ДОДАТОК А

Вихідний код програми

```
class AdaptivePReLu : NeuralNetwork
  {
    protected List<double> φR;
    protected List<double> φL;
    public AdaptivePReLu(List<double> φR, List<double> φL,List<double> w) : base(w)
    {
      this.φR = φR;
      this.φL = φL;
    }
    public static double ActivationPrelu(double value, List<double> φR, List<double> φL)
    {

      if (value >= 0)
      {
        return φR[φR.Count - 1] * value;
      }
      else
      {
        return φL[φL.Count - 1] * value;
      }
    }
    public static void ParametersφChanger(double η, double d, double y, List<double> input, List<double> weight, List<double> φR, List<double> φL)
    {
      if (Adder(weight, input) >= 0)
```

```
                {
            //    φR.Add(φR[φR.Count - 1] +  (d-y)/ Adder(weight, input)) ;
                  φR.Add(φR[φR.Count - 1] + ((d - y) * Adder(weight, input)) / (1
+ Math.Pow(Adder(weight, input), 2)));//after optimization
                }
                else
                {
                  φL.Add(φL[φL.Count - 1] + ((d-y)       *Adder(weight,
input))/(1+Math.Pow(Adder(weight, input), 2)));
                }
            }
            public static void ParametersChangerPrelu(double η, double y, double
d, List<double> Δw, List<double> input,  List<double> weight, List<double>
φR, List<double> φL)
            {
                for (int i = 0; i < Δw.ToArray().Length; i++)
                {
                    if (Adder(weight, input) >=0)
                    {
                        // Δw[i] = (d - y) * input[i] * φR[φR.Count - 1];
                     Δw[i] = ((d - Adder(weight, input) * φR[φR.Count - 1])*
φR[φR.Count - 1])/(Math.Pow(φR[φR.Count - 1],2)* input.Zip(input, (x1, x2) =>
x1 * x2).Sum());//after optimazation
                    }
                    else
                    {
                     //   Δw[i] =  (d - y) * input[i] * φL[φL.Count - 1];
                      Δw[i] = ((d - Adder(weight, input) * φL[φL.Count - 1]) *
φL[φL.Count - 1]) / (Math.Pow(φL[φL.Count - 1], 2) * input.Zip(input, (x1, x2)
=> x1 * x2).Sum());
```

```
            }
        }
    }
    public List<double> AdPreluLearning(List<double> weights, List<Input> range, List<double> Δw, List<double> mseAdPrelu, double mse,List<double >epochAdPrelu)
    {
        for (int k = 0; k < range.Count; k++)
        {
            Input value = range[k];
            double y = ActivationPrelu(Adder(weights, value.X), φR,φL);
            double d = value.Y;
            AdaptivePReLu.ParametersφChanger( η, d, y, value.X, weights, φR, φL);
            y= ActivationPrelu(Adder(weights, value.X), φR, φL);
            AdaptivePReLu.ParametersChangerPrelu( η, y, d, Δw, value.X, weights, φR, φL);
            NeuralNetwork.Weights(weights, Δw);
             mse = NeuralNetwork.Mse(k, d, y, mse);
            mseAdPrelu.Add(mse);
            epochAdPrelu.Add(k);
        }
        return epochAdPrelu;
    }
}
```

ДОДАТОК Б

Відомість атестаційної роботи

Позначення	Найменування	Дод. відомості
	Текстові документи	
1.	Пояснювальна записка	__ с.
	Інші документи	
2.	Презентаційні матеріали	__ плакатів

.	.	Прізвище та ініціали.	Підп.	Дата			
Розробив		Слепанська В.Д.			Адаптивна поліноміальна активаційна функція в глибинних нейронних мережах та алгоритм її навчання	Шифр групи	Код напр./спец.
Перевірив		Чала Л.Е.				СШІм-19-2	122
Н.контр.		Малєєва І.А.				ХНУРЕ кафедра ШІ	
Затв.		Філатов В.О.					