# Chapter 12: Variations on Backpropagation

Brandon Morgan

1/22/2021

## 12.9

We are given the following function from E12.3:

$$F(X) = \frac{1}{2}X^T \begin{bmatrix} 10 & -6 \\ -6 & 10 \end{bmatrix} X + \begin{bmatrix} 4 & 4 \end{bmatrix} X$$

With gradient:

$$\nabla F(X) = AX + D = \begin{bmatrix} 10 & -6 \\ -6 & 10 \end{bmatrix} X + \begin{bmatrix} 4 \\ 4 \end{bmatrix}$$

and initial point:

$$x_0^T = [-1, -2.5]$$

We want to perform one iteration of Conjugate Gradient with Golden Search interval reduction for the linear minimization.

First, we sut up our initial search direction, $p_0 = -g_0$:

$$-\nabla F(\begin{bmatrix} -1 \\ -2.5 \end{bmatrix}) = \begin{bmatrix} 9 \\ -15 \end{bmatrix}$$

For our first iteration of conjugate gradient, we need to minimize $F(x)$ along the line:

$$x_1 = x_0 + \alpha_0 p_0 = \begin{bmatrix} -1 \\ -2.5 \end{bmatrix} + \alpha \begin{bmatrix} 9 \\ -15 \end{bmatrix}$$

However, instead of using a learning rate, we will utilize Golden Search:

First, because an initial step size was not given, $\epsilon = 0.075$. We let our initial interval be $[a_1, b_1] = [0, 0.075]$. We now use these interval end points as the learning rate in our minimize function above until there is an increase in function value inbetween two consecutive points:

$$F(a_1) = F(\begin{bmatrix} -1 \\ -2.5 \end{bmatrix} + 0 \begin{bmatrix} 9 \\ -15 \end{bmatrix}) = 7.25$$

$$b_1 = \epsilon = 0.075, F(b_1) = F(\begin{bmatrix} -1 \\ -2.5 \end{bmatrix} + 0.075 \begin{bmatrix} 9 \\ -15 \end{bmatrix}) = 43.3625$$

As we can see from above, there is a function evaluation increase from $[a_1, b_1]$. From the example problems in the book, the previous interval endpoint and the current endpoint would be chosen for the interval, (see P12.4 where increase was in $b3, b_4$ but interval $[b_2, b_4]$ was chosen); but for our example there is no previous point before $a_1$, so the negative of the step size will be used, resulting in our interval to be: $[-0.075, 0.075]$.

$$a_1 = -0.075,\ F(a_1) = F(\begin{bmatrix} -1 \\ -2.5 \end{bmatrix} - 0.075 \begin{bmatrix} 9 \\ -15 \end{bmatrix}) = -2.5375$$

Next, we need to perform interval reduction using Golden Section Search, where $\tau = 0.618$:

$$c_1 = a_1 + (1 - \tau)(b_1 - a_1) = -0.075 + (1 - 0.618)(0.075 + 0.075) = -0.0177$$

$$F(c_1) = F(\begin{bmatrix} -1 \\ -2.5 \end{bmatrix} + -0.0177 \begin{bmatrix} 9 \\ -15 \end{bmatrix}) = 2.5669$$

$$d_1 = b_1 - (1 - \tau)(b_1 - a_1) = 0.075 - (1 - 0.618)(0.075 + 0.075) = 0.0177$$

$$F(d_1) = F(\begin{bmatrix} -1 \\ -2.5 \end{bmatrix} + 0.0177 \begin{bmatrix} 9 \\ -15 \end{bmatrix}) = 13.3993$$

$$F_a = -2.5375,\ F_b = 43.3625,\ F_c = 2.5669,\ F_d = 13.3993$$

Because $F_c < F_d$:

$$a_2 = a_1 = -0.075,\ b_2 = d_1 = 0.0177,\ d_2 = c_1 = -0.0177$$

$$c_2 = a_2 + (1 - \tau)(b_2 - a_2) = -0.075 + (1 - 0.618)(0.0177 + 0.075) = -0.03958$$

$$F(c_2) = F(\begin{bmatrix} -1 \\ -2.5 \end{bmatrix} - 0.03958 \begin{bmatrix} 9 \\ -15 \end{bmatrix}) = -1.196730$$

$$F_b = F_d = 13.3993,\ F_d = F_c = -0.03958\ F_c = F(c_2) = -1.196730$$

Now, this process continues until $b_{k+1} - a_{k+1} < tol$.

Here is the algorithm implementing this search with tolerance $1e - 7$:

```
golden_search = function(line, fun, step_size, maxIter, tol, mask) {

  fa = fun(line(0))

  bprev = 0
  fbprev = fa

  second_bprev = -step_size
  second_fbprev = fun(line(-step_size))

  if(!mask) {
    print("   INTERVAL LOCATION SEARCH   ")
```

```
  print("Initial step: 0")
  print(sprintf("Fb0: %f", fbprev))
}
index = 1

while(TRUE) {

  b = step_size*index
  x = line(b)
  fb = fun(x)

  if(!mask) {
    print(sprintf("Step increase: %f", b))
    print(sprintf("Fb: %f", fb))
  }



  if(fbprev < fb) { # increase in function within two consecutive iterations

    if(!mask) {
      print("INCREASE IN FUNCTION FOUND:")
      print(sprintf("Interval: [%f, %f]", second_bprev, b))
      print("")
    }

    break
  }
  second_bprev = bprev
  second_fbprev = fbprev

  bprev = b
  fbprev = fb
  index = index * 2
}



if(!mask) {
  print("   INTERVAL LINEAR MINIMIZATION   ")
}

a = second_bprev
fa = second_fbprev
tau = 0.618
for(i in 1:maxIter) {

  if(abs(b-a)<tol) {
    print(" ALGORITHM CONVERGED ")
    print(sprintf("Iterations taken: %d", i))
    print(sprintf("Learning Rate: %f", a))
    return(a)
  }
```

```
    c = a + (1-tau)*(b-a)
    d = b-(1-tau)*(b-a)

    fc = fun(line(c))
    fd = fun(line(d))

    if(!mask) {
      print(sprintf("  ITERATION %d", i))
      print(sprintf("c point: %f, fc: %f", c, fc))
      print(sprintf("d point: %f, fd: %f", d, fd))
    }

    if(fc > fd) {

      a=c
      c=d
      d=b-(1-tau)*(b-a)
      fa=fc
      fc=fd
      fd = fun(line(d))
      if(!mask) {
        print("F(c) > F(d)")
         print(sprintf("a=c,c=d, d=%f, fd=%f", d, fd))
      }

    }
    else {

      b=d
      d=c
      c=a+(1-tau)*(b-a)
      fb=fd
      fd=fc
      fc = fun(line(c))
      if(!mask) {
        print("F(c) < F(d)")
        print(sprintf("b=d,d=c, c=%f, fc=%f", c, fc))
      }
    }

  }

  print(" MAXIMUM ITERATIONS TAKEN")
  print(sprintf("Current Learning Rate: %f", (b+a)/2))
  print(sprintf("Error: %f", abs(b-a)))
  return((b+a)/2)
}
```

```
fun = function(X) {
  A = matrix(c(10, -6, -6, 10), ncol=2, nrow=2)
  d = matrix(c(4,4),nrow=1)
  0.5*t(X)%*%A%*%X+d%*%X
}
```

```
gradient = function(X) {
  A = matrix(c(10, -6, -6, 10), ncol=2, nrow=2)
  d = matrix(c(4,4),nrow=2)
  A%*%X+(d)
}
line = function(alpha) {
  x0 = matrix(c(-1,-2.5), nrow=2)
  g0 = matrix(c(9,-15), nrow=2)
  x0+alpha*g0
}
```

Here I call the function for only three iterations to see the output:

```
rate=golden_search(line, fun, 0.075, 3, 1e-10, 0)
```

```
## [1] "   INTERVAL LOCATION SEARCH   "
## [1] "Initial step: 0"
## [1] "Fb0: 7.250000"
## [1] "Step increase: 0.075000"
## [1] "Fb: 43.362500"
## [1] "INCREASE IN FUNCTION FOUND:"
## [1] "Interval: [-0.075000, 0.075000]"
## [1] ""
## [1] "   INTERVAL LINEAR MINIMIZATION   "
## [1] "  ITERATION 1"
## [1] "c point: -0.017700, fc: 2.566899"
## [1] "d point: 0.017700, fd: 13.399299"
## [1] "F(c) < F(d)"
## [1] "b=d,d=c, c=-0.039589, fc=-1.196730"
## [1] "  ITERATION 2"
## [1] "c point: -0.039589, fc: -1.196730"
## [1] "d point: -0.017711, fd: 2.564355"
## [1] "F(c) < F(d)"
## [1] "b=d,d=c, c=-0.053116, fc=-2.401618"
## [1] "  ITERATION 3"
## [1] "c point: -0.053116, fc: -2.401618"
## [1] "d point: -0.039596, fd: -1.197580"
## [1] "F(c) < F(d)"
## [1] "b=d,d=c, c=-0.061476, fc=-2.718089"
## [1] " MAXIMUM ITERATIONS TAKEN"
## [1] "Current Learning Rate: -0.057298"
## [1] "Error: 0.035404"
```

Now I call the function to run until convergence:

```
rate=golden_search(line, fun, 0.075, 100, 1e-10, 1)
```

```
## [1] " ALGORITHM CONVERGED "
## [1] "Iterations taken: 45"
## [1] "Learning Rate: -0.065385"
```

Now our learning rate which minimizises the line $x_1 = x_0 + \alpha g_0$ has been found:

$$x_1 = \begin{bmatrix} -1 \\ -2.5 \end{bmatrix} - 0.065385 \begin{bmatrix} 9 \\ -15 \end{bmatrix} = \begin{bmatrix} -1.5885 \\ -1.5192 \end{bmatrix}$$

This completes one iteration of conjugate gradient with Golden Section interval search. As one can see, each iteration of the algorithm is extremely computational; however, the convergence speed has increased tremendously.