# Universal Neural Network Demodulator for Software Defined Radio

Mohammad Reza Amini, Einollah Balarastaghi, and Boroujerd Branch

***Abstract*—In this paper a universal demodulator based on probabilistic neural network is presented. It is a kind of modulation (schemes) free demodulator, i.e. it can be trained for different schemes of digital modulation in order to detect incoming data bits without changing or replacing receiver hardware so it can be easily used in software defined radio structure and military communication. Furthermore it has some advantages over the other types of neural network demodulators such as fast training and fast data processing ability to detect data bits since there is no feedback layer in its structure. There is also no need to design special kind of filters except for those which are used to limit the input noise power.**

***Index Terms*—Demodulator, Probabilistic Neural Network, Simulation.**

## I. Introduction

Nowadays due to various advantages of digital systems, digital modulations are used to transmit data. In these kinds of modulations data symbols are mapped into the amplitude, phase or frequency of carrier signal which are called ASK,PSK and FSK respectively. Each of them has their own advantages and disadvantages. Generally digital demodulators are divided into two main categories: coherent and non-coherent that each of them requires its own hardware. if it is possible to design an efficient demodulator for almost every type of digital modulation schemes, a typical transmitter can send data with its desired modulation type regardless of receiver architecture, in this case there is no need to design a special receiver for particular transmitter because a universal demodulator has been achieved. Such a receiver would be widely used in software radios.

Recently neural network based demodulators have been proposed. They utilize dynamic network architectures to detect incoming symbols. These networks such as ELMAN and DTDNN2 are so time-consuming to get trained since feedback layer or time-delay lines are used in their own layers[1]–[2]. They also suffer from computational complexity therefore they are not suitable for real-time processing applications. Furthermore their performance is not good enough to detect data from different kinds of modulation schemes. Some of them are only suitable to get trained and detect data for special kind of modulation. In this paper an ANN3 demodulator based on probabilistic neural network and TDL4 placed in just first layer is presented to act as a universal demodulator. There is no complicated training process for this kind of demodulator except for setting input layer weights and defining different output classes, hence it can be trained quickly and can process data as soon as possible. In section II basic digital modulations are briefly introduced. In section III a brief overview about neural network and PNN5 are mentioned and applying PNN and defining output classes for the proposed method is discussed. In section IV two other neural networks are compared to the proposed method and in section V simulation and results are shown and finally in section VI some conclusions are given.

Manuscript received October 9, 2010; revised March 24, 2011.

[1] Distributed Time Delay Neural Network
[2] Distributed Time Delay Neural Network
[3] Artificial Neural Network
[4] Time-delay Line
[5] Probabilistic Neural Network

## II. Basic Digital Modulation Schemes

In this section, FSK demodulators is discussed and PSK and ASK demodulators will be left to reader. Normally, FSK signal is expressed by

$$z(t) = A\cos(\omega_c t + \omega_d \int^t D(\tau)d\tau) \qquad (1)$$

where $D(\tau)$ is a random binary pulse sequence with amplitude of +1 or -1 for binary bits 1 and 0 respectively in which each data bit is mapped into a special frequency related to its value. Pulse shaping filter might be used in modulator [3]. Usually, the demodulation method for FSK signal can be divided into two types: coherent demodulation (or so called matched filter) and non-coherent demodulation. They are illustrated in Fig.1.

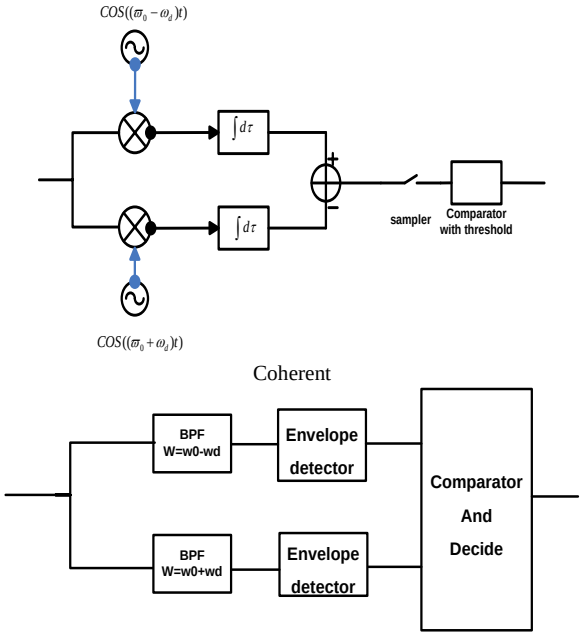



Fig. 1. Coherent and non-coherent FSK demodulator.

A coherent demodulator consists of two parallel paths that each of them computes the correlation of the received signal with symbols set produced by the transmitter. Non-coherent demodulator is also consists of two paths that each has an envelope detector and a band pass filter tuned to transmitter frequencies. PSK and ASK signals are also presented in the form of below:

$$z(t)_{PSK} = A\cos(\omega_c t + b_k \pi) \quad (2)$$

$$z(t)_{ASK} = b_k \cos(\omega_c t) \quad (3)$$

where $b_k$ represents the random binary data bit with the value of 0 and 1 for corresponding 0 and 1 data bits. Traditional demodulators of ASK and PSK are also classified to coherent and non-coherent [3]. They are not mentioned here.

## III. NEURAL NETWORK REVIEW

### A. Simple Neuron

A neuron with a single scalar input and no bias appears on the left side of Fig.2.

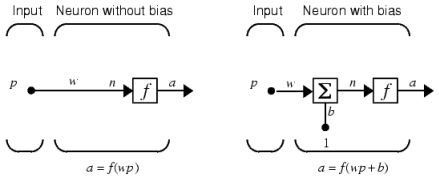



Fig.2 Neuron model

The scalar input p is transmitted through a connection that multiplies its strength by the scalar weight w to form the product wp, again a scalar. Here the weighted input wp is the only argument of the transfer function f, which produces the scalar output a. The neuron on the right has a scalar bias, b. You can view the bias as simply being added to the product wp as shown by the summing junction or as shifting the function f to the left by an amount b. The bias is much like a weight, except that it has a constant input of 1. The transfer function net input n, again a scalar, is the sum of the weighted input wp and the bias b. This sum is the argument of the transfer function f. (Radial Basis Networks, discusses a different way to form the net input n.) Here f is a transfer function, typically a step function or a sigmoid function, that takes the argument n and produces the output a. Examples of various transfer functions are in Transfer Functions. Note that w and b are both adjustable scalar parameters of the neuron. The central idea of neural networks is that such parameters can be adjusted so that the network exhibits some desired or interesting behavior. Thus, you can train the network to do a particular job by adjusting the weight or bias parameters, or perhaps the network itself will adjust these parameters to achieve some desired end.[4]–[5]–[6].

### B. Extended Neuron

A typical neuron can have multiple input called vector input. A network of neurons can consists of multiple layers in which there are multiple neurons as shown in Fig.3. [4]–[5]–[6]

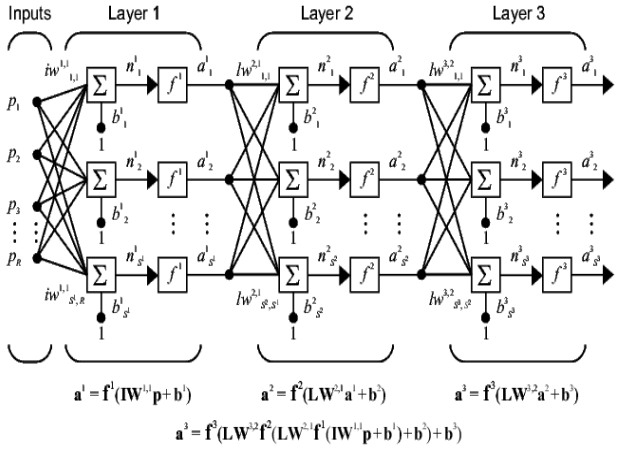



Fig.3. Typical network architecture

### C. Probabilistic neural network

Probabilistic neural networks can be used for classification problems. When an input is presented, the first layer computes distances from the input vector to the training input vectors and produces a vector whose elements indicate how close the input is to a training input. The second layer sums these contributions for each class of inputs to produce as its net output a vector of probabilities. Finally, a compete transfer function on the output of the second layer picks the maximum of these probabilities, and produces a 1 for that class and a 0 for the other classes. The architecture for this system is shown in Fig.4 [4]–[5]. It is assumed that there are Q input vector/target vector pairs. Each target vector has K elements. One of these elements is 1 and the rest are 0. Thus, each input vector is associated with one of K classes. Notice that the expression for the net input of a radbas neuron is different from that of other neurons. Here the net input to the radbas transfer function is the vector distance between its weight vector w and the input vector p, multiplied by the bias b. (The || dist || box in this figure accepts the input vector p and the single row input weight matrix, and produces the dot product of the two)[5].

The transfer function for a radial basis neuron is:

$$radbas(n) = e^{-n^2} \quad (4)$$

The radial basis function has a maximum of 1 when its input is 0. As the distance between w and p decreases, the output increases. Thus, a radial basis neuron acts as a detector that produces 1 whenever the input p is identical to its weight vector w. The bias b allows the sensitivity of the radbas neuron to be adjusted. For example, if a neuron had a bias of 0.1 it would output 0.5 for any input vector p at vector distance of 8.326 (0.8326/b) from its weight vector w. here is how ANN works: the first-layer input weights, IW1,1, are set to the transpose of the matrix formed from the Q training pairs, P'. When an input is presented, the || dist || box produces a vector whose elements indicate how close the input is to the vectors of the training set. These elements are multiplied, element by element, by the bias and sent to the radbas transfer function. An input vector close to a training vector is represented by a number close to 1 in the output vector a1. If an input is close to several training vectors of a single class, it is represented by several elements of a1 that are close to 1. The second-layer weights, LW1,2, are set to the matrix T of target vectors. Each vector has a 1 only in the row associated

with that particular class of input, and 0's elsewhere. The multiplication Ta1 sums the elements of a1 due to each of the K input classes. Finally, the second-layer transfer function, compete, produces a 1 corresponding to the largest element of n2, and 0's elsewhere. Thus, the network classifies the input vector into a specific K class because that class has the maximum probability of being correct [4]–[5].

### D. Setting input weights and defining output classes

As it can be inferred from previous section, proper defining output classes and setting weights make network to serve as a demodulator. To detect data bits efficiently with no error in noiseless environment, PNN[6] demodulator must firstly distinguish between different modulation schemes and secondly for each modulation type, it must recognize binary or M-ary symbols from each other ( e.g. bit 0 and bit 1). Thus there are P=M*K different symbols that demodulator has to recognize (M is the number of symbols for each modulation schemes and K is the number of modulation schemes, e.g. for binary transmission there are P=2*K symbols), hence P=M*K output classes should be defined for PNN demodulator and input weights vector should be also set for each output class. To distinguish between incoming symbols, there should be enough samples of each symbol in a bit duration time, then these samples should store in a shift register and should pass to the network as an input vector. To illustrate it, suppose there are N samples of incoming signal in a bit duration available at the receiver. Fig.5 shows how these signal samples are stored and fed into the network. Afterward network compares these samples with the ones which are set as weight vectors to determine which symbol are the most similar to input samples. There should be P weight vectors (one vector for each output class) and the symbol samples are stored to each vector as an input weight.

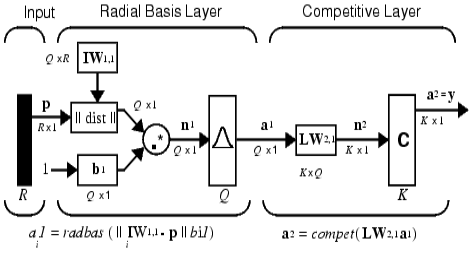



Figure 4. Probabilistic Network Architecture.

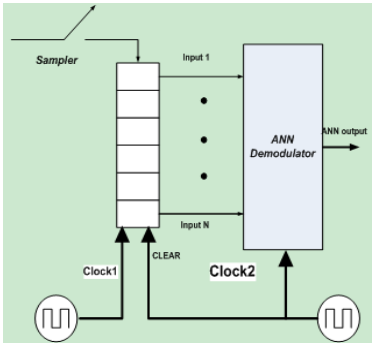



Fig.5. Feeding the ANN Demodulator

Note that the frequency of clk1 signal should be N times more than the frequency of clk2 signal.

[6] Probabilistic neural network

## IV. Comparing two Other Types of ANN

In this section two common types of ANN (ELMAN and TDNN Network) are introduced and compared to the proposed ANN. Elman network commonly is a two-layer network with feedback from the first-layer output to the first-layer input. This recurrent connection allows the Elman network to both detect and generate time-varying patterns. A two-layer Elman network is shown in Fig. 6 [1]–[7]–[8].The Elman network has tansig neurons in its hidden (recurrent) layer, and purelin neurons in its output layer. This combination is special so that two-layer networks with these transfer functions can approximate any function (with a finite number of discontinuities) with arbitrary accuracy. The only requirement is that the hidden layer must have enough neurons.

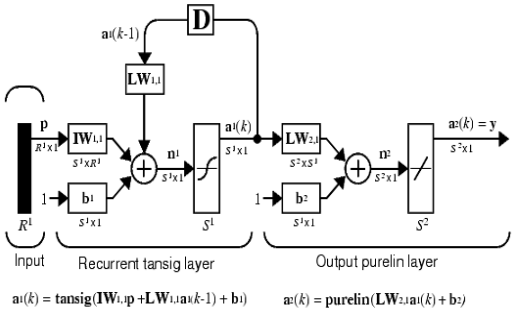



Fig. 6. Elman Network Architecture

More hidden neurons are needed as the function being fitted increases in complexity. Elman networks are not as reliable as some other kinds of networks, because both training and adaptation use an approximation of the error gradient. For an Elman to have the best chance at learning a problem, it needs more hidden neurons in its hidden layer than are actually required for a solution by another method. While a solution might be available with fewer neurons, the Elman network is less able to find the most appropriate weights for hidden neurons because the error gradient is approximated. Therefore, having a fair number of neurons to begin with makes it more likely that the hidden neurons will start out dividing up the input space in useful ways, so in real time applications and adaptive trainings it is not suitable to use it, at least because of training time and memory limitations and also training algorithm convergence (hardware resources) as indicated in [1]–[5]–[7]. A simple way to store data samples and use them to demodulate a symbol is to apply a tapped-delay line or time-delayed line (TDL) at the beginning of each layer. This is called the Distributed Time-Delay Neural Network (TDNN). The original architecture was very specialized for the particular problem. Fig. 7 shows a general two-layer distributed TDNN. This network is well suited to time-series prediction in which $f^i$ is the activation (transfer) function of layer i, IW and LW are the input weights and layer weights vector respectively and bi is the $i^{th}$ layer bias vector[1]–[5]–[8]–[9]. None of these introduced neural networks can be trained properly as a universal demodulator while they may be trained somewhere as a single-aimed demodulator ( but not universal ). To make sure in this section it is tried to train and test them to compare with the PNN type. The SIMULINK toolbox of MATLAB is used to build the

corresponding models of ANN demodulators, whose transfer function of the hidden layer is tansig and transfer function of the output layer is purelin with 8 neurons in the hidden layer. A back propagation algorithm is presented to train the ANN demodulator. For a TDNN network to simulate, time-delay lines with 4 and 3 elements are considered in hidden and output layer respectively. Table I and II show the structural specifications of the two mentioned ANN (ELMAN and TDNN) in MATLAB simulation. As it can be seen from the figure 8and 9, for both TDNN and ELMAN there is no hopeful learning curve which can show a good performance (here sum squared error is chosen for the performance function) and it means that these two networks cannot get trained satisfactorily

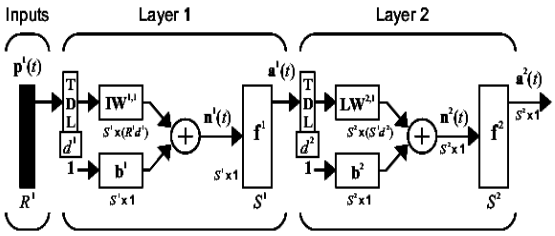



Fig. 7. TDNN Network Architecture

hence the two ANN outputs would not be as it is expected, i.e. the two ANN cannot demodulated the incoming signal perfectly. Figure 10 shows the outputs of TDNN and ELMAN along with the ideal output. It can be easily figured out that in some circumstances the two ANN demodulators detect data bits with error (most of these errors is for the PSK modulated signals because both of demodulators have some problems training of PSK modulated signals.

TABLE I. ELMAN NETWORK PARAMETERS.

```
ELMAN_NET =
  Neural Network object:
    numInputs: 1
    numLayers: 2
   biasConnect: [1; 1]
   inputConnect: [1; 0]
   layerConnect: [1 0; 1 0]
   outputConnect: [0 1]
Delay:
  numOutputs: 1 (read-only)
  numInputDelays: 0 (read-only)
  numLayerDelays: 1 (read-only)
  functions:
     adaptFcn: 'trains'
     divideFcn: 'dividerand'
    gradientFcn: 'calcjxfp'
      initFcn: 'initlay'
    performFcn: 'mse'
     plotFcns: {'plotperform','plottrainstate'}
     trainFcn: 'traingdx'
```

TABLE II. TDNN network parameters.

```
TDNN NET =
  Neural Network object:
    numInputs: 1
    numLayers: 2
   biasConnect: [1; 1]
   inputConnect: [1; 0]
   layerConnect: [0 0; 1 0]
   outputConnect: [0 1]
Delay:
  numOutputs: 1 (read-only)
  numInputDelays: 4 (read-only)
  numLayerDelays: 3 (read-only)
  functions:
     adaptFcn: 'trains'
     divideFcn: 'dividerand'
    gradientFcn: 'calcjxfp'
      initFcn: 'initlay'
    performFcn: 'mse'
     plotFcns: {'plotperform','plottrainstate'}
     trainFcn: 'trainlm'
```

## V. SIMULATION AND RESULTS

Simulation process includes constructing a proposed neural network architecture (setting input weights vectors and defining output classes), generating binary data randomly modulated by PSK, FSK, ASK schemes, and finally passing these modulated data into proposed universal demodulator. Effect of AWGN channel is later added to the simulation. Table III shows modulated signals properties. Note that sampling frequency is at least 10 times more than maximum carrier frequency.

Since there are 11 types of signal involved in the simulation (for each a unique ANN output class is defined), there should be an input weights matrix with a size of 6*N ( N is the number of signal samples in a bit duration). Figure 11 shows the signal samples set as an input weights vector.

After setting weights vector and defining output classes of ANN, binary data should be generated and modulated randomly at the transmitter. To verify the demodulator performance, these modulated data should be passed through

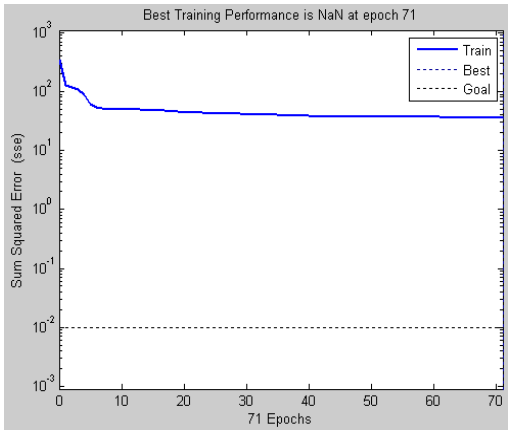



Fig.8 . Learning curve of TDNN ANN

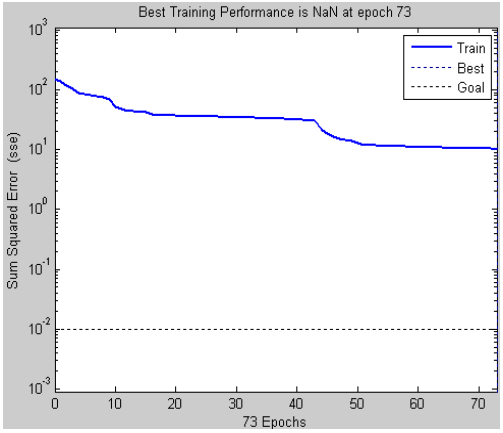



Fig. 9. Learning curve of ELMAN ANN

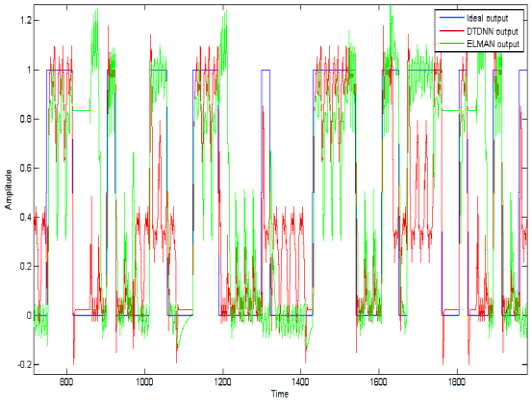



Fig.10 .Output Elman ANN (green) DTDNN (red) compared to ideal output (Blue)

the ANN demodulator and then the output should compare to the original data. No error occurs at the receiver after running a simulation. Figure 12 and 13 show the mentioned process.

To simulate the effect of noise, white Gaussian noise is added to signal after modulating at the transmitter. The resulting noisy noisy signal is then passed through the ANN demodulator and the effect of noise power on bit error rate is shown in figure 14.

TABLE III . MODULATED SIGNALS PROPERTIES

| modulation Type | BFSK | BPSK | BASK |
|---|---|---|---|
| Sampling Frequency (sample/sec) | 4500K | 4500K | 4500K |
| Carrier Frequency (KHz) | 450 | 300 | 100 |
| Bit Rate (bit/sec) | 90K | 90K | 90K |
| Corresponding ANN output class | 5 for bit 0 | 1 for bit 0 | 3 for bit 0 |
| | 6 for bit 1 | 2 for bit 1 | 4 for bit 1 |

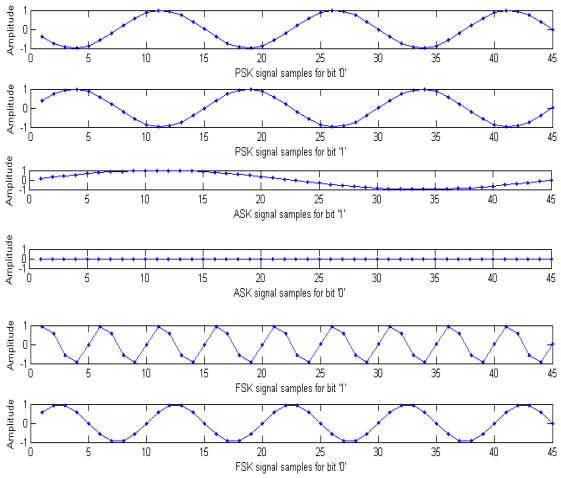
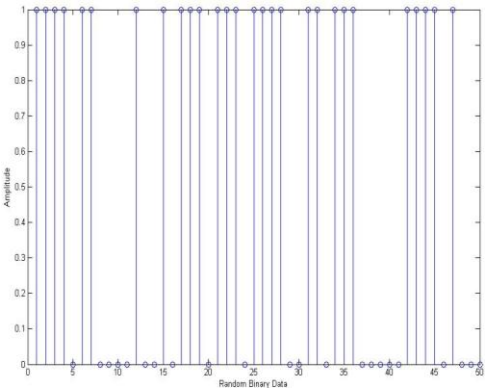
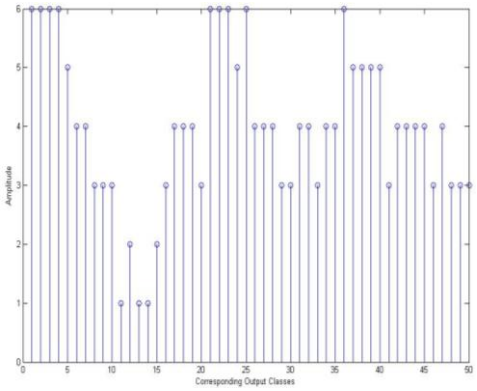



Fig. 11. samples set as a weights vector.

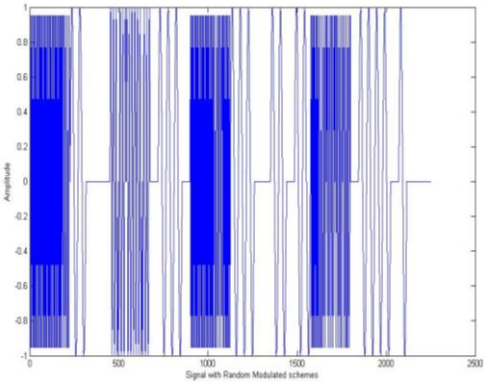



12.a-binary data

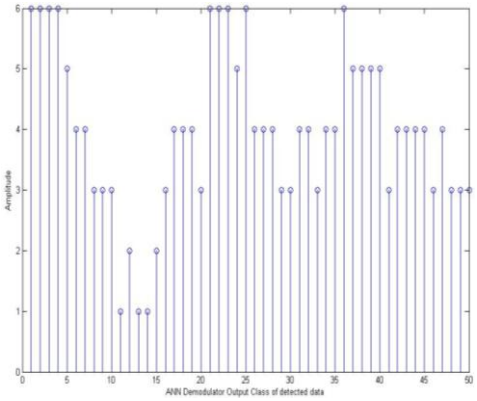



12.b-Corresponding ANN output class

Fig. 12 Simulation process signals




13.a- Randomly modulated signals




13.b- Detected demodulator output classes

Fig.13 . Simulation process signals

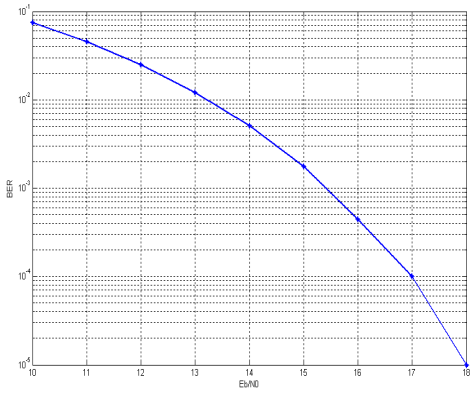



Fig. 14. BER curve versus Eb/N0

## VI. Conclusion

In this paper a universal demodulator based on probabilistic neural network was presented and its performance was shown. In such a demodulator input weight vectors and output classes should be defined for different modulation schemes in order to detect incoming signals without changing or replacing receiver hardware so it can be easily used in software defined radio structure. Furthermore it has some advantages over the other types of neural network demodulators such as fast training and fast data processing ability to detect data bits since there is no feedback layer in its structure. Finally the effect of noise in detecting data bits was shown.

## References

[1] Min Li HongSheng Zhong ; Min Li. "Neural Network Demodulator for Frequency Shift Keying," 2008 International Conference on Computer Science and Software Engineering, 978-0-7695-3336-0/08 $25.00 © 2008 IEEE,(DOI 10.1109/CSSE.2008.1440)

[2] Ohnishi, K.; Nakayama, K.,"A neural demodulator for quadrature amplitude modulation signals," Neural Networks, 1996., IEEE International Conference on Volume 4, 3-6 June 1996

[3] Shanmugam,k.sam. "digital and analog communication systems", John Wiley & Sons,1979

[4] Caudill, M., Neural Networks Primer, San Francisco, CA: Miller Freeman Publications, 1989.

[5] Matlab software documentations for neural networks Release 2009

[6] L. Fausett, Fundamentals of Neural Network: Architecture, Algorithms, and Applications, Prentice Hall, Upper Saddle River, NJ (1994).

[7] "DARPA Neural Network Study", Lexington, MA: M.I.T. Lincoln Laboratory, 1988.

[8] Nakayama, K.; Imai, K.,"A neural demodulator for amplitude shift keying signals," Neural Networks, 1994, IEEE International Conference on Volume 6, 27 June-2 July 1994

[9] Chesmore, E.D., "Neural network architectures for signal detection and demodulation," Radio Receivers and Associated Systems, 1989, Fifth International Conference on 23-27 Jul 1990 Page(s):1-4.

**Mohammad Reza Amini** was born at 1981. He is an Iranian telecommunication researcher. He is an academic member of Islamic Azad University, Borujerd branch, Iran. iman.amini@gmail.com.

**Einollah Balarastaghi** is an academic member of Islamic Azad University, Borujerd branch. Iran.meh_balarastaghi@yahoo.com.