# Algoritmos de forward y backward en ANN

AUTHOR
Nataly Quintanilla_Maria Isabel Chuya

# Algoritmos de forward y backward en ANN

En las redes neuronales artificiales (ANN), los algoritmos de avance (forward) y retroceso (backward) son esenciales para el entrenamiento y el uso de la red. Aquí tienes un resumen de estos algoritmos:

1. Algoritmo de avance (Forward algorithm): El algoritmo de avance, también conocido como propagación hacia adelante (forward propagation), se utiliza para calcular las salidas de la red neuronal a partir de una entrada dada. Este algoritmo se realiza en capas sucesivas, desde la capa de entrada hasta la capa de salida. Cada neurona en una capa toma las salidas de las neuronas en la capa anterior, las combina con los pesos correspondientes y aplica una función de activación para producir su propia salida. Este proceso continúa capa por capa hasta que se obtiene la salida final de la red.

   

2. Algoritmo de retroceso (Backward algorithm): El algoritmo de retroceso, también conocido como retropropagación del error (backpropagation), se utiliza para entrenar la red neuronal ajustando los pesos de las conexiones entre las neuronas. El objetivo del entrenamiento es minimizar una función de costo que mide la diferencia entre las salidas predichas por la red y los valores reales de entrenamiento.

El algoritmo de retroceso se realiza en dos etapas principales: a) Etapa de avance: Durante esta etapa, se utiliza el algoritmo de avance para calcular las salidas de la red neuronal a partir de las entradas de entrenamiento. b) Etapa de retroceso del error: En esta etapa, se calcula el error entre las salidas predichas y los valores reales, y se propaga hacia atrás a través de la red para ajustar los pesos. Esto se logra mediante el cálculo de gradientes y la actualización de los pesos utilizando un algoritmo de

optimización, como el descenso del gradiente.

Durante la etapa de retroceso, se calculan los gradientes de los pesos de la red utilizando la regla de la cadena y se actualizan los pesos en dirección opuesta al gradiente para minimizar la función de costo. Este proceso se repite iterativamente hasta que se alcance un criterio de convergencia deseado.

En resumen, el algoritmo de avance se utiliza para calcular las salidas de la red neuronal a partir de una entrada dada, mientras que el algoritmo de retroceso se utiliza para ajustar los pesos de la red mediante la retropropagación del error y la actualización de los pesos. Estos algoritmos son fundamentales para el entrenamiento y el uso efectivo de las redes neuronales artificiales. Los algoritmos de avance (forward) y retroceso (backward) son fundamentales en el entrenamiento y el uso de redes neuronales artificiales (ANN). Aquí tienes un resumen de estos algoritmos:

1. Algoritmo de avance (Forward algorithm): El algoritmo de avance, también conocido como propagación hacia adelante (forward propagation), es utilizado para calcular la salida de una red neuronal a partir de una entrada dada. En este algoritmo, las entradas se propagan capa por capa a través de la red neuronal, donde cada neurona en una capa toma las salidas de las neuronas en la capa anterior, las combina ponderadamente con los pesos y aplica una función de activación para producir su propia salida. Este proceso se repite hasta llegar a la capa de salida, donde se obtiene la salida final de la red.

2. Algoritmo de retroceso (Backward algorithm): El algoritmo de retroceso, también conocido como retropropagación del error (backpropagation), es utilizado para entrenar una red neuronal ajustando los pesos de las conexiones entre las neuronas. Este algoritmo se basa en la minimización de una función de costo que mide la diferencia entre las salidas predichas por la red y los valores reales de entrenamiento.

El proceso de retroceso se realiza en dos etapas: a) Etapa de avance: Durante esta etapa, se utiliza el algoritmo de avance para calcular las salidas de la red neuronal a partir de las entradas de entrenamiento. b) Etapa de retroceso del error: En esta etapa, se calcula el error entre las

salidas predichas y los valores reales, y se propaga hacia atrás a través de la red para ajustar los pesos. Esto se logra mediante el cálculo de gradientes y la actualización de los pesos utilizando algoritmos de optimización, como el descenso del gradiente.

En resumen, el algoritmo de avance se utiliza para calcular las salidas de una red neuronal a partir de una entrada dada, mientras que el algoritmo de retroceso se utiliza para ajustar los pesos de la red mediante la retropropagación del error. Estos algoritmos son esenciales en el entrenamiento y uso efectivo de las redes neuronales artificiales, ya que permiten que la red aprenda a partir de los datos de entrenamiento y realice predicciones precisas en nuevas entradas. En RStudio, los algoritmos de avance (forward) y retroceso (backward) se utilizan en el contexto de las redes neuronales artificiales (ANN) para entrenar y utilizar la red. Aquí tienes un resumen de estos algoritmos en RStudio:

1. Algoritmo de avance (Forward algorithm): El algoritmo de avance, también conocido como propagación hacia adelante (forward propagation), se utiliza para calcular las salidas de la red neuronal a partir de una entrada dada. En RStudio, puedes implementar este algoritmo utilizando paquetes como `neuralnet`, `keras` o `nnet`. Estos paquetes te permiten construir la arquitectura de la red, definir las funciones de activación y propagar los datos de entrada a través de la red para obtener la salida predicha. Puedes especificar la estructura de la red, los pesos iniciales y las funciones de activación para cada capa de la red.

2. Algoritmo de retroceso (Backward algorithm): El algoritmo de retroceso, también conocido como retropropagación del error (backpropagation), se utiliza para entrenar la red neuronal ajustando los pesos de las conexiones entre las neuronas. En RStudio, puedes implementar este algoritmo utilizando paquetes como `neuralnet`, `keras` o `nnet`. Estos paquetes proporcionan funciones para definir la función de costo, configurar el algoritmo de optimización y realizar la retropropagación del error para actualizar los pesos de la red. Puedes especificar el criterio de convergencia, el algoritmo de optimización (como el descenso del gradiente estocástico) y otros parámetros relacionados con el entrenamiento de la red.

En resumen, en RStudio, puedes implementar los algoritmos de avance y retroceso utilizando paquetes específicos para redes neuronales, como `neuralnet`, `keras` o `nnet`. Estos paquetes proporcionan funciones y estructuras de datos que te permiten construir, entrenar y utilizar redes

neuronales artificiales. Puedes especificar la arquitectura de la red, las funciones de activación, el algoritmo de optimización y otros parámetros relevantes para el entrenamiento y uso de la red en RStudio.