8

Applications of Support Vector Machines

In this chapter we illustrate the use of the algorithms described in this book, by examining some interesting applications. Support Vector Machines have been applied to many real-world problems, so that the material in this chapter is by no means exhaustive. The role of this final chapter is to show how the approach can be used successfully in very diverse fields, and how the necessary design choices can be made in each case.

Applying the Support Vector approach to a particular practical problem involves resolving a number of design questions. We do not dwell on the question of precise specification of the input domain, generation of the training data, and so on. We treat these as given, though in practice they can frequently be refined through an interaction between the SVM system designer and the domain practitioner.

Such a collaboration may also be required to resolve the first design problem, that of choosing an appropriate kernel for the given application. There are standard choices such as a Gaussian or polynomial kernel that are the default options, but if these prove ineffective or if the inputs are discrete structures more elaborate kernels will be needed. By implicitly defining a feature space, the kernel provides the description language used by the machine for viewing the data. Frequently, the designer can work with the more intuitive notion of similarity in the input space and this is where domain experts can give invaluable assistance in helping to formulate appropriate similarity measures.

Typically the choice of kernel will in fact be a family of kernels parametrised by some hyperparameter, as for example in the case of Gaussian kernels where the parameter σ remains to be determined. Again there may be heuristics for setting these parameters; in the Gaussian case a good choice of σ is the distance between closest points with different classifications; but frequently the choice must be made in response to the data. In the absence of reliable criteria, applications rely on the use of a validation set or cross-validation to set such parameters.

The next decision facing the SVM system designer is which flavour of SVM to implement. Assuming the problem requires the data to be classified, a decision must be taken whether to use the maximal margin, or one of the soft margin approaches. Here, the critical factors will be the dimensionality of the data and the type and level of noise present in their generation.

Once the choice of kernel and optimisation criterion has been made the key components of the system are in place. Experience with other learning systems

might lead one to expect that what follows is a lengthy series of experiments in which various parameters are tweaked until eventually satisfactory performance is achieved. The examples we report in this chapter we believe demonstrate that in many cases the most straightforward SVM implementations outperform other techniques without any need for further adaptation. This experience with SVMs suggests that the range of areas in which they can successfully be applied is far from fully explored.

The applications that we describe are intended to serve as examples of what can be achieved. We therefore give the main thrust of each case study and refer the reader to the relevant papers should they wish to get a detailed description of the approach and results obtained. Our focus is on describing the problem and the specific kernel together with the type of optimisation used in the given application.

We include cases of text (and hypertext) categorisation, image classification, biosequence analysis and biological data mining, hand-written character recognition, etc. We have tended to choose simple examples that are frequently more illustrative, though we will refer to more complex studies of the same problems in Section 8.5.

8.1 Text Categorisation

The task of text categorisation is the classification of natural text (or hypertext) documents into a fixed number of predefined categories based on their content. This problem arises in a number of different areas including email filtering, web searching, office automation, sorting documents by topic, and classification of news agency stories. Since a document can be assigned to more than one category this is not a multi-class classification problem, but can be viewed as a series of binary classification problems, one for each category.

One of the standard representations of text for the purposes of information retrieval (IR) provides an ideal feature mapping for constructing a Mercer kernel. Hence, in this case the use of prior knowledge developed in another field inspires the kernel design; we will see in the following sections that this is a pattern repeated in many case studies. Indeed, the kernels somehow incorporate a similarity measure between instances, and it is reasonable to assume that experts working in the specific application domain have already identified valid similarity measures, particularly in areas such as information retrieval and generative models.

8.1.1 A Kernel from IR Applied to Information Filtering

In the information retrieval literature a common technique for finding text documents is the so-called vector space model, also known as the bag-of-words representation. In this method a document x is represented by a vector $\phi(x)$ indexed by a pre-fixed set or dictionary of terms; the value of an entry can be a Boolean variable indicating the presence of the corresponding term, or can indicate the weight of the term within the document in a way we will describe below. Finally, the vectors are normalised to remove information about the

length of the text. The distance between two documents is then computed by calculating the inner product between the corresponding vectors. In this model, all information about the order of the words is lost, a situation that resembles the case of image classification discussed below in Section 8.2.

Research in information retrieval has shown that word stems provide good representation units. A word stem is derived from a word by removing case and other inflection information. An example is the words 'computer', 'computation' and 'computing', which all have the same word stem 'comput'. The dictionary can be chosen by ranking all the word stems occurring in the database according to their mutual information, removing uninformative words (such as 'and', etc.) also known as *stop-words*, and selecting the first word stems.

A standard choice for the weighting factor for each word stem is the following:

$$\phi_i(x) = \frac{\mathrm{tf}_i \log(\mathrm{idf}_i)}{\kappa},$$

where tf_i is the number of occurrences of the term i in the document x, idf_i is the ratio between the total number of documents and the number of documents containing the term, and κ is the normalisation constant ensuring that $||\phi||_2 = 1$.

In this way a document is represented by a vector in which each entry records how many times a particular word stem is used in the document weighted according to how informative that word stem is. Typically ϕ can have tens of thousands of entries, often comparable with or even larger than the number of training examples. Furthermore, for a particular document the representation is typically extremely sparse, having only a few non-zero entries. But given that only inner products between the data are needed for the algorithm and the evaluation function, a series of techniques can be exploited to facilitate the computations.

The function $K(x,z) = \langle \phi(x) \cdot \phi(z) \rangle$ is clearly a valid kernel, since it is the inner product in an explicitly constructed feature space. Its kernel matrix is therefore always positive semi-definite, and Support Vector Machines can be used with $K(x,z)$ for discrimination. Given the extreme sparseness and the high dimensionality of the vector $\phi(x)$, it is reasonable to assume that the points in the feature space are going to be easy to separate, and that a standard hard margin SVM will be sufficient. Furthermore, it is possible to devise fast techniques for computing sparse inner products.

The dataset chosen both by Joachims [67] and by Dumais et al. [36] is a collection of labelled Reuters news stories, the Reuters-21578 dataset, compiled by David Lewis from data of Reuters Newswire in 1987, and publicly available. The specific split of the data that was used is called 'ModApte' and involves a training set of 9603 documents and a test set of 3299 documents to be classified into 90 categories. After preprocessing by word stemming, and removal of the stop-words, the corpus contains 9947 distinct terms that occur in at least three documents. The stories are on average about 200 words long. Examples of categories in this dataset are corporate acquisitions, earning, money market, corn, wheat, ship, and so on, each with a different number of examples. Note that many documents are assigned to several categories.

More experiments were performed by Joachims on the Oshumed (Oregon Health Sciences University) corpus, compiled in 1991 by William Hersh, containing 50 216 documents with abstracts, the first 10 000 being used for training, and the second 10 000 for testing. The task is to assign each document to one or more of 23 categories representing diseases. After preprocessing (stemming and removal of stop-words), the training corpus contains 15 561 distinct terms that occur in at least three documents.

In Joachims' experiments, a simple maximal margin SVM was used, with a heuristic for removing documents whose α_i grew too large and that were therefore regarded as outliers. This strategy was able to replace the need for soft margin techniques because of the particular representation used for the data. Polynomial kernels of low degree and Gaussian kernels have been tested with very positive results. Independently of the choice of kernel parameter, the SVM performed better than standard methods such as naive Bayes, Rocchio, the decision tree algorithm C4.5, and the k-nearest neighbour method.

Remark 8.1 The training algorithm used in these experiments was SVM^{light}, developed by Joachim and freely available over the Internet (see Section 7.8).

Remark 8.2 It is interesting to ask why SVMs were particularly effective in this application. In traditional systems, when the number of features is higher than the training set size, one expects bad generalisation in the same way as was suggested in Chapter 4 for linear functions in high dimensional spaces. This problem can only be overcome if the learning algorithm is able to take advantage of the benign relationship between the target function and the distribution. The SVM maximisation of the margin optimises this advantage, hence overcoming the difficulties inherent in very high dimensional representations.

Remark 8.3 The features used by this approach are low level ones obtained by computing the word frequency in the documents. They only capture global properties of the document. The problem of using more sophisticated kernels with higher level features remains open. A first step would be to consider the string kernel introduced in Example 3.15 in Chapter 3. This kernel will not take into account the order in which the words or substrings occur and is unable to be selective about which words are included, but it works in a much higher dimensional space, this being possible as the feature vector is no longer computed explicitly.

8.2 Image Recognition

Large collections of digital images are becoming freely available over the Internet, or in specialised databases. Furthermore the generation of images has become extremely cheap, and is exploited in several applications. The automatic categorisation of images is a crucial task in many application areas, including Information Retrieval, filtering Internet data, medical applications, object detection in visual scenes, etc. Much of the research in image categorisation focuses on

extraction of high level features from images, for example using edge detection and shape description techniques, in order to capture relevant attributes of the image without increasing the number of features by too much. However, quick information retrieval techniques have also been developed that use only low level features.

Traditional classification approaches perform poorly when working directly on the image because of the high dimensionality of the data, but Support Vector Machines can avoid the pitfalls of very high dimensional representations, as already demonstrated for the case of text in the previous section. A very similar approach to the techniques described for text categorisation can also be used for the task of image classification, and as in that case linear hard margin machines are frequently able to generalise well. In image retrieval tasks, low level features such as histograms of the luminosity or colour levels are often used. They capture global low level properties of the image, conceptually analogous to those used in the text categorisation example. Like word stems they also fail to retain positional information. The distance between two such histograms can be estimated with a χ^2 or other measure of discrepancy between distributions.

In other cases an image is represented directly by its bitmap matrix, which is also the representation used for hand-written digit recognition described in Section 8.3. We begin by considering experiments using this representation.

The use of more sophisticated kernels is largely unexplored and likely to deliver better results. Interaction with practitioners should help in the design of such kernels.

8.2.1 Aspect Independent Classification

A study of the performance of SVMs in these conditions was performed by Pontil and Verri [118], who used SVMs for aspect independent object recognition. They used the simplest way to represent an image as a matrix of bits, or bitmap. Matrices can then be regarded as vector inputs. If the matrices have height h and width w, colour images become vectors of length $3 \times h \times w$, while grey level images have length $h \times w$. Each component of the vector is a pixel in a specific position.

The benchmarking was performed on the COIL (Columbia Object Image Library, available over the Internet) dataset, comprising 7200 images: 72 different views of 100 different three dimensional objects, where each set of 72 views corresponded to 5 degree rotations of the same object. They were transformed into grey level pictures, and their resolution was reduced from 128×128 pixels to 32×32 pixels, by averaging patches of 4×4 pixels. After this preprocessing phase, each image of the dataset was transformed into a vector of 1024 components, each grey level represented by an eight bit number. A simple linear inner product between these vectors was chosen for the kernel, and the basic maximal margin classifier was implemented.

Note that the system does not perform high level feature extraction, but classifies images by viewing them as points in the high dimensional pixel space. The 7200 images were divided into two groups of 3600 images for training and

testing, each group containing one of the subset of 10 degree rotations of each of the 100 images. The inherent multi-class nature of the problem was addressed by training a classifier for each couple of classes and combining them in a special way (see the paper for details). For each experiment, however, only a random subset of 32 of the 100 classes was used. The system was trained on 36 views of the three dimensional objects, and tested on the other 36 views of the objects.

Given the high dimensionality of the feature representation, hard margin hyperplanes with no soft margin and no kernels were sufficient to exactly separate the data. This was also sufficient to always achieve an extremely high performance on the test set. In many cases, only the addition of artificial noise to the data could induce the machine to mistake the predictions. In comparison, standard perceptrons trained in the same conditions exhibited a much worse performance.

8.2.2 Colour-Based Classification

The experiments of [118] relied entirely on pixel grey level information, and neglected important sources of information such as colours. This information can be exploited for classification with SVMs, as demonstrated in a series of experiments by Olivier Chapelle and co-workers [25] that only used colour and luminescence information.

Their approach uses features that somewhat resemble the ones described in the text categorisation section. Based on established information retrieval techniques, they use as a similarity measure between images a distance between their respective histograms of colours. Unfortunately, they do not prove that such kernels satisfy Mercer's conditions. However, it is particularly interesting and illustrative to see domain-specific knowledge used to define a kernel, subsequently applied in a Support Vector Machine. Indeed the same pattern will also be found in the biosequences example.

The kernels are specified by choosing a description of the images, technically a map ϕ from images to feature vectors, and a similarity measure on the feature vectors. As discussed above, the simplest way to represent an image is as a matrix of bits, or bitmap. This representation however is not scale and translation invariant. A slightly more sophisticated representation makes use of histograms of colour. Each colour is a point in a three dimensional colour space. Each image is associated to a histogram, encoding the fraction of pixels of a given colour. The features in this case are a set of colour regions or bins, and the dimensionality of the feature space depends on the size of the bins. An obvious advantage of this representation is that it is invariant with respect to many operations, and can compare images of different sizes. In [25] this is combined with the use of HSV (Hue Saturation Value) instead of RGB (Red Green Blue). The two representations are equivalent, but the first decorrelates the colour components (HS) from the luminance component (V), and is argued to be cognitively more plausible.

In order to specify a kernel, however, one must also decide a measure of similarity between feature vectors. A general class of kernels can be written as follows:

$$K(\mathbf{x}, \mathbf{z}) = \exp\left(-\frac{d(\mathbf{x}, \mathbf{z})}{\sigma^2}\right)$$

where d is some measure of similarity between inputs. For histograms, a possible choice is the χ^2 function, approximated by

$$d(\mathbf{x}, \mathbf{z}) = \sum_{i=1}^{n} \frac{(x_i - z_i)^2}{x_i + z_i}.$$

A second alternative is a simple p-norm

$$d_p(\mathbf{x}, \mathbf{z}) = \left(\sum_{i=1}^{n} |x_i - z_i|^p\right)^{1/p}.$$

The kernel obtained using d_p for $p = 1, 2$ satisfies Mercer's conditions, although it is not known if this is true for the χ^2 choice, which nonetheless makes intuitive sense and performs well in practice. Other types of histograms could also be extracted from the images and used as feature vectors.

An image recognition task on images extracted from the Corel Stock Photo Collection was selected for this application. The Corel dataset contains about 200 categories, each with 100 images. In the experiments, a subset denoted by Corel14 was selected comprising those pictures associated with the following fourteen categories: air shows, bears, elephants, tigers, Arabian horses, polar bears, African speciality animals, cheetahs leopards jaguars, bald eagles, mountains, fields, deserts, sunrises sunsets, and night-scenes. In this reduced dataset, there are 1400 photos. Each category was split into 2/3 for training and 1/3 for testing. The number of bins was fixed at 16 per colour dimension, so that the dimension of the histograms was $16^3 = 4096$. The kernels used are the ones described above, with the 1-norm, 2-norm and χ^2 used as similarity measures.

The results show almost a factor of 2 improvement over k-nearest neighbour, based on the same metric. Interestingly of the three metrics the performances using the 1-norm and χ^2 were very similar, while the 2-norm performed significantly worse in this case. Hence, for this problem the standard choice of a Gaussian kernel was not a good one.

Remark 8.4 Note that in both cases the number of examples was much lower than the feature space dimensionality, but that, despite this, good performance was achieved by maximising the margin. The possibility of using maximal margin machines also derives partly from the high dimensionality of the feature space.

Remark 8.5 A very important application in which SVMs have proven to be particularly effective is object detection in a visual scene. One example is face detection: given as input an arbitrary image, determine whether or not there are any human faces in the image, and if so where they are. The system developed by Osuna et al. [110], works by exhaustively scanning an image for face-like patterns at many possible scales, and uses an SVM as a classifier, to determine whether a given image is a human face. A database containing face/non-face patterns, represented as $19 \times 19 = 361$ pixel vectors, was used to train a soft margin classifier, with polynomial kernels of degree 2. A number of relevant technical details must be considered in this application, which cannot be reported here. An analogous problem involves detecting pedestrians in a visual scene for car-driving applications [108], where wavelets are used as feature extractors before being processed by an SVM with polynomial kernels. Both these applications are very important but they are too complex to describe in detail here. See Section 8.5 for pointers to the original papers.

8.3 Hand-written Digit Recognition

The first real-world task on which Support Vector Machines were tested was the problem of hand-written character recognition. This is a problem currently used for benchmarking classifiers, originally motivated by the need of the US Postal Service to automate sorting mail using the hand-written Zip codes. Different models of SVM have been tested on two databases of digits: USPS (United States Postal Service) and NIST (National Institute for Standard and Technology), both publicly available.

The USPS dataset comprises 7291 training and 2007 test patterns, represented as 256 dimensional vectors (16×16 matrices) with entries between 0 and 255. The NIST dataset, created for benchmarking purposes, contains 60 000 training patterns and 10 000 test patterns, the images are described as 20×20 matrices again with grey level entries.

This problem has been tackled by Vapnik and his co-workers [19], [27], [128], [78], both with maximal margin and with soft margin machines, mainly with polynomial and Gaussian kernels, though sigmoidal kernels have been used despite not satisfying Mercer's conditions. Furthermore, multi-class SVMs have been tested on these data. It is interesting not only to compare SVMs with other classifiers, but also to compare different SVMs amongst themselves. They turn out to have approximately the same performance, and furthermore to share most of their support vectors, independently of the chosen kernel. As expected, as the flexibility of the kernel map increases, the number of support vectors grows, but this happens quite slowly, and within a certain range of kernel parameter the performance appears to be robust. The results of experiments have been reported in a series of papers by Burges, Cortes, Schölkopf, Vapnik, etc., and summarised in [159].

For USPS data, where the input space is 256 dimensional, the following

polynomial and Gaussian kernel were used:

$$K(\mathbf{x}, \mathbf{y}) = \left(\frac{\langle \mathbf{x} \cdot \mathbf{y} \rangle}{256}\right)^d$$

and

$$K(\mathbf{x}, \mathbf{y}) = \exp\left(-\frac{\|\mathbf{x} - \mathbf{y}\|^2}{256\sigma^2}\right)$$

for different values of d and σ.

For polynomial kernels, degrees from 1 to 6 have been tested, for Gaussian kernels, values of σ between 0.1 and 4.0. The kernel parameter certainly affects the generalisation. The USPS are reported to be totally separable with a maximal margin machine starting from degree 3, whereas with degree 1 the training error is 340/7291 and with degree 2 it is 4/7291, using 1-norm soft margin optimisation. The number of support vectors grows very slowly with the degree. For Gaussian kernels, one can use the adaptive kernels algorithm [31] in conjunction with a gradient ascent algorithm like the one described in Section 7.2, automatically scanning the space of possible kernel parameters.

This set of experiments is particularly interesting, because the data have been extensively studied and there are algorithms that have been designed specifically for this dataset, hence incorporating a great deal of prior knowledge about the problem. The fact that SVM can perform as well as these systems without including any detailed prior knowledge is certainly remarkable [78].

8.4 Bioinformatics

8.4.1 Protein Homology Detection

A protein is formed from a sequence of amino-acids from a set of 20, there exist thousands of different proteins, and one of the central problems of bioinformatics is to predict structural and functional features of a protein based on its amino-acidic sequence. This can be done by relating new protein sequences to proteins whose properties are already known, i.e. by detection of protein homologies. In a sense, proteins are clustered in families, which are then grouped together into super-families based on their properties.

Many techniques exist for detecting homologies between proteins, based on their sequence. One of the most common involves constructing a generative model of a protein family from positive examples, and using it to compute the degree of similarity of a new sequence with that family. The parameters $\boldsymbol{\theta}$ of a statistical model, such as a Hidden Markov Model (HMM) $H(\boldsymbol{\theta})$, are estimated using the examples in conjunction with general a priori information about properties of the proteins.

The model $H(\boldsymbol{\theta})$ assigns a probability $P(x|H(\boldsymbol{\theta}))$ to new protein sequence x, and new proteins from the same super-family are expected to have a higher

score than those outside the family. The probability score is hence a measure of homology between a protein and a super-family. We will see that the generative model can also be exploited to produce a kernel. The aim of the experiments of Jaakkola and Haussler [65] was to see if an SVM built with such a kernel could effectively classify protein families into their super-family.

Kernel functions, however, should specify a similarity score between a pair of sequences, whereas the likelihood score from an HMM measures the closeness of the sequence to the model itself. There are, however, some intermediate quantities, computed by HMM systems during the ranking of a new query protein, that can be used to realise a mapping to a feature space, and hence also a kernel. They can be viewed as intermediate representations of the protein in the 'internal language' of the HMM. Without describing HMMs in detail, one of those quantities is the Fisher score, which for a sequence x is given by

$$U_x = \frac{\partial \log P(x|H(\boldsymbol{\theta}))}{\partial \boldsymbol{\theta}}$$

that is the gradient of the log-likelihood of the sequence x with respect to the parameters of the model $H(\boldsymbol{\theta})$. The vector U_x can be viewed as an intermediate representation of the query sequence in the HMM: a representation that reflects the assumptions that went into building the HMM and hence also biological prior knowledge. It is closely related to the vector of sufficient statistics of the probability model that give a complete summary of the sequence in the internal language of the model and it is easily computed as a by-product of the evaluation of the HMM on a given sequence.

It is reasonable to expect that the distance between these vectors provides a measure of similarity between the corresponding sequences. A kernel can then be created using a Gaussian based on the 2-norm

$$K(x, y) = \exp\left(-\frac{\|U_x - U_y\|_2^2}{2\sigma^2}\right),$$

though it might be more natural to use the metric given by the Fisher information matrix. The kernel K is the inner product in the feature space that is the image of the composition of two feature maps

$$x \longmapsto U_x \longmapsto \phi(U_x),$$

where ϕ is the feature map associated with the Gaussian. Hence, this is certainly a Mercer kernel.

This system was implemented using a gradient ascent optimiser similar to the one described in Section 7.2. The positive training examples were a super-family excluding all members of a known family that then formed the test set. Negative examples in both test and training sets came from other super-families.

The system significantly outperformed a number of state-of-the-art protein homology detection systems, proving particularly effective in the detection of remote homologies.

Remark 8.6 It is worth emphasising that again the kernel is constructed using domain knowledge encoded in the HMMs. Many other measures of distance between biological sequences have been proposed, including the use of edit distances, generative grammars, pair-HMMs, and so on. They all aim to exploit prior knowledge of the domain in the design of a similarity measure between inputs.

8.4.2 Gene Expression

Another application of SVMs to biological data mining is given by the automatic categorisation of gene expression data from DNA microarrays. The DNA microarray technology is revolutionising genetic analysis by making it possible to detect what genes are expressed in a particular tissue, and to compare the levels of expression of genes between the tissue in two different conditions. The levels of expression of thousands of genes can now easily be measured by biologists in a single experiment. As the data produced by DNA microarray experiments accumulate, it is becoming essential to have accurate means for extracting biological information and ultimately automatically assigning functions to genes. As an example one would like to use such data to determine whether a new gene encodes a protein of a certain class. Such a classifier could recognise new members of a protein class among genes of unknown function. This kind of classification task is very hard for standard pattern recognition systems as the datasets can be large, noisy and unbalanced, having many more negative than positive examples.

Brown et al. [21] describe a successful use of Support Vector Machines applied to gene expression data both for the task of classifying unseen genes and for the equally important one of cleaning existing datasets. We will not describe here the interesting experimental setup needed to produce microarray data. The experiments of Brown et al. used the following data for computer analysis. A total of 2467 genes of the budding yeast *S. cerevisiae* were represented by a 79 dimensional gene expression vector, and classified according to six functional classes. In most cases, the number of positives was less than 20 (less than 1%), and this, combined with the presence of noise, made the learning task particularly hard: a trivial solution in which all training data are classified as negative was usually found.

The SVM used was a variation of the 2-norm soft margin optimisation described in Chapter 6, where a diagonal element is added to the kernel matrix. In this case, in order to separately control the number of errors in the two classes, the diagonal element added to the kernel depended on the class. This was designed to produce smaller Lagrange multipliers in the dominant negative class, and larger multipliers in the small positive class. The ratio between the elements of the two classes was fixed to be roughly equal to the ratio between the sizes of the two classes. The kernels were chosen to be Gaussian distributions, whose σ was fixed to be roughly equal to the distance between the closest pair of examples of opposite class.

The results obtained were compared with Parzen windows, Fisher's linear

discriminant and the decision tree algorithm C4.5. The SVM outperformed the other methods on all the datasets tested, in many cases by as much as a factor of 2. The system was then used to detect outliers and as a central engine of a data browser for biological data [22].

8.5 Further Reading and Advanced Topics

The applications described in this final chapter are just some examples of the many and diverse uses that have been made of Support Vector Machines in recent years. We have selected particular examples for inclusion in this book on the grounds that they are easily accessible or that they illustrate some important aspect of learning systems design. Many other important applications could not be described due to lack of space, or because of their complexity: these, together with more recent examples, can be accessed via the website [30].

One crucial design choice is deciding on a kernel. Creating good kernels often requires lateral thinking: many measures of similarity between inputs have been developed in different contexts, and understanding which of them can provide good kernels depends on insight into the application domain. Some applications described here use rather simple kernels, motivated by established information retrieval techniques, like the ones for computer vision [129], [17], [118], [25] or the ones for text categorisation [36], [67]. Still, even with such simple kernels and even if the number of features is much larger than the number of training examples, SVMs deliver a very accurate hypothesis, often outperforming other more standard algorithms. Other applications use very sophisticated kernels, like the ones for biosequences discussed in [65], where the kernel is provided by an entire hidden Markov model system. Both Watkins and Haussler have recently proposed elegant kernels for symbol-sequences (discussed in Section 3.7), and their application to problems like biosequence analysis and text categorisation could provide very interesting results.

In the biological data mining example [21], the kernel used was a simple Gaussian kernel, but the difficulty arose in the imbalance of the dataset, a problem that was solved with a technique similar to the 2-norm soft margin. The same technique was also used in another medical application: automatic diagnosis of TBC with Support Vector Machines [168] for separately controlling specificity and sensitivity of the hypothesis. The gene expression work also used another important feature of SVMs: the possibility of using them for data cleaning, as discussed in [55], where the fact that potential outliers can be sought among the support vectors is exploited. The data and experiment are available on the web-site [22].

Other computer vision applications include the works on face detection and pedestrian detection by Tomaso Poggio, Federico Girosi and their co-workers [108], [110], tuberculosis diagnosis [168], the object recognition experiments discussed in [129]. More applications of SVM can be found in the two edited books [132] and [149], and in the website of GMD-FIRST [53] and the others accessible via the website [30].

Tuning the parameters of the system is another important design issue. The heuristics used as an example in the introduction of this chapter for the choice of σ in Gaussian kernels is due to Jaakkola; in general, one can think of schemes that automatically adapt the parameters to the data, based on some learning principle. One such simple scheme is provided by [31].

These references are also given on the website **www.support-vector.net**, which will be kept up to date with new work, pointers to software and papers that are available on-line.