To appear in *Engineering Optimization*
Vol. 00, No. 00, Month 20XX, 1–22

Support Vector Machine Classification Applied to the Parametric Design of Centrifugal Pumps†

E. Riccietti and J.Bellucci, M.Checcucci, M.Marconcini, A.Arnone

(*Received 00 Month 20XX; final version received 00 Month 20XX*)

In this article the parametric design of centrifugal pumps is addressed. To deal with this problem, an approach based on coupling expensive Computational Fluid Dynamics (CFD) computations with Artificial Neural Networks (ANN) as a regression meta-model had been proposed in Checcucci et al. (2015), 'A Novel Approach to Parametric Design of Centrifugal Pumps for a Wide Range of Specific Speeds', ISAIF 12, paper n.121. Here, the previously proposed approach is improved by including also the use of Support Vector Machines (SVM) as a classification tool. The classification process is aimed at identifying parameters combinations corresponding to manufacturable machines among the much larger number of unfeasible ones. A binary classification problem on an unbalanced dataset has to be faced. Numerical tests show that the addition of this classification tool helps to considerably reduce the number of CFD computations required for the design, providing large savings in computational time.

Keywords: Support vector machines; parametric design; binary classification; centrifugal pumps; unbalanced dataset.

1. Introduction

In the last years the approach of the designer to the aerodynamic and mechanical redesign of a turbomachinery component is changed with respect to some decades ago, (12). Often the requirements of the customer lead to analyse the performance of a component with complex three-dimensional geometry and to extend these investigations to different operating conditions simultaneously, with the aim of optimizing the performance under tight constraints. To meet all the customer requirements, it is necessary to accept a compromise between reliability, low-cost manufacturing and high aerodynamic efficiency, (12).

Nowadays, the exponential increase of computational power allows to face such problems evaluating the performance objective functions through CFD (Computational Fluid Dynamics) analysis, (20). Anyway these calculations are computationally expensive, and even if reliability is still the most important aspect that guides the choice of the final geometry, the competitiveness of the business requires the design process to be as short as possible.

The research in this field is currently active, providing a wide range of different strategies for the end user to handle the optimization procedure of a machine component. Gradient methods, methods based on the response surface approximation (e.g. Artificial Neural Network (ANN), Support Vector Machine (SVM), Design of Experiment

Dipartimento di Matematica e Informatica 'Ulisse Dini', Università di Firenze, viale G.B. Morgagni 67a, 50134 Firenze, Italia
Dipartimento di Ingegneria Industriale, Università di Firenze, via S. Marta 3, 50139 Firenze, Italia
†Work partially supported by INdAM-GNCS

(D.O.E.)), exploratory techniques (e.g. Genetic Algorithm, Simulated Annealing, Particle Swarm Optimization Algorithm), adjoint methods or a combination of these ones, are the most used techniques, see among the others (22), (4), (13), (29), (20), (10), (6). Among the different strategies, the methods based on the response surface have reached a good level of maturity and represent a good compromise in terms of time-consumption and prediction accuracy. Regression meta-models are employed to predict values of the functions describing the components performance and to build the response surface, to reduce the amount of required CFD computations, (22; 21). These indeed can be restricted just to the amount necessary to build a performance database to train the meta-model. Thus, the challenge is to perform the lowest number of computations and to obtain the highest accuracy in the approximation of the response surface of the problem, on which a multi-objective optimization algorithm is run to find the best compromise among the considered performance functions.

Usually, a redesign or an optimization procedure starts from a baseline configuration that is geometrically close to the final one. All the tools (e.g. for geometry parameterization, mesh generation, CFD solver etc.) involved in the process are automated and fine tuned for the specific application, in order to work very well within the whole design space of interest, (12). Thus, many of the issues concerning manufacturing and geometrical constraints, and the ones due to the computational setup (in particular the mesh generation) are a priori taken into account during the tuning of the tools. As a result, all (or almost all) the computations performed can be used to form the performance database.

The case in which a parametric design has to be faced is different, (15). Generally, a new design starts from scratch and relies on a quick and flexible design tool, capable of describing in a continuous manner the whole range of geometrical variability of a family of components. The design space investigated to meet the customer requirements becomes really vast, so that a high number of points is required to adequately cover it and consequently a very large amount of computations is needed to accurately train the meta-model. Moreover it is difficult to estimate the required number of computations, as it will be higher than the one the designer could expect. In fact, no matter how robust the tool is, it is impossible to take a priori into account all the manufacturing or geometrical constraints. Then a designer could experience that many of the analysed geometries will result non feasible from a manufacturing point of view, or will reach a poor computational convergence. In the following these geometries will be addressed as *unfeasible*, while all the others will be addressed as *feasible*. As an outcome, the obtained database will result strongly unbalanced, as generally the unfeasible geometries will be many more than the others. The use of such a database to train the meta-model would lead to a very poor accuracy, or even to the failure, of meta-model training. As an outcome, the number of computations necessary to generate a suitable performance database, with enough feasible features, will increase exponentially, and consequently the time needed to perform computations.

To the authors' knowledge, a strategy to efficiently handle a parametric design in which the issue of the database's lack of balancedness occurs has not been addressed yet.

Then, in this article a strategy to overcome the aforementioned problem is described and validated. A hybrid approach is proposed which is a modification of the procedure described in (9) which couples high-fidelity three-dimensional Reynolds Averaged Navier-Stokes (RANS) equations computations and ANN (regression mode). That procedure is improved, including also the use of a classifier with the aim of discarding the unfeasible parameters combinations.

Support Vector Machines (SVM) are employed as a classifier, as their capability has been proven in many different scientific fields, see for example (27; 19; 26), and also approaches based on Support Vector Machine has been successfully applied within a

hybrid structure together with Artificial Neural Networks or Genetic Algorithms, (24; 14; 17). SVM indeed, are characterized by high flexibility thanks to the possibility of choosing among various kernels, also different from the linear one. This allows to classify a wide range of datasets with high precision, improving on non linearly separable data with respect to linear classifiers. This adaptability is also improved by the possibility of tuning the free parameters for the specific application, (23). Moreover, by a simple modification of the standard approach, SVM can be used as a powerful tool to handle unbalanced datasets, (7), as it is needed in the application considered in this article.

The remainder of this article is organized as follows. First, in Section 2 a brief introduction to the machine learning tools employed in the proposed procedure is provided. Then, in Section 3 the industrial problem under consideration is presented, pointing out the drawbacks related to the approach presented in (9). In Section 4 the proposed hybrid approach is described. In Section 5 the issues related to the need of handling an unbalanced dataset in the classification process are discussed. Finally in Section 6 the benefits of the use of the hybrid approach on different datasets arising from the considered industrial application are shown.

2. Machine learning tools involved in the parametric design

In this section a brief description of the machine learning tools that will be employed in the procedure proposed in this article, which are Artificial Neural Network and Support Vector Machines, is provided.

Machine learning meta-models are able to learn a task, for example to approximate a function or classify data, from given examples. The examples are n-dimensional vectors called *features* or *samples* and in the following a sample will be denoted by $\mathbf{x} = [x_1, \ldots, x_n] \in \mathrm{R}^n$. The main feature of machine learning meta-models is that their employment is based on two different steps:

- a **training phase**, in which the model learns the task it has to perform from some given examples that form a set called *training set*.
- an **execution phase**, in which the trained model is used to perform the learned task on new samples.

2.1 *ANN: Artificial Neural Networks*

In the approach presented in this article Artificial Neural Networks (ANN) will be used to approximate values of pumps performance functions.

An Artificial Neural Network is a mathematical model whose structure is thought to try to replicate the functioning of a human brain, (11). It is composed of artificial nodes u_j known as neurons, which are connected together to form a network which mimics a biological neural network. Each neuron u_j can be thought of as a unit to which a transfer function t_j is associated and that receives some inputs $x_{1j}, \ldots, x_{mj}$ and produces a single output $y_j = t_j(I_j) = t_j(\sum_{i=1}^{m} x_{ij})$.

Usually the same transfer function is used for all neurons, i.e. $t_j = t$ for all j and the most common is the sigmoidal function:

$$t(I_j) = \frac{1}{1 + e^{-I_j}}, \tag{1}$$

that is employed also in this article. Two neurons u_i, u_j are connected to each other if there exists a weight w_{ij} such that the i-th input of u_j is obtained from the output of

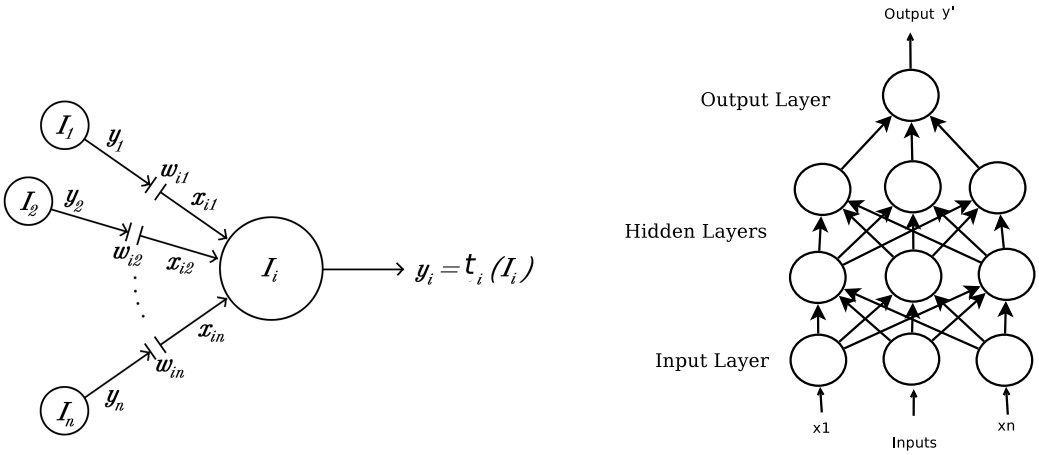

Figure 1. Graphical representation of a neuron (left) and of the network (right).

u_j weighted by w_{ij}: $x_{ij} = w_{ij}y_i$. In this way u_j receives an input which can be inhibited, enhanced or damped, with respect to the output of u_i, according to the sign and value of w_{ij}. Assuming that neuron u_j receives inputs from other neurons $u_1, \ldots, u_m$, its own output will be computed as

$$y_j = t(I_j) = t\left(\sum_{i=1}^{m} x_{ij}\right) = t\left(\sum_{i=1}^{m} w_{ij}y_i\right), \tag{2}$$

as it is shown in the left part of Figure 1. Then, the output of each neuron depends on the weights, which can be adjusted to allow the network to predict different functions. Neurons are usually stored in layers and can be connected to the others in many ways, so that different kinds of ANN can be obtained. The connections between neurons are called synapses and they store the weights. When the ANN is used for regression purposes to approximate a function f, it is assumed to have a training set at disposal given by

$$\mathcal{T} = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_{m_{train}}, y_{m_{train}}) |\ y_i = f(\mathbf{x}_i)\ i = 1, \ldots, m_{train}\}. \tag{3}$$

During the training phase the samples in the training set are given as an input to the ANN, the weights of the connections are adjusted by an iterative process to fit the data in the training set, and the network builds its own model function $\bar{f}$ approximating the desired function f: $\bar{f}(\mathbf{x}) \simeq f(\mathbf{x})$ for all inputs $\mathbf{x}$, which will be used in the execution phase to predict the outputs of new samples.

When the aim is to predict values of a scalar function usually a feed-forward ANN (i.e. without loops in the connections) trained with a back propagation algorithm is used.

In this article a network structured on four levels is employed, see right part of Figure 1: an input layer which is composed of as many neurons as the number of degrees of freedom n, two hidden layers and an output layer composed of a single neuron. The neurons in each level are connected to all the neurons in the upper level.

The *back propagation algorithm* is a supervised learning algorithm, which means that it is necessary to provide to the ANN examples of both inputs and outputs the network has to compute.

During the training the samples in the training set (3) are provided as an input to the network one by one. Each neuron in the input layer receives in input a component of the considered training sample $\mathbf{x}$. All the components are then transmitted through the network and transformed by the activation functions of the neurons they come across,

until they reach the neuron in the output layer, whose output y' is the network output and that represents the current approximation to the desired $y = f(\mathbf{x})$. Notice that the prediction, and so also the prediction error, depends on the weights (see equation (2)), so it is possible to adjust them to minimize it. Then, the weights are initialized randomly and updated if y' is different from the expected result y, to minimize the error $E = \|y - y'\|^2$. Specifically, the current weights of all the connections are stacked together forming a vector w which is updated as

$$w = w - \alpha \nabla E, \tag{4}$$

where $\alpha \in (0, 1]$ is a fixed value called learning rate. All the samples in the training set are given in input to the network, this is equivalent to performing a single step of gradient method for each training couple $(\mathbf{x}_i, y_i)$, $i = 1, \ldots, m_{train}$. To obtain values of the weight to get a good approximation to function f, the whole process is repeated for a high number of times (called epochs, $\simeq 10^5/10^6$). The final values of the weights, those obtained at the end of the last epoch, are then used in the execution phase to compute the output of new samples. To predict the function value of a new sample $\mathbf{x}$, it is provided as an input to the network, whose output is the desired approximation of $f(\mathbf{x})$.

2.2 *SVM: Support Vector Machines*

In the approach presented in this article Support Vector Machines will be used to solve a binary classification problem, namely the classification of geometries as feasible or unfeasible. Then, here a brief introduction on SVM for binary classification problems is given, for a more detailed description see (23).

Assume to have samples belonging to two different classes $\mathcal{F}$ and $\mathcal{U}$, the goal is to predict which class a new data point will be in. The training set in this case is given by

$$\mathcal{T} = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_{m_{train}}, y_{m_{train}}),\ y_i = +1 \text{ if } \mathbf{x}_i \in \mathcal{F},\ y_i = -1 \text{ if } \mathbf{x}_i \in \mathcal{U},\ i = 1, \ldots, m_{train}\},$$

i.e. y_i will be the label of the class the feature is in.

Starting from the samples in the training set, the meta-model builds a decision function that is used to assign a label to new samples. Particularly, a hyperplane is sought that separates the features belonging to different classes. If the samples are not linearly separable, they are projected in a higher dimensional space by a *kernel function* $\phi(\cdot)$ and the separating hyperplane is searched in the projected space. The hyperplane H is the set of points such that:

$$H = \{\mathbf{x} \in \mathrm{R}^n \mid h(\mathbf{x}) = w^T \Phi(\mathbf{x}) + b = 0\},$$

its equation depends on two parameters, $w \in \mathrm{R}^n$ and $b \in \mathrm{R}$. Features $\mathbf{x}$ laying on the hyperplane are such that $h(\mathbf{x}) = 0$, the others are such that either $h(\mathbf{x}) \geq 1$ or $h(\mathbf{x}) \leq -1$, choosing a suitable scaling for the free coefficients. If the features are linearly separable in the projected space

$$h(\mathbf{x}_i) = w^T \Phi(\mathbf{x}_i) + b \geq 1 \text{ for all } \mathbf{x}_i \in \mathcal{F},$$
$$h(\mathbf{x}_i) = w^T \Phi(\mathbf{x}_i) + b \leq -1 \text{ for all } \mathbf{x}_i \in \mathcal{U},$$

so that features can be assigned to one of the two classes according to the sign of function h. If it exists, the separating hyperplane is not unique. For each H the margin ρ is defined

as the minimum distance among the features and the hyperplane itself:

$$\rho(w,b) = \min_{\mathbf{x}} \frac{|w^T\mathbf{x} + b|}{\|w\|}.$$

The optimal hyperplane is defined as the one that maximizes the margin and it is found solving the following optimization problem:

$$\max_{w\in\mathrm{R}^n, b\in\mathrm{R}} \rho(w,b). \tag{5}$$

It is possible to prove (23) that the optimal hyperplane exists and is unique, and that (5) is equivalent to

$$\min_{w\in\mathrm{R}^n, b\in\mathrm{R}} \frac{1}{2}\|w\|^2 \tag{6a}$$

$$\text{s.t. } w^T\Phi(\mathbf{x}_i) + b \geq 1, \text{ for all } \mathbf{x}_i \in \mathcal{F}, \tag{6b}$$

$$w^T\Phi(\mathbf{x}_i) + b \leq -1, \text{ for all } \mathbf{x}_i \in \mathcal{U}. \tag{6c}$$

If the features are not linearly separable (6) has no solution. In the applications features are rarely linearly separable, even in the higher dimensional space, so it is necessary to allow the presence of some outliers inserting some slack variables ζ_i $i = 1, \ldots, m_{train}$ in the model, such that

$$\begin{aligned} w^T\Phi(\mathbf{x}_i) + b &\geq 1 - \zeta_i \text{ for all } \mathbf{x}_i \in \mathcal{F}, \\ w^T\Phi(\mathbf{x}_i) + b &\leq -1 + \zeta_i \text{ for all } \mathbf{x}_i \in \mathcal{U}, \\ \zeta_i &\geq 0, i = 1, \ldots, m_{train}. \end{aligned}$$

Notice that if $\mathbf{x}_i$ is incorrectly classified $\zeta_i > 1$, so $\sum_{i=1}^{m_{train}} \zeta_i$ is an upper bound of the number of training features misinterpreted. The term $C \sum_{i=1}^{m_{train}} \zeta_i$ is inserted in the objective function (6), where $C > 0$ weights the contribution of the new term. To obtain the optimal parameters w, b, given $C > 0$, this minimization problem has to be solved:

$$\min_{\omega,b,\zeta} \frac{1}{2}\|w\|^2 + C \sum_{i=1}^{m_{train}} \zeta_i \tag{7a}$$

$$\text{s.t. } y_i(w^T\Phi(\mathbf{x}_i) + b) \leq 1 - \zeta_i, \tag{7b}$$

$$\zeta_i \geq 0, \ i = 1, \ldots, m_{train}. \tag{7c}$$

Problem (7), is not actually solved, but rather the dual problem is considered:

$$\max_{w,b,\zeta,\lambda,\mu} \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{m_{train}} \zeta_i - \sum_{i=1}^{m_{train}} \lambda_i(y_i(w^T\Phi(\mathbf{x}_i)+b)-1+\zeta_i) - \sum_{i=1}^{m_{train}} \mu_i\zeta_i, \tag{8a}$$

$$\text{s.t. } w = \sum_{i=1}^{m_{train}} \lambda_i y_i \Phi(\mathbf{x}_i), \tag{8b}$$

$$\sum_{i=1}^{m_{train}} \lambda_i y_i = 0, \tag{8c}$$

$$C - \lambda_i - \mu_i = 0, i = 1, \ldots, m_{train}, \tag{8d}$$

$$\lambda, \mu \geq 0, \tag{8e}$$

where $\lambda = [\lambda_1, \ldots, \lambda_{m_{train}}]$, $\mu = [\mu_1, \ldots, \mu_{m_{train}}]$ and (8e) holds componentwise. (8) can be rewritten as

$$\min_{\lambda} \frac{1}{2}\lambda^T Q\lambda - e^T\lambda \tag{9a}$$

$$\text{s.t. } y^T\lambda = 0, \tag{9b}$$

$$0 \leq \lambda_i \leq C,\ i = 1, \ldots, m_{train}, \tag{9c}$$

where $e = [1, \ldots, 1]^T$, $y = [y_1, \ldots, y_{m_{train}}]$,

$$Q_{i,j} = y_i y_j K(\mathbf{x}_i, \mathbf{x}_j) \equiv y_i y_j \Phi(\mathbf{x}_i)^T \Phi(\mathbf{x}_j). \tag{10}$$

After solving (9), w can be obtained by (8b) and b by the complementarity conditions

$$\lambda_i(y_i(w^T\Phi(\mathbf{x}_i)+b)-1+\zeta_i) = 0,\ \ i = 1, \ldots, m_{train}.$$

The decision function is then defined as:

$$f(\mathbf{x}) = \text{sgn}(w^T\Phi(\mathbf{x}) + b).$$

3. Industrial application: parametric design of centrifugal pumps

In this section the industrial application considered in this article is presented, i.e. the parametric design of a whole family of turbomachinery components. The steps of the design process proposed in (9) are described and the arising drawbacks are pointed out, showing the results of the application of the considered procedure to a test case. Finally the use of a classification process is proposed to solve the presented issues.

The procedure proposed in (9) is outlined in Framework 3.1. It is based on coupling a geometry parameterization tool, CFD computations for solving Reynolds Averaged Navier-Stokes (RANS) equations and feed-forward Artificial Neural Networks as a regression meta-model. It is assumed that the machine performance is evaluated through h scalar performance functions: $f_1, \ldots, f_h$ and $\mathbf{f} = [f_1, \ldots, f_h]$. The procedure is composed of two parts: Phase 1 of ANN training and Phase 2 of research of an optimal solutions set. In Phase 1, h ANN models, one for each performance function, are trained that will be used in Phase 2 to build the response surface. In Phase 2 indeed, the meta-models are used to predict performance functions of new geometries, to reduce the amount of

CFD computations required for the design. On the response surface a multi-objective algorithm is then run to find the set of optimal geometries.

Framework 3.1

Parametric design of a family of turbomachinery components, coupling CFD and ANN.

Phase 1: ANN training

(1) **Geometry parameterization**. Choose n parameters (degrees of freedom) to describe the machine geometry, so that each machine will be identified by a vector $\mathbf{x} = [x_1, \ldots, x_n] \in \mathrm{R}^n$, which in the following we will address as *feature* or *sample*.

(2) **Sampling of the design space**. Taking into account the range of variation of each parameter, the design space is built. Assuming that $x_i^{\min} \leq x_i \leq x_i^{\max}$ for $i = 1, \ldots, n$, the resulting design space is defined as:

$$\mathcal{S} = [x_1^{\min}, x_1^{\max}] \times \cdots \times [x_n^{\min}, x_n^{\max}] \subseteq \mathrm{R}^n.$$

A quasi-random sequence (Sobol's or latin hypercube) is used to generate a dataset $\mathcal{D}_0 = \{\mathbf{x}_1, \ldots, \mathbf{x}_m\}$ that samples the design space.

(3) **CFD simulations**. CFD computations are performed on $\mathcal{D}_0$ to divide the features in the sets $\mathcal{F}'$ of feasible samples and $\mathcal{U}'$ of unfeasible samples, where usually $|\mathcal{F}'| \ll |\mathcal{U}'|$. The machine performance functions of feasible samples are evaluated: $\mathbf{f}_j = [f_1(\mathbf{x}_j), \ldots, f_h(\mathbf{x}_j)]^T$ for $\mathbf{x}_j \in \mathcal{F}'$ and a performance database $\mathcal{D}_{\mathcal{F}'}$ is built, which can be thought of as a set of pairs: $\mathcal{D}_{\mathcal{F}'} = \{(\mathbf{x}_j, \mathbf{f}_j),\ \ \mathbf{x}_j \in \mathcal{F}'\}$.

(4) **ANN training**. The performance database is used to train the ANN models, that learning from the examples in $\mathcal{D}_{\mathcal{F}'}$, build their own functions $\bar{f}_i$, that are approximations to the true performance functions: $\bar{f}_i \simeq f_i$, $i = 1, \ldots, h$.

Phase 2: Research of an optimal solutions set

(1) **Sampling of the design space.** The design space is sampled again producing a new dataset $\mathcal{D}_1$.

(2) **ANN execution**. ANN models are used to predict function $\mathbf{f} = [f_1, \ldots, f_h]$ on all the new samples in $\mathcal{D}_1$ through function $\bar{\mathbf{f}} = [\bar{f}_1, \ldots, \bar{f}_h]$ built at step (4) of Phase 1, thus producing the response surface.

(3) **Multi-objective algorithm**. A multi-objective algorithm is run to find the set of optimal solutions $\mathcal{D}_{ott}$.

(4) **CFD validation of the optimal solutions set**. The found solutions set $\mathcal{D}_{ott}$ is validated by CFD computations to discard the unfeasible samples, arising from the sampling at step (1).

The procedure is sketched in the flowchart in Figure 2. In this and in all the other flowcharts a rectangle represents a process, a parallelogram an input/output, a diamond indicate a decision, and specifically the result of a classification.

Specifically, in this article the parametric design of the components of a whole family of

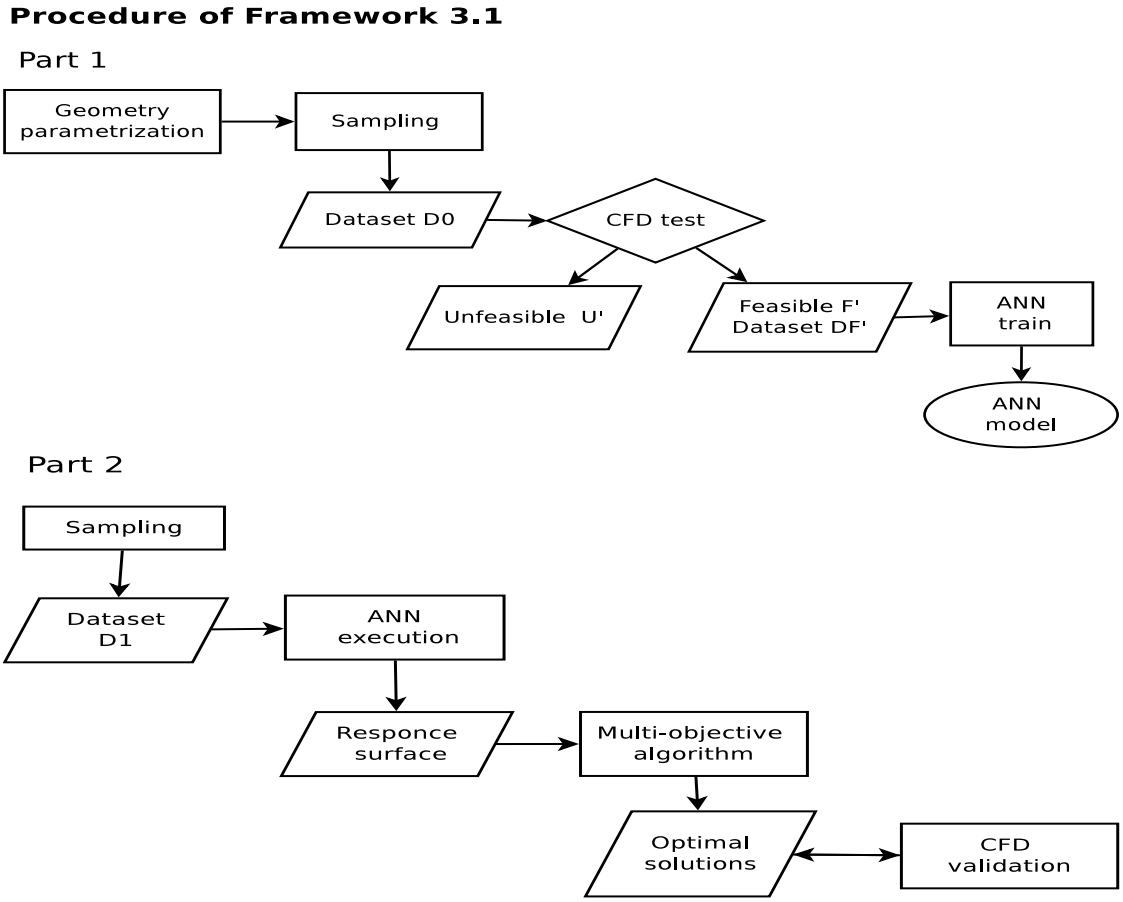

Figure 2. Flowchart of the approach outlined in Framework 3.1.

pumps with horizontal suction duct, single-shaft centrifugal impeller, as the one depicted in the left part of Figure 3, vertical discharge diffuser and volute, in a wide range of specific speed, is considered. Then, some choices made to customize the general approach outlined in Framework 3.1 to the specific application are pointed out here. The geometry parameterization is made trough an in-house tool specifically developed for the type of machines introduced above. It relies on a reduced set of parameters (integral B-splines control points) which have a strong correlation with the pump performance, rather than with the geometrical shape only, and allows to handle the three-dimensional pump geometry, that is the impeller, the diffuser and the volute, in a parametric way. It is essential to parameterize all the components using as few geometrical parameters as possible, in order to reduce the number of degrees of freedom involved and the dimensionality of the resulting design space. Anyway, tens of parameters are usually necessary. More details about the geometry parameterization can be found in (9). The CFD analysis of impeller performance relies on a fully viscous three-dimensional numerical solver (TRAF (1), (2), (3)), while the overall performance prediction of the pump (efficiency, mass flow rate, hydraulic head etc.) was obtained coupling the results of the impeller analysis (or the equivalent meta-model prediction) with a 1D correlation tool, which accounts for the losses due to the other components, in order to reduce CFD costs, (8). The CPU time for one serial CFD computation on a Intel Xeon E5-2680V2 @ 2.8 GHz CPU is about 2 hours. The impeller computational domain was discretized with a structured elliptic single-block H type grid, $225 \times 65 \times 65$ points in the streamwise, pitchwise, spanwise direction respectively, for a total number of about a million of points, that is reported in the right part of Figure 3.

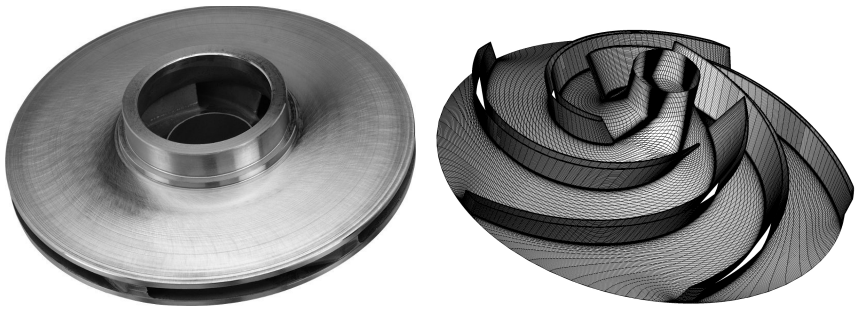

Figure 3. Left: Single-shaft centrifugal impeller. Right: 3D view of impeller H type grid.

There are mainly two drawbacks arising in the outlined procedure, that we present in the following Framework.

Framework 3.2

Drawbacks of the procedure outlined in Framework 3.1

1. The first drawback arises at step (2)-(3) of Phase 1. Even if particular attention is dedicated to the implementation of the parameterization tool and to correlate the tens of degrees of freedom, when sampling the design space generally more than 70% of the samples turn out to be unfeasible. This is an intrinsic aspect in the nature of a parametric design, in which the same geometrical parameters and range of variation are applied to pump geometries with very different characteristics (wide range of specific speed) and manufacturing constraints.

   To obtain an accurate prediction of the desired functions, the meta-model has to be trained on a set $\mathcal{F}'$ of just feasible samples, which can be found at the cost of a really large number of CFD computations, as the most part of them will yield unfeasible geometries.

2. Another analogous problem arises in Phase 2, when the so trained meta-model is used to find an optimal solutions set. The objective functions of new geometries randomly selected are predicted by means of the meta-model. The sampling is performed within the whole design space, and again the most part of the geometries provided as meta-model input will result unfeasible. The predicted performance functions values will then be meaningless and it is possible that step (3) of Phase 2 yields a set of optimal solutions consisting of just unfeasible samples, leading to the need of repeating the procedure again. This outcome is independent from the meta-model chosen, and leads to conclude that the considered approach may not be effective for a parametric design.

3.1 *Example*

In this section a test case arising from the industrial application introduced above is considered to highlight the issues just described. The pump components were parameterized using $n = 40$ degrees of freedom and it was found experimentally that the ratio between feasible and unfeasible feature is 1:3.

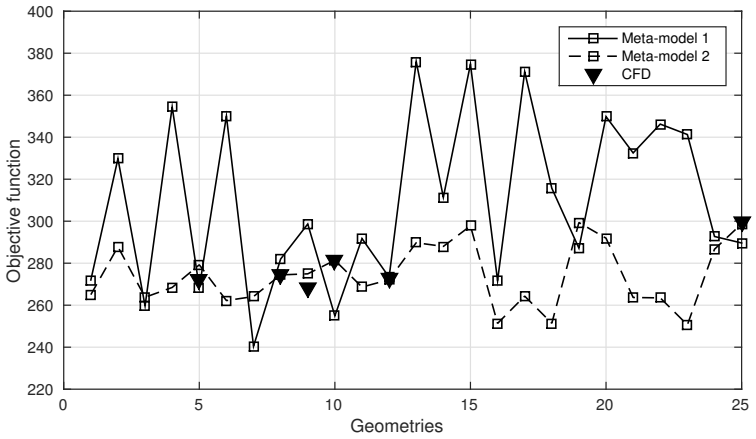

Figure 4. Meta-model 1 (straight line, SVM in regression mode) and meta-model 2 (dashed line, ANN) predictions of an objective function, CFD computations for feasible geometries (full triangle).

The procedure sketched in Framework 3.1 was performed on this database. At step (3) of Phase 1 about 50000 CFD computations were necessary to form a training set of just 12500 feasible samples, which proved to be large enough to obtain good regression results. This means that about 75% of performed expensive CFD computations were useless, as the considered features results unfeasible and therefore cannot be used in the following steps.

Two different meta-models were trained on this training set and then used to predict one of the pump objective functions of new samples. The results obtained on 25 of these features are reported in Figure 4. Here, the empty squares mark the output predicted by the first meta-model (straight line) and by the second one (dashed line), while the objective function values for feasible features, obtained by CFD calculations, are marked by a full triangle. Notice that only few samples are feasible (6/25), and so marked by triangles. Noticeably, the regression function values computed by the two models for many of the unfeasible geometries are completely different from each other, as they are meaningless. From this, it is possible to conclude that the drawbacks in Framework 3.2 are intrinsic in the problem and cannot be solved choosing a different meta-model.

3.2 *Need for a classification tool*

From all the considerations made above, it is possible to conclude that to make this approach practical, it is necessary to have a tool to distinguish the feasible geometries from the others without running CFD. This would help both to avoid a useless large amount of expensive CFD calculations to form ANN training set and to restrict the regression process to a set mainly composed of feasible geometries, to reduce the presence of unfeasible ones in the optimal solution set, making also step (4) of Phase 2 less expensive.

What has to be considered is then a binary classification problem: samples are assumed to belong to two different classes and the goal is to predict which class a new data point will be in. In our case a sample is a geometry and the classes are that of feasible geometries (the ones corresponding to manufacturable machines and to convergent CFD calculations) and that of the unfeasible geometries (all the others).

For this reason a hybrid approach has been conceived, that includes a classification tool in the previously proposed procedure. It is described in the following section.

4. Hybrid approach: SVM as a classification meta-model

The main idea of the approach devised to overcome the drawbacks in Framework 3.2, is to use a classifier to distinguish the feasible from the unfeasible features. As pointed out in Section 3.2 indeed, each one of the drawbacks outlined in Framework 3.2 could be lightened having at disposal a cheap tool to separate the features into these two classes. Then, two classification procedures, one for each drawback, were inserted in the approach outlined in Framework 3.1. The tool chosen as a classifier is Support Vector Machine. The resulting hybrid approach is outlined in Framework 4.1.

The basic scheme is the same as the one of the procedure sketched in Framework 3.1, except for the added classifications which are highlighted by italic font. The approach is still divided in Phase 1 of ANN training and Phase 2 of research of an optimal solutions set. Some steps are the same as in Framework 3.1, but their description is repeated for sake of clarity. It is assumed to have at disposal a trained SVM model which, given a sample $\mathbf{x}$ as an input, classifies it as feasible or unfeasible, i.e. it divides the features in the two classes $\mathcal{F}$ of feasible samples and $\mathcal{U}$ of unfeasible samples. As in Section 3, it is assumed that the machine performance is evaluated through h scalar performance functions: $f_1, \ldots, f_h$ and $\mathbf{f} = [f_1, \ldots, f_h]$.

Framework 4.1

Hybrid approach: Parametric design of a family of turbomachinery components coupling CFD, ANN, SVM.

Phase 1: ANN training

(1) **Geometry parameterization**. Choose n parameters (degrees of freedom) to describe the machine geometry, so that each machine will be identified by a vector $\mathbf{x} = [x_1, \ldots, x_n] \in \mathrm{R}^n$, which in the following we will address as *feature* or *sample*.

(2) **Sampling of the design space**. Taking into account the range of variation of each parameter, the design space is built. Assuming that $x_i^{\min} \leq x_i \leq x_i^{\max}$ for $i = 1, \ldots, n$, the resulting design space is defined as:

$$\mathcal{S} = [x_1^{\min}, x_1^{\max}] \times \cdots \times [x_n^{\min}, x_n^{\max}] \subseteq \mathrm{R}^n.$$

A quasi-random sequence (Sobol's or latin hypercube) is used to generate a dataset $\mathcal{D}_0 = \{\mathbf{x}_1, \ldots, \mathbf{x}_m\}$ that samples the design space.

(3) ***Classification by SVM***. *The samples in $\mathcal{D}_0$ are given in input to SVM which divides them into the two classes $\mathcal{F}$ (feasible), $\mathcal{U}$ (unfeasible). Features in $\mathcal{U}$ are not taken in further account and just those in $\mathcal{F}$ are considered in the next steps.*

(4) **CFD simulations**. CFD computations are performed on the samples in $\mathcal{F}$. The outliers are eliminated obtaining a subset $\mathcal{F}'$ of just feasible features for which the machine performance functions are evaluated: $\mathbf{f}(\mathbf{x}_j) = [f_1(\mathbf{x}_j), \ldots, f_h(\mathbf{x}_j)]$ and $\mathbf{x}_j \in \mathcal{F}'$. The performance database $\mathcal{D}_{\mathcal{F}'}$ is built, which is a set of pairs: $\mathcal{D}_{\mathcal{F}'} = \{(\mathbf{x}_j, \mathbf{f}_j), \mathbf{x}_j \in \mathcal{F}'\}$, where $\mathbf{f}_j = [f_1(\mathbf{x}_j), \ldots, f_h(\mathbf{x}_j)]$.

(5) **ANN training**. The performance database is used to train the ANN models, that learning from the examples in $\mathcal{D}_{\mathcal{F}'}$, build their own functions $\bar{f}_i$ that approximate the true performance functions: $\bar{f}_i \simeq f_i$, $i = 1, \ldots, h$.

Phase 2: Research of an optimal solutions set

(1) **Sampling of the design space.** The design space is sampled again producing a new dataset $\mathcal{D}_1$.

(2) ***Classification by SVM***. *Samples in $\mathcal{D}_1$ are given in input to SVM which divides them into the two classes $\mathcal{F}''$,$\mathcal{U}''$. Features in $\mathcal{U}''$ are not taken in further account and just those in $\mathcal{F}''$ are considered in the next step.*

(3) **ANN execution**. The ANN model is used to predict function **f** of the samples in $\mathcal{F}''$, thus producing the response surface.

(4) **Multi-objective algorithm**. A multi-objective algorithm is run to find the set of optimal solutions $\mathcal{D}_{ott}$.

(5) **CFD validation**. The optimal solutions set $\mathcal{D}_{ott}$ found is validated trough CFD computations to eliminate possible outliers, as the classification by SVM will not be 100% correct.

The classification procedures inserted at step (3) of Phase 1 and step (2) of Phase 2 are intended to mitigate drawbacks 1. and 2. in Framework 3.2 respectively.

Indeed, the first classification allows to restrict the CFD computations performed at step (4) of Phase 1 just to the set $\mathcal{F}$ of features classified as feasible by SVM, with the aim of eliminating outliers. This produces a great saving in CFD computations, as $|\mathcal{F}| \ll |\mathcal{D}_0|$ and CFD performed on samples in $\mathcal{U}$ would be of no use.

The second process allows to predict function values just of samples in $\mathcal{F}''$ that is mainly composed of feasible features. Some outliers will anyway be still present in the set, then step (5) of Phase 2 is still necessary, but it will be much less expensive than the corresponding step (4) of Phase 2 in Framework 3.1.

The approach effectiveness and the benefits of the two classification procedures will be discussed in more details in Section 6. The procedure in Framework 4.1 is sketched in the flowchart reported in Figure 5, where the two inserted classification procedures are highlighted by black boxes. It can be compared to the previous procedure, whose flowchart is reported in Figure 2. For seek of simplicity in the flowchart it is assumed to have at disposal a trained SVM.

It is worth reminding that the proposed approach, compared to the previous one, requires the additional SVM training, which however has to be done just once and is not really time consuming (in the tests performed in Section 6 the training time is about 5 minutes). The main cost related to this arises from the CFD computations necessary to form a training set of both feasible and unfeasible features of known classification. However, this is not a useless expense. On one hand, the feasible features found could be used to enlarge the ANN training set obtained at step (4) of Phase 1, to gain a more accurate regression meta-model. On the other hand the gains provided by the added classification procedures largely cover the costs of SVM training.

In Section 6 the proposed hybrid procedure will be applied to practical test cases arising from the described industrial application, highlighting the savings provided.

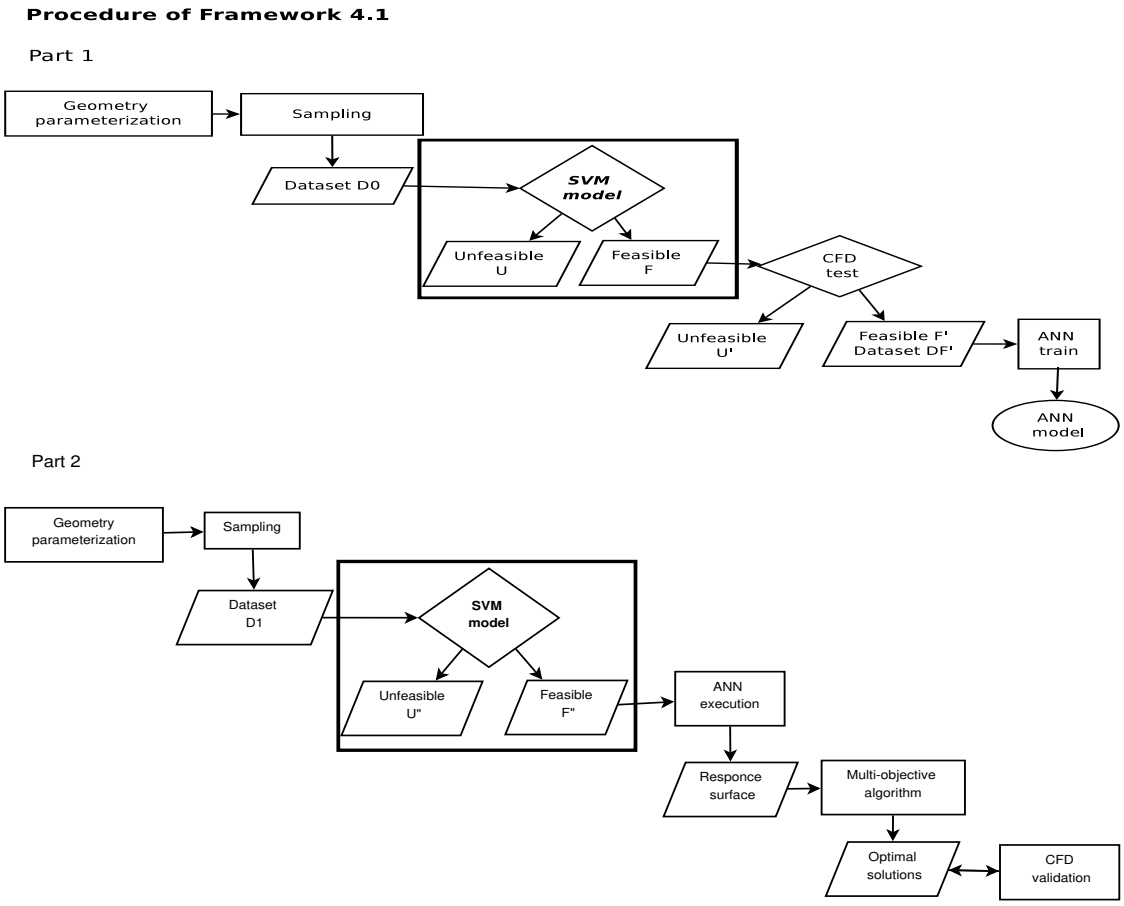

Figure 5. Flowchart of the proposed hybrid approach, outlined in Framework 4.1.

5. Consequences of dataset unbalancedness

It is important to underline that the savings in CFD computations that can be achieved by the proposed approach compared to the previously used one, depend entirely on SVM classification ability. Then it is crucial to properly train the meta-model to gain the best results. The main difficulty, intrinsic in the classification process, is that the sets of data the SVM has to be trained on and that have to be classified, are unbalanced, due to the predominant presence of unfeasible geometries over the feasible ones. In this section it is shown how this feature affects the choice of the training strategy and the criteria chosen to evaluate the procedure performance.

With unbalanced data sets, the classifier has few information about the minority class to make an accurate prediction, so it is easy to have many feasible features misclassified.

Assume to classify m unknown features in the two classes $\mathcal{F}$, labelled with $+1$, and $\mathcal{U}$, labelled with -1. Let $\mathcal{C} \in \mathrm{R}^m$ be the vector with the correct features classification, i.e. $\mathcal{C}(i) = 1$ if the i-th feature $\mathbf{x}_i \in \mathcal{F}$ and $\mathcal{C}(i) = -1$ if $\mathbf{x}_i \in \mathcal{U}$, $i = 1, \ldots, m$, and $P\mathcal{C} \in \mathrm{R}^m$ the result of the classification process, i.e. $P\mathcal{C}(i)$ is the predicted value of $\mathcal{C}(i)$, $i = 1, \ldots, m$.

For each feature four different situations can occur, that are illustrated in the confusion matrix in Table 1:

- TP=true positive: $\mathcal{C}(i) = 1, P\mathcal{C}(i) = 1$ the feature is feasible and is correctly classified,
- FP=false positive: $\mathcal{C}(i) = -1, P\mathcal{C}(i) = 1$ the feature is unfeasible but is misinterpreted and classified as feasible,
- TN=true negative: $\mathcal{C}(i) = -1, P\mathcal{C}(i) = -1$ the feature is unfeasible and is correctly

classified,
- FN= false negative: $\mathcal{C}(i) = 1, P\mathcal{C}(i) = -1$ the feature is feasible but is misinterpreted and classified as unfeasible.

	$\mathcal{C}(i) = -1$	$\mathcal{C}(i) = 1$
$P\mathcal{C}(i) = 1$	FP	TP
$P\mathcal{C}(i) = -1$	TN	FN

Table 1. Confusion matrix.

When the dataset is balanced, the probability of misinterpreting a feature is the same for the two classes, i.e. the probability of a false positive is the same of that of a false negative, then the fraction of features correctly classified $\frac{TP+TN}{TN+TP+FN+FP}$ is a good measure of accuracy. When unbalancedness occurs and the fraction of unfeasible features is predominant, the probability of a false negative is much higher than the probability of a false positive, and this value is not so meaningful. As a matter of fact, as the feasible samples represent a small percentage of the full database, one has $\frac{TP+TN}{TN+TP+FN+FP} \simeq \frac{TN}{TN+FN}$ and it is so possible to achieve a high accuracy level even if all the features in the minority class are misinterpreted.

For this reason a proper choice of the parameters to evaluate the performance of the classification process has to be made taking into account that the datasets are really unbalanced.

In this article two sets of parameters are considered to evaluate the performance, (18). The first set of parameters is given by those of the confusion matrix in percentage form:

- $TPR = \frac{TP}{TP+FN}$ *True Positive Rate* or sensitivity or recall, fraction of positive samples correctly classified over all positive samples available in the test,
- $FNR = \frac{FN}{TP+FN}$ *False Negative Rate* fraction of feasible features misinterpreted over all positive samples available in the test,
- $TNR = \frac{TN}{TN+FP}$ *True Negative Rate* or specificity, fraction of negative samples correctly classified over all negative samples available in the test,
- $FPR = \frac{FP}{TN+FP}$ *False Positive Rate*, fraction of unfeasible features misinterpreted over all negative samples available in the test.

The second set of parameters that is of interest for a designer is:

- $PPV = \frac{TP}{TP+FP}$ *Positive Predictive Value*, the fraction of true positives in the set of features classified as positive,
- $FDR = \frac{FP}{TP+FP}$ *False Discovery Rate*, the fraction of false positives in the set of features classified as positive,
- $NPV = \frac{TN}{TN+FN}$ *Negative Predictive Value*, the fraction of true negatives in the set of features classified as negative,
- $FOR = \frac{FN}{TN+FN}$ *False Omission Rate*, the fraction of false negatives in the set of features classified as negative.

The meaning of these parameters is shown in Figure 6. When features are classified by SVM they are divided into two subsets $\mathcal{F}$, features classified as positive, and $\mathcal{U}$, features classified as negative. Parameter PPV interests the designer as gives a measure of the quality of set $\mathcal{F}$, telling how many features are actually feasible, meaning that $FDR = 1 - PPV$ indicates how many useless CFD computations will be performed at step (4) of Phase 1 and how many outliers could be part of the optimal solutions set at steps (4)-(5) of Phase 2 of Framework 4.1. On the other hand, another parameter to take into account is FOR, which gives the percentage of feasible features in the set of features classified as unfeasible, which are then wrongly discarded after the classification process.

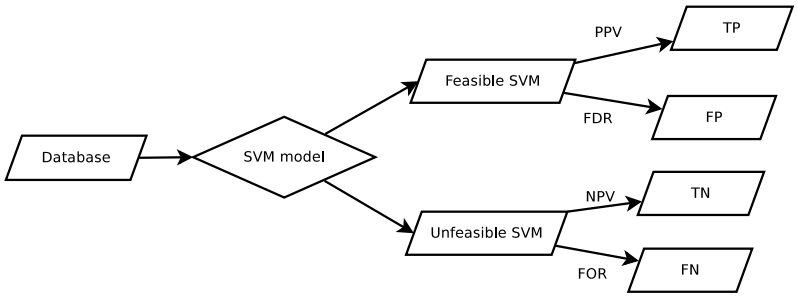

Figure 6. Meaning of parameters PPV, FDR, NPV, FOR.

Notice that the couples (TPR, FNR), (TNR, FPR), (PPV, FDR), (NPV, FOR) sum up to one, so in the tables in Section 6 just a parameter for each couple will be shown.

The dataset unbalancedness has to be taken into account also in meta-model training. The designer is mainly interested in detecting feasible samples, so it is necessary to force the classifier to take features belonging to the different classes into different consideration. Generally the lack of balancedness of the dataset is handled using two different weights for the positive and the negative features, to penalize with more severity the misinterpretation of feasible features, (25). Coefficient C in the objective function (7) of the problem to be solved during SVM training is then split in C_+ (coefficient for feasible features) and C_- (coefficient for unfeasible features), so that the objective function becomes:

$$\min_{\omega,b,\zeta} \frac{1}{2}\|w\|^2 + C_+ \sum_{\mathbf{x}_i \in \mathcal{F}} \zeta_i + C_- \sum_{\mathbf{x}_i \in \mathcal{U}} \zeta_i. \tag{11}$$

To gain good results it is necessary to properly tune these coefficients, with the aim of finding the better compromise between the need of detecting as many feasible features as possible and that of avoiding too many unfeasible features being classified as feasible ones, that would lead to useless CFD computations in the regression process. In the literature, (25), it is suggested that the ratio of the coefficients corresponding to feasible and unfeasible features should be inversely proportional to the ratio of the corresponding features set sizes:

$$\frac{C_+}{C_-} \sim \frac{|\mathcal{U}|}{|\mathcal{F}|}, \tag{12}$$

where $|\cdot|$ represents the cardinality of the set.

In the next section it will be shown how all these considerations are taken into account in practical parametric design tests.

6. Numerical results

In this section the results of the tests performed on three different databases, with different number of degrees of freedom and constraints, arising from the described industrial application are reported. It is worth remembering that the benefit granted by the proposed procedure compared to the one sketched in Framework 3.1 lays in the savings arising from the use of a classification meta-model, which makes the procedure practical. Moreover, the savings depend entirely on the classification process quality. Then in this section the focus is on the tuning of SVM free parameters to gain the best performance

and on the results obtained by the related classification process. The other steps of the procedure are not shown, as this is out of the scope of the paper.

The SVM used to perform the tests is the one implemented by Chih-Chung Chang and Chih-Jen Lin in LIBSVM - A Library for Support Vector Machines, (7). The tests were performed calling the LIBSVM package through its MEX interface, on a Intel Core(TM) i7-4510U 2.6 GHz, 8 GB RAM.

6.1 *Experimental setting*

In this section some details on the experimental setting and the data preparation are given.

For the tests the features are divided into two classes: class $\mathcal{F}$ of feasible features, corresponding to constructable machines and convergent CFD calculations, and class $\mathcal{U}$ of unfeasible ones.

CFD calculations were carried out using the TRAF code (1), a steady/unsteady, multi-grid/multiblock flow solver for the three-dimensional Reynolds-averaged Navier-Stokes equations. A detailed description of the numerical scheme can be found in (1; 2). The domain is divided in N cells. Denoted with $R(n)$ the residuals vector of the discretized equations on the n-th cell, for $n = 1, \ldots, N$, and with

$$R = \frac{1}{N} \sum_{n=1}^{N} \|R_i(n)\|, \tag{13}$$

the 2-norm of the residuals averaged on the total number of cells, the computations are considered convergent if

$$\log_{10} R \leq res_{cut} \tag{14}$$

where res_{cut} is a suitable threshold fixed by the user. In the tests presented below, it was set equal to one decade under the single precision machine zero, i.e. $res_{cut} = -8$. If $res(i)$ is the residual of the i-th feature,

- $\mathcal{F} = \{x_i | res(i) \leq res_{cut}\}$,
- $\mathcal{U} = \{x_i | res(i) > res_{cut}\}$,

assuming $res(i) = 0$ for non-convergent computations or non-manufacturable machines.

The data were prepared for the classification process, scaling the geometries degrees of freedom to have mean 0 and variance 1. In the literature indeed, it is known that a good scaling of both the training and the testing features is necessary to have good results, otherwise the contribution of the features that have a bigger values of some degree of freedom would be predominant over the contributions of the other features.

A choice that deeply influence the performance of SVM learning algorithm, is that of the kernel function (10). In LIBSVM many different choices are possible: linear, polynomial, radial basis function, sigmoid. For this specific application the best performance was obtained with the radial basis function kernel:

$$K(x, y) = e^{-\gamma \|x-y\|^2},$$

where x, y are features and γ is a parameter to be set. A good choice of free parameter γ is crucial. LIBSVM provides a tool to select it by a cross validation on a set of values. However when the dataset is large, cross validation can be a really time consuming

process, so γ was determined using the default method employed in (28), (16), that is, it was set to the average squared distance among training patterns.

For the tests the available data were divided into two subsets, a training set of $m_{train} = 30000$ geometries and a validation set of $m_{val} = 50000$ geometries, both composed of features of known classification. Notice that, even if the number of degrees of freedom is really high and a large number of points should be necessary to accurately sample the design space, the chosen value of training features turned out to be sufficient to obtain good classification results. Increasing it, leads to higher training times without significant improvement in performance.

6.2 *Tests*

For the numerical experimentation three different databases arising from the industrial application introduced before are considered, which have respectively n=40, 44, 42 degrees of freedom, and for which the ratio between unfeasible and feasible features $\frac{|\mathcal{U}|}{|\mathcal{F}|}$ is 3:1, 7:1, 6:1.

The results of the tests are shown in Tables 2, 3 and 4, in which the parameters described in Section 5 are reported, for many classification procedures performed with different choices of coefficient C_+. Indeed, test were conducted to investigate on the choice of the best combination of parameters. Then, parameter $C_- = 1$ was fixed, and different values of C_+ were tested.

Looking at the tables it is possible to see that as C_+ grows SVM is more reliable in detecting the feasible features, less and less feasible features are lost, as it is shown by the increasing value of TPR. On the other hand the model is less accurate in detecting the unfeasible features, and the number of false positives increases: when C_+ is large the machine identifies more feasible features but among them there are more false positives than for smaller values of C_+, as can be easily deduced by the decreasing value of PPV. So C_+ should be big enough to detect as many feasible solutions as possible and minimizing the number of feasible features lost due to the classification process (large TPR and small FOR), to form a reasonably large set of solutions to train the ANN, but not too much, to keep the number of unfeasible features incorrectly classified as feasible low (small FPR and high PPV), to avoid useless CFD computations and the performance evaluation of unfeasible geometries.

To investigate the choice of the best parameter combination a ROC curve could be employed, (5). In statistics, a Receiver Operating Characteristic, or ROC curve, is a graphical plot that illustrates the performance of a binary classifier as a parameter is varied. The curve is created by plotting the true positive rate (TPR) against the false positive rate (FPR) for different choices of the free parameter. The best possible prediction would yield no false negatives and no false positives and would correspond to a point in the upper left corner of coordinates (0,1) in the ROC space, while a random guess would give a point along the diagonal line from the left bottom corner to the top right one. So points above the diagonal represent classification results better than random ones, while points below the line correspond to bad results, worse than random. The best results from a confusion matrix are then the closest to the upper left corner, and the distance from the random guess line can be used as an indicator of how much predictive power a method has. In Figure 7 the ROC curves corresponding to Table 2 (top left), Table 3 (top right) and to Table 4 (bottom) are reported. For each test it is possible to deduce the best values for the free parameter, choosing the point closer to the upper left corner, which is marked by a black circle. Notice that these results are in good accordance with (12). In the tables then, the column corresponding to the best parameters combinations is highlighted by bold font.

Table 2. Results of the classification tests performed on Dataset 1. Labels meaning is introduced in Section 5.

3 : 1	$C_+ = 1$	$C_+ = 2$	$\mathbf{C_+ = 3}$	$C_+ = 5$	$C_+ = 10$
TPR	31.5%	66.1%	**78.3**%	88.5%	94.5%
FPR	4.1%	15.4%	**23.0**%	33.1%	44.7%
PPV	70.1 %	56.6%	**50.8**%	44.9 %	39.2 %
FOR	17.9 %	10.9%	**7.9**%	5.0 %	2.9%

Table 3. Results of the classification tests performed on Dataset 2. Labels meaning is introduced in Section 5.

7 : 1	$C_+ = 1$	$C_+ = 2$	$C_+ = 3$	$C_+ = 5$	$\mathbf{C_+ = 7}$	$C_+ = 10$	$C_+ = 20$
TPR	14.7%	53.5%	66.0%	77.5%	**83.0**%	86.5%	88.4%
FPR	0.7%	5.9%	9.9%	16.1%	**19.8**%	23.1%	25.7%
PPV	76.9%	58.5%	50.8%	42.7 %	**39.4**%	36.7 %	34.8 %
FOR	11.7%	7.1%	5.5%	3.9 %	**3.1**%	2.6%	2.3 %

Table 4. Results of the classification tests performed on Dataset 3. Labels meaning is introduced in Section 5.

6 : 1	$C_+ = 1$	$C_+ = 2$	$C_+ = 3$	$C_+ = 5$	$C_+ = 7$	$C_+ = 10$	$C_+ = 20$
TPR	20.5%	55.5%	67.9%	78.7%	**84.2**%	87.2%	89.7%
FPR	1.1%	6.1%	10.0%	15.0%	**18.2**%	20.7%	23.4%
PPV	74.6 %	58.9%	51.8%	45.2%	**42.1**%	39.8%	37.6 %
FOR	11.2 %	6.9 %	5.3%	3.8 %	**2.9**%	2.5%	2.1 %

Table 5. Results of the classification tests performed on Dataset 1 with optimal value of γ.

3:1	$C_+ = 1$	$C_+ = 2$	$\mathbf{C_+ = 3}$	$C_+ = 5$	$C_+ = 10$
TPR	32.7%	66.2%	**78.2**%	88.0%	93.8%
FPR	4.3%	15.4%	**22.8**%	32.5%	43.1%
PPV	70.0 %	56.6%	**51.1**%	45.2 %	39.8 %
FOR	17.6 %	10.8 %	**7.9**%	5.1 %	3.2 %

Then, independently of the choice of the parameter, even if SVM classifier allows in the feasible set a percentage of false positives, the general advantage gained by the use of the classification procedure is that it lowers the ratio between unfeasible and feasible features in the set $\mathcal{F}$ that has to be tested by CFD computations. This new ratio is given by $\frac{FDR}{PPV}$. In all the tests it is much lower then the one in the original dataset. In the datasets considered the ratio turns from 3:1 to about 1:1, from 7:1 to about 1.5:1 and from 6:1 to about 1.4:1. The practical outcome of this, is that when at step (2) of Phase 1 the design space is sampled, the most part of the unfeasible geometries is discarded by SVM. Then the CFD are performed on dataset $\mathcal{F}$ which is an almost balanced dataset. The same benefits apply to Phase 2, when the regression meta-model is applied at step (3) to $\mathcal{F}''$ that, like $\mathcal{F}$, is mainly composed of feasible features. Thus, the presence of unfeasible geometries in the optimal solutions set is reduced and the cost of step (5) is lowered.

Let us show these benefits with a practical example. The same test as in Section 3.1 is used, which is the one considered in Table 3. Let us assume to have tested with SVM enough features to have $|\mathcal{F}| \simeq 24600$, which it is worth remembering is not an expensive procedure. Then, considering that for this test case $PPV = 50.8\%$, performing about 24600 CFD computations it is possible to obtain a training set $\mathcal{F}'$ of about 12500 samples, the same size as the one yielded by the procedure sketched in Framework 3.1, that was obtained with 50000 computations (cfr. Section 3.1). The proposed approach allows then to save more than 50% CFD computations to form the ANN training set. Moreover the result of step (2) of Phase 2 is a dataset $\mathcal{F}''$ in which just half of the features will result to be unfeasible, so it is less likely to obtain optimal solutions sets composed just of

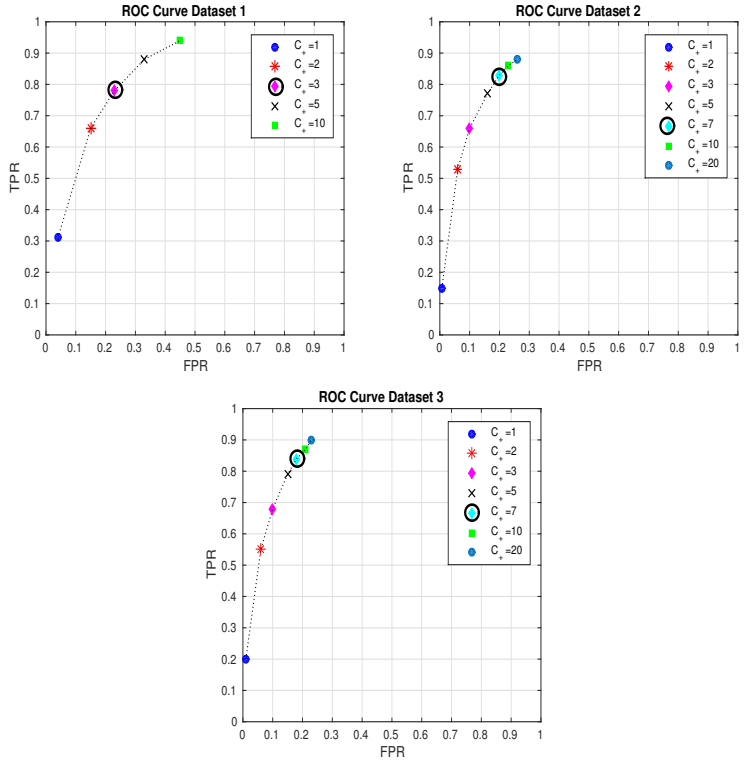

Figure 7. ROC curve, Dataset 1 top left, Dataset 2 top right, Dataset 3 bottom. C_+ weighting factor for feasible features, best value is highlighted by a black circle.

unfeasible features.

Similar remarks can be made for the others datasets. In that cases, due to the stronger unbalancedness, the savings in CFD computations are even higher, around 67%.

Notice also that the percentage of feasible features in the set of features classified as unfeasible (and then wrongly discarded after the classification process) is low, about 8% for Dataset 1 and 3% or the other two.

All of this, leads to conclude that the proposed procedure is actually effective in reducing the computational costs and improves the previously considered approach.

The strategy chosen to handle the dataset unbalancedness shows to be effective, SVM classification meta-model performs well and allows to successfully handle problems with different unbalancedness levels. In fact, even if the last two classification problems are more difficult than the first, with a suitable choice of parameter C_+ good results are obtained.

Finally, it is worth noticing that in all the tests presented above, independently of the dataset considered, the same value $\gamma = 0.0114$ is used. Finding the right value for the free parameter, using either a cross-validation or the average squared distance among training patterns is rather a time consuming computation, that should be performed each time the dataset is varied. It is convenient to use the same parameter to build models also for different kind of machines, to save computational time and to have a tool that does not need to be tuned on each test case. The γ value that was used in the tests is optimized for the dataset with $n = 44$ degrees of freedom. While the optimal parameter for the dataset with $n = 42$ degrees of freedom is quite the same, the optimal one for the other dataset is $\gamma = 0.0135$. In Table 5 the results obtained with the optimal parameter $\gamma = 0.0135$ for the dataset with 40 degrees of freedom are reported. Comparing them with those in Table 3, it is possible to see that they do not change significantly, so the same parameter γ can be used for all this dataset without loss of accuracy.

7. Conclusions and remarks

In this article a hybrid approach to face the parametric design of a centrifugal pump is presented, which is based on coupling CFD computations, SVM classification and ANN regression. An intrinsic property of a parametric design is the presence of many unfeasible geometries in the design space, so that the databases formed according to the parameterization chosen are really unbalanced. This has two main drawbacks. On one hand, a huge number of CFD computations is necessary to form a suitable training set for the regression meta-model. On the other hand, the optimal solutions set found evaluating the performance functions of new samples by means of the trained meta-model, could be made just of unfeasible geometries. A strategy to solve these two problems, based on coupling the regression process with a classification by SVM, is presented. The strategy was tested on various datasets with a large number of degrees of freedom, different constraints and different ratio between feasible and unfeasible features. It is shown that with a fine tuning of the free parameters different unbalancedness levels can be handled. Moreover, the use of the classification procedure allows to discard the most part of the unfeasible samples, making the design procedure doable and cutting the number of required CFD computations by about $50\% - 70\%$.

References

[1] A. Arnone. Viscous analysis of three-dimensional rotor flow using a multigrid method. *Journal of Turbomachinery*, 116(3):435–445, doi:10.1115/1.2929430, 1994.

[2] A. Arnone. Multigrid methods for turbomachinery navier-stokes calculations. In *Solution Techniques for Large-Scale CFD Problems*, pages 293–332. John Wiley and Sons, New York, 1995.

[3] A. Arnone and R. Pacciani. Three-dimensional viscous analysis of centrifugal impellers using the incompressible navier-stokes equations. In *Proceedings of 1st European Conference on Turbomachinery, Erlangen, Germany*, pages 181–195, 1995.

[4] D. Bonaiuti, A. Arnone, M. Ermini, and L. Baldassarre. Analysis and optimization of transonic centrifugal compressor impellers using the design of experiments technique. *Journal of Turbomachinery*, 128(4):786–797, doi:10.1115/1.1579507, 2006.

[5] A.P. Bradley. The use of the area under the roc curve in the evaluation of machine learning algorithms. *Pattern Recognition*, 30(7):1145–1159, doi: 10.1016/S0031–3203(96)00142–2, 1997.

[6] C. Chahine, J. R. Seume, and T. Verstraete. The influence of metamodeling techniques on the multidisciplinary design optimization of a radial compressor impeller. In *Proceedings of ASME Turbo Expo 2012, 11-15 June 2012, Copenhagen, Denmark*, pages 1951–1964. doi:10.1115/GT2012-68358, 2012.

[7] C.C. Chang and C.J. Lin. Libsvm: a library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2(3):Article No. 27, 2011.

[8] M. Checcucci, F. Sazzini, M. Marconcini, A. Arnone, M. Coneri, L. De Franco, and M. Toselli. Assessment of a neural-network-based optimization tool: a low specific-speed impeller application. *International Journal of Rotating Machinery*, 2011, doi:10.1155/2011/817547, 2011.

[9] M. Checcucci, A. Schneider, M. Marconcini, F. Rubechini, A. Arnone, L. De Franco, and M. Coneri. A novel approach to parametric design of centrifugal pumps for a wide range of specific speeds. In *Conference: 12th International Symposium on Experimental and Computational Aerothermodynamics of Internal Flows, 13-16 July 2015, Lerici (SP), Italy*, page ISAIF 12 paper nr.121, 2015.

[10] L. Ellbrant, L.E. Eriksson, and H. Mårtensson. Design of compressor blades considering efficiency and stability using cfd based optimization. In *Proceedings of ASME Turbo Expo 2012, 11-15 June 2012, Copenhagen, Denmark*, pages 371–382. doi:10.1115/GT2012-69272, 2012.

[11] S. Haykin. *Neural Networks: A Comprehensive Foundation. 2nd edition.* Macmillan, New York, ISBN: 978-0780334946, 1998.

[12] P.H. Hergt. Pump research and development: Past, present, and future. *Journal of Fluids Engineering*, 121(2):248–253, doi: 10.1115/1.2822198, 1999.

[13] H. Liu, K. Wang, S. Yuan, M. Tan, Y. Wang, and L. Dong. Multicondition optimization and experimental measurements of a double-blade centrifugal pump impeller. *Journal of Fluids Engineering*, 135(1):0111031–01110313, doi: 10.1115/1.4023077, 2013.

[14] Dingguo Lu and Wei Qiao. A ga-svm hybrid classifier for multiclass fault identification of drivetrain gearboxes. In *Energy Conversion Congress and Exposition (ECCE), 2014*, pages 3894–3900. IEEE, 2014.

[15] J. Monedero. Parametric design: a review and some experiences. *Automation in Construction*, 9(4):369–377, 2000.

[16] R. Ñanculef, E. Frandi, C. Sartori, and H. Allende. A novel frank–wolfe algorithm. analysis and applications to large-scale svm training. *Information Sciences*, 285:66–99, doi:10.1016/j.ins.2014.03.059, 2014.

[17] S. Nandi, Y. Badhe, J. Lonari, U. Sridevi, B.S. Rao, S.S. Tambe, and B.D. Kulkarni. Hybrid process modeling and optimization strategies integrating neural networks/support vector regression and genetic algorithms: Study of benzene isopropylation on hbeta catalyst. *Chemical Engineering Journal*, 97(2):115–129, doi:10.1016/S1385–8947(03)00150–5, 2004.

[18] D. L. Olson and D. Delen. *Advanced Data Mining Techniques.* Springer Science & Business Media, ISBN 978-3-540-76917-0, 2008.

[19] E. Osuna, R. Freund, and F. Girosit. Training support vector machines: an application to face detection. In *Proceedings of Computer Society Conference on Computer Vision and Pattern Recognition 1997, 17-19 June 1997, Puerto Rico*, pages 130–136. IEEE, 1997.

[20] S. Pierret. Turbomachinery blade design using a navier-stokes solver and artificial neural network. *ASME Journal of Turbomachinery*, 121(3):326–332, doi:10.1115/1.2841318, 1999.

[21] F. Rubechini, A. Schneider, A. Arnone, S. Cecchi, and F. A. Malavasi. A redesign strategy to improve the efficiency of a 17-stage steam turbine. In *Proceedings of ASME Turbo Expo 2009, 8–12 June 2009, Orlando, Florida.*, pages 1463–1470. doi:10.1115/GT2009-60083, 2009.

[22] F. Rubechini, A. Schneider, A. Arnone, F. Dacca C. Canelli, and P. Garibaldi. Aerodynamic redesigning of an industrial gas turbine. In *Proceedings of ASME Turbo Expo 2011, 6-10 June 2011, Vancouver, BC*, pages 1387–1394. doi:10.1115/GT2011-46258, 2011.

[23] B. Schölkopf and A. J. Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond.* MIT Press, ISBN: 9780262194754, 2001.

[24] D.H. Seo, T.S. Roh, and D.W. Choi. Defect diagnostics of gas turbine engine using hybrid svm-ann with module system in off-design condition. *Journal of Mechanical Science and Technology*, 23(3):677–685, doi:10.1007/s12206–008–1120–3, 2009.

[25] H. Shin and S. Cho. How to deal with large dataset, class imbalance and binary output in svm based response model. In *Proceedings of the Korean Data Mining Conference*, pages 93–107, 2003.

[26] F. EH. Tay and L. Cao. Application of support vector machines in financial time series forecasting. *Omega*, 29(4):309–317, doi:10.1016/S0305–0483(01)00026–3, 2001.

[27] S. Tong and D. Koller. Support vector machine active learning with applications to text classification. *Journal of Machine Learning Research*, 2(11):45–66, doi:10.1162/153244302760185243, 2001.

[28] I. W. Tsang, J. T. Kwok, and P.M. Cheung. Core vector machines: Fast svm training on very large data sets. *Journal of Machine Learning Research*, 6:363–392, 2005.

[29] A. Veress and R. Van den Braembussche. Inverse design and optimization of a return channel for a multistage centrifugal compressor. *Journal of Fluids Engineering*, 126(5):799–806, doi:10.1115/1.1792258, 2004.