Tarea 6: Método de Gradiente Conjugado No Lineal con Half Quadrature Regularization

Juan Gerardo Fuentes Almeida

Abstract—En esta práctica se implementa el algoritmo de Gradiente Conjugado No Lineal con Half Quadrature Regularization en el problema de regularización de imágenes. Asimismo se hacen comparaciones en tiempo e iteraciones contra diversos métodos de Gradiente Conjugado No Lineal.

1 Introducción

El algoritmo de Gradiente Conjugado fue estudiado para minimizar funciones cuadráticas convexas. La version no lineal del método fue adaptado para la minimización de funciones generales convexas y funciones no lineales.

2 Teoría

2.1 Gradiente Conjugado No Lineal

2.1.1 Método de Fletcher-Reeves (FR)

El método de Gradiente Conjugado se extiende para funciones no lineales haciendo dos simples cambios al algoritmo Lineal. Primero se reemplaza el calculo del tamaño de paso α_k por una búsqueda en linea que nos permita encontrar un minino de la función no linear f a lo largo de la dirección p_k. Segundo, el residuo r es reemplazado por el gradiente de la función no linear f

Algoritmo FLetcher-Reeves

Given x_0;
Evaluate $f_0 = f(x_0), \nabla f_0 = \nabla f(x_0)$;
Set $p_0 \leftarrow -\nabla f_0, k \leftarrow 0$;
while $\nabla f_k \neq 0$
  Compute α_k (line search)
  set $x_{k+1} = x_k + \alpha_k p_k$;
  Evaluate ∇f_{k+1};
  $\beta_{k+1}^{\text{FR}} \leftarrow \dfrac{\nabla f_{k+1}^T \nabla f_{k+1}}{\nabla f_k^T \nabla f_k}$;
  $p_{k+1} \leftarrow -\nabla f_{k+1} + \beta_{k+1}^{\text{FR}} p_k$;
  $k \leftarrow k + 1$;
end (**while**)

Este algoritmo es adecuado para problemas grandes de optimización no lineal porque cada iteración solo requiere la evaluación de la función objetivo y su gradiente. No se requieren operaciones matriciales y se requieren solo una cuantos vectores para almacenamiento. Para asegurar que la dirección p_k sea siempre de descenso hacemos que α_k satisfaga las condiciones fuertes de Wolfe:

$$\nabla f(x_k + \alpha_k p_k)^T p_k \geq c_2 \nabla f_k^T p_k$$

$$|\nabla f(x_k + \alpha_k p_k)^T p_k| \leq c_2 |\nabla f_k^T p_k|$$

donde $0 < c_1 < c_2 < \frac{1}{2}$

2.1.2 Método de Polak-Ribiere (PR)

Es una variante importante del método de Fletcher-Reeves, la cual define el parámetro β_k como sigue:

$$\beta_{k+1}^{PR} = \frac{\nabla f_{k+1}^T (\nabla f_{k+1} - \nabla f_k)}{\nabla f_k^T \nabla f_k}$$

Esta β es equivalente a la de FR cuando la función es cuadrática y fuertemente convexa, y además la búsqueda en linea es exacta. Sin embargo, en casos generales el comportamiento de ambos algoritmos difiere. Experimentalmente se ha determinado que el algoritmo PR tiende a ser el mas robusto y eficiente.

Un hecho acerca del algoritmo PR es que las condiciones fuertes de Wolfe no garantizan que p_k sea siempre una dirección de descenso. Para que esta condición se mantenga se redefine β de la siguiente manera:

$$\beta_{k+1}^{PR+} = max(\beta_{k+1}^{PR}, 0)$$

2.1.3 Método de Hestenes-Steifel (HS)

Al igual que el algoritmo de PR, coincide con FR cuando la función es cuadrática y la búsqueda en linea es exacta, se define β de la siguiente manera:

$$\beta_{k+1}^{HS} = \frac{\nabla f_{k+1}^T (\nabla f_{k+1} - \nabla f_k)}{(\nabla f_{k+1} - \nabla f_k)^T p_k}$$

Este algoritmo es similar a PR en términos de convergencia y desempeño.

2.2 Half Quadrature Regularization

En esta práctica se propone un método de potenciales de regularización con preservación de bordes, para lograr esto, implementamos una función potencial a la cual se le agrega un parámetro k de regularización de no penalización sobre vecinos en los bordes, además del parametro lambda de regularización o penalización sobre el suavizamiento de vecinos fuera de los bordes:

$$\rho(U_{r,s}) = \omega_s^2 U_{r,s}^2 + (1-\omega_s)^2 k$$

donde ω_s es la ponderación que se asigna a cada vecino de un pixel, $U_{r,s} = X_r - X_s$ donde X_r representa un pixel de la imagen suavizada y X_s representa un pixel vecino.

De esta forma nuestra función objetivo queda definida por el error cuadrático sobre los pixeles de la imagen suavizada con respecto a la imagen observada, más la función potencial sobre los vecinos de cada uno de los pixeles de la imagen:

$$U(x,y) = \tfrac{1}{2} \sum_{r\in\Omega} (X_r - Y_r)^2 + \tfrac{\lambda}{2} \sum_{r\in\Omega} \sum_{s\in N_r} \rho(U_{r,s})$$

3 Implementación

Se implementan versiones de Gradiente Conjugado No Lineal con cada uno de los metodos para el calculo de β descritos anteriormente, utilizando Half Quadrature Regularization. Con estas implementaciones se resuelve el problema de suavizamiento o "denoising" mediante la minimización de la función de regularización definida por la siguiente expresión:

$$U(x,y) = \tfrac{1}{2} \sum_{r\in\Omega} (X_r - Y_r)^2 + \tfrac{\lambda}{2} \sum_{r\in\Omega} \sum_{s\in N_r} \rho(U_{r,s})$$

donde N_r es la vecindad del pixel X_r de 4 vecinos, $\rho(U_{r,s})$ es la función de regularización con $U_{r,s} = X_r - X_s$, X_r es la imagen suavizada, Y_r es la imagen observada con ruido, $r = (i,j)$ son las coordenadas de un pixel, Ω es el conjunto de pixeles en común a ambas imágenes y λ es el parámetro de de regularización o penalización.

Para maximizar la función definimos la derivada para cierto elemento r_{ij}:

$$\tfrac{\partial}{\partial r_{ij}} U(X,Y) = X_r - Y_r + \lambda \sum_{s\in N_r} \omega_s^2 [X_r - X_s]$$

Con la función objetivo definida anteriormente y su vector gradiente, implementamos el algoritmo de Gradiente Conjugado No Lineal, tal como se describe en el pseudocódigo. Para la parte de Half Quadrature Regularization se calcula ω_s de forma iterativa utilizando Gauss-Seidel, derivando la función potencial, igualando a cero y obteniendo la siguiente expresión:

$$\omega_s = \tfrac{k}{k+U_{r,s}}$$

Tal como se vio en clase, iniciamos calculando los valores iniciales de la función objetivo y su gradiente, considerando $\omega_s = 1$ para todos los vecinos, y una vez obtenida una imagen nueva a partir de la primera, actualizamos los pesos utilizando la formula obtenida anteriormente.

3.1 Búsqueda en línea

De acuerdo a las recomendaciones dadas en el libro de referencia, utilizamos las condiciones fuertes de Wolfe para poder garantizar una tamaño de paso que proporcione suficiente descenso. Esta condición se combino con un modelo cuadratico para el calculo de α_k.

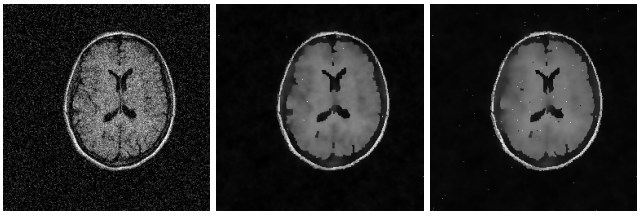
Sin embargo, se tuvieron problemas para lograr implementar un método de búsqueda en linea eficiente a partir de la implementación de la Tarea 2, se observó que en la mayoría de los casos no se obtiene un valor de α_k que cumpla con la condición fuerte de Wolfe. En esos casos, se conserva el valor de α_k que se obtuvo en la iteración anterior.

4 Resultados

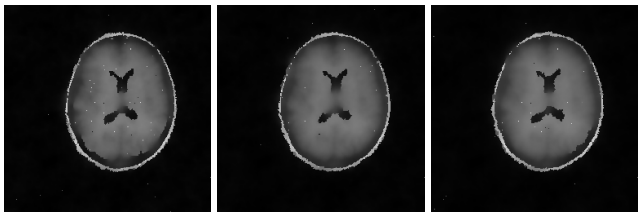
A continuación se muestran las imágenes que se obtuvieron durante la ejecución de la práctica, para diferentes valores de λ en una imagen de prueba, utilizando la β de Polak-Ribiere:

Der. a izq.: Imagen observada, $(\lambda, k) = (10, 400), (20, 400)$

Der. a izq.: $(\lambda, k) = (30, 400), (40, 400), (50, 400)$



Der. a izq.: Imagen observada, $(\lambda, k) = (10, 400), (20, 400)$



Der. a izq.: $(\lambda, k) = (30, 400), (40, 400), (50, 400)$

El programa implementado en esta práctica recibe el nombre del archivo de la imagen, el parámetro λ, la constante de regularización k y un indicador de la β a utilizar (1=Fletcher-Reeves, 2=Polak-Ribiere, 3=Hestenes-Steifel), luego ejecuta los algoritmos de Gradiente Conjugado no Lineal para suavizar la imagen con estos parámetros.

Se imprime en pantalla el numero de la iteración, el valor de la función objetivo, los valores de α, β y el valor de la norma del gradiente para cada iteración. La condición de paro se da a las 500 iteraciones o cuando el valor de la función objetivo varíe en menos de 100 unidades con respecto al valor de la iteración anterior. Se muestra también una tabla donde se comparan cada uno de métodos con relación a su tiempo de ejecución, número de iteraciones y valor de la función objetivo:

Lambda	k	Beta	Ite	T	U(X,Y)	Grad
25	100	FR	500	26.124	1.50E+008	30314
50	100	FR	500	65.34	3.80E+008	50769
25	100	PR	500	26.03	3.70E+007	267
50	100	PR	500	38.56	4.90E+007	200
25	100	HS	146	8.758	3.30E+007	706
50	100	HS	395	20.021	4.50E+007	307
25	200	FR	500	24.39	1.00E+009	30868
50	200	FR	500	25.14	2.70E+009	54694
25	200	PR	276	16.247	3.70E+007	62
50	200	PR	500	24.55	4.80E+007	364
25	200	HS	232	13.86	3.90E+007	97
50	200	HS	500	25183	1.20E+009	18165

Tabla comparativa entre los algoritmos de Gradiente Conjugado implementados.

5 Conclusiones

En general se observo durante la práctica que el método de Polak-Ribiere resultó ser mas eficiente comparado con los demás, lo cual es apoyado por la experimentación mencionada en el libro de referencia, también se observo que resulto complicado obtener un tamaño de paso adecuado para una función no lineal como la que se estuvo tratando. Sorpresivamente se encontró que el método de Fletcher Reeves no resultó tan bueno como se esperaba, basta con ver la tabla anterior para apreciar que es muy inestable para funciones no lineales y no resultó ser un minimizador eficiente.

References

[1] Jorge Nocedal and Stephen J. Wright. *Numerical optimization*. Springer series in operations research and financial engineering. Springer, New York, NY, 2. ed. edition, 2006.

Appendix

El programa está implementado tomando en cuenta todas estandarizaciones indicadas en el curso.

Un *makefile* ha sido generado, el cual soporta los comandos *make*, *run* and *clean*. El programa recibe el nombre del archivo de la imagen, el parámetro λ, la constante de regularización k y un indicador de la β a utilizar.