Maestría en Inteligencia Artificial Aplicada

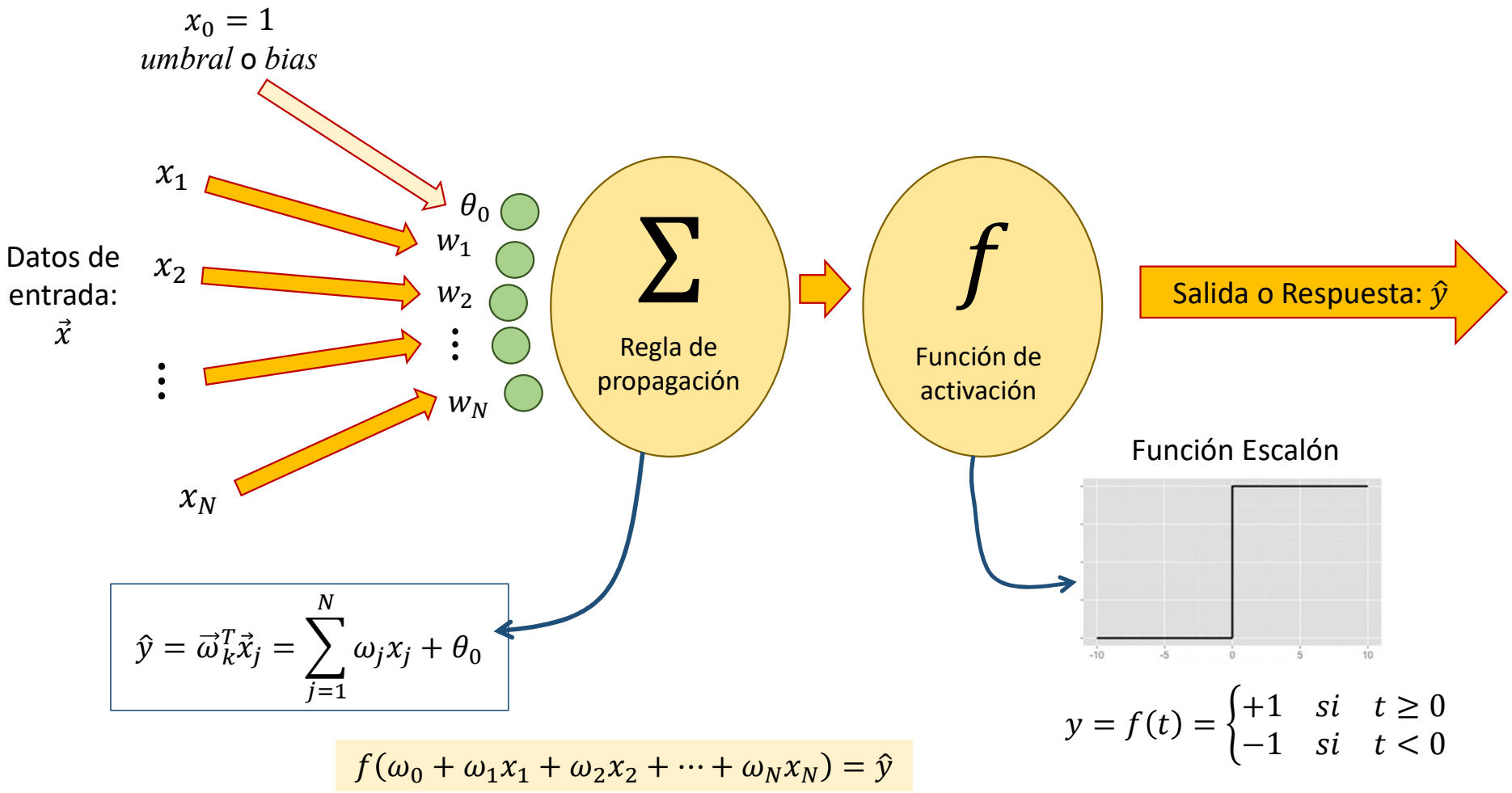
Autoencoders

como técnica de Agrupamiento

(Clustering) no supervisada

Luis Eduardo Falcón Morales

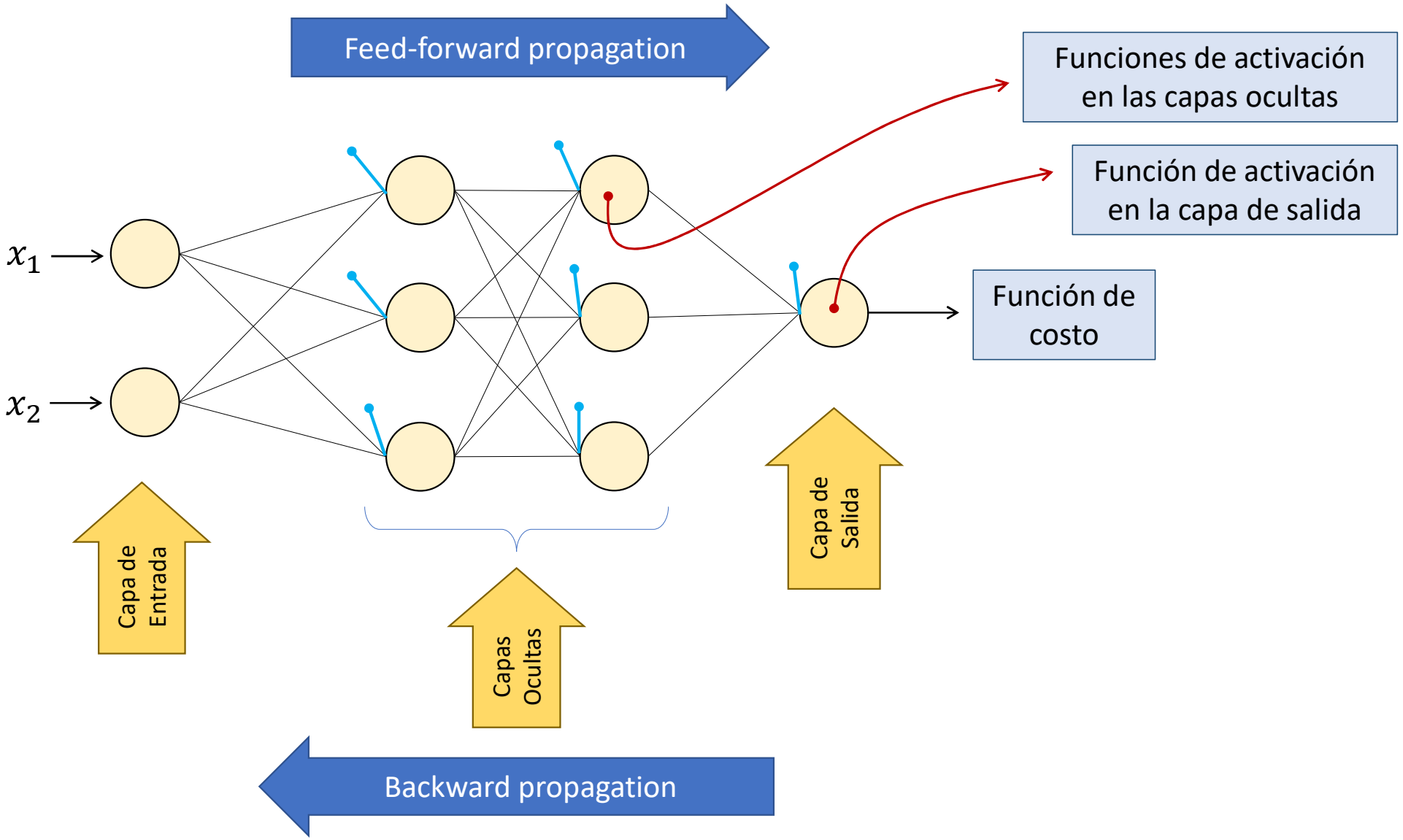
Neurona Artificial



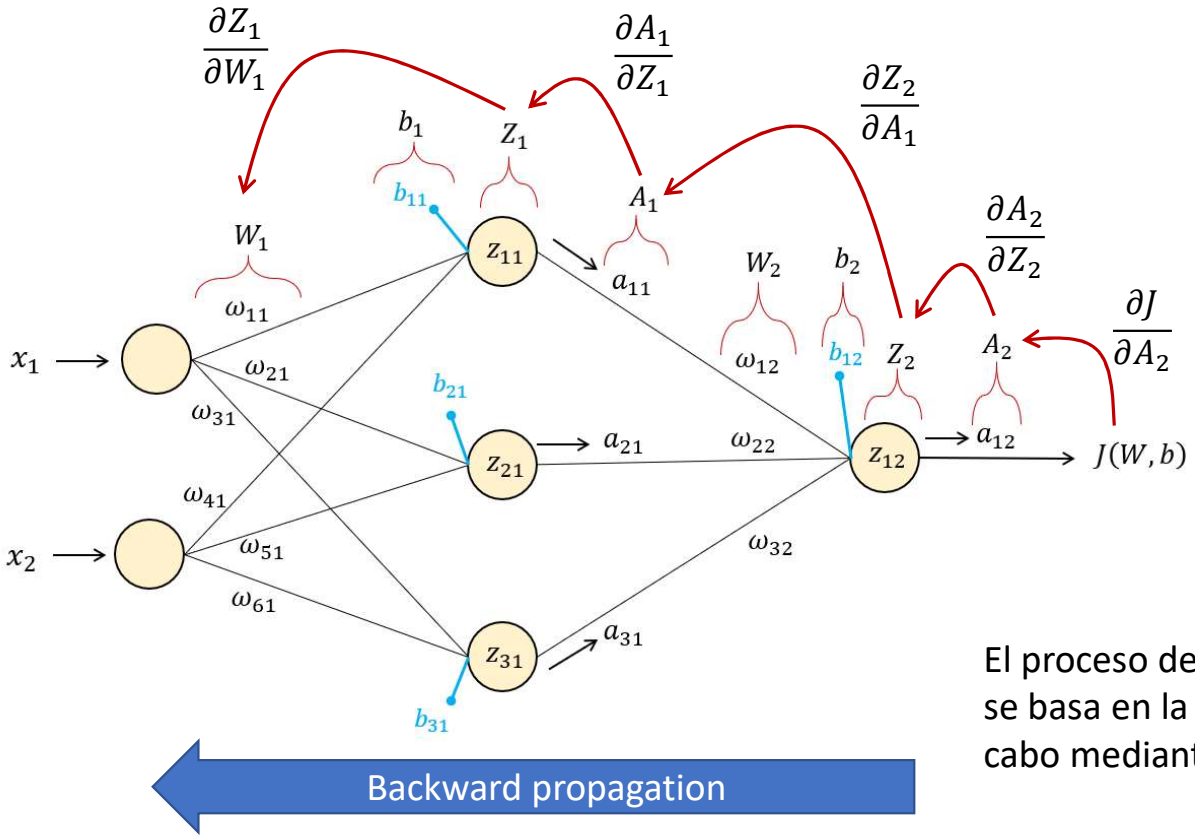
$$\hat{y} = \vec{\omega}_k^T \vec{x}_j = \sum_{j=1}^{N} \omega_j x_j + \theta_0$$

$$y = f(t) = \begin{cases} +1 & si \quad t \geq 0 \\ -1 & si \quad t < 0 \end{cases}$$

$$f(\omega_0 + \omega_1 x_1 + \omega_2 x_2 + \cdots + \omega_N x_N) = \hat{y}$$


Feed-forward propagation
Funciones de activación en las capas ocultas
Función de activación en la capa de salida
Función de costo
x_1
x_2
Capa de Entrada
Capas Ocultas
Capa de Salida
Backward propagation

Técnica de ajuste de los pesos mediante Propagación hacia Atrás (*Back-propagation*)



$$J(W,b) = -\frac{1}{m}\sum_{k=1}^{m}[y_k log(A_2) + (1-y_k) log(1-A_2)]$$

$$A_2 = sigmoid(Z_2)$$

$$Z_2 = W_2^T A_1 + b_2$$

$$A_1 = tanh(Z_1)$$

$$Z_1 = W_1^T X + b_1$$

El proceso de ajuste de los pesos de la red nueronal artificial se basa en la técnica de propagación hacia atrás, que se lleva a cabo mediante una serie de productos de derivadas parciales:

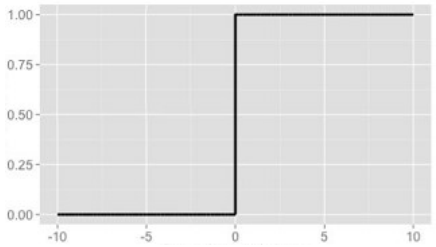
$$\frac{\partial J}{\partial W_1} = \frac{\partial J}{\partial A_2}\frac{\partial A_2}{\partial Z_2}\frac{\partial Z_2}{\partial A_1}\frac{\partial A_1}{\partial Z_1}\frac{\partial Z_1}{\partial W_1}$$

Función de Activación

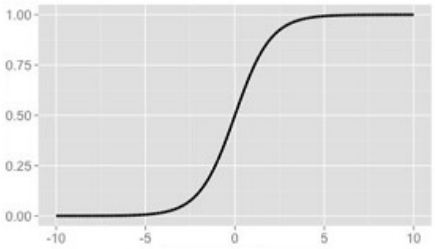
La introducción de nuevas funciones de activación, basadas en ReLU, han permitido atacar mejor el problema de desvanecimiento o explosión de los gradientes.

Escalón

Rango: $\{0, +1\}$



Logística $f(x) = \frac{1}{1 + e^{-x}}$

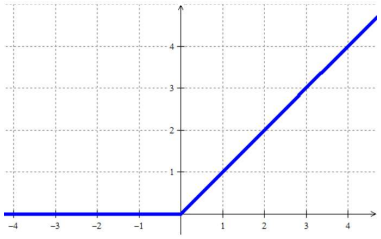


Rango: $(0, +1)$

Unidad Lineal Rectificada
(Rectified Linear Unit)

ReLU

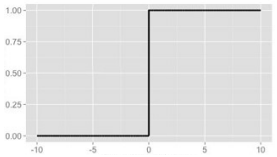
$f(x) = max(0, x)$



Más Funciones de Activación

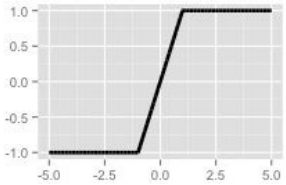
En los últimos años y debido a la popularidad de las redes neuronales profundas, se han estado utilizado ampliamente las **funciones rectificadoras** en los nodos de las capas ocultas.
Estas ayudan al problema del desvanecimiento o explosión de los gradientes.

Escalón



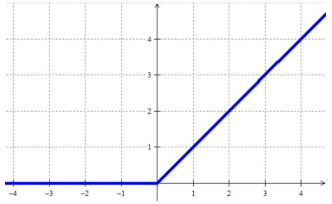
Rango: $\{ 0, +1 \}$

Lineal Seccionada



Rango: $[-1, +1]$

ReLU



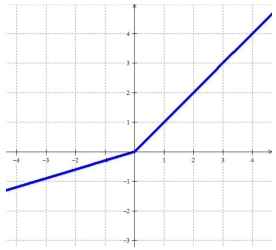
$$f(x) = max(0, x)$$

Leaky ReLU

LReLU

$$f(x) = \begin{cases} x & si \quad x \geq 0 \\ \alpha x & si \quad x < 0 \end{cases}$$

donde $\alpha > 0$



Usualmente $\alpha \in (0, 1]$

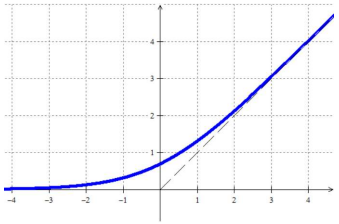
Parametric ReLU

PReLU

Cada neurona k podría tener un valor α_k diferente.

Softplus

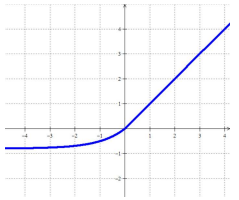
$$f(x) = ln(1 + e^x)$$



Exponential Linear Unit

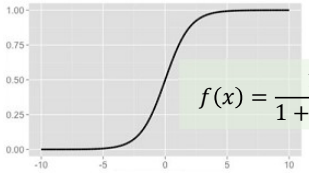
ELU

$$f(x) = \begin{cases} x & si \quad x \geq 0 \\ \alpha(e^x - 1) & si \quad x < 0 \end{cases}$$



donde $\alpha > 0$

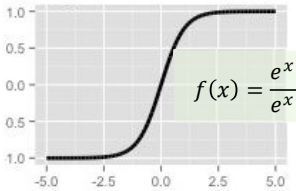
Logística



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

Rango: $(0, +1)$

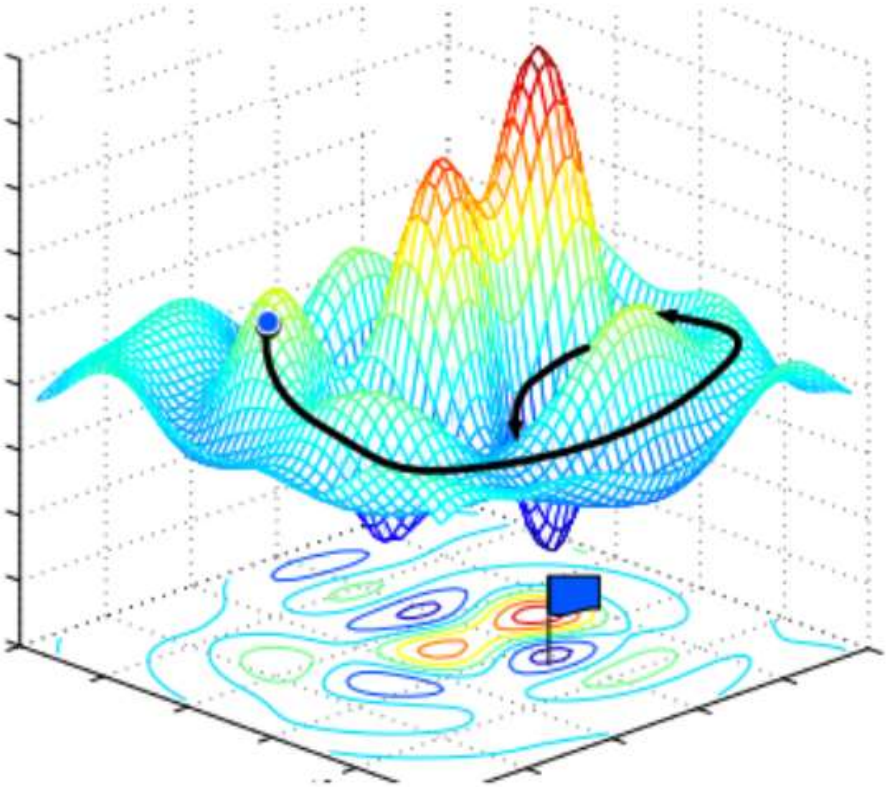
Tangente hiperbólica



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

Rango: $(-1, +1)$

Inicialización de los Pesos de la Red Neuronal



Espacio de búsqueda del mínimo en la función de costo.

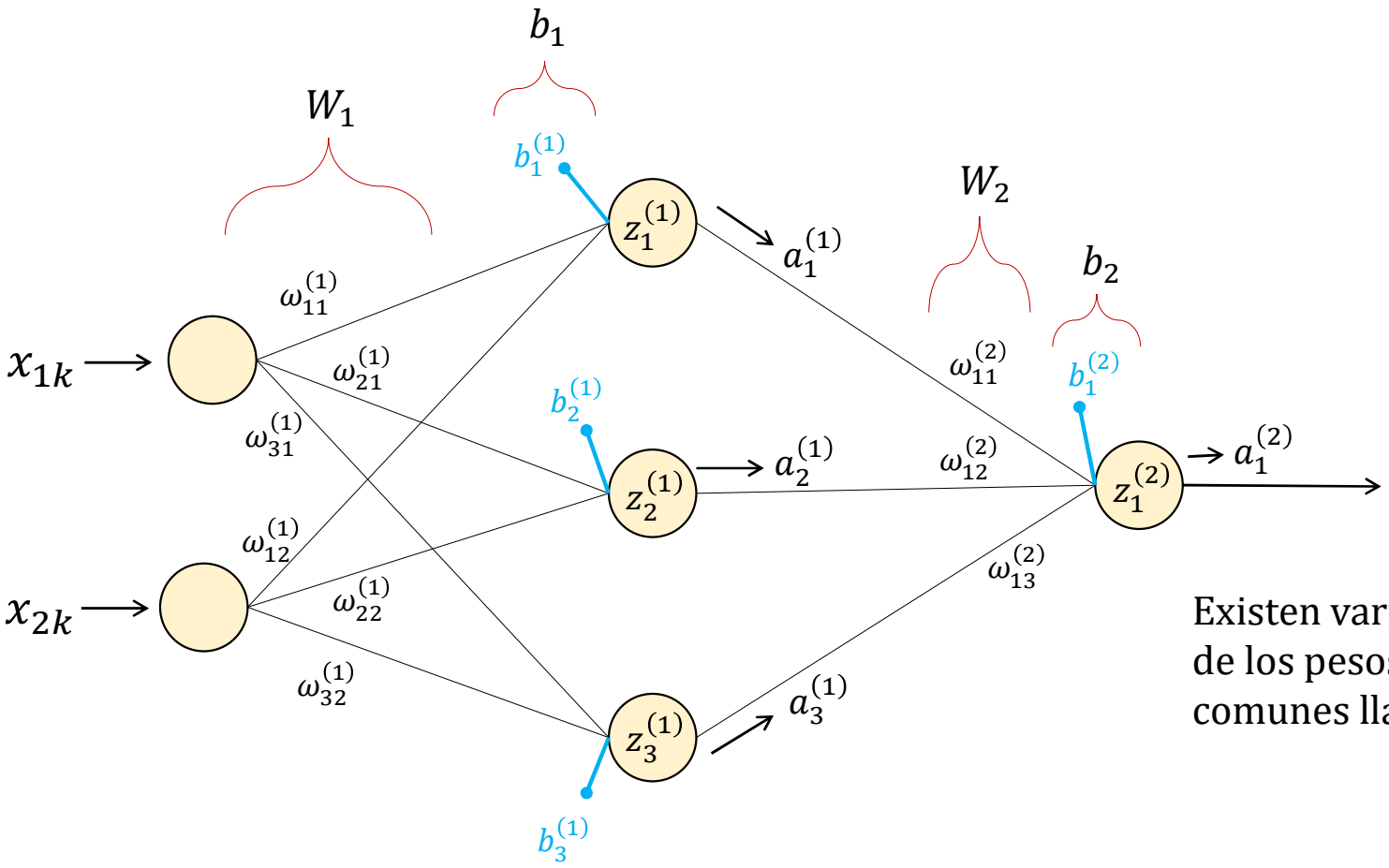
En el proceso de minimizar una función de costo se suelen utilizar métodos no deterministas, es decir aleatorios (estocásticos). En particular el gradiente descendente estocástico (SGD – Stochastic Gradient Descent) es de los más estándar.

Dichos métodos utilizan la aleatoriedad tanto en la inicialización, como durante el proceso de búsqueda del óptimo para evitar quedar atrapados en un mínimo local.

En particular, para la inicialización de los pesos existen varios métodos, veamos uno de los más conocidos llamado de **Glorot o Xavier**.

Estos procesos de inicialización de los pesos son parte de las técnicas modernas que han dado paso al área de **Aprendizaje Profundo** (**Deep Learning**).

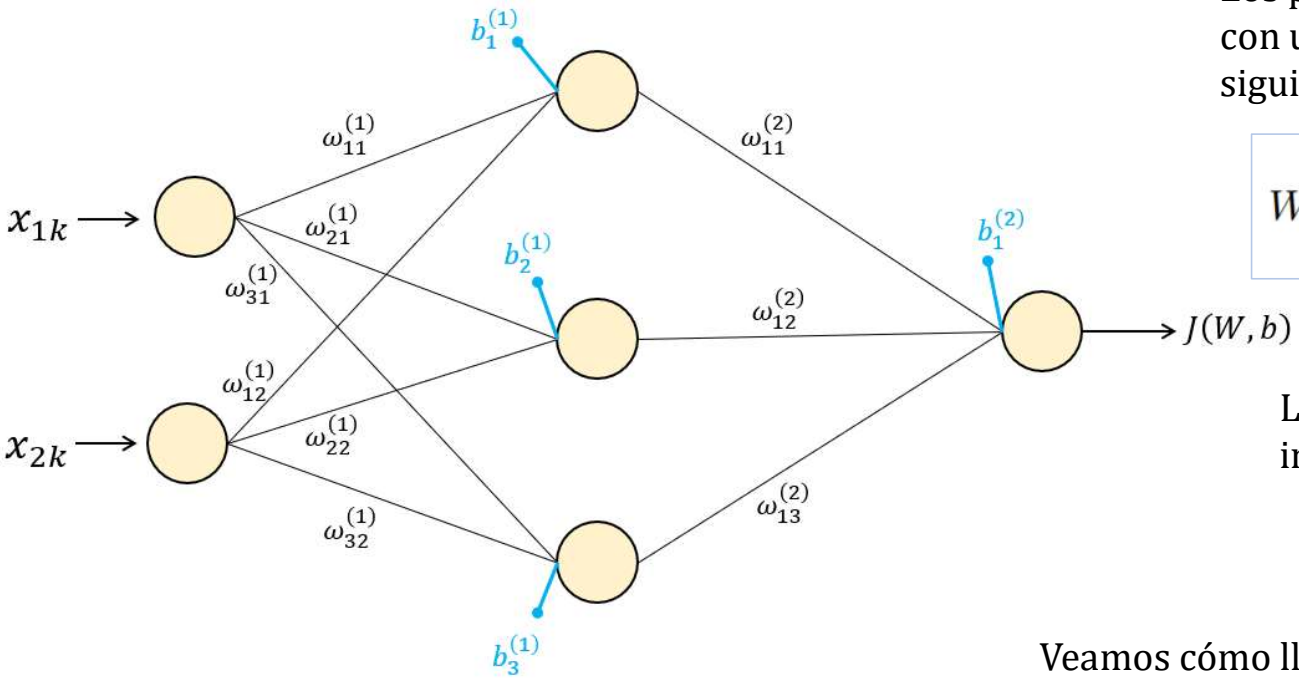
El método del gradiente descendente estocástico (SGD) y el uso de funciones de activación como la sigmoide y la tangente hiperbólica, nos llevan a inicializar aleatoriamente los pesos W_1 y W_2 con valores cercanos a cero y los vectores de sesgos b_1 y b_2 en cero.



Existen varios métodos de inicialización de los pesos, veamos uno de los más comunes llamado de **Glorot o Xavier**.

Inicialización de Xavier o Inicialización de Glorot

¿Cómo encontrar los mejores valores de inicialización de los pesos $W = \omega_{ij}^{(m)}$ y *bias* $b = b_i^{(m)}$ al aplicar el método de optimización?



Los pesos W se inicializan aleatoriamente con una distribución uniforme U entre los siguientes valores:

$$W \sim U\left[-\frac{\sqrt{6}}{\sqrt{n_j + n_{j+1}}}, \frac{\sqrt{6}}{\sqrt{n_j + n_{j+1}}}\right]$$

Los valores de los bias se pueden inicializar siempre en cero:

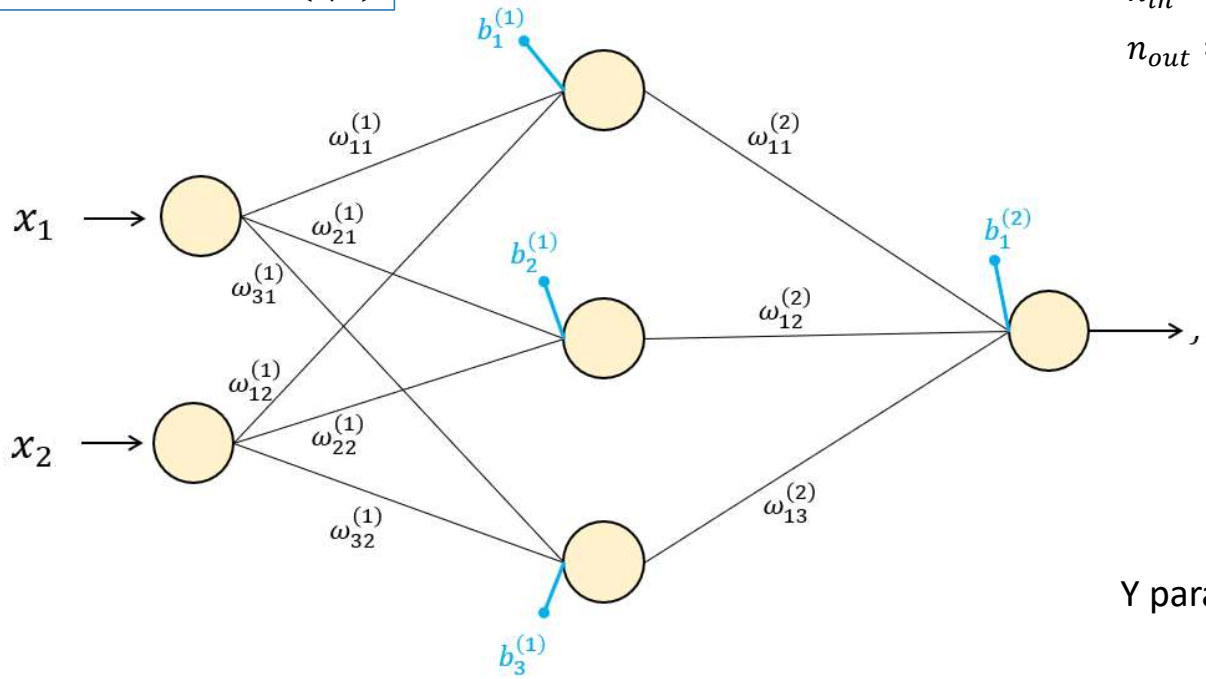
$$b = 0$$

Veamos cómo llegan los autores a estas expresiones…

Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks, 2010.

https://www.semanticscholar.org/paper/Understanding-the-difficulty-of-training-deep-Glorot-Bengio/b71ac1e9fb49420d13e084ac67254a0bbd40f83f

Inicialización de Glorot (1/2)



Por simplicidad veamos el análisis para los pesos de la capa de entrada a la capa oculta del diagrama. Supongamos que se tienen las siguientes distribuciones:

$$X \sim D(\mu_X = 0, \sigma_X^2)$$

$$W \sim D(\mu_W = 0, \sigma_W^2)$$

n_{in} : total neuronas en la capa de entrada

n_{out} : total neuronas en la capa de salida (oculta en este caso)

$$z_i = \sum_{j=1}^{n_{in}} \omega_{ij} x_j$$

Entonces, de Estadística el valor esperado de z_i:

$$E[z_i] = E\left[\sum_{j=1}^{n_{in}} \omega_{ij} x_j\right] = \sum_{j=1}^{n_{in}} E[\omega_{ij}] E[x_j] = 0$$

Y para la varianza:

$$Var[z_i] = E[z_i^2] - (E[z_i])^2 = \sum_{j=1}^{n_{in}} E[\omega_{ij}^2 x_j^2] - 0$$

$$= \sum_{j=1}^{n_{in}} E[\omega_{ij}^2] E[x_j^2] = n_{in} \sigma_W^2 \sigma_X^2$$

La cual finalmente podemos poner la restricción a los pesos W a partir de la capa oculta como:

$$n_{in} \sigma_W^2 = 1$$

Inicialización de Glorot (1/2)

Considerando ahora los pesos de la capa oculta a la capa de salida W, se puede obtener de manera análoga:

$$n_{out}\sigma_W^2 = 1$$

Observa que la varianza de los datos de entrada X no los podemos restringir, por ello aplicamos las restricciones solo a todos los pesos de la red, los cuales sí podemos controlar.

La manera de hacer que estas dos restricciones se cumplan será mediante su promedio, es decir:

$$\frac{1}{2}(n_{in} + n_{out})\sigma_W^2 = 1$$

Es decir, los pesos W los podemos inicializar a partir de una función de distribución W con media 0 y varianza:

$$\sigma_W^2 = \frac{2}{n_{in} + n_{out}}$$

o desviación estándar: $\sigma_W = \sqrt{\frac{2}{n_{in} + n_{out}}}$

Usualmente se considera la distribución normal (gaussiana) o bien la distribución uniforme, en cuyo caso coniserando esta última

$$\sigma_W = \sqrt{\frac{6}{n_{in} + n_{out}}}$$

Ya que la varianza de una distribución uniforme $U[-a, a]$ está dada como $\frac{a^2}{3}$.

A estos parámetros de inicialización se les conoce como **inicialización de Xavier**, o **inicialización de Glorot**, por uno de sus autores.

Vecindades / Máscaras / Filtros / Kernels

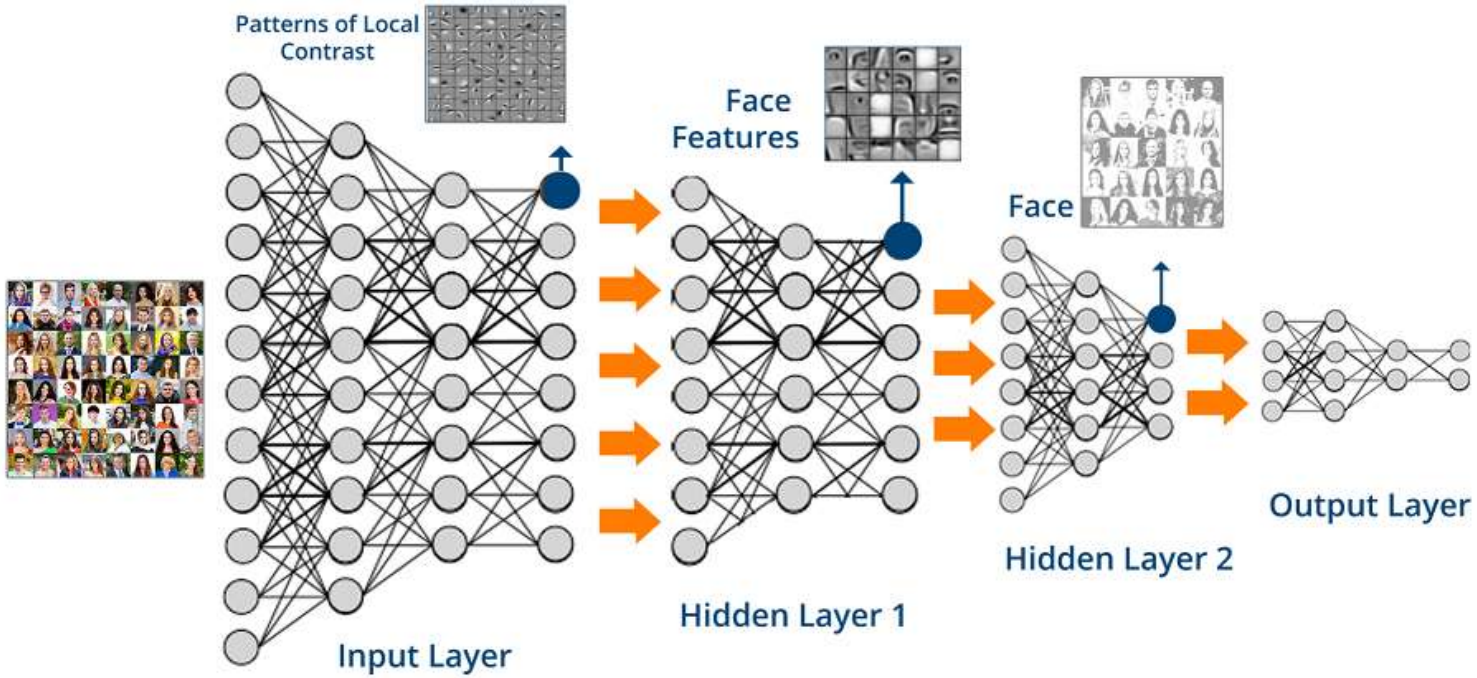
Operación de Convolución

$$g(x,y) = \sum_{a=-1}^{+1} \sum_{b=-1}^{+1} f(x-a, y-b) w(a,b)$$

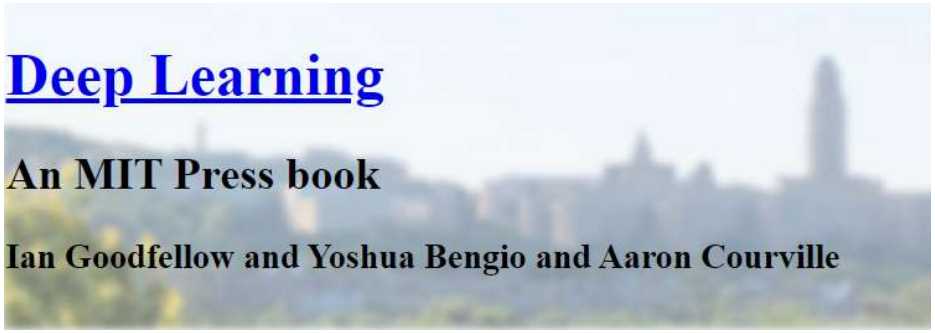
−1	0	1
−1	0	1
−1	0	1

$w(x,y)$

Redes Profundas:
Red Neuronal Convolucional
(Convolutional Neural Network : CNN)



CNN : pixel → contorno → textura → rasgos → partes → objeto



https://www.deeplearningbook.org/ 2016

Chapter 14

Autoencoders

En el primer libro sobre Aprendizaje Profundo (escrita por parte del equipo de los fundadores de dicha área), encuentras mayor información sobre los llamados "autoencoders", es decir, una red neuronal artificial que es entrenada para que aprenda a copiar los datos de entrada a la salida.

Los autoencoders se basan en el concepto de ir separando las tareas de una red neuronal artificial, en la parte de codificación (encoder) y descodificación (decoder). Este proceso se propuso para ir atacando el problema de preguntas y respuestas del área de Procesamiento de Lenguaje Natural (NLP)

[L] 1 May 2014

Auto-Encoding Variational Bayes

Diederik P. Kingma
Machine Learning Group
Universiteit van Amsterdam
dpkingma@gmail.com

Max Welling
Machine Learning Group
Universiteit van Amsterdam
welling.max@gmail.com

Abstract

How can we perform efficient inference and learning in directed probabilistic models, in the presence of continuous latent variables with intractable posterior distributions, and large datasets? We introduce a stochastic variational inference and learning algorithm that scales to large datasets and, under some mild differentiability conditions, even works in the intractable case. Our contributions is two-fold. First, we show that a reparameterization of the variational lower bound

https://arxiv.org/abs/1312.6114

[CL] 3 Sep 2014

Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation

Kyunghyun Cho
Bart van Merriënboer **Caglar Gulcehre** Université de Montréal
firstname.lastname@umontreal.ca

Dzmitry Bahdanau
Jacobs University, Germany
d.bahdanau@jacobs-university.de

Fethi Bougares **Holger Schwenk**
Université du Maine, France
firstname.lastname@lium.univ-lemans.fr

Yoshua Bengio
Université de Montréal, CIFAR Senior Fellow
find.me@on.the.web

https://arxiv.org/abs/1406.1078

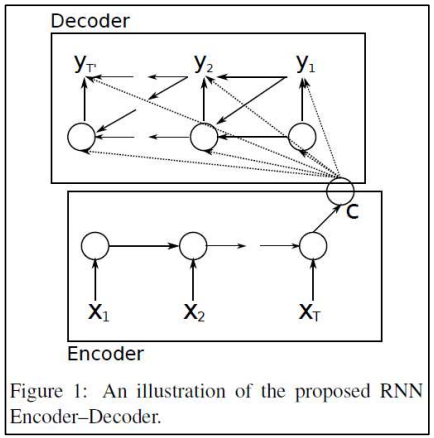


Figure 1: An illustration of the proposed RNN Encoder–Decoder.

En este artículo se propone el modelo **Encoder-Decoder** basado en dos redes neuronales profundas llamadas RNN y LSTM.

ImageNet Classification with Deep Convolutional Neural Networks

Alex Krizhevsky
University of Toronto
kriz@cs.utoronto.ca

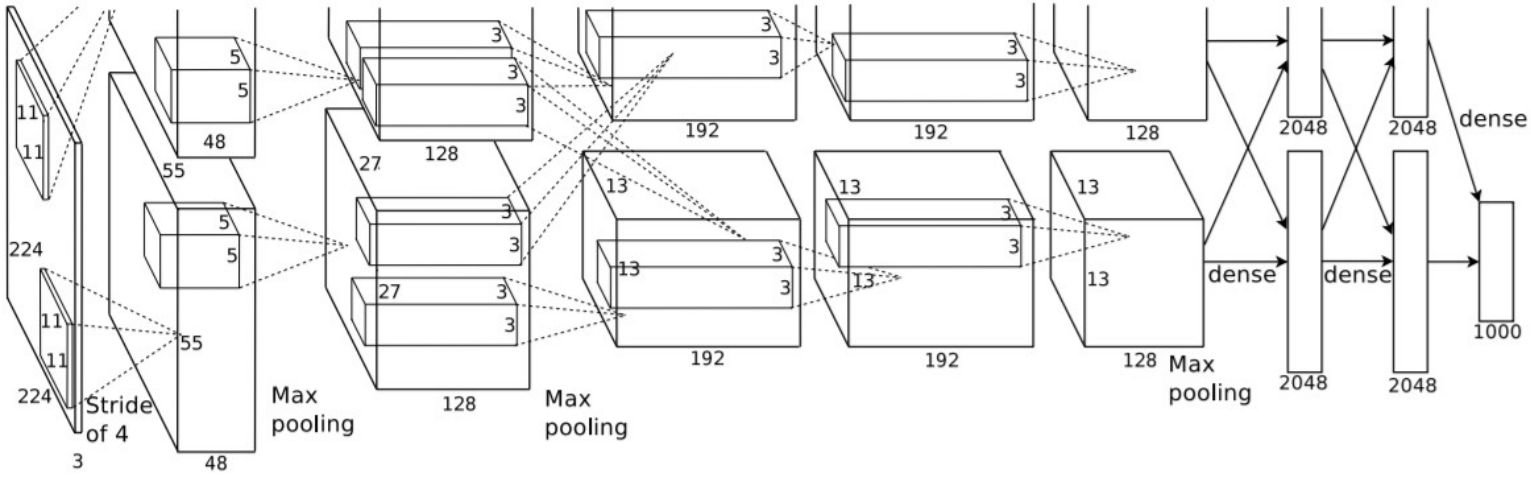
Ilya Sutskever
University of Toronto
ilya@cs.utoronto.ca

Geoffrey E. Hinton
University of Toronto
hinton@cs.utoronto.ca

Abstract

We trained a large, deep convolutional neural network to classify the 1.2 million high-resolution images in the ImageNet LSVRC-2010 contest into the 1000 different classes. On the test data, we achieved top-1 and top-5 error rates of 37.5% and 17.0% which is considerably better than the previous state-of-the-art. The neural network, which has 60 million parameters and 650,000 neurons, consists of five convolutional layers, some of which are followed by max-pooling layers, and three fully-connected layers with a final 1000-way softmax. To make train-

Artículo del 2012 donde se expone la arquitectura conocida como AlexNet.



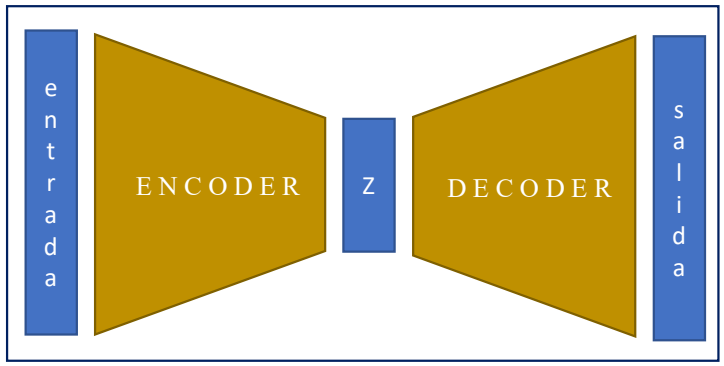
https://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf

Autoencoder: Encoder vs Decoder

El autoencoder se basa en una sucesión de dos bloques llamados **Encoder** (codificador) y **Decoder** (decodificador).

El bloque Encoder se encarga de reducir la alta dimensionalidad de los datos de entrada a una de menor dimensión y cuyas variables son llamadas **latentes**. El bloque Decoder se encarga de transformar de nuevo el espacio de variables latentes a uno de mayor dimensión, en general a uno de la misma dimensión que los datos de entrada.

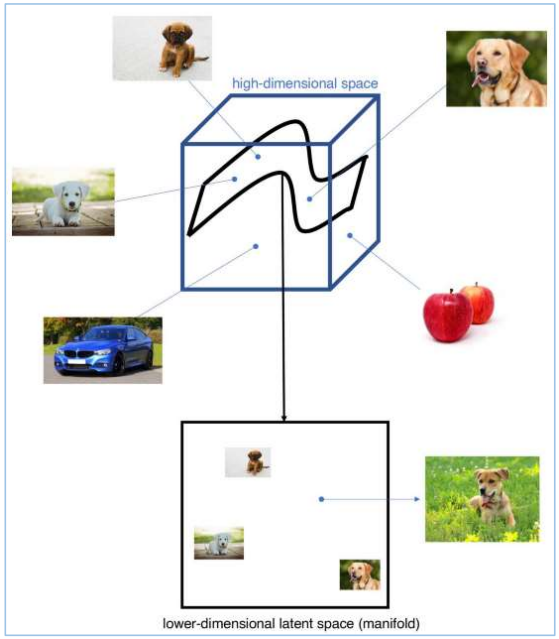
Podemos ilustrar la arquitectura del autoencoder como sigue:



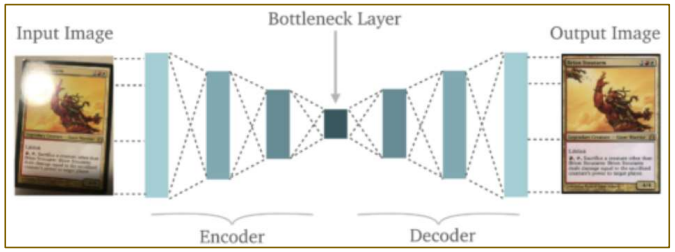
Arquitectura genérica del autoencoder, donde Z son los vectores latentes.

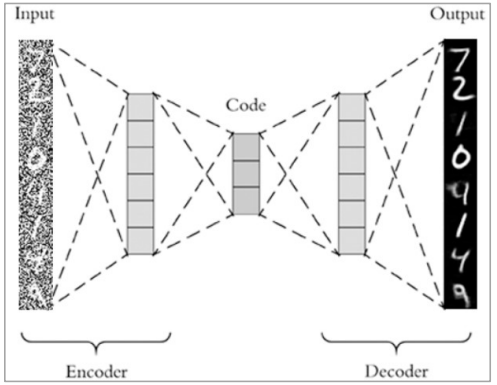
En general, los modelos de aprendizaje de máquinas (machine learning) suponen que los datos de entrada se concentran en una variedad (manifold) de menor dimensionalidad.

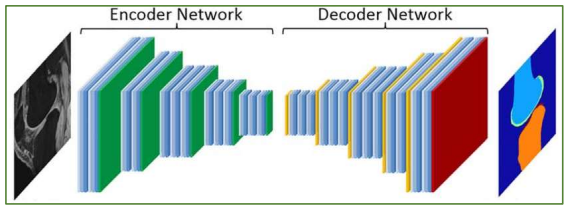
Los autoencoders proponen el mismo supuesto, pero además desaen aprender la estructura y características que describan dichas variedades (manifolds).

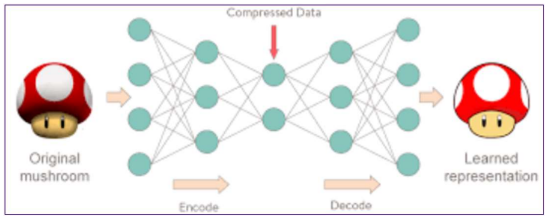


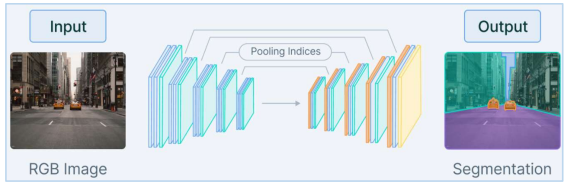
El autoencoder desea por ejemplo encontrar la variedad (manifold o espacio latente) de menor dimensión en la cual se encuentra el conjunto de vectores de los perros.

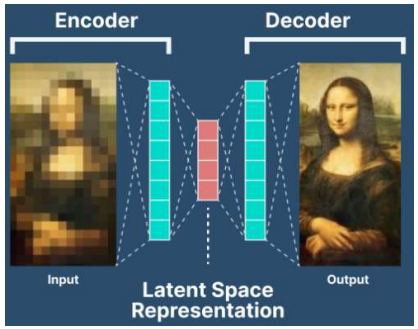

Input Image
Bottleneck Layer
Output Image
Encoder
Decoder


Input
Output
Code
Encoder
Decoder

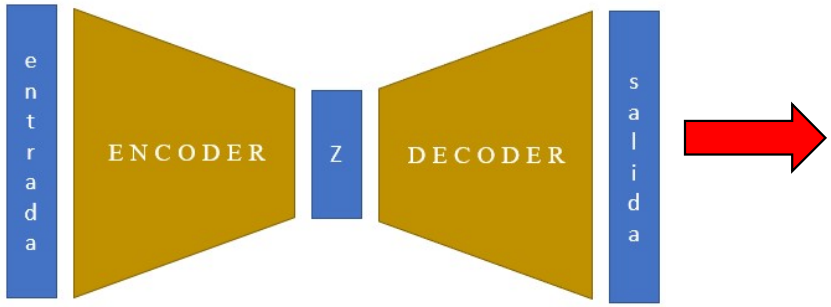

Encoder Network
Decoder Network


Compressed Data
Original mushroom
Learned representation
Encode
Decode


Input
Output
Pooling Indices
RGB Image
Segmentation


Encoder
Decoder
Input
Output
Latent Space Representation

La función de costo se aplica pixel a pixel de manera individual, entre la imagen de entrada y la reconstrucción de la imagen de salida.



Función de Costo:

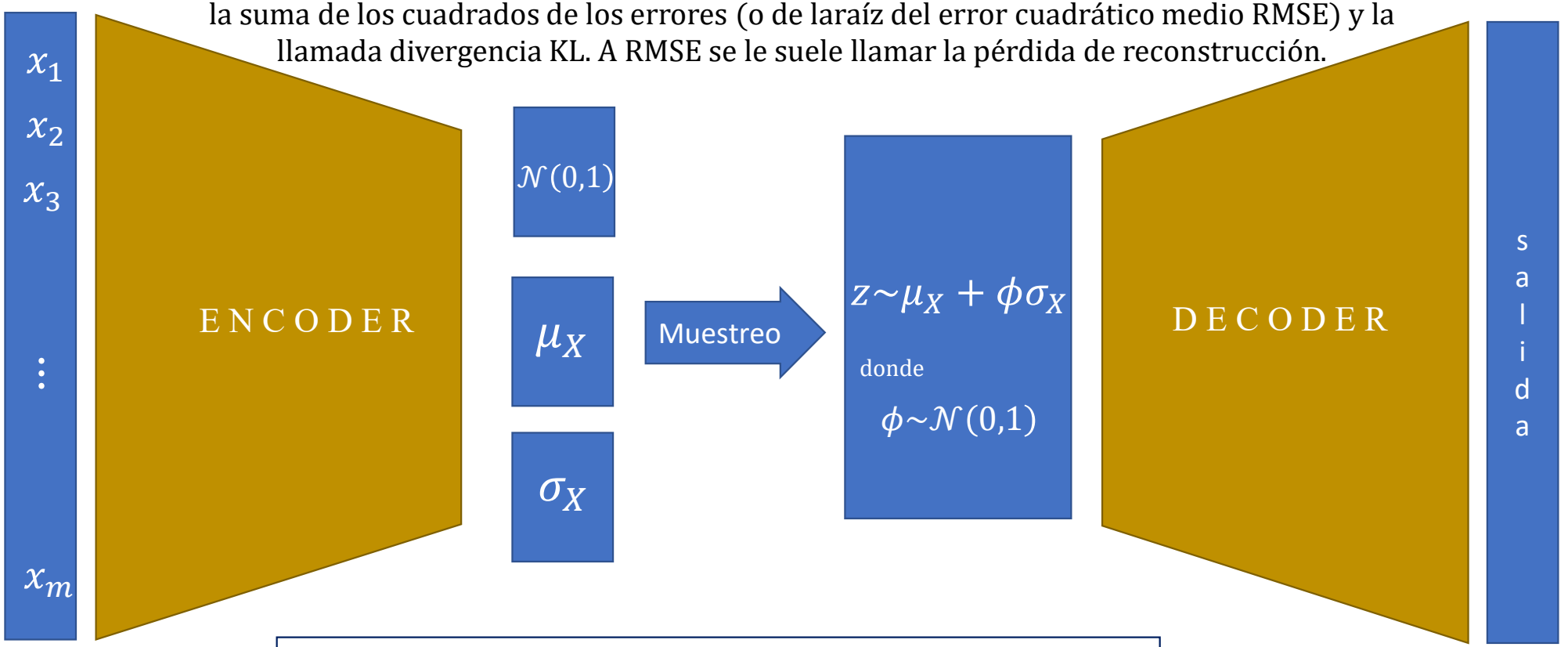
- RMSE: Root mean squared error
- Binary cross-entropy

Observa que pueden utilizarse tanto una función de costo de regresión, como una de clasificación.

Se les suele llamar métodos autosupervisados, ya que aunque no existe etiquetas, los datos de entrada son el objetivo a alcanzar.

Cuando se usa la binary cross-entropy, se castiga más fuerte a las predicciones más erróneas, lo cual hace que las predicciones de los pixeles tiendan más al centro y por ello las imágenes de salida menos “vistosas”.

La función de costo de los autocodificadores variacionales VAE suele ser una combinación de la suma de los cuadrados de los errores (o de laraíz del error cuadrático medio RMSE) y la llamada divergencia KL. A RMSE se le suele llamar la pérdida de reconstrucción.



X

$\hat{X} = d(z)$

Función de Costo:

$$\mathcal{L} = \left\|X - \hat{X}\right\|^2 + D_{KL}\left(\mathcal{N}(\mu_X, \sigma_X) \,||\, \mathcal{N}(0,1)\right)$$

$$= \|X - d(z)\|^2 + D_{KL}\left(\mathcal{N}(\mu_X, \sigma_X) \,||\, \mathcal{N}(0,1)\right)$$

Encoder

- El bloque Encoder está formado por bloques convolucionales que disminuyen la dimensión de los datos dentrada hasta el llamado espacio de variables latentes.
- Una vez llegado al espacio de variables latentes se espera que la información superflua del conjunto de datos de entrada haya sido descartada y ahora se tenga la información más relevante que mejor describe a los datos, pero en un espacio de menor dimensión.
- Los vectores del espacio latente también se dice que son vectores de características comprimidas.
- El modelo aprende o extrae información útil de los vectores latentes. Los “mejores” vectores con la mejor información de los datos de entrada se encuentran en los vectores latentes.
- Es decir, el Encoder es el proceso en el cual los datos de entrada se “descomponen”, se “fragmentan” y se proyectan en un espacio de menor dimensión.
- De aquí que se utilice también como un método de reducción de dimensionalidad.

Decoder

El proceso de decodificación es un tipo de reconstrucción de los datos originales, a partir de los vectores latentes.

Este proceso de reconstrucción del espacio latente a los datos originales no se basa en etiquetas, como en el aprendizaje supervisado, sino tratando de “aprender” las características de la variedad (manifold) que hace particular a cada dato en cuestión y que permiten reconstruir de la mejor manera los datos a su espacio original. Por ello a veces se dice que estos métodos son autosupervisados, ya que podemos decir que los datos de entrada mismo juegan el papel de etiquetas.

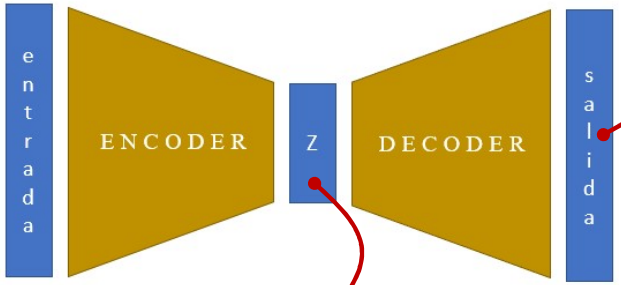
Así, el bloque Decoder realiza el proceso inverso del Encoder, donde ahora los vectores latentes se transforman incrementando su dimensión hasta llegar a igualar la dimensión de los vectores de entrada originales.

Sin embargo, las capas convolucionales del Decoder no tienen que ser necesariamente iguales y en orden inverso a los del bloque Encoder.

Al incrementar la dimensión de los mapas de características (feature maps) en el bloque Decoder, se utilizan en general técnicas de interpolación entre pixeles, que suavice y mejore la resolución del nuevo mapa de características.

Aplicaciones de Autoencoders

- Sistemas de recomendación

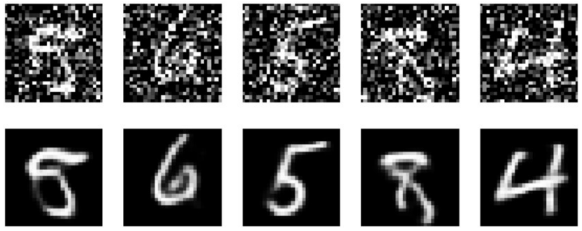


Recomienda temas no conocidos

Captura el interés de los usuarios

Aplicaciones de Autoencoders

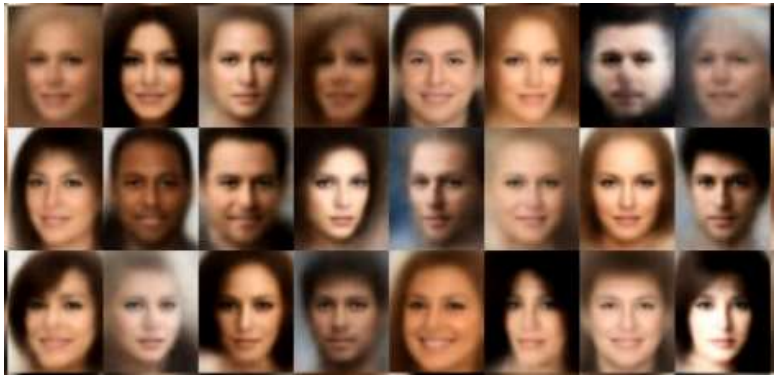
- Reducción de ruido en imágenes



- Coloreado de imágenes

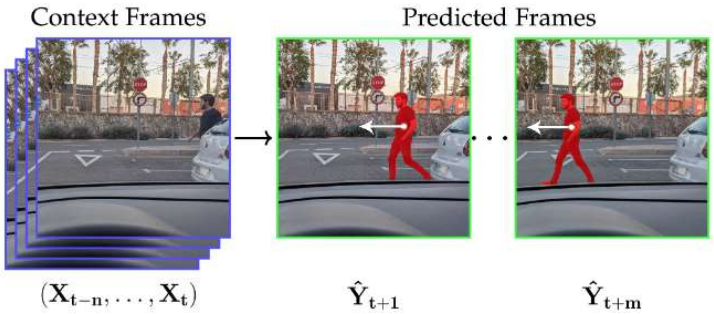
Aplicaciones de Autoencoders

- Generación de imágenes



Aplicaciones de Autoencoders

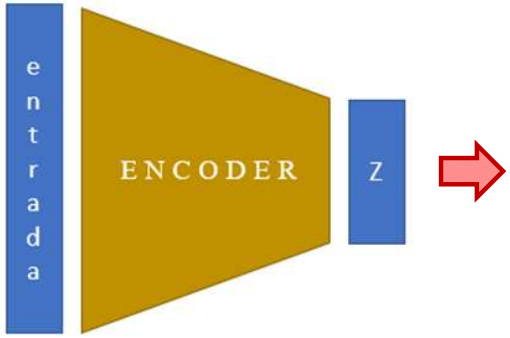
- Sequence-to-Sequence prediction con LSTM-autoencoders: video



- Generación de fake-videos

Aplicaciones de Autoencoders

- **Reducción de dimensionalidad**
- Extracción de características



Entrada a otros modelos….

En esta semana aplicaremos los Autoencoders para la reducción de dimensionalidad en combinación con las técnicas de conglomerados (clustering).

Imágenes de los dígitos MNIST proyectados en un espacio latente de dimensión 2 mediante un autoencoder definido como lo hemos descrito hasta ahora.