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

Yaqi Chen	Shao Hua	Yuhang Ming	Yang Zhang
A53204355	A53220045	A53209999	A53212762

1. Summary

1.1. Abstract

Machine learning methods are applied to ocean acoustic source localization in order to accurately learn source range in the complicated ocean environment. In this paper, three machine learning data-driven methods, support vector machines (SVM), feed-forward neural networks (FNN) and random forests (RF), are used for source localization. The FNN and the RF are focused in this paper. Principle Component Analysis (PCA) which reduces the dimension of features is applied to preprocess the shipping noise data collected from the signal of R/V New Horizon. Considered as a classification problem, the source range estimation problem has been solved. The results of three methods are compared as well. Besides comparing the performances among these three statistical methods, the robustness of our models against different levels of additional Gaussian noises has been also tested.

1.2. Conclusion

By comparing the three methods in the results section, FNN outperforms the comparison with its smallest average test errors. In addition, all the three statistical models have better performances on classifying data from the classes containing more than one training samples, which indicates the necessity of big data for machine learning. The reason is that more training samples from one class could represent the properties of that class better and this helps the models learn the underlying relationship between the input variables and the labels better as well, on the other hand, when the number of the training data is too small, the model would be affected by some outliers and it will tend to fit the noises instead of real underlying relationship. By testing the models with various levels of white Gaussian noise on our datasets, the FNN and RF algorithms performed well even when the SNR is up to -30dB. Therefore, these proposed models are robust to the noise so that they can be applied under some complicated and changing conditions.

2. Introduction

Source localization is significant to underwater surveillance, detecting and tracking. Since the underwater environment is unstable, finding a method that can be adapted to various water conditions is necessary. As traditional approaches, matched-field processing (MFP) methods have been successfully used for ocean acoustic localization[7][8][2]. The estimation methods requires maximum exploitation of signal and noise physical structure that can be paired to optimum methods for signal processing[2]. At the same time, these methods take extreme long time to match the source and environment parameters. In addition, MFP requires too many ideal assumptions to attain accurate localizations[1], therefore, it is difficult for MFP methods to localize source without good ocean environments[8].

In order to explore more complicated ocean environments, machine learning methods, such as support vector machines, feed-forward neural network and random forest, are put forward in this paper. Since SVM has been studies successfully, we focus on the FNN and RF model methods. The framework of these methods is shown in section 3. Before training and testing, Principle Component Analysis was applied for pre-processing the datasets. The theory of PCA has been demonstrated in section 3 as well.

We first built up an FNN structure with appropriate hidden layers and neurons in proposed FNN method. Then, we trained machine learning model for each dataset and the test data as input was used to verify the accuracy of our training. At the same time, cross-entropy was calculated to measure the error rate of the data. Five different Gaussian noises from -40dB to 10dB were added on each original dataset to test the stability of our FNN algorithm. The results are shown in section 4. For random forest model, PCA was applied to reduce the dimension of features. Gini impurity was chosen as the metric to split a root. The results were shown in section 4.

The shipping noise data was collected from the signal of R/V New Horizon. Figure 1 shows the experiment geometry, with the vertical linear array (VLA) indicated as a red

triangle. The five ship tracks, shown in five different color lines, were used for our estimation. The sampling rate of the hydrophone was 25 kHz.

Every dataset includes both training and test data with different time ranges (shown in Table 1). The source-receiver range is around 1.9 km and the label was created every 20m. The total training and test sample numbers of each dataset was shown in Table 2. The input data frequency range is from 311 Hz to 711 Hz. The performances of FNN, RF and SVM are shown in the results section. Result errors are calculated by mean absolute percentage error (MAPE)E_{MAPE} (the equation is discussed in section 3).

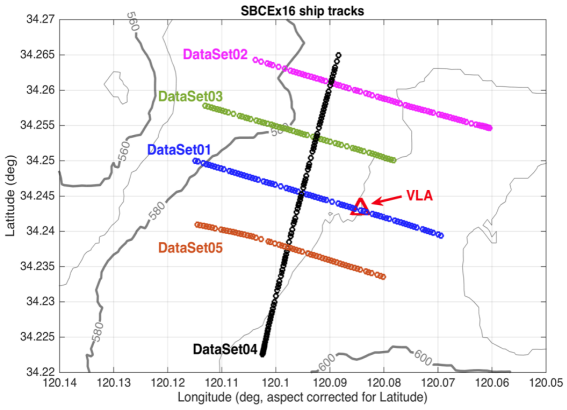


Figure 1: (Color online) Five ship tracks datasets

3. Framework

3.1. Principal Component Analysis

Principal Component Analysis is firstly brought out by Karl Pearson in 1901 [9]. It is a popular statistical tool for data processing and analysis, especially when the sample contains large number of measurements. Due to its advantage in reducing data size by projecting them into lower dimensions space, it's very prevalent in data mining as well as machine learning.

The idea of PCA is using eigenvalue decomposition or singular value decomposition to find out the most significant features and only reserving the projections of the data onto these feature vectors. Because the power of most signals is concentrated on several directions, the number of the selected feature vectors should be much less than the number of measurements. Therefore, the size of the data will be reduced.

Assume we have a $k \times N$ data matrix X, where N is the number of samples, and k is the number of measurements. Each column of X, x_i, is a data sample. Usually both N and k are rather large.

Firstly, we need to compute sample mean before we make the data centered, or zero-mean.

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

Then, we are going to compute sample covariance,

$$\hat{\Sigma} = \frac{1}{N}\sum_i (x_i - \hat{\mu})(x_i - \hat{\mu})^T \tag{2}$$

After that, we use eigenvalue decomposition to compute eigenvalues and eigenvectors of $\hat{(\Sigma)}$,

$$\hat{\Sigma} = \Phi\Lambda\Phi^T \tag{3}$$

$$\Lambda = diag(\lambda_1, \lambda_2, ..., \lambda_k) \tag{4}$$

$$\Phi\Phi^T = I \tag{5}$$

where eigenvalues are sort by descent order.

The first d largest eigenvalues and their corresponding eigenvectors are selected. And the eigenvectors ϕ_i are called Principal Components.

$$\hat{\Phi} = (\phi_1, \phi_2, ..., \phi_d) \tag{6}$$

For each k-dimension training or testing sample $t = (t_1, t_2, ..., t_k)$, the projected result is computed as below,

$$\hat{t} = t - \hat{\mu} \tag{7}$$

$$y = \hat{\Phi}^T\hat{t} \tag{8}$$

The projection matrix $\hat{\Phi}$ is $k \times d$ and t is k-dimension, so we get the new vector $\hat{t}$ d-dimension.

By controlling the parameter d, we can control the size of data and then reduce computation time. In our random forest method, we choose some ds manually and compare the performances to get a best d. In our FNN method, the parameter d is decided by this criterion.

$$r_d = \frac{\sum_{i=1}^{d}\lambda_i^2}{\sum_{i=1}^{N}\lambda_i^2} > 0.9 \tag{9}$$

3.2. Feed-forward Neural Network

Feed-forward Neural Network (FNN) is an artificial neural network where connections between the neurons form no cycle. It has been proved to be a successful statistical tool for modeling as well as prediction.

FNN is also known as multiple layer perceptrons, therefore it has at least one hidden layer. G. Cybenko [5] shows that given any continuous function $f(x)$ and $\epsilon > 0$, there exists a neural network $g(x)$ such that

$$|f(x) - g(x)| < \epsilon, \forall x \tag{10}$$

Dataset number	Training (time)	Test (time)	Range (m)
01	19:06:05-19:21:05	09:32:10-09:48:10	0:20:2960
02	11:08:05-11:22:05	03:47:10-04:02:10	1750:20:3000
03	09:37:05-09:48:30	05:45:20-05:56:05	850:20:3080
04	13:02:05-13:19:05	04:11:10-04:29:10	1000:20:2850
05	05:24:40-05:35:40	10:02:40-10:14:10	900:20:2800

Table 1: Start and end tracking time of five data sets; the range labels are provided

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

Table 2: Training and test sample numbers of five datasets

This indicates that neural network could approximate any continuous functions.

Figure 2 is an demonstration of FNN with two hidden layers. Assume the input variables $x = [x_1, x_2, ..., x_n]^T$, the jth input of hidden layer1 is a linear combination of x, which is given by

$$a_j = \sum_{i=1}^{n} w_{ji}^{(1)} x_i + w_{j0}^{(1)}, \quad j = 1, 2, ..., M \tag{11}$$

where M is the number of neurons in hidden layer1, $w_{ji}^{(1)}$s and $w_{j0}^{(1)}$ are called weights and bias, respectively.

In each hidden layer, there's an activation function which maps the input a_j nonlinearly.

$$z_j = f(a_j) \tag{12}$$

If the activation function $f(*)$ is a linear function, then our neural network will degenerate into a perceptron.

Similarly, for the hidden layer2, the inputs and outputs are given by

$$b_j = \sum_{i=1}^{n} w_{ji}^{(2)} z_i + w_{j0}^{(2)}, \quad j = 1, 2, ..., K \tag{13}$$

$$y_j = f(b_j) \tag{14}$$

where K is the number of neurons in hidden layer2.

There are several choices for the nonlinear activation functions, such as, the sigmoid function 3a, tanh function 3b, and ReLU nonlinearity 3c.

The sigmoid function is used in neural network from the very beginning.

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{15}$$

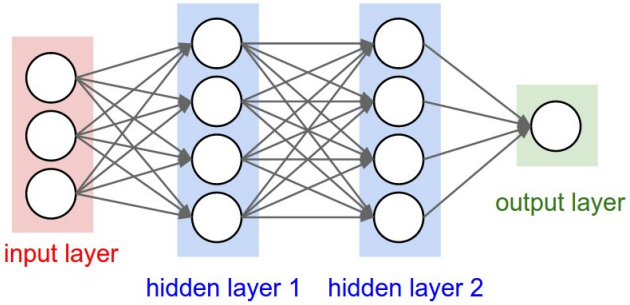
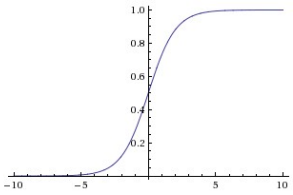
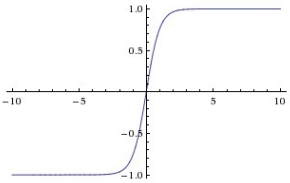


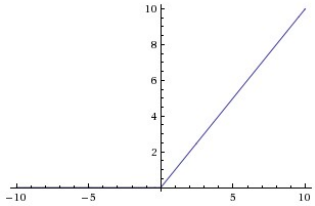
Figure 2: Neural Network with two hidden layers [6]



(a) Sigmoid



(b) Tanh



(c) ReLU

Figure 3: Three commonly used activation functions

However, when the input is very large, sigmoid becomes saturate and its gradient will vanish. This affects the process of back propagation and the network will barely learn. In addition, sigmoid is not zero-centered.

The tanh nonlinearity is actually a modified sigmoid, or a centered sigmoid.

$$tanh(x) = 2\sigma(2x) - 1 \tag{16}$$

Therefore, although it doesn't have the problem with not being zero-centered, its activations still saturate when input

is large, which would affect the learning process.

Unlike the above two activation function, the Rectified Linear Unit (ReLU) has a constant gradient and this helps accelerate the convergence of gradient descent. It also has a simple implementation than sigmoid and tanh which need exponential operation. Therefore, we choose ReLU as our activation function.

$$ReLU(x) = max(x, 0) \tag{17}$$

The last layer in FNN is called output layer, and it does not need activation function. In our multi-class classification task, we use the softmax function instead. The softmax function can transform the output results into probabilities.

$$P_i = \frac{exp(y_i)}{\sum_{j=1}^{m} exp(y_j)}, \quad i = 1, 2, ..., m \tag{18}$$

where m is the number of classes, and this implies that our output layer contains m neurons. And the input sample will be assigned to the class with largest probability.

In order to use back-propagation to make our neural network converge to the real underlying relationship between input variables and the labels, we adopt cross-entropy as our loss function E,

$$E(w) = -\frac{1}{N} \sum_{n=1}^{N} \sum_{i=1}^{m} t_i^n ln P_i^n(w) \tag{19}$$

where t_i^n is the real probability of nth sample in class i, which implies that t_i^n is binary and can only be 0 or 1, N is the number of sample, m is the number of classes.

The best parameter w is given by

$$w^* = \arg\min_w [-\frac{1}{N} \sum_{n=1}^{N} \sum_{i=1}^{m} t_i^n ln P_i^n(w)] \tag{20}$$

However, w^* cannot be computed directly as there's no closed-form formula. Lots of optimization methods are developed for computing w^*. In our paper, we adopt batch gradient descent method, and our update equation is,

$$w^{(t+1)} = w^{(t)} - \eta \sum_{n=1}^{N} \nabla E(w) \tag{21}$$

3.3. Random Forest

The second method used is Random Forest (RF) proposed by Breiman et al. [3], a popular machine learning method which consists of Classification and Regression Trees (CART) proposed also by Breiman et al. [4]. Compared to other machine learning methods like AdaBoost or Bagging, CART is a conditional weighted method which only uses important features to do the classification. Additionally, an advantage of random forest over CART is that random forest never overfits and every single tree can split to the end.

3.3.1 Classification and Regression Tree

CART is a non-parametric machine learning method which contains classification trees for categorical data and regression trees for continuous data. In this paper, classification tree has been used since the data is categorical.

It is straight-forward to implement the CART. As described in [10], the input dataset $\mathcal{D}$ which consists of N samples of d-dimension is first considered as the root of the classification tree. Then, it loops through all features to find the best feature to split the dataset into 2 subsets. There are two metrics that are mostly used to determine which specific feature to split the dataset: *Gini Impurity* and *Information Gain*. Gini impurity measures the probability of the data, which is randomly chosen, being misclassified if the label is also randomly assigned.

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

where J is the number of classes and f_i is the portion of data which belongs to class i. Information gain is calculated based on the concept of entropy which gives similar results as Gini impurity.

$$Entropy = -\sum_{i=1}^{J} p_i \log_2 p_i, \quad i = 1, 2, ..., J \tag{23}$$

where p_i is the probability of class i and the sum of p_i should be added up to 1. The information gain is calculated as the difference between the entropy of the node and its children.

$$IG = Entropy(T) - Entropy(T|a) \tag{24}$$

The dataset keeps splitting based on either of these two metrics until all the nodes cannot be splitted anymore or a specific threshold is reached.

However a CART is often overfitted if every node is split exhaustedly. Therefore many other early-stopping methods like pruning are introduced when training a CART.

3.3.2 Random Forest

In this paper, random forest is used to prevent overfitting instead of using early-stopping method in training a CART. The same input dataset $\mathcal{D}$ and a random forest $\mathcal{F}$, which is made of T CARTs, are used here. As discussed in [3], each tree is trained by using n samples ($n \leq N$) which are randomly drawn from the original dataset $\mathcal{D}$ with replacements. Additionally, every tree is grown to the largest extent which means that no pruning is needed during the training. Once a new test data is inputed into the completely trained random forest, each tree will assign a label to the input data

and the class which gets the maximum votes will be the final classification label. An example of random forest is shown in 4.

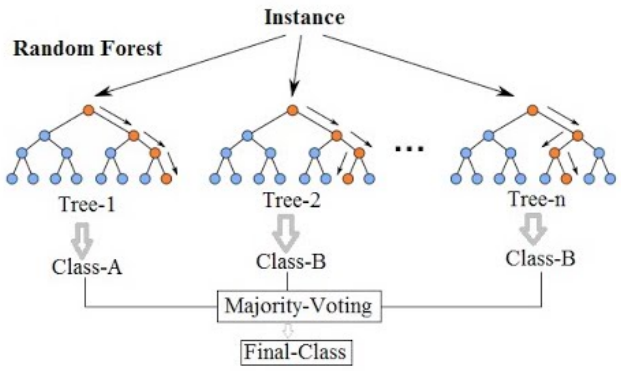


Figure 4: Illustration of random forest

3.3.3 Implementation

We implemented random forest classifier in Python by using *scikit-learn* package with that the number of trees has been set to be 500. To compare the results quantitatively, the mean absolute percentage error (MAPE) over N samples is used.

$$E_{MAPE} = \frac{100}{N}\sum_{i=1}^{N}|\frac{R_{p_i} - R_{g_i}}{R_{p_i}}| \tag{25}$$

Because of the large number of features dimension (7200) with relative small number of the training samples (around 800), feature reduction is necessary. PCA is used to reduce the dimensions of the feature space. We compared the results using different numbers of feature dimension which are used to train the random forest for each dataset (shown in Table 3). *Dataset01* is used in this test and the random forest is trained using Gini impurity and number of samples used to train a single CART is the square root of the number of total features. From the result of this test, we found out that reducing the feature space dimension to 100 could produce better results.

# of Dimension	100	200	300	400	500
E_{MAPE}	4.24	5.32	4.40	5.22	11.22

Table 3: Error with different number of feature space dimension by Gini impurity and sqrt number of features. Dataset01 is used in this test.

The second test we carried out is to test the impact of using different metrics (Gini impurity and information gain(IG)) and number of samples(square root of the total features and $\log_2$ of total features) to train one CART. The input data for this test is all the 5 datasets with feature space dimension reduced to 100 (Table 4). From the results we can tell that the combination of Gini impurity and square root of total feature numbers leads to better results.

Dataset	log2+Gini	log2+IG	sqrt+Gini	sqrt+IG
01	5.56	6.60	5.93	6.79
02	6.22	9.14	2.73	4.70
03	22.32	23.97	7.39	18.85
04	3.16	7.47	1.40	3.17
05	30.89	30.60	25.93	28.32
Avg.	13.63	15.56	8.68	12.37

Table 4: Error using Gini impurity vs. information gain, and sqrt vs. log2 number of features

Based on the results of these 2 test, we decided to use Gini impurity as the metric to split a node, square root to calculate the number of samples used to train a CART, and 100 as the dimension of the reduced feature space for tests in the next section.

4. Experimental Results

4.1. Results of FNN

Although one hidden layer performs well in [8], a more complicated machine learning algorithms need to be studied for better performance. We built an FNN with 3 hidden layers for testing. The number of neurons in the first hidden layer is 2 times of the number of features after PCA. In the second hidden layer, the number of neurons is the same as the number of features. There are 200 neurons in the third hidden layers. Since the FNN performs best with snapshots 5 in Niu[8], the number of snapshots was set to 5 during processing data.

Figure 5 shows our experimental results of FNN predication within 5 datasets. If the range of the point has two corresponding time index, the data samples performs better estimation than that range of points has only one corresponding time index. For example, from figure 5 dataset01, the range from 0 to 1500 meters, the estimation is better than the range from 1500 to 2960m. Because the more training data the system has, the better performance of representing underlying relationship between training and test data. Therefore, in the range of 1500 to 2960m, if the data is outlier, system will fit tendency of the noise instead of the underlying relationship. The calculated E_{MAPE} statistics are shown in the Table 5. The average error is 6.26 which is smallest within the three methods.

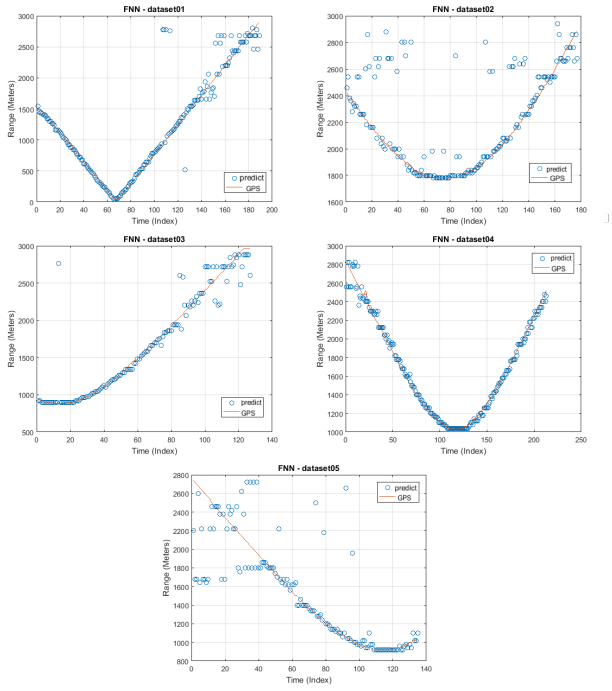


Figure 5: FNN results: from left to right, top to bottom are the results for Dataset01 - Dataset05

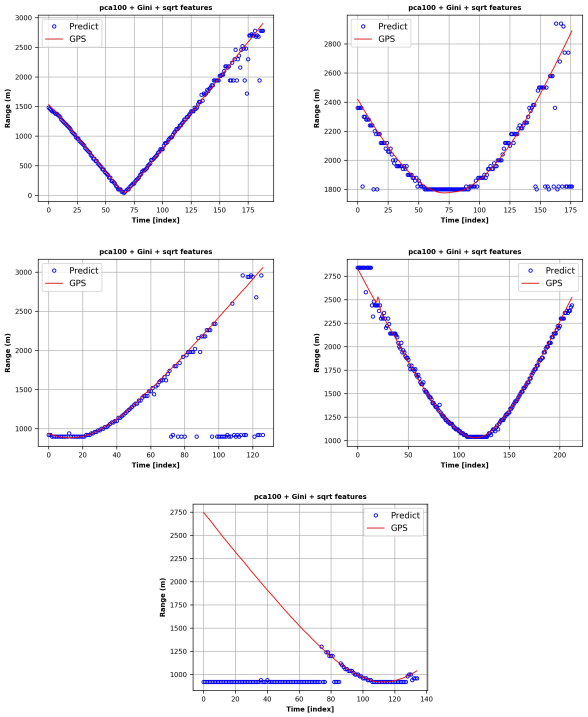


Figure 6: RF results: from left to right, top to bottom are the results for Dataset01 - Dataset05

4.2. Results of RF

As discussed in the previous section of RF, the PCA reduces the dimension feature space from 7200 to 100. Gini impurity is used to split a node and the square root is calculated for the number of samples to train a CART. From Figure 6, RF works pretty good for dataset01 and dataset04 while the performance got worse for dataset03 and dataset05. A conclusion can be drawn that the symmetry of the dataset (distance from the ship to the receiver) could impact the performance to a large extent. The average E_{MAPE} of five datasets is shown in the Table 5. The average MAPE is 8.00.

4.3. Results of SVM

The SVM method was provided by Prof.Gerstoft. The results are shown in Figure7. The E_{MAPE}, shown in table5, is 10.49 which is biggest within three methods.

4.4. Robustness against noise

Besides comparing the performances among these three statistical methods, we also test our models against different levels of additional Gaussian noises. The FNN model and random forest model are trained on the original data without noise, and the test data is added with white Gaussian noise in different levels (signal-noise-ratio(SNR) from -40dB to 10dB).

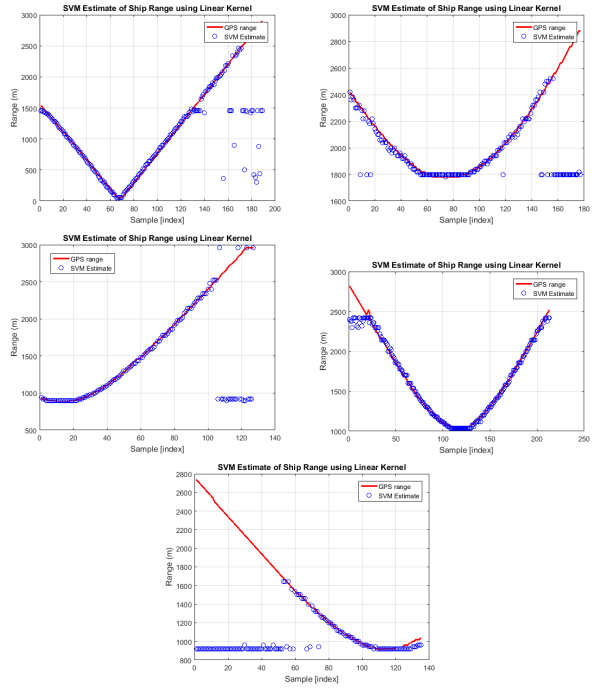


Figure 7: SVM results: from left to right, top to bottom are the results for Dataset01 - Dataset05

Dataset	FNN	RF	SVM
01	9.52	4.86	10.22
02	6.11	3.95	5.70
03	4.32	14.52	10.22
04	1.74	1.34	1.74
05	9.59	29.87	24.56
Avg.	6.26	8.00	10.49

Table 5: Comparison of three methods

We test FNN with PCA on *Dataset01*. The SNR of the noises added to the original data is $-40dB$, $-30dB$, $-20dB$, $-5dB$, $5dB$ and $10dB$. Results are shown in Table 6.

SNR(dB)	Orig	-40	-30	-20	-5
Error	9.52	191.46	22.42	9.78	9.53
SNR(dB)	Orig	5	10		
Error	9.52	8.51	9.57		

Table 6: Feed-forward neural network against noised test data

Again, We test our random forest model on *Dataset01* using Gini impurity and the square root of total number of features as the number of features used to train a CART. The input dimension of the feature space is reduced to 100. The SNR of the noises added to the original data is $-40dB$, $-30dB$, $-20dB$, $-10dB$, $-5dB$, $5dB$ and $10dB$. Results are shown in Table 7.

SNR(dB)	Orig	-40	-30	-20	-10	-5
Error	5.93	40.97	6.18	7.84	10.03	7.94
SNR(dB)	Orig	5	10			
Error	5.93	5.03	10.26			

Table 7: Random forest model against noised test data

The results in Table 6 and Table 7 indicate that our statistical models could resist addictive white Gaussian noise with SNR up to -30dB. Therefore, the statistical methods, like, FNN, random forest, are robust to noise and are able to handle some unexpected situations. This means that the models need not to be trained again and again for different conditions.

References

[1] A. B. Baggeroer, W. Kuperman, and H. Schmidt. Matched field processing: Source localization in correlated noise as an optimum parameter estimation problem. *The Journal of the Acoustical Society of America*, 83(2):571–587, 1988.

[2] A. B. Baggeroer, W. A. Kuperman, and P. N. Mikhalevsky. An overview of matched field methods in ocean acoustics. *IEEE Journal of Oceanic Engineering*, 18(4):401–424, 1993.

[3] L. Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[4] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone. Classification and regression trees. wadsworth & brooks. *Monterey, CA*, 1984.

[5] G. Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals, and Systems (MCSS)*, 2(4):303–314, 1989.

[6] A. Karpathy. Cs231n: Convolutional neural networks for visual recognition. *Online Course*, 2016.

[7] Z.-H. Michalopoulou and M. B. Porter. Matched-field processing for broad-band source localization. *IEEE Journal of Oceanic Engineering*, 21(4):384–392, 1996.

[8] R. Niu and P. K. Varshney. Source localization in sensor networks with rayleigh faded signals. In *Acoustics, Speech and Signal Processing, 2007. ICASSP 2007. IEEE International Conference on*, volume 3, pages III–1229. IEEE, 2007.

[9] K. Peason. On lines and planes of closest fit to systems of point in space. *Philosophical Magazine*, 2(11):559–572, 1901.

[10] M. P. Vayssières, 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):679–694, 2000.