T-61.231 Principles of Pattern Recognition

Answers to exercise 6: 4.11.2002

1. The SVM optimal hyperplane separates the space so that

$$\omega^T x_i + \omega_0 \geq +1, \text{ if } x_i \in \omega_1$$
$$\omega^T x_i + \omega_0 < -1, \text{ if } x_i \in \omega_2$$

Let ω_1 be on the positive side of the optimal hyperplane and ω_2 on the negative side, and d_+ and d_- be the distances from the optimal hyperplane and the nearest point in classes ω_1 and ω_2, respectively. Let $g(x) = \omega^T x_i + \omega_0$ be the distance from the optimal hyperplane ω. It can also be stated that

$$x = x_p + r\frac{\omega}{||\omega||}$$

where x_p is the projection of x onto the optimal hyperplane and r is the distance from the hyperplane. Since $g(x_p) = 0$ by definition (the point x_p lies on the optimal hyperplane),

$$g(x) = \omega^T x + \omega_0 = r||w|| \Leftrightarrow r = \frac{g(x)}{||\omega||}$$

Thus the algebraic distance for the support vectors is

$$r = \frac{g(x)}{||\omega||} = \begin{cases} \frac{1}{||\omega||} = d_+, \text{ when } x \text{ is the nearest point of } \omega_1 \\ -\frac{1}{||\omega||} = d_-, \text{ when } x \text{ is the nearest point of } \omega_2 \end{cases}$$

Here the negative sign denotes being on the negative side of the hyperplane. Thus the margin between the two classes is $\frac{2}{||\omega||}$.

2. The main idea behind finding the optimal SVM decision hyperplane is to maximize the marginal $\frac{2}{||\omega||}$. In the basic, separable case this is done through taking positive (because of the form $\ldots \geq 0$) Lagrange multipliers $\alpha_i, i = 1, \ldots, l$, where l is the number of points, for each inequity

$$y_i(\omega^T x_i + \omega_0) - 1 \geq 0$$

where y_i denotes class membership, $y_i = 1$ if $x_i \in \omega_1$ and $y_i = -1$ if $x_i \in \omega_2$. Thus the objective function to minimize is

$$L_P = \frac{||\omega||^2}{2} - \sum_{i=1}^{l} \alpha_i y_i(\omega^T x_i + \omega_0) + \sum_{i=1}^{l} \alpha_i$$

The objective is to minimize L_P with respect to ω and ω_0 and simultaneously require that the derivatives of L_P with respect to all α_i vanish, all subject to the constraints $\alpha_i \geq 0$. This is a convex quadratic programming problem, since both the objective function is convex and the points satisfying the constraints form a convex set. This means that it is equivalently possible to solve the dual problem, maximize L_P subject to the constraints that the gradient of L_P with respect to ω and ω_0 vanish and again all $\alpha_i \geq 0$. Requiring the gradient of L_P to vanish with respect to ω and ω_0 gives the additional constraints:

$$\frac{\delta L_P}{\delta \omega} = \omega - \sum_i \alpha_i y_i x_i = 0 \Rightarrow \omega = \sum_i \alpha_i y_i x_i$$
$$\frac{\delta L_P}{\delta \omega_0} = -\sum_i \alpha_i y_i = 0 \Rightarrow \sum_i \alpha_i y_i = 0$$

By substituting these conditions into the equation for L_P we obtain

$$\begin{aligned} L_D &= \tfrac{1}{2}(\sum_i \alpha_i y_i x_i)^2 + -(\sum_i \alpha_i y_i x_i)^2 - 0 * w_0 + \sum_i \alpha_i \\ &= \sum_{i=1}^{l} \alpha_i - \tfrac{1}{2}\sum_{i,j} \alpha_i \alpha_j y_i y_j x_i \cdot x_j \end{aligned}$$

Both formulations produce the same result. The latter formulation is called the Wolfe dual.

For the non-separable case the basic algorithm provides no feasible solution as the objective function grows arbitrarily large. In order to handle the non-separable case an additional cost must be introduced to loosen the original constraints when necessary. This can be done by introducing positive slack variables $\xi_i \geq 0, i = 1, \ldots, l$ into the constraints, which then become

$$\begin{aligned} \omega^T x_i + \omega_0 \geq 1 - \xi_i, \text{ if } x \in \omega_1 \\ \omega^T x_i + \omega_0 \leq -1 + \xi_i, \text{ if } x \in \omega_2 \end{aligned}$$

Thus for an error to occur $\xi_i > 1$, so $\sum_i \xi_i$ is an upper bound for training errors. The costs can be added to the objective function so that the objective function becomes $\frac{||\omega||^2}{2} + C(\sum_i \xi_i)^k$, where C is a cost parameter to be freely chosen (larger C is equivalent to a larger cost for making a mistake). It can be seen that when $k = 1$ L_P becomes

$$L_P = \frac{||w||^2}{2} + C\sum_i \xi_i - \sum_i \alpha_i[y_i(x_i \cdot \omega + \omega_0) - 1 + \xi_i] - \sum_i \mu_i \xi_i$$

In this case neither ξ_i nor their Lagrange multipliers μ_i appear in the Wolfe dual L_D, which can be seen by requiring the gradient of L_P to vanish with respect to ω, ω_0 and all ξ_i:

$$\begin{aligned} \frac{\delta L_P}{\delta \omega} &= \omega - \sum_i \alpha_i y_i x_i = 0 \\ \frac{\delta L_P}{\delta \omega_0} &= -\sum_i \alpha_i y_i = 0 \\ \frac{\delta L_P}{\delta \xi_i} &= C - \alpha_i - \mu_i = 0 \end{aligned}$$

By substituting these into the equation for L_P we get the Wolfe dual for the non-separable case

$$\begin{aligned} L_D &= \tfrac{1}{2}(\sum_i \alpha_i y_i x_i)^2 + \sum_i(\alpha_i + \mu_i)\xi_i - (\sum_i \alpha_i y_i x_i)^2 - 0 * b + \sum_i \alpha_i - \sum_i \alpha_i \xi_i - \sum_i \mu_i \xi_i \\ &= -\tfrac{1}{2}(\sum_i \alpha_i y_i x_i)^2 + \sum_i \alpha_i \\ &= \sum_i \alpha_i - \tfrac{1}{2}\sum_{i,j} \alpha_i \alpha_j y_i y_j x_i \cdot x_j \end{aligned}$$

So the problem is to maximize L_D subject to the constraints $0 \leq \alpha_i \leq C$ and $\sum_i \alpha_i y_i = 0$, and $w = \sum_{i=1}^{N_s} \alpha_i y_i x_i$, where N_s is the amount of support vectors. As we can see, the form of L_D is actually identical to that of the separable case. The only difference is in the constraints.

So, in the situation where we have the points $x_1 = [1 \quad 1]^T \in \omega_1$, $x_2 = [2 \quad 1]^T \in \omega_2$, $x_3 = [3 \ \ 2]^T \in \omega_1$ and $x_4 = [2 \ \ 3]^T \in \omega_2$, the dual problem L_D can be written as

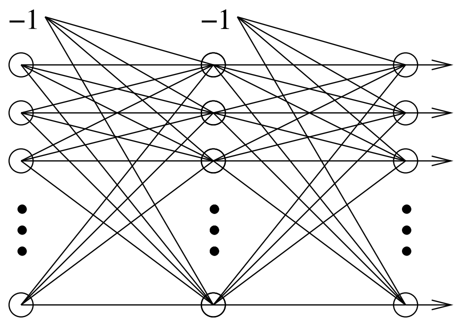
$$\begin{aligned} L_D &= \textstyle\sum_i \alpha_i - \frac{1}{2}\sum_{i,j} \alpha_i\alpha_j y_i y_j x_i \cdot x_j \\ &= \alpha_1 + \alpha_2 + \alpha_3 + \alpha_4 - \tfrac{1}{2}(2\alpha_1^2 + 5\alpha_2^2 + 13\alpha_3^2 + 13\alpha_4^2 - 6\alpha_1\alpha_2 + 10\alpha_1\alpha_3 \\ &\quad -10\alpha_1\alpha_4 - 16\alpha_2\alpha_3 + 14\alpha_2\alpha 4 - 24\alpha_3\alpha_4) \end{aligned}$$

and the constraints as

$$\alpha_1 - \alpha_2 + \alpha_3 - \alpha_4 = 0$$
$$0 \leq \alpha_i \leq C \quad \forall i$$

where C is the cost parameter to be chosen.

3. **a)** The total amount of weights is the same as the total amount of connections, including the biases. So with the network containing N_x input neurons, N_h hidden layer neurons and N_y output neurons and being fully connected, as is shown in the figure, the total amount of weights is



$$N_w = (N_x + 1)N_h + (N_h + 1)N_y = N_h(N_x + N_y + 1) + N_y$$

b) $N_y = 10$ means that the number of character classes is ten and one output neuron has been assigned to each class, and the output of the network can be seen as a posteriori distribution of the classes. Thus the result is the class whose neuron shows the most activation.

A logical number of input neurons for a 16×16 is $N_x = 16 * 16 = 256$.

Widrow rule is written as

$$N_t \geq 10\frac{N_w}{N_y}$$

In this case, since $N_y = 10$, this simplifies to $N_t = N_w$. The ratio at the limit can be explored from two directions,

1. $N_t = N_w = N_h(N_x + N_y + 1) + N_y \Rightarrow \frac{N_h}{N_t} = \frac{1-10/N_t}{267}$

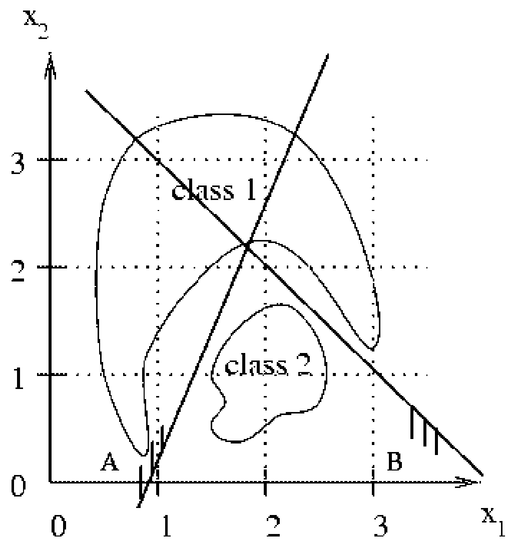
or

2. $\frac{N_h}{N_t} = \frac{N_h}{N_w} \Rightarrow \frac{N_h}{N_t} = \frac{1}{267+10/N_h}$

So if $N_h >> 1$ or $N_t >> 1$ the limit approaches $\frac{1}{267}$.

It should be noted that Widrows rule provides some kind of an estimate to the suitable network size; there is no proven analytic way of knowing the "right" network size for a given problem. But this estimate gives a reasonable value to start with.

4. A single perceptron can divide the feature space into two different parts based on which side of the decision line the point resides. Thus, two perceptrons can divide the feature space into four parts. So lets draw two decision lines into the figure as shown.



From the figure it can be estimated that for example the functions

$$x_2 = 2.5x_1 - 2.5$$
$$x_2 = -x_1 + 4$$

i.e.

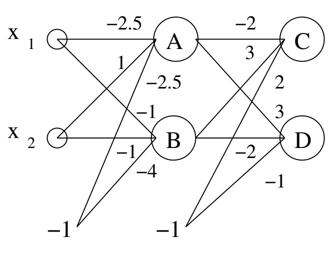
$$-2.5x_1 + x_2 + 2.5 = 0$$
$$-x_1 - x_2 + 4 = 0$$

could separate the areas.

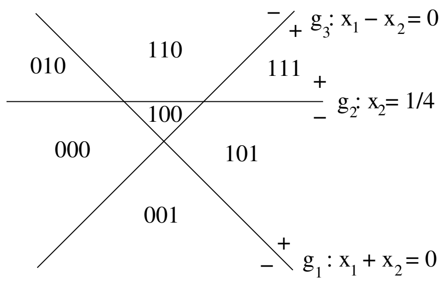
Now, all the samples of class 2 are in the area $C = \neg A \cap B$ and all the samples of class 1 are in $D = \neg(\neg A \cap B)$, with A lying above the first line and B below the second one as indicated with the short marks in the image.

A	B	C	D
0	0	0	1
0	1	1	0
1	0	0	1
1	1	0	1

Thus the network weights would be

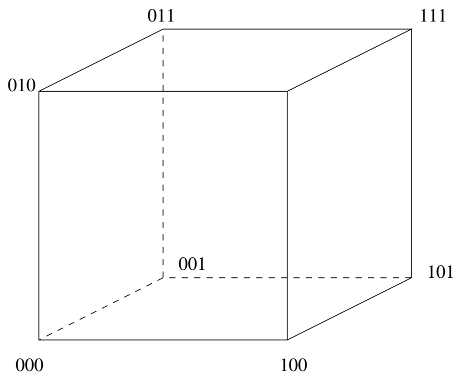


5. First, lets draw the lines together and mark which side of the line the equation gives positive values on. Then we go through each sector of the space separated by these lines and give them a binary code indicating which side of each of the lines it is on, for example 001 is on the negative side of g_1 and g_2 but on the positive side of g_3, the ith bit being one if the area resided on the positive side of g_i and zero if it resides on the negative side.

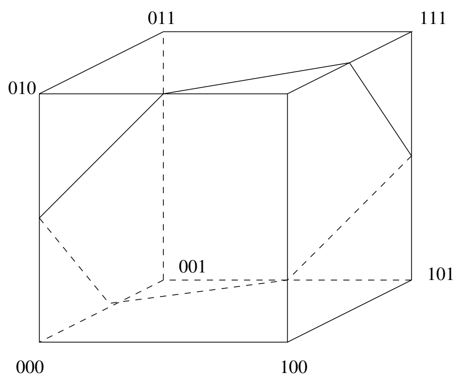


Now we can also see that no section received the value 011, this is perfectly normal. 011 is called a virtual polyhedra, which are quite common when dividing the two-dimensional space with multiple lines that do not all intersect at the same position.

Now, each of the areas is mapped onto the vertice of a three-dimensional cube.

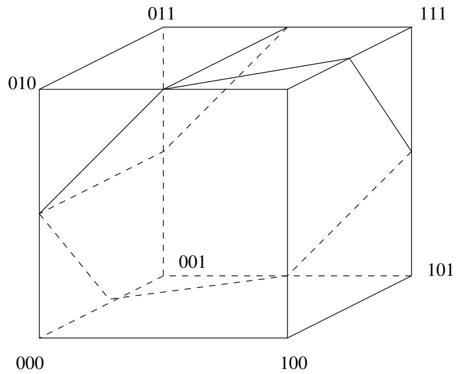


a) For the regions to be possible to classify with a two-layer network, it is necessary that the vertices of the cube that we want to join are linearly separable (in the cube projection). Thus we can choose for example the vertices 000, 100, 101 and 110. We can insert a plane of, for example, the form $y_1 - y_2 - y_3 + 1/2 = 0$, with the positive side being in the direction of 100, to separate these, as is illustrated in the image.



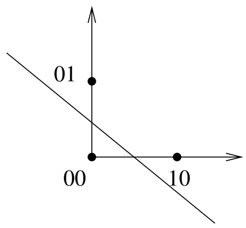
Thus we obtain the second-layer weight vector to be $\omega_{21} = [1\ -1\ -1; -1/2]^T$.

b) A three-layer network can separate any union of N_h vertices, where N_h is the amount of hidden-layer neurons. Thus it is possible to separate for example the vertices 000, 100, 101, 110, 010 and 011. (Or looking at it the other way around, the remaining vertices of 001 and 111, which just happens to be the same number as the intended amount of hidden layer neurons). This can be accomplished for example by using the plane $-y_1 + y_2 - 1/2 = 0$, the positive side is in the direction of 010 and 011.



Now we obtain the second second-layer weight vector of $\omega_{22} = [-1\ 1\ 0; 1/2]^T$.

Further, we must now combine these hidden layer outputs. This can again be visualized by plotting the possible outputs 10, 00 and 01 onto a two-dimensional space and extracting a line to separate them. (Note again, that the situation 11 is now impossible; this is clear by looking at the planes and the cube vertices.)



Now these can be separated with for example the line $z_1 + z_2 + 1/2 = 0$, resulting in the final weight vector of $\omega_3 = [1\ 1\ -1/2]^T$.