



**Министерство науки и высшего образования Российской Федерации  
Федеральное государственное бюджетное образовательное учреждение  
высшего образования  
«Московский государственный технический университет  
имени Н.Э. Баумана  
(национальный исследовательский университет)»  
(МГТУ им. Н.Э. Баумана)**

---

**ФАКУЛЬТЕТ «Информатика и системы управления» (ИУ)**

**КАФЕДРА «Информационная безопасность» (ИУ8)**

**Отчёт**

**по лабораторной работе № 1  
по дисциплине «Интеллектуальные технологии информационной безопасности»**

**Тема: «Исследование однослойных неронных сетей на примере моделирования  
булевых выражений»**

**Вариант 4**

**Выполнил: Григорьев Е.Г.,  
студент группы ИУ8-63**

**Проверил: Строганов И.С.,  
преподаватель каф. ИУ8**

**г. Москва,  
2021 г.**

## 1. Цель работы

Исследовать функционирование простейшей нейронной сети (НС) на базе нейрона с нелинейной функцией активации и обучить ее по правилу Видроу-Хоффа.

## 2. Условия

Условия согласно варианту 4:

Булева функция от 4х переменных:

$$F(x_1, x_2, x_3, x_4) = (\overline{x_1} + x_3)x_2 + x_2x_4$$

Функция активации 1:

$$f(net) = \begin{cases} 1, & net \geq 0 \\ 0, & net < 0 \end{cases}$$

Функция активации 2:

$$f(net) = \frac{1}{2} \left( \frac{net}{1 + |net|} + 1 \right)$$

Норма обучения  $\eta = 0.3$

## 3. Аналитическая часть

Алгоритм функционирования НС с пороговой ФА имеет вид

$$net = \sum_{i=1}^4 w_i x_i + w_0$$
$$y(net) = \begin{cases} 1, & net \geq 0 \\ 0, & net < 0 \end{cases}$$

Где  $net$  – сетевой(комбинированный) вход, а  $y$  – реальный выход НС.

Алгоритм функционирования НС с логической ФА выглядит следующим образом:

$$net = \sum_{i=1}^4 w_i x_i + w_0$$
$$out = f(net)$$

$$y(out) = \begin{cases} 1, & out \geq 0.5 \\ 0, & net < 0.5 \end{cases}$$

Где out – сетевой (недискретизированный) выход НС

Для необученной НС ее реальный выход  $y$  в общем случае отличается от целевого выхода  $t$ , представляющего собой значения заданной БФ нескольких переменных

$F(x_1, x_2, x_3, x_4): \{0, 1\}^4 \rightarrow \{0, 1\}$ , т. е. имеется хотя бы один набор сигналов  $(x_1, x_2, x_3, x_4)$ , для которого ошибка  $\delta = t - y \neq 0$

Правило Видроу – Хоффа (дельта правило):

$$w_i^{l+1} = w_i^l + \Delta w_i^l$$

$$\Delta w_i^l = \eta \delta^l \frac{df(net)}{d net} x_i^l$$

На каждой эпохе  $k$  суммарная квадратичная ошибка  $E(k)$  равна расстоянию Хемминга между векторами целевого и реального выходов по всем входным векторам  $x_1, x_2, x_3, x_4$

## 4. Ход работы

Получим нейросетевую модель булевой функции (таблица 1)

$$F(x_1, x_2, x_3, x_4) = x_1 + \overline{x_2} + \overline{(x_3 + x_4)}$$

Таблица 1. Таблица истинности БФ

<b>F</b>	0	0	0	0	1	1	1	1	0	0	0	0	0	1	1	1
<b>x4</b>	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
<b>x3</b>	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
<b>x2</b>	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
<b>x1</b>	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1

На начальном шаге  $l = 0$  (эпоха  $k = 0$ ) весовые коэффициенты берутся в виде:

$$w_0^0 = w_1^0 = w_2^0 = w_3^0 = w_4^0 = 0$$

Используя ФА 1. Динамика НС представлена в таблице 2, график суммарной ошибки приведен на рисунке 2.

Таблица 2. Параметры НС на последовательных эпохах (Пороговая ФА)

----- FA 1st type -----							
EPOCH	FUNCTION	WEIGHTS					ERROR
0	1111111111111111	0.0000	0.0000	0.0000	0.0000	0.0000	9
1	1111111111111111	0.0000	0.0000	0.6000	0.0000	0.3000	9
2	0000111100001111	-0.600	0.0000	0.6000	0.0000	0.3000	1
3	0101111101011111	-0.600	0.0000	0.6000	0.0000	0.6000	5
4	0101111101011111	-0.600	0.0000	0.9000	0.0000	0.6000	5
5	0000111100001111	-0.900	0.0000	0.9000	0.0000	0.6000	1
6	0101111101011111	-0.900	0.0000	0.9000	0.0000	0.9000	5
7	0101111101011111	-0.900	0.0000	1.2000	0.0000	0.9000	5
8	0000111100001111	-1.200	0.0000	1.2000	0.0000	0.9000	1
9	0101111101011111	-1.200	0.0000	1.2000	0.0000	1.2000	5
10	0001111100011111	-1.200	0.0000	1.5000	0.3000	0.9000	3
11	0000111100001111	-1.500	0.0000	1.5000	0.0000	0.9000	1
12	0000111100001111	-1.500	0.0000	1.5000	0.0000	1.2000	1
13	0001111100011111	-1.500	0.0000	1.5000	0.3000	1.2000	3
14	0000111100001111	-1.500	0.0000	1.8000	0.3000	0.9000	1
15	0000111100001111	-1.500	0.0000	1.8000	0.6000	0.9000	1
16	0001111100011111	-1.500	0.0000	1.8000	0.9000	0.9000	3
17	0000111100001111	-1.800	0.0000	1.8000	0.6000	0.9000	1
18	0000111100001111	-1.800	0.0000	1.8000	0.9000	0.9000	1
19	0000011100000111	-2.100	-0.300	1.5000	0.9000	0.9000	1
20	0000111100001111	-1.800	-0.300	1.8000	0.9000	0.9000	0
[-1.8 -0.3 1.8 0.8999999999999999 0.8999999999999999] true							

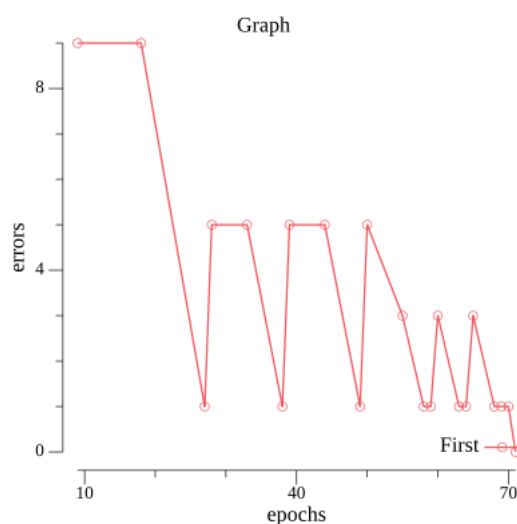


Рисунок 2 – График суммарной ошибки НС по эпохам обучения (пороговая ФА)

Используя логическую ФА и считая  $\frac{df(net)}{dnet} = \frac{1}{2(|net|+1)^2}$  получим результаты приведенные в таблице 3 и на рисунке 3.

Таблица 3. Параметры НС на последовательных эпохах (ФА Softsign)

----- FA 2nd type -----							
EPOCH	FUNCTION	WEIGHTS					ERROR
0	1111111111111111	0.0000	0.0000	0.0000	0.0000	0.0000	9
1	1111111111111111	0.0575	0.0941	0.2075	0.1362	0.0872	9
2	0000111100000011	-0.273	-0.100	0.3038	0.1362	0.0204	1
3	0111111011111111	-0.136	0.0361	0.4402	0.1362	0.1568	7
4	0001111000111111	-0.262	0.0540	0.4581	0.1362	0.1356	3
5	0000111001111111	-0.292	0.1712	0.5752	0.1379	0.0800	3
6	0000111000011111	-0.421	0.0420	0.5912	-0.007	0.2067	1
7	0000111000011111	-0.400	0.0635	0.6126	0.1164	0.2067	1
8	0000111000001111	-0.492	-0.029	0.5205	0.1164	0.2067	0

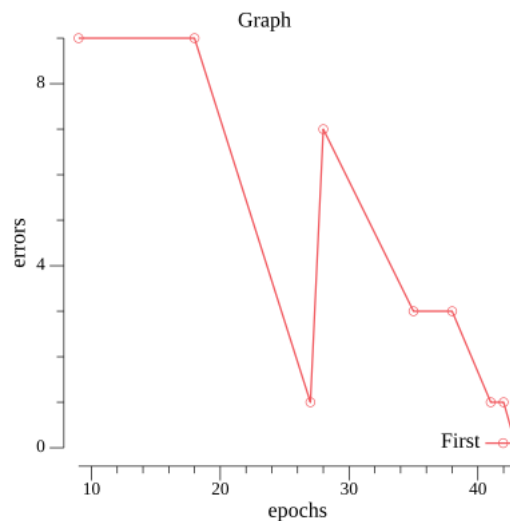


Рисунок 3 – График суммарной ошибки НС по эпохам обучения (ФА Softsign)

Найдем минимальное подмножество обучающих векторов при котором достижима нулевая ошибка.

```
----- Get smallest values -----
fast set of size: 15 epochs: 5 variables: [0 1 2 3 4 5 6 7 8 9 10 11 12 13 15]
fast set of size: 14 epochs: 4 variables: [0 1 2 3 5 6 7 8 9 10 11 12 14 15]
fast set of size: 13 epochs: 4 variables: [0 1 2 3 6 7 8 9 10 11 12 13 14]
fast set of size: 12 epochs: 4 variables: [0 1 2 3 5 6 7 8 10 11 12 13]
fast set of size: 11 epochs: 3 variables: [0 1 2 3 7 8 11 12 13 14 15]
fast set of size: 10 epochs: 2 variables: [1 2 3 7 8 9 10 11 12 15]
fast set of size: 9 epochs: 2 variables: [1 2 3 7 8 9 10 11 15]
fast set of size: 8 epochs: 2 variables: [1 2 3 7 8 9 10 15]
fast set of size: 7 epochs: 2 variables: [1 2 3 7 8 10 15]
fast set of size: 6 epochs: 2 variables: [2 3 7 8 9 15]
fast set of size: 5 epochs: 2 variables: [2 3 7 9 15]
fast set of size: 4 epochs: 2 variables: [2 7 9 15]
cannot study on this small variables n: 3
cannot study on this small variables n: 2
```

## Рисунок 4 – Поиск минимального подмножества

Уменьшим размер выборки до 4 наборов:

$$x^2 = (0, 0, 1, 0) \quad x^7 = (0, 1, 1, 1) \quad x^9 = (1, 0, 0, 1) \quad x^{15} = (1, 1, 1, 1)$$

Результаты приведены в таблице 3 и на рисунке 5

Таблица 3. Параметры НС на последовательных эпохах (ФА Softsign) с уменьшенной выборкой

----- FA smallest values -----								
EPOCH	FUNCTION	WEIGHTS						ERROR
0	1111	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	2
1	0101	-0.124	-0.062	0.1685	0.0185	0.0265	0.0000	0

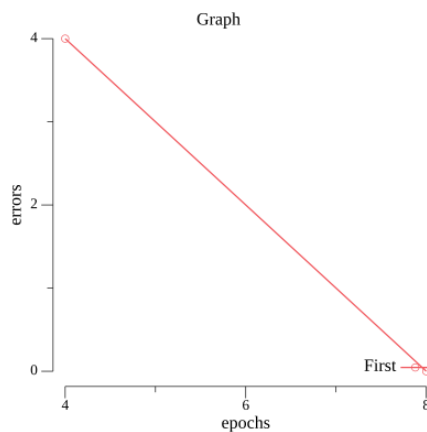


Рисунок 4 – График суммарной ошибки НС по эпохам обучения (ФА Softsign) с уменьшенной выборкой

Код программы приведен в Приложении А.

## 5. Выводы

В ходе выполнения лабораторной работы, мною было исследовано функционирование простейшей нейронной сети (НС) на базе нейрона с нелинейной функцией активации. Также она была обучена по правилу Видроу-Хоффа. Результаты совпали с ожидаемыми, что говорит о корректности работы программы.

## Приложение А. Исходный код программы

*Файл main.go*

```
package main

import (
    "fmt"
    "strings"
)

var Function = []int{0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 1, 1, 1}

func main() {
    fmt.Println(strings.Repeat("-", 30), "FA 1st type", strings.Repeat("-", 30))
    {
        nw := NewNeuralNetwork(1, 0.3, Function)
        nw.Study(Set)
        fmt.Println(nw.Weights, nw.Run())
        Plot(nw.epochsArr, nw.errorsArr, "1.png")
    }
    fmt.Println("\n\n", strings.Repeat("-", 30), "FA 2nd type", strings.Repeat("-", 30))
    {
        nw := NewNeuralNetwork(2, 0.3, Function)
        nw.Study(Set)
        Plot(nw.epochsArr, nw.errorsArr, "2.png")
    }
    fmt.Println("\n\n", strings.Repeat("-", 25), "Get smallest values",
strings.Repeat("-", 25))
    {
        FindMinSet()
    }
    fmt.Println("\n\n", strings.Repeat("-", 25), "FA smallest values",
strings.Repeat("-", 25))
    {
        nw := NewNeuralNetwork(2, 0.3, Function)
        nw.Study(MinSet)
        Plot(nw.epochsArr, nw.errorsArr, "3.png")
    }
}
```

*Файл NeuralNetwork.go*

```
package main

import (
    "fmt"
    "math"
    "strconv"
    "strings"
)

type NeuralNetwork struct {
    funcType int
    nu        float64
    Weights   [5]float64
    boolFunc  []int
    epochsArr []int
    errorsArr []int
}
```

```

}

func NewNeuralNetwork(funcType int, nu float64, boolFunc []int) *NeuralNetwork {
    return &NeuralNetwork{funcType: funcType, nu: nu, Weights: [5]float64{0, 0, 0, 0, 0}, boolFunc: boolFunc}
}

func (n *NeuralNetwork) weightsToString() string {
    weights := make([]string, 5, 5)
    for i := 0; i < 5; i++ {
        if n.Weights[i] < 0 {
            weights[i] = fmt.Sprintf("%.3f", n.Weights[i])
        } else {
            weights[i] = fmt.Sprintf("%.4f", n.Weights[i])
        }
    }
    return strings.Join(weights, ", ")
}

func (n NeuralNetwork) ThresholdFunction(net float64) int {
    if net >= 0 {
        return 1
    }
    return 0
}

func (n NeuralNetwork) ActivationFunction(net float64) int {
    if 0.5*(net/(1+math.Abs(net))+1) >= 0.5 {
        return 1
    }
    return 0
}

func (n NeuralNetwork) DiffActivationFunction(net float64) float64 {
    return 0.5 * (1 / math.Pow(1+math.Abs(net), 2))
}

func (n *NeuralNetwork) CalculateNet(x [5]int) float64 {
    net := 0.0
    for i := 0; i < 5; i++ {
        net += n.Weights[i] * float64(x[i])
    }
    return net
}

func (n *NeuralNetwork) WeightsCorrection(sigma, dfdnet float64, x [5]int) {
    for i := 0; i < 5; i++ {
        n.Weights[i] += n.nu * sigma * dfdnet * float64(x[i])
    }
}

func (n *NeuralNetwork) Study(variablesVector []int) {
    fmt.Printf("%5s %18s %39s %5s", "EPOCH", "FUNCTION", "WEIGHTS", "ERROR\n")
    fmt.Println(strings.Repeat("_", 5), strings.Repeat("_", 18),
        strings.Repeat("_", 39), strings.Repeat("_", 5))
    epoch := 0

    for {
        err := 0
        valuesVector := ""
        var predictedY int
        weightsString := n.weightsToString()

```



```

    for _, j := range variablesVector {
        x := AllVariables[j]
        net := n.CalculateNet(x)
        if n.funcType == 1 {
            predictedY = n.ThresholdFunction(net)
        } else if n.funcType == 2 {
            predictedY = n.ActivationFunction(net)
        }
        if predictedY != n.boolFunc[j] {
            err += 1
        }
        valuesVector += strconv.Itoa(predictedY)
    }
    fmt.Printf("%5d %18s %39s %5d\n", epoch, valuesVector, weightsString, err)
    n.errorsArr = append(n.errorsArr, err)
    n.epochsArr = append(n.errorsArr, epoch)

    for _, j := range variablesVector {
        x := AllVariables[j]
        net := n.CalculateNet(x)
        if n.funcType == 1 {
            predictedY = n.ThresholdFunction(net)
        } else if n.funcType == 2 {
            predictedY = n.ActivationFunction(net)
        }
        var dfdnet float64 = 1
        sigma := float64(n.boolFunc[j] - predictedY)
        n.CalculateNet(x)
        if n.funcType == 2 {
            dfdnet = n.DiffActivationFunction(net)
        }
        n.WeightsCorrection(sigma, dfdnet, x)
    }
    if err == 0 {
        return
    }

    epoch += 1
}

}

func (n NeuralNetwork) Run() bool {
    predicted := make([]int, 0, 16)
    for i := 0; i < 16; i++ {
        x := AllVariables[i]
        net := n.CalculateNet(x)
        if n.funcType == 1 {
            predicted = append(predicted, n.ThresholdFunction(net))
        } else if n.funcType == 2 {
            predicted = append(predicted, n.ActivationFunction(net))
        }
    }
    return Equal(Function, predicted)
}

func (n *NeuralNetwork) StudyWithSet(variablesVector []int) (int, bool) {
    epoch := 0
    for {
        err := 0
        predictedY := 0
        for _, j := range variablesVector {

```

```

        x := AllVariables[j]
        net := n.CalculateNet(x)
        if n.funcType == 1 {
            predictedY = n.ThresholdFunction(net)
        } else if n.funcType == 2 {
            predictedY = n.ActivationFunction(net)
        }
        if predictedY != n.boolFunc[j] {
            err += 1
        }
        var dfdnet float64 = 1
        sigma := float64(n.boolFunc[j] - predictedY)
        n.CalculateNet(x)
        if n.funcType == 2 {
            net := n.CalculateNet(x)
            dfdnet = n.DiffActivationFunction(net)
        }
        n.WeightsCorrection(sigma, dfdnet, x)
    }

    if err == 0 {
        break
    }
    epoch += 1
}
if n.Run() {
    return epoch, true
}
return 0, false

```

}  
*Файл Combinations.go*

```

package main

import (
    "fmt"
    "math"
    "math/bits"
)

var AllVariables = GetVariables()
var Set = []int{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15}
var MinSet = []int{2, 7, 9, 15}

func Combinations(set []int, n int) (subsets [][]int) {
    length := uint(len(set))
    if n > len(set) {
        n = len(set)
    }
    for subsetBits := 1; subsetBits < (1 << length); subsetBits++ {
        if n > 0 && bits.OnesCount(uint(subsetBits)) != n {
            continue
        }

        var subset []int

        for object := uint(0); object < length; object++ {
            if (subsetBits>>object)&1 == 1 {
                subset = append(subset, set[object])
            }
        }
    }
}

```

```

    }
    subsets = append(subsets, subset)
}
return subsets
}

func GetVariables() [16][5]int {
    var variables [16][5]int
    for i := 0; i < 16; i++ {
        variables[i] = [5]int{1, (i / 8) % 2, (i / 4) % 2, (i / 2) % 2, (i / 1) %
2}
    }
    return variables
}

func Equal(a, b []int) bool {
    if len(a) != len(b) {
        return false
    }
    for i := 0; i < len(a); i++ {
        if a[i] != b[i] {
            return false
        }
    }
    return true
}

func GetFastSet(n int) {
    minEpoch := math.MaxInt
    var minValues []int
    subSetN := Combinations(Set, n)
    for _, v := range subSetN {
        nw := NewNeuralNetwork(2, 0.3, Function)
        epoch, ok := nw.StudyWithSet(v)

        if epoch < minEpoch && ok {
            minEpoch = epoch
            minValues = v
        }
    }
    if minValues == nil {
        fmt.Println("cannot study on this small variables n:", n)
        return
    }
    fmt.Println("fast set of size:", n, "epochs:", minEpoch+1, "variables:",
minValues)
}

func FindMinSet() {
    for i := 15; i > 1; i-- {
        GetFastSet(i)
    }
}

```

*Файл Plot.go*

```

package main

import (

```

```

    "gonum.org/v1/plot"
    "gonum.org/v1/plot/plotter"
    "gonum.org/v1/plot/plotutil"
    "gonum.org/v1/plot/vg"
    "math/rand"
)

func Plot(epochs, errs []int, name string) {
    rand.Seed(int64(0))

    p := plot.New()

    p.Title.Text = "Graph"
    p.X.Label.Text = "epochs"
    p.Y.Label.Text = "errors"
    pts := make(plotter.XYs, len(errs))
    for i := range pts {
        if i == 0 {
            pts[i].X = float64(epochs[i])
        } else {
            pts[i].X = pts[i-1].X + float64(epochs[i-1])
        }
        pts[i].Y = float64(errs[i])
    }
    err := plotutil.AddLinePoints(p, "First", pts)
    if err != nil {
        panic(err)
    }

    if err := p.Save(4*vg.Inch, 4*vg.Inch, name); err != nil {
        panic(err)
    }
}

```