Aprendizaje profundo

Redes densas (perceptrón multicapa)

Gibran Fuentes-Pineda

Agosto 2025

Aproximación de la compuerta lógica XOR ($\oplus$)

- Minsky y Papert mostraron que no era posible aproximar una compuerta lógica XOR con una sola neurona con función de activación escalón unitario

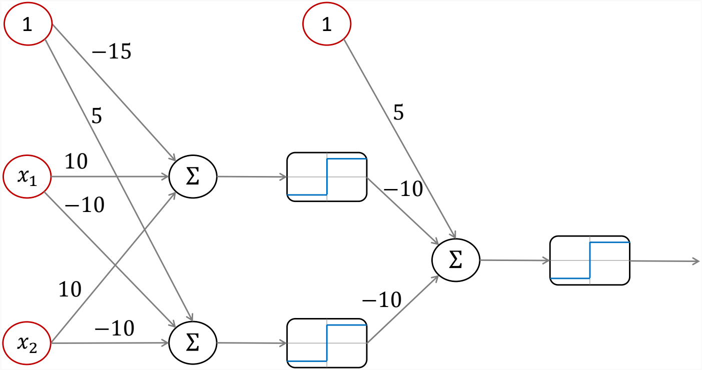
x_1	x_2	$x_1 \oplus x_2$
0	0	0
1	0	1
0	1	1
1	1	0

- Es aproximarla con múltiples neuronas conectadas entre sí (ver `https://playground.tensorflow.org/`).

Múltiples neuronas en capas

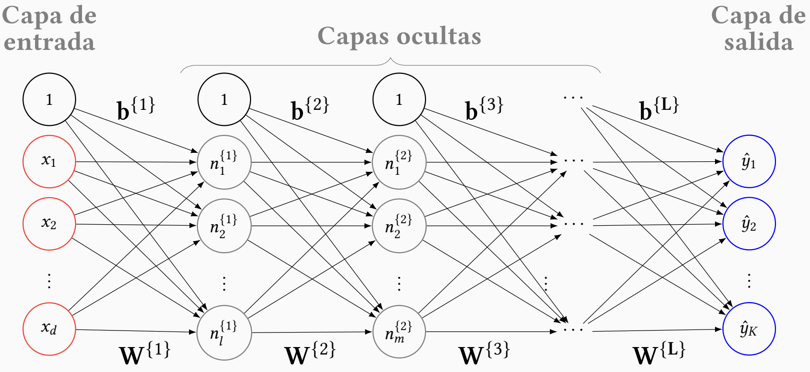
- Podemos usar la fórmula

$$x_1 \oplus x_2 = \neg((x_1 \wedge x_2) \vee \neg(x_1 \vee x_2))$$

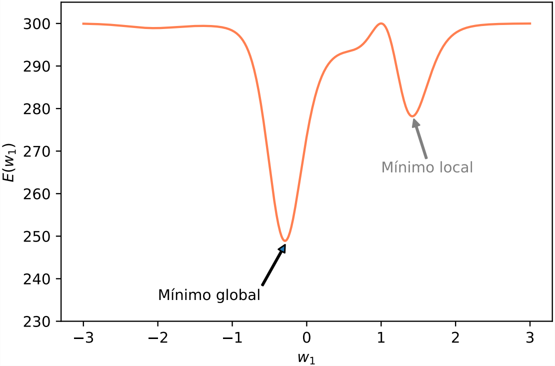


Red neuronal densa

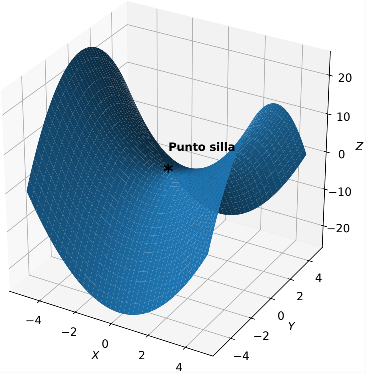
- Formada por capas densas o completamente conectadas
- Pueden modelar problemas no lineales
  - Función de activación de capas ocultas debe tener no linealidades; si no sería equivalente a tener una sola capa



- Para múltiples capas de neuronas la función de pérdida no es convexa (ver `https://losslandscape.com/explorer`)



- Pueden existir varios puntos sillas



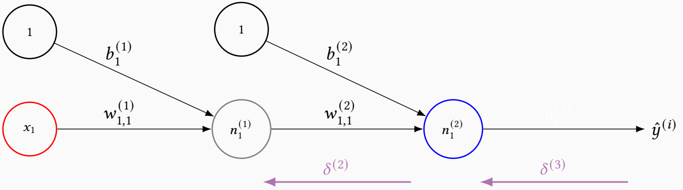
- Aunque en problemas convexos SGD aproxima al GD, en la práctica se ha observado que en el entrenamiento de redes neuronales SGD encuentra mejores soluciones, especialmente con minilotes pequeños[1,2,3]
- Problema: calcular eficientemente las derivadas parciales respecto a los pesos y sesgos de las capas ocultas

[1] Kleinberg et al. An Alternative View: When Does SGD Escape Local Minima?, 2018

[2] Zhu et al. The Anisotropic Noise in Stochastic Gradient Descent: Its Behavior of Escaping from Sharp Minima and Regularization Effects, 2019.

[3] Keskar et al. On Large-Batch Training for Deep Learning: Generalization Gap and Sharp Minima, 2017.

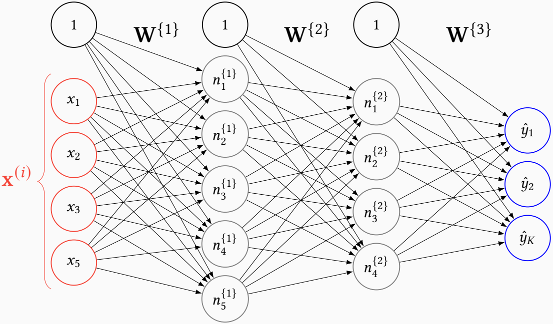
$$a_1^{(1)} = \sigma(\overbrace{w_{1,1}^{(1)} * x_1^{(i)} + b_1^{(1)}}^{z_1^{(1)}}) \qquad a_1^{(2)} = \overbrace{w_{1,1}^{(2)} * a_1^{(1)} + b_1^{(2)}}^{z_1^{(2)}}$$

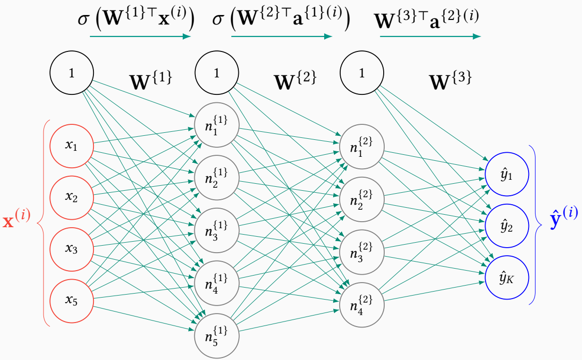


$$\frac{\partial^1\!/\!_2\left[\hat{y}^{(i)}-y^{(i)}\right]^2}{\partial w_{1,1}^{(2)}} = \overbrace{\frac{\partial^1\!/\!_2 u^2}{\partial u}\cdot\frac{\partial u}{\partial a_1^{(2)}}\cdot\frac{\partial a_1^{(2)}}{\partial z_1^{(2)}}}^{\delta^{\{3\}}=\hat{y}^{(i)}-y^{(i)}}\cdot\frac{\partial z_1^{(2)}}{\partial w_{1,1}^{(2)}}$$

$$\frac{\partial^1\!/\!_2\left[\hat{y}^{(i)}-y^{(i)}\right]^2}{\partial w_{1,1}^{(1)}} = \overbrace{\overbrace{\frac{\partial^1\!/\!_2 u^2}{\partial u}\cdot\frac{\partial u}{\partial a_1^{(2)}}\cdot\frac{\partial a_1^{(2)}}{\partial z_1^{(2)}}}^{\delta^{\{3\}}=\hat{y}^{(i)}-y^{(i)}}\cdot\frac{\partial z_1^{(2)}}{\partial a_1^{(1)}}\cdot\frac{\partial a_1^{(1)}}{\partial z_1^{(1)}}}^{\delta^{\{2\}}=(\hat{y}^{(i)}-y^{(i)})\cdot w_{1,1}^{(2)}\cdot(\sigma(z_1^{(1)})(1-\sigma(z_1^{(1)})))}\cdot\frac{\partial z_1^{(1)}}{\partial w_{1,1}^{(1)}}$$

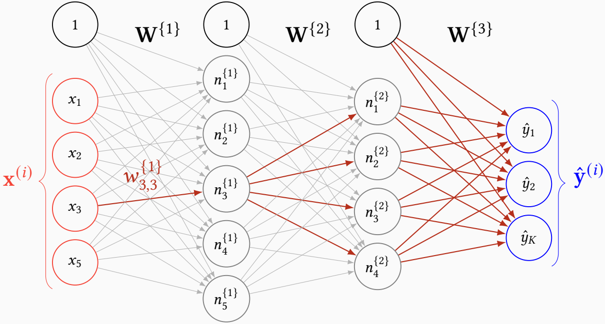
- Considera una red densa con 1 capa de entrada, 2 capas ocultas con 5 y 4 neuronas con función de activación sigmoide y 3 neuronas de salida con función de activación lineal.





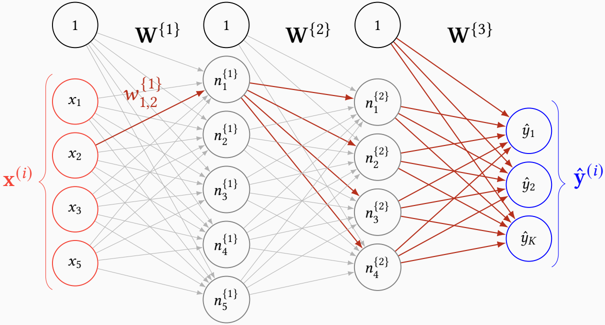
$$\hat{\mathbf{y}}^{(i)} = \mathbf{W}^{\{3\}\top} \cdot \overbrace{\sigma\left(\mathbf{W}^{\{2\}\top} \cdot \overbrace{\sigma\left(\mathbf{W}^{\{1\}\top} \cdot \mathbf{x}^{(i)}\right)}^{\mathbf{a}^{\{2\}(i)}}\right)}^{\mathbf{a}^{\{1\}(i)}}$$

- ¿Cómo calculo los gradientes de la función de pérdida respecto a los pesos de la primera capa?



Propagación hacia adelante de red densa (4)

- ¿Cómo calculo los gradientes de la función de pérdida respecto a los pesos de la primera capa?

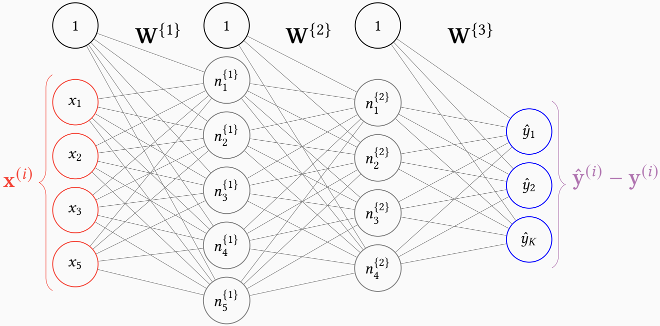


Algoritmo de retro-propagación

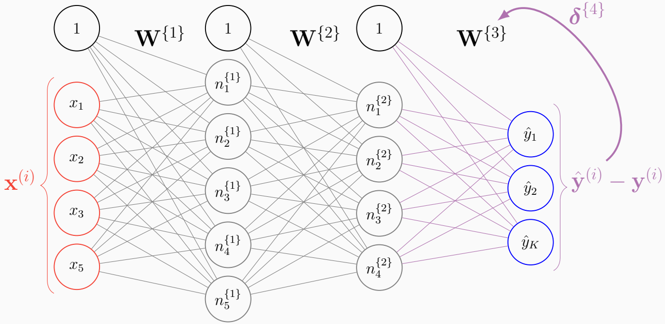
1. Propagamos cada entrada $\mathrm{x}^{(i)}$ hacia adelante para generar la correspondiente salida $\hat{\mathrm{y}}^{(i)}$

2. Calculamos derivadas parciales de la pérdida respecto a cada peso y sesgo capa por capa, empezando con la de salida y propagándolas hacia atrás para calcular las de la capa anterior

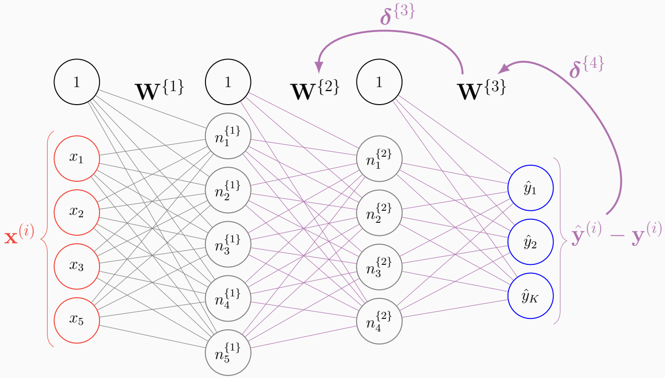
- Presuponiendo que se busca minimizar la función de pérdida de error cuadrático medio (ECM)



- Se calcula el error de la predicción y se propaga hacia la última capa

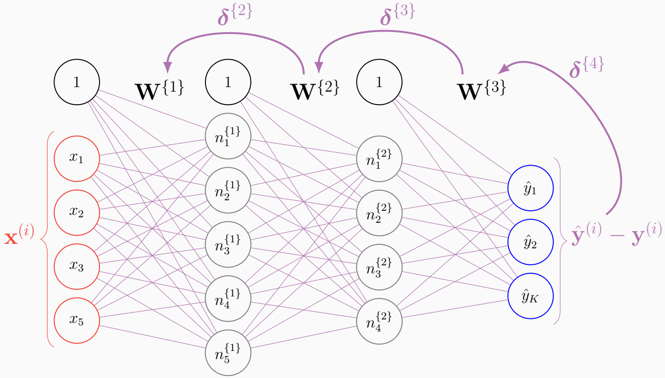


- Se calcula el error de la última capa usando el error de la predicción y se propaga hacia la penúltima capa



Propagación hacia atrás (4)

- Se calcula el error de la penúltima capa usando el error de la última capa y se propaga hacia la primera capa



- Considera una red densa con 1 capa de entrada, 1 capa oculta con M neuronas sigmoide y 1 neurona de salida con activación lineal.
- La propagación hacia adelante estaría dada por

$$
\begin{aligned}
\mathbf{z}^{\{2\}} &= \mathbf{W}^{\{1\}} \cdot \mathbf{x}^{(i)} \\
\mathbf{a}^{\{2\}} &= \phi(\mathbf{z}^{\{2\}}) \\
z^{\{3\}} &= \mathbf{W}^{\{2\}} \cdot \mathbf{a}^{\{2\}} \\
\hat{y}^{(i)} &= \phi(z^{\{3\}})
\end{aligned}
$$

- Presuponemos una tarea de regresión y la función de pérdida

$$ECM(\mathbf{y}, \hat{\mathbf{y}}) = \frac{1}{2}\sum_{i=1}^{N}(y^{(i)} - \hat{y}^{(i)})^2$$

- Calculamos el gradiente de la función de pérdida con respecto a $\mathrm{W}^{\{2\}}$ de la siguiente forma

$$
\begin{aligned}
\frac{\partial ECM}{\partial \mathrm{W}^{\{2\}}} &= \frac{\partial \sum_i \frac{1}{2}(y^{(i)} - \hat{y}^{(i)})^2}{\partial \mathrm{W}^{\{2\}}} \\
&= \frac{\sum_i \partial \frac{1}{2}(y^{(i)} - \hat{y}^{(i)})^2}{\partial \mathrm{W}^{\{2\}}} \\
\frac{\partial \frac{1}{2}(y^{(i)} - \hat{y}^{(i)})^2}{\partial \mathrm{W}^{\{2\}}} &= (y^{(i)} - \hat{y}^{(i)}) \cdot \left(-\frac{\partial \hat{y}^{(i)}}{\partial \mathrm{W}^{\{2\}}}\right) \\
&= (y^{(i)} - \hat{y}^{(i)}) \cdot \left(-\frac{\partial \hat{y}^{(i)}}{\partial z^{\{3\}}} \cdot \frac{\partial z^{\{3\}}}{\partial \mathrm{W}^{\{2\}}}\right) \\
&= \underbrace{-(y^{(i)} - \hat{y}^{(i)}) \cdot \frac{\partial \hat{y}^{(i)}}{\partial z^{\{3\}}}}_{\delta^{\{3\}}} \cdot \mathrm{a}^{\{2\}}
\end{aligned}
$$

Ejemplo: propagación hacia atrás (1)

- Calculamos el gradiente de la función de pérdida respecto a $\mathsf{W}^{\{1\}}$ de la siguiente forma

$$\begin{aligned}
\frac{\partial ECM}{\partial \mathsf{W}^{\{1\}}} &= \frac{\partial \sum_i \frac{1}{2}(y^{(i)} - \hat{y}^{(i)})^2}{\partial \mathsf{W}^{\{1\}}} \\
&= \frac{\sum_i \partial \frac{1}{2}(y^{(i)} - \hat{y}^{(i)})^2}{\partial \mathsf{W}^{\{1\}}} \\
\frac{\partial \frac{1}{2}(y^{(i)} - \hat{y}^{(i)})^2}{\partial \mathsf{W}^{\{1\}}} &= (y^{(i)} - \hat{y}^{(i)}) \left(-\frac{\partial \hat{y}^{(i)}}{\partial \mathsf{W}^{\{1\}}} \right) \\
&= (y^{(i)} - \hat{y}^{(i)}) \left(-\frac{\partial \hat{y}^{(i)}}{\partial \mathbf{z}^{\{3\}}} \cdot \frac{\partial z^{\{3\}}}{\partial \mathsf{W}^{\{1\}}} \right) \\
&= \underbrace{-(y^{(i)} - \hat{y}^{(i)}) \cdot \frac{\partial \hat{y}^{(i)}}{\partial z^{\{3\}}}}_{\delta^{\{3\}}} \cdot \frac{\partial z^{\{3\}}}{\partial \mathsf{W}^{\{1\}}} = \delta^{\{3\}} \cdot \frac{\partial z^{\{3\}}}{\partial \mathsf{W}^{\{1\}}}
\end{aligned}$$

Ejemplo: propagación hacia atrás (2)

$$
\begin{aligned}
&= \delta^{\{3\}} \cdot \left(\overbrace{\frac{\partial z^{\{3\}}}{\partial \mathrm{a}^{\{2\}}}}^{\mathrm{W}^{\{2\}}} \cdot \frac{\partial \mathrm{a}^{\{2\}}}{\partial \mathrm{W}^{\{1\}}} \right) \\
&= \delta^{\{3\}} \cdot \mathrm{W}^{\{2\}} \cdot \left(\frac{\partial \mathrm{a}^{\{2\}}}{\partial \mathrm{W}^{\{1\}}} \right) \\
&= \delta^{\{3\}} \cdot \mathrm{W}^{\{2\}} \cdot \left(\frac{\partial \mathrm{a}^{\{2\}}}{\partial \mathrm{z}^{\{2\}}} \cdot \underbrace{\frac{\partial \mathrm{z}^{\{2\}}}{\partial \mathrm{W}^{\{1\}}}}_{\mathrm{x}^{(i)}} \right) \\
&= \delta^{\{3\}} \cdot \mathrm{W}^{\{2\}} \cdot \frac{\partial \mathrm{a}^{\{2\}}}{\partial \mathrm{z}^{\{2\}}} \cdot \mathrm{x}^{(i)}
\end{aligned}
$$

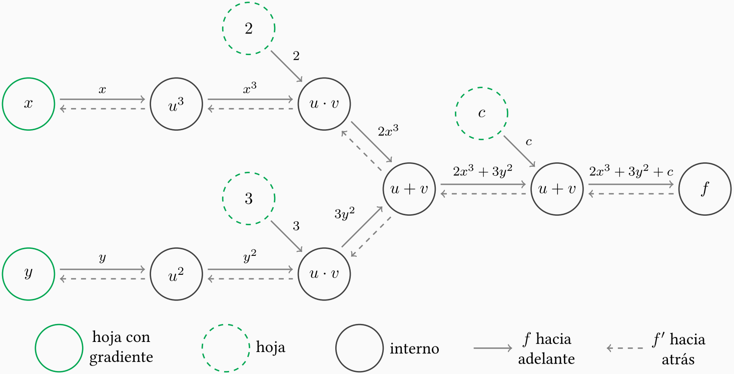


Imagen tomada de Baydin et. al. Automatic Differentiation in Machine Learning: a Survey, 2018.

Características generales de las redes neuronales densas

- Aproximadores universales (con 1 sola capa oculta con un número finito de neuronas[4,5])
- Frecuentemente sobreparametrizados[6]
- Usualmente empleados como bloques de clasificación o de proyección (no tan profundos) en conjunto con otros tipos de capas

---

[4] Cybenko. Approximation by Superpositions of a Sigmoidal Function, 1989

[5] Hornik et al. Multilayer Feedforward Networks are Universal Approximators, 1989.

[6] Allen-Zhu et al. Learning and Generalization in Overparameterized Neural Networks, Going Beyond Two Layers, 2020.