Aprendizaje por Diferencias Temporales: Aplicación a la Diagramación de Sistemas de Tiempo Real

Ing. Ricardo Cayssials - Ing. Omar Alimenti

D^{to} de Ing. Eléctrica - Instituto de Ciencias e Ing. en Computación

Universidad Nacional de Sur

email:{iealimen, iecayss}@criba.edu.ar

Resumen

Las Redes Neuronales (NN) pueden proveer soluciones a problemas en los cuales es necesario predecir, en base a observaciones del estado actual del sistema, el desenlace de un proceso. Una forma de lograr este objetivo es utilizar técnicas de **Aprendizaje por Diferencias Temporales** (TDL) para el entrenamiento de una Red Neuronal. En este trabajo se aplica dicha forma de aprendizaje para diseñar una Red Neuronal capaz de predecir con suficiente antelación una crisis en la diagramación de un Sistema de Tiempo Real.

I. Introducción

Existen trabajos en los que se aplican NN para intentar pronosticar el resultado de un determinado proceso [1],[2]. Un ejemplo de tales aplicaciones es el de predecir el desenlace de un juego observando una determinada posición del tablero; es decir si dicha posición conducirá, o no, a "ganar". Es posible utilizar una NN para elegir una movida (dentro de todas las legales) que permita llegar a una posición del tablero que tenga las mayores posibilidades de alcanzar la victoria. En dicho caso, la NN debe imitar el conocimiento que posee un jugador experto.

El empleo de técnicas de aprendizaje como el "backpropagation", son de difícil utilización en este tipo de aplicaciones debido a la gran cantidad de posiciones que se deben almacenar antes de conocer el final de la partida. La técnica de TDL[1] entrena a la NN mientras se desarrolla el juego, sin necesidad de acumular las posiciones intermedias hasta el final de la partida. De esta manera se logra disminuir considerablemente la complejidad espacial del algoritmo.

Existen dos técnicas de aprendizaje que permiten aproximar la función de evaluación para obtener una NN experta en el juego: aprendizaje supervisado (SL) y aprendizaje reforzado (RL)[1]. La primera necesita que un experto brinde las evaluaciones de las diferentes movidas que pueden ocurrir en una partida. Esto requiere grandes cantidades de ejemplos y las evaluaciones pueden no ser muy precisas.

En la segunda, la misma técnica de aprendizaje genera los ejemplos de entrenamiento, verifica los resultados y refina la función de evaluación. El RL es un algoritmo que requiere menos esfuerzo humano para generar ejemplos pues sólo es necesario especificar las reglas del juego.

En un sistema de tiempo real, no es suficiente que los resultados aritméticos y lógicos sean correctos, sino que deben ser realizados antes de un determinado tiempo, denominado *vencimiento*. Si una tarea excede su vencimiento se dice que el sistema entra en crisis. Un tema importante de investicación en tiempo real son los sistemas multitarea-monorecurso. En este tipo de sistemas existe un administrador del recurso, denominado *diagramador*, que es el encargado de evitar que tareas excedan su vencimiento.

En este trabajo se obtiene, utilizando TDL, una NN que permite determinar las acciones que el diagramador de un sistema de tiempo real debe efectuar para lograr que el sistema no entre en crisis.

II. Diagramación de Sistemas de Tiempo Real

En ambientes de tiempo real son comunes los sistemas en los que un conjunto de usuarios comparten un único recurso. Bajo el esquema de recurso compartido, pueden identificarse varios sistemas que pueden considerarse isomorfos para el análisis de su comportamiento en tiempo real. Un ejemplo de ello son los protocolos de acceso al medio en redes locales y los sistemas multitarea- monoprocesador. En particular, en lo que sigue, nos referiremos a estos últimos aún cuando los resultados son utilizables en otros sistemas de tiempo real.

Debido a que cada tarea tiene restricciones temporales para la utilización del procesador, deben garantizarse las *asignaciones* del mismo (*activación* de una tarea) que permitan una completa ejecución de cada tarea dentro de un cierto intervalo a partir del instante en que está *lista* para ser ejecutada (*generación* de la tarea). Este intervalo es denominado *plazo de vencimiento* (D, "deadline") de la tarea. Luego, a fin de permitir que toda activación se produzca antes de los vencimientos, resulta imprescindible la utilización de un diagramador de tiempo real. Si el sistema tareas-diagramador-procesador, garantiza que todas las tareas se ejecutarán antes de sus vencimientos, el sistema se dice *diagramable*. El diagramador debe determinar, mediante una disciplina de prioridades, la tarea a la que se le asigna el recurso, es decir, establecer cuál será la próxima tarea a ser ejecutada. Las disciplinas de diagramación [4] para tiempo real más estudiadas son: Rueda Cíclica, Menor Tiempo a Crisis (óptimo) y Períodos Monotónicos Crecientes (Prioridades Fijas).

El conjunto de n tareas que conforman el sistema de tiempo real será representado por $S(n) = \{ \tau_1, \tau_2, ..., \tau_n \}$. El tiempo se considera ranurado. T_r es el tiempo de duración de la ranura. Las tareas en un sistema de tiempo real son especificadas por $\tau_i = \{ C_i , T_i , D_i \}$ donde C_i , T_i y D_i representan el máximo tiempo de ejecución, el período de generación y el vencimiento de la tarea i respectivamente.

Varios métodos para comprobar la diagramabilidad han sido propuestos para las diversas disciplinas de prioridades [4][5]. Todos estos métodos necesitan conocer todos los parámetros del sistema para determinar su diagramabilidad en forma necesaria y/o suficiente. En este trabajo se describe la implementación de una NN que evalúa la diagramabilidad del sistema en tiempo de corrida, pudiéndole informar al diagramador si existe la posibilidad de que el sistema entre en crisis. En la figura 1 se muestra un esquema del sistema de tiempo real propuesto.

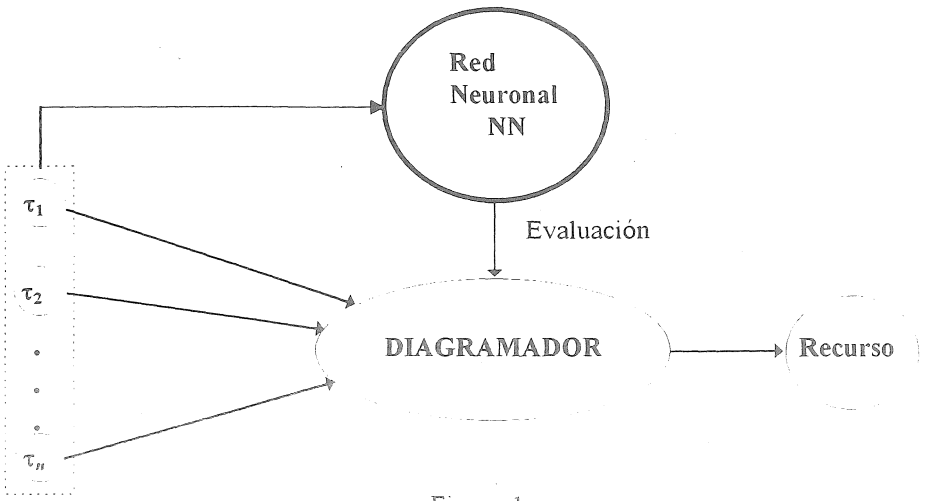

Figura 1

Con este esquema, el diagramador podrá modificar su disciplina de prioridades para minimizar el daño cuando el sistema esté por entrar en crisis. Si bien las teorías realizadas hasta la actualidad pueden determinar *a priori* la diagramabilidad de un sistema, no son tolerantes a variaciones en los parámetros de las tareas. Por otro lado, la complejidad que involucra el cálculo de diagramabilidad por dichas teorías, imposibilita que se puedan realizar en tiempo de corrida. La NN evaluará en todo momento la posibilidad de que el sistema entre en crisis, haciendo que el sistema tolere pequeñas variaciones en los parámetros de las tareas (C y T).

III. Aprendizaje por Diferencias Temporales (TDL)

Los métodos por Diferencias Temporales son procedimientos de aprendizaje especializados para problemas de predicción. En éstos, la observación de un vector de estado permite pronosticar el resultado final de la partida. De esta manera, seleccionando en cada jugada el movimiento que desemboque en el estado con mejor evaluación, será posible obtener una secuencia de movimientos ganadora.

La función de evaluación es desconocida y se tendrá que entrenar a una NN para que se comporte como un jugador experto. Una NN modela una función de evaluación V que se define como:

$$V(x_t) = E(r \mid x_t)$$

siendo:

x_t: el vector de estado después de t movidas.

$E(r \mid x_t)$: el resultado esperado del juego evaluado por la NN en el estado x_t.

Por lo tanto, dado el vector de estado, una NN actúa como un predictor del resultado del juego.

Por medio de una NN sería posible aprender el modelo, minimizando el error sobre el conjunto de vectores de estado de aprendizaje. Los pesos serán ajustados de acuerdo a una medida del error E_t de la forma:

$$E_t = \frac{1}{2}\left(r - V(x_t)\right)^2$$

siendo:

$V(x_t)$: el resultado esperado.

r: el resultado final del juego.

El error total del juego será:

$$E = \sum_{t=1}^{M}\left(r - V(x_k)\right)^2$$

El ajuste de los pesos (W) de la red, empleando el gradiente descendente, con una tasa de aprendizaje α, será:

$$\Delta W = \sum_{t=1}^{M} -\alpha\left(\frac{\partial E_t}{\partial W_t}\right) = \sum_{t=1}^{M} -\alpha\left(\frac{\partial E_t}{\partial V(x_t)}\right)\left(\frac{\partial V(x_t)}{\partial W_t}\right) \qquad (1)$$

donde α es un número comprendido en el intervalo (0,1)

La ec. (1) se puede escribir como:

$$\Delta W = \sum_{t=1}^{M} \alpha\left(r - V(x_t)\right)\nabla_W V(x_t) \qquad (2)$$

siendo $\nabla_W V(x_t)$ el gradiente de la función de evaluación.

Los métodos TDL proponen manejar el error como diferencias temporales entre sucesivas predicciones de la forma: $V(x_{t+1}) - V(x_t)$. Esta técnica hace posible el aprendizaje cuando el resultado es desconocido sin necesidad de acumular todos los valores intermedios. Considerando:

$$r - V(x_t) = \sum_{m=t}^{M}\left(V(x_{m+1}) - V(x_t)\right)$$

la ecuación (2) se puede escribir como:

$$\begin{aligned}\Delta W &= \sum_{t=1}^{M}\alpha\sum_{k=t}^{M}\left(V(x_{t+1}) - V(x_t)\right)\cdot\nabla_W V(x_t) \\ &= \sum_{t=1}^{M}\alpha\left(V(x_{t+1}) - V(x_t)\right)\cdot\sum_{k=t}^{M}\nabla_W V(x_t)\end{aligned} \qquad (3)$$

En forma incremental la ecuación (3) se puede escribir:

$$\Delta W_t = \alpha\left(V(x_{t+1}) - V(x_t)\right)\cdot\sum_{k=1}^{t}\nabla_W V(x_k) \qquad (4)$$

La ecuación (4) se denomina regla TD(1). Sutton [3] definió los algoritmos TD(λ) que ponderan el grado de aprendizaje de una movida, en forma exponencialmente menor en función de su distancia hasta el final del juego, por medio del parámetro λ.

Los métodos TD pueden usar la información contenida en predicciones intermedias sin conocer resultado final de la partida. Cuando la diferencia ($V(x_{t+1})$ - $V(x_t)$) es muy grande, como se aprecia en (4), esta diferencia se usa para ajustar las evaluaciones de los estados $x_1, x_2,\ldots,x_t$. Una modificación a la ecuación (4) permite realizar una ponderación no uniforme de los pesos de la NN, haciendo un mayor ajuste de los mismos para los estados más recientes. Para ello se emplea el parámetro $\lambda \in [0, 1]$:

$$\Delta W_t = \alpha(V(x_{t+1}) - V(x_t)) \sum_{k=1}^{\infty} \lambda^{t-k} \nabla_W V(x_k) \qquad (5)$$

Este algoritmo se denomina TD(λ), siendo:

ΔW_t : la actualización de los pesos de la red en el tiempo t.

$V(x_t)$: la evaluación de la red en el estado x_t.

α : la relación de aprendizaje.

$0 \leq \lambda \leq 1$: el factor de descuento, que se encarga de pesar al error TD(0) ($V(x_{t+1})$ - $V(x_t)$) en forma exponencial para los valores más recientes.

El caso $\lambda = 1$ corresponde al apareamiento de cada conjunto de entrada con el resultado final $r = V(x_{M+1})$. Si $\lambda = 0$, estamos en el caso explícito de apareamiento de cada estado x_t con la predicción $V(x_{t+1})$. Aquí, la diferencia TD(0) se utiliza para dirigir la red hacia el espacio de pesos.

IV. Implementación de la Red Neuronal Evaluadora del Juego

En esta sección se aplica el método de diferencias temporales a la evaluación de la diagramabilidad de un sistema de tiempo real considerándola como un juego entre dos contrincantes. Debido a que el diagramador de un sistema de tiempo real tiene que juzgar la forma en que el sistema debe responder a los eventos externos, se pueden diferenciar dos acciones contrapuestas: las respuestas del diagramador y la producción de eventos externos. De esta manera se puede suponer que cada acción está desarrollada por uno de los contrincantes que intervienen en el juego. El diagramador será considerado el jugador que tiene por objeto prevenir que el sistema entre en crisis, mientras que la generación de eventos será el jugador encargado de evitar que el diagramador cumpla su misión.

De esta manera el juego queda definido de la siguiente forma:

- Jugador1: diagramador del sistema.
- Jugador2: generador de eventos externos.

- Objetivo jugador 1: diagramar las diversas tareas del sistema para evitar que entre en crisis.
- Objetivo jugador 2: producir las generaciones de las tareas para hacer que el sistema entre en crisis.

Desarrollo del juego: Dada un posición inicial del juego, los jugadores intervienen en forma alternada para cumplir sus objetivos. La primer jugada corresponde al jugador 1. Las reglas en las acciones de cada jugador están determinadas por las condiciones del sistema de tiempo real. De esta manera, cuando le toque el turno de jugar al diagramador, las posibles jugadas serán ejecutar una de las tareas generadas o dejar la ranura libre. Cuando el turno sea para el generador de eventos, éste sólo podrá generar las tareas cuyas condiciones de período y última generación lo permitan.

Una vez establecido el objetivo de cada jugador y la reglas del juego nos encontramos en condiciones de aplicar la teoría de Diferencias Temporales para implementar una NN que evalúe el desarrollo del juego.

El objetivo de la NN a implementar es que: dada un entrada que codifique el estado del juego, entregue una salida que indique la posibilidad del Jugador 1 para lograr su objetivo (ganar). Elegimos la evaluación del Jugador 1 puesto que nos interesa saber las posibilidades que éste tiene para hacer al sistema diagramable.

La salida de la NN para la jugada s la notaremos V_s. El resultado final del juego para el jugador 1 lo notaremos r.

Ante la posibilidad de movida del jugador 1, es conveniente que el diagramador elija la alternativa que tenga mejor evaluación por la NN. De esta manera, si la NN es una buena evaluadora de la situación del juego, el diagramador ejecutará la movida que tenga más oportunidad de ganar.

En el otro extremo, y con el mismo criterio, el generador de eventos deberá elegir la alternativa que tenga mínima evaluación. Si la NN determina la posibilidad de que el jugador 1 gane, el jugador 2 debe optar por la alternativa que "menos permita ganar al jugador 1".

V. Entrenamiento de la Red Neuronal

Para el entrenamiento de la NN se utilizará, con algunas modificaciones el algoritmo de aprendizaje AHC-learning [1]. La jugada s indicará una movida del jugador 1, si s es impar y una movida del jugador 2 en caso de que sea par.

El algoritmo AHC-learning produce partidas automáticamente. La forma en que crea estas partidas se basa en que las movidas del jugador 1 están dadas por la mejor evaluación de la NN mientras que las del jugador 2 están dadas por la menor evaluación. Así se logra que la red plantee el mejor juego que puede crear para cada uno de los jugadores.

El aprendizaje de la NN se logra modificando los pesos en el sentido opuesto al gradiente del error. El error para las jugadas del jugador 1 será la diferencia entre el resultado final y la evaluación producida por la NN. Para el caso del jugador 2, el error de la red será la diferencia entre el resultado final (desde el punto de vista del jugador 1) más la evaluación de la red. En consecuencia, el error en la jugada s estará dado por:

$$E_s = \begin{cases} r - V_s & \text{si s es impar} \\ r + V_s & \text{si s es par} \end{cases}$$

El error total producido en toda la partida, y considerando que la influencia del error se incrementa exponencialmente hasta el desenlace, puede expresarse de la siguiente forma:

$$E = \frac{1}{2}\sum_{i=1}^{M} \lambda^{M-i}\left(r + (-1)^i V_i\right)^2$$

Para minimizar el error total, se deben modificar los pesos en un factor α opuesto al gradiente.

$$\Delta W = \sum_{s=1}^{M} -\alpha \frac{\partial E_s}{\partial W_s} = \sum_{s=1}^{M} -\alpha \frac{\partial E_s}{\partial V_s} \cdot \frac{\partial V_s}{\partial W_s}$$

Que resulta:

$$\Delta W = \sum_{s=1}^{M} \lambda^{M-i} \cdot \left(r + \left(-1\right)^i V_i\right) \cdot \left(-1\right)^i \cdot \frac{\partial V_s}{\partial W_s}$$

Se puede expresar el error en diferencias temporales, considerando $V_{M-1} = r$, como:

$$r + \left(-1\right)^i V_i = \left(1 + \left(-1\right)^i\right) \cdot V_i + \sum_{j=1}^{M} \left(V_{j+1} - V_j\right)$$

que reemplazando en la ecuación de la variación de los pesos (5) y desarrollando las sumatorias, se puede expresar de la siguiente forma:

$$\Delta W = -\alpha \cdot \lambda^M \cdot \left\{ \sum_{i=1}^{M} \left[\lambda^{-i} \cdot \left(1 + \left(-1\right)^i\right) \cdot V_i \cdot \frac{\partial V_i}{\partial W} + \left(V_{i+1} - V_i\right) \cdot \sum_{j=1}^{i} \lambda^{-j} \cdot \left(-1\right)^j \cdot \frac{\partial W_j}{\partial W} \right] \right\}$$

Esta variación entrena la red luego de una partida. Hay que considerar que un experto se logrará luego de varias partidas ganadas y perdidas. Esto significa que la variación de los pesos debe efectuarse luego de un conjunto partidas en donde se balancee la cantidad de resultados favorables y adversos.

VI. Conclusiones

La principal dificultad presentada fue que en un sistema de tiempo real se sabe cuando se pierde (el sistema entra en crisis), pero no cuándo funciona correctamente. Esto se debe a que si el sistema funciona correctamente no hay una jugada definitiva de la partida puesto que debe diagramar un tiempo indefinido.

Para evitar este problema se propuso que el jugador 1 gana la partida, cuando alcanza un número mínimo de movidas antes de entrar en crisis. Esto hace que la NN evalúe como ganadoras las partidas que aumentan el número de movidas. Con esta metodología de entrenamiento, el proceso se inicia con un número pequeño de movidas necesarios para ganar la partida. Una vez que se efectuaron un conjunto de partidas favorables y desfavorables para

el diagramador, se modifican las matrices de pesos, se aumenta la cantidad de jugadas necesarias para declarar al diagramador vencedor y se comienza el proceso nuevamente.

Para evitar que la red entre en un ciclo negativo de aprendizaje, la elección de la jugada se realiza mediante la asignación de probabilidades a cada movida. Esta probabilidad de elección es proporcional a la evaluación que recibe dicha movida. Esta modificación mejoró apreciablemente el aprendizaje de la NN.

Debido a que existen varios parámetros de los que depende el aprendizaje, hará falta experiencias para hallar la forma en que éstos inciden en el entrenamiento de la NN. Estas experiencias se proponen para futuros trabajos.

Referencias

[1] Wiering Marco "TD Leraning of Game Evaluation Functions with Hierarchical Nueral Architetures", Master's Thesis. Department of Computer Systems - University of Amsterdam - 1995.

[2] Hush Don and Horne Bill. "Progress in Supervised Neural Networks". IEEE Signal Processing Magazine. Pág.: 8-37. January 1993.

[3] Sutton R.. "Learning to predict by the methods of temporal differences". Machine Learning. Pág.: 3 -44. 1988.

[4] Liu C. and Layland J. "Scheduling algorithms for multiprogramming in a hard real-time environment". Journal of ACM, vol 30, pp 46-61, 1973, January.

[5] Santos J. y Orozco J.."Rate Monotonic Scheduling in Hard Reat-Time Systems" Information Processing Letters. 1993.