

Assignment 1 - Data Mining for Networks

Correia Ambre - Jeannes Théo

January 09th, 2024

1 Assignement 1

1.1 Exercice 1

1. The derivative of $f(x) = 2x^4 - 4x^3 + 3x^2 + 4x - 3$ is $f'(x) = 8x^3 - 12x^2 + 6x + 4$.
2. The algorithm to do the gradient descent is with $\alpha = 1/10$ is:

Algorithm 1 Gradient Descent

```
 $x := x_0$   
while  $|8x^3 - 12x^2 + 6x + 4| > \epsilon$  do  
   $x := x - 0.1 * (8x^3 - 12x^2 + 6x + 4)$   
end while  
return  $x$ 
```

3. If we apply the algorithm starting from 0, we get to -0.4 first and end at -0.317 . If we do the same algorithm starting from 10, we get -676.4 and then 248120326.787 .
4. The first gradient descent algorithm seems to converge to the minimum of the function, while the second one is clearly diverging. The learning rate is probably too high, and would need to be decreased to get optimal results.

1.2 Exercice 2

3. The stochastic gradient is trained on one sample in each epoch. This is very unstable, and doesn't give results here. This could be improved by adjusting the learning rate with an optimizer such as Adam. The mini-batch gradient descent is trained on a small batch of samples in each epoch. It is more stable than the stochastic gradient descent, but less stable than the batch gradient descent. It converges really fast, but tends to over fit on some batches, resulting in a very unstable loss and an approximation of the real function that is not as good as the batch gradient descent or the stochastic gradient descent.
4. The non-simultaneous gradient descent seems to converge faster than the simultaneous update. However, they both converge to the same result, and are both very similar.

2 Assignement 2

2.1 Exercice 1

We can do a algorithm that builds a supergraph, where each node is a graph, and each edge is the cost between two graphs. Using this algorithm, we can then apply a pathfinding algorithm, here a dijkstra to find the shortest path between the two graphs. Doing this, we can find the cost of the edit distance between the two graphs.

To do this efficiently, we can use heuristics to build the supergraph. First, there is no point in building any

path that has a cost longer than the cost find with a naive algorithm. We can also be sure we will need to remove or add nodes when the number of nodes between graph is different. This means that we can do that as soon as we can, at the beginning of the algorithm for adding nodes, and when there is an isolated node for deleting nodes. We can also sort edges in 3 groups : edges that are in both graphs, edges that have one node in common, and edges that are not in common. This way, we can remove edges that are not in common, and then add or edit edges that have one common node in different paths.

2.2 Exercice 2

The table describing features values for the ideal graph kernel is :





















				
	3	4	4	5
	3	4	4	6
	3	5	2	10
	1	1	0	2
	0	1	1	10
	0	1	0	5
	0	0	0	1
	0	0	0	1
	0	1	0	4
	0	0	0	4
	0	0	0	1
	0	0	0	1
	0	0	0	4
	0	0	0	2
	0	0	0	2
	0	0	0	2

Table 1: Feature Values