Chapter 15: Associative Learning

Brandon Morgan

2/1/2021

E15.5

We are given the following inputs:

$$p^0(1) = 1, p(1) = \begin{bmatrix} 0.174 \\ 0.985 \end{bmatrix}$$

$$p^0(2) = 0, p(2) = \begin{bmatrix} -0.174 \\ 0.985 \end{bmatrix}$$

$$p^0(3) = 1, p(3) = \begin{bmatrix} 0.174 \\ 0.985 \end{bmatrix}$$

$$p^0(4) = 0, p(4) = \begin{bmatrix} -0.174 \\ 0.985 \end{bmatrix}$$

$$p^0(5) = 1, p(5) = \begin{bmatrix} 0.174 \\ 0.985 \end{bmatrix}$$

$$p^0(6) = 0, p(6) = \begin{bmatrix} -0.174 \\ 0.985 \end{bmatrix}$$

And the following learning rate $\alpha = 0.6$, we will apply the instar rule to a single layer hardlim function. The instar rule is given by $w_i(q) = w_i(q-1) + \alpha a_i(q)(p(q) - w_i(q-1))$.

```
instar = function(weight, input, bias, learning_rate, maxIter, mask) {
  size = length(input)
  index = 1
  all = 0
  iter = 0
  while(all<size) {
    if(index > size) {
      index = 1
    }
    p = input[[index]]
    n = weight%*%p+bias
    hardlim = function(x) {
      if (x<0) {
        return(0)
```

```
      }
      return(1)
    }
    a = hardlim(n)
    weight[1,2:3] = weight[1,2:3]+learning_rate*a*(p[2:3,1]-weight[1,2:3])
    index = index + 1
    if(iter > maxIter-1){
       print(sprintf("   MAX ITER TAKEN"))
      print("Final Weights: ")
      print(weight)
      return(weight)
    }
    if(!mask) {
      print(sprintf("   ITERATION: %d", iter+1))
      print(sprintf("n: %f", n))
      print(sprintf("a: %f", a))
      print(sprintf("New weights:"))
      print(weight)
    }
    if(a==1) {
      all = all+1
    }
    if(iter %% size == 0 && iter > 0 && all == 0) {
      all = 0
    }

    iter = iter + 1
  }
  print(sprintf("   ALGORITHM CONVERGED"))
  print(sprintf("Iter taken: %d", iter))
  print("Final Weights: ")
  print(weight)
  return(weight)
}

p1 = matrix(c(1, 0.174, 0.985), nrow=3)
weight = matrix(c(1, 0, 0),nrow=1)
p2 = matrix(c(0, -0.174, 0.985), nrow=3)
w=instar(weight, list(p1, p2, p1, p2, p1, p2), -0.5, 0.6, 60, 0)
```

```
## [1] "   ITERATION: 1"
## [1] "n: 0.500000"
## [1] "a: 1.000000"
## [1] "New weights:"
##      [,1]   [,2]  [,3]
## [1,]    1 0.1044 0.591
## [1] "   ITERATION: 2"
## [1] "n: 0.063969"
## [1] "a: 1.000000"
## [1] "New weights:"
##      [,1]     [,2]   [,3]
## [1,]    1 -0.06264 0.8274
## [1] "   ITERATION: 3"
```

```
## [1] "n: 1.304090"
## [1] "a: 1.000000"
## [1] "New weights:"
##      [,1]     [,2]    [,3]
## [1,]    1 0.079344 0.92196
## [1] "   ITERATION: 4"
## [1] "n: 0.394325"
## [1] "a: 1.000000"
## [1] "New weights:"
##      [,1]       [,2]     [,3]
## [1,]    1 -0.0726624 0.959784
## [1] "   ITERATION: 5"
## [1] "n: 1.432744"
## [1] "a: 1.000000"
## [1] "New weights:"
##      [,1]       [,2]      [,3]
## [1,]    1 0.07533504 0.9749136
## [1] "   ITERATION: 6"
## [1] "n: 0.447182"
## [1] "a: 1.000000"
## [1] "New weights:"
##      [,1]        [,2]      [,3]
## [1,]    1 -0.07426598 0.9809654
## [1] "   ALGORITHM CONVERGED"
## [1] "Iter taken: 6"
## [1] "Final Weights: "
##      [,1]        [,2]      [,3]
## [1,]    1 -0.07426598 0.9809654
```

As we can see from the output, our final weights are almost identical to the training sequence. The sign of 0.074 fluctuates between each iteration. If we allowed our algorithm to run for many iterations, we might expect it to converge to the testing inputs.