CHARACTERISATION OF HOURLY TEMPERATURE OF A THIN-FILM MODULE FROM WEATHER CONDITIONS BY ARTIFICIAL INTELLIGENCE TECHNIQUES

M. Piliougine[1], L. Mora-López[2], J. Carretero[1] and M. Sidrach-de-Cardona[1]
[1]Dpto. de Física Aplicada II, Universidad de Málaga, 29071 Málaga, Spain
[2]Dpto. de Lenguajes y Ciencias de la Computación, Universidad de Málaga, 29071 Málaga, Spain
Telephone: +34952132772. Fax: +34952131355. Email: {michel, llanos, jecarretero, msidrach}@uma.es

ABSTRACT: The aim of this paper is the use and validation of artificial intelligence techniques to predict the temperature of a thin-film module based on tandem CdS/CdTe technology. The cell temperature of a module is usually tens of degrees above the air temperature, so that the greater the intensity of the received radiation, the greater the difference between these two temperature values. In practice, directly measuring the cell temperature is very complicated, since cells are encapsulated between insulation materials that do not allow direct access. In the literature there are several equations to obtain the cell temperature from the external conditions. However, these models use some coefficients which do not appear in the specification sheets and must be estimated experimentally. In this work, a support vector machine and a multilayer perceptron are proposed as alternative models to predict the cell temperature of a module. These methods allow us to achieve an automatic way to learn only from the underlying information extracted from the measured data, without proposing any previous equation. These proposed methods were validated through an experimental campaign of measurements. From the obtained results, it can be concluded that the proposed models can predict the cell temperature of a module with an error less than 1.5 °C.
Keywords: Multilayer Perceptron, Support Vector Machines, Thermal Modelling, Thin Film.

1 INTRODUCTION

Although the incident irradiance G is the most important parameter to determine the output power of a photovoltaic module, the cell temperature T_c has also a strong effect on its behaviour, but the directly measurement of this magnitude is not easy. Therefore, we need procedures to forecast this value from more readily available data such as the irradiance G, the air temperature T_a and the wind speed W_s. In the literature there are methods to estimate the cell temperature from these weather conditions based on mathematical expressions, where a few module-dependent parameters appear.

In Eq. (1) it can be seen the simplest method to calculate the cell temperature from the irradiance and air temperature [1]:

$$T_c = T_a + \frac{G}{800\,\mathrm{W/m^2}} \cdot \left(NOCT - 20\,^{\circ}\mathrm{C}\right) \quad (1)$$

This formula only requires one parameter known as *NOCT* (nominal operating cell temperature). This parameter is given by the manufacturers in the specification sheets, but it could also be estimated from measured data using a regression algorithm if more accuracy must be achieved. However, this method is very inaccurate and in practice its application is not worthy.

A first attempt to improve Eq. (1) could consists in determine the difference $T_c - T_a$ by a linear function of several significant magnitudes, such as the irradiance G, the air temperature T_a, the wind speed W_s and the relative humidity H_R, obtaining the following expression:

$$T_c = T_a + a \cdot G + b \cdot T_a + c \cdot W_S + d \cdot H_R \quad (2)$$

In Eq. (2) there are four free parameters (a, b, c and d) which can be determined by a fitting procedure over measured data using multivariate linear regression.

In the literature there are more complex equations to obtain the module temperature from the external conditions. Skoplaki and Palyvos [2] comprise the most relevant of these methods. For example, Servant [3] proposes the model expressed in Eq. (3) which uses as input variables G, T_a and W_s:

$$T_c = T_a + a \cdot G \cdot \left(1 + b \cdot T_a\right) \cdot \left(1 - c \cdot W_s\right) \quad (3)$$

In order to apply the method the three free parameters (a, b and c) are calculated for a specific module from measured data using least mean square regression in two steps: first a and b are fitted for data with $W_s < 1$ m/s; then c is estimated for data with $W_s \geq 1$ m/s using the previous calculated values for a and b.

Another method that is much cited in the literature is the exponential model proposed by King et al. [4] which is summarised in Eq. (4):

$$T_c = T_a + G \cdot \mathrm{e}^{(a + b \cdot W_S)} \quad (4)$$

In addition to these traditional models, the emergence of artificial intelligence and data mining has incorporated alternative methods such as artificial neural networks and support vector machines. Both of these paradigms allow us to build an operational model by learning only from the underlying information inside the measured data, without having any previous analytical model. The objective of our work is the empirical application of a support vector machine (SVM) and a multilayer perceptron (MLP) to predict the cell temperature T_c of a CdS/CdTe module.

The rest of the paper is organised as follows: In Section 2 the theoretical basis of the proposed models are provided. Section 3 is bound to detail the experiments carried out in order to acquire the dataset to train and validate the proposed models. The results of the previous models and proposed ones are compared in Section 4. Finally, in Section 5, the main conclusions that can be derived from this work are highlighted.

2 THEORETICAL BACKGROUND

2.1 Support Vector Regression

On the one hand, we can use support vector machines, which refer to a set of supervised learning techniques firmly grounded on the statistical learning theory from Vapnik [5]. At the beginning, SVMs were introduced to solve classifications problems, but later these ideas were adapted to manage with datasets where the output is a continuous variable (support vector regression, SVR) [6].

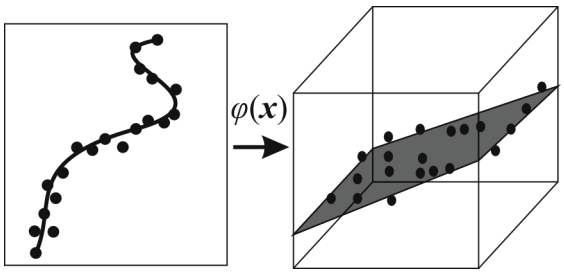

Figure 1: Transformation from input space to a higher dimensional space where the dataset is linear

Although the most basic SVM is only able to manage with linearly datasets, non-linear problems can be addressed by means of a kernel transformation (see Fig. 1). The underlying function from the measured data could be non-linear, but it is possible to apply a transformation φ which maps the points from the input space into a higher dimensional space, in which the transformed points could be fitted by a linear hyperplane. As we will see later, in the procedure of the SVM this mapping function will never be applied independently. In fact, given two points x and y, the SVM only needs to know the inner product of $\varphi(x)$ and $\varphi(y)$, that we will note as $K(x,y)$ and it is known as a kernel function. Therefore, we do not need to know the exact form of the function φ because we will never apply that transformation φ to a specific point x. In this problem, we have used as kernel a Gaussian radial basis function, provided by the following expression:

$$K(x,y) = \varphi(x)\cdot\varphi(y) = \mathrm{e}^{\left(\gamma\|x-y\|^2\right)} \quad (5)$$

where γ is an parameter that must be adjusted between a range in order to achieve the best performance.

Suppose that the dataset is composed off m samples $(z_1, y_1), (z_2, y_2),\ldots,(z_m, y_m)$ where $z_i \in \Re^N$ and $y_i \in \Re$.

Initially, in order to improve the results it is useful to apply a previous normalisation of the input data based on the mean μ and the standard deviation σ of each variable using the next equation:

$$x_i[j] = \frac{z_i[j] - \mu[j]}{\sigma[j]} \quad (6)$$

where i=1,...,m and j=1,...,N (N is the number of the number of input variables)

The basic support vector machine tries to find a function of the following form:

$$f(x) = \omega\cdot x + b \quad (7)$$

This is the hyperplane which models the dataset (see Fig. 2). Since measurements could be affected by noise, a small deviation ε is accepted. Therefore, a ε-region is defined between the two parallel hyperplanes given by the following equations (and also parallel to the solution hyperplane):

$$y = (\omega\cdot x + b) + \varepsilon$$
$$y = (\omega\cdot x + b) - \varepsilon \quad (8)$$

Each measured data point outside the region is penalised if the difference with the proposed model is upper or under the threshold ε. Points upper the ε-region are penalised with a ξ_r term (the point is over the ε-region) and points under the ε-region are penalised with a ξ_s^* term (the point is under the ε-region).

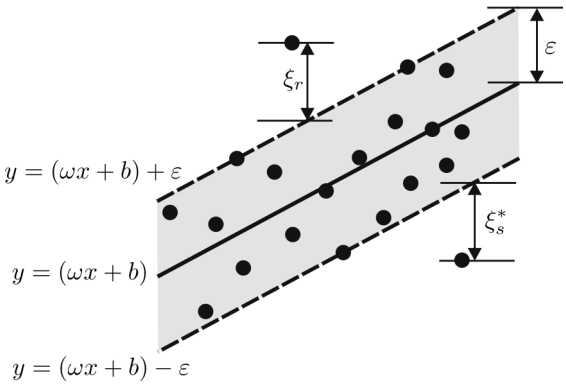

Figure 2: Hyperplane around the data point cloud. Points outside the ε-region are penalised with a ξ_i term

Finally, the SVM to solve a regression problem can be seen as an optimization problem as the one proposed in Eq. (9), where a specific objective function must be minimised taking into account a set of restrictions that ensure that the points beyond the region are penalised:

$$\min.\ \frac{1}{2}K(\omega,\omega) + C\sum_{i=1}^{m}\left(\xi_i + \xi_i^*\right)$$

s.t.

$$y_i - (K(\omega,x_i) + b) \le \varepsilon + \xi_i \qquad i = 1,\ldots,m$$
$$(K(\omega,x_i) + b) - y_i \le \varepsilon + \xi_i^* \qquad i = 1,\ldots,m$$
$$\xi_i, \xi_i^* \ge 0 \qquad i = 1,\ldots,m \quad (9)$$

In the objective function, there is a term with the sum of the ξ_i elements that must be minimised. This means that we are trying to find a hyperplane very close to the point cloud to be modelled. However, in order to avoid overfitting, an additional term, that measures the complexity of the model, is taken into account (first term of objective function). The parameter C is used to adjust the trade-off between simplicity of the model and accuracy over the training data set. Adjusting this parameter we can improve the generalisation power of the model. In practice, the optimisation problem that appears in Eq. (9) must be solved for each different combination of the adjustable parameters, which in our case are γ and C. This iterative process is implemented in a library for Matlab™ called LIBSVM developed by Chang and Lin [7] from the Department of Computer Science and Information Engineering of the National Taiwan University.

2.2 Multilayer Perceptron

On the other hand, artificial neural networks could be powerful tools to perform models based only on measured data. The most successful type of neural network is the multilayer perceptron. The operation of this type of neural networks has two different phases.

In the training phase, inputs and measured outputs are presented to the neural network, being the weights adjusted until reaching a mean square error between the network output and the measured output sufficiently low to consider that the MLP has been trained. Then, once the network has been trained, it is able to calculate the estimated output by only introducing the inputs. The algorithm used in to adjust the weights taking into account the error is the backpropagation rule [8], which propagates the error/adjust from the output layer to the input layer.

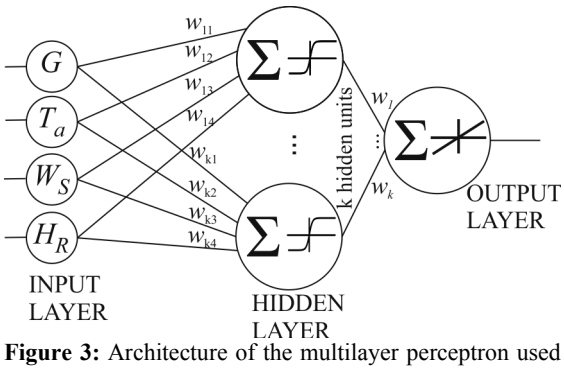

Figure 3: Architecture of the multilayer perceptron used in this work

The multilayer perceptron proposed to predict the cell temperature in this work is depicted in Fig. 3. It has an input layer, one hidden layer and an output layer. There is a neuron in the input layer for each input magnitude, that is, we have four neurons in the input layer:

- Incident irradiance G
- Air temperature T_a
- Wind speed W_s
- Relative humidity H_R

Each neuron of one layer is connected to each neuron of the next layer. Each connection has an associated weight in such a way the output of each input neuron is multiplied by this weight and then each neuron in the hidden layer sums all its weighted inputs and applies a transfer function f to the result:

$$hidden_i = f\left(\sum_{j=1}^{4} w_{ij} input_j\right) = f\left(w_{i1}G + w_{i2}T_a + w_{i3}W_S + w_{i4}H_R\right) \quad (10)$$

In this work, *tanh* (Eq. 11) is considered as the transfer function for all the units in the hidden layer:

$$f(x) = tanh\,(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} \quad (11)$$

The process performed by the neuron in the output layer is similar using the identity as the transfer function:

$$output = \sum_{n=1}^{k} w_n hidden_n \quad (12)$$

The neural network model was trained using the "Neural Networks" toolbox for Matlab™ 2013b [9].

3 EXPERIMENTAL SETUP

Two datasets of experimental measurements were acquired using a thin-film module (see its specifications in Table I). The training dataset (measurements to fit the models) was acquired for a complete meteorological year, from January 2011 to December 2011. However, additional measurements were used (30 randomly selected days from 2012) to build a dataset to test the models. The instantaneous values have been averaged through intervals of one hour to obtain hourly values. The outputs of each model over the test dataset are compared with the measures of the cell temperature T_c and the overall relative mean error of each method is calculated. Ideally having a prototype of the module with an internal temperature sensor, but for this work, we have approximated the cell temperature T_c to the temperature of a sensor at the back surface of the module.

Table I: Specifications of the thin-film module

Technology	CdS/CdTe
Manufacturer	First Solar
Model	FS-270
No. of cells	116
Electrical values at STC G=1000 W/m^2, T_c = 25 °C, solar spectrum AM1.5	
I_{SC} (A)	1.23
V_{OC} (V)	88.00
P_M (W)	70.00
I_M (A)	1.07
V_M (V)	65.50
Efficiency (%)	9.22
Electrical values at G=800 W/m^2, T_c = 20 °C solar spectrum AM1.5 and wind speed W_S = 1 m/s	
I_{SC} (A)	1.01
V_{OC} (V)	81.80
P_M (W)	52.60
I_M (A)	0.86
V_M (V)	61.40
NOCT (°C)	45

Table II: Estimation of the parameters for the classical methods and their respective RMSE error

Eq	*NOCT*	*a*	*B*	*c*	*D*	RMSE
1	45.0 °C					8.5 °C
1*	42.8 °C					7.9 °C
2		0.0278	0.0387	-1.5550	0.0147	1.9 °C
3		0.0320	-0.0100	0.0029		2.1 °C
4		-3.4737	-0.1066			1.9 C

* Idem to 1 using a *NOCT* value derived from measured data instead the one provided by the manufacturer

4 RESULTS AND DISCUSSION

The traditional models proposed has been fitted using the training dataset (measurements from 2011). Table II shows the different values for the intrinsic parameters relative to each method. Then, using the test dataset (30 selected days from 2012), the RMSE of each method has been estimated and the results are shown in the last column.

NOTE: The *NOCT* method can be used using the *NOCT* value given by the manufacturer (row labelled as 1) or using an experimental value calculated from the training dataset (row labelled as 1*).

SVM model has two parameters to be tuned: the trade-off parameter C and the value of γ used in the kernel function. The training procedure is repeated for each different combination of C and γ. As can be seen, the best result (RMSE = 1.5 °C) is obtained using $C = 10^5$ and $\gamma = 10^{-4}$. In Table III the RMSE error for each combination of the adjustable parameters is shown.

Table III: Root mean squared error for cell temperature (measured in °C), using SVM model over the test dataset for each combination of the adjustable parameters

	$\gamma=10^{-8}$	$\gamma=10^{-7}$	$\gamma=10^{-6}$	$\gamma=10^{-5}$	$\gamma=10^{-4}$	$\gamma=10^{-3}$
$C=10^{1}$	10.6	10.5	9.8	4.4	1.8	1.7
$C=10^{2}$	10.5	9.8	4.4	1.8	1.8	1.6
$C=10^{3}$	9.8	4.4	1.8	1.8	1.6	1.6
$C=10^{4}$	4.5	1.8	1.8	1.8	1.6	1.6
$C=10^{5}$	1.9	1.8	1.8	1.7	**1.5**	1.6
$C=10^{6}$	2.1	1.8	1.9	1.7	1.6	2.7

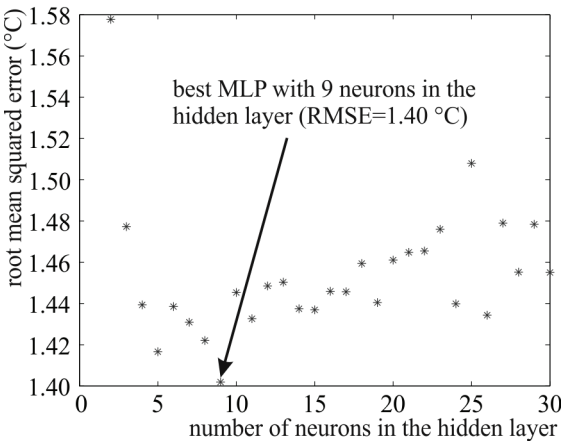

Figure 4: Root mean squared error for cell temperature over the test dataset for different number of neurons in the hidden layer

In order to find the best number of neurons in the hidden layer for this problem, the multilayer perceptron must be trained with different number of neurons in the hidden layer and for a fixed number, the training should repeated several times getting the trained network with the smallest error (the result of each training is different, even with the same number of neurons, due to the random initialisation of the MLP). In our case, the best results (RMSE=1.4 °C) are obtained with 9 neurons in the hidden layer as can be seen in Fig. 4.

5 CONCLUSIONS

In this work two alternative models from the area of artificial intelligence are proposed as an alternative ways to classical models to forecast the cell temperature of a module using easily available meteorological measurements as inputs. Both of them, the support vector machine and the multilayer perceptron, allow us to predict the cell temperature without any previous analytical model. Both techniques have been validated using an experimental campaign of measurements and the results have been compared with the obtained ones by some classical methods from the literature. Whereas the RMSE of the previous models ranges between 8.5 °C and 1.9 °C, using these artificial intelligence techniques we can estimate the cell temperature of a module with less error (around 1.5 °C in both cases). Therefore, these techniques should be incorporated to photovoltaic simulations tools in the market in addition to the previous classical methods.

ACKNOWLEDGEMENTS

This work was carried out with the financial support of the "Junta de Andalucía" (grants No. P10-TIC-6441 and No. P11-RNM-7115). In addition, some expenses were covered by "Universidad de Málaga. Campus de Excelencia Internacional Andalucía Tech".

REFERENCES

[1] L. Castañer, S. Bermejo, T. Markvart, K. Fragaki, Energy production by a PV array. Chapter IIIa-1 in "Practical Handbook of Photovoltaics. Fundamentals and Applications". T. Markvart, L. Castañer (Eds). Elsevier (2003) 517. ISBN: 978-1-85617-390-2.

[2] E. Skoplaki, J.A. Palyvos, Renewable Energy 34 (2009) 23.

[3] J.M. Servant, Proceedings 9th Biennial Congress of the International Solar Energy Society Intersol 85 (1985) Vol. 3, 1640.

[4] D.L. King, W.E. Boyson, J.A. Kratochvill, Sandia Laboratory Technical Report SAND2004-3535 (2004).

[5] V.N. Vapnik, Statistical Learning Theory. Wiley-Interscience (1998). ISBN: 978-0471030034.

[6] A.J. Smola, B.A. Schölkopf, Statistics and Computing 14 (2004) 199.

[7] S. Haykin, Multilayer perceptron in Neural Networks. A comprehensive foundation". Second Edition. Prentice Hall (1999) 156. ISBN: 0-13-273350-1.

[8] C.C. Chang, C.J. Lin, ACM Transactions Intelligent Systems and Technology 2 (2011) 27:1.

[9] M.H. Beale, M.T. Hagan, H.B. Demuth, Neural Network Toolbox™. User's Guide. R2013b. Mathworks.