A Sparse Robust Model for a Linz–Donawitz Steel Converter

József Valyon and Gábor Horváth, *Senior Member, IEEE*

***Abstract*—Steelmaking with a Linz–Donawitz converter is a complex industrial process, where, due to the lack of exact mathematical (physical–chemical) models, the construction of a black-box model based on noisy and imprecise data is required. To construct a good model, a large number of such input–output samples should be used, which calls for a method that is sparse, in the sense that the resulting model complexity is independent of the sample number, and robust to reduce the effects of noise. Lately, support vector machines (SVMs) have successfully been applied to a number of such problems. The main problem with traditional SVM is its high algorithmic complexity, which makes it infeasible for really large databases. The least-squares SVM (LS-SVM) solves this problem, but the resulting model is not sparse. Our solution uses a sparse and robust extension of LS-SVM that leads to good results compared to other methods (such as MLPs) applied to the same problem.**

***Index Terms*—Basic oxygen furnace (BOF), basic oxygen steelmaking (BOS), black-box modeling, least-squares support vector machine (LS-SVM), Linz–Donawitz (LD) converter, $LS^2 - SVM$, steel industry.**

I. Introduction

IN THE INDUSTRY, many complex modeling and control problems can be found, where exact or even approximate theoretical/mathematical relationships between inputs and outputs cannot be formulated. In these cases, if a sufficiently large number of input–output data are available, then an experimental black-box model can be constructed. Recently, kernel methods such as support vector machine (SVM) have successfully been applied to a wide variety of real-life black-box modeling problems. The most common issue when applying these methods is to achieve a sparse, thus small, model while being robust against noise burdening the sample data. The aim of this paper is to apply a sparse and robust kernel-based method to the modeling of a *Linz–Donawitz* (LD) steel converter. The results presented here are from a follow-up study of a larger work concerning the same problem, where a hybrid-neural model was built [1]–[3]; therefore, many experiences and results are reused here. As the main part of LD steelmaking is oxygen blowing, this process is often called *basic oxygen steelmaking* (BOS) or *basic oxygen furnace* (BOF) steelmaking.

Manuscript received July 5, 2007; revised August 27, 2008. First published May 2, 2009; current version published July 17, 2009. This work was partly sponsored by the Hungarian Fund for Scientific Research (OTKA) under contract T046771. The Associate Editor coordinating the review process for this paper was Dr. Richard Thorn.

The authors are with the Department of Measurement and Information Systems, Budapest University of Technology and Economics, Budapest 1117, Hungary (e-mail: horvath@mit.bme.hu; valyon@mit.bme.hu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TIM.2009.2015638

The main steps of the process are the following.

1) A large (∼150 tons) converter is filled with waste iron (∼30 tons), molten pig iron (∼110 tons), and many additives.
2) The converter is blasted through with pure oxygen to burn out the unwanted contamination (e.g., carbon, silicon, etc.). The blasting phase takes about 20–25 min.
3) At the end of the blasting, the quality of the steel is tested, its temperature is measured, and the resulting steel is tapped off for further processing.

A converter is used in many (about 2000–3000) steelmaking iterations—which is called *loads* or charges. Due to some contamination left in the converter, but more importantly due to the starting temperature of the converter, the consecutive loads are not independent of each other. During each load, there are about 30–50 input and two output parameters recorded. The input parameters originate from the following values:

1) the values of the temperature, the mass values, and the quality features of the components (pig iron, waste iron, etc.);
2) the mass values of the different additives.

The two essentially important output parameters are the following:

1) the carbon content of the steel;
2) its temperature at the end of the blasting process.

The output parameters—which determine the quality of the resulting steel—mainly depend on the amount of pure oxygen used during blasting, so this is the control variable that should properly be determined.

According to the knowledge and experience of human personnel, it turns out that it is much easier to achieve the desired carbon content than to reach the desired final temperature that must fall into a rather narrow range (typically $1670 \; -10, \; +15\,^{\circ}\mathrm{C}$). Thus, the carbon content is not taken into consideration, and the developed model only focuses on the output temperature, which mainly depends on the amount of oxygen used. Thus, an oxygen model is built, which can be regarded as the inverse model of the process [2] (see Fig. 1).

According to the results and experiments of the LD converter project, this modeling problem can be simplified. Although the oxygen is an input, and the temperature T is the output of the real system $T = f(\cdot, O_2)$, the inverse model can be constructed from the same data set by using the temperature as the input and the oxygen as the output, i.e.,

$$O_2 = f^{-1}(\cdot, T). \tag{1}$$

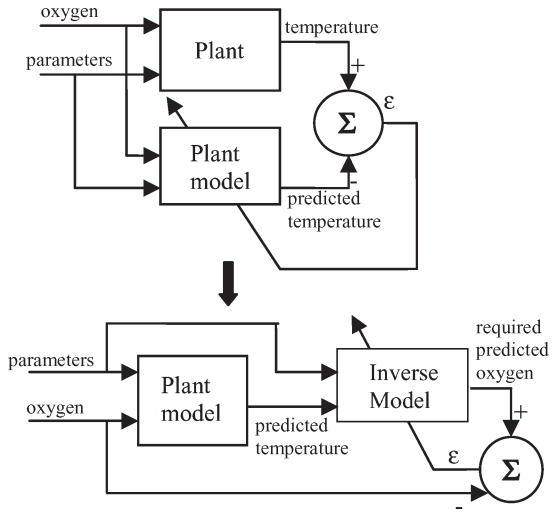


Fig. 1. Temperature (forward) and oxygen (inverse) model.

Converter modeling is a really hard task, where many different approaches have been applied so far. Classical mathematical models based on physical and chemical laws have been used in [4]–[6]. A mathematical approach, however, has some inherent limitations. The process is too complex to accurately be described by a set of equations, and these models usually require some very expensive measurements about the process. Another option is to utilize neural networks, which have been used for both control [7]–[10] and modeling purposes.

In [11], the dynamics of every single blasting is modeled; however, here, the applied technology makes it possible to sample the temperature and the main composition parameters of the liquid metal during blasting (using a sublance). Widlund and Lahiri [12] describe a neural prediction system for the heat losses and postcombustion, which can help to understand the complex process of LD steelmaking.

Due to the harsh industrial environment, only an unreliable database [1] is available, and, based on these data, the experimental black-box model has to be built. The main problem is that the data are imprecise and noisy, where the possible existence of outliers must be emphasized. To overcome these problems, extensive data analysis and preprocessing were required, where both mathematical methods and domain knowledge have been used. The consecutive loads are not independent of each other, so this dependency should be taken into consideration. This means that, although the dynamics of a single blasting is not modeled, the series of blastings are considered as consecutive events of a dynamic process, and, in this respect, a dynamic model should be constructed. Concerning the dynamics of the problem, tests previously made during neural modeling show that the best results can be obtained using second-order NARX (series–parallel) models [3] with 53 inputs, where the data are ordered as consecutive *loads* of the converter. More details of data analysis as well as of the neural modeling were presented in previous publications [1]–[3].

In this paper, kernel methods, such as SVMs [13], have been considered, because they incorporate some useful properties that make them favorable in solving such problems. The primary advantage of SVMs is that, by basing its solution on a subset of training samples (the support vectors), they automatically derive a sparse network structure, which concludes from some "optimality" criterion. Another possibility is the use of least-squares SVM (LS-SVM) [15], which has similar advantages as a traditional SVM, but it has some user-friendly properties regarding the implementation and computational issues of teaching. The training of an LS-SVM requires to solve a set of linear equations instead of the quadratic programming (QP) involved by SVM.

The complexity of a standard SVM model—because of the sparseness of this solution—is determined by the number of support vectors, which is usually much smaller than the number of all training samples. On the other hand, the model resulting from the LS-SVM method consists of exactly the same number of support vectors as many training samples were used; thus, the result is not sparse. By applying a *pruning* method, which selects a subset of the samples, sparseness can also be reached by LS-SVM, but in this case performance declines [15]. To reduce the effects of noise and, thus, achieve a robust solution, a *weighted* LS-SVM [15] should be used. Unfortunately, the LS-SVM pruning and weighting methods work in opposition; thus, they cannot be applied at the same time.

To overcome these problems, an extension of the LS-SVM method is proposed, which results in a robust and sparse model. This new model is successfully applied to the steelmaking problem.

In the following section, the basic LS-SVM and some of its extensions *as general modeling methods* are outlined. A detailed description can be found in [7] and [8]. In Section III, the sparse and robust modification of the LS-SVM method is introduced. Section IV summarizes the experiments presenting both illustrative benchmark problems and the results achieved for LD converter modeling. In Section V, some conclusions are drawn.

II. LS-SVM Modeling

Given the $\{\mathbf{x}_i, d_i\}_{i=1}^N$ training data set, where $\mathbf{x}_i \in \Re^p$ represents a p-dimensional input vector, and $d_i = y_i + z_i$, $d_i \in \Re$ is a scalar-measured output, which represents the y_i system output disturbed by some z_i noise. Our goal is to construct a $y = f(\mathbf{x})$ function, which represents the dependence of the output y on the input $\mathbf{x}$.

One of the most important aspects of kernel methods is that they apply nonlinear mapping to linearize the problem. This mapping is done by using some $K(\cdot, \mathbf{c}_i)$ kernel functions. For a given input $\mathbf{x}$, the response of an LS-SVM is a weighted sum of N kernel functions, where the center parameters $(\mathbf{c}_i)$ of these kernel functions are determined by the training input vectors $\mathbf{x}_i$ as

$$y(\mathbf{x}) = \sum_{i=1}^{N} \alpha_i K(\mathbf{x}, \mathbf{x}_i) + b. \tag{2}$$

The LS-SVM solution (thus, the α_i weights and the bias b) is calculated from the following linear equation set [15]:

$$\left[\begin{array}{c|c} 0 & \vec{\mathbf{1}}^{\mathbf{T}} \\ \hline \vec{\mathbf{1}} & \boldsymbol{\Omega} + C^{-1}\mathbf{I} \end{array}\right] \left[\begin{array}{c} b \\ \hline \boldsymbol{\alpha} \end{array}\right] = \left[\begin{array}{c} 0 \\ \hline \mathbf{d} \end{array}\right] \tag{3}$$

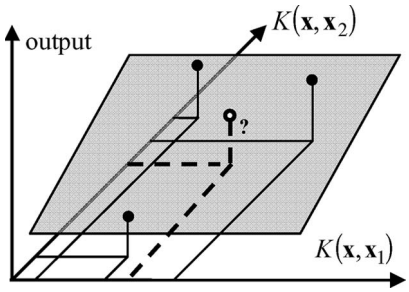


Fig. 2. Kernel space based on two SVs ($N = 2$) and the output. The black dots represent the training samples, whereas the white dot illustrates how the output is determined in the recall phase.

where C is a regularization coefficient, $\mathbf{d} = [d_1, d_2, \ldots, d_N]^T$ is the vector formed from the desired outputs, $\boldsymbol{\alpha} = [\alpha_1, \alpha_2, \ldots, \alpha_N]^T$ is the weighting coefficient vector in the kernel representation, $\vec{\mathbf{1}} = [1, \ldots, 1]^T$, and $\Omega_{i,j} = K(\mathbf{x}_i, \mathbf{x}_j)$; thus, $\boldsymbol{\Omega}$ represents the kernel matrix. Throughout this paper, a Gaussian kernel function is used, but other kernels can also be applied.

When an LS-SVM is constructed from N training samples, we have the following.

1) The training samples are mapped to an $(N+1)$-dimensional space, where N dimensions are defined by the kernel functions, and the additional one is defined by the desired output.
2) In the $(N+1)$-dimensional space, an N-dimensional hyperplane is fitted on the mapped samples. The free parameters of the hyperplane (the α_i's and b) are determined by the N-mapped training points and one additional constraint [namely, the first line of (3)]. For the sake of generalization and to avoid overfitting, the accuracy of the fit is adjusted through a regularization term $C^{-1}\mathbf{I}$.

For a new input sample $\mathbf{x}$, the response of the networks is determined by (2), which corresponds to the point of the hyperplane at its mapped coordinates. One expects that it will be close to the desired output. This kernel space view of the process is illustrated in Fig. 2.

One of the main drawbacks of the least-squares solution is that it is not sparse, because unlike the original SVM, it incorporates all training vectors in the result.

LS-SVM pruning [15]: To get a sparse solution, a pruning method must be used. The method works by iteratively leaving out the least significant data vectors. As the absolute values of the α_i coefficients $|\alpha_i|$ are proportional to the errors at data points ($\alpha_i = Ce_i$), the omitted data are the ones that have the smallest e_i errors and, thus, the smallest corresponding $|\alpha_i|$ values.

Weighted LS-SVM [15]: This method addresses the problem of noisy data-like outliers in a data set by using a weighting factor in the calculation based on the error values determined from a previous—first an unweighted—solution. The method uses a bottom-up approach by starting from a standard solution and calculating one or more weighted LS-SVM based on the previous result. The weighting is designed such that the results improve in view of robust statistics. Large e_i's mean small weights and *vice versa*.

A common property of the described methods is that they are all iterative, where every step is based on the result of an LS-SVM learning. This means that the entire large problem must be solved at least once, and a relatively large one in every further iteration step. Another drawback is that pruning and weighting cannot easily be combined, because the methods favor contradictory types of points. While pruning drops the training points belonging to small α_i's, the weighted LS-SVM increases the effects of these points.

III. Sparse Robust LS-SVM

The sparse and robust LS-SVM proposed here is achieved as follows.

1) A sparse solution is reached by using partial reduction of (3).
2) A selection method is described to determine the proper support vectors (kernel centers).
3) To reduce the effects of noise, robust methods are used to find a solution that fits the bulk of the data.

A. Sparseness by Partial Reduction ($LS^2 - SVM$)

If the training set consists of N samples, then our original linear equation set [see (3)] will have $(N+1)$ unknowns, the α_i's, $(N+1)$ equations, and $(N+1)^2$ multipliers. These factors are mainly the values of the $K(\mathbf{x}_i, \mathbf{x}_j)$ kernel function calculated for every combination of the training inputs. The cardinality of the training set, therefore, determines the size of the kernel matrix, which plays a major part in the solution, as the algorithmic complexity, the complexity of the result, etc., depend on this.

The most important component of the main matrix is the $\boldsymbol{\Omega}$ kernel matrix; its elements are the values of the kernel function for pairs of training inputs $\Omega_{i,j} = K(\mathbf{x}_i, \mathbf{x}_j)$.

To reduce the equation set, columns and/or rows may be omitted. Each column of the kernel matrix represents an additive term in the final solution with a kernel function centered on the corresponding $\mathbf{x}_i$ input weighted by α_i. It is easy to see that the network size is determined by the number of columns, which—to reach sparseness—must be reduced. The rows in (3), however, represent the input–output relations, which are described by the training points $(\mathbf{x}_i, d_i)$ and, thus, are the constraints that should be satisfied.

If the training sample $(\mathbf{x}_i, d_i)$ is discarded from the training set, then both the column and the row corresponding to this sample are eliminated. In this case, however, reduction also means that the knowledge represented by the sample is also lost. This is exactly what the traditional LS-SVM pruning does [15], since it iteratively omits some training points. The information embodied in these points is entirely lost.

To avoid this information loss, we have proposed a technique called partial reduction. In this case, the omission of a training sample $(\mathbf{x}_i, d_i)$ means that only the corresponding column is eliminated, whereas the row (which defines an input–output relation) is kept. Eliminating the ith column reduces the

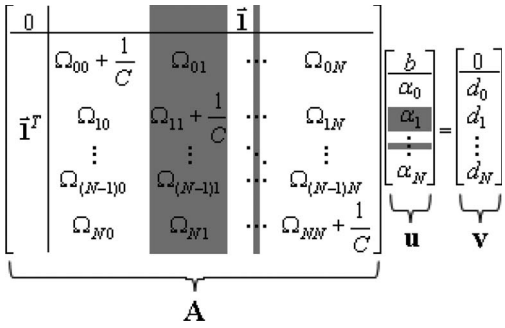


Fig. 3. Effect of partial reduction to the LS-SVM equation set (the gray elements are removed).

model complexity, whereas keeping the ith row means that the weighted sum of that row should still meet the d_k regression goal (as closely as possible).

The omission of columns while keeping the rows means that the network size is reduced; still, all the known constraints are taken into consideration. This is the key concept of keeping the quality while the equation set is simplified.

By selecting some (e.g., M, $M < N$) vectors as "*support vectors*," partial reduction concludes to an overdetermined equation set of N equations and M unknowns. This can be solved as a linear least-squares problem, where the summed square of the residuals

$$S = \sum_{i=1}^{N} e_i^2 = \sum_{i=1}^{N} (d_i - y_i)^2 \tag{4}$$

is minimized. The solution of the reduced equation set is formulated as

$$\mathbf{A}^T\mathbf{A}\mathbf{u} = \mathbf{A}^T\mathbf{v} \tag{5}$$

where $\mathbf{A}$, $\mathbf{u}$, and $\mathbf{v}$ are the reduced matrices and vectors of Fig. 3.

The modified matrix $\mathbf{A}$ has $(N+1)$ rows and $(M+1)$ columns. After the matrix multiplications, the results are obtained from a reduced equation set incorporating $\mathbf{A}^T\mathbf{A}$, which is only of size $(M+1)\times(M+1)$. We call this method LS$^2-$SVM, since it gives the least-squares solution of the LS-SVM method.

The support vectors of the partially reduced model can be selected many ways, e.g., randomly, using the traditional pruning method (based on the sorted $|\alpha_i|$ spectrum), or by using the method proposed in the sequel.

B. Selecting Support Vectors

To achieve sparseness by partial reduction, the linear equation set has to be reduced by selecting some support vectors. In the method proposed in [17], this is done by using the reduced row echelon form (RREF) of $\mathbf{A}$. This method is a slight modification of the Gaussian Jordan elimination with partial pivoting. RREF applies a tradeoff parameter ε' that determines the balance of the number of support vectors and the accuracy of the solution.

C. Robust LS$^2-$SVM

It is easy to see that partial reduction leads to a sparse solution, but having an overdetermined equation set has several other advantages. By having more equations than unknowns, we have means to statistically analyze this information set. The solution is better if it fits to the core of the data and is not corrupted by individual errors, i.e., outliers. Since the d output is not transformed in any way, the error in the kernel space (the distance to the hyperplane) exactly matches the primal space error. This means that an outlier in the problem space is also an outlier in the reduced kernel space. By having an overdetermined problem in the kernel space, we can reduce the effects of outliers by simply using robust linear fitting methods. The solution of this equation set corresponds to a linear fitting problem, where we have to fit an $(M+1)$-dimensional hyperplane on the points defined by the N rows of the matrix. Since $N \gg M+1$, this can be done in several ways.

The least-squares solution of the overdetermined system described above [see (5)] is optimal in case of Gaussian noise; but otherwise, a weighted solution is needed. In this case, the

$$S = \sum_{i=1}^{N} w_i e_i^2 = \sum_{i=1}^{N} w_i (d_i - y_i)^2 \tag{6}$$

error term is used, which leads to the solution

$$\mathbf{A}^T\mathbf{W}\mathbf{A}\mathbf{u} = \mathbf{A}^T\mathbf{W}\mathbf{v} \tag{7}$$

where $\mathbf{W}$ is a diagonal matrix containing the w_i weights. The weights can be determined based on some prior information (e.g., known noise distributions). If no prior information is available, then the weights can also be calculated from the statistical properties of the points in the kernel space. This can be done by utilizing the following robust methods [18]–[20].

1) Least Absolute Residuals (LAR): This method minimizes the absolute value of residuals. This means that extreme values have less influence on the fit.
2) Bisquare Weights: A method that minimizes a weighted sum of squares, where the weight of each data point depends on its distance from the fitted line. The farther away is the point, the less weight it gets. This method fits the hyperplane to the bulk of the data with the least-squares approach, while it minimizes the effect of outliers.
3) Least Trimmed Squares (LTS): This method defines a trimming constant h $((N/2) < h \leq N)$ and only takes the $N-h$ smallest residuals into consideration [19].

From the aforementioned methods, we will use the bisquare weights method in the robust LS-SVM, because this method was found to be the most useful (considering implementation, computational, and of course performance issues) during our experiments.

IV. Experiments

This section shows that the proposed version of the LS-SVM can efficiently be used for solving practical problems. The experiments are divided into two sections: First, simple

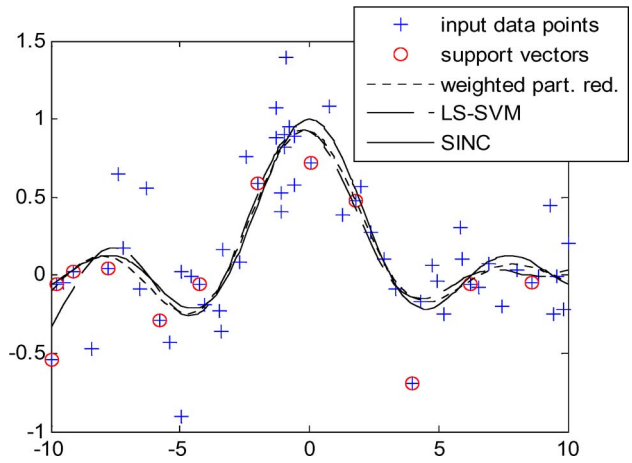


Fig. 4. Custom weighting is applied with partial reduction. (The LS–SVM is not weighted. $\mathrm{MSE}_{\mathrm{weightedpart.red.}} = 2{*}10^{-3}$, $\mathrm{MSE}_{\mathrm{LS-SVM}} = 6{*}10^{-3}$.)

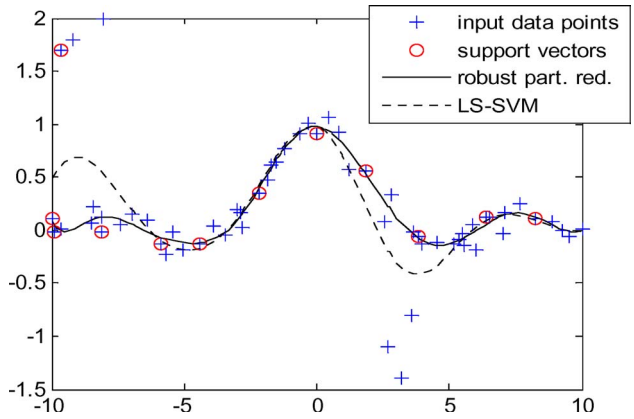


Fig. 5. Continuous black line plots the result for a partially reduced LS-SVM solved by the bisquare weight method. The dashed line is the original LS-SVM ($\mathrm{MSE}_{\mathrm{robustpart.red.}} = 2.3 * 10^{-3}, \mathrm{MSE}_{\mathrm{LS-SVM}} = 4.6 * 10^{-3}$).

demonstrative problems are presented to illustrate the results. Then, our spare and robust LS-SVM is applied for the LD converter modeling task.

A. Illustrative Experiments

The figures show the results for a simple illustrative experiment, i.e., the $sinc(x)$ regression. The training set contains 60 data samples corrupted with Gaussian noise.

Fig. 4 shows the results of custom weighting. We have 60 samples with additive Gaussian noise, where the σ of the noise is known for all samples. It can be seen that the effect of noise is reduced. The original LS-SVM is plotted, because the weighted LS-SVM would give almost the same results as the partially reduced solution, but in this case, we have a sparse solution.

The following experiment (Fig. 5) shows the $sinc(x)$ problem, where few data points are outliers.

It can be seen that by using a robust bisquare fitting, the effect of the outliers was successfully reduced.

It is important to mention that the result of the partially reduced LS-SVM is sparse, i.e., only consisting of 12 support vectors. If the number of training samples is very high for the problem complexity, then the gain in the network size can be rather large.

TABLE I
DATA SETS USED TO TEST THE MODELS

	Datasets	# of samples	#training	#test
#1	**Full dataset**	4597	3000	1597
#2	**Filtered dataset**	2821	2102	719

B. LD Steel Converter Modeling

For the experiments presented here, one of the most representative data set of the original LD converter project was used. From this data set, a filtered data set is created, where filtering is based on a large amount of complex knowledge concerning the steelmaking process. In this case, special loads (e.g., loads aiming to create special quality steel) and many other samples found somehow suspicious were removed. The filtering process and, thus, the data sets are the results of the former project. Each data set was randomly split to a training set and a test set (Table I).

Each datum contains 53 input components (the temperature and the mass values of the components, e.g., pig iron, waste iron, and the mass values of the different additives) and one output (the oxygen). Therefore, the inputs of the LS-SVM model are formed from the input parameters of the LD converter. A second-order NARX model is constructed; thus, the input vector $\mathbf{x}$ of the network is formed from the data of the current and previous loads.

The original project created a neural model, and the best results were achieved for these data sets (the results will later be shown).

To construct a good LS-SVM model, first, the optimal (or near optimal) hyperparameters C in (3) and the width parameter of the Gaussian kernel σ must be determined, which was done by using cross validation. A more detailed description of this task can be found in [21].

In case of the partially reduced LS-SVM, the optimal hyperparameter setting depends on the degree of sparseness, i.e., the number of support vectors used. To simplify the experiments, we will use the same C and σ, which were optimized for a predefined model size (tolerance), for all models, knowing that the result may be improved through optimizing these hyperparameters. A Gaussian kernel is used, where the best settings are $C = 5000$ and $\sigma = 2.75$.

After building the model from the measured data set, the results must be validated. The most important aspect of constructing a validation scheme is to simulate the real system and the real problem as close as possible. When the model is used in production, the steel temperature at the end of the process is unknown (only its desired value is given). This is an important difference, since, during training, the actual temperature value could be used, whereas in the validation (testing) phase, the desired value should be used. According to this, in the model construction, the actual measured output temperature is used (T_{first}), whereas in the model validation, the originally desired goal temperature (T_{goal}) is considered, which is unlikely to be the same, since the factory staff could

TABLE II
ERROR RATES FOR THE PROPOSED METHODS

Test	Method	SV num	Error %
#1	LS-SVM	3000	42.01
	NN (MLP)		41.78
	LS^2-SVM	98	45.64
	Robust LS-SVM	98	42.20
#2	LS-SVM	2102	25.45
	NN (MLP)		24.29
	LS^2-SVM	74	25.87
	Robust LS-SVM	74	23.78

TABLE III
TOLERANCE USED IN THE REDUCTION IS SMALL; THUS, THERE ARE MORE SUPPORT VECTORS KEPT FOR BETTER PERFORMANCE

Test	Method	ε'	SV num	Error %
#1	LS^2-SVM	0.005	169	44.52
	Robust LS-SVM	0.005	169	42.70
#2	LS^2-SVM	0.005	122	24.34
	Robust LS-SVM	0.005	122	23.64

not determine the optimal value of oxygen when the training data were collected. Using the prior knowledge provided by the steelmaking experts, a "small" difference between these temperatures can be accounted for in a linear manner. At this working point, we can state that an added 400-m^3 oxygen results in $+30^\circ$C in the temperature value (T_{first})

$$T_{\text{estimated}} = \frac{O_2^{\text{advised}} - O_2}{400} 30 + T_{\text{first}} \tag{8}$$

where O_2^{advised} is the result of the inverse (oxygen) model. Therefore, the output error is defined as

$$T_{\text{error}} = T_{\text{estimated}} - T_{\text{goal}}. \tag{9}$$

For good-quality steel, the temperature error must be in the range of $[-10, +15]^\circ$C.

Unfortunately, the results are very hard to compare due to the varying techniques and setups, and particularly due to the differences of data sets (e.g., used inputs, measurement precision, etc.) and experimental setups (e.g., interpretation of error, etc.). Based on the literature, it can be stated, in general, that any result under a 40% miss ratio is considered to be quite good! We must emphasize that the goal of this paper is not to provide the best possible model, but to demonstrate that LS^2 − SVM can successfully be applied to such a problem.

Table II shows the results achieved by different methods for the two data sets. The error rates presented represent the ratio of the test samples, where the temperature error is outside of the acceptable temperature range. The neural network results are the best results achieved by the original project using an MLP [3].

The traditional LS-SVM incorporates 3000 support vectors, whereas the sparse solutions use less than 100 support vectors.

To illustrate the tradeoff between sparseness and performance, the same problem is addressed using different tolerance settings. Tables III and IV show the results achieved using a small and a too large ε'.

To demonstrate the use of robust modeling, we must use a data set that contains non-Gaussian noise, or more likely

TABLE IV
TOLERANCE USED IS TOO LARGE, WHICH RESULTS IN VERY FEW SVs AND CONSEQUENTLY UNACCEPTABLY LARGE ERRORS

Test	Method	ε'	SV num	Error %
#1	LS^2-SVM	0.1	23	53.48
	Robust LS-SVM	0.1	23	51.9
#2	LS^2-SVM	0.1	18	44.23
	Robust LS-SVM	0.1	18	40.61

TABLE V
ERROR RATES FOR THE PROPOSED METHODS

Test	Method	SV num	Error %
#1	LS-SVM	3000	43.14
	LS^2-SVM	98	46.71
	Robust LS-SVM	98	42.20
#2	LS-SVM	2102	28.37
	LS^2-SVM	74	29.21
	Robust LS-SVM	74	23.5

outliers. To achieve this, a slightly corrupted database was used for the experiments. In each data set, ten samples have been changed to provide outliers. The results are shown in Table V, where $\varepsilon' = 0.01$ tradeoff parameter is used.

It can be seen that, due to the noise, the results are worse than those for the cleaned filtered data set; but in this case, the robust solution is much better than the other results.

V. CONCLUSION

This paper has presented how a sparse and robust version of the LS-SVM method could successfully be applied to a complex industrial problem: modeling of a steelmaking LD converter. The used approach—which is based on its sparse construction—significantly reduces the size of the model without degrading its performance. In addition, its robustness gives noise immunity, that is, it can handle such situations when there are outliers in the database. Both goals have successfully been met. The model size have significantly been reduced from 3000 kernels to only about 100, whereas in case of noisy data, the robust method provided more accuracy.

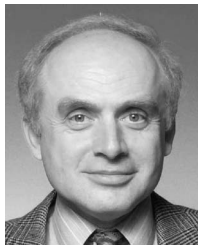
REFERENCES

[1] G. Strausz, G. Horváth, and B. Pataki, "Effects of database characteristics on the neural modeling of an industrial process," in *Proc. Int. ICSC/IFAC Symp. NC*, Vienna, Austria, Sep. 1998, pp. 834–840.
[2] G. Horváth, B. Pataki, and G. Strausz, "Black box modeling of a complex industrial process," in *Proc. IEEE Conf. Workshop Eng. Comput. Based Syst.*, Nashville, TN, 1999, pp. 60–66.
[3] P. Berényi, G. Horváth, B. Pataki, and G. Strausz, "Hybrid neural modeling of a complex industrial process," in *Proc. 18th IEEE Instrum. Meas. Technol. Conf.*, Budapest, Hungary, May 2001, vol. 3, pp. 1424–1429.
[4] Kawasaki Steel 21st Century Foundation, *An Introduction to Iron and Steel Processing Chapter 2. Smelting, Refining and Continuous Casting*. [Online]. Available: http://www.jfe-21st-cf.or.jp/index2.html
[5] R. Weeks, "Dynamic model of the BOS process," in *Proc. Conf. Math. Process Models Iron Steelmak.*, Amsterdam, The Netherlands, 1973, pp. 103–124.
[6] M. C. Abraham and A. Ghosh, "Kinetics of reduction of iron oxide by carbon," *Ironmak. Steelmak.*, vol. 1, pp. 14–23, 1979.
[7] W. E. Staib and R. B. Staib, "The intelligent arc furnace controller: A neural network electrode position optimization system for electric arc furnace," in *Proc. Int. Joint Conf. Neural Netw.*, Baltimore, MD, 1992, pp. 1–9.

[8] M. Schlang, T. Poppe, and O. Gramckow, "Neural networks for steel manufacturing," *IEEE Intell. Syst.*, vol. 11, no. 4, pp. 8–10, 1996
[9] T. J. Stich, J. K. Spoerre, and T. Velasco, "The application of artificial neural networks to monitoring and control of an induction hardening process," *J. Ind. Technol.*, vol. 16, no. 1, pp. 1–11, 2000.
[10] A. Datta, M. Hareesh, P. K. Kalra, B. Deo, and R. Boom, "Adaptive neural net (NN) models for desulphurization of hot metal steel," *Steel Res.*, vol. 65, no. 11, pp. 466–471, 1994.
[11] S. Y. Yun, K. S. Chang, and S. M. Byun, "Dynamic prediction using neural network for automation of BOF process in steel industry," *Iron Steelmak.*, vol. 23, no. 8, pp. 37–42, 1996.
[12] D. Widlund and A. K. Lahiri, "Artificial neural networks for prediction of heat losses and post-combustion in basic oxygen steelmaking," in *Proc. Steelmak. Conf.*, Chicago, IL, 1999, vol. 82, pp. 317–326.
[13] V. Vapnik, *The Nature of Statistical Learning Theory*. New York: Springer-Verlag, 1995.
[14] J. C. Platt, "Sequential minimal optimization: Fast algorithm for training support vector machines," Microsoft Res., Redmond, WA, Tech. Rep. MSR-TR-98-14, 1998.
[15] J. A. K. Suykens, V. T. Gestel, J. De Brabanter, B. De Moor, and J. Vandewalle, *Least Squares Support Vector Machines*. Singapore: World Scientific, 2002.
[16] J. Valyon and G. Horváth, "A generalized LS-SVM," in *Proc. SYSID*, Rotterdam, The Netherlands, 2003, pp. 827–832.
[17] J. Valyon and G. Horváth, "A sparse least squares support vector machine classifier," in *Proc. IJCNN*, 2004, pp. 543–548.
[18] P. Cizek, "Asymptotics of least trimmed squares regression," Tilburg Univ., Center Econ. Res., Tilburg, The Netherlands, 2004. Discussion Paper 72.
[19] P. W. Holland and R. E. Welsch, "Robust regression using iteratively reweighted least-squares," *Commun. Stat., Theory Methods*, vol. A6, pp. 813–827, 1977.
[20] P. J. Huber, *Robust Statistics*. Hoboken, NJ: Wiley, 1981.
[21] J. Valyon and G. Horváth, "A sparse robust model for a Linz–Donawitz steel converter," in *Proc. IEEE Instrum. Meas. Technol. Conf.*, Warsaw, Poland, May 1–3, 2007, pp. 1–6.

József Valyon received the Dipl. Eng. degree on informatics and the Ph.D. degree from the Budapest University of Technology and Economics, Budapest, Hungary, in 2000 and 2008, respectively.

Since 2000, he has been with the Department of Measurement and Information Systems, Budapest University of Technology and Economics, as a Ph.D. student and later as an Assistant Professor. His research and educational activity is related to the theory and application of neural networks and machine learning, particularly kernel methods, such as LS-SVM.



Gábor Horváth (M'95–SM'03) received Dipl. Eng. degree in electrical engineering from the Budapest University of Technology and Economics, Budapest, Hungary, in 1970 and the Ph.D. degree in digital signal processing from the Hungarian National Academy of Sciences, Budapest, in 1988.

Since 1970, he has been with the Department of Measurement and Information Systems, Budapest University of Technology and Economics, where he has held various teaching positions. He is currently an Associate Professor and the Head of the department. He has published more than 100 papers and is the coauthor of eight books. His research and educational activity is related to the theory and application of neural networks and machine learning.