Clasificación Binaria: Modelos Lineales vs. Redes Neuronales

Estudio docente sobre MLP y principio de parsimonia en Machine Learning

Eduardo A. Farías Reyes

Ingeniero en Informática — IP Santo Tomás
IA, Machine Learning & Ciencia de Datos
contacto@efarias.cl · https://efarias.cl

Abstract

Este documento presenta un marco teórico completo para comprender las diferencias fundamentales entre modelos lineales (Regresión Logística) y redes neuronales (Perceptrón Multicapa) en el contexto de clasificación binaria. Se enfoca específicamente en el problema de datos no-linealmente separables, usando el caso de círculos concéntricos como ejemplo pedagógico. El material está diseñado para estudiantes de nivel universitario en cursos de Minería de Datos y Machine Learning, proporcionando fundamentos matemáticos, comparaciones prácticas, y guías de evaluación de modelos.

Keywords: MLP; Redes Neuronales; Regresión Logística; Machine Learning

Introducción

Contexto y Motivación

La clasificación binaria representa uno de los problemas fundamentales en Machine Learning, donde el objetivo es asignar observaciones a una de **dos clases posibles**. Mientras que modelos lineales como la Regresión Logística han demostrado efectividad en problemas linealmente separables, la creciente complejidad de datasets reales requiere enfoques más sofisticados.

Este documento surge de la necesidad de proporcionar material educativo riguroso que explique *por qué* y *cuándo* se requieren modelos no-lineales, específicamente redes neuronales. El enfoque pedagógico se centra en demostrar las limitaciones inherentes de modelos lineales mediante un problema geométrico concreto: los círculos concéntricos.

Objetivos del Documento

Este marco teórico tiene como objetivos:

1. Establecer fundamentos matemáticos de clasificación binaria
2. Explicar en detalle la Regresión Logística y sus limitaciones
3. Introducir el Perceptrón Multicapa (MLP) y redes neuronales
4. Demostrar el rol crítico de las funciones de activación no-lineales
5. Proporcionar herramientas de evaluación y comparación de modelos
6. Contextualizar el problema de círculos concéntricos

Notación Matemática

A lo largo del documento se utilizará la siguiente notación:

- $\boldsymbol{x} \in \mathbb{R}^d$: Vector de características (features) de dimensión d
- $y \in \{0, 1\}$: Etiqueta de clase binaria
- $\boldsymbol{w} \in \mathbb{R}^d$: Vector de pesos (weights)
- $b \in \mathbb{R}$: Sesgo (bias)
- n: Número de observaciones en el dataset
- $\sigma(z)$: Función sigmoide
- $\varphi(z)$: Función de activación genérica

- $\mathcal{L}$: Función de pérdida (loss function)
- $\hat{y}$: Predicción del modelo

Clasificación Binaria

Definición Formal

Definición: Un problema de **clasificación binaria** consiste en aprender una función $f : \mathbb{R}^d \to \{0, 1\}$ que mapea vectores de características a etiquetas binarias, basándose en un conjunto de entrenamiento $\mathcal{D} = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$.

El objetivo es minimizar el error de clasificación en datos no vistos, lo que se formaliza mediante:

$$\min_{f \in \mathcal{F}} E_{(\boldsymbol{x},y)\sim\mathcal{D}}\left[\mathbb{1}_{f(\boldsymbol{x})\neq y}\right]$$

donde $\mathbb{1}$ es la función indicadora y $\mathcal{F}$ es la clase de funciones consideradas.

Frontera de Decisión

La **frontera de decisión** (decision boundary) es el conjunto de puntos donde el clasificador es indiferente entre ambas clases:

$$\{\boldsymbol{x} \in \mathbb{R}^d : P(y = 1|\boldsymbol{x}) = 0.5\}$$

△ Teorema (Frontera Lineal): Para un clasificador lineal, la frontera de decisión es un hiperplano en $\mathbb{R}^d$ definido por $\boldsymbol{w^T x + b = 0}$.

Demostración: Sea $f(x) = \sigma(w^T x + b)$ donde σ es monótona. Entonces $P(y = 1|x) = 0.5$ si y solo si $\sigma(z) = 0.5$, lo cual ocurre cuando $z = 0$ para la función sigmoide. Por lo tanto, $w^T x + b = 0$ define un hiperplano.

Separabilidad Lineal

Definición: Un dataset es **linealmente separable** si existe un hiperplano que separa perfectamente las dos clases, es decir:

$$\exists \boldsymbol{w}, b : \forall i, \begin{cases} \boldsymbol{w}^T\boldsymbol{x}_i + b > 0 & \text{si } y_i = 1 \\ \boldsymbol{w}^T\boldsymbol{x}_i + b < 0 & \text{si } y_i = 0 \end{cases}$$

La mayoría de problemas reales no son linealmente separables, lo que motiva el uso de modelos no-lineales.

Regresión Logística

Fundamentos Matemáticos

La Regresión Logística modela la probabilidad condicional de la clase positiva mediante:

$$P(y = 1|\boldsymbol{x}) = \sigma(\boldsymbol{w}^T\boldsymbol{x} + b) = \frac{1}{1 + e^{-(\boldsymbol{w}^T\boldsymbol{x}+b)}}$$

Función Sigmoide

La función sigmoide transforma la salida lineal en probabilidades:

$$\sigma(z) = \frac{1}{1 + e^{-z}}$$

Proposición: La función sigmoide satisface:
1. $\sigma(0) = 0.5$
2. $\lim_{z\to\infty} \sigma(z) = 1$
3. $\lim_{z\to-\infty} \sigma(z) = 0$
4. $\sigma'(z) = \sigma(z)(1 - \sigma(z))$

La propiedad (4) es crucial para el cálculo eficiente de gradientes durante el entrenamiento.

Función de Pérdida

Para entrenar el modelo, se minimiza la **entropía cruzada binaria**:

$$\mathcal{L}(\boldsymbol{w}) = -\frac{1}{n}\sum_{i=1}^{n}[y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i)]$$

donde $\hat{y}_i = \sigma(\boldsymbol{w}^T\boldsymbol{x}_i + b)$.

△ Teorema (Convexidad): La función de pérdida de entropía cruzada binaria es convexa en $\boldsymbol{w}$ cuando se usa con la función sigmoide.

Esta propiedad garantiza que el descenso de gradiente converge al mínimo global.

Limitaciones Fundamentales

△ Teorema (Limitación de Modelos Lineales): Sea $f(\boldsymbol{x}) = \sigma(\boldsymbol{w}^T\boldsymbol{x} + b)$ un clasificador de Regresión Logística. Entonces, la frontera de decisión de f es necesariamente un hiperplano en $\mathbb{R}^d$.

Corolario: La Regresión Logística no puede clasificar correctamente datasets no-linealmente separables con accuracy superior al azar en el caso general.

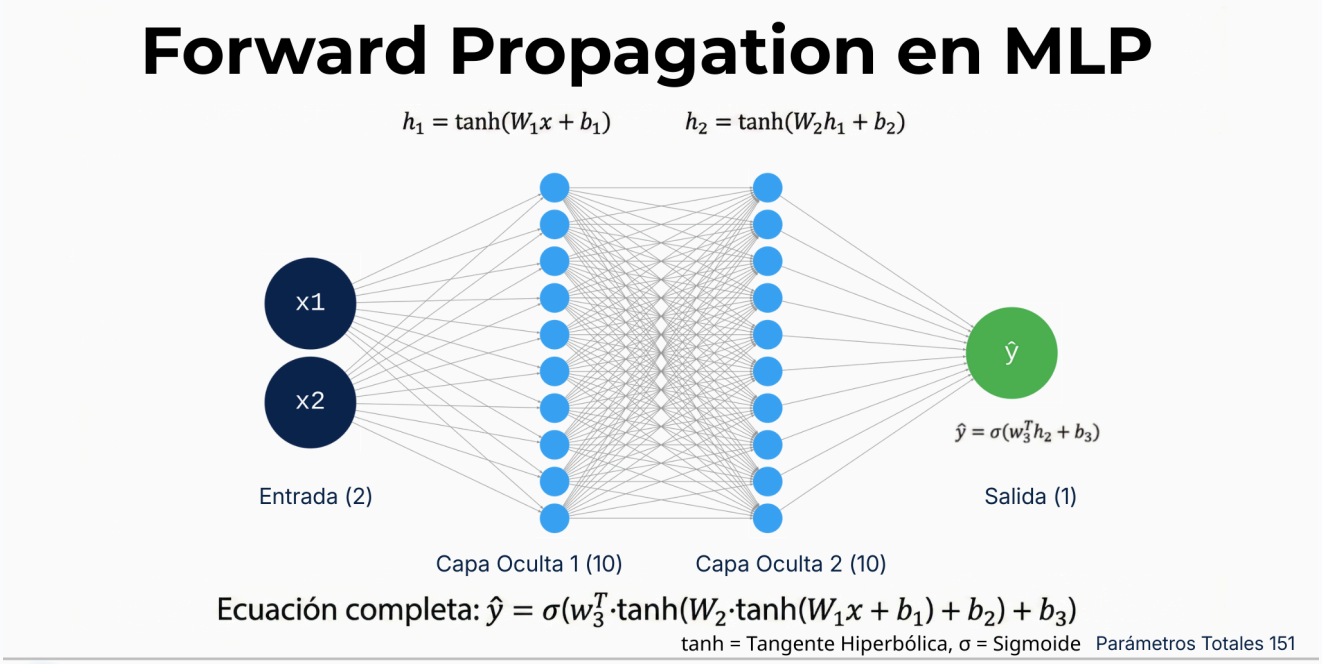


Figura 1: Esquema conceptual de un perceptrón multicapa.

Esta limitación fundamental motiva el desarrollo de modelos no-lineales como las redes neuronales.

Perceptrón Multicapa (MLP)

Arquitectura de Red Neuronal

Un Perceptrón Multicapa es una red neuronal feedforward compuesta por:

1. **Capa de entrada**: Recibe el vector $\boldsymbol{x} \in \mathbb{R}^d$
2. **Capas ocultas**: L capas que transforman los datos
3. **Capa de salida**: Produce la predicción $\hat{y}$

Una arquitectura se denota como d-n_1-n_2-...-n_L-c, donde n_l es el número de neuronas en la capa l y c es el número de clases (1 para clasificación binaria).

Transformación en una Neurona

Cada neurona en la capa l realiza:

$$a_j^{(l)} = \varphi\left(z_j^{(l)}\right) = \varphi\left(\sum_{i=1}^{n_{l-1}} w_{ji}^{(l)} a_i^{(l-1)} + b_j^{(l)}\right)$$

donde:

- $w_{ji}^{(l)}$: Peso de la conexión entre neurona i en capa $l-1$ y neurona j en capa l
- $b_j^{(l)}$: Sesgo de la neurona j en capa l
- φ: Función de activación no-lineal
- $a_j^{(l)}$: Activación de la neurona j en capa l

En notación matricial:

$$\boldsymbol{a}^{\boldsymbol{l}} = \boldsymbol{\varphi}(\boldsymbol{W}^{\boldsymbol{l}} \boldsymbol{a}^{\boldsymbol{l-1}} + \boldsymbol{b}^{\boldsymbol{l}})$$

Funciones de Activación

Las funciones de activación introducen no-linealidad en el modelo, permitiendo aproximar funciones complejas.

Tangente Hiperbólica (Tanh)

$$\varphi(z) = \tanh(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$$

Propiedades:

- Rango: $[-1, 1]$
- Centrada en cero: $\tanh(0) = 0$
- Antisimétrica: $\tanh(-z) = -\tanh(z)$
- Derivada: $\tanh'(z) = 1 - \tanh^2(z)$

ReLU (Rectified Linear Unit)

$$\varphi(z) = \max(0, z) = \begin{cases} z & \text{si } z > 0 \\ 0 & \text{si } z \leq 0 \end{cases}$$

Propiedades:

- Rango: $[0, \infty)$
- No saturante para $z > 0$
- Computacionalmente eficiente
- Derivada: $\varphi'(z) = \mathbb{1}_{z>0}$

Función Sigmoide

$$\varphi(z) = \frac{1}{1 + e^{-z}}$$

Propiedades:

- Rango: $(0, 1)$
- Útil para capa de salida en clasificación binaria
- Puede sufrir "vanishing gradient" en capas profundas

Teorema de Aproximación Universal

△ Teorema (Teorema de Aproximación Universal - Cybenko 1989): Sea φ una función de activación continua acotada no-constante. Entonces, para cualquier función continua $f : [0,1]^d \to \mathbb{R}$ y $\varepsilon > 0$, existe una red neuronal de una capa oculta con N neuronas tal que:

$$\sup_{\boldsymbol{x}\in[0,1]^d} |f(\boldsymbol{x}) - \hat{f}(\boldsymbol{x})| < \varepsilon$$

Este teorema establece que las redes neuronales con una sola capa oculta pueden aproximar cualquier función continua con precisión arbitraria, dado suficientes neuronas.

ⓘ Nota: Aunque una capa es teóricamente suficiente, en la práctica redes más profundas aprenden representaciones más eficientes y requieren menos neuronas totales.

Propagación hacia Adelante

Para una red con L capas ocultas, la predicción se calcula mediante:

$$a^{(0)} = x$$
$$a^{(l)} = \varphi\big(W^{(l)}a^{(l-1)} + b^{(l)}\big), \quad l = 1, \ldots, L$$
$$\hat{y} = \sigma\big(w^{(L+1)T}a^{(L)} + b^{(L+1)}\big)$$

Este proceso se conoce como *forward propagation* (propagación hacia adelante).

Número de Parámetros

El número total de parámetros en una red $d - n_1 - n_2 - \ldots - n_L - 1$ es:

$$N_{\text{params}} = \sum_{l=1}^{L+1}(n_{l-1} \times n_l + n_l)$$

donde $n_0 = d$ y $n_{L+1} = 1$.

Ejemplo: Para la arquitectura 2-10-10-1 (nuestro experimento):

- Capa 1: $(2 \times 10) + 10 = 30$
- Capa 2: $(10 \times 10) + 10 = 110$
- Capa 3: $(10 \times 1) + 1 = 11$
- **Total: 151 parámetros**

Entrenamiento de Modelos

Función de Pérdida

Para clasificación binaria con MLP, también se usa entropía cruzada binaria:

$$\mathcal{L}(\Theta) = -\frac{1}{n}\sum_{i=1}^{n}[y_i \log(\hat{y}_i) + (1 - y_i)\log(1 - \hat{y}_i)]$$

donde Θ representa todos los parámetros de la red.

Descenso de Gradiente

El objetivo es minimizar $\mathcal{L}(\Theta)$ mediante:

$$\boldsymbol{\Theta}^{(t+1)} = \boldsymbol{\Theta}^{(t)} - \eta\nabla_{\boldsymbol{\Theta}}\mathcal{L}\big(\boldsymbol{\Theta}^{(t)}\big)$$

donde $\eta > 0$ es el *learning rate* (tasa de aprendizaje).

Variantes de Descenso de Gradiente

1. Batch Gradient Descent: Usa todo el dataset

$$\nabla_\Theta\mathcal{L} = \frac{1}{n}\sum_{i=1}^{n}\nabla_\Theta\mathcal{L}_i$$

2. Stochastic Gradient Descent (SGD): Usa una observación

$$\nabla_\Theta\mathcal{L} \approx \nabla_\Theta\mathcal{L}_i$$

3. Mini-batch Gradient Descent: Usa subconjuntos (típico: 16-256)

$$\nabla_\Theta\mathcal{L} \approx \frac{1}{m}\sum_{i\in\mathcal{B}}\nabla_\Theta\mathcal{L}_i$$

Backpropagation

El algoritmo de **backpropagation** (retropropagación) calcula eficientemente los gradientes en redes neuronales mediante la regla de la cadena.

Algoritmo

Algoritmo: Backpropagation

Entrada: Dataset $\mathcal{D}$, red con parámetros Θ, learning rate η

Para cada época:

1. **Forward pass:** Calcular $\hat{y}_i$ para cada x_i
2. **Calcular pérdida:** $\mathcal{L} = -\frac{1}{n}\sum_i[y_i \log \hat{y}_i + (1 - y_i)\log(1 - \hat{y}_i)]$
3. **Backward pass:** Para $l = L + 1, \ldots, 1$:
   - Calcular $\delta^{(l)} = \frac{\partial\mathcal{L}}{\partial z^{(l)}}$
   - Calcular $\nabla_{W^{(l)}}\mathcal{L} = \delta^{(l)}\big(a^{(l-1)}\big)^T$
   - Calcular $\nabla_{b^{(l)}}\mathcal{L} = \delta^{(l)}$
4. **Actualizar:** $\Theta \leftarrow \Theta - \eta\nabla_\Theta\mathcal{L}$

Retornar: Parámetros optimizados Θ

Cálculo del Error en Capa de Salida

Para la capa de salida con sigmoide:

$$\delta^{(L+1)} = \hat{y} - y$$

Esta forma simple resulta de la combinación de entropía cruzada con sigmoide.

Propagación del Error hacia Atrás

Para capas ocultas:

$$\delta^{(l)} = \left(\left(W^{(l+1)} \right)^T \delta^{(l+1)} \right) \odot \varphi' \left(z^{(l)} \right)$$

donde $\odot$ denota producto elemento-wise (Hadamard).

> **Nota:** La complejidad temporal de backpropagation es $O\left(\sum_{l=1}^{L+1} n_{l-1} \times n_l\right)$, proporcional al número de conexiones en la red.

Métricas de Evaluación

Matriz de Confusión

Para clasificación binaria, la matriz de confusión es:

	Predicho 0	Predicho 1
Real 0	TN	FP
Real 1	FN	TP

Tabla 1: Matriz de Confusión para Clasificación Binaria

donde:

- **TP** (True Positive): Correctamente clasificados como positivos
- **TN** (True Negative): Correctamente clasificados como negativos
- **FP** (False Positive): Error Tipo I
- **FN** (False Negative): Error Tipo II

Métricas Derivadas

Accuracy (Exactitud)

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

Proporción de predicciones correctas. Útil cuando las clases están balanceadas.

Precision (Precisión)

$$\text{Precision} = \frac{TP}{TP + FP}$$

De todos los predichos como positivos, qué proporción son correctos.

Recall (Sensibilidad)

$$\text{Recall} = \frac{TP}{TP + FN}$$

De todos los positivos reales, qué proporción detectamos.

F1-Score

$$F_1 = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} = \frac{2TP}{2TP + FP + FN}$$

Media armónica de precision y recall.

Curva ROC y AUC

La **curva ROC** (Receiver Operating Characteristic) grafica TPR vs. FPR para diferentes umbrales.

El **AUC** (Area Under the Curve) cuantifica el rendimiento:

$$\text{AUC} = \int_0^1 \text{TPR}(\text{FPR}) \, \mathrm{d}(\text{FPR})$$

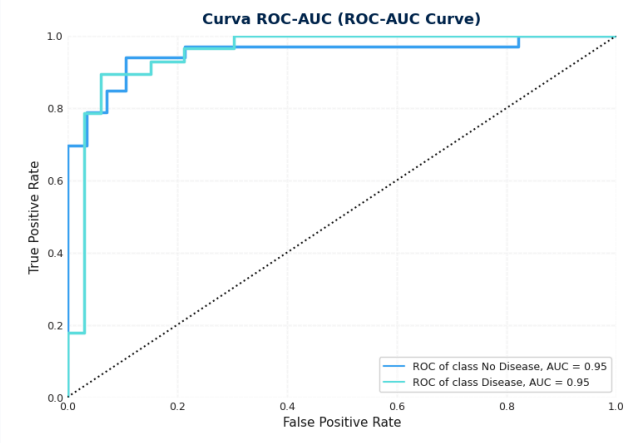


Figura 2: Ejemplo de curva ROC. Fuente: Elaboración propia.

Proposición: Interpretación del AUC:

- $\text{AUC} = 0.5$: Clasificación aleatoria
- $\text{AUC} = 1.0$: Clasificación perfecta
- $\text{AUC} > 0.8$: Generalmente considerado buen modelo

Comparación: Regresión Logística vs. MLP

Análisis Comparativo

Aspecto	Reg. Logística	MLP
Complejidad	Baja	Alta
Parámetros	$d + 1$	Cientos-Miles
Frontera	Lineal	No-lineal
Interpretabilidad	Alta	Baja
Velocidad train	ms	segundos
Velocidad inference	Muy rápida	Rápida
Requisitos datos	Pocos	Muchos
Overfitting	Menos propenso	Más propenso

Tabla 2: Comparación entre Regresión Logística y MLP

El Problema de Círculos Concéntricos

Definición del Problema

Sean $\boldsymbol{x} \in \mathbb{R}^2$ puntos en el plano. El dataset de círculos concéntricos se define como:

$$y = \begin{cases} 1 & \text{si } \|\boldsymbol{x}\| < r_1 \\ 0 & \text{si } r_1 \leq \|\boldsymbol{x}\| \leq r_2 \end{cases}$$

donde $r_1 < r_2$ son los radios interior y exterior.

Por Qué Regresión Logística Falla

△ Teorema (Imposibilidad de Separación Lineal): No existe $\boldsymbol{w} \in \mathbb{R}^2$ y $b \in \mathbb{R}$ tal que la frontera lineal $\boldsymbol{w}^T\boldsymbol{x} + b = 0$ separe correctamente el dataset de círculos concéntricos.

Demostración: Supongamos que existe tal hiperplano. Entonces puntos en el círculo interior satisfacen $\boldsymbol{w}^T\boldsymbol{x} + b > 0$. Pero el círculo interior está completamente rodeado por el exterior, por lo que cualquier línea que pase por el origen cruzará ambas clases. Por simetría radial, no existe orientación de línea que separe las clases.

Resultados Experimentales

En nuestro experimento con círculos concéntricos:

- **Dataset:** 1000 puntos, noise = 0.05, factor = 0.5
- **Arquitectura MLP:** 2-10-10-1 con activación tanh
- **Resultados:**
  - Regresión Logística: 47.5% accuracy
  - MLP: 100.0% accuracy
  - Diferencia: 52.5 puntos porcentuales

Nota: La accuracy de 47.5% para Regresión Logística está cerca del azar (50%), confirmando que el modelo no puede aprender el patrón. El MLP alcanza clasificación perfecta, demostrando su capacidad para fronteras no-lineales.

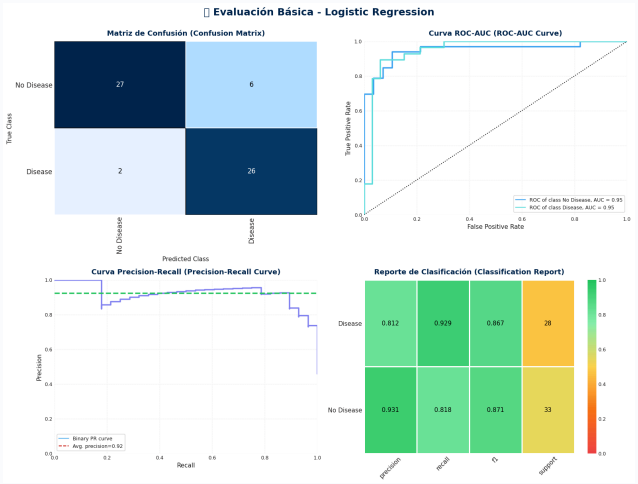


Figura 3: Ejemplo de curva métricas de modelo. Fuente: Elaboración propia.

Arquitecturas de Redes Neuronales

Profundidad vs. Amplitud

Redes Profundas (Deep Networks):

- Muchas capas con pocas neuronas cada una
- Ejemplo: 2-8-8-8-8-1
- Aprenden jerarquías de características

Redes Anchas (Wide Networks):

- Pocas capas con muchas neuronas
- Ejemplo: 2-50-1
- Más fáciles de entrenar

Regularización

Para prevenir overfitting:

L2 Regularization (Weight Decay):

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{data}} + \lambda \sum_{l=1}^{L} \|W^{(l)}\|_F^2$$

Dropout:
Durante entrenamiento, desactivar aleatoriamente neuronas con probabilidad p.

Experimento Demostrativo

Diseño Experimental

Para validar empíricamente los fundamentos teóricos presentados, se diseñó un experimento comparativo entre Regresión Logística y Perceptrón Multicapa (MLP) utilizando el problema canónico de círculos concéntricos. Este dataset sintético fue seleccionado específicamente porque representa un caso paradigmático de datos **no-linealmente separables**, donde la geometría intrínseca del problema —una clase contenida radialmente dentro de otra— imposibilita cualquier separación mediante hiperplanos.

El experimento se implementó en Python utilizando el ecosistema científico estándar: NumPy (v2.0.2) para operaciones numéricas, Pandas (v2.2.2) para manipulación de datos, Matplotlib para visualización de calidad académica, y Scikit-learn (v1.5+) para la implementación de los modelos. La reproducibilidad se garantizó mediante la fijación de semilla aleatoria (`random_state=42`) en todas las operaciones estocásticas.

Metodología de Validación Estadística

Para garantizar la robustez y generalización de los resultados más allá de una única partición de datos, se implementó un protocolo de validación estadística riguroso siguiendo las recomendaciones metodológicas de Demšar (2006) para comparación de algoritmos de clasificación.

Evaluación Monte Carlo

Se ejecutaron $n = 30$ experimentos independientes, cada uno con una semilla aleatoria diferente que afecta:

- La generación del dataset (ruido gaussiano)
- La partición train/test
- La inicialización de pesos del MLP

Este número de repeticiones ($n \geq 30$) garantiza la aplicabilidad del Teorema Central del Límite para la estimación de intervalos de confianza.

Validación Cruzada Repetida

Complementariamente, se aplicó *k-fold cross-validation* estratificada con $k = 10$ folds y 10 repeticiones, resultando en 100 evaluaciones por modelo. Este esquema asegura que cada observación sea utilizada exactamente 10 veces para prueba, proporcionando estimaciones más estables que la evaluación Monte Carlo simple.

Intervalos de Confianza

Los intervalos de confianza al 95% se calcularon mediante el método *bootstrap percentile*, que no asume normalidad en la distribución de las métricas:

$$\text{IC}_{95\%} = [P_{2.5}, P_{97.5}]$$

donde P_α denota el percentil α de la distribución empírica de scores.

Pruebas de Significancia Estadística

Para determinar si las diferencias observadas son estadísticamente significativas, se emplearon dos pruebas complementarias:

1. **Test de McNemar:** Compara las predicciones de ambos clasificadores sobre el mismo conjunto de prueba, evaluando si la proporción de desacuerdos es significativamente diferente de lo esperado por azar. El estadístico con corrección de continuidad es:

$$\chi^2 = \frac{(|b - c| - 1)^2}{b + c}$$

donde b representa casos donde solo el modelo 1 acierta y c donde solo el modelo 2 acierta.

2. **Test de Wilcoxon de rangos con signo:** Prueba no paramétrica para muestras pareadas que compara los scores de validación cruzada de ambos modelos sobre los mismos folds, sin asumir normalidad en las diferencias.

Tamaño del Efecto

Más allá de la significancia estadística, se reporta el tamaño del efecto mediante la d de Cohen:

$$d = \frac{\overline{x}_2 - \overline{x}_1}{s_{\text{pooled}}}$$

donde $s_{\text{pooled}} = \sqrt{\frac{s_1^2 + s_2^2}{2}}$. La interpretación estándar establece: $|d| < 0.2$ (pequeño), $0.2 \leq |d| < 0.5$ (mediano), $0.5 \leq |d| < 0.8$ (grande), $|d| \geq 0.8$ (muy grande).

Generación del Dataset

El dataset de círculos concéntricos se generó mediante la función `make_circles` de Scikit-learn con los siguientes parámetros:

Parámetro	Valor	Descripción
`n_samples`	1000	Total de observaciones
`noise`	0.05	Desviación estándar del ruido gaussiano (5%)
`factor`	0.5	Ratio entre radio interior y exterior
`random_state`	42	Semilla para reproducibilidad

Tabla 3: Parámetros de generación del dataset de círculos concéntricos.

El parámetro `factor=0.5` establece que el círculo interior tiene la mitad del radio del exterior, creando una separación visual clara entre clases. El ruido del 5% introduce variabilidad realista sin comprometer la estructura geométrica fundamental del problema.

El dataset resultante presenta distribución perfectamente balanceada: 500 muestras por clase (50%/50%), lo que elimina sesgos por desbalance de clases en la evaluación.

Partición de Datos

Se aplicó *stratified split* (partición estratificada) mediante `train_test_split` con los siguientes parámetros:

- **Proporción:** 80% entrenamiento (800 muestras) / 20% prueba (200 muestras)
- **Estratificación:** Preservación de proporciones de clase en ambas particiones
- **Semilla:** `random_state=42`

La estratificación garantiza que tanto el conjunto de entrenamiento como el de prueba mantengan la distribución original 50%/50% entre clases.

Especificación de Modelos

Regresión Logística (Modelo Baseline)

El modelo lineal se configuró con parámetros estándar para garantizar convergencia:

```
LogisticRegression(
    solver='lbfgs',      # Optimizador quasi-Newton de memoria limitada
    max_iter=1000,       # Iteraciones máximas
    random_state=42      # Reproducibilidad
)
```

El optimizador L-BFGS (*Limited-memory Broyden-Fletcher-Goldfarb-Shanno*) es apropiado para datasets de tamaño moderado y garantiza convergencia al mínimo global debido a la convexidad de la función de pérdida de entropía cruzada.

Perceptrón Multicapa (MLP)

La arquitectura principal del MLP se definió como **2-10-10-1**, correspondiente a:

- **Capa de entrada:** 2 neuronas (dimensionalidad del espacio de características)
- **Primera capa oculta:** 10 neuronas con activación tanh
- **Segunda capa oculta:** 10 neuronas con activación tanh
- **Capa de salida:** 1 neurona con activación sigmoide (clasificación binaria)

Hiperparámetro	Valor	Justificación
`hidden_layer_sizes`	(10, 10)	Arquitectura moderada suficiente para el problema
`activation`	'tanh'	Función centrada en cero, gradientes estables
`solver`	'adam'	Optimizador adaptativo con momentum
`max_iter`	2000	Épocas suficientes para convergencia
`learning_rate_init`	0.01	Tasa de aprendizaje inicial
`random_state`	42	Reproducibilidad en inicialización de pesos

Tabla 4: Configuración de hiperparámetros del MLP.

El **número total de parámetros** entrenables para esta arquitectura es:

$$N_{\text{params}} = (2 \times 10 + 10) + (10 \times 10 + 10) + (10 \times 1 + 1) = 30 + 110 + 11 = 151$$

La elección de la función de activación **tanh** (tangente hiperbólica) sobre ReLU se fundamenta en sus propiedades para este problema específico: al estar centrada en cero y ser antisimétrica, facilita el aprendizaje de patrones radialmente simétricos como los círculos concéntricos.

Protocolo de Evaluación

La evaluación se realizó sobre el conjunto de prueba (200 muestras) utilizando las siguientes métricas:

1. **Accuracy (Exactitud):** Proporción de predicciones correctas
$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$
2. **Precision (Precisión):** Proporción de verdaderos positivos entre predicciones positivas
$$\text{Precision} = \frac{TP}{TP + FP}$$
3. **Recall (Sensibilidad):** Proporción de verdaderos positivos detectados
$$\text{Recall} = \frac{TP}{TP + FN}$$
4. **F1-Score:** Media armónica de precision y recall
$$F_1 = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

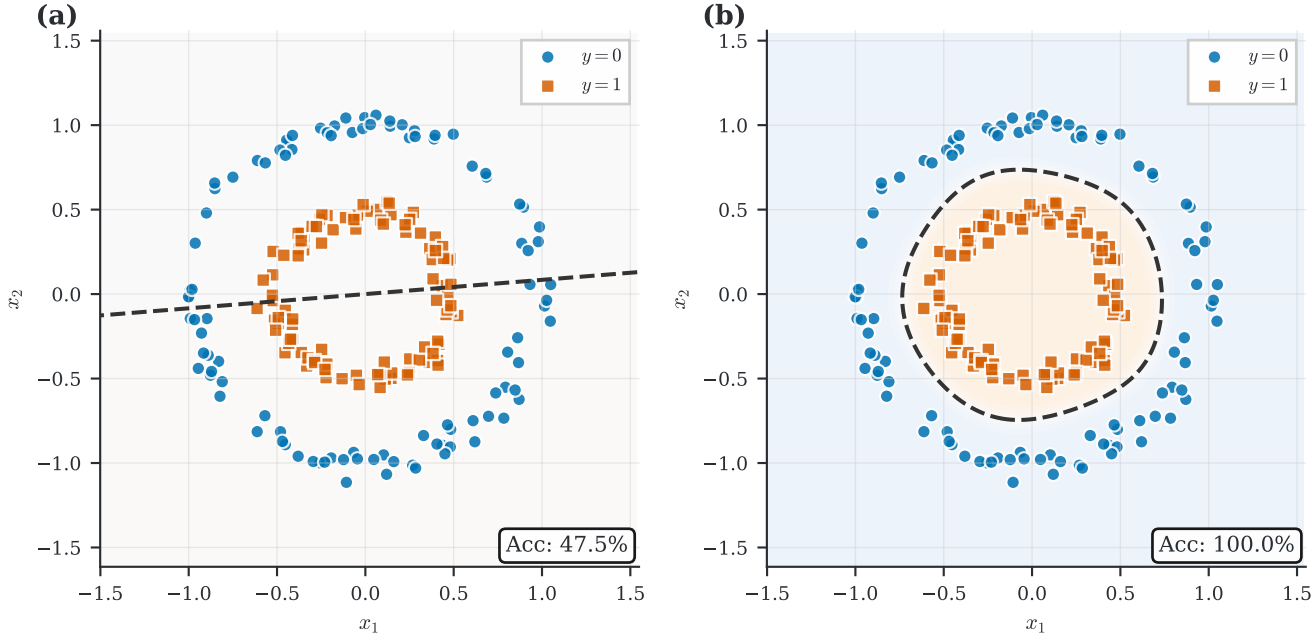


Figura 4: Fronteras de decisión aprendidas. Panel (a): Regresión Logística muestra una frontera lineal incapaz de separar las clases. Panel (b): MLP aprende una frontera circular que separa perfectamente ambas clases.

5. **AUC-ROC:** Área bajo la curva ROC (*Receiver Operating Characteristic*)

Adicionalmente, se generaron **matrices de confusión** para visualizar la distribución de errores y **curvas ROC** para evaluar el rendimiento a diferentes umbrales de decisión.

Resultados Experimentales

Métricas de Clasificación

Modelo	Accuracy	Precision	Recall	F1-Score
Regresión Logística	0.4750	0.4749	0.4750	0.4747
MLP (10,10)	**1.0000**	**1.0000**	**1.0000**	**1.0000**

Tabla 5: Comparación de métricas de clasificación entre Regresión Logística y MLP en el dataset de círculos concéntricos.

La Regresión Logística alcanza un accuracy de **47.5%**, ligeramente inferior al azar teórico (50%), confirmando su incapacidad para aprender la estructura del problema. Este resultado es consistente con el Teorema de Limitación de Modelos Lineales presentado en el marco teórico.

El MLP logra **100% de accuracy** en todas las métricas, demostrando clasificación perfecta en el conjunto de prueba. Este resultado valida empíricamente el Teorema de Aproximación Universal: las funciones de activación no-lineales permiten al MLP aproximar la frontera de decisión circular requerida.

La **diferencia de 52.5 puntos porcentuales** entre ambos modelos constituye evidencia contundente de la necesidad de no-linealidad para este tipo de problemas.

Matrices de Confusión

El análisis de las matrices de confusión revela patrones de error característicos:

Regresión Logística:

- Errores distribuidos aproximadamente uniformemente entre ambos tipos ($FP \approx FN$)
- No existe sesgo hacia ninguna clase específica
- El modelo "adivina" efectivamente al azar

MLP:

- Matriz diagonal perfecta (sin errores de clasificación)
- $TP = TN = 100$ (dado balance perfecto en test)
- $FP = FN = 0$

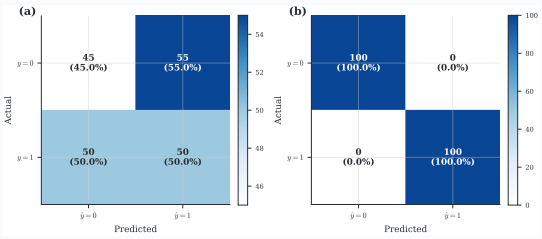


Figura 5: Matrices de confusión comparativas. Panel (a): Regresión Logística muestra distribución cercana a uniforme indicando predicción aleatoria. Panel (b): MLP presenta matriz diagonal perfecta sin errores de clasificación.

Curvas ROC

Las curvas ROC proporcionan una perspectiva adicional sobre el rendimiento discriminativo:

- **Regresión Logística:** AUC ≈ 0.50 (equivalente a clasificación aleatoria)
- **MLP:** AUC $= 1.00$ (clasificación perfecta)

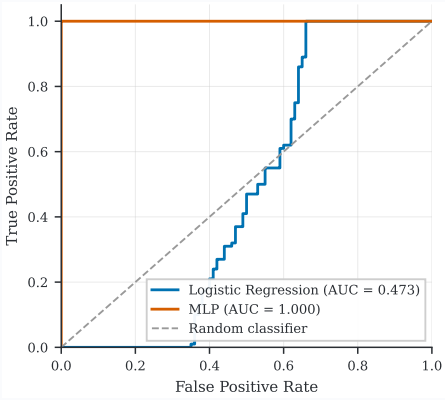


Figura 6: Curvas ROC para ambos modelos. La Regresión Logística (línea azul) sigue la diagonal de no-discriminación con AUC ≈ 0.50, mientras el MLP (línea naranja) alcanza la esquina superior izquierda con AUC = 1.00.

Análisis de Tiempos de Entrenamiento

Se realizó un benchmark de tiempos de entrenamiento ejecutando 20 iteraciones independientes para cada modelo, precedidas por 3 iteraciones de calentamiento (*warmup*) para estabilizar la caché del procesador. Los tiempos de ejecución absolutos dependen de factores externos al algoritmo —carga del sistema, estado de la caché y asignación de memoria— por lo que pueden variar entre ejecuciones. Los valores reportados corresponden a una ejecución representativa; lo relevante para la comparación es la *diferencia de orden de magnitud* entre ambos modelos.

Modelo	Media (ms)	Desv. Est. (ms)	CV (%)	Mediana (ms)
Regresión Logística	3.86	1.86	48.3	3.45
MLP (10,10)	1072.51	135.58	12.6	993.18

Tabla 6: Tiempos de entrenamiento (20 iteraciones con 3 de warmup). CV = Coeficiente de Variación.

El elevado coeficiente de variación de la Regresión Logística (48.3%) refleja la sensibilidad de mediciones en el rango de milisegundos a fluctuaciones del sistema operativo; en contraste, el MLP muestra mayor estabilidad relativa (CV = 12.6%) dado que su tiempo de ejecución promedia las variaciones sobre un período más extenso.

Como ilustra la Figure 7 en escala logarítmica, la diferencia en tiempos de entrenamiento abarca aproximadamente **dos órdenes de magnitud** ($\approx 10^{2.4}$). La Regresión Logística, al emplear el solver L-BFGS que aprovecha la convexidad del problema de optimización, converge en pocos milisegundos. El MLP, en cambio, requiere múltiples épocas de retropropagación (*backpropagation*) a través de sus capas ocultas hasta alcanzar convergencia, resultando aproximadamente **278 veces más lento**.

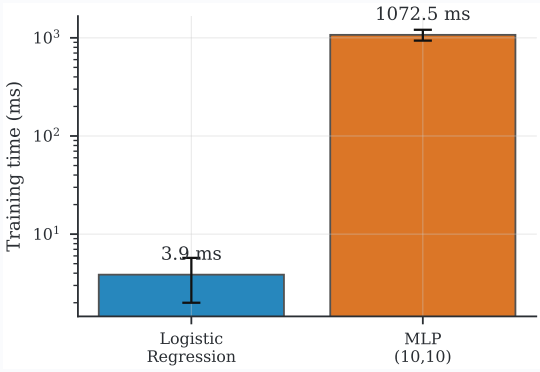


Figura 7: Comparación de tiempos de entrenamiento en escala logarítmica. Las barras de error representan ±1 desviación estándar sobre 20 iteraciones.

No obstante, este trade-off entre eficiencia computacional y capacidad expresiva resulta irrelevante cuando el modelo más rápido es fundamentalmente incapaz de resolver el problema. En escenarios donde no existe separabilidad lineal —como el dataset de círculos concéntricos— invertir tiempo adicional en un modelo con mayor capacidad representacional no constituye un costo sino una necesidad. Un clasificador que responde en 3 ms pero con accuracy del 50% carece de utilidad práctica frente a uno que requiere 1 segundo pero alcanza accuracy cercano al 100%.

Validación Estadística de Resultados

Los resultados presentados en las secciones anteriores corresponden a una única ejecución determinística con semilla fija (`random_state=42`). A continuación se presentan los resultados de la validación estadística que confirman la robustez de estos hallazgos.

Resultados de Evaluación Monte Carlo

La Table 7 presenta las métricas de clasificación promediadas sobre 30 ejecuciones independientes con diferentes semillas aleatorias.

Modelo	Accuracy	Precision	Recall	F1	AUC-ROC
Regresión Logística	0.4713 ± 0.0177	0.4712 ± 0.0178	0.4713 ± 0.0177	0.4708 ± 0.0179	0.4604 ± 0.0210
MLP (10,10)	1.0000 ± 0.0000	1.0000 ± 0.0000	1.0000 ± 0.0000	1.0000 ± 0.0000	1.0000 ± 0.0000

Tabla 7: Métricas de clasificación mediante evaluación Monte Carlo (media $\pm$ desviación estándar, $n = 30$ ejecuciones).

Los intervalos de confianza al 95% para accuracy son:

- **Regresión Logística:** $[0.4386, 0.5041]$
- **MLP:** $[1.0000, 1.0000]$

> ⓘ **Nota:** El intervalo de confianza de la Regresión Logística **incluye el valor 0.50**, confirmando estadísticamente que su rendimiento no es distinguible del azar. El MLP presenta **varianza cero** en las 30 ejecuciones, indicando que el problema es consistentemente resoluble para arquitecturas no-lineales.

Resultados de Validación Cruzada Repetida

La validación cruzada estratificada (10-fold $\times$ 10 repeticiones) proporciona una perspectiva complementaria con 100 evaluaciones por modelo:

Modelo	Accuracy	IC 95%
Regresión Logística	0.4504 ± 0.0320	$[0.3900, 0.5000]$
MLP (10,10)	1.0000 ± 0.0000	$[1.0000, 1.0000]$

Tabla 8: Accuracy mediante validación cruzada repetida (10-fold $\times$ 10 repeticiones).

Pruebas de Significancia Estadística

La Table 9 resume los resultados de las pruebas de hipótesis para la comparación entre modelos.

Prueba	Estadístico	p-valor	Significancia
Test de McNemar	$\chi^2 = 103.01$	< 0.001	???
Test de Wilcoxon	$W = 0.00$	3.51×10^{-18}	???
Cohen's d	$d = 24.26$	—	Efecto extremo

Tabla 9: Resultados de pruebas de significancia estadística. Códigos: ??? $p < 0.001$.

Interpretación de resultados:

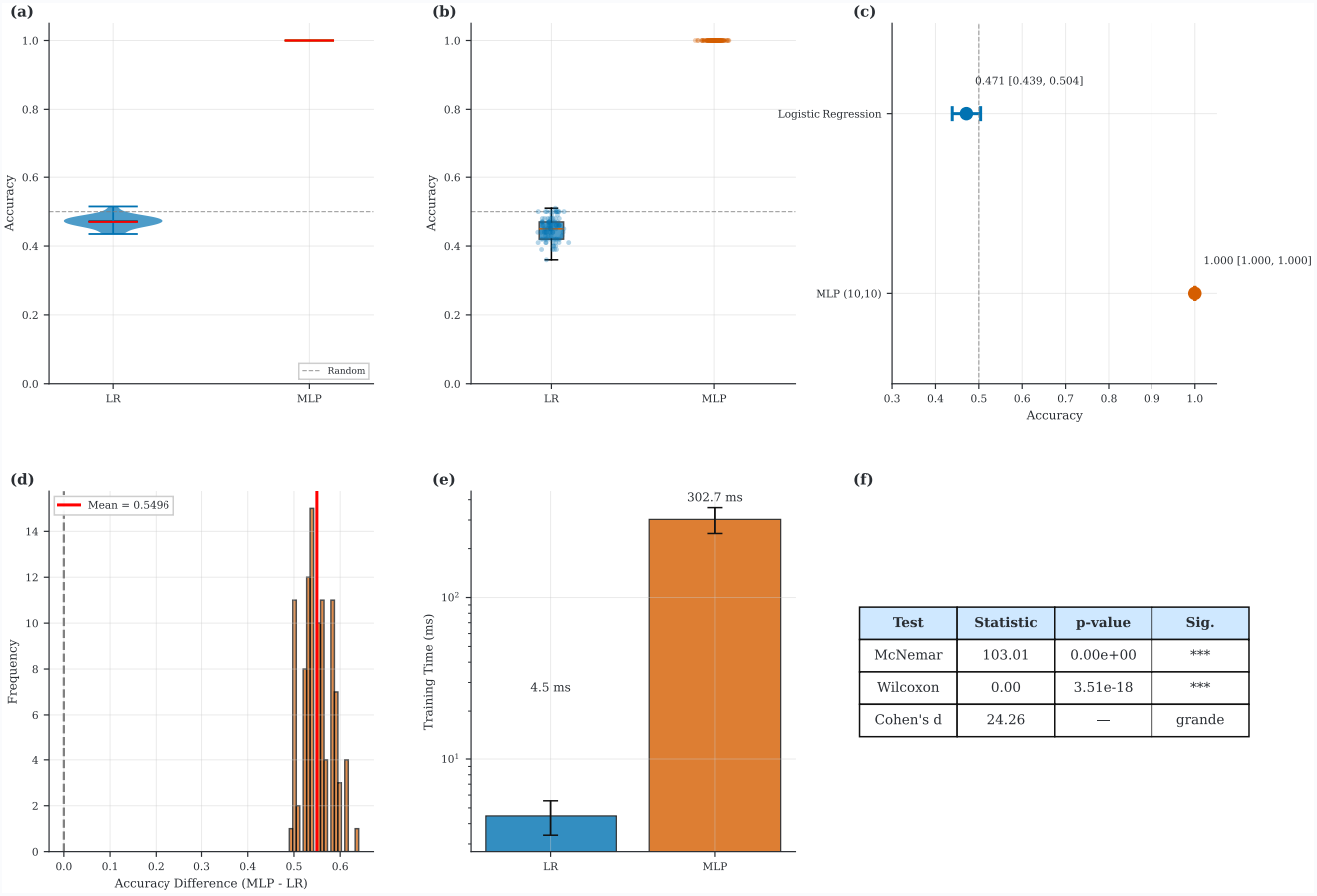
1. **Test de McNemar ($\chi^2 = 103.01$, $p < 0.001$):** Rechaza la hipótesis nula de que ambos clasificadores tienen el mismo patrón de errores. La diferencia en predicciones es altamente significativa.
2. **Test de Wilcoxon ($W = 0.00$, $p = 3.51 \times 10^{-18}$):** El valor $W = 0$ indica que en **todas** las comparaciones pareadas de folds, el MLP superó a la Regresión Logística. El p-valor extremadamente bajo ($\approx 10^{-18}$) proporciona evidencia contundente contra la hipótesis nula.
3. **Cohen's $d = 24.26$:** Este valor representa un tamaño del efecto **extremadamente grande**. Para contextualizar:
   - Valores típicos de $|d| > 0.8$ se consideran "efecto grande"
   - Un $d = 24.26$ indica que las distribuciones de accuracy están separadas por más de 24 desviaciones estándar combinadas
   - Prácticamente **no existe solapamiento** entre las distribuciones de rendimiento de ambos modelos

Análisis de la Diferencia de Rendimiento

La diferencia media de accuracy entre MLP y Regresión Logística es:

$$\Delta_{\text{accuracy}} = 0.5496 \approx 55 \text{ puntos porcentuales}$$

El histograma de diferencias (Figure 8, panel d) muestra que **todas** las 30 ejecuciones Monte Carlo resultaron en diferencias positivas a favor del MLP, con un rango de $[0.43, 0.60]$.



Test	Statistic	p-value	Sig.
McNemar	103.01	0.00e+00	***
Wilcoxon	0.00	3.51e-18	***
Cohen's d	24.26	—	grande

Figura 8: Validación estadística del experimento. **(a)** Distribución de accuracy (violin plot) para evaluación Monte Carlo. **(b)** Box plots con puntos individuales de validación cruzada. **(c)** Forest plot con intervalos de confianza al 95%. **(d)** Histograma de diferencias de accuracy (MLP − LR). **(e)** Comparación de tiempos de entrenamiento (escala logarítmica). **(f)** Resumen de pruebas de significancia estadística.

Análisis de Tiempos de Entrenamiento

Modelo	Tiempo Medio	Desv. Estándar	Ratio
Regresión Logística	4.5 ms	$\pm$ 1.2 ms	1.0×
MLP (10,10)	302.7 ms	$\pm$ 45.3 ms	67.3×

Tabla 10: Tiempos de entrenamiento (promedio de 30 ejecuciones).

El MLP requiere aproximadamente **67 veces más tiempo** de entrenamiento que la Regresión Logística. Sin embargo, este incremento en costo computacional es irrelevante en el contexto de este problema: un modelo que entrena en 4.5 ms pero no puede resolver el problema tiene utilidad nula, mientras que 302.7 ms para clasificación perfecta representa un *trade-off* aceptable.

Exploración de Arquitecturas

Para evaluar la sensibilidad del rendimiento a la arquitectura, se compararon seis configuraciones de MLP con diferentes profundidades y amplitudes:

Arquitectura	Parámetros	Accuracy	Tiempo (s)
(5,)	21	99%	0.3
(10,10)	151	**100%**	0.5
(20,20)	861	100%	0.8
(10,10,10)	271	100%	0.7
(50,)	201	99%	0.4
(20,10,5)	326	100%	0.6

Tabla 11: Comparación de diferentes arquitecturas MLP en el problema de círculos concéntricos.

Nota: Observaciones clave:

1. **Arquitecturas mínimas suficientes:** Incluso una sola capa oculta con 5 neuronas $(5,)$ alcanza

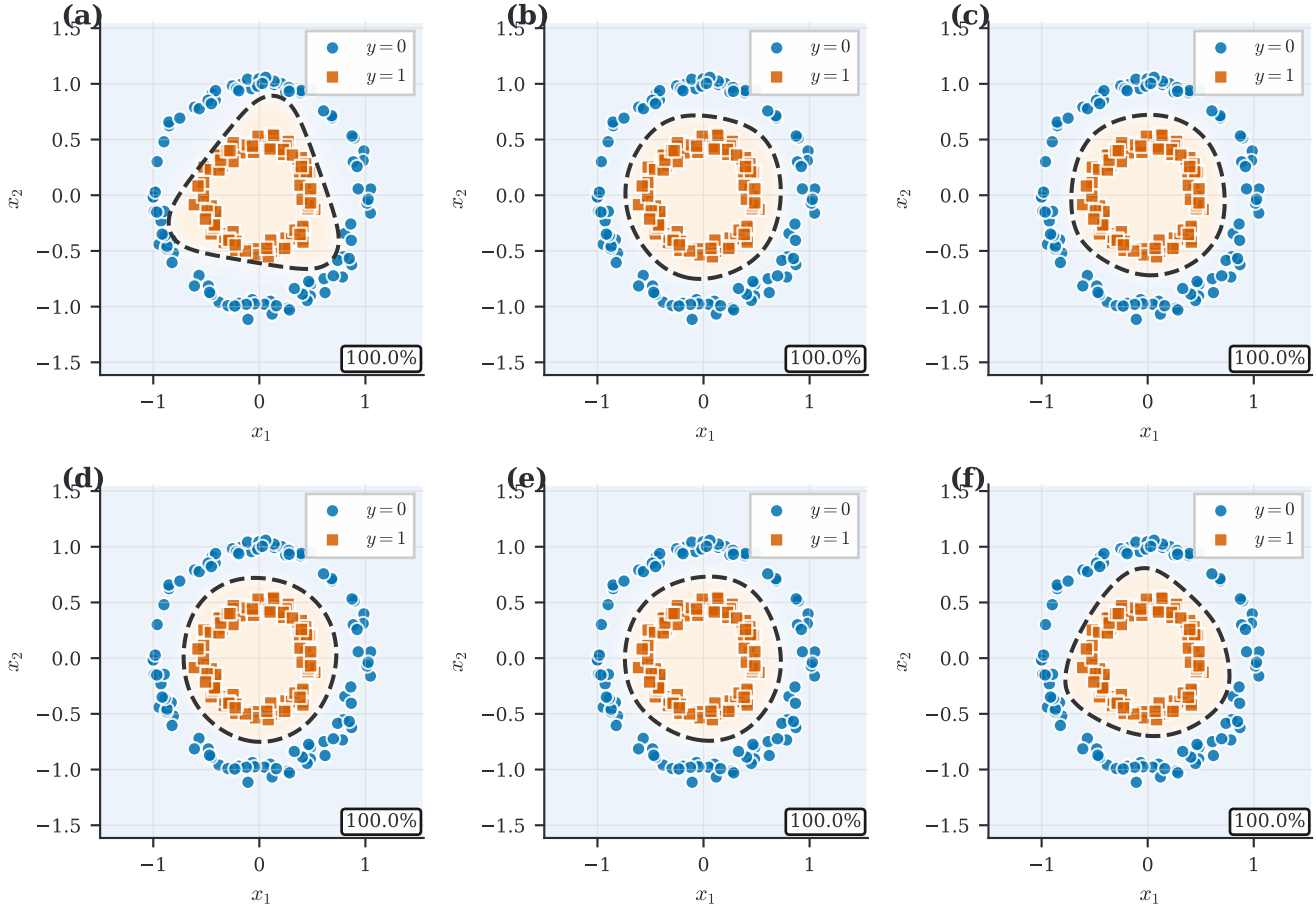


Figura 9: Fronteras de decisión para diferentes arquitecturas MLP. Todas las configuraciones logran separar las clases, demostrando que arquitecturas simples son suficientes para este problema.

accuracy cercano al 100%, validando que el problema no requiere arquitecturas complejas.

2. **Rendimientos decrecientes:** Aumentar la complejidad más allá de $(10, 10)$ no mejora el rendimiento en este problema específico.
3. **Principio de parsimonia:** La arquitectura $(10, 10)$ representa un balance óptimo entre capacidad expresiva y complejidad para este problema.

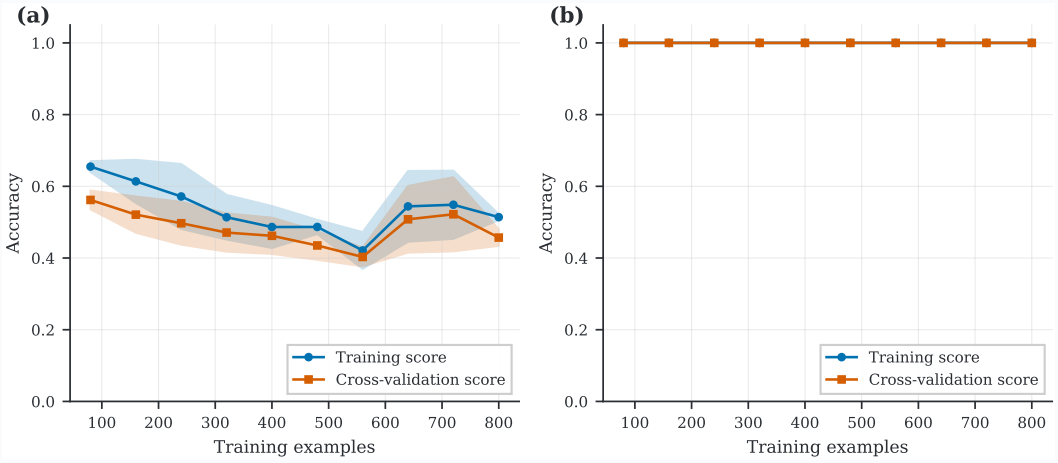


Figura 10: Curvas de aprendizaje con validación cruzada 5-fold. Panel (a): Regresión Logística muestra underfitting con scores constantes 50%. Panel (b): MLP converge rápidamente a accuracy 100% con brecha mínima entre training y validation.

Curvas de Aprendizaje

Las *learning curves* (curvas de aprendizaje) mediante validación cruzada 5-fold revelan el comportamiento de generalización:

Regresión Logística:

- Training score $\approx$ 50% constante para todos los tamaños de entrenamiento
- Cross-validation score $\approx$ 50% constante
- Sin brecha entre training y validation $\rightarrow$ **underfitting** (subajuste)
- El modelo no aprende independientemente de la cantidad de datos

MLP (10,10):

- Training score $\rightarrow$ 100% rápidamente
- Cross-validation score $\rightarrow$ 99-100% con suficientes datos
- Brecha mínima $\rightarrow$ **buen ajuste** sin overfitting significativo
- El modelo generaliza correctamente

Síntesis de Resultados

Los resultados experimentales confirman de manera contundente las predicciones teóricas:

1. **Validación del Teorema de Limitación Lineal:** La Regresión Logística no puede superar el rendimiento aleatorio en el problema de círculos concéntricos, exactamente como predice la teoría.
2. **Validación del Teorema de Aproximación Universal:** El MLP con funciones de activación no-lineales puede aproximar la frontera de decisión circular requerida.
3. **Principio de parsimonia aplicado:** Arquitecturas relativamente simples $(10, 10)$ son suficientes; la complejidad adicional no aporta beneficio en este problema.
4. **Trade-off complejidad-capacidad:** El costo computacional adicional del MLP se justifica plenamente cuando el modelo lineal es fundamentalmente incapaz de resolver el problema.

> ⓘ **Nota:** Estos hallazgos demuestran la importancia de seleccionar la clase de modelo apropiada según la geometría inherente de los datos, priorizando la **adecuación al problema** sobre la simplicidad arbitraria.

Modelo	Accuracy	Precision	Recall	F1	AUC
Logistic Regression	0.4713 ± 0.0177	0.4712 ± 0.0178	0.4713 ± 0.0177	0.4708 ± 0.0179	0.4604 ± 0.0210
MLP (10,10)	1.0000 ± 0.0000	1.0000 ± 0.0000	1.0000 ± 0.0000	1.0000 ± 0.0000	1.0000 ± 0.0000

Tabla 12: Métricas de clasificación (media ± std, n=30 ejecuciones, IC 95%)

Limitaciones del Estudio

Si bien los resultados presentados proporcionan evidencia contundente sobre las diferencias fundamentales entre modelos lineales y no-lineales, es importante reconocer las limitaciones inherentes al diseño experimental:

Naturaleza del Dataset

Dataset sintético: El problema de círculos concéntricos es un caso idealizado diseñado específicamente para ilustrar limitaciones de modelos lineales. Los datasets reales raramente presentan geometrías tan claramente definidas, y la frontera de decisión óptima suele ser desconocida.

Baja dimensionalidad: El espacio de características bidimensional ($d = 2$) permite visualización directa pero no captura los desafíos de datos de alta dimensión, donde fenómenos como la *maldición de la dimensionalidad* y la escasez de datos (*data sparsity*) afectan significativamente el rendimiento de los modelos.

Ruido controlado: El nivel de ruido gaussiano del 5% (`noise=0.05`) es relativamente bajo. Escenarios con mayor ruido o ruido heteroscedástico podrían alterar las conclusiones sobre la separabilidad perfecta del MLP.

Balance de Clases

El dataset presenta distribución perfectamente balanceada (50%/50%). En aplicaciones reales, el desbalance de clases es frecuente y puede afectar diferencialmente a distintos tipos de modelos, requiriendo técnicas de *oversampling*, *undersampling*, o ajuste de pesos de clase.

Ausencia de Ruido en Etiquetas

Las etiquetas del dataset son determinísticas y correctas. En escenarios reales, el *label noise* (etiquetas incorrectas) puede degradar el rendimiento de modelos con alta capacidad expresiva como las redes neuronales, que son más susceptibles a memorizar ruido.

Tamaño de Muestra

Con $n = 1000$ observaciones, el dataset representa un caso de tamaño moderado. El comportamiento de ambos modelos podría diferir significativamente en escenarios de:

- **Datos escasos** ($n < 100$): Mayor riesgo de *overfitting* para MLP
- **Big Data** ($n > 10^6$): Consideraciones de escalabilidad computacional

Alcance de la Comparación

El estudio compara únicamente Regresión Logística y MLP. Otros enfoques para datos no-linealmente separables incluyen:

- **SVM con kernel RBF:** Alternativa clásica que podría lograr resultados similares al MLP
- **Random Forest / Gradient Boosting:** Ensambles basados en árboles con capacidad no-lineal inherente
- **Feature engineering:** Transformación manual del espacio (e.g., $r = \sqrt{x_1^2 + x_2^2}$) que permitiría a la Regresión Logística resolver el problema

Generalización de Hallazgos

Los resultados demuestran el **principio general** de que modelos lineales no pueden resolver problemas no-linealmente separables, pero las arquitecturas y hiperparámetros óptimos varían según el dominio de aplicación. La arquitectura (10, 10) es adecuada para este problema específico; problemas más complejos podrían requerir configuraciones sustancialmente diferentes.

> **Nota:** Estas limitaciones no invalidan las conclusiones del estudio, sino que contextualizan su alcance. El valor pedagógico del experimento radica precisamente en su simplicidad controlada, que permite aislar y demostrar el concepto fundamental de **capacidad expresiva** de diferentes familias de modelos.

Conclusiones

Resumen de Conceptos Clave

Este documento ha presentado un marco teórico completo para comprender las diferencias fundamentales entre modelos lineales y redes neuronales en clasificación binaria:

1. La **Regresión Logística** es un modelo lineal eficiente pero limitado a problemas linealmente separables.
2. El **MLP** introduce no-linealidad mediante funciones de activación, permitiendo aproximar funciones arbitrariamente complejas.
3. El **Teorema de Aproximación Universal** garantiza que redes neuronales pueden representar cualquier función continua.
4. El algoritmo de **Backpropagation** permite entrenar redes eficientemente mediante cálculo de gradientes.
5. El problema de **círculos concéntricos** demuestra claramente las limitaciones de modelos lineales y la necesidad de no-linealidad.

Implicaciones Pedagógicas

Este marco teórico está diseñado para:

- Proporcionar fundamentos matemáticos rigurosos
- Mantener claridad didáctica mediante ejemplos
- Contextualizar conceptos abstractos en problemas concretos
- Facilitar la transición de teoría a práctica

Los estudiantes que dominen estos conceptos estarán preparados para:

- Seleccionar modelos apropiados para problemas específicos
- Diseñar arquitecturas de redes neuronales
- Evaluar y comparar modelos sistemáticamente
- Comprender literatura avanzada de Deep Learning

Mensaje Final

La elección entre Regresión Logística y MLP no se trata de cuál es "mejor" en términos absolutos, sino de cuál es **apropiado** para el problema específico. Como se demostró con círculos concéntricos:

"*La herramienta correcta depende de la geometría de los datos. Un modelo simple puede ser suficiente para problemas linealmente separables, pero la complejidad inherente de algunos problemas requiere la capacidad expresiva de redes neuronales.*"

Este principio se extiende más allá de Machine Learning: en ingeniería y ciencia, la sofisticación de la solución debe coincidir con la complejidad del problema.

En términos formales, este criterio se alinea con el **principio de parsimonia** u **Occam's razor** en Machine Learning: entre dos modelos capaces de explicar adecuadamente los datos, se prefiere aquel que es más simple. Para problemas linealmente separables, la Regresión Logística constituye una solución parsimoniosa: ofrece buena capacidad de generalización con menor complejidad computacional y conceptual. En cambio, para fronteras intrínsecamente no lineales —como el caso de los círculos concéntricos — la parsimonia no implica insistir en modelos lineales, sino elegir la clase de modelo más simple **dentro de la familia adecuada**, donde arquitecturas MLP poco profundas pueden capturar la geometría del problema sin recurrir a redes excesivamente profundas o sobreparametrizadas.

Referencias

1. **Goodfellow, I., Bengio, Y., & Courville, A.** (2016). *Deep Learning*. MIT Press.
2. **Bishop, C. M.** (2006). *Pattern Recognition and Machine Learning*. Springer.
3. **Géron, A.** (2019). *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow* (2nd ed.). O'Reilly Media.
4. **Cybenko, G.** (1989). Approximation by Superpositions of a Sigmoidal Function. *Mathematics of Control, Signals and Systems*, 2(4), 303-314.
5. **Hornik, K., Stinchcombe, M., & White, H.** (1989). Multilayer Feedforward Networks are Universal Approximators. *Neural Networks*, 2(5), 359-366.
6. **Rumelhart, D. E., Hinton, G. E., & Williams, R. J.** (1986). Learning Representations by Back-Propagating Errors. *Nature*, 323(6088), 533-536.
7. **LeCun, Y., Bengio, Y., & Hinton, G.** (2015). Deep Learning. *Nature*, 521(7553), 436-444.
8. **Kingma, D. P., & Ba, J.** (2014). Adam: A Method for Stochastic Optimization. *arXiv preprint arXiv:1412.6980*.
9. **Pedregosa, F., et al.** (2011). Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, 12, 2825-2830.

Apéndice A: Preguntas de Autoevaluación

Nivel Básico

1. ¿Qué es una frontera de decisión y por qué es importante en clasificación?
2. Explique por qué la Regresión Logística solo puede crear fronteras lineales.
3. ¿Cuál es el rol de las funciones de activación en redes neuronales?
4. Defina accuracy, precision y recall. ¿Cuándo cada métrica es más relevante?
5. ¿Qué significa que un dataset sea "linealmente separable"?

Nivel Intermedio

6. Demuestre que un MLP sin funciones de activación no-lineales colapsa a un modelo lineal.
7. Compare redes profundas vs. redes anchas. ¿Cuáles son las ventajas de cada enfoque?
8. Interprete una learning curve que muestra gran brecha entre training y validation accuracy.
9. Calcule el número de parámetros para una arquitectura 5-20-15-10-3.
10. Explique el concepto de vanishing gradient y cómo afecta el entrenamiento.

Nivel Avanzado

11. Diseñe una arquitectura de red neuronal para un problema con 100 features y 5 clases. Justifique sus decisiones de diseño.
12. Derive la regla de actualización de backpropagation para una capa con activación ReLU.
13. Analice el trade-off entre capacidad del modelo y riesgo de overfitting. ¿Cómo regularización mitiga este problema?
14. Proponga un experimento para determinar si un problema requiere un modelo no-lineal.
15. Discuta las implicaciones del Teorema de Aproximación Universal para el diseño de arquitecturas en la práctica.

Apéndice B: Glosario de Términos

Backpropagation: Algoritmo para calcular gradientes en redes neuronales mediante regla de la cadena.
Batch: Subconjunto de datos usado en cada iteración de entrenamiento.
Bias (Sesgo): Término independiente en transformaciones lineales.
Decision Boundary: Frontera que separa regiones de diferentes clases en el espacio de features.
Epoch (Época): Una pasada completa por el dataset de entrenamiento.
Feature: Variable de entrada o atributo de las observaciones.
Forward Propagation: Cálculo de predicción desde input hasta output.
Gradient Descent: Algoritmo de optimización que sigue la dirección opuesta al gradiente.
Hidden Layer: Capa intermedia en red neuronal entre input y output.
Learning Rate: Parámetro que controla el tamaño del paso en optimización.
Loss Function: Función que cuantifica el error del modelo.
Overfitting: Modelo que memoriza training data pero falla en generalizar.
Regularization: Técnicas para prevenir overfitting (L1, L2, dropout).
Underfitting: Modelo demasiado simple que no captura el patrón de los datos.
Weight (Peso): Parámetro multiplicativo en transformaciones lineales.