FEATURE SELECTION FOR RBF NETWORKS

Jürgen Paetz

J.W. Goethe-Universität Frankfurt am Main,
Fachbereich Biologie und Informatik, Institut für Informatik
Robert-Mayer-Straße 11-15, D-60054 Frankfurt am Main, Germany
email: paetz@informatik.uni-frankfurt.de

ABSTRACT

Radial basis function networks (RBFN) can be used for data classification. We present an a-posteriori feature selection method for trained RBFN that is not related to the specific learning procedure as long as every neuron belongs to exactly one class. Considering the center positions, the radii, the weights and the class labels of the neurons, we can easily calculate an index for feature selection that is based on one-dimensional projections. We present examples on different data sets by using Berthold and Diamond's RBFN.

1. INTRODUCTION

In the last decade the paradigm of RBFN has become an accepted, useful technique for data approximation and classification. Many variants have been proposed that are often specialized on an application task (e.g. [1], [2], [3]). One interesting variant is the RBFN with dynamic decay adjustment (RBFN-DDA) [4], [5].

RBFN-DDA is a dynamically growing neural network that adapts the numbers of neurons to the data space. The network is well suited for a-posteriori feature selection because it uses separate neurons for every class. Thus, the class information and the overlapping behavior of neurons from trained networks can be evaluated directly for feature selection. Other methods for feature selection [6] do not use network properties (a-priori filter such as information gain) or are costly (wrapper approach).

We review the RBFN-DDA and discuss its usage in Sect. 2. We explain in detail our new feature selection method that is based on projections of the radial basis functions to intervals in the one-dimensional space in Sect. 3. We demonstrate the usefulness of our Interval Projection Method (IPM) by evaluating benchmark data in Sect. 4. We end with a conclusion in Sect. 5.

2. REVIEW OF RBFN WITH DYNAMIC DECAY ADJUSTMENT

We describe the learning procedure RBFN-DDA [4], [5]. Originally RBFN-DDA was proposed as an extension of static probabilistic neural networks (PNN) [7] by using techniques of Restricted Coulomb Energy Networks (RCEN) [8]. PNN uses one neuron for every data sample, so that memory requirements are very high. RCEN training is performed by an emulation of the repulsion reaction of electrically charged particles (Coulomb’s law), but its classification performance was not sufficient [4], [5]. RBFN-DDA combines the ideas to a dynamically growing network using less neurons than PNN with a reliable classification performance.

We give a description of the RBFN-DDA and its learning algorithm. In Fig. 1 its network topology is shown.

Figure 1: RBFN-DDA topology with neurons separately for every class.
Algorithm 1: (RBFN-DDA)

Parameters (cf. Fig. 1):
- \( p^c_i \): \( i \)-th neuron of class \( c \), \( z^c_i \): center of neuron \( p^c_i \), \( r^c_i \): radius of neuron \( p^c_i \), \( w^c_i \): weight of neuron \( p^c_i \).
- \( R^c(x) \): activation of neuron \( p^c_i \) for input \( x \).
- \( x \): input sample, normalized in \([0,1]\) in each dimension.
- \( m_c \): number of neurons for class \( c \).
- \( \theta^- \), \( \theta^+ \): parameters for controlling the size of overlapping regions of neurons; set to \( \theta^- = 0.4 \) and \( \theta^+ = 0.2 \).

Training of one epoch (with permuted training data):
1. % reset weights
   for \( s = 1 \) to \( k \) do % all classes
      for \( i = 1 \) to \( m_c \) do % all neurons per class
         \( w^c_i = 0 \);
      end
   end
2. % consider all samples \((x;c)\):
   for \( r = 1 \) to number of samples do
      if \( R^c_r(x) \geq \theta^+ \) for a neuron then
      3. % sample covered by \( p^c_i \):
         \( w^c_i = w^c_i + 1 \);
      4. % commit new neuron:
         \( m_c = m_c + 1 \);
         \( w^c_m = 1 \);
         \( z^c_m = x \);
      5. % adapt radii:
         \( r^c_m = \min_{i,j \in \mathcal{C}} \sqrt{\frac{||x^c - z^c_i||^2}{\ln \theta^+}} \) ;
      end
6. % shrink radii of conflicting neurons:
   for \( s \neq c \) do
      for \( j = 1 \) to \( m_c \) do
         \( r^c_j = \min \left\{ r^c_j \sqrt{\frac{||x^c - z^c_i||^2}{\ln \theta^-}} \right\} ;
      end
   end
Activation for class \( c \) is calculated by the weighted sum of the activation \( R^c(x) \) of the neurons \( p^c_i \) with weights \( w^c_i \).

In the next section, trained RBFN are used as the starting point for our a-posteriori feature selection method.

### 3. INTERVAL PROJECTION METHOD FOR FEATURE SELECTION

We describe the idea of projecting radial basis functions to intervals in the one-dimensional space in Sect. 3.1. Based on these projections, we introduce our feature selection method in Sect. 3.2. We give a concrete example including all calculations for clearness in Sect. 3.3.

#### 3.1. Interval Projections

We use the notations that were introduced in the previous section for algorithm RBFN-DDA. Let us consider a neuron \( p^c_i \) with center \( z^c_i \), radius \( r^c_i \) and weight \( w^c_i \).

Definition 1:
For dimension \( d \) the interval projection \( P^c_{r^c} \) of neuron \( p^c_i \) is defined as:
\[
P^c_{r^c}(p^c_i) := [z^c_i - r^c_i, z^c_i + r^c_i]
\]

For two dimensions the interval projections are shown in Fig. 2.

![Figure 2: A neuron \( p \) (of a fixed class) and its interval projections. Other indices are omitted for brevity.](image)

These projections are the basis for our feature selection method in the next section.

#### 3.2. Interval Projection Method

For feature selection we have to calculate an index for every dimension. If the index is high for a dimension, then the dimension (resp. the variable) should be more relevant for classification. If the index is low for a dimension, then the dimension should be less relevant for classification. Additionally we demand such an index to be from the interval \([0,1]\).

We will derive such an index, considering the following fundamental idea: Interval projections in
relevant dimensions will less overlap, and interval projections in less relevant dimensions will more overlap.

To calculate such an overlap index, we have to take care about the weights of the neurons and a-priori class probabilities since pure interval overlaps consider only geometrical and not statistical properties of the data. How can we assign the statistical properties (weights, a-priori probabilities) to the geometrical ones (overlap)? For obtaining a feature selection index this problem has to be solved. We call our method interval projection method (IPM). It is described in the following.

Algorithm 2: (IPM)

1. consider a-priori probabilities:
   Let \( w_c \) be the absolute number of data samples of class \( c \).

   for all neurons \( p^c_i, c=1, \ldots, \text{number of classes} \) do
     \( v^c_i = \frac{w^c_i}{w_c} \);
   end

2. calculate interval projections for dimension \( d \):
   \( q = 1 \); % neuron counter, not considering class labels

   for all neurons \( p^c_i, c=1, \ldots, k, i=1, \ldots, m_i \) do
     \( P^c_i = P^c_i \oplus (p^c_i) = [z^c_{q}, r^c, z^c_{q} + r^c] \);
     \( v^c_i = v^c_i \);
     % we use \( P^c_i = [a^c, b^c] \) and \( v^c_i \) as short notations
     \( q = q + 1 \);
   end

3. order interval projections in list \( L \), and calculate entries in interval weight lists \( V^c, c=1, \ldots, k \):

   for every \( P^c_i = [a^c, b^c] \) of class \( c \) do
     if interval list \( L \) is empty then
       \( L = (a^c, b^c) \);
       \( V^c = (v, Nil) \); % weight list for class \( c \)
       \( V^c = (0, Nil) \) for all \( s \neq c \);
     else \( L = (a_1, \ldots, a_h) \) not empty (contains \( h \) elements)
     if \( a_{q}, b_{q} \neq a_s, g=1, \ldots, h \) then
       add interval borders \( a_{q}, b_{q} \) and/or \( b_{q} \) in list in ascending order;
     end
     end

4. calculate overlap index \( \varphi_d \):

   for every \( g \) do
     % consider every interval in list \( V^c \)
     \( \varphi_d = \frac{\sum l^c \cdot \sigma_c}{\sum l} \); % overlap index \( \in [0,1] \) for dimension \( d \)

   with \( l^c = [a^c, b^c] \) (length of interval).

5. calculate feature selection index \( \in [0,1] \) for dimension \( d \):

   \( \sigma_d = 1 - \varphi_d \);

The algorithm follows an intuitive idea. To understand better the more technically notation of the algorithm, we will calculate an example that will be visualized (cf. Fig. 3) in the next section.

3.3. Example

Let us consider nine two-dimensional neurons with the following parameters (center \( z(z_1, z_2) \); weight \( w \); class \( c \)):
\[
(0.3, 0.6; 8; 2), (0.3, 0.45; 6; 2), (0.3, 0.3; 6; 2), (0.45, 0.6; 5; 1),
(0.45, 0.45; 10; 1), (0.45, 0.3; 5; 1), (0.6, 0.6; 2; 2), (0.6, 0.45; 4; 2)
and (0.6, 0.3; 8; 2).
\]
All radii are set to 0.1. A-priori frequency of the samples is 15 for class 1 and 30 for class 2. Notice that every sample can count for more than one weight. Dividing the weights by 15 resp. 30, we obtain after step 1 of IMP: (0.3, 0.6; 0.52; 0.27), (0.3, 0.45; 0.2; 0.2), (0.3, 0.3; 0.2; 0.2), (0.45, 0.6; 0.33; 0.1), (0.45, 0.45; 0.67; 0.1), (0.45, 0.3; 0.33; 0.1), (0.6, 0.6; 0.07; 0.2), (0.6, 0.45; 0.15; 0.2) and (0.6, 0.3; 0.33; 0.2). These neurons are visualized in Fig. 3 where also the lists \( L \), \( V^c \), and \( V' \) from steps 2 and 3 are noted. This figure shows the intuitive character of the more formal notations of the algorithm.
The next step 4 of algorithm IPM requires the calculation of the weighted sum \( \varphi_d \) for every dimension, here \( d=1,2 \):

\[
\varphi_1 = \frac{0.05 \cdot 0.67 + 0.05 \cdot 0.57}{0.15 + 0.05 + 0.1 + 0.05 + 0.15} = 0.12 \tag{2}
\]

\[
\varphi_2 = \frac{0.15 \cdot (0.77 + 0.99) + 0.05 \cdot (0.92 + 0.8) + 0.1 \cdot 0.66}{0.15 + 0.05 + 0.1 + 0.05 + 0.15} = 0.84 \tag{3}
\]

As we expected, dimension 1 is much more relevant for correct classification (step 5: \( \varphi_1 = 0.12 \Rightarrow \sigma_1 = 0.88 \) than dimension 2 (step 5: \( \varphi_2 = 0.84 \Rightarrow \sigma_2 = 0.16 \)). If all weights of the neurons are set to 1 (a-priori frequency 3 resp. 6), then \( \varphi_1 = 0.93 \) and \( \varphi_2 = 0 \). If the neurons would not overlap, then \( \varphi_1 = 1 \) and \( \varphi_2 = 0 \).

### 4. EXPERIMENTS

We demonstrate our method by applying it to the benchmark datasets IRIS [9], Diabetes and Horse [10]. All the experiments are done with a randomly selected partition of the data into 50% training and 50% test data. The results are mean results of 8 repetitions, each with different random test data. In Table 1 we see the performance on test data, the number of neurons, the number of variables and the number of relevant variables identified by IPM - for every dataset.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Correct classified (%)</th>
<th>Neurons (std.)</th>
<th>Var.</th>
<th>Relevant var.</th>
</tr>
</thead>
<tbody>
<tr>
<td>IRIS</td>
<td>92.83 (4.73)</td>
<td>16.5 (2.6)</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>Diabetes</td>
<td>74.35 (1.76)</td>
<td>288.5 (6.1)</td>
<td>8</td>
<td>3</td>
</tr>
<tr>
<td>Horse</td>
<td>61.95 (1.51)</td>
<td>162.5 (5.2)</td>
<td>58</td>
<td>10</td>
</tr>
</tbody>
</table>

Table 1: Classification performance, number of neurons, number of variables (var.) and number of relevant variables, identified by IPM. Standard deviation (std.) is put in brackets.

For all datasets the number of dimensions can be reduced. In Table 2 we show the results of a repetition of classification for the datasets IRIS, Diabetes and Horse when trained only with the relevant variables.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Correct classified (%)</th>
<th>Neurons (std.)</th>
</tr>
</thead>
<tbody>
<tr>
<td>IRIS</td>
<td>94.50 (1.50)</td>
<td>11.8 (2.9)</td>
</tr>
<tr>
<td>Diabetes</td>
<td>73.11 (1.74)</td>
<td>247.0 (7.9)</td>
</tr>
<tr>
<td>Horse</td>
<td>61.95 (2.39)</td>
<td>127.3 (10.3)</td>
</tr>
</tbody>
</table>

Table 2: Classification performance and number of neurons with reduced number of variables. Standard deviation (std.) is put in brackets.

We discuss the results for the datasets IRIS, Horse (3 class problems) and Diabetes (2 class problem).

**IRIS**: IRIS data is very well classifiable. The feature selection indices are \( \sigma = (0.41,0.43,0.71,0.75) \). The index \( \sigma_5 \) is lower for dimension 1 and 2. In Table 2 the results, using dimensions 3 and 4 only, are shown. With 4.7 (28.5%) less neurons we obtain a similar classification performance (92.83% resp. 94.50%).

**Diabetes**: Diabetes is more difficult to classify than IRIS. The feature selection indices for Diabetes are \( \sigma = (0.40,0.60,0.31,0.48,0.38,0.52,0.38,0.38) \). Due to more overlapping in Diabetes no values are very low for one dimension. With the help of the order we choose the variables with the three highest values (dimensions 2, 4 and 6). Classification performance is again similar (74.35% resp. 73.11%). Using only three variables, we need 41.5 (14.4%) less neurons for the networks with lower dimension.

**Horse**: Horse has the highest number of dimensions, exactly 58. The highest \( \sigma_d \) is achieved for dimension \( d=21 \) (\( \sigma_{21} = 0.196 \)), the lowest for \( d=11 \) (\( \sigma_{11} = 0.081 \)). For the experiment in Table 2 we choose the variables with \( \sigma_d \geq 0.160 \), i.e. variables 4, 10, 12, 21, 33, 39, 44, 45, 51 and 57. Classification performance is identical (61.95%). In Table 2 we notice 35.2 (21.7%) less
neurons than in Table 1. For the Horse data standard deviation becomes higher after feature selection.

IPM has led to obvious dimension reduction (50%, 37.5% resp. 17.2% of all variables, cf. Table 1) using an order of feature significance. It helps also reducing network complexity (less number of neurons) without notable classification performance loss. If no dimension is absolutely unnecessary, then the order of \( \sigma_d \) helps identifying the most relevant dimensions. Remember that \( \sigma_d \) never becomes exactly 0 since there is always overlap between neurons due to algorithm 1.

5. CONCLUSION

We have presented a feature selection method based on interval projections. Our interval projection method (IPM) can be used for a-posteriori feature selection. A precondition for the usage of IPM is a trained RBFN that has neurons arranged separately for every class like RBFN-DDA. A-posteriori feature selection is useful since the actual trained network is evaluated, compared to other feature selection methods that estimate the relevance of features a-priori or that are costly.

We applied IPM to several benchmark problems. We achieved reliable dimension reduction resp. feature selection results using the resulting overlap index resp. feature selection index. Due to the fast learning algorithm of RBFN-DDA, the network can be trained using all the features. Then, IPM can be applied. Other a-posteriori feature selection algorithms for other networks have recently been proposed [11], [12]. These algorithms can be used especially when training is fast. An adaption of IPM to other neural networks, e.g. [2] could be work for the future. More detailed explorations and work on more examples is planned.

Note: Matlab source code will be made available online after conference meeting [13].

6. REFERENCES


APPENDIX

When is the condition \( \{a_d \leq \alpha_g \} \cap \{a_g > \alpha_g \} \neq \emptyset \) in step 3 of algorithm IPM fulfilled?

This is the case if \( \alpha_g < b_d \leq \alpha_{g+1} \) or \( \alpha_g < a_d < \alpha_{g+1} \) or \( (\{a_d \leq \alpha_g \} \land \{b_d > \alpha_{g+1} \}) \).