

Redes neuronales

Dra. Ximena Gutierrez Vasques & Mtro. Víctor Mijangos

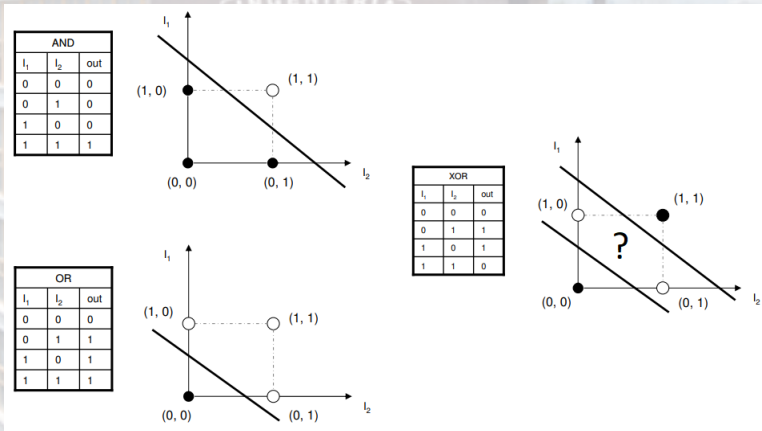
Facultad de Ingeniería

13-17 de enero, 2020



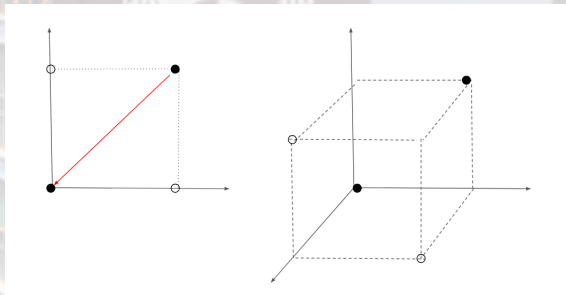
Problemas con el perceptrón

El perceptrón es un método de separación lineal. Por tanto su capacidad es limitada.



¿Cómo separar un problema XOR?

La idea es modificar los datos para que estos sean separables por un plano o hiperplano. En otras palabras, buscamos transformar los datos. Esto se puede hacer de distintas formas.



Solución al problema XOR

El problema XOR puede resolverse por la composición de diferentes perceptrones.

En **términos lógicos** el problema XOR es la composición de problemas lógicos básicos.

En **términos geométricos** se transforman los datos a un espacio donde sean linealmente separables.

Se pueden utilizar **transformaciones lineales**; sin embargo éstas **son insuficientes** en muchos casos.



¿Cómo separar un problema XOR?

Por ejemplo, se puede tomar la transformación lineal:

$$L^T = \begin{pmatrix} 1 & 0 \\ -1 & -1 \end{pmatrix}$$

Esta transformación hace que los datos sean

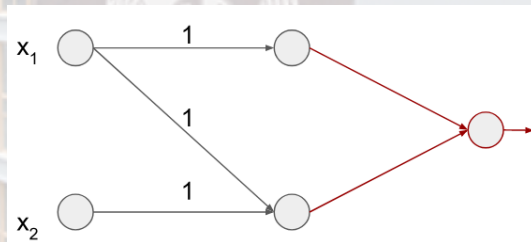
$$XL = \begin{pmatrix} 0 & 0 \\ 0 & 1 \\ 1 & 0 \\ 1 & 1 \end{pmatrix} \begin{pmatrix} 1 & -1 \\ 0 & -1 \end{pmatrix} = \begin{pmatrix} 0 & 0 \\ 0 & -1 \\ 1 & -1 \\ 1 & 0 \end{pmatrix}, Y = \begin{pmatrix} 0 \\ 1 \\ 1 \\ 0 \end{pmatrix}$$

Que se puede resolver con un perceptrón con parámetros $w = (0 \quad -1)$, $b = -1$



Incorporación de la transformación en la red neuronal

La transformación L se puede expresar por medio de un grafo dirigido, particularmente como un perceptrón.

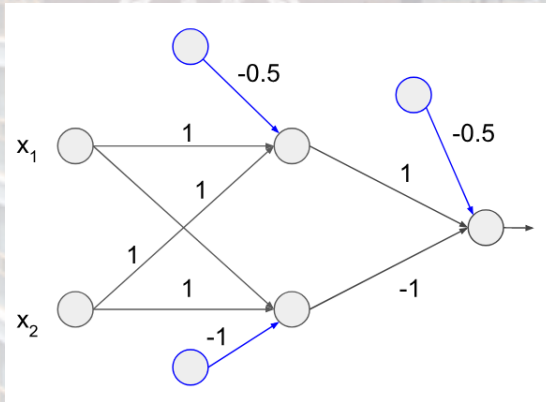


La parte roja representa el perceptrón que trabaja con los datos transformados.



Capas ocultas

Más aún, se puede pensar en transformaciones que no sean meramente lineales. Así, los nodos que preseden al “perceptrón de salida” o capa de salida pueden formalizarse como un perceptrón, con bias y función de activación.



Capas ocultas

Los nodos intermedios en esta estructura conforman una capa oculta. Por tanto, podemos pensar una red neuronal con una capa oculta a partir de las funciones:

$$h(x) = g(W^{(1)}x + b^{(1)}) \quad (1)$$

$$f(x) = \phi(W^{(2)}x + b^{(2)}) \quad (2)$$

Donde $W^{(1)} \in \mathbb{R}^{d \times m}$, $b^{(1)} \in \mathbb{R}^m$, $W^{(2)} \in \mathbb{R}^{m \times o}$ y $b^{(2)} \in \mathbb{R}^o$ (o es el número de salidas). Además, g y ϕ son funciones de activación (se busca que sean **derivables**).



Redes neuronales pre-alimentadas

Una red neuronal pre-alimentada (o *feedforward NN*) es una red de este tipo con K capas ocultas. Cuenta con:

1. Una pre-activación para cada capa $0 < k \leq K$, dada por:

$$a^k(x) = W^{(k)}h^{(k-1)}(x) + b^{(k)}$$

2. Una función de activación para cada capa $0 < k \leq K$:

$$h^k(x) = g(a^k(x))$$

3. Una activación de salida:

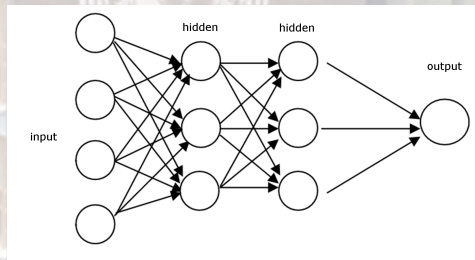
$$f(x) = \phi[W^{(K+1)}h^{(K)}(x) + b^{(K+1)}]$$



Redes neuronales pre-alimentadas

Nótese que $W^{(1)}$ es una matriz de $d \times m_1$, $W_{(K+1)}$ es una matriz de $m_K \times o$, y para todo $0 < k \leq K$ $W^{(k)}$ es una matriz de $m_{k-1} \times m_k$.

La arquitectura de una red neuronal pre-alimentada especifica el número de capas ocultas y las unidades de cada una (K y m_1, \dots, m_K) [3].



Funciones de activación

Para las capas ocultas, se pueden determinar diferentes funciones de activación g :

- ▶ Sigmoides:

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

- ▶ Tangente hiperbólica:

$$\tanh(a) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$

- ▶ Softplus:

$$S(a) = \log(1 + e^x)$$

- ▶ Rectified Linear Unit:

$$\text{ReLU}(a) = \max\{0, a\}$$



Funciones de activación

Estas funciones de activación son comunes porque son fácilmente derivables. Sus **derivadas** son:

- ▶ Sigmoide:

$$\sigma'(a) = \sigma(a)(1 - \sigma(a))$$

- ▶ Tangente hiperbólica:

$$\tanh'(a) = 1 - \tanh^2(a)$$

- ▶ Softplus:

$$S(a) = \sigma(a)$$

- ▶ Rectified Linear Unit:

$$\text{ReLU}(a) = \begin{cases} 1 & \text{si } a > 0 \\ 0 & \text{si } a < 0 \end{cases}$$



Función Softmax

Si la salida es una sola unidad, se puede usar el sigmoide. Si son varias unidades, suele usarse la función Softmax:

$$\text{Softmax}(a_j) = \frac{e^{a_j}}{\sum_{i=1}^o e^{a_i}}$$

La función Softmax define una función de probabilidad sobre las unidades de salida. Para la clasificación en las clases de salida, tenemos que

$$\hat{y} = \arg \max_j \{\text{Softmax}(a_j)\}$$



¿Por qué nos gustan tanto las redes neuronales?

Las redes neuronales pre-alimentadas son una herramienta de gran importancia, pues permiten aproximar funciones $f : \mathbb{R}^d \rightarrow \mathbb{R}^o$.

Teorema (Aproximador universal [1])

Una red neuronal f con m unidades ocultas (una sola capa) puede aproximar tanto como se quiera cualquier función Borel-medible.

El problema al que nos enfrentamos es elegir la arquitectura adecuada para nuestro problema y, más aún, tener una muestra suficientemente rica para lograr la aproximación.



Capacidad de una red FeedForward

La capacidad, en términos de **dimensión VC**, de una red es difícil de determinar. La capacidad de una red neuronal depende de varios factores:

1. Número de capas ocultas
2. Número de unidades ocultas por capa
3. Hiperparámetros: funciones de activación, función de riesgo, rango de aprendizaje, método de optimización...

Podemos formular el siguiente resultado [2]:

Teorema

La dimensión VC de una red neuronal es $O(|\theta| \log |\theta|)$, donde θ son los parámetros de la red.



Relevancia de las redes neuronales

Algunas **ventajas** del uso de redes son:

- ▶ Una red con una sola capa puede aproximar adecuadamente cualquier función de interés.
- ▶ Son capaces de encontrar representaciones abstractas que permitan una clasificación adecuada.
- ▶ Alta flexibilidad, permiten un buen número de arquitecturas.
- ▶ Requieren poco conocimiento del problema.

Algunas **desventajas** son:

- ▶ Requieren una buena cantidad de datos para converger adecuadamente.
- ▶ Requieren poder de procesamiento alto. Muchas veces poco accesible.
- ▶ Son poco interpretables.



Flexibilidad de las redes neuronales

Las redes FeedForward son una arquitectura simple.

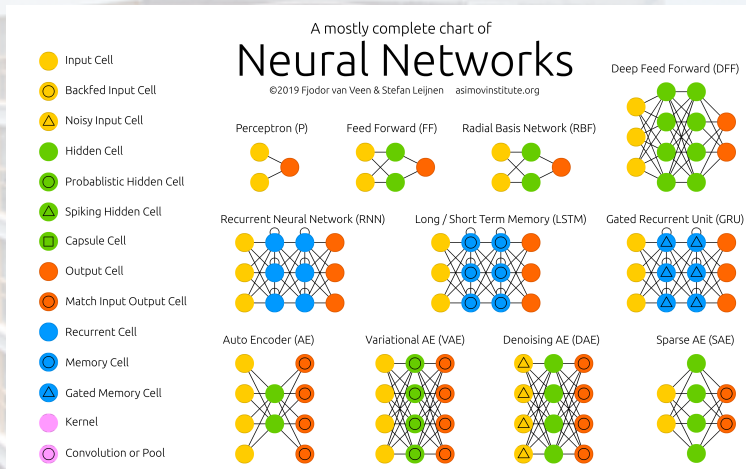
Pero éstas pueden verse como una particularidad de las **redes reccurentes**.

Asimismo, existen diferentes formas de conceptualizar una red, lo que determina diferentes algoritmos basados en redes neuronales.



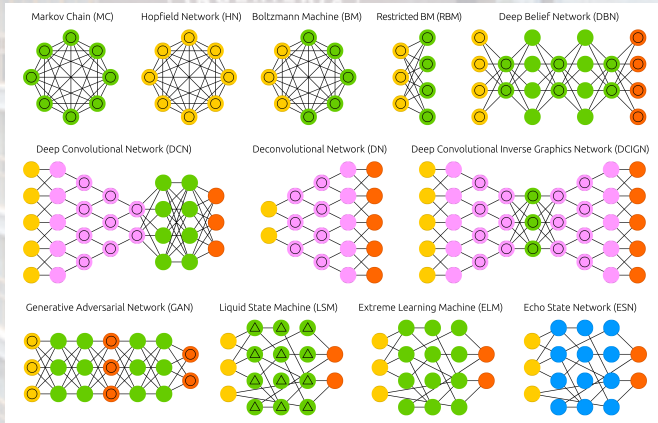
Familia de redes neuronales

Algunas arquitecturas conocidas son:



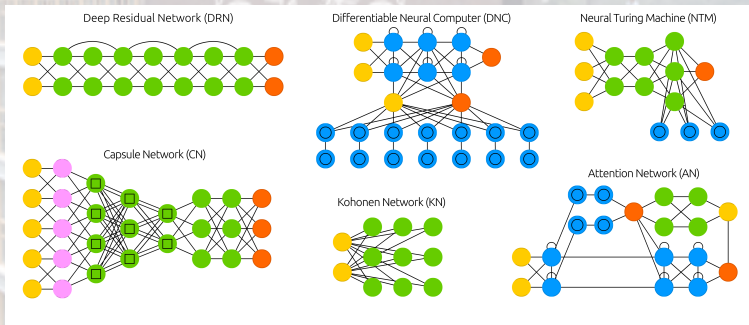
Familia de redes neuronales

Otras arquitecturas son:



Familia de redes neuronales

Otras arquitecturas son:



Ingredientes de una red neuronal

Para construir una red neuronal, entonces, se requiere:

- ▶ Un conjunto de datos de entrada X .
- ▶ Determinar un tipo de red neuronal que se acople al problema.
- ▶ Una función de salida ϕ (Softmax, sigmoide, escalón, ...).
- ▶ Una arquitectura $\{K, m_1, \dots, m_K\}$ (tal que $|\theta| = \sum_{k=1}^K m_{k-1} \cdot m_k + \sum_{k=1}^K m_k$).
- ▶ Funciones de activación.
- ▶ Una función de riesgo $R(\theta)$.



Ejemplo de red neuronal

Supóngase que se tiene un conjunto de datos supervisados:

$$\mathcal{S} = \{(x, y) : x \in \mathbb{R}^5, y \in \{1, 2, 3\}\}$$

Buscamos clasificar los ejemplos en 3 clases.

Suponemos que los datos **no son linealmente separables**. Por tanto, requerimos de una red **FeedForward**.



Arquitectura de la red

Definiremos nuestra red con las siguientes propiedades:

1. Una capa oculta con 10 unidades. Esto define una matriz $W^{(1)} \in \mathbb{R}^{10 \times 5}$ y un bias $b^{(1)} \in \mathbb{R}^{10}$.
2. Una función de activación en la capa oculta \tanh .
3. Una función de salida dada por Softmax (por tanto, se trata de una probabilidad de clase).

La salida de la red contará con 3 nodos (uno para cada clase) determinados por:

$$\phi_y(x) = p(Y = y|x)$$



Implementación de la red neuronal

Dado un ejemplo de entrada x , la red calculará:

- Capa oculta ($\mathbb{R}^5 \rightarrow \mathbb{R}^{10}$):

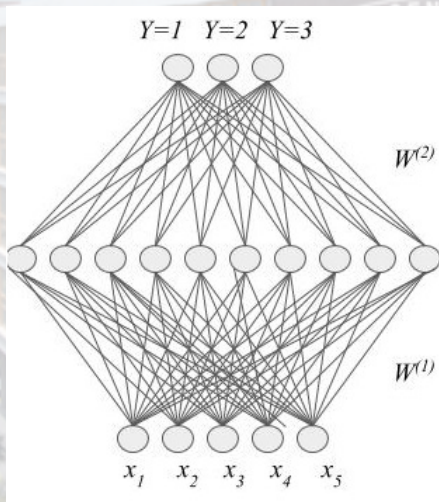
$$h = \tanh(W^{(1)}x + b^{(1)})$$

- Pre-activación de la capa de salida (para una sola clase):

$$a_y = W_y^{(2)}h + b_y^{(2)}$$

- Activación en la capa de salida (probabilidad de una clase):

$$\phi_y(x) = \frac{e^{a_y}}{\sum_{y'} e^{a_{y'}}}$$



Implementación de la red neuronal

Finalmente, se determinará que x pertenece a una clase a partir de la regla:

$$\hat{y} = \arg \max_y \phi_y(x)$$

A todo este paso se le conoce como **Forward**.

La red así constituida tiene un **número de parámetros**

$$|\theta| = (5 \times 10 + 10) + (10 \times 3) + 3 = (50 + 10) + (30 + 3) = 83$$




El problema de aprendizaje consiste en encontrar los parámetros óptimos:

$$\theta = \{W^{(1)}, W^{(2)}, b^{(1)}, b^{(2)}\}$$

¿Cómo se puede realizar esto en redes de este tipo?



References

-  Kurt Hornik, Maxwell Stinchcombe, and Halbert White.
Multilayer feedforward networks are universal approximators.
Neural networks, 2(5):359–366, 1989.
-  Shai Shalev-Shwartz and Shai Ben-David.
Understanding machine learning: From theory to algorithms.
Cambridge university press, 2014.
-  Hao Shen.
A differential topological view of challenges in learning with feedforward neural networks.
arXiv preprint arXiv:1811.10304, 2018.



The End

