```csharp
using System;
using System.Collections.Generic;
using System.Linq;

namespace Neural_Network
{
    class Trainer
    {
        private Perceptron perceptron;
        private Random r = new Random();
        public List<TrainingInstance> training;

        public Trainer() {
            // {1,10,1} -> 1 neuron in input layer, 10 in hidden layer, 1 in output layer
            List<int> numberOfNeurons = new List<int>(new int[] {1,10,1});
            // random init weights in -0.5 0.5
            perceptron = new Perceptron(numberOfNeurons, -0.5, 0.5);
            createTrainingSet();
        }

        private double f(double x) { // the function to be approximated
            return (Math.Cos(x / 3) + Math.Sin(10 / (Math.Abs(x) + 0.1)) + 0.1 * x);
        }

        public void createTrainingSet() {
            training = new List<TrainingInstance>();
            for (int i = 0; i < 1001; ++i) {
                training.Add(new TrainingInstance(new List<double>(new double[] { -10.0 + i * 20.0 / 1001.0 }), f(-10.0 + i * 20.0 / 1001.0)));
            }
        }

        public List<List<double>> trainingResults(){
            List<List<double>> results = new List<List<double>>();
            foreach (TrainingInstance ti in training) {
                results.Add(perceptron.feedForward(ti));
            }
            return results;
        }

        public void trainOutputLayer(){
            var permutated = training.OrderBy(item => r.Next());
            foreach (TrainingInstance ti in permutated) {
                perceptron.feedForward(ti);
                foreach (Neuron n in perceptron.outputLayer.neurons){
                    n.learn(ti);
                }
            }
        }

        public void trainHiddenLayer() {
            var permutated = training.OrderBy(item => r.Next());
            foreach (TrainingInstance ti in permutated) {
                perceptron.feedForward(ti);
                foreach(Neuron n in perceptron.hiddenLayers[0].neurons) {
                    n.learn(ti);
                }
            }
        }

        public double meanSquareError() {
            double d=0.0;
            foreach (TrainingInstance ti in training) {
                perceptron.feedForward(ti);
                d+=Math.Pow(perceptron.outputLayer.neurons[0].getCurrentOutputValue()-ti.expectedOutput, 2.0);
            }
            d /= (2*training.Count);
            return d;
        }
    }
}
```