ESANN'2001 proceedings - European Symposium on Artificial Neural Networks
Bruges (Belgium), 25-27 April 2001, D-Facto public., ISBN 2-930307-01-3, pp. 75-80

Intelligent Hardware for Identification and Control of Non–Linear Systems with SVM

Andrea Boni , Fabio Bardi [*]
University of Trento, Dept. of Materials Engineering,

Via Mesiano, 77, I–38050 Povo (TN), Italy,
email: andrea.boni@ing.unitn.it

Abstract. Support Vector Machines are gaining more and more acceptance thanks to their success in many real–world problems. We address in this work some issues related to their hardware implementations for identification and control of a thermal model of an extruder for injection molding process.

1 Introduction

A regression problem is the process through which an unknown function $\phi : \Re^d \rightarrow \Re$, is estimated on the basis of some its samples[1] $(\boldsymbol{x}_i, t_i)_{i=1\ldots n}$, $\boldsymbol{x}_i \in \Re^d$ and $t_i \in \Re$. Usually, one faces with this problem when is able to observe and measure the input/output signals of the system under exam, but does not know its dynamic (i.e., the structure of $\phi(\cdot)$). Typical applications are: time series forecasting, identification and control of non–linear systems, signals and images processing, etc. The one of interest in this work is the identification and control of a thermal model representing an extruder of an Injection Molding Process (IMP) [8]. Briefly, the process consists in a screw, controlled by hydraulic or electric actuators, which runs in a barrel, and by a mold, from which the final plastic part is ejected; the barrel is kept at a uniform high temperature with a simple temperature control; the material is inserted at a solid state (*feed*), runs in the heated barrel and melts; then it is injected at a very high pressure in the mold, where cools down and takes the form of the final part. The cycle time of such a process can be in the order of seconds, but it depends on the size of the machines. It is well known that the value of the temperature of the polymeric material at the nozzle is one of the most critical parameter for final quality of the part; furthermore, it is very difficult to obtain the function $\phi(\cdot)$, where T_p, the temperature of the barrel, is its input, or the control variable, and T, the temperature of the material at the nozzle, its output.

[*] Fabio Bardi is with University of Genoa, DIBE, Via all'Opera Pia 11a, 16145 Genova, Italy.
[1] In the following text we will indicate vectors and matrices with respectively, lowercase and uppercase bold letters.

Support Vector Machines (SVMs) [6], are a new paradigm that have been recently proposed for solving pattern recognition and function approximation tasks. One of the most appealing properties of SVM is certainly the need of solving a quadratic programming problem subject to linear constraints. Briefly, the main goal of ϵ-SV regression is to find a function $\phi^*(\cdot)$ having at most ϵ deviation from the targets t_i for all the points and, at the same time, as flat as possible (for more details on SVM for regression see [6]). The function $\phi^*(\cdot)$ is given by $\phi^*(\boldsymbol{x}) = \sum_{i=1}^{n} (\gamma_i - \gamma_i^*) K(\boldsymbol{x_i}, \boldsymbol{x}) + b$. Where the free parameters are found by solving:

$$\min E(\boldsymbol{\alpha}) \quad = \quad \frac{1}{2}\boldsymbol{\alpha}^t \boldsymbol{Q} \boldsymbol{\alpha} + \boldsymbol{r}^t \boldsymbol{\alpha} \tag{1}$$

subjected to the constraints $0 \leq \boldsymbol{\alpha} \leq C$ and $\boldsymbol{\alpha}^t \boldsymbol{y} = 0$. C measures the tradeoff between the deviation of each target from the function and the flatness of the function itself;

$$\boldsymbol{Q} = \begin{bmatrix} \boldsymbol{D} & -\boldsymbol{D} \\ -\boldsymbol{D} & \boldsymbol{D} \end{bmatrix} \quad \boldsymbol{\alpha} = \begin{bmatrix} \boldsymbol{\gamma} \\ \boldsymbol{\gamma}^* \end{bmatrix} \quad \boldsymbol{r} = \begin{bmatrix} \boldsymbol{\epsilon} - \boldsymbol{t} \\ \boldsymbol{\epsilon} + \boldsymbol{t} \end{bmatrix} \quad \boldsymbol{y} = \begin{bmatrix} \boldsymbol{1} \\ -\boldsymbol{1} \end{bmatrix} \tag{2}$$

$d_{ij} = d_{ji} = K(\boldsymbol{x}_i, \boldsymbol{x}_j)$, $\epsilon_i = \epsilon, \forall i$, $\boldsymbol{\alpha}, \boldsymbol{r}, \boldsymbol{y} \in \Re^{2n}$, $\boldsymbol{\gamma}, \boldsymbol{\gamma}^* \in \Re^n$; $K(\cdot,\cdot)$ is a *kernel function* (i.e. a Gaussian function with variance σ^2) [6].

We show here some preliminary results obtained by authors, concerning the hardware realization of intelligent systems for the control of an injection molding process; thus, the purpose of this paper is twofold: 1) to collect the work published so far and to sketch some further directions of research on the hardware implementation of SVM (section 2); 2) to develop physical and thermodynamics models in order to verify the capacity of learning of SVM when applied to such kind of systems (section 3). In the end, we give some conclusion in section 4.

2 Analog and digital models for SVM

The hardware implementation of artificial neural networks has always been a very interesting research field for the electronics and computer science communities. The solutions proposed in the literature are too numerous to be even mentioned here. After a seminal work on theoretical aspects of SVM, some attempts at implementing them on analog or digital hardware have started to emerge [1, 2, 3, 4]. The hardware implementation can be addressed by defining a dynamical system whose stable point coincides with the solution of (1). If the system is described by a continuous–time differential equation, it can be implemented in analog hardware using op–amps as integrators and some linear or nonlinear devices. If a discrete–time approach is followed, the resulting system can be easily implemented in digital hardware, through adders, multipliers and comparators and using the time step as the system clock. In the following we briefly describe both solutions. In a previous work [1], authors proposed the Chua's recurrent network [5] to solve the problem of SVMs learning. Briefly, such a network is based on the use of a penalty function that forces the solution to fulfill the constraints of the problem. the circuit showed in [1] can be used to solve our problem. The penalty function is simply :$g(v) = 0$ if $v > 0$, Gv otherwise. The constraints are written in a matrix form ($f(\boldsymbol{\alpha}) = B\boldsymbol{\alpha} - \boldsymbol{e} \geq 0$), where $e_i = \{0, C\}$, $\boldsymbol{B}^T = \begin{bmatrix} -\boldsymbol{y} & \boldsymbol{y} & -\boldsymbol{I} & \boldsymbol{I} \end{bmatrix}$, and dim($\boldsymbol{B}$) $= 2n \times n$. G is the penalty term to be setted. Then, as the gain G of the limiters goes to ∞, the constraints are satisfied as desired, otherwise the solution found is only an approximation of the correct one.

A more correct methodology for handling the equality constraint, which exists only if the bias term is considered, is to build models based on the framework of the Lagrange Multipliers theory for solving constrained optimization problems. Xia et. al. have extensively developed this research area, and have proposed different models for solving such kind of problems. Their main works are [9, 10]. Basically, the structure of these models, is based on the joined work of two kinds of variables (also called the *neurons* of the Neural Network model): the *functional variables* $\boldsymbol{\alpha}$, that is the variables of our problem, lead to the minimum of the cost function, while the *Lagrangian variable* λ has the role of forcing the solution to stay in the feasible region. In practice it acts as a force on the dynamic system describing the evolution of $\boldsymbol{\alpha}$ towards the optimum, in order to impose a path to it in the space of the solutions, in such a way that it can fulfill the Karush Kuhn Tucker conditions (KKTs) at optimality. The evolution of the recurrent network is defined by the following system:

$$\begin{cases} \boldsymbol{C}_\alpha \frac{d\boldsymbol{\alpha}}{dt} = (\boldsymbol{I} + \boldsymbol{Q})(\boldsymbol{v} - \boldsymbol{\alpha}) - \boldsymbol{m}^* \\ C_\lambda \frac{d\lambda}{dt} = -\boldsymbol{y}^T \boldsymbol{v} \end{cases} \tag{3}$$

where $\boldsymbol{m}^* = m\boldsymbol{y}$, $m = \boldsymbol{y}^T\boldsymbol{\alpha}$, $\boldsymbol{v} = P_\Omega(\boldsymbol{\alpha} - \boldsymbol{Q}\boldsymbol{\alpha} - \boldsymbol{e} + \boldsymbol{y}\lambda)$, and $\boldsymbol{C}_\alpha$, C_λ, are the capacitors of the circuit. $P_\Omega : \Re^n \to \Omega$ is a projection operator, with $\Omega = \{\boldsymbol{v} \in \Re^n | 0 \leq v_i \leq C, \quad \forall i\}$. Note that, differently from the model given previously, the variables can assume any value during time. That is, it is not important whether the constraints are fulfilled or not; we are only sure that at the equilibrium the solution satisfies the KKTs. The above circuit has the structure analogous to the one showed in [10]. A block diagram of the general architecture of this network, suitable for SVM learning, is shown in [4]; in such a paper the reader can find an extensive description of the model; furthermore, we have shown, with simple straightforward considerations from the KKTs, that the bias term can be easily computed ($b = -\lambda$); the global simplification that results from this observation is remarkable with respect to the one showed in [10].

We have extensively described in [3] some algorithms and architectures for the digital implementation of SVM. In the following we briefly review some of the most important models. The digital implementation can completely avoid the penalty function and realize the $2n$ inequalities with hard–limiting comparators[2]. The following model describes the updating rule for $\boldsymbol{\alpha}$ values:

$$\boldsymbol{z}^k = \boldsymbol{\alpha}^k - \eta\nabla\boldsymbol{E} \tag{4}$$

$$\alpha_i^{k+1} = P_\Omega\left(z_i^k\right) \tag{5}$$

[2]This simplifies the learning problem by avoiding the equality constraint in the minimization. The omission can be very helpful due to unavoidable numerical errors in hardware implementations that prevent to fully satisfy the equality constraint. On the other hand, SVMs without bias can be used only when the dimensionality of the feature space is large enough to avoid suffering from the lack of it.

ESANN'2001 proceedings - European Symposium on Artificial Neural Networks
Bruges (Belgium), 25-27 April 2001, D-Facto public., ISBN 2-930307-01-3, pp. 75-80

This updating scheme is particularly attractive for a VLSI implementation thanks to its simplicity, and because an optimal descent step is given (see [2, 3] for more details). We can exploit the same model, by adding a penalty term of the form $G||\boldsymbol{y} \cdot \boldsymbol{\alpha}||^2$ to the quadratic function, which allows to satisfy the equality constraint (though, really, only when $G \to \infty$). The recurrent relation is then modified by substituting $\boldsymbol{Q}$ with $\boldsymbol{Q}_G = \boldsymbol{Q} + G\boldsymbol{Y} = \boldsymbol{Q} + G\boldsymbol{y}\boldsymbol{y}^T$.

3 Identification and control of the IMP

An unknown non–linear dynamic system can be controlled by using the well known *identify + control* scheme [7] (see figure 1). The identification is realized by an intelligent system (i.e. a Support Vector Machine), that is able *to learn* and *to emulate,* the system to be controlled through some its input/output (i/o) samples. The controller, designed on the basis of the parameters of the identification block (IB), permits the tracking of the desired reference r. The discrete–time i/o relationship of the system is represented by: $t_{k+1} = \phi\left[t\left(k\right), \cdots, t\left(k-n+1\right), u\left(k\right), \cdots, u\left(k-m+1\right)\right] = \phi\left(\boldsymbol{x}_k\right)$, $\phi\left(\cdot\right)$ is unknown, $n + m = d$, and the sample time $\Delta\tau$ is given. If $\phi\left(\cdot\right)$ represents the injection molding process previously described, t_k is the temperature of the material at the nozzle, and u_k the temperature of the heater; $\phi\left(\cdot\right)$ is the unknown function to be estimated by the learning system, in our case an SVM. The output of the SVM will be of the form $t^*_{k+1} = \phi^*\left(\boldsymbol{x}_k\right)$. In order to verify the capacity of the SVM of learning the dynamic of the injection molding process, and to build and test the structure of the whole control system, we have built a simplified thermal model of the extruder, which is able to simulate the relationships among the different magnitudes of the system; furthermore, following such approach, we have an automatic source of samples $(\boldsymbol{x}_i, t_i)$ to design the SVM. Such a model will be extensively described in a paper that we are writing, here we focus on the design of the controller, as the IB can be designed following the specifications provided in the previous section. Let us consider the control scheme of figure 1. The controller provides the temperature of the heater u, in such a way to minimize e, on the basis of the structure of the IB (the analytic expression of $\phi\left(\cdot\right)$ is supposed to be unknown). The idea is quite simple, and follows this principle [7]: let us define the cost $J = \frac{1}{2}e^2\left(k+1\right) = \frac{1}{2}\left[r\left(k+1\right) - t^*\left(k+1\right)\right]^2$. The goal is to minimize J; this leads to the definition of the following control law: $u\left(k+1\right) = u\left(k\right) - \eta_c \frac{\partial J}{\partial u(k)}$, where η_c is a descent step to be set. It can be easily shown, that if a Gaussian kernel is supposed to be used, the control block has the following structure:

$$u\left(k+1\right) = u\left(k\right) + \frac{\eta_c}{\sigma^2} e\left(k+1\right) \sum_i \left(\gamma_i - \gamma_i^*\right)\left(\boldsymbol{x}_i - \boldsymbol{x}\right)_{n+1} e^{-\frac{\|\boldsymbol{x}_i - \boldsymbol{x}\|^2}{2\sigma^2}} \qquad (6)$$

$\left(\boldsymbol{x}_i - \boldsymbol{x}\right)_{n+1}$ indicates the component $n+1$ of the vector $\left(\boldsymbol{x}_i - \boldsymbol{x}\right)$. Analogous laws can be found by using different kind of kernel functions. Figure 2 shows the result of an experiment; the reference r was set to 466.3 and 468 [K] Kelvin degree, respectively; we also used a sample time of 0.2 sec, a descent step of $\eta_c = 0.8$ and $\sigma^2 = 0.02$.

Figure 1: A typical identify–control scheme

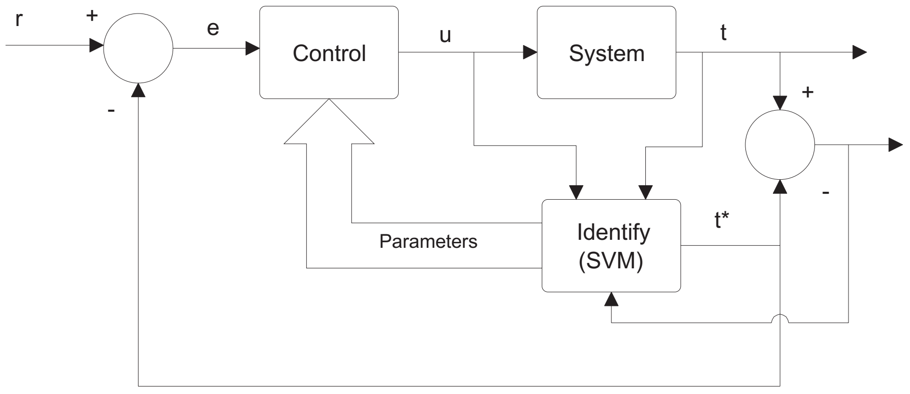
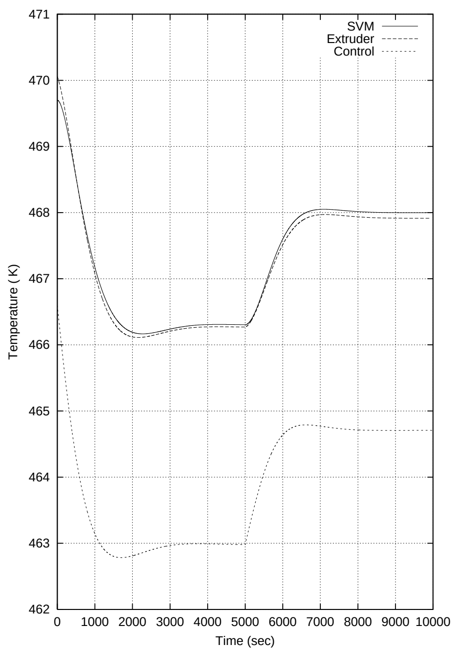


Figure 2: Control signal, system (model) and SVM outputs for $r = 466.3$ and $r = 468$ Kelvin degree.



4 Conclusion

This work collects some results obtained by authors about analog and digital models suitable for the hardware implementation of SVMs. Obviously, more work has to be done concerning the actual circuital implementation, in particular more work has to be addressed towards the search of optimal models for the digital implementation with bias. We have briefly described a real–world application that can benefit of such realizations: the temperature control of an injection model process; this is a typical application where a real time response is often necessary, so the intelligent hardware described in this paper could help to obtain the required performances.

References

[1] D. Anguita, S. Ridella, and S. Rovetta: Circuital implementation of support vector machines. Electr. Letters, 34, 1596–1597 (1998).

[2] D. Anguita, A. Boni, and S. Ridella: Learning algorithm for nonlinear support vector machine suited for digital VLSI. Electr. Letters, 35, 1349–1350 (1999)

[3] D. Anguita, A. Boni, and S. Ridella: VLSI friendly training algorithms and architectures for Support Vector Machines. Inter. Journal of Neural Systems, 10, 159-170 (2000).

[4] D. Anguita, A. Boni, and S. Ridella: A globally convergent circuit for support vector machines. IEE Proc. Circ., Dev. and Sys., in review.

[5] L. Chua and M.P. Kennedy: Neural networks for nonlinear programming. IEEE Trans. on Circ. and Sys., 35, 554–562 (1988).

[6] N. Cristianini and J. Shawe–Taylor: An introduction to support vector machines. Cambridge University Press (2000).

[7] J. R. Noriega and H. Wang: A direct adaptive neural-network control for unknown nonlinear system and its application. IEEE Trans. on Neural Networks, 9, (1998).

[8] D. V. Rosato: Injection molding handbook: the complete molding operation technology, performance, economics, 2nd ed., New York (1995).

[9] Y. Xia: A new neural network for solving linear and quadratic programming problems. IEEE Trans. on Neural Networks, 7, 1544–1547 (1996).

[10] Y. Xia, Y. Tan, and J. Wang: Neural network realization of support vector methods for pattern classification. Proc. of IJCNN'00, Como, Italy, July (2000).