Chapter 10
Machine Learning

10.1 Introduction

Machine Learning[6][8][12] is concerned with the study of building computer programs that automatically improve and/or adapt their performance through experience. Machine learning can be thought of as “programming by example" [11]. Machine learning has many common things with other domains such as statistics and probability theory (understanding the phenomena that have generated the data), data mining (finding patterns in the data that are understandable by people) and cognitive sciences (human learning aspire to understand the mechanisms underlying the various learning behaviors exhibited by people such as concept learning, skill acquisition, strategy change, etc.) [1].

The goal of machine learning is to devise learning algorithms that do the learning automatically without human intervention or assistance. Rather than program the computer to solve the task directly, in machine learning, we seek methods by which the computer will come up with its own program based on examples that we provide [11].

Dietterich [1] mentioned 4 situations in which it is not easy for software engineers to design the software for solving a problem, but there are more similar situations:

- problems for which there exist no human experts. As an example, the need to predict machine failures before they occur in modern automated manufacturing facilities. This can be performed by analyzing sensor readings. There are no human experts who can be interviewed by a programmer to provide the knowledge necessary to build a computer system. A machine learning system can study recorded data and subsequent machine failures and learn prediction rules.
- problems where human experts exist, but where they are unable to explain their expertise. This in domains such as speech recognition, handwriting recognition, and natural language understanding. Experts cannot describe the detailed steps that they follow as they perform them. Humans can provide machines with examples of the inputs and correct

outputs for these tasks, so machine learning algorithms can learn to map the inputs to the outputs.

- dynamic problems where phenomena are changing rapidly. Many a times, people would like to be able predict the future behavior of certain phenomena such as the stock market, exchange rates or even weather forecast. These behaviors change frequently, so that even if a programmer could construct a good predictive computer program, it would need to be rewritten frequently. A learning program can relieve the programmer of this burden by constantly modifying and tuning a set of learned prediction rules.
- applications that need to be customized for each computer user separately. For instance, a program to filter unwanted electronic mail messages. Different users will need different filters. A machine learning system can learn which mail messages the user rejects and maintain the filtering rules automatically.

Some examples of machine learning problems are [3][4][5][9][10][11]:

- character (including digit) recognition
- handwriting recognition
- face detection
- spam filtering
- sound recognition
- spoken language understanding
- stock market prediction
- weather prediction
- medical diagnosis
- fraud detection
- fingerprint matching
- etc.

A computer program is said to *learn* 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* [2].

A learning system is characterized by the following elements:

- task (or tasks) T
- experience E
- performance measure P.

For example, a learning system for playing tic-tac-toe game (or nuggets and crosses) will have the following corresponding elements:

- T: Play tic-tac-toe;
- P: Percentage of games won (and eventually drawn);
- E: Playing against itself (can also be playing against others).

Thus, a generic learning system can be defined by the following components [2] (see Figure 10.1):

- *Goal:* Defined with respect to the task to be performed by the system;
- *Model:* A mathematical function which maps perception to actions;
- *Learning rules:* Used to update the model parameters with new experience in a way which optimizes the performance measures with respect to the goals. Learning rules help the algorithm search for the best model;
- *Experience:* A set of perception (and possibly the corresponding actions).

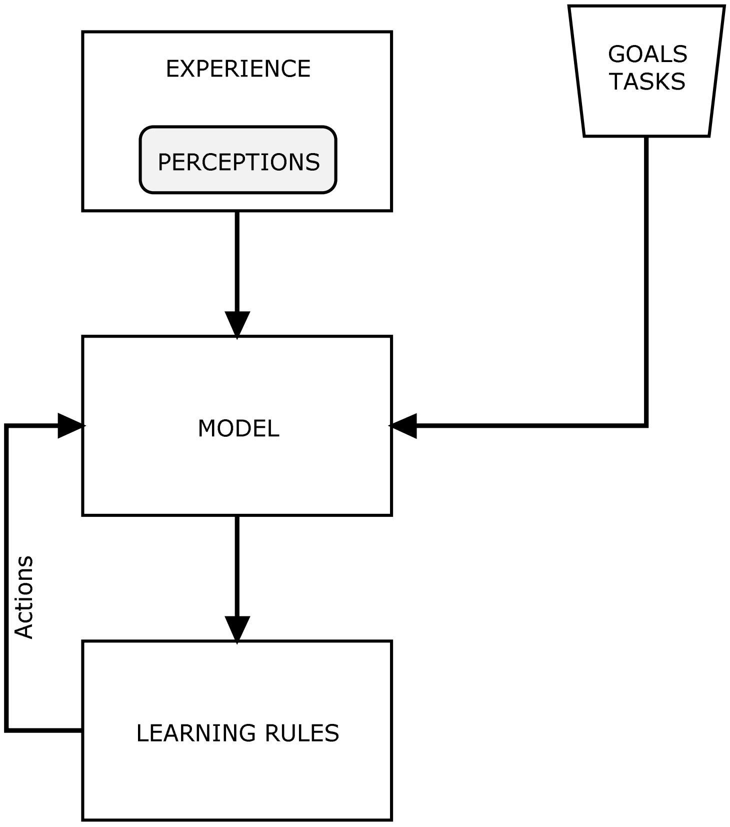

Fig. 10.1 Generic scheme of a learning system

10.2 Terminology

Before getting into the core of a learning system, we first need to define the basic notions [11].

An *example* (sometimes also called an *instance*) is the object that is being classified. For instance, if we consider the cancer tumor patients classification, then the patients are the examples.

An example is described by a set of *attributes*, also known as *features* or *variables*.

For instance, the cancer tumor patients classification, a patient might be described by attributes such as gender, age, weight, tumor size, tumor shape, blood pressure, etc.

The *label* is the category that we are trying to predict. For instance, in the cancer patient classification, the labels can be "avascular", "vascular" and "angiogenesis".

During training, the learning algorithm is supplied with *labeled examples*, while during testing, only *unlabeled examples* are provided.

In certain situation we can assume that only two labels are possible that we might as well call 0 and 1 (for instance, cancer or not cancer). This will make the things much simple.

We will also make the simplifying assumption that there is a mapping from examples to labels. This mapping is called a *concept*. Thus, a concept is a function of the form

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

where X is the space of all possible examples called the *domain* or *instance space*. A collection of concepts is called a *concept class*. We will often assume that the examples have been labeled by an unknown concept from a *known* concept class.

10.3 Learning Steps

The main steps in a learning process are as follows:

- *data and assumptions*
  Data refers to the data available for the learning task and assumptions represent what we can assume about the problem.
- *representation*
  We should define how to represent the examples to be classified. There are many representation methods for the same data. The choice of representation may determine whether the learning task is very easy or very difficult.
- *method and estimation*
  Method takes into account what are the possible hypotheses and estimation helps to adjust the predictions based on the feedback (such as updating the parameters when there is a mistake, etc).
- *evaluation*
  This evaluates how well the method is working (for instance, considering the ratio of wrong classified data and the whole dataset).

- *model selection*

  It tells us whether we can rethink the approach to do even better or to make it more flexible, or, whether we should choose an entirely different model that would be more suitable.

10.4 Learning Systems Classification

Learning systems can be classified based on their components.

10.4.1 Classification Based on Goal, Tasks, Target Function

Based on the goals of a learning system, we can have the following classification [2]:

- *Prediction*: the system predicts the desired output for a given input based on previous input/output pairs.

  Example: prediction of a stock value given other (input) parameters values like market index, interest rates, currency conversion, etc.
- *Regression*: the system estimates a function of many variables (multivariate) or single variable (univariate) from scattered data.

  Example: a simple univariate regression problem is $x^4+ x^3+ x^2+x+1$
- *Classification (categorization)*: the system classifies an object into one of several categories (or classes) based on features of the object.

  Example: A diagnosis system which has to classify a patient's cancer tumor into one of the three categories: avascular, vascular, angiogenesis.
- *Clustering*: the system task is to organize a group of objects into homogeneous segments.

  Example: a satellite image analysis system which groups land areas into forest, urban and water body, for better utilization of natural resources.
- *Planning*: the system has to generate an optimal sequence of actions to solve a particular problem.

  Example: robot path planning (to perform a certain task or to move from one place to another, etc).

Learning tasks can be classified (among others) in [1]:

- empirical learning and
- analytical learning.

Empirical learning is learning that relies on some form of external experience (the program cannot infer the rules of the game analytically - it must interact with a teacher to learn them), while analytical learning requires no external inputs (the program is able to improve its performance just by analyzing the problem).

10.4.2 Classification Based on the Model

The model is actually the algorithm used and there are several learning models:

- Decision trees
- Linear separators (perceptron model)
- Neural networks
- Genetic programming
- Evolutionary algorithms
- Graphical models
- Support vector machines
- Hidden Markov models

10.4.3 Classification Based on the Learning Rules

Learning rules are usually related with the model of learning used. Some common rules are:

- gradient descent
- least square error
- expectation maximization
- margin maximization.

10.4.4 Classification Based on Experience

The nature of experiences available varies with applications. Some common situations are [2] [7].

- *Supervised learning:*
  In supervised learning, the machine is given the desired outputs and its goal is to learn to produce the correct output given a new input.

  In supervised learning a teacher or oracle is available which provides the desired action corresponding to a perception. A set of perception action pair provides what is called a training set. Examples include an automated vehicle where a set of vision inputs and the corresponding steering actions are available to the learner.

- *Unsupervised learning*:
  In unsupervised learning the goal of the machine is to build a model of input that can be used for reasoning, decision making, predicting things, and communicating.

  In unsupervised learning no teacher is available. The learner only discovers persistent patterns in the data consisting of a collection of perceptions. This is also called exploratory learning. Finding out malicious network attacks from a sequence of anomalous data packets is an example of unsupervised learning.

- *Active learning:*
  Here not only a teacher is available, the learner has the freedom to ask the teacher for suitable perception-action example pairs which will help the learner to improve its performance. Consider a news recommender system which tries to learn a user's preference and categorize news articles as interesting or uninteresting to the user. The system may present a particular article (of which it is not sure) to the user and ask whether it is interesting or not.

- *Reinforcement learning*:
  In reinforcement learning, the machine can also produce actions which affect the state of the world, and receive rewards (or punishments). The goal is to learn to act in a way that maximizes rewards in the long term.

  In reinforcement learning a teacher is available, but the teacher instead of directly providing the desired action corresponding to a perception, return reward and punishment to the learner for its action corresponding to a perception. Examples include a robot in an unknown terrain where its get a punishment when its hits an obstacle and reward when it moves smoothly.

10.5 Machine Learning Example

Consider the data given in Table 10.1.

	INPUT						OUTPUT
Nr.	x_1	x_2	x_3	x_4	x_5	x_6	Y
1.	1	0	2	0	1	1	0
2.	0	2	0	2	3	0	1
3.	1	1	0	3	0	0	0
4.	0	0	3	1	0	1	0
5.	4	0	0	1	0	0	1
6.	1	1	2	3	0	0	1
7.	3	2	5	0	1	3	1
8.	2	1	0	0	2	3	1
9.	1	1	1	0	2	2	0
10.	1	1	2	1	2	1	0
11.	0	6	1	1	1	3	0
12.	1	0	0	2	0	0	1
13.	0	0	3	3	3	0	0
14.	0	3	1	1	1	0	1
15.	2	0	1	1	0	1	0
16	1	2	1	1	0	0	?

For this example we have:

- Space of all possible examples: X = (x_1, x_2, x_3, x_4, x_5, x_6)
- Instance space = |{X}| = 15
- Concept learning/regression: we have to find *f* such as $Y=f(X)$
- Prediction: $f(1, 2, 1, 1, 0, 0)$ =?
- Evaluation:

$$\text{Error}\ (f) = \frac{\left|\left\{X / f(X) \neq Y\right\}\right|}{15}$$

The following chapters will cover various topics of machine learning such as learning decision trees, neural network learning, statistical learning methods, evolutionary computation and reinforcement learning.

References

1. Dietterich, T.G.: Machine Learning. In: Nature Encyclopedia of Cognitive Science. Macmillan, London (2003)
2. `http://www.onlinefreeebooks.net/free-ebooks-computer-programming-technology/artificial-intelligence/artificial-intelligence-course-material-pdf.html` (accessed on February 10, 2011)
3. Jordan, M.: An introduction to Graphical Models, Center for Biological and Computational Learning, Massachusetts Institute of Technology (1997), `http://www.cis.upenn.edu/~mkearns/papers/barbados/jordan-tut.pdf` (accessed on February 10, 2011)
4. Duda, R., Hart, P., Stork, D.: Pattern Classification, 2nd edn. Wiley and Sons, New York (2001)
5. Bishop, C.M.: Neural Networks for Pattern Recognition. Oxford University Press, Oxford (1995)
6. Mitchell, T.: Machine Learning. McGraw-Hill, New York (1997)
7. Machine Learning course, University of Cambridge, UK, `http://learning.eng.cam.ac.uk/zoubin/ml06/index.html` (accessed on February 10, 2011)
8. Kearns, M.J., Vazirani, U.V.: An introduction to computational learning theory. MIT Press, Cambridge (1994)
9. Vapnik, V.N.: The nature of statistical learning theory. Springer, Heidelberg (1995)
10. Vapnik, V.N.: Statistical learning theory. Wiley and Sons, Chichester (1998)
11. Foundations of machine learning theory course, Princeton University, `http://www.cs.princeton.edu/courses/archive/spr03/cs511/` (accessed on February 10, 2011)
12. `http://www-it.fmi.uni-sofia.bg/markov/courses/ml.html#IntroductiontoMachineLearning` (accessed on February 10, 2011)

Chapter 11
Decision Trees

11.1 Introduction

Decision trees are suitable for scientific problems entail labeling data items with one of a given, finite set of classes based on features of the data items. Decision Trees are classifiers that predict class labels for data items [3]. A decision tree learning algorithm approximates a target concept using a tree representation, where each internal node corresponds to an attribute, and every terminal node corresponds to a class[5][6][10].

There are two types of nodes in the tree:

- Internal node: splits into different branches according to the different values the corresponding attribute can take. Example: fever < 37 or fever > 37, cough weak or cough strong in the example below.
- Terminal Node: decides the class assigned.

A decision tree is a branching structure, which consists of nodes and leafs. The root node is at the top and leafs at the bottom. Each node tests the value of some feature of an example, and each leaf assigns a class label to the example. Consider a simple example of classifying a patient's symptoms into two classes – class1 and class2 – corresponding to whether the patient has or has not a cold (see Figure 11.1).

The elements of a decision tree representation have the following meaning:

- each internal node tests an attribute;
- each branch corresponds to an attribute value;
- each leaf node assigns a classification.

One can also use a re-representation as if-then rules: disjunction of conjunctions of constraints on the attribute value instances.

To classify an example X we start at the root of the tree, and check the value of that attribute on X. We follow the branch corresponding to that value and jump to the next node. We continue until we reach a terminal node and take that class as our best prediction [1].

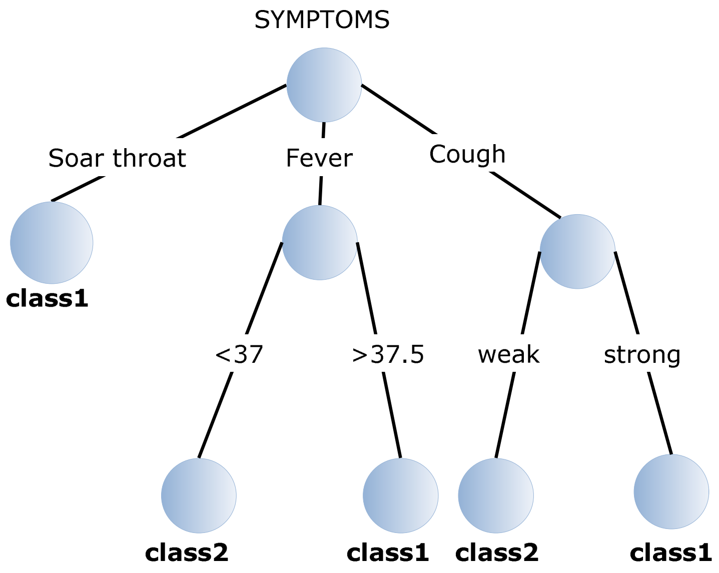

Fig. 11.1 Decision tree for classifying whether a patient has cold (class1) or not (class2).

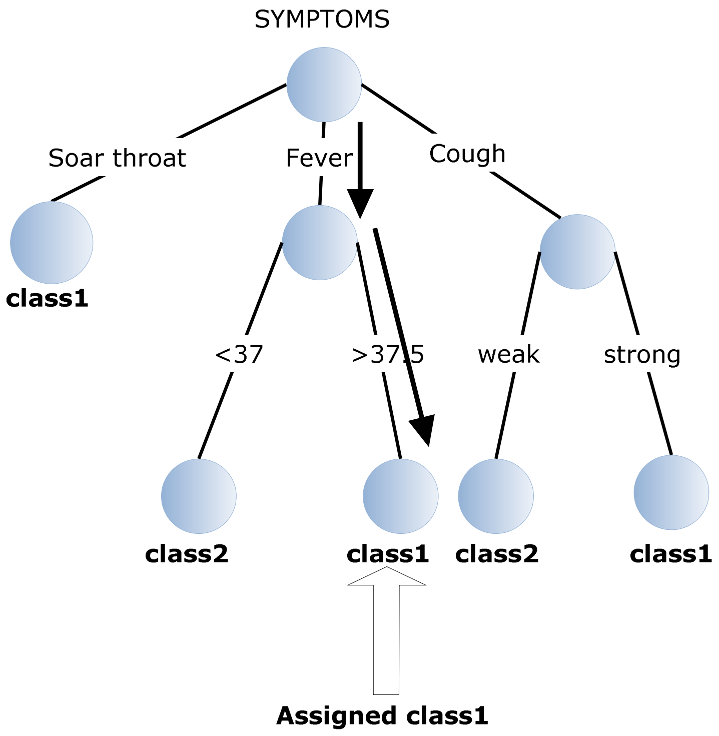

Fig. 11.2 Example of assigning a class to a patient based on the symptoms.

In our example, if we have for instance

X = (Symptoms: fever; fever >37.5)

then the assigned class is represented in Figure 11.2

11.2 Building a Decision Tree

Decision trees are constructed by analyzing a set of training examples for which the class labels are known. They are then applied to classify previously unseen examples.

There are different ways to construct trees from data. In what follows we will present the top-down approach.

11.2.1 Top-Down Induction of Decision Tree

The algorithm is presented below:

```
Step 1. Create a root for the tree
Step 2.
        Step 2.1. If all examples are of the same class
        or the number of examples is below a threshold
        Then return that class
        Step 2.2. If no attributes available return
        majority class
Step 3.
        Step 3.1. Let A be the best attribute for the
        next node
        Step 3.2. Assign A as decision attribute for
        node
Step 4. For each possible value v of A
        Create a new descendant of node. Add a branch
        below A labeled "A = v"
Step 5. Let Sv be the subsets of example where attribute
        A=v
        Recursively apply the algorithm to Sv
Step 6. If training examples are perfectly classified
        Then Stop
        Else iterate over new leaf nodes.
End
```

The algorithm terminates either when all the attributes have been exhausted, or the decision tree perfectly classifies the examples.

Example
Let us illustrate this using the following example (adapted from [1]) for classifying a patient with cold symptoms into one of the two classes: cold or not-cold.

The graphical representation of the initial data is given in Figure 11.3.

We can observe we have two attributes: fever and cough.

Fever has three values:

- < 37;
- >37 and < 38;
- >38.

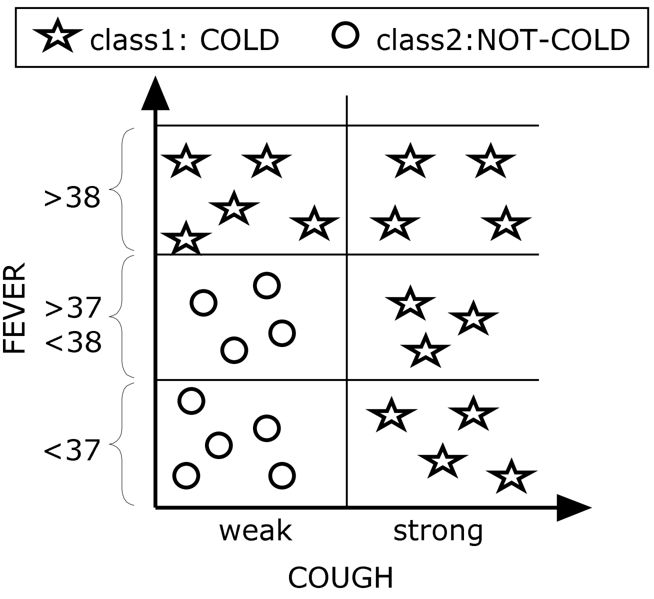

Fig. 11.3 Graphical representation of the initial data.

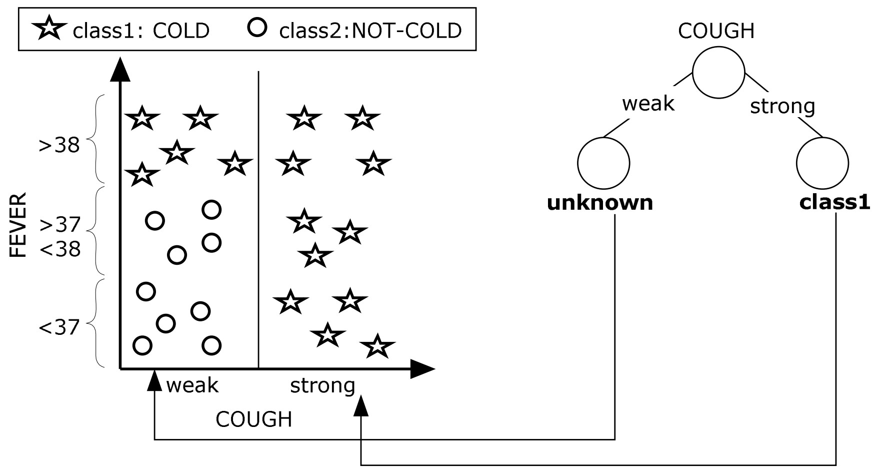

Fig. 11.4 Decision tree construction after considering the first attribute: *cough*.

Cough attribute has two values:

- weak;
- strong.

Suppose we have cough as best attribute. The decision tree obtained until now is depicted in Figure 11.4.

By considering the first attribute – cough – we can now have the root of the tree and two branches corresponding to the two values – weak and strong.

The right branch – strong – leads to a class (which is class1 - cold), but we still don’t have a final classification on the left branch. We should now consider the next best attribute – which is the only remaining one: fever. The decision tree obtained is depicted in Figure 11.5.

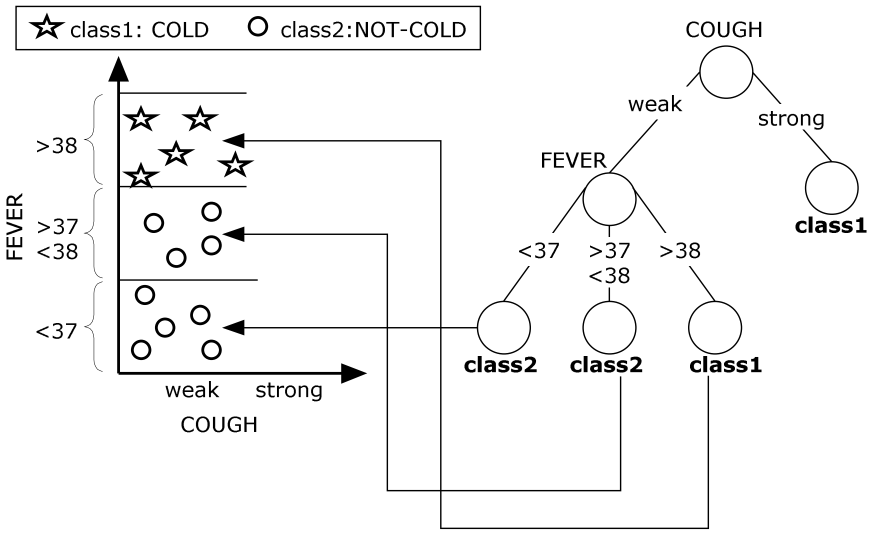

Fig. 11.5 Constructed decision tree.

11.2.2 How to Chose the Best Attribute?

Putting together a decision tree is all a matter of choosing which attribute to test at each node in the tree. We shall define a measure called *information gain* which will be used to decide which attribute to test at each node. Information gain is itself calculated using a measure called *entropy*, which we first define for the case of a binary decision problem and then define it for the general case.

In the example above we considered *cough* as the best attribute. But how to chose this or how to define this in a consequent manner?

In order to choose an attribute to best split the data we will use entropy based splitting function.

Let us use the following notations:

- S – sample of training data
- p_+– the proportion of positive examples in S
- p_-– the proportion of negative examples in S

Entropy measures the impurity in S and it is given by:

$$\textit{Entropy}\ (S) = -\ p_+ \log_2 p_+ -\ p_- \log_2 p_-$$

Entropy (S) is the expected number of bits needed to encode the class (+ or -) of randomly drawn member of S under the optimal, shortest length code [2].

As known from information theory, the optimal length code assigns - $\log_2$ p bits to message having probability p.

Thus, the expected number of bits to encode + or – random member of S is:

$$\sum_{p \in \{p_+, p_-\}} p(-\log_2 p)$$

And thus:

$$\textit{Entropy}(S) = \sum_{p \in \{p_+, p_-\}} p(-\log_2 p)$$

Coming back to our example, the attribute *cough* divides the sample set into two subsets S1 and S2 (as shown in Figure 11.6):

- S1 = {5 +, 9 -}
- S2 = {11 +, 0 -}.

Then we have:

$$\textit{Entropy}(S1) = -\frac{5}{14}\log_2\left(\frac{5}{14}\right) - \frac{9}{14}\log_2\left(\frac{9}{14}\right)$$

$$\textit{Entropy}(S2) = 0$$

The attribute *fever* divides the sample set into three subsets S1, S2 and S3 (as shown in Figure 11.8):

- S1 = {9 +, 0 -}
- S2 = {3 +, 4 -}.
- S3 = {4+, 5-}

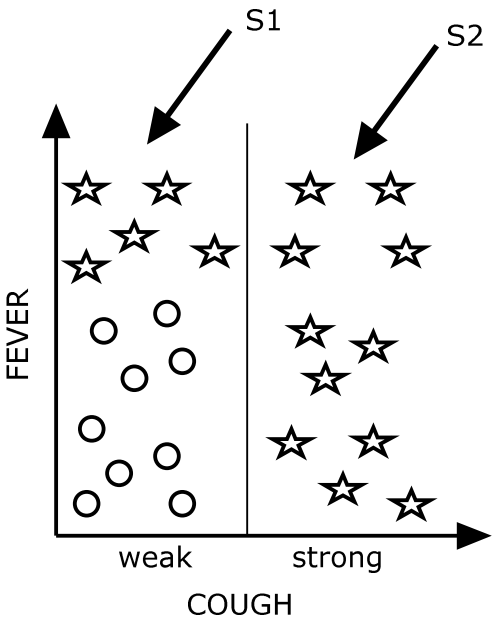

Fig. 11.6 The split of data into two subsets S1 and S2 using the attribute *cough*.

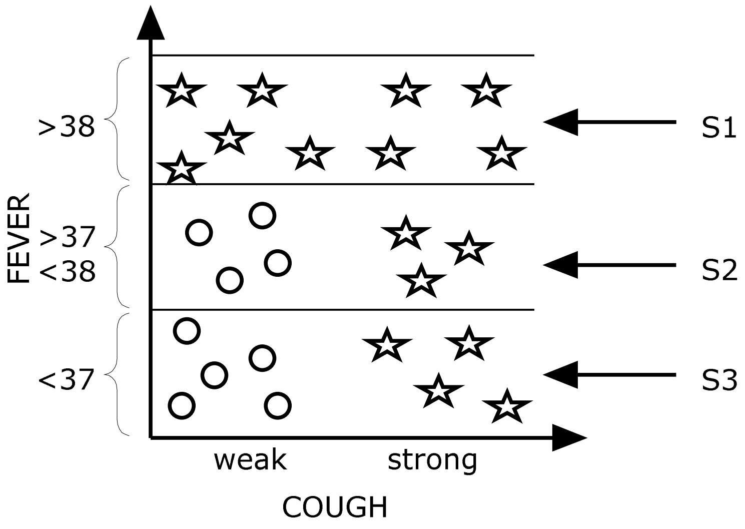

Fig. 11.7 The split of data into three subsets S1, S2 and S3 using the attribute *fever*.

In this case we have:

$$Entropy(\mathrm{S1}) = 0$$

$$Entropy(\mathrm{S2}) = -\frac{3}{7}\log_2\left(\frac{3}{7}\right) - \frac{4}{7}\log_2\left(\frac{4}{7}\right)$$

$$Entropy(\mathrm{S3}) = -\frac{4}{9}\log_2\left(\frac{4}{9}\right) - \frac{5}{9}\log_2\left(\frac{5}{9}\right)$$

Information gain gives the expected reduction in entropy due to sorting on A. Information gain is given by the formula:

$$gain(S, A) = Entropy(S) - \sum_{v \in values(A)} \frac{|Sv|}{|S|} \cdot Entropy(Sv)$$

where *Entropy*(*Sv*) is the entropy of one sub-sample after partitioning S based on all possible values of attribute A.

For the example considered in Figure 11.6 we have:

$$Entropy(\mathrm{S1}) = -\frac{5}{14}\log_2\left(\frac{5}{14}\right) - \frac{9}{14}\log_2\left(\frac{9}{14}\right)$$

$$Entropy(\mathrm{S2}) = 0$$

$$Entropy(S) = -\frac{16}{25}\log_2\left(\frac{16}{25}\right) - \frac{9}{25}\log_2\left(\frac{9}{25}\right)$$

$$\frac{|S1|}{|S|} = \frac{14}{25}$$

$$\frac{|S2|}{|S|} = \frac{11}{25}.$$

11.3 Overfitting in Decision Trees

Overfitting is a common problem in machine learning. Decision trees suffer from this, because they are trained to stop when they have perfectly classified all the training data, i.e., each branch is extended just far enough to correctly categorize the examples relevant to that branch. Many approaches to overcoming overfitting in decision trees have been attempted.

In order to define overfitting consider an hypothesis h and $error_{train}(h)$ the error of hypothesis h over training data and $error_D(h)$ the error of hypothesis h over the entire distribution D of data.

Hypothesis h overfits training data if there is an alternative hypothesis h' such that [2]:

(i) $error_{train}(h) < error_{train}(h')$

and

(ii) $error_D(h) > error_D(h')$.

The depth of the tree is related to the generalization capability of the tree. If not carefully chosen it may lead to overfitting. A tree *overfits* the data if we let it grow deep enough so that it begins to capture "aberrations" in the data that harm the predictive power on unseen examples [1].

If we add some noise in our example given in Figure 11.3, the tree will grow deeper to capture this noise (as it can be observed in Figure 11.8).

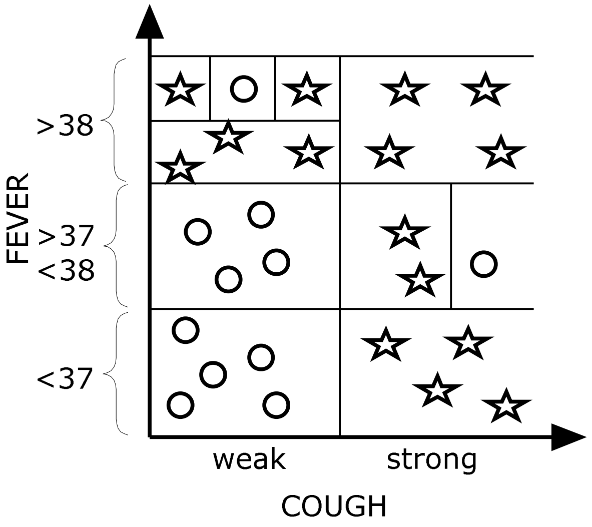

Fig. 11.8 Example from figure 11.3 modified by adding some noise which increases the tree size.

There are two main ways to avoid overfitting [1][2]:

1) Stop growing the tree when data split is not statistically significant.
   This is hard to implement in practice because it is not clear what a good stopping point is.
2) Grow the full tree and then post-prune.

11.3.1 Pruning a Decision Tree

The algorithm below describes the main steps required to prune a decision tree [1][2]:

```
Step 1. Split data into training and validation set.
Step 2. Consider all internal nodes in the tree.
Step 3. Do
        Step 3.1 For each node in the tree evaluate im-
        pact on validation set of pruning it (plus those
        nodes below it) and assign it to the most common
        class.

        Step 3.2 In a Greedy way remove the one (and its
        sub-tree) that most improves validation set accu-
        racy (yields the best performance).
Until further pruning is harmful (or no more improve-
ments are possible):
end
```

Approaches for extracting decision rules from decision trees have also been successful.

Post-pruning of rules follows the steps [2]:

1) Convert the tree to an equivalent set of rules;
2) Prune each rule independently of others;
3) Sort the final rules into a desired sequence for use.

11.4 Decision Trees Variants

Although single decision trees can be excellent classifiers, increased accuracy often can be achieved by combining the results of a collection of decision trees. This forms ensembles of decision trees and are sometimes among the best performing types of classifiers [3] [7] [8] [9].

Two of the strategies for combining decision trees are:

- random forests and
- boosting.

Random forests is a machine learning ensemble classifier in which many different decision trees are grown by a randomized tree-building algorithm. The training set is sampled with replacement to produce a modified training set of equal size to the original but with some training items included more than once. In addition, when choosing the question at each node, only a small, random subset of the features is considered. With these two modifications, each run may result in a slightly different tree. The predictions of the resulting ensemble of decision trees are combined by taking the most common prediction [3].

One of the random forests disadvantages is that does not handle large numbers of irrelevant features as well as ensembles of entropy-reducing decision trees.

Maintaining a collection of good hypotheses, rather than committing to a single tree, reduces the chance that a new example will be misclassified by being assigned the wrong class by many of the trees.

Boosting is a machine-learning method used to combine multiple classifiers into a stronger classifier by repeatedly re-weighting training examples to focus on the most problematic.

In practice, boosting is often applied to combine decision trees [3]. Although it is usually applied to decision tree models, it can be used with any type of model and it is a special case of the model averaging approach.

Alternating decision trees are a generalization of decision trees that result from applying a variant of boosting to combine weak classifiers based on decision stumps, which are decision trees that consist of a single question. In alternating decision trees, the levels of the tree alternate between standard question nodes and nodes that contain weights and have an arbitrary number of children. In contrast to standard decision trees, items can take multiple paths and are assigned classes based on the weights that the paths encounter.

Alternating decision trees can produce smaller and more interpretable classifiers than those obtained from applying boosting directly to standard decision trees [3].

Summaries

There are many different learning algorithms that have been developed for supervised classification and regression. These can be grouped according to the formalism they employ for representing the learned classifier or predictor: decision trees, decision rules, neural networks, linear discriminant functions, Bayesian networks, support vector machines, etc. [4].

A decision tree is a branching structure, which consists of nodes and leafs. The root node is at the top and leafs are at the bottom. Each node tests the value of some feature of an example, and each leaf assigns a class label to the example. This chapter presented a top-down algorithm for learning decision trees.

Decision tree learning provides a practical method for concept learning/learning discrete-valued functions. Decision trees are sometimes more interpretable than other classifiers such as neural networks and support vector machines because they combine simple questions about the data in an understandable way [3].

This algorithm gets into trouble overfitting the data. This occurs with noise and correlations in the training set that are not reflected in the data as a whole. In order to deal with overfitting, one can restrict the splitting, so that it splits only when the split is useful or can allow unrestricted splitting and prune the resulting tree where it makes unwarranted distinctions. One of the advantages of using decision trees is that, if they are not too large, they can be interpreted by humans. This can be useful both for gaining insight into the data and also for validating the reasonableness of the learned tree [4].

We should consider decision trees in the following situations [2]:

- instances can be described by attribute-value pairs;
- attributes are both numeric and nominal.
- target function is discrete valued;
- disjunctive hypothesis may be required;
- possibly noisy training data;
- data may have errors.

References

1. `http://www.onlinefreeebooks.net/free-ebooks-computer-programming-technology/artificial-intelligence/artificial-intelligence-course-material-pdf.html` (accessed on February 10, 2011)
2. Mitchell, T.M.: Machine learning. McGraw-Hill, New York (1997)
3. Kingsford, C., Salzberg, S.L.: What are decision trees? Nature Biotechnology 26, 1011–1013 (2008)
4. Dietterich, T.G.: Machine Learning. In: Nature Encyclopedia of Cognitive Science. Macmillan, London (2003)
5. Quinlan, J.R.: C4.5: Programs for Machine Learning. Morgan Kaufmann Publishers, San Mateo (1993)
6. Breiman, L., Friedman, J., Olshen, R., Stone, C.: Classification and Regression Trees. Wadsworth International Group, Belmont (1984)
7. Breiman, L.: Random forests. Machine Learning 45, 5–32 (2001)
8. Heath, D., Kasif, S., Salzberg, C.: Committees of decision trees. In: Gorayska, B., Mey, J. (eds.) Cognitive Technology: In Search of a Human Interface, pp. 305–317. Elsevier Science, Amsterdam (1996)
9. Schapire, R.E.: The boosting approach to machine learning: an overview. In: Denison, D.D., Hansen, M.H., Holmes, C.C., Mallick, B., Yu, B. (eds.) Nonlinear Estimation and Classification, pp. 141–171. Springer, New York (2003)
10. Freund, Y., Mason, L.: The alternating decision tree learning algorithm. In: Bratko, I., Džeroski, S. (eds.) Proceedings of the 16th International Conference on Machine Learning, pp. 124–133. Morgan Kaufmann, San Francisco (1999)

Verification Questions

1) What are the steps required to build a decision tree?
2) Explain some ways to choose the best attribute
3) What is overfitting?
4) Enumerate the steps for pruning a decision tree.
5) Nominate some decision trees variants.