Chapter 16: Competitive Networks

Brandon Morgan

2/1/2021

E16.3

We are given the following input patterns:

$$p_1 = \begin{bmatrix} 1 \\ -1 \end{bmatrix}, p_2 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}, p_3 = \begin{bmatrix} -1 \\ -1 \end{bmatrix}$$

1)

With the following initial weight matrix:

$$W_0 = \begin{bmatrix} \sqrt{2} & 0 \\ 0 & \sqrt{2} \end{bmatrix}$$

We will use Kohonen learning rule with $\alpha = 0.5$ for one iteration.

Before we begin, we must remember that the competitive layer, *compet*, refers to a recurrent layer of the form given in Figure 3.5, where $a^2(0) = a^1, a^2(t+1) = poslin(W^2 a^2(t))$, where $W^2 = \begin{bmatrix} 1 & -\epsilon \\ -\epsilon & 1 \end{bmatrix}$, where ϵ is some number between $0 \leq \epsilon < \frac{1}{S-1}$, where S is the number of rows for the weight matrix.

Let's set up our competitive layer. Because the output of our first layer will be a $2x1$ matrix, $S = 2$, therefore we can choose our $\epsilon = \frac{3}{4}$ (this was chosen randomly); therefore, $W^2 = \begin{bmatrix} 1 & -\frac{3}{4} \\ -\frac{3}{4} & 1 \end{bmatrix}$. Then,

```
compet = function(init, weight, bias, 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]
          }
```

```r
        }
        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))
}
weight = matrix(c(1, -.75, -0.75, 1), ncol=2)
```

```r
init = matrix(c(-0.7071, -1.7071), ncol=1)
#compet(init, weight, bias, 1e-7, 10)
```

First, we present p_1 :

$$a = compet(W_0 p_1) = compet(\begin{bmatrix} \sqrt{2} & 0 \\ 0 & \sqrt{2} \end{bmatrix} \begin{bmatrix} 1 \\ -1 \end{bmatrix}) = compet(\begin{bmatrix} \sqrt{2} \\ -\sqrt{2} \end{bmatrix}) = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

As we can see, our first neuron responded with the first row of the weight matrix as it was closest. Therefore, we update that row with Kohonen rule:

$$w_1 = w_1 + \alpha(p_1 - w_1) = \begin{bmatrix} \sqrt{2} \\ 0 \end{bmatrix} + 0.5(\begin{bmatrix} 1 \\ -1 \end{bmatrix} - \begin{bmatrix} \sqrt{2} \\ 0 \end{bmatrix}) = \begin{pmatrix} 1.20710 \\ -0.5 \end{pmatrix}$$

Now, our weight matrix becomes:

$$W_1 = \begin{bmatrix} 1.20710 & -0.5 \\ 0 & \sqrt{2} \end{bmatrix}$$

Now, we present p_2 :

$$a = compet(W_1 p_2) = compet(\begin{bmatrix} 1.20710 & -0.5 \\ 0 & \sqrt{2} \end{bmatrix} \begin{bmatrix} 1 \\ 1 \end{bmatrix}) = compet(\begin{bmatrix} 0.7071 \\ \sqrt{2} \end{bmatrix}) = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

As we can see, our second neuron responded with the second row of the weight matrix as it was closest. Therefore, we update that row with Kohonen rule:

$$w_2 = w_2 + \alpha(p_2 - w_2) = \begin{bmatrix} 0 \\ \sqrt{2} \end{bmatrix} + 0.5(\begin{bmatrix} 1 \\ 1 \end{bmatrix} - \begin{bmatrix} 0 \\ \sqrt{2} \end{bmatrix}) = \begin{pmatrix} 0.5 \\ 1.20710 \end{pmatrix}$$

Now, our weight matrix becomes:

$$W_2 = \begin{bmatrix} 1.20710 & -0.5 \\ 0.5 & 1.20710 \end{bmatrix}$$

Now, we present p_3 :

$$a = compet(W_2 p_3) = compet(\begin{bmatrix} 1.20710 & -0.5 \\ 0.5 & 1.20710 \end{bmatrix} \begin{bmatrix} -1 \\ -1 \end{bmatrix}) = compet(\begin{bmatrix} -0.7071 \\ -1.7071 \end{bmatrix}) = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

As we can see, our third neuron responded with the first row of the weight matrix as it was closest. Therefore, we update that row with Kohonen rule:

$$w_1 = w_1 + \alpha(p_3 - w_1) = \begin{bmatrix} 1.20710 \\ -0.5 \end{bmatrix} + 0.5(\begin{bmatrix} -1 \\ -1 \end{bmatrix} - \begin{bmatrix} 1.20710 \\ -0.5 \end{bmatrix}) = \begin{pmatrix} 1.10355 \\ -0.75 \end{pmatrix}$$

Now, our weight matrix becomes:

$$W_2 = \begin{bmatrix} 1.10355 & -0.75 \\ 0.5 & 1.20710 \end{bmatrix}$$

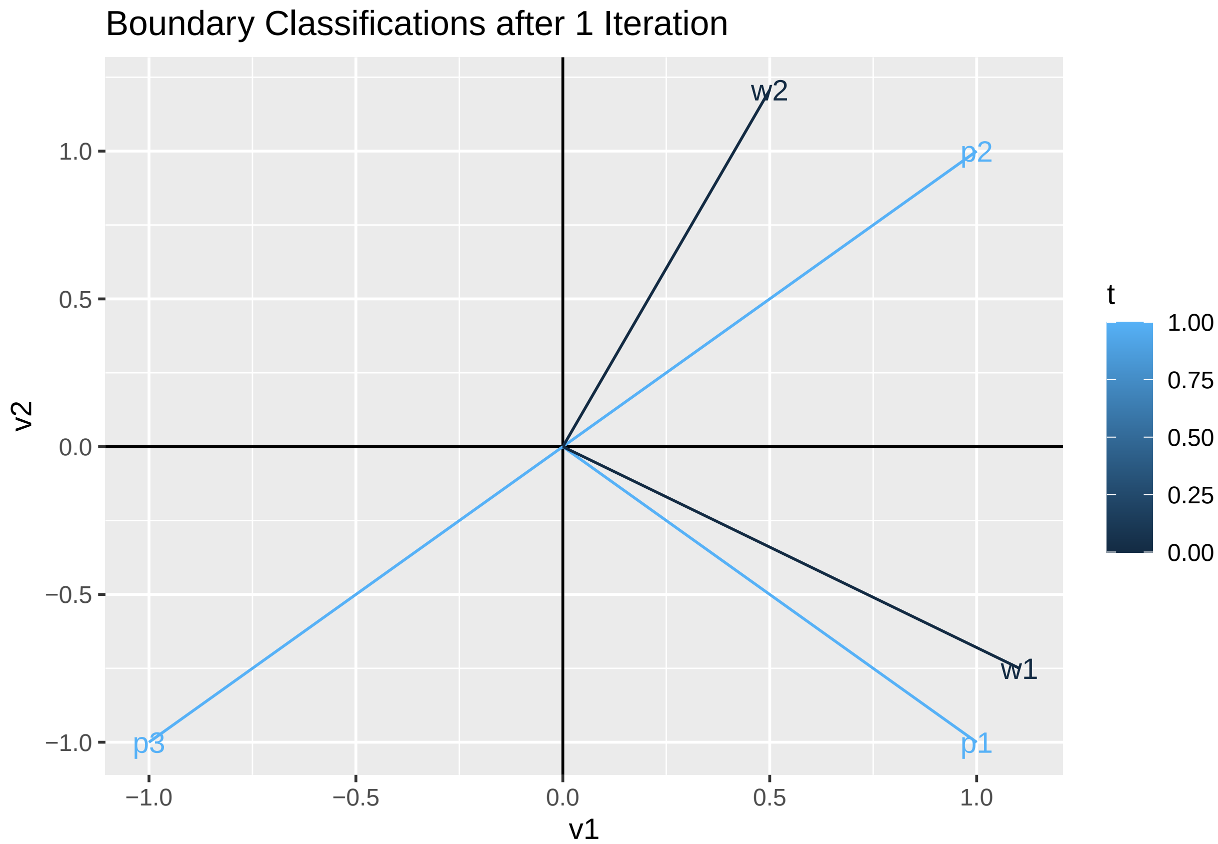
This completes one iteration.

2

```r
library(ggplot2)
```

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

```r
data = data.frame(v1=c(1, 1, -1, 1.10355, 0.5), v2= c(-1, 1, -1, -0.75, 1.20710),
                  t=c(1, 1, 1, 0, 0), names=c("p1", "p2", "p3", "w1", "w2"))
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=names))+
  ggtitle("Boundary Classifications after 1 Iteration")
```


Boundary Classifications after 1 Iteration
w2
p2
p3
p1
w1
v1
v2
t
1.00
0.75
0.50
0.25
0.00
1.0
0.5
0.0
−0.5
−1.0
−1.0
−0.5
0.0
0.5
1.0