# Tema 11: Aprendizaje automático

**José A. Alonso Jiménez**
**Miguel A. Gutiérrez Naranjo**
**Francisco J. Martín Mateos**

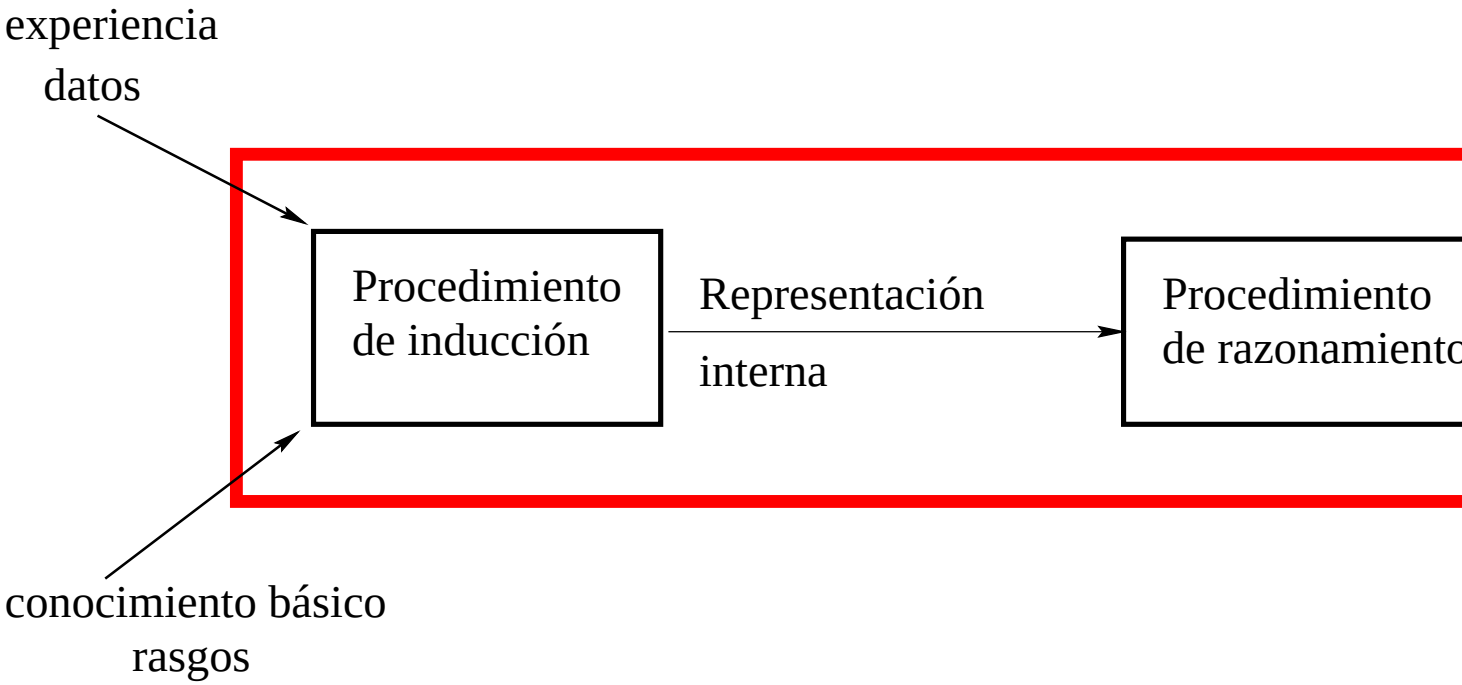
**Dpto. de Ciencias de la Computación e Inteligencia Artificial**

Universidad de Sevilla

# Aprendizaje automático

- **El aprendizaje automático estudia cómo construir programas que mejoren automáticamente con la experiencia**
- **Aspectos a mejorar**
  - **La amplitud: hacer más tareas**
  - **La calidad: hacer mejor**
  - **La eficiencia: hacer más rápido**
- **Razones de interés**
  - **Recientes avances en la teoría y los algoritmos y máquinas.**
  - **Crecimiento desbordante de datos "en línea" (on line).**
  - **Interés económico**

# Arquitectura de aprendizaje automático



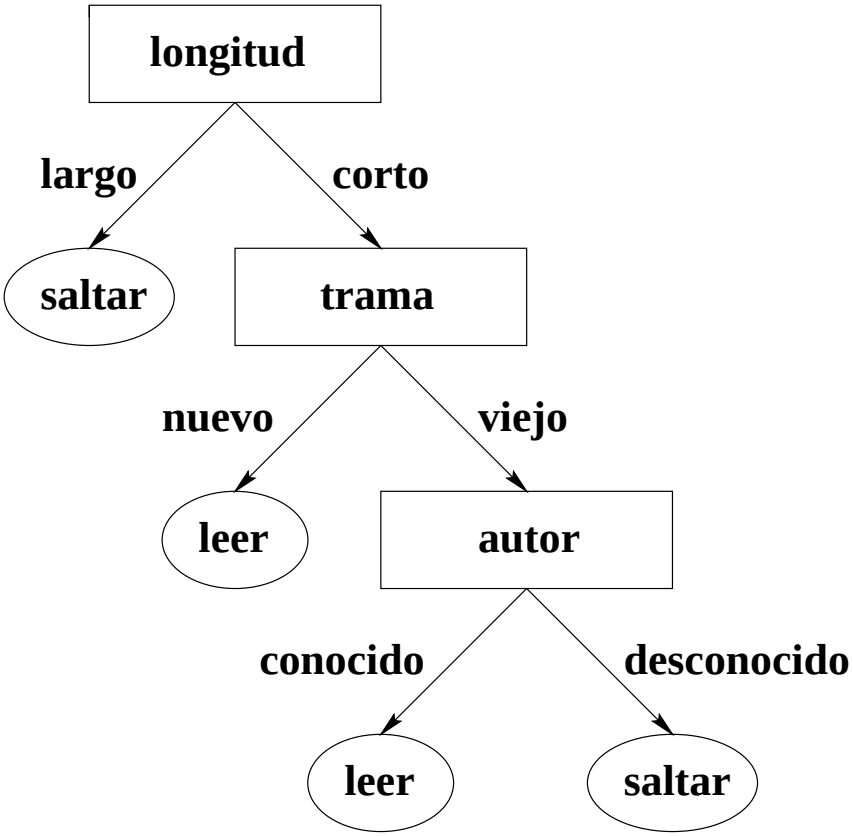
# Aspectos en los problemas de aprendizaje

- *Tarea:* Comportamiento o tarea a mejorar
  - Clasificación supervisada
  - Aprendizaje no supervisado
  - Aprendizaje analítico
- *Datos:* Experiencia usada para mejorar
- *Representación del conocimiento*
  - Potencia expresiva vs. complejidad de aprendizaje
- *Interacción*
- *Sesgos*

# Ejemplo de datos

| Ejemplo | Acción | Autor | Tema | Longitud | Sitio |
|---|---|---|---|---|---|
| e1 | saltar | conocido | nuevo | largo | casa |
| e2 | leer | desconocido | nuevo | corto | trabajo |
| e3 | saltar | desconocido | viejo | largo | trabajo |
| e4 | saltar | conocido | viejo | largo | casa |
| e5 | leer | conocido | nuevo | corto | casa |
| e6 | saltar | conocido | viejo | largo | trabajo |
| e7 | saltar | desconocido | viejo | corto | trabajo |
| e8 | leer | desconocido | nuevo | corto | trabajo |
| e9 | saltar | conocido | viejo | largo | casa |
| e10 | saltar | conocido | nuevo | largo | trabajo |
| e11 | saltar | desconocido | viejo | corto | casa |
| e12 | saltar | conocido | nuevo | largo | trabajo |
| e13 | leer | conocido | viejo | corto | casa |
| e14 | leer | conocido | nuevo | corto | trabajo |
| e15 | leer | conocido | nuevo | corto | casa |
| e16 | leer | conocido | viejo | corto | trabajo |
| e17 | leer | conocido | nuevo | corto | casa |
| e18 | leer | desconocido | nuevo | corto | trabajo |

# Arbol de decisión



# Descripción de árboles de decisión

- **Elementos del árbol de decisión:**
  - **Nodos distintos de las hojas: atributos**
  - **Arcos: posibles valores del atributo**
  - **Hojas: clasificaciones**
- **Reglas correspondientes a un árbol de decisión**

```
longitud(E)=largo -> accion(E)=saltar
longitud(E)=corto y trama(E)=nuevo -> accion(E)=leer
longitud(E)=corto y trama(E)=viejo y autor(E)=conocido -> ac
longitud(E)=corto y trama(E)=viejo y autor(E)=desconocido ->
```

- **Fórmula correspondiente a un árbol de decisión**

# Cuestiones sobre árboles de decisión

- ¿Cuál es el árbol de decisión correcto?
  - Navaja de Occam
  - El mundo es inherentemente simple
  - El árbol de decisión más pequeño consistente con la muestra es el que tiene más probabilidades de identificar objetos desconocidos de manera correcta
  - Menor profundidad
  - Menor número de nodos
- ¿Cómo se puede construir el árbol de decisión más pequeño?
  - Búsqueda en escalada

# Búsqueda del árbol de decisión

- **¿Están todos los ejemplos en la misma clase?**

```
Ejemplos positivos = {E: accion(E)=leer} =
                       = {e2, e5, e8, e13, e14, e15, e16, e1
Ejemplos negativos = {E: accion(E)=saltar} =
                       = {e1, e3, e4, e6,  e7,  e9,  e10, e1
P = número de ejemplos positivos = 9
N = número de ejemplos negativos = 9
T = número total de ejemplos = P + N = 18
```

- **Información**
  - **Fórmula:** $\boldsymbol{I(P,N) = \begin{cases} 0, & \text{si } N*P=0; \\ -\frac{P}{T}\log_2\frac{P}{T} - \frac{N}{T}\log_2\frac{N}{T}, & \text{si } N*P\neq 0. \end{cases}}$
  - **Ejemplo:** $\boldsymbol{I(9,9) = -\frac{9}{18}\log_2\frac{9}{18} - \frac{9}{18}\log_2\frac{9}{18} = 1}$

# Búsqueda del árbol de decisión

- **Información tras la división por un `Atributo`:** $I = \frac{N1*I1+N2*I2}{N1+N2}$, **donde**
  - **N1 = número de ejemplos en la clase 1**
  - **N2 = número de ejemplos en la clase 2**
  - **I1 = cantidad de información en los ejemplos de la clase 1**
  - **I2 = cantidad de información en los ejemplos de la clase 2**

# Búsqueda del árbol de decisión

- **Ganancia de información al dividir por longitud**
  - **Distribución**

| | leer | saltar |
|---|---|---|
| largo | | 01,03,04,06,09,10, |
| corto | 02,05,08,13,14,15,16,17,18 | 07,11 |

  - **Información de** `longitud(E)=largo`: $I(0,7) = 0$
  - **Información de** `longitud(E)=corto` $I(9,2) = -\frac{9}{11}\log_2\frac{9}{11} - \frac{2}{11}\log_2\frac{2}{11} = 0.684$
  - **Información de la división por** `longitud`: $\frac{7*0+11*0.684}{18} = 0.418$

# Búsqueda del árbol de decisión

- **Comparación de ganancias de información:**
  -

| Atributo | Información |
|---|---|
| autor | 1 |
| longitud | 0.418 |
| tema | 0.850 |
| sitio | 1 |

- **Conclusiones**
  - **Mejor atributo para dividir: longitud**
  - Clase_1 = {E: longitud(E)=largo} = {} U {1,3,4,6,9,10,12}
  - Clase_2 = {E: longitud(E)=corto} = {2,5,8,13,14,15,16,17,1 U {7,11}

# Búsqueda del árbol de decisión

- **Clasificación de `Clase_1`**
  - **Están todos los ejemplos en la misma clase: `accion(E)=saltar`**
- **Clasificación de `Clase_2`**
  - **No están todos los ejemplos en la misma clase**
  - **Repetir sobre `Clase_2` con los restantes atributos (autor, tema y sitio)**
  - **Información de `longitud(E)=corto`**
    $I(9,2) = -\frac{9}{11}\log_2\frac{9}{11} - \frac{2}{11}\log_2\frac{2}{11} = 0.684$

# Búsqueda del árbol de decisión

- **Ganancia de información al dividir `Clase_2` por tema**
  - **Distribución**

| | leer | saltar |
|---|---|---|
| nuevo | 02,05,08,14,15,17,18 | |
| viejo | 13,16 | 07,11 |

  - **Información de `tema(E)=nuevo`** $\boldsymbol{I(7,0) = 0}$
  - **Información de `tema(E)=viejo`** $\boldsymbol{I(2,2) = -\frac{2}{4}\log_2\frac{2}{4} - \frac{2}{4}\log_2\frac{2}{4} =}$
  - **Información de la división por `tema`:** $\boldsymbol{\frac{7*0+4*1}{11} = 0.364}$

# Búsqueda del árbol de decisión

- **Ganancias de información al dividir** `Clase_2`
- 

| Atributo | Información |
|---|---|
| autor | 0.441 |
| tema | 0.364 |
| sitio | 0.683 |

- **Conclusiones**
  - **Mejor atributo para dividir** `Clase_2`**: tema**
  - `Clase_3 = {E: longitud(E)=corto, tema(E)=nuevo} = {2,5,8,14,15,17,18} U {}`
  - `Clase_4 = {E: longitud(E)=corto, tema(E)=viejo} = {13,16} U {7,11}`

# Búsqueda del árbol de decisión

- **Clasificación de `Clase_3`**
  - **Están todos los ejemplos en la misma clase: `accion(E)=leer`**
- **Clasificación de `Clase_4`**
  - **No están todos los ejemplos en la misma clase**
  - **Repetir sobre `Clase_4` con los restantes atributos (autor y sitio)**
  - **Información de `longitud(E)=corto, tema(E)=viejo`**
    $I(2,2) = -\frac{2}{4}\log_2\frac{2}{4} - \frac{2}{4}\log_2\frac{2}{4} = 1$

# Búsqueda del árbol de decisión

- **Ganancia de información al dividir `Clase_4` por autor**
  - **Distribución**

| | leer | saltar |
|---|---|---|
| conocido | 13,16 | |
| desconocido | | 07,11 |

  - **Información de `autor(E)=conocido` $I(2,0) = 0$**
  - **Información de `autor(E)=desconocido` $I(0,2) = 0$**
  - **Información de la división por `autor`:** $\frac{2*0+2*0}{4} = 0$

# Búsqueda del árbol de decisión

- **Ganancias de información al dividir** `Clase_4`
  -

| Atributo | Información |
|---|---|
| autor | 0 |
| sitio | 1 |

- **Conclusiones**
  - **Mejor atributo para dividir** `Clase_4`**: autor**
  - `Clase_5 = {E: longitud(E)=corto, tema(E)=nuevo, autor(E)= conocido}`
    `= {13,16} U {}`
  - `Clase_6 = {E: longitud(E)=corto, tema(E)=viejo, autor(E)= desconocido}`
    `= {} U {7,11}`

# Búsqueda del árbol de decisión

- **Clasificación de `Clase_5`**
  - **Están todos los ejemplos en la misma clase:** `accion(E)=leer`
- **Clasificación de `Clase_6`**
  - **Están todos los ejemplos en la misma clase:** `accion(E)=saltar`

# Algoritmo ID3

- **Sesión**

```
?- ['aprende_ad.pl','aprende_ad_e1.pl'].
Yes

?- aprende_ad(accion,
                  [e1,e2,e3,e4,e5,e6,e7,e8,e9,e10,e11,e12,e1
                  [autor,tema,longitud,sitio],
                  AD).
AD = si(longitud=largo, saltar,
            si(tema=nuevo, leer,
               si(autor=desconocido, saltar,
                  leer)))
Yes
```

# Algoritmo ID3

- **Representación del problema** `aprende_ad_e1.pl`
  - `val(Objeto,Atributo,Valor)` **se verifica si el valor del** `Atributo` **del** `Objeto` **es** `Valor`

```
val(e1,accion,saltar).
val(e1,autor,conocido).
val(e1,tema,nuevo).
val(e1,longitud,largo).
val(e1,sitio,casa ).

.........................

val(e18,accion,leer).
val(e18,autor,desconocido).
val(e18,tema,nuevo).
val(e18,longitud,corto).
val(e18,sitio,trabajo).
```

# Algoritmo ID3

- **Algoritmo de aprendizaje de árboles de decisión**
  - `aprende_ad(+Objetivo,+Ejemplos,+Atributos,-AD)` **se verifica si** `AD` **es el árbol de decisión inducido para el** `Objetivo` **a partir de la lista de** `Ejemplos` **y** `Atributos`

```
aprende_ad(Objetivo, Ejemplos, _ , Val) :-
   coinciden_todos_ejemplos(Objetivo, Ejemplos, Val).
aprende_ad(Objetivo, Ejemplos, Atributos, si(Atr=ValPos,AP
   not(coinciden_todos_ejemplos(Objetivo, Ejemplos, _)),
   selecciona_division(Objetivo, Ejemplos, Atributos, Atr,
   divide(Ejemplos, Atr, ValPos, Positivos, Negativos),
   aprende_ad(Objetivo, Positivos, Resto_Atr, APos),
   aprende_ad(Objetivo, Negativos, Resto_Atr, ANeg).
```

# Algoritmo ID3

- **¿Están todos los ejemplos en la misma clase?**
  - `coindiden_todos_ejemplos(+Objetivo,+Ejemplos,-Valor)` **se verifica si el valor del atributo** `Objetivo` **de todos los** `Ejemplos` **es Valor**

```
coinciden_todos_ejemplos(_,[],_).
coinciden_todos_ejemplos(Atributo,[Obj|Resto],Val) :-
   val(Obj,Atributo,Val),
   coinciden_todos_ejemplos(Atributo,Resto,Val).
```

# Algoritmo ID3

- **Selección del mejor atributo para dividir**
  - `selecciona_division(+Objetivo,+Ejemplos,+Atributos, -Atributo,-Restantes_atributos)` **se verifica si** `Atributo` **es el mejor elemento de la lista de** `Atributos` **para determinar el** `Objetivo` **a partir de los** `Ejemplos` **(es decir, la información resultante del** `Objetivo` **en los** `Ejemplos` **usando como división el** `Atributo` **es mínima), y** `Restantes_atributos` **es la lista de los restantes** `Atributos`**. Falla si para ningún atributo se gana en información.**

```
selecciona_division(Objetivo, Ejemplos, [A|R], Atributo, R
    informacion_division(Objetivo,Ejemplos,A,I),
    selecciona_max_ganancia_informacion(Objetivo,Ejemplos,
                                        Atributo,[],Resto_
```

# Algoritmo ID3

- `informacion_division(+Objetivo,+Ejemplos,+Atributo,-I)` **se verifica si** `I` **es la información resultante del** `Objetivo` **en los** `Ejemplos` **usando como división el** `Atributo`**; es decir,** `I = (N1*I1 + N2*I2) / (N1+N2)`

```
informacion_division(Objetivo,Ejemplos,Atributo,Inf) :-
   divide(Ejemplos,Atributo,_,Clase_1,Clase_2),
   informacion(Objetivo,Clase_1,I1),
   informacion(Objetivo,Clase_2,I2),
   length(Clase_1,N1),
   length(Clase_2,N2),
   Inf is (N1*I1 + N2*I2)/(N1+N2).
```

# Algoritmo ID3

- `informacion(+Objetivo,+Ejemplos,-I)` **se verifica si** `I` **es la cantidad de información en los** `Ejemplos` **respecto del** `Objetivo`**; es decir,**

$$\boldsymbol{I} = \begin{cases} \mathbf{0}, & \textbf{si } \boldsymbol{N} * \boldsymbol{P} = \mathbf{0}; \\ -\frac{\boldsymbol{P}}{\boldsymbol{T}} \log_2 \frac{\boldsymbol{P}}{\boldsymbol{T}} - \frac{\boldsymbol{N}}{\boldsymbol{T}} \log_2 \frac{\boldsymbol{N}}{\boldsymbol{T}}, & \textbf{si } \boldsymbol{N} * \boldsymbol{P} \neq \mathbf{0}. \end{cases}$$

```
informacion(Objetivo,Ejemplos,I) :-
   cuenta(Objetivo,_,Ejemplos,NP,NN),
   ( (NP=0 ; NN=0) -> I=0
     ;
     NT is NP + NN,
     I is - NP/NT * log2(NP/NT) - NN/NT * log2(NN/NT)).
```

# Algoritmo ID3

- `cuenta(+Atributo,?VP,+Ejemplos,-NP,-NN)` **se verifica si** `NP` **es el número de ejemplos positivos (es decir, elementos de** `Ejemplos` **tales que el valor de su** `Atributo` **es** `VP` **(valor positivo)) y** `NN` **es el número de ejemplos negativos**

```
cuenta(_,_,[],0,0).
cuenta(Atributo,VP,[E|R],NP,NN) :-
   val(E,Atributo,VP),
   cuenta(Atributo,VP,R,NP1,NN),
   NP is NP1+1.
cuenta(Atributo,VP,[E|R],NP,NN) :-
   val(E,Atributo,VNeg),
   not(VP=VNeg),
   cuenta(Atributo,VP,R,NP,NN1),
   NN is NN1+1.
```

# Algoritmo ID3

- `selecciona_max_ganancia_informacion(+Objetivo, +Ejemplos, +Atributos, +Mejor_atributo_actual, +Mejor_info_actual, -Atributo, +Atributos_analizados, -Resto_atributos)` **se verifica si** `Atributo` **es el elemento de** `Atributos` **tal la información resultante del** `Objetivo` **en los** `Ejemplos` **usando como división el** `Atributo` **es mínima**

```
selecciona_max_ganancia_informacion(_,_,[],MejorA,_,
                                    MejorA, A_analizados,
selecciona_max_ganancia_informacion(Objetivo, Ejs, [A|R],
                                    Atributo, A_analizados
   informacion_division(Objetivo,Ejs,A,Informacion),
   ( Informacion > MejorI
     -> selecciona_max_ganancia_informacion(
         Objetivo,Ejs,R,MejorA,MejorI,Atributo,[A|A_analiz
     ;  selecciona_max_ganancia_informacion(
         Objetivo,Ejs,R,A,Informacion,Atributo,[MejorA|A_a
   ).
```

# Algoritmo ID3

- **División de los ejemplos**
  - `divide(+Ejemplos,+Atributo,?Valor,-Positivos,-Negativos)` **se verifica si** `Positivos` **es la lista de elementos de** `Ejemplos` **tales que el valor de** `Atributo` **es** `Valor` **y** `Negativos` **es la lista de los restantes** `Ejemplos`

```
divide([],_,_,[],[]).
divide([Ej|Rest],Atributo,ValPos,[Ej|Positivos],Negativos)
   val(Ej,Atributo,ValPos),
   divide(Rest,Atributo,ValPos,Positivos,Negativos).
divide([Ej|Rest],Atributo,ValPos,Positivos,[Ej|Negativos])
   val(Ej,Atributo,ValNeg),
   not(ValNeg = ValPos),
   divide(Rest,Atributo,ValPos,Positivos,Negativos).
```

# Sistemas TDIDP

- **Los sistemas basados en árboles de decisión forman una familia llamada TDIDT (*Top–Down Induction of Decision Tree*)**
- **Representantes de TDIDT:**
  - **ID3 (Interactive Dichotomizer) [Quinlan, 1986]**
  - **C4.5 [Quinlan, 93] es una variante de ID3 que permite clasificar ejemplos con atributos que toman valores continuos**
- **Quinlan, J. R. *C4.5: Programs for Machine Learning* (Morgan Kaufmann, 1993)**

# Limitaciones de árboles de decisión

- **Una representación formal limitada (lenguaje de pares atributo–valor equivalente al de la lógica proposicional)**
- **Tienden a ser demasiado grandes en aplicaciones reales**
- **Dificultad del manejo del conocimiento base**

# Programación Lógica Inductiva

- Datos:
  - Ejemplos positivos: $\boldsymbol{E}^{\oplus}$
  - Ejemplos negativos: $\boldsymbol{E}^{\ominus}$
  - Conocimiento base: $\boldsymbol{T}$
  - Lenguaje de hipótesis: $\boldsymbol{L}$
- Condiciones:
  - *Necesidad a priori:* $(\exists \boldsymbol{e}^{\oplus} \in \boldsymbol{E}^{\oplus})[\boldsymbol{T} \nvdash \boldsymbol{e}^{\oplus}]$
  - *Consistencia a priori:* $(\forall \boldsymbol{e}^{\ominus} \in \boldsymbol{E}^{\ominus})[\boldsymbol{T} \nvdash \boldsymbol{e}^{\ominus}]$
- Objetivo:
  - Encontrar un conjunto finito $\boldsymbol{H} \subset \boldsymbol{L}$ tal que se cumplan
    - *Suficiencia a posteriori:* $(\forall \boldsymbol{e}^{\oplus} \in \boldsymbol{E}^{\oplus})[\boldsymbol{T} \bigcup \boldsymbol{H} \vdash \boldsymbol{e}^{\oplus}]$
    - *Consistencia a posteriori:* $(\forall \boldsymbol{e}^{\ominus} \in \boldsymbol{E}^{\ominus})[\boldsymbol{T} \bigcup \boldsymbol{H} \nvdash \boldsymbol{e}^{\ominus}]$

# Ejemplo con FOIL

- **Descripción del problema** `familia.pl`
  - **Ejemplos positivos**

```
padre(carlos,juan).        padre(carlos,eva).
```

  - **Ejemplos negativos**

```
no(padre(carlos,aurora)).  no(padre(aurora,juan)).
```

  - **Conocimiento básico**

```
hombre(carlos).            hombre(juan).
mujer(eva).                mujer(aurora).
progenitor(carlos,juan).   progenitor(carlos,eva).
progenitor(aurora,juan).   progenitor(aurora,eva).
```

  - **Parámetros**

```
foil_predicates([padre/2,hombre/1,mujer/1,progenitor/2]).
foil_cwa(false).           % No usa la hipótesis del mundo
foil_use_negations(false). % No usa información negativa e
foil_det_lit_bound(0).     % No añade literales determinad
```

# Ejemplo con FOIL

- **Sesión**

```
?- [foil, familia].
Yes

?- foil(padre/2).
Positivos no cubiertos: [padre(carlos, juan), padre(carlos,

Cláusula actual: padre(A, B).

Negativos cubiertos: [padre(carlos, aurora), padre(aurora, j
Positivos cubiertos: [padre(carlos, juan), padre(carlos, eva

Especializando la cláusula actual:
Ganancia:  0.830 Cláusula: padre(A, B):-hombre(A)
Ganancia:  0.000 Cláusula: padre(A, B):-hombre(B)
...
```

# Ejemplo con FOIL

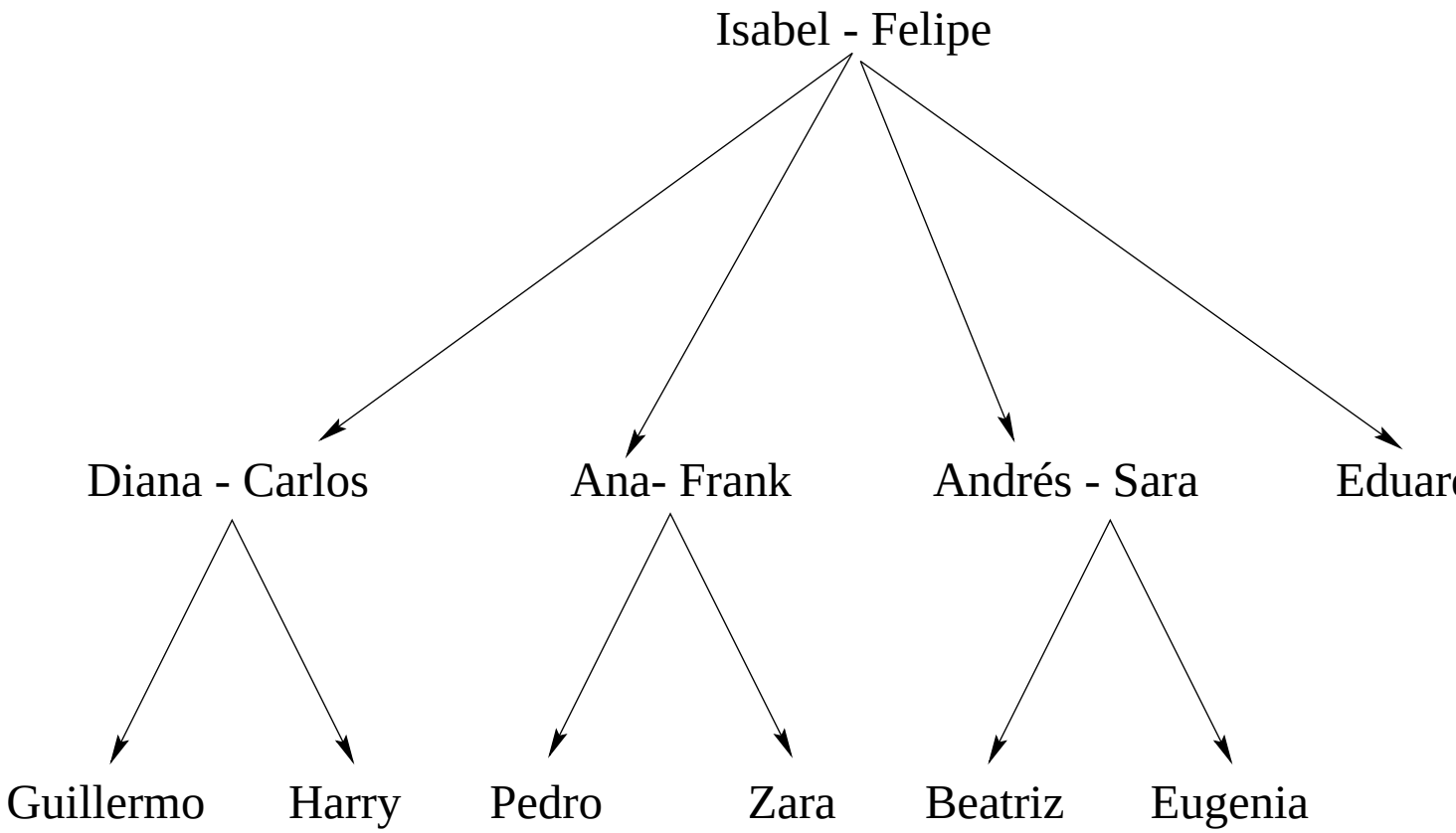
```
Cláusula actual: padre(A, B) :- hombre(A).
Negativos cubiertos: [padre(carlos, aurora)]
Positivos cubiertos: [padre(carlos, juan), padre(carlos, eva

Especializando la cláusula actual:
Ganancia:  0.000 Cláusula: padre(A, B):-hombre(A), hombre(A)
...
Ganancia:  1.170 Cláusula: padre(A, B):-hombre(A), progenito
...

Cláusula encontrada: padre(A, B) :- hombre(A), progenitor(A,

Definición encontrada: padre(A, B) :- hombre(A), progenitor(
```

# Ganancia de información en FOIL

- **Información correspondiente a una cláusula:**
  **Sean $P$ el número de ejemplos positivos cubiertos por la cláusula y $N$ el número de ejemplos negativos cubiertos por la cláusula.**
  $$I(C) = \begin{cases} 0, & \text{si } P = 0; \\ -\log_2 \frac{P}{P+N}, & \text{si } P \neq 0. \end{cases}$$

- **Ganancia de información de la cláusula $C_1$ a la cláusula $C_2$:**
  **Sean $P$ el número de ejemplos positivos cubiertos por $C_2$ y $N$ el número de ejemplos negativos cubiertos $C_2$.**
  $$G(C_1, C_2) = \begin{cases} 0, & \text{si } P = 0; \\ P * (I(C_1) - I(C_2)), & \text{si } P \neq 0. \end{cases}$$

# Ejemplo de FOIL con CWA



# Ejemplo de FOIL con CWA

- **Representación** `familia_1.pl`
  - **Parámetros**

```
foil_predicates([padre/2, madre/2, abuelo/2]).
foil_cwa(true).                % Usa la hipótesis del mundo
foil_use_negations(false).     % No usa información negativa
foil_det_lit_bound(0).         % No añade literales determin
```

  - **Ejemplos positivos**

```
padre(felipe,carlos).       padre(felipe,ana).      padre(fe
madre(isabel,carlos).       madre(isabel,ana).      madre(is
abuelo(felipe,guillermo).   abuelo(felipe,harry).   abuelo(f
```

# Ejemplo de FOIL con CWA

- **Sesión**

```
?- [foil, familia_1].
Yes
?- foil(abuelo/2).

Positivos no cubiertos: [(felipe,guillermo),(felipe,harry),(
                         (felipe,zara),(felipe,beatriz),(fel

Cláusula actual: abuelo(A,B).
Negativos cubiertos: [(ana,ana),(ana,andres),...]
Positivos cubiertos: [(felipe,guillermo),(felipe,harry),(fel
                      (felipe,zara),(felipe,beatriz),(felipe

Cláusula actual: abuelo(A,B) :- padre(A,C).
Negativos cubiertos: [(andres,ana),(andres,andres),...]
Positivos cubiertos: [(felipe,guillermo),(felipe,harry),(fel
                     (felipe,zara),(felipe,beatriz),(felipe,e

Cláusula encontrada: abuelo(A,B) :- padre(A,C),padre(C,B).
```

# Ejemplo de FOIL con CWA

```
Positivos no cubiertos: [(felipe,pedro),(felipe,zara)]

Añadiendo una nueva cláusula ...
Cláusula actual: abuelo(A,B).

Negativos cubiertos: [(ana,ana),(ana,andres),...]
Positivos cubiertos: [(felipe,pedro),(felipe,zara)]

Cláusula actual: abuelo(A,B) :- padre(A,C).
Negativos cubiertos: [(andres,ana),(andres,andres),...]
Positivos cubiertos: [(felipe,pedro),(felipe,zara)]

Cláusula encontrada: abuelo(A,B) :- padre(A,C),madre(C,B).

Definición encontrada:
abuelo(A,B) :- padre(A,C),madre(C,B).
abuelo(A,B) :- padre(A,C),padre(C,B).
```

# Ejemplo de FOIL con CWA y conocimiento

- **Representación** `familia_2.pl`
  - **Parámetros**

```
foil_predicates([padre/2, madre/2, abuelo/2, progenitor/2]
foil_cwa(true).              % Usa la hipótesis del mundo c
foil_use_negations(false).   % No usa información negativa
foil_det_lit_bound(0).       % No añade literales determina
```

  - **Ejemplos positivos**

```
padre(felipe,carlos).       padre(felipe,ana).      padre(fe
madre(isabel,carlos).       madre(isabel,ana).      madre(is
abuelo(felipe,guillermo).   abuelo(felipe,harry).   abuelo(f
```

  - **Conocimiento básico**

```
progenitor(X,Y) :- padre(X,Y).
progenitor(X,Y) :- madre(X,Y).
```

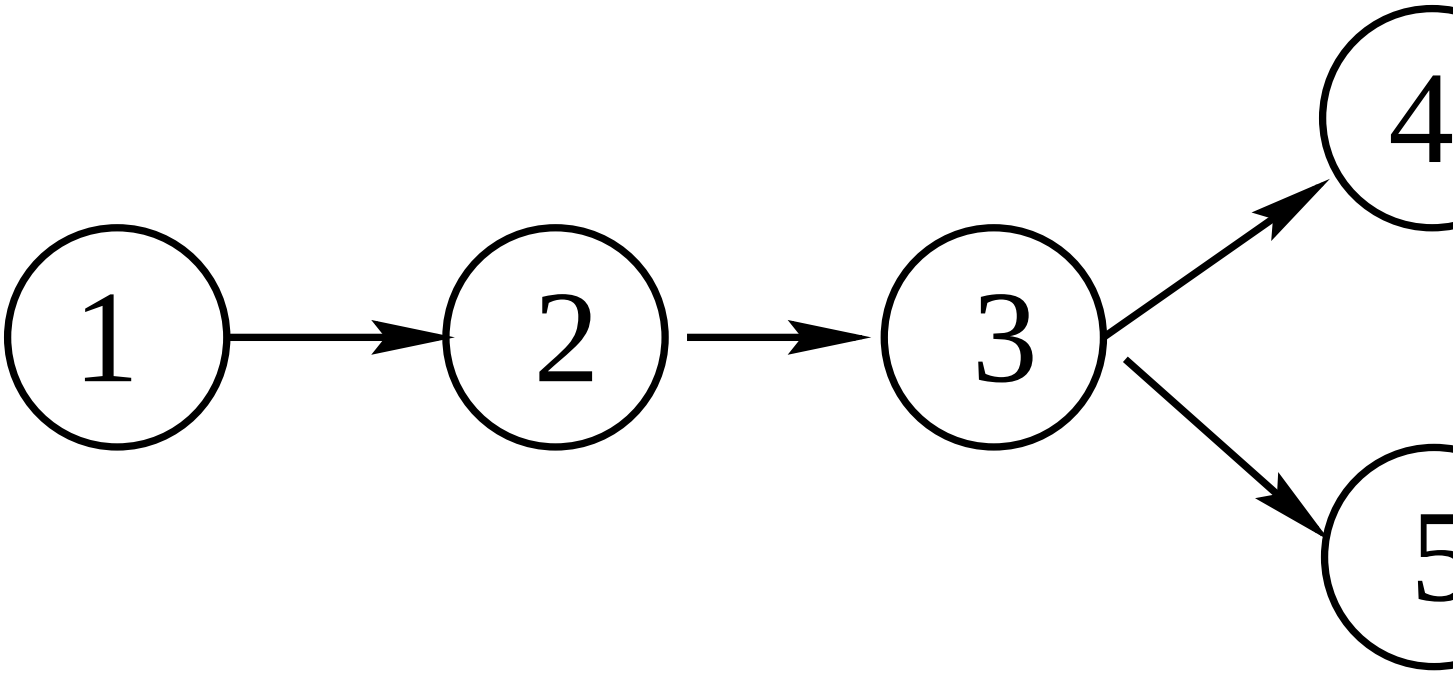
# Ejemplo de FOIL con CWA y conocimiento

- **Sesión**

```
?- [foil, familia_2].
Yes
?- foil(abuelo/2).
Positivos no cubiertos: [(felipe, guillermo), (felipe, harry
                         (felipe, zara), (felipe, beatriz),

Cláusula actual: abuelo(A, B).
Negativos cubiertos: [(ana, ana), (ana, andres), ...]
Positivos cubiertos: [(felipe, guillermo), (felipe, harry),
                      (felipe, zara), (felipe, beatriz), (fe

Negativos cubiertos: [(andres, ana), (andres, andres), ...]
Positivos cubiertos: [(felipe, guillermo), (felipe, harry),
                      (felipe, zara), (felipe, beatriz), (fe

Clause encontrada: abuelo(A, B) :- padre(A, C), progenitor(C

Definición encontrada: abuelo(A, B) :- padre(A, C), progenit
```

# Ejemplo con FOIL de relación recursiva

- **Grafo**



# Ejemplo con FOIL de relación recursiva

- **Representación** `camino.pl`
  - **Parámetros**

```
foil_predicates([camino/2, enlace/2]).
foil_cwa(true).                % Usa la hipótesis del mundo
foil_use_negations(false).     % No usa información negativa
foil_det_lit_bound(0).         % No añade literales determin
```

  - **Ejemplos**

```
enlace(1,2).  enlace(2,3).  enlace(3,4).  enlace(3,5).
camino(1,2).  camino(1,3).  camino(1,4).  camino(1,5).
camino(2,3).  camino(2,4).  camino(2,5).
camino(3,4).  camino(3,5).
```

# Ejemplo con FOIL de relación recursiva

```
?- [foil,camino].
?- foil(camino/2).
Positivos no cubiertos: [(1,2),(1,3),(1,4),(1,5),(2,3),(2,4)
Añadiendo Cláusula actual: camino(A,B).
Negativos cubiertos: [(1,1),(2,1),(2,2),(3,1),(3,2),(3,3),(4
                      (4,4),(4,5),(5,1),(5,2),(5,3),(5,4),(5
Positivos cubiertos: [(1,2),(1,3),(1,4),(1,5),(2,3),(2,4),(2
Cláusula encontrada: camino(A,B) :- enlace(A,B).
Positivos no cubiertos: [(1,3),(1,4),(1,5),(2,4),(2,5)]
Añadiendo Cláusula actual: camino(A,B).
Negativos cubiertos: [(1,1),(2,1),(2,2),(3,1),(3,2),(3,3),(4
                      (4,4),(4,5),(5,1),(5,2),(5,3),(5,4),(5
Positivos cubiertos: [(1,3),(1,4),(1,5),(2,4),(2,5)]
Cláusula actual: camino(A,B) :- enlace(A,C).
Negativos cubiertos: [(1,1),(2,1),(2,2),(3,1),(3,2),(3,3)]
Positivos cubiertos: [(1,3),(1,4),(1,5),(2,4),(2,5)]
Cláusula encontrada: camino(A,B) :- enlace(A,C), camino(C,B)

Definición encontrada:
camino(A,B) :- enlace(A,C), camino(C,B).
camino(A,B) :- enlace(A,B).
```

# Algoritmo de FOIL

```
% foil(+Positivos, +Objetivo, +Negativos, -Clausulas)
foil(Positivos, Objetivo, Negativos, Clausulas) :-
   foil(Positivos, Objetivo, Negativos, [], Clausulas).

foil(Positivos, Objetivo, Negativos, Acumulador, Clausulas)
   ( Positivos = [] -> Clausulas = Acumulador
   ; extiende_clausula(Negativos, Positivos, (Objetivo :- tr
     ejemplos_no_cubiertos(Clausula, Positivos, Positivos1),
     foil(Positivos1, Objetivo, Negativos, [Clausula|Acumula

% extiende_clausula(+Negativos, +Positivos, +Actual, -Clausu
extiende_clausula(Neg0, Pos0, Clausula0, Clausula) :-
   ( Neg0 = [] -> Clausula = Clausula0
   ; genera_posibles_extensiones(Clausula0, L),
     informacion(Clausula0, Pos0, Neg0, Info),
     mejor_extension(L, Neg0, Pos0, Clausula0, Info, 0, Clau
     Clausula0 \== Clausula1,
     ejemplos_cubiertos(Clausula1, Pos0, Pos1),
     ejemplos_cubiertos(Clausula1, Neg0, Neg1),
     extiende_clausula(Neg1, Pos1, Clausula1, Clausula))).
```

# Bibliografía


- Cazorla, M.A. et als. *Técnicas de Inteligencia Artificial* (Publicaciones de la Universidad de Alicante, 1999)
  - Cap. 9: "Aprendizaje inductivo"
- Flach, P. *Simply Logical (Intelligent Reasoning by Example)* (John Wiley, 1994)
  - Cap. 9: "Inductive reasoning"
- Mitchell, T.M. *Machine learning* (McGraw–Hill, 1997)
  - Cap. 3: "Decision tree learning"
  - Cap. 11: "Learning sets of rules"

# Bibliografía


- Moreno, A. et als. *Aprendizaje automático* (Edicions UPC, 1994)
  - Cap. 2: "Aprendizaje inductivo"
- Poole, D.; Mackworth, A. y Goebel, R. *Computational Intelligence (A Logical Approach)* (Oxford University Press, 1998)
  - Cap. 11: "Learning"
- Russell, S. y Norvig, P. *Inteligencia artificial (Un enfoque moderno)* (Prentice–Hall Hispanoamericana, 1996)
  - Cap. 18: "Aprendizaje a partir de la observación"
  - Cap. 21: "El conocimiento en el aprendizaje"
- Quinlan, J. R. *C4.5: Programs for Machine Learning* (Morgan Kaufmann, 1993)