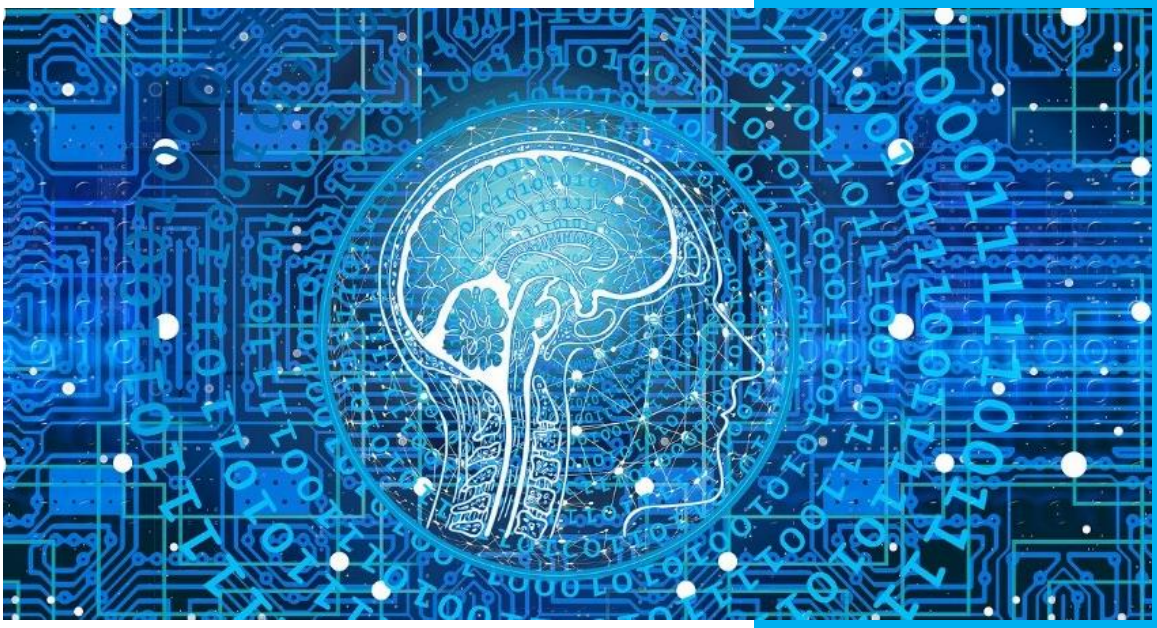
UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE ESTUDIOS SUPERIORES CUAUTITLÁN

# Manual de prácticas de Sistemas Basados en Redes Neuronales



**Departamento: Ingeniería**

**Sección: Electrónica**

**Clave de Carrera: 130**

**Asignatura: Sistemas basados en redes neuronales**

**Clave de la asignatura: 1839**

**Autor: David Tinoco Varela**

**Fecha de elaboración: Junio de 2020**

**Fecha de revisión: Junio de 2024**

# ÍNDICE

## OBJETIVOS GENERALES DE LA ASIGNATURA

- Al finalizar el curso, el alumno conocerá y aplicará los conceptos fundamentales de las redes neuronales y utilizará estos elementos para planear y diseñar una red neuronal artificial empleando sistemas digitales (*Hardware*) y programación (*Software*) para resolver una aplicación específica.

## OBJETIVOS DEL CURSO EXPERIMENTAL

- Aprender a generar interacciones entre hardware y software para crear redes neuronales alimentadas con datos de dispositivos electrónicos.
- Aprender el comportamiento de una red neuronal de forma experimental.
- Aprender a diseñar sistemas basados en redes neuronales desde una perspectiva de ingeniería.
- Utilizar bases de datos existentes para el desarrollo de nuevas tecnologías

## REGLAMENTO INTERNO DEL LABORATORIO

El presente reglamento de la sección electrónica tiene por objetivo establecer los lineamientos para el uso y seguridad de laboratorios, condiciones de operación y evaluación, que deberán de conocer y aplicar, estudiantes y profesores en sus cuatro áreas: comunicaciones, control, sistemas analógicos y sistemas digitales.

1. Queda estrictamente prohibido, al interior de los laboratorios

   a) Correr, jugar, gritar o hacer cualquier otra clase de desorden.
   b) Dejar basura en las mesas de trabajo y/o pisos.
   c) Fumar, consumir alimentos y/o bebidas.
   d) Realizar o responder llamadas telefónicas y/o el envío de cualquier tipo de mensajería.
   e) La presencia de personas ajenas en los horarios de laboratorio.
   f) Dejar los bancos en desorden y/o sobre las mesas.
   g) Mover equipos o quitar accesorios de una mesa de trabajo.
   h) Usar o manipular el equipo sin la autorización del profesor.
   i) Rayar y/o sentarse en las mesas del laboratorio.
   j) Energizar algún circuito sin antes verificar que las conexiones sean las correctas (polaridad de las fuentes de voltaje, multímetros, etc.).
   k) Hacer cambios en las conexiones o desconectar el equipo estando energizado.
   l) Hacer trabajos pesados (taladrar, martillar, etc.) en las mesas de trabajo.
   m) Instalar software y/o guardar información en los equipos de cómputo de los laboratorios.
   n) El uso de cualquier aparato o dispositivo electrónico ajeno al propósito para la realización de la práctica.

o) Impartir clases teóricas, su uso es exclusivo para las sesiones de laboratorio.

2. Es responsabilidad del profesor y de los estudiantes revisar las condiciones del equipo e instalaciones del laboratorio al inicio de cada práctica (encendido, dañado, sin funcionar, maltratado, etc.). El profesor deberá generar el reporte de fallas de equipo o de cualquier anomalía y entregarlo al responsable de laboratorio o al jefe de sección.

3. Los profesores deberán de cumplir con las actividades y tiempos indicados en el "cronograma de actividades de laboratorio".

4. Es requisito indispensable para la realización de las prácticas que el estudiante:

a) Descargue el manual completo y actualizado al semestre en curso, el cual podrá obtener en (http://olimpia.cuautitlan2.unam.mx/pagina_ingenieria/)
b) Presente su circuito armado en la tableta de conexiones para poder realizar la práctica (cuando aplique), de no ser así, tendrá una evaluación de cero en la sesión correspondiente.
c) Realizar las actividades previas y entregarlas antes del inicio de la sesión de práctica, de no ser así, tendrá una evaluación de cero en la sesión correspondiente.

5. Estudiante que no asista a la sesión de práctica de laboratorio será evaluado con cero.

6. La evaluación de cada sesión debe realizarse con base en los criterios de evaluación incluidos en los manuales de prácticas de laboratorio y no podrán ser modificados. En caso contrario, el estudiante deberá reportarlo al jefe de sección.

7. La evaluación final del estudiante en los laboratorios será con base en lo siguiente:

a) (Aprobado) Cuando el promedio total de todas las prácticas de laboratorio sea mayor o igual a 6 siempre y cuando tengan el 90% de asistencia y el 80% de prácticas acreditadas con base en los criterios de evaluación.
b) (No Aprobado) No cumplió con los requisitos mínimos establecidos en el punto anterior.
c) (No Presentó) Cuando no asistió a ninguna sesión de laboratorio o que no haya entregado actividades previas o reporte alguno.

8. Profesores que requieran hacer uso de las instalaciones de laboratorio para realizar trabajos o proyectos, es requisito indispensable que las soliciten por escrito al jefe de sección. Siempre y cuando no interfiera con los horarios de los laboratorios.

9. Estudiantes que requieran realizar trabajos o proyectos en las instalaciones de los laboratorios, es requisito indispensable que esté presente el profesor responsable del trabajo o proyecto. En caso contrario no podrán hacer uso de las instalaciones.

10. Correo electrónico del buzón para quejas y sugerencias para cualquier asunto relacionado con los laboratorios (seccion_electronica@cuautitlan.unam.mx).

11. El incumplimiento a estas disposiciones faculta al profesor para que instruya la salida del infractor y en caso de resistencia, la suspensión de la práctica.

12. A los usuarios que, por su negligencia o descuido inexcusable, cause daños al laboratorio, materiales o equipo deberá cubrir los gastos que se generen con motivo de la reparación o reposición, indicándose en el reporte de fallas correspondiente.

13. Los usuarios de laboratorio que sean sorprendidos haciendo uso indebido de equipos, materiales, instalaciones y demás implementos, serán sancionados conforme a la legislación universitaria que le corresponda, según la gravedad de la falta cometida.

14. Los casos no previstos en el presente reglamento serán resueltos por el Jefe de Sección, de acuerdo con los lineamientos generales para el uso de los laboratorios en la Universidad Nacional Autónoma de México.

## CRITERIOS DE EVALUACIÓN

| | | |
|---|---|---|
| C1 | Actividades previas indicadas en el manual de practicas | 10 % |
| C2 | Análisis e interpretación de resultados | 20% |
| C3 | Toma de lecturas correctas | 30% |
| C4 | Reporte entregado con todos los puntos indicados | 40% |

## INSTALACIÓN DE MATLAB

Para el desarrollo de las prácticas de este manual es necesario la descarga y uso de Matlab, el cual puede ser obtenido e instalado desde https://www.software.unam.mx/producto/matlab/, recordando que esta es una licencia libre para toda la comunidad UNAM. Para descargar el software es necesario tener una cuenta institucional de correo electrónico, lo que cubrirá la versión de trabajo *offline* y la versión *online*.

Los requisitos mínimos de instalación para la última versión (2022) son: Sistema operativo Windows 10 o mayor (Windows 7 sigue siendo soportado en versiones anteriores de Matlab); Procesador AMD o Intel de x86-64; Memoria RAM de 4 GB y recomendada 8 GB; Y una capacidad de almacenamiento de 5 a 8 GB para una instalación típica, considerando 31.5 GB para una instalación completa (Lo cual no es necesario para estas prácticas).

## INTRODUCCIÓN

Las redes neuronales artificiales, pertenecen al área de desarrollo de la inteligencia artificial, estas han tomado gran relevancia en los últimos años, debido a la cantidad de problemas que pueden resolver.

Las redes neuronales, son utilizadas cuando se tiene un problema que procesos matemáticos estándar no pueden modelar (al menos no con los conocimientos existentes). Algunas de las aplicaciones más comunes pueden encontrarse en el reconocimiento de imágenes, tales como el reconocimiento de rostros y facciones, o el reconocimiento de objetos; la clasificación de

elementos, tales como la clasificación de frutos podridos y frutos sanos, la clasificación de tipos de flores, la clasificación de personalidades; la predicción de eventos, tales como predicción del clima, predicción de movimientos económicos, predicción de comportamiento de consumo; entre otras aplicaciones.

Aunque existen aplicaciones particulares, como las ya mencionadas, la verdad es que estos esquemas se han ido posicionando prácticamente en cualquier área de conocimiento, colocándose como centro de cálculo de sistemas electrónicos, químicos, financieros, biológicos, etc. Esto debido a sus características primordiales.

Algunas de las características más importantes de estos esquemas es que pueden ejecutarse inherentemente de forma paralela, es decir, pueden realizar cálculos de forma más rápida que otro tipo de esquemas computacionales. Estos esquemas presentan gran tolerancia a fallos, por lo que pueden ser alimentados con información falsa, y aun así lograran llegar al resultado correcto, obviamente cuando la información falsa no eclipse a la información verdadera. Pueden adaptarse a una gran cantidad de problemas, sin modificar su estructura base, es decir, basta con que ellas sean alimentadas con los datos de una determinada área o problemática para que puedan llegar a la solución esperada.

En ingeniería, estos sistemas han cobrado mucha relevancia, ya que son las encargadas de procesar los datos que diferentes dispositivos electrónicos adquieren, y de esta manera lograr que los dispositivos se puedan comportar como nodos activos en esquemas tecnológicos tales como el internet de las cosas o la industria 4.0.

**Agradecimiento.**

Manual apoyado con los proyectos PAPIME PE100221.

## BIBLIOGRAFÍA

1. Mandic, D. P., & Chambers, J. (2001). *Recurrent neural networks for prediction: learning algorithms, architectures and stability*. John Wiley & Sons, Inc.
2. Luo, F. L., & Unbehauen, R. (1998). *Applied neural networks for signal processing*. Cambridge university press.
3. Simon Haykin (2008). *Neural Networks and Learning Machines*, 3d Edition. Pearson.

**NOTA:** La imagen que se presenta en la portada de este manual de prácticas, fue obtenida de https://pixabay.com/es/illustrations/web-red-programaci%C3%B3n-3706562/ Agradecemos al autor de tal imagen.

# Laboratorio de Sistemas Basados en Redes Neuronales

## Práctica 1

## Introducción: Matlab

### TEMAS

1.8. Aplicaciones.

2.6. Modelos Electrónicos.

### OBJETIVOS

Al término de esta práctica el alumno podrá:

- Interconectar dispositivos electrónicos con la herramienta de trabajo Matlab.
- Generar programas en Matlab que ingresen datos desde un sensor.
- Utilizar estructuras tales como vectores y matrices.

### INTRODUCCIÓN

Actualmente, existe una gran cantidad de herramientas computacionales que permiten realizar procesos relacionados a inteligencia artificial, en general y redes neuronales, en particular. Se pueden encontrar diferentes herramientas libres, de cobro, especializadas, no especializadas, etc.

Para el caso de este laboratorio, se ha optado por utilizar la herramienta Matlab, si bien es cierto que Matlab no es una herramienta especializada en inteligencia artificial, también es cierto que en los últimos años ha generado una gran cantidad de paqueterías enfocadas a las diferentes ramas de esta área de conocimiento, por lo que es posible encontrar prácticamente cualquier concepto relacionado al área en este software. Por otro lado, Matlab es una herramienta comercial que implica gasto (actualmente la comunidad UNAM cuenta con cuentas gratuitas), sin embargo, la ventaja que se considera que pueden tener los alumnos, es que es una herramienta utilizada ampliamente no solo en el sector académico, sino también en el sector profesional y comercial, es decir, no es una herramienta que se utilice sólo en las escuelas y posteriormente puede ser de utilidad del alumnado en su vida laboral.

En esta práctica, se busca que los alumnos tengan un acercamiento a esta herramienta, y a través de ella gestionar diferentes tipos de datos por medio de elementos electrónicos de entrada y salida. Esta práctica se ha considerado, con la intención de que los alumnos verifiquen el control de sus herramientas y que ellas están trabajando adecuadamente, para posteriormente entrar en las aplicaciones y en la adquisición y procesamiento de datos de mayor tamaño y relevancia.

Adicionalmente, se dará gran relevancia al uso de estructuras computacionales tales como vectores y matrices, ya que serán estructuras que posteriormente serán ampliamente utilizadas.

## ACTIVIDADES PREVIAS A LA PRÁCTICA

1. El alumno realizará la lectura de la práctica.
2. El alumno descargará el script utilizado en esta práctica de http://virtual.cuautitlan.unam.mx/intar/?page_id=793 e interpretará cada uno de los elementos y comandos que componen el script.
3. El alumno debe de haber adquirido conocimiento de las herramientas a utilizar antes de la práctica.

   **NOTA:** Aunque el script ya tiene la configuración correcta en código ASCII, puede ocurrir que cuando se copie la información entre diferentes plataformas, esta codificación se vea comprometida, uno de los errores más comunes que se pueden tener es el cambio de una comilla simple en un acento, lo que obviamente desembocará en errores en la lectura del script, basta que el alumno verifique estas posibles malinterpretaciones entre formatos de cada plataforma utilizada.

## MATERIAL Y EQUIPO

1. Una computadora con Matlab (https://www.software.unam.mx/producto/matlab/ ) instalado y con las paqueterías de "MATLAB Support Package for Arduino Hardware" y "Deep learning" y "Neural network toolbox" cargadas en el software.
2. Una tarjeta de desarrollo Arduino (Preferentemente la versión UNO).
3. Un switch normalmente abierto.
4. Un LED.
5. Una resistencia de 1 K.
6. Un potenciómetro.
7. Un LDR.
8. Cables de conexión.
9. Tableta Protoboard.

## PROCEDIMIENTO EXPERIMENTAL

En este proceso, se realizarán pequeñas prácticas para verificar que se tiene el control adecuado de Matlab y de su comunicación con la Tarjeta Arduino. Se hace hincapié en que dependiendo de la tarjeta Arduino utilizada, original o genérica, la computadora podrá identificar diferentes tipos de puertos, siendo los puertos COM 1 y COM 3 los más comunes.

1. Se armará el sistema tal cual se puede observar en la figura 1. Este ejercicio es el más simple y sólo está destinado a verificar la correcta comunicación entre hardware y software.

2. El alumno cargara el programa 1 de la dirección http://virtual.cuautitlan.unam.mx/intar/?page_id=793 en Matlab, y lo ejecutará.

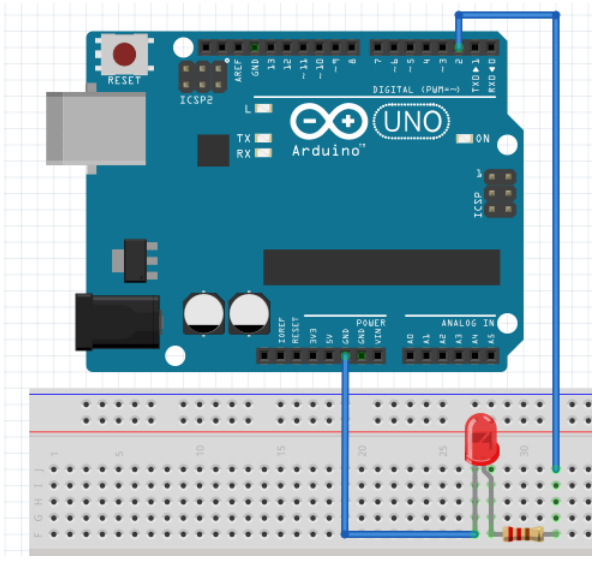



Fig. 1.- Primer sistema para probar la conexión entre Matlab y Arduino.

3. Es posible observar que este sistema sólo prende y apaga un LED, por lo que el alumno modificará este primer script para que esa secuencia se realice 10 veces, utilizar el ciclo "for" para esta actividad, mostrar el correcto funcionamiento a su profesor.

4. Armar el sistema mostrado en la figura 2.

5. El alumno cargara el programa 2 de la dirección http://virtual.cuautitlan.unam.mx/intar/?page_id=793 en Matlab, y lo ejecutará.

6. Girar el potenciómetro de un lado a otro lentamente y observar que sucede con el LED. Verificar el comportamiento con lo definido en el script.

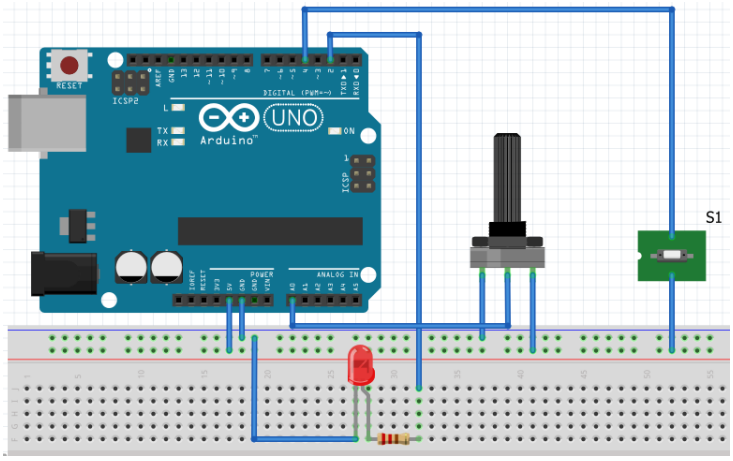



Fig. 2.- Segundo ejemplo de aplicación, control de un LED por medio de un potenciómetro.

7. Sustituyendo el potenciómetro con un LDR, el alumno debe realizar un detector de obscuridad. Mostrar al profesor el correcto funcionamiento.

8. Una vez concluida esta parte, es necesario iniciar con el comportamiento de las estructuras tales como vectores y matrices. Para tal fin, el alumno debe armar el circuito como se muestra en la figura 3.

9. El alumno cargara el script 3 de la dirección http://virtual.cuautitlan.unam.mx/intar/?page_id=793 en Matlab, y lo ejecutará.

10. Una vez en ejecución hay que girar lentamente el potenciómetro, haciendo cambios a razón de un segundo. Este pequeño sistema generará el llenado de un vector. Es importante mencionar que el alumno definirá la cantidad de registros que tenga el vector.

11. Ver que los valores del vector resultante, correspondan con los datos que el alumno ingresó por medio del potenciómetro.

12. El alumno debe diseñar un script que llene una matriz, con elementos provenientes del potenciómetro. Mostrar el correcto funcionamiento al profesor.

13. Generar un script que ingrese dos vectores por medio del potenciómetro, y estos los convierta posteriormente en una sola matriz.

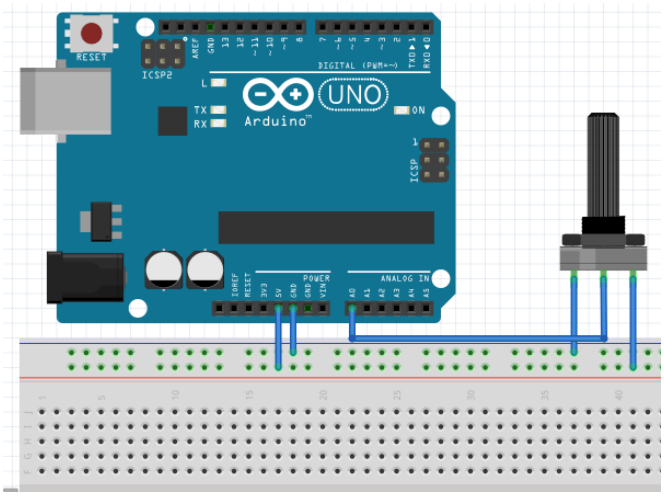



Fig. 3.- Control de ingreso de datos a estructuras computacionales.

## CUESTIONARIO

1.- Escribir un script que ingrese un valor escalar utilizando un potenciómetro, también, que genere una matriz con valores aleatorios ingresados con el mismo potenciómetro y, posteriormente, obtenga la multiplicación del escalar por la matriz.

2.- Escribir un script que ingrese dos vectores utilizando un potenciómetro, y posteriormente obtenga la multiplicación de tales vectores.

3.- Escribir un script que ingrese un vector utilizando un potenciómetro, y que muestre el recorrido de tal vector, encendiendo un LED si el valor almacenado es mayor a 2.5.

## CONCLUSIONES

Escribe tus conclusiones de la práctica.

## BIBLIOGRAFÍA

# Laboratorio de Sistemas Basados en Redes Neuronales

## Práctica 2

## **Perceptrón**

### TEMAS

2. RED NEURONAL PERCEPTRÓN.

2.2. Estructura de la Red.

2.3. Regla de Aprendizaje.

### OBJETIVOS

Al término de esta práctica el alumno podrá:

- Identificar la forma de aprender por medio de una neurona artificial.
- Apreciará el concepto de aprendizaje supervisado.
- Entrenará una neurona para que aprenda los esquemas básicos de lógica digital.

### INTRODUCCIÓN

El perceptrón desarrollado por Rosenblatt en 1958, es la neurona artificial más simple. Esta puede comportarse como un discriminador lineal, es decir, permite resolver problemas de clasificación o de aprendizaje cuando existen elementos a clasificar linealmente independientes.

El perceptrón realiza aprendizaje de forma supervisada, es decir, el usuario necesita darle una muestra de los ejemplos que desea que la neurona aprenda y los resultados esperados de cada uno de tales ejemplos, de esta manera la neurona ira verificando si su configuración es compatible con los resultados esperados o *target*.

El perceptrón basa su aprendizaje en la modificación de los pesos sinápticos que se encuentran ligados a cada una de las entradas que tal neurona presenta, estos pesos se irán modificando y adaptando al problema que le sea planteado, hasta llegar al resultado esperado.

La forma básica de un perceptrón puede visualizarse en la figura 1, donde se muestra un perceptrón de *n* entradas.

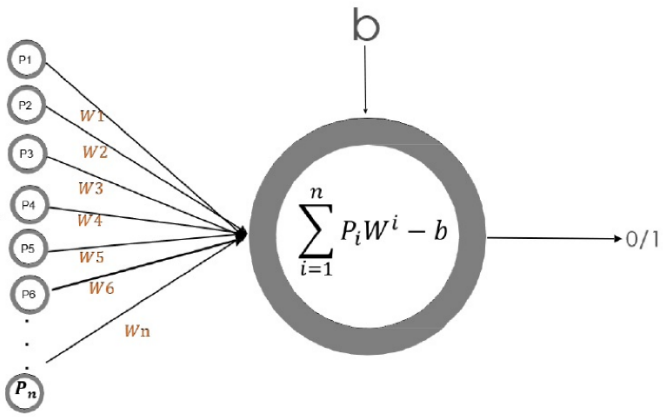



Fig 1.- Configuración y representación básica de un perceptrón.

## ACTIVIDADES PREVIAS A LA PRÁCTICA

1. El alumno realizará la lectura de la práctica.
2. El alumno descargará el script utilizado en esta práctica de http://virtual.cuautitlan.unam.mx/intar/?page_id=793 e interpretará cada uno de los elementos y comandos que componen el script.
3. El alumno debe de haber adquirido conocimiento de las herramientas a utilizar antes de la práctica.

## MATERIAL Y EQUIPO

1. Una computadora con Matlab (https://www.software.unam.mx/producto/matlab/ ) instalado y con las paqueterías de "MATLAB Support Package for Arduino Hardware", "Deep learning" y "Neural network toolbox" cargadas en el software.
2. Una tarjeta de desarrollo Arduino (Preferentemente la versión UNO).
3. Un switch normalmente abierto.
4. Un LED.
5. Una resistencia de 1 K.
6. Cables de conexión.
7. Tableta Protoboard.

## PROCEDIMIENTO EXPERIMENTAL

1. El alumno armara el sistema que es muestra en la figura 2.
2. El alumno descargará el script que se encuentra en la dirección http://virtual.cuautitlan.unam.mx/intar/?page_id=793 y lo ejecutara en Matlab.
3. Con el script, se realizará el entrenamiento de la neurona para que aprenda el comportamiento de una compuerta AND, comportamiento que puede verse en la tabla 1.
4. En la ejecución del script, primero se solicitará el número de muestras, en este caso son 4 muestras, es decir, cada uno de los casos que compone la compuerta lógica.

5. Acto seguido, se solicitará que se ingrese el vector correspondiente a la entrada 1, para este caso, la entrada se realizará por medio del switch. Considerando un switch NA (Normalmente Abierto), sin apretar el botón se considera un 0 y apretando el botón se considera un 1; para indicar que se va a ingresar el siguiente valor es necesario dar un Enter. **Es importante mencionar que antes de dar el Enter, ya se debe de tener presionado o no el botón del switch.**

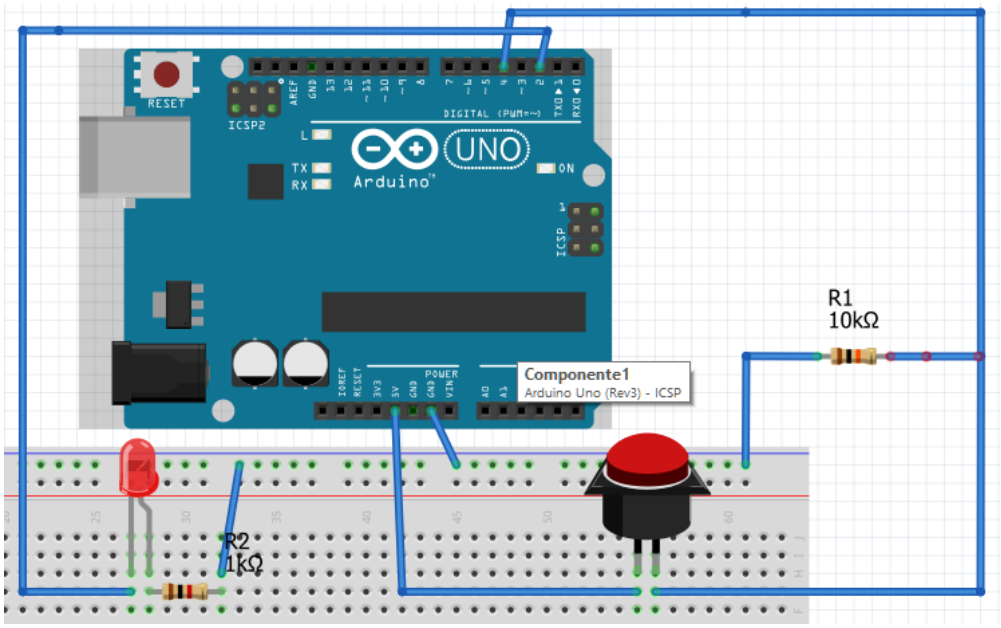



Fig. 2.- Esquema de armado.

Tabla 1. Comportamiento de una compuerta lógica AND

| Entrada 1 | Entrada 2 | Salida Esperada |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

6. Después solicitará el ingreso del segundo vector, en este caso hay que meter en orden los valores de la tabla 1 de la columna “Entrada 2”.

7. Por último solicitará que ingresemos el target, en este caso es el vector correspondiente a “Salida esperada” de la tabla 1.

8. Una vez ingresados los datos, el script comenzará a entrenar al perceptrón, por lo cual ahora se pedirá si se quiere verificar algún vector de entrada. Estos vectores de entrada, son las combinaciones [Entrada 1; Entrada 2], tal como se ven la tabla 1.

9. El alumno debe comprobar las entradas de la tabla y el LED indicara la salida obtenida. Si la salida obtenida no es la misma que la deseada, es necesario realizar otro entrenamiento, o modificando la línea “net.trainParam.epochs=5;” del script, colocando más épocas.
10. Ya verificado el correcto aprendizaje de la compuerta AND, el alumno tiene que repetir los paso 3 al 9 pero entrenando con las tablas correspondientes a las compuertas OR, NAND y NOR.
11. Si es necesario agregar más épocas para el correcto entrenamiento, el alumno debe realizarlo.

## CUESTIONARIO

1. ¿Describa por qué no es necesario modificar el programa para realizar el entrenamiento de diferentes problemas planteados, en este caso las tablas de verdad de las compuertas lógicas?
2. ¿Qué sucede si la neurona tiene pocas épocas de entrenamiento?
3. ¿En cuál de los casos planteados, le costó más trabajo a la neurona entrenarse?
4. ¿Cómo se puede saber cuáles son los valores de los pesos sinápticos que la neurona calculó para resolver los problemas planteados?
5. Defina los pesos sinápticos de al menos uno de los casos planteados.

## CONCLUSIONES

Escribe tus conclusiones de la práctica.

## BIBLIOGRAFÍA

# Laboratorio de Sistemas Basados en Redes Neuronales

## Práctica 3

## Errores de entrenamiento

### TEMAS

2. RED NEURONAL PERCEPTRÓN

2.4. Limitación del Perceptrón.

### OBJETIVOS

Al término de esta práctica el alumno podrá:

- Identificar los errores en el momento de ingresar datos a un clasificador y como estos afectan el correcto funcionamiento del sistema.
- Verificar en qué casos no se puede entrenar una sola neurona.

### INTRODUCCIÓN

Uno de los principales problemas cuando se trata de entrenar un discriminador lineal, en este caso una neurona, es la forma en la que se ingresan los datos de entrenamiento. Un buen conjunto de datos de entrenamiento, permitirá que el discriminador lineal pueda responder de una forma apropiada ante el problema planteado, sin embargo, un mal conjunto de datos puede hacer que el discriminador sea inexacto en el momento de la clasificación, dando como resultado falsos positivos o falsos negativos.

Escoger un correcto conjunto de datos, puede ser una tarea difícil de llevar a cabo, ya que para un problema que no podemos definir por medio de una función matemática tampoco podemos definir exactamente qué puntos corresponden o no a tal función, por lo tanto definir los datos correctos de entrenamiento no es una tarea sencilla.

Para este tipo de situaciones, una solución empírica, es seleccionar los casos existentes relacionados a tal problemática y meterlos como datos de entrenamiento del clasificador, esperando que sea precisamente este, el que encuentre el patrón adecuado de clasificación.

En este sentido, el objetivo de esta práctica es definir algunos de los casos en los cuales el entrenamiento de una neurona falla y que hacer para solucionarlo.

### ACTIVIDADES PREVIAS A LA PRÁCTICA

1. El alumno realizará la lectura de la práctica.
2. El alumno interpretará cada uno de los elementos y comandos que componen esta práctica.

3. El alumno debe de haber adquirido conocimiento de las herramientas a utilizar antes de la práctica.

## MATERIAL Y EQUIPO

1. Una computadora con Matlab (https://www.software.unam.mx/producto/matlab/ ) instalado y con las paqueterías de "MATLAB Support Package for Arduino Hardware" y "Deep learning" cargadas en el software.

## PROCEDIMIENTO EXPERIMENTAL

Esta sección está enfocada a entrenar correctamente el comportamiento de una pendiente, primero se graficará, para posteriormente obtener el correcto entrenamiento de una neurona y así se adapte al comportamiento de tal función. Asumimos que toda el área debajo de la pendiente es 0 y encima de la pendiente es 1. Por lo tanto, las coordenadas definidas se posicionarán en 0 o en 1.

1. Por medio de Matlab, el alumno debe graficar la función $f(x) = 2x - 1$. Guarde esta imagen para comparaciones posteriores.

2. Primero hay que definir una neurona de dos entradas, por medio de

```
perce=newp([-2 2; -2 2],1);
```

3. Definimos las características de la neurona, indicando que los pesos sinápticos de inicio serán valores aleatorios y que tendremos un determinado número de épocas, todo por medio de

```
perce.inputweights{1,1}.initFcn='rands';

perce.trainParam.epochs=100;
```

4. Se inicia la neurona, y ya queda lista para ser entrenada y ejecutada.

```
perce=init(perce);
```

5. EL alumno debe verificar que los pesos iniciales son aleatorios, por medio de

```
perce.iw{1}
```

6. El alumno debe de entrenar la neurona con los valores de la tabla 1. Considerando que los valores de la tabla son coordenadas correctas dentro del plan de la pendiente trazada anteriormente.

Tabla 1.- Primer tabla de entrenamiento.

| Entrada 1 | Entrada 2 | Target |
|---|---|---|
| 1.5 | .5 | 0 |
| 1 | 1.5 | 1 |
| 1 | 3 | 1 |
| 1 | 4 | 1 |

7. Para ingresar los vectores, vamos a utilizar las siguientes variables de entrada y salida:

```
adIn=[ 1.5 1 1 1;.5 1.5 3 4];

adOut=[0 1 1 1];
```

8. Iniciamos y verificamos los pesos sinápticos de inicio, por medio de

```
perce=init(perce);

perce.iw{1}
```

9. El alumno debe de entrenar la neurona por medio de

```
perce=train(perce, adIn, adOut);
```

10. Revisa los pesos después del entrenamiento

```
perce.iw{1}
```

11. Ejecuta la neurona ya entrenada, con la matriz de entrada, y el resultado debe de ser igual al vector Target

```
simulado=sim(perce,adIn)
```

12. Por medio de las siguientes líneas gráfica los elementos de entrada y el comportamiento de los pesos sinápticos y el bias.

```
figure(1)

plotpv(adIn, adOut);

plotpc(perce.IW{1}, perce.b{1});
```

13. Compara con la gráfica obtenida al inicio de la práctica. Mostrar al profesor. Los datos de entrada son datos reales que corresponden al comportamiento real de la pendiente previamente dibujada, sin embargo, ¿La pendiente de los pesos sinápticos es igual a la pendiente original? ¿Cuál sería el valor de “b” en esta gráfica?

14. Repitiendo los pasos anteriores, el alumno debe de agregar más datos a la matriz de entrada y al vector de salida, agregue los datos de la tabla 2.

Tabla 2.- Datos para agregar a los elementos de entrenamiento de la neurona.

| Entrada 1 | Entrada 2 | Target |
|---|---|---|
| 0 | 1 | 1 |
| 0 | 2 | 1 |
| 1 | -1 | 0 |

15. Con un total de 7 filas, el alumno debe nuevamente hacer el proceso de entrenamiento y verificación de pesos sinápticos de la neurona.

16. EL alumno debe simular la neurona entrenada, con esto nuevos elementos de entrada y salida.

17. Dibuje nuevamente la gráfica de comportamiento de las entradas, las salidas y los pesos sinápticos.

18. ¿Se parece a la gráfica realizada en el paso 1?, ¿Qué valor de "b" tiene la pendiente generada con el entrenamiento?

19. ¿Por qué a pesar de que estamos entrenando con coordenadas que representan a la función $f(x) = 2x - 1$ de manera correcta, no podemos igualarla si se supone que la neurona debe poder responder a este problema sin mayor complicación?

20. El alumno debe proponer, basándose en la imagen de la pendiente obtenida en el punto 1, una matriz de entrada y un vector de salida que considere resolverá el problema adecuadamente. Mostrar al profesor los valores asignados, el entrenamiento, verificación de datos y gráfica de salida de su propuesta.

21. Entrene nuevamente la neurona utilizando la tabla 3.

Tabla 3.- Primera tabla de entrenamiento.

| Entrada 1 | Entrada 2 | Target |
|---|---|---|
| -1 | -4 | 0 |
| -2 | -4 | 1 |
| -1 | -3.5 | 0 |
| -1 | -2 | 1 |
| 1 | 0 | 0 |
| 1 | 1.5 | 1 |
| 2 | 2 | 0 |
| 2 | 4 | 1 |

22. Hay que hacer la mención que para este fin se tiene que redefinir la declaración de la neurona por

```
perce=newp([-4 4; -4 4],1);
```

23. Vuelva a realizar todo el proceso de ingreso de datos, entrenamiento, verificación del vector de entrenamiento, verificación de pesos y generación de la gráfica representativa.

24. ¿Qué observa con la gráfica de salida?, ¿Qué valor tiene el "b" en esta función?, ¿Se acoplo a la pendiente original?

## CUESTIONARIO

1. ¿Por qué no se adaptaba la neurona al comportamiento de la pendiente original?
2. ¿Qué se tuvo que realizar para que la neurona se adaptara al comportamiento de la pendiente original?
3. El alumno debe realizar un script que realice todo lo hecho en la práctica. NO tendrá entradas ni salidas pre definidas, el usuario las debe ingresar. Y el usuario debe poder decidir si quiere evaluar diferentes opciones de entrada y ver su comportamiento.

4. EL alumno debe ingresar una tabla similar a la tabla 3 en su script diseñado, los valores deben de ser aleatorios y ver qué sucede con el entrenamiento. ¿La pendiente dibujada clasifica adecuadamente?, ¿Por qué?

## CONCLUSIONES

Escribe tus conclusiones de la práctica.

## BIBLIOGRAFÍA

# Laboratorio de Sistemas Basados en Redes Neuronales

## Práctica 4

## ADALINE

### TEMAS

3. RED NEURONAL ADALINE
   3.1. Antecedentes.
   3.2. Estructura de la Red.
   3.4. Modelos Electrónicos.

### OBJETIVOS

Al término de esta práctica el alumno podrá:

- Identificar una neurona lineal adaptativa.
- Definir la diferencia entre una neurona Perceptrón y una neurona adaptativa.
- Visualizar la adaptación de la neurona a un problema dado, a través de los cambios en los pesos sinápticos de la neurona.

### INTRODUCCIÓN

Una neurona lineal adaptativa, es un modelo de una neurona simple. Este modelo fue desarrollado por Bernard Widrow y su estudiante Marcian Hoff en 1960, en la universidad de Stanford.

Las neuronas adaptativas, son similares al perceptrón, sin embargo, a diferencia del perceptrón que posee una función de activación escalón, la neurona adaptativa tiene su núcleo en funciones lineales.

Este tipo de neuronas, al igual que el perceptrón, sólo pueden resolver problemas que sean linealmente separables, teniendo problemas en la resolución de otro tipo de funciones más complejas.

El perceptrón registra en su error solamente si se ha equivocado o no en su salida, sin embargo, la neurona adaptable identifica cuanto se ha equivocado en el resultado de salida, ajustándose de una forma más eficiente a la resolución del problema.

Este tipo de neuronas pueden ser utilizadas para la predicción de señales, asociación de patrones (que cumplan con la condición ya mencionada), como filtros de ruido, entre otras aplicaciones.

El modelo básico de esta neurona, puede verse en la figura 1. Esta figura puede ser comparada con el esquema del perceptrón mostrado en la figura 1 de la práctica 2.

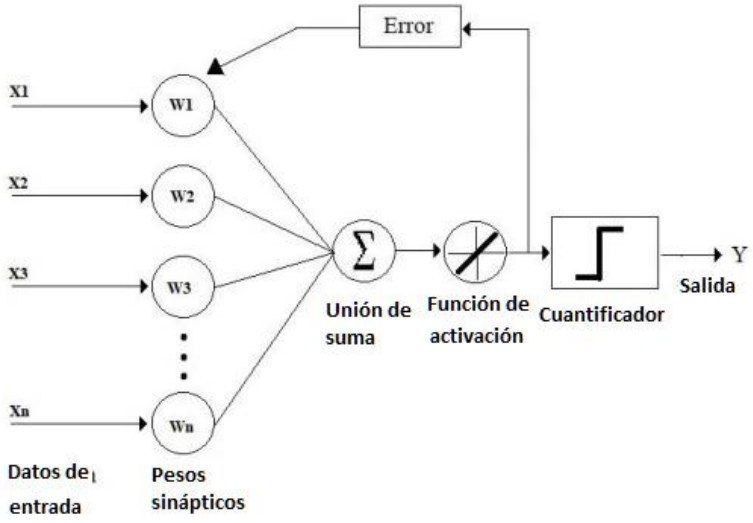



Fig. 1.- Esquema de funcionamiento de la neurona lineal adaptativa.

## ACTIVIDADES PREVIAS A LA PRÁCTICA

1. El alumno realizará la lectura de la práctica.
2. El alumno descargará el script utilizado en esta práctica de http://virtual.cuautitlan.unam.mx/intar/?page_id=793 e interpretará cada uno de los elementos y comandos que componen el script.
3. El alumno debe de haber adquirido conocimiento de las herramientas a utilizar antes de la práctica. Considerando principalmente el uso de comandos para graficar datos decimales.

## MATERIAL Y EQUIPO

1. Una computadora con Matlab (https://www.software.unam.mx/producto/matlab/ ) instalado y con las paqueterías de “MATLAB Support Package for Arduino Hardware” y “Deep learning” cargadas en el software.
2. Una Tarjeta de desarrollo Arduino (Preferentemente la versión UNO).
3. Un potenciómetro de 50 KΩ.
4. Cables de conexión.
5. Tableta Protoboard.

## PROCEDIMIENTO EXPERIMENTAL

En estos experimentos, primero que nada se mostrara el uso de la neuronal lineal adaptativa, para después ingresar datos a través de elementos electrónicos físicos y de esta forma visualizar su funcionamiento.

1. En esta primera parte de la práctica, vamos a definir una neurona adaptativa, esto lo hacemos por medio del comando, que escribiremos en la ventana de comandos de Matlab.

```
ada=linearlayer
```

2. Por medio del siguiente comando, también escrito en Matlab, se pueden definir las características de la neurona, en este caso, le estamos indicando que la neurona "ada" tendrá dos entradas y una salida.

```
ada=configure(ada, [0;0],0);
```

3. Ya habiendo definido sus características, se define que los pesos sinápticos iniciales de la neurona serán aleatorios, por medio del comando

```
ada.inputweights{1,1}.initFcn='rands'
```

4. Iniciamos la neurona, por medio del comando

```
ada=init(ada)
```

5. Verificamos que los pesos sinápticos son aleatorios con

```
ada.iw{1,1}
```

6. Repetimos los pasos 4 y 5 las veces necesarias para comprobar la activación de los pesos de forma aleatoria.

7. En este momento, se procede a la realización de un codificador binario decimal. Al igual que en el perceptrón, es necesario ingresar una matriz de datos, o características y un vector de Target. Para lo que el alumno va a ingresar los siguientes valores de adIn y adOut, que representan la matriz de entrada y el vector de salida, respectivamente. Esta representación, es equivalente a la mostrada en la tabla 1.

```
adIn=[0 0 0 1 1 1 1;0 1 1 0 0 1 1;1 0 1 0 1 0 1]
adOut=[1 2 3 4 5 6 7]
```

Tabla 1.- Representación de entrada-salida de un codificador.

| Entrada 1 | Entrada 2 | Entrada 3 | Target |
|---|---|---|---|
| 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 2 |
| 0 | 1 | 1 | 3 |
| 1 | 0 | 0 | 4 |
| 1 | 0 | 1 | 5 |
| 1 | 1 | 0 | 6 |
| 1 | 1 | 1 | 7 |

8. Para entrenar la neurona Adaline, basta ingresar el siguiente comando

```
ada=newlind(adIn,adOut);
```

9. En este momento, la neurona se ha adaptado a los requisitos de salida, para visualizar la adaptación de la neurona, es necesario ver los pesos sinápticos a los cuales la neurona llegó, por medio de:

ada.iw{1,1}

10. El alumno debe describir, ¿Qué representan esos pesos sinápticos?, ¿Qué representan los valores obtenidos?

11. Pasamos a la segunda parte de la práctica. Para esto hay que realizar la conexión que se muestra en la figura 2, conectando la terminal USB a una computadora con Matlab.

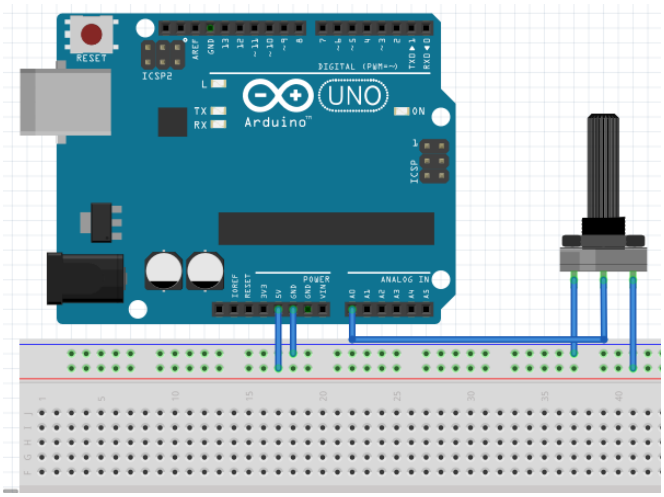



Fig. 2.- Conexión básica para la práctica.

12. El alumno descargará el script que se encuentra en la dirección http://virtual.cuautitlan.unam.mx/intar/?page_id=793 y lo ejecutara en Matlab.
13. Con el script, se realizará el entrenamiento de la neurona para que aprenda el comportamiento que puede verse en la tabla 2. Para el llenado de estos vectores dentro de Matlab, se consideran los niveles en escala de 0 a 5, valores que se obtienen a través del potenciómetro. Suponemos que el potenciómetro lo colocamos en la posición más a la izquierda, y si capturamos el dato, este será cercano a cero **(Verificar la posición correcta nates de comenzar**). Ahora, el nivel bajo se consideran posiciones del potenciómetro que se encuentren entre 0 y 1.5; el nivel medio se consideran posiciones del potenciómetro que ingresen valores entre 1.8 y 3.5; se considera un nivel alto, las posiciones del potenciómetro que ingresen valores entre 3.8 y 5. Estos valores no son específicos, ya que sería complicado el acertar a valores determinados.

14. En la tabla 2, se puede observar que en la primera columna existen series de tres “Bajo”, tres “Medio” y tres “Alto”, esto nos indica que vamos a colocar el potenciómetro en su posición de más a la izquierda y comenzar a dar dos pequeños giros hacia la derecha. Después damos un giro mayor (esperando poder posicionarnos cerca de los límites de Medio definidos en el paso anterior), seguido de dos giros pequeños hacia la derecha, para

concluir con otro giro grande (esperando poder posicionarnos cerca de los límites de Alto definidos en el paso anterior), seguido de dos pequeños.

15. Para llenar la columna dos, nos posicionaremos cerca del extremo izquierdo del potenciómetro, luego otro giro grande que nos posicione cerca del centro y un nuevo giro grande que nos posicione cerca del extremo derecho; realizado esto, regresamos el potenciómetro a una posición cercana hacía el lado izquierdo y repetimos el proceso dos veces más.

16. Para el llenado del Target, vamos a posicionarnos cerca del límite izquierdo del potenciómetro y vamos a ir girando el potenciómetro hacía la derecha, esperando que en nueve giros lleguemos al extremo derecho del potenciómetro. Con todos los pasos realizados, quedará una tabla similar a la tabla 3.

Tabla 2.- Comportamiento a entrenar por la neurona adaptativa.

| Entrada 1 | Entrada 2 | Target |
|---|---|---|
| Bajo | Bajo | |
| Bajo | Medio | |
| Bajo | Alto | |
| Medio | Bajo | |
| Medio | Medio | |
| Medio | Alto | |
| Alto | Bajo | |
| Alto | Medio | |
| Alto | Alto | |

Tabla 3.- Tabla aproximada (y solo para ejemplificar) que se debe obtener al llenar los vectores de entrada y salida de la ejecución del script.

| Entrada 1 | Entrada 2 | Target |
|---|---|---|
| .1 | .3 | .1 |
| .5 | 1.5 | .8 |
| 1.2 | 4.34 | 1.5 |
| 2.3 | .01 | 2.6 |
| 2.7 | 2.333 | 3.45 |
| 3.24 | 4.41 | 3.98 |
| 3.8 | 1.002 | 4.1 |
| 4.3 | 2.7 | 4.7 |
| 4.7 | 4.955 | 5 |

17. En la ejecución del script, primero se solicitará el número de muestras, en este caso son 9 muestras, es decir, cada uno de los casos que compone la tabla 2.

18. Acto seguido, se solicitará que se ingrese el vector correspondiente a la entrada 1, para este caso la entrada se realizará por medio del potenciómetro. Antes de dar inicio al ingreso de datos, recuerde que el potenciómetro debe estar en una posición cercana a 0, para indicar

que se va a ingresar el siguiente valor es necesario dar un Enter. Es importante mencionar que antes de dar el Enter, ya se debe de tener modificado el potenciómetro en el siguiente valor a ingresar.

19. El programa solicitará el ingreso del segundo vector, en este caso hay que meter en orden los valores de la tabla 2 de la columna “Entrada 2”.
20. Por último, solicitará que ingresemos el target, en este caso es el vector correspondiente a “Target” de la tabla 2.
21. Una vez ingresados los datos, el script comenzará a entrenar a la neurona adaptativa. Una vez entrenada, pedirá si se quiere verificar algún vector de entrada. Estos vectores de entrada, son las combinaciones [Entrada 1; Entrada 2], tal como se ven la tabla 1. El alumno colocara algunos valores aleatorios con rango de entre 0 y 5 y mostrará al profesor el correcto funcionamiento.
22. Después de la ejecución del script, el alumno verificará cuales han sido los pesos sinápticos obtenidos.
23. **BUSCAR LA FORMA DE GRAFICAR ESTE, ya que no hay valores de 0 y 1, sino que son decimales**. Graficar los vectores de entrenamiento y Target, así como el comportamiento de los pesos sinápticos.
24. Una vez concluido esto, el alumno ingresara la siguiente línea a la ventana de comandos:

    salida=sim(ada,entradas)

25. Verificará si la salida es igual al vector de Target.

## CUESTIONARIO

1. El alumno debe de modificar las entradas y salidas del paso 7 para obtener dos salidas en la neurona, una salida va a ser un codificador binario a exceso 3 y la otra salida será el exceso 5. Discutir el análisis de los pesos sinápticos resultantes.
2. En el paso 24, si el vector de salida no es igual al vector de Target, ¿A qué se debe esta situación?, ¿Qué tan parecidos o lejanos son los vectores entre sí?
3. El alumno debe realizar la misma práctica para ingresar los valores por medio de un potenciómetro, pero en lugar de llenar dos vectores independientes, debe de llenar una sola matriz con el mismo esquema de entrada de la tabla 2.

## CONCLUSIONES

Escribe tus conclusiones de la práctica.

## BIBLIOGRAFÍA

# Laboratorio de Sistemas Basados en Redes Neuronales

## Práctica 5

## Clasificación con redes neuronales

### TEMAS

4. MÉTODO DE ENTRENAMIENTO HACIA ATRÁS (BACK - PROPAGATION)

4.1. Antecedentes.

4.2. Estructura de la Red.

4.3. Regla de Aprendizaje.

4.4. Modelos Electrónicos.

### OBJETIVOS

- Al término de esta práctica el alumno podrá:
- Identificar la clasificación de datos por medio de una red neuronal.
- La forma en la que la red clasifica los datos.
- Identificar los puntos de aprendizaje de la red.

### INTRODUCCIÓN

Las redes neuronales artificiales, también llamadas sistemas conexionistas, son modelos computacionales que se componen de diferentes neuronas artificiales conectadas entre sí. Estos esquemas tratan de emular la forma en la que los cerebros biológicos se encuentran constituidos, y como a partir de tal construcción, se logra el aprendizaje. Un esquema básico de una red neuronal, se puede visualizar en la figura 1 de esta práctica.

Una red neuronal artificial busca “aprender” patrones a partir de conjuntos de datos que sirven como entrada a la red y que se presentan como punto de “observación” del fenómeno a analizar. Partiendo de una cantidad elevada de información (datos), una red neuronal irá encontrando patrones dentro de tales conjuntos de información, hasta lograr un aprendizaje con el cual es posible clasificar comportamientos particulares dentro de un problema o fenómeno general.

Las redes neuronales tienen una gran cantidad de aplicaciones, sin embargo, las más comunes se encuentran en la clasificación de datos y en la búsqueda de patrones de información. En el caso de esta práctica, se utilizan para clasificar el comportamiento de elementos resistivos que se conectan en serie o en paralelo. Se consideran circuitos que contienen tres resistencias, y se calcula la resistencia equivalente, serie o paralela, según sea el caso.

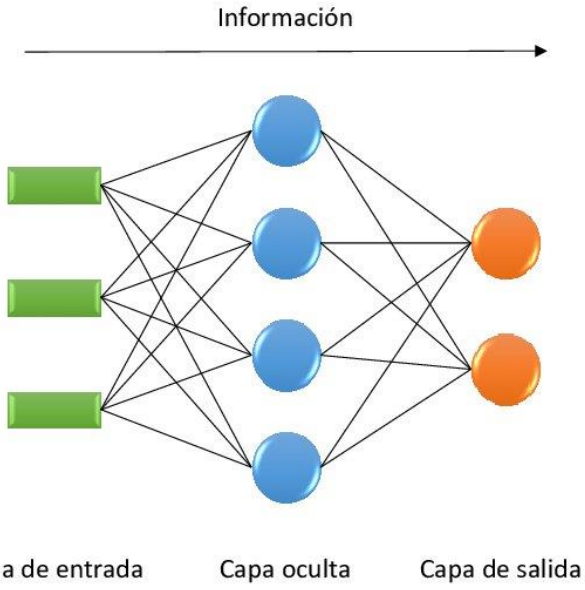



Fig. 1.- Esquema básico de una red neuronal, se ve una red con tres neuronas de entrada, dos neuronas de salida, y cuatro neuronas en su capa oculta. Imagen recuperada de file:///C:/Users/Vegueta/Downloads/Tesis-LinoFranciscoManjarrezMontao.pdf

## ACTIVIDADES PREVIAS A LA PRÁCTICA

1. El alumno realizará la lectura de la práctica.
2. El alumno descargará los scripts utilizado en esta práctica de http://virtual.cuautitlan.unam.mx/intar/?page_id=793 e interpretará cada uno de los elementos y comandos que componen el script.
3. El alumno debe de haber adquirido conocimiento de las herramientas a utilizar antes de la práctica.

## MATERIAL Y EQUIPO

1. Una computadora con Matlab (https://www.software.unam.mx/producto/matlab/ ) instalado y con las paqueterías de “MATLAB Support Package for Arduino Hardware” y “Deep learning” cargadas en el software.

## PROCEDIMIENTO EXPERIMENTAL

En el transcurso de esta práctica, se implementará un sistema de datos que simula la formación de circuitos resistivos, así como la obtención del resultado del cálculo de una resistencia equivalente en serie o en paralelo, resultado de tres resistencias con valores aleatorios. Estos conjuntos de valores de resistencias son generados en los scripts. En el primer script se genera una matriz que se compone de 3 resistencias y su resultado en serie o en paralelo. La mitad de las filas son resultados en serie y la otra mitad en paralelo. Así como también, se genera un vector denominado TargetResistencia, que indica con un 0 si la combinación de resistencias da un valor en serie, e indica un valor igual a 1, si son resistencias en paralelo.

El segundo script, genera matrices completas de combinaciones de resistencias en serie y de combinaciones en paralelo; estas matrices son utilizadas para probar la red neuronal ya entrenada.

Cada muestra corresponde a las tres resistencias aleatorias y un resultado de si se tiene un circuito en serie o en paralelo, obteniendo que cada vector de entrada posea 4 columnas. Las matrices representan esquemas como los mostrados en la figura 2.

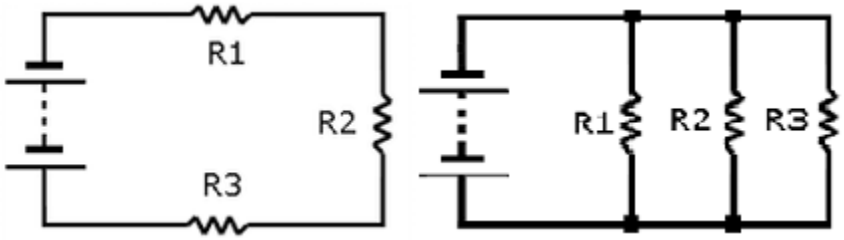



Fig. 2.- Circuitos en serie y paralelo, constituidos con tres resistencias, con las cuales se obtiene una resistencia equivalente.

1. El alumno abrirá los dos scripts en Matlab que se encuentran en la página http://virtual.cuautitlan.unam.mx/intar/?page_id=793
2. El alumno debe ejecutar ambos scripts, teniendo cuidado de que cuando el script solicite el número de muestras, estas sean la misma cantidad de muestras en ambas ejecuciones, ya que si no se coloca la misma cantidad de muestras, existirán errores de no compatibilidad de datos. El número de muestras debe de ser un valor par.
3. Para este punto, se debe de seleccionar una cantidad de muestras igual a 10.
4. Una vez realizado esto, el alumno podrá observar en el workspace cada una de las matrices generadas. Las de entrenamiento y las de prueba.
5. Hecho esto, el alumno debe de teclear el comando

   nnstart

6. Se desplegará una ventana denominada “neural network start”, en ella hay que seleccionar la opción “pattern recognition”. Lo que nos enviará hacía la configuración de una red neuronal de reconocimiento de patrones.
7. En la ventana se debe seleccionar el botón “Import”, lo que desplegara una nueva ventana en la que seleccionaremos en “Predictors”, matrizEntrenamiento; En “Responses” se debe de seleccionar TargetResistencia, y presionar el botón OK. Debe observar el alumno que estos son los valores previamente calculados por los scripts ejecutados.
8. Presionar next, hasta que se nos despliegue el número de neuronas que la red debe tener en la capa intermedia, en este caso, las neuronas por default son adecuadas, 10.
9. En la interfaz gráfica presionar el botón de entrenamiento “Train”, en este punto, la red neuronal ha sido configurada con una sola capa intermedia y 10 neuronas en ella. Como ya fu indicado cuales son los datos de entrenamiento y cuales son los Targets, o referencias, el alumno debe presionar el botón “Train”.

10. Una vez finalizado el entrenamiento, hay que guardar la red neuronal entrenada por medio de “Export Model -> Export to Workspace”. Este punto es muy importante, ya que aquí se guarda la red neuronal creada, así como su proceso de entrenamiento como un script. Para guardar tal script, selecciona el botón "Generate code -> Generate Simple Training Script".

11. En el cuestionario, el alumno debe dar nombre a la red neuronal realizada y a los datos que se quieran exportar al workspace, para fines de esta práctica basta con enviar al workspace el objeto con la red neuronal llamándola “resistencia”. Se desplegará la ventana "Export the trained network and results to a worspase variable named: “ y colocar el nombre de “resistencia”, podrá ver en el workspace los nuevos elementos. Presionar el botón OK.

12. En este punto, ya se tiene una red neuronal definida y entrenada, por lo que es posible hacer pruebas para verificar su funcionamiento.

13. El alumno puede ver la arquitectura de la red generada por medio de

    view(resistencia.Network)

14. La primera prueba es realizar la verificación de los valores de entrada con respecto al Target, por medio del siguiente comando:

    salidaEntrenamiento=sim(resistencia.Network, matrizEntrenamiento)

    Dicho comando simula la red neuronal “resistencia” con los valores de entrada “matrizEntrenamiento”

15. El alumno debe de verificar si el resultado es parecido al Target. Mostrar al profesor.

16. Para verificar el funcionamiento de la red neuronal, se puede observar la matriz de confusión y la gráfica ROC, por medio de

    plotconfusion(tarjetResistencia,salidaEntrenamiento)

    plotroc(tarjetResistencia,salidaEntrenamiento)

17. El alumno debe interpretar tales gráficos, mostrar a su profesor.

18. Ya probado el funcionamiento de la red neuronal, hay que hacer pruebas con las matrices de resultados en serie y en paralelo, por medio de:

    salidaPruebas=sim(resistencia.Network, matrizSerie)

    salidaPruebas=sim(resistencia.Network, matrizParalelo)

19. Comprobar sus resultados con lo aprendido por la red, ¿Clasificó adecuadamente las muestras?, es decir, ¿A los resultados en paralelo los clasificó como 1, y a los resultados en serie como 0? Mostrar al profesor.

20. Ahora el alumno debe de ingresar vectores que representen muestras de forma aleatoria para verificar la correcta clasificación de la red, obviando que estas muestras tengan

resultados correctos, manteniendo el orden de los vectores, tal que se componen en la siguiente manera:

salidaPruebas=sim(resistencia, [valor serie o paralelo; resistencia 1 ; resistencia 2; resistencia 3])

21. Si se ingresa un valor correcto de resultado en paralelo, la red debe de arrojar un valor 1; si se ingresa un valor correcto de resultado en serie, la red debe arrojar un valor 0. ¿Está correctamente clasificado?
22. El alumno debe ingresar ahora vectores aleatorios, pero con resultados incorrectos, ya no se debe de calcular el valor en serie o en paralelo correcto. ¿Qué sucede con los resultados de la clasificación?
23. Ingresar la siguiente secuencia de consultas de verificación, y observar que sucede con la salida de la red neuronal:

    salidaPruebas=sim(resistencia.Network, [1; 34 ; 12; 21])

    salidaPruebas=sim(resistencia.Network, [12; 34 ; 12; 21])

    salidaPruebas=sim(resistencia.Network, [34; 34 ; 12; 21])

    salidaPruebas=sim(resistencia.Network, [39; 34 ; 12; 21])

    salidaPruebas=sim(resistencia.Network, [80; 34 ; 12; 21])

24. Anotar los resultados de cada una de las consultas.
25. El alumno debe volver a ejecutar los scripts uno y dos, pero en esta ocasión va a seleccionar 100 muestras.
26. Debe de entrenar la red por medio de

    resistencia=train(resistencia.Network, matrizEntrenamiento, tarjetResistencia);

27. En este momento ya se tiene una red neuronal entrenada pero con 100 muestras de entrenamiento.
28. El alumno debe de ejecutar los pasos 15 a 25 nuevamente, ingresando la misma secuencia de entradas mostrada en el paso 24. Observe que es lo que sucede con la clasificación, y anote los resultados. Mostrar a su profesor.
29. El alumno debe volver a ejecutar los scripts uno y dos, pero en esta ocasión va a seleccionar 1000 muestras.
30. El alumno debe de ejecutar los pasos 15 a 25 nuevamente, ingresando la misma secuencia de entradas mostrada en el paso 24. Observe que es lo que sucede con la clasificación, y anote los resultados. Mostrar a su profesor.
31. Debe de entrenar la red por medio de

    resistencia=train(resistencia.Network, matrizEntrenamiento, TargetResistencia);

32. En este momento ya se tiene una red neuronal entrenada pero con 1000 muestras de entrenamiento.

33. El alumno debe de ejecutar los pasos 15 a 25 nuevamente, ingresando la misma secuencia de entradas mostrada en el paso 24. Observe que es lo que sucede con la clasificación, y anote los resultados. Mostrar a su profesor.

## CUESTIONARIO

1. ¿Qué es lo que está tomando como punto de clasificación la red neuronal?
2. ¿Qué se puede concluir acerca de las secuencias de datos de ingreso que se realizaron primero en el paso 24, y posteriormente en la repetición de tal paso, y de los resultados obtenidos para cada vector de ingreso con respecto a la cantidad de muestras de entrenamiento?
3. ¿La clasificación de la red neuronal es correcta?, ¿Por qué da datos tan “extraños” cuando se meten valores aleatorios incorrectos?, ¿Qué significan tales resultados?
4. El alumno debe modificar los scripts para que el seleccione desde la línea de comandos cuando quiere generar nuevas matrices de entrenamiento y de prueba, sin necesidad de estar ejecutando dos scripts por vez.

## CONCLUSIONES

Escribe tus conclusiones de la práctica.

## BIBLIOGRAFÍA

# Laboratorio de Sistemas Basados en Redes Neuronales

## Práctica 6

## Red de Hopfield

### TEMAS

7. REDES RECURRENTES

7.1. Red de Hopfield.

### OBJETIVOS

Al término de esta práctica el alumno podrá:

- Trabajar con redes neuronales de forma práctica que reconstruyan información.
- Identificará el correcto funcionamiento de las redes de Hopfield.
- Podrá ingresar patrones de entrenamiento desde el entorno físico.

### INTRODUCCIÓN

Las redes de Hopfield, son redes neuronales de tipo recurrente, es decir, se considera como una red completamente conectada, esto también indica, que cuando se ingresan datos de entrada, esta información se propaga hacía adelante y hacía atrás. Esta interacción permitirá que la rede se vaya adecuando a la información de entrada y se genere un estado de estabilidad.

Este tipo de redes, son utilizadas principalmente con entradas binarias, por lo que en esta práctica se llevara a cabo el ingreso de datos binarios. Estas redes son utilizadas como memoria asociativa, es decir, permite recuperar información a partir de conocimiento parcial de su contenido.

Por lo mencionado, este tipo de redes neuronales, es utilizado para la reconstrucción de información, a partir de información previamente “aprendida”.

En el cómputo formal, si sev requiere recordar un alumno existente en su base de datos, busca el registro y dirección de memoria en el cual se almacenó su información, tal como nombre o carrera, sin embargo, el cerebro humano no funciona de esta manera. Para identificar a un alumno de una carrera, el cerebro asocia información como rasgos faciales o letra con la que inicia su nombre. En un conjunto de estudiantes, el cerebro humano buscará y asociará información hasta encontrar un ser que se acerque al recuerdo facial que del alumno tiene, o encuentre un individuo que tenga un nombre que inicia con la letra recordada. Esta analogía es la forma en la que la red de Hopfield funciona, como si del proceso de asociación de un cerebro humano se tratara.

El esquema básico de la red de Hopfield puede verse en la figura 1.

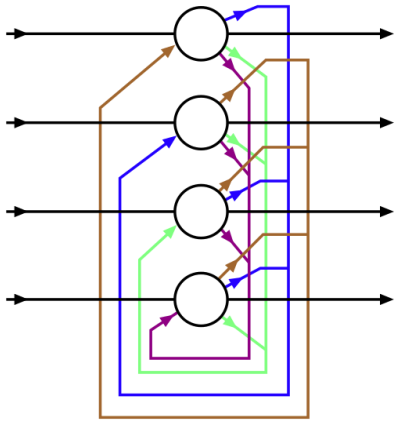


Fig. 1.- Esquema básico representativo de la red neuronal de Hopfield. Imagen recuperada de https://es.wikipedia.org/wiki/Archivo:Hopfield-net-vector.svg.

## ACTIVIDADES PREVIAS A LA PRÁCTICA

1. El alumno realizará la lectura de la práctica.
2. El alumno descargará el script utilizado en esta práctica de http://virtual.cuautitlan.unam.mx/intar/?page_id=793 e interpretará cada uno de los elementos y comandos que componen el script.
3. El alumno descargara las plantillas de entrenamiento y de prueba de la página http://virtual.cuautitlan.unam.mx/intar/?page_id=793 , estas deben ser impresas en negro, preferentemente en hojas recicladas, y llevarlas el día de la práctica.
4. El alumno debe de haber adquirido conocimiento de las herramientas a utilizar antes de la práctica.

## MATERIAL Y EQUIPO

1. Una computadora con Matlab (https://www.software.unam.mx/producto/matlab/ ) instalado y con las paqueterías de “MATLAB Support Package for Arduino Hardware” y “Deep learning” y “Neural network toolbox” cargadas en el software.
2. Una tarjeta de desarrollo Arduino (Preferentemente la versión UNO).
3. Una resistencia de 1 K.
4. Un LED.
5. Un sensor infrarrojo TCRT 5000, preferiblemente el módulo ya armado.
6. Un potenciómetro.
7. Cables de conexión, incluir cables macho-hembra.
8. Tableta Protoboard.
9. Un plumón negro.

## PROCEDIMIENTO EXPERIMENTAL

En este proceso experimental, el alumno ingresara datos del exterior que representan vectores de entrenamiento (en este caso letras), estos vectores están compuestos de matrices de **5 por 5 celdas**, por lo que el vector total comprende 25 elementos. Una vez ingresados los patrones de entrenamiento, se probaran figuras con errores, y se verificará si la red neuronal puede reconstruir la figura original. Se colocó una resolución baja de entrenamiento, para que el alumno pueda visualizar el proceso de generación de los vectores de entrada, así como el procesamiento de los mismos, obviamente, es posible ingresar vectores grandes de información.

El sensor debe tener la libertad de ser manipulado, es decir, no debe ser colocado en la protoboard, debe tener conexiones libres para poder recorrer las matrices de los patrones de entrada.

El recorrido lexicográfico de las figuras será de izquierda a derecha, y de arriba hacia abajo, es decir, iniciaremos el recorrido de las celdas en la esquina superior izquierda de la figura, y se terminará en la celda de la esquina inferior derecha de la misma, tal como se ve en la figura 2.

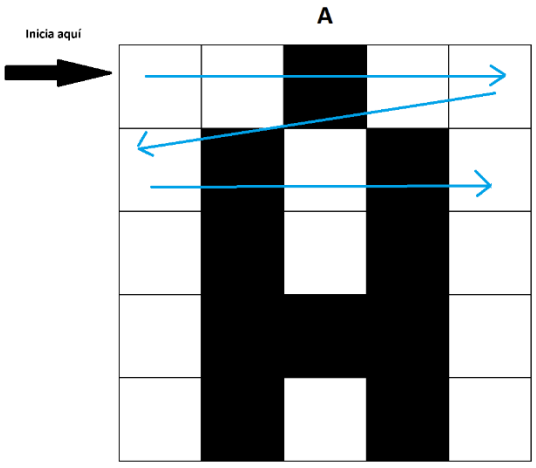



Fig. 2.- Forma correcta de leer la matriz de entrada de cada uno de los patrones a aprender.

1. Hay que **verificar el correcto funcionamiento del módulo infrarrojo**. El alumno debe conectar las terminales de alimentación del sensor, y pasarlo sobre superficies blancas y negras, este módulo cuenta con un LED embebido, si se coloca sobre una superficie blanca, el LED se prende; si se coloca sobre una superficie negra, el LED se apaga. De esta **forma el alumno debe de comprobar el correcto funcionamiento del sensor, y también debe de encontrar la distancia adecuada para leer los datos de entrada**. Como recomendación, el sensor debe estar con mucha cercanía a la página donde se encuentran los patrones.

2. El alumno descargara el script 1 de la dirección http://virtual.cuautitlan.unam.mx/intar/?page_id=793 y lo ejecutará en Matlab. Descomprimir el archivo zip en la carpeta por defecto de Matlab, generalmente está dentro de documentos. Ya que el script utiliza las imágenes comprimidas.

3. El alumno debe armar el sistema mostrado en la figura 3. Considera colocar cables de conexión largos al sensor infrarrojo para que este pueda tener movilidad, y pueda ser manipulado de acuerdo a las figuras a ingresar.

4. El alumno ejecutará el script de Matlab, el script solicitará el número de muestras por vector de entrada, en este caso serán 25 muestras; y solicitará el número de vectores de entrenamiento, en este caso serán 3, correspondientes a las letras A, B y C.

5. El ingreso de las muestras iniciará; es necesario que en el momento en que se inicie el conteo, el sensor debe de estar colocado en la primera casilla del primer patrón a ingresar. Cada que el LED cambia de estado, el sensor debe ser dirigido a la siguiente casilla, y esperar a que el LED vuelva a cambiar para dirigirlo nuevamente a la siguiente casilla. Hay que tener cuidado de que el sensor este captando la información correcta, un 1 para blanco y un 0 para negro.

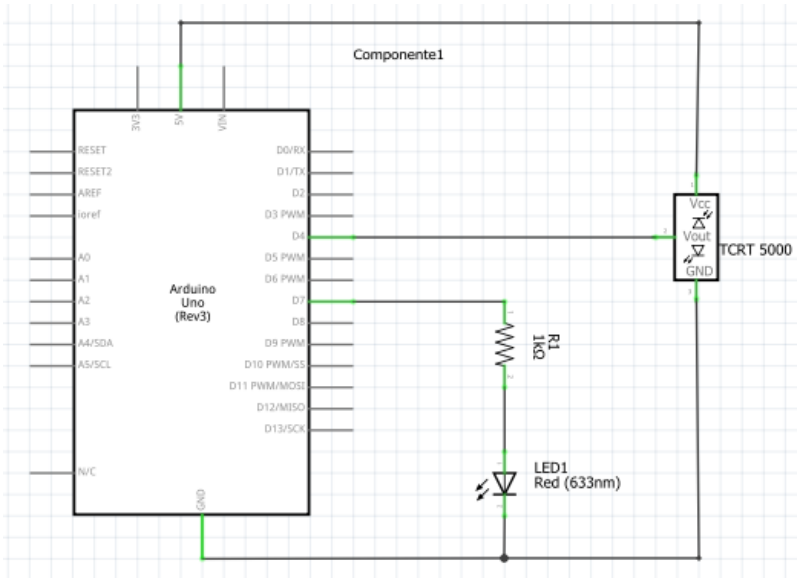



Fig. 3.- Sistema para probar la conexión entre Matlab y Arduino.

6. Cuando termina de leer las 25 casillas del primer patrón, el script dará oportunidad de descanso, por lo que, para ingresar el segundo patrón, hay que teclear Enter. De igual forma, es necesario que ya este el sensor en la primera casilla cuando se inicia el ingreso de datos del nuevo patrón.

7. Al terminar este proceso de ingreso de datos, se calcula y se obtiene la matriz de pesos, por lo que ahora el programa solicitará patrones para revisión, el alumno debe de colocar el primer patrón erróneo, la rede de Hopfield debe reconstruir el patrón adecuado, desplegando una imagen de una letra A. Mostrar al profesor.

8. El alumno modificará el primer patrón con error, correspondiente a la letra A, pintando una casilla de la matriz con negro, y volviendo a probar este nuevo vector. Verificar si lo reconoce.

9. El alumno irá repitiendo el mismo proceso, que en el paso 8, pintar una nueva casilla, y ver si la puede reconstruir, y continuar hasta que la red ya no pueda reconstruir el valor correcto. Anotar en que porcentaje de error, la red ya no reconstruye. Mostrar al profesor.

10. Repetir los pasos 7 a 9, pero ahora ingresando por primera vez el patrón con error correspondiente a la letra B, e ir modificando hasta que ya no se pueda reconstruir.

11. Repetir los pasos 7 a 9, pero ahora ingresando por primera vez el patrón con error correspondiente a la letra C, e ir modificando hasta que ya no se pueda reconstruir.

12. Es importante mencionar que en el primer patrón con error de cada una de las letras, la red neuronal si reconstruye correctamente la letra representada.

13. El script da como resultado valores -1 y 1, para estar en concordancia con el algoritmo, sin embargo, la red de hopfield de Matlab da resultados reales (que posteriormente se modifican a -1 y 1), para ver los valores reales del algoritmo, es necesario teclear.

    simulado{1},

14. Para poder visualizar los pesos generados en la red neuronal, es necesario utilizar el comando

    redHopfield.LW{1,1}

    Donde Net, es el nombre de la red entrenada, identifica dentro del script el nombre de la red, y encuentra la matriz de pesos generada.

## CUESTIONARIO

1. El alumno debe realizar un script para ingresar nuevos vectores de prueba, esto sin la necesidad de recalcular la matriz de pesos, es decir, el entrenamiento de la red. Los valores deben ser ingresados por medio del sensor, y la salida debe de estar dada en vectores compuestos de 1 y -1.

2. ¿Cuántas neuronas tiene esta red?, explique porque.

3. ¿En qué momento deja de reconstruir la imagen correcta?

4. ¿Qué sucede si existe un vector de prueba que tiene la misma distancia de hamming con dos o más vectores de entrenamiento?, realizar el experimento.

5.- Identifica el vector de bias, que le corresponde a la red neuronal entrenada.

## CONCLUSIONES

Escribe tus conclusiones de la práctica.

## BIBLIOGRAFÍA

# Laboratorio de Sistemas Basados en Redes Neuronales

## Bibliografía

En este apartado se muestra bibliografía complementaria para el desarrollo de estas prácticas.


1. Kim, P. (2017). Matlab deep learning. With machine learning, neural networks and artificial intelligence, 130(21). Springer
2. Paluszek, M., & Thomas, S. (2020). Practical Matlab deep learning. A Project-Based Approach, Michael Paluszek and Stephanie Thomas. Springer
3. Paluszek, M., & Thomas, S. (2016). MATLAB machine learning. Apress.
4. Ciaburro, G. (2017). MATLAB for machine learning. Packt Publishing Ltd.
5. Jiménez, E. C. (2021). Introducción al Machine Learning con MATLAB. Marcombo.

# Laboratorio de Sistemas Basados en Redes Neuronales

## Anexos

A continuación se presentan las páginas electrónicas en las que se encuentran las características y herramientas utilizadas en este manual de prácticas.

1. Matlab gratis para comunidad UNAM

   https://www.software.unam.mx/producto/matlab/

2. SpringerLink para poder tener acceso a algunos libros enlistados en la bibliografía, de acceso gratuito a la comunidad UNAM teniendo una cuenta activaen Bidi UNAM.

   https://link.springer.com/

3. Crear cuenta de Bidi UNAM

   https://www.bidi.unam.mx/

4. Características de la tarjeta Arduino

   https://www.arduino.cc/

5. Ayuda en los procesos de aprendizaje de Matlab

   https://www.mathworks.com/

6. Descarga de material necesario para estas prácticas

   http://virtual.cuautitlan.unam.mx/intar/?page_id=793