Diplomado "Redes Neuronales Artificiales"

Modulo 1: Redes Neuronales Completamente Conectadas
Fecha de elaboración: 23 de agosto del 2025

En este documento se describen los actividades a realizar como parte de la evaluación del primer modulo del diplomado. Las actividades constan de ejercicios donde es necesario realizar un desarrollo matemático y posteriormente un código de programación para validar los resultados matemáticos. La evidencia de la elaboración de los ejercicios se deberá entregar en forma de reporte.

Periodo de entrega: A partir de la publicación de este documento y hasta antes del 30 de septiembre del 2025.

Requisitos para la entrega del reporte:

1. La elaboración de los ejercicios (tanto desarrollo matemático como la programación) se debe realizar de manera **individual**.
2. La evidencia de la elaboración de los ejercicios se deberá entregar en forma de **reporte** a través de un documento en formato pdf que contenga las imágenes escaneadas del desarrollo matemático **hecho a mano** para cada ejercicio. Los escaneos deben abarcar la hoja completa.
3. El mismo reporte, deberá contener imágenes que ilustren los resultados de la programación de los ejercicios.
4. El reporte deberá estar debidamente organizado, respetando la colocación de imágenes (Escaneo de apuntes y resultados de la programación) en el mismo orden de aparición de los ejercicios descritos en este documento. Asimismo, se deberá agregar un indice que indique la pagina en que se encuentra cada elaboración de ejercicio.
5. Cada hoja/pagina empleada para el desarrollo de los ejercicios debe contener el nombre completo del estudiante en la parte superior de la misma (**escrito a mano**).
6. El lenguaje de programación es python. No esta permitido el uso de librerias o funciones "especiales" para resolver los ejercicios.
7. El envió del reporte junto con los códigos de programación se deberá enviar al profesor vía correo electrónico diplomado.nn@outlook.com .

Nota: En caso de no cumplir la totalidad de los requisitos el reporte no será considerado para evaluación.

Ponderación: El peso de esta tarea es 100% sobre la calificación del modulo 1. La ponderación se indica en cada ejercicio y se asigna si se muestra en el reporte la evidencia de la elaboración completa y correcta del mismo.

Elaboró:
Dr. Juan Moisés Arredondo Velázquez
email: juan.arredondo@cinvestav.mx

Vo.Bo.
M.C. Gerardo Uriel Pérez Rojas
Lic. Luis Rey Vargas Guadarrama

Ejercicio 1 : Optimizadores (Ponderación 5/100)

Proponer una función $y = f(x) + g(x) + \ldots$ que sea el resultado de la suma de al menos otras dos funciones (por ejemplo $g(x) = (x-3)^2 + 8$ y $f(x) = 20sen((x+\pi) * 5) - 1$). La idea es tener como referencia una función $y = f(x) + g(x) + \ldots$ que tenga al menos dos mínimos, preferentemente cada mínimo en distinta posición en y. Posteriormente, implementar un algoritmo numérico basado en gradiente descendente (GD) para encontrar la posición en x de un mínimo. Se deberá hacer una comparación del método numérico contra el método analítico.

Ejercicio 2 : Regresión lineal (Ponderación 10/100)

Realizar el ajuste de una hipótesis propuesta (por ejemplo $p = f(x) = \omega x$ o $p = \omega_0 + \omega_1 x$) a un conjunto de datos (conjunto de entrenamiento) generado en Python. Los puntos para elaborar son:

- Generar el conjunto de entrenamiento (X, Y), tal que $X \in [-8, 8]$ y se cuente con al menos 100 muestras $(m \geq 100)$
- Proponer una hipótesis $p = f(x)$
- Seleccionar alguna de la siguientes funciones de costo $(C(\omega))$ y obtener su derivada.

  1. $C(\omega) = \dfrac{1}{m}\sum_{i=1}^{m} |(P_i - T_i)|$ (MAE)
  2. $C(\omega) = \sqrt{\dfrac{1}{m}\sum_{i=1}^{m}(P_i - T_i)^2}$ (RMSE)

- Elaborar en Python el algoritmo de entrenamiento basado en GD generando como resultado tres graficas:

  1. T vs X y sobre la misma grafica $P(x)$ vs X
  2. $C(\omega)$ vs Iteraciones
  3. $C(\omega)$ vs ω

Ejercicio 3: Regresión polinómica (Ponderación 10/100)

Bajo el paradigma de regresión polinómica, realizar en Python el algoritmo de entrenamiento para lograr el ajuste de una hipótesis a los datos de los conjuntos de entrenamiento generados por la siguiente funcion:

1. $y = sin(x) \quad \text{para} \quad x = [0, 2\pi]$ (10 puntos para x)

se deben probar al menos tres hipótesis distintas (tres grados diferentes, por ejemplo: 1,2 y 3) ademas de explorar el entrenar con datos normalizados y sin normalizar. Adicionalmente se deben mostrar al menos las siguientes dos graficas para cada hipótesis evaluada:

1. Graficar y vs x y sobre la misma grafica $P(x)$ vs x (Mostrar la predicción de la hipótesis para al menos un 50% adicional al rango de x)
2. Grafica Costo vs Iteraciones

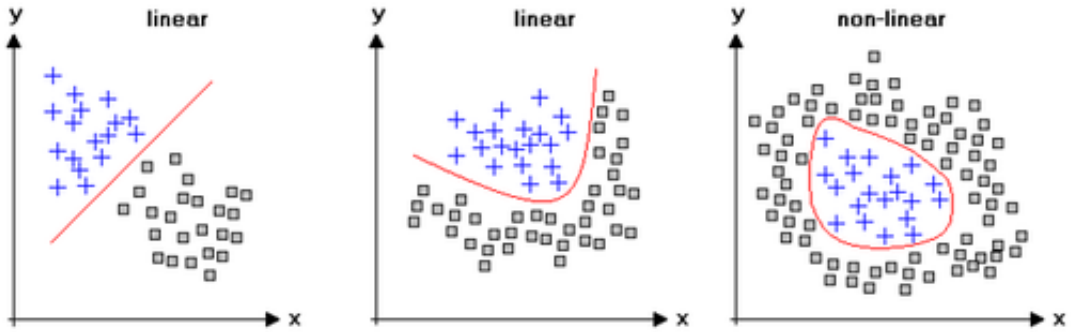

Figure 1: Ejemplos de conjuntos de entrenamiento de dos características

Ejercicio 4 : Clasificación binaria (Ponderación 15/100)

Elaborar en Python un algoritmo de regresión logística que sea capaz de clasificar un conjunto de entrenamiento de dos características generado artificialmente. Para ello se deberán considerar (crear) al menos tres diferentes distribuciones de las características. Como referencia, se pueden crear tres conjuntos de entrenamiento que visualmente se asemejen a los mostrados en la Figura 1. El número total de muestras debe ser de al menos 100 y la posición exacta de cada muestra debe ser generada de forma aleatoria.

Para cada conjunto de entrenamiento, los paso a seguir son los siguientes:

1. Generar el conjunto de entrenamiento
2. Emplear entropía cruzada como función de costo:

$$C(W) = -\frac{1}{m}\sum_{i=1}^{m} T^{(i)}\log(P^{(i)}) + (1-T^{(i)})\log(1-P^{(i)}) \tag{1}$$

3. Obtener su derivada con respecto a cada uno de los parámetros, considerando la siguiente hipótesis:

$$P_i = \frac{1}{1+e^{-z^{(i)}}} \tag{2}$$

Pudiendo z_i tener las siguientes formas:

- $z^{(i)} = \omega_0 + \omega_1 x_1^{(i)} + \omega_2 x_2^{(i)}$
- $z^{(i)} = \omega_0 + \omega_1 x_1^{(i)} + \omega_2 (x_1^{(i)})^2 + \omega_3 x_2^{(i)} + \omega_4 (x_2^{(i)})^2$
- Una ecuación (polinomio) propuesta por el estudiante

4. Para cada variante de hipótesis se deberá:
   - Implementar un algoritmo de entrenamiento basado en gradiente descendente.
   - Obtener la precisión de clasificación.
   - Generar el mapa de predicciones.
   - Mostrar la frontera de decisión.

Ejercicio 5 : Clasificación multiclase (Ponderación 20/100)

Elaborar en python algoritmos basados en redes neuronales para clasificar correctamente los datos mostrados en la Figura 2. Este conjunto de datos, etiquetado para cuatro clases, puede ser descargado desde el siguiente enlace (Conjunto de entrenamiento "spiral4classes.csv").

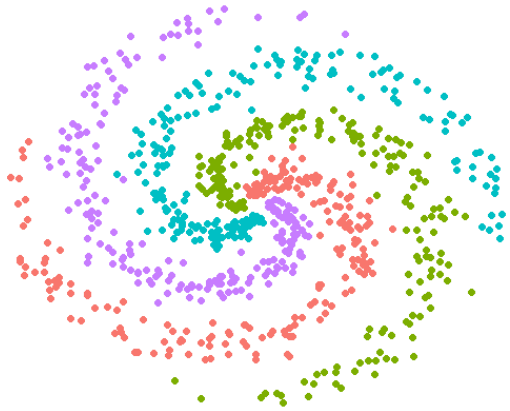

Figure 2: Visualización de conjunto de entrenamiento

La clasificación se deberá realizar para al menos dos configuraciones de red neuronal:

1. **Configuracion 1:** Red neuronal de tres capas
   - Una capa oculta, 50 neuronas, función de activación ReLU
2. **Configuracion 2:** Red neuronal de cuatro capas
   - Capa oculta 1, 25 neuronas, función de activación ReLU
   - Capa oculta 2, 25 neuronas, función de activación ReLU

Todas las capas, excepto la de salida, deben incluir sesgo ("bias") y todas las configuraciones tienen softmax como función de activación en la capa de salida.

Queda a criterio del estudiante el como llevar a cabo los siguientes procedimientos:

- Normalización
- Inicialización de los parámetros
- Definir función de costo
- Entrenamiento (Método de actualización de parámetros)

Se deberá reportar (Mostrar):

- Mapas de probabilidades
- Frontera de decisión
- Precisión (conjunto de entrenamiento, validación y prueba)
- Grafica costo vs épocas (conjunto de entrenamiento y validación)

Proyecto (Ponderación 40/100)

En el resto del documento se describen dos proyectos. El estudiante deberá realizar solo uno, así que queda a criterio del estudiante cual realizar. Ambas opciones implican procesamiento de imagenes, el cual podrá ser llevado a cabo por los estudiante mediante python y funciones especiales. Solo lo correspondiente a clasificación mediante algoritmos de redes neuronales artificiales debe ser resuelto limitándose al uso de los librerías "numpy" y "matplotlib".

Opción de proyecto 1:

Se deberán elaborar en python dos algoritmos basados en redes neuronales para el reconocimiento (clasificación) de caracteres. Para la etapa de entrenamiento y prueba de estos algoritmos se deberá utilizar una base de datos propia generada de la siguiente manera:

1. En hojas blancas dibujar una cuadricula de dos o tres centímetros por división vertical y horizontal.
2. En cada casilla escribir un carácter de tal manera que se completen al menos cinco caracteres en mayúsculas (A-Z) y/o dígitos (0-9). Repetir este proceso hasta completar al menos treinta veces cada carácter y/o cada dígito.
3. Para el proceso de digitalización se deberán escanear las hojas (preferentemente con un escaner) a fin de tener una imagen de cada hoja.
4. En python, se deberá realizar un código que abra secuencialmente cada imagen escaneada y extraiga subimagen correspondiente a cada región donde se encuentra cada dígito o carácter. A partir de esta extracción se generara el conjunto de entrenamiento (70%), validación (15%) y prueba (15%).

Los tres algoritmos a desarrollar comparten algunas similaridades tales como que:

- El número de neuronas en la capa de salida será igual al número de clases.
- La función de activación en la capa de salida es softmax.
- El número de neuronas y capas ocultas queda a criterio del estudiante.
- El algoritmo de optimización será mini-batch gradient descent (MBGD) con momentum

Sin embargo, las características especificas de cada algoritmo se describen a continuación:

1. **1er Algoritmo:**
   (a) El clasificador tomará como características de entrada cada uno de los pixeles de la imagen a clasificar
   (b) La normalización se deberá realizar para cada imagen por separado de tal forma que los valores de los pixeles estén en un rango [-1,1].
2. **2do Algoritmo:**
   (a) Aplicar el algoritmo *Local Binary Patterns* (LBP) a cada imagen a fin de obtener sus características. (El clasificador se entrena a partir de las características extraídas por LBP).
   (b) El clasificador tomará como características de entrada aquellas obtenidas por el algoritmo LBP.
   (c) La normalización se deberá aplicar a las características extraídas por el algoritmo LBP de tal forma que los valores de los pixeles estén en un rango [-1,1].

Para cada uno de los algoritmo se deberá reportar:

1. Precisión global para en conjunto de entrenamiento y prueba al final del entrenamiento.

2. (Figura 1) Grafica de costo vs épocas para el conjunto de entrenamiento (El costo para el conjunto de entrenamiento se calcula cada que se actualizan los parámetros) y sobre la misma figura, costo vs épocas para el conjunto de validación (El costo en para el conjunto de validación se calcula al final de cada época).

3. (Figura 2) Grafica de precisión vs épocas para el conjunto de entrenamiento (La precisión para el conjunto de entrenamiento se calcula cada que se actualizan los parámetros) y sobre la misma figura, precisión vs épocas para el conjunto de validación (La precisión en para el conjunto de validación se calcula al final de cada época).

4. En el caso del 2do algoritmo, se debe documentar en que consiste el método LBP.

Adicionalmente, se deberá aportar una discusión de las ventajas y desventajas de cada algoritmo.

Opción de proyecto 2:

Se deberá elaborar en python un código que capture imágenes desde la cámara web de la computadora y sea capaz de clasificar por lo menos tres diferentes letras formadas con las manos: (Figura 3)

Figure 3: Letras formadas por manos

Para realizar este proyecto se deben seguir los siguientes pasos:

1. Elaborar un código en python que permita capturar un secuencia de imágenes (video) con la cámara web. (Configurar la cámara a la menor resolución permitida y capturar a 15 cuadros por segundo (FPS)).

2. Empleando guantes rojos (también pueden ser verdes o azules), iniciar la captura de imágenes con la cámara web y formar con las manos las distintas letras que se pretenden clasificar.

3. Del video capturado, seleccionar los cuadros (imágenes o "frames") que serán procesadas para el entrenamiento.

4. Para cada imagen seleccionada se deberá tomar solo el mapa rojo (si es que se utilizaron guantes rojos) y aplicar una binarización. El objetivo es contar con imágenes binarias donde el objeto sean las manos (1 lógico) y el resto de la imagen sea el fondo (0 lógico).

5. Multiplicar (elemento por elemento) los mapas rojos de cada imagen por su correspondiente imagen binaria (mascara) a fin de tener una imagen en escala de grises donde solo existan las manos. En este proceso se remueve el fondo y resultan imágenes con las manos segmentadas.

6. Aplicar el algoritmo "Histogram Oriented Gradients" (HOG) a todas las imágenes segmentadas (mapas rojos por su correspondiente binarización). En este paso se forma el conjunto de entrenamiento (La descripción de este método se debe incluir en el reporte).

7. Normalizar el conjunto de entrenamiento de tal forma que el valor de cada característica se encuentre en un rango [0,1].

8. Realizar el entrenamiento de un modelo de red neuronal con una configuración definida por el estudiante. La función de activación será softmax y se empleará el optimizador mini-batch gradient descent (MBGD) con momentum.

9. Verificar que el entrenamiento fue "bueno" en términos generales

10. Una vez entrenando el modelo, probarlo de la siguiente manera:

    (a) Elaborar un código en python que permita capturar un secuencia de imágenes con la cámara web. (Configurar la cámara a la menor resolución permitida). Este código será uno distinto al mencionado en el inciso uno, puesto que se harán modificaciones

    (b) Para cada imagen capturada se deberá realizar el proceso de: binarización del mapa rojo, segmentación, extracción de características, normalización y clasificación.

    (c) Mostrar cada imagen capturada en secuencia junto a un indicador del resultado de la clasificación (En la misma figura).

    Considerar que una clase debería ser aquella que indique que no existen manos en la imagen o que no se esta formando ninguna letra que se pretenda clasificar.