Chapter 16: Competitive Networks

Brandon Morgan

2/1/2021

E16.10

1)

Here we have our 200 random input vectors such that $0 \leq p_1 \leq 1$ and $2 \leq p_2 \leq 3$.

```
p1 = runif(200, 0, 1)
p2 = runif(200, 2, 3)
p = matrix(c(p1, p2), nrow=2, byrow = TRUE)
# initial values for weight matrix
p1 = runif(16, 0, 1)
p2 = runif(16, 2, 3)
init1 = matrix(c(p1, p2), nrow=16)
```

2)

We want to classify sixteen classes; therefore our weight matrix will be 16 by 2 in dimension. See above for the initial weight matrix. For our competitive layer, $0 < \epsilon < \frac{1}{S-1} = \frac{1}{16-1} = 0.0\bar{6}$; therefore, $\epsilon = 0.05$ will be chosen.

```
init2 = matrix(-0.05, ncol=16, nrow=16)
for(i in 1:16) {
  for(j in 1:16) {
    if(i==j) {
      init2[i,j] = 1
    }
  }
}
```

```
compet_layer = function(init, weight, tol, maxIter) {



  transfer = function(x, poslin) {
    result = matrix(0,ncol=ncol(x), nrow=nrow(x))
    max = x[1,1]
    indices = c(1,1)
    for(i in 1:nrow(x)) {
      for(j in 1:ncol(x)) {
```

```
        if(poslin) {
          if(x[i,j] < 0) {
            result[i,j] = 0
          }
          else {
            result[i,j] = x[i,j]
          }
        }
        else {  # else competitive, 1 for largest value, 0 else
          if(max < x[i,j]) {
            max = x[i,j]
            indices = c(i,j)
          }
          result[i,j] = 0
        }

      }
    }

    if(poslin) {
      return(result)
    }
    else {
      result[indices[1], indices[2]] = 1
    }
    return(result)
  }


  a = init
  prev = init

  for(i in 1:maxIter) {
    n = weight%*%a
    a = transfer(n, TRUE)
    if(norm(a-prev, "F")<tol) {
      return(transfer(a, FALSE))
    }
    prev = a
  }
  return(transfer(a, FALSE))
}
```

```
neural_network = function(init1, init2, inputs, tol, maxIter1, maxIter2, learning_rate) {

  w = init1
  size = ncol(inputs)
  rownum = nrow(w)
  prev = w
  for( i in 1:maxIter1) {
    for(j in 1:size) {
      n = matrix(0, nrow=rownum, ncol=1)
      for(k in 1:rownum) {
```

```
        n[k,] = -norm(w[k,]-t(inputs[,j]), "F")
      }
      a = compet_layer(n, init2, tol, maxIter2)
      p_index = which.max(a[,1]) # returns index of largest value
      w[p_index,] = t(t(w[p_index,])+learning_rate*(inputs[,p_index]-t(w[p_index,])))
    }
    if(norm(w-prev, "F") < tol) {
      print(sprintf("  Algorithm Converged "))
      print(sprintf("Iterations Needed; %d", i))
      print("weights: ")
      print(w)
      return(w)
    }
    prev = w
  }
  print(sprintf("  Maximum Iterations Reached "))
  print(sprintf("Error: %f", norm(w-prev, "F")))
  print("weights: ")
  print(w)
  return(w)
}
```

```
b=neural_network(init1, init2, p, 1e-7, 100, 100, 0.5)
```

```
## [1] "  Algorithm Converged "
## [1] "Iterations Needed; 6"
## [1] "weights: "
##              [,1]     [,2]
##  [1,] 0.97666786 2.888028
##  [2,] 0.37294800 2.846182
##  [3,] 0.46218927 2.684200
##  [4,] 0.21049513 2.853154
##  [5,] 0.60696308 2.014700
##  [6,] 0.09563571 2.574882
##  [7,] 0.65547049 2.032424
##  [8,] 0.90543564 2.957371
##  [9,] 0.13227963 2.574027
## [10,] 0.92078066 2.910428
## [11,] 0.06742173 2.693059
## [12,] 0.33460810 2.440252
## [13,] 0.66776520 2.818622
## [14,] 0.17347927 2.482667
## [15,] 0.63343525 2.282541
## [16,] 0.19449856 2.052039
```

Here are our final decision boundaries

```
library(ggplot2)
```

```
## Warning: package 'ggplot2' was built under R version 3.6.3
```

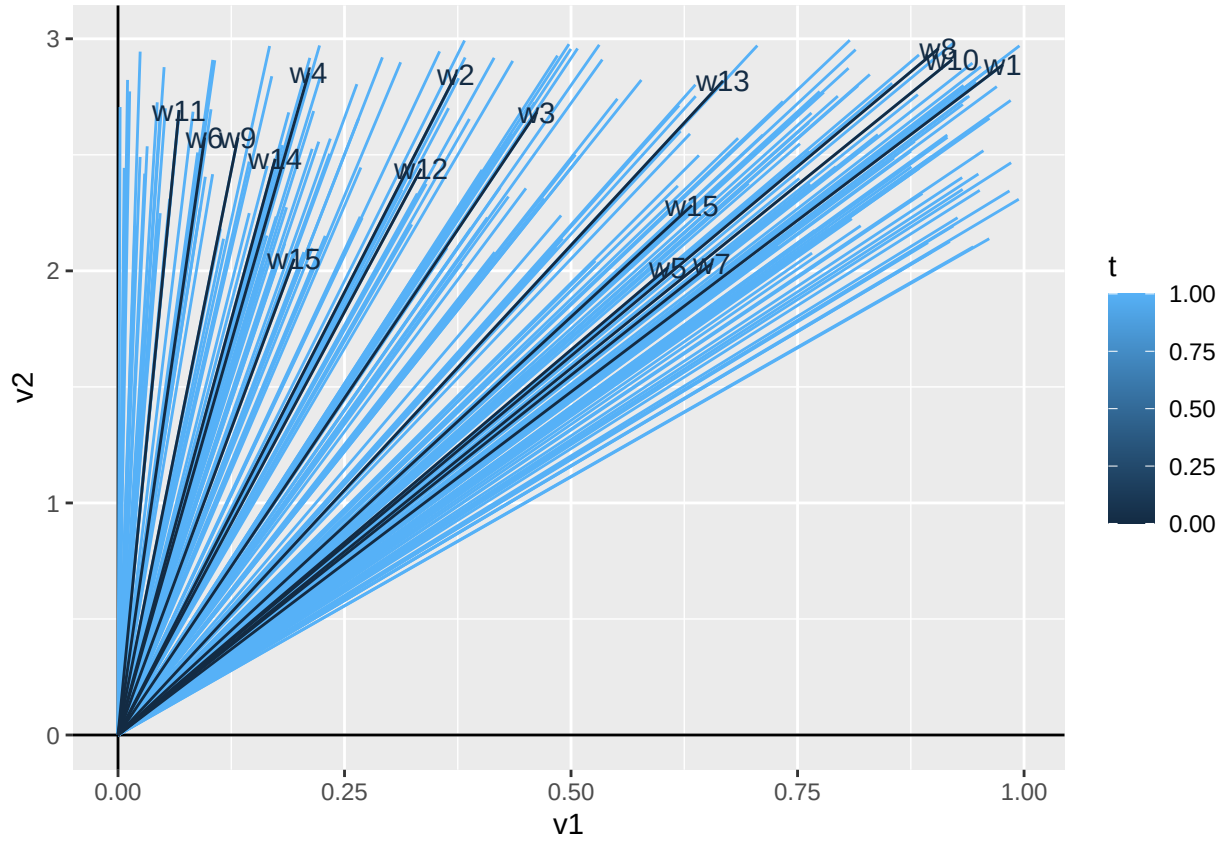
```
# all input and decision boundaries
all_data = matrix(0, ncol=2, nrow=216)
all_data[1:200, 1:2] = t(p)
all_data[201:216,1:2]=b

# used to label decision boundaries
labels = matrix("", nrow=216, ncol=1)
labels[201:216, 1] = matrix(c("w1", "w2", "w3", "w4", "w5", "w6", "w7", "w8",
                              "w9", "w10", "w11", "w12", "w13", "w14", "w15", "w15"),
                            ncol=1)

t =matrix(1, nrow=216, ncol=1) # used as colors
t[201:216,1] = matrix(0, nrow=16, ncol=1)

data = data.frame(v1=all_data[,1], v2= all_data[,2],
t=t[,1])


ggplot(data=data, aes(x=v1, y=v2, color=t))+geom_hline(yintercept=0)+geom_vline(xintercept=0)+
  geom_segment(data=data, aes(x=0, xend=v1, y=0, yend=v2, color=t))+
  geom_text(data=data, aes(x=v1, y=v2, label=labels))
```



Now if we were allowed to change the input values to all range from $-1 \leq p_i \leq 1$:

```r
p1 = runif(200, -1, 1)
p2 = runif(200, -1, 1)
p = matrix(c(p1, p2), nrow=2, byrow = TRUE)
# initial values for weight matrix
p1 = runif(16, -1, 1)
p2 = runif(16, -1, 1)
init1 = matrix(c(p1, p2), nrow=16)

b=neural_network(init1, init2, p, 1e-7, 100, 100, 0.5)
```

```
## [1] "  Algorithm Converged "
## [1] "Iterations Needed; 7"
## [1] "weights: "
##             [,1]        [,2]
##  [1,] -0.2423756 -0.28214001
##  [2,] -0.3519209 -0.60267718
##  [3,]  0.9030735 -0.59665230
##  [4,] -0.9044326  0.31657699
##  [5,]  0.8297407  0.95440147
##  [6,] -0.5956554  0.89995367
##  [7,]  0.8140031 -0.86996985
##  [8,]  0.8387576  0.33416557
##  [9,]  0.4950875 -0.66495735
## [10,] -0.7109280  0.85450565
## [11,] -0.1765840 -0.17273424
## [12,]  0.4935648 -0.80169284
## [13,]  0.1275655 -0.25729543
## [14,]  0.4982106 -0.32859124
## [15,]  0.6219688  0.26089138
## [16,]  0.3535049 -0.09956948
```

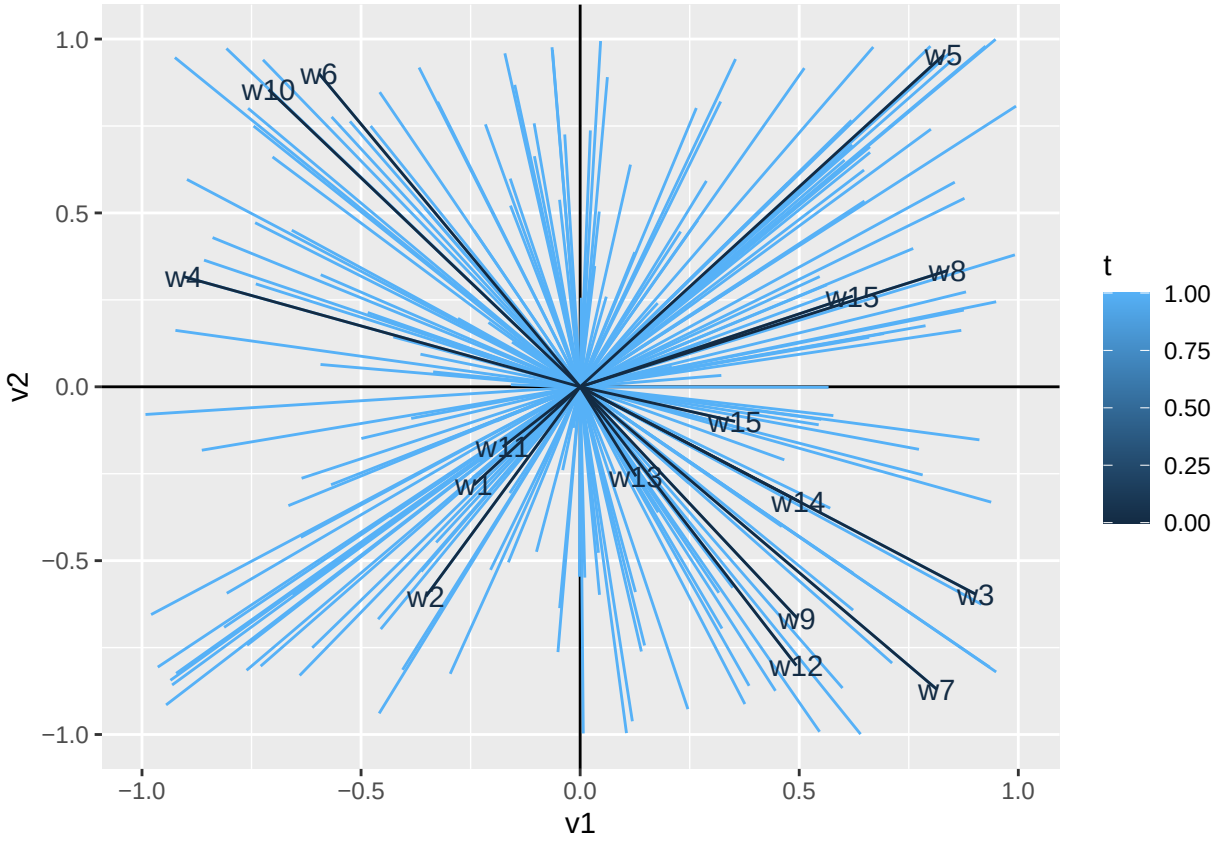
```r
# all input and decision boundaries
all_data = matrix(0, ncol=2, nrow=216)
all_data[1:200, 1:2] = t(p)
all_data[201:216,1:2]=b

# used to label decision boundaries
labels = matrix("", nrow=216, ncol=1)
labels[201:216, 1] = matrix(c("w1", "w2", "w3", "w4", "w5", "w6", "w7", "w8",
                              "w9", "w10", "w11", "w12", "w13", "w14", "w15", "w15"),
                            ncol=1)

t =matrix(1, nrow=216, ncol=1) # used as colors
t[201:216,1] = matrix(0, nrow=16, ncol=1)

data = data.frame(v1=all_data[,1], v2= all_data[,2],
t=t[,1])


ggplot(data=data, aes(x=v1, y=v2, color=t))+geom_hline(yintercept=0)+geom_vline(xintercept=0)+
  geom_segment(data=data, aes(x=0, xend=v1, y=0, yend=v2, color=t))+
  geom_text(data=data, aes(x=v1, y=v2, label=labels))
```


w5
w6
w10
w4
w8
w15
w15
w11
w1
w13
w14
w3
w2
w9
w12
w7
1.0
0.5
0.0
−0.5
−1.0
v2
−1.0
−0.5
0.0
0.5
1.0
v1
t
1.00
0.75
0.50
0.25
0.00