Classification of Infrasound Events with Various Machine Learning Techniques

José CHILO
Royal Institute of Technology, S-106 91 Stockholm
and University of Gävle, S-801 76 Gävle, Sweden
Roland OLSSON
Ostfold University College, N-1757 Halden, Norway
Stig-Erland HANSEN
Ostfold University College, N-1757 Halden, Norway
and
Thomas LINDBLAD
Royal Institute of Technology, S-106 91 Stockholm, Sweden

ABSTRACT

This paper presents classification results for infrasonic events using practically all well-known machine learning algorithms together with wavelet transforms for pre-processing. We show that there are great differences between different groups of classification algorithms and that nearest neighbor classifiers are superior to all others for accurate classification of infrasonic events.

Keywords: Classification, Machine Learning, Pattern Recognition, Wavelets

1. INTRODUCTION

Infrasound is low frequency sound, typically of a frequency of a few Hertz to 20 Hertz. Due to its inherent properties, infrasound can travel distances of many hundreds of kilometers. Infrasound signals can result from nuclear explosions, volcanic eruptions, mountain associated waves, auroral waves, earthquakes, meteors, avalanches, severe weather, quarry blasting, air/spacecraft, gravity waves, microbaroms, opening and closing of doors, trains and helicopters to name but a few. An infrasound monitoring system operating locally like the Swedish-Finnish Infrasound Network[1] or worldwide like CTBTO[2] must be capable of detecting and verifying infrasonic signals of interest and discriminating them from other unwanted infrasonic signals. Characterizing, discriminating and classifying infrasonic events therefore are tasks with possibly far reaching applications in very different disciplines.

An important element for successful classification of infrasound data is the pre-processing techniques used to form a set of feature vectors that can be used to train and test the classifiers. In this work we use continuous wavelet transforms to pre-process infrasound data. Wavelet transformations have proven to be a valuable tool for signal characterization [1,2]. The wavelet transform methods developed over the years at IRF Umeå [3, 4] are used in this paper.

[1] http://www.umea.irf.se/maps/
[2] http://www.ctbto.org/

Machine learning provides the technical basis to extract implicit, previously unknown, and potentially useful information from infrasound data. The idea is to build computer programs that sift through infrasound datasets automatically, seeking regularities or patterns. Strong patterns, if found, will likely generalize to make accurate classifications on new data. We use a variety of machine learning methods, including neural nets, support vector machines, decision trees, association rules, linear models, Bayes nets and others.

The advantages of neural network based approaches for classifying infrasonic events have been recognized for a while [5, 6]. Neural networks are considered to be powerful classification tools because of their non-linear properties and the fact that they make no explicit assumptions about the distribution of the data. We experimented with several neural network classifiers, including back-propagation classifiers, minimum least square linear classifiers, normal densities based quadratic classifiers, automatic neural network classifiers and random neural network classifiers. Comparing their performance, we reached the conclusion that neural network classifiers by back-propagation work the best among neural net techniques in our case, but are clearly inferior to many other machine learning methods.

2. FEATURE SELECTION

Wavelet transforms methods are used to pre-process infrasound data. The data pre-processing steps to extract feature vectors are as follows.

1.For a time series of N values a Morlet wavelet transform is performed with 128 dilations. Thus, three matrices, **A**, **R** and **I**, are obtained. The matrix **A** is a matrix of magnitudes of wavelet coefficients, w_{ij}:

$$A = \{| w_{ij} |\} \quad i = 1,...N \quad j = 1,...128 \qquad (1)$$

R and **I** contain the real and imaginary parts of w_{ij}.

2.A kind of band-pass filtering of wavelet coefficient magnitudes is performed. The entire range of coefficient magnitudes, 0 to max (w_{max}), is divided into 20 intervals

such that the *k-th* interval is limited by:

$$w_{\max} * \frac{(k-1)}{20} \text{ and } w_{\max} * \frac{k}{20} \; ; k = 1,...20 \qquad (2)$$

For each *k* the coefficients outside the range defined by Eq. (2) are identified and zeroed in matrices **R** and **I**, creating two new matrices $\mathbf{R_k}$ and $\mathbf{I_k}$. The inverse wavelet transform is performed using $\mathbf{R_k}$ and $\mathbf{I_k}$ and a new version of the original time series, $y_k(t_i)$ is created. Thus, the time series $y_k(t_i)$ is what the signal would look like if only a narrow range of spectral densities would be present in the signal.

3.The operation is repeated 20 times over the range of coefficient magnitudes. A new real-valued matrix **Z**, consisting of 20 rows and N columns is created. Each row corresponds to a time series, $y_k(t_i)$.

4.Each row of the matrix **Z** is wavelet transformed as in step 1, resulting in 20 matrices. Then these matrices are time-averaged (average along rows) leading to 20 arrays with 128 elements. A new matrix **Y** is constructed with the 20 arrays as rows. This matrix **Y** is what we call Time Scale Spectrum (TSS) of the time series.

A 3-D plot of the matrix **Y** may be constructed, showing the time scale (1/frequency) of the signal on the x-axis, the wavelet coefficient magnitude of the original signal, in percent of its max value, on the y-axis and the wavelet coefficient magnitude (power spectral density) of the decomposed components as the colour scale.

For the infrasound signals the TSS may be useful to resolve different frequency components. This feature extraction process is invariant with respect to record length, sampling frequency, signal amplitude and time sequence length. Figure 1 shows an infrasound signal train with 2^{12} = 4096 components, representing 227.55 seconds sampled at 18 Hz and its TSS.

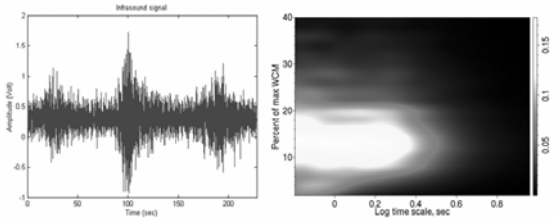

Fig. 1. *The infrasound signal from a meteorite and its TSS*

3. CLASSIFIERS

In this section, we first describe classical neural nets trained with back-propagation and then machine learning algorithms in WEKA.

Back-propagation neural nets (BPNNs) typify supervised learning, where the task is to learn to map input vectors to desired output vectors. The back-propagation learning algorithm modifies feed-forward connections between the input and the hidden units, and the hidden and outputs units, so that when an input vector is presented to the input layer, the output layer's response should be the desired output vector. During training, the error caused by the difference between the desired output vector and the output layer's response to an input vector propagates back through connections between layers and adjusts appropriate connection weights so as to minimize the error [7].

WEKA [8] contains practically all common machine learning algorithms except neural nets. However, not all of those algorithms have support for both numerical features and nominal classes. In addition, we experienced problems with four of the algorithms, making it impossible to use them. All in all, we ended up running 22 different algorithms with their default parameters if nothing else is stated.

These algorithms are grouped into five groups in WEKA according to what models they create. The first group, Bayes, includes algorithms where learning results in Bayesian models**.** NaiveBayes is an implementation of the standard naïve Bayes algorithm, where a normal distribution is used for numerical features. BayesNet creates a Bayesian Network with the ability to represent the same model as NaiveBayes or other more complex models where the independence between features is not assumed.

The second group, Lazy, is comprised of algorithms that delay construction of classifiers until classification time. IB1 is a nearest-neighbor algorithm classifying an instance according to the nearest neighbor identified by the Euclidean distance as explained in [9]. IBK is similar to IB1 except that the k nearest neighbors are used instead of only one. We determined the appropriate number of neighbors using leave-one-out cross-validation. Another algorithm is LWL (Locally weighted learning), which differs from the other two algorithms since it only uses a nearest-neighbor algorithm to weight the instances in the training set before applying another classification algorithm to them. We chose naïve Bayes because it is recommended for classification problems by the creators of WEKA.

The third group, Rules, contains methods that create classification rules. OneR is the simplest of all the rule inducers and learns a single rule using only a single feature. The other four algorithms are more complex since they create several rules. NNge is a nearest-neighbor algorithm which learns rules based on the hyper rectangles it divides the instance space into [10]. JRip is an implementation Cohen's RIPPER [11]. RIPPER creates first a default rule and then recursively develops exceptions to it. Part constructs rules based on partial decision trees.

The fourth group, Functions, contains algorithms representing their learnt models as mathematical formulas.. SMO is a sequential optimization algorithm for building Support Vector Machines (SVMs) [12]. We used a polynomial kernel which is the default in WEKA. RBFNetwork is an implementation of radial basis functions, and SimpleLogistic constructs linear logistic regression models.

The fifth group, Trees, includes algorithms that create trees as models. Four of the six tree inducers create trees with a single class at the leaves. RandomTree learns a multi-level tree constructed by randomly choosing the splitting criterion. RandomForest is an implementation of Breiman's random forest [13], where bagging and random trees are combined. J48 is an implementation of the popular C4.5 [14]. REPTree is similar to C4.5 since it finds the splitting criteria based on information gain, but it uses reduced-error-pruning to prune the tree instead of pessimistic training error.

The last two algorithms have models in the leaves instead of a specific class. NBTree builds a tree with naïve Bayes classifiers at the leaves, where reduced-error-pruning controls the depth of the tree. LMT creates a tree with linear logistic regression models at the leaves.

The last group, Miscellaneous, contains algorithms that do not fit into any of the other groups. HyperPipes finds ranges (max and min values for numerical features) for each feature and class pair. An instance is classified as the class with the most "hits" into its ranges. VFI, on the other hand, finds intervals for each feature, and attributes each class according to number of instances with the class in the training set for the specific interval. Voting is used to select the final class for an instance. Both of these algorithms are simple compared to the other algorithms and extremely fast.

4. EXPERIMENTAL RESULTS

The feature vectors, TSS, were extracted from time-domain event signals resulting in two dimensional 20x128 matrices. These matrices were converted to 2560-element one dimensional feature vectors. Figure 2 shows two sets of feature vectors of the two different types of events.

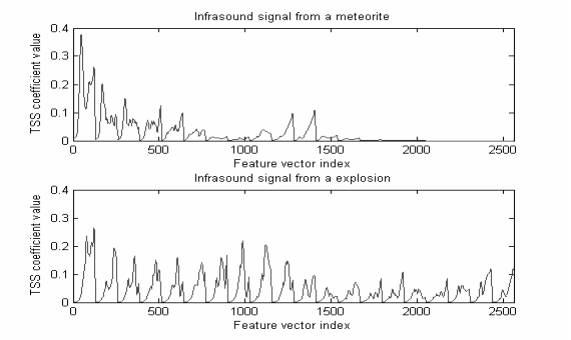

Fig. 2. *Two feature vectors*

Experiment 1: One Infrasound Category

In this experiment, we made a total of 200 infrasound measurements of 10 different doors being opened and closed. We chose to use 100 of these examples for training and the remaining 100 for testing.

The experiment was conducted with the MatLab neural network toolbox for BPNN and the WEKA toolbox for other machine learning algorithms. Each classifier was trained using 10 samples of every door and tested with a different set of 10 samples of every door. The classification results are shown in table 1 for BPNN and in table 2 for the WEKA machine learning algorithms.

The 2560/200/10 architecture was selected for BPNN, which means that the input layer has 2560 neurons, the hidden layer 200 neurons (this number was picked after we experimented with different number of neurons, see Table 1) and the output layer 10 neurons. The output layer is to produce target output as [{1000000000},{0100000000},…}]. The network is a two-layer log-sigmoid/log-sigmoid network. The log-sigmoid transfer function was picked because its output range (0 to 1) is perfect for learning to output Boolean values. All training was done using back-propagation with both adaptive learning rate and momentum. The network was trained for a maximum of 5000 epochs or until the network mean squared error (MSE) falls beneath 0.01. The final MSE was 0.00982 after 300 training epochs. The results give a 24 % error.

Table 1. *Results from BPNN*

Architecture	Training epochs	Error %
2560/20/10	903	34
2560/40/10	1281	31
2560/100/10	307	28
2560/150/10	300	27
2560/200/10	300	24
2560/300/10	315	23
2560/400/10	290	22

The classification results for machine learning methods in WEKA vary greatly between algorithms and between different groups of algorithms, see Table 2.

Table 2. *Results from WEKA*

Group	Algorithm	Error %
Bayes	NaiveBayes	12
	BayesNet	12
Lazy	IB1	7
	IBK (Cross-validation)	7
	LWL (NaïveBayes)	11
	IBK (5)	12
Rules	NNge	10
	Part	27
	Ridor	34
	JRip	40
	DecisionTable	41
	OneR	60
Functions	SMO	8
	RBFNetwork	13
	SimpleLogistic	15
Trees	RandomForest	14
	LMT	15
	J48	23
	REPTree	33
	RandomTree	37
Misc	Hyperpipes	11
	VFI	42

Bayes is one of the better groups. BayesNet have similar performance to NaiveBayes, possibly due to construction of a network similar to NaiveBayes.

Lazy is by far the best group of algorithms. The best algorithms in this group are IB1 and IBK with seven percent error on the test data. In addition, the equal result of IB1 and IBK suggest that a single neighbor was chosen during cross-validation. To investigate this further, we tested running IBK with five neighbors which resulted in an error percent of 12.

LWL achieved poorer results than the other two algorithms in the group. This might be a result of it not using the nearest neighbor algorithm directly, but only for weighting the training set. Interestingly, it has similar performance to NaiveBayes, which is the algorithm used as base learner for LWL. Thus, it appears that local weighting has a small or no impact for this dataset.
Functions have done well as a group, which is not particularly surprising based on the fact that the algorithms in this group tend to handle numerical features well. SMO is one of the best algorithms overall, which shows that support vector machines are worth trying for signal classification. It seems that the more complex the method, the better the result.

The Trees group is split into two groups according to the results. In the first group are RandomForest and LMT. These are methods that either build trees with models at their leaves or combine several trees. The other tree inducers, which create trees with a single class at the leaves, achieve much poorer results. These differences in performance might be the result of more complex trees having to be induced by the simpler algorithms due to simpler leaves, numerous classes and only numerical features resulting in binary splitting. In addition, the models of LMT and NBTree, logistic regression and naïve Bayes, perform well separately for this dataset.
The Rules group is probably the worst group of all. NNge is the only rule inducer that performs well, and it is a nearest neighbor algorithm. The other algorithms perform poorly, possibly due to being better at handling nominal rather than numerical features.

The algorithms in the Miscellaneous group vary greatly in performance. Hyperpipes have a low error percent, while VF1 have a high error percent. This shows that one should always test the simplest algorithms first before using the more complex and computational intensive methods.

Experiment 2: Four Categories

Four categories of infrasound events are of interest in this section. The data were collected from different infrasound sensor arrays with different geometries and different locations. Details of the data are given in Table 3.

Table 3. *Infrasound data summary*

Event Type	No. of Events	No. of Samples
Meteorites	4	30
Vehicle	3	27
Man-made explosion	8	24
Opening-closing doors	10	27

To measure the classification accuracy, a 10-fold cross-validation technique is used in this experiment. That is, the whole dataset is partitioned into 10 subsets. Then 9 of the subsets are used as the training set, and the tenth is used as the test set. This process is repeated 10 times, once for each subset used as the training set. Classification performance comes from the average of these 10 runs. This technique ensures that the training and test sets are disjoint, see Table 4.

Table 4. *Results from WEKA*

Group	Algorithm	Error %
Bayes	NaiveBayes	16
	BayesNet	6
Lazy	IB1	1
	IBK (Cross-validation)	1
	LWL (NaïveBayes)	12
	IBK (5)	6
Rules	NNge	19
	Part	12
	Ridor	13
	JRip	19
	DecisionTable	10
	OneR	21
Functions	SMO	6
	RBFNetwork	12
	SimpleLogistic	6
Trees	RandomForest	7
	LMT	5
	J48	11
	REPTree	19
	RandomTree	18
Misc	Hyperpipes	20
	VFI	6

5. CONCLUSIONS

Features based on wavelet transform methods proven effective for the analysis and characterization of infrasound signals when combined with the best state-of-the-art machine learning methods from the WEKA toolbox. The best of these methods, IB1, yields only seven percent error on the one category test data whereas neural networks as implemented in MatLab gave more than a twenty percent error for all the architectures that we tried.
This shows how important it is to choose the appropriate machine learning algorithm for a given problem domain and that there are huge and domain specific variations between the algorithms.

ACKNOWLEDGMENT

The authors would like to thank Dr. Clark S. Lindsey for valuable comments on the manuscript.

References

[1] Liszka, L. “Categorization of Infrasonic Sources”. Paper presented at CTBT Infrasound Workshop 2000, Passau, Germany.

[2] Schmitter, E. D. “Characterisation and Classification of Natural Transients”, *Transactions on Engineering, Computing and Technology*, vol. 13, May 2006.

[3] Liszka L. and Holmström M., “Extraction of a deterministic component from ROSAT X-ray data using a wavelet transform and the principal component analysis”. *Astron. Astrophys. Suppl. Ser.* , 140, 125-134, November 1999.

[4] Liszka L., *Cognitive Information Processing in Space Physics and Astrophysics*, Pachart Publishing House, Tucson, Arizona, 2003.

[5] Ham F. and Park S., “A Robust Neural Network Classifier for Infrasound Events Using Multiple Array data”, *IEEE International Joint Conference NN*, vol. 3, 2615-2619, May 2002.

[6] Ham F., Rekab K., Park S., Acharyya R. and Lee Y., “Classification of infrasound Events Using Radial Basis Function Neural Network”. *IEEE International Joint Conference NN*, vol. 4, 2649-2654, July 2005.

[7] Wang D., “Pattern Recognition: Neural Networks in Perspective”. Ohio State University 1993.

[8] Witten I. H. and Frank E., *Data mining: Practical Machine Learning Tools and Techniques*, 2nd Edition, Morgan Kaufmann Publishers, San Mateo, CA, 2005.

[9] Aha D.W., “Tolerating noisy, irrelevant, and novel attributes in instance-based learning algorithms”, *International Journal of Man-Machine Studies*, vol. 36(2), 267-287, 1992.

[10] Martin B., “Instance-Based learning: Nearest Neighbor With Generalization”, *Master Thesis*, University of Waikato, Hamilton, New Zealand, 1995.

[11] Cohen W. W., "Fast Effective Rule Induction", *Proceedings of the 12th International Conference, on Machine Learning*, 115-123, 1995.

[12] Platt J., “Sequential minimal optimization: A fast algorithm for training support vector machines”, *Technical Report 98-14*, Microsoft Research, Redmond, Washington, April 1998.

[13] Breiman L., ”Random Forests”, *Machine Learning*, vol. 45(1), 5-32, 2001.

[14] Quinlan R., *C4.5: Programs for Machine Learning*, Morgan Kaufmann Publishers, San Mateo, CA, 1993.