Decision tree algorithm
Weka tutorial

Croce Danilo

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

April 28th, 2010

Machine Learning: brief summary

Example

You need to write a program that:

- given a Level Hierarchy of a company
- given an employe described trough some *attributes* (the number of attributes can be very high)
- assign to the employe the correct level into the hierarchy.

Machine Learning: brief summary

Example

You need to write a program that:

- given a Level Hierarchy of a company
- given an employe described trough some *attributes* (the number of attributes can be very high)
- assign to the employe the correct level into the hierarchy.

How many *if* are necessary to select the correct level?

Example

You need to write a program that:

- given a Level Hierarchy of a company
- given an employe described trough some *attributes* (the number of attributes can be very high)
- assign to the employe the correct level into the hierarchy.

How many *if* are necessary to select the correct level?
How many time is necessary to study the relations between the hierarchy and attributes?

Machine Learning: brief summary

Example

You need to write a program that:

- given a Level Hierarchy of a company
- given an employe described trough some *attributes* (the number of attributes can be very high)
- assign to the employe the correct level into the hierarchy.

How many *if* are necessary to select the correct level?
How many time is necessary to study the relations between the hierarchy and attributes?

Solution

Learn the *function* to link each employe to the correct level.

Supervised Learning process: two steps

Learning (Training)

Learn a *model* using the training data

Supervised Learning process: two steps

Learning (Training)

Learn a *model* using the training data

Testing

Test the model using unseen test data to assess the model accuracy

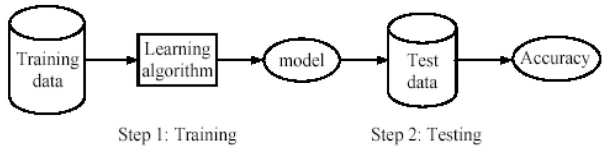
Supervised Learning process: two steps

Learning (Training)

Learn a *model* using the training data

Testing

Test the model using unseen test data to assess the model accuracy



Learning Algorithms

- Probabilistic Functions (Bayesian Classifier)
- Functions to partitioning Vector Space
  - Non-Linear: KNN, Neural Networks, ...
  - Linear: Support Vector Machines, Perceptron, ...

Learning Algorithms

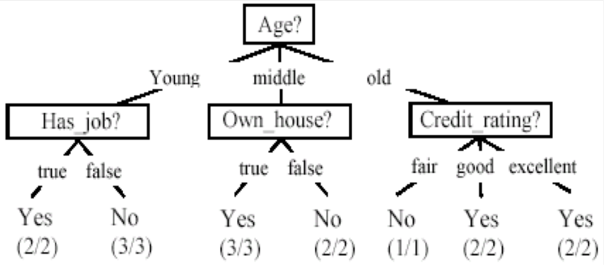
- Probabilistic Functions (Bayesian Classifier)
- Functions to partitioning Vector Space
  - Non-Linear: KNN, Neural Networks, ...
  - Linear: Support Vector Machines, Perceptron, ...
- Boolean Functions (Decision Trees)

Decision Tree: Domain Example

The class to learn is: approve a loan

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

Decision Tree



Decision Tree example for the loan problem

Is the decision tree unique?

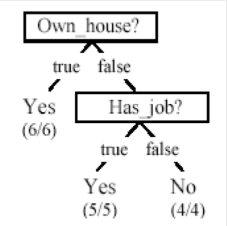
- No. Here is a simpler tree.

Is the decision tree unique?

- No. Here is a simpler tree.
- We want smaller tree and accurate tree.
    - Easy to understand and perform better.

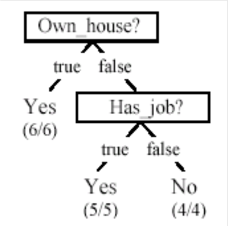
Is the decision tree unique?

- No. Here is a simpler tree.
- We want smaller tree and accurate tree.
  - Easy to understand and perform better.



Is the decision tree unique?

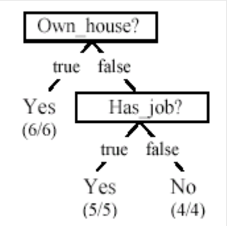
- No. Here is a simpler tree.
- We want smaller tree and accurate tree.
    - Easy to understand and perform better.



- Finding the best tree is NP-hard.

Is the decision tree unique?

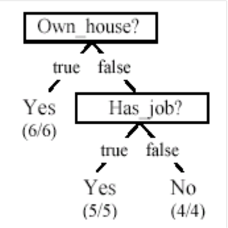
- No. Here is a simpler tree.
- We want smaller tree and accurate tree.
  - Easy to understand and perform better.



- Finding the best tree is NP-hard.
- All current tree building algorithms are heuristic algorithms

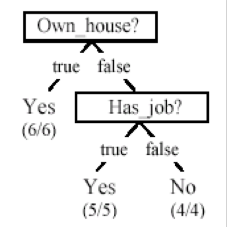
Is the decision tree unique?

- No. Here is a simpler tree.
- We want smaller tree and accurate tree.
    - Easy to understand and perform better.

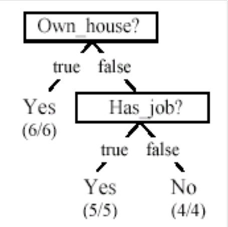


- Finding the best tree is NP-hard.
- All current tree building algorithms are heuristic algorithms
- A decision tree can be converted to a set of rules .

From a decision tree to a set of rules

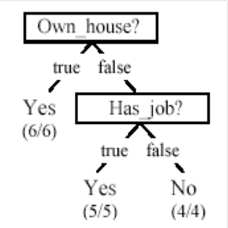


From a decision tree to a set of rules



Each path from the root to a leaf is a rule

From a decision tree to a set of rules



Each path from the root to a leaf is a rule

Rules

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

Choose an attribute to partition data

How chose the best attribute set?

Choose an attribute to partition data

How chose the best attribute set?

The objective is to reduce the impurity or uncertainty in data as much as possible

Choose an attribute to partition data

How chose the best attribute set?

The objective is to reduce the impurity or uncertainty in data as much as possible

- A subset of data is *pure* if all instances belong to the same class.

Choose an attribute to partition data

How chose the best attribute set?

The objective is to reduce the impurity or uncertainty in data as much as possible

- A subset of data is *pure* if all instances belong to the same class.

The heuristic is to choose the attribute with the maximum *Information Gain* or *Gain Ratio* based on information theory.

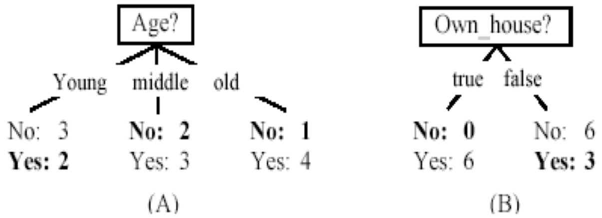
Choose an attribute to partition data

How chose the best attribute set?

The objective is to reduce the impurity or uncertainty in data as much as possible

- A subset of data is *pure* if all instances belong to the same class.

The heuristic is to choose the attribute with the maximum *Information Gain* or *Gain Ratio* based on information theory.



Information Gain

Entropy of D

- Entropy is a measure of the uncertainty associated with a random variable.
- Given a set of examples D is possible to compute the original entropy of the dataset such as:

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

where C is the set of desired class.

Entropy

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, ($\Pr(negative) = 0$).

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

As the data become purer and purer, the entropy value becomes smaller and smaller.

Information Gain

Entropy of D

Given a set of examples D is possible to compute the original entropy of the dataset such as:

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

where C is the set of desired class.

Entropy of an attribute A_i

If we make attribute A_i, with v values, the root of the current tree, this will partition D into v subsets $D_1, D_2, \ldots, D_v$. The expected entropy if A_i is used as the current root:

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

Information Gain

Information Gain

Information gained by selecting attribute A_i to branch or to partition the data is given by the difference of *prior* entropy and the entropy of selected branch

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

Information Gain

Information Gain

Information gained by selecting attribute A_i to branch or to partition the data is given by the difference of *prior* entropy and the entropy of selected branch

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

We choose the attribute with the *highest gain* to branch/split the current tree.

Example

9 examples belong to "YES" category and 6 to "NO". Exploiting prior knowledge we have:

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

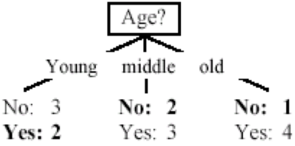
$$H[D] = -\frac{6}{15} log_2 \frac{6}{15} - \frac{9}{15} log_2 \frac{9}{15} = 0.971$$

Example

9 examples belong to "YES" category and 6 to "NO". Exploiting prior knowledge we have:

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

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



(A)

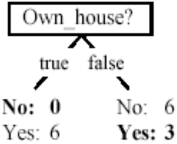
while partitioning through the *Age* feature:

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

where

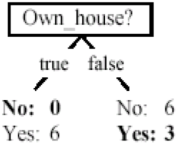
$$H[D_1] = -\frac{3}{3+2} \cdot log_2(\frac{3}{3+2}) - \frac{2}{3+2} \cdot log_2(\frac{2}{3+2}) = 0.971$$
$$H[D_2] = -\frac{2}{2+3} \cdot log_2(\frac{2}{2+3}) - \frac{3}{2+3} \cdot log_2(\frac{3}{2+3}) = 0.971$$
$$H[D_3] = -\frac{1}{1+4} \cdot log_2(\frac{1}{1+4}) - \frac{4}{1+4} \cdot log_2(\frac{4}{1+4}) = 0.722$$

Example



(B)

Example



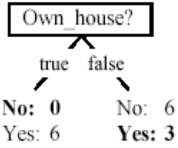
(B)

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

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

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

Example



(B)

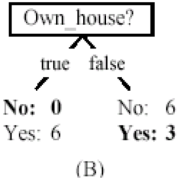
$$H[D] = -\frac{6}{15}log_2\frac{6}{15} - \frac{9}{15}log_2\frac{9}{15} = 0.971$$

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

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

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

Example

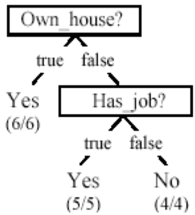


$$H[D] = -\frac{6}{15}log_2\frac{6}{15} - \frac{9}{15}log_2\frac{9}{15} = 0.971$$

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

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

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



Algorithm for decision tree learning

Basic algorithm (a greedy divide-and-conquer algorithm)

- Assume attributes are categorical now

Algorithm for decision tree learning

Basic algorithm (a greedy divide-and-conquer algorithm)

- Assume attributes are categorical now
- Tree is constructed in a top-down recursive manner

Algorithm for decision tree learning

Basic algorithm (a greedy divide-and-conquer algorithm)

- Assume attributes are categorical now
- Tree is constructed in a top-down recursive manner
- At start, all the training examples are at the root

Algorithm for decision tree learning

Basic algorithm (a greedy divide-and-conquer algorithm)

- Assume attributes are categorical now
- Tree is constructed in a top-down recursive manner
- At start, all the training examples are at the root
- Examples are partitioned recursively based on selected attributes

Algorithm for decision tree learning

Basic algorithm (a greedy divide-and-conquer algorithm)

- Assume attributes are categorical now
- Tree is constructed in a top-down recursive manner
- At start, all the training examples are at the root
- Examples are partitioned recursively based on selected attributes
- Attributes are selected on the basis of an impurity function (e.g., information gain)

Algorithm for decision tree learning

Basic algorithm (a greedy divide-and-conquer algorithm)

- Assume attributes are categorical now
- Tree is constructed in a top-down recursive manner
- At start, all the training examples are at the root
- Examples are partitioned recursively based on selected attributes
- Attributes are selected on the basis of an impurity function (e.g., information gain)

Conditions for stopping partitioning

- All examples for a given node belong to the same class

Algorithm for decision tree learning

Basic algorithm (a greedy divide-and-conquer algorithm)

- Assume attributes are categorical now
- Tree is constructed in a top-down recursive manner
- At start, all the training examples are at the root
- Examples are partitioned recursively based on selected attributes
- Attributes are selected on the basis of an impurity function (e.g., information gain)

Conditions for stopping partitioning

- All examples for a given node belong to the same class
- There are no remaining attributes for further partitioning ? majority class is the leaf

Algorithm for decision tree learning

Basic algorithm (a greedy divide-and-conquer algorithm)

- Assume attributes are categorical now
- Tree is constructed in a top-down recursive manner
- At start, all the training examples are at the root
- Examples are partitioned recursively based on selected attributes
- Attributes are selected on the basis of an impurity function (e.g., information gain)

Conditions for stopping partitioning

- All examples for a given node belong to the same class
- There are no remaining attributes for further partitioning ? majority class is the leaf
- There are no examples left

Algorithm for decision tree learning

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

    if D contains only training examples of the same class $c_j \in C$ **then**
        make T a leaf node labeled with class c_j;
    elseif $A = \emptyset$ **then**
        make T a leaf node labeled with c_j, which is the most frequent class in D
    else // D contains examples belonging to a mixture of classes. We select a single
            // attribute to partition D into subsets so that each subset is purer
        p_0 = impurityEval-1(D);
        for each attribute $A_i \in \{A_1, A_2, \ldots, A_k\}$ **do**
            p_i = impurityEval-2(A_i, D)
        end
        Select $A_g \in \{A_1, A_2, \ldots, A_k\}$ that gives the biggest impurity reduction, computed using $p_0 - p_i$;
        if $p_0 - p_g < threshold$ **then** // A_g does not significantly reduce impurity p_0
            make T a leaf node labeled with c_j, the most frequent class in D.
        else // A_g is able to reduce impurity p_0
            Make T a decision node on A_g;
            Let the possible values of A_g be $v_1, v_2, \ldots, v_m$. Partition D into m disjoint subsets $D_1, D_2, \ldots, D_m$ based on the m values of A_g.
            for each D_j in $\{D_1, D_2, \ldots, D_m\}$ **do**
                if $D_j \neq \emptyset$ **then**
                    create a branch (edge) node T_j for v_j as a child node of T;
                    decisionTree(D_j, A-$\{A_g\}$, T_j)// A_g is removed
                end
            end
        end
    end

What is WEKA?

What is WEKA?

- Collection of ML algorithms - open-source Java package

  Site:
  `http://www.cs.waikato.ac.nz/ml/weka/`
  Documentation:
  `http://www.cs.waikato.ac.nz/ml/weka/index_documentation.html`

What is WEKA?

- Collection of ML algorithms - open-source Java package

  Site:
  `http://www.cs.waikato.ac.nz/ml/weka/`
  Documentation:
  `http://www.cs.waikato.ac.nz/ml/weka/index_documentation.html`
- Schemes for classification include:
  - decision trees, rule learners, naive Bayes, decision tables, locally weighted regression, SVMs, instance-based learners, logistic regression, voted perceptrons, multi-layer perceptron

What is WEKA?

- Collection of ML algorithms - open-source Java package

  Site:
  `http://www.cs.waikato.ac.nz/ml/weka/`
  Documentation:
  `http://www.cs.waikato.ac.nz/ml/weka/index_documentation.html`
- Schemes for classification include:
  - decision trees, rule learners, naive Bayes, decision tables, locally weighted regression, SVMs, instance-based learners, logistic regression, voted perceptrons, multi-layer perceptron
- For classification, Weka allows train/test split or Cross-fold validation
- Schemes for clustering:
  - EM and Cobweb

ARFF File Format

- Require declarations of **@RELATION**, **@ATTRIBUTE** and **@DATA**

ARFF File Format

- Require declarations of **@RELATION**, **@ATTRIBUTE** and **@DATA**
- **@RELATION** declaration associates a name with the dataset
  - `@RELATION <relation-name>`

ARFF File Format

- Require declarations of **@RELATION**, **@ATTRIBUTE** and **@DATA**
- **@RELATION** declaration associates a name with the dataset
  - `@RELATION <relation-name>`
- **@ATTRIBUTE** declaration specifies the name and type of an attribute
  - `@ATTRIBUTE <attribute-name> <datatype>`
  - Datatype can be numeric, nominal, string or date

ARFF File Format

- Require declarations of **@RELATION**, **@ATTRIBUTE** and **@DATA**
- **@RELATION** declaration associates a name with the dataset
  - `@RELATION <relation-name>`
- **@ATTRIBUTE** declaration specifies the name and type of an attribute
  - `@ATTRIBUTE <attribute-name> <datatype>`
  - Datatype can be numeric, nominal, string or date

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

ARFF File Format

- Require declarations of **@RELATION**, **@ATTRIBUTE** and **@DATA**
- **@RELATION** declaration associates a name with the dataset
  - `@RELATION <relation-name>`
- **@ATTRIBUTE** declaration specifies the name and type of an attribute
  - `@ATTRIBUTE <attribute-name> <datatype>`
  - Datatype can be numeric, nominal, string or date

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

- **@DATA** declaration is a single line denoting the start of the data segment
  - Missing values are represented by ?

ARFF File Format

- Require declarations of **@RELATION**, **@ATTRIBUTE** and **@DATA**
- **@RELATION** declaration associates a name with the dataset
    - `@RELATION <relation-name>`
- **@ATTRIBUTE** declaration specifies the name and type of an attribute
    - `@ATTRIBUTE <attribute-name> <datatype>`
    - Datatype can be numeric, nominal, string or date

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

- **@DATA** declaration is a single line denoting the start of the data segment
    - Missing values are represented by ?

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

ARFF Sparse File Format

- Similar to AARF files except that data value 0 are not represented

ARFF Sparse File Format

- Similar to AARF files except that data value 0 are not represented
- Non-zero attributes are specified by attribute number and value

ARFF Sparse File Format

- Similar to AARF files except that data value 0 are not represented
- Non-zero attributes are specified by attribute number and value
- Full:

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

ARFF Sparse File Format

- Similar to AARF files except that data value 0 are not represented
- Non-zero attributes are specified by attribute number and value
- Full:

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

- Sparse:

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

ARFF Sparse File Format

- Similar to AARF files except that data value 0 are not represented
- Non-zero attributes are specified by attribute number and value
- Full:

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

- Sparse:

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

- Note that the omitted values in a sparse instance are 0, they **are not** *missing* values! If a value is unknown, you must explicitly represent it with a question mark (?)

Running Learning Schemes

- `java -Xmx512m -cp weka.jar <learner class> [options]`

Running Learning Schemes

- `java -Xmx512m -cp weka.jar <learner class> [options]`
- Example learner classes:
  - Decision Tree: `weka.classifiers.trees.J48`
  - Naive Bayes: `weka.classifiers.bayes.NaiveBayes`
  - k-NN: `weka.classifiers.lazy.IBk`

Running Learning Schemes

- `java -Xmx512m -cp weka.jar <learner class> [options]`
- Example learner classes:
  - Decision Tree: `weka.classifiers.trees.J48`
  - Naive Bayes: `weka.classifiers.bayes.NaiveBayes`
  - k-NN: `weka.classifiers.lazy.IBk`
- Important generic options:
  - `-t <training file>` Specify training file
  - `-T <test files>` Specify Test file. If none testing is performed on training data
  - `-x <number of folds>` Number of folds for cross-validation
  - `-l <input file>` Use saved model
  - `-d <output file>` Output model to file
  - `-split-percentage <train size>` Size of training set
  - `-c <class index>` Index of attribute to use as class (NB: the index start from 1)
  - `-p <attribute index>` Only output the predictions and one attribute (0 for none) for all test instances.