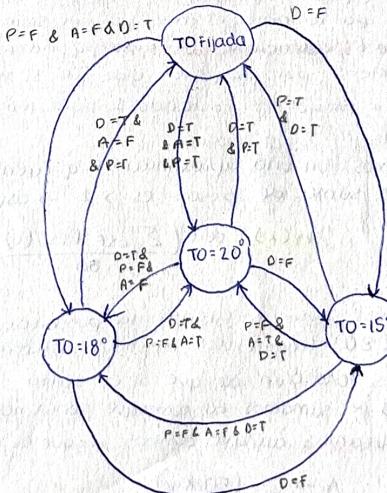
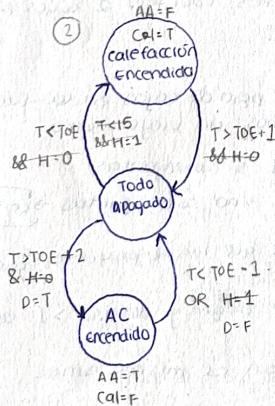


- EJERCICIOS / HOJA 1**
- ① Formalizar sistema usando REAS
- medida de rendimiento: temperatura media de la habitación → y tra la temp min max
  - en torno a temperatura objetivo ("si... baja la temperatura objetivo") → CAMBIA TEMP
  - temperatura obj. establecida (TOE) ("si la temp es más baja que la establecida... → OBSERVA EST.)
  - ventana ("si hay una ventana abierta" → OBSERVA VENTANA)
  - calefacción (lo enciende y lo apaga → lo CAMBIA)
  - aire acondicionado (lo enciende y apaga → lo CAMBIA)
  - humanos y animales ("hay..." → OBSERVA)
  - temperatura ("si la temp es más baja que la establecida") → 2 diff variables por temp. → OBSERVA
  - yo pondría hora (dia/noche) lo pero el profe no lo pone. → OBSERVA
  - alarma no pq no está en el entorno sino en el propio sistema del agente.
- actuadores:
- calefacción → cal: bool → to las variables del entorno que ponía CAMBIA (en vez de OBSERVA)
  - aire acondicionado → AA: bool → ahora la alarma si es relevante. Es parte del sistema y el agente la enciende y apaga.
  - alarma → Alam: bool
  - modificar temp objetivo → setTo (float)
- sensores:
- temperatura → T: real
  - ventana → V: bool
  - humanos → H: bool
  - animales → A: bool
  - hora → H: Dia / Noche (0,1)
- b) Que tipo de agente es? → Agente basado en modelo Pq guarda info y actualiza estados. Usar máquina de estados jerárquica para modelarlo. Tenemos 3 claros sub-sistemas que se ejecutan en paralelo:
- subsistema que detecta si hay o no habitantes en la hab
  - subsistema que controla la temperatura
  - subsistema que controla la alarma

①



②



③

- ② coche autónomo : tiene sensores para detectar objetos, se mueve en línea recta  
 - detecta desniveles tb  
 - marcha atrás y gira ruedas izq y dcha.  
 - enciende faros blancos cuando oscuro o rojo peligro

### comportamiento:

- observa obstáculo de frente → cambia dirección Izq/dcha randomly
- si mientras gira observa otro obstáculo → sigue girando
- si da vuelta completa → cambia y enciende luz roja bloqueo y cambia a parado y observa obstáculo (→ no se sitúa a la pac.)
- } observa
- si al girar/detecta golpe → cambia dir. de giro
- ↳ si no observa otro golpe → cambia a marcha atrás

- si dando marcha atrás } observa /detecta golpe → cambia a marcha adelante
- si observa que se cae por escaleras → cambia y enciende luz roja → cambia a parado.
- si observa escalera de frente, cambia de sentido → marcha atrás + giro aleatorio + marcha adelante
- si observa algún problema → cambia según lo que encuentre.
- observa oscuridad → cambia faros ON & OFF luz blanca

### 1. REAS

medida rendimiento: moverse en línea recta y detectar obstáculos

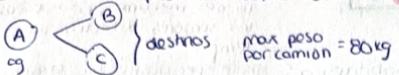
- entorno:
- obstáculo de frente
  - dirección (izq o dcha) giro
  - luz roja de bloqueo
  - parar coche
  - golpe
  - marcha atrás
  - marcha adelante
  - escaleras de frente
  - caída de escaleras
  - cambiar sentido (no corremos)
  - oscurecimiento
  - luz blanca
- actuadores:
- dirección giro → G:0, G:1:bool
  - luz roja → LR:bool
  - en marcha → M:bool
  - marcha atrás → Atras:bool
  - luz blanca → LB:bool
- sensores:
- obstáculo → Obst:bool
  - golpe → Golpe = bool
  - escaleras → ESC = bool
  - caída → Ctsc = bool
  - oscuridad → OSC = bool

### 2. Tipo de agente.

Agente basado en objetivos por que las metas cambian según lo que se vaya encontrando el coche. El agente puede empetar un giro pero no sabe si va a terminar en golpe, escalera, obstáculo, en nada, ... etc.

Las decisiones serán acertadas según la meta actual.

⑧ Formaliza el problema como uno de **BUSQUEDA** → necesitamos definir = 1) como modelarlos prob.  
 2) estado ini  
 3) estado final  
 4) acciones  
 5) coste del camino



# busqueda

## 1. Modelamos problema

→ asumimos un vector de cajas y posición del coche  
 las cajas guardan =

- posición (puede cambiar) entre A,B,C, o F
- destino (no cambia)
- peso (no cambia)
- id

La furgoneta guarda

- posición → A,B,C (cambia)

## 2. Estado inicial → miramos config inicial

cajas	id	1	2	3	4	5	6	7	8
pos	A	A	A	A	A	A	B	C	

## 3. Estado final

cajas	id	1	2	3	4	5	6	7	8
pos	C	B	C	C	B	C	A	A	

furgujo : A (lo dice el problema que es origen)

furgujo = \* → significa que no sabemos / da igual

## 4. Acciones → como methods en POC

- cargar(caja) → cambia posición para esa caja a F.
- descargar(caja) → cambia pos a posición de furgoneta. F = A, B, or C
- moverA: cambia pos furgujo a A. Solo si furgujo pos = B ahora.
- moverC: cambia pos furgujo to C. Solo si furgujo pos = B ahora.
- moverB: cambia pos furgujo to B. If furgujo = A or furgujo = C pos.

## 5. Función de coste → lo principal q queremos q haga → mover furgujo de pos a pos → este es el coste.

- moverA, moverB, moverC tienen coste = 1
- cargar, descargar coste 0.

④ calcula función heurística el sea admisible. Por ej de valor q teníaria la función h si partíri de no do cajas mal colocadas significa q origen ≠ destino (nec a moverlas). Y sus descendientes  
 la cosa q podemos relajar: q A, B, y C estén todas conectadas y q por lo tanto, el coste mínimo siempre sea 1.

OJO! NO es admisible del todo pq no tenemos en cuenta q no siempre tiene q ser coste = 1 por separado ya q si metemos (por ej) 2 cajas q pesen < 80 kg, serán 2 paquetes costando 1 para moverse!

mejoramos con una aproximación q cuente el peso de cajas en la furgujo.  
 Si esta suma de pesos es > 1 → asumimos un viaje extra.

$$h(x) = \text{ceil} \left( \frac{\sum_{c \in F} \text{Peso}(c)}{80} \right) < 1 \text{ si caben juntas.}$$

> 1 → no caben juntas → coste + 1

Pero aquí no estamos teniendo en cuenta donde van!!

Si queremos tener en cuenta la posibilidad de q los 2 paquetes (cuales pesan < 80) tengan dos destinos diferentes....

ponemos condición de q si el salto entre origen y destino > 1 de alguna caja, sumamos al coste q teníamos 1.

generalizamos, aunque ambos paquetes vayan al mismo sitio.

$$\begin{aligned} A \rightarrow C & (50 \text{ kg}) \\ A \rightarrow C & (10 \text{ kg}) \end{aligned} \quad \left\{ \begin{array}{l} \text{interpretamos} \\ \text{como coste = 2.} \end{array} \right.$$

$$\text{if and only if } \Leftrightarrow \exists c \in F \mid \text{dist}(\text{pos}(c), \text{dest}(c)) > 1$$

## ⑤ CUBO RUBIK

## EJERCICIOS HOJA 1 cont.

busqueda

Nótese: problema como uno de búsqueda.

1) summos vector de cuadraditos que guardan lado origen y lado destino.

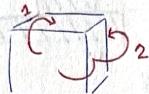
2) Estado inicial = cualquier lado

3) Estado final = cubo ordenado con cada lado de un color

4) Acciones:

- mover fila izq      • mover col arriba
- mover fila dcha      • mover col abajo

5) coste por cualquier movimiento (+1).



Hay que tener en cuenta que multiplicaremos el coste por los movimientos que necesito para llegar al ~~lado~~ lado destino.

$$\begin{array}{|c|} \hline \text{max movimientos} = 2 \\ \hline \end{array}$$

$$\begin{array}{|c|} \hline \text{min mov} = 1 \\ \hline \end{array}$$

## ⑥ función heurística para ↑. (cubo de 2x2)

Primer solución mágica que se te venga

→ levantar pegatinas y pegar en el lado donde van.

→ Aquí el coste = cada pegatina mal pegada \* 1 (coste de moverla)

Otra heurística

→ si sabemos a la distancia que está cada lado,

miramos si en un lado hay alguna pegatina con distancia destino = 2

si la tenemos = coste + 2 automatically.

## ⑦ Tenemos 5 anuncios: A1, A2, A3, A4, A5 y 3 huecos con corresp. Price benefits

Pi	Si	Si
----	----	----

No se pueden repetir anuncios en un individuo.

→ queremos maximizar ingresos para resolver mediante un alg. genético

genes

Pi > Si

## ① Función fitness → elegimos sec de anuncios que nos da más ingresos

id	P	S
A1	100	10
A2	200	10
A3	300	10
A4	400	50
A5	500	100

↳ [A5 | A4 | A3] y [A4 | A5 | A2]

↓  
[A5 | A5 | A2]

lo malo → no tenemos ninguna regla que impida que se repitan anuncios al hacer crossover

" "

→ una solución para esto sería penalizar los indiv. invalidos con un fitness negativo.

Otra opción es no permitir que se produzcan individuos invalidos en primer lugar. Para esto hay que plantear codificar el problema de otra forma.

Metemos los anuncios en <sup>un array</sup> posiciones que a medida de que seleccionemos anuncios los vamos sacando del array y reorganizamos los que quedan, quitando el hueco

A1	A2	A3	A4	A5
----	----	----	----	----

De esta manera cuando ponemos 3 posiciones

en un vector [1 | 1 | 1] significa se van a referir al mismo anuncio.

0	1	2	3	4
---	---	---	---	---

1 1 2

0	1	2
---	---	---

0 0 1

G5! Ahora cuando cortemos dos vectores, lo que hacemos es cruzar POSICIONES (no VALORES!!)

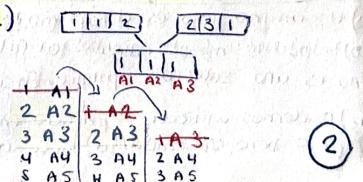
A3	A4	A5
----	----	----

0	1	2
---	---	---

Luego necesitaremos traducir y pillar los valores de los anuncios en orden !!

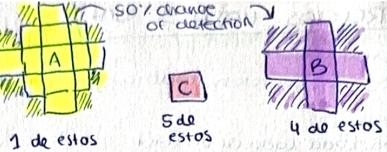
Desventaja: esto será costoso y no funcionaría si cambia el orden

Ventaja: no se producen individuos invalidos

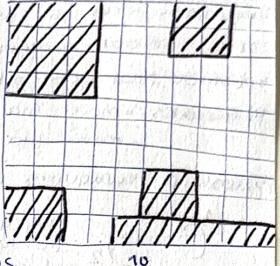


(2)

⑧ 10 sensores



y una habitación  $10 \times 10$  con algunos cuadros donde no deberían estar los sensores.



Definir problema para ser optimizado por alg. genéticos para cubrir mayor parte posible de la habitación casillas q pertenecen a habitación  $\rightarrow$  true, las que no  $\rightarrow$  false.

La función fitness será calcular el número de casillas que están protegidas por los sensores. Para ello cogemos la matriz  $10 \times 10$  y vamos colocando sensores en las casillas. Ponemos un 1 si está totalmente cubierta y un 0.5 si lo está parcialmente. La suma de estas casillas será el fitness. Quedan que tener ciertas consideraciones con los individuos inválidos. En este caso esos son los sensores colocados en los muros, como siempre tenemos varias formas de controlar estos indi. inválidos:

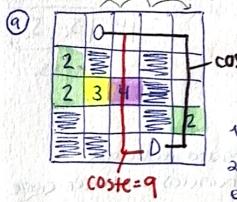
- Podemos penalizar la casilla con valor false q que su fitness sea 0 (no válida) y así no contribuye al fitness total
- o bien que directamente no se pueda asignar un sensor a una casilla con valor false (no válida)

Hay que tener en cuenta tb q una casilla puede estar al 50% con sensor A y retenerse con otro 50% del sensor B.  $50\% + 50\% = 100\%$

TD tenemos que ver si tenemos un vector de 20

0 10 | 93 | 41 | 88 | ... donde cada num rep un par

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
5	3	4	1	1	8	8	1	1	1	1	1	1	1	1	1	1	1	1	1
X	Y	X	Y	X	Y	X	Y	X	Y	X	Y	X	Y	X	Y	X	Y	X	Y



Dibujamos el proceso completo q haría el alg. A\*.

la función heurística es la distancia Manhattan.

Nota: El agente solo puede moverse en horizontal y vertical y no en diagonal.

1) camino más corto = hay q de tamaño 9.

2) si usamos dist. euclídea lo resolveríamos en menos pasos?

en este caso la euclídea es menos informada q Manhattan pq el agente no se puede mover en diagonal y no tendría en cuenta las casillas con más coste.

3) qué pasa si el agente puede moverse en diagonal con coste 1? Sería admisible la dist. de Manhattan? No porque podríamos resolver el problema en menos pasos q la heurística, por lo tanto, esta heurística se volvería inadmisible ya q siempre debe tener un coste menor q el real.

⑩ Ahora haremos usando alg. voraz  $\rightarrow$  que no tiene en cuenta coste acumulado!!

El alg. expande el nodo más prometedor = el q tenga menor heurística = el que tarda menos en llegar al destino. En este caso distancia Manhattan.

Si partimos del origen  $(1,0)$  a la izq. tenemos  $(0,0)$  y a la dcha  $(2,0)$ . De estos dos elegimos nodo  $(2,0)$  pq su heurística es menor a la de  $(0,0)$ .

Si expandimos el nodo  $(2,0)$  tenemos un camino a la dcha  $(3,0)$  y otro hacia abajo  $(2,1)$ . Una vez más elegimos el camino rojo ya q tiene menor distancia al destino (menor heurística).

a) El camino rojo es el más prometedor pero al no tener en cuenta el coste del camino al llegar a un nodo (solo miramos al futuro) y replanificarse si nos topamos con un camino sin salida), no es una solución óptima. El clás. A\* si tiene esto en cuenta y nos daría un camino más óptimo.

b) Podemos utilizar "primero en profundidad" ya q actuaria similar al alg. voraz pero tiene, sin embargo, una complejidad inferior. Acabarria más rápido.

① Tenemos dataset con datos almacenados sobre lluvias desde hace 10 años y queremos hacer predicciones en función del año (int), la humedad (real), cantidad de nubosidad (baja, media, alta), la temperatura (real) y la presión atmosférica (real)

Resultados de accuracy:

- KNN =  $K=5$ , 88% distancia euclídea

- Decision tree: 90%, profundidad del árbol 5

- Random Forest: 98%

{ Percepción multicapa: 10 neuronas ocultas en 1era capa y 5 en 2da. factor de aprendizaje de 0 a 1 valores normalizados 0-1: 90%

→ Estos valores han sido bajos

a) como podemos mejorarlo?

Podemos reentrenar la red con otro factor de aprendizaje más bajo, ya que suelta funcionar mejor con valores pequeños.

Tb podemos modificar la arquitectura de la red como el num de capas y neuronas por capas o la selección de atributos. Hay que recordar que las entradas mejoran si están normalizadas a valores (0/1). Ir probando y ver como evolucionan.

b) que modelo elegirías?

Lo primero es ponerte un poco en perspectiva en respecto a la seriedad de estas predicciones. Como estamos hablando de lluvia, que no es un tema super importante podemos elegir el árbol de decisión <sup>que</sup> da buenos resultados (9 de cada 10 veces) y ademas es un algoritmo de caja blanca. Si queremos mas precisión (si la info fuese para agencias de viaje u hoteles por ejemplo) podemos elegir el Random Forest, teniendo en cuenta que es mas difícil de interpretar.

c) En KNN el valor categórico ha sido codificado como 0, 1, 2 y el resto de atributos para calcular la diferencia se realizan restando sus valores sin ninguna transformación.

como podemos mejorar KNN para conseguir mejores resultados?

Podemos aplicar onehot encoding para codificar los valores categóricos que se quedaron sin ninguna transformación. Tb podemos normalizar los datos haciendo que estén entre 0 y 1. Este haría que la distancia euclídea represente mejor al individuo.

Podemos normalizar los valores aplicando una resta de la norma y dividiendo por la desviación típica también.

② Utilizamos lib train-test-split para dividir conjuntos entrenamiento y test, con valores =  
train-test-split (data, y, train\_size = 0.8, random\_state = 1)

Sin embargo haciendo pruebas vemos que en función del random\_state o tam de datos de ent. el resultado varía hasta un 5% en accuracy.

como abordamos problema?

utilizamos cross validation para minimizar desviación que puede producir una mala partición del dataset entre entrenamiento y validación.

\* MÁS INFO EN PÁG 1.5 DE TEMA 5 !!

machine  
learning

③ Queremos detectar a partir de imágenes X-ray si existe un tumor o no.

Tenemos dataset con 100.000 imágenes de 1024x1024 en escala de grises.

## ① PASOS PARA PREPARAR DATOS:

- reducir escala de grises de 1024x1024 a 64x64 por ejemplo → **así tenemos menos atributos** con un auto encoder o pca,

Ambos intentan buscar un problema de menor dimensión en el dominio de mayor dimensión → conserva solo parte de la info

PCA = coge pixeles y se queda <sup>con</sup> una transformación lineal de esos pixeles.

Auto encoder = encuentra niveles patrones, es decir, puede extraer características de más alto nivel.

pero guarda rep. interna de la imagen (pca no), no una copia.

mejor este!

## ② MEJOR MODELO PARA PROCESAR IMÁGENES? tema 4

→ Redes neuronales convolucionales y los autoencoders

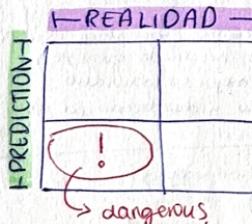
→ A priori lo el perceptrón multicapa se comporta mejor en este tipo de probs pero habrá que probarlo.

## ③ CON 2000 DATOS TENEMOS:

		tumor	no tumor
tumor	tumor	800	200
	no-tumor	100	900

		tumor	no tumor
tumor	tumor	900	100
	no tumor	200	800



↳ Esta es la mejor. Clasifica menos no-tumores cuando en realidad sí hay tumor.

implica que me muero si no me tratan!!!

que mío es peor a que me traten sin tener tumor...

## ④ PROBLEMA CLASIFICACIÓN 5 CLASES. QUEREMOS DISEÑAR RED NEURONAL. DESCRIBIR CAPA SALIDA, NEURONAS DE SALIDA, Y TIPO.

→ 5 neuronas, una por cada clase. Tendríamos que poner una capa (función de activación) softmax ya que siempre te da una probabilidad, que es lo que buscamos. Te coges la suma y la divides entre el número de elementos.

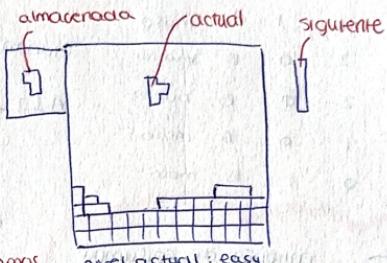
## ⑤ DONDE APLICARÍAMOS CLUSTERING? TEMA 5 pg. 9

SI

## HOJA 2 CONT.

### ④ TETRIS

- modulo q. captura eventos del mando para construir la q. juega con humano.  
almacenamos historico del estado de juego y acción  
entre
- MOV Izq M1
  - MOV Derecha MD
  - Rotar pieza R
  - Bajar pieza B
  - guardar pieza G



si queremos utilizar KNN...

### ① COMO CODIFICAMOS DATOS Y QUE MEDIDA DE DISTANCIA USAMOS?

Aqui hablamos del encoding (convertir binario, etc)

- tablero ( $m \times m$ ) atributos

- pieza actual

- posición de pieza

- siguiente pieza

- pieza guardada

- acción jugador (M1, ...)

- rotación (categoría) → habría que convertir en datos binarios que son cuantitativos.

### ② Para ver si sistema ha aprendido bien, que medida de similitud usarías?

M1, MD, R, B, M1, M1, B, G, MD, M1, MD, R, R, B

la idea es que el agente intente imitar lo mas posible al humano, so debería copiar basically sus acciones (con pequeño margen tipo M1 instead of MD, pero no M1 instead of B → aquí big difference)

medida de distancia podría ser:

• Si comprobando una a una viendo si las acciones son las mismas. Contamos veces que falla. misma = punto igual. Distinto = sumo fallos.

• Manhattan asumiendo q distancia entre valores es 1, sea cual sea el valor. LO MALO es que Manhattan repite fallos.

HOLA - HOAL

→ hay 2 fallos → 1 en la A → debería ser 1 en la L el mismo fallo.

• distancia de edición - WINNER!

Tiene sustitución de un valor por otro y rotación de los adyacentes, es decir cuenta un fallo en vez de dos.

HOLA - HOAL

→ daría un fallo en la A pues rotaría letras.

MI MD MI  
MI MI MD } rot same moves but same result  
permutación

mejor contarlo como 1 fallo y no 2.

## ⑤ ENTROPIA

A1	A2	A3	A4	clase
2	0	3	A	bueno
1	1	1	B	malo
2	2	0	B	bueno
1	0	1	B	bueno
0	1	0	A	malo
0	1	1	B	malo

1. Calcular cuál será atributo q. primero elegiré ID3.

$$A_1 = \{0, 1, 2, 3\}$$

Ganancia = freq. valores mult. por E para cada caso.

$$A_1 = \underbrace{\frac{1}{6} \cdot E_{10}}_0 + \underbrace{\frac{2}{6} \cdot E_{11}}_1 + \underbrace{\frac{2}{6} \cdot E_{12}}_2 + \underbrace{\frac{1}{6} \cdot E_{13}}_3$$

$E_{10}$  = Entropía de columna A1 para el valor 0.

$$\text{clase malo} \quad \text{valor 0} = \frac{\text{la fracción de 0 = malo}}{\text{veces que sale 0}}$$

$$\text{clase bueno} \quad \text{valor 0} = \frac{\text{la fracción de 0 = bueno}}{\text{veces que sale 0}}$$

shortcuts

$$-1 \cdot \log_2 1 = 0$$

$$-\frac{1}{2} \cdot \log_2 \frac{1}{2} + \left( -\frac{1}{2} \cdot \log_2 \frac{1}{2} \right) = 1$$

$$\underbrace{-0.5}_{=0.5} \quad \underbrace{=0.5}_{=0.5}$$

deja cosas q  
más clasificadas  
y con menos  
variabilidad.

$$A_2 = 0.33$$

$$A_3 = 0.795 \quad A_4 = 1$$

Si sale 1 sabemos  
que es malo.

$$A_1 = \frac{1}{6} \cdot 0 + \frac{2}{6} \cdot 1 + \frac{2}{6} \cdot 1 + \frac{1}{6} \cdot 0 = 0.66$$

$$A_1 = \frac{1}{6} \cdot 0 + \frac{2}{6} \cdot 1 + \frac{2}{6} \cdot 1 + \frac{1}{6} \cdot 0 = 0.66$$

⑥ escalar img usando alg de escalado: método conocido y rápido, no permite extraer info más allá de concentrar info en menos pixels.

autoencoder: permite extraer resumen de red más complejo y rico, encuentra nuevos patrones.

Pero se necesita un gran volumen de datos para entrenar

convolucional: permite detectar caras, concretas dentro de una img. solo detecta una cara, concreta por cada capa convolucional. No reduce la dimensionalidad tanto como pca.

pca: coge pixeles que hay y se queda con una transformación lineal de esos pixeles. y autoencoder.

No encuentra nuevos patrones ni caras de más alto nivel, solo trans. lineal.

⑦ que alg. clasifica mejor un prob. no linealmente separable (a priori)?

• arbol decisión: sabemos que no es lineal. pq al pintar el cluster, vemos que los datos no están segmentados entonces no podemos clasificarlos - (NO).

• red de neuronas con capas convolucionales: son un conjunto de funciones lineales ligeras. (NO)

• perceptrón multicapa: es el mejor ya que es multicapa! (si fuera una capa = no lo sería) con función Relu.

y ya q. Relu es una función de activación no lineal.

(podría ser sigmoidal).  $\star$  winner!

## PASOS A SEGUIR PARA UN PROBLEMA DE GENETICOS

1. Lo primero que tenemos que ver es cómo va a ser el vector o la matriz de genes; es decir, el individuo.
2. Normalmente, nos dan la función fitness de cada gen. En el caso de los anuncios es el precio, en el caso de las casillas es la protección que le da el sensor a la casilla, y en el caso del viajero es la distancia que hay entre las ciudades.
3. Tenemos que fijarnos en el enunciado para ver si lo que queremos es maximizar o minimizar la solución, y en base a esto; debemos crear los individuos con los genes que nos den una mejor función fitness. (no es necesario crear todos los individuos, sino algunos para ver la idea de cómo va a ser el problema)
4. De los individuos formados seleccionar cuáles serán los mejores candidatos para la reproducción. ¡OJO!: no debemos rechazar individuos cuya función fitness sea peor, puesto que pueden contener genes que a la hora de reproducirse sean favorables, y de esta manera evitamos caer en mínimos locales(se pierde variabilidad genética).
5. Generamos la población intermedia, probando a hacer las reproducciones con los individuos anteriores, y vemos si se mantienen las restricciones del enunciado, o si se nos generan individuos inválidos.
6. Normalmente, se nos generarán algunos sucesores inválidos (porque repitan números, porque tengan en cuenta casillas no disponibles, etc); en estos casos debemos pensar en las medidas que debemos aplicar a estos individuos. Algunas medidas:
  - Penalizar a esos individuos con un fitness muy malo para que sea improbable que sea elegido.
  - Descartar el individuo y volver a seleccionar otros dos para reproducirse.
  - Pensar en una nueva forma de realizar el problema para que se reduzcan el número de individuos inválidos, puesto que, aunque se descarten o se penalicen, es costoso el hecho de que se lleguen a crear esos individuos. **Ejemplo del problema de los anuncios:** tener en cuenta la posición de la lista en lugar del id del anuncio, y una vez ese anuncio sea elegido, eliminarlo de la lista y reajustarla, de forma que cambian las posiciones. ( La desventaja de esta codificación es que se necesita un proceso de traducción del gen para ver cuál es la solución correcta, ya que los valores dependen del orden que fijemos en la tabla inicial y la sol no es aplicable si el orden cambia).