RPC Project Ideas

Contents

1 Introduction

High Performance Computing systems are growing increasingly complex. The error rate of computation is growing and faults are becoming harder to diagnose and correct. Resilience develops methods to keep applications running to a correct solution in spite of errors. The current approaches to resilience are based on fault tolerance, a system of error detection and correction that involves redundancy in time or space, and expenditure in energy. Nondeterministic hardware and programming methods offer a more efficient approach to resilience. *Nondeterministic computing* refers to a model of computation where the nondeterminism is intrinsic to the system (i.e., nondeterministic hardware). Exploring the feasibility of various nondeterministic hardware methods is the primary focuse of our team.

2 Reverse Ising Problem (ZOMBIECUPCAKE)

The basis of the Ising model is a collection of spins, typically viewed as magnets. These spins are arranged in a graph. Each spin is in one of two possible states: "spin up" or "spin down" (corresponding to the poles of a magnet). Each spin is free to switch between states, as well as having some amount of influence over the states of other spins. Each spin has a *local bias*, which governs what state the spin will tend ("prefer") to be in, and how strong that "preference" is. Each pair of spins can be *coupled*, which determines if the spins "prefer" to be in the same state, and each coupled pair has a *coupling strength* which defines how strong this "preference" is. The system naturally evolves to configurations that have low energy, where the energy of a configuration of the system is given by the Hamiltonian:

$$H = \sum_i h_i s_i + \sum_{\langle i,j \rangle} J_{i,j} s_i s_j, \tag{1}$$

where $\langle i, j \rangle$ denotes adjacent sites (each pair is only counted once), h_i denotes the local bias on spin s_i, and $J_{i,j}$ denotes the coupling strength between spin s_i and s_j.

A key feature of the Ising model is that the local biases and coupling strengths govern which configurations are the low energy states of an Ising system, i.e., which combinations of states each spin is in correspond to the minima of the Hamiltonian. This feature enables the Ising model to function as a model of computation. For a given operation, designate a portion of the spins to correspond to each input; a disjoint portion of spins are designated to correspond to the output of the computation. The binary representation of the input values are encoded onto the designated "input spins" (which are then fixed). The system then evolves to an equilibrium configuration. For an appropriate choice of local biases and coupling strengths, the states of the designated "output spins" will correspond to the binary representation of the correct answer of the computation (with some probability).[1]

In the reverse Ising problem, a set of configurations (the set of configurations that correspond to the correct output spins for each set of input spins) are chosen to correspond to the minima of the Hamiltonian. The goal is to determine the local bias and coupling strengths that would cause the system to have these chosen states as the equilibrium states. In order to make this problem solvable, additional degrees of freedom are required. These are obtained by adding *auxiliary spins* – extra spins to the system that have no role in the operation being performed. In the process of calculating the local biases and coupling strengths, the values of the auxiliary spins need to be computed. Adding these auxiliary spins converts the problem from a linear system into one that is mixed integer (since the auxiliaries are binary variables) and either a quadratic or a cubic programming problem.

ZOMBIECUPCAKE is an exploration into different methods of solving the reverse Ising problem, how these methods perform, and how these methods scale.

2.1 Classifying Auxiliary Combinations

The inverse Ising problem is a doubly exponentially hard constraint satisfaction problem that is computational intractable using traditional solvers. It can be approached however by first solving an exponentially hard feasibility problem which, if successful, then reduces the problem to a

[1] For a fixed set of input values, the probability of the equilibrium configuration corresponding to the correct answer is a function of the Hamiltonians of all the configurations of the system with input spins corresponding to those input values. (Note that the rules of commutativity may not be respected, i.e., the input values may not be interchangeable.)

straightforward linear program. There is evidence to suggest that ML techniques may be effective at classifying solutions to the exponentially hard feasibility problem in polynomial time. This project involves training a variety of supervised ML approaches to data generated from sample problems and testing the performance of these models on new problems.

2.2 Implementing Machine Learning Approaches

Solving the reverse Ising problem involves classifying sets of values as feasible or infeasible. This inherently seems like a problem that machine learning (ML) strategies could perform well.There are various (ML) strategies that can easily be applied to the system of equations, e.g., gradient descent based methods.
There are various options for the approach to solving the reverse Ising Problem:

- The auxiliary spins are binary variables. However, they could be treated as continuous.
- It is possible to solve for a feasible set of auxiliary spins first, and then solve the resulting linear system for the local biases and coupling strengths. Alternatively, it is possible to solve for both sets of unknown simultaneously (these are typically quadratic or cubic systems).
- Some approaches may look at all possible combinations of the auxiliary spins simultaneously. Others may try and move around the auxiliary spin space, using simulated annealing or other such methods.

This project involves developing and implementing different ML strategies, and then benchmarking their performance on various operations and scaling capabilities.

2.3 Exploring Alternate Optimization Methods

The reverse Ising problem can be formulated as an optimization problem with penalty function that is a function of the Hamiltonians. The majority of our current optimization approaches rely on gradients, or gradient based approaches. Are there other optimization approaches, e.g., using the higher dimensional gradient information, that can be employed? How does the performance of these methods compare to others that have been implemented?

One Approach: Unconstrained Optimization Using the Hessian

UnconOpt is a C++ program that uses information about the gradients to solve the reverse Ising problem as an unconstrained optimiziaton problem.

- Create an optimized C version of fme.py to generate UnconOpt inputs faster.
- Implement and optimize the explicit Hessian calculation is UnconOpt.
- Write an optimized multi-precision Newton solver in C using gmp.
- Create a multi-precision hyper-dual library in C using gmp.
- Implement UnconOpt in the domain of the hyper-duals.
- Calculate points on the Julia set of the hyper-dual multivariate optimization problem.
- Compute the number of unique zeros of the Newton optimization.

2.4 Implementing a Fourier Motzkin Elimination Approach

Fourier Motzkin elimination (FME decomposition) can reduce the number of variables in the system of equations in exchange for simultaneously increasing the number of equations. This increase in the number of equations can be overwhelmingly large. However, due to the symmetry in this particular system of equations, this increase in the number of equations can be significantly reduced. However,

this is only true when the variable eliminations are performed in a particular order. If performed in an incorrect order, the number of equations produced quickly becomes unmanageable.

- What is the correct order to eliminate variables?
- Can this be generalized to any operation?
- How does this approach scale?
- How many variables can be reduced?
- What is the best algorithm and/or software to solve the resulting system?

3 Exploring Random Variable Arithmetic (SNAPDRAGON)

A consequence of typical models of computing is that in order for the output of a computation to be correct, every intermediate computation has to be correct. Thus, the archetypal goal is to design systems with low fault rates. What if instead systems were designed nondeterministically, but with the peculiar property that errors canceled rather than accumulating? Is this theoretically feasible? In other words, does there exist a mechanism whereby we can recover the correct solution to a deterministic problem with some fixed probability even if there are incorrect intermediate results along the way? This is the question motivating this exploration.

We are modeling this mechanism with *random variables*, variables whose values depend on the outcome of random phenomena. Every random variable has a corresponding *probability distribution*, which describes the probability that the random variable is in any particular range of values. *Random variable arithmetic (RVA)* is a system of arithmetic for random variables; it is the study of the properties and manipulations of random variables.

The goal of this exploration is to perform arbitrary operations on a set of random variables and predict and control the dispersion, shape, and central tendency of the output random variable. SNAPDRAGON is a simulation framework that implements the procedure outlined in the previous section via numerical methods. The probability functions of the input random variables are approximated via interpolation (thereby removing any requirement on the input random variable to have closed-form expressions for their probability functions); these interpolation functions are then used to construct the joint input distribution $h(x_1, \ldots, x_d)$ using a Gaussian copula. Numerical integration methods are used to compute the parameters of interest of the output random variable.

3.1 Integral Kernel Transformation

SNAPDRAGON relies on numerical integration. The shape of the integral kernel (aka the integrand) affect the accuracy of the numerical integration. The shape of the integral kernel varies across the parameter space that we are interested in, and for certain subsets of parameters the current numerical integration techniques give accuracy warnings and/or run much slower. Designing and implementing an integral kernel transformation that maintains optimal speed and accuracy of the integration across the entire parameter space is an important part of SNAPDRAGON.

3.2 Post Processing Visualization

A SNAPDRAGON experiment will run numerous optimization problems, and write the output of each optimization problem to a file. We will likely be interested in comparing various subsets of these output files that correspond to certain subsets of the parameter space. Which output files subsets to compare and how to visualize the results are currently undecided. Visualization will likely involve writing code modules to parse all the output files for one (or more) experiment and grab the appropriate data from those files.

3.3 Generation of Generalized Gamma Convolution Marginal Distributions

Generalized Gamma Convolutions (GGC) are a class of distributions that are closed with respect to RVA addition and RVA multiplication of independent random variables. A GGC is defined via a Laplace transform. To work with distributions, we need the probability density function (PDF) and cumulative density function (CDF), it is currently unknown how to map from Laplace transforms to the PDF and CDF. Note that we do not need closed-form expressions for the PDF and CDF

(which may not even exist), we need to be able to generate points form these distributions efficiently and accurately.

3.4 Ising Generated Joint Probability Distribution

The output of an operation performed on an Ising machine simulator has a distribution (since the Ising machine is not deterministic, wrong answers occur with some probability).

- Is it possible to use the the output distributions from the Ising simulator as marginals (input distributions) for SNAPDARAGON?
- Is it possible to obtain the output distribution that we see in the Ising simulator using SNAPDRAGON? (i.e., is there a combination of marginals and dependence structures that allow us to model the Ising simulator with SNAPDRAGON?)

4 Models of Computation

Models of computation, such as Turing Machines, are theoretical machines designed to simulate physical computers. They enable theoretical exploration of nondeterministic computing.

4.1 Mutable Tape, Turing Machines

Random variable arithmetic (RVA) is a theoretical exploration into whether it is feasible to use nondeterministic models of computation to solve deterministic problems. There do exist novel nondeterministic computer architectures, but these tend to be application specific. Helping bridge the gap between the theoretical world of RVA and actual general purpose nondeterministic hardware (that to our knowledge doesn't yet exist) are models of computation. Models of computation, such as Turing Machines, are theoretical machines designed to simulate physical computers. Creating a RVA model of computation, i.e., mapping RVA onto a new or existing model of computation, is thus a step towards building a nondeterministic computer, albeit still a theoretical step.

A Mutable Tape, Turing Machine (MTTM) is a Turing Machine with nondeterminism in the tape (other Turing Machines only allow for nondeterminism in the state transitions of the finite state machine). The tape can be viewed as an infinite list of cells. Each cell on the tape has a probability distribution associated to it that governs the probability of a value appearing on that cell of the tape at any instant in time.

- Randomly generate errors on the tape for both the MTTM (and MTPTM). Compare the results to the outputs from a PTM.
- Experiment with making the tapes dynamic over the course of an operation.
- The tape of the Turing Machine represent memory. The MTTM is a model of computation where the memory get corrupted. There are techniques for data correction. It should be possible to apply those to a MTTM. Is it possible to model hamming code for correcting errors (those codes were designed to correct memory errors)?
- Correlate the input values on the tape of a MTTM. The MTTM will then model operations on dependent random variables.
- Simulate a different approach to modeling mutable tape. Some possible approaches:
  - Friendly and fickle aliens (see next section)
  - Model the mutable tape by applying a transformation to values when reading them from the tape. This will leave the input tapes unchanged, the transitions occurs in the state machine.
  - Similar to how we model PTMs, create a transition matrix for each cell. This allows every cell to have a different distribution.
  - Group cells together to form "words." Each word has a probability distribution associated to it. Thus, each word is a set of correlated cells.

Two Approaches to Modeling Mutable Tape

Friendly Aliens One approach to modeling mutable tape is to imagine a happy little alien in every cell on the tape. Every clock cycle, each aliens replace the value on their cell with another value drawn randomly from the distribution of values corresponding to that cell (they sample a new sock from the sock drawer every clock cycle). In other words, the probability of value appearing on the tape corresponds to number of times it appears on the tape during some fixed amount of time.

Fickle Aliens Another approach to modeling mutable tape is to imagine a fickle alien in every cell. However, the aliens don't switch the value on the cell at every clock cycle, instead on each clock cycle they have some probability of replacing the value on the tape, but they have some probability of doing nothing. (The fickle aliens sample a new sock from the sock drawer after a random number of clock cycles). In other words, the probability fo a value appearing on the tape corresponds to the amount of time that the value remains on the tape (assuming that it appears on the tape). Thus the probability of nothing changing on a clock cycle is a function of the value on the tape.

- What are the effects of these two approaches? Does one of them have more expressive power?
- Do the results of an operation look different when using these two approaches? Are the errors different? For example, if a Turing Machine reads the input tapes multiple times during a computation, is the fickle aliens formulation more fault tolerant (since the input tape may remain unchanged over the course of the entire computation)?
- Is one of these approaches better suited to model correlation between the cells on the tape?

4.2 Fault Tolerant, Probabilistic Turing Machine

Fault tolerant probabilistic Turing Machines explore reliable nondeterministic computing. Probabilistic Turing Machines (PTMs) are Turing Machines for which "correct" state transitions in the finite state machine (FSM) occur with probability less than one. Incorrect state transitions are logic faults – faults that occur during an operation (as opposed to memory faults that occur in memory). The question of interest is whether it is possible to design and implement fault tolerance in a PTM.

- The mtm scripts need debugging (see Alex Craig's documentation). Global start and halt states need to be added, along with the error checking to ensure that the TM simulator doesn't prematurely halt.
- Currently, it is not possible to know how many times the TM restarts (using Ryan's scripts). Can we count how many times we get wrong answer?
- Currently naf_validate is purely deterministic. Add the capability for this to be probabilistic.
- Run an experiment to determine the comparative error rates when mult.tm is run with and without naf_validate.
- Implement different fault tolerant mechanisms: add states to the Turing Machine perform basic error checking, and restart a calculation if one of these checks fails. (See Ryan Mcdowells final presentation/writeup for more details).
  - Checking the signs of the factors and comparing it with the sign of the product (this is possible in NAF by examining the sign of the most significant digit).
  - Checking if the product is zero if factors are nonzero.
  - Checking if the answer is in proper NAF format at the end of the computation.
- Other fault tolerant methods: Adding redundancy:
  - Is it possible to run multiple identical PTMs and have them vote on each step?
  - "Another solution would be to have multiple smaller TMs, each computing a smaller part of the step, but with redundancy between them (just like the bits in a Hamming code have redundancy)." (Alex Craig)
- Can we run the PTM in lockstep with the TM to catch errors? This will "insert" some transparency about where the errors are happening.

- Fault tolerance adds additional complexity, and thus creates the opportunity for new faults. So, when the probability of a fault occurring becomes too large the fault tolerance may actually increase the number of errors created by the PTM (see Figure 1). What is the fault probability 'threshold' above which the fault tolerance increases the error rate?

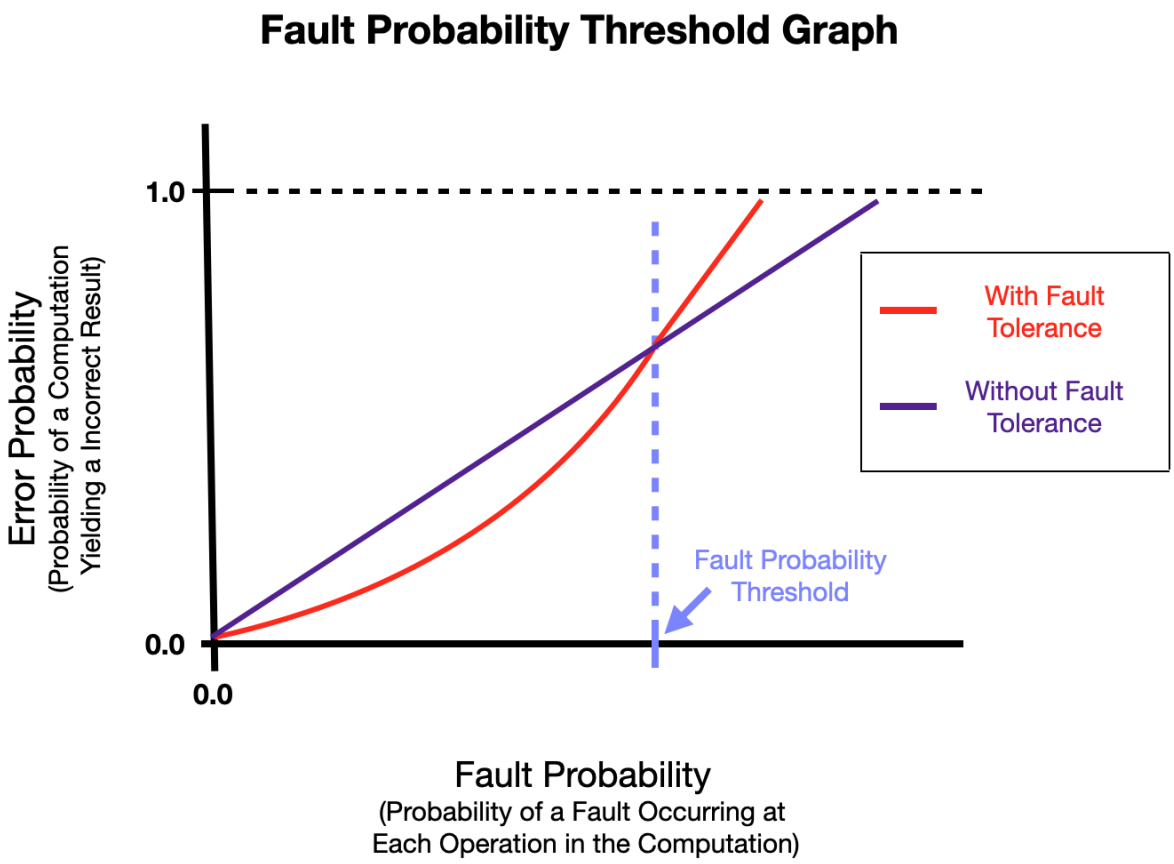

Figure 1: *A representation of the relationship between faults and errors in a model of computation. When there are no faults there are no errors; when there are faults, errors can occur. A given fault tolerance mechanism can detect and correct some proportion of the faults that occur during a computation. However, a fault tolerance mechanism also adds complexity to the system (e.g., system states, hardware resources) and, just as the rest of the system, these additional components are subject to faults. At sufficiently large fault probabilities the increase in the number of faults generated due to this increased complexity outweighs the number of faults that the fault tolerance mechanism can correct. The fault probability value at which this transition occurs is the fault probability threshold. In other words, a given fault tolerance mechanism can detect and correct up to some maximum number of faults. When the fault probability is greater than the fault probability threshold, faults are being introduced into the computation at a rate that is faster than they can be corrected or faults are being introduced into the computation at such a rate that they compound and become undetectable to the fault tolerance mechanism.*
The purple line depicts a possible relationship between the fault and the error probabilities on a system (for a given computation). In actuality this relationship may not be a linear: in this example, linearity is based on the assumption that a fixed proportion of faults propagate into errors. (Note that this curve is also a function of the particular computation performed.)
The red curve depicts the standard characterization of how the relationship between the fault and the error probabilities changes when the system is augmented with a fault tolerance mechanism (for the same computation). For fault probabilities lower than the fault probability threshold, the system with the fault tolerance mechanism has a lower error probability than the system with no fault tolerance mechanisms. However, for fault probabilities higher than the fault probability threshold, the added complexity of the fault tolerance mechanism increases the error probability when compared to the system with no fault tolerance mechanisms.

5 Optical Devices

A goal of this exploration is to determine the feasibility of using optical devices to build an arithmetic logic unit (ALU). In particular, the question of interest is whether there is improvement in computational efficiency (operations per Joule) in using optical devices to perform modular exponentiation compared using traditional transistor-based computers.
In order to take advantage of any new technology it is necessary to leverage its strengths. This may require revising the algorithms used to perform a task. One of the potential advantages of using an optical device is that Fourier transforms (and inverse Fourier transforms), which are computationally expensive on transistor-based computers, are basically free in optics.
Modular exponentiation involves performing numerous multiplications, which are typically an $O(n^2)$ operation. However, by converting the factors of a multiplication problem into functions and taking their Fourier transform, the operation is transformed into a bit-wise multiply, an $O(n)$ operation, in Fourier space.
A *Fourier Optics Multiplication Accelerator* utilizes the features of optical devices to convert regular multiplication into pointwise multiplication in the Fourier space: it takes two inputs, converts them into the Fourier space via a Fourier transform, performs a pointwise multiply on these transformed factors by adding the light intensities, and the converts them back out of the Fourier space via the inverse Fourier transform. The final output is not in binary, since the pointwise multiply is does not perform any carries.

5.1 Developing Efficient Optical to Electronic Conversions

The output from the Fourier Optics Multiplication Accelerator is not in binary, but the result of a convolution in Fourier transform space. It is necessary to efficiently transform output of the convolution over the reals into digital electronic binary – without sacrificing accuracy or the energy efficiency gains from the optical computation. This is complicated by the fact that implementing these methods in the optical hardware may be both novel and nontrivial.

5.2 Parameterizing the Fourier Optics Multiplication Accelerator

The particular implementation of modular exponentiation used relies on Montgomery multiplication. During the computation, a mask is used to separate a number into two strings of digits corresponding to the low and high bits of the number. However, there is some number of *overlap bits* that need to appear in both the high-bit and low-bit digit strings to ensure that information isn't lost. (This is a consequence of the fact that pointwise multiplication is not binary, and the low-bit string may consequently contain information that belongs in the high-bit string.) The number of overlap bits needed for any particular computation are unknown.

5.3 Error Distributions on the Fourier Optics Multiplication Accelerator

Since the Fourier Optics Multiplication Accelerator is a new device, the types of errors that it produces are unknown. This is important information because error correcting mechanisms are designed for particular errors, and thus the type of error mechanism are device specific. This type of exploration would probably occur mostly on a simulator.

- What type of errors occur on the Fourier Optics Multiplication Accelerator? Do they have a known distribution?

- What types of fault tolerance and error correction mechanisms address these types of errors? Are there any fault tolerance and error correction mechanisms that are a definite bad choice?
- How would these fault tolerance and error correction mechanisms be implemented? Would they be present in hardware, or occur after the transition back to digital after the computation has finished?

5.4 Implementing the Guard Approach to Error Detection

A known type of undetectable fault that can occur in the Fourier Optics Multiplication Accelerator is that the filters can become misaligned. A potential error correction strategy to address this is the use of *guard approach*. This strategy consists of premultiplying the factors and constants in the Montgomery multiplication by a guard value g (a positive prime number). Then because of the distributive properties, the result of the computation should be divisible by g. If it isn't, an error has occurred at some point during the computation. The result is divided by g to obtain the desired output. "Note that ... the guard approach is not guaranteed to detect all errors. It is a probabilistic correctness check that trades off accuracy for performance. It is believed that the probability of a false positive using this approach is proportional to 1/g, however that has not be definitively proven."[2]

- What is the best way to implement this strategy in the Fourier Optics Multiplication Accelerator? Is it possible and better to implement it completely in optics, or in a combination of optical and electronic components?
- What are the effects of adding the extract hardware required? Do they outweigh the benefits from the error correction mechanism? Are there other trade-offs?
- How large do the guard values need to be? There are hardware downsides to increasing the size of the guard value.
- How effective is this error detection strategy? The paper postulates that is proportional to $1/g$. Is this supported empirically for the Fourier Optics Multiplication Accelerator?

6 Matrix Compression

Matrices, and matrix arithmetic, are an important component of many applications in physics, computer science, and statistics. Efficient and effective matrix compression, as well as how methods to effectively and efficiently perform operations on the compressed matrices, are thus topics of interest.

6.1 Recursive Matrix Compression

Compressed Fractal Array (CFA) is a compression scheme that relies on Hilbert curves. Further, when operations are implemented using a block recursive pattern, CFA allows the computation to be performed efficiently on matrices in the compressed form. This project involves exploring different types of problems and computer architectures in order to benchmark the performance of CFA compared to other compression schemes.[3]

[2] Quoted from "A Fault-Tolerant Modular Power Function (PMOD)", Daly, Monroe, Timmel, Mead, section III.B
[3] Refer to "Fractals, Matrices, and Parallelization- A New Approach to Matrix Arithmetic", Andrew Searns

6.2 Nearest Kronecker Product Approximations

Nearest Kronecker Product (NKP) approximation involves rewriting a matrix in terms of Kronecker products of lower-dimensional matrices. For example, suppose M is a matrix with dimensions $(n_1 n_2) \times (m_1 m_2)$. Let $\otimes$ denote the Kronecker product, and let $\| \ \|_F$ denote the Frobenius norm. Then it is possible to approximate M as the Kronecker product of A_1 and A_2, with dimensions $n_1 \times m_1$ and $n_2 \times m_2$, respectively.[4] That is,

$$M \approx A_1 \otimes A_2 \tag{2}$$

such that A_1 and A_2 minimize

$$\left\| M - A_1 \otimes A_2 \right\|_F \ . \tag{3}$$

Motivation

NKP approximation offers several potential benefits:

Conditioning Matrices NKP approximation offers a strategy to build better conditioned matrices for dense linear algebra problems. For example, if $M \approx A_1 \otimes A_2$ then $A_1^{-1} \otimes A_2^{-1}$ is potentially a effective preconditioner (reference).

Compression A square matrix C with n rows has n^2 elements. If this matrix is approximated with the Kronecker product B_1 and B_2, with dimensions $n \times 1$ and $1 \times n$ respectively, then the approximation has only $2n$ elements. In other words, the number of elements of C scales quadratically, while the number of elements of the approximation scale linearly.

CFA Compression Kronecker products are easily compressed using CFA compression. Thus it is worthwhile to accurately construct a NKP approximation matrix for arbitrary matrices. Further, the particular form that the NKP approximation takes may be an indicator for the amount of compression possible.

Objectives

There are two important parameters of the NKP approximation. The first is the accuracy of the approximation; the second is the amount of compression enabled by the approximation relative to that achievable on the original matrix. The objective is the maximize both of these parameters. Possible approaches to improving each of these parameters are now discussed.

Improving the Accuracy of the Approximation

Let M be an $n \times m$ matrix. Suppose C_1 and C_2 are matrices of dimension $n \times k$ and $k \times m$, where $k \leq \min(m, n)$, such that

$$M \approx C_1 C_2 \ . \tag{4}$$

(Note that C_1 and C_2 are multiplied using standard matrix multiplication.) Then, $(C_1 C_2)$ is a rank k approximation of M. Further $(C_1 C_2)$ has $k(n+m)$ elements, while M has nm elements. It follows that the amount of information lost, i.e., the accuracy of the approximation, decreases as k increases.

[4] For the reference on how to do this, see "Approximation with Kronecker Products",Van Loan and Pitsianis

It is possible to write C_1C_2 as the sum of k Kronecker products:

$$C_1C_2 = \sum_{r=1}^{k} K_{r,1} \otimes K_{r,2} \ . \tag{5}$$

Note that the number of sums in this sum is equal to the rank of (C_1C_2). Thus, a rank k approximation of M is equivalent to a sum with k terms, where each term is a Kronecker product of two matrices.

Further, there is flexibility in the choice of the dimensions of the K's.Suppose $n = n_1n_2$ and $m = m_1m_2$. It is possible to rearrange the matrixes involved such that $K_{r,1}$ and $K_{r,2}$ have dimensions $n_1 \times m_1$ and $n_2 \times m_2$, respectively. [5]

- In equation 2, A_1 and A_2 are guaranteed to minimize the error of the approximation. However, there is no guarantee that the minimum error is zero. Increasing the rank of the approximation (i.e., increasing the number of terms in the sum) should increase the accuracy of the approximation. What is the minimum rank (number of terms in the sum) required to exactly represent the value of M, i.e., to cause the value of the error to be zero? How does this minimum rank scale as the dimensions of M increase?
- Is the kronecker product approximation easier to calculate than the original approximation (C_1C_2)? How do we calculate the Kronecker product approximation?

Improving the Achievable Compression of the Approximation

Suppose M is a matrix with dimensions $(n_1n_2) \times (m_1m_2)$, where $n_1, n_2, m_1, m_2 \geq 1$. Then M could be approximated with either of the following Kronecker products:

$$(M)_{(n_1n_2)\times(m_1m_2)} \approx (A_1)_{(n_1n_2)\times 1} \otimes (A_2)_{1\times(m_1m_2)} \tag{6}$$

$$(M)_{(n_1n_2)\times(m_1m_2)} \approx (B_1)_{n_1\times 1} \otimes (B_2)_{1\times m_1} \otimes (B_3)_{n_2\times 1} \otimes (B_4)_{1\times m_2} \tag{7}$$

The original matrix M has $n_1n_2m_1m_2$ elements. The approximation in equation 6 has $n_1n_2+m_1m_2$ elements. The approximation in equation 6 has $n_1+n_2+m_1+m_2$ elements. This demonstrates that increasing the number of matrices in the Kronecker product increases the compression possible.

- Are these two approximations equivalent approximations of M? In other words, can the elements of the B_i's be expressed as a function of the terms of the A_i's?
- Increasing the number of matrices in the Kronecker product increases the amount of compression possible. Does it affect the accuracy of the approximation?
- When there are only two matrices in the Kronecker product, they can be calculated using the SVD approach in "Approximation with Kronecker Products",Van Loan and Pitsianis. How are they computed when there are more than two matrices? Is recursion the best approach?

Next Steps

Combining the previous two ideas, M could be approximated by the following equation:

$$M \approx \sum_{k=1}^{r} \bigotimes_{j=1}^{n} A_{k,j} = A_{1,1} \otimes \cdots \otimes A_{1,n} + \cdots + A_{r,1} \otimes \cdots \otimes A_{r,n} \tag{8}$$

where $r, n \geq 1$.

[5] This is theorem 2.1 in "Approximation with Kronecker Products",Van Loan and Pitsianis.

- How do r and n interact? In other words, does increasing the number of matrices (n) in each Kronecker product reduce the minimum rank (number of terms, r) required to cause the approximation to exact?

7 Miscellaneous

7.1 Applications of Analog Content Addressable Memory

Analog content addressable memory (ACAM) is memory that returns, when passed a value, returns all the adresses of that value in memory. This is in contrast to random access memory (RAM), which when passed an address returns the value at that address.

Some potential applications for ACAM (that are related to SNAPDRAGON):

Numerical integration We compute the probability density function of the output random variable via a multidimensional integral. Implementing numerical integration techniques that are both accurate and efficient is thus an important aspect of the problem. Currently we are using quadrature techniques, we plan to transition to Monte Carlo methods. These integration techniques, especially Monte Carlo methods, involve generating tremendous numbers of sample points from the integrand. This is a potential application where architecture with content addressable memory may be more efficient.

Linear interpolation We are currently using linear interpolations to approximate the probability functions of the input random variables. This approach gives us a suitable balance of accuracy and efficiency. However, for certain operations and input random variables, up to half of the time in the numerical integration routine is spent calling and evaluating these linear interpolation functions. While contingent on the specific approach to numerical integration (more accuracy may be required), linear interpolation is a potential application where architecture with content addressable memory may be more efficient.

Calculating median and MAD We are parameterizing central tendency with the median. We calculate $\widetilde{\mu}_Y$, the median of the output random variable Y, by working with F_Y, the cumulative distribution of Y. However, another approach would be to use the inverse of the cumulative distribution function of Y:

$$\widetilde{\mu}_Y = F_Y^{-1}\left(\frac{1}{2}\right) \ . \tag{9}$$

Like linear interpolation, this is a problem that architecture with content addressable memory may be more efficient at solving since it involves a lookup of the number corresponding to a specific function value.
We compute the median and MAD of the output random variable by first calculating cumulative distribution functions. However, it is possible to compute both of these values from the probability density function of the output random variable. It is possible that with different architectures, this approach is easier than the cumulative distribution approach.

- Are some of these applications better or worse for ACAM? Why?
- Are there other applications for ACAM that are less project-specific?