Tecnológico de Monterrey

Aprendizaje automático

Ajuste de redes neuronales

Dr. Omar Mendoza Montoya

04/09/2024

Aprendizaje basado en descenso de gradiente

Descenso de gradiente

El esquema general para la optimización de los parámetros de varias de las arquitecturas de redes neuronales sigue el clásico algoritmo de descenso de gradiente.

$$W \Leftarrow W - \alpha \nabla L(D)$$

donde D es el conjunto de datos de entrenamiento del modelo, $L(D)$ es la función de costo que compara los valores de la variable de respuesta con los valores predichos por el modelo, W es el vector de tamaño p con los pesos del modelo, α es la razón de aprendizaje (constante pequeña), y $\nabla L(D)$ es el vector de derivadas parciales de la función de costo respecto a cada uno de los parámetros del modelo

Gradiente de la función de costo

$$\nabla L(D) = \begin{bmatrix} \frac{\partial L(D)}{\partial w_1} \\ \frac{\partial L(D)}{\partial w_2} \\ \vdots \\ \frac{\partial L(D)}{\partial w_p} \end{bmatrix}$$

Descenso de gradiente

- Las funciones de costo se pueden escribir como:

$$L(D) = \sum_{i=1}^{n} L_i$$

donde L_i es el costo de la observación i.

- Con ello, podemos escribir el gradiente de la función de costo como:

$$\nabla L(D) = \sum_{i=1}^{n} \nabla L_i$$

donde ∇L_i es el gradiente del costo para la observación i.

- Finalmente, descenso de gradiente quedaría como:

$$W \Leftarrow W - \alpha \sum_{i=1}^{n} \nabla L_i$$

Reglas de aprendizaje para descenso de gradiente

- En lugar de calcular $\nabla L(D)$ para un conjunto grande de datos (descenso de gradiente de lote), existe la posibilidad de actualizar los pesos W tomando una a una las observaciones del conjunto D y calculando el gradiente ∇L_i para dicha observación i (descenso de gradiente estocástico).

- A su vez, otra alternativa para actualizar W consiste en obtener $\nabla L(D_i)$ para un subconjunto $D_i \subset D$, de tal manera que se procesan al mismo tiempo un conjunto reducido de observaciones (descenso de gradiente de mini lote).

Reglas de aprendizaje

- Descenso de gradiente estocástico (stochastic gradiente descent)

$$W \Leftarrow W - \alpha \nabla L_i$$

- Descenso de gradiente de lote (batch gradiente descent).

$$W \Leftarrow W - \alpha \sum_{i=1}^{n} \nabla L_i$$

- Descenso de gradiente de mini lote (mini-batch gradiente descent)

$$W \Leftarrow W - \alpha \sum_{i \in B} \nabla L_i$$

donde B es el conjunto de observaciones del lote.

Derivadas de las funciones de costo

Derivadas de las funciones de costo

- Error cuadrático medio

$$L(D) = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$$

$$\frac{\partial L(D)}{\partial w_j} = -\frac{1}{n} \sum_{i=1}^{n} 2(y_i - \hat{y}_i) \frac{\partial \hat{y}_i}{\partial w_j}$$

$$\frac{\partial L_i}{\partial w_j} = -2(y_i - \hat{y}_i) \frac{\partial \hat{y}_i}{\partial w_j}$$

También podemos escribir únicamente $\frac{\partial L_i}{\partial \hat{y}_i}$, la cual es más útil a la hora de evaluar derivadas con el algoritmo de propagación de regreso:

$$\frac{\partial L_i}{\partial \hat{y}_i} = -2(y_i - \hat{y}_i)$$

Derivadas de las funciones de costo

- Error absoluto medio

$$L(D) = \sum_{i=1}^{n} |y_i - \hat{y}_i|$$

$$\frac{\partial L(D)}{\partial w_j} = -\frac{1}{n} \sum_{i=1}^{n} \text{sign}(y_i - \hat{y}_i) \frac{\partial \hat{y}_i}{\partial w_j}$$

$$\frac{\partial L_i}{\partial w_j} = -\text{sign}(y_i - \hat{y}_i) \frac{\partial \hat{y}_i}{\partial w_j}$$

$$\frac{\partial L_i}{\partial \hat{y}_i} = -\text{sign}(y_i - \hat{y}_i)$$

para $y_i - \hat{y}_i \neq 0$

Derivadas de las funciones de costo

- Huber

$$L(D) = \sum_{i=1}^{n} H(y - \hat{y}_i)$$

donde

$$H(x) = \begin{cases} \frac{1}{2}x^2 & \text{si } |x| < \delta \\ \delta\left(|x| - \frac{1}{2}\delta\right) & \text{en otros casos} \end{cases}$$

$$\frac{\partial L_i}{\partial \hat{y}_i} = \begin{cases} (y_i - \hat{y}_i) & \text{si } |y - \hat{y}_i| < \delta \\ -\delta & \text{en otros casos} \end{cases}$$

Derivadas de las funciones de costo

- Pérdida de bisagra (hinge-loss)

$$L(D) = \sum_{i=1}^{n} \max\{1 - y_i \hat{y}_i, 0\}$$

donde y_i sólo puede ser -1 o 1, y $\hat{y}_i$ es un número continuo.

$$\frac{\partial L_i}{\partial \hat{y}_i} = \begin{cases} -y_i & \text{si } 1 - y_i \hat{y}_i > 0 \\ 0 & \text{en otros casos} \end{cases}$$

Derivadas de las funciones de costo

- Entropía cruzada binaria

$$L(D) = \sum_{i=1}^{n} -(y_i \ln(p_i) + (1 - y_i) \ln(1 - p_i))$$

donde y_i sólo puede ser 0 o 1, y p_i es la salida de la red (probabilidad de la clase 1).

$$\frac{\partial L_i}{\partial p_i} = -\left(\frac{y_i - p_i}{p_i(1 - p_i)}\right)$$

Derivadas de las funciones de costo

- Entropía cruzada categórica (m clases)

$$L(D) = \sum_{i=1}^{n} \left(- \sum_{k=1}^{m} y_{ik} \ln p_{ik} \right)$$

Donde y_{ik} es 1 cuando k es la clase correcta y 0 en otros casos, y p_{ik} es la probabilidad para la clase k.

$$\frac{\partial L_i}{\partial p_{ik}} = - \sum_{k=1}^{m} \frac{y_{ik}}{p_{ik}}$$

Es importante recordar que:

$$\frac{\partial L_i}{\partial w_j} = \frac{\partial L_i}{\partial \hat{y}_i} \frac{\partial \hat{y}_i}{\partial w_j}$$

$$\frac{\partial L_i}{\partial w_j} = \frac{\partial L_i}{\partial p_i} \frac{\partial p_i}{\partial w_j}$$

$$\frac{\partial L_i}{\partial w_j} = \frac{\partial L_i}{\partial p_{ik}} \frac{\partial p_{ik}}{\partial w_j}$$

¿Qué hacemos en el caso de modelos con múltiples neuronas y múltiples capas para encontrar el gradiente de la función de costo respecto a los pesos?

Propagación de regreso

Propagación de regreso

- El problema para redes con más de una neurona está en que la salida de la red es una función composición que involucra a los pesos de la red de capas tanto externas como profundas.

- Por ello, es necesario un algoritmo que obtenga el gradiente de $L(D)$ de manera eficiente aprovechando la estructura misma de la red.

Backprogation – Propagación de regreso

Propagación de regreso

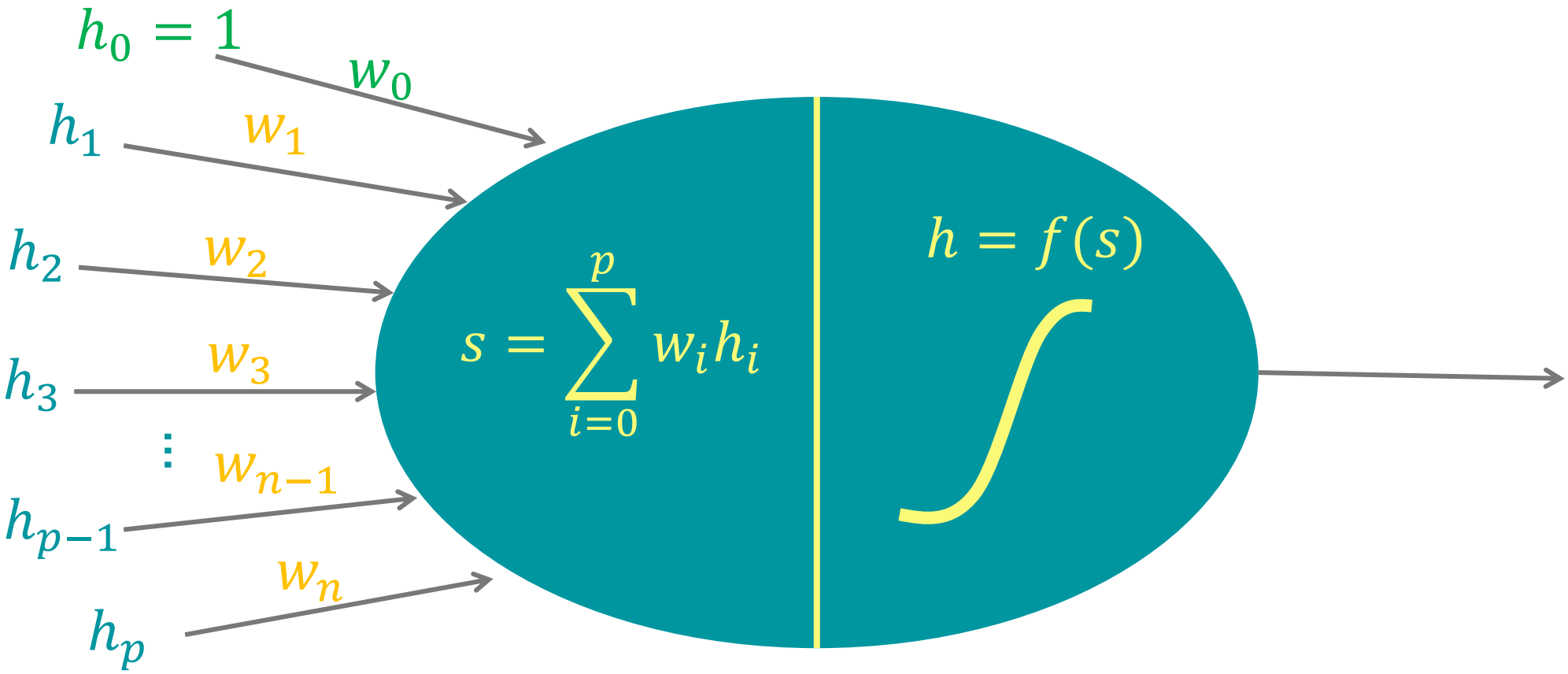
- Este algoritmo es una aplicación directa de la programación dinámica.

- Consiste de dos fases:

  - Fase de ida (fordward phase): Se alimenta las entradas de la red con una observación, lo que ocasiona una serie de cálculos en cascada a través de las capas de la red utilizando el conjunto de pesos actual. El resultado es una predicción para los valores de entrada, la cual se compara con el valor verdadero de acuerdo a la función de costo seleccionada.

  - Fase de regreso (backward phase): Se calcula el gradiente con respecto a los diferentes pesos de la red utilizando la regla de cadena del cálculo diferencial. Con estos gradientes se actualizan los pesos de la red.

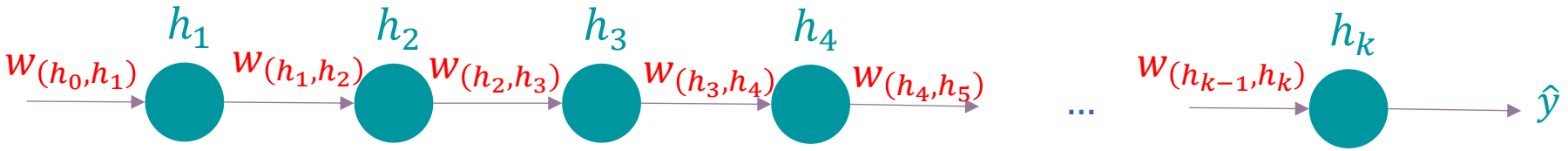
Derivadas en una neurona



$$\frac{\partial h}{\partial w_i} = \frac{df}{ds}\frac{\partial s}{\partial w_i} = h_i \frac{df}{ds}$$

$$\frac{\partial h}{\partial h_i} = \frac{df}{ds}\frac{\partial s}{\partial h_i} = w_i \frac{df}{ds}$$

Derivadas en un grafo dirigido



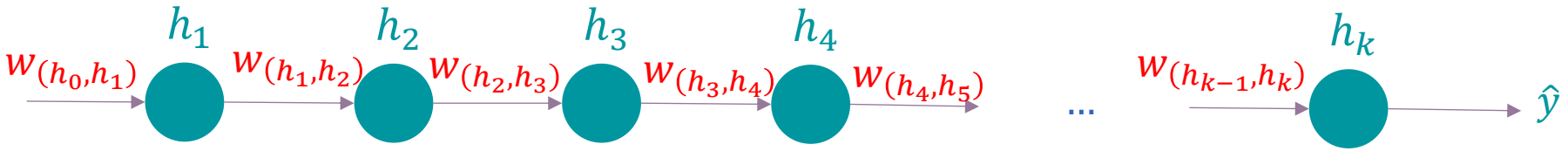
- Considere una red de $k-1$ capas h_1, h_2, h_3, …, h_k, seguidas de una salida $\hat{y}$, consideremos que únicamente tenemos un camino de h_1 a $\hat{y}$.

- El peso $w_{(h_r,h_{r+1})}$ conecta a la capa h_r con la capa h_{r+1}.

- En esta configuración:

$$\frac{\partial L}{\partial w_{(h_{k-1},h_k)}} = \frac{\partial L}{\partial h_k}\frac{\partial h_k}{\partial w_{(h_{k-1},h_k)}}$$

Derivadas en un grafo dirigido



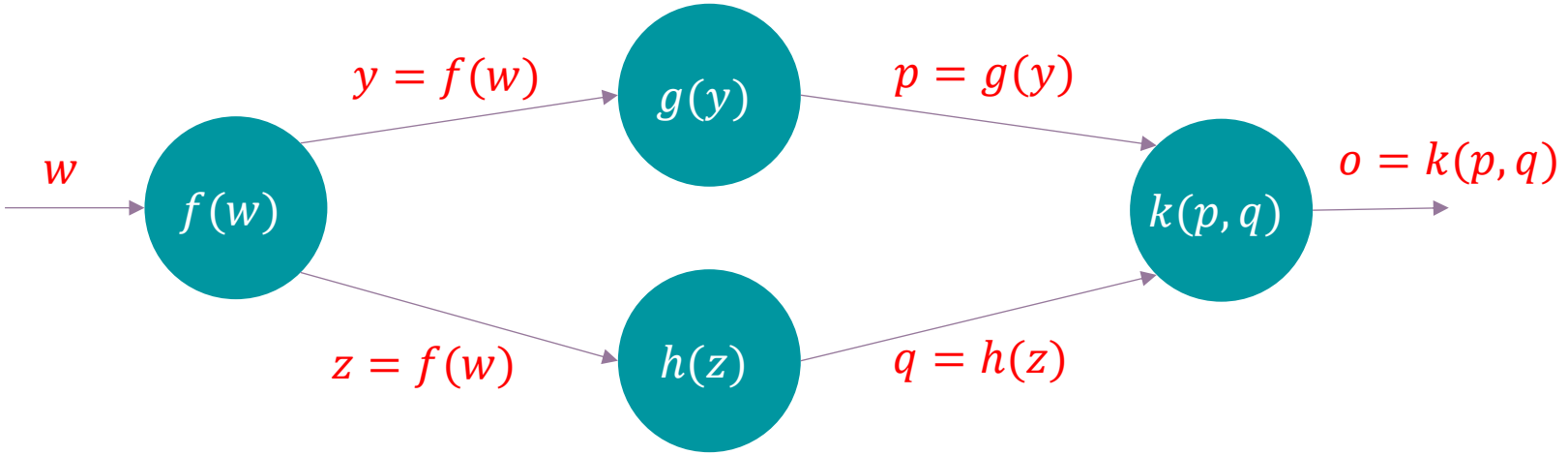
$$\frac{\partial L}{\partial w_{(h_{k-2},h_{k-1})}} = \frac{\partial L}{\partial h_k}\frac{\partial h_k}{\partial h_{k-1}}\frac{\partial h_{k-1}}{\partial w_{(h_{k-2},h_{k-1})}}$$

$$\frac{\partial L}{\partial w_{(h_{k-3},h_{k-2})}} = \frac{\partial L}{\partial h_k}\frac{\partial h_k}{\partial h_{k-1}}\frac{\partial h_{k-1}}{\partial h_{k-2}}\frac{\partial h_{k-2}}{\partial w_{(h_{k-3},h_{k-2})}}$$

$$\frac{\partial L}{\partial w_{(h_{r-1},h_r)}} = \frac{\partial L}{\partial h_k}\left[\prod_{i=r}^{k-1}\frac{\partial h_{i+1}}{\partial h_i}\right]\frac{\partial h_r}{\partial w_{(h_{r-1},h_r)}} \qquad \forall r \in \{1,2,\dots,k\}$$

- Esta regla asume que hay un solo camino de h_1 a o. Sin embargo, en la realidad hay un número exponencial de caminos en una red neuronal. Para estos casos, necesitamos ver cómo funciona la regla de la cadena para funciones multivariadas.

Derivadas en un grafo dirigido



$$\frac{\partial o}{\partial w} = \frac{\partial o}{\partial p}\frac{\partial p}{\partial w} + \frac{\partial o}{\partial q}\frac{\partial q}{\partial w}$$ [regla de la cadena multivariada]

$$\frac{\partial o}{\partial w} = \underbrace{\frac{\partial o}{\partial p}\frac{\partial p}{\partial y}\frac{\partial y}{\partial w}}_{\text{Primer camino}} + \underbrace{\frac{\partial o}{\partial q}\frac{\partial q}{\partial z}\frac{\partial z}{\partial w}}_{\text{Segundo camino}}$$ [regla de la univariada]

Derivadas en un grafo dirigido

- Juntando ambas ideas, obtenemos una expresión para calcular $\frac{\partial L}{\partial w_{(h_{r-1},h_r)}}$ para cuando se tienen $\mathcal{P}$ caminos de h_r a h_k:

$$\frac{\partial L}{\partial w_{(h_{r-1},h_r)}} = \frac{\partial L}{\partial h_k}\left[\sum_{[h_r,h_{r+1},\dots h_k,\hat{y}]\in\mathcal{P}} \prod_{i=r}^{k-1} \frac{\partial h_{i+1}}{\partial h_i}\right]\frac{\partial h_r}{\partial w_{(h_{r-1},h_r)}} \quad \forall r \in \{1,2,\dots,k\}$$

donde $[h_r, h_{r+1}, \dots h_k, \hat{y}] \in \mathcal{P}$ indica el conjunto de funciones de activación de algún camino de h_r a $\hat{y}$.

Propagación de regreso

- Para facilitar el cálculo de cada $\frac{\partial L}{\partial w_{(h_{r-1},h_r)}}$, se utiliza el algoritmo de propagación de regreso, el cual calcula $\frac{\partial L}{\partial h_r}$ comenzando desde la salida y yendo hacia atrás en la red.

- Es decir, el algoritmo calcula:

$$\Delta(h_r) = \frac{\partial L}{\partial h_r} = \frac{\partial L}{\partial h_k}\left[\sum_{[h_r,h_{r+1},\dots h_k,\hat{y}]\in\mathcal{P}} \prod_{i=r}^{k-1} \frac{\partial h_{i+1}}{\partial h_i}\right]$$

comenzando en la última capa y luego propagándose hacia atrás.

Propagación de regreso

- Si la red no tiene ciclos, para poder calcular $\Delta(h_r)$, se comienza calculando $\Delta(h_k)$, el cual es el nodo más cercano a $\hat{y}$, y luego recursivamente se calculan los valores de $\Delta(h_{k-1})$, $\Delta(h_{k-2})$, y así sucesivamente hasta llegar a $\Delta(h_r)$.

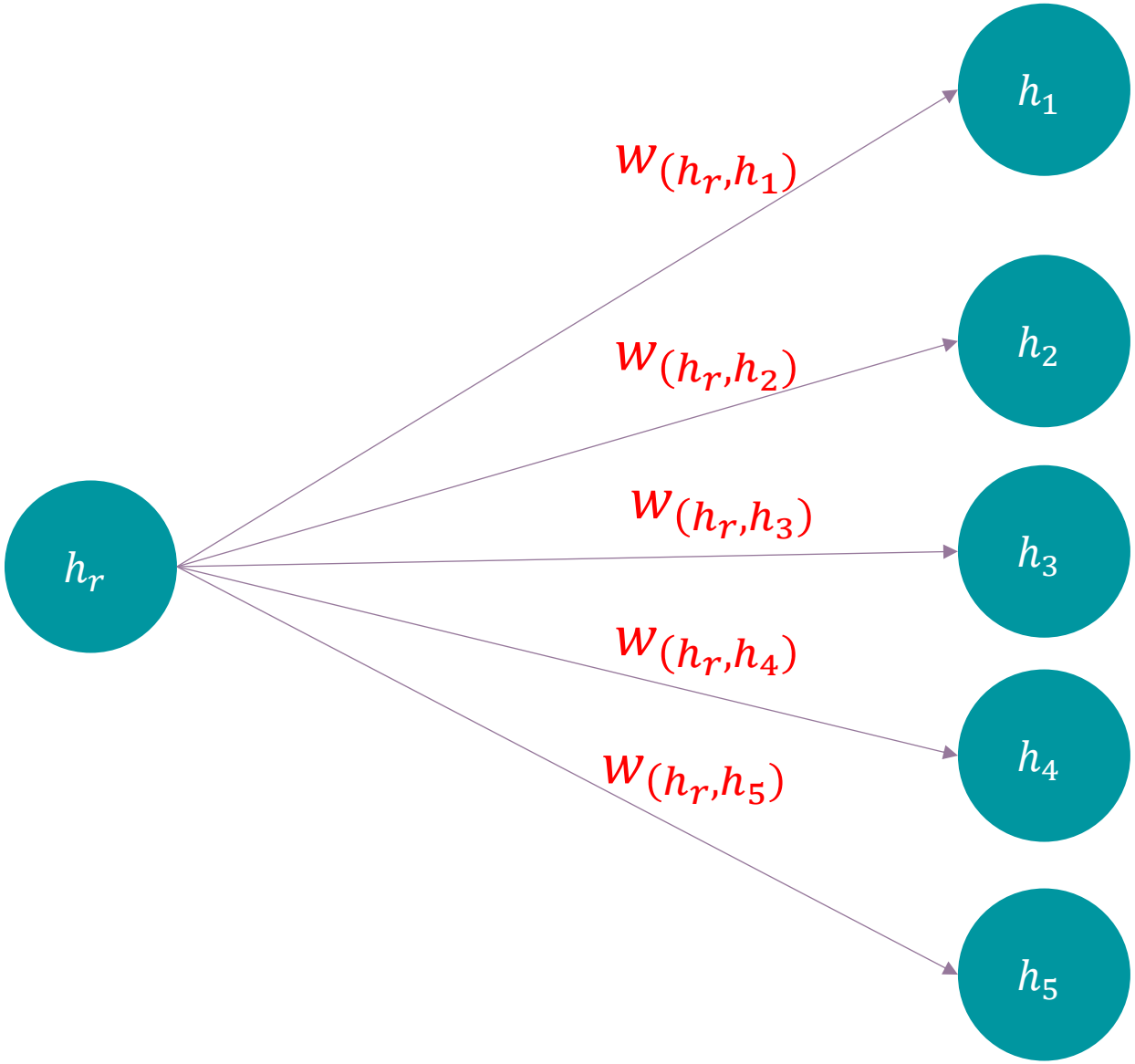
- Se comienza con el valor de $\Delta(h_k) = \frac{\partial L}{\partial h_k}$.

- La recursión puede de nuevo ser derivada de la regla de la cadena multivariada:

$$\Delta(h_r) = \frac{\partial L}{\partial h_r} = \sum_{h:h_r \Rightarrow h} \frac{\partial L}{\partial h} \frac{\partial h}{\partial h_r} = \sum_{h:h_r \Rightarrow h} \Delta(h) \frac{\partial h}{\partial h_r}$$

donde $h: h_r \Rightarrow h$ indica los nodos que parten de h_r a la siguiente capa.

Propagación de regreso



Propagación de regreso

1. Propagar hacia delante con una observación (X, y) para evaluar todas las salidas de las unidades computacionales, la salida $\hat{y}$ y la función de costo L.

2. Inicializar $\Delta\,(h_k) = \frac{\partial L}{\partial h_k}$.

3. Calcular $\Delta\,(h_r)$ en la dirección de regreso utilizando la recursión:

$$\Delta\,(h_r) = \sum_{h:h_r \Rightarrow h} \Delta(h) \frac{\partial h}{\partial h_r}$$

4. Para cada cálculo de $\Delta\,(h_r)$, también encontrar $\frac{\partial L}{\partial w_{(h_{r-1},h_r)}}$:

$$\frac{\partial L}{\partial w_{(h_{r-1},h_r)}} = \Delta(h_r) \frac{\partial h_r}{\partial w_{(h_{r-1},h_r)}}$$

5. Utilizar las derivadas para actualizar los pesos de la red.

Bibliografía

- Aggarwal , C. C. (2023). *Neural networks and deep learning* (2da ed.). Springer.
  - Capítulo 2