

Interpretazione

Valore di verità per un enunciato

Soddisfacibilità

$A \in \text{soddisf.}$ se esiste un modello per A

Modello

Interpretazione che rende vera una formula

Tautologia o Valida

A è valida se è vera in tutte le interpretazioni

A valida $\Leftrightarrow \neg A$ iuoddisf.

Conseguenza logica

$$KB \models \alpha \Leftrightarrow M(KB) \subseteq M(\alpha)$$

Yun. dei Modelli

Algo DPLL (Alg. Soddisfacibilità)

Algo TT-Entails (Model checking)

1) Enumera tutti i modelli della KB

2) Vedo se ogni modello della KB è un modello anche per $\alpha \Rightarrow KB \models \alpha$

Algo Walk-SAT (Alg. di Ric. locale)

1) Inizializzo con interpretazione casuale

- 2)
- WALK-SAT ad ogni passo:
 - Scelgo a caso una clausola non ancora soddisfatta
 - Scelgo un simbolo da modificare (flip) scegliendo con probabilità p (di solito 0,5) tra una delle due:
 - Passo casuale: un simbolo a caso
 - Passo di ottimizzazione: sceglie quello che rende più clausole soddisfatte
 - Si arrende dopo un certo numero di flip predefinito (variabile max-flips)

- Se la formula è iuoddisf. e max-flips = ∞ , l'algo non termina.

- Se soddisf. e max-flip = ∞ trova sol. in tempo: bnevi

- Si usa quindi per trovare un modello in tempi bnevi

Algo DPLL (Alg. Soddisfacibilità)

1) trasformo in forma norm. congiuntiva

2) Vedo se $KB \wedge \neg \alpha = \{\}$

- Team. anticipata: Basta un simbolo T per rendere vera la clausola
- Simboli fuai: simbolo che appare sempre con lo stesso segno $\begin{array}{c} T \text{ ne} \\ F \text{ ne} \end{array}$
- Clausole unitarie: clausole con un solo letterale non anegnato $\begin{array}{c} T \text{ ne} \\ F \text{ ne} \end{array}$

Completo e termina sempre

1. Eliminazione della \Leftrightarrow : $(A \Leftrightarrow B) \equiv (A \Rightarrow B) \wedge (B \Rightarrow A)$

2. Eliminazione dell' \Rightarrow : $(A \Rightarrow B) \equiv (\neg A \vee B)$

3. Negazioni all'interno:

• $\neg(A \vee B) \equiv (\neg A \wedge \neg B)$ (de Morgan)

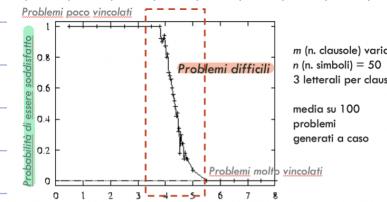
• $\neg(A \wedge B) \equiv (\neg A \vee \neg B)$

4. Distribuzione di \vee su \wedge : $(A \vee (B \wedge C)) \equiv (A \vee B) \wedge (A \vee C)$

Vincolatura di un problema

Viene indicato dal rapporto

num clausole
num simboli



Regola di Risoluzione (Metodo alt. per verificare $KB \models \alpha$)

1) La regola utilizzata è $\{P, Q\} \wedge \{\neg P, R\} = \{Q, R\}$

2) La regola di ris. è corretta ma non completa

• $KB \models A \Rightarrow KB \models A$ ma non è detto $KB \models A \Rightarrow KB \models A$

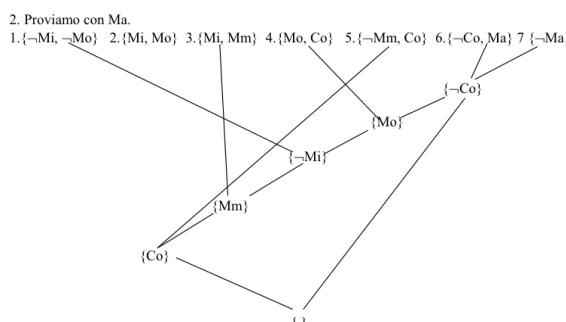
Si applica la ris. per refutazione che è completa

Esempio:

$$KB \wedge \{\neg \alpha\} \text{ iuoddisf.} \Leftrightarrow KB \models \alpha$$

↓
Otengo cose tipo

$$\{P\} \wedge \{\neg P\} = \{\}$$



Teo. Refutazione: $KB \models \alpha \Leftrightarrow KB \cup \{\neg \alpha\}$ è iuoddisf.

Teo. Risoluzione: $KB \wedge \{\neg \alpha\}$ iuoddisf. $\Leftrightarrow KB \wedge \{\neg \alpha\}$ è iuoddisf.

Logica del Primo Ordine (FOL)

$$\forall x (\forall y \text{Animale}(y) \Rightarrow \text{Ama}(x, y)) \Rightarrow (\exists y \text{Ama}(y, x))$$

1. Eliminazione delle implicazioni ($\Rightarrow, \Leftrightarrow$):

- $A \Rightarrow B$ diventa $\neg A \vee B$
- $A \Leftrightarrow B$ diventa $(\neg A \vee B) \wedge (\neg B \vee A)$

$$\begin{aligned} \forall x (\forall y \text{Animale}(y) \Rightarrow \text{Ama}(x, y)) &\Rightarrow (\exists y \text{Ama}(y, x)) \\ \forall x (\neg(\forall y \text{Animale}(y) \Rightarrow \text{Ama}(x, y))) \vee (\exists y \text{Ama}(y, x)) &\\ \forall x (\neg(\forall y \text{Animale}(y) \vee \neg \text{Ama}(x, y))) \vee (\exists y \text{Ama}(y, x)) & \end{aligned}$$

2. Negazioni all'interno:

- $\neg A$ diventa A
- $\neg(A \wedge B)$ diventa $\neg A \vee \neg B$
- $\neg(A \vee B)$ diventa $\neg A \wedge \neg B$
- $\neg \forall x A$ diventa $\exists x \neg A$
- $\neg \exists x A$ diventa $\forall x \neg A$

$$\begin{aligned} \forall x (\neg(\forall y \text{Animale}(y) \vee \neg \text{Ama}(x, y))) \vee (\exists y \text{Ama}(y, x)) &\\ \forall x (\neg(\forall y \text{Animale}(y) \vee \neg \text{Ama}(x, y))) \vee (\exists y \text{Ama}(y, x)) &\\ \forall x (\neg(\forall y \text{Animale}(y) \wedge \neg \text{Ama}(x, y))) \vee (\exists y \text{Ama}(y, x)) & \end{aligned}$$

3. Standardizzazione delle variabili: ogni quantificatore una variabile diversa

$$\forall x (\exists y \text{Animale}(y) \wedge \neg \text{Ama}(x, y)) \vee (\exists z \text{Ama}(z, x))$$

4. Skolemizzazione: eliminazione dei quantificatori esistenziali.

In questo caso essendo che i due quantificatori esistenziali sono nell'ambito di uno universale dobbiamo introdurre due funzioni di Skolem.

$$\forall x (\text{Animal}(F(x)) \wedge \neg \text{Ama}(x, F(x))) \vee (\text{Ama}(G(x), x))$$

5. Eliminazione quantificatori universali: possiamo portarli tutti davanti e poi eliminarli usando la convenzione che le variabili libere sono quantificate universalmente.

$$(\text{Animal}(F(x)) \wedge \neg \text{Ama}(x, F(x))) \vee (\text{Ama}(G(x), x))$$

6. Forma normale congiuntiva (congiunzione di disgiunzioni di letterali):

$$(\text{Animal}(F(x)) \vee (\text{Ama}(G(x), x))) \wedge (\neg \text{Ama}(x, F(x)) \vee (\text{Ama}(G(x), x)))$$

7. Notazione a clausole

$$\{\text{Animal}(F(x)), (\text{Ama}(G(x), x))\} \neg \{\text{Ama}(x, F(x)), (\text{Ama}(G(x), x))\}$$

8. Separazione delle variabili: clausole diverse, variabili diverse

$$\{\text{Animal}(F(x_1)), (\text{Ama}(G(x_1), x_1))\} \neg \{\text{Ama}(x_2, F(x_2)), (\text{Ama}(G(x_2), x_2))\}$$

Trovata la forma a clausole

Sostituzione

Data una sost. σ e un'espns. A, otengo una

nuova espressione $A\sigma$ con i val. di A sostituiti:

Esempio:
 $\text{Subst}(\{x/A, y/f(B), z/W\}, P(x, y, z)) = P(A, f(B), W)$
 $\text{Subst}(\{x/g(y), y/z, z/f(x)\}, Q(x, y, z)) = Q(g(y), z, f(x))$

le sost. vengono fatte simultaneamente
perché faccio sost. simultanee \Rightarrow Non sost. successivamente.

sulla σ del / posso comparsire solo variabili, a dx dei termini in generale.

Normalizzazione: Una var. su σ non può comparsire anche a dx, es: $\{x, f(x)\}$ $\{x/g(y), y/z\}$ \Rightarrow NO!

27

Normalizzazione: Una var. su σ non può comparsire anche a dx, es: $\{x, f(x)\}$ $\{x/g(y), y/z\}$ \Rightarrow NO!

Regola di Ris. FOL

Usa lo stesso metodo della ris. prop. ma usa le

unifications per rendere uguali ma inv. di segno

all'interno delle clausole

TASK: Supervised, Unsupervised o Reinforcement

Model: Definisce lo spazio delle ipotesi H

learning Algo: Ricerca nello spazio delle ipotesi

Generalizzazione: capacità di fare inferenza su esempi mai visti

Regole congiuntive (spazio ipotesi incompleto, licenza completa)

$h_k(x)$ è più generale di $h_j(x)$ se tutte le volte che $h_k(x) = 1$ lo è anche $h_j(x)$. Più spec: $h = <0,0,0,0,0>$

Più generale: $h = <?, ?, ?, ?, ?>$

Version Spaces: Sottoinsieme delle ipotesi prese da H coerenti con tutti gli esempi di training (coerenti con D)

VS_{H,D} cioè: $\forall x \in D . h(x) = c(x)$ con $h \in H$

IL VS viene definito da una funzione più generale G e una più specifica S, tutte le funzioni che stanno nel VS saranno meno generali di G e meno specifiche di S

Find-S Algo (Non usa VS)

1. Inizializza h con l'ipotesi più specifica nello spazio delle ipotesi H ($h_0 = <0,0,0,0,0>$)

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

PROBLEMA: trova una ipotesi unica, non ammette numeri

Bias:

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.

Algo di unification

Vogliamo trovare una sost. che rende identiche 2 espressioni.

Algo: Trova il Most General Unifier (MGU) (sost. min per unificare)

```
function UNIFY(x, y, θ) returns a substitution
θ, the substitution built up so far (optional, defaults to empty)
if θ = failure then return failure
else if x = y then return θ
else if VARIABLE?(x) then return UNIFY-VAR(x, y, θ)
else if VARIABLE?(y) then return UNIFY-VAR(y, x, θ)
else if COMPOUND?(x) and COMPOUND?(y) then
    return UNIFY(x.ARGS, y.ARGS, UNIFY(x.OP, y.OP, θ))
else if LIST?(x) and LIST?(y) then
    return UNIFY(x.REST, y.REST, UNIFY(x.FIRST, y.FIRST, θ))
else return failure

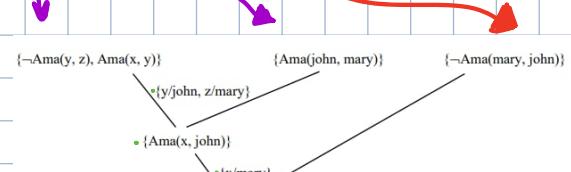
function UNIFY-VAR(var, x, θ) returns a substitution
if [var/x] ∈ θ then return UNIFY(var, x, θ)
else if [x/var] ∈ θ then return UNIFY(var, val, θ)
else if OCCUR-CHECK?(var, x) then return failure
else return EXTEND([var/x], θ)

Esempio 1
1. UNIFY(P(A, y, z), P(x, B, z), {})
2. UNIFY((A, y, z), (x, B, z), {})
3. UNIFY((A, y, z), (B, z), {})
4. UNIFY((y, z), (B, z), {})
5. UNIFY((y, z), (B, x), {})
6. UNIFY((x, z), (B, x), {})
7. UNIFY(z, x, {})
```

var ha già un valore
x ha già un valore
controllo di occorrenza

Esempio:

KB $\wedge \{\neg \alpha\}$ e cenco $\{ \}$



TRAINING SET

Un insieme di istante $\langle x, c(x) \rangle \leftarrow c(x) = \text{etichetta}$

1) L'ipotesi h nodalista $x \in h(x) = 1$

2) Un ipotesi è coerente con \uparrow insieme di tutte le istante possibili, non solo training set

→ Un esempio (istante), se: $\langle x, c(x) \rangle, x \in X \text{ t.c } h(x) = c(x)$

3) Training set D, se: $\forall x \in D . h(x) = c(x)$

cioè per ogni istante del training set prende il valore corretto.

Def. di h + generale

h_3 è più generale di $h_1 \Leftrightarrow \forall x \in X : h_1(x) = 1 \Rightarrow h_3(x) = 1$

Algoritmo test-then-eliminate (USA VS)

1. Iniziamo impostando il nostro Version Space con una lista contenente ogni ipotesi in H

2. Per ogni esempio di allenamento $\langle x, c(x) \rangle$ 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

PROBLEMA: Richiede elenco esaustivo delle ipotesi (cosa non realistica)

Candidate Elimination Algo

1. Inizializza G con l'insieme di ipotesi massimamente generali in H $G_0 = \{?, ?, ?, ?, ?, ?\}$
2. Inizializza S con l'insieme di ipotesi massimamente specifiche in H $S_0 = \{0, 0, 0, 0, 0, 0\}$
3. Per ogni esempio di training $d = \langle x, c(x) \rangle$:

Generalizzo G	<ul style="list-style-type: none"> • se d è positivo: <ul style="list-style-type: none"> - Rimuovere da G ogni ipotesi incompatibile con d - per ogni ipotesi s in S che non è coerente con d: (Generalizziamo S) <ul style="list-style-type: none"> • 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
Specializza G	<ul style="list-style-type: none"> • se d è negativo: <ul style="list-style-type: none"> - Rimuovere da S ogni ipotesi incompatibile con d - per ogni ipotesi g in G che non è coerente con d: (Specializziamo G) <ul style="list-style-type: none"> • 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

Quando \exists ha un VS \exists classifica in base alla maggioranza dei risultati delle ipotesi nel VS

Bias: \exists il concetto target è contenuto nello spazio di ipotesi H

Modelli lineari (Spazio ipotesi completo, ric. incompleta) \Rightarrow Bias:

Sono modelli lineari in w , parametri del modello, esempio: $h(x) = w_0 + w_1 x$

Vogliamo trovare w t.c. minimizza loss(h_w) = $\sum_{p=1}^l (y_p - h_w(x_p))^2$ = MSE

Si usa il gradiente il quale indica la direzione di crescita della funz., quindi lo prendiamo negativo

Singolo esempio

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

Batch Exampi

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

Aggi w $w_{new} = w_{old} + \eta * \Delta w$

Ridge Regression

Aggiungo una penalizzazione alla complessità del modello della loss.

Con il parametro λ gestisco quanto regolarizzazione, cioè gestisco complessità.

$$Loss(h_w) = \sum_p (y_p - h_w(x_p))^2 + \lambda \|w\|^2 \quad w_{new} = w_{old} + \eta * \Delta w - 2\lambda w_{old}$$

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

\downarrow riduce la complessità

λ alto = underfitting, λ basso = overfitting

Allenamento: Usa LMS e gradiente, come la regressione

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

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

$$w_{new} = w_{old} + \eta * \Delta w$$

Learning Biased e Unbiased

Un sistema per saper generalizzare ha bisogno di un bias (assunzioni preliminari) perché con un sistema unbiased \nexists ottiene che ogni nuova istanza verrà classificata positivamente precisamente dalla metà delle ipotesi nel VS e negativa dall'altra metà.

\exists bias \nexists chiama "inductive bias" ed è un insieme di assunzioni t.c.

$$\forall x_i \in X . \quad B \wedge D \wedge x_i \vdash L(x_i, D_c)$$

Bias

Candidate Elimination e Find-s : Spazio ipotesi limitato

• 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 è troppo alto quindi:

• Δw_0 negativo \rightarrow riduco w_0

• se $(x_{input} > 0)$ e Δw_1 negativo \rightarrow riduco w_1 altrimenti incremento w_1

• se $(output < y_{target})$ cioè $(y - h) > 0$ vuol dire che l'output trovato è troppo basso quindi:

• Δw_0 negativo \rightarrow aumento w_0

• se $(x_{input} < 0)$ e Δw_1 positivo \rightarrow aumento w_1 altrimenti diminuisco w_1

Linear basis Expansion

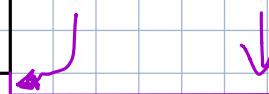
Prendo trainabili non lineari di x in modo da fittarli meglio gli esempi di training (se ho underfitting)

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

Prob: Overfitting dovuto alla complessità del modello

Classification con modelli lineari

Utilizzo una soglia, $L+U \Rightarrow w^T x + w_0 = 0$



emone su tot inferenze

Inferenza

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

loss inferenza

accuratezza:

ristante classificate correttamente

ristante totali (l)

$$h(x) = \begin{cases} 1 & \text{se } w \cdot x > T \\ 0 & \text{altrimenti} \end{cases}$$

h non è differenziabile, non posso utilizzarla nella $E(w)$!

Alberi di decisione (Bias = Ricerca Greedy senza backtracking)

Ogni attributo viene preso e viene espanso con i suoi valori in modo da generare l'albero di decisione (Algo ID3). L'ordine di esp. degli attributi viene deciso dal loro valore di Information Gain.

Terminologia:

- S è una collezione di esempi
- p_+ è una porzione di esempi positivi in $S \rightarrow p_+ = \frac{\text{numero di esempi positivi di } S}{\text{numero di esempi totali di } S}$
- p_- è una porzione di esempi negativi in $S \rightarrow p_- = \frac{\text{numero di esempi negativi di } S}{\text{numero di esempi totali di } S}$

$$\text{Entropy}(S) = -p_+ \log_2 p_+ - p_- \log_2 p_-$$

Valore tra 0 e 1, se gli esempi in S sono tutti positivi o tutti negativi l'entropia vale 0

PROBLEMA: Se ho attn. che ha un istanza sempre diversa (es: date) ottengo entropia 0 \rightarrow inf. gain più alto di tutti. In questi casi:

1. Calcolare il Gain per ogni attributo

2. Applicare il GainRatio solo a quegli attributi con il Gain superiore alla media (Ad esempio date ecc...)

$$\text{GainRatio}(S, Attr) = \frac{\text{Gain}(S, Attr)}{\text{SplitInformation}(S, Attr)}$$

$$\text{SplitInformation}(S, Attr) = - \sum_{i=1}^c \frac{|S_i|}{|S|} \log_2 \frac{|S_i|}{|S|}$$

$$\text{Gain}(S, Attr) = \text{Entropy}(S) - \sum_{v \in \text{Values}(Attr)} \frac{|S_v|}{|S|} \text{Entropy}(S_v)$$

↑
INF. GAIN di
UN ATTRIBUTO
↓
Cardinalità di tutti gli esempi
di esempio
Ogni valore dell'attributo

Si prende l'attn. con entropia più alta

SplitInformation misura l'entropia di S rispetto ai valori di A . Più i dati sono dispersi in modo uniforme, più è alto. GainRatio penalizza gli attributi che dividono gli esempi in tante piccole classi come per l'esempio dell'attributo della data. Sia $|S|$ — numerose divide gli esempi in n sottogruppi.

Inductive Bias Decision trees

Il Bias dei DT si basa sul fatto che la ricerca è incompleta poiché ID3 è un algo greedy

Bias di ricerca (ID3): Spazio completo delle ipotesi ma ricerca incompleta

Bias di linguaggio (CANDIDATE ELIMINATION): Spazio incompleto ma ricerca completa

Si preferisce il Bias di ricerca perché include sicuramente la funzione target.

Problema Overfitting: Gli alberi di decisione si adattano agli esempi rumorosi.

Def Overfitting: h overfittà D se $\exists h' \in H$. $\epsilon_D(h) < \epsilon_D(h') \wedge \epsilon_X(h') < \epsilon_X(h)$

Soluzione:

1. fermare la crescita dell'albero prima che abbia una classificazione perfetta
2. permettere all'albero di overfittare i dati e successivamente eseguire un post-potatura dei rami

6.9.2 Potatura con errore ridotto

Come possiamo usare il validation set per evitare l'overfitting? Nel reduced error pruning ogni nodo è un candidato per la potatura: la potatura consiste nella rimozione di un sottoalbero radicata in un nodo, quest'ultimo diventa una foglia e viene assegnata la classificazione più comune. I nodi vengono rimossi solo se l'albero risultante non ha prestazioni peggiori sul set di validazione. I nodi vengono eliminati in modo iterativo: ad ogni iterazione viene eliminato il nodo la cui rimozione aumenta la precisione del set di validazione. La potatura si interrompe quando nessuna potatura aumenta la precisione. Uno svantaggio è sacrificare dei dati del training set per formare il validation set.

Altri problemi:

1. Valori continui degli attributi

Fino ad ora abbiamo usato valori discreti per gli attributi, come possiamo affrontare quelli a valori continu?

Dato un attributo a valore continu A_i , si crea dinamicamente un nuovo attributo A_c tale che: $A_c = \text{True} \text{ se } A_i < c, \text{ False altrimenti.}$

Ma come determinare il valore di soglia c ? Un esempio sul tennis potrebbe essere:

Temperature	40	48	54	60	72	80	90
PlayTennis	No	No	Yes	Yes	Yes	Yes	No

2. Dati di Training Incompleti

• Si assegna il valore più comune tra tutti gli esempi di training sul nodo o quelli della stessa classe.

• Si assegna una probabilità a ciascun valore possibile da assegnare, in base alle frequenze, si associa il valore all'attributo mancante secondo la distribuzione di probabilità.

3. Attributi con costi differenti

In alcuni problemi le istanze degli attributi possono avere un costo associato, cioè possiamo dare più importanza ad alcuni attributi rispetto ad altri. Preferiamo gli alberi che usano costi bassi per gli attributi. L'algoritmo ID3 può essere modificato per lavorare anche con i costi:

• Tan e Schlimmer

$$\frac{\text{Gain}^2(S, \text{Attr})}{\text{Cost}(S, \text{Attr})} \quad (27)$$

• Nunes

$$\frac{2^{\text{Gain}(S, \text{Attr})}}{(\text{Cost}(\text{Attr}) + 1)^w} \text{ con } w=0/1 \quad (28)$$

Hold out Cross Validation

• separare TR, VL e TS (y_n iniziali disegnati)

Holdout
su VS

- per ogni valore differente dell'iper-parametro λ : cercare la migliore ipotesi che minimizza l'errore / perdita empirica (fittando i dati sul TR set) trovando i migliori parametri w . (Minimizziamo la funz. regolarizzata)
- selezionare la migliore ipotesi $h_{w,\lambda}$ dove migliore significa con il minor errore sul Validation Set
- opzionale: è possibile far fittare h su TR+VL con λ ormai fissato
- valutare la h finale sul Test Set (Accumuliamo su dati mai visti) \rightarrow si usato, "blind test set". Autentico allineamento una stima ottimistica

7.7 Statistical Learning Theory

Mettendo insieme:

- La capacità di generalizzazione (misurata come errore di test) di un modello rispetto all'errore di training, delineando le zone di overfitting o underfitting
- Il ruolo della complessità del modello
- Il ruolo del numero di dati

troviamo la Statistical Learning Theory (SLT):

- si approssima una funzione sconosciuta $f(x)$
- si cerca di minimizzare la funzione di rischio $R = \int Loss(t, h(x)) dP(x, t)$ (Integrale della Loss su tutti i dati).
- vengono dati un valore target t , una probabilità di distribuzione $P(x, t)$ e una funzione Loss del tipo $L(h(x), t) = (t - h(x))^2$
- il compito finale è quello di cercare una ipotesi h nello spazio delle ipotesi H , dove il valore di rischio generale R è minimo; ma noi abbiamo solo un data set finito $TR(x_i, t_i)$ con $i=1\dots l$
- per cercare dobbiamo quindi minimizzare l'errore empirico (training error) trovando il miglior valore per il modello con parametri liberi w .

$$R_{emp} = \frac{1}{l} \sum_{i=1}^l (t_i - h(x_i))^2$$

Questo si chiama principio induttivo della minimizzazione del rischio empirico (Empirical Risk Minimization). La domanda che dobbiamo porci è: possiamo usare R_{emp} per approssimare R ?

7.8 Teoria di Vapnik-Chervonenkis + SLT

- VC-dimension è una misura per la complessità dello spazio delle ipotesi H (flessibilità per fittare i dati, come ad esempio il numero di parametri per modelli lineari/grado del polinomio)
- VC-bounds afferma che con una probabilità di $1 - \delta$ che

$$R \leq R_{emp} + \epsilon \left(\frac{1}{l}, VC, \frac{1}{\delta} \right)$$

cioè che il rischio generale R ha un limite superiore definito dal rischio empirico (che decresce con la complessità dei modelli) più il VC-confidence. Il VC-confidence (ϵ) aumenta all'aumentare della complessità del modello (indicata da VC) e diminuisce quando il numero dei dati aumenta. ($1/\delta$ indica la probabilità che valga questo bound)

Possiamo usare il rischio empirico per approssimare R ? Sì, il Machine Learning è ben fondato, l'intuizione sta nel fatto che un numero alto di dati portano ad un R basso, mentre un alto VC-dim abbassa R_{emp} ma potrebbe alzare R (overfitting).

Complessità del modello: Diminuisce il R_{emp} , Aumenta VC-Confidence \Rightarrow potrebbe aumentare $R \Rightarrow$ Overfitting
(vc-dim)

Aumentare dei dati disp: Diminuisce VC-Confidence $\Rightarrow R$ basso \Rightarrow Sape Generalizzare

6.9.3 Regola della post-potatura

Nella pratica un metodo che funziona spesso è la post-pruning rule. Le seguenti regole sono solo euristiche: non garantiscono l'ottimo a priori.

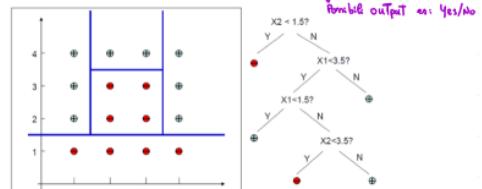
- Si crea l'albero di decisione a partire dal training set fino a che i dati di training fittano al loro meglio. (stiamo facendo overfitting appositamente)
- Si converte l'albero in un insieme equivalente di regole
 - Ogni path corrisponde a una regola
 - Ogni nodo lungo un path corrisponde a una pre-condizione
 - Ogni nodo foglia classifica
- per esempio $(Outlook = Sunny) \wedge (Humidity = High) \rightarrow (PlayTennis = No)$
- Potare (generalizzare) ogni regola rimuovendo quelle pre-condizioni la cui rimozione migliora l'accuratezza sia sul validation set, sia sul training set.
- Ordinare le regole in ordine di precisione stimato e considerarle in sequenza quando si classificano nuove istanze

Una Regola

Perché trasformare ogni path in regola? Ogni percorso distinto produce una regola diversa, la rimozione di una condizione può essere basata su un criterio locale. La potatura delle pre-condizioni è specifica delle regole mentre la potatura dei nodi è globale e influisce su tutte le regole!

6.13 Visione Geometrica (Decision Boundaries dei DT)

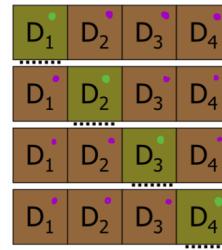
I decision boundaries, che possono essere prodotti da un albero di decisione, dividono lo spazio di input in rettangoli paralleli agli assi ed etichettano ogni rettangolo con una delle classi K (foglia dell'albero).



k-Fold cross Validation

Test set / Validation set

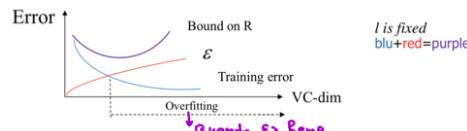
Training set



Scegliere un λ . Dividere il data set in D porzioni mutualmente esclusive di dati. Allenare l'algoritmo di apprendimento sull'insieme $D - D_i$ e testarlo su D_i . Calcolo la media sui risultati trovati su D_i . Cambio λ e rieseguo il tutto. Alla fine scelgo il modello con il minimo errore di validation, sulla base del valore calcolato con la media sui vari D_i .

Utilizza tutti i dati per training, validazione o test. NOTA: questa tecnica può essere utilizzata sia per il validation set, sia per il test set. In quale parte dobbiamo dividere? 3,5,10... ma spesso è computazionalmente costoso.

7.9 Structural Risk Minimization SRM



Sfruttiamo il concetto di controllo della complessità del modello, eseguiamo un trade-off tra la complessità del modello e l'accuratezza sul TR set.

7.10 Conclusioni

La STL permette di inquadrare formalmente il problema della generalizzazione e overfitting, fornendo limitazioni superiori analitiche e quantitative al rischio R di predizione su tutti i dati, indipendentemente dal tipo di learning algorithm o dettagli del modello. Il ML è ben fondato, il rischio del learning (e dell'errore di generalizzazione) può essere analiticamente limitato! Si può trovare un buona approssimazione della f target da esempi, a patto di avere un buon numero di dati e una adeguata complessità del modello (misurabile formalmente con la VC-dim). Questo ci porta a nuovi modelli (SVM) (e altri metodi che direttamente considerano il controllo della complessità nella costruzione del modello). La STL fonda uno dei principi induttivi sul controllo della complessità.

Alcuni esempi di controllo della complessità sui modelli lineari sono il numero di parametri liberi w o la dimensione in input. Mentre per i Decision Trees il numero di nodi. Vedremo un approccio diretto alla complessità attraverso il modello SVM.

Support Vector machine \rightarrow Modello del tipo $h(x) = \text{Sign}(w^T x + b)$

Maximum Margin Classifier (No nonlinearity = linear)

Margine = Doppio della distanza tra l'iperpiano e i punti più vicini

C'è l'iperpiano di margine massimo che suddivide due dom.

Support Vectors: Sono i punti sul margine $|w^T x_i + b| = 1$

Tutti gli altri punti devono non essere dentro il margine, cioè $(w^T x_i + b)y_i \geq 1 \quad \forall i$

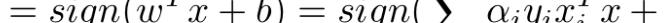
Due fatti a noi utili sono:

1. Margine = $2/\|w\|$ e sappiamo anche che $\|w\|^2 = (w^T w)$.
Quindi per massimizzare il margine \rightarrow minimizzare $\|w\| \rightarrow$ minimizzare $\|w\|^2/2$
 2. Il VC-dim del SVM è inversamente proporzionale al margine, quindi controlliamo la complessità del modello utilizzando il margine

Usando il problema duale otteniamo

$w = \sum \alpha_i y_i x_i$ e $b = y_k - w^T x_k$ per qualsiasi $\alpha_k > 0$.

$$h(x) = \text{sign}(w^T x + b) = \text{sign}\left(\sum_{i=1}^l \alpha_i y_i x_i^T x + b\right)$$



Value Training Set

Mangine Soft (Ammelittium dati numerosi)

Sappiamo che valere degli enzi E (slack), i.e.

problema (primal) viene mod. t.c:

minimizzare $|w|^2/2 + C \sum_i \xi_i$ tale che $(wx_i + b)y_i \geq 1 - \xi_i$ e $\xi_i \geq 0$ per ogni i

permessi. Alto = pochi errori, basso = molti errori
" Poss. overfitting Poss. Underfitting

l' Hand Margin si ottiene con un C molto alto

\Rightarrow la class. dipende solo dai support
vector perché per le altre i slante
 $\alpha = 0$!

SVM Spazi non lineari

Non usiamo la linear basis expansion perché rischiamo di aumentare il costo computazionale e l'overfitting se non gestiamo la complessità del modello e la dim. del feature space.

Uniamo il kernel trick per gestire la complessità del modello att. il margine e quindi mappare il feature space nel contesto di un modello regolarizzato automaticamente.

Feature Space ampio, ma migliore ampio =
complessità
negligeabile

- 7.1 Esempi di Kernel**

 - Lineare: $K(x_i, x_j) = x_i^T x_j$
Mapping $\phi: x \rightarrow \psi(x)$ dove $\psi(x)$ è x stesso
 - Polinomiale: $K(x_i, x_j) = (1 + x_i^T x_j)^p$ con p iper-parametro che indica il grado del polinomio
Mapping $\phi: x \rightarrow \psi(x)$ dove $\psi(x)$ ha dimensione esponenziale rispetto a p
 - RBF (radial-basis-function) Gaussiana: $K(x_i, x_j) = e^{-\|x_i - x_j\|^2/2\sigma^2}$ dove σ è un iper-parametro
Mapping $\phi: x \rightarrow \psi(x)$ dove $\psi(x)$ è di dimensione infinita
(molto potente sul TR ma può portare ad overfitting)

$(n + x_i^T x_j)^p$ dove $n \in \mathbb{R}$ determinato
 Tramite Cross Validation

Bias = Assumere che classi distinte vengono separate da un margine grande

8.8 Riassunto procedimento SVM con Kernel Fun.

- Scegliamo un parametro C (trade-off)
- Scegliamo una funzione di Kernel K (e i suoi parametri)
- Troviamo il miglior α
- Usiamo il modello finale

$$h(x) = \text{sign}(\sum_{i \in SV} \alpha_i y_i K(x_i, x))$$

K - NN (supervised) ↗ **Nessun enone di training**

9.1 1-Nearest Neighbor (Flessibilissimo - overfitting)

Questo algoritmo non impara, ma sfrutta tutti i valori del training set. **Decision-B. irregolare**

- Salviamo i dati di training nella forma (x_j, y_j) con $j=1\dots n$
- Dato un input x di dimensione n dobbiamo trovare l'esempio di training più vicino x_i tale che $d(x, x_i)$ è minimo
dove

$$\text{Distanza } d(x, x_j) = \sqrt{\sum_{t=1}^n (x_t - x_{jt})^2} = \|x - x_j\|$$

x_t è la componente t-esima di x , x_{jt} è la componente t-esima di x_j

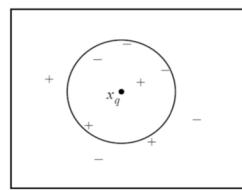
- Diamo come output y_i (in pratica stiamo vedendo l'esempio che più assomiglia ai nostri dati e rispondiamo come ha risposto lui)

9.2 K-NN (Meno fless., rispetto a K=1, ammette enone di training)

È la versione in cui si "guarda" il comportamento dei K vicini

Decision-B. più regolare

Rispetto a $K=1$



In questo caso 1-NN risponderebbe + per x_q , valutando invece con 5-NN viene restituito - per x_q . Come succedeva con il polinomio di alto grado che andava in overfitting, anche qui dobbiamo renderlo più "smooth" valutando su un insieme di vicini.

Per questo possiamo "dare un occhiata" a tutti (K) vicini e restituire una media:

$$\text{avg}_k(x) = \frac{1}{k} \sum_{x_i \in N_k(x)} y_i \quad \text{Media } K \text{ nodi vicini}$$

dove $N_k(x)$ è il vicinato di x che contiene esattamente K vicini. Se c'è una chiara dominanza di una delle classi nel vicinato di x , allora è probabile che anche x appartenga a quella classe. Quindi la regola di classificazione è la maggioranza che vota tra i membri di $N_k(x)$.

$$h(x) = 1 \text{ se } \text{avg}_k(x) > 0.5, 0 \text{ altrimenti} \quad \text{classificazione}$$

Dobbiamo trovare il giusto trade-off tra underfitting e overfitting trovando il giusto valore K . (Nota: è possibile usare K-NN anche se ci sono più classi).

K=1 overfitting
K=2 underfitting

È un modello: *lazy, memory based, instance-based, distance-based method*
Poco

Quale metà per la distanza uso

9.3 Considerazioni su K-NN

Non c'è una ipotesi globale per tutte le istanze: non c'è quindi nessun modello da allenare. Dobbiamo semplicemente memorizzare gli esempi di input, infatti è un metodo basato sulla memoria, sull'istanza e sulla distanza. Il bias inductive è scegliere il risultato in base alla distanza.

Notare che K-NN fa una approssimazione locale della funzione target per ogni nuovo esempio da classificare, il costo computazionale è tutto contenuto nella fase di predizione. Inoltre è molto costoso a livello computazionale perché per ogni nuovo input bisogna calcolare le distanze dall'esempio a tutti i vettori memorizzati. Il tempo è proporzionale al numero di modelli memorizzati, questo ci fa notare che anche il costo per lo spazio è alto, dato che tutti i dati sono memorizzati.

Quando abbiamo molte variabili in input K-NN spesso fallisce a causa del "curse of dimensionality": quando la dimensione aumenta, il volume dello spazio aumenta in modo tale che i dati disponibili diventano sparsi. Ad esempio la quantità di dati che servono per supportare un risultato spesso cresce esponenzialmente con la dimensione. Troviamo anche il problema denominato "curse of noisy": se il target dipende da poche altre variabili, potremmo trovarci un "modello simile" con la somiglianza dominata dal gran numero di funzioni irrilevanti.

9.4 Distance Based Methods

Quando utilizzi degli approcci basati su distanza (come K-NN o alcune funzioni Kernel) devi assicurarti che la distanza calcolata sia davvero un fattore discriminante. Stiamo misurando quando una coppia di pattern si assomiglia, inoltre dare una metrica pone un bias rilevante sulla soluzione. Possiamo imparare in qualche modo la metrica? Sì (Reti Neurali, Learning K...).

Bias: Ricerca = Scelta delle metrice per distanza
ling = Annunziate che punti vicini appartengono alla stessa classe

Altri task che non fanno parte del supervised o unsupervised learning.

Apprendimento con ricompense

- Reinforcement Learning: si utilizza come metodo di adattamento per sistemi autonomi (in particolare in robotica). L'algoritmo apprende un criterio su come agire data un'osservazione del mondo. Ogni azione ha un certo impatto sull'ambiente e l'ambiente fornisce un feedback che guida l'algoritmo di apprendimento. Invece di avere una supervisione per ogni passaggio, abbiamo informazioni su vittorie / perdite per lo stato finale. Le azioni devono massimizzare la quantità di vittorie ricevute. L'apprendimento decide quali azioni sono state maggiormente responsabili di vittorie / perdite.
- Semi-Supervised Learning: combina esempi etichettati e non etichettati (in genere in numero maggiore) per generare una funzione o un classificatore appropriati.
- Learn to Rank: (utilizzato per i motori di ricerca) quando l'input è un insieme di oggetti e l'output desiderato è una classifica (un ranking) di tali oggetti.
- On-Line Learning: nuovi esempi sono imparati al momento
- Structured domain learning and relational learning: il dominio di input e output può essere strutturato sotto forma di sequenze (segnali, serie temporali, ...) o in modo più complesso: alberi, grafici, reti sociali.

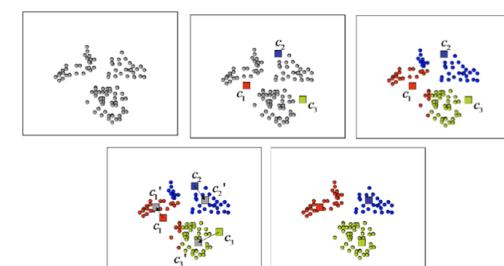
8.9 Utilizzare bene SVM

Evitare errori di interpretazione tipici nell'uso di SVM. È possibile che si verifichi un overfitting senza un'attenta selezione dei parametri: C, funzione Kernel, parametri del Kernel, ecc... Il trattamento implicito dello spazio dimensionale ampio deve avvenire nello spazio delle caratteristiche e non in quello di input. Anche la tecnica di validation vista fino ad ora per la selezione del modello e la valutazione del modello devono essere utilizzate rigorosamente. Quindi:

- Trasformare i dati in un formato leggibile da un software SVM (ad esempio {red, green, blue} → (0,0,1), (0,1,0), (1,0,0))
- Eseguire una semplice scalatura dei dati (ad esempio in un intervallo da [-1,1] o [0,1])
- Considerare il kernel $K(x_i, x_j) = e^{-||x_i - x_j||^2 / 2\sigma^2}$
- Usare la cross-validation per trovare i migliori parametri C e σ
- Riutilizzare il TR set però con i migliori C e σ trovati
- Eseguire il testing su un Test Set esterno

k-means (unsupervised)

- centroide** **centroide** **numero di raggruppamenti (cluster)**
1. Scelgono i centri di cluster in modo che coincidano con k modelli scelti casualmente (o k punti definiti casualmente all'interno dell'ipervolume contenente il set di pattern) $c_1 \dots c_k$ con k fissato che sceglio noi.
 2. Assegnare ad ogni modello il centro del cluster più vicino (il vincitore) per ogni x calcoliamo
- $$i^*(x) = \underset{i=1 \dots k}{\operatorname{argmin}} \|x - c_i\|_2^2 = \sum_{j=1}^n (x_j - c_{ij})^2$$
- cioè provo tutti i i e quello che ha distanza minima è il vincitore (mi interessa l'indice del vincitore i^*). Adesso x appartiene al cluster i^* (cluster con distanza minima del centroide rispetto a x)
3. Ricalcolare i centri del cluster (centroide) utilizzando le nuove appartenenze al cluster corrente
- $$c_i = \frac{1}{|\text{num di membri cluster}|} \sum_{x_j \in \text{cluster}_i} x_j$$
4. Se non viene soddisfatto un criterio di convergenza, andare al passaggio 2 (criteri come nessuna o minima riassegnazione di schemi a nuovi centri di cluster oppure una minima diminuzione dell'errore quadratico)



10.2 Limitazioni K-means

Il numero di cluster da trovare deve essere fornito (questo porta a fare trial and error per trovare il K che fitta meglio). I minimi locali della Loss rendono il metodo dipendente dall'inizializzazione, si esegue più volte da diverse inizializzazioni casuali (ci sono anche dei metodi che inizializzano con un'evidenza). K-means può funzionare bene per cluster compatti, ma non consente di proiettare i dati in uno spazio di dimensione minore.

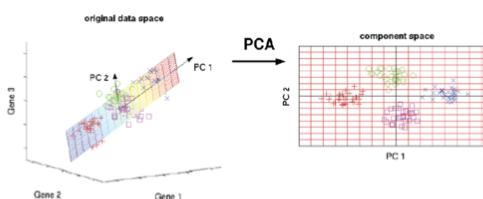
Come viene valutato l'output di un algoritmo di clustering? Che cosa caratterizza un risultato di raggruppamento "buono" e uno "scarsio"? Nel clustering esiste ben poco in termini di "standard di riferimento", tranne nei sottodominio specifici in cui conosci la metrica (conosci bene il problema e so riconoscere una classificazione sensata). Misure oggettive (non trattate qui) come ad esempio l'errore di quantizzazione. Noi sappiamo a quali classi appartengono i dati, ma usiamo un algoritmo di clustering, e controlliamo che le classi corrispondano. È un modo sbagliato per valutare la bontà di un algoritmo di clustering!

10.3 Preprocessing dei dati

Abbiamo visto che Unsupervised Learning viene utilizzato per il preprocessing dei dati, ma in che modo? Menzioniamo la riduzione della dimensionalità cioè trasformare la dimensione dei dati in una più piccola. (non è importante la quantità, bensì la dimensione dei dati!)

$< x_1, x_2, \dots, x_n > \rightarrow < x'_1, x'_2, \dots, x'_m >$ con $n > m$.

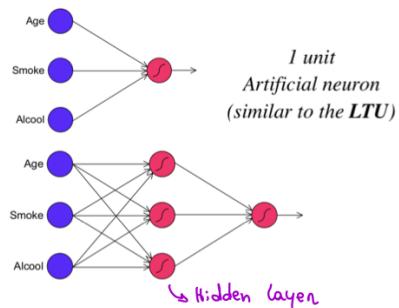
Un esempio è il PCA (principal component analysis) dove i nuovi assi sono calcolati nella direzione di massima varianza dei dati



Un altro approccio per il preprocessing dei dati è la feature selection: è una particolare tecnica di riduzione della dimensionalità nella quale scelgo le variabili più significative per il task, anziché trasformarle. In pratica scegliamo un sottosinsieme di tutte le caratteristiche. Un ultimo approccio dal preprocessing che vediamo è Outlier detection: si cercano valori inusuali dei dati che non sono consistenti con gli altri (valori scorretti a causa di misurazioni errate).

12.1 Reti Neurali

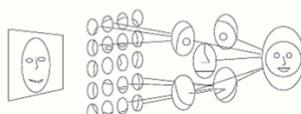
Vengono utilizzate sia nel Supervised che nel Unsupervised Learning. Sono simili alla visione del Linear Threshold Unit, una rete neurale è una rete di modelli non lineari con una capacità di approssimazione universale e capacità predittive. Gli strati interni sono "nascosti" e ogni unità è non lineare, fornendo alla rete neurale la capacità di estrarre (imparando) una nuova rappresentazione dei dati. Questa nuova rappresentazione semplifica il compito di classificazione nell'ultimo livello della rete. Abbiamo una espansione della base non lineare in w , e le ϕ sono imparate automaticamente, ma purtroppo questo scaturisce in un problema di ottimizzazione non lineare.



12.2 Deep Learning

Il Deep Learning è quel campo di ricerca dell'apprendimento automatico (machine learning) e dell'intelligenza artificiale che si basa su diversi livelli di rappresentazione, corrispondenti a gerarchie di caratteristiche di fattori o concetti, dove i concetti di alto livello sono definiti sulla base di quelli di basso. In altre parole, si intende un insieme di tecniche basate su reti neurali artificiali organizzate in diversi strati, dove ogni strato calcola i valori per quello successivo affinché l'informazione venga elaborata in maniera sempre più completa.

Più tecnicamente, quando si parla di Deep Learning, si fa riferimento a reti neurali multistrato profonde, nel senso che abbiamo molteplici layer di unità di processing non lineari. Indipendentemente se si usa il Supervised o Unsupervised Learning per la rappresentazione di caratteristiche in ogni livello, i vari livelli vanno a formare una gerarchia di caratteristiche / rappresentazioni da basso livello ad alto livello (diversi livelli di astrazione). Imparare automaticamente come rappresentare i dati, e capire come classificarli aumenta il livello di astrazione tra i vari layer. Ad esempio, un'immagine può essere rappresentata in molti modi: come un vettore di pixel o in un modo più astratto come un insieme di bordi, regioni di forma particolare, ecc... Per questo il Deep Learning funziona meglio quando i dati in input hanno una sorta di struttura: spaziale, temporale, linguistica ecc...



I vantaggi del Deep Learning sono svariati, come ad esempio la possibilità di sfruttare la composizionalità della rappresentazione interna che porta ad un guadagno esponenziale nel potere di rappresentazione. I concetti più semplici vengono rappresentati in uno strato della rete, i quali poi possono essere sfruttati come dati primitivi dal livello successivo per rappresentare concetti più complessi. Sono necessari meno esempi per raggiungere una buona capacità di generalizzazione. Le Deep Networks erano difficili da addestrare in passato, ma combinando tecniche per l'addestramento di modelli grandi, HPC (ad esempio GPU) e grandi raccolte di dati da applicazioni reali (ad esempio milioni di immagini), al giorno d'oggi lavorano molto bene conseguendo una rivoluzione nell'approccio AI alle soluzioni del mondo reale.