Algoritmo di Albero di Decisione
Guida Weka

Croce Danilo

Web Mining e *Retrieval*
a.a. 2009-2010

April 28th, 2010

Apprendimento Macchina: Sommario

Esempio

Devi scrivere un programma che:

- dato un Livello Gerarchico di una compagnia
- dato un impiegato descritto attraverso alcuni *attributi* (il numero di attributi può essere molto alto)
- assegna all'impiegato il livello corretto nella gerarchia

Quanti *if* sono necessari per selezionare il corretto livello?
Quanto tempo è necessario per studiare le relazioni tra la gerarchia e gli attributi?

Soluzione

Impara la funzione per collegare ogni impiegato al corretto livello.

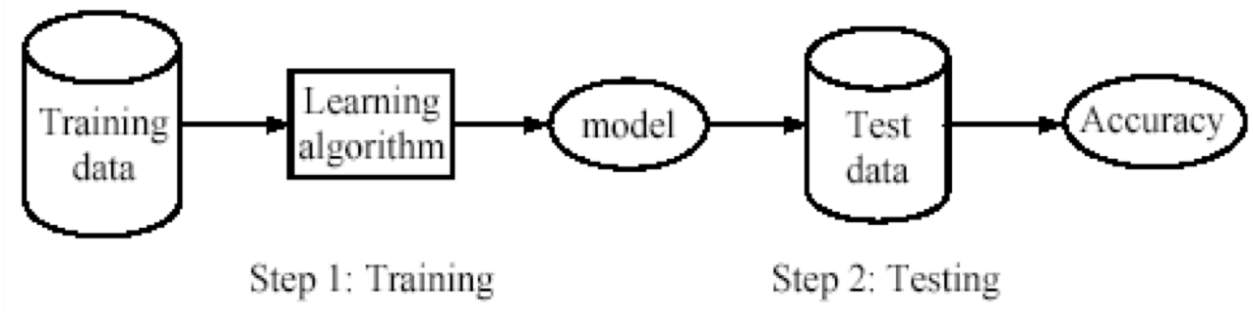
Processo di Apprendimento Supervisionato: due passi

Apprendimento (Addestramento)

Impara un *modello* usando i dati di training

Testing

Testa il modello usando dati di test non visti per valutare l'accuratezza del modello



Algoritmi di Apprendimento

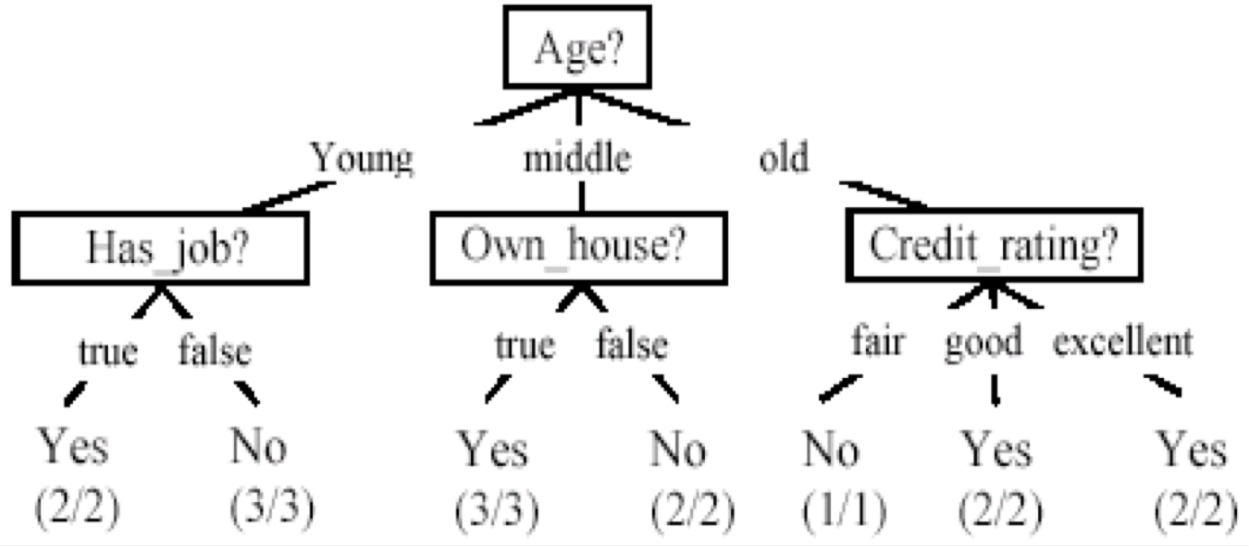
- Funzioni Probabilistiche (Classificatore Bayesiano)
- Funzioni per partizionare lo Spazio Vettoriale
  - Non-Lineari: KNN, Reti Neurali, ...
  - Lineari : Macchine a Supporto Vettoriale, Percettrone, ...
- Funzioni Booleane (Alberi di Decisione)

Albero di Decisione: Esempio di Dominio

La classe da imparare è: approva un prestito

ID	Age	Has_Job	Own_House	Credit_Rating	Class
1	young	false	false	fair	**No**
2	young	false	false	good	**No**
3	young	true	false	good	**Yes**
4	young	true	true	fair	**Yes**
5	young	false	false	fair	**No**
6	middle	false	false	fair	**No**
7	middle	false	false	good	**No**
8	middle	true	true	good	**Yes**
9	middle	false	true	excellent	**Yes**
10	middle	false	true	excellent	**Yes**
11	old	false	true	excellent	**Yes**
12	old	false	true	good	**Yes**
13	old	true	false	good	**Yes**
14	old	true	false	excellent	**Yes**
15	old	false	false	fair	**No**

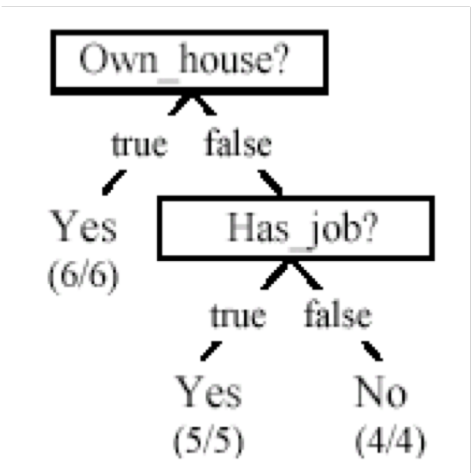
Albero di Decisione



Esempio di Albero di Decisione per il problema del prestito

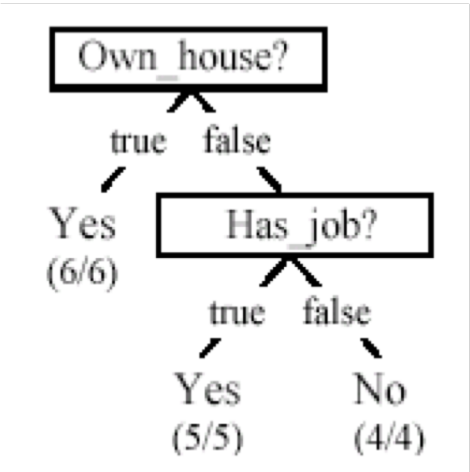
E' l'albero di decisione unico?

- No. Qui c'è un albero più semplice.
- Vogliamo alberi più piccoli e più accurati.
  - Facili da capire e si comportano meglio.



- Trovare il miglior albero è NP-hard.
- Tutte i correnti algoritmi di costruzione albero sono algoritmi euristici
- Un albero di decisione può essere convertito in un insieme di regole.

Da un albero di decisione ad un insieme di regole



Ogni percorso dalla radice alla foglia è una regola

Regole

Own_house = true $\rightarrow$ Class = yes
Own_house = false , Has_job = true $\rightarrow$ Class = yes
Own_house = false , Has_job = false $\rightarrow$ Class = no

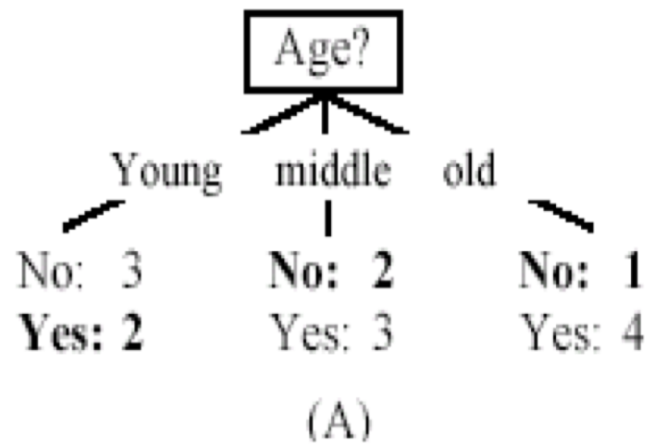
Scegli un attributo per partizionare i dati

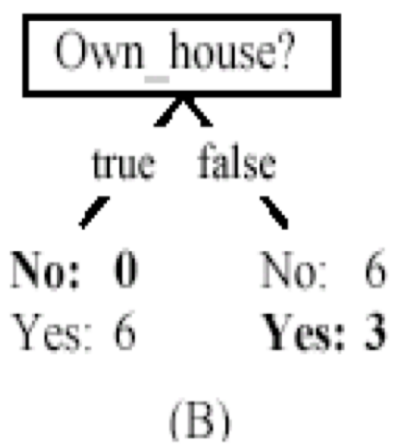
Come scegliere il migliore insieme di attributi?

L'obiettivo è di ridurre l'impurità o l'incertezza nei dati il più possibile

- Un sottoinsieme dei dati è *puro* se tutte le istanze appartengono alla stessa classe.

L'euristica è di scegliere l'attributo con il massimo *Recupero Dati* o *Percentuale di Recupero* basato sulla teoria dell'informazione.





Entropia di D

- L'entropia è una misura di incertezza associata ad una variabile random.
- Dato un insieme di esempi D è possibile computare l'entropia originale del dataset come:

$$H[D] = -\sum_{j=1}^{|C|} P(c_j) log_2 P(c_j)$$

dove C è l'insieme della classe desiderata.

Entropia

1. The data set D has 50% positive examples (Pr(*positive*) = 0.5) and 50% negative examples (Pr(*negative*) = 0.5).

$$entropy(D) = -0.5 \times \log_2 0.5 - 0.5 \times \log_2 0.5 = 1$$

2. The data set D has 20% positive examples (Pr(*positive*) = 0.2) and 80% negative examples (Pr(*negative*) = 0.8).

$$entropy(D) = -0.2 \times \log_2 0.2 - 0.8 \times \log_2 0.8 = 0.722$$

3. The data set D has 100% positive examples (Pr(*positive*) = 1) and no negative examples, (Pr(*negative*) = 0).

$$entropy(D) = -1 \times \log_2 1 - 0 \times \log_2 0 = 0$$

Come i dati diventano sempre più puri, il valore dell'entropia diventa sempre più piccolo.

Entropia di D

Dato un insieme di esempi D è possibile computare l'originale entropia del dataset come:

$$H[D] = -\sum_{j=1}^{|C|} P(c_j) log_2 P(c_j)$$

dove C è l'insieme della classe desiderata.

Entropia di un attributo A_i

Se creiamo l'attributo A_i, con v valori, la radice dell'albero corrente, questo partizionerà D in v sottoinsiemi $D_1, D_2, \ldots, D_v$ L'entropia attesa se A_i è usata come radice corrente:

$$H_{A_i}[D] = \sum_{j=1}^{v} \frac{|D_j|}{|D|} H[D_j]$$

Recupero Informazioni

Recupero Informazioni

L'informazione ottenuta selezionando l'attributo A_i per dividere o partizionare i dati è data dalla differenza dell'entropia *a priori* e l'entropia del percorso selezionato

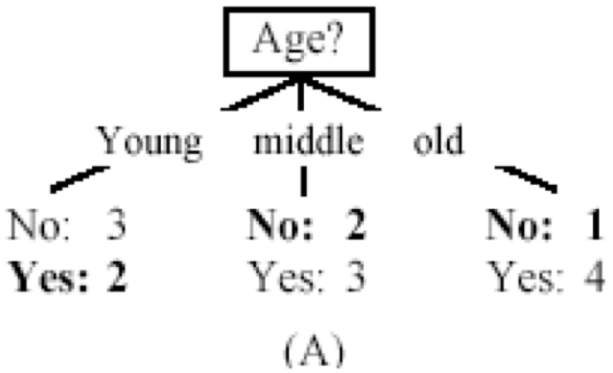
$$gain(D, A_i) = H[D] - H_{A_i}[D]$$

Scegliamo l'attributo con il *più alto recupero* per dividere/partizionare l'albero corrente.

Esempio

9 esempi appartengono alla categoria "YES" e 6 a "NO". Sfruttando la conoscenza a priori abbiamo:

$$H[D] = -\sum_{j=1}^{|C|} P(c_j) log_2 P(c_j)$$



$$H[D] = -\frac{6}{15} log_2 \frac{6}{15} - \frac{9}{15} log_2 \frac{9}{15} = 0.971$$

mentre partizionando attraverso la caratteristica Age:

$$H_{Age}[D] = -\frac{5}{15} H[D_1] - \frac{5}{15} H[D_2] - \frac{5}{15} H[D_3] = 0.888$$

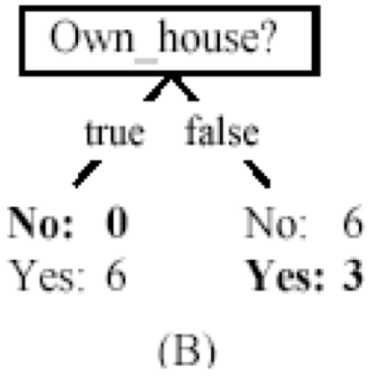
dove

$$H[D_1] = -\frac{3}{3+2} \cdot log_2(\frac{3}{3+2}) - \frac{2}{3+2} \cdot log_2(\frac{2}{3+2}) = 0.971$$

$$H[D_2] = -\frac{2}{2+3} \cdot log_2(\frac{2}{2+3}) - \frac{3}{2+3} \cdot log_2(\frac{3}{2+3}) = 0.971$$

$$H[D_3] = -\frac{1}{1+4} \cdot log_2(\frac{1}{1+4}) - \frac{4}{1+4} \cdot log_2(\frac{4}{1+4}) = 0.722$$

Esempio



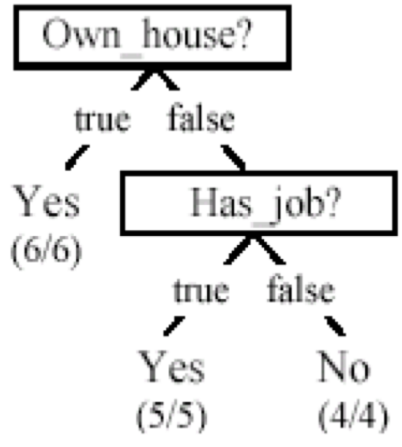
(B)

$$H[D] = -\frac{6}{15} log_2 \frac{6}{15} - \frac{9}{15} log_2 \frac{9}{15} = 0.971$$

$$H_{OH}[D] = -\frac{6}{15} H[D_1] - \frac{9}{15} H[D_2] =$$

$$-\frac{6}{15} \times 0 + \frac{9}{15} \times 0.918 = 0.551$$

$gain(D, Age) = 0.971 - 0.888 = 0.083$
$gain(D, Own\_House) = 0.971 - 0.551 = 0.420$
$gain(D, Has\_Job) = 0.971 - 0.647 = 0.324$
$gain(D, Credit) = 0.971 - 0.608 = 0.363$



Algoritmo per l'apprendimento dell'albero di decisione

Algoritmo Base (un algoritmo dividi-e-conquista greedy)

- Assume che gli attributi siano categoricamente bassi
- L'albero è costruito in maniera ricorsiva top-down
- All'inizio, tutti gli esempi di training sono alla radice
- Gli esempi sono partizionati ricorsivamente in base agli attributi selezionati
- Gli attributi sono selezionati sulla base di una funzione di impurità (es. recupero informazioni)

Condizioni per stoppare il partizionamento

- Tutti gli esempi per un dato nodo appartengono alla stessa classe
- Non ci sono attributi rimanenti per successivi partizionamenti? La classe maggioritaria è la foglia
- Non ci sono esempi rimasti

Algoritmo per l'apprendimento con gli alberi di decisione

. **Algorithm** decisionTree(D, A, T)

```
1   if D contains only training examples of the same class c_j ∈ C then
2       make T a leaf node labeled with class c_j;
3   elseif A = ∅ then
4       make T a leaf node labeled with c_j, which is the most frequent class in D
5   else  // D contains examples belonging to a mixture of classes. We select a single
6         // attribute to partition D into subsets so that each subset is purer
7       p_0 = impurityEval-1(D);
8       for each attribute A_i ∈ {A_1, A_2, ..., A_k} do
9           p_i = impurityEval-2(A_i, D)
10      end
11      Select A_g ∈ {A_1, A_2, ..., A_k} that gives the biggest impurity reduction,
            computed using p_0 − p_i;
12      if p_0 − p_g < threshold then    // A_g does not significantly reduce impurity p_0
13        make T a leaf node labeled with c_j, the most frequent class in D.
14      else                             // A_g is able to reduce impurity p_0
15          Make T a decision node on A_g;
16          Let the possible values of A_g be v_1, v_2, ..., v_m. Partition D into m
                disjoint subsets D_1, D_2, ..., D_m based on the m values of A_g.
17          for each D_j in {D_1, D_2, ..., D_m} do
18              if D_j ≠ ∅ then
19                  create a branch (edge) node T_j for v_j as a child node of T;
20                  decisionTree(D_j, A-{A_g}, T_j)// A_g is removed
21              end
22          end
23      end
24  end
```

Cos'è WEKA?

- Collezione di algoritmi ML - pacchetto Java open-source

  Sito:
  `http://www.cs.waikato.ac.nz/ml/weka/`
  Documentazione:
  `http://www.cs.waikato.ac.nz/ml/weka/index_documentation.html`
- Gli schemi per la classificazione includono:
  - alberi di decisione, memorizzatori di regole, naive Bayes, tabelle di decisione regressione pesata locale, SVM, memorizzatori basati sull'istanza, regressione logistica percettrone votato, percettrone multi-strato
- Per la classificazione, Weka permette divisione appresa/testata o validazione Cross-Fold
- Schemi per la divisione:
  - EM e Cobweb

Formato File ARFF

- Richiede dichiarazione di **@RELATION**, **@ATTRIBUTE** e **@DATA**
- **@RELATION** : questa dichiarazione associa un nome con il dataset
  - `@RELATION <relation-name>`
- **@ATTRIBUTE** : questa dichiarazione specifica il nome e tipo di un attributo
  - `@ATTRIBUTE <attribute-name> <datatype>`
  - Il tipo di dato può essere numerico, nominale, stringa o data

```
@ATTRIBUTE sepallength NUMERIC
@ATTRIBUTE petalwidth NUMERIC
@ATTRIBUTE class {Setosa,Versicolor,Virginica}
```

- **@DATA** : questa dichiarazione è una singola linea denotante l'inizio del segmento dati
  - I valori mancanti sono rappresentati da ?

```
@DATA
1.4, 0.2, Setosa
1.4, ?, Versicolor
```

ARFF Formato File Scarso

- Simile ai files AARF eccetto che i valori 0 dei dati non sono rappresentati
- Gli attributi non-zero sono specificati da numero attributi e valore
- Pieno:

```
@DATA
0 , X , 0 , Y , ”class A"
0 , 0 , W , 0 , ”class B"
```

- Scarso:

```
@DATA
{1 X, 3 Y, 4 ”class A"}
{2 W, 4 ”class B"}
```

- Nota che i valori omessi in una istanza sparsa sono 0, loro **non sono** valori mancanti! Se un valore è sconosciuto, devi esplicitamente rappresentarlo con un punto interrogativo (?)

Eseguire gli Schemi di Apprendimento

- `java -Xmx512m -cp weka.jar <learner class> [options]`
- Esempio di classi di imparatori:
  - Albero di Decisione: `weka.classifiers.trees.J48`
  - Naive Bayes: `weka.classifiers.bayes.NaiveBayes`
  - k-NN: `weka.classifiers.lazy.IBk`
- Importanti opzioni generiche:
  - `-t <training file>` Specifica il file di allenamento
  - `-T <test files>` Specifica il file di Test. Se nessun testing è eseguito sui dati di apprendimento
  - `-x <number of folds>` Numero di pieghe per la validazione incrociata
  - `-l <input file>` Usa modello salvato
  - `-d <output file>` Modello di Output su file
  - `-split-percentage <train size>` Grandezza dell'insieme di apprendimento
  - `-c <class index>` Indice di attributo per uso come classe (NB: l'indice inizia da 1)
  - `-p <attribute index>` Mette in output la predizione ed un attributo (0 per nessuno) per tutte le istanze di test.