Spherical Decision Surfaces Using Conformal Modelling

Christian Perwass, Vladimir Banarer, Gerald Sommer

Institut für Informatik und Praktische Mathematik
Christian-Albrechts-Universität zu Kiel
Christian-Albrechts-Platz 4, 24118 Kiel, Germany
{chp,vlb,gs}@ks.informatik.uni-kiel.de

Abstract. In this paper a special higher order neuron, the hypersphere neuron, is introduced. By embedding Euclidean space in a conformal space, hyperspheres can be expressed as vectors. The scalar product of points and spheres in conformal space, gives a measure for how far a point lies inside or outside a hypersphere. It will be shown that a hypersphere neuron may be implemented as a perceptron with two bias inputs. By using hyperspheres instead of hyperplanes as decision surfaces, a reduction in computational complexity can be achieved for certain types of problems. Furthermore, it will be shown that Multi-Layer Percerptrons (MLP) based on such neurons are similar to Radial Basis Function (RBF) networks. It is also found that such MLPs can give better results than RBF networks of the same complexity. The abilities of the proposed MLPs are demonstrated on some classical data for neural computing, as well as on real data from a particular computer vision problem.

1 Introduction

The basic idea behind a single standard perceptron is that it separates its input space into two classes by a hyperplane [13]. For most practical purposes such a linear separation is, of course, not sufficient. In general, data is to be separated into a number of classes, where each class covers a particular region in the input space. The basic idea behind classifying using a multi-layer perceptron (MLP), is to use a number of perceptrons and to combine their linear decision planes, to approximate the surfaces of the different class regions. In principle, a MLP can approximate any type of class configuration, which implies that it is an universal approximator [4, 7].

However, being an universal approximator alone says nothing about the complexity a MLP would need to have in order to approximate a particular surface. In fact, depending on the structure of the data it may be advantageous to not use perceptrons but instead another type of neuron which uses a non-linear 'decision surface' to separate classes. Such neurons are called *higher-order* neurons. There has been a lot of effort to design higher-order neurons for different applications. For example, there are hyperbolic neurons [3], tensor neurons [12] and hyperbolic SOMs [14]. Typically, the more complex the decision surface a neuron has is, the higher its computational complexity. It is hoped that a complex decision

surface will allow to solve a task with fewer neurons. However, the computational complexity of each neuron should not offset this advantage.

In this paper we present a simple extension of a perceptron, such that its decision surface is not a hyperplane but a hypersphere. The representation used is taken from a conformal space representation introduced in the context of Clifford algebra [11]. The advantage of this representation is that only a standard scalar product has to be evaluated in order to decide whether an input vector is inside or outside a hypersphere. That is, the computational complexity stays low, while a non-linear decision plane is obtained. Furthermore, a hypersphere neuron with sigmoidal activation function can be regarded as a generalization of a classical RBF neuron. Multi-layer networks based on hypersphere neurons are therefore similar to RBF networks of the same complexity. This will be explained in some detail later on. The main advantages of such a hypersphere neuron over a standard perceptron are the following:

- A hypersphere with infinite radius becomes a hyperplane. Since the hypersphere representation used is homogeneous, hyperspheres with infinite radius can be represented through finite vectors. Therefore, a standard perceptron is just a special case of a hypersphere neuron.
- The VC-dimension [1] of a hypersphere neuron for a 1-dimensional input space is three and not two, as it is for a standard perceptron. However, for higher input dimensions, the VC-dimensions of a hypersphere neuron and a standard perceptron are the same.

Although the VC-dimensions of a hypersphere neuron and a standard perceptron are the same for input dimensions higher than one, it is advantageous to use a hypersphere neuron, if the classification of the data is isotropic about some point in the input space. See [2] for more details.

The remainder of this paper is structured as follows. First the representation of hyperspheres used is described in some more detail. Then some important aspects concerning the actual implementation of a hypersphere neuron in a single- and multi-layer network are discussed. The comparison to classical RBF neurons is made. Afterwards some experiments with the Iris data set and the two spirals benchmark are presented. In a further experiment the abilities of a hypersphere multi-layer perceptron as classifier are tested on some real data taken from a particular computer vision problem. Finally, some conclusions are drawn from this work.

2 The Representation of Hyperspheres

There is not enough space here to give a full treatment of the mathematics involved. Therefore, only the most important aspects will be discussed. For a more detailed introduction see [10, 11].

Consider the Minkowski space $\mathbb{R}^{1,1}$ with basis $\{e_+, e_-\}$, where $e_+^2 = +1$ and $e_-^2 = -1$. The following two null-vectors can be constructed from this basis, $e_\infty := e_- + e_+$ and $e_0 := \frac{1}{2}(e_- - e_+)$, such that $e_\infty^2 = e_0^2 = 0$ and $e_\infty \cdot e_0 = -1$.

Given a n-dimensional Euclidean vector space $\mathbb{R}^n$, the conformal space $\mathbb{R}^{n+1,1} = \mathbb{R}^n \otimes \mathbb{R}^{1,1}$ can be constructed. Such a conformal space will also be denoted as $\mathbb{ME}^n \equiv \mathbb{R}^{n+1,1}$. A vector $\mathbf{x} \in \mathbb{R}^n$ may be embedded in conformal space as

$$X = \mathbf{x} + \tfrac{1}{2}\,\mathbf{x}^2\,e_\infty + e_0, \tag{1}$$

such that $X^2 = 0$. It may be shown that this embedding represents the stereographic projection of $\mathbf{x} \in \mathbb{R}^n$ onto an appropriately defined projection sphere in $\mathbb{ME}^n$. Note that the embedding is also homogeneous, i.e. αX, with $\alpha \in \mathbb{R}$, represents the same vector $\mathbf{x}$ as X. In other words, any vector $A \in \mathbb{ME}^n$ that lies in the null space of X, i.e. satisfies $A \cdot X = 0$, represents the same vector $\mathbf{x}$.

The nomenclature e_0 and e_∞ is motivated by the fact that the origin of $\mathbb{R}^n$ maps to e_0 when using equation (1). Furthermore, as $|\mathbf{x}|$ with $\mathbf{x} \in \mathbb{R}^n$ tends to infinity, the dominant term of the mapping of $\mathbf{x}$ into $\mathbb{ME}^n$ is e_∞.

A null-vector in $\mathbb{ME}^n$ whose e_0 component is unity, is called *normalized*. Given the normalized null-vector X from equation (1) and $Y = \mathbf{y} + \frac{1}{2}\,\mathbf{y}^2\,e_\infty + e_0$, it can be shown that $X \cdot Y = -\frac{1}{2}(\mathbf{x} - \mathbf{y})^2$. That is, the scalar product of two null-vectors in conformal space, gives a distance measure of the corresponding Euclidean vectors. This forms the foundation for the representation of hyperspheres. A normalized hypersphere $S \in \mathbb{ME}^n$ with center $Y \in \mathbb{ME}^n$ and radius $r \in \mathbb{R}$ is given by $S = Y - \frac{1}{2}\,r^2\,e_\infty$, since then

$$X \cdot S = X \cdot Y - \tfrac{1}{2}\,r^2\;X \cdot e_\infty = -\tfrac{1}{2}(\mathbf{x} - \mathbf{y})^2 + \tfrac{1}{2}\,r^2, \tag{2}$$

and thus $X \cdot S = 0$ iff $|\mathbf{x} - \mathbf{y}| = |r|$. That is, the null space of S consists of all those vectors $X \in \mathbb{ME}^n$ that represent vectors in $\mathbb{R}^n$ that lie on a hypersphere. It can also be seen that the scalar product of a null-vector X with a normalized hypersphere S is negative, zero or positive, if X is outside, on or inside the hypersphere. Scaling the normalized hypersphere vector S with a scalar does not change the hypersphere it represents. However, scaling S with a negative scalar interchanges the signs that indicate inside and outside of the hypersphere.

The change in sign of $X \cdot S$ between X being inside and outside the hypersphere, may be used to classify a data vector $\mathbf{x} \in \mathbb{R}^n$ embedded in $\mathbb{ME}^n$. That is, by interpreting the components of S as the weights of a perceptron, and embedding the data points into $\mathbb{ME}^n$, a perceptron can be constructed whose decision plane is a hypersphere.

From the definition of a hypersphere in $\mathbb{ME}^n$ it follows that a null-vector $X \in \mathbb{ME}^n$ may be interpreted as a sphere with zero radius. Similarly, a vector in $\mathbb{ME}^n$ with no e_0 component represents a hypersphere with infinite radius, i.e. a hyperplane.

3 Implementation

The propagation function of a hypersphere neuron may actually be implemented as a standard scalar product, by representing the input data as follows. Let a data vector $\mathbf{x} = (x_1, x_2, \ldots, x_n) \in \mathbb{R}^n$ be embedded in $\mathbb{R}^{n+2}$ (*not* $\mathbb{ME}^n$) as

$\boldsymbol{X} = (x_1, \ldots, x_n, -1, -\frac{1}{2}\,\mathbf{x}^2) \in \mathbb{R}^{n+2}$. Then, representing a hypersphere $S = \mathbf{c} + \frac{1}{2}\,(\mathbf{c}^2 - r^2)e_\infty + e_0 \in \mathbb{ME}^n$ in $\mathbb{R}^{n+2}$ as $\boldsymbol{S} = (c_1, \ldots, c_n, \frac{1}{2}(\mathbf{c}^2 - r^2), 1)$, one finds that $X \cdot S = \boldsymbol{X} \cdot \boldsymbol{S}$. During the training phase of a hypersphere neuron, the components of $\boldsymbol{S}$ are regarded as independent, such that $\boldsymbol{S}$ may simply be written as $\boldsymbol{S} = (s_1, \ldots, s_{n+2})$.

Therefore, a hypersphere neuron may be regarded as a standard perceptron with a second 'bias' component. Of course, the input data must be of a particular form. That is, after embedding the input data in $\mathbb{R}^{n+2}$ appropriately, a decision plane in $\mathbb{R}^{n+2}$ represents a decision hypersphere in $\mathbb{R}^n$. In this respect, it is similar to a kernel method, where the embedding of the data in a different space is implicit in the scalar product.

The computational complexity of a hypersphere neuron is as follows. Apart from the standard bias, which is simply set to unity, the magnitude of the input data vector has to be evaluated. However, for a multi-layer hypersphere network, this magnitude only has to be evaluated once for each layer. In terms of complexity this compares to adding an additional perceptron to each layer in a MLP.

The multi-layer perceptron based on hypersphere neurons (MLHP) can be interpreted as an extended RBF network with an equal number of neurons.

Let the activation function of the hypersphere neuron be the sigmoidal function $\sigma(\lambda, z) = (1 + \exp(-\lambda z))^{-1}$. In general a hypersphere neuron represents a non-normalized hypersphere. Therefore the propagation function becomes $X \cdot \kappa S$, $\kappa \in \mathbb{R}$ (cf. equation (2)), see [2] for more details. Thus the output y of the neuron can be written as

$$y = \sigma(\lambda, (X \cdot \kappa S)) = \sigma(\lambda, -\tfrac{1}{2}\kappa(\|\mathbf{x} - \mathbf{c}\|_2^2 - r^2)) = \frac{1}{1+\exp(\frac{1}{2}\lambda\kappa(\|\mathbf{x}-\mathbf{c}\|_2^2 - r^2))} \qquad (3)$$

This equation shows, that the output is an isotropic function similar to a Gauss with extremum at $\mathbf{x} = \mathbf{c}$ and asymptotical behavior for $\|\mathbf{x} - \mathbf{c}\|_2 \to \infty$.

For positive values of κ, y is positive for points lying within the hypersphere and negative for points lying outside the hypersphere. For negative values of κ we obtain the inverse behavior.

Not only the position of the extremum of this functions (center of hypersphere) but also the size of the support area (radius of hypersphere) can be learned.

4 Experiments

In an initial experiment, the simplest form of a multi-layer hypersphere perceptron, a single-layer perceptron, was tested on Fisher's Iris data set [6]. This set consists of 150 four-dimensional data vectors, which are classified into three classes. Visualizing the data [8] shows that one class can be separated linearly from the other two. The two remaining classes are, however, somewhat entangled. The data set was separated into a training data set of 39 randomly chosen data vectors and a test data set of the remaining 111 data vectors. A standard

single-layer perceptron (SLP) and a single-layer hypersphere perceptron (SLHP) were then trained on the training data set in two different configurations. In the first configuration (C1) the network consisted of one layer with three neurons, each representing one class. The classes were coded by 3 three-dimensional vectors $(1,0,0)$, $(0,1,0)$ and $(0,0,1)$, respectively. In the second configuration (C2) there was a single layer with only two neurons, whereby the three classes were coded in a binary code. That is, the output of the two neurons had to be $(1,0)$, $(0,1)$ and $(1,1)$, respectively, to indicate the three classes.

Table 1 shows the number of *incorrectly* classified data vectors after training in configuration C1 and C2, respectively, for the training and the test data set using the SLP, the SLHP and RBF networks.

Network	C1 Train. Data	C1 Test Data
SLHP	0	7
SLP	0	2
RBF	2	11

Network	C2 Train. Data	C2 Test Data
SLHP	0	7
SLP	9	31
RBF	10	20

Table 1. Comparison of classification results for SLHP, SLP and RBF on IRIS data.

It can be seen that both the SLP and the SLHP in C1, classify the training data perfectly. However, the SLP is somewhat better in the classification of the test data set. For C2, where only two neurons were used, the SLP cannot give an error free classification of the training data set. This is in contrast to the SLHP where an error free classification is still possible. Also for the test data set the SLHP gives much better results than the SLP. In fact, the SLHP does equally well with two and with three neurons. The results in C2 basically show that the data set cannot be separated into three classes by two hyperplanes. However, such a separation is possible with two hyperspheres. Although RBF networks contain two layers in contrast to the tested single layered models, they classify worse with the same amount of neurons in the hidden layer. In this experiment one needs at least ten neurons in the hidden layer of a RBF network to achieve similar results as with the SLHP.

In the second experiment the two spirals benchmark [5] was used, to compare a MLHP with a classical MLP and a RBF network. The task of this benchmark is to learn to discriminate between two sets of training points, which lie on two distinct spirals in the 2D plane. These spirals coil three times around the origin and around one another. This can be a very difficult task for back-propagation networks and comparable networks [9, 15].

Figure 1 shows the results of training for two-layer networks (i.e. one hidden layer) with classical perceptrons (MLP), hypersphere neurons (MLHP) and a RBF network. MLP and MLHP were trained with a backpropagation-algorithm. For each kind of network the minimal amount of neurons needed for almost perfect classification is taken for the visualization. The MLHP with 10 neurons

Fig. 1. Two spirals benchmark. Visualization of nearly perfect classification for different network types. White and black colors represent the two classes, that are to be learned. Gray color represents an area of unreliable decision. Left - MLHP with 10 neurons in hidden layer; Middle - MLP with 60 neurons in hidden layer; Right - RBF with 80 neurons in hidden layer.

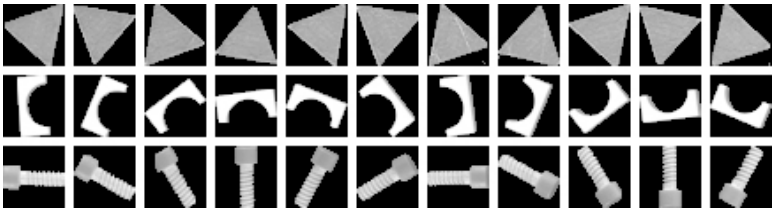

Fig. 2. Each object ist automatically detected, cropped and rescaled to a size of 35×35 pixels.

in the hidden layer can do perfect classification (100%). To achieve the same result a RBF network with 80 neurons in the hidden layer is required. A SLP with 60 neurons in the hidden layer can do nearly perfect classification (97%).

In the third experiment the classification abilities of MLHPs were tested on real data. The goal of this experiment was to associate an object (top view) with one of three given classes: screw, bridge or triangle. The data was coded in the following way.

In a preprocessing stage for each object that was to be classified, the data was generated from 360 top views of the object, whereby the object was rotated in one degree steps. The object was automatically detected, cropped and rescaled to a size of 35×35 pixels. Some views of the objects used are shown in figure 2. For further processing the images were interpreted as vectors of length 1225.

For each set of 360 data vectors, a PCA was performed (figure 3). Then all data vectors from all three classes were projected onto the first three principal components of the bridge. The resulting three dimensional data is visualized in figure 4. The associated classes were coded in a two-dimensional binary code $(1, 0)$, $(0, 1)$ and $(1, 1)$. From 1080 data vectors, 360 were taken for training and 720 for testing. Different types of networks were tested.

The best results (in relation to number of neurons) were achieved by a MLHP with two neurons in the hidden layer. For similar results a MLP with three

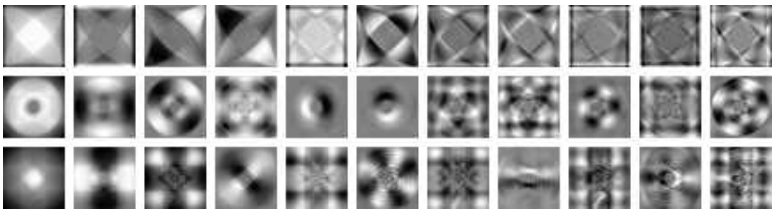

Fig. 3. Mean value and first ten principle components for triangle (top), bridge (middle) and screw (bottom).

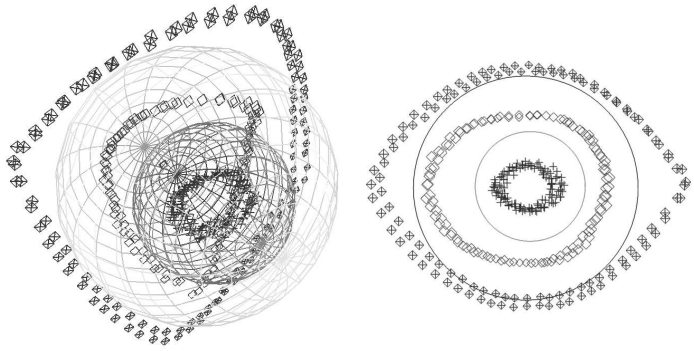

Fig. 4. Left – 3D-visualization of the classification (crosses - triangle, diamonds - bridge, crossed diamonds - screw). The two spheres represent the decision surfaces of the hypersphere neurons. Right – Projecting the data onto two principle components, demonstrates that each of the three classes builds a compact area in the input space and can be easily separated by two hyperspheres.

neurons in the hidden layer or a RBF network with 8 neurons was necessary. This result was expected due to the compactness of the classes, that had to be separated.

5 Conclusions

In this paper a higher-order neuron was presented which has the effect of placing a decision hypersphere in the input space, whereas a standard perceptron uses a hyperplane to linearly separate the input data. It was shown that a hypersphere neuron may also represent a hypersphere with infinite radius, i.e. a hyperplane, and thus includes the case of a standard perceptron. Advantages that may be gained by using hypersphere neurons, are the possibility to classify compact regions with a single neuron in n-dimensions, while the computational complexity is kept low. A single-layer hypersphere perceptron was tested and compared to a standard single-layer perceptron on the Iris data of R.A. Fisher. The data could be successfully classified with two hypersphere neurons. At least three standard

neurons or a RBF network with ten neurons in the hidden layer were necessary to achieve similar results. Furthermore MLP, MLHP and RBF networks were tested with the two spirals benchmark. Also in this case better results were achieved with hypersphere neurons than with a classical MLP or RBF network. In a real data scenario the advantages of a MLHP were also shown. This demonstrates that using hypersphere neurons is advantageous for certain types of data.

Acknowledgment

This work has been supported by DFG Graduiertenkolleg No. 357 and by EC Grant IST-2001-3422 (VISATEC).

References

1. Y. S. Abu-Mostafa. The Vapnik-Chervonenkis dimension: Information versus complexity in learning. *Neural Computation*, 1(3):312–317, 1989.
2. V. Banarer, C. Perwass, and G. Sommer. The hypersphere neuron. In *11th European Symposium on Artificial Neural Networks, ESANN 2003, Bruges*, pages 469–474. d-side publications, Evere, Belgium, 2003.
3. S. Buchholz and G. Sommer. A hyperbolic multilayer perceptron. In S.-I. Amari, C.L. Giles, M. Gori, and V. Piuri, editors, *International Joint Conference on Neural Networks, IJCNN 2000, Como, Italy*, volume 2, pages 129–133. IEEE Computer Society Press, 2000.
4. G. Cybenko. Approximation by superposition of a sigmoidal function. *Mathematics of Control, Signals and Systems*, 2:303–314, 1989.
5. S. E. Fahlman and C. Lebiere. The cascade-correlation learning architecture. In D. S. Touretzky, editor, *Advances in Neural Information Processing Systems*, volume 2, pages 524–532, Denver 1989, 1990. Morgan Kaufmann, San Mateo.
6. R. A. Fisher. The use of multiple measurements in axonomic problems. *Annals of Eugenics 7*, pages 179–188, 1936.
7. K. Hornik. Approximation capabilities of multilayer feedforward neural networks. *Neural Networks*, 4:251–257, 1990.
8. L. Hoyle. http://www.ku.edu/cwis/units/IPPBR/java/iris/irisglyph.html.
9. K.J. Lang and M.J. Witbrock. Learning to tell two spirals apart. In D.S. Touretzky, G.E. Hinton, and T. Sejnowski, editors, *Connectionist Models Summer School*. Morgan Kaufmann, 1988.
10. H. Li, D. Hestenes, and A. Rockwood. Generalized homogeneous coordinates for computational geometry. In G. Sommer, editor, *Geometric Computing with Clifford Algebra*, pages 27–52. Springer-Verlag, 2001.
11. H. Li, D. Hestenes, and A. Rockwood. A universal model for conformal geometries. In G. Sommer, editor, *Geometric Computing with Clifford Algebra*, pages 77–118. Springer-Verlag, 2001.
12. H. Lipson and H.T. Siegelmann. Clustering irregular shapes using high-order neurons. *Neural Computation*, 12(10):2331–2353, 2000.
13. M. Minsky and S. Papert. *Perceptrons*. Cambridge: MIT Press, 1969.
14. H. Ritter. Self-organising maps in non-Euclidean spaces. In E. Oja and S. Kaski, editors, *Kohonen Maps*, pages 97–108. Amer Elsevier, 1999.
15. A. Wieland and S. E. Fahlman. http://www.ibiblio.org/pub/academic/computer-science/neural-networks/programs/bench/two-spirals, 1993.