INNER PRODUCT NETWORKS

Research student: Gonçalo Xufre Silva[(a)];

Advisor: António José Rodrigues[(b)]

[(a)] Centro de Investigação Operacional – FCUL e Centro de Matemática – ISEL, Rua Conselheiro Emídio Navarro, Nº1, 1949-014 Lisboa, Portugal, Email: goncalo@dem.isel.ipl.pt;

[(b)] Centro de Investigação Operacional – Faculdade de Ciências da Universidade de Lisboa, 1749-016 Lisboa, Portugal, Email: ajrodrigues@fc.ul.pt;

1. Introduction

The two types of neural networks most used for supervised learning problems are multilayer perceptrons (MLPs) and radial basis function (RBF) networks. The main difference between them is that RBFs are linear in the parameters and MLPs are not. The only way to have MLPs linear in the parameters is to impose that the weights of the connections between input and hidden units are pre-defined and fixed during all the process of training (in a three layer network). Also, the output units have to be a linear combination of the hidden unit outputs. When we study a novel type of learning models, the support vector machines, we observe that after the learning process these models have a structure similarly to neural networks. They use a kernel function, $K(X,Y)$, that represents the inner product between $\phi(X)$ and $\phi(Y)$ where $\phi(\cdot)$ is a transformation of the training data to a higher dimension space. $K(X,Y)$ allow us to calculate the inner products between $\phi(X)$ and $\phi(Y)$ without having to know the explicitly form of the transformation function. The most used kernel functions are:

$$K(X,Y) = e^{\frac{\|X-Y\|^2}{2\sigma^2}} \quad (1.1)$$

$$K(X,Y) = \tanh(k \cdot X \cdot Y - \delta) \quad (1.2)$$

$$K(X,Y) = \left(X \cdot Y\right)^p \quad (1.3)$$

These functions satisfy the Mercer condition [Vapnik 95] that guarantees to represent an inner product for the transformation of the variables into a higher dimension space. In the case of (1.2) this is true only for some values of k and δ. A relevant difference between SVMs and neural networks is that the structure of a SVM doesn't have to be pre-defined but is determined during raining.

If we use the kernel (1.1) the SVM has a structure identical to a radial basis function. If the kernel used is (1.2) we have a structure similar to a multilayer perceptron where the weights form the connections between input and hidden units form the inner product between input training vectors. The observation of this last structure inspired us to think in a multilayer perceptron where the weights from the connections between input and hidden units are input training vectors. The only parameters to be estimated will be the weights from the connections between hidden and output units. In this way we have a multilayer perceptron that is linear in the parameters.

2. Inner product networks

An initialization method for the weights of a multilayer perceptron was proposed by [Denoeux 93] that consists in the use of the input training vectors after one type of normalization. This initialization procedure was motivated by the fact that if the training vectors were normalized their inner product reflects the distance between them. Using the training vectors (after normalization) for the weights of the connections between input and hidden units, the neural networks that we propose and will refer as Inner Product networks presents hidden units with local influence as RBFs units, but with the ability to influence the entire input training space has the MLPs. On the other hand, the network proposed is linear in the parameters that we have to estimated, what can be achieve with the calculation of the pseudo-inverse matrix. These networks have an activation function given by $f(X) = \tanh\left(k\left(1 - X \cdot W\right)\right)$, meaning that we have a model hyper parameter, k, that we need to define.

3. Application examples

In order to test the network proposed we used two artificial problems. The first one, a regression type problem, that we call "function detection problem", has an input training set of 100 points uniformly randomly chosen in the interval $[-5,5]$ where the desired outputs where generated by $y_i = 5\sin(x_i) + 0.3x_i^2 + \varepsilon_i$ where $x_i = [-5,5]$, and $\varepsilon_i \cap N(0,1)$ is gaussian noise, added to the generation model function. An Inner Product network was used in order to estimate the generation model of the data.

After calculating the weights for the connections between hidden and output units with the pseudo-inverse matrix , we obtain the network performance by calculating the medium square error (MSE) in a new set, the validation set, composed by 100 other points generated the same way as the training set.
In this first study of the Inner Product networks we were interested in finding out the influence of the model hyper parameter, k, used in the activation function. We analysed the relationship between the network performance (the MSE in the validation set) and the variation of k. In table 1 we have some of the results obtained.

Value of k	MSE in the training set	MSE in the validation set
0.00001	14.330	15.208
0.0001	1.611	3.689
0.001	1.611	3.688
0.01	0.856	1.023
0.1	0.841	1.1747
1.0	0.647	213990.903

Table 1 – Variation of the network performance (MSE in the training and the validation set) in function of the value used for *k*.

It's possible to observe a regularization behaviour in the value of k. For small values of k the network doesn't have the capability to learn the structure of the data generation model (ex: $k = 0.00001$), on the other hand, for higher values of k the network presents a over learning behaviour (ex: $k = 0.00001$).The choice of the value of k becomes a critical point for the Inner Product networks. We are convicted that the optimal value, of k, is problem dependent and we suggest to use an intensive searching procedure with cross validation for choice criterion. Let us refer that for the "function detection problem" the optimal value found for k (in the sense of minimizing the MSE in the validation set) was 0.0014.

The second problem used was a classification type problem. The input data are $\Re^2$ values classified in one of two classes. The frontier separating the two classes forms a double "F". We generate 800 data points following a uniformly randomly distribution over the $[0,3]\times[0,4]$ rectangle. For the validation set, another 800 points were generated using the some procedure. Several Inner Product networks were trained for different values of k. For each case the MSE in the training and validation set were calculated as well the classification error in both sets. In table 2 we may observe the results obtained for some values of k.

Value of k	MSE in the training set	Classification error in the training set	MSE in the validation set	Classification error in the validation set
0.0001	0.378263	13.125%	0.375909	13.333%
0.001	0.319434	12.000%	0,316013	12.000%
0.01	0.266083	0.875%	0.291842	1.033%
0.1	0.208813	0.475%	0.238269	0.500%
1.0	0.118934	0.263%	0.145862	0.367%
5.0	0.062200	0.075%	0.561323	0.133%
10.0	0.041209	0.025%	0.300351	0.267%
15.0	0.026433	0.025%	17.203581	0.300%
20.0	0.019240	0.000%	99.613418	0.0433%

Table 2 – Variation of the MSE and classification error in the training and validation set, for some values of *k*.

References

[Denoeux93] T. Denoueux, R. Lengellé, "Initializing Back Propagation Networks With Prototypes" Neural Networks Vol. 6, pp. 351-363, 1993.

[Vapnik95] V. Vapnik, "The Nature of Statistical Learning Theory", Springer, 1995.

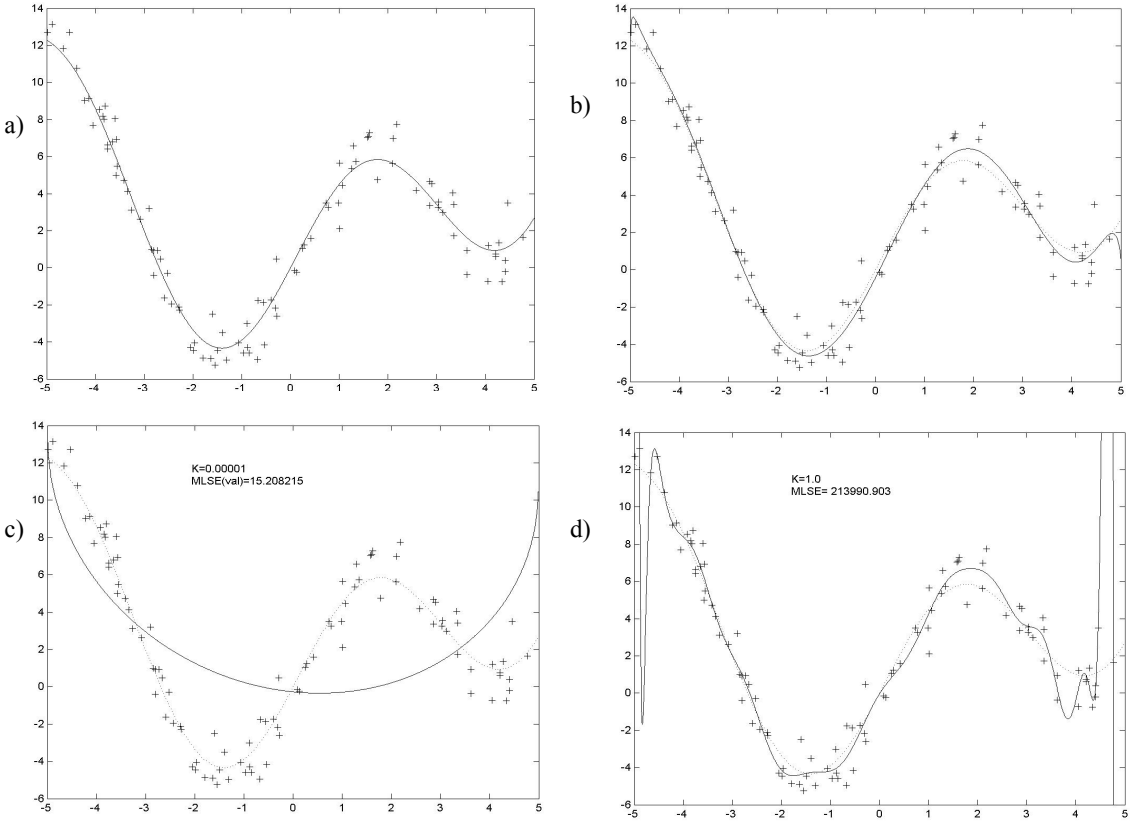

Figure 1 – a) Training data and data generating function. b), c) and d) Function estimated by an IP network whit k=0.0014, k=0.00001 and k= 1.0, respectively.

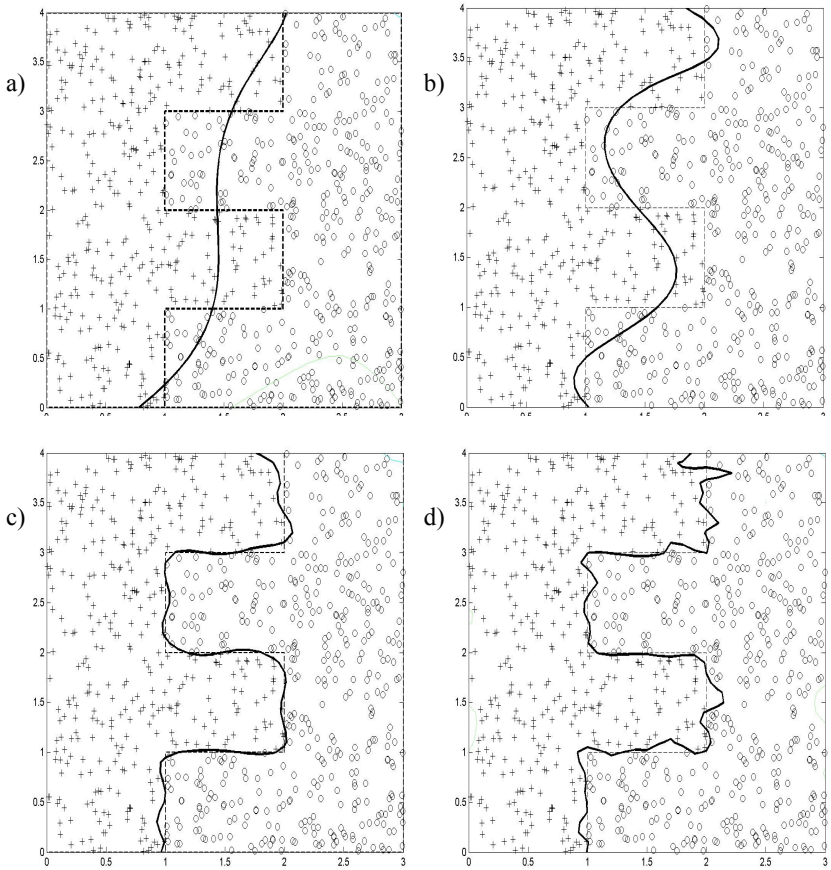

Figure 2 – Training data and estimated frontier between the two classes. a) k=0.001 b) k=0.01 c) k=5.0 and d) k=20.0