Aplicaciones de la Regresión Logística para la Predicción de Ingresos Superiores a $25,000 Pesos: Un Análisis de Factores Demográficos y Laborales

Joseph Shakalo Paz A01784107, Miguel Ángel Noriega Bedolla A01658032 y Gabriel Edid Harari A01782146, estudiantes del curso *Inteligencia artificial avanzada para la ciencia de datos I* en el ITESM.

Resumen - Este trabajo compara tres redes asociativas, Hopfield, BAM y LAM, para reconocer dígitos binarios de 28×28. Se usaron protocolos 1/1, 2/2 y 4/4 (entrenamiento/prueba por dígito) con entradas completas sin ruido. BAM y LAM alcanzaron 1.000 de exactitud en 1/1 y 2/2, en 4/4 bajaron a 0.775 y 0.850, respectivamente. Hopfield obtuvo 0.200 (1/1), 0.100 (2/2) y 0.100 (4/4), mostrando colapso hacia pocos estados. Se concluyó que estas redes recuerdan bien patrones vistos y sirven para depurar ruido, pero generalizan poco ante muchas variantes. Se determinó que Hopfield es más útil como denoiser que como clasificador, mientras BAM y LAM responden con alta precisión cuando los patrones de prueba son iguales a los de entrenamiento.

**Índice de Términos - Clasificación binaria, *Cross-validation*, Ingresos, *Machine learning*, Preprocesamiento de datos, Python, Regresión logística.**

# introducción

La regresión logística es un modelo estadístico ampliamente usado en problemas de clasificación binaria. A diferencia de la regresión lineal, cuyo objetivo es predecir valores continuos, la regresión logística transforma la salida en una probabilidad entre 0 y 1 mediante la función sigmoide. Esto permite tomar decisiones al asignar una clase según un umbral definido, lo que la hace adecuada para problemas donde solo existen dos categorías de respuesta.

En este trabajo se utiliza la regresión logística para predecir si una persona percibe un sueldo mensual mayor a $25,000 pesos o si se encuentra en el grupo de ingresos iguales o menores. La información utilizada proviene de un formulario que recopila variables como edad, nivel educativo, tipo de empleo, estado civil, sector económico, horas trabajadas y otros datos demográficos. Estas variables sirven como insumo para entrenar y evaluar el modelo.

El flujo seguido incluye limpieza de datos, imputación de valores faltantes, codificación de variables categóricas mediante One-Hot Encoding, estandarización de variables numéricas y validación cruzada para garantizar un desempeño estable. También se calcula un umbral de decisión que maximiza la métrica F1, buscando un equilibrio entre precisión y exhaustividad en la clasificación.

El objetivo de este reporte es documentar de forma clara y reproducible cómo un modelo de regresión logística puede identificar patrones relacionados con el nivel de ingresos. Si bien no se busca una aplicación directa en producción, los resultados permiten visualizar su utilidad en contextos como estudios laborales, planeación educativa y análisis de equidad salarial.

# Marco teórico

En ejercicios previos de clasificación binaria, la regresión logística se había aplicado en escenarios sencillos, con pocas variables y sin procesos avanzados de limpieza o validación. Estos trabajos iniciales ayudaron a comprender la función sigmoide y la interpretación de coeficientes, pero no permitían evaluar el modelo en situaciones más complejas con múltiples predictores y datos heterogéneos.

En este proyecto se amplía ese enfoque aplicando regresión logística a un conjunto de datos con variables demográficas, educativas y laborales. El modelo estima la probabilidad de pertenecer a la clase positiva mediante la función logística:

donde 𝜃 es el vector de parámetros y 𝑋 el vector de características. La salida es una probabilidad entre 0 y 1 que se convierte en clase al fijar un umbral.

El flujo metodológico integra limpieza de datos, imputación de valores faltantes, One-Hot Encoding, estandarización y validación cruzada. Además, se ajusta un umbral de decisión que maximiza la métrica F1. Estos elementos hacen que el modelo sea más robusto y cercano a un escenario real.

Gracias a su simplicidad e interpretabilidad, la regresión logística sigue siendo un modelo de referencia en predicción de ingresos y en aplicaciones como riesgo crediticio, estudios médicos o análisis laboral.

# Metodología

Para llevar a cabo el experimento se desarrollaron implementaciones en Python de las tres arquitecturas: Hopfield, BAM y LAM. Cada modelo fue programado como un módulo independiente, empleando estructuras de matrices y operaciones vectoriales para representar los pesos y las actualizaciones.

Los patrones de entrada se generaron a partir de imágenes binarias de dígitos de 28x28 píxeles, transformadas en vectores de 784 elementos. Para todos los modelos se aplicaron tres protocolos idénticos por dígito: 1/1, 2/2 y 4/4 (entrenamiento/prueba). El procedimiento de conversión mapea cada píxel a un valor binario, permitiendo trabajar con representaciones uniformes y comparables en los tres modelos.

A number set on a white background

AI-generated content may be incorrect.

Fig. 1. Conjunto de dígitos manuscritos en formato binario de 28×28 píxeles, utilizados como ejemplos de patrones de entrada para el entrenamiento y prueba de las redes.

En la red de Hopfield, los pesos se calcularon mediante la regla de Hebb y las actualizaciones se realizaron de manera iterativa hasta alcanzar un estado estable. Después se utilizó un método de similaridad de vectores para comparar el resultado de la convergencia con el dígito más similar. En BAM se construyó una matriz de asociación entre dos conjuntos de patrones, de modo que al presentar un vector de entrada se mapea a un vector de dimensión del número de clases, con one hot encoding para representar la clase de la que se trata. Por último, la ALM se implementó como un mapeo lineal directo, en el que los pesos se obtuvieron a partir de correlaciones entre vectores de entrada y salida. De igual forma, los vectores de salida se componían de la misma forma que en BAM, con su dimensión igual al número de clases, en formato one hot encoding.

Para la evaluación se realizaron pruebas presentando a cada red patrones completos, es decir, imágenes distintas a las usadas en el entrenamiento, pero sin modificaciones ni ruido. Se verificó si cada modelo lograba identificar correctamente el dígito correspondiente. Este procedimiento permitió contrastar los modelos en condiciones equivalentes.

# Resultados

Se evaluaron tres protocolos idénticos para los modelos: 1/1 (10 pruebas), 2/2 (20 pruebas) y 4/4 (40 pruebas), correspondientes a la proporción entrenamiento/prueba por dígito. En todas las corridas se utilizaron dígitos binarios de 28×28 sin ruido.

**Tabla 1**

Exactitud por modelo y protocolo

|  |  |  |  |
| --- | --- | --- | --- |
| ***Modelo*** | ***1/1*** | ***2/2*** | ***4/4*** |
| *Hopfield* | 20% | 10% | 10% |
| *BAM* | 100% | 100% | 77.5% |
| *LAM* | 100% | 100% | 85% |

**Tabla 2**

Errores por dígito en 1/1 (conteos)

|  |  |
| --- | --- |
| ***Modelo*** | ***Real → Predicho (solo errores)*** |
| *Hopfield* | 0→9×1; 1→2×1; 3→9×1; 4→2×1; 5→9×1; 6→2×1; 7→9×1; 8→2×1 |
| *BAM* | — |
| *LAM* | — |

**Tabla 3**

Errores por dígito en 2/2 (conteos)

|  |  |
| --- | --- |
| ***Modelo*** | ***Real → Predicho (solo errores)*** |
| *Hopfield* | 0→2×2; 1→2×2; 3→2×2; 4→2×2; 5→2×2; 6→2×2; 7→2×2; 8→2×2; 9→2×2 |
| *BAM* | — |
| *LAM* | — |

**Tabla 4**

Errores por dígito en 4/4 (conteos)

|  |  |
| --- | --- |
| ***Modelo*** | ***Real → Predicho (solo errores)*** |
| *Hopfield* | 0→4×4; 1→4×4; 2→4×4; 3→4×4; 5→4×4; 6→4×4; 7→4×4; 8→4×4; 9→4×4 |
| *BAM* | 0→1×1, 0→3×1; 4→1×1; 5→3×1; 7→3×1; 9→3×2, 9→1×1, 9→7×1 |
| *LAM* | 0→6×1; 1→4×1, 1→6×1, 1→9×1, 1→8×1; 5→8×1 |

**Nota de lectura.** El formato **Real→Predicho×n** indica que el dígito Real se clasificó como Predicho **n** veces en el protocolo señalado. Un guion (—) implica **cero errores**. Si un dígito no aparece para un modelo, todas sus predicciones para ese dígito fueron correctas en ese protocolo.

Fig. 2. Del lado izquierdo, entradas a la red de *Hopfield* y del lado derecho, la salidas, es decir lo que la red logró reconocer para ese dígito con el protocolo 1/1.

Fig. 3. Del lado izquierdo, entradas a la red de *LAM* y del lado derecho, la salidas, es decir lo que la red logró reconocer para ese dígito con el protocolo 2/2.

Fig. 4. Del lado izquierdo, entradas a la red de *BAM* y del lado derecho, la salidas, es decir lo que la red logró reconocer para ese dígito con el protocolo 4/4. Ejemplo de cómo en una de las entradas si logró reconocer el 0, pero en otra entrada reconoció un 1.

# Análisis de resultados

En 1/1 y 2/2, BAM y LAM alcanzaron 100% de exactitud. Hopfield obtuvo 20% y 10%, respectivamente. En 4/4, el rendimiento bajó a 77.5% en BAM, 85% en LAM y 10% en Hopfield.

Por modelo, Hopfield predijo siempre 2 en 2/2 y siempre 4 en 4/4, lo que indica colapso hacia pocos estados finales; en 1/1 distribuyó errores entre 2 y 9. BAM no presentó errores en 1/1 y 2/2; en 4/4 mostró confusiones puntuales (0→1/3, 4→1, 5→3, 7→3, 9→3/1/7). LAM tampoco tuvo errores en 1/1 y 2/2; en 4/4 falló sobre todo en 0 y 1 (0→6; 1→4/6/9/8) y registró 5→8.

Con patrones completos y protocolos balanceados, LAM mantiene ligera ventaja sobre BAM en 4/4, y ambos superan ampliamente a Hopfield en todos los escenarios. Para robustecer la comparación, conviene repetir corridas con distintos muestreos por protocolo y reportar métricas por clase y matrices de confusión por protocolo; estas salidas se derivan directamente de los programas.

# Conclusiones

Las redes asociativas funcionan bien cuando el patrón de prueba coincide con uno visto en el entrenamiento, pero su desempeño cae al aumentar la variabilidad de ejemplos por dígito. En los protocolos 1/1 y 2/2, BAM y LAM alcanzaron exactitud perfecta; al pasar a 4/4, ambas disminuyeron y LAM conservó ligera ventaja. Esto indica una capacidad limitada para generalizar a múltiples instancias por clase bajo las configuraciones probadas.

Hopfield fue el modelo más sensible al incremento de ejemplos: mostró colapsos hacia pocos estados finales y baja exactitud en todos los protocolos. Esto es consistente con la interferencia entre patrones almacenados en su matriz de pesos, que provoca confusiones cuando crece el número de asociaciones que deben coexistir.

Estas arquitecturas son útiles para recordar patrones concretos y para tareas donde se busca recuperar exactamente lo aprendido. En ese contexto, BAM y LAM responden con alta precisión cuando el patrón de prueba es igual al de entrenamiento. Sin embargo, cuando los datos cambian (nuevas instancias del mismo dígito), la exactitud se degrada.

En particular, Hopfield es más adecuado como filtro de ruido que como clasificador de dígitos. Al actualizar sus estados, la red tiende a llevar una imagen ruidosa hacia el patrón almacenado más cercano, lo que ayuda a limpiar la entrada. Sin embargo, cuando hay muchas variantes del mismo dígito, las asociaciones en la matriz de pesos interfieren entre sí y la red confunde clases, por lo que su poder de predicción es limitado.

En síntesis, las redes asociativas sirven para recordar patrones específicos y depurar ruido cercano a lo aprendido, pero no para generalizar a muchas variantes. Para escenarios con alta variabilidad, conviene aplicar preprocesamiento que separe mejor las clases o usar clasificadores supervisados de mayor capacidad. Se podría utilizar Hopfield como *denoiser* y BAM/LAM como recuperadores exactos en casos controlados.

Apéndice A. Código fuente y material suplementario

El código completo de las implementaciones (Hopfield, BAM y LAM), junto con utilidades de preprocesamiento, está disponible en:

<https://drive.google.com/file/d/1lqyOCGPypJ95kNGoGjWS4eqMvAdiL-KN/view?usp=sharing>

Accedido: 15-sep-2025.

Referencias

1. GeeksforGeeks, “Logistic Regression in Machine Learning,” GeeksforGeeks, 2024. [En línea]. Disponible en: https://www.geeksforgeeks.org/machine-learning/understanding-logistic-regression/