$$\forall c \in \mathcal{N} \; ; \quad \mathbf{w}'_c = \mathbf{w}_c \cdot \mathbf{P} \tag{2.5}$$

$$\mathbf{P}_{n+k,n} = \begin{pmatrix} \mathbf{I}_{n,n} \\ \mathbf{0}_{k,n} \end{pmatrix} \tag{2.6}$$

where $\mathbf{I}_{n,n}$ is an identity matrix and $\mathbf{0}_{k,n}$ is a zero matrix. Now, we can use a common strategy and find the winner between restricted reference vectors $\mathbf{w}'_c$:

$$\mathbf{w}'_w = \arg\min_{c \in \mathcal{N}} \|\xi - \mathbf{w}'_c\| \tag{2.7}$$

The output of the network in predictive phase is constructed from the original reference vector of the winner i.e. the output vector $\mathbf{y} = (y_1, ..., y_k)$ is filled with class attributes of $\mathbf{w}_w$:

$$\mathbf{y} = \mathbf{w}_w \cdot \mathbf{L} \tag{2.8}$$

$$\mathbf{L}_{n+k,n} = \begin{pmatrix} \mathbf{0}_{n,n} \\ \mathbf{I}_{k,n} \end{pmatrix} \tag{2.9}$$

2.2.5 Class probability prediction

As it was presented in the previous section, resulting neurons positions produced by the GNG algorithm trained on joined dataset (2.4) can be used for predictions. Due to the one-of-C coding used for input signals labels, the classifier produces an output vector $\mathbf{y}$ filled with positive values greater then 0 and lesser then 1. This means that the classifier is making soft decisions **REF** . This is highly demanded and classifiers manifesting such behavior are considered better in general. However, there is another request for soft-decisioning classifiers and it is called *probabilistic classification.*

The idea of probabilistic classification is very simple and it means that the classifier results can be understood as posterior probability estaminate for predicted classes. It is obvious, that there is a condition for the output of the classifier; the sum of output values has to be 1. There are several methods how this condition can be met (for example feet-forward networks often use *soft-max function* as the transfer function for output units **REF**) but the classifier presented in previous section does not need any output transformation at all.

The rest of this section is focused to proving the fact that the presented classifier is making probalilistic classifications.

Low-dimensional insight into joined weight space

For the sake of presentation, I would like to give a simple insight into the problem through the low dimensional example.

Let there is a two dimensional dataset created by several points where every point (input signal) has been labeled with a class flag and let there are only two classes used for labels. This give us 4-dimensional joined dataset (fig. 2.2) which may be viewed in few different projections. Input signals form three clusters in original input space (fig. 2.2 left). Notice the ill pattern in one of the clusters. This pattern will cause an uncertainity in the classification process.

With the omition of the second element, the joined dataset may be viewed in 3D (fig. 2.2 right). Due to the one-of-C coding, inputs are situated within vertical lines.

Let the GNG algorithm is run on such dataset. Final configuration of neurons may be plotted into the the weight space too (fig. 2.2, white circles). The 3D figure show us the neuron c which is producing soft-classification. As it was explained above, the sum of all class variables within the weight vector $\mathbf{w}_c$ has to be 1. Geometrically speaking, it has to be positioned on

the plane $\rho : w_3 + w_4 = 1$. Unfortunetaly, the random initialization of the GNG algorithm does not guarantee such position. But, if one carefully analyzes the GNG algorithm it becomes clear that whatever initial positions of neurons were, all final positions will be on the plane ρ when the learning process is finished.

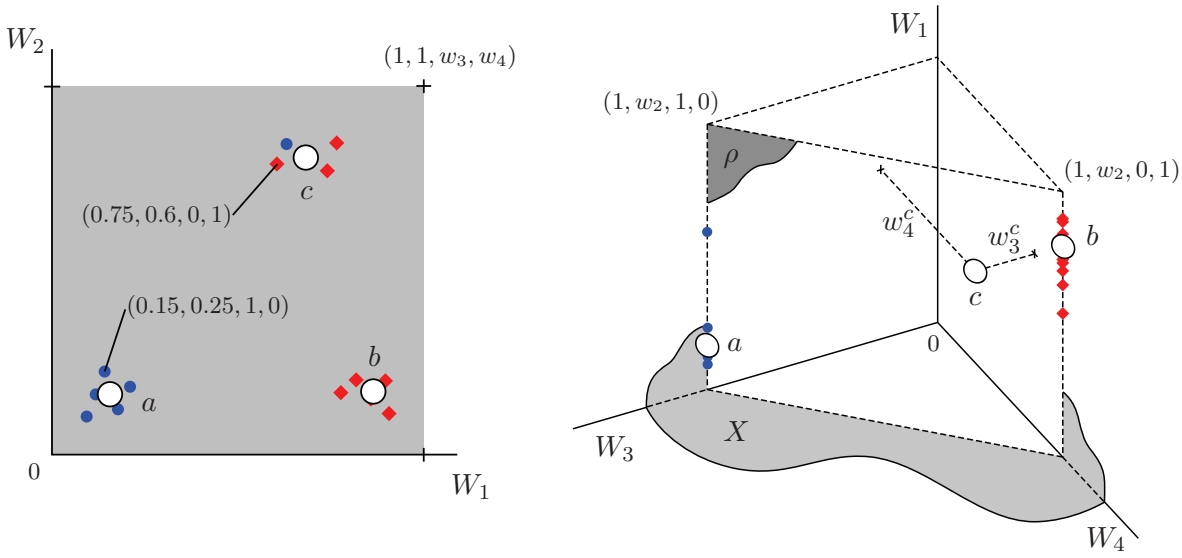


Figure 2.2: Class probability – low-dimensional example.

Arbitrary dimension approach

As it was defined above, the condition for probabilistic classification is $|\mathbf{y}| = 1$ where $\mathbf{y} \in \mathbb{R}^k$ and $\mathbf{y} = (w_{n+1}, w_{n+2}, \ldots, w_{n+k})$ (2.8) i.e. the output must lie on the hyperplane:

$$\rho : w_{n+1} + w_{n+2} + \ldots + w_{n+k} = 1 \tag{2.10}$$

The initial position of any neuron is given by its weight vector which is initialized at random. In general, the output vector is inside the subspace X or X':

$$X \quad : \quad w_{n+1} + w_{n+2} + \ldots + w_{n+k} > 1 \tag{2.11}$$
$$X' \quad : \quad w_{n+1} + w_{n+2} + \ldots + w_{n+k} < 1 \tag{2.12}$$

The adaptation of the weight vector of the winner is defined as (see eq. 1.20 and 2.4):

$$\begin{aligned} \Delta\mathbf{w} &= \eta(\mathbf{j} - \mathbf{w}) \\ \mathbf{w} &= \mathbf{j} - \frac{\Delta\mathbf{w}}{\eta} \end{aligned} \tag{2.13}$$

where $\mathbf{j}$ is a random pattern chosen from the training set and the learning factor is $0 < \eta < 1$. If any neuron is inside X, the following inequation is fulfilled:

$$\begin{aligned} \left(j_{n+1} - \frac{\Delta w_{n+1}}{\eta}\right) + \ldots + \left(j_{n+k} - \frac{\Delta w_{n+k}}{\eta}\right) &> 1 \\ j_{n+1} + \ldots + j_{n+k} - \frac{\Delta w_{n+1} + \ldots + \Delta w_{n+k}}{\eta} &> 1 \end{aligned} \tag{2.14}$$

With the help of one-of-C coding of class variables ($j_{n+1} + j_{n+2} + \ldots + j_{n+k} = 1$ for any training pattern) we may continue to:

$$\frac{\Delta w_{n+1} + \ldots + \Delta w_{n+k}}{\eta} \quad < \quad 0 \tag{2.15}$$

We are almost finished because vector $\Delta \mathbf{w}^{out} = (\Delta w_{n+1}, \ldots, \Delta w_{n+k})$ denotes the change of neuron's position in the subspace from which is the output vector $\mathbf{y}$ generated (2.9). In different words, the inequation above (2.15) tells us that the scalar product of this change vector and the vector $\mathbf{n} = (1, 1, \ldots, 1) \in \mathbb{R}^k$ is negative. Obviously, the vector $\mathbf{n}$ is the normal vector of the hyperplane ρ. The negative scalar product means that the vector $\Delta \mathbf{w}^{out}$ is **pointing towards** the hyperplane ρ. If we change the position of the weight vector $\mathbf{w}$ beeing in the opposite subspace X' (2.12) i.e. the neuron lies "under" the hyperplane ρ, the scalar product $\Delta \mathbf{w}^{out} \cdot \mathbf{n}$ become positive. Again, this means that the change vector is pointing toward hyperplane ρ.

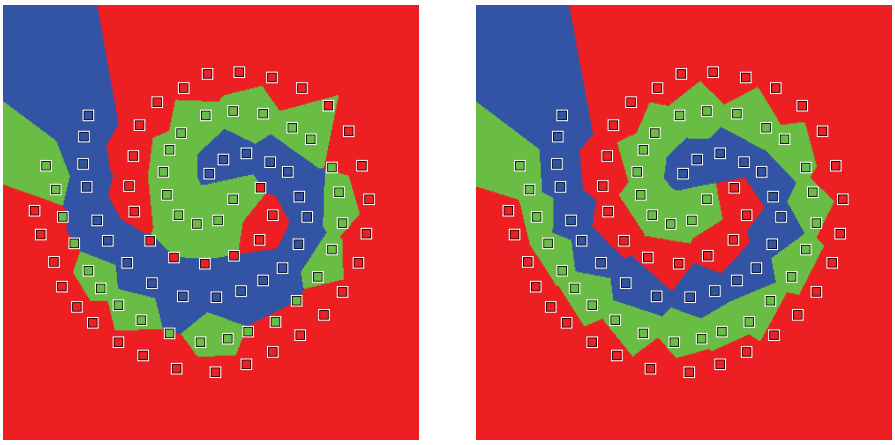
We showed that for any legal training pattern and any initial positions of neurons, all neurons will convergate to the positions which mets the condition for probabilistic classification. Of course there some other conditions (e.g. a reasoneble number of iterations of the GNG algorithm or a suitable value of learning factor η) but in practice this can be achieved easily.

2.3 Experiments

The following sections presents some experiments done by described Extended GNG in comparison with standard LVQ method.

2.3.1 Concentric spirals dataset

The first experiment has been run on the concentric spirals dataset already presented above. The advantage of this dataset is that it is possible to visualize the result of classification on a plane. For the sake of presentation, this low-dimensional experiment is shown on following pictures.



(a) LVQ classification (b) Extended GNG classification

Figure 2.3: Difference between the standard LVQ classifier and the Extended GNG classifier. The training set contains two-dimensional points distributed as concentric spirals. There are three different classes in the training set denoted by color. Colored areas show how the network will predict the class of an arbitrary point within the input space.