Práctica: Optimización Funcional

1 Teoría

Consideremos una función $f : \mathbb{R}^n \to \mathbb{R}$ ("función costo"), el objetivo es encontrar x^* que sea el mínimo de la función. Es decir, $f(\mathbf{x}^*) \leq f(\mathbf{x})\ \forall x \in \mathbb{R}^n$. Supongamos conocida $\nabla f(\mathbf{x}) = \left(\frac{\partial f}{\partial x_1}(\mathbf{x}), \ldots, \frac{\partial f}{\partial x_n}(\mathbf{x})\right)$ (el gradiente de f) en todo punto $\mathbf{x} \in \mathbb{R}^n$, se propone el siguiente algoritmo:

1. Fijar un valor $\alpha > 0 \in \mathbb{R}$, la "tasa" del descenso.
2. Fijar un valor $\mathbf{x}_0$, el valor inicial.
3. Determinar $\mathbf{x}_{n+1} = \mathbf{x}_n - \alpha \nabla f(\mathbf{x}_n)$ (descenso por el gradiente).
4. Repetir una cantidad fija N de pasos, o bien hasta que $\|\nabla f(\mathbf{x})\|$ sea menor que un valor $\epsilon > 0$ prefijado.

2 Práctica

1. Implemente una función **descGradiente** que reciba los siguiente parámetros:
   - una función F, que es la función a minimizar.
   - una función $gradF$, que es el gradiente de la función a minimizar (en caso de que sea "difícil" de calcular, acá se podrá pasar una discretización por cociente incremental).
   - un valor x_0, que es el valor inicial a considerar.
   - un valor $\alpha > 0$, la tasa del método.
   - un N, la cantidad máxima de pasos a ejecutar.
   - un ϵ, el umbral de convergencia para el gradiente.

   La función **descGradiente** habrá de ejecutar el mecanismo propuesto en la parte teórica N veces y devolver como candidato a mínimo local el valor x obtenido en la última iteración.
2. Utilizando la función antes programada, hallar el mínimo de la función $f(x) = x^2 + 5$, empezando desde $x_0 = 3$.
3. Haga lo mismo para la función $f(x_1, x_2) = 5x_1^2 + x_2^2 + 4x_1x_2 - 14x_1 - 6x_2 + 20$. Como comentario, el mínimo global de esta función es $(x_1^*, x_2^*) = (1, 1)$.
4. Implemente un mecanismo de regresión lineal basado en el método de descenso por el gradiente. Para eso, tengamos en cuenta la siguiente notación:

$$h_\beta(x) = \beta_0 * 1 + \beta_0 x_1 + \ldots + \beta_p x_p.$$

Recordemos que la función de costo sobre una muestra aleatoria (total, de training o de testing) $(x^{(1)}, y^{(1)}), \ldots, (x^{(n)}, y^{(n)})$ es

$$J(\beta) = \sum_{i=1}^{n} (h_\theta(x^{(i)}) - y_i)^2 = \sum_{i=1}^{n} (\beta^T x^{(i)} - y_i)^2.$$

(a) Obtenga analíticamente $\frac{\partial J}{\partial \beta_j}$ y en base a eso construya ∇J.

(b) Obtenga la expresión de actualización de β_j mediante descenso por el gradiente.

(c) Implemente un algoritmo en R que permita resolver el problema de regresión por cuadrados mínimos a través del esquema de descenso por el gradiente. Para eso, haga que las actualizaciones sigan el esquema **batch update**. Para eso:

i. Se procede parámetro por parámetro iniciando un acumulador en cero.

ii. Se actualiza el acumulador con el aporte del dato i-ésimo.

iii. Se actualiza el parámetro en cuestión.

iv. Se repite el esquema N veces o bien hasta cuando la diferencia entre todos los paráemtros no sea significativa.

Algunos autores sugieren que para que el mecanismo de descenso por el gradiente funcione más eficientemente, todos los datos debieran estar "estandarizados". Es decir, que los datos estén más o menos dentro de un mismo rango numérico. Existen dos formas clásicas de estandarizar las covariables (salvando el intercept):
Opción 1:

$$\tilde{X}_j^{(i)} = \frac{X_j^{(i)} - \min(X_J)}{\max(X_j) - \min(X_j)}$$

Opción 2:

$$\tilde{X}_j^{(i)} = \frac{X_j^{(i)} - \mu_j}{\sigma_j}$$

Si hace uso de estos mecanismos, recuede que una vez obtenidos los parámetros estos sirven para los datos estandarizados; hay que "desestandarizar" para definir los parámetros finales con los datos originales. O bien, alternativamente, a la hora de hacer predicciones se sigue con los datos estandarizados pero cada vez que se desea hacer predicción se estandarizan las covariables con los valores obtenidos en el entrenamiento.

5. Proceda con la siguiente simulación:

(a) Genere 100 datos $x_i \sim U(-3, 3)$.

(b) Defina $y_i = 2 + 0.5 * x + 0.1 * x^2 + \epsilon_i$ con ϵ_i una normal estándar.

(c) Armar la matriz X de diseño con los términos de intercept, x y x^2.

(d) Considere estandarizar los datos (o no) con alguno de los dos mecanismos, obviando la columna de intercept.

(e) Ensayando con distintos valores de α y de corte entre diferencia de normas de *beta* de iteración en iteración, efectúe una regresión por descenso por el gradiente. Vaya comparando con el valor "exacto" devuelto por el comando **lm** con los datos estandarizados.

6. Tome cualquiera de los problemas hechos en las prácticas anteriores referidas a regresión lineal y reimpleméntelos utilizando descenso por el gradiente. No nos es de interés fundamental la selección de modelo, con lo cual considere cualquier modelo y entrene con la totalidad de los datos. Compare los parámetros obtenidos con el devuelto por **lm**.

7. Investigue la función **optim** de R, que ofrece diversos métdos de optimización (gradiente conjugado, BFGS, etc), los cuales en general no necesitan que se especifique una tasa α. Utilizando alguno de estos métodos rehaga el ejercicio de simulación y evalúe diferencias, en performance, en resultados, en facilidad de uso, etc. Evalúe también la necesidad de tener que estandarizar o no los datos.