Parabola Volume 42, Issue 2 (2006)

Support Vector Machine Classification

M.P. Wand[1]

Support vector machines emerged in the mid-1990s as a flexible and powerful means of classification. Classification is a very old problem in Statistics but, in our increasingly data-rich age, remains as important as ever. Some examples of classification are:

- classify a patient as high or low risk of prostate cancer based on personal attributes and some medical measurements;
- classify an e-mail message as 'spam' or normal based on features in the text such as the frequency of capital letters;
- classify a person as an employee or intruder at a workplace site based on a retina scan.

One area of classification that has led to an enormous amount of research is computer-aided mail sorting. Figure -2 shows three sets of handwritten digits. Suppose that some of these appeared on the post code of an addressed envelope. How might we get computers to 'read' the post code?

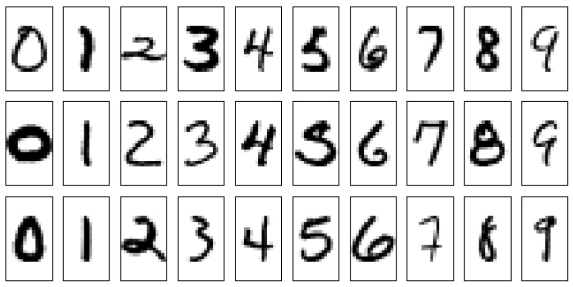


Figure -2: *Three sets of handwritten digits.*

Classification involves the use of *training data* to construct rules for classification of future observations based on their *features*. We can think of training data as points in high-dimensional space, with the points coloured according to the known class. An example of a set of training data is given in the left panel of Figure -1. The points correspond to age in years and prostate weight (on a logarithm scale) for 96 men. The black points correspond to men with high risk of prostate cancer. Those with low risk are shown as white points. Classification is concerned with using these data to help diagnose future male patients as high or low risk for prostate cancer.

[1]M.P. Wand is a Professor of Statistics, School of Mathematics and Statistics, University of New South Wales.

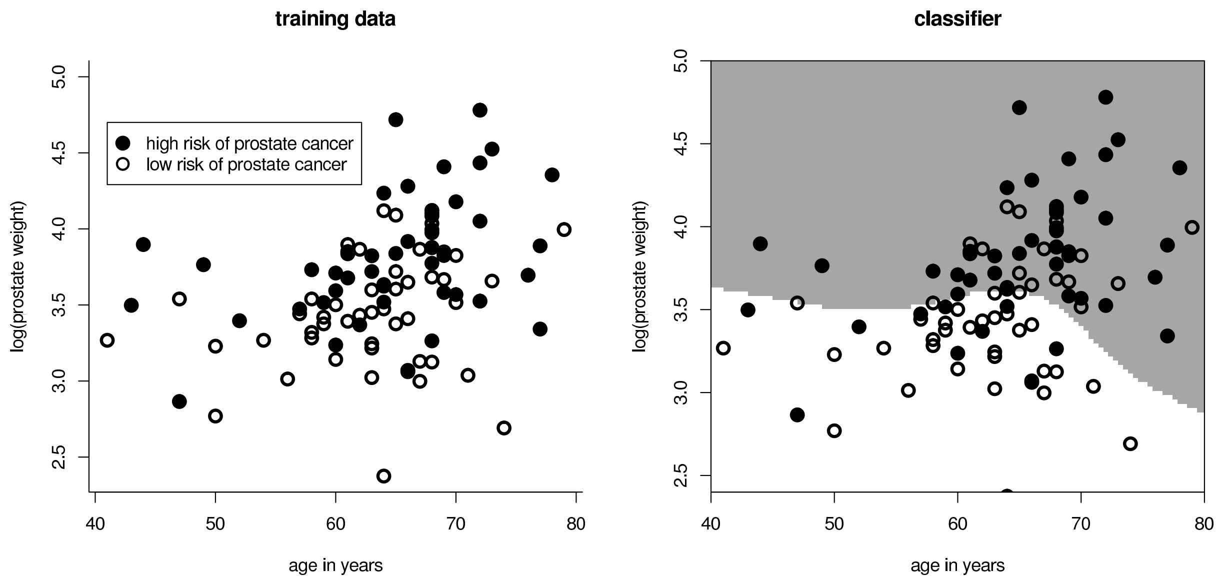


Figure -1: *The left panel shows an example of training data set. There are two features:* age *and* log(prostate weight). *The problem is to classify future male patients as high or low risk of prostate cancer using these training data. The right panel shows an example of a classifier. Future patients with measurements landing in the grey region are classified as 'high risk of prostate cancer'.*

The right panel of Figure -1 depicts a classifier for these data. A future male patient whose age and log(prostate weight) measurements land in the grey region will be classified as 'high risk of prostate cancer'. If those measurements are in the white region then he will be classified as 'low risk of prostate cancer'. Clearly the classifier rule isn't perfect. If it were applied to the 96 men in the training data then 23 men (about 24%) would be *misclassified*. If more features are added (e.g. cholesterol level, average number of cigarettes per day) then we might be able to lower misclassification percentages.

Support vector machines are an effective means of dealing with such classification problems. They are also relatively simple. Indeed, the two ingredients of support vector machines: *maximum margin lines* (and their extension to higher-dimensions) and *kernelisation*, can be
largely explained using high school-level mathematics.

Maximum margin lines. Figure 0 shows a small two-dimensional training data set. Note that we are using x_1 and x_2, rather than x and y, for the coordinate axes. This notation extends more readily to higher dimensions. The two classes correspond to the black and white colouring. We will now describe a method, known as *maximum margin lines*, for classifying future points as either 'black' or 'white'.

Suppose that we restrict the classifier to the family of straight lines, then Figure 1 shows three lines that provide perfect separation. Clearly there are an infinite number

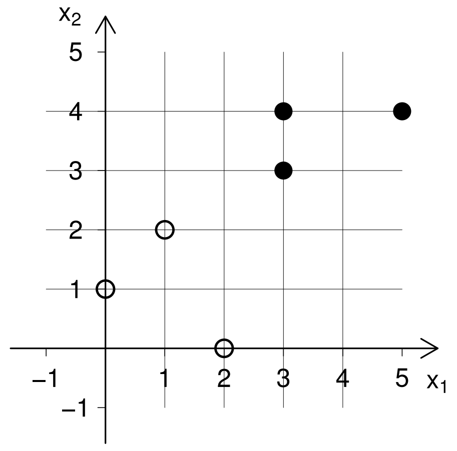


Figure 0: *A small two-dimensional training data set.*

of such lines, but which one is 'best'? One way to define 'best' is via the concept of *maximising the margin*. For any separating line we can form a *margin region* by considering rectangles that are bisected by the line and do not contain any of the training data points. We then find the line that produces the widest rectangle.

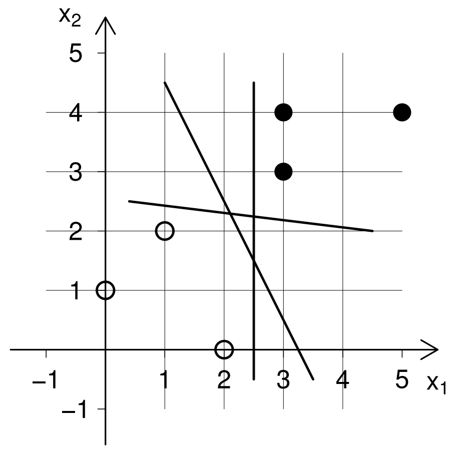


Figure 1: *Three lines that separate black and white points.*

The answer turns out to be the line

$$x_2 = 6.5 - 2x_1 \tag{1}$$

shown in Figure 2 along with the maximum margin rectangle, shown in grey. The *margin*, denoted by M, is the width of the rectangle and is equal to $\sqrt{5}$ in this case. We call (1) the *maximum margin line*. The maximum margin rectangle is "supported" on the 3 circled points. The points are known as *support vectors*, which is why the classification method being described in this article is so-named. For this simple example you could obtain the maximum margin line using eye-sight and trial and error. But how do you get a computer to find this line? What about the situation when there are ten times as many points? Finally, we want to be able to extend the concept of maximum margin lines to higher dimensions, where they are called *maximum margin hyperplanes*. There visualisation of the training data is difficult or impossible. We therefore require a mathematical solution to the maximum margin problem.

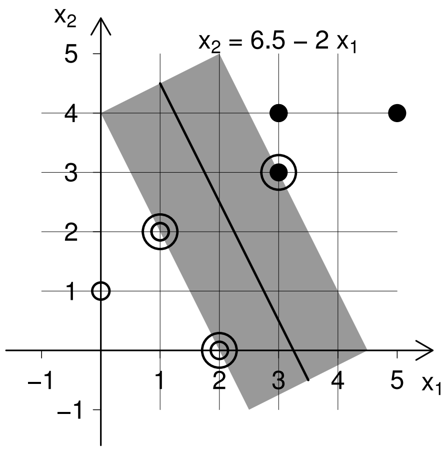


Figure 2: *The maximum margin line for the data of Figure 0. The circled points are the support vectors.*

We begin with:

Fundamental Result from Analytic Geometry

The *signed distance* from the point (x_1^*, x_2^*) to the line

$$A + B\,x_1 + C\,x_2 = 0$$

is

$$\frac{A + B\,x_1^* + C\,x_2^*}{\sqrt{B^2 + C^2}}.$$

Remark 1: The signed distance is such that points on opposite sides of the line have opposite signs. However, the actual signs depend on the choice of A, B and C. If they are multiplied through by a negative number then the positive distances become negative and vice versa.

□

Remark 2: Note that in the general line formulation

$$A + B\,x_1 + C\,x_2 = 0$$

A, B and C are only defined up to a scalar multiplication. For example, the line

$$3 - 7x_1 + 4x_2 = 0$$

can also be expressed as

$$15 - 35x_1 + 20x_2 = 0$$

or

$$-6 + 14x_1 - 8x_2 = 0.$$

$\square$

Let

$$A + B\,x_1 + C\,x_2 = 0$$

be a general line separating the points in Figure 0 and let $M = M(A, B, C)$ denote the margin of the line, so that

$$M/2 = \text{half the margin}.$$

Label the points as follows:

$$\left.\begin{array}{l}(x_{11}, x_{12}) = (5,4)\\ (x_{21}, x_{22}) = (3,4)\\ (x_{31}, x_{32}) = (3,3)\end{array}\right\}\text{ black points}$$

$$\left.\begin{array}{l}(x_{41}, x_{42}) = (1,2)\\ (x_{51}, x_{52}) = (2,0)\\ (x_{61}, x_{62}) = (0,1)\end{array}\right\}\text{white points.}$$

We will suppose without loss of generality (courtesy of Remark 1) that the A, B and C are such that black points have negative signed distance to the line. Then from the above result:

$$\frac{A + B\,x_{i1} + C\,x_{i2}}{\sqrt{B^2 + C^2}} \leq -\frac{M}{2}, \quad 1 \leq i \leq 3.$$

The white points must then have positive signed distance to the line and from the above result:

$$\frac{A + B\,x_{i1} + C\,x_{i2}}{\sqrt{B^2 + C^2}} \geq \frac{M}{2}, \quad 4 \leq i \leq 6.$$

Let y_i, where $1 \leq i \leq 6$, denote the class labels as follows:

$$y_i = \begin{cases} -1, & 1 \leq i \leq 3 \quad \text{(black points)} \\ 1, & 4 \leq i \leq 6 \quad \text{(white points).} \end{cases}$$

Then the previous two inequalities can be combined into the set of constraints

$$\frac{y_i(A + B\,x_{i1} + C\,x_{i2})}{\sqrt{B^2 + C^2}} \geq \frac{M}{2}, \quad 1 \leq i \leq 6.$$

The problem is to maximise $M = M(A, B, C)$ subject to these constraints. Formally, we have the *constrained optimisation problem*:

$$\begin{aligned} &\text{find } A, B \text{ and } C \text{ to maximise } M = M(A, B, C) \\ &\text{subject to} \quad \frac{y_i(A + B\,x_{i1} + C\,x_{i2})}{\sqrt{B^2 + C^2}} \geq \frac{M}{2} \text{ for all } 1 \leq i \leq 6. \end{aligned} \tag{2}$$

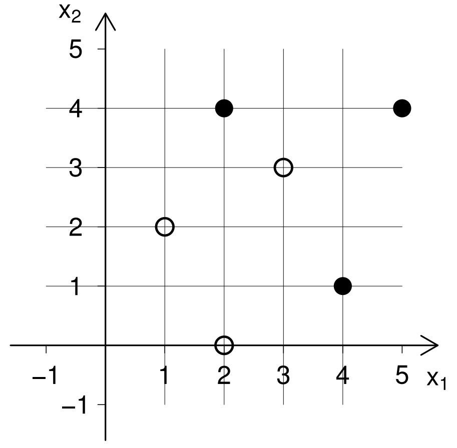


Figure 3: *An example two-dimensional training data set where the points are not separable by a straight line.*

Since, as mentioned in Remark 2, A, B and C can be arbitrarily rescaled we can choose them to satisfy

$$\sqrt{B^2+C^2} = 2/M.$$

Then $M = 2/\sqrt{B^2+C^2}$ and maximising M is equivalent to minimising B^2+C^2. So the constrained optimisation problem becomes

$$\text{find } A, B \text{ and } C \text{ to minimise } (B^2+C^2)$$

$$\text{subject to} \quad y_i(A + B\,x_{i1} + C\,x_{i2}) \geq 1 \text{ for all } 1 \leq i \leq 6.$$

This can be solved using the optimisation technique known as *quadratic programming*. Many popular programming languages, such as `Matlab` and `R`, have quadratic programming capability these days.

In practice it is more common that the black and white points cannot be separated by a straight line. The data depicted in Figure 3 is of this type. Figure 4 shows an example line and corresponding margin region for the data of Figure 3. The dotted lines in Figure 4 correspond to the the non-zero values of

$$\xi_i = \text{extent to which the } i\text{th point is `on the wrong side'} \\ \text{of the margin region boundary as a proportion of } M.$$

The maximum margin problem in this non-separable case is then of the form:

$$\max_{\beta_i,\xi_i} M = M(A,B,C,\xi_1,\ldots,\xi_6)$$

$$\text{subject to} \quad \frac{y_i(A + B\,x_{i1} + C\,x_{i2})}{\sqrt{B^2+C^2}} \geq \frac{M(1-\xi_i)}{2}, \tag{3}$$

$$\xi_i \geq 0 \text{ for all } 1 \leq i \leq 6 \quad \text{and} \quad \textstyle\sum_{i=1}^{6} \xi_i < c.$$

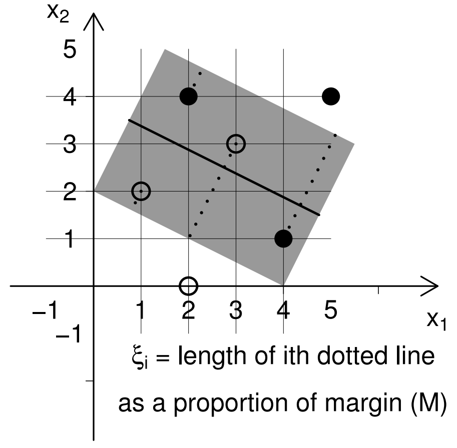


Figure 4: *An example line and margin with the ξ_i's denoting the extent to which some of the points are 'on the wrong side' of the margin boundary.*

Here $c > 0$ is a tuning parameter. It sometimes referred to as a *cost* parameter since it controls the degree of violation ('cost') of separability by the maximum margin line. We can also convert this to a quadratic programming problem and solve for A, B and C for any given value of $c > 0$.

In closing, it should be noted that while we have just treated two specific problems with 6 points the extension to n points is quite straightforward. Similarly, the extension to higher dimensions and maximum margin hyperplanes involves algebra not much more involved than that given here. This results in what are now called *linear support vector machine* classifiers.

Kernelisation. Now consider the points plotted in Figure 5. Clearly the black and white points are not well-separated by any straight line. However, as shown in Figure 6, they can be separated by an ellipse of the form:

$$\frac{x_1^2}{a^2} + \frac{x_2^2}{b^2} = 1. \tag{4}$$

How might we come up with a 'good' separating ellipse for these data? In particular, can we use the maximum margin technology from the previous section? It turns out that we can – by transforming the training data to a new coordinate space.

Introduce the new variables z_1 and z_2:

$$z_1 = x_1^2 \qquad \text{and} \qquad z_2 = x_2^2.$$

Then (4) becomes

$$\frac{z_1}{a^2} + \frac{z_2}{b^2} = 1$$

which corresponds to the family of (negatively sloped) straight lines in the (z_1, z_2) co-ordinate space. Figure 7 shows the points in this transformed space and separability

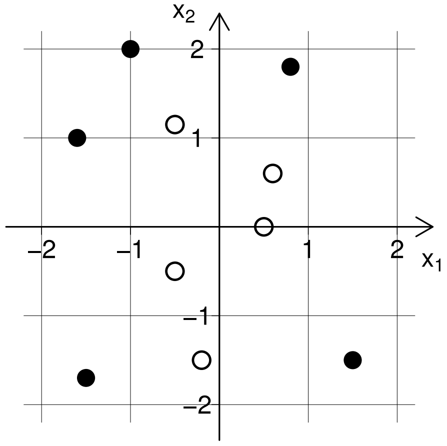


Figure 5: *A training data set which is not naturally separable by a straight line.*

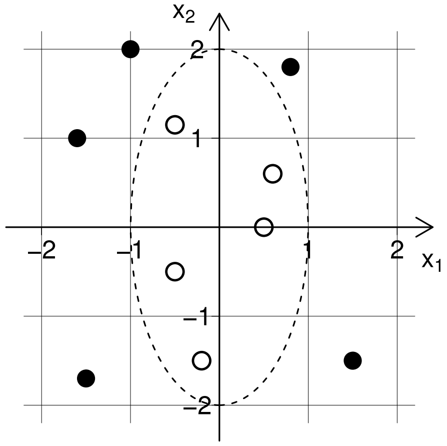


Figure 6: *An ellipse that separates the black and white points.*

by straight lines is apparent. If we now apply the maximum margin line methodology from the previous section to these data then we obtain the result shown in Figure 8.

Transforming back to the original space leads to the ellipsoidal separator shown in Figure 9. Notice that the margin region becomes an 'elastic band'-like region in the (x_1, x_2) space.

The process by which the points are transformed to a new coordinate system, have a maximum margin line (or hyperplane) fitted to them, which is then transformed back to the original space allows for complex non-linear separating boundaries. It is known as *kernelisation* and the remainder of this section will be concerned with explaining the reason for this name.

Let $(x_{11}, x_{21}), (x_{12}, x_{22}), \ldots, (x_{1n}, x_{2n})$ be a general two-feature set of training data of size n. It turns out that the maximum margin line algorithm of the previous section depends on these training data only through expressions of the form

$$x_{1i}x_{1j} + x_{2i}x_{2j},$$

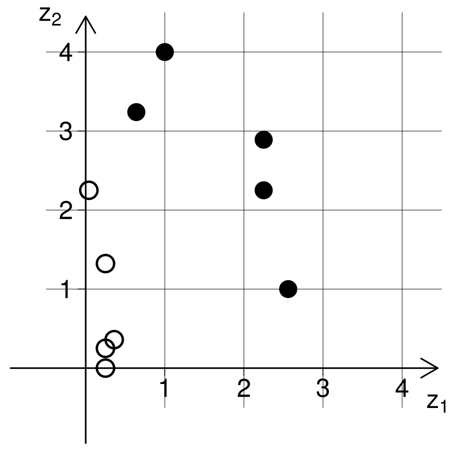


Figure 7: *The training data in the* (z_1, z_2) *space.*

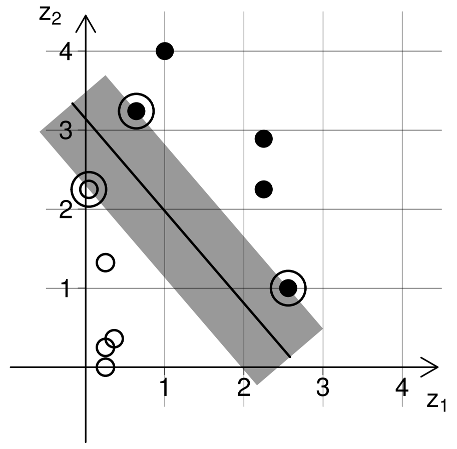


Figure 8: *The maximum margin line for the training data in the* (z_1, z_2) *space. The circled points are the support vectors.*

known as the *inner product* of the pair $\mathbf{x}_i = (x_{1i}, x_{2i})$ with the $\mathbf{x}_j = (x_{1j}, x_{2j})$, where i and j range over $1, \ldots, n$. So the maximum margin line for the z_1, z_2 data (Figure 8) depends only on the z_{1i} and z_{2i} through the inner products

$$z_{1i}z_{1j} + z_{2i}z_{2j} = x_{1i}^2 x_{1j}^2 + x_{2i}^2 x_{2j}^2 = K(\mathbf{x}_i, \mathbf{x}_j)$$

where, for general pairs $\mathbf{s} = (s_1, s_2)$ and $\mathbf{t} = (t_1, t_2)$,

$$K(\mathbf{s}, \mathbf{t}) = s_1^2 t_1^2 + s_2^2 t_2^2.$$

The bivariate function K is known as a *kernel* function and characterises the transformation from the (x_1, x_2) space to the (z_1, z_2) space. However, note that the maximum margin line algorithm depends only on the values of:

$$K(\mathbf{x}_i, \mathbf{x}_j) \quad \text{for} \quad i, j = 1, \ldots, n.$$

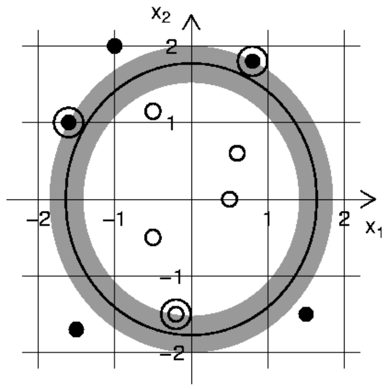


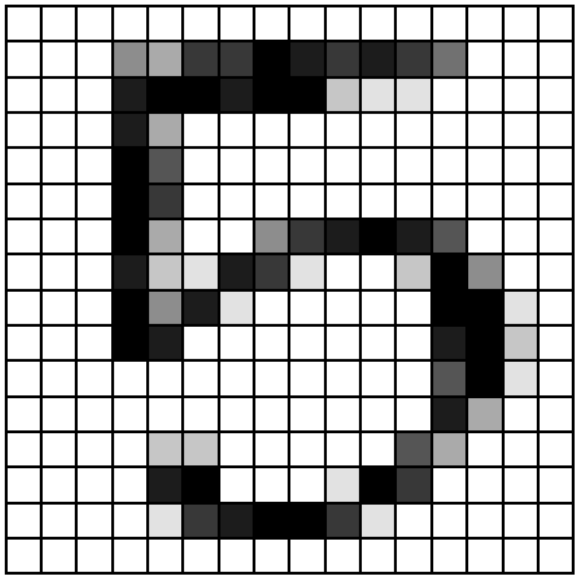
Figure 9: *The classifier and margin region from Figure 8 when transformed back to the* (x_1, x_2) *space. The circled points are the support vectors.*

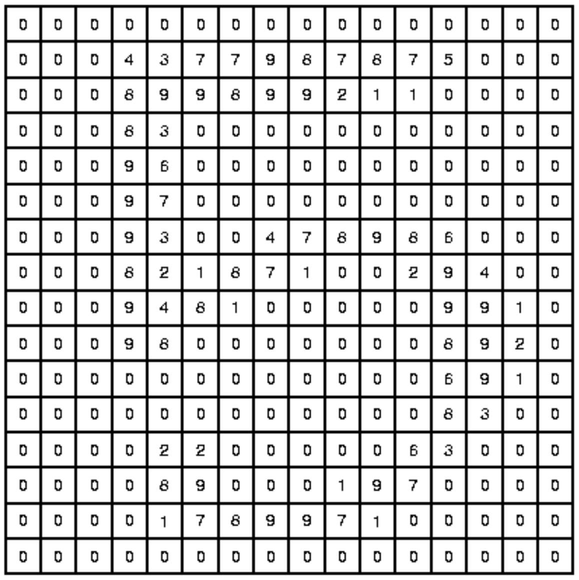
We do not have to explicitly work with the (z_{1i}, z_{2i}) values. This opens up the possibility of using other bivariate functions with the idea that they will allow more complex separating boundaries. Some kernel functions that are commonly used in practice are:

Radial basis:	$K(\mathbf{s}, \mathbf{t}) = e^{-\gamma \sum_{\ell=1}^{2} (s_{1\ell} - t_{2\ell})^2}$
pth degree polynomial:	$K(\mathbf{s}, \mathbf{t}) = (1 + \sum_{\ell=1}^{2} s_{1\ell} t_{2\ell})^p$

The radial basis kernel is the most popular. It corresponds to transforming from (x_1, x_2) to an 'infinite dimensional' space.

Computer-aided Mail Sorting. We now return to the problem of computer-aided mail sorting. Note, again, the handwritten digits in Figure -2. How can we teach a computer to distinguish between a handwritten '5' and a handwritten '6', say? The first step is to turn a handwritten digit into a numerical object. This can done by scanning the digit into a computer and obtaining a *grey level* representation. The left panel of Figure 10 shows such a representation. The scanning process has produced a 16×16 array of *pixels* of shades of grey. The next step is to convert the shades of grey to numbers. A common convention is to use the integers 0,1,...,255 where 0=white and 255=black. For illustration purposes we will use the simpler scale: 0=white and 9=black. This leads to the 16×16 array of numbers shown in the right panel of Figure 10.





0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	4	3	7	7	9	8	7	8	7	5	0	0	0
0	0	0	8	9	9	8	9	9	2	1	1	0	0	0	0
0	0	0	8	3	0	0	0	0	0	0	0	0	0	0	0
0	0	0	9	6	0	0	0	0	0	0	0	0	0	0	0
0	0	0	9	7	0	0	0	0	0	0	0	0	0	0	0
0	0	0	9	3	0	0	4	7	8	9	8	6	0	0	0
0	0	0	8	2	1	8	7	1	0	0	2	9	4	0	0
0	0	0	9	4	8	1	0	0	0	0	0	9	9	1	0
0	0	0	9	8	0	0	0	0	0	0	0	8	9	2	0
0	0	0	0	0	0	0	0	0	0	0	0	6	9	1	0
0	0	0	0	0	0	0	0	0	0	0	0	8	3	0	0
0	0	0	0	2	2	0	0	0	0	0	6	3	0	0	0
0	0	0	0	8	9	0	0	0	1	9	7	0	0	0	0
0	0	0	0	1	7	8	9	9	7	1	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 10: *Grey level representation of a handwritten '5', over a 16×16 pixel array and its corresponding numerical (or digital) representation.*

The *vector* representation of the handwritten '5' is then the 256-tuple:

$$(0, 0, 0, \ldots, 0, 4, 3, 7, 7, 9, 8, 7, 8, 7, 5, 0, 0, \ldots, 0, 0, 0).$$

which lives in 256-dimensional space. It has been reported in the literature that 9th-degree polynomial kernels have lead to good classification performance for handwritten digits. This kernel is given by

$$K(\mathbf{s}, \mathbf{t}) = \left(1 + \sum_{\ell=1}^{256} s_\ell t_\ell\right)^9.$$

If

$$\mathbf{x}_1 = (x_{11}, x_{12}, \ldots, x_{1,256}).$$

is a digital version of a handwritten '5' then $y_1 = -1$. If

$$\mathbf{x}_2 = (x_{21}, x_{12}, \ldots, x_{2,256})$$

is a digital version of a handwritten '6' then $y_2 = 1$. Then the support vector machine requires

$$K(\mathbf{x}_1, \mathbf{x}_2) = \left(1 + \sum_{\ell=1}^{256} x_{1\ell} x_{2\ell}\right)^9 .$$

Suppose that we have available 1000 such pairs:

$$(\mathbf{x}_i, y_i), \quad i = 1, \ldots, 1000.$$

Then we repeat the above calculation to obtain

$$K(\mathbf{x}_i, \mathbf{x}_j), \quad i, j = 1, \ldots, 1000.$$

These values are then fed into a quadratic programming algorithm arising from the maximum margin line (or hyperplane) problem described earlier. A support vector machine classifier for distinguishing a '5' from a '6' results. A computer can then use this to classify future images as a '5' or '6'. Researchers are now getting misclassification rates as low as 0.8% using support vector machine classification with the 9th-degree polynomial kernel. Thanks to mathematics, mail sorting is becoming faster, less tedious and highly accurate!

Further reading

If you would like to read further on classification methods, and the role played by mathematics, then a recommended book is *The Elements of Statistical Learning* by Trevor Hastie, Robert Tibshirani & Jerome Friedman (2001, Springer-Verlag). All three authors are Statistics professors at Stanford University in California, USA, and they have gone to considerable trouble to make this area of research accessible to a wider audience.