# Integración de Bases de ConocimientoDatos

2do Cuatrimestre de 2020

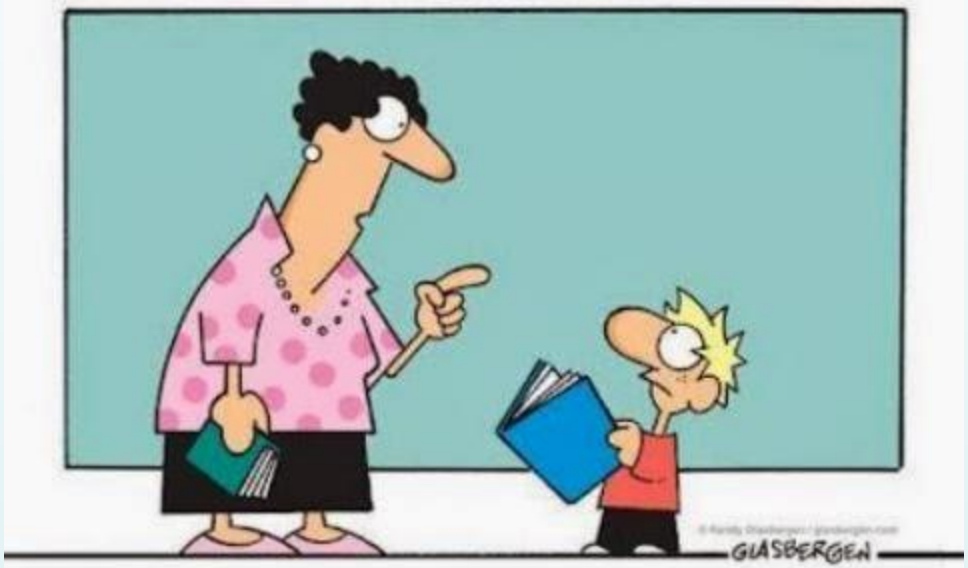
Clase 8: Aprendizaje

DEPARTAMENTO DE COMPUTACION

Facultad de Ciencias Exactas y Naturales - UBA

Profesora: Vanina Martinez
mvmartinez@dc.uba.ar

# Aprendizaje



# Aprendizaje


**"Once I learn how to use Google, isn't that all the education I really need?"**

# Aprendizaje

- La capacidad de *aprender* es una parte esencial de la inteligencia.
- Hasta ahora hemos supuesto que la inteligencia del agente fue *"incorporada"* en él por el diseñador.
- Luego, el agente se introduce en el entorno actuando lo mejor posible de acuerdo a su programación.
- Pero esto no es la mejor alternativa.

# Aprendizaje

- Normalmente, se diseña un agente para un entorno sobre el que se tiene *conocimiento incompleto*.
- La introducción de alguna forma de aprendizaje es la única manera de lograr que adquiera el conocimiento que necesitará.
- Así, el aprendizaje introduce un grado de autonomía mayor en el agente.
- También, al mejorar su conocimiento del entorno por el aprendizaje el agente puede mejorar su desempeño.

# Aprendizaje

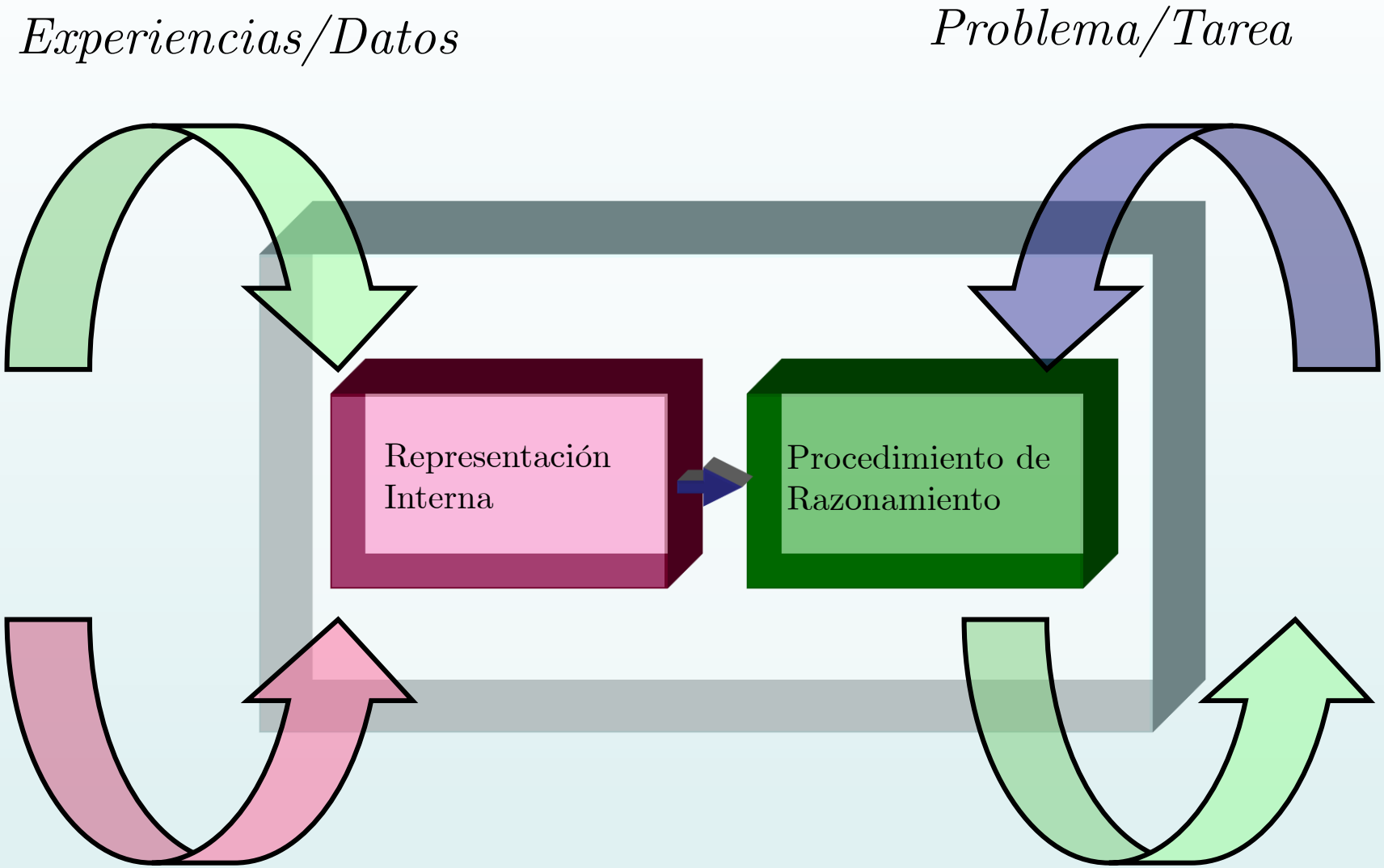
*Aprender* es la habilidad de mejorar el comportamiento utilizando la experiencia propia:

- ✓ El rango de comportamiento se expande: es decir, el agente adquiere la capacidad de hacer más.
- ✓ Se mejora la precisión con que se ejecutan las tareas: esto es, al aprender el agente puede mejorar su desempeño.
- ✓ Se acelera la ejecución de las tareas: es decir, el agente puede completar sus tareas más rápidamente.

# Aprendizaje

Los siguientes *componentes* son parte de cualquier problema de aprendizaje:

- ✓ Tarea: Es el comportamiento o tarea que se mejora.
- ✓ Datos: Representan las experiencias que se utilizan para mejorar la perfomance en la tarea.
- ✓ Medida de la mejora: Es la forma en como se va a evaluar la mejora en el desempeño de la tarea.




Arquitectura para el Aprendizaje

# Representación

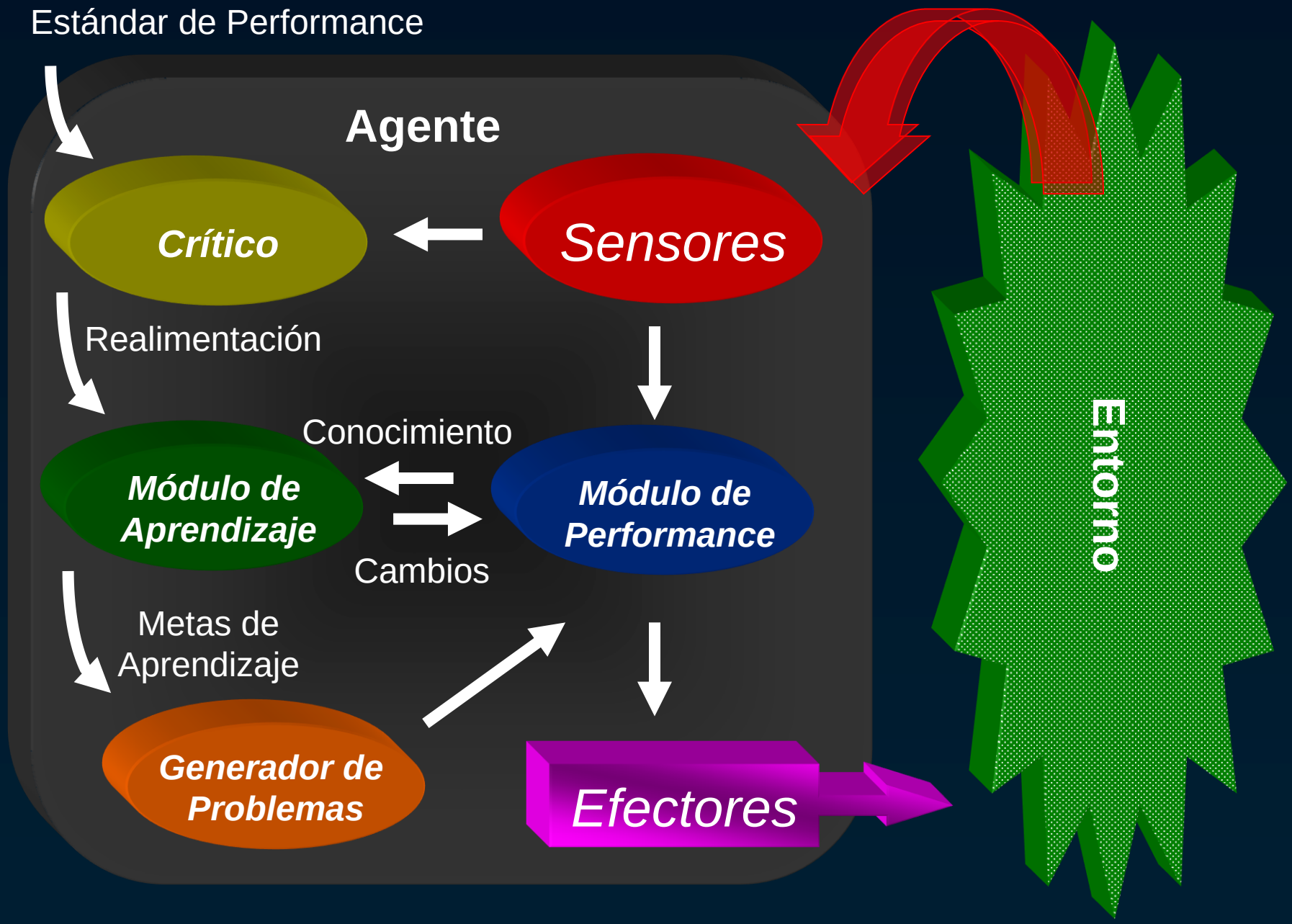
- En la figura se observa que se toman *datos*, *experiencias*, *conocimiento de fondo* (*background*) y cierta inclinación o tendencia para interpretar todo (*bias*) para obtener una *representación interna de los datos*.
- Se denomina *inducción* al problema de inferir una representación interna (*modelo*) que produce las observaciones.
- Sobre el tipo de representación que se desea obtener subsisten los problemas clásicos de complejidad versus utilidad.

# Representación

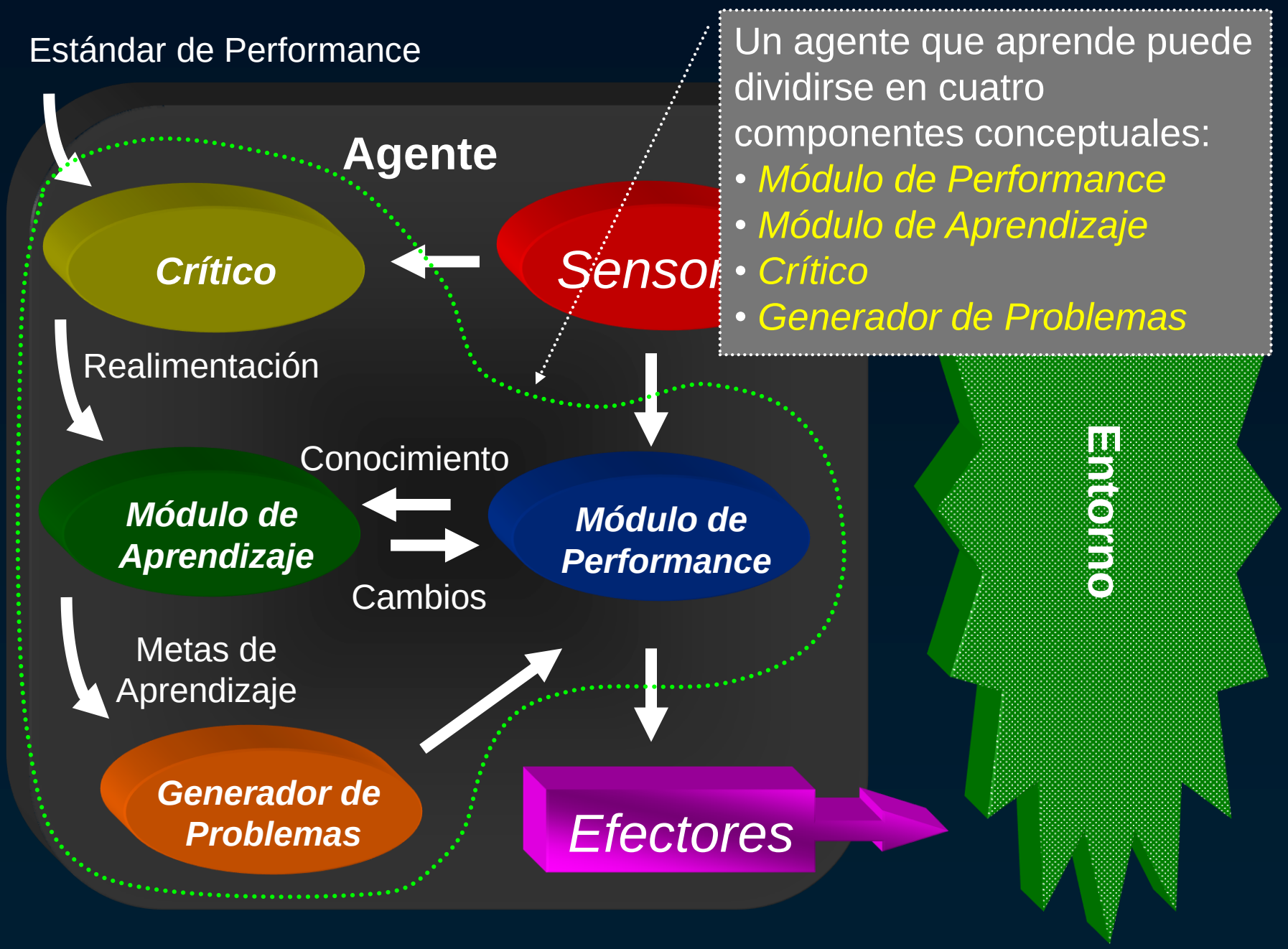
El compromiso que se establece al elegir la representación sigue dos ejes sobre los que hay que establecer un balance:

- Cuanto más rica y elaborada es la representación *más útil* resulta en la resolución de problemas.
- Cuanto más rica y elaborada es la representación *más difícil* es aprender.

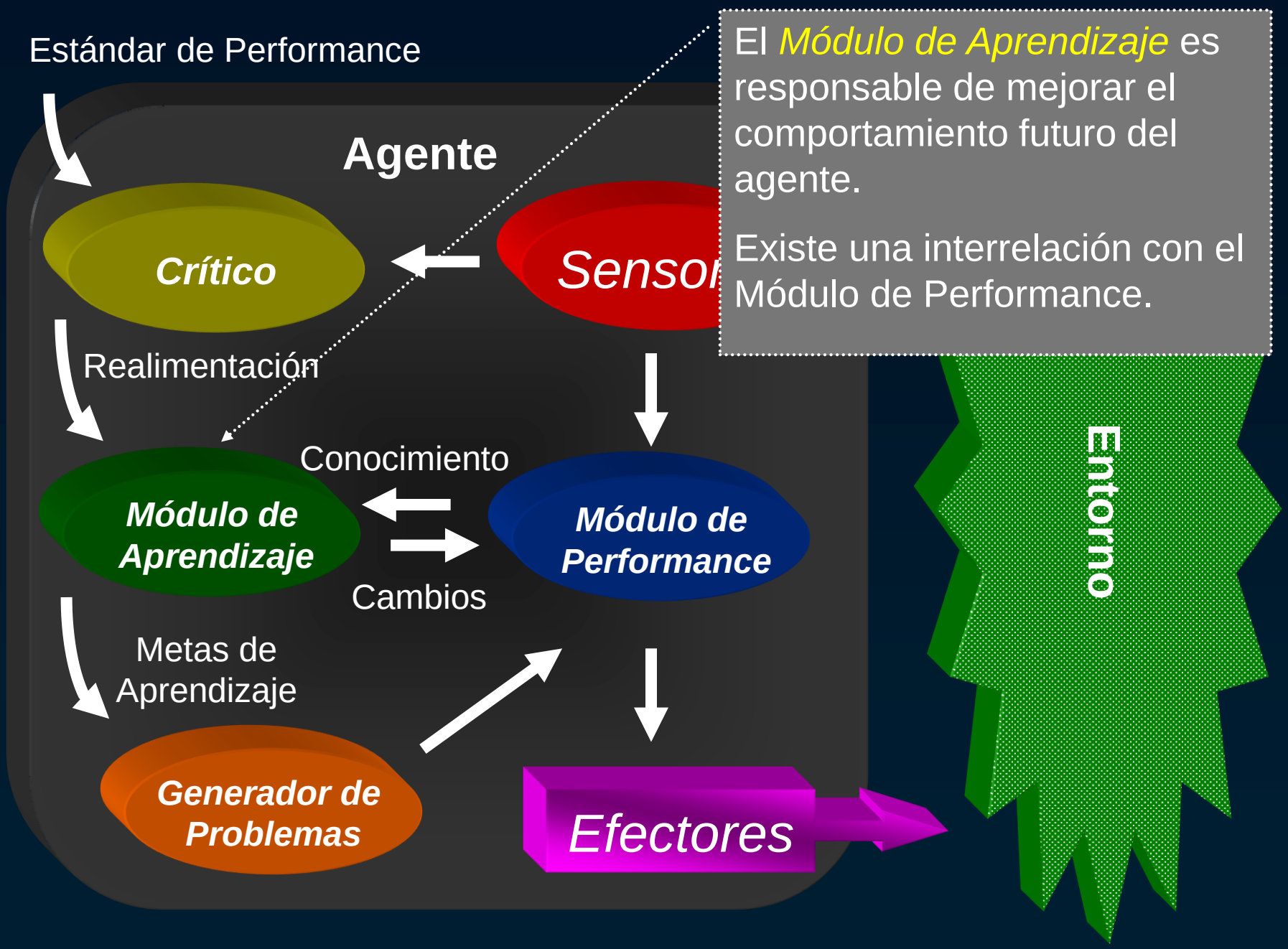
# Arquitectura para un Agente que Aprende



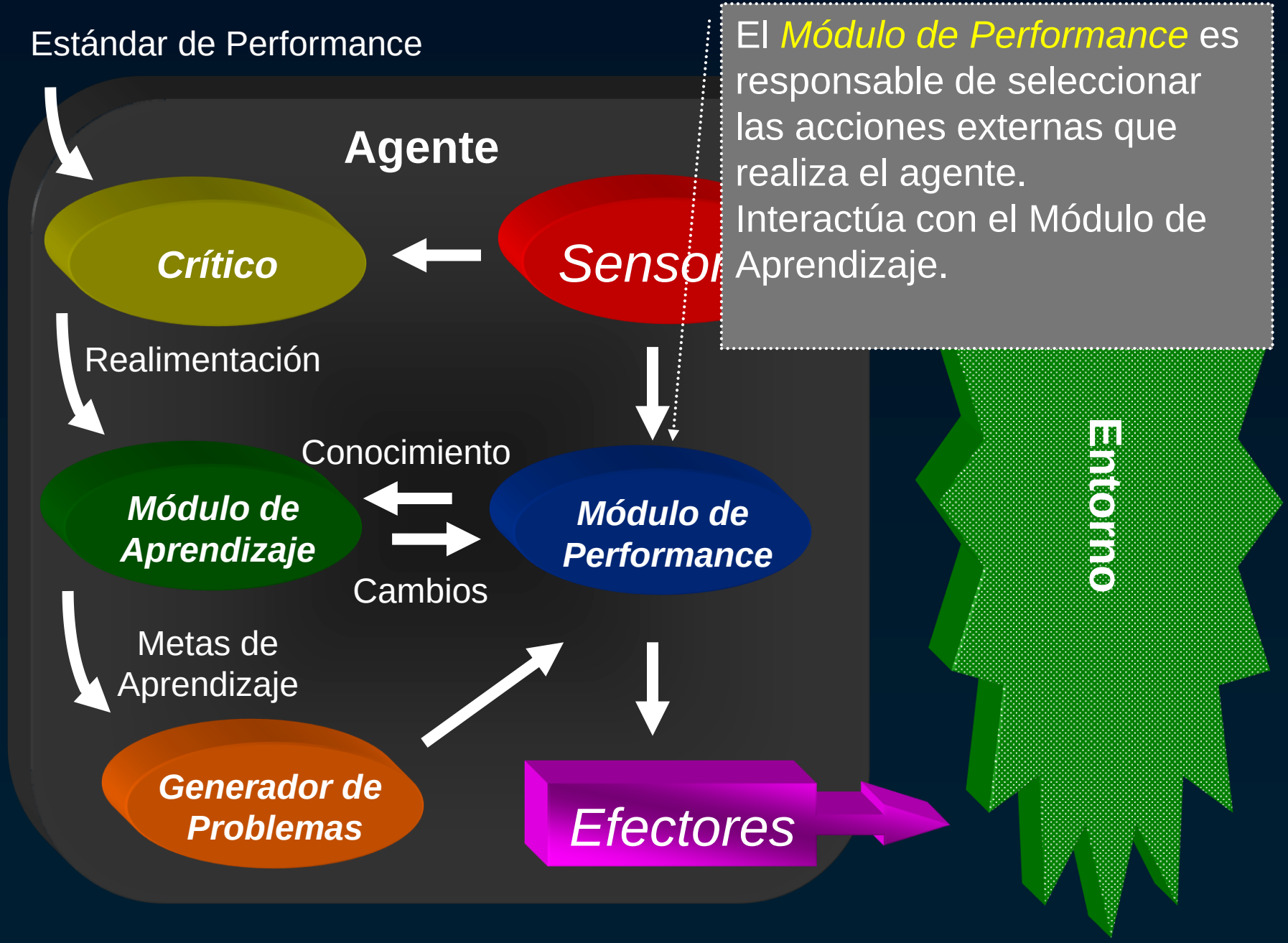

Un modelo general de Agentes que aprenden.



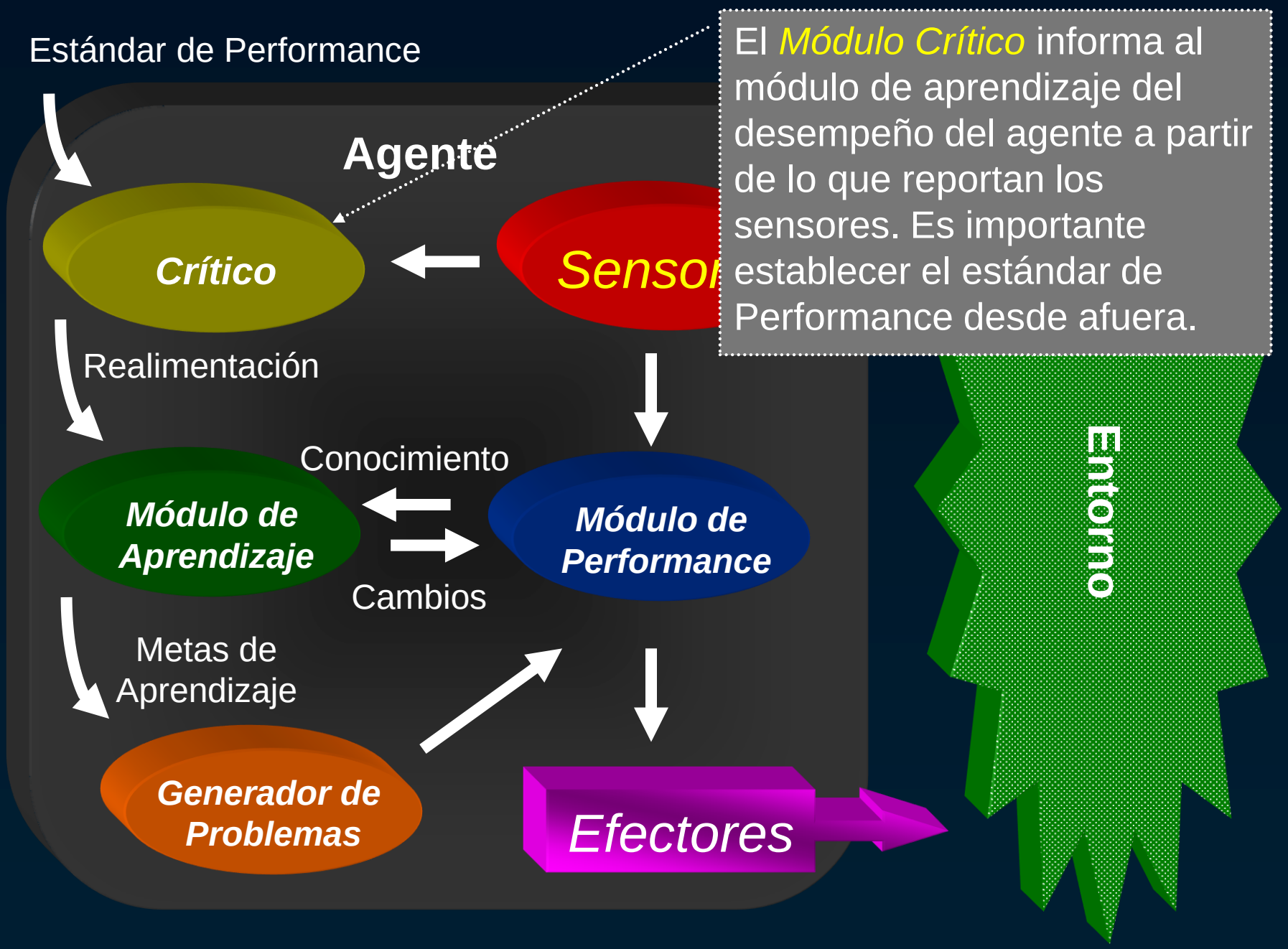

Un modelo general de Agentes que aprenden.



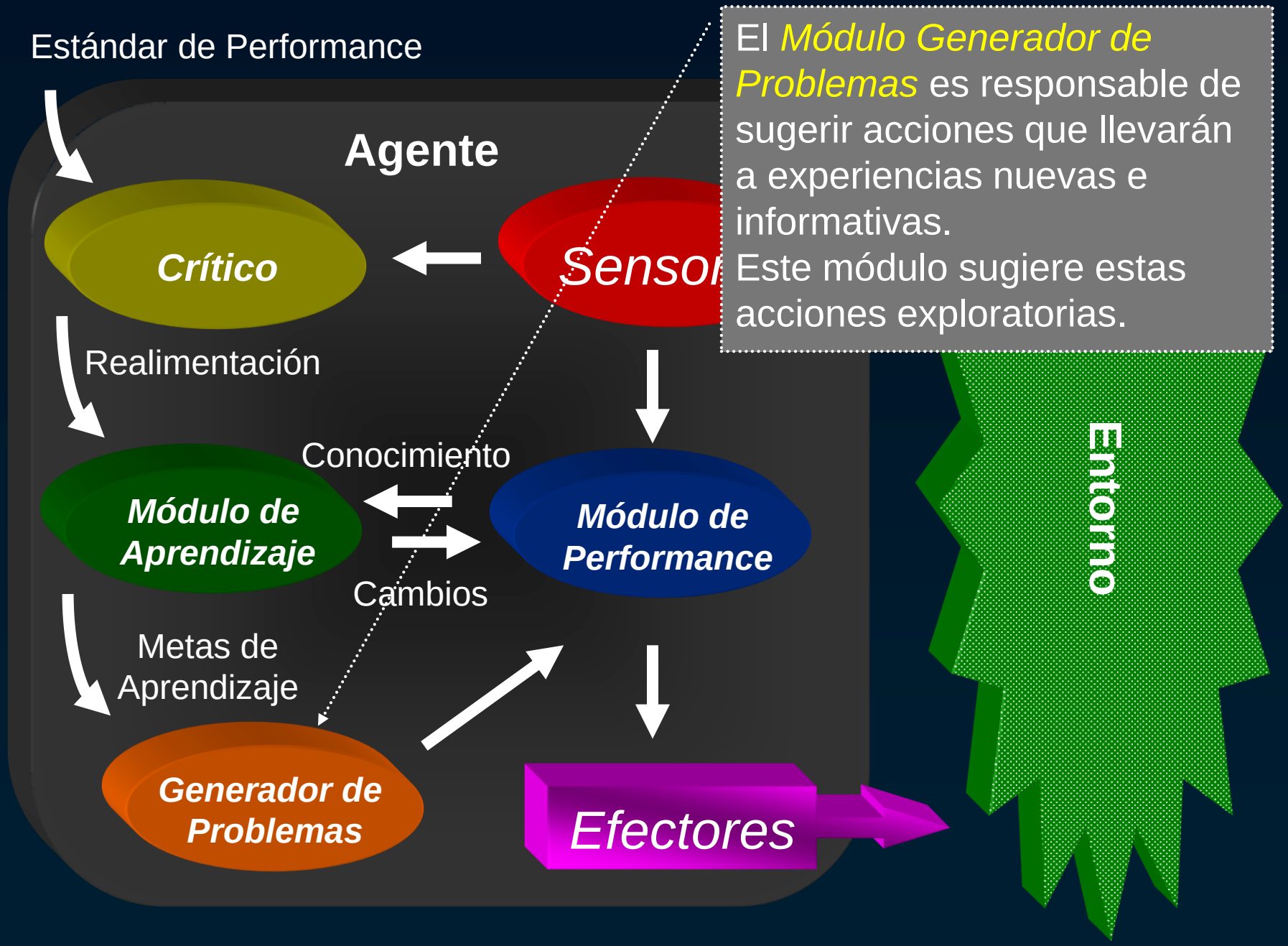

Un modelo general de Agentes que aprenden.




Un modelo general de Agentes que aprenden.




Un modelo general de Agentes que aprenden.




Un modelo general de Agentes que aprenden.

# Temas Importantes en Aprendizaje

# Tareas de Aprendizaje

- Clasificación Supervisada*:* A partir de un conjunto de ejemplos de entrenamiento *preclasificados* clasificar una nueva instancia.
- Aprendizaje no Supervisado*:* Encontrar *clases* que resulten naturales para los ejemplos.
- Aprendizaje por Refuerzo*:* Determinar *qué* es importante en función de *premios* y *castigos*.
- Aprendizaje Analítico*:* Razonar rápidamente *utilizando experiencia*.
- Programación en Lógica Inductiva*: Enriquecer los Modelos* utilizando programación en lógica.

# Ejemplo de Tarea de Clasificación

- Veremos un ejemplo en el que un *infobot* observa actuar a un usuario.
- El usuario lee listas de discusión de determinadas características.
- Selecciona artículos para leer basandose en varios parámetros:
  - El autor es conocido o no (Author).
  - El artículo comenzó una nueva hebra de discusión o es continuación de una existente (Thread).
  - El artículo es largo o corto (Length).
  - El lugar de lectura del usuario, su casa o su trabajo (Where).

# Ejemplo (cont.)

| | Action | Author | Thread | Length | Where |
|---|---|---|---|---|---|
| e1 | skips | known | new | long | home |
| e2 | reads | unknown | new | short | work |
| e3 | skips | unknown | follow_up | long | work |
| e4 | skips | known | follow_up | long | home |
| e5 | reads | known | new | short | home |
| e6 | skips | known | follow_up | long | work |

*Computational Intelligence* D. Poole, A. Mackworth, R. Goebel

Se busca clasificar nuevos ejemplos usando la propiedad Action utilizando como ejemplos Author, Thread, Length y Where.

# Feedback

Las tareas de aprendizaje pueden caracterizarse por la realimentación ($feedback$) que recibe el agente:

- Aprendizaje Supervisado: Se especifica que se debe aprender por cada ejemplo.
- Aprendizaje no Supervisado: No se dan clasificaciones y el agente debe descubrir categorías y regularidades en los datos.
- Aprendizaje por Refuerzo: La realimentación sucede cuando se completa una secuencia de acciones.

# Ponderación del Éxito

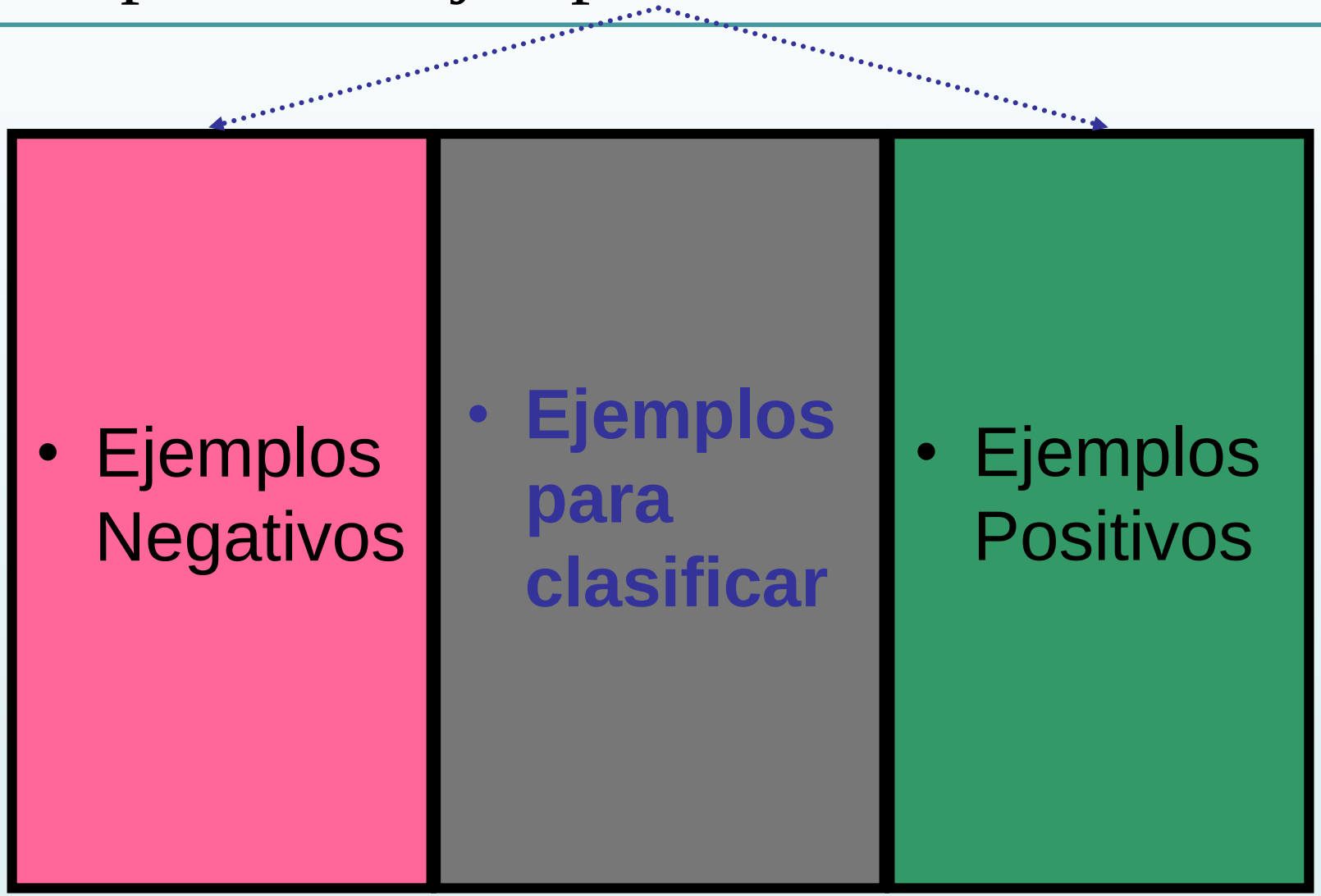
- La medida del éxito debe tomarse sobre la performance del agente en los ejemplos nuevos y no sobre los ejemplos de entrenamiento.

Consideremos dos agentes:

- $P$ que afirma que los ***ejemplos negativos*** que se observan son los únicos que existen, *i.e.* *todos los otros ejemplos son Positivos*.
- $N$ que afirma que los ***ejemplos positivos*** que se observan son los únicos que existen, *i.e.* *todos los otros ejemplos son Negativos*.

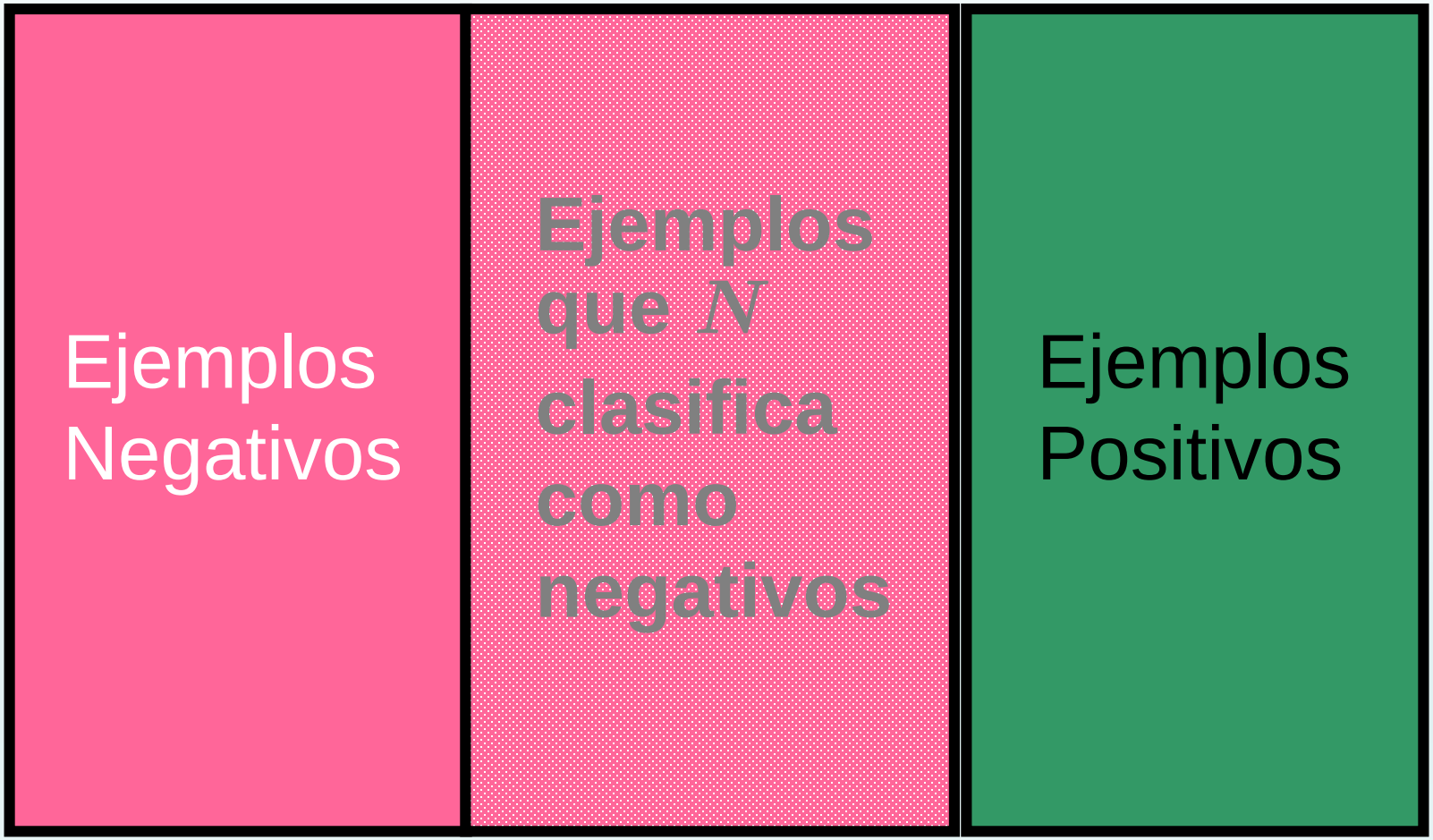
*Notemos que ambos agentes clasifican correctamente los ejemplos de entrenamiento pero están en desacuerdo en todos los nuevos ejemplos.*

# Espacio de Ejemplos de Entrenamiento



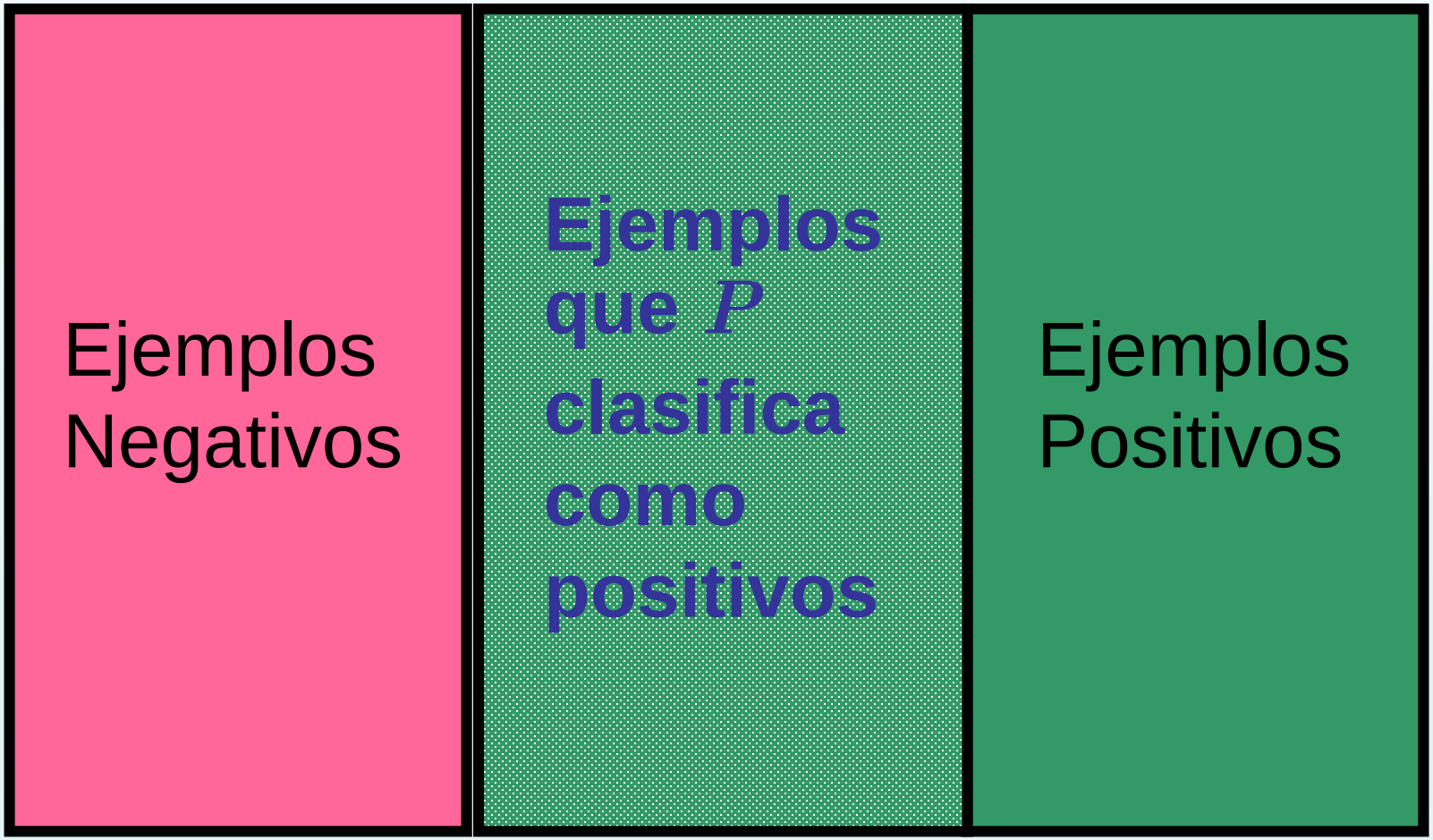
# Agente $N$

Para $N$ los ejemplos positivos que se observan son los únicos que existen



# Agente $P$

Para $P$ los ejemplos negativos observados son los únicos que existen



# Bias

- La tendencia a preferir una hipótesis sobre otra se denomina *bias*.
- Preferir una hipótesis sobre los extremos $N$ y $P$ es algo que no se pueda hacer a partir de los datos.
- Para poder realizar predicciones en un proceso inductivo el agente necesita adoptar algún tipo de *bias.*
- La selección de un *bias* es un proceso empírico que permite verificar que es lo que trabaja mejor en la práctica.

# Aprendizaje y Búsqueda

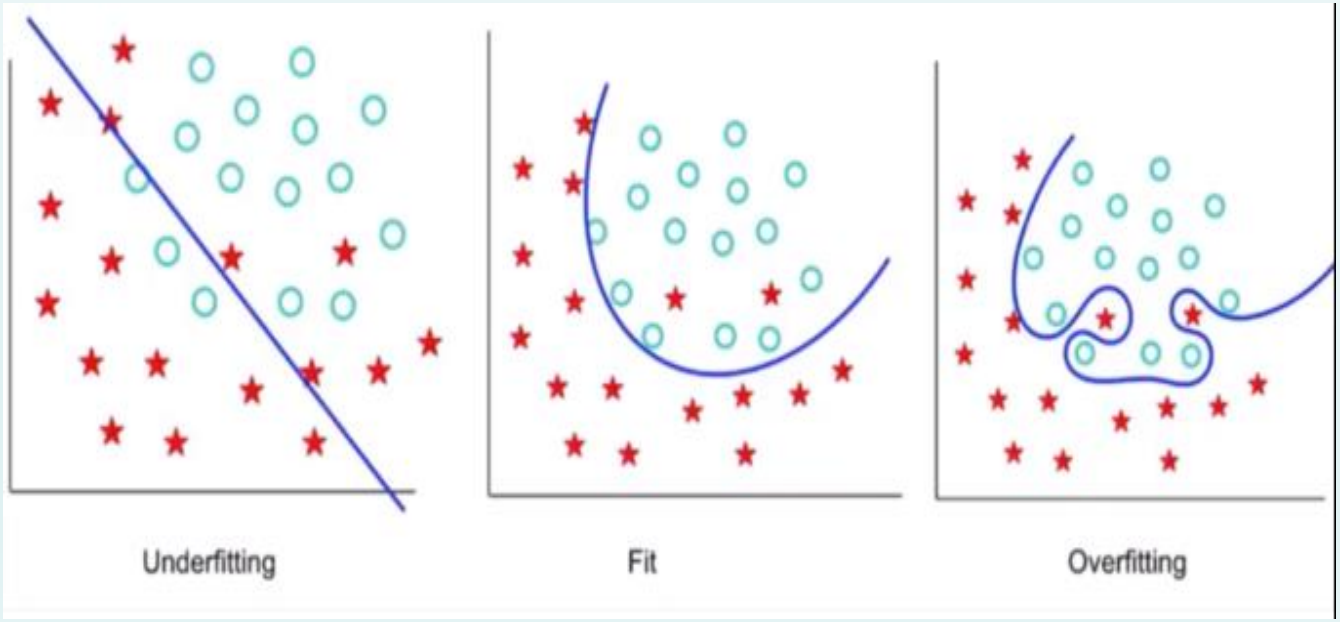
- Dada una representación y un *bias* el problema de aprendizaje puede reducirse a un *problema de búsqueda*.
- *Aprender es buscar en el espacio de posibles representaciones,* considerando la representación o representaciones, que mejor se ajustan a los datos para un bias dado.
- Estos espacios de búsqueda típicamente resultan prohibitivamente grandes para búsqueda ciega.
- Un algoritmo de aprendizaje comprende un *espacio de búsqueda*, una *función de evaluación* y un *método de búsqueda* (usando algún método heurístico).

# Problemas con los Datos

- Algunos de los atributos de los datos reciben un *valor erróneo*.
- Los atributos *se eligen de manera equivocada* para predecir la clasificación.
- Hay ejemplos en los que faltan datos de algunos atributos, no necesariamente los mismos en cada dato.
- Los datos no son representativos del fenómeno que se desea aprender:
  - O bien están mal recolectados, o no es posible encontrar un conjunto de datos adecuados. Ej., en [Popejoy2016], se estudio que de 2,511 estudios el 81% de los participantes de estudios de mapeo genético son descendientes de Europeos.

# Problemas con los Datos

*Sobre Ajuste* ( *overfitting* ) se presenta cuando la distinción aparece en los datos pero no aparece en los ejemplos aún no vistos. Esto puede ser provocado por las características del conjunto de entrenamiento.



# Reglas de Asociación

# Reglas de Asociación

- Generar reglas del tipo:
  - IF (SI) condición ENTONCES (THEN) resultado
- Ejemplo:
  - **Si** producto B ENTONCES producto C
- Minado de reglas de asociación:
  - “Encontrar patrones, asociaciones, correlaciones o estructuras causales frecuentes entre conjuntos de items u objetos in bases de datos transaccionales, relacionales u otro tipo de repositorios de información.”

# Tipos de reglas según su utilidad

- Utiles/aplicables: reglas que contienen buena calidad de información que pueden traducirse en acciones de negocio.
- Triviales: reglas ya conocidas en el negocio por su frecuente ocurrencia
- Inexplicables: curiosidades arbitrarias sin aplicación práctica

# ¿Cuán buena es una regla?

- Medidas que califican a una regla:
  - Soporte
  - Confianza
  - Lift (Improvement)

# Ejemplo Soporte

- Es la cantidad (%) de transacciones en donde se encuentra la regla.
  - Ej : "Si B entonces C" está presente en 4 de 6 transacciones.
  - Soporte (B/C) : 66.6%

T1 = {A, B, C, D}

T2 = {B, C}

T3 = {A, B, C}

T4 = {B, C, D}

T5 = {A, D}

T6 = {A, B}

# Confianza

- Cantidad (%) de transacciones que contienen la regla referida a la cantidad de transacciones que contienen la cláusula condicional
  - Ej : Para el caso anterior,  B está presente en 5 transacciones (83.33%)
  - Confianza (B/C) = 66.6/83.3 = 80%

T1 = {A, B, C, D}

T2 = {B, C}

T3 = {A, B, C}

T4 = {B, C, D}

T5 = {A, D}

T6 = {A, B}

# Mejora (Improvement)

- Capacidad predictiva de la regla:
  - Mejora = p(B/C) / p(B) * p(C)
  - Ej:

p(B/C) = 0,67 ; p(B) = 0,833;
p(C) = 0,67

Improv(B/C) = 0,67(0,833*0,67)
= 1.2

Mayor a 1 : la regla tiene valor predictivo

T1 = {A, B, C, D}

T2 = {B, C}

T3 = {A, B, C}

T4 = {B, C, D}

T5 = {A, D}

T6 = {A, B}

# Tipos de Reglas

- Booleanas o cuantitativas (de acuerdo a los valores que manejan)
  - buys(x, “SQLServer”) ^ buys(x, “DMBook”) $\rightarrow$ buys(x, “DBMiner”) [0.2%, 60%]
  - age(x, “30..39”) ^ income(x, “42..48K”) $\rightarrow$ buys(x, “PC”) [1%, 75%]
- Una dimensión o varias dimensiones
- Con manejo de jerarquías entre los elementos (taxonomías que vienen dadas) o con elementos simples.

# Árboles de Decision

# Árboles de Decisión

- La *representación* es un árbol de decisión (que describiremos en un momento).
- El *bias* se inclina hacia la preferencia de los árboles de decisión simples.
- La *búsqueda* se realiza a través del espacio de los árboles de decisión de los más simples a los más complejos.

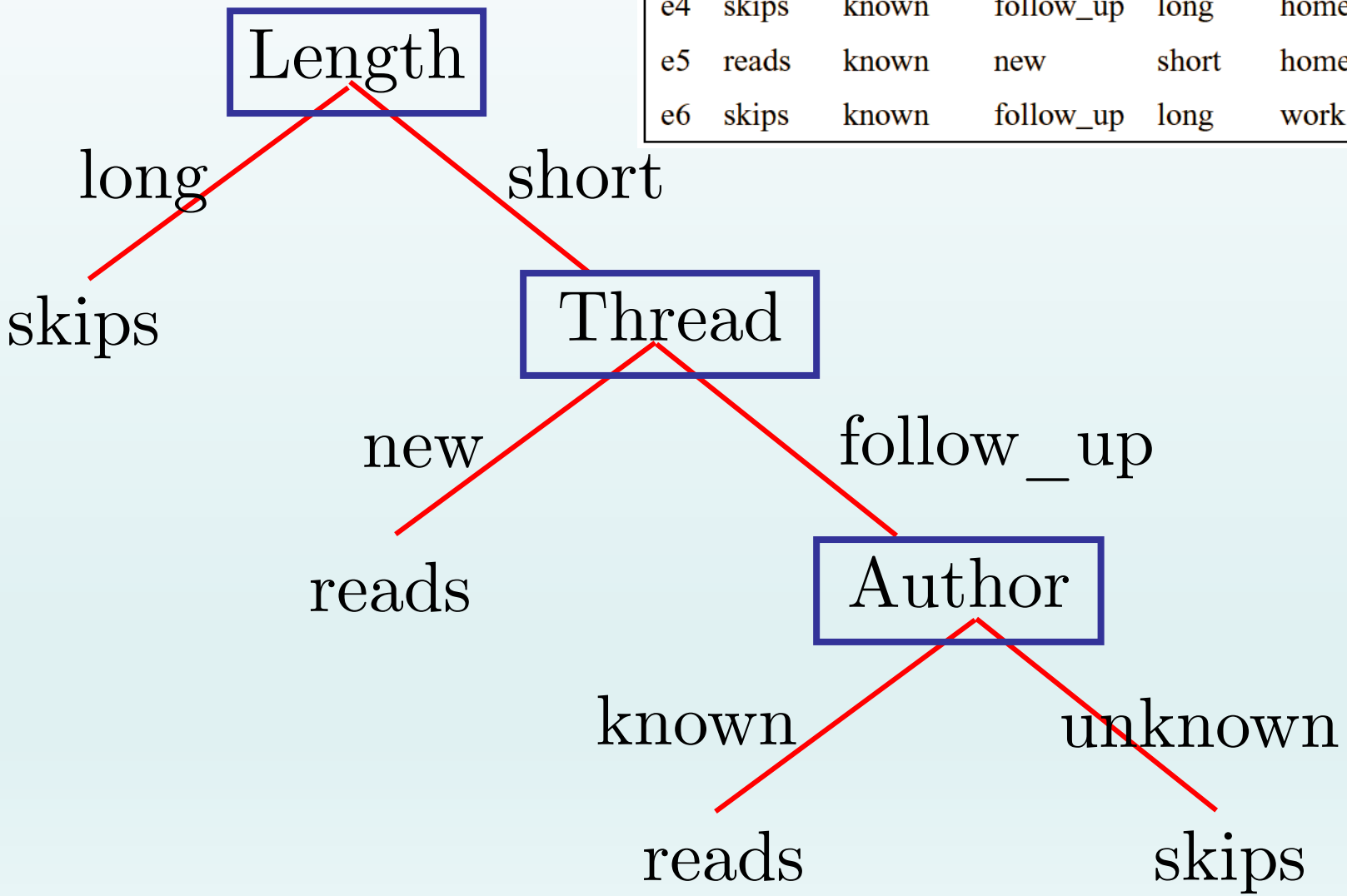
# Árboles de Decisión

Un árbol de decisión es un árbol donde:

- Los nodos se etiquetan con $atributos$ que caracterizan a los datos o experiencias.
- Los arcos salientes de un nodo etiquetado con el atributo $A$ se etiquetan con cada uno de los $posibles\ valores$ del atributo $A$.
- Las hojas del árbol se etiquetan con $clasificaciones$.

# Ejemplo

| | Action | Author | Thread | Length | Where |
|---|---|---|---|---|---|
| e1 | skips | known | new | long | home |
| e2 | reads | unknown | new | short | work |
| e3 | skips | unknown | follow_up | long | work |
| e4 | skips | known | follow_up | long | home |
| e5 | reads | known | new | short | home |
| e6 | skips | known | follow_up | long | work |



# Aprendizaje de Árboles de Decisión

- Dado un conjunto de datos ¿cuál es el árbol de decisión generado?
- Se necesita un bias. Por ejemplo, preferir el más pequeño, o el menos profundo, o el que tiene menos nodos.
- ¿Cuáles son los mejores árboles que predicen los datos aún desconocidos?
- ¿Cómo se construye un árbol de decisión? El espacio de los árboles de decisión es muy grande para utilizar búsqueda sistemática por el árbol más pequeño generando todos los posibles.

# Búsqueda de Arboles de Decisión

Input: atributo destacado (la META), un conjunto de ejemplos y un conjunto de atributos:

1. *Parar si todos los ejemplos tienen la misma clasificación.*
2. *Si no es así, elegir un atributo en el que se debe producir el split:*
   - *Por cada valor de este atributo construir un subárbol para aquellos ejemplos con este valor.*

# Búsqueda de Arboles de Decisión

**function** DTL(*examples, attributes, default*) **returns** a decision tree

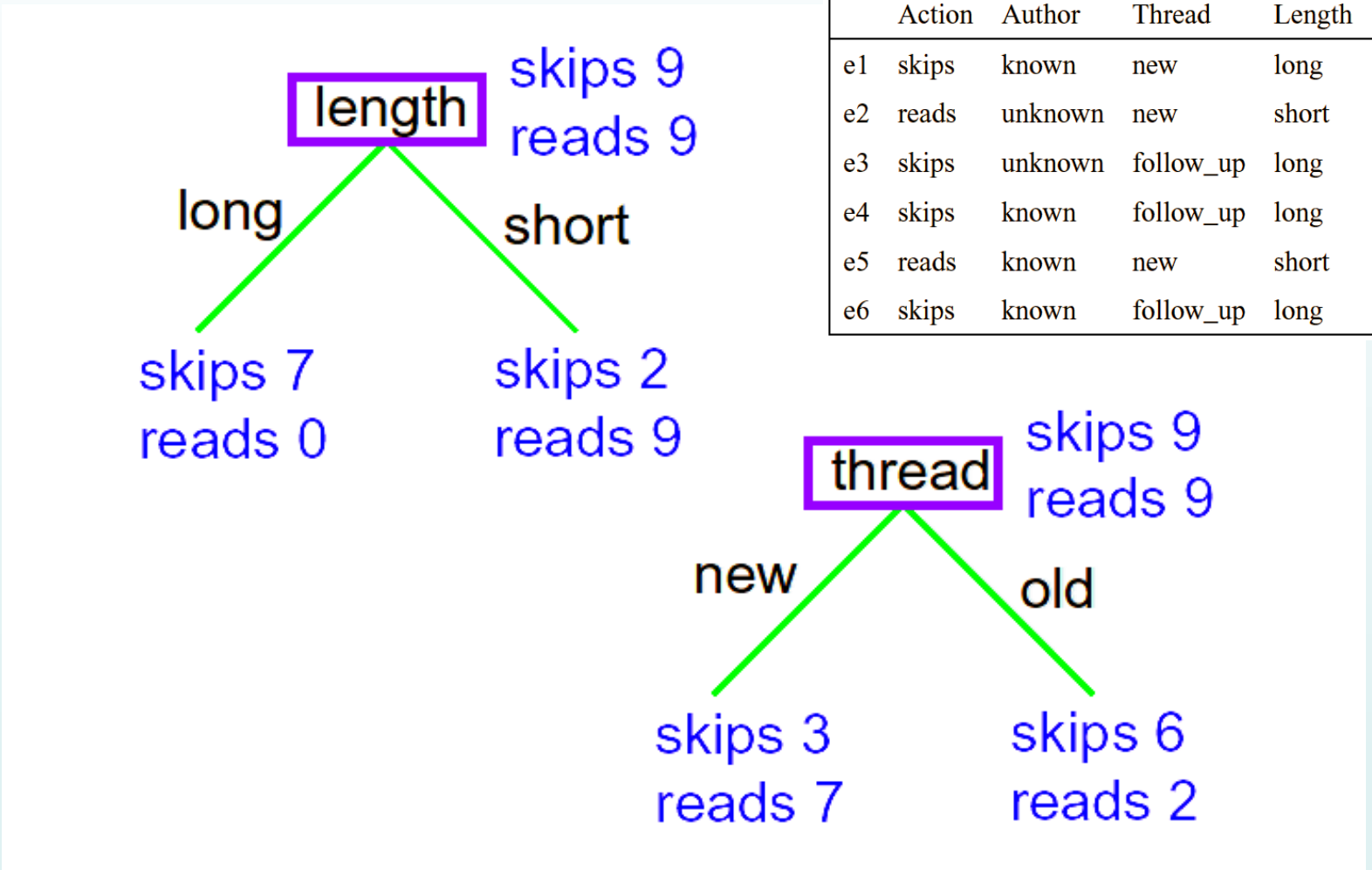
**if** *examples* is empty **then return** *default*
**else if** all *examples* have the same classification **then return** the classification
**else if** *attributes* is empty **then return** MODE(*examples*)
**else**
  $best \leftarrow$ CHOOSE-ATTRIBUTE(*attributes, examples*)
  $tree \leftarrow$ a new decision tree with root test *best*
  **for each** value $v_i$ of *best* **do**
    $examples_i \leftarrow$ {elements of *examples* with $best = v_i$}
    $subtree \leftarrow$ DTL($examples_i, attributes - best,$ MODE(*examples*))
    add a branch to *tree* with label $v_i$ and subtree *subtree*
  **return** *tree*

# Observaciones sobre el Algoritmo

- Es posible que los atributos tengan más de dos valores y esto complica la topología de los árboles.
- En el algoritmo se supone que los atributos son adecuados como representación del concepto que tratan de representar (es posible utilizar probabilidades para esto).
- En principio no está definido cual es el atributo que se debe elegir para Split.
- Idea: un buen atributo divide los ejemplo en subconjuntos que son (idealmente) “todos positivos” o “todos negativos”.

# Splits posibles

| | Action | Author | Thread | Length | Where |
|---|---|---|---|---|---|
| e1 | skips | known | new | long | home |
| e2 | reads | unknown | new | short | work |
| e3 | skips | unknown | follow_up | long | work |
| e4 | skips | known | follow_up | long | home |
| e5 | reads | known | new | short | home |
| e6 | skips | known | follow_up | long | work |




Computational Intelligence D. Poole, A. Mackworth, R. Goebel

# Overfitting

- El algoritmo que presentamos tiene problemas con el overfitting de los datos y esto es motivado por la aparición de ruido y correlaciones en el conjunto de entrenamiento que el conjunto completo de los datos no contiene.
- Para mejorar esa situación es posible:
  - Restringir el split de manera de únicamente usarlo cuando es útil, o
  - Se puede permitir el split irrestricto pero luego podar el árbol resultante donde se observen distinciones que no deberían existir.

# Redes Neuronales

# Redes Neuronales

- Se inspiran en las *neuronas biológicas* y su *esquema de conexiones en el cerebro*, pero no simulan al sistema nervioso.
- También se los conoce como Sistemas Conexionistas o Sistemas de Procesamiento Paralelo Distribuido o Parallel Distributed Processing ( *PDP* ).
- La inteligencia *aparece* en sistemas compuestos de elementos simples (neuronas naturales o artificiales) que interactúan.
- El paralelismo se corporiza en el procesamiento simultáneo de todas las neuronas y está distribuido en toda la red.

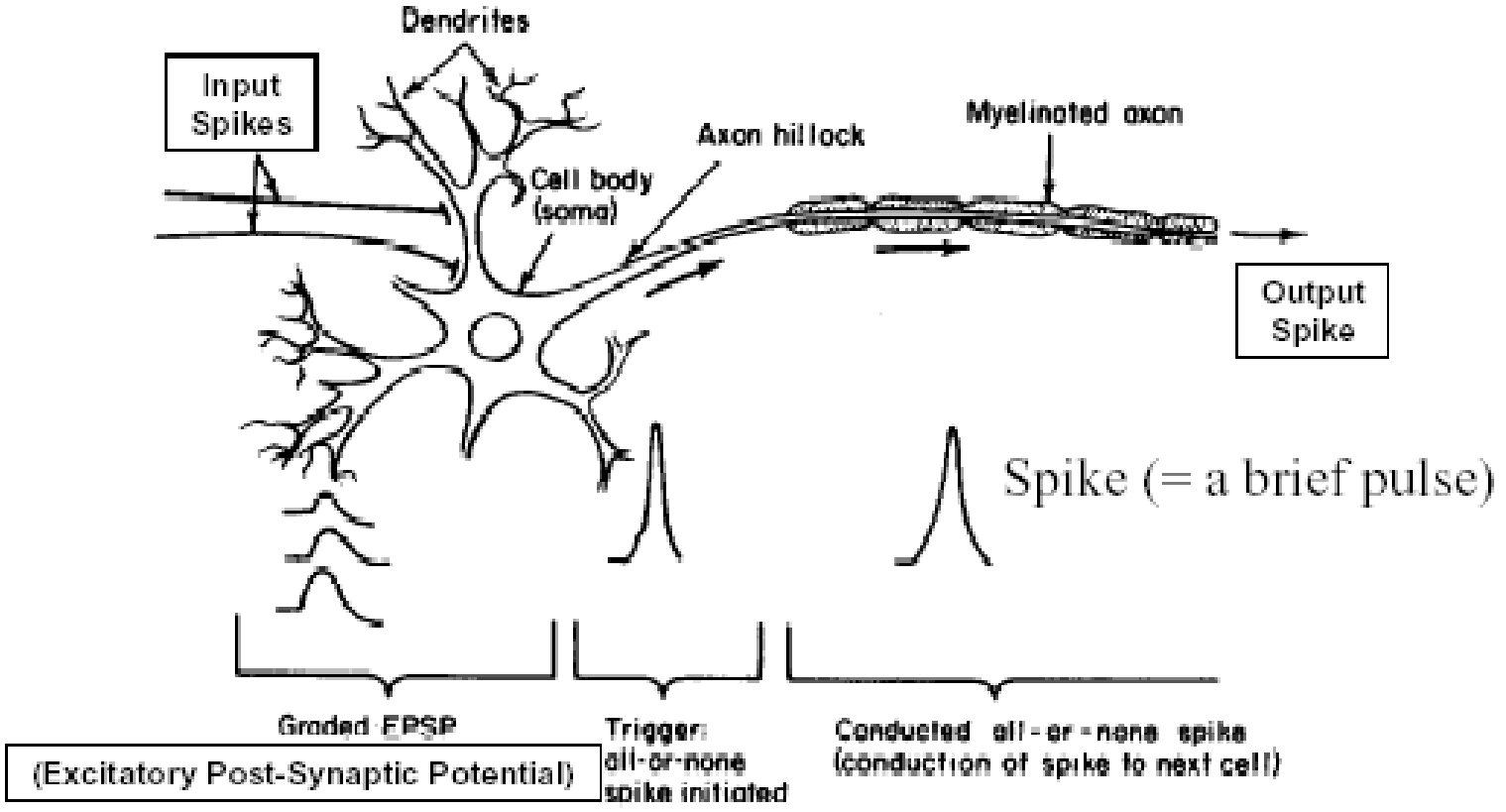
# Observaciones

- Como parte de la investigación biológica del sistema nervioso (neurología), los investigadores han desarrollado simulaciones de los sistemas neurales de animales simples tales como ciertos gusanos.
- La idea de construir la funcionalidad del cerebro simulando su estructura física, con abstracciones adecuadas, aparece como un objetivo válido.
- En respuesta, este estudio del cerebro sugiere nuevas maneras de conceptualizar el significado de *computar*.
- Las Redes Neuronales dan una manera simple de introducir el bias en el aprendizaje.

# Redes Neuronales

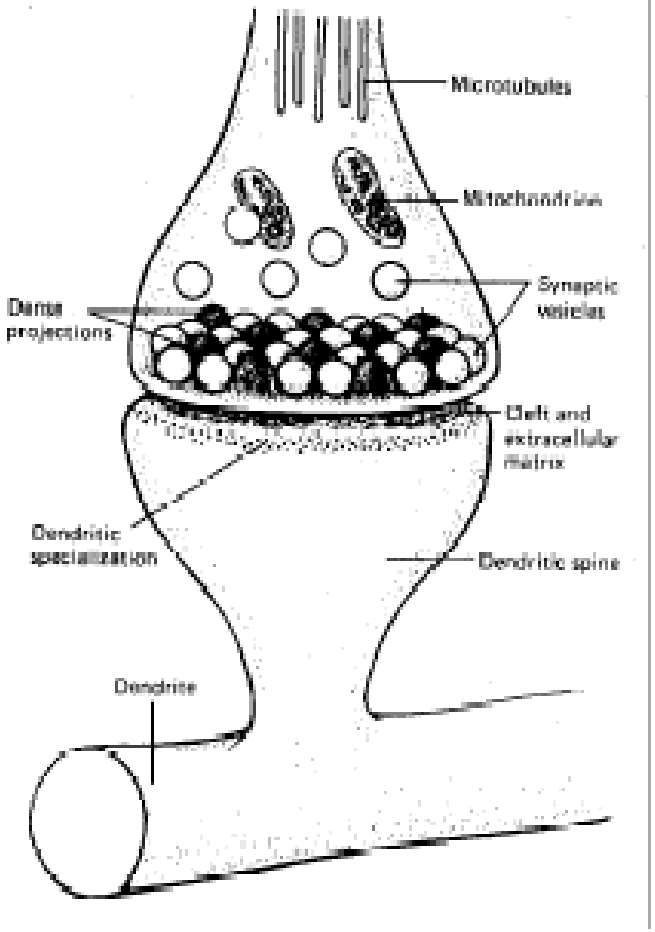
- Las *neuronas artificiales*, también llamadas *unidades*, tienen *entradas* y una *salida.*
- La salida de una neurona artificial puede conectarse a las entradas de otra.
- Esta salida es una función *no-lineal* de sus entradas y estas representan sus parámetros.
- El aprendizaje ocurre al ajustar los parámetros para hacerlos corresponder a los datos.
- Las Redes Neuronales, multicapa, pueden representar una aproximación a cualquier función lógica.

# Basic Input-Output Transformation in a Neuron

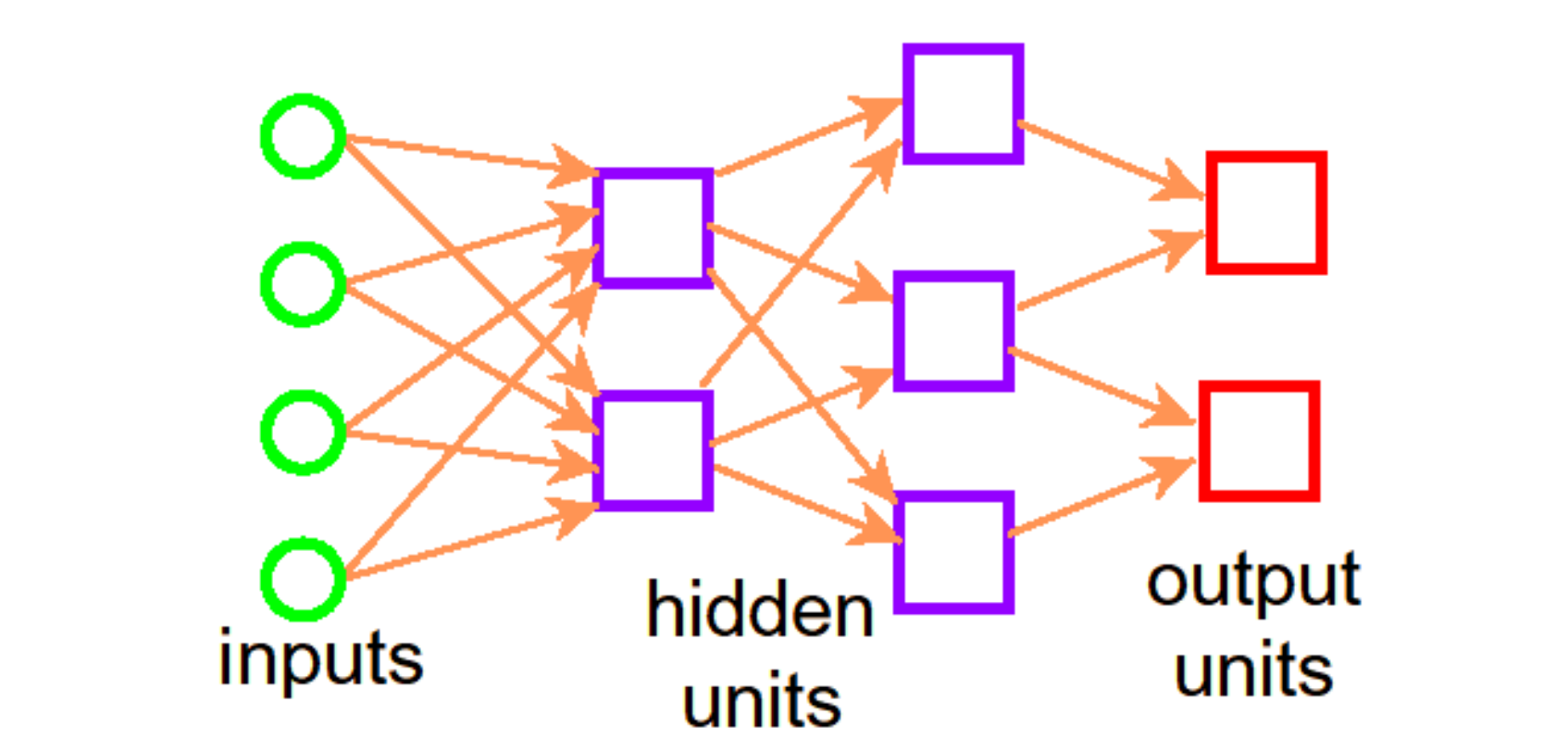


# Communication between neurons: Synapses

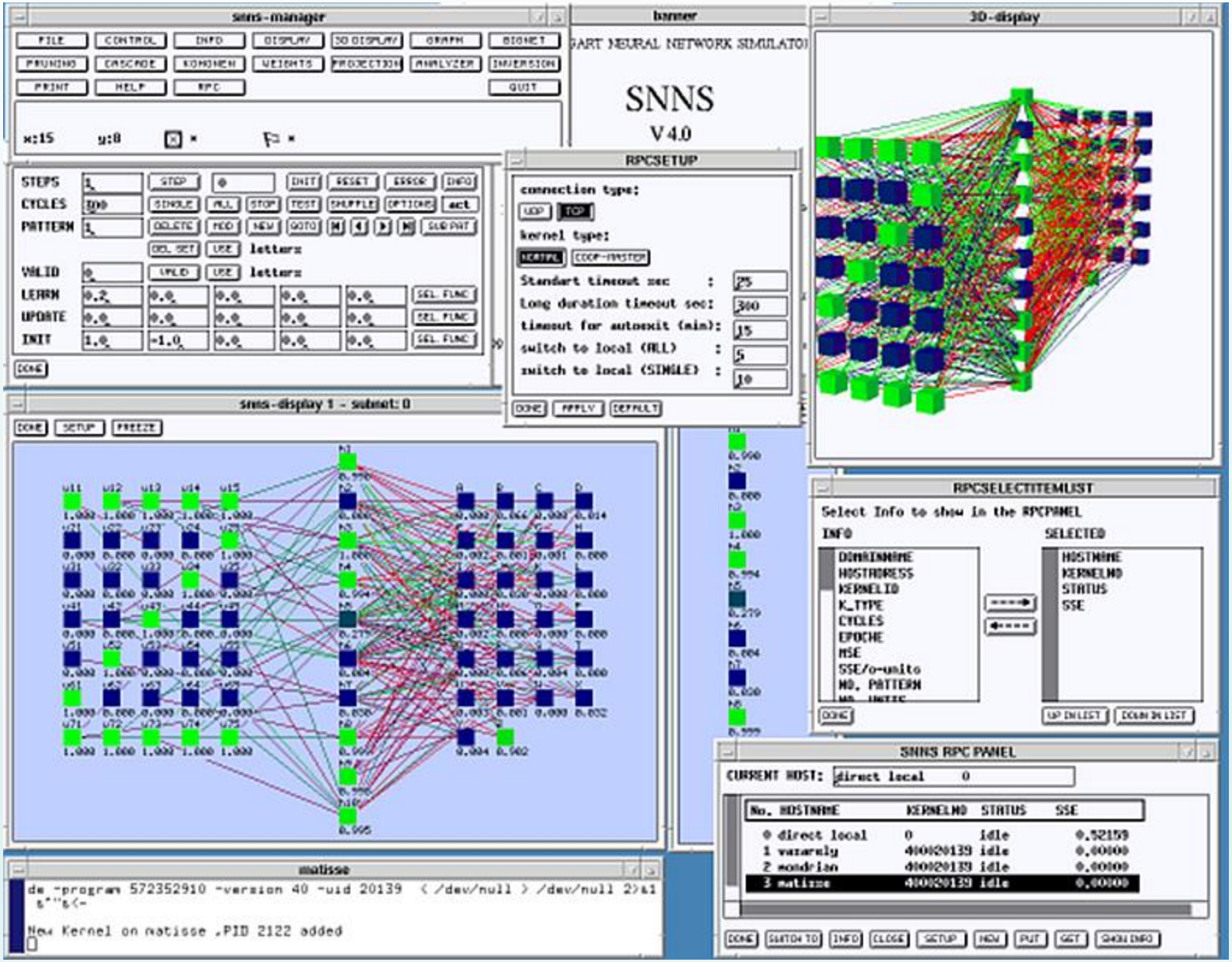
- Synapses: Connections between neurons
  - Electrical synapses (gap junctions)
  - Chemical synapses (use neurotransmitters)
- Synapses can be excitatory or inhibitory
- Synapses are integral to memory and learning



# Redes Neuronales




Computational Intelligence D. Poole, A. Mackworth, R. Goebel

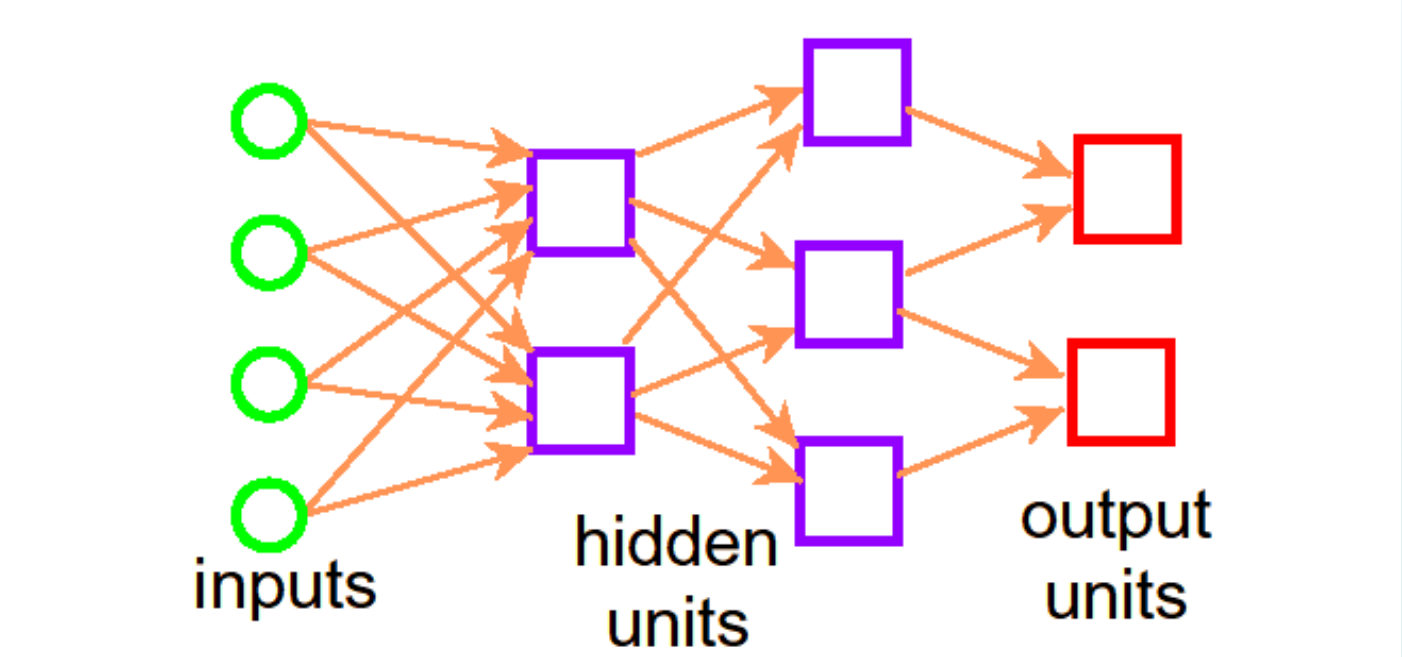

snns-manager
FILE
CONTROL
INFO
DISPLAY
3D DISPLAY
GRAPH
BIGNET
PRUNING
CASCADE
KOHONEN
WEIGHTS
PROJECTION
ANALYZER
INVERSION
PRINT
HELP
RPC
QUIT
x:15
y:8
banner
SNNS
V4.0
3D-display
RPCSETUP
connection type:
UDP
TCP
kernel type:
Standart timeout sec :
Long duration timeout sec:
timeout for autoexit (min):
switch to local (ALL) :
switch to local (SINGLE) :
DONE
APPLY
DEFAULT
snns-display 1 - subnet: 0
SETUP
FREEZE
RPCSELECTITEMLIST
Select Info to show in the RPCPANEL
INFO
SELECTED
DOMAINNAME
HOSTADRESS
KERNELID
K_TYPE
CYCLES
EPOCHE
MSE
SSE/o-units
NO. PATTERN
HOSTNAME
KERNELNO
STATUS
SSE
SNNS RPC PANEL
CURRENT HOST:
direct local 0
No. HOSTNAME KERNELNO STATUS SSE
0 direct local 0 idle 0.52159
1 vasarely 400020139 idle 0.00000
2 mondrian 400020139 idle 0.00000
3 matisse 400020139 idle 0.00000
matisse
de -program 572352910 -version 40 -uid 20139 < /dev/null > /dev/null 2>&1
New Kernel on matisse ,PID 2122 added

# Usos de Redes Neuronales

- Reconocimiento de Patrones.
- Clasificación.
- Acceso por contenido a memoria.
- Predicción.
- Optimización.
- Filtrado de ruido.

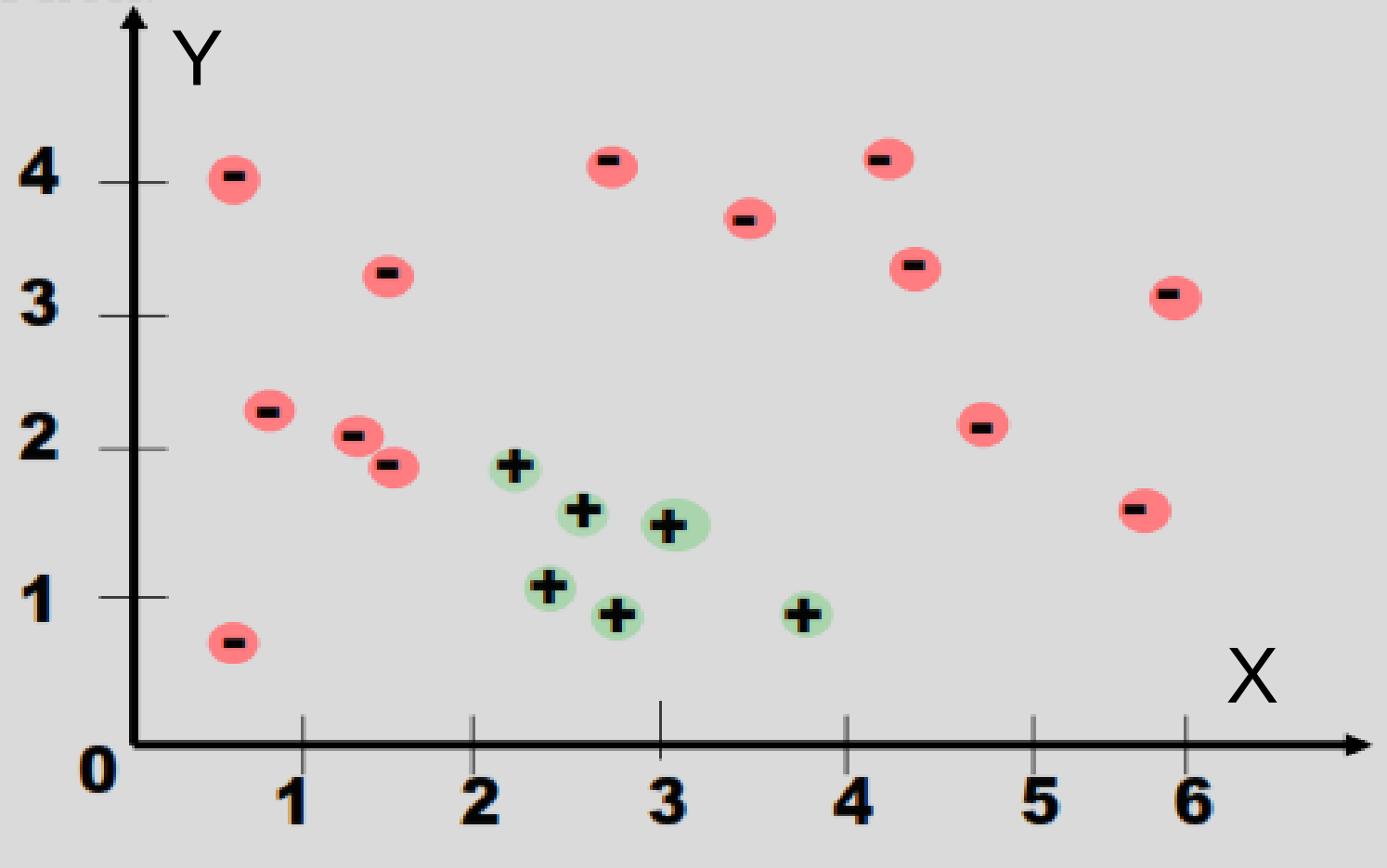
# Alimentación hacia adelante

- Redes neuronales con alimentación hacia adelante (feed-forward) representan el modelo más simple.
- Son grafos dirigidos acíclicos:



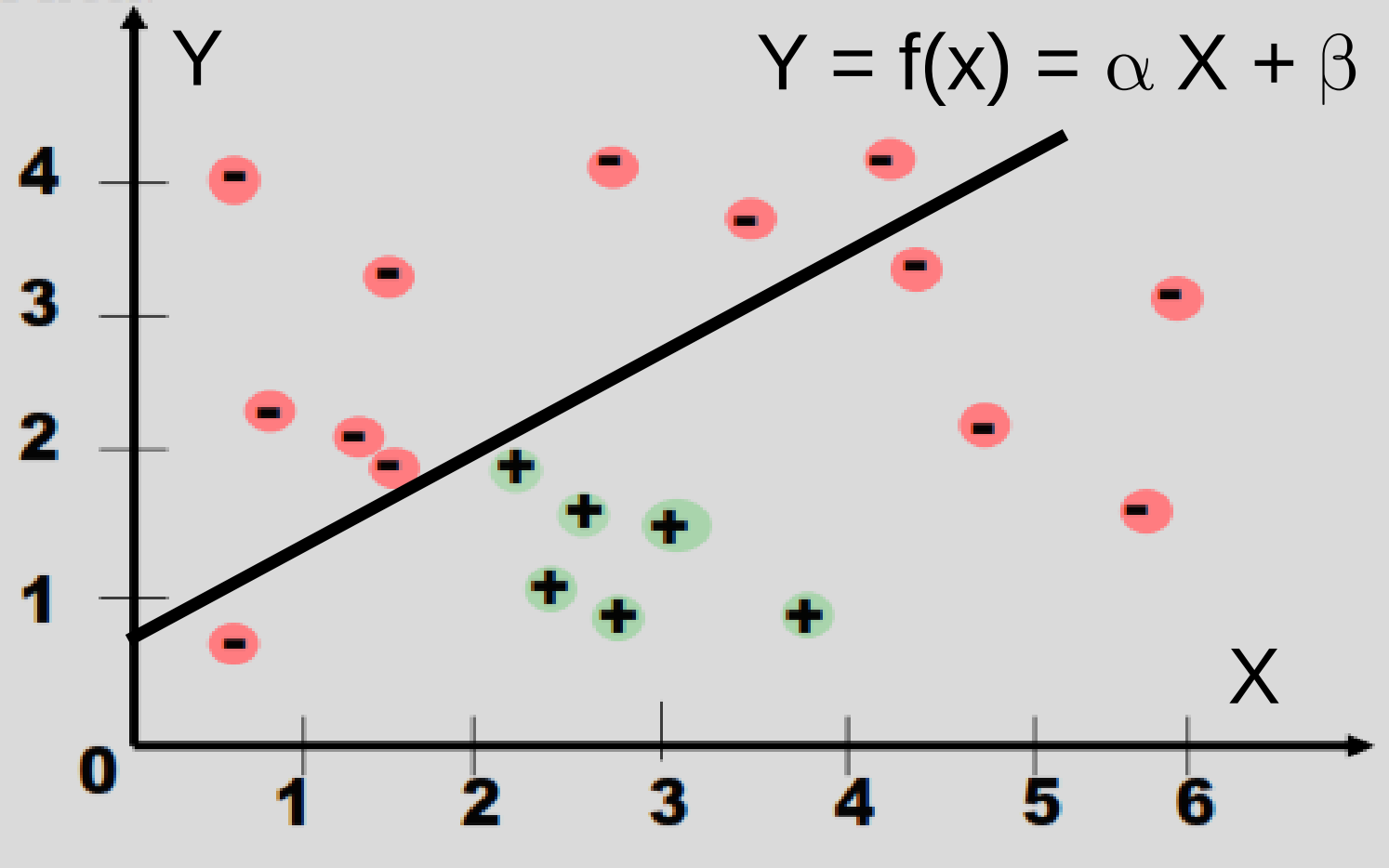

Computational Intelligence D. Poole, A. Mackworth, R. Goebel

# Ajustando Funciones

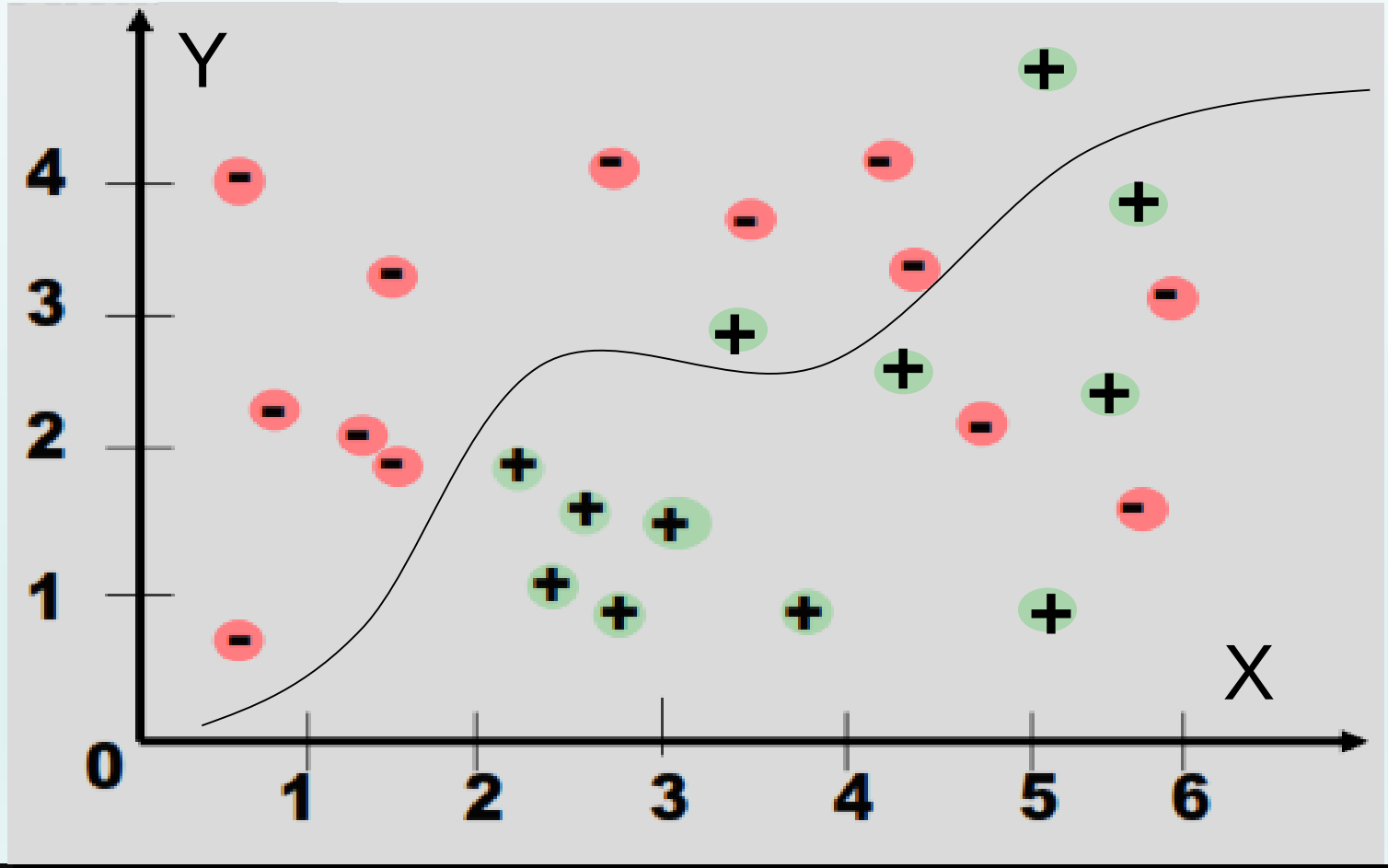


# Aprendiendo Funciones

**Como aprender un concepto: Regresión Lineal**

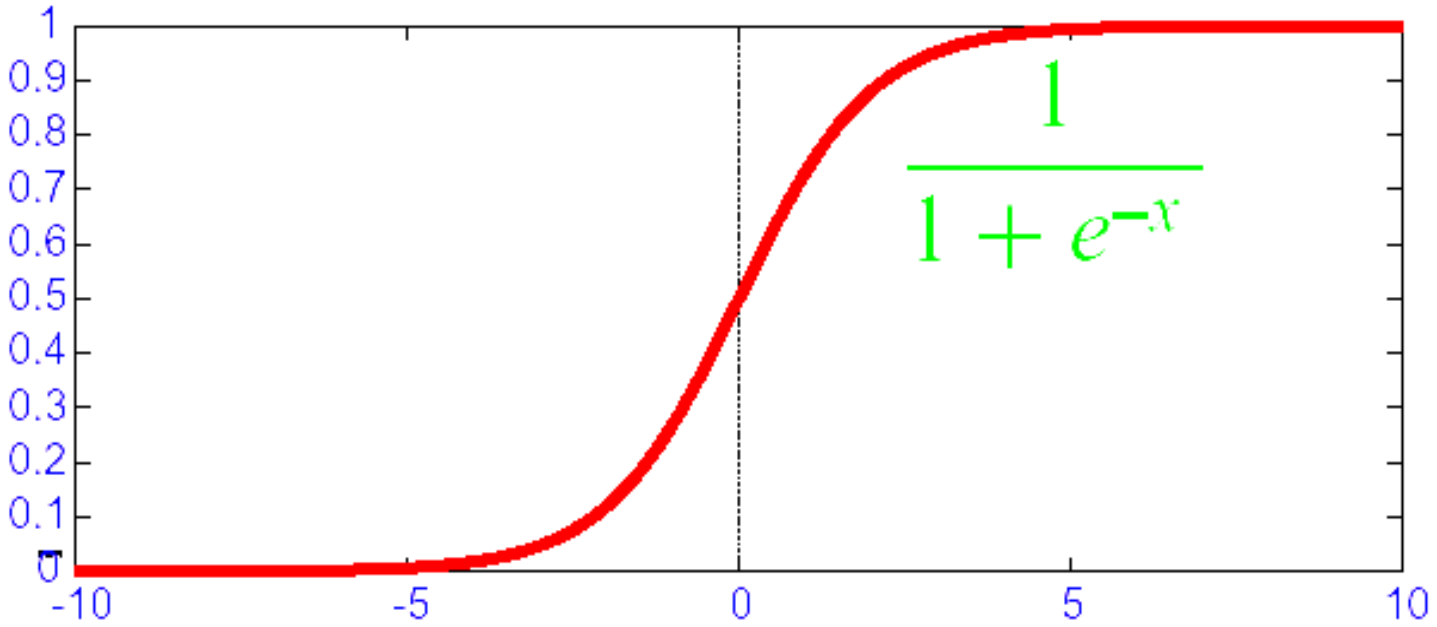


# Reconociendo Hongos



# Función de Activación

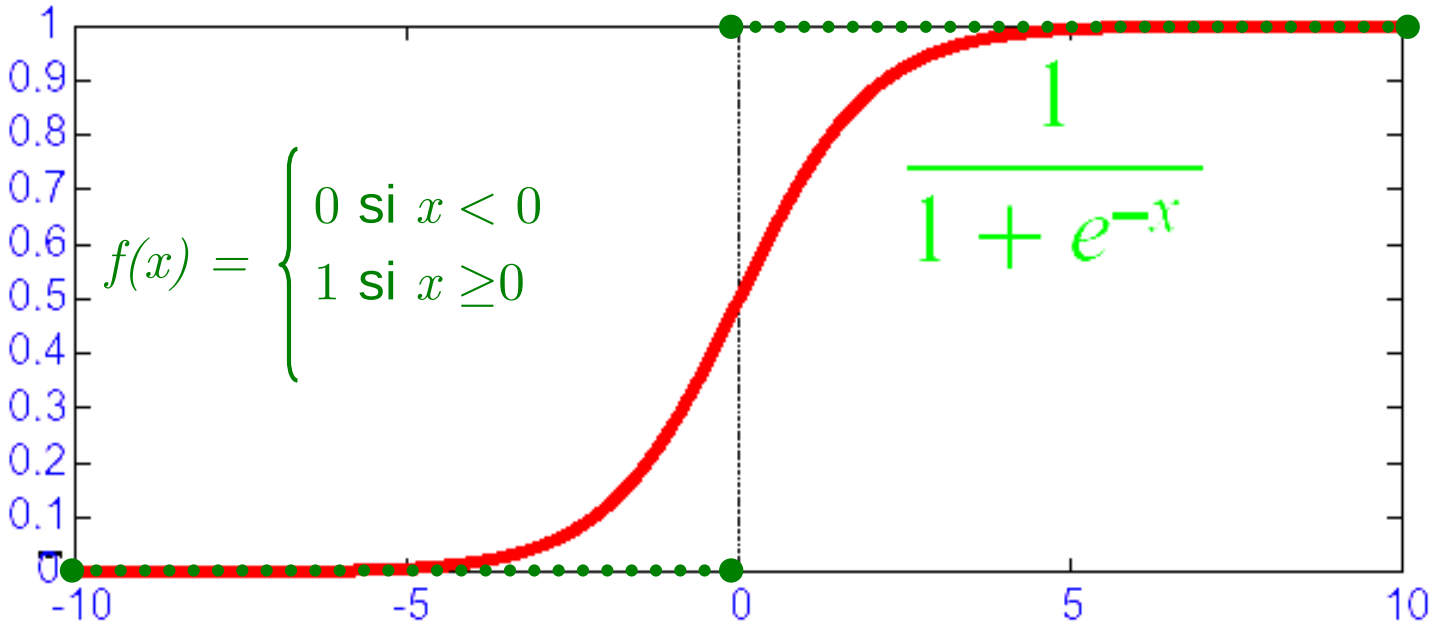
Una función típica de activación es la *función sigmoidal*:




$$f(x) = \frac{1}{1+e^{-x}} \qquad f'(x) = f(x)(1-f(x))$$

Computational Intelligence D. Poole, A. Mackworth, R. Goebel

# Función de Activación

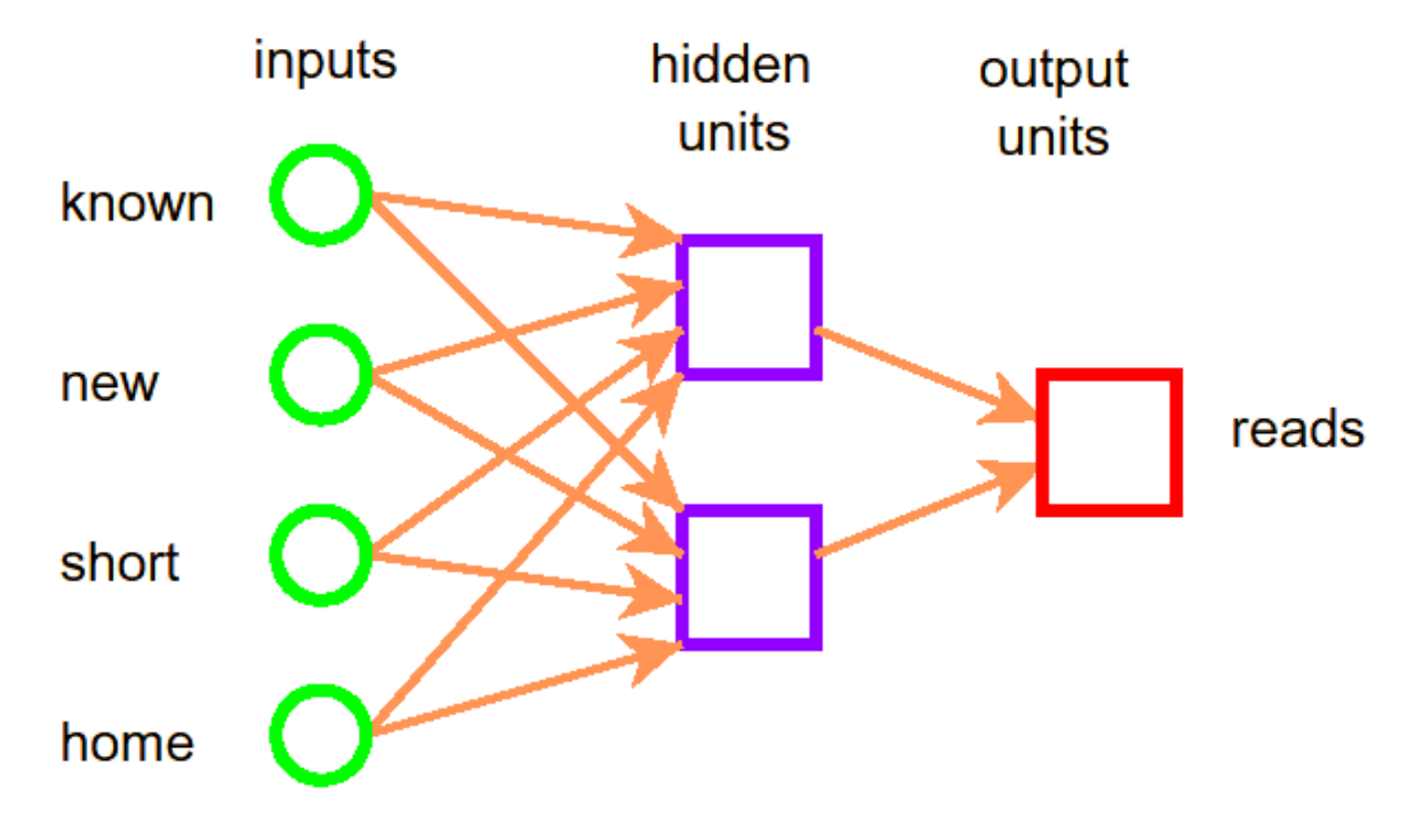



Función escalera superpuesta a la sigmoidal

$$f(x) = \frac{1}{1+e^{-x}} \qquad f'(x) = f(x)(1-f(x))$$

# Ejemplo

Una red neuronal posible para el ejemplo ya visto del agente que lee *news:*




*Computational Intelligence* D. Poole, A. Mackworth, R. Goebel

# Axiomatización

- Los valores de los atributos son números reales.
- Los trece parámetros $w_0, \ldots, w_{12}$ son números reales.
- Deben aprenderse 13 números reales, *i.e.* el espacio de hipótesis es $\mathbb{R}^{13}$, un espacio real de 13 dimensiones.
- Cada punto en este espacio corresponde a un programa lógico particular que predice el valor para el atributo *reads* dados los atributos $known$, $new$, $short$ y $home$.

# Error de Predicción

- Para errores particulares de los parámetros $w = w_0, w_1, \ldots, w_m$ y un conjunto $E$ de ejemplos, el *error cuadrático* es:

$$Error_E(w) = \sum_{e \in E} (p_e^{\overline{w}} - o_e)^2$$

  - $p_e$ es la *salida predecida* por la red neuronal que recibe los parámetros dados por $w$ para el ejemplo $e$
  - $o_e$ es la *salida (output) observada* (conjunto de test) para el ejemplo $e$

# Aprendizaje

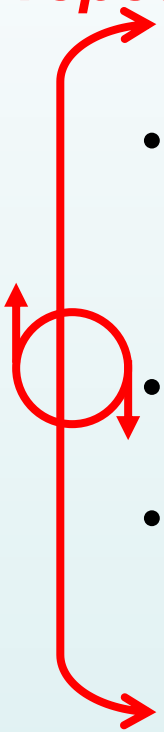
- El objetivo del aprendizaje de una red neuronal puede ser descripto como un problema de minimización: *dado un conjunto de ejemplos encontrar un conjunto de valores de los parámetros que minimize el error cuadrático.*
- El aprendizaje por *back-propagation* es una búsqueda siguiendo la disminución del gradiente a través del espacio de parámetros para minimizar el error cuadrático.

# Back-propagation

- Entradas:
  - Una red (incluyendo todas las unidades y sus conexiones).
  - Un criterio de parada.
  - Un factor de aprendizaje (una constante de proporcionalidad para la búsqueda por descenso del gradiente).
  - Valores iniciales para los parámetros.
  - Un conjunto de datos de entrenamiento clasificados.
- Salida: nuevos valores para los parámetros.

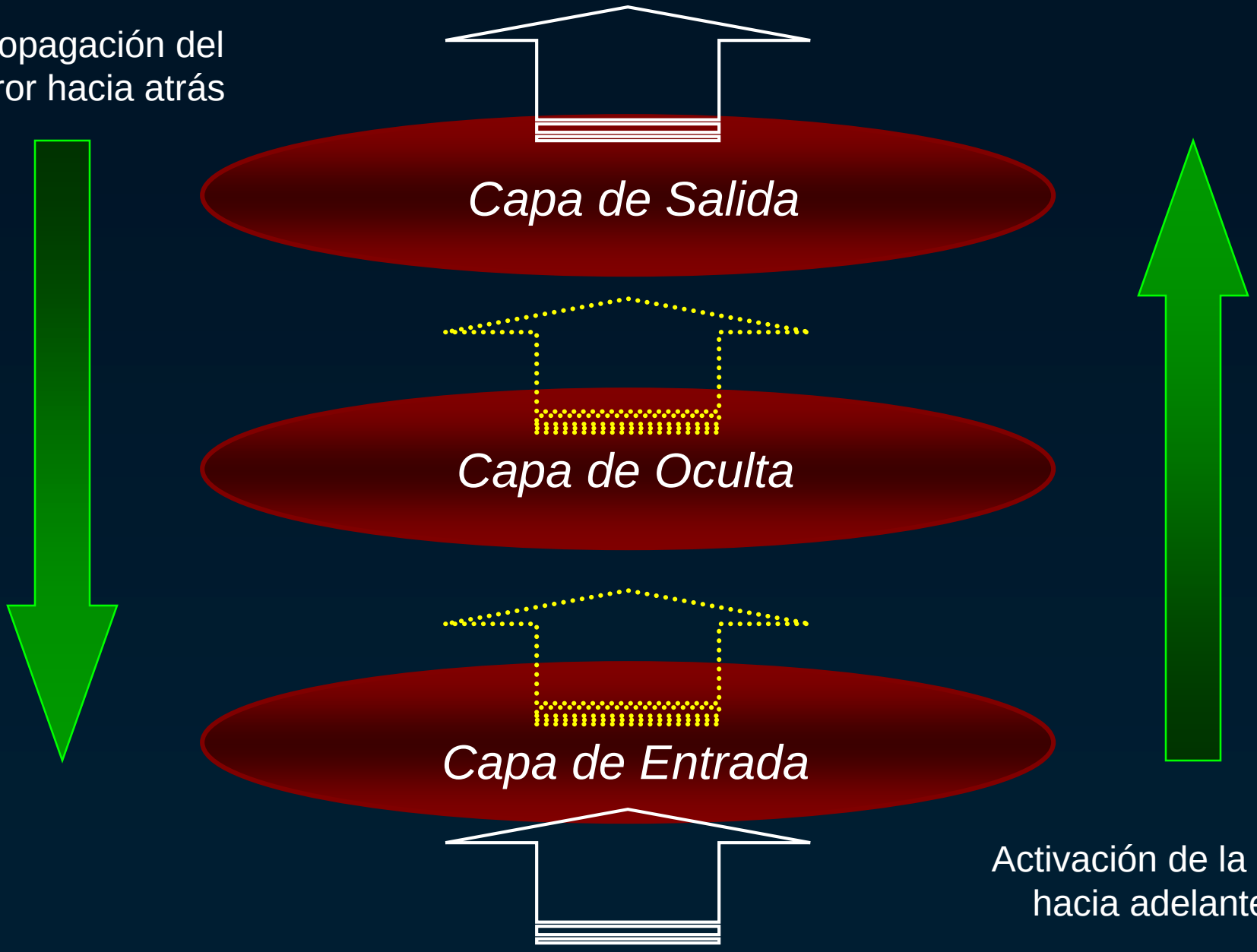
# Algoritmo

***repetir***



- Evaluar la red en cada ejemplo dados los valores de los parámetros.
- Determinar la derivada del error por cada parámetro.
- Actualizar cada parámetro en proporción a su derivada.

***hasta*** que se satisfaga el criterio de parada.


Propagación del error hacia atrás
Capa de Salida
Capa de Oculta
Capa de Entrada
Activación de la red hacia adelante

# Métodos No supervisados

- Los datos de entrada *no vienen* asociados con resultados, los algoritmos deben encontrar *por sí solos* la estructura subyacente en los datos de entrada.
- Clustering es uno método de aprendizaje no supervisados.
- Un buen método de *clustering* produce *clusters* de alta calidad con:
  - Alta similitud en la clase
  - Baja similitud entre clases
- La calidad de un *clustering* depende de la medida de "similitud" usada por el método y de la forma en que está implementado.

# Medición de la calidad de un cluster

- Medida de similitud: La similitud está expresada en base a una función de distancia.
- Hay una función separada que mide la *bondad* del clustering.
- Las funciones de distancia a utilizar son muy diferentes de acuerdo al tipo de dato.
- Algunas veces es necesario asignarle “peso” a las variables dependiendo del significado que tienen para el problema.
- Las distancias se usan habitualmente para medir la similitud entre dos objetos.

# Distancias

$$d_{ij} = \sum_{k=1}^{p} W_k \left| x_{ik} - x_{jk} \right|$$

City-Block (Manhatan)

$$d_{ij} = \sqrt{\sum_{k=1}^{p} W_k \left( x_{ik} - x_{jk} \right)^2}$$

Euclídea

$$d_{ij} = \sqrt[\lambda]{\sum_{k=1}^{p} W_k \left( x_{ik} - x_{jk} \right)^{\lambda}} \quad \lambda > 0$$

Minkowski

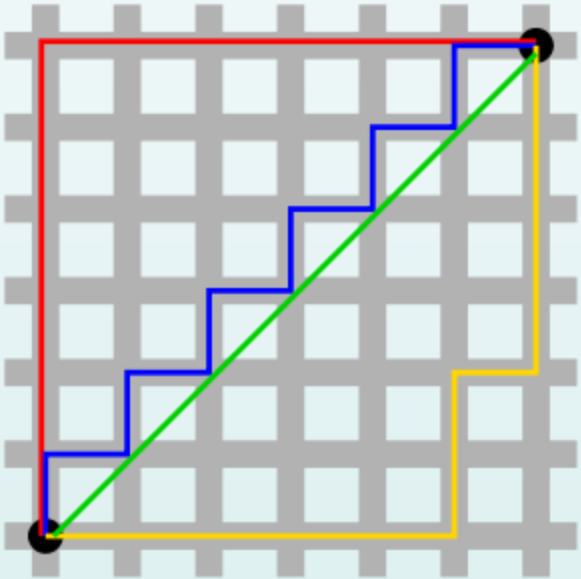
Otras

$$d_{ij} = \frac{\sum_{k=1}^{p} x_{ik} \cdot x_{jk}}{\sqrt{\sum_{k=1}^{p} x_{ik}^2} \cdot \sqrt{\sum_{l=1}^{p} x_{jl}^2}}$$

$$d_{ij} = \frac{\sum_{k=1}^{p} \left( x_{ik} - \bar{x}_i \right) \cdot \left( x_{jk} - \bar{x}_j \right)}{\sqrt{\sum_{k=1}^{p} \left( x_{ik} - \bar{x}_i \right)^2} \cdot \sqrt{\sum_{l=1}^{p} \left( x_{jl} - \bar{x}_j \right)^2}}$$

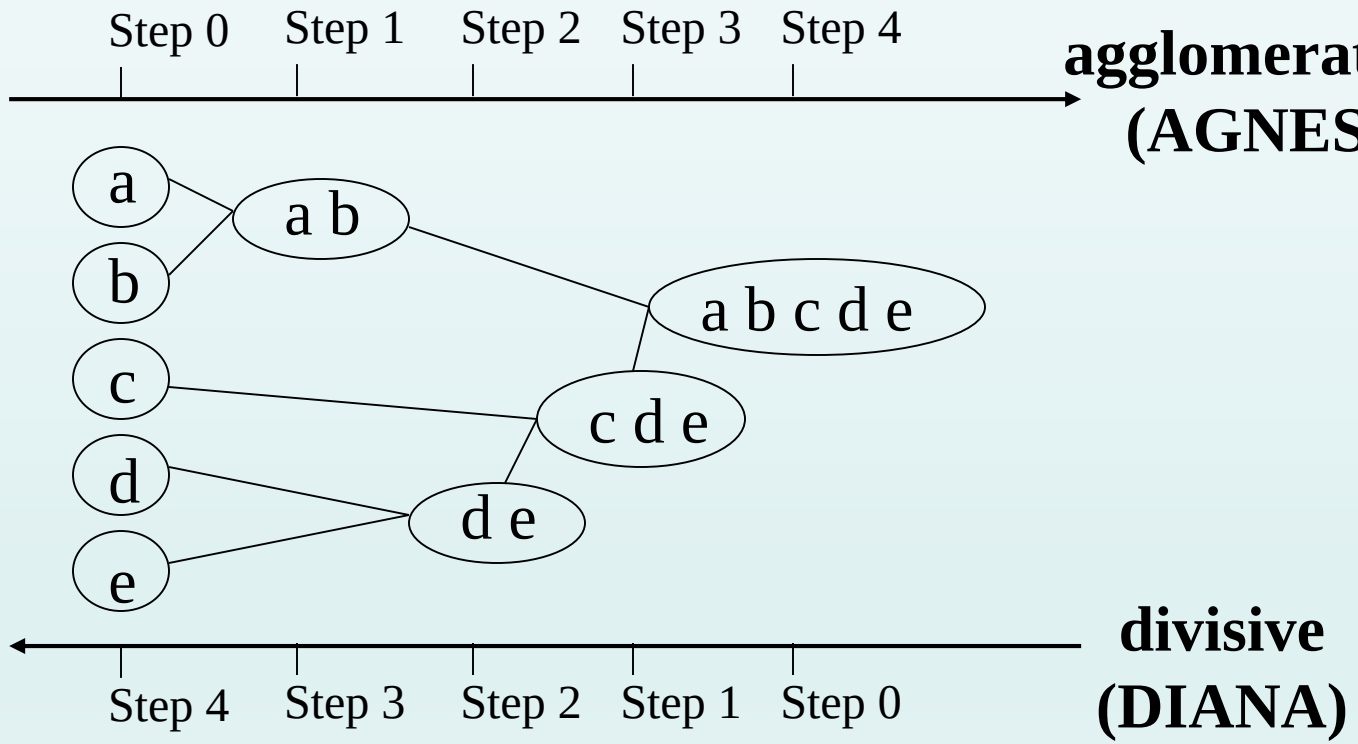
# Manhattan versus Euclidean

El rojo, azul, y amarillo representan la distancia Manhattan, todas tienen el mismo largo(12),mientras que la verde representa la distancia Euclidia con largo de $6 \times \sqrt{2} \approx 8.48$.



# Clustering Jerárquico

- Usa la matriz de distancia como criterio. No requiere que el número de cluster sea uno de los parámetros de input



# Agrupamiento aglomerativo

**Criterio de enlace:** determina qué distancia utilizar entre conjuntos de observación. El algoritmo fusionará los pares de clústeres que minimicen este criterio.

- Ward minimiza la variación de los grupos que se fusionan.
- Promedio utiliza el promedio de las distancias de cada observación de los dos conjuntos.
- El enlace completo utiliza las distancias máximas entre todas las observaciones de los dos conjuntos.
- El enlace simple utiliza el mínimo de las distancias entre todas las observaciones de los dos conjuntos.

# No Jerárquicas: algoritmo básico

Método de particionamiento: Construir una partición de la base de datos D de *n* objetos en *k* clusters

- Dado *k* encontrar una partición de *k* clusters que optimice el criterio de partición usado:
  - Optimo Global: enumerar todas las particiones posibles
  - Métodos heurísticos:
    - *k-means* (MacQueen'67): cada cluster esta representado por el centro del cluster.
    - *k-medoids* or PAM (Partition around medoids) (Kaufman & Rousseeuw'87): cada cluster está representado por uno de los objetos del cluster.

# Métodos jerarquicos vs no jerarquicos

**Agrupamiento jerarquico**

- No hay decisión acerca del número de clusters
- Existen problemas cuando los datos contienen un alto nivel de error
- Puede ser muy lento

**Agrupamiento no jerarquico**

- Es necesario especificar el numero de clusters (arbitrario)
- Es necesario establecer la semilla inicial
- Más rapido y más fiable

# Referencias


- Computational Intelligence: A Logical Approach - D.Poole, A. Mackworth, R. Goebel Oxford University Press. Capitulo 11.
- [Popejoy2016] Popejoy, A. B., & Fullerton, S. M. (2016). Genomics is failing on diversity. *Nature*, *538*(7624), 161–164. doi:10.1038/538161a
- Han, Jiawei., Micheline Kamber, and Jian Pei. Data Mining: Concepts and Techniques, Third Edition. 3rd ed. Waltham, Mass.: Morgan Kaufmann Publishers, 2012.