Artifical Intelligence Course Notes

Gabriele Pappalardo

Machine Learning

Iniziamo questo paragrafo definendo cosa si intende con *Machine Learning*:

> (**Machine Learning**). A computer program is said to lean from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E.

Possiamo vedere il Machine Learning come un problema di ricerca se osserviamo l'argomento dal punto di vista dell'Intelligenza Artificiale.

Concept Learning

> (**Concept Learning**). Inferring a boolean-valued function from training examples of its input and output.

L'insieme degli oggetti sul quale è definito un concetto è chiamato insieme delle **instanze**, che indichiamo con X. Il concetto o la funzione da apprendere viene chiamata **target concept** e viene denotata con c. Generalmente la c puo' essere una qualsiasi funzione booleana definita sulle instanze $x \in X$, cioè:

$$c : X \to \{0, 1\}$$

Nel Concept Learning le ipotesi $h \in H$ sono descritte da formule normali congiuntive di vincoli sugli attributi. I vincoli potrebbero assumere valore "?" (il vincolo puo' assumere qualsiasi valore) o "$\emptyset$" (il vincolo **NON** puo' assumere nessun valore). Il training set viene indicato con l'insieme D.

Lo scopo da raggiungere è quello di determinare una funzione ipotesi $h \in H$ tale per cui vale:

$$\forall x \in X . h(x) = c(x)$$

Le istanze per cui vale $c(x) = 1$ sono dette instanze **positive**, invece, le instanze in c assumere 0 sono dette **negative**. Un training example viene definito dalla coppia $\langle x, c(x) \rangle$.

Dato un insieme di training examples del target concept c, lo scopo del learner è quello di di ipotizzare, o stimare, la funzione c.

> (**The inductive learning hypothesis**). Any hyphotesis found to approximate the target function

> well over a sufficiently large set of training examples will also approximate the target function well over other unobserved examples.

Il Concept Learning puo' essere visto come un task di ricerca su un grosso spazio di ipotesi implicitamente definite dalla rappresentazione di ipotesi. Lo scopo di questa ricerca è trovare la funzione che "fits" i training examples.

Molti algoritmi per il concept learning organizzano la ricerca sullo spazio delle ipotesi affidandosi su una struttura che esiste per un qualsiasi problema di concept learning: *un ordine di ipotesi dalle piu' generale alle piu' specifiche*. Avendo questo ordinamento possiamo disegnare algoritmi di apprendimento che visitano lo spazio di ricerca in maniera esaustiva senza andare ad enumerare tutte le ipotesi contenute in H.

Definiziamo il concetto di funzione "piu' generica di" (una relazione d'ordine matematica). Per prima cosa, per ogni instanza $x \in X$ e ipotesi $h \in H$, diciamo che x **soddisfa** h se e solo se $h(x) = 1$. Definiamo ora la relazione *mgtoet* (_more general than or equal to) in termini degli insiemi di instanze che soddisfano due ipotesi: date le ipotesi h_j e h_k, diciamo che h_j è piu' generale o uguale a h_k se e solo se ogni instanza che soddisfa h_k soddisfa anche h_j.

> (**More General than or equal to**). Let h_j and h_k be boolean-valued functions defined over X. Then h_j is **more general than or equal to** h_k (written $h_j \geq_g h_k$) iff:
>
> $$\forall x \in X.[h_k(x) = 1 \implies h_j(x) = 1]$$

La relazione $\geq_g$ definisce un ordine parziale sullo spazio delle ipotesi H, poichè vi possono essere ipotesi che non sono confrontabili tra di loro.

Vedremo negli algoritmi mostrati di seguti come questo ordinamento parziale ci sara' di grande aiuto.

Ipotesi piu' specifica:

$$h = \langle \emptyset, \ldots, \emptyset \rangle$$

Ipotesi piu' generale:

$$h = \langle ?, \ldots, ? \rangle$$

Find-S Algorithm (Find-Specific)

Definiamo il pseudo-codice dell'algoritmo *Find-S*:

```
1. Initialize h to the most specific hypothesis in H
2. for each positive training instance x:
     for each attribute constraint a[i]:
       if the constraint a[i] is satisfied by x then
```

```
        do nothing
      else
        replace a[i] in h by the next more general constraint that is
           satisfied by x
3. Output hypothesis h
```

Grazie all'ordinamento parziale espresso dalla relazione "*more general or equal to*" possiamo costruire un algoritmo che va alla ricerca della funzione di ipotesi h "maximally" specifica all'interno dello spazio delle ipotesi H. Per far cio' partiamo dall'ipotesi piu' specifica contenuta nello spazio H:

$$h = \langle \emptyset, \ldots, \emptyset \rangle$$

Questa funzione di ipotesi è molto specifica. In particalore, nessuno dei vincoli "$\emptyset$" è soddisfatto in h, dunque ogni vincolo viene sostituito dal vincolo piu' generale che soddisfa la training instance x. L'algoritmo dunque procede analizzando le altre training instances, ignorando quelle negative (che sono automaticamente soddisfatte!). La proprieta' chiave dell'algoritmo Find-S è che per lo spazio delel ipotesi descritto dai vincoli di attributo, Find-S garantisce di dare in ouput l'ipotesi piu' specifica contenuta in H che è consistente con i training examples positivi. La sua ipotesi finale sara' anche consistente con i negative examples forniti dalla corretta target concept che è contenuta in H.

L'algoritmo Find-S ignora ogni training example negativo.

Version Spaces and the Candidate-Elimination Algorithm

Il *Candidate-Elimination Algorithm* trova tutte le ipotesi che sono consistenti con i training examples osservati. Per poter definire questo algoritmo necessitiamo delle piccole definizioni di base.

> (**Consistent**). A hypothesis h is consistent with a set of trainning examples D if and only if $h(x) = c(x)$ for each example $\langle x, c(x) \rangle \in D$.
>
> $$\text{Consistent}(h, D) \equiv (\forall \langle x, c(x) \rangle \in D).h(x) = c(x)$$

Il *Candidate-Elimination* algorithm rappresenta l'insieme di tutte le ipotesi consistenti con i training examples osservati. Questo sotto-insieme di ipotesi è chiamato *version space* rispetto allo spazio delle ipotesi H e il training set D, perchè contiene tutte le versioni plausibili del *target concept* c.

> (Version Space). The version space, denoted $\text{VS}_{H,D}$ with respect to hypothesis space H and training examples D, is the subset of hypothesis from H consistent with the training examples in D.
>
> $$\text{VS}_{H,D} \equiv \{h \in H | \text{Consistent}(h, D)\}$$

List-Then-Eliminate Algorithm

```
1. VS = a list containing every hypothesis in H
2. For each training example, <x, c(x)>
    remove from VS any hypothesis h for which h(x) != c(x)
3. Output the list of hypothesis in VS
```

A more compact representation for Version Spaces

Il problema del *List-Then-Eliminate* algorithm è che enumera tutte le ipotesi che sono contenute nel Version Space, il che purtroppo non è molto efficiente. Il *Candidate Elimination* algorithm rappresenta il Version Space in una maniera piu' furba, praticamente si usano i membri piu' generici e meno generici possibili. Questi membri formano un limite generale e uno specifico che delimita il version space contenuto nello spazio delle ipotesi parzialmente ordinato.

Il *Candidate Elimination* algorithm rappresenta il Versions Space tenendosi in memoria solo le ipotesi piu' generiche (denotate con l'insieme G) e le ipotesi piu' specifiche (denotate con l'insieme S). Avendo solo G e S è possibile enumerare tutti i i membri del version space necessari, generando le ipotesi che sono contenute tra i due insiemi in un ordine parziale *general-to-specific*.

> (General Boundary). The **general boundary** G, with respect to hypothesis space H and training data D, is the set fo maximally general members of H consistent with D.

$$G \equiv \{g \in H\}$$

> (Specific Boundary). The **specific boundary** G, with respect to hypothesis space H and training data D, is the set fo minimally general members of H consistent with D.

$$S \equiv \{s \in H\}$$

Finchè gli insiemi G ed S sono ben definiti, allora possono specificare il version space completo. In particolare, possiamo dimostrare che il versiin space è l'insieme delle ipotesi contenute in G, piu' quelle contenute in S, piu' quelle contenute tra G e S in uno spazio delle ipotesi parzialmente ordinato.

> (Theorem: Version space representation theorem). Let X be an arbitrary set of instances and let H be a set of boolean-valued hypothesis defined over X. Let $c : X \to \{0, 1\}$ be an arbitrary target concept defined over X, and let D be an arbitrary set of training examples $\{\langle x, c(x)\rangle\}$. For

> all X, H, c and D such that S and G are well defined:
>
> $$\text{VS}_{H,D} = h \in H | (\exists s \in S)(\exists g \in G)(g \geq_g h \geq_g s)$$

Candidate-Elimination Learning Algorithm

```
Initialize G to the set of maximally general hyphotheses in H
Initialize S to the set of maximally specific hyphotheses in H

for each traing example d, do
  if d is a positive example then
    remove from G any hypothesis incosistent with d
    for each hypothesis s in S that is not consistent with d
      remove s from S
      add to S all minimal generalizations h of s such that h is
         consistent with d, and some member of G is more general than h
      remove from S any hypothesis that is more general than another
         hypothesis in S
  else
    remove from S any hypothesis incosistent with d
    for each hypothesis g in G that is not consistent with d
      remove g from G
      add to g all minimal specializations h of g such that h is
         consistent with d, and some member of S is more specific than
         h
      remove from G any hyphothesis that is less general than another
         hyphothesis in G
```

Linear Models

(Regressioni, Classificazioni, Basis Expansion, Regularization)

Regression

Processo di stima di funzioni a valori reali sulla base di un insieme di campioni rumorosi. Vengono date un insieme di coppie $x, f(x) + \text{random noise}$.

$$h(x) = w_1 x + w_0$$

Vogliamo un algoritmo che trovi i valori di w_i in modo intuitivo e "sistematico". La h in questo caso è una funzione lineare (noi trattiamo solo i casi di regressione lineari).

Noi ci concentriamo su una versione semplificata del problema, chiamata Univariate Linear Regression dove abbiamo una sola variabile di input e una sola variabile di output.

> We assume a model $h_w(x)$ expressed as $\text{out} = w_1 x + w_0$

> Trovare una h (linear model) che fitta i dati osservati.

Dobbiamo trovare i valori del parametro w in modo da minimizzare l'errore del modello in output. Purtroppo abbiamo un infinito spazio delle ipotesi ma grazie alle soluzioni della matematica classica abbiamo degli strumenti che ci permettono di fare "apprendimento" (nel senso moderno).

Least Mean Square (LMS) Il nostro scopo di apprendimento è trovare un vettore w che minimizza l'error fitting sul training set fornitoci.

> (Find). Give a set of l training example, find $h_w(x)$ in the form $w_1 x + w_0$ that minimizes the expected loss on the training data.

> (LMS): Find w to minimize the residual sum of squares:
>
> $$\text{Loss}(h_w) = E(w) = \sum_{p=1}^{l} (y_p - h_w(x))^2$$
>
> Usiamo l'elevamento al quadrato perchè si rischierebbe di non approssimare correttamente la funzione (gli errori si potrebbero compensare).

Perchè usiamo il LMS per trovare h? Perchè ci permette di trovare w per minimizzare la somma dei residui quadratici.

Ricordiamo che il minimo locale corisponde a punto stazionario dove il gradiente è nullo, dobbiamo trovare una soluzione dove la loss è al minimo possibile.

$$\frac{\partial E(w)}{\partial w_i} = 0$$

Questa equazione ci permette di trovare w_0 e w_1.]

Gradienti Quella vista prima è la soluzione diretta, ma nel nostro corso utilizzermo maggiormente la soluzione presentata con la ricerca locale.

Il Gradiente ci permette di impostare un algoritmo di ricerca locale. (Fa da bussola in uno spazio inifinito).

> (Gradient). Ascent direction: we can move towrd the minimum with a gradient descent ($\Delta w = -\text{gradient of } E(w)$)

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

Eta è una costante chiamata **learning rate**.

Questo del metodo del gradiente è una regola di "correzione di errore" (anche chiamata **delta rule**)che cambia w proporzionalmente al errore.

- ($y - \text{output} = \text{err} = 0$): nessuna correzione
- $\text{output} > \text{target} \implies (y - h) < 0$: output is too high
- $\text{output} < \text{target} \implies (y - h) > 0$: output is too low

Classifications

TODO

Basis Expansion

TODO

Regularization

TODO

Decision Trees

…