Apéndice: Herramientas conceptuales para comprender la imposibilidad de la seguridad total en IA

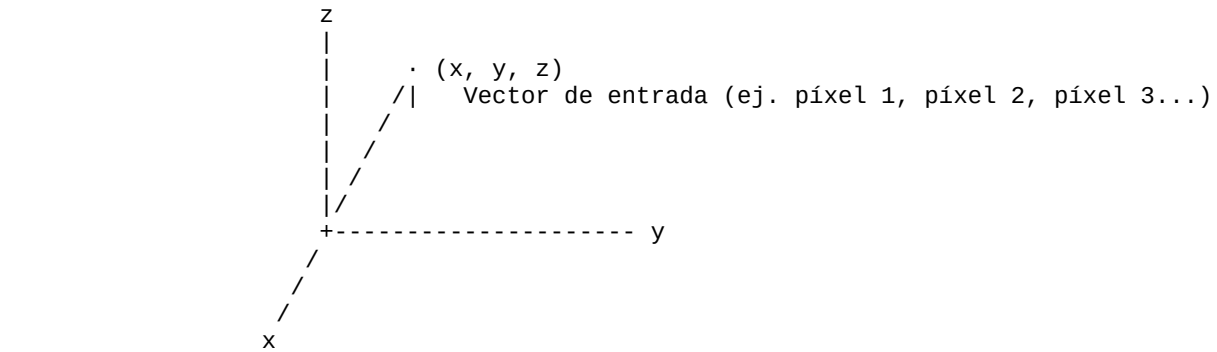
La seguridad total de los modelos de IA es inalcanzable por razones matemáticas, no por falta de pericia. Para entender el argumento central, se requieren unas pocas nociones precisas que aquí se presentan de forma autocontenida y rigurosa.

1. Vectores y dimensiones: el espacio donde viven los datos

Cualquier dato que un modelo procesa –una imagen, un fragmento de texto, un registro médico– se convierte en un vector: una lista ordenada de números. Una imagen en escala de grises de 28×28 píxeles, por ejemplo, es un vector con 784 componentes (una por píxel). El conjunto de todos los vectores posibles de ese tamaño forma un espacio de 784 dimensiones.

Una dimensión no es más que una dirección independiente en ese espacio. En un espacio de alta dimensionalidad (los modelos modernos operan con millones o miles de millones de dimensiones), la cantidad de direcciones mutuamente perpendiculares es enorme. Esta alta dimensionalidad no es un simple detalle técnico: es el sustrato geométrico donde germinan las vulnerabilidades estructurales.

Esquema 1: Un vector de entrada como punto en un espacio tridimensional (representación reducida)



Cada eje es una dimensión independiente.
En un espacio real de 784 dimensiones, cada dirección añade un eje más.

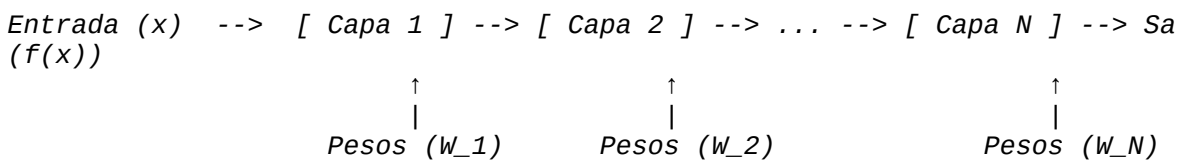
2. El modelo como función diferenciable: pesos, entradas y salidas

Un modelo de aprendizaje profundo es, matemáticamente, una función que toma un vector de entrada y produce un vector de salida (por ejemplo, probabilidades para cada clase). Internamente, la función se construye encadenando operaciones simples (sumas, multiplicaciones, activaciones) que dependen de un conjunto de parámetros llamados pesos.

Lo esencial es que esta función es diferenciable (o lo es a trozos, como cuando se usa ReLU). La diferenciabilidad permite calcular cómo varía la salida al modificar infinitesimalmente la entrada o

los pesos. Durante el entrenamiento, los pesos se ajustan para minimizar un error utilizando precisamente esa información de variación.

Esquema 2: El modelo como función



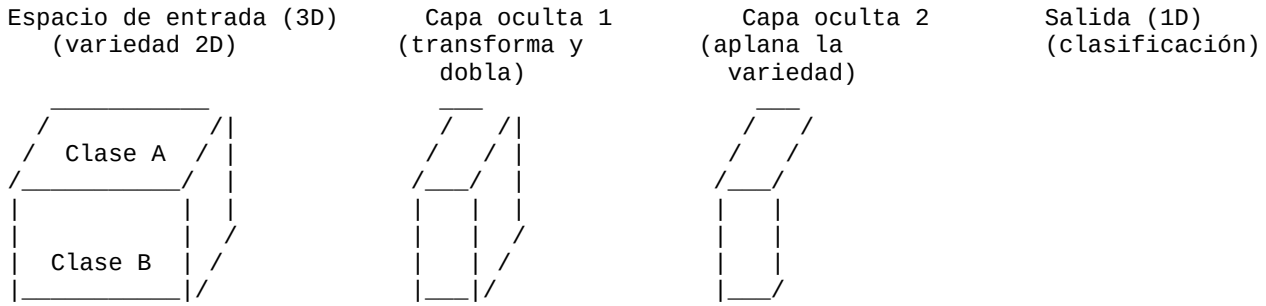
f es diferenciable respecto a la entrada x y a los pesos W.
Esto permite calcular el gradiente de la salida (o del error) respecto a cada píxel de entrada.

3. Cómo las funciones del modelo pliegan las dimensiones del espacio

Aquí reside la clave del peligro. Un modelo de clasificación transforma vectores de entrada de altísima dimensión en vectores de salida de dimensión mucho menor (típicamente el número de clases). Para lograrlo sin perder la información relevante, la red explota una hipótesis fundamental: los datos de entrada no ocupan todo el espacio RDRD, sino que viven en una variedad (*manifold*) de dimensión intrínseca baja, incrustada en el espacio de alta dimensión. Por ejemplo, todas las imágenes naturales de dígitos manuscritos forman una estructura geométrica de dimensión mucho menor que 784, plegada y curvada sobre sí misma, con solo unas pocas direcciones intrínsecas de variación (la forma del trazo, la inclinación, el grosor).

La función del modelo, mediante sucesivas capas de operaciones lineales y no lineales, pliega, estira y aplana esa variedad hasta situarla en un espacio de dimensión baja donde las clases queden bien separadas. Cada capa aplica una transformación afín (multiplicación por una matriz de pesos y suma de un sesgo) seguida de una función de activación no lineal (como ReLU o sigmoide). La no linealidad es la que permite "doblar" el espacio: sin ella, todo el proceso se reduciría a una simple transformación lineal que no puede separar clases que no lo estuvieran ya.

Esquema 3: Plegado de la variedad de datos (conceptual)



La variedad original (izquierda) es una superficie de dos dimensiones sumergida en 3D.
La primera capa oculta introduce pliegues que acercan puntos de la misma clase.
Las capas sucesivas aplanan la estructura hasta que, en el espacio de salida (derecha), la variedad se ha proyectado a una línea recta sobre la que se puede trazar una frontera lineal entre clases.

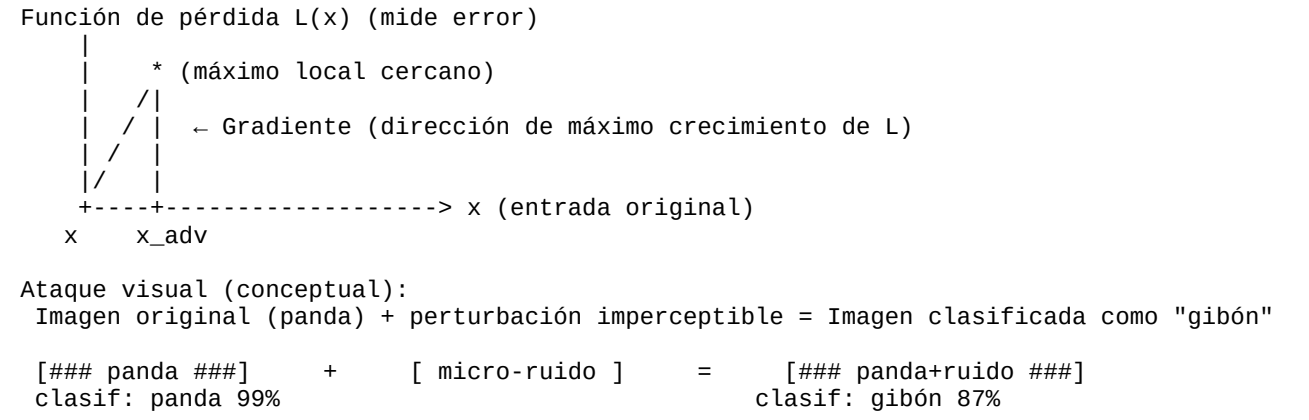
Lo crucial es que, durante este plegado, la red ignora las direcciones del espacio de entrada que son ortogonales a la variedad de datos. Dos vectores son ortogonales cuando su producto escalar es cero: forman un ángulo de 90 grados y no comparten ninguna componente en la dirección del otro. Esto implica que moverse a lo largo de una dirección ortogonal no produce ningún desplazamiento a lo largo de la variedad. Las direcciones ortogonales, por tanto, "salen" de la variedad y apuntan hacia regiones del espacio que no contienen ejemplos naturales. Como en esas direcciones no hay señal de supervisión durante el entrenamiento, la función aprendida se vuelve prácticamente constante (o muy plana) a lo largo de ellas.

Esta es la razón de la eficacia del aprendizaje profundo, pero también de su fragilidad: las direcciones ignoradas constituyen el subespacio adversarial. Una perturbación pequeña en esas direcciones apenas afecta a la apariencia del dato (sigue pareciendo legítimo a un humano) porque no modifica su proyección sobre la variedad, pero puede arrastrar el punto fuera de ella y hacerle cruzar las fronteras de decisión que la red trazó en el espacio plegado de baja dimensión.

4. Gradiente: la dirección de máxima sensibilidad

El gradiente de una función en un punto indica la dirección en la que la función crece más rápidamente. Si tenemos una función de pérdida que mide el error del modelo para una entrada concreta, el gradiente de esa pérdida respecto a la entrada nos dice qué pequeña perturbación en los píxeles aumentaría más el error.

Esquema 4: Gradiente y ataque adversarial en una entrada



La perturbación adversarial más simple (método *Fast Gradient Sign Method*, FGSM, Goodfellow et al., 2015) se construye como:

$$x_{adv} = x + \varepsilon \cdot \text{sign}(\nabla_x L(x, y_{true}))$$

donde ε es un escalar pequeño que controla la magnitud del ruido, y la función signsign se aplica componente a componente.

> Nota: Es crucial recordar que $\nabla_x L$ es el gradiente (un vector de la misma dimensión que x), no el valor de la pérdida $L(x)$ (un escalar). Sumar un escalar a un vector ($x + \varepsilon L(x)$) no tiene ningún significado matemático en este contexto.

5. Subespacio adversarial: no puntos aislados, sino toda una región de fallo

El hallazgo fundamental no es que existan ejemplos adversariales como puntos aislados. Lo preocupante es que esos vectores de ataque no son independientes: forman un subespacio vectorial.

Un subespacio es un conjunto de vectores que cumple dos propiedades fundamentales:

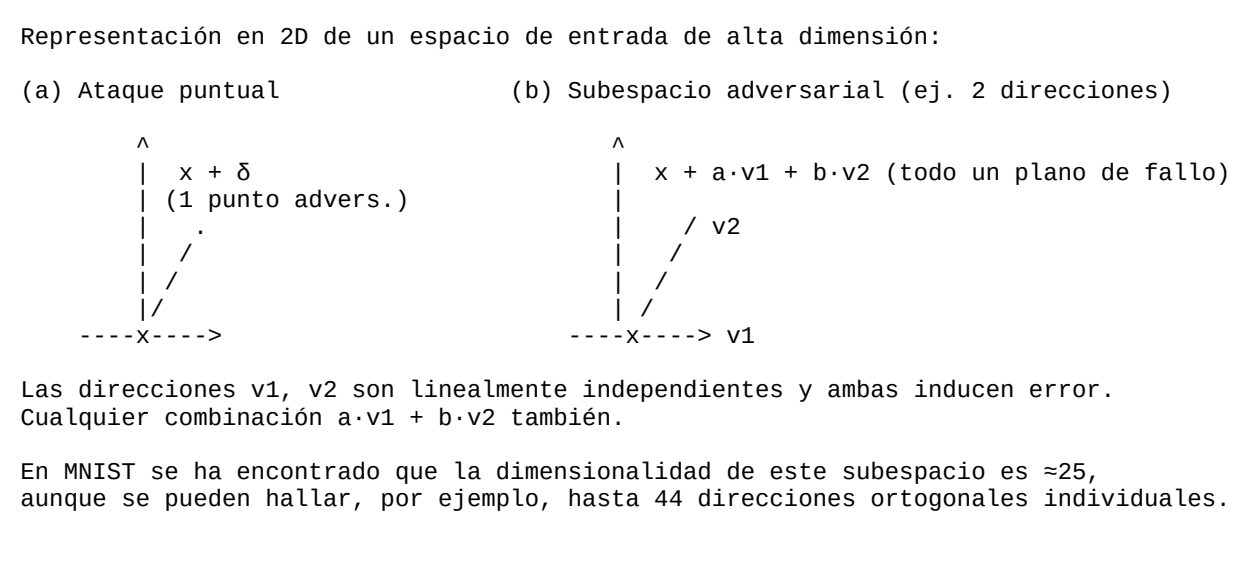
1. Clausura bajo la suma: Si tomas dos vectores del conjunto, su suma sigue en el conjunto.
2. Clausura bajo la multiplicación por un escalar: Si tomas un vector y lo multiplicas por cualquier número real (que puede estirarlo, encogerlo o invertir su dirección), el resultado también pertenece al conjunto.
3. Contiene al vector cero: El origen del espacio siempre forma parte del subespacio.

En términos prácticos, esto implica que si se descubre una dirección adversarial (un vector que provoca error) y otra dirección distinta que también lo es, cualquier combinación de ambas construida sumándolas o multiplicándolas por números reales cualesquiera seguirá siendo adversarial. Como consecuencia, no estamos ante puntos de fallo aislados: existe todo un subespacio de puntos en el espacio de entrada donde el modelo se equivoca con alta probabilidad.

Geometría del fenómeno: En un espacio de entrada de alta dimensión, los ejemplos adversariales no son puntos aislados, sino que forman un subespacio contiguo de dimensionalidad considerable. Según Tramèr et al. (2017):

> La dimensión de este subespacio de ejemplos adversariales es de aproximadamente 25.
>
> Aunque se pueden encontrar, por ejemplo, hasta 44 direcciones ortogonales individuales para un solo punto de datos, es la dimensionalidad de ≈25 la que define la verdadera extensión del espacio de ataque.
>
> Subespacios con mayor dimensionalidad son más propensos a intersecarse, lo que explica por qué los ataques son transferibles entre modelos.

Esquema 5: De puntos aislados a un subespacio adversarial



Es importante señalar que, aunque el *volumen* de este subespacio es cero en el espacio de entrada (tiene medida de Lebesgue nula), un atacante puede navegar por él de forma deliberada, generando ataques diversos sin apenas esfuerzo.

6. ¿Por qué no se puede "parchear" este subespacio? El dilema del gradiente puntiagudo

Uno podría pensar que el entrenamiento puede eliminar esas direcciones de vulnerabilidad. Para ello, el modelo tendría que volverse extremadamente insensible a variaciones en esas direcciones, es decir, tendría que presentar un gradiente de pérdida casi nulo donde ahora es aprovechable. Pero esto tiene un coste devastador para la generalización.

Un modelo que es prácticamente constante en un subespacio entero de dimensiones significativas (≈25 en MNIST) estaría, en la práctica, obligado a ignorar información relevante de la entrada, o bien a generar una frontera de decisión increíblemente abrupta (un gradiente "muy puntiagudo"). Ambas opciones degradan su capacidad de clasificar ejemplos nuevos no vistos durante el entrenamiento. La vulnerabilidad adversarial y la capacidad de generalizar parecen ser dos caras de una misma moneda geométrica.

7. Transferibilidad: la geometría compartida de los modelos

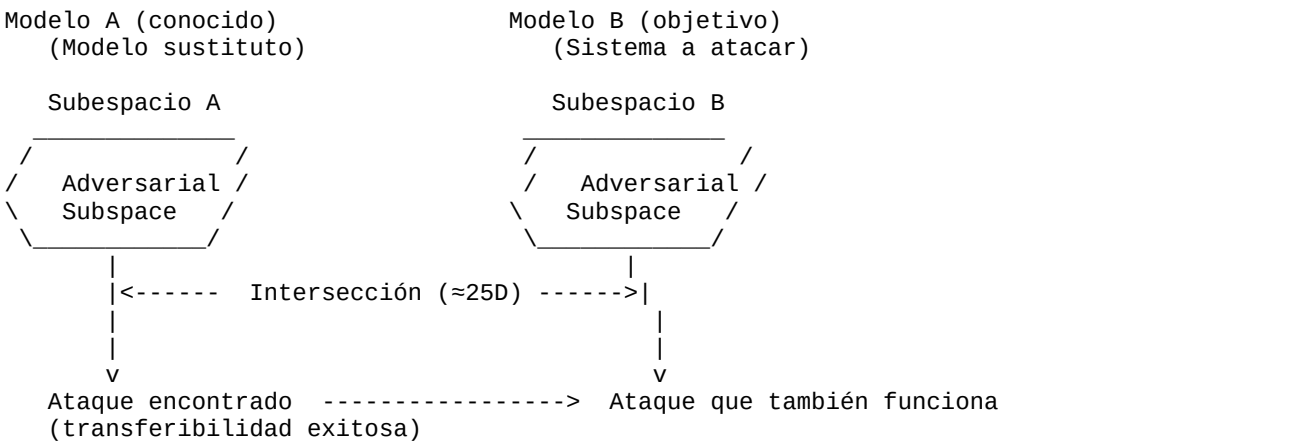
Dos modelos distintos, entrenados para la misma tarea con arquitecturas y datos similares, acaban generando fronteras de decisión muy próximas entre sí en el espacio de entrada. Esto implica que una dirección adversarial encontrada para el modelo A (fácil de interrogar, por ejemplo uno de código abierto) tiene una alta probabilidad de ser también efectiva contra el modelo B (el objetivo real).

Esta transferibilidad no es casual y tiene límites bien definidos. Tramèr et al. (2017) derivaron formalmente:

1. Condiciones suficientes sobre la distribución de los datos que implican transferibilidad para clases de modelos simples.
2. Ejemplos de escenarios en los que la transferencia no ocurre.

Más aún, estudios recientes han cuantificado que la transferibilidad está estrechamente relacionada con la similitud geométrica entre los modelos. En concreto, el grado de solapamiento de los subespacios adversariales puede predecirse mediante métricas como la *Centered Kernel Alignment* (CKA). Esto significa que un atacante puede seleccionar estratégicamente modelos sustitutos que maximicen la probabilidad de éxito del ataque transferido.

Esquema 6: Transferibilidad entre dos modelos



El atacante no necesita acceso al modelo defendido; puede explorar el subespacio adversarial de un modelo sustituto y lanzar el ataque con éxito con alta probabilidad.

8. Intratabilidad computacional: la imposibilidad de una verificación exhaustiva

Eliminar las vulnerabilidades exigiría, como mínimo, verificar que ninguna de las infinitas direcciones en el espacio de entrada conduce a un error. Pero el espacio de posibles modelos sustitutos es igualmente infinito, y la búsqueda de subespacios adversariales en todos ellos es una tarea computacionalmente intratable.

No se trata solo de que hoy no tengamos suficiente potencia de cálculo: se ha demostrado que verificar la robustez de una red neuronal general con activaciones ReLU es un problema NP-completo. Problemas de esta clase no admiten una solución eficiente general (en tiempo polinómico). En esencia, la dificultad de encontrar y eliminar ataques no puede ser erradicada porque anida en la propia complejidad computacional de la función que el modelo implementa.

9. Dónde reside el peligro adversarial, en síntesis

El peligro no procede de un error de código ni de un conjunto finito de puntos que se puedan corregir. Procede de que los clasificadores útiles, al plegar la variedad de datos para reducir la dimensionalidad y separar clases, ignoran inevitablemente un número enorme de direcciones perpendiculares. Esas direcciones ignoradas constituyen un subespacio adversarial de dimensión significativa (por ejemplo, ~25 en el caso de MNIST) que:

- es casi invisible para un muestreo aleatorio (no aparece por casualidad),
- puede ser descubierto y explotado con técnicas de gradiente,
- se transfiere entre modelos (con mayor o menor éxito dependiendo de su similitud geométrica),

y no puede ser eliminado sin inutilizar la capacidad de generalización o sin afrontar una tarea de verificación computacionalmente intratable.

La seguridad total es una imposibilidad matemática porque el problema no reside en los detalles de implementación, sino en la geometría del espacio de alta dimensión y en la naturaleza diferenciable de los modelos que aprenden a partir de datos. Entender estos conceptos no elimina el riesgo, pero permite dimensionarlo y, sobre todo, desconfiar de cualquier promesa de protección absoluta.