dello spazio delle ipotesi per il nuovo linguaggio e lo si confronti con lo spazio originario e con uno spazio senza bias, assumendo di avere F features discrete con $V > 2$ valori ognuno.

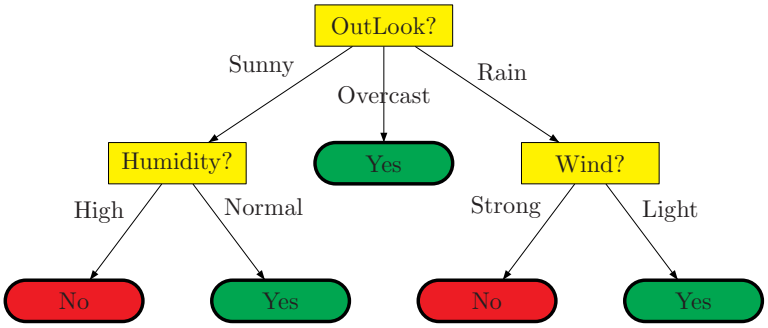
3.2 Decision-Tree Learning

L'apprendimento tramite alberi di decisione forse la tecnica pi utilizzata nell'ambito del data mining. Un albero di decisione rappresenta una ipotesi disgiuntiva per la classificazione di un concetto discreto. E' naturale ed intuitivo che una ipotesi possa essere strutturata come una sequenza di domande, in cui la domanda successiva dipende dalle risposte alle domande precedenti. Tale sequenza di domande pu essere strutturata in un albero diretto, dove per convenzione il primo nodo (la *radice*) visualizzata in testa, ed connessa tramite rami agli altri nodi. Ogni nodo connesso ad altri nodi, fino ai nodo *foglia*, che non collegata a nessun nodo. In un *albero di decisione*, i nodi interni dell'albero sono nodi di *scelta*, mentre le radici sono nodi di *decisione*. In altre parole, i nodi di scelta caratterizzano un concetto, mentre i nodi di decisione esprimono lo stesso.

Consideriamo ad esempio il seguente (classico) dataset, che descrive 14 giorni di osservazioni su condizioni atmosferiche che hanno permesso/impedito di giocare una partita a Tennis:

Day	*Outlook*	*Temperature*	*Humidity*	*Wind*	*PlayTennis*
1	Sunny	Hot	High	Light	No
2	Sunny	Hot	High	Strong	No
3	Overcast	Hot	High	Light	Yes
4	Rain	Mild	High	Light	Yes
5	Rain	Cool	Normal	Light	Yes
6	Rain	Cool	Normal	Strong	No
7	Overcast	Cool	Normal	Strong	Yes
8	Sunny	Mild	High	Light	No
9	Sunny	Cool	Normal	Light	Yes
10	Rain	Mild	Normal	Light	Yes
11	Sunny	Mild	Normal	Strong	Yes
12	Overcast	Mild	High	Strong	Yes
13	Overcast	Hot	Normal	Light	Yes
14	Rain	Mild	High	Strong	No

Supponiamo che il concetto da apprendere sia *PlayTennis*: ovvero, si vuole capire quando sia opportuno, in base alle condizioni del tempo, giocare una partita di Tennis. Un albero di decisione per tale concetto il seguente:



La classificazione di un'istanza utilizzando un albero di decisione effettuata attraversando l'albero, fino ad arrivare ad un nodo di decisione. Si parte dalla radice dell'albero valutando una specifica propriet dell'istanza. In base al responso di tale valutazione, si passa ad un nodo *discendente*, seguendo il ramo compatibile con il responso ottenuto. I vari rami corrispondono a differenti valori possibili per la valutazione. Quando il nodo successivo identificato, si passa su tale nodo e si re-itera la procedura ricorsivamente sul sottoalbero la cui radice il nodo in analisi. Se il nodo in questione un nodo foglia, allora la procedura termina e il responso che viene restituito il concetto associato alla foglia. Nell'albero di cui sopra, all'istanza $\langle Sunny, Mild, High, Strong\rangle$ viene associato il concetto *No* come segue. Si comincia valutando il valore dell'attributo *OutLook* alla radice dell'albero. Poich il responso *Sunny*, si passa al nodo contenente l'attributo *Humidity*. La valutazione di *Humidity* sull'istanza ha esito $High$: di conseguenza, si passa sul nodo pi a sinistra dell'albero. Poich tale nodo un nodo foglia, si etichetta l'istanza il concetto associato.

L'aspetto interessante degli alberi di decisione il loro potere espressivo: infatti, il linguaggio degli alberi di decisione permette di esprimere qualsiasi concetto appartenente all'insieme X di tutte le possibili istanze.

3.2.1 Apprendimento

L'algoritmo di apprendimento che ci interessa definire essenzialmente quello che ci permette di generare un albero compatto che sia consistente con il training set D. La compattezza d maggiori garanzie di generalit, in base al ben noto principio del *rasorio di Occam*. C' una spiegazione statistica in ci: le ipotesi pi compatte sono meno frequenti nello spazio delle ipotesi, e quindi meno probabile che rappresentino una coincidenza. Per contro, le ipotesi pi ridondanti sono molto frequenti, per cui pi difficile sceglierne una tra esse.

Purtroppo, trovare un albero minimale che sia consistente con il training set un problema NP-hard. Questo ci costringe a cercare dei metodi euristici che possano risolvere in maniera approssimata il problema descritto.

Lo schema generale di algoritmo di decision-tree learning che andremo ad esaminare evidenziato in figura **??**. Si tratta essenzialmente di un algoritmo che genera l'albero in maniera top-down, partendo dalla radice e fino alle foglie. Ad ogni nodo, si esamina una porzione dei dati e si sviluppa un sotto-albero. i

```
Function Build_DT(D, Attrs);
 1    if tutte le istanze di D hanno la stessa etichetta c
 2      return LEAF(c)
 3    else
 4      if Attrs = ∅
 5        return LEAF(majority(D))
 6      else
 7        Crea un nuovo nodo N con il "migliore" attributo A ∈ Attrs come radice;
 8        for each v ∈ dom(A)
 9          Crea una diramazione da N con etichetta A = v;
10          if {x ∈ D|x.A = v} = ∅
11            return LEAF(majority(D))
12          else
13            return Build_DT({x ∈ D|x.A = v}, Attrs − {A});
```

Figura 3.1: Algoritmo di apprendimento

vari sottoalberi sviluppati dalle partizioni esaminate vengono quindi ricombinate sulla radice. Come si pu notare, la strategia adottata *divide-et-impera*.

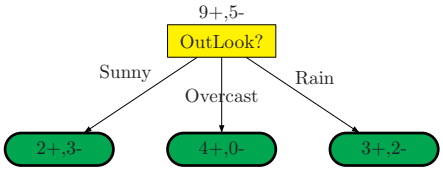
Nel cercare di raggiungere il requisito della minimalit, l'algoritmo adotta un approccio greedy, che comunque non garantisce che il risultato finale sia l'albero migliore possibile. Al passo 7, l'algoritmo sceglie l'attributo pi promettente: l'attributo, cio, che presumibilmente lo far giungere pi rapidamente degli altri a dei nodi foglia. Sostanzialmente, si assume di avere a disposizione una funzione H di valutazione della bont di un attributo. A questo punto si pu utilizzare H per scegliere l'attributo A che massimizza $H(A)$. Nella scelta di H, esistono varie possibilit in merito, che si rifanno a diversi aspetti che possono essere valutati, e riportano a diverse istanze dell'algoritmo di decision-tree learning. Nel seguito ne esaminiamo alcune:

- H misura il guadagno in informazione, ovvero quanto scegliendo un attributo piuttosto che un altro la purezza del nodo che si vuole costruire migliora;
- H misura il guadagno di varianza nei dati che supportano il nodo;
- H misura la correlazione tra l'attributo candidato e l'attributo che rappresenta il concetto da apprendere.

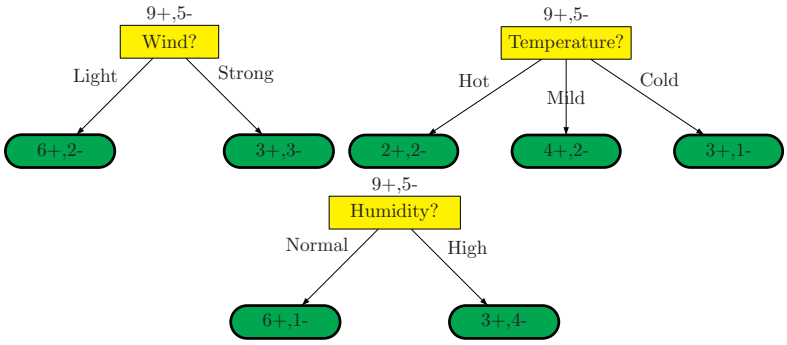
C4.5

Il primo caso che andiamo ad esaminare considera come migliore l'attributo che presenta la minore impurit nei dati che sottendono il nodo in esame. L'impurit rappresentata da un mescolamento di classi nei dati in esame; per contro, la purezza rappresentata dalla presenta di poche classi. Il problema : come misurare tale impurit? Consideriamo ad esempio il dataset *PlayTennis* descritto precedentemente. La costruzione di un albero di decisione a partire dalla radice ci porta a dover scegliere tra i quattro attributi disponibili. Se proviamo a graficare le distribuzioni dei dati nei quattro attributi, possiamo avere un'idea

del grado di purezza del nodo. Ad esempio, iniziare la costruzione dell'albero utilzzando l'attributo *OutLook* porta alla seguente situazione:



In tale figura, sopra il nodo radice evidenziata la distribuzione dei due concetti *Yes* (+) e *No* (-) nell'intero dataset. Nel nostro esempio, il dataset contiene 9 istanze classificate *Yes* e 5 istanze classificate *No*. Splittando il dataset in base ai valori di *OutLook*, otteniamo un partizionamento in tre datasets che evidenziano le distribuzioni mostrate in figura. Ad esempio, la partizione $\sigma_{OutLook=Sunny}$ contiene 2 esempi positivi e 3 negativi. Come si pu notare, rispetto alla situazione di partenza, lo split porta alla diminuzione di incertezza in almeno un sottoinsieme dei dati: quello relativo a $OutLook = OverCast$. Se proviamo a vedere gli split alternativi,



ci rendiamo facilmente conto che non facile confrontare in maniera qualitativa i risultati dei vari split. Tutti gli attributi infatti presentano un miglioramento in qualche partizione, e un peggioramento in altre partizioni. Qual' quindi una misura quantitativa che permette di scegliere tra le varie alternative. R. Quinlan ha proposto di utilizzare l'*entropia* come misura statistica della purezza di un dataset, e di misurare la bont di uno split con il miglioramento che si avrebbe in tale misura di entropia. Il concetto di entropia deriva direttamente dalla teoria dell'informazione, e rappresenta la misura del disordine relativo alla "trasmissione di informazione": se devo trasmettere informazione attraverso un canale, una misura da associare a tale trasmissione il numero di bit che mi servono per codificare l'informazione da trasmettere. In particolare, l'informazione da trasmettere a cui siamo interessati relativa alla classe da associare ad un arbitrario esempio di D. Supponiamo per semplicit di aver a che fare con due

classi a e b, che evidenziano le probabilit $\Pr(c(x) = a) = p_a$ e $\Pr(c(x) = b) = p_b$. Se $p_a = 0$, allora non ci sar bisogno di nessun bit per codificare l'informazione relativa all'esempio: in pratica, baster indicare con un numero costante di bit che l'intero dataset D composto da elementi con classe b. Si dice in tal caso che l'entropia di D 0. Analogamente, l'entropia 0 se $p_a = 1$. Supponiamo invece che $p_a = p_b = 0.5$. In tal caso, per ogni esempio c' bisogno di un bit che specifichi se l'esempio ha classe a o b. Consideriamo infine il caso in cui $p_a = 0.8$ e $p_b = 0.2$. In quest'ultimo caso, si pu pensare di associare una codifica pi compatta per gli esempi in classe a, e una codifica pi ridondante per gli esempi in classe b.

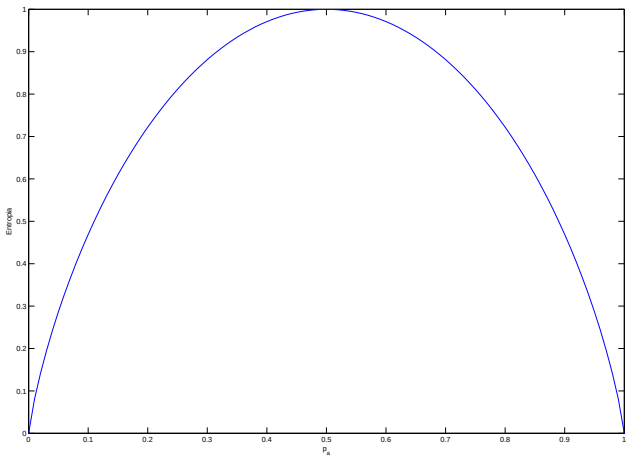
In generale, l'entropia pu essere misurata dalla funzione

$$E(D) = p_a \log \frac{1}{p_a} + p_b \log \frac{1}{p_b} = -p_a \log p_a - p_b \log p_b$$

Il termine $\log 1/p_a$ (e il suo duale $\log 1/p_b$) rappresenta il *contenuto informativo* della classe a (risp. della classe b). La ratio della formula $\log 1/p$ che le classi meno probabili hanno pi contenuto informativo delle classi pi probabili. Poich il contributo informativo di ogni esempio proporzionale al numero di istanze della stessa classe, otteniamo la formula che ci interessa:

$$\begin{aligned} IC(D) &= \sum_{x:c(x)=a} \log \frac{1}{p_a} + \sum_{x:c(x)=b} \log \frac{1}{p_b} \\ &= Np_a \log \frac{1}{p_a} + Np_b \log \frac{1}{p_b} \\ &\approx E(D) \end{aligned}$$

L'andamento grafico dell'entropia il seguente:



Come si pu notare, la funzione concava e assume i valori minimi ai due estremi, ovver quando $p_a = 0$ o $p_a = 1$. E' ovviamente possibile generalizzare il concetto di entropia a k classi (con probabilit $p_1, \ldots, p_k$):

$$E(D) = -\sum_{i_1}^{k} p_i log p_i$$

Nella formula, assumiamo la convenzione che, quando $p_i = 0$, allora $p_i \log p_i = 0$.

Se l'entropia rappresenta l'impurit presente in un dataset, la misura quantitativa che ci pu aiutare a catturare la nozione di "migliore attributo": il migliore attributo sar infatti quello che ci permetter di ottenere il pi alto *guadagno informativo* nello split. In altre parole, se uno split porta ad una riduzione dell'entropia originaria, allora l'attributo corrispondente pu essere considerato un buon candidato. Il miglior candidato sar quello che porter alla maggiore riduzione di entropia. La funzione $Gain(D, A)$ esprime il guadagno informativo di A rispetto a D: Se $v_1, \ldots v_m$ sono i possibili valori che A pu assumere, il guadagno informativo espresso da

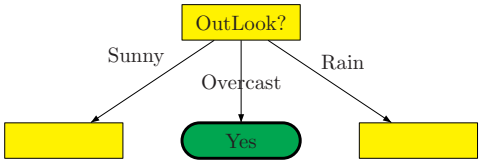
$$Gain(D, A) = E(D) - \sum_{i=1}^{m} \frac{n_{v_i}}{n} E(\sigma_{A=v_i}(D))$$

Proviamo ad applicare l'algoritmo *Build_DT* sul dataset *PlayTennis* utilizzando la funzione *Gain* per valutare l'attributo migliore ad ogni split. nel seguito rappresenteremo l'entropia di un dataset in maniera compatta esprimendola come funzione delle frequenze delle varie classi coinvolte. Ad esempio, $E(2, 3)$ rappresenta l'entropia di un dataset che ha un totale di 5 istanze, di cui 2 sono di classe *Yes* e 3 di classe *No*. Scegliamo, inizialmente, il nodo radice. Poich al nodo radice associato l'intero dataset, $E(9, 14) = -9/14 \log(9/14) - 5/14 \log(5/14) = 0.6429 \times 0.6374 + 0.3571 \times 1.4854 = 0.9402$. Per quel che riguarda gli attributi,

$$\begin{aligned}
Gain(PlayTennis, OutLook) &= 0.9402 - \frac{5}{14}E(2,3) - \frac{4}{14}E(4,0) - \frac{5}{14}E(3,2) \\
&= 0.9402 - \frac{5}{14} \times 0.9710 - 0 - \frac{5}{14} \times 0.9710 \\
&= 0.2647 \\
Gain(PlayTennis, Temperature) &= 0.9402 - \frac{4}{14}E(2,2) - \frac{6}{14}E(4,2) - \frac{4}{14}E(3,1) \\
&= 0.9402 - \frac{4}{14} - \frac{6}{14} \times 0.9183 - \frac{4}{14} \times 0.8113 \\
&= 0.0292 \\
Gain(PlayTennis, Wind) &= 0.9402 - \frac{8}{14}E(6,2) - \frac{6}{14}E(3,3) \\
&= 0.9402 - \frac{8}{14} \times 0.8113 - \frac{6}{14} \times 0.8113 \\
&= 0.0481
\end{aligned}$$

$$\begin{aligned} Gain(PlayTennis, Humidity) &= 0.9402 - \frac{7}{14}E(6,1) - \frac{7}{14}E(3,7) \\ &= 0.9402 - \frac{7}{14} \times 0.5917 - \frac{7}{14} \times 0.9852 \\ &= 0.1518 \end{aligned}$$

Risulta quindi che l'attributo con il miglior guadagno informativo l'attributo *OutLook*, che pu essere posto come attributo di split alla radice dell'albero.



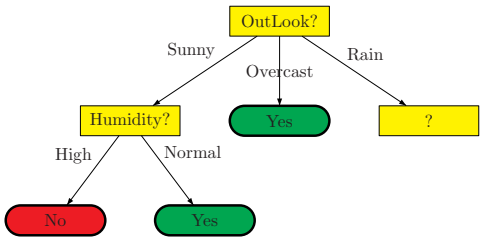
Si noti che, applicando ricorsivamente l'algoritmo, il nodo centrale diventa un nodo di decisione: infatti, tutte le istanze che ricadono in tale nodo sono della classe *Yes*. Proviamo invece a sviluppare il nodo pi a sinistra. La porzione dei dati che ci interessa quindi la seguente:

Day	*Outlook*	*Temperature*	*Humidity*	*Wind*	*PlayTennis*
1	Sunny	Hot	High	Light	No
2	Sunny	Hot	High	Strong	No
8	Sunny	Mild	High	Light	No
9	Sunny	Cool	Normal	Light	Yes
11	Sunny	Mild	Normal	Strong	Yes

Su tale porzione, possiamo valutare gli attributi restanti, a partire dalla situazione $E(2,3) = -2/5 \log 2/5 - 3/5 \log 3/5 = 0.4 \times 1.3219 + 0.6 \times 0.7370 = 0.9710$:

$$\begin{aligned} Gain([2,3], Temperature) &= 0.9710 - \frac{2}{5}E(2,0) - \frac{2}{5}E(1,1) - \frac{1}{5}E(0,1) \\ &= 0.9402 - 0 - \frac{2}{5} - 0 \\ &= 0.5710 \\ Gain([2,3], Wind) &= 0.9710 - \frac{3}{5}E(2,1) - \frac{2}{5}E(1,1) \\ &= 0.9402 - \frac{3}{5} \times 0.9183 - \frac{2}{5} \\ &= 0.02 \\ Gain([2,3], Humidity) &= 0.9402 - \frac{3}{5}E(0,3) - \frac{2}{5}E(2,0) \\ &= 0.9402 \end{aligned}$$

Il miglior attributo di split risulta quindi essere *Humidity*. L'albero sviluppato fino a questo punto risulta essere:



Entrambi i due nodi figli sono foglie, in quanto nodi puri. L'ultimo nodo che rimane da sviluppare il nodo pi a destra. Analizziamo i dati sottostanti:

Day	*Outlook*	*Temperature*	*Humidity*	*Wind*	*PlayTennis*
4	Rain	Mild	High	Light	Yes
5	Rain	Cool	Normal	Light	Yes
6	Rain	Cool	Normal	Strong	No
10	Rain	Mild	Normal	Light	Yes
14	Rain	Mild	High	Strong	No

Ancora una volta, valutiamo il guadagno relativo agli atributi rimanenti. Osservando che E(3,2) = 0.9710, abbiamo:

$$\begin{aligned}
Gain([3,2], Temperature) &= 0.9710 - \frac{3}{5}E(2,1) - \frac{2}{5}E(1,1) \\
&= 0.9402 - \frac{3}{5} \times 0.9183 - \frac{2}{5} \\
&= 0.02 \\
Gain([3,2], Wind) &= 0.9710 - \frac{3}{5}E(3,0) - \frac{2}{5}E(0,2) \\
&= 0.9402 \\
Gain([3,2], Humidity) &= 0.9402 - \frac{3}{5}E(1,1) - \frac{2}{5}E(2,1) \\
&= 0.9402 - \frac{3}{5} - \frac{2}{5} \times 0.9183 \\
&= 0.02
\end{aligned}$$

In questo caso, l'attributo *Wind* il pi appropriato. Il risultato a questo punto l'albero di decisione analizzato all'inizio della sezione.

CART

Un approccio alternativo a quello che misura il guadagno di entropia basato su considerazioni statistiche. Cos come nel caso precedente, assumiamo di voler apprendere un concetto binario $\{a, b\}$. Consideriamo cos un generico dataset D caratterizzato dalle probabilit a priori $\Pr(c(x) = a) = p_a$ e $\Pr(c(x) = b) = p_b$. Se associamo ad ogni $x_i \in D$ la variabile aleatoria X_i che pu assumere valore 1 se $c(x_i) = a$, e valore 0 se $c(x_i) = b$, possiamo osservare il seguente valor medio:

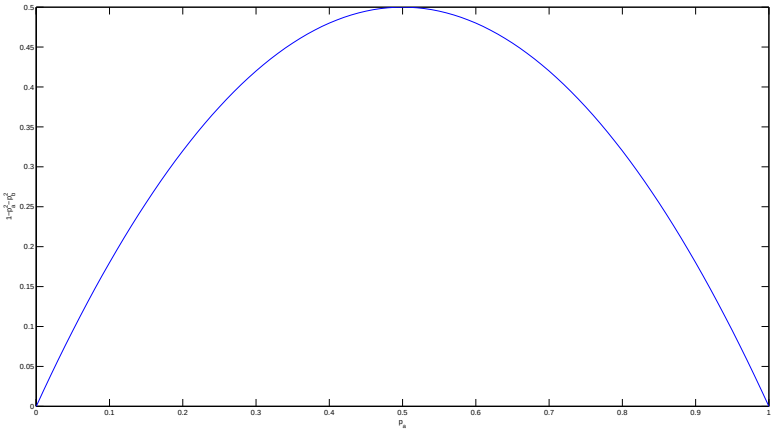
$$\overline{x} \quad = \quad 1/n \sum_i X_i = p_a$$

Infatti, se D contiene n_a istanze con etichetta a e n_b istanze con etichetta b (con $n_a + n_b = n$, il numero totale di istanze in D), avremo che $p_a = n_a/n$ e $p_b = n_b/n$. Inoltre, ci sono n_b casi in cui X_i vale 0, e n_a casi in cui X_i vale 1.

Se inotre calcoliamo la varianza campione, avremo:

$$\begin{aligned}
s &= 1/n \sum (X_i - \overline{x})^2 \\
&= \frac{n_a}{n}(1 - p_a)^2 + \frac{n_b}{n}(-p_a)^2 \\
&= p_a p_b^2 + p_b p_a^2 \\
&= p_a p_b (p_a + p_b) \\
&= p_a p_b \\
&= 1/2 p_a p_b + 1/2 p_a p_b \\
&= 1/2[p_a(1 - p_a) + p_a(1 - p_a)] \\
&= 1/2(p_a + p_b - p_a^2 - p_b^2) \\
&= 1/2(1 - p_a^2 - p_b^2)
\end{aligned}$$

Il valore $1 - p_a^2 - p_b^2$ chiamato *indice di Gini*, e rappresenta l'errore medio che si commette associando ad una generica istanza un'etichetta pescata in maniera random da D. E' chiaro che l'errore tanto pi alto quanto pi le classi sono equiprobabili. Per contro, se una delle due classi dominante, l'errore medio sar estremamente basso. La seguente figura descrive graficamente l'andamento dell'indice di Gini al variare dei valori di p_a.



Come si pu notare, l'andamento parabolico dell'indice lo rende estremamente adeguato ad essere utilizzato come alternativa alla misura di entropia.

La generalizzazione naturale del concetto sopra esposto la seguente. Si consideri un dataset D con unisieme k di classi, in cui la generica classe j esibisce la frequenza p_j. L'indice di Gini quindi definito come segue:

$$Gini(D) = 1 - \sum_{j=1}^{k} p_j^2$$

Tale indice pu essere utilizzato direttamente per misurare la diminuzione di varianza che si otterrebbe splittando un dataset D di dimensione n in m partizioni $D_1, \ldots, D_m$ (in cui ogni partizione D_i ha dimensione n_i):

$$Gini_{split}(D_1, \ldots D_k) = Gini(D) - \sum_{i=1}^{m} \frac{n_i}{n} Gini(D_i)$$

Il termine n_i/n all'interno della formula serve a pesare adeguatamente l'importanza di uno split in base al numero di istanze del dataset originario che contiene. Nel seguito, indicheremo con $Gini(A)$ il valore $Gini_{split}(D_1, \ldots, D_k)$ tale per cui la partizione $D_1, \ldots, D_k$ ottenuta da D splittando in base ai valori possibili di A.

Proviamo a vedere come si comporta l'algoritmo *Build_DT*, equipaggiato con l'indice di Gini, sui dati *PlayTennis*. Notiamo innanzitutto che, per l'intero dataset , abbiamo $Gini(PlayTennis) = 1 - (9/14)^2 - (5/14)^2 = 0.4592$. Utilizzando le formule di cui sopra, otteniamo

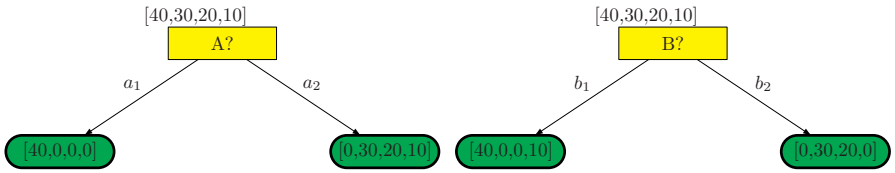
$$\begin{aligned} Gini(OutLook) \quad = \quad & 0.5492 - \frac{5}{14}\left[1 - \left(\frac{2}{5}\right)^2 - \left(\frac{3}{5}\right)^2\right] \\ & - \frac{4}{14}(1-1) \\ & - \frac{5}{14}\left[1 - \left(\frac{3}{5}\right)^2 - \left(\frac{2}{5}\right)^2\right] \end{aligned}$$

$$
\begin{aligned}
&= 0.4592 - 0.1714 - 0 - 0.1714 = 0.1163 \\
Gini(Temperature) &= 0.5492 - \frac{4}{14}\left[1 - \left(\frac{2}{4}\right)^2 - \left(\frac{2}{4}\right)^2\right] \\
&\quad - \frac{4}{14}\left[1 - \left(\frac{4}{6}\right)^2 - \left(\frac{2}{6}\right)^2\right] \\
&\quad - \frac{6}{14}\left[1 - \left(\frac{3}{4}\right)^2 - \left(\frac{1}{4}\right)^2\right] \\
&= 0.4592 - 0.1429 - 0.1905 - 1071 = 0.0187 \\
Gini(Humidity) &= 0.5492 - \frac{7}{14}\left[1 - \left(\frac{6}{7}\right)^2 - \left(\frac{1}{7}\right)^2\right] \\
&\quad - \frac{7}{14}\left[1 - \left(\frac{3}{7}\right)^2 - \left(\frac{4}{7}\right)^2\right] \\
&= 0.4592 - 0.1224 - 0.2449 = 0.0918 \\
Gini(Wind) &= 0.5492 - \frac{8}{14}\left[1 - \left(\frac{6}{8}\right)^2 - \left(\frac{2}{8}\right)^2\right] \\
&\quad - \frac{6}{14}\left[1 - \left(\frac{3}{6}\right)^2 - \left(\frac{3}{6}\right)^2\right] \\
&= 0.4592 - 0.2143 - 0.2143 = 0.0306
\end{aligned}
$$

In base ai calcoli, il miglior attributo risulta essere ancora *OutLook*, che viene quindi posto in cima all'albero.

L'istanza dell'algoritmo *Build_DT* equipaggiata dell'indice di Gini conosciuta in letteratura con il nome di *CART* (Classification and Regression Trees), ed stata introdotta in [**?**]. Nella sua versione originaria, l'algoritmo CART ammetteva solo split binari. Si noti a questo proposito che il potere espressivo di un albero binario non cambia rispetto a quello di un albero generico: infatti il fattore di branching di un albero di decisione pu essere sempre riportato a 2. In altri termini, dato un qualsiasi albero esiste sempre un albero binario equivalente, che implementa cio lo stesso concetto.

Il fatto comunque che CART scelga sempre alberi binari, combinato con l'indice di Gini, fa s che l'algoritmo tenda a scegliere attributi che presentano picchi nelle loro partizioni. Questa una differenza sostanziale rispetto all'entropia, esemplificata dalla seguente situazione: supponiamo di poter scegliere ad un certo punto di splittare il nodo corrente (che permette di poter scegliere tra quattro classi: a, b, c o d) in base a due attributi A o B. Le due alternative sono esemplificate come segue:



In pratica, se si sceglie B invece di A, le due partizioni tendono ad essere bilanciate. Il punto che l'indice di Gini tender a scegliere l'attributo A, mentre l'entropia tender a scegliere l'attributo B.

CHAID

L'ultimo caso che andiamo ad analizzare riguarda l'utilizzo di una statistica di correlazione gi utilizzata nel precedente capitolo. La variante di *Build_DT* che andiamo ad analizzare sceglie l'attributo su cui partizionale in base alla sua significativit rispetto all'attributo di categoria. Tale signifiativit misurata utilizzando il test di indipendenza del χ^2 sulla tabella di contingenza costruita a partire dall'attributo da analizzare e l'attributo di categoria. In pratica, considerando come esempio ricorrente il caso di una variabile binaria, su un attributo A si calcola la tabella

A/C	**a**	**b**	**Totale**
v_1	n_a^1	n_b^1	n^1
$\dots$	$\dots$	$\dots$	$\dots$
v_m	n_a^m	n_b^m	n^m
Totale	n_a	n_b	n

L'analisi di tale tabella ci permette di utilizzare il comune test di indipendenza utilizzato in statistica. Se i due attributi sono indipendenti, allora vale $\Pr(C = a|A = v_i) \approx \Pr(C = a) \times \Pr(A = v_i)$ e $\Pr(C = b|A = v_i) = \Pr(C = b) \times \Pr(A = v_i)$ per ogni i. Ad esempio, la seguente tabella

A/C	*a*	*b*	*Totale*
v_1	3	4	7
v_2	6	7	13
Totale	9	11	20

evidenzia un palese caso di indipendenza: $3 \approx 7 \times 9/20$, $4 \approx 7 \times 11/20$, $6 \approx 13 \times 9/20$ e $7 \approx 13 \times 11/20$. Viceversa, nella tabella

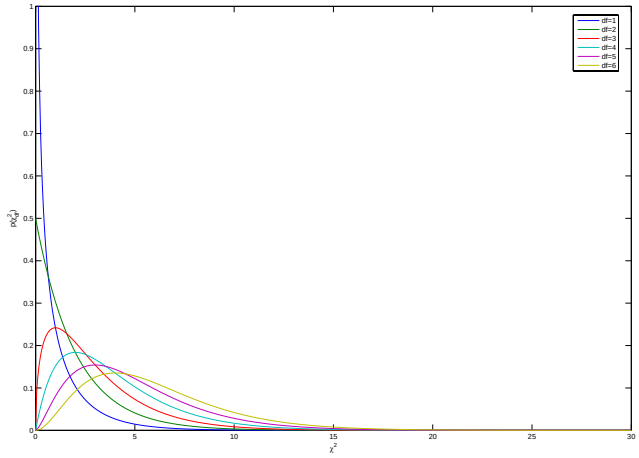
A/C	*a*	*b*	*Totale*
v_1	7	0	7
v_2	0	13	13
Totale	7	13	20

le approssimazioni auspicate non valgono. E' chiaro che una siffatta tabella estremamente interessante dal punto di vista predittivo, poich i due attributi sembrano molto correlati.

A partire dalla tabella di contingenza, sappiamo che il valore

$$\chi_0^2 = \sum_{j=1}^{m} \left[\frac{(n_a^j - E_a^j)^2}{E_a^j} + \frac{(n_b^j - E_b^j)^2}{E_b^j} \right]$$

, dove $E_a^j = n_a \times n_a^j/n$ e $E_b^j = n_b \times n_b^j/n$, segue la distribuzione χ^2_{m-1}, dove $m-1$ sono i gradi di libert.



Tale distribuzione ci dice molto sulla correlazione degli attributi: in pratica, quanto pi χ_0^2 piccolo, tanto pi probabile che gli attributi siano indipendenti. Viceversa, quanto pi χ_0^2 grande, tanto meno probabile che gli attributi siano indipendenti. La misura dell'indipendenza data dal valore di probabilit associata alχ_0^2: il p-value, che rappresenta la probabilit $\Pr(\chi^2_{m-1} > \chi_0^2)$, codifica l'indipendenza dei due attributi. Un p-value basso indica che l'associazione osservata tra l'attributo in analisi e l'attributo di categoria molto probabile. Se, viceversa. il p-value esibisce un valore alto, allora l'associazione improbabile. L'attributo che esibisce il p-value migliore pu essere scelto come attributo di split.

Si osservi che la scelta dell'attributo non pu essere effettuata guardando esclusivamente a χ_0^2: infatti, le tabelle di contingenza dei vari attributi non sempre hanno lo stesso numero di righe, e di conseguenza i vari χ_0^2 si rifanno a gradi di libert differenti.

Consideriamo l'esempio *PlayTennis*. le tabelle di contingenza sono: per *Humidity*,

Humidity/*PlayTennis*	*Yes*	*No*	*Totale*
High	3	4	7
Normal	6	1	7
Totale	9	5	14

che esibisce $\chi_1^2 = 2.5$ e conseguentemente $p = 0.0588$; per *Wind*

Wind/PlayTennis	*Yes*	*No*	*Totale*
Light	6	2	8
Strong	3	3	6
Totale	9	5	14

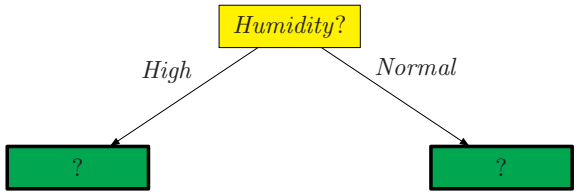
che esibisce $\chi_1^2 = 0.9$ e conseguentemente $p = 0.2590$; per *OutLook*

Wind/PlayTennis	*Yes*	*No*	*Totale*
Sunny	2	3	5
Overcast	4	0	4
Rain	3	2	5
Totale	9	5	14

che esibisce $\chi_2^2 = 3.5467$ e conseguentemente $p = 0.0849$; infine, per *Temperature*

Wind/PlayTennis	*Yes*	*No*	*Totale*
Hot	2	2	4
Mild	4	2	6
Cool	3	1	4
Totale	9	5	14

che esibisce $\chi_2^2 = 0.5704$ e conseguentemente $p = 0.3759$. Dalla precedente analisi si deduce che l'attributo pi correlato alla categoria *PlayTennis* l'attributo *Humidity*. Di conseguenza, l'albero si sviluppa a partire dall'attributo *Humidity* come mostrato:



Continuando a sviluppare l'albero, per il nodo di sinistra otteniamo il seguente insieme,

Day	*Outlook*	*Temperature*	*Humidity*	*Wind*	*PlayTennis*
1	Sunny	Hot	High	Light	No
2	Sunny	Hot	High	Strong	No
3	Overcast	Hot	High	Light	Yes
4	Rain	Mild	High	Light	Yes
8	Sunny	Mild	High	Light	No
12	Overcast	Mild	High	Strong	Yes
14	Rain	Mild	High	Strong	No

che esibisce i seguenti valori:

$$\begin{aligned} \textit{OutLook} &: \chi^2_2 = 4.8583, p = 0.0419 \\ \textit{Wind} &: \chi^2_1 = 0.2250, p = 0.4468 \\ \textit{Temperature} &: \chi^2_1 = 0.1944, p = 0.8209 \end{aligned}$$

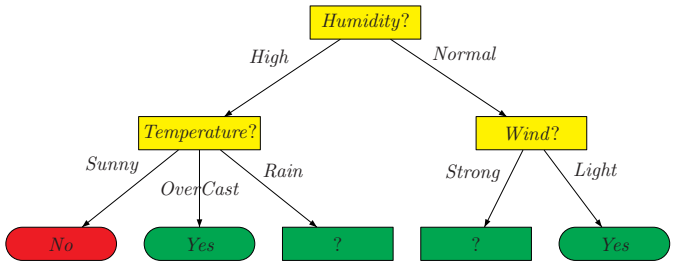
Per quello pi a sinistra otteniamo:

Day	*Outlook*	*Temperature*	*Humidity*	*Wind*	*PlayTennis*
5	Rain	Cool	Normal	Light	Yes
6	Rain	Cool	Normal	Strong	No
7	Overcast	Cool	Normal	Strong	Yes
9	Sunny	Cool	Normal	Light	Yes
10	Rain	Mild	Normal	Light	Yes
11	Sunny	Mild	Normal	Strong	Yes
13	Overcast	Hot	Normal	Light	Yes

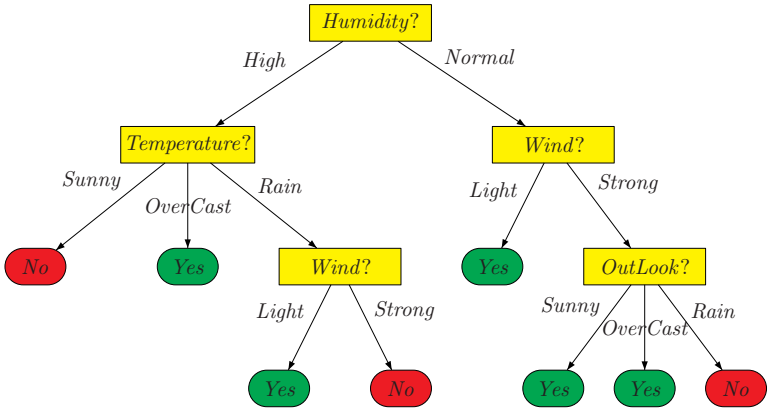
che esibisce:

$$\begin{aligned} \textit{OutLook} &: \chi^2_2 = 1.5556, p = 0.2297 \\ \textit{Wind} &: \chi^2_1 = 1.5556, p = 0.1470 \\ \textit{Temperature} &: \chi^2_2 = 0.8750, p = 0.3228 \end{aligned}$$

Di conseguenza, l'albero si sviluppa come segue:



Proseguendo lo sviluppo ai nodi rimanenti, facile verificare che la scelta ricade su *Temperature* per il primo nodo, e su *OutLook* per il secondo nodo:


Humidity?
High
Normal
Temperature?
Wind?
Sunny
OverCast
Rain
Light
Strong
No
Yes
Wind?
Yes
OutLook?
Light
Strong
Sunny
OverCast
Rain
Yes
No
Yes
Yes
No