SSRG International Journal of Industrial Engineering
ISSN: 2349 – 9362 /doi:10.14445/23499362/IJIE-V8I1P103

Volume 8 Issue 1, 14-20, Jan-Apr 2021
© 2021 Seventh Sense Research Group®

Machine Learning Can Maximize Efficiency in an Industrial Process

Wolfgang Mergenthaler[#1], Daniel Jaroszewski[#1], Salah-Eddine Morsili[#2], Benedikt Sturm[#1]

[#1] FCE Frankfurt Consulting Engineers GmbH, Bessie- Coleman-Strasse 7, D-65439 Frankfurt, Germany
[#2] EDF DPNT – Nuclear and Thermal Generation Division – DTEAM, Site Cap Ampère 1, Place Pleyel 93 282 Saint-Denis Cedex

Abstract - *Continuous industrial manufacturing processes are generally controlled by a set of continuous control variables. The process usually produces a steady flow of output material, such as cement, food, milk, chemicals, and sugar or electrical power in a power plant. Control variables may be electrical or fossil power, cooling or heating, lubrication, pressure, etc. The process responds with a given flow of output measured in tons per day or power expressed in Megawatt. Dividing the input power response yields a variable proportional to the degree of efficiency of the process, which is a very important parameter in most cases. To understand, analyze or predict the process, in a first step, we will approximate the empirical response values by a smooth function, mapping the space of controls onto the interval [0,100%], using Machine Learning Techniques and Multivariate Statistics such as Tensor Flow or Generalized Linear Models (GLMs), respectively. Both approaches provide suitable approximation measures. In a second step, the process will be optimized within a given set of constraints concerning the control variables. This step will be illustrated by GLMs only due to their lack of overfitting and their continuous differentiability properties. This way, optimal set points, and sensitivity coefficients will be given.*

Keywords — *Machine Learning, Tensor Flow, Generalized Linear Models, Nonlinear Optimization, Sensitivity Coefficients*

I. INTRODUCTION

Industrial processes, from the data processing point of view, are usually captured as a sequence of records along an equidistant time grid. Each record consists of a timestamp, a set of discrete switch variables, a set of control variables, and finally, a set of response variables. In this paper, we will look for continuous, preferably continuously differentiable functions, which will map, for each combination of switches, the space of controls to the set of responses. Whereas linear function fitting calls for univariate or multivariate linear regression, the multidimensional nonlinear case is what we consider here, using Machine Learning or Multivariate Statistics such as Tensor Flow or Response Surface Methodology, respectively, see [9] and [15].

In the context of Experimental Design, Process Analysis, Optimal Process Planning, etc., process optimization has been an ongoing endeavor in the manufacturing world for many years and from several points of view. For instance, in the context of Experimental Design, Response Surface Methodology has been widely applied in search of new operating points, see [14]. Quality assurance has brought about Statistical Process Control (SPC), mostly focusing on one-dimensional processes, see [10] or [13]. Process optimization has also been influenced by the emergence of Local Search Techniques such a Genetic Algorithms, see [4]. All of these efforts, however, have been struggling with either high dimensions or non-linearity or both. The current paper shows that multidimensional industrial processes can be both approximated and optimized efficiently. We show that there is a logical path from raw data to optimal set points and sensitivity coefficients using data analysis and mathematics. In the future, environmental and sustainability requirements will increasingly dictate industrial process optimization measures. It is shown that the slope of the linear part and the eigenvalues of the quadratic form in the response function provide a sufficient criterion for the suitability of a given rectangle when it comes to estimating sensitivity coefficients. We also show that the estimators of the sensitivity coefficients can be used to formulate a linear optimization problem on the control variables under cost constraints. Whereas this paper focuses on one response variable at a time, an approximation can be done on several variables simultaneously, such as efficiency, emissions, vibration, etc. Optimization, however, in this case, needs a multiple-criteria decision-making framework such as Pareto optimization, which is not considered here.

II. INPUT DATA

Let $T := \{t_1, \dots t_N\}$ be a set of time indices. Consider a particular time $t \in T$. Let n_b be a number of binary switch variables and $z_{t,i} \in \{0,1\}$ for all $i \in \{1, \dots, n_b\}$. Furthermore, let n_c be a number of continuous, independent control variables of the process under consideration and let $x_{t,i} \in \mathbb{R}, i \in 1, \dots, n_c\}$ be the corresponding variables. Finally, let m
be a number of response variables and let $y_{t,i} \in \mathbb{R}, i \in$

$\{1,\ldots,m\}$ be the corresponding values. Set, for all $t \in T$

(1) $z_t \coloneqq (z_{t,i}, i \in \{1,\ldots,n_b\})$

(2) $x_t \coloneqq (x_{t,i}, i \in \{1,\ldots,n_c\})$

(3) $y_t \coloneqq (y_{t,i}, i \in \{1,\ldots,m\})$

The input data consist of a sequence of tuples

(4) $\{z_t, x_t, y_t\}, t \in T$

And take the form of a table. With these data as input, we can focus on the task of finding a function $f: [0,1]^{n_b} \times \mathbb{R}^{n_c} \to \mathbb{R}^m$ such that

(5) $y_t = f(z_t, x_t) + \epsilon_t$

whereby ϵ_t is an m-dimensional random variable with

(6) $E[\epsilon_t] = 0$

(7) $Var[\epsilon_t] = \begin{pmatrix} \sigma_1^2 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \sigma_m^2 \end{pmatrix}$

for some given variances $\sigma_1^2,\ldots,\sigma_m^2$ and for all $t \in T$. Finding the function f is the objective of the regression step. For the rest of the paper - for the sake of simplicity- we will assume

(8) $m = 1$

i.e., only one response variable will be considered, as mentioned. Furthermore, we assume that $z_t, t \in T$ is constant throughout T and equal to one dominating combination of switch variables $z \in \{0,1\}^{n_b}$ Only representing a full load situation, for example. The argument z in $f(z,x)$ will therefore be suppressed from now on, and we set $n \coloneqq n_c$.

III. REGRESSION

There are various approaches to finding f. One method uses a Generalized Linear Model (GLM), see [5], another one a Machine Learning approach, such as a Deep Neural Network, see [1], [6], [15]. Before looking at the details of the regression step, we fix a rectilinear region of the input data. Therefore let

(9) $A \coloneqq \{x \in \mathbb{R}^n\}$

(10) $l_1 \le x_1 \le u_1, \ldots, l_n \le x_n \le u_n$

whereby l_i and $u_i, i \in \{1,\ldots,n\}$ are lower and upper bounds of the continuous control variables. Figure 1 shows a typical response surface in the case of two control variables:

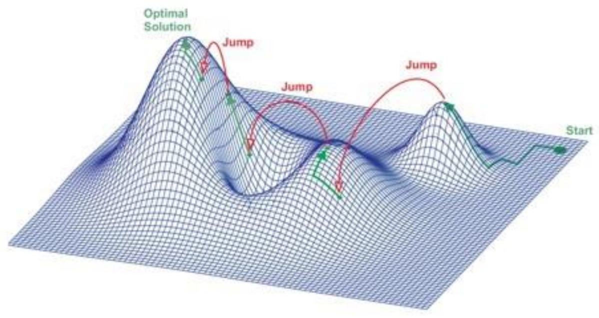


Figure 1: Typical response surface

A. *Generalized Linear Models*

Henceforth we assume

(11) $f(x) = a + \sum_{i=1}^{n} b_i x_i + \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij}\, x_i x_j$

The Generalized Linear Model paradigm becomes obvious when we introduce the following substitutions:

(12) $\tilde{n} = n + n^2$

(13) $g(i) = \left\lfloor \frac{i-n}{n} \right\rfloor, n+1 \le i \le \tilde{n}$

(14) $h(i) = (i-n) \bmod n,\ n+1 \le i \le \tilde{n}$

With

(15) $v_i \coloneqq \begin{cases} x_i, 1 \le i \le n \\ x_{g(i)} x_{h(i)}, n+1 \le i \le \tilde{n} \end{cases}$

(16) $\alpha_0 = a$

(17) $\alpha_i = \begin{cases} b_i, 1 \le i \le n \\ c_{g(i),h(i)}, n+1 \le i \le \tilde{n} \end{cases}$

one obtains

(18) $f(v) = \alpha_0 + \sum_{i=1}^{\tilde{n}} \alpha_i v_i$

Now, the usual way to determine the parameter vector $\alpha \coloneqq = (\alpha_0, \ldots \alpha_{\tilde{n}})^T$ seeks to minimize the sum of squares of the deviations between predicted and observed y −values. Therefore, we define the $|T| \times (1+n)$- matrix H as

(19) $H = \begin{bmatrix} 1, v_1^{(1)}, v_2^{(1)} & \cdots & v_{\tilde{n}}^{(1)} \\ \vdots & \ddots & \vdots \\ 1, v_1^{(|T|)}, v_2^{(|T|)} & \cdots & v_{\tilde{n}}^{(|T|)} \end{bmatrix}$

With upper indices counting records and lower indices counting variables. Let $\hat{y} \coloneqq (\hat{y}_1, \ldots \hat{y}_{|T|})^T$ be the vector of observed values. We then want to find that particular value

α^*, which solves the following minimization problem:

$$(20)\quad (H\alpha^* - \hat{y})^T(H\alpha^* - \hat{y}) = Min_{\alpha\in\mathbb{R}^{n+1}}(H\alpha - \hat{y})^T(H\alpha - \hat{y})$$

Expanding, differentiating with respect to α, setting the result equal to zero and solving for α^* yields

$$(21)\quad \alpha^* = (H^TH)^{-1}\hat{y}^TH$$

provided (H^TH) is non-singular, see [3]. Abbreviating

$$(22)\quad \gamma_{ij} = \alpha_{i*n+j},$$
$$(23)\quad 1 \le i \le n, 1 \le j \le n$$

and dropping the asterisk, we can express the expected value of our response variable as

$$(24)\quad f(x) = \alpha_0 + \sum_{i=1}^n \alpha_i x_i + \sum_{k=1}^n \sum_{l=1}^n \gamma_{kl} x_k x_l$$

upon replacing $v_i, i \in \{1, \dots, \tilde{n}\}$ by their definitions.

B. Neural Networks

The most prominent functional approximation technique in use today is the so-called Deep Neural Network. Google's famous Tensor Flow package, as described in [1], is a very convenient platform for the Deep Learning Technique. It can be used both for classification as well as for regression purposes. In the examples, we will show the extremely precise approximation capability of this technique, which is used in applications as diverse as time series forecasting and image recognition.

However, to the knowledge of the authors, there is no fast, functional differentiation routine of a Deep Neural Network with respect to the input pattern. Therefore, in the example shown below, we will use Deep Neural Networks for the sole purpose of illustrating its approximation capabilities and then proceed with the Generalized Linear Model.

IV. NONLINEAR OPTIMIZATION

Now assume that we would like to find the maximum of $f(x)$ within a region $A \subseteq \mathbb{R}^n$, i.e.

$$(25)\quad x^*: f(x^*) = max_{x\in A} f(x)$$

We must discriminate between two different cases:

A. Continuous Differentiability

As a prototype for this case, we assume that $f(x)$ is represented by a Generalized Linear Model. Defining vector α and matrix Γ as

$$(26)\quad \alpha := (\alpha_1, \dots, \alpha_n)^T$$

$$(27)\quad \Gamma = \begin{pmatrix} \gamma_{11} & \cdots & \gamma_{1n} \\ \vdots & \ddots & \vdots \\ \gamma_{n1} & \cdots & \gamma_{nn} \end{pmatrix}$$

we can express $f(x)$ as

$$(28)\quad f(x) = \alpha_0 + \alpha^T x + x^T \Gamma x$$

The first attempt to find x^* requires finding the gradient of $f(x)$, setting it equal to zero and solving the resulting system of equations, i.e.

$$(29)\quad \nabla f(x)|_{x=x^*} = \alpha + 2\Gamma x = 0$$

i.e.

$$(30)\quad x^* = -\frac{1}{2}\Gamma^{-1}\alpha$$

Now, if $x^* \in A$, we are done. This can be proven by applying the set of inequalities (10) to x^*. If however $x^* \notin A$, then either one of the following may be the case:

- x^* is on one of the faces of A
- x^* is on one of the edges of A
- ...
- x^* is in one of the corners of A

B. Local Search Procedures

In each of the cases above, there is no easy way to find x^* With methods using derivatives. In this case, we must resort to derivative-free minimization or maximization problems, such as Genetic Algorithms, see [4] or the Simulated Annealing Local Search Technique, see [8]. Local Search tries to find the optimum in a sequence of approximations x^k to x^*. Whereas Local Search can be very effective in the case of discontinuous or non-differentiable response functions, it tends to converge slowly. Therefore, in this paper, we revert to Generalized Linear Models and gradient descent.

V. SENSITIVITY COEFFICIENTS

In many cases, it is of interest to study the impact of each of the regressors on the response variable. One possible way to measure this effect is by means of the average partial derivative of the response with respect to the regressor in question, whereby averaging is done over the region of interest A. Let

(31) $\rho_{i,A} = \frac{1}{|A|}\sum_{x\in A}\frac{\partial f(x)}{\partial x_i}$

It is obvious that the averaging step as applied above can destroy meaningful information when the variance of $\frac{\partial f(x)}{\partial x_i}$ Across A is large. It is reasonable, therefore, to concentrate on small enough regions A. Therefore, we formulate the following lemma:

Lemma 1: Let $f(x)$ be Lipschitz continuous with Lipschitz constant L. Then, for any $x_1 \in A$, $x_2 \in A$ we have

(32) $\underline{|f(x_1) - f(x_2)| \leq L|x_1 - x_1|}$

Proof: The proof is straightforward by the definition of Lipschitz continuity. On a detailed account of Lipschitz continuity and its use in the calculus of variations, see, for instance [2].

Corollary 1: Since lemma 1 holds for the two special variables

(33) $l := (l_1, \ldots, l_n)^T$
(34) $u := (u_1, \ldots, u_n)^T$

one obtains for any Lipschitz continuous function $f(x)$ for the maximum variation of $f(x)$ across A - termed $V(f)_A$ for the time being –

(35) $V(f)_A \leq L|u - l|$

Lemma 2: Let Γ be asymmetric, positive definite matrix and let $p_i, \lambda_i, i \in \{1, \ldots, n\}$ be the eigenvectors and eigenvalues of matrix Γ, respectively. If $f(x)$ can be expressed as our quadratic regression function, the difference between the response values for two arbitrary regressor values $w \in A$ and $v \in A$ takes the following form:

(36) $f(w) - f(v) = \alpha^T(w - v) +$

$+\sum_{i=1}^{n}\lambda_i((w^T p_i)^2 - (v^T p_i)^2$

Proof: Express both w and v in terms of the eigenvectors of Γ, i.e., $w = \sum_{i=1}^{n}(w^T p_i)\, p_i$ and $v = \sum_{i=1}^{n}(v^T p_i)\, p_i$. Then

(37) $f(w) = \alpha_0 + \alpha^T w +$

$$+\sum_{i=1}^{n}\sum_{j=1}^{n}(w^T p_i)p_i^T \Gamma\, (w^T p_j)p_j =$$

$$= \alpha_0 + \alpha^T w + \sum_{i=1}^{n} p_i (w^T p_i)^2$$

And analogously for $f(v)$.

Corollary 2:

(38) $V(f)_A = \ max_{w\in A, v\in A}|f(w) - f(v)| \leq$

$$\leq max_{w\in A,v\in A}\{|\alpha||w - v|$$
$$+\sum_{i=1}^{n}\lambda_i|(w^T p_i)^2 - (v^T p_i)^2|\}$$

Proof: Using the above lemma on both $f(w)$ and $f(v)$ we have for any $w \in A$ and $v \in A$

(39) $f(w) - f(v) = \alpha^T(u - v)+$
$+\sum_{i=1}^{n}\lambda_i((u^T p_i)^2\text{-}\,(v^T p_i)^2)$

Taking the maximum with respect to w and v proves the statement. Assuming that the maximum is obtained for $w = u$ and $v = l$, we can say that

(40) $V(f)_A \approx \alpha^T(u - l) + \sum_{i=1}^{n}\lambda_i((u^T p_i)^2 - (l^T p_i)^2)$

This corollary provides a sufficient criterion for the suitability of region A, when it comes to estimating the sensitivity coefficients by averaging over this region. That means that if – within the largest distance possible inside A – variation $V(f)_A$ is small enough, then A is appropriate. From a practical point of view, it might be important to rank $\rho_{i,A}$ with respect to size for a very simple reason: If maximizing $f(x)$ is the goal dictated by business, then it would be plausible to increase – within technical limits – those regressors with either maximum sensitivity and/or minimal cost.

Assume that changing the value of x_i by an amount of Δx_i amounts to $c_i * \Delta x_i$ cost units. The overall cost arising from changing regressor $i, i \in \{1, \ldots, n\}$ by an amount Δx_i is then given as

(41) $\Delta c(\Delta x) = \sum_{i=1}^{n} c_i \, \Delta x_i$

defining the vector of changes as $\Delta x := \{\Delta x_1, \ldots, \Delta x_n\}$. The approximate amount by which the response variable changes is

(42) $\Delta f(\Delta x) = \sum_{i=1}^{n} \rho_{i,A} \Delta x_i$

An interesting linear program is then given by

(43) $Max_{\Delta x \in \mathbb{R}^n} \, \Delta f(\Delta x) = \sum_{i=1}^{n} \rho_{i,A} \, \Delta x_i$

such that

(44) $\Delta c(\Delta x) = \sum_{i=1}^{n} c_i \Delta x_i \leq c_0$

for some c_0 and

(45) $\Delta x_i^l \leq x_i < \Delta x_i^u, i \in \{1, \ldots, n\}$

Are lower and upper limits for the possible changes.

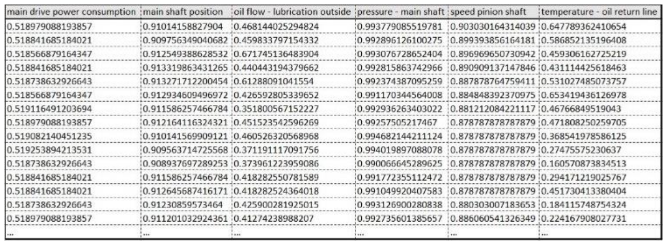
VI. EXAMPLES

Here we look at two examples. Example 1 illustrates the precise approximation capabilities of the Tensor Flow Deep Learning Technique. Example 2 uses gas turbine data and illustrates – in addition – the optimization and the sensitivity analysis step. Hereby we replace efficiency with power, which, when applied in combination with a given set of control variables, will be almost equivalent to efficiency when the region of interest is small. In this case, there will be no wide change in the fuel flow, meaning that power is nearly proportional to efficiency.

A. Crusher Data in a Cement Plant

A crusher in a cement plant will take large chunks of raw material and grind them into smaller, pebble-like structures. Table 1 shows a section of the training input data, consisting of 32000 records:

TABLE 1: CEMENT DATA



main drive power consumption	main shaft position	oil flow - lubrication outside	pressure - main shaft	speed pinion shaft	temperature - oil return line
0.518979088193857	0.91014158827904	0.468144025294824	0.993779085519781	0.903030164314039	0.647789362410654
0.518841685184021	0.909756349040682	0.459833797154332	0.992896126100275	0.899393856164181	0.586852135196408
0.518566879164347	0.912549388628532	0.671745136483904	0.993076728652404	0.896969650730942	0.459306162725219
0.518841685184021	0.913319863431265	0.440443194379662	0.992815863742966	0.890909137147846	0.431114425618463
0.518738632926643	0.913271712200454	0.61288091041554	0.992374387095259	0.887878764759411	0.531027485073757
0.518566879164347	0.912934609496972	0.426592805339652	0.991170344564008	0.884848392370975	0.653419436126978
0.519116491203694	0.911586257466784	0.351800567152227	0.992936263403022	0.881212084221117	0.46766849519043
0.518979088193857	0.912164116324321	0.451523542596269	0.99257505217467	0.878787878787879	0.471808250259705
0.519082140451235	0.910141569909121	0.460526320568968	0.994682144211124	0.878787878787879	0.368541978586125
0.519253894213531	0.909563714725568	0.371191117091756	0.994019897088078	0.878787878787879	0.27475575230637
0.518738632926643	0.908937697289253	0.373961223959086	0.990066645289625	0.878787878787879	0.160570873834513
0.518841685184021	0.911586257466784	0.418282550781589	0.991772355112472	0.878787878787879	0.294171219025767
0.518841685184021	0.912645687416171	0.418282524364018	0.991049920407583	0.878787878787879	0.451730413380404
0.518738632926643	0.91230859573464	0.425900281925015	0.993126900280838	0.880303007183653	0.184115748754324
0.518979088193857	0.911201032924361	0.41274238988207	0.992735601385657	0.886060541326349	0.224167908027731
...	...	...	...	...	...

Please note that this example was computed with variables normalized to values between *0* and *1*. The validation file will consist of 8000 corresponding records. The main purpose of this section is the application of the *Tensor Flow Deep Learning Technique* as a means to approximate the main drive power consumption response variable. Representing the response as a function of the remaining variables, we obtain the graph shown in figure 2:

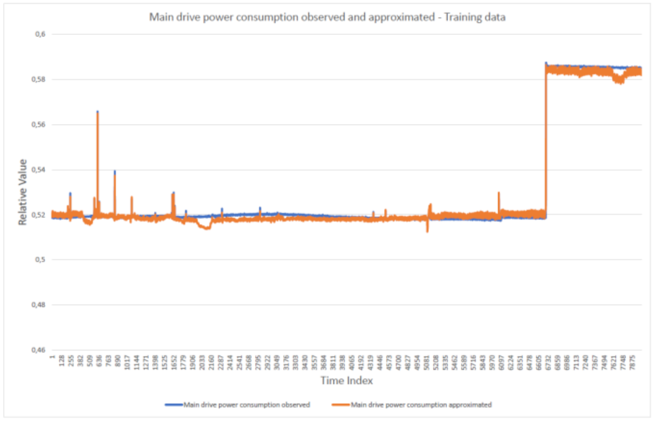


Figure 2: Main drive power observed vs. approximated-Training Data

When applying the Deep Learning approximation technique to the validation data, we obtain figure 3:

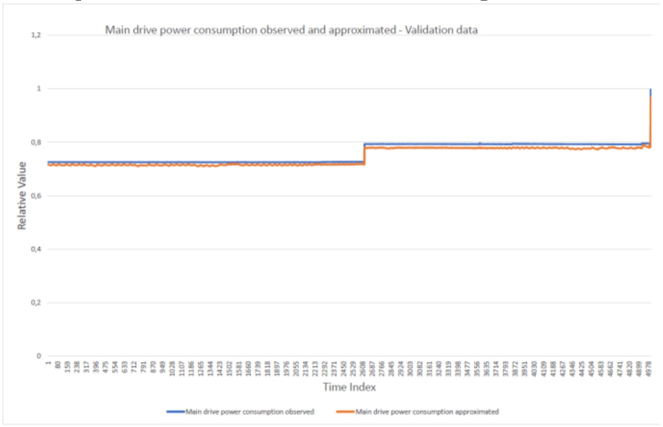


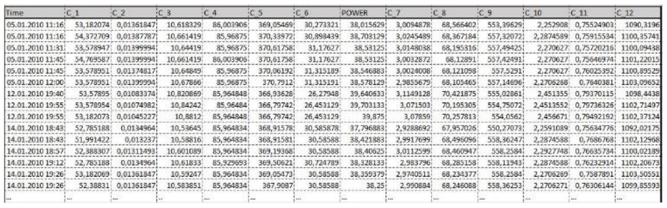
Figure 3. Main drive power observed vs. approximated - Validation Data

These figures show that Tensor Flow can cope very well with virtually discontinuous steps in the data. However, due to its tendency to overfit and since there is no convenient analytical derivative routine available, only the Generalized Linear Model is being used to do process optimization in a second step.

B. Gas Turbine Data

As another example, we look at the data of a gas turbine. We have a dataset of 1533 records; see table 2. The file will be split into a file with 1226 training records, used to do the nonlinear regression step and 317 validation records. The dataset contains a timestamp, 12 control variables, and power as the response variable.

TABLE 2: GAS TURBINE DATA



Time	C_1	C_2	C_3	C_4	C_5	C_6	POWER	C_7	C_8	C_9	C_10	C_11	C_12
05.01.2010 11:16	53,182074	0,01361847	10,618329	86,003906	369,05469	30,273321	38,015629	3,0094878	68,566402	553,39629	2,252908	0,75524903	1090,3196
05.01.2010 11:16	54,372709	0,01387787	10,661419	85,96875	370,33972	30,898439	38,703129	3,0245489	68,367184	557,32072	2,2874589	0,75915534	1100,35741
05.01.2010 11:31	53,578947	0,01399994	10,64419	85,96875	370,61758	31,17627	38,53125	3,0148038	68,195316	557,49425	2,270627	0,75720216	1100,09438
05.01.2010 11:45	54,769587	0,01399994	10,661419	86,003906	370,61758	31,17627	38,53125	3,0032872	68,12891	557,42491	2,270627	0,75646974	1101,22015
05.01.2010 11:45	53,578951	0,01374817	10,64849	85,96875	370,06192	31,315189	38,546883	3,0024008	68,121098	557,5291	2,270627	0,76025392	1100,89529
05.01.2010 12:00	53,578951	0,01399994	10,67866	85,96875	370,7912	31,315191	38,578129	2,9855679	68,105465	557,14696	2,2706268	0,7640381	1103,09652
12.01.2010 19:40	53,57895	0,01083374	10,820869	85,964848	366,93628	26,27948	39,640633	3,1149128	70,421875	555,02861	2,451355	0,79370115	1098,4438
12.01.2010 19:55	53,578954	0,01074982	10,84242	85,96484	366,79742	26,453129	39,703133	3,071503	70,195305	554,75072	2,4513552	0,79736326	1102,71497
12.01.2010 19:55	53,182073	0,01045227	10,8812	85,964848	366,79742	26,453129	39,875	3,07859	70,257813	554,0562	2,456671	0,79492192	1102,37124
14.01.2010 18:43	52,785188	0,0134964	10,53645	85,964834	368,91578	30,585878	37,796883	2,9288692	67,957026	550,27073	2,2591089	0,75634776	1092,02175
14.01.2010 18:43	51,991422	0,013237	10,58816	85,964834	368,91581	30,58588	38,421883	2,9917699	68,496096	558,36247	2,2874588	0,7686768	1102,12968
14.01.2010 18:57	52,388307	0,01311493	10,601089	85,964834	369,19368	30,58588	38,40625	3,0112599	68,460947	558,2584	2,2927748	0,76635734	1100,02189
14.01.2010 19:12	52,785188	0,0134964	10,61833	85,929693	369,50621	30,724789	38,328133	2,983796	68,285158	558,11943	2,2874588	0,76232914	1102,20673
14.01.2010 19:26	53,182069	0,01361847	10,59247	85,964834	369,05473	30,58588	38,359379	2,9740511	68,234377	558,2584	2,2706269	0,7587891	1103,50551
14.01.2010 19:26	52,38831	0,01361847	10,583851	85,964834	367,9087	30,58588	38,25	2,990884	68,246088	558,36253	2,2706271	0,76306144	1099,85593
...	...	...	...	...	...	...	...	...	...	...	...	...	...

Regression using Neural Networks: Applying Tensor Flow to the input data and modeling gas turbine power as a function of the control variables shows the results as given in figures 4 and 5:

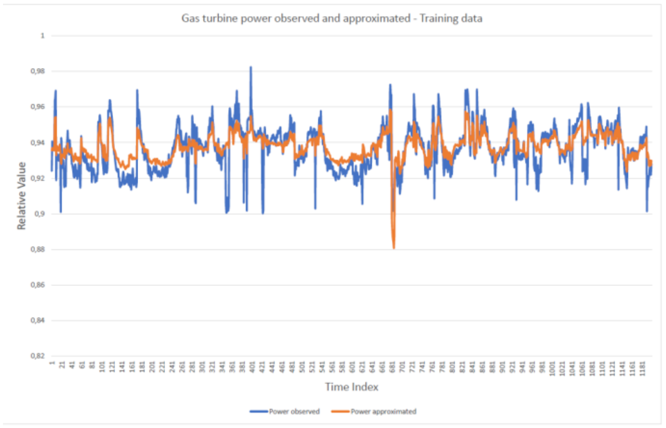


Figure 4: Tensor Flow approximation of the power signal on the training Data

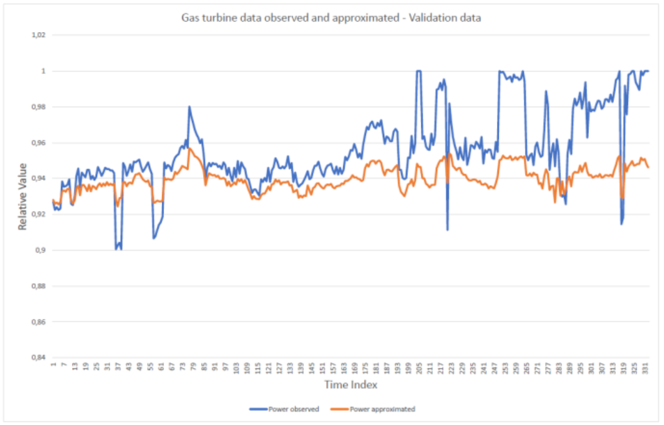


Figure 5: Tensor Flow approximation of the power signal on the validation data

These figures show that Neural Networks have a tendency to overfit the training data. As a consequence, the goodness of approximation is less good on the validation data than on the training data. Applying the GLM model to the training data and the validation data results in figures 6 and 7:

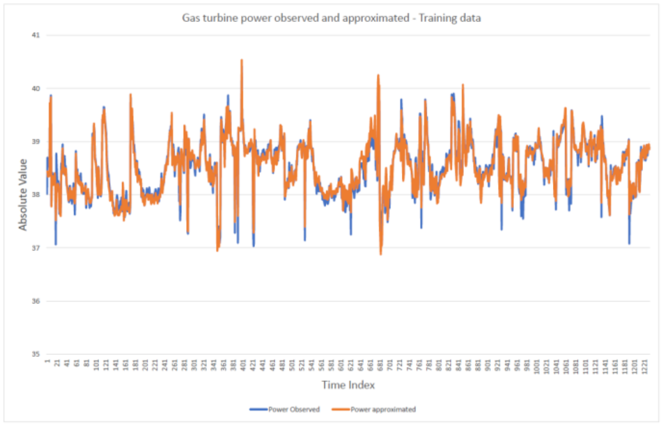


Figure 6: GLM approximation of the power signal on the training Data

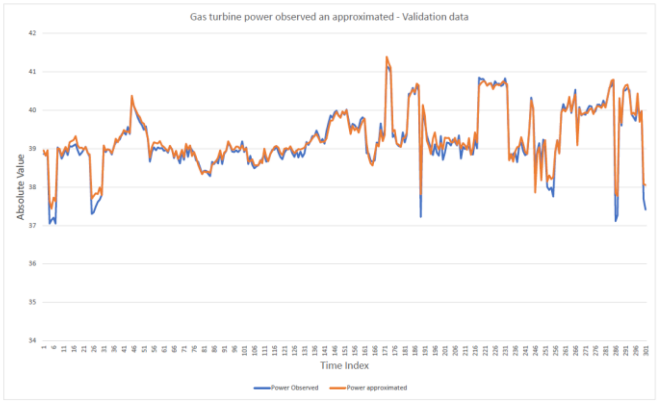


Figure 7: GLM approximation of the power signal on the validation data

Optimization: For the rest of this paper, we will use the GLM method exclusively, the reason being that we will make extensive use of the gradient of the response function, which is very convenient to compute for GLM and less easy for Neural Networks. When it comes to optimizing the response variable in a given domain *A*, we must specify the latter. One simple way to do this is to give lower and upper limits for the individual regression variables, as shown in table 3:

TABLE 3: SPECIFYING THE FEASIBLE REGION

Control	Lower bound	Upper bund
C_2	0.00815	0.01669
C_3	10.4632	11.1656
C_6	22.8760	33.7115
C_7	2.75611	3.16896
C_8	62.9648	71.8008

Table 4 shows the optimization results, i.e., the values for the regressors which will achieve maximum power.

TABLE 4: OPTIMIZATION RESULTS

Control	Lower bound	Upper bound	Empirical Max. relative	Theoretical Max. relative	Empirical Max. abs.
C_2	0.00815	0.0167	0.3634	0.150	0.0112
C_3	10.4632	11.1656	0.6074	0.95	10.8898
C_6	22.8760	33.7115	0.4391	0.80	27.6339
C_7	2.7561	3.1690	0.9185	0.60	3.13453
C_8	62.9648	71.8008	0.9359	0.75	71.2344
Power	37.0313	41.1406	0.8251	0.9999	40.4219

Sensitivity Coefficients: If we want to know by how much the response variable changes when we change the regressors by 10%, we will again be asked for the region in which this analysis should take place, see table 3:

TABLE 5: SENSITIVITY COEFFICIENTS -ACTIVE REGION

Control	Lower bound	Upper bound
C_2	0.0082	0.0167
C_3	10.4632	11.1656
C_6	22.8760	33.7115
C_7	2.7561	3.1690
C_8	62.9648	71.8008

Results are shown in Table 6:

TABLE 6: SENSITIVITY COEFFICIENTS

Control	Relative Sensitivity	Absolute Sensitivity	Abs. Change for 10% change in control
C_2	-0.0996	-47.8959	-0.0409
C_3	0.3966	2.3201	0.1630
C_6	0.1511	0.0573	0.0621
C_7	0.1226	1.2199	0.0504
C_8	0.8696	0.4044	0.3573

VII. Conclusions

On the path to finding optimal set points in a controlled industrial process with regard to power in a constrained operating region, two necessary steps were established: Approximating the process by a smooth response surface and finding an optimal point within the constraints and on the response surface. The approximation step requires tools from Machine Learning or Multivariate Statistics such as Tensor Flow or Generalized Linear Models, respectively. Two examples have been studied. The first example simply served to illustrate the extreme goodness of fit for Tensor Flow, even in the presence of steep steps in the response. The second example showed that Tensor Flow, on the data considered, tends to overfit. Therefore – and due to the fact that Tensor Flow, in contrast to a Generalized Linear Model – has no straightforward derivative computation interface – we proceeded with Generalized Linear Models, which approximate well and can be used to do nonlinear optimization and to compute sensitivity coefficients.

REFERENCES

[1] M. Abadi et al., Tensor Flow: Large Scale Machine Learning on Heterogeneous Distributed Systems (Preliminary White Paper, November 09, 2015). http://download.tensorflow.org/paper/whitepaper pdf ., (2015).

[2] B. Benesova, M. Kruzik, Weak Lower Semi continuity of Integral Functionals and Applications, SIAM Review, 59(4)(2017) 703-766

[3] J.L. Crassidis, J.L. Junkins, Optimal Estimation of Dynamic Systems. Chapman & Hall/CRC, Boca Raton (2004).

[4] D. E. Goldberg, Genetic Algorithms in Search, Optimization, and Machine Learning. Addison-Wesley, Reading (MA) (1989).

[5] I. Koch, Analysis of Multivariate and High-Dimensional Data Cambridge University Press, New York (2014) 148

[6] D. Kriese, Ein kleiner Ueberblick ueber Neuronale Netze, www.dkriesel.com/imprint, Rheinbach, Germany., (2018)

[7] F. Kuebler et al., Resource efficiency optimization of manufacturing processes using evolutionary computation: A turning case. Procedia CIRP 29(2015) 822-827, www.elsevier.com/locate/procedia

[8] Z. Michalewicz, How to Solve It: Modern Heuristics. Springer, Berlin-Heidelberg (2000).

[9] H. R. Myers, D.C. Montgomery, C.M. Anderson Cook, Response Surface Methodology. Wiley, New York (2009).

[10] K. Naeem et al., Process Optimization of a Local Steel Bar Manufacturing Firm Using SPC and ANOVA, International Journal of Engineering Research and Development,10 (7)(2014) 10-15.

[11] Klaus Neumann, Martin Morlock, Operations Research. Carl Hanser, Muenchen, Wien (1993) 337

[12] M. L. Pinedo, Planning and Scheduling in Manufacturing and Services. Springer, New York (2005).

[13] S. Sousa et al., Application of SPC and quality tools for process improvement, 27th International Conference on Flexible Automation and Intelligent Manufacturing, FAIM, 27-30 (2017) Modena, Italy, ELSEVIER, Science Direct, Procedia Manufacturing, www.sciencedirect.com.

[14] E. K. Tetteh, S. Rathilal, Application of Response Surface. Methodology (RSM) - Reduction of Industrial Wastewater Chemical Oxygen Demand. CBU International Conference on Innovations in Science and Education, Prague, (2017).

[15] M. Lapan, Deep Reinforcement Learning, Hands-On, Packt Publishing, Birmingham (2018).