Convex Optimization for Big Data

Optimización convexa a raíz de Big Data

Convexidad en procesamiento de señales, tiene mucho tiempo ya de ser un campo de estudio, pero esto ha incrementado en la última década debido al surgimiento de “structured sparsity” y “Rank minization” así como SVM. Las dos razones principales de este crecimiento son: la existencia de algoritmos eficientes para realizar cómputo global y la habilidad de usar geometría convexa para probar propiedades de la solución

La optimización convexa se está reinventando para Big Data, donde el tamaño de los datos y los parámetros son demasiado grandes para procesarse localmente; y donde rutinas básicas de algebra lineal no se pueden realizar

Conceptos básicos

Los fundamentos de la optimización de Big Data, se pueden describir con la siguiente fórmula, donde ‘f’ y ‘g’ son funciones convexas.



Los tres pilares para poder comprender los algoritmos de optimización de Big Data son los siguientes:

1. Métodos de primer orden: Los métodos de primer orden obtienen una precisión numérica baja o media, utilizando información de primer orden del objetivo. También son capaces de manejar variantes “no-suaves” utilizando el principio de mapeo próximo. Estos métodos obtienen ritmos de convergencia casi dimensionalmente independientes. Teóricamente son robustos para realizar optimizaciones y típicamente se realizan en cómputo distribuido y en paralelo.
2. Aleatorización: Las técnicas de aleatorización sobresalen particularmente sobre otras técnicas de aproximación para mejorar la escalabilidad de métodos de primer orden debido a que se puede controlar su comportamiento esperado. Las ideas principales en la aleatorización son: actualizaciones parciales aleatorias de variables de optimización, reemplazo de gradiente determinístico y cálculos próximos con estimadores estadísticos baratos.
3. Cómputo distribuido y en paralelo: Los métodos de primer orden naturalmente proveen un marco de referencia para distribuir tareas de optimización y realizar cómputo en paralelo. Estos métodos pueden ser complementados incrementando niveles de escalabilidad, desde algoritmos paralelos síncronos idealizados con comunicación centralizada a algoritmos asíncronos enormemente escalables con comunicaciones descentralizadas.

Métodos de primer orden para optimización convexa savizada y no suavizada

Objetivos suavizados:

Este caso se enfoca para los casos donde el objetivo F solo consiste de una función diferenciable convexa ‘f’. El método gradiente que solo utiliza ∇f(x) e iterativamente realiza la siguiente actualización:



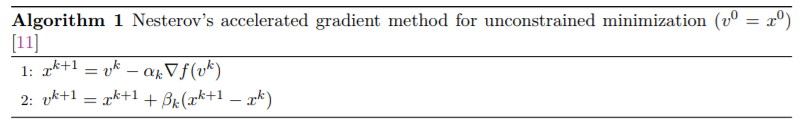
Donde ‘k’ es la cuenta de la iteración y αk es el tamaño de paso apropiado que asegura convergencia.

Al realizar suposiciones simples acerca de ‘f’, podemos analizar cuantas iteraciones tiene que realizar el método gradiente para alcanzar una solución precisa. Normalmente asumimos que el gradiente de ‘f’ es continuo Lipschitz:



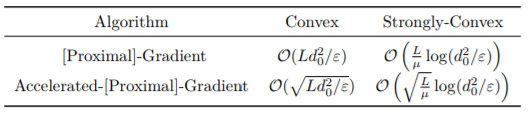
Para una constante L.

A continuación, podremos observar el algoritmo de Nesterov para el método del gradiente acelerado para minimización sin restricciones:



Este algoritmo alcanza la mejor posible peor razón de error, por lo tanto, es típicamente referido como un método de primer orden óptimo.

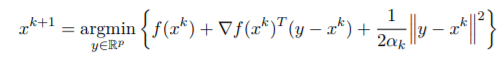
El método de gradiente acelerado también puede ser beneficiado de convexidad fuerte con la apropiada selección del término de momento βk. En contraste el método del gradiente explota automáticamente convexidad fuerte sin ningún conocimiento de µ. En la siguiente tabla se puede observar el número de iteraciones para alcanzar la precisión de ϵ para las diferentes configuraciones discutidas:



Finalmente, los algoritmos de gradiente rápido descritos, también aplican a minimización ‘no suave’ utilizando la técnica de suavizamiento de Nesterov

Objetivos compuestos

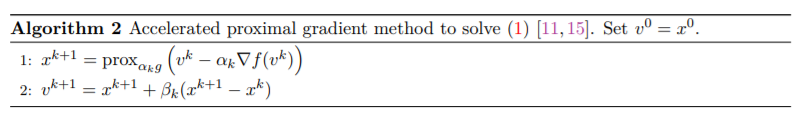
Aquí se considera el problema canónico compuesto, donde el objetivo F consiste de funciones ‘f’ diferenciables y una función no suavizada convexa ‘g’. Los métodos de gradientes proximales aprovechan la estructura compuesta para retener las mismas razones del método gradiente para las clases de problema suave. Problema de optimización:



Que está basado en una aproximación cuadrática local de ‘f’. Métodos de gradiente proximal utilizan la misma aproximación de f, pero incluyen el término de no-suavizamiento ‘g’ de manera explícita:



A continuación, se define el método acelerado de gradiente proximal:



Cuando ‘g’ representa la función indicadora, de un set compacto, el método Frank-Wolfe resuelve la ecuación de arriba sin el término cuadrático y puede obtener una velocidad de convergencia O(1/ϵ) en el caso convexo.

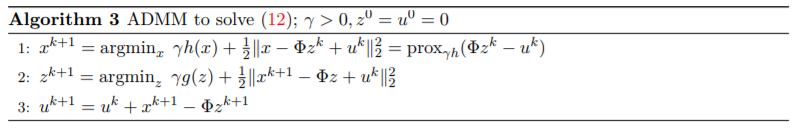
Finalmente, los métodos de gradiente proximal que explotan óptimamente las propiedades de auto concordancia de ‘f’ también son exploradas.

Objetivos Proximales

Para muchas aplicaciones, los métodos de primer orden que de los que se han hablado no son directamente aplicables. Como resultado, analizar la forma compuesta será muy útil, para esto la siguiente formulación:

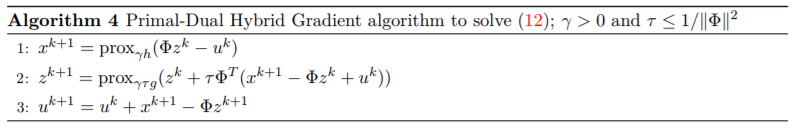


A continuación, se encuentra al algoritmo ADMM que utiliza técnicas Lagrangiana y descomposición dual:



Este algoritmo es muy apropiado para optimización distribuida y es equivalente o muy relacionado con otros algoritmos como por ejemplo el “Douglas-Rachford”. ADMM requiere de un parámetro de penalización γ como entrada y produce una secuencia que itera la factibilidad de ese acercamiento y produce un valor objetivo óptimo en el límite. A continuación, se mencionan dos advertencias del ADMM:

1. Se tiene que resolver numéricamente el paso 2 del algoritmo, excepto para cuando ΦTΦ
2. La extensión ‘naïve’ de ADMM a problemas con más de dos términos objetivos no tiene la convergencia que garantiza



El algoritmo anterior muestra una variante precondicionada de ADMM.

Para objetivos con más de dos términos, podemos utilizar técnicas de descomposición dual para tratar múltiples términos del objetivo como problemas separados y resolverlos en paralelo.

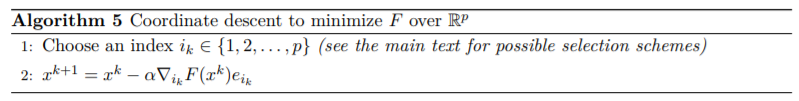
Escalamiento de Big Data por medio de aleatorización

En teoría los métodos de primer orden están correctamente posicionados para atender problemas de gran escala. A pesar de esto, en la práctica el cómputo numéricamente exacto demandado por las iteraciones puede hacer que métodos simples no factibles sean el problema a medida que las dimensiones crecen.

Métodos de descenso coordinado

Calcular el gradiente completo para PageRank requiere operaciones matriz-vector en cada iteración. Un vector más baratosería seleccionar una coordenada i de x y sólo modificar la variable correspondiente xi para mejorar la función objetivo. Esta idea captura la esencia de los métodos de descenso coordinado, que tienen una larga historia en optimización.

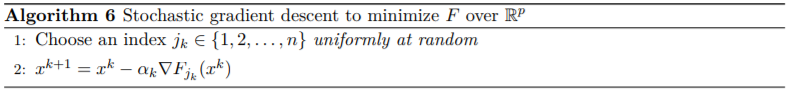
La forma general de los métodos de descenso coordinado se puede observar en el siguiente algoritmo, donde ei es el vector canónico coordinado in.



Las versiones aceleradas y compuestas de los métodos de descenso coordinado también han sido recientemente exploradas, aunque los métodos acelerados normalmente no preservan un costo iterativo bajo de las versiones no aceleradas.

Métodos de descenso gradiente

En contraste con métodos aleatorizados des 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.



Las iteraciones de gradiente estocástico se respaldan de la descomponibilidad que se mostró en el algoritmo anterior. Similar a lo establecido en los métodos de gradiente coordinado, el problema crucial de diseño en métodos de es la selección de los puntos ‘j’.

De manera general, el método de gradiente estocástico, utiliza una secuencia decreciente de tamaños de paso. Desafortunadamente esto nos lleva a una convergencia lenta del método de sub-gradiente; pero si se utiliza un tamaño de paso constate en cada iteración para el algoritmo del método de gradiente estocástico, se puede reducir rápidamente el error inicial.

Algebra Lineal Aleatorizada

Para problemas de Big Data, las operaciones básicas de álgebra lineal, tal como descomposición matricial y multiplicaciones matriz-matriz pueden ser cuellos de botella mayores debido a su dependencia superlineal en dimensiones. En caso en que objetos matriz relevantes, tengan representaciones de bajo rango la eficiencia de estos métodos mejora uniformemente.

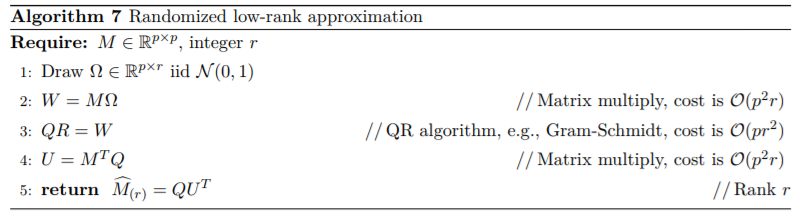
La idea detrás del álgebra lineal aleatorizada es aproximar:

 con 

O construir una representación por columna de bajo rango o una selección subset para mejorar la velocidad de cómputo. Realizar esto de manera aleatoria nos da control sobre la distribución de errores. Esta idea se generaliza a matrices de cualquier dimensión y tiene agregado el beneficio de explotar rutinas computacionales maduras en casi todos los lenguajes de programación.

A continuación, los tres impactos de aleatorizar rutinas de algebra lineal en optimización:

1. Se puede acelerar el cómputo de operadores de proximidad de funciones que dependen de valores espectrales de una matriz
2. La idea también funciona para obtener gradiente no sesgado cuando la aleatorización es seleccionada apropiadamente y aplica para todos los algoritmos de gradiente estocástico.
3. El acercamiento aleatorizado puede ser utilizado para realizar un bosquejo de funciones objetivo



El algoritmo anterior es un ejemplo de una aproximación aleatorizada de bajo rango, que es un solo paso de la iteración QR utilizando un valor inicial aleatorio

El rol de computo paralelo y distribuido

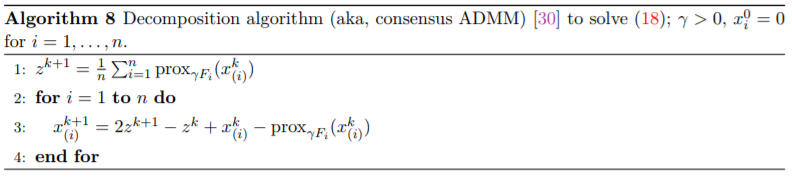
A continuación, los dos problemas al utilizar cómputo distribuido y hardware heterogéneo para métodos de primer orden.

1. Comunicación: comunicación con fallas o 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.
2. 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.

Embarrassingly parallel first-order methods

Los métodos de primer orden pueden verse beneficiados significativamente del cómputo en paralelo.

A continuación podemos observar el algoritmo de descomposición ADMM para poderlo aplicar a nuestro modelo.



Métodos de primer orden con comunicaciones reducidas o descentralizadas

En sistemas muy grandes, comunicar el gradiente o sus elementos puede crear un cuello de botella de comunicación. Los métodos de descenso coordenado proveen de un acercamiento para reducir comunicaciones. La idea básica es simplemente aplicar diversos descensos coordenados al mismo tiempo en paralelo.

Cuando el objetivo es descomponible esto es simplemente una adecuación de la versión anterior, pero de un algoritmo serial. Podemos descentralizar los requerimientos de comunicación de métodos de gradiente para objetivos descomponibles solo con modificaciones menores. El algoritmo resultante realiza una actualización de gradiente modificado al promedio de vectores parámetro solo con los vecinos con los que se comunica. Esta estrategia logra rangos de convergencia similar al método de gradiente con comunicaciones centralizadas, donde la velocidad de degradación depende de la gráfica Laplaciana de la comunicación de la red.

Modelos de primer orden asíncronos con comunicaciones descentralizadas

El gradiente y los métodos de descomposición antes vistos aún requieren de una sincronización global para poder manejar problemas descomponibles. Los algoritmos de primer orden con aleatorización pueden ser efectivos con configuraciones descentralizadas y asíncronas con la posibilidad de fallas en las comunicaciones.

Panorama de optimización convexa

Los problemas de Big Data, necesitan una revisión de como diseñamos algoritmos de optimización convexa y sugieren elecciones computaciones no convencionales. Para resolver problemas de optimización convexa en crecimiento con recursos computacionales modestos. Este artículo nos deja claro que debemos identificar los trade-offs de la estructura dependiente algorítmica.

Como todavía existen restricciones de hardware y esto guía nuestros algoritmos, podemos esperar que nuevas herramientas seguirán siendo desarrolladas para adaptar algoritmos convexos a las plataformas computaciones hetereogéneos. Para poder obtener más de la misma información debemos utilizar modelos compuestos. Esto también nos hace preguntarnos si podemos utilizar modelos compuestos para obtener la misma información pero a una mayor velocidad.