Perceptron Exercise

Andey Nunes, MS
Jordan Hilton
Mengyu Li
Peter Boss

January 26, 2019

Setup and definitions

Let's define some of our inputs. I'm going to modify the lists for X to include the coefficient of "1" for the bias to make multiplication easier later, and I'm going to define our step function a to check thresholds.

```
t<-c(0,1,1)
x1<-c(1,0,0)
x2<-c(1,0,1)
x3<-c(1,1,1)
x<-matrix(c(x1,x2,x3), ncol=3, byrow=TRUE)
w<-c(.1,.1,-.3)
a<-function(x) if(x>0){1}else{0}
pander(cbind(x, t))
```

			t
1	0	0	0
1	0	1	1
1	1	1	1

Part a

Let's calculate the accuracy by comparing t to $y = a(w \cdot x)$: where a is the step function defined in the lecture:

```
thresh1<-w%*%x1
thresh2<-w%*%x2
thresh3<-w%*%x3 ## probably a cleaner way to do this with simply w%*%x, but would need to define x diffe
thresh<-c(thresh1,thresh2,thresh3)
y<-sapply(thresh, a)
pander(rbind(thresh,y,t))
```

thresh	0.1	-0.2	-0.1
y	1	0	0
t	0	1	1

You can see that the accuracy is 0/3: none of the y values match the t values.

Part b

Let's apply the learning rule for one epoch:

```
eta<-.2
w<-c(w[1]+eta*(t[1]-y[1])*x[1,1], w[2]+eta*(t[1]-y[1])*x[1,2], w[3]+eta*(t[1]-y[1])*x[1,3]) ##applying l
w<-c(w[1]+eta*(t[2]-y[2])*x[1,1], w[2]+eta*(t[2]-y[2])*x[2,2], w[3]+eta*(t[2]-y[2])*x[2,3]) ##for each e
```

```
w<-c(w[1]+eta*(t[3]-y[3])*x[1,1], w[2]+eta*(t[3]-y[3])*x[3,2], w[3]+eta*(t[3]-y[3])*x[3,3]) ## training
w
```

```
## [1] 0.3 0.3 0.1
```

Part c

Let's check the accuracy as above:

```
thresh1<-w%*%x1
thresh2<-w%*%x2
thresh3<-w%*%x3
thresh<-c(thresh1,thresh2,thresh3)
y<-sapply(thresh, a)
pander(rbind(thresh,y,t))
```

thresh	0.3	0.4	0.7
y	1	1	1
t	0	1	1

Yes, the accuracy has improved to 2/3!