ITAM
Roberto Acevedo 89750
MNO-2018
Reporte "Convex Optimization for Big Data"

Reporte de Lectura: "Convex Optimization for Big Data by Volkan Cevher, Stephen Becker and Mark Schmidt"

1. Objetivo

El objetivo del artículo es introducir al lector a algunas técnicas de optimización modernas que son útiles para la solución de problemas que incorporan grandes volúmenes de datos, haciendo particular énfasis en aplicaciones de procesamiento de señales.

2. Generalidades

En general, el artículo revisa avances recientes en optimización convexa para Big Data, los cuales buscan reducir costos computacionales y de almacenamiento. Se revisarán métodos de primer órden, aleatorización y la importancia del cómputo en paralelo.

3. Introducción

A grandes rasgos, los autores hablan de que el campo de convexidad en el procesamiento de señales lleva ya mucho tiempo, pero que a partir del surgimiento de las nuevas teorías de "Structured Sparcity", de "Rank Minimization" y de máquinas de soporte vectorial, la importancia de dicho campo se ha incrementado en la última década. Sin embargo, dicha importancia pone una gran presión en los algoritmos para poder procesar grandes volúmenes de datos. Es por esto que la optimización convexa se está reinventando en términos de simplicidad.

Posteriormente, se plantean los fundamentos de optimización de Big Data a partir de la siguiente fórmula en donde f y g son funciones convexas:

$$F^* \stackrel{\text{def}}{=} \min_x \left\{ F(x) \stackrel{\text{def}}{=} f(x) + g(x) : x \in \mathbb{R}^p \right\}$$

Para luego hablar de los 3 pilares en donde descansan los algoritmos de optimización en Big Data:

- **Métodos de Primer Órden**. Los cuáles obtienen una precisión numérica baja o media y que son capaces de manejar variantes no suaves utilizando el principio de "proximal mapping". También son teóricamente robustos y generalmentedependen de cómputo en paralelo o distribuido.

- **Aleatorización**. Mejoran la escalabilidad de métodos de primer órden gracias a que se puede controlar su comportamiento.Algunas ideas pueden ser actualizaciones parciales aleatorias, reemplazo de gradiente determinístico o aceleración de rutinas de algebra lineal.

- **Cómputo en Paralelo o Distribuido**. De manera natural, los métodos de primer órden proveen un marco flexible para distribuir de manera óptima instrucciones y realizar cómputo en paralelo

4. Métodos de Primer Órden para Optimización Convexa

Los autores abordan dos clases de métodos de primer órden. Por una parte los de objetivos suaves, y por otro los de objetivos compuestos.

Como solución a los métodos de primer órden para objetivos suaves, se propone el algoritmo de Nesterov para el método de gradiente acelerado como método para la solución de minimización sin restricciones. Dicho algoritmo alcanza la mejor posible peor razón del error y típicamente es conocido como un método de primer órden óptimo. El algoritmo es el siguiente:

Algorithm 1 Nesterov's accelerated gradient method for unconstrained minimization ($v^0 = x^0$) [11]

1: $x^{k+1} = v^k - \alpha_k \nabla f(v^k)$
2: $v^{k+1} = x^{k+1} + \beta_k (x^{k+1} - x^k)$

Por otra parte, también se propone el algoritmo de aceleración del gradiente próximo como una solución a los métodos de primer órden para objetivos compuestos, es decir, los que consideran el problema canónico compuesto de la fórmula inicial y donde el objetivo F consiste de funciones "f" diferenciables y una función no suavizada convexa"g". El algoritmo es el siguiente:

Algorithm 2 Accelerated proximal gradient method to solve (1) [11, 15]. Set $v^0 = x^0$.

1: $x^{k+1} = \text{prox}_{\alpha_k g}\left(v^k - \alpha_k \nabla f(v^k)\right)$
2: $v^{k+1} = x^{k+1} + \beta_k (x^{k+1} - x^k)$

Finalmente, dentro de los métodos de primer órden para objetivos de proximidad, se propone el algoritmo del método de alternancia de direcciones de multiplicadores (ADMM por sus siglas en ingles). Este método brinda poderosas técnicas de de lagrangeanas y descomposición dual. El algoritmo es el siguiente:

Algorithm 3 ADMM to solve (12); $\gamma > 0, z^0 = u^0 = 0$

1: $x^{k+1} = \text{argmin}_x \ \gamma h(x) + \frac{1}{2}\|x - \Phi z^k + u^k\|_2^2 = \text{prox}_{\gamma h}(\Phi z^k - u^k)$
2: $z^{k+1} = \text{argmin}_z \ \gamma g(z) + \frac{1}{2}\|x^{k+1} - \Phi z + u^k\|_2^2$
3: $u^{k+1} = u^k + x^{k+1} - \Phi z^{k+1}$

5. Escalamiento de Big Data a través de Aleatorización

En teoría, los métodos de primer órden sirven para atender problemas de gran escala, sin embargo, en la práctica la exactitud en el cómputo puede hacer que dicho métodos no sean factibles en la medida en que las dimensiones crecen. Para esos casos, las aproximaciones aleatorias pueden servir para incrementar los alcances de dichos métodos.

Como calcular el gradiente completo para PageRank requiere operaciones matriz-vector en cada iteración, los autores proponen seleccionar una coordenada i de x y sólo modificar la variable correspondiente xi para mejorar la función objetivo. Esta idea representa un método de descenso coordinado, y su forma general esta representada por el siguiente algoritmo:

Algorithm 5 Coordinate descent to minimize F over $\mathbb{R}^p$

1: Choose an index $i_k \in \{1, 2, \ldots, p\}$ *(see the main text for possible selection schemes)*
2: $x^{k+1} = x^k - \alpha \nabla_{i_k} F(x^k) e_{i_k}$

En contraste con métodos aleatorizados de descenso gradiente que actualizan una sola coordenada a la vez con su gradiente exacto, métodos de gradiente estocástico actualizan todas las coordenadas simultáneamente, pero utilizan gradientes aproximados. A continuación, se puede observar el algoritmo de descenso gradiente estocástico para minimizar F sobre Rp.

Algorithm 6 Stochastic gradient descent to minimize F over $\mathbb{R}^p$

1: Choose an index $j_k \in \{1, 2, \ldots, n\}$ *uniformly at random*
2: $x^{k+1} = x^k - \alpha_k \nabla F_{j_k}(x^k)$

6. Cómputo en Paralelo y Distribuido

En teoría, los métodos de primer órden parecen ser ideales de implementarse con fines de acelerar el procesamiento, sin embargo, pueden surgir 2 problemas al momento de utilizar hardware heterogéneo al momento de distribuir instrucciones:

- Cominicación. Puede ser con fallas o de manera dispareja entre computadoras y entre jerarquía de memoria local, puede reducir significantemente la eficiencia numérica en general de los métodos de primer orden. Para solucionar esto se puede diseñar algoritmos que minimicen la comunicación, así como eliminar el vector xk y en lugar trabajar con una copia local en cada máquina que nos lleve a un consenso x* en convergencia.

- Sincronización. Para realizar exactamente el cómputo en una manera distribuida, los métodos de primer orden deben coordinar las actividades de diferentes computadoras las cuales los números primitivos dependen en el mismo vector xk en cada iteración.

En general, los métodos de primer orden pueden verse beneficiados significativamente del cómputo en paralelo. Para esto es necesario descomponerlos.

Un ejemplo puede ser la descomposición de ADMM de la siguiente manera:

Algorithm 8 Decomposition algorithm (aka, consensus ADMM) [30] to solve (18); $\gamma > 0$, $x_i^0 = 0$ for $i = 1, \ldots, n$.

1: $z^{k+1} = \frac{1}{n} \sum_{i=1}^{n} \text{prox}_{\gamma F_i}(x_{(i)}^k)$
2: **for** $i = 1$ **to** n **do**
3: $\quad x_{(i)}^{k+1} = 2z^{k+1} - z^k + x_{(i)}^k - \text{prox}_{\gamma F_i}(x_{(i)}^k)$
4: **end for**