Ingeniería del Conocimiento CIF-8458

Técnicas de Entrenamiento para Redes Neuronales Artificiales

Carlos Valle Vidal

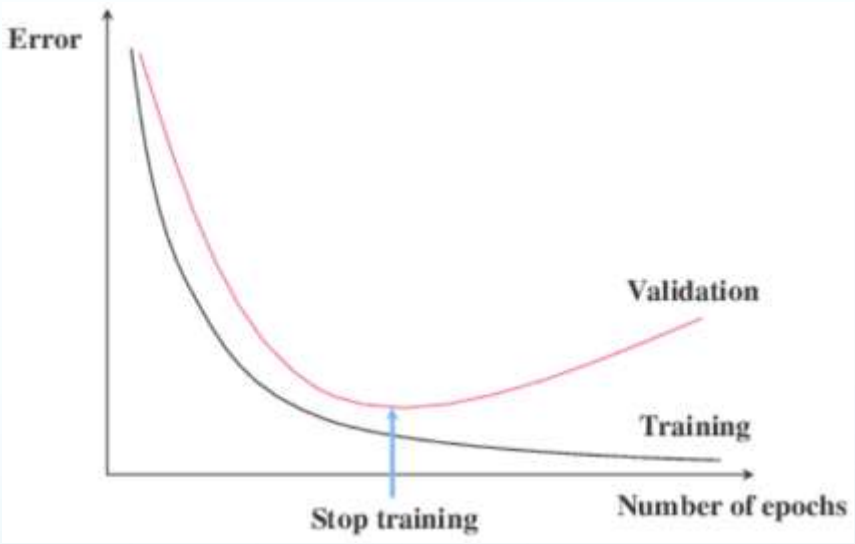
Segundo Semestre 2023

Motivación

- Como hemos visto, el gran número de parámetros que suele tener una red neuronal profunda hace que existan muchos óptimos que tienen el mismo valor para la función de error (o costo) que minimiza el riesgo empírico (error sobre el conjunto de entrenamiento).
- Eso quiere decir, que podríamos estar escoger una red que es muy buena para modelar los datos de entrenamiento, pero que no generaliza bien.
- Como vimos en Inteligencia artificial, podemos usar técnicas de validación para estimar la capacidad de generalización de la red que estamos entrenando.

Early stopping

- Podemos detener el entrenamiento en la iteración (epoch) en el que la red empiece a sobreajustar.



- Podemos definir un factor de paciencia **p**, esto significa, darle la oportunidad a la red de entrenar **p** epochs, y si no ha conseguido mejorar el error de validación el entrenamiento se detiene.

Regularización

- Como vimos en el curso de Inteligencia Artificial, una alternativa para evitar el sobreajuste es penalizar la complejidad del modelo.

$$\text{Costo}(h) = R_{empírico}(h) + \lambda \,\text{complejidad}(h)$$

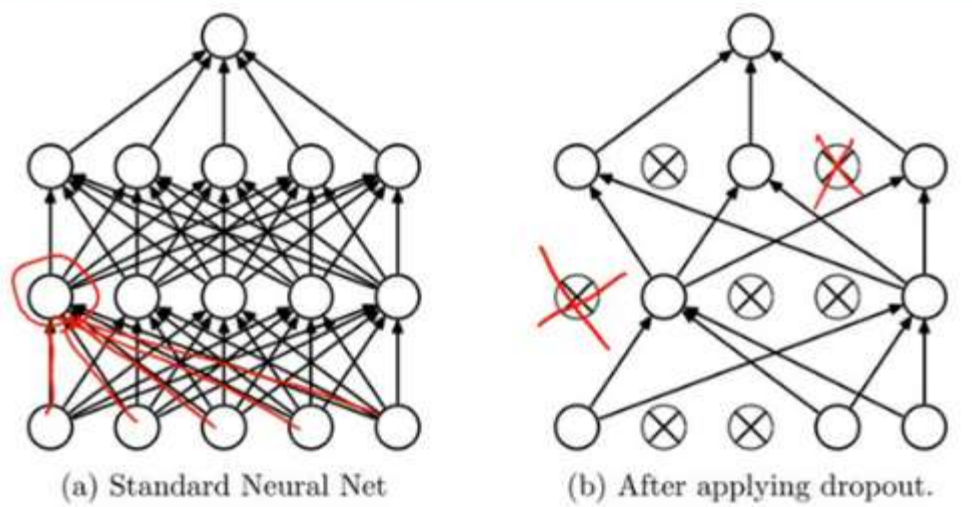
- Donde $\lambda > 0$ se conoce como el parámetro de regularización que controla cuanto consideraremos la complejidad de la hipótesis.

- Entonces el algoritmo de aprendizaje escogerá:

$$\hat{h}^* = \underset{h \in H}{\operatorname{argmin}} \,\text{Cost}(h)$$

Dropout

- Para cada iteración del proceso de entrenamiento, se asigna una probabilidad **p** de descartar cada nodo de la red.



- Dicha probabilidad puede ser igual para toda la red, o distinta en cada capa.

Dropout

- Con esto se trata de lograr que ninguna neurona memorice parte de la entrada, ya que esto nos podría llevar al sobreajuste.
- Dropout es considerado un método de regularización, ya que en cada iteración se reduce la complejidad de la red neuronal.
- El efecto práctico que provoca dropout es que la red tiene que dedicar varias neuronas para aprender un atributo, en lugar de dedicar a una sola neurona por atributo, esto le da mucha más robustez al aprendizaje.

Regularización L2

- La idea detrás de este tipo de regularización es reducir el valor de los parámetros para que sean pequeños.
- En ésta técnica la $\text{complejidad}(h)$ se calcula como la suma de los cuadrados de los parámetros w (pesos y biases).
- Dado que el término que calcula la complejidad puede ser alto, la red podría asignar todos sus parámetros muy cercanos a 0, lo que no sería conveniente.
- Es por ello que el hiperparámetro de regularización λ debe ser pequeño, cuyo valor se escoge usando técnicas de validación.

Regularización L2 (2)

- La función de error de la red se calcula:
- $Error_{L2}(w) = Error_{empírico}(w) + \frac{\lambda}{2} \sum \left({w_{ij}}^{(\ell)^2} \right)$
- Esto hace que los gradientes de la función de error regularizada solo tengan un término extra:
- $\nabla Error_{L2}(w) = \nabla Error_{empírico}(w) + \lambda {w_{ij}}^{(\ell)}$
- En regression lineal, la regularización L2 se conoce como **ridge regression**.

Regularización L1

- Otra alternativa es usar los valores absolutos de los pesos, en lugar de sus cuadrados.
- Esto hace que la solución sea mucho más **sparse**, es decir, muchos pesos pasan a ser exactamente cero.
- La función de error de la red se calcula:
- $Error_{L1}(w) = Error_{empírico}(w) + \lambda \sum \left| w_{ij}^{(\ell)} \right|$

Regularización L1 (2)

- Esto hace que los gradientes de la función de error regularizada solo tengan un término extra:
- $\nabla Error_{L1}(w) = \nabla Error_{empírico}(w) + \lambda \operatorname{sign}(w_{ij}^{(\ell)})$
- En regression lineal, la regularización L1 se conoce como **lasso**.