Reimplementation of Source Localization in an Ocean Waveguide Using Supervised Machine Learning

Jinzhao Feng
jif078@ucsd.edu

Zhuoxi Zeng
z4zeng@ucsd.edu

Yu Zhang
yuz083@ucsd.edu

***Abstract*—Supervised machine learning can be effectively used to solve source localization problem. Machine learning methods, such as feed-forward neural networks(FNN), support vector machine(SVM) and random forests(RF), work on the data directly. The Inputs are normalized sample covariance matrices(SCMs), and classifiers will be trained on that. Testing results will be used to compare and demonstrate the advantages of each method for under water source localization.The performance of each method is evaluated by mean absolute percentage error (MAPE), and SVM gives the best result with an average of 3.7% in all datasets.**

I. Introduction

Source localization problem in ocean acoustic can be solved by observed data in different ways. According to the result from matched-field-processing(MFP), a classical method, there are some obvious errors due to sidelobes [1]. The reason for this is that MFP can produce accurate prediction only if the ocean environment is modeled accurately. Intuitively, due to its complicated and unstable, other approaches may give better results.

Machine Learning methods used in this problem work on observed data directly and then predict source ranges. In this paper, three popular machine learning algorithms, feed-forward neural network(FNN), support vector machine(SVM), and random forest(RF), are explored to address the range estimation. For FNN, the deep learning framework we use is tensorflow, while SVM, RF and feature selection are implemented based on Scikit-learn. These well-developed machine learning libraries simplify the implementation task and save computational time a lot for us.

The input data are normalized sample covariance matrices(SCMs), where amplitude and phase information are included. Specially, for different machine learning method, the data will be preprocessed slightly different, and this will be talked later in this paper. The output is the prediction on source range, or location in the ocean based on models that have been trained on training datasets.

II. Dataset and Features

The shipping noise data was collected from the signal of R/V New Horizon. Figure1 shows the dataset geometrically, with the vertical linear array (VLA) indicated as a red triangle. The five ship tracks shown in five colors are used for estimation, and each dataset includes both training and test data as summarized in tableI with different time ranges [1].

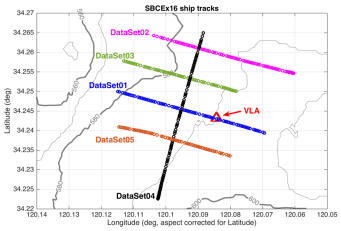

Fig. 1. The structure of SVM with slack variable

Dataset	Training samples	Test samples
01	890	189
02	830	177
03	650	127
04	1010	213
05	615	135

TABLE I
NUMBER OF TRAINING AND TEST SAMPLES OF FIVE DATASET

The received pressure phase and amplitude is preprocessed to a normalized sample covariance matrix (SCM). Since there are more than 7,000 features in each dataset, Principal Component Analysis (PCA) is used to reduce the dimension of feature space, which significantly helps decrease the computation time and improve the performance of each model. Principal components are chosen such that it will retain 90% of the original variance of data.

A. Sample Covariance Matrix

As discussed in [1], the received pressure array is transformed to a normalized sample covariance matrix (SCM) as input. The DFT of the sound pressure at frequency f at L sensors is denoted as

$$p(f) = [p_1(f), p_2(f), ..., p_L(f)]^T \tag{1}$$

and the sound pressure model is:

$$p(f) = S(f)g(f, r) + \eta \tag{2}$$

where $S(f)$ is the source term, g is the Green's function, and η is the noise.

In order to diminish the effect of source amplitude $|S(f)|$, the complex pressure is normalized as:

$$\tilde{p}(f) = \frac{|p(f)|}{\sqrt{\sum_{l=1}^{L} |p_l(f)|^2}} = \frac{p(f)}{||p(f)||_2} \tag{3}$$

Then the conjugate symmetric matrix of the normalized sample covariance matrices over N_S snapshots is:

$$C(f) = \frac{1}{N_s} \sum_{s=1}^{N_s} \tilde{p}_s(f) \tilde{p}_s^H(f) \tag{4}$$

Where H denotes the hermitian matrix, and $\tilde{p}_s$ denotes the sound pressure over the s^{th} snapshot.

As in classification problem, the source range is equally divided into K bins, $r_1, r_2, ..., r_K$ with same width δr, and hence provides labels $t_n \in r_k$ for each input vector x_n. Especially in FNN, the class label t_n is transformed into a $1 \times K$ binary vector $\mathbf{t}_n$ such that

$$\mathbf{t}_n k = \begin{cases} 1, & if |t_n - r_k| \leq \frac{\delta r}{2} \\ 0, & otherwise \end{cases} \tag{5}$$

where $\mathbf{t}_n = t_{n,1}, ..., t_{n,K}$.

These labels are later used in all methods when we build the training models.

B. Principal Component Analysis

Principal Component Analysis was first introduced by Karl Pearson in 1901 [1] and is one of the oldest techniques to understand which dimensions of a high dimensional dataset are "important" by projecting data into a lower dimension space. PCA takes a high dimensional input matrix and select important components and hence compress the data by ignoring features which are useless.

The main idea of PCA is using eigenvalue decomposition to select features [3]. Given an input matrix $X \in \mathbb{R}^{M \times N}$, where N is the number of samples, and M is the number of features. Each column of X, x_i is one data point with M features. First, sample mean is needed:

$$\mu = \frac{1}{N} \sum_i x_i \tag{6}$$

Then, sample covariance,

$$\Sigma_x = \frac{1}{N} \sum_i (x_i - \mu)(x_i - \mu)^T \tag{7}$$

Next, we use eigenvalue decomposition to find the eigenvalues and eigenvectors of the covariance.

$$\Sigma_x = \Phi \Lambda \Phi^T \tag{8}$$

where

$$\Lambda = diag(\lambda_1, \lambda_2, ..., \lambda_M) \tag{9}$$

and

$$\Phi \Phi^T = I \tag{10}$$

λ_i and ϕ_i are placed in a descending order. We want to keep the dimensions with the highest variance and discard the dimensions with the lowest variance. In some sense to maximize the amount of "randomness" that get preserved when we compress the data.

The first k largest eigenvalues are selected with their corresponding eigenvectors, the selected eigenvectors ϕ_i are called principal components:

$$\hat{\phi} \in \mathbb{R}^{M \times k} = (\phi_1, \phi_2, ..., \phi_k) \tag{11}$$

Then, the data is compressed and is reconstructed as:

$$y = \hat{\phi}^T (X - \mu) \in \mathbb{R}^{k \times N} \tag{12}$$

We can adjust the parameter k to control the size of the data and the computation time. In order to keep the consistency, the same preprocessed data, SCMs, which retains 90% of the original variance of data is used as input for all three methods.

III. Methods

A. Feed-forward Neural Network

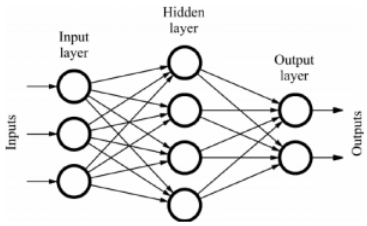

Fig. 2. Illustration of FNN

Feed-forward Neural Network is a kind of deep learning method where data flows in one direction and all neurons are fully connected. A Feed-forward Neural Network may have many layers hence the deep learning aspect, where the output of each layer can be described by the equation:

$$a_j = f(\sum_{i=i}^{D} w_{ij} x_i + b_j) where j = 1, 2, 3, 4...M$$

where f is an activation function which is non-linear function used to give gradient to the ooutput in order for the system to be trained by backpropagation. W_{ij} and b_j are the weights and bias of the hidden layer which can be thought of as a matrix. J is the number of hidden units within this layer and i is i-th sample of the input into the layer.

For this project, we will be using the Rectified Linear Unit, otherwise known as ReLU which has the following characteristics:

$$f(x) = max(0, x)$$

This has the advantage over the traditional sigmoid activation function because there is no gradient vanishing unlike the sigmoid function which has parts where gradient is 0, which would lead to no updates to the weight and biase hence no learning. In this project, since we are classifying things into

distance ranges, this is in essence a classification problem, therefore we will be using a softmax as our final layer:

$$y_k(x,w) = \frac{exp(a_k(x,w))}{\sum_j (a_j(x,w))}$$

with cross entropy cost defined as:

$$E_n(t_n, y_n) = -\sum_k t_{nk} ln y_{nk}$$

With multiple layers of Feed forward network, we are hoping that it will capture the non-linearity natural of the data and give good predication for distance.

B. Support Vector Machine

In machine learning, support vector machines are supervised learning models with associated learning algorithms that analyze data used for classification and regression analysis [4]. For linear separable data points, SVM is going to find a

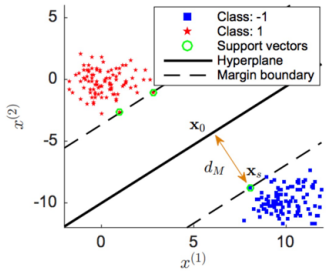

Fig. 3. The structure of SVM

hyperplane, defined as:

$$\omega^T x + b = 0 \tag{13}$$

that maximize the margin, define as:

$$\frac{|\omega^T x_n + b|}{||\omega||_2} \tag{14}$$

By applying normalization that $\min_i |\omega^T x_i + b| \equiv 1$, the problem becomes:

$$\min_{\omega,b} ||\omega||^2 \tag{15}$$

subject to

$$y_i(\omega^T x_i + b) \geq 1 \forall i \tag{16}$$

This convex problem can be solved by its dual problem, and the answer is:

$$\omega^* = \sum_{i \in SV} \alpha_i^* y_i x_i \tag{17}$$

$$b^* = -\frac{1}{2} \sum_{i \in SV} y_i \alpha_i^* (x_i^T x^+ + x_i^T x^-) \tag{18}$$

For prediction, the output will be:

$$f(x) = sgn[\sum_{i \in SV} y_i \alpha_i^* x_i^T x + b^*] \tag{19}$$

If the training set is not linear separable, the slack variable $\zeta \geq 0$is introduced to allow misclassification happen. Then the optimization problem is:

$$arg \min_{\omega,b} \frac{1}{2}||\omega||^2 + C \sum_{n=1}^{N} \zeta_n \tag{20}$$

subject to

$$s_n y_n \geq 1 - \zeta_n, n = 1, ..., N \tag{21}$$

The value of C controls the tradeoff between max margin and misclassification [4]. In addition to performing linear

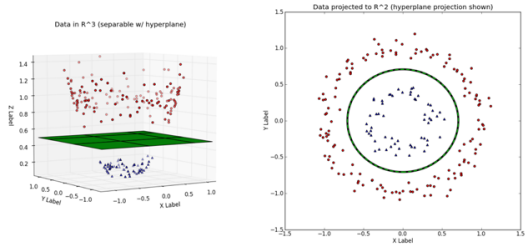

Fig. 4. The structure of SVM with slack variable

classification, SVMs can efficiently perform a non-linear classification using what is called the kernel trick, implicitly mapping their inputs into high-dimensional feature spaces. In our study, three types of kernel are used

1) Linear Kernel
2) RBF(Gaussian) Kernel
3) Polynomial kernel

C. Random Forest

RF consists of Classification and Regression Trees (CART), a conditional weighted method which only uses important features to do the classification [6]. Gini impurity is chosen as the metric to split the root in CART:

$$Gini(f) = \sum_{i=1}^{J} f_i(1 - f_i), i = 1, 2, ..., J \tag{22}$$

where J is the number of classes and f_i is the portion of data which belongs to class i [7]. The dataset keeps splitting based on Gini index until all the nodes is splitted to a maximum or when a specific threshold value is reached. However, a CART would cause over-fitting if the node in the tree is splitted too robustly [5].

Random Forest is a generalization of the decision tree model, which helps control this over-fitting problem. Assume the number of classes in the training set is N. Then sample of these N classes is taken at random but with replacement. If there are M input features, a number $m < M$ is specified such that at each node, m variables are selected at random out of M. The best split on these m is used to split the node. The value of m is held constant while the forest is building. Each tree grows to the largest extent possible, and there is no pruning. Aggregating the predictions of the n tree trees is used to predict new data point. The outcome is determined by the

majority votes for classification problem and the average for regression problem [6]. Figure5 shows the general idea of RF.

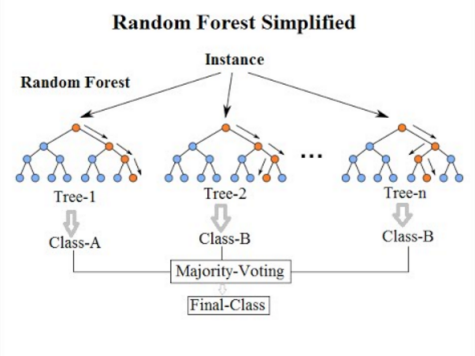

Fig. 5. The structure of SVM with slack variable

D. Error Metric

In order to compare three methods quantitatively, we use the mean absolute percentage error (MAPE) over N samples to evaluate the model:

$$MAPE = \frac{100}{N}\sum_{i=1}^{N}\left|\frac{R_{pi} - Rgi}{R_{pi}}\right| \tag{23}$$

where R_{pi} and R_{gi} are the predicted range and the ground truth range respectively. MAPE is chosen because it considers "the magnitude of error in faulty range estimates as well as the frequency of correct estimates" [1] [8].

IV. Results and Discussion

A. Result of FNN

Feed Forward Network is implemented using Tensorflow. For this project, we have found that a three layer network with 64, 128 and 256 hidden units performed reasonably well. Figure 6 shows the results of FNN on all datasets where the red line represent the truth value and the blue circles represent the predictions.

B. Result of SVM

When implementing SVM by using Scikit-learn(Python), there are two ways, Linear SVC and SVC. We tested both of them. Also, the corresponding parameters for different SVMs are:

1) SVC (linear kernel): C=0.1
2) SVC (polynomial kernel): C=10, degree=1
3) SVC (rbf kernel): C=10
4) Linear SVC: C=0.001

Fig7 results for Dataset01. Blue circle is prediction, and red line is ground truth. TableII shows the results on all datasets.

C. Result of RF

Figure8 shows the result of RF. From the figure, RF performs good on dataset01 and dataset04 while performs not so good on dataset03 and dataset 05. A possible conclusion can be driven that performance of RF is related to the symmetry of the dataset to a large extent. MAPE of RF is shown in tableIII.

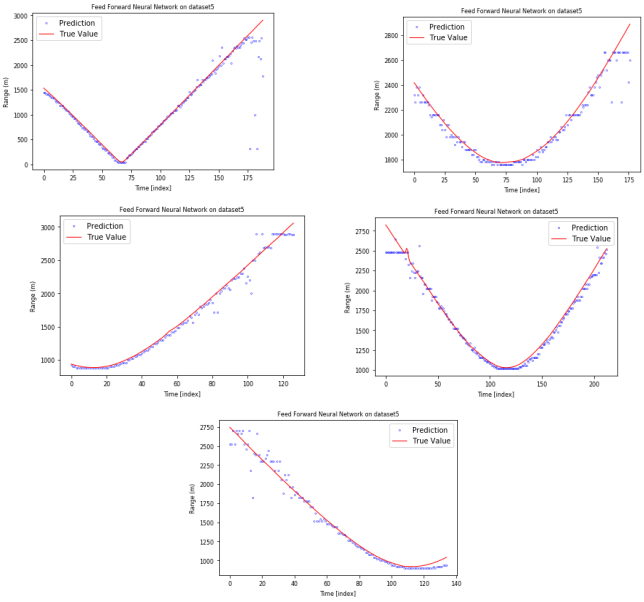

Fig. 6. Figure showing FNN results of all datasets

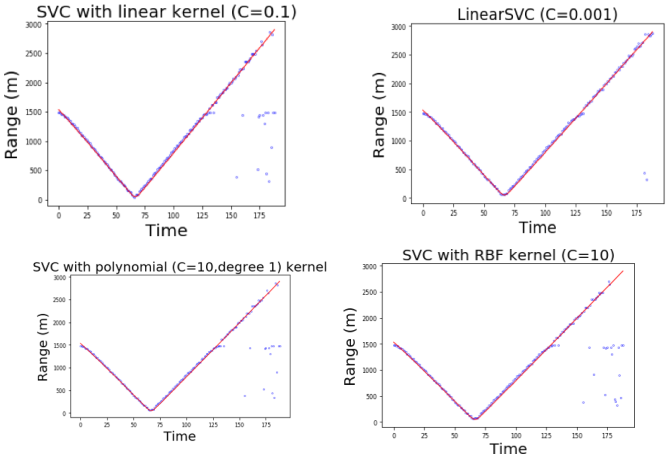

Fig. 7. SVM result with different kernel type

V. Conclusion

By comparing the methods, SVM outperforms the comparison with its smallest average MAPE as low as 3.7%. We obtained similar MAPE of the first two datasets as the reference paper did, which is as expected. FNN performs quite well, and SVM gives the nest results with sacrificed kernel type. RF gives a relatively high MAPE in dataset 03 and dataset 05, and the reason is maybe due to the symmetry

Dataset	Polynomial	RBF	linear kernel	LinearSVC
Dataset01	8.07%	9.43%	8.35%	4.99%
Dataset02	5.28%	5.10%	5.31%	3.75%
Dataset03	4.41%	9.27%	4.42%	2.64%
Dataset04	1.83%	1.76%	1.78%	2.33%
Dataset05	23.55%	2.42%	23.61%	5.40%

TABLE II
SVM Result of all dataset with different kernel types

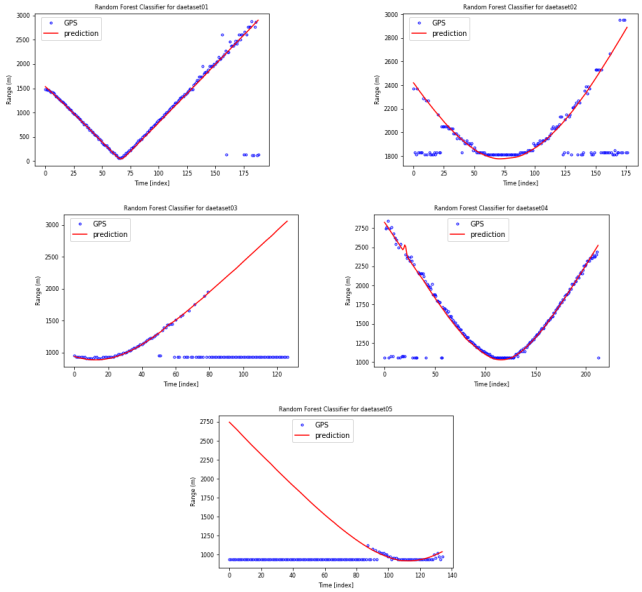

Fig. 8. RF result on all dataset

Dataset	FNN	SVM	RF
Dataset01	7.31%	4.99%(LinearSVM)	7.71%
Dataset02	1.91%	3.75%(LinaerSVM)	10.12%
Dataset03	2.66%	2.64%(LinaerSVM)	28.11%
Dataset04	2.58%	1.76%(RBF)	7.51%
Dataset05	3.32 %	2.42%(RBF)	31.04%

TABLE III
COMPARISON OF MODEL

distribution of the dataset.

In this paper, there are limited amount of data and five tracks are trained and tested as five independent datasets. In the future, we are going to combine all five dataset as one big training set to see how these three models work. In addition, we have started to experiment with using RNN to capture the temporal information of the data, in the future we would like to develop upon this a little more.

REFERENCES

[1] Niu, Haiqiang, Emma Reeves, and Peter Gerstoft. "Source localization in an ocean waveguide using supervised machine learning." The Journal of the Acoustical Society of America 142.3 (2017): 1176-1188.
[2] K. Peason. On lines and planes of closest fit to systems of point in space. Philosophical Magazine, 2(11):559572, 1901.
[3] Wold, Svante, Kim Esbensen, and Paul Geladi. "Principal component analysis." Chemometrics and intelligent laboratory systems 2.1-3 (1987): 37-52.
[4] C Bishop. Pattern recognition and machine learning (information science and statistics), 1st edn. 2006. corr. 2nd printing edn. Springer, New York, 2007.
[5] M. P. Vayssieres, R. E. Plant, and B. H. Allen-Diaz. Classification trees: An alternative non-parametric approach for predicting species distributions. Journal of vegetation science, 11(5):679694, 2000.
[6] L. Breiman. Random forests. Machine learning, 45(1):532,2001.
[7] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone. Classification and regression trees. wadsworth & brooks. Monterey, CA, 1984.
[8] P. Goodwin and R. Lawton, On the asymmetry of the symmetric MAPE, Int. J. Forecasting 15, 405408, (1999).