Curso DM
Redes Neuronales

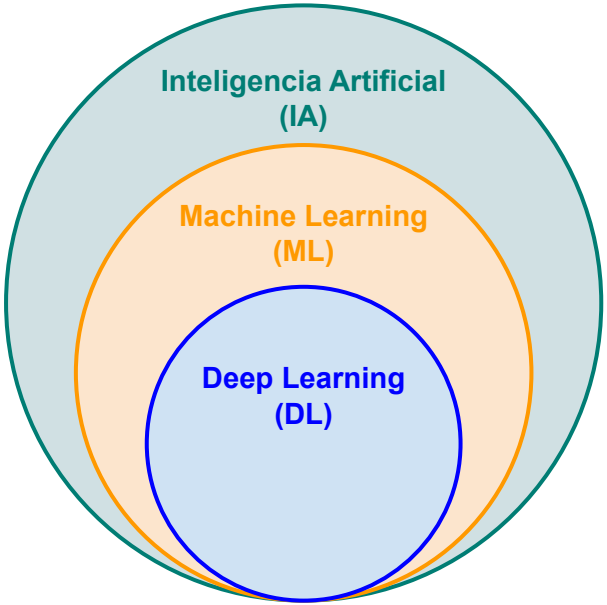
Primavera 2023

Profesoras: Jazmine Maldonado y Cinthia Sánchez

Inteligencia Artificial

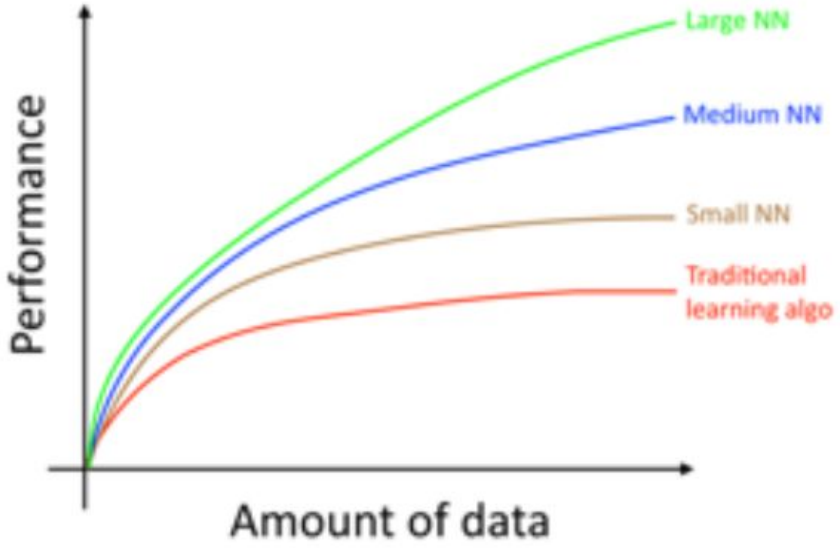
RAE: “Disciplina científica que se ocupa de crear programas informáticos que ejecutan operaciones comparables a las que realiza la mente humana, como el aprendizaje o el razonamiento lógico.”

ML: Estudio, diseño y desarrollo de algoritmos que permiten a los computadores aprender sin ser explícitamente programados (Arthur Samuel). Técnicas genéricas, aplicables a varios dominios. **DL (representation learning):** Algoritmos de ML basados en redes neuronales artificiales (ANN) profundas.



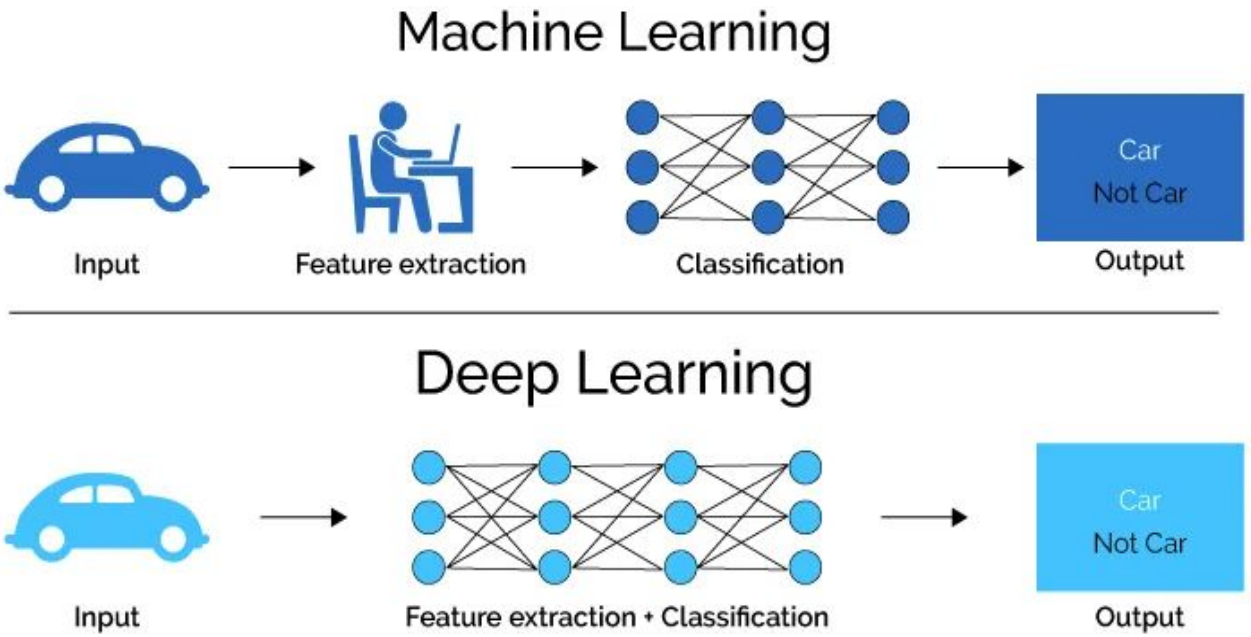
Jorge Pérez. CC6204 Deep Learning. Canal youtube

Performance vs Cantidad de datos



Andrew Ng, Deeplearning.ai course on Coursera. https://www.oreilly.com/api/v2/epubs/9781492083443/files/assets/gsa2_0101.png

¿Cuál es la diferencia entre ML y DL?



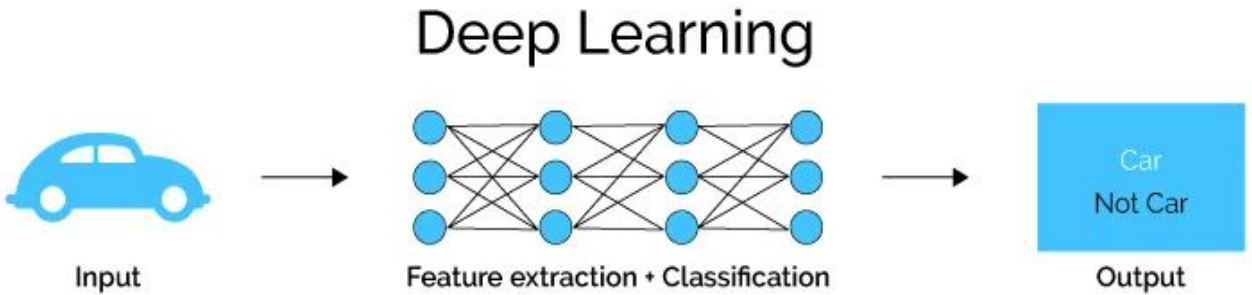
https://lawtomated.com/a-i-technical-machine-vs-deep-learning/

**Como se mencionó, DL es un subconjunto de ML. Esta pregunta hace referencia a ML sin DL vs DL.*

¿Cuál es la diferencia entre ML y DL?

DL: Algoritmos de ML basados en redes neuronales artificiales (ANN) profundas.

ANN: Modelos de aprendizaje automático muy populares formados por unidades llamadas **neuronas**.

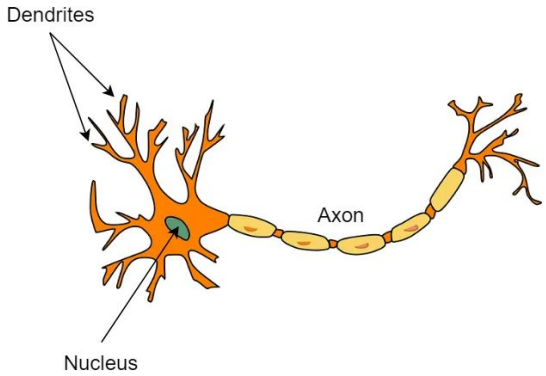


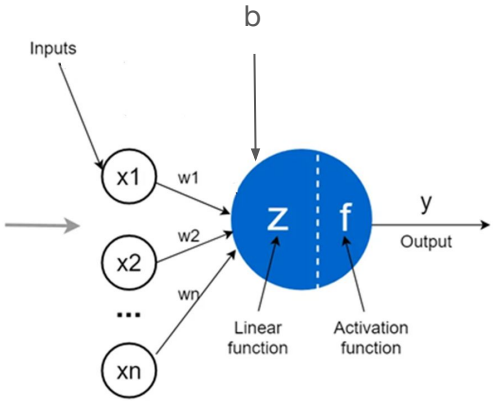
https://lawtomated.com/a-i-technical-machine-vs-deep-learning/

**Como se mencionó, DL es un subconjunto de ML. Esta pregunta hace referencia a ML sin DL vs DL.*

Neurona artificial

- El modelo de neurona artificial es una simplificación de una neurona (natural) que representa la activación de una neurona como una combinación lineal de los estímulos recibidos.
- Una neurona es una unidad computacional que tiene entradas y salidas escalares.
- Cada entrada tiene un peso asociado ***w***.
- La neurona multiplica cada entrada por su peso y luego las suma (se pueden usar otras funciones de agregación como max). (Cada neurona es un modelo lineal con una función de activación).
- Luego aplica una función de activación **g** (generalmente no lineal) al resultado, y la pasa a su salida.





$$z = x_1 w_1 + x_2 w_2 + \ldots + x_n w_n + b$$

$$y = f(z)$$

https://towardsdatascience.com/

Función de activación

Las funciones de activación desempeñan un papel importante en la transformación de la salida lineal en no lineal, lo que permite que los modelos aprendan patrones complejos de manera eficiente. Si quitamos la no-linealidad aportada por la función de activación, la red neuronal sólo podría representar transformaciones lineales de la entrada.

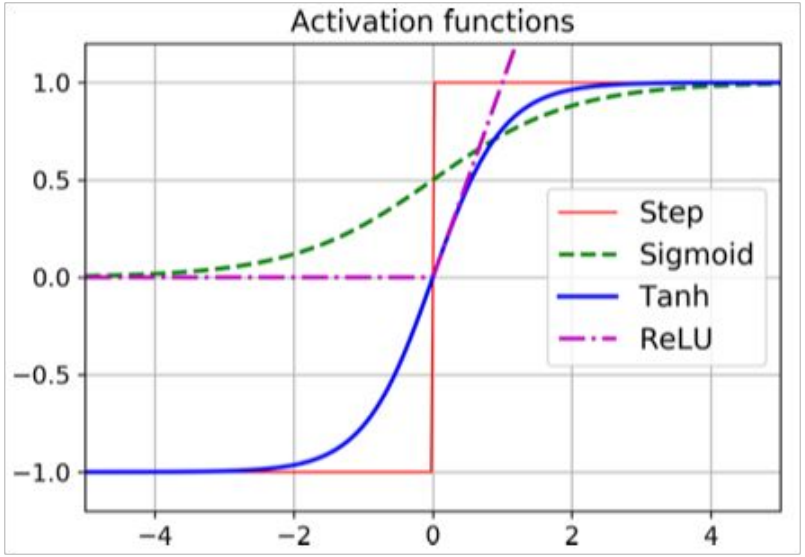
Step
basada en umbral

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

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

ReLU
$\max(0, x)$

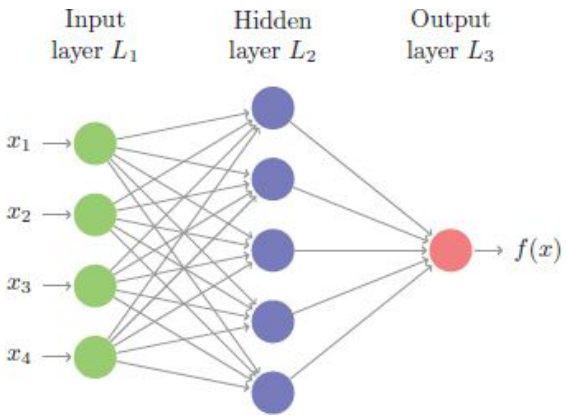
Se utilizan principalmente en las capas ocultas.



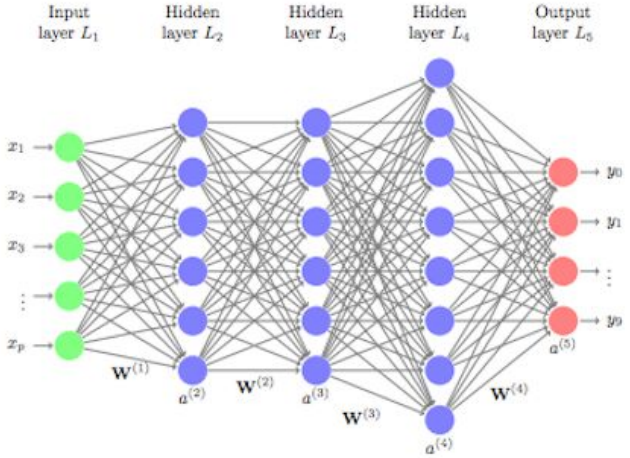
https://kinder-chen.medium.com/hidden-layer-activation-functions-6fd65489ed25

Redes neuronales artificiales

- Son modelos de aprendizaje automático formados por unidades llamadas **neuronas**, capaces de aprender relaciones no-lineales entre x y y.
- Varias neuronas pueden agruparse en una capa de neuronas.
- La salida de una capa puede pasarse como input a otra capa de forma de cascada.
- A este tipo de redes se les conoce como Feedforward Networks o Multi-Layer Perceptron.



Simple feedforward neural network



Deep feedforward neural network

https://afit-r.github.io/feedforward_DNN

Función de salida (para mapear logits a distribuciones)

Va a depender del problema a resolver. Interpreto la salida como una distribución de probabilidad sobre las clases. Hay varias opciones, pero suele usarse Softmax (salida rango 0-1, todos suman 1).

SoftMax

k>2 classes

$$\sigma(x_j) = \frac{e^{x_j}}{\sum_i e^{x_i}}$$

$$\text{out} = \begin{bmatrix} P(Y=class1|X) \\ P(Y=class2|X) \\ P(Y=class3|X) \\ \vdots \\ P(Y=classk|X) \end{bmatrix}$$

https://towardsdatascience.com/sigmoid-and-softmax-functions-in-5-minutes-f516c80ea1f9

Función de salida (para mapear logits a distribuciones)

Va a depender del problema a resolver. Interpreto la salida como una distribución de probabilidad sobre las clases. Hay varias opciones, pero suele usarse Softmax (salida rango 0-1, todos suman 1).

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

Sigmoid

2 classes

out = P(Y=class1|X)

SoftMax

k>2 classes

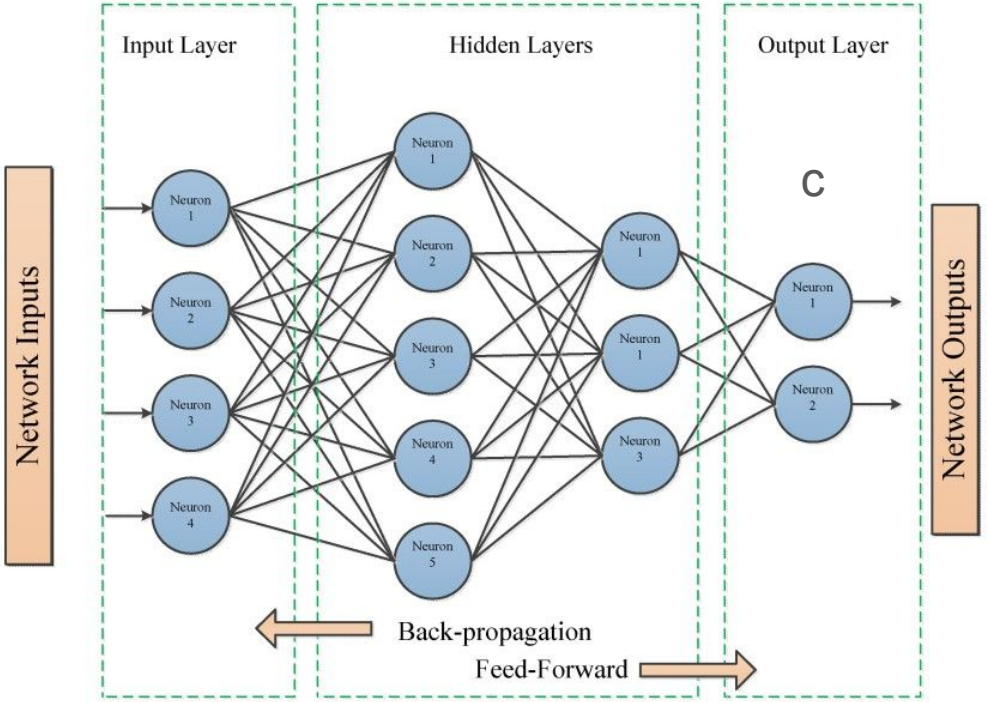
$$\sigma(x_j) = \frac{e^{x_j}}{\sum_i e^{x_i}}$$

$$\text{out} = \begin{bmatrix} P(Y=class1|X) \\ P(Y=class2|X) \\ P(Y=class3|X) \\ \vdots \\ P(Y=classk|X) \end{bmatrix}$$

https://towardsdatascience.com/sigmoid-and-softmax-functions-in-5-minutes-f516c80ea1f9

Descenso de Gradiente para encontrar los parámetros

Dados los datos ($D = \{(x^1, y^1), (x^2, y^2), \ldots, (x^n, y^n)\}$ y un los parámetros de la red $\theta = (W^1, b^1, \ldots, W^L, b^L, c)$



$\hat{y}^i$ y^i = gato

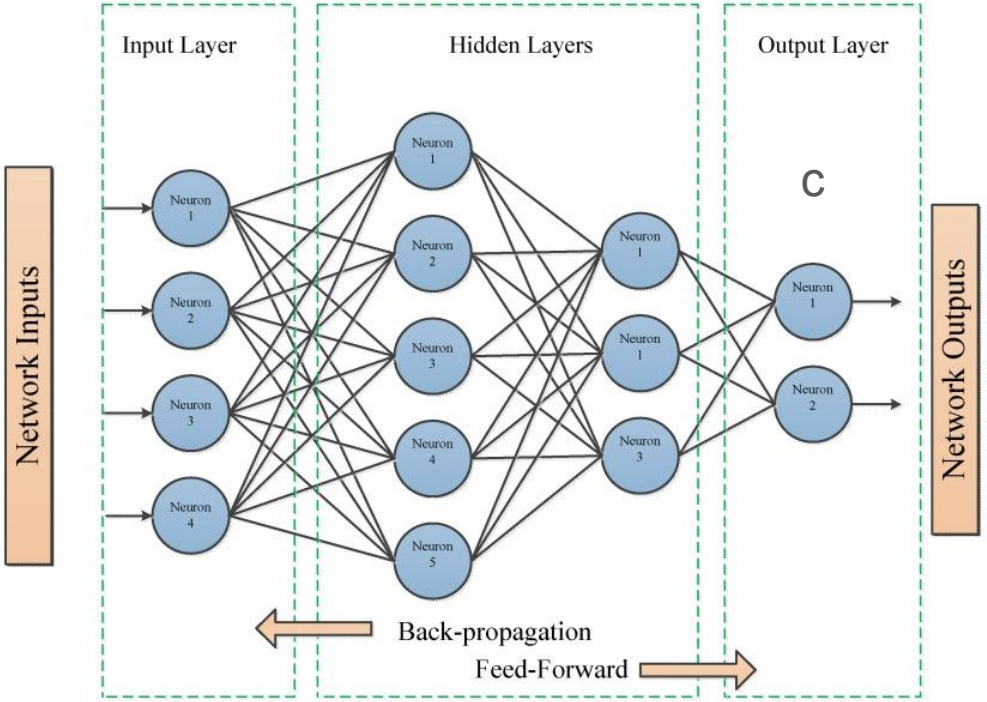
$$L = \frac{1}{n} \sum_{i=1}^{n} \text{error}(\hat{y}^i, y^i)$$

Error (o loss)

https://www.linkedin.com/pulse/feedforward-vs-backpropagation-ann-saffronedge1/

Descenso de Gradiente para encontrar los parámetros

Dados los datos ($D = \{(x^1, y^1), (x^2, y^2), \ldots, (x^n, y^n)\}$ y un los parámetros de la red $\theta = (W^1, b^1, \ldots, W^L, b^L, c)$



$$L = \frac{1}{n} \sum_{i=1}^{n} \text{error}(\hat{y}^i, y^i)$$

$$\underset{\theta}{\text{argmin}} \; L(\theta)$$

$\hat{y}^i$

Objetivo de entrenar la red: Encontrar θ que minimiza la función de error.

https://www.linkedin.com/pulse/feedforward-vs-backpropagation-ann-saffronedge1/

Funciones de costo

El objetivo del entrenamiento es minimizar la pérdida en los datos de entrenamiento.

Para clasificación: Ej: CrossEntropy, que mide distancia entre dos distribuciones de probabilidades.

$$L(\hat{y}, y) = -\sum_{k}^{K} y^{(k)} \log \hat{y}^{(k)}$$

Para regresión: Ej: Error cuadrático medio (MSE), que es el SSE normalizado por la cantidad de ejemplos.

$$\text{MSE} = \frac{1}{N} \sum_{i=1}^{N} (y_i - \hat{y}_i)^2$$

Descenso del Gradiente

- El modelo se puede entrenar usando métodos iterativos basados en **gradientes**.
- Se calculan los gradientes de los parámetros con respecto a la pérdida **L**, y se mueven los parámetros en las direcciones opuestas del gradiente.
- Diferentes métodos de **optimización** difieren en cómo se calcula la estimación del error y cómo se define el movimiento en la dirección opuesta al gradiente.
- El algoritmo de **backpropagation** (backward pass) esencialmente sigue la regla de la cadena de derivación.

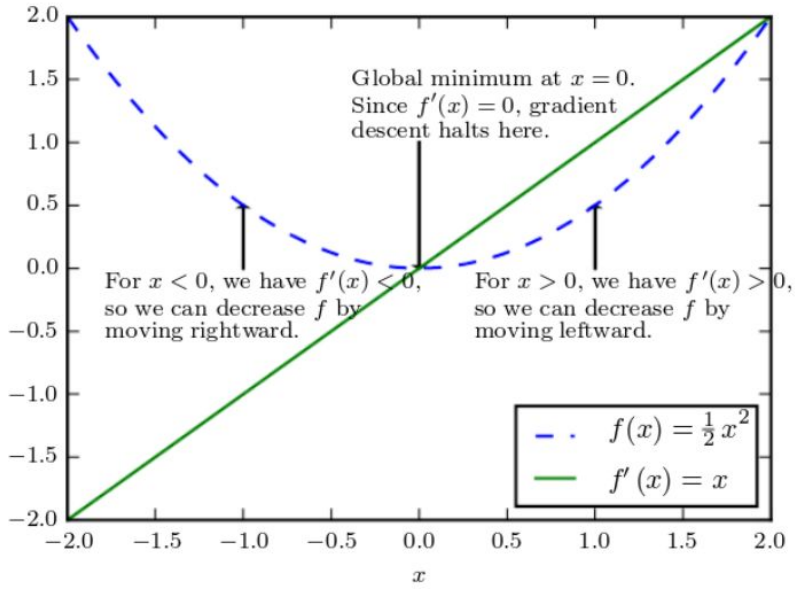


Figure 4.1: Gradient descent. An illustration of how the gradient descent algorithm uses the derivatives of a function to follow the function downhill to a minimum.

Fuente imagen: L. Wasserman All of Statistics: A Concise Course in Statistical Inference, Springer Texts in Statistics, 2005.

Descenso de Gradiente por Mini-batches

- SGD es susceptible al ruido inducido por un único ejemplo (un outlier puede mover mucho el gradiente).
- Una forma común para reducir este ruido es estimar el error y los gradientes sobre muestras de **m** ejemplos.
- Esto se llama **minibatch** SGD.
- Valores grandes para **m** dan una mejor estimación del gradiente en base al dataset completo, mientras que valores más pequeños permiten realizar más actualizaciones y **converger más rápido**.
- Para tamaños razonables de **m**, algunas arquitecturas computacionales (i.e., **GPUs, TPUs**) permiten paralelizar SGD eficientemente (es la única forma de entrenar redes de muchos parámetros en tiempo razonable).

Un muy buen tutorial del algoritmo backpropagation usando la abstracción del grafo de cómputo: https://colah.github.io/posts/2015-08-Backprop/

Aplicabilidad versus interpretabilidad

- Aumentando el número de capas y de neuronas por capas, podemos conseguir estructuras más y más complejas.
- Sin embargo, la red es un modelo general requiere entrenamiento, es decir, encontrar los parámetros.
- Para redes grandes, encontrar los parámetros es un desafío.
- Los avances recientes en DL han permitido entrenar redes con millones (e incluso billones) de parámetros.

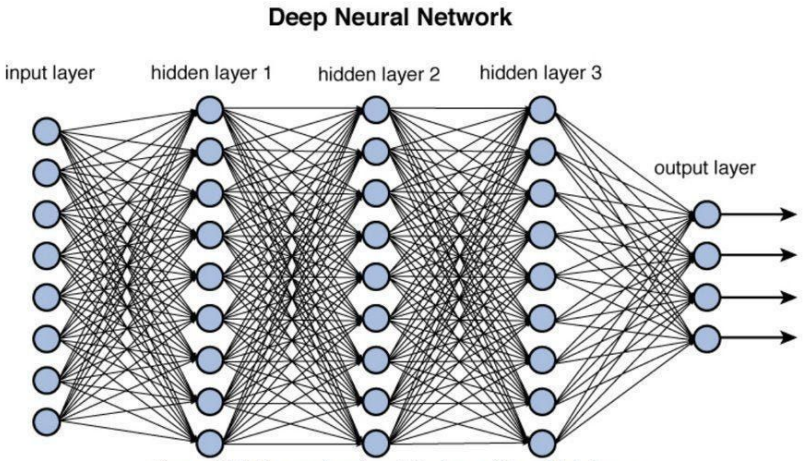


Figure 12.2 Deep network architecture with multiple layers.

https://towardsdatascience.com/training-deep-neural-networks-9fdb1964b964

Redes neuronales artificiales

- Las redes neuronales son modelos muy poderosos para regresión y clasificación.
- El uso de redes con varias capas se llama popularmente como **Deep Learning**.
- Existen arquitecturas de red que son buenas para aprender representaciones sobre datos complejos: texto, imágenes, audio, video.
- Arquitecturas famosas: redes neuronales convolucionales, redes recurrentes y redes de atención.
- La alta capacidad de estas redes las hace muy proclives al *overfitting*. Hay varias técnicas para mitigarlo: regularización, drop-out, batch normalization.