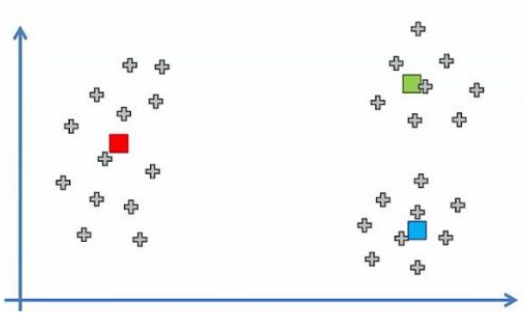
Streamcluster Overview

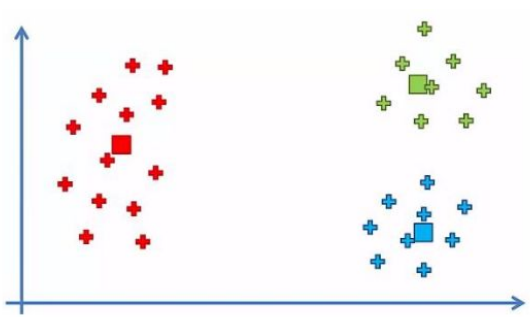
- É uma aplicação de clusterização
- Uso do algoritmo de K-means ++
- principal aplicação em dados não supervisionado

Algoritmo K-means

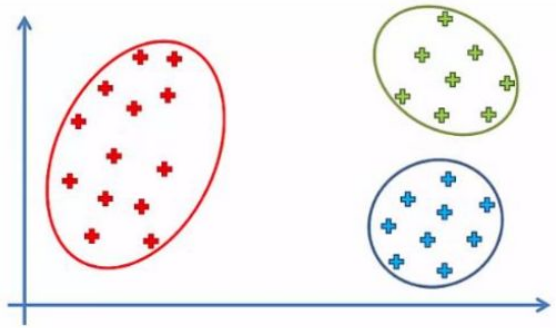
Etapa 1



Etapa 2



Etapa 3



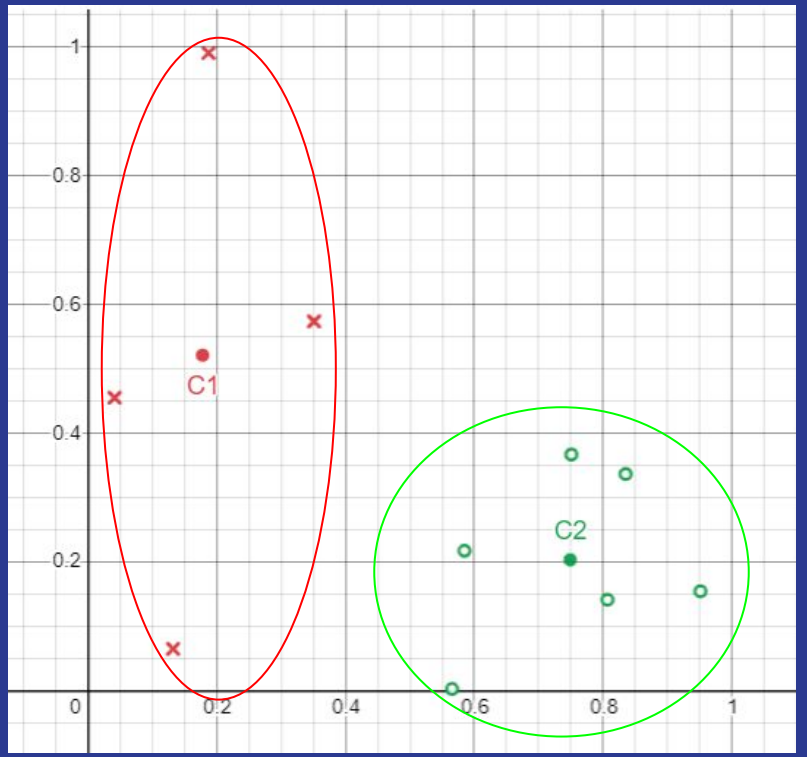
https://drive.google.com/file/d/16ML_03OIGYIrbRgwKSDoWqDvtL_tmv4t/view?usp=sharing

Implementação

X	Y
0.041630	0.454492
0.834817	0.335986
0.565489	0.001767
0.187590	0.990434
0.750497	0.366274
0.351209	0.573345
0.132554	0.064166
0.950854	0.153560
0.584649	0.216588
0.806502	0.140473

Resultado aplicado a 2 centros foram:

4	6
4.000000	6.000000
0.178246 0.520609	0.748801 0.202441



Perfilagem

```
./streamcluster k1 k2 d n chunksize clustersize infile outfile nproc
```

```
./streamcluster 50 100 250 10000 1 10000 myin myout 2
Each sample counts as 0.01 seconds.
  %   cumulative   self              self     total
 time   seconds   seconds    calls  s/call   s/call  name
 82.98     16.75    16.75    24508     0.00     0.00  pgain(long, Points*, double, long*, int, parsec_barrier_t*)
 16.76     20.13     3.38   251412     0.00     0.00  parsec_barrier_wait(parsec_barrier_t*)
  0.30     20.19     0.06       14     0.00     0.00  pspeedy(Points*, float, long*, int, parsec_barrier_t*)
  0.00     20.19     0.00    19991     0.00     0.00  pkmedian(Points*, long, long, long*, int, parsec_barrier_t*)
  0.00     20.19     0.00    10001     0.00     0.00  contcenters(Points*)
  0.00     20.19     0.00    10001     0.00     0.00  localSearch(Points*, long, long, long*)
  0.00     20.19     0.00    10001     0.00     0.00  parsec_barrier_init(parsec_barrier_t*, int const*, unsigned int)
  0.00     20.19     0.00    10001     0.00     0.00  parsec_barrier_destroy(parsec_barrier_t*)
  0.00     20.19     0.00    10000     0.00     0.00  copycenters(Points*, Points*, long*, long)
  0.00     20.19     0.00    10000     0.00     0.00  SimStream::feof()
  0.00     20.19     0.00    10000     0.00     0.00  SimStream::read(float*, int, int)
  0.00     20.19     0.00    10000     0.00     0.00  SimStream::ferror()
  0.00     20.19     0.00        9     0.00     2.18  pFL(Points*, int*, int, float, long*, double, long, float, int,
parsec_barrier_t*)
  0.00     20.19     0.00        1     0.00     0.00  _GLOBAL__sub_I__Z11isIdenticalPfS_i
  0.00     20.19     0.00        1     0.00     0.00  outcenterIDs(Points*, long*, char*)
  0.00     20.19     0.00        1     0.00     0.00  selectfeasible_fast(Points*, int**, int, int, parsec_barrier_t*)
```

Conclusão

- uso de *chunksize*
- *pgain*
- *pthread barriers*

```
1076    // at this time, we can calculate the cost of opening a center
1077    // at x; if it is negative, we'll go through with opening it
1078
1079    for ( int i = k1; i < k2; i++ ) {
1080      if( is_center[i] ) {
1081        double low = z;
1082        //aggregate from all threads
1083        for( int p = 0; p < nproc; p++ ) {
1084    low += work_mem[center_table[i]+p*stride];
1085        }
1086        gl_lower[center_table[i]] = low;
1087        if ( low > 0 ) {
1088    // i is a median, and
1089    // if we were to open x (which we still may not) we'd close i
1090
```