Aprendizaje de conceptos. El aprendizaje como generalización

Aprendizaje de conceptos

- Consideraremos el problema de inferir automáticamente la definición general de un concepto o clase a partir de un conjunto de ejemplos que pueden ser o no miembros de esa clase.
- Denominaremos a esta tarea *Aprendizaje de conceptos (Concept learning)* como la inferencia de una función booleana (es decir, me dirá si/no ese ejemplo queda correctamente clasificado por el concepto general) a partir de ejemplos de entrenamiento, ya sean como salida o como entrada.

Aprendizaje de conceptos

- Por ejemplo, queremos saber qué representa el concepto definido por la frase "Días en los que mi amigo Aldo disfruta jugando al tenis" . La tabla 2.1 describe un conjunto de ejemplos (tanto positivos como negativos), cada uno representado por un conjunto de atributos. Uno de estos atributos es juega, y la tarea es inferir el valor de este atributo para un día cualquiera (futuro o pasado) a partir del conocimiento de estos ejemplos.
- ¿Es decir, qué concepto genérico clasifica todos o el mayor número posible de ejemplos?
- Para cada atributo la hipótesis puede ser:
  - Una ? si cualquier valor es aceptable para ese atributo
  - Un valor concreto (por ejemplo, *fría*)
  - Un – si no hay ningún valor aceptable.

Aprendizaje de conceptos

- Para cada atributo la hipótesis puede ser:
  - Una ? si cualquier valor es aceptable para ese atributo
  - Un valor concreto (por ejemplo, *fría*)
  - Un – si no hay ningún valor aceptable.
- Así la hipótesis más general (que cualquier día es un ejemplo positivo) se representa por (?, ?, ?, ?, ?)
- Y la hipótesis posible más específica (que ningún día es un ejemplo positivo) se representa por (-, -, -, -, -)

Aprendizaje de conceptos

Ejemplo	Cielo	Temperatura	Humedad	Viento	Agua	¿Juega al tenis?
1	Soleado	Caliente	Normal	Fuerte	Caliente	SI
2	Soleado	Caliente	Alta	Fuerte	Caliente	SI
3	Soleado	Fría	Alta	Fuerte	Caliente	NO
4	Soleado	Caliente	Alta	Fuerte	Fría	SI

Aprendizaje de conceptos

- Notación:
  - Al conjunto de items sobre los cuales se define el concepto se llama el conjunto de ejemplos (instancias), y se denota por X
    - En el ejemplo anterior X es el conjunto de todos los días representados por sus atributos: cielo, humedad...
  - Al concepto o función que debe ser aprendido se le llama concepto objetivo (target concept), y se denotas por c.
    - En general c puede ser una función booleana definida sobre los ejemplos X; esto es, c: X -> {0,1}
      - En este ejemplo, el concepto objetivo se corresponde con el valor del atributo ¿juega al tenis?, es decir, c(x)=1 si ¿juega al tenis?=SI, y c(x)=0 si ¿juega al tenis?=NO.

Aprendizaje de conceptos

- Conjunto de entrenamiento D: son todos los x de X con sus respectivos c(x).
  - Los ejemplos para los cuales c(x)=1 se llaman ejemplos positivos, mientras que los ejemplos para los cuales c(x)=0 se llaman ejemplos negativos.
  - Describiremos así los ejemplos como un par <x, c(x)>, y a su conjunto de todos los ejemplos de entrenemiento lo representaremos por la letra D.

Aprendizaje de conceptos

- Dado un conjunto de entrenemiento el problema es encontrar la hipótesis h (o el concepto objetivo c) que clasifique esos ejemplos.
  - Llamaremos H al conjunto de todas esas hipótesis h. En general cada hipótesis h en H representa una función booleana definida sobre X, es decir, h: X-> {0,1}.
  - El objetivo del sistema que aprende es encontrar una hipótesis h tal que h(x)=c(x) para todo x en X.

Aprendizaje de conceptos

- Hipótesis de aprendizaje inductivo: *cualquier hipótesis encontrada que clasifique un número suficientemente grande de ejemplos de entrenamiento clasificará otros ejemplos no observados.*

Aprendizaje de conceptos como un problema de búsqueda

- El aprendizaje de conceptos puede verse como un problema de búsqueda en un espacio de estados (donde cada estado es una hipótesis).
- El objetivo de esta búsqueda es encontrar la hipótesis que mejor clasifica el conjunto de ejemplos de entrenamiento.
- Vamos a ver algunos algoritmos para realizar esta búsqueda.

Ordenación del espacio de hipótesis

- Sean h_j y h_k dos hipótesis. Diremos que h_j *es_más_general_o_igual_que* h_k (escrito $h_j \geq h_k$) si hay más ejemplos que satisfacen h_j que h_k. Es decir, si

- Analogamente se define el más general estricto y se representa por $>_g$
- Y lo mismo ocurre con la relación *más_específico_ que*.

Algoritmo Find-S

- Se trata de encontrar la hipótesis más especifíca (en el sentido ya visto) que clasifique el mayor número de ejemplos.
- Empieza con la hipótesis más específica y según se va aplicando a cada uno de los ejemplos se va generalizando para intentar clasificarlos todos.
  - En el ejemplo:
    1. <soleado, caliente, normal, fuerte, caliente> para el ejemplo 1
    2. <soleado, caliente, ?, fuerte, caliente> para el ejemplo 2
    3. <soleado, caliente, ?, fuerte, caliente> para el ejemplo 3 (notar que es un ejemplo negativo, por lo que lo ignora,aunque de hecho lo que hace es clasificarlo como negativo)
    4. <soleado, caliente, ?, fuerte, ?> para el ejemplo 4

Algoritmo Find-S

- Algoritmo Find-S
  - 1. Inicializa *h* a la hipótesis más específica.
  - 2. Para cada ejemplo de entrenamiento *x*
    - 2.1 Si *x* es negativo, no hacer nada.
    - 2.2 Si *x* es positivo, para cada restricción de atributo *ai* en *h*
  - **IF** la restricción *ai* es satisfecha por *x*
    - **THEN** no hacer nada.
    - **ELSE** reemplazar *ai* en *h* por la siguiente restricción más general que es satisfecha por x.
  - 3. Hipótesis de salida h.

- El algoritmo Find-S no trata los ejemplos negativos

Algoritmo Find-S

- Find-S garantiza encontrar la hipótesis h en H más específica que es consistente con todos los ejemplos de entrenamiento positivos (y negativos)
- Cuestiones sin responder:
  - No me dice si es o no la única (es decir, si hay más hipótesis igual de específicas)
  - Como ignora los ejemplos negativos no es sensible al ruido (ejemplos inconsistentes en algunos de sus atributos)
  - No esta claro por qué es mejor la hipótesis más específica, es decir, por qué debo preferir la más específica sobre la más general.

El algoritmo list_then_eliminate

1. VersionSpace <- a list containing every hypothesis in H
2. For each training example, <x,c(x)> remove from VersionSpace any hypothesis h for which h(x)#c(x)
3. Output the list of hypotheses in VersionSpace

Es decir, va de lo más general a lo más particular

Espacio de Versiones y el algoritmo de *eliminación_de_candidatos*

- Este algoritmo nos dará el conjunto de todas las hipótesis que son consistentes con los ejemplos de entrenamiento, es decir:
  - Def: una hipótesis h es consistente con un conjunto de ejemplos de entrenamiento D si y solo si h(x)=c(x) para cada ejemplo <x, c(x)> en D.

- Notése la diferencia clave entre esta definición de *consistente* y la anterior de *satisface*. Un ejemplo se dice que satisface la hipótesis h cuando h(x)=1, independientemente de si x es un ejemplo positivo o negativo, mientra que sea consistente dependerá del concepto objetivo, y debe clasificar todos los ejemplos.

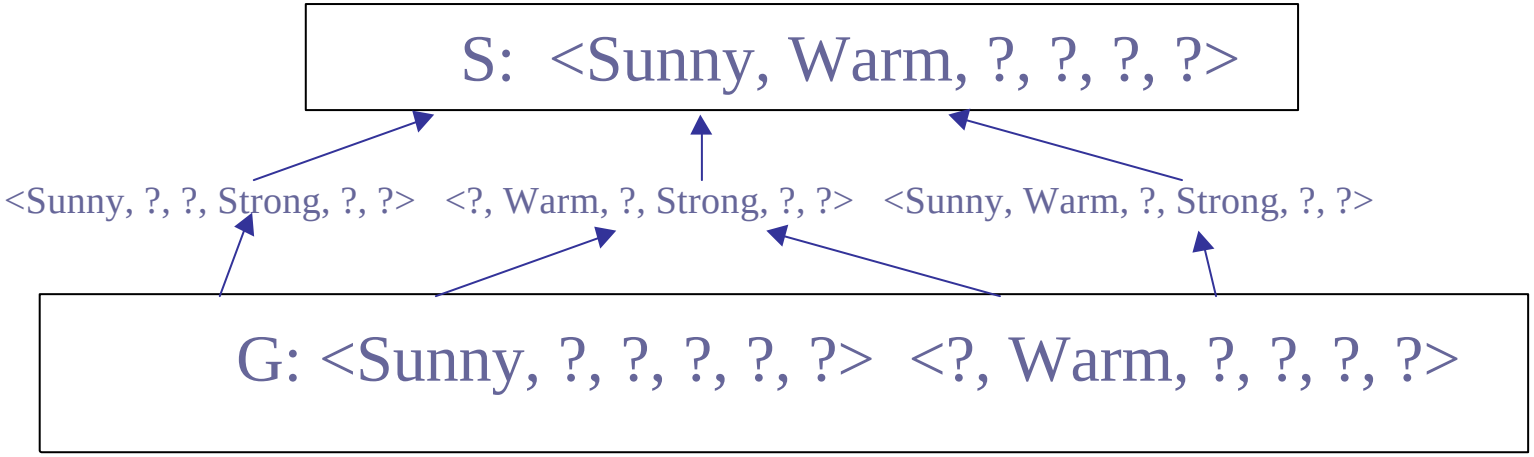
Espacio de Versiones y el algoritmo de *eliminación_de_candidatos*

- El algoritmo de *eliminación_de_candidatos* nos dará el conjunto de todas las hipótesis que son consistentes con los ejemplos de entrenamiento. Este subconjunto de H se denomina Espacio de Versiones $VS_{H,D}$
- Empieza con el Espacio de Versiones que contiene todas las hipótesis de H y luego elimina cualquier hipótesis consistente con cualquier ejemplo.

Representar el Espacio de Versiones

- El límite general G del espacio de versiones V_{SH} en D es el conjunto de todos las hipótesis más generales
- El límite específico S del espacio de versiones V_{SH} en D es el conjunto de todos las hipótesis más específicas
- Todo miembro del espacio de versiones se encuentra entre estos dos límites

Ejemplo de espacio de versiones



- De las 6 hipótesis elijo la más general (que coincide con el que me da el Find-S) y las más específicas que clasifiquen todos los ejemplos de la tabla

Espacio de Versiones y el algoritmo de *eliminación_de_candidatos*

- [1] Inicialización
  - [1.1] Inicializar *G* al conjunto de hipótesis generales maximales de H.
  - [1.2] Inicializar *S* al conjunto de hipótesis específicas maximales de H.
- [2] Para cada ejemplo de entrenamiento *d* .*D* hacer:
  - [2.1] Si *d* es un ejemplo positivo
    - [2.1.1] Eliminar de *G* cualquier hipótesis inconsistente con d.
    - [2.1.2] Para cada hipótesis *s* .*S* que no sea consistente con *d*
      - [2.1.2.1] Eliminar *s* de S.
      - [2.1.2.2] Añadir a *S* todas las generalizaciones minimales *h* de *s* tales que *h* sea consistente con *d* y algún miembro de *G* sea más general que h.
    - [2.1.3] Eliminar de *S* cualquier hipótesis que sea más general que otra hipótesis de S.

Espacio de Versiones y el algoritmo de *eliminación_de_candidatos*

- [2.2] Si *d* es un ejemplo negativo:
  - [2.2.1] Eliminar de *S* cualquier hipótesis inconsistente con d.
  - [2.2.2] Para cada hipótesis *g* .*G* que no sea consistente con *d*
    - [2.2.2.1] Eliminar *g* de G.
    - [2.2.2.2] Añadir a *G* todas las especializaciones más específicas *h* de *g* tales que *h* sea consistente con *d* y algún miembro de *S* es más específico que h.
    - [2.2.3] Eliminar de *G* cualquier hipótesis que sea menos general que otra hipótesis de G.
- [2.3] Si al procesar *d* se tiene que *S* = G, fin del algoritmo *(S* ó *G* es el concepto aprendido).