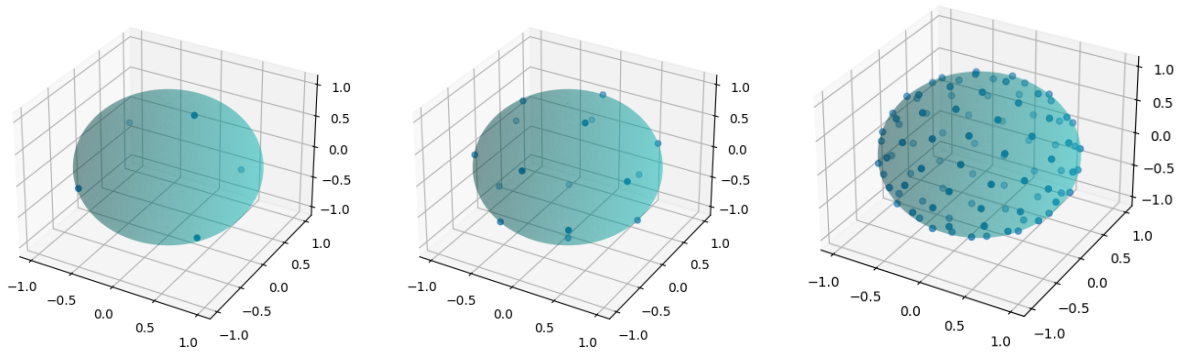
Algorithm 2 Descenso por Gradiente por Punto Proyectado

Require: α el valor del paso y tol la tolerancia
  Sea $\mathbf{X_0}$ la matriz inicial de puntos, $p_i(\mathbf{x})$ la proyección a la superficie de interés del punto $\mathbf{x}$ y $g_i(\mathbf{x})$ el gradiente según el punto $\mathbf{x}$.
  $\mathbf{X} \leftarrow \mathbf{X_0}$
  do
    $\mathbf{X^{ant}} \leftarrow \mathbf{X}$
    for cada punto de $\mathbf{X}$ **do**
      do
        Sea $\mathbf{x_i}$ ese punto
        $\mathbf{x_i^{ant}} \leftarrow \mathbf{x_i}$
        $\mathbf{x_i} \leftarrow \mathbf{x_i} - \alpha g_i(\mathbf{x})$
        $\mathbf{x_i} \leftarrow p_i(\mathbf{x_i})$
      while $||\mathbf{x_i} - \mathbf{x_i^{ant}}||_2 > tol$
      Actualizar $\mathbf{X}$ con el nuevo punto
    end for
  while $||\mathbf{X} - \mathbf{X^{ant}}||_2 > tol$
  return X

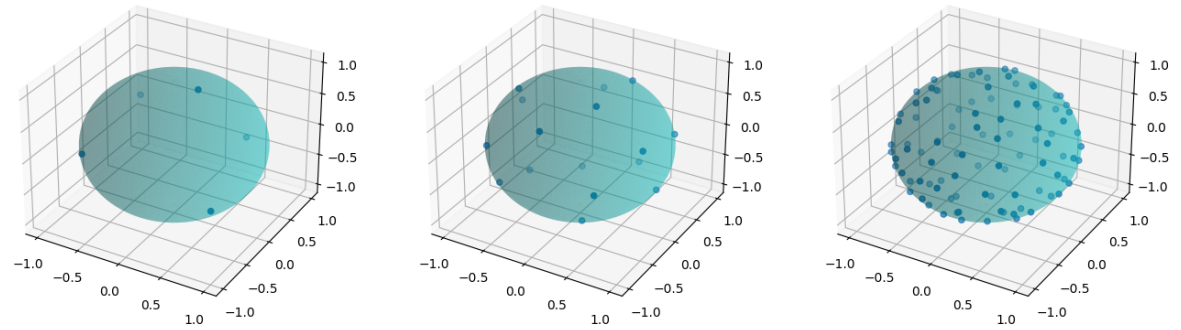
Notar que este algoritmo tiene una clara ventaja y una clara desventaja con respecto al anterior. La ventaja es que en cada iteración solo es necesario calcular el gradiente respecto a un punto, por lo que cada iteración es menos costosa computacionalmente. Sin embargo, parece razonable pensar que la cantidad de iteraciones sera ampliamente mayor respecto al método anterior, dado que solamente se mueve un punto por vez. Las virtudes y los defectos de cada algoritmo se explorarán en la siguiente sección.

5. Parte 4

Previo a realizar la comparación entre ambos métodos de forma cuantitativa, observar la imagen 1. En la misma, se observan las configuraciones resultantes para distinta cantidad de puntos luego de aplicar cada algoritmo propuesto. En ambos casos se utilizaron las mismas condiciones iniciales, para así poder comparar mas justamente los métodos. Este comentario aplica también para el resto del informe. Notar como, a primera vista, ambos métodos parecen resultar en configuraciones similares, indicando que ambas llegaron a un óptimo similar. Sin embargo, si se observan diferencias, por lo que esto abre paso a cuestionarse cuan grande es dicha diferencia. ¿Es significativa? ¿O ambos algoritmos llegan a un igual valor de la energía?



(a) PGD para N=5 (b) PGD para N=16 (c) PGD para N=100



(d) PGDP para N=5 (e) PGDP para N=16 (f) PGDP para N=100

Figura 1: Configuración de puntos resultantes para los distintos métodos y cantidad de puntos

Las tablas 1 y 2 contienen de forma resumida los resultados de 100 experimentos, tomando un paso de 10 y una tolerancia de 1×10^{-2} para ambos métodos. Para conseguir dichos valores simplemente se realizaron los experimentos con distintas condiciones iniciales, para luego promediar cada característica. De la misma se pueden extraer algunas observaciones interesantes.

Viendo primero el lado positivo, se observa como ambos métodos llegan aproximadamente al mismo resultado, indicando que están logrando encontrar un mínimo. Ahora, para analizar las diferencias, primero observar como la cantidad de iteraciones es similar a la cantidad de épocas, independientemente de la cantidad de puntos. Sin embargo, el tiempo de demora no es similar, siendo mayor en el caso de PGDP por sobre PGD. En particular, esta diferencia se hace mas notoria a medida que se aumentan la cantidad de puntos, indicando cierta proporcionalidad con este parámetro.

#{Puntos} / Característica	5	16	100	1000
Épocas	14	36	56	56
Tiempo	9,0ms	40ms	0,41s	7,83s
Energía	-4.42	-36.1	-1083.1	-98323.9

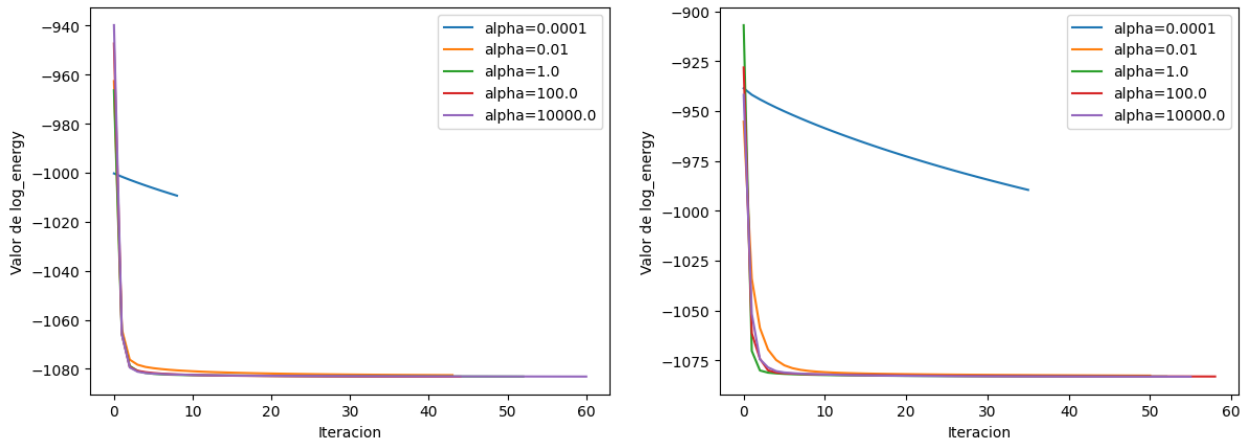
Cuadro 1: Desempeño de PGDP para distintas cantidades de puntos

Característica \ #{Puntos}	5	16	100	1000
Iteraciones	14	37	50	53
Tiempo	2,7ms	17ms	0,19s	5,1s
Energía	-4.42	-36.1	-1083.1	-98323.2

Cuadro 2: Desempeño de PGD para distintas cantidades de puntos

Además, si se compara la cantidad de iteraciones del algoritmo PGD contra la cantidad de épocas del algoritmo PGDP se ve que son casi idénticas, por lo que la diferencia de puntos no yace en la cantidad de veces que se actualiza y compara la matriz de puntos. Haciendo un análisis de la cantidad de veces que es necesario calcular el gradiente según cada punto, parece claro ver que en el caso de PGD sera igual a $\#\{iteraciones\} \cdot \#\{puntos\}$, aunque para el caso de PGDP es un poco mas sutil, ya que depende de cuantas veces se repita el movimiento de cada punto dentro de cada época. Viéndolo empíricamente, se llega a que se calcula el gradiente [100, 730, 6031, 57497] veces para [5,16,100,1000] puntos respectivamente, siendo considerablemente mayor a las $[14 \cdot 5, 37 \cdot 16, 50 \cdot 100, 53 \cdot 1000] = [70, 592{,}5000, 53000]$ veces que se calculan en el caso de PGD.

Por ultimo, en la figura 2 se ve la evolución del valor de la función objetivo en función de las iteraciones. La conclusión parece bastante directa y similar para ambas: un paso muy chico consigue estancarse, no logrando modificar lo suficiente la matriz en cierta iteración o época. Un valor demasiado grande, enlentece la convergencia, eventualmente pudiéndola impedir. Como siempre, un tamaño razonable del paso no solo asegura la convergencia sino que además la acelera.



(a) Método PGDP (b) Método PGD

Figura 2: Distintos tamaños de paso para cada método

6. Conclusiones

Se exploro el Problema de Fekete mediante distintas variantes de descenso por gradiente, en particular haciendo uso de descenso por gradiente proyectado. Se vio que ambos métodos explorados llegan al mismo resultado en cuanto valor de la función objetivo se refiere. Sin embargo, calcular el gradiente respecto a toda la configuración de puntos y mover acorde todos los puntos simultáneamente logra una convergencia de forma mas directa, tanto en tiempo como en cantidad de iteraciones.

Referencias

[1] Diego Armentano. Problema de fekete. *CMAT*, 2018.

[2] K. B. Petersen and M. S. Pedersen. The matrix cookbook, October 2008. Version 20081110.