Aprendizaje Conceptual.

Inferir un concepto - funcion booleana - a partir de ejemplos.

Se define el concepto objetivo como una función booleana desconocida $f : X \rightarrow \{0, 1\}$, donde X es el conjunto de instancias o ejemplos.

En la clase vimos el ejemplo de pedro, es estudiante, y tiene un examen, y el concepto es si aprueba o no el examen basado en un vector de características.

- $x = (dedicacion, dificultad, horario, humedad, humor_docente)$
- $f(x) = 1$ $ si aprueba, 0 si no aprueba

El objetivo es encontrar una hipótesis $h : X \rightarrow \{0, 1\}$ que aproxime a f.

H representa una conjuncion de restricciones sobre los atributos de x. x es una n-upla de valores de atributos:

- Valor especifico: alto, bajo, medio
- todo es aceptable: ?
- ningun valor es aceptable: null

Bajo esta representacion de valores sabemos que: - $h(x) = 1$ si x satisface todas las restricciones - $h(x) = 0$ si x no satisface alguna restriccion - [?, ?, ?, ?, ?] siempre es 1 - [null, null, null, null, null] siempre es 0

Hay hipotesis que no se pueden representar con esta representacion, por ejemplo: pedro aprueba si la dedicacion es alta o la dificultad es baja.

Vamos a suponer para estas clases que todas las hipotesis se pueden representar con esta representacion.

Objetivo General

De todas las hipotesis posibles, encontrar la que mejor aproxime a f.

- Dominio de instancias X
- Funcion objetivo $f : X \rightarrow \{0, 1\}$
- Espacio de Hipotesis $h : X \rightarrow \{0, 1\}$
- Conjunto de entrenamiento $D = \{(x1, f(x1)), ...\}$

Definicion

- La unica informacion que tenemos de c son los valores en el conjunto de entrenamiento.
- Nada nos garantiza que el concepto objetivo pertenezca al espacio de hipotesis.
- Toda hipótesis que aproxime correctamente el concepto objetivo en un conjunto de ejemplos lo suficientemente grande también lo hará sobre las instancias aún no observadas.

El problema de aprendizaje visto desde otro enfoque.

Este problema se puede ver como un problema de busqueda en un espacio de hipotesis.

Si tomamos cada combinacion del espacio de hipotesis, en el ejemplo de pedro, tendriamos 5 * 5 * 4 * 5 * 4 hipotesis sintacticamente distintas. Evaluar esto es equivalente a un problema de conjuntos, la categoria de estos problemas es NP-Completo.

Nota: Con MMC se puede resolver en tiempo polinomial obtener una estimacion de la cardinalidad del espacio de hipotesis que cumplen f(x) = 1.

Se necesitan estrategias para reducir el espacio de busqueda:

- Mas general o igual: hj, hk, hj >= hk si hk(x) = 1 implica que hj(x) = 1
- Mas especifica o igual: hj, hk, hj <= hk si hj(x) = 1 implica que hk(x) = 1 o hk(x) = 0 implica que hj(x) = 0

De esta froma se puede definir una relacion de orden sobre el espacio de hipotesis. donde.

- [null, null, null, null, null] es la menos general
- [?, ?, ?, ?, ?] es la hipotesis mas general

Find-S

Algoritmo de busqueda en el espacio de hipotesis, que empieza desde la hipotesis mas especifica [null, null, null, null, null] y va generalizando hasta encontrar una hipotesis que cumpla con todos los ejemplos.

para esto se van incorporando las tuplas de datps del conjunto de entrenamiento a la hipotesis.

Mientras la hipotesis sea consistente con los ejemplos se mantiene, al momento de encontrar un ejemplo que no cumpla con la hipotesis se generaliza.

Algoritmo

```
h0 = hipotesis mas especifica
para cada instancia positiva x
  para cada restriccion r en h0
    si r no satisface x
      reemplazar r por la restriccion mas general que si satisface x
devolver h0
```

En este ejemplo tenemos una unica hipotesis mas especifica, en otros casos puede haber mas de una.

Candidate Elimination

- **hipotesis consistente**: una hipotesis h es consistente con un conjunto de entrenamiento D si y solo si h(x) = f(x) para todo x en D.
- **espacio de versiones**: conjunto de hipotesis consistentes con D.

El espacio de versiones representa a todas las hipotesis que son candidatas a ser la hipotesis objetivo, dado un conjunto de entrenamiento.

El metodo de eliminacion de candidatos es un algoritmo que permite encontrar el espacio de versiones.

List-Then-Eliminate

```
VS = conjunto con todas las hipotesis
para cada ejemplo [x, f(x)]
  eliminar todas las h tq h(x) != f(x)
Devolver VS
```

Enumera todo el espacio de hipotesis y elimina las que no son consistentes con el conjunto de entrenamiento. En espacios H infinitos no es posible.

Candidate-Elimination Algorithm

Representamos el espacio de versiones de versiones con las hipotesis consistentes mas especificas y mas generales.

- **Limite general G**: g pertenece a H, es consistente y no existe g' tq g' > g y g' es consistente.
- **Limite especifico S**: s pertenece a H, es consistente y no existe s' tq s > s' y s' es consistente.

Bajo esta definicion de limites, VS son todas las hipotesis h tq s <= h <= g. s y g son hipotesis pertenecientes a S y G respectivamente.

```
S = conjunto de hipotesis especificas
G = conjunto de hipotesis generales

para cada instancia x del conjunto de entrenamiento.
  si f(x) = 1
    remover de G cualquier hipotesis inconsistente con x
    para cada hipotesis s de S incosistente con x
      cambiarla por todas las generalizaciones minimas de s, consistente con x, para las cua
      remover de S cualquier hipotesis mas general que otra de S
  si f(x) = 0
    remover de S cualquier hipotesis inconsistente con x
    para cada hipotesis g de G incosistente con x
      cambiarla por todas las especializaciones minimas de g, consistente con x, para las cu
      remover de G cualquier hipotesis mas especifica que otra de G
```

El calculo de S es una generalizacion de Find-S. Generalizar y especificar depende fuertemente del espacio de trabajo H elegido.

- El agloritmo converge a una hipotesis correcta cuando no existe ruido en los datos de entrenamiento o cuando el concepto objetivo pertenece al espacio de hipotesis.

Repaso y conclusiones

- Si todo mi conjunto de hipotesis positivas(generales) evaluan false para un ejemplo, entonces ese ejemplo es negativo.
- Analogamente, si todo mi conjunto de hipotesis negativas(especificas) evaluan true para un ejemplo, entonces ese ejemplo es positivo.

Sesgo Inductivo

1. **Sesgo Inductivo**: Para que un algoritmo de aprendizaje pueda generalizar más allá de los datos de entrenamiento, debe tener un conjunto de suposiciones previas, conocido como sesgo inductivo. Sin este sesgo, el algoritmo no podría hacer predicciones sobre datos no vistos.
2. **Algoritmo CANDIDATE-ELIMINATION**: Este algoritmo convergerá al concepto objetivo real si el concepto objetivo está dentro de su espacio de hipótesis y los ejemplos de entrenamiento son precisos. Sin embargo, si el concepto objetivo no está en el espacio de hipótesis, el algoritmo fallará.
3. **Espacio de Hipótesis Sesgado vs. No Sesgado**:
   - Un espacio de hipótesis sesgado restringe las posibles hipótesis, lo que puede llevar a perder el concepto objetivo. Por ejemplo, si el espacio solo considera conjunciones de atributos, podría perder conceptos disyuntivos.
   - Un espacio de hipótesis no sesgado, que incluye todas las hipótesis posibles, asegura que el concepto objetivo es expresable. Sin embargo, esto puede llevar a un sobreajuste, donde el algoritmo no puede generalizar más allá de los ejemplos observados.
4. **Futilidad del Aprendizaje Sin Sesgo**: Sin ningún sesgo inductivo, un algoritmo de aprendizaje no puede hacer clasificaciones racionales para instancias no vistas. La única forma en que puede clasificar es si ha visto la instancia exacta antes.
5. **Definición de Sesgo Inductivo**: El sesgo inductivo de un algoritmo de aprendizaje es el conjunto de suposiciones que, combinadas con los datos de entrenamiento, permiten que las clasificaciones del algoritmo se infieran deductivamente. Para el algoritmo CANDIDATE-ELIMINATION,

su sesgo inductivo es la suposición de que el concepto objetivo está contenido dentro de su espacio de hipótesis dado.

6. **Comparación de Algoritmos de Aprendizaje por Sesgo**:
   - **ROTE-LEARNER**: Almacena cada ejemplo de entrenamiento y clasifica nuevas instancias mediante búsqueda en memoria. No tiene sesgo inductivo.
   - **CANDIDATE-ELIMINATION**: Clasifica nuevas instancias solo si todas las hipótesis en su espacio de versión están de acuerdo con la clasificación. Su sesgo es que el concepto objetivo es representable en su espacio de hipótesis.
   - **FIND-S**: Encuentra la hipótesis más específica consistente con ejemplos de entrenamiento y la usa para clasificar todas las instancias posteriores. Asume que todas las instancias son negativas a menos que se demuestre lo contrario.
7. **Fuerza del Sesgo Inductivo**: Los métodos con un sesgo más fuerte pueden clasificar una mayor proporción de instancias no vistas. Sin embargo, la corrección de dichas clasificaciones depende de la precisión del sesgo.

En esencia, el sesgo inductivo es crucial para que cualquier algoritmo de aprendizaje automático haga predicciones significativas sobre nuevos datos no vistos. Sin él, el algoritmo podría ajustarse demasiado a los datos de entrenamiento o ser incapaz de hacer cualquier predicción.