UNIVERSIDAD TECNICA FEDERICO SANTA MARIA

Repositorio Digital USM **https://repositorio.usm.cl**

Departamento de Arquitectura Arq_paso

2019-08-12

UN MÉTODO DE DESCENSO POR COORDENADAS DOBLEMENTE ACELERADO, PROXIMAL Y PARALELO

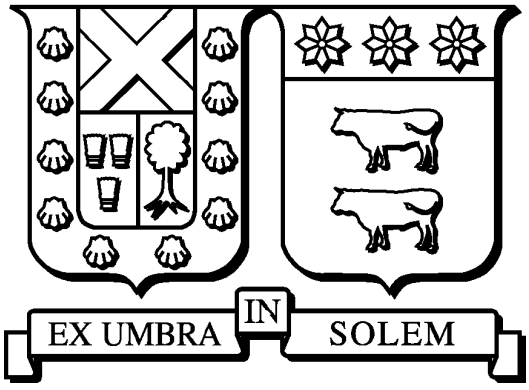
BENAVIDES LORCA, FRANCISCO JAVIER

https://hdl.handle.net/11673/55799

Repositorio Digital USM, UNIVERSIDAD TECNICA FEDERICO SANTA MARIA

UNIVERSIDAD TÉCNICA FEDERICO SANTA MARÍA

Departamento de Matemática
Valparaíso-Chile



Un método de descenso por coordenadas doblemente acelerado, proximal y paralelo.

Tesis presentada por:
Francisco Javier Benavides Lorca

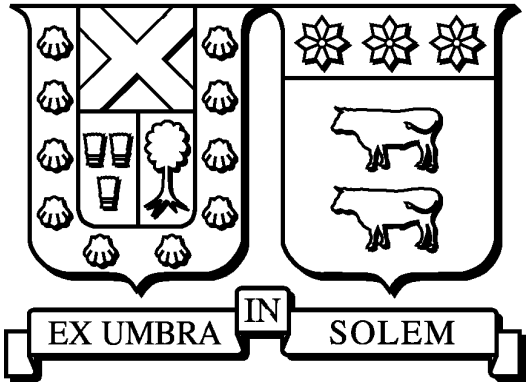
Como requisito parcial
para optar al grado de Magíster en Ciencias, mención Matemática.

Profesor Guía/Co Director:
Juan G. Peypouquet/ Pedro Gajardo A.

12 de Agosto, 2019

Universidad Técnica Federico Santa María

Departamento de Matemática
Valparaíso-Chile



Un método de descenso por coordenadas doblemente acelerado, proximal y paralelo.

Tesis presentada por:
Francisco Javier Benavides Lorca

Como requisito parcial
para optar al grado de Magíster en Ciencias, mención Matemática.

Profesor Guía y Co Director:
Juan G. Peypouquet
Pedro Gajardo A.

Examinadores:
Luis Briceño
Claudio Morales

12 de Agosto, 2019

Material de referencia, su uso no involucra responsabilidad del autor o de la Institución.

TÍTULO DE LA MEMORIA:

Un método de descenso por coordenadas doblemente acelerado, proximal y paralelo.

AUTOR: Francisco Javier Benavides Lorca.

TRABAJO DE TESIS, presentado como requisito parcial para optar al grado de Magíster en Ciencias, mención Matemática.

COMISIÓN EVALUADORA:

Integrantes	Firma
Luis Briceño Universidad Técnica Federico Santa María, Chile.	_____
Pedro Gajardo Universidad Técnica Federico Santa María, Chile.	_____
Claudio Morales Universidad de Alabama-Huntsville, USA.	_____
Juan Peypouquet Universidad de Chile, Chile	_____

Valparaíso, 12 de Agosto, 2019.

Agradecimientos

Durante los últimos años muchas personas me han ayudado, ya sea de manera directa o indirecta a finalizar este trabajo. Me gustaría comenzar mencionando a mi madre, quien siempre ha estado motivándome a ser cada vez mejor y ser fiel a mí mismo. En general podría repetir a las mismas personas a las que ya agradecí al finalizar mi ingeniería, a mis amigos de la vida y de la universidad, a mi polola y las diferentes personas que han aportado su pequeño grano de arena en mi formación. Sin embargo, me gustaría resaltar a personas que han sido claves durante estos años.

Me gustaría comenzar con la profesora Claudia Sagastizabal, pues sin su guía, esta tesis nunca hubiera visto la luz. También al departamento de matemáticas de la universidad, principalmente al profesor Pablo Aguirre, quien con su gestión y ayuda, permitió que este trabajo llegara a buen puerto, además también destacar al profesor Pedro Gajardo por tomar una carga (yo) que no le correspondía inicialmente. Además destacar el aporte de Conicyt, quien financió mis años de estudio y al IMPA, institución que me acogió durante un semestre y con ello conocer a la profesora Claudia Sagastizabal y al profesor Welington de Oliveira. Finalmente, me gustaría mencionar al profesor Juan Peypouquet, quien a pesar de todos nuestros desencuentros, fue quien motivó en mi interior la inquietud de profundizar en mis estudios y realizar una maestría.

A lo anterior, quiero añadir a las personas con las que he trabajado desde hace ya un año en Walmart Chile, a quienes agradezco profundamente haber conocido. A Juan Eduardo, Javier Burgos, Adolfo Fuentes y, con mayor fuerza, a Juan Barra y Samuel Ramos, quienes me han apoyado constantemente en lo laboral y personal.

Una persona sola nunca gana, cuando trabaja en equipo gana siempre.

Sam Walton

Índice general

Un método de descenso por coordenadas doblemente acelerado, proximal y paralelo.

por Francisco Javier Benavides Lorca

Resumen

En este trabajo proponemos una variante del algoritmo APProx (ver [13]), el cual pertenece a la familia de los algoritmos de descenso por bloque aleatorios, los que se caracterizan por actuar en cada iteración sobre una cantidad de coordenadas (o bloques de ellas), igual o menor al total disponible, las cuales se seleccionan de manera aleatoria. Este algoritmo busca resolver el siguiente problema de optimización convexa:

$$\begin{cases} \text{mín} & F(x) := f(x) + \psi(x) \\ \text{s.t.} & x = \left(x^{(1)}, \ldots, x^{(n)}\right) \in \mathbb{R}^N \\ & \text{for } x^{(i)} \in \mathbb{R}^{N_i}, i = 1, \ldots, n \end{cases} \quad \text{donde} \quad \begin{cases} f(x) &= \sum_{j=1}^{m} f_j(x) \\ \psi(x) &= \sum_{i=1}^{n} \psi_i\left(x^{(i)}\right). \end{cases} \tag{1}$$

Donde (f) es una la suma de funciones convexas suaves f_j con estructura separable por bloques, mientras que el segundo termino (ψ) es un regularizador convexo, no necesariamente suave, con estructura separable por bloque (por ejemplo $\psi(x) = \|x\|_1$). Este problema es relevante, pues es el problema principal de muchos algoritmos de machine learning (aprendizaje supervisado) e inteligencia artificial (redes neuronales).

La importancia del algoritmo APProx esta dada por la particularidad de ser acelerado, proximal y paralelizable. Paralelizable, en el sentido de que cada bloque puede ser actualizado en una CPU diferente; Proximal, al poder lidiar con funciones subdiferenciables; Y acelerado, pues posee una tasa de convergencia de $O(1/k^2)$. Nuestra variante, la cual nombramos *aAPProx* posee estas mismas cualidades y, conjeturamos, posee una tasa de convergencia menor $(o(1/k^2))$.

En concreto, presentamos un bosquejo de la demostración de dicha convergencia, la cual se obtiene al extrapolar las ideas presentes en [1], en donde se obtiene este mismo resultado para el algoritmo Backward-Forward de Nesterov, el cual es (a grandes rasgos) la versión determinista de APPROX. Además se presenta de manera empírica este resultado, mediante un problema de recuperación de imágenes.

A double acellerate, proximal and parallel block coordinate descent method.

by Francisco Javier Benavides Lorca

Abstract

In this work, we propose a variant of the APProx algorithm (see [13]). This belongs to the block coordinate descent algorithm family, which principal characteristic is, on each interation, update just some coordinates (or blocks of them) less or equal to the total available, selected by random. This algorithm solves the following convex optimization problem:

$$\begin{cases} \text{mín} & F(x) := f(x) + \psi(x) \\ \text{s.t.} & x = \left(x^{(1)}, \ldots, x^{(n)}\right) \in \mathbb{R}^N \\ & \text{for } x^{(i)} \in \mathbb{R}^{N_i}, i = 1, \ldots, n \end{cases} \quad \text{where} \quad \begin{cases} f(x) &= \sum_{j=1}^{m} f_j(x) \\ \psi(x) &= \sum_{i=1}^{n} \psi_i\left(x^{(i)}\right). \end{cases} \tag{2}$$

Where (f) is the sum of smooth convex functions f_j with block separate structure, while the second term (ψ) is a convex regularizer, not necessary smooth, with block separate structure (for example $\psi(x) = \|x\|_1$). This is a relevance problem because this is uses on most of the machine learning and IA problems.

The importance of APProx, is because this is acccelerated, proximal and parallel. Parallel in the sense of each block can be updated on each CPU; Proximal, because APProx can handle with not smooth convex functions; And Accelerated, because this has a convergence rate of $O(1/k^2)$. Our variant, named as *aAPProx* has this same properties and, we guess, a convergence rate of $(o(1/k^2))$.

In fact, we presented a scheme of the proof for this convergence rate, obtained using some ideas presented on [1], where they obtained this result for the Backward-Forward Nesterov algorithm, which could be considered as the deterministic version of APProx. Also we presented some empirical results, for the recovery images problem.

Capítulo 1

Introducción

En los últimos años el aumento de la capacidad computacional para medir, almacenar y procesar información ha aumentado exponencialmente la cantidad de variables que se deben considerar a la hora de tomar decisiones que se basen en dichos datos, generando un nuevo contexto denominado *big data*. Esto motiva un nuevo enfoque en la matemática aplicada, pues áreas como el análisis numérico de EDPs, aprendizaje de máquinas, optimización combinatorial, análisis de redes, entre otros, han ido tomando mayor participación dentro de este nuevo contexto mediante la optimización convexa, en donde los métodos clásicos toman fuerza mediante variantes que explotan la estructura de los datos.

Dentro de esta clase de problemas destacan los relacionados a los conceptos de *aleatoriedad* y *aprendizaje*. Usualmente estos buscan determinar la relación entre datos provenientes del mundo real y un modelo, el cual es usado para predecir estados futuros o información faltante. Los modelos que se usan para estos problemas, similares a los utilizados para los problemas de mínimos cuadrados y regresión, dependen de parámetros que se ajustan mediante un problema de minimización que mide alguna clase de desviación, discrepancia o pérdida (ver [5], [15], [12]).

Una manera popular para enfrentar este tipo de problemas en el contexto del *big data* son los denominados *coordinate descend methods* (CDM) o *métodos de descenso por bloques*, los cuales actualizan solo algunas variables/coordenadas (o bloques de ellas) en cada iteración, mediante algoritmos tipo gradiente, como el de máximo descenso y el gradiente-proximal, reduciendo la memoria computacional necesaria y la complejidad aritmética de cada iteración, debido a la capacidad de operar en procesos paralelos o distribuidos.

El esquema general considerado por los métodos de descenso por bloque es el siguiente: Para un valor grande de variables $N > 0$ y enteros positivos $N_1, \ldots, N_n$, tales que $\sum_{i=1}^{n} N_i = N$,

consideramos el siguiente problema de optimización con estructura por bloques,

$$\begin{cases} \text{mín} & F(x) := f(x) + \psi(x) \\ \text{s.t.} & x = \left(x^{(1)}, \ldots, x^{(n)}\right) \in \mathbb{R}^N \\ & \text{para } x^{(i)} \in \mathbb{R}^{N_i}, i = 1, \ldots, n \end{cases} \quad \text{donde} \quad \begin{cases} f(x) &= \sum_{j=1}^{m} f_j(x), \\ \psi(x) &= \sum_{i=1}^{n} \psi_i\left(x^{(i)}\right). \end{cases} \tag{1.1}$$

En este contexto $F : \mathbb{R}^N \longrightarrow \mathbb{R}$, además $\mathbb{R}^N$ está compuesto por $1 \leq n \leq N$ bloques, donde $x^{(i)} \in \mathbb{R}^{N_i}, i = \{1, \ldots, n\}$. La función f está dada por la suma de funciones convexas diferenciables f_j, las que dependen solo de un subconjunto de bloques $C_j \subset [n] = \{1, \ldots, n\}$ y con gradiente Lipschitz-continuo en cada bloque; mientras que la función ψ es un regularizador convexo (posiblemente no diferenciable) separable por bloques, es decir, la función ψ puede ser escrita como la suma de funciones ψ_i que dependen solo del i-ésimo bloque, por ejemplo $\psi(x) = \|x\|_1$.

Cuando un problema de optimización es de la forma (1.1), los CDM habitualmente son consideramos como la manera natural para lidiar con ellos. Sin embargo, estos presentan una tasa de convergencia menor que el método de gradiente clásico, pues la tasa de los CDM depende del número de bloques que se actualizan en cada iteración, lo cual relentiza el algoritmo.

Para acotar la brecha entre los CDM y el método clásico del gradiente se ha estudiado intensivamente la manera en la que se eligen los bloques que se actualizan en cada iteración. Cuando los bloques son elegidos de manera determinista, la regla utilizada suele ser actualizarlos en un orden predeterminado, generalmente de manera cíclica, en donde los esfuerzos se han enfocado en mejorar las cotas teóricas de complejidad, de manera que la cantidad de bloques por iteración no influya de manera excesiva en la tasa de convergencia (ver [17], [30]). Por otro lado, cuando los bloques son elegidos de manera aleatoria, estos métodos alcanzan la misma tasa de convergencia que el método del gradiente clásico, pero en un sentido más débil, pues al poseer una componente estocástica, los resultados son obtenidos con respecto al valor esperado (la tasa de convergencia se obtiene para un valor promedio al correr varias veces el algoritmo). Cuando un CDM utiliza un criterio aleatorio para la selección de bloques se denomina *Random Coordinate Descend Method* (RCDM) o *métodos de descenso por bloques aleatorios.*

Los RCDM han probado ser una herramienta eficaz al resolver problemas relacionados al *big data*, debido a que éstos suelen requerir soluciones a baja o mediana precisión. Esta cualidad también ha provocado que métodos de primer orden cobren popularidad para resolver (1.1) (ver [4], [20], [27]), además como el número de variables N es grande, el solo hecho de evaluar el (sub)gradiente de F puede tener un alto costo computacional, lo cual hace prácticamente inviable los métodos de orden superior. En este contexto, la aleatoriedad

nos permite obtener estimaciones insesgadas del gradiente completo mediante promedios que involucren solo algunos bloques, por ejemplo, considerando solo $\nabla_i f$, el cual es la parte del gradiente de f asociado al i-ésimo bloque, y ψ_i (ver (2.5) y el paso 7 en el algoritmo 2). De esto modo, podemos revisar solo algunas muestras aleatorias de bloques en vez de trabajar con todas las variables.

La mayoría de los RCDM tradicionales, derivados del trabajo presente en [23] (ver [35],[16]), consideran funciones diferenciables F para el problema (1.1). Uno de los primeros algoritmos en considerar funciones no diferenciables es el algoritmo proximal paralelo acelerado o APProx por sus siglas en inglés, presentado en [13], el cual posee una tasa de convergencia de $O\left(\frac{1}{k^2}\right)$, es decir, obtenemos un valor óptimo de la función a la misma velocidad con la que $\frac{1}{k^2} \to 0$ cuando $k \to \infty$, en donde k corresponde al número de iteraciones.

El esquema considerado por APProx resulta ser eficiente, en el sentido de que, cuando las componentes f_j de la función diferenciable f en (1.1) son de la forma $f_j(x) = \phi_j(a_j^T x)$, con ϕ_j una función escalar convexa con derivada Lipschitz-continua y los vectores a_j poseen estructura de bloque, evita realizar operaciones sobre todas las variables. Por ejemplo las funciones cuadráticas $f(x) = \frac{1}{2}\|Ax - b\|^2 = \frac{1}{2}\sum_{j=1}^{m}\left(e_j^\top Ax - b_j\right)^2$ pueden ser escritas de esta forma al considerar $f_j = \phi_j(e_j^\top Ax)$, donde $\phi_j(s) = \frac{1}{2}(s - b_j)^2$.

Esta peculiaridad es vital para evitar los cuellos de botella que hacen inviables a los métodos de descenso por bloques tradicionales (ver [23]). Además APProx resulta ser ampliamente versátil al poder modificar sus cuatro componentes principales: La definición del término regularizador ψ, el número de bloques por iteración, la manera en que se eligen los bloques, y la elección de los parámetros que influyen en el término diferenciable de F (ver algoritmo 2). Al modificar estas componentes, APProx puede emular métodos ya conocidos para resolver (1.1), tales como el método del gradiente, tanto en versiones aceleradas, proyectadas o proximales; el método proximal acelerado de Tseng, y el método gradiente-proximal y sus variantes.

En este trabajo propondremos una variante de APProx que denominaremos **aAPProx**, el cual, conjeturamos, converge con una tasa de $o(\frac{1}{k^2})$, es decir, va un poco más rápido que $\frac{1}{k^2} \to 0$ cuando $k \to \infty$. Dicha aceleración sería posible mediante una nueva elección de parámetros, basada en [1]. La facultad de **aAPProx** es que, además de ser una posible versión acelerada de APPROX, podría ser visto como una versión aleatorizada de los métodos presentes en [1]. Esta nueva tasa de convergencia se obtendría combinando las técnicas presentes en [13] y el enfoque determinista de [1], además de adaptar algunos argumentos presentes en [33] y [7] a nuestro contexto por bloques.

Este trabajo está organizado de la siguiente forma: En el capítulo 2 introduciremos conceptos teóricos que sustentan nuestro trabajo, principalmente relacionados con el análisis convexo

y algunos algoritmos de optimización; En el capítulo 3 hablaremos sobre APProx y aPProx, en donde mostraremos un bosquejo de la posible demostración de dicho resultado, además de resultados empíricos que avalan nuestra hipótesis; En el capítulo 4 mencionaremos las principales conclusiones y trabajos futuros; Finalmente en el capítulo 5 estarán presente tópicos anexos que complementan el trabajo realizado.

Capítulo 2

Definiciones y notación

En este capítulo daremos los conocimientos básicos en los que se fundamenta nuestro trabajo. Repasaremos conceptos básicos del análisis convexo, además de los métodos iterativos clásicos para lidiar con el problema (1.1). Finalmente veremos versiones aceleradas de estos algoritmos y su aplicación en los métodos de descenso por bloques.

2.1. Algunas propiedades de funciones

En esta sección hablaremos sobre algunas propiedades de funciones $F : \mathbb{R}^N \to \bar{\mathbb{R}}$, donde $\bar{\mathbb{R}} \stackrel{\text{def}}{=} \mathbb{R} \cup \{+\infty\}$, con las convenciones: $a \pm \infty = \pm\infty$, para $a \in \mathbb{R}$ y $0 * \infty = \infty * 0 = 0$.

Definición 1. Para una función $F : \mathbb{R}^N \to \bar{\mathbb{R}}$, se define el *dominio* de F como el conjunto de todos los puntos donde F es finito, es decir,

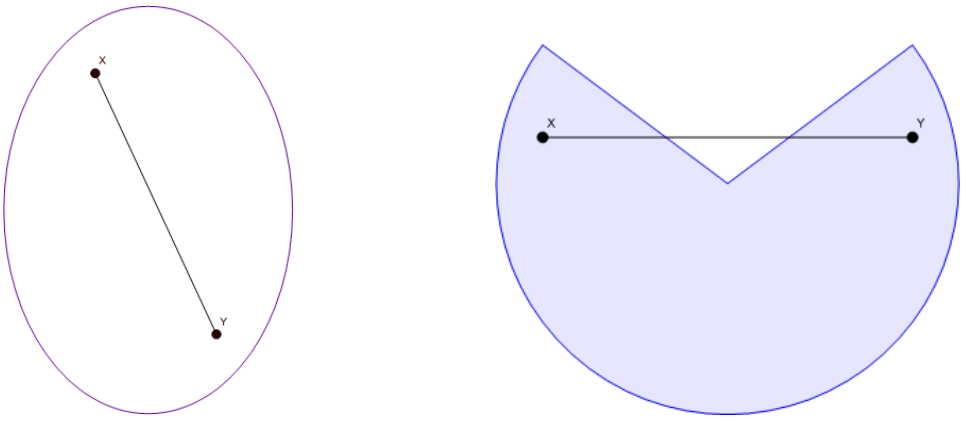
$$\text{dom}(F) = \{x \in \mathbb{R}^N : \quad F(x) < +\infty\}.$$

Si el dominio es no vacío, decimos que la función F es *propia*. Para este conjunto, necesitamos la propiedad de convexidad.

Definición 2. Sea $C \subset \mathbb{R}^N$. Decimos que C es convexo si, para todo $x, y \in C$, la linea que une a estos dos puntos está contenida en C, es decir,

$$\forall x, y \in C, \lambda \in (0, 1) : \lambda x + (1 - \lambda) y \in C.$$

La figura 2.1 muestra un ejemplo de conjunto convexo y no convexo.



(a) (b)

FIGURA 2.1: Ejemplo de conjunto (a) convexo (b) no convexo.

Definición 3. Una función $F : \mathbb{R}^N \to \bar{\mathbb{R}}$ es *coerciva* si

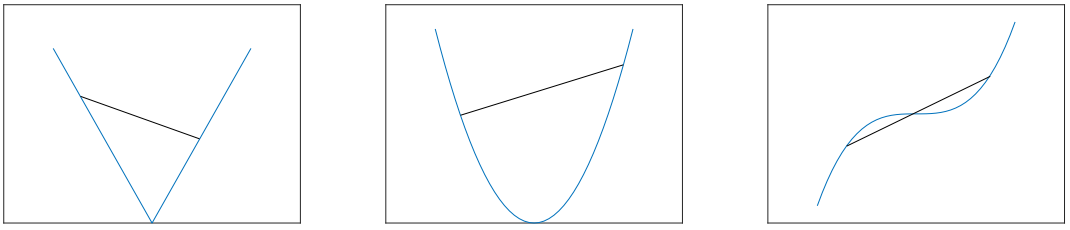
$$\lim_{\|x\|\to\infty} F(x) = +\infty.$$

En general, resolver un problema de optimización para una función objetivo cualquiera puede resultar muy complicado, por lo que se trabaja con un conjunto de funciones que, por su geometría, facilitan la tarea de encontrar mínimos. Estas funciones son las denominadas convexas.

Definición 4. Una función $F : \mathbb{R}^N \longmapsto \bar{\mathbb{R}}$ es *convexa* si, para todo $x, y \in \text{dom}(F)$ y $\lambda \in (0, 1)$, se tiene que

$$F(\lambda x + (1 - \lambda)y) \leq \lambda F(x) + (1 - \lambda)F(y). \tag{2.1}$$

Geométricamente, la desigualdad (2.1) es interpretada como el segmento que une $(x, f(x))$ e $(y, f(y))$, está por arriba del gráfico de F. Una función es *estrictamente convexa* si (2.1) se cumple de manera estricta cuando $x \neq y$. La figura 2.2 muestra una función convexa, estrictamente convexa y no convexa.



(a) (b) (c)

FIGURA 2.2: Una función (a) Convexa, (b) Estricta convexa y (c) No convexa.

Esta propiedad facilita encontrar minimizadores para F, pues su geometría nos da indicios sobre en cual dirección podemos encontrar dichos puntos.

2.2. Algunas propiedades de regularidad

Definición 5. Una función $F : A \subset \mathbb{R}^N \to \mathbb{R}$ es *semi continua inferior* (sci) en $x_0 \in A$ si:

$$\liminf_{x \to x_0} F(x) \geq F(x_0).$$

La Figura 2.3 ilustra como se ve una función semi continua inferior en el punto x_0.

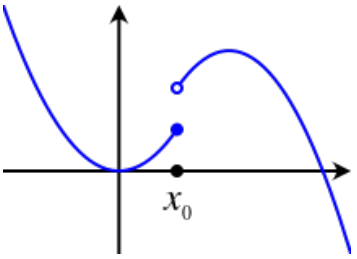


FIGURA 2.3: Ejemplo de función sci en x_0.

Definición 6. Una función $F : A \subset \mathbb{R}^N \to \mathbb{R}$ es *continua* en $x_0 \in A$ si

$$\forall \epsilon > 0 \quad \exists \delta > 0 \quad \text{tal que} \quad \|x - y\| \leq \delta \Rightarrow \|F(x) - F(y)\| \leq \epsilon, \quad \text{para} \quad y \in A.$$

Definición 7. Una función $F : \mathbb{R}^N \to \mathbb{R}$ es *Lipschitz-continua* con constante L si

$$\|F(x) - F(y)\| \leq L\|x - y\|, \forall x, y \in \mathbb{R}^N.$$

Si F es convexa, podemos establecer equivalencias entre estas dos propiedades.

Proposición 1. Sea $F : \mathbb{R}^N \to \bar{\mathbb{R}}$ una función convexa y $x_0 \in \mathbb{R}^N$ fijo. Las siguientes expresiones son equivalentes.

i) F es acotada superiormente en una vecindad de x_0.

ii) F es Lipschitz-continua en una vecindad de x_0.

iii) F es continua en x_0.

iv) F es continua en $\text{int}(\text{dom}(F))$ y $x_0 \in \text{int}(\text{dom}(F))$.

Además, cuando la función F está definida sobre $\mathbb{R}^N$, el cual es un espacio (Banach) de dimensión finita (como en (1.1)), la Proposición 1 garantiza que F es continua en el interior

de su dominio (como consecuencia de [26, Proposición 3.5]). Para poder caracterizar las soluciones del problema (1.1), la continuidad de la función F no es suficiente, por lo que debemos introducir algunas nociones de diferenciablidad para las funciones que componen a F, es decir, para f y ψ.

Definición 8. Consideremos $A \subset \mathbb{R}^N$ un conjunto abierto no vació y una función $f : A \to \mathbb{R}$. La *derivada direccional* de f en $x \in A$ en la dirección $d \in \mathbb{R}^N$ está definida por

$$f'(x; d) = \lim_{t \to +0} \frac{f(x + td) - f(x)}{t},$$

cuanto este límite existe.

La función f es *Gâteaux diferenciable* en x si $f'(x; d)$ existe para toda dirección $d \in \mathbb{R}^N$ y la función $d \longmapsto f'(x; d)$ es lineal y continua. En este contexto, la derivada direccional es llamada *gradiente* de f y $\nabla f(x) = f'(x; \cdot)$. Si el gradiente de f es Lipschitz-continuo, podemos obtener una aproximación de primer orden para los valores de la función.

Lema 1. Si $f : \mathbb{R}^N \to \mathbb{R}$ es diferenciable y ∇f es Lipschitz-continua con constante L, entonces

$$f(y) \leq f(x) + \langle \nabla f(x), y - x \rangle + \frac{L}{2} \|y - x\|^2 \tag{2.2}$$

para cada $x, y \in \mathbb{R}^N$. En particular f es continua.

Proposición 2. Sea $A \subset \mathbb{R}^N$ un conjunto abierto y convexo, y $f : A \to \mathbb{R}$ una función diferenciable. Entonces las siguientes expresiones son equivalentes:

i) f es convexa.

ii) $f(y) \geq f(x) + \langle \nabla f(x), y - x \rangle$ para cada $x, y \in A$.

iii) $\langle \nabla f(x) - \nabla f(y), x - y \rangle \geq 0$ para cada $x, y \in A$.

Si f es estrictamente convexa, las desigualdades se satisfacen de manera estricta (cuando $x \neq y$).

La proposición 2 parte (ii), nos da una aproximación lineal de la función f en el punto x, que va por debajo del gráfico de la función, con pendiente $\nabla f(x)$ (ver figura 2.4). Cuando la función es convexa y diferenciable en x, esta aproximación está definida de manera única, debido a que $\nabla f(x)$ es la única pendiente que satisface dicha propiedad, pero cuando la función solo es convexa, esto no es necesariamente cierto. En el problema 1.1, la función ψ puede no ser diferenciable (por ejemplo $\psi(x) = \|x\|_1$), por lo que es necesario definir un concepto que generalice al gradiente. Para ello utilizamos esta idea de pendientes de aproximaciones lineales.

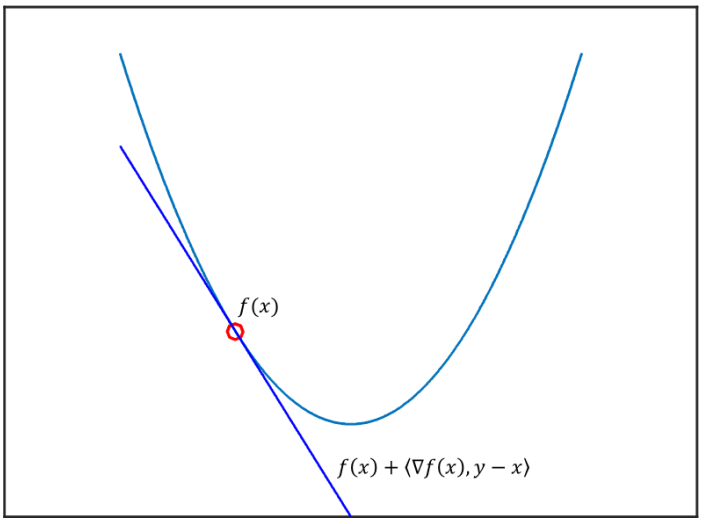


FIGURA 2.4: Aproximación lineal dada por el gradiente de la función convexa.

Definición 9. Un punto $\bar{x} \in \mathbb{R}^N$ es un subgradiente de ψ en el punto $x \in \mathbb{R}^N$ si

$$\psi(y) \geq \psi(x) + \langle \bar{x}, y - x \rangle, \quad \forall y \in \mathbb{R}^N.$$

El conjunto de todos los subgradientes de ψ en x se denomina el *subdiferencial* de ψ en x y es denotado por $\partial\psi(x)$.

Si $\partial\psi(x) \neq \emptyset$ decimos que ψ es *subdiferenciable* en el punto x. El subdiferencial $\partial\psi(x)$ corresponde a las pendientes de todas las aproximaciones lineales de ψ en x que van por debajo del gráfico de la función. El ejemplo tradicional para el subdiferencial es considerar la función $\psi(x) = |x|$, la cual no es diferenciable en 0, pero su subdiferencial $\partial\psi$ está dado por

$$\partial\psi(x) = \begin{cases} -1, & \text{if} \quad x < 0 \\ [-1, 1], & \text{if} \quad x = 0 \\ 1, & \text{if} \quad x > 0. \end{cases}$$

Como nuestra función objetivo es de la forma $F = f + \psi$, donde f es diferenciable y ψ es subdiferenciable, necesitamos establecer cómo opera el subdiferencial para la suma de funciones.

Teorema 1. *Moreau-Rockafellar* Sea $f, \psi : \mathbb{R}^N \to \bar{\mathbb{R}}$ una función propia y convexa. Si f es continua en algún punto $x_0 \in \text{dom}(\psi)$, entonces

$$\partial(f + \psi)(x) = \partial f(x) + \partial\psi(x).$$

Nuestra configuración para el problema 1.1 satisface el Teorema 1, así tenemos que para nuestra función F el subdiferencial está dado por

$$\partial F(x) = \nabla f(x) + \partial \psi(x).$$

2.3. Nociones de optimización

Definición 10. Para el problema general de optimización

$$\min_{x \in \mathbb{R}^N} F(x),$$

se definen:

- El conjunto solución, $C^* \stackrel{\text{def}}{=} \operatorname{argmin}(F) = \{x \in \mathbb{R}^N : F(x) \leq F(y), \quad \forall y \in \mathbb{R}^N\}$.
- El valor óptimo, $F^* \stackrel{\text{def}}{=} F(x^*)$, donde $x^* \in C^*$.

. El siguiente teorema nos permite caracterizar soluciones para 1.1.

Teorema 2. (Regla de Fermat) La función F posee un mínimo en el punto x^* si y solo si $0 \in \partial F(x^*)$.

El teorema anterior no nos permite identificar fácilmente si 1.1 posee (o no) soluciones, por lo que haremos uso del siguiente resultado, el cual nos permite garantizar minimizadores para ciertos tipos de funciones.

Teorema 3. (de Weierstrass - en $\mathbb{R}^N$) Si la función $F : \mathbb{R}^N \to \bar{\mathbb{R}}$ es propia, coerciva y sci, entonces $\operatorname{argmin}(F)$ posee, al menos, un elemento.

Sumado a lo anterior, si la función F es estrictamente convexa, entonces $\operatorname{argmin}(F)$ posee un solo elemento. Para poder tener una noción de la velocidad con la que nuestros algoritmos obtienen una solución de (1.1) usaremos las siguientes definiciones.

Definición 11.

1. Decimos que F es de orden $O(g)$ si:

$$\lim_{k \to \infty} \frac{F(k)}{g(k)} < \infty.$$

2. Decimos que F es de orden $o(g)$ si:

$$\lim_{k \to \infty} \frac{F(k)}{g(k)} = 0.$$

Si consideramos $g(k) = k^{-2}$, el primer caso nos dice que F va a cero a la misma velocidad que k^{-2}, mientras que en el segundo caso, decimos que F va a cero más rápido que k^{-2}.

2.4. Métodos iterativos

En esta sección, daremos los conceptos generales sobre métodos iterativos, los que permiten resolver computacionalmente el problema (1.1). Un método iterativo es un algoritmo que genera una sucesión $\{x_k\}$, desde un punto inicial x_0 de acuerdo a una regla

$$x_{k+1} = G_k(x_k),$$

donde $G_k : \mathbb{R}^N \longmapsto \mathbb{R}^N$ es una función que puede depender de k. La elección de G_k depende de la regularidad de la función objetivo F, por ejemplo, si nuestra función es diferenciable, se suelen considerar funciones G_k que aprovechen la información entregada por el gradiente de la función. Por otro lado, si la función solo es convexa, podemos usar algoritmos basados en el subgradiente. Nuestra función objetivo posee componentes con ambas propiedades, por lo que nuestra elección de G_k debe tomar ventaja tanto de ∇f como de $\partial\psi$.

Nosotros usaremos el método gradiente-proximal (ver [3],[14]), el cual combina dos métodos ampliamente conocidos, el de máximo descenso [29] y punto proximal [10], siendo capaz de usar las diferentes propiedades de las funciones f y ψ.

2.4.1. El método gradiente-proximal.

En esta sección, presentamos el método gradiente-proximal para resolver el problema de optimización convexo

$$\min_{x\in\mathbb{R}^N} F(x) = f(x) + \psi(x), \tag{2.3}$$

donde $f : \mathbb{R}^N \longmapsto \mathbb{R}$ es una función propia, convexa y diferenciable, mientras que la función $\psi : \mathbb{R}^N \to \mathbb{R}$ es propia, convexa y no necesariamente diferenciable. El método gradiente-proximal puede ser visto como una composición del método del máximo descenso y el método del punto proximal.

El método de máximo descenso utiliza el gradiente de la función f, realizando en cada iteración un paso de longitud $\theta_k > 0$ en la dirección $-\nabla f(x_k)$, la cual es la dirección de máximo descenso a partir del punto x_k. La k-ésima iteración de este método es de la forma

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

Para la convergencia de este algoritmo, nos basta que ∇f sea Lipschitz-continuo con constante L y $\theta := \sup_{k\in\mathbb{N}} \theta_k < \dfrac{2}{L}$.

Por otra parte, el método del punto proximal es generalmente usado para funciones no diferenciables, haciendo uso de un operador auxiliar llamado *operador proximal*.

Definición 12. Sea $\psi : \mathbb{R}^N \to \bar{\mathbb{R}}$ una función propia y convexa. El operador proximal de ψ con parametro θ, $\text{Prox}_{\psi,\theta} : \mathbb{R}^N \to \mathbb{R}^N$ se define como

$$\text{Prox}_{\psi,\theta}(x) = \min_{y\in\mathbb{R}^N}\{\psi(y) + \frac{1}{2\theta}\|x-y\|^2\}. \tag{2.4}$$

La función $\|x - \cdot\|^2$ es estrictamente convexa y coerciva, por lo que el problema de optimización al lado derecho de (2.4) posee solución única (ver teorema 3). El método del punto proximal resuelve el problema (2.4) en cada iteración, donde la k-ésima iteración está dada por:

$$\{x_{k+1}\} = \text{Prox}_{\psi,\theta_k}(x_k) = \min_{y\in\mathbb{R}^n}\{\psi(y) + \frac{1}{2\theta_k}\|x_k-y\|^2\}.$$

Finalmente, el método gradiente-proximal consiste en que, en cada iteración, se hace un paso gradiente para la función f y luego aplicar el operar proximal $\text{Prox}_{\psi,\theta_k}$ al paso gradiente $(x_k - \theta_k\nabla f(x_k))$, por lo que el paso principal de este método está dado por:

$$\begin{aligned}
\{x_{k+1}\} &= \text{Prox}_{\psi,\theta_k}(x_k - \theta_k\nabla f(x_k)) \\
&= \operatorname*{argmin}_{y\in\mathbb{R}^N} \psi(y) + \frac{1}{2\theta_k}\|\,(x_k - \theta_k\nabla f(x_k)) - y\|^2, \\
&= \operatorname*{argmin}_{y\in\mathbb{R}^N} \psi(y) + \frac{1}{2\theta_k}\left(\theta_k^2\|\nabla f(x_k)\|^2 - 2\theta_k\langle\nabla f(x_k), x_k - y\rangle + \|x_k-y\|^2\right), \\
&= \operatorname*{argmin}_{y\in\mathbb{R}^N} \psi(y) + \langle\nabla f(x_k), y - x_k\rangle + \frac{1}{2\theta_k}\|x_k-y\|^2.
\end{aligned}$$

Para la convergencia de este método también se necesita que $\nabla f(x_k)$ sea Lipschitz-continuo con constante L y $\theta := \sup_{k\in\mathbb{N}}\theta_k < \frac{2}{L}$. El método gradiente-proximal puede ser escrito como:

Algorithm 1 Gradiente-Proximal

1: Elegir $x_0 \in \mathbb{R}^N$ y $\theta_0 < \frac{1}{L}$
2: **for** $k \geq 0$ **do**
3: $\quad x_{k+1} = \operatorname*{argmin}_{y\in\mathbb{R}^N} \psi(y) + \langle\nabla f(x_k), y - x_k\rangle + \frac{1}{2\theta_k}\|x_k-y\|^2.$
4: $\quad$ Actualizar $\theta_{k+1} < \frac{2}{L}$.
5: **end for**

Cuando $\psi(x) = \|x\|_1$ este algoritmo se conoce también como *ISTA* (Iterative Shrinkage-Thresholding Algorithm, ver [9],[8]). Para este método, tenemos el siguiente resultado de convergencia.

Lema 2. El método gradiente-proximal, con parámetro $\theta_k < \dfrac{2}{L}$ satisface

$$0 \leq F(x_k) - F^* \leq \frac{\|x_0 - x^*\|^2}{2k\theta_k}.$$

En general, esta clase de algoritmos no son suficientes para resolver problemas de gran cantidad de variables. En la siguiente sección mencionaremos algunas modificaciones al método gradiente-proximal de manera que sea capaz de resolver problemas con gran cantidad de variables.

2.5. Mejorando el algoritmo gradiente-proximal

Los métodos clásicos de optimización, como el gradiente-proximal, presentan limitaciones que impiden su uso para resolver problemas con una gran cantidad de variables, pues las operaciones relacionadas al gradiente o subgradiente sobre todo el espacio de variables implica un gran costo computacional, comprometiendo el desempeño de estos métodos.

Es por esto que el desarrollo de variantes que mejoren las tasas de convergencia y reduzcan la complejidad computacional de cada iteración, evitando operaciones que involucren a todas las variables, ha cobrado mayor relevancia en el último tiempo. En particular estamos interesado en dos enfoques: Los métodos acelerados desarrollados por Nesterov (ver [21]), el cual mejora la tasa de convergencia del método gradiente-proximal sin aumentar el costo computacional; y los métodos de descenso por bloque en junto con su variante aleatoria (ver [23]), los cuales actualizan solo una cierta cantidad de variables en cada iteración.

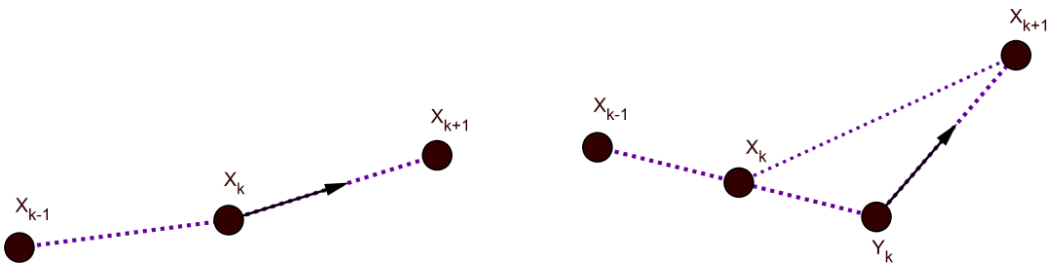
2.5.1. Método acelerado de Nesterov

El método acelerado de Nesterov es una modificación del método gradiente-proximal, desarrollado en los años ochenta para el caso discreto [21], logrando mayor relevancia al ser extendido a problemas de la forma $F(x) = f(x) + \psi(x)$ (ver[24], [25]). La idea es usar un punto auxiliar y_k, definido como una combinación lineal entre x_k and x_{k-1}. Para el método gradiente-proximal, el paso principal de su versión acelerada es de la forma

$$\begin{cases} y_k & = x_k + \frac{k-1}{k+2}(x_k - x_{k-1}) \\ x_{k+1} & = \mathrm{Prox}_{\psi,\theta_k}(y_k - \theta_k \nabla f(y_k)). \end{cases}$$

Mientras que el método gradiente-proximal itera desde el punto actual x_k (Figura 2.5 (a)), su versión acelerada se mueve en la dirección anterior (desde x_k a y_k) para luego iterar y

moverse hacia x_{k+1} (Figura 2.5 (b)). Este algoritmo, cuando $\psi(x) = \|x\|_1$, también se conoce como *FISTA* (Fast ISTA, ver [2]).



(a) (b)

FIGURA 2.5: Una interación del método gradiente-proximal (a)Normal y (b) Acelerada.

Con esta pequeña modificación, este método posee una tasa de convergencia de $O\left(\frac{1}{k^2}\right)$, en vez de $O\left(\frac{1}{k}\right)$ (ver [21, Teorema 1]).

2.5.2. Métodos de descenso por bloque aleatorios

Los *block coordinate descent methods* (BCD) o *métodos de descenso por bloque* [23], [34], son técnicas basadas en algoritmos clásicos de optimización, como método gradiente-proximal, que evitan realizar operaciones sobre todo el vector de decisión x, actuando solo en algunas componentes, las cuales suelen ser elegidas según un criterio determinista. Usualmente se considera usar un criterio cíclico, es decir, las componentes son actualizadas en un orden predeterminado; También se pueden elegir las componentes en las que el gradiente presenta mayor magnitud.

La versión clásica de estos métodos selecciona la coordenada a actualizar en cada iteración de manera cíclica. Otra posibilidad es moverse en la dirección que presenta mayor magnitud en la coordenada del gradiente. Estas versiones presentan tasas de convergencia peores al método gradiente tradicional, pues al no actualizar todos los bloques por iteración, su convergencia depende de la cantidad de bloques que se actualizan simultáneamente. Sin embargo, diversos trabajos han logrado disminuir esta dependencia, logrando versiones cada vez más rápidas. (ver [30], [17]).

Por otra parte, los *random block coordinate descend methods* (RBCD) o *métodos de descenso por bloque aleatorios* [23] seleccionan los bloques que se actualizan en cada iteración según alguna regla estocástica, generalmente uniforme (cada bloque tiene la misma probabilidad de ser seleccionado). Esta idea logra mejorar la convergencia, alcanzado tasas de $O(\frac{1}{k})$ y $O(\frac{1}{k^2})$ en versiones aceleradas, pero en promedio, por lo que hay que correr el algoritmo varias veces. Matemáticamente, en vez de buscar cotas para $F(x_k) - F(x^*)$, trabajamos con

$E\left[F(x_k) - F(x^*)\right]$, donde $E[\cdot]$ denota el operador esperanza con respecto a la selección de bloques.

Nuestra intención es trabajar con una versión del método gradiente-proximal que utilice las ideas de los RBCD. Para esto consideramos la siguiente notación e hipótesis.

2.5.2.1. Notación de bloque

Supongamos que se tiene una variable de decisión $x \in \mathbb{R}^4$, el cual posee tres bloques, por ejemplo $x = \left(x^{(1)}, x^{(2)}, x^{(3)}\right)$, donde $x^{(1)}, x^{(3)} \in \mathbb{R}$ y $x^{(2)} \in \mathbb{R}^2$. Para lidiar con este tipo de problemas, introduciremos los elementos principales de descomposición por bloque [13].

Sea $x \in \mathbb{R}^N$ con $n \leq N$ bloques y $U = [U_1, \ldots, U_n]$ la matriz identidad de tamaño $N \times N$, donde cada $U_i \in \mathbb{R}^{N \times N_i}$ está asociada al i-ésimo bloque. En nuestro ejemplo, la matriz identidad se descompone de la siguiente forma:

$$U = \left[\begin{array}{c|cc|c} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{array}\right].$$

De esta manera, el i-ésimo bloque del vector $x \in \mathbb{R}^N$ se puede escribir como $x^{(i)} = U_i^T x \in \mathbb{R}_i^N$, con $i = 1, \ldots, n$ y así, la siguiente descomposición por bloques está definida de manera única:

$$x = \sum_{i=1}^{n} U_i x^{(i)}.$$

En vez de actualizar cada uno de los n subvectores $x^{(i)}$, solo los pertenecientes a un subconjunto $\emptyset \neq S \subset [n] := \{1, \ldots, n\}$ son considerados, dando, para un vector $h \in \mathbb{R}^N$, la siguiente S-descomposición:

$$h_{[S]} = \sum_{i \in S} U_i h^{(i)},$$

este es un vector N-dimensional, en donde todas las componentes de h que no pertenecen a S son cero. En nuestro ejemplo, si $S = \{2, 3\}$, entonces

$$h_{[S]} = \begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} h_2 \\ h_3 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} h_4 = \begin{bmatrix} 0 \\ h_2 \\ h_3 \\ h_4 \end{bmatrix}.$$

Para el bloque $i \in [n]$, definimos el bloque-gradiente asociado, como el i-ésimo bloque del vector gradiente, es decir,

$$\nabla_i f(x) := (\nabla f(x))^{(i)} = U_i^T \nabla f(x) \in \mathbb{R}^{N_i}. \tag{2.5}$$

Además, para cada bloque $i \in [n]$, consideramos la matriz definida positiva $B_i \in \mathbb{R}^{N_i \times N_i}$ y el escalar positivo v_i para dotar a los espacios $\mathbb{R}^{N_i}$ y $\mathbb{R}^N$ con las normas

$$\|x^{(i)}\|_{(i)} = \left\langle B_i x^{(i)}, x^{(i)} \right\rangle^{1/2}, \quad \text{y} \quad \|x\|_v = \left(\sum_{i=1}^{n} v_i \|x^{(i)}\|_{(i)} \right)^{1/2},$$

respectivamente. Además, las normas conjugadas están dadas por

$$\|x^{(i)}\|_{(i)}^* = \left\langle B_i^{-1} x^{(i)}, x^{(i)} \right\rangle^{1/2}, \quad \text{y} \quad \|x\|_v^* = \left(\sum_{i=1}^{n} v_i^{-1} \left(\|x^{(i)}\|_{(i)}^* \right)^2 \right)^{1/2}.$$

Supongamos que en la k-ésima iteración, seleccionamos un grupo S_k de bloques. La *esperanza condicional por bloque* $E_k[\cdot]$ es el operador esperanza con respecto a S_k, manteniendo el resto de los bloques fijos (sus valores son conocidos por el algoritmo). Por otro lado la *esperanza condicional* $E[\cdot]$ se considera en el sentido usual, donde todos los bloques son aleatorios. La propiedad de proyección (ver [11, Teorema 5.1.5]) nos da la siguiente relación entre estos operadores:

$$E[E_k[\cdot]] = E[\cdot]. \tag{2.6}$$

Las hipótesis principales que utilizaremos sobre los RBCD son:

Hipótesis 1. Para el problema (1.1) y una muestra de bloques S_k, suponemos que.

1. La función f, el primer término de la función objetivo, es convexa y diferenciable.
2. Para cada $j = 1, 2, \ldots, m$ y $i = 1, 2, \ldots, n$ las funciones por bloque f_j dependen de un conjunto de bloques C_j y los gradientes por bloque $\nabla_i f_j$ son Lipschitz-continuos con constante $L_{ji} \geq 0$:

$$\|\nabla_i f_j(x + U_i t) - \nabla_i f_j(x)\|_{(i)}^* \leq L_{ji} \|t\|_{(i)}, \quad x \in \mathbb{R}^N, t \in \mathbb{R}^{N_i}. \tag{2.7}$$

3. Suponemos que los bloques se eligen aleatoriamente de manera uniforme, es decir, cada muestra de bloques $S_k \subset [n]$ posee la propiedad: $P(i \notin S_k) = P(j \notin S_k)$ para todo $i, j \in [n]$.
4. Definamos por τ al número esperado de bloques que se actualizan en cada iteración, es decir, $\tau := E[|S_k|] \leq n$. Supondremos que existen constantes (que se pueden calcular)

$v = (v_1, \ldots, v_n) > 0$ tal que

$$E\left[f\left(x + h_{[S_k]}\right)\right] \leq f(x) + \frac{\tau}{n}\left(\langle \nabla f(x), h\rangle + \frac{1}{2}\|h\|_v^2\right), \quad x, h \in \mathbb{R}^N. \tag{2.8}$$

Esta desigualdad recibe el nombre de *Expected Separable Overapproximation* (ESO).

Notemos que la condición de Lipschitz por bloque nos obliga a definir $L_{ji} = 0$ en cada $i \notin C_j$. Además, en el caso determinista, en el cual $\tau = n$, la desigualdad (2.8) es equivalente a que la función f sea Lipschitz-continua (ver ecuación (2.2)) con respecto a la norma $\|\cdot\|_{(1)}$ y con constantes de Lipschitz v en cada uno de los bloques.

En el siguiente capítulo hablaremos del algoritmo APProx, el cual es un RCBD basado en el método gradiente-proximal y sobre cómo podríamos acelerar aún más las variantes aceleradas de estos dos métodos. Propondremos una variante de APProx, la cual, conjeturamos, posee una tasa de convergencia mayor a $O(\frac{1}{k^2})$ y hereda sus características, tales como englobar una gran cantidad de técnicas clásicas de optimización, como el método clásico del gradiente y su versión conjugada, métodos proximales y el gradiente proximal.

Capítulo 3

APPROX y aAPPROX

En este capítulo daremos las nociones detrás del mejoramiento en la tasa de convergencia para el algoritmo APProx. Revisaremos brevemente el caso para el algoritmo gradiente-proximal, el cual fue discutido en [1], para luego extrapolar estas ideas para el caso de interés. Finalmente mencionaremos un caso aún más general, el cual contempla los dos mencionados, además del esquema que, conjeturamos, permitiría probar el resultado correspondiente.

3.1. El algoritmo APPROX

El algoritmo acelerado paralelo proximal o APProx (ver [13]) pertenece a la familia de los métodos de descensos por bloque aleatorios, tomando como base la versión acelerada del método gradiente-proximal. Este algoritmo es conocido por ser uno de los primeros en ser: acelerado, en el sentido de poseer una tasa de convergencia de $O(\frac{1}{k^2})$; paralelo, pues permite actualizar bloques de manera simultánea mediante programación paralela; y proximal, lidiando con problemas de minimización tipo LASSO y Elastic Net, los cuales consideran como funciones de regularización a la norma L1.

Los parámetros $\{\theta_k\}$ utilizados para obtener esta tasa de convergencia satisfacen la desigualdad para variantes aceleradas de algoritmo, presente en [33], la cual estable que para $k = 0, 1, \ldots$ y $\theta_k \in (0, 1]$ se debe cumplir que:

$$\frac{1 - \theta_{k+1}}{\theta_{k+1}^2} \leq \frac{1}{\theta_k^2}. \tag{3.1}$$

En [13], la secuencia $\left\{\theta_k^{[13]}\right\}$ es generada al resolver (3.1) como una igualdad, obteniendo:

$$\theta_{k+1} = \frac{\sqrt{\theta_k^4 + 4\theta_k^2} - \theta_k^2}{2}. \tag{3.2}$$

Al comparar APProx con el método gradiente-proximal acelerado (ver algoritmo 1 y [22]), los pasos 3 y 7 coinciden, presentando solo diferencias en el uso de APProx de un punto auxiliar z_k (ver algoritmo 2).

El teorema 3 presente en [13] estable que, bajo las Hipótesis 1

$$E\left[F(x_k) - F(x^*)\right] \leq \frac{4n^2 M}{\left((k-1)\tau + 2n\right)^2}, \tag{3.3}$$

donde x^* es solución de (1.1) y M es una constante que depende de la diferencia entre $F(x_0)$ y $F(x^*)$, y la distancia entre los puntos x_0 y x^* con respecto a alguna norma. La relación (3.3) establece que, en valor esperado, APProx encuentra un valor óptimo para F tan rápido como $1/k^2$ converge a 0 cuando $k \to \infty$, es decir, posee una tasa de convergencia de $O\left(\frac{1}{k^2}\right)$.

Algorithm 2 APPROX: Accelerated Parallel Proximal Coordinate Descent

1: Elegir $x_0 \in \mathbb{R}^N$, fijar $z_0 = x_0$ y tomar

$$\theta_0^{[13]} = \frac{\tau}{n}. \tag{3.4}$$

2: **for** $k \geq 0$ **do**
3: $y_k = (1 - \theta_k)x_k + \theta_k z_k$
4: Elegir aleatoriamente un set de bloques S_k
5: $z_{k+1} = z_k$
6: **for** $i \in S_k$ **do**
7: $z_{k+1}^{(i)} = \underset{z \in \mathbb{R}^{N_i}}{\operatorname{argmin}} \left\{ \left\langle \nabla_i f(y_k), z - y_k^{(i)} \right\rangle + \frac{n\theta_k v_i}{2\tau} \|z - z_k^{(i)}\|_{(i)}^2 + \psi_i(z) \right\}$
8: **end for**
9: $x_{k+1} = y_k + \frac{n}{\tau}\theta_k(z_{k+1} - z_k)$
10: Actualizar los parámetros $\theta_{k+1} \in (0, 1]$ según

$$\theta_{k+1}^{[13]} = \frac{\sqrt{{\theta_k^{[13]}}^4 + 4{\theta_k^{[13]}}^2} - {\theta_k^{[13]}}^2}{2}.$$

11: **end for**

A continuación, mostraremos la idea principal detrás de la aceleración de APProx, la cual nace al estudiar las propiedades del método gradiente-proximal.

3.2. Mejorando el método gradiente-proximal

Como vimos anteriormente, el método gradiente-proximal posee una variante acelerada, la cual se obtiene al introducir un punto auxiliar y_k, pasando de una tasa de convergencia de $O\left(\frac{1}{k^1}\right)$ a $O\left(\frac{1}{k^2}\right)$, pero en realidad, aún existe un margen de mejora, a través de una pequeña modificación a la propuesta de Nesterov. Consideremos la siguiente modificación para el paso principal del método proximal gradiente:

$$\begin{cases} y_k & = x_k + \frac{k-1}{k+\alpha-1}(x_k - x_{k-1}) \\ x_{k+1} & = \mathrm{Prox}_{\psi,\theta_k}(y_k - \theta_k \nabla f(y_k)), \end{cases}$$

donde $\alpha > 0$. Si tomamos $\alpha = 3$ recuperamos la versión acelerada de Nesterov (Ver capitulo 2, sección 2.5.1). En [1] se prueba que, al considerar $\alpha > 3$, la tasa de convergencia mejora. Si al considerar $\alpha = 3$ se tiene que el algoritmo converge al valor óptimo al igual que $1/k^2$ converge a cero cuando $k \to \infty$ $\left(O(\frac{1}{k^2})\right)$, al tomar $\alpha > 3$ este algoritmo es ligeramente más rápido que k^{-2} $\left(o(\frac{1}{k^2})\right)$, es decir:

$$\lim_{k\to\infty} \left(F(x_k) - F(x^*)\right) k^2 = 0.$$

Este resultado es obtenido en [1] al estudiar las propiedades del sistema dinámico asociado al paso principal del método gradiente-proximal y su comportamiento para diferentes valores de $\alpha > 0$:

$$\ddot{x}(t) + \frac{\alpha}{t}\dot{x}(t) + \partial\psi(x(t)) + \nabla f(x(t)) \in 0.$$

Con esta misma idea, agregando el parámetro α a APProx, conjeturamos que, con un reemplazo apropiado para θ_k, es posible obtener una mejora en la tasa de convergencia, tal que

$$\lim_{k\to\infty} k^2 \Big(E\left[F(x_k) - F(x^*)\right] \Big) = 0\,,$$

donde un diferenciador respecto al resulto presente en [1] es el supuesto de que α no necesariamente debe ser fijo, si no que podría variar en cada iteración.

3.3. El algoritmo aAPProx

Comenzamos discutiendo un criterio alternativo para θ_k, que incluya al parámetro α como variable en cada iteración, además estableceremos algunas relaciones, a modo de presentar un bosquejo que, teorizamos, permitiría demostrar el resultado principal, obteniendo una versión acelerada de APProx que denominamos **aAPProx**.

3.3.1. Incremento inverso

Para justificar de forma clara nuestra elección del parámetro θ_k, daremos un criterio equivalente a (3.2) para el inverso del parámetro, es decir, para

$$(0 <) \quad \nu_k^{[13]} := \frac{1}{\theta_k^{[13]}}\,.$$

Como los parámetros dados en [13] satisfacen la relación (3.1) en la igualdad,

$$\Big(1-\frac{1}{\nu_{k+1}^{[13]}}\Big)(\nu_{k+1}^{[13]})^2 = (\nu_k^{[13]})^2 \iff (\nu_{k+1}^{[13]})^2-\nu_{k+1}^{[13]} = (\nu_k^{[13]})^2 \implies \nu_{k+1}^{[13]} = \frac{1}{2}+\sqrt{\frac{1}{4}+(\nu_k^{[13]})^2}\,.$$

De esta manera, el incremento inverso, definido a continuación, es estrictamente mayor a un medio:

$$\mathtt{V}_k^{[13]} := \nu_{k+1}^{[13]} - \nu_k^{[13]} = \frac{1}{2} + \sqrt{\frac{1}{4}+(\nu_k^{[13]})^2} - \nu_k^{[13]} > \frac{1}{2}\,.$$

Como se puede apreciar en el Lema 3, con nuestra elección de parámetros, el incremento inverso es estrictamente menor a un medio, lo que equivale a aumentar la diferencia $\theta_{k+1}-\theta_k$, con respecto a los parámetros mostrados en [13], lo que en teoría produciría la aceleración deseada para APProx.

Lema 3. Sea $0 < \mathtt{v} \leq \frac{1}{2} - \ell$ con $\ell \in (0, \frac{1}{2})$ y $\nu_0 = \dfrac{n}{\tau}$, consideramos la siguiente actualización para los parámetros inversos

$$\begin{cases} \text{tomar} & & \nu_0 & = & \frac{n}{\tau}\,, \\ \text{para} & k \geq 0 \text{ considerar} & \nu_{k+1} & := & \nu_k + \mathtt{V}_k \quad \text{con} \quad \mathtt{v} \leq \mathtt{V}_k \leq \frac{1}{2} - \ell\,. \end{cases} \tag{3.5}$$

Esta secuencia, además de ser estrictamente creciente, satisface la siguiente desigualdad para todo $k \geq 0$:

$$\nu_{k+1}^2 - \nu_{k+1} \leq \nu_k^2 - 2k\ell\mathtt{v}\,. \tag{3.6}$$

Demostración. Los parámetros inversos $\{\nu_k\}$ son estrictamente crecientes, pues $\mathtt{V}_k \geq \mathtt{v} > 0$. El lado izquierdo en la ecuación (3.6), al usar el criterio (3.5) es equivalente a

$$\begin{aligned}\nu_{k+1}^2 - \nu_{k+1} &= (\nu_k + \mathtt{V}_k)^2 - (\nu_k + \mathtt{V}_k) \\ &= \nu_k^2 + 2\mathtt{V}_k\nu_k + \mathtt{V}_k^2 - \nu_k - \mathtt{V}_k \\ &= \nu_k^2 + \Big(\mathtt{V}_k(\mathtt{V}_k - 1) + (2\mathtt{V}_k - 1)\nu_k\Big)\,.\end{aligned}$$

Como $\mathtt{V}_k \in [\mathtt{v}, \frac{1}{2} - \ell]$, existe algún $\eta_k \in [0,1]$ tal que $V_k = \mathtt{v} + \eta_k(\frac{1}{2} - \ell - \mathtt{v})$. Aplicando (3.5) de manera recursiva, obtenemos $\nu_k = \nu_0 + k\mathtt{v} + (\frac{1}{2} - \ell - \mathtt{v})\sum_{j=0}^{k-1}\eta_j$, por lo que

$$\nu_k \geq k\mathtt{v}\,, \tag{3.7}$$

además $\mathtt{V}_k \leq \frac{1}{2} - \ell$ es equivalente a

$$(1 - 2\mathtt{V}_k) \geq 2\ell\,.$$

Como las dos desigualdades anteriores son no negativas, al multiplicarlas obtenemos

$$(1 - 2\mathtt{V}_k)\nu_k \geq 2\ell k\mathtt{v}$$

o equivalentemente,

$$(2\mathtt{V}_k - 1)\nu_k \leq -2\ell k\mathtt{v}$$

y así

$$\begin{aligned}\mathtt{V}_k(\mathtt{V}_k - 1) &\leq \mathtt{V}_k(\tfrac{1}{2} - \ell - 1), \\ &\leq \mathtt{V}_k(-\tfrac{1}{2} - \ell), \\ &\leq -\mathtt{V}_k(\tfrac{1}{2} + \ell), \\ &\leq -\mathtt{v}(\tfrac{1}{2} + \ell), \quad (\mathtt{v} \leq V_k) \\ &\leq 0\,.\end{aligned}$$

lo que concluye la demostración. ■

Algunas observaciones de (3.6) útiles:

1. Con respecto a la elección de parámetros en [1], el incremento

$$\mathtt{V}_k^{[1]} = \frac{1}{\alpha - 1} \quad \text{donde } \alpha > 3\,,$$

corresponde a considerar $\mathtt{V}_k \equiv \mathtt{v}$ con $\mathtt{v} = \frac{1}{\alpha-1}$, en este caso, desde $\mathtt{v} \leq \frac{1}{2} - \ell$, obtenemos

$$\begin{aligned}
\frac{1}{2} - \ell &\geq \frac{1}{\alpha - 1}, \\
\alpha - 1 &\geq \frac{2}{1-2\ell}, \\
\alpha &\geq \frac{2}{1-2\ell} + 1, \\
&\geq \frac{1-2\ell+2}{1-2\ell}, \\
&\geq \frac{3-2\ell+(4\ell-4\ell)}{1-2\ell}, \\
&\geq \frac{3(1-2\ell)+4\ell}{1-2\ell}, \\
&\geq 3 + \frac{4\ell}{1-2\ell},
\end{aligned}$$

por lo que, la condición $\alpha > 3$ es equivalente a $\dfrac{4\ell}{1-2\ell} > 0$, lo cual es lo mismo que considerar $\ell \in (0, \frac{1}{2})$.

2. El lado izquierdo en (3.6) es igual a $\nu_{k+1}^2(1-\theta_{k+1})$. Por lo que la desigualdad (3.6), al ser re escrita para $k+1$ en vez de k, nos da

$$\nu_k^2(1-\theta_k) \leq \nu_{k-1}^2 - 2(k-1)\ell\,. \tag{3.8}$$

3.3.2. Incremento directo

La variante acelerada de APProx se obtiene al reemplazar (3.2) por el siguiente criterio de selección de parámetros:

Dado $\mathtt{v} \leq \frac{1}{2} - \ell$ con $\ell \in (0, \frac{1}{2})$,

$$\begin{cases} \text{tomar} & \theta_0 := \dfrac{\tau}{n}\,, \\ \text{para} \quad k \geq 0 \quad \text{actualizar} & \theta_{k+1} := \theta_k\Big(\dfrac{1}{1+\theta_k \mathtt{V}_k}\Big) \quad \text{con } \mathtt{v} \leq \mathtt{V}_k \leq \dfrac{1}{2} - \ell\,. \end{cases} \tag{3.9}$$

Las propiedades a continuación, que son consecuencias del Lema 3, nos dicen que (3.9) definen una subclase del criterio (3.1) presente en [13].

Corolario 1. Para la recursión definida en (3.9), las siguientes expresiones son verdaderas, para todo $k \geq 1$:

(i) La secuencia $\{\theta_k\}$ es estrictamente decreciente; en particular, $0 < \theta_k < \tau/n \leq 1$ para todo $k \geq 1$.

(ii) La desigualdad presente en (3.1) se satisface de manera estricta:

$$\frac{1-\theta_{k+1}}{\theta_{k+1}^2} \leq \frac{1}{\theta_k^2} - 2k\ell\mathtt{v} < \frac{1}{\theta^2}\,.$$

(iii) En la identidad

$$\frac{1}{\theta_{k+1}} = \frac{1}{\theta_k} + \mathtt{V}_k,$$

los incrementos satisfacen $\mathtt{v} \leq \mathtt{V}_k \leq \dfrac{1}{2} - \ell$.

Demostración. El primer y tercer punto vienen de reescribir el Lema 3. Para el segundo punto, notemos que

$$\frac{1-\theta_{k+1}}{\theta_{k+1}^2} = \frac{1}{\theta_{k+1}^2} - \frac{1}{\theta_{k+1}} = \nu_{k+1}^2 - \nu_{k+1}\,,$$

de esta manera, el punto (ii) es equivalente a (3.6). ■

El criterio (3.9) es una generalización del criterio presente en [1]. Notemos que, desde la elección de parámetros en [1] es equivalente a considerar, para todo $k \in \mathbb{N}$,

$$\mathtt{V}_k^{[1]} = \frac{1}{\alpha - 1} \quad \text{donde } \alpha > 3\,. \tag{3.10}$$

Como se vio anteriormente, esta elección satisface nuestras hipótesis y además,

$$\nu_{k+1}^{[1]} = \nu_k^{[1]} + \frac{1}{\alpha-1} = \nu_0^{[1]} + \frac{k+1}{\alpha-1} = \frac{n}{\tau} + \frac{k+1}{\alpha-1} = \frac{\frac{n}{\tau}(\alpha-1)+(k+1)}{\alpha-1}\,,$$

por lo que

$$\theta_{k+1}^{[1]} = \frac{\alpha-1}{k+1+(\alpha-1)\dfrac{n}{\tau}}\,.$$

Así, en el caso determinista en el que todos los bloques de variables son actualizados en cada iteración ($n = \tau$), la expresión anterior para θ_{k+1} coincide con la presente en [1] y APProx coincide con el método gradiente-proximal acelerado. Así, nuestra variante acelerada de APProx, **aAPProx**, luce de la siguiente manera:

Algorithm 3 aAPPROX: accelerated APPROX

Elegir $x_0 \in \mathbb{R}^N$, fijar $z_0 = x_0$ y tomar

$$\theta_0^{[13]} = \frac{\tau}{n}\,. \tag{3.11}$$

for $k \geq 0$ **do**
  $y_k = (1-\theta_k)x_k + \theta_k z_k$
  Elegir aleatoriamente un set de bloques S_k
  $z_{k+1} = z_k$
  for $i \in S_k$ **do**
    $z_{k+1}^{(i)} = \underset{z\in\mathbb{R}^{N_i}}{\operatorname{argmin}} \left\{ \left\langle \nabla_i f(y_k), z - y_k^{(i)} \right\rangle + \frac{n\theta_k v_i}{2\tau}\|z - z_k^{(i)}\|_{(i)}^2 + \psi_i(z) \right\}$
  end for
  $x_{k+1} = y_k + \frac{n}{\tau}\theta_k(z_{k+1} - z_k)$
  Actualizar los parámetros $\theta_{k+1} \in (0,1]$ según (3.9)
end for

3.3.3. Algunas aproximaciones útiles

Comenzamos con algunos resultados preliminares, presentes en [13], los cuales son parte fundamentales en la demostración, tanto en nuestro caso, como en [13].

Lema 4. Sea $\{x_k, z_k\}_{k\geq 0}$ la secuencia generada por APProx o aAPProx, entonces, para todo $k \geq 0$,

$$x_k = \sum_{l=0}^{k} \gamma_k^l z_l$$

donde los coeficientes $\gamma_k^0, \gamma_k^1, \ldots, \gamma_k^k$ son no negativos y suman 1. Así, x_k es una combinación convexa de los vectores $z_0, \ldots, z_k$. En particular, las constantes se definen de manera recursiva k, comenzando con $\gamma_0^0 = 1, \gamma_0^1 = 0, \gamma_1^1 = 1$ y para $k \geq 1$

$$\gamma_{k+1}^l = \begin{cases} (1-\theta_k)\gamma_k^l, & l = 0, \ldots, k-1 \\ \theta_k\left(1 - \frac{n}{\tau}\theta_{k-1}\right) + \frac{n}{\tau}(\theta_{k-1} - \theta_k), & l = k \\ \frac{n}{\tau}\theta_k, & l = k+1. \end{cases} \tag{3.12}$$

Más aun, para todo $k \geq 0$, tenemos la siguiente igualdad:

$$\gamma_{k+1}^k + \frac{n-\tau}{\tau}\theta_k = (1-\theta_k)\gamma_k^k.$$

La prueba de este resultado puede ser vista en [13, Lemma 2]. A través de Lema 4 podemos obtener cotas superiores para $\psi(x_k)$ y $F(x_k)$, dadas por:

$$\hat{\psi}_k \overset{\text{def}}{=} \sum_{l=0}^{k} \gamma_k^l \psi(z_l) \geq \psi\left(\sum_{l=0}^{k} \gamma_k^l z_l\right) = \psi(x_k), \quad \text{y} \quad \hat{F}_k \overset{\text{def}}{=} \hat{\psi}_k + f(x_k) \geq F(x_k). \tag{3.13}$$

Para los próximos resultados, definimos

$$\begin{aligned} \tilde{z}_{k+1} &\overset{\text{def}}{=} \operatorname*{argmin}_{z \in \mathbb{R}^N} \left\{ \psi(z) + \langle \nabla f(y_k), z - y_k \rangle + \frac{n\theta_k}{2\tau} \|z - z_k\|_{(i)}^2 \right\} \\ &= \operatorname*{argmin}_{z = \left(z^{(1)}, \ldots, z^{(n)}\right)} \sum_{i=1}^{n} \left\{ \psi_i(z^{(i)}) + \left\langle \nabla_i f(y_k), z^{(i)} - y_k^{(i)} \right\rangle + \frac{n\theta_k v_i}{2\tau} \|z^{(i)} - z_k^{(i)}\|_{(i)}^2 \right\}. \end{aligned}$$

Notemos que $\tilde{z}_{k+1}$ es el resultado del paso de optimización sobre todos los bloques. Usando la definición de $\tilde{z}_{k+1}$ y z_{k+1} (ver algoritmo 2), observamos que

$$z_{k+1}^{(i)} = \begin{cases} \tilde{z}_{k+1}^{(i)}, & i \in S_k, \\ z_k^{(i)}, & i \notin S_k. \end{cases}$$

Con esto, enunciamos los siguientes dos resultados, también presentes en [13]:

Lema 5. Sean $\xi(u) \overset{\text{def}}{=} f(y_k) + \langle \nabla f(y_k), u - y_k \rangle + \frac{n\theta_k}{2\tau} \|u - z_k\|_v^2$, entonces

$$\psi(\tilde{z}_{k+1}) + \xi(\tilde{z}_{k+1}) \leq \psi(x^*) + \xi(x^*) - \frac{n\theta_k}{2\tau} \|x^* - \tilde{z}_{k+1}\|_v^2. \tag{3.14}$$

Lema 6. Para cualquier $x \in \mathbb{R}^N$ y $k \geq 0$,

$$E_k\left[\|z_{k+1} - x\|_v^2 - \|z_k - x\|_v^2\right] = \frac{\tau}{n}\left(\|\tilde{z}_{k+1} - x\|_v^2 - \|z_k - x\|_v^2\right). \tag{3.15}$$

Más aun,

$$E_k\left[\psi(z_{k+1})\right] = \left(1 - \frac{\tau}{n}\right)\psi(z_k) + \frac{\tau}{n}\psi(\tilde{z}_{k+1}). \tag{3.16}$$

La demostración de los Lemas 5 y 6 pueden ser vistas en [33] y [13], respectivamente. El siguiente resultado es parte de la demostración presente en [13, Teorema 3].

Lema 7. Para todo $k \geq 0$ se tiene

$$E_k\left[\hat{F}_{k+1}\right] \leq (1 - \theta_k)\hat{F}_k + \theta_k F(x^*) + \frac{n^2\theta_k^2}{2\tau^2}\left(\|x^* - z_k\|_v^2 - E_k\left[\|x^* - z_{k+1}\|_v^2\right]\right). \tag{3.17}$$

Demostración. Usando (3.13) y la linealidad de $E_k[\cdot]$ tenemos que $E_k\left[\hat{F}_{k+1}\right] = E_k\left[\hat{\psi}_k\right] + E_k\left[f(x_k)\right]$,por lo que basta con acotar cada una de las componentes de $E_k\left[\hat{F}_{k+1}\right]$ para

obtener el resultado. Comenzamos con $E_k\left[\hat{\psi}_{k+1}\right]$:

$$\begin{aligned}
E_k\left[\hat{\psi}_{k+1}\right] &= \sum_{l=0}^{k+1} \gamma_{k+1}^l E_k\left[\psi(z_l)\right] \\
&= \sum_{l=0}^{k} \gamma_{k+1}^l \psi(z_l) + \gamma_{k+1}^{k+1} E_k[\psi(z_{k+1})] \\
&\overset{(3.12)}{=} \sum_{l=0}^{k} \gamma_{k+1}^l \psi(z_l) + \frac{n}{\tau}\theta_k E_k[\psi(z_{k+1})] \\
&\overset{(3.16)}{=} \sum_{l=0}^{k} \gamma_{k+1}^l \psi(z_l) + \frac{n}{\tau}\theta_k \left(\left(1-\frac{\tau}{n}\right)\psi(z_k) + \frac{\tau}{n}\psi(\tilde{z}_{k+1})\right) \\
&= \sum_{l=0}^{k} \gamma_{k+1}^l \psi(z_l) + \left(\frac{n}{\tau}-1\right)\theta_k\psi(z_k) + \theta_k\psi(\tilde{z}_{k+1}). \qquad (3.18)
\end{aligned}$$

Para $E_k\left[f(x_k)\right]$ usamos el hecho de que $x_{k+1} = y_k + h_{[S_k]}$, donde $h_{[S_k]} = \frac{n}{\tau}\theta_k(\tilde{z}_{k+1} - z_k)$, y la desigualdad ESO (2.8) para obtener:

$$\begin{aligned}
E_k[f(x_{k+1})] &\overset{(2.8)}{\leq} f(y_k) + \theta_k\left\langle \nabla f(y_k), \tilde{z}_{k+1} - z_k\right\rangle + \frac{n\theta_k^2}{2\tau}\|\tilde{z}_{k+1} - z_k\|_v^2 \\
&= (1-\theta_k)f(y_k) - \theta_k\left\langle \nabla f(y_k), z_k - y_k\right\rangle + \\
&\quad \theta_k\left(f(y_k) + \left\langle \nabla f(y_k), \tilde{z}_{k+1} - y_k\right\rangle + \frac{n\theta_k}{2\tau}\|\tilde{z}_{k+1} - z_k\|_v^2\right). \qquad (3.19)
\end{aligned}$$

De la definición de y_k (ver algoritmo 2), sabemos que $z_k = (1-\theta_k)x_k - y_k$, de donde podemos obtener que:

$$\theta_k(y_k - z_k) = ((1-\theta_k)x_k - y_k) + \theta_k y_k = (1-\theta_k)(x_k - y_k). \qquad (3.20)$$

Así, la cota de $E_k[\hat{F}_{k+1}]$ está dada por

$$\begin{aligned}
E_k[\hat{F}_{k+1}] &= E_k[\hat{\psi}_{k+1}] + E_k[f(x_{k+1})] \\
&\overset{(3.18)+(3.19)}{\leq} \sum_{l=0}^{k} \gamma_{k+1}^l \psi(z_l) + \left(\frac{n}{\tau} - 1\right) \theta_k \psi(z_k) + \theta_k \psi(\tilde{z}_{k+1}) + (1-\theta_k) f(y_k) \\
&\quad - \theta_k \langle \nabla f(y_k), z_k - y_k \rangle + \theta_k \left(f(y_k) + \langle \nabla f(y_k), \tilde{z}_{k+1} - y_k \rangle + \frac{n\theta_k}{2\tau} \|\tilde{z}_{k+1} - z_k\|_v^2 \right) \\
&= \sum_{l=0}^{k} \gamma_{k+1}^l \psi(z_l) + \frac{n-\tau}{\tau} \theta_k \psi(z_k) + (1-\theta_k) f(y_k) - \theta_k \langle \nabla f(y_k), z_k - y_k \rangle \\
&\quad \theta_k \left(\psi(\tilde{z}_{k+1}) + f(y_k) + \langle \nabla f(y_k), \tilde{z}_{k+1} - y_k \rangle + \frac{n\theta_k}{2\tau} \|\tilde{z}_{k+1} - z_k\|_v^2 \right) \\
&\overset{(3.14)}{\leq} \sum_{l=0}^{k} \gamma_{k+1}^l \psi(z_l) + \frac{n-\tau}{\tau} \theta_k \psi(z_k) + (1-\theta_k) f(y_k) - \theta_k \langle \nabla f(y_k), z_k - y_k \rangle \\
&\quad + \theta_k \left(\psi(x^*) + f(y_k) + \langle \nabla f(y_k), x^* - y_k \rangle \right) + \frac{n\theta_k}{2\tau} (\|x^* - z_k\|_v^2 - \|x^* - \tilde{z}_{k+1}\|_v^2).
\end{aligned} \tag{3.21}$$

Usando (3.20) podemos acotar $E_k\left[\hat{F}_{k+1}\right]$ aún más:

$$\begin{aligned}
E_k\left[\hat{F}_{k+1}\right] \overset{(3.20)+(3.21)}{\leq} & \sum_{l=0}^{k-1} \gamma_{k+1}^l \psi(z_l) + \left(\gamma_{k+1}^k + \frac{n-\tau}{\tau}\theta_k\right) \psi(z_k) \\
& + (1-\theta_k) f(y_k) + (1-\theta_k) \langle \nabla f(y_k), x_k - y_k \rangle \\
& + \theta_k \left(\psi(x^*) + f(y_k) + \langle \nabla f(y_k), x^* - y_k \rangle \right) \\
& + \frac{n\theta_k^2}{2\tau} \|x^* - z_k\|_v^2 - \frac{n\theta_k^2}{2\tau} \|x^* - \tilde{z}_{k+1}\|_v^2.
\end{aligned}$$

Finalmente, usando (3.13), obtenemos

$$\begin{aligned}
E_k\left[\hat{F}_{k+1}\right] &\leq (1-\theta_k)\hat{F}_k + \theta_k F(x^*) + \frac{n\theta_k^2}{2\tau} \left(\|x^* - x_k\|_v^2 - \|x^* - \tilde{z}_{k+1}\|_v^2 \right) \\
&= (1-\theta_k)\hat{F}_k + \theta_k F(x^*) + \frac{n\theta_k^2}{2\tau} \left(\|x^* - x_k\|_v^2 - E_k\left[\|x^* - z_{k+1}\|_v^2 \right] \right).
\end{aligned}$$

■

El Lema 7 marca el punto en el cual nuestra propuesta para demostrar la aceleración de APProx comienza a diferenciarse de la presente en [13], por lo que necesitaríamos un par de definiciones y resultados extras.

3.3.4. Resultados específicos de la aceleración

A continuación, presentamos los pasos específicos que, conjeturamos, permitirán acelerar el algoritmo APProx. Comenzamos obteniendo una ecuación similar a la presentada en [1, eq (16)], donde la diferencia primordial viene dada por la introducción de las características aleatorias presentes en APProx.

Lema 8. Sea x^* una solución del problema (1.1). Para los parámetros elegidos mediante el criterio (3.9), y la siguiente notación abreviada

$$\Delta F_k := E\left[\hat{F}_{k+1} - F(x^*)\right] \quad \text{and} \quad \Delta z_k := \frac{n^2}{2\tau^2} E\left[\|x^* - z_k\|_v^2\right] , \tag{3.22}$$

donde $\hat{F}_k$ este definido como en (3.13). Definimos, para cada $k = 0, 1, 2, \ldots$,

$$\varepsilon(k) := \nu_{k-1}^2 \Delta F_k + \Delta z_k \, , \tag{3.23}$$

la cual posee las siguientes propiedades:

(i) La secuencia $\{\varepsilon(k)\}$ con $k \in \mathbb{N}$ es no creciente, además $\lim_{k\to\infty} \varepsilon(k)$ existe. En particular $\varepsilon(k) \leq \varepsilon(0)$.

(ii) Para cada $k \in \mathbb{N}$, $\Delta F_k \leq \theta_{k-1}^2 \varepsilon(0)$ y $\Delta z_k \leq \varepsilon(0)$.

(iii) Además, $\displaystyle\sum_{k=1}^{\infty} (k-1)\Delta F_k \leq \frac{1}{2\ell \mathtt{v}} \varepsilon(1)$.

Demostración. Comenzamos tomando la esperanza a la ecuación (3.17) presente en el Lema 4, item (iv), y así,

$$E\Big[E_k\left[\hat{F}_{k+1}\right]\Big] \leq E\Big[(1-\theta_k)\hat{F}_k + \theta_k F(x^*) + \frac{n^2\theta_k^2}{2\tau^2}\left(\|x^* - z_k\|_v^2 - E_k\left[\|x^* - z_{k+1}\|_v^2\right]\right)\Big] .$$

Multiplicando por ν_k^2, y usando (2.6) obtenemos, en la notación abreviada (3.22),

$$\begin{aligned} \nu_k^2 E\left[\hat{F}_{k+1}\right] &\leq \nu_k^2(1-\theta_k)E\left[\hat{F}_k\right] + \nu_k^2\theta_k E\left[F(x^*)\right] + \frac{n^2}{2\tau^2}\left(E(\left[\|x^* - z_k\|_v^2\right] - E\left[\|x^* - z_{k+1}\|_v^2\right]\right) \\ &= \nu_k^2(1-\theta_k)E\left[\hat{F}_k\right] + \nu_k^2\theta_k E\left[F(x^*)\right] + (\Delta z_k - \Delta z_{k+1}) \, . \end{aligned}$$

Restando en ambos lados $\nu_k^2 E[F(x^*)]$, al usar nuevamente la notación abreviada (3.22),

$$\nu_k^2 \Delta F_{k+1} \leq \nu_k^2(1-\theta_k)\Delta F_k + (\Delta z_k - \Delta z_{k+1}) \, ,$$

Así, usando la desigualdad (3.8),

$$\nu_k^2 \Delta F_{k+1} \leq (\nu_{k-1}^2 - 2(k-1)\ell \mathtt{v})\Delta F_k + (\Delta z_k - \Delta z_{z+1}) \, .$$

Reordenando términos,

$$\nu_k^2 \Delta F_{k+1} + \Delta z_{k+1} + 2(k-1)\ell \mathtt{v} \Delta F_k \leq \nu_{k-1}^2 \Delta F_k + \Delta z_k \,, \tag{3.24}$$

Notemos que la desigualdad anterior es equivalente a

$$\varepsilon(k+1) + 2(k-1)\ell \mathtt{v} \Delta F_k \leq \varepsilon(k) \,. \tag{3.25}$$

Desde (3.25) podemos concluir:

- El punto (i): usando que $2(k-1)\ell \mathtt{v} \Delta F_k \geq 0$ para $k \geq 1$.
- El punto (ii): considerando que cada termino de $\varepsilon(k)$ es positivo y usando el punto(i).
- El punto (iii): realizando, desde la desigualdad (3.25), una suma telescópica para $\varepsilon(k)$.

■

En este punto, el Lema 8, item (ii), garantiza para aAPProx una tasa de convergencia de $O(\frac{1}{k^2})$, siendo la misma que la obtenida en [13] para APProx, salvo una diferencia en las constantes, pues si definimos $\mathtt{V}_k$ como en (3.10) y $\theta_{-1} = \theta_0$,como en nuestro caso, se tiene

$$\varepsilon(0) = \theta_0(F_0 - F(x^*)) + \frac{n^2}{2\tau^2}\|x_0 - x^*\|_v^2.$$

y así

$$\begin{aligned} E[F(x_k) - F(x^*)] \leq \Delta F_k &\leq \theta_{k-1}^2 \varepsilon(0), \\ &\leq \frac{(\alpha-1)\varepsilon(0)}{(k-1+(\alpha-1)n/\tau)^2}, \end{aligned}$$

lo cual es similar a (3.3), salvo las constantes.
El siguiente resultado, que toma inspiración directa de [1], sería la clave para pasar desde $O(\frac{1}{k^2})$ a $o(\frac{1}{k^2})$

Lema 9. $\lim_{k\to\infty} \left[(k+1)^2 \Delta F_{k+1} + \frac{1}{\mathtt{v}} \Delta z_{k+1}\right]$ existe.

Demostración. Multiplicando por $1/\nu_k^2$ la desigualdad (3.24), la relación $\nu_{k-1}^2 \leq \nu_k^2$ nos da

$$\Delta F_{k+1} \leq \Delta F_k + \frac{1}{\nu_k^2}(\Delta z_k - \Delta z_{k+1}) \,,$$

la desigualdad (3.5), nos permite obtener

$$\Delta F_{k+1} \leq \Delta F_k + \frac{1}{k^2 \mathtt{v}^2}(\Delta z_k - \Delta z_{k+1}) \,,$$

reordenando términos y multiplicando por k^2

$$k^2(\Delta F_{k+1} - \Delta F_k) \leq \frac{1}{\mathtt{v}^2}(\Delta z_k - \Delta z_{k+1})\,. \tag{3.26}$$

Mediante manipulación algebraica es posible obtener la siguiente desigualdad

$$\begin{aligned}(k+1)^2\Delta F_{k+1} - k^2\Delta F_k &= k^2(\Delta F_{k+1} - \Delta F_k) + (2k+1)\Delta F_{k+1},\\ &\leq k^2(\Delta F_{k+1} - \Delta F_k) + 2(k+1)\Delta F_{k+1},\end{aligned} \tag{3.27}$$

de esta forma, al usar la desigualdad (3.27) para acotar el lado izquierdo de (3.26) obtenemos

$$(k+1)^2\Delta F_{k+1} - k^2\Delta F_k - 2(k+1)\Delta F_{k+1} \leq \frac{1}{\mathtt{v}^2}(\Delta z_k - \Delta z_{k+1})\,.$$

Si definimos $\vartheta_k = k^2\Delta F_k + \frac{1}{\mathtt{v}^2}\Delta z_k$ y ordenamos los términos

$$\vartheta_{k+1} - \vartheta_k \leq 2(k+1)\Delta F_{k+1} \leq 2(k+k)\Delta F_{k+1} = 4k\Delta F_{k+1} \quad (\text{for} \quad k \geq 1)\,. \tag{3.28}$$

Finalmente, notemos que $\{\vartheta_k\}$ está acotado inferiormente y el lado derecho de (3.28) es sumable (por el Lema 8, item (iii)). ■

En [1, Lema 2], muestran el resultado análogo a este lema, pero en nuestro caso, no es suficiente para demostrar la aceleración de APProx. Para ello también debemos demostrar que:

$$\lim_{k\to\infty}(k+1)^2\Delta F_{k+1} \quad \text{existe.} \tag{3.29}$$

Al suponer que este límite existe, es posible demostrar el resultado principal.

Conjetura 1. Sea $\{x_k\}$ la secuencia generada por aAPPROX y x^* una solución del problema (1.1). Entonces

$$\lim_{k\to\infty} k^2 E\left[F(x_k) - F(x^*)\right] = 0,$$

en otras palabras $E\left[F(x_k) - F(x^*)\right] = o\left(\frac{1}{k^2}\right)$.

Demostración. Desde el Lema 8, item (iii), y la desigualdad $E[F(x_k) - F(x^*)] \leq \Delta F_k$, dada por (3.13), obtenemos que

$$\sum_{k=1}^{\infty}\frac{1}{k}(k+1)^2 E[F(x_{k+1}) - F(x^*)] \leq \infty\,.$$

Combinando esto (3.29)

$$\lim_{k\to\infty} k^2 E[F(x_{k+1}) - F(x^*)] = 0\,.$$

lo que da por finalizada la demostración. ■

Con lo anterior, hemos dado un esquema de demostración que teorizamos, es capaz de probar que con una elección adecuada para los parámetros $\{\theta_k\}$ es posible obtener una variante acelerada del algoritmo APProx, que denominamos aAPProx.

A continuación, daremos paso a mostrar los resultados computacionales obtenidos al comparar APProx con aPProx.

3.4. Resultados computacionales

En esta sección discutiremos los resultados computacionales obtenidos al comparar APProx con aAPProx al considerar:

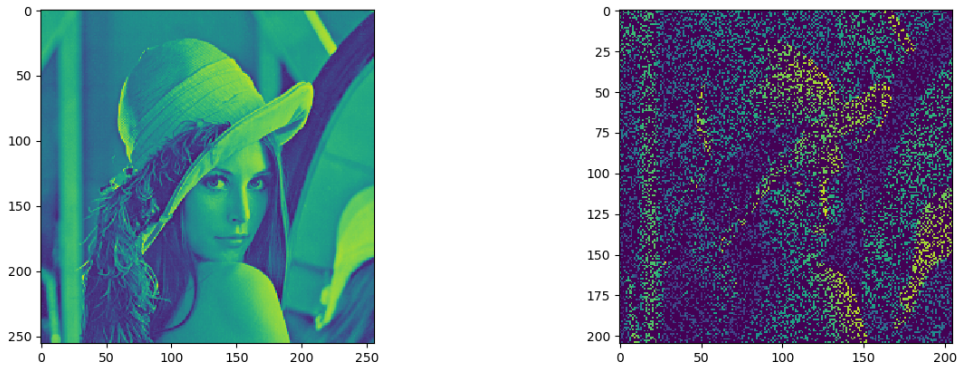
- Todas las variables como un solo bloque.
- Cada variable como un bloque individual.
- Considerando una cantidad arbitraria de variables para los bloques.

Para ello consideramos el problema conocido como *compressed sensing* o *sparse sampling*, el cual consiste en re construir una señal usando solo una muestra aleatoria de ella. En este contexto consideramos la siguiente función objetivo:

$$F(x) = \|Ax - b\|_2^2 + \|x\|_1,$$

donde $A \in \mathbb{R}^{(m,N)}$ es la matriz que lleva la señal al espacio de frecuencias y selecciona una muestra de ella, $b \in \mathbb{R}^m$ es la muestra de la señal original y $x \in \mathbb{R}^N$ corresponde a la señal completa, la cual se desea recuperar (para más detalles ver Capitulo 6, sección 2).

Para las pruebas consideramos la clásica imagen de lena y una muestra del 30 % esta (ver Figura 3.1)



(a) (b)

FIGURA 3.1: (a)Imagen Original, (b)Muestra del 30 % de la imagen.

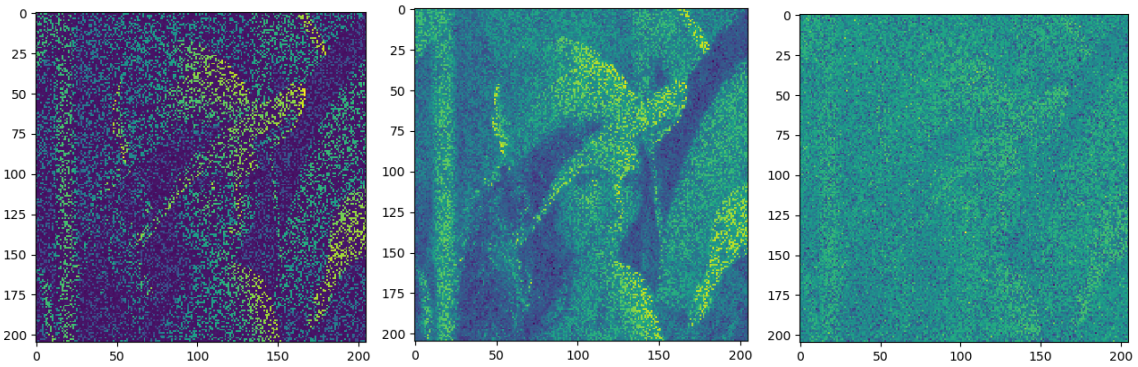
La imagen original posee una resolución de 205×205, lo cual corresponde a un vector de largo 42.025, por lo que una muestra del 30 % corresponde a un vector de largo 12.608, lo que hace que este tipo de problemas escale rápidamente en la cantidad de variables al considerar imágenes de mayor resolución. En nuestro caso, tenemos que $m = 12{,}608$ y $N = 42{,}025$.

Como el desempeño de APProx frente a otros algoritmos (como el gradiente clásico) es comparado en [13, sección 6], nos limitaremos a comparar aAPProx contra APProx.

Para la elección de los parámetro $\{\theta_k\}$ de aAPProx consideramos el criterio mencionado en la seccion 3.3.2, es decir:

- tomar $\theta_0 = \frac{\tau}{n}$.
- actualizar $\theta_{k+1} = \dfrac{\alpha - 1}{k + 1 + (\alpha - 1)\frac{n}{\tau}}$, con $\alpha > 3$.

Este coincide en el caso determinista con el criterio presente en [1]. Esta forma de actualizar $\{\theta_k\}$ es la que consideramos en cada uno de los escenarios, en particular, consideramos $\alpha = 3,1$.

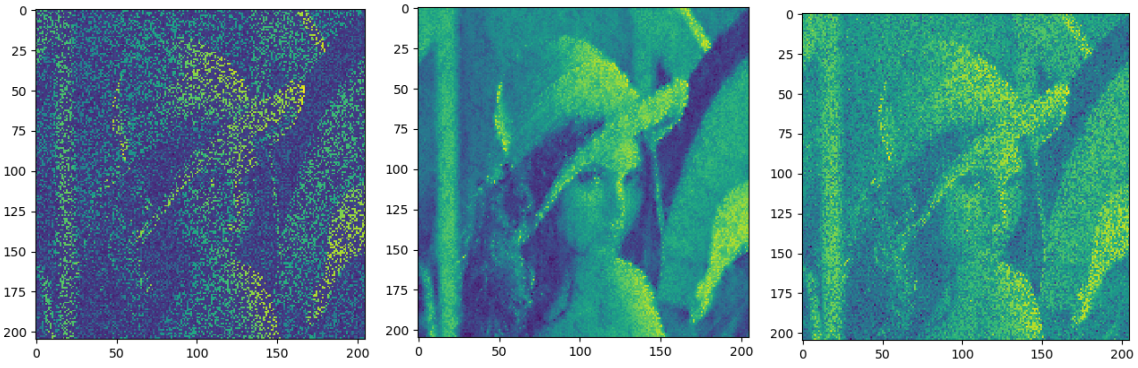


(A) Un bloque (B) Bloques arbitrarios (C) Bloques unitarios

FIGURA 3.2: Comparativa entre los tres esquemas considerados para aAPProx después de (a),(b) 500 y (c) 100 iteraciones.

Para los esquemas en los que tomamos cada coordenada como un bloque y bloques de tamaño arbitrario, se tienen las siguientes consideraciones:

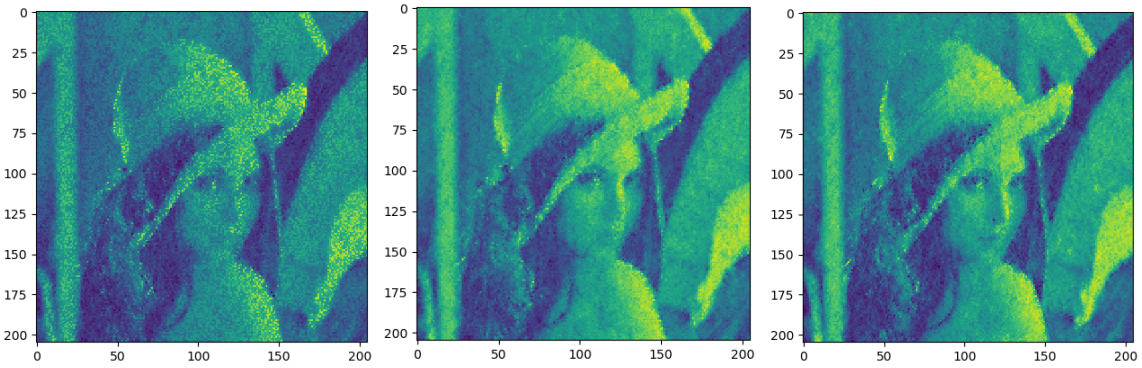
- Cuando cada componente del vector x como un bloque, la cantidad de ellos (τ) que actualizamos por iteración es de 5000, lo que corresponde aproximadamente al 12 % del total. En cada iteración se actualizan los mismos bloques para APProx y aAPProx.
- En el esquema en el que consideramos bloques de tamaño arbitrario consideramos un total de 100, de los cuales 99 poseen 420 variables y el último posee 25. La selección de que variable pertenece a cada bloque se hace de manera aleatoria, con probabilidad uniforme (cada variable tiene la misma probabilidad de pertenecer a cada bloque). En cada iteración actualizamos 50 bloques, por lo que en cada una se actualizan 21000 o 20605, dependiendo si el bloque de menor variables se actualiza o no, lo cual corresponde aproximadamente a la mitad de las variables.



(A) Un bloque (B) Bloques arbitrarios (C) Bloques unitarios

FIGURA 3.3: Comparativa entre los tres esquemas considerados para aAPProx después de (a),(b) 1000 y (c) 200 iteraciones

Al mirar las figuras 3.2, 3.3 y 3.4 observamos que al considerar bloque más finos (de menor cantidad de variables) el algoritmo converge con mayor rapidez a una imagen más nítida y completa. De hecho, al transcurso de 500 iteraciones, la única versión que aún no recupera una imagen completa es la que considera todas las variables dentro de un solo bloque (ver Figura 3.5).



(A) Un bloque (B) Bloques arbitrarios (C) Bloques unitarios

FIGURA 3.4: Comparativa entre los tres esquemas considerados para aAPProx después de (a) 5000 (b) 2000 y (c) 1000 iteraciones.

La explicación de lo anterior puede ser encontrada en las constantes v_i, las cuales juegan un rol importante en la desigualdad (2.8) y actúan de manera similar que las constantes de Lipschitz. Al considerar todas las componentes como un solo bloque, tomamos una constante $v_i = v \in \mathbb{R}, \forall i$, la cual pondera a todas las coordenadas al mismo tiempo; Mientras que en el caso de que cada coordenada es un bloque, las constantes v_i se calculan específicamente para cada bloque, permitiendo que cada actualización sea más precisa (ver Anexo 1).

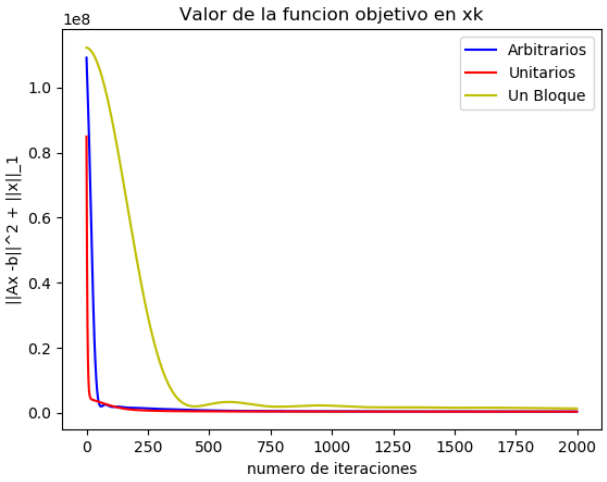
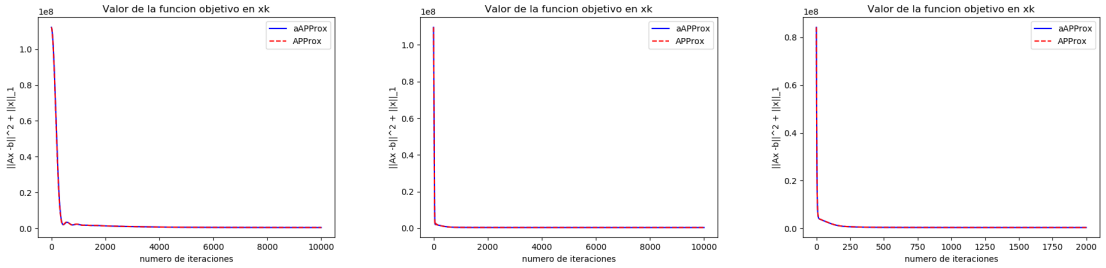


FIGURA 3.5: Comparativa de la velocidad entre los tres esquemas iterativos propuestos

Al comparar el desempeño de aAPProx contra APProx en cada uno de estos tres casos, podemos observar que aAPProx es ligeramente más rápido que APProx, sobre todo en las primeras iteraciones, pues una vez que se alcanzan valores cercanos al optimo, ambos se comportan de manera similar (ver Figura 3.6)



(A) Un bloque (B) Bloques arbitrarios (C) Bloques unitarios

FIGURA 3.6: Comparativa entre los tres esquemas considerados para aAPProx despues de (a) 5000, (b) 2000 y (c) 1000 iteraciones.

En conclusión, hemos mostrado empíricamente que aAPProx es más rápido que APProx, además de observar que la velocidad de ambos depende de la estructura de bloques considerada para resolver el problema.

Capítulo 4

Conclusiones y trabajos futuros

.

En este trabajo hemos propuesto una demostración para probar que el algoritmo APProx, el cual pertenece a la familia de descenso por bloques aleatorio, puede pasar de una convergencia de $O(\frac{1}{k^2})$ a $o(\frac{1}{k^2})$ al modificar sus parámetros de forma adecuada, lo que daría paso a un nuevo algoritmo que hemos denominado aAPProx. Además, hemos mostrado empíricamente, mediante un problema de recuperación de señales a través de una muestra aleatoria, que nuestra propuesta efectivamente posee un mejor desempeño que APProx.

A pesar de que no presentamos una demostración completa, debemos resaltad la novedad de nuestra propuesta, pues busca ir un paso más allá del estado actual de la investigación en cuanto a la tasa de convergencia de los métodos de descenso por bloque, ya sean deterministas o aleatorios, pues busca bajar el orden en la tasa de convergencia, mientras que la mayoría de los trabajos se centran principalmente en obtener cotas más pequeñas para diferentes configuraciones. Por ejemplo en [32] trabajan sobre una versión acelerada que considera restricciones con estructura de bloques diagonal en la matriz; En [19] incorporan información de segundo orden para obtener mejores tasas; Mientras que en [31] trabajan sobre un algoritmo similar a APProx (ver [28]), obteniendo cotas menores y resultados de convergencia en probabilidad para la función objetivo.

Además de lo anterior, en el caso determinista, nuestra propuesta generaría una familia de parámetros más amplia a la presente en [1], para los cuales el algoritmo gradiente-proximal alcanza la tasa de convergencia de al menos $o(\frac{1}{k^2})$, abriendo la puerta a posibles mejores en cuanto a la convergencia y a la complejidad de este, al seleccionar de mejor manera $\mathtt{V}_k$.

Dicho lo anterior, creemos que sentamos las directrices generales para que nuestra hipótesis pueda ser probada en un futuro, además de mostrar resultados empíricos en un escenario similar al *big data*, que sustentan nuestra hipótesis.

Finalmente, como trabajos futuros, se plantean los siguientes puntos:

- Dar una demostración completa para nuestro resultado. Para ello se sugiere estudiar más en profundidad el caso particular en que

$$\theta_{k+1} = \frac{\alpha - 1}{k + 1 + (\alpha - 1)\frac{n}{\tau}}.$$

- Estudiar la convergencia, en algún sentido, de la sucesión $\{x_k\}$ generada por APProx o aAPProx, ya que al día de hoy no hay trabajos (o no dimos con ellos) en donde esté presente algún resultado de ello. Una opción para ello es completar el paralelismo entre [1] y [13].

- Estudiar las consecuencias, en caso de que nuestra hipótesis sea cierta, para el algoritmo gradiente-proximal la existencia de una familia de parámetros más amplia a la propuesta en [1].

Capítulo 5

Apéndice

.

En este capítulo complementaremos el trabajo realizado en la tesis profundizando un poco más en dos puntos: La elección de las constantes v_i, utilizadas en la desigualdad (2.8) (Expected Separable Overapproximation) siendo de vital importancia en aPProx y APProx; además de las nociones básicas del problema de *compressed sensing*, las cuales son necesarias para construir la matriz A del problema de minimización (1.1).

5.1. Más sobre la desigualdad ESO

En esta sección enunciaremos los resultados presentes en [13] que permiten calcular las constantes v_i, presentes en la desigualdad (2.8) y usadas en nuestro algoritmo. El primer teorema nos da la forma exacta de dicha desigualdad y el valor de las constantes v_i para una función f que cumpla con nuestras hipótesis (ver capitulo 2, sección 5.2).

Teorema 4. Supongamos que f satisface la Hipótesis 1.2, entonces:

i) Si $\hat{S}$ es una muestra de bloques de tamaño τ elegida de manera uniforme, entonces para todo $x, h \in \mathbb{R}^N$,

$$E\left[f(x + h_{[\hat{S}]})\right] \leq f(x) + \frac{\tau}{n}\left(\langle \nabla f(x), h\rangle + \frac{1}{2}\|h\|_v^2\right),$$

donde:

$$v_i \overset{\text{def}}{=} \sum_{j=1}^{m} \beta_j L_{ji} = \sum_{j:i\in C_j} \beta_j L_{ji}, \quad i = 1, 2, \ddots, n,$$
$$\beta_j \overset{\text{def}}{=} 1 + \frac{(\omega_j - 1)(\tau - 1)}{max\{1, n-1\}}, \quad j = 1, 2, \ddots, m.$$

Es decir, el par $(f, \hat{S})$ satisfacen ESO con constantes $v = (v_1, \ldots, v_n)$

ii) Más aún, para todo $x, h \in \mathbb{R}^N$ tenemos que:

$$f(x+h) \leq f(x) + < \nabla f(x), h > + \frac{\overline{\omega}\overline{L}}{2}\|h\|_w^2$$

donde:

$$\overline{\omega} \overset{\text{def}}{=} \sum_j \omega_j \frac{\sum_i L_{ji}}{\sum_{k,i} L_{ki}}, \quad \overline{L} \overset{\text{def}}{=} \frac{\sum_{ji} L_{ji}}{n}, \quad w_i \overset{\text{def}}{=} \frac{n}{\sum_{ji} \omega_j L_{ji}} \sum_j \omega_j L_{ji}. \tag{5.1}$$

De esta forma $\overline{\omega}$ es el promedio ponderado de los valores $\{\omega_j\}$ y $\sum w_i = n$

Haciendo uso de esta forma general de calcular las constantes v_i, podemos obtenerlas para una familia particular de funciones, las cuales engloban a $f(x) = \|Ax - b\|^2$.

Teorema 5. Sea $f_j(x) = \phi_j(e_j^T Ax)$, donde $\phi_j : \mathbb{R} \mapsto \mathbb{R}$ es una función con derivada L_{ϕ_j}-Lipschitz. Entonces el gradiente de f_j es Lipschitz por bloques, con constantes

$$L_{ji} = L_{\phi_j}\left(\|A^T L_{ji}\|_{(i)}^*\right)^2, \quad i = 1, 2, \ldots, n.$$

En otras palabras, f_j satisface (2.8) con constante Lji.

Al considerar $f(x) = \frac{1}{2}\|Ax - b\|^2 = \frac{1}{2}\sum_{j=1}^m \left(e_j^T Ax - b\right)^2$, basta tomar:

$$f_j(x) = \psi_j(e_j^T Ax), \quad \text{donde} \quad \psi_j(s) = \frac{1}{2}(s - b_j)^2, \tag{5.2}$$

así los teoremas anteriores nos permiten obtener que [13]:

$$v_i = \sum_{j=1}^{m} \left(1 + \frac{(\omega_j - 1)(\tau - 1)}{max\{1, n-1\}}\right) A_{ji}^2. \tag{5.3}$$

De esta forma cuando:

- Consideramos el vector $x \in R^n$ como un solo bloque, tenemos que $v_i = v \in R$ y

$$v = \sum_{i=1}^{N} \sum_{j=1}^{m} A_{ji}^2.$$

- Consideramos que cada coordenada de x es un bloque, tenemos que $v_i \in \mathbb{R}^N$ y

$$v_i = \sum_{j=1}^{m} A_{ji}^2.$$

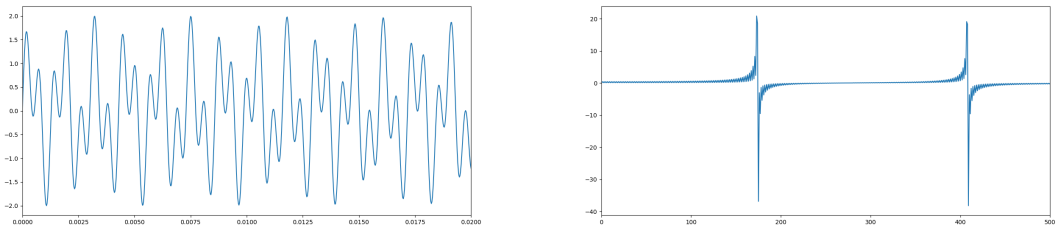
- Al considerar bloques de tamaño no triviales, v_i se calcula según (5.3).

5.2. Sobre Procesamiento de imágenes

En esta sección discutiremos las nociones básicas de nuestro problema relacionada al ámbito del procesamiento de imágenes, en particular, discutiremos sobre la construcción de la matriz A (para más detalles, ver [6]).

Sea $x \in R^N$ un vector que representa una señal en el espacio temporal, es decir, cada componente de x corresponde a la observación de la señal en un instante de tiempo. En el caso de tener una señal en dos dimensiones (como una imagen), basta con considerar la versión vectorizada de ella, obtenida al agrupar todas sus columnas en una sola.

Para trabajar de mejor manera con la señal x, la llevaremos al espacio de frecuencia, pues nos permite trabajar en un espacio en donde el espectro de la señal es cero, salvo en las frecuencias relevantes. Por ejemplo, consideremos x como la suma de dos sinusoidales y su respectiva transformación en el espacio de frecuencias, tal como se representa en la figura 5.1, en la cual se puede observar que en el espacio de frecuencia solo hay dos valores peaks, correspondiente a las frecuencias de dichas sinuosidales.



(a) (b)

FIGURA 5.1: Ejemplo de una señal compuesta por la suma de dos sinousidales, donde (a) es la señal original y (b) es la señal en el espacio de frecuencias.

Dicho lo anterior, ¿Como obtenemos la matriz $A \in \mathbb{R}^{(m,N)}$?. Para ello recordemos que deseamos encontrar un vector $x \in \mathbb{R}^N$ tal que $Ax = b$, donde $b \in \mathbb{R}^m$ corresponde a la muestra de nuestra señal original. Como x lo consideramos en el espacio de frecuencias y b está en el espacio temporal, la matriz A cumple la función de transformar la señal de un espacio a otro, además de obtener las muestras de dicha señal.

Sea $r \in \mathbb{R}^N$ la señal original completa y definamos como D la matriz que genera la muestra b a partir de r, es decir:

$$b = Dr$$

Ahora consideremos Γ a la matriz que lleva una señal del espacio de frecuencia al espacio temporal, de manera que:

$$\Gamma x = r$$

De esta forma, al combinar las dos ecuaciones:

$$Ax = b, \quad \text{donde} \quad A \equiv D\Gamma,$$

por lo que A es simplemente las filas de la matriz Γ que coinciden con la muestra b tomada desde r. Además Γ corresponde a la inversa de la transformación discreta en cosenos aplicada a las columnas de la matriz identidad. De esta forma, basta con aplicar dicha transformación al vector x_k obtenido mediante la rutina de optimización para recuperar la señal estimada en el espacio temporal.

Bibliografía

[1] H. Attouch and J. Peypouquet. The rate of convergence of nesterov's accelerated forward-backward method is actually faster than $1/k^2$. *SIAM Journal on Optimization*, 26(3):1824–1834, 2016. doi: 10.1137/15M1046095. URL http://dx.doi.org/10.1137/15M1046095.

[2] A. Beck and M. Teboulle. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM journal on imaging sciences*, 2(1):183–202, 2009.

[3] R. E. Bruck Jr. An iterative solution of a variational inequality for certain monotone operators in hilbert space. *Bulletin of the American Mathematical Society*, 81(5):890–892, 1975.

[4] G. C. Calafiore. Parallel block coordinate minimization with application to group regularized regression. *Optimization and Engineering*, 17(4):941–964, Dec 2016. ISSN 1573-2924. doi: 10.1007/s11081-016-9336-z. URL https://doi.org/10.1007/s11081-016-9336-z.

[5] E. J. Candès and B. Recht. Exact matrix completion via convex optimization. *Found. Comput. Math.*, 9(6):717–772, Dec. 2009. ISSN 1615-3375. doi: 10.1007/s10208-009-9045-5. URL http://dx.doi.org/10.1007/s10208-009-9045-5.

[6] E. J. Candès and M. B. Wakin. An introduction to compressive sampling [a sensing/sampling paradigm that goes against the common knowledge in data acquisition]. *IEEE signal processing magazine*, 25(2):21–30, 2008.

[7] A. Chambolle and T. Pock. A remark on accelerated block coordinate descent for computing the proximity operators of a sum of convex functions. working paper or preprint, Jan. 2015. URL https://hal.archives-ouvertes.fr/hal-01099182.

[8] P. L. Combettes and V. R. Wajs. Signal recovery by proximal forward-backward splitting. *Multiscale Modeling & Simulation*, 4(4):1168–1200, 2005.

[9] I. Daubechies, M. Defrise, and C. De Mol. An iterative thresholding algorithm for linear inverse problems with a sparsity constraint. *Communications on Pure and Applied*

Mathematics: A Journal Issued by the Courant Institute of Mathematical Sciences, 57 (11):1413–1457, 2004.

[10] D. Drusvyatskiy. The proximal point method revisited. *SIAG/OPT Views and News*, 26(1), 2018.

[11] R. Durrett. *Probability: theory and examples*, volume 49. Cambridge university press, 2019.

[12] T. Eitrich and B. Lang. Efficient optimization of support vector machine learning parameters for unbalanced datasets. *Journal of Computational and Applied Mathematics*, 196(2):425 – 436, 2006. ISSN 0377-0427. doi: https://doi.org/10.1016/j.cam.2005.09.009. URL `http://www.sciencedirect.com/science/article/pii/S0377042705005856`.

[13] O. Fercoq and P. Richtárik. Accelerated, parallel, and proximal coordinate descent. *SIAM Journal on Optimization*, 25(4):1997–2023, 2015. doi: 10.1137/130949993. URL `http://dx.doi.org/10.1137/130949993`.

[14] D. H. Gutman and J. F. Pena. Convergence rates of proximal gradient methods via the convex conjugate. *SIAM Journal on Optimization*, 29(1):162–174, 2019.

[15] J. Hainmueller and C. Hazlett. Kernel regularized least squares: Reducing misspecification bias with a flexible and interpretable machine learning approach. *Political Analysis*, 22(02):143–168, 2014. URL `https://EconPapers.repec.org/RePEc:cup:polals:v:22:y:2014:i:02:p:143-168_01`.

[16] J. Konečnỳ, Z. Qu, and P. Richtárik. Semi-stochastic coordinate descent. *Optimization Methods and Software*, 32(5):993–1005, 2017.

[17] X. Li, T. Zhao, R. Arora, H. Liu, and M. Hong. On faster convergence of cyclic block coordinate descent-type methods for strongly convex minimization. *The Journal of Machine Learning Research*, 18(1):6741–6764, 2017.

[18] J. Liu, S. J. Wright, C. Ré, V. Bittorf, and S. Sridhar. An asynchronous parallel stochastic coordinate descent algorithm. *J. Mach. Learn. Res.*, 16(1):285–322, Jan. 2015. ISSN 1532-4435. URL `http://dl.acm.org/citation.cfm?id=2789272.2789282`.

[19] S. Lu, M. Hong, and Z. Wang. On the sublinear convergence of randomly perturbed alternating gradient descent to second order stationary solutions. *Proc. International Conference on Machine Learning (ICML)*, 2018.

[20] I. Necoara, Y. Nesterov, and F. Glineur. Random block coordinate descent methods for linearly constrained optimization over networks. *Journal of Optimization Theory and Applications*, 173(1):227–254, Apr 2017. ISSN 1573-2878. doi: 10.1007/s10957-016-1058-z. URL `https://doi.org/10.1007/s10957-016-1058-z`.

[21] Y. Nesterov. A method of solving a convex programming problem with convergence rate o (1/k2). In *Soviet Mathematics Doklady*, volume 27, pages 372–376, 1983.

[22] Y. Nesterov. Gradient methods for minimizing composite objective function. CORE Discussion Papers 2007076, Université catholique de Louvain, Center for Operations Research and Econometrics (CORE), 2007. URL `https://EconPapers.repec.org/RePEc:cor:louvco:2007076`.

[23] Y. Nesterov. Efficiency of coordinate descent methods on huge-scale optimization problems. *SIAM Journal on Optimization*, 22(2):341–362, 2012. URL `http://dblp.uni-trier.de/db/journals/siamjo/siamjo22.html#Nesterov12`.

[24] Y. Nesterov. *Introductory lectures on convex optimization: A basic course*, volume 87. Springer Science & Business Media, 2013.

[25] Y. Nesterov et al. Gradient methods for minimizing composite objective function, 2007.

[26] J. Peypouquet. *Convex optimization in normed spaces*. SpringerBriefs in Optimization. Springer, Cham, 2015. Theory, methods and examples, With a foreword by Hedy Attouch.

[27] P. Richtárik and M. Takáč. Parallel coordinate descent methods for big data optimization. *Mathematical Programming*, 156(1):433–484, Mar 2016. ISSN 1436-4646. doi: 10.1007/s10107-015-0901-6. URL `https://doi.org/10.1007/s10107-015-0901-6`.

[28] P. Richtárik and M. Takáč. Parallel coordinate descent methods for big data optimization. *Mathematical Programming*, 156(1-2):433–484, 2016.

[29] S. Ruder. An overview of gradient descent optimization algorithms. *CoRR*, abs/1609.04747, 2016. URL `http://arxiv.org/abs/1609.04747`.

[30] Z. Shi and R. Liu. Better worst-case complexity analysis of the block coordinate descent method for large scale machine learning. *16th IEEE International Conference on Machine Learning and Applications, Cancun, Mexico, December 18-21*, pages 889–892, 2017.

[31] R. Tappenden, M. Takáč, and P. Richtárik. On the complexity of parallel coordinate descent. *Optimization Methods and Software*, 33(2):372–395, 2018.

[32] Y. Tian, K. Khosoussi, and J. P. How. Block-coordinate minimization for large sdps with block-diagonal constraints, 2019.

[33] P. Tseng. On accelerated proximal gradient methods for convex-concave optimization. *SIAM Journal on Optimization*, 2008.

[34] S. J. Wright. Coordinate descent algorithms. *Mathematical Programming*, 151(1):3–34, 2015.

[35] Z. Zhu and A. J. Storkey. Stochastic parallel block coordinate descent for large-scale saddle point problems. In *AAAI*, pages 2429–2437, 2016.