Simple Perceptron (binary linear classifier)

Zbigniew Dendzik
Institute of Physics, University of Silesia

2025

Introduction

The perceptron is one of the simplest neural network models and at the same time an example of a binary linear classifier. In the basic version:

- we represent an example as a feature vector $(x_1, \ldots, x_n)$,
- the perceptron stores a weight vector $(w_1, \ldots, w_n)$ and a bias b,
- for a given point we compute a weighted sum $s = w_1x_1 + \ldots + w_nx_n + b$,
- if $s \geq 0$ we predict class $+1$, otherwise class -1.

Training is iterative: we go through the training set and whenever the perceptron misclassifies an example, we update the weights in a direction that should fix this mistake. This is a simple example of supervised learning.

In this exercise set we will:

- implement a perceptron for 2D data,
- train it to separate two linearly separable classes,
- test its behaviour and then observe what happens for non-linearly separable data (e.g. XOR).

Example I – point and training data representation

We start with a simple `Point2DLabel` class representing a 2D point with label -1 or $+1$.

```
class Point2DLabel
{
   private double x;
   private double y;
   private int label; //-1 or +1

   public Point2DLabel(double x, double y, int label)
   {
      this.x=x;
      this.y=y;
```

```
        this.label=label;
    }

    public double getX()
    {
        return x;
    }

    public double getY()
    {
        return y;
    }

    public int getLabel()
    {
        return label;
    }

    public String toString()
    {
        return "Point2DLabel("+x+", "+y+", "+label+")";
    }
}
```

Listing 1: Point2DLabel.java

We also define a simple training set container:

```
import java.util.*;

class TrainingSet2D
{
    private List<Point2DLabel> points;

    public TrainingSet2D()
    {
        points=new ArrayList<Point2DLabel>();
    }

    public void add(Point2DLabel p)
    {
        points.add(p);
    }

    public List<Point2DLabel> getPoints()
    {
        return points;
    }

    public int size()
    {
        return points.size();
    }
```

```
}
```

Listing 2: TrainingSet2D.java

Ex. 4.1

Write a program `TestTrainingSet2D` that:

- creates a `TrainingSet2D`,
- adds several points with labels -1 and $+1$, e.g. one side of the line $y = x$ labelled -1, the other side $+1$,
- prints all points to the screen.

```
public class TestTrainingSet2D
{
   public static void main(String[] args)
   {
      TrainingSet2D set=new TrainingSet2D();

      set.add(new Point2DLabel(1,0,-1));
      set.add(new Point2DLabel(2,0,-1));
      set.add(new Point2DLabel(0,1,+1));
      set.add(new Point2DLabel(0,2,+1));
      set.add(new Point2DLabel(2,3,+1));
      set.add(new Point2DLabel(3,1,-1));

      for(Point2DLabel p : set.getPoints())
      {
         System.out.println(p);
      }
   }
}
```

Listing 3: TestTrainingSet2D.java – skeleton

Example II – perceptron definition

We now define a `Perceptron2D` class with weights (w_1, w_2), bias b, activation function (sign) and learning rule.

```
class Perceptron2D
{
   private double w1;
   private double w2;
   private double bias;
   private double learningRate;

   public Perceptron2D(double learningRate)
   {
```

```
10        this.w1=0.0;
11        this.w2=0.0;
12        this.bias=0.0;
13        this.learningRate=learningRate;
14    }
15
16    public double sum(Point2DLabel p)
17    {
18        return w1*p.getX() + w2*p.getY() + bias;
19    }
20
21    public int predict(Point2DLabel p)
22    {
23        double s=sum(p);
24        if(s>=0)
25            return +1;
26        else
27            return -1;
28    }
29
30    public void trainOnExample(Point2DLabel p)
31    {
32        int y=p.getLabel();
33        int yPred=predict(p);
34        int error=y - yPred;
35
36        if(error!=0)
37        {
38            w1 = w1 + learningRate * error * p.getX();
39            w2 = w2 + learningRate * error * p.getY();
40            bias = bias + learningRate * error * 1.0;
41        }
42    }
43
44    public double getW1() { return w1; }
45    public double getW2() { return w2; }
46    public double getBias() { return bias; }
47
48    public String toString()
49    {
50        return "Perceptron2D[w1="+w1+", w2="+w2+
51               ", bias="+bias+"]";
52    }
53}
```

Listing 4: Perceptron2D.java – skeleton

Ex. 4.2

Write a program `TestPerceptron2D` that:

- creates a perceptron with a chosen learning rate (e.g. 0.1),

- creates a simple training set (as in Ex. 4.1),
- loops over the training set for several epochs, calling `trainOnExample` for each point,
- after each epoch prints the weights `w1`, `w2` and `bias`.

```
public class TestPerceptron2D
{
   public static void main(String[] args)
   {
      TrainingSet2D set=new TrainingSet2D();
      set.add(new Point2DLabel(1,0,-1));
      set.add(new Point2DLabel(2,0,-1));
      set.add(new Point2DLabel(0,1,+1));
      set.add(new Point2DLabel(0,2,+1));
      set.add(new Point2DLabel(2,3,+1));
      set.add(new Point2DLabel(3,1,-1));

      Perceptron2D p=new Perceptron2D(0.1);

      int epochs=10;
      for(int e=0;e<epochs;e++)
      {
         for(Point2DLabel pt : set.getPoints())
         {
            p.trainOnExample(pt);
         }
         System.out.println("Epoch "+e+": "+p);
      }
   }
}
```

Listing 5: TestPerceptron2D.java – skeleton

Example III – training until convergence

We extend `Perceptron2D` with:

- a method counting misclassified examples on a training set,
- a method performing training for up to a given number of epochs, stopping early if there are no errors.

```
   public int countErrors(TrainingSet2D set)
   {
      int errors=0;
      for(Point2DLabel p : set.getPoints())
      {
         int y=p.getLabel();
         int yPred=predict(p);
         if(y!=yPred)
```

```
            errors++;
        }
        return errors;
    }

    public void train(TrainingSet2D set,
                      int maxEpochs)
    {
        for(int e=0;e<maxEpochs;e++)
        {
            for(Point2DLabel p : set.getPoints())
            {
                trainOnExample(p);
            }
            int errors=countErrors(set);
            System.out.println("Epoch "+e+
                " errors="+errors+" "+this);
            if(errors==0)
                break;
        }
    }
```

Listing 6: Perceptron2D.java – extension

Ex. 4.3

Modify `TestPerceptron2D` so that:

- it uses the `train(set,maxEpochs)` method instead of a manual epoch loop,
- after training it tests the perceptron on all training points and prints the predicted label and whether it is correct.

```
public class TestPerceptron2D
{
    public static void main(String[] args)
    {
        TrainingSet2D set=new TrainingSet2D();
        set.add(new Point2DLabel(1,0,-1));
        set.add(new Point2DLabel(2,0,-1));
        set.add(new Point2DLabel(0,1,+1));
        set.add(new Point2DLabel(0,2,+1));
        set.add(new Point2DLabel(2,3,+1));
        set.add(new Point2DLabel(3,1,-1));

        Perceptron2D p=new Perceptron2D(0.1);
        p.train(set,50);

        System.out.println("Test on training set:");
        for(Point2DLabel pt : set.getPoints())
        {
```

```
            int yPred=p.predict(pt);
            System.out.println(pt+" pred="+yPred+
               (yPred==pt.getLabel() ? " OK" : " ERROR"));
        }
    }
}
```

Listing 7: TestPerceptron2D.java – variant

Example IV – decision line geometry

A 2D perceptron defines a decision line:

$$w_1 x + w_2 y + b = 0.$$

If $w_2 \neq 0$, we can rewrite it as $y = ax + c$ with:

$$a = -\frac{w_1}{w_2}, \quad c = -\frac{b}{w_2}.$$

We add methods to compute a and c:

```
    public double getA()
    {
        return -w1 / w2;
    }

    public double getC()
    {
        return -bias / w2;
    }
```

Listing 8: Perceptron2D.java – decision line

Ex. 4.4

Extend `TestPerceptron2D` so that after training it:

- prints the decision line in the form $y = ax + c$,
- classifies several new points (not present in the training set) and prints the predicted labels,
- considers what happens for points lying exactly on the decision line.

Example V – non-linearly separable data (XOR)

The perceptron convergence theorem guarantees convergence only for linearly separable data. For a classical XOR pattern:

- $(0, 0)$ and $(1, 1)$ – label -1,
- $(0, 1)$ and $(1, 0)$ – label $+1$,

there is no single straight line separating the two classes.

Ex. 4.5

Build a training set with XOR data, train the perceptron and observe:

- whether the number of errors goes to zero,
- how the weights evolve across epochs.

Think about why a single perceptron cannot learn XOR and how this relates to more complex neural networks (multi-layer perceptrons, MLP) that can solve such tasks.