Laurea Magistrale in Informatica A.A. 2020/2021
Università degli Studi di Milano-Bicocca

Appunti Machine Learning

Pelusi Marta
@Marta629

Indice

1 Apprendimento **4**
1.1 Apprendimento supervisionato 4
1.2 Apprendimento non supervisionato 4
1.3 Apprendimento attivo e passivo 4
1.4 Apprendimento induttivo 4
1.5 Definizioni 4

2 Concept Learning **5**
2.1 General to specific relationship - GSR 5
2.2 Consistenza e soddisfacibilità 5

3 Cardinalità **6**
3.1 Cardinalità spazio delle istanze X 6
3.2 Cardinalità spazio dei concetti C 6
3.3 Cardinalità dello spazio delle ipotesi H 6

4 Find-S **6**
4.1 Insieme specifico e generale 6
4.2 Algoritmo Find-S 7
4.3 Version Space 7

5 Alberi di decisione - DT **7**
5.1 Numero alberi di decisione 8
5.2 Algoritmo 8

6 ID3 **8**
6.1 Proprietà ID3 8
6.2 Overfitting 8
6.3 Entropia 9
6.4 Information gain - IG 9

7 Percettrone e Reti neurali **9**
7.1 Perceptron learning - PLR 9
7.2 Separabilità lineare 10
7.3 Neural Networks 10

8 Byas concept learning **10**
8.1 Probabilità 10
8.2 Maximum A Posteriori hypothesis - MAP hypothesis 11
8.3 Maximum Likelihood Hypothesis - ML hypothesis 11

9 SVM, Kernel **11**
9.1 Support Vector Machine - SVM 11
9.1.1 Classificazione usando un iperpiano 11
9.1.2 Margine funzionale 12
9.1.3 Margine geometrico 12
9.1.4 SVM 12
9.2 Metodo Kernel 13

9.2.1 Kernel 13
9.2.2 Strategia di Kernel - kernellizzazione 13

1 Apprendimento

1.1 Apprendimento supervisionato

Si parla di **apprendimento supervisionato** se il training set è formato da istanze x_i che hanno tutte una propria etichetta associata $t(x_i)$: $\{(x_1, t(x_1), ..., (x_n, t(x_n)))\}$.
In questo modo si addestra il modello partendo da esempi di input-output.

L'algoritmo, attraverso una funzione f, trova il valore esatto corrispondente al valore target dell'input $f(x) = c(x)$.

1.2 Apprendimento non supervisionato

Si parla di **apprendimento non supervisionato** se il training set è formato da sole istanze x_i: $\{x_1, ..., x_n\}$.
L'apprendimento non supervisionato è utilizzato quando si vogliono riconoscere pattern o schemi nell'input. Non esiste un modello che fornisce risposte corrette per gli input. Un esempio di apprendimento non supervisionato è il *clustering*.

1.3 Apprendimento attivo e passivo

L'apprendimento può essere **passivo** se si può apprendere solo dai dati che vengono messi a disposizione, oppure **attivo** se si possono fare anche domande ed esperimenti.

1.4 Apprendimento induttivo

Si parla di **apprendimento induttivo** quando data una collezione di esempi $(x_i, f(x_i))$ con f funzione target, si vuole trovare un'ipotesi h che, dato un training set, approssima f il meglio possibile. Due esempi di apprendimento induttivo sono:

- classificazione
- regressione

1.5 Definizioni

Istanza x: singolo oggetto dell'universo del discorso utile a fare delle predizioni.
Spazio delle istanze X: spazio in cui sono raccolte le istanze $x \in X$.

Spazio dei concetti C: è un sottoinsieme dello spazio delle istanze $C \subseteq X$ che descrive una classe di oggetti ai quali siamo interessati per la nostra predizione. Rappresenta quindi tutti i possibili sottoinsiemi dello spazio delle istanze $C = \{c_i \subseteq X\} = \mathcal{P}(X)$.
L'obiettivo è trovare un'ipotesi più vicina possibile al concetto di nostro interesse.
Concetto: $c \subseteq X$ oppure $c : X \to \{0, 1\}$. Il concetto, quindi, è un insieme di istanze, ad esempio l'insieme delle persone alte più di 1.80m.

Ipotesi h: oggetto utile per fare predizioni. Ogni ipotesi è una congiunzione di vincoli su attributi di istanze.
Spazio delle ipotesi H: insieme di tutte le ipotesi considerate $h \in H$.

Ipotesi consistente: ipotesi h che accetta tutti gli esempi considerati $h(x) = 1$.

Modello: ipotesi finale ricavata tramite un algoritmo che mi consente di fare predizioni sulla base delle istanze.

Esempio: $(x, t(x))$ esempio formato da un'istanza x e la sua label $t(x)$. La collezione degli esempi $\{(x_1, t(x_1)), ..., (x_n, t(x_n))\}$ è chiamata **training set**.

Inductive bias: insieme di assunzioni che si fanno per predire l'output di istanze mai viste.

Cross validation: tecnica statistica utilizzabile in presenza di una buona numerosità del training set che consiste nella suddivisione del dataset in k parti e, ad ogni passo, la parte $1/k$-esima del training set diventa il validation set, mentre il resto costituisce il training set, e così via per ognuna delle k parti. In questo modo si allena il modello evitando il problema dell'overfitting e del campionamento asimmetrico.

Version space VS: insieme di ipotesi H consistenti con gli esempi di train in D. È indicato con:

$$VS_{H,D} \equiv \{h \in H \ tc: \ \text{consistent(h, D)}\}$$

2 Concept Learning

Si parla di **concept learning** ogni qualvolta si consideri il problema di cercare in X spazio delle istanze funzioni $f : X \to \{0, 1\}$ che assumano solamente valore $0/1$.
Si vuole trovare la migliore ipotesi $h \in H$ che si adatta meglio al concetto "celato" dietro il training set.

$$f : X \to \{0, 1\} \qquad f(x) = \begin{cases} 1 & x \in A \\ 0 & x \notin A \end{cases}, \quad \text{con } A \subseteq X$$

Prese delle ipotesi $h \in H \subseteq X$ si vuole cercare la miglior ipotesi

$$h \in H \ tc: \ h(x) = c(x), \ \forall x \in D$$

dove $c(x)$ è la label associata all'istanza x e D è il training set.
In questo caso l'ipotesi h viene chiamata **ipotesi consistente**, quindi $h(x) = c(x) = 1$ perchè x soddisfa l'ipotesi h.

2.1 General to specific relationship - GSR

$h_j \geq_g h_k$ (relazione d'ordine parziale), cioè h_j è GSR h_k se e solo se

$$\forall x \in X \quad h_k(x) = 1 \Rightarrow h_j(x) = 1$$

2.2 Consistenza e soddisfacibilità

Si dice che un'ipotesi h è **consistente** con un training set D se

$$\text{consistent}(h, D) \equiv \forall \langle x, c(x) \rangle \in D \quad h(x) = c(x)$$

ovvero se azzecca le etichette presenti nel training set in riferimento all'istanza x.

Si dice che un'istanza x è **soddsifacibile** per un'ipotesi h se $h(x) = 1$ indipendentemente dalla classe di appartenenza di x.

3 Cardinalità

3.1 Cardinalità spazio delle istanze X

Dati due attributi $A_1 = \{0, 1, 2\},\ A_2 = \{0, 1\}$, si ha che

$$|X| = |A_1 \cdot A_2| = 3 \cdot 2 = 6$$

3.2 Cardinalità spazio dei concetti C

Dati tre attributi $A_i = \{0, 1, 2\},\ i = 1...3$, si ha che $|X| = |A_1 \cdot A_2 \cdot A_3| = 3 \cdot 3 \cdot 3 = 27$ e quindi si ha che

$$|C| = |\mathcal{P}(X)| = 2^{|X|} = 2^{27}$$

3.3 Cardinalità dello spazio delle ipotesi H

$|X|$ è il numero delle diverse combinazioni di possibili valori per ogni attributo.

Dati tre attributi $A = \{0, 1\},\ B = \{0, 1, 2\},\ C = \{0, 1, 2, 3\},\ |A| = 2,\ |B| = 3,\ |C| = 3$ si ha:

Cardinalità semantica: $(2 + 1) \cdot (3 + 1) \cdot (4 + 1) + 1 = 61$
(i "+1" intermedi sono per l'ipotesi più generale, mentre il "+1" finale è per l'ipotesi più specifica).
Cardinalità sintattica: $(2 + 1 + 1) \cdot (3 + 1 + 1) \cdot (4 + 1 + 1) = 120$
(i "+1+1" sono sia per l'ipotesi più generale sia per l'ipotesi più specifica).

NB semantica: si parla di semantica ogni qualvolta ci si riferisce al *significato* di qualcosa, in questo caso al significato di ipotesi.
Ad esempio le ipotesi $h_1 = \langle 1, 0, \emptyset, 0\rangle$ *e* $h_2 = \langle 1, 0, 1, \emptyset\rangle$ sono semanticamente equivalenti perchè sono classificate entrambe come negative, in quanto hanno un attributo che non accetta niente ($\emptyset$).

NB sintattica: si parla di sintattica ogni qualvolta ci si riferisce alla *"grammatica"/simboli* di qualcosa, in questo caso alla "grammatica"/simboli delle ipotesi.
Ad esempio le ipotesi $h_1 = \langle 1, 0, \emptyset, 0\rangle$ *e* $h_2 = \langle 1, 0, 1, \emptyset\rangle$ non sono sintatticamente equivalenti perchè presentano attributi diversi nelle rispettive posizioni f_1, f_2, f_3, f_4.

4 Find-S

4.1 Insieme specifico e generale

L'**insieme generale** delle ipotesi rappresenta l'insieme delle ipotesi più generiche, ovvero $h = \langle ?, ..., ?\rangle$.

L'**insieme specifico** delle ipotesi rappresenta l'insieme delle ipotesi più specifiche, ovvero $h = \langle \emptyset, ..., \emptyset \rangle$.

4.2 Algoritmo Find-S

1. inizializzo h all'ipotesi più specifica in H, ovvero $h = \langle \emptyset, ..., \emptyset \rangle \in H$
2. $\forall d \in D$ (istanze del training set)
   - salta gli esempi con target negativo $c(x) = 0$
   - per gli esempi con target positivo $c(x) = 1$: per ogni attributo $a_i \in h,\ i = 1...n$ se a_i è soddisfatto da d non faccio niente, altrimenti sostituisco $a_i \in h$ mettendo il vlore corretto che soddisfa l'istanza d
3. l'output finale è h

Proprietà Find-S

L'output è l'ipotesi più specifica in $h \in H$ che è consistente con tutti gli esempi positivi. L'potesi h sarà consistente anche con gli esempi negativi a patto che il concetto target è contenuto in H.

Problemi Find-S

Non si può sapere se si è trovato un'unica ipotesi consistente e se gli esempi di train sono inconsistenti tra loro.

4.3 Version Space

Il **version space** VS rispetto ad uno spazio delle ipotesi H e un training set D è il sottoinsieme di ipotesi di H, $VS \subseteq H$ che è consistente con tutti gli esempi di train

$$VS = \{h \in Htc : \text{ consistent}(h, D)\}$$

5 Alberi di decisione - DT

Un **albero di decisione** è un elemento esecutivo che prende in ingresso un oggetto o una situazione descritta mediante un insieme di attributi e restituisce una deicisione.

Gli alberi di decisione sono caratterizzati da:

- **Nodo**: test sul valore di un attributo
- **Ramo**: valore che può assumere l'attributo
- **Foglia**: classificazione del problema (output)

Con un albero di decisione è possibile rappresentare una qualsiasi funzione booleana. L'intero albero è una disgiunzione di congiunzioni e il percorso da radice a foglia è una congiunzione di attributi e di valori.

5.1 Numero alberi di decisione

Se ho n attributi quanti sono i possibili alberi di decisione che posso costruire?
Con n attributi posso costruire una tabella di verità con 2^n possibili righe (perchè gli attributi sono booleani 0/1). Di conseguenza avrò 2^{2^n} possibili assegnamenti di verità perchè ad ogni riga posso assegnare 2 possibili assegnamenti (0/1 oppure F/T). 2^{2^n} sono i possibili alberi di decisione.

5.2 Algoritmo

Algoritmo di costruzione di un albero di decisione:

1. inizializzo un albero vuoto
2. seleziono un attributo per splittare i dati
3. se non c'è più niente da fare → nodo foglia
4. continuo ricorsivamente dallo step 2

Lo **splitting** è il partizionamento dei dati. I dati ben partizionati sono quelli che hanno una distribuzione omogenea, cioè hanno un'incertezza molto bassa, al contrario dei dati mal partizionati che hanno una alta incertezza, ad esempio $[2-, 2+]$ è un dato mal partizionato.

6 ID3

L'**algoritmo ID3** è un approccio greedy di costruzione di un albero di decisione selezionando volta per volta l'attributo migliore, ovvero quello con partizionamento degli attributi migliore, ovvero quello che ha incertezza/entropia minima e information gain maggiore.

Lo scopo è ottenere alberi di decisione meno profondi possibile consistenti con gli esempi di train e con contributo informativo maggiore più in alto (verso il nodo radice).

6.1 Proprietà ID3

L'algoritmo ID3 gode di due principali proprietà:

- lo spazio delle ipotesi è completo
- l'algoritmo è robusto a rumori nei dati

6.2 Overfitting

L'**overfitting** è un concetto fondamentale che indica il rischio di sovradettamento durante il processo di apprendimento induttivo. Ciò significa che l'algoritmo si adatta (*fitting*) troppo bene (*over*) alle istanze di train perdendo in generalità, ovvero si otterranno risultati ottimali in fase di train e risultati poco ottimali in fase di test.

Il tutto può essere riassunto dalla seguente formula: considerato D il training set e presa un'ipotesi $h \in H$, h va in overfit con il training set se $\exists\ h' \in H$ tale che

$$\text{error}_{\text{train}}(h) < \text{error}_{\text{train}}(h') \wedge \text{error}_{\text{D}}(h) > \text{error}_{\text{D}}(h')$$

6.3 Entropia

L'**entropia** H (non è lo stesso H di prima - stessa notazione ma cambia il contesto) rappresenta il grado di incertezza.

L'incertezza di un evento E è data dalla formula:

$$I(E) = -\log_2 P(E)$$

Entropia di una tabella T:

$$H[T] = -\sum_{i=1}^{m} P_i \log_2 P_i$$

Entropia di una distribuzione condizionale:

$$H_{T|X=x_i} = -\sum_{j=1}^{m} P_{T|X}(T_j|x_i) \log_2 P_{T|X}(T_j|x_i)$$

Entropia condizionata:

$$H[T|X = x_i] = \sum_{j=1}^{m} P(x_i) H[T_j|x_i]$$

Nota: *per avere chiare queste formule guarda esempi di esercizi negli appunti dell'esercitazione 3 del 28/10/2020.*

6.4 Information gain - IG

L'**information gain** è il valore che indica quanto è buono il partizionamento/split dell'albero di decisione.
Maggiore è il valore di IG e migliore è il partizionamento/split dell'albero.

L'information gain si calcola attraverso la seguente formula:

$$IG[T; X = x_i] = H[T] - H[T|x_i]$$

7 Percettrone e Reti neurali

7.1 Perceptron learning - PLR

Il **percettrone** è la più semplice rete neurale.

La rete è passata alla funzione di attivazione e l'output di questa funzione è usato per aggiustare i pesi.

Gli output generalmente sono ristretti ai valori -1, 0, 1.
Il segnale d'apprendimento è la differenza tra la risposta desiderata e la risposta effettiva di un neurone.

L'errore σ nella regola delta rule (DR) non si limita ad avere valori -1, 0, 1 come nel PLR, ma può avere qualsiasi valore.

Perceptron usa le etichette delle classi per imparare i coefficienti del modello, ovvero prende una risposta binaria come risultato della classificazione e lo usa per aggiornare i pesi.
Adaline usa i valori continui previsti dall'input per imparare i coefficienti del modello, il quale è più potente perchè dice "quanto abbiamo ragione o torto".

7.2 Separabilità lineare

Un insieme di due tipologie di punti è **linearmente separabile** se esiste un iperpiano/retta che può separare le due tipologie di punti in modo netto.

7.3 Neural Networks

Una **rete neurale** è formata da uno strato di input, uno o più strati di livelli nascosti (hidden layers) e uno strato di output. Gli hidden layer, di norma, sono di numero inferiore rispetto agli input.
Ogni nodo in uno strato nascosto opera sulle attivazioni dello strato precedente e trasmette le attivazioni in avanti ai nodi dello strato successivo. Per questo le reti neurali sono anche chiamate "feedforward neural networks".

8 Byas concept learning

8.1 Probabilità

La miglior ipotesi è contenuta nello spazio delle ipotesi H avente traininset D.

Teorema di Bayes è un metodo per calcolare la probabilità a posteriori di h dalla probabilità a priori $P(h)$ unita a $P(D)$ e $P(D|h)$, ovvero

$$P(h|D) = \frac{P(D|h)P(h)}{P(D)}$$

dove

- D è il dataset
- h è il parametro
- $P(D|h)$ è la probabilità
- $P(h|D)$ è la probabilità a posteriori
- $P(h)$ è la probabilità a priori

L'**evidenza** $P(D)$ è calcolata tramite la formula

$$P(D) = \sum_i P(D,h_i) = \sum_i P(D|h_i)P(h_i)$$

Spesso $P(D)$ è difficile da calcolare, così è considerato solo il prodotto tra la probabilità a priori e la probabilità

$$P(h_i|D) \propto P(D|h_i)P(h_i)$$

8.2 Maximum A Posteriori hypothesis - MAP hypothesis

Un'ipotesi massimamente probabile è detta **massima ipotesi a posteriori - MAP ipotesi**.

$$h_{MAP} = \arg\max_{h\in H} P(h|D) = \arg\max_{h\in H} \frac{P(D|h)P(h)}{P(D)} = \arg\max_{h\in H} P(D|h)P(h)$$

8.3 Maximum Likelihood Hypothesis - ML hypothesis

$P(D|h)$ è l'ipotesi di D dato h.
Se ogni ipotesi a propri fosse equiprobabile, ovvero $P(h_i) = P(h_j),\ \forall h_i, h_j$ allora ogni ipotesi che massimizza $P(D|h)$ è chiamata **maximum likelihood hypothesis - ML hypotesis - hML**

$$h_{ML} = \arg\max_{h\in H} P(D|h)$$

9 SVM, Kernel

9.1 Support Vector Machine - SVM

Un **iperpiano** in uno spazio euclideo n-dimensionale è un sottoinsieme piatto 1-dimensionale che divide lo spazio in due parti sconnesse.
Un iperpiano è un insieme di punti che soddisfa la seguente relazione

$$w \cdot x + b = 0$$

- in 1-dimensione un iperpiano è un **punto**
- in 2-dimensioni un iperpiano è una **retta**
- in 3-dimensione un iperpiano è un **piano**

9.1.1 Classificazione usando un iperpiano

Si può definire una funzione di ipotesi

$$h(x_i) = \begin{cases} +1 & \text{se } w \cdot x_i + b \geq 0 \\ -1 & \text{se } w \cdot x_i + b < 0 \end{cases}$$

che è equivalente a

$$h(x_i) = sign(w \cdot x_i + b)$$

9.1.2 Margine funzionale

Il **margine funzionale** è usato come espressione della bontà della classificazione, ovvero ci dà la misura di confidenza.

Si vuole computare il valore $\beta = w \cdot x + b$ per capire quanto i punti sono distanti dall'iperpiano.
Dato un esempio di train (x, y) si fa $f = y \cdot \beta = y \cdot (w \cdot x + b)$ (margine funzionale per un singolo esempio).
Il segno di f potrà essere

- positivo se il punto è classificato correttamente
- negativo se il punto non è classificato correttamente

Se $y = 1$ affinchè il margine sia ampio (previsione sicura e corretta) si ha bisogno che $w \cdot x + b \gg 0$.
Se $y = -1$ affinchè il margine sia ampio (previsione sicura e corretta) si ha bisogno che $w \cdot x + b \ll 0$.
Se $y \cdot (w \cdot x + b) > 0$ allora la previsione su (x, y) è corretta.

Il margine funzionale per tutti gli esempi che si stanno considerando (e non più per il singolo esempio) è denotato secondo la seguente formula:

$$\widehat{\gamma}^{(i)} = y^{(i)}(w^T x + b),\ \widehat{\gamma} = \min_{i=1\ldots m} \widehat{\gamma}^{(i)}$$

Osservazioni

La classe predetta dipende solo dal valore di $h(x_i) = sign(w \cdot x + b)$. Infatti se si moltiplica $2w$ e $2b$ si arriva ad ottenere lo stesso iperpiano: $h(wx + b) = h(2wx + 2b)$ producono rispettivamente gli iperpiani (w, b) e $(2w, 2b)$ che sono lo stesso.
Questo sembra che il margine funzionale aumenti, ma non è così (falsa confidenza)!

Si vuole avere, dunque, una maggiore confidenza, per questo si introduce il concetto di margine geometrico.

9.1.3 Margine geometrico

Il **margine geometrico** è invariante rispetto al fattore di scala ed è definito, considerando l'intero dataset, dalla seguente relazione:

$$\gamma^{(i)} = y^{(i)} \left(\left(\frac{w}{||w||} \right)^T \cdot x^{(i)} + \frac{b}{||w||} \right),\ \gamma = \min_{i=1\ldots m} \gamma^{(i)}$$

9.1.4 SVM

Ipotizzando di avere punti linearmente separabili, l'obiettivo è trovare un iperpiano di separazione ottimale, ovvero l'iperpiano che rende massimo il margine per entrambe le classi che si stanno considerando nel traininset, dove con margine si intende la distanza tra i punti più vicini delle diverse classi.
Il problema della SVM è massimizzare il minimo margine geomtrico.

9.2 Metodo Kernel

Se i punti non sono lienarmente separabili conviene rimapparli in uno spazio di dimensione maggiore in cui si possono separare lineamente. Ovviamente i punti poi avranno più componenti perchè risiederanno in uno spazio con dimensione maggiore.

9.2.1 Kernel

Funzione $K : X \times X \to \mathbb{R}$, $\phi : X \to F$ che rappresenta lo spazio di arrivo/spazio delle features attraverso un prodotto interno, tale che

$$K(x, z) = \langle \phi(x), \phi(z) \rangle \, , \ \forall x, z \in X$$

dove

- X è lo spazio delle istanze
- F è lo spazio delle features

Il **kernel** può essere pensato come una somiglianza, cioè una funzione di similarità che dice quanto sono simili gli oggetti che ha funzione stessa ha come argomenti.

Il kernel definisce un prodotto tra due vettori e uno spazio che ha il prodotto tra due vettori automaticamente ha anche la norma: $||x|| = \sqrt{dot(x, x)}$.
Se si ha la norma si ha anche la distanza tra due vettori calcolata come $d(x, y) = ||x-y||$.
La misura di similarità della funzione kernel è l'inverso della distanza, che si ha automaticamente.

Kernel matrix - Gram matrix

$$G_{i,j} = \langle \phi(x_i), \phi(x_j) \rangle_H = K(x_i, x_j), \ \forall \text{ compoente di una funzione kernel}$$

9.2.2 Strategia di Kernel - kernellizzazione

Si può sempre pensare la seguente uguaglianza:

kernel nello spazio originale $= K(x, x') = \langle \phi(x), \phi(x') \rangle_H =$ prodotto interno nel nuovo spazio delle fe

Poiché è possibile costruire una funzione $K(x, x') = \langle \phi(x), \phi(x') \rangle$, in modo tale che, durante l'apprendimento, ogni volta che abbiamo bisogno di calcolare $\langle \phi(x), \phi(x') \rangle$, possiamo solo valutare $K(x, x')$, cioè non abbiamo bisogno di applicare ed esprimere effettivamente la funzione $\phi(x)$. In questo modo, diciamo che la trasformazione esisterebbe solo "implicitamente".

Un metodo è **kernellizzato** se ogni feature di un vettore $\psi(x)$ appare solo all'interno di un prodotto interno con un'altra caratteristica di un vettore $\psi(x')$.
Si usa il **metodo kernellizzato** se $\langle \phi(x_i), \phi(x_j) \rangle$ può essere valutato da una qualsiasi funzione kernel $K(x, x')$ nello spazio di rappresentazione originario.