Investigate machine learning methods based on Source Localization in an ocean waveguide

Chen Du
PID: A53223649

Yi Zheng
PID: A53098162

Yuehao Jia
PID: A53100711

***Abstract*— Supervised machine learning methods have been introduced to learn the potential relationship between source range and acoustic data. For multi-frequency acoustic data, there are too many redundant features which will cause overfitting. Principal component analysis is one efficient way to reduce dimension of data while retaining variance. In this project, we investigate the acoustic source localization using support vector machine (SVM), random forest (RF) and feed-forward neural networks (FNN) combined with PCA. The performance of 3 methods are compared with different parameters and datasets. Results show that with suitable parameters, SVM has outstanding performance in classification but relatively poor performance in regression; FNN combined with PCA has good performance in both classification and regression. RF combined with PCA has better performance in regression but lower precision in classification compared with SVM. Generally speaking, PCA efficiently improves the performance of FNN and saves lots of the training time of RF.**

I. INTRODUCTION

Acoustic source localization is widely used in modern ocean engineering to localize ships or animals.[1][2] By measuring sound pressure, it is possible to obtain a source direction using various technologies. Since the unstable and complicated ocean environment is too difficult to be accurately modeled, the conventional matched-field processing (MFP)[3] is limited in some practical applications.[4]

In recent years, machine learning has drawn great attention to deal with problems in physical science. One significant advantage of machine learning is that algorithms directly learning features from data, making it possible to find the potential relationship between features and labels. Haiqiang et.al has applied supervised machine learning methods in acoustic source range localization, including support vector machine (SVM), random forest (RF), and feed-forward neural networks (FNN)[4]. Instead of model-generated fields, the acoustic observations are used to train the machine learning models and are preprocessed into normalized sample covariance matrices (SCM) as the input. The models include both classification and regression, based on TensorFlow using scikit-learn package[5]. However, considering multiple-frequency SCMs as input of FNN and RF, the data's feature is of high dimension. In this case, overfitting would be a problem if we train the model directly. One efficient way to deal with overfitting caused by high-dimensional data is the principal component analysis (PCA)[6], which could map the original data onto a lower dimensional space, retaining most of the variance. This method is widely used in preprocessing high dimensional data[7] [8].

In this project, we applied 3 kinds of supervised machine learning methods with PCA in acoustic source localization based on the work of Haiqiang et.al[4]. The performance of SVM, FNN and RF are compared with different preprocessing and model parameters. In particular, for preprocessing, we considered the selection of frequency and snapshot number. For PCA, we transform data into different dimensions. For SVM, we tuned the penalty parameter and scaling parameter, and compare the difference between two $scikit-learn$ inbuilt functions LienarSVC and SVC (kernel = $linear$), as well as different kernel functions. For FNN, we consider different number of hidden neurons. For RF, we tested different tree numbers and the maximum tree depths. For each method, both classification and regression are considered.

The rest of the paper consists of 3 parts: in section II we describe the mathematic models of the system including preprocessing, PCA, SVM, RF and FNN. In section III we introduce the experiment design and analyze the results. Finally we draw conclusion in section IV.

II. MATHEMATICAL SYSTEM MODELS

The system consists of 5 parts, including data preprocessing, principal component analysis, support vector machine, random forest, feed-forward neural networks. After preprocessing, the acoustic dataset is transformed into training sample vectors and test sample vectors. The labelled training samples are used to train the machine learning models, and the trained models are used to predict the labels of test data. The input of RF and FNN are mapped to low dimension by PCA before training and test.

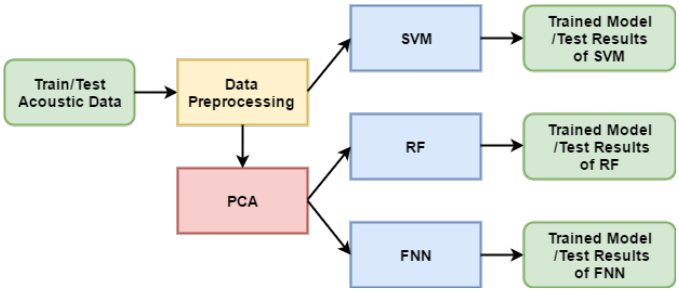

Fig. 1: Diagram of the system

A. Data Preprocessing

We use the data preprocessing method proposed by Haiqiang et.al [4], which could transfer the array pres-

sure time series into a normalized sample covariance matrix. The sound pressure at frequency f $p(f) = [p_1(f), p_2(f), p_3(f) \dots p_L(f)]^T$ is the DFT of the input pressure data at L sensors,the complex pressure is normalized to reduce the effect of source amplitude.

$$\tilde{p}(f) = \frac{p(f)}{\sqrt{\sum_{l=1}^{L} |p_l(f)|^2}} = \frac{p(f)}{\|p(f)\|_2} \tag{1}$$

"Then the normalized Sample Covariance Matrices(SCMs) is computed and averaged over Ns snapshots"[4]

$$C(f) = \frac{1}{N_s} \sum_{s=1}^{N_s} \tilde{p_s}(f) \tilde{p_s}^H(f) \tag{2}$$

Only the diagonal and upper triangular matrix in $C(f)$ are used as input, since $C(f)$ is complex matrix, the real and imaginary parts forms a real-valued input vector x of length $L \times (L+1)$. For multiple frequencies, the input vector is of size $N_f \times L \times (L+1)$.

In classification problem, the source ranges are discretized into K bins, r_1, r_2, r_3, $\dots$ r_K, of equal width Δr. Each input vector x_n, $n = 1, 2, 3, \dots N$ is labeled by t_n where $t_n \in r_K, k = 1, 2, 3, \dots K$. Specially for FNN, the input data is transformed into a $1 \times K$ binary vector t_n such that

$$t_k = \begin{cases} 1 \;\; if \;\; |t_n - r_k| \leq \frac{\Delta r}{2} \\ 0 \;\; otherwise \end{cases} \tag{3}$$

where $t_n = t_{n,1}, t_{n,2}, t_{n,3}, t_{n,\dots}, t_{n,K}$, therefore represents the expected output probability of the neural network. In the regression problem, the target output r_n is a continuous range variable for all three models.[4]

B. Principal Component Analysis

Principal Component Analysis invented by Karl Pearson [9], and later independently developed by Harold Hotelling[6]. For conventional PCA, given the preprocessed SCM samples $X^{N \times D}$, where N is the number of samples and D is the dimension of features of each sample, the scatter matrix is calculated as

$$\Sigma_X = (X - \mu_X)^T (X - \mu_X) \tag{4}$$

where μ_X is the mean of all samples. The eigenvalues λs and corresponding eigenvectors ϑs of Σ_X are computed by eigenvalue decomposition, where

$$\Sigma_X \vartheta = \lambda \vartheta \tag{5}$$

The N eigenvectors are normalized and sorted in terms of corresponding eigenvalues from large to small

$$\vartheta_1, \vartheta_2, \dots, \vartheta_N \; with \; \lambda_1 > \lambda_2 > \dots > \lambda_N \tag{6}$$

The first k eigenvectors with the largest eigenvalues are chosen to construct the $d \times k$ dimension eigenvector matrix W

$$W = [\vartheta_1, \vartheta_2, \dots \vartheta_k] \tag{7}$$

Finally, both training data and test data are transformed onto new subspace by multiplying W

$$y = x \times W \tag{8}$$

Since the time complexity of eigenvalue decomposition of a DxD matrix is $O(D^3)$, the computing load becomes very high when the samples dimension D is large. A compact trick[10] could significantly reduce the complexity. The basic idea is shown below

$$(X - \mu_X)(X - \mu_X)^T \vartheta = \lambda \times \vartheta \tag{9}$$

$$(X - \mu_X)^T (X - \mu_X)(X - \mu_X)^T \vartheta = \lambda \times (X - \mu_X)^T \vartheta \tag{10}$$

We can see $(X - \mu_X)^T \vartheta$ is the corresponding eigenvector of scatter matrix. When $N \ll D$, we only need to perform the eigenvalue decomposition of a $N \times N$ matrix $(X - \mu_X)(X - \mu_X)^T$ instead of a $D \times D$ scatter matrix, which could reduce the computation load significantly. Note all the eigenvectors should be normalized before transformation.

C. Support Vector Machine

Support vector machine (SVM) is a discriminative supervised machine learning method which is popular in solving classification and regression problems[11]. An important property of support vector machine is that the determination of the model parameters corresponds to a convex optimization problem, so any local solution is also a global optimum[12]. The data is divided into two (or more) classes by defining a hyperplane that maximally separates the classes.

For two-class problem where the class label $s_n \in \{-1, 1\}$, the class of each input point x_n is determined by the form

$$y_n = w^T x_n + \mathbf{b} \tag{11}$$

A hyperplane satisfying $w^T x_n + \mathbf{b} = \mathbf{0}$ is used to separate the two classes. If $y_n > 0$, estimated class label $\hat{s_n}$ = 1; if $y_n < 0$, then $\hat{s_n}$ = -1.

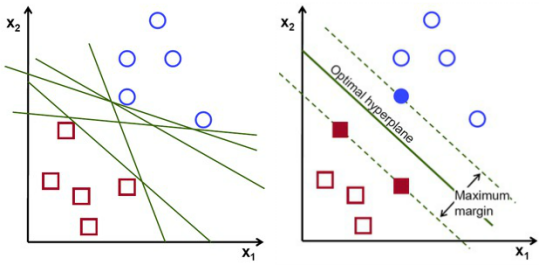

Fig. 2: A linear separable example of two-dimensional hyperplane between two classes

If the training data are linearly separable, we can select two parallel hyperplanes that separate the two classes of data and try to maximize the distance between them, these two hyperplanes can be described by the equations $w^T x_n + \mathbf{b} = 1$ and $w^T x_n + \mathbf{b} = -1$. The region bounded by these two hyperplanes is called the margin. The distance between these two hyperplane is $\frac{2}{\|\vec{w}\|^2}$, so to maximize the distance is equal

to minimize $\|w\|$. Since the data points should be correctly separated out of the margin, for each input point we must have

$$\hat{s_n}(w^T x_s + \mathbf{b}) \geq 1, \quad n = 1, 2, 3 \ldots N \tag{12}$$

Putting together, we get the optimization problem[12]

$$argmin \, \|w\|^2 \tag{13}$$

$$subject \; to \; \hat{s_n}(w^T x_s + \mathbf{b}) \geq 1, \quad n = 1, 2 \ldots N \tag{14}$$

Since the data are usually not linearly separable, the $hinge\ loss$ function[13] is introduced for soft-margin SVM.

$$L(x_n) = max(0, 1 - \hat{s_n}(w^T x_s + \mathbf{b})) \tag{15}$$

For input points lying on the correct side, the correspond hinge loss is 0, while for misclassified points, the functions value is proportional to the distance from the margin.Then we wish to minimize

$$C \times [\frac{1}{N} \sum_{n=1}^{N} max(0, 1 - \hat{s_n}(w^T x_s + \mathbf{b}))] + \|w\|^2 \tag{16}$$

where parameter C determines the trade-off between increasing the margin-size and ensuring that x_n is correctly classified.

For non-linear classification problems, the kernel trick[14] is introduced to make data separable in a feature space. A similarity function k is calculated between each prediction of unlabeled input x_i and labeled input x_j. We implemented four kernel functions in following table.

Linear Kernel	Polynomial Kernel
$k = x_i^T x_j$	$k = (\gamma x_i^T x_j + a)^d$
Sigmoid Kernel	RBF Kernel
$k = tanh(\gamma x_i^T x_j + a)$	$k = exp(-\gamma \lvert x_i - x_j \rvert)^2$

For regression problem, support vector regression aims to minimize the ε - sensitive error function

$$\varepsilon_\epsilon(y_n - r_n) = \begin{cases} 0, & if |y_n - r_n| < \epsilon \\ |y_n - r_n| - \epsilon, & otherwise \end{cases} \tag{17}$$

where r_n is the true source range at sample n and ϵ defines a region on either side of the hyperplane.

Since the source localization is a multi-class classification problem. A multi-class SVM is created by training one-vs-one models on all possible pairs of classes [12]. We use the SVC and SVR functions provided by scikit-learn to implement SVM. And the LinearSVC function is also applied as a comparison (see Section III-D).

D. Random Forest

Random forest is an ensemble machine learning method for classification and regression. A random forest classifier consists of multiple decision trees trained by different subset of the same training data. Though each single decision tree overfits its training subset, RF is robust by averaging over multiple decision trees. The class with most votes over all decision trees will be chosen as the prediction.

In training step, each decision tree is randomly assigned a subset of training data. For each node of the tree, the input data is split according to a cutoff value c along the i_{th} dimension.

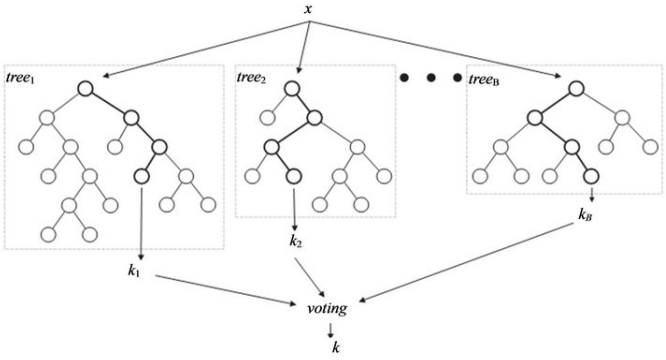

Fig. 3: Random Forest [15]

$$\begin{aligned} x_n \in x_{left} \;\; if \; x_{ni} > c \\ x_n \in x_{right} \;\; if \; x_{ni} \leq c \end{aligned} \tag{18}$$

x_{left} and x_{right} are the left and right regions. The optimal cutoff values are determined by minimize the cost function G.

$$c^* = \underset{c}{argmin} \;\; G(c) \tag{19}$$

$$G(c) = \frac{n_{left}}{N} H(x_{left}) + \frac{n_{right}}{N} H(x_{right}) \tag{20}$$

where n_{left} and n_{right} are the numbers of points in the regions x_{left} and x_{right}. $H(\cdot)$ is an impurity function. We apply $Gini\ impurity$ [16] as the impurity function.

$$H(x_m) = \sum_{k=1}^{K} f_{r_k}(1 - f_{r_k}) \tag{21}$$

$$f_{r_k} = \frac{1}{n_m} \sum_{x_n \in x_m} I(t_n, r_k) \tag{22}$$

where $r_k, k = 1, 2, \ldots K$ are the source range classes and t_n is the label of point x_n in region m, f_{r_k} is the fraction of points labeled with class r_k in region m. The region m is labeled by class l_m which represent the most common class in the region.

$$l_m = \underset{r_k}{argmax} \sum_{x_n \in x_m} I(t_n, r_k) \tag{23}$$

$$I(t_n, r_k) = \begin{cases} 1, & if \;\; t_n = r_k \\ 0, & otherwise \end{cases} \tag{24}$$

The remaining regions are partitioned iteratively until the tree reaches the maximum depth.

For regression problem, "the estimated class for each region is defined as the mean of the rue class for all points in the region, and the mean squared error is used as the impurity function"[4].

$$l_m = \frac{1}{n_m} \sum_{x_n \in x_m} r_n \tag{25}$$

$$H(x_m) = \sum_{x_n \in x_m} (l_m - r_n)^2 \tag{26}$$

where r_n is the source range at sample n.

Even though RF is robust to overfitting with sufficient trees and max depth, it takes a long time to train the model for high dimension data. PCA could effectively reduce the training time, as well as retain most of the precision.

E. Feedforward Neural Network

Feedforward neural network (FNN) is an artificial neural network which can be described a series of functional transformations [17], wherein the information moves only forward from input through the hidden layer to the output, as figure 4

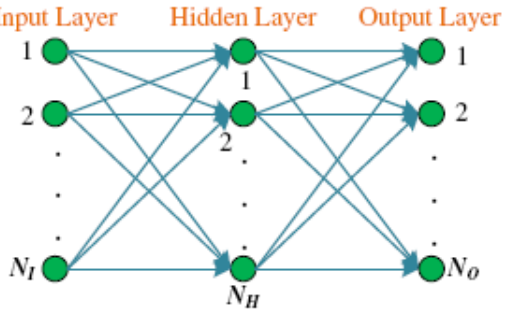

Fig. 4: One-hidden-layer feedforward neural network [17]

The input layer consists of D neurons where D is the dimension of the feature of each sample. For node j in hidden layer, the linear combination of the input variables is computed as activation a_j

$$a_j = \sum_{i=1}^{D} w_{ji}^{(L)} x_i + w_{j0}^{(L)}, \quad j = 1, 2, 3 \dots M \tag{27}$$

where M is the number of neurons in hidden layer. The parameters w_{ji} and w_{j0} are called the weights and biases. Then a_j is transformed using an activation function.

$$z_j = f(a_j) \tag{28}$$

we choose the sigmoid function as the activation function for hidden layer

$$f(a) = \sigma(a) = \frac{1}{1 + e^{-a}} \tag{29}$$

For node k in output layer, the linear combinations of z_j is computed as

$$a_k = \sum_{j=1}^{M} w_{kj}^{(2)} z_j + w_{k0}^{(2)}, \ k = 1, 2, \dots K \tag{30}$$

Note K is the number of range bins. And the $softmax$ function[18] is used as the activation function for output layer, which could constrain the output class $y_k(x, w)$ to be probability that the source is at range r_k:

$$y_k(x, w) = \frac{exp(a_k(x, w))}{\sum_{j=1}^{K} exp(a_j(x, w))}, k = 1, 2, 3 \dots K \tag{31}$$

where w is the set of all weight and bias parameters.

During training, the averaged cross entropy is chosen as the cost function

$$E(w) = -\frac{1}{N} \sum_{n=1}^{N} \sum_{k=1}^{K} t_{nk} \ln y_{nk} \tag{32}$$

and resulting weights and biases are to minimize the cost function

$$\hat{w} = \underset{w}{argmin} \ [-\frac{1}{N} \sum_{n=1}^{N} \sum_{k=1}^{K} t_{nk} \ln y_{nk}] \tag{33}$$

For regression problem, the output layer consists of only one neuron which represents continuous range variable. And the sum-of-squares error function[12] is chosen as cost function

$$E(w) = \frac{1}{2} \sum_{n=1}^{N} |y(x_n, w) - r_n|^2 \tag{34}$$

The Adaptive Moment estimation method[19] is used for optimization. The FNN is trained by specific iterations with a step length. For high dimension training data without enough samples, FNN will overfit the training data and has low precision. PCA is performed before training to avoid overfitting.

III. EXPERIMENT AND RESULT ANALYSIS

We use 2 of the given 5 datasets as the inputs of our system: $Dataset01$ and $Dataset03$. Each dataset contains two raw time series of pressure (Pa) acoustic data files received by vertical linear array (VLA) consisting of 15 hydrophones, one for training and another for test. The given corresponding true ranges and times recorded by GPS were used to create data labels by linear interpolation.

In $Dataset01$, the data from period $J255$ 19:06:05-19:21:05 are used as training set and $J256$ 09:32:10-09:48:10 are used as test set. In $Dataset03$, the data from period $J257$ 09:37:05-09:48:30 are used as training set and $J258$ 05:45:20-05:56:05 are used as test set.

After preprocessing, the 15×15 SCMs of every selected frequency are averaged over chosen snapshots with 1-sec succession. For single frequency SCM, the input vector is of length (feature dimension) $240(15 \times 16)$. For multi-frequencies SCMs, the input vector is of length $240 \times N_f$, where N_f is the number of selected frequencies. For example, if we select 30 frequencies to compute the input SCM, the dimension D of each sample vector is $240 \times 30 = 7200$.

The training samples are picked every second, while the test samples are picked every 5 seconds. For $Dataset01$, the training set has $(895 - snapshot)$ samples, and test set has $(955 - snapshot)/5$ samples; for $Dataset03$, the training set has $(655 - snapshot)$ samples, and test set has $(645 - snapshot)/5$ samples. Using $20m$ as the range interval for classification, in $Dataset01$, the source-receiver range $0 - 2960m$ are divided into $K = 149$ discrete range points; in $Dataset03$, the source-receiver ranges $850 - 3080m$ are divided into $K = 112$ discrete range points.

The mean absolute percentage error (MAPE) over N samples are used to quantitatively evaluate the results, which is defined as

$$E_{MAPE} = \frac{100}{N}\sum_{i=1}^{N}|\frac{R_{p_i} - R_{g_i}}{R_{g_i}}| \tag{35}$$

where R_{p_i} and R_{g_i} are the predicted range and the ground truth range respectively.

By using different parameters at each stage of the system, the performance of FNN, SVM, RF are discussed in following sections. Note there are many parameters to be tuned, limited by pages we cannot list all combinations, so when we focus on one parameter, the rest parameters are chosen as the same value which is relatively good by comparing the overall results.

A. Frequency Selection

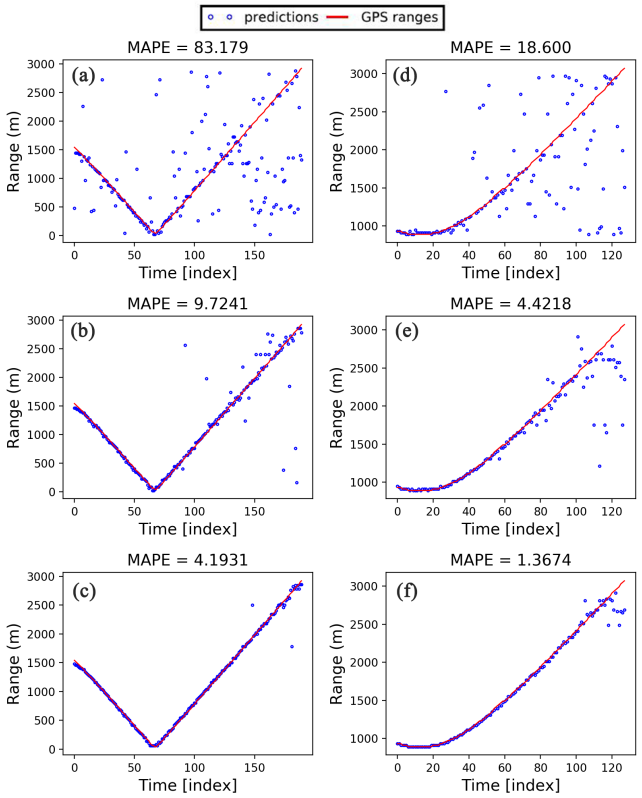

Fig. 5: Range Predictions on Dataset01 (a, b, c) and Dataset03 (d, e, f) by FNN. (a)(d) 515Hz, (b)(e) 300-700 Hz with 100 Hz increment, i.e. 5 frequencies. (c)(f) 300-700Hz with 13 Hz increment, i.e. 30 frequencies.

Figure 5 shows the FNN prediction results based on different frequency selections. The PCA dimension is set 100, and the hidden layer contains 1000 neurons. We can see for both $Dataset01$ and $Dataset03$, predictions based on 30 frequencies have the lowest MAPE. This is because multi-frequency features are more robust to noise with more uncorrelated features.

B. Snapshots

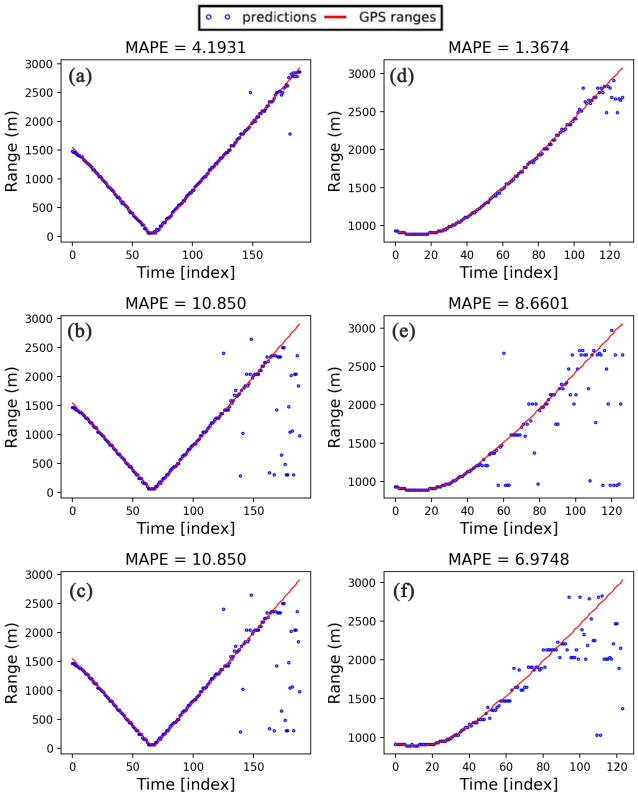

Fig. 6: Range Predictions on Dataset01 (a, b, c) and Dataset03 (d, e, f) by FNN. (a)(d) snapshot = 1, (b)(e) snapshot = 5, (c)(f) snapshot = 20.

In preprocessing, SCMs are averaged by snapshots to reduce the noise. But since all the variables are averaged, some features such as periodical variables will also be impaired, which results in higher ratio of misclassification. Since we have applied PCA to filter out the noise, there is no need to do a snapshot average. From figure 6 we can see that FNN has the best performance when snapshot=1. To compare the effects of different snapshots, the input data is of 30 frequencies and the dimension after PCA is set 100, while the hidden layer contains 1000 neurons.

C. Support Vector Machine

We applied four different kernels for SVM including linear, sigmoid, radial basis function and polynomial. For polynomial and sigmoid kernel, the coefficient a are set to 0, and the degree of polynomial kernel is set to 1. So the only difference between polynomial kernel and linear kernel is the coefficient gamma, which perform just a scaling.

For classification problem, figure 7 shows the results using four different kernels on Dataset01 with $C = 1$, $gamma = 1/149$.

We can see the linear kernel works well, but other 3 kernels have a continuous misclassification part. Recall that the penalty parameter C controls the trade-off between increasing the margin-size and ensuring that sample points

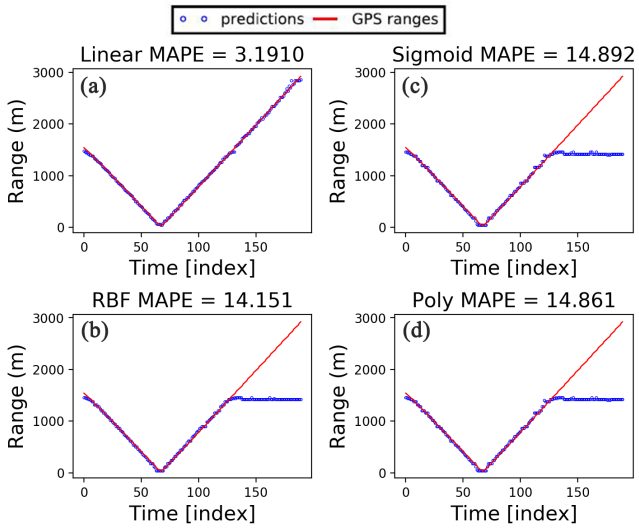

Fig. 7: SVM Classification on Dataset01, C=1, gamma=1/149

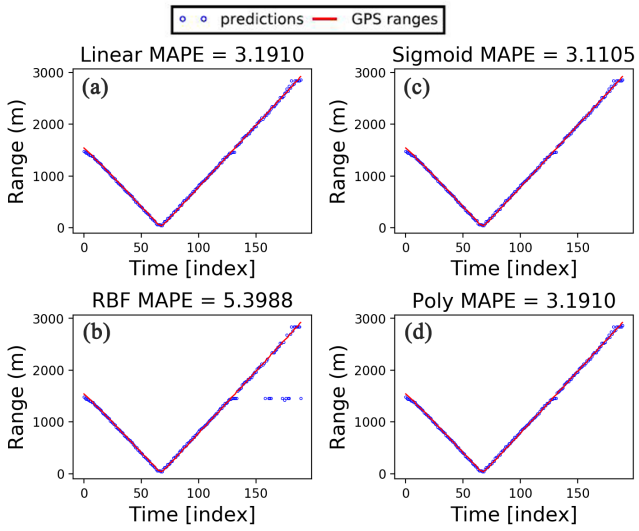

Fig. 8: SVM Classification on Dataset01, C=1000, gamma=1/149

are correctly classified, a large C means a soft boundary while a small C means a hard boundary. For polynomial kernel with $degree = 1$ and $a = 1$, its indeed a linear kernel whose penalty parameter C is multiplied by $gamma$. Once we increase the value of C to 1000, we can see all 4 kernels have outstanding performance since the minimum MAPE is 3.19 for $Dataset01$, see figure 8. While the minimum MAPE for $Dataset03$ is 0.71.

By comparing figure 7 and figure 8, we can see when $C = 1$, we give too much attention on maximizing the margin so there is a continuous points lies on the wrong sides. The effect of C can be also observed by setting gamma to 1 for polynomial kernel (See Fig. 10).

For regression problem, figure 9 shows the results on $Dataset01$ with $C = 1000$, $gamma = 1/149$. We can

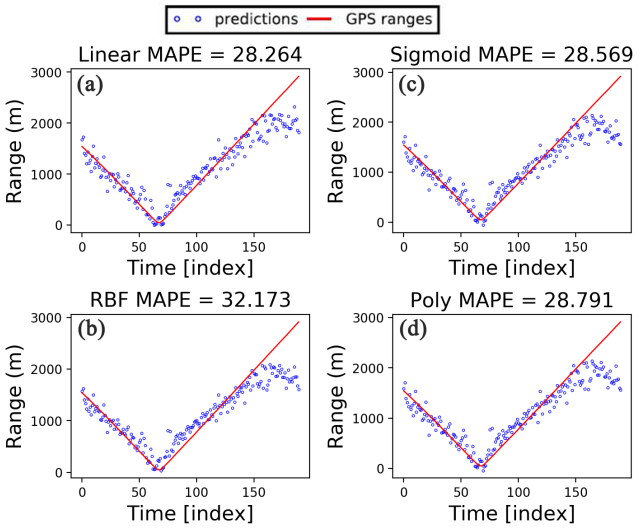

Fig. 9: SVM Regression on Dataset01, C = 1000, gamma = 1/149

see the SVM regression does not have as good performance as classification since the minimum MAPE is 28.2 for $Dataset01$ and 11.2 for $Dataset03$.

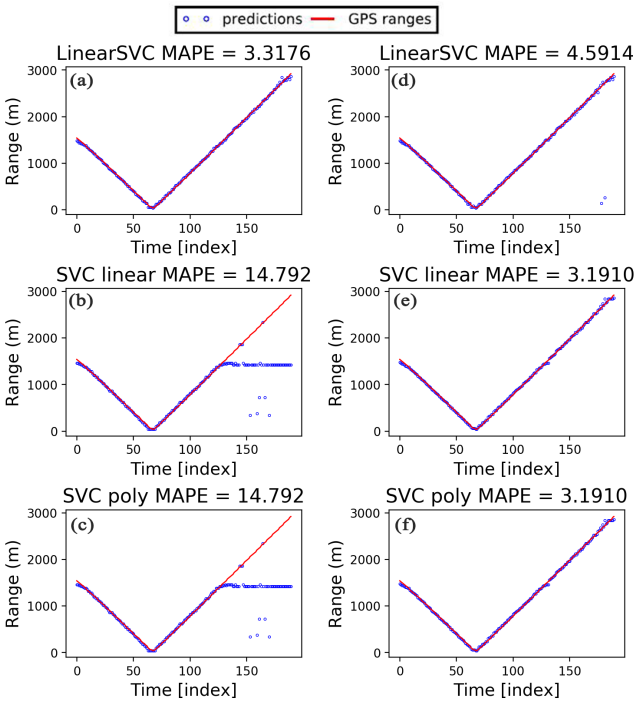

Fig. 10: Classification using SVC and LinearSVC on Dataset01. (a)(b)(c) C = 1, gamma = 1, (d)(e)(f), C=0.01, gamma=1

D. LinearSVC vs. SVC

Besides SVC, there is another function provided by $scikit - learn$ named $LinearSVC$. Though named as SVC, this function is actually not a conventional SVM

classifier. The default loss function of LinearSVC is the squared hinge loss, and it will penalize the intercept b. So instead of minimizing $\|w\|^2$, LinearSVC aims at minimizing $\|w, b\|^2$. Moreover, for multi-class problem, LinearSVC performs one-vs-rest while SVC performs one-vs-one.[20] These changes result in the different performance of Linear SVC and SVC.

Figure 10 shows the classification results on $Dataset01$ using LinearSVC and SVC with linear and polynomial kernel (degree=1). The gamma is set to 1 so the polynomial kernel is actually linear kernel. We can see when $C = 1$, the SVC has better performance than LinearSVC, but when $C = 0.01$, LinearSVC has lower MAPE.

E. Random Forest

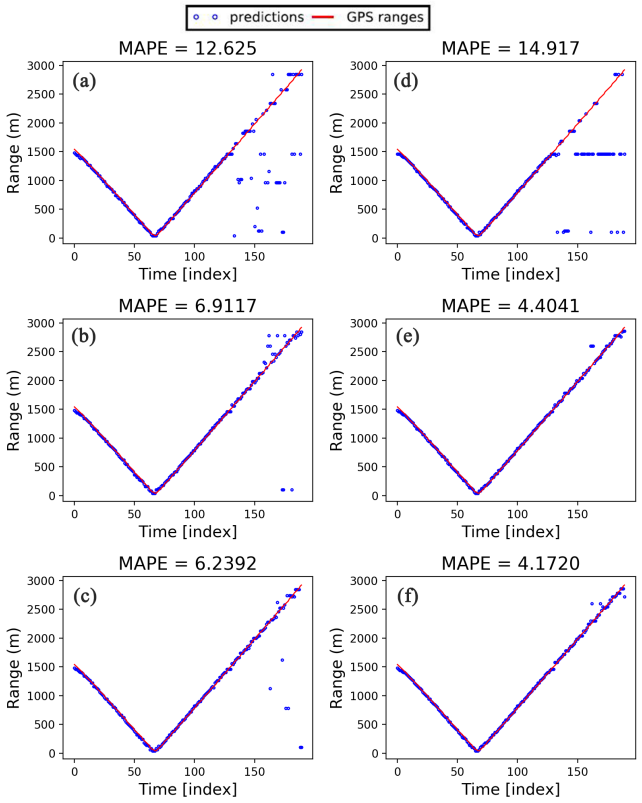

Fig. 11: RF Classification Results on Dataset01. (a)(b)(c) tree number = 50, (d)(e)(f) tree number = 500. (a)(d) max depth = 5, (b)(e) max depth = 20, (c)(f) max depth = 100. PCA = 100.

For classification problem, figure 11 shows the prediction results on $Dataset01$ using different tree number and depth. We can see when tree number = 50, the MAPE of the prediction is large, this is because RF needs sufficient amount of trees to avoid overfitting, when the tree number is low, the prediction variance increases. Also we can see the decision tree needs sufficient depth to fit the data. With PCA = 100, RF with 500 trees and 100 depth has the best performance. The minimum MAPE obtained is 4.17 for $Dataset01$ and 0.82 for $Dataset03$.

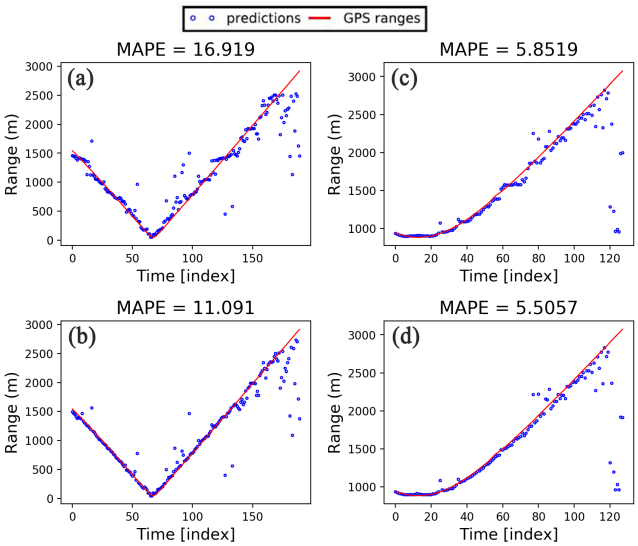

Fig. 12: RF Regression Results on Dataset01 (a)(b) and Dataset03 (c)(d). (a)(c) tree number = 50, (b)(d) tree number = 500. Max depth = 100 and PCA = 100.

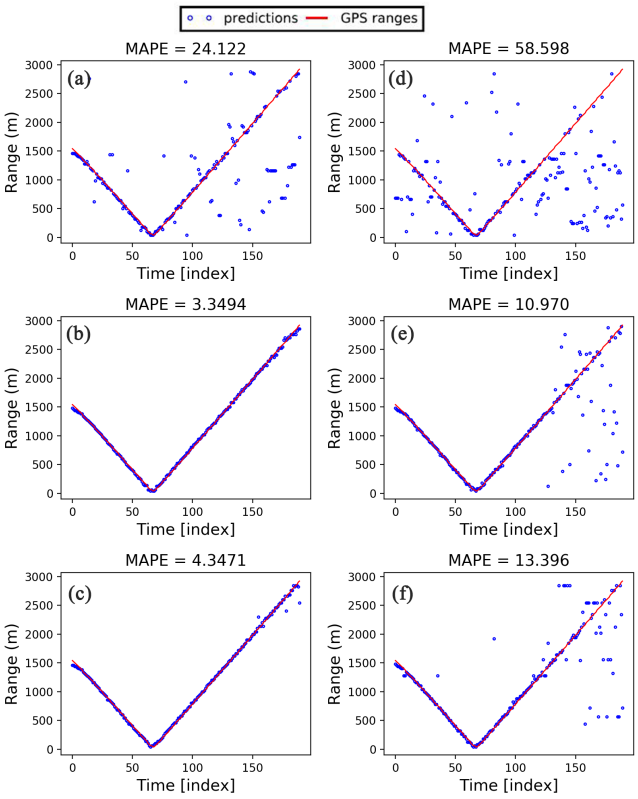

Fig. 13: FNN Classification Results on Dataset01. (a)(b)(c) PCA = 100, (d)(e)(f) PCA = 500, (a)(d) hidden neurons = 5, (b)(e) hidden neurons = 100, (c)(f) hidden neurons = 1000.

Figure 12 shows the results of RF regression on $Dataset01$ with different tree numbers. We can see again RF needs sufficient number of trees to ensure precision. Comparing figure 9 and figure 12 we can see RF has better

performance in regression problem than SVM, with PCA = 100. The minimum MAPE of $Dataset01$ is 11.09, while the minimum MAPE of $Dataset03$ is 5.5.

F. Feed-forward neural Network

For classification problem, figure 13 shows the results of FNN on $Dataset01$ with different numbers of hidden neurons and PCA dimensions. From figure 13(a)(d), we can see that the FNN has relatively poor performance with limited hidden neurons. However, from figure 13(b)(c), and (e)(f), it is clearly that too many hidden neurons will also reduce the performance because of overfitting. After tuning, we found 100 hidden neurons is a suitable choice.We can also see that FNN with PCA=100 has better performance than FNN with PCA = 500. This is because PCA=100 actually filters out more redundant features than PCA=500 from the original data, which avoids overfitting more effectively. The minimum MAPE we obtained is 3.35 for $Dataset01$, and 1.37 for $Dataset03$.

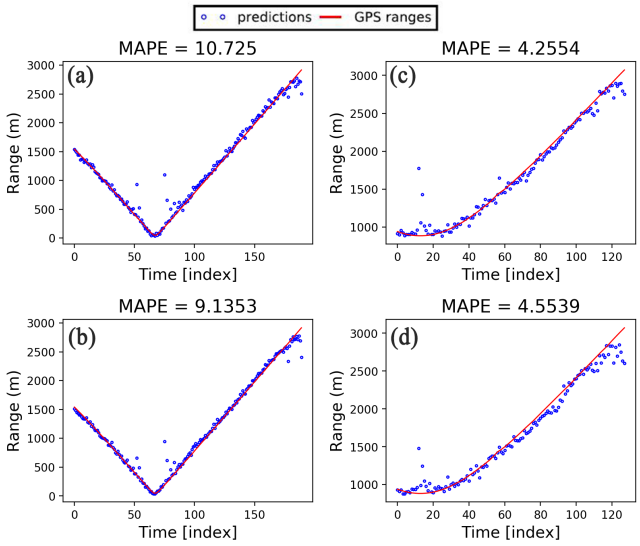

Fig. 14: FNN Regression Results on Dataset01 (a)(b) and Dataset03 (c)(d). (a)(c) Hidden neurons = 100, (b)(d) Hidden neurons = 1000. PCA = 20.

Figure 14 shows the regression results on $Dataset01$ and $Dataset03$ with different hidden neurons, After tuning, the dimension of PCA is set 20. Comparing figure 9, figure 12 and figure 14, we can see FNN has relatively the best performance for regression problem, since the minimum MAPE obtained is 9.14 for $Dataset01$ and 4.26 for $Dataset03$.

IV. CONCLUSION

Considering the experiment results, for classification problem, SVM has the overall best performance with the minimum MAPE 3.19 for $Dataset01$ and 0.71 for $Dataset03$. For regression problem, FNN has the overall best performance with the minimum MAPE 9.14 for $Dataset01$ and 4.26 for $Dataset03$. PCA effectively avoids the overfitting problem of FNN, and retains most of RF's precision while reduces the training time significantly. In acoustic data analysis, PCA is, to some degree, like an adaptive filter which could efficiently remove the redundant variables and noise, and preserve only important features for source localization.

REFERENCES

[1] Roee Diamant, Hwee-Pink Tan, and Lutz Lampe. Los and nlos classification for underwater acoustic localization. *IEEE Transactions on mobile Computing*, 13(2):311–323, 2014.

[2] Christopher MA Verlinden, Jit Sarkar, WS Hodgkiss, WA Kuperman, and KG Sabra. Passive acoustic source localization using sources of opportunity. *The Journal of the Acoustical Society of America*, 138(1):EL54–EL59, 2015.

[3] Alexandra Tolstoy. *Matched field processing for underwater acoustics*. World Scientific, 1993.

[4] Haiqiang Niu, Peter Gerstoft, and Emma Reeves. Source localization in an ocean waveguide using supervised machine learning. *arXiv preprint arXiv:1701.08431*, 2017.

[5] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in python. *Journal of Machine Learning Research*, 12(Oct):2825–2830, 2011.

[6] Harold Hotelling. Analysis of a complex of statistical variables into principal components. *Journal of educational psychology*, 24(6):417, 1933.

[7] Yacine Ait-Sahalia and Dacheng Xiu. Using principal component analysis to estimate a high dimensional factor model with high-frequency data. 2016.

[8] Shabnam N Kadir, Dan FM Goodman, and Kenneth D Harris. High-dimensional cluster analysis with the masked em algorithm. *Neural computation*, 2014.

[9] Karl Pearson. Liii. on lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 2(11):559–572, 1901.

[10] Matthew Turk and Alex Pentland. Eigenfaces for recognition. *Journal of cognitive neuroscience*, 3(1):71–86, 1991.

[11] Zhuowen Tu. Learning generative models via discriminative approaches. In *Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on*, pages 1–8. IEEE, 2007.

[12] C Bishop. Pattern recognition and machine learning (information science and statistics), 1st edn. 2006. corr. 2nd printing edn. *Springer, New York*, 2007.

[13] Urün Dogan, Tobias Glasmachers, and Christian Igel. A unified view on multi-class support vector classification. *Journal of Machine Learning Research*, 17(45):1–32, 2016.

[14] Bernhard Schölkopf. The kernel trick for distances. In *Advances in neural information processing systems*, pages 301–307, 2001.

[15] Cuong Nguyen, Yong Wang, and Ha Nam Nguyen. Random forest classifier combined with feature selection for breast cancer diagnosis and prognostic. 2013.

[16] Leo Breiman. Technical note: Some properties of splitting criteria. *Machine Learning*, 24(1):41–47, 1996.

[17] Yudong Zhang, Shuihua Wang, Genlin Ji, and Preetha Phillips. Fruit classification using computer vision and feedforward neural network. *Journal of Food Engineering*, 143:167–177, 2014.

[18] Kaibo Duan, S Sathiya Keerthi, Wei Chu, Shirish Krishnaj Shevade, and Aun Neow Poo. Multi-category classification by soft-max combination of binary classifiers. In *International Workshop on Multiple Classifier Systems*, pages 125–134. Springer, 2003.

[19] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[20] Wojciech Marian Czarnecki. *Under what parameters are SVC and LinearSVC in scikit-learn equivalent?*, 2015. Available at `https://goo.gl/PnkHBe`.