UNIVERSITY
OF TRENTO

DEPARTMENT OF INFORMATION AND COMMUNICATION TECHNOLOGY

38050 Povo – Trento (Italy), Via Sommarive 14
http://www.dit.unitn.it

THE DIGITAL KERNEL PERCEPTRON

Davide Anguita, Andrea Boni and Sandro Ridella

2002

Technical Report # DIT-02-0032

Also: to be published in Electronics Letters Vol. 38 No. 10 pp. 445 446, 2002

The Digital Kernel Perceptron

Davide Anguita, Andrea Boni, and Sandro Ridella.

In this paper, we show that a kernel-based perceptron can be efficiently implemented in digital hardware using very few components. Despite its simplicity, the experimental results on standard data sets show remarkable performance in terms of generalization error.

Introduction: A practical way to build non-linear classifiers is to map, via special *kernel* functions, the input space to a higher, possibly infinite, feature space where one can manipulate simpler linear operators [1]. This is a well-known theory that exploits the Reproducing Kernel Hilbert Space (RKHS) framework, and recently has been applied with success by the machine learning community. Support Vector Machines (SVMs), designed by Vapnik [2], represent one of the most successful examples of a learning machine based on a kernel method. Other algorithms, that follow the same underlying approach, have been successfully proposed. Here we discuss the use of the Rosenblatt's perceptron in the dual, kernel-based formulation: the Kernel Perceptron (KP). The choice of the KP is justified by recent studies that have revalued its role in classification tasks, showing the relation between its generalization ability and the sparsity of the final solution [3].

We show how to let the KP algorithm work only with integer values and suggest an efficient digital architecture for its implementation. We call this variant, the Digital

Kernel Perceptron (DKP). Tests on several data-sets show the good performance of our proposal in terms of architecture complexity and generalization performance, which make the KP an appealing approach for building VLSI-based learning systems.

The Kernel Perceptron: Here we face a classification problem, where a two-class training set, ($\underline{x}_1$,y_1), . . ., ($\underline{x}_n$, y_n), with $\underline{x}_i \in \Re^m$ and $y_i = \{+1,-1\}$ is considered. The simplest machine able to classify such a set, if linearly separable, is the perceptron:

$$f(\underline{x}) = \underline{w} \cdot \underline{x} + b \tag{1}$$

Rosenblatt's learning algorithm is well known and shown in Table 1a. The final vector of the weights can be represented as a linear combination of the input patterns, as follows:

$$\underline{w} = \sum_{i=1}^{n} a_i y_i \underline{x}_i \tag{2}$$

Such an observation permits one to change the domain of work from the vector $\underline{w}$ to its *dual* $\underline{a}$. As a result, one can exploit the well-known theory of kernels by using the following final structure of function f, which allows an implicit nonlinear transformation in a new feature space via the unknown mapping $\Phi(\cdot)$:

$$f(\underline{x}) = \sum_{i=1}^{n} a_i y_i \Phi(\underline{x}_i) \cdot \Phi(\underline{x}) + b = \sum_{i=1}^{n} a_i y_i K(\underline{x}_i, \underline{x}) + b \tag{3}$$

This kind of function is well known in the machine-learning community and used for classification and function approximation tasks [4], [5]. $K(,)$ is a kernel function that realizes a dot product in the feature space.

Some examples of typical kernels are: $K(\underline{x}_i, \underline{x}_j) = \underline{x}_i \cdot \underline{x}_j$ (linear), $K(\underline{x}_i, \underline{x}_j) = \exp\left(-\left\|\underline{x}_i - \underline{x}_j\right\|^2 / 2s^2\right)$ (RBF) and $K(\underline{x}_i, \underline{x}_j) = (\underline{x}_i \cdot \underline{x}_j + 1)^p$ (polynomial). In this work we focus on the RBF kernel and set $b = 0$, as in the higher feature space the absence of the bias does not affect, in practice, the performance of the machine (note that the feature space of Gaussian kernels is of infinite dimension [6]). The dual algorithm (KP) is very simple and summarized in Table 1b, where $q_{ij} = y_i y_j K(\underline{x}_i, \underline{x}_j)$.

The most important KP property, which permits us to build a very simple architecture, is its convergence theorem (a result by Novikoff [3]), which links the maximum number k of updates to the margin. This result applies to any choice of the updating step h, therefore we propose to choose $h = 1$, because it leads to an integer representation of the dual vector $\underline{a}$.

The DKP architecture: The architecture sketched in Figure 1 implements the DKP learning algorithm. The Processing Element (***PE***) accomplishes the task of computing the quantity $MSB_i = \mathrm{sgn}\left(\sum_{j=1}^{n} a_j q_{ij}\right)$, then, on the basis of the obtained value, the corresponding a_i will be updated according to the algorithm presented in Table 1b.

The very important advantage over a traditional non-linear perceptron lies in the fact that the non-linear mapping, implicitly embedded in the kernel matrix Q, can be processed a priori and stored in a RAM. In this way, any

computation of nonlinear functions, during learning, is avoided.

The DKP algorithm suggests that each update due to a misclassified point can be implemented by using simple saturated up-counters, as described in Figure 1. Each component *count* works as follows: when the enable-input *EN* is active high, at the following clock transition the counter increments itself, if the maximum value has not been reached. The *PE* is also very simple: the multiplier unit, which computes the product between a_j and q_{ij}, can exploit the fact that the operand stored in the register *REG* takes on only integer values. MSB_i is simply the most significant bit of the accumulator at the end of the elaboration. Following the DKP algorithm, each *MUX* connected to the *EN*-input of the corresponding counter is selected at each step by the controller (not shown in the figure), which supervises the correct flow of data. As a final remark note that the *UCPE* is a specific control unit that acts as the controller of the memory, and delivers at each step the elements a_j and q_{ij} to the *PE*.

Results: We tested the described architecture on different data sets. Each element q_{ij} is coded in fixed-point math. Our experiments show the effectiveness of the proposed method even when only few bits are used to code each q_{ij}. This is the expected behavior, as we are only interested in knowing the sign (instead of the exact value) of the quantity $\sum_j a_j q_{ij}$, for each pattern *i*.

The most remarkable performance was obtained on the SONAR data set. It is known that it consists of 104 learning

patterns and 104 test patterns of 60 features each. As noted in previous works, this is a difficult classification problem, due to its high dimensionality and the small overlapping of the training and test sets. Whereas a linear separator is expected to misclassify approximately 22% of the test patterns, a traditional RBF-SVM (floating-point solution) misclassifies 8 test patterns [7]. Our DKP, despite its simplicity, obtains the same performance, even when using only 3-bit counters and 3 bits to code each q_{ij}. Obviously, such a result does not allow us to claim the good generalization capability of the DKP, but it is important to note that it agrees with the theory developed in [3]. For the same problem, the Digital Support Vector Machine (DSVM) needs 24 bits for each element q_{ij} and 8 bits for a_i [7].

Another experiment has been carried out on the well-known two-dimensional synthetic data set considered in [8], consisting of 250 training and 1000 test patterns. The DKP misclassifies only 140 test patterns (14%), with $s^2 = 0.002$, 12-bits counters and 12 bit for coding each q_{ij}, whereas the DSVM (floating-point solution) misclassifies 155 test patterns (15.5%). Note that the fixed-point version of DSVM is not able to classify correctly the training set, with 24 bits for q_{ij} and 14 bits for a_i, making 2 and 159 training and test errors, respectively.

Conclusions: In this paper, we have proposed a simple digital counter-based architecture that implements a Kernel Perceptron. The experiments presented here show the

higher effectiveness of our method in terms of architecture complexity and generalization performance, when compared with a digital implementation of a SVM. It is important to point out that the on-going research on kernel-based methods suggests that many classifiers perform comparably, provided that a good kernel has been chosen; therefore, simple classifiers, like the DKP, are very appealing and show their superiority when targeting VLSI implementations.

References

[1] Aizerman, M. A., Braverman, E. M. and Rozonoer, L. I. 'Theoretical Foundations of the Potential Function Method in Pattern Recognition Learning,' Automation and Remote Control, 25, 821-837, 1964.

[2] Vapnik, V.N.: `The Nature of Statistical Learning Theory' (John Wiley & Sons, NY, USA, 2nd edition, 1999).

[3] Herbrich, R., Graepel, T., and Williamson, R.: 'From Margin to Sparsity' NIPS 13, 2001.

[4] Freund Y. and Shapire, R.E.: 'Large Margin Classification Using the Perceptron Algorithm,' Machine Learning, 37(3), 1999, 277-296.

[5] Friess, T.T. and Harrison, R.F.: 'A Kernel-Based Adaline for Function Approximation,' Intelligent Data Analysis, 3, 1999, 307-313.

[6] Burges C.J.C.: 'A Tutorial on Support Vector Machines for Pattern Recognition,' Data Mining and Knowledge Discovery, V. 2, 1--47, 1998.

[7] Anguita, D., Boni, A., and Ridella, S.: 'Learning Algorithm for Nonlinear Support Vector Machines Suited for Digital VLSI,' Electronics Letters, Vol. 35 (16), 1999.

[8] Ripley, B.D.: 'Pattern Recognition and Neural Networks' (Cambridge University Press, 1996).

Authors' affiliations:

Davide Anguita and Sandro Ridella (University of Genova, Dept. of Biophysical and Electronic Engineering, Via Opera Pia 11A, 16145 Genova, Italy).

Corresponding author:

Andrea Boni (University of Trento, Faculty of Engineering, Via Mesiano 77, 38050 Trento, Italy, tel: +39 0461 882080 Email: andrea.boni@ing.unitn.it)

Table 1. – (1a) Classic and (1b) kernel-based perceptron algorithms

Set:

$\underline{w} = 0, b = 0, k = 0$

Repeat until no mistakes occur

For i=1 to n do

Compute

$$MSB = \operatorname{sgn}\left[y_i \left(\sum_{j=1}^{m} w_j x_{ij} + b \right) \right]$$

If MSB<0 then

$\underline{w} = \underline{w} + \eta \underline{x}_i y_i$

$b = b + y_i$

$k = k + 1$

end for

1a

Set:

$\underline{a} = 0, k = 0$

Repeat until no mistakes occur

For i=1 to n do

Compute

$$MSB_i = \operatorname{sgn}\left(\sum_{j=1}^{n} a_j q_{ij} \right)$$

If MSB_i<0 then

$a_i = a_i + 1$

$k = k + 1$

end for

1b

Figure 1. – The digital architecture for the DKP.