Chapter 10: Widrow-Hoff Learning

Brandon Morgan

1/18/2021

E10.10

We are wanting to classify two groups of patterns:

Class 1

$$p_1^T = \begin{bmatrix} 2 \\ -4 \end{bmatrix}, t_1 = 1, \theta_1 = 0.25$$

$$p_2^T = \begin{bmatrix} 4 \\ 2 \end{bmatrix}, t_2 = 1, \theta_1 = 0.50$$

Class 2

$$p_3^T = \begin{bmatrix} -4 & 4 \end{bmatrix}, t_3 = -1, \theta_3 = 0.25$$

Where θ_i denotes the probability of the input vector occuring.

1

This part was skipped as I was unable to create a diagram.

3

The maximum stable learning rate is built off the Hessian matrix, H. or the Correlation matrix, R. Our performance index can be written as $F(x) = c - 2x^T h + x^T R x$, where $c = E[t^2]$, $R = E[zz^T]$ and $h = E[tz]$.

$$c = E[t^2] = (1)^2(0.25) + (1)^2(0.5) + (-1)^2(0.25) = 1$$

$$h = E[tz] = 0.25(1)\begin{bmatrix} 2 \\ -4 \end{bmatrix} + 0.5(1)\begin{bmatrix} 4 \\ 2 \end{bmatrix} + 0.25(-1)\begin{bmatrix} -4 \\ 4 \end{bmatrix} = \begin{bmatrix} 3.5 \\ -1 \end{bmatrix}$$

$$R = E[zz^T] = 0.25\begin{bmatrix} 2 \\ -4 \end{bmatrix}\begin{bmatrix} 2 & -4 \end{bmatrix} + 0.5\begin{bmatrix} 4 \\ 2 \end{bmatrix}\begin{bmatrix} 4 & 2 \end{bmatrix} + 0.25\begin{bmatrix} -4 \\ 4 \end{bmatrix}\begin{bmatrix} -4 & 4 \end{bmatrix} = \begin{bmatrix} 13 & -2 \\ -2 & 10 \end{bmatrix}$$

Thus, our optimal weight vector with no bias is:

$$x^* = \begin{bmatrix} 13 & -2 \\ -2 & 10 \end{bmatrix}^{-1} \begin{bmatrix} 3.5 \\ -1 \end{bmatrix} = \begin{bmatrix} 0.26190 \\ -0.04761 \end{bmatrix}$$

The maximum stable learning rate is given by $\alpha < \frac{2}{\lambda_{max}}$, where λ_{max} denotes the largest eigen value of the Hessian matrix, $H = 2R$; or, $\alpha < \frac{1}{\lambda_{max}}$, where λ_{max} denotes the largest eigen value of the correlation matrix, R.

```r
R = matrix(c(13, -2, -2, 10), ncol=2)
H = 2*R
eigen(R)
```

```
## eigen() decomposition
## $values
## [1] 14  9
##
## $vectors
##            [,1]       [,2]
## [1,] -0.8944272 -0.4472136
## [2,]  0.4472136 -0.8944272
```

```r
eigen(H)
```

```
## eigen() decomposition
## $values
## [1] 28 18
##
## $vectors
##            [,1]       [,2]
## [1,] -0.8944272 -0.4472136
## [2,]  0.4472136 -0.8944272
```

For R, the largest eigen value is 14; thus, $\alpha < 1/14 = 0.0714$. For H, the largest eigen value is 28; thus, $\alpha < 2/28 = 0.0714$.

3

```r
LMS = function(init, bias, learning_rate, input, expect, maxIter, tol, mask, useBias) {

  n = ncol(input)
  w = init
  b = bias

  for(i in 1:maxIter) { # loop over iterations

    if(!mask) { # if the user does want to print out statements
      print(sprintf("     Iteration %d", i))
    }

    for(j in 1:n) { # loop over input vectors
```

```r
    if(useBias) { # does the user want to include a bias
      a = w%*%input[,j]+b
    }
    else {
      a = w%*%input[,j]
    }


    e = expect[j] - a

    w = w + 2*learning_rate*e%*%t(input[,j])

    if(useBias) {
      b = b + 2*learning_rate*e
    }

    if(!mask) { # if the user does want to print out statements
      print(sprintf("INPUT VECTOR: %d", j))
      print("a value: ")
      print(a)
      print("error value: ")
      print(e)
      print("new weight value: ")
      print(w)
      print("new bias value: ")
      print(b)
    }
  }


  error = 0

  for(j in 1:n) {

    if(useBias) { # does the user want to include a bias
      a = w%*%input[,j]+b
    }
    else {
      a = w%*%input[,j]
    }

    e = expect[j] - a

    error = error + e^2

  }

  # if the square root of the sum of the square
  # errors is greater than the tol, then algo
  # has not converged
  converged = 1
  if(sqrt(error)>tol) {
    converged = 0
```

```
    }

    if(converged) {
      print("    Algorithm CONVERGED:")
      print("Current weight is: ")
      print(w)
      if(useBias) { # does the user want to include a bias
        print("Current bias is: ")
        print(b)
      }

      print(sprintf("Iterations taken: %d", i))
      if(useBias) { # does the user want to include a bias
        return(list(w=w, b=b))
      }
      return(list(w=w))
    }


  }

  error = 0

  for(j in 1:n) {

    a = w%*%input[,j]+b

    e = expect[j] - a

    error = error + abs(e)

  }

  print("MAXIMUM ITERATIONS REACHED")
  print("Current weight is: ")
  print(w)
  if(useBias) { # does the user want to include a bias
        print("Current bias is: ")
        print(b)
  }
  print(sprintf("Iterations taken: %d", i))
  print("ERROR: ")
  print(error)
  if(useBias) { # does the user want to include a bias
        return(list(w=w, b=b))
  }
  return(list(w=w))

}
```

Now we will run our algorithm with the initial zero vector. Even after 1000 our algorithm does not minimize the Least Square Error; however, it did converge to a solution.

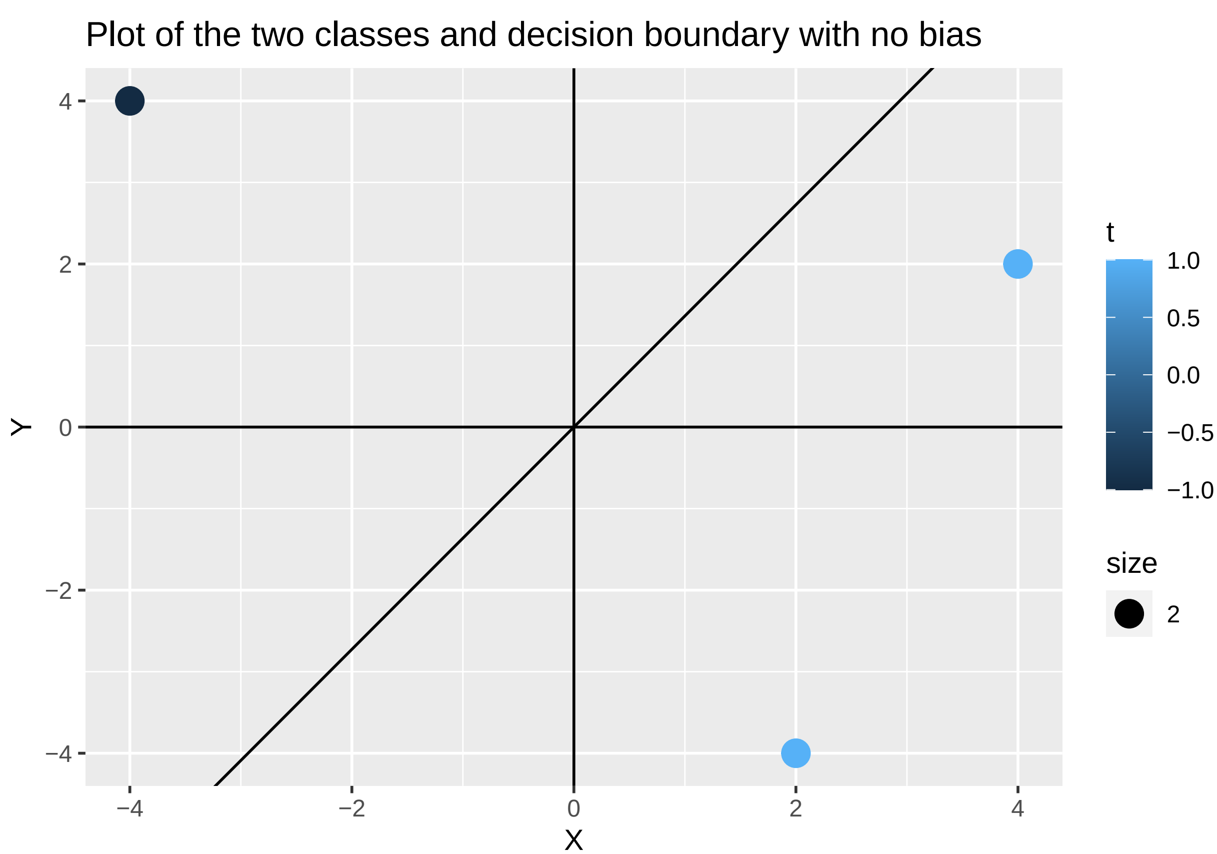
```r
init = matrix(c(0, 0), ncol=2)
input = matrix(c(2, -4, -4, 4, 4, 2), ncol=3, byrow=FALSE)
expected=c(1, -1, 1)
val = LMS(init, 0, 0.01, input, expected, 1000, 1e-10, 1, 0)
```

```
## [1] "MAXIMUM ITERATIONS REACHED"
## [1] "Current weight is: "
##           [,1]        [,2]
## [1,] 0.2485714 -0.03387755
## [1] "Iterations taken: 1000"
## [1] "ERROR: "
##           [,1]
## [1,] 0.5706122
```

As one can see, our outcome weight matrix is the following: $w = [0.24857, -0.033877]$ with zero bias. This leads to the following decision boundary: $p_1 = \frac{0.33877}{0.2485714} p_1$

```r
library(ggplot2) # used for plots
```

```r
data =data.frame(p1=c(2, -4, 4), p2=c(-4, 4, 2), t=c(1, -1, 1))
ggplot(data=data, aes(x=p1, y=p2, color=t))+geom_point(aes(size=2))+xlab("X")+
  ylab("Y")+ggtitle("Plot of the two classes and decision boundary with no bias")+
  geom_hline(yintercept=0)+geom_vline(xintercept=0)+
  geom_abline(intercept=0, slope=0.33877/0.2485714)
```



Now we will run our algorithm again with the initial zero vector, but this time allow bias.

```
val = LMS(init, 0, 0.01, input, expected, 1000, 1e-10, 1, 1)
```

```
## [1] "    Algorithm CONVERGED:"
## [1] "Current weight is: "
##           [,1]        [,2]
## [1,] 0.2307692 -0.07692308
## [1] "Current bias is: "
##           [,1]
## [1,] 0.2307692
## [1] "Iterations taken: 321"
```

After 321 iterations, our algorithm converged to a solution, we get the following weight matrix is the following: $w = [0.23076, -0.07692]$ with bias $b = 0.23076$. This leads to the following decision boundary: $p_2 = \frac{0.2307692}{0.07692308} + \frac{0.2307692}{0.07692308} p_2$

```
data =data.frame(p1=c(2, -4, 4), p2=c(-4, 4, 2), t=c(1, -1, 1))
ggplot(data=data, aes(x=p1, y=p2, color=t))+geom_point(aes(size=2))+xlab("X")+
  ylab("Y")+ggtitle("Plot of the two classes and decision boundary with bias")+
  geom_hline(yintercept=0)+geom_vline(xintercept=0)+
  geom_abline(intercept=0.2307692/0.07692308, slope=0.2307692/0.07692308)
```