

Hoja de referencia VIP: Aprendizaje profundo

Afshine AMIDI y Shervine AMIDI

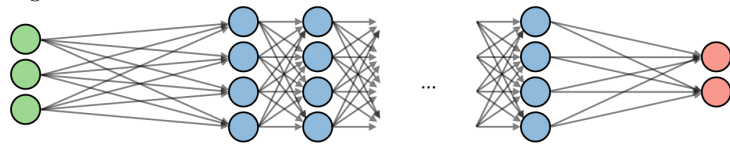
6 de octubre de 2018

Traducido por Erick Gabriel Mendoza Flores. Revisado por Fernando González-Herrera, Mariano Ramírez, Juan P. Chavat, Alonso Melgar López, Gustavo Velasco-Hernández, Juan Manuel Nava Zamudio y Fernando Díaz.

Redes neuronales

Las redes neuronales (en inglés, *Neural Networks*) son una clase de modelos construidos a base de capas. Los tipos más utilizados de redes neuronales incluyen las redes neuronales convolucionales y las redes neuronales recurrentes.

□ **Arquitectura** – El vocabulario en torno a arquitecturas de redes neuronales se describe en la siguiente figura:



Capa de entrada Capa oculta 1 ... Capa oculta k Capa de salida

Denotando i en la i -ésima capa de la red y j en la j -ésima unidad oculta de la capa, tenemos:

$$z_j^{[i]} = w_j^{[i]T} x + b_j^{[i]}$$

donde w , b y z son el peso, el sesgo y la salida, respectivamente.

□ **Función de activación** – Las funciones de activación son utilizadas al final de una unidad oculta para introducir complejidades no lineales al modelo. A continuación las más comunes:

Sigmoide	Tanh	ReLU	Leaky ReLU
$g(z) = \frac{1}{1 + e^{-z}}$	$g(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$	$g(z) = \max(0, z)$	$g(z) = \max(\epsilon z, z)$ with $\epsilon \ll 1$

□ **Pérdida de cross-entropy** – En el contexto de las redes neuronales, la pérdida de cross-entropy $L(z, y)$ es utilizada comúnmente y definida de la siguiente manera:

$$L(z, y) = - \left[y \log(z) + (1 - y) \log(1 - z) \right]$$

□ **Velocidad de aprendizaje** – La velocidad de aprendizaje (en inglés, *Learning rate*), denotada como α o algunas veces η , indica a qué ritmo los pesos son actualizados. Este valor puede ser fijo o cambiar de forma adaptativa. El método más popular en este momento es llamado Adam, que es un método que adapta a la velocidad de aprendizaje.

□ **Retropropagación** – La retropropagación (en inglés, *Backpropagation*), o propagación inversa, es un método de actualización de los pesos en una red neuronal, teniendo en cuenta la salida actual y la salida esperada. La derivada respecto al peso w es calculada utilizando la regla de la cadena y se expresa de la siguiente forma:

$$\frac{\partial L(z, y)}{\partial w} = \frac{\partial L(z, y)}{\partial a} \times \frac{\partial a}{\partial z} \times \frac{\partial z}{\partial w}$$

Como resultado, el peso es actualizado de la siguiente forma:

$$w \leftarrow w - \eta \frac{\partial L(z, y)}{\partial w}$$

□ **Actualizando pesos** – En una red neuronal, los pesos son actualizados de la siguiente forma:

- **Paso 1** : Tomar un lote de los datos de entrenamiento.
- **Paso 2** : Realizar propagación hacia adelante para obtener la pérdida correspondiente.
- **Paso 3** : Propagar inversamente la pérdida para obtener los gradientes.
- **Paso 4** : Utiliza los gradientes para actualizar los pesos de la red.

□ **Retiro** – El retiro (en inglés, *Dropout*) es una técnica para prevenir el sobreajuste de los datos de aprendizaje descartando unidades en una red neuronal. En la práctica, las neuronas son retiradas con una probabilidad de p o se mantienen con una probabilidad de $1 - p$.

Redes neuronales convolucionales

□ **Requisito de la capa convolucional** – Notando que W es el volumen de la entrada, F el tamaño de las neuronas de la capa convolucional, P la cantidad de relleno con ceros, entonces el número de neuronas N que entran en el volumen dado es tal que:

$$N = \frac{W - F + 2P}{S} + 1$$

□ **Normalización por lotes** – Es un paso de hiperparámetro γ, β que normaliza el grupo $\{x_i\}$. Denotando μ_B, σ_B^2 la media y varianza del lote que queremos corregir, se realiza de la siguiente manera:

$$x_i \leftarrow \gamma \frac{x_i - \mu_B}{\sqrt{\sigma_B^2 + \epsilon}} + \beta$$

Se realiza usualmente después de una capa completamente conectada/convolucional y antes de una capa no-lineal y su objetivo es permitir velocidad más altas de aprendizaje y reducir su fuerte dependencia sobre la inicialización.

Redes neuronales recurrentes

□ **Tipos de puerta** – A continuación, tenemos los diferentes tipos de compuertas que encontramos en una red neuronal recurrente típica:

Puerta de entrada	Puerta de olvido	Puerta	Puerta de salida
¿Escribir?	¿Borrar?	¿Cuánto escribir?	¿Cuánto revelar?

□ **LSTM** – Una red de memoria a corto y largo plazo (en inglés, *Long Short Term Memory*) es un tipo de modelo de red neuronal recurrente que evita el problema del desvanecimiento del gradiente añadiendo puertas de 'olvido'.

Aprendizaje por refuerzo

El objetivo del aprendizaje por refuerzo es hacer que un agente aprenda como evolucionar en un entorno.

□ **Procesos de decisión de Markov** – Un procesos de decisión de Markov (en inglés, *Markov Decision Process*) es una 5-tupla $(\mathcal{S}, \mathcal{A}, \{P_{sa}\}, \gamma, R)$ donde:

- \mathcal{S} es el conjunto de estados
- \mathcal{A} es el conjunto de acciones
- $\{P_{sa}\}$ son las probabilidades de transición de estado para $s \in \mathcal{S}$ y $a \in \mathcal{A}$
- $\gamma \in [0, 1]$ es el factor de descuento
- $R : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$ o $R : \mathcal{S} \rightarrow \mathbb{R}$ es la función recompensa que el algoritmo pretende maximizar

□ **Política** – Una política π es una función $\pi : \mathcal{S} \rightarrow \mathcal{A}$ que asigna estados a acciones.

Observación: decimos que ejecutamos una política π dada si dado un estado s tomamos la acción $a = \pi(s)$.

□ **Función valor** – Para una política dada π y un estado dado s , definimos el valor de la función V^π de la siguiente manera:

$$V^\pi(s) = E \left[R(s_0) + \gamma R(s_1) + \gamma^2 R(s_2) + \dots | s_0 = s, \pi \right]$$

□ **Ecuación de Bellman** – Las ecuaciones óptimas de Bellman, caracterizan la función valor V^{π^*} de la política óptima π^* :

$$V^{\pi^*}(s) = R(s) + \max_{a \in \mathcal{A}} \gamma \sum_{s' \in \mathcal{S}} P_{sa}(s') V^{\pi^*}(s')$$

Observación: denotamos que la política óptima π^ para un estado dado s es tal que:*

$$\pi^*(s) = \operatorname{argmax}_{a \in \mathcal{A}} \sum_{s' \in \mathcal{S}} P_{sa}(s') V^*(s')$$

□ **Algoritmo de iteración valor** – El algoritmo de iteración valor es en dos pasos:

- Inicializamos el valor:

$$V_0(s) = 0$$

- Iteramos el valor con base en los valores de antes:

$$V_{i+1}(s) = R(s) + \max_{a \in \mathcal{A}} \left[\sum_{s' \in \mathcal{S}} \gamma P_{sa}(s') V_i(s') \right]$$

□ **Estimación por máxima verosimilitud** – El estimación de probabilidad máximo para las probabilidades de transición de estado son como se muestra a continuación:

$$P_{sa}(s') = \frac{\text{\# veces que se tomó la acción } a \text{ en el estado } s \text{ y llevó a } s'}{\text{\# veces que se tomó la acción } a \text{ en el estado } s}$$

□ **Q-learning** – Q-learning es una estimación libre de modelo de Q , que se realiza de la siguiente forma:

$$Q(s, a) \leftarrow Q(s, a) + \alpha \left[R(s, a, s') + \gamma \max_{a'} Q(s', a') - Q(s, a) \right]$$