On the use of simple classifiers for the initialisation of one-hidden-layer neural nets

Jan C. Bioch, Robert Carsouw, Rob Potharst
Department of Computer Science
Erasmus University Rotterdam
P.O. Box 1738, 3000 DR Rotterdam
The Netherlands
{bioch, robp}@cs.few.eur.nl

Technical Report eur-cs-95-08

Abstract

In this report we discuss the use of two simple classifiers to initialise the input-to-hidden layer of a one-hidden-layer neural network. These classifiers divide the input space in convex regions that can be represented by membership functions. These functions are then used to determine the weights of the first layer of a feedforward network.

Keywords and phrases: mapping decision trees onto neural networks, simple perceptrons, LVQ-networks, initialisation of feedforward networks.

1 Introduction

In this report we discuss how two well-known classifiers can be used to initialise the first layer of a neural network. These classifiers divide the input space in convex regions. These regions are represented by so-called membership functions generated during training. Subsequently, the coefficients of these membership functions are used to initialise (the weights of) a neural network. This is called a mapping of the classifiers onto the neural net. The neural net is then further trained to improve the performance of the classifier. According to Park [Par94] such a mapping results in a faster convergence of the neural net and in avoiding local minima in network training. In general these mappings are also interesting because they determine an appropriate architecture of the neural net.

In this report we mainly discuss a mapping of a linear tree classifier (LTC) onto a feedforward neural net classifier (NNC) with one hidden layer. The LTC used here is a hierarchical classifier that employs linear functions at each node in the tree. For the construction of decision trees we refer to [PaS90, FaI92, Qui93]. It is known that both the LTC and NNC classifier share the property of universal approximation, i.e both classsifiers are able to solve arbitrary classification problems by forming complex decision boundaries in the feature space. Several authors [Set90, IKP94, Par94] discuss the mapping of an LTC onto a feedforward net with one or two hidden layers, see also [Car95]. A discusion of a mapping onto a net with two hidden layers can be found in Sethi [Set90] and Ivanova&Kubat [IKP94]. A mapping onto a net with one hidden layer is discussed in Park [Par94]. However, the question of how to map an LTC directly on a neural net with one hidden layer is still open. Park [Par94] has suggested an interesting approach of such a mapping based on representing the convex regions induced by an LTC by linear membership functions. However, in Park [Par94] no explicit expression for the coefficients of the membership functions is given. These coefficients depend on a parameter ρ that has to be supplied by the user. In section 3 we show that in general it is not possible to find linear membership functions that represent the convex regions induced by an LTC. It is however possible to find subregions that can be represented by linear membership functions. We derive explicit expressions for the aforementioned parameter ρ, in section 4. This makes it possible to control the approximation of the convex regions by membership functions and therefore of the initialisation of the neural net. In section 5 we discuss the use of LVQ-networks to initialise a feedforward network. It appears that unlike an LTC, an LVQ-network divides the input space in convex regions that can be represented by linear membership functions.

2 Mapping decision trees to neural nets

Suppose we are given a multivariate decision tree $(\mathcal{N}, \mathcal{L}, \mathcal{D})$. In this notation, $\mathcal{N}$ is the set of nodes of the tree, $\mathcal{L}$ is the set of leaves of the tree and $\mathcal{D}$ is the set of linear [1] functions $d_k : \mathbb{R}^n \to \mathbb{R}, k \in \mathcal{N}$. In any node k of the tree, the linear function d_k is used to decide which branch to take. Specifically, we go left if $d_k(x) > 0$, right if $d_k(x) \leq 0$, see figure 1.

A decision tree induces a partitioning of $\mathbb{R}^n$: each leaf ℓ corresponds with a convex region R_ℓ, which consists of all points $x \in \mathbb{R}^n$, that get assigned to leaf ℓ by the decision tree. For example, region R_5 consists of all $x \in \mathbb{R}^n$ with $d_1(x) \leq 0$ and $d_3(x) \leq 0$.

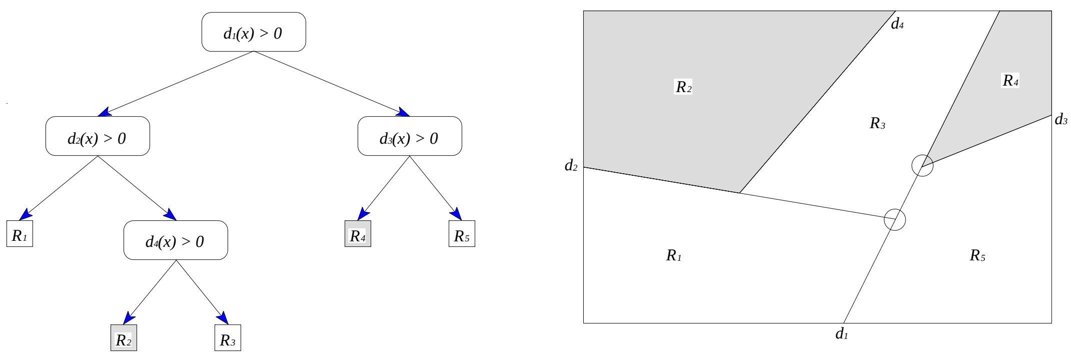


Figure 1: The convex regions induced by a classification tree

In [Set90] Sethi shows a straightforward mapping of an LTC onto an NNC with two hidden layers, see figure 2. In this approach the coefficients of the linear decision functions are used to define the weights in the input to partition layer. Each node in the partition layer

[1] A linear function is understood to be a function with the following form : $d_k(x) = \sum_{j=1}^{n} w_{kj} x_j + w_{k0}$, for $x = (x_1, \ldots, x_n) \in \mathbb{R}^n, k \in \mathcal{N}$.

corresponds to a decision node in the tree. The nodes in the AND-layer correspond with the decision regions induced by the LTC. Each node in the AND-layer represents a path from the root to a leaf. Subsequently, this decision regions are joined in the OR-layer, to represent the classes. An exact mapping employs threshold units in the neural net. However, as shown in [Set90] continuous transfer functions such as the sigmoid function are more appropriate. Therefore the mapping is only used as an initialisation of a two-hidden-layer neural net. An important feature of Sethi's mapping is that it solves the credit assignment problem for the hidden layers. Therefore it is possible to train the two layers separately. For another interesting approach of an LTC to NNC mapping we refer to [IKP94]. Although a direct mapping onto a one-hidden-layer is not yet known, Park [Par94] describes an approach to initialise the weights in the input to hidden layer. In this approach, discussed in sections 3 and 4 the coefficients of the linear membership functions are obtained by combining the decision functions employed by the tree to define the aforementioned weights, see figure 3.

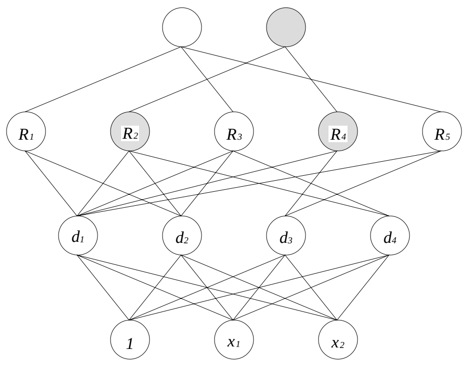


Figure 2: Sethi's mapping

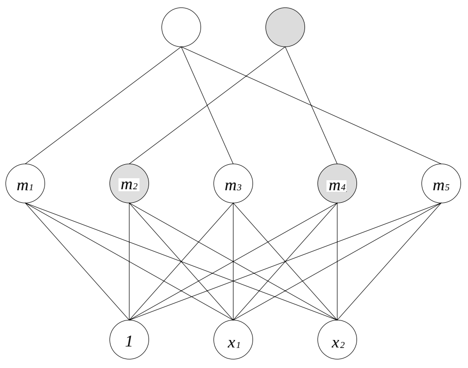


Figure 3: Park's mapping

3 Non-existence of linear membership functions for LTCs

In this section we discuss the idea of linear membership functions to represent the convex regions induced by an LTC, and we show that these functions are in general not possible.

In [Par94] the following 'theorem' is given without proof, though supplied with heuristic reasoning for its plausibility, see also equation 5 in the next section:

Conjecture (Park[Par94]) *For every decision tree $(\mathcal{N}, \mathcal{L}, \mathcal{D})$ there exists a set of linear membership functions $M = \{m_\ell, \ell \in \mathcal{L}\}$, such that for any $\ell, \ell' \in \mathcal{L}$, with $\ell \neq \ell'$:*

$$m_\ell(x) > m_{\ell'}(x), \forall x \in R_\ell. \tag{1}$$

We first will refute this conjecture by giving a (smallest) *counterexample*. Then we show that in general linear membership functions cannot exist, and finally we discuss the possibility of polynomial membership functions.

Set $n = 2$; we then have points $(x, y) \in \mathbb{R}^2$ as our decision vectors. Now, consider the following decision tree $(\mathcal{N}, \mathcal{L}, \mathcal{D})$: $\mathcal{N} = \{1, 2\}, \mathcal{L} = \{1, 2, 3\}$ and $d_1(x, y) = x$, $d_2(x, y) = y$.

This decision tree and its corresponding partitioning of the sample space are shown in figure 4:

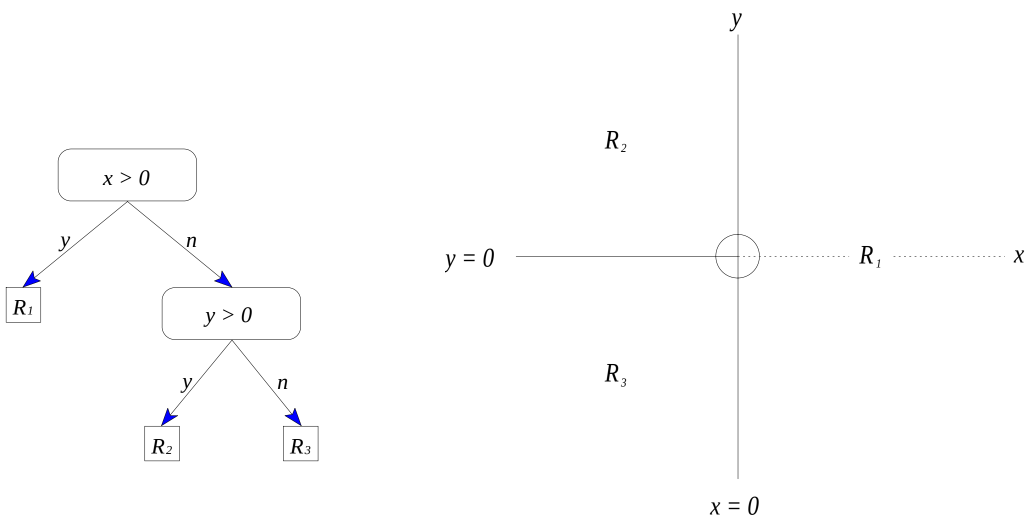


Figure 4: The simplest counterexample to Park's conjecture

Suppose now, that linear functions m_1, m_2 and m_3 exist, for which the inequality (1) is true. We then have

$$\begin{aligned} m_1 &= p_1x + q_1y + r_1 \\ m_2 &= p_2x + q_2y + r_2 \\ m_3 &= p_3x + q_3y + r_3 \end{aligned}$$

with

$$\begin{aligned} &m_1 > m_2 \text{ and } m_1 > m_3 \text{ for all } (x,y) \in R_1 \\ &m_2 > m_1 \text{ and } m_2 > m_3 \text{ for all } (x,y) \in R_2 \\ &m_3 > m_1 \text{ and } m_3 > m_2 \text{ for all } (x,y) \in R_3. \end{aligned}$$

For the linear function $u(x,y) = m_1 - m_2 = (p_1 - p_2)x + (q_1 - q_2)y + (r_1 - r_2)$ we then see that it is positive on R_1 and negative on R_2. Consequently, $u = 0$ on the borderline between R_1 and R_2, which is the vertical axis. It is then easy to see that

$$p_1 > p_2,\ q_1 = q_2 \text{ and } r_1 = r_2 \tag{2}$$

must hold.
If next we consider the linear function $v(x,y) = m_1 - m_3 = (p_1 - p_3)x + (q_1 - q_3)y + (r_1 - r_3)$, it will be seen that v is positive on R_1 and negative on R_3. Analogously, we then find that

$$p_1 > p_3,\ q_1 = q_3 \text{ and } r_1 = r_3. \tag{3}$$

By combining (2) and (3) we find that $u - v = m_3 - m_2 = (p_3 - p_2)x$ and, consequently,

$$(p_3 - p_2)x > 0 \text{ on } R_3, \text{ and } < 0 \text{ on } R_2. \tag{4}$$

For points $(x, y) \in R_2 \cup R_3$, we have $x < 0$. Thus, it follows from (4) that both $p_3 < p_2$ and $p_3 > p_2$ must be true, which is a contradiction. We must conclude that the hypothesised functions m_1, m_2 and m_3 cannot exist. Thus, the conjecture is refuted.

Remark. If the decision functions $d_1, d_2, \ldots$ are chosen in such a way that the hyperplanes $d_i(x) = 0$ are parallel, then in general it will be possible to construct the above mentioned m-functions. For example, in $\mathbb{R}^2$ it can be done as follows:

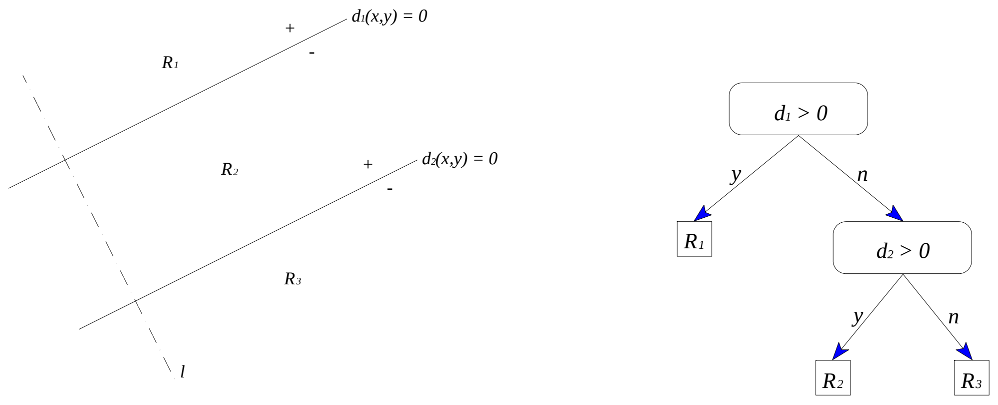
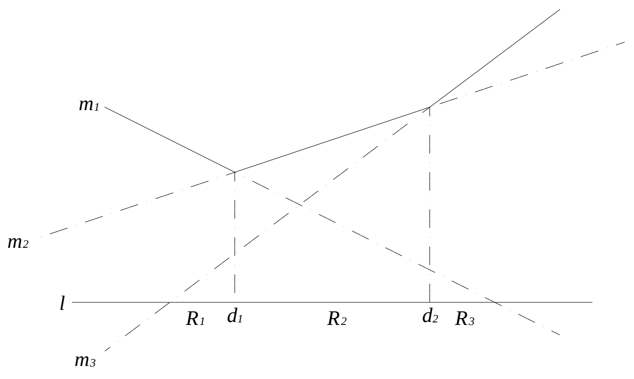


Figure 5: A decision tree with parallel decision boundaries

The graphs of the functions m_1, m_2 and m_3, intersected with a plane through ℓ (= a perpendicular to d_1 and d_2) then look as shown in figure 6:



$$m_1 > \max(m_2, m_3) \text{ on } R_1$$
$$m_2 > \max(m_1, m_3) \text{ on } R_2$$
$$m_3 > \max(m_1, m_2) \text{ on } R_3.$$

Figure 6: A solution in the case of parallel decision boundaries

In the next theorem we show that linear membership functions that represent the convex regions induced by an LTC cannot exist.

Theorem 1 *Let $(\mathcal{N}, \mathcal{L}, \mathcal{D})$ be a decision tree, with at least two non-parallel decision boundaries. Then the convex regions induced by this tree cannot be represented by a set of linear membership functions.*

Proof Let d_1 and d_2 be two n-dimensional linear decision functions such that the hyperplanes $d_1 = 0$ and $d_2 = 0$ are non-parallel. Then for our discussion it is no restriction to consider only the convex regions R_1, R_2 and R_3 induced by these hyperplanes, see figure 7 (left side). The corresponding membership functions of these regions are denoted by m_1, m_2

and m_3, and the differences $m_1 - m_2$, $m_1 - m_3$ and $m_2 - m_3$ are respectively denoted by u, v and w. By definition we have: $u > 0$ on R_1 and $u < 0$ on R_2. Since linear functions are continuous, it is easy to see that the inequalities imply that the linear function u is zero on the hyperplane $d_1 = 0$. Similarly it follows that the function v is zero on $d_1 = 0$ and that w is zero on $d_2 = 0$. Since $w = u - v$, w is also zero on $d_1 = 0$. Using the fact that w is a linear function which is zero on two non-parallel hyperplanes we conclude that w is identical to the zero-function. This contradicts the fact that $w > 0$ on R_2. $\square$.

Having shown that, in general, linear membership functions cannot represent the convex regions induced by an LTC, the question now arises whether membership functions of LTCs can be represented by multivariate polynomials. The following theorem shows that these polynomials cannot be quadratic.

Theorem 2 *Let $(\mathcal{N}, \mathcal{L}, \mathcal{D})$ be a decision tree, with at least two non-parallel decision boundaries. Then the convex regions induced by this tree cannot be represented by a set of quadratic polynomials.*
Proof Let R_1, R_2 and R_3 be the regions induced by a decision tree, see figure 7 (left side).

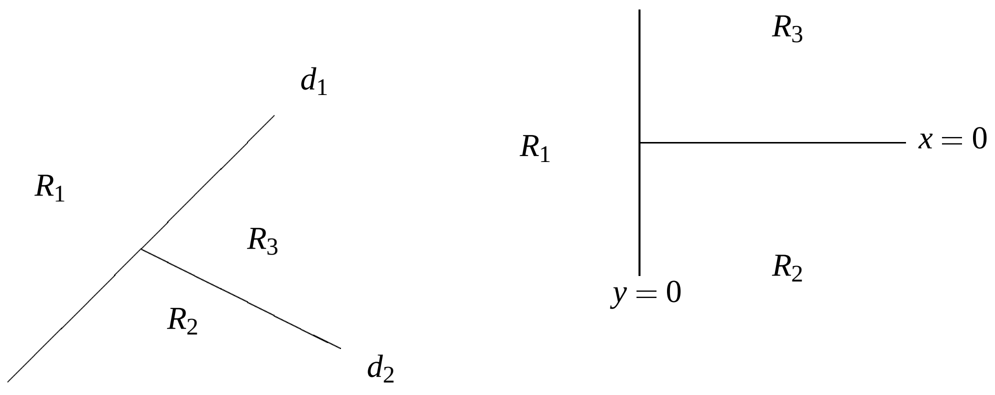


Figure 7: Decision regions in space

Note that such regions will always be induced by a subtree of a decision tree, unless all decision boundaries are parallel. We assume that the regions R_1 and $R_2 \cup R_3$ are separated by the hyperplane $d_1(x) = 0, x \in \mathbb{R}^n$ such that $d_1(x) > 0$ on R_1 and $d_1(x) < 0$ on $R_2 \cup R_3$. Similarly, R_2 and R_3 are separated by $d_2(x) = 0$, such that $d_2(x) > 0$ on R_2 and $d_2(x) < 0$ on R_3. Let m_1, m_2 and m_3 respectively denote the membership functions of R_1, R_2 and R_3. By definition $m_1 = 0$ on the hyperplane $d_1(x) = 0$. Similarly, $m_2 = 0$ on $d_2(x) = 0$. (Note, that we actually know only that m_2 is zero on half of the hyperplane $d_2(x) = 0$. However, using a simple result from algebraic geometry it follows that m_2 must be zero on the whole hyperplane $d_2(x) = 0$.)

Now, let $D_{12} = m_1 - m_2$. Then D_{12} is zero on $d_1(x) = 0$, because $D_{12} > 0$ on R_1 and $D_{12} < 0$ on $R_2 \cup R_3$. As a consequence of Hilbert's Nullstellensatz d_1 is a factor of D_{12}. Therefore, there exists a polynomial function e such that

$$D_{12} = d_1 e.$$

Since D_{12} is at most quadratic by assumption, we conclude that e is a constant or a linear function. However, since both $d_1 e$ and d_1 are positive on R_1 and negative on R_2, the function e is positive on $R_1 \cup R_2$. Since the degree of e is ≤ 1, e must be a positive constant.

Similarly, we have $D_{13} = d_1 f$, where f is a positive constant. Therefore $D_{23} = d_1(f - e)$. This contradicts the fact that D_{23} is zero on $d_2(x) = 0$. $\square$.

Finally we discuss a small example, to show that multivariate polynomials can represent membership functions, although the degree of these polynomials is at least 5.

Example Let R_1, R_2 and R_3 be the plane regions as given in figure 7 (right side). Then using the same notation as in the above theorem, we can prove that the polynomial D_{23} has the form $D_{23} = xyg$, where g is a polynomial of even degree. The reader can verify that the regions R_i can be separated by the following polynomials of degree at most 5:

$$\begin{aligned} m_1 = &\ 0 \\ m_2 = &\ x(y^4+1) \\ m_3 = &\ x(y^4+y^3+1). \end{aligned}$$

4 An approximated mapping of a decision tree onto a one layer neural network

Let H_k be the hyperplane given by $d_k(x) = \sum_{j=1}^{n} w_{kj}x_j + w_{k0}$, for $x = (x_1, \ldots, x_n) \in \mathbb{R}^n, k \in \mathcal{N}$. Then the distance of a point x to H_k is given by

$$|w_k^T x + w_{k0}|/||w_k||, \text{ where } w_k^T = (w_{k1}, \ldots, w_{kn}).$$

Without restriction we may assume that $||w_k|| = 1$. Therefore, $|d_k(x)|$ is the distance of a point x to the hyperplane H_k. We will show in this section that the difficulties encountered in the preceding section may be circumvented by requiring that the points we consider are not too close to the hyperplanes associated with the decision tree.

Let $(\mathcal{N}, \mathcal{L}, \mathcal{D})$ be a decision tree. We will restrict the regions R_ℓ by assuming that $\forall k \in \mathcal{N}:$ $0 < e \leq |d_k(x)| \leq E$, where e and E are positive constants. The set of points in R_ℓ satisfying this condition will be denoted by S_ℓ. Hence, S_ℓ is a convex subregion of R_ℓ. Note also that R_ℓ can be approximated by S_ℓ with arbitrary precision, by varying the constants e and E.

In [Par94] Park considers the following set of linear membership functions:

$$m_\ell(x) = \sum_{k \in P_\ell} s_{\ell k} c_k d_k(x), \tag{5}$$

where P_ℓ is the set of nodes on the path from the root to the leaf ℓ. The constants $s_{\ell k}$ are defined as:

$$s_{\ell k} = \begin{cases} +1 & \text{if } d_k(x) > 0, \forall x \in R_\ell \\ -1 & \text{if } d_k(x) < 0, \forall x \in R_\ell. \end{cases} \tag{6}$$

In this way it is secured that $\forall x \in R_\ell, \forall k \in P_\ell:$

$$s_{\ell k} d_k(x) > 0. \tag{7}$$

The constants c_k are determined experimentally in [Par94]. Here we will derive an explicit expression for these constants. Since as we have shown above that in general these constants cannot exist if $x \in R_\ell, \ell \in \mathcal{L}$, we will now assume that $x \in S_\ell$.

Theorem 1 *Let* $(\mathcal{N}, \mathcal{L}, \mathcal{D})$ *be a decision tree. Then there exists a set of linear functions* $M = \{m_\ell, \ell \in \mathcal{L}\}$*, such that for any* $\ell, \ell' \in \mathcal{L}$*, with* $\ell \neq \ell'$*:*

$$\forall x \in S_\ell : m_\ell(x) > m_{\ell'}(x). \tag{8}$$

Proof Let $\mathcal{T}$ be the set of terminal nodes of the tree. An internal node t is called terminal if both children of t are leaves. Further, if n_1 and n_2 are two nodes, then we write $n_1 < n_2$ if n_1 is an ancestor of n_2. Suppose that $n_a \notin \mathcal{T}$, and ℓ, ℓ' are two leaves such that n_a is the *last common ancestor* of ℓ and ℓ', see figure 8:

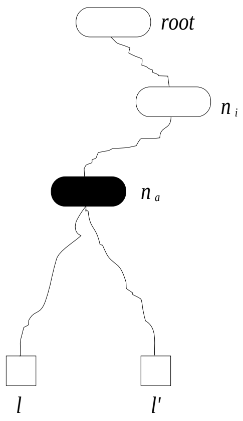


Figure 8: The last common ancestor of ℓ and ℓ'

We decompose the function $m_{\ell'}$ as follows:

$$m_{\ell'}(x) = \sum_{n_i < n_a} s_{\ell i} c_i d_i(x) + \sum_{n_j \geq n_a} s_{\ell' j} c_i d_j(x),$$

where $n_i \in P_\ell$ and $n_j \in P_{\ell'}$. The following crucial observation

$$s_{\ell a} = -s_{\ell' a}$$

can be seen by applying (6) and (7) to the node n_a. From this we conclude:

$$m_\ell(x) - m_{\ell'}(x) = 2 s_{\ell a} c_a d_a(x) + \Sigma_1 - \Sigma_2,$$

where

$$\Sigma_1 = \sum_{n_i > n_a} s_{\ell i} c_i d_i(x), \text{ with } n_i \in P_\ell$$

and

$$\Sigma_2 = \sum_{n_j > n_a} s_{\ell' j} c_j d_j(x), \text{ with } n_j \in P_{\ell'}.$$

To assure that (8) holds, we require that

$$\frac{\Sigma_2 - \Sigma_1}{2 s_{\ell a} d_a(x)} < c_a. \tag{9}$$

Since Σ_1 is a positive number we can satisfy condition (9) by taking

$$\frac{\Sigma_2}{2s_{\ell_a} d_a(x)} < c_a,$$

or

$$\frac{E}{2e} \sum_{n_j > n_a} c_j \leq c_a.$$

This yields the following sufficient condition for the constants c_j:

$$\frac{E}{2e} \max_{\ell \in \mathcal{L}_a} \left\{ \sum_{n_j > n_a} c_j \right\} \leq c_a, \tag{10}$$

where $n_j \in P_\ell$ and $\mathcal{L}_a$ is the set of leaves of the subtree with root n_a. Condition (10) implies: $c_a \geq \frac{E}{2e} \sum_{n_j > n_a} c_j$, where $n_j \in \mathcal{P}$ and $\mathcal{P}$ is the *longest* possible path from node n_a to some leaf. From condition (10) it also follows that the constants c_a are determined up to a constant factor. It is easy to see that the constants can be recursively determined by choosing positive values c_t for the set of terminal nodes $t \in \mathcal{T}$. As an example we show in figure 9 the values of all the constants c_j for a tree, when we take $c_t = 1$ for all $t \in \mathcal{T}$ and $\frac{E}{2e} = 1$. □

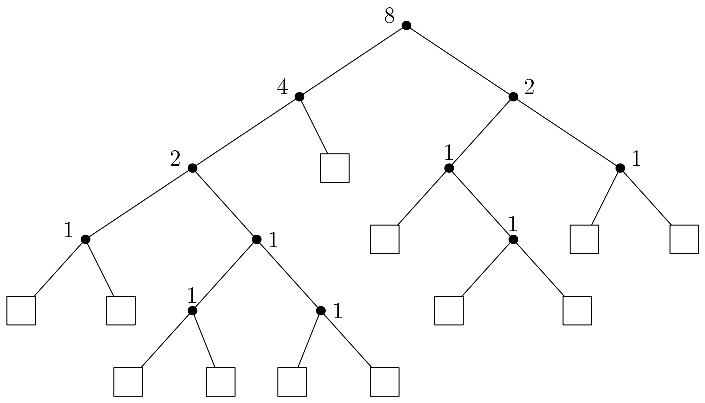


Figure 9: Determination of constants c_j in an example tree.

To derive an an explicit expression for c_a for any node n_a we will introduce some notation. Let $\mathcal{P}_a = \{t = n_0 < n_1 < n_2 < \ldots < n_k = n_a\}$ be the longest possible path from node n_a to a terminal node t. The length k of $\mathcal{P}_a$ is denoted by $\delta(a)$. Similarly, $\delta(r)$ is the length of the longest path starting in the root r of the tree.

Lemma *Let* $(a_i)_{i=0}^{\infty}$ *be the sequence defined by* $a_k = \lambda(a_0 + a_1 + \ldots + a_{k-1})$ *for* $k \geq 2$ *and* $a_1 = \lambda a_0$. *Then* $a_k = a_0 \lambda(\lambda + 1)^{k-1}, k \geq 1$.
Proof For $k \geq 2$ we have: $a_{k+1} - a_k = \lambda a_k$. This implies $a_k = (1+\lambda)^{k-1} a_1 = \lambda a_0 (1 + \lambda)^{k-1}, k \geq 1$. □

Theorem 2 *The constants*

$$c_a = \left(\frac{1}{1 + \frac{E}{2e}} \right)^{\delta(r) - \delta(a)}$$

satisfy condition (10).
Proof Let $A_a = \frac{E}{2e}\max_{\ell\in\mathcal{L}_a}\left\{\sum_{j\in P_\ell} A_j\right\}$, and $A_\ell = 1$. Then $A_a = \lambda(1+\lambda)^{\delta(a)-1}$ (see the above Lemma), with $\lambda = \frac{E}{2e}$. Since the constants c_a are defined up to a constant factor, we choose $c_a = \frac{A_a}{A_r}$. Then $c_a = \left(\frac{1}{1+\lambda}\right)^{\delta(r)-\delta(a)}$. $\square$

Corollary *Let* $\rho = 1/(1+\frac{E}{2e})$. *Then* $m_\ell(x) = \sum_k s_{k\ell}\rho^{\delta(r)-\delta(k)} d_k(x)$ *are linear functions for which (8) holds.*

Another expression for the constants c_a can be obtained by using the fact that a decision tree $(\mathcal{N},\mathcal{L},\mathcal{D})$ in practical situations is derived from a finite set of examples $V \subset \mathbb{R}^n$. It is no restriction to assume that $\forall x \in V : |d_k(x)| > 0, k \in \mathcal{N}$.
Let $M = \max_{i,j}\max_{x\in V}\left|\frac{d_i(x)}{d_j(x)}\right|$, where $n_i, n_j \in \mathcal{N}$ and $n_i < n_j$. The number M can be easily computed from the set of examples V.

Theorem 3 *Let* $\rho = 2/(3M)$. *Then the constants* $c_a = \rho^{\delta(r)-\delta(a)}$ *satisfy condition (10).*
Proof Suppose P_ℓ is the longest path from node n_a to a leaf ℓ, and n_j, n_{j+1} are two consecutive nodes on P_ℓ with $n_j < n_{j+1}$. Then $d_j(x) \le M d_{j+1}(x)$, so that $d_j(x) \le M^{\delta(a)-\delta(j)}|d_a(x)|$. To satisfy condition (9) we want to choose constants A_j such that

$$\Sigma = \sum_{n_j > n_a} \frac{s_{mj} A_j d_j(x)}{2 s_{\ell a} d_a(x)} \le A_a.$$

Since

$$\Sigma \le \frac{1}{2}\sum_j \frac{A_j}{M^{\delta(j)}} M^{\delta(a)} \le A_a,$$

we have

$$\frac{1}{2}\sum_j \frac{A_j}{M^{\delta(j)}} \le \frac{A_a}{M^{\delta(a)}},$$

or $\frac{1}{2}\sum_j B_j \le B_a$, where $B_j = A_j/M^{\delta(j)}$. Now we choose B_a such that $B_a = \frac{1}{2}\sum_j B_j$. Then by the Lemma preceding Theorem 2 we have $B_a = \lambda(\lambda+1)^{\delta(a)-1}$, with $\lambda = \frac{1}{2}$. Finally, let $c_a = B_a/B_r$; then we have

$$c_a = \left(\frac{1}{M(\lambda+1)}\right)^{\delta(r)-\delta(a)} = \rho^{\delta(r)-\delta(a)}, \text{ with } \rho = \frac{2}{3M}. \quad \square$$

Discussion: We have found two possible values for ρ: $\rho_1 = 1/(1+E/(2e))$ and $\rho_2 = 2/(3M)$. From the definition of M it follows that $M \le E/e$. Since $E/e \ge 1$, we have $\rho_1 \le 1/(1+\frac{1}{2}) = 2/3$. On the other hand, it is easy to verify that $\rho_1 \ge \rho_2$ if and only if $E/e \le 3M-2$.

Since $\rho \to 0$ if $e \to 0$, in practice we will choose the parameter e not too small. On the other hand, the approximation of the regions R_ℓ by S_ℓ will be less accurate if e becomes larger. In a tree to neural net mapping the functions m_ℓ are used to define the initial weights between the input and hidden layer. To improve the performance of the neural net these weights are subsequently updated. It is therefore plausible, and in accordance with some experiments of Park [Par94], and with the data used in an experiment described in [Car95] that the accuracy of the approximation of the regions R_ℓ must be less, the more the data set used to generate the decision tree is non-linear.

5 Another initialisation method

In this section we consider another well-known classifier: Learning Vector Quantisation (LVQ) networks. This method can be used to solve a classification problem with m classes and data-vectors $x \in \mathbb{R}^n$. It is known that the LVQ-network induces a so-called Voronoi tesselation of the input space, see [HKP91] chapter 9. Training of an LVQ-network yields prototype vectors $w_j \in \mathbb{R}^n, j = 1, \ldots, m$ such that an input vector x belongs to class j iff the distance to w_j is smallest:

$$\forall i \neq j : ||w_j - x|| \leq ||w_i - x|| \Rightarrow x \in R_j.$$

It is easy to see that this is equivalent with

$$w_j^T x - \frac{1}{2} w_j^T w_j \geq w_i^T x - \frac{1}{2} w_i^T w_i.$$

Now define the linear membership function m_i as:

$$m_i(x) = w_i^T x - \frac{1}{2} w_i^T w_i.$$

Then

$$x \in R_i \iff m_i(x) > m_j(x), \forall j \neq i.$$

Since an LVQ-network is a relatively good classifier and can be trained relatively fast, it is a good alternative for the initialisation of a neural net using linear membership functions. Apparently, an LVQ-network cannot induce the convex regions induced by an LTC.

Example Let R_1, R_2 and R_3 be such as in figure 10 (right side). Now let w_1 be the prototype vector of R_1. Then the situation of figure 10 (left) can only be obtained if $||w_1|| \to \infty$.

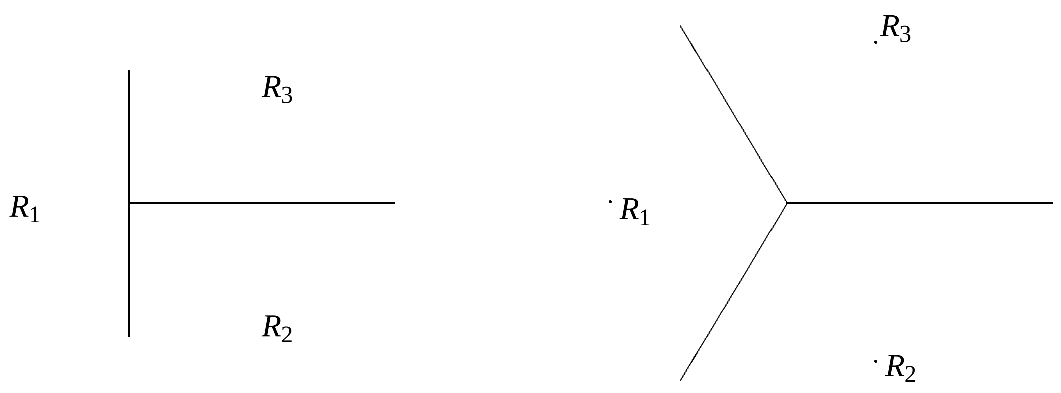


Figure 10: Different types of convex regions

6 Conclusions and further research

We have proven that linear (or quadratic) membership functions representing the convex regions of a linear classifier in general do not exist. However, we give explicit formulae for the approximation of such functions. This allows us to control the degree of approximation. This is useful, for in practical applications the membership functions are 'only' used to initialise a one-hidden-layer neural net. Subsequently this net is further trained with a standard training algorithm. Currently we are investigating how to determine an appropriate approximation in a given application. We are also investigating two other approaches, see [Car95] to obtain a mapping from am LTC to a one-hidden-layer neural net. In the first approach we combine the first two layers of Sethi's net into one layer. The other approach uses appropriate transfer functions for the hidden layer to represent a join of two half spaces, because it is clear that an exact representation of the convex regions can be obtained by using piecewise linear functions. Finally, we have discussed LVQ-networks as an alternative that is of interest for the initialisation of the input-to-hidden layer based on membership functions of convex regions. This architecture divides the input space in polyhedral regions, see [MP88, HKP91] with linear membership functions.

Acknowledgement

We are grateful to Reino de Boer for his help with the pictures and advice on $\LaTeX$.

References

[BDV94] J.C. Bioch, M. van Dijk and W. Verbeke, Neural Networks: New Tools for Data Analysis?, In: M. Taylor and P. Lisboa (eds.) Proceedings of Neural Networks Applications and Tools, *IEEE Computer Society Press*, pp. 28-38, 1994.

[Car95] R. Carsouw, Learning to Classify: Classification by neural nets based on decision trees, Masterthesis (in Dutch), Dept. of Computer Science, Erasmus University Rotterdam, Februari 1995.

[IKP94] I. Ivanova, M. Kubat and G. Pfurtscheller, The System TBNN for Learning of 'Difficult' Concepts. In: J.C. Bioch and S.H. Nienhuys-Cheng (eds), *Proceedings of Benelearn94*, Tech. Rep. eur-09-94, Dept. of Computer Science, Erasmus University Rotterdam, pp. 230-241, 1994.

[FaI92] U.M. Fayad and K.B. Irani, On the Handling of Continuous-Valued Attributes in Decision Tree Generation, *Machine Learning*, vol. 8, pp. 88-102, 1992.

[HKP91] J. Hertz, A. Krogh, R.G. Palmer, *Introduction to the theory of neural computation*, Addison-Wesley, 1991.

[MP88] M.L. Minsky, S.A. Papert, *Perceptrons, 2nd ed.*, MIT Press, 1988.

[Par94] Y. Park, A Mapping From Linear Tree Classifiers to Neural Net Classifiers, *Proceedings of IEEE ICNN*, vol. I, pp. 94-100, Orlando, Florida, 1994.

[PaS90] Y. Park and J. Sklansky, Automated Design of Linear Tree Classifiers, *Pattern Recognition*, vol. 23, no. 12, pp. 1393-1412, 1990.

[Set90] A.K. Sethi, Entropy Nets: From Decision Trees to Neural Networks, *Proceedings of the IEEE*, vol. 78, no. 10, pp. 1606-1613, 1990.

[Qui93] J.R. Quinlan, *C4.5 Programs for Machine Learning*, Morgan Kaufmann, San Mateo, California, San Mateo, 1993.