Tecnológico de Monterrey

Aprendizaje automático

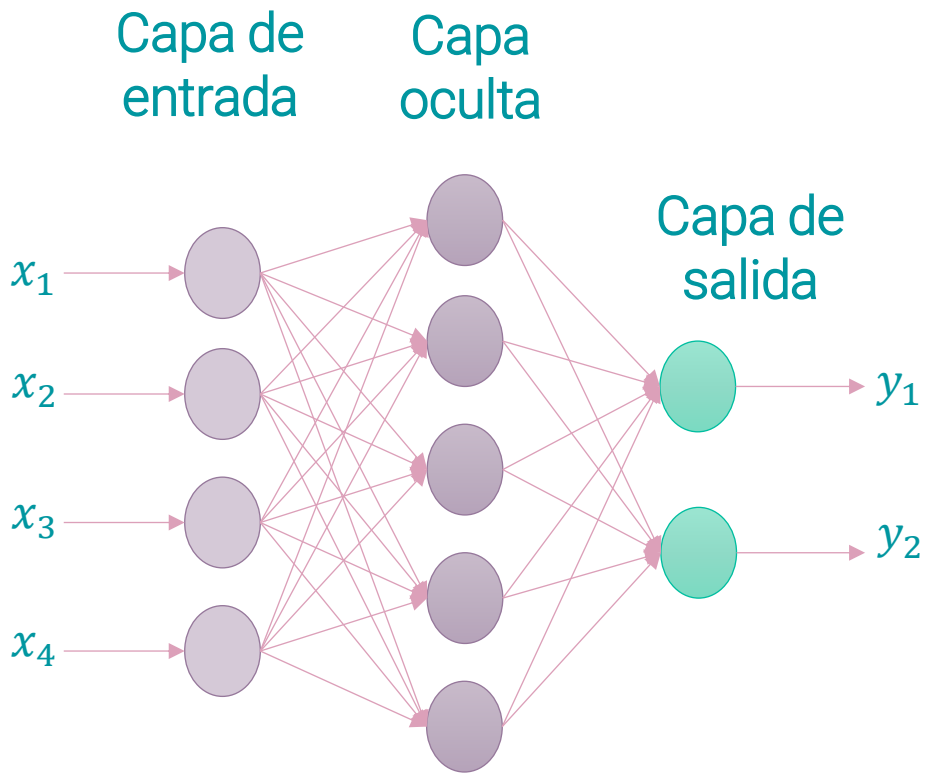
El modelo perceptrón multicapa

Dr. Omar Mendoza Montoya

04/09/2024

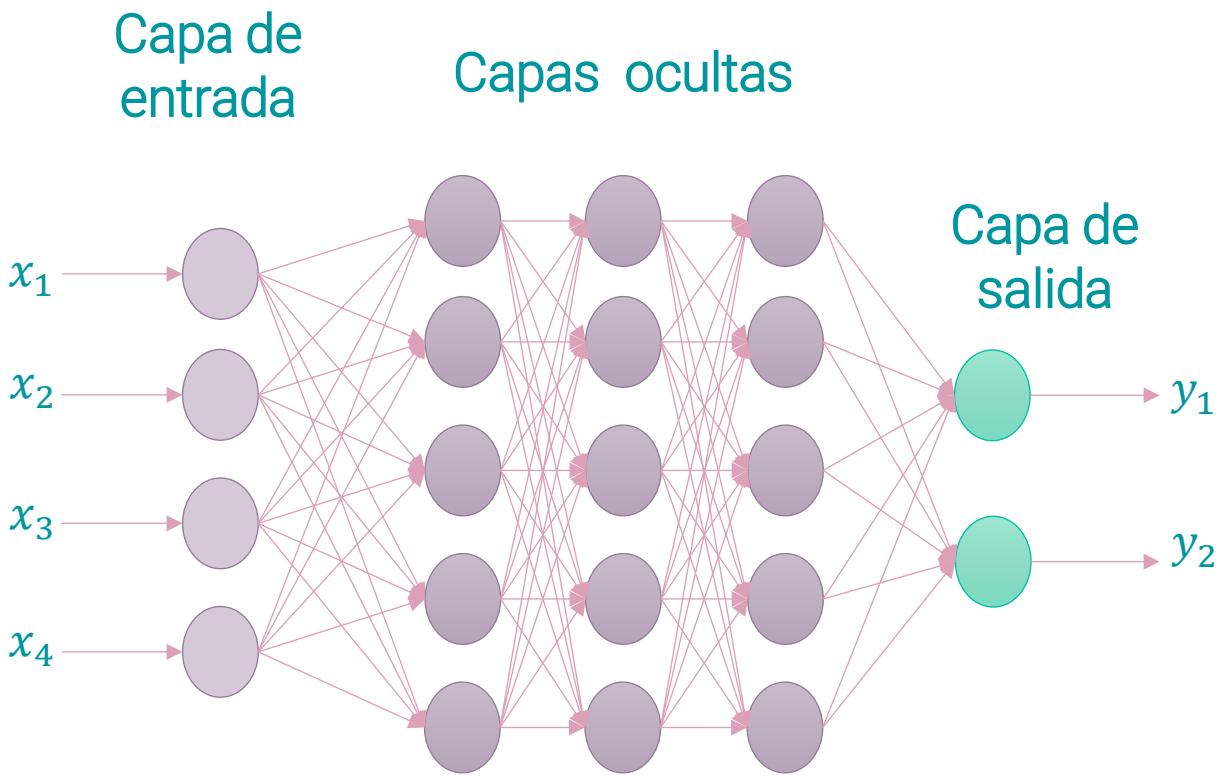
Single-layer perceptron (SLP)

- Una red perceptrón de una capa es la primera versión de red neuronal práctica (versión vainilla).
- SLPs sólo tienen una capa entre las entradas y las salidas.
- El método de optimización para encontrar los parámetros se llama back-propagation, y básicamente sigue el mismo esquema de descenso de gradiente visto con el modelo perceptrón de una neurona.

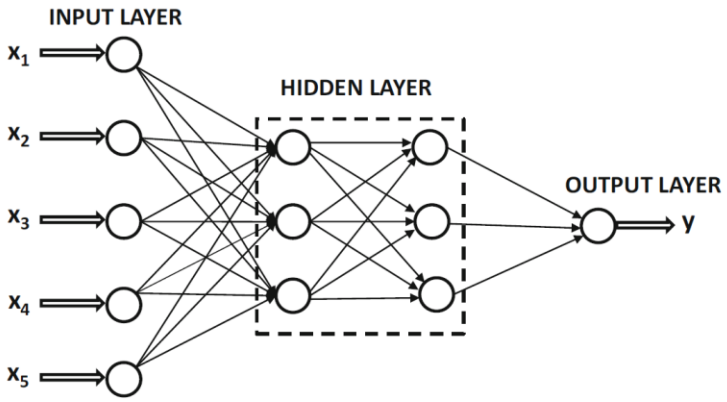


Multilayer perceptron (MLP)

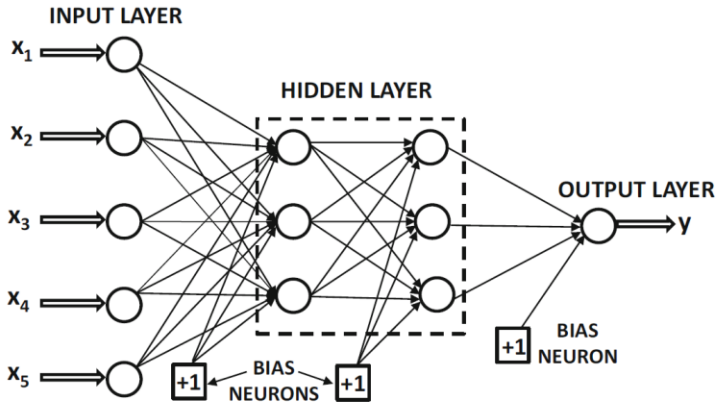
- Una red perceptrón de múltiples capas tiene más de una capa oculta de procesamiento.
- Cada capa alimenta a la capa sucesiva de forma directa sin retroalimentación (feed-forward network).
- La arquitectura por defecto asume que todas las neuronas de una capa se interconectan con las neuronas de la capa sucesiva.



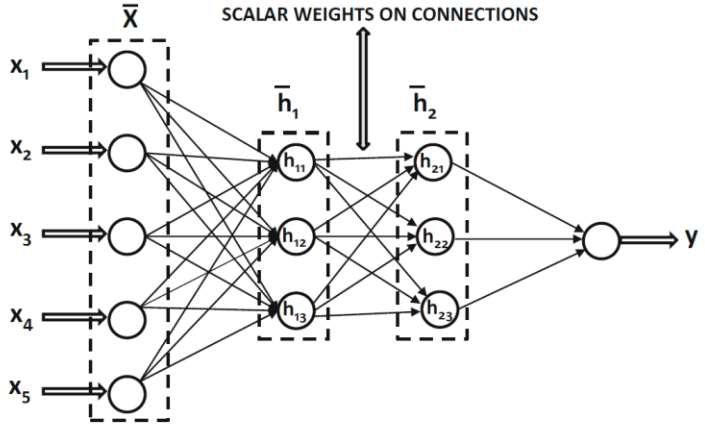
Representación gráfica de una red



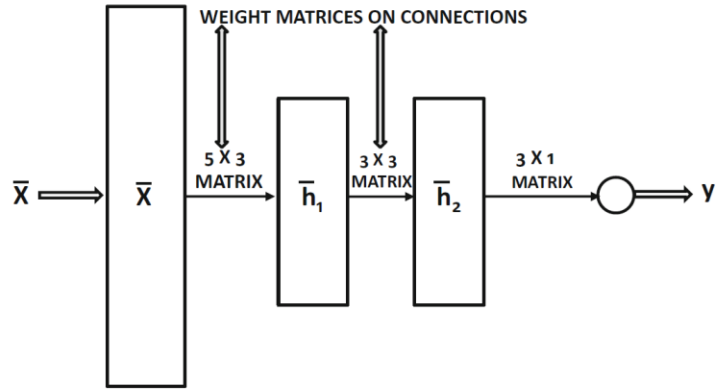
(a) No bias neurons



(b) With bias neurons



(c) Scalar notation and architecture



(d) Vector notation and architecture

Aggarwal, 2023

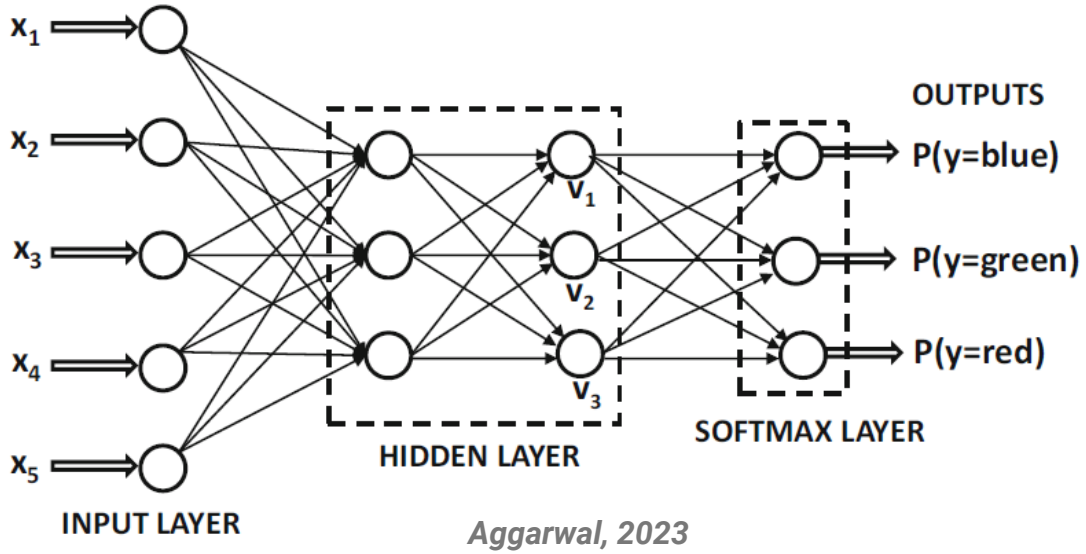
Las aplicaciones típicas del modelo perceptrón multicapa son clasificación de dos clases, clasificación multiclase, regresión de una variable, regresión multivariada.

Neuronas en un modelo perceptron multicapa

- Neuronas de entrada.
  - Representan a las variables predictoras. No se suelen realizar operaciones sobre los datos.
- Neuronas de las capas ocultas.
  - Llevan a cabo la suma ponderada de sus entradas y aplican una función de activación (neurona estándar).
  - Se suele utilizar la misma función de activación en todas las neuronas de la capa oculta.
- Neuronas de salida.
  - Preparan el resultado final.
  - Su número depende del tipo de problema (clasificación de 2 clases, multiclase, regresión de una variable, regresión multivariables).
  - Se comportan como la neurona estándar, excepto e el caso de clasificación multiclase.

Neuronas de salida

- Para problemas de regresión, cada nodo de salida puede representar una variable dependiente y la función de activación puede ser lineal o cualquier otra transformación.
- Para un clasificador binario, sólo es necesario una neurona con una función de umbral o la sigmoide.
- Para un clasificador de k clases, una elección común es k neuronas combinadas con una función de activación especial (softmax).



Aggarwal, 2023

$$y_i = \frac{e^{v_i}}{\sum_{j=1}^{k} e^{v_j}}$$

Softmax

- Si $s_1, s_2, s_3, \ldots, s_k$ son las sumas ponderadas de las neuronas de la capa de salida, la función Softmax se define por:

$$p_i = \frac{e^{s_i}}{\sum_{j=1}^{k} e^{s_j}}$$

donde p_i es la salida de la neurona i de la capa de salida.

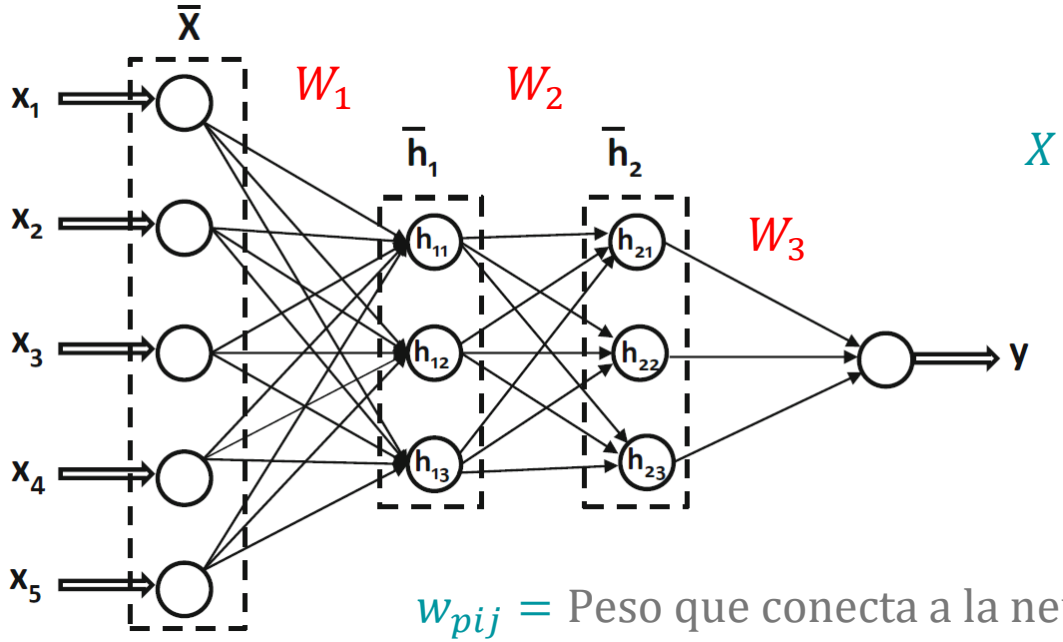
- A su vez, las derivadas de la capa de salida están dadas por:

$$p_i' = p_i(1 - p_i)$$

¿Cómo se implementa y evalúa una red neuronal perceptrón multicapa?

La implementación de una red neuronal de múltiples capas suele aprovechar la representación matricial, en la cual el resultado final se obtiene como productos de matrices y operaciones vectoriales.

Representación matricial



$$X = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_{n_0} \end{bmatrix} \qquad h_p = \begin{bmatrix} h_{p1} \\ h_{p2} \\ \vdots \\ h_{n_p} \end{bmatrix} \qquad W_p = \begin{bmatrix} w_{p11} & w_{p12} & \cdots & w_{p1n_{p-1}} \\ w_{p21} & w_{p22} & \cdots & w_{p2n_{p-1}} \\ \vdots & \vdots & \ddots & \vdots \\ w_{pn_p1} & w_{pn_p2} & \cdots & w_{pn_pn_{p-1}} \end{bmatrix}$$

$n_0 =$ Número de neuronas de entrada

$n_p =$ Número de neuronas de la capa p

$x_i =$ Entrada i

$h_{pi} =$ Salida de la neurona i de la capa p

$w_{pij} =$ Peso que conecta a la neurona i de la capa p con la neurona j de la capa $p-1$

$h_1 = G_1(W_1X)$ (Salida de la capa oculta 1)

$h_p = G_p(W_ph_{p-1})$ (Salida de la capa oculta p)

$$G_p(S) = \begin{bmatrix} g_p(s_1) \\ g_p(s_2) \\ \vdots \\ g_p(s_{n_p}) \end{bmatrix}$$ (Función de activación de la capa p)

$$y = G_m\left(W_mG_{m-1}\left(W_{m-1}G_{m-2}(\dots W_2G_1(W_1X))\right)\right)$$

$m-1$ capaz ocultas

Composición de composición de funciones

¿Cómo se selecciona el número de capas, neuronas y funciones de activación?

Selección de la arquitectura de la red

- Para la capa oculta, se suele utilizar funciones de activación no lineales.
- La función ReLu es una opción viable como función de activación no lineal, ya que es fácil de evaluar y optimizar en una arquitectura de GPUs. Además, esta función no satura como la tangente hiperbólica o la sigmoide.
- Junto con las funciones de activación no lineales, incrementar el número de neuronas y capas permite modelar datos con estructuras complejas.
- El tamaño de las capas ocultas suele ser mayor al número de neuronas de entrada y de salida.

Selección de la arquitectura de la red

- Para problemas de clasificación de dos clases, se utiliza una sola neurona de salida con función de activación signmoide. Dicha neurona devuelve un valor entre 0 y 1, de tal forma que valores menores a 0.5 representan una clase y por encima de 0.5 representan a la otra clase.

- Para problemas de clasificación de más de dos clases, se tienen tantas neuronas de salida como clases con función de activación softmax. Cada neurona devuelve la probabilidad que corresponde a cada clase, por lo que la clase es aquella con mayor probabilidad.

- Para problemas de regresión, se tienen tantas neuronas como variables depedientes con función de activación lineal.

¿Qué más es necesario definir para construir un modelo perceptrón multicapa?

Al igual que cualquier otro problema de aprendizaje supervisado, la tarea de encontrar los parámetros de una red neuronal es un problema de optimización en el cual, dado un conjunto de datos, se minimiza una función que representa el error de la red respecto al conjunto de entrenamiento.

A esta función se le conoce como función de pérdida.

Funciones de pérdida

- El algoritmo de aprendizaje para encontrar el conjunto de pesos W suele involucrar un proceso de minimización de una métrica que nos indica de manera general qué tan mala es la red neuronal para predecir o generar un resultado satisfactorio.

- En aprendizaje supervisado, las métricas más utilizadas son funciones de error o pérdida, las cuales comparan a la respuesta de la red neuronal con el resultado deseado.

- Para un conjunto de datos $D = \{(X_1, y_1), (X_2, y_2), \dots, (X_n, y_n)\}$, donde (X_i, y_i) es la observación i-ésima del conjunto con vector de características X_i y valor de respuesta y_i, el problema es encontrar W^* que minimice a la función de pérdida $L(D)$.

$$W^* = \min_W L(D)$$

El tipo de función de pérdida en aprendizaje supervisado se suele seleccionar de acuerdo con el tipo de problema que se quiere resolver (clasificación o regresión).

En redes neuronales se prefiere utilizar funciones continuas aun en problemas de clasificación.

Funciones de pérdida para regresión

- Error cuadrático medio

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

donde n es el número de observaciones en el conjunto de datos X, y_i es el valor de la variable de respuesta para el dato i, y $\hat{y}_i$ es el valor predicho por el modelo para el mismo dato.

- Error absoluto medio

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

Funciones de pérdida para regresión

- Huber

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

donde

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

Funciones de pérdida para clasificación

- Pérdida de bisagra (hinge-loss) (2 clases)

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

- Entropía cruzada (2 clases)

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

donde p_i es la salida de la red (probabilidad de clase).

Funciones de pérdida para clasificación

- Entropía cruzada (m clases)

$$L(D) = \frac{1}{n}\sum_{i=1}^{n}\sum_{c=1}^{m} y_{ic} \ln p_{ic}$$

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

Bibliografía

- Aggarwal , C. C. (2023). *Neural networks and deep learning* (2da ed.). Springer.
  - Capítulo 1