Complessità dello Spazio delle Ipotesi

Richiamo

Supervised Learning
(apprendimento supervisionato)

$\{(x_i, f(x_i))\}$

- dato un insieme di dati preclassificati (esempi di apprendimento) apprendere una descrizione generale che incapsula l'informazione contenuta negli esempi
- tale descrizione deve poter essere usata in modo predittivo

  dato un nuovo ingresso $\tilde{x}$, predire $f(\tilde{x})$
- si assume che un esperto (o maestro) ci fornisca la supervisione

  i valori $f(x_i)$

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

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

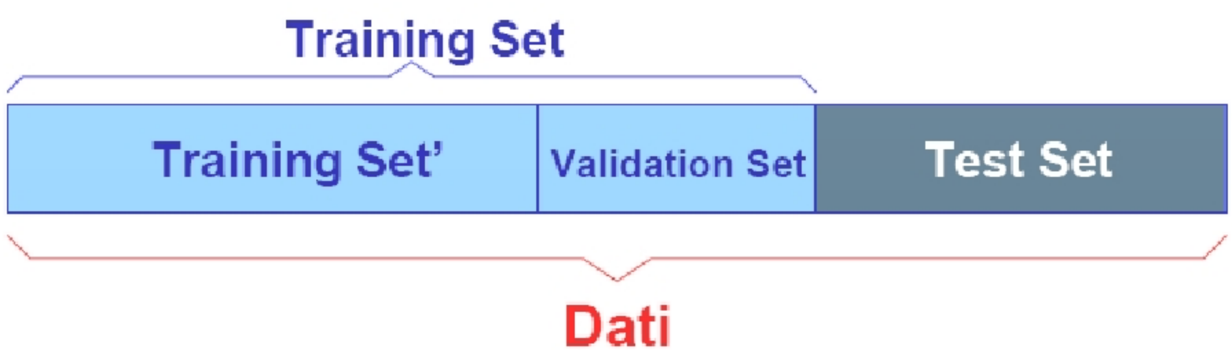
Dati (cont.)

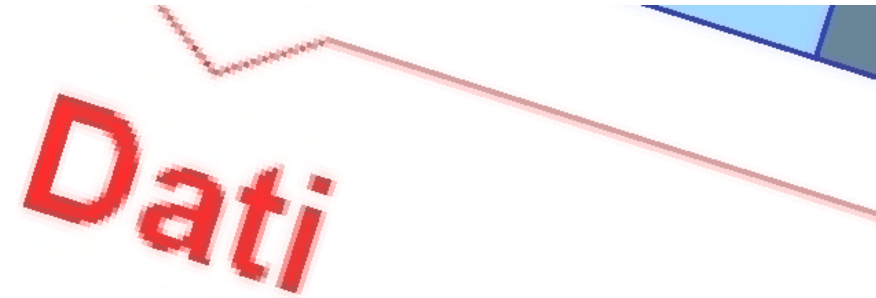
Se N abbastanza grande il Training Set è ulteriormente suddiviso in due sottoinsiemi $(N_{tr} = N_{\widehat{tr}} + N_{val})$:

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

- $\boxed{\text{Validation Set} = \{(x^{(1)}, f(x^{(1)})), \ldots, (x^{(N_{val})}, f(x^{(N_{val})}))\}}$
  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'





evitare l'overfitting!

Soluzione

utilizzare uno spazio delle ipotesi che non sia

- né troppo semplice (underfitting)
- né troppo complesso (overfitting)

Occorre "misurare" la complessità dello spazio delle ipotesi

Soluzione

utilizzare uno spazio delle ipotesi che non sia

- né troppo semplice (underfitting)
- né troppo complesso (overfitting)

Occorre "misurare" la complessità dello spazio delle ipotesi

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 $\mathcal{H}$ defi nito su uno spazio delle istanze X è data dalla cardinalità del sottoinsieme più grande di X che è frammentato da $\mathcal{H}$:

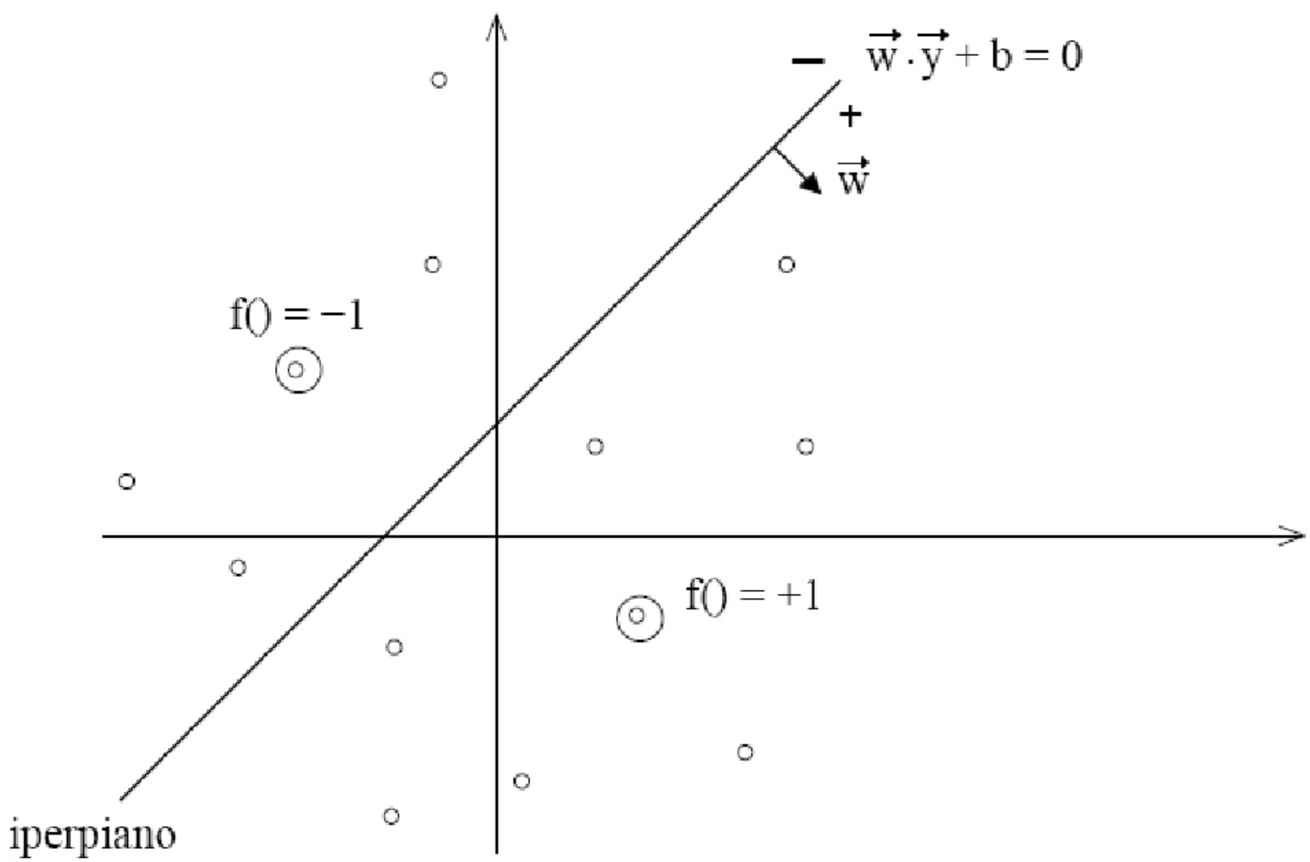
$$VC(\mathcal{H}) = \max_{S \subseteq X} |S| \; : \; \mathcal{H} \text{ frammenta } S$$

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

VC-dimension: Esempio

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

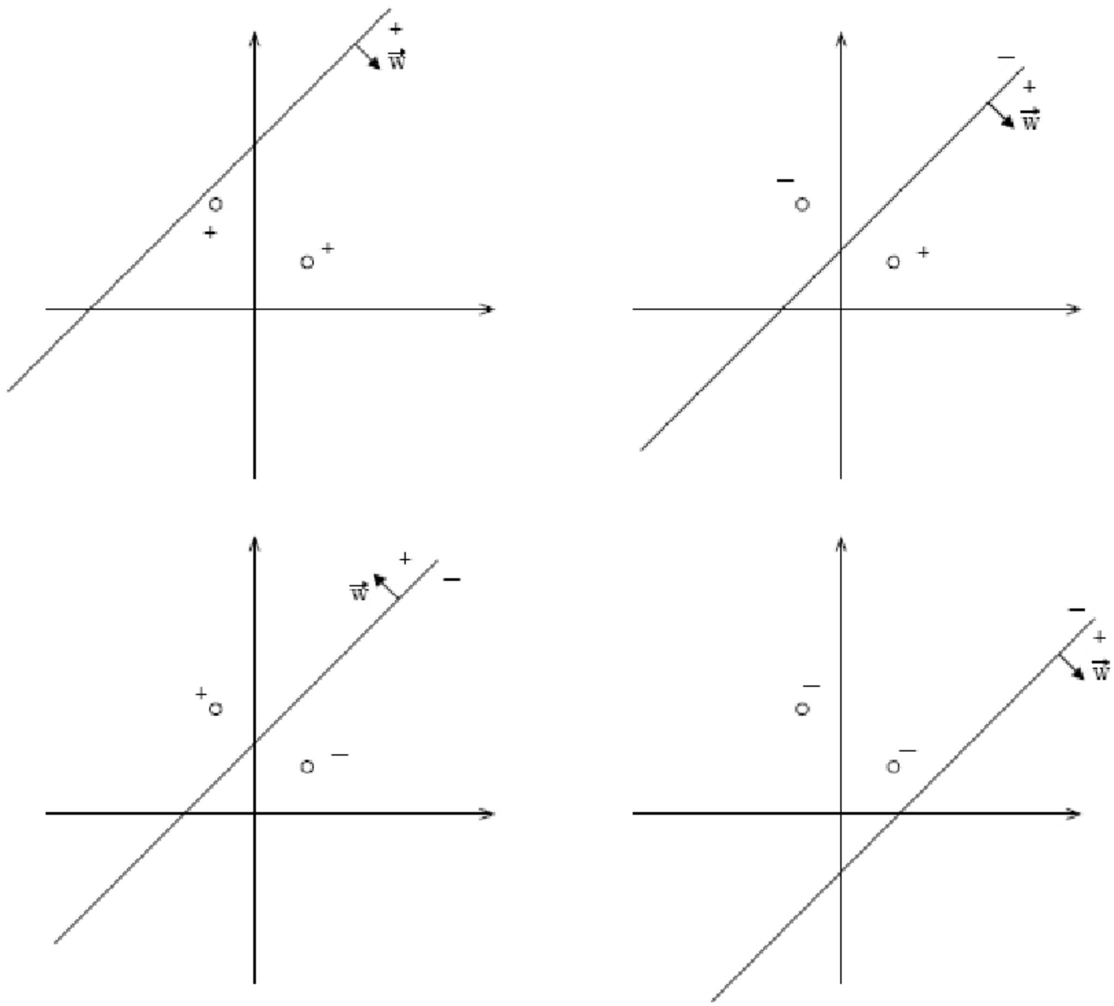
$$\mathcal{H}_1 = \{f_{(\vec{w},b)}(\vec{y}) | f_{(\vec{w},b)}(\vec{y}) = sign(\vec{w} \cdot \vec{y} + b), \vec{w} \in \mathbb{R}^2, b \in \mathbb{R}\}$$



VC-dimension: Esempio

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

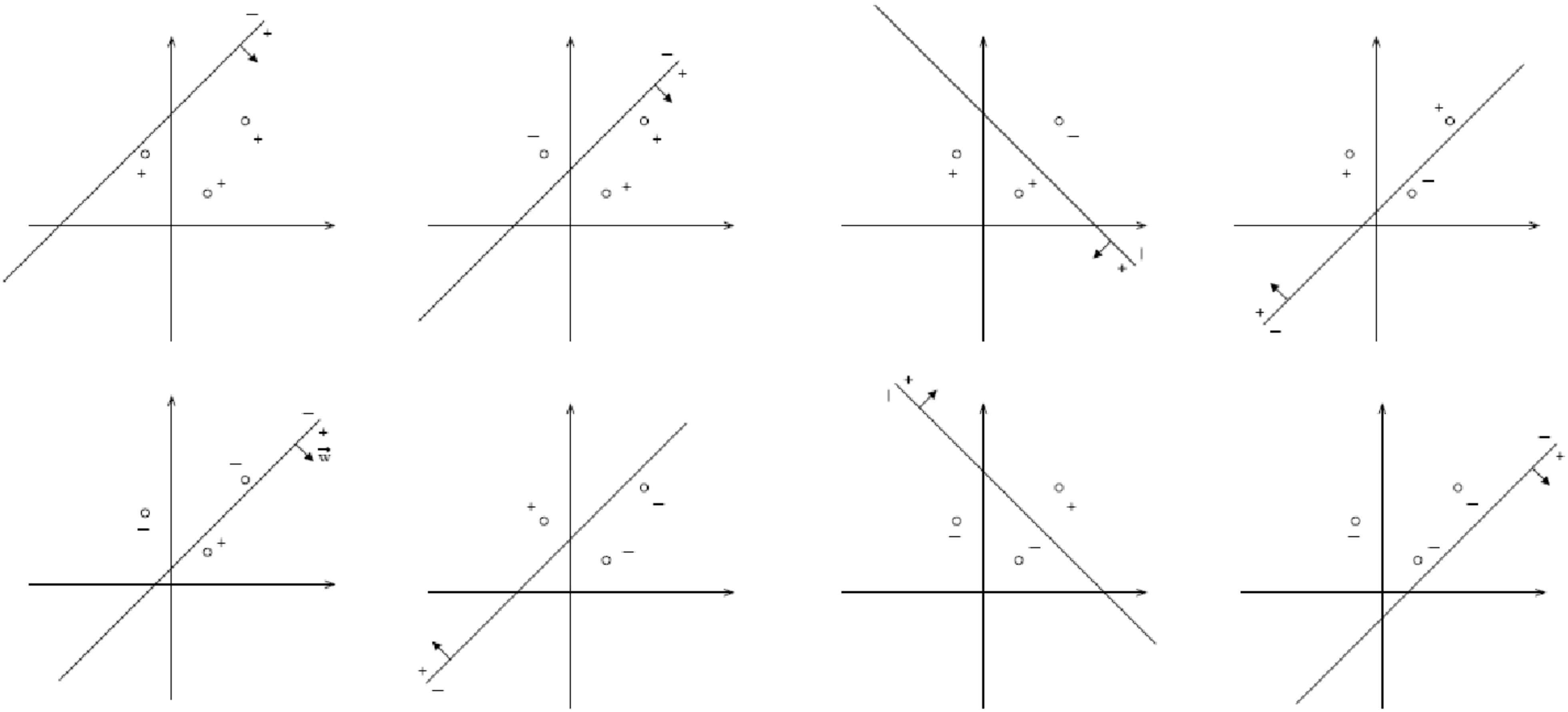
$VC(\mathcal{H}) \geq 1$ banale. Vediamo cosa succede con 2 punti:



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 $\mathcal{H}_1$?

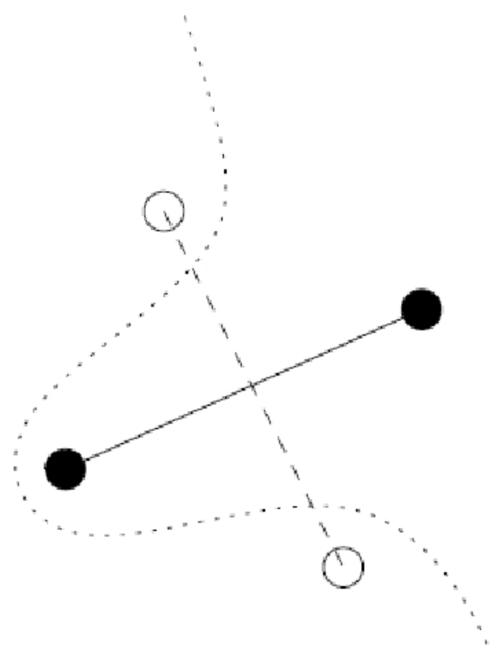
Quindi $VC(\mathcal{H}) \geq 3$. Cosa succede con 4 punti ?

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:

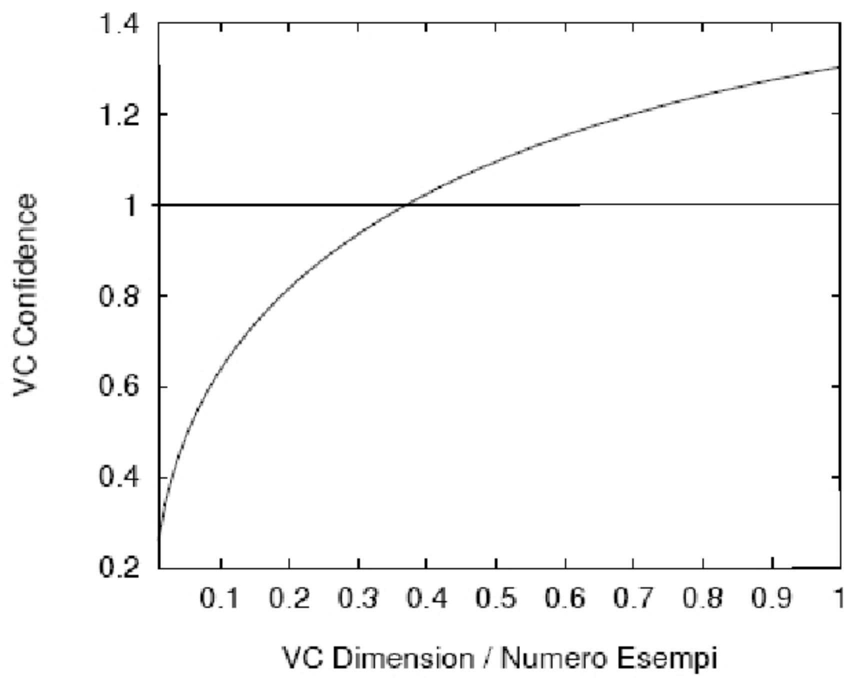
$$error_{\mathcal{D}}(h_{\boldsymbol{w}^*}(\boldsymbol{x})) \leq \underbrace{error_{Tr}(h_{\boldsymbol{w}^*}(\boldsymbol{x}))}_{A} + \underbrace{\sqrt{\frac{VC(\mathcal{H})}{N_{tr}}\left(\log\left(\frac{2N_{tr}}{VC(\mathcal{H})}\right) + 1\right) - \frac{1}{N_{tr}}\log(\delta)}}_{B}$$

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 $VC(\mathcal{H})$.



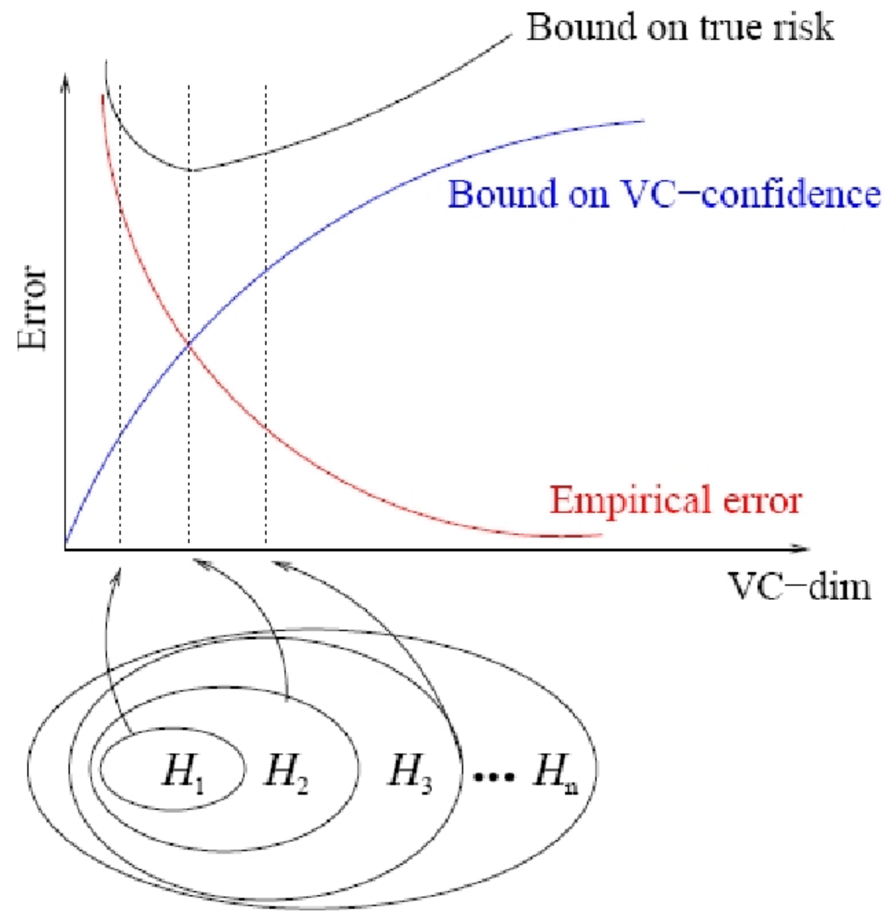
Structural Risk Minimization

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

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 $\boxed{h_0 \leq_g h_1 \leq_g h_2 \leq_g h_3 \leq_g h_4 \leq_g h_5}$

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 $h \equiv c$.