EE 456
Gradient Descent

Aishwarye Omer - ado5146@psu.edu

October 3, 2020

Question: Gradient descent one iteration and error calculation.

Solution: Following is the code and result of the network:

- $x_1 = [\,2, 1, -1\,]$
- $x_2 = [\,0, -1, -1\,]$
- $T_1 = -1$, $T_2 = 1$ reverse of the example did in class

[11]:
```
from sympy import symbols, diff
import numpy as np
np.set_printoptions(precision=2)

#Initialize variables as real numbers for algebraic equations
w1, w2, w3 = symbols('w1 w2 w3', real=True)

#Given initial weight
w = np.array([-2, 1, -1])

#x1 = [2, 1, -1]
#x2 = [0, -1, -1]

#learning rate
n = 0.2

#Error function
e = 0.5*( (-1-(2*w1+w2-w3))**2 + (1-(-w2-w3))**2 )
print("\n Error function \n", e)


#Partial Differentiate
a = diff(e, w1)
print("\n Partial differentiation w.r.t w1 - \t", a)
b = diff(e, w2)
print("\n Partial differentiation w.r.t w2 - \t", b)
```

```
c = diff(e, w3)
print("\n Partial differentiation w.r.t w3 - \t", c)


#Create vector of partial differentation
#Subsitute current weight values using function 'subs'
deltaE = np.array([diff(e, w1).subs({w1:w[0], w2:w[1], w3:w[2]}),
                   diff(e, w2).subs({w1:w[0], w2:w[1], w3:w[2]}),
                   diff(e, w3).subs({w1:w[0], w2:w[1], w3:w[2]})])


#Compute new weight
w_new = w - n * deltaE
np.set_printoptions(precision=2)
print("\n w_new = ", w_new)

#Calculate Error after first training
#substitue new weights
e_new = e.subs({w1:w_new[0],
                w2:w_new[1],
                w3:w_new[2]})
print("\n Error = ", round(e_new,2))
```

```
Error function
0.5*(w2 + w3 + 1)**2 + 0.5*(-2*w1 - w2 + w3 - 1)**2

Partial differentiation w.r.t w1 -      4.0*w1 + 2.0*w2 - 2.0*w3 + 2.0

Partial differentiation w.r.t w2 -      2.0*w1 + 2.0*w2 + 2.0

Partial differentiation w.r.t w3 -      -2.0*w1 + 2.0*w3

w_new =  [-1.60000000000000 1 -1.40000000000000]

Error =  0.20 -> new error
```