Neural net decoders of binary codes

V V Butov and V N Dumachev
Department of Mathematics, Voronezh Institute of the Ministry of Internal Affairs of Russia, pr. Patriotov 53, Voronezh, Russia
E-mail: butovvladislav@yandex.ru, dumv@comch.ru

Abstract. In the article, neural net decoders for binary error-correcting codes are constructed. Analytical methods for calculating of synapse weights are proposed. Algorithms of neural net coding and decoding are considered on example Viterbi convolutional code, LDPC and Reed-Muller code.

1. Introduction
For the first time, the idea of using neural network computing [1] to build error correction codes [2] was proposed in the 90s of the XX century [3–5]. In these papers, codes of small dimensions \((n; k)\) were considered and synaptic weight of neural net decoders was calculated analytically. At the same time, it was immediately noticed that an increase in the size of the information word \(k\) leads to an increase in the dimension of the neural network in proportion to \(2^k\). This fact stopped the development of this direction for ten years. In 2002, J.Wu, Y.Theng and Y.Huang [6] proposed their own version of a 2-layer BCH decoder. To adjust synaptic weights of the code \((n; k)\), they were solved by a system of \(k\) nonlinear equations in \(n\) variables using genetic algorithms. Moreover, the degree of equations in some cases reached \(n\). Computer capacities allowed them to train decoders up to \((n; k) = (21, 6)\) inclusive. Further studies were again stopped by the nonpolynomial dependence of the neural net dimension on the codeword length. The explosive increase in computer performance has recently allowed researchers to draw an attention to this task again. So, in the works [7–10] the soft decoders BCH(63,45), BCH(127,106) — BCH(127,64) are analyzed. To construct such decoders, neural net deep learning is used, the number of layers in which reaches 1200 [9].

2. Neural net encoder
Each layer of the direct propagation neural net is described by equation

\[ y = f(Wx + b). \]

Here \(x\) is the input and \(y\) is the output vector of the layer, \(W\) is a matrix of weight coefficients, \(b\) is a bias, \(f(x)\) is an activation function.

Let \(a = (a_1a_2...a_k)\) be an information word. If \(G\) is the generator matrix of the code, then the codeword \(c = Ga\) is formed by a single-layer feedforward neural net with \(W = G, b = 0\) and the activation function as \(f(z) = z \mod 2\) (figure 1).
Example 1. Consider the convolutional code for informational sequences $a = (010011)$. Since

$$G = \begin{pmatrix} 11 & 10 & 11 \\ 11 & 10 & 11 \\ 11 & 10 & 11 \\ 11 & 10 & 11 \\ \cdots \end{pmatrix} = W,$$

we have

$$y = f(Wa) = (001110111101) = c.$$

3. Neural net Boolean decoder

One of the simplest neural net decoder can be constructed on the base of Boolean functions. For code $(n, k)$, the input sequence is a Boolean $n$-dimensional vector. The output sequence is a Boolean $k$-dimensional vector. Therefore, a Boolean decoder is a set of $k$ Boolean functions of $n$ variables $(\varphi_1, \varphi_2, \ldots, \varphi_k)$. It is known that the number of all $k$-ary Boolean functions is equal to $2^k$. We are interested only in those in which the number of 1 is equal to the number of 0. The simplest way to construct all necessary functions is to present them in a disjunctive normal form (DNF). After that, the task of neural net design of coders is reduced to the task of a neural net realization of DNF.

Recall that if a boolean function of $n$ variables has a unit in the truth table on $(i_1, i_2, \ldots, i_m)$ places, then for DNF we have

$$\varphi(i_1, i_2, \ldots, i_m) = \bigvee_{p=1}^{m} (-\bar{T}_{n-1}x_{n-1} \& (-\bar{T}_{n-2}x_{n-2} \& \cdots \& (-\bar{T}_1x_1 \& (-\bar{T}_0x_0).$$

Here, $\bar{T}_j = 1 - I_j$, $(I_{n-1}, I_{n-2}, \ldots, I_1, I_0)$ is a vector of binary representation of the number $i_p$, i.e.

$$i_p = \sum_{j=0}^{n-1} I_p 2^j.$$

For any full DNF of boolean function $\varphi(i_1, i_2, \ldots, i_m)$, we can construct a two-layer neural network. The first layer calculates all conjunctions. For each conjunction

$$(-\bar{T}_{n-1}x_{n-1} \& (-\bar{T}_{n-2}x_{n-2} \& \cdots \& (-\bar{T}_1x_1 \& (-\bar{T}_0x_0,$$
we get a perceptron with a separating hyperplane

$$-(\sum_{j=0}^{n-1} I_{j} p_{j} n_{j}) + \sum_{j=0}^{n-1} I_{j} x_{j} n_{j} = n - \sum_{j=0}^{n-1} f_{j}.$$  

For each disjunction

$$x_{1} \lor x_{2} \lor \ldots \lor x_{m}$$

we get a perceptron with a separating hyperplane

$$\sum_{p=1}^{m} x_{p} = \frac{1}{2}.$$ 

For both layers, we will use the Heaviside activation function $f(z) = \theta(z)$, $\theta(0) = 1$.

**Example 2.** Consider the Hamming code $(n, k) = (5, 2)$ with the generator matrix $G$. This code converts four possible informational words $A = (a_{1} a_{2} a_{3} a_{4}) = \begin{pmatrix} 0011 \\ 0101 \end{pmatrix}$ into codewords

$$M = GA = \begin{pmatrix} 0011 \\ 0101 \\ 0110 \\ 0011 \\ 0101 \end{pmatrix}. $$

This code allows to correct only one error, so we will write out all valid and distorted code words that can be decoded

$$\overline{M} = \begin{pmatrix} 010000 \\ 001000 \\ 101111 \\ 101111 \\ 001000 \\ 110111 \\ 001000 \\ 110111 \\ 000100 \\ 111011 \\ 111011 \\ 000100 \\ 000010 \\ 111101 \\ 111101 \\ 000010 \\ 111110 \end{pmatrix} = (\overline{M}_{1} \overline{M}_{2} \overline{M}_{3} \overline{M}_{4}). $$

For any input signal (column), the output is the information word. That is, in our neural network the input matrix $\overline{M}$ will transformed into an output matrix of corrected information words

$$\overline{A} = \begin{pmatrix} 000000 \\ 000000 \\ 111111 \\ 111111 \\ 000000 \\ 111111 \\ 000000 \\ 111111 \end{pmatrix}. $$

Consider the first row of the matrix $\overline{A}$. This row corresponds to the boolean function of 5 variables

$$\varphi_{1}(22, 6, 30, 18, 20, 23, 27, 11, 19, 31, 25, 26) = x_{5} \overline{x}_{4} x_{3} x_{2} x_{1} \lor x_{5} x_{4} x_{3} x_{2} x_{1} \lor x_{5} x_{4} x_{3} x_{2} x_{1}$$

$$\lor x_{5} x_{4} x_{3} x_{2} x_{1} \lor x_{5} x_{4} x_{3} x_{2} x_{1} \lor x_{5} x_{4} x_{3} x_{2} x_{1} \lor x_{5} x_{4} x_{3} x_{2} x_{1} \lor x_{5} x_{4} x_{3} x_{2} x_{1} \lor x_{5} x_{4} x_{3} x_{2} x_{1}.$$ 

The first layer realize all conjunctions

$$y^{1} = \theta(W_{1}^{1} \overline{M} + b^{1}),$$
where 
\[
W_1^i = 2( MM_1) - J, \quad b^1 = -(324224 433533)^T.
\]
The second layer realize all disjunctions
\[
y^2 = \theta \left( W^2 y^1 + b^2 \right),
\]
where
\[
W^2 = J, \quad b^2 = -1/2.
\]
Consider the second row of the matrix \( A \). This row corresponds to the boolean function
\[
\varphi_2(13, 29, 5, 9, 15, 12, 27, 11, 19, 31, 25, 26) = \mathcal{I}_{5x4x3x2x1} \lor x_{5x4x3x2x1} \lor \mathcal{I}_{5x4x3x2x1} \lor x_{5x4x3x2x1} \lor \mathcal{I}_{5x4x3x2x1} \lor x_{5x4x3x2x1} \lor \mathcal{I}_{5x4x3x2x1} \lor x_{5x4x3x2x1} \lor \mathcal{I}_{5x4x3x2x1} \lor x_{5x4x3x2x1} \lor \mathcal{I}_{5x4x3x2x1} \lor x_{5x4x3x2x1} \lor \mathcal{I}_{5x4x3x2x1} \lor x_{5x4x3x2x1} \lor \mathcal{I}_{5x4x3x2x1} \lor x_{5x4x3x2x1}.
\]
Then for first layer we have
\[
W_2^i = 2( MM_1) - J, \quad b^1 = -(324224 433533)^T.
\]
All coefficients of the second layer for \( \varphi_2 \) coincide with coefficients for \( \varphi_1 \).
Let \( a = (10) \) is information word. Suppose that during transmission of codeword \( m = (101110) \), an error occurred in the second symbol: \( m_2 = (111110) \). The output of the neural net for the first bit is
\[
y_1^1 = \theta \left( W_2^1 m + b^1 \right) = (001000 000000), \quad y_2^2 = \theta \left( W_2^2 y_1^1 + b^2 \right) = 1.
\]
The output of the neural net for the second bit is
\[
y_1^1 = \theta \left( W_2^1 m + b^1 \right) = (000000 000000), \quad y_2^2 = \theta \left( W_2^2 y_1^1 + b^2 \right) = 0.
\]
Since the weights of the second layer are the same for all information symbols, they are combined in the figure 2.
The obvious direction of optimizing for the Boolean decoder is the use of alternative basis.
This is due to the fact that a small formula on one basis can have an exponential size explosion on another basis, and vice versa. For example, for code \( x = (x_4x_3x_2x_1x_0) \) with information bits \( x_4x_3 \) we have the Zhegalkin decoder in the form
\[
x_4' = x_4x_2 \oplus x_4x_3 \oplus x_4x_0 \oplus x_2x_1 \oplus x_1x_0,
x_3' = x_3x_2 \oplus x_3x_1 \oplus x_3x_0 \oplus x_2x_0 \oplus x_1x_0.
\]
For reference, we present the Zhegalkin decoder for Hamming code (6.3) with \( x = (x_5x_4x_3x_2x_1x_0): \)
\[
x_5' = x_5x_1 \oplus x_5x_3 \oplus x_5x_0 \oplus x_3x_2 \oplus x_2x_1, 
x_4' = x_4x_1 \oplus x_4x_3 \oplus x_4x_0 \oplus x_3x_2 \oplus x_2x_1, 
x_3' = x_3x_1 \oplus x_3x_3 \oplus x_3x_0 \oplus x_3x_1 \oplus x_3x_0 \oplus x_1x_0.
\]
In this case, for neural net decoder, we have
\[
W_1^1 = \begin{pmatrix} 110000 \\ 101000 \\ 100100 \\ 100010 \\ 100001 \\ 011000 \\ 010010 \\ 001100 \\ 000110 \end{pmatrix}, \quad W_2^2 = \begin{pmatrix} 110000 \\ 101000 \\ 100010 \\ 100001 \\ 011000 \\ 010010 \\ 001100 \\ 000110 \end{pmatrix}, \quad W_3^1 = \begin{pmatrix} 110000 \\ 101000 \\ 100010 \\ 100001 \\ 011000 \\ 010010 \\ 001100 \\ 000110 \end{pmatrix}, \quad W^2 = J, \quad b^1 = -2J, \quad b^2 = 0, \quad f^1(z) = \theta(z), \quad \theta(0) = 1, \quad f^2(z) = z \mod 2.
4. Neural net G-decoder

If an error occurs, the valid codeword transforms into corrupt. All forbidden words that formed
from permitted using the same number of errors lie at the same distance from the original
codeword. In other words, every allowed codeword, together with own distorted codewords will
form a cluster. Since all points of this cluster are located on the binary Hamming sphere, they
belong to the same spherical segment, i.e. can be separated by cutting planes. It’s obvious that
the number of such separating planes for the code \((n, k)\) is equal to \(2^k\). In other words, the
simplest neural net decoder-classifier must contain at least one layer with \(2^k\) neurons.

We describe the algorithm for calculating the weighting coefficients this neural net. From the
set of information words \(A = (a_1, a_2, ..., a_{2^k})\), we create the \(2^k\) cluster centers:
\(M = G A^T\). Then
the weights of the first layer are calculated by the formula

\[ W_1 = 2M - J. \]

If we take the activation function of the first layer in the form

\[ f^1(z) = \theta(z - z_{\text{max}}), \]

then at the output of the first layer, we get a vector that determines the position of the allowed
codeword in the matrix \(A\).

To highlight this word, we need a second layer with \(W^2 = A^T\) and linear activation function
\(f^2(z) = z\).

Example 3. Consider the code from Example 1: \(c = (00111011)\). Let during transmission
an error occurred in the sixth character, i.e.

\[ x = b \oplus e = (001110111101) \oplus (000001000000) = (001111111101). \]

The neural net decoder will work as follows (figure 3). Suppose that the information words are
grouped in matrix $A$ in the following way:

$$
A = \begin{pmatrix}
101011 & \ldots & 0 & \ldots & 10101 \\
011001 & \ldots & 1 & \ldots & 10011 \\
000111 & \ldots & 0 & \ldots & 01111 \\
000000 & \ldots & 0 & \ldots & 11111 \\
000000 & \ldots & 1 & \ldots & 11111 \\
000000 & \ldots & 151 & \ldots & 11111_{26}
\end{pmatrix}.
$$

A binary vector in bipolar form $\tilde{x} = 2x - 1$ is fed to the input of the first layer:

$$
x = (-1 - 11111111111 - 11).
$$

Since $z_{max} = max(W^1\tilde{x}) = 10$ corresponds to the 51st element of the vector $W^1\tilde{x}$, then output of the first layer has the form

$$
y^1 = f^1(W^1\tilde{x}) = (000\ldots015100\ldots000_{26}).
$$

This vector determines position (in our case the 51st) of the codeword in the matrix $A$. To restore information sequence we need a second layer, the output of which is

$$
y^2 = f^2(W^2y^1) = (010011) = a.
$$

5. Neural net H-decoder

Recall that linear block code is an error-correcting code, for which codewords $x$ forms a linear $n$-dimensional space $C$. If $A$ is $k$-dimensional space of information words $a$, then for it $r$-dimensional orthogonal subspace can be defined as $H \cong A^\perp$, where $H$ is the matrix which defining the basis of this subspace. It follows that for any $x \in C$ there is a vector $h \in H$ such that

$$
h = Hx.
$$

It is obvious that if $x = c = Ga$, then

$$
h = Hc = 0.
$$

This follows from the fact that for an exact chain complex

$$
0 \rightarrow A^k \xrightarrow{G[n \times k]} C^n \xrightarrow{H[r \times n]} H^r \rightarrow 0
$$
the kernel of the operator $\mathbf{H}$ coincides with the image of $\mathbf{G}$: $\mathbf{H}\mathbf{G} = \mathbf{0}$.

These arguments provide the base for building of the neural net decoder which works in the feature space $\mathcal{H}$, i.e. in the space of syndromes with basis $\mathbf{H}$. Obviously, for $k \ll r$ we have $\dim \mathcal{A} \ll \dim \mathcal{H}$, i.e. maximum number of neurons (classes) for syndrome classifier is limited to $2^r$. However, for recovery of information word with uses syndromes we need to introduce a few additional layers.

Consider the algorithm of an analytical learning of the neural net decoder that built on the base of the parity-check matrix $\mathbf{H}$. For the first layer, we set $\mathbf{W}^1 = \mathbf{H}$, $f_1(z) = 2(z \mod 2) - 1$. The output of the first layer will give a binary representation of the syndrome that defining the error. To localize this error we will use the matrix of syndromes $\mathbf{W}^2 = 2(\mathbf{H}\mathbf{W}^3)^T - \mathbf{J}$, where $\mathbf{W}^3$ is a matrix of all possible error vectors that can be corrected. As activation functions of neurons for second and third layer we take $f_2(z) = \theta(z - r)$ and $f_3(z) = z$. So thus, the output of the third layer will be the error vector.

In the fourth layer we add modulo 2 the origin codeword to error vector.

**Example 4.** Consider the LDPC code (15.5) that correcting $t = 3$ errors

$$
\begin{pmatrix}
0000111111 \\
0111000111 \\
1011011001 \\
1101101010 \\
1110110100 \\
1111111111
\end{pmatrix}
$$

Then the parity-check matrix will be written as $\mathbf{H} = (\mathbf{G}'', \mathbf{I})$, and the generator matrix as $\mathbf{G} =

$$
\begin{pmatrix}
\mathbf{I} \\
\mathbf{G}''
\end{pmatrix}
$$

For information word $\mathbf{a} = (10101)$ we get codeword $\mathbf{c} = \mathbf{G}\mathbf{a} = (101010101010010)$. Suppose that during transmission in the codeword $\mathbf{c}$ an error occurred in the second, third and fourth symbols, i.e.

$$
\mathbf{x} = \mathbf{c} \oplus \mathbf{e} = (110110101010010).
$$

The neural net decoder will work as follows. The output of the first layer is a vector-syndrome in bipolar form

$$
\mathbf{y}^1 = f^1(\mathbf{W}^1\mathbf{x}) = (-1 - 1 - 1111 - 11 - 1 - 1).
$$

At the output of the second layer, we obtain the vector

$$
\mathbf{y}^2 = f^2(\mathbf{W}^2\mathbf{y}^1) = (000...1212...0000576)^T,
$$

which determines the position of the error vector in the matrix of syndromes $\mathbf{W}^2$. We describe an algorithm for constructing this matrix. Denote by $\mathbf{W}^3$ the matrix whose columns are all possible errors that can be corrected:

$$
\mathbf{W}^3 =

$$

As mentioned earlier, $\mathbf{W}^2 = 2(\mathbf{H}\mathbf{W}^3)^T - \mathbf{J}$. 


Since in vector
\[ W^2 y^1 = (2, 2, -2, -2, 2, 0, 0, ..., 10, 12, ..., 4, 4, 0, 0, 0, -4, 0, 576) \]
maximum element "10" stands in 212st place, we need to take column 212 from matrix \( W^3 \). It will coincide with the error vector for the received code combination:

\[ y^3 = W^3 y^2 = (0111000000000015) . \]

After adding the error vector to the origin codeword at the output of the fourth layer we get the corrected information word

\[ y^4 = f^4 \left( W^4 \cdot (x \oplus y^3) \right) = (10101) . \]

Obviously, this neural net decoder is universal, because of as generator \( G \) well as a parity-check matrix \( H \) of any linear block systematic code can be used as matrix of layer weights \( W \).

If we take into account that \( W^4 = \begin{pmatrix} I_k & 0_{r,k} \end{pmatrix} \) and \( f^3(z) = z \) is linear activation function, then this scheme can be reduced to three-layer (figure 4).

Further modifications of the neural net decoder can be made for \( t = 1 \) error correction codes. Note that in this case, the matrix \( W^3 = I \). In other words, the third layer can be combined with the second one and the result will be two-layer neural net H-decoder.

**Example 5.** Reed-Muller code \((16,5)\) which correcting up to 2 errors has a generator matrix

\[
G = \begin{pmatrix}
1111111111111111 \\
0000000011111111 \\
0001111000111111 \\
0011001100110011 \\
0101010101010101
\end{pmatrix}^T .
\]

For the information sequence \( a = (11111) \) the codeword has the form

\[ c = Ga = (1001011001101001) . \]

Let for example, during transmission over a communication channel in a codeword \( c \) an error occurred in the second and third bits, i.e.

\[ x = c \oplus e = (1111011001101001) . \]
To decode the input vector, we construct a two-layer neural net with weights $(W_1, W_2) \approx H_4$, where $H_4$ is the Hadamard matrix, which is formed by the rule

$$H_{k+1} = \begin{pmatrix} H_k & H_k \\ H_k & -H_k \end{pmatrix}, \quad H_0 = 1.$$ 

The input of the first layer is a normalized vector in a bipolar form $\tilde{x} = (1111\ 111\ 1\ 1\ 1\ 1\ 1\ 1)$. The output of the first layer is a vector

$$y^1 = f^1(W_1\tilde{x} + b_1) = (0000000000000001),$$

where $W_1 = H_4$, $b_1 = 0$, $f^1 = \text{arg max}$. This signal marked the position of the most likely corrected codeword in the Hadamard matrix $H_4$. To separate this codeword, we need the second and third layers

$$y^2 = f^2(W_2y^1 + b_2) = (1001010011010111), \quad y^3 = f^3(W_3y^2) = (11111) = a,$$

where $W_2 = H_4$, $b_2 = J$, $W_3 = D$, $f^2(z) = \frac{\theta(z)}{2}$, $f^3(z) = z \mod 2$,

$$D = \begin{pmatrix} 1000000000000000 \\ 1000000010000000 \\ 1001000000000000 \\ 1010000000000000 \\ 1100000000000000 \end{pmatrix}.$$ 

### 6. Conclusions

The paper considers two approaches to the use of neural networks in the problem of decoding of error-correcting codes. This is a classification by the space of information words using generator matrix $G$ and classification in error space (using parity-check matrix $H$). It is possible to determine the limits of applicability for one or another method depending on the number of corrected errors. For code that correcting $t$ errors, there are $2^k$ information vectors and $2^r$ possible syndromes. Obviously, when $r \ll k$ then decoder based on parity-check operator $H$ will contain the less number of elements than the classifier based on the generator matrix $G$. However, when the number of errors $t$ will increase, then the number of information classes will rapidly decrease. As a limit state, we can suggest repeated code $(n, k) = (2t + 1, 1)$ which has only two information classes $0$ and $1$ but $2^{2t+1}$ syndromes. In addition, the parity-check matrix decoders require a construction of deep multilayer net and this increases number of necessary coefficients and shifts bound of choice to the direction of the generator matrix classifier.

### References

[1] Haykin S 2008 *Neural Networks and Learning Machines* (New Jersey: Pearson)
[2] Morelos-Zaragoza R H 2006 *The Art of Error Correcting Coding* (New York: Wiley)
[3] Zeng G, Hush D and Ahmed N 1989 An application of neural net in decoding error-correcting codes *IEEE International Symposium on Circuits and Systems* 2 782–5
[4] Htay M M 1992 A computational framework for efficient error correcting codes using an artificial neural network paradigm *PhD Dissertation. Louisiana State University and Agricultural & Mechanical College* pp 91
[5] Ortuó I, Ortuó M and Delgado J 1992 Error correcting neural networks for channels with Gaussian noise *IJCNN International Joint Conference on Neural Networks* 4 295–300
[6] Ja-Ling Wu, Yuen-Hsien Tseng and Yuh-Ming Huang 2002 Neural network decoders for linear block codes *International Journal of Computational Engineering Science* 3 235–55
[7] Nachmani E, Marciano E, Lugosch L, Gross W J, Burshtein D and Beery Y 2018 Deep learning methods for improved decoding of linear codes *IEEE Journal of Selected Topics in Signal Processing* 12 119–31 (*Preprint* arXiv:cs.IT/1706.07043)
[8] Lugosch L and Gross W J 2017 Neural offset min-sum decoding *IEEE International Symp on Information Theory ISIT 2017 Aachen Germany June 25-30 2017* 1361–5

[9] Nachmani E, Marciano E, Burshtein D and Beery Y 2018 Near maximum likelihood decoding with deep learning *(Preprint cs.IT/1801.02726)*

[10] Bennatan A, Choukroun Y and Kisilev P 2018 Deep learning for decoding of linear codes - a syndrome-based approach *IEEE International Symposium on Information Theory, ISIT 2018 Vail CO USA June 17–22 2018* 1595–9 *(Preprint cs.IT/1802.04741)*