

# Appunti di Introduzione all’Intelligenza Artificiale Unipi - 3

## Parte

Raffaele Apetino

Marzo 2020

## Contents

<b>1</b>	<b>Introduzione - Machine Learning</b>	<b>3</b>
1.1	Un esempio concreto . . . . .	3
1.2	Quando usare il machine learning? . . . . .	3
1.3	Perché usare il machine learning? . . . . .	3
<b>2</b>	<b>Overview di un sistema ML</b>	<b>4</b>
2.0.1	Riconoscimento caratteri . . . . .	4
2.1	TASK: Supervised Learning . . . . .	5
2.2	TASK: UNsupervised Learning . . . . .	5
2.3	MODEL . . . . .	6
2.3.1	Esempi di Modelli . . . . .	6
2.4	LEARNING ALGORITHM . . . . .	6
2.5	VALIDATION - Generalizzazione . . . . .	7
<b>3</b>	<b>Concept Learning</b>	<b>7</b>
3.0.1	Esempio: Apprendimento Funzione Booleana . . . . .	8
3.1	Regole Congiuntive . . . . .	8
3.2	Problema: Enjoy Sport . . . . .	9
3.3	Rappresentare le Ipotesi . . . . .	9
3.4	TASK del Concept Learning . . . . .	9
3.5	Apprendimento Induttivo . . . . .	10
3.6	Qual é il numero di istanze, concetti e ipotesi di un problema? . . . . .	10
3.7	Da ordine generale a ordine specifico . . . . .	10
3.8	Algoritmo Find-S (Specific) . . . . .	11
3.8.1	Esempio su Enjoy Sport . . . . .	12
3.8.2	Proprietà di Find-S . . . . .	12
3.8.3	Aspetti negativi di Find-S . . . . .	12
3.9	Version Spaces . . . . .	12
3.9.1	Esempio di Version Space . . . . .	13
3.10	Algoritmo List-Then-Eliminate . . . . .	13
3.11	Algoritmo Candidate Elimination . . . . .	13
3.11.1	Esempio di Candidate Elimination . . . . .	15

<b>4 Inductive Bias</b>	<b>15</b>
4.1 Sistema di apprendimento Unbiased . . . . .	16
4.1.1 Il learning Unbiased è utile? . . . . .	16
4.2 Definizione Formale di Inductive Bias . . . . .	16
4.3 Sistemi Induttivi e Deduttivi equivalenti . . . . .	17
4.4 Tre (algoritmi) sistemi di apprendimento con Bias differenti . . . . .	17
<b>5 Modelli Lineari</b>	<b>18</b>
5.1 Esempio di regressione . . . . .	18
5.2 Modello di Regressione Lineare Semplice (Univariata) . . . . .	18
5.3 Costruzione della funzione via LMS . . . . .	19
5.3.1 Gradiente discendente . . . . .	21
5.4 Modello di Regressione Lineare Multivariato . . . . .	22
5.4.1 Notazione dei dati . . . . .	22
5.5 Hyperplane . . . . .	23
5.6 Gradiente Discendente Algoritmo . . . . .	23
5.7 Vantaggi dei modelli lineari . . . . .	24
5.8 Limitazioni . . . . .	24
5.8.1 Esempio . . . . .	25
5.9 Linear Basis Expansion . . . . .	25
5.9.1 Esempi . . . . .	25
5.9.2 Tradeoff per la complessità . . . . .	26
5.10 Regolarizzazione - Ridge Regression . . . . .	27
5.11 Limitazione delle funzioni a base fissa . . . . .	28
5.12 Classificazione con il modello lineare . . . . .	28
5.12.1 Visione geometrica dell'iperpiano . . . . .	29
5.12.2 Classificazione attraverso Decision Boundary lineare . . . . .	30
5.12.3 Esempio Terremoti/Esplosioni Nucleari . . . . .	30
5.12.4 Esempio Email Spam . . . . .	30
5.13 Il problema di apprendimento (per classificatori lineari) . . . . .	31
5.13.1 $\Delta w$ come regola della correzione dell'errore . . . . .	31
5.14 Classificazione dei pattern . . . . .	31
5.15 Congiunzioni nel caso di Modello Lineare . . . . .	32
5.15.1 Limitazioni . . . . .	32
5.16 Altri sistemi di apprendimento per la classificazione . . . . .	33
5.17 Conclusioni sui Modelli Lineari . . . . .	33

# 1 Introduzione - Machine Learning

Il problema dell'apprendimento è uno dei problemi principali dell'intelligenza sia artificiale che biologica. L'apprendimento è un obiettivo complesso con un campo di ricerca in continua crescita. L'apprendimento automatico (Machine Learning) è emerso come un'area di ricerca che combina gli obiettivi della creazione di macchine in grado di apprendere e strumenti statistici/adattivi. Perché le macchine dovrebbero imparare da sole? Poiché c'è una crescente necessità di analizzare dati empirici difficili da gestire con la normale programmazione (paradigma DATA-DRIVEN). Il compito dell'apprendimento automatico è la creazione di sistemi intelligenti adattivi, in grado di analizzare dati.

## 1.1 Un esempio concreto

Esempi concreti di applicazione di ML possono essere la classificazione delle email spam oppure il riconoscimento di caratteri scritti a mano, dei visi o del parlato. Non ci sono regole o conoscenza pregressa per trovare la soluzione, ma diventa più facile avendo una fonte di esperienza per apprendere (dati di cui conosciamo già il risultato). Ad esempio nel riconoscimento facciale si combinano reti neurali e altri approcci di ML partendo da milioni di immagini facciali appartenenti a diversi individui.

## 1.2 Quando usare il machine learning?

Il machine learning è una grande opportunità ma deve essere controllato. Usiamo l'apprendimento automatico quando:

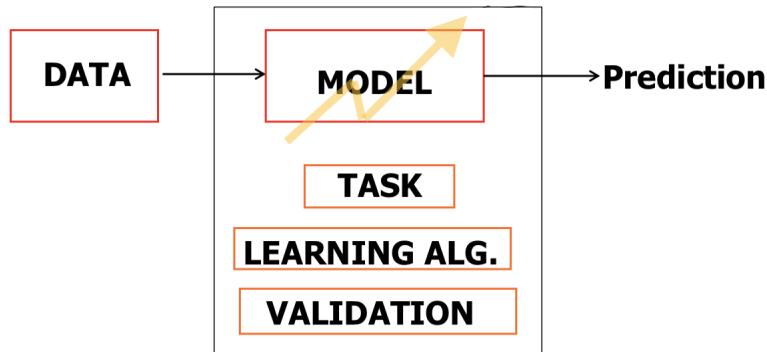
- non abbiamo conoscenza per spiegare il fenomeno oppure è difficile formalizzarlo
- abbiamo dati incerti, rumorosi o incompleti, che ostacolano la formalizzazione di soluzioni
- ci troviamo in ambienti dinamici non conosciuti in anticipo

Abbiamo però anche dei requisiti per poter applicare ML ai nostri problemi. È necessaria una fonte di esperienza formativa ma è difficile raccogliere molti dati rappresentativi, per esempio per una malattia rara. Dobbiamo anche considerare una tolleranza sulla precisione dei dati (se accettiamo di dare pochi dati in input, o poco significativi, dobbiamo anche tollerare una certa percentuale di errore, che però in alcuni casi non può essere accettata).

## 1.3 Perché usare il machine learning?

È una opportunità per conoscere nuovi paradigmi informatici con un approccio differente dalla programmazione standard, algoritmica e dall'IA classica. Serve per trovare soluzioni approssimate a problemi molto difficili, ma non è una metodologia approssimativa, è un approccio rigoroso per trovare funzioni approssimative per affrontare problemi complessi.

## 2 Overview di un sistema ML

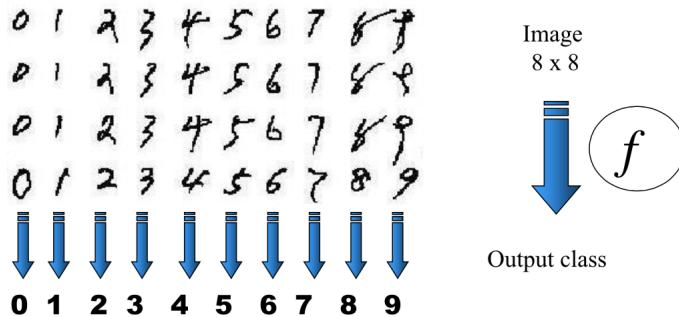


- DATA: i dati li ricaviamo dalle osservazioni del mondo
- MODEL: i dati passano attraverso un modello. Il modello non è fissato: ha dei parametri liberi, modificabili per fornire le predizioni in modo che siano quelle che vorremmo che fossero fornite.
  - TASK: Supervised o Unsupervised Learning
  - LEARNING ALGORITHM: algoritmo che cerca la soluzione in uno spazio di stati
  - VALIDATION: valutazione statistica del modello (quanto riesce ad essere accurato sui dati futuri).
- PREDICTION: è l'output restituito dal modello

"Apprendere" viene visto come una approssimazione di una funzione sconosciuta ricavata dagli esempi.

### 2.0.1 Riconoscimento caratteri

Un esempio "pilota" che possiamo fare è il riconoscimento di caratteri scritti a mano. Il problema è racchiuso nel riconoscere una funzione che passa per un insieme di punti. Come input abbiamo una collezione di immagini di caratteri scritti a mano. Il problema è costruire un modello che riceve in input queste matrici di 8x8 pixel e come output restituire la cifra riconosciuta.



Non sappiamo quale sia la funzione, perché in quel caso la scriveremmo direttamente in un algoritmo. Cerchiamo quindi di approssima una funzione facendo restituire come output una classificazione della matrice di pixel. Il problema di classificazione è dato dalla formalizzazione della soluzione esatta, poiché ci possiamo trovare con dati rumorosi o ambigui. Risulta però facile raccogliere collezioni di

esempi etichettati, cioè esempi con soluzioni associate.

Generalizzando il problema possiamo usare la tecnica del Supervised Learning. In particolare abbiamo in input uno spazio  $x$  di dati etichettati, dobbiamo costruire una funzione generale a partire dagli esempi e come output dare una categoria o valori reali.

## 2.1 TASK: Supervised Learning

Il Supervised Learning è una tecnica di apprendimento automatico che mira a istruire un sistema informatico in modo da consentirgli di elaborare automaticamente previsioni sui valori di uscita di un sistema rispetto ad un input sulla base di una serie di esempi ideali, costituiti da coppie di input e di output, che gli vengono inizialmente forniti.

Quindi vengono dati esempi di training nella forma di coppie  $\langle \text{input}, \text{output} \rangle = \langle x, d \rangle$  (esempi etichettati) per la creazione di una funzione sconosciuta  $f$ . Definiamo il valore target come il valore "d" che vogliamo ottenere come output (e che ci viene dato tramite gli esempi).

Bisogna trovare una buona approssimazione di  $f$ , cioè una ipotesi  $h$  che può essere usata per predire l'output sui dati sconosciuti  $x'$ .

L'obiettivo è dare in output una etichetta numerica o la classificazione del dato.

- Classificazione: la funzione a valori discreti  $f(x)$  restituisce la presunta classe corretta per  $x$ .
- Regressione: consiste nell'approssimare a valori reali la funzione target.

Entrambi sono compiti di approssimazione di funzione.

## 2.2 TASK: UNsupervised Learning

L'apprendimento non supervisionato è una tecnica di apprendimento automatico che consiste nel fornire al sistema informatico una serie di input (esperienza del sistema) che egli riclassificherà ed organizzerà sulla base di caratteristiche comuni per cercare di effettuare ragionamenti e previsioni sugli input successivi. Abbiamo quindi un TR set che è un insieme di dati non etichettati, il compito è quello di raggruppare i dati in insiemi consistenti. I principali algoritmi sono:

- Clustering: raggruppamento di elementi omogenei in un insieme di dati (cluster) identificando un centroide.
- Dimensionality reduction / Visualization / Preprocessing
- Modeling the data density

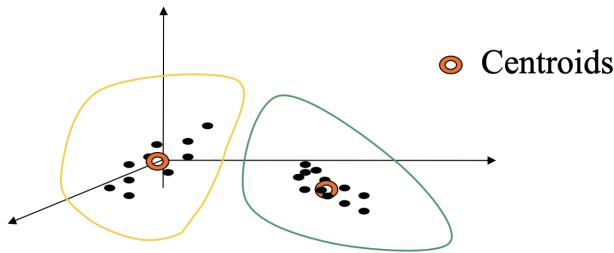


Figure 1: Esempio di clustering

## 2.3 MODEL

Il compito è quello di catturare e descrivere le relazioni tra i dati sulla base del TASK. Il modello definisce la classe delle funzioni che la macchina può implementare (cioè definisce lo spazio delle ipotesi).

Concetti utili:

- Training examples: fanno parte del Supervised Learning ed è un insieme di esempi nella forma  $(x, f(x))$  dove  $x$  è un vettore di caratteristiche e  $f(x)$  è il valore obiettivo.
- Target function: la funzione obiettivo  $f$  il più possibile corretta
- Hypothesis: è una funzione  $h$  che è ritenuta di essere simile a  $f$ . La funzione  $h$  è data in un linguaggio capace di esprimere le relazioni tra i dati.
- Hypotheses space: è lo spazio di tutte le ipotesi che possono essere output dell'algoritmo. È dove cerchiamo la funzione  $h$ .

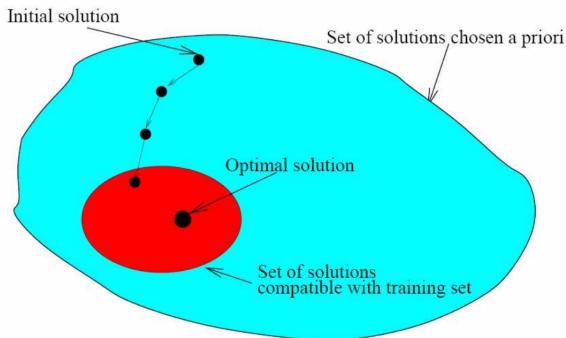
I linguaggi per le ipotesi  $h$  devono saper esprimere modelli di ML: ad esempio la logica del primo ordine, equazioni numeriche oppure calcolo delle probabilità.

### 2.3.1 Esempi di Modelli

- Modelli Lineari: la rappresentazione di  $H$  definisce uno spazio continuo parametrizzato di potenziali ipotesi. Come parametro abbiamo  $w$ , ed ogni assegnamento di  $w$  è una ipotesi differente. (es:  $h_w(x) = w_1x + w_0$ )
- Regole simboliche: lo spazio delle ipotesi è basato su rappresentazioni discrete, sono possibili differenti regole come ad esempio: if  $(x_1=0 \text{ and } x_2=1)$  then  $h(x)=1$  else  $h(x)=0$
- Modelli probabilistici: stimare  $p(x,y)$
- Approcci basati sull'istanza: confronta le nuove istanze del problema con le istanze osservate durante l'addestramento, che sono state memorizzate. Si chiama basato sull'istanza perché costruisce ipotesi direttamente dalle istanze di addestramento stesse.

## 2.4 LEARNING ALGORITHM

Gli algoritmi di apprendimento si basano su DATA, TASK e MODEL. L'euristica consiste nella ricerca delle migliori ipotesi nello spazio delle ipotesi  $H$ , cioè la miglior approssimazione della funzione obiettivo  $f$  sconosciuta (ad esempio i parametri liberi del modello vengono adattati al compito da svolgere: il miglior  $w$  nel modello lineare, la miglior regola per regole simboliche, ecc...).  $H$  potrebbe non coincidere con l'insieme di tutte le possibili funzioni e la ricerca può non risultare esaustiva, per questo bisogna fare assunzioni (vedremo in seguito il ruolo dell'Inductive Bias).



## 2.5 VALIDATION - Generalizzazione

L'apprendimento può essere quindi definito come la ricerca una buona funzione, in uno spazio di funzioni restituito da dati conosciuti. "Buona funzione" considerando l'errore di generalizzazione (o capacità di generalizzazione) cioè quanto accuratamente il modello prevede nuovi campioni di dati. La generalizzazione è cruciale in ML → strumenti corretti di ML.

- Fase di apprendimento (training, fitting): costruire il modello dai dati di training conosciuti (Training Set).
- Fase predittiva (test): si applica la funzione costruita a un nuovo esempio. Si prende in input  $x$ , si computa la risposta dal modello e confrontiamo l'output con il nostro target (che il modello non ha mai visto). Si esegue una valutazione dell'ipotesi predittiva, ad esempio la valutazione della capacità di generalizzazione (valutazione statistica del modello, quanto riesce ad essere accurato sui dati futuri). È bene notare che essere performanti in ML significa predire accuratamente, cioè la performance è stimata dagli errori effettuati calcolando la funzione sul Test Set (che è diverso dal Training set!).

## 3 Concept Learning

Il Concept Learning anche noto come apprendimento di categoria è definito come "la ricerca di attributi che possono essere utilizzati per distinguere i modelli dai non modelli di varie categorie". Più semplicemente, i concetti sono le categorie che ci aiutano a classificare oggetti, eventi o idee, basandoci sul fatto che ogni oggetto, evento o idea ha un insieme di caratteristiche rilevanti comuni. In un compito di Concept Learning, una macchina viene addestrata per classificare gli oggetti mostrandole una serie di oggetti di esempio insieme alle loro etichette di classe. La macchina semplifica ciò che ha osservato "condensandolo" sotto forma di un esempio. Questa versione semplificata di ciò che è stato appreso viene quindi applicata a esempi futuri.

Lavoriamo in uno spazio discreto strutturato come lo spazio delle ipotesi, useremo come rappresentazione la congiunzione di letterali e vedremo algoritmi come Find-S ed Candidate Elimination. Praticamente in questa sezione andremo ad vedere in cosa consiste la classificazione del Supervised Learning, cioè come la funzione  $f(x)$  restituisce la presunta classe corretta per  $x$ .

Concept Learning: dedurre una funzione booleana (con dominio  $X$  e codominio  $\{t,f\}$ ) da esempi di allenamento positivi e negativi (dove  $X$  è lo spazio che contiene le istanze)

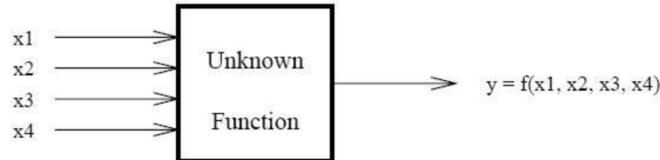
Un Training example è definito come coppia  $\langle x, c(x) \rangle$  (le coppie sono contenute nel Training Set). L'ipotesi  $h: X \rightarrow 0, 1$  soddisfa  $x$  se  $h(x)=1$ .

Una ipotesi  $h$  è consistente con un esempio di allenamento  $\langle x, c(x) \rangle$  se  $h(x)=c(x)$  con  $x$  appartenente a  $X$ . È consistente anche con  $D^1$  se  $h(x)=c(x)$  per ogni esempio di allenamento  $\langle x, c(x) \rangle$  in  $D$ .

---

<sup>1</sup>è sempre il Training Set insieme degli esempi

### 3.0.1 Esempio: Apprendimento Funzione Booleana



Example	$x_1$	$x_2$	$x_3$	$x_4$	$y$
1	0	0	1	0	0
2	0	1	0	0	0
3	0	0	1	1	1
4	1	0	0	1	1
5	0	1	1	0	0
6	1	1	0	0	0
7	0	1	0	1	0

Questo é un problema mal posto poiché potremmo violare l'esistenza, unicità o stabilità della soluzione. Come possiamo vedere dall'immagine abbiamo in input 4 variabili e in output un singolo valore. Nella tabella troviamo il nostro training set ( $D$ ) e il nostro compito é quello di trovare la funzione che dati in input quei valori restituisce l'output definito dalla tabella. Ci sono  $2^{2^4}$  possibili funzioni booleane per 4 ingressi, questo perché abbiamo  $2^4$  possibili ingressi e per ognuno possiamo rispondere 0 o 1. Non possiamo sapere quale sia la funzione corretta fino a che non abbiamo visto tutte le possibili coppie di input/output. Dopo aver analizzato i nostri 7 esempi ci rimangono ancora  $2^9$  possibilità. Nel caso generale per input/output binari la formula é  $|H| = 2^{\text{numero istanze}} = 2^n$  dove  $|H|$  é la dimensione dello spazio delle ipotesi ed  $n$  é la dimensione dell'input.

Lavoreremo con uno spazio delle ipotesi ristretto: partiamo scegliendo uno spazio delle ipotesi  $H$  che é considerevolmente più piccolo dello spazio di tutte le possibili funzioni. Vedremo:

- proposizioni formate da solo "and" (regole congiuntive) in uno spazio di ipotesi  $H$  finito e discreto
- funzioni lineari, in uno spazio di ipotesi  $H$  continuo e infinito

### 3.1 Regole Congiuntive

Quante h diverse possiamo avere? Cioè quante semplici regole congiuntive?

Nel caso generale:

- Letterali positivi: ad esempio  $h_1 = l_2$ ,  $h_2 = (l_1 \wedge l_2)$ ,  $h_3 = \text{true}$  ecc... sono semplici regole congiuntive. Abbiamo ridotto lo spazio delle ipotesi così da ottenere  $|H| = 2^n$  basti immaginare  $l_i$  come una stringa di bit di lunghezza  $n$ .
- Letterali: (di cui fa parte anche il  $\text{not}(l_i)$ ) ottenendo uno spazio delle ipotesi di dimensione  $3^n + 1$

Vedremo come organizzare e cercare in modo efficiente attraverso uno spazio di ipotesi, sfruttando algoritmi per un insieme di ipotesi molto limitato semplificandoci il lavoro usando dati non rumorosi.

### 3.2 Problema: Enjoy Sport

Concept: "giorni in cui piace fare sport acquatici"

Task: predire il valore di "Enjoy Sport" per un giorno arbitrario in base ai valori degli attributi.

Sky	Temp	Humid	Wind	Water	Forecast	Enjoy Sport?
Sunny	Warm	Normal	Strong	Warm	Same	Yes
Sunny	Warm	High	Strong	Warm	Same	Yes
Rainy	Cold	High	Strong	Warm	Change	No
Sunny	Warm	High	Strong	Cool	Change	Yes

Una riga della tabella è una istanza.

### 3.3 Rappresentare le Ipotesi

L'ipotesi  $h$  è una congiunzione di vincoli sugli attributi. Ogni vincolo può essere:

- Uno specifico valore: ad esempio Water = Warm
- Un valore irrilevante: ad esempio Water = ?
- Nessun valore consentito (ipotesi nulla): ad esempio Water = 0

Esempio di ipotesi  $h$ :

Sky	Temp	Humid	Wind	Water	Forecast
Sunny	?	?	Strong	?	Same

corrispondente alla funzione booleana

$$Sky = Sunny \wedge Wind = Strong \wedge Forecast = Same$$

L'ipotesi più specifica (risponde sempre false) e più generale (risponde sempre true) sono rispettivamente

Sky	Temp	Humid	Wind	Water	Forecast
0	0	0	0	0	0 //specifico
?	?	?	?	?	? //generale

### 3.4 TASK del Concept Learning

Vengono dati:

- Le istanze  $X$ : i possibili giorni vengono descritti da attributi Sky, Temp, Humid, Wind, Water e Forecast.
- Funzione Target:  $c: EnjoySport : X \rightarrow \{0, 1\}$ .
- Spazio delle ipotesi ( $H$ ): insieme finito di coniuzioni di letterali
- Esempi di allenamento ( $D$ ): esempi positivi e negativi:  $\langle x_1, c(x_1) \rangle, \dots, \langle x_n, c(x_n) \rangle$

Bisogna trovare:

- Una ipotesi  $h$  in  $H$  tale che  $h(x)=c(x)$  per tutti gli  $x$  in  $X$

L'apprendimento sta nella ricerca nello spazio delle ipotesi  $H$ .

### 3.5 Apprendimento Induttivo

Assunzione: "Qualsiasi ipotesi trovata per approssimare la funzione target sugli esempi di allenamento, approssimerà anche la funzione target sugli esempi non osservati". Non è detto che il Training Set sia corretto (dati rumorosi, ...)

Quindi  $h(x)=c(x)$  per ogni  $x$  in  $D$  (funzione consistente con il training set) ma  $h(x)=c(x)$  per ogni  $x$  in  $X$ ? Problema fondamentale del ML.

### 3.6 Qual é il numero di istanze, concetti e ipotesi di un problema?

- Sky: Sunny, Cloudy, Rainy
- Temp: Warm, Cold
- Humid: Normal, High
- Wind: Strong, Weak
- Water: Warm, Cold
- Forecast: Same, Change

La scelta della rappresentazione di  $H$  determina lo spazio di ricerca!

Numero di possibili istanze:  $3*2*2*2*2*2 = 96$

Numero di concetti distinti:  $2^96 = 2^{numero\ istanze}$

Numero di ipotesi sintatticamente distinte:  $5*4*4*4*4*4$  (poiché per ogni attributo devo aggiungere 0 o ?)

Numero di ipotesi semanticamente distinte:  $1+4*3*3*3*3*3$  (poiché le ipotesi con almeno uno 0 sono equivalenti a false, i valori sono in and!)

Strutturare uno spazio di ricerca può aiutare a cercare in modo piú efficiente.

### 3.7 Da ordine generale a ordine specifico

Consideriamo due ipotesi

1.  $h_1 = < Sunny, ?, ?, Strong, ?, ?, ? >$
2.  $h_2 = < Sunny, ?, ?, ?, ?, ?, ? >$

L'insieme di istanze coperte da  $h_1$  e da  $h_2$  sono differenti, infatti  $h_2$  impone meno vincoli rispetto a  $h_1$  e quindi classifica più istanze x positive ( $h(x)=1$ ).

Siano  $h_j$  e  $h_k$  funzioni booleane definite su  $X$ .  $h_j$  é piú generale o equivalente a  $h_k$  ( $h_j \geq h_k$ ) se solo se  $\forall x \in X : [(h_k(x) = 1) \rightarrow (h_j(x) = 1)]$

Un esempio sulle ipotesi binarie:  $l_1 \geq (l_1 \wedge l_2)$  non sono comparabili, la relazione  $\geq$  impone un ordine parziale sullo spazio delle ipotesi  $H$ . Possiamo sfruttare questo ordine parziale per organizzare più efficientemente la nostra ricerca in  $H$ .

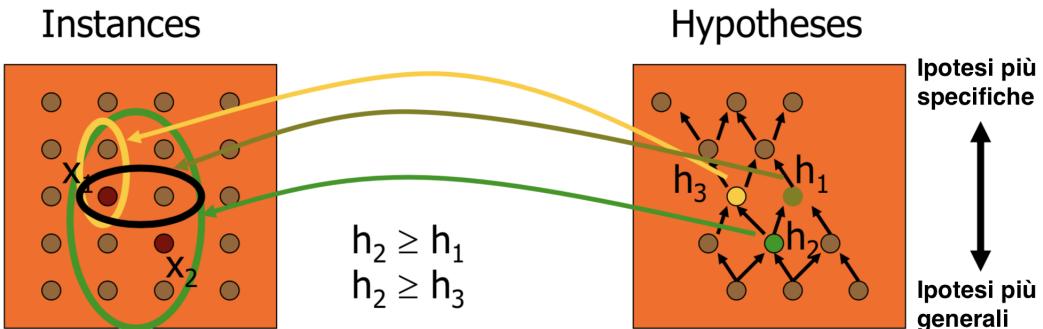


Figure 2: Partial Order

Istanze:

- $x_1 = < \text{Sunny}, \text{Warm}, \text{High}, \text{Strong}, \text{Cool}, \text{Same} >$
- $x_2 = < \text{Sunny}, \text{Warm}, \text{High}, \text{Light}, \text{Warm}, \text{Same} >$

Ipotesi:

- $h_1 = < \text{Sunny}, ?, ?, \text{Strong}, ?, ? >$
- $h_2 = < \text{Sunny}, ?, ?, ?, ?, ? >$
- $h_3 = < \text{Sunny}, ?, ?, ?, \text{Cool}, ? >$

### 3.8 Algoritmo Find-S (Specific)

L'algoritmo Find-S sfrutta l'ordine parziale di gestione per cercare in modo efficiente una  $h$  consistente (senza elencare esplicitamente ogni  $h$  in  $H$ ).

1. Inizializza  $h$  con l'ipotesi più specifica nello spazio delle ipotesi  $H$  ( $h_0 = < 0, 0, 0, 0, 0, 0 >$ )
2. Prendo il prossimo esempio di training:

- Per ogni attributo  $a_i$  dell'ipotesi  $h$ , se  $a_i$  è soddisfatto in  $h$  da  $x$  allora non fare nulla (quindi se l'esempio è negativo non ci sono cambiamenti sull'ipotesi), altrimenti se  $a_i$  NON è soddisfatto e l'esempio è positivo, sostituisce  $a_i$  con il prossimo vincolo più generale soddisfatto da  $x$ .  
Ripeto il passaggio per ogni esempio positivo.

3. Viene dato in output l'ipotesi  $h$

L'algoritmo parte dall'ipotesi più specifica, poi scende in maniera conservativa (generalizza il minimo possibile) in modo da rimanere consistente sui positivi, e di conseguenza contemporaneamente sui negativi. Trovo un'ipotesi che copre tutti i positivi e tutti i negativi. In questo modo evito di esplorare tutto  $H$ , scorrendo solo una volta il training set, perché si segue il Partial Order.

### 3.8.1 Esempio su Enjoy Sport

1.  $h_0 = \langle 0, 0, 0, 0, 0, 0 \rangle$
2.  $x_1 = \langle \text{Sunny}, \text{Warm}, \text{Normal}, \text{Strong}, \text{Warm}, \text{Same} \rangle$  questo esempio è positivo<sup>2</sup> e  $h_0$  non è soddisfatta perché False. Quindi devo prendere i vincoli più generali da  $x_1$  e metterli in  $h$ , in questo caso sono le singole istanze poiché tutte più generali di 0.
3.  $h_1 = \langle \text{Sunny}, \text{Warm}, \text{Normal}, \text{Strong}, \text{Warm}, \text{Same} \rangle$
4.  $x_2 = \langle \text{Sunny}, \text{Warm}, \text{High}, \text{Strong}, \text{Warm}, \text{Same} \rangle$  esempio positivo, notiamo che High è in contrasto con Normal. Quindi cambio quell'attributo con uno più generale, in questo caso è "?".
5.  $h_2 = \langle \text{Sunny}, \text{Warm}, ?, \text{Strong}, \text{Warm}, \text{Same} \rangle$
6.  $x_3 = \langle \text{Rainy}, \text{Cold}, \text{High}, \text{Strong}, \text{Warm}, \text{Change} \rangle$  esempio è negativo quindi non faccio nulla
7.  $h_3 = \langle \text{Sunny}, \text{Warm}, ?, \text{Strong}, \text{Warm}, \text{Same} \rangle$
8.  $x_4 = \langle \text{Sunny}, \text{Warm}, \text{High}, \text{Strong}, \text{Cool}, \text{Change} \rangle$  esempio positivo
9.  $h_3 = \langle \text{Sunny}, \text{Warm}, ?, \text{Strong}, ?, ? \rangle$

### 3.8.2 Proprietà di Find-S

Lo spazio delle ipotesi è rappresentato come congiunzione di attributi (molto limitativo), l'algoritmo darà in output l'ipotesi più specifica, nello spazio H, che è consistente con gli esempi positivi del training set. L'ipotesi di output h sarà anche consistente con gli esempi negativi, a condizione che il concetto target sia contenuto in H.

### 3.8.3 Aspetti negativi di Find-S

Non so se il sistema di apprendimento converga con il concetto target, nel senso che non è in grado di determinare se ha trovato l'unica ipotesi coerente con gli esempi di allenamento. Inoltre non so quando i dati di allenamento sono incoerenti, in quanto ignora gli esempi di allenamento negativi → nessuna tolleranza al rumore sui dati!

## 3.9 Version Spaces

Find-S fornisce una ipotesi da H che è coerente con gli esempi di allenamento, questa è solo una delle tante ipotesi da H che potrebbero adattarsi ugualmente bene ai dati di allenamento. L'idea è quindi quella di dare in output l'insieme di tutte le possibili h consistenti con D.

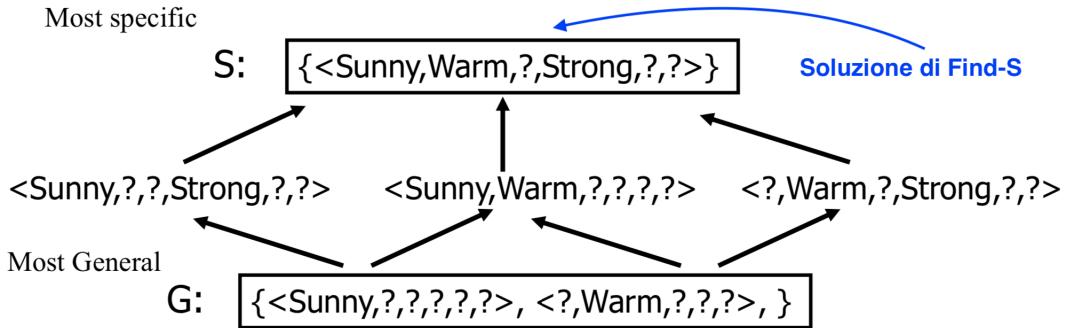
Il Version Space  $VS_{H,D}$  rispetto allo spazio delle ipotesi H e il training set D, è il sottinsieme delle ipotesi prese da h coerenti con tutti gli esempi di training:  $VS_{H,D} = h \in H | Consistent(h, D)$  dove  $Consistent(h, D) = \forall \langle x, c(x) \rangle \in D | h(x) = c(x)$

Si dice che un esempio x soddisfa l'ipotesi h quando  $h(x) = 1$ , indipendentemente dal fatto che x sia un esempio positivo o negativo del concetto target. Tuttavia, se un tale esempio sia coerente con h dipende dal concetto target e, in particolare, se  $h(x) = c(x)$ .

---

<sup>2</sup>l'esempio è preso dalla tabella a pagina 9

### 3.9.1 Esempio di Version Space



$$x_1 = <\text{Sunny Warm Normal Strong Warm Same}> +$$

$$x_2 = <\text{Sunny Warm High Strong Warm Same}> +$$

$$x_3 = <\text{Rainy Cold High Strong Warm Change}> -$$

$$x_4 = <\text{Sunny Warm High Strong Cool Change}> +$$

Il Version Space dell'algoritmo viene rappresentato solo dal suo membro più generale G e dal più specifico S (che è la soluzione di Find-S!).

Teorema: ogni membro del Version Space si trova tra:  $VS_{H,D} = \{h \in H | (\exists s \in S)(\exists g \in G)(g \geq h \geq s)\}$

dove  $x \geq y$  significa  $x$  è più generale o uguale a  $y$ .

### 3.10 Algoritmo List-Then-Eliminate

Il Version Space si riduce quindi man mano che si osservano più esempi, fino a quando idealmente rimane solo un'ipotesi coerente con tutti gli esempi osservati, presumibilmente, questo è il concetto target desiderato. Se non sono disponibili dati sufficienti per restringere lo spazio l'algoritmo può generare l'intero insieme di ipotesi coerenti con i dati osservati. L'algoritmo List-Then-Eliminate può essere applicato ogni volta che lo spazio di ipotesi  $H$  è finito. Ha molti vantaggi, tra cui il fatto che è garantito il risultato di tutte le ipotesi coerenti con i dati di allenamento. Sfortunatamente, richiede un elenco esaustivo di tutte le ipotesi in  $H$ , un requisito non realistico per tutti gli spazi di ipotesi tranne quelli più banali.

1. Iniziamo impostando il nostro Version Space con una lista contenente ogni ipotesi in  $H$
2. Per ogni esempio di allenamento  $< x, c(x) >$  rimuoviamo dal Version Space ogni ipotesi che è inconsistente con gli esempi di allenamento cioè  $h(x) \neq c(x)$
3. Viene dato in output la lista delle ipotesi ora contenute nel Version Space

### 3.11 Algoritmo Candidate Elimination

L'algoritmo di Candidate Elimination calcola il Version Space contenente tutte le ipotesi di  $H$  che sono coerenti con una sequenza osservata di esempi di addestramento. L'algoritmo di eliminazione dei candidati calcola la descrizione di questo set senza elencare esplicitamente tutti i suoi membri. Ciò si ottiene utilizzando l'ordinamento più generale che parziale e mantenendo una rappresentazione compatta dell'insieme di ipotesi coerenti.

Inizia inizializzando il Version Space sull'insieme di tutte le ipotesi in  $H$ , cioè inizializzando il set di limiti  $G$  per contenere l'ipotesi più generale in  $H$   $G_0 = \langle ?, ?, ?, ?, ?, ? \rangle$  e inizializzando il set di limiti  $S$  per contenere l'ipotesi più specifica  $S_0 = \langle 0, 0, 0, 0, 0, 0 \rangle$ . Questi due insiemi di limiti delimitano l'intero spazio delle ipotesi, poiché ogni altra ipotesi in  $H$  è sia più generale di  $S_0$  sia più specifica di  $G_0$ . Dopo che tutti gli esempi sono stati elaborati, lo spazio versione calcolato contiene tutte le ipotesi coerenti con questi esempi.

1. Inizializza  $G$  con l'insieme di ipotesi massimamente generali in  $H$   $G_0 = \langle ?, ?, ?, ?, ?, ? \rangle$
2. Inizializza  $S$  con l'insieme di ipotesi massimamente specifiche in  $H$   $S_0 = \langle 0, 0, 0, 0, 0, 0 \rangle$
3. Per ogni esempio di training  $d = \langle x, c(x) \rangle$ :
  - se  $d$  è positivo:
    - Rimuovere da  $G$  ogni ipotesi incompatibile con  $d$
    - per ogni ipotesi  $s$  in  $S$  che non è coerente con  $d$ : (Generalizziamo  $S$ )
      - \* Rimuovere  $s$  da  $S$
      - \* Aggiungi a  $S$  tutte le minime generalizzazioni  $h$  di  $s$  tali che  $h$  è consistente con  $d$  e alcuni membri di  $G$  sono più generali di  $h$
      - \* Rimuovi da  $S$  ogni ipotesi più generale di un'altra ipotesi in  $S$
  - se  $d$  è negativo:
    - Rimuovere da  $S$  ogni ipotesi incompatibile con  $d$
    - per ogni ipotesi  $g$  in  $G$  che non è coerente con  $d$ : (Specializziamo  $G$ )
      - \* Rimuovere  $g$  da  $G$
      - \* Aggiungi a  $G$  tutte le minime specializzazioni  $h$  di  $g$  tali che  $h$  è consistente con  $d$  e alcuni membri di  $S$  sono più specifici di  $h$
      - \* Rimuovi da  $G$  ogni ipotesi meno generale di un'altra ipotesi in  $G$

### 3.11.1 Esempio di Candidate Elimination

S:  $\{\langle \emptyset, \emptyset, \emptyset, \emptyset, \emptyset, \emptyset \rangle\}$

G:  $\{\langle ?, ?, ?, ?, ?, ? \rangle\}$

$x_1 = \langle \text{Sunny Warm Normal Strong Warm Same} \rangle +$

S:  $\{\langle \text{Sunny Warm Normal Strong Warm Same} \rangle\}$

G:  $\{\langle ?, ?, ?, ?, ?, ? \rangle\}$

$x_2 = \langle \text{Sunny Warm High Strong Warm Same} \rangle +$

S:  $\{\langle \text{Sunny Warm ? Strong Warm Same} \rangle\}$

G:  $\{\langle ?, ?, ?, ?, ?, ? \rangle\}$

$x_3 = \langle \text{Rainy Cold High Strong Warm Change} \rangle -$

esempio di  
allenamento  
negativo, ora  
specializzo G  
lascio il più  
generale dove  
non collide

S:  $\{\langle \text{Sunny Warm ? Strong Warm Same} \rangle\}$

G:  $\{\langle \text{Sunny, ?, ?, ?, ?, ?} \rangle, \langle ?, \text{Warm, ?, ?, ?, ?} \rangle, \langle ?, ?, ?, ?, ?, \text{Same} \rangle\}$

$x_4 = \langle \text{Sunny Warm High Strong Cool Change} \rangle +$

S:  $\{\langle \text{Sunny Warm ? Strong ? ?} \rangle\}$

G:  $\{\langle \text{Sunny, ?, ?, ?, ?, ?} \rangle, \langle ?, \text{Warm, ?, ?, ?, ?} \rangle\}$

G al passaggio 3 contiene le specializzazioni minime che mi permettono di coprire  $x_3$  come negativa, senza rendere G più specializzato di S. S e G sono le "guardie" di tutti i positivi e di tutti i negativi trovati finora. Il fatto che se specializzo troppo G rendo false alcuni input già accettati, deriva dal partial order: intelligentemente il controllo di consistenza di tutti i vecchi input non è effettuato dall'algoritmo. Infatti, il limite S del Version Space costituisce un riepilogo degli esempi positivi riscontrati in precedenza che possono essere utilizzati per determinare se una ipotesi è coerente con questi esempi. Il limite G riassume le informazioni da esempi negativi riscontrati in precedenza. Ogni ipotesi più specifica di G è garantita per essere coerente con esempi negativi del passato.

## 4 Inductive Bias

Il nostro spazio delle ipotesi non è in grado di rappresentare target disgiuntivi come ad esempio  $(\text{Sky} = \text{Sunny}) \vee (\text{Sky} = \text{Cloudy})$ .

Se avessimo due esempi del tipo:

$x_1 = < Sunny, Warm, Normal, Strong, Cool, Change >$  (positivo)  
 $x_2 = < Cloudy, Warm, Normal, Strong, Cool, Change >$  (positivo)  
troveremo che  $S = \{ < ?, Warm, Normal, Strong, Cool, Change > \}$   
che è inconsistente con l'esempio  $x_3 = < Rainy, Warm, Normal, Strong, Cool, Change >$  (negativo)  
trovando  $S = \{ \}$ , cioè S collassa, perché la funzione target è fatta solo di and.  
PROBLEMA: abbiamo vincolato il sistema di apprendimento considerando solo ipotesi congiuntive.  
Abbiamo bisogno di uno spazio di ipotesi più espressivo. Supponiamo che lo spazio delle ipotesi  $H$  contenga il concetto target  $c$ . In altre parole, che  $c$  può essere descritta da una congiunzione di letterali.

## 4.1 Sistema di apprendimento Unbiased

La ovvia soluzione al problema di assicurare che il concetto target sia nello spazio di ipotesi  $H$  è di fornire uno spazio di ipotesi in grado di rappresentare ogni concetto insegnabile.  
Significa che  $H$  è l'insieme di tutti i possibili sottoinsiemi di  $X$ : l'insieme di potenza  $P(X)$ . In EnjoySport  $|H| = 96$ ,  $|P(X)| = 2^96$  concetti distinti. Un'ipotesi può essere rappresentata con disgiunzioni, congiunzioni e negazioni delle nostre precedenti ipotesi.  $H$  sicuramente contiene il concetto target. Con questa modifica quali sono i nostri  $G$  e  $S$ ?  
NUOVO PROBLEMA: il nostro algoritmo di apprendimento dei concetti non è ora in grado di generalizzare oltre gli esempi osservati. Assumiamo tre esempi positivi  $(x_1, x_2, x_3)$  e due esempi negativi  $(x_4, x_5)$ .  $S = \{(x_1 \vee x_2 \vee x_3)\}$  e  $G = \{\neg(x_4 \vee x_5)\} \rightarrow$  Non abbiamo generalizzazione! Gli unici esempi classificati in modo non ambiguo sono gli stessi esempi di addestramento.

Proprietà: un sistema di apprendimento imparziale non è in grado di generalizzare

Prova: ogni istanza non osservata verrà classificata positivamente precisamente da metà delle ipotesi in Version Space e negativa dell'altra metà. Infatti  $\forall h$  consistente con  $x_i(test)$ ,  $\exists h'$  identico ad  $h$  con eccezione che  $h'(x_i) \neq h(x_i)$  cioè  $h \in VS \rightarrow h' \in VS$  (sono identici in  $D$ )

### 4.1.1 Il learning Unbiased è utile?

Un sistema di apprendimento che non fa assunzioni preliminari riguardo all'identità del concetto target non ha basi razionali per classificare eventuali istanze non conosciute. Uno sistema di apprendimento dovrebbe essere in grado di generalizzare i dati di allenamento usando presupposti precedenti al fine di classificare istanze invisibili. La generalizzazione è nota come salto induttivo e le nostre assunzioni precedenti sono il vincolo induttivo del sistema di apprendimento. Il bias non è solo assunto per efficienza, ma è necessario per la generalizzazione, tuttavia non ci dice quale sia la migliore soluzione per la generalizzazione.

## 4.2 Definizione Formale di Inductive Bias

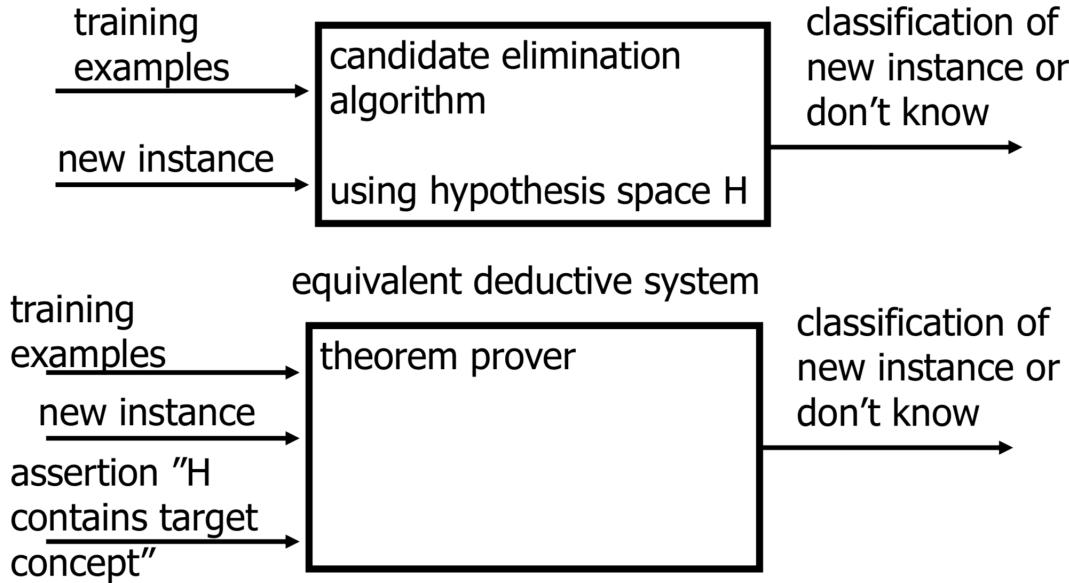
Consideriamo:

- Un algoritmo di apprendimento concettuale  $L$
- Un insieme di istanze  $X$  e un concetto target  $c$
- Sia  $D_c = \{ < x, c(x) > \}$  un insieme di esempi di training di  $c$
- Sia  $L(x_i, D_c)$  la classificazione assegnata all'istanza  $x_i$  da  $L$  dopo l'allenamento su  $D_c$ .

L'Inductive Bias di  $L$  è un insieme minimo di asserzioni  $B$  tale che per qualsiasi concetto target  $c$  e corrispondenti dati di allenamento  $D_c$ :

$(\forall x_i \in X)[B \wedge D_c \wedge x_i] \vdash L(x_i, D_c)$  dove  $A \vdash B$  significa che  $B$  è deducibile da  $A$ .

### 4.3 Sistemi Induttivi e Deduttivi equivalenti



Il Bias è l'insieme delle asserzioni che ci permette di trasformare il problema da induttivo a deduttivo.

### 4.4 Tre (algoritmi) sistemi di apprendimento con Bias differenti

1. Rote Learner: (lookup table) L'apprendimento corrisponde semplicemente alla memorizzazione di ogni esempio di allenamento osservato nella memoria. Le istanze successive vengono classificate osservandole nella memoria. Se l'istanza viene trovata in memoria, viene restituita la classificazione memorizzata. In caso contrario, il sistema rifiuta di classificare la nuova istanza. In poche parole salva gli esempi, classifica x se e solo se si "accoppiano" con uno degli esempi visti precedentemente.  
Non c'è Bias Induttivo → nessuna generalizzazione
2. Candidate Elimination: Le nuove istanze vengono classificate solo nel caso in cui tutti i membri del Version Space corrente concordino la classificazione. In caso contrario, il sistema rifiuta di classificare la nuova istanza.  
Bias Induttivo → il concetto target può essere rappresentato nel suo spazio di ipotesi.
3. Find-S: Questo algoritmo, descritto in precedenza, trova l'ipotesi più specifica coerente con gli esempi di addestramento. Quindi utilizza questa ipotesi per classificare tutte le istanze successive.  
Bias Induttivo → il concetto target può essere rappresentato nel suo spazio delle ipotesi e tutte le istanze sono istanze negative a meno che l'opposto non sia implicato da altre sue conoscenze.

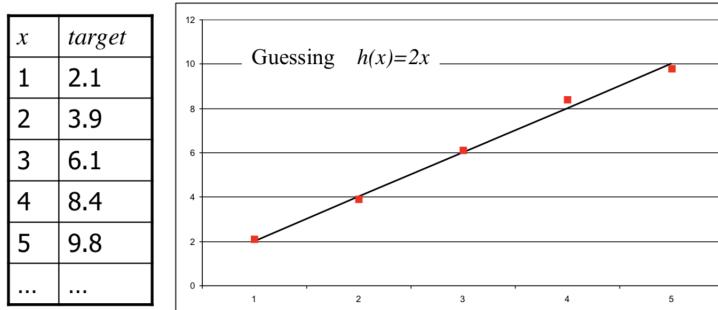
## 5 Modelli Lineari

Abbiamo visto che lo spazio delle Ipotesi  $H$  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  $H$  (selezione di  $h$  attraverso i dati di apprendimento) o che almeno una ipotesi  $h$  in  $H$  sia simile a  $f$  (approssimazione). Abbiamo compreso che un Algoritmo di Ricerca nello Spazio delle Ipotesi è rappresentato in ML tramite un algoritmo di apprendimento (ad esempio adattamento dei parametri liberi del modello al task). NOTA:  $H$  non può coincidere con l'insieme di tutte le funzioni possibili ma la ricerca deve essere esaustiva → Bias Induttivo.

Sia la regressione che la classificazione (quest'ultima vista nel Concept Learning) appartengono alla categoria delle tecniche di Supervised Learning, ovvero un tipo di apprendimento che fornisce a priori una serie di dati e informazioni al sistema, prima che questo cominci ad apprendere. Andremo a vedere come i Modelli lineare possono essere usati sia per la regressione (Regressione lineare) sia per la classificazione. La regressione lineare si differenzia nettamente dalla classificazione, poiché la classificazione si limita a discriminare gli elementi in un determinato numero di classi, mentre nella regressione l'input è un dato e il sistema ci restituisce un output reale approssimando una funzione. La regressione è un processo statistico che cerca di stabilire una relazione tra due o più variabili. Fornendo a un modello di regressione un valore  $x$ , questo restituirà il corrispondente valore  $y$  generato dall'elaborazione di  $x$ .

### 5.1 Esempio di regressione

La regressione come abbiamo detto è un processo di stima di una funzione a valori reali sulla base di una serie finita di campioni "rumorosi" (conosciamo le coppie  $(x, f(x)+\text{random noise})$ ).



### 5.2 Modello di Regressione Lineare Semplice (Univariata)

Si parte con una variabile input  $x$  e una in output  $y$ , assumiamo il modello  $h_w(x)$  espresso come  $y = w_1x + w_0$  dove  $w_0$  e  $w_1$  sono parametri liberi a valori reali da apprendere. Usiamo la lettera  $w$  perché immaginiamo questi coefficienti come weights (pesi) poiché il valore  $y$  cambia in base a questi valori. Come si può intuire cerchiamo di adattarci ai dati con una linea retta.

Lavoriamo in uno spazio di ipotesi infinite (i valori  $w_i$  sono continui) ma abbiamo una buona soluzione grazie alla matematica classica. Gauss dimostrò che se i valori  $y$  hanno del "rumore" distribuito allora i valori  $w_1$  e  $w_2$  possono essere trovati minimizzando la somma dei quadrati degli errori (dopo vedremo meglio cosa significa).

Assumiamo che la variabile  $y$  sia (linearmente) correlata a un'altra variabile  $x$  o ad altre variabili per cui  $y = w_1x + w_0 + \text{noise}$ , dove i  $w_i$  sono i parametri liberi. Il "noise" è l'errore nella misurazione dei valori target con distribuzione normale. Cerchiamo di costruire un modello (trovare le variabili  $w_i$ ) per predire/stimare la  $y$  per dei valori  $x$  non visti precedentemente.

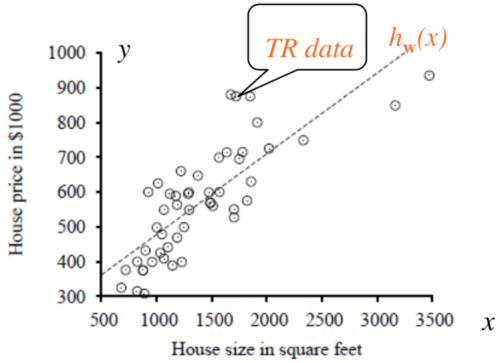


Figure 3: Esempio che associa il prezzo delle case in base alla metratura. (y costo della casa, x metratura)

Un esempio di regressione lineare per l'esempio sovrastante é la funzione  $h_w(x) = 0.232 * x + 246$  ma come arrivare a questa funzione?

### 5.3 Costruzione della funzione via LMS

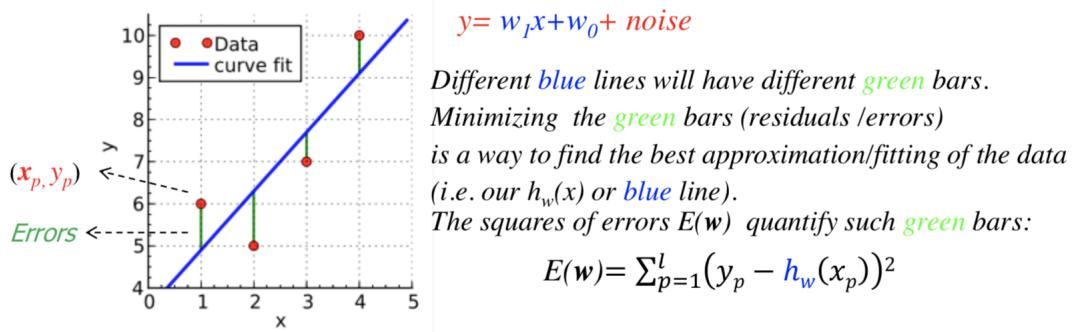
L'algoritmo LeastMeanSquare consiste nel trovare gli  $w$  per cui l'errore é minimizzato (miglior adattamento dei dati sul training set con "1" esempi).

- Dato un insieme di "1" esempi di allenamento  $(x_p, y_p)$
- Trovare  $h_w(x) = w_1x + w_0$  che minimizza l'errore medio sul Training Set
- Per calcolare la funzione di errore Loss usiamo la somma dei quadrati delle differenze tra il valore dato dall'esempio  $y_p$  e il valore calcolato dalla funzione  $h_w(x)$ . Il quadrato serve per avere solo valori positivi (si potrebbe anche usare il valore assoluto, ma è meglio di no, e il perché lo vedremo in seguito)

$$Loss(h_w) = E(w) = \sum_{p=1}^l (y_p - h_w(x_p))^2 = \sum_{p=1}^l (y_p - (w_1 * x_p + w_0))^2 \quad (1)$$

dove  $x_p$  é il p-esimo input e  $y_p$  é il p-esimo output dell'esempio p. Dividendo per l ottengo la media.

Quindi dobbiamo trovare l'argomento minimo  $w$  tale che l'errore è minimo in L2 (norma 2, le norme inducono una distanza):  $w = argmin_w Loss(h_w) = argmin_w E(w)$  in L2. A livello grafico lo rappresentiamo così:



Il metodo dei minimi quadrati è un approccio standard alla soluzione approssimata di sistemi sovr-determinati, ovvero insiemi di equazioni in cui vi sono più equazioni che incognite.

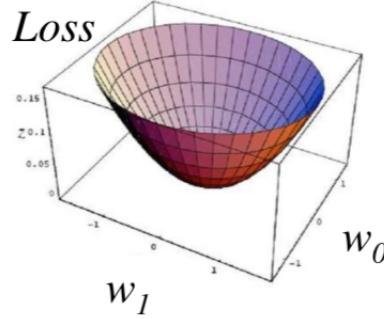
Il minimo locale è un punto stazionario dove il gradiente è nullo.

$$\frac{\partial E(w)}{\partial w_i} = 0 \quad \text{per } i = 0 \dots \text{dimensione\_spazio} \quad (2)$$

per la regressione lineare semplice abbiamo che la Loss è minimizzata quando le sue derivate parziali rispetto a  $w_0$  e  $w_1$  sono 0:

$$\frac{\partial E(x)}{\partial w_0} = 0 \quad \frac{\partial E(x)}{\partial w_1} = 0 \quad (3)$$

Se la funzione di perdita è convessa abbiamo la seguente soluzione diretta poiché non abbiamo nessun minimo locale.



per  $w_1$  abbiamo che

$$w_1 = \frac{l * (\sum x_p y_p) - (\sum x_p)(\sum y_p)}{l * (\sum x_p^2) - (\sum x_p)^2} = 0 \quad (4)$$

mentre per  $w_0$

$$w_0 = \frac{\sum y_p - w_1 * \sum x_p}{l} \quad (5)$$

Un approccio differente serve invece per quelle equazioni per cui il minimo Loss spesso non ha una soluzione definita: infatti useremo una sorta di Hill Climbing iterativo che segue il gradiente discendente. Sfruttiamo le seguenti regole delle derivate parziali:

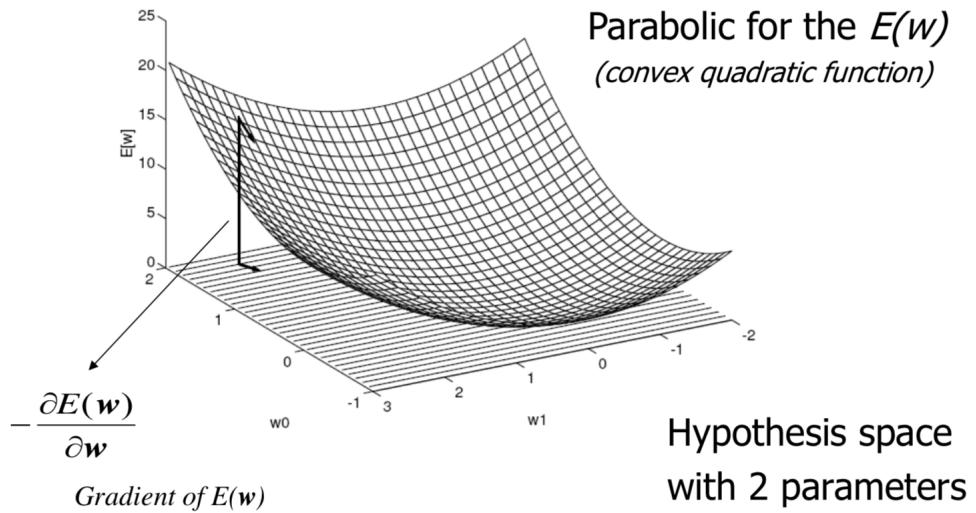
$$\frac{\partial}{\partial w} k = 0 \quad \frac{\partial}{\partial w} w = 1 \quad \frac{\partial}{\partial w^2} w = 2w \quad \frac{\partial(f(w))^2}{\partial w} = 2f(w) \frac{\partial f(w)}{\partial w} \quad (6)$$

ora cerchiamo il gradiente

$$\frac{\partial E(w)}{\partial w_i} = \frac{\partial}{\partial w_i} (y - h_w(x))^2 = 2(y - h_w(x)) \frac{\partial}{\partial w_i} (y - h_w(x)) = 2(y - h_w(x)) \frac{\partial}{\partial w_i} (y - (w_1 x + w_0)) \quad (7)$$

applicando  $w_0$  e  $w_1$  nella derivata ottengo due equazioni

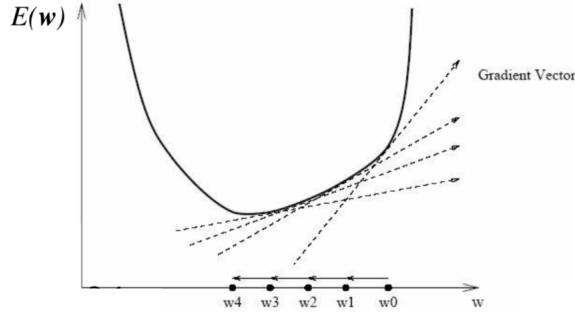
$$\frac{\partial E(w)}{\partial w_0} = -2(y - h_w(x)) \quad \text{e} \quad \frac{\partial E(w)}{\partial w_1} = -2(y - h_w(x)) * x \quad (8)$$



### 5.3.1 Gradiente discendente

La derivazione precedente suggerisce la costruzione di un algoritmo iterativo basato su  $\partial E(w)/\partial w_i$ . Il gradiente ci indica la direzione di salita, possiamo spostarci verso il minimo con discesa del gradiente  $\Delta w = -\text{gradiente}E(w)$ . La ricerca locale inizia con il vettore che contiene i valori dei pesi, viene modificato iterativamente per diminuire fino a minimizzare l'errore della funzione.

$w_{new} = w_{old} + \eta * \Delta w$  dove  $\eta$  é una costante che indicare il rateo di apprendimento



Le nostre regole di correzione dell'errore (chiamate delta-rule) cambiano i valori w proporzionalmente all'errore:

$$\Delta w_0 = -\frac{\partial E(w)}{\partial w_0} = 2(y - h_w(x)) \quad e \quad \Delta w_1 = -\frac{\partial E(w)}{\partial w_1} = 2(y - h_w(x)) * x \quad (9)$$

- se  $(y_{target} - output) = 0$  significa che non abbiamo errore e quindi nessuna correzione da fare
- se  $(output > y_{target})$  cioè  $(y - h) < 0$  vuol dire che l'output trovato é troppo alto quindi:
  - $\Delta w_0$  negativo → riduco  $w_0$
  - se  $(x_{input} > 0)$  e  $\Delta w_1$  negativo → riduco  $w_1$  altrimenti incremento  $w_1$
- se  $output < y_{target}$  cioè  $(y - h) > 0$  vuol dire che l'output trovato é troppo basso quindi:

- $\Delta w_0$  positivo  $\rightarrow$  aumento  $w_0$
- se  $(x_{input} > 0 \text{ e } \Delta w_1 \text{ positivo}) \rightarrow$  aumento  $w_1$  altrimenti diminuisco  $w_1$

Questo ci permette la ricerca in uno spazio delle ipotesi infinito, e può essere applicato facilmente per spazi H continui e con perdita differenziabile (? boh) altrimenti non posso calcolare il gradiente. Per questo motivo il valore assoluto è scomodo: potrebbe rendere non differenziabile la funzione.  
Arriviamo quindi a definire le funzioni su 1 esempi:

$$\Delta w_0 = -\frac{\partial E(w)}{\partial w_0} = 2 \sum_{p=0}^l (y_p - h_w(x_p)) \quad e \quad \Delta w_1 = -\frac{\partial E(w)}{\partial w_1} = 2 \sum_{p=0}^l (y_p - h_w(x_p)) * x_p \quad (10)$$

Ora può venirci in mente una domanda, aggiorniamo i  $w_i$  dopo aver ispezionato un insieme di esempi di allenamento 1, oppure dopo ogni singolo esempio analizzato?

- Batch Algorithm: calcoliamo la sommatoria dell'errore quadratico su l pattern diversi e otteniamo il gradiente, dopo l dati di training sommati abbiamo una "epoca" e aggiorniamo i  $w_i$ . Può essere molto lento...
- On-Line Algorithm: calcoliamo il gradiente su un pattern p e poi aggiorniamo subito  $w_i$ . Solitamente è più veloce del Batch...

## 5.4 Modello di Regressione Lineare Multivariato

Il caso standard è con due variabili, ma in realtà possiamo avere in input centinaia di variabili, l'esempio sul prezzo delle case in realtà conteneva anche le variabili che indicavano il numero delle stanze, l'età della casa ecc...

### 5.4.1 Notazione dei dati

Pattern	$x_1$	$x_2$	$x_j$	$x_n$
Pat 1	$x_{1,1}$	$x_{1,2}$		$x_{1,n}$
...				
Pat $p$	$x_{p,1}$	$x_{p,2}$	$x_{p,j}$	$x_{p,n}$
...				

X è una matrice l righe \* n colonne (numero di pattern \* numero di variabili), ogni riga della matrice  $X_i$  è un vettore che rappresenta un esempio.  $x_{i,j}$  è uno scalare cioè la componente j della riga i.  
Stiamo quindi dicendo che ogni esempio  $x_p$  è un vettore di n elementi.

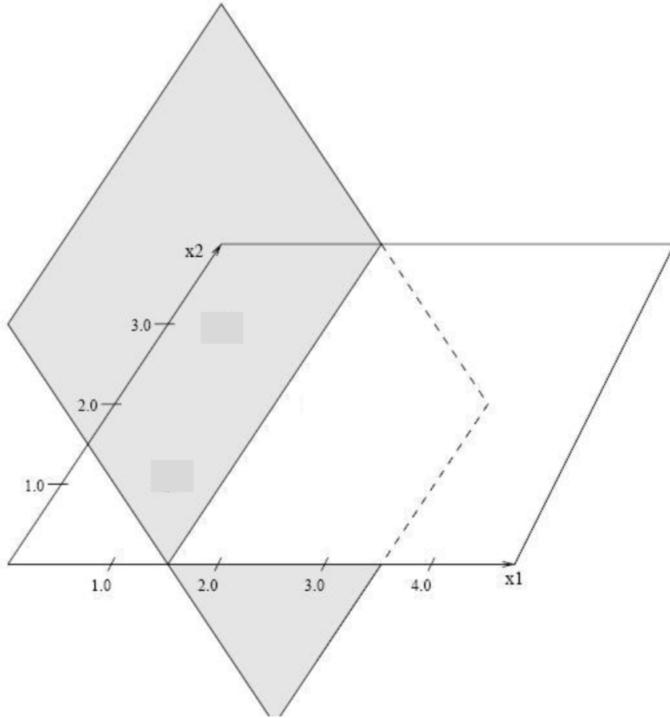
Supponendo vettori colonna x e w, con numero di esempi l e dimensione in input n, possiamo scrivere

$$w^T x + w_0 = w_0 + w_1 x_1 + w_2 x_2 + \dots + w_n x_n \quad (11)$$

dove  $w_0$  è il bias (non c'entra con il Concept Learning) è la distanza dalla componente 0. Possiamo anche scrivere che  $x_0 = 1$  in modo tale da poter scrivere che  $w^T x = x^T w$  quindi:  $x^T = [1, x_1, \dots, x_n]$  e  $w^T = [w_0, w_1, \dots, w_n]$

## 5.5 Hyperplane

Anziché adattare una retta, adattiamo un iperpiano.

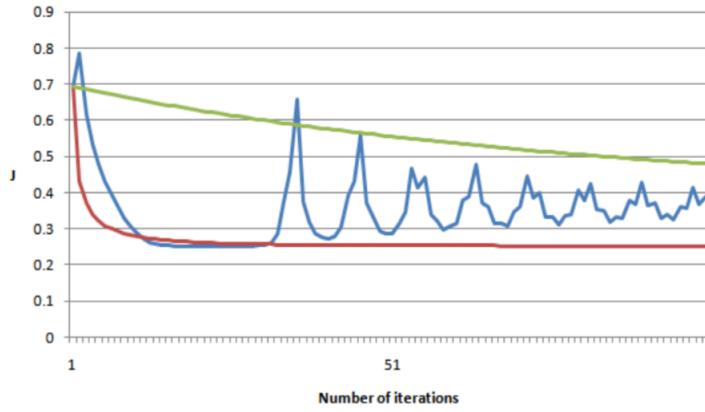


quindi abbiamo che  $w^T x = w_1 x_1 + w_2 x_2 + w_0$  e  $h(x_p) = x_p^T w = \sum_{i=0}^n x_{p,i} w_i$  dove  $x_{p,i}$  è la componente i-esima del p-esimo esempio.

## 5.6 Gradiente Discendente Algoritmo

1. si inizia con il vettore dei pesi  $w_{iniziale}$  e si fissa un  $0 < \eta < 1$
2. si calcola il gradiente  $\Delta w = -gradienteE(w) = -(\partial E(w)/\partial w)$
3. si calcola  $w_{new} = w_{old} + \eta * \Delta w$
4. ripetere dal passo 2 finché  $E(w)$  è sufficientemente piccolo

La versione batch dell'algoritmo si occupa di calcolare  $\Delta w$  dopo un insieme di esempi l. La versione online invece aggiorna  $\Delta w$  dopo ogni esempio p al posto di aspettare la sommatoria su l, ma ha bisogno di uno step size  $\eta$  più piccolo.



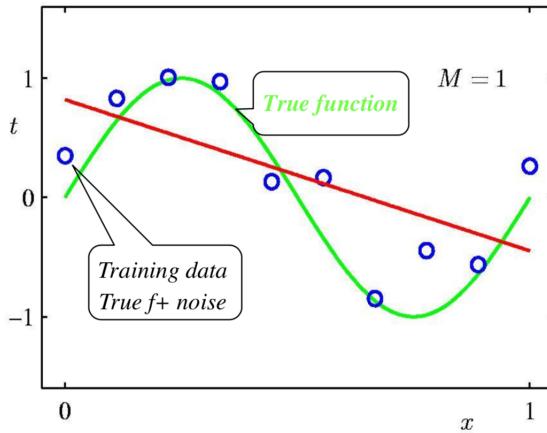
Questi sono tre esempi di curve, dove noi andremo a preferire quella che diminuisce nel minor tempo possibile (quella che impiega meno iterazioni).

## 5.7 Vantaggi dei modelli lineari

Se funziona bene è un modello "meraviglioso", molto semplice dove tutte le informazioni sui dati sono contenute nel vettore  $w$ . Facile da interpretare tant'è che viene usato in medicina, biologia, chimica, economia... Una delle cose fondamentali è che sono ammessi dati rumorosi e viene usato/incluso in modelli più complessi.

## 5.8 Limitazioni

La regressione lineare presenta delle limitazioni quando si parla di problemi non lineari.



Notiamo che  $h_w(X) = w_1x + w_0$  definita "lineare" non significa che è una linea retta, ma piuttosto come il modo in cui i coefficienti  $w$  occorrono nell'equazione di regressione (cioè lineare in  $w$ , non in  $x$ ). Quindi, possiamo anche trasformare gli input, come  $x, x^2, x^3, \dots$  con relazioni non lineari tra input e output. Sfruttando l'algoritmo LMS utilizzato fino ad ora:

$$y(x, w) = w_0 + w_1x + w_2x^2 + \dots + w_Mx^M = \sum_{j=0}^M w_jx^j \quad (12)$$

### 5.8.1 Esempio

Rispetto all'equazione scritta precedente in questo caso abbiamo  $M=2$ . La funzione non deve essere lineare nell'argomento (variabili in input) ma solo nei parametri che sono determinati per dare il miglior adattamento.

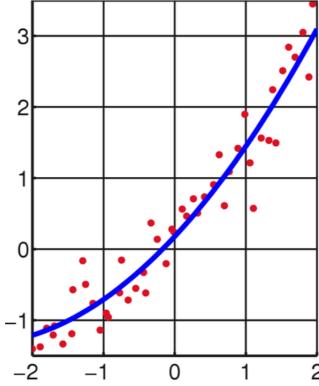


Figure 4: fitting su una funzione quadratica (linea blu) attraverso un set di dati (punti rossi)

## 5.9 Linear Basis Expansion

Prima calcolavamo il prodotto scalare tra  $w$  e  $x$  ( $w$  trasposto per  $x$ ) ora prendo delle trasformazioni (anche non lineari) del vettore  $x$  e le uso come un'espansione della base, combinandole in maniera lineare in accordo con la funzione  $\phi_k$ .

$$h_w(x) = \sum_{k=0}^K w_k \phi_k(x) \quad (13)$$

La funzione  $\phi$  può essere espressa come rappresentazioni polinomiali del tipo  $\phi(x) = x_j^2$  oppure  $\phi(x) = x_j * x_i$ , o ancora trasformazioni non lineari come  $\phi(x) = \log(x_j)$ . Ora il numero dei parametri è  $K > n$ , soprattutto il modello è lineare nei parametri quindi possiamo usare lo stesso algoritmo di prima.

### 5.9.1 Esempi

- dimensione di  $x = 1$  e  $\phi_j(x) = x^j$

$$h(x) = w_0 + w_1 x + w_2 x^2 + \dots + w_M x^M = \sum_{j=0}^M w_j x^j \quad (14)$$

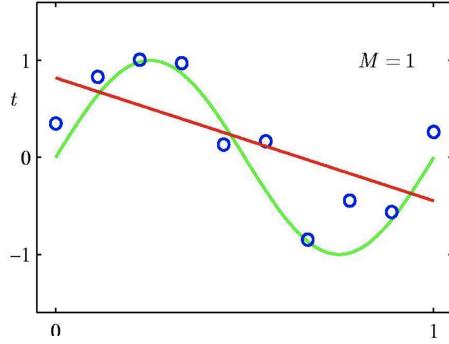
regressione polinomiale ad 1 dimensione ( $K=M$ )

- $\phi(x) = \phi([x_1, x_2, x_3])$

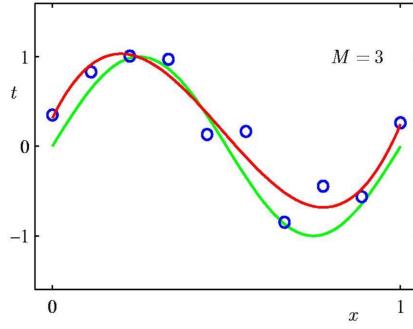
$$h(x) = w_1 x_1 + w_2 x_2 + w_3 \log(x_2) + w_4 \log(x_3) + w_5 (x_2 x_3) + w_0 \quad (15)$$

Quale  $\phi$  scegliere? Adottiamo un modello in stile dizionarioario? I punti a favore sono la maggiore espressività poiché può modellare relazioni più complesse di quelle lineari. I punti contro sono l'avere una grossa base di funzioni che richiedono metodi per controllare la complessità del modello, vediamo perché...

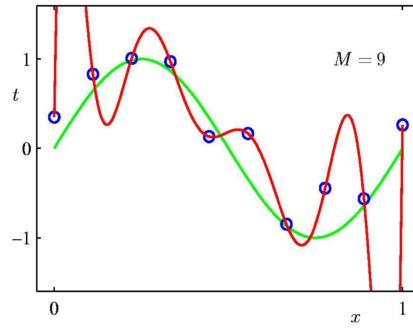
- grado del polinomio = 1, notiamo che è esegue troppo "underfitting".



- ordine del polinomio = 3, notiamo che abbiamo molta più flessibilità ma usciamo dal modello con variabili lineari.



- grado del polinomio = 9, notiamo che è flessibilissimo ma eccessivo. Cioé  $E(w)=0$  ma dobbiamo tenere conto degli errori sul test set, perché se i dati fossero rumorosi noi stiamo addatando al rumore.



### 5.9.2 Tradeoff per la complessità

- Se scegliamo un modello troppo semplice non fitta bene i dati e abbiamo una soluzione vincolata (ce ne accorgiamo perché  $E(w)$  è alto) → Underfitting
- Se scegliamo un modello troppo complesso siamo troppo sensibili alla piccole perturbazioni dei dati → Overfitting

Vogliamo scegliere la regolarizzazione per bilanciare bias e varianza e lo faremo attraverso il controllo della complessità del modello.

## 5.10 Regolarizzazione - Ridge Regression

Possiamo evitare l'overfitting penalizzando le funzioni complicate mantenendo comunque la flessibilità dello spazio delle ipotesi (un po' come il rasoio di ockham, si preferisce lo spazio delle ipotesi più semplice che fitta i dati).

Chiamata anche regolarizzazione di Tikhonov è sempre un modello LMS ma regolarizzato. Per esempio è possibile aggiungere vincoli alla sommatoria del valore di  $|w_j|$  favorendo modelli "sparsi", con meno termini dovuti a pesi  $w_j = 0$  (ovvero soluzioni meno complesse).

Un modello non regolarizzato fitta bene ma generalizza male (non ci serve a niente). Vorrei un modello semplice (magari con alcuni coefficienti che valgono zero). Come si fa? Aggiungiamo una penalizzazione alla complessità del modello della loss, sommando la norma del vettore  $w$ , moltiplicato per un parametro  $\lambda$  chiamato coefficiente di regolarizzazione. Con il parametro lambda (che è una costante), non ho più bisogno di modificare il grado dei polinomi, o analizzare funzioni più complesse.

$$\text{definito } E_D(w) + \lambda E_W(w) \text{ otteniamo } Loss(h_w) = \sum_p (y_p - h_w(x_p))^2 + \lambda \|w\|^2 \quad (16)$$

Ottenendo quindi la nuova regola di apprendimento caratterizzata dal weight decay (aggiunta di  $2\lambda w$ )

$$w_{new} = w_{old} + \eta * \Delta w - 2\lambda w_{old} \quad (17)$$

Per via di  $-2\lambda w_{old}$  decremo il parametro se positivo, lo incremento se negativo (in pratica lo avvicino a zero → weight decay). Quindi possiamo controllare la complessità del polinomio sfruttando solamente  $\lambda$ .

Finora abbiamo posto tutta la complessità nel bias di linguaggio, cioè un vincolo posto sul modello (lineare, quadratico, etc). Con la regolarizzazione sposto tutto sul bias di ricerca, cioè sull'algoritmo, come l'algoritmo riesce a trovare la soluzione. Noi non poniamo a priori una complessità al modello, ma sarà il mio metodo di ricerca intelligente a scegliere la funzione più semplice.

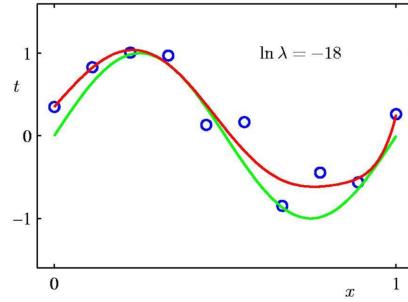


Figure 5: Polinomio di grado 9 con  $\log_e(\lambda) = -18$  quindi  $\lambda$  basso

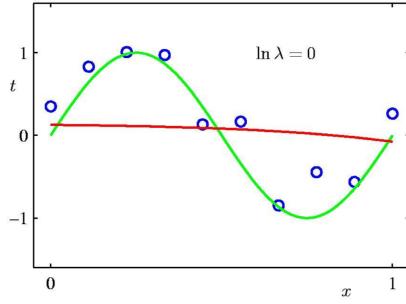


Figure 6: Polinomio di grado 9 con  $\log_e(\lambda) = 0$  quindi  $\lambda$  troppo alto

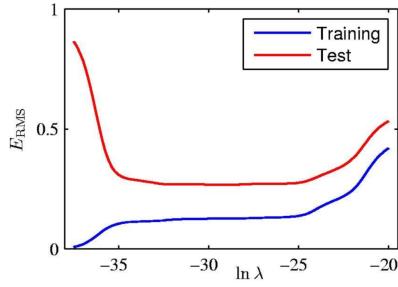


Figure 7:  $E_{RMS}$  vs  $\log_e(\lambda)$

Possiamo concludere quindi dicendo che se  $\lambda$  é basso  $\rightarrow$  overfitting, mentre se  $\lambda$  alto  $\rightarrow$  underfitting.

### 5.11 Limitazione delle funzioni a base fissa

Quando facciamo un'espansione polinomiale, la dimensionalità del problema cresce molto, e i dati che prima erano densi, potrebbero diventare sparsi. Quando la dimensione dell'input aumenta, il problema diventa enormemente difficile (la difficoltà cresce esponenzialmente, proprio come il volume con le dimensioni). Inoltre, se inizio a mettere dati a caso, rischio di aumentare il rumore. Avere la funzione base che opera su ogni dimensione di uno spazio implica che lo spazio di input richieda un numero combinatorio di funzioni. Per esempio un polinomio dell'ordine 3, utilizza tutte le combinazioni di variabili di input dovute ai prodotti  $x_1x_2, x_2x_3, \dots, x_1x_2x_3$  ecc...  $\rightarrow D^3$  (approssimare all'aumentare di D)

In altri modelli, vedremo come possiamo farcela con meno funzioni di base. Si scelgono le funzioni dal dizionario e si mettono dei parametri liberi, per adattare la phi ad H. Ho il vantaggio di essere più generale, ma il modello diventa non lineare rispetto a w (reti neurali).

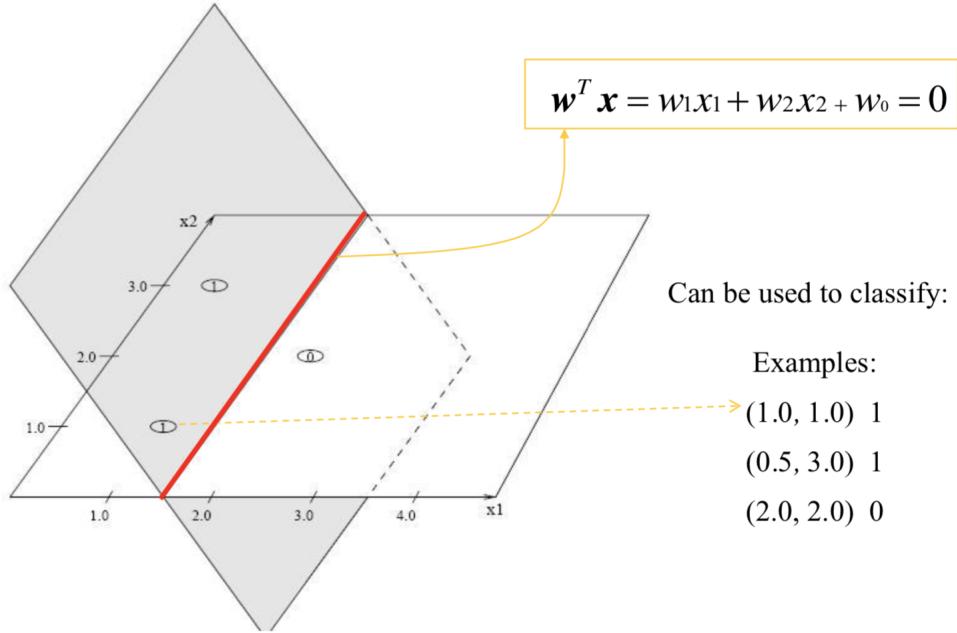
In altri modelli invece di calcolare esplicitamente la phi, viene fatta in maniera implicita, col concetto di funzioni kernel, controllando la complessità del modello direttamente con la funzione obiettivo, non più con una penalità aggiunta dopo (SVM).

### 5.12 Classificazione con il modello lineare

Possiamo utilizzare il modello lineare per la classificazione, in questo caso usiamo un iperpiano ( $w^T * x$ ) assumendo valori positivi o negativi. Sfruttiamo questi modelli per decidere se un punto x appartiene alla zona positiva o negativa dell'iperpiano. Quindi vogliamo impostare w vettore (imparando) in modo tale da ottenere una buona precisione di classificazione.

### 5.12.1 Visione geometrica dell'iperpiano

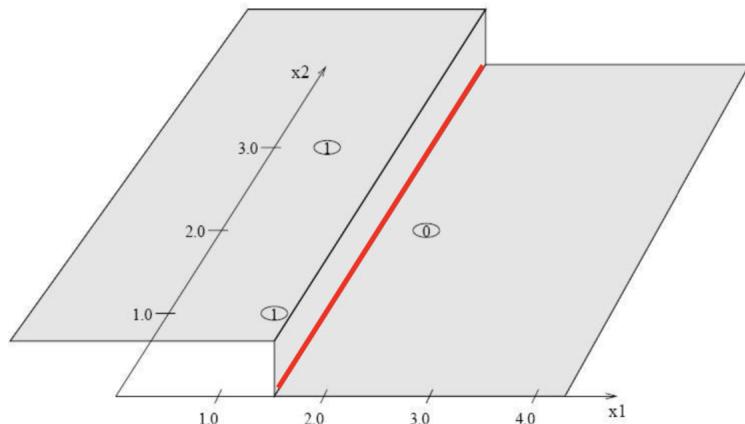
$w^T * x$  (prodotto di vettore riga per vettore colonna) definisce un iperpiano, il decision boundary è il luogo geometrico<sup>3</sup> dei punti dove l'iperpiano vale zero e può essere utilizzato per classificare.



Quindi possiamo definire una funzione di classificazione del tipo:

$$h(x) = \begin{cases} 1 & \text{if } w^T x + w_0 \geq 0. \\ 0 & \text{altrimenti.} \end{cases} \quad (18)$$

che possiamo scrivere come  $h(x) = \text{segno}(w^T x + w_0) \rightarrow h(x_p) = \text{sign}(x_p^T w) = \text{sign}(\sum_{i=0}^n x_{p,i} w_i)$



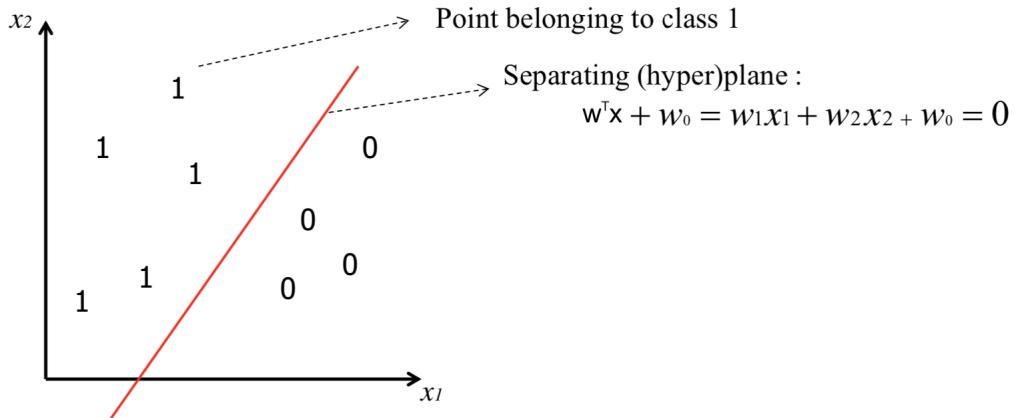

---

<sup>3</sup>è una linea, o un piano (quando ci troviamo in dimensioni più gradi)

### 5.12.2 Classificazione attraverso Decision Boundary lineare

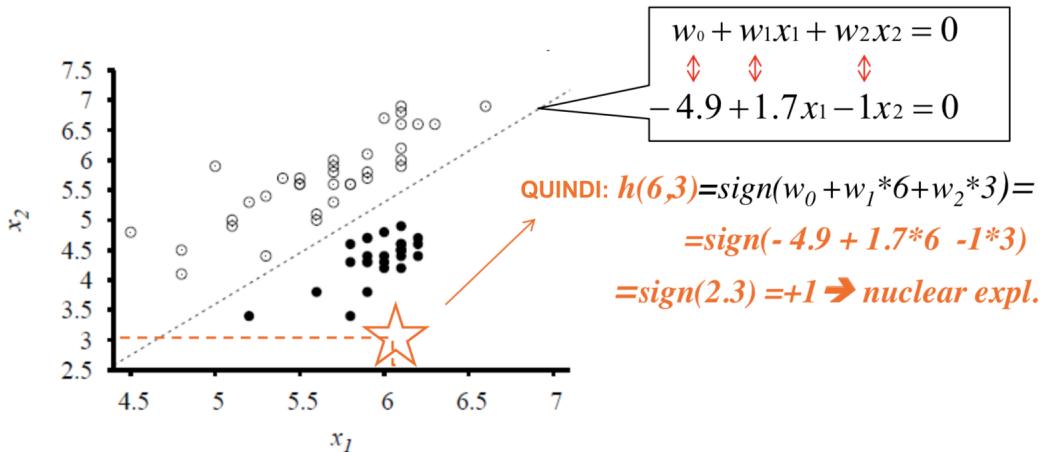
La classificazione può essere vista come l'allocazione dello spazio in input in diverse regioni di decisione. Ad esempio in uno spazio di due dimensioni  $x = (x_1, x_2)$  in  $R^2$  e con  $f(x) = 0 \vee 1$  il decision boundary sarà una retta.

Linear Threshold Unit (LTU) = indicatore di funzioni, quanti ne abbiamo? un insieme di dicotomie indotte da iperpiani, sono tutti i modi che abbiamo per muovere la linea rossa.



### 5.12.3 Esempio Terremoti/Esplosioni Nucleari

Trovare  $h$  in modo tale che il dato  $(x_1, x_2)$  ritorni  $(0 \vee -1)$  per i terremoti e  $(1)$  per esplosioni nucleari. I dati sismici sono rappresentati da  $x_1$  che è la magnitudine dell'onda del corpo e  $x_2$  che è la magnitudine dell'onda della superficie. L'algoritmo trova il decision boundary.



### 5.12.4 Esempio Email Spam

Trovare  $h(\text{mail})$  in modo tale che restituisca  $+1$  se è spam e  $-1$  se non è una mail spam.

Ad esempio  $\phi(\text{mail}) = \text{txtmail.contains("denaro gratis")}$ ,  $w$  vettore di pesi per le funzioni di input che aiutano alla previsione, cioè peso positivo per "denaro gratis", negativo altrimenti.  $w^T x$  è la

combinazione dei pesi.  $h_w(x)$  provvede a dare il limite per decidere se una mail é spam o non é spam.

$$h_w(x) = \text{segno}(\sum_k w_k \phi_k(x)) \quad (19)$$

### 5.13 Il problema di apprendimento (per classificatori lineari)

Assumendo che usiamo ancora il metodo dei minimi quadrati, dato un insieme grande "l" di esempi di allenamento, bisogna trovare  $w$  che minimizza la somma residua dei quadrati

$$E(w) = \sum_{p=0}^l (y_p - x_p^T w)^2 = \|y - Xw\|^2 \quad (20)$$

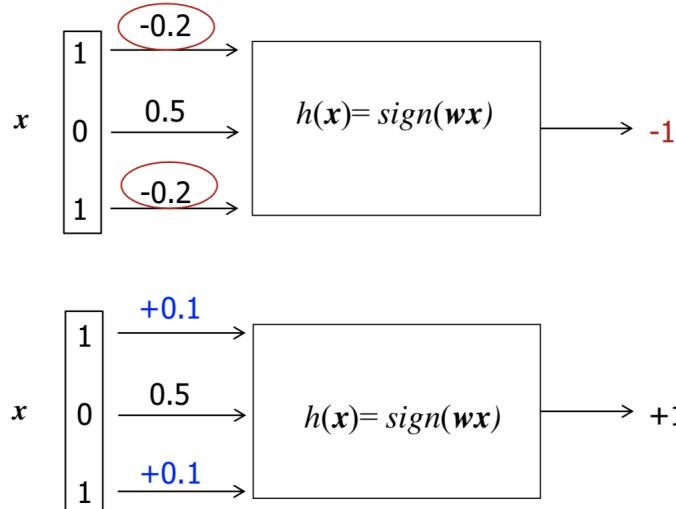
L'errore minimo: se  $y_p = 1$  allora  $x_p^T w \rightarrow 1$  mentre se  $y_p = 0 \vee -1$  allora  $x_p^T w \rightarrow 0 \vee -1$ .

Abbiamo ancora l'algoritmo di discesa del gradiente:

$$\Delta w_i = -\frac{\partial E(w)}{\partial w_i} = \sum_{p=1}^l (y_p - (x_p^T w)) * x_{p,i} \quad (21)$$

Nella derivata ignoriamo il 2 che viene fuori dal quadrato. È una costante che non incide sull'eta (il coefficiente di apprendimento).

#### 5.13.1 $\Delta w$ come regola della correzione dell'errore



Se un input viene classificato erroneamente, quindi il  $\Delta$  positivo per  $w_1$  e  $w_3$ , bisogna aumentare  $w_1$  e  $w_3$  proporzionalmente (con eta) al delta, quindi un valore positivo in questo caso. Correzione d'errore: alzo i valori dove sono troppo bassi, e viceversa.

### 5.14 Classificazione dei pattern

Il modello viene allenato su un insieme di esempi con il calcolo LMS su  $w^T x$  dall'algoritmo di discesa del gradiente, usato per la regressione lineare. Il modello viene usato per la classificazione applicando

la funzione di soglia  $h(x) = \text{segno}(w^T x)$ . L'errore può essere calcolato come errore di classificazione o numero di pattern classificati erroneamente.

$$L(h(x_p), d_p) = \begin{cases} 0 & \text{se } h(x_p) = d_p \\ 1 & \text{altrimenti.} \end{cases} \quad \text{mean\_err} = \frac{1}{l} \sum_{i=1}^l L(h(x_i), d_i) \quad (22)$$

dove  $d_p$  è l'output dato dagli esempi di allenamento, quindi stiamo classificando i valori calcolati dalla nostra  $h$  con i valori degli esempi etichettati.

$$\begin{aligned} \text{Accuratezza} &= \text{media dei pattern correttamente classificati} = (l - \text{numerr})/l \\ \text{dove } \text{numerr} &= \sum_{i=1}^l L(h(x_i), d_i) \end{aligned}$$

L'algoritmo è lo stesso come per la regressione → Algoritmo gradiente discendente (anche l'espansione lineare e Tikhonov possono essere applicati). Notare che l'algoritmo converge asintoticamente al Min-LeastSquare, non necessariamente al numero minimo di errori di classificazione.

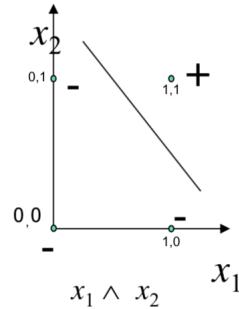
## 5.15 Congiunzioni nel caso di Modello Lineare

Possiamo rappresentare congiunzioni con i modelli lineari, ad esempio:

$x_1 \wedge x_2 \wedge x_4 \Leftrightarrow y$  cioè  $1x_1 + 1x_2 + 0x_3 + 1x_4 > 2$  dove  $w_1 = 1, w_2 = 1, w_3 = 0, w_4 = 1, w_0 = 2$  (vettore dei pesi)

Altro esempio con grafico:

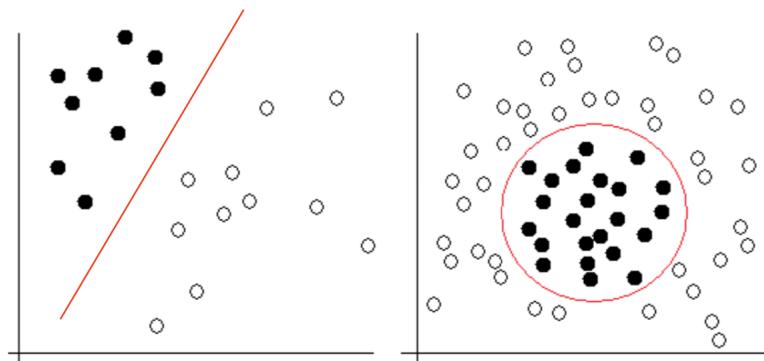
$x_1 \wedge x_2$  cioè  $1x_1 + 1x_2 > 1$



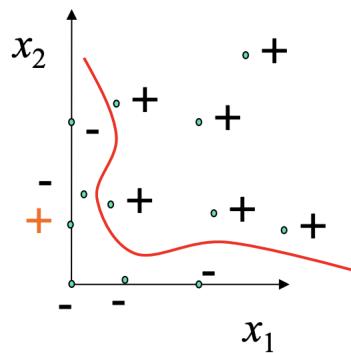
In generale  $w$  può essere appreso per trovare la soluzione.

### 5.15.1 Limitazioni

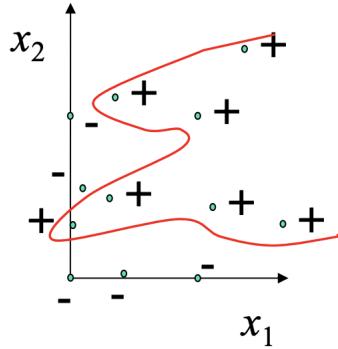
Nella geometria due insiemi di punti in un grafico a due dimensioni sono linearmente separabili quando possono esser separati da una singola linea.



In generale due gruppi di punti sono linearmente separabili in uno spazio di dimensione  $n$  se possono essere separati da un iperpiano di dimensione  $n - 1$ . Il confine lineare dà la soluzione esatta solo per insiemi di punti linearmente separabili.



*Non linear decision boundary  
(by a basis expansion)*  
*In this example 1 error is admitted  
(orange)*



*A not-smooth classifier  
that may need  
regularization*

## 5.16 Altri sistemi di apprendimento per la classificazione

- Linear Discriminant Analysis
- Logistic Regression: Stima la/le proprietà di  $x$  dato  $y$ ,  $P(y/x)$  partendo modellando la densità della classe come densità nota. Abbiamo una soglia soft (continua, differenziabile) con funzione logica (anziché soglia hard 0/1)
- Reti Neurali e SVM: modelli flessibili che includono approssimazione non lineare sia per la regressione che per la classificazione.

## 5.17 Conclusioni sui Modelli Lineari

I modelli lineari sono un approccio fondato di base per la regressione e la classificazione, è un modo molto compatto per rappresentare la conoscenza (tutti i dati sono nel vettore  $w$ ) ma con una forte assunzione sulla relazione tra i dati (Assumiamo che ci sia una relazione lineare tra i dati, o comunque in accordo alla  $\phi$  che abbiamo scelto. È una assunzione molto forte sullo spazio delle ipotesi). Abbiamo visto un algoritmo di correzione iterativa dell'errore (LMS) che ricerca spazi di ipotesi continui, una visione dei limiti ad approcci lineari e delle esigenze per modelli di ML più flessibili e anche i loro problemi: un'estensione del modello lineare per attività non lineari e un'introduzione al controllo della complessità (regolarizzazione).