Ayuda para comprender el artículo de G Tesio *«IA» desmitificada*

La base conceptual: información, dato e informática

Para Tesio, el malentendido actual sobre la «inteligencia artificial» nace de un error más profundo: no saber qué es la informática. Su definición, expresada en *What is Informatics?*, es el cimiento de su crítica:

> ***Información:*** Del latín *informo*, «construyo dentro (de mí mismo)». Es una idea, un constructo que existe única y exclusivamente en la mente de un ser humano y que puede ser compartido con otros. Es el ladrillo del conocimiento humano.
>
> ***Dato***: Del latín *datum*, «dado». Es una de las posibles representaciones de una información, susceptible de ser transferida e interpretada por otros humanos. Un dato no es la información, sino un vehículo para la misma.
>
> ***Relación entre ambos:*** Al escribir, convierto la información de mi mente en datos (texto). Al leer, conviertes esos datos en información en tu mente, pero inevitablemente se generan ambigüedades y se añade información no intencionada (mi estilo, tu estado de ánimo al leer…). La correspondencia no es perfecta.
>
> ***Informática:*** No es la «ciencia de las computadoras». Es el campo del conocimiento humano que estudia cómo la información puede ser transferida, almacenada, representada, interpretada y transformada de forma automática. Las computadoras son sus herramientas, no su fin. Su propósito último es servir de medio para la comunicación entre humanos.

Bajo esta luz, llamar «inteligente» a una máquina que solo procesa datos es un error categorial. La inteligencia, como la información, reside exclusivamente en los humanos.

La máquina desmitificada: Máquina de Mapeo Vectorial Virtual (VMM)

Lo que el marketing llama *«red neuronal artificial»* o *«modelo de IA»* es, para Tesio, una Máquina de Mapeo Vectorial Virtual (VMM): un software diseñado para transformar una lista de números (vector de entrada) en otra lista de números (vector de salida) mediante una composición de operaciones matemáticas.

Esa máquina no «aprende»; es programada estadísticamente (o «compilada») a partir de un conjunto de datos. El resultado de esa compilación —el *«modelo»*— es un archivo ejecutable derivado de los datos de origen, análogo a un programa compilado a partir de su código fuente.

Componentes fundamentales de una VMM

Vector

Una lista ordenada de números que representa una entidad. Cada número es una característica medible. *Ejemplo:* Los píxeles de una imagen de 28×28 pueden disponerse como un vector de 784 números (cada número es la intensidad de un píxel). No hay «significado», solo una secuencia de valores.

Reductor vectorial

Es la unidad mínima de cálculo. Para entenderlo, primero debemos imaginar que nuestro vector de entrada no es solo una "lista de números". Es un punto dentro de un espacio vectorial de *n* dimensiones.

Por ejemplo, un vector de 784 números (como los píxeles de una imagen de 28x28) es un punto en un espacio de 784 dimensiones. Para un "humano 3D" es imposible visualizarlo, pero las matemáticas lo manejan sin problema.

La función de un reductor vectorial es tomar un punto de un espacio de *n* dimensiones (el vector de entrada) y transformarlo en un punto de un espacio de dimensión 1 (un simple escalar). Es decir, pliega la información de ese punto concreto reduciendo su dimensionalidad, o en otras palabras, realizar una operación que, punto a punto, pliega la información de ese espacio de *n* dimensiones hasta reducirla a un escalar.

Para lograr este "plegado", ejecuta tres pasos internos:

1. Producto escalar entre el vector de entrada (el punto de alta dimensión) y un vector de parámetros (llamados *«pesos»*), almacenado en el propio reductor. Matemáticamente:
   (x1·w1) + (x2·w2) + … + (xn·wn)
   Esta operación proyecta el vector de entrada sobre la dirección marcada por los pesos. El resultado es un único número que nos dice cuánto se alinea la entrada con esa dirección.
2. Adición de un sesgo (*bias*): un número fijo que se suma al resultado del producto escalar. Permite desplazar el umbral de operación.
3. Función diferenciable: el valor anterior se pasa por una función matemática no lineal y diferenciable (por ejemplo, la función sigmoide o ReLU).

La fórmula completa de un reductor es:
salida = f((entrada · pesos) + sesgo)

Función diferenciable

Una fórmula cuya derivada existe en todos los puntos relevantes y nos dice, para un cambio infinitesimal en la entrada, cómo varía la salida. Esto permite calcular el gradiente de manera eficiente.

Gradiente

Un vector de la misma dimensión que el conjunto total de parámetros de la máquina. Cada componente indica en qué dirección y con qué magnitud debe modificarse un parámetro concreto (un peso o un sesgo) para que la discrepancia entre la salida obtenida y la salida deseada disminuya.
Es la «receta de corrección» que guía todo el proceso de compilación.

El proceso de compilación estadística

Con estos ingredientes, la VMM se programa (se «compila») mediante un ciclo que se repite millones de veces sobre un conjunto de datos. He aquí la secuencia real, sin antropomorfismos:

1. Propagación hacia adelante (*forward pass*)
   Se toma un vector de entrada del conjunto de datos y se hace avanzar a través de todas las capas de

reductores vectoriales. Cada reductor aplica su producto escalar, sesgo y función diferenciable. Al final, la máquina produce un vector de salida.

2. Medición del error (función de pérdida)
   Se compara la salida producida con la salida esperada (por ejemplo, la etiqueta correcta) mediante una función matemática que arroja un número: el error. Cuanto mayor es el número, peor es el ajuste.

3. Retropropagación y cálculo del gradiente
   Aplicando la regla de la cadena del cálculo diferencial, se propaga el error desde la salida hacia la entrada y se calcula, para cada parámetro (cada peso y cada sesgo), la derivada parcial de la función de pérdida respecto a ese parámetro. Todas esas derivadas juntas forman el gradiente. Este paso es posible únicamente porque todas las operaciones (producto escalar, suma del sesgo, función diferenciable) son diferenciables.

4. Actualización de parámetros
   Cada peso y sesgo se modifica restando una pequeña fracción de su gradiente respectivo (la fracción la determina la «tasa de aprendizaje», un valor fijado de antemano).
   peso_nuevo = peso_viejo − (tasa_aprendizaje × gradiente_del_peso)
   Con ello, la máquina reduce su error para ese ejemplo.

5. Iteración
   Se repite el ciclo con otra muestra del conjunto de datos, millones de veces. Progresivamente, los parámetros se estabilizan en valores que minimizan el error medio sobre el conjunto de compilación.

Al finalizar, los parámetros quedan fijados y la máquina está lista para ser ejecutada (lo que el marketing llama *«inferencia»*). En ejecución, solo se realiza el paso 1: propagación hacia adelante, sin modificar nada.

Analogía formal: El mecanismo de una cerradura de caja fuerte

Para hacer tangible la idea, comparemos la VMM con un sistema mecánico: la cerradura de combinación de una caja fuerte. Esta comparación es formal: establece una correspondencia estructural entre los elementos de ambos sistemas.

Componente en la VMM (término de Tesio)	Función técnica real	Componente análogo en la cerradura	Por qué es una analogía precisa
Vector de entrada	Lista numérica que representa los datos a procesar (p.ej., píxeles)	La secuencia de números que se introduce en la cerradura (la combinación)	Ambos son una serie ordenada de valores que inician el proceso.
Parámetros (pesos y sesgos)	Los números que se ajustan durante la compilación	La posición de los discos y levas internas (cada disco tiene una muesca en una ubicación precisa)	Almacenan la «configuración correcta» que determina el comportamiento del sistema.
Producto escalar	Cuantifica la coincidencia entre la entrada y los pesos	El acto de hacer girar los discos con la combinación: cada número hace girar un disco y, si coincide con la muesca, se alinea correctamente. La suma de alineaciones decide si la cerradura se abrirá.	En ambos casos se evalúa si cada elemento de la entrada concuerda con un valor preestablecido.
Función diferenciable	Introduce no linealidad y permite el cálculo de derivadas	El mecanismo de la leva que libera el pestillo solo cuando todos los discos están perfectamente alineados	Transforma un valor intermedio (suma de alineaciones) en una decisión binaria (abrir o no), y pequeñas variaciones en los discos producen cambios graduales en la «probabilidad de apertura».
Gradiente	Vector que indica cómo modificar cada peso y sesgo para reducir el error	La instrucción precisa del cerrajero: «El cuarto disco está girado 2 muescas a la izquierda de su posición correcta; gíralo 2 muescas a la derecha para alinearlo».	El gradiente proporciona, parámetro a parámetro, la dirección y magnitud exacta del ajuste necesario.
Compilación estadística	Ciclo iterativo de ajuste de parámetros basado en miles de ejemplos	El proceso de configuración inicial, donde el cerrajero prueba combinaciones, comprueba si la caja se abre y, disco a disco, va colocando cada muesca en su posición exacta hasta que solo la combinación correcta funciona.	Es una fase de ajuste determinista y reversible (dentro del sistema) que termina cuando el error es mínimo.
Ejecución	Uso de la VMM ya compilada con datos nuevos	Introducir una combinación y comprobar si la caja se abre. No se modifica ningún disco.	Solo se verifica una condición predefinida; no hay «aprendizaje» ni adaptación.

Tabla de traducción: antropomorfismo vs. realidad técnica

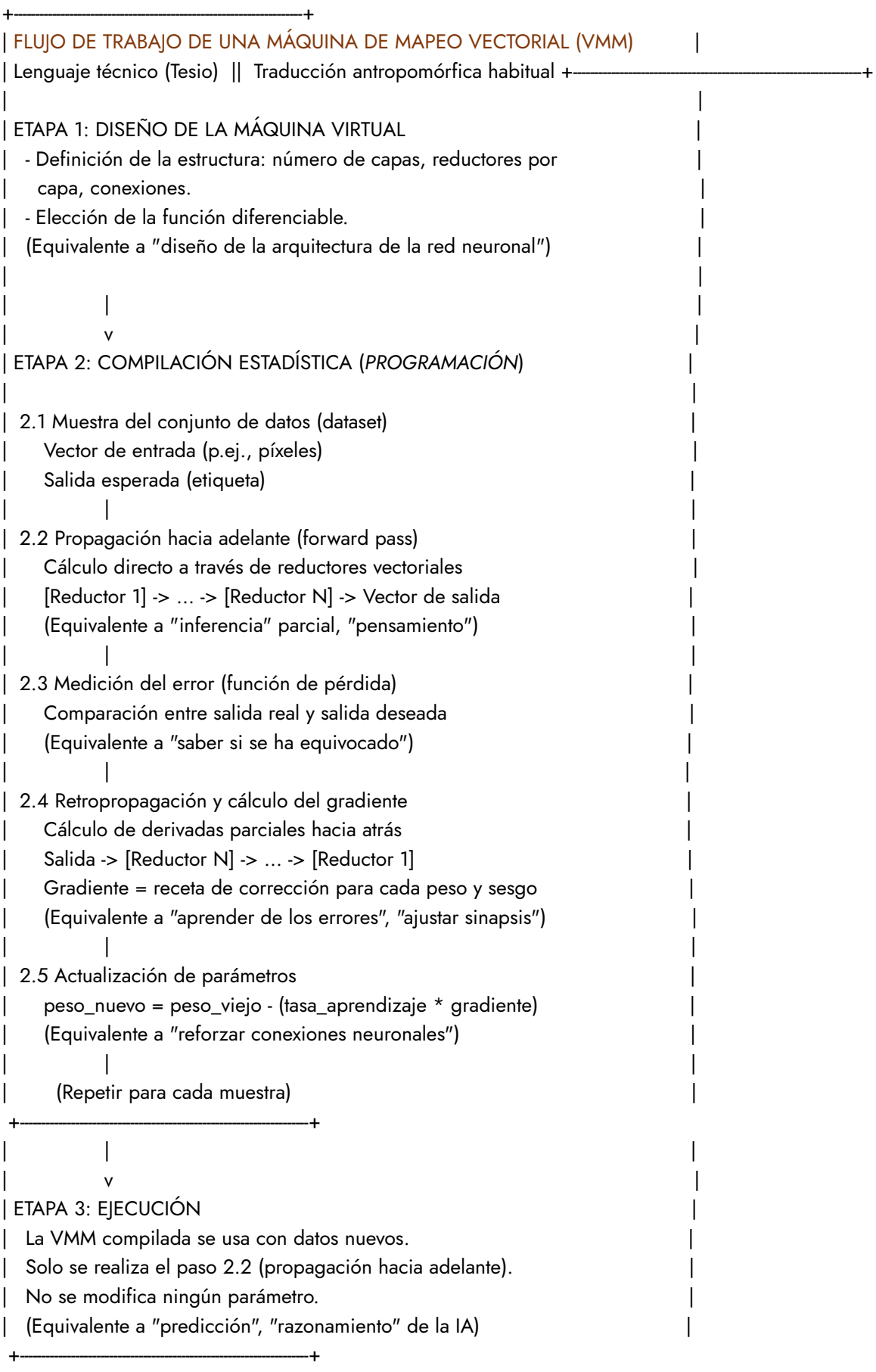
- Término antropomórfico (marketing)	+ Término de Tesio (realidad técnica)	¿Qué ocurre realmente?
Inteligencia Artificial (IA)	Máquina de Mapeo Vectorial Virtual (VMM)	Software especializado en transformar vectores mediante operaciones matemáticas.
Red neuronal artificial	VMM compuesta por capas de reductores vectoriales	Conjunto de fórmulas encadenadas sin ninguna analogía biológica real.
Neurona	Reductor vectorial	Unidad que realiza producto escalar + sesgo + función diferenciable.
Entrenamiento / Aprendizaje	Programación estadística / Compilación	Ajuste automático de parámetros para minimizar una función de error.
Pesos sinápticos	Parámetros (pesos y sesgos)	Números reales que multiplican las entradas.
Modelo	Software / Ejecutable derivado de los datos	Archivo binario que contiene los parámetros ajustados y las instrucciones de cálculo.
Inferencia / Predicción	Ejecución / Propagación hacia adelante	Aplicación de las fórmulas ya fijadas a datos nuevos, sin modificación alguna.
Hiperparámetros	Configuración de la máquina virtual	Decisiones de diseño (número de capas, función de activación) tomadas por un ingeniero.

Conclusión: lo humano en el centro

La desmitificación de Tesio no niega la potencia de estas máquinas, sino que las sitúa en el lugar que les corresponde: herramientas de procesamiento de datos. La inteligencia y la información no están en la máquina, sino en las personas:

- En quienes diseñan la arquitectura de la VMM.
- En quienes generan los datos que se usan para la compilación estadística (cada imagen etiquetada, cada texto escrito).
- En quienes interpretan los resultados y los convierten de nuevo en información.

La máquina no entiende, no sabe, no razona: lo que hace es ejecutar un programa compilado a partir de datos producidos por humanos. Es una obra derivada, no un autor. Genera datos, no crea información, la información la construye el ser humano como buenamente puede con los datos que esta máquina le «escupa».


FLUJO DE TRABAJO DE UNA MÁQUINA DE MAPEO VECTORIAL (VMM)
Lenguaje técnico (Tesio) || Traducción antropomórfica habitual
ETAPA 1: DISEÑO DE LA MÁQUINA VIRTUAL
- Definición de la estructura: número de capas, reductores por capa, conexiones.
- Elección de la función diferenciable.
(Equivalente a "diseño de la arquitectura de la red neuronal")
ETAPA 2: COMPILACIÓN ESTADÍSTICA (PROGRAMACIÓN)
2.1 Muestra del conjunto de datos (dataset)
Vector de entrada (p.ej., píxeles)
Salida esperada (etiqueta)
2.2 Propagación hacia adelante (forward pass)
Cálculo directo a través de reductores vectoriales
[Reductor 1] -> ... -> [Reductor N] -> Vector de salida
(Equivalente a "inferencia" parcial, "pensamiento")
2.3 Medición del error (función de pérdida)
Comparación entre salida real y salida deseada
(Equivalente a "saber si se ha equivocado")
2.4 Retropropagación y cálculo del gradiente
Cálculo de derivadas parciales hacia atrás
Salida -> [Reductor N] -> ... -> [Reductor 1]
Gradiente = receta de corrección para cada peso y sesgo
(Equivalente a "aprender de los errores", "ajustar sinapsis")
2.5 Actualización de parámetros
peso_nuevo = peso_viejo - (tasa_aprendizaje * gradiente)
(Equivalente a "reforzar conexiones neuronales")
(Repetir para cada muestra)
ETAPA 3: EJECUCIÓN
La VMM compilada se usa con datos nuevos.
Solo se realiza el paso 2.2 (propagación hacia adelante).
No se modifica ningún parámetro.
(Equivalente a "predicción", "razonamiento" de la IA)

Explicación del proceso matemático mediante la analogía formal de la cerradura de combinación

Voy a intentar explicar el proceso matemático completo de compilación estadística de una Máquina de Mapeo Vectorial Virtual (VMM) usando, como analogía formal, el mecanismo de una cerradura de combinación de caja fuerte de alta precisión. No es una metáfora literaria; es una correspondencia estructural entre dos sistemas: uno numérico y otro mecánico. Cada elemento matemático tiene su contrapartida mecánica.

1. El reductor vectorial como un único disco de la cerradura

Matemáticamente, un reductor vectorial es una función que recibe un vector de entrada $x = (x_1, x_2, \ldots, x_n)$ y produce un escalar y. Internamente:

$z = (x_1 \cdot w_1 + x_2 \cdot w_2 + \ldots + x_n \cdot w_n) + b$ (producto escalar + sesgo)

$y = f(z)$ (función diferenciable)

Los números $w = (w_1, w_2, \ldots, w_n)$ son los parámetros (pesos) del reductor.

b es el sesgo.

f es una función no lineal y diferenciable, como la sigmoide: $f(z) = 1 / (1 + e^{-z})$.

Analogía mecánica:
Imaginemos un solo disco dentado dentro de la cerradura. Ese disco tiene una muesca en una posición angular secreta. La combinación que introducimos es una secuencia de números; cada número se traduce en una rotación. Pero, en lugar de una secuencia temporal, nuestro «disco» recibe simultáneamente un vector de muescas (el vector de entrada) que describen la profundidad de varios dientes de una llave.

Para un solo disco, lo que realmente importa es una única posición angular; sin embargo, el modelo de reductor vectorial es más general: el disco dispone de n agujas o sensores, cada una con una altura de referencia w_i.

El plegado de dimensiones en la cerradura
Cuando el cerrajero diseña un disco, no le basta con comparar un solo diente de la llave. La llave real puede tener muchos dientes (por ejemplo, 784). Cada uno de esos dientes es una "dimensión" distinta de la llave. El disco, sin embargo, solo puede producir un único valor de alineación (¿cuánto ha girado la leva?).

Para lograrlo, el disco realiza un producto escalar: multiplica cada diente de la llave por una referencia interna (la altura de una aguja) y suma todos esos productos. Esa suma es un solo número que pliega toda la información de los 784 dientes en una única medida de coincidencia. Si los dientes se parecen a las referencias, el número es grande; si no, pequeño o negativo.

A eso llamamos plegado: transformar un *punto* de muchas dimensiones (la llave completa) en un *punto* de una sola dimensión (el grado de alineación del disco). Es decir, pliega la información de ese punto concreto, reduciendo su dimensionalidad. Luego, el sesgo desplaza ese valor y la leva (función diferenciable) lo suaviza. Y sí, ya se que por definición un punto no tiene dimensiones, pero en este contexto analógico creo que se entiende.

Durante la compilación estadística (ajuste de la cerradura), el cerrajero itera millones de veces: introduce una llave, cada disco pliega su parte, y al final mide el error. El gradiente le dice cuánto debe girar cada aguja (cada peso) para que el plegado sea más preciso. En cada iteración, los pliegues se van perfeccionando.

En la ejecución, la cerradura ya está fijada. Cuando introduces una llave nueva, cada disco aplica su plegado sin modificar nada. No hay nuevo aprendizaje, solo se pliega la información paso a paso.

La «llave» de entrada trae sus propias alturas `x_i`. El producto escalar `x·w` mide cuánto coinciden perfil de la llave y perfil de referencia del disco. Matemáticamente:

Si `x_i` y `w_i` tienen el mismo signo y magnitudes parecidas, el producto escalar es grande.

Si son muy diferentes, es pequeño (incluso negativo).

El sesgo `b` desplaza el punto de decisión: es como añadir una cuña fija que hace que, para que el disco considere que la llave es válida, el producto escalar deba superar cierto umbral.

Finalmente, la función diferenciable `f` convierte la alineación total `z` en una medida acotada, por ejemplo entre 0 (totalmente bloqueado) y 1 (totalmente libre). La sigmoide modela un mecanismo de leva: cuando `z` es muy negativo, la leva está fuertemente bloqueada (salida ≈ 0); cuando `z` es muy positivo, la leva se libera casi por completo (salida ≈ 1); en la zona intermedia hay una transición suave. La suavidad es crucial: podemos saber exactamente cómo un pequeño cambio en `z` afecta a la salida, y por tanto cómo un pequeño giro de los discos modifica la probabilidad de apertura.

2. Varios discos en cascada (capas de reductores)

Una VMM se construye apilando muchos reductores, organizados en capas. La salida de un reductor puede ser la entrada de varios reductores de la capa siguiente. Matemáticamente, la capa `l` toma un vector de entrada `a^(l-1)` (de dimensión `m`) y produce un vector de salida `a^(l)` (de dimensión `n`). Para cada reductor `j` de esa capa:

$z_j^{(l)} = (a^{(l-1)} \cdot w_j^{(l)}) + b_j^{(l)}$

$a_j^{(l)} = f(z_j^{(l)})$

Los parámetros de la capa son la matriz `W^(l)` (cuyas filas son los vectores de pesos de cada reductor) y el vector de sesgos `b^(l)`.

Analogía:
Ahora la cerradura tiene varias ruedas (discos) acopladas. La primera rueda recibe directamente la llave (el vector de entrada de la VMM). Su salida —la «posición de su leva»— se convierte en parte de la llave que se presenta a la segunda rueda, y así sucesivamente. El estado final de la última rueda determina si el pestillo se retrae (vector de salida).
Esta concatenación permite condiciones muy complejas: para que la cerradura se abra, deben cumplirse simultáneamente muchas condiciones distribuidas por todas las ruedas, y las condiciones se construyen a partir de combinaciones no lineales de las señales previas.

3. La función de pérdida: medir cuánto falta para abrir la caja

Durante la compilación, disponemos de un conjunto de ejemplos: para cada llave (vector de entrada `x`) sabemos si la caja debería abrirse o no (o, en una clasificación con varias cajas, cuál es la caja correcta). La salida deseada se representa como un vector `y_true`. La máquina produce una salida `y_pred = a^(L)` (capa final).

Una función de pérdida `L(y_pred, y_true)` cuantifica la discrepancia. Por ejemplo, si la salida es un único número entre 0 y 1 (probabilidad de que la caja se abra) y la etiqueta es 1 (debe abrirse), la pérdida podría ser:

L = -(y_true · log(y_pred) + (1 - y_true) · log(1 - y_pred))

(entropía cruzada binaria). Si `y_true=1` y `y_pred=0.8`, la pérdida es `-log(0.8) ≈ 0.223`. Si `y_pred=0.2`, la pérdida es mucho mayor: `-log(0.2) ≈ 1.609`.

Analogía:
El cerrajero dispone de un torquímetro que mide la fuerza que el mecanismo opone a la apertura. Si la combinación es correcta, la resistencia debe ser casi nula (pérdida baja). Si la combinación es incorrecta, la resistencia es alta (pérdida alta). La función de pérdida es una lectura numérica de esa resistencia. El objetivo de la compilación es ajustar los discos (parámetros) para que, para todas las llaves de entrenamiento, la resistencia sea mínima cuando y solo cuando la llave es la correcta.

4. Retropropagación y gradiente: la receta de ajuste

El gradiente indica, para cada parámetro (cada peso `w` y cada sesgo `b` de cada reductor), en qué dirección y cuánto debe modificarse para reducir la pérdida. Se calcula mediante la retropropagación del error, aplicando la regla de la cadena del cálculo diferencial desde la salida hasta la entrada.

Partimos de la pérdida `L`. El primer paso difiere ligeramente para la capa de salida y las capas ocultas:

> Para la capa de salida (capa L):
> Calculamos directamente `δ^(L)`, la derivada de la pérdida respecto a `z^(L)` (los valores antes de aplicar la función de activación en la última capa). Se obtiene multiplicando la derivada de la pérdida respecto a la activación de salida por la derivada de la función de activación, elemento a elemento (producto de Hadamard, denotado con ⊙):
>
> δ^(L) = (∂L/∂a^(L)) ⊙ f′(z^(L))
>
> Para las capas anteriores (l = L-1, L-2, …, 1):
> Obtenemos `δ^(l)` propagando el δ de la capa siguiente hacia atrás, utilizando la matriz de pesos transpuesta de la capa `l+1` y multiplicando por la derivada de la función de activación de la capa actual:
>
> δ^(l) = ((W^(l+1))ᵗ · δ^(l+1)) ⊙ f′(z^(l))
>
> Aquí `(W^(l+1))ᵗ` es la matriz de pesos de la capa `l+1` traspuesta, de modo que las conexiones se recorren en sentido inverso y las contribuciones de cada reductor vectorial («*neurona*») de la capa siguiente se distribuyen correctamente hacia las reductores vectoriales de la capa actual.

Una vez conocido `δ^(l)`, las derivadas respecto a los pesos y sesgos de esa capa son:

∂L/∂w_ij^(l) = a_i^(l-1) · δ_j^(l)

∂L/∂b_j^(l) = δ_j^(l)

Estas derivadas son los componentes del gradiente.

Analogía paso a paso con la cerradura (tres discos en serie y una leva final):
Hemos introducido una llave; la leva final se ha movido solo un 20% (salida 0.2), pero debería haberse movido al 100% (etiqueta 1). La pérdida es alta.

Paso 1 (capa de salida): El torquímetro nos da la derivada ∂L/∂a (cuánto cambia la pérdida al variar la posición de la leva). La leva actual está en 0.2 y queremos llevarla a 1.0. Aplicamos `f'(z^(3))` (la sensibilidad de la leva en ese punto) para obtener `δ^(3)`, es decir, cuánto debe cambiar la señal que recibe la leva (`z^(3)`) para que la pérdida disminuya.

Paso 2 (propagación al disco anterior): Para ajustar los pesos que conectan el disco 2 con el disco 3, necesitamos saber cómo contribuye cada salida del disco 2 (`a_i^(2)`) al error. El `δ^(2)` se calcula distribuyendo el `δ^(3)` hacia atrás a través de los pesos del disco 3 traspuestos (es decir, cada peso que conectaba una salida del disco 2 con una entrada del disco 3 ahora lleva la responsabilidad del error en sentido inverso) y multiplicando por `f'(z^(2))`, la sensibilidad del disco 2.

Paso 3 (cálculo de gradientes en cada disco): Para una conexión entre la salida `a_i^(2)` y el disco 3, su gradiente es `a_i^(2) * δ_j^(3)`. Si el diente `a_i^(2)` es grande y `δ_j^(3)` es positivo (necesitamos aumentar `z_j^(3)`), la instrucción será girar el disco 3 para aumentar el peso de ese diente. Si el diente es casi cero, su peso no se modifica significativamente.

Paso 4 (iteración hacia atrás): Se repite el proceso para el disco 2 y luego para el disco 1, propagando los δ con las transpuestas de las matrices de pesos y las derivadas de las funciones de activación.

Al final del recorrido, disponemos de una lista completa de ajustes micrométricos para cada disco: el gradiente. Por ejemplo: «Al disco 3, giro de +0.05 muescas en su cuarto sensor. Al disco 2, giro de -0.12 muescas en su primer sensor...».

5. Actualización de parámetros: el cerrajero ejecuta los ajustes

Una vez calculado el gradiente para una llave (o un pequeño lote de llaves), se actualizan todos los discos:

$w_{nuevo} = w_{viejo} - \eta \cdot (\partial L/\partial w)$

$b_{nuevo} = b_{viejo} - \eta \cdot (\partial L/\partial b)$

donde η (*tasa de aprendizaje*) es un número pequeño, por ejemplo 0.01.

Analogía: El cerrajero no gira cada disco de golpe hasta la posición perfecta, porque esa posición perfecta no la conoce todavía (depende de todas las demás). En lugar de eso, para cada llave de entrenamiento, aplica la corrección que el gradiente le dicta, pero solo una pequeña fracción (η). Va dando pequeños toques a los discos, llave tras llave. Con miles de ejemplos, los discos convergen a una configuración que minimiza la resistencia para todas las llaves correctas y la maximiza para las incorrectas.

6. Fin de la compilación y ejecución

Cuando el error medio ya no disminuye de manera significativa, la compilación termina. Los parámetros quedan fijados: la cerradura está completamente configurada. A partir de ese momento, la ejecución (*inferencia*) consiste simplemente en introducir una llave, dejar que el mecanismo de discos y levas opere (propagación hacia

adelante) y medir la posición final de la leva. No se modifica ningún disco; la máquina no «*aprende*» de los nuevos datos, solo aplica su programa precompilado.

Resumen de la correspondencia matemático-mecánica

Concepto matemático	Función en la VMM	Pieza mecánica análoga
Vector de entrada x	Datos a procesar	Llave con perfil de dientes (muescas)
Parámetros W, b	Pesos y sesgos	Posiciones de referencia de las agujas de cada disco
Producto escalar x·w	Alineación entre datos y referencia	Medida de coincidencia entre perfil de la llave y perfil del disco
Sesgo b	Desplazamiento del umbral	Cuña fija que predispone el bloqueo/liberación
Función diferenciable f(z)	No linealidad y suavidad	Leva de transición suave, que permite calcular el efecto de pequeños giros
Capa de reductores	Transformación vector a vector	Conjunto de discos cuyas levas forman la llave para la siguiente etapa
Función de pérdida L	Medición del error	Torquímetro que mide resistencia a la apertura
Gradiente ∂L/∂w	Receta de ajuste por parámetro	Instrucciones exactas: «gira este disco tantas muescas»
Retropropagación con $\delta^{(l)}$ y transpuesta de W	Asignación de responsabilidad hacia atrás	El cerrajero recorre los discos en orden inverso, anotando cuánto contribuye cada uno al error final y usando un esquema de distribución simétrico (las conexiones se evalúan al revés)
Actualización $w \leftarrow w - \eta \cdot \nabla L$	Ajuste iterativo	Pequeños toques del cerrajero sobre cada disco
Compilación completa	Parámetros fijados	Cerradura configurada, lista para ser usada

La operación es íntegramente determinista y se reduce a la aplicación repetida de la regla de la cadena sobre una función compuesta diferenciable. Desde la perspectiva de Tesio, la máquina no alberga inteligencia: es un programa que transforma datos y cuyos parámetros han sido compilados a partir de datos generados por humanos. La inteligencia permanece en las personas que diseñaron la máquina, en las perosnas que etiquetaron los datos y en la persona que interpreta la salida.

Aunque el proceso matemático descrito es determinista, conviene matizar que en la práctica se introducen deliberadamente elementos estocásticos y que, además, pueden observarse comportamientos emergentes que no estaban programados explícitamente. Ninguno de estos matices contradice el marco conceptual de Tesio, pero añaden una precisión que considero necesaria.

En primer lugar, la compilación estadística incorpora aleatoriedad controlada: los parámetros iniciales (pesos y sesgos) suelen generarse aleatoriamente para romper simetrías; las muestras del conjunto de datos se barajan e incluso se presentan en lotes aleatorios; y en ocasiones se emplean técnicas como el *dropout*, que desactiva aleatoriamente reductores durante la compilación. Esta estocasticidad no convierte a la máquina en no determinista (los mecanismos de generación pseudoaleatoria son deterministas dada una semilla), pero sí introduce variabilidad en el resultado final.

En segundo lugar, tras la compilación pueden emerger comportamientos no previstos explícitamente: la interacción no lineal de miles de millones de parámetros produce patrones de transformación vectorial que nadie programó directamente. Son los llamados fenómenos emergentes. En la analogía de la cerradura, sería como si, tras configurar minuciosamente los discos para que solo ciertas llaves abran la caja, descubriéramos que algunas llaves nuevas, que comparten rasgos estadísticos con las originales pero que nunca vimos durante la configuración, también logran abrirla parcialmente. Esa generalización no fue programada disco a disco; emergió de la combinación global de ajustes. Estos fenómenos, sin embargo, no añaden voluntad, conciencia ni comprensión: simplemente reflejan la capacidad de la máquina para extrapolar patrones estadísticos a partir de los datos de compilación, un resultado que sigue dependiendo por completo de la estructura diseñada por humanos y de los datos humanos usados para programarla.

Los errores de esta interpretación son míos, de Aitor Saiz Lasheras, y no tienen nada que ver con la interpretación de Giacomo Tesio. Puedo estar confundido y probablemente lo esté, así que les recomiendo encarecidamente que estudien las fuentes, hagan sus propias interpretaciones y comentan sus propios errores. Yo espero que alguien corrija los míos, a ser posible.

Fuentes, a mayo de 2026:

https://encrypted.tesio.it/2019/06/03/what-is-informatics.html

https://cuestiondemetodo.com/bildung/ia-desmitificada-articulo-de-g-tesio/