NEURONAS ARTIFICIALES

IMPLEMENTACIÓN BÁSICA DE REDES MONO NEURONALES

David Gómez | Redes Neuronales Artificiales | 29 de octubre de 2014

Planteamiento

Se quería diseñar un programa para representar las formas más básicas de redes neuronales, redes de una sola neurona. Más concretamente, un Perceptron simple y un Adaline; que fueran capaces de aprender, a partir de un conjunto de un máximo de 100 puntos, a identificar el hiperplano separador presente entre ellos.

Base conceptual

El perceptron es la neurona artificial y unidad básica de inferencia en forma de discriminador lineal, es decir, un algoritmo capaz de generar un criterio para seleccionar un sub-grupo, de un grupo de componentes más grande. La limitación de este algoritmo es que si dibujamos los elementos en un plano, se deben poder separar con un hiperplano los elementos "deseados" de los "no deseados". El perceptrón puede utilizarse con otros perceptrones u otro tipo de neurona artificial, para formar redes neuronales más complicadas.

Un perceptron dentro de una red neuronal se comporta como un nodo, estando formado por un conjunto de pesos de entrada (arcos), un algoritmo de aprendizaje (que genera el contenido del nodo) y una vía de salida que puede ser una conexión hacia otros perceptrones (por lo general un arco o a veces una variable que aloje el resultado de las operaciones)

El modelo biológico más simple de un perceptrón es una neurona y viceversa. Es decir, el modelo matemático más simple de una neurona es un perceptrón. La neurona es una célula especializada y caracterizada por poseer una cantidad indefinida de canales de entrada llamados dendritas y un canal de salida llamado axón. Las dendritas operan como sensores que recogen información de la región donde se hallan y la derivan hacia el cuerpo de la neurona que reacciona mediante una sinapsis que envía una respuesta hacia el cerebro, esto en el caso de los seres vivos.

El Adaline es muy similar al perceptron, de hecho la única diferencia con el perceptron está en el algoritmo de aprendizaje. Mientras el perceptron itera hasta que la respuesta obtenida coincida con la respuesta deseada o hasta que se alcance un tope máximo de iteraciones; el Adaline tiene en cuenta además el grado de corrección de la salida obtenida respecto a la deseada. Esto se consigue mediante la aplicación de la regla Delta, que se define como el valor absoluto de la diferencia entre la salida obtenida y la deseada.

Tanto el Adaline como el perceptron pueden usarse como unidades de procesamiento para armar redes neuronales más complejas.

Desarrollo de la aplicación

Se comenzó investigando la base conceptual de ambas clases de neurona e inspeccionando el código de aplicaciones similares para obtener guías y referencias sobre los valores óptimos de las variables básicas usadas en cada neurona: como el valor de la taza de aprendizaje, número de iteraciones óptimo o el valor del umbral (neurona o entrada sub-cero, que se conecta para balancear el perceptron a su valor de "vacio" o nulo) para salidas de tipo bipolar {-1,1}).

Tras haber formado la interfaz y estructuras de clase necesarias, se comenzó a desarrollar el algoritmo de aprendizaje del perceptron, lo que puede considerarse la base algorítmica de la aplicación.

Perceptron:

A manera general: Los puntos a diferenciar, se almacenan en una lista de objetos de tipo Pun, una estructura que aloja las coordenadas X y Y del punto y su signo. Esta lista de puntos es el parámetro que se suministra al método Learn() para el aprendizaje del perceptron.

El método consiste en un ciclo mientras que itera mientras no se alcance el límite de iteraciones y suceda que la salida actual del perceptron sea distinta a la salida deseada. Dentro de este bucle, se va iterando la lista de puntos y en cada iteración:

1. Se asigna el valor de la coordenada X a la entrada 1.
2. Se asigna el valor de la coordenada Y a la entrada 2.
3. Se obtiene la salida actual mediante la función signo de la sumatoria de la multiplicación del valor de cada entrada por su peso correspondiente.
4. Si hay error (discrepancia de valores)
    4.1. Se actualizan los pesos de las entradas al sumarle la multiplicación de la taza de aprendizaje, por la resta de 1, menos la multiplicación de la salida deseada por la salida actual, por la multiplicación de la deseada con el valor de la entrada correspondiente. Siendo que esta es la ecuación para perceptrones con salidas bipolares.
5. Si no hay error, se avanza al próximo punto.

El final del ciclo mientras ocurre si en ninguno de los puntos ha habido error. La autocorrección que el perceptron aplica a si mismo ocurre porque la salida depende de dos factores: las entradas y los pesos de esas entradas. Entradas con pesos mayores tendrán más influencia en el valor final del perceptron, y entradas con menos peso menor influencia. Así, cada vez que hay un error, aquellas entradas que acercan el valor a la salida deseada ganan más peso que aquellas que no lo acercan tanto.

Idealmente, el perceptron encontrara el balance adecuado de los pesos que permite que todos y cada uno de los escenarios obtenga la salida deseada.

Adaline:

El desarrollo del adaline fue absolutamente similar, dado que amabas neuronas tienen mucho en común. El único cambio ocurrió dentro del método de aprendizaje y las funciones de transferencia (aquellas que efectúan el sumario de las entradas y sus pesos).

Mientras el perceptron utiliza la función signo para decir si hay o no error, el adaline requiere un aproximamiento más sutil dado que además se necesita saber "cuanto error" hay. Para ello, la función de transferencia ha de ser cambiada a la tangente hiperbólica de la sumatoria de las entradas multiplicadas por sus pesos.

El método de aprendizaje se modificó a:

1. Por cada punto en la lista de puntos:
2. Se obtiene la salida deseada del punto.
3. Se obtiene la salida actual por medio de un método al que se le suministra la el valor de la coordenada X y la coordenada Y.
4. Se obtiene el error resultante de esta salida como: un medio, del cuadrado de la resta entre la salida deseada menos la salida actual.
5. Mientras el error obtenido supera el error mínimo tolerado
    5.1. Se iterara por cada punto en la lista de puntos:
        5.1.1. Se actualizan los pesos de las entradas al sumarle la multiplicación de la taza de aprendizaje, por la resta de la salida deseada menos la salida obtenida para este punto. Multiplicada por la salida para el punto actual (pero obtenida mediante la derivada de la función de transferencia), por la entrada correspondiente para el punto actual. Siendo que esta es la ecuación para perceptrones con salidas bipolares con entrenamiento patrón a patrón. Esto es, reentrenar la neurona por cada vez que se presente un patrón nuevo.
    5.2. Se actualizan tanto la salida actual como el error para esta nueva salida y si el error está dentro de lo tolerable, se avanza al próximo punto.

Tanto para el perceptron como el adaline, el trazo se obtiene al final del aprendizaje al evaluar el valor de los pesos. Básicamente se obtienen las coordenadas de las esquinas del plano donde se va a dibujar el trazo, se multiplica por los pesos para obtener el ángulo con que se desplaza la recta y se incremente según sea el tamaño del plano.

Si los elementos son linealmente separables, la recta debiera marcar la frontera entre ambos grupos de puntos.

Nota: ver el código del programa para mejor referencia.

Uso de la interfaz

Las interfaces del perceptron y el adaline son bastante similares y funcionan más o menos de igual manera.

Botones:

Ingresar puntos:

Permite registrar puntos en el panel al hacer clic con el botón izquierdo (para puntos azules) o el botón derecho (para puntos fucsias).

Limpiar:

Borra los puntos que se ingresaron en el panel.

Guardar:

Indica que se ha terminado de ingresar todos los puntos y habilita el botón de procesar.

Procesar:

Suministra la colección de puntos al perceptron o adaline y traza la línea divisora según los resultados obtenidos.

Sliders:

Los sliders indican los valores fundamentales que necesita la neurona para trabajar.

Archivo de puntos

Los puntos ingresados en el panel quedan guardados en un archivo texto que se carga automáticamente la próxima vez que inicie el programa.