El trabajo presentado por Cevher, Becker y Schmidt revisa los últimos avances en algoritmos de optimización convexa para Big Data. Al igual que el resumen anterior, la idea principal es que si bien ya hay métodos teóricamente sólidos la parte computacional está lejos de ser óptima. En ese sentido, con este análisis se revisa los algoritmos que disminuyen los problemas en almacenamiento, cómputo y comunicaciones (que los autores llaman cuellos de botella). Además, un punto muy importante, se discute el tema cómputo en paralelo que como se vio a lo largo del curso resulta una alternativa que resuelve problemas en la parte práctica.

Lo básico

Los autores señalan que la optimización de Big Data se da a través de la siguiente formulación compuesta:

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

La formulación presenta las funciones f y g que son convexas. En ese sentido los autores señalan que los pilares fundamentales de los algoritmos de optimización descansan sobre:

1. Métodos de primer orden
2. Aleatorización
3. Cómputo en paralelo y distribuido

Métodos de primer orden

En esta sección los autores describen los métodos de primer orden enfatizando algoritmos específicos que garantizan la convergencia global. Uno de los métodos más populares en muchas disciplinas es la regresión lineal de la forma:

$$y = \Phi x_0 + z$$

Donde x es un parámetro desconocido, Φ es una matriz conocida y z una perturbación desconocida. Generalmente, la información en este tipo de modelos surge de la experimentación u observación directa del fenómeno de interés. Sin embargo, se requiere de una formulación compuesta para tener mejores resultados. Recordemos que la idea es tener mejores herramientas con las cuales analizar conjuntos de datos enormes. En ese sentido, se propone utilizar métodos distintos dependiendo su la optimización convexa es suave o no.

Objetivos suaves

Cuando tenemos el objetivo F que sólo consiste de una función diferenciable f, podemos utilizar un método de gradiente.

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

Para tal problema tenemos las alternativas

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

Objetivos compuestos

Cuando tenemos el objetivo F que sólo consiste de una función diferenciable f, y una función convexa g no-suave. En este caso resulta conveniente utilizar el descenso de gradiente próximo.

$$x^{k+1} = \underset{y \in \mathbb{R}^p}{\operatorname{argmin}} \left\{ f(x^k) + \nabla f(x^k)^T (y - x^k) + \frac{1}{2\alpha_k} \left\| y - x^k \right\|^2 \right\},$$

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

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

Objetivos próximos

Cuando los métodos antes descritos no se pueden aplicar y tenemos el siguiente problema:

$$\min_{x,z \in \mathbb{R}^p} \left\{ F(x,z) \overset{\text{def}}{=} h(x) + g(z) : \Phi z = x \right\},$$

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: $u^{k+1} = u^k + x^{k+1} - \Phi z^{k+1}$

Algorithm 4 Primal-Dual Hybrid Gradient algorithm to solve (12); $\gamma > 0$ and $\tau \leq 1/\|\Phi\|^2$

1: $x^{k+1} = \text{prox}_{\gamma h}(\Phi z^k - u^k)$

2: $z^{k+1} = \text{prox}_{\gamma \tau g}(z^k + \tau \Phi^T(x^{k+1} - \Phi z^k + u^k))$

3: $u^{k+1} = u^k + x^{k+1} - \Phi z^{k+1}$

Con estos algoritmos podemos obtener soluciones numéricas a los problemas que estudiamos independientemente de la función objetivo del problema. Estos algoritmos sin duda presentan soluciones que requieren menos tiempo que otras que existen. Por otra parte, el replanteamiento de un problema de minimización permite mejorar las capacidades de respuesta al mismo. La modificación de la forma funcional permite utilizar el algoritmo de Alterning Direction Method of Multipliers que además permite paralelización. Lo anterior, permite que podamos utilizar métodos de primer orden en nuestro día a día.

Aleatorización

En primera instancia nos bastaría con los métodos presentados anteriormente, sin embargo el número de operaciones es descomunal debido al alto número de iteraciones que tienen. En ese sentido, se requieren técnicas distintas que nos ayuden a atacar el problema. Existen algoritmos como el de PageRank que nos permite atacar este problema. La idea básica detrás de estos algoritmos es la popularidad de un sitio de internet. Este algoritmo fue creado por los fundadores de Google. La idea detrás es que el valor de una página está en función de la estructura de enlaces que tiene. Un enlace de la página A a B es como un “voto”. Además, se toma en cuenta la página que emite el “voto”. Los votos emitidos por las páginas consideradas "importantes", es decir con un PageRank elevado, valen más, y ayudan a hacer a otras páginas "importantes". Por lo tanto, el PageRank de una página refleja la importancia de la misma en Internet. En esta sección los autores proponen los siguientes algoritmos:

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: $x^{k+1} = x^k - \alpha \nabla_{i_k} F(x^k) e_{i_k}$

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: $x^{k+1} = x^k - \alpha_k \nabla F_{j_k}(x^k)$

Algorithm 7 Randomized low-rank approximation

Require: $M \in \mathbb{R}^{p \times p}$, integer r

1: Draw $\Omega \in \mathbb{R}^{p \times r}$ iid $\mathcal{N}(0,1)$
2: $W = M\Omega$ // Matrix multiply, cost is $\mathcal{O}(p^2 r)$
3: $QR = W$ // QR algorithm, e.g., Gram-Schmidt, cost is $\mathcal{O}(pr^2)$
4: $U = M^T Q$ // Matrix multiply, cost is $\mathcal{O}(p^2 r)$
5: **return** $\widehat{M}_{(r)} = QU^T$ // Rank r

Las técnias de aleatorización se utilizan para mejorar la escalabilidad de los métodos de primer orden. Detrás de esto podemos señalar el uso de estimadores estadísticos para la sustitución del gradiente determinístico, que resulta en ahorros a nivel computacional además de incrementar la velocidad de cálculo.

Cómputo en paralelo y distribuido

Como conocemos las capacidades computaciones se han incrementado de manera exponencial en los últimos años. Un caso de ejemplo es el procesamiento con tarjetas gráficas que ha aumentado el potencial computacional. Aunado a lo anterior la posibilidad de paralelizar y distribuir las tareas en varios nodos ayuda a incrementar la velocidad de cómputo. Sin embargo la infraestructura física compuesta en su mayoría de silicón, que tiene ya signos de estancamiento, aunado a la gran cantidad de energía requerida para su funcionamiento, han puesto en entredicho estas ventajas. Sin embargo, el uso del cómputo en paralelo también permite el uso de algoritmos y técnicas distintas a las tradicionales. En ese sentido, los autores proponen el siguiente algoritmo:

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**

El uso de algoritmos de cómputo en paralelo y distribuido como el aquí analizado permite dar mejor respuesta a nuestro problema mediante el uso de varios núcleos para el procesamiento, contribuyendo de esta manera al incremento de la velocidad de cómputo sin sacrificio de la precisión de los resultados. Esto incrementa la eficiencia de nuestros procesos.

Conclusiones

Los problemas analizados en este paper, presentan soluciones poco convencionales, pero que son viables desde el punto de vista computacional (en las condiciones actuales es decir con las limitantes que hoy se tienen). En ese sentido los autores proponen que el uso de formas compuestas se incrementará para poder obtener más de los mismos datos. Por ejemplo, señala el modelo Lasso que vimos en nuestros cursos de aprendizaje de máquina. El otro aspecto interesante es el tiempo de procesamiento que resulta fundamental en la actualidad.

Referencias

Link del paper:

https://arxiv.org/pdf/1411.0972.pdf