**Concept learning and Version space**

Sommario

[Introduzione 2](#_Toc57817499)

[Obiettivo dell’apprendimento 2](#_Toc57817500)

[Apprendimento di un concetto 5](#_Toc57817501)

[VC dimension 8](#_Toc57817502)

[Ipotesi generali e specifiche 8](#_Toc57817503)

[Algoritmo Find-S 9](#_Toc57817504)

[Algoritmo List-Then Eliminate 11](#_Toc57817505)

[Version Space 11](#_Toc57817506)

[Algoritmo Candidate Elimination 12](#_Toc57817507)

[Richiesta di query 13](#_Toc57817508)

[Bias induttivo 15](#_Toc57817509)

Lezione 6

PS. Lo stesso task può essere risolto da più modelli, ma anche lo stesso modello può essere usato per risolvere diversi task (es. alberi si usano per classificazione, regressione, …)

PS. Il bias ci aiuterà a generalizzare il nostro apprendimento a partire dagli esempi dati

Min 11

# Introduzione

## Obiettivo dell’apprendimento

L’apprendimento è un task in cui noi partiamo dalle descrizioni degli esempi fornite dentro un insieme di dati (data set) e cerchiamo di apprendere un modello che ci dia una risposta rispetto a quegli esempi. L’obiettivo dipende dal task di cui stiamo parlando (classificazione, regressione, …)

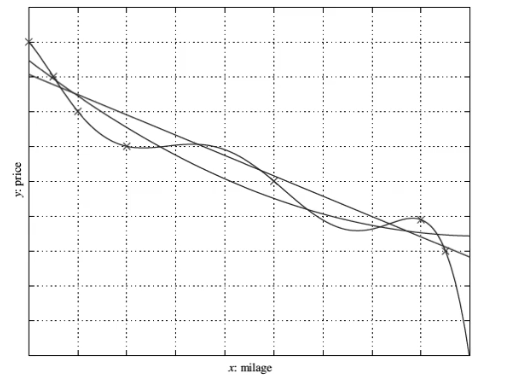
Quindi si vorrà indurre un modello generale (ad esempio una funzione matematica), a cui viene dato una variabile di input, a restituirne una di output.

Le funzioni più semplici ottenute con dei polinomi con grado più basso sono quelle con bias maggiore, perché vincolano maggiormente l’andamento della funzione e sono costruiti su un numero minore di parametri (ovvero i pesi del polinomio).

Con polinomi di grado maggiore la funzione si adegua maggiormente agli esempi ma su futuri dati hanno una minore capacità di generalizzare.

Ricapitolando l’obiettivo è quello di costruire un modello generale in grado di rappresentare il concetto ignoto su futuri esempi a partire da specifici esempi. Il bias nel caso di una funzione è rappresentato dal grado del polinomio

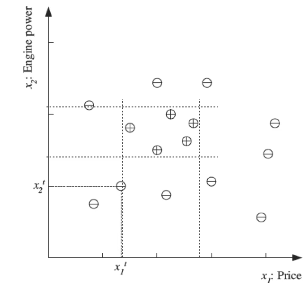
Di seguito abbiamo un esempio di **task** **di** **regressione**. Si vuole predire un valore continuo. È un task semplice con sole due feature (price e milage, cioè numero di km della macchina). Dati i km vogliamo predire il prezzo di vendita della macchina. I pallini sono gli esempi nel dataset



Nel caso di una funzione la x è l’input del modello e la y è l’output. Per creare il modello andremo a creare un polinomio (bisognerà decidere il grado). Nel grafico ci sono diversi polinomi, quello lineare è il più facile e generale, quello di quinto grado è invece quello più specifico. Il polinomio migliore probabilmente è quello di terzo grado. Per vedere se il modello predice correttamente ci baseremo sull’errore che commette (occhio a overfitting e underfitting)

Si andrà a dividere il dataset in training set e test set.

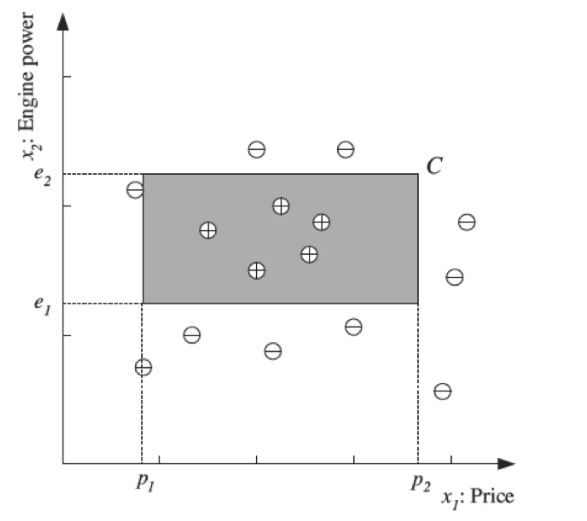
Vediamo ora un **task di classificazione** (binaria).



L’esempio è sulle macchine familiari, i punti con il segno + indicano le auto che fanno parte della categoria macchine familiari. Ognuno di questi esempi è rappresentato attraverso due caratteristiche (due features), una delle quali è il prezzo dell’auto (x1) e la seconda è la potenza del motore (x2). La terza caratteristica è l’etichetta data

Qual è l’assunzione che noi facciamo alla base degli algoritmi induttivi? Noi cerchiamo di descrivere il concetto ignoto (ad esempio la categoria delle auto familiari) attraverso delle ipotesi e cerchiamo di approssimare sempre di più la funzione target usando i dati forniti dagli esempi di training.

Nell’esempio sopra un’ipotesi possibile può essere rappresentata attraverso due intervalli di valori. Se il prezzo appartiene tra x1’ e la seconda linea e se la potenza del motore sta tra la seconda e la terza linea partendo dal basso, allora il rettangolo che si ottiene nel grafico tracciando le linee contiene le macchine familiari (o almeno la maggior parte). Queste ipotesi sono state apprese dagli esempi di training. Se il training set fosse ben rappresentativo allora anche su esempi futuri i vincoli dovrebbero essere validi.



C è il rettangolo grigio ed è l’ipotesi fatta. Se il rettangolo grigio includesse il rettangolo sottostante allora si tratterebbe di aver fatto delle ipotesi troppo generali perché includono all’interno anche esempi della classe negativa e quindi non abbiamo appreso bene i dati.

Quindi

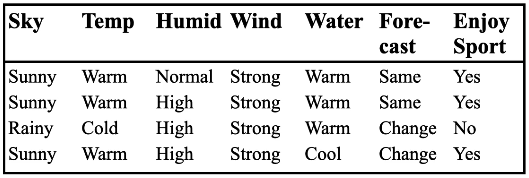
* Nel primo esempio (task di regressione) le ipotesi si riferiscono al grado del polinomio
* Nel secondo esempio (task di classificazione) le ipotesi si riferiscono agli intervalli di valori delle due feature

Qualunque sia l’ipotesi vogliamo che permetta di approssimare il nostro output al meglio. Per capirlo basterà valutare come lavora il modello (quindi le sue performance predittive) sul test set.

Se parliamo di apprendimento di concetto, in questo caso, il concetto potrebbe essere una descrizione riassuntiva delle caratteristiche delle auto definita in base alla feature del prezzo e della feature della potenza del motore, tale per cui quelle macchine che soddisferanno le ipotesi faranno parte della categoria auto familiari (l’apprendimento permetterà di generare l’ipotesi)

# Apprendimento di un concetto

Vediamo un esempio. Si ipotizza di apprendere un concetto ignoto che rappresenta le giornate nella quale l’amico Aldo adora fare sport d’acqua (le etichette Enjoy sport cambiano in base all’amico che scelgo). Noi non sappiamo in quali giorni Aldo fa sport, ma cerchiamo di capirlo dal meteo. Descriviamo questi esempi attraverso delle caratteristiche descrittive del meteo. Le caratteristiche descrittive del meteo sono discrete, tramite stringhe descrittive che rappresentano delle situazioni (ad esempio vento forte o debole, soleggiato o no).



Features

* Sky: feature di tipo categorico che può assumere i valori di {soleggiato, piovoso, nuvoloso}
* Temp: feature di tipo numerico discretizzata a valori categorici {calda, fredda} (oltre una certa soglia sarà calda)
* Humid: feature di tipo numerico discretizzata a valori categorici {normale, alta}
* Wind: feature di tipo numerico discretizzata a valori categorici
* Water: feature di tipo numerico discretizzata a valori categorici
* Forecast: feature categorica che ci dice se le condizioni metereologiche sono cambiate rispetto al giorno precedente o meno {uguali, cambiate}, ciò per capire se le condizioni sono variabili o stabili

Dalla seguente tabella. In quali giornate Aldo ha preferito fare sport? Ce lo dice l’ultima colonna della tabella. Il task è quello di riuscire a capire i vincoli sugli attributi descrittivi del meteo che ci permettono di capire in quali giornate Enjoy sport è Yes.

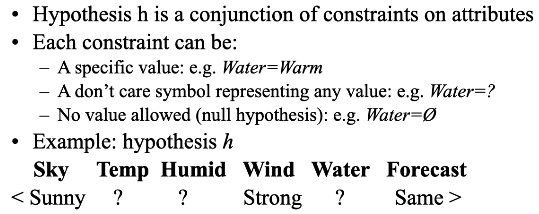
Come rappresentiamo l’ipotesi? Abbiamo una collezione di variabili categoriche a valori discreti. Ci viene bene usare dei **predicati logici** **del prim’ordine**. Quindi useremo un operatore di uguaglianza e confronteremo il valore della feature con uno dei suoi valori del dominio (es. Water = Warm è true se per un certo esempio l’uguaglianza è soddisfatta). Se invece non mi interessa il valore di quella caratteristica uso ? (ad esempio Water=?). Se nessun valore è ammesso per quella caratteristica uso l’insieme vuoto (Water= Ø). Con Water = ? si ha sempre true, con Water = Ø si ha sempre false. Sono i due estremi di tutto lo spettro possibile.

Il fatto di scegliere solo tre tipologie di predicati

* Feature = tipo feature
* Feature = ?
* Feature = Ø

Introduce un bias (è nostro compito scegliere i predicati migliori, non per forza bisogna mettere questi 3).

Vediamo un ipotesi su queste feature descrittive degli esempi:



h è una possibile ipotesi. Questa ipotesi se soddisfatta ci da output Yes.

Si tratta di un linguaggio di tipo **congiuntivo** perché tutti i valori di h devono essere soddisfatti contemporaneamente per far sì che Aldo faccia sport quel giorno.

Cerchiamo di capire qual è il task dell’apprendimento dei concetti.

Noi partiamo dall’insieme delle **istanze X**, descritto attraverso gli attributi Sky, Temp, Humidity, Wind, Water, Forecast.

Avremo uno spazio a 7 dimensioni (le 7 feature) di tipo categorico, ed è lo spazio X degli esempi, cioè gli esempi si troveranno all’interno di questo spazio.

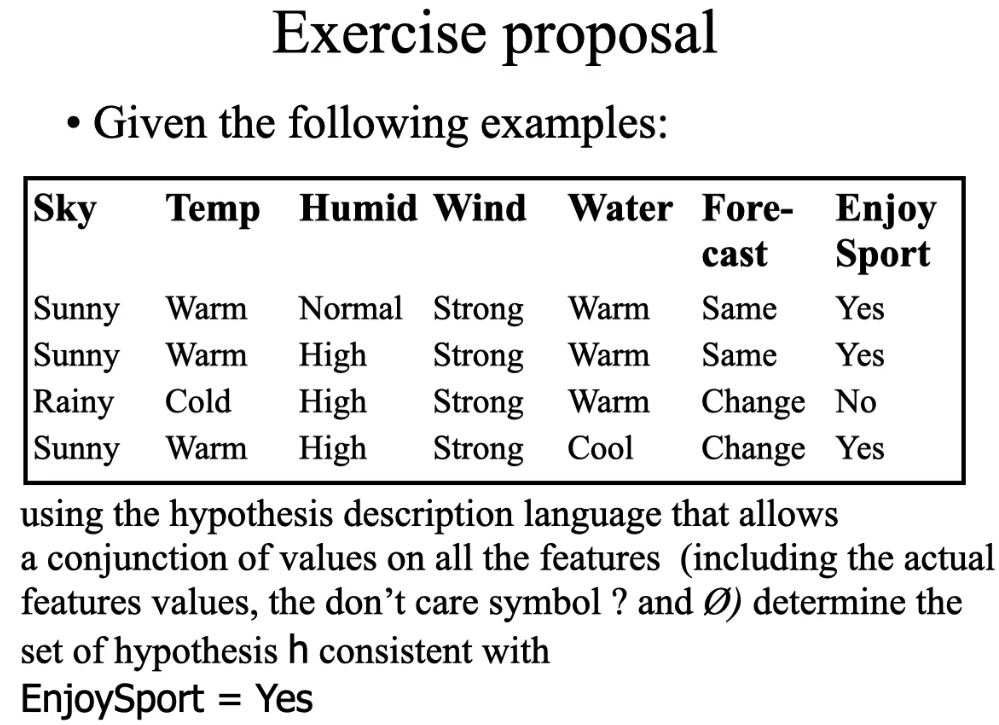
Abbiamo la **funzione target** **c** che vogliamo apprendere, ovvero il concetto ignoto viene descritto attraverso l’attributo c: EnjoySport X 🡪 {0,1}, associa il valore 0 o 1 a ciascun esempio dello spazio delle istanze X.

Lo **spazio delle ipotesi H** è dato da un linguaggio di tipo congiuntivo, i predicati delle ipotesi vengono messi in and tra di loro (<Sunny ? ? Strong ? Same> è un’ipotesi e per essere verificata tutti i predicati devono essere true contemporaneamente)

Abbiamo poi un insieme di **esempi di training D** e ognuno degli esempi è descritto da un vettore xi contenente le caratteristiche descrittive Sky, Temp,… e per qualcuno di questi esempi viene associato il valore della funzione target c. Quindi avremo D:<x1,c(x1)>, …,<xn,c(xn)>

PS. Non faremo ipotesi di indipendenza tra le feature (come vedremo nel modello naive-bayes). Le ipotesi da noi fatte sono quelle dei tipi di uguaglianza e del fatto di mettere in congiunzione i predicati delle ipotesi

L’obiettivo è determinare quelle ipotesi h nello spazio delle ipotesi H tale che h(x) = c(x) per ogni x in D. se h è vera per tutto il training set quando Aldo vuole fare sport allora l’ipotesi è corretta e descrive il concetto ignoto.



Ci possono essere più ipotesi h consistenti con i dati

1. <Sunny Warm ? Strong ? ?>, la più specifica (aggiunge il maggior numero di vincoli alle tre feature)
2. <Sunny Warm ? ? ? ?> <Sunny ? ? Strong ? ?> <? Warm ? Strong ? ?>, meno specifica di quella precedente (tra di loro è difficile dire chi è più generale tra queste 3)
3. <Sunny ? ? ? ? ? > <? Warm ? ? ? ?> l’ipotesi più generale di tutte

L’ideale è trovare una ipotesi unica che ci permetta di raggiungere il nostro task; quindi, avere una descrizione univoca che ci garantisce di riconoscere solo gli esempi positivi lasciando fuori quelli negativi.

L’ipotesi (1) è coerente con il nostro dataset. Quindi se è soddisfatta su tutti gli esempi l’ipotesi più specifica allora saranno soddisfatte anche quelle più generali.

Non abbiamo elementi per escludere alcuna ipotesi, abbiamo bisogno di più dati.

Quindi quello che bisogna andare a fare è

* Prendere un primo esempio e costruire un’ipotesi che sia vera per quell’esempio (avrò un ipotesi specifica solo per quell’esempio)
* Successivamente guardando gli altri esempi vado a rilassare (generalizzare) l’ipotesi

E se l’ipotesi da noi appresa è soddisfatta da un esempio negativo? Non possiamo generalizzare in quel caso perché andrebbe comunque a soddisfare l’esempio negativo. Non possiamo neanche specificare l’ipotesi (quindi tornando indietro di un passo) perché la generalizzazione ci era servita per coprire ulteriori esempi positivi. L’unica cosa da fare è **droppare l’ipotesi** andando a costruirne delle altre considerando gli esempi negativi. Se non riusciamo ad arrivare ad un’ipotesi valida vuol dire che ci servono più dati

PS. Se ci accorgiamo che una delle ipotesi è troppo restrittiva, quindi non generale, possiamo rilassare l’ipotesi sostituendo uno dei valori con ?.

Lezione 7

Ora affrontiamo la dimensione dello spazio degli esempi con la dimensione dello spazio delle ipotesi. La dimensione dello spazio delle ipotesi cresce in maniera esponenziale nel numero delle caratteristiche che utilizziamo per descrivere gli esempi. Nel nostro esempio

1. Sky: Sunny, Cloudy, Rainy
2. AirTemp: Warm, Cold
3. Humidity: Normal, High
4. Wind: Strong, Weak
5. Water: Warm, Cold
6. Forecast: Same, Change

PS. Siccome il dataset è consistente non esistono esempi con valori di features uguali e target diverso.

Il numero dei possibili esempi che possiamo rappresentare è 3\*2\*2\*2\*2\*2 = 96 (che sono il tipo di giornata)

Quanti sono i concetti diversi che potrei apprendere? Sono 2^96, 2 perché è un problema di classificazione binario, 96 invece sono le giornate (noi abbiamo visto solo il concetto di Aldo, ma il problema potrebbe essere espanso).

Noi abbiamo scelto un linguaggio di tipo congiuntivo con determinati valori per le feature.

Quanto è il numero delle possibili ipotesi sintatticamente rappresentabili? Aggiungendo al conto del numero di valori per ogni feature anche ? e insieme vuoto allora diventa 5\*4\*4\*4\*4\*4 = 5120

Equivale alla produttoria della cardinalità di ogni feature (con aggiunta di Ø e ?).

In realtà l’insieme vuoto è utilizzabile in una ipotesi soltanto, l’ipotesi iniziale che facciamo (perché poi dallo start in poi andremo solo a rendere più specifica l’ipotesi, non più generale), quando ancora nessun valore è specificato, e quindi c’è solo una ipotesi con l’insieme vuoto. Perciò il numero delle ipotesi semanticamente distinte è 1 + 4\*3\*3\*3\*3\*3 = 973

Noi possiamo scrivere 973 ipotesi al massimo, non è detto che riusciremo a esprimere ogni concetto (perché sono 2^96)

L’apprendimento dei concetti è possibile, apprenderli vuol dire apprendere l’insieme delle ipotesi espresse nel nostro linguaggio consistenti con gli esempi forniti da cui apprendere. Il target concept viene espresso negli esempi attraverso un valore booleano.

Le 973 ipotesi possibili le andremo a restringere al numero delle ipotesi consistenti con gli esempi dati.

Andremo a vedere un algoritmo che esplora lo **spazio delle ipotesi**, ne vedremo uno che non esplora tutte le 973 ipotesi.

## VC dimension

(si tratta di un argomento utile ma non presente nel libro. Ha a che vedere con il design dei linguaggi di rappresentazione delle ipotesi).

Questo problema ha a che vedere con le **Vapink-Chervonenkis (VC) dimension**. È una teoria matematica per misurare la capacità espressiva di un modello di adattarsi con flessibilità alla complessità di un problema. Lo faccio introducendo parametri che introducono dei vincoli nel linguaggio di rappresentazione delle ipotesi e quindi permettono all’algoritmo di apprendere un modello consistente con gli esempi dati. La VC dimension è la cardinalità, il numero più grande degli esempi che l’algoritmo di apprendimento è in grado di spezzettare (shatter) lo spazio degli esempi in porzioni distinte che siano omogenee rispetto al valore della funzione. Se sono capace di distinguere queste aree ho imparato a capire il concetto perché so che in quelle aree ci sono etichette uniche.

Si tratta del massimo numero di esempi (indipendentemente dall’etichetta) tale che un modello sia in grado di catturare tutti i possibili concetti su certi esempi

La VC dimension ci dice quanto può essere complesso il problema dato dal numero massimo di punti che l’algoritmo è in grado di associare all’etichetta. Più è alto il valore più sarà complesso il problema che il modello riesce ad affrontare correttamente

## Da ipotesi generali a ipotesi specifiche

Ora facciamo ordine nello spazio delle ipotesi. Per apprendere un concetto dobbiamo navigare nello spazio delle ipotesi. Per farlo in modo facile ordiniamo le ipotesi. Faremo un ordinamento parziale.

Prendiamo l’esempio precedente

* h1 = <Sunny ? ? Strong ? ?>
* h2 = <Sunny ? ? ? ? ?>

Notiamo subito che h2 è più generale, teoricamente dovrebbe coprire più esempi.

Un’ipotesi meno vincoli mette e più generale è (quindi sarà più ampio l’insieme di esempi su cui sarà soddisfatta).

Definizione:

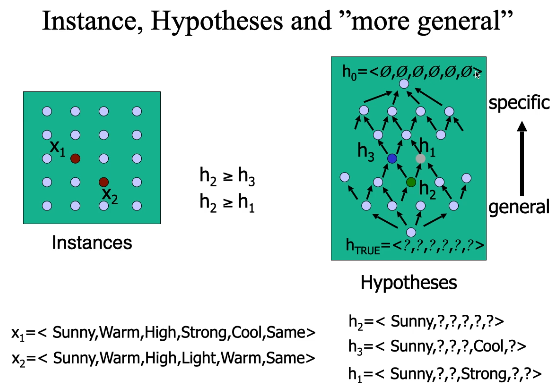
Date due generiche ipotesi hj e hk, ipotesi che possono essere soddisfatte da degli esempi nello spazio degli esempi X, allora possiamo dire che hj è più generale, o al massimo uguale ad hk (hj >= hk) sse per qualunque possibile esempio nello spazio degli esempi x se è vera l’ipotesi più specifica hk allora di sicuro sarà vera anche hj

Grazie a ciò riusciamo a creare un **ordine parziale** sullo spazio delle ipotesi H. la relazione è riflessiva, antisimmetrica e transitiva. È parziale perché possono esserci ipotesi se ipotesi che sono figlie dello stesso antenato non possono essere ordinate (saranno allo stesso livello)

La relazione di ordine parziale gode della relazione di

* riflessività (un ipotesi è uguale a se stessa)
* transitività
* antisimmetria (non è simmetrica e gode della proprietà di uguaglianza solo se l’ipotesi è uguale a se stessa)

Questa relazione di ordine parziale sullo spazio delle ipotesi viene utilizzata in modo utile da molti algoritmi di ordinamento.



Si va dal basso verso l’alto.

Le ipotesi più specifiche copriranno un numero più piccolo di esempi. L’ordine è parziale perché non tutto l’insieme può essere ordinato (l’ipotesi sorelle, come h3 e h1, non possono essere ordinate).

Introduciamo le assunzioni che facciamo per poter apprendere in maniera consistente delle ipotesi da degli esempi di partenza.

1. Assumiamo che lo spazio delle ipotesi H che è possibile navigare attraverso il linguaggio che usiamo per descrivere le ipotesi permetta di descrivere correttamente il concetto target c (se abbiamo scelto un numero di features sufficienti, un numero di valori sufficienti in un linguaggio che rappresenta correttamente il concetto)
2. Assumiamo che gli esempi di training non contengano errore, cioè che siano stati compilati correttamente e siano consistenti

## Algoritmi per navigare lo spazio delle ipotesi

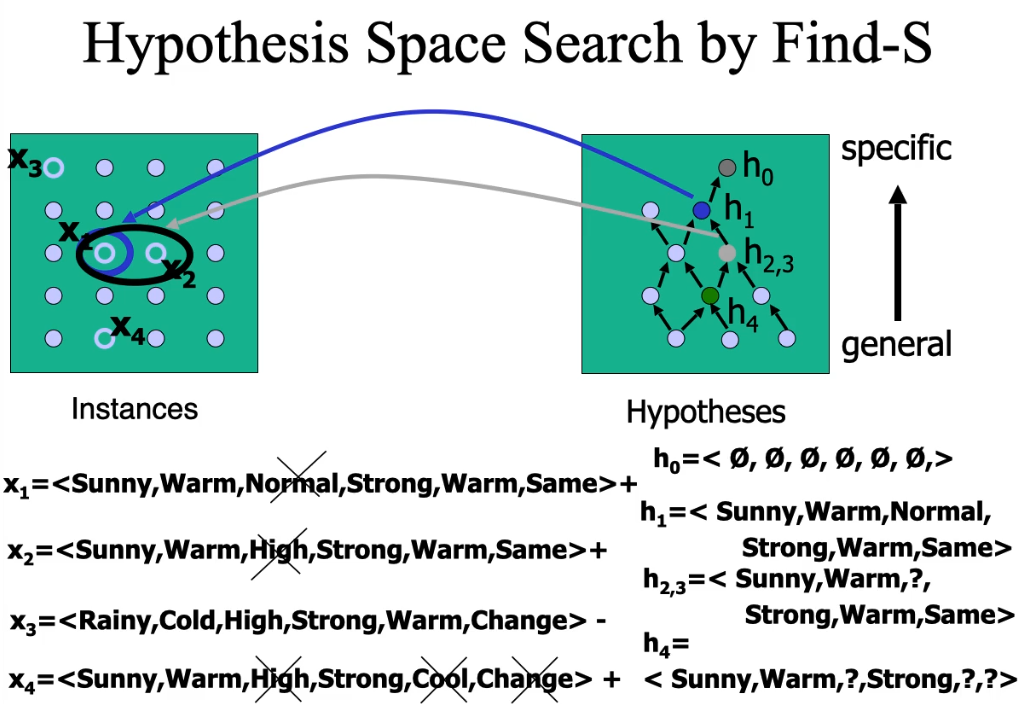
### Algoritmo Find-S

Vediamo l’algoritmo **Find-S**. Il termine S indica che l’algoritmo va alla ricerca dell’ipotesi specifica.

1. Inizializza l’ipotesi corrente h all’ipotesi più specifica di tutte nello spazio delle ipotesi (ovvero h0 = <Ø, Ø, Ø, Ø, Ø, Ø>
2. Per ogni esempio positivo x
   1. Per ogni vincolo ai nell’ipotesi corrente h si chiede se il vincolo è soddisfatto da x. Se lo è allora non fa nulla, altrimenti sostituisce il vincolo ai con un vincolo più generale per soddisfare x
3. Ritorna l’ipotesi h

Si concentra solo sugli esempi positivi e non sui negativi. Generalizza le ipotesi andando a trovare l’ipotesi più specifica possibile. Se il dataset contiene solo esempi negativi diremo che l’ipotesi è sempre falsa. Non va quindi molto bene. Si dice che l’**algoritmo è polarizzato verso gli esempi della classe positiva** (perché sa sole generalizzare le ipotesi)

Vediamo un esempio



Quello che ha fatto è detto **salto induttivo**, perché con l’ipotesi ottenuta coprirà anche esempi mai visti (chiaro che potrebbe non essere sempre corretta, ma ciò dipende dal numero di esempi)

Find-s ha alcune proprietà

* Utilizza delle ipotesi descritte da un linguaggio di tipo congiuntivo
* Da in output l’ipotesi più specifica nello spazio delle ipotesi H consistente con gli esempi positivi forniti
* L’ipotesi data in output è anche consistente con gli esempi negativi nell’ipotesi fatta in assunzione che il concetto target sia consistente con gli esempi forniti ed è esprimibile nel linguaggio dato ed è contenuto nello spazio delle ipotesi che stiamo navigando.

Limiti dell’algoritmo Find-s

* Non riusciamo a dire se l’algoritmo ha raggiunto una convergenza alla fine della propria esecuzione trovando una ipotesi che coincide con il concetto target, perché non siamo capaci di determinare se il concetto target coincideva con quell’ipotesi in output o era un concetto più generale contenuto nello spazio delle ipotesi H ma non avevamo ricevuto abbastanza esempi positivi per generalizzare ulteriormente l’ipotesi corrente. Quindi non siamo in grado di dire se l’ipotesi trovata è la singola ipotesi valida.
* Non riesce a riconoscere quando i dati sono inconsistenti perché ignora gli esempi negativi (idealmente si vorrebbe che l’algoritmo riconoscesse le inconsistenze e le correggesse)
* L’ipotesi più generale potenzialmente coprirebbe più esempi
* Cosa potrebbe fare l’algoritmo se come output potesse restituire un numero vario di ipotesi specifiche?

Successivamente vedremo una rappresentazione più evoluta dell’algoritmo chiamata Version space

Lezione 6

### Algoritmo List-Then Eliminate

Find-s da in output soltanto una delle ipotesi valide e consistenti con i dati, la più specifica. Visto che lo spazio delle ipotesi è ampio è stato ideato un algoritmo che eviti di navigare tutte le possibili alternative enumerandole.

È stato ideato uno spazio di rappresentazione delle ipotesi chiamato **version space**, quello spazio delle varianti delle ipotesi consistenti con il training set.

Prima di introdurre l’algoritmo version space introduciamo un algoritmo brute force **List-Then Eliminate** che enumera tutte le possibili alternative di ipotesi e verifica se sono consistenti, se no le elimina.

1. VersionSpace 🡨 contiene una lista contenente tutte le ipotesi contenute nello spazio delle ipotesi H
2. Verifica che per ogni esempio di training <x, c(x)> l’ipotesi sia consistente (h(x) = c(x)), altrimenti la rimuove
3. Ritorna il VersionSpace scremato

Il difetto di questo algoritmo è che la lista potrebbe essere troppo lunga. Dobbiamo trovare un algoritmo più intelligente.

### Version Space

Definiamo com’è costituito il Version Space. Si tratta di un insieme di ipotesi consistenti con il training set. Una ipotesi h è definita **consistente** con il training set D del concetto target sse h(x)=c(x) vero per ogni esempio <x ,c(x)> del training set. Quindi

***Consistent(h, D) := ∀ <x, c(x)> ∈ D | h(x)=c(x)***

Il **Version Space VSH,D** è sottoinsieme delle ipotesi H che risultano consistenti con D

***VSH,D = { h ∈ H | Consistent(h, D)}***

Per rappresentare il version space introduciamo

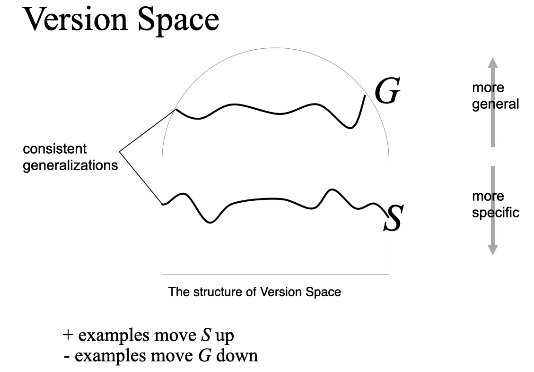
1. Il **bordo G**, o bordo generale, costituito dall’insieme delle ipotesi massivamente generali
2. Il **bordo S**, o bordo specifico, costituito dall’insieme delle ipotesi massimamente specifiche

***VSH,D = {h ∈ H | (∃ s ∈ S) (∃ g ∈ G) (g >= h >= s)}***

Dove x >= y significa che x è più generale oppure uguale a y.

Le ipotesi che stanno tra G ed S sono comunque valide. Ci basteranno bordo G ed S, le ipotesi in mezzo non devono essere esplicitate, perchè sappiamo che saranno comunque valide.

Tutte le ipotesi nel VS sono tutte ipotesi valide e candidate per essere l’ipotesi corretta. Avremo un insieme di ipotesi perché sicuramente mancheranno esempi che ci facciano arrivare alla singola ipotesi corretta.



L’obiettivo sarà portare S verso l’alto e G verso il basso, cosi da ridurre il numero di ipotesi valide

### Algoritmo Candidate Elimination

L’algoritmo fa uso di una versione compatta di rappresentazione delle ipotesi consistenti con gli esempi tramite l’uso di due bordi G ed S.

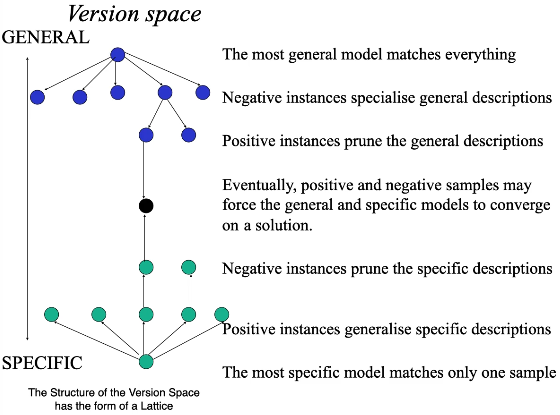
L’algoritmo guarderà sia esempi positivi che negativi e ad ogni esempio aggiornerà i bordi.

L’algoritmo ha due sezioni, la prima dedicata agli esempi positivi, la seconda dedicata a quelli negativi. Testa se l’esempio è positivo e se lo è aggiorna i 2 bordi e si chiede se l’insieme G contiene qualche ipotesi non consistente allora viene scartata. Il bordo S viene usato per generalizzare le ipotesi più specifiche, mentre il bordo G viene utilizzato per renderle più specifiche.

* G <- ipotesi più generale
* S <- ipotesi più specifica
* Per ogni esempio di training d = <x, c(x)>
  + Se d è un esempio positivo
    - Rimuove da G tutte le ipotesi inconsistenti con d.
    - Per ogni ipotesi s in S non consistente con d
      * Rimuove s da S
      * Aggiunge a S tutte le generalizzazioni minime h di s tale che
        + h sia consistente con d
        + Qualche membro di G sia più generale di h
      * Rimuove da S le ipotesi più generali di altre ipotesi in S
  + Se d è un esempio negativo (duale alla prima fase)
    - Rimuove da S tutte le ipotesi inconsistenti con d
    - Per ogni ipotesi g in G non consistente con d
      * Rimuove g da G
      * Aggiunge a G tutte le specializzazioni minime h di g tale che
        + h sia consistente con d
        + Qualche membro di S sia più specifico di h

(per fare la minima specializzazione di una ipotesi generale che soddisfa la seconda richiesta bisogna prendere le ipotesi in S e andare ad aggiungere un ? all’ipotesi e poi

* + - * Rimuove da G le ipotesi meno generali di altre ipotesi in G



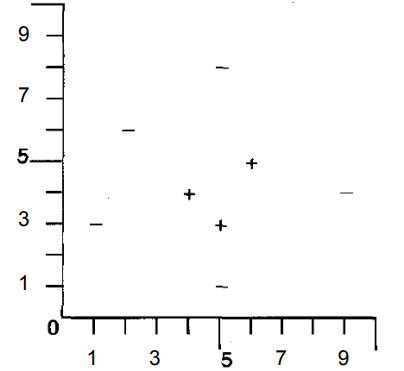
L’algoritmo convergerà allo stesso risultato del version space senza considerare l’odine nella quale gli esempi del training set sono stati presentati.

In caso non ci siano sufficienti esempi nel training set per apprendere un’unica ipotesi rappresentate il concetto, l’algoritmo potrebbe convergere a differenti version space a seconda del training set.

Vedi esempio nelle slide

PS. L’algoritmo converge alle stesse ipotesi indipendentemente dall’ordine degli esempi. Le ipotesi però possono cambiare cambiando gli esempi.

Lezione 8

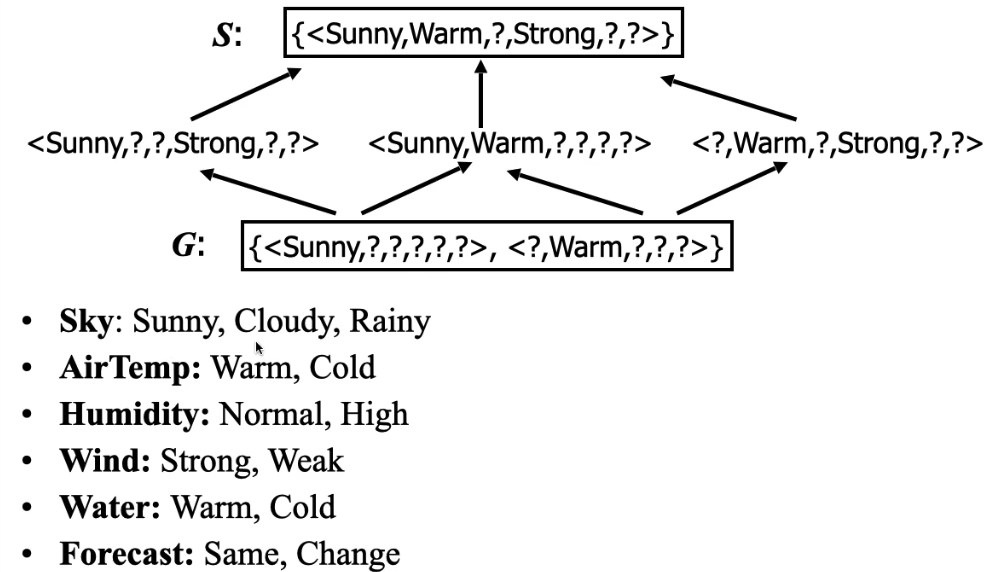


## Richiesta di query

Vediamo ora l’aspetto delle query per richiedere al teacher degli esempi per ridurre il version space ad un'unica ipotesi alla quale l’algoritmo completa il proprio apprendimento del concetto ignoto. Sta all’intelligenza trovare degli esempi opportuni. Se gli esempi non fossero sufficienti si avrebbe sempre più di una ipotesi.

Come capire quali esempi richiedere alla teacher per ridurre il version space? Potremmo pensare che qualunque esempio che chiediamo, se soddisfatto dall’ipotesi non ne elimina nessuna, altrimenti la elimina dal version space.

Qua sotto è rappresentato un reticolo (lattice in inglese)

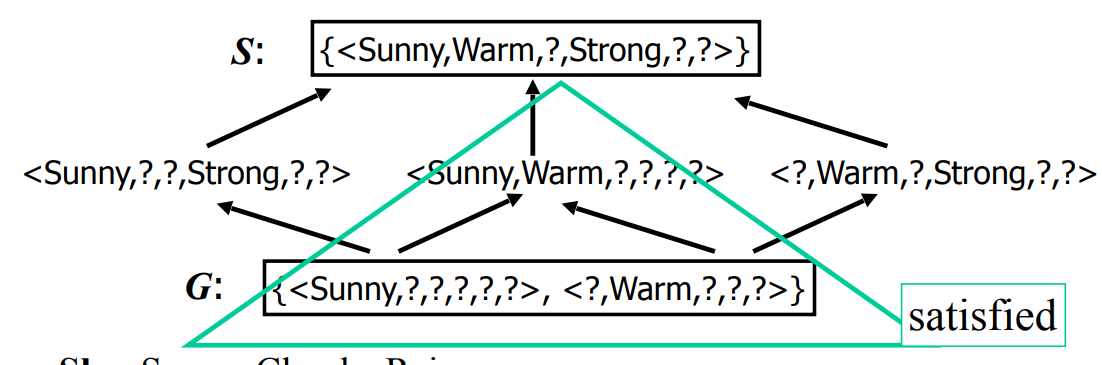


Dato il nostro esempio possiamo pensare di considerare le ipotesi in G + <Sunny, Warm, ?, ?, ?, ?> e pensare a degli esempi non ancora analizzati che soddisfino le tre ipotesi.

Ad esempio chiediamo

* <**Sunny**, **Warm**, Normal, **Weak**, Warm, Change>
* <**Sunny**, **Warm**, High, **Weak**, Cold, Same>

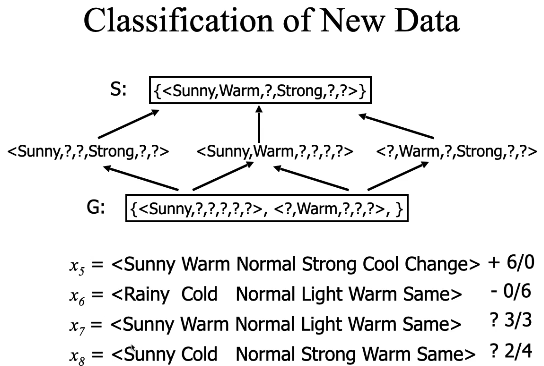
Che soddisfano i requisiti specificati. Se il primo o il secondo esempio fosse positivo andremo ad eliminare le ipotesi fuori dal triangolo.



Per generare query ci basiamo su quel triangolo e cerchiamo di vedere se gli esempi sono negativi. Se sono positivi tolgo le ipotesi fuori dal triangolo, se negativi tolgo quelle dentro il triangolo. Da 6 passiamo a 3 in ogni caso.

Con questa strategia possiamo pensare, nel caso più fortunato, di trovare sempre un esempio che ci permette di eliminare la metà delle ipotesi del version space e quindi il numero di passi sarà log2(|VS|), dove VS è il versione space di partenza (nell’esempio, quindi, 6).

Usiamo le ipotesi rimanenti sul VS per classificare esempi futuri



Cerchiamo di usare la nostra conoscenza per fare una previsione, quindi dato un esempio vediamo a che classe appartiene in base alle nostre ipotesi:

* X5 è della classe positiva
* X6 è della classe negativa
* X7 la soddisfano 3 ipotesi e le altre 3 invece non la soddisfano, positiva o negativa?
* X8 la soddisfano solo 2 ipotesi su 6, positiva o negativa?

X7 sarebbe un ottimo esempio di chiedere all’oracolo.

Abbiamo visto come è utile usare il version space per fare la previsione di nuovi esempi. Questo viene chiamato **salto induttivo**, o **inductive leap** (cioè data un’ipotesi più generale siamo in grado di coprire esempi che non abbiamo mai visto).

Cosi facendo stiamo introducendo un **bias** (pregiudizio, polarizzazione) per arrivare a una soluzione. I nostri bias sono

* Scelta del linguaggio congiuntivo
* Ipotesi che abbiamo

Vediamo un caso particolare dove al viene usato un linguaggio disgiuntivo

Immagine che contiene testo, persona, screenshot, documento

Descrizione generata automaticamente

(vedi rec)

## Unbiased learner

Vediamo ora se il nostro apprendimento possa essere guidato da un linguaggio senza bias, quindi nell’ipotesi che il nostro modello non sia vincolato dai vincoli di rappresentazione delle ipotesi. Questo vuol dire che il nostro spazio delle ipotesi H permette di rappresentare qualunque concetto mai ideabile attraverso l’uso della logica.

Noi abbiamo 96 possibili rappresentazioni di giornate, posso costruire 2^96 concetti, sicuramente rappresentabili con la logica.

L.H ci dice tutto ciò che possiamo usare per la rappresentazione dei concetti

Immagine che contiene testo

Descrizione generata automaticamente

Usando l’algoritmo Candidate Elimination, quale potrebbero essere i 2 insiemi S e G? Assumiamo tre esempi positivi (x1, x2 e x3) e due esempi negativi (x4 e x5). I bordi potrebbero essere

S : { (x1 V x2 V x3 ) } G : { -(x4 V x5) }

**Assumere che questi siano i due bordi permette di classificare soltanto gli esempi forniti senza avere la capacità di predizione di ulteriori esempi**. Ciò non ci permette di fare l’inductive leap. Avremo un training set coincidente con lo spazio degli esempi X. Se il training set contenesse un numero più ristretto di esempi vorrebbe dire che qualunque istanza non osservabile verrebbe classificata come positiva esattamente da metà ipotesi mentre negativa dall’altra metà, quindi non si avrà la capacità di predire ulteriori esempi non rappresentati esplicitamente nel training set.

Questo vuol dire che non è utile pensare di avere un modello di apprendimento senza bias, **sarebbe inutile perché non permetterebbe di generalizzare dal training set il concetto e non permetterebbe di predire la classe di ulteriori esempi che non sono stati osservati nel training set.**

## Bias induttivo

Consideriamo di avere

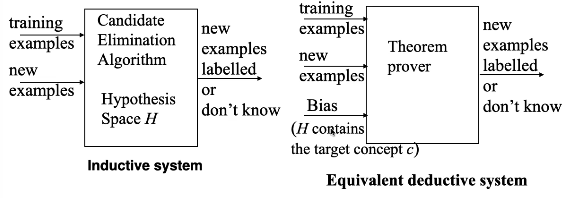
* Un algoritmo di apprendimento L
* Uno spazio delle istanze X, a cui c’è una funzione che associa il concetto target a ciascuna di queste istanze assegnando l’etichetta
* Un Training set Dc = { < x, c(x) > }
* Se applichiamo L a tutto il training set Dc, l’applicazione di L all’esempio xi futuro potremmo cercare di predire l’etichetta di xi 🡪 L(xi, Dc))

**Definizione bias induttivo**

Consiste in un insieme di asserzioni minimale B (presunte) tale che per qualunque concetto target c che dobbiamo apprendere, una volta che abbiamo il corrispondente training set Dc per apprendere il modello, dato un qualunque esempio futuro xi è possibile dedurre con una logica le etichette da assegnare all’etichetta xi con una conseguenza logica del bias delle insieme delle asserzioni, del training set e dell’esempio stesso



Vediamo ora un aspetto molto utile per quanto riguarda il bias induttivo. Ci permette di fare una equivalenza tra un sistema induttivo e uno deduttivo. Questa equivalenza è utile perché ci permette di descrivere un sistema induttivo con un sistema di tipo deduttivo.



Nei sistemi induttivi lo spazio delle ipotesi contiene all’interno il bias. Se esplicitiamo la presenza del bias come input al sistema (che fa solo conseguenze logiche attraverso le premesse) si ottiene un sistema deduttivo equivalente al sistema induttivo visto. Quindi il sistema deduttivo si occupa di applicare solo la logica agli esempi.

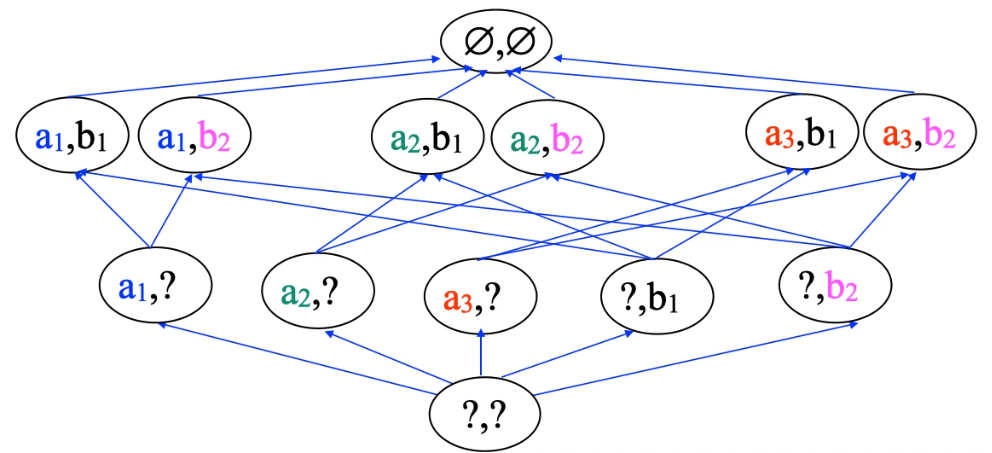
Il sistema deduttivo ragiona come un sistema di ragionamento che in maniera logica deduce le etichette a partire dall’input.

**Vediamo i bias usati dagli algoritmi visti.**

* **Rote learner**, restituisce l’etichetta degli esempi memorizzati in memoria, non usa Bias e non sa generalizzare
* **Version space candidate elimination algorithm**, il bias è che permette di rappresentare il concetto c nel linguaggio adottato. Permette la classificazione se tutte le ipotesi nel version space forniscono una risposta, altrimenti non si esprime.
* **Find-S**, ha un bias che si limita a guardare solo gli esempi positivi escludendo quelli negativi (è un bias più forte)

PS. Bias quindi fondamentale per permettere il salto induttivo.

PARTE SALTATA

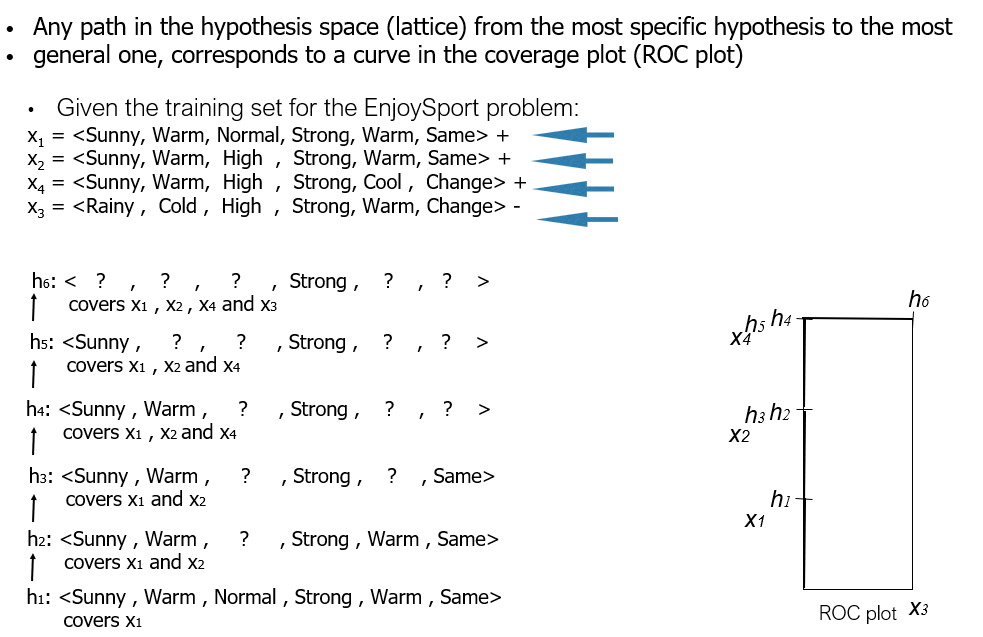


Possiamo anche vedere l’applicazione di queste strategie di navigazione del reticolo che ordina le ipotesi dalle più specifiche a quelle più generali, con solo 2 features (A e B) a cui vengono assegnati ad A tre valori (a1, a2 e a3) e a B due (b1 e b2).

Potremmo pensare ad un algoritmo che a fronte di un esempio negativo ci permetta di eliminare tutte le generalizzazione che coprirebbero l’esempio negativo. Viceversa, quando c’è un esempio positivo ci permette di andare a eliminare tutte le ipotesi più specifiche di quella non generale non soddisfatta dall’esempio.

# Path in uno spazio delle ipotesi

Rappresentiamo il nostro reticolo in un grafico ROC



H2 copre x1, x2 e x3, idem h3 🡪 non tutte le generalizzazioni ci fanno spostare nel roc plot (perché rimaniamo fermi a coprire lo stesso numero di esempi positivi, non sbagliando nessun esempio) 🡪 questo per dire che ci possono essere ipotesi che sembrano apparentemente una più generale dell’altra (come h2 e h3), ma senza avere certezza dai dati.

Con h6 arriviamo ad una generalizzazione troppo elevata che andrà a sbagliare su x3

Sui dati noi saremmo in grado di costruire ipotesi che sul roc plot sono equivalenti (non possiamo discriminare le ipotesi più specifiche come per h2 e h3). Quindi esistono modelli diversi (che facendo concept learning, attraverso il VS, sappiamo essere diversi per specificità) che si trovano nello stesso punto nel ROC, quindi che avranno stesse performance sui nostri dati.

Il task di concept learning può essere visto come una ricerca nello spazio delle ipotesi che passi attraverso il ROC heaven.

Ciò è il punto di congiunzione tra reticolo e ROC