Publicaciones Matemáticas del Uruguay
Volumen 17, Julio 2019, Páginas 1–21
S 0797-1443

UN PASEO POR LA OPTIMIZACIÓN

MARCELO FIORI

1. Introducción

En general, un problema de optimización consiste en minimizar una función en cierto conjunto, y lo escribimos como:

$$\min_{x \in \mathcal{X}} f(x).$$

Muchas veces lo que importa, más que el valor mínimo, es dónde se alcanza el mismo. El problema de encontrar el o los puntos donde se minimiza la función, se expresa así:

$$\arg\min_{x \in \mathcal{X}} f(x).$$

La gran variabilidad de funciones f, conjuntos $\mathcal{X}$, y estructuras posibles, dan lugar a una gran cantidad de aplicaciones, preguntas interesantes, métodos, etc. Por ejemplo, f puede ser una función convexa o no, diferenciable o no, $\mathcal{X}$ puede ser un subconjunto convexo de $\mathbb{R}^n$ o no, puede ser una variedad, puede ser un espacio de dimensión finita o no, etc.

Hay muchos ejemplos que ya son conocidos por todos los que pasaron por algun curso universitario de matemática. Por citar dos nada más: el problema de mínimos cuadrados y el de proyección de un punto a un conjunto dado (es decir, encontrar el punto del conjunto que minimiza la distancia al punto dado).

A lo largo de este mini curso intentaremos mostrar algunas aplicaciones o conexiones con diferentes áreas (geometría, álgebra, estadística, física, machine learning), algunos aspectos teóricos, algorítmicos, y algunas interpretaciones interesantes.

Una parte muy importante de la optimización es que podamos resolver el problema (sea lo que sea que esto quiere decir) en un tiempo *razonable* en una computadora. Durante mucho tiempo, el límite entre los problemas de optimización que se consideraban "realizables" y los que no, era la linealidad. Es decir, que la función f fuera lineal, y el conjunto $\mathcal{X}$ fuera dado por igualdades y desigualdades de funciones lineales. Hoy se considera que este límite es la convexidad (que f sea una función convexa, y $\mathcal{X}$ un conjunto convexo). Esto significa que hay mucho interés en algoritmos para problemas convexos (porque son los que se pueden resolver), pero también hay mucho interés en problemas no convexos (porque son los que "no se pueden resolver", y son por lo tanto un interesante desafío).

2. Motivación y ejemplos

Los siguientes ejemplos no son todos completamente independientes entre sí, y no están pensados para leer necesariamente en este orden, ni en este lugar. Es decir, se pueden saltear todos los ejemplos y pasar directamente a la sección 3, se puede

leer linealmente, o se puede ir alternando entre secciones de ejemplos y el resto de las secciones[1].

2.1. El problema de Thomson (o el séptimo problema de Smale). Consideremos N electrones en la esfera S^2, llamemos x_i al vector de $\mathbb{R}^3$ con las coordenadas del electrón número i, y consideremos todas las posiciones juntas en $x = (x_1, x_2, \dots, x_N)$. La energía de la interacción entre los electrones i y j (con cargas iguales $e_i = e_j = e$) es $U_{ij} = \frac{k_e e_i e_j}{\|x_i - x_j\|}$, donde k_e es la constante de Coulomb. Entonces la energía potencial total del sistema, simplificando las unidades para eliminar las constantes, es la siguiente:

$$U(X) = \sum_{i<j} \frac{1}{\|x_i - x_j\|}.$$

El problema consiste en encontrar configuraciones de energía mínima. Es decir, encontrar las posiciones x_i que minimicen la función U:

$$\arg \min_{s.t. \|x_i\|=1 \forall i} U(x),$$

donde la restricción $\|x_i\| = 1$ impone que los electrones se encuentren en la esfera.

Las configuraciones de mínima energía han sido estudiadas en profundidad solamente para algunos pocos valores de N. Por ejemplo, si tenemos dos electones, la solución es trivial, y consiste en posiciones diametralmente opuestas, que dan lugar a una energía de $\frac{1}{2}$. Para $N = 3$, los electrones se encuentran en los vértices de un triángulo equilátero en una circunferencia maximal. Para $N = 4$, los electrones forman un tetraedro regular. Sin embargo el problema todavía es de mucho interés para muchos valores de N, comportamiento asintótico, o algoritmos para hallar configuraciones mínimas.

Este problema es un caso particular del denominado séptimo problema de Smale [22], cuya motivación es encontrar buenos puntos de partida (raíces de polinomios) para algoritmos de homotopía que busquen todas las raíces de un polinomio dado en $\mathbb{C}$. Muy resumidamente, dado un polinomio en $\mathbb{C}$, un algoritmo de homotopía parte de un polinomio con raíces conocidas, y construye un camino desde el polinomio conocido al polinomio objetivo, intentando seguir el camino que recorren las raíces. Para esto, es necesario que las raíces del polinomio inicial estén tan alejadas entre sí como sea posible, y por eso se buscan configuraciones que minimicen $U(x)$.

Con las herramientas básicas descritas en este texto (específicamente en la sección 4.2), se puede programar muy fácilmente un algoritmo para encontrar soluciones aceptables para valores moderados de N.

2.2. Problema de valores propios. Uno de los problemas más fáciles de entender, pero a la vez más complejos, es el problema de encontrar algoritmos para hallar valores y vectores propios.

Sea A una matriz simétrica real, a la que le queremos hallar sus valores y vectores propios. Es decir, buscamos una matriz U ortogonal ($U \in \mathcal{O}(n)$) y una matriz D diagonal tales que $A = UDU^T$. Si bien existen algoritmos muy utilizados como el método QR, que no están inspirados explícitamente en un problema de optimización, vamos a describir aquí otros dos enfoques que sí lo están.

[1]Un orden posible, al estilo Rayuela, puede ser 1, 2.1, 3, 2.4, 4, 2.2, 5, 2.3, 6, 2.5.

Ordenemos los valores propios de A: $\lambda_1 \geqslant \lambda_2 \geqslant \ldots \geqslant \lambda_n$, y sean $v_1, \ldots, v_n$ los vectores propios asociados. Comencemos definiendo el cociente de Rayleigh de A, que es la función $r_A : \mathbb{R}^n \setminus \{0\} \to \mathbb{R}$ dada por

$$r_A(x) = \frac{x^T Ax}{x^T x} = \frac{\langle Ax, x\rangle}{\langle x, x\rangle}.$$

Observar que r_A es invariante por cambios de escala, o sea, $r_A(tx) = r_A(x)$, $\forall t \in \mathbb{R} \setminus \{0\}$, por lo que basta estudiar la función en $S^{n-1} = \{x \in \mathbb{R}^n : \|x\| = 1\}$. En este caso, podemos obviar la normalización del denominador, por lo que la función r_A básicamente toma un vector de norma uno, y mide "cuán colineal" es Ax con x (y cuánto expande A en esa dirección). El siguiente resultado captura esta idea:

Teorema (Courant-Fisher)**.** Se tiene que

$$\lambda_1 = \max_{\|x\|=1} r_A(x), \quad \lambda_n = \min_{\|x\|=1} r_A(x).$$

Además, los puntos críticos de r_A en S^{n-1} son los vectores propios v_i de A.

De esta manera, formulamos el problema de hallar valores propios como un problema de optimización (por lo menos los valores propios más grande y más chico, aunque se puede extender esta idea fácilmente).

Veamos otra forma de atacar el problema. Como tenemos (o buscamos tener) $A = UDU^T$, podemos pensar equivalentemente que buscamos una matriz ortogonal U tal que $U^T AU$ sea diagonal. Una manera de expresar esto, es intentando minimizar los elementos fuera de la diagonal de $U^T AU$. Si denotamos como $diag(M)$ a la matriz que tiene solamente los elementos en la diagonal de M, entonces buscamos

$$min_{U \in \mathcal{O}(n)} \|diag(U^T AU) - U^T AU\|_F^2.$$

Esta función suma los cuadrados de las entradas fuera de la diagonal de $U^T AU$. Como A es simétrica, sabemos que tiene que haber una matriz U que haga que $U^T AU$ sea diagonal, y por lo tanto el mínimo de la función objetivo será cero, y se alcanzará en una matriz ortogonal que diagonaliza A.

En realidad, se puede ver que si consideramos una matriz diagonal N cualquiera, el problema

$$min_{U \in \mathcal{O}(n)} \|N - U^T AU\|_F^2$$

también tiene como soluciones las matrices U que diagonalizan A. Más aún, si N tiene los elementos ordenados (por ejemplo $N = diag(1, 2, 3, \ldots, n)$), entonces el punto crítico correspondiente a los valores propios ordenados de menor a mayor, es un atractor.

Tenemos entonces otras dos formulaciones en términos de problemas optimización. Sobre los métodos para resolverlos, se comenta brevemente en algunos pasajes de estas notas.

2.3. Problemas de isomorfismos de grafos. Los que siguen son otros problemas simples de enunciar, pero de los que todavía queda mucho por entender. Hace muy poco, László Babai revolucionó a la comunidad con un resultado sobre el problema de isomorfismo de grafos [4], pero aún quedan muchas preguntas planteadas. En esta sección describiremos los problemas, formularemos un problema de optimización equivalente, y comentaremos algunos resultados recientes.

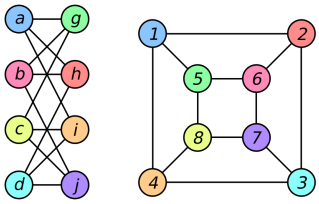
Definición. Un grafo $G = (V, E)$ consiste en un conjunto $V = \{v_1, v_2, \ldots, v_n\}$ de vértices y un conjunto de aristas E, que son pares no ordenados de V.

Intuitivamente, un grafo es un conjunto de puntos unidos por aristas, por lo que hay muchas formas de "dibujarlo". En particular, cualquier re-ordenación de los nodos, da lugar al mismo dibujo. El concepto de isomorfismo captura este concepto:

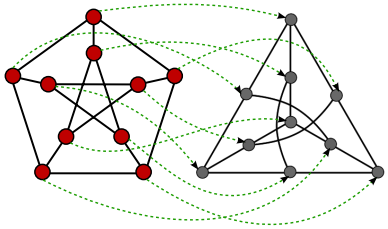
Definición. Decimos que dos grafos $G_1 = (V_1, E_1)$ y $G_2 = (V_2, E_2)$ son isomorfos sii existe una biyección $\varphi : V_1 \to V_2$ que preserve la estructura. Esto es, que $(v_i, v_j) \in E_1 \iff (\varphi(v_i), \varphi(v_j)) \in E_2$.

Un isomorfismo de un grafo en sí mismo se denomina automorfismo, y el conjunto de automorfismos de un grafo forma un grupo con la composición.

Ejemplo. *Los grafos de la siguiente figura son isomorfos, siendo φ la que asigna a un vértice de un grafo, el vértice del mismo color en el otro grafo.*



Ejemplo. *Los grafos de la siguiente figura son isomorfos, con la biyección representada con las flechas punteadas.*

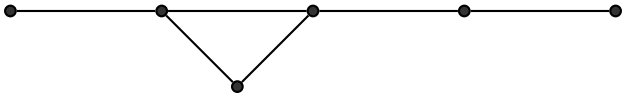


El problema de isomorfismo de grafos (GIP por sus siglas en inglés) consiste en, dados dos grafos, decidir si son isomorfos o no. Computacionalmente, es un problema difícil, cuya complejidad se encuentra estrictamente entre P y NP (a no ser que estas clases colapsen).

Otro problema relacionado es el de automorfismo de grafos, que consiste determinar si el grupo de automorfismos de un grafo es trivial o no.

Observar que los dos problemas presentados arriba busca respuestas del tipo *Sí o no*, pero no busca encontrar elementos (isomorfismos o automorfismos) explícitamente.

Ejemplo. *El grafo de la figura tiene grupo de automorfismo trivial.*



Dado un grafo $G = (V, E)$, definimos su matriz de adyacencia como $A \in \mathrm{Sym}(n)$

$$A_{ij} = \begin{cases} 1 & \text{si } (v_i, v_j) \in E \\ 0 & \text{si no} \end{cases}$$

Comentario
Si A y B son matrices de adyacencia de dos grafos isomorfos, entones $A = PBP^T$ para alguna matrix de permutación $P \in \mathcal{P}(n)$.

En términos de las matrices de adyacencia, el problema de isomorfismo de grafos es: ¿$\exists P \in \mathcal{P}(n) / A = PBP^T$?

Por otro lado, el denominado Graph Matching Problem (GMP) consiste en encontrar el isomorfismo entre dos grafos (o el "más cercano" en algún sentido, si no son isomorfos). Esto es:

$$P^* = \arg \min_{P \in \mathcal{P}(n)} \|A - PBP^T\|_F^2,$$

o lo que es lo mismo (multiplicando por P a derecha, que no modifica la norma),

$$P^* = \arg \min_{P \in \mathcal{P}(n)} \|AP - PB\|_F^2, \tag{1}$$

El problema es que $\mathcal{P}(n)$ es un conjunto discreto con $n!$ elementos, por lo que computacionalmente no es viable probar todos los elementos del conjunto (que son todas las formas de re-ordenar los nodos) para encontrar la solución.

Un camino posible consiste en relajar el dominio, pasando del conjunto de permutaciones $\mathcal{P}(n)$ a su envolvente convexa, que resulta ser (teorema de Birkhoff-von Neumann) el conjunto de matrices de entradas no negativas cuyas filas y columnas suman uno. Estas matrices se denominan *doblemente estocásticas*, y denotaremos a este conjunto como $\mathcal{D}$. Es decir: $\mathcal{D}(n) = \{M \in \mathbb{R}^{n \times n} : M\mathbf{1} = \mathbf{1}, M^T\mathbf{1} = \mathbf{1}, M_{ij} \geqslant 0 \, \forall i, j\}$.

El problema resultante es entonces

$$\hat{P} = \arg \min_{P \in \mathcal{D}(n)} \|AP - PB\|_F^2, \tag{2}$$

que es ahora un problema convexo, y por lo tanto lo podemos resolver (por ejemplo, con técnicas de 4.2). Por supuesto que la solución $\hat{P}$ de este problema no necesariamente será solución de (1), ya que agrandamos el conjunto. Una opción es resolver (2), y luego buscar la matriz de permutación más cercana a $\hat{P}$ (que se puede hacer en tiempo polinomial), pero en principio no hay garantías de que esta matriz de permutación corresponda a la solución del problema original. Hay también otros enfoques posibles [14, 17].

Una pregunta interesante es bajo qué condiciones la solución del problema (2) que *podemos* resolver, coincide con la solución del problema que *queremos* resolver. Algunos resultados parciales se pueden encontrar en [2, 16], y resultados asintóticos en [19].

En particular, algunos de estos resultados, que vienen de estudiar en qué casos dos problemas de optimización son equivalentes, dan lugar a resultados puramente algebraicos, como por ejemplo que si el espectro de la matriz de adyacencia de un grafo tiene ciertas propiedades, entonces el grupo de automorfismos es trivial.

2.4. Machine Learning. Uno de los campos que está más de moda, donde la optimización es fundamental, es el aprendizaje automático (Machine Learning, o Inteligencia Artificial, son también denominaciones de conceptos muy similares). Dicho muy breve e incompletamente, consiste en hacer algoritmos para poder tomar decisiones de manera automática, generalmente basados en datos.

Ejemplos de aplicaciones de estos métodos se ven todos los días: sistemas de recomendación (por ejemplo los de Netflix o Spotify), etiquetado automático de imágenes, transcripción automática de música, traducción entre idiomas en tiempo real, o el AlphaGo (que logró un nivel superior al humano en el ancestral juego chino Go), son algunos de los miles de ejemplos.

Quizás el problema más sencillo de comprender es el de clasificación. Supongamos que tenemos elementos que queremos separar en dos clases, por ejemplo peras de manzanas[2], y para ello contamos con medidas de los elementos (digamos, ancho y alto). A partir de ver solamente las medidas, queremos decidir si se trata de una pera o de una manzana. Para ello, en general se cuenta con cierta cantidad de datos de ejemplo, para poder entrenar al clasificador.

Es decir, tenemos dos conjuntos de puntos $(x_1, x_2, \ldots, x_N)$ y $(y_1, y_2, \ldots, y_M)$, distribuidos en $\mathbb{R}^n$, y buscamos una función $f : \mathbb{R}^n \to \mathbb{R}$ tal que f sea negativa para una clase, y positiva para la otra:

$$f(x_i) > 0, \quad i = 1, \ldots, N, \qquad\qquad f(y_i) < 0, \quad i = 1, \ldots, M.$$

Un primer paso, que da lugar luego a uno de los clasificadores más usados (SVM), consiste en intentar separar linealmente los conjuntos. Es decir, buscamos un hiperplano que deje al conjunto de los x_i de un lado, y al conjunto de los y_i del otro. Por supuesto que esto no siempre es posible, y para esto están las modificaciones y variaciones que no mencionaremos en esta oportunidad.

Si los conjuntos son linealmente separables, en general hay muchos hiperplanos que cumplen la función. Pero el problema de clasificación consiste en aprender el hiperplano, para luego clasificar nuevas instancias que llegarán. Por lo tanto no todos los hiperplanos son igualmente buenos.

Buscamos entonces el hiperplano $a^T x + b$ que ofrezca el mayor margen de separación posible, para que cuando llegue una nueva instancia, sea más probable clasificarla correctamente. Una manera de plantear esto es mediante la variable auxiliar t, que mide justamente el margen (en realidad el margen es $2t$, ver figura):

$$\begin{array}{ll} \text{maximizar} & t \\ \text{s.t.} & a^T x_i + b \geqslant t, i = 1, \ldots, N \\ & a^T y_i + b \leqslant -t, i = 1, \ldots, N \\ & \|a\|_2 \leqslant 1 \end{array}$$

Es decir, maximizar el margen de separación, con las restricciones de separar los conjuntos de puntos. La restricción sobre la norma de a es porque sino se podría escalar todo por una constante y obtener márgenes arbitrariamente grandes. Este problema es convexo, y por lo tanto se puede resolver eficientemente. Se puede ver una discusión más amplia sobre esto en [8].

[2]Puede ser también correo spam de no-spam, usos fraudulentos de tarjetas de crédito de usos normales, imágenes, etc.

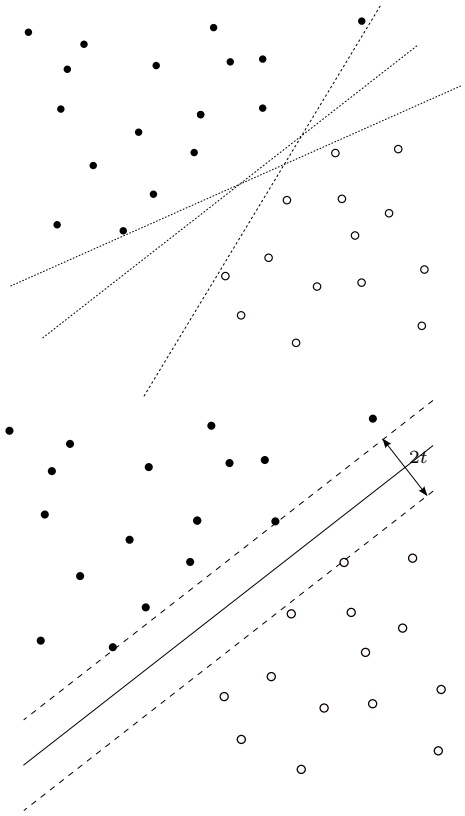


FIGURA 1. Conjuntos de puntos linealmente separables. A a izquierda, distintos hiperplanos que separan los conjuntos. A la derecha, el hiperplano con mayor margen.

2.4.1. Matrix completion. El problema de *completado de matrices* consiste, como lo indica su nombre, en completar una matriz conociendo solamente algunas de sus entradas. Por supuesto que este problema, así enunciado, no tiene solución (o mejor dicho, tiene infinitas soluciones), a no ser que asumamos alguna estructura sobre la matriz. Generalmente se denomina como *matrix completion* al problema de encontrar la matriz de menor rango, que complete las entradas conocidas. Este modelo responde a varios problemas importantes, de los cuales mencionaremos un par de ejemplos.

El primero es quizás el que hizo famoso el problema de matrix completion, y es el denominado problema de Netflix, que cobró relevancia en medios de prensa especializados porque ofrecía un premio de un millón de dólares. Se pretendía mejorar el sistema de recomendación de Netflix. La matriz sintetiza la información sobre los gustos de los usuarios. Cada fila de la matriz corresponde a un usuario, y cada columna a una película. En la entrada (i, j), si el usuario i vio la película j, se encuentra el puntaje que le asignó. Es por lo tanto una matriz que tiene muy pocos elementos conocidos (pues cada usuario ve relativamente pocas películas), pero de la que es deseable inferir el resto de las entradas, porque de esta manera, el proveedor del servicio puede saber si recomendar o no determinada película a determinado

usuario. ¿Por qué esta matriz debería ser de bajo rango? Porque se cree que son algunos pocos factores los que contribuyen al gusto de cada persona. Desde otro punto de vista, también se puede pensar que cada usuario es (aproximadamente) combinación lineal de otros pocos usuarios.

Para el otro ejemplo, supongamos que tenemos una red de sensores inalámbricos, distribuidos aleatoriamente en alguna región, y cada sensor puede estimar la distancia a sus vecinos más cercanos, basándose en la potencia de la señal. Con estos datos, formamos una matriz (incompleta) de distancias, y nos gustaría reconstruir la geometría de la red de sensores (o sea, el resto de la matriz). Si los sensores están ubicados en un plano, entonces esta matriz tiene rango dos (y rango tres si se encuentran en el espacio tridimensional).

Volviendo a la formulación del problema, si denotamos por Ω al conjunto de índices para los cuales se conocen las entradas de la matriz, y por M_{ij} a dichas entradas, entonces el problema es:

$$\begin{array}{ll} \min\limits_{X\in\mathbb{R}^{n\times n}} & rank(X) \\ \text{s.t.} & X_{ij} = M_{ij} \quad \forall (i,j) \in \Omega. \end{array} \tag{3}$$

Lamentablemente este problema es NP-hard (y no solo eso, sino que además todos los algoritmos conocidos requieren tiempo doblemente exponencial, tanto en teoría como en práctica), por lo tanto es necesario buscar alternativas.

La más exitosa surge de observar que el rango de una matriz coincide con la cantidad de valores singulares no nulos. Por lo tanto, el problema (3) es equivalente a minimizar la cantidad de valores singulares no nulos. En vez de hacer esto, una relajación de este problema consiste en considerar la denominada norma nuclear, que es la suma de los valores singulares:

$$\|X\|_* = \sum_{i=1}^{n} \sigma_i(X)$$

de donde el problema relajado es:

$$\begin{array}{ll} \min\limits_{X\in\mathbb{R}^{n\times n}} & \|X\|_* \\ \text{s.t.} & X_{ij} = M_{ij} \quad \forall (i,j) \in \Omega. \end{array} \tag{4}$$

Este problema tiene la ventaja de ser convexo, y por lo tanto se puede resolver eficientemente. La relajación del rango a la norma nuclear está fuertemente relacionada con el contenido de la próxima sección. La pregunta, como en varios de los ejemplos aquí presentados donde aparecen relajaciones de problemas, es cuándo la solución del problema (4) coincide con la solución de (3). Una discusión detallada del problema, y resultados en el sentido de la equivalencia de los problemas, se puede encontrar por ejemplo en [10, 12].

2.4.2. Sparse modeling. Consideremos una matriz de rango completo $A \in \mathbb{R}^{n\times m}$ con $n < m$, y el sistema de ecuaciones indeterminado $Ax = b$. Este sistema tiene infinitas soluciones, y para determinadas aplicaciones, a uno le gustaría elegir alguna de ellas, con cierto criterio, que sea particularmente útil para la aplicación. Una opción, por ejemplo, es elegir la solución de menor norma:

$$\begin{array}{ll} \min\limits_{x\in\mathbb{R}^{m}} & \|x\|_2^2 \\ \text{s.t.} & Ax = b. \end{array} \tag{5}$$

Este problema tiene solución en forma cerrada, que es la conocida $x = A^T(AA^T)^{-1}b$. Sin embargo, supongamos que queremos buscar la solución más "esparsa", la que tiene menor cantidad de elementos no nulos. Esto puede estar justificado de alguna manera, por el deseo de encontrar la solución "más simple", y de hecho en la última década ha habido una cantidad increíble de resultados teóricos y aplicaciones de este problema. Haciendo un poco de abuso de notación (ya que no es formalmente una norma), llamaremos $\|x\|_0$ a la cantidad de elementos no nulos del vector x. Se puede pensar $\|x\|_0$ como el límite de $\|x\|_p^p$ cuando $p \to 0$. El problema entonces lo podemos formular de la siguiente manera:

$$\begin{aligned} \min_{x\in\mathbb{R}^m} \quad & \|x\|_0 \\ \text{s.t.} \quad & Ax = b. \end{aligned} \tag{6}$$

Este problema, sin embargo, resulta ser NP-hard. Una manera de resolver una versión aproximada de (6) (ciertamente no la única), es considerar la norma uno, que es la norma convexa más cercana a $\|x\|_0$. El problema resultante, que es ahora convexo, y por lo tanto se puede resolver, es

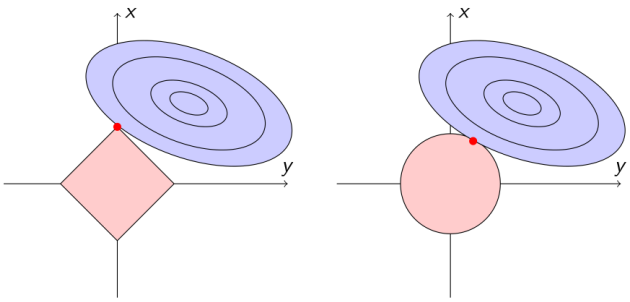
$$\begin{aligned} \min_{x\in\mathbb{R}^m} \quad & \|x\|_1 \\ \text{s.t.} \quad & Ax = b. \end{aligned} \tag{7}$$

Observar que la función $\|x\|_1$ es convexa pero no diferenciable, por lo cual algoritmos que puedan operar en estas condiciones son de particular interés.

La pregunta, como en varios otros pasajes de estas notas, es cuándo la solución del problema (7), que podemos resolver, coincide con la solución del problema (6), que es el que queremos resolver originalmente. Hay varios resultados en este sentido [9, 11]. Daremos aquí simplemente una intuición geométrica. Para esto, consideremos el problema similar:

$$\begin{aligned} \min_{x\in\mathbb{R}^m} \quad & \|Ax - b\|_2^2 \\ \text{s.t.} \quad & \|x\|_1 \leqslant \alpha. \end{aligned} \tag{8}$$

En la siguiente figura, se encuentran las curvas de nivel de $\|Ax-b\|_2^2$, y la restricción, que es simplemente que x pertenezca a la bola de radio α con la norma uno. Al costado, se observa la misma situación pero utilizando la norma dos para la restricción. Se puede obsevar que en el caso de la norma uno, la solución se encuentra en un vértice (y esta propiedad es genérica), por lo que varias de sus coordenadas son cero.



Comentario
Relacionando estos conceptos con el problema de la sección 2.4.1, observar que la

norma nuclear es al rango, lo que la norma uno es a la pseudo-norma cero. Mientras una cuenta elementos no nulos (en el caso del rango, valores singulares no nulos), la otra suma sus valores absolutos.

La cantidad de aplicaciones de estas técnicas es tan grande que no podríamos listarlas aquí. Mostramos solamente una de ellas, para ejemplificar la fortaleza de la formulación, y referimos al lector interesado a otros textos para profundizar [9, 20].

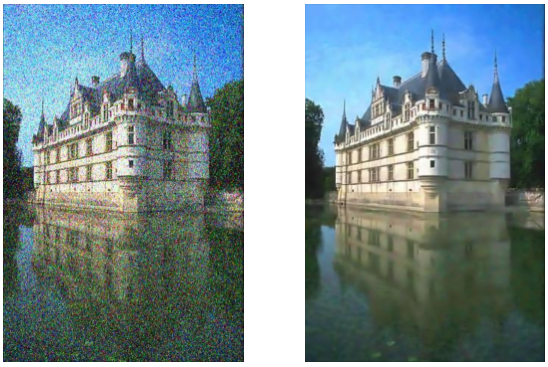


FIGURA 2. Denoising de imágenes utilizando técnicas de sparse modeling. A la izquierda la imagen de entrada, a la derecha el resultado de eliminar el ruido mediante el problema de optimización descrito, y una buena elección de la matriz A.

2.5. Maximum Likelihood. Consideremos una familia de distribuciones de probabilidad en $\mathbb{R}^m$, indexadas por un vector $x \in \mathbb{R}^n$, con densidades $p_x(\cdot)$ (un modelo paramétrico). Cuando las consideramos como función del parámetro x, con un $y \in \mathbb{R}^m$ fijo, a la función $p_x(y)$ se le suele llamar función de verosimilitud (likelihood en inglés), y en general es conveniente trabajar con el logaritmo de la misma (log-likelihood): $l(x) = \log p_x(y)$.

Un problema usual es estimar el valor del parámetro x, basados en una observación y. Uno de los métodos más usados es el llamado de Máxima Verosimilitud (o Maximum Likelihood), que consiste en estimar el parámetro x como:

$$\hat{x}_{ML} = \arg\max_x p_x(y) = \arg\max_x l(x).$$

Intuitivamente, se selecciona el valor del parámetro x que hace los datos más probables.

Cuando por ejemplo la función $l(x)$ es cóncava, el problema resulta convexo. Veamos algunos ejemplos particulares.

2.5.1. Modelos lineales con ruido. Supongamos que tenemos un modelo lineal:

$$y_i = a_i^T x + v_i, \qquad i = 1, \dots m,$$

donde $x \in \mathbb{R}^n$ es el vector de parámetros que queremos estimar, y_i son las medidas con las que contamos, y los v_i representan el ruido o errores de medida, que asumiremos i.i.d. con densidad $p(\cdot)$ en $\mathbb{R}$. La función de verosimilitud $p_x(y)$ es entonces:

$$p_x(y) = \prod_{i=1}^{m} p(y_i - a_i^T x),$$

de donde la función de log-likelihood resulta

$$l(x) = \sum_{i=1}^{m} \log p(y_i - a_i^T x).$$

Si asumimos que el ruido es Gaussiano, $p(z) = \dfrac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{z^2}{2\sigma^2}}$, entonces resulta

$$l(x) = -\frac{m}{2}\log(2\pi\sigma^2) - \frac{1}{2\sigma^2}\|Ax - y\|_2^2,$$

donde A es la matriz formada por los a_i. El estimador de máxima verosimilitud entonces coincide con el problema de mínimos cuadrados, o dicho de otra forma, cuando hacemos mínimos cuadrados estamos asumiendo que el ruido es Gaussiano.

Si el ruido fuera Laplaciano, $p(z) = \frac{1}{2a} e^{-\frac{|z|}{a}}$, con $a > 0$, entonces se puede ver que el estimador de ML consiste en:

$$\hat{x} = \arg\min_x \|Ax - y\|_1,$$

que también es un problema convexo.

2.5.2. Inversa de covarianza para variables Gaussianas. Supongamos que $y = (y_1, \ldots, y_n)$ es una variable aleatoria Gaussiana $y \sim \mathcal{N}(0, \Sigma)$, y sea $\Theta = \Sigma^{-1}$, que se denomina matriz de concentración o matriz de precisión, y también resulta definida positiva, al igual que Σ. El *problema de selección de covarianza* [13] consiste en estimar el patrón de ceros de la matriz de precisión. Este problema es de particular interés para analizar dependencias entre variables, dado que dos coordenadas y_i e y_j son condicionalmente independientes dadas las otras coordenadas, si y solo si $\Theta(i,j) = 0$. Esta propiedad, además, motiva la representación de la estructura de dependencia condicional en términos de un grafo (con aristas entre los nodos i y j cuando $\Theta(i,j) \neq 0$).

Empezemos de todas maneras con Σ como parámetro a estimar. La densidad de y es:

$$p_\Sigma(y) = \frac{1}{\sqrt{(2\pi)^n \det(\Sigma)}} e^{-\frac{y^T \Sigma^{-1} y}{2}}.$$

Dadas N muestras independientes $Y_1, \ldots, Y_N \in \mathbb{R}^n$, queremos estimar Σ. El logaritmo de la función de verosimilitud es entonces

$$\begin{aligned} l(\Sigma) &= \log p_\Sigma(Y_1, \ldots, Y_N) \\ &= -\frac{Nn}{2}\log(2\pi) - \frac{N}{2}\log\det\Sigma - \frac{1}{2}\sum_{i=1}^{N} Y_i^T \Sigma^{-1} Y_i \\ &= -\frac{Nn}{2}\log(2\pi) - \frac{N}{2}\log\det\Sigma - \frac{N}{2}\operatorname{tr}(\Sigma^{-1} S) \end{aligned}$$

donde $S = \frac{1}{N}\sum_{i=1}^{N} Y_i Y_i^T$ es la matriz empírica de covarianza.

Así planteado, el problema de optimizar $l(\Sigma)$ no es convexo, pero sin embargo se puede trabajar un poco para dejar el problema más amigable. Pasemos entonces, ahora sí, a usar la inversa de Σ como parámetro. Resulta

$$l(\Theta) = -\frac{Nn}{2}\log(2\pi) + \frac{N}{2}\log\det\Theta - \frac{N}{2}\operatorname{tr}(\Theta S),$$

por lo que el estimador de máxima verosimilitud es

$$\max_{\Theta \succ 0} \log\det\Theta - \operatorname{tr}(\Theta S).$$

Ahora la función objetivo es cóncava, por lo que el problema resulta convexo (pues queremos maximizar, y no minimizar). Se puede ver que el gradiente de la función objetivo es $\Theta^{-1} - S$, por lo que (utilizando conceptos básicos, que son recordados en 3) cuando no tenemos más información o restricciones sobre Σ, el estimador de máxima verosimilitud de la matriz de covarianza es simplemente $\hat{\Sigma}_{ML} = S$, la covarianza empírica.

Volviendo al problema de selección de covarianza, muchas veces se tiene como información que los elementos no nulos de Θ son relativamente pocos, por lo que, a la hora de estimar la inversa de covarianza, se desea penalizar la cantidad de entradas no nulas. Una manera de hacer esto (como fue mencionado en 2.4.2), consiste en agregar una penalización de norma ℓ_1, quedando el problema:

$$\max_{\Theta>0} \log\det\Theta - \operatorname{tr}(S\Theta) - \lambda\|\Theta\|_1.$$

Este problema sigue siendo convexo, y tiene muchísimas aplicaciones, generalizaciones, e implementaciones eficientes [5, 15].

3. (Algunas) condiciones de optimalidad y subdiferenciales

El objetivo más ambicioso en optimización es conseguir el mínimo global de la función. Esto es posible para algunos casos, pero para muchos otros casos no lo es. Entonces uno se conforma con encontrar un mínimo local, o incluso algún punto que verifique alguna condición relacionada con esto. Veamos algunos ejemplos.

Desde los primeros cursos de cálculo vemos condiciones necesarias de optimalidad. Por ejemplo, para funciones derivables de una variable, si la función tiene un mínimo relativo en x_0, entonces su derivada se anula en ese punto. Por supuesto que esto es solamente una condición necesaria (el contraejemplo más usado es x^3 en el origen), pero muchas veces buscar puntos donde se anule la derivada es lo mejor que podemos hacer.

Esta condición de optimalidad se generaliza de manera directa a funciones diferenciables de $\mathbb{R}^n$ a $\mathbb{R}$, donde resulta que el gradiente debe anularse, pero también tiene generalizaciones para funciones no diferenciables, y para puntos que no son interiores al dominio. Tenemos por ejemplo el siguiente resultado inmediato:

Proposición 1. Sea $f : \mathcal{X} \to \mathbb{R}$ diferenciable, donde $\mathcal{X}$ es un subconjunto convexo de $\mathbb{R}^n$.

a) Si f tiene un mínimo local en x^*, entonces $\langle \nabla f(x^*), x - x^* \rangle \geqslant 0$ para todo $x \in \mathcal{X}$.
b) Si además f es convexa, entonces la condición anterior es suficiente.

La interpretación de la primera condición es clara: dado que el conjunto $\mathcal{X}$ es convexo, las direcciones $(x - x^*)$ son las factibles, es decir, las direcciones hacia las cuales uno se puede mover desde x^*, manteniéndose dentro del conjunto $\mathcal{X}$. El producto interno $\langle \nabla f(x^*), x - x^* \rangle$ no es otra cosa que la derivada direccional de f en la dirección $(x - x^*)$, y por lo tanto la condición se puede leer así: en todas las direcciones que uno se pueda mover desde x^*, la función f *crece*, y por lo tanto x^* tiene que ser un mínimo local.

Si llamamos $P_{\mathcal{X}} : \mathbb{R}^n \to \mathcal{X}$ a la proyección sobre $\mathcal{X}$ ($P_{\mathcal{X}}(z) = \arg\min_{x\in\mathcal{X}} \|z - x\|$), entonces la condición a) es equivalente a que $x^* = P_{\mathcal{X}}(x^* - \alpha\nabla f(x^*))$ para todo $\alpha \geqslant 0$. Esto es: si estando en un mínimo local x^*, nos movemos cualquier distancia α en la dirección de máximo descenso, y proyectamos de nuevo al conjunto $\mathcal{X}$, entonces

volvemos a caer en x^*. Esta interpretación está muy relacionada con la forma de algunos algoritmos iterativos que comentaremos más adelante.

Para poder extender estas condiciones a funciones no diferenciables (y darles una interpretación geométrica), necesitamos una extensión del concepto de diferencial. Haremos esto para funciones convexas:

Definición. Dada una función $f : \mathbb{R}^n \to \mathbb{R}$ convexa y $x \in \mathbb{R}^n$, decimos que $d \in \mathbb{R}^n$ es un subgradiente de f en x sii $f(z) \geqslant f(x) + \langle z - x, d\rangle$ para todo $z \in \mathbb{R}^n$.

Esto es una generalización del concepto de plano tangente en un punto: d es un subgradiente si el plano que define queda completamente por debajo de la función. En la figura 3 se puede ver una interpretación geométrica en el caso unidimensional.

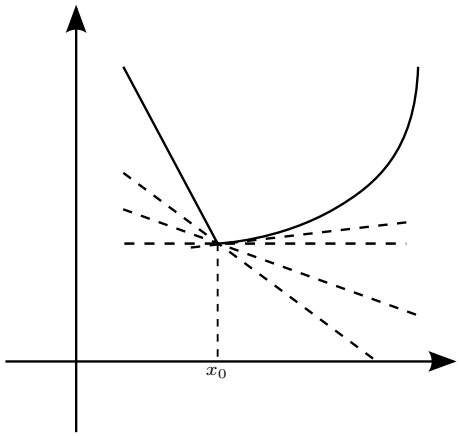


FIGURA 3. Función no diferenciable en x_0, con rectas que dejan el gráfico de la función completamente por arriba (subgradientes).

Cuando f es diferenciable en un punto, entonces el único subgradiente posible es el gradiente tradicional. En los puntos de no diferenciabilidad, puede haber más de un subradiente,como se ve en la figura. Al conjunto de todos los subgradientes de f en un punto x se denomina subdiferencial, y se denota como $\partial f(x)$. Para la función $f(x) = |x|$, tenemos por ejemplo $\partial f(1) = \{1\}$ y $\partial f(0) = [-1, 1]$. Es decir, todas las rectas por el origen con pendiente entre -1 y 1, dejan al gráfico de f por arriba.

Ahora podemos dar una condición de optimalidad más general:

Proposición 2. Sea $f : \mathcal{X} \to \mathbb{R}$ convexa, donde $\mathcal{X}$ es un subconjunto convexo de $\mathbb{R}^n$. Entonces x^* minimiza f sii existe un subgradiente $d \in \partial f(x^*)$ tal que $\langle d, z - x^*\rangle \geqslant 0$ para todo $z \in \mathcal{X}$.

Observar que si f es diferenciable, obtenemos la condición anterior, y que si x^* está en el interior de $\mathcal{X}$, la condición queda $0 \in \partial f(x^*)$, que generaliza la condición de anular el gradiente.

Veamos una interpretación geométrica de las condiciones de optimalidad. Para esto, definamos antes el cono normal:

Definición. Dado un conjunto convexo $\mathcal{X}$ y un punto $x \in \mathcal{X}$, llamamos cono normal a $\mathcal{X}$ por x al conjunto

$$N_{\mathcal{X}}(x) = \{g \in \mathbb{R}^n : \langle g, (z - x)\rangle \leqslant 0, \forall z \in \mathcal{X}\}.$$

Es decir, son las direcciones g que forman un ángulo de al menos $\pi/2$ con las direcciones factibles $z - x$. Dicho de otra manera, son los vectores g que definen un hiperplano que soporta al conjunto $\mathcal{X}$ en x.

Esto permite generalizar el concepto que el gradiente es normal a las curvas de nivel. Si f es una función convexa (no necesariamente diferenciable), fijamos $x_0 \in \mathbb{R}^n$, y consideramos el conjunto de nivel $C = \{z \in \mathbb{R}^n : f(x) \leqslant f(x_0)\}$. Entones es fácil ver que la condición para que $g \in \partial f(x_0)$, es equivalente a que g pertenezca al cono normal al conjunto de nivel por x_0: $g \in N_C(x_0)$. Si el borde de C es diferenciable, entones el cono normal es simplemente la semirrecta perpendicular al borde, por lo que en este caso, recuperamos lo sabido del gradiente y las curvas de nivel.

Volvamos ahora a las condiciones de optimalidad. La condición dada en la proposición 2 (que exista un subgradiente d tal que $\langle d, z - x^* \rangle \geqslant 0$ para todo $z \in \mathcal{X}$), se traduce en que exista un subgradiente $d \in \partial f(x^*)$ tal que $-d \in N_{\mathcal{X}}(x^*)$. Veamos un par de ejemplos.

Consideremos primero una función f diferenciable, pero un conjunto $\mathcal{X}$ convexo, con borde no diferenciable. Como f es diferenciable (y por lo tanto el único subgradiente es el gradiente), la condición de optimalidad es $-\nabla f(x^*) \in N_{\mathcal{X}}(x^*)$. Este es el caso que se puede ver en la figura 4.

Si ahora f es no diferenciable, pero asumimos que el mínimo se alcanza en un punto x^* donde el borde del conjunto $\mathcal{X}$ es diferenciable, entonces el cono normal a $\mathcal{X}$ será simplemente una semirrecta, a la cual deberá pertenecer el opuesto de algún subgradiente de f. Este es el caso de la figura 5. Se puede pensar, como ejercicio, cómo es geométricamente la condición de optimalidad cuando la función es no diferenciable y el mínimo se alcanza en un punto de no diferenciabilidad del borde de $\mathcal{X}$.

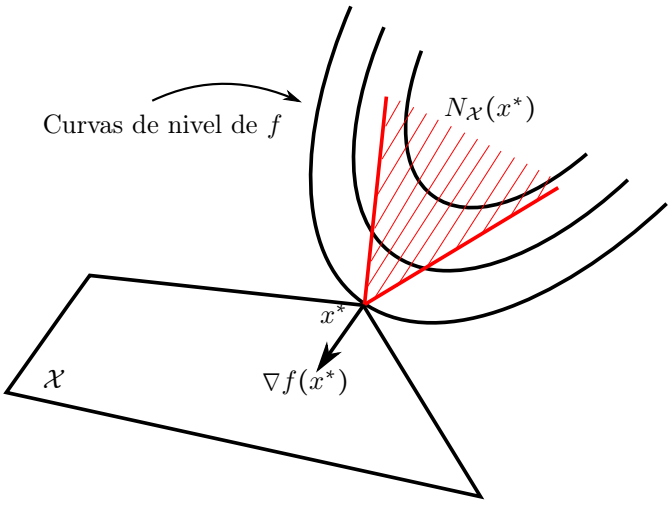


FIGURA 4. Curvas de nivel de una función diferenciable, y el mínimo se alcanza en un vértice de $\mathcal{X}$. En rojo, el cono normal a $\mathcal{X}$ por x^*, y se puede observar que el opuesto del gradiente pertenece al cono normal.

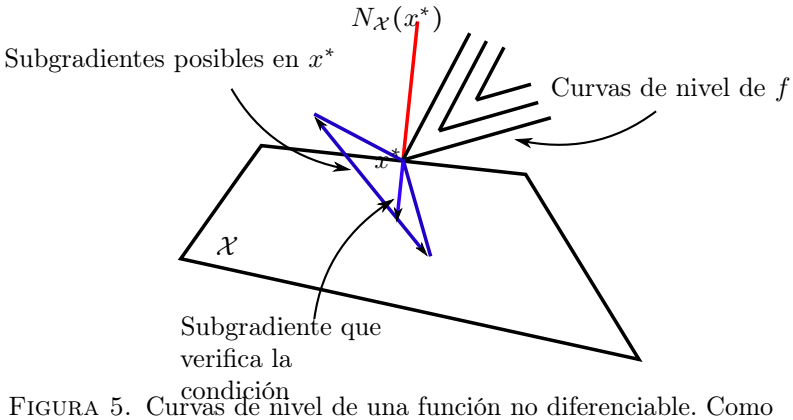


FIGURA 5. Curvas de nivel de una función no diferenciable. Como el mínimo se alcanza en un punto de diferenciabilidad del borde de $\mathcal{X}$, el cono normal es una semirrecta (en rojo). En azul, los posibles subgradientes de f en x^* (observar que forman un ángulo de $\pi/2$ con la curva de nivel por x^*), y el subgradiente cuyo opuesto pertenece al cono normal.

4. Métodos iterativos

4.1. Optimización sin restricciones. Hay una gran cantidad de métodos para resolver problemas de optimización, algunos bastante generales, y otros que dependen de la estructura del problema. La gran mayoría de los métodos son iterativos, y muchos de ellos son además de la siguiente forma: se comienza desde un punto x_0, y luego se construye una sucesión mediante $x_{k+1} = T(x_k)$, donde T es un operador que depende obviamente del problema. Este operador T puede incluir por ejemplo información de derivadas de distintos órdenes de f, o no. De esto depende, entre otras cosas, la complejidad (la cantidad de cálculos que hay que hacer para pasar de x_k a x_{k+1}), pero también la velocidad de convergencia, es decir, cuántas iteraciones k se necesitan para llegar a estar "cerca" de la solución.

El primer ejemplo es siempre el descenso por gradiente, o métodos relacionados. Consideremos primero que queremos minimizar una función diferenciable $f : \mathbb{R}^n \to \mathbb{R}$ (es decir, el conjunto $\mathcal{X}$ es todo el espacio, lo que usualmente se denomina como optimización sin restricciones). Lo que buscamos son puntos estacionarios, donde el gradiente de f se anule. Entonces, si en determinado momento estamos parados en un punto x_k, y su gradiente es no nulo, significa que (al menos) en la dirección $-\nabla f(x_k)$, la función es decreciente. Más aún, es la dirección de máximo decrecimiento de f, localmente. Por lo tanto tiene sentido moverse un poco en esa dirección, para calcular un nuevo punto x_{k+1}, que será mejor que x_k, en el sentido que su valor funcional será menor. Esto es:

$$x_{k+1} = x_k - \alpha \nabla f(x_k),$$

donde α es el "poco" que nos queremos mover. Este valor de α hay que elegirlo en cada iteración k, de manera de asegurarse (al menos) que $f(x_{k+1}) < f(x_k)$.[3] Una

[3]Tenemos garantizado, puesto que la derivada de f en la dirección $-\nabla f(x_k)$ es negativa, que existe un entorno $(0, \varepsilon)$ tal que para todo α en ese entorno, se tiene $f(x_{k+1}) < f(x_k)$.

opción es elegir el "mejor" valor de α, en el sentido que sea el que lleve a $f(x_{k+1})$ a ser el más chico posible (recordemos que la dirección ya está elegida)[4].

Surgen varias preguntas. Por ejemplo, con alguna elección "razonable" del paso α en cada iteración, la sucesión generada x_k, ¿converge a un punto estacionario? ¿Hay alguna dirección mejor que la de máximo descenso?

La respuesta a ambas preguntas es positiva.

En el caso de la convergencia, pidiendo algunas hipótesis más, se puede ver que hay valores de α fijos, constantes para toda iteración (es decir, que no hay que elegirlos en cada k), que garantizan convergencia de x_k a un punto estacionario.

Un algoritmo aceptable entonces para hallar puntos estacionarios (y si la función es convexa, la solución del problema de optimización), sería algo así:

Input: punto inicial x_0, paso $\bar{\alpha}$, tolerancia tol
Output: punto x_k con $\|\nabla f(x_k)\| < tol$
while $\|\nabla f(x_k)\| \geqslant tol$ **do**
  Calcular $\nabla f(x_k)$;
  Hacer $x_{k+1} = x_k - \bar{\alpha}\nabla f(x_k)$
end

Algoritmo 1: Descenso por gradiente

Sobre la segunda pregunta. ¿Por qué habría mejores direcciones que la dirección de máximo decenso?. Localmente, mirar hacia $-\nabla f$ es lo mejor que podemos hacer. Sin embargo, cuando las curvas de nivel son "estiradas", se produce un zigzagueo no deseado:

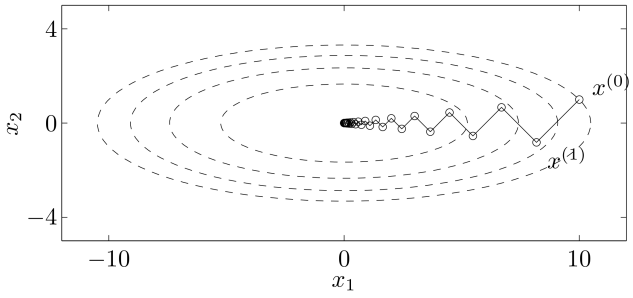


FIGURA 6. Descenso por gradiente para la función $f(x_1, x_2) = x_1^2 + 10x_2^2$. En punteado las curvas de nivel. Figura tomada de [8].

La información de esta geometría está en las derivadas segundas, por lo tanto debemos incorporar estas derivadas para lograr mejores trayectorias. Esta mejora viene de la mano de un mayor costo computacional: hay que calcular derivadas primeras y derivadas segundas, y eventualmente invertir una matriz.

El método de segundo orden más clásico es el de Newton, que consiste en corregir parcialmente la dirección del gradiente, usando información de segundo orden. Específicamente, el método consiste en iterar:

$$x_{k+1} = x_k - \alpha \left(\nabla^2 f(x_k)\right)^{-1} \nabla f(x_k),$$

[4]Observar que esto implica resolver otro problema de optimización, esta vez en $\mathbb{R}$. Es decir, hallar el valor de α que minimice $f(x_k - \alpha\nabla f(x_k))$. Hay soluciones eficientes que no requieren resolver esta minimización, como por ejemplo el método de Armijo (ver en [6]).

donde $\nabla^2 f(x_k)$ es la matriz Hessiana. Esta iteración surge de considerar la aproximación de Taylor de segundo orden alrededor de x_k, y hallar el mínimo de esa función.

El método de Newton requiere calcular muchas derivadas más, pero da lugar a convergencias más rápidas. Existen métodos intermedios (considerando la diagonal de la Hessiana solamente, o aproximaciones similares), que manejan un compromiso entre la convergenia y la cantidad de cálculos.

De la misma manera, hay mejoras al método de Newton, algunos que tienen en cuenta no solo x_k sino también el punto anterior x_{k-1}, haciendo uso de la historia de la iteración, o también considerando derivadas de orden superior. En cualquier caso, a no ser que los métodos sean el objeto principal de estudio, o se requiera una convergencia ridículamente rápida, en general, para minimizar funciones diferenciables en $\mathbb{R}^n$, basta con el método de descenso por gradiente o (algún) método de Newton.

4.2. Optimización con restricciones. Cuando se tienen algunas restricciones en la optimización (es decir, que el conjunto $\mathcal{X}$ no es todo el espacio), hay que hacer modificaciones a los métodos presentados. Describiremos aquí solamente uno de ellos.

Supongamos entonces que tenemos una función $f : \mathcal{X} \to \mathbb{R}$, con $\mathcal{X} \subset \mathbb{R}^n$ convexo, y partimos de cierto punto $x_0 \in \mathcal{X}$ como antes. El objetivo es construir una sucesión x_k que converja a un mínimo local (o al menos a un punto que satisfaga la condición de optimalidad).

En cualquier momento, estando en x_k, podemos calcular una dirección de descenso (la dirección opuesta al gradiente por ejemplo), e intentar movenos en esa dirección un paso α. El problema es que este movimiento nos puede llevar fuera del conjunto factible $\mathcal{X}$. Para asegurarnos que esto no pasa, y aprovechando la convexidad de $\mathcal{X}$, la dirección en la que nos moveremos estará dada por $\bar{x} - x_k$ para algún $\bar{x} \in \mathcal{X}$. Es decir, estando en x_k, nos moveremos en dirección a $\bar{x}$.

Estos métodos se pueden escribir entonces de forma general como:

$$x_{k+1} = x_k + \alpha(\bar{x} - x_k),$$

donde resta definir la elección de $\bar{x} \in \mathcal{X}$, y el paso α.

Una opción, que da lugar al *Método de gradiente proyectado*, es tomar $\bar{x} = P_{\mathcal{X}}(x_k - s\nabla f(x_k))$. Es decir, nos movemos en la dirección opuesta al gradiente, y si caemos fuera de $\mathcal{X}$, lo proyectamos, y usamos este punto para la dirección. En particular si tomamos $\alpha = 1$ (que es una posible elección dentro de la familia del método de gradiente proyectado), la iteración resulta:

$$x_{k+1} = P_{\mathcal{X}}\left(x_k - s_k \nabla f(x_k)\right).$$

Como primer comentario sobre la efectividad de este método, basta decir que tenemos resultados de convergencia similares al descenso por gradiente clásico.

De todas maneras, hagamos un chequeo sobre el comportamiento del método. ¿Cuándo se estanca el método? Mirándolo en la iteración

$$x_{k+1} = x_k + \alpha(\bar{x} - x_k),$$

es claro que el algoritmo se estanca cuando $\bar{x} = x_k$. Esto, en este caso particular que estamos viendo aquí, significa que

$$x_k = P_{\mathcal{X}}\left(x_k - s\nabla f(x_k)\right),$$

que, si recordamos, es exactamente la condición de estacionareidad, equivalente a la condición de optimalidad que vimos al principio.

5. Algoritmos iterativos como discretizaciones de flujos

En la sección anterior vimos algoritmos iterativos, presentados directamente en su naturaleza discreta. Básicamente definimos las sucesiones expresando, a partir de la posición en determinado momento x_k, hacia dónde y cuánto nos movemos, para luego calcular el siguiente iterado x_{k+1}. Sin embargo, algunos de estos métodos pueden ser interpretados como discretizaciones de ecuaciones diferenciales convenientes.

Volvamos a considerar el problema de optimización sin restricciones (supongamos por ahora que f es diferenciable y su gradiente es Lipschitz):

$$\arg\min_{x\in\mathbb{R}^n} f(x)$$

donde recordemos que buscamos un punto donde se anule el gradiente.

Es natural entonces pensar en el flujo gradiente:

$$\begin{cases} \dot{x}(t) = -\nabla f\left(x(t)\right) & \forall\, t > 0 \\ x(0) = x_0 \end{cases}$$

Claramente los puntos críticos de esta ecuación diferencial son los que buscamos, y además es fácil ver que si $x(t)$ es una solución, entonces $\frac{df(x(t))}{dt} = -\|\nabla f(x(t))\|^2 < 0$, y por lo tanto $f(x(t))$ es decreciente con t. Si además tenemos la hipótesis de convexidad, entonces $x(t)$ converge al mínimo global.

Si discretizamos la ecuación, aproximando la derivada por el cociente incremental en iterados sucesivos, tenemos:

$$\frac{x_{k+1} - x_k}{h} = -\nabla f(x_k).$$

Esta discretización, que corresponde al método de Euler (o de Euler explícito, o *forward Euler*), da lugar justamente a la iteración estudiada arriba como el método de descenso por gradiente. En efecto, resulta:

$$x_{k+1} = x_k - h\nabla f(x_k).$$

Hay otro método de discretización, que en general presenta mejores condiciones de estabilidad, llamado el método de Euler implícito (o *backward Euler*). La diferencia está en el punto donde se evalúa la derivada:

$$\frac{x_{k+1} - x_k}{h} = -\nabla f(x_{k+1}).$$

Si bien ahora no podemos despejar x_{k+1} para poder entender directamente la iteración correspondiente, resulta que es exactamente la condición de optimalidad de la iteración $x_{k+1} = prox_{hf}(x_k)$, donde $prox$ es el denominado operador proximal, que es la base de varios métodos modernos de optimización, sobretodo para funciones no diferenciables. No comentaremos detalles de esto aquí, pero referimos a [6, 7] para encontrar una profundización del tema.

Volviendo al flujo gradiente, una propiedad interesante sobre la unicidad de la solución y la convergencia al mínimo global, es que vale también en casos bastante

más generales. Por ejemplo, si f es una función convexa pero no necesariamente diferenciable, podemos estudiar la siguiente inclusión diferencial:

$$\begin{cases} \dot{x}(t) \in -\partial f\,(x(t)) & \text{para casi todo } t > 0 \\ x(0) = x_0 \end{cases}$$

donde ∂f denota el subdiferencial de f como fue definido anteriormente.

Se puede probar fácilmente la siguiente

Proposición 3. Sean x_1 y x_2 dos soluciones de la inclusión diferencial, entonces tenemos que para todo t se cumple $|x_1(t) - x_2(t)| \leqslant |x_1(0) - x_2(0)|$. En particular, esto implica la unicidad de la solución.

La discretización de esta inclusión diferencial con el método *forward Euler* da lugar al método proximal para funciones no diferenciables.

Sobre flujos gradiente en espacios más generales, se puede leer en [21].

6. Optimización en variedades

En la sección 4.2 vimos un ejemplo para optimizar una función f sobre un conjunto $\mathcal{X} \subset \mathbb{R}^n$, pero que no utilizaba la estructura o la geometría de $\mathcal{X}$ (más que para la proyección sobre el conjunto). En muchos casos, el conjunto $\mathcal{X}$ tiene una estructura diferenciable que puede ayudarnos para diseñar métodos de optimización, que por algún motivo pueden ser más convenientes.

Hay muchos ejemplos de esto, pero en particular los problemas de las secciones 2.2 y 2.3 responden a esta situación.

En efecto, en el problema de isomorfismo de grafos (o de automorfismo de grafos) buscamos una matriz de permutación P que conjugue A con B, es decir que $A = PBP^T$. Ahora, las matrices de permutación son en particular ortogonales[5], y por lo tanto es interesante buscar matrices Q ortogonales tales que $A = QBQ^T$, o equivalentemente, estudiar el siguiente problema:

$$\arg \min_{Q \in \mathcal{O}(n)} \|A - QBQ^T\|_F^2,$$

donde $\mathcal{O}(n)$ es el grupo ortogonal. Por supuesto que esto no soluciona el problema de isomorfismo de grafos (pues el resultado no necesariamente es una matriz de permutación), pero de todas maneras es útil.

En el caso del problema de valores y vectores propios sucede algo similar. Dada una matriz A a diagonalizar, se desea encontrar una matriz ortogonal que resuelva

$$min_{U \in \mathcal{O}(n)} \|N - U^T AU\|_F^2$$

donde N es una matriz diagonal, por ejemplo $N = diag(1, 2, 3, \ldots, n)$.

Como fue mencionado arriba, una opción es considerar la función a minimzar $f : \mathbb{R}^{n \times n} \to \mathbb{R}$, y un conjunto $\mathcal{X} = \{U \in \mathbb{R}^{n \times n} : UU^T = Id\}$, y utilizar un método de optimización con restricciones como por ejemplo: $U_{k+1} = P_{\mathcal{X}}\,(U_k - \alpha \nabla f(U_k))$.

Otra opción es considerar directamente $f : \mathcal{O}(n) \to \mathbb{R}$. La función f (cualquiera de las presentadas como ejemplo) es diferenciable, y como $\mathcal{O}(n)$ tiene una estructura de variedad Riemanniana, podemos definir el flujo gradiente asociado a f sobre $\mathcal{O}(n)$:

$$\dot{U}(t) = -\operatorname{grad} f(U(t)), \tag{9}$$

[5]De hecho, son la intersección delas matrices ortogonales con las de entradas no negativas.

donde grad f es el gradiente de f definido por la métrica en $\mathcal{O}(n)$[6]. Se tiene por lo tanto que si $U(t)$ es solución de (9), entonces $U(t) \in \mathcal{O}(n)$ para todo t.

Se puede estudiar entonces la dinámica de (9), y obtener resultados como por ejemplo (para el problema de valores propios) que el flujo es completo, que converge a puntos de equilibrio, y que hay un abierto denso de condiciones iniciales tales que el flujo converge a la solución deseada. Ver por ejemplo [3].

Ahora, esto define un flujo continuo, que hay que discretizar para poder implementarlo. Como antes, hay que tener ciertos cuidados a la hora de discretizar. Por ejemplo, cuando en determinado momento se está en un punto U_k, y se desea mover una cantidad α en la dirección $-\operatorname{grad} f(U_k)$, que es un elemento del espacio tangente, en general se caerá fuera de la variedad, por lo que hay que proyectar de nuevo a la misma. Una forma eficiente de hacer esto es mediante las llamadas retracciones.

Se puede encontrar un estudio detallado de esta teoría en [1, 18], y en las notas de curso [3].

Referencias

[1] P.A. Absil, R. Mahony, and R. Sepulchre, *Optimization algorithms on matrix manifolds*, Princeton University Press, Princeton, NJ, USA, 2007.

[2] Y. Aflalo, A. Bronstein, and R. Kimmel, *Graph matching: relax or not?*, arXiv:1401.7623 (2014).

[3] Diego Armentano, *Notas de curso: Optimización en variedades*, 2017. `http://www.cmat.edu.uy/~diego/documents/notas-opt-var-2017.pdf`.

[4] László Babai, *Graph isomorphism in quasipolynomial time*, arXiv preprint arXiv:1512.03547 (2016).

[5] O. Banerjee, L. El Ghaoui, A. D'Aspremont, and G. Natsoulis, *Convex optimization techniques for fitting sparse Gaussian graphical models*, Proceedings of the 23rd international conference on Machine learning - ICML '06 (2006), 89–96.

[6] Dimitri P Bertsekas, *Nonlinear programming*, Athena scientific Belmont, 1999.

[7] ———, *Convex optimization algorithms*, Athena Scientific Belmont, 2015.

[8] Stephen Boyd and Lieven Vandenberghe, *Convex optimization*, Cambridge university press, 2004.

[9] Alfred M Bruckstein, David L Donoho, and Michael Elad, *From sparse solutions of systems of equations to sparse modeling of signals and images*, SIAM review **51** (2009), no. 1, 34–81.

[10] Emmanuel J Candès and Benjamin Recht, *Exact matrix completion via convex optimization*, Foundations of Computational mathematics **9** (2009), no. 6, 717.

[11] Emmanuel J Candès, Justin Romberg, and Terence Tao, *Robust uncertainty principles: Exact signal reconstruction from highly incomplete frequency information*, IEEE Transactions on information theory **52** (2006), no. 2, 489–509.

[12] Emmanuel J Candès and Terence Tao, *The power of convex relaxation: Near-optimal matrix completion*, IEEE Transactions on Information Theory **56** (2010), no. 5, 2053–2080.

[13] A. Dempster, *Covariance selection*, Biometrics **28** (1972), no. 1, 157–175.

[14] Nadav Dym, Haggai Maron, and Yaron Lipman, *Ds++: A flexible, scalable and provably tight relaxation for matching problems*, ACM Trans. Graph. **36** (November 2017), no. 6, 184:1–184:14.

[15] Marcelo Fiori, Pablo Musé, Ahamd Hariri, and Guillermo Sapiro, *Multimodal graphical models via group lasso*, Signal Processing with Adaptive Sparse Structured Representations (2013).

[6] Es decir: sea $(M, \langle\cdot,\cdot,\rangle_x)$ una variedad Riemanniana, y $f : M \to \mathbb{R}$ una función diferenciable. Definimos el *gradiente* de f como el vector tangente $\operatorname{grad} f(x) \in T_x M$, $x \in M$, que satisface

$$Df(x)v_x = \langle \operatorname{grad} f(x), v_x \rangle, \quad \forall\, v_x \in T_x M.$$

[16] Marcelo Fiori and Guillermo Sapiro, *On spectral properties for graph matching and graph isomorphism problems*, Information and Inference: A Journal of the IMA **4** (2015), no. 1, 63–76.

[17] Marcelo Fiori, Pablo Sprechmann, J.T. Vogelstein, Pablo Musé, and Guillermo Sapiro, *Robust multimodal graph matching: Sparse coding meets graph matching*, Advances in Neural Information Processing Systems 26 (2013), 127–135.

[18] Uwe Helmke and John B Moore, *Optimization and dynamical systems*, Springer Science & Business Media, 2012.

[19] V. Lyzinski, D. Fishkind, M. Fiori, J.T. Vogelstein, C.E. Priebe, and G. Sapiro, *Graph matching: Relax at your own risk*, arXiv preprint arXiv:1405.3133 (2014).

[20] Julien Mairal, Francis Bach, and Jean Ponce, *Sparse modeling for image and vision processing*, Foundations and Trends® in Computer Graphics and Vision **8** (2014), no. 2-3, 85–283.

[21] Filippo Santambrogio, {*Euclidean, metric, and Wasserstein*} *gradient flows: an overview*, Bulletin of Mathematical Sciences **7** (2017), no. 1, 87–154.

[22] Steve Smale, *Mathematical problems for the next century*, Mathematical Intelligencer **20** (1998), 7–15.

IMERL, Facultad de Ingeniería, Universidad de la República. Montevideo, Uruguay.
Email address: mfiori@fing.edu.uy