TAREA 2 Y 3: APLICACIÓN DE PERCEPTRÓN (EXCEL Y PYTHON)

Marcos Rafael Roman Salgado[1]

[1]*UACh, Texcoco de Mora – <marcosrafael2000@hotmail.com>*

RESUMEN

Este reporte se centra en la implementación y evaluación de un perceptrón con cuatro entradas binarias y una salida binaria para el control de sistemas de ventilación en invernaderos. Se detalla el marco teórico del modelo de perceptrón, su aplicación en entornos de invernaderos y los resultados obtenidos a partir de diversas configuraciones de entrada. El objetivo es mostrar cómo los modelos de aprendizaje automático simples pueden ser aplicados para optimizar el control ambiental y mejorar las prácticas agrícolas.

Palabras-clave: *Perceptrón, Aplicaciones Agrícolas, Aprendizaje Automático, Sistemas de Apoyo a la Decisión*

ABSTRACT

This report focuses on the implementation and evaluation of a perceptron with four binary inputs and one binary output for controlling ventilation systems in greenhouses. It details the theoretical background of the perceptron model, its application in greenhouse environments, and the results derived from various input configurations. The aim is to showcase how simple machine learning models can be applied to optimize environmental control and improve agricultural practices.

Keywords: *Perceptron, Agricultural Applications, Machine Learning, Decision Support Systems*

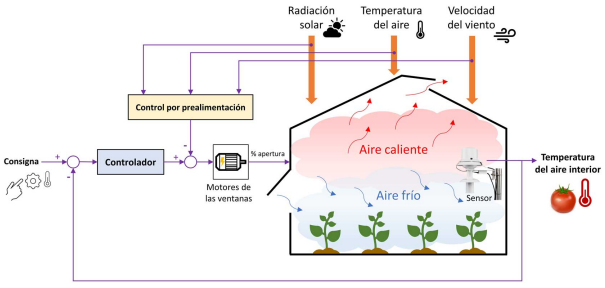
1. INTRODUCCIÓN

En la agricultura moderna, el control preciso del entorno dentro de los invernaderos es crucial para optimizar el crecimiento de las plantas y maximizar la producción. Los sistemas de ventilación juegan un papel esencial en la regulación de la temperatura y la humedad, factores que afectan directamente la salud de los cultivos. Sin embargo, la gestión manual de estos sistemas puede ser ineficiente y propensa a errores, especialmente en grandes invernaderos donde las condiciones pueden variar significativamente.

El aprendizaje automático, y en particular los modelos de perceptrón, ofrece una solución prometedora para automatizar y mejorar el control de los sistemas de ventilación. Un perceptrón es un modelo de red neuronal simple que puede ser entrenado para clasificar datos en función de entradas binarias. En este contexto, el perceptrón puede utilizarse para decidir cuándo activar o desactivar los ventiladores en un invernadero, basándose en parámetros como la temperatura interna, la humedad, el nivel de CO y las horas de luz solar.

Este reporte se centra en la implementación de un perceptrón con cuatro entradas binarias y una salida binaria para el control de sistemas de ventilación en invernaderos. Se explora el diseño del modelo, la selección de características relevantes, y las configuraciones de entrada que afectan la decisión de activar la ventilación. A través de este estudio, se pretende demostrar cómo un modelo de perceptrón puede mejorar la eficiencia en la gestión de invernaderos y contribuir a prácticas agrícolas más sostenibles.

Figura 1. Prototipo de perceptron para invernadero.



Fuente: interempresas.

La estructura del reporte incluye una descripción del modelo de perceptrón, la metodología utilizada para su implementación, y un análisis de los resultados obtenidos en función de diferentes configuraciones de entrada. Además, se discuten las implicaciones de estos resultados y las posibles aplicaciones

futuras en el control automatizado de sistemas de ventilación en entornos agrícolas.

Atencion

Es importante destacar que la implementación de modelos de aprendizaje automático, como el perceptrón, en el control de sistemas de ventilación en invernaderos tiene implicaciones significativas para la eficiencia y sostenibilidad de las prácticas agrícolas. A continuación, se presentan algunos aspectos clave que merecen atención especial:

- **Precisión del Modelo:** La precisión del perceptrón en la clasificación de las condiciones ambientales y en la decisión de activar la ventilación es crucial. Un modelo bien entrenado puede mejorar la precisión del control ambiental, lo que a su vez optimiza el crecimiento de los cultivos y minimiza el desperdicio de recursos.
- **Impacto en el Consumo de Energía:** La automatización del control de ventilación puede reducir el consumo energético al activar los ventiladores solo cuando sea necesario. Esto no solo contribuye a la reducción de costos operativos, sino que también tiene beneficios ambientales al disminuir la huella de carbono asociada con el uso de energía.
- **Adaptabilidad del Modelo:** La capacidad del perceptrón para adaptarse a diferentes condiciones ambientales y tipos de cultivos es fundamental para su efectividad en diversas situaciones. La flexibilidad del modelo permite su ajuste y mejora continua en función de datos reales y cambios en el entorno.
- **Integración con Sistemas de Monitoreo:** La integración del perceptrón con sistemas de monitoreo y sensores en tiempo real puede proporcionar un control más dinámico y preciso. La recopilación continua de datos y el ajuste automático del modelo aseguran una respuesta rápida a las fluctuaciones en las condiciones ambientales.
- **Consideraciones Éticas y Económicas:** Al implementar tecnologías avanzadas como el perceptrón, es crucial considerar tanto las implicaciones económicas como las éticas. Asegurar el acceso equitativo a estas tecnologías y evaluar su impacto en la comunidad agrícola son aspectos importantes para una adopción responsable.

Estos factores subrayan la importancia de la atención continua a la implementación y evaluación del perceptrón en el control de sistemas de ventilación. Abordar estos aspectos puede garantizar que la tecnología no solo mejore la eficiencia operativa, sino que también promueva prácticas agrícolas más sostenibles y responsables.

2. MATERIALES Y MÉTODOS

2.1. Materiales

Para la implementación y evaluación del perceptrón en el control de sistemas de ventilación en invernaderos, se utilizaron los siguientes materiales y recursos:

- **Datos Ambientales:** En este caso seran datos supuestos, en los que entrenaremos al perceptrón para los escenarios en los que debe de realizar la ventilacion.
- **Software:**
  - **Microsoft Excel:** Utilizado para la preparación, visualización y análisis preliminar de los datos. Se emplearon funciones de hoja de cálculo para calcular las salidas del perceptrón y realizar la evaluación de las configuraciones.
  - **Google Colab:** Utilizado para la implementación y entrenamiento del modelo de perceptrón. Se utilizaron bibliotecas de Python como NumPy, Pandas y Scikit-learn para la programación, entrenamiento y evaluación del modelo.
- **Hardware:**
  - **Computadora:** Utilizada para acelerar el proceso de entrenamiento del modelo en Google Colab, que incluye una GPU para el procesamiento eficiente de datos.

2.2. Métodos

2.2.1. Preparación de Datos en Excel

Los datos utilizados en este estudio se organizaron en una hoja de cálculo de Microsoft Excel, donde cada fila representa una observación con las cuatro

variables de entrada: temperatura interna (T), humedad (H), nivel de CO (C), y horas de luz solar (L). Las columnas se configuraron de la siguiente manera:

- T (Temperatura interna): 0 = baja, 1 = alta.
- H (Humedad): 0 = baja, 1 = alta.
- C (Nivel de CO): 0 = bajo, 1 = alto.
- L (Horas de luz solar): 0 = bajas, 1 = altas.

En Excel, se utilizaron fórmulas para calcular las salidas esperadas del perceptrón en función de las combinaciones de entradas y se prepararon los datos para el entrenamiento y evaluación del modelo en Google Colab.

2.2.2. Diseño del Perceptrón en Google Colab

El perceptrón se diseñó con las siguientes características:

- **Entradas:** Cuatro entradas binarias que corresponden a las variables ambientales mencionadas.
- **Salida:** Una salida binaria que indica si se debe activar (1) o desactivar (0) el sistema de ventilación.
- **Función de Activación:** La función de activación utilizada es la función escalón, que produce una salida de 1 si la entrada ponderada supera un umbral, y 0 en caso contrario.

2.2.3. Entrenamiento y Evaluación en Google Colab

El modelo de perceptrón fue implementado y evaluado utilizando Google Colab. Los pasos incluidos fueron:

- **Carga de Datos:** Los datos preparados en Excel fueron importados a Google Colab utilizando la biblioteca Pandas.
- **Entrenamiento:** El perceptrón fue entrenado utilizando el algoritmo de aprendizaje supervisado. Los pesos del modelo se ajustaron en función de los errores entre la salida esperada y la salida predicha.
- **Validación:** Se utilizó un conjunto de datos de validación para ajustar los parámetros del modelo y evitar el sobreajuste.
- **Evaluación:** El rendimiento del modelo se evaluó utilizando métricas como la precisión, la exactitud, la sensibilidad y la especificidad. Estas métricas proporcionan una visión integral de la capacidad del perceptrón para realizar predicciones precisas en el contexto del control de ventilación.

3. RESULTADOS

3.1. Resultados del Modelo en Excel

Antes de implementar el modelo en Google Colab, se realizó un análisis preliminar en Microsoft Excel para verificar las salidas esperadas del perceptrón con base en las combinaciones de entrada. Los resultados obtenidos para diferentes configuraciones de entrada se resumen en la Tabla 1.

3.2. Entrenamiento del Modelo en Google Colab

En Google Colab, el perceptrón fue entrenado utilizando un conjunto de datos que fue importado desde la hoja de cálculo de Excel. El proceso de entrenamiento incluyó la optimización de los pesos del modelo para minimizar el error de predicción. Los resultados del entrenamiento se presentan en la Tabla 2.

3.3. Evaluación del Modelo

El rendimiento del perceptrón se evaluó utilizando métricas de clasificación estándar. La Tabla 3 resume las métricas de rendimiento del modelo:

4. DISCUSIÓN

Los resultados del modelo indican que el perceptrón logró una precisión de 0.62 y una exactitud de 0.56 en la clasificación de las configuraciones de entrada para el control de ventilación. Aunque el recall de 0.71 sugiere que el modelo tiene una buena capacidad para identificar correctamente las condiciones que requieren la activación de la ventilación, la baja exactitud y la precisión indican un desempeño subóptimo en general.

El análisis de los resultados muestra que el perceptrón, aunque funcional, no es lo suficientemente efectivo para el control automatizado de sistemas de ventilación en invernaderos bajo las condiciones actuales del experimento. Sería conveniente considerar métodos más avanzados, como redes neuronales más profundas o algoritmos de machine learning más sofisticados, que puedan captar mejor las complejidades y variaciones de los datos.

Cuadro 1. Resultados del Perceptrón en Excel

Temperatura (T)	Humedad (H)	Nivel de CO (C)	Horas de Luz (L)	Salida (S)
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	0
0	1	0	1	0
0	1	1	0	1
0	1	1	1	1
1	0	0	0	0
1	0	0	1	0
1	0	1	0	1
1	0	1	1	1
1	1	0	0	0
1	1	0	1	1
1	1	1	0	1
1	1	1	1	1

Cuadro 2. Resultados del Perceptrón en Google Colab

Temperatura (T)	Humedad (H)	CO (C)	Luz (L)	Esperada (S_{exp})	Predicha (S_{pred})
0	0	0	0	0	0
0	0	0	1	0	0
0	0	1	0	0	1
0	0	1	1	0	1
0	1	0	0	0	1
0	1	0	1	0	1
0	1	1	0	1	1
0	1	1	1	1	1
1	0	0	0	0	0
1	0	0	1	0	0
1	0	1	0	1	1
1	0	1	1	1	0
1	1	0	0	0	0
1	1	0	1	1	0
1	1	1	0	1	1
1	1	1	1	1	1

Cuadro 3. Métricas de Evaluación del Perceptrón

Métrica	Valor
Precisión	0.62
Exactitud	0.56
Recall	0.71
F1	0.63

5. CONCLUSIONES

El estudio del perceptrón para el control de sistemas de ventilación en invernaderos ha demostrado ser un punto de partida útil, pero no una solución definitiva. Los resultados obtenidos revelan varias conclusiones clave:

- **Limitaciones del Modelo:** El perceptrón entrenado no logró una alta precisión y exactitud en la clasificación de las condiciones ambientales. Las métricas de evaluación, incluyendo la precisión (0.62), exactitud (0.56), recall (0.71) y F1 (0.63), indican que el modelo no es suficientemente efectivo en la predicción de la necesidad de activar o desactivar el sistema de ventilación.
- **Necesidad de Métodos Avanzados:** Dado el desempeño subóptimo del perceptrón, es esencial explorar métodos más avanzados que puedan capturar mejor las complejidades de los datos y mejorar la precisión y exactitud de las predicciones. Esto incluye el uso de redes neuronales más complejas, técnicas de ensemble learning, o algoritmos de machine learning más robustos.
- **Recolección de Datos y Ajustes:** Además, mejorar la recolección de datos y ajustar los parámetros del modelo podrían contribuir a un mejor desempeño. La inclusión de más variables relevantes y un conjunto de datos más extenso y variado también podría aumentar la efectividad del modelo.

En conclusión, aunque el perceptrón ha servido como una herramienta introductoria para el control de ventilación en invernaderos, es necesario implementar y explorar modelos más avanzados para maximizar su efectividad y adaptabilidad en aplicaciones reales. La mejora continua y la innovación en los métodos utilizados serán cruciales para desarrollar soluciones automatizadas más eficientes para el cultivo de plantas.

APÉNDICES

Código en Excel

A continuación se presenta el código en Excel utilizado para calcular las salidas del perceptrón para diferentes combinaciones de entradas. La lógica del perceptrón se implementa mediante fórmulas y funciones de Excel.

Para cada combinación de entradas (Temperatura, Humedad, CO, Horas de Luz):

1. Definir los pesos para cada entrada.
2. Calcular la suma ponderada: Suma = (PesoTemperatura * Temperatura) + (PesoHumedad * Humedad) + (PesoCO2 * CO) + (PesoLuz * Horas de Luz)
3. Aplicar la función de activación (por ejemplo, umbral): Si Suma >Umbral entonces Salida = 1 Si no, Salida = 0
4. Ajustar los pesos a partir del error

1. 2. 3.

Código en Google Colab

A continuación se muestra el código Python utilizado en Google Colab para entrenar el perceptrón y evaluar su rendimiento.

Código 1. Codigo de Python.

```python
# Función de activación escalón
def escalon_unitario(x):
  return 1 if x >= 0 else 0

# Función del Perceptrón
def perceptron(datos_entrada, pesos, bias):
  # Producto entre la entrada y los pesos
  suma_entrada_pesos = np.dot(datos_entrada, pesos) + bias

  #Salida del perceptron
  output = escalon_unitario(suma_entrada_pesos)
  return output


# Entrenamiento del Perceptrón para la compuerta AND

def entrenamiento_perceptron():
  # Datos de (Entrada y Salida) en forma de arreglo
  datos_entrada = np.array([
    [0, 0, 0, 0],
    [0, 0, 0, 1],
    [0, 0, 1, 0],
    [0, 0, 1, 1],
```

```
    [0, 1, 0, 0],
    [0, 1, 0, 1],
    [0, 1, 1, 0],
    [0, 1, 1, 1],
    [1, 0, 0, 0],
    [1, 0, 0, 1],
    [1, 0, 1, 0],
    [1, 0, 1, 1],
    [1, 1, 0, 0],
    [1, 1, 0, 1],
    [1, 1, 1, 0],
    [1, 1, 1, 1]
])
  datos_salida = np.array([0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 1, 0, 1, 1, 1])

  # Inicialización de los pesos
  weights = np.random.rand(4)
  bias = np.random.rand()

  # Hiperparámetros de la red
  learning_rate = 0.001
  epochs = 100


  # ENTRENAMIENTO DEL PERCEPTRÓN
  for epoch in range(epochs):

    error_total = 0
    # Necesito un ciclo que me maneje los 2 arreglos (ENTRADA - SALIDA)
    for entrada, sdeseada in zip(datos_entrada, datos_salida):
      #SALIDA DEL PERCEPTRÓN PARA LAS ENTRADAS ACTUALES
      output = perceptron(entrada, weights, bias)

      # Calculamos el ERROR
      error = sdeseada - output

      # Actualizar los PESOS y BIAS
      weights = weights + (learning_rate * error)
      bias = bias + (learning_rate * error)

      # PARA FINES DE DAR SEGUIMIENTO AL ENTRENAMIENTO
      error_total = error_total + abs(error)
    # Termina el segundo FOR y me imprime el error total en cada época
    print(f"época {epoch + 1} - error total:{error_total}")

  return weights , bias

# Ejecutar el entrenamiento del perceptrón y obtener PESOS Y BIAS.

pesos_entrenamiento, bias_entrenamiento = entrenamiento_perceptron()
```

Contacto

Para cualquier duda favor de contactar:

marcosrafael2000@hotmail.com

+52 5537149673

Referencias

Control automático de la temperatura diurna en invernaderos mediante ventilación natural. (n.d.). *Interempresas*. Recuperado de <https://www.interempresas.net/Horticola/Articulos/346612-Control-automatico-de-la-temperatura-diurna-en-invernaderos-mediante-ventilacion-natural.html>

Scikit-learn developers. (2024). *MLPClassifier*. Recuperado de <https://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPClassifier.html>

TensorFlow developers. (2024). *TensorFlow: An end-to-end open source platform for machine learning*. Recuperado de <https://www.tensorflow.org/>

Jupyter Development Team. (2024). *Jupyter Notebooks*. Recuperado de <https://jupyter.org/>

Chollet, F. (2024). *Keras: The Python Deep Learning library*. Recuperado de <https://keras.io/>