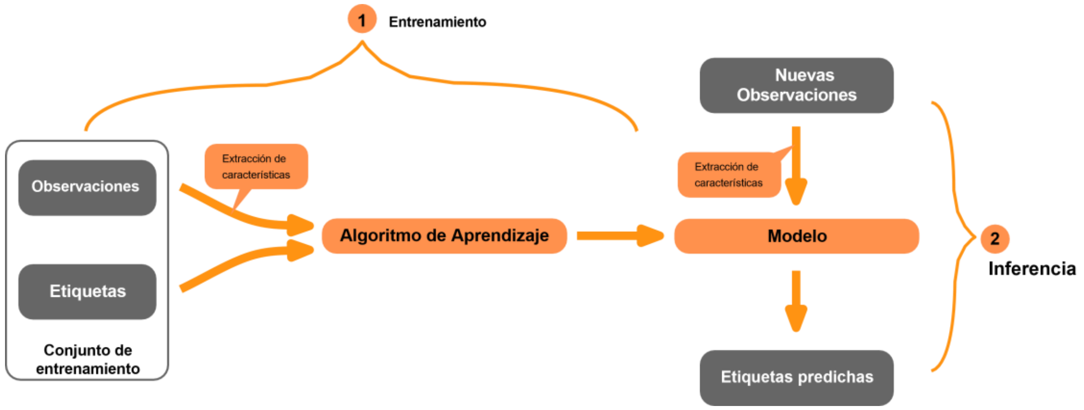
Lecture 02

Modelos lineales para regresión y clasificación

Temas de la Lección

- Regresión logísitca
- Regresión softmax

Aprendizaje supervisado



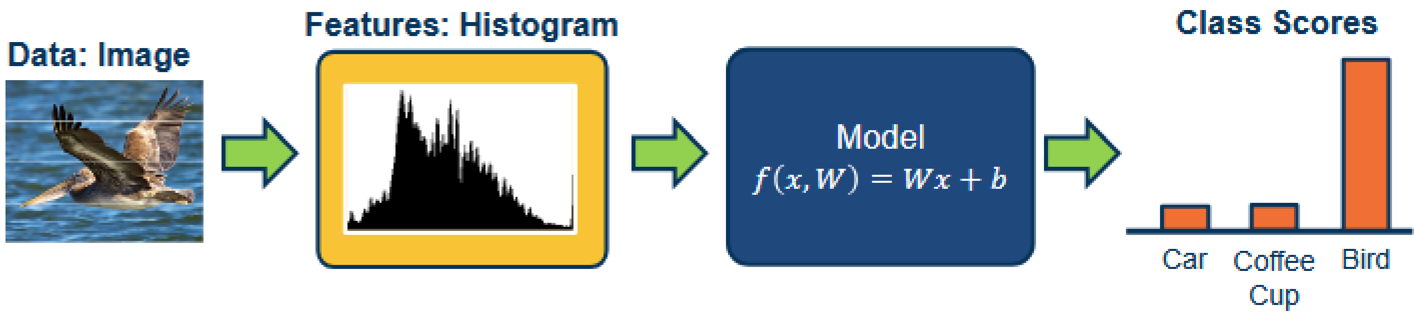
Dado un conjunto de ejemplos:

$$\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^{N}$$

- Entrenamiento: Ajustar un modelo $f : \mathbf{x} \mapsto y$ a partir de los datos
- Prueba: Utilizar el modelo para predecir y dado $\mathbf{x}$, i.e,

$$\hat{y} = f(\mathbf{x})$$

Problema de Clasificación



Input Image

$$x = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1n} \\ x_{21} & x_{22} & \cdots & x_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n1} & x_{n2} & \cdots & x_{nn} \end{bmatrix}$$

- **Antes de Deep Learning:** Las características se diseñaban a mano
- **Con Deep Learning:** Las características se aprenden automáticamente junto con el clasificador

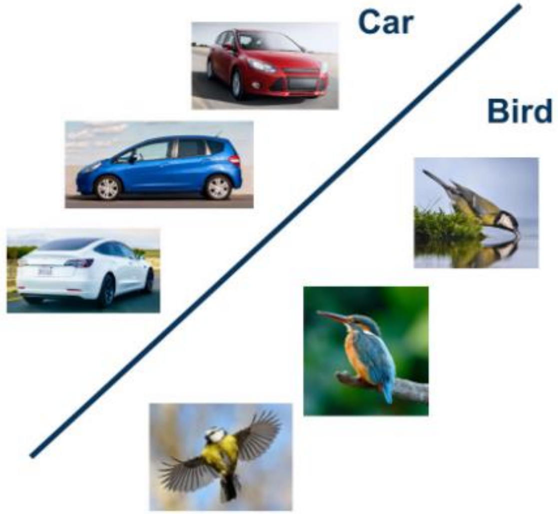
Regresión logística

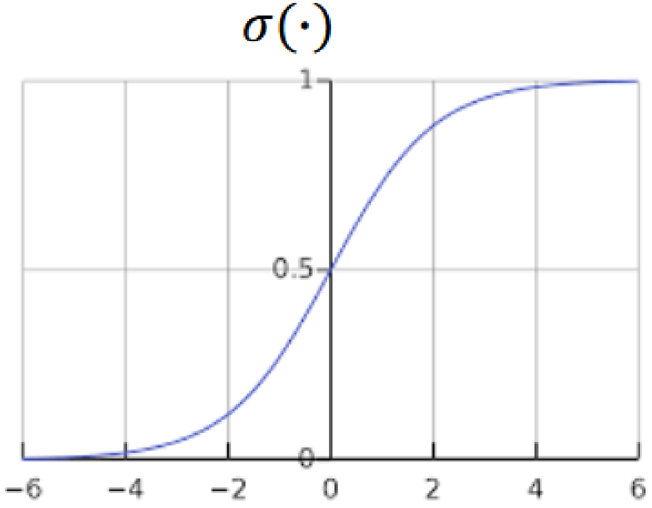
- El modelo aprende un **hiperplano** para separar los datos en dos clases.
- La salida es la probabilidad de que una muestra pertenezca a la clase 1:

$$p(y = 1|\mathbf{x}) = \sigma(\mathbf{w}^T\mathbf{x} + b)$$

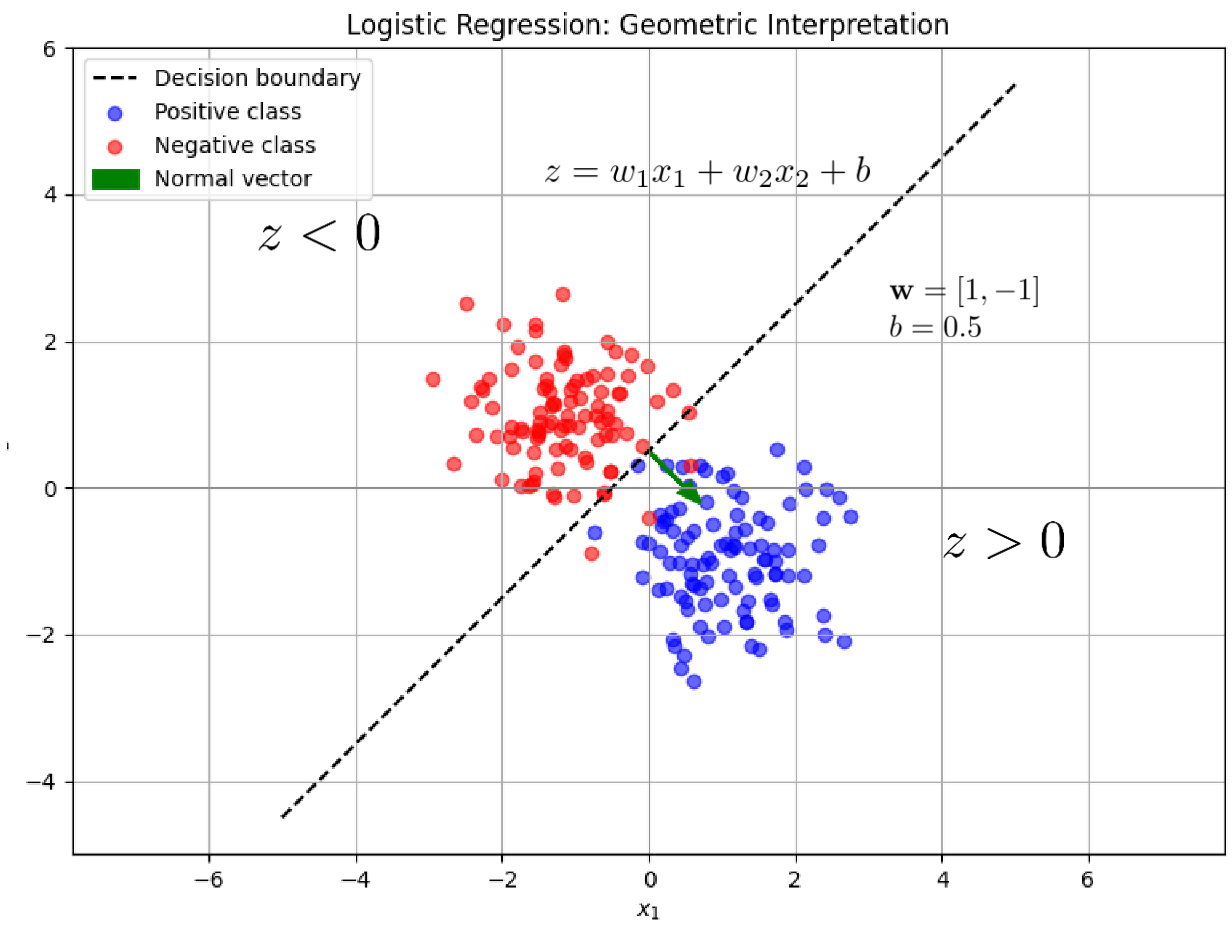
- La entrada lineal $(\mathbf{w}^T\mathbf{x} + b)$, llamada **logit**, se transforma en una probabilidad (un valor entre 0 y 1) usando la **función sigmoide** $\sigma(z)$:

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





Regresión logística



Regresión logística

- Sea $y \in \{0, 1\}$ una variable aleatoria binaria.
- Suponemos que la probabilidad de que $y = 1$ depende de la entrada $\mathbf{x}$ mediante:

$$\hat{y} = p(y = 1 \mid \mathbf{x}) = \sigma(\mathbf{w}^\top \mathbf{x} + b)$$

- Entonces la distribución condicional de y dada $\mathbf{x}$ es:

$$P(y \mid \mathbf{x}) = \begin{cases} \hat{y} & \text{si } y = 1 \\ 1 - \hat{y} & \text{si } y = 0 \end{cases}$$

- Esta expresión se puede escribir de forma más compacta como:

$$\boxed{p(y \mid \mathbf{x}) = \hat{y}^y (1 - \hat{y})^{1-y}}$$

Regresión logística

- Función de verosimilitud:

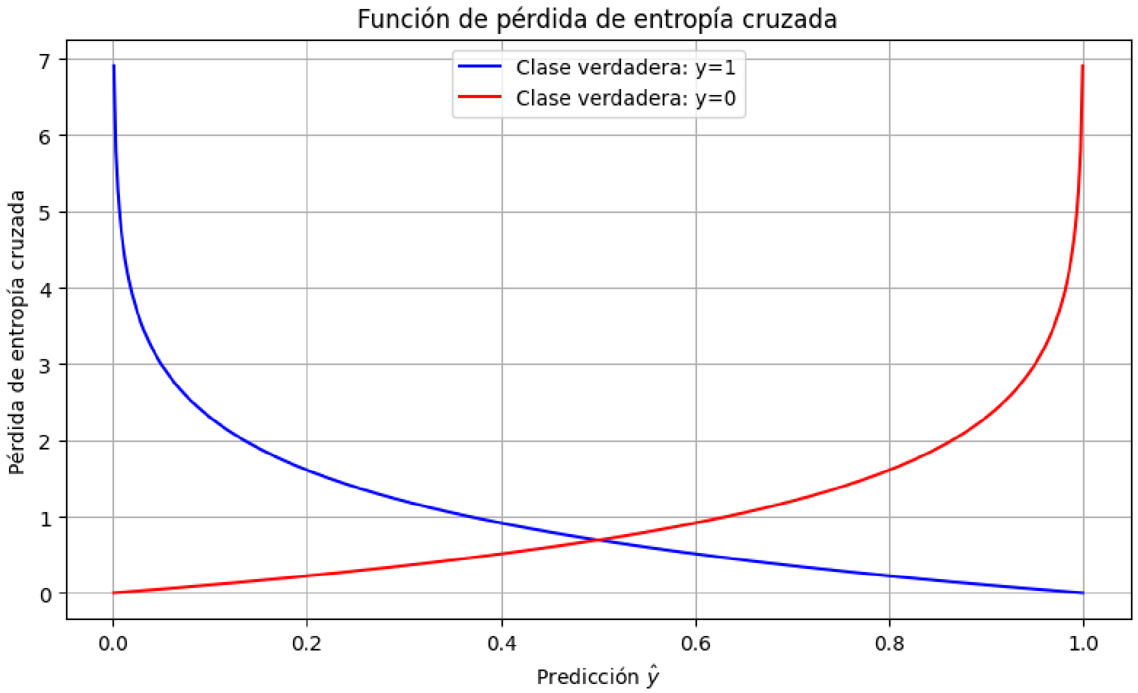
$$\mathcal{L} = \prod_{i=1}^{N} P(y^{(i)} \mid \mathbf{x}^{(i)}) = \prod_{i=1}^{N} \hat{y}^{(i)y^{(i)}} (1-\hat{y}^{(i)})^{1-y^{(i)}}$$

- Tomando logaritmo:

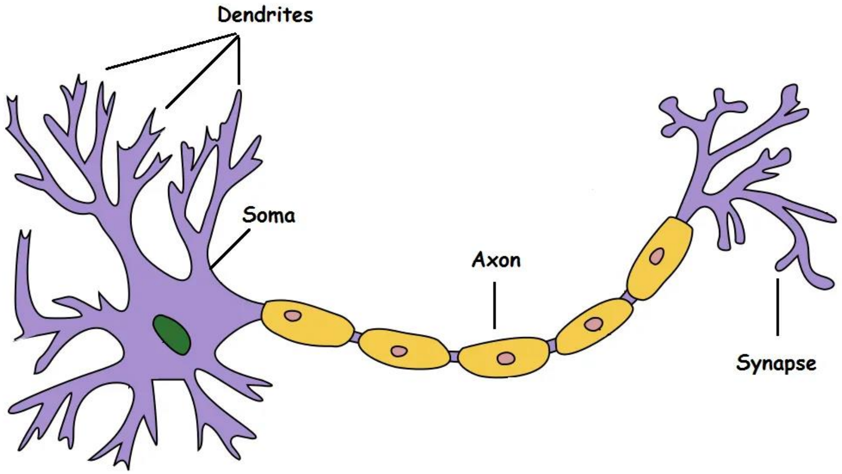
$$\log \mathcal{L} = \sum_{i=1}^{N} \left[y^{(i)} \log \hat{y}^{(i)} + (1 - y^{(i)}) \log(1 - \hat{y}^{(i)}) \right]$$

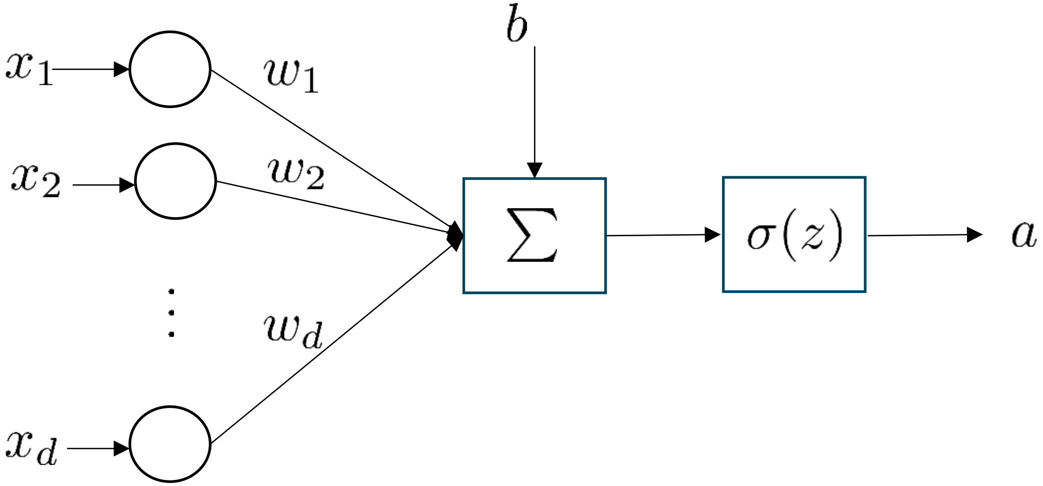
- Entropía cruzada:

$$\boxed{\mathcal{L}_{\text{CE}} = -\frac{1}{N} \sum_{i=1}^{N} \left[y^{(i)} \log \hat{y}^{(i)} + (1 - y^{(i)}) \log(1 - \hat{y}^{(i)}) \right]}$$



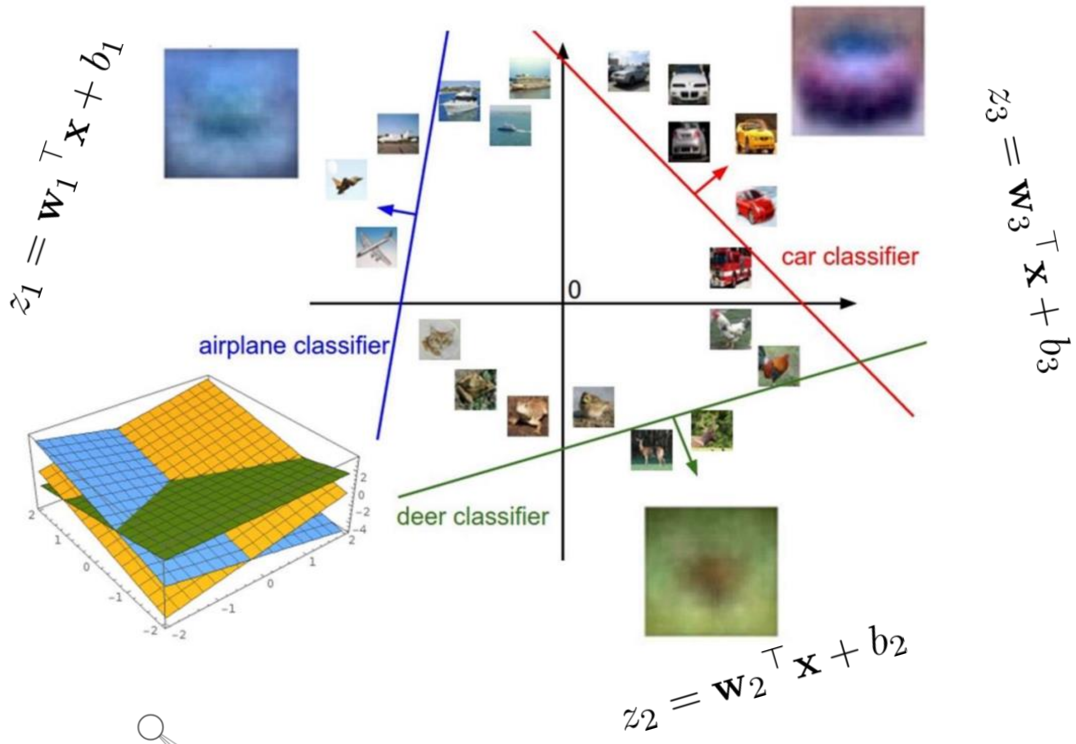
Neurona de McCulloch-Pitts

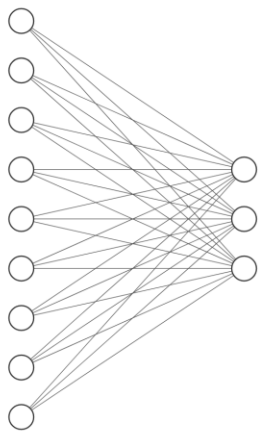




Regresión softmax

- Se definen múltiples hiperplanos:
  $$\{(\mathbf{w}_c, b_c)\}, \quad c = 1, \ldots, C$$
- Logits:
  $$z_c = \mathbf{w}_c^\top \mathbf{x} + b_c$$
- Probabilidad de clase:
  $$P(y = c \mid \mathbf{x}) = \mathrm{softmax}_c(z_c)$$
- Función softmax:
  $$\mathrm{softmax}_c(\{z_i\}) = \frac{e^{z_c}}{\sum_{i=1}^{C} e^{z_i}}$$





$$\mathbf{z} = \mathbf{W}\mathbf{x} + \mathbf{b}$$

$$\mathbf{W} = \begin{bmatrix} - & {\mathbf{w}^1}^\top & - \\ - & {\mathbf{w}^2}^\top & - \\ - & {\mathbf{w}^3}^\top & - \end{bmatrix} \quad \mathbf{b} = \begin{bmatrix} b^1 \\ b^2 \\ b^3 \end{bmatrix}$$

Regresión softmax

- $y \in \{1, \ldots, C\}$, vector one-hot $\mathbf{y} \in \{0,1\}^C$
- Probabilidad multiclase:

$$\hat{\mathbf{y}} = \text{softmax}(\mathbf{W}\mathbf{x} + \mathbf{b})$$

- PDF categórica:

$$P(y \mid \mathbf{x}) = \prod_{c=1}^{C} \hat{y}_c^{y_c}$$

- Log-verosimilitud:

$$\log \mathcal{L} = \sum_{i=1}^{N} \sum_{c=1}^{C} y_c^{(i)} \log \hat{y}_c^{(i)}$$

- Función de pérdida:

$$\boxed{\mathcal{L}_{\text{CE}} = -\frac{1}{N} \sum_{i=1}^{N} \sum_{c=1}^{C} y_c^{(i)} \log \hat{y}_c^{(i)}}$$

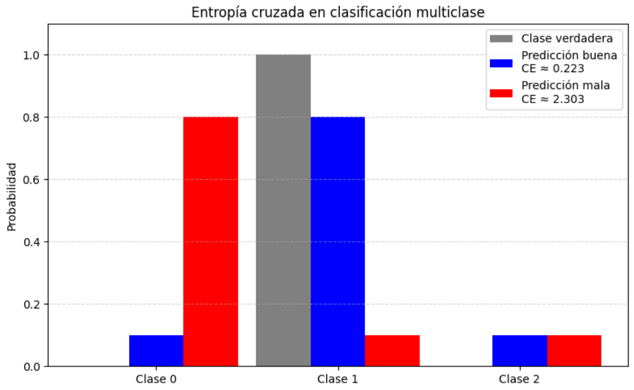
Ejemplo 1 (predicción correcta):

$$\mathbf{y} = [0,1,0] \qquad \hat{\mathbf{y}} = [0.1,\ 0.8,\ 0.1]$$

$$\mathcal{L}_{\text{CE}} = -\log(0.8) \approx 0.223$$

Ejemplo 2 (predicción incorrecta):

$$\mathbf{y} = [0,1,0] \qquad \hat{\mathbf{y}} = [0.8,\ 0.1,\ 0.1]$$



Optimización

Objetivo: minimizar la función de pérdida $\mathcal{L}(\mathbf{w})$

Algoritmo:

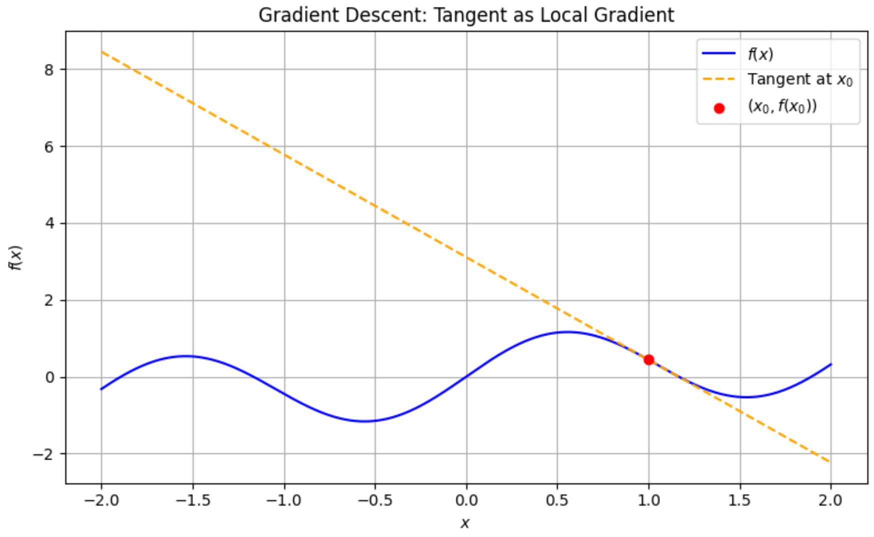
1. Inicializar los pesos: $\mathbf{w}^{(0)} \sim$ aleatorio
2. Para cada iteración $t = 0, 1, 2, \ldots$:
   - Calcular el gradiente:

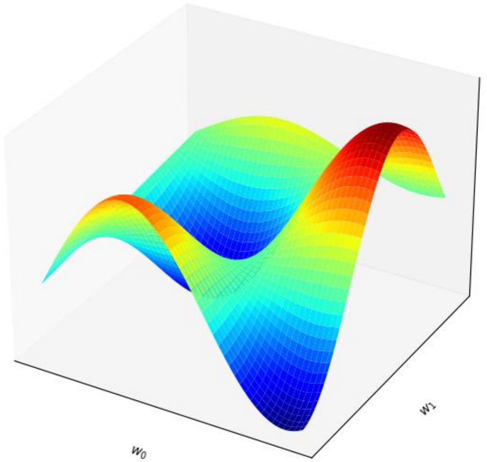
     $$\nabla_{\mathbf{w}} \mathcal{L}(\mathbf{w}^{(t)})$$

   - Actualizar los pesos:

     $$\mathbf{w}^{(t+1)} = \mathbf{w}^{(t)} - \eta \nabla_{\mathbf{w}} \mathcal{L}(\mathbf{w}^{(t)})$$

     donde $\eta > 0$ es la tasa de aprendizaje.
3. Repetir hasta convergencia





Optimización

Input: Dataset $\mathcal{D} = \{(\mathbf{x}^{(i)}, y^{(i)})\}_{i=1}^{n}$, learning rate η
Output: Parameters $\mathbf{W}, \mathbf{b}$
Initialize $\mathbf{W}, \mathbf{b}$;
for *epoch* $t = 1$ **to** T **do**
    Compute gradient over entire dataset:;
    $\nabla_{\mathbf{W}}\mathcal{L} = \frac{1}{n}\sum_{i=1}^{n} \nabla_{\mathbf{W}}\ell(\hat{y}^{(i)}, y^{(i)})$;
    $\nabla_{b}\mathcal{L} = \frac{1}{n}\sum_{i=1}^{n} \nabla_{b}\ell(\hat{y}^{(i)}, y^{(i)})$;
    Update parameters:;
    $\mathbf{W} \leftarrow \mathbf{W} - \eta\nabla_{\mathbf{W}}\mathcal{L}$;
    $\mathbf{b} \leftarrow \mathbf{b} - \eta\nabla_{b}\mathcal{L}$;
end

Batch Gradient Descent

Input: Dataset $\mathcal{D} = \{(\mathbf{x}^{(i)}, y^{(i)})\}_{i=1}^{n}$, learning rate η
Output: Parameters $\mathbf{W}, \mathbf{b}$
Initialize $\mathbf{W}, \mathbf{b}$;
for *epoch* $t = 1$ **to** T **do**
    foreach *example* $(\mathbf{x}^{(i)}, y^{(i)}) \in \mathcal{D}$ **do**
        Compute gradient:;
        $\nabla_{\mathbf{W}}\ell(\hat{y}^{(i)}, y^{(i)})$, $\nabla_{b}\ell(\hat{y}^{(i)}, y^{(i)})$;
        Update:;
        $\mathbf{W} \leftarrow \mathbf{W} - \eta\nabla_{\mathbf{W}}\ell(\hat{y}^{(i)}, y^{(i)})$;
        $\mathbf{b} \leftarrow \mathbf{b} - \eta\nabla_{b}\ell(\hat{y}^{(i)}, y^{(i)})$;
    end
end

Online Gradient Descent

Optimización

UNIVERSIDAD EAFIT
Escuela de Ciencias Aplicadas e Ingeniería

Input: Dataset $D = \{(\mathbf{x}^{(i)}, y^{(i)})\}$, learning rate η
Output: Parameters $\mathbf{W}, \mathbf{b}$
Initialize $\mathbf{W}, \mathbf{b}$;
for *epoch* $t = 1$ **to** T **do**
    foreach *mini-batch* $\mathcal{B} \subset D$ **do**
        Compute gradients $\nabla_{\mathbf{W}}\mathcal{L}_{\mathcal{B}},\ \nabla_{b}\mathcal{L}_{\mathcal{B}}$;
        Update parameters:;
        $\mathbf{W} \leftarrow \mathbf{W} - \eta\nabla_{\mathbf{W}}\mathcal{L}_{\mathcal{B}}$;
        $\mathbf{b} \leftarrow \mathbf{b} - \eta\nabla_{\mathbf{b}}\mathcal{L}_{\mathcal{B}}$;
    end
end

Mini-Batch Gradient Descent

Optimización

UNIVERSIDAD EAFIT
Escuela de Ciencias Aplicadas e Ingeniería

Objetivo: derivar el gradiente de $\mathcal{L}$ respecto a $\mathbf{W}$

Para una sola muestra $(\mathbf{x}, \mathbf{y})$, la derivada de la pérdida es:

$$\frac{\partial \mathcal{L}}{\partial \mathbf{W}} = (\hat{\mathbf{y}} - \mathbf{y})\mathbf{x}^\top$$

Interpretación:

- $\hat{\mathbf{y}} - \mathbf{y} \in \mathbb{R}^C$: error de predicción por clase
- $\mathbf{x}^\top \in \mathbb{R}^{1\times d}$: entrada transpuesta
- Resultado: matriz $C \times d$, mismo tamaño que $\mathbf{W}$

Gradiente respecto a $\mathbf{b}$:

$$\frac{\partial \mathcal{L}}{\partial \mathbf{b}} = \hat{\mathbf{y}} - \mathbf{y}$$

Optimización

Modelo: softmax + entropía cruzada

$$\mathcal{L}(\mathbf{W}, \mathbf{b}) = -\frac{1}{N}\sum_{i=1}^{N}\sum_{c=1}^{C} y_c^{(i)} \log \hat{y}_c^{(i)}$$

Gradientes:

$$\frac{\partial \mathcal{L}}{\partial \mathbf{W}} = \frac{1}{N}\sum_{i=1}^{N}\left(\hat{\mathbf{y}}^{(i)} - \mathbf{y}^{(i)}\right)\mathbf{x}^{(i)\top}$$

$$\frac{\partial \mathcal{L}}{\partial \mathbf{b}} = \frac{1}{N}\sum_{i=1}^{N}\left(\hat{\mathbf{y}}^{(i)} - \mathbf{y}^{(i)}\right)$$

Actualización de parámetros:

$$\mathbf{W} \leftarrow \mathbf{W} - \eta\frac{\partial \mathcal{L}}{\partial \mathbf{W}}$$

$$\mathbf{b} \leftarrow \mathbf{b} - \eta\frac{\partial \mathcal{L}}{\partial \mathbf{b}}$$

Características:

- Usa *todos* los datos en cada paso.
- Gradientes estables.
- Costoso para datasets grandes.

Optimización

Universidad EAFIT — Escuela de Ciencias Aplicadas e Ingeniería

Datos:

- $\mathbf{X} \in \mathbb{R}^{N\times d}$: entradas
- $\mathbf{Y} \in \mathbb{R}^{N\times C}$: etiquetas one-hot

Modelo:

$$\mathbf{Z} = \mathbf{X}\mathbf{W}^\top + \mathbf{b} \quad \Rightarrow \quad \hat{\mathbf{Y}} = \text{softmax}(\mathbf{Z})$$

Gradientes:

$$\frac{\partial \mathcal{L}}{\partial \mathbf{W}} = \frac{1}{N}(\hat{\mathbf{Y}} - \mathbf{Y})^\top \mathbf{X}$$

$$\frac{\partial \mathcal{L}}{\partial \mathbf{b}} = \frac{1}{N}\sum_{i=1}^{N}(\hat{\mathbf{y}}^{(i)} - \mathbf{y}^{(i)})$$

Actualización:

$$\mathbf{W} \leftarrow \mathbf{W} - \eta\frac{\partial \mathcal{L}}{\partial \mathbf{W}} \quad \mathbf{b} \leftarrow \mathbf{b} - \eta\frac{\partial \mathcal{L}}{\partial \mathbf{b}}$$