Neural network techniques for earthquake detection in high noise

Kislov K.V. (kvkislov@yandex.ru), Gravirov V.V.

International Institute of Earthquake Prediction Theory and Mathematical Geophysics, Russian Academy of Science (IIEPT RAS)

Introduction

Many problems of seismology involve signal detection in the presence of high noise due to very diverse sources. This situation arises because of several factors:

- when monitoring in urbanized areas is concerned, one has to deal with high levels of manmade noise;
- the identification of events in noise permits one to consider lower magnitudes of events;
- reliable identification of signals in noise reduces the costs required to protect the seismic sensors from the environmental hazards [1].

Reliable signal detection is especially important in those early warning systems which are based on the principle of a single station [11]. When this is the case, the sensors are installed at end users, viz., at various facilities, in residential areas, in locations where diverse kinds of work are going on, that is, where the noise level is high. Environments like those just described lead to inadmissibly high errors of the first kind (false alarm). This imposes restrictions on the usability of the single station principle and makes us employ costly methods based on deployment of dense seismographic networks to cover extensive areas [7, 14].

One encounters as many difficulties in the detection of small earthquakes in the presence of high noise due to various sources. We have to deal with two problems: the discrimination between small earthquakes (magnitude ≤ 4) and large events that can cause damage (magnitude ≥ 5).

The standard detection methods are based on the assumption of stationary noise over sufficiently long segments of records. A seismogram recorded in a megacity contains kinds of noise with very different characteristics and origin, the noise level being comparable with the amplitude of the signal to be detected. At the same time, classical statistical theory requires only independent signal parameters to be included in the model. However, dependent parameters taken as a whole can also provide useful information. The number of dynamical parameters can be very large, and some of these can even be not regarded for inclusion in the model. This circumstance impedes, and occasionally totally precludes, the use of conventional methods.

The method of neural networks (NN) can prove to be an efficient tool for detecting events in noise [8].

In recent years the algorithms for data analysis based on NN techniques are increasingly becoming more popular. There are both applied studies using well-known algorithms for the purpose and the development of new approaches [4, 3, 16, 17, 10].

However, this powerful tool has as yet found a very limited application as a classifier in the detection of seismic signals recorded in noise [13].

The use of neural networks as classifiers possesses undoubted advantages.

- NNs are processing the incoming information with all neurons simultaneously, which makes it possible to carry out real-time analysis of complex signals using instrumental means.
- NNs can fit any continuous function. One is thus not forced to assume any hypotheses and, in a number of cases, to make hypotheses about which variables are really important.
- NNs examine how the input parameters affect the result. Key characteristics are identified during the training. Significant information can be extracted from redundant data.

- NNs can deal with problems using incomplete, distorted, noise-contaminated, and internally inconsistent data.
- NNs are invariant under change of dimension of the feature space.
- Trained NNs can be used even by untrained users.

We now list several problems facing a widespread use of NN technologies.

- NNs cannot solve all problems. There is no guarantee that a NN can be trained during a finite time. Frequently, the effort and machine time expenditures for training do not yield the desired results.
- Training can result in finding a local error minimum, and the best solution may prove unattainable.
- All kinds of NN training methods are subject to limitations.
- Classification criteria are hidden when neural networks are used. It is impossible to say how a problem is being solved and to interpret the solution in conventional analytical terms that are commonly used for developing the theory of the phenomenon being dealt with.
- An NN yields its result with a degree of probability attached, so that the networks are inapplicable to problems where absolute accuracy is required.

The NN architecture is chosen so as to be suitable for the particular application in hand. We chose the Neural Networks package in the MatLab system to simulate neural network structure [9]. The package includes about twenty neural networks and training rules which help select the most suitable architecture for the problem in hand. Classification problems are most frequently dealt with using networks of the following types: the multilayer perceptron (MLP), the radial basis network, the Kohonen network, and the probabilistic neural network. Different neural networks can be used, since their areas of application intersect. Nevertheless, the choice of the optimal network is a non-trivial problem. Frequently, many successive training steps result in an inadmissibly large error. In that case the network configuration has to be modified, or even a different network chosen.

Networks of the radial basis function type for use in classification problems divide the input data space by hyperspheres [15]. The response surface of a radial unit is the bell-shaped Gaussian function with its summit at the center and falloff toward the edges. Such networks possess several advantages. They have a single hidden layer, experience no difficulties with local minima, hence are trained very fast. However, we need more radial elements than in an MLP, we also need to specify their central positions and the deviations. The relevant algorithms are less suitable for finding the optimal solutions. It follows that the network will operate more slowly and will need more memory than for the corresponding MLP. In addition, networks of the radial basis function type poorly extrapolate data that lie far from the training sample.

The Kohonen neural network (the Kohonen self-organizing map) also deals with the classification of high-dimensional vectors. The network has the advantage over the other algorithms in ease of visualization and interpretation of the results. The network can be trained without a teacher, solely based on a sample of input data.

The advantage of the probabilistic NN consists in the fact that it is trained fast and that the output has a probabilistic meaning and is easier to interpret, but that kind of network requires extensive memory and may operate slowly. The radial units are copied directly from the training data, one per observation. Each is a Gaussian function centered at that observation. Each output unit is connected to all radial units that belong to its class, while to all the other radial units it has zero connections. Hence the output units simply add up the responses of all the units belonging to their own classes. The outputs are each proportional to the estimates of the probability density functions of the various classes, and by normalizing these we get the final probability estimates.
During the initial stage it is logical that we use the simplest configuration, modifying it to become more complex whenever required. Consequently, to deal with classification problems (earthquakes v. noise) we shall use a fully-connected NN with a feedforward structure, an MLP with a single hidden layer (see Fig. 1). The use of this architecture is frequently justified for dealing with many problems.

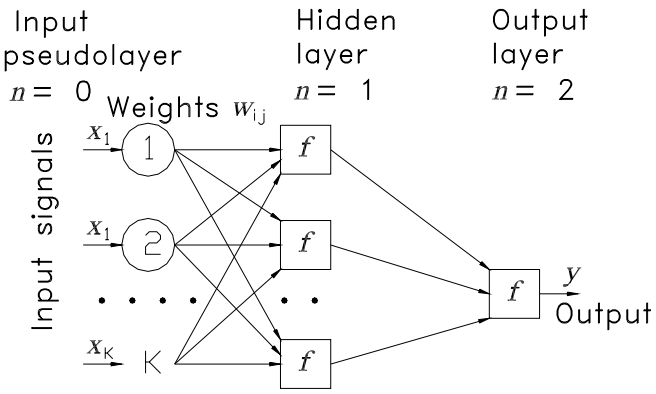


Fig. 1. Structure of a multilayer perceptron with a single hidden layer and a single neuron in the output layer.

MLP neurons are organized into several layers [6]. Each neuron receives and processes information coming from each neuron at a lower level. Signals are thus propagating in the network from the input layer via the hidden layers to the output layer. The input layer merely receives and transmits input information throughout the network. Each hidden layer neuron has one output. A neuron calculates a weighted sum of its inputs and transforms this in a nonlinear manner into an output signal. The activation function f (a nonlinear transformation) must be differentiable. This requirement is met by the sigmoid function we chose to use:

$$f(x) = \frac{1}{1+\exp(-x)}$$

Since the classification is to be into two classes (earthquakes and noise), the perceptron will contain a single output unit that describes the classification result y.

We now state the main requirements on a neural networks system for earthquake detection:

- compatibility between data storage formats and data processing tools;
- the possibility of normalizing and decimating the training data set;
- data classification by a trained NN;
- generation of an output signal that is convenient to use.

The data and methods to study

The structural diagram of the neural networks system for earthquake detection can be represented as follows (see Fig. 2).

The preprocessing module assembles and stores a set of training data. It converts seismic records stored in several formats to codes that are convenient to handle in the MatLab environment, synthesizes a single sampling rate for the sensors by data decimation, and corrects the data for the response of a particular sensor. Whenever required, the industrial frequency is to be filtered out. The next step is to form training records of a specified length. To do this, we count 1024 values from the beginning of the event, and the data are normalized and stored (see Fig. 3).

Data normalization is carried out by rescaling the values to the range [-1;1]. The data are then multiplexed, the initial point being shifted by two values, and the operation is repeated. In this way fifty training records are formed forward and backward of the beginning of the event. (The generation of such a data array for each event allows detection in pseudoreal time by successively shifting the window of analysis by one second on the seismic record.) Further, feature vectors are generated based on each training record with classification results corresponding to each. It is this normalized set of vector-result pairs which is the input material for NN training.

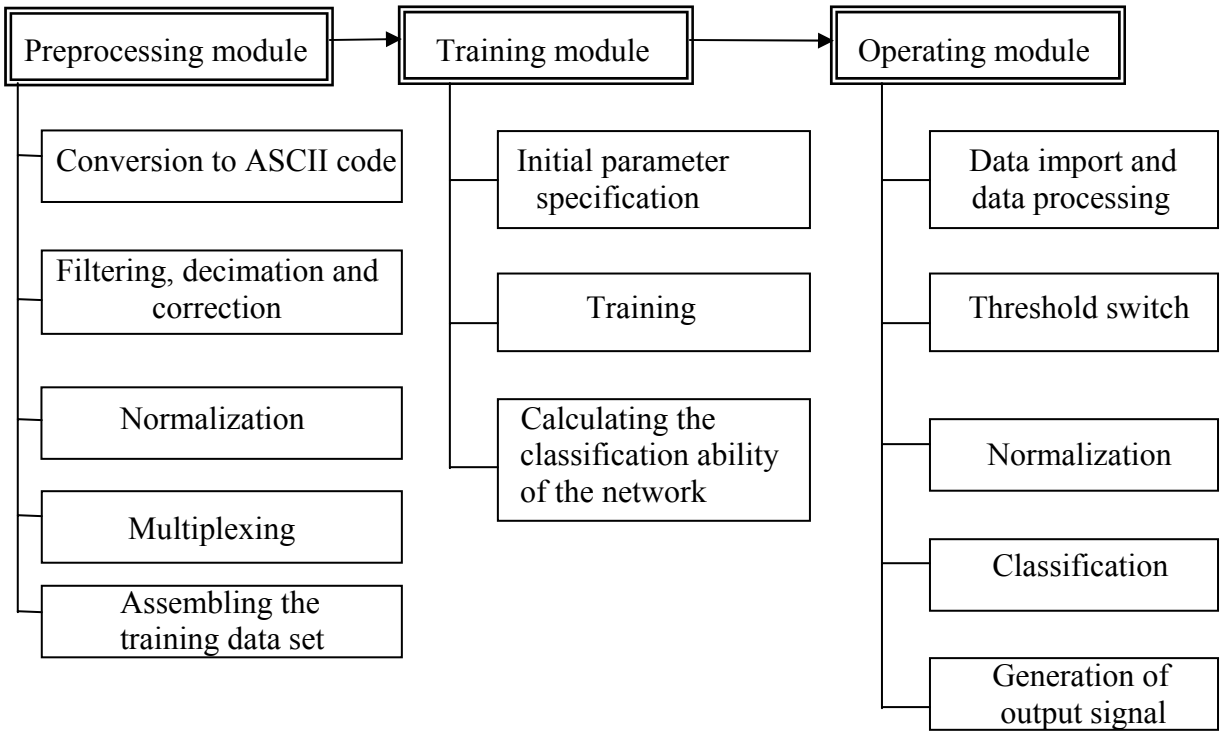


Fig. 2. Structural diagram of the neural networks system for earthquake detection

Originally, the feature vector is assigned intuitively. It is rather difficult to decide which parameters are the most significant. This can be explained, first, by the fact that earthquake signals are different, even signals due to one and the same event as recorded at different stations are rather different (see Figs. 3 and 4), and secondly, by noise diversity (see Figs. 5 and 6), especially manmade noise. One can proceed in a straightforward manner by submitting normalized amplitude values of the signal to the 1024 neurons in the input layer. To these may be added the spectrum values, the coefficients in the wavelet transform [5] or still more exotic parameters [11]. Hence the number of neurons in a network is a function of window length, the frequency interval under analysis, and the presence of extra parameters. Some program packages envisage automatic experimental selection of influential parameters.

This training module (see Fig. 2) is the main factor controlling the ability of the network to deal with problems of interest. All training algorithms are based on the same principle, namely, minimization of empirical error. Network configuration is gradually modified by modifying the states of synaptic weights so as to minimize the error.

The error function can be represented as follows. To each of the N weights there is a corresponding coordinate in a multidimensional space. The N+1-th measurement gives the network error. The network error can be visualized as a point in the (N+1)-dimensional space. The totality of these points forms the error surface. The goal of NN training consists in finding the lowest point on that surface.

It is impossible to determine the position of the global minimum on the error surface using analytical methods alone. Starting from a random point on the surface, the training algorithm gradually moves towards the minimum. To do this, one calculates the gradient of the error surface at a point and uses this information to move further down the slope. The algorithm stops training at the lowest point.

NN training uses methods that differ in efficiency and training speed: back propagation (BP), gradient descent with adaptive training, gradient descent with momentum and adaptive training, conjugate gradient method, Levenberg-Marquardt and others.

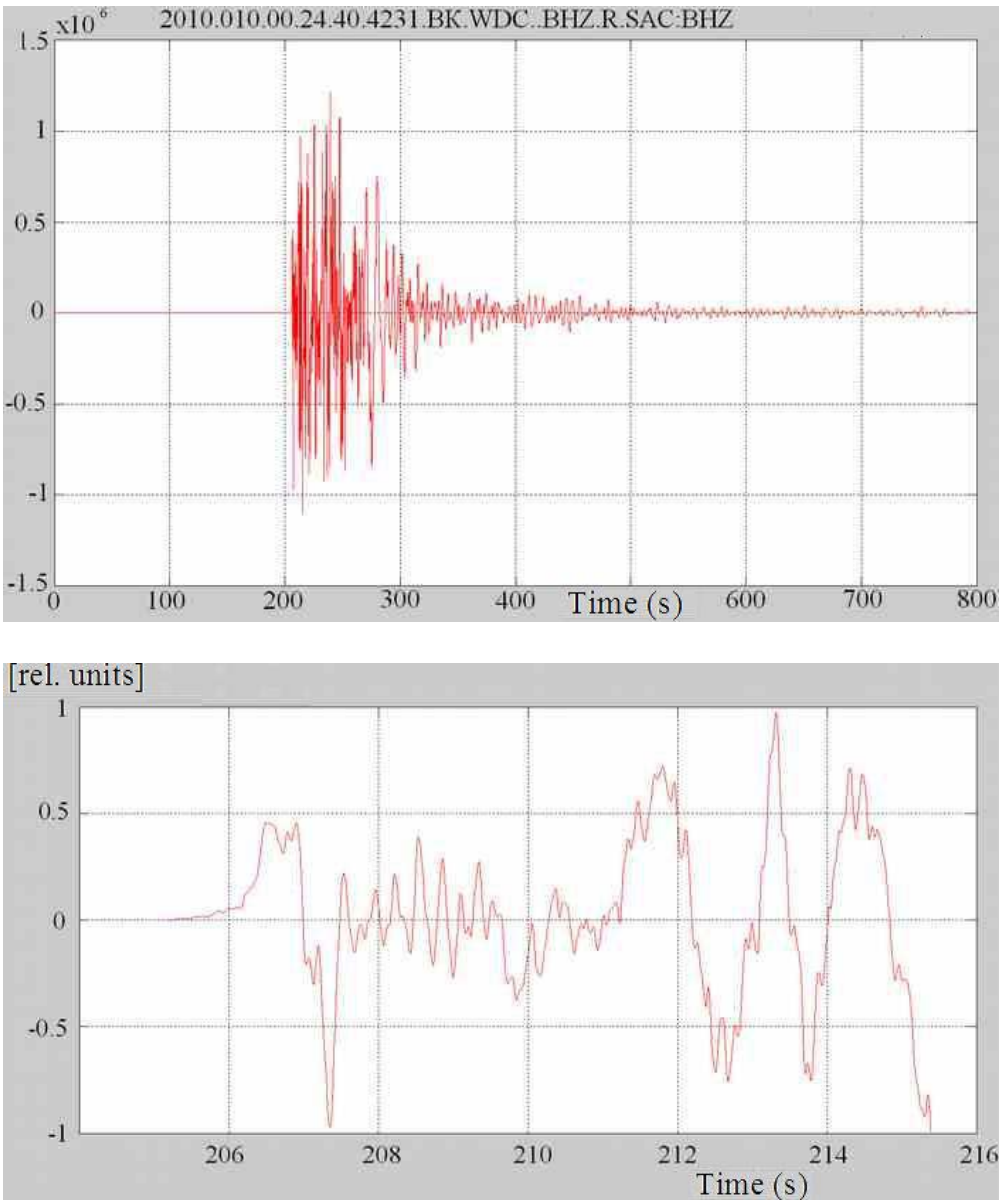


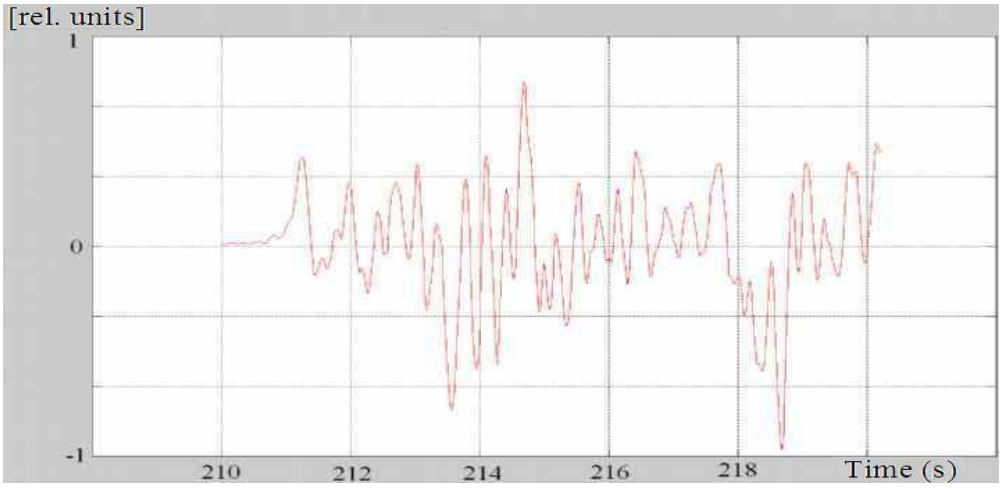
Fig. 3. Generation of the first training file (WDC station: latitude 40.58, longitude -122.54)
(a) original record of a standard event (magnitude 6.5) which occurred on January 10, 2010 at 00:27:39.3 off California coast (latitude 40.65, longitude -124.69);
(b) first 1024 values of the event normalized by rescaling into the range [-1;1] (the initial point corresponds to the time mark 205.4 s)

NN training uses methods that differ in efficiency and training speed: back propagation (BP), gradient descent with adaptive training, gradient descent with momentum and adaptive training, conjugate gradient method, Levenberg-Marquardt and others.

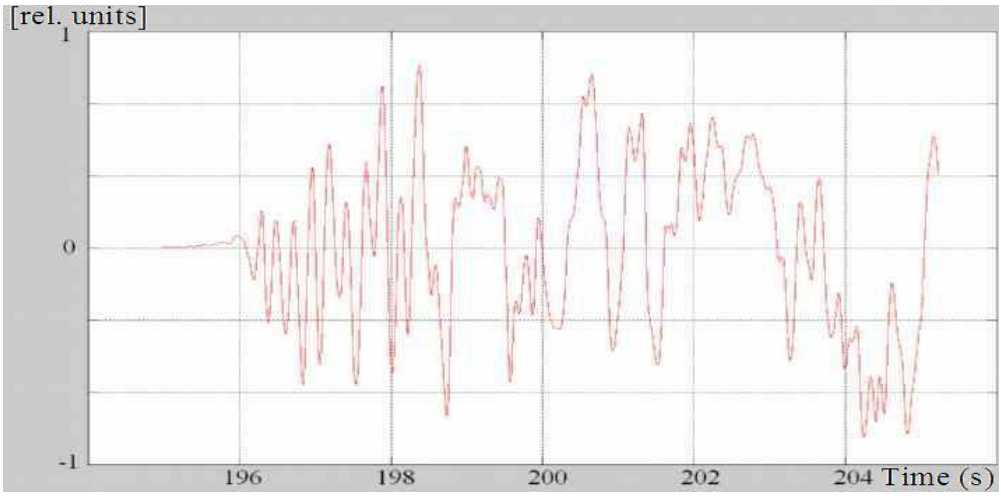
Among the training algorithms with a teacher (where we know the desired classification result for each data set) the most successful is BP. Training starts by submitting a feature vector and calculating the corresponding response. Comparison with the desired response guides the modification of the weights so as to make the network yield a more accurate result at the next step. The synaptic weights are modified using the local gradient in the error function. Information on network outputs is input for neurons in the preceding layers. The error function is the difference between the current network output and the ideal output. According to the method of least squares, the NN error function is

$$E(w) = \frac{1}{2}\sum_{j,p}(y_{jpn} - d_{jp})^2,$$

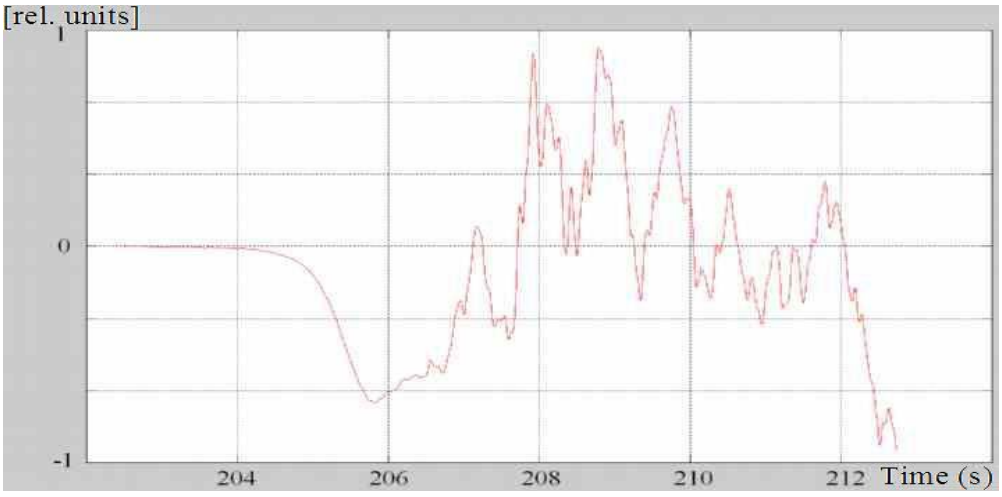
where y_{jpn} is the output state of the j-th neuron in the n-th layer as the p-th training image is submitted to the inputs; d_{jp} is the desired output state of that neuron.



a



b



c

Fig. 4. Comparing the first training files of the event with magnitude 6.5, January 10, 2010 at 00:27:39.3, latitude 40.65, longitude -124.69 recorded from different azimuths and at different epicentral distances:
(a) BEK station: latitude 39.87, longitude -120.36; epicentral distance Δ 360 km;
(b) M04C station: latitude 41.78, longitude -121.84; epicentral distance Δ 400 km;
(c) LCCR station: latitude 45.21, longitude -122.48; epicentral distance Δ 560 km

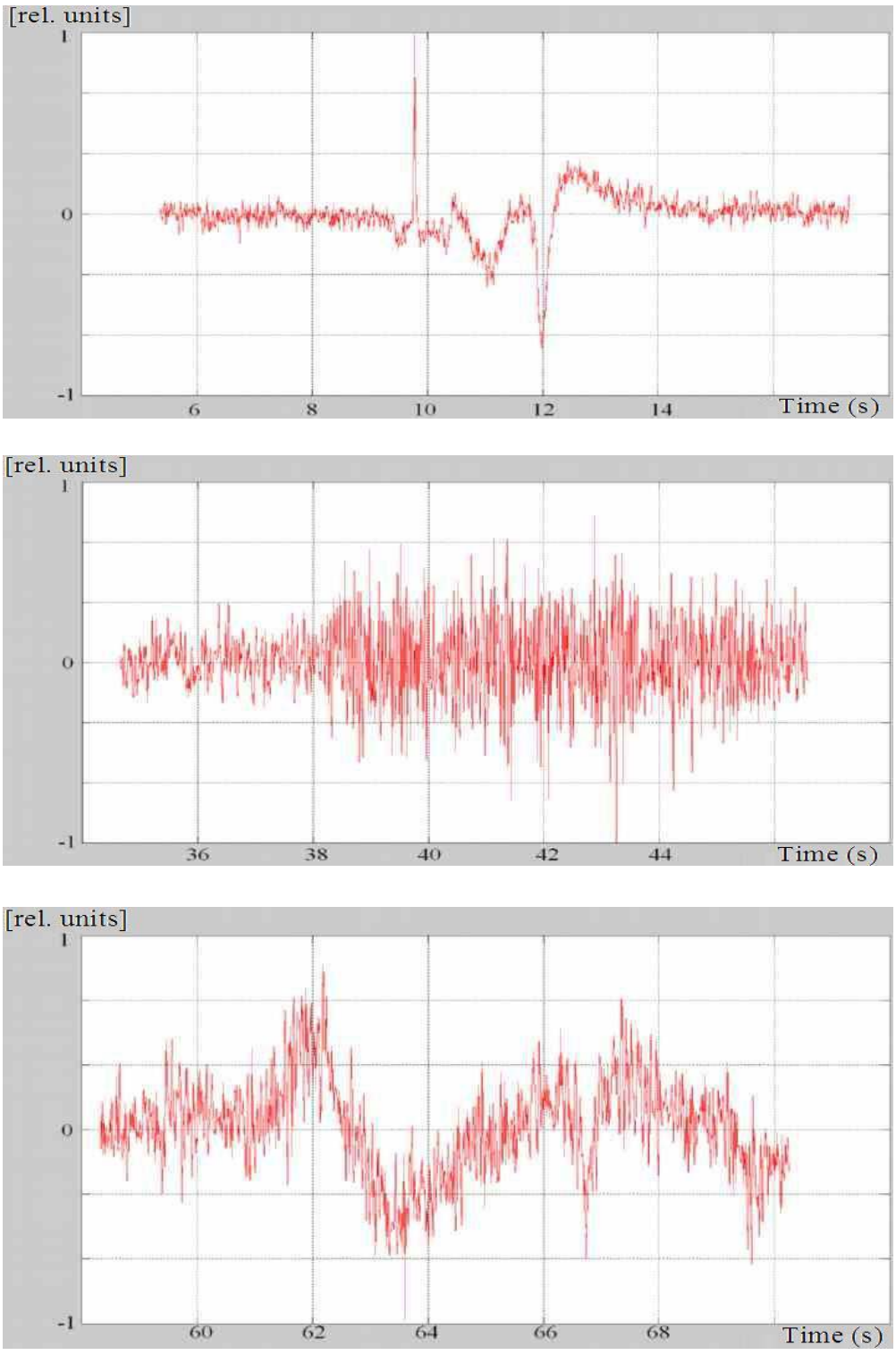


Fig. 5. Training files: noise due to diverse sources:
(a) movement of a heavy truck;
(b) electric railway;
(c) industrial noise

The network output should be made as close to the desired as possible. The synaptic coefficients are modified as follows:

$$w_{ij}(t+1) = w_{ij}(t) + \eta \frac{dE}{dw_{ij}},$$

where $w_{ij}(t)$ and $w_{ij}(t+1)$ are the weights of the connections between the i-th and j-th neurons at the current step and at the next training step (see Fig. 1), $\frac{dE}{dw_{ij}}$ is the derivative of the error function, and η the learning rate parameter. The greater is η, the less accurate will be the subsequent decrease in the total network error, but the smaller that parameter, the longer will be the training process and the chance of the network falling in a local minimum the greater. The simplest training rule is steepest descent, that is, modifying the synaptic weights in direct proportion to their contributions into total error. One further development of this algorithm is gradient descent with momentum. To the weight correction is added a value that is proportional to the change in that weight at the preceding step:

$$w_{ij}(t+1) = w_{ij}(t) + \eta \frac{dE}{dw_{ij}} + \alpha \cdot w_{ij}(t),$$

where α is the inertia coefficient.

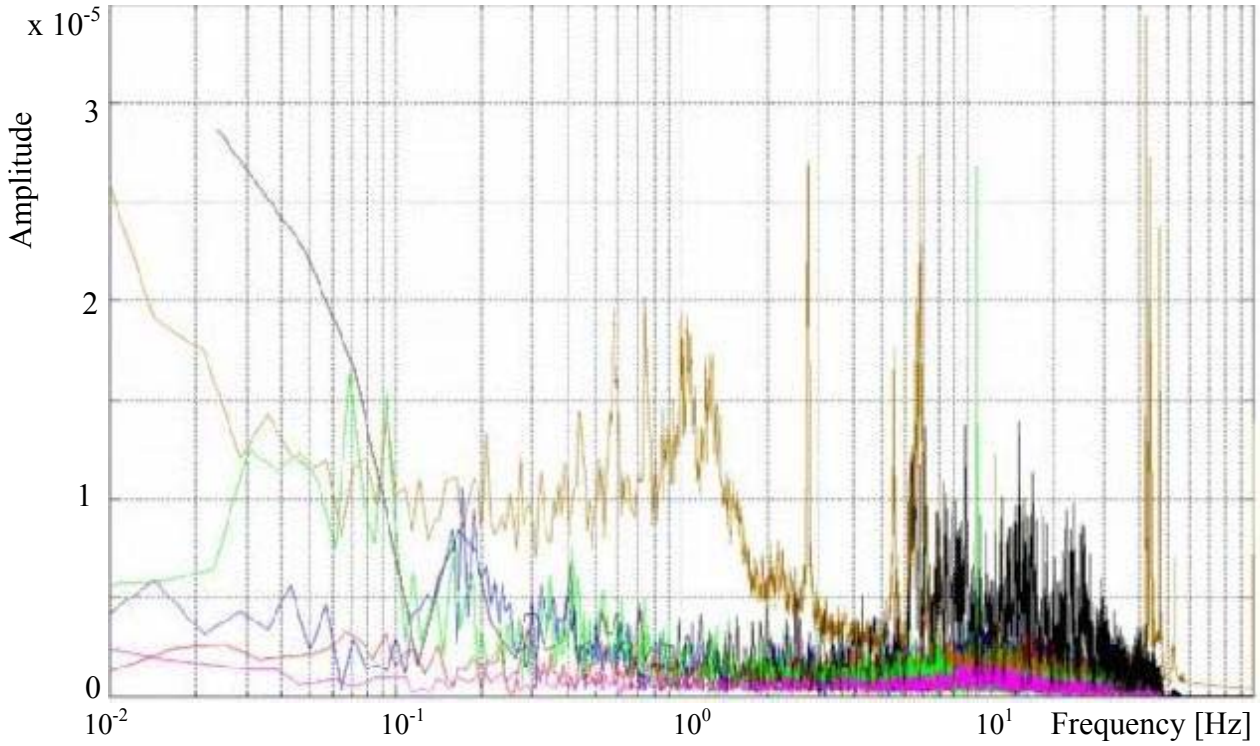


Fig. 6. Amplitude spectra of noise due to diverse sources; colors denote the following: blue stands for traffic, red for wind vibrations of a tall building, green for heavy trucks, brown for the sixth floor of an office building, violet for a vacant lot within the city limits, black for electric railway

BP is based on a method that can calculate the gradient of the network error function (the vector of partial derivatives) of many variables. The structure of the network error function is completely controlled by the NN architecture. BP can be very slow, especially with multilayered networks where the error surface is high-dimensional and involves many local minima or flat spots. In spite of the many successful applications of back propagation in complex problems, the network may happen to get no training at all or may get a false training. In the first place, the so-called "network paralysis" may occur, with corrections leading to very great weights, so that the neurons will operate in a region where the derivative of the squashing function is very small, and the training will practically stop. Secondly, the NN may happen to stick in a local minimum. Thirdly, "over-learning" may occur, with the training signals being classified very efficiently, while other signals, even ones very close to these, are incorrectly classified.

The gradient descent method with adaptive training does a parcel processing, which means that we calculate the gradient of the error surface averaged over the entire training set, and the weights are adjusted once at the end of each epoch (an epoch is running over all training cases with a check based on a test set). The training process is much faster; however, if the error surface is not concave, the algorithm may go in a wrong direction. As well, small changes in gradient may produce very large steps in the algorithm, even infinite ones, while when the gradient vanishes, the weights are no longer adjusted.

The basic idea of gradient descent with momentum and adaptive training consists in variable learning rates for the different weights. If the derivative has the same sign during several successive iterations, the learning rate is increased; conversely, when the sign changes, the rate is diminished. The algorithm may perform poorly on noise-contaminated error surfaces where the derivative may frequently change sign. In that case the derivative is smoothed. Nevertheless, the method in question is more sensitive to the effects of local minima than is BP.

The conjugate gradient method consists in the following procedure. One chooses the direction of steepest descent on a multidimensional surface and looks for a minimum (where the second derivative of the error surface is zero). The conjugate direction is chosen so as to maintain this zero derivative. In that case the minimum is reached after N epochs. On complex error surfaces the conjugacy deteriorates, but the algorithm requires less epochs than BP and also gives a better minimum.

Levenberg-Marquardt is used for single output networks, which is the case in our problem. The method is generally the fastest and the most reliable training algorithm, but requires a memory that is proportional to the number of weights in the network; this makes it impractical for reasonably big networks (on the order of thousands or more).

We have used the simplest method, BP. This is to be recommended when the data set is large and there are redundant data.

The initial adjustment of the weights significantly affects the number of training iterations BP requires. The synaptic coefficients should be selected at random, but in such a way that the activation function should be activated in a linear region, that is, the selected activation function and the initial values of the weights should match. The initial synaptic coefficients for the sigmoid activation function chosen for this study were taken at random from the normal distribution with zero mean and the standard deviation

$\sigma_{ц} = \sqrt{N}$ where N is the number of connections to a neuron.

The Neural Networks package envisages calculating the performance of a trained algorithm. The process can be visualized by a training curve, which shows the decreasing error versus the number of training epochs. The criterion is the error level obtained on the training data set, while the derivative of the training curve based on the test set must remain negative. Unfortunately, this calculation gives an estimate of how the network architecture fits the training data set and is not a measure to provide an unambiguous statement of the subsequent performance of a trained network.

Another stopping condition may be the training time (number of epochs).

Classification performance can only be calculated when the NN has been trained. To do this one uses a test set, which must also be representative enough.

The neural networks system for earthquake detection operates as follows (see Fig. 2, "Operating module"). The data are imported into the system straight from the seismometer or from any archive. Just as with the training data, these data are converted to ASCII, are corrected for the seismometer response, and a unified sampling rate for the seismometers is synthesized. A simple threshold switch is then used to specify the start of a segment for further analysis. An analysis window is formed, the data from a segment of definite length being converted, normalized, and transmitted to the inputs of the trained NN. Based on the classification results, an output signal is generated to alert about an earthquake or about the presence of high-level seismic noise.

Results

The training was carried out using data from California stations.

Twenty five events were processed with data coming from more than 20 stations. Since the data have been multiplexed, this gave more than 50,000 vectors consisting of characteristic features indicating the "earthquake" class. The processing of seismic noise records, both due to natural and manmade sources, gave a second data set (the "noise" class) for NN training. One hundred vectors from each class were reserved for the test data set.

The training continued until an error of 0.001 was reached on the training data set, but with no more than 50,000 epochs. The classification accuracy based on the test data set was 93%. Since this accuracy is high enough, though somewhat below the desired value, the NN here proposed can be used as the basic NN for developing a single sensor early warning system. Naturally enough, the neural network earthquake detection system as it now exists cannot be fully used. In the first place, the analysis of so long a time window (a little longer than 10 s) will inflate the dead zone of the system to as large a figure as 150-200 km (that zone is the area around the earthquake epicenter where the warning signal can only be generated after the damaging waves have arrived). Secondly, the network was trained on similar-slip earthquakes recorded at the same stations, hence is applicable to that region alone. Thirdly, the signal detection accuracy is not yet high enough, as has been mentioned above. Nevertheless, considering that improvements can be introduced into the NN and the methods can be used to generate the feature vectors, and considering the vast potential of neural networks methods for signal detection, one can be certain that the present method for this problem domain has promise.

Conclusion

This paper is no manual on NN applications to seismology and does not aspire to an exhaustive treatment of the topic. Many questions arising in network design have not been touched upon. To take an example, we did not give any recommendations as to the selection of the activation function and initial parameter adjustment, while this significantly affects the learning rate.

The principal goal of this paper was to study whether it is possible to use NNs for the detection of earthquake signals in noise due to diverse sources. We have examined the performance of the neural networks solution for this problem, discussed the different available NN types, the training algorithms for these, the selection of initial weights, and the methods applicable to estimation of NN performance.

Network design consists of the following steps:

1. Find the form and structure of input variables and assemble the training data set. Part of the data is reserved for the selection set and the test set.
2. Select several reasonable network configurations (the number of layers, connections between layers, the activation function, initial weights, etc.), the error function, i.e., a characteristic of the deviation of the network output signal from the desired output, and the criterion of network performance.
3. Find the selection error for each configuration from several training processes in order to escape a local minimum. If the desired accuracy has not been reached, the NN should be made more complex by adding neurons to the hidden layer or increasing the number of layers. In the case of over-learning, that is, when the selection error starts increasing, several neurons or layers should be removed.
4. Determine the network performance using the test data set.

The neural networks technologies can deal with more complex problems than mere signal detection. It is possible to address many applied, theoretical, and methodological problems. For example, the analysis of weights in a network can yield estimates of the relative influences of individual basic variables or types of response data representation (ranging from spectral and wavelet transforms to co-occurrence matrices and the characteristics based on these, viz.,

entropy, energy, contrast, homogeneity, etc.), that is, one can effectively perform an analysis of network sensitivity based on network inputs. The sensitivity of an input variable is found from network error: one calculates the increase in total prediction error for the case where the variable under analysis has been excluded from the input. The use of complex representations of input data with composite networks or sets of networks can not only help distinguish an earthquake signal from noise, but also perform the recognition of earthquake phases, more complex classification of earthquakes based on these characteristics, and so on. Since most of the computer time is expended on NN training, while little time is required for actual operation, it is possible to develop both software and hardware facilities operating in real time.

Acknowledgments

The work reported here is conducted at the International Institute of Earthquake Prediction Theory and Mathematical Geophysics, Russian Academy of Sciences (IIEP RAS) jointly with specialists at the Federal Unitary State Enterprise "Federal Research Institute of Aviation Systems" (FRIAS). Financial support comes from the European Community as represented by the International Science and Technology Center.

We are grateful to our foreign collaborators: Shigeki Horiushi, Senior Researcher at the National Research Institute for Earth Sciences and Disaster Prevention and Th. Camelbeek, Head of Seismology Department, Royal Observatory of Belgium for useful discussions and the data they lent us.

We also acknowledge useful consultations with Dr. P.G. Krug on neural network design.

Preliminary results from this study were reported at a conference held by the European Geophysical Union [12]. We thank all scientists who were interested in our work and took part in discussions of this.

References

1. Kislov, K.V., Theory and Methods for Protection of Broadband Seismometers from Ambient Factors (in Russian), Dissertation for Candidate of Phys.-Math. Sci., Moscow, 2009.
2. Medvedev, V.S. and Potemkin, V.G., Neural Networks (in Russian), MATLAB 6, Moscow: Dialog-MIFI, 2002, 496 pp.
3. Pustarnakova, Yu.A. and Akhmetova, E.R., Artificial neural networks as a tool for predicting geological parameters based on seismic attributes and logging data, in Geofizika, special issue "Seismic Prospecting Technologies" (in Russian) , 2002, pp. 117-121.
4. Senilov, M.A. and Lyalin, V.E., A neural networks model for pointwise interpretation of geophysical data, Vestnik Nizhegor. Un-ta, Mathematical Modeling and Optimal Control (in Russian), 2005, no. 1, pp. 215-223.
5. Tupitsyn, A.N., Signal Recognition and the Analysis of Nonstationary Point Processes Using the Wavelet Transform (in Russian), Extended Abstract, Diss. Cand. Phys.-Math. Sci., Saratov, 2009.
6. Haykin, S., Neural Networks: A Comprehensive Foundation. New York: Macmillan Publishing, 1994.
7. Allen R.M., The current status of earthquake early warning around the world, EOS Trans. AGU, 90(52), Fall Meet. Suppl., Abstract S21C-01, 2009.
8. Dayan P. and Abbott L. F. Theoretical Neuroscience, MIT, Berkeley, 2002, 432 pp.
9. Demuth H. and Beale M. Neural Network Toolbox. For Use with MATLAB, The MathWorks Inc., 1992-2000.
10. Gentili S. and Michelini A. Automatic picking of P and S phases using a neural tree, Journal of Seismology, vol. 10, no. 1, 2006, pp. 39-63.

11. Gravirov V.V. and Kislov K.V., The First Results of Testing Methods and Algorithms for Automatic Real Time Identification of Waveforms Introduction from Local Earthquakes in Increased Level of Man-induced Noises for the Purposes of Ultra-short-term Warning about an Occurred Earthquake, EOS Trans. AGU, 90(52), Fall Meet. Suppl., Abstract S13A-1731, 2009.
12. Gravirov V.V., Kislov K.V., and Ovchinnikova T.V. Neural network method for identification of earthquake phases in increased noise level conditions, EGU General Assembly 2010, Geophysical Research Abstracts, vol. 12, EGU2010-2434-1, 2010.
13. Gutierrez L.H.O., Morales J.L., Vargas-Jimenez C.A., and Niño L.F. Fast Seismic Event Classification Based On Magnitude-Distance Relation Based On Support Vector Machines Using Only One Three Component Station, EOS Trans. AGU, 90(52), Fall Meet. Suppl., Abstract S22A-03, 2009.
14. Kanamori H. Real-time seismology and earthquake damage mitigation, Ann. Rev. Earth Planet. Sci., 2005, vol. 33, pp. 195–214.
15. Suresh S., Sundararajan N., and Saratchandran P. A sequential multi-category classifier using radial basis function networks, Neurocomputing, vol. 71, nos. 7-9, pp. 1345-1358.
16. US Patent 5742740: Adaptive Network for Automated First Break Picking of Seismic Refraction Events and Method of Operating theSame, US Patent Issued on April 21, 1998.
17. Zhao Y. and Takano K. An artificial neural network approach for broadband seismic phase picking, Bulletin of the Seismological Society of America, 1999, vol. 89, no. 3, pp. 670-680.