Elementos de Cálculo Numérico

Primer Cuatrimestre 2026

Laboratorio N° 6: Entrenamiento de una Red Neuronal

Red neuronal de una capa

Una **red neuronal de una capa oculta** con m neuronas es una función $F : \mathbb{R}^d \to \mathbb{R}$ de la forma

$$F(x; W, b, \alpha) = \sum_{j=1}^{m} \alpha_j \, \sigma\left(\sum_{k=1}^{d} w_{jk} \, x_k + b_j\right) = \sum_{j=1}^{m} \alpha_j \, \sigma(w_j \cdot x + b_j),$$

donde:

- $x \in \mathbb{R}^d$ es la entrada,
- $W = (w_1, \ldots, w_m) \in \mathbb{R}^{m \times d}$ son los *pesos* de la capa oculta, $b \in \mathbb{R}^m$ los *sesgos*,
- $\alpha \in \mathbb{R}^m$ son los pesos de la capa de salida,
- $\sigma : \mathbb{R} \to \mathbb{R}$ es la *función de activación*, que se aplica componente a componente.

El conjunto de todos los parámetros es $\theta = (W, b, \alpha) \in \mathbb{R}^p$ con $p = m(d + 2)$.

Usaremos la función de activación **sigmoide**:

$$\sigma(z) = \frac{1}{1 + e^{-z}}.$$

Problema de regresión

Dados N datos de entrenamiento $(x^{(i)}, y^{(i)}) \in \mathbb{R}^d \times \mathbb{R}$ $(i = 1, \ldots, N)$, buscamos los parámetros θ que minimicen el *error cuadrático medio*:

$$L(\theta) = \frac{1}{2N} \sum_{i=1}^{N} \left(F(x^{(i)}; \theta) - y^{(i)}\right)^2.$$

Ejercicio 1 (Derivada de la sigmoide.) Demuestre que $\sigma'(z) = \sigma(z)(1 - \sigma(z))$. Programe una función que evalúe $\sigma(z)$ y su derivada.

Ejercicio 2 (Gradiente de la función de pérdida.) Sea $r_i(\theta) = F(x^{(i)}; \theta) - y^{(i)}$ el residuo en el dato i-ésimo.

1. Demuestre que las derivadas parciales de L respecto de los parámetros son:

$$\frac{\partial L}{\partial \alpha_j} = \frac{1}{N} \sum_{i=1}^{N} r_i \, \sigma(w_j \cdot x^{(i)} + b_j),$$

$$\frac{\partial L}{\partial b_j} = \frac{1}{N} \sum_{i=1}^{N} r_i \, \alpha_j \, \sigma'(w_j \cdot x^{(i)} + b_j),$$

$$\frac{\partial L}{\partial w_{jk}} = \frac{1}{N} \sum_{i=1}^{N} r_i \, \alpha_j \, \sigma'(w_j \cdot x^{(i)} + b_j) \, x_k^{(i)}.$$

2. Programe una función `gradiente(theta, X, y)` que, dados los parámetros y los datos, calcule $\nabla L(\theta) \in \mathbb{R}^p$.

3. Verifique numéricamente su gradiente comparando con diferencias finitas:

$$\frac{\partial L}{\partial \theta_\ell} \approx \frac{L(\theta + h\, e_\ell) - L(\theta - h\, e_\ell)}{2h}$$

para h pequeño (por ejemplo $h = 10^{-5}$) y varias componentes ℓ.

Ejercicio 3 (Descenso por gradiente.) El método de descenso por gradiente con paso constante $\eta > 0$ actualiza los parámetros iterativamente:

$$\theta^{(t+1)} = \theta^{(t)} - \eta \, \nabla L(\theta^{(t)}).$$

Considere la función $f(x) = \sin(2\pi x)$ muestreada en $N = 50$ puntos equiespaciados en $[0, 1]$, es decir, $x^{(i)} = i/N$ e $y^{(i)} = \sin(2\pi x^{(i)})$. Use $d = 1$ y $m = 10$ neuronas. Inicialice los parámetros $\theta^{(0)}$ con valores aleatorios pequeños (por ejemplo, normales con media 0 y desvío 0.5).

1. Programe el algoritmo de descenso por gradiente. En cada iteración, registre el valor de $L(\theta^{(t)})$.

2. Corra el algoritmo con $\eta = 0.1$ durante $T = 5000$ iteraciones. Grafique $L(\theta^{(t)})$ en función de t. ¿Converge? ¿Es rápido o lento?

3. Repita con $\eta = 1$ y $\eta = 10$. ¿Qué observa? ¿Alguno de estos valores diverge?

4. Experimente con distintos valores de η hasta encontrar el mayor paso para el cual el algoritmo converge de manera estable. Grafique la curva de pérdida para su elección.

5. Con el mejor η encontrado, grafique la función aprendida $F(x; \theta^{(T)})$ junto con los datos de entrenamiento. ¿Se ajusta bien?

Ejercicio 4 (Efecto del número de neuronas.) Repita el ejercicio anterior con $m = 2, 5, 20, 50$ neuronas (ajustando η en cada caso si es necesario).

1. ¿Cómo cambia la calidad del ajuste al aumentar m?

2. ¿Cómo cambia la velocidad de convergencia?

3. ¿Qué ocurre si m es muy grande comparado con el número de datos N?

Ejercicio 5 (Una función más difícil.) Repita el entrenamiento para la función $f(x) = |x - 0.5|$ en $[0, 1]$. Esta función no es suave.

1. ¿Logra la red aproximar bien la función? ¿Cuántas neuronas necesita?

2. Compare la curva de pérdida con la del caso $\sin(2\pi x)$. ¿Converge más lento? ¿Por qué podría ser?