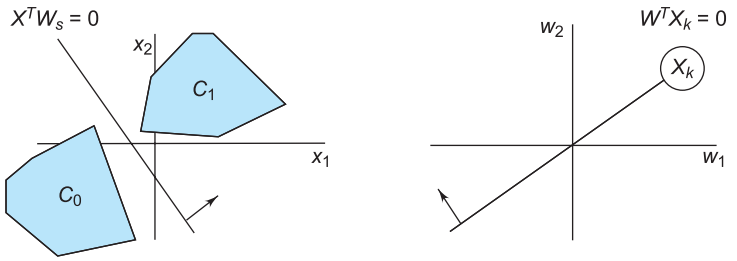
5.6 Pattern Space and Weight Space

First consider the following. The activation of the neuron $y = X^T W_S$ is the inner product of vectors X and W_S. In pattern space, points that satisfy $X^T W_S = 0$ define a separating hyperplane as illustrated in Fig. 5.9(a) for the two dimensional case. Pattern space points on one side of this hyperplane (with an orientation indicated by the arrow in Fig. 5.9(a)) yield positive inner products with W_S and thus generate a +1 neuron signal. Pattern space points on the other side of the hyperplane generate a negative inner product with W_S and consequently a neuron signal equal to 0. Points in $\mathcal{C}_0$ and $\mathcal{C}_1$ are thus correctly classified by such a placement of the hyperplane.

Now consider the inner product $X^T W$ from a different point of view: for a specific pattern $X_k \in \mathbb{R}^{n+1}$ let the weight vector be the variable vector. The inner product $W^T X_k = 0$ now represents a hyperplane in *weight space*. This hyperplane always passes through the origin since $W = 0$ is a trivial solution of $W^T X_k = 0$. We call this weight space hyperplane the *pattern hyperplane* of pattern X_k. It is the locus of all points W such that $W^T X_k = 0$. It divides the weight space into two parts: one which generates a positive inner product $W^T X_k > 0$ (as indicated by the arrow in Fig. 5.9(b)), and the other where the inner product $W^T X_k$ is negative. For each pattern X_k in pattern space there will be a corresponding hyperplane in weight space. Similarly, for every point in weight space there is a corresponding hyperplane in pattern space.

Each hyperplane in pattern space is defined by a specific instance of weights W_k. Each hyperplane in weight space is defined for a specific instance pattern X_k.



(a) Pattern space (b) Weight space

Fig. 5.9 The geometry of pattern and weight space

Consider the case of four patterns divided into two pattern sets $\mathcal{X}_0$ and $\mathcal{X}_1$, belonging to classes $\mathcal{C}_0$ and $\mathcal{C}_1$ respectively: $\mathcal{X}_1 = \{X_1, X_2\}$ and $\mathcal{X}_0 = \{X_3, X_4\}$. Figure 5.10 depicts four hyperplanes, one for each pattern as depicted by the labels. The pattern hyperplane orientation arrows in the figure indicate weight space regions of positive inner products. As mentioned earlier, when using a TLN classifier to solve this classification problem, we identify $\mathcal{C}_0$ with a neuron signal 0, and $\mathcal{C}_1$ with a neuron signal 1. This puts forth the requirement for positive inner products for vectors

in $\mathcal{X}_1$, and negative inner products for vectors in $\mathcal{X}_0$. The question that then arises is: Which weight vectors in weight space generate positive inner products $W^T X_1$ and $W^T X_2$ (since X_1, X_2 belong to $\mathcal{C}_1$) and *simultaneously* generate negative inner products $W^T X_3$ and $W^T X_4$ (since X_3, X_4 belong to $\mathcal{C}_0$)?

It is straightforward to identify a *solution region* knowing the orientation of the individual pattern hyperplanes as shown in Fig. 5.10. The shaded

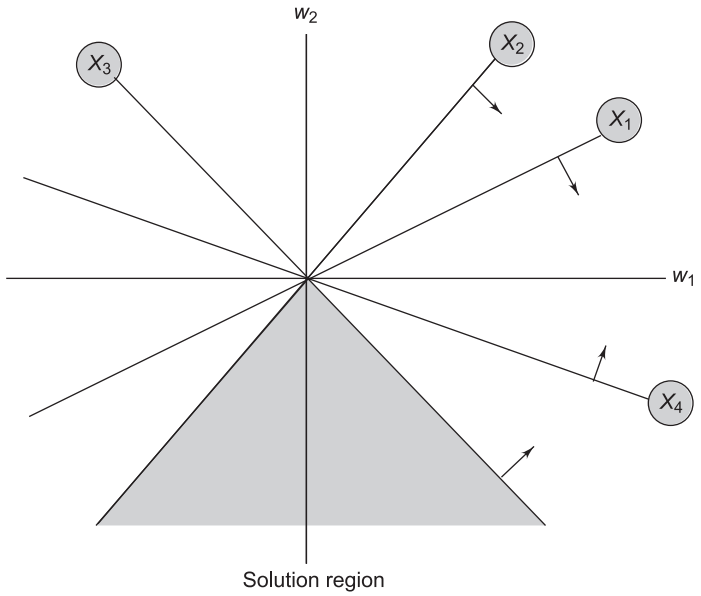


Fig. 5.10 A solution region in weight space with four pattern hyperplanes as marked: $\mathcal{X}_1 = \{X_1, X_2\}$ and $\mathcal{X}_0 = \{X_3, X_4\}$

cone in Fig. 5.10 represents the solution region for the four patterns with an infinite number of weight vectors, each of which represents a single feasible solution to the classification problem at hand.

Linear separability guarantees the existence of such a solution region. In the design of an automated weight update procedure that can search out a solution weight vector, starting out at an arbitrary initial weight vector, the following points need to be kept in mind:

- The procedure must consider each pattern in turn to assess the correctness of the present classification.
- It must subsequently adjust the weight vector to eliminate a classification error, if any.
- Since the set of all solution vectors forms a *convex cone*, the weight update procedure should terminate as soon as it penetrates the boundary of this cone.

We now proceed to study the design of a very important learning algorithm:
the Perceptron learning algorithm.