Project Report: Source localization in an ocean waveguide using supervised machine learning

Yuwei Li, Ruting Yin, Wen Liang, Chao Yu

June 16, 2017

Abstract

Machine learning methods is used in source localization problem in ocean acoustics to obtain source ranges directly from observed data. The complex pressure received by a vertical linear array is preprocessed to construct normalized sample covariance matrices (SCMs) and used as the input. The range estimation problem is solved both as a classification problem and as a regression problem by three machine learning methods (feed-forward neural networks (FNN), support vector machines (SVM) and random forests (RF)), with focus on the FNN. The testing results for experimental data are calculated to compare FNN, SVM, RF and demonstrate the feasibility of machine learning methods for underwater source localization.

1 Introduction

Source localization in ocean acoustic is often solved by matched-field processing (MFP). Establish the sound propagation model and compare the modeled sound pressure and observations. However, due to its sensitivity to the mismatch between model-generated replica fields and measurements. MFP can only provide reasonable prediction only if the ocean environment can be accurately modeled. Unfortunately, it is widely known that the ocean environment in reality is complicated and difficult to control [3].

Finding features directly from data is a good method to solve source range localization. Machine learning technologies have drawn much attention due to increased computational resources as well as their ability to learn nonlinear relationships. Our project applies the use of current machine learning approaches for source range localization, the feed-forward neural network (FNN), support vector machine (SVN) and random forest (RF).

In our work, the well-developed machine learning libraries are adopted. TensorFlow is used to implement FFN. Due to its simple architecture and wide user case [3], it is easy to use with just modeling the neural network as a graph. In addition, improved optimization algorithms with better convergence, more robust model with high computational efficiency give TensorFlow a more obvious advantage. Since TensorFlow version doesn't include SVM and RF, Scikit-learn is used to implement SVM, RF and Feature Selection, which is a simple and efficient tools for data analysis and machine learning.

2 Source range localization through machine learning

The dynamics of the ocean has an stochastic influence between the received pressure phase and amplitude at the array and the source range. Assuming a deterministic relationship between ship range and sample covariance matrix, we preprocess the received pressure and apply it as the input data of the machine learning models. After preprocessing we assume a deterministic relationship between ship range and sample covariance matrix, which is be implicit but can be discovered through machine learning methods. The received pressure is preprocessed and used as the input of the machine learning models (Sec. $\mathbf{2.1}$). The desired output may be either discrete (classification) or continuous (regression)(Sec. $\mathbf{2.2}$). The theory of FNN, SVM, and RF are described in Secs. $\mathbf{2.3}-\mathbf{2.5}$.

2.1 Input data preprocessing

The original data we get are samples of complex pressure $\mathbf{p}(t)$ the sensor array detects in the time series and the GPS data(represents a distance, denoted as $d(t)$) as the label. To make the processing independent of the complex source spectra, the received array pressure is transformed to a normalized sample covariance matrix [3].

At first, we average $\mathbf{p}(t)$ over N_s snapshots [1].

$$\mathbf{p}(t) = \frac{1}{N_s}\sum_{s=1}^{N_s}\mathbf{p}(t+(s-\left\lceil\frac{N_s}{2}\right\rceil)t_s) \tag{1}$$

where t_s represents the interval of snapshots and 0 lies at the center of the sequence $s-\left\lceil\frac{N_s}{2}\right\rceil$, $s=1,\cdots,N_s$. Next we take DFT of $\mathbf{p}(t)$ in a snapshot to get complex pressure at frequency f, denoted by $\mathbf{p}(f)$. To reduce the effect of the source amplitude, $\mathbf{p}(f)$ is normalized according to

$$\tilde{\mathbf{p}}(f) = \frac{\mathbf{p}(f)}{\|\mathbf{p}(f)\|_2} \tag{2}$$

The normalized sample covariance matrices(SCMS) is produced according to

$$\mathbf{C}(f) = \tilde{\mathbf{p}}(f)\tilde{\mathbf{p}}^H(f) \tag{3}$$

where H denotes conjugate transpose operator. Because $\mathbf{C}(f) = \mathbf{C}^H(f)$, only the real and imaginary parts of the

[1] It's different from Niu et al's paper [3] and is inferred from the matlab code.

complex valued entries of diagonal and upper triangular matrix in $\mathbf{C}(f)$ are used as input. These entries are vectorized to form the real-valued input x of size $L \times (L+1)$ to the FNN, SVM and RF [3].

2.2 Source range mapping

Before implementing the model, we need to do source range mapping. For the classification problem, the total source range are segmented into K bins, $r_1, \cdots, r_K$(r_k is a number, but here it represents the interval $[r_k - \frac{\Delta r}{2}, r_k + \frac{\Delta r}{2}]$), of equal width Δr.

For SVM and RF, denote the label for each input vector $\mathbf{x}_n$ as t_n. The mapping from original data d_n(GPS distance) to label t_n is

$$t_n = r_k \ s.t. \ |d(n) - r_k| \leq \frac{\Delta r}{2} \tag{4}$$

For FNN, the label is a $1 \times K$ binary vector, $\mathbf{t}_n$. The mapping is

$$t_{nk} = \begin{cases} 1 & if \ |d(n) - r_k| \leq \frac{\Delta r}{2} \\ 0 & otherwise \end{cases} \tag{5}$$

For the regression problem, we just use continuous range variable d_n [2] as true source range and target output for all three models.

2.3 Feed-forward neural networks

The structure of feed-forward neural network (FNN) see Fig 1. Here, FNN is constructed by three layers((input layer L_1, hidden layer L_2 and output layer L_3). Assume the input layer has D units, i.e. D input variables, then the input a_j of the j-th unit in the hidden layer L_2 is

$$a_j = \sum_{i=1}^{D} w_{ji}^{(1)} x_i + w_{j0}^{(1)}, \ j = 1, 2, \cdots, M \tag{6}$$

where M is the number of units in the hidden layer. Parameters $w_{ji}^{(1)}, w_{j0}^{(1)}$ are called weights and bias, respectively and their linear combinations a_j are called activations. In the hidden layer, activations z_j are transformed to output z_j using an activation function $f(\cdot)$.

$$z_j = f(a_j) \tag{7}$$

Here, sigmoid function is chosen, see Fig. 1(b).

$$f(a) = \sigma(a) = \frac{1}{1 + e^{-a}} \tag{8}$$

Similarly, the input b_k of the k-th unit in the output layer is

$$a_k = \sum_{j=1}^{M} w_{kj}^{(2)} z_j + w_{k0}^{(2)}, \ k = 1, 2, \cdots, K \tag{9}$$

[2] we use d_n here rather than r_n in [3] to avoid aliasing

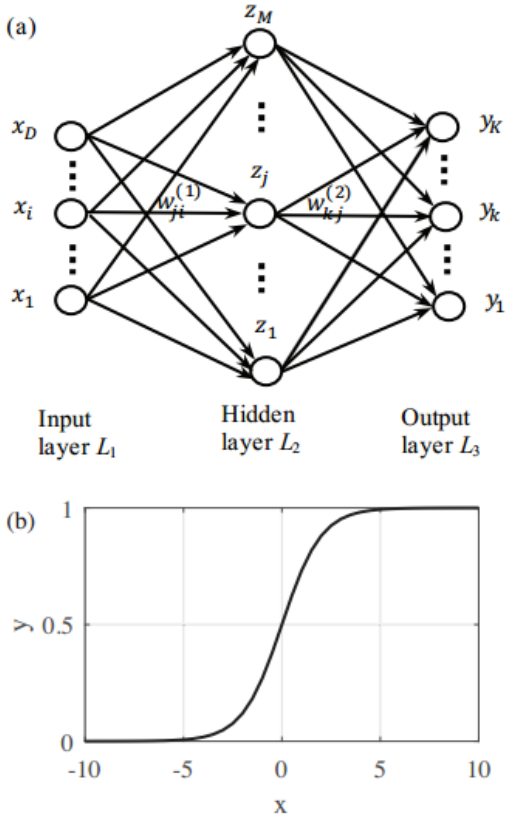


Figure 1: (a) Diagram of a feed-forward neural network and (b) Sigmoid function. The figure is from [3].

where $w_{kj}^{(2)}$ and $w_{k0}^{(2)}$ represents the weights and bias.

In the output layer, the softmax function is used as the activation function. It is a common choice for multi-class classification problems [1]. Here, it constrains the output class, $y_k(\mathbf{x}, \mathbf{w})$, to be the probability that the source is at range r_k [1]:

$$y_k(\mathbf{x}, \mathbf{w}) = \frac{exp(b_k(\mathbf{x}, \mathbf{w}))}{\sum_{j=1}^{K} b_j(\mathbf{x}, \mathbf{w})}, \ k = 1, \cdots, K \tag{10}$$

where $\mathbf{w}$ is set of all weights and bias. $y_k \geq 0$, $\sum_k y_k = 1 \Rightarrow$ y_k can be regarded as probabilities.

When training the model, a cross entropy loss E_n is used.

$$E_n(\mathbf{t}_n, \mathbf{y}_n(\mathbf{x}, \mathbf{w})) = -\sum_k t_{nk} \ln(y_{nk}) \tag{11}$$

Average it on N observations,

$$E(\mathbf{w}) = -\frac{1}{N} \sum_{n=1}^{N} \sum_{k=1}^{K} t_{nk} \ln(y_{nk}) \tag{12}$$

Resulting weights and biases are

$$\hat{\mathbf{w}} = \arg\min_{\mathbf{w}} \left[-\frac{1}{N} \sum_{n=1}^{N} \sum_{k=1}^{K} t_{nk} \ln(y_{nk}) \right] \tag{13}$$

When predicting using the trained model, the source range is decided to be at the most probable range, i.e., the predicted label $\mathbf{t}'$ for the input data $\mathbf{x}$ is

$$t'_k = \begin{cases} 1 & if\ k = \arg\max_k y_k(\mathbf{x}, \hat{\mathbf{w}}) \\ 0 & otherwise \end{cases} \tag{14}$$

For the regression case, there is only one unit in the output layer, which represents the continuous label d_n. A sum-of-squares error [1] is minimized instead of Eq. 12

$$E(\mathbf{w}) = \frac{1}{2N} \sum_{n=1}^{N} |y_n(\mathbf{x}_n, \mathbf{w}) - d_n|^2 \tag{15}$$

where d_n is the true source range at sample n. Resulting weights and biases are

$$\hat{\mathbf{w}} = \arg\min_{\mathbf{w}} \left[\frac{1}{2N} \sum_{n=1}^{N} |y_n(\mathbf{x}_n, \mathbf{w}) - d_n|^2 \right] \tag{16}$$

When predicting using the trained model, the output of the unit in the output layer is taken as the source range, i.e., the predicted label d' for the input data $\mathbf{x}$ is

$$d' = y(\mathbf{x}, \hat{\mathbf{w}}) \tag{17}$$

2.4 Support Vector Machine

For Support Vector Machines(SVM), the input data are separated into two(or more) classes by defining a separating hyperplane that maximally separates the classes [3].

See Fig. 2, for linearly separable input data $\mathbf{x}_1, \cdots, \mathbf{x}_N$, we seek a hyperplane defined by $(\mathbf{w}, b)$

$$\mathbf{w}^T\mathbf{x} + b = 0 \tag{18}$$

For every $(\mathbf{w}, b)$, define a margin d_M

$$d_M = \min_n \frac{|\mathbf{w}^T\mathbf{x_n} + b|}{\|\mathbf{w}\|_2} \tag{19}$$

SVM maximizes the margin while successfully separate the two classes of input data.

$$\begin{aligned} &\max_{\mathbf{w},b} d_M \\ &s.t.\ y_n(\mathbf{w}^T\mathbf{x}_n + b) \geq 0,\ n = 1, \cdots, N \end{aligned} \tag{20}$$

Eq. 20 can be transformed to

$$\begin{aligned} &\min_{\mathbf{w},b} \tfrac{1}{2}\mathbf{w}^T\mathbf{w} \\ &s.t.\ y_n(\mathbf{w}^T\mathbf{x}_n + b) \geq 1,\ n = 1, \cdots, N \end{aligned} \tag{21}$$

If the training set is linearly non-separable, slack variables ξ_n are introduced to allow some of the training points to be misclassified. In this case, the optimization problem becomes

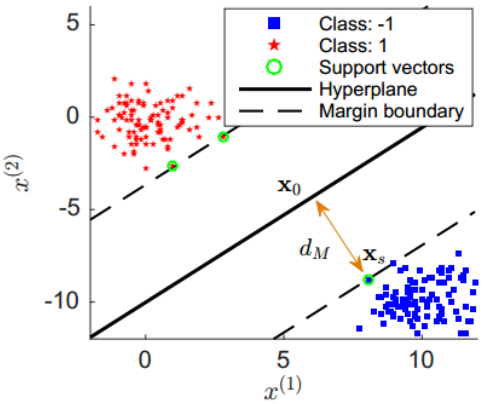


Figure 2: A linear hyperplane learned by training an SVM in two dimensions (D = 2). The figure is from [3].

$$\begin{aligned} &\min_{\mathbf{w},b} \tfrac{1}{2}\mathbf{w}^T\mathbf{w} \\ &s.t.\ y_n(\mathbf{w}^T\mathbf{x}_n + b) \geq 1 - \xi_n,\ n = 1, \cdots, N \\ &\xi_n \geq 0,\ n = 1, \cdots, N \\ &\sum_{n=1}^{N} \xi_n \leq Z \end{aligned} \tag{22}$$

where Z is a empirical parameter.

For non-linear classification problem, kernel trick is used. In this study, a radial basis function(RBF) kernel is used:

$$K(\mathbf{x}, \mathbf{x}') = exp(-\gamma\|\mathbf{x} - \mathbf{x}'\|_2^2) \tag{23}$$

γ is a parameter here set to $1/K$, where K is the number of total classes.

Because SVM model only separate two classes, for the K-classes case, we train $K(K-1)$ models on all possible pairs of classes. The points that are assigned to the same class most frequently are considered to be in the same class. This approach is known at the "one-versus-one" scheme [1].

For Support Vector Regression(SVR), a $\epsilon-$sensitive error function is minimized

$$\varepsilon_\epsilon(y_n - d_n) = \begin{cases} 0, & if\ |y_n - d_n| < \epsilon \\ |y_n - d_n| - \epsilon, & otherwise \end{cases} \tag{24}$$

where y_n is the predicted source range and d_n is the true source range at sample n. In SVR, the support vectors are points outside the ϵ region.

2.5 Random forests

The random forest(RF) trains decision tree models on randomly selected subset of input data and get results by averaging over those models. So random forest is a more robust generalization of decision tree model.

Consider a decision tree(see Fig. 3) trained on all the input data. The input data can be partitioned into two regions by defining a cutoff along the i-th dimension.

$$\begin{aligned} \mathbf{x}_n \in \boldsymbol{x}_{left} \quad & if\ x_{ni} < c, \\ \mathbf{x}_n \in \boldsymbol{x}_{right} \quad & if\ x_{ni} \geq c. \end{aligned} \tag{25}$$

c is the cutoff value and $\boldsymbol{x}_{left}$, $\boldsymbol{x}_{right}$ represents the left and right regions. c is decided by minimizing the cost function $G(c)$

$$G(c) = \frac{n_{left}}{N} H(\boldsymbol{x}_{left}) + \frac{n_{right}}{N} H(\boldsymbol{x}_{right}) \tag{26}$$

n_{left}, n_{right} are number of samples in the left and right region. $H(\cdot)$ is an impurity function.

For the classification problem,

$$H(\boldsymbol{x}_m) = \frac{1}{n_m} \sum_{\mathbf{x}_n \in \boldsymbol{x}_m} I(t_n, \ell_m) \left[1 - \frac{1}{n_m} I(t_n, \ell_m) \right] \tag{27}$$

where n_m is the number of samples in region $\boldsymbol{x}_m$ and ℓ_m represents the assigned label for each region, corresponding to the most common class in the region [2]:

$$\ell_m = \arg\max_{r_k} \sum_{\mathbf{x}_n \in \boldsymbol{x}_m} I(t_n, r_k) \tag{28}$$

$$I(t_n, r_k) = \begin{cases} 1 & if\ t_n = r_k, \\ 0 & otherwise. \end{cases} \tag{29}$$

The remaining regions are partitioned iteratively until regions $\boldsymbol{x}_1, \cdots, \boldsymbol{x}_M$ are defined. In this paper, the number of regions, M, is determined by the minimum number of points allowed in a region which is set to 50 in this papre [3]. Fig. 3 shows a decision tree with $M = 3$ and cutoff values 1.9 and 4.6.

To conclude, in the training process of a decision tree, we partition the region $\boldsymbol{x}_m$ continuously, which is assigned a label ℓ_m each and finally get a series of cutoff values c, which is the parameters of the decision tree model.

When predicting using the trained decision tree, test input data $\mathbf{x}$ is assigned to a region $\boldsymbol{x}_m$ according to those cutoff values. Then the predicted label $\hat{f}^{tree}(\mathbf{x})$ for it it is just the label assigned to the region.

$$\hat{f}^{tree}(\mathbf{x}) = \ell_m \tag{30}$$

For RF regression, training and testing are the same as in the classification problem except that we use the mean of the true class for all points in the region as the assigned label for the region and the mean squared error [3] as the impurity function.

$$H(\boldsymbol{x}_m) = \frac{1}{n_m} \sum_{\mathbf{x}_n \in \boldsymbol{x}_m} (d_n - \ell_m)^2 \tag{31}$$

$$\ell_m = \frac{1}{n_m} \sum_{\mathbf{x}_n \in \boldsymbol{x}_m} d_n \tag{32}$$

where d_n is the true source range at sample n.

[3] The formula here is different from the one in [3] with a factor $\frac{1}{n_m}$, which is necessary by contrast to eq. 26.

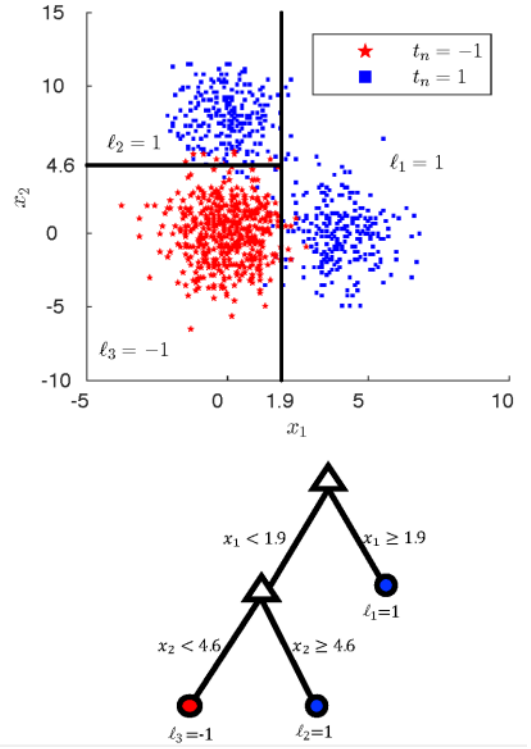


Figure 3: Decision tree classifier and corresponding rectangular regions shown for two–dimensional data with $K = 2$ classes ($D = 2, M = 3$) and 1000 training points. The figure is from [3].

As the decision tree may overfit the data, we use a more robust model, random forest. We draw B subset of data uniformly at random from the full training set and for each draw a decision tree is fitted to the subset of data.

When predicting using the trained model, for the classification problem, test data $\mathbf{x}$, is assigned to its most frequent class among all draws:

$$\hat{f}(\mathbf{x}) = \arg\max_{r_k} \sum_{b=1}^{B} I(\hat{f}^{tree,b}(\mathbf{x}), r_k) \tag{33}$$

where $\hat{f}^{tree,b}(\mathbf{x})$ is the class assigned to $\mathbf{x}$ in the b-th tree.

While for the regression problem, the predicted source range of test data $\mathbf{x}$ is the average of labels produced in all draws:

$$\hat{f}(\mathbf{x}) = \frac{1}{B} \sum_{b=1}^{B} \hat{f}^{tree,b}(\mathbf{x}) \tag{34}$$

where $\hat{f}^{tree,b}(\mathbf{x})$ is the source range assigned to $\mathbf{x}$ in the b-th tree.

2.6 Source localization algorithm

Algorithm to solve localization problem:

1.Data preprocessing. According to the type of problem(classification or regression) and type of model(FNN,

SVM or RF), preprocess the input data and source range according to Sec. **2.1**, **2.2**.

2.Train the model.

3.Use the trained model to predict the source range for the unlabeled test data. The resulting output is mapped back to range, and the prediction error is reported by the mean absolute percentage error (see Sec).

3 EXPERIMENTAL RESULTS

3.1 Introduction to the experiment

The dataset we used contains acoustic data with time series and the location data GPS range. According to the previous section, we derived the input SCMs from the acoustic data as input for the machine learning algorithms. The SCM for a n-elements vertical array should be a $n \times n$ symmetric matrix with both real and imaginary parts. Then, we can convert this SCM into a $n \times (n+1)$ dimensional data for a single frequency. For multi-frequency data, the dimension will be extended to $k \times n \times (n+1)$, where k is the number of frequencies.

In classification problem, we assigned the each location in GPS range data to a specific class. For regression problem, we use the original GPS location data. We were provided with 5 datasets. We mainly used dataset01 and also used other 4 datasets to represents the generalization of the methods.

To evaluate the prediction results, we used the mean absolute percentage error (MAPE) which is defined as

$$E_{MAPE} = \frac{100}{N} \sum_{i=1}^{N} |\frac{R_{p_i} - R_{g_i}}{R_{g_i}}|,$$

where the R_{p_i} is the prediction range and the R_{g_i} is original range data. We can see that the M_{MAPE} evaluate the ratio of absolute difference between prediction and truth range to the truth range. Therefore, it is a proper evaluation method for continuous values.

In this section, we used FNN, SVM and random forests methods to do the same source localization task, compared the performance of each method for this machine learning task, tried to model the problem as both classification and regression problem, tried to add noises to the original acoustic data and represented some character of the FNN.

In the next section, we dived into more about feed forward neural network.

3.2 FNN

In figure 4, we can see that the error rate on training set and test set are decreasing through epochs but also has small vibrations.

As described in the last section, the output of FNN is a prediction of probability distribution over all the given classes. In the figure 5, the prediction result of the first epoch is rather noisy and the "spike" of the maximum argument is wide, which means the neural network is unsure what the result is. In the following epochs, the neural network keep learning from the training data and continuously perfect the

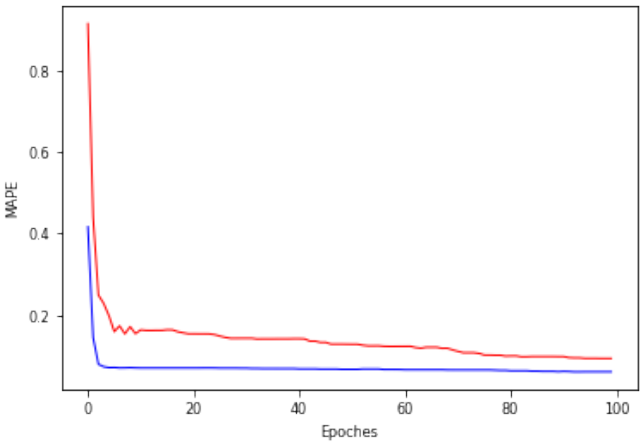


Figure 4: FNN learning curve

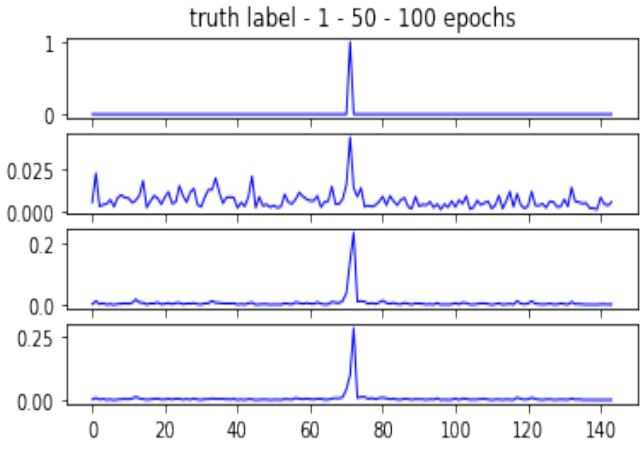


Figure 5: Output probability distribution of truth label and predictions after epochs (1, 50, 100)

performance so we can find a thinner "spike" and less probability on other classes. This illustrates that the training process keeps improving the FNN.

We did not find too much overfit in this original dataset. Generally, the MAPE and accuracy on test set and training set decrease together and both have very low MAPE. Thus, we can hypotheses that the datasets provided have rather low level of noises. Therefore, although we did not know the specific SNR of the original data, we have reasons to add noise (very small random number) to the dataset, get the leaning curve and analyze the noise and overfit. For noisy dataset, we should carefully control the learning process of FNN by adding more regularization to parameters, trying more times and keeping track of the learning curve. Then, we got the result in figure 6.

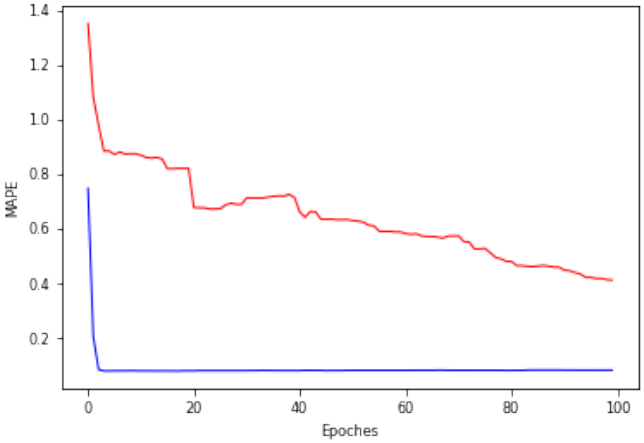


Figure 6: FNN learning curve with noisy dataset

For this "dirty" dataset, the FNN classifier performs great for training set but generates poor predictions on test set, which indicates overfitting. The noise added to the original dataset makes it harder for classifiers catch the relationship between input data and labels.

3.3 Single frequency SCM inputs and multi-frequency input

For single frequency SCM data, they are extracted from the original acoustic data at 550Hz and 950Hz separately. The multi-frequency SCM data are formed by concatenating multiple single-frequency SCM data into one vector. Therefore, the input data is much more than the single frequency SCMs. To verify whether more multi-frequency input provides more useful information and increase the performance. Here, we compared the performance from 2 datasets and used 3 different frequencies.

The FNN range prediction results for single frequency SCM data (550Hz and 950Hz separately) are shown in the first 2 rows in the figure 7 and the results for multi-frequency SCM data (300-950Hz with 10Hz increment, i.e. 66 frequencies) are given in the last row. Here, we use MAPE to evaluate the results. The MAPE for 550Hz single frequency data are 56% and 10%, for 950Hz single frequency data are 43% and 9%, for multi-frequency data are 19% and 4%. All those results are also given in the Table 4.

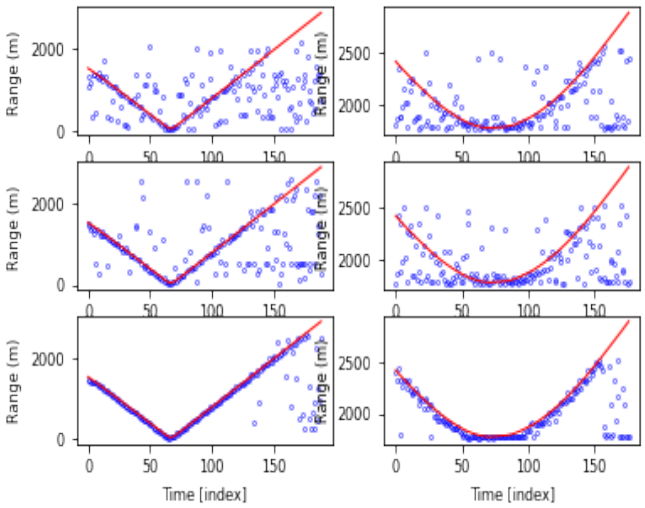


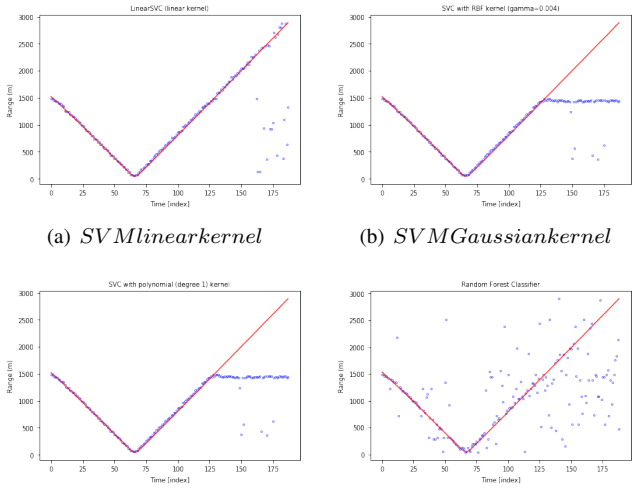
Figure 7: Range predictions on the test set of dataset01 and dataset02. Figures at first row 550Hz, Second row 950Hz, Third row 300-950Hz with 10Hz increment, i.e. 66 frequencies. The red line is the ground truth data and the blue dots are predictions

Compared to the single frequency SCMs results, the multi-frequency MAPE results are much lower and the prediction is much more accurate. Thus, the multi-frequency SCM input can provide more useful information and improve the performance of source localization classification model dramatically.

3.4 Support vector machine and random forests

In this section, we used SVM with linear, Gaussian, polynomial kernel and random forests algorithms to solve this source localization problem. SVM and random forests are very popular and powerful machine learning methods. Now, we compared their performance with the forward neural network.

Figure 8 shows range predictions on test set by SVM with linear kernel, Gaussian radial basis function kernel, 2-degree polynomial kernel and the results by random forests. The MAPE statistic of predictions is shown in Part II of Table 4. From the predictions MAPE results, we can see that SVM with linear kernel got a very good result. However, SVM using Gaussian kernel and polynomial kernel cannot predict well at right part of the dataset. The reason is that we have less training samples at the right part and the SVM using Gaussian kernel and polynomial kernel are vulnerable to unbalanced data and prone to overfit but the linear SVM only find linear boundaries for each classes and can catch the trend in the training data successfully. Moreover, random forests do not performs well for this classification task because the size of training set is too small but the number of classes is rather big. This situation may be hard for random forests to



(a) $SVMlinearkernel$ (b) $SVMGaussiankernel$

(c) $SVMpolynomialkernel$ (d) $randomforests$

Figure 8: The classification results from SVM using different kernels and random forests methods

solve. Compare with the MAPE with FNN, the FNN is more robust than SVM using Gaussian and polynomial kernel and the FNNs with more than 2 layers are better than linear SVM because FNN is more flexible and SVM suffers from the limitation of linear boundaries.

3.5 Source localization as a regression problem

In this section, we solve the source localization problem as a regression problem so the output is not classes but a continuous range. This time, the input training data and test data is same with the previous classification problem but the output y is GPS range data directly. Then we used FNN as regressor and got the result in figure 9.

The detailed prediction statistics of MAPE are also given in the table 4 part III. The MAPEs on test set for each regressor are 38%, 36% and 36%. Then we also used SVM and random forests to solve this regression problem and results are shown in figure 10.

Compare with classifiers, the FNN, SVM and RF degrade significantly for solving regression tasks. Thus, it is more proper to model the source localization as a classification problem.

4 Discussion on FNN

In the project, we test the performance of Forward-Feed Neural Network from different angles, such as number of snapshots, number of hidden layers and number of neurons for each layer. These experiments can give us some instructions of tunning hyperparameters of FNN model and deeper insight of this source localization problem.

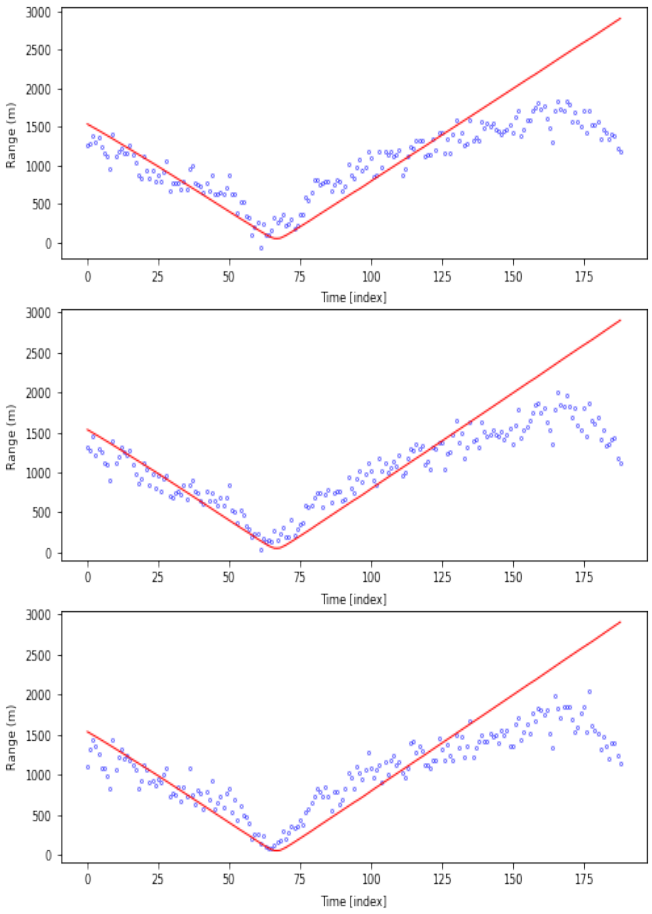


Figure 9: Source localization as a regression problem. Range prediction on test set of dataset01 by FNN for 300-950 Hz with 10 Hz increment, i.e. 66 frequencies. From top to bottom, the first figure is the result using 1 hidden layer FNN, the second one using 2 hidden layers FNN, the third one using 3 hidden layers. Each hidden layer consists of 512 neurons.

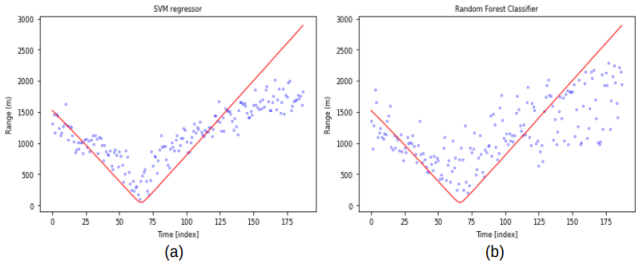
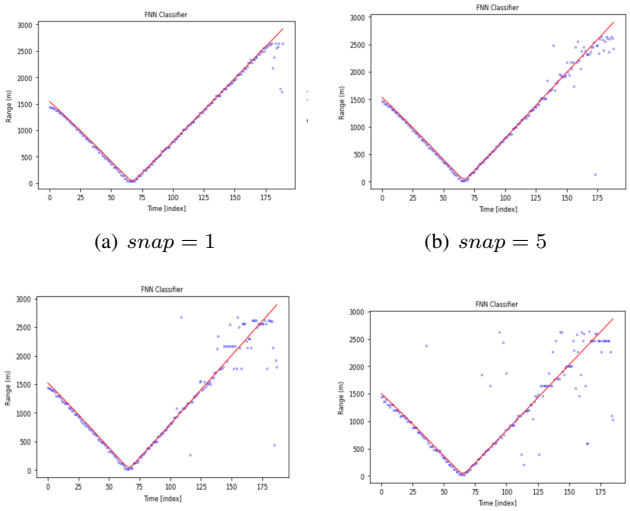


Figure 10: Source localization as a regression problem. Range prediction on test set of dataset01 by SVM and random forest for 300-950 Hz with 10 Hz increment, i.e. 66 frequencies. (a) SVM for regression, (b) RF for regression.

4.1 Number of snapshots

Then different number of snapshots have been tested in the project, Figure 11 shows the results with different number of snapshots of the same input data. When snapshots equals to 1, MAPE equals to 5.74293%. When snapshots equals to 20, MAPE equals to 16.826350% MAPE increases as the number of snapshots increases. The reason is because More snapshots average the noises in the data, but introduce mismatch if the source is moving or the environment is evolving. The results are shown in Table 1.



(a) $snap = 1$ (b) $snap = 5$

(c) $snap = 10$ (d) $snap = 20$

Figure 11: number of snapshots and MAPE

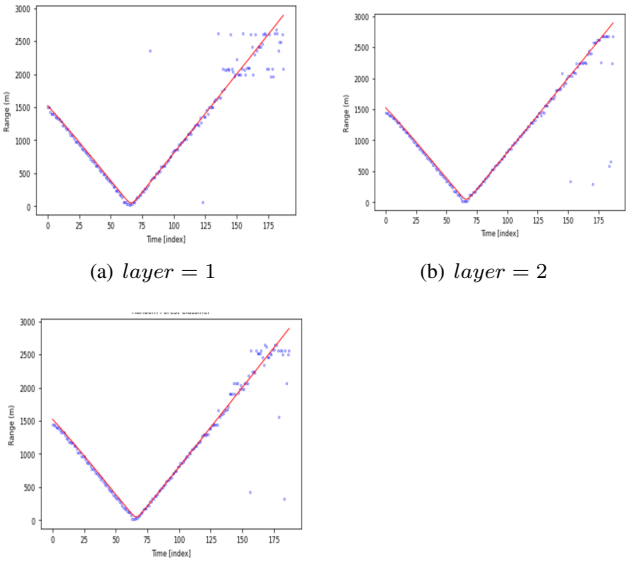
number of snap shots	MAPE(%)
1	5.74293
5	8.761923
10	10.01532
20	16.826350

Table 1: number of snapshots and MAPE.

From the results on test set, we can see that more snapshots introduce higher error, which implies for this data set with rather high SNR, the NN can handle the noises but suffers more from the bias caused by snapshots. However, If the dataset has more noises, more snapshots may help to reduce the effect of noises by averaging the data.

4.2 Number of layers

In addition,we test different number of layers to see how FNN performs. The number of hidden layers also affects FNN prediction performance.In Figure 12, the three plots shown the result of the FNN with 1, 2 and 3 layers with the data 300-950 Hz with 10 Hz increment. The MAPE are 11.08%, 7.64% and 7.03% corresponding to the number of hidden layers 1, 2 and 3. The results show that the MAPE decreases with the increase of number of hidden layers.



(a) $layer = 1$ (b) $layer = 2$

(c) $layer = 3$

Figure 12: number of layer and MAPE

number of hidden layer	MAPE(%)
1	11.083
2	7.639
3	7.0319

Table 2: number of hidden layers and MAPE.

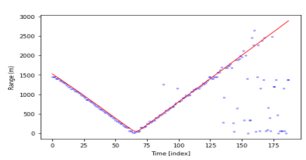
4.3 Number of Neurons

For each layer, different number of neurons for each layer have also been tested. Hidden layers plays a vital role in the performance of FNN. Thus, deciding the number of neurons in each hidden layer is significant issue while considering any complex problem. In this section, we tried and tested 64,128,256,512, 1024 number of neurons with one hidden layer forward neural network model to compare. In Figure 13, MAPE decreases as the number of neurons increases.

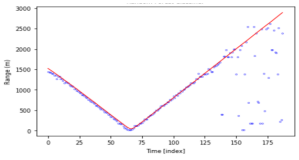
number of neurons	MAPE(%)
64	15.1452
128	13.2979
256	11.0830
512	9.81127
1024	8.7899

Table 3: number of neurons and MAPE.

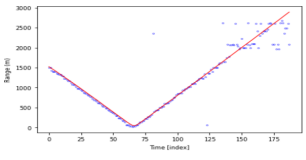
We also tried to use 2 hidden layer FNN to test and all the MAPE statistics are shown in table 4 part VI and VII.



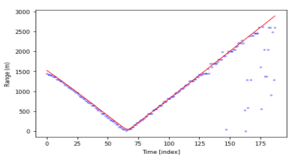
(a) $neuron = 64$



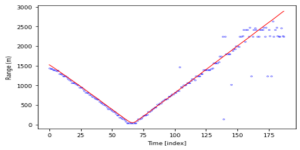
(b) $neuron = 128$



(c) $neuron = 256$



(d) $neuron = 512$

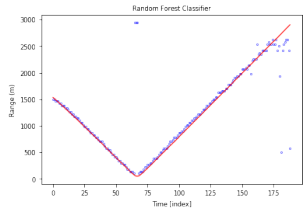


(e) $neuron = 1024$

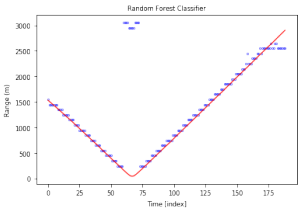
Figure 13: number of neurons for each layer and MAPE (1 hidden layer FNN)

4.4 Range resolution

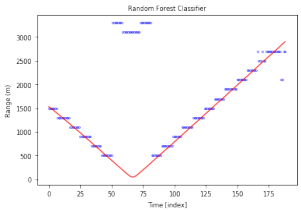
In this section, we decrease the original resolution by 2 times, 5 times and 10 times. As before, there are same number of input data, same number of neurons at input layer, 5 snapshots but different number neurons at output layer according to the number of labels which is related to the value of range and range resolution. Another difference is that, the number of samples for each classes is enlarged by assigning samples to less classes. The results are shown in figure 14.



(a) $2X$



(b) $5X$



(c) $10X$

Figure 14: Range predictions on dataset01 with 300-950 Hz input with different range resolution (2X, 5X, 10X) and 5 snapshots.

The MAPE statistics are provided in the table 4 part IV.

We can see that FNN performs better for 2X resolution test set, which implies that FNN can predict better with more samples for each classes. However, for other dataset, the biases eliminate the advantages and get higher MAPE.

5 Conclusion

This paper referenced the paper "Source localization in an ocean waveguide using supervised machine learning" and represented how to use machine learning methods to solve physics problems. The classic solution for those problems need complex physics and mathematic analysis and calculation. However, this paper introduced a supervised machine learning algorithm using some modern machine learning libraries such as Scikit-learn and Tensorflow, to solve acoustic source localization problem. Normalized SCMs are used as input vectors to the models. We mainly used FNN classifier model to solve this problem and got a very good performance. We also used SVM and random forests classifiers and regressors to compare with the model. The result shows that multi-frequency input helps the model to performs better. In the comparison between classifiers and regressors, it is more proper to model source localization as a classification problem rather than regression. Moreover, the FNN models have advantages of robustness and flexibility compared to the SVM and RF methods.

In the section 4, we concentrated on FNN and did a series of experiments on number of layers, number of hidden neurons, resolutions and number of snapshots. By analyzing the changes of performance, we got a deeper insight of application of feed-forward neural network used to solve source localization problem .

References

[1] C. M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.

[2] F. Pedregosa et al. Scikit-learn: Machine learning in python. *J. Mach. Learn*, 12:2825–2830, 2011.

[3] Haiqiang Niu, Emma Reeves, and Peter Gerstoft. Source localization in an ocean waveguide using supervised machine learning. *arXiv*, June 2017.

Table 4: Experiment results

Part	Frequency (Hz)	No. of hidden layers	No. of hidden neurons	No. of output neurons	No. of snapshots	MAPE (%)
I	550	1	256	144	10	95.152
	950	1	256	144	10	60.723
	300-950, $\Delta f = 10$	1	256	144	10	11.083
II	300-950, $\Delta f = 10$	SVM for classification				9.952
	300-950, $\Delta f = 10$	RF for classification				50.508
III	300-950, $\Delta f = 10$ (FNN Regression)	1	256	1	10	42.926
		2	256	1	10	38.004
		3	256	1	10	36.340
	300-950, $\Delta f = 10$	SVM for regression				60.912
	300-950, $\Delta f = 10$	RF for regression				91.134
IV	300-950, $\Delta f = 10$	1	256	72	5	9.388
		1	256	29	5	25.460
		1	256	15	5	37.941
V	300-950, $\Delta f = 10$	1	256	144	1	5.743
		1	256	144	5	8.762
		1	256	144	20	16.826
VI	300-950, $\Delta f = 10$	1	64	144	10	15.145
		1	128	144	10	13.298
		1	256	144	10	11.083
		1	512	144	10	9.811
		1	1024	144	10	8.790
VII	300-950, $\Delta f = 10$	2	64	144	10	8.191
		2	128	144	10	7.439
		2	256	144	10	7.084
		2	512	144	10	6.680