Assignment 7 - Perceptron

Meta

Author: Parmandeep Chaddha.

Date: Feb 19, 2022.

Objective

1. Demonstrate the linear separability in 2D using a Perceptron.
2. Demonstrate the linear separability in 1D using a Perceptron.
3. Demonstrate the linear separability in 3D using a Perceptron.

Requirements

The requirements for the Perceptron module are:

1. `Random (base)`
2. `LinearAlgebra`
3. `CSV`
4. `DataFrames`
5. `Plots`

Setup

1. An updated `project.toml` file should be downloaded from: **Git Hub Link** or the directory should be cloned.
2. Instll the project.toml file using the instructions:

```
- `activate .` (in the directory where the project.toml is stored)
- `instantiate`
```

```
begin
    using Pkg

    # Change this line to the directory in which the project.toml is stored. If they
are in the same directory, use the commented activation command.
    # Pkg.activate(@__DIR__)
    Pkg.activate("/Users/pchaddha/OneDrive - University of Waterloo/Waterloo -
4B/psych_420_intro_to_computational_neuroscience/compNeuroIntro420/juliapsych420")

end
```

```
Main.workspace#3.Perceptron
```

```
begin
    include(joinpath(@__DIR__, "perceptron.jl"))
end
```

The Perceptron

2D Separable Data

Separe data with 2 input dimensions.

```
data =
▸Dict("labels" ⇒ [-1, -1, -1, 1, -1, 1, 1, 1, 1, ⋯ more ,1], "inputs" ⇒ 500×2 Matrix{Floa
                                                                   -0.84496     -0.
                                                                   -0.667401     0.
                                                                   -0.56701      0.
                                                                    0.395156    -0.
                                                                   -0.315384     0.
                                                                    0.491642    -0.
                                                                    0.00160144   0.
                                                                    ⋮
                                                                    0.0682614   -0.
                                                                    0.87829     -0.
                                                                    0.885397     0.
                                                                   -0.823942    -0.
                                                                   -0.313782     0.
                                                                    0.172055    -0.
```

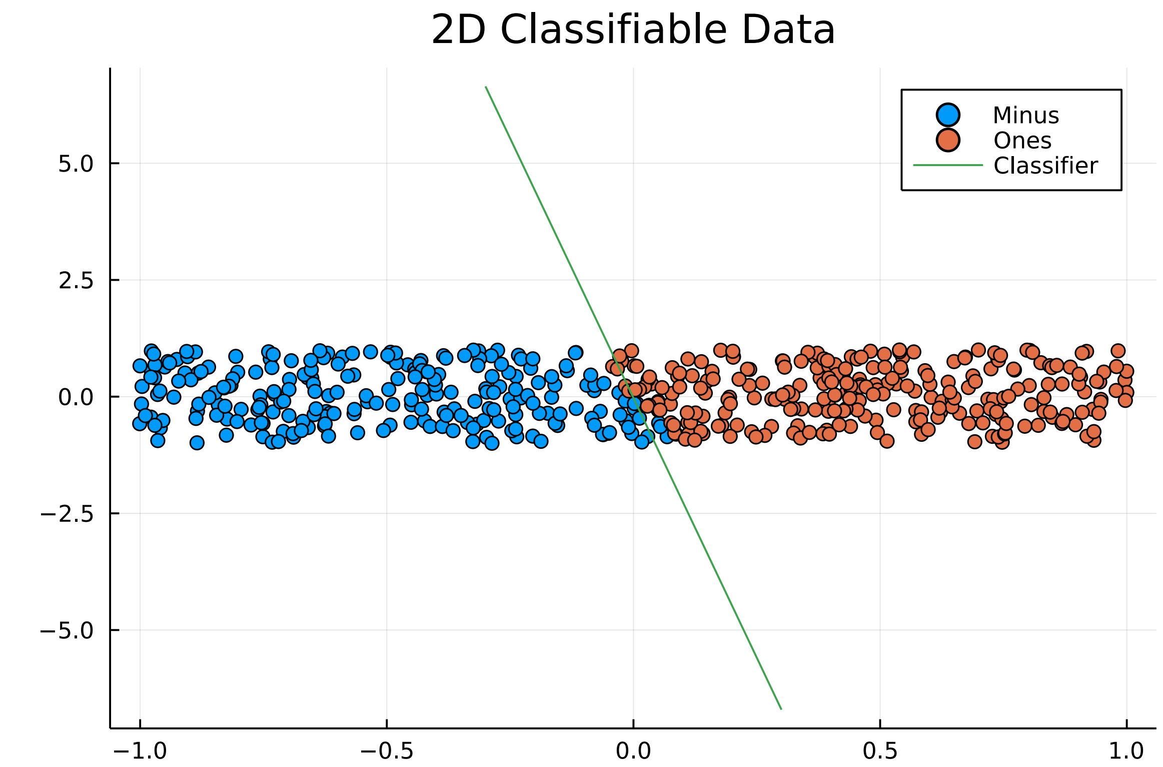
```
data = Perceptron.readSavedData()
```

```
▾Main.workspace#3.Perceptron.PerceptronNeuron(
    bias = 1.0
    biasWeight = 0.029874075060453O3
    learningRate = 0.0009118819655545162
    weights = ▸[1.50239, 0.06751]
    oldWeights = ▸[[0.44, -0.03], [2.25318, 0.209337], [1.43945, -0.146487], [1.5105, 0.24
    accuracy = ▸[0.704, 0.914, 0.948, 0.968, 0.98, 0.982, 0.976, 0.982, 0.982, 0.982]
    error = ▸[527.398, 255.158, 221.284, 218.071, 214.125, 213.545, 212.616, 212.559, 212.
)
```

```
begin
    p = Perceptron.initalizeNeuron(2, learningRate=1.0, bias=1.0)
    Perceptron.trainNeuron(data, p, minAccuracy=0.99, maxIterations=10);
    p
end
```

```
▸(212.54, 0.984)
```

```
loss, accuracy = Perceptron.runEpoch(data, p, withUpdate=false)
```



```
Perceptron.plotClassifier(data, p)
```

1D Separable Data

Let's see if the perceptron can work with 1D separable data.

```
data1d =
▸Dict("labels" => [1, -1, -1, -1, 1, 1, 1, 1, 1, ⋯ more ,1], "inputs" => 500×1 Matrix{Float
                                                                            -0.267067
                                                                             0.440841
                                                                             0.392793
                                                                             0.338138
                                                                            -0.00860861
                                                                             0.161962
                                                                            -0.535035
                                                                             ⋮
                                                                             0.754354
                                                                             0.47978
                                                                             0.606707
                                                                             0.514715
                                                                            -0.986386
                                                                             0.186086
```

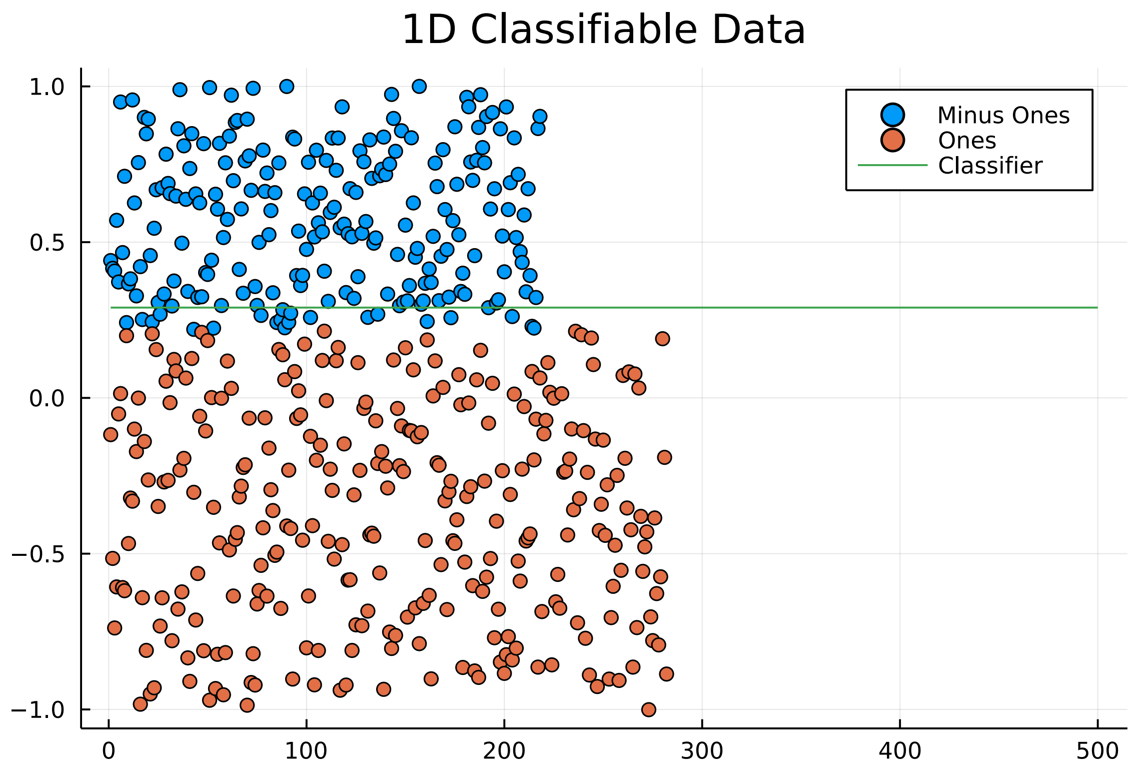
```
data1d = Perceptron.readSavedData(which="1D")
```

```
▾Main.workspace#3.Perceptron.PerceptronNeuron(
    bias = -1.0
    biasWeight = -0.1921832202330019
    learningRate = 0.0009118819655545162
    weights = ▸[-1.50885]
    oldWeights = ▸[[-0.37], [-1.49007], [-1.71493], [-1.39689], [-1.32311], [-1.49694], [-

    accuracy = ▸[0.784, 0.91, 0.946, 0.954, 0.958, 0.968, 0.956, 0.964, 0.964, ⋯ more ,0.9

    error = ▸[350.527, 250.302, 232.233, 224.052, 221.582, 221.744, 220.458, 220.498, 220.
)
```

```
begin
    p1 = Perceptron.initalizeNeuron(1, learningRate=1.0, bias=-1.0)
    Perceptron.trainNeuron(data1d, p1, minAccuracy=0.99, maxIterations=100);
    p1
end
```



```
Perceptron.plotClassifier1D(data1d, p1)
```

3D Separable Data

Let's see if the perceptron can separate data in three dimensions

```
md"
### 3D Separable Data
Let's see if the perceptron can separate data in three dimensions
"
```

```
data3d =
▸Dict("labels" ⇒ [1, 1, -1, 1, 1, 1, 1, 1, -1, ⋯ more ,1], "inputs" ⇒ 500×3 Matrix{Float6
                                                                      -0.0697418   -0.178
                                                                       0.564339     0.594
                                                                      -0.680208    -0.040
                                                                       0.957675     0.526
                                                                      -0.389334    -0.894
                                                                       0.577947     0.222
                                                                      -0.0090054   -0.093
                                                                       ⋮
                                                                       0.503102     0.142
                                                                       0.21693     -0.847
                                                                      -0.479788    -0.064
                                                                      -0.693916    -0.507
                                                                      -0.975085     0.003
                                                                       0.889334    -0.028
```

```
data3d = Perceptron.readSavedData(which="3D")
```

```
▾Main.workspace#3.Perceptron.PerceptronNeuron(
    bias = -1.0
    biasWeight = 0.007391906256739234
    learningRate = 0.0009118819655545162
    weights = ▸[1.02225, -0.576085, 0.752304]

    oldWeights = ▸[[-0.44], [43.4012, 165.033, -239.151], [1.22921, -0.174378, 1.3584], [0

    accuracy = ▸[0.512, 0.816, 0.92, 0.956, 0.966, 0.972, 0.97, 0.974, 0.976, ⋯ more ,0.97

    error = ▸[50967.5, 2558.73, 264.653, 250.233, 248.781, 247.087, 246.353, 246.085, 246.
)
```

```
begin
    p3 = Perceptron.initalizeNeuron(1, learningRate=1.0, bias=-1.0)
    Perceptron.trainNeuron(data3d, p3, minAccuracy=0.99, maxIterations=100);
    p3
end
```

Conclusion

The Perceptron is a fairly good classifier of liner data in both 1D, 2D, and 3D domains. However, depending on the `margin` of the linear classification, it may not converge at a rapid speed with `online` training. However, the accuracy in all cases is above 0.96:

1. 0.96 in 1D
2. 0.98 in 2D
3. 0.97 in 3D

These are well within acceptable range, especially for such a simple classifier.