Aprendizaje y Evolución Orientado al Control

Introducción a redes neuronales y al aprendizaje supervisado

Nicanor Quijano
Juan Martinez-Piazuelo

Departamento de Ingeniería Eléctrica y Electrónica
Universidad de los Andes

Semestre 2020-10

Organización de la clase

1. Introducción al aprendizaje supervisado.
2. Redes neuronales: motivación, componentes y arquitectura básica.
3. Entrenamiento de una neurona.
4. Entrenamiento de una red neuronal.
5. Arquitecturas avanzadas y aprendizaje profundo (deep learning).
6. Taller de programación en Python + Numpy.

Introducción al aprendizaje supervisado

En aprendizaje supervisado se busca aprender una función desconocida $\mathbf{y}^* = J^*(\mathbf{x})$, a partir de un set de datos $\mathcal{D} = \{\mathbf{x}_i, \mathbf{y}_i^*\}$.

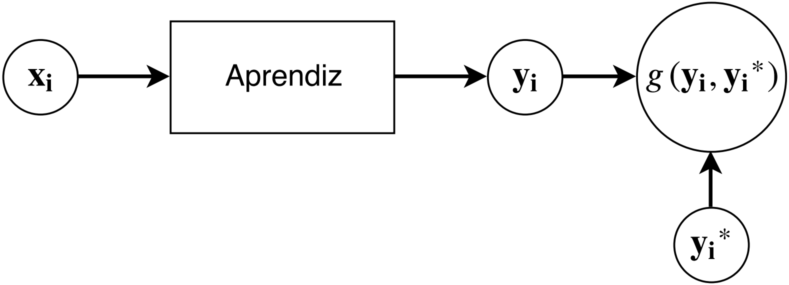
Se busca aprender a predecir las respuestas correctas $\mathbf{y}_i^*$ a partir de los datos de entrada $\mathbf{x}_i \in \mathcal{D}$.

El objetivo es predecir correctamente $\mathbf{y}^*$ para todo posible $\mathbf{x}$ (incluyendo posibles $\mathbf{x}$ no disponibles en $\mathcal{D}$).

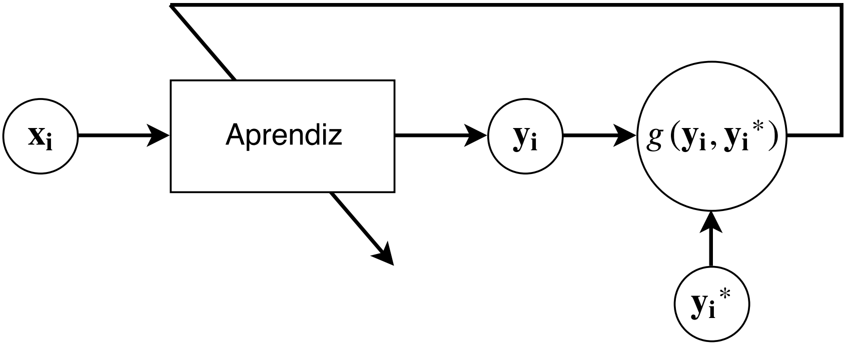
Introducción al aprendizaje supervisado

El **aprendiz** es una función parámetrizada que busca aproximar a $J^*(\mathbf{x})$ para todo posible $\mathbf{x} \in \mathbb{R}^d$.

Introducción al aprendizaje supervisado

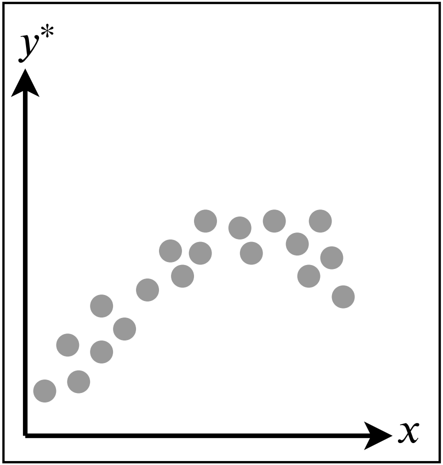


El **aprendiz** es una función parámetrizada que busca aproximar a $J^*(\mathbf{x})$ para todo posible $\mathbf{x} \in \mathbb{R}^d$.

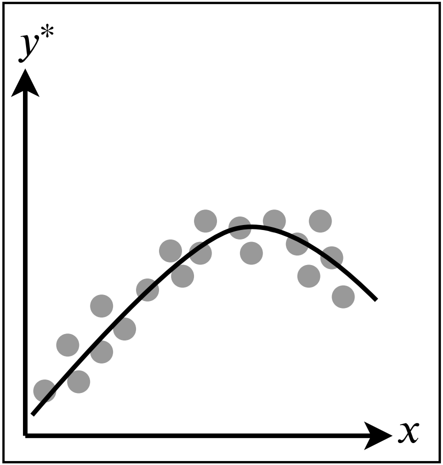


El **aprendiz** es una función parámetrizada que busca aproximar a $J^*(\mathbf{x})$ para todo posible $\mathbf{x} \in \mathbb{R}^d$.

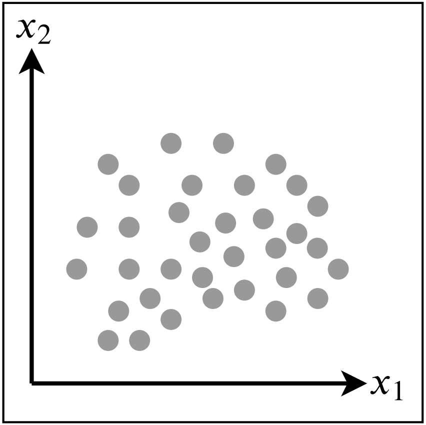
Regresión no lineal:



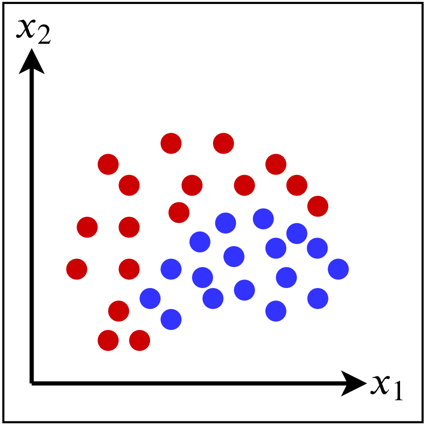
Regresión no lineal:



Clasificación no lineal:

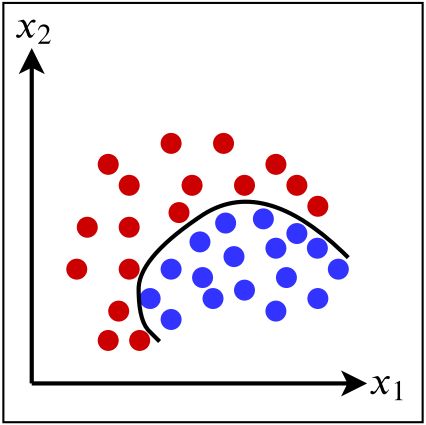


Clasificación no lineal:



Introducción al aprendizaje supervisado

Clasificación no lineal:



La neurona:

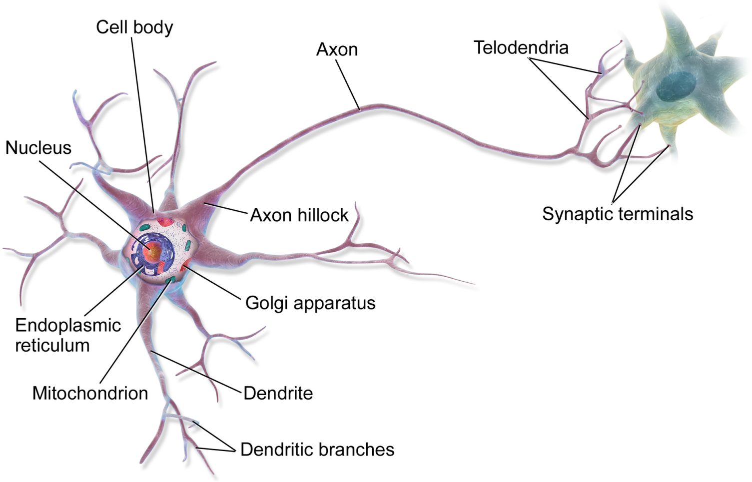
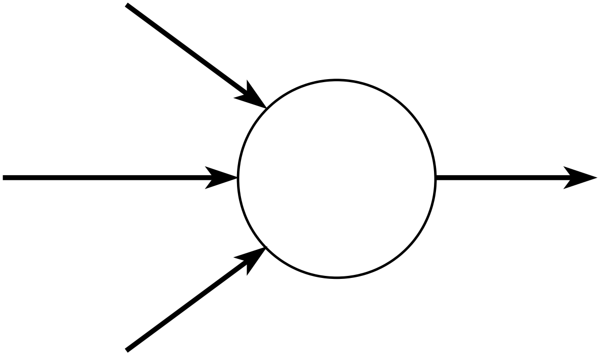
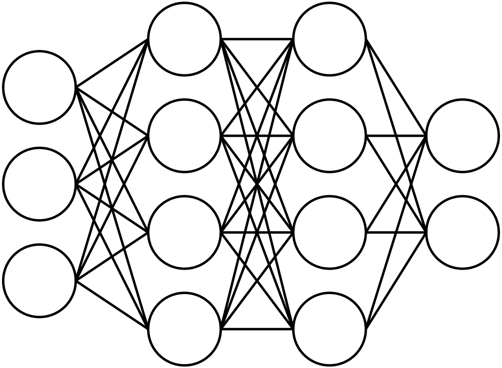


Imagen tomada de: en.wikipedia.org/wiki/Neuron

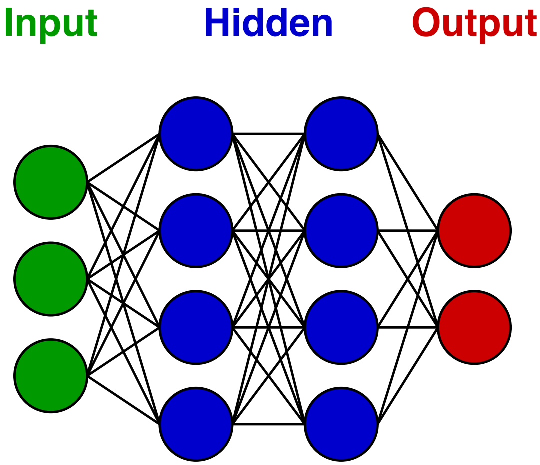
La neurona artificial:



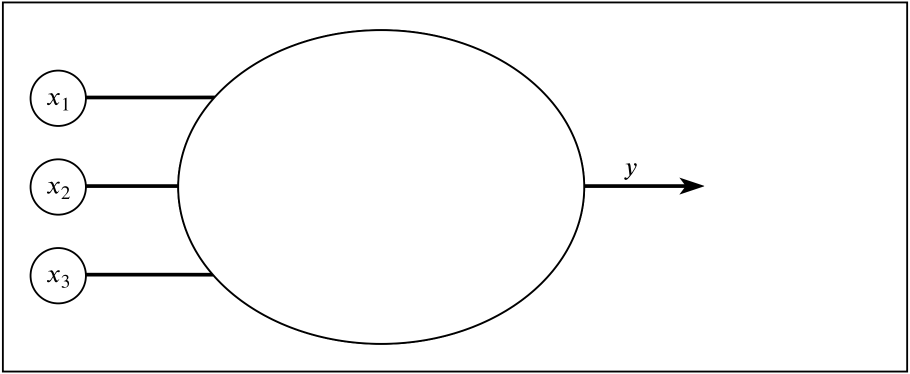
Red neuronal artificial:



Capas de una red neuronal artificial:

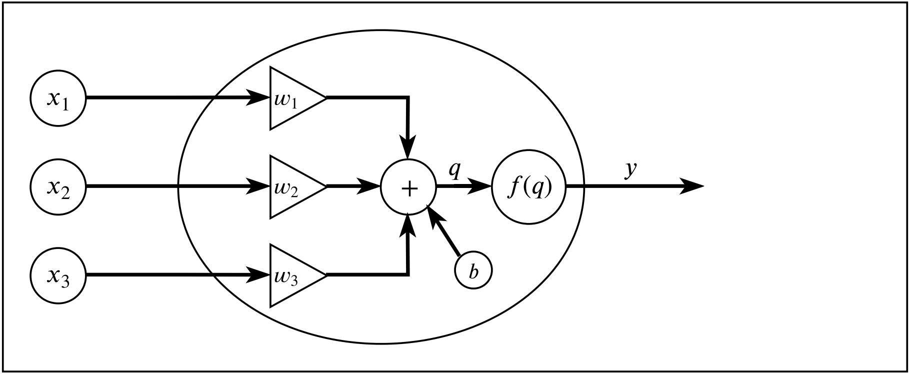


Entrenamiento de una neurona



$x_i \in \mathbb{R}\, \forall i, \quad y \in \mathbb{R},$

Entrenamiento de una neurona



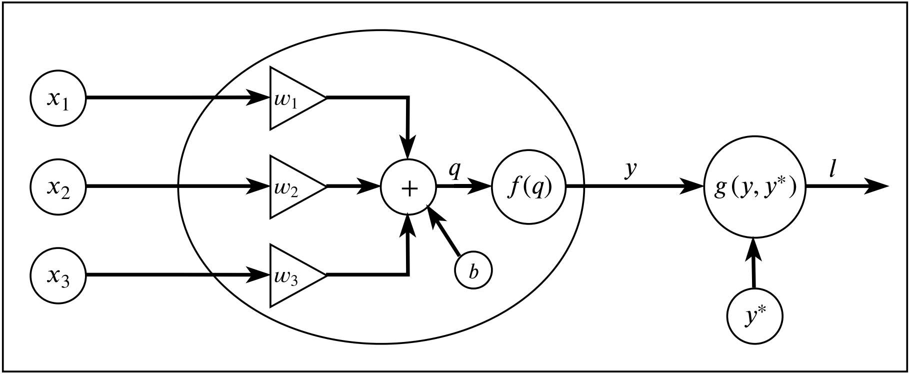
$$x_i \in \mathbb{R}\,\forall i, \quad y \in \mathbb{R},$$

$$w_i \in \mathbb{R}\,\forall i, \quad b \in \mathbb{R},$$

$$q = w_1x_1 + w_2x_2 + w_3x_3 + b,$$

$$y = f(q),$$

Entrenamiento de una neurona



$x_i \in \mathbb{R}\,\forall i, \quad y \in \mathbb{R},$

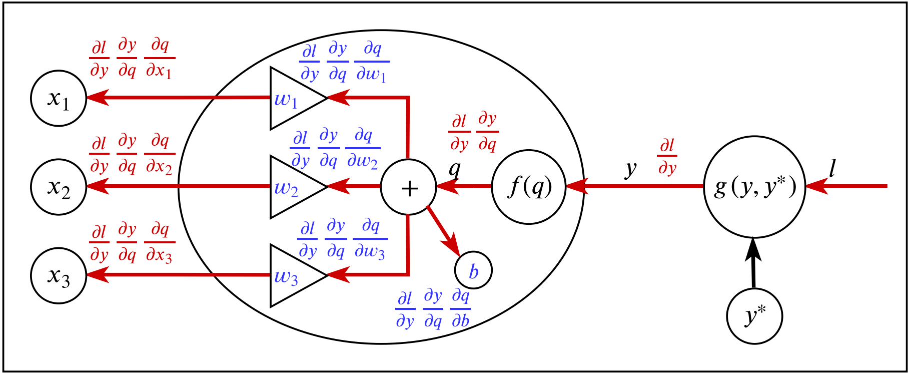
$w_i \in \mathbb{R}\,\forall i, \quad b \in \mathbb{R},$

$q = w_1 x_1 + w_2 x_2 + w_3 x_3 + b,$

$y = f(q), \quad l = g\left(y, y^*\right), \quad y^* \in \mathbb{R},$

$f(\cdot)$ y $g(\cdot,\cdot)$ son **diferenciables,**

Entrenamiento de una neurona



$x_i \in \mathbb{R}\,\forall i, \quad y \in \mathbb{R},$

$w_i \in \mathbb{R}\,\forall i, \quad b \in \mathbb{R},$

$q = w_1x_1 + w_2x_2 + w_3x_3 + b,$

$y = f(q), \quad l = g\left(y, y^*\right), \quad y^* \in \mathbb{R},$

$f(\cdot)$ y $g(\cdot,\cdot)$ son **diferenciables,**

$$\frac{\partial q}{\partial w_i} = x_i, \quad \frac{\partial q}{\partial b} = 1, \quad \frac{\partial q}{\partial x_i} = w_i,$$

$$\frac{\partial y}{\partial q} = \frac{\partial f(q)}{\partial q}, \quad \frac{\partial l}{\partial y} = \frac{\partial g\left(y, y^*\right)}{\partial y}.$$

Entrenamiento de una neurona

1. Inicializar $\mathbf{w}$ y b
2. Computar: $q = \mathbf{w}^\top \mathbf{x} + b$
3. Computar: $y = f(q)$, $\frac{\partial y}{\partial q} = \frac{\partial f(q)}{\partial q}$
4. Computar: $l = g\left(y, y^*\right)$, $\frac{\partial l}{\partial y} = \frac{\partial g(y,y^*)}{\partial y}$
5. Computar: $\frac{\partial l}{\partial w_i} = \frac{\partial g(y,y^*)}{\partial y} \frac{\partial f(q)}{\partial q} x_i \; \forall i$
6. Computar: $\frac{\partial l}{\partial b} = \frac{\partial g(y,y^*)}{\partial y} \frac{\partial f(q)}{\partial q}$
7. Actualizar $\mathbf{w}$ y b usando algún método de descenso de gradiente
8. Iterar

Ejemplo: entrenamiento de una neurona

Sea $y^* = 1$, $\mathbf{x} = \begin{bmatrix} -1 & 2 & 1 \end{bmatrix}^\top$, $\mathbf{w} = \begin{bmatrix} 0.1 & -0.1 & 0.1 \end{bmatrix}^\top$, $b = 0$, $f(q) = 1/\left(1+e^{-q}\right)$, y $g\left(y, y^*\right) = \frac{1}{2}\left(y^* - y\right)^2$, realice una actualización de los parámetros $\mathbf{w}$ y b usando:

$$w_i \leftarrow w_i - \alpha \frac{\partial l}{\partial w_i}, \quad b \leftarrow b - \alpha \frac{\partial l}{\partial b},$$

con $\alpha = 1$. **Ayuda:** $\partial f(q)/\partial q = f(q)\left[1 - f(q)\right]$.

Solución:

1. $q = \mathbf{w}^\top \mathbf{x} + b = -0.2000$
2. $y = f(q) = 0.4502$
3. $\partial f(q)/\partial q = 0.4502\left[1 - 0.4502\right] = 0.2475$
4. $\partial g\left(y, y^*\right)/\partial y = -\left(y^* - y\right) = -0.5498$
5. $\mathbf{w} = \begin{bmatrix} -0.0361 & 0.1722 & 0.2361 \end{bmatrix}^\top$, $b = 0.1361$

Ejemplo: entrenamiento de una neurona

Para los parámetros iniciales se tiene que:

$$l = \frac{1}{2}\left(1 - 0.4502\right)^2 = 0.1512.$$

Para los nuevos parámetros se tiene que:

$$\begin{aligned} q &= \begin{bmatrix} -0.0361 & 0.1722 & 0.2361 \end{bmatrix}^\top \mathbf{x} + 0.1361 = 0.7527 \\ f(q) &= 0.6798 \\ l &= \frac{1}{2}\left(1 - 0.6798\right)^2 = 0.0512. \end{aligned}$$

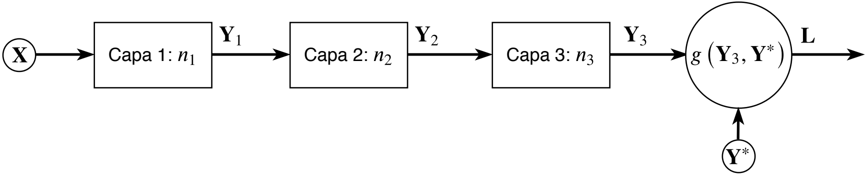
Observe que: $0.0512 < 0.1512$, i.e., la predicción sobre $\mathbf{x}$ mejoró.

Entrenamiento de una red neuronal

Para extender el procedimiento descrito al entrenamiento de una red neuronal hay que reescribir las operaciones anteriores en forma matricial.

Las operaciones vectorizadas se ejecutan más eficientemente!

Considere la siguiente red con 3 capas:



$$\mathbf{X} \in \mathbb{R}^{d\times m}, \quad \mathbf{Y}_1 \in \mathbb{R}^{n_1\times m}, \quad \mathbf{Y}_2 \in \mathbb{R}^{n_2\times m}, \quad \mathbf{Y}_3 \in \mathbb{R}^{n_3\times m}, \quad \mathbf{Y}^* \in \mathbb{R}^{n_3\times m},$$

$$\mathbf{L} \in \mathbb{R}^{n_3\times m}, \quad \mathbf{W}_1 \in \mathbb{R}^{n_1\times d}, \quad \mathbf{W}_2 \in \mathbb{R}^{n_2\times n_1}, \quad \mathbf{W}_3 \in \mathbb{R}^{n_3\times n_2},$$

$$\mathbf{b}_1 \in \mathbb{R}^{n_1}, \quad \mathbf{b}_2 \in \mathbb{R}^{n_2}, \quad \mathbf{b}_3 \in \mathbb{R}^{n_3}.$$

Nota: d es la dimensión de los datos (3 en el ejemplo de una neurona) y m es el tamaño del batch de datos.

Entrenamiento de una red neuronal

Entonces:

$$
\begin{aligned}
\mathbf{Q}_1 &= \mathbf{W}_1\mathbf{X} + \mathbf{b}_1\mathbb{1}_m^\top & \mathbf{Q}_1 &\in \mathbb{R}^{n_1\times m}\\
\mathbf{Y}_1 &= f_1\left(\mathbf{Q}_1\right) & \mathbf{Y}_1 &\in \mathbb{R}^{n_1\times m}\\
\mathbf{Q}_2 &= \mathbf{W}_2\mathbf{Y}_1 + \mathbf{b}_2\mathbb{1}_m^\top & \mathbf{Q}_2 &\in \mathbb{R}^{n_2\times m}\\
\mathbf{Y}_2 &= f_2\left(\mathbf{Q}_2\right) & \mathbf{Y}_2 &\in \mathbb{R}^{n_2\times m}\\
\mathbf{Q}_3 &= \mathbf{W}_3\mathbf{Y}_2 + \mathbf{b}_3\mathbb{1}_m^\top & \mathbf{Q}_3 &\in \mathbb{R}^{n_3\times m}\\
\mathbf{Y}_3 &= f_3\left(\mathbf{Q}_3\right) & \mathbf{Y}_3 &\in \mathbb{R}^{n_3\times m}
\end{aligned}
$$

Las funciones $f_i(\cdot)$ y $g(\cdot,\cdot)$ se operan por elementos (*element-wise*).

Nota: $\mathbb{1}_m^\top = \begin{bmatrix} 1 & 1 & \cdots & 1 \end{bmatrix} \in \mathbb{R}^m$

Entrenamiento de una red neuronal

Teniendo en cuenta que $g(\cdot,\cdot)$ y $f_i(\cdot)$ se operan por elementos, denotamos:

$$\frac{\partial \mathbf{L}}{\partial \mathbf{Y}_3} := \frac{\partial g}{\partial y}\left(\mathbf{Y}_3, \mathbf{Y}_3^*\right) \in \mathbb{R}^{n_3 \times m}, \quad \frac{\partial \mathbf{Y}_3}{\partial \mathbf{Q}_3} := \frac{\partial f_3}{\partial q}\left(\mathbf{Q}_3\right) \in \mathbb{R}^{n_3 \times m}.$$

Es decir:

$$\frac{\partial \mathbf{L}}{\partial \mathbf{Y}_3} := \begin{bmatrix} \frac{\partial g}{\partial y}\left(\mathbf{Y}_3(1,1), \mathbf{Y}_3^*(1,1)\right) & \cdots & \frac{\partial g}{\partial y}\left(\mathbf{Y}_3(1,m), \mathbf{Y}_3^*(1,m)\right) \\ \vdots & \ddots & \vdots \\ \frac{\partial g}{\partial y}\left(\mathbf{Y}_3(n_3,1), \mathbf{Y}_3^*(n_3,1)\right) & \cdots & \frac{\partial g}{\partial y}\left(\mathbf{Y}_3(n_3,m), \mathbf{Y}_3^*(n_3,m)\right) \end{bmatrix}$$

$$\frac{\partial \mathbf{Y}_3}{\partial \mathbf{Q}_3} := \begin{bmatrix} \frac{\partial f_3}{\partial q}\left(\mathbf{Q}_3(1,1)\right) & \cdots & \frac{\partial f_3}{\partial q}\left(\mathbf{Q}_3(1,m)\right) \\ \vdots & \ddots & \vdots \\ \frac{\partial f_3}{\partial q}\left(\mathbf{Q}_3(n_3,1)\right) & \cdots & \frac{\partial f_3}{\partial q}\left(\mathbf{Q}_3(n_3,m)\right) \end{bmatrix}$$

Nota: la notación $\mathbf{A}(j,k)$ denota el elemento (j,k) de la matriz $\mathbf{A}$.

Entrenamiento de una red neuronal

Para el caso de una neurona teníamos que:

$$\frac{\partial l}{\partial w_i} = \frac{\partial g\left(y, y^*\right)}{\partial y}\frac{\partial f(q)}{\partial q}x_i \quad \forall i, \qquad \frac{\partial l}{\partial b} = \frac{\partial g\left(y, y^*\right)}{\partial y}\frac{\partial f(q)}{\partial q}.$$

Extrapolando para el caso con múltiples neuronas encontramos que para la tercera capa se tiene:

$$\frac{\partial \mathbf{L}}{\partial \mathbf{W}_3} := \left[\frac{\partial \mathbf{L}}{\partial \mathbf{Y}_3} \odot \frac{\partial \mathbf{Y}_3}{\partial \mathbf{Q}_3}\right] \mathbf{Y}_2^\top, \quad \frac{\partial \mathbf{L}}{\partial \mathbf{W}_3} \in \mathbb{R}^{n_3 \times n_2}.$$

$$\frac{\partial \mathbf{L}}{\partial \mathbf{b}_3} := \left[\frac{\partial \mathbf{L}}{\partial \mathbf{Y}_3} \odot \frac{\partial \mathbf{Y}_3}{\partial \mathbf{Q}_3}\right] \mathbb{1}_m, \quad \frac{\partial \mathbf{L}}{\partial \mathbf{b}_3} \in \mathbb{R}^{n_3}.$$

Nota: $\odot$ denota el producto de Hadamard (*element-wise*).

Por lo tanto, para la tercera capa tenemos que:

$$\mathbf{W}_3 \leftarrow \mathbf{W}_3 - \frac{\alpha}{m}\frac{\partial \mathbf{L}}{\partial \mathbf{W}_3}$$

$$\mathbf{b}_3 \leftarrow \mathbf{b}_3 - \frac{\alpha}{m}\frac{\partial \mathbf{L}}{\partial \mathbf{b}_3}$$

Esto es asumiendo que empleamos un descenso de gradiente simple como algoritmo de actualización.

Nota: los parámetros se actualizan únicamente al finalizar la propagación de gradientes por TODA la red.

Entrenamiento de una red neuronal

¿Cómo entrenamos los pesos de la segunda capa?

Para el caso de una neurona teníamos que:

$$\frac{\partial l}{\partial x_i} = \frac{\partial g\left(y, y^*\right)}{\partial y} \frac{\partial f(q)}{\partial q} w_i \quad \forall i$$

Por lo tanto:

$$\frac{\partial \mathbf{L}}{\partial \mathbf{Y}_2} := \mathbf{W}_3^\top \left[\frac{\partial \mathbf{L}}{\partial \mathbf{Y}_3} \odot \frac{\partial \mathbf{Y}_3}{\partial \mathbf{Q}_3}\right], \quad \frac{\partial \mathbf{L}}{\partial \mathbf{Y}_2} \in \mathbb{R}^{n_2 \times m}.$$

$$\frac{\partial \mathbf{L}}{\partial \mathbf{W}_2} := \left[\frac{\partial \mathbf{L}}{\partial \mathbf{Y}_2} \odot \frac{\partial \mathbf{Y}_2}{\partial \mathbf{Q}_2}\right] \mathbf{Y}_1^\top, \quad \frac{\partial \mathbf{L}}{\partial \mathbf{W}_2} \in \mathbb{R}^{n_2 \times n_1}.$$

$$\frac{\partial \mathbf{L}}{\partial \mathbf{b}_2} := \left[\frac{\partial \mathbf{L}}{\partial \mathbf{Y}_2} \odot \frac{\partial \mathbf{Y}_2}{\partial \mathbf{Q}_2}\right] \mathbb{1}_m, \quad \frac{\partial \mathbf{L}}{\partial \mathbf{b}_2} \in \mathbb{R}^{n_2}.$$

De forma similar, para la primera capa se tiene:

$$\frac{\partial \mathbf{L}}{\partial \mathbf{Y}_1} := \mathbf{W}_2^\top \left[\frac{\partial \mathbf{L}}{\partial \mathbf{Y}_2} \odot \frac{\partial \mathbf{Y}_2}{\partial \mathbf{Q}_2} \right], \quad \frac{\partial \mathbf{L}}{\partial \mathbf{Y}_1} \in \mathbb{R}^{n_1 \times m}.$$

$$\frac{\partial \mathbf{L}}{\partial \mathbf{W}_1} := \left[\frac{\partial \mathbf{L}}{\partial \mathbf{Y}_1} \odot \frac{\partial \mathbf{Y}_1}{\partial \mathbf{Q}_1} \right] \mathbf{X}^\top, \quad \frac{\partial \mathbf{L}}{\partial \mathbf{W}_1} \in \mathbb{R}^{n_1 \times d}.$$

$$\frac{\partial \mathbf{L}}{\partial \mathbf{b}_1} := \left[\frac{\partial \mathbf{L}}{\partial \mathbf{Y}_1} \odot \frac{\partial \mathbf{Y}_1}{\partial \mathbf{Q}_1} \right] \mathbb{1}_m, \quad \frac{\partial \mathbf{L}}{\partial \mathbf{b}_1} \in \mathbb{R}^{n_1}.$$

De manera generalizada, para una red con N capas se tiene:
Forward pass:

$$\begin{aligned}\mathbf{Q}_i &= \mathbf{W}_i\mathbf{Y}_{i-1} + \mathbf{b}_i\mathbb{1}_m^\top\\ \mathbf{Y}_i &= f_i(\mathbf{Q}_i)\\ \mathbf{Y}_0 &= \mathbf{X}\end{aligned}$$

Backward pass:

$$\boldsymbol{\Phi}(i) = \frac{\partial \mathbf{L}}{\partial \mathbf{Y}_i} \odot \frac{\partial \mathbf{Y}_i}{\partial \mathbf{Q}_i}$$

$$\frac{\partial \mathbf{L}}{\partial \mathbf{Y}_N} := \frac{\partial g}{\partial y}\left(\mathbf{Y}_N, \mathbf{Y}_N^*\right)$$

$$\frac{\partial \mathbf{L}}{\partial \mathbf{Y}_j} := \mathbf{W}_{j+1}^\top \boldsymbol{\Phi}(j+1), \quad j < N$$

$$\frac{\partial \mathbf{Y}_i}{\partial \mathbf{Q}_i} := \frac{\partial f_i}{\partial q}\left(\mathbf{Q}_i\right)$$

$$\frac{\partial \mathbf{L}}{\partial \mathbf{W}_i} := \boldsymbol{\Phi}(i)\mathbf{Y}_{i-1}^\top$$

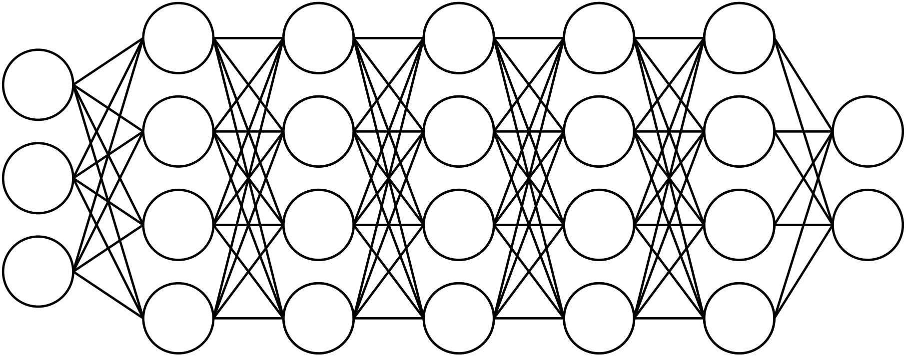
$$\frac{\partial \mathbf{L}}{\partial \mathbf{b}_i} := \boldsymbol{\Phi}(i)\mathbb{1}_m$$

Actualización: métodos de descenso de gradiente.

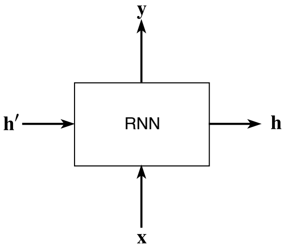
Nota: $i \in \{1, 2, \ldots, N\}$.

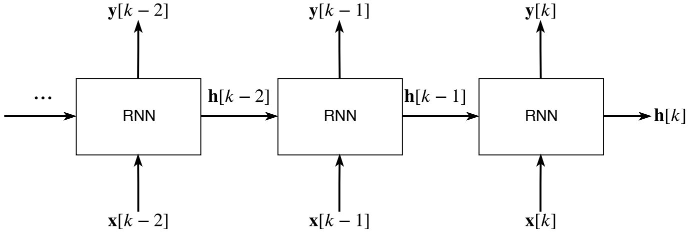
Arquitecturas avanzadas

Deep learning:

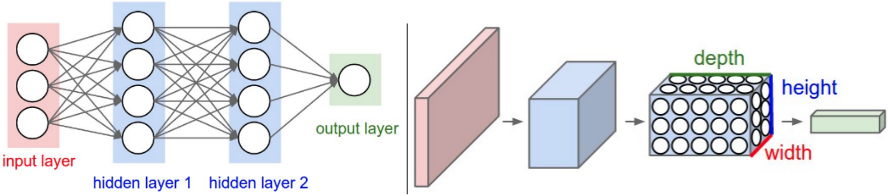


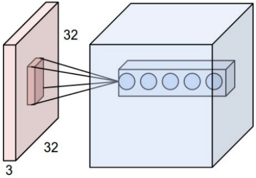
Recurrent neural networks:





Convolutional neural networks:





Ejemplo de aplicación: Convolutional Neural Networks in Action

Imágenes tomadas de: cs231n.github.io/convolutional-networks/

Arquitecturas avanzadas

Autoencoders:

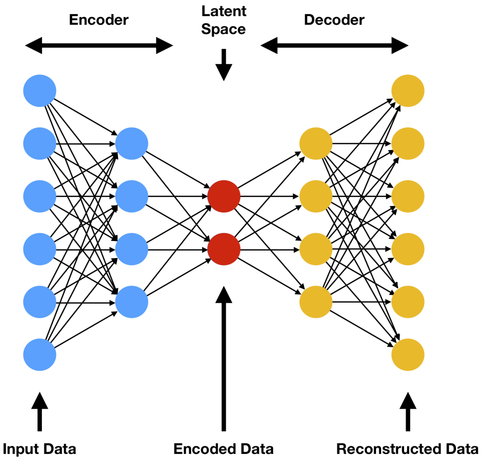
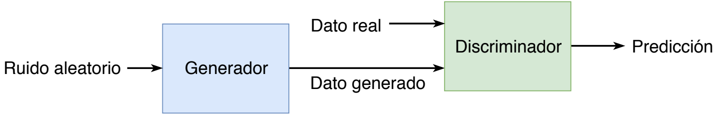
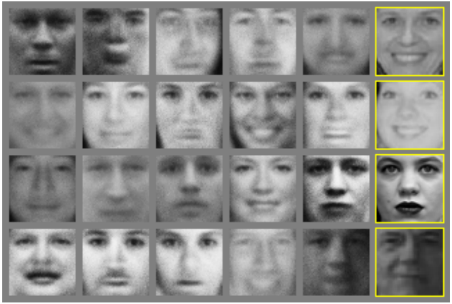


Imagen tomada de: www.compthree.com/blog/autoencoder/

Arquitecturas avanzadas

Generative adversarial networks:





Ver: This cat does not exist, This person does not exist

Ian J. Goodfellow, et. al., Generative Adversarial Nets, NIPS Proceedings, 2014.

Taller de programación

Taller de programación en Python + Numpy

github.com/Martinez-Piazuelo/AEOC-NeuralNetworks-2020