Q2_Data_Science_Assessment

2024-08-07

Question 2

Let x and y be vectors of length n. Consider minimizing the loss L(b) = ||y - b x||^2 over b where b is a scalar. (The solution is b = <x, y> / ||x||^2 .)

First, we have the loss function

$$L(b) = ||y - b \cdot x||^2$$

which is equivalently $\sum_{i=1}^{n}(y_i - b \cdot x_i)^2$

The gradient is the derivative of $L(b)$ with respect to b:

$$\frac{dL(b)}{db} = \frac{d}{db}\left(\sum_{i=1}^{n}(y_i - b \cdot x_i)^2\right) = \frac{d}{db}\left(\sum_{i=1}^{n} y_i^2 - 2b\sum_{i=1}^{n} y_i \cdot x_i + b^2\sum_{i=1}^{n} x_i^2\right)$$

We can simplify to yield the gradient to be:

$$\frac{dL(b)}{db} = -2\sum_{i=1}^{n} y_i \cdot x_i + 2b\sum_{i=1}^{n} x_i^2$$

We can check that this is correct since we are given the proper solution $b = \frac{<x,y>}{||x||^2}$. Setting $\frac{dL(b)}{db} = 0$ and solving for b gives: $\frac{\sum_{i=1}^{n} y_i \cdot x_i}{\sum_{i=1}^{n} x_i^2} = b$

Q2 (a) Write a function in R or python that takes two vectors or numpy vectors and iterates to solve for b using gradient descent. That is, the update is: Update(b) = Current value of b - e * Derivative of L with respect to b evaluated at the current value of b

Where e is a user-supplied real number usually called the learning rate or step size.

```
gradient_descent <- function(x, y, e, iterations) {
  n <- length(x) # length of vector
  b <- 0 # starting value
  for (i in 1:iterations) {
    gradient <- -2*(sum(x*y) - b*sum(x^2)) # derived above
    b <- b - e*gradient
  }
  return(b)
}

# a random set of standard normal vectors of length 7
x = rnorm(7)
y = rnorm(7)
```

```r
# check that b_est matches true solution (reasonable e chosen)
b_est = gradient_descent(x, y, e = 0.005, iterations = 2000)
b_est
```

```
## [1] 0.5551779
```

```r
# true solution
b = dot(x, y) / sum(x^2)
b
```

```
## [1] 0.5551779
```

Q2 (b) Test your function out on some randomly generated normal vectors where you know the value of b. How does the performance of the algorithm's depend on e? When does the algorithm fail and why?

Note that we set the number of iterations to 2000 (as this specification was sufficient for the algorithm to converge when an appropriate e was chosen) and that we are looking at standard normal vectors for simplicity.

The performance of the algorithm depends on the learning rate (e) such that the minimum loss is achieved in a range of e values, before which the loss is minimally higher and after which the loss exponentially increases to infinity. For example 1, where we generated x and y from a standard normal distribution of length 3, the well-behaved range of e is between 0.00068 and 0.17. When our learning rate is too large (>0.17 in this example), the algorithm is overshooting the minima of the loss function and we never converge to the true optimal value of b. It appears that the range of an appropriate learning rate (and the minimal loss value achieved) is dependent on the details of the random vectors which were chosen, however the general trend described previously holds for examples 2 and 3.