Tema 5

Adquisición de conceptos

Ciencias de la Computación e Inteligencia Artificial
Universidad de Huelva

noviembre 2023

Índice

Introducción

Adquisición de conceptos

Espacio de versiones
- List-Then-Eliminate
- Find-S
- Eliminación de candidatos

Índice

Introducción

Adquisición de conceptos

Espacio de versiones
- List-Then-Eliminate
- Find-S
- Eliminación de candidatos

Definición

- El **aprendizaje inductivo** consiste en inducir información de un concepto a partir de un conjunto de ejemplos de cosas concretas. No requiere información previa del dominio.
- Ejemplo: ese gato tiene 4 patas, **POR TANTO**, todos los gatos tienen 4 patas.
- Si veo 1 caso, supongo que todos los casos son iguales hasta que encuentre un nuevo caso que lo contradiga, y eso me obligue a remodelar las informaciones.

Introducción

Aprendizaje inductivo

- **Simbólico**
  - Utiliza una representación simbólica de los conceptos y sus relaciones (redes semánticas, reglas, programación lógica, ...)
- **Subsimbólico**
  - Utiliza una representación mezclada entre símbolos y números (conjuntos difusos,...)
  - Habitualmente se desarrolla por medio de algoritmos de ajuste paramétrico

Introducción

En este curso nos vamos a centrar en las formas de aprendizaje inductivo **simbólico supervisado**

Tipos:

- Si los ejemplos reflejan situaciones con múltiples objetos y relaciones
  - **Adquisición de conceptos**
- Si los ejemplos se refieren a conjuntos atributo-valor
  - **Clasificación supervisada**
- Si se pretende adquirir un modelo lógico
  - **Programación Lógica Inductiva**

Índice

Introducción

Adquisición de conceptos

Espacio de versiones
- List-Then-Eliminate
- Find-S
- Eliminación de candidatos

Adquisición de conceptos

Para la adquisición de conceptos vamos a necesitar algunos elementos, similares a la lógica: :

- Un lenguaje de **representación**: Es decir, una forma de escribir nuestras **Hipótesis (y modelos)**
- Un motor (bidireccional) de **razonamiento**::
  - Un proceso de **generalización**: para incorporar ejemplos positivos al modelo (bottom-up)
  - Un proceso de **especialización**: para rechazar ejemplos negativos (*quasi-ejemplos*) con el modelo (top-down)

Adquisición de conceptos

Para la adquisición de conceptos vamos a necesitar algunos elementos, similares a la lógica: :

- Un lenguaje de **representación**: Es decir, una forma de escribir nuestras **Hipótesis (y modelos)**
- Un motor (bidireccional) de **razonamiento**::
  - Un proceso de **generalización**: para incorporar ejemplos positivos al modelo (bottom-up)
  - Un proceso de **especialización**: para rechazar ejemplos negativos (*quasi-ejemplos*) con el modelo (top-down)

Adquisición de conceptos

Para la adquisición de conceptos vamos a necesitar algunos elementos, similares a la lógica: :

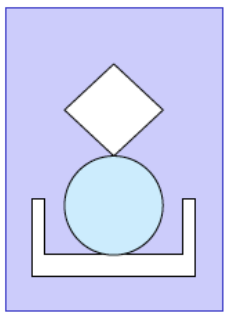
- Un lenguaje de **representación**: Es decir, una forma de escribir nuestras **Hipótesis (y modelos)**
- Un motor (bidireccional) de **razonamiento**::
  - Un proceso de **generalización**: para incorporar ejemplos positivos al modelo (bottom-up)
  - Un proceso de **especialización**: para rechazar ejemplos negativos (*quasi-ejemplos*) con el modelo (top-down)

Hipótesis y Modelos

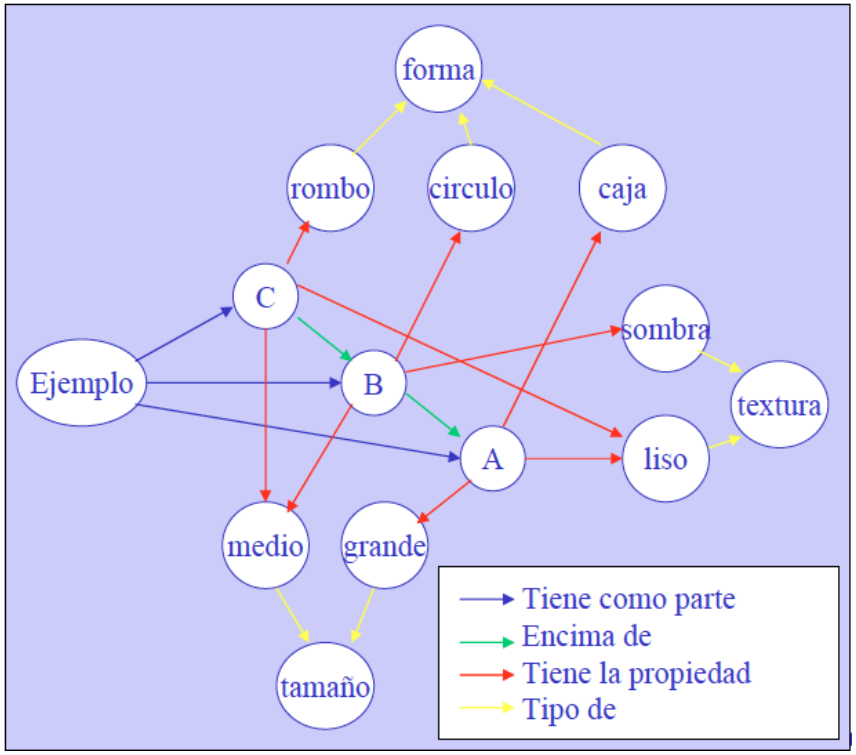
- Un modelo se puede ver cómo la representación o abstracción de un sistema
- En nuestro caso, de aprendizaje automático un modelo va a representar a un conjunto de datos mediante un conjunto de fórmulas.
- La utilidad es poder usar el modelo para deducir/predecir cosas, en lugar de tener que estar manejando TODOS los datos.
- Hay muchas formas de escribir modelos: redes semánticas, lógica, fórmulas matemáticas, etc . . . .

Hipótesis y Modelos

Como ejemplo, podemos ver la figura



Modelo



Red semántica

Hipótesis y Modelos

Más concretamente, podemos hacer las siguientes definiciones:

- **Hipótesis**: Posible representación de un concepto
- **Modelo**: Es una hipótesis que es consistente[1] con todos los datos
- **Orden**: Se puede decir que una hipótesis es más grande que otra, $H_1 \geq_g H_2$ si y solo si tiene la posibilidad de contener a más ejemplos.
- Si $H1 \geq_g H2$ se dice que $H1$ es **más general que** $H2$
  Y, por tanto, $H2$ es **más específica que** $H1$

[1] que se cumple para todos los ejemplos positivos y para ninguno de los negativos

Hipótesis y Modelos

Más concretamente, podemos hacer las siguientes definiciones:

- **Hipótesis**: Posible representación de un concepto
- **Modelo**: Es una hipótesis que es consistente[1] con todos los datos
- **Orden**: Se puede decir que una hipótesis es más grande que otra, $H_1 \geq_g H_2$ si y solo si tiene la posibilidad de contener a más ejemplos.
- Si $H1 \geq_g H2$ se dice que $H1$ es **más general que** $H2$
  Y, por tanto, $H2$ es **más específica que** $H1$

[1] que se cumple para todos los ejemplos positivos y para ninguno de los negativos

Hipótesis y Modelos

Más concretamente, podemos hacer las siguientes definiciones:

- **Hipótesis**: Posible representación de un concepto
- **Modelo**: Es una hipótesis que es consistente[1] con todos los datos
- **Orden**: Se puede decir que una hipótesis es más grande que otra, $H_1 \geq_g H_2$ si y solo si tiene la posibilidad de contener a más ejemplos.
- Si $H1 \geq_g H2$ se dice que $H1$ es **más general que** $H2$
  Y, por tanto, $H2$ es **más específica que** $H1$

[1] que se cumple para todos los ejemplos positivos y para ninguno de los negativos

Hipótesis y Modelos

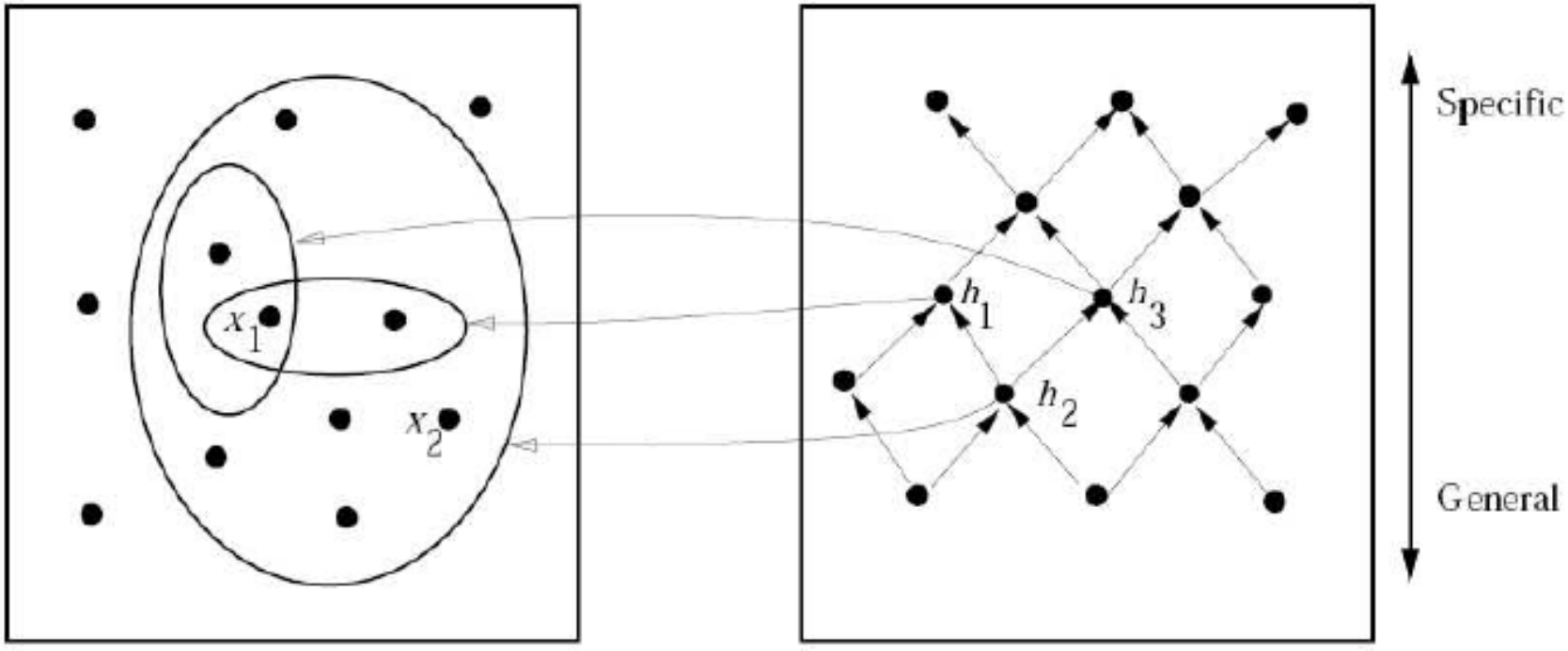
Más concretamente, podemos hacer las siguientes definiciones:

- **Hipótesis**: Posible representación de un concepto
- **Modelo**: Es una hipótesis que es consistente[1] con todos los datos
- **Orden**: Se puede decir que una hipótesis es más grande que otra, $H_1 \geq_g H_2$ si y solo si tiene la posibilidad de contener a más ejemplos.
- Si $H1 \geq_g H2$ se dice que $H1$ es **más general que** $H2$
  Y, por tanto, $H2$ es **más específica que** $H1$

[1] que se cumple para todos los ejemplos positivos y para ninguno de los negativos

Hipótesis y Modelos

- En este ejemplo podemos decir que $h2 \geq_g h1$ y que $h2 \geq_g h3$
- No podemos decir nada sobre h1 y h3, ya que ninguna es subconjunto de la otra
- Esto se llama **Orden Parcial**



Razonamiento

- Para la tarea de aprendizaje necesitaremos razonar sobre los objetos y las hipótesis.
- Si una hipótesis no acepta a un ejemplo positivo o no rechaza a un negativo, tenemos que reformarla para que lo haga.
- El proceso de agrandar una hipótesis para aceptar un ejemplo positivo se llama **Generalización**

```
Gen:: (Modelo, Ejemplo) -> [Modelo]
```

- El proceso para evitar aceptar ejemplos negativos se llama **Especificación**

```
Spec:: (Modelo, Ejemplo) -> [Modelo]
```

Proceso de **Generalización**

```
Gen:: (Modelo, Ejemplo) -> [Modelo]
```

- Este proceso es, básicamente, la eliminación de restricciones para hacer el modelo más permisivo y que pueda admitir al nuevo ejemplo positivo
- Como se ve en la cabecera de la función, la generalización no es única, puede dar lugar a diferentes modelos
- El método de generalización depende mucho del tipo de datos que estemos tratando.
- $\forall B \in Gen(A), B \geq_g A$

Proceso de **Especialización**

```
Gen:: (Modelo, Ejemplo) -> [Modelo]
```

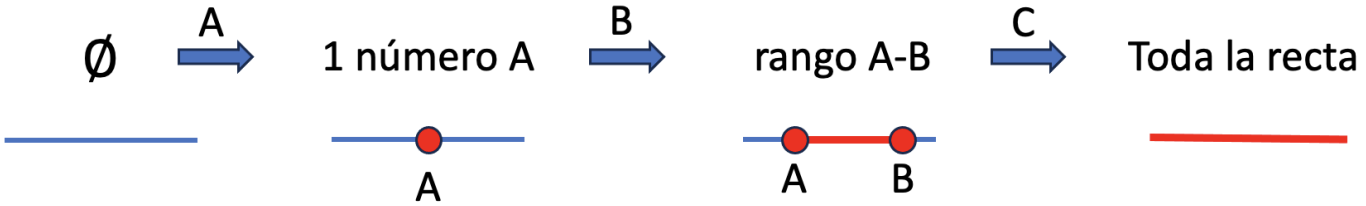
- En la especificación se trata de rechazar a ejemplos negativos que el modelo admite, por lo que, principalmente, incluiremos nuevas restricciones que dejen afuera al ejemplo negativo.
- La especificación tampoco es única y,por tanto, puede dar lugar a diferentes modelos
- Este método también depende del tipo de dato
- $\forall B \in Spec(A), A \geq_g B$

Algunos ejemplos

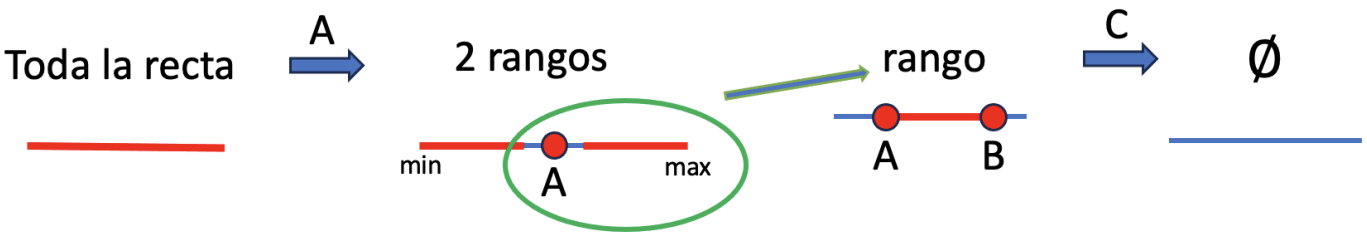
- Tipo de dato: Categórico
- Generalización:
  - $(\emptyset, value) \rightarrow [(value)]$
  - $((value), value) \rightarrow [(value)]$
  - $((value1), value2) \rightarrow [(?)]$
  - $((?), value) \rightarrow [(?)]$
- Especificación:
  - $(?, value) \rightarrow [(value1), (value2) \ldots (valuen)]$ Todos diferentes de value
  - $((value), value) \rightarrow [(\emptyset)]$
  - $((value1), value2) \rightarrow [(value1)]$
  - $((\emptyset), value) \rightarrow [(\emptyset)]$

Algunos ejemplos

- Tipo de dato: Numérico
- Generalización:

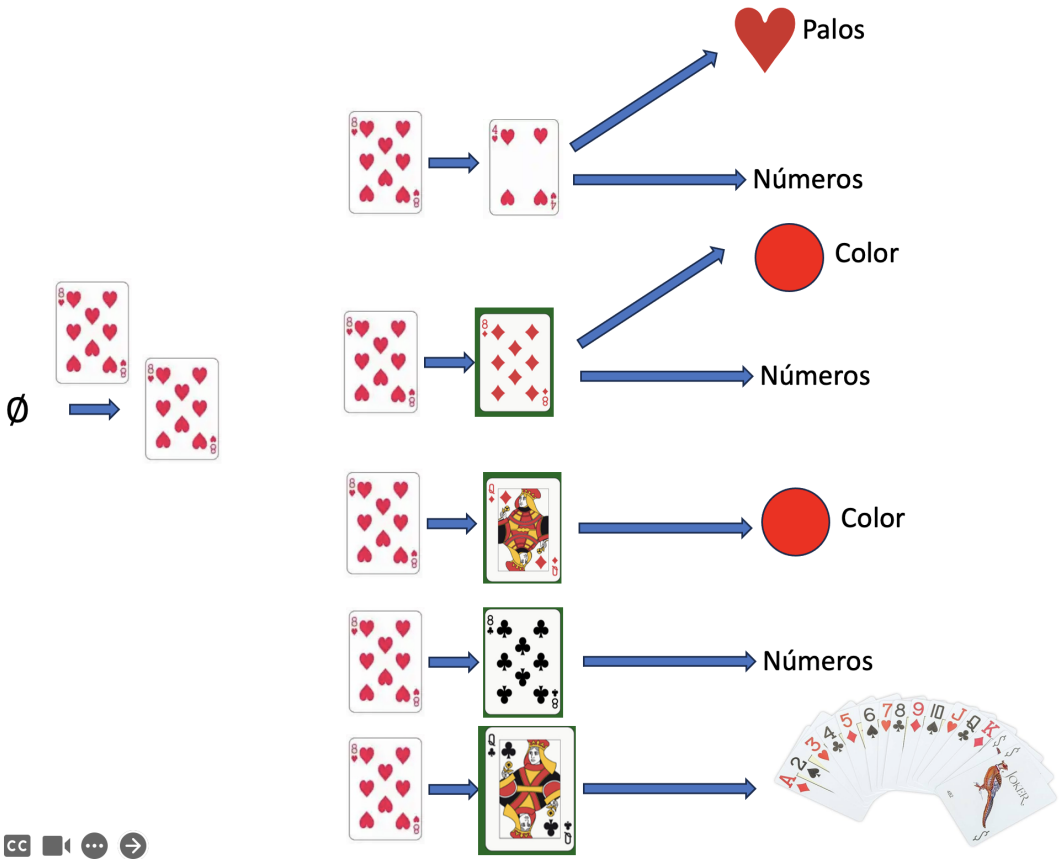


- Especificación:



Algunos ejemplos

- Tipo de dato: Complejo
- Generalización:



Representaciones

Necesitamos una representación de los objetos y unos métodos iniciales.:

- Necesitamos representar el VACÍO y el TODO:

```
1  EMPTY = "-"
2  ALL   = "?"
```

- Dado un dataset de *n* atributos, escribiremos las hipótesis sobre ellos en un objeto (tupla o lista) de n-1 componentes (1 es la clase

```
1  HIP = (EMPTY, valuei, ..., ALL, valuek)
```

Representaciones

- Una hipótesis acepta un ejemplo (es consistente con él) cuando es consistente para todos sus atributos
  H consistente con $e \iff \forall h_i \in H \; consist\_at(h, e)$,
  $i = 1, \ldots, n-1$

```
1 def consistente(Hypo,example):
2     for h in H:
3         if !consist_at(h,e):
4             return False
5     return True
```

- Una componente de una hipótesis acepta un atributo, si:
  - Si el ejemplo es positivo, cumple el patrón de la hipótesis
  - Si el ejemplo es negativo, no cumple el patrón de la hipótesis

```
1 def consist_at(h,attribute):
2     if positive(attribute):
3         return holds(h,attribute)
4     else:
5         return !holds(h,attribute)
```

Índice

Introducción

Adquisición de conceptos

Espacio de versiones
- List-Then-Eliminate
- Find-S
- Eliminación de candidatos

Espacio de versiones

- El espacio de versiones es un framework diseñado para la tarea de aprendizaje inductivo simbólico.

  Propuesto por Mitchell en 1982

- El objetivo es producir una descripción de un concepto a partir de un entrenamiento con ejemplos positivos y negativos.

- Es independiente de la tarea a aprender

- No se ve afectado por el orden en que se presentan los ejemplos.

Espacio de versiones

- El espacio de versiones es un framework diseñado para la tarea de aprendizaje inductivo simbólico.

  Propuesto por Mitchell en 1982

- El objetivo es producir una descripción de un concepto a partir de un entrenamiento con ejemplos positivos y negativos.
- Es independiente de la tarea a aprender
- No se ve afectado por el orden en que se presentan los ejemplos.

Espacio de versiones

- Necesitamos algunas definiciones previas:
  - **Espacio de hipótesis**: el conjunto de todas las hipótesis que se pueden escribir en el lenguaje elegido
  - **Espacio de versiones**: Conjunto de todas las hipótesis consistentes con el conjunto de ejemplos.

Espacio de versiones

- Necesitamos algunas definiciones previas:
  - **Espacio de hipótesis**: el conjunto de todas las hipótesis que se pueden escribir en el lenguaje elegido
  - **Espacio de versiones**: Conjunto de todas las hipótesis consistentes con el conjunto de ejemplos.

Espacio de versiones

- Necesitamos algunas definiciones previas:
  - **Espacio de hipótesis**: el conjunto de todas las hipótesis que se pueden escribir en el lenguaje elegido
  - **Espacio de versiones**: Conjunto de todas las hipótesis consistentes con el conjunto de ejemplos.

Algoritmos

En función a cómo tratemos estos espacios de hipótesis y versiones, presentamos varias aproximaciones a los algoritmos:

- Con todo el espacio de hipótesis: **List-Then-Eliminate**
- Con una cota superior o inferior: **Find-S y Dual-Find-S**
- Con cota superior e inferior a la vez: **Eliminación de candidatos**

Algoritmos

En función a cómo tratemos estos espacios de hipótesis y versiones, presentamos varias aproximaciones a los algoritmos:

- Con todo el espacio de hipótesis: **List-Then-Eliminate**
- Con una cota superior o inferior: **Find-S y Dual-Find-S**
- Con cota superior e inferior a la vez: **Eliminación de candidatos**

Algoritmos

En función a cómo tratemos estos espacios de hipótesis y versiones, presentamos varias aproximaciones a los algoritmos:

- Con todo el espacio de hipótesis: **List-Then-Eliminate**
- Con una cota superior o inferior: **Find-S y Dual-Find-S**
- Con cota superior e inferior a la vez: **Eliminación de candidatos**

Algoritmos

En función a cómo tratemos estos espacios de hipótesis y versiones, presentamos varias aproximaciones a los algoritmos:

- Con todo el espacio de hipótesis: **List-Then-Eliminate**
- Con una cota superior o inferior: **Find-S y Dual-Find-S**
- Con cota superior e inferior a la vez: **Eliminación de candidatos**

List-Then-Eliminate

- Este es un algoritmo de fuerza bruta.
- Genera TODO el espacio de hipótesis posible. Esto en la mayoría de los casos es inabordable.
- Después analiza todos los ejemplos, uno a uno, eliminando las hipótesis que no son consistentes[2]

```
1 LTE :: DataSet -> [Hypo]
2 def LTE(dataset):
3     vspace = []
4     hspace = genera(dataset)
5     for h in hspace:
6         if consistent(dataset,h):
7             vspace.append(h)
8     return vspace
```

[2] las que rechazan a los positivos o admiten a los negativos

Find-S

- En este algoritmo establecemos la hipótesis más pequeña que cubre a todos los positivos
- Partimos de la hipótesis más específica y vamos generalizando (“agrandando”) con cada ejemplo positivo que no esté cubierto.

```
1 FindS :: DataSet -> Hypo
2
3 def FindS(Dataset):
4     H = (EMPTY,EMPTY,...,EMPTY)
5     for example in dataset:
6         if positive(example):
7             if !consistent(example, H):
8                 H = Generalize(H)
9     return H
```

Dual Find-S

Es similar al algoritmo Find-S, pero en orden inverso

- Partimos de la hipótesis más general y especificamos por cada ejemplo negativo que sea cubierto la hipótesis.
- Partimos de la hipótesis más específica y vamos generalizando (“agrandando”) con cada ejemplo positivo que no esté cubierto.

```
1 DualFindS :: DataSet -> Hypo
2
3 def FindS(Dataset):
4     H = (ALL,ALL,...,ALL)
5     for example in dataset:
6         if negative(example):
7             if !consistent(example, H):
8                 H = Specialize(H)
9     return H
```

Algoritmo Find-S

Comenzamos con la Hip Más específica	$h_0 = (\emptyset, \emptyset, \emptyset, \emptyset, \emptyset)$
Ejemplo Positivo	
x_1 = <Sunny Warm Normal Strong Warm Same>, +	h_1 = (Sunny,Warm,Normal,Strong,Warm,Same)
Ejemplo Positivo	
x_2 = <Sunny Warm High Strong Warm Same>, +	h_2 = (Sunny,Warm,?,Strong,Warm,Same)
Ejemplo Negativo	
x_3 = <Rainy Cold High Strong Warm Change>, -	h_3 = (Sunny,Warm,?,Strong,Warm,Same)
Ejemplo Positivo	
x_4 = <Sunny Warm High Strong Cool Change>, +	h_2 = (Sunny,Warm,?,Strong,?,?)

Algoritmo Find-S

Comenzamos con la Hip Más específica	$h_0 = (\emptyset, \emptyset, \emptyset, \emptyset, \emptyset)$
Ejemplo Positivo	
x_1 = <Sunny Warm Normal Strong Warm Same>, +	h_1 = (Sunny,Warm,Normal,Strong,Warm,Same)
Ejemplo Positivo	
x_2 = <Sunny Warm High Strong Warm Same>, +	h_2 = (Sunny,Warm,?,Strong,Warm,Same)
Ejemplo Negativo	
x_3 = <Rainy Cold High Strong Warm Change>, -	h_3 = (Sunny,Warm,?,Strong,Warm,Same)
Ejemplo Positivo	
x_4 = <Sunny Warm High Strong Cool Change>, +	h_2 = (Sunny,Warm,?,Strong,?,?)

Algoritmo Find-S

Comenzamos con la Hip Más específica	$h_0 = (\emptyset, \emptyset, \emptyset, \emptyset, \emptyset)$
Ejemplo Positivo	
x_1 = <Sunny Warm Normal Strong Warm Same>, +	h_1 = (Sunny,Warm,Normal,Strong,Warm,Same)
Ejemplo Positivo	
x_2 = <Sunny Warm High Strong Warm Same>, +	h_2 = (Sunny,Warm,?,Strong,Warm,Same)
Ejemplo Negativo	
x_3 = <Rainy Cold High Strong Warm Change>, -	h_3 = (Sunny,Warm,?,Strong,Warm,Same)
Ejemplo Positivo	
x_4 = <Sunny Warm High Strong Cool Change>, +	h_2 = (Sunny,Warm,?,Strong,?,?)

Algoritmo Find-S

Comenzamos con la Hip Más específica	$h_0 = (\emptyset, \emptyset, \emptyset, \emptyset, \emptyset)$
Ejemplo Positivo	
x_1 = <Sunny Warm Normal Strong Warm Same>, +	h_1 = (Sunny,Warm,Normal,Strong,Warm,Same)
Ejemplo Positivo	
x_2 = <Sunny Warm High Strong Warm Same>, +	h_2 = (Sunny,Warm,?,Strong,Warm,Same)
Ejemplo Negativo	
x_3 = <Rainy Cold High Strong Warm Change>, -	h_3 = (Sunny,Warm,?,Strong,Warm,Same)
Ejemplo Positivo	
x_4 = <Sunny Warm High Strong Cool Change>, +	h_2 = (Sunny,Warm,?,Strong,?,?)

Algoritmo Find-S

Comenzamos con la Hip Más específica	$h_0 = (\emptyset, \emptyset, \emptyset, \emptyset, \emptyset)$
Ejemplo Positivo	
x_1 = <Sunny Warm Normal Strong Warm Same>, +	h_1 = (Sunny,Warm,Normal,Strong,Warm,Same)
Ejemplo Positivo	
x_2 = <Sunny Warm High Strong Warm Same>, +	h_2 = (Sunny,Warm,?,Strong,Warm,Same)
Ejemplo Negativo	
x_3 = <Rainy Cold High Strong Warm Change>, -	h_3 = (Sunny,Warm,?,Strong,Warm,Same)
Ejemplo Positivo	
x_4 = <Sunny Warm High Strong Cool Change>, +	h_2 = (Sunny,Warm,?,Strong,?,?)

Aspectos negativos

- Siempre se va a generar una hipótesis consistente con los ejemplos (se ignoran los negativos)
- No puede asegurar que se haya aprendido el concepto correcto, porque coge una de las hipótesis posibles.
- No soporta ruido en los ejemplos positivos

Eliminación de candidatos

- Las opciones anteriores son demasiado costosas.
- Debemos de tener una opción que sea más tratable.
- El algoritmo de Eliminación de Candidatos mantiene “las cotas” de hipótesis superior e inferior de todas las hipótesis consistentes con los ejemplos.
- Cada tratamiento de ejemplos generaliza y especializa el grafo de las hipótesis, para obtener un espacio final compatible.

Eliminación de candidatos

Algoritmo:

- Entrada: Conjunto de datos.
- Salida:
  - G = Hipótesis genéricas minimales
  - S = Hipótesis específicas maximales
- Las hipótesis se representan en un retículo con orden parcial

Eliminación de Candidatos

Algorithm 1: Candidate Elimination Algorithm

Data: D: a dataset of objects labeled as positive or negative
Result: V: the version space of hypotheses consistent with D
Initialize G to the set containing the most general hypothesis
Initialize S to the set containing the most specific hypothesis
for *each object* $x \in D$ **do**
  if x *is a positive object* **then**
    Remove from G any hypothesis inconsistent with x
    for *each hypothesis* $s \in S$ *that is inconsistent with* x **do**
      Remove s from S
      Add to S all the minimal generalizations h of s such that h is consistent with x and for some member g of G it holds that $g \geq h$
      Remove from S any hypothesis that's more general than another hypothesis in S
    end
  end
  else
    Remove from S any hypothesis inconsistent with x
    for *each hypothesis* $g \in G$ *that is inconsistent with* x **do**
      Remove g from G
      Add to G all the minimal specializations h of g such that h is consistent with x and for some member s of S it holds that $h \geq s$
      Remove from G any hypothesis that's less general than another hypothesis in G
    end
  end
end
return V as $V(G, S)$

Eliminación de Candidatos

```
G = (ALL,ALL, ... , ALL)                # Hipótesis más genérica
S = (EMPTY, EMPTY, ... , EMPTY)         # Hipótesis más específica

for ejemplo in dataset:
    if positivo(ejemplo):
        G = Eliminar_Incons(G,ejemplo)
        for s in S:
            if not consistente(s, ejemplo):
                S.remove(s)
                gms = Generalize(s)
                for g in gms:
                    if consistente(g,ejemplo) and exists(mas_general(G,g)):
                        S.append(g)
        for s1 in S:
            for s2 in S:
                if mas_general(s2,s1):
                    S.remove(s2)

    else:
......
```

Eliminación de Candidatos

```
.......
    else:
        S = Eliminar_Incons(S,ejemplo)
        for g in G:
            if not consistente(g, ejemplo):
                G.remove(g)
                sms = Specialize(g)
                for s in sms:
                    if consistente(s,ejemplo) and exists(mas_general(s,S)):
                        G.append(s)
        for g1 in G:
            for g2 in G:
                if mas_general(g1,g2):
                    S.remove(s2)
return (G,S)
```

Ejemplo

Aplicar el algoritmo de Eliminación de candidatos al siguiente conjunto de datos

Cielo	Temperatura	Humedad	Viento	Agua	Previsión	Hacer Deporte
Soleado	Templada	Normal	Fuerte	Templada	Igual	Sí
Soleado	Templada	Alta	Fuerte	Templada	Igual	Sí
Lluvia	Fría	Alta	Fuerte	Templada	Cambio	No
Soleado	Templada	Alta	Fuerte	Fría	Cambio	Sí

Ejemplo

Paso 0: $S_0 = \{< \emptyset, \emptyset, \emptyset, \emptyset, \emptyset >\}$, $G_0 = \{<?, ?, ?, ?, ? >\}$

Ejemplo

Paso 1:

- Ejemplo **positivo**:
  $< Sol; Templ; Normal; Fuerte; Templ; Igual >$
- Nada que eliminar de G_0
- Generalización minimal de S_0:
  $< Sol; Templ; Normal; Fuerte; Templ; Igual >$
- Esta generalización es más específica que la hipótesis de G0
- Luego:
  - $G1 = \{<?; ?; ?; ?; ?; ? >\}$
  - $S1 = \{< Sol; Templ; Normal; Fuerte; Templ; Igual >\}$

Ejemplo

Paso 2:

- Ejemplo **positivo**:
  $< Sol; Templ; Alta; Fuerte; Templ; Igual >$
- Nada que eliminar de G1
- Generalización minimal de
  $S1 :< Sol; Templ; ?; Fuerte; Templ; Igual >$
- Esta generalización es más específica que la hipótesis de G1
- Luego:
  - $G2 = \{<?; ?; ?; ?; ?; ? >\}$
  - $S2 = \{< Sol; Templ; ?; Fuerte; Templ; Igual >\}$

Ejemplo

Paso 3:

- Ejemplo **negativo**:
  $< \mathit{Lluvia}; \mathit{Fria}; \mathit{Alta}; \mathit{Fuerte}; \mathit{Templada}; \mathit{Cambio} >$
- Nada que eliminar de S2.
- Especializaciones minimales de G2 que son mas generales que la hipotesis de S2:
  $< \mathit{Sol}; ?; ?; ?; ?; ? >$, $<?; \mathit{Templ}; ?; ?; ?; ? >$ y
  $<?; ?; ?; ?; ?; \mathit{Igual} >$.
- Luego:
  - $S3 = \{< \mathit{Sol}; \mathit{Templ}; ?; \mathit{Fuerte}; \mathit{Templ}; \mathit{Igual} >\}$
  - $G3 = \{< \mathit{Sol}; ?; ?; ?; ?; ? >, <?; \mathit{Templ}; ?; ?; ?; ? >$ y
    $<?; ?; ?; ?; ?; \mathit{Igual} >\}$

Ejemplo

Paso 4:

- Ejemplo **positivo**:
  $< Sol; Templ; Alta; Fuerte; Fria; Cambio >$
- Eliminamos de G3 la hipótesis: $<?; ?; ?; ?; ?; Igual >$
- Generalización minimal de S3:
  $< Sol; Templ; ?; Fuerte; ?; ? >$.
- Luego:
  - $S4 = \{< Sol; Templ; ?; Fuerte; ?; ? >\}$
  - $G4 = \{< Sol; ?; ?; ?; ?; ? >, <?; Templ; ?; ?; ?; ? >\}$

Propiedades

Sean S y G obtenidos por eliminación de candidatos

- Si S y G son no vacíos, resultan ser respectivamente la cota específica y cota general del espacio de versiones (respecto del conjunto de entrenamiento)
- Si S = G = $\{h\}$, entonces h es la única hipótesis de H consistente con todos los ejemplos
- Si S = G = $\emptyset$;, no existe $h \in H$ consistente con los ejemplos

Propiedades

Convergencia hacia el concepto objetivo, siempre que:

- Conjunto de entrenamiento suficientemente grande
- Ejemplos sin errores (ausencia de ruido)
- El concepto objetivo esta en H