Aprendizaje en Redes Coulombianas

J.A. HORAS*, E.A. BEA y P.M. PASINETTI

UNIVERSIDAD NACIONAL DE SAN LUIS. FACULTAD DE CIENCIAS FÍSICO-MATEMÁTICAS Y NATURALES
DEPARTAMENTO DE FÍSICA. INSTITUTO DE MATEMÁTICA APLICADA SAN LUIS
EJÉRCITO DE LOS ANDES 950 - (5700) SAN LUIS - ARGENTINA
e-mail: jhoras@unsl.edu.ar

Resumen

Se define la energia potencial de una colección de sitios de memoria, haciendo que los de la mísma clase se atraigan y los de distinta clase se repelan. Usando el potencial coulombiano, que representa las características antedichas, es posible obtener un algoritmo de aprendizaje supervisado aplicable a una red de una o más capas. El mencionado algoritmo se implementa en una red neuronal de estructura modular y se muestra su funcionamiento.

Se analizan curvas de aprendizaje, estudiandose tambien la calidad de la convergencia para el caso del XOR.

Abstract

The energy of a memory sites ensamble is defined using the coulombian potential, this makes sites belonging to the same class become mutually attractive and the ones belonging to the different class be repelled.

Following the above, the known coulombian learning algorithm is implemented in a modular neural networth in which each layer can be trained independently.

We analize learning curves, studying also the convergence quality in the typical case of XOR.

Introducción

Analizamos en este trabajo un algoritmo de aprendizaje para la red Coulombiana N-dimensional [1,2,4] que es aplicable para redes de una o mas capas y desarrolla representaciones distribuidas. Este algoritmo no depende de la propagación de errores a través de las capas o sea que el sistema generado es modular.

La idea central es definir una energía potencial de una colección de sitios de memoria que pueden agruparse en una o más clases. Asignando entonces potenciales atractivos y repulsivos entre ellos, los patrones de la misma clase se atraerán entre sí y los de distintas clases se repelerán .

Existen gran variedad de algoritmos de aprendizaje [3],el que analizamos aquí tiene la particularidad de que cada capa puede ser entrenada *independientemente.*

El algoritmo de aprendizaje

Consideremos un sistema de M sitios de memoria $\mathbf{x}_1,\ldots,\mathbf{x}_M$ en R^N y asumamos que pueden agruparse en subconjuntos pertenecientes a varias clases de patrones. Los mencionados sitios se definen según [4]

$$\mathbf{x}_i = \sum_{n=1}^{N} \mathbf{e}_n F_n\left(\sum_{m=1}^{K} \omega_{nm} f_{mi}\right) \qquad (1)$$

donde $\mathbf{e}_n$ es el vector unitario que identifica el enésimo neurodo y $\mathbf{f}_i$ es un patrón de entrada en el espacio R^K . $F_n(h_i)$ es cualquier función de activación, en particular tomamos la función logística

$$F_n(h_i) = \frac{1}{1+e^{-\frac{1}{T}(h_i+\theta)}} \qquad (2)$$

donde

θ : es el bias que puede tomarse también como ω_{no}.

ω_{nm}: componente de la matriz de pesos.

El vector $\mathbf{x}_i$ es entonces el resultado de mapear el patrón de entrada $\mathbf{f}_i$ a través de la matriz de pesos $\mathbf{W}$.

* Investigador del CONICET.

La "Energía Potencial Electroestática" de una configuración dada de estos sitios de memorias es

$$\Psi = \frac{1}{2L}\sum_{i=1}^{M}\sum_{j=1}^{M} Q_i Q_j \left\| \mathbf{x}_i - \mathbf{x}_j \right\|^{-L} \qquad (3)$$

Tomamos Q_i (c) la "carga" en un sitio de memoria i de clase c, tal que

$$\begin{aligned} \text{sign}(Q_i(c)) &\neq \text{sign}(Q_j(c')) \quad \text{for } c = c \\ \text{sign}(Q_i(c)) &= \text{sign}(Q_j(c')) \quad \text{for } c \neq c \end{aligned} \qquad (4)$$

Nótese que las "cargas" deben ser tomadas de a pares.

Si permitimos que la posición de las cargas cambie, intuitivamennte se ve que los sitios de la misma clase (pero con pares de cargas de signos diferentes) se atraerán, ocurriendo lo contrario para aquellos sitios que tengan carga del mismo signo. Tales movimientos en el espacio de cargas son faciles de lograr en una red Coulombiana permitiendo que la matriz de pesos **W** evolucione. En consecuencia, aprendizaje [4] en estas redes implica direccionar la evolución de la matriz de pesos **W** de forma tal de minimizar la energia potencial Ψ (ec.(3)) de la colección de cargas. Usamos para ello gradiente descendente, que para el cambio de pesos da

$$\delta\omega_{nm} = -\eta\frac{\partial\Psi}{\partial\omega_{nm}} \qquad (5)$$

$$= \frac{1}{2}\eta\sum_{i=1}^{M}\sum_{j=1}^{M} Q_i Q_j \left\| \mathbf{R}_{ij} \right\|^{-(L+2)} \Delta_{nm}\left(\mathbf{f}_i, \mathbf{f}_j\right)$$

con

$$\mathbf{R}_{ij} = \mathbf{x}_i - \mathbf{x}_j \qquad (6a)$$

y

$$\frac{\partial \mathbf{x}_i}{\partial\omega_{nm}} = \mathbf{e}_n F_n\left(1 - F_n\right) f_{mi} \qquad (6b)$$

$$\Delta_{nm}\left(\mathbf{f}_i, \mathbf{f}_j\right) = \mathbf{R}_{ij} \cdot \frac{\partial \mathbf{R}_{ij}}{\partial\omega_{nm}} \qquad (6c)$$

$$= \left(F_n\left(h_i\right) - F_n\left(h_j\right)\right)\left(F_n\left(h_i\right)\left(1 - F_n\left(h_i\right)\right) f_{mi} - F_n\left(h_j\right)\left(1 - F_n\left(h_j\right)\right) f_{mj}\right)$$

La variación de pesos ec.(5), puede calcularse también mediante la sucesiva acumulación de

$$\delta\omega_{nm} = \pm\eta\left\|\mathbf{x}(t) - \mathbf{x}(t+1)\right\|^{-(L+2)} \Delta_{nm}\left(\mathbf{f}(t), \mathbf{f}(t+1)\right) \qquad (7)$$

donde η es la "velocidad" de aprendizaje, el signo negativo es para patrones subsecuentes (tomados al azar) de la misma clase y el signo positivo para patrones de diferentes clases.

La implementación de la regla de actualización de pesos según ecs. (6) o (7) da lugar a la actualización periódica (tipo batch) o contínua (tipo on-line) respectivamente.

Simulaciones

Con el objetivo de estudiar tanto la velocidad como la calidad de convergencia del algoritmo de aprendizaje de la red coulombiana, se produjeron curvas de aprendizaje para el caso del XOR [3]. En este conocido benchmark para aprendizaje exhaustivo la salida debe ser *cero* para entradas (0,0) y (1,1) debiendo ser *uno* para entradas (0,1) y (1,0). La regla de actualización de pesos para aprendizaje supervisado (ec.(7)) se implementó en una red neuronal estudiándose también los resultados obtenidos para distintos valores de la energía en la primer capa.

La arquitectura de la red utilizada y las características del entrenamiento fueron las siguientes:

1) Dos capas. Dos o más neurodos en la primer capa y uno en la segunda (arquitectura fija).
2) Actualización de pesos tipo on-line.
3) Los parámetros utilizados fueron: El parámetro de aprendizaje η=2.25. El exponente L=1 (ec.(3)). La "Temperatura" T=1 (ec.(2)). Inicialización de pesos, al azar entre [-0.05,0.05].
4) Otras Condiciones.

- Para evitar divergencias en la actualización de pesos, se sumó la cantidad constante 1 en el denominador de la ec.(4). Ello es equivalente a adicionar un término repulsivo al potencial Coulombiano. El efecto de diversas formas de potencial será estudiado con más detalle en futuras presentaciones.

- Se adicionaron cargas "virtuales" que tienen una posición fija en el espacio de actividad de las capas y por ello no intervienen en la actualización de pesos.

Los resultados se muestran en las figuras 1, 2, 3 y 4. En la fig. 1 se grafica el n° de errores (normalizados a uno) vs. el n° de presentaciones. En el modo de operación contínua los pesos son actualizados coincidentemente con la presentación de cada par de ejemplos de entrada

y salida. Número de presentaciones se refiere en este trabajo al número de veces en que fueron presentados los cuatro patrones.

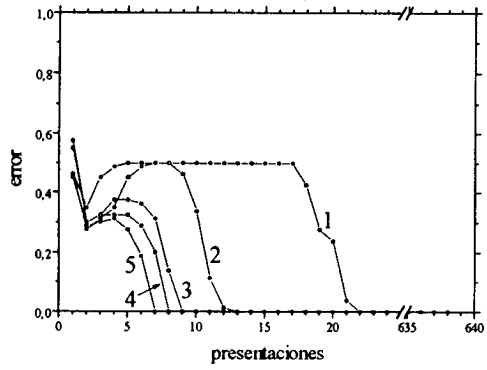

Figura 1: n° de errores versus n° de presentaciones. Los puntos son un promedio de 20 ensayos y su dispersión varía entre 0 y 0.25 .

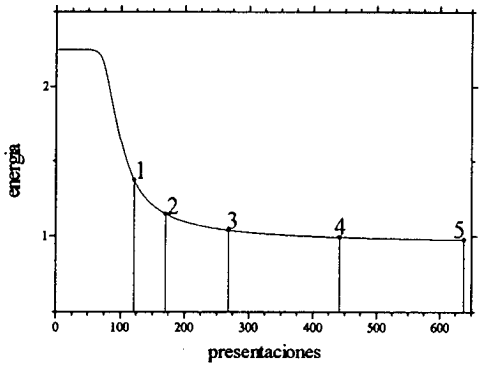

Figura 2: energia versus el n° de presentaciones en la primer capa. Las cifras ilustran los puntos en que se interrumpió el entrenamiento en esta capa.

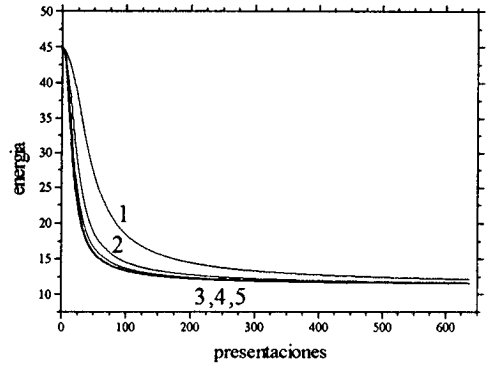

Figura 3: energia versus n° de presentaciones en la segunda capa. Los números identificatorios se refieren al punto en que se interrumpió el entrenamiento en la primer capa.

Las cifras que identifican las graficas en la fig. 1 coinciden con los mostrados en la fig. 2, donde se grafican el valor de la energia vrs. el n° de presentaciones en la primer capa. Para ilustrar y mostrar la calidad de la convergencia referida a la cantidad de entrenamiento de la segunda capa, se muestra en la fig. 3 el gráfico de la energia de esta capa vrs. el n° de presentaciones en ella.

Por último, en la fig. 4 se muestra que la adición de nodos en la primer capa no produce efectos significativos, puesto que las variaciones estan por debajo de la dispersión estadística. El entrenamiento fue: primer capa, cantidad fija de 638 presentaciones. Segunda capa, el necesario para obtener error cero.

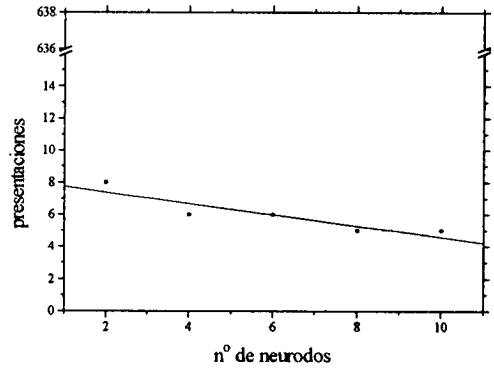

Figura 4: En la ordenada se muestra el n° de presentaciones en la segunda capa necesarias para lograr error cero. En la abscisa se muestra el n° de neurodos en la primer capa (ver texto). La linea contínua es solo una guía para el ojo.

Conclusiones

Los resultados obtenidos permiten concluir básicamente lo siguiente:

- La implementación de este algoritmo resulta interesante para ser utilizado en lugar del conocido algoritmo tipo perceptrón dado que permite resolver un problema linealmente no separable como el XOR mediante el entrenamiento independiente de solo dos capas.

- Las curvas de aprendizaje muestran una buena convergencia no fuertemente dependiente del entrenamiento de la primer capa. En otras palabras, no existiría peligro de sobrentrenamiento en este tipo de red.

- El número de neurodos en la primer capa no afecta fuertemente la convergencia, lo cual

implicaría la tendencia de este algoritmo a la generación de redes de mínima complejidad.

Referencias

1 - C. M. Bachmann, L. N. Cooper, A. Dembo, and O. Zeitouni, Proc. Natl. Acad. Sci. U.S.A. **84**, 7529-7531 (1987).

2 - A. Dembo and O. Zeitouni, Physical Review A **37**, 2134-2143 (1988).

3 - J. Hertz, A. Krogh, and R. G. Palmer, *Introduction to the theory of neural computation* (Addison-Wesley, New York, 1991).

4 - C. L. Scofield, ICNN Vol. 1, 271-275 (1988).