Generation and Prediction of Markov Processes

Joshua B. Ruebeck[*]
Physics Department, Carleton College and
Complexity Sciences Center and Physics Department,
University of California at Davis, One Shields Avenue, Davis, CA 95616

(Dated: August 24, 2016)

We present the minimal generators for all binary Markov chains and calculate C_g, their generative complexity. While this is in general a non-convex minimization that must be calculated numerically, we are able to find an analytical solution for this particular set of processes. We find that generically, but not universally, processes in this class have strictly less generative complexity than statistical complexity ($C_g < C_\mu$). We then examine other properties of the minimal generators, including their crypticity, oracular information, and gauge information, and compare the minimal generators to minimal predictors. These results, besides being interesting in their own right, also suggest properties of minimal generators that may generalize and aid in finding the minimal generators of more complex processes.

INTRODUCTION

In real-world examples of stochastic processes, there exist some that are simultaneously being generated by one agent and predicted by another. For example, a mouse being chased by a fox desires to choose a path that is difficult for the fox to predict. However, the mouse itself has limited computational resources, and so would also like the path to be easy to generate. Are these different tasks, and if so, do there exist processes that are easier to generate than predict?

We are able to address these questions quantitatively through the use of hidden Markov models (HMM). The statistical complexity, C_μ, of a process is the state-entropy of its minimal predictive HMM, also called the ϵ-machine. C_μ quantifies how hard a process is to predict. The ϵ-machine representation of the process is well-studied and can be constructed for arbitrary processes [1, 2]. The generative complexity, C_g, of a process is the state-entropy of its minimal generative HMM. This is in general much harder to calculate, as it involves a non-convex constrained minimization over high-dimensional spaces. There are some known bounds on C_g and restrictions on the construction of generative HMM's [3], but this area has received significantly less attention than the predictive case and is less well understood.

In this paper, I present a construction of the minimal generators for an arbitrary stationary binary Markov process. This allows the calculation of C_g as well as investigation into other properties of generative models. By comparing minimal generators with minimal predictors, we can begin to understand the differences between the tasks of generation and prediction and where one type of model might be advantageous over the other.

COMPUTATIONAL MECHANICS BACKGROUND

Elements of information theory

Let Z be a random variable (RV) with alphabet $\mathcal{A}$ and a probability mass function $p(z)$ for $z \in \mathcal{A}$. The (Shannon) entropy of Z is defined as [4]

$$H(Z) = -\sum_{z\in\mathcal{A}} p(z)\log_2 p(z). \tag{1}$$

Entropy can have different interpretations in different contexts. Roughly, it quantifies the amount of uncertainty present in the random variable and can often be interpreted as the amount of information contained. Another useful quantity is the conditional entropy,

$$H(Z|Z') = -\sum_{z\in\mathcal{A}}\sum_{z'\in\mathcal{A}'} p(z,z')\log_2 p(z|z'), \tag{2}$$

where $p(z,z')$ is the probability of z and z', and $p(z|z')$ is the probability of z given $Z' = z'$. The conditional entropy describes the uncertainty in Z given knowledge of a second random variable Z'. Using these two tools, we can define the mutual information of two random variables:

$$I(Z;Z') = H(Z) - H(Z|Z'). \tag{3}$$

In other words, the mutual information is the difference in uncertainty between Z with no prior knowledge and Z given knowledge of Z'. It usually represents the amount of information shared between two random variables. These quantities can be represented in a Venn-like diagram called an information diagram, with the size of various circles representing the entropy of a process, and the size of their intersections corresponding to the mutual

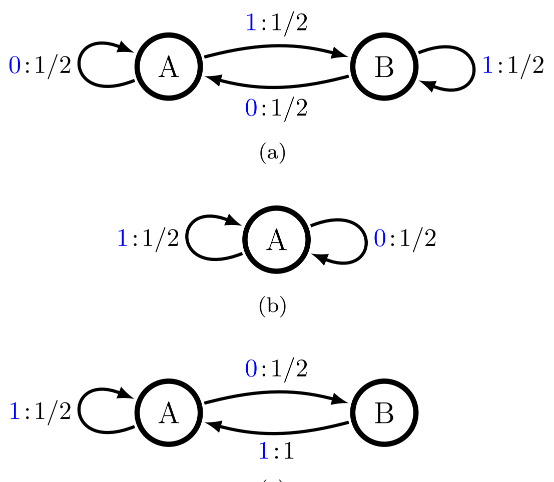


FIG. 1. Model representations of the Fair coin (a, b) and Golden Mean (c) processes. Models (b) and (c) are the ϵ-machine representations of their respective processes.

information.

Processes and hidden Markov models

A stochastic process over an alphabet $\mathcal{A}$ is a bi-infinite sequence of random variables

$$\dots X_{-2} X_{-1} X_0 X_1 X_2 \dots \tag{4}$$

characterized by a joint probability distribution over all strings $\dots x_{-2} x_{-1} x_0 x_1 x_2 \dots$ with $x_t \in \mathcal{A}$. A Markov chain is a process where this probability distribution can be factored into single-symbol conditional distributions:

$$\begin{aligned}\Pr(\dots X_{-1} X_0 X_1 X_2 \dots &= \dots x_{-1} x_0 x_1 x_2 \dots) \\ = \dots &\Pr(X_0 = x_0 | X_{-1} = x_{-1}) \\ &\times \Pr(X_1 = x_1 | X_0 = x_0) \\ &\times \Pr(X_2 = x_2 | X_1 = x_1) \dots \end{aligned} \tag{5}$$

In other words, the probability distribution of the RV at time t only depends on the previous symbol generated. A time-invariant Markov chain additionally satisfies $\forall t \in \mathbb{Z}$ and $\forall x_0, x_1 \in \mathcal{A}$

$$\Pr(X_t = x_1 | X_{t-1} = x_0) = \Pr(X_1 = x_1 | X_0 = x_0). \tag{6}$$

When I refer to a Markov chain (or synonymously, a Markov process), time invariance is implicit. For two examples of Markov processes, we can take the Fair Coin and the Golden Mean processes. The Fair Coin is generated by flipping a coin at every time step, and writing down a 1 for every heads and a 0 for every tails. A string generated by the process might look like

$$01110110101011101010100100 1011. \tag{7}$$

The Golden Mean has a little more structure: If the previous symbol was a 1, then a coin is flipped as in the Fair Coin. If the previous symbol was a 0, then the next symbol must be a one. A sample string looks like

$$0111010111011101110101111 10111. \tag{8}$$

Clearly there can never be more than one consecutive 0. Markov processes can all be characterized uniquely by their transition matrix, which describes the probability distribution of the current symbol (at time t) given the last symbol (at time $t-1$). For the Fair Coin and Golden Mean, they are

$$p_{X_t|X_{t-1}} = \begin{array}{c c} & \begin{array}{cc} 0 & 1 \end{array} \\ \begin{array}{c} 0 \\ 1 \end{array} & \begin{pmatrix} 1/2 & 1/2 \\ 1/2 & 1/2 \end{pmatrix} \end{array} \tag{9}$$

and

$$p_{X_t|X_{t-1}} = \begin{array}{c c} & \begin{array}{cc} 0 & 1 \end{array} \\ \begin{array}{c} 0 \\ 1 \end{array} & \begin{pmatrix} 0 & 1 \\ 1/2 & 1/2 \end{pmatrix} \end{array}, \tag{10}$$

respectively. So, for example, we can read off that for the Golden Mean, the probability of seeing a 1 given a previous 0 is 1, while the probability of seeing a 0 given a previous 0 is 0. If the previous symbol was a 1, then there is equal probability 1/2 of seeing either symbol next.

We can also represent each process by a model (machine) representation, shown in Fig. 1. To generate a process from a machine, choose a state to start in. Each edge is marked as 'symbol:probability.' From a state, you can move along an edge with the labeled probability while emitting the given symbol. These representations are not unique, as emphasized by the two representations of the fair coin shown.

The information diagram in Fig. 2 is a useful way of visualizing properties of these models. It describes the amount of information contained in and shared between the process's past, the model, and the process's future. The size of each RV's circle is given by its Shannon entropy. We can define a generator (which I will also refer to as model) as any random variable R that contains all of the region labeled **E**. This means that any information shared between the past and future is contained in the model. A predictor is additionally constrained to have

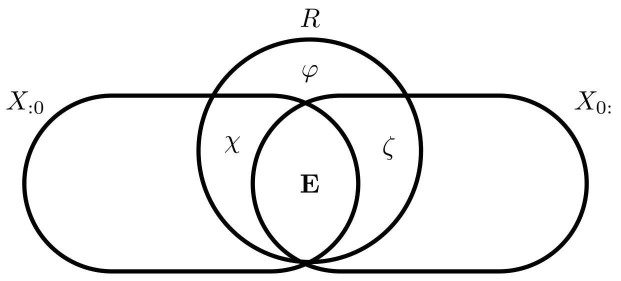


FIG. 2. An information-diagram of a process's past $X_{:0}$ and future $X_{0:}$ with a model R. The quantities labeled are the excess entropy $\mathbf{E} = I(X_{:0}; X_{0:})$, crypticity $\chi = I(X_{:0}; R|X_{0:})$, oracular information $\zeta = I(X_{0:}; R|X_{:0})$, and gauge information $\varphi = H(R|X_{:0}, X_{0:})$.

$\varphi, \zeta = 0$. The minimal predictor is the smallest predictor (entropy-wise), and has been well studied as the ϵ-machine of the process [1, 2]. The ϵ-machine can be efficiently constructed for any process, and the statistical complexity C_μ of a process is defined as the state-entropy of its ϵ-machine. To calculate state-entropy, one can let a process run for a long time and then take the Shannon entropy of the resulting distribution over time spent in each state (or do the equivalent analytically).

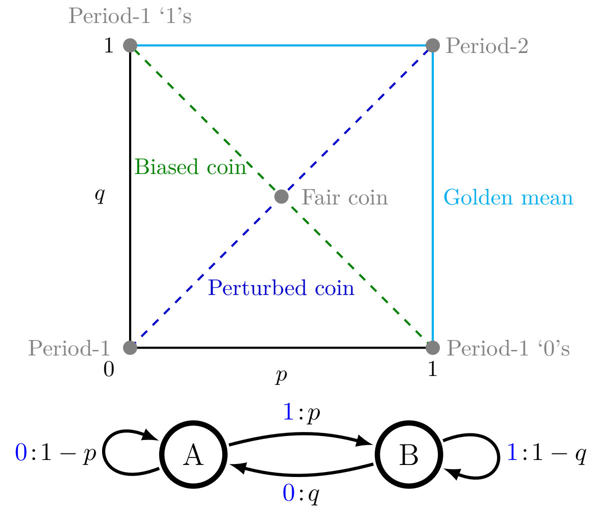


FIG. 3. The process space and ϵ-machine for binary Markov chains parametrized by $p, q \in [0, 1]$. Edge cases and special cases are labeled in color. The Golden Mean shown along two edges of the space is a generalization of the Golden Mean presented in Fig. 1 and Eq. 10

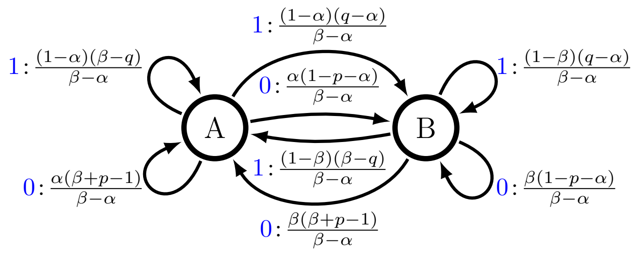

$1: \frac{(1-\alpha)(q-\alpha)}{\beta-\alpha}$
$1: \frac{(1-\alpha)(\beta-q)}{\beta-\alpha}$
$0: \frac{\alpha(1-p-\alpha)}{\beta-\alpha}$
$1: \frac{(1-\beta)(q-\alpha)}{\beta-\alpha}$
A
B
$0: \frac{\alpha(\beta+p-1)}{\beta-\alpha}$
$1: \frac{(1-\beta)(\beta-q)}{\beta-\alpha}$
$0: \frac{\beta(1-p-\alpha)}{\beta-\alpha}$
$0: \frac{\beta(\beta+p-1)}{\beta-\alpha}$

FIG. 4. $\alpha \in [0, \min(q, 1-p)]$ and $\beta \in [\max(q, 1-p), 1]$ give the complete set of 2-state machines that generate a binary Markov chain given by p, q.

GENERATIVE COMPLEXITY: THE GENERAL PROBLEM

The generative complexity is defined as

$$C_g = \min_{X_{:0} \to R \to X_{0:}} H(R). \tag{11}$$

Here $X_{:0} \to R \to X_{0:}$ indicates that the past, model, and future form a Markov chain, which is equivalent to the condition that R contains all of $\mathbf{E}$:

$$\begin{aligned} &\Pr(X_{:0}, R, X_{0:}) = \Pr(X_{:0}) \Pr(R|X_{:0}) Pr(X_{0:}|R) \\ &\qquad\qquad\qquad \Longleftrightarrow \\ &I(X_{:0}, X_{0:}|R) = 0. \end{aligned} \tag{12}$$

The model R that achieves Eq. 11 is a minimal generator of the process. In general, this minimization is nonconvex, making it difficult to find the global minimum with certainty. Additionally, it takes place in many dimensions with many constraints—the simplest case being 8 dimensional with 6 constraints. The size of these problems means that, for a generic process, they must be solved numerically. Development of this code is underway, but has been halted by additional complications arising from extra constraints that were not initially considered. Luckily, the simplest case is accessible analytically and allows us to build intuition for the more complicated cases as well as establish heuristics and hypotheses for characteristics of minimal generators in general. Additionally, since these models have largely not been studied up to this point, even this simplest case provides the first example of provably minimal generators as well as the first examples of processes where $C_g < C_\mu$.

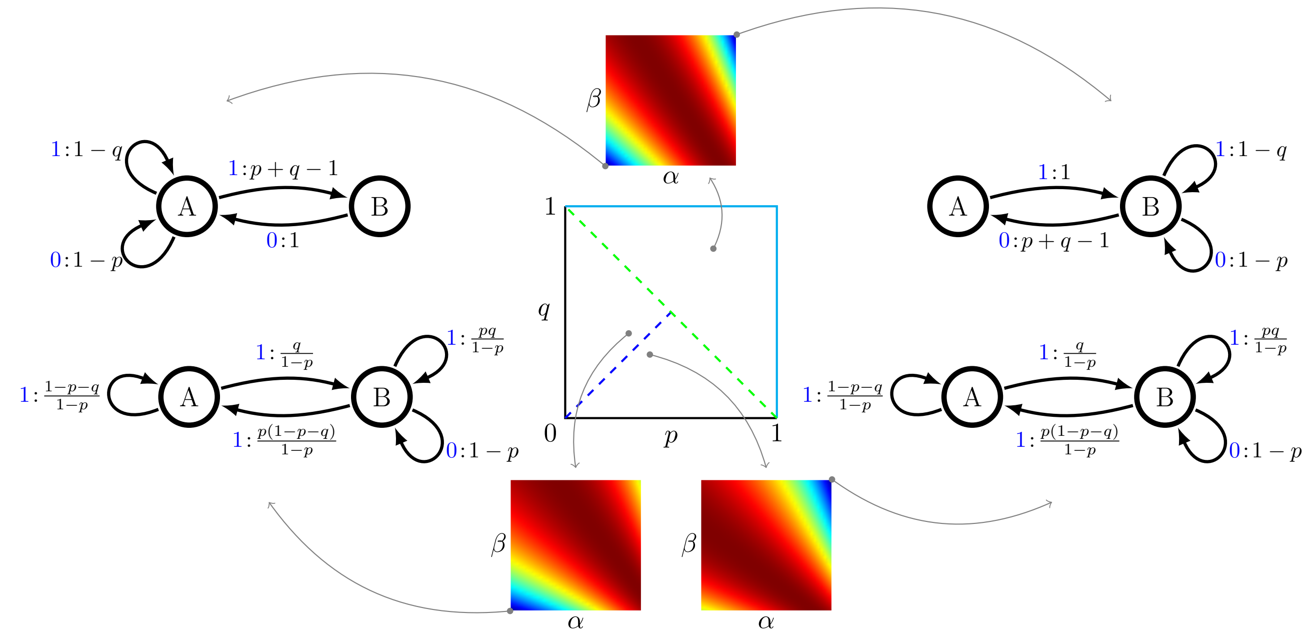


FIG. 5. The minimal generators for all binary Markov chains. The three regions of the process-space (parametrized by p, q) result in three different cases for minimization. A characteristic model-space (parametrized by α, β) is shown, with a heat-map of the entropy $H(R)$ superimposed. Each global minimum (of which there are two in the top case) results in a minimal generator, connected to it by an arrow.

RESULTS

Analytical solution for Binary Markov chains

Any binary Markov process can be characterized by a transition matrix

$$p_{X_t|X_{t-1}} = \begin{array}{c c} & \begin{array}{cc} 0 & 1 \end{array} \\ \begin{array}{c} 0 \\ 1 \end{array} & \begin{pmatrix} 1-p & p \\ q & 1-q \end{pmatrix} \end{array} \tag{13}$$

for $p, q \in [0, 1]$. The matrix is constrained so that all rows add to one (called row-stochastic), which ensures valid probabilities. It has a stationary distribution

$$p_{X_t} = \begin{array}{c} \begin{array}{cc} 0 & 1 \end{array} \\ \begin{pmatrix} \frac{q}{p+q} & \frac{p}{p+q} \end{pmatrix} \end{array}, \tag{14}$$

which describes the fraction of 0's and 1's emitted by the process. The stationary distribution is found by taking the left-eigenvector of $p_{X_t|X_{t-1}}$ that has eigenvalue 1. We can construct its ϵ-machine for $p \neq 1 - q$, shown in Fig. 3. For $p = 1-q$ (the line labeled "Biased Coin"), the ϵ-machine is a single-state machine that simply has the probability of seeing a 0 or 1 as its edges. This discontinuity is termed "causal collapse," since there is no longer any dependence on the previous symbol; each symbol is independent and identically distributed and the process can now be entirely characterized by its stationary distribution $p(0) = p$, $p(1) = 1 - p$. In the same figure is a visualization of the space of binary Markov chains; special cases of interest are marked, including the Fair Coin and a generalization of the Golden Mean already discussed.

In order to find the minimal generator, we first enumerate all possible generators of the process. It can be shown [3] that in this case, the minimal generator will be a two-state model. From either expression of Eq. 12 along with row-stochasticity, we can describe any model of this process with two transition matrices:

$$p_{X_t|R} = \begin{array}{c c} & \begin{array}{cc} 0 & 1 \end{array} \\ \begin{array}{c} A \\ B \end{array} & \begin{pmatrix} \alpha & 1-\alpha \\ \beta & 1-\beta \end{pmatrix} \end{array} \tag{15}$$

and

$$p_{R|X_{t-1}} = \begin{array}{c c} & \begin{array}{cc} A & B \end{array} \\ \begin{array}{c} 0 \\ 1 \end{array} & \begin{pmatrix} \frac{\beta-(1-p)}{\beta-\alpha} & \frac{(1-p)-\alpha}{\beta-\alpha} \\ \frac{\beta-q}{\beta-\alpha} & \frac{q-\alpha}{\beta-\alpha} \end{pmatrix} \end{array}. \tag{16}$$

The eight elements of these matrices are the eight dimensions mentioned earlier; four of the constraints come from row-stochasticity, while two come from Eq. 12.

From these transition matrices, we can calculate the machine representation as shown in Fig. 4. It is worth re-emphasizing at this point that p, q parametrize our *process space* while α, β parametrize our *model space*. For a given p and q, scanning through values of α and β will result in the same process but in a different model of that process. With these representations, we can now calculate the state-entropy to be

$$\begin{aligned} H(R) = \Bigg[&(q-\alpha(p+q))\log_2\left(\frac{\alpha(p+q)-q}{(\alpha-\beta)(p+q)}\right) \\ &+ (\alpha(p+q)-q)\log_2\left(\frac{q-\beta(p+q)}{(\alpha-\beta)(p+q)}\right)\Bigg] \\ &/\left[(\alpha-\beta)(p+q)\right]. \end{aligned} \tag{17}$$

Fig. 5 shows the three different cases that emerge when minimizing $H(R)$. For three representative points in process space, the model space has been plotted as a heatmap of $H(R)$. In one case, we observe that the global minimum is in the bottom left; in another, the top right. For the entire region $p+q>1$, there are two equivalent global minima.

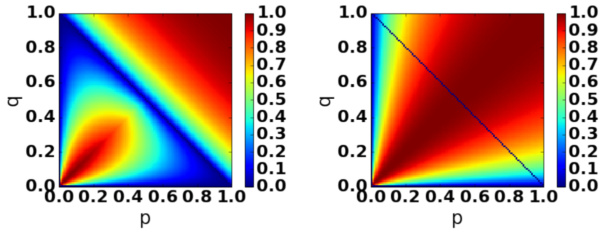
By substituting the values of α, β found in the minimization, we obtain the minimal generators which are shown in the same figure connected to their corresponding minima.

Features of generative models

With these models in hand, we can calculate the generative complexity C_g of a process, which we plot in Fig. 6a. The difference between C_g and C_μ, which we call the predictive overhead, is shown in Fig. 6c. We can also calculate other properties of interest. Since the ϵ-machine is constrained to have $\varphi, \zeta = 0$, we are interested in seeing the behavior of these quantities along with their counterpart χ for the minimal generators. These are plotted in Fig. 7.

DISCUSSION & CONCLUSION

Even in this simplest case, many features emerge that suggest possible general features of minimal generators. First of all, from Fig. 6, we see that for the vast majority of our process space, $C_g < C_\mu$. The only cases in which $C_g = C_\mu$ are along all four edges of the process space (Fig. 3) and along the diagonal $p = 1-q$. Each of these cases involves some kind of loss of support; the edges all have some element of their transition matrix equal to


(a) Generative complexityC_g (b) Statistical complexity C_μ

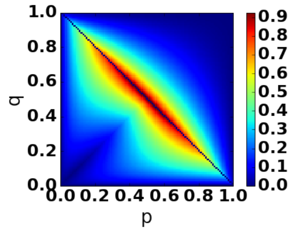
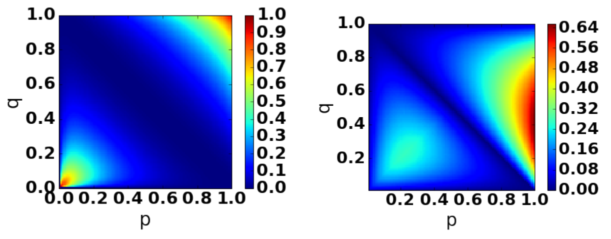

(c) Predictive overhead: $C_\mu - C_g$

FIG. 6. A comparison of the generative complexity and statistical (predictive) complexity. The predictive overhead is highest when p is close but not equal to $1-q$


(a) Excess entropy $\mathbf{E}$ (b) Crypticity χ

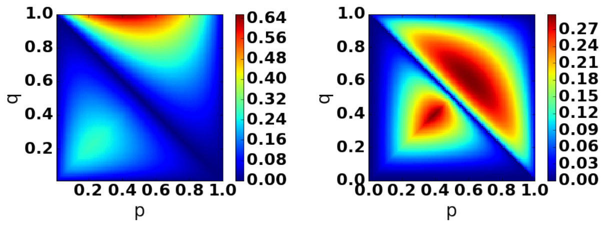

(c) Oracular information ζ (d) Gauge information φ

FIG. 7. The informational breakdown of the state complexity of a minimal generator.

0, and the line $p = 1-q$ is actually independent and identically distributed rather than a true Markov chain. Interestingly, the region of maximum predictive overhead (Fig. 6c) is centered around the Biased Coin ($p = 1 - q$) diagonal. This suggests that processes 'near' causal collapse may in general have a large difference between C_g and C_μ.

Additionally, we notice that the minima observed in model space occur at the extreme allowed values of α

and β. While in general this type of parametrization is not possible, this suggests a place to look for minima, which may increase the efficiency of finding them.

While ϵ-machines are provably unique [1, 2], we have found an example of two non-isomorphic minimal generators for the same process (any process with p+q>1, the upper right region Fig. 3).

Fig. 7 shows that generically, all of the atoms of the minimal generators are nonzero. In the upper-right portion of the plot, the symmetry breaking is due to the fact that there are two minimal generators, and we have only shown the values calculated for one of the two. The other machine's crypticity and oracular information are identical to the oracular information and crypticity (respectively) shown.

Besides demonstrating conclusively the non-equivalence of C_μ and C_g, these minimal generators show a number of traits that provide insight into the generative nature of Markov chains. Additionally, this work lays the groundwork for further numerical pursuit of higher Markov order, larger-alphabet processes.

ACKNOWLEDGMENTS

I would like to thank Prof. James Crutchfield, Dr. John Mahoney and Dr. Ryan James for their guidance on this project. Thanks to Prof. Rena Zieve for coordinating this REU and the NSF for funding this opportunity.

* josh.ruebeck@gmail.com

[1] James P. Crutchfield. Between order and chaos. *Nat Phys*, 8(1):17–24, 2011. 1, 3, 6

[2] Cosma Rohilla Shalizi and James P. Crutchfield. Computational mechanics: Pattern and prediction, structure and simplicity. *Journal of Statistical Physics*, 104(3/4):817–879, 2001. 1, 3, 6

[3] Gowtham Ramani Kumar, Cheuk Ting Li, and Abbas El Gamal. Exact common information. In *2014 IEEE International Symposium on Information Theory*, 2014. 1, 4

[4] Thomas M. Cover and Joy A. Thomas. *Elements of Information Theory*. Wiley-Blackwell, 2005. 1

Appendix A

Besides the minimal generator project, I also worked on writing python code to calculate two quantum information quantities: entanglement of formation and accessible information. These problems are related to the generative complexity problem in that they involve global non-convex optimization. As a brief summary of that work, this appendix includes the documentation on the three main classes used in that code.

```
Help on module ensemble:


NAME
    ensemble


FILE
    /Users/josh/Dropbox/REU stuff/EoF/EoFlib/EoF/ensemble.py


CLASSES
    Ensemble


    class Ensemble
     |  Represents a quantum ensemble.
     |
     |  Methods defined here:
     |
     |  EoF_pure(self, rho)
     |      The entanglement of formation of a pure state is equal to the von
     |      Neumann entropy of either of the two subsystems.
     |
     |      rho: a density matrix representing a pure state (of the whole system)
     |
     |  __init__(self, dist, inputmode='ens', partition_dimension=2)
     |      dist: the distribution of the ensemble. depending on inputmode,
     |      this is:
     |
     |          inputmode='ens': (probabilities, states)
     |              probabilities: an array of probabilities
     |              states: an array of 1D vectors representing quantum states
     |
     |          inputmode='ensdm': (probabilities, dms)
     |              probabilities: an array of probabilities
     |              dms: an array of density matrices (as numpy arrays)
     |
     |          inputmode='npdm': dm (numpy.array)
     |              dm: the combined density matrix of the whole ensemble.
     |              ensemble is taken to be eigen-ensemble
     |
     |          inputmode='qtdm': dm (QObject)
     |              dm: a Qutip density matrix (NOT YET IMPLEMENTED)
     |
     |      partition_dimension: the dimension of the first subsystem for
     |      use in EoF calculation
     |
     |  ave_EoF(self)
     |      This is simply the average of the EoF values for each pure rho.
     |
     |      Given some decomposition of arbitrary state rhoprime = sum_i
     |      ps_i rhos_i in terms of pure states rhos_i, we can compute
     |      this average EoF.
     |
     |      Given that the EoF is defined as the minimum over all such
     |      decompositions, this provides an upper bound.
     |
     |  partial_trace(self, rho)
     |      Returns $rho^A =  r_B(rho)$, where A is the subsystem dictated
     |      by self.partition_dimension. This one is what Wei Cai used. Checks
     |      against qt.ptrace([1])
     |
     |      rho: array ($rho^{AB}$)
     |
     |  qt_dm(self)
     |
     |  to_U_ensemble(self, U)
     |      Changes decomposition to one given by the process described in Ryu.
     |
     |      U : first N columns of a unitary matrix of dimension M >= N,
     |      (<= N^2)
     |
     |  ----------------------------------------------------------------------
     |  Static methods defined here:
     |
     |  von_neumann(rho)
     |      Returns the von Neumann entropy of rho
```

```
Help on module eof_optimizer:

NAME
    eof_optimizer

FILE
    /Users/josh/Dropbox/REU stuff/EoF/EoFlib/EoF/eof_optimizer.py

CLASSES
    EoF

    class EoF
     |  An optimizer to find the entanglement of formation of an ensemble, based on
     |
     |  Ryu, S., Cai, W., & Caro, A. (2008).
     |  Quantum entanglement of formation between qudits.
     |  Physical Review A, 77(5), 052312.
     |
     |  Default behaviour is to use their 'hybrid' method of conjugate
     |  gradient + steepest descent, with basinhopping for Hilbert spaces
     |  of dimension > 4.
     |
     |  Methods defined here:
     |
     |  G(self)
     |      the gradient G evaluated at the current self.Utilde
     |
     |  Utilde2vec(self, Utilde)
     |      Utilde: an MxM unitary matrix
     |
     |      returns: a 2*M^2-length vector for use in scipy's optimize
     |      functions
     |
     |  __init__(self, ensemble)
     |      ensemble: an Ensemble object
     |
     |  g(self)
     |      Returns the steepest descent direction for H evaluated at the
     |      current self.Utilde
     |
     |  local_optimize_sp_wrap(self, fun, x0, args, mode='hybrid',
     |                         step_size_start=1.0, tol=1e-08, iter_limit=1000, **options)
     |      Wraps the local optimization functions so they can be plugged into
     |      basinhopping
     |
     |  optimize(self, seed, mode='hybrid', tol=1e-08, step_size_start=1.0,
     |           iter_limit=1000, nhops=10, basinhopping_callback=None)
     |      seed: an MxM unitary matrix, where M is the number of states in the
     |      ensemble
     |
     |      mode: 'hybrid,' 'conjugate_gradient,' or 'steepest_descent.'
     |
     |      tol: the tolerance for convergence
     |
     |      step_size_start: the initial step size for steepest descent/hybrid
     |
     |      iter_limit: the maximum number of iterations allowed
     |
     |      nhops: how many times basinhopping should run (for N>4)
     |
     |      basinhopping_callback: a callable to be provided to scipy's
     |      basinhopping as a callback
     |
     |  vec2Utilde(self, vec)
     |      vec: a 2*M^2-length vector for use in scipy's optimize functions
     |
     |      returns: an MxM complex matrix constructed from vec
     |
     |  ----------------------------------------------------------------------
```

```
     |  Static methods defined here:
     |  
     |  min_of_parabola(x, y)

DATA
    DEFAULT_BASINHOPPING_HOPS = 10
    DEFAULT_ITER_LIMIT = 1000
    DEFAULT_STEP_START = 1.0
    DEFAULT_TOL = 1e-08
```

```
Help on module ai_optimizer:

NAME
    ai_optimizer

FILE
    /Users/josh/Dropbox/REU stuff/EoF/EoFlib/EoF/ai_optimizer.py

CLASSES
    AccessibleInfo
    
    class AccessibleInfo
     |  An optimizer to find the accessible information of an ensemble.
     |  
     |  Based on
     |  Rehacek, J.; Englert, B.; Kaszlikowski, D.
     |  Iterative procedure for computing accessible information
     |  in quantum communication (2005)
     |  Phys. Rev. A
     |  http://dx.doi.org/10.1103/PhysRevA.71.054303
     |  
     |  Methods defined here:
     |  
     |  __init__(self, ensemble)
     |      ensemble: an Ensemble object
     |  
     |  optimize_steepest_descent(self, seed, tol=1e-08, step_size_start=0.1, iter_limit=1000)
     |      seed:
     |          a povm, represented by a K-dimensional list of NxN arrays
     |      
     |      tol:
     |          the tolerance of the convergence criterion
     |      
     |      step_size_start:
     |          should be < 1
     |      
     |      iter_limit:
     |          the limit on the number of iterations of the optimizer
```

```
DATA
    DEFAULT_BASINHOPPING_HOPS = 10
    DEFAULT_ITER_LIMIT = 1000
    DEFAULT_STEP_START = 0.1
    DEFAULT_TOL = 1e-08
    stdout = <open file '<stdout>', mode 'w'>
```