



SEMANA 6

Técnicas de detección de anomalías.
Aprendizaje por refuerzo y control.
Parametrización automática y optimización
de algoritmos.

Prof. Luis Torrejón



Al finalizar, el alumno conocerá sobre el funcionamiento de las técnicas de detección de anomalías, el aprendizaje por refuerzo y control y la parametrización automática y optimización de algoritmos.





Recordemos - Comparación entre cerebro y computadoras

Estructura

El cerebro humano se compone de billones de neuronas vinculadas, en cambio, las computadoras se fundamentan en circuitos electrónicos y chips. Cada uno posee una configuración esencialmente distinta para el procesamiento de la información.

Paralelismo

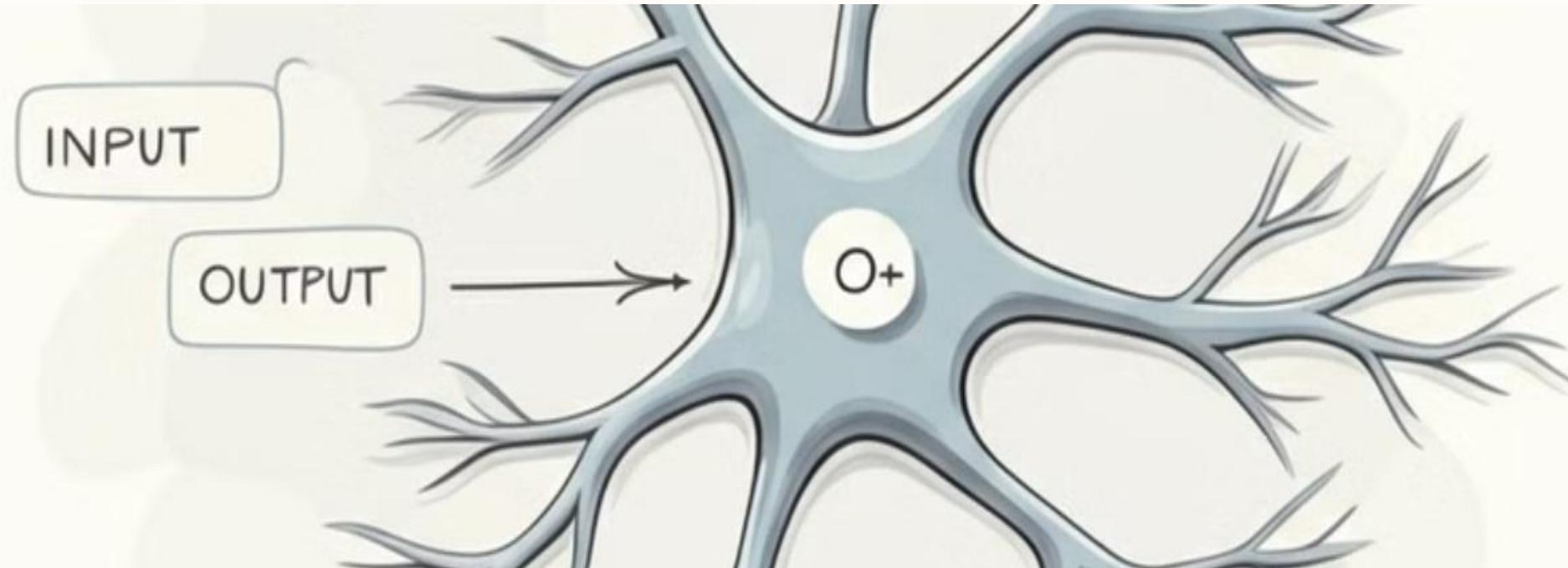
El cerebro opera de forma altamente paralela, llevando a cabo diversas funciones al mismo tiempo. Las computadoras convencionales manejan la información de forma secuencial, pese a que los progresos en arquitecturas paralelas han potenciado su habilidad para paralelismo.

Eficiencia energética

El cerebro es sumamente eficaz en la utilización de energía, consumiendo apenas 20 vatios, mientras que las computadoras actuales pueden necesitar cientos o miles de vatios para funcionar.



Recordemos - ¿Qué es una neurona artificial?



Modelo computacional

Una neurona artificial es un modelo computacional inspirado en el funcionamiento de las neuronas biológicas del cerebro humano. Busca imitar la forma en que las neuronas procesan y transmiten información.



Procesamiento de datos

Al igual que las neuronas biológicas, las neuronas artificiales reciben señales de entrada, las procesan y generan una señal de salida. Este proceso simula la capacidad de aprendizaje y adaptación del cerebro.

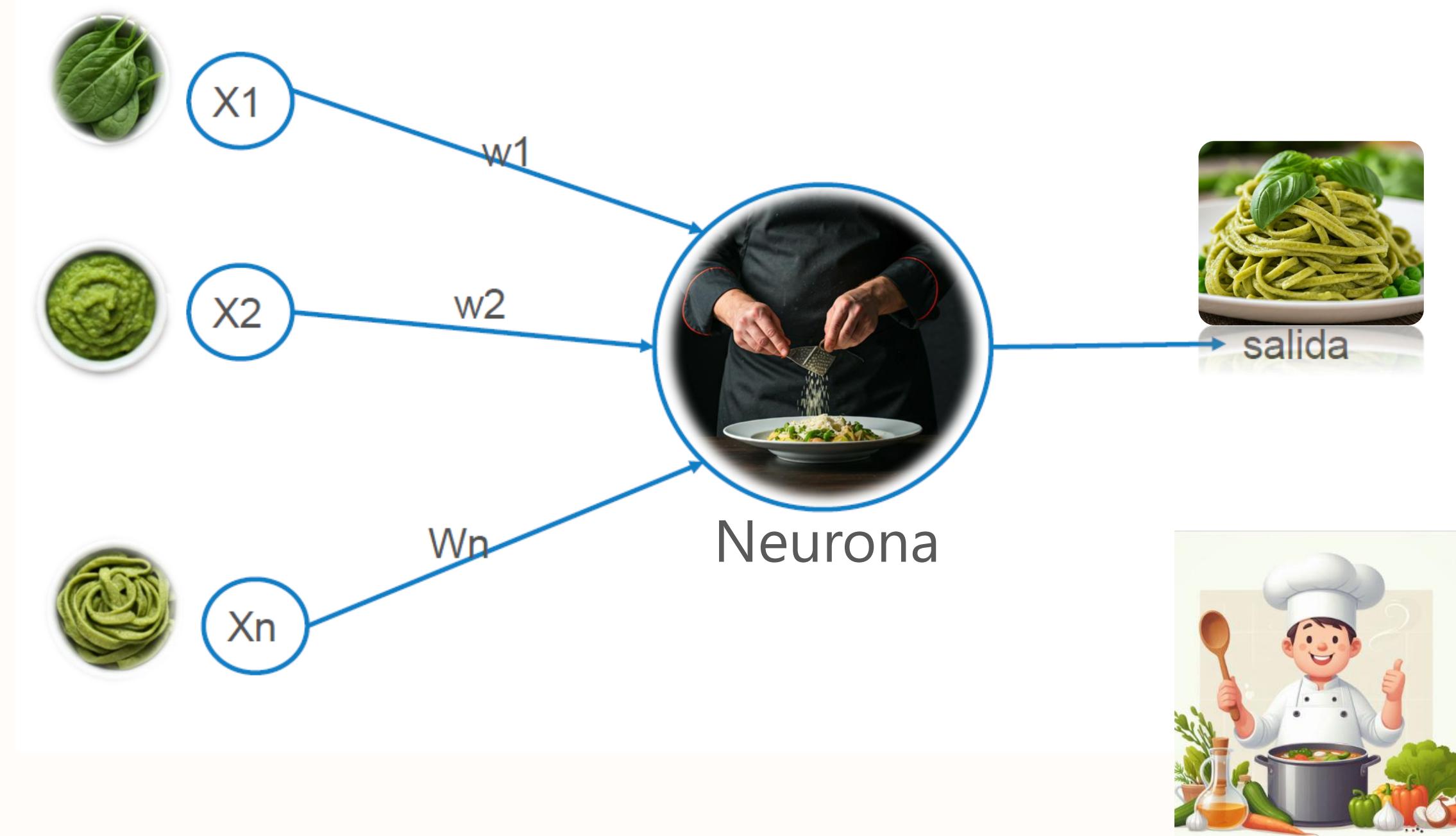


Conexiones entre neuronas

Las neuronas artificiales se conectan entre sí formando una red neuronal, asemejándose a la estructura del cerebro. Estas conexiones permiten que la red aprenda y resuelva problemas complejos.



Recordemos - ¿Qué es una neurona artificial?



Recordemos - Funciones de activación



Sigmoid



$$y = \frac{1}{1+e^{-x}}$$

Tanh



$$y = \tanh(x)$$

Step Function



$$y = \begin{cases} 0, & x < n \\ 1, & x \geq n \end{cases}$$

Softplus



$$y = \ln(1+e^x)$$

ReLU



$$y = \begin{cases} 0, & x < 0 \\ x, & x \geq 0 \end{cases}$$

Softsign



$$y = \frac{x}{(1+|x|)}$$

ELU



$$y = \begin{cases} \alpha(e^x-1), & x < 0 \\ x, & x \geq 0 \end{cases}$$

Log of Sigmoid



$$y = \ln\left(\frac{1}{1+e^{-x}}\right)$$

Swish



$$y = \frac{x}{1+e^{-x}}$$

Sinc



$$y = \frac{\sin(x)}{x}$$

Leaky ReLU



$$y = \max(0.1x, x)$$

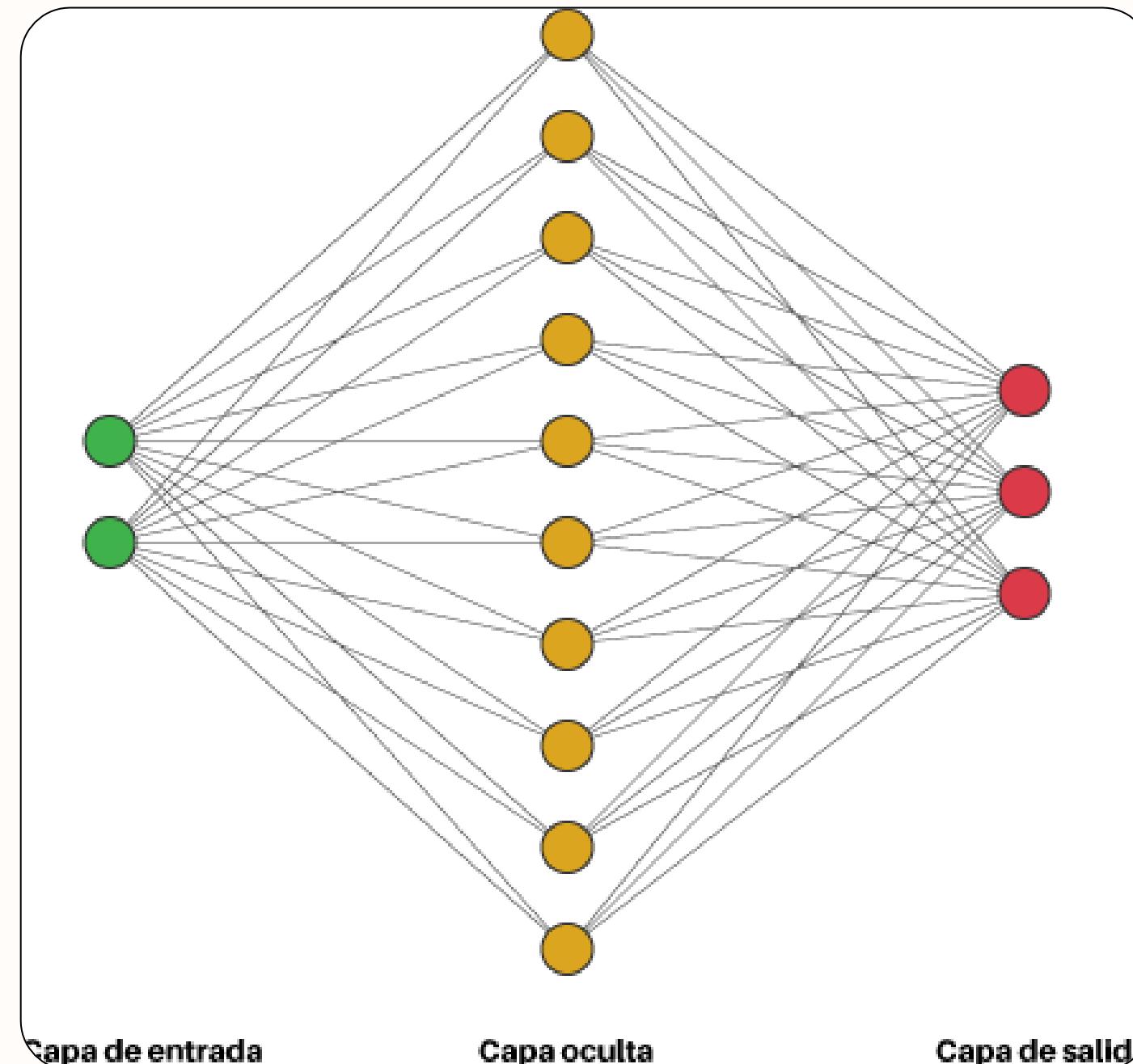
Mish



$$y = x(\tanh(\text{softplus}(x)))$$



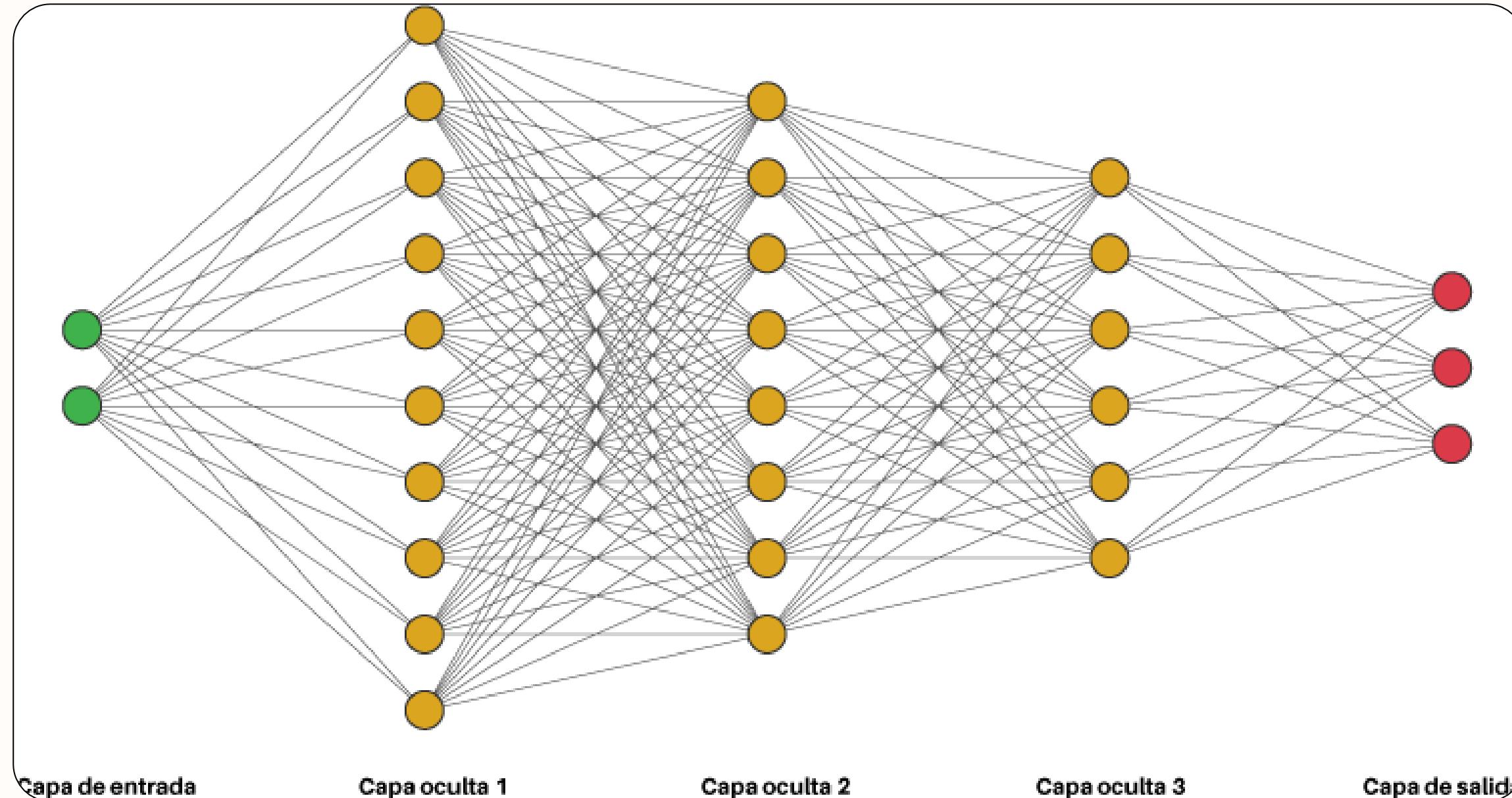
Recordemos - Redes neuronales simples



Fuente de la imagen: <https://codificandobits.com/blog/que-es-una-red-neuronal/>



Recordemos - Redes neuronales profundas



Fuente de la imagen: <https://codificandobits.com/blog/que-es-una-red-neuronal/>

Recordemos - Aprendizaje no supervisado

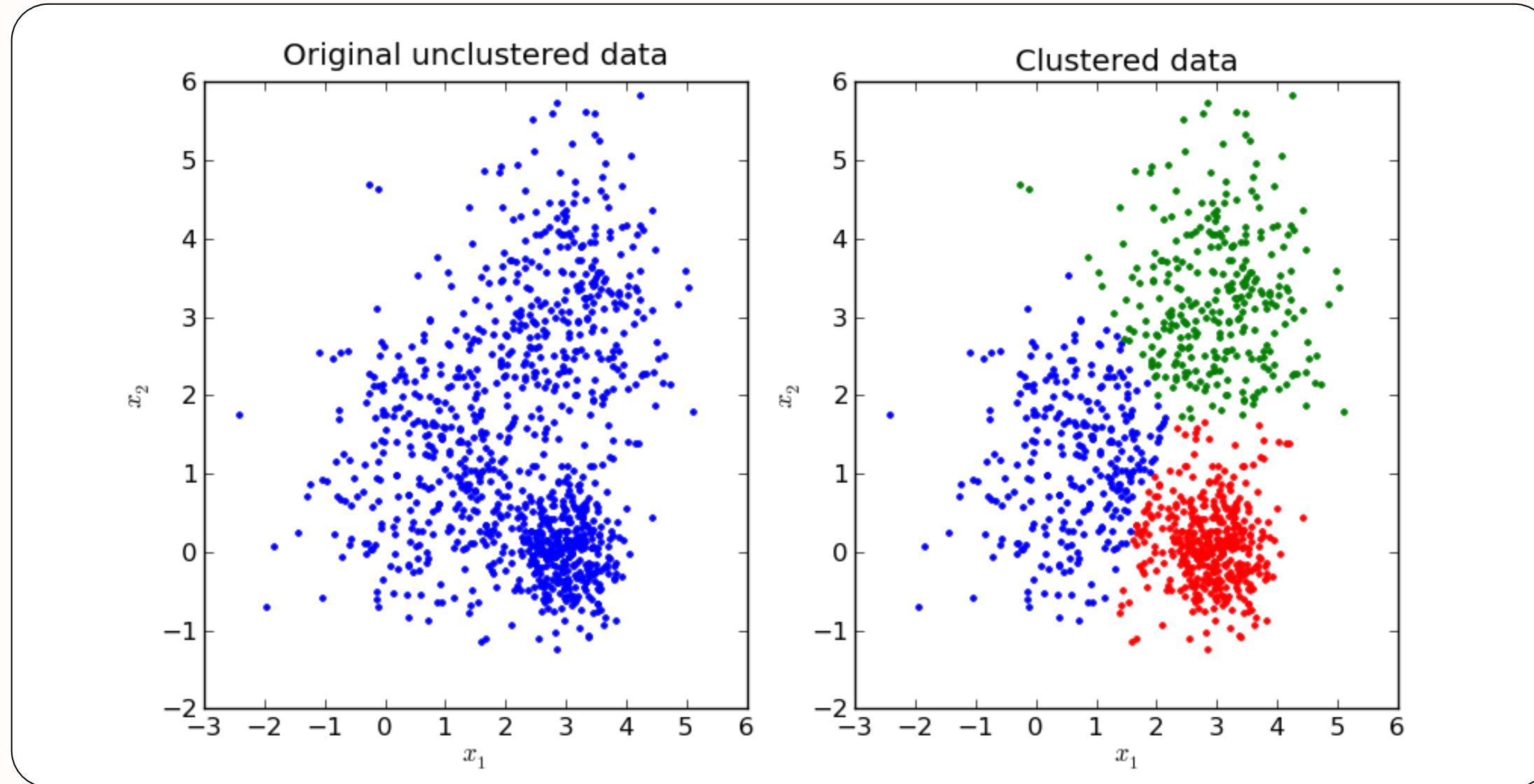


El aprendizaje no supervisado es una técnica de aprendizaje automático que se utiliza para encontrar patrones ocultos y estructuras en datos sin etiquetar.

A diferencia del aprendizaje supervisado, donde se proporcionan ejemplos etiquetados para entrenar un modelo, en el aprendizaje no supervisado el objetivo es descubrir por sí mismo la estructura inherente de los datos.



Recordemos - Clustering



https://pypr.sourceforge.net/_images/kmeans_2d.png



TEMARIO

- 1 — Técnicas de detección de anomalías.
- 2 — Aprendizaje por refuerzo y control.
- 3 — Parametrización automática y optimización de algoritmos.





Definición - Casos

En el ámbito de la seguridad cibernética:

Detección de intrusos en sistemas informáticos: Identificar patrones de acceso inusuales a sistemas o redes, como intentos de login fallidos repetidos desde direcciones IP desconocidas o accesos a archivos no autorizados.

Análisis de tráfico de red: Detectar picos inusuales en el tráfico de red, conexiones a puertos desconocidos o protocolos poco comunes, lo cual podría indicar un ataque en curso.

Análisis de logs de aplicaciones: Identificar errores o excepciones inusuales en los logs de aplicaciones que puedan indicar vulnerabilidades o ataques.

En el ámbito industrial:

Mantenimiento predictivo: Detectar cambios anómalos en las lecturas de sensores de maquinaria industrial para predecir fallas antes de que ocurran, evitando costosas paradas de producción.

Control de calidad: Identificar productos defectuosos en una línea de producción mediante el análisis de imágenes o datos de sensores.

Monitoreo de procesos industriales: Detectar desviaciones en los parámetros de un proceso industrial que puedan indicar problemas de calidad o seguridad.

En el ámbito financiero:

Detección de fraudes en seguros: Identificar reclamos fraudulentos mediante el análisis de patrones de comportamiento inusuales en los datos de los asegurados.

Monitoreo de mercados financieros: Detectar fluctuaciones inusuales en los precios de las acciones o divisas que puedan indicar manipulación del mercado o eventos inesperados.

En el ámbito del medio ambiente:

Monitoreo de la calidad del aire: Identificar picos inusuales en los niveles de contaminantes en el aire, lo cual podría indicar una emergencia ambiental.

Monitoreo de la calidad del agua: Detectar cambios anómalos en los parámetros del agua, como la temperatura, el pH o la concentración de sustancias químicas, lo cual podría indicar contaminación.

Técnicas de detección de anomalías



La detección de anomalías es un campo importante en el análisis de datos y la minería de datos, cuyo objetivo es identificar patrones inusuales o poco comunes que se desvían significativamente del comportamiento esperado.

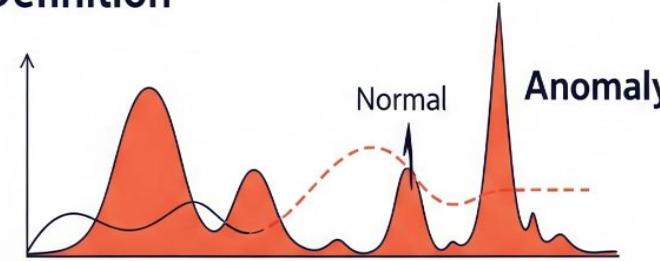
Estas técnicas son fundamentales para la detección de fraudes, la seguridad informática, el control de calidad y muchas otras aplicaciones.





¿Qué son las anomalías?

Definition



Detection & Action



Importance



Fraud Prevention



System Failure Detection



Cybersecurity

Definición

Las anomalías son patrones de datos, comportamientos o eventos que se desvían significativamente de lo que se considera "normal" o esperado. Representan observaciones atípicas que se distinguen del resto de los datos.

Detección

La detección de anomalías implica identificar estos patrones anormales dentro de conjuntos de datos, con el objetivo de comprender mejor los fenómenos subyacentes y tomar medidas apropiadas.

Importancia

La detección de anomalías es crucial en diversas áreas, como la prevención de fraudes, la detección de fallos en sistemas, la seguridad cibernética y el análisis de comportamiento del usuario, entre otras.

Importancia



1

Identificar amenazas y riesgos

La detección de anomalías permite identificar patrones inusuales o comportamientos sospechosos que pueden indicar fraude, ataques cibernéticos, fallos de producción u otros problemas potencialmente dañinos.

2

Optimizar procesos y tomar decisiones informadas

Al analizar las anomalías, las empresas pueden identificar áreas de mejora, optimizar procesos y tomar decisiones más acertadas para maximizar la eficiencia y la rentabilidad.

3

Mejorar la seguridad y prevenir problemas

La detección temprana de anomalías ayuda a las organizaciones a anticipar y prevenir problemas antes de que causen daños significativos, mejorando la seguridad general y la confianza de los clientes.

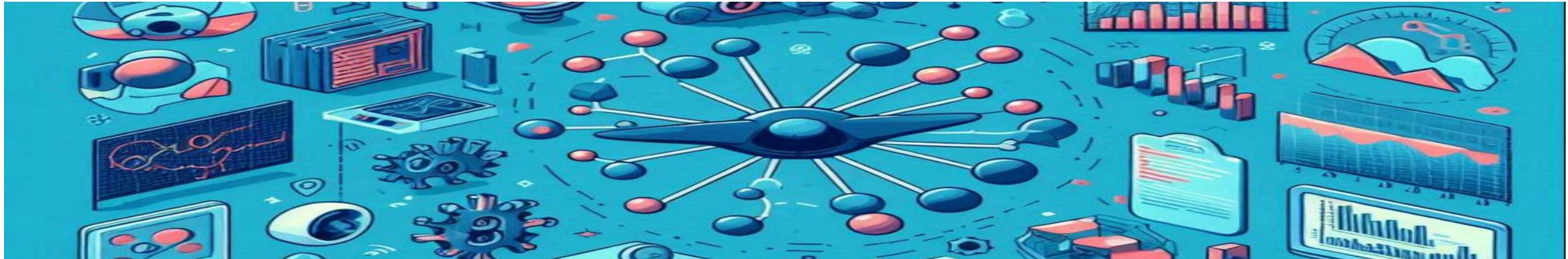
4

Potenciar la innovación y el crecimiento

Al identificar patrones inesperados, la detección de anomalías puede inspirar nuevas ideas y oportunidades, lo que lleva a la innovación y al crecimiento a largo plazo.



Tipos de detección de anomalías



Outlier detection (Detección de valores atípicos)

- **Datos de entrenamiento:** Incluyen tanto datos normales como valores atípicos.
- **Objetivo:** Identificar los valores atípicos existentes en el conjunto de datos.
- **Ejemplo:** En un conjunto de datos de transacciones bancarias, un gasto inusualmente alto en un lugar poco frecuente podría ser considerado un outlier y potencialmente un fraude.

Novelty detection (Detección de novedades)

- **Datos de entrenamiento:** Solo incluyen datos normales.
- **Objetivo:** Identificar nuevos datos que no se ajustan al patrón aprendido a partir de los datos normales.
- **Ejemplo:** En un sistema de monitoreo de maquinaria, el modelo se entrena con datos de funcionamiento normal. Si aparece una nueva lectura de un sensor que no se ajusta a ese patrón, se considera una novedad y podría indicar una falla inminente.



¿Cómo funciona en la práctica?

Aseguradora cuenta con una gran base de datos de reclamos históricos. Cada reclamo está compuesto por una serie de características como:

- **Datos del asegurado:** Edad, género, historial de reclamos previos, tipo de póliza, etc.
- **Detalles del siniestro:** Fecha, lugar, descripción del daño, monto del reclamo, etc.
- **Información del taller o proveedor:** Historial de pagos, tipo de servicios, etc.

Se puede identificar:

- **Patrones normales:** Identificar que la mayoría de los reclamos por robo de automóviles ocurren en ciertas zonas de la ciudad durante las horas nocturnas.
- **Anomalías:** Reclamos que se desvían significativamente de estos patrones normales se marcan como posibles fraudes. Por ejemplo, un reclamo por robo de un auto modelo antiguo en una zona rural a las 3 de la mañana podría despertar sospechas.



¿Cómo funciona en la práctica?

Empresa con una extensa red informática. Analizar el tráfico de red para identificar patrones inusuales que podrían indicar un ataque cibernético.

- **Horarios de conexión:** A qué horas del día son comunes las conexiones desde diferentes dispositivos y ubicaciones.
- **Tipos de tráfico:** Qué tipo de datos se transmiten con mayor frecuencia (correo electrónico, navegación web, transferencias de archivos, etc.).
- **Volumetría:** Cuál es el volumen normal de tráfico en diferentes momentos del día.

Se puede identificar:

- **Intentos de acceso no autorizados:** Conexiones desde direcciones IP desconocidas o inusuales.
- **Escaneo de puertos:** Actividades que buscan vulnerabilidades en los sistemas de la red.
- **Infiltración de malware:** Comportamientos inusuales de los dispositivos infectados.
- **Exfiltración de datos:** Transferencias de grandes cantidades de datos hacia el exterior de la red.

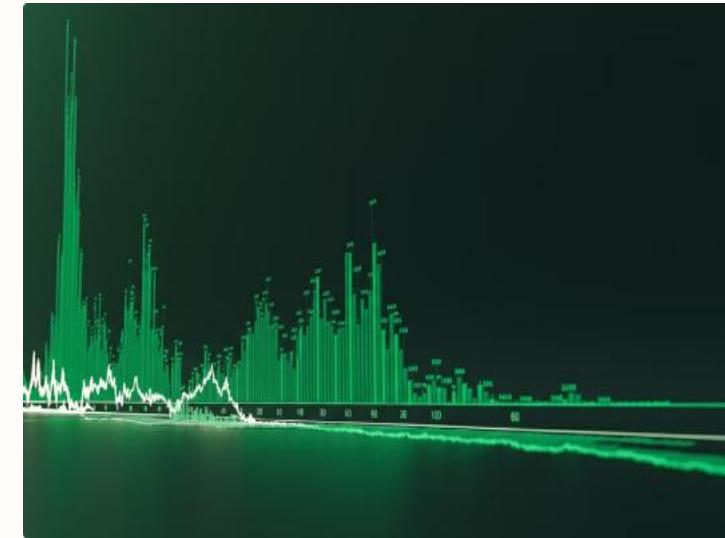
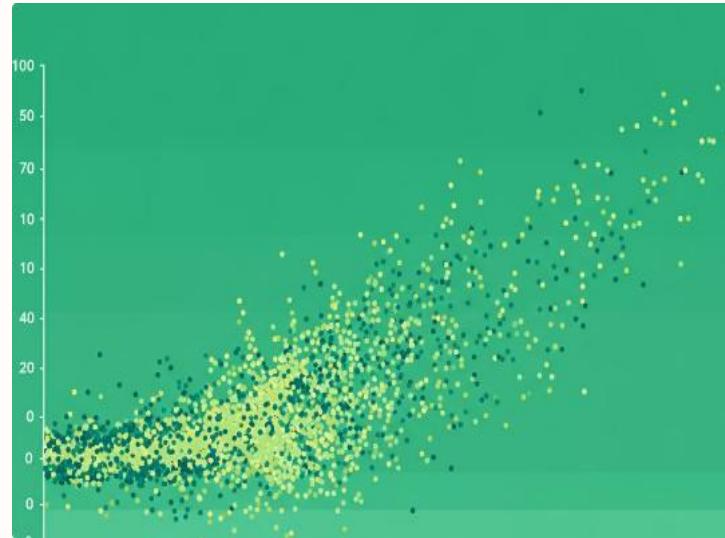
Retos

- Establecer la región de normalidad
- Determinar qué se considera "normal" frente a lo "anómalo" puede ser ambiguo y depende del contexto.
- La definición de anomalía varía según el área de aplicación.
- Es esencial contar con datos suficientes y representativos para entrenar el modelo.





Tipos de anomalías



Anomalías Puntuales

Las anomalías puntuales son observaciones que se desvían significativamente del comportamiento normal de los datos. Pueden ser valores atípicos aislados, picos repentinos o caídas drásticas que no siguen el patrón general de los datos.

Anomalías Contextuales

Las anomalías contextuales se definen por su relación con el contexto de los datos. Estos tipos de anomalías ocurren cuando una observación es anormal en un contexto específico, pero normal en otro. Por ejemplo, un valor alto de temperatura en verano vs. en invierno.

Anomalías Colectivas

Las anomalías colectivas se refieren a un grupo de observaciones que, en conjunto, se desvían del comportamiento normal, incluso si individualmente no se considerarían anómalas. Estas anomalías se detectan al identificar patrones inusuales o tendencias en un conjunto de datos.

Enfoques de detección de anomalías



Detección basada en estadística

Estos métodos se basan en la identificación de valores atípicos o desviaciones significativas con respecto a un modelo estadístico de referencia. Utilizan técnicas como análisis de distribución de datos, análisis de regresión y pruebas de hipótesis para detectar patrones anormales.

Detección basada en aprendizaje de máquina

Estos enfoques utilizan algoritmos de aprendizaje automático para aprender modelos a partir de datos históricos y, posteriormente, identificar instancias que se desvían significativamente de dichos modelos. Incluyen técnicas de clasificación, agrupamiento y aprendizaje de representaciones.

Detección basada en redes neuronales

Estos métodos aprovechan la capacidad de las redes neuronales para aprender representaciones compactas y detectar anomalías a partir de reconstrucciones deficientes o generación de datos sintéticos anormales. Arquitecturas como autoencoders y redes generativas adversarias (GANs) son ampliamente utilizadas.

Detección en series de tiempo

Estos enfoques se centran en la identificación de patrones anormales en datos secuenciales, como series temporales. Utilizan técnicas como modelos de predicción, descomposición de señales y detección de cambios estructurales.

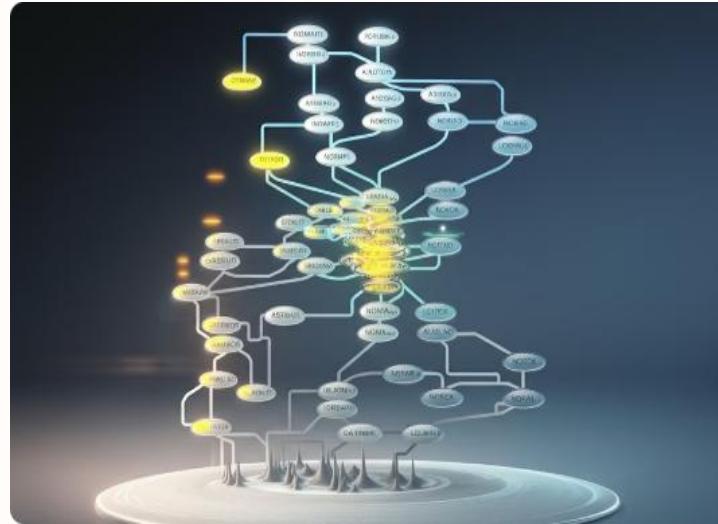
Metodología

- 1. Recopilación de Datos:** Reunir datos relevantes y de calidad.
 - 2. Preprocesamiento:** Limpiar, normalizar, y transformar los datos.
 - 3. Selección de Técnicas:** Elegir la metodología basada en el tipo de datos y anomalías esperadas.
 - 4. Implementación del Modelo:** Entrenar y validar el modelo.
 - 5. Evaluación y Ajustes:** Analizar la efectividad y refinar.



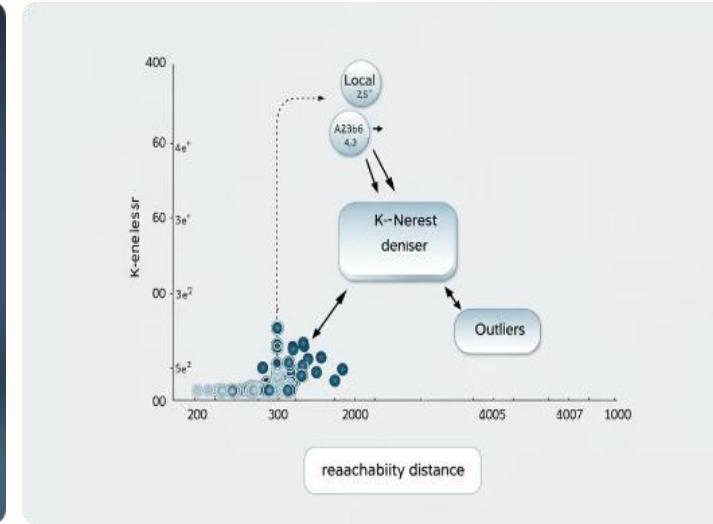


Tipos de algoritmos



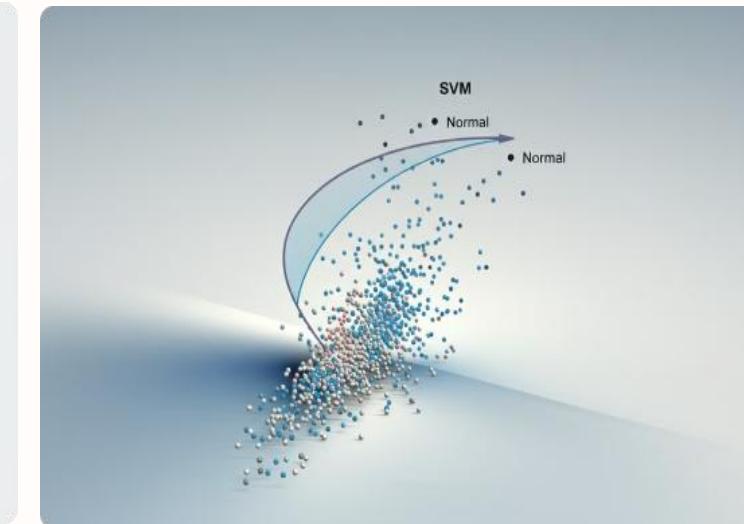
Isolation Forest

El algoritmo Isolation Forest se basa en la premisa de que las anomalías son diferentes del resto de los datos y, por lo tanto, son más fáciles de aislar. Identifica los puntos anómalos aislando aleatoriamente los valores de los atributos y midiendo la distancia de los puntos desde la raíz.



Local Outlier Factor (LOF)

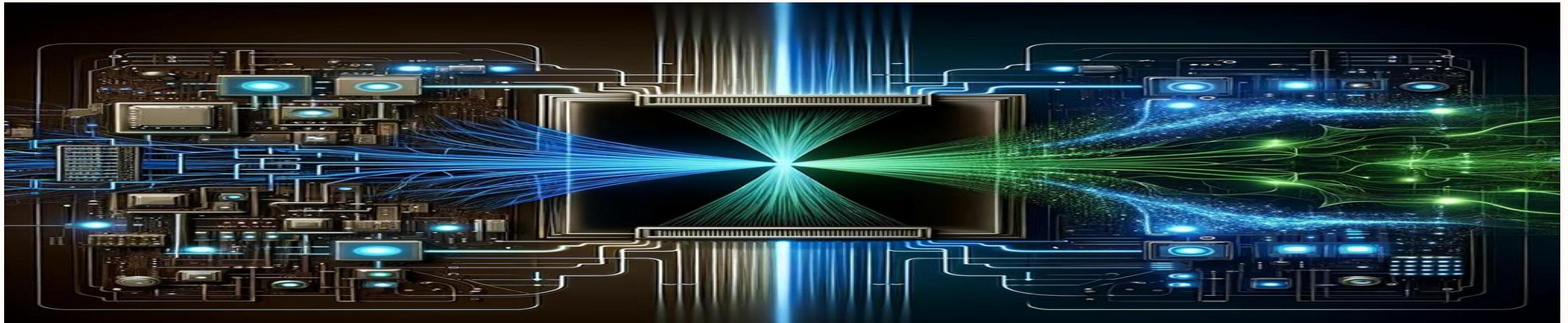
LOF es un algoritmo que identifica las anomalías basándose en la densidad local de los datos. Calcula un puntaje de anomalía para cada punto en función de la densidad de sus vecinos, lo que permite detectar puntos atípicos en conjuntos de datos heterogéneos.



One-Class SVM

One-Class SVM es un método de aprendizaje automático que se utiliza para la detección de anomalías. Construye un hiperplano que separa los datos normales de las anomalías, permitiendo identificar puntos fuera de ese hiperplano como outliers.

Tipos de algoritmos



Autoencoders

1 ¿Qué son los autoencoders?

Los autoencoders son un tipo de red neuronal artificial que se utiliza para el aprendizaje no supervisado de representaciones. Su objetivo es aprender una codificación compacta y eficiente de los datos de entrada, permitiendo la reconstrucción de esos datos a partir de la representación codificada.

2 Aplicaciones en detección de anomalías

Los autoencoders se pueden utilizar para la detección de anomalías al entrenarlos con datos normales y luego identificar aquellas muestras que no puedan ser reconstruidas con precisión, lo que indica la presencia de un comportamiento anómalo.

3 Ventajas de los autoencoders

Una de las principales ventajas de los autoencoders es su capacidad para aprender representaciones de los datos sin necesidad de etiquetas. Además, son flexibles y pueden adaptarse a una amplia variedad de problemas, desde la detección de fraude hasta la identificación de anomalías en imágenes y series temporales.



Tipos de algoritmos

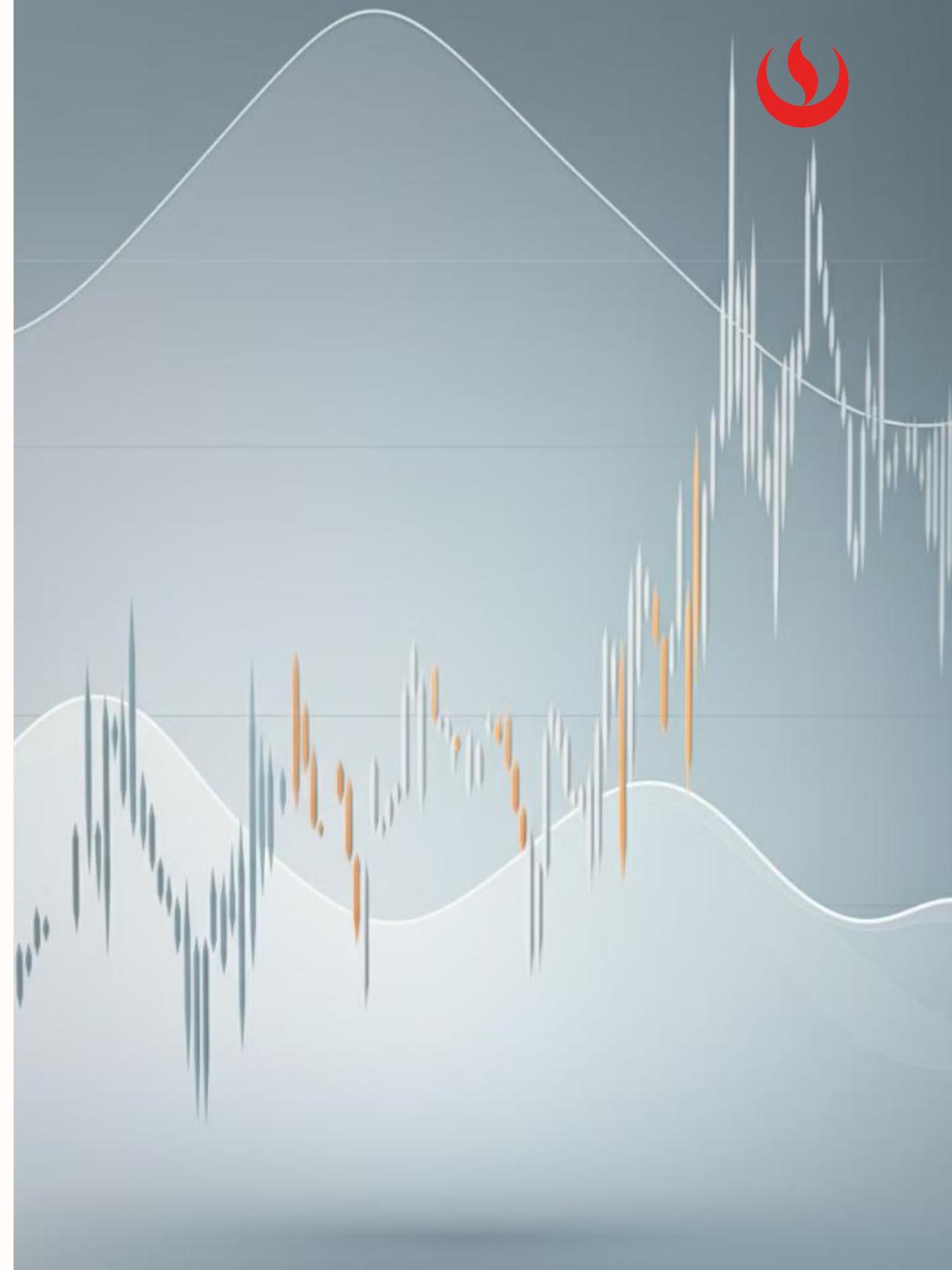
Detección de Anomalías en Series Temporales

LSTM (Long Short-Term Memory)

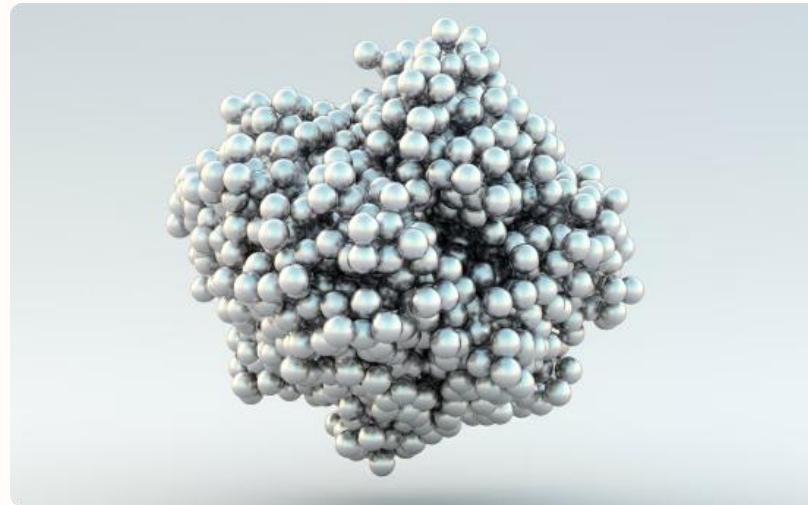
Son redes neuronales recurrentes capaces de aprender patrones complejos en series temporales. Pueden detectar anomalías identificando desviaciones significativas de los patrones aprendidos.

ARIMA (Autorregresivo Integrado de Media Móvil)

Los modelos ARIMA modelan la estructura de correlación en los datos de series temporales. Las anomalías se detectan como observaciones que no se ajustan bien al modelo.



Tipos de algoritmos



Algoritmo DBSCAN

DBSCAN es un algoritmo de agrupamiento (clustering) basado en densidad que identifica grupos de datos con alta densidad y separa eficazmente los datos atípicos o anomalías. A diferencia de otros algoritmos, DBSCAN no requiere que se especifique el número de grupos de antemano.



Detección de Anomalías

En el ámbito de la seguridad informática, DBSCAN se utiliza para detectar patrones de tráfico de red anómalos que pueden indicar actividades maliciosas, como ataques cibernéticos o intrusiones. Al identificar estos comportamientos fuera de lo común, DBSCAN ayuda a los equipos de seguridad a responder rápidamente y mitigar posibles amenazas.



Aplicación en IoT y Manufactura

En el contexto de IoT y manufactura, DBSCAN se emplea para identificar patrones anómalos en los datos de sensores y procesos de producción. Esto permite detectar problemas en tiempo real, como fallos en maquinaria, desviaciones en la calidad o ineficiencias en los procesos, lo que ayuda a optimizar la producción y minimizar los costos.

Tipos de algoritmos



Arquitectura GAN

Las Redes Adversarias Generativas (GANs) constan de dos modelos de aprendizaje profundo que compiten entre sí: un generador que crea muestras artificiales y un discriminador que intenta distinguir entre muestras reales y generadas. Esta dinámica competitiva permite a las GANs generar contenido altamente realista e innovador.



Generación de Imágenes Realistas

Una de las aplicaciones más destacadas de las GANs es la generación de imágenes fotorrealistas. Estos modelos pueden crear rostros, escenas y objetos que parecen auténticos, lo que los convierte en una herramienta poderosa para la creación de contenido visual, la síntesis de datos y diversas aplicaciones artísticas.



Creación de Arte Generativo

Más allá de la generación de imágenes realistas, las GANs también pueden producir arte digital y obras de arte generativas de carácter más abstracto y surrealista. Estos modelos son capaces de generar patrones, formas y composiciones únicas que desafían los límites de la creatividad humana.



Python IA





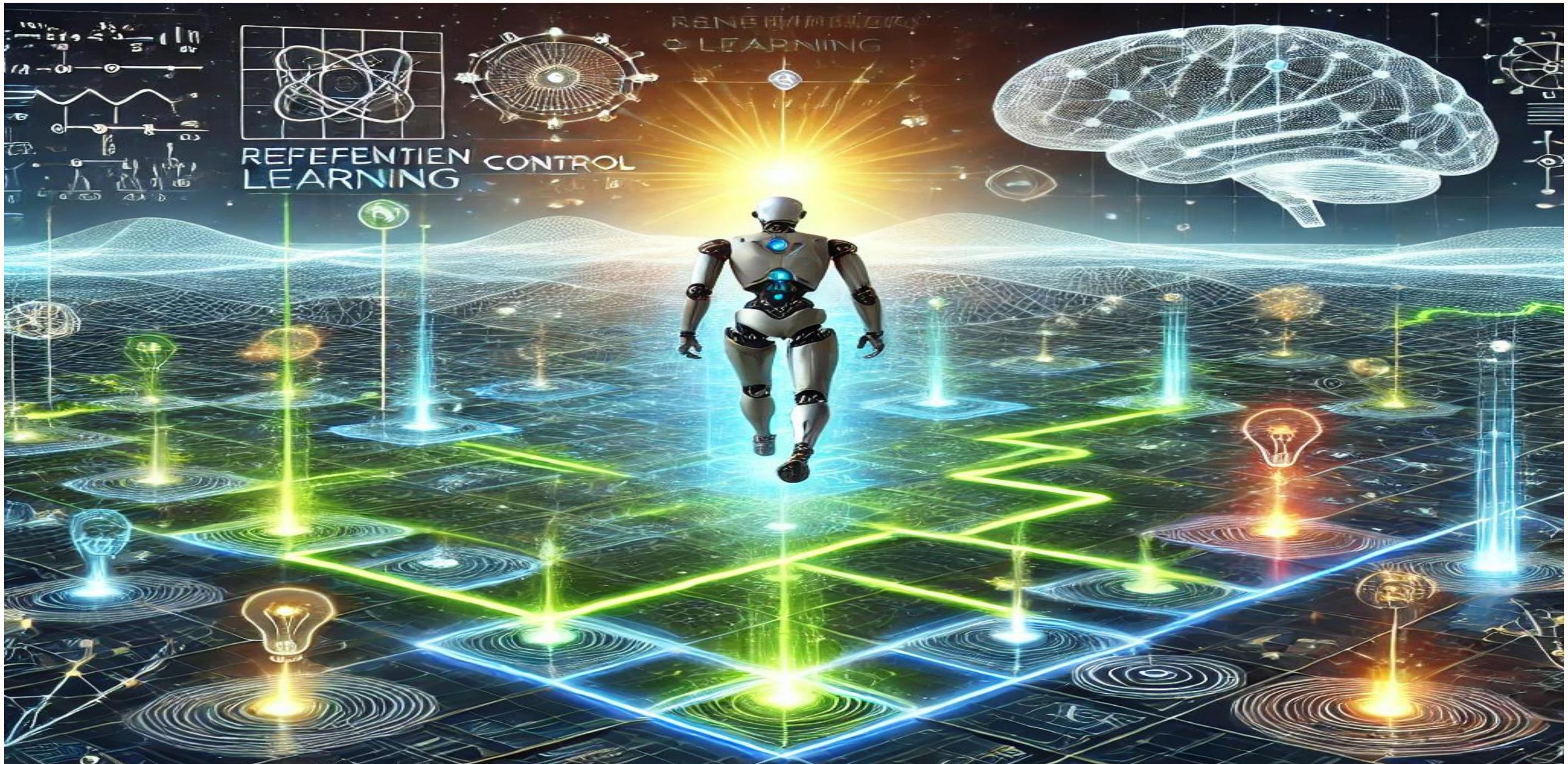
TEMARIO

- 1 — Técnicas de detección de anomalías.
- 2 — Aprendizaje por refuerzo y control.
- 3 — Parametrización automática y optimización de algoritmos.





Aprendizaje por refuerzo y control



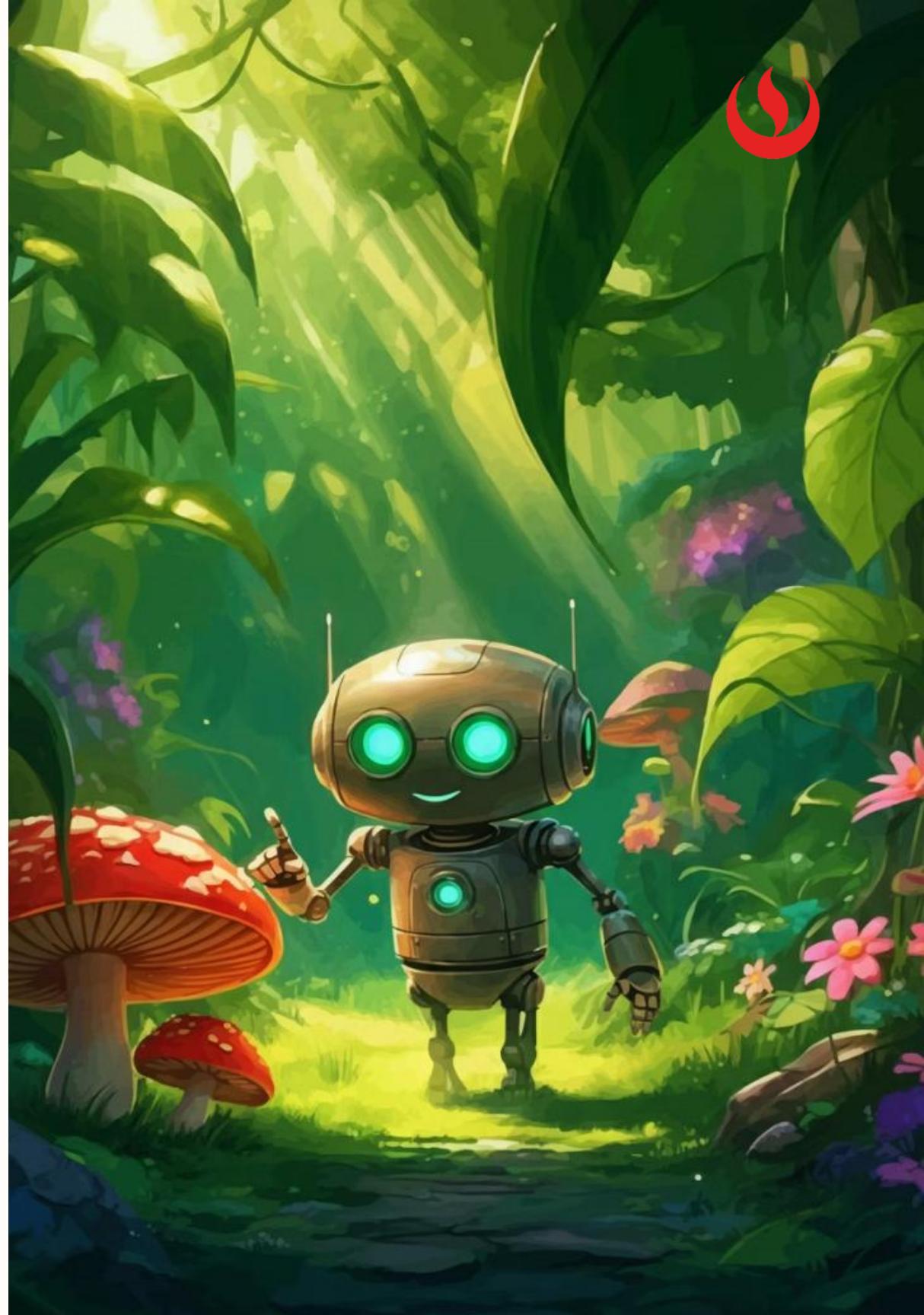


Definición

El aprendizaje por refuerzo y control es una técnica de aprendizaje automático en la que un agente software aprende a través de interacciones con un entorno. El agente recibe recompensas o castigos por sus acciones, lo que le permite ajustar su comportamiento para maximizar las recompensas a lo largo del tiempo.

Este enfoque se basa en la idea de que el aprendizaje se produce a través de la experiencia y la retroalimentación. El agente explora su entorno, toma acciones y recibe recompensas o castigos, lo que le permite mejorar gradualmente su desempeño y tomar mejores decisiones en el futuro.

<https://www.youtube.com/watch?v=KPLYhRBCCvk>





Definición

Distinciones con el aprendizaje supervisado

- Los algoritmos de aprendizaje por refuerzo obtienen retroalimentación del entorno sobre qué acciones son adecuadas o inadecuadas.
- En el aprendizaje supervisado, se optimiza una acción específica para alcanzar un objetivo inmediato, sin considerar las consecuencias de acciones futuras.
- En el aprendizaje por refuerzo, el agente enfrenta un número considerable de posibles combinaciones de acciones para alcanzar su meta

Aspecto	Aprendizaje Supervisado	Aprendizaje por Refuerzo
Retroalimentación	Recibe la respuesta correcta de los datos	Recibe recompensas o castigos del entorno
Optimización	Optimiza una acción específica	Considera el impacto a largo plazo de cada acción
Combinación de Acciones	No tiene en cuenta secuencias de acciones	Debe evaluar una gran cantidad de combinaciones posibles



Definición

Distinciones con el aprendizaje no supervisado

- En el aprendizaje por refuerzo, hay una conexión entre la entrada y la salida que no se observa en el aprendizaje no supervisado.
- En el aprendizaje no supervisado, el propósito es identificar patrones subyacentes, en lugar de establecer una relación entre acción y resultado.

Aspecto	Aprendizaje por Refuerzo	Aprendizaje No Supervisado
Relación entrada-salida	Existe una relación clara entre las acciones (entrada) y las recompensas o castigos (salida).	El algoritmo busca encontrar patrones ocultos en los datos sin ninguna guía externa.
Objetivo del aprendizaje	Aprender a tomar decisiones secuenciales para maximizar una recompensa a largo plazo.	El objetivo es encontrar patrones interesantes sin una tarea específica.



Ejemplos de aplicaciones

Juego de Atari

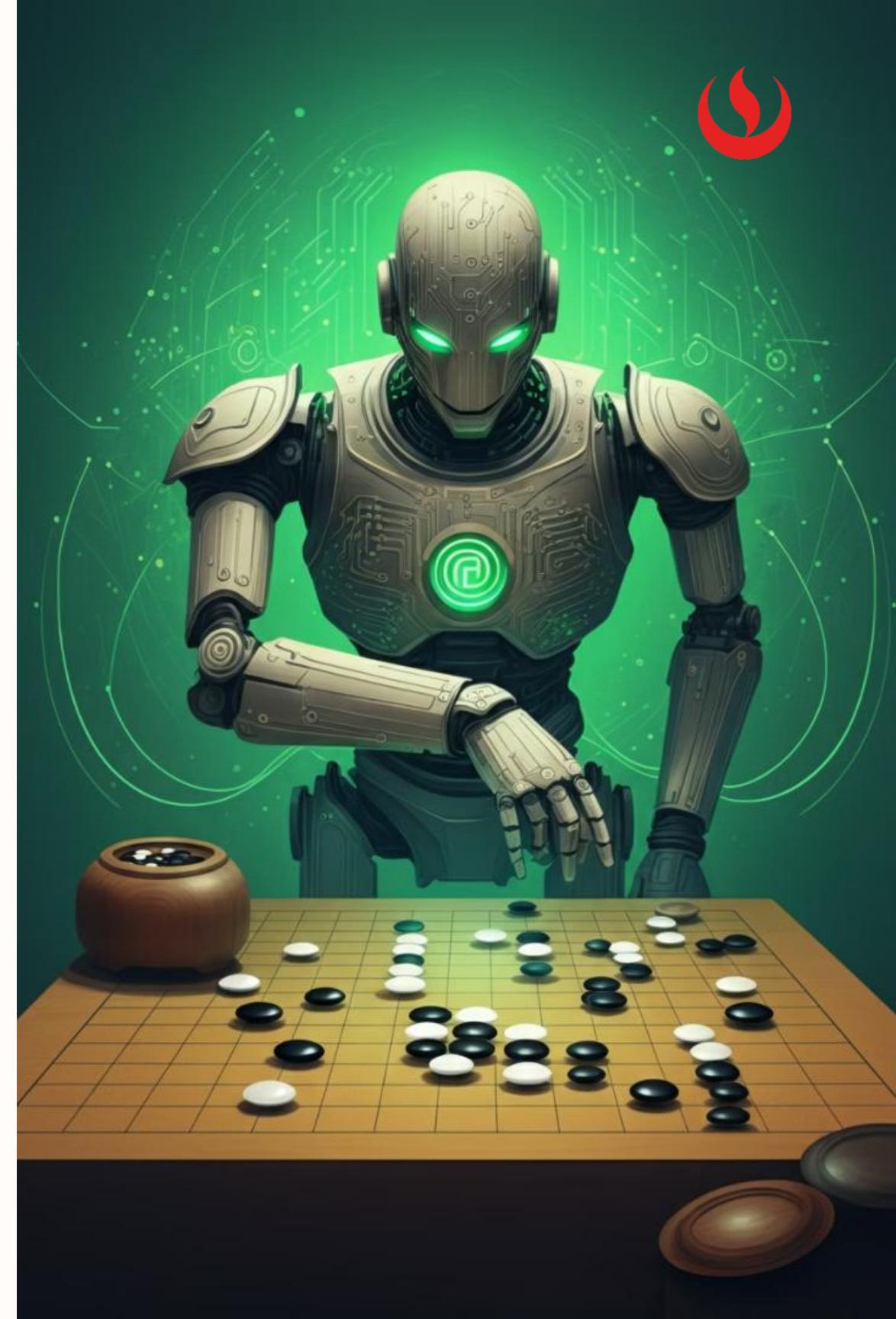
Los algoritmos de aprendizaje por refuerzo han logrado superar a los humanos en una amplia gama de juegos clásicos de Atari, como Breakout y Pong, aprendiendo a través de la interacción con el entorno y la maximización de las recompensas.

AlphaGo

El sistema de inteligencia artificial AlphaGo, desarrollado por Google DeepMind, utilizó técnicas de aprendizaje por refuerzo para derrotar al campeón mundial de Go, Lee Sedol, demostrando la capacidad de esta tecnología para dominar juegos complejos de estrategia.

Conducción autónoma

Los vehículos autónomos aprovechan el aprendizaje por refuerzo para tomar decisiones en tiempo real, navegando por entornos complejos y adaptándose a situaciones impredecibles en la carretera.



Elementos principales



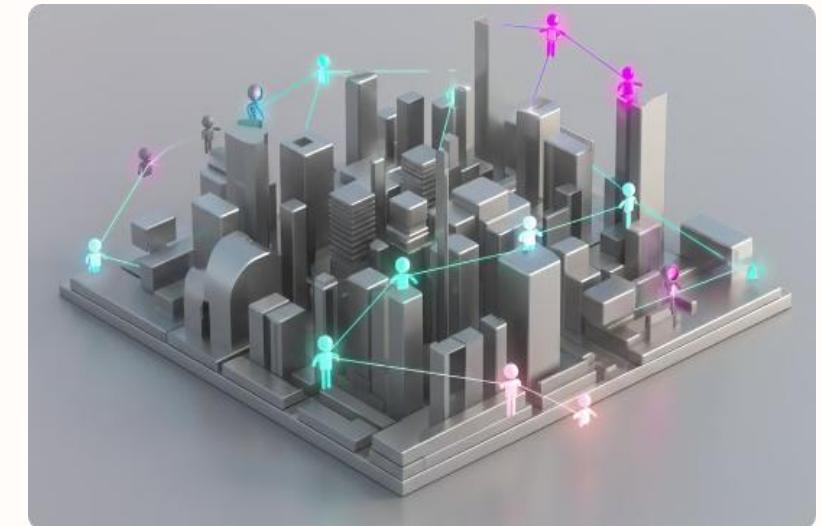
Agente y entorno

El agente, ya sea un robot, un programa de software o un ser humano, interactúa con un entorno y recibe recompensas o penalizaciones en función de sus acciones. Esto motiva al agente a aprender y tomar decisiones que maximicen las recompensas a largo plazo.



Exploración vs Explotación

El agente debe encontrar un equilibrio entre explorar nuevas opciones y explotar lo que ya ha aprendido. Esto implica un delicado balance entre la toma de riesgos y la optimización de los resultados.



Función de valor

El agente debe aprender a estimar el valor futuro de sus acciones, lo que se conoce como la función de valor. Esta función ayuda al agente a tomar decisiones que maximicen sus recompensas a largo plazo.

Elementos principales



Función de recompensa

Es el mecanismo que define las metas u objetivos que el agente de aprendizaje por refuerzo debe maximizar. Esta función determina qué acciones son deseables y cuáles no.



Política

Es la estrategia que el agente usa para seleccionar las acciones a tomar en cada estado del entorno. La política se va refinando a través del proceso de aprendizaje para mejorar la toma de decisiones.



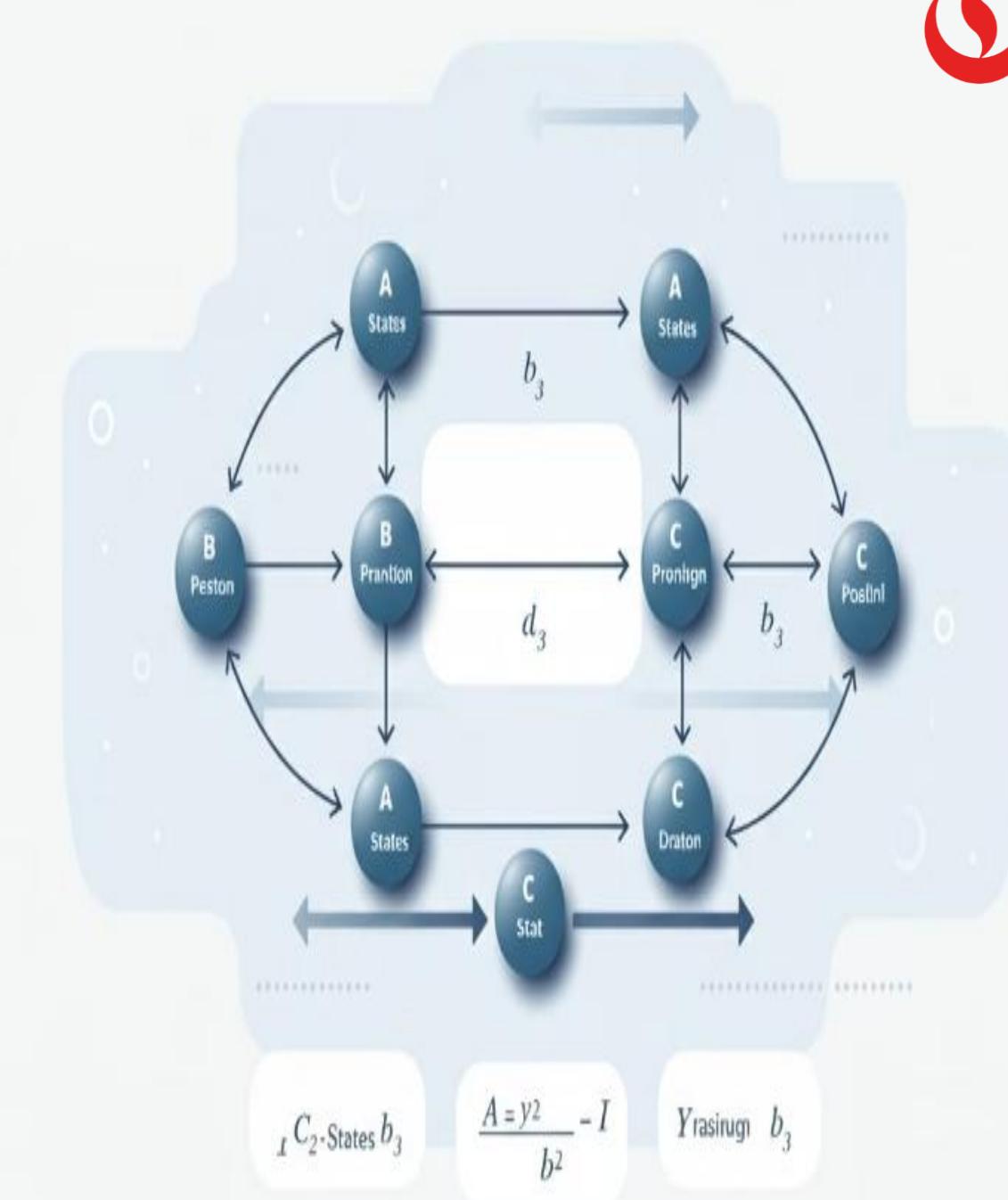
Modelo de entorno (opcional)

Es una representación matemática que describe cómo un agente interactúa con su entorno. Es una especie de "mapa" que el agente utiliza para comprender las consecuencias de sus acciones.



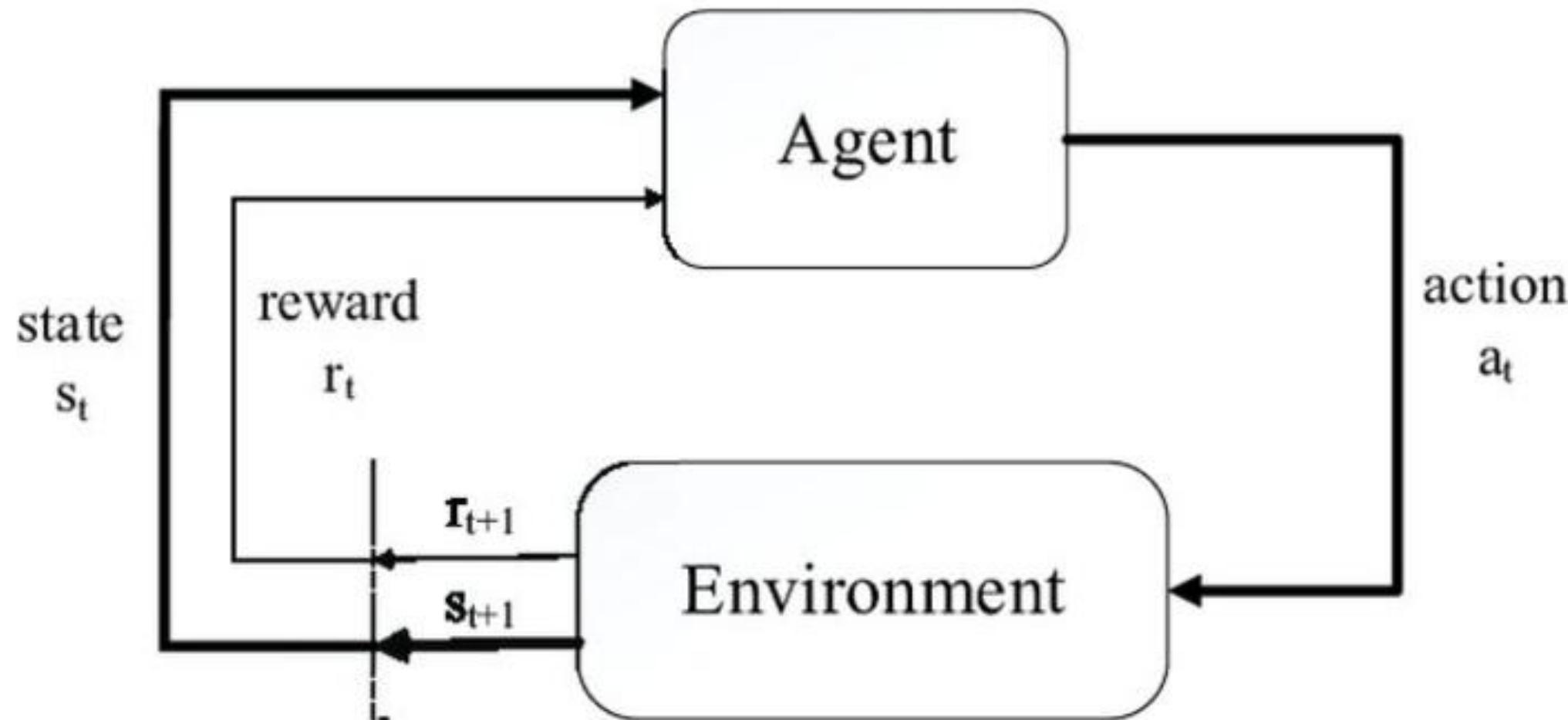
Problema de Markov

- El problema de Markov es un modelo matemático que describe un proceso de toma de decisiones secuencial en el que el resultado de cada acción depende únicamente del estado actual del sistema, y no de los estados o acciones anteriores. Esto se conoce como la propiedad de Markov.
- Este modelo se utiliza ampliamente en los algoritmos de aprendizaje por refuerzo, ya que permite simplificar el proceso de toma de decisiones al considerar solo el estado actual, sin tener que almacenar o procesar toda la historia del sistema.





Marco de aprendizaje por refuerzo



[https://www.researchgate.net/publication/317160630 Integrating Reinforcement Learning with Multi-Agent Techniques for Adaptive Service Composition](https://www.researchgate.net/publication/317160630_Integrating_Reinforcement_Learning_with_Multi-Agent_Techniques_for_Adaptive_Service_Composition)

Papers



- Ferens, M. (2020). Enrutamiento y establecimiento dinámico de conexiones en redes de transporte mediante aprendizaje por refuerzo. <https://uvadoc.uva.es/handle/10324/42740>
- XiaoqiangLi, ChunxiaoSong, ShipengZhai,"ResearchonDynamic PathPlanningofWheeledRobot BasedonDeep ReinforcementLearningontheSlopeGround",*JournalofRobotics*,vol.2020,ArticleID7167243,10pages,2020.<https://doi.org/10.1155/2020/7167243>
<https://www.hindawi.com/journals/jr/2020/7167243/>
- Gutiérrez Ovando, Jorge MarioValladares Muñoz, William Fernando(2019)*Optimización de energía al controlar ventilación y aire acondicionado por medio de un algoritmo de aprendizaje por refuerzo profundo, implementado en el departamento de Ingeniería Mecánica, Universidad Nacional ChiaoTun, Hsinchu, Taiwán.*Licenciatura thesis, Universidad de San Carlos de Guatemala.
<http://www.repositorio.usac.edu.gt/14116/>
- Sierra-García, J. E. y Santos, M. (2021) «Redes neuronales y aprendizaje por refuerzo en el control de turbinas eólicas», *Revista Iberoamericana de Automática e Informática industrial*, 18(4), pp. 327–335. doi: 10.4995/riai.2021.16111.
<http://polipapers.upv.es/index.php/RIAI/article/view/16111/14274>



Algoritmos



Q-Learning

Es un algoritmo de aprendizaje por refuerzo que aprende a estimar el valor a largo plazo de cada acción en un estado determinado. Utiliza una función de valor Q que se actualiza iterativamente a medida que el agente interactúa con el entorno.

SARSA

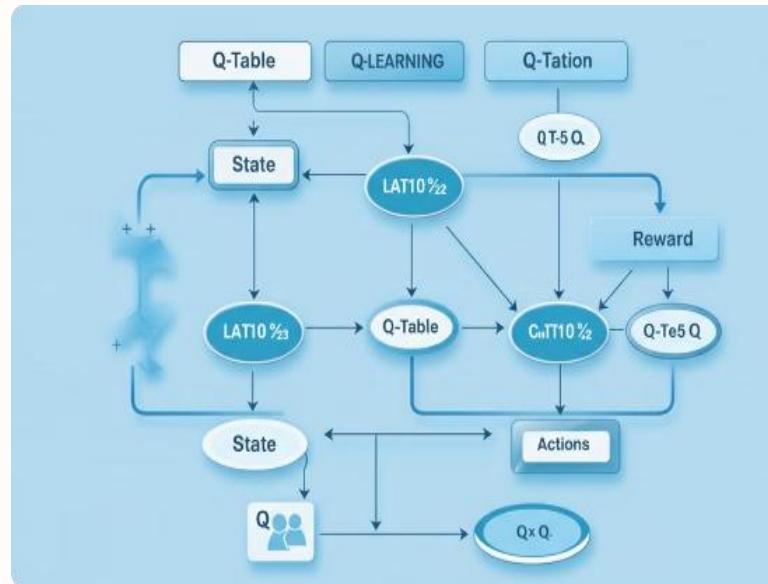
Es un algoritmo de aprendizaje por refuerzo que actualiza la función de valor en función de la acción actual y la siguiente acción que el agente planea tomar. Es un método de control de políticas on-policy.

Diferencia Temporal

Es una familia de algoritmos de aprendizaje por refuerzo que actualizan las estimaciones de valor a medida que el agente interactúa con el entorno, sin necesidad de esperar hasta el final del episodio para recibir la recompensa total.



Q-Learning



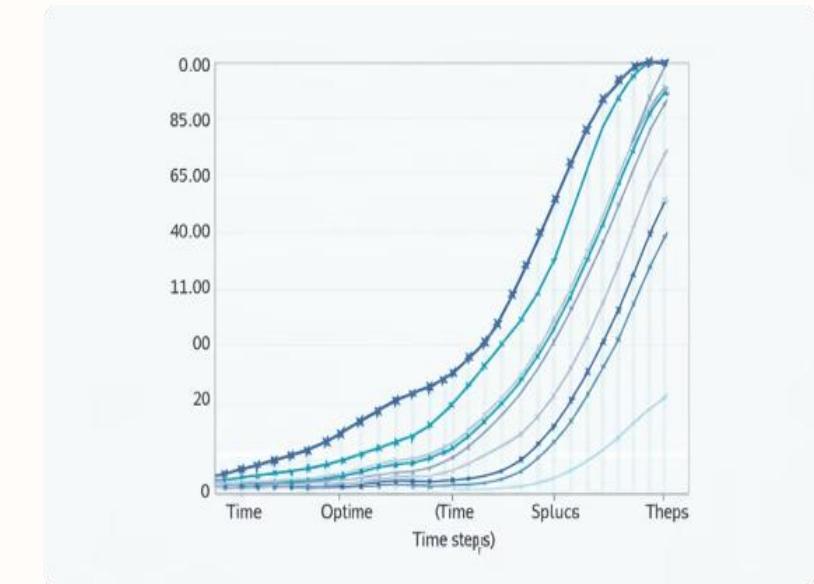
Funcionamiento del Q-Learning

El algoritmo Q-Learning es un método de aprendizaje por refuerzo que aprende a partir de las interacciones del agente con su entorno. En cada iteración, el agente observa el estado actual, selecciona una acción, recibe una recompensa y aprende a actualizar una tabla Q, que almacena las estimaciones de la recompensa futura esperada para cada par estado-acción.



Exploración vs Explotación

El algoritmo Q-Learning busca equilibrar la exploración del entorno para descubrir nuevas acciones rentables, y la explotación de las acciones que ya se sabe que dan buenos resultados. Esto se logra ajustando parámetros como la tasa de aprendizaje y el factor de descuento, lo que permite que el agente aprenda a tomar decisiones óptimas a largo plazo.

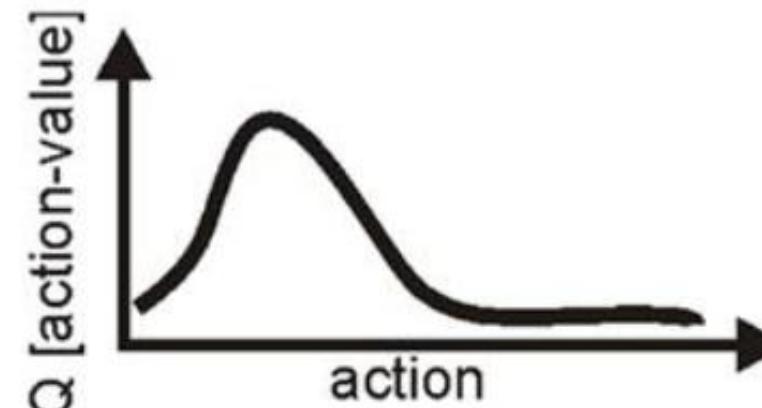


Convergencia a la Política Óptima

A medida que el algoritmo Q-Learning itera, los valores de la tabla Q convergen a la política óptima, lo que significa que el agente aprende a tomar las acciones que maximizan la recompensa acumulada a largo plazo. Esto lo convierte en una herramienta poderosa para resolver problemas de toma de decisiones complejos en una amplia gama de dominios.

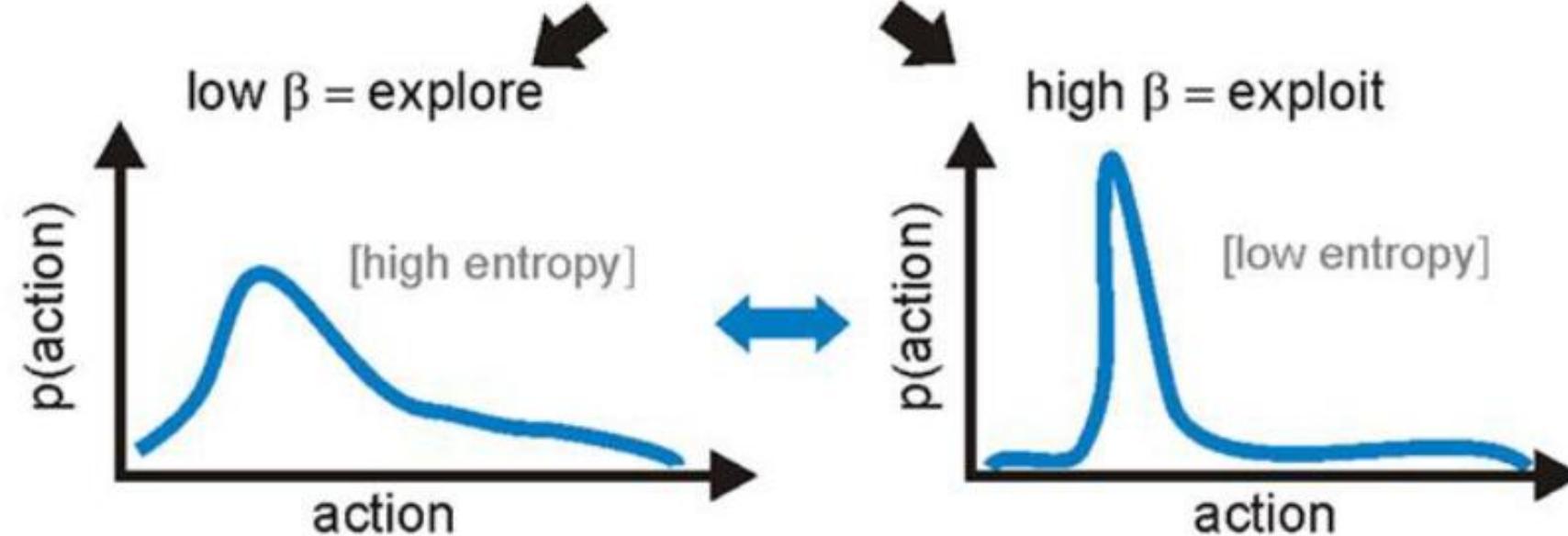


Q-Learning



Action-selection probability

$$p(a) = \exp(-Q(a)\beta) / \sum_a \exp(-Q(a)\beta)$$



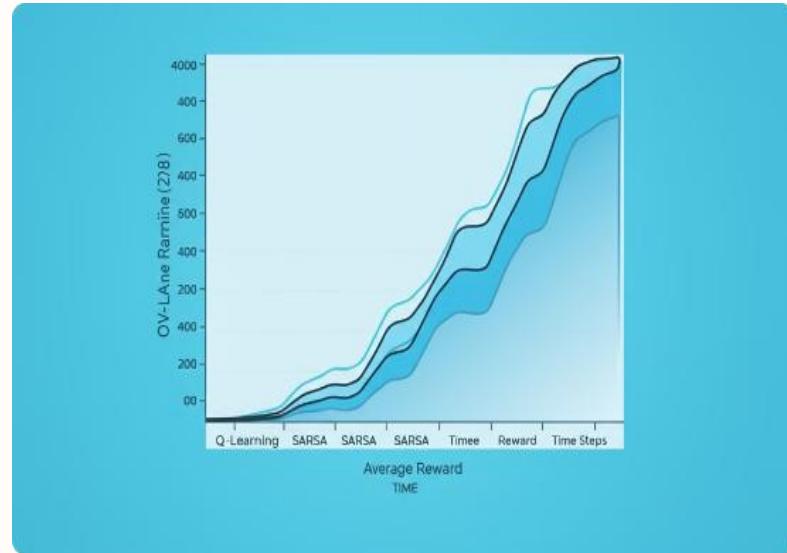


SARSA



Funcionamiento de SARSA

SARSA (State-Action-Reward-State-Action) es un algoritmo de aprendizaje por refuerzo que aprende una política óptima a través de la interacción directa con el entorno. A diferencia de Q-Learning, que aprende una función de valor independiente de la política, SARSA actualiza la función de valor según la acción real tomada en el siguiente estado.



Actualización de la Función de Valor

En cada paso, SARSA actualiza la función de valor de la acción tomada en el estado actual, utilizando la recompensa recibida y la función de valor de la siguiente acción y estado.

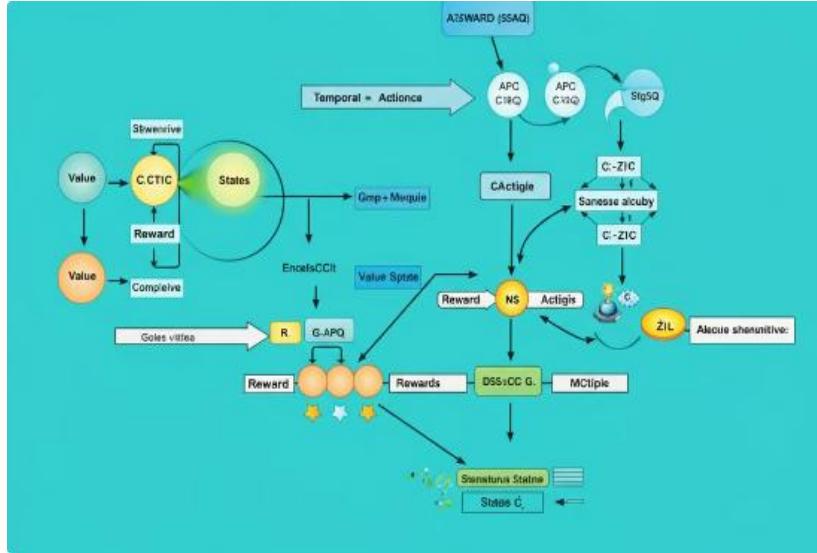


Exploración y Explotación

SARSA combina exploración y explotación al seleccionar acciones de acuerdo a una política como ϵ -greedy o softmax. Esto permite al agente aprender la función de valor de la política actual mientras explora nuevas posibilidades.



Diferencia Temporal



Definición

El algoritmo de Diferencia Temporal (TD) es un método de aprendizaje por refuerzo que combina ideas del aprendizaje supervisado y el aprendizaje por refuerzo. A diferencia de Q-Learning y SARSA, TD actualiza las estimaciones de valor de forma incremental a medida que se reciben nuevas observaciones.



Funcionamiento

TD aprende estimando el valor futuro esperado a partir de los valores de estado actuales y siguientes, en lugar de esperar hasta el final del episodio. Esto le permite aprender más rápidamente y adaptarse mejor a los cambios en el entorno.



Ventajas

TD es más eficiente que los métodos de Monte Carlo, ya que no requiere esperar hasta el final del episodio para actualizar las estimaciones. También es más flexible que Q-Learning, ya que no requiere un modelo de transición de estados.

Diferencia Temporal



Característica	Q-learning	SARSA	Aprendizaje por Diferencia Temporal (TD)
Tipo	Off-policy	On-policy	On-policy y Off-policy
Actualización de la función Q	Utiliza la acción con mayor valor Q en el siguiente estado (max Q)	Utiliza la acción realmente tomada en el siguiente estado	Utiliza una estimación basada en la diferencia entre el valor estimado actual y el valor obtenido en el siguiente paso
Convergencia	Tiende a converger a la política óptima	Tiende a converger a la política seguida	Puede converger a diferentes políticas dependiendo de la implementación
Exploración vs. Explotación	Balancea exploración y explotación utilizando ϵ -greedy o otras estrategias	Balancea exploración y explotación utilizando ϵ -greedy u otras estrategias	Balancea exploración y explotación
Ventajas	Puede aprender políticas óptimas más rápido	Puede ser más estable en entornos estocásticos	Flexible y adaptable a diferentes problemas
Desventajas	Puede ser más susceptible a sobreestimación de valores Q	Puede ser más lento en converger	Puede ser sensible a la elección de hiperparámetros



Python IA - Librerías

- OpenAlgym - <https://www.gymlibrary.ml/>
- KerasRL – Aprendizaje por refuerzo profundo para Keras.
 - ✓ Keras-RL implementa algunos de los algoritmos de aprendizaje por refuerzo más avanzados en Python y se integra perfectamente con la biblioteca de aprendizaje profundo Keras.
 - ✓ <https://github.com/keras-rl/keras-rl>
- Tensorforce (Tensorforce): Una biblioteca de TensorFlow para aprendizaje por refuerzo aplicado, incluyendo políticas basadas en valor y políticas basadas en políticas.
 - ✓ <https://github.com/tensorforce/tensorforce>





Python IA





TEMARIO

- 1 — Técnicas de detección de anomalías.
- 2 — Aprendizaje por refuerzo y control.
- 3 — Parametrización automática y optimización de algoritmos.



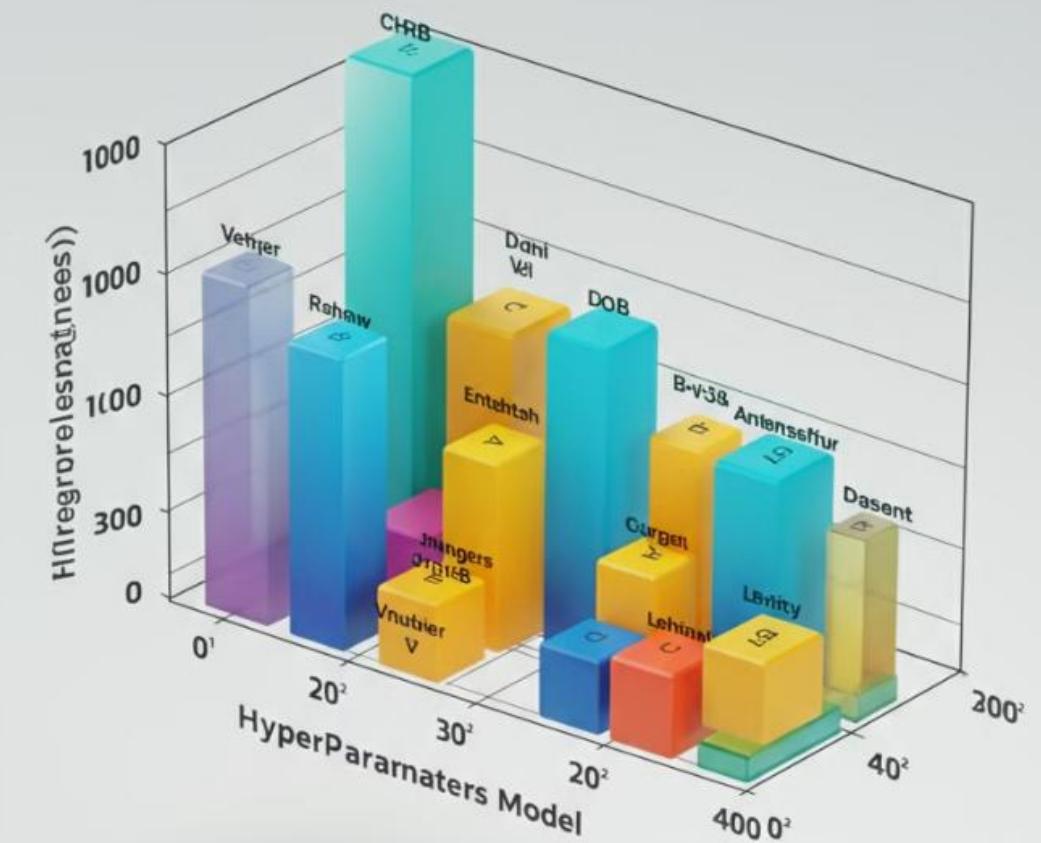
Parametrización automática y optimización de algoritmos



Hiperparámetros



Los hiperparámetros son variables de configuración que se establecen antes del proceso de entrenamiento de un modelo de machine learning. Estos parámetros afectan indirectamente el rendimiento del modelo y deben ser ajustados cuidadosamente para optimizar los resultados.





Hiperparámetros



Definición de Hiperparámetros

Los hiperparámetros son variables ajustables que se establecen antes de que un algoritmo de aprendizaje automático comience a funcionar. A diferencia de los parámetros del modelo, que se aprenden durante el entrenamiento, los hiperparámetros se deben establecer manualmente y afectan el rendimiento general del modelo.



Ajuste de Hiperparámetros

El ajuste de hiperparámetros es el proceso de encontrar el conjunto óptimo de hiperparámetros para un modelo de aprendizaje automático dado. Esto es crucial, ya que los hiperparámetros afectan significativamente la capacidad del modelo para generalizar y aprender de manera efectiva.



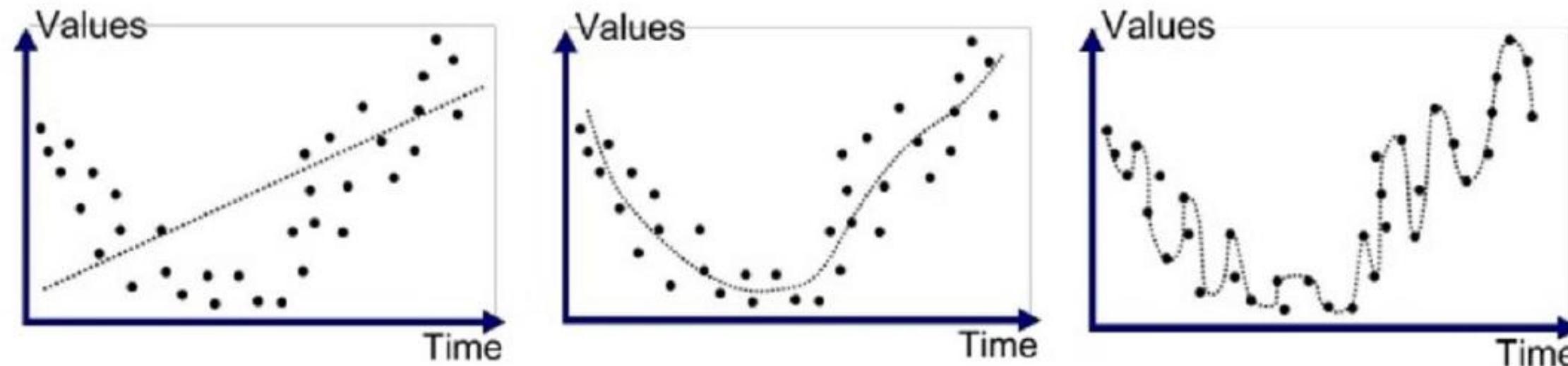
Importancia de los Hiperparámetros

Los hiperparámetros desempeñan un papel fundamental en el rendimiento de los modelos de aprendizaje automático. Determinar los hiperparámetros correctos puede mejorar significativamente la precisión, la velocidad de convergencia, la estabilidad y otros aspectos clave del modelo.



Hiperparámetros

- En la etapa de optimización de hiperparámetros, es crucial gestionar el sobreajuste.
- El sobreajuste ocurre cuando el algoritmo memoriza los datos existentes en lugar de identificar patrones aplicables a datos futuros.



Underfitted

Good Fit/Robust

Overfitted



Hiperparámetros

Importancia de los hiperparámetros en el aprendizaje automático

Ajuste Fino del Modelo

Los hiperparámetros permiten ajustar y optimizar el rendimiento de los modelos de aprendizaje automático. Permiten refinar la capacidad de aprendizaje y generalización de los algoritmos.

Prevención del Sobreajuste

Una selección cuidadosa de los hiperparámetros es crucial para evitar el sobreajuste de los modelos a los datos de entrenamiento. Esto mejora la capacidad de generalización.

Mejora del Rendimiento

El ajuste de hiperparámetros puede llevar a mejoras significativas en métricas clave como precisión, recall, F1-score y otras métricas relevantes para la tarea.





Hiperparámetros - SVM

Parámetros de Regularización

El parámetro C controla el equilibrio entre maximizar el margen y minimizar el error de clasificación, afectando la complejidad y el ajuste del modelo.

Función Kernel

La elección de la función kernel, como lineal, polinomial o RBF, determina cómo se mapean los datos a un espacio de mayor dimensión.

Parámetros del Kernel

Dependiendo de la función kernel, pueden existir parámetros adicionales como el grado del polinomio o la anchura de la función RBF.





Hiperparámetros - SVM

Definir la "parrilla" de hiperparámetros que queremos probar # Le damos a GridSearchCV una lista de opciones para cada hiperparámetro.

¡Probará todas las combinaciones posibles!

```
param_grid = {  
    'C': [0.1, 1, 10, 100], # El parámetro de regularización (el "guardia de seguridad")  
    'gamma': [1, 0.1, 0.01, 0.001], # El parámetro del kernel RBF (la "influencia")  
    'kernel': ['rbf'] # Nos enfocaremos en el kernel RBF que usa C y gamma  
}
```

Crear y configurar el objeto GridSearchCV

- estimator: El modelo que queremos optimizar (un SVC).

- param_grid: La parrilla de hiperparámetros que definimos.

- cv=5: Usará validación cruzada de 5 "folds" para evaluar cada combinación de forma robusta.

- verbose=2: Muestra información mientras se ejecuta.

```
grid_search = GridSearchCV(estimator=SVC(), param_grid=param_grid, cv=5, verbose=2)
```

Ejecutar la búsqueda

Este es el paso que puede tardar un tiempo, ya que entrena y evalúa muchos modelos.

En este caso: 4 (valores de C) * 4 (valores de gamma) * 1 (kernel) * 5 (folds) = 80 entrenamientos.

```
print("Iniciando la búsqueda de hiperparámetros...")  
grid_search.fit(X_train, y_train)
```

Hiperparámetros - Random Forest



Principales Hiperparámetros

En el algoritmo de Random Forest, los hiperparámetros clave incluyen el número de árboles de decisión, la profundidad máxima de los árboles, el número mínimo de muestras en una hoja y el número de características a considerar en cada división.

Ajuste de Hiperparámetros

El ajuste adecuado de estos hiperparámetros es crucial para lograr un rendimiento óptimo del modelo de Random Forest, equilibrando el sesgo y la varianza del clasificador.

Evaluación del Modelo

La selección de los mejores hiperparámetros se realiza mediante técnicas como validación cruzada, que permiten evaluar el rendimiento del modelo en datos independientes.



Hiperparámetros - Random Forest

Definir la "parrilla" de hiperparámetros a probar

Estos son los parámetros clave que menciona la diapositiva.

```
param_grid = {  
    'n_estimators': [100, 200], # Número de árboles en el bosque  
    'max_depth': [10, 20, None], # Profundidad máxima de cada árbol (None = sin límite)  
    'min_samples_leaf': [1, 2, 4], # Número mínimo de muestras en una hoja final  
    'max_features': ['sqrt', 'log2'] # Número de características a considerar en cada división  
}
```

Crear y configurar el objeto GridSearchCV

- estimator: El modelo a optimizar (RandomForestClassifier).

- param_grid: Nuestra parrilla de opciones.

- cv=3: Usaremos validación cruzada de 3 "folds". Es más rápido para el ejemplo.

- n_jobs=-1: Usa todos los procesadores disponibles para acelerar la búsqueda.

```
grid_search = GridSearchCV(  
    estimator=RandomForestClassifier(random_state=42),  
    param_grid=param_grid,  
    cv=3,  
    n_jobs=-1,  
    verbose=2)
```

Ejecutar la búsqueda de los mejores hiperparámetros

```
print("Iniciando la búsqueda para Random Forest...")  
grid_search.fit(X_train, y_train)
```

Hiperparámetros - Refuerzo



Exploración vs. Explotación

Los hiperparámetros clave en Reinforcement Learning (RL) determinan el equilibrio entre explorar nuevas acciones y explotar las que se conocen como más efectivas. Esto es crucial para que el agente RL aprenda de manera eficiente.

Función de Recompensa

La función de recompensa define cómo se recompensa al agente RL por sus acciones. Ajustar los pesos y umbrales de esta función es fundamental para guiar al agente hacia los comportamientos deseados.

Arquitectura de la Red

En RL basado en redes neuronales, los hiperparámetros como el número de capas, neuronas y conexiones determinan la capacidad de aprendizaje y representación del modelo.

Optimización del Aprendizaje

Parámetros como la tasa de aprendizaje, el factor de descuento y el método de actualización del valor Q afectan la velocidad y estabilidad del proceso de aprendizaje.

Hiperparámetros - Redes Neuronales



Arquitectura de la Red

El número de capas ocultas, neuronas por capa y funciones de activación son hiperparámetros clave que definen la complejidad del modelo.

Optimización del Entrenamiento

Parámetros como la tasa de aprendizaje, momentum y regularización afectan la convergencia y la generalización del modelo.

Prevención del Sobreajuste

El uso de técnicas como Dropout y Batch Normalization ayudan a evitar el sobreajuste de los datos de entrenamiento.



Optimización de algoritmos – Búsqueda de Hiperparámetros

GridSearch



1

Definición

GridSearch es una técnica exhaustiva de búsqueda de hiperparámetros que evalúa todas las combinaciones posibles dentro de un espacio de parámetros definido.

2

Proceso

Se define un rango de valores para cada hiperparámetro y se prueba cada combinación posible, seleccionando el mejor resultado.

3

Ventajas

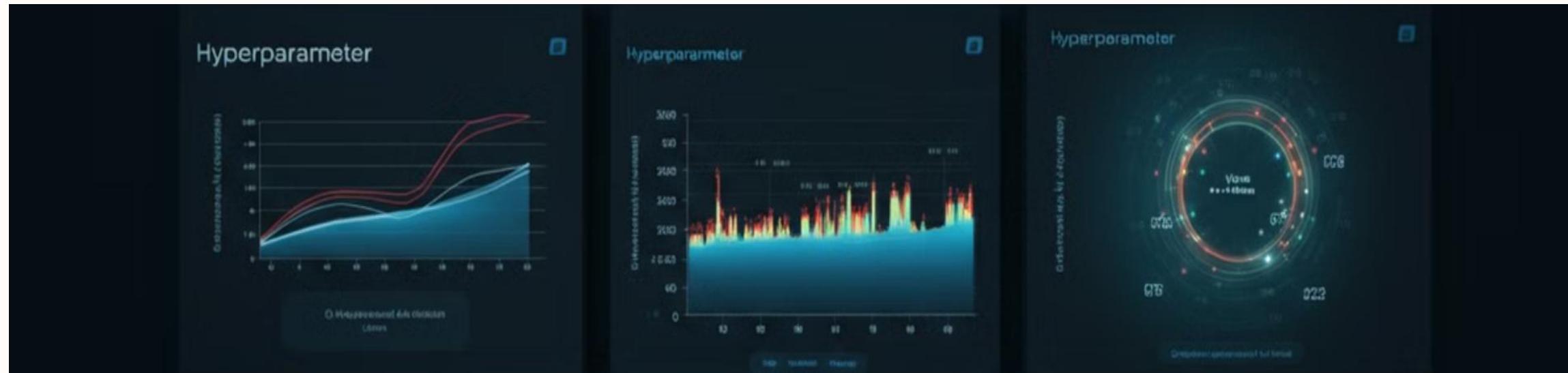
Garantiza encontrar la mejor combinación de hiperparámetros, pero puede ser computacionalmente costoso.

GridSearch es una técnica de búsqueda de hiperparámetros ampliamente utilizada en el campo del Aprendizaje Automático. Permite explorar de manera exhaustiva un espacio de parámetros definido, garantizando encontrar la mejor combinación para un modelo dado.



Optimización de algoritmos – Búsqueda de Hiperparámetros

GridSearch



Ventajas de GridSearch

Exhaustividad

GridSearch examina todas las posibles combinaciones de hiperparámetros, asegurando que no se deje ninguna opción sin explorar.

Interpretabilidad

Los resultados de GridSearch son fáciles de interpretar, ya que se pueden visualizar y comparar los rendimientos de cada configuración de hiperparámetros.

Paralelización

GridSearch se puede paralelizar fácilmente, lo que permite acelerar el proceso de búsqueda en entornos con múltiples núcleos.

Optimización de algoritmos – Búsqueda de Hiperparámetros

GridSearch



Desventajas de GridSearch

Complejidad Computacional

GridSearch debe evaluar todas las posibles combinaciones de hiperparámetros, lo que puede ser computacionalmente costoso, especialmente cuando hay un gran espacio de búsqueda.

Ineficiencia para Espacios Grandes

GridSearch puede ser ineficiente cuando se tienen espacios de búsqueda muy grandes, ya que evalúa cada combinación de forma exhaustiva, lo que puede tardar mucho tiempo.

Escalado Limitado

A medida que aumenta el número de hiperparámetros y posibles valores, el espacio de búsqueda crece exponencialmente, lo que dificulta el escalado de GridSearch.

Falta de Adaptabilidad

GridSearch no se adapta dinámicamente a los mejores valores de hiperparámetros a medida que avanza la búsqueda, lo que puede limitar su eficiencia.



Optimización de algoritmos – Búsqueda de Hiperparámetros - RandomSearch

Flexibilidad

1

A diferencia de GridSearch, RandomSearch permite una exploración más flexible de los espacios de hiperparámetros. En lugar de evaluar todas las combinaciones posibles, RandomSearch selecciona aleatoriamente valores dentro de los rangos definidos.

Eficiencia

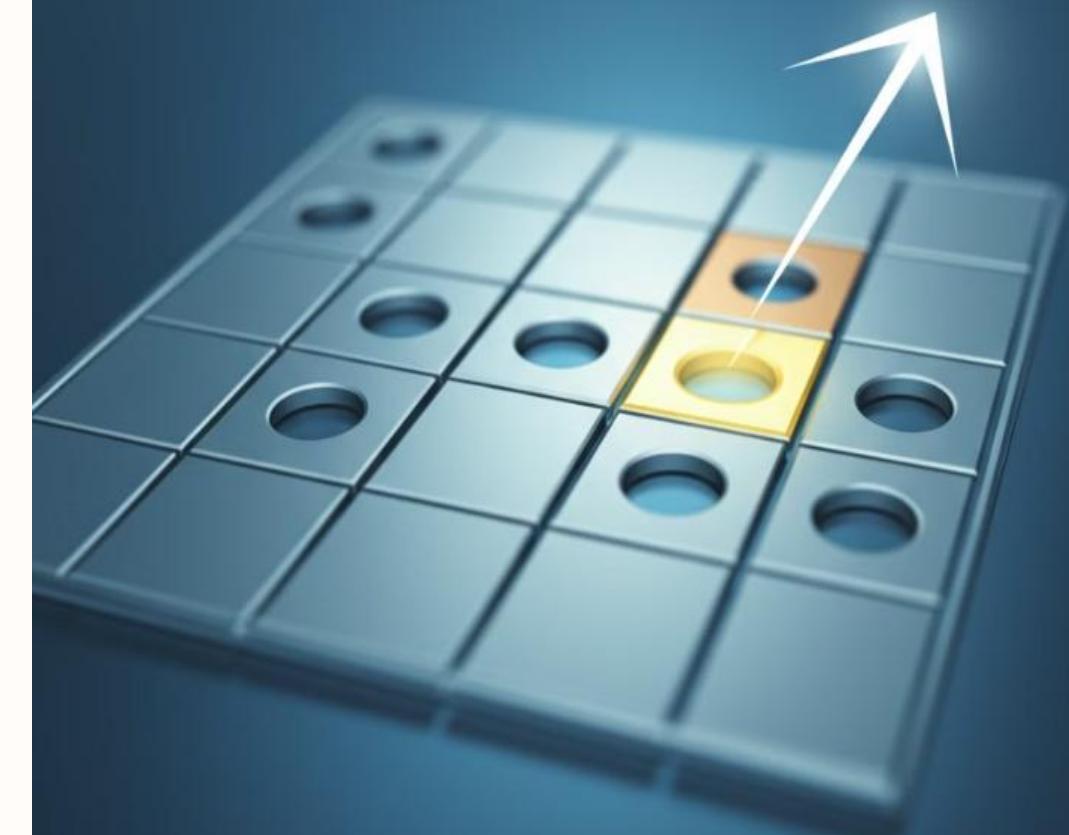
2

RandomSearch tiende a ser más eficiente cuando se tienen espacios de búsqueda grandes, ya que evita evaluar combinaciones innecesarias. Esto se traduce en un menor tiempo de cómputo y mayor rapidez en la optimización de hiperparámetros.

Exploración

3

Al ser aleatorio, RandomSearch permite una exploración más diversa del espacio de hiperparámetros. Esto puede llevar a la identificación de soluciones óptimas que podrían ser pasadas por alto en una búsqueda más sistemática.



Optimización de algoritmos – Búsqueda de Hiperparámetros

RandomSearch



Ventajas de RandomSearch



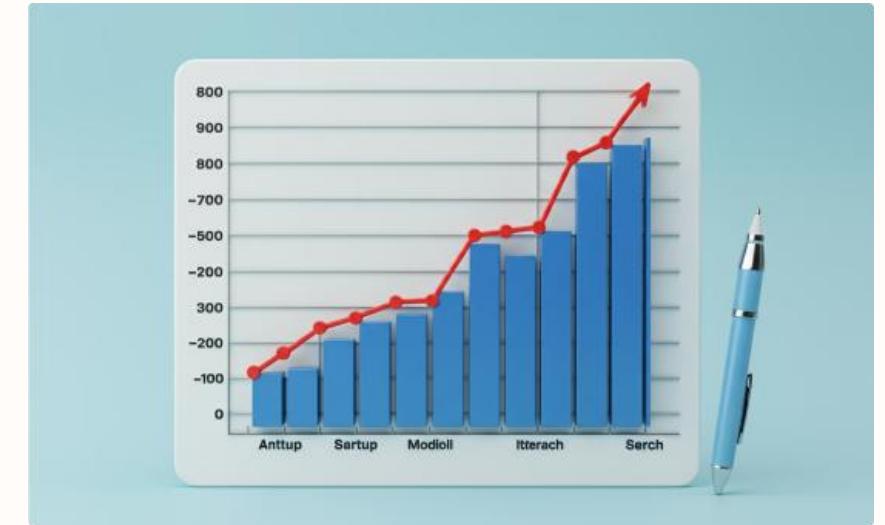
Exploración Eficiente

RandomSearch permite explorar el espacio de hiperparámetros de una manera más eficiente que GridSearch, al evitar la búsqueda exhaustiva de todas las combinaciones posibles. Esto resulta especialmente ventajoso cuando se tiene un gran número de hiperparámetros a optimizar.



Mayor Flexibilidad

A diferencia de GridSearch, RandomSearch no requiere definir un conjunto fijo de valores a probar para cada hiperparámetro. Esto brinda mayor flexibilidad, permitiendo explorar un espacio de hiperparámetros más amplio y complejo.



Mejor Rendimiento

En muchos casos, RandomSearch puede conducir a la identificación de mejores combinaciones de hiperparámetros que GridSearch, lo que se traduce en un mejor rendimiento del modelo de machine learning.



Optimización de algoritmos – Búsqueda de Hiperparámetros - RandomSearch

Desventajas de RandomSearch

Falta de Eficiencia

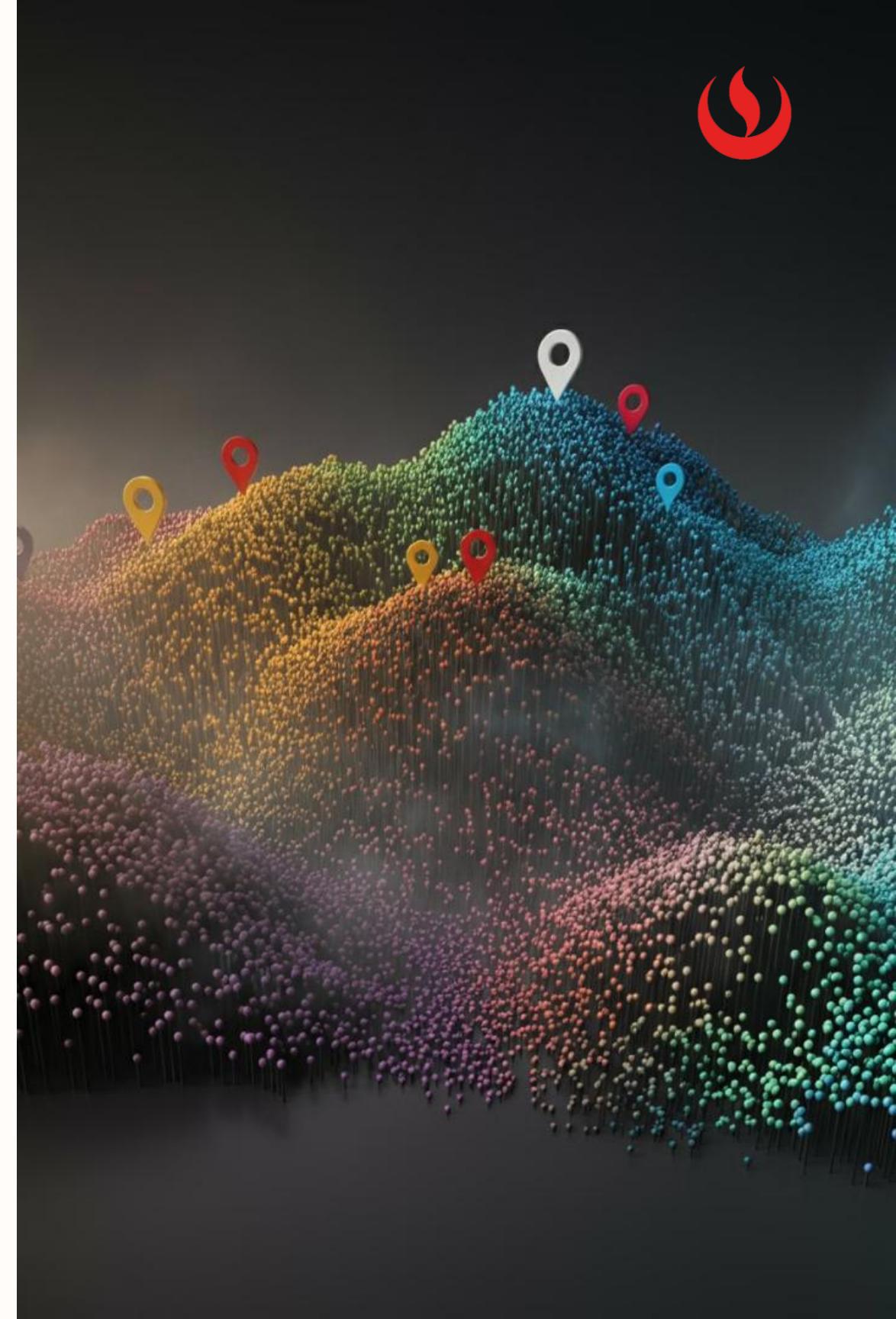
RandomSearch puede ser menos eficiente que GridSearch, especialmente cuando se tienen pocas combinaciones de hiperparámetros a evaluar. Al elegir aleatoriamente las combinaciones, es posible que algunas regiones prometedoras del espacio de búsqueda se pasen por alto.

Necesidad de Más Iteraciones

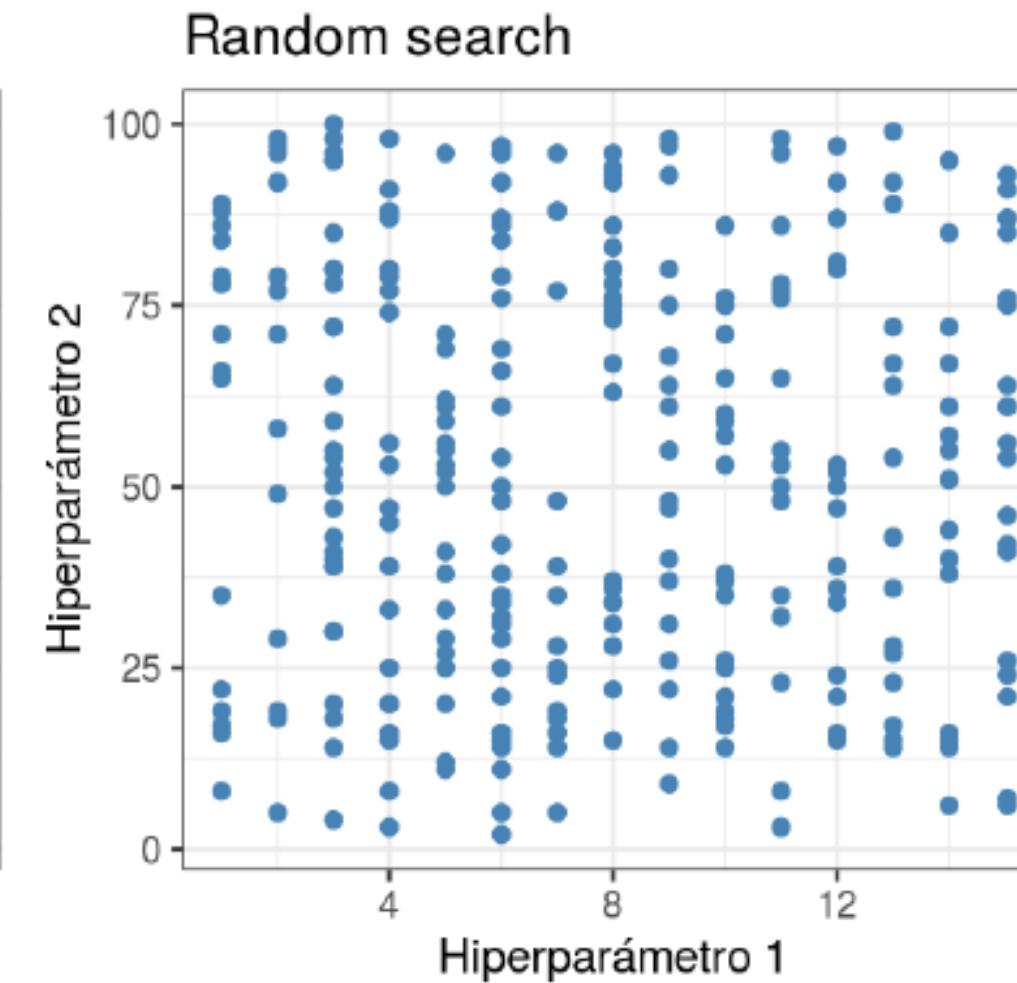
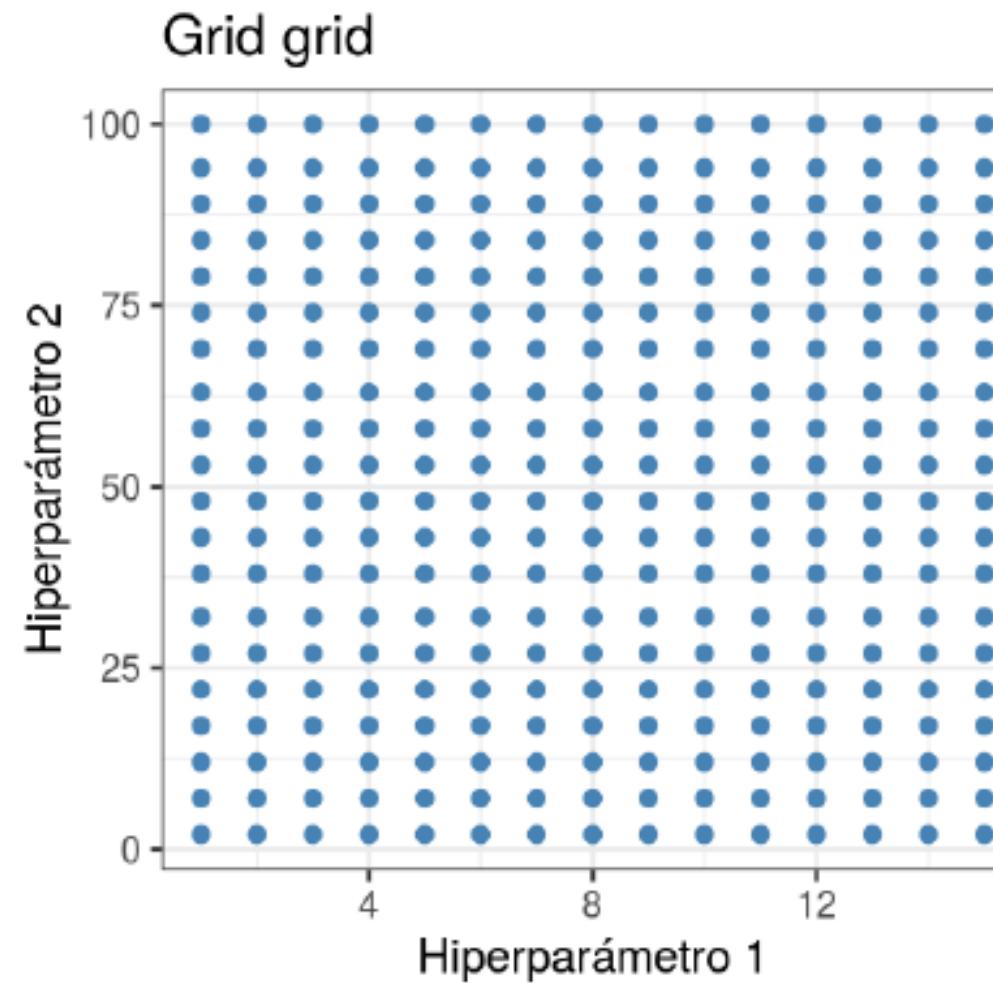
Para lograr resultados confiables, RandomSearch generalmente requiere más iteraciones que GridSearch, lo cual puede implicar un mayor costo computacional y de tiempo.

Selección de Parámetros

A diferencia de GridSearch, RandomSearch no garantiza que se evalúen todas las posibles combinaciones de hiperparámetros, lo cual puede llevar a una selección subóptima de los mismos.



Optimización de algoritmos – GridSearch vs RandomSearch





Python IA





Actividad Grupal Técnicas de agrupamiento



CONCLUSIONES



01

La técnica de detección de anomalías identifica patrones atípicos en datos usando enfoques supervisados, no supervisados y probabilísticos, esenciales en ciberseguridad y mantenimiento.

02

El aprendizaje por refuerzo y control permite a agentes aprender a través de recompensas, aplicándose en robótica, juegos y control adaptativo en entornos complejos.

03

La Parametrización automática permite seleccionar hiperparámetros con diferentes técnicas, acelerando la construcción de modelos precisos y eficientes.

04

La optimización de algoritmos mejora la eficiencia y precisión de los modelos con enfoques como gradiente descendente logrando soluciones escalables.



Bibliografía

- Nassif, A. B., Talib, M. A., Nasir, Q., & Dakalbab , F. M. (2021). Machine learning for anomaly detection: A systematic review. Ieee Access 9 , 78658 78700.
- Dogo , E. M., Nwulu , N. I., Twala, B., & Aigbavboa , C. (2019). A survey of machine learning methods applied to anomaly detection on drinking water quality data. Urban Water Journal 16 (3), 235 248
- Peng Wang, Xiaoqiang Li, Chunxiao Song , Shipeng Zhai Research on Dynamic Path Planning of Wheeled Robot Based on Deep Reinforcement Learning on the Slope Ground Journal of Robotics vol. 2020, Article ID 7167243, 10 pages 2020. <https://doi.org/10.1155/2020/7167243>
- GridSearch o RandomSearch : <https://towardsdatascience.com/gridsearchcv-or-randomsearchcv-5aa4acf5348c>