The Hypersphere Neuron

Vladimir Banarer, Christian Perwass, Gerald Sommer

Institut für Informatik, CAU Kiel, Preusserstr. 1-9, 24105 Kiel, Germany
vlb, chp, 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, in this setup, a reliability measure can be associated with data points in a straight forward way.

1 Introduction

The basic idea behind a single standard perceptron is that it separates its input space into two classes by a hyperplane [8]. For most practical purposes such a linear separation is, of course, not sufficient. In general, data is to 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 [3, 4].

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 [2], tensor neurons [7] and hyperbolic SOMs [9]. Typically, the more complex the decision surface a neuron has is, the higher its computational complexity. It is hoped

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

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 [6]. 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. 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 of 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 orientation invariant about some point in the input space. For example, let $\{\mathbf{x}_i\} \subseteq \mathbb{R}^n$ and $\{\mathbf{y}_i\} \subseteq \mathbb{R}^n$ denote the input vectors of two different classes. If there exists a point $\mathbf{c} \in \mathbb{R}^n$, such that $\max_i |\mathbf{x}_i - \mathbf{c}| < \min_i |\mathbf{y}_i - \mathbf{c}|$ or $\max_i |\mathbf{y}_i - \mathbf{c}| < \min_i |\mathbf{x}_i - \mathbf{c}|$, then the classification of the data is basically a 1-dimensional problem, and the two classes can be separated by a single hypersphere, independent of the input dimension. A multi-layer hypersphere perceptron (MLHP), therefore separates the input space into regions where the classification is orientation invariant. Figure 1 gives an example of this.

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. 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 [5, 6].

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

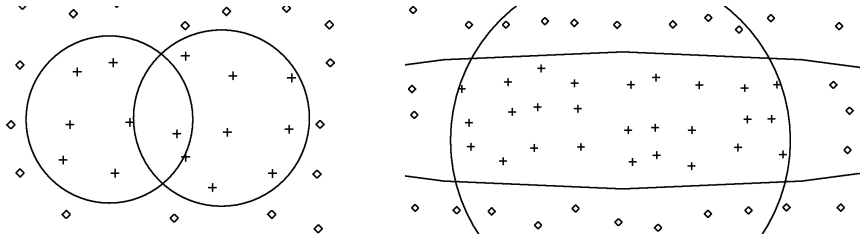

Figure 1: Learning of two different data sets by two 2-layer networks. Data points of one class lying as a compact cluster within the other class. The separation is done with two (left) respectively three (right) hypersphere neurons in the first layer and one neuron in the second layer. A classical MLP with two layers needs at least four neurons in the first layer.

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 an 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. This also motivates the nomenclature e_0 and e_∞, since e_0 represents the origin of $\mathbb{R}^n$ and e_∞ the point at infinity. A null-vector in $\mathbb{ME}^n$ whose e_0 component is unity, is called *normalized*. Given a second normalized null-vector $Y = \mathbf{y} + \frac{1}{2}\,\mathbf{y}^2\, e_\infty + e_0$, 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 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 plane. In fact, given two normalized null-vectors $X, Y \in \mathbb{ME}^n$, $X - Y$ represents the plane located half way between $\mathbf{x}$ and $\mathbf{y}$ with normal $\mathbf{x}-\mathbf{y}$. Such a plane still has a sidedness, that is, the scalar product of a null-vector with a plane is either positive, zero or negative depending on whether the test vector is off to one side, on the plane or off to the other side. Therefore, a hypersphere neuron may also represent 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 $\vec{X} = (x_1, \ldots, x_n, -1, -\frac{1}{2}\,\mathbf{x}^2) \in \mathbb{R}^{n+2}$. 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 $\vec{S} = (c_1, \ldots, c_n, \frac{1}{2}(\mathbf{c}^2 - r^2), 1)$, one finds that $X \cdot S = \vec{X} \cdot \vec{S}$. During the training phase of a hypersphere neuron, the components of $\vec{S}$ are regarded as independent, such that $\vec{S}$ may simply be written as $\vec{S} = (s_1, \ldots, s_{n+2})$. This embedding also allows hyperspheres with imaginary radii. However, since such a hypersphere cannot include any points, it does not produce spurious solutions. It may indeed contribute to a successful learning.

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.

It follows from equation (2), that the value of the scalar product of a data point with a normalized hypersphere is bounded by the radius of the hypersphere for data points lying within (class $\mathcal{I}$), but it is not limited for data points lying outside (class $\mathcal{O}$). Since the result of this scalar product is the input to an activation function, the type of activation function appears to have an influence on how large the radius of a hypersphere will tend to be. However, since the weights of a hypersphere neuron are treated as independent components, they represent an un-normalized hypersphere. The overall scale factor of the hypersphere vector then allows the scalar product of the hypersphere

with points lying within it to take on arbitrarily large values.

For example, denote by $X \in \mathbb{ME}^n$ the representation of data point $\mathbf{x} \in \mathbb{R}^n$, and denote by $S \in \mathbb{ME}^n$ the representation of a hypersphere neuron with center $\mathbf{c} \in \mathbb{R}^n$, radius $r \in \mathbb{R}^+$ and scale $\kappa \in \mathbb{R}\backslash\{0\}$. Furthermore, let the activation function of the hypersphere neuron be the sigmoidal function $\sigma(\lambda, z) = (1 + e^{-\lambda z})^{-1}$. Training the hypersphere neuron to classify $\mathbf{x}$ as belonging to $\mathcal{I}$ then means to vary $\mathbf{c}$, r and κ, such that $\sigma(\lambda, X \cdot S) > 1 - \epsilon$, where $\epsilon \in \mathbb{R}^+$ gives the decision threshold. If $\mathbf{x}$ is to be classified as belonging to $\mathcal{O}$, then one demands that $\sigma(\lambda, X \cdot S) < \epsilon$. With respect to the radius this means that

$$r^2 > \frac{2}{\lambda\kappa} \ln \frac{1-\epsilon}{\epsilon} + (\mathbf{c} - \mathbf{x})^2 \quad \text{if} \quad \mathbf{x} \in \mathcal{I}, \tag{3}$$

$$r^2 < \frac{2}{\lambda\kappa} \ln \frac{\epsilon}{1-\epsilon} + (\mathbf{c} - \mathbf{x})^2 \quad \text{if} \quad \mathbf{x} \in \mathcal{O}, \tag{4}$$

It can be seen that for fixed ϵ, $\mathbf{c}$ and κ, the radius of the hypersphere depends on the parameter λ of the sigmoid function. The effect of this is that the smaller λ, the larger the radius of the hypersphere tends to be. Note that the above equations are valid for $\kappa > 0$, whence $X \cdot S = \frac{1}{2}|\kappa| \left(r^2 - (\mathbf{x} - \mathbf{y})^2\right)$. However, for $\kappa < 0$, this becomes $X \cdot S = \frac{1}{2}|\kappa| \left((\mathbf{x} - \mathbf{y})^2 - r^2\right)$, such that data points inside S belong to class $\mathcal{O}$ and outside S to class $\mathcal{I}$.

We can introduce a measure for the reliability of a particular data point by extending data points in the following way. Given a data point $\mathbf{x}$ with some confidence measure r_{conf}, it is embedded in $\mathbb{ME}^n$ as $X_{\text{conf}} = \mathbf{x} + \frac{1}{2}\left(\mathbf{x}^2 + r_{\text{conf}}^2\right) e_\infty + e_0$. This is equivalent to a hypersphere with imaginary radius. It will therefore be called an imaginary hypersphere. The scalar product between a hypersphere S and X then yields,

$$S \cdot X_{\text{conf}} = \tfrac{1}{2}\Big(r^2 - \big((\mathbf{c} - \mathbf{x})^2 + r_{\text{conf}}^2\big)\Big). \tag{5}$$

That is, the vector $\mathbf{x}$ appears to be further away from the center $\mathbf{c}$ than it actually is. Therefore, a training algorithm will try to place a decision hypersphere such that $\mathbf{x}$ lies further to the inside of the hypersphere's surface, than without confidence. This effect is shown in figure 2.

4 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. The synthetic experiments presented in this paper give examples where the use of a hypersphere neuron is advantageous.

Figure 2: Position of the decision hypersphere can be influenced by confidence. Left picture shows the position of decision hypersphere (black circle) for uniformly distributed confidences (grey circles). After increasing of confidence for left bottom point, the decision circle is moved in such a way, that the affected point is placed further inside.

The concept of the hypersphere neuron may be extended to other geometric entities. In conformal space $\mathbb{ME}^n$ a hypersphere basically represents an $(n-1)$-dimensional subspace. Using the Clifford algebra of $\mathbb{ME}^n$, also lower dimensional subspaces can be expressed in a linear fashion, which could extend the set of decision surfaces available.

References

[1] Y. S. Abu-Mostafa. The Vapnik-Chervonenkis dimension: Information versus complexity in learning. *Neural Computation*, 1(3):312–317, 1989.

[2] 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.

[3] G. Cybenko. Approximation by superposition of a sigmoidal function. *Mathematics of Control, Signals and Systems*, 2:303–314, 1989.

[4] K. Hornik. Approximation capabilities of multilayer feedforward neural networks. *Neural Networks*, 4:251–257, 1990.

[5] 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.

[6] 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.

[7] H. Lipson and H.T. Siegelmann. Clustering irregular shapes using high-order neurons. *Neural Computation*, 12(10):2331–2353, 2000.

[8] M. Minsky and S. Papert. *Perceptrons*. Cambridge: MIT Press, 1969.

[9] H. Ritter. Self-organising maps in non-Euclidean spaces. In E. Oja and S. Kaski, editors, *Kohonen Maps*, pages 97–108. Amer Elsevier, 1999.