7. Aprendizaje en Redes Multicapa, Retropropagación

7.1 El Problema: El Juego de la Culpa

En un **Perceptrón Simple** (una sola neurona), si el sistema falla, sabemos que la culpa es de esa única neurona. El error es fácil de calcular: `Objetivo - Salida`.

En una **Red Multicapa** (Deep Learning), tenemos un problema grave: **El Problema de Asignación de Crédito**.

- Si la red dice "Gato" y era "Perro", la neurona de salida se equivocó.
- Pero, ¿por qué se equivocó? Quizás porque la neurona oculta de la capa anterior le pasó un dato malo.
- ¿Y por qué esa neurona oculta le pasó un dato malo? Quizás porque la anterior a ella falló.

No tenemos un "profesor" para las capas intermedias. **La Retropropagación** es el método matemático para enviar la "culpa" del error desde el final hacia atrás, capa por capa, para saber cuánto ajustar cada peso interno.

7.2 Definiciones Previas: Anatomía de la Señal

Para entender las fórmulas, primero debemos distinguir qué pasa dentro de cada neurona.

1. Entrada Neta (Net o z): Es la suma bruta de lo que recibe la neurona.

$$Net_j = \sum (x_i \cdot w_{ij})$$

   - *Es el "ruido total" antes de filtrar.*

2. Salida (y o Out): Es el resultado procesado tras pasar el filtro (sigmoide).

$$y_j = f(Net_j)$$

   - *Es lo que la neurona "grita" a la siguiente capa.*

> **Importante**
>
> Para poder usar Retropropagación, necesitamos una función de activación que sea **suave y continua** (derivable en todo punto). Usamos la **Sigmoide**:
>
> $$f(z) = \frac{1}{1 + e^{-z}}$$

> Esta función curva suavemente la entrada, transformando cualquier número (del $-\infty$ al $+\infty$) en un valor entre **0 y 1**.
>
> Lo mejor de la Sigmoide no es solo su forma, sino que su derivada es increíblemente fácil de calcular en un ordenador. La derivada se puede expresar en función de la propia salida de la neurona (y o $f(z)$):
>
> $$f'(z) = f(z) \cdot (1 - f(z))$$
>
> Para calcular la pendiente (sensibilidad) de la neurona, no necesitas volver a hacer cálculos complejos con exponenciales (e^{-z}). Si la neurona ya sabe su salida (y), simplemente calculas $y \cdot (1 - y)$.

3. **Objetivo (t):** El valor correcto que *deberíamos* haber obtenido (solo existe al final de la red).

7.3 El Fundamento Matemático: El Descenso del Gradiente

Nuestro objetivo es minimizar el **Error Global (E)**. Definimos el error cuadrático medio:

$$E = \frac{1}{2}(t - y)^2$$

- $\frac{1}{2}$: Es un truco matemático. Al derivar x^2, el 2 baja ($2x$). Al multiplicarlo por $1/2$, se cancelan. Nos simplifica la vida.
- **Derivada**: Para aprender, necesitamos saber la pendiente. Si cambio el peso w, ¿el error sube o baja?

La Necesidad de la Derivada (La Regla de la Cadena)

Queremos cambiar un peso w que está en el medio de la red. Pero ese peso no toca el Error (E) directamente.

1. El peso w cambia la **Entrada Neta (Net)**.
2. La Entrada Neta cambia la **Salida (y)**.
3. La Salida cambia el **Error (E)**.

Por la Regla de la Cadena, la derivada total es la multiplicación de estos pasos:

$$\frac{\partial E}{\partial w} = \frac{\partial E}{\partial y} \cdot \frac{\partial y}{\partial Net} \cdot \frac{\partial Net}{\partial w}$$

Aquí es donde nace el concepto de **DELTA (δ)**. Para no escribir todo eso, llamamos δ (Delta o Error Local) a la parte que combina el Error y la Activación: **"Cuánto contribuye la neurona al error total"**.

Tienes esta fórmula general para calcular cómo cambiar el peso:

7.4 El Algoritmo: Ciclo de Dos Pasos

El aprendizaje ocurre repitiendo estos dos pasos.

Paso 1: Propagación Hacia Adelante (Forward Pass)

La red funciona normalmente.

1. Metemos los datos en la entrada.
2. Calculamos **Entradas Netas** y **Salidas** capa por capa hasta el final.
3. Comparamos la salida final con el objetivo real.

Paso 2: Propagación Hacia Atrás (Backward Pass) - Calculando Deltas

Ahora calculamos el error local (δ) de cada neurona.

A. Para la Neurona de Salida (Sabemos la respuesta):
Aquí es fácil. La culpa es la diferencia directa con la realidad, multiplicada por la sensibilidad de la neurona (su derivada).

$$\delta_{salida} = (t - y) \cdot f'(Net)$$

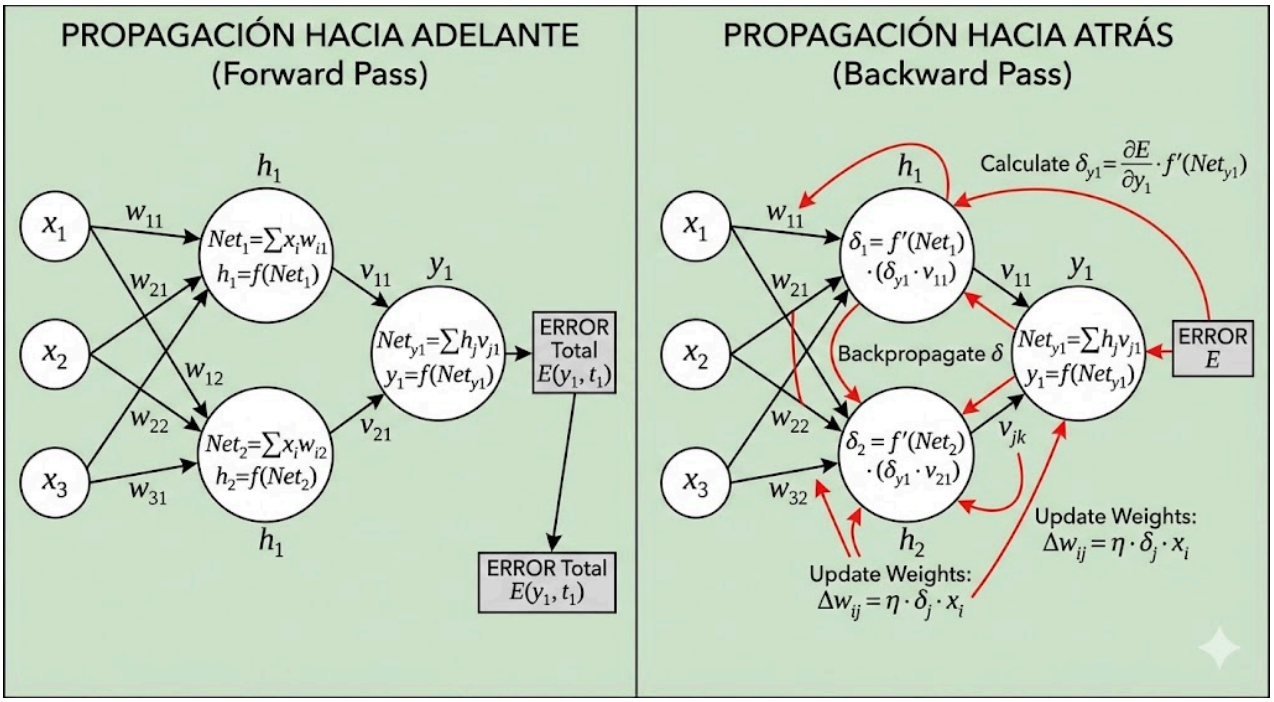
- $(t - y)$: El error bruto.
- $f'(Net)$: La derivada de la función sigmoide. Significa "¿Qué tan fácil era hacer cambiar de opinión a esta neurona?".

B. Para las Neuronas Ocultas (La Magia):
No tenemos $(t - y)$. Calculamos la culpa "escuchando las quejas" de las neuronas de la siguiente capa a las que alimentamos.

$$\delta_{oculta} = f'(Net) \cdot \sum(\delta_{siguiente} \cdot w_{conexion})$$

- $\sum(\delta \cdot w)$: Sumo las culpas de las neuronas siguientes ponderadas por mi conexión con ellas. "Si la neurona siguiente tiene un error gigante (δ alto) y yo le grité muy fuerte (w alto), yo tengo mucha culpa".

> **Nota**
>
> Esta imagen no se si está bien pero bueno es un poco para entender como se monta todo este tinglao.



7.5 Actualización de Pesos (El Aprendizaje)

Una vez que cada neurona tiene su "nota de culpa" (δ), actualizamos los pesos para corregir el error.

$$w_{nuevo} = w_{actual} + \Delta w$$

La corrección (Δw) se calcula así:

$$\Delta w = \eta \cdot \delta \cdot \text{Entrada}_{origen}$$

Desglose de la fórmula:

1. η **(Tasa de aprendizaje):** La **Prudencia**. "Aunque el error sea grande, corregimos poco a poco para no pasarnos".
2. δ **(Delta del destino):** La **Dirección del Error**. "¿Debo subir o bajar el peso?".
3. Entrada_{origen} **(Salida de la anterior):** La **Evidencia**. "Solo corregimos el peso si la neurona de origen estaba activa. Si envió un 0, este peso no tuvo nada que ver en el resultado, así que no se toca".

7.6 Relación entre todas las fórmulas

Partimos de la fórmula del inicio

$$\frac{\partial E}{\partial w} = \underbrace{\frac{\partial E}{\partial y} \cdot \frac{\partial y}{\partial Net}}_{\text{Parte A}} \cdot \underbrace{\frac{\partial Net}{\partial w}}_{\text{Parte B}}$$

Si te fijas en la **Parte A** (los dos primeros términos), verás que coinciden exactamente con la definición de δ_{salida}:

- **1º Término:** $\frac{\partial E}{\partial y}$

    - Como calculamos antes, esto es igual a $-(t - y)$.
    - *(El signo menos se suele absorber en la fórmula final de actualización de pesos).*
- **2º Término:** $\frac{\partial y}{\partial Net}$
    - Esto es la derivada de la función de activación, es decir, $f'(Net)$.

Si multiplicas esos dos términos, obtienes exactamente tu fórmula de delta:

$$\delta_{salida} \approx \frac{\partial E}{\partial y} \cdot \frac{\partial y}{\partial Net} = (t - y) \cdot f'(Net)$$

> *Nota técnica: Matemáticamente, δ se define como $-\frac{\partial E}{\partial Net}$. El signo negativo del error $(y - t)$ se cancela con el menos de la definición, quedando el positivo $(t - y)$.*

La **Parte B** ($\frac{\partial Net}{\partial w}$) es lo que nos faltaba, esta derivada es simplemente la entrada x.

Gracias a esta relación, podemos reescribir la terrorífica Regla de la Cadena (primera fórmula) de una forma súper simple y elegante para programar:

$$\frac{\partial E}{\partial w} = -\delta \cdot x_{entrada}$$

Y de ahí sale la famosa regla de actualización:

$$\Delta w = \eta \cdot \delta \cdot x_{entrada}$$

En resumen:

- La **primera fórmula** es la explicación matemática completa (el "por qué").
- La **segunda fórmula** (δ) es el resumen práctico de los dos primeros pasos, que representa la "culpa local" de la neurona antes de mirar sus cables de entrada.

7.7 Resumen del Algoritmo Completo

1. **Inicializar pesos:** Valores pequeños aleatorios (nunca todos a cero).
2. **Repetir** hasta que el error sea bajo:
    - **Forward:** Calcular todas las Net y y desde el inicio hasta el fin.
    - **Cálculo de Error (δ) Salida:** Usando $(t - y)$.
    - **Backpropagation:** Calcular los δ ocultos trayendo el error desde el futuro hacia el pasado usando los pesos.
    - **Update:** Modificar todos los pesos (w) usando la fórmula $\eta \cdot \delta \cdot y$.

7.8 Conclusión de la Asignatura

Una vez más se demuestra que el grado presenta una dejadez considerable por parte de las "vacas sagradas" de la facultad. Y es que no puede ser: estas personas son reconocidísimas en sus ámbitos, publican 300 artículos, acuden a congresos

internacionales... pero cuando llega el momento de pensar en sus alumnos y proporcionarles una bibliografía de calidad con la cual puedan aprender los conceptos de forma intuitiva, te sueltan un PDF de 2001 robado de otra universidad o presentaciones sin orden lógico alguno, llenas de palabras sueltas y esquemas incomprensibles.

Eso sí, si te quejas, la alternativa que te ofrecen es el libro de turno de 1990 escrito en alemán por un médico francés. Y apáñate tú para conseguirlo y entenderlo.

La situación es insostenible. Es frecuente encontrarse con:

- Documentación completamente desactualizada
- Materiales de dudosa procedencia sin adaptar al contexto actual
- Presentaciones caóticas sin estructura pedagógica
- Bibliografía obsoleta, inaccesible o en idiomas que nadie domina

¿Tan difícil es hacer las cosas bien? Otras carreras universitarias lo consiguen. No se pide la perfección, pero sí un mínimo de coherencia y actualización en los materiales docentes.

Considero que el grado debería aprender de otras titulaciones y mejorar urgentemente este aspecto. Con la jubilación progresiva de las viejas vacas sagradas de la universidad, espero sinceramente que esta situación mejore con el tiempo y que la nueva generación de profesorado traiga una renovación real en la calidad docente, no solo en la investigadora.

Otra asignatura te lo paso, pero en esta no me jodas. Es imposible que con todos los conocimientos sobre IA que tienes y la cantidad de herramientas que existen, no te salga de dentro redactar unos apuntes propios mediante el uso de la IA.