COMS BC3159: Problem Set 2

Due Friday, October 11, 2024 11:59 PM

Please show your work for all solutions to receive partial/full credit. Always turn in *only* your own, independent work to **Gradescope** (assign each question to a page in your submission). For our late policy, refer to the syllabus. All other questions can be posted on Slack #ps1.

Collaborators:

AI Tool Disclosure:

The following exercises will continue to explore GPU programming and begin to explore numerical optimization. In addition to these written problems, you will also complete the coding part of the assignment, found in the GitHub repo.

---

1 True or False? (4 Points)

For the following statements, please indicate if the statement is true or false and then briefly explain your answer in 1-2 sentences.

(a) Gradient descent will always find a global minima.

(b) The critical points of a function are where the first derivative (or gradient) of that function is equal to zero. These points are guaranteed to be local minima or maxima.

(c) If a function is convex then the line segment connecting two points on the graph of f always lies above the graph of f.

(d) Practical gradient descent methods often do not apply the full gradient update at each iteration.

Solution:

(a) ☐ True ☐ False
Explanation:

(b) ☐ True ☐ False
Explanation:

(c) ☐ True ☐ False
Explanation:

(d) ☐ True ☐ False
Explanation:

2 The *Other* Taylor (8 points)

For this problem we are going to work with the function:

$$y(x) = \frac{3}{2e}(ex - e)^2 - e^x$$

(a) (3 points) Compute a quadratic approximation $\tilde{y}(x)$ of the function at the point $x = 1$ through the use of a Taylor expansion. Please show your work. You do not need to simplify your answer completely.

(b) (2 points) What are the critical point(s) of the approximation you computed in part (a)? Please show your work.

(c) (1 point) What are the critical point(s) of the original function? Note: You do not need to solve this by hand but explain what you did.

(d) (2 points) Are the critical points the same or different? Why or why not? Please briefly explain.

Solution:

(a)

(b)

(c)

(d)

3 Critical Points (12 Points)

For this problem we are going to work with the function:

$$y(x) = x^4 - 3x^2 + 2x$$

Note: You do not need to solve this by hand but explain what you did in each step (do not just write down the final answer).

(a) (2 points) What are the critical points of this function?

(b) (2 points) Compute one step of gradient descent starting from $x = -2$ with $\alpha = 0.025$. Where do you end up?

(c) (1 point) Based on that step do you think $\alpha = 0.025$ was a good choice? If you were to do it again would you leave it the same? Increase it? Decrease it?

(d) (2 points) Starting at the point computed by part b, take one more step of gradient descent this time using $\alpha = 0.1$. Where do you end up?

(e) (1 point) Based on that step do you think $\alpha = 0.1$ was a good choice? If you were to do it again would you leave it the same? Increase it? Decrease it?

(f) (3 points) Now assume you are starting again from $x = -2$ and now doing gradient descent with a backtracking line search starting with $\alpha = 1$ and updating $\alpha = \alpha/2$ on every backtracking step. With what value of α will the *first* step occur under the cost function $J(x) = (x - nearest\_local\_min)^2$ where $nearest\_local\_min$ is (one of) your value(s) computed in part (a)? Where will that take you?

(g) (1 point) Based on everything you have explored in this problem, do you think that resulting α from part f was a good choice? If you were to do it again would you leave it the same? Increase it? Decrease it?

Solution:

(a)

(b)

(c)

(d)

(e)

(f)

(g)

4 CUDA Kernels (8 Points)

Below you will find multiple different examples of GPU kernels designed to add one to the input array in parallel. For this question please assume that the input array is of length 128 and the host code was done correctly. That is, `N=128` and `d_data` was correctly allocated on the GPU to be that length. For each kernel indicate ALL of the kernel launch configurations that will produce the correct output. If none are correct please specify what configuration would work! Launch configuration options are:

i. `increment_kernel<<<1,1,0,0>>>(d_data,128);`

ii. `increment_kernel<<<1,128,0,0>>>(d_data,128);`

iii. `increment_kernel<<<1,256,128*sizeof(int),0>>>(d_data,128);`

iv.
```
dim3 tDims(128,128);
increment_kernel<<<1,tDims,0,0>>>(d_data,128);
```

v.
```
dim3 bDims(128,128); tDims(128,128);
increment_kernel<<<bDims,tDims,0,0>>>(d_data,128);
```

vi. None of the above (please specify the correct configuration)

4.1 Kernel (a)

```
__global__
void increment_kernel(int *d_data, int N) {
    if (threadIdx.x < N) {
        d_data[threadIdx.x] += 1;
    }
}
```

4.2 Kernel (b)

```
__global__
void increment_kernel(int *d_data, int N) {
    for (int i = threadIdx.x; i < N; i += blockDim.x) {
        d_data[i] += 1;
    }
}
```

4.3 Kernel (c)

```
__global__
void increment_kernel(int *d_data, int N) {
    d_data[threadIdx.x] += 1;
}
```

4.4 Kernel (d)

```
__global__
void increment_kernel(int *d_data, int N) {
  int thread_rank = threadIdx.x + blockDim.x*threadIdx.y;
  int threads_per_block = blockDim.x*blockDim.y;
  int block_rank = blockIdx.x + gridDim.x*blockIdx.y;
  int blocks_per_grid = gridDim.x*gridDim.y;
  int grid_rank = thread_rank + block_rank*threads_per_block;
  int threads_per_grid = threads_per_block*blocks_per_grid;
  for (int i = grid_rank; i < N; i += threads_per_grid) {
    d_data[i] += 1;
  }
}
```