MEMOCODE 2014 Design Contest: k-Nearest Neighbors with Mahalanobis Distance Metric

Peter Milder
Department of Electrical and Computer Engineering
Stony Brook University
Stony Brook, NY 11794–2350
peter.milder@stonybrook.edu

***Abstract*—The MEMOCODE 2014 hardware/software co-design contest problem is k-Nearest Neighbor search using the Mahalanobis distance metric. Given a data set of points in multi-dimensional space, the goal is to find the k points that are nearest to any given point in that space (quantified with the given distance metric). Contestants were given one month to develop a system to perform the kNN search, aiming to maximize performance or cost-adjusted performance. The two winning teams, which have been invited to contribute papers describing their techniques, combined algorithmic and implementation optimizations. The pure-performance winners targeted the Convey HC-2ex hybrid FPGA/multicore system, while the winners for cost-adjusted performance targeted an Intel multicore.**

I. Introduction

A yearly tradition since 2007, the MEMOCODE design contest presents a problem and challenges participants from around the world to develop effective hardware/software solutions. Previous problems have included stereo matching [1], DNA sequence alignment [2], NoC simulation [3], packet inspection [4], rectangular-to-polar interpolation [5], sorting of encrypted data [6], and matrix-matrix multiplication [7].

This year's problem is to find, given a point in multidimensional space, the k-nearest neighbors of that point in a data set. k-nearest neighbors (kNN) [8] is frequently used in pattern recognition and machine learning applications to find how a newly observed datapoint relates to previously observed ones. Given a set of points in a multidimensional feature space, contestants must find the k nearest points to a given input point. The Mahalanobis distance metric [9], used in biomedical applications, allows quantification of the distance between two points in a way that takes into account the covariance across the dimensions of the feature space. The challenges of this problem lie in the computation to be performed, efficient access of data, and re-organization of the algorithm to allow pre-processing of the data set.

II. Problem

k-Nearest Neighbors is used for problems such as classification ("Which observed class does a new data point resemble most closely?") and regression ("Given a new data point, what was the average behavior of the k nearest previously observed points?"). In both of these problems, it is necessary to find the k points in a known data set which are closest to a given input point. This year's design contest focuses on this search process. Participants are given a *data set* and an *input set* which each contain a set of points in D dimensional space. The goal of the search is to find, for each point in the input set, the k nearest points in the data set.

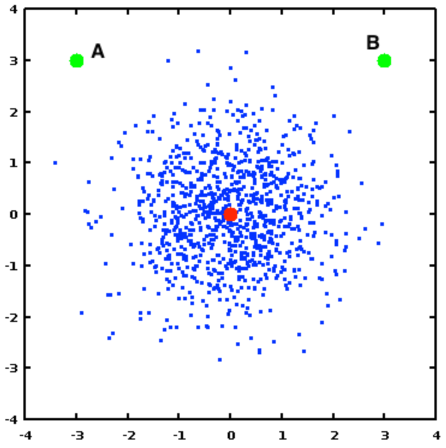

Fig. 1. Example with uniform distribution, $D = 2$.

In order to quantify the distance between two points in D dimensional space, a distance metric must be chosen. To motivate this choice, we first consider examples in two dimensions (i.e, $D = 2$). In Figure 1, the blue dots represent all points of the data set, the red dot represents a particular point we consider, and the two green dots (A and B) represent two points from the input set. We would like to quantify the distance from each of A and B to the red point. In this example, the two dimensions of the data set are not correlated and have equal variance, giving the blue cloud a roughly circular shape. Given this data set and its distribution, the Euclidean distance between two points provides a good metric for the dissimilarity between them. Thus, we could calculate that the distance from either green point to the red point is $\sqrt{3^2 + 3^2} = \sqrt{18} \approx 4.2426$, and we can conclude that A and B are "equally similar" to the red point.

However, many data sets exhibit correlation and differing levels of variance among their dimensions. For example, Figure 2 shows an example where there is a correlation between the two dimensions (giving the area of blue points a generally diagonal shape), and the variance is not uniform. Each of the green points still has the same Euclidean distance from the red point as in the previous example. However now we can see that in the context of the data set, B can be considered to be "more similar" to the red point than A is.

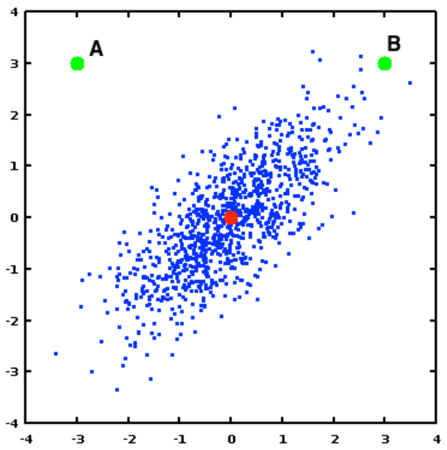

Fig. 2. Example with correlated distribution, $D = 2$.

The Mahalanobis distance metric takes this into account by measuring the distance between two points relative to the covariance of the data set. Let S represent the covariance matrix of the data set (or an estimate of it). (For this two-dimensional example, S is a 2×2 matrix.) The Mahalanobis distance between two points (represented by D-dimensional column vectors $\vec{x}$ and $\vec{y}$) is computed according to

$$\text{dist}(\vec{x}, \vec{y}) = \sqrt{(\vec{x} - \vec{y})^T S^{-1} (\vec{x} - \vec{y})}.$$

For the example shown in Figure 2,

$$S = \begin{bmatrix} 0.96 & 0.69 \\ 0.69 & 0.94 \end{bmatrix} \quad \text{and} \quad S^{-1} = \begin{bmatrix} 2.20 & -1.61 \\ -1.61 & 2.25 \end{bmatrix}.$$

So,

$$\text{dist}\left(\begin{bmatrix} -3 \\ 3 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \end{bmatrix} \right) = 8.315$$

$$\text{dist}\left(\begin{bmatrix} 3 \\ 3 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \end{bmatrix} \right) = 3.318$$

We see that using this distance metric, point B is much closer to the origin than point A is. To avoid the square root operator, we will use the squared Mahalanobis distance as a distance metric, giving:

$$\text{distSq}(\vec{x}, \vec{y}) = (\vec{x} - \vec{y})^T S^{-1} (\vec{x} - \vec{y}) \qquad (1)$$

Problem specification. Given the following:

- a data set C (analogous to the blue dots in the example figures above), comprised of a number of points in 32-dimensional space (i.e., $D = 32$),
- a 32×32 matrix S^{-1}, where S is the covariance matrix of C, and
- an input set N comprised of a given number of points in 32-dimensional space,

the system must, for each point in N, compute the $k = 10$ nearest neighbors in C, sorted from lowest distance to tenth-lowest distance, and report the computed distance values. The timed portion of the implementation must start and end with all data in main system memory.

All initial input values are scaled so they can be represented as 12-bit two's complement integers. As computation progresses, the maximum data size will grow; solutions will be considered correct if they deviate from exact solutions only within the two least significant bits.

III. Functional Reference Implementation and Supplied Data Sets

An unoptimized software implementation was provided to serve as the functional reference for the contestants' optimized implementations. The reference implementation reads input files for the data set C, input set N, and inverse covariance matrix S^{-1}. It then performs the kNN search and writes the resulting indices and distances to output files. This functionality is summarized as:

```
initialize data;
// begin timing here
for i = 0 to input_set_size {
   for j = 0 to data_set_size {
      d = distSq(inputSet[i], dataSet[j],
                    inverseS);
      updateKNearest(d, i, j);
   }
}
// end timing here
output results;
```

Also included was software to compare the computed output with a reference solution, and verify that the results are within the acceptable tolerance.

Three data sets were provided to aid in validation and development. Each set contains a comparison data set C, an inverse covariance matrix S^{-1}, and a set of input vectors N. Each set is accompanied by reference solutions, which provide correct nearest neighbor and distance results. The three data sets are named *small*, *medium*, and *large*:

- Small: C: 1,000 elements; N: 10 elements
- Medium: C: 10,000 elements; N: 100 elements
- Large: C: 10,000,000 elements; N: 1,000 elements

In all cases S^{-1} is a 32×32 matrix, $D = 32$, and $k = 10$. Assuming 12 bits per input number, this means the large data set's inputs require approximately 480 MB of memory.

All data were generated synthetically to have the desired size and covariance characteristics.

IV. The Contest

Participants were given a month to implement a solution using platforms such as FPGAs, GPUs, and CPUs. The solutions were validated using the supplied reference data sets. Performance was measured using the *large* data set. The time taken to initialize data and read out the result was excluded from the runtime measurement. Additionally, any pre-processing of the data set C and inverse covariance matrix S^{-1} was allowed as long as the preprocessing did not include the input set N; this pre-processing was not included in the runtime measurement.

TABLE I. RUNTIMES FOR THE LARGE DATA SET.

Team	Platform	Runtime (sec.)
Iowa State, USA	Convey HC-2ex	1
IPM Multi-Core 1, Iran	Intel Xeon E5-2650	15
IPM Multi-Core 2, Iran	Intel Xeon E5-2650	22
IPM Many-Core, Iran	Intel Xeon Phi 5110P	54

TABLE II. COST-NORMALIZED RUNTIMES FOR THE LARGE DATA SET.

Team	Platform	Cost (USD)	Runtime×Cost
IPM Multi-Core 1, Iran	Intel Xeon X5650	130	4550
IPM Multi-Core 2, Iran	Intel Xeon X5650	130	9360
IPM Many-Core, Iran	NVIDIA GTX 480	219	25185
Iowa State, USA	Convey HC-2ex	100000	107000

The contest included two categories of awards: pure performance and cost-adjusted performance. The pure-performance award was based solely on runtime, while the cost-adjusted award was measured as of the product of runtime and system cost. The system cost was determined based on the lowest listed price; if no price is available, the system cost would be estimated by the judges. Contestants were encouraged to include an estimate of system cost with their submissions.

V. RESULTS

Approximately ten institutions began the contest. Full working implementations were submitted by four teams from two different institutions. A variety of platforms were targeted, including FPGAs, GPUs, and CPUs.

All of the submissions employed a form of pre-processing that was able to reduce the amount of computation and communication required. As one example, the Iowa State team, which used the Convey HC2-ex hybrid FPGA and CPU system, restructured (1) as

$$(\vec{x}-\vec{y})^T(S^{-1}x - S^{-1}y),$$

where $S^{-1}y$ can be precomputed once for the entire data set, $S^{-1}x$ is computed on the CPU, and the remaining operations are computed on the FPGA. Others employed similar techniques, optimized for their platforms.

Tables I and II summarize the results for performance and cost-adjusted performance, respectively. Note that some teams reported the runtime of their implementation on multiple processors; only the fastest or most cost-efficient solution from each team is included in the tables. The cost of the Convey HC-2ex was estimated by the contestants at $100,000; the cost of Intel and NVIDIA systems was based on the lowest published prices found online.

As shown in the tables, the winner for the fastest performance was Iowa State University's implementation on the Convey HC-2ex. The Iowa State team employed efficient pre-processing and utilized an extremely parallel systolic array architecture, employing four FPGAs. The team reports that the limiting factor of their implementation was the number of DSP units available on the FPGAs.

The winner for best cost-adjusted performance was the IPM Multi-Core 1 team from the Institute for Research in Fundamental Sciences (IPM), Iran, whose software implementation on an Intel Xeon X5650 CPU exhibited the lowest runtime-cost product. The IPM Multi-Core 1 team employed another form of pre-processing, and implemented a software solution on a variety of platforms. They reported runtime results on four Intel CPUs; the fastest runtime was obtained on the Intel Xeon E5-2650, but the lower cost of the Intel Xeon X5650 made it a more cost-effective solution. Also of note is the IPM Many-Core team, who used similar pre-processing techniques and targeted their implementation for the NVIDIA GTX-480 GPU and the Intel Xeon Phi 5110P many-core processor.

VI. CONCLUSION

The 2014 MEMOCODE Design Contest received submissions using FPGAs, GPUs and CPUs. Working solutions were submitted by four teams at two institutions. The winning teams made effective use of pre-processing and adapted the algorithms to their intended platforms.

ACKNOWLEDGMENTS

I would like to thank all of the contestants for their hard work and congratulate them on their excellent results.

Thank you also to James Hoe, Sandeep Shukla, and Jean-Pierre Talpin for their help in planning and organizing this contest.

REFERENCES

[1] E. Nurvitadhi, "MEMOCODE 2013 Hardware/Software Co-design Contest: Stereo Matching," in *IEEE/ACM International Conference on Formal Methods and Models for Codesign (MEMOCODE)*, 2013.

[2] S. A. Edwards, "MEMOCODE 2012 Hardware/Software codesign contest: DNA sequence aligner," in *IEEE/ACM International Conference on Formal Methods and Models for Codesign (MEMOCODE)*, 2012.

[3] D. Chiou, "MEMOCODE 2011 Hardware/Software CoDesign Contest: NoC simulator," in *IEEE/ACM International Conference on Formal Methods and Models for Codesign (MEMOCODE)*, 2011.

[4] M. Pellauer, A. Agarwal, A. Khan, M. C. N. Ng, M. Vijayaraghavan, F. Brewer, and J. Emer, "Design Contest Overview: Combined Architecture for Network Stream Categorization and Intrusion Detection (CANSCID)," in *IEEE/ACM International Conference on Formal Methods and Models for Codesign (MEMOCODE)*, 2010.

[5] F. Brewer and J. C. Hoe, "2009 MEMOCODE Co-Design Contest," in *IEEE/ACM International Conference on Formal Methods and Models for Codesign (MEMOCODE)*, 2009.

[6] P. Schaumont, K. Asanovic, and J. C. Hoe, "MEMOCODE 2008 Co-Design Contest," in *ACM/IEEE International Conference on Formal Methods and Models for Codesign (MEMOCODE)*, 2008.

[7] F. Brewer and J. Hoe, "MEMOCODE 2007 co-design contest," in *IEEE/ACM International Conference on Formal Methods and Models for Codesign (MEMOCODE)*, 2007.

[8] T. Cover and P. Hart, "Nearest neighbor pattern classification," *IEEE Trans. Inf. Theor.*, vol. 13, no. 1, pp. 21–27, Sep. 1967.

[9] P. C. Mahalanobis, "On the generalized distance in statistics," *Proceedings of the National Institute of Sciences of India*, vol. 2, no. 1, 1936.