“Redes Neuronales Artificiales (ANN)”

APRENDIZAJE AUTOMÁTICO

PROGRAMA DE CIENCIA DE DATOS

Profesor: MSc. Felipe Meza

TEC | Tecnológico de Costa Rica

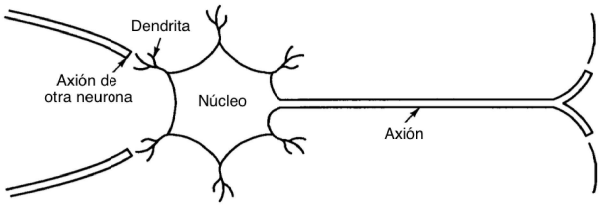
Redes Neuronales Artificiales (ANN)

- Introducción
- El modelo neuronal biológico
- El modelo neuronal artificial - Funciones de activación
- Capas, nodos y pesos (feedforward)
- Determinación de Error
- Back-Propagation - Gradiente descendiente
- Tasa de aprendizaje
- La función `SOFTMAX` y regularización `DROPOUT`
- Optimización
- Ejemplos

Introducción

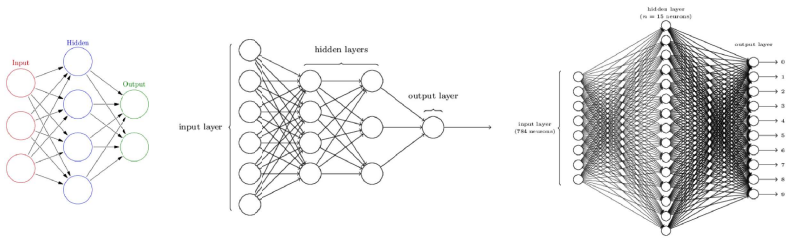
- **1943** McCulloch/Pitts proponen un modelo neuronal MP.
- **1958** Rosenblatt introduce el concepto de "perceptrón".
- **1969** Minsky/Papert lanzan un libro en el que se demuestran las limitaciones del "perceptrón".
- **1986** Con el concepto de Back-Propagation y MLP, se vuelve a popularizar el tema de ANN's.
- **1998** Yann LeCun, CNN's.
- **2006** Deep Learning, como concepto.
- **2014** Ian Goodfellow, GAN's.

El modelo neuronal biológico



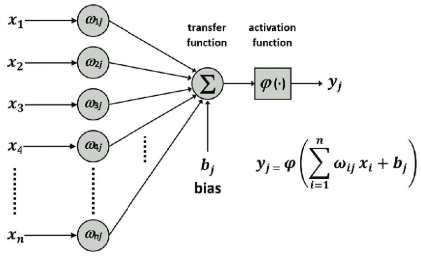
- 86-100 mil millones de neuronas.
- Señales electro-químicas.
- Dendritas reciben información de otra neuronas.
- Axones envían información a otras neuronas.
- El aprendizaje ocurre por activaciones repetidas de ciertas conexiones sobre otras.

El modelo neuronal artificial



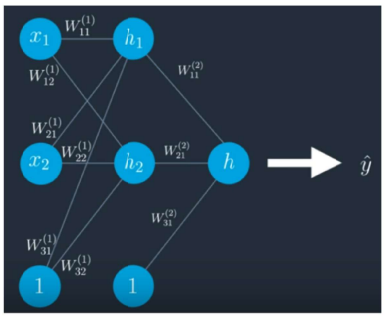
- Arquitectura **inspirada** en el modelo neuronal biológico.
- Compuesta por un capa de **entrada**, una o varias capas **ocultas** y una capa de **salida.**
- Proveer una **salida** deseada a partir de una **entrada** específica.
- Puede llevar a cabo tareas de **clasificación** o **regresión.**

Perceptrón - Funciones de activación



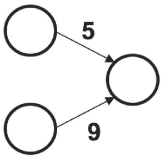
Activation function	Equation	Example	1D Graph
Unit step (Heaviside)	$\phi(z) = \begin{cases} 0, & z < 0, \\ 0.5, & z = 0, \\ 1, & z > 0, \end{cases}$	Perceptron variant	
Sign (Signum)	$\phi(z) = \begin{cases} -1, & z < 0, \\ 0, & z = 0, \\ 1, & z > 0, \end{cases}$	Perceptron variant	
Linear	$\phi(z) = z$	Adaline, linear regression	
Piece-wise linear	$\phi(z) = \begin{cases} 1, & z \geq \frac{1}{2}, \\ z + \frac{1}{2}, & -\frac{1}{2} < z < \frac{1}{2}, \\ 0, & z \leq -\frac{1}{2}, \end{cases}$	Support vector machine	
Logistic (sigmoid)	$\phi(z) = \frac{1}{1 + e^{-z}}$	Logistic regression, Multi-layer NN	
Hyperbolic tangent	$\phi(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$	Multi-layer Neural Networks	
Rectifier, ReLU (Rectified Linear Unit)	$\phi(z) = max(0, z)$	Multi-layer Neural Networks	
Rectifier, softplus	$\phi(z) = \ln(1 + e^z)$	Multi-layer Neural Networks	

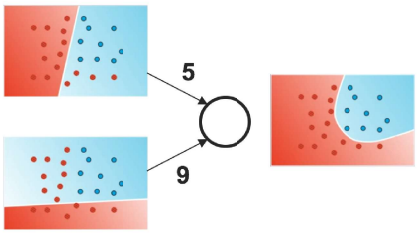
Capas, nodos y pesos (feedforward)



$$h_1 = W_{11}^{(1)}x_1 + W_{12}^{(1)}x_2 + W_{13}^{(1)}$$
$$h_2 = W_{21}^{(1)}x_1 + W_{22}^{(1)}x_2 + W_{23}^{(1)}$$
$$h = W_{11}^{(2)}\sigma(h_1) + W_{21}^{(2)}\sigma(h_2) + W_{31}^{(2)}$$
$$\hat{y} = \sigma(h)$$

Capas, nodos y pesos (visualización)





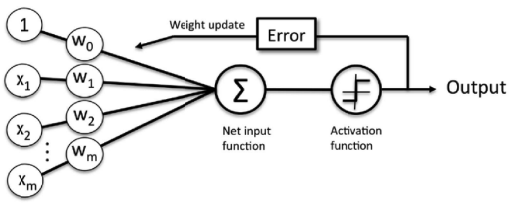
Determinación de Error

$$E_{total} = \sum \frac{1}{2}(target - output)^2$$

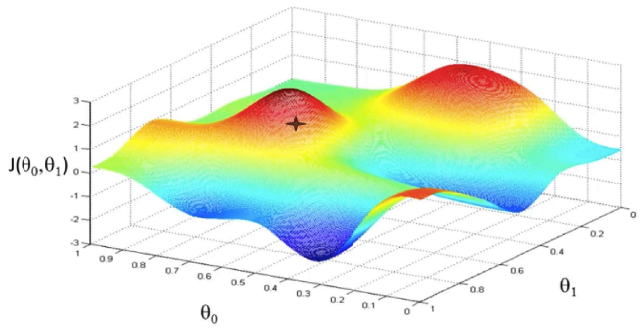
$$E = -\frac{1}{m}\sum_{i=1}^{m} \left(y_i \ln(\hat{y}_i) + (1-y_i) \ln(1-\hat{y}_i) \right)$$

Back-Propagation

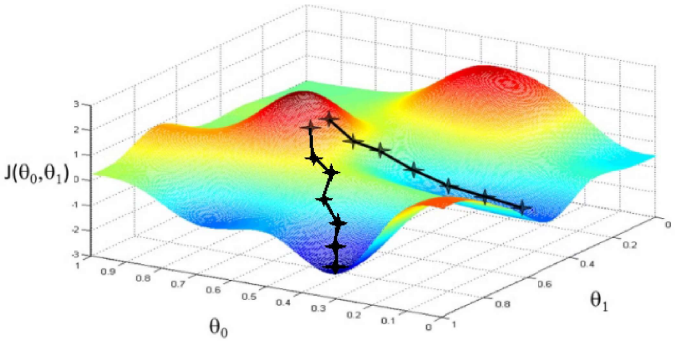
- Se hace feedforward.
- Comparación de la salida obtenida con la deseada.
- Cálculo del error.
- Se hace feedforward al revés (backpropagation) del error.
- Se actualizan los pesos para un mejor modelo.
- Se continua hasta tener el mejor modelo.



Gradiente descendiente

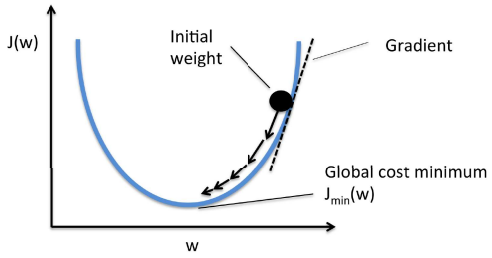


Gradiente descendiente



Gradiente descendiente

- Minimizar el error en la salida al variar el valor de los pesos (w) en el gradiente descendiente.

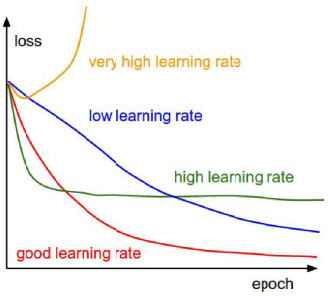


$$\nabla E = \left(\frac{\partial}{\partial w_1} E, \cdots, \frac{\partial}{\partial w_n} E, \frac{\partial}{\partial b} E \right)$$

Tasa de aprendizaje

- Variable denominada como α que se usa de la mano el gradiente descendiente para dar pasos más pequeños.

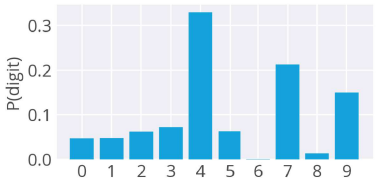
$$w_i(final) = w_i(inicial) + \alpha \times \nabla E_i$$



La función SOFTMAX

- La salida queda distribuida entre 0 y 1, como una distribución de probabilidad.
- Se normaliza respecto a la totalidad de las salidas.
- Para clasificación múltiple se usa en las capas de salida.

$$\begin{bmatrix} 1.2 \\ 0.9 \\ 0.4 \end{bmatrix} \rightarrow \boxed{\text{Softmax}} \rightarrow \begin{bmatrix} 0.46 \\ 0.34 \\ 0.20 \end{bmatrix}$$



One Hot Encode

- Permite convertir datos categóricos en numéricos (vectores binarios).

Sample	Category	Numerical
1	Human	1
2	Human	1
3	Penguin	2
4	Octopus	3
5	Alien	4
6	Octopus	3
7	Alien	4

Datos Categóricos

Sample	Human	Penguin	Octopus	Alien
1	1	0	0	0
2	1	0	0	0
3	0	1	0	0
4	0	0	1	0
5	0	0	0	1
6	0	0	1	0
7	0	0	0	1

Vectores Binarios

$$[0\ \ 0\ \ 0\ \ \boxed{1}\ \ 0] \times \begin{bmatrix} 17 & 24 & 1 \\ 23 & 5 & 7 \\ 4 & 6 & 13 \\ \boxed{10} & \boxed{12} & \boxed{19} \\ 11 & 18 & 25 \end{bmatrix} = [10\ \ 12\ \ 19]$$

One Hot Encode

Optimización

- Un algoritmo de optimización tiene como objeto minimizar la *función de error*.
- Hay que recordar que la *función de error* opera sobre los **parámetros** del modelo de aprendizaje (no confundir con los **hiper-parámetros**).
- Hay dos tipos de algoritmos:
  - Primer Orden: Usa la derivada de primer orden (gradiente). Muy utilizada, **gradiente descendiente** en el más popular.
  - Segundo Orden: Usa la derivada de segundo orden (hessian). Poco utilizada por costo computacional.

Optimización

- Tipos de **gradiente descendiente (GD)**:
  - Stochastic gradient descent
  - Mini Batch Gradient Descent
- Métodos para optimizar el **GD**:
  - **Momentum**: Suaviza las oscilaciones en direcciones no relevantes.
  - **Nesterov accelerated gradient**: Hace más lento el movimiento cuando llega a un mínimo.
  - **Adagrad**: El *learning rate* se ajusta de acuerdo a todos los parámetros.
  - **AdaDelta**: Mejora al Adagrad.
  - **Adam**: El *learning rate* se ajusta de acuerdo a cada parámetro.

Epochs, Batch, Iterations

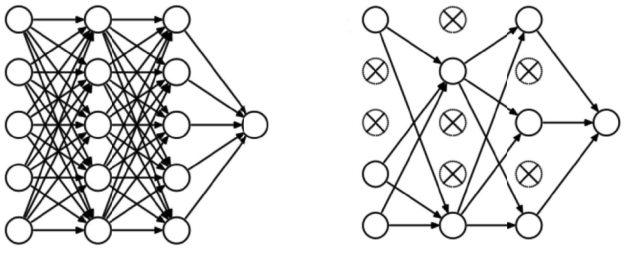
- **Epoch**: Conjunto entero de datos que se hace pasar por una ANN hacia adelante y hacia atrás, una sola vez.
- **Batch**: Conjunto de datos para entrenamiento, un Epoch puede estar compuesto por varios batch.
- **Iterations**: Conjunto necesario de batches para completar un epoch.

Podemos dividir el conjunto de datos de 2000 ejemplos en batches de 500, luego se necesitarán 4 iteraciones para completar 1 epoch.

Batch Size = 500 y Iterations = 4 ...para completar 1 epoch.

Regularización - Dropout

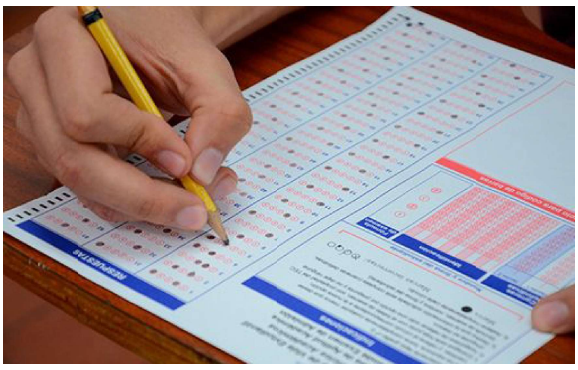
- Neuronas con altos pesos pueden acaparar el entrenamiento.
- Consiste en "apagar" algunas neuronas artificiales para darle oportunidad a otras de ser parte del entrenamiento.



Regularización - Dropout

- Se activa por capa oculta y se puede usar para todas o solo para algunas.
- Se usa en diferentes tipos de ANN.
- La eliminación de neuronas artificiales es aleatoria y se da a partir de una probabilidad.

Caso REGRESIÓN - KERAS



Proyección de ANN's

http://playground.tensorflow.org

Questions?

Felipe Meza - fmezacr@gmail.com