NEUROCOMPUTACIÓN 2020 - PRÁCTICA 1

Introducción a las Redes Neuronales Artificiales

Fecha de entrega: 23:55 del 3 de marzo

1. Introducción

El sistema nervioso procesa la información de una manera eficiente y presenta una resistencia ante errores muy interesante. Por estos motivos imitar este paradigma es útil en diversos campos de la ciencia y la ingeniería. La Neurocomputación Artificial intenta proporcionar a los sistemas artificiales alguna de estas habilidades inspirándose en cómo los sistemas biológicos procesan la información [1,2].

Las redes neuronales artificiales surgieron a mediados del siglo XX [3,4], heredando dos características significativas del sistema nervioso: la capacidad de detectar patrones de comportamiento y la de aprender a partir de ejemplos sin necesidad de formalizar el conocimiento a tratar. Estas características, el aprendizaje adaptativo y la autoorganización, son dos de sus mayores ventajas.

El objetivo de esta práctica es diseñar e implementar distintos modelos neuronales artificiales “tradicionales” aplicables a la detección de distintos patrones de comportamiento.

2. Diseño de una librería para el manejo de redes neuronales

Para facilitar el desarrollo de esta práctica y las posteriores lo mas interesante es plantear el diseño de una librería que permita gestionar los valores internos de las neuronas, agruparlas en capas y realizar sinapsis. Incluid en la memoria un esquema de la librería diseñada y una pequeña explicación de su funcionamiento.

3. Neuronas de McCulloch-Pitts

Warren McCulloch y Walter Pitts diseñaron en 1943 uno de los primeros modelos de neuronas artificiales. En una neurona típica de McCulloch–Pitts la activación es binaria (su salida es 0 o 1), las conexiones excitadoras que llegan a una misma neurona tienen el mismo peso, el umbral se escoge de forma que una sola conexión inhibidora fuerce a que la salida de la neurona sea 0 en ese paso y las salidas tardan un intervalo de tiempo en llegar a las neuronas receptoras. Todas estas propiedades hacen que las neuronas de McCulloch-Pitts sean útiles para modelar fenómenos que requieran funciones lógicas y retrasos temporales.

Diseña una red McCulloch-Pitts que distinga la dirección de un estímulo presentado:

- Un estímulo se mueve a una neurona contigua superior, inferior o se queda quieto.
- Para ello la red contará con tres neuronas de entrada y dos de salida:
- Entre las tres neuronas de entrada existe continuidad (x_1 es la consecutiva de x_3)

- La salida en un tiempo t será (0 1) si entre t-1 y t-2 se produjo un mov. hacia abajo
- La salida en un tiempo t será (1 0) si entre t-1 y t-2 se produjo un mov. hacia arriba
- Ante un estímulo que permanece en la misma neurona la salida será (0 0)

Ejemplo de los estímulos en la red:

	Arriba-Abajo				Abajo-Arriba			
t	1	2	3	4	1	2	3	4
x_1	1	0	0	1	0	0	1	0
x_2	0	1	0	0	0	1	0	0
x_3	0	0	1	0	1	0	0	1

La dependencia del resultado con t-1 y t-2 para un tiempo t nos indica que la red debe recordar los resultados de esos instantes para proporcionar una salida.

El programa obtendrá los impulsos de un fichero de texto cuyo nombre será indicado como parámetro del programa. El formato del fichero será el siguiente:
1 0 0
0 1 0
0 0 1
1 0 0
La longitud del fichero es arbitraria y puede cambiar el sentido del impulso (o mantenerlo).

En un fichero “McCulloch_Pitts.out” se guardará el valor de las neuronas de salida. Ejemplo:
0 0
0 0
0 0
0 1
0 1
0 1
Nota: los tres primeros instantes de tiempo son (0 0) debido a la falta de información.
Nota: cuando la lectura del fichero de entrada ha finalizado aún deben realizarse resultados.

Memoria:

- Incluir el diseño de la red con los sesgos, conexiones y pesos utilizados.
- Discutir la validez de vuestro diseño e incluir explicaciones de ejemplos que demuestren el correcto funcionamiento interno de cada elemento que conforma el circuito.

Código:

Makefile (hacer sin importar el lenguaje de programación) con los siguientes objetivos:

- mp_help: explicación de los argumentos necesario para ejecutar el programa
- mp_exec: un ejemplo de ejecución

3. Perceptrón y Adaline

Una de las características más destacables de las redes neuronales es su capacidad de aprendizaje y generalización. El objetivo de esta práctica es el estudio de las reglas de aprendizaje que utilizan el Perceptrón y el Adaline y su aplicación a la clasificación de patrones.

- Diseñar un Perceptrón y comprobar que solucione “problema_real1.txt”
- Diseñar un Adaline y comprobar que solucione “problema_real1.txt”

NOTAS GENERALES sobre la implementación de estas redes neuronales y las del resto de prácticas:

- Deben existir tres modos de funcionamiento a la hora de leer archivos.

  - (Modo 1) El primer modo requiere de un solo archivo, se debe indicar por tanto que porción es utilizada como fichero de entrenamiento. Los datos de train se eligen aleatoriamente en cada ejecución. Con el resto de datos se realizará la validación.

  - (Modo 2) Si existe un solo archivo y el porcentaje de train es del 100% todos los datos del archivo serán utilizados en entrenamiento y test.

  - (Modo 3) Si se indican dos archivos se utilizará el primero como entrenamiento y el segundo como test. Las predicciones para el segundo archivo deberán ser guardadas en un fichero de texto **CON EL MISMO FORMATO A LOS DE LECTURA** (cabecera, entradas y salida, codificación….). La capa de salida corresponderá al resultado de predicción de la red. No cambiar el orden de lectura del segundo archivo.

- Los parámetros que afectan a la red y el aprendizaje (umbrales, tasa de aprendizaje…) deben ser incluidos como argumentos de los programas.

Para analizar y explicar los resultados obtenidos **en esta y posteriores prácticas** deberá proporcionarse la siguiente información:

- Épocas de entrenamiento, causa de parada, error final de entrenamiento, error cuadrático medio (ECM) final…
- Matriz de confusión, error de test...

Memoria:

- Descripción breve de la implementación de ambas redes neuronales.

- ¿Que influencia presentan en el resultado cada uno de los parámetros que definen el comportamiento de las dos redes? Presentad gráficas que muestren como varía el resultado modificando un solo parámetro de control (ej: umbral). Utilizad el problema_real1 y el Modo 1.

- Compara ambas redes, ¿Que sucede con el Error Cuadrático Medio (ECM) en cada una de ellas?.

- Intenta predecir ahora los problemas lógicos “nand.txt”, “nor.txt” y “xor.txt” (Modo 2). ¿Es posible al 100%? En caso de no ser posible, ¿Cual es el problema y como puede ser solucionado?

- Mediante el Modo 3 realiza predicciones para el problema_real2. No te preocupes por el porcentaje de test, el segundo fichero esta sin etiquetar. Incluye los dos fichero (prediccion_perceptron.txt y prediccion_adaline.txt) en una carpeta predicciones en la carpeta raiz de tu entrega.

Código:

Makefile (hacer sin importar el lenguaje de programación) con los siguientes objetivos:

- make: compilación de todo, incluir aunque esté vacío

- per_help: explicación de los argumentos necesario para ejecutar el programa
- per_exec: ejemplo de ejecución (problema real2 y modo1 con buen resultado)

- ada_help: explicación de los argumentos necesario para ejecutar el programa
- ada_exec: ejemplo de ejecución (problema real2 y modo1 con buen resultado)
- Más ejemplos del resto de modos y ficheros puertas lógicas

4. Entrega y memoria

- Contestar a las cuestiones planteadas en los diferentes apartados en una memoria de como máximo 10 páginas. Incluir los resultados en gráficas que muestren datos.

- Utilización del GitLab de la Escuela (https://git.eps.uam.es/). Crear un repositorio **privado** Neuro20-Pract1-ParejaXX y añadir al usuario manuel.reyes

- Realizar commits coherentes, que implementen funcionalidad específica. Las dudas, problemas y errores serán añadidos como issues. Marcarlos como resueltos, clasificarlos, indicar la solución. Estructurar de manera adecuada vuestras carpetas (src, data, predicciones…)

- Cuando se hagan consultas al email del profesor hacer referencia al issue en cuestión y al commit que implementa la función (enlaces!!). Todo lo anterior sera tenido en cuenta.

- Guardar la carpeta desde el repositorio con regularidad por si existiera algún problema con el servidor de Git.

- Por motivos de seguridad y limitaciones del Git para subir binarios (recordar las condiciones de uso) es necesario que entreguéis en moodle la práctica con el siguiente formato: Neuro20-Pract1-ParejaXX.zip

Referencias

[1] L.V. Fausett. Fundamentals of Neural Networks. Prentice-Hall Inc., 1994.
[2] S. Haykin. Neural Networks: A Comprehensive Foundation. Prentice- Hall Inc., Englewood Cliffs, NJ, 1998.
[3] D.O. Hebbs. The organization of Behavior: A Neuropsychological Theory. Wiley. New York, 1949.
[4] W.S. McCulloch and W. Pitts. A logical calculus of the ideas immanent in nervous activity. Bulletin of Mathematical Biophysics, 5(1-2): 115–133, 1943.