# Richiamo di Concetti di Apprendimento Automatico ed altre nozioni aggiuntive

Libro di riferimento: T. Mitchell

# Ingredienti Fondamentali Apprendimento Automatico

- Dati di Allenamento (estratti dallo Spazio delle Istanze, $X$)
- Spazio delle Ipotesi, $\mathcal{H}$
  - costituisce l'insieme delle funzioni che possono essere realizzate dal sistema di apprendimento;
  - si assume che la funzione da apprendere $f$ possa essere rappresentata da una ipotesi $h \in \mathcal{H}$... (selezione di $h$ attraverso i dati di apprendimento)
  - o che almeno una ipotesi $h \in \mathcal{H}$ sia simile a $f$ (approssimazione);
- Algoritmo di Ricerca nello Spazio delle Ipotesi, alg. di apprendimento

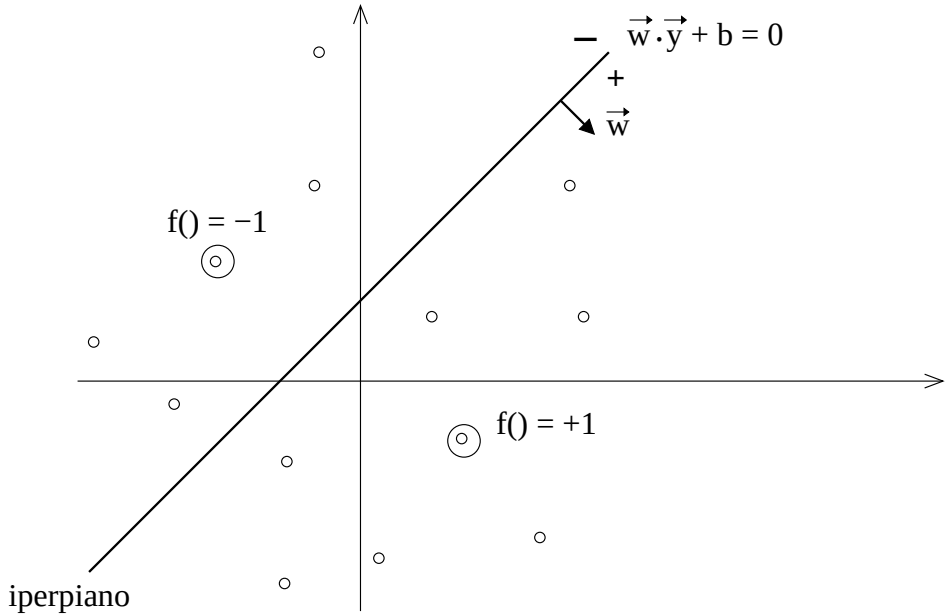
**ATTENZIONE:** $\mathcal{H}$ non può coincidere con l'insieme di tutte le funzioni possibili e la ricerca essere esaustiva $\rightarrow$ Apprendimento è inutile!!!

Si parla di **Bias Induttivo**: sulla rappresentazione ($\mathcal{H}$) e/o sulla ricerca (alg. di apprendimento)

# Spazio delle Ipotesi: Esempio 1

Iperpiani in $\mathbb{R}^2$

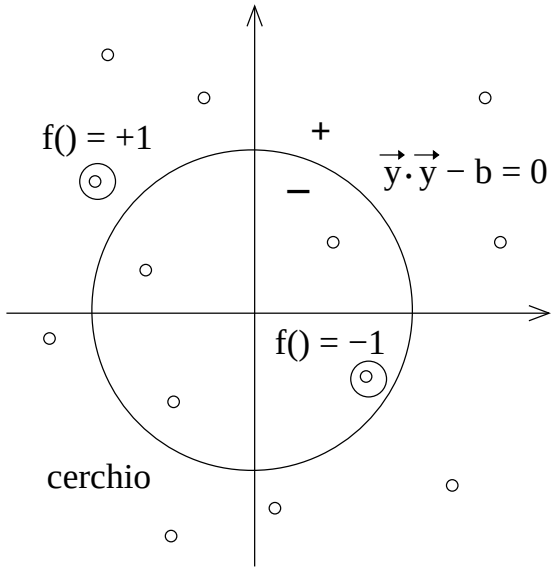
- Spazio delle Istanze $\rightarrow$ punti nel piano: $X = \{\vec{y} \in \mathbb{R}^2\}$
- Spazio delle Ipotesi $\rightarrow$ dicotomie indotte da iperpiani in $\mathbb{R}^2$:
  $\mathcal{H} = \{f_{(\vec{w},b)}(\vec{y}) | f_{(\vec{w},b)}(\vec{y}) = sign(\vec{w} \cdot \vec{y} + b), \vec{w} \in \mathbb{R}^2, b \in \mathbb{R}\}$



# Spazio delle Ipotesi: Esempio 2

Dischi in $\mathbb{R}^2$

- Spazio delle Istanze $\rightarrow$ punti nel piano: $X = \{\vec{y} \in \mathbb{R}^2\}$
- Spazio delle Ipotesi $\rightarrow$ dicotomie indotte da dischi in $\mathbb{R}^2$ centrati nell'origine:
  $\mathcal{H} = \{f_b(\vec{y}) | f_b(\vec{y}) = sign(\vec{y} \cdot \vec{y} - b), b \in \mathbb{R}\}$



# Spazio delle Ipotesi: Esempio 3

Congiunzione di $m$ letterali positivi

- Spazio delle Istanze $\rightarrow$ stringhe di $m$ bit: $X = \{s | s \in \{0,1\}^m\}$
- Spazio delle Ipotesi $\rightarrow$ tutte le sentenze logiche che riguardano i letterali positivi $l_1, \ldots, l_m$ ($l_1$ è vero se il primo bit vale 1, $l_2$ è vero se il secondo bit vale 1, etc.) e che contengono solo l'operatore $\wedge$ (**and**):

$$\mathcal{H} = \{f_{\{i_1,\ldots,i_j\}}(s) | f_{\{i_1,\ldots,i_j\}}(s) \equiv l_{i_1} \wedge l_{i_2} \wedge \cdots \wedge l_{i_j}, \{i_1, \ldots, i_j\} \subseteq \{1, \ldots, m\}\}$$

Es. $m = 3$, $X = \{0,1\}^3$

Esempi di istanze $\rightarrow$ $s1 = 101$, $s2 = 001$, $s3 = 100$, $s4 = 111$

Esempi di ipotesi $\rightarrow$ $h_1 \equiv l_2$, $h_2 \equiv l_1 \wedge l_2$, $h_3 \equiv true$, $h_4 \equiv l_1 \wedge l_3$, $h_5 \equiv l_1 \wedge l_2 \wedge l_3$

Notare che: $h_1$, $h_2$, e $h_5$ sono false per $s1$, $s2$ e $s3$ e vere per $s4$; $h_3$ è vera per ogni istanza; $h_4$ è vera per $s1$ e $s4$ ma falsa per $s2$ e $s3$

# Principali Paradigmi di Apprendimento: Richiamo

Apprendimento Supervisionato:

- dato in insieme di esempi pre-classificati, $Tr = \{(x^{(i)}, f(x^{(i)}))\}$, apprendere una descrizione generale che incapsula l'informazione contenuta negli esempi
  (regole valide su tutto il dominio di ingresso)
- tale descrizione deve poter essere usata in modo predittivo
  (dato un nuovo ingresso $\tilde{x}$ predire l'output associato $f(\tilde{x})$)
- si assume che un esperto (o maestro) ci fornisca la supervisone
  (cioè i valori della $f()$ per le istanze $x$ dell'insieme di apprendimento)

**Find-S** è un algoritmo di apprendimento supervisionato

# Dati

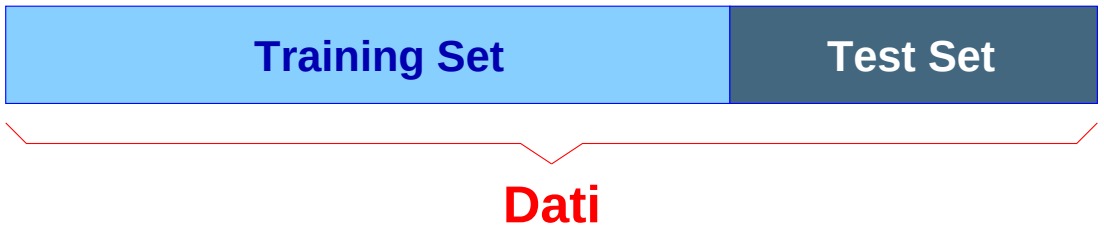
Consideriamo il paradigma di Apprendimento Supervisonato

Dati a nostra disposizione (off-line)

$$\text{Dati} = \{(x^{(1)}, f(x^{(1)})), \ldots, (x^{(N)}, f(x^{(N)}))\}$$

Suddivisione tipica ($N = N_{tr} + N_{ts}$) :

- $\text{Training Set} = \{(x^{(1)}, f(x^{(1)})), \ldots, (x^{(N_{tr})}, f(x^{(N_{tr})}))\}$
  usato direttamente dall'algoritmo di apprendimento;
- $\text{Test Set} = \{(x^{(1)}, f(x^{(1)})), \ldots, (x^{(N_{ts})}, f(x^{(N_{ts})}))\}$
  usato alla fine dell'apprendimento per stimare la bontà della soluzione.

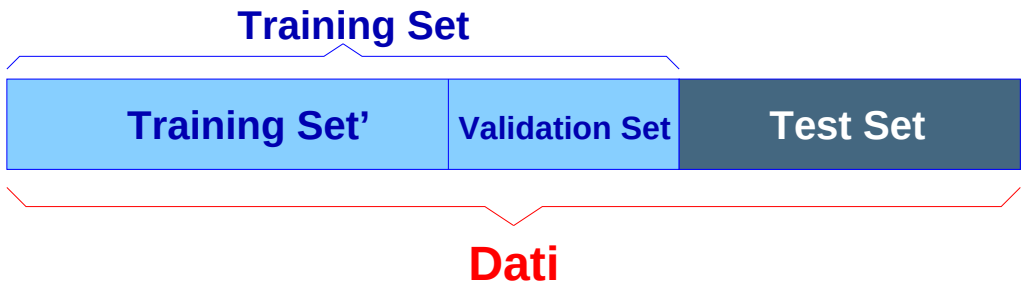


# Dati (cont.)

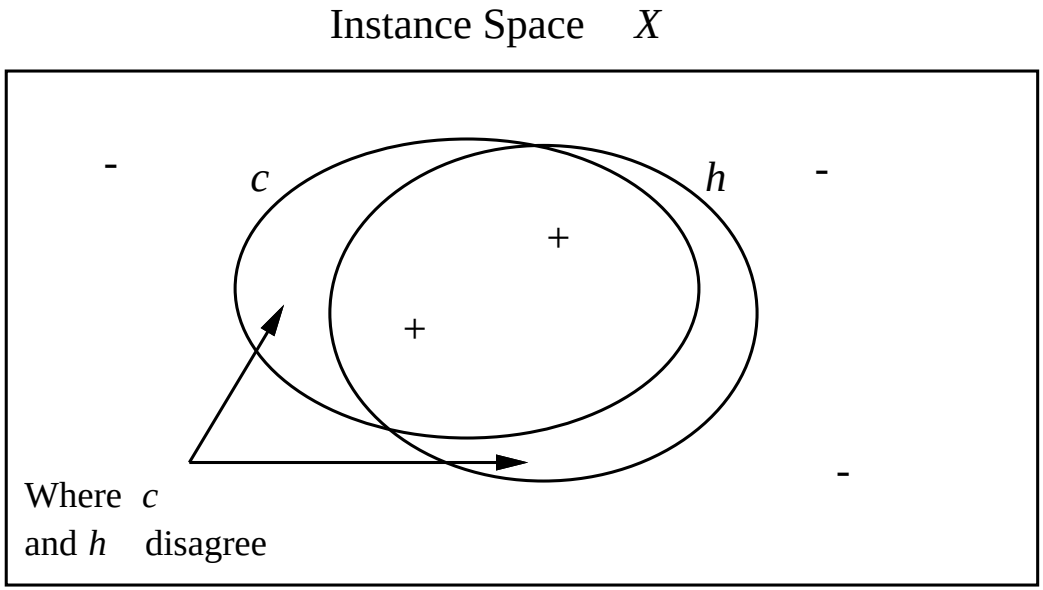
Se $N$ abbastanza grande il Training Set è ulteriormente suddiviso in due sottoinsiemi ($N_{tr} = N_{\widehat{tr}} + N_{val}$) :

- Training Set' $= \{(x^{(1)}, f(x^{(1)})), \ldots, (x^{(N_{\widehat{tr}})}, f(x^{(N_{\widehat{tr}})})\}$
  usato direttamente dall'algoritmo di apprendimento;
- Validation Set $= \{(x^{(1)}, f(x^{(1)})), \ldots, (x^{(N_{val})}, f(x^{(N_{val})})\}$
  usato indirettamente dall'algoritmo di apprendimento.

Il Validation Set serve per scegliere l'ipotesi $h \in \mathcal{H}$ migliore fra quelle consistenti con il Training Set'



# Errore Ideale

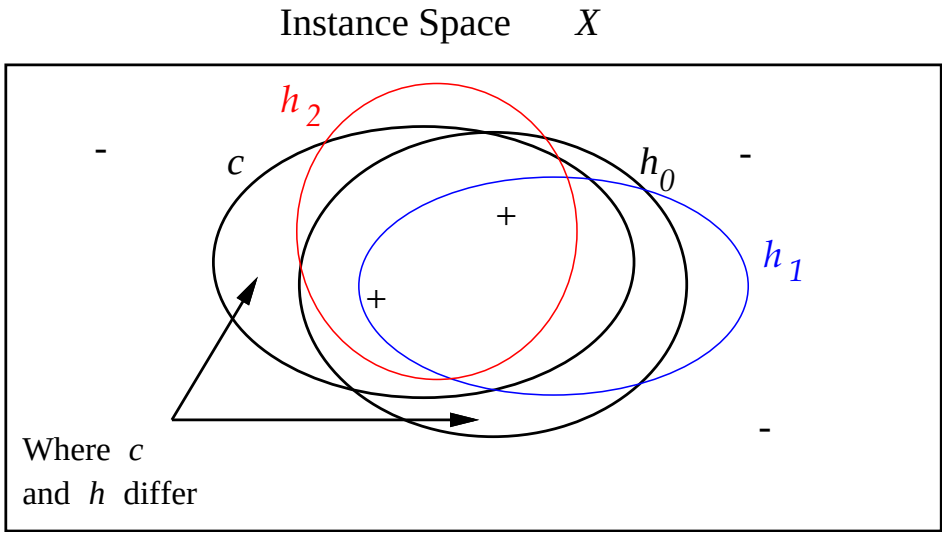



Supponiamo che la funzione $f$ da apprendere sia una funzione booleana (concetto):

$$f : X \rightarrow \{0, 1\} \ (\{-, +\})$$

**Def:** L'**Errore Ideale** ($error_{\mathcal{D}}(h)$) di una ipotesi $h$ rispetto al concetto $f$ e la distribuzione di probabilità $\mathcal{D}$ (probabilità di osservare l'ingresso $x \in X$) è la probabilità che $h$ classifichi erroneamente un input selezionato a caso secondo $\mathcal{D}$: $error_{\mathcal{D}}(h) \equiv \underset{x \in \mathcal{D}}{Pr}[f(x) \neq h(x)]$

# Errore di Apprendimento




Dato $Tr$ = Training Set', più ipotesi possono essere consistenti: $h_0$, $h_1$, $h_2$ quale scegliere ?

**Def:** L'**Errore Empirico** ($error_{Tr}(h)$) di una ipotesi $h$ rispetto a $Tr$ è il numero di esempi che $h$ classifica erroneamente: $error_{Tr}(h) \equiv \#\{(x, f(x)) \in Tr | f(x) \neq h(x)\}$

**Def:** Una ipotesi $h \in \mathcal{H}$ è sovraspecializzata (**overfit**) $Tr$ se $\exists h' \in \mathcal{H}$ tale che $error_{Tr}(h) < error_{Tr}(h')$, ma $error_{\mathcal{D}}(h) > error_{\mathcal{D}}(h')$.

Il Validation Set serve per cercare di selezionare l'ipotesi migliore (evitare **overfit**).

# VC-dimension

**Definizione:** Frammentazione (Shattering)

Dato $S \subset X$, $S$ è frammentato (shattered) dallo spazio delle ipotesi $\mathcal{H}$ se e solo se

$$\forall S' \subseteq S,\ \exists h \in \mathcal{H},\ \text{tale che } \forall x \in S,\ h(x) = 1 \Leftrightarrow x \in S'$$

($\mathcal{H}$ realizza tutte le possibili dicotomie di $S$)

**Definizione:** VC-dimension

La VC-dimension di uno spazio delle ipotesi $\mathcal{H}$ definito su uno spazio delle istanze $X$ è data dalla cardinalità del sottoinsieme più grande di $X$ che è frammentato da $\mathcal{H}$:

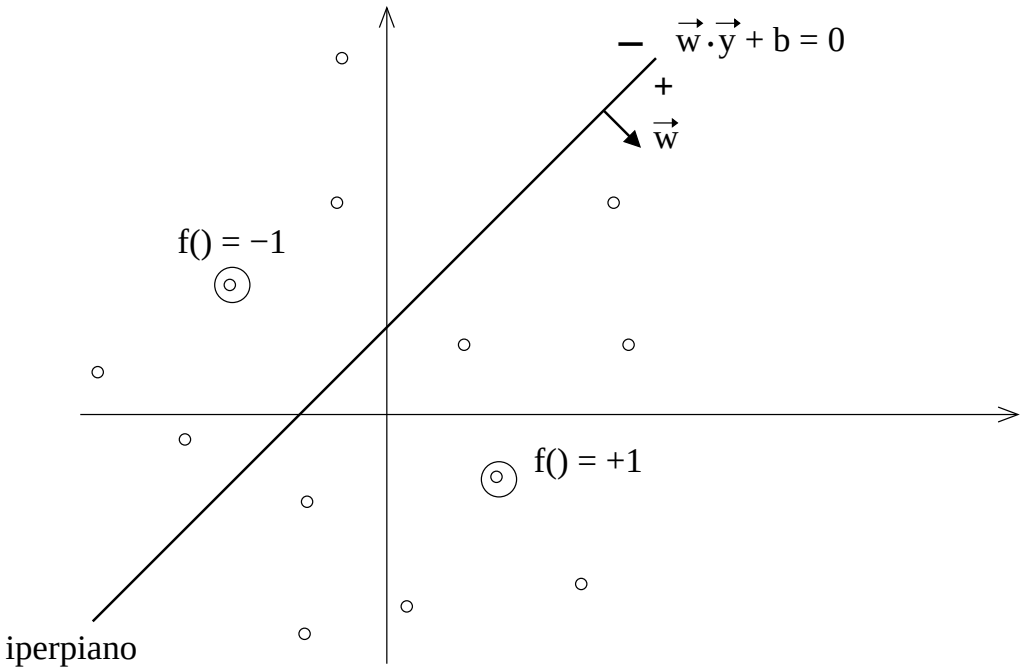
$$VC(\mathcal{H}) = \max_{S \subseteq X} |S| :\ \mathcal{H} \text{ frammenta } S$$

$VC(\mathcal{H}) = \infty$ se $S$ non è limitato

# VC-dimension: Esempio

Quale è la VC-dimension di $\mathcal{H}_1$ ?

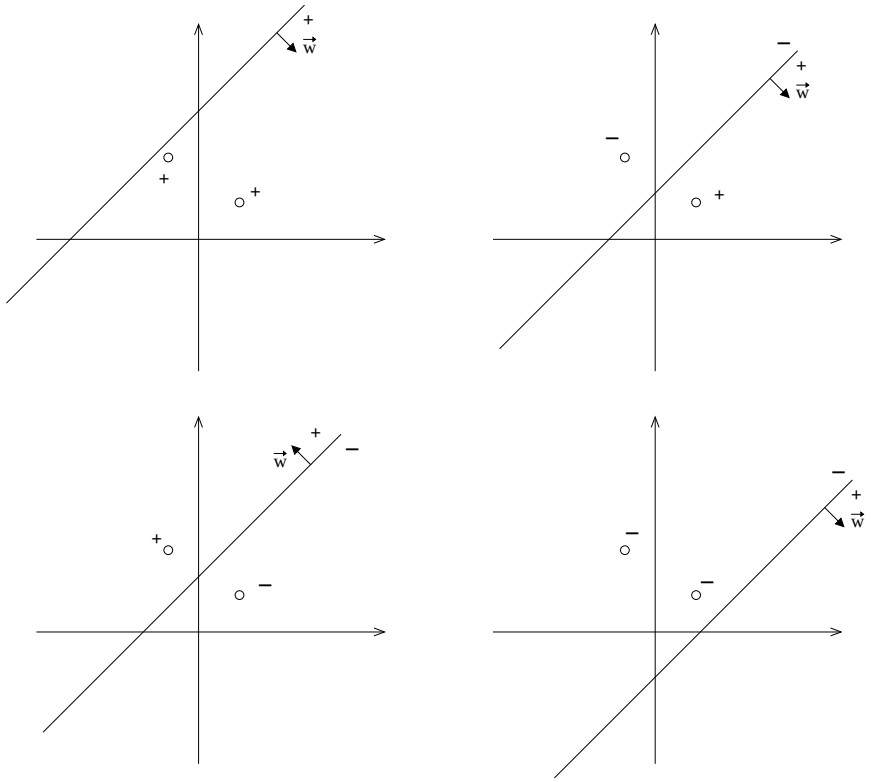
$$\mathcal{H}_1 = \{f_{(\vec{w},b)}(\vec{y}) | f_{(\vec{w},b)}(\vec{y}) = sign(\vec{w} \cdot \vec{y} + b), \vec{w} \in \mathbb{R}^2, b \in \mathbb{R}\}$$



# VC-dimension: Esempio

Quale è la VC-dimension di $\mathcal{H}_1$ ?

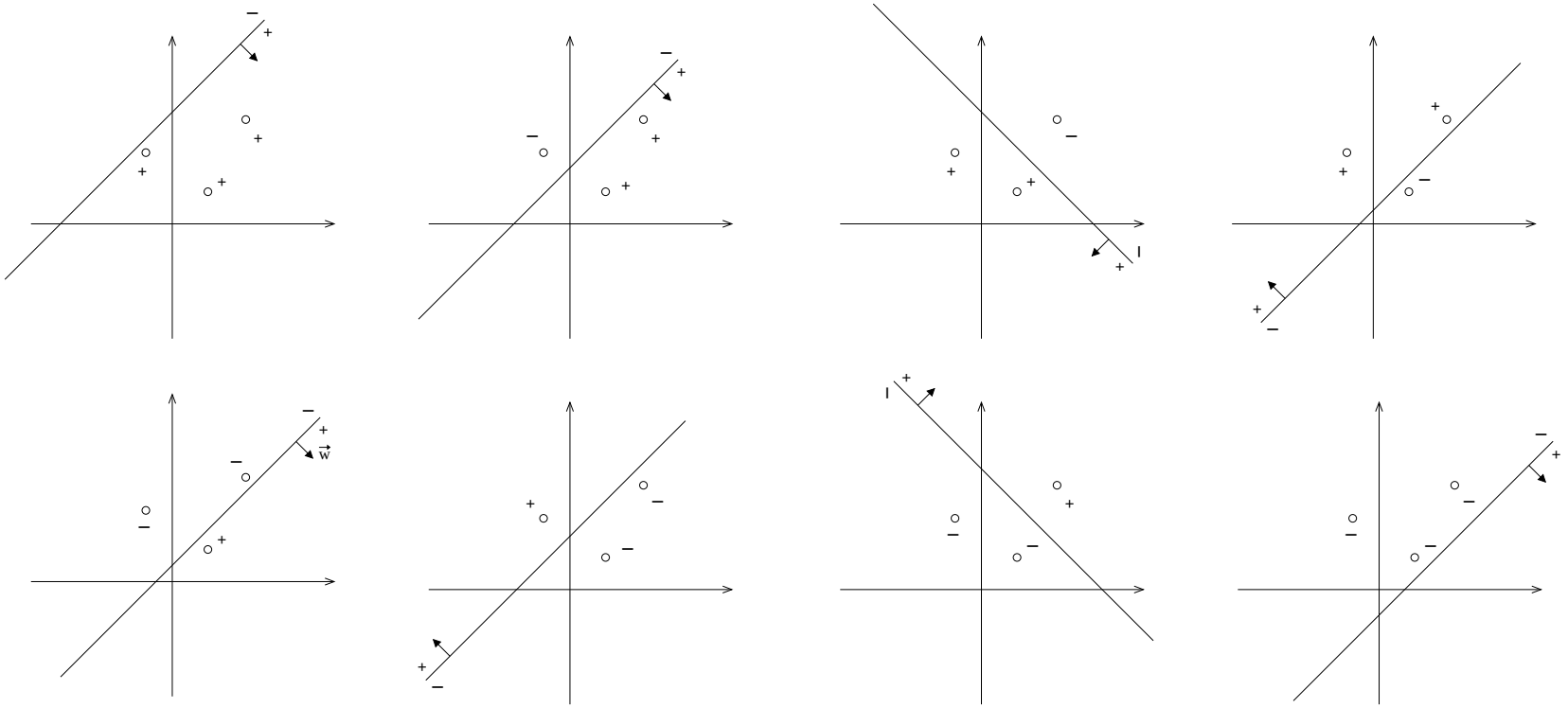
$VC(\mathcal{H}) \geq 1$ banale. Vediamo cosa succede con 2 punti:



# VC-dimension: Esempio

Quale è la VC-dimension di $\mathcal{H}_1$ ?

Quindi $VC(\mathcal{H}) \geq 2$. Vediamo cosa succede con 3 punti:



# VC-dimension: Esempio

Quale è la VC-dimension di $\mathcal{H}_1$ ?

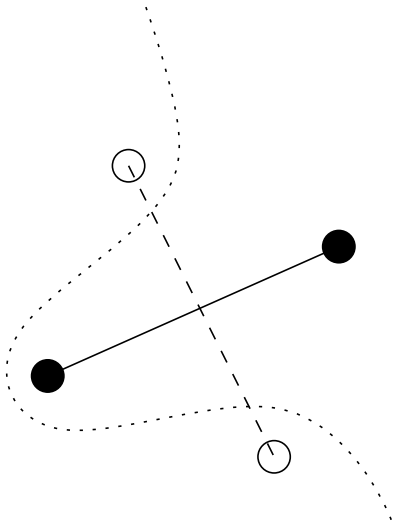
Quindi $VC(\mathcal{H}) \geq 3$. Cosa succede con 4 punti ?

# VC-dimension: Esempio

Quale è la VC-dimension di $\mathcal{H}_1$ ?

Quindi $VC(\mathcal{H}) \geq 3$. Cosa succede con 4 punti ? Non si riesce a frammentare 4 punti!!

Infatti esisteranno sempre due coppie di punti che se unite con un segmento provocano una intersezione fra i due segmenti e quindi, ponendo ogni coppia di punti in classi diverse, per separarli non basta una retta, ma occorre una curva. Quindi $VC(\mathcal{H}) = 3$



# Bound sull'Erorre Ideale per Classificazione Binaria

Consideriamo un problema di classificazione binario (i.e., apprendimento di concetti). Dati

- Training Set $Tr = \{(\boldsymbol{x}^{(1)}, f(\boldsymbol{x}^{(1)})), \ldots, (\boldsymbol{x}^{(N_{tr})}, f(\boldsymbol{x}^{(N_{tr})}))\}$
- Spazio delle Ipotesi $\mathcal{H} = \{h_{\boldsymbol{w}}(\boldsymbol{x}) | \boldsymbol{w} \in \mathbb{R}^k\}$
- Algoritmo di Apprendimento $L$ che restituisce l'ipotesi $h_{\boldsymbol{w}^*}(\boldsymbol{x})$, dove $\boldsymbol{w}^*$ minimizza l'errore empirico $\boldsymbol{error_{Tr}(h_w(x))}$

è possibile derivare dei bound sull'errore ideale (detto anche errore di generalizzazione), validi con probabilità $1 - \delta$, che hanno una forma del tipo

$$error_{\mathcal{D}}(h_{\boldsymbol{w}^*}(\boldsymbol{x})) \leq error_{Tr}(h_{\boldsymbol{w}^*}(\boldsymbol{x})) + \epsilon(N_{tr}, VC(\mathcal{H}), \delta)$$

Esempio:

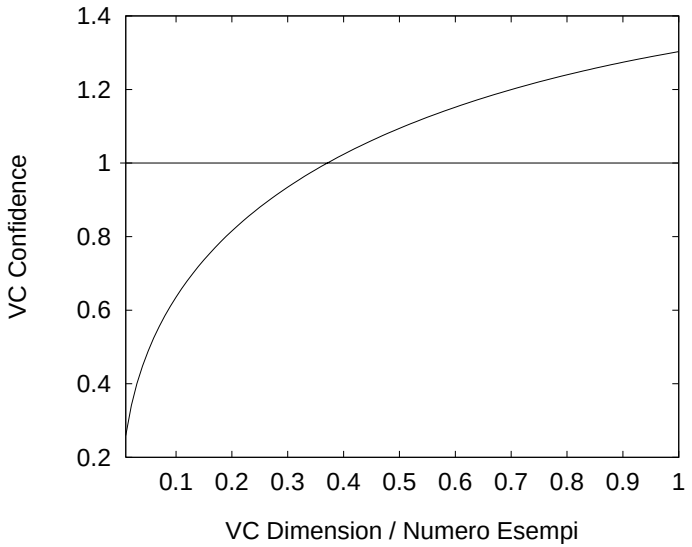
$$error_{\mathcal{D}}(h_{\boldsymbol{w}^*}(\boldsymbol{x})) \leq \underbrace{error_{Tr}(h_{\boldsymbol{w}^*}(\boldsymbol{x}))}_{A} + \underbrace{\sqrt{\frac{VC(\mathcal{H})}{N_{tr}}(\log(\frac{2N_{tr}}{VC(\mathcal{H})}) + 1) - \frac{1}{N_{tr}}\log(\delta)}}_{B}$$

# Bound sull'Erorre Ideale per Classificazione Binaria

Si noti che

- il termine A DIPENDE SOLO dalla ipotesi restituita dall'algoritmo di apprendimento $L$;
- il termine B è INDIPENDENTE dalla ipotesi restituita dall'algoritmo di apprendimento $L$; in particolare dipende dal rapporto fra VC-dimension dello spazio delle ipotesi $\mathcal{H}$ e il numero di esempi di apprendimento ($N_{tr}$), oltre ovviamente che dalla confidenza $(1 - \delta)$ con cui il bound è valido.

Il termine B è usualmente chiamato VC-confidence e risulta essere monotono rispetto al rapporto $\frac{VC(\mathcal{H})}{N_{tr}}$; fissato $N_{tr}$ aumenta all'aumentare di $VC(\mathcal{H})$.



# Structural Risk Minimization

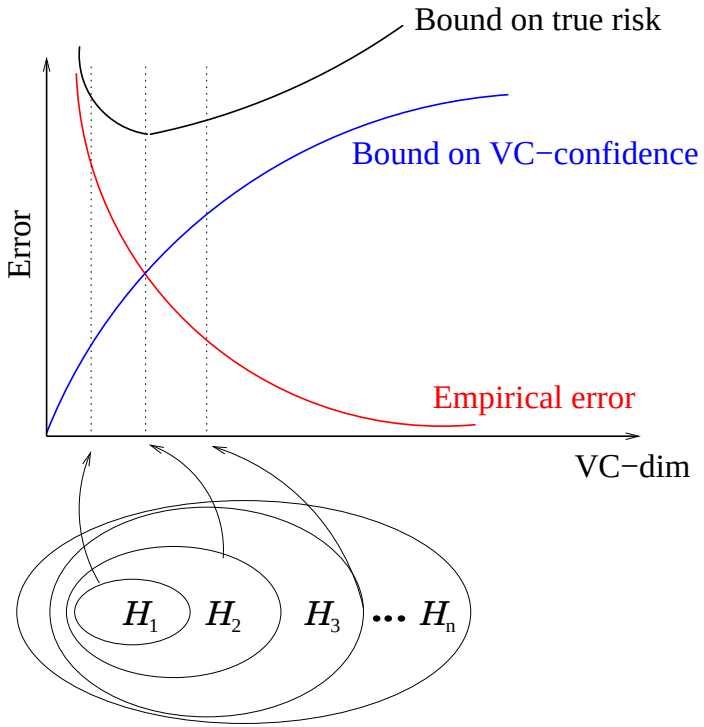
Problema: all'aumentare della VC-dimension diminuisce l'errore empirico (termine A), ma aumenta la VC confidence (termine B)!

L'approccio Structural Risk Minimization tenta di trovare un compromesso tra i due termini:

Si considerano $\mathcal{H}_i$ tali che

- $\mathcal{H}_1 \subseteq \mathcal{H}_2 \subseteq \cdots \subseteq \mathcal{H}_n$
- $VC(\mathcal{H}_1) \leq \cdots \leq VC(\mathcal{H}_n)$
- si seleziona l'ipotesi che ha il bound sull'errore ideale più basso

Esempio: Reti neurali con un numero crescente di neuroni nascosti



# Apprendimento di concetti: alcune definizioni

**Definizione:** Un concetto in uno Spazio delle Istanze (Instance Space) $X$ è definito come una funzione booleana su $X$.

**Definizione:** Un esempio di un concetto $c$ su uno Spazio delle Istanze $X$ è definito come una coppia $(x, c(x))$, dove $x \in X$ e $c()$ è una funzione booleana.

**Definizione:** Poniamo $h$ essere una funzione booleana definita sullo Spazio delle Istanze $X$. Diciamo che $h$ soddisfa $x \in X$ se $h(x) = 1$ ($true$).

**Definizione:** Poniamo $h$ essere una funzione booleana definita sullo Spazio delle Istanze $X$ e $(x, c(x))$ un esempio di $c()$. Diciamo che $h$ è consistente con l'esempio se $h(x) = c(x)$. In più diciamo che $h$ è consistente con un insieme di esempi $Tr$ se $h$ è consistente con ogni esempio in $Tr$.

# Spazio delle Ipotesi: ordine parziale

**Definizione:** Siano $h_i$ e $h_j$ funzioni booleane definite su uno Spazio delle Istanze $X$. Diciamo che $h_i$ è più generale o equivalente di $h_j$ ($h_i \geq_g h_j$) se e solo se

$$(\forall x \in X)[(h_j(x) = 1) \rightarrow (h_i(x) = 1)]$$

Esempi

- $l_1 \geq_g (l_1 \wedge l_2)$
- $l_2 \geq_g (l_1 \wedge l_2)$
- $l_1 \not\geq_g l_2$ e $l_2 \not\geq_g l_1$ (non comparabili)

# Esercizio: apprendimento di congiunzioni di letterali

Algoritmo **Find-S**

/* trova l'ipotesi più specifica che è consistente con l'insieme di apprendimento */

- input: insieme di apprendimento $Tr$
- inizializza $h$ con ipotesi più specifica
  $h \equiv l_1 \wedge \neg l_1 \wedge l_2 \wedge \neg l_2 \wedge \cdots \wedge l_m \wedge \neg l_m$
- per ogni istanza di apprendimento positiva $(x, true) \in Tr$
  - rimuovi da $h$ ogni letterale che non sia soddisfatto da $x$
- restituisci $h$

# Esempio di applicazione: $m = 5$

| Esempio (positivo) | ipotesi corrente |
|---|---|
| | $h_0 \equiv l_1 \wedge \neg l_1 \wedge l_2 \wedge \neg l_2 \wedge l_3 \wedge \neg l_3 \wedge l_4 \wedge \neg l_4 \wedge l_5 \wedge \neg l_5$ |
| 1 1 0 1 0 | $h_1 \equiv l_1 \wedge l_2 \wedge \neg l_3 \wedge l_4 \wedge \neg l_5$ |
| 1 0 0 1 0 | $h_2 \equiv l_1 \wedge \neg l_3 \wedge l_4 \wedge \neg l_5$ |
| 1 0 1 1 0 | $h_3 \equiv l_1 \wedge l_4 \wedge \neg l_5$ |
| 1 0 1 0 0 | $h_4 \equiv l_1 \wedge \neg l_5$ |
| 0 0 1 0 0 | $h_5 \equiv \neg l_5$ |

Notare che $h_0 \leq_g h_1 \leq_g h_2 \leq_g h_3 \leq_g h_4 \leq_g h_5$

Inoltre, ad ogni passo l'ipotesi corrente $h_i$ è sostituita dall'ipotesi $h_{i+1}$ che costituisce una *generalizzazione minima* di $h_i$ consistente con l'esempio corrente.

Pertanto **Find-S** restituisce l'ipotesi più specifica consistente con $Tr$

# Osservazioni su **Find-S**

**Find-S** può essere adattato ad altri Spazi delle Istanze ed Ipotesi.

L'idea base dell'algoritmo consiste nel calcolare una *generalizzazione minima* della ipotesi corrente quando questa non è consistente con l'esempio corrente.

Si noti che, ogni qualvolta che l'ipotesi corrente $h$ è *generalizzata* portando ad una nuova ipotesi $h'$ ($h' \geq_g h$), tutti gli esempi positivi presentati in precedenza sono soddisfatti dalla nuova ipotesi $h'$ (infatti, poiché $h' \geq_g h$, si ha che
$\forall x \in X, \, (h(x) = 1) \rightarrow (h'(x) = 1)$)

Infine, se il concetto da apprendere è contenuto in $\mathcal{H}$, tutti gli esempi negativi (per cui, $c(x) = 0$) sono soddisfatti automaticamente dalla ipotesi restituita da **Find-S** poiché tale ipotesi è la più specifica fra quelle consistenti, cioè quella che assegna il numero più piccolo di "1" alle istanze in $X$.

Esiste un motivo valido per preferire l'ipotesi consistente più specifica ?

# Uso degli esempi negativi...

Esiste un motivo valido per preferire l'ipotesi consistente più specifica ? NO!

Pertanto cerchiamo di capire come trovare l'insieme di TUTTE le ipotesi che sono consistenti con l'insieme di apprendimento (detto **Version Space**).

(Per semplificare l'esposizione, assumiamo che il Version Space abbia un' ipotesi più specifica di tutte, cioè quella restituita da **Find-S**).

Una ipotesi nel Version Space sarà più generale od equivalente a quella restituita da **Find-S**; in aggiunta, deve essere consistente con tutti gli esempi negativi.

Pertanto l'intuizione è di partire con un Version Space candidato inizialmente equivalente all'intero Spazio delle Ipotesi (nessun esempio ancora presentato), e poi usare gli esempi positivi per rimuovere ipotesi che sono troppo specifiche, e gli esempi negativi per rimuovere le ipotesi che sono troppo generali (algoritmo **Candidate-Elimination**).

# Algoritmo **Candidate-Elimination**

Definisce implicitamente il Version Space tramite

- **Confine più Specifico (Specific Boundary)**

$$S \equiv \{s \in \mathcal{H} | consistente(s, Tr) \wedge (\neg \exists s' \in \mathcal{H})[s >_g s') \wedge consistente(s', Tr)])\}$$

- **Confine più Generale (General Boundary)**

$$G \equiv \{g \in \mathcal{H} | consistente(g, Tr) \wedge (\neg \exists g' \in \mathcal{H})[g' >_g g) \wedge consistente(g', Tr)])\}$$

Il Version Space è definito come

$$VS_{\mathcal{H},Tr} = \{h \in \mathcal{H} | (\exists s \in S)(\exists g \in G)(g \geq_g h \geq_g s)\}$$

Provate ad immaginare come funziona l'algoritmo...

# Candidate-Elimination

inizializza $G$ all'insieme delle ipotesi più generale e $S$ all'insieme delle ipotesi più specifiche

**for each** $d \equiv (x, c(x)) \in Tr$ **do**

**if** $c(x) = 1$ /* esempio positivo */

rimuovi da $G$ ogni ipotesi inconsistente con $d$

$\forall$ ipotesi $s \in S$ inconsistente con $d$

rimuovi $s$ da $S$

aggiungi ad $S$ tutte le generalizzazioni minime $h$ di $s$ t.c.

$consistente(h, d)$ ed $\exists g \in G$ t.c. $g \geq_g h$

rimuovi da $S$ tutte le ipotesi $s$ per cui $\exists s' \in S$ t.c. $s \geq_g s'$

**if** $c(x) = 0$ /* esempio negativo */

rimuovi da $S$ ogni ipotesi inconsistente con $d$

$\forall$ ipotesi $g \in G$ inconsistente con $d$

rimuovi $g$ da $G$

aggiungi a $G$ tutte le specializzazioni minime $h$ di $s$ t.c.

$consistente(h, d)$ ed $\exists s \in S$ t.c. $h \geq_g s$

rimuovi da $G$ tutte le ipotesi $g$ per cui $\exists g' \in S$ t.c. $g' \geq_g g$

# Version Space

Notare che la cardinalità del Version Space:

- può essere infinita (se $\mathcal{H}$ è infinito), ma sia $S$ che $G$ possono avere una rappresentazione finita
- in generale la sua cardinalità decresce con il crescere della cardinalità di $Tr$ (più vincoli per le ipotesi)

Assumendo che $c \in \mathcal{H}$, più è piccola la cardinalità del Version Space, più alta è la probabilità che selezionando un' ipotesi a caso $h \in VS_{\mathcal{H},Tr}$ si ottenga il concetto desiderato $c$, cioè $h \equiv c$.

# Apprendimento PAC

PAC Learning (Probably Approximately Correct Learning)

Assume che: input ed output sono binari, non c'è rumore, le istanze sono estratte da $X$ concordemente ad una distribuzione di probabilità $\mathcal{D}$ *arbitraria* ma *stazionaria*.

Il framework di apprendimento PAC cerca di rispondere alle seguenti domande:

- Sotto quali condizioni apprendere con successo è possibile o impossibile ?
- Sotto quali condizioni si può assicurare che un particolare algoritmo di apprendimento apprenda con successo ?

# Apprendimento PAC

Consideriamo una classe $C$ di possibili concetti target (funzioni che vogliamo apprendere) definita su uno Spazio delle Istanze $X$ (con istanze di dimensione $m$), ed un algoritmo di apprendimento $L$ che utilizza uno Spazio delle Ipotesi $\mathcal{H}$.

**Def.:** $C$ è PAC-apprendibile da $L$ usando $\mathcal{H}$ se per ogni

- $c \in C$,
- distribuzione $\mathcal{D}$ su $X$,
- $\epsilon$ tale che $0 < \epsilon < 1/2$,
- $\delta$ tale che $0 < \delta < 1/2$,

l'algoritmo di apprendimento $L$ con probabilità almeno $(1 - \delta)$ restituisce una ipotesi $h \in \mathcal{H}$ tale che $error_{\mathcal{D}}(h) \leq \epsilon$, in tempo che è **polinomiale** in $1/\epsilon$, $1/\delta$, $m$, e $size(c)$ (spazio di memoria necessario per rappresentare $c$).

# Apprendimento PAC

Si può dimostrare che assumendo:

- $c \in \mathcal{H}$
- $L$ consistente, cioè che restituisce una ipotesi $h$ consistente con $Tr$, o equivalentemente $h \in VS_{\mathcal{H},Tr}$ (ad esempio, **Find-S** è consistente)

allora, con probabilità almeno $(1-\delta)$, $L$ restituisce una ipotesi $h \in \mathcal{H}$ tale che $error_{\mathcal{D}}(h) < \epsilon$ se il numero di esempi di apprendimento $n$ soddisfa la seguente disuguaglianza:

$$n \geq \frac{1}{\epsilon}(ln(|\mathcal{H}|) + ln(\frac{1}{\delta}))$$

dove $|\mathcal{H}|$ è la cardinalità dello Spazio delle Ipotesi e $ln()$ è il logaritmo naturale

## Prova...

Idea base: bisogna dare un limite al numero di esempi necessario ad assicurare che il Version Space (ricordiamo che $L$ è consistente) non contiene ipotesi non "accettabili":

$$(\forall h \in VS_{\mathcal{H},Tr})\ error_{\mathcal{D}}(h) < \epsilon$$

# Prova...

$$(\forall h \in VS_{\mathcal{H},Tr})\ error_{\mathcal{D}}(h) < \epsilon$$

**Primo risultato:** se $|\mathcal{H}| < \infty$, $|Tr| = n > 0$ e $Tr$ è costituito da esempi di un concetto target $c$ estratti indipendentemente ed a caso, allora per ogni $0 \leq \epsilon \leq 1$, la probabilità che la disuguaglianza **NON** sia soddisfatta è minore od uguale a $|\mathcal{H}|e^{-\epsilon n}$:

- siano $h_1, \ldots, h_k$ tutte le ipotesi con $error_{\mathcal{D}}(h) \geq \epsilon$; la disuguaglianza NON è soddisfatta se e solo se ALMENO una di tali ipotesi è consistente con $Tr$
- la probabilità che una di tali ipotesi sia consistente con un singolo esempio è $(1-\epsilon)$, e quindi per $n$ esempi indipendenti la probabilità è $(1-\epsilon)^n$
- poiché esistono $k$ di tali ipotesi, la probabilità che ci interessa è a più $k(1-\epsilon)^n \leq |\mathcal{H}|(1-\epsilon)^n$ (poiché $k \leq |\mathcal{H}|$)
- infine, poiché $(1-\epsilon) \leq e^{-\epsilon}$ se $0 \leq \epsilon \leq 1$, abbiamo $|\mathcal{H}|(1-\epsilon)^n \leq |\mathcal{H}|e^{-\epsilon n}$

# Prova...

**Cosa abbiamo ottenuto:** abbiamo un limite superiore alla probabilità che usando un $Tr$ con $n$ esempi, il Version Space contenga qualche ipotesi CATTIVA, cioè che $L$ restituisca $h$, una delle ipotesi cattive, per cui $error_{\mathcal{D}}(h) \geq \epsilon$!

Quindi, se vogliamo che tale probabilità sia inferiore a livello desiderato $\delta$

$$|\mathcal{H}|e^{-\epsilon n} \leq \delta$$

bisogna usare un valore per $n$ per cui

$$n \geq \frac{1}{\epsilon}(ln(|\mathcal{H}|) + ln(\frac{1}{\delta}))$$

o, alternativamente, se si usa un tale valore per $n$, con probabilità $1 - \delta$ l'ipotesi $h$ restituita da $L$ avrà $error_{\mathcal{D}}(h) < \epsilon$

# Apprendimento PAC: esempio $\mathcal{H}_4$

Congiunzione di $m$ letterali

- Spazio delle Istanze $\rightarrow$ stringhe di $m$ bit: $X = \{s | s \in \{0,1\}^m\}$
- Spazio delle Ipotesi $\rightarrow$ tutte le sentenze logiche che riguardano i letterali $l_1, \ldots, l_m$ (anche in forma negata, $\neg l_i$) e che contengono solo l'operatore $\wedge$ (**and**):

$$\mathcal{H} = \{f_{\{i_1,\ldots,i_j\}}(s) | f_{\{i_1,\ldots,i_j\}}(s) \equiv L_{i_1} \wedge L_{i_2} \wedge \cdots \wedge L_{i_j},$$
$$\text{dove } L_{i_k} = l_{i_k} \text{ oppure } \neg l_{i_k}, \{i_1, \ldots, i_j\} \subseteq \{1, \ldots, 2m\}\}$$

Notare che se in una formula un letterale compare sia affermato che negato, allora la formula ha sempre valore di verità $false$ (formula non soddisfacibile)

Quindi, tutte le formule che contengono almeno un letterale sia affermato che negato sono equivalenti alla funzione che vale sempre $false$

# Apprendimento PAC: esempio $\mathcal{H}_4$

E' la congiunzione di $m$ letterali PAC-apprendibile usando **Find-S** con $\mathcal{H}_4$ ?

# Apprendimento PAC: esempio $\mathcal{H}_4$

E' la congiunzione di $m$ letterali PAC-apprendibile usando **Find-S** con $\mathcal{H}_4$ ? Si !

Infatti:

- ogni congiunzione di $m$ letterali è inclusa in $\mathcal{H}_4$

# Apprendimento PAC: esempio $\mathcal{H}_4$

E' la congiunzione di $m$ letterali PAC-apprendibile usando **Find-S** con $\mathcal{H}_4$ ? Si !

Infatti:

- ogni congiunzione di $m$ letterali è inclusa in $\mathcal{H}_4$
- **Find-S** è consistente

# Apprendimento PAC: esempio $\mathcal{H}_4$

E' la congiunzione di $m$ letterali PAC-apprendibile usando **Find-S** con $\mathcal{H}_4$ ? Si !

Infatti:

- ogni congiunzione di $m$ letterali è inclusa in $\mathcal{H}_4$
- **Find-S** è consistente
- $|\mathcal{H}_4| = 3^m + 1$ e poiché $\frac{1}{\epsilon}(ln(3^{m+1}) + ln(\frac{1}{\delta})) > \frac{1}{\epsilon}(ln(3^m + 1) + ln(\frac{1}{\delta}))$

$$n \geq \frac{1}{\epsilon}((m+1)ln(3) + ln(\frac{1}{\delta}))$$

quindi $n$ è polinomiale in $1/\epsilon$, $1/\delta$, $m$, e $size(c)$ (che non compare)

# Apprendimento PAC: esempio $\mathcal{H}_4$

E' la congiunzione di $m$ letterali PAC-apprendibile usando **Find-S** con $\mathcal{H}_4$ ? Si !

Infatti:

- ogni congiunzione di $m$ letterali è inclusa in $\mathcal{H}_4$
- **Find-S** è consistente
- $|\mathcal{H}_4| = 3^m + 1$ e poiché $\frac{1}{\epsilon}(ln(3^{m+1}) + ln(\frac{1}{\delta})) > \frac{1}{\epsilon}(ln(3^m + 1) + ln(\frac{1}{\delta}))$

$$n \geq \frac{1}{\epsilon}((m+1)ln(3) + ln(\frac{1}{\delta}))$$

  quindi $n$ è polinomiale in $1/\epsilon$, $1/\delta$, $m$, e $size(c)$ (che non compare)

- per ogni esempio di apprendimento **Find-S** impiega tempo lineare nella dimensione della ipotesi corrente (e tale dimensione è $\geq size(c)$) e nella dimensione dell'input ($m$), quindi di nuovo polinomiale, e globalmente è polinomiale per $Tr$

# Apprendimento PAC: esempio $\mathcal{H}_4$

E' la congiunzione di $m$ letterali PAC-apprendibile usando **Find-S** con $\mathcal{H}_4$ ? Si !

Infatti:

- ogni congiunzione di $m$ letterali è inclusa in $\mathcal{H}_4$
- **Find-S** è consistente
- $|\mathcal{H}_4| = 3^m + 1$ e poiché $\frac{1}{\epsilon}(ln(3^{m+1}) + ln(\frac{1}{\delta})) > \frac{1}{\epsilon}(ln(3^m + 1) + ln(\frac{1}{\delta}))$

$$n \geq \frac{1}{\epsilon}((m+1)ln(3) + ln(\frac{1}{\delta}))$$

  quindi $n$ è polinomiale in $1/\epsilon$, $1/\delta$, $m$, e $size(c)$ (che non compare)
- per ogni esempio di apprendimento **Find-S** impiega tempo lineare nella dimensione della ipotesi corrente (e tale dimensione è $\geq size(c)$) e nella dimensione dell'input ($m$), quindi di nuovo polinomiale, e globalmente è polinomiale per $Tr$

Quindi tutte le condizioni per la PAC-apprendibilità sono soddisfatte !

# Apprendimento PAC

Usando la disuguaglianza precedente ed altre considerazioni è possibile mostrare che alcune classi di concetti non sono PAC-apprendibili dato uno specifico algoritmo di apprendimento $L$ e $\mathcal{H}$.

In particolare è possibile mostrare che:

- se $\mathcal{H}$ contiene tutte le funzioni booleane definite su $X$ allora $C = \mathcal{H}$ non è PAC-apprendibile da algloritmi consistenti
- esistono classi di concetti $C$ che non sono PAC-apprendibili da algoritmi consistenti che usano $C$ come Spazio delle Ipotesi, tuttavia diventano PAC-apprendibili se uno Spazio delle Ipotesi "più grande" è usato, cioè $C \subset \mathcal{H}$

Il problema con la disuguaglianza data è che questa non può essere usata se $|\mathcal{H}| = \infty$

Tuttavia, il fattore chiave non è quante funzioni diverse sono contenute in $\mathcal{H}$, ma quante funzioni "utili" sono in $\mathcal{H}$: VC-dimension dà una risposta a questa domanda

# Apprendimento PAC

Si può mostrare che, se assumiamo:

- $c \in \mathcal{H}$
- $L$ consistente

allora con probabilità almeno $(1 - \delta)$, l'algoritmo di apprendimento $L$ restituisce una ipotesi $h \in \mathcal{H}$ tale che $error_{\mathcal{D}}(h) \leq \epsilon$ se il numero di esempi di apprendimento $n$ soddisfa la seguente disuguaglianza:

$$n \geq \frac{1}{\epsilon}(4 \log_2(\frac{1}{\delta}) + 8VC(\mathcal{H}) \log_2(\frac{13}{\epsilon}))$$

Notare che $VC(\mathcal{H}) \leq \log_2(|\mathcal{H}|)$

$$VC(\mathcal{H}) \leq \log_2(|\mathcal{H}|)$$

Mostriamo che $VC(\mathcal{H}) \leq \log_2(|\mathcal{H}|)$

- per ogni $S$ tale che $\mathcal{H}$ frammenta $S$ abbiamo $|\mathcal{H}| \geq 2^{|S|}$, infatti $\mathcal{H}$ può implementare tutte le possibili dicotomie di $S$, che sono esattamente $2^{|S|}$.
- scegliendo un $S$ tale che $|S| = VC(\mathcal{H})$, otteniamo $|\mathcal{H}| \geq 2^{VC(\mathcal{H})}$

Quindi, applicando $\log_2$ ad entrambi i lati della disuguaglianza, possiamo concludere che $\log_2(|\mathcal{H}|) \geq VC(\mathcal{H})$

# Mistake Bounds per algoritmi On-line

Quando si considerano algoritmi on-line per l'apprendimento di concetti, è ragionevole essere interessati ad un limite superiore al numero di errori commessi prima di apprendere *esattamente* il concetto target

Il **Modello Mistake Bound** è stato definito per questo scopo:

- le istanze $x_i$ sono presentate ad $L$ una alla volta
- data una istanza $x$, $L$ deve "indovinare" il valore target $c(x)$
- solo dopo, il valore corretto è fornito ad $L$ ai fini dell'apprendimento
- se la predizione di $L$ era sbagliata, allora si ha un errore (mistake)

Bisogna rispondere alla seguente domanda:

> "Quanti errori farà $L$ prima di apprendere esattamente il concetto target ?"

# Mistake bound per la versione on-line di Find-S

L'algoritmo **Find-S** può essere usato in versione on-line !

/* versione on-line di Find-S per congiunzione di $m$ letterali */

inizializza $h$ alla ipotesi più specifica $h \equiv l_1 \wedge \neg l_1 \wedge l_2 \wedge \neg l_2 \wedge \cdots \wedge l_m \wedge \neg l_m$

**do forever** /* in effetti, fino a quando arrivano esempi */

leggi la nuova istanza $x$ e predici $h(x)$

leggi $c(x)$, cioè il valore target per $x$

/* errore ! */
**if** $(h(x) \neq c(x)$ rimuovi da $h$ ogni letterale che non è soddisfatto da $x$

Assumendo $c \in \mathcal{H}$, e assenza di rumore negli esempi

E' possibile dare un limite superiore al numero di errori ?

# Mistake bound per la versione on-line di Find-S

E' possibile dare un limite superiore al numero di errori ?

Si!

- l'ipotesi iniziale contiene $2m$ letterali
- dopo il primo sbaglio (che occorre subito!), solo $m$ letterali rimangono nella ipotesi corrente
- dopo ogni altro errore, almeno un letterale è rimosso dalla ipotesi corrente

Quindi il numero totale di errori che **Find-S** commette prima di convergere alla ipotesi corretta è $\leq m + 1$

# Mistake bound per Halving

L'algoritmo **Halving** è una versione on-line di **Candidate-Elimination**

/* Aggiorna il Version Space ($VS$) on-line */

inizializza $S$ e $G$ come in **Candidate-Elimination**

**do forever** /* in effetti, fino a che $VS$ contiene più di 1 ipotesi */

leggi una nuova istanza $x$ e predici tramite voto a maggioranza delle ipotesi in $VS$

leggi $c(x)$, cioè il valore target per $x$

aggiorna $S$ e $G$ in modo da rimuovere da $VS$ le ipotesi $h$ per cui $h(x) \neq c(x)$

/* un errore occorre solo se la maggioranza delle ipotesi in $VS$ sono sbagliate */

Assumendo $c \in \mathcal{H}$, $|\mathcal{H}| < \infty$, e nessun rumore negli esempi

E' possibile dare un limite superiore al numero di errori ?

# Mistake bound per Halving

E' possibile dare un limite superiore al numero di errori ?

Si!

- il $VS$ iniziale contiene $|\mathcal{H}|$ ipotesi
- dopo un errore (che occorre quando il voto a maggioranza è sbagliato), ALMENO $|VS|/2$ ipotesi sono rimosse da $VS$

Quindi il numero totale di errori che **Halving** commette prima di convergere alla ipotesi corretta è $\leq log_2|\mathcal{H}|$

ATTENZIONE: **Halving** può convergere alla ipotesi corretta senza errori !! Accade quando il voto a maggioranza è sempre corretto e solo le ipotesi minoritarie sono rimosse da $VS$

# Optimal Mistake Bounds

Fino ad ora abbiamo considerato limiti al numero di errori (caso pessimo) per algoritmi *specifici*.

E' possibile dare il più basso fra i limiti di errore su tutti i possibili algoritmi di apprendimento (*optimal mistake bound*) ?

- Assumiamo $C = \mathcal{H}$
- Sia $M_L(c)$ il massimo su tutte le possibili sequenze di esempi di apprendimento del numero di errori commesso da $L$ per apprendere esattamente il concetto $c \in C$
- Sia $M_L(C) \equiv \max_{c \in C} M_L(c)$

**Definizione:** Sia $C$ una classe non vuota di concetti. L' **optimal mistake bound** per C, denotato *Opt(C)*, è il minimo su tutti i possibili algoritmi di apprendimento $L$ di $M_L(C)$:

$$Opt(C) \equiv \min_{L \in AlgoritmiApprendimento} M_L(C)$$

# Optimal Mistake Bounds

Littlestone (1987) ha mostrato che

$$VC(C) \leq Opt(C) \leq M_{Halving}(C) \leq log_2(|C|)$$

# Un esempio finale di Mistake Bound...

Fino ad ora abbiamo visto esempi di algoritmi di apprendimento capaci di trattare con esempi di apprendimento consistenti (cioè che non danno luogo a contraddizioni)

Non è difficile modificare **Halving** in modo da ottenere un algoritmo capace di trattare dati inconsistenti: Algoritmo **Weighted-Majority**

- l'idea base è di assegnare ad ogni ipotesi (ma possiamo estendere l'idea ad ogni insieme di predittori) un peso che è usato per pesare il voto associato ad ogni ipotesi
- quindi, invece di considerare il voto a maggioranza, consideriamo il voto a maggioranza pesata
- quando una ipotesi commette un errore, invece di rimuoverla, si moltiplica il peso a lei associata per un fattore $\beta < 1$ (riduzione del peso)

# Weighted-Majority

/* Weighted-Majority: l' algoritmo Halving... con pesi! */

per ogni ipotesi (o predittore) $h_j$ definire il peso $w_j$, inizialmente uguale a $1$

**do forever** /* in effetti, fino a che ci sono esempi */

leggi una nuova istanza $x$

calcola $q_0 \leftarrow \sum_{j:h_j(x)=0} w_j$ e $q_1 \leftarrow \sum_{j:h_j(x)=1} w_j$

**if** $q_0 > q_1$ **then** predici $0$

**if** $q_1 > q_0$ **then** predici $1$

**if** $q_0 = q_1$ **then** predici $0$ o $1$ a caso

read $c(x)$, i.e. the target value for $x$

**for each** $h_j$ **do**

**if** $h_j(x) \neq c(x)$ **then** $w_j \leftarrow \beta w_j$ /* $0 \leq \beta < 1$ */

Cosa accade se $\beta = 0$ ?

# Mistake bound per Weighted-Majority

**Teorema:** Sia $seq \equiv (x_1, c(x_1)), (x_2, c(x_2)), \ldots$ una qualunque sequenza di esempi di allenamento e sia $k$ il numero minimo di errori commesso su $seq$ da una qualunque ipotesi dello Spazio delle Ipotesi. Allora il numero di errori su $seq$ commesso da **Weighted-Majority** usando $\beta = \frac{1}{2}$ è al più

$$2.4(k + \log_2(|\mathcal{H}|))$$

(lo proviamo!)

Per $0 \leq \beta < 1$, Littlestone e Warmuth (1991) hanno provato che il bound di sopra diventa

$$\frac{k \log_2 \frac{1}{\beta} + \log_2(|\mathcal{H}|)}{\log_2 \frac{2}{1+\beta}}$$

(questo non lo proviamo...)

# Prova...

Per provare il teorema confrontiamo il peso finale $w^*$ della migliore ipotesi $h^*$, cioè quella che produce il numero minore di errori $k$, con la somma finale dei pesi su tutte le ipotesi $W = \sum_{j=1}^{|\mathcal{H}|} w_j$ (naturalmente abbiamo $w^* \leq W$)

- **fatto 1:** $w^* = (\frac{1}{2})^k$, infatti $h^*$ commette esattamente $k$ errori
- **fatto 2:** per ogni errore di **Weighted-Majority**, $W$ si riduce di al più $\frac{3}{4}W$, infatti se **Weighted-Majority** commette un errore, tutte le ipotesi $h_j$ che hanno contribuito alla maggioranza pesata avranno il loro peso ridotto di metà; quindi almeno $\frac{1}{2}W$ (il voto a maggioranza pesata è $\geq \frac{1}{2}W$) del peso totale è ridotto a metà, cioè il nuovo peso totale è **minore o uguale a**

$$\underbrace{\frac{1}{2}W}_{\text{minoranza}} + \underbrace{\frac{1}{2}(\frac{1}{2}W)}_{\text{maggioranza}} = \frac{3}{4}W$$

- **fatto 3:** se $M$ è il numero totale di errori prodotti da **Weighted-Majority**, allora per il peso totale finale $W \leq |\mathcal{H}|(\frac{3}{4})^M$, a causa del fatto 2 e la condizione iniziale $W = |\mathcal{H}|$

# Prova...

Ricordando che $w^* \leq W$, ed a causa dei fatti 1-3, abbiamo

$$(\tfrac{1}{2})^k = w^* \leq W \leq |\mathcal{H}|(\tfrac{3}{4})^M$$

Prendendo il logaritmo (in base $2$) dei termini più a sinistra e più a destra, otteniamo

$$k \log_2(\tfrac{1}{2}) \leq \log_2(|\mathcal{H}|) + M \log_2(\tfrac{3}{4})$$

ed isolando $M$ otteniamo (notare che $\log_2(\frac{3}{4})$ è una quantità negativa)

$$M \leq \frac{k + \log_2(|\mathcal{H}|)}{-\log_2(\frac{3}{4})} \leq 2.4(k + \log_2(|\mathcal{H}|))$$