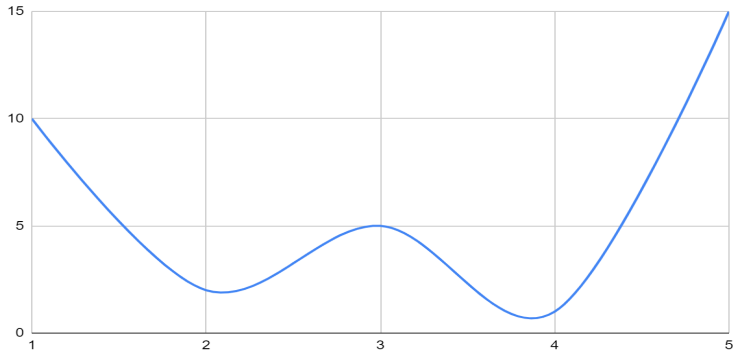
Kahbod Aeini - 98101209

1. a) Wrong. Large ε may mislead to a wrong answer or never converge to the global minimum. For example if $f(x) = |x|$ and set $\varepsilon = 1$. Clearly global minimum is at $x^* = 0$. Now if we set initial point as $x^0 = 1$, then the algorithm only chooses 1 and -1 repeatedly and never converges to the global minimum.

   b) True. If the algorithm leads us to choose a local minimum and we have $x^{(t)} = x'$ where x' is the local minimum, we never converge to the global minimum and the algorithm stops at local minimum. This is because in local minimum, first derivative equals to zero and based on the algorithm formulate we have:
   $x^{(t+1)} = x^{(t)} + \varepsilon f'(x^{(t)}) \Rightarrow x^{(t+1)} = x^{(t)} + \varepsilon(0) \Rightarrow x^{(t+1)} = x^{(t)}$
   And the algorithm never converges to the global minimum.

   c) If algorithm converges to a minimum and the function is convex, the converged point is the global minimum since gradient descent is capable of solving such functions with proper ε.
   But the opposite of the theorem is not true and is easy to prove with a counterexample.

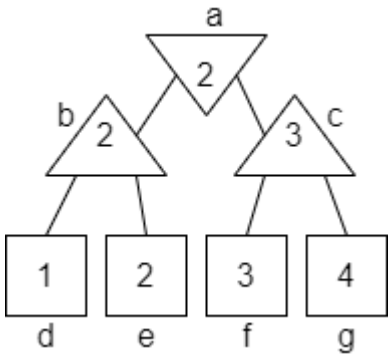


   Easy to notice, the function has a unique global minimum, and if we set the initial point before between 3 and 4 (e.g. 3.5), the algorithm converges to 3.9, which is the global minimum, while the function is not convex.

   d) We have $f(x) = y^2 - 2wx + w^2x^2$, and the second derivative of the function is clearly $f''(x) = 2w^2$ and always true, which causes the function to be convex, and hence, according to the previous section, the algorithm converges to the global minimum.

2.

3. Assume that we start iteration from the most left leaf in each height. Accordingly, the most right leaf can be pruned and will not be checked.
   For example, consider the tree below.

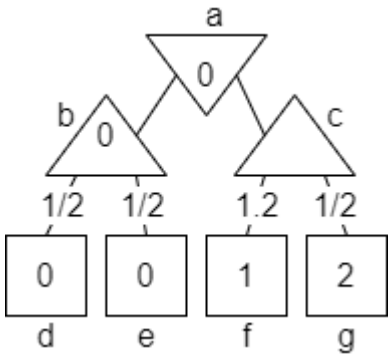


First we check b node children (d, e) nodes. Node b clearly equals 2, due to maximizing. Now we start to check c children to evaluate it. Since we start iteration from left, f will be checked sooner than g. Node f equals 3 and since node c is a maximizer node, it equals 3 or a greater number, if there were in g. So node c at least equals 3 and since node a is a minimizer node, it definitely chooses b node over c and hence there is no need to check g node and it can be pruned.

a) In general case, without knowing anything about the nodes and without having any condition, we can not prune any node since all leaves have a chance to be chosen and we can not ignore this likelihood.

b) The given tree in the first section of the problem has the required condition (all nodes are non-negative) and is an example of an edge being pruned.

Now we consider substituting maximizer nodes with expectimax ones and having non-negative values for all nodes. Take the tree below as an example.



After checking d and e nodes we know that b node will be 0 definitely, and since a is a minimizer node and has a zero node as child, which is the smallest possible number satisfying the condition, it will definitely equal zero and we do not need to check f, g and c nodes.