PURM 2020: Generating Time Series Data

8/12/20 Project Overview (revised 8/19/20)

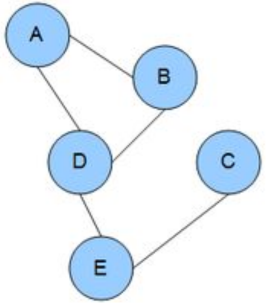
Alan Ismaiel and Jason Shu

Motivation

There exist increasingly many tools to generate large amounts of time series data, which is significantly affordable than collecting the same amount of data.

- However, many of the tools we surveyed use deterministic ways to specify time series
- Other tools used an ARMA approach, which is more random but doesn't intuitively represent relationships between random variables

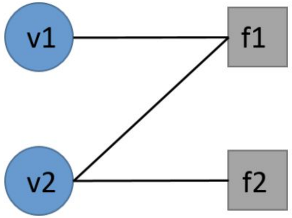
Existing Approaches



Bayesian Graphs/Markov Networks/Mixed Graphs

- Captures similar information as our time series system, but is limited by independence/conditional independence assumptions

Factor Graphs/Machine Learning Models



- Uses existing datasets to generate an approximation of the time series behavior: imprecise and requires data that may not exist

Autoregressive Models/Moving Average Models

- Calculates time series systems as a function in terms of all previous values in the time series, as well as approximations for white noise and error, concepts not considered in our implementation

Existing Tools

- Faker
  - Generates data based on user-defined/pre-existing data types, but has no notion of time or dependence
- DataGenerator
  - Capable of modeling complex probability distributions, but runs on an outdated XML system and has no built-in notion of time series
- Pandas (Time Series Functionality)
  - Good way to organize/analyze time series data, no inherent ability to specify dependencies between variables
- T Simulus
  - Time series data points are hard-coded into JSON objects, or ARMA model is used
  - Simple time series can be composited to make more complex ones
  - Would work better as a platform to plug into *after* determining distribution
- Pyro
  - Claims to be able to represent any computable distribution, built-in generation abilities
  - Complicated input system without inherent representations of independence/dependence

Penn Engineering

PRECISE
PENN RESEARCH IN EMBEDDED COMPUTING AND INTEGRATED SYSTEMS ENGINEERING

Our Problem Formulation

To make as expressive a tool as possible, our goal is to take user-inputted probability/independence statements and fully determine distributions using them through a distinct parameterization method.

- To best represent stochastic processes, we aim to encode dependency between time steps (e.g. past results influence future results)
- By utilizing independence specifications, the complexity of defining probability systems can be **exponentially reduced**

Problem Statement

Our considerations of time series can be distinguished into three separate cases:

- **Static Time Series Case**
- **Time Invariant Time Series Case**
- **Time Variant Time Series Case**

Outside of the static case, we define a notion of **relevant time steps**. Put simply, variables defined at a time t are exclusively dependent on a set number of the same variables defined at previous time steps. These previous time step variables defined in a set B_t are the relevant time steps

Problem Statement: Static Case

Static Case

- The notion of time is **not** considered
- **Input:**
  - A set of categorical variables and each variable's values
  - A set of specifications which describe the distribution over the variables
- **Goal:** To find the joint distribution over all variables, or to find whether the specifications are underdetermined or contradictory

Case Examples

- The probability system of drawing a card from a deck 100 times

Problem Statement: Time Invariant Case

Time Invariant Case

- At any time t, the corresponding probability system S_t is under the stationary assumption
- **Stationary Assumption**: Probability distributions don't change when shifted in time

Case Examples

- The probability system of deciding what to wear based only on what was worn the previous day

Problem Statement: Time Variant Case

Time Variant Case

- For all variables generated in a unit of time t, their probabilities are a function of the relevant time steps probability distributions
- Considers the distribution of past time steps to calculate the current one

Case Examples

- The probability system of robot components that have a higher likelihood of wearing down the more time they are functional
- The probability system representing sunny and rainy days that change probabilities based on how many prior days

Solution Approach

Our approach to constructing a solution to this problem takes on the following steps:

1. **Parameterizing** the user input and specifications in an understandable way
2. **Solving** the probability system in terms of the parameterization
3. **Generating** data in the event that the system is fully defined

Detailed Solution

Solutions for the three cases differ, but they generally follow these steps:

1. **Import** user-provided text file containing information about data generation (including variables, variable values, specifications, etc.) into Mathematica
2. **Parse** text file and translate information into standard Mathematica language
3. **Convert** probability symbols (e.g. P[A | B], P[C && D]) to combinations of elementary probabilities, which are denoted by “O parameters”
4. **Solve** the equations for the O parameters. Note whether the system of equations is well-determined, contradictory, or under-determined. If well-determined, map the O values to a distribution
5. **Generate** data from said distribution
6. In a **non-static** case, **repeat step 5** for as many steps as the user would like to generate.

Time Invariant Example

Consider a single boolean variable T, defined with 2 relevant time steps

$P(T_t \cap T_{t-1} \cap T_{t-2})$	o000
$P(T_t \cap T_{t-1} \cap \neg T_{t-2})$	o001
$P(T_t \cap \neg T_{t-1} \cap T_{t-2})$	o010
$P(T_t \cap \neg T_{t-1} \cap \neg T_{t-2})$	o011
$P(\neg T_t \cap T_{t-1} \cap T_{t-2})$	o100
$P(\neg T_t \cap T_{t-1} \cap \neg T_{t-2})$	o101
$P(\neg T_t \cap \neg T_{t-1} \cap T_{t-2})$	o110
$P(\neg T_t \cap \neg T_{t-1} \cap \neg T_{t-2})$	o111

Time Invariant Example

We can assume the following information:

- 1 = o000 + … + o111

Stationary Assumption information:

- P(T_t) = P(T_t-1) = P(T_t-2)
- P(T_t AND T_t-1) = P(T_t-1 AND T_t-2)
- P(T_t AND not T_t-1) = P(T_t-1 AND not T_t-2)
- P(not T_t AND T_t-1) = P(not T_t-1 AND T_t-2)

It can be proven that by defining P(T_t AND T_t-1) = P(T_t-1 AND T_t-2), P(T_t AND not T_t-1) = P(T_t-1 AND not T_t-2), and P(not T_t AND T_t-1) = P(not T_t-1 AND T_t-2), then all the necessary equalities between time steps required for the stationary assumption will hold true. Thus, 4 more equations needed.

Time Invariant Example

Add the following information:

- P(T_t | T_t-1 and T_t-2) = .2
- P(T_t | not T_t-1 and not T_t-2) = .8
- P(T_t | T_t-1 and not T_t-2) = .5
- P(T_t | not T_t-1 and not T_t-2) = .5

Mathematica Output: {{o000 -> {0.0384615}, o001 -> {0.153846}, o010 -> {0.153846}, o011 -> {0.153846}, o100 -> {0.153846}, o101 -> {0.153846}, o110 -> {0.153846}, o111 -> {0.0384615}}}

Notes:

- The amount of relations that must hold for the stationary assumption increases exponentially as the number of relevant time steps increases
- We CANNOT conclude that only a single time step is relevant: the conditional expressions clearly show that both their results could impact the outcome
  - As such, the distribution of these variables currently break rules that would be required to be considered a Markov Chain

Generating Time Invariant Data

From last slide:

{{o000 -> {0.0384615}, o001 -> {0.153846}, o010 -> {0.153846}, o011 -> {0.153846}, o100 -> {0.153846}, o101 -> {0.153846}, o110 -> {0.153846}, o111 -> {0.0384615}}}

Calculate conditional probabilities for current step given past two:

1. P(T_t | T_t-1 && T_t-2) = o000 / (o000 + o100) = .2
2. P(T_t | T_t-1 && not T_t-2) = o001 / (o001 + o101) = .5
3. P(T_t | not T_t-1 && T_t-2) = o010 / (o010 + o110) = .5
4. P(T_t | not T_t-1 && not T_t-2) = o011 / (o011 + o111) = .8

Suppose T_1 was False and T_2 was True. Generate T_3.

From Equation 3 above, P(T_3 | not T_2 && T_1) = .5; generate T_3 using a random number generator.

Suppose T_3 was True. Now generate T_4 using Equation 1, as T_2 and T_3 were True. Repeat generation.

Solution: Time Variant Case

While in the time-invariant case, the stationary assumption guaranteed that distributions remain identical from one time step to another, there is no such assumption for the time-variant case.

- The time-variant case allows probabilities from **past time steps** to influence probabilities in following steps
- Since distributions can change from one step to another, probability quantities from past time steps are represented **symbolically** by the user, and the equations are also solved symbolically
- The literal values of those past probabilities are **substituted** into the symbolic solution before generating data for each step

Time Variant Example

Consider a single boolean variable T, defined with 2 relevant time steps (same as before)

$P(T_t \cap T_{t-1} \cap T_{t-2})$	o000
$P(T_t \cap T_{t-1} \cap \neg T_{t-2})$	o001
$P(T_t \cap \neg T_{t-1} \cap T_{t-2})$	o010
$P(T_t \cap \neg T_{t-1} \cap \neg T_{t-2})$	o011
$P(\neg T_t \cap T_{t-1} \cap T_{t-2})$	o100
$P(\neg T_t \cap T_{t-1} \cap \neg T_{t-2})$	o101
$P(\neg T_t \cap \neg T_{t-1} \cap T_{t-2})$	o110
$P(\neg T_t \cap \neg T_{t-1} \cap \neg T_{t-2})$	o111

Time Variant Example

Additionally, consider the existence of 4 q parameters

$P(T_{t-1} \cap T_{t-2})$	q00
$P(T_{t-1} \cap \neg T_{t-2})$	q01
$P(\neg T_{t-1} \cap T_{t-2})$	q10
$P(\neg T_{t-1} \cap \neg T_{t-2})$	q11

These are defined similarly to the o parameters, albeit without any variables at time step t. We can assume that when generating data for any time t, we will already have the values of these 4 q parameters.
NOTE: Due to the given equation, we can put one q value in terms of the other 3. Therefore, only 3 of these values are necessary to define.

Time Variant Example

Consider 4 different specifications (Mathematica input format in blue):

- P(T_t) = .5*P(T_t-1) + .5*P(T_t-2)
  - o000 + o001 + o010 + o011 == .5 * (**q00** + **q01**) + .5 * (**q00** + **q10**)
- P(T_t and T_t-1) = .9 * P(T_t-1 and T_t-2)
  - o000 + o001 == .9 * **q00**
- P(T_t and T_t-2) = .3
  - o000 + o010 == .3
- P(T_t | T_t-1 AND T_t-2) = .2
  - o000 / (o000 + o001) == .2

This is in addition to the 4 specifications initially given (defining 3 of the **q parameters** in terms of o parameters, and the common given equation)

Time Variant Example

Output:

```
Out[26]= {{o000 → {1. - 0.1 (3. - 2. q00) - 0.8 q00 - 0.1 (-7. q00 + 10. q01) - 0.1 (13. - 20. q00 - 5. q01 - 1. (-7. q00 + 10. q01) - 15. q10) -
    1. (0.1 (-3. + 2. q00) + q10) - 1. (1. - 1. q00 + 0.1 (-3. + 2. q00) - 1. q01 - 0.1 (13. - 20. q00 - 5. q01 - 1. (-7. q00 + 10. q01) - 15. q10) - 1. (0.1 (-3. + 2. q00) + q10)) -
    1. (1. - 0.1 (3. - 2. q00) - 1. q00 - 0.1 (-7. q00 + 10. q01) - 0.1 (13. - 20. q00 - 5. q01 - 1. (-7. q00 + 10. q01) - 15. q10) - 1. (0.1 (-3. + 2. q00) + q10) -
      1. (1. - 1. q00 + 0.1 (-3. + 2. q00) - 1. q01 - 0.1 (13. - 20. q00 - 5. q01 - 1. (-7. q00 + 10. q01) - 15. q10) - 1. (0.1 (-3. + 2. q00) + q10)))} if condition,
  o001 → {1. - 0.1 (3. - 2. q00) - 1. q00 - 0.1 (-7. q00 + 10. q01) - 0.1 (13. - 20. q00 - 5. q01 - 1. (-7. q00 + 10. q01) - 15. q10) - 1. (0.1 (-3. + 2. q00) + q10) -
    1. (1. - 1. q00 + 0.1 (-3. + 2. q00) - 1. q01 - 0.1 (13. - 20. q00 - 5. q01 - 1. (-7. q00 + 10. q01) - 15. q10) - 1. (0.1 (-3. + 2. q00) + q10))} if condition,
  o010 → {0.1 (3. - 2. q00)} if condition, o011 → {1. - 1. q00 + 0.1 (-3. + 2. q00) - 1. q01 - 0.1 (13. - 20. q00 - 5. q01 - 1. (-7. q00 + 10. q01) - 15. q10) - 1. (0.1 (-3. + 2. q00) + q10)} if condition,
  o100 → {0.8 q00} if condition, o101 → {0.1 (-7. q00 + 10. q01)} if condition, o110 → {0.1 (-3. + 2. q00) + q10} if condition,
  o111 → {0.1 (13. - 20. q00 - 5. q01 - 1. (-7. q00 + 10. q01) - 15. q10)} if condition}}
```

Q Conditions (system is conflicting if Qs fall outside these constraints):

```
(q00 > 0 && 0 < q01 < 0.190909 && q00 - 1.42857 q01 < 0 && 0.6 - 0.6 q00 - 1. q01 - q10 < 0 && -0.866667 + 0.866667 q00 + 1. q01 + q10 < 0) ||
  (q00 > 0 && 0.190909 < q01 < 0.290244 && 0.6 - 0.6 q00 - 1. q01 - q10 < 0 &&
    -0.866667 + 0.866667 q00 + 1. q01 + q10 < 0 && -0.75 + q00 + 2.5 q01 < 0) || (q00 > 0 && 0.290244 < q01 < 0.3 &&
    0.6 - 0.6 q00 - 1. q01 - q10 < 0 && -0.866667 + 0.866667 q00 + 1. q01 + q10 < 0 && -0.75 + q00 + 2.5 q01 < 0) ||
  (q00 > 0 && 0.3 < q01 < 0.566667 && -0.866667 + 0.866667 q00 + 1. q01 + q10 < 0 && 0.3 - 0.2 q00 - q10 < 0 && -0.85 + q00 + 1.5 q01 < 0) ||
  (0.190909 < q01 < 0.290244 && q00 - 1.42857 q01 < 0 && -0.866667 + 0.866667 q00 + 1. q01 + q10 < 0 &&
    0.3 - 0.2 q00 - q10 < 0 && 0.75 - q00 - 2.5 q01 < 0) || (0.290244 < q01 < 0.3 && -0.866667 + 0.866667 q00 + 1. q01 + q10 < 0 &&
    0.3 - 0.2 q00 - q10 < 0 && -0.85 + q00 + 1.5 q01 < 0 && 0.75 - q00 - 2.5 q01 < 0)
```

Generating Time Variant Data

STEP 1 (BASE CASE):

- q00 == .3
- q01 == .25
- q10 == .25

{{o000 → {0.06}, o001 → {0.21}, o010 → {0.24},
o011 → {0.04}, o100 → {0.24}, o101 → {0.04}, o110 → {0.01}, o111 → {0.16}}}

From here, deduce the next q values as follows:

- q00 == o000 + o001 == .27
- q01 == o010 + o011 == .28
- q10 == o100 + o101 == .28

{{o000 → {0.054}, o001 → {0.189}, o010 → {0.246}, o011 → {0.061},
o100 → {0.216}, o101 → {0.091}, o110 → {0.034}, o111 → {0.109}}}

STEP 2:

- Replace the previous q values with the ones deduced from step 1

From here, deduce the next q values as follows:

- q00 == o000 + o001 == .243
- q01 == o010 + o011 == .307
- q10 == o100 + o101 == .307

{{o000 → {0.0486}, o001 → {0.1701}, o010 → {0.2514}, o011 → {0.0799},
o100 → {0.1944}, o101 → {0.1369}, o110 → {0.0556}, o111 → {0.0631}}}

STEP 3:

- Replace the previous q values with the ones deduced from step 2

And so on…

Program Demonstration

To program these cases to generate, **Mathematica** is our program of choice

- Possesses exceptional capabilities in solving and notating complex systems of equations
- Extensive and accessible documentation for its various features
- Built in discrete probability distributions and random generation capabilities

In the two weeks since we have begun programming with Mathematica, we have created a foundation to build our time series systems off of, and a working implementation of the static case

Penn Engineering

PRECISE

PENN RESEARCH IN EMBEDDED COMPUTING AND INTEGRATED SYSTEMS ENGINEERING

Limitations

There are some limitations to consider in our solution approach

Complex

- Even when utilizing independence statements, these systems grow at an exponential rate, and fully providing functioning input specifications for more complicated systems becomes increasingly difficult.

Costly

- Though we have yet to reach a limitation of Mathematica's high-end solver, we can't confidently claim that one won't exist.

Theoretical

- The more complicated the probability systems get, the more likely users may be inclined to approximate values they don't know, leading to error in the generation

References

Castanon, D. and Karl, W.C., 2004, *Stochastic Processes*, Lecture Notes, Class Notes, SC505, Boston University, http://www.mit.edu/people/hmsallum/GradSchool/sc505notes.pdf

DeepDive Project at Stanford University, “Factor Graphs”, n.d., http://deepdive.stanford.edu/assets/factor_graph.pdf

Gillespie, Daniel T., “Exact stochastic simulation of coupled chemical reactions”, 1977, https://pubs.acs.org/doi/10.1021/j100540a008

Henderson, T.C. et al., “Probabilistic Logic for Intelligent Systems”, 2018, http://www.cs.utah.edu/~tch/publications/pub302.pdf

Hu, Z. and Hong, L.J., “Robust Simulation Of Stochastic Systems With Input Uncertainties Modeled By Statistical Divergences”, 2015, https://www.informs-sim.org/wsc15papers/055.pdf

Kang, Yanfei et al., “GRATIS: Generating Time Series with Diverse and Controllable Characters”, 2019, https://arxiv.org/ftp/arxiv/papers/1903/1903.02787.pdf

Pandey, A. et al., May 2017, “Towards a Formal Framework for Hybrid Planning in Self-Adaptation”, Presentation, Carnegie Mellon University, https://www.slideshare.net/ivanruchkin/towards-a-formal-framework-for-hybrid-planning-in-selfadaptation