CSC 592 Deep Learning
Project Abstract

Matt Daily

In CSC 581 Introduction to Machine Learning with Support Vector Machines, I explored the applicability of using SVMs, Random Forests (RFs), and simple Artificial Neural Nets (ANNs) to do the detection and classification for an active Doppler sensitive sonar. Due to a lack of training data, I used a simulated signal generator to create automatically tagged training and evaluation data, trained the various ML approaches across a number of settings for their meta-parameters, and compared the results to a state of the art split window normalizer. The ML technologies were shown to out-perform the normal detector, by a wide margin in some cases. [Find Prior Work Attached]

It is my intention to continue this work here, where I will generate much more detailed data using an updated simulation that has added fidelity, more realistic clutter, and other forms of interference that often render the normal detectors useless. I will use complex target models, ones that include non-linear scattering effects, and I will also add jammers to the signal stream. The dimensionality of the input data will be two orders of magnitude higher than in the prior work and will require a Deep Neural Net to operate.

Specifically, I am imagining the input “image” to be approximately 15x128x64 16 bit level values, where 15 is the number of beams, 128 the number of frequency bins, and 64 the number of frames in a listening cycle. To start each point will be a 16 bit unsigned value.

It is my intention to use a net consisting of several convolutional layers to do the detection and clasification. I also intend to explore the potential of using a Generating Adversarial Network to cope with the various interference sources in the data. This is an area of great interest (not that I really understand it right now) because, with this high fidelity simulation in hand, the generator could be seeded from the simulation in some way.

The output will be a 64x128x3 matrix of values, 3 per time/frequency bin, with the 3 being a probability of detection and two angle estimates. The detection plane may have a softmax layer, though it is entirely possible that there are multiple detections in a cycle and so that may not make sense.

Using the simulation I hope to generate on the order of 30,000 training instances. The simulation is written in MATLAB, and as this will require computational capability, I will be looking to run it on a 48 core XEON computer at work.

CSC 581 Final
Simulation Based Training of Active Sonar Further Investigation

Matt Daily
CSC 581 Mid Term Project
Due 5/10/2018

Introduction and Mid Term Project Summary

My specific area of interest in machine learning is in simulation-based training, specifically for sensor systems. I do work with sonar systems and I believe that the many legacy detection systems, which require hundreds of hours of development, can be replaced with properly trained machine learning algorithms. I believe these models can be trained with simulated data, and I have developed simulations for other purposes in the past that can be used for this.

The cost of developing detectors for these sonars is very high: the algorithms are a mix of matched filters, detectors, clusterers, classifiers, etc. Some are mathematically based but many are heuristically developed and require the input of a Subject Matter Expert (SME) and hundreds of hours of develop/test/modify work. These algorithms are then tested in a set of limited environments and encounters and the algorithms modified to maximize effectiveness in these often very artificial settings.

With a high fidelity simulation in hand, developing a ML algorithm for a given sonar is mostly a matter of crank turning, so doing this would represent a huge savings in time and development cost. It would also allow us to train for environments and encounters that cannot be easily tested. This could result in a considerable reduction in risk.

In the midterm project, I adapted a simple simulation and used it to train a small SVM for the purposes of detection. I did an exploration of different kernels, different cost factors and kernel parameters. I compared the performance of this trained detector to a state of the art detection method called a split window normalizer. I found that the SVM was more effective than the split window normalizer. I also found that a radial kernel with a cost of 100 and a gamma of 0.01 was the best performing.

Objective and Tasks

The primary objective of this investigation is to prepare a machine learning detector for a real world sonar, as the mid-term investigation was really targeted at a basic proof of concept. In preparation for a real application, the following steps will be executed:

1. Simulation improvements to reflect more of the effects of an actual sonar
2. Assess the impact of the new simulation on the mid-term SVM design
3. Investigate other ML approaches including Random Forest and ANN
4. Investigate modifications to the feature set to overcome any reduction in effectiveness found as a result of simulation improvements

Because of run time issues encounter in the prior work, all work for this effort was undertaking not in R but using the Python scikit-learn package. The models used were specifically:

- `svm` from `sklearn`
- `RandomForestClassifie` from `sklearn.ensemble`, and
- `MPLClassifier` from `sklearn.neural_network`

Simulation Improvements

The mid-term project resulted in the creation of a basic MATLAB sonar simulation. This simulation included no beam pattern, no boundary reverberation, and a simplified target model. This simulation was improved in several ways to more accurately reflect real world effects, specifically those we expect to see in the sonars of interest. A quick run down of the additions and their expected effects is given.

Target Alignment

The sonar detector developed for this work takes its features from a time-frequency spectrogram of the sonar return as seen in Figure 1. This image consists of the power of an FFT executed. in this case, on 64 continuous samples. Each such FFT is termed a “frame”. The frames of samples, all the same length, are taken over different sets of points. The sonar pulse in this case is a simple windowed CW, and the spectrogram frames correspond to the length of this pulse.

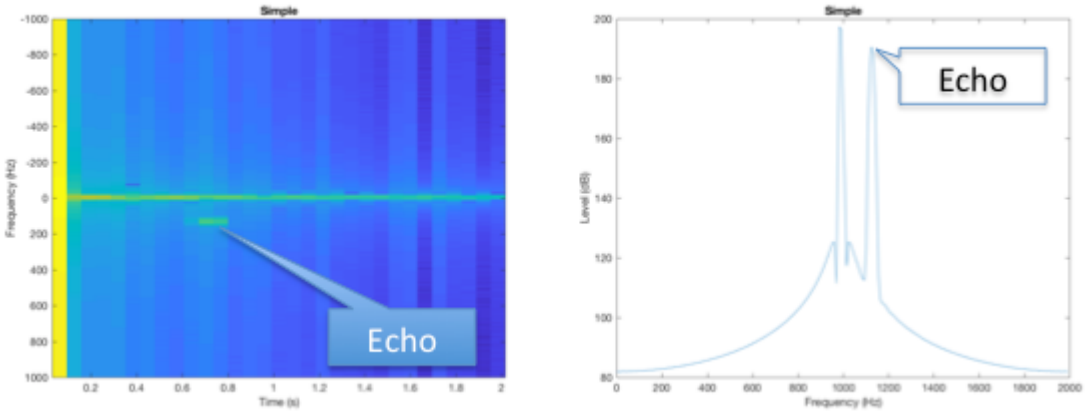

Figure 1: Simple Echo Example

In the mid-term project, the spectrogram frames were spaced so that the spacing was the same as the pulse length, and the echoes were constrained to fall entirely within one frame. This was done to simplify the labeling process: one frame, one echo.

For this work, the simulation was modified to relax this limitation. The echoes were allowed to fall anywhere in the output stream so that the echo would, in general, contribute energy to two frames as it would generally span two frames. For labeling purposes, both frames to which an echo contributes are labeled as containing a target.

Boundary Reverberation

In real water, depending on the bottom compositions (rock, sand, silt, mud) and roughness of the water, the reverberation from the boundaries can be much higher than the reverberation from the volume. Worse yet, it changes drastically over time as the scattering area evolves with the outgoing wave front. The mid-term simulation had no boundary reverberation. For this work, a simple boundary reverberation model was added to the simulation to reflect this.

Details of such a model are the subject of much study but a basic understanding of their impact is given in Figure 2. This corresponds to a 600 meter water depth and a 200 meter platform depth. The parameters of the environment were picked to illustrate the issues of boundary reverberation, which is in this case highly non-stationary. We can see how the onset of the surface reverberation peaks quickly then falls off, and then the bottom reverberation goes through a similar evolution later in the cycle. We can also notice how these components have Doppler evolution as the scattering annuli spread out from the source.

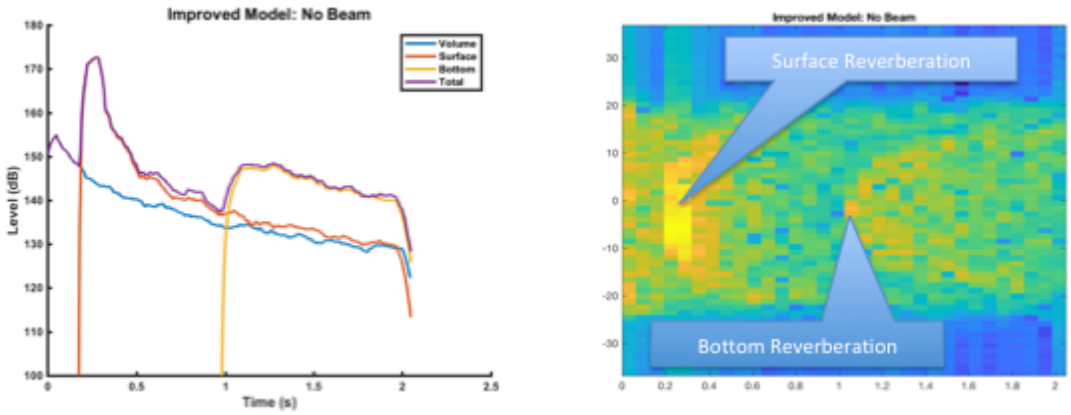

Figure 2: Boundary Reverberation Example

Beam Pattern

I order to battle boundary reverberation, real world sonars use directional transmitters. For this work, a representative transmit beam pattern was added to the simulation. The beam pattern used is a simple circular piston with an aperture of 4 wavelengths. This reduces the reverberation and enhances the echo when it is directed at the target. This becomes especially important with the boundary reverberation. For a moving target it also shapes the reverberation spectrum by selectively enhancing reverberation in the direction of the beam. In order to keep the dimensionality of the training data down, in this work, the beam is always aimed at the target: future work would relax this assumption.

In Figure 3, we can see this effect. The reverberation, which was spread across the whole band in Figure 2, is now peaked at the highest Doppler, as the beam is steered in the direction of motion. Also notice that surface reverb onset is very target like now, concentrated at a low Doppler.

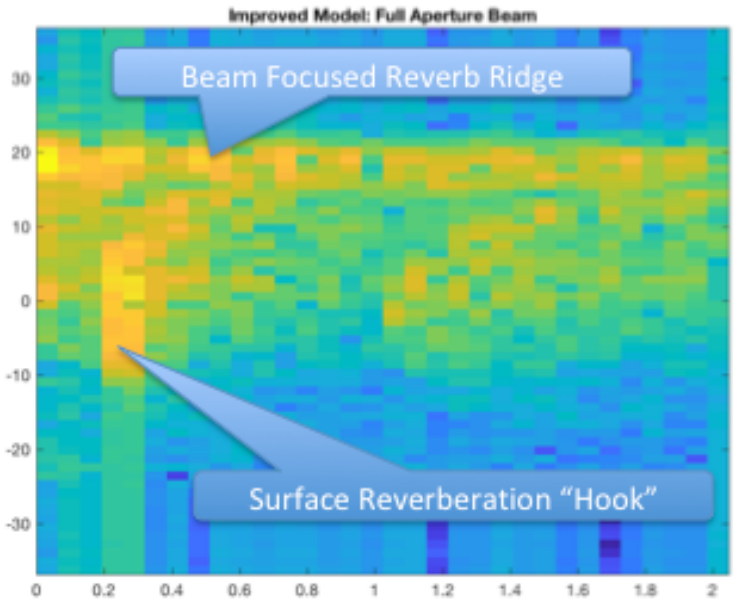

Figure 3: Boundary Reverb With Beam Pattern

Shallow Water

Reverberation in shallow water is much more a problem than in deeper water. Thus an environment was created that was only 150 meters deep with a rough surface and a moderate bottom (silt). This was picked to choose a very challenging environment.

Training Data Sets

Using this updated model, five training data sets were generated with each of these effects turned on one by one. Specifically these data sets are termed:

- **Legacy** – the "Quiet" data set used in the mid-term project
- **Unaligned** – the legacy data with the echos not aligned with the frames and overlap enabled
- **Boundary** – the unaligned data with boundary reverberation added
- **BeamPattern** – the boundary data with a transmit beam pattern added
- **Shallow** – the beam pattern data but switched to a shallow water environment

Each data set has the same target data in it but with each of the new capabilities enabled. The target strength is -5 dB

Neural Net Investigation

In order to assess the effects of the different models, several structures of neural nets were tried. Specifically, 1 and 2 layer designs with 4, 8, 16, 32, and 64 neurons per layer were tried. The training was done using the stochastic gradient descent solver and a sigmoid activation function. The results are given in Table 1, where we can see that a single layer with 64 neurons provides the best balance of performance and simplicity.

Neuron Count	1-Layer	2-Layer
4	82.0	79.7
8	76.7	79.1
16	76.8	83.5
32	77.3	86.1
64	86.4	85.4

Table 1: Neural Net Performance

Environmental Effects Investigation

A single model of each of the three classes was used to compute a 10 fold Cross Validation Error on each of the data sets.

- For the SVM, the optimal parameters for the training were taken from the mid-term project: a gamma of 0.001 and a cost of 100 and a radial kernel
- For the random forest, the parameters chosen were the defaults which are fully populated bagged trees
- For the ANN, the parameters from the prior section were used: 1 hidden layer of 64 neurons with a sigmoid activation function.

The performance, measured as percent correct classification, is given in Table 2.

Feature	SVM(0.001,100)	Forest	ANN(1x64, sigmoid)
Legacy	85.1	89.2	86.0
Unaligned Targets	60.8	63.2	64.3
Boundary	53.6	56.0	53.7
Transmit Beam	69.3	73.7	71.3
Shallow Water	80.1	73.4	70.9

Table 2: Baseline Model Performance

From this table we notice two things. First, the three different ML models perform very similarly: there is no one model that significantly outperforms the others in all environments. To test this, I generated confidence intervals for the legacy data set. We can see that the three models are statistically the same in this case. We presume the rest of the environments would show similar interval overlap.

Model Type	Lower Bound	Upper Bound
SVM(0.001,100)	84.2	87.6
Forest	85.1	91.2
ANN(1x64,sigmoid)	85.5	87.0

Table 3: Confidence Intervals For Legacy Data Set

The second thing to notice is the general trend. The loss of alignment of the echo with the frames costs 20-25% loss of performance, and the addition of boundary reverberation another 7% loss. The addition of the transmit beam gets back about 16%. The shallow water does not impact performance very much, which is initially surprising. However, as shown in Figure 4, the shallower water results in a less prominent boundary onset which may have unintentionally helped the models.

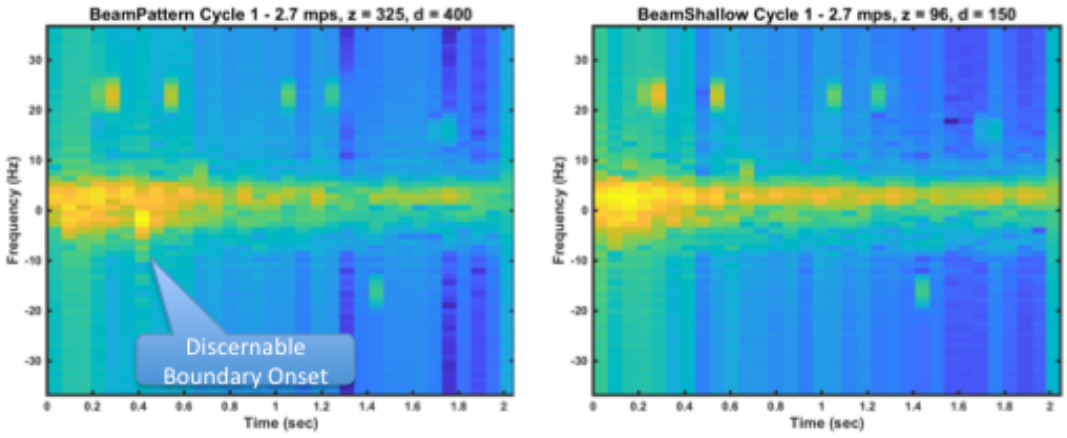

Figure 4: Deep .vs. Shallow Cycles

Inclusion of Time History

All the learning methods suffered significantly when the echoes ceased being aligned with analysis frames. Figure 4 shows the effect of this, which is to spread the energy of each echo across multiple frames and thus reduce the level and spread the spectrum a little bit.

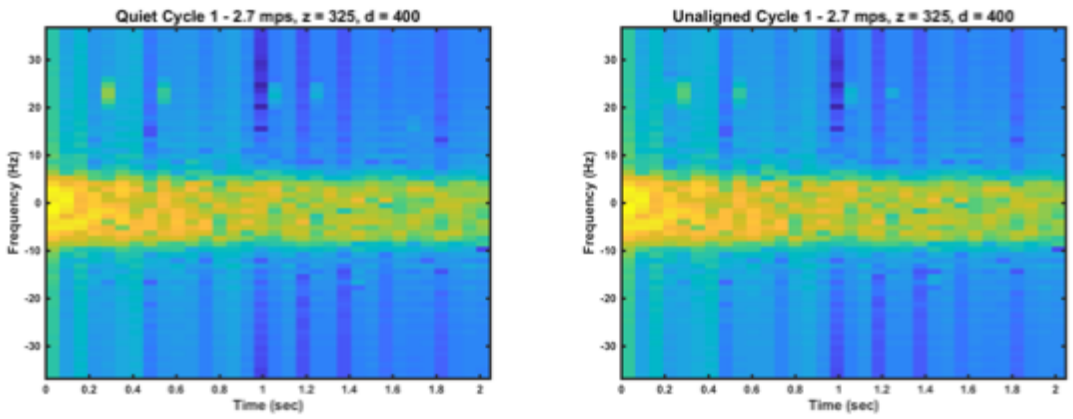

Figure 5: Comparison of Aligned and Unaligned Targets

One suggestion would be to increase the number of features to include the spectral estimates just ahead and just behind each frame. This way all the energy of the pulse is available in the training and prediction. This was attempted with all methods on the unaligned data set.

The number of frames before and after the current frame (termed the extent) included in the feature set was varied from 0 to 4, with the results given in Table 3. Note that because of this inclusion, there are slightly fewer data sets to train and evaluate as this method requires extra contiguous frames before and after each data record. The training run times were not excessive so it was not necessary to break the frames into sub-spectra to do this investigation as was feared in the proposal.

Extent	SVM(0.001,100)	Random Forest	ANN(1x64,sigmoid)
0	60.8	64.0	56.3
1	53.1	59.4	53.3
2	52.0	60.6	51.8
3	53.8	60.4	53.0
4	52.1	60.6	52.1

Table 4: Effect of Time History on Unaligned Data Set Performance

It is very interesting to note that generally the performance degrades when any extension is given and stays constant regardless of non-zero extent. While I didn't have time to run CI on this data, we can presume that all changes other than the transition to non-0 extent are not statistically relevant.

I found this to be surprising. Standard signal processing algorithms for this problem generally involve “template normalizers” which normalize the energy in any pixel by the energy around it in time and frequency. The result is an SNR estimate which is used directly as a detection statistic and also include the energy nearest to the pixel.

Rescaling Effects

It is known that some ML algorithms are sensitive to input scaling, and the python SVM library I am using here specifically state a lack of any input data conditioning unless optionally enabled.

With this in mind, the above algorithms were re-run with the spectral values for each record normalized to exist in the [0 1] domain and results given in Table 4.

The performance of the SVM is significantly improved while the performance of the other two models either unchanged or slightly better. Again, this is very interesting but perhaps not surprising, as SVMs are known to be more sensitive to scaling issues than forests or neural nets.

Extent	SVM(0.001,100)	RandomForest	ANN(1x64,sigmoid)
0	63.3	64.7	60.9
1	59.3	59.0	53.4
2	62.1	59.8	55.4
3	61.2	58.9	58.1
4	62.6	57.7	57.0

Table 5: Effect of Normalization on Performance

Effect Of Annotation

In all the results so far quoted, the only information given to the ML algorithms is the spectral values. There are other bits of information that are often used by conventional detectors in this setting, specifically state information of the transmitting body and knowledge of the sonar.

The data records were annotated with the following pieces of information:

- **Speed –** the speed of the body for that cycle
- **Steering –** if a beam was used, the steering for that beam
- **PlatformDepth –** the depth of the platform
- **FrameRange –** the range of the given frame

The unaligned data set was then re-run with the normalization in place but no extension. The results are given in Table 6. It is very interesting to note the effect of the annotation with frame range. The targets are only pseudo-randomly distributed in range in the simulation. I wonder if the models are reverse-engineering the mechanism by which the ranges are distributed by the simulation. On the other hand, it could be modifying thresholds as a function of range, which very similar to the way the conventional template normalizer works in the next section and is common in the literature.

Annotation	SVM(0.001,100)	RandomForest	Ann(1x64, sigmoid)
FrameRange	79.6	67.6	56.2
Speed	62.0	63.7	64.2
Steering	61.2	63.7	61.4
PlatformDepth	62.1	63.6	62.2
All	79.8	68.3	58.5

Table 6: Effects of Data Annotation

When taking as a whole, it is interesting that, with this annotation, the SVM significantly outperforms the other two models models. This is an interesting point and bears further investigation and verification. Therefore, all data sets were run with annotation included, and the results are in Table 7. This table verifies that, with these annotations included, the SVM outperforms the other two models significantly in all data sets.

Feature	SVM(0.001,100)	Forest	ANN(1x64, sigmoid)
Legacy	79.2	89.3	82.2
Unaligned Targets	79.1	66.7	59.5
Boundary	78.7	60.5	57.8
Transmit Beam	82.3	76.5	66.6
Shallow Water	82.3	76.2	67.1

Table 7: Model Comparisons w/ Annotations

Just to insure that this is a valid discover, 5/95% confidence intervals were computed for the SVM and Forest models using the worst case Boundary data set. The results are given in Table 8. Here we can see that the difference is statistically significant. Also importantly, the increase in SVM performance begins to recover all of the performance lost to the random alignment of the echoes and the boundary reverberation in all environments.

Model	5%	95%
SVM(0.001,100)	77.9	79.6
Forest	58.6	61.2

Table 8: Boundary Data Set Best Case Confidence Intervals

Comparison With Conventional

To compare this to a conventional approach, a standard template normalizer was applied to the data and a detection statistic derived from it. The size and shape of the normalizer and the threshold were gridded to find the optimal performance on the boundary dataset. Then the performance of the detector was measured in all data sets.

An example of the effect of the normalizer is given in Figure 6. The background is at a constant level more or less except around areas of detection, where the detection is enhanced and the noise reduced. This algorithm uses a template that is 5 frames by 9 bins to account for the spreading in the two domains, which is not equal due to the nature of the pulse.

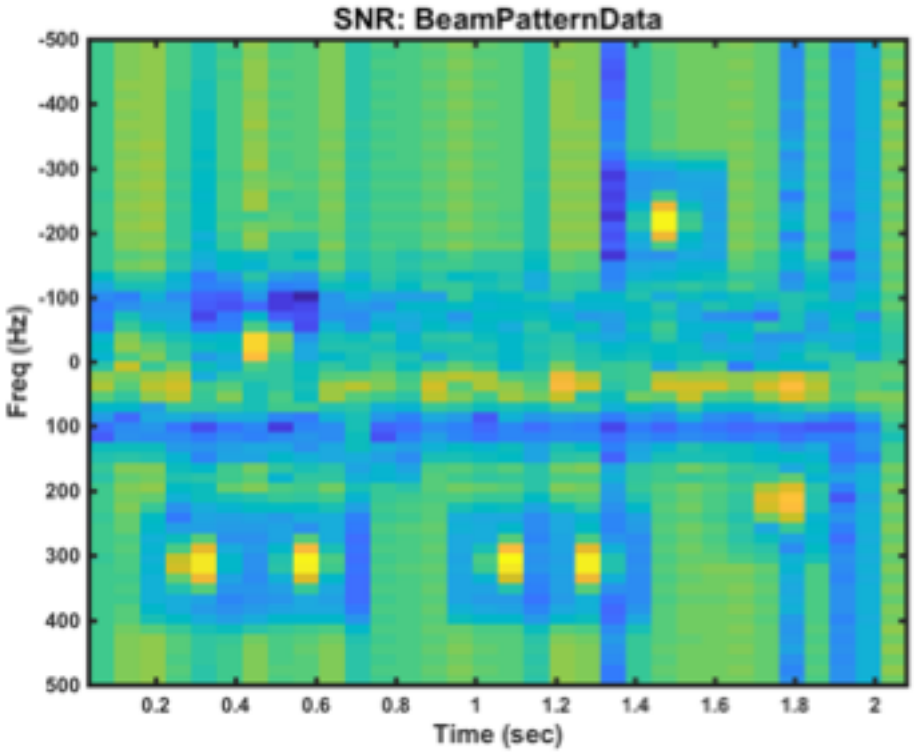

Figure 6: Normalizer Output

Using the optimal normalizer was run and the results given in Table 9. We can see that they fall far below all of the models above

Environment	Accuracy
Legacy	57.82
Unaligned Targets	53.43
Boundary	52.19
Transmit Beam	56.26
Shallow Water	54.24

Table 9: Conventional Detector Performance

Conclusion

This report looked at applying SVMs, Random Forests, and Artificial Neural Nets to the problem of detection echoes simulated sonar returns from complex environments. The simulation includes state of the art reverberation models and limited false alarm models. The best meta-parameters for these models were identified and investigations carried out on various enhancements with the following results

- As with conventional detectors, the trained ML detectors are significantly hindered by high levels of boundary reverberation.
- Including extended frame history does not significantly improve detector performance for these ML approaches, which was suprising
- Including sonar state annotations into the feature set provides a large increase in performance for the SVM and a small increase with the other two
- ***Taking the best results of all models, and including annotation and normalization, an SVM with a radial kernel, gamma 0.001, and cost factor of 100 generated the best detection result, approaching 80% accuracy***
- This compares favorably with the current conventional state of the art algorithm, the template normalizer, which is 54% accurate on the same data set.

Unfortunately due to programmatic issues, the resulting SVM was not able to be tested against a real in water data set in time for this submission. Such a data set has been identified, but access to it is in limbo, though I continue to pursue that objective. I am also considering application of these technologies to later stages of the processing chain: trackers, multi-ping classifiers, and adaptive sonar controllers. I am pursuing funding to do this as best I can.

If you have any questions please see the code I uploaded or feel free to contact me by email.