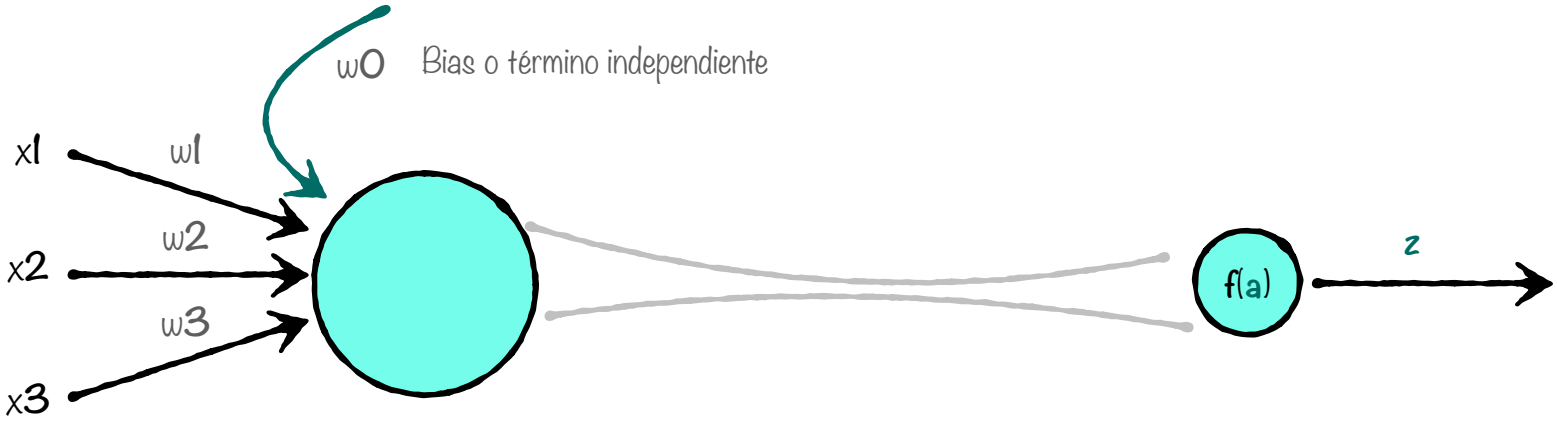
Estructura de **una neurona**



La neurona es entonces una **combinación lineal**

$$a = w0 + x1 \cdot w1 + x2 \cdot w2 + x3 \cdot w3$$

Dado que en general trabajamos con **vectores**, podemos expresar la combinación lineal así:

$$a = w^T x$$

Donde:

$$x = [1, x1, x2, ..., xn]^T$$
$$w = [w0, w1, w2, ..., wn]^T$$

La colita de la neurona calibra la salida y se llama la **función de activación**

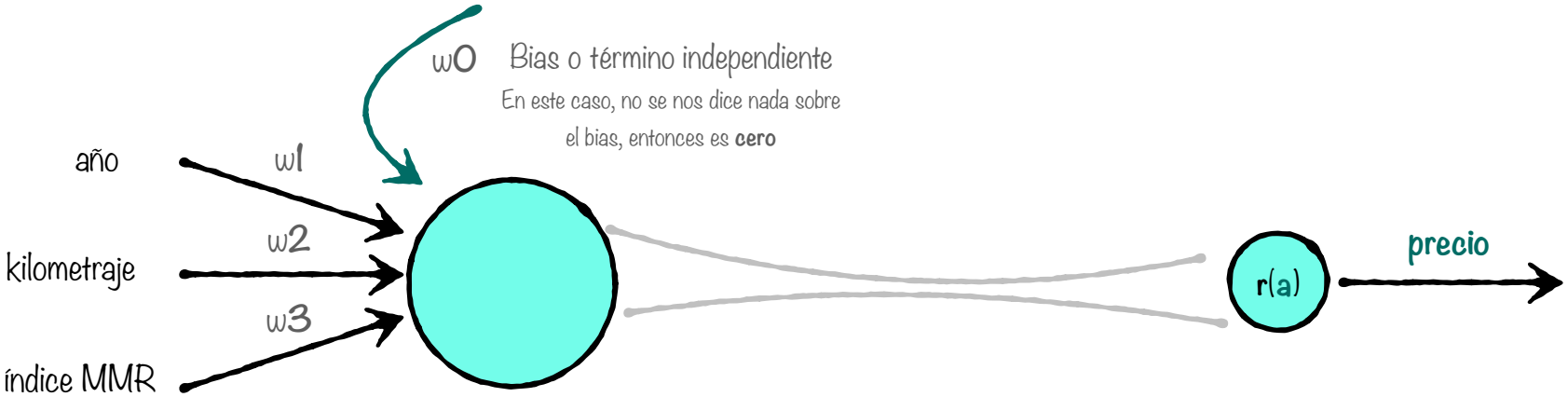
$$z = f(a)$$

Depende de la tarea. Por ejemplo, para la regresión ayuda **reLu**, para la clasificación binaria ayuda **sigmoide**. Cuando se tiene una clasificación multi-clase, ayuda **softmax** que es una generalización de la función sigmoide retornando probabilidades.

¿Para qué sirve **una neurona**?

Una neurona se puede entrenar para resolver un sistema de ecuaciones de **n** variables; donde **n** corresponde al número de entradas de la neurona. La diferencia entre una neurona y un modelo simple de regresión lineal está en la función de activación —que no está presente en la regresión lineal—.

Ejemplo: Predecir el precio de un carro a partir de su año de construcción, su kilometraje y el índice MMR.



La neurona es entonces una **combinación lineal**

$a = w0 + año \cdot w1 + kilometraje \cdot w2 + MMR \cdot w3$

La colita de la neurona calibra la salida y se llama la **función de activación**

$precio = \mathbf{reLu}(a)$

La tarea ahora es entrenar la neurona. Es decir, encontrar $w0$, $w1$, $w2$ y $w3$ de manera que el error en la estimación sea mínimo

¿Para qué sirve **una neurona**?

Para entrenar la neurona necesitamos un conjunto de datos de entrenamiento —en este ejemplo estamos en el contexto de machine learning **supervisado**.

año	kilometraje	índice MMR	precio
2015	16,639	20,500	21,500 USD
2015	9,393	20,800	21,500 USD
2014	1,331	31,900	30,000 USD
2015	14,282	27,500	27,750 USD
2015	2,641	66,000	67,000 USD
...	...	...	...

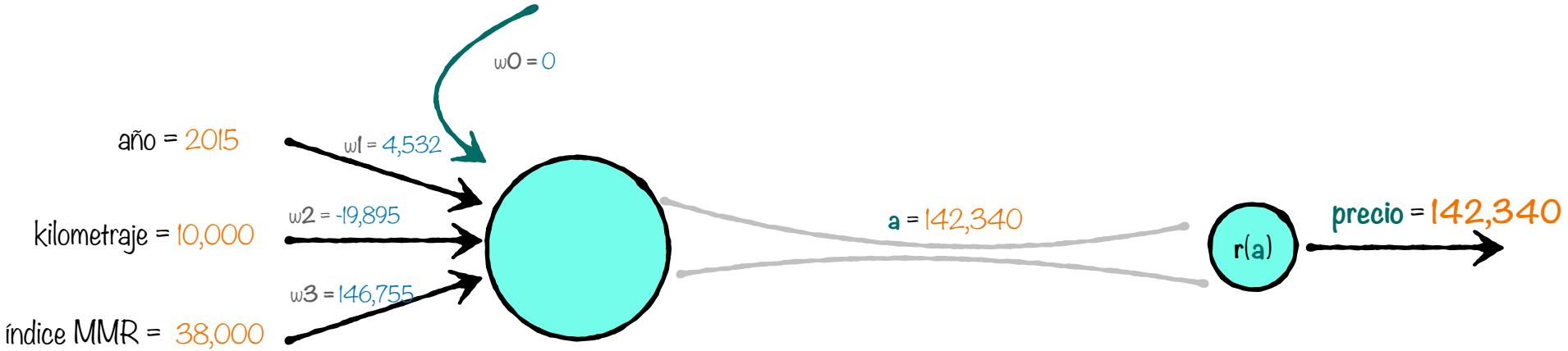
El entrenamiento de la neurona usualmente se hace mediante el algoritmo de **descenso de gradiente**.

— Generar valores aleatorios para w0, w1, w2 y w3.
— Calcule el error.
— Derive la función de error en el punto actual, es decir en los valores actuales de w0, w1, w2 y w3
— Actualice los valores de w0, w1, w2 y w3 proponiendo nuevos valores en dirección de la derivada direccional que encontró en en paso anterior.
— Itere hasta que el criterio de salida se cumpla.

Note que hay **hiperparámetros** que se deben refinar. **Ejemplo**: taza de aprendizaje.

¿Para qué sirve **una neurona**?

Al ejecutar el algoritmo de descenso de gradiente, encontramos los valores para w0, w1, w2 y w3 y entonces podemos usar la neurona (entrenada) para predecir el valor de un carro a partir de sus características.



La neurona es entonces una **combinación lineal**

a = 0 + 4,532•año - 19,895•kilometraje + 146,755•mmr

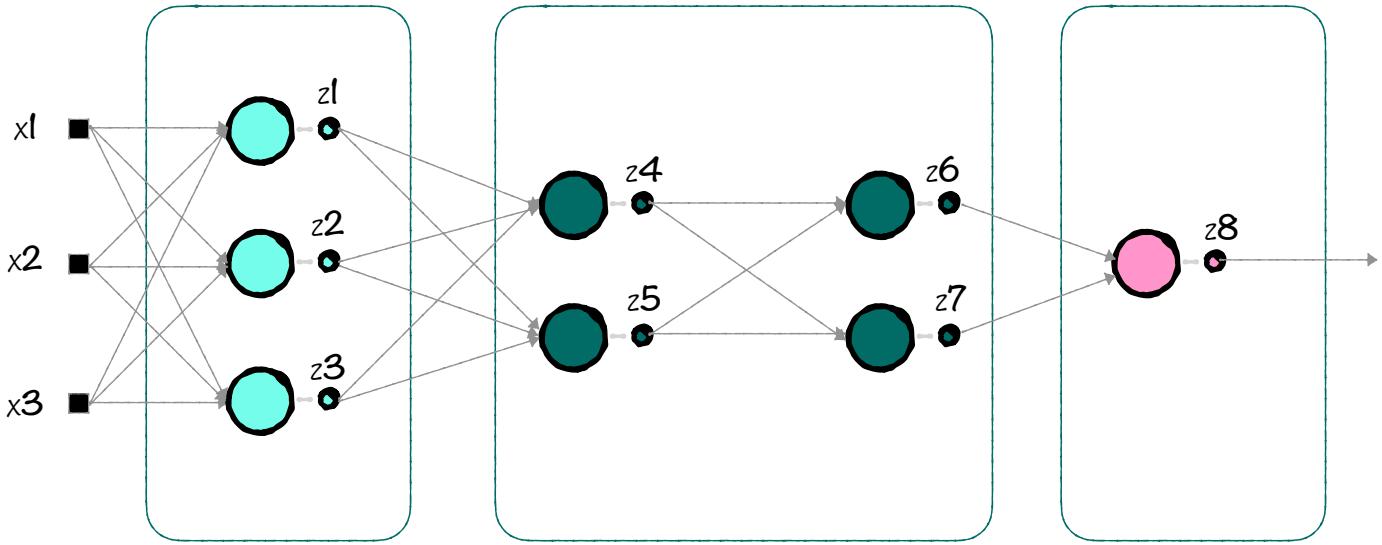
La colita de la neurona calibra la salida y se llama la **función de activación**

precio = **reLu**(**a**)

- Los valores de las variables independientes deben racionalizarse para que descenso gradiente funcione correctamente.
- Los valores de a tienen sentido como predicciones (no son necesariamente buenas, pero tienen sentido dado que están en el orden de magnitud correcto).
- El resultado luego de pasar la función de activación tiene sentido como predicción si se usa la función de activación apropiada.

Estructura de **red neuronal**

Arquitectura en capas (una de las más utilizadas)



Capa de entrada

Tiene tantas neuronas como variables independientes

Capas ocultas/escondidas

Capa de salida

Una neurona para el caso de la regresión y tantas neuronas como clases para el caso de la clasificación

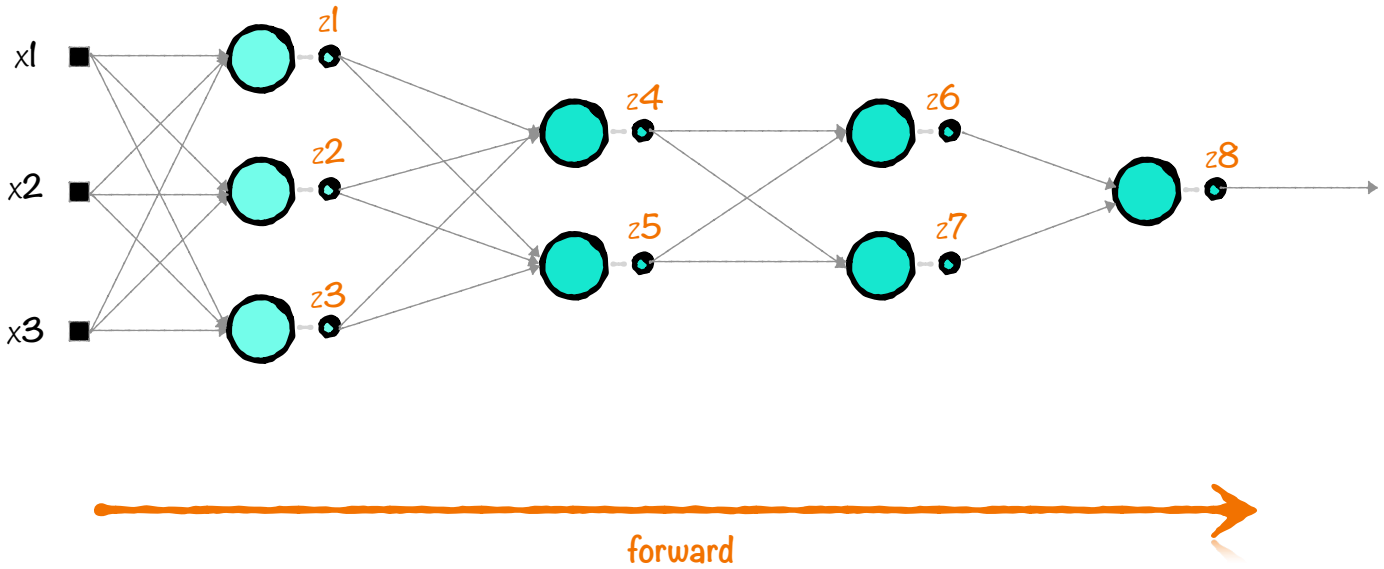
$z8 = L(w, x)$

Función paramétrica donde los parámetros son todos los pesos de la red neuronal

Entrenamiento de una **red neuronal**

Se usa el algoritmo de descenso de gradiente estocástico donde:

- **Primero se calculan los zetas en un paso "forward".**
- Luego se calculan los gradientes propagándolos de atrás hacia adelante. i.e., back-propagation



Entrenamiento de una **red neuronal**

Se usa el algoritmo de descenso de gradiente estocástico donde:

- Primero se calculan los zetas en un paso "forward".
- **Luego se calculan los gradientes propagándolos de atrás hacia adelante. i.e., back-propagation**