Aprendizaje Automático

Javier Béjar

Inteligencia Artificial - 2021/2022 1Q

CS - GEI- FIB

Introducción

- Las técnicas que hemos visto hasta ahora nos permiten crear sistemas que resuelven tareas que necesitan inteligencia
- La limitación de estos sistemas reside en que sólo resuelven los problemas ya previstos
- Sólo podemos considerar que un sistema es realmente inteligente si es capaz de observar su entorno y aprender de él
- La autentica inteligencia reside en adaptarse, tener capacidad de integrar nuevo conocimiento, resolver nuevos problemas, aprender de errores

- No se pretende modelar el aprendizaje humano
- Busca aumentar las capacidades de los programas de IA (SBC, planificación, TLN, búsqueda, ...):
  - Su límite está en el conocimiento que se les ha introducido
  - No resuelven problemas mas allá de esos límites
- Es imposible prever todos los problemas desde el principio
- Buscamos dar a programas la capacidad de adaptarse sin tener que ser reprogramados

Does Machine Learning Really Work? Tom Mitchell. AI Magazine 1997

¿Donde y para que se puede usar el aprendizaje automático?

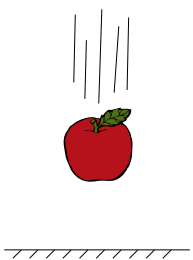
- Tareas difíciles de programar (adquisición de conocimiento, reconocimiento de caras, voz, ...)
- Aplicaciones auto adaptables (interfaces inteligentes, spam filters, sistemas recomendadores, ...)
- Minería de datos/Descubrimiento de conocimiento (análisis de datos inteligente)

- **Aprendizaje inductivo:** Creamos modelos de conceptos a partir de *generalizar* ejemplos simples. Buscamos patrones comunes que expliquen los ejemplos.
- **Aprendizaje analítico o deductivo:** Aplicamos la deducción para obtener descripciones generales a partir de un ejemplo de concepto y su explicación.
- **Aprendizaje genético:** Aplica algoritmos inspirados en la teoría de la evolución para encontrar descripciones generales a conjuntos de ejemplos.
- **Aprendizaje conexionista:** Busca descripciones generales mediante el uso de la capacidad de adaptación de redes de neuronas artificiales

Aprendizaje inductivo

- Los métodos más utilizados en aplicaciones provienen del aprendizaje inductivo supervisado
- **Inducción**: Pasamos de lo específico a lo general
- **Supervisión**: Conocemos el concepto al que pertenece cada ejemplo
- A partir de un conjunto de ejemplos etiquetados obtenemos un modelo
- El modelo generaliza los ejemplos, representando los conceptos que definen las etiquetas
- Obtenemos lo que es común entre los ejemplos de un concepto que les diferencia de los otros

- Desde el punto de vista formal lo que se obtiene mediante aprendizaje inductivo no es válido
- Asumimos que un número limitado de ejemplos representa las características del concepto que queremos aprender
- Solo hace falta un contraejemplo para invalidar el resultado
- Pero, !una gran parte del aprendizaje humano es inductivo!

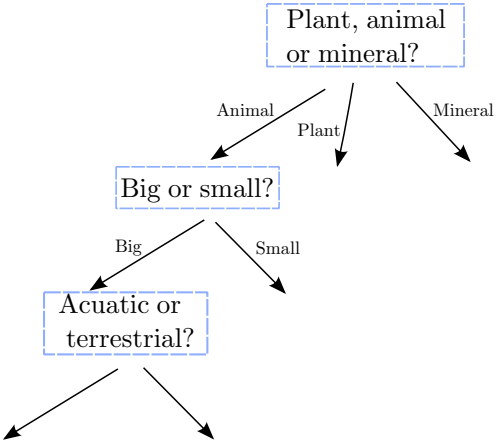


$$F = G\frac{m_1 \cdot m_2}{r^2}$$

- Tipos de métodos de aprendizaje inductivo supervisado
  - Modelos caja blanca (podemos inspeccionar el modelo)
    - Árboles de decisión/reglas de inducción
    - Modelos probabilísticos
  - Modelos caja negra
    - Redes de neuronas artificiales
    - Máquinas de soporte vectorial
- Podemos plantear el problema como:
  - *Clasificación*: un conjunto finito de conceptos
  - *Regresión*: una función continua

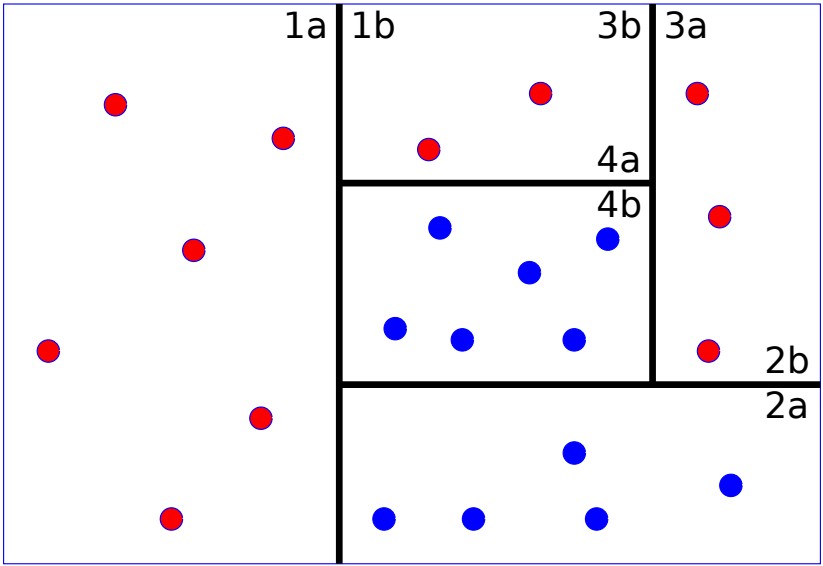
Árboles de decisión

- Podemos aprender un concepto como la aplicación de un algoritmo que busca el conjunto de preguntas que lo distingue de otros
- El árbol de preguntas nos sirve de lenguaje de representación, cada nodo es un test sobre un atributo
- Esta representacion es equivalente a una FND (2^{2^n} conceptos)
- Para aprender hemos de realizar una búsqueda en el espacio de árboles de preguntas



- Buscar en el espacio de todas las FND es demasiado costoso
- Para reducir el coste computacional imponemos un *sesgo* (las descripciones que se prefieren)
- **Restricción:** Queremos el árbol que represente la mínima descripción del concepto objetivo dados los ejemplos
- **Justificacion:** Este árbol será el mejor clasificando nuevos ejemplos (la probabilidad de tener preguntas innecesarias se reduce)
- **Navaja de Occam**: "*En igualdad de condiciones, la explicación más sencilla suele ser la correcta*"

- Existen muchos algoritmos para aprender árboles de decisión
- El más simple es el denominado **ID3**
- Este algoritmo realiza una búsqueda por ascenso en el espacio de árboles
  - Para cada nuevo nodo de decisión un atributo es elegido y los ejemplos son distribuidos según sus valores
  - Este procedimiento es repetido recursivamente hasta que todos los ejemplos son del mismo concepto
- La selección de cada atributo es decidida mediante una función heurística que tiene preferencia a formar árboles mínimos


1a
1b
3b
3a
4a
4b
2b
2a

- Una heurística es un método aproximado para la solución de un problema
- Las heurísticas para árboles de decisión miden lo adecuado que es un atributo para formar un árbol mínimo
- Esta decisión se realiza de manera local (en cada nodo del árbol) aproximando el problema global
- Heurísticas utilizadas:
  - Entropía, entropía normalizada
  - GINI index
  - Rough sets
  - ...

- La teoría de la información estudia entre otras cosas los mecanismos de codificación de mensajes y el coste de su transmisión
- Si definimos un conjunto de mensajes $M = \{m_1, m_2, ..., m_n\}$, cada uno de ellos con una probabilidad $P(m_i)$, podemos definir la cantidad de información (I) contenida en un mensaje de M como:

$$I(M) = \sum_{i=1}^{n} -P(m_i)log(P(m_i))$$

- Este valor se puede interpretar como la información necesaria para distinguir entre los mensajes de M (Cuantos bits de información son necesarios para codificarlos)

- Podemos hacer la analogía con la codificación de mensajes suponiendo que las clases son los mensajes y la proporción de ejemplos de cada clase su probabilidad
- Podemos ver un árbol de decisión como la codificación que permite distinguir entre las diferentes clases
  (Aprender un árbol de decisión $\Longleftrightarrow$ Aprender un código)
- Buscamos el mínimo código que distingue entre las clases
- Cada atributo se deberá evaluar para decidir si se le incluye en el código

- En cada nivel del árbol debemos evaluar qué atributo permite minimizar el código
- Este atributo será el que haga que la cantidad de información que quede por cubrir sea la menor (bits restantes por codificar)
- La elección de un atributo debería resultar en una partición donde los ejemplos en cada una de ellas estén sesgados hacia una clase
- Necesitamos una medida de la cantidad de información que no cubre un atributo (Entropia, E)

- Dado un conjunto de ejemplos $\mathcal{X}$ y siendo $\mathcal{C}$ su clasificación

$$I(\mathcal{X},\mathcal{C}) = \sum_{\forall c_i \in \mathcal{C}} -\frac{\sharp c_i}{\sharp \mathcal{X}} \, log(\frac{\sharp c_i}{\sharp \mathcal{X}})$$

- Bits necesarios para codificar los ejemplos sin ninguna información adicional

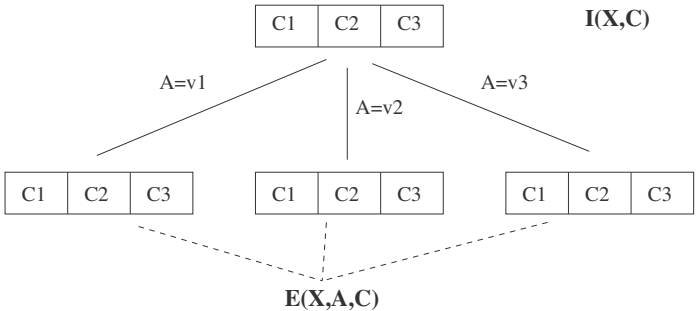
- Dado un atributo A y siendo $[A(x) = v_i]$ los ejemplos con valor v_i en el atributo

$$E(\mathcal{X}, A, \mathcal{C}) = \sum_{\forall v_i \in A} \frac{\sharp[A(x) = v_i]}{\sharp\mathcal{X}} I([A(x) = v_i], \mathcal{C})$$

- Bits necesarios para codificar los ejemplos dado un atributo
  (Simplemente la suma ponderada de la cantidad de información de cada partición)

- Bits que aún han de ser codificados

$$G(\mathcal{X}, A, \mathcal{C}) = I(\mathcal{X}, \mathcal{C}) - E(\mathcal{X}, A, \mathcal{C})$$



G(X,A,C)= I(X,C) – E(X,A,C)

Algorithm: ID3 ($\mathcal{X}$: Ejemplos, $\mathcal{C}$: Clasificación, $\mathcal{A}$: Atributos)

if *todos los ejemplos son de la misma clase*
then
    return *una hoja con el nombre de la clase*
else
    Calcular la cantidad de información de los ejemplos (**I**)
    foreach *atribute en* $\mathcal{A}$ **do**
        Calcular la entropia (**E**) y la ganancia de información (**G**)
    Escoger el atributo que maximiza **G** (**a**)
    Borrar **a** de la lista de atributos ($\mathcal{A}$)
    Generar el nodo raíz para el atributo **a**
    foreach *partición generada por los valores del atributo* **a** **do**
        Árbol$_i$=ID3(ejemplos de $\mathcal{X}$ con **a**=**v**$_i$, clasificación, resto de atributos)
        generar una nueva rama con **a**=**v**$_i$ y Árbol$_i$
    return *El nodo raíz para* ***a***

Tomemos el siguiente conjunto de ejemplos de películas

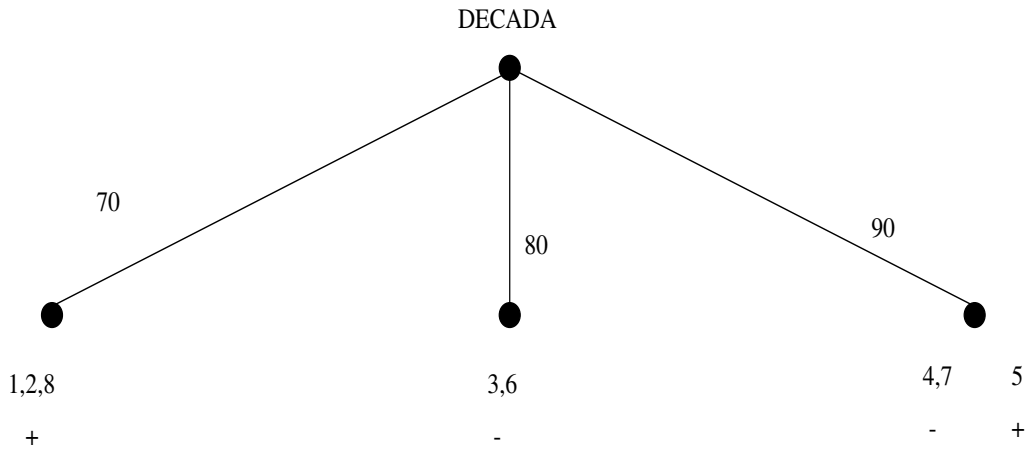
Ej.	Década	País	Género	Gusta
1	70	USA	Drama	+
2	70	no USA	Comedia	+
3	80	no USA	Drama	−
4	90	no USA	Drama	−
5	90	no USA	Comedia	+
6	80	no USA	Acción	−
7	90	USA	Acción	−
8	70	no USA	Drama	+

$$
\begin{aligned}
I(X, C) &= -1/2 \cdot log(1/2) - 1/2 \cdot log(1/2) = 1 \\
E(X, decada) &= (70)\ 3/8 \cdot (-1 \cdot log(1) - 0 \cdot log(0)) \\
&+ (80)\ 2/8 \cdot (-1 \cdot log(1) - 0 \cdot log(0)) \\
&+ (90)\ 3/8 \cdot (-1/3 \cdot log(1/3) - 2/3 \cdot log(2/3)) \\
&= 0{,}344 \\
E(X, pais) &= (USA)\ 2/8 \cdot (-1/2 \cdot log(1/2) - 1/2 \cdot log(1/2)) \\
&+ (noUSA)\ 6/8 \cdot (-1/2 \cdot log(1/2) - 1/2 \cdot log(1/2)) \\
&= 1 \\
E(X, genero) &= (comedia)\ 2/8 \cdot (-1 \cdot log(1) - 0 \cdot log(0)) \\
&+ (drama)\ 4/8 \cdot (-1/2 \cdot log(1/2) - 1/2 \cdot log(1/2)) \\
&+ (accion)\ 2/8 \cdot (0 \cdot log(0) - 1 \cdot log(1)) \\
&= 0{,}5
\end{aligned}
$$

Como podemos comprobar, es el atributo **década** el que maximiza la función.

$$
\begin{aligned}
G(X, decada) &= 1 - 0{,}344 = 0{,}656 * \\
G(X, pais) &= 1 - 1 = 0 \\
G(X, genero) &= 1 - 0{,}5 = 0{,}5
\end{aligned}
$$

Este atributo nos genera una partición que forma el primer nivel del árbol.



Ahora solo en el nodo correspondiente al valor **90s** tenemos mezclados objetos de las dos clases, por lo que repetimos el proceso con esos objetos.

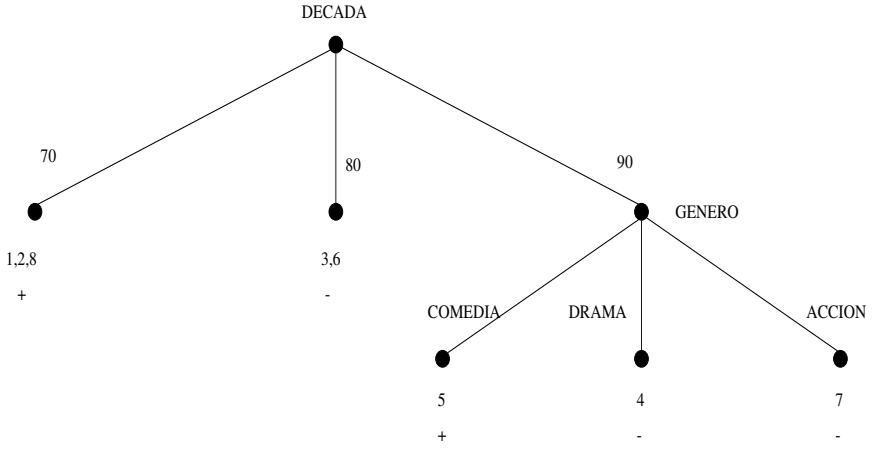
Ej.	País	Género	Gusta
4	no USA	drama	$-$
5	no USA	comedia	$+$
7	USA	acción	$-$

$$
\begin{aligned}
I(X,C) &= -1/3 \cdot log(1/3) - 2/3 \cdot log(2/3) = 0{,}918 \\
E(X,pais) &= (USA)\ 1/3 \cdot (0 \cdot log(0) - 1 \cdot log(1)) \\
&+ (noUSA)\ 2/3 \cdot (-1/2 \cdot log(1/2) - 1/2 \cdot log(1/2)) \\
&= 0{,}666 \\
E(X,genero) &= (comedia)\ 1/3 \cdot (0log(0) - 1 \cdot log(1)) \\
&+ (drama)\ 1/3 \cdot (-1 \cdot log(1) - 0 \cdot log(0)) \\
&+ (accion)\ 1/3 \cdot (0 \cdot log(0) - 1 \cdot log(1)) \\
&= 0
\end{aligned}
$$

Ahora el atributo que maximiza la función es **género**.

$$\begin{aligned} G(X, pais) &= 0{,}918 - 0{,}666 = 0{,}252 \\ G(X, genero) &= 0{,}918 - 0 = 0{,}918* \end{aligned}$$

El árbol resultante es ya totalmente discriminante y podemos usarlo como descripción del perfil



Aprendizaje Bayesiano

- Los modelos basados en árboles de decisión o reglas asumen que hay una división clara entre conceptos (solo una respuesta correcta)
- Para algunos problemas es más interesante tener decisiones difusas (soft)
- Esto se puede modelar usando distribuciones de probabilidad para representar los conceptos

- El elemento principal del aprendizaje bayesiano es el teorema de Bayes
- Este teorema liga hipótesis con observaciones

$$P(h|\mathcal{E}) = \frac{P(h) \cdot P(\mathcal{E}|h)}{P(\mathcal{E})}$$

- Esto significa que si tenemos una muestra de nuestro problema podemos evaluar nuestra hipótesis (la clasificación de nuestros datos) calculando un conjunto de probabilidades

- Supongamos que queremos recomendar una novela a un amigo, esta decisión se basará en nuestra opinión acerca de la novela
- Sabemos que la probabilidad a priori de que a nuestro amigo le guste una novela es del 60 % ($p(h)$)
- Sabemos que nuestro amigo tiene un gusto similar al nuestro ($P(\mathcal{E}|h)$)
- Esa suposición es nuestro modelo de la tarea, con los siguientes parámetros estimados:
  - La probabilidad de tener nosotros una opinión positiva cuando nuestro amigo tiene una opinión positiva es del 90 %
  - La probabilidad de tener nosotros una opinión negativa cuando nuestro amigo tiene una opinión negativa es del 95 %
- A nosotros nos ha gustado la novela ¿deberíamos recomendársela a nuestro amigo? (¿cual sería su predicción?) ($P(h|\mathcal{E})$)

Si enumeramos todas las probabilidades:

- P(Amigo) = $\langle 0{,}60; 0{,}40 \rangle$ (pos/neg)
- P(Nuestra | Amigo=pos) = $\langle 0{,}9; 0{,}1 \rangle$ (pos/neg)
- P(Nuestra | Amigo=neg) = $\langle 0{,}05; 0{,}95 \rangle$ (pos/neg)

El teorema de bayes nos dice:

$$P(Amigo|Nuestra) = \frac{P(Amigo) \cdot P(Nuestra|Amigo)}{P(Nuestra)}$$

Dado que nuestra opinión es positiva (los datos) y dado que el resultado ha de sumar 1 para ser una probabilidad:

$$\begin{aligned} P(Amigo|Nuestra = pos) &= \langle P(A = pos) \cdot P(N = pos|A = pos), \\ &\quad P(A = neg) \cdot P(N = pos|A = neg)\rangle \\ &= \langle 0{,}6 \times 0{,}9; 0{,}4 \times 0{,}05\rangle \\ &= \langle 0{,}94; 0{,}06\rangle \quad (normalizada) \end{aligned}$$

Esto quiere decir que es muy probable que a nuestro amigo le vaya a gustar la novela

- Podemos aplicar el aprendizaje bayesiano para hacer clasificación de diferentes maneras
- Considerando solo la clase con la mayor probabilidad a posteriori (*maximum a posteriori hypothesis*, h_{MAP})

$$P(h|\mathcal{E}) = \underset{h \in \mathcal{H}}{\operatorname{argmax}} P(h) \cdot P(\mathcal{E}|h)$$

- Es exactamente lo mismo que hacemos en la estimación de las redes bayesianas

- El objetivo de aprendizaje es estimar la *función de densidad de probabilidad* (FDP) de los datos
- Para estimar una FDP debemos hacer ciertas suposiciones sobre:
  - El modelo de distribución que describe los atributos (continuos, discretos)
  - El modelo de distribución que describe las hipótesis
  - La dependencia entre las variables (todas independientes, algunas independientes, ...)

- La aproximación más simple es asumir que todos los atributos son independientes (no es cierto en general)
- La FDP para los atributos de los datos se puede expresar como:

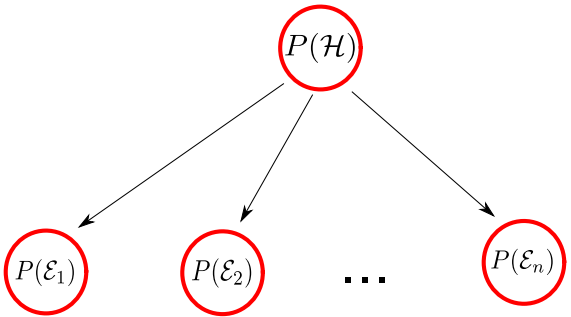
$$P(\mathcal{E}|h) = \prod_{\forall i \in attr} P(\mathcal{E}_i|h)$$

- La hipótesis maximum a posteriori puede transformarse en:

$$P(h|\mathcal{E}) = \underset{h \in \mathcal{H}}{\operatorname{argmax}} \left[P(h) \times \prod_{\forall i \in attr} P(\mathcal{E}_i|h) \right]$$

- Podemos estimar separadamente cada $P(\mathcal{E}_i|h)$ a partir de los datos

Este modelo es equivalente a la red bayesiana:



$$P(\mathcal{H}|\mathcal{E}) = P(\mathcal{H}) \times P(\mathcal{E}_1|\mathcal{H}) \times P(\mathcal{E}_2|\mathcal{H}) \cdots \times P(\mathcal{E}_n|\mathcal{H})$$

Algorithm: Naive Bayes

Entrada: $\mathcal{E}$ ejemplos, $\mathcal{A}$ atributos, $\mathcal{H}$ hipótesis/clases
Salida : $P(\mathcal{H})$,$P(\mathcal{E}_{\mathcal{A}}|\mathcal{H})$
foreach $h \in \mathcal{H}$ **do**
    $P(h) \leftarrow$ Estimar la probabilidad a priori de la clase ($\mathcal{E}$,h)
    foreach $a \in \mathcal{A}$ **do**
        $P(\mathcal{E}_a|h) \leftarrow$ Estimar la FDP del atributo de la clase ($\mathcal{E}$, h, a)

- Para predecir nuevos ejemplos solo hemos de calcular la probabilidad de la hipótesis
- Sorprendentemente esta aproximación funciona en algunos dominios mejor que métodos más complejos

- Para atributos discretos podemos estimar $P(\mathcal{E}_i|h)$ como la frecuencia de los valores del atributo del conjunto de datos para cada clase (distribución multinomial)
- Para atributos continuos podemos estimar $P(\mathcal{E}_i|h)$ asumiendo que los datos provienen de una distribución continua (por ejemplo gausiana) y estimar los parámetros de la distribución a partir de los datos
- Existen técnicas más avanzadas de estimación de probabilidad que no asumen una distribución de probabilidad específica (no paramétricas) y pueden evaluar una probabilidad a partir los datos mediante una estimación local

Ej.	Década	País	Género	Gusta
1	70	USA	Drama	+
2	70	no USA	Comedia	+
3	80	no USA	Drama	−
4	90	no USA	Drama	−
5	90	no USA	Comedia	+
6	80	no USA	Acción	−
7	90	USA	Acción	−
8	70	no USA	Drama	+

Década			País		Género			
70	80	90	USA	noUSA	Comedia	Drama	Acción	Gusta
1	0	.33	.5	.5	1	.5	0	+ (.5)
0	1	.66	.5	.5	0	.5	1	- (.5)

Ej: (90, USA, Drama)

$$\underset{h\in\{+,-\}}{\text{argmax}}\langle 0{,}5 \times 0{,}33 \times 0{,}5 \times 0{,}5, 0{,}5 \times 0{,}66 \times 0{,}5 \times 0{,}5\rangle =$$

$$\underset{h\in\{+,-\}}{\text{argmax}}\langle 0{,}33, 0{,}66\rangle = 0{,}66 \Rightarrow -$$