Tarea 9: Método BFGS y Gradiente Estocástico

Juan Gerardo Fuentes Almeida

Abstract—En esta práctica se implementan los algoritmos BFGS y BFGS estocástico para resolver problemas de optimización con Mínimos Cuadrados No Lineales, aplicados a desenvolvimiento de fases.

1 Introducción

En Problemas de Mínimos Cuadrados, la función objetivo f tiene la forma especial:

$$f(x) = \frac{1}{2}\sum_{j=1}^{m} r_j^2(x)$$

donde cada r_j es una función residual suave de $\Re^n$ a $\Re$, esta forma especial hace que el problema sea mas fácil de resolver comparado con un problema general de optimización sin restricciones.

Ensamblamos los componentes individuales de r_j en un vector residual $r : \Re^n \to \Re^m$, de la siguiente manera:

$$r(x) = (r_1(x), r_2(x), ..., r_m(x))^T$$

Con esta notación, podemos reescribir f como $f(x) = \frac{1}{2} \parallel r_j^2(x) \parallel_2^2$, y sus derivadas en términos del Jacobiano $J(x)$:

$$J(x) = \left[\frac{\partial r_j}{\partial x_i}\right]_{\substack{j=1,2,...,m \\ i=1,2,...,n}} = \begin{bmatrix} \nabla r_1(x)^T \\ \nabla r_2(x)^T \\ \vdots \\ \nabla r_m(x)^T \end{bmatrix}$$

El gradiente y el Hessiano de f se pueden expresar de la siguiente manera:

$$\begin{aligned} \nabla f(x) &= \sum_{j=1}^{m} r_j(x)\nabla r_j(x) = J(x)^T r(x), \\ \nabla^2 f(x) &= \sum_{j=1}^{m} \nabla r_j(x)\nabla r_j(x)^T + \sum_{j=1}^{m} r_j(x)\nabla^2 r_j(x) \\ &= J(x)^T J(x) + \sum_{j=1}^{m} r_j(x)\nabla^2 r_j(x). \end{aligned}$$

En muchas aplicaciones, las primeras derivadas parciales de los residuales, y por tanto, de la Matriz Jacobiana $J(x)$ son relativamente fáciles de calcular. Asimismo, utilizando $J(x)$ podemos calcular el primer término $J(x)^T J(x)$ del Hessiano sin evaluar las segundas derivadas de la función $r(x)$, esto es distintivo de los problemas de mínimos cuadrados, ya que los residuos r_j están cerca de la solución, y por tanto, $\nabla^2 r_j(x)$ es relativamente pequeño.

2 Teoría

2.1 Método BFGS

Es el algoritmo Quasi-Newton más utilizado, se deriva de la optimización de la serie de Taylor:

$$m_k(p) = f_k + \nabla f_k^T p + \frac{1}{2} p^T B_k p$$

Para una matriz B_k simétrica y positiva definida, la cual se actualiza en cada iteración, como ya antes se ha demostrado, el minimizador para esta función esta dado por:

$$p_k = -B_k^{-1} f_k$$

esta dirección de búsqueda se utiliza para calcular la nueva iteración $x_{k+1} = x_k + \alpha_k p_k$, donde el tamaño de paso α_k satisface las condiciones de Wolfe.

Para determinar la siguiente iteración B_{k+1}, se impone la siguiente condición, denominada Ecuación Secante:

$$B_{k+1} s_k = y_k$$

Para $s_k = x_{k+1} - x_k$ y $y_k = \nabla f_{k+1} - \nabla f_k$, además, se impone la condición de que esa nueva iteración debe ser lo mas cercana posible a la iteración anterior, lo que equivale a resolver el problema:

$$min_B ||B - B_k||$$

sujeto a $B = B^T$ y $B_{k+1} s_k = y_k$

Davidon, Fletcher y Powell en 1959 propusieron la siguiente formula para actualizar la matriz B_k:

$$B_{k+1} = (I - \rho_k y_k s_k^T) B_k (I - \rho_k s_k y_k^T) + \rho_k y_k y_k^T$$

con $\rho = 1/y_k^T s_k$, pero después fue sobrepasada por la formula BFGS, en la cual se actualiza la inversa de la matriz B_k denotada como H_k:

$$H_{k+1} = (I - \rho_k s_k y_k^T) H_k (I - \rho_k y_k s_k^T) + \rho_k s_k s_k^T$$

Algorithm 1 (BFGS Method).
Given starting point x_0, convergence tolerance $\epsilon > 0$,
inverse Hessian approximation H_0;
$k \leftarrow 0$;
while $\|\nabla f_k\| > \epsilon$;
Compute search direction $p_k = -H_k \nabla f_k$;
Set $x_{k+1} = x_k + \alpha_k p_k$ where α_k is computed from a line search procedure to satisfy the Wolfe conditions;
Define $s_k = x_{k+1} - x_k$ and $y_k = \nabla f_{k+1} - \nabla f_k$;
$H_{k+1} = (I - \rho_k s_k y_k^T) H_k (I - \rho_k y_k s_k^T) + \rho_k s_k s_k^T$;
$k \leftarrow k + 1$;
end (while)

2.2 Gradiente Estocastico

Si queremos resolver el problema general de optimización:

$$min_{x \in \Re^n} f(x) = \sum_{i=1}^{n} f_i(x)$$

Sabemos que equivale a resolver:

$$\nabla f(x) = \sum_{i=1}^{n} \nabla f_i(x)$$

por analogía

$$\nabla f(x) = nE[\nabla f_i(x)]$$

luego podemos interpretar el gradiente de $f(x)$ como una valor esperado y podemos estimarlo sobre una muestra de una población U:

$$S = U([1, n], m)$$

con $m << n$, por tanto podemos definir el Jacobiano del problema de la siguiente manera:

$$\nabla f \simeq \sum_{i \in S} \nabla f_i(x) = \tilde{J}^T \tilde{r}$$

3 IMPLEMENTACIÓN

Sea $f(x)$ la fase desenvuelta de una imagen, y $g(x)$ su fase envuelta, por la propiedad de Nyquist, si la derivada de una función esta acotada en $(-\pi, \pi)$, es decir, $f(x) - f(x - e_1) \in (-\pi, \pi)$, entonces:

$$f(x) - f(y) = W[g(x) - q(y)]$$

siendo W el operador de envolvimiento y $\| y - x \| = 1$

luego, para recuperar $f(x)$ de $g(x)$, calculamos:

$$g_1'(x) = g(x) - g(x - e_1),\ g_2'(x) = g(x) - g(x - e_2)$$

y envolvemos la función resultante:

$$\widehat{g_1'(x)} = W[g_1'(x)],\ \widehat{g_2'(x)} = W[g_2'(x)]$$

Por tanto, por el criterio de Nyquist, podemos aproximar el residuo entre esta función y la derivada de la fase desenvuelta mediante mínimos cuadrados:

$$U(f) = \sum_{\Omega} [\widehat{g_1'(x)} - f_1'(x)]^2 + \sum_{\Omega} [\widehat{g_2'(x)} - f_2'(x)]^2$$

donde $f_1'(x) = f(x) - f(x - e_1)$ y $f_2'(x) = f(x) - f(x - e_2)$ son las derivadas de la fase desenvuelta, las cuales podemos aproximar mediante Bases de Funciones Radiales:

$$f(x) = \sum_{ij} \alpha_{ij} \phi(i, j, x)$$
$$\text{con } \phi(i, j, x) = \exp[\tfrac{-1}{2\sigma_{ij}}(x - \mu_{ij})^2]$$

luego
$f_1'(x) = \sum_{ij} \alpha_{ij} [\phi(i, j, x) - \phi(i, j, x - e_1)] = \sum_{ij} \alpha_{ij} \varphi_1(i, j, x)$
y $f_2'(x) = \sum_{ij} \alpha_{ij} [\phi(i, j, x) - \phi(i, j, x - e_2)] = \sum_{ij} \alpha_{ij} \varphi_2(i, j, x)$

luego, para recuperar $f(x)$ de $g(x)$, tendremos que minimizar la siguiente expresión:

$$U(f) = \sum_{\Omega} [\widehat{g_1'(x)} - f_1'(x)]^2 + \sum_{\Omega} [\widehat{g_2'(x)} - f_2'(x)]^2$$

3.1 Optimización de α

Para optimizar el valor de α, el cual es la amplitud de las funciones radiales, se construye un Jacobiano cuyas filas son los componentes en x y en y de la función objetivo (2 veces el numero de pixeles de la imagen)

$$J(\alpha) = \begin{pmatrix} \frac{\partial r(x_1)}{\partial \alpha_1} & \frac{\partial r(x_1)}{\partial \alpha_2} & \cdots & \frac{\partial r(x_1)}{\partial \alpha_m} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial r(x_N)}{\partial \alpha_1} & \frac{\partial r(y_N)}{\partial \alpha_2} & \cdots & \frac{\partial r(x_N)}{\partial \alpha_m} \\ \frac{\partial r(y_1)}{\partial \alpha_1} & \frac{\partial r(y_1)}{\partial \alpha_2} & \cdots & \frac{\partial r(y_1)}{\partial \alpha_m} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial r(y_N)}{\partial \alpha_1} & \frac{\partial r(y_N)}{\partial \alpha_2} & \cdots & \frac{\partial r(y_N)}{\partial \alpha_m} \end{pmatrix}$$

El gradiente de la función esta determinado por $J_k^T r_k$.

Cada elemento de esta matriz esta definido por la siguiente expresión:

$$\frac{\partial r(x_k)}{\partial \alpha_{ij}} = -\varphi_1(i, j, x_k) = -\frac{(x_{k_1} - \mu_{ij_1})}{\sigma_{ij}^2} \phi(i, j, x_k)$$
$$\frac{\partial r(y_k)}{\partial \alpha_{ij}} = -\varphi_2(i, j, x_k) = -\frac{(x_{k_2} - \mu_{ij_2})}{\sigma_{ij}^2} \phi(i, j, x_k)$$

3.2 Optimización de μ

Para optimizar el valor de μ, el cual es la media de las funciones radiales, se construye un Jacobiano cuyas filas son los componentes en x y en y de la función objetivo (2 veces el numero de pixeles de la imagen), y cuyas columnas

están repartidas entre las componentes en x y en y de la medias (2 veces el numero de funciones radiales):

$$J(\mu) = \begin{pmatrix} \frac{\partial r(x_1)}{\partial \mu(x_1)} & \cdots & \frac{\partial r(x_1)}{\partial \mu(x_m)} & \frac{\partial r(x_1)}{\partial \mu(y_1)} & \cdots & \frac{\partial r(x_1)}{\partial \mu(x_m)} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ \frac{\partial r(x_N)}{\partial \mu(x_1)} & \cdots & \frac{\partial r(x_N)}{\partial \mu(x_m)} & \frac{\partial r(x_N)}{\partial \mu(y_1)} & \cdots & \frac{\partial r(x_N)}{\partial \mu(x_m)} \\ \frac{\partial r(y_1)}{\partial \mu(x_1)} & \cdots & \frac{\partial r(y_1)}{\partial \mu(x_m)} & \frac{\partial r(y_1)}{\partial \mu(y_1)} & \cdots & \frac{\partial r(y_1)}{\partial \mu(x_m)} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ \frac{\partial r(y_N)}{\partial \mu(x_1)} & \cdots & \frac{\partial r(y_N)}{\partial \mu(x_m)} & \frac{\partial r(y_N)}{\partial \mu(y_1)} & \cdots & \frac{\partial r(y_N)}{\partial \mu(x_m)} \end{pmatrix}$$

El gradiente de la función esta determinado por $J_k^T r_k$.

Cada elemento de esta matriz esta definido por la siguiente expresión:

$$\frac{\partial r(x_k)}{\partial \mu(x_{ij})} = -\alpha_{ij} \frac{\partial \varphi_1(i,j,x_k)}{\partial \mu(x_{ij})} = -\frac{\alpha_{ij}}{\sigma_{ij}^2} \phi(i,j,x_k)[1 - \frac{(x_{k_1} - \mu_{ij_1})^2}{\sigma_{ij}^2}]$$

$$\frac{\partial r(x_k)}{\partial \mu(y_{ij})} = \frac{\alpha_{ij}}{\sigma_{ij}^4} \phi(i,j,x_k)(x_{k_1} - \mu_{ij_1})(x_{k_2} - \mu_{ij_2})$$

$$\frac{\partial r(y_k)}{\partial \mu(x_{ij})} = \frac{\alpha_{ij}}{\sigma_{ij}^4} \phi(i,j,x_k)(x_{k_1} - \mu_{ij_1})(x_{k_2} - \mu_{ij_2})$$

$$\frac{\partial r(y_k)}{\partial \mu(y_{ij})} = -\alpha_{ij} \frac{\partial \varphi_1(i,j,x_k)}{\partial \mu(y_{ij})} = -\frac{\alpha_{ij}}{\sigma_{ij}^2} \phi(i,j,x_k)[1 - \frac{(x_{k_2} - \mu_{ij_2})^2}{\sigma_{ij}^2}]$$

3.3 Optimización de σ

Para optimizar el valor de σ, el cual es la desviación estándar de las funciones radiales, se construye un Jacobiano cuyas filas son los componentes en x y en y de la función objetivo (2 veces el numero de pixeles de la imagen)

$$J(\alpha) = \begin{pmatrix} \frac{\partial r(x_1)}{\partial \sigma_1} & \frac{\partial r(x_1)}{\partial \sigma_2} & \cdots & \frac{\partial r(x_1)}{\partial \sigma_m} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial r(x_N)}{\partial \sigma_1} & \frac{\partial r(y_N)}{\partial \sigma_2} & \cdots & \frac{\partial r(x_N)}{\partial \sigma_m} \\ \frac{\partial r(y_1)}{\partial \sigma_1} & \frac{\partial r(y_1)}{\partial \sigma_2} & \cdots & \frac{\partial r(y_1)}{\partial \sigma_m} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial r(y_N)}{\partial \sigma_1} & \frac{\partial r(y_N)}{\partial \sigma_2} & \cdots & \frac{\partial r(y_N)}{\partial \sigma_m} \end{pmatrix}$$

El gradiente de la función esta determinado por $J_k^T r_k$.

Cada elemento de esta matriz esta definido por la siguiente expresión:

$$\frac{\partial r(x_k)}{\partial \sigma_{ij}} = -\frac{\alpha_{ij}}{\sigma_{ij}^3}(x_{k_1} - \mu_{ij_1})\phi(i,j,x_k)[2 - \frac{(x_k - \mu_{ij})^2}{\sigma_{ij}}]$$

$$\frac{\partial r(y_k)}{\partial \sigma_{ij}} = -\frac{\alpha_{ij}}{\sigma_{ij}^3}(x_{k_2} - \mu_{ij_2})\phi(i,j,x_k)[2 - \frac{(x_k - \mu_{ij})^2}{\sigma_{ij}}]$$

4 Resultados

A continuación se muestran las imágenes que se obtuvieron durante la ejecución de la práctica, para diferente número de funciones radiales en una imagen de prueba, utilizando los métodos de BFGS y BFGS con gradiente estocástico:

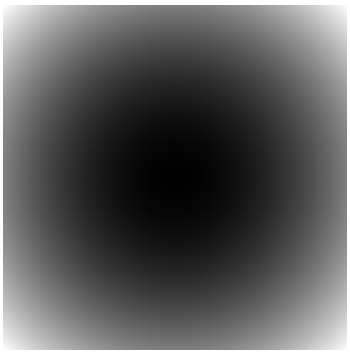
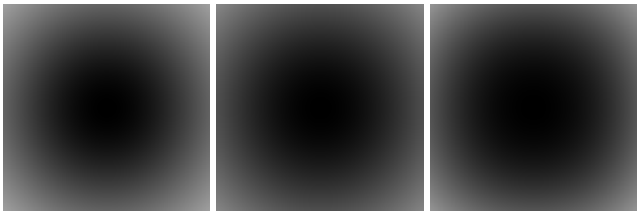
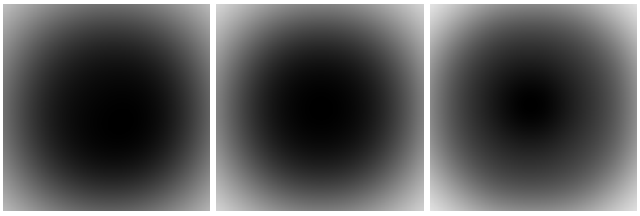


Imagen Desenvuelta

Los siguientes resultados fueron obtenidos con el método BFGS:

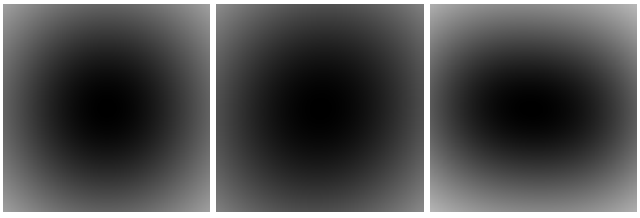


De derecha a izquierda: Resultados para 1, 4 y 9 funciones radiales.



De derecha a izquierda: Resultados para 16, 25 y 36 funciones radiales.

Los siguientes resultados fueron obtenidos con el método SGD:



De derecha a izquierda: Resultados para 1, 4 y 9 funciones radiales.

5 Conclusiones

La siguiente tabla muestra el desempeño de los algoritmos implementados en esta práctica, para diferente numero de RBF's. Como se observa, el algoritmo SGD, que fue implementado para una muestra del 50 por ciento, en general realiza más iteraciones, lo que afecta directamente en el

tiempo de ejecución, esto resultó contraproducente, ya que como se definió el criterio de paro en función de la razón de cambio entre la evaluación de la función objetivo, el algoritmo sigue iterando cuando el resultado todavía puede ser mejorado, por lo que no se observo ninguna mejora al implementarse el método estocástico.

Método	No. de RBF	Iteraciones	Tiempo	F. Objetivo
BFGS	1	2	1.996	25.030559
BFGS	4	2	3.223	47.530311
BFGS	9	6	11.372	31.844772
BFGS	16	5	24.467	47.571586
BFGS	25	4	39.852	66.790142
BFGS	36	5	59.891	56.426846
SGD	1	3	1.37	25.030674
SGD	4	3	4.668	53.823218
SGD	9	5	28.27	66.607019
SGD	16	5	26.854	64.13952
SGD	25	6	44.769	164.398782
SGD	36	5	55.208	225.142078

Desempeño de los métodos BFGS y SGD

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.

El *makefile* que se ha generado, incluye los comandos *make*, *run*, *test* y *clean*. El programa recibe el nombre del archivo de la imagen y el parámetro *methodID*, el cual indica al programa que método de optimización se va a usar, 1 para BFGS y 2 para SGD, el comando *test* ejecuta 20 iteraciones del método de BFGS con 4 gaussianas.