Midpoint report

1. What is the status of the CPU version? If you are using existing code for this part, cite the source of the code.
   I have finished the cpu part of the k-means, but use struct of array to store the center, and I am re-writing the center part to make it same with `new_center` in the function `update_cluster_center`

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <float.h>

#define DEFAULT_CLUSTER_NUM 3
typedef struct obj_params params;
typedef struct obj_cluster_center cluster_center;
struct obj_params{
   int obj_num;
   int obj_dimension;
   int cluster_num;
   int max_iterator;
   float* data;
};
/* should be rewritten*/
struct obj_cluster_center{
   int dimension;
   int cluster_idx;
   float *center;
};
params read_data(char *filename){
   if(!filename || filename[0] == '\0'){
       fprintf(stderr,"No file\n");
       exit(EXIT_FAILURE);
   }
   FILE *fp;
   fprintf(stderr, "read %s \n", filename);
   fp = fopen(filename, "r");
   if(!fp){
```

```
        fprintf(stderr, "fopen fail\n");
        exit(EXIT_FAILURE);
    }
    params params;
    int width = 1, height = 0;
    char *line = NULL;
    size_t size;
    ssize_t read_num;
    while((read_num = getline(&line, &size, fp)) != -1){
        if(line[0] == '%' || line[0] == '\n')
            continue;
        if((strcmp(line, "@data\n")) == 0){
            height++;
            continue;
        }
        if(height){
            if(height == 1){
                char *token = NULL;
                if((token = strtok(line, " ,\t")) == NULL){
                    fprintf(stderr,"read err with strtok\n");
                    exit(EXIT_FAILURE);
                }
                while((token = strtok(NULL, " ,\t")) != NULL)
                    width++;
            }
            height++;
        }
    }
    height--;
    fprintf(stderr,"width: %d, height: %d\n" ,width, height);
    if((params.data = (float *)malloc(height * width * sizeof(float))) ==
NULL){
        fprintf(stderr, "malloc fail\n");
        exit(EXIT_FAILURE);
    }
    params.obj_dimension = width;
    params.obj_num = height;
    fseek(fp,0,SEEK_SET);
    height = 0;
    int i = 0;
    while((read_num = getline(&line, &size, fp)) != -1){
```

```
        if(line[0] == '%' || line[0] == '\n')
            continue;
        if((strcmp(line, "@data\n")) == 0){
            height++;
            continue;
        }
        if(height){
            char *token = NULL;
            if((token = strtok(line, " ,\t")) == NULL){
                fprintf(stderr,"read err with strtok\n");
                exit(EXIT_FAILURE);
            }
            params.data[i] = atof(token);
            i++;
            while((token = strtok(NULL, " ,\t")) != NULL){
                params.data[i] = atof(token);
                ++i;
            }
        }
    }
    if(line)
        free(line);
    return params;
}
cluster_center* init_cluster_center(params params,cluster_center
*cluster_centers_new){
    int dimension = params.obj_dimension;
    int k = params.cluster_num;
    cluster_centers_new = (cluster_center*)malloc(sizeof(cluster_center) *
k);
    if(!cluster_centers_new){
        fprintf(stderr, "init cluster_center fail\n");
        return NULL;
    }
    for (int i = 0; i < k; ++i) {
        cluster_centers_new[i].cluster_idx = i;
        cluster_centers_new[i].dimension = dimension;
        cluster_centers_new[i].center = (float*)malloc(sizeof(float) *
dimension);
        if(!cluster_centers_new[i].center){
            fprintf(stderr, "%d cluster center init err\n", i);
```

```
            return NULL;
        }
        for (int j = 0; j < params.obj_dimension; ++j)
            cluster_centers_new[i].center[j] = params.data[i * dimension +
j];
    }
    return cluster_centers_new;
}

void update_cluter_info(params params, cluster_center *cluster_centers_new,
int *cluster_info){
    int k = params.cluster_num;
    int dimension = params.obj_dimension;
    float distance;
    float tmp;
    int idx;
    for (int i = 0 ; i < params.obj_num; ++i) {
        idx = 0;
        distance = FLT_MAX;
        for (int j = 0; j < k; ++j) {
            tmp = 0;
            for (int m = 0; m < dimension; ++m) {
                tmp += (params.data[i * dimension + m] -
cluster_centers_new[j].center[m]) *
                    (params.data[i * dimension + m] -
cluster_centers_new[j].center[m]);
                /* printf("%f %f\n", params.data[i * dimension + m],
cluster_centers_new[j].center[m]); */
            }
            if(tmp < distance){
                idx = j;
                distance = tmp;
            }
        }
        cluster_info[i] = idx;
    }
}
int update_cluster_center(params params, cluster_center
*cluster_centers_new, int *cluster_info){
    int flag = 0;
    int cluster_idx;
```

```
    int count[params.cluster_num];
    memset(count, 0, sizeof(count));
    float *new_center = (float*)calloc(1, sizeof(float) *
params.obj_dimension * params.cluster_num);
    if(!new_center){
        fprintf(stderr, "calloc new_center error");
        exit(EXIT_FAILURE);
    }
    for (int i = 0; i < params.obj_num; ++i) {
        cluster_idx = cluster_info[i];
        count[cluster_idx]++;
        for (int j = 0; j < params.obj_dimension; ++j) {
            new_center[cluster_idx * params.obj_dimension + j] +=
params.data[i * params.obj_dimension + j];
        }
    }
    for (int i = 0; i < params.cluster_num; ++i) {
        if(count[i] == 0){
            fprintf(stderr,"cluster_center %d has no element.\n",i);
            exit(EXIT_FAILURE);
        }
        /* printf("cluster %d: %d\n",i, count[i]); */
        for (int j = 0; j < params.obj_dimension; ++j)
            new_center[i * params.obj_dimension + j] /= count[i];
    }
    for (int i = 0; i < params.cluster_num; ++i) {
        for (int j = 0; j < params.obj_dimension; ++j) {
            if(cluster_centers_new[i].center[j] != new_center[i *
params.obj_dimension + j]){
                cluster_centers_new[i].center[j] = new_center[i *
params.obj_dimension + j];
                flag = 1;
            }
        }
    }
    return flag;
}
void kmeans_cpu(params params){
    cluster_center *cluster_centers_new = NULL;
    if((cluster_centers_new = init_cluster_center(params,
cluster_centers_new)) == NULL)
```

```
        exit(EXIT_FAILURE);

    params.max_iterator = 10000;
    int *cluster_info = (int*)malloc(sizeof(int) * params.obj_num);
    if(!cluster_info){
        fprintf(stderr, "calloc cluster_info error\n");
        exit(EXIT_FAILURE);
    }
    int n = 0;
    for (int i = 0; i < params.max_iterator; ++i) {
        n++;
        update_cluter_info(params, cluster_centers_new, cluster_info);
        if(update_cluster_center(params, cluster_centers_new, cluster_info)
== 0)
            break;
    }
    fprintf(stderr,"iterator %d times\n",n);
    for (int i = 0; i < params.cluster_num; ++i) {
        fprintf(stderr,"centers%d: ",i);
        for (int j=0; j < params.obj_dimension; ++j) {
            printf("%f ", cluster_centers_new[i].center[j]);
        }
        printf("\n");
    }
    /* for (int i = 0; i < params.obj_num; ++i) { */
    /*     printf("obj%d is in cluster%d\n",i,cluster_info[i]); */
    /* } */

    /* for (int i = 0; i < params.obj_num; ++i) */
    /*     printf("Index %d : %d\n",i,cluster_info[i]); */
}
int main(int argc, char *argv[]){
    params params;
    params = read_data(argv[1]);
    if (argc == 3 && atoi(argv[2]) > 1)
        params.cluster_num = atoi(argv[2]);
    else
        params.cluster_num = DEFAULT_CLUSTER_NUM;
    /* for (int i = 0; i < params.obj_dimension * params.obj_num; ++i) { */
    /*     printf("%f ",params.data[i]); */
```

```
    /*      if((i - params.obj_dimension + 1) % (params.obj_dimension) == 0)
*/
    /*          printf("\n"); */
    /* } */
    kmeans_cpu(params);
    free(params.data);
    exit(EXIT_SUCCESS);
}
```

```
hikarisBookpuro:final guangqiqing$ ./a.out phpPQrHPH.arff.txt
read phpPQrHPH.arff.txt
width: 8, height: 10218
iterator 43 times
centers0: 1.508929 1.480952 10.802381 3.832738 25.627380 17.731079 39.834969 5.303571
centers1: 2.374820 8.958193 6.681403 10.244594 8.632388 13.062881 35.354942 5.916386
centers2: 2.877596 7.298220 34.005936 14.813056 34.436943 16.018652 49.133102 3.649852
hikarisBookpuro:final guangqiqing$ ./a.out phpPQrHPH.arff.txt 20
read phpPQrHPH.arff.txt
width: 8, height: 10218
iterator 34 times
centers0: 0.105058 0.719844 0.408560 0.622568 20.949417 20.639950 48.263206 5.739300
centers1: 3.019048 29.476191 4.733333 26.142857 9.952381 10.059910 51.645546 6.314286
centers2: 3.786517 0.000000 6.640450 20.786516 2.501873 22.791653 45.563221 5.722847
centers3: 1.340101 0.527919 28.654821 1.507614 47.939087 19.426094 54.452892 3.883249
centers4: 4.824324 17.905405 22.918919 2.101351 13.405405 20.445402 54.461620 2.722973
centers5: 0.741240 1.148248 3.479784 0.649596 12.862534 10.364747 35.770531 5.846361
centers6: 1.663102 19.508022 4.090909 1.481283 6.604278 20.111300 41.116116 5.245989
centers7: 2.674510 3.131372 19.592157 8.039216 21.013725 9.296300 31.105804 4.900000
centers8: 0.034483 0.089655 0.158621 0.544828 34.220688 34.083614 51.633514 3.055172
centers9: 3.485075 19.261194 11.350746 19.067163 30.835821 20.166578 42.379559 5.044776
centers10: 2.762376 5.549505 48.673267 15.712872 42.574257 13.589759 55.129883 2.480198
centers11: 3.090909 4.353535 33.434345 11.176767 31.969696 11.264546 41.866753 3.545455
centers12: 0.843866 0.706320 3.267658 1.390335 24.468401 21.289881 32.894604 5.397769
centers13: 2.760135 18.496622 8.033784 17.202703 10.239865 9.288213 36.873241 5.837838
centers14: 2.313167 0.558719 20.427046 20.274021 20.195730 20.796101 40.563255 5.391459
centers15: 1.827493 0.380054 17.797844 1.622642 32.530998 15.225607 42.977627 5.908356
centers16: 3.606383 22.898935 34.622341 23.409575 25.090425 12.001864 52.954895 2.898936
centers17: 2.801527 2.259542 36.534351 30.251909 37.702290 29.342371 52.658104 5.541985
centers18: 1.675159 2.130573 5.178344 2.751592 10.270700 7.319814 20.805420 7.487261
centers19: 2.576471 7.620588 3.808824 8.288236 4.700000 10.886470 28.680330 6.252941
```

2. What is the status of the GPU version in terms of completeness? Which functionalities have been implemented and what is missing?
   Now, I only test the reduction add function, the function on device to calculate distance have some problems when compile.
3. What is the status of the GPU version in terms of correctness? Is the, potentially unoptimized, GPU version correct? If not, what is your plan for achieving correctness?
   The GPU version of distance calculate can not working correctly, I may try some simple