Centro de Ciencias Básicas

Depto. Ciencias de la Computación

Área Académica: IAyFC

Aprendizaje Inteligente

Ingeniería en Computación Inteligente

6° “A”

“Examen I: Técnica Feed-Foward”

Alumnos:

César Eduardo Elias del Hoyo

José Luis Sandoval Pérez

Diego Emanuel Saucedo Ortega

Carlos Daniel Torres Macías

Profesor: Dr. Francisco Javier Luna Rosas

Aguascalientes, Ags. Agosto, 21 de 2023

EVIDENCIA DEL EXAMEN

Introducción:

a) ¿Qué es la técnica de propagación hacia atrás (Feed-Forward)?

Es un método de cálculo utilizado en algoritmos diseñados para entrenar el funcionamiento de redes neuronales, donde la información va en una sola dirección (hacia adelante) desde los nodos de entrada, a través de los ocultos (si los hay) y hasta los de salida. No existen ciclos ni bucles en la red. Este tipo de redes neuronales fueron el primer tipo de red neuronal artificial y son más simples que otras.

b) Explique la topología (arquitectura) de las redes neuronales Feed-Forward

La arquitectura de una red neuronal Feed-Forward consta de 3 tipos de capas: capa de entrada, capas ocultas y capa de salida. Cada capa está formada por unidades conocidas como neuronas y las capas esta interconectadas por pesos.

Capa de entrada: consta de neuronas que reciben entradas y las pasan a la siguiente capa. El número de capas de entrada está determinado por las dimensiones de los datos de entrada.

Capas ocultas: estas capas no están expuestas a la entrada ni a la salida y pueden considerarse como el motor computacional de la red neuronal. Las neuronas de cada capa oculta toman la suma ponderada de las salidas de la capa anterior, aplicando una función de activación y pasan el resultado a la siguiente capa. La red puede tener cero o más capas ocultas.

Capa de salida: la capa final que produce la salida de las entradas dadas. La cantidad de neuronas en la capa de salida depende de la cantidad de salidas posibles para las que está diseñada la red.

c) Aplicaciones de la red neuronal Feed-Forward

Este tipo de red se utilizan en una variedad de tareas de aprendizaje automático, tales como:

- Reconocimiento de patrones: se centra en la identificación de patrones y regularidades en los datos. Se basa en la clasificación de datos basándose en conocimientos a priori o información estadística extraída de los patrones.
- Análisis de regresión: método estadístico que estima las relaciones entre una variable dependiente y una o más variables independientes. Explora la forma y la fuerza de relación, permitiendo realizar predicciones o pronósticos basados en los datos observados.
- Reconocimiento de imagen
- Predicción de series de tiempo
- A pesar de su simplicidad, este tipo de redes pueden modelar relaciones complejas en datos y han sido base para arquitecturas de redes neuronales más complejas.

Explicación y análisis

a. **En que consiste los datos de entrenamiento y sus valores target asociados:**

Los datos de entrenamiento son un conjunto de datos extremadamente grandes que se utilizan para enseñar un modelo de aprendizaje. Estos datos se usan para enseñar a los modelos de predicción como extraer características relevantes para producir resultados deseados. Dentro del aprendizaje supervisado los datos de entrenamiento son etiquetados mientras en el no supervisado no están etiquetados. Ahora bien, los valores target sirvan como base de entrenamiento para el modelo, ya que guían el proceso de aprendizaje al proporcionar un punto de referencia para el desempeño del modelo.

b. **La razón de aprendizaje de las redes neuronales Feed-Forward**

La razón de aprendizaje o tasa de aprendizaje es un parámetro que controla cuanto cambia el modelo en respuesta al error estimado cada vez que se actualizan los pesos del modelo. Elegir la tasa de aprendizaje es un desafío ya que un valor demasiado pequeño puede resultar en un proceso de entrenamiento largo que podría atascarse, por otro lado, un valor muy alto puede resultar en aprender un conjunto de pesos demasiado rápido o un entrenamiento inestable. Metafóricamente se puede definir como la velocidad a la que “aprende” el modelo de aprendizaje.

c. **El bias de las redes neuronales Feed-Forward**

El bias o sesgo en una red neuronal se define como la constante que se suma al producto de características y pesos. Se utiliza para compensar el resultado, ayudando así a los modelos a cambiar la función de activación hacia el lado positivo o negativo. También cuenta con una función principal que es desplazar la función de activación de la neurona, permitiendo así que la red aprenda a modelar relaciones no lineales entre las entradas y salidas. Tener un bias evita que la neurona tenga una entrada neta siempre negativa, lo que haría que nunca

se activase. Cabe destacar que el valor del bias se puede ajustar durante el entrenamiento para optimizar su rendimiento.

d. Los pesos de entrenamiento de las redes neuronales Feed-Forward

Los pesos son valores numéricos asociados con las conexiones entre neuronas en diferentes capas de la red. Cada conexión de una neurona a otro tiene un peso asociado que significa la fuerza y la dirección de la influencia que una neurona tiene sobre otra. Los pesos son cruciales porque influyen directamente en las señales transmitidas a través de la red y determinan la salida de la red. En la fase de entrenamiento la red aprende ajustando iterativamente estos pesos para predecir la salida correcta para un conjunto determinado de entradas.

Explicación de las salidas de la red

a) **Propaga las entradas hacia delante en base a la siguiente formula:**

$$I_j = \Sigma w_{ij} Oi + \theta_j$$

Calcular las entradas/salidas de las neuronas en base a la siguiente formula:

$$O_j = \frac{1}{1 + e^{-I_j}}$$

Considerando las Figuras 1 y 2, podemos explicar las salidas de la red neuronal paso a paso:

Propagación hacia adelante:

La red neuronal recibe tres entradas (X1, X2, X3) como se muestra en la Figura 1.

Para cada neurona en la capa oculta (j = 1, 2, 3, 4), se calcula la entrada neta (I_j) utilizando la siguiente fórmula:

$$I_j = \Sigma w_{ij} Oi + \theta_j$$

- **W_{ij}:** Peso que conecta la neurona i a la neurona j
- **O_i:** Salida de la neurona i
- **Θ_j:** Bias de la neurona j

En la Figura 1, se muestran los valores específicos de W_{ij} y θ_j para cada neurona.

La salida (O_j) de cada neurona en la capa oculta se calcula utilizando la función de activación sigmoidea:

$$O_j = \frac{1}{1 + e^{-I_j}}$$

Esta función transforma la entrada neta en una salida entre 0 y 1.

El mismo proceso se repite para la neurona de salida, utilizando las salidas de las neuronas ocultas como entradas. La entrada neta (I_j) y la salida (O_j) de la neurona de salida se calculan de la siguiente manera:

$$\boldsymbol{I_j = \Sigma w_{ij} Oi + \theta_j}$$

$$\boldsymbol{O_j = \frac{1}{1 + e^{-I_j}}}$$

La salida final de la red neuronal es O_j, un valor entre 0 y 1. Esta salida puede interpretarse como la probabilidad de que la entrada pertenezca a una clase específica.

Suponiendo que las entradas son X1 = 0.5, X2 = 0.7 y X3 = 0.8, podemos calcular la salida de la red neuronal paso a paso:

Capa oculta:

- Neurona 1:
  - I_1 = (0.2 * 0.5) + (0.1 * 0.7) + (0.2 * 0.8) + 0.2 = 0.69
  - O_1 = 1 / (1 + e^(-0.69)) = 0.6645
- Neurona 2:
  - I_2 = (0.4 * 0.5) + (-0.5 * 0.7) + (-0.3 * 0.8) + 0.1 = 0.01
  - O_2 = 1 / (1 + e^(-0.01)) = 0.5025
- Neurona 3:
  - I_3 = (0.2 * 0.5) + (-0.3 * 0.7) + (-0.2 * 0.8) + 0.4 = 0.22
  - O_3 = 1 / (1 + e^(-0.22)) = 0.5556
- Neurona 4:
  - I_4 = (0.2 * 0.5) + (-0.4 * 0.7) + (-0.3 * 0.8) + 0.4 = 0.06
  - O_4 = 1 / (1 + e^(-0.06)) = 0.5128

Capa de salida:

- I_j = (-0.2 * 0.6645) + (-0.4 * 0.5025) + (-0.3 * 0.5556) + 0.4273 = 0.4022

- O_j = 1 / (1 + e^(-0.4022)) = 0.6327

En este ejemplo, la salida O_j = 0.6327 indica que la red neuronal tiene una probabilidad del 63.27% de que la entrada pertenezca a una clase específica.

b) Calcular el error de salida de la red en base a la siguiente formula:

$$\boldsymbol{Err_j = O_j(1 - O_j)(T_j - O_j)}$$

Calcular el error de las capas oculta en base a la siguiente formula:

$$\boldsymbol{Err_j = O_j(1 - O_j)\sum_k Err_j W_{jk}}$$

El error de la neurona de salida (Err_j) se calcula utilizando la siguiente fórmula:

$$\boldsymbol{Err_j = O_j(1 - O_j)(T_j - O_j)}$$

- **O_j:** Salida de la neurona de salida
- **T_j:** Valor objetivo (valor deseado para la salida)

El error de cada neurona en la capa oculta (Err_j) se calcula utilizando la siguiente fórmula:

$$\boldsymbol{Err_j = O_j(1 - O_j)\sum_k Err_j W_{jk}}$$

- **O_j:** Salida de la neurona j
- **W_{jk}:** Peso que conecta la neurona j a la neurona k
- **Err_k:** Error de la neurona k

Suponiendo que el valor objetivo (T_j) para la salida es 1, podemos calcular el error de la red neuronal:

Error de salida:

Err_j = 0.6327(1-0.6327)(1-0.6327) = 0.0952

Error de las capas ocultas:

- Neurona 1:
  - Err_1 = 0.6645(1 - 0.6645) * (0.0952 * -0.2 + 0.0952 * -0.4 + 0.0952 * -0.3) = -0.0096
- Neurona 2:
  - Err_2 = 0.5025(1 - 0.5025) * (0.0952 * -0.2 + 0.0952 * -0.4 + 0.0952 * -0.3) = -0.0072
- Neurona 3:
  - Err_3 = 0.5556(1 - 0.5556) * (0.0952 * -0.2 + 0.0952 * -0.4 + 0.0952 * -0.3) = -0.0081
- Neurona 4:
  - Err_4 = 0.5128(1 - 0.5128) * (0.0952 * -0.2 + 0.0952 * -0.4 + 0.0952 * -0.3) = -0.0075

El error de la red neuronal indica qué tan lejos está la salida real (O_j) de la salida deseada (T_j). En este caso, el error es relativamente pequeño, lo que indica que la red neuronal está funcionando bien.

El error de la red neuronal se utiliza para actualizar los pesos y bias durante el entrenamiento. El objetivo del entrenamiento es minimizar el error de la red neuronal para que pueda realizar predicciones precisas.

c) Ajustes de pesos en base a la siguiente formula:

$$\Delta \boldsymbol{W_{ij}} = (\boldsymbol{I})\boldsymbol{Err_j I_i}$$

$$\boldsymbol{W_{ij}} = \boldsymbol{W_{ij}} + \Delta \boldsymbol{W_{ij}}$$

Ajustes del Bias en base a la siguiente formula:

$$\Delta \boldsymbol{\theta_j} = (\boldsymbol{I})\boldsymbol{Err_j}$$

$$\boldsymbol{\theta_j} = \boldsymbol{\theta_j} + \Delta \boldsymbol{\theta_j}$$

El ajuste de cada peso (ΔW_{ij}) se calcula utilizando la siguiente fórmula:

$$\Delta W_{ij} = (I) Err_j I_i$$

I: Tasa de aprendizaje

Err_j: Error de la neurona j

O_j: Salida de la neurona i

Luego, el peso (W_{ij}) se actualiza de la siguiente manera:

$$W_{ij} = W_{ij} + \Delta W_{ij}$$

El ajuste de cada bias ($\Delta\theta_j$) se calcula utilizando la siguiente fórmula:

$$\Delta\theta_j = (I) Err_j$$

Luego, el bias (θ_j) se actualiza de la siguiente manera:

$$\theta_j = \theta_j + \Delta\theta_j$$

Suponiendo que la tasa de aprendizaje (η) es 0.1, podemos calcular los ajustes de pesos y bias:

Neurona 1:

ΔW_14 = 0.1 * -0.0096 * 0.5 = -0.00048

ΔW_15 = 0.1 * -0.0096 * 0.7 = -0.000672

...

Neurona 2:

ΔW_24 = 0.1 * -0.0072 * 0.5 = -0.00036

ΔW_25 = 0.1 * -0.0072 * 0.7 = -0.000504

...

Neurona 3:

$\Delta W_{34} = 0.1 * -0.0081 * 0.5 = -0.000405$

$\Delta W_{35} = 0.1 * -0.0081 * 0.7 = -0.000567$

...

Neurona 4:

$\Delta W_{46} = 0.1 * -0.0075 * 0.6645 = -0.00049875$

$\Delta W_{56} = 0.1 * -0.0075 * 0.5025 = -0.000376875$

Actualización de pesos:

$W_{14} = 0.2 - 0.00048 = 0.19952$

$W_{15} = 0.4 - 0.000672 = 0.399328$

...

$W_{46} = -0.2 - 0.00049875 = -0.20049875$

$W_{56} = -0.4 - 0.000376875 = -0.400376875$

Ajustes de bias:

$\Delta\theta_4 = 0.1 * -0.0096 = -0.00096$

$\Delta\theta_5 = 0.1 * -0.0072 = -0.00072$

$\Delta\theta_6 = 0.1 * 0.0952 = 0.00952$

Actualización de bias:

$\theta_4 = 0.2 - 0.00096 = 0.19904$

$\theta_5 = 0.1 - 0.00072 = 0.09928$

$\theta_6 = 0.4273 + 0.00952 = 0.43682$

Los ajustes de pesos y bias se utilizan para mejorar el rendimiento de la red neuronal. Al ajustar los pesos y bias, la red neuronal puede aprender a realizar predicciones más precisas.

Implementación de la red neuronal

La implementación se encuentra en el código realizado en Notebook Jupyter anexado en el archivo comprimido correspondiente entregado por parte de cada uno de los integrantes que formaron parte de la realización de este trabajo

Conclusiones

Durante la realización del examen, comprendimos el concepto de la técnica de propagación hacia atrás, también conocida como *Feed-Forward,* la cual nos permitió llevar un a cabo procedimiento efectivo mediante la organización y disposición de una red formando capas neuronales. Este procedimiento de aprendizaje inteligente nos permitió evaluar y enseñarle a la red neuronal hacer un análisis por capaz en una única dirección, como su nombre lo indica, hacia adelante. Comprendimos el funcionamiento de cada una de las capaz (de entrada, ocultas y de salida) de forma que estás van analizando y transformando los vectores, de forma que se obtiene una respuesta a la red neuronal. Por último, realizamos el procedimiento manual y en código que nos permitió tener una mejor visión y comprehensión de como se maneja la red neuronal y como podemos emplear distintos métodos de aprendizaje en si misma. En conclusión, este examen fue un trabajo que realizamos en equipo con el fin de aportar ideas entre los integrantes y obtener un resultado final acertado para la práctica planteada, comprendimos todos los temas planteados y los procesos a emplear fueron un poco largos, pero con un resultado efectivo que buscábamos encontrar.

Bibliografía

BitBoss. (27 de Febrero de 2020). *Redes Neuronales: De la Neurona al Perceptrón Multicapa en 9 minutos. El problema XOR*. Obtenido de YouTube: https://www.youtube.com/watch?v=MU3cLsSfnME

Brownlee, J. (12 de Septiembre de 2020). *Understand the Impact of Learning Rate on Neural Network Performance*. Obtenido de Machine Learning Mistery: https://machinelearningmastery.com/understand-the-dynamics-of-learning-rate-on-deep-learning-neural-networks/

Coach, I. (15 de Noviembre de 2021). *Construcción de una red feedforward en Python*. Obtenido de YouTube: https://www.youtube.com/watch?v=U2Cs9kdwmmU

Guzman, D. S. (22 de Mayo de 2020). *Perceptrón Multicapa Python*. Obtenido de YouTube: https://www.youtube.com/watch?v=1_w8xoKE_J0

Rouse, M. (27 de Junio de 2023). *Training Data*. Obtenido de Techopedia: https://www.techopedia.com/definition/33181/training-data

Smolic, H. (12 de Julio de 2023). *Understanding Target Variables in Machine Learning*. Obtenido de Graphite: https://graphite-note.com/understanding-target-variables-in-machine-learning

Vasquez, I. (03 de Octubre de 2023). *Red neuronal multicapa explicada (perceptrón multicapa)*. Obtenido de YouTube: https://www.youtube.com/watch?v=BGT46tf7iCM