Instituto Tecnológico Autónomo de México
Maestría en Ciencia de Datos
Mayo 2018

Reporte

Convex Optimization for Big Data

Por

Diego Alejandro Estrada Rivera

165352

1 Introducción

La importancia de la optimización ha ido en incremento a lo largo de la última década debido al surgimiento de nueva teoría para la dispersión estructurada, minimización de rango y algoritmos de aprendizaje estadístico como support vector machines. Sin embargo, esto ha puesto presión sobre la optimización convexa para adaptarse a las grandes cantidades de datos (Big Data) que han empezado a surgir conforme avanzan la tecnología y los tiempos, aún a pesar de los avances que se han logrado en el computo en paralelo y la distribución del procesamiento.

Es así que la optimización conveza se ha empezado a reinventar para porder manejar Big Data, donde los tamaños de los datos y los parámetros de los problemas de optimización son demasiados grandes para ser procesados localmente, y donde incluso rutinas de álgebra lineal básicas, tienes que re implementarse.

Se describen los fundamentos de la optimización en Big Data mediante la siguiente fórumla:

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

donde f y g son funciones convexas. El entendimiento básico de algoritmos para optimización en Big Data se basa en tres pilares básicos:

- Métodos de primer órden
- Aleatorización
- Cómputo en paralelo y distribuido

Estos tres conceptos ofrecen grandes beneficios de escalabilidad para optimización en Big Data.

2 Métodos de primer órden para optimización convexa suave y no suave

Iniciamos analizando los objetivos suaves pensando en una función F que solo consiste de una función convexa diferenciable f. La técnica de primer orden por excelencia para este caso seríe el método de gradiente, que usa el gradiente local de la función para actualizar de la siguiente manera:

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

donde k es el conteo de la iteración y α_k es el tamaño de la misma. Una suposición común sustenta que en muchas aplicaciones el gradiente de la función f es Lipschitz continua

$$\forall x, y \in R^k, || \bigtriangledown f(x) - \bigtriangledown f(y)||_2 \leq L||x - y||_2$$

para una constante L. Cuando la función se puede diferenciar dos veces, una condición suficiente es que el eigenvalor de su Hessiana están delimitadas arriba por L. Por lo mismo, podemos estimar $L = ||\Phi||_2^2$ y establecer así el tamaño del paso a 1/L. Desafortunadamente, esta tasa de convergencia no se atiene al limite inferior de complejidad que se mantiene para todas las funciones Lipschitz gradiente continuas. Sorprendentemente, una modificación menor por Nesterov logra convergencia optima por la simple selección de paso 1/L y un paso extra con un parametro $\beta_k = \frac{k}{k+3}$. De esta manera obtenemos el algoritmo:

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

Este metodos de gradiente acelerado logra el mejor "peor" posible caso en cuanto a error, y por lo tanto, es tipicamente referido como un optimo caso de primer orden. Muchas funciones tambien presentan estructuras adicionales utiles para optimización numérica. Entre ellas, la fuerte convexidad merece atención especial ya que esta estructura ofrece beneficios clave como la existencia de un miimizador único y mejorada eficiencia en optimización. Así podemos transformar cualquier problema convexo en un problema fuertemente convexo tan solo añadiendo un término cuadrado de regularización. Lo algoritmos de gradiente rápido también se aplican a problemas de minimización no suave usando la técnica de suavizamiento de Nesterov.

Consideramos ahora problemas compuestos, donde el objetivo F consiste de una funcion convexa diferenciable f y una no convexa g. Afortunadamente objetivos compuestosestan lejos de problemas de optimizacion convexos no suaves. Los metodos de gradiente cercano especificamente toman ventaja de la estructura compuesta para mantener las mismas tasas de convergencia de los metodos de gradiente utilizados para problemas suaves.

3 Big Data mediante aleatorización

En teoría, los métodos de primer orden estan bien posicionados para atacar problemas de gran escala. En la práctica, sin embargo, los calculos necesarios para sus iteraciones pueden hacer que incluso metodos sencillos se vuelvan impracticos conforme las dimensiones crecen. Afortunadamente, resulta que los metodos de primer orden son lo suficientemente robustos como para utilizar aproximaciones de sus optimizaciones primiticas, tales como gradiente y calculos proximos. Un ejemplo sencillo para entender estas ideas es el algoritmo de PageRank de Google, que mide la importancia de los nodos en un grafo dado por medio de la matriz de incidencia $M \in R^{pxp}$, considerando que p esta en el orden de los miles de millones. Asumiendo que nodos más importantes tienen más conexiones, el problema apunta a encontrar los mejores vectores singulares de la matriz estocástica $\Phi = Mdiag(M^T 1_p)^{-1}$, donde $1_p \in R^p$ es un vector de solo 1's.
El algoritmo de PageRank simplemente resuelve este basico problema de algebra lineal con el power method. Sin embargo, podemos aproximarnos a esta meta usando un problema de minimos cuadrados cuando relajamos las limitantes con un parametro de penalización $\gamma > 0$:

$$\min_{x \in R^p} \{F(x) \stackrel{\text{def}}{=} \frac{1}{2}||x - \Phi x||_2^2 + \frac{\gamma}{2}(1_p^T x - 1)^2\},$$

Debemos notar que podemos trabajar con una versión restringida de este problema que incluye positividad, but como la formulacionón de PageRank por si misma no es un modelo exacto de la realidad, el problema más sencillo puede ser preferible por obvias razones computacionales. Claramente, nos gustaría minimizar el numero de operaciones involucradas en la matriz Φ en cualquier método solución.

4 Computo paralelo y distribuido

Gracias a las leyes de Moore para escalar desidad silicon, la capacidad de computo y almacenamiento han incrementado a un paso exponencial desde mediados de los años 2000, brindándole a los algoritmos de optimización convexa una gran mejora en rendimiento. Sin embargo, mientras que se espera que las leyes de Moore sigan vigentes por años, las eficiencia de transistores se ha estancado. Como dicta la ley de Dennard, escalad densidad silicon ahora resulta en niveles de consumo de energía sin precedentes. Para manejar la masiva capacidad de computo y almacenamiento requeridas por Big Data con un nivel razonable de consumo de energía, debemos confiar cada vez más en el computo en paralelo y distribuido.

Mientras los métodos de primer orden parecen ser ideales para mejoras de rendimiento, surgen dos problemas cuando se utiliza hardware distribuido y diferente, sincronización y comunicación.

5 Conclusiones

Manjejar problemas de Big Data no solo requiere un conocimiento sólido y bastante experiencia manjeando algoritmos de optimización convexa, sino que también requiere de la capacidad de idear ciertos "trucos" cuando se trata de computación. Se deben sacrificar ciertos aspectos en los algoritmos para lograr ganar las capacidades necesarias para resolver los problemas que se presentan.

El problema de sincronización y comunicación ha limitado en muchas formas a los algortimos ya existentes, pero es también por esto que surgen nuevas, formas de atacar los problemas de Big Data, adaptadas ya a esta forma de pensar de computo paralelo y distribuido. Es poco a poco que el cambio en la metodología con que se resuelven los problemas ira cambiando y adaptandose, dando lugar a capacidades de computo que hace varios años no podríamos haber concebido, sin embargo, es tan bien de esperar que surjan nuevos retos y limitaciones conforme avanza la tecnología.