Chapter 8
A Few Instructive Applications

For someone wishing to become an expert on machine learning, mastering a handful of baseline techniques is not enough. Far from it. The world lurking behind a textbook's toy domains has a way of complicating things, frustrating the engineer with unexpected obstacles, and challenging everybody's notion of what exactly the induced classifier is supposed to do and why. Just as in any other field of technology, success is hard to achieve without a healthy dose of creativity.

Some practical experience helps, too, either your own, or at least of those who have tried and succeeded (or failed) before you. And this is what this chapter wants to offer. Using a few carefully selected case studies, it will acquaint you with some issues typically encountered in realistic applications—and with practical ways of dealing with them.

8.1 Character Recognition

The techniques from the previous chapters target recognition skills that are too difficult to be hard-coded in a computer program, but can be conveyed by means of pre-classified training examples. The ability to read text, even hand-written text, belongs to this category. Applications are legion: automated pre-processing of various forms, software for converting a text scribbled on a notepad into a format to be used by a text editor, and various newspaper-digitization programs. A mere generation or two ago, an undertaking of this kind was judged so ambitious as to be almost unrealistic; today, no one finds it extraordinary.

The Task To describe each character in a way that facilitates its recognition by a computer is not easy. To get the idea, just try to explain, in plain English, how to distinguish digit "5" from digit "6," or what constitutes the differencebetween a

M. Kubat, *An Introduction to Machine Learning*,
DOI 10.1007/978-3-319-63913-0_8

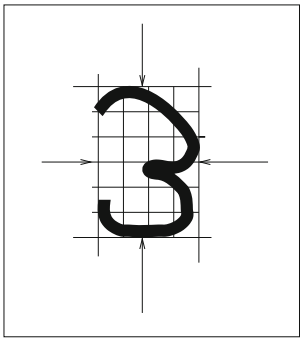

Fig. 8.1 A simple way to convert the image of a hand-written character into a vector of 64 continuous attributes, each giving the mean intensity of the corresponding field

hand-written "t" and "l." You will be surprised how difficult this is. And to convert the plain-English explanation to a code that a computer can understand is harder still.

This is why machine learning has been asked to help. If we prepare a collection of training examples, and describe them by pre-classified attribute vectors, then surely some of the techniques described in the previous chapters can induce the requisite classifier. The process is so simple as to seem almost trivial. Let us take a look at some of its crucial aspects.

Examples and Attribute Vectors To begin with, the engineer has to define the attributes to describe the examples. Figure 8.1 illustrates one possibility. The first step identifies a rectangular region in which the digit is located; the second divides this region into $6 \times 4 = 64$ equally sized fields, each characterized by a continuous attribute whose value gives the field's mean intensity. More ink means the field reflects less light, and thus results in a lower value of the attribute, and, conversely, the value is maximized if the field contains no ink at all. Of course, this is just the principle. There is no reason why there should be only 64 attributes. Indeed, the higher the level of detail the program is expected to discern, the smaller the size of the fields it needs to rely on, and thus the greater the number of attributes that will be used to describe the examples.

In a realistic application, the classifier will have to recognize not just one isolated character (such as the "3" in Fig. 8.1), but rather to "read" a whole text consisting of many such characters. This generalization, however, does not represent any major complication. As long as the individual characters can be isolated, and then treated separately, the same principle of describing each of them by an attribute vector can be applied.

Nowadays it is possible to download datasets of hundreds of thousands of hand-written characters, some described in the way presented above, some relying on other approaches.

Choosing the Classifier Now that we know how the training examples are going to be described, we are ready to proceed to the choice of the induction technique.

The first thing to be considered is that, in the attribute-vector obtained by the mechanism from Fig. 8.1, only a small percentage of the attributes (if any) are likely to be irrelevant or redundant, which means that this is not an issue to worry about (unless the number of attributes is increased way beyond those shown in Fig. 8.1). Another aspect guiding our choice of an appropriate machine-learning paradigm is the fact that we do not know whether the classes are linearly separable, and therefore hesitate to use linear classifiers. Finally, the classifiers need not be capable of offering explanations. If the intention is to read and convert to a text editor a long text, the user will hardly care to know the exact reason why a concrete character was classified as a "P" and not as a "D."

Based on these observations, the simple and easy-to-implement k-NN classifier looks like a good candidate. A cautious engineer will perhaps be concerned about the computational costs incurred in a domain with hundreds of thousands of training and testing examples. But as long as the number of attributes is moderate, these costs are unlikely to be prohibitive. From the practical point of view, a reasonable limit on what constitutes "prohibitive costs" will be determined by the computations associated with the isolation of the individual characters and the conversion of their images to attribute vectors. As long as these are comparable with the costs of classification (and they usually are), the classifier is affordable.

And indeed, the nearest-neighbor classifier is the most common choice, in this application, typically exhibiting an error rate of less than 2%. In some really illegible hand-writings, the error rate will of course be higher. But then, we should not be too harsh on the innocent machine, knowing as we do that even an experienced pharmacist finds it difficult to read certain hand-written prescriptions.

The Number of Classes In a domain where all characters are capitalized, the induced classifier is to discern 10 digits and 26 letters, which amounts to 36 classes. If both lowercase and uppercase letters are allowed, the total increases to $10 + 2 \times 26 = 62$ classes, and to these we may have to add special characters such as "?," "!," "$," and so on. This relatively high number of classes is not without consequences, and these deserve our attention.

The most immediate concern is the induced product's evaluation. Mere information about the error rate is somewhat inadequate here. Thus the performance of a classifier that correctly identifies 98% characters may appear good enough, even impressive; what this value fails to tell us, though, is how the errors are distributed. Typically, some characters will be correctly identified most of the time, while others pose difficulties—and as such, deserve further attention. For instance, certain pairs of similar characters tend to be mutually confused; the practically minded engineer then wants to know *which* pairs so as to mitigate the problem by providing additional training examples for the "difficult" classes.

Moreover, some letters will be less common than others. In a situation of this kind, it is known that the rarer classes get "overlooked" by the classifier-inducing algorithms unless special precautions have been taken. Section 10.2 will have more to say about this issue.

The Classifier Should Be Allowed to Reject an Example To decipher a person's handwriting is far from easy. Certain letters are so ambiguous as to make the reader shrug his shoulders in despair. Yet the *k*-NN classifier from Chap. 3 is undeterred: it always finds a nearest neighbor, and then simply returns its class, no matter how arbitrary this class is.

Practical experience shows this circumstance to be harmful because the costs of getting the wrong class can be greater than the costs of not knowing the class at all. Thus in an automated reader of postal codes, an incorrectly read digit can result in the letter being sent to a wrong destination, which, in turn, may cause great delay in delivery. On the other hand, if the classifier does not give *any* answer, a human employee will have to do the reading. The costs of manual processing may then be lower than the costs of getting the wrong address.

We have convinced ourselves that the classifier should be implemented in a way that makes it possible to refuse to classify an example if the evidence supporting the winning class is insufficient. The simplest way of doing so in the context of the *k*-NN classifier is to require a certain minimum margin between the number of votes supporting the winner and the number of votes supporting the runner-up. For instance, if the winning class in a 7-NN classifier receives only four votes as compared to the three votes supporting another class, the example is rejected as ambiguous.

Something similar is easy to accomplish also in some other paradigms such as the Bayesian classifiers or neural networks: the classifier simply compares the probabilities (or output signals) of the two most likely classes, and rejects this example if the difference does not exceed a predefined threshold.

Error Rate Versus Rejection Rate A classifier that rejects ambiguous examples will surely reduce its error rate; on the other hand, excessive reluctance to classify will not be beneficial, either. What if *all* examples are rejected? The error rate then drops to zero—and yet the user will question the tool's practical utility.

The lesson is, the engineer needs to consider the trade-off between the rejection rate and the error rate. True enough, increasing the former is likely to reduce the latter; but overdoing it may render the classifier useless.

What Have You Learned?

To make sure you understand this topic, try to answer the following questions. If you have problems, return to the corresponding place in the preceding text.

- Explain how to describe hand-written characters by attribute vectors.
- What aspects did we consider (and why) when looking for the most appropriate machine-learning tool to be used here?
- What is the immediate consequence of the relatively high number of classes in this domain? Why is here the error rate unable to give the full picture?

- Why should the classifier be allowed to refuse to classify certain examples? Discuss the trade-off between error rate and rejection rate; comment on the interplay between performance and utility.

8.2 Oil-Spill Recognition

Figure 8.2 shows a radar image of the sea surface as taken by a satellite-born device. Against the grayish background, the reader can see several dark regions of the most varied characteristics: small or large, sharp or barely discernible, and of all possible shapes. What interests us here primarily is the sharp and elongated object in the vicinity of the upper-right corner: an oil spill, illegally dumped by a tanker's captain who has chosen to get rid of the residues in its bilges in the open sea rather than doing so in a specially designed terminal as required by the law. As such, this particular oil spoil is of great interest to the Coast Guard.

For all relevant sea-surface areas (close to major ports, for example), satellites take many of such "snapshots" and send them down to collaborating ground stations. Here, experts pore over these images in search for signs of illegal oil spills. Whenever they detect one, an airplane is dispatched and verifies the suspicion by a spectrometer (which is unavailable to the satellite), collects evidence, and perhaps even identifies the perpetrator.

Unfortunately, human experts are expensive—and not always available. They may be on holidays, on a sick leave, or not present for any number of other reasons. Besides, in view of the high number of images, the work is tedious, and prone to human errors. This is why the idea came up to develop a computer program to automate the oil-spill recognition process.

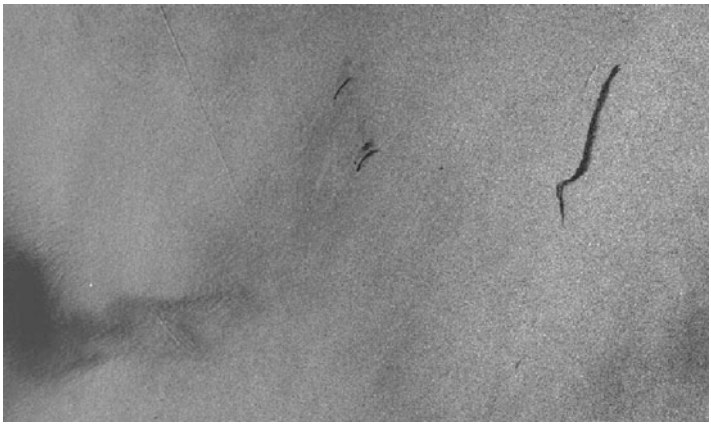

Fig. 8.2 A radar image of a sea surface. The "wiggly" elongated dark region in the upper-right corner represents environmental hazard: an oil spill

The picture shown in Fig. 8.2 has been selected out of many, the main criterion being its rare clarity. Indeed, the oil spill it contains is so different from the other dark regions that even an untrained eye will easily recognize it as such. Even so, the reader will find it difficult to specify the oil-spill's distinguishing features in a manner that can be used in a computer program. In the case of more realistic objects, the task will be even more difficult. At any rate, to hard-code the oil-spill recognition ability is quite a challenge.

Again, machine learning got its chance, the intention being to let the machine develop the requisite skills automatically, by induction from training examples. The general scenario of the adventure can be summarized into the following steps.

1. collect a set of radar images containing oil spills;
2. use some image-processing software capable of identifying, in these images, the dark regions of interest;
3. ask an expert to label the oil spills as positive examples, and the other dark regions (so-called "look-alikes") as negative examples;
4. describe all examples by attribute vectors, and let a machine-learning program induce the requisite classifier from the training set thus obtained.

As in the previous application, we will try to glean some useful lessons by taking a closer look at certain critical aspects of this undertaking.

Attributes and Class Labels State-of-the-art image-processing techniques easily discover dark regions in an image. For these to be used by the machine-learning tool, we need to describe them by attributes that are hoped to capture those aspects that distinguish spills from "look-alikes." Preferably, their values should be unaffected by the given object's size and orientation.

The attributes that have been used in this project include the region's mean intensity, average edge-gradient (which quantifies the sharpness of the edges), the ratio between the lengths of the object's minor-axis and major-axis, variance of the background intensity, variance of the edge gradient, and so on. All in all, more than forty such attributes were selected in a rather ad hoc manner. Which of them would really be useful was hard to tell because experts were unable to reach consensus about the attributes' respective relevance and redundancy. The final choice was left to the machine-learning software.

Labeling the training examples with classes was not any easier. The objects in Fig. 8.2 were easy to categorize; in other images, they were much more ambiguous. On many occasions, the best the expert could say was, "yes, this looks like an oil spill" or, "I rather doubt this is what we are looking for." The correctness of the selected class labels was thus uncertain, resulting in class-label noise. The presence of class-label noise reduces our expectations: the classifier can be only as good as the data that have been used during its induction.

Choosing the Classifier The examples were described by some forty attributes. Among these, some, perhaps most, were suspected of being either irrelevant or redundant. This is an important circumstance; the reader will recall that the presence of irrelevant and redundant attributes makes some classifiers underperform. In this

particular project, the problem was side-stepped by first inducing a decision tree, and then eliminating all attributes that never appeared in any of the tree's tests. This was quite logical. After all, the choice of which attributes to include in the tree has been made based on the attributes' information contents. Since information contents of irrelevant attributes is low, this method of identifying them is quite reliable. Also redundant attribute can thus be eliminated, at least to some extent.

When the k-NN classifier was applied to examples described by attributes that "survived" this decision-tree-based elimination process, the classification performance turned out to be acceptable. The oil-spill problem was thus solved with a very simple machine-learning technique—which is good: having a choice, the engineer is always well advised to give preference to the simpler tool.

The decision to use the k-NN classifier was also driven by another important consideration. Since this is typical of many realistic applications, let us discuss it in some detail.

Cost of Errors Performance evaluation is here not as straightforward as it was in the previous chapters. For one thing, error rate can be a fairly misleading indicator in domains where each type of error carries a different penalty.

This is manifestly the case in the oil-spill adventure. Here, a false positive results in an aircraft being unnecessarily dispatched to a suspicious though "innocent" region, and this means a waste of time and resources (note that the costs are not only financial, but also moral because these kind of failures tend to undermine the user's trust in the classifier). On the other hand, a false negative means an undetected environmental hazard whose consequences (financial, environmental, and political) are hard to predict. In view of all this, the reader will agree that the two types of cost are of such a different nature that it is almost impossible to compare them. This, of course, makes it difficult to specify the project's goal.

Experiments on pre-classified testing data indicated that most of the errors made by the induced classifier were of the false-positive kind (i.e., false alarms). As for false negatives, these were relatively rare. But then: was this good, or should this observation be taken as a signal of a need to modify the classifier in order to change the ratio of the two types of errors?

The question cannot be answered in isolation from the application's momentary needs. Financial constraints may force the user occasionally to accept the risk of environmental hazard, simply because the budget can no longer tolerate false alarms. The user then wants to reduce the frequency of false positives even if this means to pay the price of an increased number of undetected oil spills (false negatives).

However, the situation will change in more prosperous times when the user who does not want to miss an oil spill is prepared to accept higher frequency of false positives for the sake of making false negatives rare.

Leaning Toward One or the Other Class In view of these trade-offs, it is necessary to give the user the opportunity to adjust the classifier's behavior so as to modify the frequency of one or the other type of error.

As already mentioned, this project relied on the *k*-NN classifier where this requirement is easy to satisfy: the trick consists in manipulating the margin between the number of votes supporting either of the two classes. For the sake of illustration, suppose the 7-NN classifier is used. Here, the number of false positives can be reduced if we instruct the classifier to label as positive only those examples where, say, at least five of the nearest neighbors are positive; any example that fails to satisfy this condition is deemed negative. Conversely, if we desire to lower the frequency of false negatives, we tell the classifier to return the positive label whenever, say, at least three of the nearest neighbors are positive. The user's preference for either type of error is thus expressed in terms of the number of votes that are necessary for the example to be labeled as positive.

What Have You Learned?

To make sure you understand this topic, try to answer the following questions. If you have problems, return to the corresponding place in the preceding text.

- How did the engineers deal with the fact that many attributes were redundant or irrelevant? What did they identify these less-then-useful attributes?
- What can be said about the reliability of the class labels in the training set? What does it mean for the classifier's expected performance?
- Discuss the respective costs, in this domain, of the two types of error: false positive versus false negative. Can they be compared using the same units? Why does the user need a mechanism to reduce one type of error at the cost of the other?
- Explain the essence of the mechanism that enables the *k*-NN classifier to increase or reduce either of the two errors.

8.3 Sleep Classification

Throughout the night, we go through different sleep stages such as deep, shallow, or rapid-eye movements (REM, this is when we dream). To identify these stages in a concrete sleeping subject, advanced instrumentation is used: an electrooculogram to record eye movements, an electromyogram to record muscle contractions, and contact electrodes attached to the scalp to record the brain's neural signals. Based on the readings of all these instruments, a medical expert can decide what stage the sleeping subject is in at any given moment, and can even draw a *hypnogram* such as the one shown in Fig. 8.3.

Note that the deepest sleep stage occurs here only three times during the 8-h sleep, and that it usually does not last long. Note also the `move` stage; this occurs, for instance, when the subject turns from one side to another, moves the head or an arm, and the like.

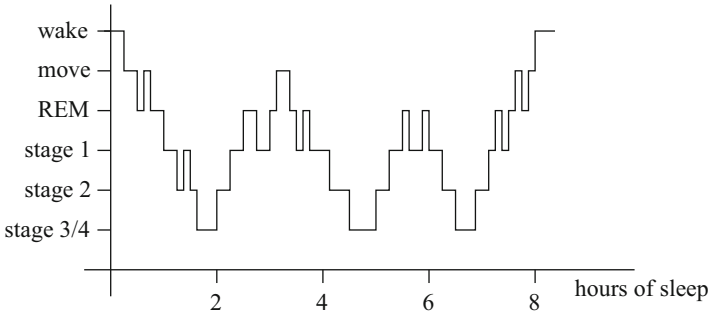

Fig. 8.3 An example hypnogram that records the sleep stages experienced by a subject during an 8-h sleep

Why Is It Important Medical practice needs to be able to recognize specific sleep stages. A case in point is the so-called *sudden infant death syndrome* (SIDS): an infant dies without any apparent cause. A newborn suspected of being in danger has to be watched, in a hospital, so that resuscitation can be started immediately. Fortunately, SIDS is known almost always to occur during the REM stage. This means that it is not necessary to watch the patient all the time, but only during this period of increased risk. For instance, a device capable of recognizing the onset of the REM stage might alert the nurse that more attention might be needed during the next five of so minutes.

The hypnogram, in turn, is a useful diagnostic tool because the distribution of the sleep stages during the night may indicate specific neural disorders such as epilepsy.

Why Machine Learning To draw the hypnogram manually is a slow and tedious undertaking, easily taking 3–5 h of a highly qualified expert's time. Moreover, the expert is not always available. This is why efforts have been made to develop a computer program capable of identifying the individual sleep stages based on observed data, and, hopefully, even to draw the hypnogram.

To be able to help, the computer needs a description of the individual stages. Such description, however, is difficult to obtain. Medical experts rely on skills obtained in the course of long training, and they use features and indications that are too subjective to be converted to a computer program.

This motivated an attempt to induce the classifier from pre-classified data. Specifically, the data-acquisition process divided the 8-h sleep period into 30-s samples, each of them treated as a separate training example. All in all, a few hours' sleep thus provided hundreds of training examples. Diverse instruments than provide data to describe each 30-s sample.

Attributes and Classes Again, the first task was to remove attributes suspected of being irrelevant or redundant. In the previous application, oil-spill recognition, this removal was motivated by the intention to increase the performance of the k-NN classifier (which is known to be sensitive to their presence). In sleep classification,

another reason comes to the fore: the physician wants to minimize the number of measurement devices attached to the sleeping subject. Not only does their presence make the subject feel uncomfortable; they also disturb the sleep, and thus interfere with the results of the sleep analysis.

As for the class labels, these are even less reliable than in the oil-spills domain. The differences between "neighboring" (similar) sleep stages are so poorly defined that any two experts rarely agree on more than 70–80% of the class labels. No wonder that the training set contains a lot of class-label noise, and the low quality of the data imposes a limit on the minimum error rate that any realistic classifier induced from the data can achieve.

The Classifier and Its Performance The classifier employed in this particular case combined decision trees with a neural network in a manner whose details are unimportant of our needs here. Suffice it to say that the classifier's accuracy on independent data indeed achieved those 70–80% observed in human experts, which means that the natural performance limits have been reached. It is perhaps worth noting that plain decision trees were a few percent weaker than that.

This said, it is important to understand that classification accuracy does not give the full picture of the classifier's behavior (similarly as in the OCR domain from Sect. 8.1). For one thing, the classifier correctly recognized some of the seven classes most of the time, while failing on others. Particularly disappointing was its treatment of the REM state. Here, classification accuracy was in the range 90–95%, which, at first sight, looked good enough. However, closer inspection of the training data revealed that only less than 10% of all examples belonged to the REM class; this means that a comparable classification accuracy could be achieved by a classifier that says, "there is not a single REM example"—and yet this is hardly what the engineers hoped to achieve.

We realize that this way of measuring performance is not without its limitations, and that other criteria have to be found. These indeed exist. They will be discussed in Chap. 11.

Improving Classification Performance by Post-processing The accuracy of the hypnogram can be improved by post-processing whose nature relies on the domain's logic. Indeed, several rules of thumb can be used here. For instance, the deepest sleep (`stage 3/4`) is unlikely to occur right after the REM stage, and `stage 2` does not happen after `move`. Also, the hypnogram can be "smoothed out" by the removal of any stage lasting only one 30-s period. Applying such rules in the course of post-processing makes it possible to improve the hypnogram's informational value.

The lesson is worth remembering. In domains where the examples are ordered in time, the classes of the individual examples may not be independent of those preceding or following them. In this event, post-processing can improve the classifier's performance.

Proper Use of the Induced Classifier The original idea was to induce the classifier from examples obtained from a few subjects, and then to classify future data using this induced classifier instead of the much more expensive "manual" classification.

This ambition, however, turned out to be unrealistic. Practical experience showed that no "universal classifier" of sleep data could be obtained in this manner: a classifier induced from one person's data could not be used to draw a hypnogram for another person without serious degradation in classification performance.[1]

This does not mean that machine learning is in this domain totally disqualified. Far from it. However, the user needs to modify his or her expectations: instead of being universal, the classifier will be induced separately for each subject. The expert's efforts are still significantly reduced. The following three-step scenario is indicated:

1. The expert determines the class labels of a subset of the available examples, thus creating a training set.
2. From this training set, a classifier is induced.
3. The induced classifier is used to classify the remaining examples, thus saving the expert's time.

What Have You Learned?

To make sure you understand this topic, try to answer the following questions. If you have problems, return to the corresponding place in the preceding text.

- Why was it important to minimize the number of attributes in this domain? How were the relevant attributes identified?
- The sleep-classification domain has seven classes. What does it mean for an engineer trying to evaluate the induced classifier's performance?
- In the hypnogram, the examples are ordered in time. How can this circumstance be exploited in data post-processing?
- In what manner can machine learning reduce the burden imposed on someone who seeks to classify available data?

8.4 Brain–Computer Interface

The muscle-controlling commands are issued at specific locations of *motor cortex*, a relatively well-understood region of the cortex. Even the brain of many totally paralyzed patients is fully capable of generating these commands; unfortunately, the information fails to reach the muscles. The fact that these signals can be detected by contact electrodes inspired a fantastic idea: can these signals, properly recorded and interpreted, be used to control a cursor on a computer screen? If yes, then there

[1] One can speculate that a different set of attributes might perhaps make this possible; the case study reported here did not attempt to do so.

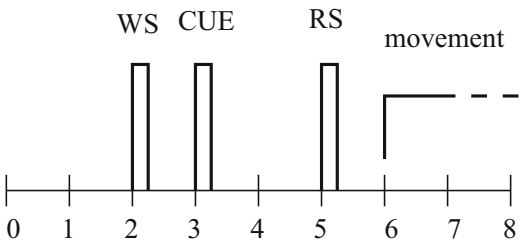

Fig. 8.4 Organization of the experiment. After a "ready" signal (WS) comes the CUE ("left" vs. "right"). Once RS appears on the screen, the subject waits 1 s, then presses the button indicated by the CUE

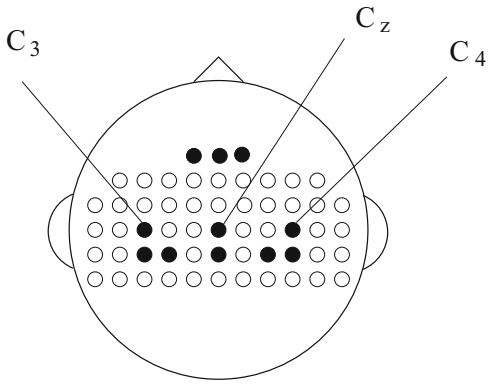

Fig. 8.5 Arrangement of electrodes on the subject's scalp. Only some of the electrodes are actually used (*highlighted*)

might be a way for the patient to communicate with the outside world, in spite of being unable to speak, in spite of being unable even to move his or her pupils.

The exact nature of the motor commands is too complicated to be expressed by a mathematical formula. But we can record the signals, describe them by attributes, and label with the concrete motor commands. If we do so, we have a training set from which a machine-learning program can induce the classifier.

The Training Examples In the specific case to be described in this section, only two classes were considered: `left` and `right`. The training and testing examples were obtained in the course of the following experiment. A subject was sitting in front of a computer monitor. On the desk in front of him was a wooden board with two buttons, one to be pressed by the left index finger, the other by the right index finger, according to instructions displayed on the monitor (Fig. 8.4).

The whole scenario followed the time-line shown in the upper part of Fig. 8.5. The contact electrodes attached to the scalp (see the bottom part of Fig. 8.5) record the intensity of the neural signals during a certain period of time before the button is pressed. The signals from each electrode are characterized by five numeric attributes, each representing the power of the signal over a specific time interval (a fraction of a second).

As indicated in the picture, only some (typically 11) of the electrodes were actually used. For 11 electrodes, each represented by 5 attributes, this amounted to 55 attributes. All in all, the subject "produced" a few hundred examples with both classes equally represented. Since it was always known which button was

actually pressed, the training set was free of class-label noise. As for the attributes, the experimenters suspected many of them to be irrelevant.

Classifier Induction and Its Limitations Several machine learning paradigms were experimented with, including multilayer perceptrons and nearest-neighbor classifiers with attribute-value ranges normalized so as to avoid scaling-related problems. Relevant attributes were selected by a decision tree, similarly as in the applications discussed in the previous sections.

Typical error rates of these induced classifiers on testing data were in the range 20–30%, depending on the concrete subject. It is quite possible that better accuracy could not be achieved: the information provided by the electrodes might have been insufficient for this kind of experiment.

Importantly, just as in the sleep-classification domain, the classifier could only be applied to the subject from whose training data it has been induced. All attempts to induce a "general classifier" (to be used on any future subject) failed—the error rates were so high as to make the classifications process look random.

Testing the Classifier's Utility As in the previous domain, the error rate measured on independent data does not give the whole picture; the classifier's practical utility depends on how it is actually employed. In our case, the real question is: will the classifier succeed in sending the cursor to the correct location?

The question was answered by an experiment illustrated in Fig. 8.6. The subject is still sitting in front of a computer monitor, with electrodes attached to the scalp. However, the board with the two buttons has been removed. On the monitor, we can see two rectangles, one on the left and one on the right. In the middle is a cursor in the shape of a little cross.

When instructed to move the cursor, say, to the right, the subject only *imagined* he was pushing the right button. The electrodes recorded the neural signals and passed them to the classifier. Based on these, the classifier selected the side (the class, `left` or `right`) to which the cursor is to be moved. The movement was very slow so as to give the subject the opportunity to "correct" the wrong direction. To make this possible, the neural signals were sampled repeatedly, each sample becoming one example to be passed on to the classifier. Suppose the subject's intention is to move the cursor to the right. It is possible that the first sample will be misclassified as

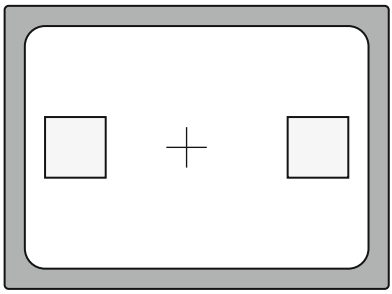

Fig. 8.6 The classifier's task is to move the cross into either the *left box* or the *right box*, according to the subject's "thoughts"

`left`, sending the cursor in the wrong direction; but if the following samples are correctly classified as `right`, the cursor will, after all, land in the correct rectangle.

What Error Rate Is Acceptable? Under the scenario just described, an error rate of 30% (which seems quite high) turned out to be fully acceptable. Even when the cursor occasionally did start in the wrong direction (on account of one "example" being misclassified), there was still the chance of correcting the mistake before the cursor reached its destination. As a result, the cursor almost always landed in the correct box, even if in a somewhat hesitating manner in which it moved back and forth before finding the right direction.

An impartial observer, uninformed about how the classifier was actually used, rarely noticed anything was going wrong. The occasional errors on the part of the classifier (manifested by the cursor's back-and-forth movements) were sometimes perceived as a sign of indecisiveness on the part of the subject, and not of the classifier's imperfection.

Only when the classifier's error rate dropped well below those 30% did the cursor miss its target frequently enough to cause disappointment. One can speculate that a better measure of the classifier's performance would in this case be the average time needed for moving the cursor.

An Honest Historical Remark The classifier's performance differed from one person to another. On quite a high percentage of the subjects, the induced classifiers exhibited an almost prohibitively high error rate. Also the fact that the classifier had to be trained on the same patient on which it was supposed to be used was perceived as a shortcoming. Further on, two classes are not enough. At the very least, also `up` and `down` classes would be needed. On these, the experimental results were less impressive. In the end, other methods of communication with paralyzed patients came to be studied. Though successful as a machine-learning exercise, the project did not satisfy the expectations of the medical domain.

What Have You Learned?

To make sure you understand this topic, try to answer the following questions. If you have problems, return to the corresponding place in the preceding text.

- Discuss the experience made in this application: a classifier induced from the data of one subject cannot be used to classify examples in another subject.
- Explain why the error rate of 30% was in this domain still deemed acceptable. What other methods of measuring performance, in this application would you recommend?

8.5 Medical Diagnosis

A physician attempting to find the cause of her patient's complaints behaves a bit like a classifier: based on certain attributes (the patient's symptoms, results of laboratory tests), she suggests a diagnosis—and a treatment. No wonder that, in the early days of machine learning, many believed medical diagnosis to be its natural, almost straightforward application.

Some Results The optimism was motivated by early studies such as the one whose results are summarized in Table 8.1. The numbers give the classification accuracy achieved by Bayesian classifiers, decision trees, and physicians on the same testing set—patients described on attribute vectors. The four domains differ in the number of classes and in the difficulty of the task (e.g., noise in the data, relevance and reliability of the available information).

The values indeed seem to confirm machine-learning's ability to induce classifiers capable of competing with human experts; indeed, they seem to outperform them. In the multi-class `primary tumor` domain, the margin is perhaps unimpressive, but in all the other domains, the machine learning classifiers appear to be clearly better. It is only a pity that the table does not tell us more about the variation in the results. At least standard deviations would help, here, but there are other ways how to ascertain whether the results are statistically reliable—Chaps. 11 and 12 will have more to say about the methods to be used here.

Are the Results Encouraging? The results seem impressive, but we must not jump to conclusions. The first question to ask is whether the comparison was fair. Since the examples were described by attributes, the data available to the machine-learning tool, and to the induced classifier, could hardly be the same as used by the physician who could rely also on subjective information that might not have been available to the machine. In this respect, the human enjoyed a certain advantage, and this makes the machine learning's results all the more impressive.

On the other hand, the authors of the study admit that the physicians participating in the study were not necessarily top specialists in the given field, and machine learning thus outperformed only general practitioners. Another aspect worth our attention is that this study was conducted in the 1980s when the diagnostic tools were less sophisticated than those in use today. It is hard to tell who will benefit more from modern laboratory tests: human experts or machine learning?

Table 8.1 Classification accuracy of two classifiers compared to the performance of physicians in four medical domains

	Bayesian classifier	Decision tree	General practitioner
Primary tumor	0.49	0.44	0.42
Breast cancer	0.78	0.77	0.64
Thyroid	0.70	0.73	0.64
Rheumatology	0.67	0.61	0.56

Apart from Classification, Also Explanation Is Needed For medical diagnosis, classification accuracy is not enough; more accurately, it is not enough for the diagnosis to be correct. A patient will hardly agree with a surgery if the only argument supporting this recommendation is, "this is what the machine says." The statement that "the machine is on average a few percentage points better than a physician" is unlikely to convince the patient, either.

What a reasonable person wants is a convincing explanation of why the surgery is to be preferred over a more conservative treatment. In the case of the domains from Table 8.1, the doctor is usually able to itemize what evidence supports the concrete diagnosis and why, and what treatment options are recommended in this particular case and why. Unfortunately, the baseline machine learning techniques are not very good at explaining their decisions (perhaps with the exception of decision trees). Software for automated reasoning would be needed, here. This, however, is beyond the scope of this book.

The Need for Measuring Confidence There is another problem to be aware of. Most of the classifiers induced by the techniques treated in the previous chapters only give the final verdict; for instance, "the example belongs to class 77." The physician (as well as the patient) needs to know more. For instance, is the recommended class absolutely certain or is it only just a little more likely than some other class? And if so, which other classes should be considered?

In other words, medical domains usually call for classifiers capable of telling us how confident they are in the returned class. For example, such classifier should say that "the example belongs to C_1 with probability 0.8, and to C_5 with probability 0.7." There probabilities are explicitly calculated by Bayesian classifiers, so these may be a good choice; similar information can be obtained also from neural networks. Both of these approaches, however, are rather unable to offer explanations.

On the other hand, decision trees *are* capable of offering some kind of explanation. As for the confidence, this is only possible with additional functions that we have not treated here.

Cultural Barriers There is one last reason why the encouraging results of the early applications of machine learning to medical diagnosis failed to inspire followers. With only minor injustice, they can be called *cultural barriers.* In a way, they are understandable. Medical doctors surely did not like to be told how easily they would be replaced by computers. No wonder that, in the early days of machine learning, many of them were not eager to collaborate on the development of the requisite software.

This, of course, is a misunderstanding. Machine learning is not meant to replace a human expert. At its very best, is only offer advice; the final diagnostic decision will always be the responsibility (even in legal terms) of the physician treating the patient. Still, the value of the advice should not be underestimated. For instance, the classifier can alert the doctor that additional, previously unsuspected, disorders my accompany the current diagnosis (the patient often suffers from more than just one problem at a time), and it can even point out the need to take specific additional laboratory tests.

What Have You Learned?

To make sure you understand this topic, try to answer the following questions. If you have problems, return to the corresponding place in the preceding text.

- As for the results mentioned in this section, why was it impossible to conclude from them that the induced classifiers outperformed the human expert?
- Apart from classification accuracy, what is needed in medical diagnosis?
- Discuss the limitations of machine-based diagnosis. Suggest a reasonable way of realistic application of machine learning in medical domains.

8.6 Text Classification

Suppose you have a vast collection of text documents, and you need to decide which of them is relevant to a specific topic of interest. If there are really a great many of them, there is no way you can do so manually. However, taking inspiration for some of the above-described applications, you choose a manageable subset of these documents, read them, assign the class labels to them (positive versus negative), and induce a classifier that will then be used to classify the rest.

Attributes A common way to describe a document is by the frequency of the words it contains. Each attribute represents one word, and its value represents the frequency with which it appears in the text. For instance, in a document that is 983 words long, the term "classifier" may appear five times, in which case its frequency is $5/983 = 0.005$.

Since the vocabulary typically contains tens of thousands of words, the attribute vectors will be very long. And of course, since only a small subset of the vocabulary finds its way into the document, most attribute values will be zero. This being somewhat unwieldy, simple applications prefer to work with only a subset of the whole vocabulary, say, 1000 most common words.

Class Labels The class labels for the training examples may be difficult to determine. Even if we want only something very easy such as to decide, for each document, if it deals with `computer science`, whether an example clearly belongs into this class may not be easy to decide because some documents are more relevant to this category than others. For instance, a very clear-cut case will be a scientific paper dealing with algorithm analysis; on the other hand, a less relevant document will only mention in passing that some algorithms are more expensive than others; and yet another document will be an article from a popular magazine.

One possibility to deal with this circumstance is to "grade" the class labels. For instance, if the document is most certainly relevant, the class is 2; if it is only somewhat relevant, the class is 1; and only if it is totally irrelevant, the class is 0. The user can then decide what level of relevance is needed for the given application.

Typical Observations The great number of attributes makes the induction computationally expensive. Thus in the case of decision trees, where one has to calculate the information gain separately for each attribute, this means that tens of thousands of attributes need to be investigated for each of the tests, and these calculations can take a lot of time, especially if the number of examples is high. Upon some thought, the reader will agree that Bayesian classifiers and neural networks are quite expensive, too.

One can suggest that induction will be cheaper in the case of the k-NN classifier or a linear classifier. Here, however, another difficulty comes to the fore: for each given topic, the vast majority of the attributes will be irrelevant, a circumstance that reduces the utility of both of these paradigms.

The Problem of Multi-Label Examples The applications discussed in the previous sections assumed that each example is labeled with one and only one class. Here, however, each document can belong to two or more (sometimes many) classes at the same time.

The most commonly used way of dealing with the situation is to induce a different classifier for each class. Whenever a future document's class is needed, the document's attribute vector is submitted to all of these classifiers in parallel; some of them will return "1" (meaning the document belongs to the corresponding class), others will return a "0."

This, however, makes the adventure even more expensive because of the need to induce hundreds of classifiers. And given that the induction of each of these classifiers is expensive in its own right, the result can be prohibitive.

Moreover, the classes are not independent of each other. At the very least, some are subclasses or others, giving rise to a whole hierarchy of classes. This means that their corresponding classifiers perhaps should not be induced in a manner totally independent of the induction of the other classifiers.

What Have You Learned?

To make sure you understand this topic, try to answer the following questions. If you have problems, return to the corresponding place in the preceding text.

- How are examples typically described in text-classification domains? Why are the attribute vectors so long?
- What did this section mean by "graded class labels"?
- What makes induction of text classifiers expensive?
- Discuss the problem of multi-label examples.

8.7 Summary and Historical Remarks

- The number of examples that can be used for learning depends on the particular application. In some domains, examples are abundant; for instance, if they can be automatically extracted from a database. In others, they may be rare and expensive, as was the case of the oil spill domain.
- We may be totally in the dark as to which attributes really matter. This was the case of the oil spill domain where the shape and other characteristics could be described by a virtually unlimited number of attributes obtained by image-processing techniques.
- In some domains, the really important attributes are not known or cannot be obtained at reasonable costs. Inevitably then, the induction has to use those attributes that are available, even if the performance of the classifier thus induced will be diminished.
- In the brain–computer-interface domain, it turned out to be enough that a majority of the decisions were correct: in this event, the cursor landed in the intended rectangle. We can see that the maximum classification accuracy may not be strictly necessary.
- In domains with more than just two classes, the error rate does not give the full picture of the classifier's behavior. Quite often, some classes will be almost always perfectly recognized while others will pose problems. It is therefore important to know not only the average performance, but also the performance on each individual class.
- In some domains, we want to be able to explain the decision. High classification accuracy is not enough. This is the case of medical diagnosis.
- The costs of false positives can be different from the costs of false negatives. In the oil-spill domain, it was difficult to express these costs in monetary terms.
- Many applications are interested in minimizing error rate. Sometimes, however, error rate fails to give the full picture.
- Another important requirement: the possibility that the user be able to tweak a parameter that will trade false positives for false negatives (see the oil spill domain)

Historical Remarks The results from Table 8.1 are taken from Kononenko et al. [48] who refer there to an older project of theirs. The oil-spill project was reported in Kubat et al. [52]. The sleep classification task is addressed by Kubat et al. [50], and the experience with using machine learning in brain–computer interface was published by Kubat et al. [53]. The character-recognition problem has been addressed for decades by the field of Computer Vision; the first major text systematically addressing the issue from the machine-learning perspective seems to be Mori et al. [69]. The text-classification task was first studied by Lewis and Gale [55].

A minor remark is in place, here. Of these six applications, the author of this book participated in two. His motivation for including his own work was not to stun the reader with his incredible scholarship and inestimable contribution to the field

of machine learning. Far from it. The truth is, if you work on a project for quite some time, you develop not only personal attachment to it, but also certain deeper understanding that makes this particular application more appropriate for instruction purposes than those which you only read about in the literature.

8.8 Exercises and Thought Experiments

The exercises are to solidify the acquired knowledge. The ambition of the suggested thought experiments is to let the reader see this chapter's ideas in a different light and, somewhat immodestly, to provoke his or her independent thinking.

Give It Some Thought

1. Suppose that you know that the correctness of some class labels in your training set is not certain. Would you recommend that these "unreliable" examples be removed from the training set? In your considerations, do not forget that some of the pre-classified examples will be used as testing examples to assess the classification performance of the induced classifier.
2. Discuss the reasons why application-domain users may be reluctant to accept machine-learning tools. Suggest ways to eliminate, or at least diminish, their suspicions and concerns.
3. Section 8.2 mentioned a simple mechanism by which the k-NN classifier can manipulate the two kinds of error (false negatives versus false positives). Suggest a mechanism that a Bayesian classifier or a neural network can use to the same end. How would you go about implementing this mechanism in a linear classifier?
4. In more than one of the domains discussed in this chapter, it was necessary to reduce the number of irrelevant and/or redundant attributes. In the projects reported here, decision trees were used. Suggest another possibility that would use a technique from one of the previous chapters. Discuss its advantages and disadvantages.
5. Suppose you wanted to implement a program that would decide whether a given position in the tic-tac-toe game (see Fig. 8.7) is winning. What attributes would you use? How would you collect the training examples? What can you say about expected noise in the data thus collected? What classifier would you use? What difficulties are to be expected?

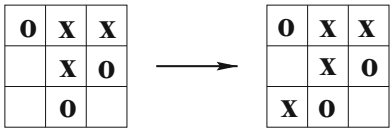

Fig. 8.7 In tic-tac-toe, the goal is to achieve three *crosses* (or *circles*) in a row or a column or on a diagonal

Computer Assignments

1. Some realistic data sets for machine-learning experimentation can be found on the website of the National Institute of Standards and Technology (NIST). Find this web site, then experiment with some of these domains.
2. Go to the web and find a website about the demography of the 50 states in the U.S. Identify an output variable whose value will be deemed positive if it is above the U.S. average and negative otherwise. Each state thus constitutes an example. Based on the information provided by the website, identify the attributes to describe the examples. From the data thus obtained, induce a classifier to predict the output variable's value.