REDES NEURONALES DE COMPETENCIA DE TIPO GANADOR TOMA TODO. APLICACIONES

Dra. Yesenia Eleonor González Navarro
ygonzalezn@ipn.mx
M. en C. Paola Nayeli Cortez Herrera
pcortez@ipn.mx
M. en C. Maricela Serrano Fragoso
mserranof@ipn.mx

Unidad Profesional Interdisciplinaria en Ingeniería y Tecnologías Avanzadas-IPN. Ciudad de México.

Resumen

En este trabajo se presenta la topología y algoritmo de entrenamiento de las redes neuronales artificiales de competencia de tipo ganador toma todo (Hagan M., Demuth H., 2014; Jang J., Sun C., 1997). Se han desarrollado 2 aplicaciones, la primera de ellas muestra el uso de estas redes para búsqueda de patronesp rototipod eu na distribución de datose n2D y l a segunda muestra su comportamiento como mapas de rasgos auto-organizados, ingresando una imagene ne scalad eg risesc omop atrónd ee ntrenamiento.

Palabras clave: Redes neuronales de competencia, competencia de tipo ganadort omat odo, mapasd er asgosa uto-organizados.

Introducción

Una clasificación que puede realizarse de acuerdo a su topología en las redes neuronales artificiales es por el número de capas que poseen, de tal forma que se pueden diseñar redes monocapa (una sola capa) y multicapa (de 2 capas en adelante). Otra clasificación para las redes neuronales artificiales es de acuerdo a si pueden establecer una relación únicamente con sus entradas actuales o bien con sus entradas actuales y previas, llamándolas redes estáticas o dinámicas (Hagan M., Demuth H., 2014). El tipo de red que se describirá a continuación es estática y utiliza una topología monocapa, se conocen como redes de competencia. Existen dos tipos de competencia: competencia de tipo ganador toma todo (WTA, por sus siglas en inglés) que es la más usada, y competencia de tipo ganador comparte todo (WSA). Las redes de competencia son sistemas no supervisados, es decir, no requieren que se le indique al sistema la salida deseada para cada de entrada (Hagan M., Demuth H., 2014; Jang J., Sun C., 1997).

Para el tipo de competencia **ganador toma todo**, como su nombre lo dice, la neurona que tenga una mayor semejanza con el patrón que se está presentando al sistema será la única que emita una salida en alto y las neuronas perdedoras darán una salida en bajo. Si el tipo de competencia es **ganador comparte todo**, será un grupo de neuronas quienes generen la salida en alto y la selección de ese grupo dependerá del algoritmo de entrenamiento que se vaya a utilizar.

En este trabajo se describe la topología y el algoritmo de entrenamiento para redes de competencia de tipo **ganador toma todo** y para una mejor comprensión, se han desarrollado 2 aplicaciones.

Desarrollo

A. Topología de red

La topología que utilizan las redes de competencia es sencilla y se conoce como asociador lineal (Hagan M., Demuth H., 2014), que utiliza una función de activación de transferencia lineal, ver Figura 1.

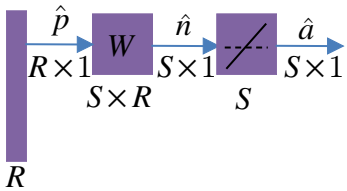


Figura 1. Topología para redes de competencia (asociador lineal).

La ecuación de salida de la red está dada por:

$$\hat{a}=\hat{n}, \quad (1)$$

donde

$$\hat{n}=W\hat{p} \quad (2)$$

y

$$W=\begin{bmatrix}\hat{w}_1^T\\ \hat{w}_2^T\\ \vdots\\ \hat{w}_S^T\end{bmatrix} \quad (3)$$

En (3), el superíndice T representa la transpuesta del vector, los subíndices indican a qué neurona pertenece cada vector de peso. La matriz W será de dimensiones $S\times R$ (S número de neuronas por R número de elementos de entrada).

Esta topología considera que se tienen patrones de entrada $\hat{p}$ de R número de elementos. Cada neurona está representada únicamente por su vector de peso $\hat{w}$ y, para S número de neuronas, puede crearse una matriz W compuesta de los vectores $\hat{w}$ transpuestos (3). Se tendrá un vector de salida $\hat{a}$ y su número de elementos es igual al número de neuronasd elared.

B. Proceso de competencia

En (Hagan M., Demuth H., 2014) se utiliza el fenómeno de competencia entre neuronas basado en (2). Si se retoma la ecuación de producto punto entre 2 vectores (por ejemplo $\hat{p}_1$y $\hat{p}_2$)se tiene (Swokowski E., Cole J,2009):

$$\hat{w} \bullet \hat{p} = |\hat{w}||\hat{p}| \cos\theta, \qquad (4)$$

donde θ es el ángulo entre los 2 vectores. Si ambos vectores tienen magnitud unitaria entonces el valor resultante en (3) únicamente dependerá del ángulo entre ellos. ¿Cómo se pueden aprovechar esta ecuación desdee lpuntodevistad ereconocimientod e patrones?

Tomando como ejemplo un sistema en donde se presenta un patrón $\hat{p}$ de 2 elementos, y existen 2 neuronas representadas por sus vectores de peso (también de 2 elementos) $\hat{w}_1$ y $\hat{w}_2$, respectivamente, de acuerdo a la representación de la Figura 2.a (todos los vectores están normalizados), se dice que el patrón $\hat{p}$ "se parece más" a $\hat{w}_1$ que a $\hat{w}_2$, ya que el ángulo $\hat{\theta}_1$ es menor que el ángulo $\hat{\theta}_2$. Se asume que todos los vectores parten del origen y tienen como punto final sus valores numéricosc orrespondientes.

Por lo tanto, el producto punto en (2) da como resultado un valor numérico proporcional al ángulo entre los vectores y de esa forma puede establecerse la semejanza o parecido entre los patrones de entraday lasneuronasd euna red (Hagan M., Demuth H., 2014).

Sin embargo, usando el producto punto para establecer semejanza entre vectores se da el inconveniente de que, al realizar la normalización de los vectores en el sistema, la distribución de datos original sufre una modificación y esto la mayoría de las veces es perjudicialparae lprocesod ereconocimiento.

Una variante que puede implementarse entonces para establecer semejanza o parecido entre vectores es obtener la distancia euclidiana

entre ellos (Jang J, Sun C., 1997). Observe (a partir de la Figura 2.a) que el punto final de cada vector también puede visualizarse como un punto en un plano cartesiano referido a 2 ejes, por lo que la misma información se interpreta ahora como una distribución de datos en 2D (Figura 2.b). Observe que la distancia euclidiana d_1 entre el vector $\hat{w}_1$ y $\hat{p}$ será menor que la distancia euclidiana d_2 (entre el vector $\hat{w}_2$ y $\hat{p}$), lo que indicaría, al igual que con el cálculo del producto punto que $\hat{w}_1$ "se parece más" a $\hat{p}$. La ventaja de aplicar el cálculo de distancia euclidiana es que no es necesario normalizar los vectores, por lo que la distribución de datos original del sistema se mantiene.

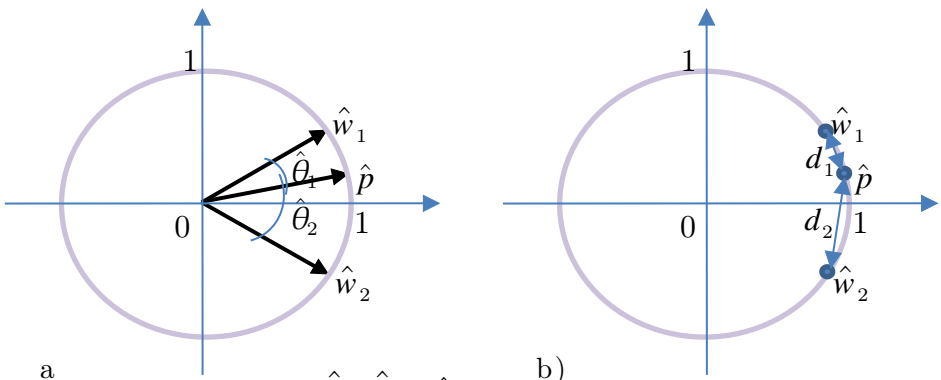


Figura 2. a) Vectores normalizados $\hat{w}_1$, $\hat{w}_2$ y $\hat{p}$. b) Distribución de datos en 2D que muestra el punto final de los vectores $\hat{w}_1$, $\hat{w}_2$ y $\hat{p}$ y la distancia euclidiana entre cada uno de los vectores de peso y el vector de entrada $\hat{p}$.

C. Algoritmo de entrenamiento

Para el tipo de competencia **ganador toma todo**, puede utilizarse la regla de aprendizaje Instar (Hagan M., Demuth H., 2014):

$$\hat{w}_i(k) = \hat{w}_i(k-1) + \alpha a_i(k)(\hat{p}(k) - \hat{w}_i(k-1)). \quad (5)$$

La actualización de los vectores de peso en la k-ésima iteración está en función del valor de peso anterior, del valor del parámetro α conocido como razón de aprendizaje (que es el que controla la rapidez o tamaño de paso con que el algoritmo tiende a la solución y satisface $0<\alpha$), de la salida de la neurona a la cual está asignado ese vector de peso $\hat{w}$ y del vector $\hat{p}$ que ingresa al sistema. Se asume que la neurona que "gana" su salida es en alto y las neuronas "perdedoras" tendrán salida en bajo. En (5) observe que, si la salida de la neurona es cero, el valor de peso nuevo es igual al valor de peso anterior (no se actualiza). Como ya se

mencionó, el entrenamiento competitivo es no supervisado, ya que la ecuación de aprendizaje (5) no incluye un error, que implicaría monitorear que el sistema entregue un valor de salida deseado para cadaentrada.

El proceso de entrenamiento consiste en que una vez establecida la topología de red (número de neuronas a utilizar), se inicializan los vectores de peso $\hat{w}$ y se ingresa al sistema uno de los vectores $\hat{p}$ de entrenamiento. Aplicando el cálculo de distancia euclidiana del patrón de entrada $\hat{p}$ hacia cada uno de los vectores $\hat{w}$ se establece cuál neurona se parece más al patrón que ingresó (menor distancia) y se le asigna salida con valor 1, las demás neuronas tendrán salida con valor 0. Las salidas de cada neurona se sustituyen en sus ecuaciones de actualización de valores de peso (5) respectivas y, de acuerdo con esa ecuación en esa iteración únicamente se actualizará el vector de peso $\hat{w}$ de la neurona ganadora. El proceso de entrenamiento prosigue ingresando uno a uno los siguientes vectores de entrenamiento $\hat{p}$ y actualizando en cada iteración las ecuaciones de $\hat{w}$. Regularmente deben aplicarse varios barridos de todos los patrones de entrenamiento (épocasdeentrenamiento)paraqueelsistemaencuentrelasolución.

D. Aplicaciones

D.1 Búsqueda de patrones prototipo en una distribución de datos en 2D

La primera aplicación propuesta para una red de competencia con entrenamiento de tipo **ganador toma todo** es que, a partir de una distribución de datos en dos dimensiones (ver Figura 3), el algoritmo de entrenamiento encuentre los patrones prototipo (que representen a cadagrupoclase)delsistema.

Figura 3. Distribución de datos en 2 dimensiones. Implementación de una red competitiva de 3 neuronas
para la búsqueda de patrones prototipo.

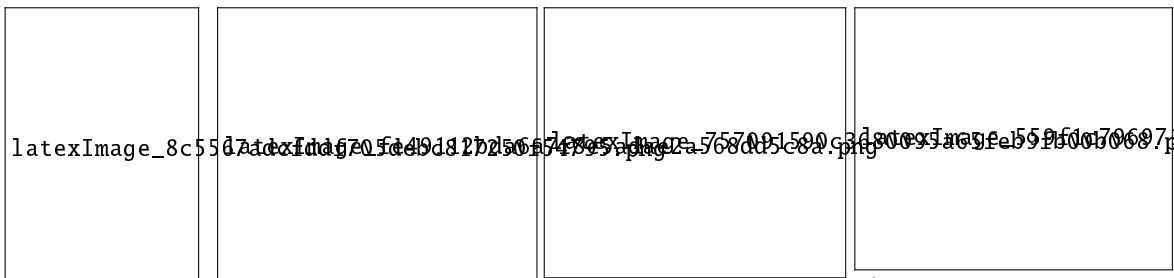
Para el ejemplo de la Figura 3 se propuso una topología de 3 neuronas con valores aleatorios de inicio (se localizan en el extremo sin carácter de cada una de las 3 curvas de aproximación en la imagen), después de aplicar el algoritmo de entrenamiento competitivo cada una de las neuronas (representadas por sus vectores de peso $\hat{w}$) se posicionaron en el centro de una región de datos (cúmulo), volviéndose sus patrones "representativos" o "prototipo" (extremo de las curvas con carácter triangular). Debido a que el entrenamiento es no supervisado, no se puede elegir a priori qué neurona deberá irse a cierto cúmulo, sino que dependerá de los valores iniciales de peso. En la Figura 3 pueden observarse las trayectorias que cada una de las 3 neuronas siguió para colocarse al centro de cada cúmulo. De acuerdo al número de neuronas propuestas, el sistema separará la distribución de datos en ese número de cúmulos.

D.2 Mapas de rasgos auto-organizados

Aunque en (Hagan M., Demuth H., 2014; Jang J, Sun C., 1997) se describen a los mapas de rasgos auto-organizados como sistemas que se basan en el tipo de competencia **ganador comparte todo**, es posible aplicar el tipo de competencia **ganador toma todo**. El principio de entrenamiento es idéntico al utilizado en la aplicación D.1, pero ahora seutilizará unnúmero elevadode neuronas, por lo que elresultado final es que cada neurona tenderá hacia el centro de cúmulos formados por pocos datos de entrenamiento. No puede usarse la palabra

"aproximación" en estricto para el resultado final, debido a que la ecuación de aprendizaje es no supervisada y no se puede establecer el errord ea proximación final.

La Figura 4.a muestra la imagen en escala de grises que se introdujo a la red. La Figura 4.b muestra la representación en 3D de cada pixel de la imagen, referido a sus coordenadas horizontal (n), vertical (m) y nivel de gris (intensidad de pixel). La Figura 4.c muestra la nube de neuronas (vectores de peso $\hat{w}$ iniciales) y la Figura 4.d muestra la posición final (despuésd ele ntrenamiento)d eca dauno d elos vectoresd ep eso $\hat{w}$.



a) b) c) d)

Figura 4. a) Imagen de entrada en escala de grises. b) Representación de cada pixel de la imagen referido a sus coordenadas horizontal, vertical y nivel de gris. c) Nube de vectores de peso $\hat{w}$ iniciales. d) Posición final (después del entrenamiento) de cada uno de los vectores de peso $\hat{w}$.

Para los resultados mostrados en la Figura 4, se utilizó una imagen de entrada de 120x160 pixeles. Se emplearon 4048 neuronas y 20 épocas de entrenamiento. Ambas aplicaciones se implementaron en el lenguaje deprogramación Matlab®.

Referencias

Hagan M., Demuth H. (2014). Neural Network Design (2nd Ed.). Ed. Martin Hagan.

Jang J., Sun C. (1997). Neuro-Fuzzy and Soft Computing. A Computational Approach to Learning and Machine Intelligence. Ed. Prentice Hall.

Swokowski E., Cole J. (2009). Algebra y Trigonometría con Geometría Analítica (12da Ed.). Ed. Cengage Learning.