Lezione 3 13/10/2023

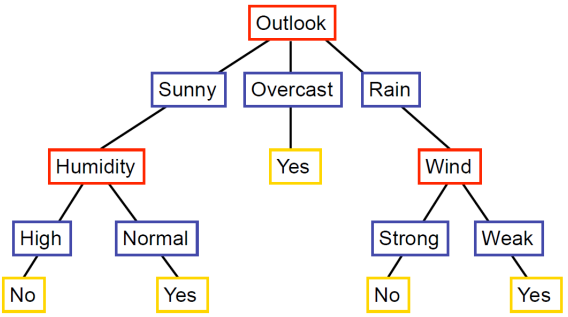
Alberi di decisione

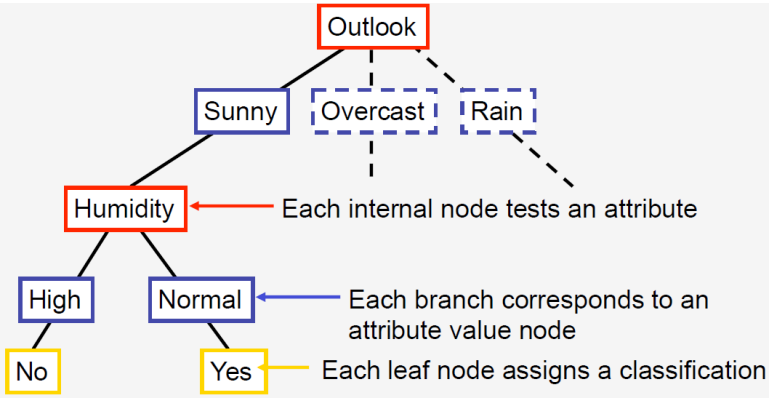
Per alcuni sistemi di apprendimento basta il sistema degli alberi di decisione. Con l'algoritmo ID3 sarà possibile scegliere l'albero di apprendimento migliore.

Torniamo agli esempi dove il risultato è booleano e l'algoritmo deve trovarne la logica.

L'albero descrive le istanze date in input.

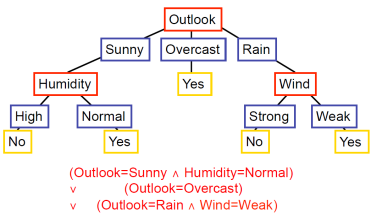
Il sistema dovrà scegliere tra tutte le possibili ipotesi quella che dopo tutti gli esempi sembra buona.





Nel mio sistema di apprendimento dovrò scegliere l'albero buono.

Grazie agli alberi posso rappresentare anche disgiunzioni (or) e sono quindi più espressivi delle congiunzioni semplici.



How many distinct decision trees with n Boolean attributes?

= number of Boolean functions

= number of distinct truth tables with 2^n rows = 2^{2^n}

- E.g., with 6 Boolean attributes, there are

18,446,744,073,709,551,616

Bisogna quindi capire come scegliere l'albero buono.

ID3 - top-down induction of DTs

Dalla cima al fondo si fa crescere l'albero di decisione, a induzione.

Servirà di vedere gli esempi a **gruppi**, non uno alla volta. L'iterazione che migliora man mano l'albero sarà fatta per gruppi. Questi gruppi verranno analizzati scegliendo un attributo nel nodo.

Scelgo l'attributo, per ognuno dei rami che scendono scelgo altri attributi. Gli attributi non si ripetono.

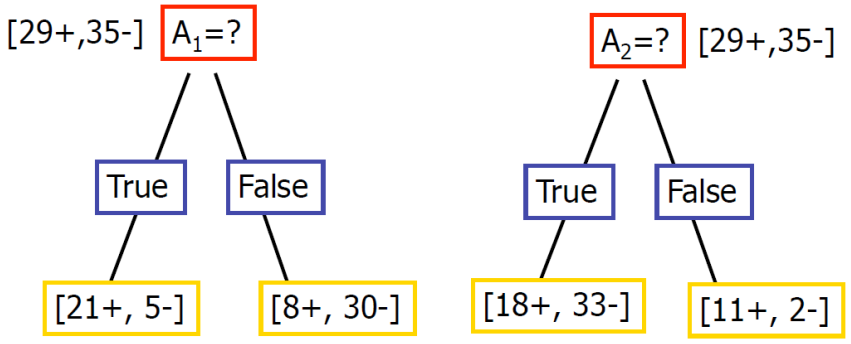
Quindi la parte difficile è la scelta dell'attributo migliore da mettere nell'albero.

- Aim: find a small tree consistent with the training examples
- Idea: (recursively) choose "most significant" attribute as root of (sub)tree

1. A ← "best" decision attribute for next node
2. Assign A as decision attribute for node
3. For each value of A create new descendant
4. Sort training examples to leaf node according to the attribute value of the branch
5. If all training examples are perfectly classified (same value of target attribute) stop, else iterate over new leaf nodes

Qual è l'attributo migliore?

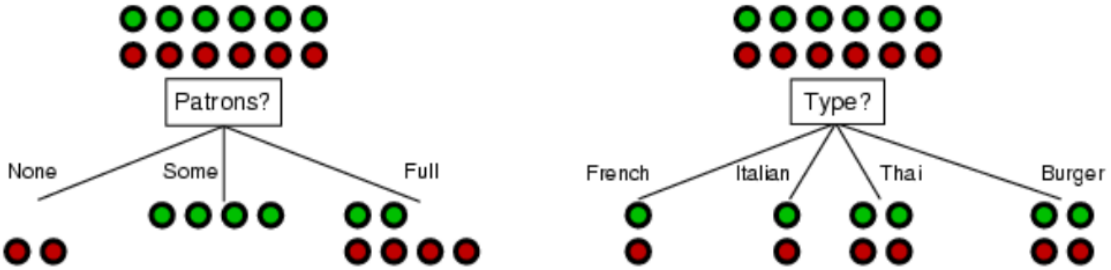
Sceglieremo l'attributo che **separa meglio le istanze**.



Per esempio se abbiamo un training set dove abbiamo 29 istanze positive e 35 negative. Mi pongo la domanda se è meglio identificare prima l'attributo A_1 o A_2. Il migliore sarà quello che **separa meglio** le istanze.

Guardando gli esempi scopro che quando A_1 è true ho un totale di 26 esempi dei 64 totali, nel false invece trovo i rimanenti 28. Quando A_2 è true invece ho 51 esempi e

quando è false 13. Quindi A_1 ha diviso le istanze meglio perché i due rami hanno proporzioni di esempi più vicini rispetto ad A_2.



Sarebbe anche meglio separare gli esempi positivi da quelli negativi, quindi in questo caso “Patrons” è un attributo migliore perché ha **discriminato** meglio. Questa è una divisione migliore rispetto a quella precedente basata soltanto da dalla proporzione della divisione.

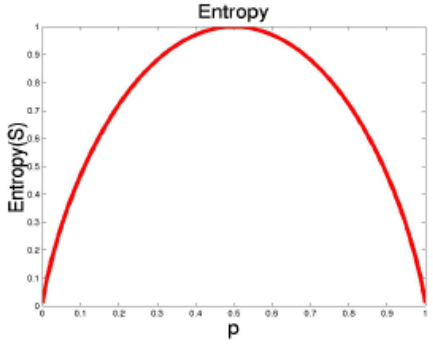
Formule per la separazione

Il contenuto separativo (entropia) di un insieme è misurabile.

- Training set S, possible values v_i, i = 1…n
- Def. Information Content (Entropy) of S:

  $I(P(v_1), \dots, P(v_n)) = \Sigma_{i=1} -P(v_i) \log_2 P(v_i)$
- In the boolean case, with S containing *p* positive examples and *n* negative examples:

$$I(\frac{p}{p+n}, \frac{n}{p+n}) = -\frac{p}{p+n}\log_2\frac{p}{p+n} - \frac{n}{p+n}\log_2\frac{n}{p+n}$$



- S is a set of training examples
- p_+ is the proportion of positive examples
- p_- is the proportion of negative examples
- Entropy measures the impurity of S

  $Entropy(S) = -p_+ \log_2 p_+ - p_- \log_2 p_-$

L'entropia misura quindi l'impurità del training set.

Seconda formula:

- Entropy(S) = expected number of bits needed to encode class (+ or -) of randomly drawn members of S (under the optimal, shortest length-code)
- Information theory optimal length code assigns $-\log_2 p$ bits to messages having probability p
- So the expected number of bits to encode (+ or -) of random member of S:

$$-p_+ \log_2 p_+ - p_- \log_2 p_-$$

Information Gain

- A chosen attribute A, with v distinct values, divides the training set E into subsets $E_1, \ldots, E_v$ according to their values.

$$remainder(A) = \sum_{i=1}^{v} \frac{p_i+n_i}{p+n} I\left(\frac{p_i}{p_i+n_i}, \frac{n_i}{p_i+n_i}\right)$$

- Information Gain (IG) or expected reduction in entropy due to sorting S on attribute A

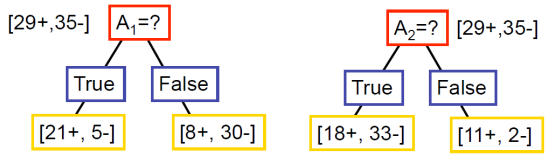
$$IG(A) = I\left(\frac{p}{p+n}, \frac{n}{p+n}\right) - remainder(A)$$

- Choose the attribute with the largest IG

Per l'ID3 useremo l'attributo con l'**information gain maggiore**.

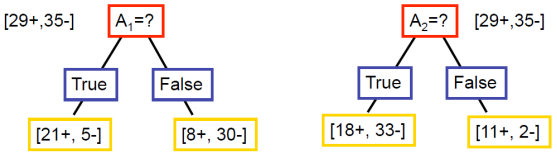
$IG(S,A) = Entropy(S) - \sum_{v \in values(A)} |S_v|/|S| \; Entropy(S_v)$

$Entropy([29+,35-]) = -29/64 \log_2 29/64 - 35/64 \log_2 35/64 = 0.99$



Entropy([21+,5-]) = 0.71
Entropy([8+,30-]) = 0.74
IG(S,A1)=Entropy(S)
-26/64*Entropy([21+,5-])
-38/64*Entropy([8+,30-])
=0.27

Entropy([18+,33-]) = 0.94
Entropy([11+,2-]) = 0.62
IG(S,A2)=Entropy(S)
-51/64*Entropy([18+,33-])
-13/64*Entropy([11+,2-])
=0.12



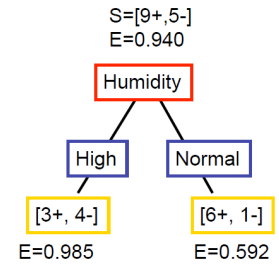
Quindi calcolo l'entropia del totale degli esempi, cioè [29+,35-]. Poi calcolo le entropie degli esempi di ciascun ramo. Inserisco questi dati nella formula dell'information gain che prende l'entropia totale e ci sottrare la somma pesata delle entropie dei rami (pesata in base a quanti casi vengono considerati in ciascun ramo)

Più i nuovi insiemi (sotto i rami) si sbilanciano, più il remainder sarà piccolo e la scelta sarà migliore.

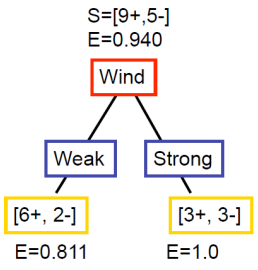
Un altro esempio

Day	Outlook	Temp.	Humidity	Wind	Play Tennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Weak	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cold	Normal	Weak	Yes
D10	Rain	Mild	Normal	Strong	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

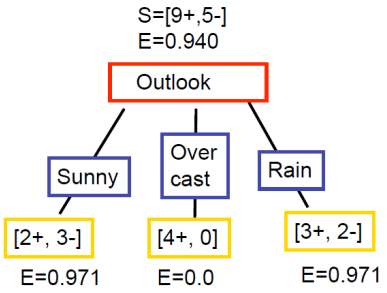
Per ogni attributo vado a calcolare l'entropia basata sul numero di esempi positivi e negativi nel subset. Vado quindi a calcolare l'entropia iniziale, e uso la formula dell'information gain, pesando le entropie degli attributi.



IG(S,Humidity)
=0.940-(7/14)*0.985
– (7/14)*0.592
=0.151



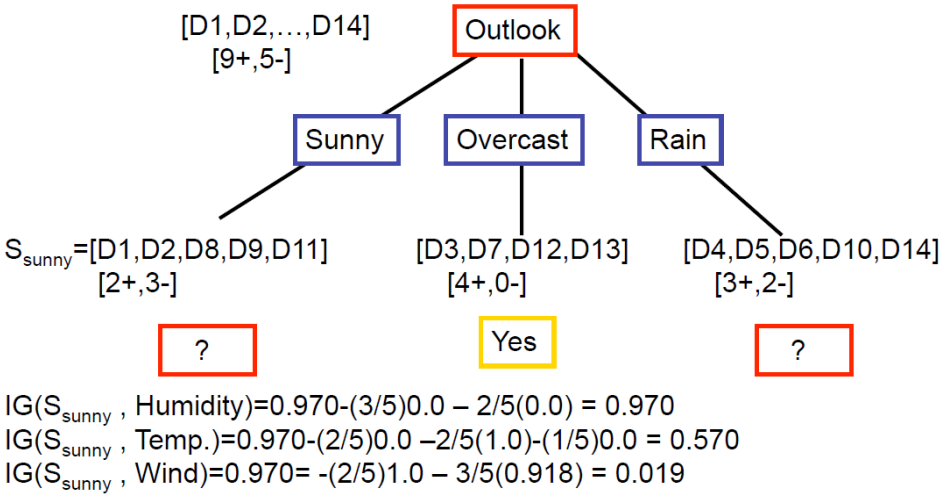
IG(S,Wind)
=0.940-(8/14)*0.811
– (6/14)*1.0
=0.048



IG(S,Outlook)
=0.940-(5/14)*0.971
-(4/14)*0.0 – (5/14)*0.0971
=0.247

L'attributo migliore è quindi Outlook perché ha l'information gain più alto.

Continuo quindi con le possibili scelte di outlook e vado a contare come vengono divisi gli esempi positivi e negativi in ciascun caso.



Continuo a vedere gli attributi dei sotto alberi a destra e a sinistra dove però non potrò trovare outlook di nuovo perché l'ho già visto.

Osservazioni

A differenza di find-s qui l'insieme delle ipotesi è completo. Qualunque formula booleana è rappresentabile tramite albero di decisioni. L'output sarà una singola ipotesi e non sicuramente sarà la migliore perché non faccio backtracking ma faccio scelte locali greedy.

Questo algoritmo può funzionare anche se ci sono degli errori nel dataset iniziale, è quindi robusto a "noisy data".

L'algoritmo ha un bias perché preferisce creare un albero corto mettendo gli attributi che discriminano meglio per primi. Find-s aveva un bias forte perché escludeva delle opzioni, ID3 ha un bias ma è di preferenza e quindi meno grave.

Overfit

Può comunque capitare overfitting, perché l'algoritmo si basa sul dataset iniziale, non sappiamo come si comporta con il resto dei dati possibili. L'overfitting è dato dalla generazione di più istanze da mettere nel training set. Una singola ipotesi tra quelle di esempio potrebbe fare overfitting se esiste una seconda ipotesi che ha un errore maggiore nel training set, ma minore nella totalità dei dati possibili.

Come faccio a capire se ho overfitting?

Una parte del dataset sarà tenuta da parte per rappresentare le "future istanze" e con quelle istanze andrò a verificare la performance del sistema che ho allenato. Quindi il ragionamento di prima va bene, ma usando questo set per validare e non l'interezza delle ipotesi possibili.

Posso anche usare questi esempi che ho messo da parte durante il training per aggiustare il processo, controllando la performance dell'allenamento, per vedere se ho preso le scelte giuste.

Il set che uso per l'allenamento lo chiamo "**test set**", mentre quello che metto da parte per la validazione è chiamato "**validation set**". La proporzione potrebbe essere 90%-10%, 80%-20% …

Per avere risultati più stabili e usare tutti i dati disponibili nel training, un data set può essere ripetutamente diviso in multipli training e validation sets applicando la "**cross-validation**". Per esempio dividendo 80%-10%-10%, usando l'80% per l'addestramento, e producendo due modelli diversi, uno validato con il primo validation set e uno con il secondo.