MSE 2017 Project Individual Report

Optimization of Wave Energy Converters

Yuanzhong Xia
The University of Adelaide
Adelaide, SA 5005
a1700831@student.adelaide.edu.au

ABSTRACT

Wave energy is a massive potential energy form in the world, but using it in a efficient way is always a historical issue. This report mainly describes our progress and important details about this wave energy optimization project. In this project we are working on building fast and accurate surrogate models for real computational fluid models, and we provide a web application for researchers to do wave energy research easily. My individual contributions form a fair part of the report, while shared project information is mentioned as well. In terms of research aspect, "scikit-learn" package will be mainly introduced; for web application, the development on both web app hosting server and high performance computer will be covered. Last but not least, future works and conclusions are also included in this report.

KEYWORDS

Machine Learning, Wave Energy, Optimisation, Web application, High performance computer

1 INTRODUCTION AND MOTIVATION

Ocean wave energy is a massive sustainable energy resource, the harvesting of which relies on the design and control of wave-induced oscillating systems. In Australian, according to the preliminary studies of the Commonwealth Scientific and Industrial Research Organisation (CSIRO), southern coastline of Australia has a great wave resource, because the strong Southern Ocean winds generate large waves consistently and which travel northwards to Australia's southern coastline. The research from CSIRO also shows that wave energy could contribute up to 11% of Australia's energy, which is quite enough for powering up a city of the size like Melbourne fully by 2050, making it a strong contender in Australia's renewable energy mix. [1] Although, wave energy is a new type of renewable energy and it is with a concentrated form witch is less variable on an hourly basis than wind energy, it's lack of researching and exploration currently. If wave energy can be largely explored, it must make a significant contribution to global energy structure.

As far as we know from previous researches, the wave energy cannot be used directly, it has to be captured and converted to electricity power by some extra devices. The devices which can achieve this goal are generally called wave energy converter (WEC). It is developed to capture and convert wave energy to electricity energy.

[1]Information from: https://www.csiro.au/en/Research/OandA/Areas/Marine-technologies/Ocean-energy/Wave-energy.

Our work is based on existing researches by Wu *et al.*[1], Arbonès *et al.*[2], and Ding *et al.*[3]. We try to maximize the use of wave energy by finding the best layout of WECs through machine learning technologies, and provide researchers/users with a friendly web app containing lots of helpful tools to accelerate the research. Because the capacity of energy captured by a single buoy is limited, thus, applying multiple WECs is necessary for real world problems. However, if WECs are applied in a large area and each of them does not affect each other, the cost of deployment is super large. Here is why we are doing this research, our purpose is to maximize the energy production with a limited area (also called a wave energy farm) for a limited number of WECs. That means using lowest cost on deployment to get highest energy production.

In this project, WEC is mostly called buoy, and more specifically each buoy is connected with three tethers to seafloor. Buoys can have various types, floating buoys, submerged buoys and etc. Only submerged buoys are considered in this project, because the researches we are based on are for submerged buoys only.

We spent most time on discovering the ways to train fast and accurate surrogate machine learning models, to replace the original time-consuming physical computational fluid dynamics (CFD) models. In this way, researchers can do more experiments within much shorter times without losing much accuracy. In addition, CFD models related to this projects are described in Scruggs *et al.*'s work [4] and Wu's work [5]. Lots of physical formulas are involved in this models, therefore it cannot be described here in detail.

Our web app can generate training data in a highly customized way, and it provides different training methods for users to choose. All of them can reach an acceptable accuracy, and the model predicting time is generally 30 times faster than CFO models. In addition, some visualization tools are included for users' observing and tuning configurations. Meanwhile, the model training process takes a lot of time, therefore it's handled by high performance computer (HPC) provided by University of Adelaide. Users can check the progress regularly on the web app. When the training process is finished, email notifications are sent by default, and users can play around with the super quick prediction models to find the layouts they like.

2 MY COMPONENTS

In this project, I am mainly responsible for the machine learning methods provided by "scikit-learn" package [6] (Section 4.2), finding and configuring proper web app hosting servers (Section 4.3), the server-side app codes (Section 4.4.1) and HPC-side codes (Section 4.4.2).

Actually, each of them requires lots of hard works, which are not quite possible to be finished within this semester completely. My

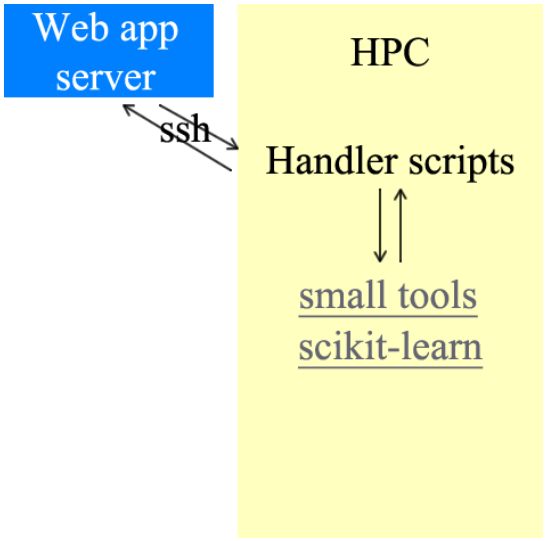


Figure 1: The interactive relationship among my components.

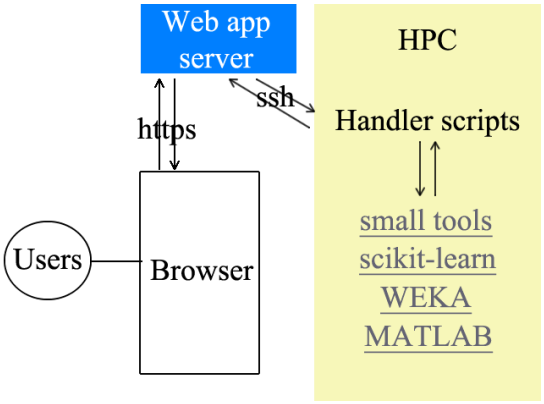


Figure 2: The interactive relationship between my components and the whole system.

teammates and I spent most time on the researching of the whole project in this semester - using machine learning methods to train better CFD surrogate models, and each of us contributed a lot to improve the accuracy and speed performance of trained surrogate models, thus the web app is just a prototype with basic functions and not well structured.

The interactive relationship between my components are shown in Figure 1. "Scikit-learn" is a package installed on HPC, and my machine learning codes will run on HPC with the dependency on "scikit-learn". Some small tools, like the tool for generating random buoy coordinates, are also hosted on HPC for reducing the loads of web app server (more details will be discussed in Section 4.3). The web server-side programs are back-end programs, handling the network traffic between itself and HPC via "ssh" tunnel. To handle the request, there are scripts for server-side programs to invoke because HPC does not have the ability to receive HTTP/HTTPS requests.

The interactions between other teammates' components and my components are shown in Figure 2. Although "scikit-learn" is just one of the machine learning components on HPC, my handler codes will handle all the network traffics between web app server and machine learning components. Meanwhile, the web app server handles HTTPS requests from users' browsers as well, which looks like a middleware of the whole system.

3 RELATED WORKS

Although, wave energy is a new type of energy and it has not been largely explored, there are still some previous researches on this topic.

3.1 Buoy design

Buoy design belongs to mechanical engineering, however, it is the foundation of all optimization works. Buoys can be roughly divided into six types (Figure 3), and most of them can be both floating and submerged.

A research from Ding *et al.* [3] shows the hardware implementation named as three-tether submerged buoy. This type of buoy is chosen to be used during the whole project, because we have the corresponding CFO model codes implementation; plus, the research team is exactly from our university.

Additionally, there are also other designs for submerged buoy. Song *et al.*'s research [12] describes a design of one-tether submerged buoy which introduced some Internet of Things concepts. Reikard *et al.*'s research [10] describes three types of buoys (an attenuator buoy, a floating heave buoy array, and an oscillating flap device), and their CFD models in different ocean areas all over the world.

3.2 Previous optimization works

A research from Wu *et al.* [1] describes the single three-tether buoy optimization (like buoy submerged depth and the angle between tether and seafloor), and regular honeycomb-shape buoy layout optimization. It is different from our project which is based on a random layout optimization.

Fernández *et al.* did a research [9] for marine energy prediction using a ordinal classifier method. Their work aims to predict the wave energy in a different direction, they predict the wave height and energy flux range in an area. In addition, their training data are not from a CFD model, but from National Oceanic and Atmospheric Administration (NOAA), National Data Buoy Center (NDBC) and a part of the National Weather Service (NWS). Therefore, it is different from out project, what we are working on is predicting buoys' power outputs.

Dripta *et al.*'s work [11] is quite similar to ours. They used Gaussian process regression combining with CFD models with dynamic submerged depths depending on the wave energy distribution, it also combines the genetic programming for generating the best buoy 3D layout, but the number of buoys is fixed to 40. This research is much more complicated, involving with lots of physical models, and it is not very easy to be ported into various number of buoy cases, since some buoys are bind with each other in a specified

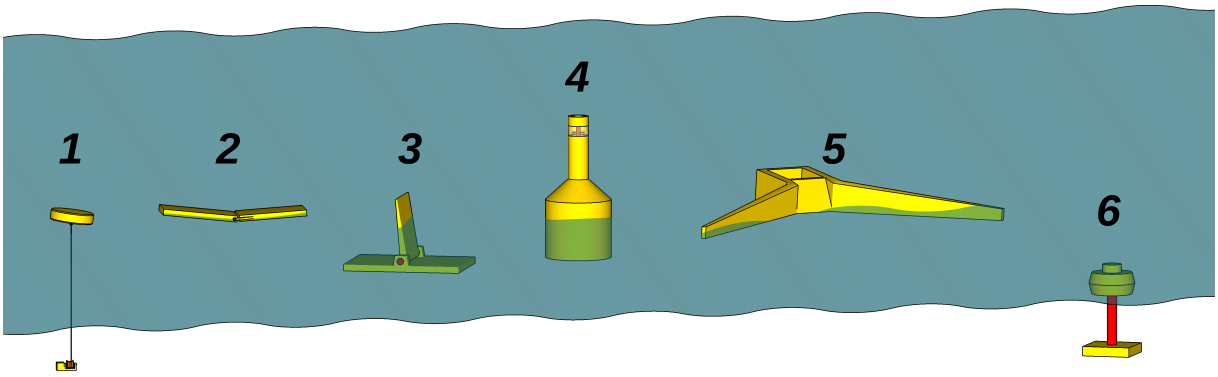


Figure 3: Wave energy converter (buoy) types [13]: 1. Point absorber, 2. Attenuator, 3. Oscillating wave surge converter, 4. Oscillating water column, 5. Overtopping device, 6. Submerged pressure differential.

ocean environment. Our work is a universal one, we purpose to make the universal machine learning models fitting for dynamic number of buoys.

As a result, using machine learning methods to predict the buoy power outputs is still a fresh and less explored topic in research area.

4 PROGRESS TO DATE

By working on this project so far, I gained a lot of maching learning model training experience, and had a first-hand view on machine learning. As I mentioned in Section 2, my works are mainly adjusting machine learning methods from "scikit-learn" package, and doing server side works, including the web app hosting server and cloud computing server (HPC, discussed in Section 4.4.2). However, I took part in almost every stages and components, thus I would describe my progress in detail of every step here.

4.1 Data pre-processing

Initially, we have the training input which is made up of only buoy coordinates, but there is a potential issue (Figure 4) if we train the models using coordinates only. The issues is: the following 24 input sequences represent exactly the same layout:

- $(x_A, y_A, x_B, y_B, x_C, y_C, x_D, y_D)$
- $(x_A, y_A, x_B, y_B, x_D, y_D, x_C, y_C)$
- $(x_A, y_A, x_C, y_C, x_B, y_B, x_D, y_D)$
- $(x_A, y_A, x_C, y_C, x_D, y_D, x_B, y_B)$
- $(x_A, y_A, x_D, y_D, x_B, y_B, x_C, y_C)$
- $(x_A, y_A, x_D, y_D, x_C, y_C, x_B, y_B)$
- $(x_B, y_B, x_A, y_A, x_C, y_C, x_D, y_D)$
- $(x_B, y_B, x_A, y_A, x_D, y_D, x_C, y_C)$
- $(x_B, y_B, x_C, y_C, x_A, y_A, x_D, y_D)$
- $(x_B, y_B, x_C, y_C, x_D, y_D, x_A, y_A)$
- $(x_B, y_B, x_D, y_D, x_A, y_A, x_C, y_C)$
- $(x_B, y_B, x_D, y_D, x_C, y_C, x_A, y_A)$
- ... 12 more combinations

For machine learning, totally 24 different training records describing the same thing will be learnt by the model, which is obviously a waste of training inputs because lots of inputs are duplicated potentially. Moreover, this can mess up the fitting model very heavily, and the most important thing is that it is definitely avoidable.

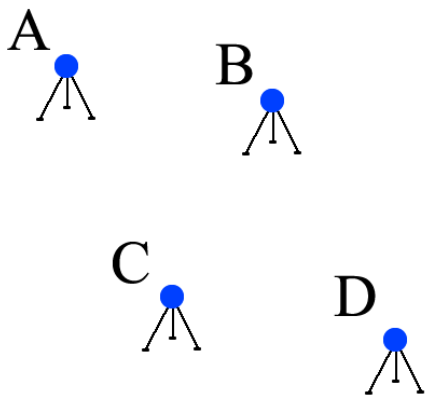


Figure 4: A random four buoy wave energy farm layout, buoys are marked as "Buoy A", "Buoy B", "Buoy C" and "Buoy D".

Therefore, we have ideas and names for pre-processing the buoys coordinates, and I implemented all of them in Python. [2]

The randomly generated buoy coordinates are called "raw data", and we have see the weakness of using those data directly above already. We used root relative squared error (RRSE) to measure the accuracy of our model (more details are discussed in Section 5). For raw data, 10,000 records are used for training and another 10,000 records are used for testing in random forest regressor model, with 500 estimators. (More details on random forest will be introduced in Section 4.2.) Finally, we got the average RRSE of 0.6072, and this value was used as the baseline accuracy performance.

4.1.1 High precision raw data. This was a mistake of using MATLAB CFD model codes, the outputs of real numbers from MATLAB were cut off (like $1.767E+06$) by default, we named it "low precision

[2] Codes can be found in https://github.cs.adelaide.edu.au/a1700831/mse-buoy/tree/dev/tools/.

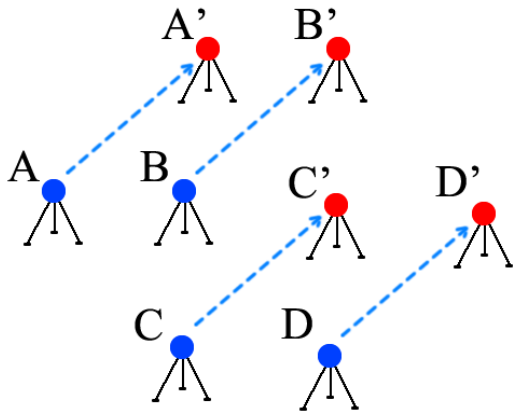


Figure 5: The blue buoy layout moves to red buoy layout by shifting, without changing the distance between any of them.

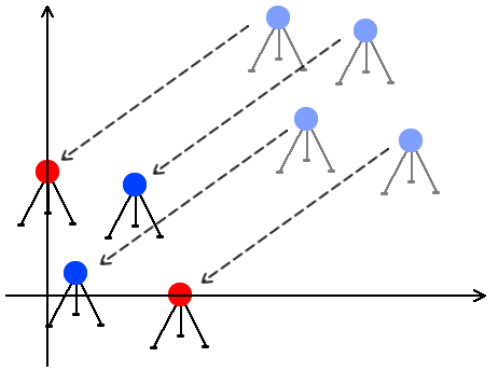


Figure 6: The buoy layout shifts from top right to bottom left, so that at least two buoys are in both x-axis and y-axis in total, and the rest buoys are all in the first quadrant.

raw data". With low precision raw data, our algorithm performed actually much better than using high precision raw data. For example, the RRSE of "low precision raw data" is around 0.5015 using the same configurations as mentioned above. Thus, after we found the problem, I re-evaluated all the training data with high precision output (like 1767307.77) using HPC (details are mentioned in Section 4.4.2). Then, evaluation results are combined and shared to teammates.

4.1.2 Raw sorted data. Mengyu firstly came up with this idea. Sorting the coordinates by x first; if x values are equal, sorting by y in either increasing or decreasing order. This ensures the relative positions within the layout can have an order.

With this pre-processing operation, the average RRSE of random forest models reduced to 0.6007. Although, it is a very slight improvement, this is still a precious attempt which indicates that data pre-processing is not that easy. However, sorting the raw training data works to some extend, but not very significantly as expected. There might be potential reasons like what is shown in Figure 5, shifting buoy layout from a position to a new position within the farm area does not affect the power output, i.e. the farm containing exactly "buoy A, B, C, and D" generates the same energy as the farm containing "buoy A', B', C', and D'" only.

4.1.3 Dimension-reduced data. To fix the issue found in Section 4.1.2, we decided to use the property that shifting buoy layout from a position to a new position within the farm area does not affect the power output. Therefore, we align the buoy points to bottom left (like Figure 6). The total power output keeps the same, but this can eliminate the case shown in Figure 5.

Actually, this operator makes at least two training input values become 0. In Figure 6, the red points have either x or y coordinate with value of 0. Finally I get the average RRSE 0.5170 by applying this operator, and this is already a significant improvement. Combining this operator with sorting operator, I get an even better RRSE - 0.4234.

4.1.4 Feature data. I contributed this idea, and extracted the features inspired by a research paper by Pilát *et al.*'s work [7]. Unlike the mentioned two pre-processing operators in Section 4.1.2 and Section 4.1.3 reduce the potential duplicated patterns by reducing complexity (like dimension) of input training data, this operation will increase the training data's dimension. However, this operation will break the relationship between buoys, because it actually extracts the relationships between buoys. In four buoy case, I extracted 12 features:

(1) max distance between any two buoys
(2) min distance between any two buoys,
(3) average distance between any two buoys,
(4) max degree d between any two buoy lines, $d \in [0, 180)$,
(5) min degree d between any two buoy lines, $d \in [0, 180)$,
(6) max X difference between any two buoys,
(7) min X difference between any two buoys,
(8) max Y difference between any two buoys,
(9) min Y difference between any two buoys,
(10) max angle between X-axis and any buoy,
(11) min angle between X-axis and any buoy,
(12) average angle between X-axis and any buoy.

The guideline for extracting features mentioned by Pilát *et al.* is extracting as many meaningful features as possible, thus I extracted even more features than the dimension number of original raw data. This work was done at the very beginning of project, hence it contributed some significant improvement at the beginning, and the average RRSE was 0.5295.

Later, I found this was actually a very common optimization for machine learning methods in various areas. For example, Bartz-Beielstein and Zaefferer's research [8] also used feature extraction for training linear models and tree-based models.

4.2 Scikit-learn

The use of "scikit-learn" package is inspired by Pilát *et al.*'s work [7] as well. Beside their paper, they provided all the source codes, thus I downloaded and looked into them. They used a piece of very

Table 1: RRSE comparison among models from "scikit-learn", with default configurations

Model	Using features	RRSE
Random Forest	yes	**0.5434**
	no	**0.6453**
AdaBoost	yes	0.6196
	no	0.8760
Bagging	yes	0.5727
	no	0.6772
Extra Trees	yes	0.5683
	no	0.6609
Gradient Boosting	yes	0.5577
	no	0.7832

simple codes to achieve a very high accurate prediction rate from a machine learning method called "random forest". That's the direct reason why I choose this package, it contains lots of implemented training algorithms like: AdaBoost classifier and regressor, bagging classifier and regressor, random forest classifier and regressor, gradient boosting classifier and regressor, etc. In this project, only regressors are taken into consideration because of the characteristics of power output values. As well, influenced by Pilát *et al.*'s paper, I firstly chose random forest regressor as the starter machine learning model.

4.2.1 Model comparison. "Scikit-learn" package provides lots of machine learning methods for using, but I do not want to use them all. But for the best accuracy, I need to use the best machine learning method in this project. Ideally, there should be a model which can solve this problem very well (accurate and fast).

The way that I figured out the most suitable method was by trying different models with default configurations with three repeats. Results are in Table 1, all the models are regression models with 6000 thousands records for training, 2000 for testing. It is very obvious that random forest performs the best on training with both "raw data" and "feature data".

4.2.2 Random forest regressor. As questioned by project supervisor, I dumped the random forest by accessing undocumented variables which store the actual decision trees (Figure 7). As well, the variable names were referred by looking into the actual source codes.

Sometimes, the default configurations could not lead to a best performance, thus I spent some times on finding the affecting factors,

- Training data size. The training data size matters a lot, little data size can lead to incomplete training, whereas large data size might cause over-fitting problems.
- Number of estimators. With more trees, the model performs more "careful" and accurate, however, the training speed has a positive correlation with it.
- Depth of each estimators. Not only training speed, but also prediction speed has a positive correlation with it. Normally, this value is found by the algorithm itself, but a proper depth does no harm to the prediction accuracy, but improve the speed.

From my discovery above, the default depth of each estimator looks quite acceptable, as the speed of random forest is super fast than most machine learning algorithms. Therefore, I tested the algorithms with different training data sizes and different estimator numbers.

Results can be found in Table 2 and Table 3. The tables are made of the average results of five runs, and during the experiment, we found the RRSE difference of multiple runs with the same configuration are around 0.01. Therefore, we assume in the whole project, average RRSE of multiple runs are roughly equal to RRSE of a single run. The results from tables are obvious, using features has a better RRSE than using raw data only, and using more trees improves the performance significantly. Although, using more data does perform better than using more trees in random forest regressor model, 1000-tree random forest is selected as default training model for "scikit-learn" as a balanced choice.

4.2.3 Four buoy layout. At the beginning, we started making surrogate models to calculate the power output of a wave energy farm with exactly four buoys inside. The area of the farm is defined as 283×283 square meters, and the minimum distance between any of two buoys is 50 meters. Plus, there are also other definitions for the CFD model, for example, submerge depth is defined as 3.5 meters, but they are not related to generating coordinates. All those specifications are applied because of the time-consuming CFD models implemented in MATLAB by Ding *et al.* [3].

I wrote a universal program for generating any number of buoys' coordinates randomly, then those generated coordinates would be evaluated by CFD models. I also wrote some helpful tools for unifying the file format of training data, and some scripts for parallel evaluating generated buoy coordinates. In addition, plenty of experiments on four buoy layout using "scikit-learn" are done be me as well.

4.2.4 Two buoy layout. In the regular collaborative meeting with school of mechanical engineering and school of mathematics on 16th May, we came up with an idea to evaluate the correctness of machine learning methodologies by training with two buoys, being an even simpler scenario. If we cannot learn this, then chances are low that more complex ones can be learned. As well, the area of the farm is defined as 200×200 square meters, and the minimum distance between any of two buoys is still 50 meters. I generated and evaluated the two buoy layout data for training and testing, and I did the experiments (results are shown in Section 5) as well. Two-buoy feature extraction was also implemented by me.

By using two buoy layout and reducing the dimension, the RRSE is reduced to 0.0150 which is super good! Additionally, RRSE of two buoy layout without data pre-processing can reach 0.2911, which is much better than four buoy layout at the beginning.

As a result of only two dimensions (two coordinates have four values, and two of them are eliminated to 0) in training data, it is easy to draw 3-D plotting using the non-zero two dimensions as x-axis and y-axis to see the model prediction accuracy. (Shown in Figure 8a, 8b, 8c and 8d.)

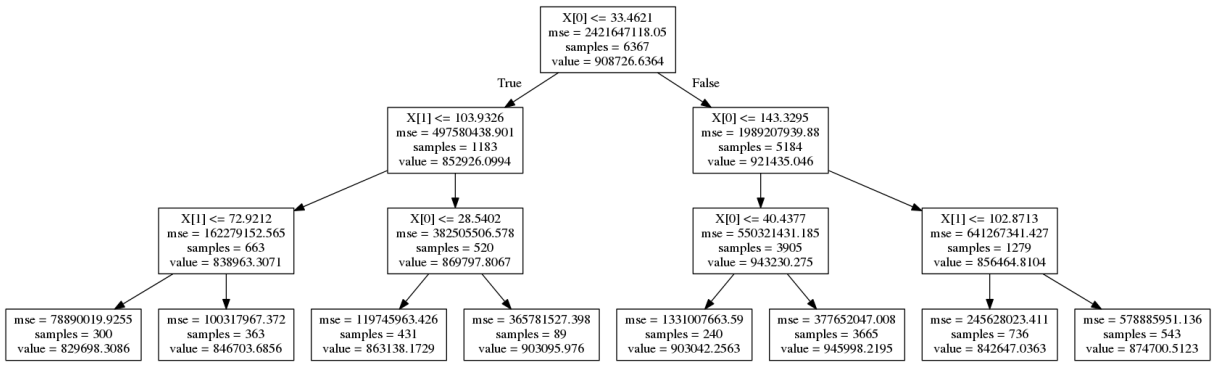


Figure 7: A tree (or an estimator) in the trained random forest regressor model. This tree has a specified depth of three. In a random forest, the number of estimator is exactly the number of this kind of decision trees. As a regression model, it is very obvious that the prediction values are determined by leaf nodes, and prediction values are certain.

Table 2: Average RRSE on different number of trees in random forest model of three runs, with different size of input "raw data".

Input size	1000	2000	3000	4000	5000	6000	7000	8000	9000	10000
10 trees	0.8300	0.7770	0.7890	0.7770	0.7240	0.6920	0.7220	0.7140	0.7040	0.6750
50 trees	0.8150	0.7420	0.7450	0.7350	0.6820	0.6670	0.6820	0.6740	0.6610	0.6430
100 trees	0.7940	0.7470	0.7380	0.7290	0.6830	0.6590	0.6790	0.6730	0.6580	0.6380
200 trees	0.8000	0.7400	0.7340	0.7230	0.6800	0.6580	0.6770	0.6710	0.6560	0.6360
500 trees	0.8010	0.7410	0.7390	0.7220	0.6770	0.6540	0.6740	0.6690	0.6550	0.6380
1000 trees	0.7960	0.7430	0.7350	0.7210	0.6760	0.6550	0.6730	0.6700	0.6550	0.6350

Table 3: Average RRSE on different number of trees in random forest model of three runs, with different size of input "feature data".

Input size	1000	2000	3000	4000	5000	6000	7000	8000	9000	10000
10 trees	0.6050	0.6030	0.6580	0.5980	0.5700	0.5800	0.5810	0.5840	0.5770	0.5690
50 trees	0.5690	0.5730	0.6270	0.5770	0.5490	0.5520	0.5500	0.5570	0.5560	0.5520
100 trees	0.5660	0.5760	0.6240	0.5710	0.5460	0.5510	0.5460	0.5530	0.5530	0.5460
200 trees	0.5720	0.5680	0.6250	0.5710	0.5480	0.5500	0.5450	0.5510	0.5510	0.5470
500 trees	0.5660	0.5690	0.6260	0.5690	0.5450	0.5480	0.5430	0.5530	0.5490	0.5460
1000 trees	0.5640	0.5680	0.6250	0.5680	0.5440	0.5480	0.5430	0.5510	0.5480	0.5460

To make it more clear and I was suggested by our supervisor, I also trained a bad model which is shown in Figure 9. It is quite easy to distinguish that the prediction surface of the bad model cannot predict the raised power output areas correctly.

4.3 Web server selection

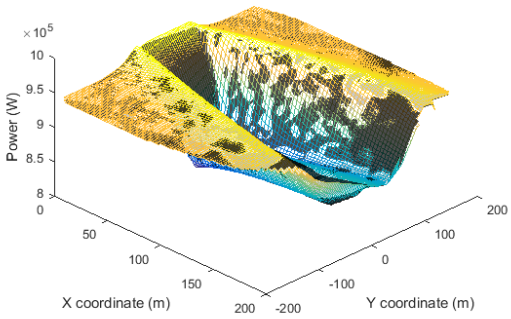
When I worded with my team, I mainly contributed to the back-end affairs, including server selection and server app programming.

In our current demonstration web app, only the following requests are handled:

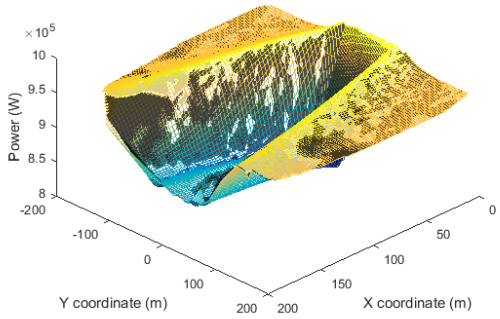
- Generating buoy coordinates randomly;
- Evaluating buoy layouts with CFD models;
- Training selected models with evaluated data;
- Predicting a user-input buoy layout using just-trained model;

The server does not have lot of requests and jobs to handle, because most works will be run on HPC. The only thing that our web server should do is to maintain the only SSH connection with HPC, and the HTTPS connections with users' browsers. Therefore, the requirements of the server does not matter too much, even the cheapest Virtual Private Server (VPS) should work properly. Additionally, I did some more researches on non-VPS platforms, because "Software as a Service" (SaaS) platforms were getting more and more popular nowadays.

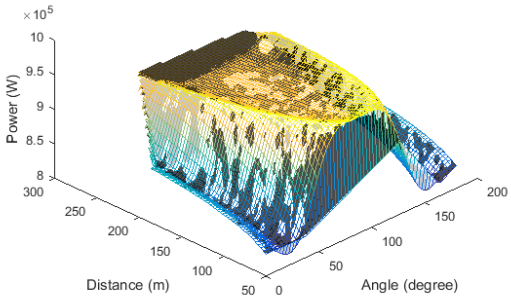
4.3.1 Heroku. In the searching process, I started from something free and popular. Heroku [14] was firstly mentioned by out course supervisor Christoph during an regular weekly meeting. This platform is very likely to other SaaS platforms, like SinaAppEngine [15]. By comparing to other platforms I used before, Heroku



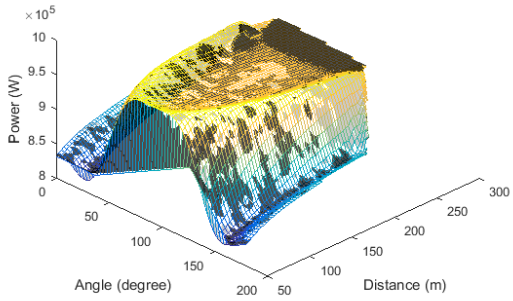
(a) Raw data with view point (45, 45).



(b) Raw data with view point (135, 45).



(c) Feature data with view point (315, 45).



(d) Feature data with view point (45, 45).

Figure 8: Two buoy layout, the first two plotting results are trained with raw data whose dimensions with full zeros are eliminated; whereas the last two plotting results are trained with feature data: the distance between two buoys, and the angle between the line through two buoys and x-axis $\in [0°, 180°)$. x-axis and y-axis are the two training input dimensions, z-axis is the power output (W). The black color represents the prediction power output map, while the rest colors represent the real one. In non-black colors, the brighter the colors are, the more power outputs this layout generates. Note: MATLAB view point specification: https://au.mathworks.com/help/matlab/ref/view.html**.**

is more powerful that it can import codes from external git repositories directly, whereas other SaaS platforms require user to maintain a new git repository or SVN repository, which is not convenient.

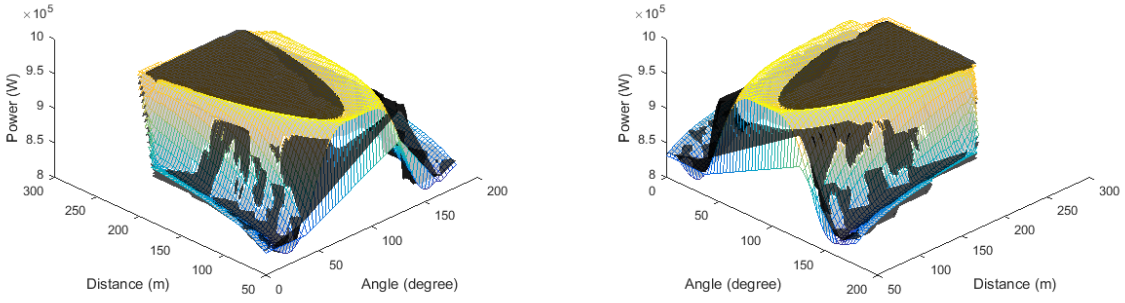
However, for this project, Heroku is not suitable directly. The reasons are:

(1) Free accounts do not have database access. The databases are counted as plug-in modules, and I have to bind my credit card information to them, then I can use the module service.
(2) Heroku doesn not support multiple web app runtimes. Since ideally we want the server can run PHP, Java and Python, because the tools we wrote were in different languages. Due to this limitation, we have to rewrite some of our programs which will take a lot of time.
(3) It does not support SSH connection in runtime, I looked at the command line tool. It connects to a virtual space which does have SSH tools but it is not where the app runs. But due to the search results from Google, the running environment does have SSH, but the public key is unknown. As well, reverse SSH tunnel does not work directly.

Therefore, Heroku potentially works, but it takes time for finding its SSH configuration and move all codes into the same languages.

4.3.2 OpenShift. OpenShift [16] is provided by Red Had Ltd., and it also provides a free-to-use platform. For enterprise version, it provides a fully functional Red Hat Linux platform for user to deploy their web apps, thus this plan can be thought to be an VPS. But for free users, it provides a limited web app container, which is quite similar to Heroku that a terminal tool provided for remote configuration. However, unlike Heroku, the network speed of accessing OpenShift is super slow, thus it seems worse than Heroku in terms of the network speed.

4.3.3 OpenStack. First of all, it is not easy to register that it requires lots of information on affiliation. Thus I struggled on the



(a) View point (315, 45).

(b) View point (45, 45).

Figure 9: Two buoy layout, trained with two features: the distance between two buoys, and the angle between the line through two buoys and x-axis $\in [0°, 180°)$. x-axis and y-axis are the two training input features, z-axis is the power output (W). The black color represents the prediction power output map, while the rest colors represent the real one.

registration for over 20 minutes but stilled failed with "invalid captcha" every time on 28th May. Hence, I decide not to use this service.

4.3.4 Amazon Web Services. Amazon Web Services (AWS) are quite popular, and it is actually a kind of VPS service. In this project, VPS is always acceptable and usable in that any software can be installed freely. However, because of the same reason with Heroku, it requires binding credit card at the beginning which I do not want to proceed, I did not choose it.

4.3.5 Cheap VPS. This VPS provided by Sentris Network Ltd. is bought by me one year ago, and it lasts for three years. The specs of this server are:

- One core 1.1 GHz CPU;
- 128 MB memory;
- 1.0 GB disk quota, 60% of which are taken by Lubuntu [17].

It is ready-to-use and our web app service is quite light weight, thus I decide to use it directly. There are Nginx, PHP and SQLite running on this server already. In addition, as the result of using PHP Data Object (PDO), we can connect our codes to any database.

The most important thing is that almost every calculation job is finished on HPC, therefore, the server requirement is super low.

4.4 Server-side functions

In this project, there are two remote servers involved: the web app hosting server and HPC.

As is shown in Figure 2, Section 2, the web server is quite light-weight, just like a middleware for accessing HPC. Hence the functions in web server are actually forwarding requests to HPC.

For HPC, it is installed with all the software we need: Python 3.5.2, Java 1.8, WEKA 3.7, MATLAB 2017, etc. It took quite a bit time since it was not the same as normal operating system, and more details are mentioned in Section 4.4.2.

4.4.1 Web server development. After the server selection, I worked on the connection from user's browser to server, then to HPC. The connection between browser and server are under HTTPS, whose certification is granted by "Let's Encrypt" [3]. And the connection between server and HPC are under SSH (Secure SHell), therefore, all the connections in this system are secured. Due to the security issues, the backup plan will be set to be at every day 3 am, backing up databases into a HPC folder. Our latest live demo can be access at: https://mse.mewx.org/.

Technique stack. Front-end codes were written by Mengyu, and the used technique are very typical: JavaScript/jQuery, HTML5, CSS3(LESS).

Server programs are written in PHP 5.6, because it is temporary and very quick to write. We might change the programming language in the next semester as well. PHP in the server is running on a light-weight web server: Nginx. Although it is said to be light-weight, it is still powerful enough to drive the whole website. To enable HTTPS in Nginx, the HTTPS certification is granted by "Let's Encrypt".

In terms of database, we currently use PHP Data Object (PDO) to connect to database. Using PDO, we can connect to any database program without modifying any codes, and as well, we are currently limited by the cheap VPS hardware, we are now using SQLite.

All the techniques mentioned above are running on web hosting server. On HPC, the techniques we are using are: Python 3.5.2, Java 1.8, WEKA 3.7, MATLAB 2017.

Those are the techniques we are using in current project.

Code deployment. We maintain a branch for deploying which is called "web-app-UI", every time we need to update the codes, we just need to pull the latest codes. A very convenient thing is that GitHub allows to set deploy key, it reduces the process of entering password in production environment.

4.4.2 HPC development. HPC contains two main partitions: home partition and fast partition. I, the HPC developer, have 10

[3] "Let's Encrypt" is a free HTTPS certification provider: https://letsencrypt.org/.

GB quota on home partition, and 4TB quota on fast partition. That means only the important data should be stored in home partition. Therefore, in the next semester, the backed-up database files will be stored in home partition.

HPC operating system (OS) is quite different from other OSs, and it does not have a widely used packaging tool, like "yum", "apt-get", "yast", etc. Instead, the official supported pack manager is called "EasyBuild", which is used to install software in users' directories. However, in most cases, it is very easy to use, when I tried to install "scikit-learn" package, it reminded me to install more than 20 dependencies. And there were conflicts between the installed package version and will-be-installed package version. That was quite annoying, thus I skipped using "EasyBuild" to install "scikit-learn". I used "pip install –user" to install python package without root access. Only in this way solved installing "scikit-learn" easily. The rest packages were easy to install with "EasyBuild" and worked perfectly.

In the whole project, I used HPC a lot, especially evaluating buoy coordinates using CFD models. HPC takes a script for each job, as the result of that CFD models are single-threading programs, I split input data intro hundreds of scripts. By this pseudo-parallel computation method, I got 100, 000 evaluations in five hours.

In terms of the interaction between web app server and HPC. HPC does not have an web server program running, and with "EasyBuild", it is still not allowed to run. Therefore, the only way to interact with HPC is to activate corresponding scripts on HPC. I am responsible for writing them and I call those scripts "handler scripts" which must be written in CSH (C SHell). Once web app server sends an "ssh connection request" and the connection is established, the scripts will be invoked to complete a set of actions, like training model and predicting model.

5 EVALUATION

5.1 Evaluating machine learning models

There are many ways to evaluate machine learning models. The most popular way is mean squared error (MSE). Lots of papers use MSE to evaluate their models, for example, Bartz-Beielstein and Zaefferer's research [8] uses MSE to evaluate all their models (linear, ensemble, etc.).

The way we evaluate our trained models comes from "WEKA" which is another machine learning tool used by Chenwei. Its outputs contain the RRSE value, which is derived from root mean squared error (RMSE) and said to be more sensitive to outliers (bad predictions), and this is exactly what we want. [4]

This is the expression of RRSE:

$$RRSE = \sqrt{\frac{\sum_{i=1}^{N}\left(\hat{\theta}_i - \theta_i\right)^2}{\sum_{i=1}^{N}\left(\bar{\theta} - \theta_i\right)^2}} \tag{1}$$

The smaller the value is, the better the model performs. What we are keep working on is to reducing the RRSE indicators of our machine learning models.

[4]The derivation steps can be found here: https://stats.stackexchange.com/a/131273/129956.

5.2 Evaluating the web app

The web app program is currently super simple which handles only 4 types of request, and it is stable because it passes requests to HPC directly.

The only issue that we find is that the "SSH connection" takes very long time for establishing connection and transferring data. In our web app, there is a function that user can drag and drop buoy icons and give real-time prediction values. We measured the average time for each prediction with 5 runs, it was 4.13 seconds with a 155 MB model file; whereas the same run on HPC instantly took about 1.77 seconds. Therefore, this is the potential improvement that we need to spend time on.

6 NEXT STEPS

Regarding to the properties random forest regressor, the more estimators it has, the more accurate the final model will be. As well, the more training data we get, the more accurate the model will be. Therefore, HPC will be used for further hypothesises and attempts. These works will be tried and finished before the start of next semester.

"TensorFlow" is also in our supporting plan and it's super popular now, but the learning curve is steep. However, it's worth trying and it's very friendly to Python developers which is very similar to "scikit-learn". This will be done within the first six weeks of next semester.

The web app server configuration is almost done, and it is very stable now. As well, I did not put too much calculation on the server, thus only persistent storing should be taken into consideration. Therefore, I need to spend more times on deciding the suitable database service. This work is supposed to be done within the first four weeks which is more urgent than applying "TensorFlow".

As is mentioned in Section 5.2, the networking issues are the bottle necks of current system, which takes too much time on networking. The potential reason might be that our current server is located in western America and it takes double distance from Australia to America for each request, but this is an important issues that we need to solve within next semester. This is not urgent because we treat it as an extension, and will be looked after the forth week in next semester (after solving database issue).

Currently, the web app server program is just a simple PHP file, and no framework used or pattern followed. In the following times, I should select a popular framework and redesign the program structure to make everything following software engineering patterns. Building web app is the top task and will run through the whole next semester. The new technique stack will be decided within the first four weeks in next semester.

7 CONCLUSION

It is very priceless chance to know and work on the new type of renewable energy - wave energy. And it looks so powerful that I want to contribute to the development of it.

In this project, I learned the methodology of doing a research, and I was involved in training a machine learning regression model, which is not like the classification problems I learnt in "Introduction to Statistical Machine Learning" course. Then, we are trying to

improve it via different approaches and get significant progress to some extend.

I programmed most machine learning components in Python 3.5, and web app server program in PHP 5.6, because both of them can help me to make thing work in a short time. Although, PHP 5.6 is quite old, it is now a temporary solution. In the next semester, new techniques will definitely be used, like PHP 7.1 or else.

In addition, it is my first time to work with HPC and new PaaS platforms like Heroku and OpenShift, thus some configuration confused for me, however, I overcame them and gained lots of precious experience.

To conclude, this project is challenging and in the front line, and I have learnt lots of new things from this project so far. In the next semester, with less courses disrupting me, I can spend much more time in discovering new things from the project.

REFERENCES

[1] J. Wu, S. Shekh, N.Y. Sergiienkoy, B.S. Cazzolatoy, B. Ding, F. Neumann, M. Wagner, Fast and Effective Optimisation of Arrays of Submerged Wave Energy Converters, *Proceedings of the Genetic and Evolutionary Computation Conference*, 2016.

[2] D.R. Arbonès, B. Ding, N.Y. Sergiienko, M. Wagner, Fast and Effective Multi-Objective Optimisation of Submerged Wave Energy Converters, *Parallel Problem Solving from Nature*, pp.675-685.

[3] B. Ding, L.S. Silva, N.S., F. Meng, J. D. Piper, L. Bennetts, M. Wagner, B. Cazzolato, M. Arjomandi, Study of fully submerged point absorber wave energy converter - modelling, simulation and scaled experiment, *The 32nd International Workshop on Water Waves and Floating Bodies*, Dalian, China, 23-26 April, 2017.

[4] J.T. Scruggs, S.M. Lattanzio, A.A. Taflanidis, I.L. Cassidy, Optimal causal control of a wave energy converter in a random sea, *Applied Ocean Research*, vol.42, 2013.

[5] G.X. Wu, The interaction of water waves with a group of submerged spheres, *Applied Ocean Research*, vol.17, 1995.

[6] scikit-learn: machine learning in Python, http://scikit-learn.org/stable/

[7] M. Pilát, R. Neruda, Feature Extraction for Surrogate Models in Genetic Programming, *Parallel Problem Solving from Nature*, 2016.

[8] T. Bartz-Beielstein, M. Zaefferer, Model-based methods for continuous and discrete global optimization, *Applied Soft Computing*, 2017.

[9] J.C. Fernández, S. Salcedo-Sanz, P.A. Gutiérrez, E. Alexandre, C. Hervás-Martínez, Significant wave height and energy flux range forecast with machine learning classifiers, *Engineering Applications of Artificial Intelligence*, vol.43, 2015.

[10] G. Reikard, B. Robertson, J. Bidlot, Wave energy worldwide: Simulating wave farms, forecasting, and calculating reserves, *International Journal of Marine Energy*, vol.17, pp.156-185, 2017.

[11] S. Dripta, C. Emile, V. Nicolas, D. Frederic, Prediction and optimization of wave energy converter arrays using a machine learning approach, *Renewable Energy*, Vol.97, pp.504-517, 2016.

[12] D. Song, S. Gao, M. Xu, X, Wang, Y Wang, Hardware design of a submerged buoy system based on electromagnetic inductive coupling, *MATEC Web of Conferences*, 2016.

[13] Buoy - Wikipedia, https://en.wikipedia.org/wiki/Wave_power.

[14] Heroku: Cloud Application Platform, https://www.heroku.com/.

[15] SinaAppEngine (SAE), https://sae.sina.com.cn/.

[16] OpenShift: Container Application Platform by Red Hat, Built on Docker and Kubernetes, view-source:https://www.openshift.com/.

[17] lubuntu - lightweight, fast, easier, http://lubuntu.net/.