Combining experimental data and physical simulation models in Support Vector learning

Gérard Bloch Fabien Lauer Guillaume Colin Yann Chamaillard

Centre de Recherche en Automatique de Nancy (CRAN UMR 7039), Nancy–University, CNRS, France,
gerard.bloch@esstin.uhp-nancy.fr
fabien.lauer@esstin.uhp-nancy.fr

Laboratoire de Mécanique et Energétique (LME, EA 1206), Université d'Orléans, France,
guillaume.colin@univ-orleans.fr
yann.chamaillard@univ-orleans.fr

Abstract

This paper considers modeling the in-cylinder residual gas fraction in Spark Ignition (SI) engine with Variable Camshaft Timing (VCT) based on a limited amount of experimental data and a simulator built from prior knowledge. The problem of how to best incorporate the data provided by the simulator, possibly biased, into the learning of the model is addressed. This problem, although particular, is very representative of numerous situations met in engine control, and more generally in engineering, where complex models, more or less accurate, exist and where the experimental data which can be used for calibration are difficult or expensive to obtain. The first proposed method applies a different loss function on the simulation data allowing for a certain level of inaccuracy. The second method constrains the derivatives of the model to be determined to fit to the derivatives of a prior model previously estimated on the simulation data. Finally, a third method considers the combination of these two forms of prior knowledge. These approaches are implemented in the linear programming support vector regression (LP-SVR) framework by the addition of constraints linear in the parameters to the optimization problem. Promising results are obtained on the application.

1 Introduction

The paper deals with the modeling of the in-cylinder residual gas fraction in Spark Ignition (SI) engine with Variable Camshaft Timing (VCT) for engine control. In this context, experimental measurements are complex and costly to obtain. On the other hand, a simulator built from physical knowledge can be available but cannot be embedded in a real time controller.

In engine control design (modeling, simulation, control synthesis, implementation and test), two types of models are commonly used:

- Low frequency models or Mean Value Engine Models (MVEM) with average parameters on the engine cycle. These models are often used in real time engine control [2, 6]. However, they must be calibrated on experiments in sufficiently large number in order to be representative.
- High frequency simulation models that can simulate the evolution of the variables during the engine cycle [7]. These models, of various complexity from zero-dimensional to three-dimensional models, are mostly based on fewer parameters with physical meaning. However, they cannot be embedded in real time controllers.

The idea is thus to build an embeddable

black box model by taking into account a prior simulation model, which is representative but possibly biased, in order to limit the number of required measurements. This problem amounts to a regression problem in which prior knowledge is available in the form of simulation data.

In non-linear function approximation, kernel methods, and more particularly Support Vector Regression (SVR) [25], have proved to be able to give excellent performances in various applications [17, 24, 16]. SVR originally consists in finding the function that has at most a deviation ε from the training samples with the smallest complexity [22]. Thus, SVR amounts to solve a constrained optimization problem, in which the complexity, measured by the norm of the parameters, is minimized. Allowing for the cases where the constraints can not all be satisfied (some points have larger deviation than ε) leads to minimize an ε-insensitive loss function, which yields a zero loss for a point with error less than ε and corresponds to an absolute loss for the others. The SVR algorithm can thus be written as a quadratic programming (QP) problem, where both the ℓ_1-norm of the errors larger than ε and the ℓ_2-norm of the parameters are minimized. To deal with non-linear tasks, SVR uses kernel functions, such as the Radial Basis Function (RBF) kernel, which allow to extend linear methods to non-linear problems via an implicit mapping in a higher dimensional feature space. Compared to neural networks, SVR has the following advantages: automatic selection and sparsity of RBF centers, intrinsic regularization, no local minima (convex problem with a unique solution), and good generalization ability from a limited amount of samples.

Other formulations of the SVR problem minimizing the ℓ_1-norm of the parameters can be derived to yield linear programs (LP) [26, 1, 23, 15]. Some advantages of this latter approach can be noticed compared to the QP formulation such as an increased sparsity of support vectors [26, 1, 23] or the ability to use more general kernels [14]. The remaining of the paper will thus focus on the LP formulation of SVR (LP-SVR).

Support Vector Regression aims at learning an unknown function based only on a training set of N input-output pairs $(\boldsymbol{x}_i, y_i)$ in a black box modeling approach. Incorporating prior knowledge into support vector learning is not trivial and is still a partially open issue. After a presentation of the LP-SVR problem (section 2), the paper proposes to extend it with additional constraints, that are linear in the parameters, in order to include prior knowledge in the learning (section 3). Three methods are exposed respectively for the inclusion of knowledge on the output values (section 3.1), on the derivatives of the model (section 3.2) and the combination of both (section 3.3). Section 4 discusses the various ways of incorporating prior knowledge in the form of simulation data with these techniques. Finally, the methods are tested on the in-cylinder residual gas fraction data in section 5.

Notations: all vectors are column vectors written in boldface and lowercase letters whereas matrices are boldface and uppercase, except for the ith column of a matrix $\boldsymbol{A}$ that is denoted $\boldsymbol{A}_i$. The vectors $\mathbf{0}$ and $\mathbf{1}$ are vectors of appropriate dimensions with all their components respectively equal to 0 and 1. For $\boldsymbol{A} \in \mathbb{R}^{d\times m}$ and $\boldsymbol{B} \in \mathbb{R}^{d\times n}$ containing d-dimensional sample vectors, the "kernel" $\boldsymbol{K}(\boldsymbol{A}, \boldsymbol{B})$ maps $\mathbb{R}^{d\times m} \times \mathbb{R}^{d\times n}$ in $\mathbb{R}^{m\times n}$ with $\boldsymbol{K}(\boldsymbol{A}, \boldsymbol{B})_{i,j} = k(\boldsymbol{A}_i, \boldsymbol{B}_j)$, where $k : \mathbb{R}^{d\times d} \to \mathbb{R}$ is the kernel function. In particular, if $\boldsymbol{x} \in \mathbb{R}^d$ is a column vector then $\boldsymbol{K}(\boldsymbol{x}, \boldsymbol{B})$ is a row vector in $\mathbb{R}^{1\times n}$. The matrix $\boldsymbol{X} \in \mathbb{R}^{N\times d}$ contains all the training samples $\boldsymbol{x}_i$, $i = 1, \dots, N$, as rows. The vector $\boldsymbol{y} \in \mathbb{R}^N$ gathers all the target values y_i for these samples. The kernel matrix $\boldsymbol{K}(\boldsymbol{X}^T, \boldsymbol{X}^T)$ will be written $\boldsymbol{K}$ for short. Uppercase Z is a set containing $|Z|$ vectors that constitute the rows of the matrix $\boldsymbol{Z}$.

2 Kernel regression

In non-linear regression by kernel methods, the function of input $\boldsymbol{x} \in \mathbb{R}^d$ is approximated by a kernel expansion

$$f(\boldsymbol{x}) = \sum_i \alpha_i k(\boldsymbol{x}, \boldsymbol{x}_i) + b = \boldsymbol{K}(\boldsymbol{x}, \boldsymbol{X}^T)\boldsymbol{\alpha} + b \ , \tag{1}$$

where the α_i, contained in the vector $\boldsymbol{\alpha}$, and b are the parameters of the model and $k(.,.)$ is the kernel function. Typical kernel functions are the linear, Gaussian RBF, polynomial and sigmoidal kernels. In this paper, a Gaussian RBF kernel $k(\boldsymbol{x}, \boldsymbol{x}_i) = \exp\left(-\|\boldsymbol{x} - \boldsymbol{x}_i\|^2/2\sigma^2\right)$ is used.

2.1 Linear programming (LP)

In kernel regression via linear programming (LP), the ℓ_1-norm of the parameters $\boldsymbol{\alpha}$ of the kernel expansion is minimized together with the ℓ_1-norm of the errors $\xi_i = y_i - f(\boldsymbol{x}_i)$ by

$$\min_{(\boldsymbol{\alpha}, b)} \|\boldsymbol{\alpha}\|_1 + C \sum_{i=1}^{N} |\xi_i| \ , \tag{2}$$

where a hyperparameter C is introduced to tune the trade-off between the error minimization and the maximization of the function flatness. Instead of the absolute value $|\xi|$ involved in the ℓ_1-norm of the errors, the ε-insensitive loss function defined as

$$|\xi|_\varepsilon = \begin{cases} 0 & \text{if } |\xi| \leq \varepsilon \ , \\ |\xi| - \varepsilon & \text{otherwise,} \end{cases} \tag{3}$$

can be used to yield Linear Programming Support Vector Regression (LP-SVR). A possible formulation of the corresponding problem involves $4N + 1$ variables [23]. In this paper, we will follow the approach of [15] that involves only $3N + 1$ variables. Introducing two sets of optimization variables, in two positive slack vectors $\boldsymbol{a}$ and $\boldsymbol{\xi}$, this problem can be implemented as the linear program solvable by standard optimization softwares such as MATLAB linprog. In this scheme, the LP-SVR problem becomes

$$\begin{array}{rrcl} & \displaystyle\min_{(\boldsymbol{\alpha}, b, \boldsymbol{\xi}, \boldsymbol{a})} & \mathbf{1}^T \boldsymbol{a} + C \mathbf{1}^T \boldsymbol{\xi} & \\ \text{s.t.} & -\boldsymbol{\xi} \leq & \boldsymbol{K}\boldsymbol{\alpha} + b\mathbf{1} - \boldsymbol{y} & \leq \boldsymbol{\xi} \\ & \mathbf{0} \leq & \mathbf{1}\varepsilon & \leq \boldsymbol{\xi} \\ & -\boldsymbol{a} \leq & \boldsymbol{\alpha} & \leq \boldsymbol{a} \ . \end{array} \tag{4}$$

The last set of constraints ensures that $\mathbf{1}^T \boldsymbol{a}$, which is minimized, bounds $\|\boldsymbol{\alpha}\|_1$. In practice, sparsity is obtained as a certain number of parameters α_i will tend to zero. The input vectors $\boldsymbol{x}_i$ for which the corresponding α_i are non-zero are called *support vectors* (SVs).

The parameter ε can also be introduced as a variable in the cost function to be tuned automatically by the algorithm [23, 15].

In the LP formulation, only symmetry of the kernel is required [15]. It is not necessary for the kernel to satisfy Mercer's conditions (or positive semidefiniteness) as in the original QP form of the SVR problem [22].

3 Incorporating prior knowledge

In this section, two different forms of prior knowledge on the function to be approximated are included in the learning. The first one considers a set Z of points $\boldsymbol{x}_p$, $p = 1, \dots, |Z|$, regrouped in the matrix $\boldsymbol{Z}$, for which the output values y_p, $p = 1, \dots, |Z|$, regrouped in $\boldsymbol{y}_p$, are provided by a prior model or a simulator. In this setting, the aim is to enforce the equality constraints

$$f(\boldsymbol{x}_p) = y_p, \ p = 1, \dots |Z| \ , \tag{5}$$

on the model f.

The second setting considers the incorporation of prior knowledge on the derivatives of the function, possibly given by a prior model. The combination of both types of prior knowledge is also considered at the end of the section.

3.1 Knowledge on output values

Prior knowledge on the function to approximate is assumed to take the form of $|Z|$ particular points $\boldsymbol{x}_p$ in Z for which the output values y_p are known. However, applying constraints such as (5) to the problem (4) would lead to an exact fit to these points, which may not be advised if these are given by a simulator possibly biased. Moreover, all these constraints may lead to an unfeasible problem if they cannot all be satisfied simultaneously. To deal with this case, the equalities (5) can rather be included as soft constraints by introducing a vector $\boldsymbol{z} = [z_1 \ \dots \ z_p \ \dots \ z_{|Z|}]^T$ of positive slack variables bounding the error on (5) as

$$|y_p - f(\boldsymbol{x}_p)| \leq z_p, \ p = 1, \dots |Z| \ . \qquad (6)$$

The ℓ_1-norm of the slack vector $\boldsymbol{z}$ is then added to the criterion of (4), with a trade-off parameter λ, in order to be minimized. The trade-off parameter λ allows to tune the influence of the prior knowledge on the model and thus incorporate approximate knowledge.

It is also possible to include almost exact or biased knowledge by authorizing violations of the equality constraints (5) that are less than a threshold. Using the ε-insensitive loss function (3) on the prior knowledge errors z_p with a threshold ε_p, different than the one, ε, used for the training set, leads to the following linear program

$$\begin{aligned}
&\min_{(\boldsymbol{\alpha}, b, \boldsymbol{\xi}, \boldsymbol{a}, \boldsymbol{z})} \ \mathbf{1}^T\boldsymbol{a} + C\mathbf{1}^T\boldsymbol{\xi} + \lambda\mathbf{1}^T\boldsymbol{z} \\
&\text{s.t.} \\
&\begin{array}{rcl}
-\boldsymbol{\xi} \leq & \boldsymbol{K}\boldsymbol{\alpha} + b\mathbf{1} - \boldsymbol{y} & \leq \boldsymbol{\xi} \\
\mathbf{0} \leq & \mathbf{1}\varepsilon & \leq \boldsymbol{\xi} \\
-\boldsymbol{a} \leq & \boldsymbol{\alpha} & \leq \boldsymbol{a} \\
-\boldsymbol{z} \leq & \boldsymbol{K}(\boldsymbol{Z}^T, \boldsymbol{X}^T)\boldsymbol{\alpha} + b\mathbf{1} - \boldsymbol{y}_p & \leq \boldsymbol{z} \\
\mathbf{0} \leq & \mathbf{1}\varepsilon_p & \leq \boldsymbol{z} \ ,
\end{array}
\end{aligned} \qquad (7)$$

where the two last sets of $|Z|$ constraints stand for the inclusion of prior knowledge.

3.2 Knowledge on the derivatives

Knowledge on the derivatives allows to include information on local maxima or minima, saddle-points and so on. In some applications, one may also wish to retain certain properties of a prior model such as the shape or the roughness at some specific points $\boldsymbol{x}_p$ in Z. This problem corresponds to learn a new model while constraining certain of its derivatives to equal those of the prior model at these particular points.

Consider that prior knowledge on the derivative of the function f with respect to the jth component x^j of $\boldsymbol{x} \in \mathbb{R}^d$ is available as

$$\left.\frac{\partial f(\boldsymbol{x})}{\partial x^j}\right|_{\boldsymbol{x}_p} = y'_p, \ \forall \boldsymbol{x}_p \in Z \ . \qquad (8)$$

This prior knowledge can be enforced in the training by noticing that the kernel expansion (1) is linear in the parameters $\boldsymbol{\alpha}$, which allows to write the derivative of the model output with respect to x^j as

$$\frac{\partial f(\boldsymbol{x})}{\partial x^j} = \sum_i \alpha_i \frac{\partial k(\boldsymbol{x}, \boldsymbol{x}_i)}{\partial x^j} = \boldsymbol{r}_j(\boldsymbol{x})^T\boldsymbol{\alpha} \ , \quad (9)$$

where $\boldsymbol{r}_j(\boldsymbol{x}) = [\partial_{x^j}k(\boldsymbol{x}, \boldsymbol{x}_1) \ \dots \ \partial_{x^j}k(\boldsymbol{x}, \boldsymbol{x}_i) \ \dots \ \partial_{x^j}k(\boldsymbol{x}, \boldsymbol{x}_N)]^T$ is of dimension N. The derivative (9) is linear in $\boldsymbol{\alpha}$. In fact, the form of the kernel expansion implies that the derivatives of any order with respect to any component are linear in $\boldsymbol{\alpha}$. The prior knowledge (8) can thus be included by defining $\boldsymbol{R} = \left[\boldsymbol{r}_j(\boldsymbol{x}_1) \ \dots \ \boldsymbol{r}_j(\boldsymbol{x}_p) \ \dots \ \boldsymbol{r}_j(\boldsymbol{x}_{|Z|})\right]^T$ of dimension $|Z| \times N$, $\boldsymbol{y}'_p = [y'_1 \ \dots \ y'_p \ \dots \ y'_{|Z|}]^T$ and solving the problem

$$\begin{aligned}
&\min_{(\boldsymbol{\alpha}, b, \boldsymbol{\xi}, \boldsymbol{a}, \boldsymbol{z})} \ \mathbf{1}^T\boldsymbol{a} + C\mathbf{1}^T\boldsymbol{\xi} + \lambda\mathbf{1}^T\boldsymbol{z} \\
&\begin{array}{rrcl}
\text{s.t.} & -\boldsymbol{\xi} \leq & \boldsymbol{K}\boldsymbol{\alpha} + b\mathbf{1} - \boldsymbol{y} & \leq \boldsymbol{\xi} \\
& \mathbf{0} \leq & \mathbf{1}\varepsilon & \leq \boldsymbol{\xi} \\
& -\boldsymbol{a} \leq & \boldsymbol{\alpha} & \leq \boldsymbol{a} \\
& -\boldsymbol{z} \leq & \boldsymbol{R}\boldsymbol{\alpha} - \boldsymbol{y}'_p & \leq \boldsymbol{z} \ .
\end{array}
\end{aligned} \qquad (10)$$

The extension to any order of derivative is straightforward. Thus, this method allows to incorporate prior knowledge on any derivative for any set of points possibly different for each derivative. In comparison, the methods described in [13] and [12] requires the prior derivative values on all the training points and these points only.

3.3 Combining both forms of constraints

In the case where the training data do not cover the whole input space, extrapolation occurs, which can become a problem when using local kernels such as the RBF kernel. To avoid this problem, the points $\boldsymbol{x}_p$ can be introduced in the training set as potential support vectors (or RBF centers). Instead of simply adding them in the training set, it is proposed to use a loss function with a different insensitivity width ε_p for these points. This method considers the case where the information on y_p is less accurate than on y'_p. In other words, the available prior model is accurate in shape but biased.

The resulting complete problem reads

$$\begin{aligned}
&\min_{(\boldsymbol{\alpha},b,\boldsymbol{\xi},\boldsymbol{a},\boldsymbol{z},\boldsymbol{v})} \quad \mathbf{1}^T\boldsymbol{a} + C\mathbf{1}^T\boldsymbol{\xi} + \lambda_1\mathbf{1}^T\boldsymbol{z} + \lambda_2\mathbf{1}^T\boldsymbol{v}\\
&\text{s.t.}\\
&\begin{array}{rcl}
-\boldsymbol{\xi} \leq & \boldsymbol{K}(\boldsymbol{X}^T, \begin{bmatrix}\boldsymbol{X}\\ \boldsymbol{Z}\end{bmatrix}^T)\boldsymbol{\alpha} + b\mathbf{1} - \boldsymbol{y} & \leq \boldsymbol{\xi}\\
\mathbf{0} \leq & \mathbf{1}\varepsilon & \leq \boldsymbol{\xi}\\
-\boldsymbol{a} \leq & \boldsymbol{\alpha} & \leq \boldsymbol{a}\\
-\boldsymbol{z} \leq & \boldsymbol{K}(\boldsymbol{Z}^T, \begin{bmatrix}\boldsymbol{X}\\ \boldsymbol{Z}\end{bmatrix}^T)\boldsymbol{\alpha} + b\mathbf{1} - \boldsymbol{y}_p & \leq \boldsymbol{z}\\
\mathbf{0} \leq & \mathbf{1}\varepsilon_p & \leq \boldsymbol{z}\\
-\boldsymbol{v} \leq & \boldsymbol{R}\boldsymbol{\alpha} - \boldsymbol{y}'_p & \leq \boldsymbol{v}\ ,
\end{array}
\end{aligned} \tag{11}$$

where λ_1 and λ_2 are respectively the trade-off parameters for the knowledge on the output values and on the derivatives. In this problem, the first three sets of constraints correspond to the standard learning (4) on the training set $(\boldsymbol{X}, \boldsymbol{y})$. The difference is that the samples in $\boldsymbol{Z}$ are added as potential support vectors. The next two sets of constraints include prior knowledge on the output values as in section 3.1. The last set of constraints involving the slack vector $\boldsymbol{v}$ incorporates the information on the derivatives as in (10). It must be noted that the number of parameters in the vector $\boldsymbol{\alpha}$ is now $N + |Z|$ and that the dimension of $\boldsymbol{R}$ has increased to $|Z| \times (N + |Z|)$.

4 Best use of simulation data

This section aims at studying the various ways of enhancing the learning with simulation data. Consider the problem of modeling a function based on a limited amount of experimental data (the training set) and a physical simulation model (or simulator). If large regions of the input space are not covered by the experimental data, extrapolation in these areas will be difficult. The simulator can provide data in these regions. The question is: what is the best way of incorporating these data in the learning of the model?

The first and most simple method consists in extending the training set of real data with simulation samples. Then a standard learning algorithm can take into account all the information and hopefully provide a better model. This approach is similar in flavor to the virtual sample approach extensively studied in pattern recognition for the incorporation of transformation-invariance [18, 19, 10]. In this classification framework, virtual samples are generated by known transformations of the training patterns that leave their class labels unchanged and then added to the training set. This approach has been very successful in the pattern recognition field [21, 11], where the prior knowledge of transformation-invariance is exact, i.e., for example, the statement "the image of a character still represents the same character if translated or rotated by a small amount" holds true. However, in the case considered here, the simulation data can be biased as the physical model may not be fully accurate.

There is therefore a need to control the influence of simulation data on the learning so that the resulting model do not fit exactly to them. The method proposed in section 3.1 uses two different loss functions: one on the training set and another one, possibly less restrictive, on the simulation samples. In practice, the algorithm (7) is used with the parameter ε_p set to take into account the possible bias between the simulation and real data.

If the physical model is known to be accurate in shape but biased, another solution consists in constraining the derivatives of the model to fit to the derivatives of the physical model. Doing so, one may expect to retain the overall shape of the physical model while fitting it to the available real data. In the setting of the following experiments (section 5), only the simulation data are available. In practice, a first model is trained by a standard LP-SVR algorithm (4) on the simulation data to yield a prior model. This prior model can then be used to compute the derivatives at these same points. The algorithm (10) of section 3.2 is then applied to build a new model with constrained derivatives.

As noted in section 3.3, the local behavior of RBF kernels may become a serious drawback when reducing the number of training samples. To circumvent this difficulty, the simulation data should be added as potential support vectors. However, if these data are known to be inaccurate, simply adding them to the training set will not lead to a good model. The algorithm (11) allows to take these potential SVs into account without constraining the model to exactly fit to them.

5 Application

The various ways of incorporating simulation data into the training will now be studied on a real-life application described below.

5.1 Estimation of in-cylinder residual gas fraction

The application deals with the estimation of residual gases in the cylinders of Spark Ignition (SI) engines with Variable Camshaft Timing (VCT). VCT allows the timing of the intake and exhaust valves to be changed while the engine is in operation. VCT is used to improve performance in terms of emissions, fuel economy, peak torque, and peak power [9].

The air path control of SI engines is a crucial task because the torque provided by the engine is directly linked to the air mass trapped in the cylinders [3]. When considering new air actuators such as VCT, the estimation of in-cylinder air mass is more involved than for basic SI engines. Indeed, VCT authorizes phenomena such as air scavenging (from intake to exhaust manifolds, with turbocharging) or backflow (from exhaust manifold to cylinders).

In this context, it is important to estimate the residual gas mass fraction

$$\chi_{res} = \frac{m_{res}}{m_{tot}}, \tag{12}$$

where m_{tot} is the total gas mass trapped in the cylinder and m_{res} is the mass of residual gases, which are burned gases present in the cylinder when the valves are closed before the new combustion and which are due to the dead volumes or the backflow. Knowing this fraction allows to control torque as well as pollutant emissions.

The residual gas mass fraction χ_{res} can be expressed as a function of the engine speed N_e, the ratio p_{man}/p_{exh}, where p_{man} and p_{exh} are respectively the (intake) manifold pressure and the exhaust pressure, and an overlapping factor OF, which is an image of the time during which the valves are opened together. There exists a corresponding mean value model [4], that includes some constants to be identified from experiments on a particular engine. Beside this mean value model, there is no standard sensor to measure this fraction online.

The available data are provided, on one hand, from the modeling and simulation environment Amesim [8], which uses a high frequency zero-dimensional thermodynamic model [5] and, on the other hand, from off line measurements, which are accurate, but complex and costly to obtain, by direct in-cylinder sampling [5]. The problem is thus as follows. How to obtain a simple, embeddable, black box model with a good accuracy and a large validity range for the real engine, from precise real measurements as less numerous as possible and a representative, but possibly biased, prior simulation model?

The problem thus posed, although particular, is very representative of numerous situations met in engine control, and more generally in engineering, where complex models, more or less accurate, exist and where the experimental data which can be used for calibration are difficult or expensive to obtain.

5.2 Experiments and results

Three datasets are built from the available data composed of 26 experimental samples plus 26 simulation samples:

- the training set composed of a limited amount of real data (N samples),
- the test set composed of independent real data ($26 - N$ samples),
- the simulation set composed of data provided by the simulator (26 samples).

Various sizes N of the training set will be considered in order to study the effect of the number of training samples on the model. The test samples are assumed to be unknown during the training and are retained for testing only.

The residual gas mass fraction χ_{res}, given in percentages, takes values in the range $[5, 30]$. The ranges of values for the three inputs are: N_e (rpm) $\in \{1000, 2000\}$, $p_{man}/p_{exh} \in [0.397, 0.910]$ and OF ($°CA/m$) $\in [0, 2.8255]$. The datasets are shown in Figure 1 for $N = 3$.

The different situations and the various ways of incorporating the simulation data are considered in the following models.

1. *Experimental model*: the simulation data are not available. Standard LP-SVR training (4) on the training set only is used to determine the model.
2. *Prior model*: the experimental data are not available. LP-SVR training (4) on the simulation set only is used to determine the model.
3. *Mixed model*: all the data are available and mixed together. LP-SVR training (4) on the training set extended with the simulation data is used to determine the model.
4. *O-model*: the simulation data are considered as prior knowledge on *Output* values. Algorithm (7) is used to train the model.
5. *D-model*: the simulation data are used to build a *prior model*, which is then used to provide prior knowledge on *Derivative* values with respect to the input p_{man}/p_{exh}. Algorithm (10) with derivative constraints is used to train the model.
6. *OD-model*: the simulation data are both considered as prior knowledge on *Output* values and used to build a *prior model* in order to give prior knowledge on *Derivative* values as for model 5. Algorithm (11) is used to train the OD-model.

These models are evaluated on the basis of three indicators defined below.

- RMSE total: root mean square error on the whole real dataset (26 samples)
- RMSE test: root mean square error on the test set ($26 - N$ samples) that is

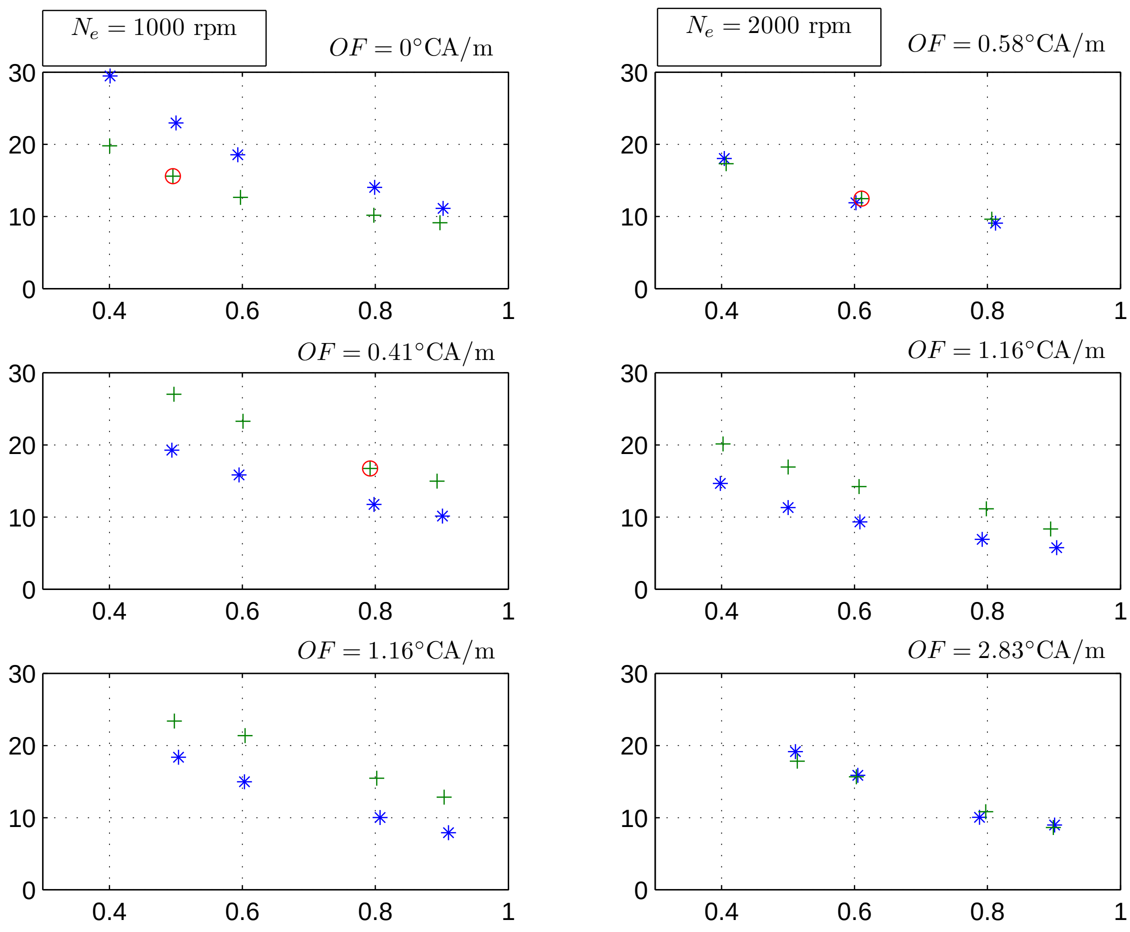

Figure 1. Residual gas mass fraction χ_{res} in percentages as a function of the ratio p_{man}/p_{exh} for two engine speeds N_e and different overlapping factors OF. The 26 experimental data are represented by plus signs ($+$) with a superposed circle ($\oplus$) for the 3 points retained as training samples. The 26 simulation data appear as asterisks ($*$).

not used for training

- MAE total: maximum absolute error on the whole real dataset (26 samples)

Before training, the variables are normalized with respect to their mean and standard deviation. When both experimental and simulation data are available, the simulation data are preferred since they are supposed to cover a wider region of the input space. Thus, the mean and standard deviation are determined on the simulation data for all the models except for the *experimental model*, in which case the training set must be used to determine the normalization parameters.

The different hyperparameters are set according to the following heuristics. One goal of the problem is to obtain a model that is accurate on both the training and test samples (the training points are part of the performance index *RMSE total*). Thus C is set to a large value ($C = 1000$) in order to ensure a good approximation of the training points. Accordingly, ε is set to 0.001 in order to approximate the real data well. Since all standard deviations of the inputs equal 1 after normalization, the RBF kernel width σ is set to 1. Besides, when using prior knowledge on output values (model 4) and setting ε_p, one has both the real and simulation data at hand. Thus, it is possible to compute an estimate of the maximum absolute error (MAE) obtained by the simulator by looking at the MAE obtained by the *prior model* on the training set (available real data). ε_p is then set accordingly by taking into account the normalization step ($\varepsilon_p = 1.6$). For the training of both models 4 and 5, the trade-off parameter λ is set to 10. For model 6, λ_1 is set to 1 since including the prior knowledge weighted by λ_1 is not the main concern. It is mainly used to add potential support vectors in the training with a large ε_p in order not to fit exactly to these points. The information assumed to be more relevant in this case concerns the derivatives. λ_2 is thus set to 10.

Results. The number N of points retained as training samples is first set to 15, which is about half of the available experimental data. The results in this setting appear at the top of Table 1. These show that the model of the simulator, the *prior model*, is actually biased and leads a large error when tested on the real data. As a consequence, the *mixed model* that simply considers simulation data as training samples cannot yield good results from inhomogeneous and contradictory data. However, when the simulation data are incorporated in the *O-model* by the proposed method of section 3.1, which allows for a certain amount of inaccuracy with respect to these data, better results can be obtained in comparison to the *experimental model* trained on the limited amount of real data. Interestingly, though including prior knowledge on the derivatives in the *D-model* does not yield better results, combining both types of prior knowledge on the function values and on the derivatives in the *OD-model* leads to a notable decrease of the error. This improvement may also be the result of adding the simulation data as potential support vectors. This effect will become more obvious in the following experiments.

Now, the effect of reducing the number of training samples N is studied in order to test the limits of the method. Two sets of experiments are performed for $N = 6$ and $N = 3$. The results in Table 1 show that models 1 and 3 cannot deal with these cases. Moreover, when the number of training points becomes too small ($N = 3$), models 4 and 5, that do not incorporate the simulation data as potential support vectors, become almost constant and thus inefficient. This is due to the fact that these models have not enough free parameters (only 3 plus a bias term) and thus cannot accurately model the data. In these cases, the RMSE is irrelevant. On the contrary, the *OD-model* does not suffer from this problem and can yield good results

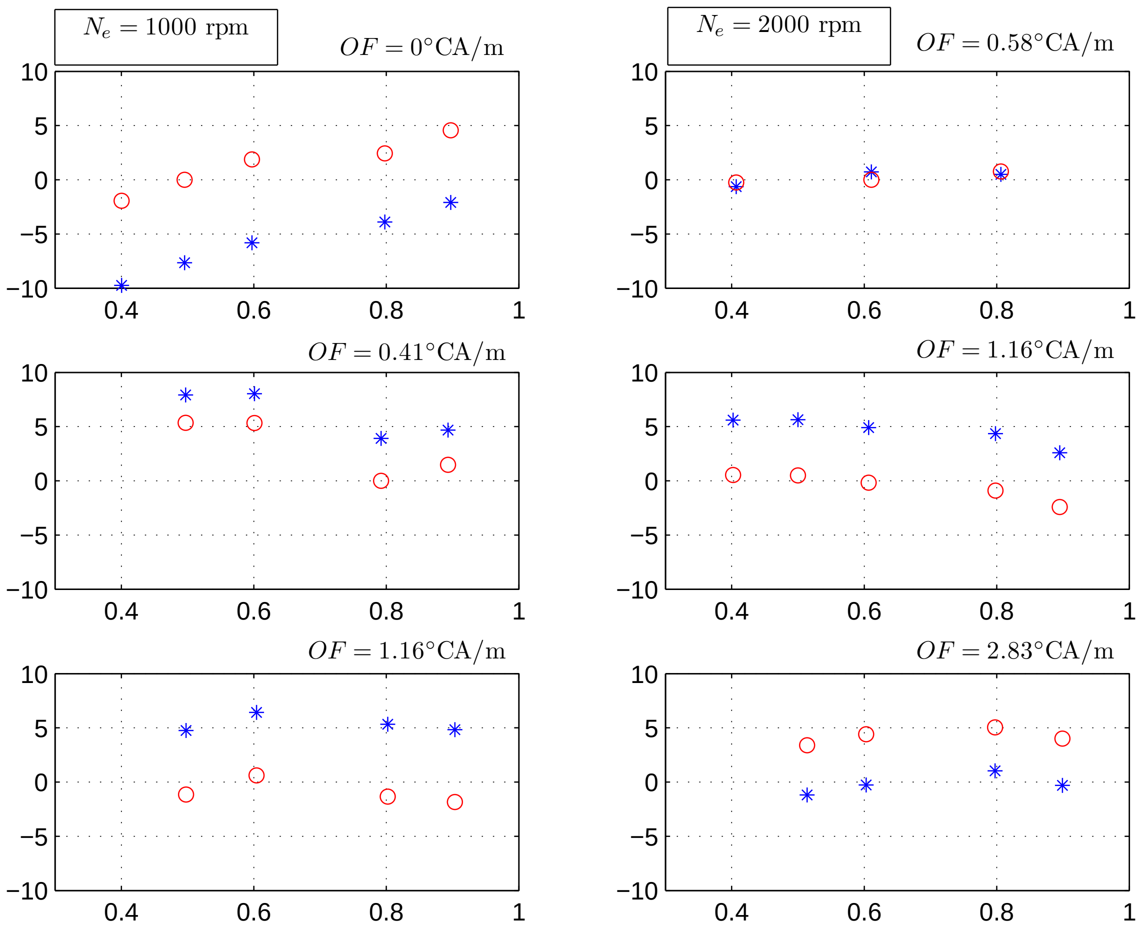

Figure 2. Errors on the experimental data (training and test samples) for the *prior model* ($*$) and the *OD-model* ($\circ$) trained with only three samples ($N = 3$).

Table 1. Errors on the residual gas mass fraction for various training set sizes N. '–' appears when the result is irrelevant (model mostly constant).

N	Model	RMSE total	RMSE test	MAE total
15	1 (experimental model)	1.53	2.35	4.10
	2 (prior model)	4.93	5.02	9.74
	3 (mixed model)	4.22	5.09	7.80
	4 (O-model)	1.49	2.29	4.05
	5 (D-model)	1.60	2.46	3.93
	6 (OD-model)	**0.73**	**1.12**	**2.29**
6	1 (experimental model)	6.00	6.84	15.83
	2 (prior model)	4.93	4.86	9.74
	3 (mixed model)	4.88	4.85	9.75
	4 (O-model)	3.38	3.86	9.78
	5 (D-model)	3.50	3.99	10.42
	6 (OD-model)	**2.14**	**2.44**	**5.94**
3	1 (experimental model)	–	–	–
	2 (prior model)	4.93	4.93	9.74
	3 (mixed model)	4.86	4.89	9.75
	4 (O-model)	–	–	–
	5 (D-model)	–	–	–
	6 (OD-model)	**2.64**	**2.81**	**5.36**

compared to the *prior model* with very few training samples, as shown in Figure 2. The performance is still reasonable, though being less than for $N = 15$, and decreases only slightly when reducing the training set size from 6 to 3. It must be noted that the error of this model is about half of the errors obtained by the *prior* and *mixed models*, which correspond to the standard methods to include simulation data.

6 Conclusion

This paper provided simple and effective techniques for the incorporation of prior knowledge into LP-SVR learning. The prior information that can be taken into account may be given in terms of output values as well as derivative values on a set of points. Various methods based on these techniques have been studied for the inclusion of knowledge in the form of simulation data. The proposed methods have been tested on the estimation of in-cylinder residual gas fraction application. In this context, real data are available only in a limited number due to the cost of experimental measurements but additional data can be obtained thanks to a complex physical simulator. The output of the simulator being biased but providing rather good information on the overall shape of the model, the best method constrains the derivatives of the model to fit to the ones of a prior model trained on the simulation data.

Acknowledgments

The authors thank P. Giansetti for providing the data used in this work.

References

[1] K. P. Bennett. Combining support vector and mathematical programming methods for classification. In Schölkopf et al. [20], pages 307–326.

[2] A. Chevalier, M. Müller, and E. Hendricks. On the validity of mean value engine models during transient operation. *SAE Technical Papers*, (2000-01-1261), 2000.

[3] G. Colin, Y. Chamaillard, G. Bloch, and G. Corde. Neural control of fast nonlinear systems, Application to a turbocharged SI engine with VCT. *IEEE Trans. on Neural Networks*, 18(4), 2007.

[4] J. W. Fox, W. K. Cheng, and J. B. Heywood. A model for predicting residual gas fraction in spark-ignition engines. *SAE Technical Papers*, (931025), 1993.

[5] P. Giansetti, G. Colin, P. Higelin, and Y. Chamaillard. Residual gas fraction measurement and computation. *International Journal of Engine Research*, to appear, 2007.

[6] L. Guzzella and C.H. Onder. *Introduction to Modeling and control of Internal Combustion Engine Systems*. Springer, 2004.

[7] J. B. Heywood. *Internal Combustion Engines Fundamentals*. McGraw-Hill, New York, NY, USA, 1988.

[8] Imagine. Amesim web site. www.amesim.com, 2006.

[9] M. Jankovic, S. Magner, S. Hsieh, and J. Konesol. Transient effects and torque control of engines with variable cam timing. In *Proc. of the American Control Conference, San Francisco, CA*, volume 1, pages 50–54, Sept. 2000.

[10] F. Lauer and G. Bloch. Incorporating prior knowledge in support vector machines for classification: a review. *Neurocomputing*, 2007.

[11] F. Lauer, C. Y. Suen, and G. Bloch. A trainable feature extractor for handwritten digit recognition. *Pattern Recognition*, 40:1816–1824, 2007.

[12] M. Lázaro, F. Pérez-Cruz, and A. Artés-Rodriguez. Learning a function and its derivative forcing the support vector expansion. *IEEE Signal Processing Letters*, 12:194–197, 2005.

[13] M. Lázaro, I. Santamaria, F. Pérez-Cruz, and A. Artés-Rodriguez. Support vector regression for the simultaneous learning of a multivariate function and its derivatives. *Neurocomputing*, 69:42–61, 2005.

[14] O. Mangasarian. Generalized support vector machines. In A. Smola, P. Bartlett, B. Schölkopf, and D. Schuurmans, editors, *Advances in Large Margin Classifiers*, pages 135–146, Cambridge, MA, USA, 2000. MIT Press.

[15] O. L. Mangasarian and D. R. Musicant. Large scale kernel regression via linear programming. *Machine Learning*, 46(1-3):255–269, 2002.

[16] D. Mattera and S. Haykin. Support vector machines for dynamic reconstruction of a chaotic system. In *Advances in kernel methods: support vector learning* [20], pages 211–241.

[17] K.R. Müller, A. Smola, G. Rätsch, B. Schölkopf, J. Kohlmorgen, and V. Vapnik. Predicting time series with support vector machines. In *Proc. of the Int. Conf. on Artificial Neural Networks*, pages 999–1004, 1997.

[18] T. Poggio and T. Vetter. Recognition and structure from one 2D model view: Observations on prototypes, object classes and symmetries. Technical Report AIM-1347, Massachusetts Institute of Technology, Cambridge, MA, USA, 1992.

[19] B. Schölkopf, C. Burges, and V. Vapnik. Incorporating invariances in support vector learning machines. In C. von der Malsburg, W. von Seelen, J. C. Vorbrüggen, and B. Sendhoff, editors, *ICANN*, volume 1112 of *Lecture Notes in Computer Science*, pages 47–52. Springer, 1996.

[20] B. Schölkopf, C. J.C. Burges, and A. J. Smola, editors. *Advances in kernel methods: Support vector learning*, Cambridge, MA, USA, 1999. MIT Press.

[21] P. Y. Simard, D. Steinkraus, and J. C. Platt. Best practices for convolutional neural networks applied to visual document analysis. In *Proc. of the 7th Int. Conf. on Document Analysis and Recognition, Edinburgh, Scotland, UK*, volume 2, pages 958–962, 2003.

[22] A. J. Smola and B. Schölkopf. A tutorial on support vector regression. *Statistics and Computing*, 14(3):199–222, 2004.

[23] A. J. Smola, B. Schölkopf, and G. Rätsch. Linear programs for automatic accuracy control in regression. In *Proc. of the 9th Int. Conf. on Artificial Neural Networks, Edinburgh, UK*, volume 2, pages 575–580, 1999.

[24] M. O. Stitson, A. Gammerman, V. Vapnik, V. Vovk, C. Watkins, and J. Weston. Support vector regression with ANOVA decomposition kernels. In Schölkopf et al. [20], pages 285–291.

[25] V. N. Vapnik. *The nature of statistical learning theory*. Springer-Verlag, New York, NY, USA, 1995.

[26] J. Weston, A. Gammerman, M. O. Stitson, V. Vapnik, V. Vovk, and C. Watkins. Support vector density estimation. In Schölkopf et al. [20], pages 293–305.