
# PRESENTACIÓN 1: REDES NEURONALES

*Machine Learning II*

Jose Alexander Fuentes Montoya Ph.D(c)

# CONTENIDO DEL CURSO

- **Semana 1**:
  Fundamentos del Machine Learning y Fundamentos de Deep Learning:
  *Introducción, Métricas*
- **Semana 2**:
  Redes Neuronales XOR
- **Semana 3**:
  Redes Neuronales Convolucionales Y Optimización

- **Semana 4**:
  Aplicaciones del Deep Learning al procesamiento de lenguaje natural y al procesamiento de imágenes, junto con modelos generativos.
- **Proyecto integrador**
  Tarde semana 4

# ESQUEMA DE EVALUACIÓN

- **Guía – Semana 1**: 20%
- **Guía – Semana 2**: 20%
- **Guía – Semana 3**: 20%
- **Exposición**: 15%
- **Proyecto Integrador**: 25%

# PRIMERA PARTE

# REVISITANDO EL APRENDIZAJE AUTOMÁTICO

Imagina que te transportan a finales de la década de 1990 en los Estados Unidos, donde las autoridades descubren tu experiencia en Machine Learning (ML). Se ponen en contacto contigo para solicitar tu asistencia en la automatización de una tarea que consume mucho tiempo: leer el código postal (pin code) en las cartas.

Supongamos que hay 500 empleados en varias oficinas de correos de todo el país realizando esta tarea, y cada empleado gana 2000 dólares al mes por ello. Esto se traduce en un gasto mensual de 1.000.000 de dólares, resultando en un coste anual de 12 millones de dólares y, a nivel nacional, la asombrosa cifra de 60 millones de dólares en los próximos cinco años.

Para ayudar al gobierno a ahorrar fondos, se te asigna el diseño de un programa que pueda leer e interpretar, de forma eficiente, los códigos postales de las cartas. Esta solución no solo ayudará a reducir costos, sino que también incrementará considerablemente la exactitud y la velocidad del proceso. ¿Se te ocurre algún algoritmo para cumplir con esta tarea?

Para entender el problema, comencemos con un algoritmo que reconozca el dígito “1” en una imagen de 28×28 píxeles. Idealmente, podríamos considerar los píxeles cercanos a la columna central para identificar si la imagen contiene un “1”. Sin embargo, el número es escrito a mano, y por lo tanto puede variar en escala, orientación, estilo, etc.

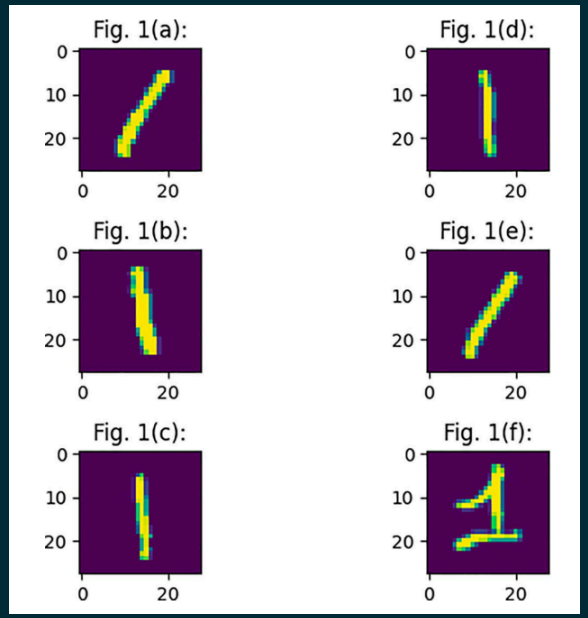



Figura 1-1: Algunas imágenes de “1” manuscritos del conjunto de datos MNIST

Para los seres humanos, reconocer dígitos manuscritos es una tarea sencilla, pero es muy difícil describir un conjunto de reglas o algoritmos que reconozcan el dígito en una imagen. Por tanto, necesitamos algún sistema que imite la capacidad humana para lograr esta tarea. Aquí es donde puede ayudar el Aprendizaje Automático (ML). Informalmente, podemos definir el ML de la siguiente manera:

> El Aprendizaje Automático es un subconjunto de la Inteligencia Artificial, que puede considerarse la capacidad de las máquinas para imitar a los seres humanos. [1]

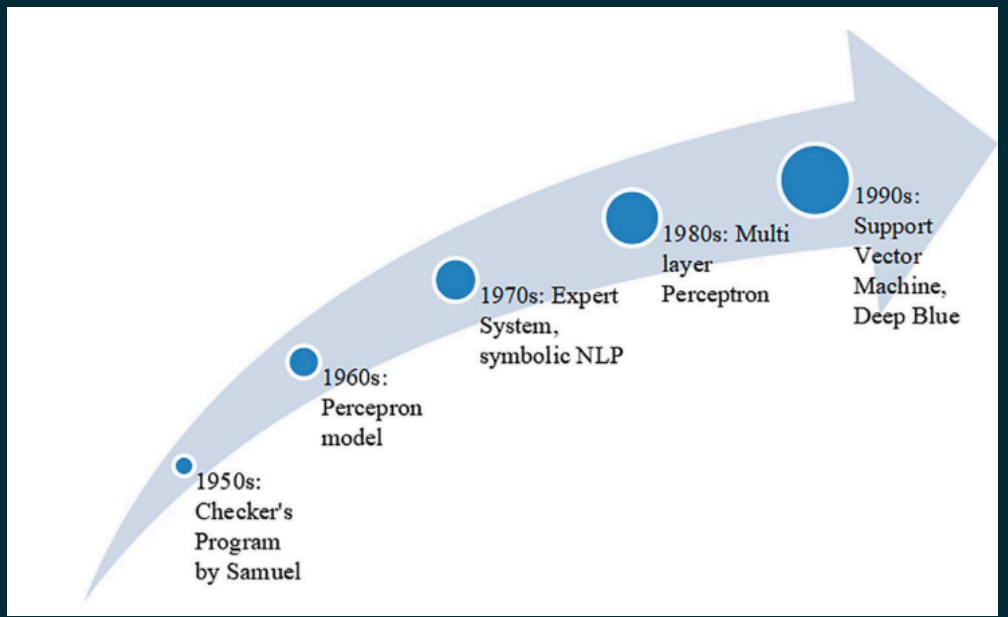
El ML nos ayuda a realizar tareas como clasificación de enfermedades, predicción y pronóstico, reconocimiento de objetos, clasificación de sentimientos, etc.

Se aborda un método importante de reducción de dimensiones llamado Análisis de Componentes Principales (PCA). Revisarmos el caso de estudio de clasificación del conjunto de datos MNIST usando una pipeline convencional de Machine Learning, que emplea la técnica de extracción de características LBP (Local Binary Pattern), la selección de características mediante un método filtro y la clasificación mediante Support Vector Machine (SVM).

# BREVE HISTORIA, DEFINICIÓN Y APLICACIONES DEL APRENDIZAJE AUTOMÁTICO

Desde tiempos inmemoriales, los seres humanos han intentado desarrollar máquinas que sean intelectualmente tan buenas como los humanos. El deseo de que las máquinas aprendan como lo hacen los humanos y mejoren en una tarea con la experiencia nos ha llevado a la época actual de extraordinario avance tecnológico. Dicha mejora debe poderse medir. El desarrollo del Programa de Damas (Checkers Program) de Samuel, en IBM, en la década de 1950, puede considerarse uno de los primeros pasos hacia este objetivo.

La década de 1960 vio progresos en el campo del reconocimiento de patrones, especialmente después de los trabajos de Rosenblatt sobre perceptron, seguidos por los de Minsky y Papert, que describieron las limitaciones de este. En la década de 1970 se desarrollaron sistemas expertos y procesamiento simbólico del lenguaje natural.




ML antes del 2000

La siguiente década fue testigo de avances en árboles de decisión y del desarrollo del Multi-Layer Perceptron (MLP). Algunos de los métodos de aprendizaje más relevantes, como Support Vector Machines, Reinforcement Learning y modelos de ensemble, surgieron en la década de 1990.

El deseo de la comunidad científica de desarrollar máquinas que pudieran superar a los seres humanos en ciertas tareas cognitivas recibió un gran impulso con el desarrollo de Deep Blue, en IBM, que derrotó al entonces campeón de ajedrez, Garry Kasparov. Desde entonces, los esfuerzos por diseñar coches autónomos utilizando, inicialmente, estas metodologías han avanzado muchísimo.

El Aprendizaje Automático (Machine Learning, ML) es un subconjunto de la Inteligencia Artificial (IA). Los algoritmos de ML se entrenan y prueban usando conjuntos de datos, y nos ayudan a realizar tareas que, de manera natural, los humanos dominan mejor. El conjunto de datos puede estar o no etiquetado. La IA, por otro lado, se esfuerza por desarrollar máquinas con “capacidades cognitivas similares a las humanas” [2]. Para entender el concepto, tomemos un ejemplo:

Supongamos que necesitas desarrollar un sistema que tome una imagen como entrada y la clasifique como “gato” o “no gato”. Las imágenes de entrada tienen tamaño 100×100, y la salida es un número binario que vale 0 (no es gato) o 1 (sí es gato). De alguna manera, desarrollas este sistema y lo pruebas con 1000 imágenes nuevas, de las cuales el sistema identifica correctamente 673. El porcentaje de acierto (exactitud) es, por tanto, 67.3 %.

Le pides a un amigo, ingeniero en Aprendizaje Automático, que te ayude a mejorar el sistema. Él lo modifica, y entonces el sistema clasifica correctamente 721 imágenes, mejorando la exactitud en un 4.8 %. Además, conforme el sistema se entrena con más imágenes, la exactitud aumenta. Si consideramos la exactitud (porcentaje de muestras no vistas clasificadas correctamente) como la medida de rendimiento (P), y los datos como la experiencia (E), en la tarea (T) de clasificación, podemos decir que el sistema "aprende".

Formalmente, el Aprendizaje Automático se define así:

> Un sistema se considera que aprende cuando el rendimiento (P) mejora con la experiencia (E), en la tarea (T). [3]

Actualmente, el ML se utiliza en una gran variedad de dominios: desde la recomendación de productos, pasando por la predicción de la bolsa, hasta la detección de enfermedades, entre otros.

# EJEMPLOS INTERESANTES DE APLICACIONES DEL ML

- **Sistemas de Recomendación**

  Harry tenía una cuenta en Amazon y comenzó a comprar sus cosas favoritas cuando recibió su primer salario. Le encantaban los libros, papelería y la música. Compró libros como Bajo la misma estrella, cuadernos bonitos e incluso un instrumento de percusión. Al mes siguiente, compró artículos similares. Cuando visitó la plataforma de nuevo, en la sección de “recomendaciones” aparecieron libros de John Green y otros, más instrumentos musicales, cuadernos e incluso barras de sonido.

¿A qué se debe que aparezcan libros de John Green y barras de sonido en las recomendaciones?

Es porque la plataforma “aprende” con Machine Learning, aprovechando datos de usuarios y valoraciones, además de Natural Language Processing. Si comparas lo que te recomienda YouTube a ti y a otra persona, notarás diferencias debidas justamente a estos sistemas de ML.

- **Google Maps**

  Supón que necesitas ir a una entrevista en una empresa situada en Gurugram, una ciudad cercana a Nueva Delhi, la capital de la India. Vives en Delhi y nunca has estado en esa empresa. Decides manejar tu coche para llegar al destino y buscar la mejor ruta con Google Maps.

¿Cómo sabe la app cuál es la mejor ruta desde tu ubicación hasta tu destino? Además, la app afirma que ciertas rutas son mejores que otras, ya sea por congestión, distancia u otros factores.

Esta app utiliza Machine Learning para encontrar el camino óptimo. Obtiene datos de tráfico de “Waze”, una app que Google adquirió en 2013. Si la usas desde hace tiempo, habrás notado que su rendimiento ha mejorado notablemente; el crédito también es para el ML. (¡Claro que al tener tu localización activada, Google Maps se beneficia de esos datos!)

- Otros ejemplos relevantes:
  - Detección y predicción de enfermedades
  - Amazon Alexa
  - Vehículos autónomos
  - Análisis de sentimientos
  - Predicción de customer churning (pérdida de clientes)

# TIPOS DE APRENDIZAJE AUTOMÁTICO: LA TAREA (T)

El Aprendizaje Automático puede clasificarse como supervisado, no supervisado, semisupervisado o por refuerzo (Reinforcement Learning).

- **Aprendizaje Supervisado**
  El sistema se entrena con muestras y sus etiquetas correspondientes. Durante la prueba, se ingresa la entrada y se obtiene una etiqueta predicha. El algoritmo de aprendizaje intenta ajustar los parámetros del modelo para disminuir la diferencia entre la etiqueta predicha y la etiqueta real. Puede ser:
  - Clasificación: las etiquetas son valores discretos.
  - Regresión: las etiquetas son valores continuos.

- **Aprendizaje No Supervisado**
  El sistema solo recibe las características, sin etiquetas. Estos algoritmos buscan patrones en los datos. Ejemplos: búsqueda de tendencias en redes sociales, asociación entre productos, etc.

- **Aprendizaje Semisupervisado**
  Combina ambos enfoques. Se tienen datos sin etiquetar, pero también se dispone de etiquetas para parte de las muestras (no para todas).

- **Aprendizaje por Refuerzo (Reinforcement Learning)**
  El sistema actúa sobre un entorno y recibe retroalimentación. En función de esa retroalimentación, ajusta sus acciones. Es común en drones automatizados, entre otros.

  El siguiente elemento en la definición de ML es el rendimiento, P. Veamos ahora algunas medidas de rendimiento.

# RENDIMIENTO (P)

Considera un problema de clasificación con dos clases: Positivo (P) y Negativo (N). Diseñas un sistema que, dada una muestra desconocida, la clasifica como Positivo o Negativo. Las predicciones posibles son:

- **Verdadero Positivo (TP)**: la muestra es realmente positiva y el modelo la clasifica correctamente como tal.
- **Verdadero Negativo (TN)**: la muestra es realmente negativa y el modelo la clasifica correctamente como tal.
- **Falso Positivo (FP)**: la muestra es negativa, pero el modelo la clasifica incorrectamente como positiva (Error de Tipo I).
- **Falso Negativo (FN)**: la muestra es positiva, pero el modelo la clasifica incorrectamente como negativa (Error de Tipo II).

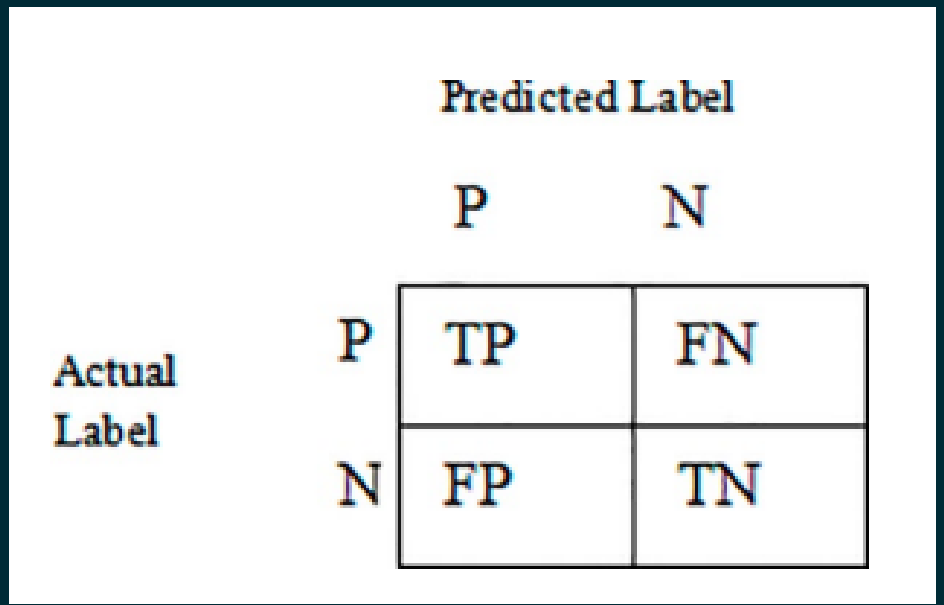



Figura 1-3: Matriz de confusión en un problema de dos clases

Con base en TP, TN, FP y FN, se pueden calcular varias métricas para evaluar el rendimiento de un modelo de clasificación, por **ejemplo**: Exactitud `(accuracy)`, Sensibilidad o recall, Precisión, Puntaje F1 `(F1 score)` y Especificidad `(o True Negative Rate)`. En general, queremos minimizar FP y FN, y maximizar TP y TN.

# TABLA 1-1. MÉTRICAS DE CLASIFICACIÓN

accuracy = $$\frac{TP + TN}{TP + TN + FN + FP}$$

specificity = $$\frac{TN}{TN + FP}$$

**sensitivity (recall) =**

$$\frac{TP}{TP + FN}$$

**precision =**

$$\frac{TP}{TP + FP}$$

F-score =

$$\frac{2 \times (\text{recall} \times \text{precision})}{(\text{recall} + \text{precision})}$$

# En `sklearn`, se implementa con `sklearn.metrics.confusion_matrix.

***Table 1-1.*** *Classification Metrics*

| Performance Measure | Formula | Description | Keras Implementation | sklearn Implementation |
|---|---|---|---|---|
| Accuracy | $\frac{TP+TN}{TP+TN+FN+FP}$ | Total number of test cases correctly classified. | tf.keras.metrics.Accuracy[1] | sklearn.metrics.accuracy_score |
| Specificity (False Positive Rate) | $\frac{TN}{TN+FP}$ | Total number of negative test cases correctly classified_. | | |
| Sensitivity/ recall (True Positive Rate) | $\frac{TP}{TP+FN}$ | Total number of positive test cases correctly classified. | tf.keras.metrics.Recall1[1] | sklearn.metrics.recall_score |
| Precision | $\frac{TP}{TP+FP}$ | Goodness of positive predictions. | tf.keras.metrics.Precision[1] | precision_score[2] |
| F-score | $(2 \times Recall \times Precision)/(Recall + Precision)$ | It is used for unbalanced class problems, where accuracy may be misleading. | tf.keras.metrics.F1Score[1] | f1_score[3] |

La curva ROC (Receiver Operating Characteristic) se traza representando la sensibilidad (TPR) frente a (1 – especificidad) para distintos umbrales de decisión. El área bajo esta curva se denomina AUC (Área Bajo la Curva ROC).

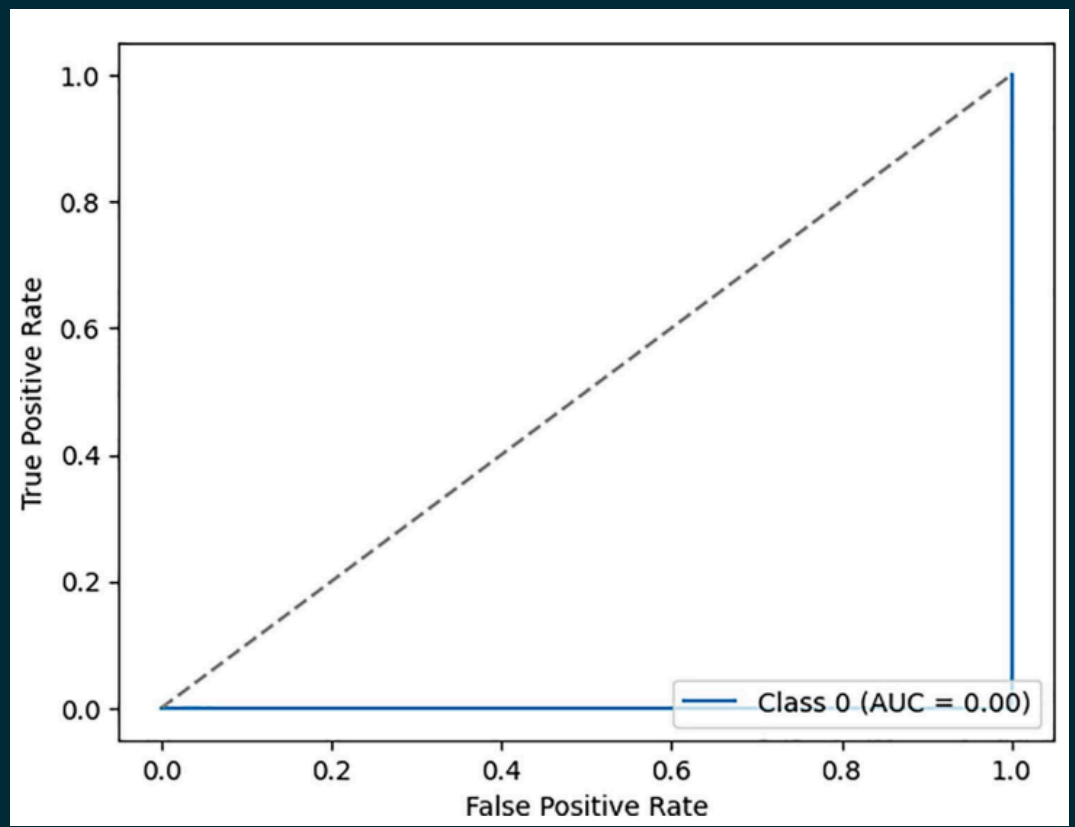



Figura 1-4: Un ejemplo de curva ROC-AUC

# MÉTRICAS DE RENDIMIENTO EN REGRESIÓN

En problemas de regresión, donde la etiqueta es un valor continuo, se utilizan otras métricas, como:

- **MSE** (Mean Squared Error)
- **MAE** (Mean Absolute Error)
- **RMSE** (Root MSE)
- **R^2** (R-squared)

***Table 1-2.*** *Regression Metrics*

| Performance Measure | Formula | sklearn Implementation | Keras Implementation |
|---|---|---|---|
| Mean Squared Error | $\frac{1}{N}\sum_{i=1}^{N}(y_i-\hat{y})^2$ | sklearn.metrics.mean_ squared_error | tf.keras.metrics. MeanSquaredError |
| Root Mean Squared Error | $\sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i-\hat{y})^2}$ | sklearn.metrics.mean_ squared_error squared = False, returns RMSE | tf.keras.metrics. RootMeanSquaredError |
| Mean Absolute Error | $\frac{1}{N}\sum_{i=1}^{N}\lvert y_i-\hat{y}\rvert$ | sklearn.metrics.median_ absolute_error | tf.keras.metrics. MeanAbsoluteError |
| R-Squared | $1-\frac{\frac{1}{N}\sum_{i=1}^{N}(y_i-\hat{y})^2}{\frac{1}{N}\sum_{i=1}^{N}(y_i-\bar{y})^2}$ | sklearn.metrics.r2_score | tf.keras.metrics. R2Score |

Tabla 1-2: Métricas de Regresión

# PIPELINE CONVENCIONAL DE MACHINE LEARNING

La pipeline convencional de ML incluye el proceso completo de desarrollo de un modelo de Machine Learning, desde la recolección de datos hasta el despliegue. Sus pasos principales son:

1. **Definición del Problema**
2. **Recolección y Preprocesamiento de Datos**
3. **Análisis Exploratorio de Datos (EDA)**
4. **Ingeniería de Características**
5. **División de Datos** (train, validation, test)
6. **Elección del Modelo**
7. **Entrenamiento del Modelo**
8. **Evaluación del Modelo**
9. **Análisis y despliegue**

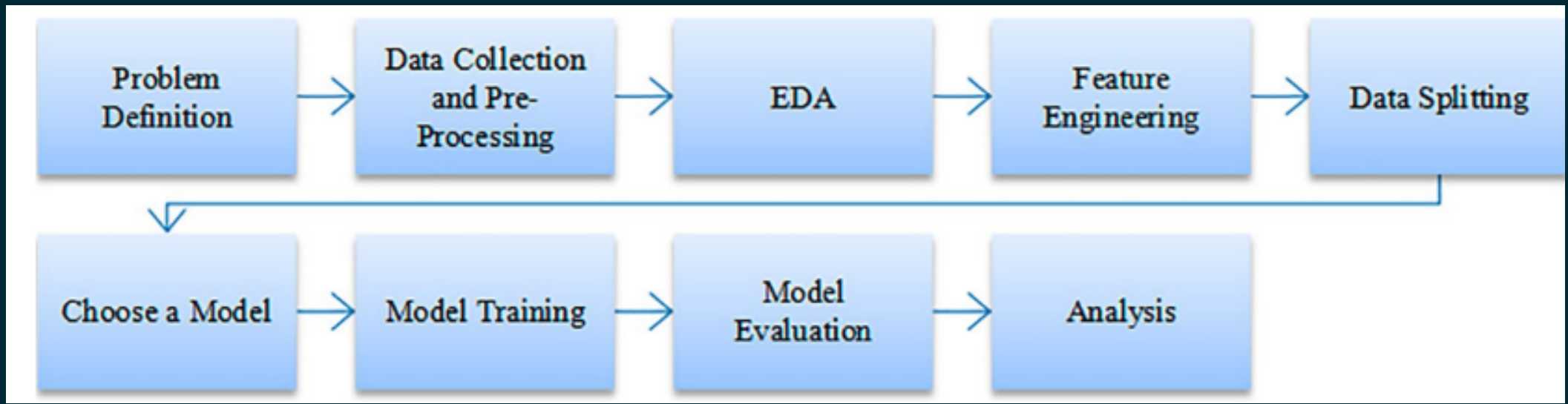



Figura 1-5: Pipeline convencional de Machine Learning

# REGRESIÓN

La regresión es un tipo de aprendizaje supervisado donde tenemos $(X, y)$, con $y \in \mathbb{R}$ e $X \in \mathbb{R}^d$. Es decir, las etiquetas son valores continuos. El objetivo es desarrollar un modelo que prediga $\hat{y}$ para un $X$ desconocido, habiendo sido entrenado en un conjunto de datos de entrenamiento.

Los parámetros del modelo se aprenden minimizando la diferencia cuadrática entre $\hat{y}$ y $y$. Ejemplo:

En el caso de regresión lineal, la etiqueta $y$ se modela como combinación lineal de $X_i^m$:

$$y\_\text{pred} = \sum_i w_i X_i^m.$$

Estos pesos $w_i$ se ajustan con métodos como descenso de gradiente.

# SELECCIÓN DE CARACTERÍSTICAS

La selección de características busca elegir un subconjunto de las características originales para minimizar el error de clasificación o regresión, reduciendo la dimensionalidad y los requisitos de memoria y cómputo. Algunas características no aportan valor y otras incluso pueden empeorar el rendimiento.

La selección de características se ve como un problema de optimización. Hay dos enfoques principales:

- **Métodos Filtro**
- **Métodos Wrapper**

# MÉTODOS FILTRO

En los métodos filtro, la selección de características es independiente del algoritmo de aprendizaje. Se usan métricas como la Razón Discriminante de Fisher (FDR) para problemas de dos clases, donde

$$\mathrm{FDR} = \frac{(\mu_1 - \mu_2)^2}{s_1^2 + s_2^2}.$$

Se puede usar en la Selección Secuencial Adelante (FFS), eligiendo las características con mayor FDR y añadiéndolas progresivamente hasta optimizar la performance.

# MÉTODOS WRAPPER

En los métodos wrapper, se evalúan iterativamente distintos subconjuntos de características en conjunto con el clasificador para ver cuál da mejor rendimiento. Un método popular es **RFE** (Recursive Feature Elimination), que elimina características una por una y mide el impacto en el rendimiento, hasta converger al mejor subconjunto.

En la práctica, los métodos filtro son más rápidos, pero a menudo retienen más características. Los métodos wrapper consiguen mejor rendimiento pero son más lentos y dependen de un clasificador concreto.

# EXTRACCIÓN DE CARACTERÍSTICAS

En clasificación de imágenes, por ejemplo, a menudo se extraen características para conseguir una representación compacta y discriminativa, mitigando la maldición de la dimensionalidad. Algunos métodos comunes:

- Histogramas de niveles de gris (GLCM, GLRL, etc.)
- Local Binary Pattern (LBP)
- Histogram of Oriented Gradients (HOG)
- PCA (para transformación de características)

# GLCM (GRAY-LEVEL CO-OCCURRENCE MATRIX)

Se calcula la matriz de co-ocurrencia de los niveles de gris en varias direcciones (0°, 45°, 90°, 135°). De ahí se derivan características como Contraste, Disimilitud, Homogeneidad, ASM, Energía y Correlación.

# LOCAL BINARY PATTERN (LBP)

Para cada píxel se compara con sus vecinos (8 vecinos en un radio (R)) y se asigna un valor binario. Se crea luego un histograma con los valores obtenidos. Hay varias variantes (`default`, `ror`, `uniform`, `nri_uniform`). Se ha usado frecuentemente en reconocimiento de texturas.

# HISTOGRAM OF ORIENTED GRADIENTS (HOG)

Se divide la imagen en bloques y se calculan gradientes horizontales (H) y verticales (V). Luego, se crea un histograma de direcciones  y se construye un vector que representa la distribución de los gradientes.

$$\theta = \arctan\left(\frac{V}{H}\right)$$

# ANÁLISIS DE COMPONENTES PRINCIPALES (PCA)

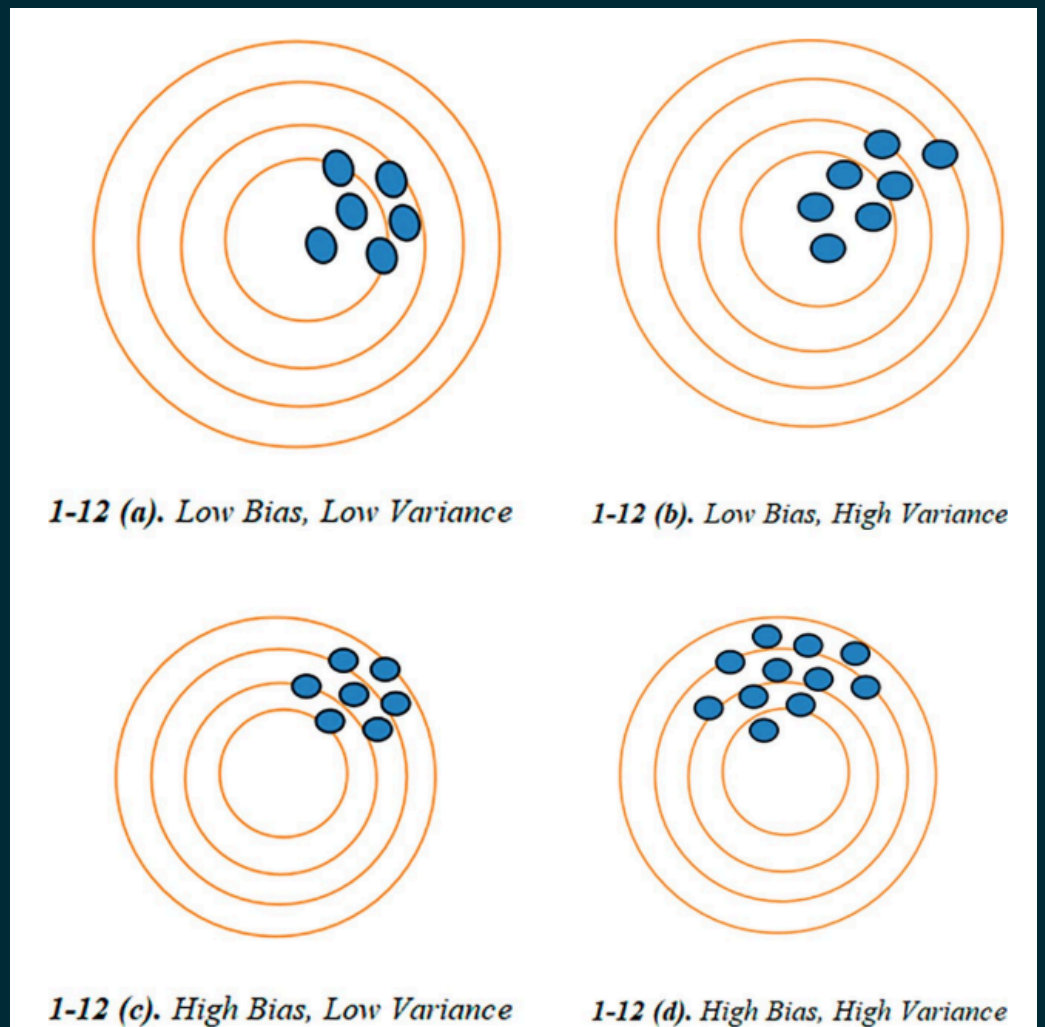
PCA encuentra un nuevo sistema de ejes (componentes principales) donde la varianza de los datos es máxima y las correlaciones se reducen. Se basa en calcular la matriz de covarianza y sus eigenvectores/eigenvalores, ordenarlos en función de la magnitud de los eigenvalores y proyectar los datos en las componentes principales seleccionadas. Se usa tanto para visualización como para reducción de dimensionalidad.

# COMPROMISO BIAS–VARIANCE (SESGO–VARIANZA)

Es uno de los conceptos clave en ML. Un modelo debe rendir bien tanto en entrenamiento como en prueba. Si un modelo rinde bien en entrenamiento pero mal en prueba, se denomina **overfitting**. Si rinde mal en ambos, es **underfitting**.

- **Bias** (sesgo): la diferencia entre la predicción promedio de nuestro modelo y el valor real. Un bias alto implica subajuste.
- **Variance** (varianza): la variabilidad de las predicciones si entrenamos el modelo con diferentes muestras de entrenamiento. Una varianza alta implica sobreajuste.

El objetivo es encontrar un equilibrio. La Figura 1-6 ilustra cuatro escenarios posibles (bias bajo/alto, varianza baja/alta).



***1-12 (a).*** *Low Bias, Low Variance*

***1-12 (b).*** *Low Bias, High Variance*

***1-12 (c).*** *High Bias, Low Variance*

***1-12 (d).*** *High Bias, High Variance*

# APLICACIÓN: CLASIFICACIÓN DE DÍGITOS MANUSCRITOS USANDO UNA PIPELINE CONVENCIONAL DE ML

Como ejemplo, clasificaremos dígitos del dataset MNIST usando la pipeline:

1. Preprocesamiento
2. Extracción de características (LBP)
3. Selección de características
4. Entrenamiento de un clasificador (KNN, NN o SVM)
5. Evaluación y análisis de resultados

MNIST contiene 70.000 imágenes de dígitos (0 a 9), en escala de grises, de 28×28. Se entrenan 60.000 y se prueban 10.000.

# PREPROCESAMIENTO

Cada imagen se representa en escala de grises (0 a 255). Se aplica LBP para convertir cada píxel en un patrón binario según sus vecinos, creando un histograma de 256 bins.

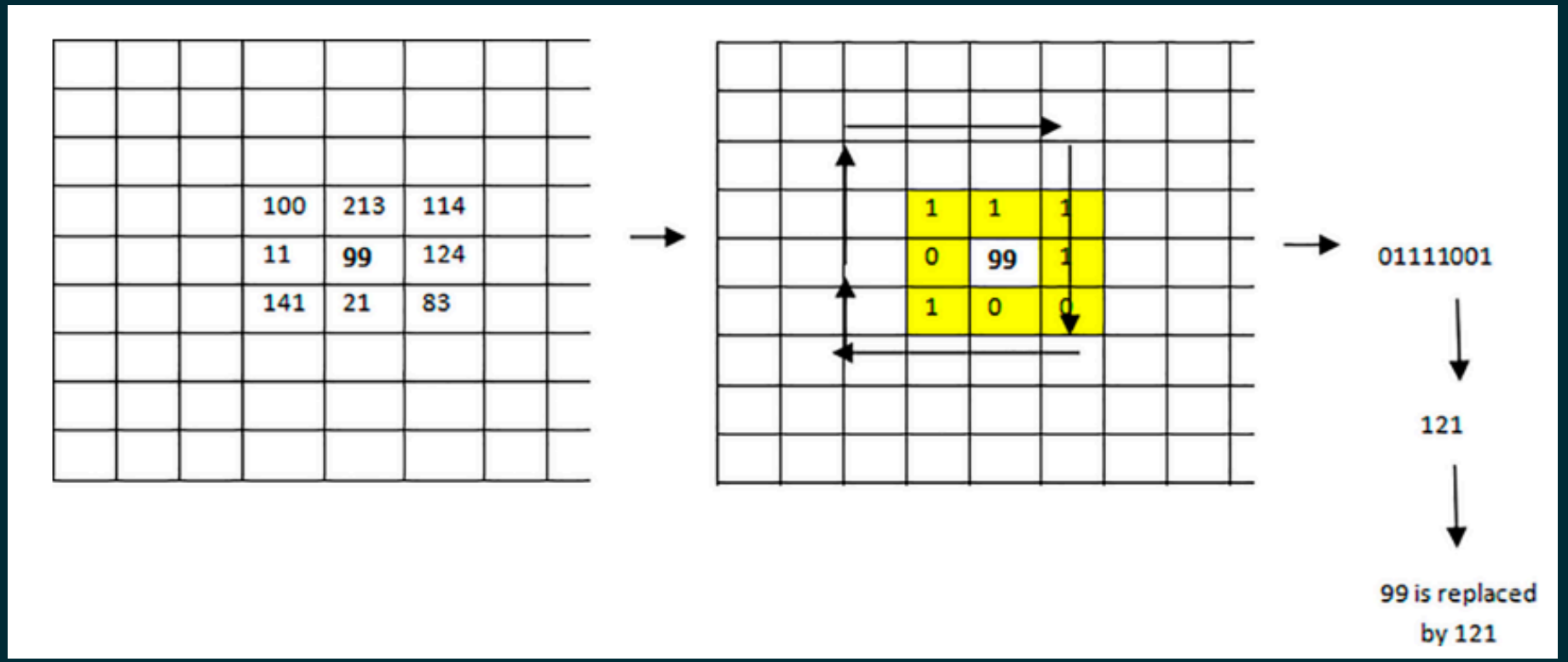



Figura 1-7: Computo LBP

# CONCLUSIÓN

En esta parte se ha presentado una visión general del Aprendizaje Automático, su evolución y sus tipos. También se han explorado métodos para la extracción y selección de características, y se describió la pipeline de ML con un caso de estudio sobre la clasificación de dígitos MNIST usando un enfoque tradicional (handcrafted features + clasificador).

No siempre es sencillo escoger el mejor método de extracción o selección de características, y ajustar cuidadosamente la relación entre bias y variance requiere esfuerzo. Para afrontar estos retos, el Deep Learning ofrece una alternativa que aprende automáticamente las características más representativas. El siguiente capítulo introduce precisamente el Deep Learning.

# REFERENCIAS


1. Definición informal de ML.
2. Definición de IA.
3. Definición formal de ML (Tom Mitchell).
4. Minsky, M. & Papert, S. (1969). *Perceptrons*. MIT Press.
5. Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). *Learning representations by back-propagating errors*. *Nature*, 323(6088), 533–536.

6. LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). *Gradient-based learning applied to document recognition*. *Proceedings of the IEEE*, 86(11), 2278–2324.
7. Hubel, D. H., & Wiesel, T. N. (1962). *Receptive fields, binocular interaction and functional architecture in the cat's visual cortex*. *Journal of Physiology*, 160, 106–154.
8. Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). *ImageNet classification with deep convolutional neural networks*. In *Proceedings of the 25th International Conference on Neural Information Processing Systems* (NeurIPS).
9. He, K., Zhang, X., Ren, S., & Sun, J. (2016). *Deep residual learning for image recognition*. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (CVPR).