Cost analysis

Define

- n - number of points in the dataset
- k - parameter of the LOF algorithm
- dim - dimensionality of the dataset

Name	Size [bytes]	L1 32KB	L2 256KB	L3 6MB	L4 128MB
input points ptr	$8 \times n \times dim$	2000	16000	375000	8000000
distances indexed ptr	$8 \times n \times n$	64	179	867	4000
k distances indexed ptr	$8 \times n$	4000	32000	750000	16000000
neighborhood index table ptr	$2 \times n \times k$	3200	25600	600000	12800000
reachability distances indexed ptr	$8 \times n \times n$	64	179	867	4000
lrd score table ptr	$8 \times n$	4000	32000	750000	16000000
lof score table ptr	$8 \times n$	4000	32000	750000	16000000
Total	$8 \cdot n \cdot (3 + dim + 0.25k + 2n)$				

Table 1: Data used by the algorithms.
assume sizeof(int) = 2, sizeof(double) = 4. Assume dim = 2, k = 5

Questions:

- at what values of n, k, dim will the data stop fitting in different cache levels?
- what is the maximum number of objects that have to be in the memory simultaneously?

operation	name	FV lat	FV thr	PR lat	PR thr	UB lat	UB thr	AS lat	AS thr
addition	FADD							3	1
mul	FMUL							5	1
division	FDIV							7-27	7-27
fma									
sqrt	FSQRT							27	1
comparison	FCOM							1	1
load from L1									
load from L2									
load from L3									

Table 2: throughput - reciprocal throughput from Aigner's instruction tables

AS: Newhalem i core 7 TODO: add table with the information about operations for our computers / systems

Questions

Function	Denote	Cost Analysis	Flops	Run times
F1	C^{dist}	$n \cdot \frac{n}{2} \cdot (dim * (3 \times C^{add} + C^{mul}) + C^{sqrt})$	$n \times \frac{n}{2}(4 \cdot dim + 1)$	
F2	C^{objAll}	$n \cdot (n-1)\log(n-1)$	$n(n-1)\log(n-1)$	
F3	$C^{neighAll}$	$n \cdot ((n-1) \cdot C^{compare} + k \cdot C^{add})$	$n \times (n-1)$	
F4	$C^{reachAll}$	$n \cdot n \cdot C^{compare}$	n^2	
F5	C^{lrd}	$2 \cdot n \cdot C^{div} + n \cdot k \cdot C^{add}$	$n(k+2)$	
F6	C^{Lof}	$2 \cdot n \cdot C^{div} + n \cdot k \cdot C^{add}$	$n(k+2)$	
Total		$C^{Lof} + C^{lrd} + C^{reachAll} + C^{neighAll} + C^{objAll} + C^{dist}$		

- **ComputeKDistanceObject** should I count accessing array *distances indexed ptr* num pts times?

Which flags:

Bottlenecks

- Compute Pairwise Distance
- Compute Reachability Density

Compilers & flags/Processors

Roofline

AS work in progress

Function	$W(n)$	$Q(n)$	$I(n)$	$T(n)$	$P(n)$
F1	$n \cdot \frac{n}{2}(4 \cdot dim + 1)$	$8(n^2 + n \cdot dim)$	$\frac{dim}{4}$	$\frac{1}{2}n \cdot n \cdot (lat^{sqrt} + dim(3\frac{lat^{add}}{thr^{add}} + \frac{lat^{mul}}{thr^{mul}}))$	$\frac{4 \cdot dim+1}{lat^{sqrt}+dim(3\frac{lat^{add}}{thr^{add}}+\frac{lat^{mul}}{thr^{mul}})}$
F2	$n(n-1)\log(n-1)$	$8(n^2+n)$	$\mathbf{O}(\log(\mathbf{n}))$	$n(n-1)\log(n-1)\frac{lat^{comp}}{thr^{comp}}$	$\frac{thr^{comp}}{lat^{comp}}$
F3	$n \times (n-1)$	$10n^2 + 8n$	$\frac{1}{10}$	$n \cdot (n-1)\frac{lat^{comp}}{thr^{comp}}$	$\frac{thr^{comp}}{lat^{comp}}$
F4	n^2	$8(n^2+2n)$	$\frac{1}{8}$	$n^2\frac{lat^{comp}}{thr^{comp}}$	$\frac{thr^{comp}}{lat^{comp}}$
F5	$n(k+2)$	$2nk + 16n$	$\frac{k+2}{k+8} \approx 1$	$n \cdot (2 \cdot lat^{div} + k\frac{lat^{add}}{th^{add}})$	$\frac{k+2}{k\frac{lat^{add}}{th^{add}}+2 \cdot lat^{div}}$
F6	$n(k+2)$	$2nk + 16n$	$\frac{k+2}{k+8} \approx 1$	$n \cdot (2 \cdot lat^{div} + k\frac{lat^{add}}{th^{add}})$	$\frac{k+2}{k\frac{lat^{add}}{th^{add}}+2 \cdot lat^{div}}$

Table 3: To compute $P(n)$ I have assumed Hashwell process same as introduced during the lecture. $T(n)$ is computed as $numop \cdot latency \cdot throughput$

- **Compute bound** if it has high operational intensity: - **F2**?

- **Memory bound** if it has low operational intensity: - **F1**, **F3**, **F4**, **F5**, **F6**

Memory Optimization

Distance Calculation

Denote

- B_n: number of points loaded in a block
- B_d: number of dimensions, processed in a block
- L: cache size in bytes

Working set [bytes]: $2 \cdot B_n \cdot B_d \cdot sizeof(double) \rightarrow$ fits in cache $\iff B_n \cdot B_d \leq \frac{L}{16} \overset{\text{cache size}=32KB}{=} 2000$

Meeting Notes 2.05.2020

1. KD-trees & Lattice: the main goal of the project is to use the techniques discussed during the lectures to improve the performance. Consequently - algorithmic changes are nice add ons but not mandatory if they take too much time
2. KD-trees, performance measurement: measure execution time with and without building the index tree.
3. KD-trees: for certain non-essential parts of the code, it is possible to use existing libraries instead of implementing everything from scratch
4. Performance measurement: both cold and hot cache measurements can be included for comparison
5. Performance plots: better to use medians and (or) boxplots to account for the runtime variability.
6. Try to show peak performance on the plots !
7. Try *perf* (linux) for code profiling
8. Compilation: check whether compiler performs vectorization
9. Determine which functions are compute and which are memory bound. For memory bound functions try **memory blocking**
10. Python benchmark: make sure sklearn implementation does not use multiple cores. (BLAS ?)
11. **Evaluation:** does not necessarily depend on the achieved speedup. Important: evaluate what to try for which functions and explain why or why not different methods might work.