Gradient boosting models for photovoltaic power estimation under partial shading conditions

Nikolaos Nikolaou[1], Efstratios Batzelis[2] and Gavin Brown[1]

[1] School of Computer Science, University of Manchester,
Kilburn Building, Oxford Road, Manchester, M13 9PL, UK
{nikolaos.nikolaou,gavin.brown}@manchester.ac.uk
[2] Department of Electrical and Electronic Engineering, Imperial College London,
Exhibition Road, London, SW7 2AZ, UK
e.batzelis@imperial.ac.uk

Abstract. The energy yield estimation of a photovoltaic (PV) system operating under partially shaded conditions is a challenging task and a very active area of research. In this paper, we attack this problem with the aid of machine learning techniques. Using data simulated by the equivalent circuit of a PV string operating under partial shading, we train and evaluate three different gradient boosted regression tree models to predict the global maximum power point (MPP). Our results show that all three approaches improve upon the state-of-the-art closed-form estimates, in terms of both average and worst-case performance. Moreover, we show that even a small number of training examples is sufficient to achieve improved global MPP estimation. The methods proposed are fast to train and deploy and allow for further improvements in performance should more computational resources be available.

Keywords: Gradient boosting, Solar energy, Photovoltaic (PV) system, Maximum power point (MPP), Partial shading, Machine learning

1 Introduction

The photovoltaic (PV) penetration has remarkably increased worldwide the last decades, with several applications ranging from rooftop and building-integrated systems to MW-scale power plants. Especially in the former cases installed in urban environments, the operating conditions are often non-ideal with surrounding obstacles casting shadows on the PV system, leading to non-uniform illumination. Under such conditions, the *power-voltage (P-V) characteristic curve* presents several local *maximum-power-points (MPPs)*, a situation that hinders the effective tracking of the *global MPP* which provides the maximum power output. This phenomenon, commonly referred to as *partial shading*, has gathered the interest of researchers lately due to its non-linear nature and strong effect on the energy yield of the PV system.

There are several PV models in the literature, varying in terms of accuracy, complexity and scope of application, classified into two generic categories: the

circuit-based models and the *heuristic methods*. The former models have strong theoretical foundation and can provide any needed information, but require tedious simulations of complicated circuits. The latter approaches, on the other hand, are simpler and provide directly the global MPP, but generally suffer from lower accuracy [2]. In PV energy yield studies, there is a need for a fast and reliable method to easily calculate the maximum power of the PV system at numerous different scenarios; in these applications, the heuristic alternatives seem more appropriate.

According to [2], these methods can be further classified to: (a) *Empirical efficiency-based models* which are derived from empirical observations and have simple formulation, but exhibit moderate accuracy due to their weak theoretical background [4,8,18]; (b) *Explicit mathematical equations* that are based on the equivalent circuit and provide all local MPPs, presenting good average accuracy in principle, yet occasionally high estimation errors [3,13,17]; and (c) *Artificial Neural Networks (ANN)* trained on the actual data of the study-case PV system which provide adequately fast execution and probably the best estimations [9,14].

Even though several applications of machine learning algorithms are reported in the literature for the simple uniform illumination case [10], the relevant research for partial shading conditions is still limited to only the two aforementioned studies [9, 14]. As an alternative, we investigate in this paper *gradient boosting* [11, 12] models trained on data generated from the equivalent circuit. In the following, three gradient boosting models are implemented and evaluated across a wide range of operating conditions, concluding to very interesting and promising results. This is the first paper in the literature to apply this method in PV energy forecasting under partial shading.

2 PV power generation under partial shading

2.1 Main concepts & examples

The smallest commercially available PV unit is the *module* (or *panel*). Usually several PV modules are connected in series to form a *string* in order to produce appropriate levels of *voltage* and *power* output. A typical PV string composed by 12 modules is depicted in Fig. 1 [TOP], operating under partial shading conditions (common case of two different *irradiance* levels). Based on the notation of [1,3], 10 out of 12 modules are unshaded and illuminated at full irradiance $G = 800W/m^2$ (or 0.8 *per unit* (*pu*)), whereas the 2 remaining shaded modules are subject to half the irradiance (*shading ratio* $s = 50\%$); this corresponds to a *shadow extent* of $n_{sh} = 2/12 = 0.17$. Common *temperature* $T = 45\ ^\circ C$ is assumed across the entire string, as usually the temperature difference between the shaded and unshaded part is small [3]. In the general case, the *operating conditions* of a PV string partially shaded at two different irradiance levels are uniquely identified by the vector $[G, T, s, n_{sh}]$, which in this example equals to $[0.8, 45, 0.5, 0.17]$.

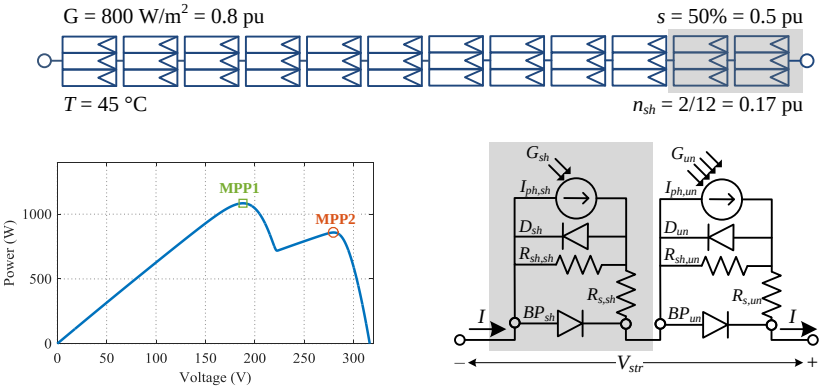

Fig. 1. [TOP] Topology of a PV string consisting of 12 PV modules connected in series, 2 of them being shaded at 50%. [BOTTOM-LEFT] Respective P-V curve indicating the 2 local maxima: MPP1 and MPP2. [BOTTOM-RIGHT] Electrical equivalent circuit.

The P-V characteristic at partial shading conditions presents up to two local MPPs (or power peaks), as illustrated in Fig. 1 [BOTTOM-LEFT] for this particular scenario. These two MPPs are studied and characterized as *MPP1* and *MPP2* in [3, 16], as they exhibit different behaviour and dependence on the operating conditions. Only one of them or both (most often) may appear and the power peak that provides the maximum power is denoted as the *global MPP*. This corresponds to either MPP1 or MPP2 depending on the scenario (MPP1 in Fig. 1), so determining its power P_{max} and voltage $V_{P_{max}}$ is not a trivial task.

To generate this P-V curve, one has to adopt an *electrical equivalent circuit*, modelled and simulated in appropriate software (e.g. MATLAB/Simulink). According to the PV modelling theory, the equivalent circuit of this system is described by Fig.1 [BOTTOM-RIGHT] [1,3]. Each part of the string, shaded and unshaded, is modelled using the single-diode equivalent and a bypass diode, the respective circuit parameters being 11 in total and strongly dependent on the operating conditions [1]. Calculating the maximum power according to the circuit-based models, involves the laborious steps of (i) extracting these parameters at STC[3], (ii) translating their values to the actual conditions and (iii) simulating a highly non-linear circuit. These procedures are tedious and time-consuming, thus they are more suited for research, rather than practical applications [2].

2.2 Closed-form equations

Among the heuristic methods mentioned in the Introduction, the closed-form equations introduced in [3] and further improved in [17] are considered here, as they are shown to outperform other explicit mathematical approaches. These are probably the best candidates to be compared against the machine learning models investigated in this paper. For completeness, these expressions are given

[3] *Standard Test Conditions*: 1000 W/m^2 irradiance, 25 $°C$ temperature & 1.5 air mass.

in Eq. (1) –more details can be found in [3,17]. Given the conditions $[G, T, s, n_{sh}]$, the *system structural characteristics* (PV module datasheet information V_{mp0}, I_{mp0}, V_{oc0}, α_{Imp}, β_{Vmp}, β_{Voc}, total number of modules N_{tot}, voltage drop on the bypass diode $\Delta V_D = 1.0\ V$) and the empirical coefficient $\lambda = 0.06$:

$$
\begin{aligned}
MPP1: &\begin{cases} V_1 = N_{tot}[(1-n_{sh})V_{mp}^T + n_{sh}\Delta V_D] \\ I_1 = GI_{mp}^T \\ P_1 = V_1 I_1 \end{cases} \\
MPP2: &\begin{cases} V_2 = N_{tot}[(1-n_{sh})(sV_{mp}^T + (1-s)V_{oc}^T) + n_{sh}V_{mp}^T] \\ I_2 = sI_{mp}^T[1+\lambda(1-nsh)] \\ P_2 = V_2 I_2 \end{cases}
\end{aligned}
\tag{1}
$$

where $I_{mp}^T = I_{mp0}[1 + \alpha_{Imp}(T-25)]$, $V_{mp}^T = V_{mp0}[1 + \beta_{Vmp}(T-25)]$ and $V_{oc}^T = V_{oc0}[1 + \beta_{Voc}(T-25)]$. Although these expressions yield in general fairly acceptable average accuracy, in some cases they exhibit high errors, as observed in [3] and further verified in this paper. This motivates an alternative approach with machine learning models, which, once trained, could give a better estimate of the global MPP in a similarly straightforward and cost-efficient way.

3 Gradient boosting models for MPP prediction

We simulate the behaviour of the circuit of Fig. 1 generating examples of input $[G, T, s, n_{sh}]$ and output $[P_1, V_1, P_2, V_2, P_{max}, V_{P_{max}}]$ vector pairs –note that we also include the 'intermediate' outputs of the local MPP1 & MPP2 power-voltage pairs, (P_1, V_1) & (P_2, V_2), respectively. Using these examples, we train regression models to predict the values P_{max} & $V_{P_{max}}$. The goal is to better approximate these quantities than the closed-form estimates of Eq. (1).

In machine learning terminology, this is a *multi-output regression problem*. Normally, we would like to take into account the *covariance* of the targets in our model. In this work, we start with the simpler approach of modelling the individual targets *independently*[4], i.e. we train *multiple single-output regressors*.

There are many possible choices of regression algorithms to use. As a reliable off-the-shelf learning meta-algorithm, *gradient boosting* [11, 12] was chosen, as its variants have proven very successful in large scale experimental comparisons of learning algorithms [6, 10], industrial applications [5, 19] and competitions [7] alike –often outperforming other powerful *kernel-based* or *deep learning* methods.

More specifically, deep learning methods tend to be outperformed by gradient boosting on tabular data –as is the case here. The problem here is not expected to be particularly noisy; another reason to expect boosting to be good at modelling it. Compared to kernel-based methods, like *Support Vector Machines (SVMs)*,

[4] Initial attempts at exploiting interactions between P_{max} & $V_{P_{max}}$ by first predicting the value of one and then using it to predict the other, yielded worse results than assuming independence. Their further investigation is left for future work.

that shift the computational burden from the number of features to the number of examples, gradient boosting is computationally preferable in our task, since the feature space is considerably smaller than the number of examples.

Furthermore, gradient boosting constructs models that can be more easily interpreted than those of the aforementioned methods. For example, allowing for better approximations to Eq. (1) to be derived as closed-form formulas. Finally and perhaps more importantly, gradient boosting has relatively few *hyperparameters* to tune compared to the aforementioned methods, the most important being the *ensemble size* M and the *complexity* of its *base learner*. Very recent research [20] suggests that the higher both of these are, the better the ensemble's generalization behaviour (although the computational cost increases).

Gradient boosting constructs an ensemble of M *additive components* (base learners) in a forward stagewise manner; it allows for the optimization of arbitrary differentiable *loss functions* (here *quadratic loss*). In each stage a base learner (here *regression tree*) is fit on the negative gradient of the given loss function. The final output, i.e. the *estimate* $\hat{X}$ of the target variable in question[5] in each case, will be a *weighted linear combination* $\hat{X} = \sum_{m=1}^{M} a_m f_m(G, T, s, n_{sh})$ of the M base learners' outputs, each of which –being a regression tree– is a *piecewise linear* function $f_m(G, T, s, n_{sh})$ of the inputs. The *weights* a_m and the additive components f_m are learned form the data.

In the remainder of this section, we shall discuss the three different models we will compare against the closed-form estimation. The models will differ in what target outputs $\hat{X}$ they estimate, and how they combine them to predict the global MPP $(P_{max}, V_{P_{max}})$.

3.1 Direct modelling of MPP

First we take a direct approach to learning a model for predicting the global MPP $(P_{max}, V_{P_{max}})$. We simply train a regressor to map input vectors $[G, T, s, n_{sh}]$ to P_{max} and another to map them to $V_{P_{max}}$ independently, without making use of the local MPP1 and MPP2. We will henceforth refer to this as the '*Direct*' approach. Note that as the two constituent regression tasks are independent, they can be parallelized to reduce the computational cost.

3.2 Stagewise modelling of MPP

Next we take a 2-stage approach for predicting the global MPP. We train four regressors: one to map input vectors $[G, T, s, n_{sh}]$ to each of the 'intermediate outputs' P_1, V_1, P_2 & V_2. In other words, we first model the MPP1 and the MPP2 independently (and in each case P_i independently of its corresponding V_i). Then, we predict P_{max} & $V_{P_{max}}$ by

$$global\ MPP: \begin{cases} P_{max} = max\{P_1, P_2\} \\ V_{P_{max}} = \{V_{i^*} : i^* = \arg\max_{i \in \{1,2\}} P_i\}. \end{cases} \tag{2}$$

[5] As we will see in this section, $\hat{X}$ can be an estimate of P_{max} or $V_{P_{max}}$, or of any of the 'intermediate outputs' –P_1, V_1, P_2, V_2– depending on the method.

Note that in some scenarios a single power peak appears (MPP1 or MPP2), rather than both. In these cases, we disregard these examples for the purpose of training models to predict P_2 & V_2 or P_1 & V_1, respectively. By this approach, which we shall call '*Stagewise*', we encode more domain knowledge in our model about the structure of the targets: there exists at least one MPP and at most two, each of which has different characteristics and the global MPP is the maximum of the existing ones. Again, since the four constituent regression tasks are independent, they can be parallelized to reduce computational costs.

3.3 Classifier-assisted stagewise modelling of MPP

As we will see in the experimental section, the *Stagewise* approach achieves very low error on the intermediate estimates of P_1, V_1, P_2 & V_2 and also on P_{max}. Yet, its estimates of $V_{P_{max}}$ –even though much better than those of both the closed-form ones and the *Direct* ones– seem to not be on par with those of all other target quantities. This seems to stem from situations in which P_1 & P_2 are very close. When this happens, a small estimation error on either of these can lead us to select the wrong P_i as the P_{max}. The resulting error on P_{max} will still be small (the values of P_1 & P_2 being very close), but by choosing the wrong V_i as the $V_{P_{max}}$, the error on $V_{P_{max}}$ is high.

Motivated by this, the final approach we shall examine is a variant of the *Stagewise* method, where the selection of the global MPP $(P_{max}, V_{P_{max}})$, is not based on the comparison of the predicted values of P_1 & P_2, but is instead decided by a *binary classifier* trained to predict whether an input vector $[G, T, s, n_{sh}]$ leads to MPP1 or MPP2 being the global MPP. To get the label of each example, it is sufficient to compare if $P1 > P2$, in which case MPP1 is the global MPP, otherwise it is MPP2. Based on the classifier's prediction, we pick the appropriate P_i & V_i pair calculated by the regressors as the P_{max} & $V_{P_{max}}$, respectively. The classification subtask is accomplished by training a gradient boosting ensemble of *decision trees*[6]. We call this method '*StagewiseC*'. As with the previous approaches, to reduce computational costs we can parallelize the training of all 5 predictors.

4 Experimental Investigation

4.1 Experimental Setup

We generated 94905 examples from the circuit shown in Fig. 1, considering irradiance $G = [0.1 : 0.05 : 1.0]$ *pu*, temperature $T = [-5 : 5 : 65]$ $^\circ C$, shade ratio $s = [0.1 : 0.1 : 0.9]$ and shade extent $n_{sh} = [0 : 1/36 : 1]$. The examples consist of

[6] Denoting MPP1 & MPP2 with '1' & '−1', respectively, the classifier's prediction is of the form $\hat{H} = sign\big[\sum_{m=1}^{M} a_m h_m(G, T, s, n_{sh})\big] \in \{-1, 1\}$, where $h_m(G, T, s, n_{sh}) \in \{-1, 1\}$ is the prediction of the base learner added on the m-th round and a_m its voting weight, both the learner and a_m being the learned parameters of the model.

input vector $[G, T, s, n_{sh}]$ and output vector $[P_1, V_1, P_2, V_2, P_{max}, V_{P_{max}}]$ pairs – where we also included the 'intermediate' outputs of MPP1 and MPP2. On each example, we also provide the closed-form estimate for P_1, V_1, P_2 & V_2, along with the resulting final P_{max} and $V_{P_{max}}$ by evaluating Eq. (1). We compare the estimate of the learned models described in Section 3 to the closed-form ones. As ground truth, we consider the true outputs of the circuit.

All learners, unless otherwise specified, were gradient boosting ensembles of size $M = 1000$ trained on a 75% of the data chosen uniformly at random and evaluated on the remaining 25%. Regression trees were chosen as base learners for the regression subtasks, and for the classification step of *StagewiseC*, we used a decision tree. In all cases, the trees had a maximum depth of 3. All remaining hyperparameters were left to the default values of *scikit-learn*[7].

To evaluate the quality of the estimates (both learned models' and closed-form ones), for each target quantity X, we calculate on the test set the *normalized root-mean squared error* of the predictions $\hat{X}$,

$$NRMSE = \sqrt{\sum_n (X_n - \hat{X_n})^2} \Big/ \mu_X,$$

that uses as normalization factor the mean of quantity X on the test set, μ_X.

To get a more robust measure of central tendency than the NRMSE, we also provide the *median* of the *normalized absolute error* on each example,

$$NAE_n = |X_n - \hat{X_n}|/\mu_X.$$

We also report the *maximum* NAE as a measure of worst-case performance and *percentiles* of the NAE (95th, 99th, 99.9th & 99.99th), the purpose of which is to allow us to observe the frequency of large estimation errors.

We perform 10 train-test splits, shuffling the dataset before each split and report averages and 95% *confidence intervals* for all evaluation measures estimated. Finally, we also provide some characteristic *learning curves* to demonstrate how the quality of estimates of each model changes with the size of the training set.

4.2 Experimental Results

Results for intermediate problems As we saw, the two stagewise models (*Stagewise* & *StagewiseC*) use as intermediate steps the predictions of four regressors (predicting P_1, V_1, P_2 & V_2). In addition, *StagewiseC* predicts whether the global MPP is MPP1 or MPP2 by the use of a classifier. We first present results that showcase the predictive performance of these learners, before moving on to the results on the final P_{max} & $V_{P_{max}}$ predictions. On Table 1 we compare the approximation on P_1 & P_2 of the two stagewise models to that of the closed-form estimates of Eq. (1). On Table 2 we do the same for V_1 & V_2.

[7] http://scikit-learn.org/stable/

Table 1. Statistics of the errors of P_1 & P_2 estimates under the closed-form solutions and the two stagewise models. Averages and 95% CIs across 10 runs shown. Results rounded to four decimal places. Lowest values shown in bold.

	P_1		P_2	
	Closed Form	Learned Model	*Closed Form*	Learned Model
NRMSE	0.0388 ± 0.0006	$\mathbf{0.0120 \pm 0.0002}$	0.0475 ± 0.0002	$\mathbf{0.0098 \pm 0.0002}$
Median NAE	0.0084 ± 0.0000	$\mathbf{0.0071 \pm 0.0002}$	0.0214 ± 0.0001	$\mathbf{0.0061 \pm 0.0001}$
95th %ile NAE	0.0394 ± 0.0002	$\mathbf{0.0244 \pm 0.0005}$	0.1065 ± 0.0007	$\mathbf{0.0198 \pm 0.0004}$
99th %ile NAE	0.1413 ± 0.0066	$\mathbf{0.0350 \pm 0.0012}$	0.1659 ± 0.0009	$\mathbf{0.0275 \pm 0.0005}$
99.9th %ile NAE	0.5193 ± 0.0117	$\mathbf{0.0500 \pm 0.0010}$	0.2207 ± 0.0021	$\mathbf{0.0365 \pm 0.0007}$
99.99th %ile NAE	0.6608 ± 0.0081	$\mathbf{0.0693 \pm 0.0063}$	0.2465 ± 0.0021	$\mathbf{0.0440 \pm 0.0021}$
Maximum NAE	0.7171 ± 0.0241	$\mathbf{0.0812 \pm 0.0084}$	0.2551 ± 0.0027	$\mathbf{0.0464 \pm 0.0023}$

Table 2. Statistics of the errors of V_1 & V_2 estimates under the closed-form solutions and the two stagewise models. Averages and 95% CIs across 10 runs shown. Results rounded to four decimal places. Lowest values shown in bold.

	V_1		V_2	
	Closed Form	Learned Model	*Closed Form*	Learned Model
NRMSE	0.0478 ± 0.0007	$\mathbf{0.0041 \pm 0.0001}$	0.0352 ± 0.0001	$\mathbf{0.0047 \pm 0.0000}$
Median NAE	0.0127 ± 0.0000	$\mathbf{0.0020 \pm 0.0000}$	0.0147 ± 0.0001	$\mathbf{0.0020 \pm 0.0000}$
95th %ile NAE	0.0342 ± 0.0001	$\mathbf{0.0076 \pm 0.0002}$	0.0797 ± 0.0005	$\mathbf{0.0094 \pm 0.0001}$
99th %ile NAE	0.2554 ± 0.0078	$\mathbf{0.0118 \pm 0.0004}$	0.1158 ± 0.0008	$\mathbf{0.0174 \pm 0.0003}$
99.9th %ile NAE	0.5403 ± 0.0063	$\mathbf{0.0269 \pm 0.0038}$	0.1398 ± 0.0004	$\mathbf{0.0317 \pm 0.0009}$
99.99th %ile NAE	0.6326 ± 0.0046	$\mathbf{0.0756 \pm 0.0055}$	0.1741 ± 0.0112	$\mathbf{0.0565 \pm 0.0066}$
Maximum NAE	0.6596 ± 0.0062	$\mathbf{0.0880 \pm 0.0036}$	0.2101 ± 0.0081	$\mathbf{0.0797 \pm 0.0079}$

As we can see, the learned model estimates for each of the four quantities are considerably better than the closed-form ones. This is not only true as a central tendency (lower NRMSE, lower median NAE), but also holds for the worst case (lower maximum NAE). Inspecting the NAE percentile results given, we can also conclude that only a very small number of the errors on the estimates are high.

Next, on Table 3, we can inspect the confusion matrix of the classifier used by *StagewiseC* for predicting whether the global MPP is MPP1 or MPP2. We can see that it is very accurate on both classes.

Table 3. Confusion matrix for the classifier used by *StagewiseC* for predicting whether the global MPP corresponds to MPP1 or MPP2. Averages and 95% CIs across 10 runs shown. Results rounded to one decimal place. The classifier is very accurate on both classes, having an average accuracy of 99.74% with an average F_1-score of 0.998 and very low variance across runs. The data is slightly skewed towards MPP1 being the global MPP in approximately 58.92% of the cases.

		Prediction	
		MPP1	MPP2
Truth	MPP1	13945.5 ± 24.8	33.5 ± 2.2
	MPP2	27.8 ± 3.6	9720.2 ± 24.3

Results for final predictions We can expect that these results on the intermediate prediction tasks will translate to better estimates than the closed-form methods for P_{max} & $V_{P_{max}}$ under the stagewise models. Next, in Tables 4 & 5 we present the results of each model for predicting the global MPP.

Table 4. Statistics of the errors of P_{max} estimates under the different models. Averages and 95% CIs across 10 runs shown. Results rounded to four decimal places. Lowest –and tied for lowest, in the sense of overlapping CIs– values shown in bold.

	Closed Form	Learned Models		
		Direct	*Stagewise*	*StagewiseC*
NRMSE	0.0300 ± 0.0001	0.0213 ± 0.0003	$\mathbf{0.0093 \pm 0.0001}$	0.0308 ± 0.0044
Median NAE	0.0159 ± 0.0001	0.0126 ± 0.0002	$\mathbf{0.0056 \pm 0.0001}$	$\mathbf{0.0056 \pm 0.0001}$
95th %ile NAE	0.0645 ± 0.0004	0.0431 ± 0.0007	$\mathbf{0.0189 \pm 0.0003}$	$\mathbf{0.0189 \pm 0.0003}$
99th %ile NAE	0.1038 ± 0.0006	0.0646 ± 0.0009	$\mathbf{0.0267 \pm 0.0007}$	$\mathbf{0.0269 \pm 0.0007}$
99.9th %ile NAE	0.1369 ± 0.0014	0.0952 ± 0.0025	$\mathbf{0.0387 \pm 0.0009}$	$\mathbf{0.0403 \pm 0.0009}$
99.99th %ile NAE	0.1530 ± 0.0025	0.1229 ± 0.0080	$\mathbf{0.0548 \pm 0.0045}$	1.8218 ± 0.4035
Maximum NAE	0.1616 ± 0.0029	0.1303 ± 0.0080	$\mathbf{0.0662 \pm 0.0069}$	2.4909 ± 0.1408

Table 5. Statistics of the errors of $V_{P_{max}}$ estimates under the different models. Averages and 95% CIs across 10 runs shown. Results rounded to four decimal places. Lowest –and tied for lowest, in the sense of overlapping CIs– values shown in bold.

	Closed Form	Learned Models		
		Direct	*Stagewise*	*StagewiseC*
NRMSE	0.0697 ± 0.0013	0.0597 ± 0.0005	0.0507 ± 0.0013	$\mathbf{0.0333 \pm 0.0016}$
Median NAE	0.0098 ± 0.0000	0.0121 ± 0.0002	$\mathbf{0.0017 \pm 0.0000}$	$\mathbf{0.0017 \pm 0.0000}$
95th %ile NAE	0.0735 ± 0.0006	0.1111 ± 0.0023	0.0075 ± 0.0001	$\mathbf{0.0072 \pm 0.0001}$
99th %ile NAE	0.2609 ± 0.0063	0.2548 ± 0.0051	0.0221 ± 0.0015	$\mathbf{0.0146 \pm 0.0002}$
99.9th %ile NAE	0.9623 ± 0.0048	$\mathbf{0.6133 \pm 0.0341}$	0.8539 ± 0.0137	$\mathbf{0.6623 \pm 0.0708}$
99.99th %ile NAE	1.1801 ± 0.0377	0.9041 ± 0.0431	$\mathbf{1.0789 \pm 0.0144}$	1.1334 ± 0.0228
Maximum NAE	1.2222 ± 0.0007	$\mathbf{0.9603 \pm 0.0192}$	1.1512 ± 0.0133	1.1934 ± 0.0173

Dependence on training set size We saw that the learned models outperform the closed form estimates. But how many examples do the learners need to see, before they can do so? In Fig. 2, we provide learning curves for the NRMSE on the final P_{max} & $V_{P_{max}}$ estimates. We conclude that we do not need to use a large training set to exceed the quality of the closed-form estimates. We can also see that beyond –say 5000– training examples, the increase in training set size only marginally improves the quality of the estimates for the *Direct* & *Stagewise* estimators. *StagewiseC* is an exception in two ways: (i) it needs a larger sample size for its estimates of P_{max} to be comparable to the closed-form ones, (ii) after seeing about 10000 training examples, it is already outperforming all other approaches in terms $V_{P_{max}}$ estimation, yet it can take advantage of a larger sample size to produce even better estimates.

4.3 Analysis of the results

As discussed, all trained models examined produced better estimates than the closed-form ones. Indicatively, inspecting Tables 4 & 5, we find that the reduction in NRMSE over the closed-form estimate for the *Stagewise* model's estimates in P_{max} is about 69%. Similarly, the reduction in NRMSE over the closed-form estimate for the *StagewiseC* model's estimates in $V_{P_{max}}$ is about 52%. The results are similar when the goal is not just good average performance, but also to minimize the maximum prediction error: *Stagewise* reduces the closed-form estimates' maximum NAE in P_{max} by about 59% and *Direct* reduces the closed-form estimates' maximum NAE in $V_{P_{max}}$ by about 21%. On the intermediate estimates of Tables 1 & 2, we saw that the relative improvement of the learned model estimates over the closed-form estimates is even more impressive, for every quantity and every evaluation measure examined. In short, we get both lowest average and lowest maximum errors by the learned models.

We attributed the relatively large errors in the estimates (both closed-form & learned) of $V_{P_{max}}$ to cases in which P_1 & P_2 are close, and a small error in either results to choosing the wrong global MPP (V_1 & V_2 may significantly differ even when $P_1 \approx P_2$). The models we discussed tackle this issue in different ways: *Direct* models $V_{P_{max}}$ directly, without separately modelling the two MPPs. *Stagewise* estimates P_1 & P_2 with low error –and so does the corresponding V_i it assigns as $V_{P_{max}}$. Finally, *StagewiseC* trains an accurate classifier to recognise the global MPP, circumventing the need to compare the estimates of P_1 & P_2.

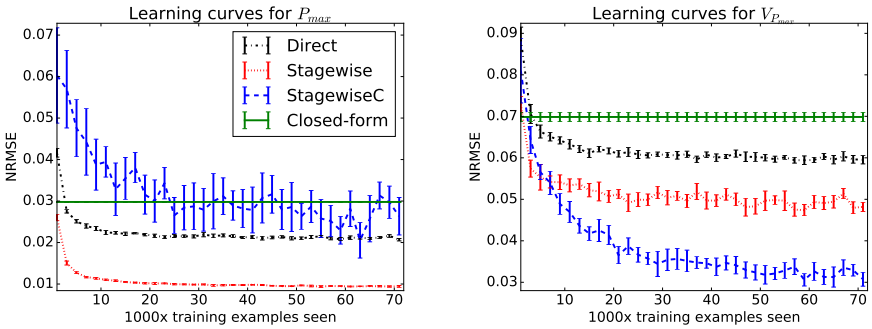

Fig. 2. Learning curves (NRMSE vs. training set size) on the final P_{max} [LEFT] & $V_{P_{max}}$ [RIGHT] estimates under each model. Averages and 95% CIs across 10 runs shown. We investigate training set sizes of 1000 up to the 75% of the total available examples (71178) taken at steps of 1000.

The percentile results provided, allow the reader to see that indeed, such large errors appear very rarely under the learned models. For instance, Table 4 shows that only one in 10000 (99.99th %ile) estimates of the *Stagewise* model for P_{max} is expected to have a NAE greater than 5.48%[8], whereas one in 20 (95th %ile) of

[8] More precisely, 95% of the times, 99.99% of the estimates of P_{max} under the *Stagewise* model will have a NAE smaller than some value that lies between 5.03% and 5.93%. In the discussion, we sacrifice this level of mathematical rigour for simplicity.

the closed-form estimates will have a NAE greater than 6.45%. Similarly, Table 5 shows that only one in 100 (99th %ile) estimates of the *StagewiseC* model for $V_{P_{max}}$ is expected to have a NAE greater than 1.46%, when the same number of closed-form estimates will have a NAE greater than 26.09%.

Which model is to be preferred in practice will depend on the exact guarantees we want for our system and the design limitations imposed. *Direct* produces the smallest maximum error in terms of $V_{P_{max}}$. *Stagewise* outperforms all other approaches in terms of estimation of P_{max}. *StagewiseC* produces the smallest number of high-value errors on $V_{P_{max}}$. To practitioners, we can propose 'combining' *Stagewise* & *StagewiseC*: train all 5 learners used by *StagewiseC*, use the classifier's prediction for selecting the V_i for $V_{P_{max}}$, but ignore the classifier and directly compare the estimates of P_1 & P_2 to choose the P_{max} as in Eq. (2).

Finally, we also saw in Fig. 2 that even a small fraction of the available training examples (e.g. fewer than 5000) is more than enough for the learned models to considerably improve upon the closed-form estimates. Initial explorations of the ensemble size M and the depth of the trees –omitted due to space limitations– suggest that increasing either leads to better predictive performance for all models albeit at a higher computational cost.

5 Conclusion and Future Work

We introduced 3 gradient boosted tree models to perform PV power estimation under partial shading conditions. Our empirical results show that, compared to the current state-of-the-art closed-form estimates, the errors on the gradient boosting models' predictions are smaller on average, in the worst case and also the large errors committed are fewer in number. Only a small number of training examples is needed to improve upon the closed-form estimates. Depending on the computational resources, we can opt to parallelize the training of the individual ensembles each model involves or to improve the quality of our estimation at the cost of increased training time by increasing the number or depth of the trees.

A next step would be to take into account the covariance among targets. A more detailed investigation of learning algorithms, hyperparameter optimization and analysis of the resulting computational cost/approximation tradeoffs is also left for future work. So is the derivation of a simpler, interpretable model to replace the closed-form estimates. The skew in the global MPP distribution could be dealt with cost-sensitive adaptations of boosting [15], improving performance.

The same ideas could also be applied to more complicated scenarios, such as different PV configurations (arrays of modules connected in series-parallel, bridged-linked etc), multiple irradiance levels where more than two MPPs appear or in the more realistic case of non-uniform temperature across the PV system.

Acknowledgements This project was partially supported by the EPSRC Centre for Doctoral Training [EP/I028099/1] & the EPSRC LAMBDA [EP/N035127/1] & Anyscale Apps [EP/L000725/1] project grants. N. Nikolaou acknowledges the support of the EPSRC Doctoral Prize Fellowship. E. Batzelis carried out this research at NTUA, Athens, Greece under the support of the 'IKY Fellowships of Excellence for Postgraduate Studies in Greece-Siemens Program'.

References

1. E. Batzelis, G. E. Kampitsis, and S. A. Papathanassiou. A MPPT algorithm for partial shading conditions employing curve fitting. In *EU PVSEC*, pages 1502–1507, 2016.
2. E. I. Batzelis, P. S. Georgilakis, and S. A. Papathanassiou. Energy models for photovoltaic systems under partial shading conditions: a comprehensive review. *IET Renew. Power Gener.*, 9(4):340–349, 2015.
3. E. I. Batzelis, I. A. Routsolias, and S. A. Papathanassiou. An explicit PV string model based on the Lambert W function and simplified MPP expressions for operation under partial shading. *IEEE Trans. Sustain. Energy*, 5(1):301–312, 2014.
4. K. Brecl, K. Topič, and M. Topič. Self-shading losses of fixed free-standing PV arrays. *Renew. Energy*, 36(11):3211–3216, 2011.
5. R. Busa-Fekete, B. Kégl, T. Éltető, and G. Szarvas. Ranking by calibrated AdaBoost. In *Proc. of the Learning to Rank Challenge*, pages 37–48, 2011.
6. R. Caruana and A. Niculescu-Mizil. An empirical comparison of supervised learning algorithms. In *ICML*, pages 161–168, 2006.
7. T. Chen and C. Guestrin. Xgboost: A scalable tree boosting system. In *SIGKDD*, pages 785–794, 2016.
8. C. Deline, A. Dobos, S. Janzou, J. Meydbray, and M. Donovan. A simplified model of uniform shading in large photovoltaic arrays. *Sol. Energy*, 96:274–282, 2013.
9. J. A. Dolan, E. Lee, C. Yeh, S. Ben-Menahem, and A. K. Ishihara. Neural network estimation of photovoltaic IV curves under partially shaded conditions. In *IJCNN*, pages 1358–1365, 2011.
10. M. Fernández-Delgado, E. Cernadas, S. Barro, and D. Amorim. Do we need hundreds of classifiers to solve real world classification problems? *JMLR*, 15:3133–3181, 2014.
11. J. H. Friedman. Greedy function approximation: A gradient boosting machine. *Annals of Statistics*, 29:1189–1232, 2000.
12. L. Mason, J. Baxter, P. Bartlett, and M. Frean. Boosting algorithms as gradient descent. In *NIPS*, pages 512–518. MIT Press, 2000.
13. S. Moballegh and J. Jiang. Modeling, prediction, and experimental validations of power peaks of PV arrays under partial shading conditions. *Sustain. Energy*, 5(1):293–300, 2014.
14. D. D. Nguyen, B. Lehman, and S. Kamarthi. Performance evaluation of solar photovoltaic arrays including shadow effects using neural network. *IEEE Energy Convers. Congr. Expo.*, 6(2):3357–3362, 2009.
15. N. Nikolaou, N. Edakunni, M. Kull, P. Flach, and G. Brown. Cost-sensitive boosting algorithms: Do we really need them? *Machine Learning*, 104(2):359–384, 2016.
16. G. Psarros, E. Batzelis, and Papathanassiou S. Analysis of local mpps on the p-v curve of a partially shaded pv string. In *EU PVSEC*, pages 3383–3389, 2014.
17. G. N. Psarros, E. I. Batzelis, and S. A. Papathanassiou. Partial shading analysis of multistring PV arrays and derivation of simplified MPP expressions. *IEEE Trans. Sustain. Energy*, 6(2):499–508, 2015.
18. P. Rodrigo, F. Fernández, F. Almonacid, and J. Pérez-Higueras. A simple accurate model for the calculation of shading power losses in photovoltaic generators. *Sol. Energy*, 93:322–333, 2013.
19. P. Viola and M. J. Jones. Robust real-time face detection. *IJCV*, 57(2):137–154, 2004.
20. A. J. Wyner, M. Olson, J. Bleich, and D. Mease. Explaining the success of adaboost and random forests as interpolating classifiers. *arXiv:1504.07676v2*, 2017.