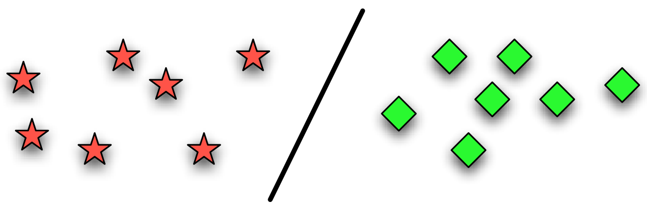

Fig. 1.5. Binary classification; separate stars from diamonds. In this example we are able to do so by drawing a straight line which separates both sets. We will see later that this is an important example of what is called a *linear classifier*.

want to build a system which is able to *learn* how to classify new e-mails. A seemingly unrelated problem, that of cancer diagnosis shares a common structure: given histological data (e.g. from a microarray analysis of a patient's tissue) infer whether a patient is healthy or not. Again, we are asked to generate a yes/no answer given a set of observations. See Figure 1.5 for an example.

1.1.2 Data

It is useful to characterize learning problems according to the type of data they use. This is a great help when encountering new challenges, since quite often problems on similar data types can be solved with very similar techniques. For instance natural language processing and bioinformatics use very similar tools for strings of natural language text and for DNA sequences. **Vectors** constitute the most basic entity we might encounter in our work. For instance, a life insurance company might be interesting in obtaining the vector of variables (blood pressure, heart rate, height, weight, cholesterol level, smoker, gender) to infer the life expectancy of a potential customer. A farmer might be interested in determining the ripeness of fruit based on (size, weight, spectral data). An engineer might want to find dependencies in (voltage, current) pairs. Likewise one might want to represent documents by a vector of counts which describe the occurrence of words. The latter is commonly referred to as bag of words features.

One of the challenges in dealing with vectors is that the *scales* and units of different coordinates may vary widely. For instance, we could measure the height in kilograms, pounds, grams, tons, stones, all of which would amount to multiplicative changes. Likewise, when representing temperatures, we have a full class of affine transformations, depending on whether we represent them in terms of Celsius, Kelvin or Farenheit. One way of dealing with those issues in an automatic fashion is to normalize the data. We will discuss means of doing so in an automatic fashion.

Lists: In some cases the vectors we obtain may contain a variable number of features. For instance, a physician might not necessarily decide to perform a full battery of diagnostic tests if the patient appears to be healthy.

Sets may appear in learning problems whenever there is a large number of potential causes of an effect, which are not well determined. For instance, it is relatively easy to obtain data concerning the toxicity of mushrooms. It would be desirable to use such data to infer the toxicity of a new mushroom given information about its chemical compounds. However, mushrooms contain a cocktail of compounds out of which one or more may be toxic. Consequently we need to infer the properties of an object given a *set* of features, whose composition and number may vary considerably.

Matrices are a convenient means of representing pairwise relationships. For instance, in collaborative filtering applications the rows of the matrix may represent users whereas the columns correspond to products. Only in some cases we will have knowledge about a given (user, product) combination, such as the rating of the product by a user.

A related situation occurs whenever we only have similarity information between observations, as implemented by a semi-empirical distance measure. Some homology searches in bioinformatics, e.g. variants of BLAST [AGML90], only return a similarity score which does not necessarily satisfy the requirements of a metric.

Images could be thought of as two dimensional arrays of numbers, that is, matrices. This representation is very crude, though, since they exhibit spatial coherence (lines, shapes) and (natural images exhibit) a multiresolution structure. That is, downsampling an image leads to an object which has very similar statistics to the original image. Computer vision and psychooptics have created a raft of tools for describing these phenomena.

Video adds a temporal dimension to images. Again, we could represent them as a three dimensional array. Good algorithms, however, take the temporal coherence of the image sequence into account.

Trees and Graphs are often used to describe relations between collections of objects. For instance the ontology of webpages of the DMOZ project (`www.dmoz.org`) has the form of a tree with topics becoming increasingly refined as we traverse from the root to one of the leaves (Arts → Animation → Anime → General Fan Pages → Official Sites). In the case of gene ontology the relationships form a directed acyclic graph, also referred to as the GO-DAG [ABB+00].

Both examples above describe estimation problems where our observations are vertices of a tree or graph. However, graphs themselves may be the observations. For instance, the DOM-tree of a webpage, the call-graph of a computer program, or the protein-protein interaction networks may form the basis upon which we may want to perform inference.

Strings occur frequently, mainly in the area of bioinformatics and natural language processing. They may be the input to our estimation problems, e.g. when classifying an e-mail as spam, when attempting to locate all names of persons and organizations in a text, or when modeling the topic structure of a document. Equally well they may constitute the output of a system. For instance, we may want to perform document summarization, automatic translation, or attempt to answer natural language queries.

Compound structures are the most commonly occurring object. That is, in most situations we will have a structured mix of different data types. For instance, a webpage might contain images, text, tables, which in turn contain numbers, and lists, all of which might constitute nodes on a graph of webpages linked among each other. Good statistical modelling takes such dependencies and structures into account in order to tailor sufficiently flexible models.

1.1.3 Problems

The range of learning problems is clearly large, as we saw when discussing applications. That said, researchers have identified an ever growing number of templates which can be used to address a large set of situations. It is those templates which make deployment of machine learning in practice easy and our discussion will largely focus on a choice set of such problems. We now give a by no means complete list of templates.

Binary Classification is probably the most frequently studied problem in machine learning and it has led to a large number of important algorithmic and theoretic developments over the past century. In its simplest form it reduces to the question: given a pattern x drawn from a domain $\mathcal{X}$, estimate which value an associated binary random variable $y \in \{\pm 1\}$ will assume. For instance, given pictures of apples and oranges, we might want to state whether the object in question is an apple or an orange. Equally well, we might want to predict whether a home owner might default on his loan, given income data, his credit history, or whether a given e-mail is spam or ham. The ability to solve this basic problem already allows us to address a large variety of practical settings.

There are many variants exist with regard to the protocol in which we are required to make our estimation:

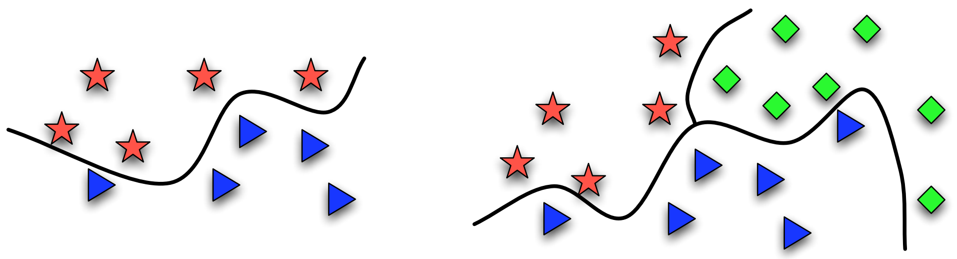

Fig. 1.6. Left: binary classification. Right: 3-class classification. Note that in the latter case we have much more degree for ambiguity. For instance, being able to distinguish stars from diamonds may not suffice to identify either of them correctly, since we also need to distinguish both of them from triangles.

- We might see a sequence of (x_i, y_i) pairs for which y_i needs to be estimated in an instantaneous online fashion. This is commonly referred to as online learning.
- We might observe a collection $\mathbf{X} := \{x_1, \ldots x_m\}$ and $\mathbf{Y} := \{y_1, \ldots y_m\}$ of pairs (x_i, y_i) which are then used to estimate y for a (set of) so-far unseen $\mathbf{X}' = \left\{x'_1, \ldots, x'_{m'}\right\}$. This is commonly referred to as batch learning.
- We might be allowed to know $\mathbf{X}'$ already at the time of constructing the model. This is commonly referred to as transduction.
- We might be allowed to choose $\mathbf{X}$ for the purpose of model building. This is known as active learning.
- We might not have full information about $\mathbf{X}$, e.g. some of the coordinates of the x_i might be missing, leading to the problem of estimation with missing variables.
- The sets $\mathbf{X}$ and $\mathbf{X}'$ might come from different data sources, leading to the problem of covariate shift correction.
- We might be given observations stemming from two problems at the same time with the side information that both problems are somehow related. This is known as co-training.
- Mistakes of estimation might be penalized differently depending on the type of error, e.g. when trying to distinguish diamonds from rocks a very asymmetric loss applies.

Multiclass Classification is the logical extension of binary classification. The main difference is that now $y \in \{1, \ldots, n\}$ may assume a range of different values. For instance, we might want to classify a document according to the language it was written in (English, French, German, Spanish, Hindi, Japanese, Chinese, ...). See Figure 1.6 for an example. The main difference to before is that the cost of error may heavily depend on the type of

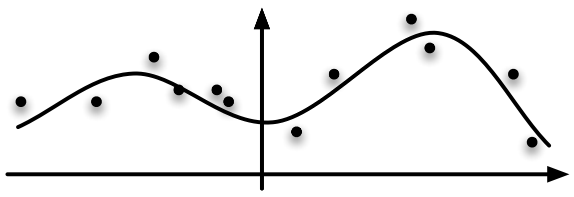

Fig. 1.7. Regression estimation. We are given a number of instances (indicated by black dots) and would like to find some function f mapping the observations $\mathcal{X}$ to $\mathbb{R}$ such that $f(x)$ is close to the observed values.

error we make. For instance, in the problem of assessing the risk of cancer, it makes a significant difference whether we mis-classify an early stage of cancer as healthy (in which case the patient is likely to die) or as an advanced stage of cancer (in which case the patient is likely to be inconvenienced from overly aggressive treatment).

Structured Estimation goes beyond simple multiclass estimation by assuming that the labels y have some additional structure which can be used in the estimation process. For instance, y might be a path in an ontology, when attempting to classify webpages, y might be a permutation, when attempting to match objects, to perform collaborative filtering, or to rank documents in a retrieval setting. Equally well, y might be an annotation of a text, when performing named entity recognition. Each of those problems has its own properties in terms of the set of y which we might consider admissible, or how to search this space. We will discuss a number of those problems in Chapter **??**.

Regression is another prototypical application. Here the goal is to estimate a real-valued variable $y \in \mathbb{R}$ given a pattern x (see e.g. Figure 1.7). For instance, we might want to estimate the value of a stock the next day, the yield of a semiconductor fab given the current process, the iron content of ore given mass spectroscopy measurements, or the heart rate of an athlete, given accelerometer data. One of the key issues in which regression problems differ from each other is the choice of a loss. For instance, when estimating stock values our loss for a put option will be decidedly one-sided. On the other hand, a hobby athlete might only care that our estimate of the heart rate matches the actual on average.

Novelty Detection is a rather ill-defined problem. It describes the issue of determining "unusual" observations given a set of past measurements. Clearly, the choice of what is to be considered unusual is very subjective. A commonly accepted notion is that unusual events occur rarely. Hence a possible goal is to design a system which assigns to each observation a rating