Técnicas de optimización numérica

Gradiente

Si $f: \mathbb{R}^n \to \mathbb{R}$ es un campo escalar entonces el gradiente de f en $\boldsymbol{x} = (x_1, \ldots, x_i, \ldots, x_n)$ se define como el campo vectorial $\nabla f: \mathbb{R}^n \to \mathbb{R}^n$ cuyas componentes son las derivadas parciales del campo escalar, esto es:

$$\nabla f(\boldsymbol{x}) = \left(\frac{\partial f(\boldsymbol{x})}{\partial x_1}, \ldots, \frac{\partial f(\boldsymbol{x})}{\partial x_i}, \ldots, \frac{\partial f(\boldsymbol{x})}{\partial x_n} \right)$$

Ejemplo:

1. Dada la función $f(x, y, z) = 2x + 3y^2 - \operatorname{sen}(z)$ su vector gradiente es el siguiente:

$$\nabla f = \left(\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z} \right) = (\, 2, 6y, -\cos(z) \,).$$

Jacobiana

Suponga $\boldsymbol{F}: \mathbb{R}^n \to \mathbb{R}^m$ que es una función tal que sus derivadas parciales de primer orden existen en $\mathbb{R}^n$. Esta función toma un punto $\boldsymbol{x} \in \mathbb{R}^n$ y devuelve un vector $\boldsymbol{F}(\boldsymbol{x}) \in \mathbb{R}^m$. La matriz Jacobiana de $\boldsymbol{F}$, denotada por $\boldsymbol{J}$, está definida como una matriz de tamaño $m \times n$ cuya (i, j)-ésima entrada es :

$$\boldsymbol{J_{i,j}} = \frac{\partial \boldsymbol{f_i}(\boldsymbol{x})}{\partial x_j}$$

o de forma explícita

$$\mathbf{J} = \begin{bmatrix} \dfrac{\partial \mathbf{F}}{\partial x_1} & \cdots & \dfrac{\partial \mathbf{F}}{\partial x_n} \end{bmatrix} = \begin{bmatrix} \nabla^T f_1 \\ \vdots \\ \nabla^T f_m \end{bmatrix} = \begin{bmatrix} \dfrac{\partial f_1}{\partial x_1} & \cdots & \dfrac{\partial f_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \dfrac{\partial f_m}{\partial x_1} & \cdots & \dfrac{\partial f_m}{\partial x_n} \end{bmatrix}$$

donde $\nabla^T f_i(\boldsymbol{x})$ es la traspuesta del gradiente de la i-ésima componente.

Ejemplo:

Supóngase la función $\mathbf{F} : \mathbb{R}^3 \to \mathbb{R}^4$, cuyas funciones componentes son:

$$
\begin{aligned}
y_1 &= x_1 \\
y_2 &= 5x_3 \\
y_3 &= 4x_2^2 - 2x_3 \\
y_4 &= x_3 \operatorname{sen}(x_1)
\end{aligned}
$$

tiene asociada como matriz jacobiana

$$
\mathbf{J_F}(x_1, x_2, x_3) = \begin{bmatrix} \dfrac{\partial y_1}{\partial x_1} & \dfrac{\partial y_1}{\partial x_2} & \dfrac{\partial y_1}{\partial x_3} \\ \dfrac{\partial y_2}{\partial x_1} & \dfrac{\partial y_2}{\partial x_2} & \dfrac{\partial y_2}{\partial x_3} \\ \dfrac{\partial y_3}{\partial x_1} & \dfrac{\partial y_3}{\partial x_2} & \dfrac{\partial y_3}{\partial x_3} \\ \dfrac{\partial y_4}{\partial x_1} & \dfrac{\partial y_4}{\partial x_2} & \dfrac{\partial y_4}{\partial x_3} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 5 \\ 0 & 8x_2 & -2 \\ x_3 \cos x_1 & 0 & \operatorname{sen} x_1 \end{bmatrix}.
$$

Hessiana

Sea $f\colon \mathbb{R}^n \to \mathbb{R}$ un campo escalar cuyas derivadas parciales de segundo orden existen. La **matriz hessiana** de f, denotada por $Hess\ f(\boldsymbol{x})$, $H_f(\boldsymbol{x})$, es una matriz cuadrada $n \times n$ definida como:

$$
Hess\ f(\boldsymbol{x}) = \begin{bmatrix} \dfrac{\partial^2 f(x)}{\partial x_1^2} & \dfrac{\partial^2 f(x)}{\partial x_1 \partial x_2} & \cdots & \dfrac{\partial^2 f(x)}{\partial x_1 \partial x_n} \\ \dfrac{\partial^2 f(x)}{\partial x_2 \partial x_1} & \dfrac{\partial^2 f(x)}{\partial x_2^2} & \cdots & \dfrac{\partial^2 f(x)}{\partial x_2 \partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \dfrac{\partial^2 f(x)}{\partial x_n \partial x_1} & \dfrac{\partial^2 f(x)}{\partial x_n \partial x_2} & \cdots & \dfrac{\partial^2 f(x)}{\partial x_n^2} \end{bmatrix}
$$

Teorema de Schwarz: El orden de derivación no importa, es decir derivar parcialmente primero con respecto la variable x_1 y después respecto la variable x_2 es lo mismo que derivar parcialmente primero con respecto x_2 y luego respecto x_1, esto es:

$$
\frac{\partial^2 f(\boldsymbol{x})}{\partial x_i \partial x_j} = \frac{\partial^2 f(\boldsymbol{x})}{\partial x_j \partial x_i}
$$

La matriz Hessiana de una función f es la matriz jacobiana, J, del gradiente de la misma función: $H_f(\boldsymbol{x}) = J\left(\nabla^{\mathsf{T}} f(\boldsymbol{x})\right)$

La traza de la matriz Hessiana es equivalente al **operador de Laplace**: $tr\left(H_f(\boldsymbol{x})\right) = \Delta f(\boldsymbol{x})$. Esta igualdad se puede demostrar fácilmente, ya que la definición del operador de Laplace es la **divergencia del gradiente** de una función: $\Delta f(\boldsymbol{x}) = \nabla \cdot \left(\nabla f(\mathbf{x})\right) = (\nabla \cdot \nabla) f(\boldsymbol{x}) = \nabla^2 f(\boldsymbol{x})$. Por lo tanto su expresión es:

$$\Delta f(\boldsymbol{x}) = \frac{\partial^2 f(\boldsymbol{x})}{\partial x_1^2} + \frac{\partial^2 f(\boldsymbol{x})}{\partial x_2^2} + \cdots + \frac{\partial^2 f(\boldsymbol{x})}{\partial x_n^2}$$

Ejemplo:

Calcula la hessiana de $f(\boldsymbol{x}) = x_1^3 - 2x_1x_2 - x_2^6$

1. Calculamos las derivadas de primer orden:

$$f_{x_1}(\boldsymbol{x}) = \frac{\partial f(\boldsymbol{x})}{\partial x_1} = \frac{\partial}{\partial x_1}(x_1^3 - 2x_1x_2 - x_2^6) = 3x_1^2 - 2x_2$$

$$f_{x_2}(\boldsymbol{x}) = \frac{\partial f(\boldsymbol{x})}{\partial x_2} = \frac{\partial}{\partial x_2}(x_1^3 - 2x_1x_2 - x_2^6) = -2x_1 - 6x_2^5$$

$$\nabla f(\boldsymbol{x}) = \left(\frac{\partial f(\boldsymbol{x})}{\partial x_1}, \frac{\partial f(\boldsymbol{x})}{\partial x_2}\right) = (3x_1^2 - 2x_2, -2x_1 - 6x_2^5)$$

2. Calculamos las derivadas de segundo orden

$$f_{x_1x_1}(\boldsymbol{x}) = \frac{\partial}{\partial x_1}\left(\frac{\partial f(\boldsymbol{x})}{\partial x_1}\right) = \frac{\partial^2 f(\boldsymbol{x})}{\partial x_1^2} = \frac{\partial}{\partial x_1}(3x_1^2 - 2x_2) = 6x_1^2$$

$$f_{x_1x_2}(\boldsymbol{x}) = \frac{\partial}{\partial x_2}\left(\frac{\partial f(\boldsymbol{x})}{\partial x_1}\right) = \frac{\partial^2 f(\boldsymbol{x})}{\partial x_2 \partial x_1} = \frac{\partial}{\partial x_2}(3x_1^2 - 2x_2) = -2$$

$$f_{x_2x_1}(\boldsymbol{x}) = \frac{\partial}{\partial x_1}\left(\frac{\partial f(\boldsymbol{x})}{\partial x_2}\right) = \frac{\partial^2 f(\boldsymbol{x})}{\partial x_1 \partial x_2} = \frac{\partial}{\partial x_1}(-2x_1 - 6x_2^5) = -2$$

$$f_{x_2x_2}(\boldsymbol{x}) = \frac{\partial}{\partial x_2}\left(\frac{\partial f(\boldsymbol{x})}{\partial x_2}\right) = \frac{\partial^2 f(\boldsymbol{x})}{\partial x_2^2} = \frac{\partial}{\partial x_2}(-2x_1 - 6x_2^5) = -30x_2^4$$

$$H_f(\boldsymbol{x}) = J\left(\nabla^{\intercal} f(\boldsymbol{x})\right) = J\begin{pmatrix} \dfrac{\partial f(\boldsymbol{x})}{\partial x_1} \\ \dfrac{\partial f(\boldsymbol{x})}{\partial x_2} \end{pmatrix} = \begin{pmatrix} \dfrac{\partial^2 f(\boldsymbol{x})}{\partial x_1^2} & \dfrac{\partial^2 f(\boldsymbol{x})}{\partial x_2 \partial x_1} \\ \dfrac{\partial^2 f(\boldsymbol{x})}{\partial x_1 \partial x_2} & \dfrac{\partial^2 f(\boldsymbol{x})}{\partial x_2^2} \end{pmatrix}$$

$$H_f(\boldsymbol{x}) = \begin{pmatrix} 6x_1^2 & -2 \\ -2 & -30x_2^4 \end{pmatrix}$$

Hessiana Orlada

La matriz hessiana orlada es una variante de la matriz hessiana utilizada en problemas de optimización condicionada.

Dada la función $f\colon \mathbb{R}^n \to \mathbb{R}$ y las condiciónes o restricciones son: $g_1(\boldsymbol{x}) = c_1$, $g_2(\boldsymbol{x}) = c_2$, … , $g_m(\boldsymbol{x}) = c_m$, es decir, $\boldsymbol{g}(\boldsymbol{x}) = \boldsymbol{c}$, donde $\boldsymbol{g}\colon \mathbb{R}^n \to \mathbb{R}^m$, entonces la matriz hessiana orlada de la función Lagrangiana, $\Lambda(\boldsymbol{x}, \boldsymbol{\lambda}) = f(\boldsymbol{x}) + \boldsymbol{\lambda}^{\intercal}[\boldsymbol{g}(\boldsymbol{x}) - \boldsymbol{c}]$ asociada al problema de extremos condicionados es: $Hess\ \Lambda(\boldsymbol{v}) = J_{\boldsymbol{v}}\left(\nabla^{\intercal}\Lambda(\boldsymbol{v})\right)$, de tamaño $(n+m) \times (n+m)$, donde $\boldsymbol{v} = (\boldsymbol{x}, \boldsymbol{\lambda})$ es un vector de tamaño $1 \times (n+m)$

$$Hess\ \Lambda(\boldsymbol{x}, \boldsymbol{\lambda}) = \begin{bmatrix} J_{\mathbf{x}}(\nabla_{\mathbf{x}}^{\intercal}\Lambda) & \nabla^{\intercal}\mathbf{g}(\mathbf{x}) \\ \nabla\mathbf{g}(\mathbf{x}) & 0 \end{bmatrix}$$

$$Hess\ \Lambda(\boldsymbol{x}, \boldsymbol{\lambda}) = \begin{bmatrix} \dfrac{\partial^2 \Lambda}{\partial x_1^2} & \cdots & \dfrac{\partial^2 \Lambda}{\partial x_1 \partial x_n} & \dfrac{\partial g_1}{\partial x_1} & \cdots & \dfrac{\partial g_m}{\partial x_1} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ \dfrac{\partial^2 \Lambda}{\partial x_n \partial x_1} & \cdots & \dfrac{\partial^2 \Lambda}{\partial x_n^2} & \dfrac{\partial g_1}{\partial x_n} & \cdots & \dfrac{\partial g_m}{\partial x_n} \\ \dfrac{\partial g_1}{\partial x_1} & \cdots & \dfrac{\partial g_1}{\partial x_n} & 0 & \cdots & 0 \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ \dfrac{\partial g_m}{\partial x_1} & \cdots & \dfrac{\partial g_m}{\partial x_n} & 0 & \cdots & 0 \end{bmatrix}$$

Si hay *m* condiciones, el bloque de ceros en la esquina inferior derecha es de tamaño *m* × *m* y hay *m* filas y *m* columnas bordeando por abajo y por la derecha.

Ejemplo: Calcular la hessiana orlada de la siguiente función de Lagrange, $\Lambda(\boldsymbol{x}, \boldsymbol{\lambda})$, con función objetivo $f: \mathbb{R}^2 \to \mathbb{R}$ y restricciones $\boldsymbol{g}: \mathbb{R}^2 \to \mathbb{R}^2$.

Función de costo:

$$f(\boldsymbol{x}) = f(x_1, x_2)$$

Sujeto a las restricciones:

$$g_1(x_1, x_2) = c_1$$

$$g_2(x_1, x_2) = c_2$$

donde c_1 y c_2 son constantes reales

$$\Lambda(\boldsymbol{x}, \boldsymbol{\lambda}) = f(\boldsymbol{x}) + \lambda_1(g_1(\boldsymbol{x}) - c_1) + \lambda_2(g_2(\boldsymbol{x}) - c_2)$$

Derivadas de primer orden

$$\frac{\partial \Lambda(\boldsymbol{x}, \boldsymbol{\lambda})}{\partial \mathrm{x}_1} = \frac{\partial f(\boldsymbol{x})}{\partial \mathrm{x}_1} + \lambda_1 \frac{\partial g_1(\boldsymbol{x})}{\partial \mathrm{x}_1} + \lambda_2 \frac{\partial g_2(\boldsymbol{x})}{\partial \mathrm{x}_1}$$

$$\frac{\partial \Lambda(\boldsymbol{x}, \boldsymbol{\lambda})}{\partial \mathrm{x}_2} = \frac{\partial f(\boldsymbol{x})}{\partial \mathrm{x}_2} + \lambda_1 \frac{\partial g_1(\boldsymbol{x})}{\partial \mathrm{x}_2} + \lambda_2 \frac{\partial g_2(\boldsymbol{x})}{\partial \mathrm{x}_2}$$

$$\frac{\partial \Lambda(\boldsymbol{x}, \boldsymbol{\lambda})}{\partial \lambda_1} = g_1(\boldsymbol{x}) - c_1$$

$$\frac{\partial \Lambda(\boldsymbol{x}, \boldsymbol{\lambda})}{\partial \lambda_2} = g_2(\boldsymbol{x}) - c_2$$

$$\nabla \Lambda(\boldsymbol{x}, \boldsymbol{\lambda}) = \begin{bmatrix} \frac{\partial \Lambda(\boldsymbol{x}, \boldsymbol{\lambda})}{\partial \mathrm{x}_1} \\ \frac{\partial \Lambda(\boldsymbol{x}, \boldsymbol{\lambda})}{\partial \mathrm{x}_2} \\ \frac{\partial \Lambda(\boldsymbol{x}, \boldsymbol{\lambda})}{\partial \lambda_1} \\ \frac{\partial \Lambda(\boldsymbol{x}, \boldsymbol{\lambda})}{\partial \lambda_2} \end{bmatrix} = \begin{bmatrix} \frac{\partial f(\boldsymbol{x})}{\partial \mathrm{x}_1} + \lambda_1 \frac{\partial g_1(\boldsymbol{x})}{\partial \mathrm{x}_1} + \lambda_2 \frac{\partial g_2(\boldsymbol{x})}{\partial \mathrm{x}_1} \\ \frac{\partial f(\boldsymbol{x})}{\partial \mathrm{x}_2} + \lambda_1 \frac{\partial g_1(\boldsymbol{x})}{\partial \mathrm{x}_2} + \lambda_2 \frac{\partial g_2(\boldsymbol{x})}{\partial \mathrm{x}_2} \\ g_1(\boldsymbol{x}) - c_1 \\ g_2(\boldsymbol{x}) - c_2 \end{bmatrix}$$

Derivadas de segundo orden

$$\frac{\partial}{\partial x_1}\left(\frac{\partial \Lambda(\boldsymbol{x},\boldsymbol{\lambda})}{\partial \mathrm{x}_1}\right)=\frac{\partial^2 \Lambda(\boldsymbol{x},\boldsymbol{\lambda})}{\partial \mathrm{x}_1^2}=\frac{\partial}{\partial x_1}\left(\frac{\partial f(\boldsymbol{x})}{\partial \mathrm{x}_1}\right)+\frac{\partial}{\partial x_1}\left(\lambda_1\frac{\partial g_1(\boldsymbol{x})}{\partial \mathrm{x}_1}\right)+\frac{\partial}{\partial x_1}\left(\lambda_2\frac{\partial g_2(\boldsymbol{x})}{\partial \mathrm{x}_1}\right)$$

$$=\frac{\partial^2 f(\boldsymbol{x})}{\partial \mathrm{x}_1^2}+\lambda_1\frac{\partial^2 g_1(\boldsymbol{x})}{\partial \mathrm{x}_1^2}+\lambda_2\frac{\partial^2 g_2(\boldsymbol{x})}{\partial \mathrm{x}_1^2}$$

$$\frac{\partial}{\partial x_2}\left(\frac{\partial \Lambda(\boldsymbol{x},\boldsymbol{\lambda})}{\partial \mathrm{x}_1}\right)=\frac{\partial^2 \Lambda(\boldsymbol{x},\boldsymbol{\lambda})}{\partial x_2\,\partial \mathrm{x}_1}=\frac{\partial}{\partial x_2}\left(\frac{\partial f(\boldsymbol{x})}{\partial \mathrm{x}_1}\right)+\frac{\partial}{\partial x_2}\left(\lambda_1\frac{\partial g_1(\boldsymbol{x})}{\partial \mathrm{x}_1}\right)+\frac{\partial}{\partial x_2}\left(\lambda_2\frac{\partial g_2(\boldsymbol{x})}{\partial \mathrm{x}_1}\right)$$

$$=\frac{\partial^2 f(\boldsymbol{x})}{\partial x_2\,\partial \mathrm{x}_1}+\lambda_1\frac{\partial^2 g_1(\boldsymbol{x})}{\partial x_2\,\partial \mathrm{x}_1}+\lambda_2\frac{\partial^2 g_2(\boldsymbol{x})}{\partial x_2\,\partial \mathrm{x}_1}$$

$$\frac{\partial}{\partial \lambda_1}\left(\frac{\partial \Lambda(\boldsymbol{x},\boldsymbol{\lambda})}{\partial \mathrm{x}_1}\right)=\frac{\partial^2 \Lambda(\boldsymbol{x},\boldsymbol{\lambda})}{\partial \lambda_1\,\partial \mathrm{x}_1}=\frac{\partial}{\partial \lambda_1}\left(\frac{\partial f(\boldsymbol{x})}{\partial \mathrm{x}_1}\right)+\frac{\partial}{\partial \lambda_1}\left(\lambda_1\frac{\partial g_1(\boldsymbol{x})}{\partial \mathrm{x}_1}\right)+\frac{\partial}{\partial \lambda_1}\left(\lambda_2\frac{\partial g_2(\boldsymbol{x})}{\partial \mathrm{x}_1}\right)$$

$$=\frac{\partial g_1(\boldsymbol{x})}{\partial \mathrm{x}_1}$$

$$\frac{\partial}{\partial \lambda_2}\left(\frac{\partial \Lambda(\boldsymbol{x},\boldsymbol{\lambda})}{\partial \mathrm{x}_1}\right)=\frac{\partial^2 \Lambda(\boldsymbol{x},\boldsymbol{\lambda})}{\partial \lambda_2\,\partial \mathrm{x}_1}=\frac{\partial}{\partial \lambda_2}\left(\frac{\partial f(\boldsymbol{x})}{\partial \mathrm{x}_1}\right)+\frac{\partial}{\partial \lambda_2}\left(\lambda_1\frac{\partial g_1(\boldsymbol{x})}{\partial \mathrm{x}_1}\right)+\frac{\partial}{\partial \lambda_2}\left(\lambda_2\frac{\partial g_2(\boldsymbol{x})}{\partial \mathrm{x}_1}\right)$$

$$=\frac{\partial g_2(\boldsymbol{x})}{\partial \mathrm{x}_1}$$

$$\frac{\partial}{\partial x_1}\left(\frac{\partial \Lambda(\boldsymbol{x},\boldsymbol{\lambda})}{\partial \mathrm{x}_2}\right)=\frac{\partial^2 \Lambda(\boldsymbol{x},\boldsymbol{\lambda})}{\partial x_1\,\partial \mathrm{x}_2}=\frac{\partial}{\partial x_1}\left(\frac{\partial f(\boldsymbol{x})}{\partial \mathrm{x}_2}\right)+\frac{\partial}{\partial x_1}\left(\lambda_1\frac{\partial g_1(\boldsymbol{x})}{\partial \mathrm{x}_2}\right)+\frac{\partial}{\partial x_1}\left(\lambda_2\frac{\partial g_2(\boldsymbol{x})}{\partial \mathrm{x}_2}\right)$$

$$=\frac{\partial^2 f(\boldsymbol{x})}{\partial x_1\,\partial \mathrm{x}_2}+\lambda_1\frac{\partial^2 g_1(\boldsymbol{x})}{\partial x_1\,\partial \mathrm{x}_2}+\lambda_2\frac{\partial^2 g_2(\boldsymbol{x})}{\partial x_1\,\partial \mathrm{x}_2}$$

$$\frac{\partial}{\partial x_2}\left(\frac{\partial \Lambda(\boldsymbol{x},\boldsymbol{\lambda})}{\partial \mathrm{x}_2}\right)=\frac{\partial^2 \Lambda(\boldsymbol{x},\boldsymbol{\lambda})}{\partial \mathrm{x}_2^2}=\frac{\partial}{\partial x_2}\frac{\partial f(\boldsymbol{x})}{\partial \mathrm{x}_2}+\frac{\partial}{\partial x_2}\left(\lambda_1\frac{\partial g_1(\boldsymbol{x})}{\partial \mathrm{x}_2}\right)+\frac{\partial}{\partial x_2}\left(\lambda_2\frac{\partial g_2(\boldsymbol{x})}{\partial \mathrm{x}_2}\right)$$

$$=\frac{\partial^2 f(\boldsymbol{x})}{\partial \mathrm{x}_2^2}+\lambda_1\frac{\partial^2 g_1(\boldsymbol{x})}{\partial \mathrm{x}_2^2}+\lambda_2\frac{\partial^2 g_2(\boldsymbol{x})}{\partial \mathrm{x}_2^2}$$

$$\frac{\partial}{\partial \lambda_1}\left(\frac{\partial \Lambda(\boldsymbol{x},\boldsymbol{\lambda})}{\partial \mathrm{x}_2}\right)=\frac{\partial^2 \Lambda(\boldsymbol{x},\boldsymbol{\lambda})}{\partial \lambda_1\,\partial \mathrm{x}_2}=\frac{\partial}{\partial \lambda_1}\left(\frac{\partial f(\boldsymbol{x})}{\partial \mathrm{x}_2}\right)+\frac{\partial}{\partial \lambda_1}\left(\lambda_1\frac{\partial g_1(\boldsymbol{x})}{\partial \mathrm{x}_2}\right)+\frac{\partial}{\partial \lambda_1}\left(\lambda_2\frac{\partial g_2(\boldsymbol{x})}{\partial \mathrm{x}_2}\right)$$

$$= \frac{\partial g_1(\boldsymbol{x})}{\partial \mathrm{x}_2}$$

$$\frac{\partial}{\partial \lambda_2}\left(\frac{\partial \Lambda(\boldsymbol{x},\boldsymbol{\lambda})}{\partial \mathrm{x}_2}\right) = \frac{\partial^2 \Lambda(\boldsymbol{x},\boldsymbol{\lambda})}{\partial \lambda_2\, \partial \mathrm{x}_2} = \frac{\partial}{\partial \lambda_2}\left(\frac{\partial f(\boldsymbol{x})}{\partial \mathrm{x}_2}\right) + \frac{\partial}{\partial \lambda_2}\left(\lambda_1 \frac{\partial g_1(\boldsymbol{x})}{\partial \mathrm{x}_2}\right) + \frac{\partial}{\partial \lambda_2}\left(\lambda_2 \frac{\partial g_2(\boldsymbol{x})}{\partial \mathrm{x}_2}\right)$$

$$= \frac{\partial g_2(\boldsymbol{x})}{\partial \mathrm{x}_2}$$

$$\frac{\partial}{\partial x_1}\left(\frac{\partial \Lambda(\boldsymbol{x},\boldsymbol{\lambda})}{\partial \lambda_1}\right) = \frac{\partial^2 \Lambda(\boldsymbol{x},\boldsymbol{\lambda})}{\partial \mathrm{x}_1 \partial \lambda_1} = \frac{\partial}{\partial x_1}(g_1(\boldsymbol{x}) - c_1) = \frac{\partial g_1(\boldsymbol{x})}{\partial x_1}$$

$$\frac{\partial}{\partial x_2}\left(\frac{\partial \Lambda(\boldsymbol{x},\boldsymbol{\lambda})}{\partial \lambda_1}\right) = \frac{\partial^2 \Lambda(\boldsymbol{x},\boldsymbol{\lambda})}{\partial \mathrm{x}_2 \partial \lambda_1} = \frac{\partial}{\partial x_2}(g_1(\boldsymbol{x}) - c_1) = \frac{\partial g_1(\boldsymbol{x})}{\partial x_2}$$

$$\frac{\partial}{\partial \lambda_1}\left(\frac{\partial \Lambda(\boldsymbol{x},\boldsymbol{\lambda})}{\partial \lambda_1}\right) = \frac{\partial^2 \Lambda(\boldsymbol{x},\boldsymbol{\lambda})}{\partial \lambda_1^2} = \frac{\partial}{\partial \lambda_1}(g_1(\boldsymbol{x}) - c_1) = 0$$

$$\frac{\partial}{\partial \lambda_2}\left(\frac{\partial \Lambda(\boldsymbol{x},\boldsymbol{\lambda})}{\partial \lambda_1}\right) = \frac{\partial^2 \Lambda(\boldsymbol{x},\boldsymbol{\lambda})}{\partial \lambda_2\, \partial \lambda_1} = \frac{\partial}{\partial \lambda_2}(g_1(\boldsymbol{x}) - c_1) = 0$$

$$\frac{\partial}{\partial x_1}\left(\frac{\partial \Lambda(\boldsymbol{x},\boldsymbol{\lambda})}{\partial \lambda_2}\right) = \frac{\partial^2 \Lambda(\boldsymbol{x},\boldsymbol{\lambda})}{\partial x_1\, \partial \lambda_2} = \frac{\partial}{\partial x_1}(g_2(\boldsymbol{x}) - c_2) = \frac{\partial g_2(\boldsymbol{x})}{\partial x_1}$$

$$\frac{\partial}{\partial x_2}\left(\frac{\partial \Lambda(\boldsymbol{x},\boldsymbol{\lambda})}{\partial \lambda_2}\right) = \frac{\partial^2 \Lambda(\boldsymbol{x},\boldsymbol{\lambda})}{\partial x_2\, \partial \lambda_2} = \frac{\partial}{\partial x_2}(g_2(\boldsymbol{x}) - c_2) = \frac{\partial g_2(\boldsymbol{x})}{\partial x_2}$$

$$\frac{\partial}{\partial \lambda_1}\left(\frac{\partial \Lambda(\boldsymbol{x},\boldsymbol{\lambda})}{\partial \lambda_2}\right) = \frac{\partial^2 \Lambda(\boldsymbol{x},\boldsymbol{\lambda})}{\partial \lambda_1\, \partial \lambda_2} = \frac{\partial}{\partial \lambda_1}(g_2(\boldsymbol{x}) - c_2) = 0$$

$$\frac{\partial}{\partial \lambda_2}\left(\frac{\partial \Lambda(\boldsymbol{x},\boldsymbol{\lambda})}{\partial \lambda_2}\right) = \frac{\partial \Lambda(\boldsymbol{x},\boldsymbol{\lambda})}{\partial \lambda_2^2} = \frac{\partial}{\partial \lambda_2}(g_2(\boldsymbol{x}) - c_2) = 0$$

$$Hess\ \Lambda(\boldsymbol{x},\boldsymbol{\lambda}) = \begin{bmatrix} \frac{\partial^2 \Lambda(\boldsymbol{x},\boldsymbol{\lambda})}{\partial \mathrm{x}_1^2} & \frac{\partial^2 \Lambda(\boldsymbol{x},\boldsymbol{\lambda})}{\partial x_2\, \partial \mathrm{x}_1} & \frac{\partial^2 \Lambda(\boldsymbol{x},\boldsymbol{\lambda})}{\partial \lambda_1\, \partial \mathrm{x}_1} & \frac{\partial^2 \Lambda(\boldsymbol{x},\boldsymbol{\lambda})}{\partial \lambda_2\, \partial \mathrm{x}_1} \\ \frac{\partial^2 \Lambda(\boldsymbol{x},\boldsymbol{\lambda})}{\partial x_1\, \partial \mathrm{x}_2} & \frac{\partial^2 \Lambda(\boldsymbol{x},\boldsymbol{\lambda})}{\partial \mathrm{x}_2^2} & \frac{\partial^2 \Lambda(\boldsymbol{x},\boldsymbol{\lambda})}{\partial \lambda_1\, \partial \mathrm{x}_2} & \frac{\partial^2 \Lambda(\boldsymbol{x},\boldsymbol{\lambda})}{\partial \lambda_2\, \partial \mathrm{x}_2} \\ \frac{\partial^2 \Lambda(\boldsymbol{x},\boldsymbol{\lambda})}{\partial \mathrm{x}_1 \partial \lambda_1} & \frac{\partial^2 \Lambda(\boldsymbol{x},\boldsymbol{\lambda})}{\partial \mathrm{x}_2 \partial \lambda_1} & \frac{\partial^2 \Lambda(\boldsymbol{x},\boldsymbol{\lambda})}{\partial \lambda_1^2} & \frac{\partial^2 \Lambda(\boldsymbol{x},\boldsymbol{\lambda})}{\partial \lambda_2\, \partial \lambda_1} \\ \frac{\partial^2 \Lambda(\boldsymbol{x},\boldsymbol{\lambda})}{\partial x_1\, \partial \lambda_2} & \frac{\partial^2 \Lambda(\boldsymbol{x},\boldsymbol{\lambda})}{\partial x_2\, \partial \lambda_2} & \frac{\partial^2 \Lambda(\boldsymbol{x},\boldsymbol{\lambda})}{\partial \lambda_1\, \partial \lambda_2} & \frac{\partial \Lambda(\boldsymbol{x},\boldsymbol{\lambda})}{\partial \lambda_2^2} \end{bmatrix}$$

$$Hess\ \Lambda(\boldsymbol{x},\boldsymbol{\lambda}) = \begin{bmatrix} \frac{\partial^2 f(\boldsymbol{x})}{\partial \mathrm{x}_1^2} + \lambda_1 \frac{\partial^2 g_1(\boldsymbol{x})}{\partial \mathrm{x}_1^2} + \lambda_2 \frac{\partial^2 g_2(\boldsymbol{x})}{\partial \mathrm{x}_1^2} & \frac{\partial^2 f(\boldsymbol{x})}{\partial x_2\, \partial \mathrm{x}_1} + \lambda_1 \frac{\partial^2 g_1(\boldsymbol{x})}{\partial x_2\, \partial \mathrm{x}_1} + \lambda_2 \frac{\partial^2 g_2(\boldsymbol{x})}{\partial x_2\, \partial \mathrm{x}_1} & \frac{\partial g_1(\boldsymbol{x})}{\partial \mathrm{x}_1} & \frac{\partial g_2(\boldsymbol{x})}{\partial \mathrm{x}_1} \\ \frac{\partial^2 f(\boldsymbol{x})}{\partial x_1\, \partial \mathrm{x}_2} + \lambda_1 \frac{\partial^2 g_1(\boldsymbol{x})}{\partial x_1\, \partial \mathrm{x}_2} + \lambda_2 \frac{\partial^2 g_2(\boldsymbol{x})}{\partial x_1\, \partial \mathrm{x}_2} & \frac{\partial^2 f(\boldsymbol{x})}{\partial \mathrm{x}_2^2} + \lambda_1 \frac{\partial^2 g_1(\boldsymbol{x})}{\partial \mathrm{x}_2^2} + \lambda_2 \frac{\partial^2 g_2(\boldsymbol{x})}{\partial \mathrm{x}_2^2} & \frac{\partial g_1(\boldsymbol{x})}{\partial \mathrm{x}_2} & \frac{\partial g_2(\boldsymbol{x})}{\partial \mathrm{x}_2} \\ \frac{\partial g_1(\boldsymbol{x})}{\partial x_1} & \frac{\partial g_1(\boldsymbol{x})}{\partial x_2} & 0 & 0 \\ \frac{\partial g_2(\boldsymbol{x})}{\partial x_1} & \frac{\partial g_2(\boldsymbol{x})}{\partial x_2} & 0 & 0 \end{bmatrix}$$

Gradient descent optimization algorithms

1. Adadelta

trainAdadelta(*params*, *lr=1.0*, *rho=0.9*, *eps=1e-06*, *weight_decay=0*)

Implements Adadelta algorithm.

$$\textbf{input} : \gamma \text{ (lr)}, \ \theta_0 \text{ (params)}, \ f(\theta) \text{ (objective)}, \ \rho \text{ (decay)}, \ \lambda \text{ (weight decay)}$$
$$\textbf{initialize} : v_0 \leftarrow 0 \ \text{ (square avg)}, \ u_0 \leftarrow 0 \ \text{ (accumulate variables)}$$

$$\textbf{for } t = 1 \textbf{ to } \ldots \textbf{ do}$$
$$g_t \leftarrow \nabla_\theta f_t(\theta_{t-1})$$
$$if \ \lambda \neq 0$$
$$g_t \leftarrow g_t + \lambda \theta_{t-1}$$
$$v_t \leftarrow v_{t-1}\rho + g_t^2(1-\rho)$$
$$\Delta x_t \leftarrow \frac{\sqrt{u_{t-1} + \epsilon}}{\sqrt{v_t + \epsilon}} g_t$$
$$u_t \leftarrow u_{t-1}\rho + \Delta x_t^2(1-\rho)$$
$$\theta_t \leftarrow \theta_{t-1} - \gamma \Delta x_t$$

$$\textbf{return } \theta_t$$

For further details regarding the algorithm we refer to [ADADELTA: An Adaptive Learning Rate Method].

2. Adagrad

trainAdagrad(*params, lr=0.01, lr_decay=0, weight_decay=0, initial_accumulator_value=0, eps=1e-10*)

Implements Adagrad algorithm.

$$\textbf{input} : \gamma \text{ (lr)}, \ \theta_0 \text{ (params)}, \ f(\theta) \text{ (objective)}, \ \lambda \text{ (weight decay)},$$
$$\tau \text{ (initial accumulator value)}, \ \eta \text{ (lr decay)}$$
$$\textbf{initialize} : state\_sum_0 \leftarrow 0$$

$$\textbf{for } t = 1 \textbf{ to } \ldots \textbf{ do}$$
$$g_t \leftarrow \nabla_\theta f_t(\theta_{t-1})$$
$$\tilde{\gamma} \leftarrow \gamma / (1 + (t-1)\eta)$$
$$\textbf{if } \lambda \neq 0$$
$$g_t \leftarrow g_t + \lambda \theta_{t-1}$$
$$state\_sum_t \leftarrow state\_sum_{t-1} + g_t^2$$
$$\theta_t \leftarrow \theta_{t-1} - \tilde{\gamma} \frac{g_t}{\sqrt{state\_sum_t} + \epsilon}$$

$$\textbf{return } \theta_t$$

For further details regarding the algorithm we refer to Adaptive Subgradient Methods for Online Learning and Stochastic Optimization.

3. Adam

trainAdam(*params*, *lr=0.001*, *betas=(0.9, 0.999)*, *eps=1e-08*, *weight_decay=0*, *amsgrad=False*, *maximize=False*)

Implements Adam algorithm.

$$
\begin{aligned}
&\textbf{input}: \gamma \text{ (lr)}, \beta_1, \beta_2 \text{ (betas)}, \theta_0 \text{ (params)}, f(\theta) \text{ (objective)} \\
&\qquad \lambda \text{ (weight decay)}, \ \textit{amsgrad}, \ \textit{maximize} \\
&\textbf{initialize}: m_0 \leftarrow 0 \text{ (first moment)}, v_0 \leftarrow 0 \text{ (second moment)}, \ \widehat{v_0}^{max} \leftarrow 0 \\
&\textbf{for } t = 1 \textbf{ to } \ldots \textbf{ do} \\
&\quad \textbf{if } \textit{maximize}: \\
&\qquad g_t \leftarrow -\nabla_\theta f_t(\theta_{t-1}) \\
&\quad \textbf{else} \\
&\qquad g_t \leftarrow \nabla_\theta f_t(\theta_{t-1}) \\
&\quad \textbf{if } \lambda \neq 0 \\
&\qquad g_t \leftarrow g_t + \lambda \theta_{t-1} \\
&\quad m_t \leftarrow \beta_1 m_{t-1} + (1 - \beta_1) g_t \\
&\quad v_t \leftarrow \beta_2 v_{t-1} + (1 - \beta_2) g_t^2 \\
&\quad \widehat{m_t} \leftarrow m_t / \big(1 - \beta_1^t\big) \\
&\quad \widehat{v_t} \leftarrow v_t / \big(1 - \beta_2^t\big) \\
&\quad \textbf{if } \textit{amsgrad} \\
&\qquad \widehat{v_t}^{max} \leftarrow \max(\widehat{v_t}^{max}, \widehat{v_t}) \\
&\qquad \theta_t \leftarrow \theta_{t-1} - \gamma \widehat{m_t} / \big(\sqrt{\widehat{v_t}^{max}} + \epsilon\big) \\
&\quad \textbf{else} \\
&\qquad \theta_t \leftarrow \theta_{t-1} - \gamma \widehat{m_t} / \big(\sqrt{\widehat{v_t}} + \epsilon\big) \\
&\textbf{return } \theta_t
\end{aligned}
$$

For further details regarding the algorithm we refer to Adam: A Method for Stochastic Optimization.

4. AdamW

trainAdamW(*params*, *lr=0.001*, *betas=(0.9, 0.999)*, *eps=1e-08*, *weight_decay=0.01*, *amsgrad=False*, *maximize=False*)

Implements AdamW algorithm.

$$\textbf{input} : \gamma(\text{lr}),\ \beta_1, \beta_2(\text{betas}),\ \theta_0(\text{params}),\ f(\theta)(\text{objective}),\ \epsilon\ (\text{epsilon})$$
$$\lambda(\text{weight decay}),\ \textit{amsgrad},\ \textit{maximize}$$
$$\textbf{initialize} : m_0 \leftarrow 0\ (\text{first moment}), v_0 \leftarrow 0\ (\text{ second moment}),\ \widehat{v_0}^{max} \leftarrow 0$$

$$\textbf{for}\ t = 1\ \textbf{to}\ \ldots\ \textbf{do}$$
$$\quad \textbf{if}\ \textit{maximize}:$$
$$\quad\quad g_t \leftarrow -\nabla_\theta f_t(\theta_{t-1})$$
$$\quad \textbf{else}$$
$$\quad\quad g_t \leftarrow \nabla_\theta f_t(\theta_{t-1})$$
$$\quad \theta_t \leftarrow \theta_{t-1} - \gamma\lambda\theta_{t-1}$$
$$\quad m_t \leftarrow \beta_1 m_{t-1} + (1-\beta_1) g_t$$
$$\quad v_t \leftarrow \beta_2 v_{t-1} + (1-\beta_2) g_t^2$$
$$\quad \widehat{m_t} \leftarrow m_t / \big(1-\beta_1^t\big)$$
$$\quad \widehat{v_t} \leftarrow v_t / \big(1-\beta_2^t\big)$$
$$\quad \textbf{if}\ \textit{amsgrad}$$
$$\quad\quad \widehat{v_t}^{max} \leftarrow \max(\widehat{v_t}^{max}, \widehat{v_t})$$
$$\quad\quad \theta_t \leftarrow \theta_t - \gamma \widehat{m_t} / \big(\sqrt{\widehat{v_t}^{max}} + \epsilon\big)$$
$$\quad \textbf{else}$$
$$\quad\quad \theta_t \leftarrow \theta_t - \gamma \widehat{m_t} / \big(\sqrt{\widehat{v_t}} + \epsilon\big)$$

$$\textbf{return}\ \theta_t$$

For further details regarding the algorithm we refer to Decoupled Weight Decay Regularization.

5. Adamax

trainAdamax(*params*, *lr=0.002*, *betas=(0.9, 0.999)*, *eps=1e-08*, *weight_decay=0*)

Implements Adamax algorithm (a variant of Adam based on infinity norm).

$$\textbf{input} : \gamma \text{ (lr)}, \beta_1, \beta_2 \text{ (betas)}, \theta_0 \text{ (params)}, f(\theta) \text{ (objective)}, \lambda \text{ (weight decay)}$$
$$\epsilon \text{ (epsilon)}$$
$$\textbf{initialize} : m_0 \leftarrow 0 \text{ (first moment)}, u_0 \leftarrow 0 \text{ (infinity norm)}$$

$$\textbf{for } t = 1 \textbf{ to } \ldots \textbf{ do}$$
$$g_t \leftarrow \nabla_\theta f_t(\theta_{t-1})$$
$$if\ \lambda \neq 0$$
$$g_t \leftarrow g_t + \lambda \theta_{t-1}$$
$$m_t \leftarrow \beta_1 m_{t-1} + (1 - \beta_1) g_t$$
$$u_t \leftarrow \max(\beta_2 u_{t-1}, |g_t| + \epsilon)$$
$$\theta_t \leftarrow \theta_{t-1} - \frac{\gamma m_t}{(1 - \beta_1^t) u_t}$$

$$\textbf{return } \theta_t$$

For further details regarding the algorithm we refer to Adam: A Method for Stochastic Optimization.

6. Lion

The Lion optimizer is a stochastic-gradient-descent method that uses the sign operator to control the magnitude of the update, unlike other adaptive optimizers such as Adam that rely on second order moments. This make Lion more memory-efficient as it only keeps track of the momentum. According to the authors (see reference), its performance gain over Adam grows with the batch size. Because the update of Lion is produced through the sign operation, resulting in a larger norm, a suitable learning rate for Lion is typically 3-10x smaller than that for AdamW. The weight decay for Lion should be in turn 3-10x larger than that for AdamW to maintain a similar strength (lr * wd).

Algorithm 2 Lion Optimizer (ours)

given $\beta_1, \beta_2, \lambda, \eta, f$
initialize $\theta_0, m_0 \leftarrow 0$
while θ_t not converged **do**
  $g_t \leftarrow \nabla_\theta f(\theta_{t-1})$
  update model parameters
  $c_t \leftarrow \beta_1 m_{t-1} + (1-\beta_1) g_t$
  $\theta_t \leftarrow \theta_{t-1} - \eta_t(\text{sign}(c_t) + \lambda\theta_{t-1})$
  update EMA of g_t
  $m_t \leftarrow \beta_2 m_{t-1} + (1-\beta_2) g_t$
end while
return θ_t

7. NAdam

trainNAdam(*params*, *lr=0.002*, *betas=(0.9, 0.999)*, *eps=1e-08*, *weight_decay=0*, *momentum_decay=0.004*)

Implements NAdam algorithm.

$$
\begin{aligned}
&\rule{110mm}{0.4pt} \\
&\textbf{input} : \gamma_t \text{ (lr)}, \ \beta_1, \beta_2 \text{ (betas)}, \ \theta_0 \text{ (params)}, \ f(\theta) \text{ (objective)} \\
&\qquad \lambda \text{ (weight decay)}, \ \psi \text{ (momentum decay)} \\
&\textbf{initialize} : m_0 \leftarrow 0 \text{ (first moment)}, v_0 \leftarrow 0 \text{ (second moment)} \\
&\rule{110mm}{0.4pt} \\
&\textbf{for } t = 1 \textbf{ to } \ldots \textbf{ do} \\
&\quad g_t \leftarrow \nabla_\theta f_t(\theta_{t-1}) \\
&\quad if \ \lambda \neq 0 \\
&\quad\quad g_t \leftarrow g_t + \lambda \theta_{t-1} \\
&\quad \mu_t \leftarrow \beta_1 \big(1 - \frac{1}{2} 0.96^{t\psi}\big) \\
&\quad \mu_{t+1} \leftarrow \beta_1 \big(1 - \frac{1}{2} 0.96^{(t+1)\psi}\big) \\
&\quad m_t \leftarrow \beta_1 m_{t-1} + (1 - \beta_1) g_t \\
&\quad v_t \leftarrow \beta_2 v_{t-1} + (1-\beta_2) g_t^2 \\
&\quad \widehat{m_t} \leftarrow \mu_{t+1} m_t / (1 - \prod_{i=1}^{t+1} \mu_i) \\
&\quad\quad + (1 - \mu_t) g_t / (1 - \prod_{i=1}^{t} \mu_i) \\
&\quad \widehat{v_t} \leftarrow v_t / \big(1 - \beta_2^t\big) \\
&\quad \theta_t \leftarrow \theta_{t-1} - \gamma \widehat{m_t} / \big(\sqrt{\widehat{v_t}} + \epsilon\big) \\
&\rule{110mm}{0.4pt} \\
&\textbf{return } \theta_t
\end{aligned}
$$

For further details regarding the algorithm we refer to [Incorporating Nesterov Momentum into Adam](#).

8. RAdam

trainRAdam(*params*, *lr=0.001*, *betas=(0.9, 0.999)*, *eps=1e-08*, *weight_decay=0*)

Implements RAdam algorithm.

---

input : γ (lr), β_1, β_2 (betas), θ_0 (params), $f(\theta)$ (objective), λ (weightdecay), ϵ (epsilon)

initialize : $m_0 \leftarrow 0$ (first moment), $v_0 \leftarrow 0$ (second moment), $\rho_\infty \leftarrow 2/(1-\beta_2) - 1$

---

for $t = 1$ **to** $\ldots$ **do**

$g_t \leftarrow \nabla_\theta f_t(\theta_{t-1})$

if $\lambda \neq 0$

$g_t \leftarrow g_t + \lambda \theta_{t-1}$

$m_t \leftarrow \beta_1 m_{t-1} + (1-\beta_1) g_t$

$v_t \leftarrow \beta_2 v_{t-1} + (1-\beta_2) g_t^2$

$\widehat{m_t} \leftarrow m_t / \left(1-\beta_1^t\right)$

$\rho_t \leftarrow \rho_\infty - 2t\beta_2^t / \left(1-\beta_2^t\right)$

if $\rho_t > 5$

$l_t \leftarrow \frac{\sqrt{(1-\beta_2^t)}}{\sqrt{v_t} + \epsilon}$

$r_t \leftarrow \sqrt{\frac{(\rho_t - 4)(\rho_t - 2)\rho_\infty}{(\rho_\infty - 4)(\rho_\infty - 2)\rho_t}}$

$\theta_t \leftarrow \theta_{t-1} - \gamma \widehat{m_t} r_t l_t$

else

$\theta_t \leftarrow \theta_{t-1} - \gamma \widehat{m_t}$

---

return θ_t

For further details regarding the algorithm we refer to On the variance of the adaptive learning rate and beyond.

9. RMSprop

trainRMSprop(*params*, *lr=0.01*, *alpha=0.99*, *eps=1e-08*, *weight_decay=0*, *momentum=0*, *centered=False*)

Implements RMSprop algorithm.

$$
\begin{aligned}
&\rule{110mm}{0.4pt} \\
&\textbf{input} : \alpha \text{ (alpha)}, \gamma \text{ (lr)}, \theta_0 \text{ (params)}, f(\theta) \text{ (objective)} \\
&\hspace{13mm} \lambda \text{ (weight decay)}, \mu \text{ (momentum)}, \textit{centered} \\
&\textbf{initialize} : v_0 \leftarrow 0 \text{ (square average)}, \mathbf{b}_0 \leftarrow 0 \text{ (buffer)}, g_0^{ave} \leftarrow 0 \\
&\rule{110mm}{0.4pt} \\
&\textbf{for } t=1 \textbf{ to } \ldots \textbf{ do} \\
&\quad g_t \leftarrow \nabla_\theta f_t(\theta_{t-1}) \\
&\quad if \ \lambda \neq 0 \\
&\quad\quad g_t \leftarrow g_t + \lambda \theta_{t-1} \\
&\quad v_t \leftarrow \alpha v_{t-1} + (1-\alpha) g_t^2 \\
&\quad \tilde{v_t} \leftarrow v_t \\
&\quad if \ centered \\
&\quad\quad g_t^{ave} \leftarrow g_{t-1}^{ave} \alpha + (1-\alpha) g_t \\
&\quad\quad \tilde{v_t} \leftarrow \tilde{v_t} - \big(g_t^{ave}\big)^2 \\
&\quad if \ \mu > 0 \\
&\quad\quad \mathbf{b}_t \leftarrow \mu \mathbf{b}_{t-1} + g_t/\big(\sqrt{\tilde{v_t}} + \epsilon\big) \\
&\quad\quad \theta_t \leftarrow \theta_{t-1} - \gamma \mathbf{b}_t \\
&\quad else \\
&\quad\quad \theta_t \leftarrow \theta_{t-1} - \gamma g_t/\big(\sqrt{\tilde{v_t}} + \epsilon\big) \\
&\rule{110mm}{0.4pt} \\
&\textbf{return } \theta_t
\end{aligned}
$$

For further details regarding the algorithm we refer to lecture notes by G. Hinton. and centered version Generating Sequences With Recurrent Neural Networks.

10. SMORMS3

SMORMS3 (Squared Mean Over Root Mean Squared Cubed) is an optimization algorithm used in machine learning and deep learning to optimize the training of neural networks. It is a variant of the RMSProp algorithm and was introduced in 2017 by Daniel Fortunato, et al.

SMORMS3 modifies the way the moving average of the squared gradients is calculated in RMSProp. Instead of taking the simple average of the squared gradients, SMORMS3 takes the cube root of the moving average of the cube of the squared gradients. This modification helps to normalize the scale of the moving average, which can prevent the learning rate from decreasing too quickly.

Overall, SMORMS3 is a powerful optimization algorithm that can help accelerate the training of deep neural networks and improve their performance, particularly in situations where the gradients have a high variance.

Algorithm 4 Algorithm of SMORMS3

Require: lr: Learning rate
Require: θ_0: Initial parameter
Require: $f(\theta)$: Loss function with parameter θ
Require: $g(\theta)$: Gradient of loss function with parameter θ
Require: ε: Small constant
$t \leftarrow 0$
$m_0 \leftarrow 0$
$v_0 \leftarrow 0$
$s_0 \leftarrow 1$
while Exit condition is not satisfied **do**
  $t \leftarrow t + 1$
  $g(\theta_t) \leftarrow \nabla_{\theta_t} f(\theta_t)$
  $s_t \leftarrow 1 + (1 - x_{t-1}) \cdot s_{t-1}$
  $\rho_t \leftarrow \frac{1}{s_t+1}$
  $m_t \leftarrow (1 - \rho_t) \cdot m_{t-1} + \rho_t \cdot g(\theta_t)$
  $v_t \leftarrow (1 - \rho_t) \cdot v_{t-1} + \rho_t \cdot g^2(\theta_t)$
  $x_t \leftarrow \frac{m_t^2}{v_t+\varepsilon}$
  $\theta_t \leftarrow \theta_{t-1} - \frac{\min\{lr, x_t\}}{\sqrt{v_t}+\varepsilon} \cdot g(\theta_t)$
end while

11. Rprop

trainRprop(*params*, *lr=0.01*, *etas=(0.5, 1.2)*, *step_sizes=(1e-06, 50)*)

Implements the resilient backpropagation algorithm.

---

input : $\theta_0 \in \mathbf{R}^d$ (params), $f(\theta)$ (objective),
$\eta_{+/-}$ (etaplus, etaminus), $\Gamma_{max/min}$ (step sizes)
initialize : $g^0_{prev} \leftarrow 0$, $\eta_0 \leftarrow \text{lr}$ (learning rate)

---

for $t = 1$ **to** $\ldots$ **do**
  $g_t \leftarrow \nabla_\theta f_t(\theta_{t-1})$
  for $i = 0, 1, \ldots, d-1$ **do**
    if $g^i_{prev} g^i_t > 0$
      $\eta^i_t \leftarrow \min(\eta^i_{t-1}\eta_+, \Gamma_{max})$
    else if $g^i_{prev} g^i_t < 0$
      $\eta^i_t \leftarrow \max(\eta^i_{t-1}\eta_-, \Gamma_{min})$
      $g^i_t \leftarrow 0$
    else
      $\eta^i_t \leftarrow \eta^i_{t-1}$
  $\theta_t \leftarrow \theta_{t-1} - \eta_t \text{sign}(g_t)$
  $g_{prev} \leftarrow g_t$

---

return θ_t

For further details regarding the algorithm we refer to the paper A Direct Adaptive Method for Faster Backpropagation Learning: The RPROP Algorithm.

12. SGD

trainSGD(*params, lr=<required parameter>, momentum=0, dampening=0, weight_decay=0, nesterov=False, maximize=False*)

Implements stochastic gradient descent (optionally with momentum).

---

input : γ (lr), θ_0 (params), $f(\theta)$ (objective), λ (weight decay),
μ (momentum), τ (dampening), *nesterov*, *maximize*

---

for $t = 1$ **to** $\ldots$ **do**
  $g_t \leftarrow \nabla_\theta f_t(\theta_{t-1})$
  if $\lambda \neq 0$
    $g_t \leftarrow g_t + \lambda\theta_{t-1}$
  if $\mu \neq 0$
    if $t > 1$
      $\mathbf{b}_t \leftarrow \mu\mathbf{b}_{t-1} + (1-\tau)g_t$
    else
      $\mathbf{b}_t \leftarrow g_t$
    if *nesterov*
      $g_t \leftarrow g_t + \mu\mathbf{b}_t$
    else
      $g_t \leftarrow \mathbf{b}_t$
  if *maximize*
    $\theta_t \leftarrow \theta_{t-1} + \gamma g_t$
  else
    $\theta_t \leftarrow \theta_{t-1} - \gamma g_t$

---

return θ_t

Nesterov momentum is based on the formula from On the importance of initialization and momentum in deep learning.

13. SGD with U-Clip

SGD with U-Clip, a novel method for achieving on-average unbiased stochastic gradient clipping.

Algorithm 1 SGD with U-Clip

Input: $f(\theta, \cdot)$ (loss function), S (dataset), $\theta_1 \in \mathbb{R}^p$ (initial parameters), $\eta \in \mathbb{R}_{>0}$ (learning rate), $\gamma \in \mathbb{R}_{>0}$ (clipping region), $T \in \mathbb{N}$ (number of timesteps).
Return: θ_{T+1} (optimized parameters).
$\Delta_1 \leftarrow 0$
for $t = 1, \ldots, T$ **do**
  Sample $Z_t \sim \text{Unif}\, S$
  $g_t \leftarrow \nabla_\theta f(\theta_t, Z_t)$
  $u_t \leftarrow \texttt{clip}(g_t + \Delta_t, \gamma)$
  $\Delta_{t+1} \leftarrow g_t + \Delta_t - u_t$
  $\theta_{t+1} \leftarrow \theta_t - \eta u_t$
end for

Proposition 2.4 (Carry bound: component-wise clipping)**.** *In this result, all variables are real scalars. Consider the following update rules with* $\Delta_1 = 0$

$$u_t = \texttt{clip}(g_t + \Delta_t, \gamma_t) \quad \textit{and} \quad \Delta_{t+1} = g_t + \Delta_t - u_t$$

where $\texttt{clip}(x, \gamma) = \min\{\max\{g_t + \Delta_t, -\gamma\}, \gamma\}$ *projects* x *onto the interval* $[-\gamma, \gamma]$.

14. Gradient Descent

trainGD(*params, lr=<required parameter>*)

implements gradient descent

inputs: $\gamma(lr = 0.001$:learning rate), θ_0(params), $f(\theta)$ (objective)

for $t = 1$ ***to*** ... ***do***
  $g_t \leftarrow \nabla f(\theta_{t-1})$
  $\theta_t \leftarrow \theta_{t-1} - \gamma g_t$
return θ_t

15. Gradient Descent with momentum

trainGDM(*params*, *lr=<required parameter>, momentum=<required parameter>*)

implements gradient descent with momentum

inputs: γ(*lr = 0.001*: learning rate), θ_0(params), $f(\theta)$ (objective), μ (momentum = 0.90)
$g_0 \leftarrow \nabla f(\theta_0)$
$\Delta\theta_0 \leftarrow \gamma\ g_0$
for $t = 1$ ***to*** ... ***do***
    $\Delta\theta_t \leftarrow \mu\ \Delta\theta_{t-1} - (1-\mu)\ \gamma\ g_{t-1}$
    $\theta_t \leftarrow \theta_{t-1} + \Delta\theta_t$
    $g_t \leftarrow \nabla f(\theta_t)$
return θ_t

16. Variable Learning Rate Gradient Descent

trainGDX(*params*, *lr=<required parameter>, momentum=<required parameter>,* lr_dec *=<required parameter>,* lr_inc=*<required parameter>,* max_per_inc=*<required parameter>*)

inputs: γ(*lr*: learning rate), θ_0(params), $f(\theta)$ (objective), μ (momentum = 0.90), γ_{dec}(lr_dec = 0.70: Ratio to decrease learning rate), γ_{inc}(lr_inc = 1.05: Ratio to increase learning rate), max_per_inc (maximum performance increase = 1.04)

$g_0 \leftarrow \nabla f(\theta_0)$
$\Delta\theta_0 \leftarrow \gamma\ g_0$
for $t = 1$ ***to*** ... ***do***
gradiente actual
gradiente sig $\Delta\theta_t \leftarrow \mu\ \Delta\theta_{t-1} - (1-\mu)\ \gamma\ g_{t-1}$
    $\theta_t \leftarrow \theta_{t-1} + \Delta\theta_t$
    if $\frac{f(\theta_t)}{f(\theta_{t-1})} >$ max_$perf_inc$
        $\gamma \leftarrow \gamma\ \gamma_{dec}$
        $g_t \leftarrow g_{t-1}$
        $\Delta\theta_t \leftarrow \gamma\ g_t$
    else
        if $f(\theta_t) < f(\theta_{t-1})$
            $\gamma = \gamma\ \gamma_{inc}$
        end
        $g_t \leftarrow \nabla f(\theta_t)$
    end
return θ_t

17. Levenberg-Marquardt

trainLM(*params*, *lr=<required parameter>,* lr_dec *=<required parameter>,* lr_inc=*<required parameter>,* lr_max *=<required parameter>*)

inputs: γ(*lr = 0.001*: Initial damping factor), θ_0(params), $f(\theta)$ (objective), γ_{dec}(lr_dec = 0.1: decrease damping factor), γ_{inc}(lr_inc = 10.0: increase damping factor), γ_{max}(lr_max = 10^{10} : máximum damping factor)

$$
\begin{aligned}
&\boldsymbol{for}\ t = 1\ \boldsymbol{to}\ \ldots\ \boldsymbol{do}\\
&\quad \boldsymbol{while}\ (\gamma \leq \gamma_{max})\\
&\qquad // \left(J_f'(\theta_{t-1})\, J_f(\theta_{t-1}) + \gamma I\right)\Delta\theta_t = -J_F'(\theta_{t-1})\, f(\theta_{t-1})\\
&\qquad \Delta\theta_t = -\left(J_f'(\theta_{t-1})\, J_f(\theta_{t-1}) + \gamma I\right)^{-1} J_f'(\theta_{t-1})\, f(\theta_{t-1})\\
&\qquad \theta_t = \theta_{t-1} + \Delta\theta_t\\
&\qquad \boldsymbol{if}\ f(\theta_t) < f(\theta_{t-1})\\
&\qquad\quad \gamma = \gamma\, \gamma_{dec}\\
&\qquad\quad \text{break}\ \ // \text{ terminates the execution of a for while loop.}\\
&\qquad \boldsymbol{end}\\
&\qquad \gamma = \gamma\, \gamma_{inc}\\
&\quad \boldsymbol{end}\\
&\boldsymbol{end}
\end{aligned}
$$

Validación cruzada

Los procesos iterativos por los cuales se efectúa el ajuste de un modelo, en base a datos experimentales, generalmente necesitan un criterio que les indique en qué momento detenerse. El criterio que se elija para detener el proceso de ajuste o entrenamiento debe tomar en cuenta algunas consideraciones, por ejemplo, la calidad de la salida del modelo, la cantidad en que se filtra el ruido experimental, la presencia de un sobreajuste de los datos, etc.

La validación cruzada es una técnica que indica el momento oportuno para detener el entrenamiento de una RNA, de modo que indica la capacidad de generalización de la RNA durante su proceso de entrenamiento.

La validación cruzada consiste en dividir los datos experimentales de entrada ($\boldsymbol{X} = [\boldsymbol{x_{i,p}}]$ para i=1,…,n y p=1,…,q) y datos de salida deseados $\boldsymbol{T} = \left[\boldsymbol{t_{j,p}}\right]$ para j=1,…,m y p=1,…,q; en dos conjuntos. Uno de ellos es el *conjunto de entrenamiento* y el otro es llamado *conjunto de validación*. El conjunto de entrenamiento es utilizado para ajustar los parámetros de la red durante el proceso de entrenamiento, a su vez, el conjunto de validación se utiliza para medir el rendimiento de la red mientras es entrenada de forma iterativa.

Cuando se construye una gráfica de la función error a partir del conjunto de validación respecto del número de iteraciones, Figura 2.7, se observa un comportamiento irregular, es decir, el error presenta muchos picos. La evolución del error de validación, aunque irregular, empieza a disminuir hasta alcanzar un mínimo para después aumentar. Por lo tanto, el error de validación funciona como criterio para determinar el momento adecuado en que debe detenerse el proceso de entrenamiento.

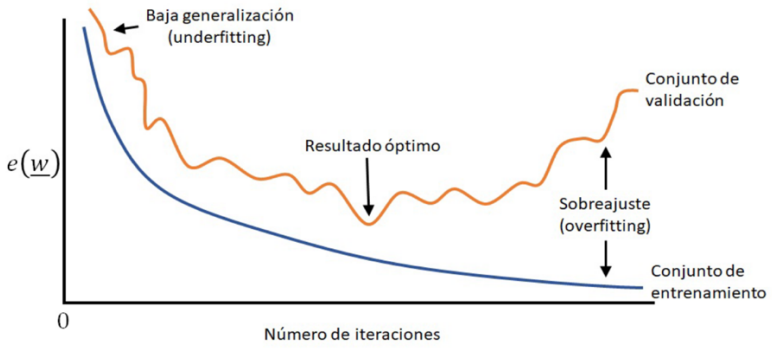

Figura 2.7: Comportamiento de los errores del conjunto de entrenamiento y validación durante el ajuste de parámetros de la RNA.

Durante el proceso de entrenamiento es posible identificar tres etapas: 1) la etapa de sobreajuste (overfitting), Figura 2.8a, es decir, que la red no tiene la capacidad de distinguir entre la señal y el ruido que contienen los datos de entrada, el entrenamiento debe detenerse antes de llegar a este punto; 2) la etapa de baja generalización(underfitting), Figura 2.8c, se presenta cuando los parámetros no se ajustan adecuadamente y la red no puede predecir el comportamiento real del sistema, generalmente ocurre cuando el proceso de entrenamiento es detenido antes de tiempo; 3) resultado óptimo, Figura 2.8b, es donde la red presenta la mejor capacidad de generalización.

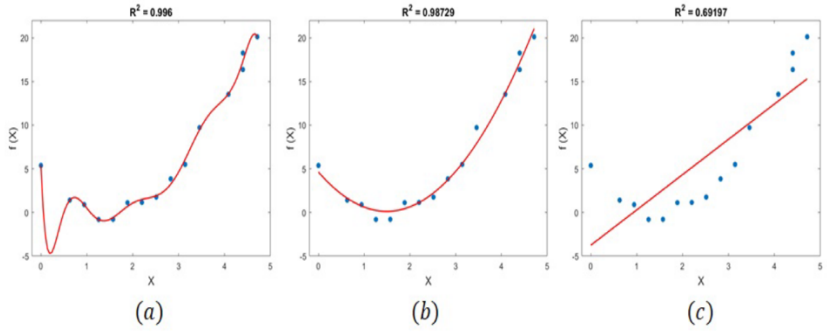

Figura 2.8: Distintos tipos de ajuste en el entrenamiento de una RNA: a) sobreajuste, b) ajuste óptimo, c) bajo-ajuste.

En el entrenamiento usando los distintos algoritmos de optimización, se detiene cuando ocurre cualquiera de estas condiciones:

1) Se alcanza el número máximo de épocas (repeticiones).

2) Se ha excedido la cantidad máxima de tiempo.

3) El rendimiento se ha minimizado hasta la meta de la función objetivo ($\boldsymbol{goal = 10^{-8}}$).

4) El gradiente de rendimiento cae por debajo de $\boldsymbol{mingrad = 10^{-10}}$

5) El rendimiento de la validación ha aumentado más de (max_fail = 6) veces desde la última vez que disminuyó (al usar la validación).

La idea de construir un modelo de Machine Learning no es sólo que lo haga bien con los conjuntos de entrenamiento y validación. La idea es que además lo haga bien con datos que nunca antes haya visto, pues de esta manera veremos qué tan robusto es y cómo se comportará con nuevos datos. Esto se conoce como generalización.

Para ver la capacidad de generalización del modelo seleccionado debemos usar un tercer conjunto de datos, que se conoce como el *conjunto de prueba*. Este conjunto de prueba es un conjunto de datos diferente al conjunto de entrenamiento y validación pero proveniente de la misma distribución. Con este tercer conjunto podemos finalmente poner a prueba el modelo para ver qué tan bien lo hace con datos nunca antes vistos, es decir su capacidad de generalización.

Idealmente para entrenar diferentes modelos de Machine Learning y poder escoger el mejor de todos deberíamos tener tres conjuntos: **entrenamiento**, **validación** y **prueba**

- El ***conjunto de entrenamiento*** se usa para entrenar como tal cada modelo (es decir, para aprender los parámetros)
- El ***conjunto de validación*** lo usamos para poder escoger el valor más adecuado de los hiperparámetros de cada modelo y para elegir al final el mejor modelo de todos los que hayamos entrenado
- El ***conjunto de prueba*** lo usamos para medir la capacidad de generalización del modelo seleccionado.

Es importante además tener en cuenta que los tres conjuntos deben ser diferentes pero a la vez deben provenir de la misma distribución.