Krzysztof Sekuła

Sprawozdanie z projektu nr 3

Celem ćwiczenia było poznanie budowy i działania wielowarstwowych sieci neuronowych poprzez uczenie z użyciem algorytmu wstecznej propagacji błędu rozpoznawania konkretnych liter alfabetu.

Opis budowy algorytmu:

Podobnie jak w przypadku pojedynczego neuronu, główną zaletą sieci neuronowej jest to, że nie musimy "ręcznie" dobierać wag. Możemy te wagi wytrenować, czyli znaleźć ich w przybliżeniu optymalny zestaw za pomocą metody obliczeniowej zwanej wsteczną propagacją błędu. Jest to metoda umożliwiająca modyfikację wag w sieci o architekturze warstwowej we wszystkich jej warstwach.

Ogólny schemat procesu uczenia sieci wygląda następująco:

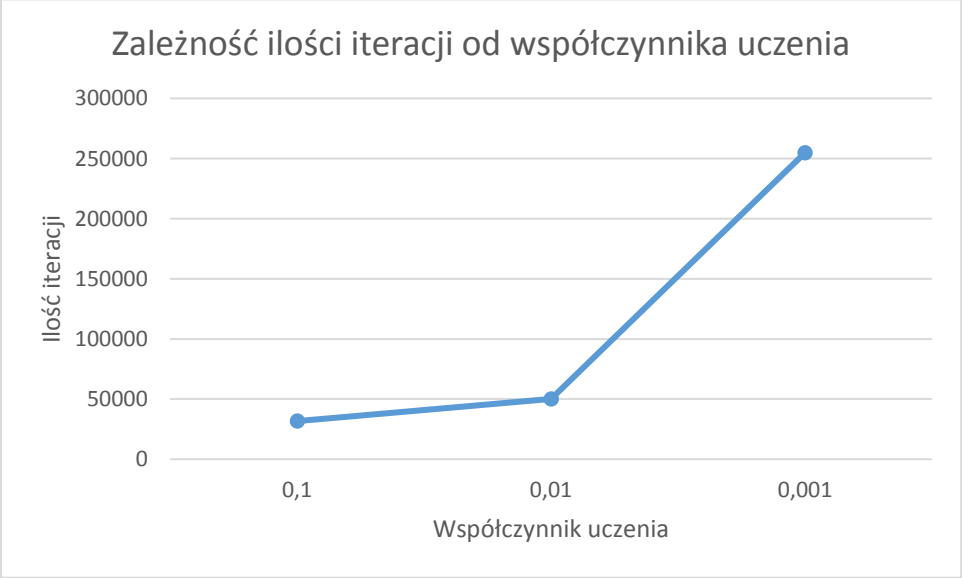
1. Ustalamy topologię sieci, tzn. liczbę warstw, liczbę neuronów w warstwach.
2. Inicjujemy wagi losowo
3. Dla danego wektora uczącego obliczamy odpowiedź sieci warstwa po warstwie
4. Każdy neuron wyjściowy oblicza swój błąd, oparty na różnicy pomiędzy obliczoną odpowiedzią *y* oraz poprawną odpowiedzią *t*
5. Błędy propagowane są do wcześniejszych warstw
6. Każdy neuron ,również w warstwach ukrytych modyfikuje wagi na podstawie wartości błędu i wielkości przetwarzanych w tym kroku sygnałów
7. Powtarzamy od punktu 3 dla kolejnych wektorów uczących. Gdy wszystkie wektory zostaną użyte, losowo zmieniamy ich kolejność i zaczynamy wykorzystywać powtórnie.
8. Zatrzymujemy się, gdy średni błąd na danych treningowych przestanie maleć. Możemy też co jakiś czas testować sieć na specjalnej puli nieużywanych do treningu próbek testowych i kończyć trenowanie, gdy błąd przestanie maleć.

Zestawienie wyników:

Jako dane uczące wykorzystałem 20 liter, 10 dużych oraz 10 małych, w postaci dwuwymiarowej tablicy 4x5. Testy przeprowadziłem 10 razy dla trzech różnych współczynników uczenia.

LP	Współczynnik uczenia	0,1	0,01	0,001
1	ilość iteracji:	42364	30584	245719
2	ilość iteracji:	23464	38358	167344
3	ilość iteracji:	43603	87876	345743
4	ilość iteracji:	31580	80608	261660
5	ilość iteracji:	13438	29538	215930
6	ilość iteracji:	45567	53568	156420
7	ilość iteracji:	51409	83095	397260
8	ilość iteracji:	25026	35864	241184
9	ilość iteracji:	21772	39555	318800
10	ilość iteracji:	18426	21944	197165

Współczynnik uczenia	Średnia ilość iteracji
0,1	31664,9
0,01	50099
0,001	254722,5



Przykładowy wynik dla współczynnika uczenia 0.01:

Wejsciowy wektor: 8
-0,07, -0,07, -0,07, -0,06, -0,07, -0,07, -0,06, -0,06, 0,94, -0,07, -0,06, -0,06, -0,05, -0,07, -0,07, -0,05, -0,08, -0,07, -0,05, -0,08,

Wynik oczekiwany:
0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,

Wnioski:

Z otrzymanych wyników możemy wnioskować że współczynnik uczenia wpływa na szybkość uczenia jak i na poprawność wyników. Dla współczynnika 0.1 ilość iteracji potrzebnych do nauczenia sieci była najmniejsza. Im mniejszy współczynnik tym ilość iteracji rosła, natomiast zmniejszał się błąd.

Listing kodu:

```
public class Main {
    private static int n = 20;
    private static double e = 2.718281828459045235;

    public static void main(String[] args) {

        Random rand = new Random();
        double[][] all = new double[n][n];
        double[][] answers = new double[20][5];
        double[] firstLayerAnswers = new double[n];
        double[] secondLayerAnswers = new double[5];
        double[] thirdLayerAnswers = new double[n];
        double[] errors;
        double learningRate = 0.001;
        createVectors(all);
        setAnswerVectors(answers);
        double error = 0;
        int ni = 0;
        Perc1[] firstLayer = new Perc1[n];
        Perc2[] secondLayer = new Perc2[5];
        Perc2[] thirdLayer = new Perc2[n];
        java.text.DecimalFormat df = new java.text.DecimalFormat();
        df.setMaximumFractionDigits(2);
        df.setMinimumFractionDigits(2);

        for (int i = 0; i < 20; i++) {
            firstLayer[i] = new Perc1();
```

```
        }
        for (int i = 0; i < 5; i++) {
            secondLayer[i] = new Perc2(20);
        }
        for (int i = 0; i < 20; i++) {
            thirdLayer[i] = new Perc2(5);
        }



        do {

            int which = setVectorsFirstLayer(firstLayer, all, rand);
            for (int i = 0; i < 20; i++) {
                firstLayerAnswers[i] = 1 / (1 + Math.pow(e, -
firstLayer[i].sum()));
            }
            setVectorsSecondLayer(firstLayerAnswers, secondLayer);
            for (int i = 0; i < 5; i++) {
                secondLayerAnswers[i] = 1 / (1 + Math.pow(e, -
secondLayer[i].sum()));
            }
            setVectorsThirdLayer(secondLayerAnswers, thirdLayer);
            for (int i = 0; i < 20; i++) {
                thirdLayerAnswers[i] = 1 / (1 + Math.pow(e, -
thirdLayer[i].sum()));
            }


            double[] macierzBledowW1 = new double[20];
            double[] macierzBledowW2 = new double[5];
            errors = new double[20];

            for (int i = 0; i < 20; i++) {
                errors[i] = answers[which][i] - thirdLayerAnswers[i];
            }

            for (int j = 0; j < 20; j++) {
                for (int i = 0; i < 5; i++) {
                    macierzBledowW2[i] = +errors[j] *
thirdLayer[j].getWeights()[i];
                }
            }
            for (int i = 0; i < 5; i++) {
                for (int j = 0; j < 20; j++) {
                    macierzBledowW1[j] += macierzBledowW2[i] *
secondLayer[i].getWeights()[j];
                }
            }



            for (int i = 0; i < 20; i++) {
                double newWeight = firstLayer[i].getWeight() + learningRate
* macierzBledowW1[i] * all[which][i]
                        * (Math.pow(e, firstLayer[i].getSuma()) /
Math.pow((Math.pow(e, firstLayer[i].getSuma()) + 1), 2));
                firstLayer[i].updateWeight(newWeight);
            }
            for (int i = 0; i < 5; i++) {
                for (int j = 0; j < 20; j++) {
```

```
                    secondLayer[i].getWeights()[j] =
secondLayer[i].getWeights()[j] + learningRate * macierzBledowW2[i]
                            * (Math.pow(e, secondLayer[i].getSuma()) /
Math.pow((Math.pow(e, secondLayer[i].getSuma()) + 1), 2))
                            * firstLayerAnswers[j];
                }
            }
            for (int i = 0; i < 20; i++) {
                for (int j = 0; j < 5; j++) {
                    double newWeight = thirdLayer[i].getWeights()[j] +
learningRate * errors[i] * secondLayerAnswers[j]
                            * (Math.pow(e, thirdLayer[i].getSuma()) /
Math.pow((Math.pow(e, thirdLayer[i].getSuma()) + 1), 2));
                    thirdLayer[i].setWeight(j, newWeight);
                }
            }

            error = 0;
            for (int i = 0; i < 20; i++) {
                error += errors[i];

            }
            ni++;
        } while (error > 0.00001 || error < -0.00001);

        System.out.println("Zakonczono proces uczenia po: " + ni + "
iteracjach");


        for (int z = 0; z < 20; z++) {
            int which = setVectorsFirstLayer(firstLayer, all, rand);
            for (int i = 0; i < 20; i++) {
                firstLayerAnswers[i] = 1 / (1 + Math.pow(e, -
firstLayer[i].sum()));
            }
            setVectorsSecondLayer(firstLayerAnswers, secondLayer);
            for (int i = 0; i < 5; i++) {
                secondLayerAnswers[i] = 1 / (1 + Math.pow(e, -
secondLayer[i].sum()));
            }
            setVectorsThirdLayer(secondLayerAnswers, thirdLayer);
            for (int i = 0; i < 20; i++) {
                thirdLayerAnswers[i] = 1 / (1 + Math.pow(e, -
thirdLayer[i].sum()));
            }

            for (int i = 0; i < 20; i++) {
                errors[i] = answers[which][i] - thirdLayerAnswers[i];
            }
            for (int i = 0; i < 20; i++) {
                System.out.print(df.format(errors[i]) + ", ");
            }
            System.out.print("\nWynik oczekiwany:\n");
            for (int i = 0; i < 20; i++) {
                System.out.print(answers[which][i] + ", ");
            }

            System.out.println("");
        }

    }
```

```
    private static void setVectorsThirdLayer(double[] secondLayerAnswers,
Perc2[] thirdLayer) {

        for (int i = 0; i < 20; i++) {
            thirdLayer[i].setVector(secondLayerAnswers);
        }
    }

    private static void setAnswerVectors(double[][] answers) {

        double[] row = {1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0};
        answers[0] = row;
        double[] row1 = {0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0};
        answers[1] = row1;
        double[] row2 = {0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0};
        answers[2] = row2;
        double[] row3 = {0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0};
        answers[3] = row3;
        double[] row4 = {0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0};
        answers[4] = row4;
        double[] row5 = {0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0};
        answers[5] = row5;
        double[] row6 = {0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0};
        answers[6] = row6;
        double[] row7 = {0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0};
        answers[7] = row7;
        double[] row8 = {0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0};
        answers[8] = row8;
        double[] row9 = {0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0,
0, 0, 0};
        answers[9] = row9;
        double[] row10 = {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0,
0, 0, 0, 0};
        answers[10] = row10;
        double[] row11 = {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0,
0, 0, 0, 0};
        answers[11] = row11;
        double[] row12 = {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0,
0, 0, 0, 0};
        answers[12] = row12;
        double[] row13 = {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0,
0, 0, 0, 0};
        answers[13] = row13;
        double[] row14 = {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0,
0, 0, 0, 0};
        answers[14] = row14;
        double[] row15 = {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1,
0, 0, 0, 0};
        answers[15] = row15;
        double[] row16 = {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
1, 0, 0, 0};
```

```
        answers[16] = row16;
        double[] row17 = {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 1, 0, 0};
        answers[17] = row17;
        double[] row18 = {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 1, 0};
        answers[18] = row18;
        double[] row19 = {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 1};
        answers[19] = row19;
    }

    private static void setVectorsSecondLayer(double[] firstLayerAnswers,
Perc2[] secondLayer) {

        for (int i = 0; i < 5; i++) {
            secondLayer[i].setVector(firstLayerAnswers);
        }
    }

    public static void createWectors(double[][] all) {
        double[] A = {0, 1, 1, 0, 1, 0, 0, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 0,
0, 1};
        double[] B = {1, 1, 1, 0, 1, 0, 0, 1, 1, 1, 1, 0, 1, 0, 0, 1, 1, 1,
1, 0};
        double[] C = {0, 1, 1, 0, 1, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 1, 0, 1,
1, 0};
        double[] D = {1, 1, 1, 0, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 1,
1, 0};
        double[] E = {1, 1, 1, 1, 1, 0, 0, 0, 1, 1, 1, 1, 1, 0, 0, 0, 1, 1,
1, 1};
        double[] F = {1, 1, 1, 1, 1, 0, 0, 0, 1, 1, 1, 0, 1, 0, 0, 0, 1, 0,
0, 0};
        double[] G = {0, 1, 1, 0, 1, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 1, 0, 1,
1, 0};
        double[] H = {1, 0, 0, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 0,
0, 1};
        double[] I = {0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1,
1, 0};
        double[] J = {0, 1, 1, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 1, 1,
1, 0};
        double[] a = {0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 1, 0, 0, 1,
1, 0};
        double[] b = {0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 1, 0, 1, 0, 0, 1,
0, 0};
        double[] c = {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0,
1, 0};
        double[] d = {0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 0, 1, 0, 1, 0, 0, 1,
1, 0};
        double[] e = {0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 1,
1, 0};
        double[] f = {0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 1, 1, 0, 0, 1, 0,
0, 0};
        double[] g = {0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 1,
0, 0};
        double[] h = {0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 1, 1, 0, 1, 0,
1, 0};
        double[] i = {0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1,
0, 0};
        double[] j = {0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1,
0, 0};
```

```
        all[0] = A;
        all[1] = B;
        all[2] = C;
        all[3] = D;
        all[4] = E;
        all[5] = F;
        all[6] = G;
        all[7] = H;
        all[8] = I;
        all[9] = J;
        all[10] = a;
        all[11] = b;
        all[12] = c;
        all[13] = d;
        all[14] = e;
        all[15] = f;
        all[16] = g;
        all[17] = h;
        all[18] = i;
        all[19] = j;
    }

    public static int setVectorsFirstLayer(Perc1[] firstLayer, double[][]
all, Random rand) {
        int wejscie = rand.nextInt(20);
        System.out.println("Wejsciowy wektor: " + wejscie);
        for (int i = 0; i < 20; i++) {
            double w1 = all[wejscie][i];
            firstLayer[i].setVariable(w1);
        }

        return wejscie;
    }
}
```

Źródła:

http://edu.pjwstk.edu.pl/wyklady/nai/scb/wyklad3/w3.htm