Neural net decoders of nonbinary codes

V V Butov, V N Dumachev and S E Fedyaeva
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, svetlanafedyaeva20@gmail.com;

Abstract. In paper, neural net decoders of nonbinary error correction codes are considered. Analytic methods for calculating of synapse weight coefficients are proposed. It is shown that for codes \((n, k)\) with a small number of corrected errors \((n - k \ll k)\), it is advisable to use a 6-layer universal classifier based on the feature space of parity symbols. For codes with \(k \ll n - k\), it is proposed to use a 3-layer classifier on feature space of information symbols.

1. Introduction
For the first time the idea of using neural net calculations [1] for building of error correction codes [2] was proposed in the 90s of the XX century [3–5]. In these papers was considered small dimensions codes \((n, k)\) and weight coefficients were analytically calculated. At the same time, it was immediately noticed that the increase of size of an information word \(k\) lead to an increase of neural net dimension as proportional to \(2^k\). This fact stopped the development of directing for ten years. In 2002, J.Wu, Y.Theng and Y. Huang [6] proposed own version of a 2-layer BCH decoder. For finding weight coefficients of the code \((n, k)\) with the help of genetic algorithms they solved systems of \(k\) nonlinear equations from \(n\) variables. The degree of equations in some cases reached \(n\). Computer capacities allowed them to learning decoders up to order \((n, k) = (21, 6)\) inclusive. Further research was again stopped by nonlinear dependence of the neural net dimension from the codeword length. The explosive increase of computer performance in recent years has allowed researchers to pay attention to this task again. For example decoders BCH(63,45), BCH(127,106), BCH (127,64) are analyzed [7–10]. To build such decoders, neural net with 1200 layers are used [9]. In [11], an analytical algorithm for adjustment of the weight coefficients for the universal four-layer neural net decoder is proposed. Further, this algorithm was used for Hamming code, LDPC, convolution code of Viterbi and Reed-Muller codes [12]. In this paper, we propose an analytic algorithm setting of weighting coefficients for a nonbinary neural net decoder. As an example, a model of a 6-layer neural net for the Reed-Solomon code is considered.

2. Neural net encoder
In order for the algorithm to be implemented by a neural network, it must be transformed to a matrix form. We give a matrix realization for algorithm of the Reed-Solomon coding in \(GF(2^m)\). For this, we represent the information sequence \(a = (a_1a_2a_3...a_k)\) in a polynomial form

\[ a(x) = \sum_{i=1}^{k} a_ix^{k-i} = a_1x^{k-1} + a_2x^{k-2} + ... + a_{k-2}x^2 + a_{k-1}x + a_k. \]
We shift the information sequence by \( r = 2t \) digits to the left, i.e. we multiply the information polynomial \( a(x) \) by \( x^r \): \( q(x) = x^r a(x) \). Next, we divide obtained polynomial into generator \( p(x) \):

\[
\frac{q(x)}{p(x)} = C(x) \oplus \frac{R(x)}{p(x)}, \quad q(x) = C(x) \cdot p(x) \oplus R(x).
\]

The transmitted code combination now has the form

\[
c(x) = q(x) \oplus R(x) \quad \text{or} \quad c = (a_1a_2a_3...a_k e_1e_2...e_r).
\]

Generally, the generator polynomial of Reed-Solomon code has the form

\[
p(x) = \prod_{i=0}^{u+2t-1} (x \oplus 2^i),
\]

where \( t \) is the number of errors that can be corrected by the code, \( u \) is an arbitrary integer \( 0 \leq u < m \). To each generator polynomial \( p(x) \) we can associate generator vectors \( d^i \), which allow to compute check symbols \( e_i \) for information vector \( a \):

\[
e_k = (a \cdot d^k).
\]

Recall that all operations must be carried out in the Galois field \( GF(2^m) \).

**Example 1.** Consider the Galois field \( GF(2^3) \) constructed from the irreducible polynomial of degree 3: \( g(x) = x^3 + x + 1 \) (i.e. \( GF(2^3) = GF(2)[X]/(X^3 + X + 1) \)). For the Reed-Solomon code in \( GF(2^3) \), which may corrects \( t = 1 \) error, the correspondence between the generator vector \( d \) and the generator polynomial \( p(x) \) has the form

\[
\begin{array}{cccc}
u & p(x) & d^1 & d^2 \\
0 & x^2 + 3x + 2 & 52473 & 43562 \\
1 & x^2 + 6x + 3 & 67716 & 22313 \\
2 & x^2 + 7x + 7 & 44247 & 15137 \\
3 & x^2 + 5x + 1 & 15665 & 56651 \\
4 & x^2 + 1x + 4 & 73151 & 74244 \\
5 & x^2 + 2x + 6 & 31322 & 61776 \\
6 & x^2 + 4x + 5 & 26534 & 37425 \\
\end{array}
\]

Then, for information sequence \( a = (54321) \) with \( u = 2 \), we have

\[
e_1 = (54321) \cdot (44247) = 2 \oplus 6 \oplus 6 \oplus 3 \oplus 7 = 6,
\]

\[
e_2 = (54321) \cdot (15137) = 5 \oplus 2 \oplus 3 \oplus 6 \oplus 7 = 5,
\]

i.e. codeword \( c = (5432165) \).

Hence it is easy to construct the generator matrix \( G \) of the Reed-Solomon code

\[
G = \left( \begin{array}{c}
I \\
d^1 \\
d^2
\end{array} \right)
\]

and get \( c = Ga \).

Each layer of a feedforward neural net is described by equation

\[
y = f(Wx + b).
\]
Here, \( x \) is the input and \( y \) is the output vector of the layer with the weight matrix \( W \), the shift \( b \) and the activation function \( f(z) \).

To construct a neural net encoder we need to obtain a matrix analogs of operations in the Galois field \( GF(2^m) \) and build the generator matrix \( G \). For each field \( GF(2^m) \), we construct own addition table and the multiplication table. For example, for the field \( GF(2^3) \), which is constructed using the polynomial \( g(x) = x^3 + x + 1 \), these tables has the form:

| \( \oplus \) | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|------------|---|---|---|---|---|---|---|
| 1          | 0 | 3 | 2 | 5 | 4 | 7 | 6 |
| 2          | 3 | 0 | 1 | 6 | 7 | 4 | 5 |
| 3          | 2 | 1 | 0 | 7 | 6 | 5 | 4 |
| 4          | 5 | 6 | 7 | 0 | 1 | 2 | 3 |
| 5          | 4 | 7 | 6 | 1 | 0 | 3 | 2 |
| 6          | 7 | 4 | 5 | 2 | 3 | 0 | 1 |
| 7          | 6 | 5 | 4 | 3 | 2 | 1 | 0 |

| \( \times \) | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|------------|---|---|---|---|---|---|---|
| 1          | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 2          | 2 | 4 | 6 | 3 | 1 | 7 | 5 |
| 3          | 3 | 6 | 5 | 7 | 4 | 1 | 2 |
| 4          | 4 | 3 | 7 | 6 | 2 | 5 | 1 |
| 5          | 5 | 4 | 2 | 7 | 3 | 6 | 1 |
| 6          | 6 | 7 | 1 | 5 | 3 | 2 | 4 |
| 7          | 7 | 5 | 2 | 1 | 6 | 4 | 3 |

Next, we will use the binary representation of multiplication operations in \( GF(2^3) \): 

\[
1 \times = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}, \quad 2 \times = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}, \quad 3 \times = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}, \quad 4 \times = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix},
\]

\[
5 \times = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}, \quad 6 \times = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}, \quad 7 \times = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}.
\]

Then the generator matrix for the \( RS(n,k)=RS(7,5) \) code in \( GF(2^3) \) with \( u = 2 \) that corrects \( t = 1 \) error will have the form:

\[
G = \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 4 \\ 4 \\ 2 \\ 4 \\ 7 \\ 1 \\ 5 \\ 1 \\ 3 \\ 7 \end{pmatrix} = \begin{pmatrix} 1 & I & I & I & I \\ 101 & 101 & 010 & 101 & 011 \\ 110 & 110 & 101 & 110 & 001 \\ 010 & 010 & 100 & 010 & 111 \\ 100 & 001 & 100 & 110 & 011 \\ 010 & 100 & 010 & 111 & 001 \\ 001 & 011 & 001 & 101 & 111 \end{pmatrix} = \left( I \quad G' \right).
\]

The neural net encoder will work as follows (figure 1). If input of first layer is the information word \( a = (a_1, a_2, a_3, ... , a_k) \) then the weight matrix \( D[k,3k] \) has the form:

\[
D = \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{pmatrix}
\]
The activation function of the first layer written as
\[ f^1(z) = \begin{pmatrix} \varphi^1(z) \\ \varphi^2(z) \\ \vdots \\ \varphi^k(z) \end{pmatrix}, \quad \varphi(z) = \begin{pmatrix} \lfloor \frac{z}{2^m} \rfloor \mod 2 \\ \lfloor \frac{z}{2^t} \rfloor \mod 2 \\ \lfloor \frac{z}{2^1} \rfloor \mod 2 \end{pmatrix}, \quad (3) \]

where \( \lfloor z \rfloor \) is the floor function.

The weight matrix and activation function of second layer are \( W^2 = G \) and \( f^2(z) = z \mod 2 \) respectively. Third layer with the linear activation function \( f^3(z) = z \) is needed to convert the sequence from binary to decimal. We denote the matrix realizing this operation over each 3-bit block as
\[
D = \begin{pmatrix} 421 \\ 421 \\ \vdots \\ 421 \end{pmatrix}.
\]

Note that to configure this neural net we need to calculate
\[
W^1[(2^m - 1 - 2t) \times (2^m - 1 - 2t)m], \\
W^2[(2^m - 1 - 2t)m \times (2^m - 1)m], \\
W^3 [(2^m - 1)m \times (2^m - 1)].
\]

It will be necessary to determine
\[ N = m \cdot (k^2 + nkm + n^2) \]
synaptic weights.

**Figure 1.** Neural net RS(2^m - 1, 2^m - 1 - 2t) encoder in GF(2^m). There \( t \) is a number of corrected errors.

**Example 2.** For information sequence from the previous example \( a = (54321) = (101100011010001) \) we get
\[
y^1 = f^1(W^1a) = f^1(555444333222111) = (101100011010001), \\
y^2 = f^2(W^2y^1) = f^2(101100011010001552343) = (101100011010001110101), \\
y^3 = f^3(W^3y^2) = (5432165) = c,
\]

where \( W^1 = D, \; W^2 = G, \; W^3 = D. \) ▲
3. Neural net G-decoder

If an error occurs, the correct codeword transforms to corrupt [2]. Each word can be represented by a point in Hamming space. For binary words the Hamming space is a sphere [11]. In this case each original codeword, together with its own distorted words, can be grouped into clusters. Since all cluster points are located on the Hamming sphere, they belong to the same spherical segment, i.e. can be separated from each other by cutting planes. Obviously, that the number of such separating planes for the code \((n, k)\) is equal to \(2^k\) (the number of possible information words). In other words the simplest neural net decoder-classifier should contain at least one layer with \(2^k\) neurons.

Recall the simplest binary neural net classifier [11] based on generator matrix of code G. Let the information word \(a = (a_1a_2...a_k)^T\) be encoded as \(c = Ga\). We denote the received code combination by the symbol \(x\) and apply to input it normalized value \(\tilde{x}_i = 2x_i - 1\), i.e. \(\tilde{x}_i \in [-1, 1]\). Then from the set of all \(2^k\) information words \(A = (a_1a_2...a_k)\) we can create \(2^k\) centers of clusters: \(M = GA\). The transformation \(W^1 = 2M - J\) gives synaptic weights for first hidden layer our neural net. The output vector of the first layer is determines the position of information word in matrix A. To select this word, we need the second layer with \(W^2 = 2A - J\), where \(J\) is the matrix of units. Thus, to training neural network, we need to calculate \(W^1[2^k \times n]\) and \(W^2[2^k \times k]\). This construction can be applied to any linear binary block codes and we call it as G-decoder.

In the non-binary case, all calculations are carried out in Galois fields \(GF(2^m)\). This leads to the fact that the distances between two codewords in the Hamming space are not equal to the distance in the metric of \(GF(2^m)\). For example, the distance between the numbers 6 and 5 in Galois field is calculated as \(dist(6, 5) = 6 \oplus 5 = 110 \oplus 101 = 011 = 3\). Whereas in Hamming space this distance is \(dist(6, 5) = \sum (6 \oplus 5) = \sum (110 \oplus 101) = \sum 011 = 2\). We want to apply the binary decoder obtained earlier to the decoding of non-binary words. The structure of the neural net G-decoder is shown in figure 2.

![Figure 2. Neural net RS(2^m - 1, 2^m - 1 - 2t) G-decoder.](image-url)

Let the input of the first layer be a matrix whose columns are the received codeword \(X = (xxx...x_{3m})\). Depending on the number of correctable errors \(t\), the dimension of this matrix is \(X[2m(2^m - 1 - 2t)]\). Let \(W^1 = GA\) be the matrix of all possible codewords obtained from all possible information words \(A = (a_1a_2...a_{2mk})\). The operation of the first layer is the difference between the distorted received codeword and the undistorted one. Since the activation function \(f^1(z) = \theta(z)\) is Heaviside function, then output matrix of the first layer contains "1" if the components of the two input matrices coincide and "0" otherwise. The operator of the second layer \(W^2 = (111...1_{2m-1})^T\) gives the number of coincident nonzero components in each column of the input matrix. We need to determine the position of the maximum element in this vector. For this, the offset Heaviside function \(f^2(z) = \theta(z - z_{max})\) is used. Now the output of the
second layer is a zero vector with unity at the position of maximum coincidence of the distorted code word \( x \) with the standard \( c = G\alpha \). This is essentially a maximum likelihood assessment in its purest form. The third layer highlights the most likely element in the reference matrix \( W^3 = A \).

Note that to configure this neural net we need to calculate

\[
W^1[(2^m - 1) \times 2^{km}], \\
W^2[(2^m - 1) \times 1], \\
W^3 \left[ (2^m - 1 - 2t) \times 2^{mk} \right].
\]

It will be necessary to determine

\[
N = n + (n + k)2^{mk}
\]
synaptic weights.

**Example 3.** Consider RSC(7.5) from the previous example \( c = (5432165) \). Assume during transmission in the codeword \( c \) an error occurred in the first symbol with the amplitude \( E_1 = 5 \), i.e.

\[
x = c \oplus e = (5432165) \oplus (5000000) = (0432165).
\]

The neural net G-decoder will work according to the following scheme. After filling of the matrix of the input layer, it added to the matrix of all possible codewords

\[
X = (xxx...x_{2^{15}}) = \begin{pmatrix}
000...0 \\
444...4 \\
333...3 \\
222...2 \\
111...1 \\
666...6 \\
555...5_{2^{15}}
\end{pmatrix}.
\]

The output of the first layer is a matrix of coincide elements of possible codewords and the input vector

\[
Y^1 = f^1(W^1 - X) = \begin{pmatrix}
000...0 & 0 \\
000...1 & 0 \\
000...1 & 0 \\
000...1 & 0 \\
000...1 & 0 \\
010...1 & 0 \\
000...1_{17806} & 0_{2^{15}}
\end{pmatrix},
\]

\[
W^1 = GA = \begin{pmatrix}
0426153704261537043...5 & \ldots61537 \\
00000000444444444222...4 & \ldots777777 \\
00000000000000000000003 & \ldots777777 \\
000000000000000000000002 & \ldots777777 \\
000000000000000000000001 & \ldots777777 \\
0635427160532417350...6 & \ldots32417 \\
0426153726043715153...5_{17806} & \ldots61537_{2^{15}}
\end{pmatrix}.
\]

The second layer selects the column with the most number of coinciding elements in this matrix.

\[
y^2 = f^2(W^2Y^1) = f^2(01010100211...6_{17806}...10100_{2^{15}}) = (000...1_{17806}...000_{2^{15}}),
\]
where $W^2 = (1111111)^T$.

The third layer extracts the corrected information word from matrix $A$

$$ y^3 = f^3(W^2y^2) = Ay^2 = (54321) = a, \quad f^3(z) = z. $$

4. Neural net H-decoder

Further we consider an analytic tuning algorithm for a neural net decoder which built on the check matrix basis $H = (G', I)$. This construction we call as H-decoder (figure 3).

![Figure 3. Neural net RS H-decoder for $t = 1$ error.](image)

The first layer with the matrix $W^1 = D$ (see (2)) and activation function (3) is needed to convert the input vector $x$ from the decimal to binary. For the second layer, we put $W^2 = H$, $f^2(z) = 2(z \mod 2) - 1$. The output of the second layer will give a binary representation of the syndrome that determines the error. To localize an error we use the matrix of syndromes $W^3$.

Then the output of the third layer is a vector, which determines the position of the syndrome in the matrix $W^4$. Wherein, $W^3$ can be obtained by the mapping $W^3 = 2(HW^4)^T - J$, where $W^4$ is a matrix consisting of all possible error vectors that can be corrected. As functions of activation for the third and fourth layer, we take $f^3(z) = \theta(z - r)$ and $f^4(z) = z$. Thus, the output of the fourth layer is the error vector. In the fifth layer, we add the original code sequence in binary $y^1$ and the error vector $e$. The activation function is $f^5(z) = z \mod 2$. The six layer converts the vector from a binary representation to a decimal.

Note that to configure this neural net, we need to calculate

$$ W^1[(2^m - 1) \times (2^m - 1)m], $$
$$ W^2[(2^m - 1)m \times 2tm], $$
$$ W^3[2tm \times (2^m - 1)^2 + 1], $$
$$ W^4 [(2^m - 1)^2 + 1 \times (2^m - 1)m], $$
$$ W^5 [(2^m - 1)m \times (2^m - 1)m], $$
$$ W^6[(2^m - 1)m \times (2^m - 1 - 2t)]. $$

It will be necessary to determine

$$ N = nm(n + 2nm - km + k) + (2n - k)(n^2 + 1)m $$

weight coefficients.
Example 4. Consider RSC(7.5) from the previous example $c = (5432165)$. Assume during transmission in the codeword $c$ an error occurred in third symbol with the amplitude $E_3 = 7$, i.e.

$$x = c \oplus e = (5432165) \oplus (0070000) = (5442165).$$

The classical solution to this task is given by the following sequence of operations [2]. Using the check matrix in Wandermond form

$$H' = \begin{pmatrix}
(2^u)_6 & (2^u+1)^6 \\
(2^u)_5 & (2^u+1)^5 \\
(2^u)_4 & (2^u+1)^4 \\
(2^u)_3 & (2^u+1)^3 \\
(2^u)_2 & (2^u+1)^2 \\
(2^u)_1 & (2^u+1)^1 \\
(2^u)_0 & (2^u+1)^0
\end{pmatrix},$$

we find the syndrome vector $s = H'x$. Solving the equation $s_1 \sigma = s_2$, we obtain the coefficients of the polynomial of error locators $\Theta = 1 + \sigma x = 1 + 2^\alpha x$. The exponent of the root of the locator polynomial $\alpha$ is the position of the error in codeword $c = (c_6c_5c_4c_3c_2c_1c_0)$. The amplitude of the error is calculated by the formula $\sigma^\alpha E_\alpha = s_1$.

In our case, we get

$$s = H'x = \begin{pmatrix} 7325641 \\ 6274531 \end{pmatrix} = \begin{pmatrix} 5 \\ 3 \end{pmatrix}.$$

Solving the equation $5\sigma = 3$, we obtain $\sigma = 6 = 2^4$, i.e. $\alpha = 4$ is the position of the error. The amplitude of the error is calculated by the formula $6^2E_4 = 5$, i.e. $E_4 = 7$. Now the corrected codeword is equal

$$c = x \oplus e = (5442165) \oplus (0070000) = (5432165).$$

Recall that we perform all calculations in $GF(2^3)$ with addition and multiplication tables (1).

The neural net decoder will work according to the following scheme. The output of the first layer is a binary vector

$$y^1 = f_1(Dx) = \begin{pmatrix} 1 \ 0 \ 1 \ 1 \ 0 \ 0 \ 0 \ 1 \ 1 \ 0 \ 1 \ 1 \ 0 \ 1 \ 1 \ 1 \ 0 \ 1 \ 0 \ 0 \ 1 \ 1 \ 0 \ 1 \ 0 \ 1 \ 0 \ 0 \ 1 \ 1 \ 1 \ 1 \ 0 \ 1 \end{pmatrix},$$

where $f_1(z)$ is the activation function (3). The output of the second layer is a vector-syndrome

$$y^2 = f_2(W^2y^1) = (1, -1, 1, 1, 1).$$

The output of the third layer is a vector

$$y^3 = f_3(W^3y^2) = (000...122...000050)^T,$$

which determines the position of the error vector in the matrix of syndromes $W^3$. We describe the algorithm for constructing this matrix. Let $W^4$ is a matrix whose columns are all possible errors that can be corrected:

$$W^4 = \begin{pmatrix}
01010101 \\
00110011 \\
00001111 \\
10101011 \\
01100111 \\
00011111 \\
... \\
10101011 \\
01100111 \\
00011111
\end{pmatrix}.$$
Now describes in more detail the method of obtaining this matrix. Let $E$ be a matrix in $GF(2^m)$ whose columns are all possible codeword errors

$$
E = \begin{pmatrix}
01234567 & 0000000 \\
1234567 & 1234567 \\
& 1234567 \\
& 1234567 \\
& \vdots \\
0000000 & 1234567 \\
\end{pmatrix}.
$$

Converting this matrix using the operator $\overline{D}$ we get its binary representation $W^4 = f^1(\overline{DE})$. Note that there are no errors in the first column $E$ (or $W^4$).

As it was said earlier $W^3 = 2(HW^4)^T - J$. Since in the vector

$$
y^2 = (-4, -2, -2, 4, 0, \ldots, 6_{22}, \ldots, -2, 0, 0, 2_{50})
$$

the maximum element "6" is located on the 22-st place, then we need take 22 columns from the matrix $W^4$. It will coincide with the error vector for the received code combination:

$$
y^4 = W^4y^3 = (000 000 111 000 000 000 000).
$$

Adding the error vector to the input code combination, we get the corrected codeword in bit form, i.e. output of the five layer

$$
y^5 = f^5(W^5 \cdot (x \oplus y^4)) = (101 100 011 010 001 110 101), \ W^5 = I.
$$

The output of the six layer is the correct information word

$$
y^6 = f^6(W^6y^5) = Dy^5 = (54321) = a. \bigtriangleup
$$

5. Conclusions

We considered two basic approaches to use of neural net in problem decoding of the error correction code. This is a classification by the information space and by the error space. We can define limits for applicability of a method in depending on the number of corrected errors. For the code correcting the $t$ errors there are $2^k$ information vectors and $2^r$ possible syndromes. It’s obvious that at $r < k$ the decoder on the basis of the checking operator $H$ will contain fewer elements than the classifier on basis of the generator matrix $G$. However, with increasing number of bugs fixed $t$ number of information classes sharply decreases. As a limit case, one can propose the repeated code $(n, k) = (2^t + 1, 1)$ having two information classes $(0,1)$ and $2^{2t+1}$ check classes. Therefore, for $k < r$ the use of $G$-decoders is more preferable than $H$-decoders.

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 p 91
[5] Ortúo I, Ortúo 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 Beéry Y 2018 Deep learning methods for
improved decoding of linear codes IEEE Journal of Selected Topics in Signal Processing 12 119–31 (Preprint
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 Beéry 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)

[11] Dumachev V N, Kopylov A N and Butov V V 2019 Neural net decoders for linear block codes Bulletin of the
South Ural State University. Ser. Mathematical Modelling, Programming & Computer Software 12 128–35.

[12] Butov V V and Dumachev V N 2019 Neural net decoders of binary codes Journal of Physics: Conf. Series.
1203 012002.