A Probabilistic Programming Approach To Probabilistic Data Analysis

Authored by:

Vikash K. Mansinghka
Feras Saad

Abstract

Probabilistic techniques are central to data analysis, but different approaches can be challenging to apply, combine, and compare. This paper introduces composable generative population models (CGPMs), a computational abstraction that extends directed graphical models and can be used to describe and compose a broad class of probabilistic data analysis techniques. Examples include discriminative machine learning, hierarchical Bayesian models, multivariate kernel methods, clustering algorithms, and arbitrary probabilistic programs. We demonstrate the integration of CGPMs into BayesDB, a probabilistic programming platform that can express data analysis tasks using a modeling definition language and structured query language. The practical value is illustrated in two ways. First, the paper describes an analysis on a database of Earth satellites, which identifies records that probably violate Kepler?s Third Law by composing causal probabilistic programs with non-parametric Bayes in 50 lines of probabilistic code. Second, it reports the lines of code and accuracy of CGPMs compared with baseline solutions from standard machine learning libraries.

1 Paper Body

Probabilistic techniques are central to data analysis, but can be difficult to apply, combine, and compare. Such difficulties arise because families of approaches such as parametric statistical modeling, machine learning and probabilistic programming are each associated with different formalisms and assumptions. The contributions of this paper are (i) a way to address these challenges by defining CGPMs, a new family of composable probabilistic models; (ii) an integration of this family into BayesDB [10], a probabilistic programming platform for data analysis; and (iii) empirical illustrations of the efficacy of the framework for analyzing a real-world database of Earth satellites. We introduce composable generative population models (CGPMs), a computational formalism that generalizes directed graphical models. CGPMs specify a table of observable random variables with a finite number of columns and countably infinitely many rows. They support complex intra-row dependencies among the observables, as well as inter-row dependencies among a field of latent random variables. CGPMs are described by a computational interface for generating samples and evaluating densities for random variables derived from the base table by conditioning and marginalization. This paper shows how to package discriminative statistical learning techniques, dimensionality reduction methods, arbitrary probabilistic programs, and their combinations, as CGPMs. We also describe algorithms and illustrate new syntaxes in the probabilistic Metamodeling Language for building composite CGPMs that can interoperate with BayesDB. The practical value is illustrated in two ways. First, we describe a 50-line analysis that identifies satellite data records that probably violate their theoretical orbital characteristics. The BayesDB script builds models that combine non-parametric Bayesian structure learning with a causal probabilistic program that implements a stochastic variant of Kepler?s Third Law. Second, we illustrate coverage and conciseness of the CGPM abstraction by quantifying the improvement in accuracy and reduction in lines of code achieved on a representative data analysis task. 30th Conference on Neural Information Processing Systems (NIPS 2016), Barcelona, Spain.

2

Composable Generative Population Models

A composable generative population model represents a data generating process for an exchangeable sequence of random vectors (x1 , x2 , . . .), called a population. Each member xr is T -dimensional, and element x[r,t] takes values in an observation space Xt , for t ? [T] and r ? N. A CGPM G is formally represented by a collection of variables that characterize the data generating process: G = (?, ?, Z = {zr : r ? N}, X = {xr : r ? N}, Y = {yr : r ? N}). ? ?: Known, fixed quantities about the population, such as metadata and hyperparameters. ? ?: Population-level latent variables relevant to all members of the population. ? zr = (z[r,1] , . . . z[r,L]): Member-specific latent variables that govern only member r directly. ? xr = (x[r,1] , . . . x[r,T]): Observable output variables for member r. A subset of these variables may be observed and recorded in a dataset D. ? yr = (y[r,1] , . . . y[r,I]): Input variables, such as ?feature vectors? in a purely discriminative model. A CGPM is required to satisfy the following conditional independence constraint: ?r 6= r0 ? N, ?t, t0 ? [T] : x[r,t] ?? x[r0 ,t0] — {?, ?, zr , zr0 }. (1) Eq (1) formalizes the notion that all dependencies across members r ? N are completely mediated by the population parameters ? and member-specific variables zr . However, elements x[r,i] and x[r,j] within a member are generally free to assume any dependence structure. Similarly, the memberspecific latents in Z may be either uncoupled or highly-coupled given population parameters ?. CGPMs differ from the standard mathematical definition of a joint density in that they are defined in terms of a computational interface (Listing 1). As computational objects, they explicitly distinguish between the sampler for the random variables from their joint distribution, and the assessor of their joint density. In particular, a CGPM is required to sample/assess the joint distribution of a subset of output variables x[r,Q] conditioned on another subset x[r,E] , and marginalizing over x[r,[T](Q?E)] . Listing 1 Computational interface for composable generative population models. ? s ? simulate (G, member: r, query: Q = {qk }, evidence : x[r,E] , input : yr) Generate a sample from the distribution s ?G x[r,Q] —{x[r,E] , yr , D}. ? c ? logpdf (G, member: r, query : x[r,Q] , evidence : x[r,E] , input : yr) Evaluate the log density log pG (x[r,Q] —{x[r,E] , yr , D}). ? G 0 ? incorporate (G, measurement : x[r,t] or yr) Record a measurement x[r,t] ? Xt (or yr) into the dataset D. ? G 0 ? unincorporate (G, member : r) Eliminate all measurements of input and output variables for member r. ? G 0 ? infer (G, program : T) Adjust internal latent state in accordance with the learning procedure specified by program T .

2.1

Primitive univariate CGPMs and their statistical data types

The statistical data type (Figure 1) of a population variable xt generated by a CGPM provides a more refined taxonomy than its ?observation space? Xt . The (parameterized) support of a statistical type is the set in which samples from simulate take values. Each statistical type is also associated with a base measure which ensures logpdf is well-defined. In high-dimensional populations with heterogeneous types, logpdf is taken against the product measure of these base measures. The statistical type also identifies invariants that the variable maintains. For instance, the values of a NOMINAL variable are permutation-invariant. Figure 1 shows statistical data types provided by the Metamodeling Language from BayesDB. The final column shows some examples of primitive CGPMs that are compatible with each statistical type; they implement logpdf directly using univariate probability density functions, and algorithms for simulate are well known [4]. For infer their parameters may be fixed, or learned from data using, e.g., maximum likelihood [2, Chapter 7] or Bayesian priors [5]. We refer to an extended version of this paper [14, Section 3] for using these primitives to implement CGPMs for a broad collection of model classes, including non-parametric Bayes, nearest neighbors, PCA, discriminative machine learning, and multivariate kernel methods. 2

Statistical Data Type BINARY NOMINAL COUNT/RATE CYCLIC MAGNITUDE NUMERICAL NUMERICAL-RANGED Nominal

Parameters symbols: S base: b period: p ? ? low: l, high:h Count

Measure/?-Algebra

{0, 1} {0 . . . S?1} {0, 1b , 2b , . . .} (0, p) (0, ?) (??, ?) (l, h) ? R Magnitude

Poisson Geometric

{0,1}

(#, 2) (#, 2[S]) (#, 2N) (?, B(R)) (?, B(R)) (?, B(R)) (?, B(R))

BERNOULLI CATEGORICAL POISSON, GEOMETRIC VON-MISES LOG-NORMAL, EXPON NORMAL BETA, NORMAL-TRUNC

Cyclic

Numerical

Von-Mises

Lognormal Exponential

Primitive CGPM
Numerical-Ranged
Normal
NormalTrunc Beta
Frequency
Categorical
Support
Figure 1: Statistical data types for population variables generated by CGPMs available in the BayesDB Metamodeling Language, and samples from their marginal distributions. 2.2

Implementing general CGPMs as probabilistic programs in VentureScript

In this section, we show how to implement simulate and logpdf (Listing 1) for composable generative models written in VentureScript [8], a probabilistic programming language with programmable inference. For simplicity, this section assumes a stronger conditional independence constraint, ?l, l0 ? [L] such that (r, t) 6= (r0 , t0) =? x[r,t] ?? x[r0 ,t0] — {?, ?, z[r,l] , z[r0 ,l0] , yr , yr0 }.

(2)

In words, for every observable element x[r,t] , there exists a latent variable z[r,l] which (in addition to ?) mediates all coupling with other variables in the population. The member latents Z may still exhibit arbitrary dependencies. The approach for simulate and logpdf described below is based on approximate inference in tagged subparts of the Venture trace, which carries a full realization of all random choices (population and member-specific latent variables) made by the program. The runtime system carries a set of K traces {(? k , Z k)}K k=1 sampled from an approximate posterior pG (?, Z—D). These traces are assigned weights depending on the user-specified evidence x[r,E] in the simulate/logpdf function call. G represents the CGPM as a probabilistic program, and the input yr and latent variables Z k are treated as ambient quantities in ? k . The distribution of interest is Z pG (x[r,Q] —x[r,E] , D) = pG (x[r,Q] —x[r,E] , ?, D)pG (?—x[r,E] , D)d? ?

Z pG (x[r,E] —?, D)pG (?—D) = pG (x[r,Q] —x[r,E] , ?, D) d? (3) pG (x[r,E] —D) ? ? PK

K X

1

k k=1 w

pG (x[r,Q] —x[r,E] , ? k , D)wk

where ? k ?G —D. (4)

k=1

The weight wk = pG (x[r,E] —? k , D) of trace ? k is the likelihood of the evidence. The weighting scheme (4) is a computational trade-off avoiding the requirement to run posterior inference on population parameters ? for a query about member r. It suffices to derive the distribution for only ? k , Z pG (x[r,Q] —x[r,E] , ? k , D) = pG (x[r,Q] , zrk —x[r,E] , ? k , D)dzrk (5) zrk

Z =

Y

zrk q?Q

J

1X Y pG (x[r,q] —zrk , ? k) pG (zrk —x[r,E] , ? k , D)dzrk ? pG (x[r,q] —zrk,j , ? k), J j=1

(6)

q?Q

where zrk,j ?G —{x[r,E] , ? k , D}. Eq (5) suggests that simulate can be implemented by sampling (x[r,Q] , zrk) ?G —{x[r,E] , ? k , D} from the joint local posterior, then returning elements x[r,Q] . Eq (6) shows that logpdf can be implemented by first sampling the member latents zrk ?G —{x[r,E] , ? k , D} from the local posterior; using the conditional independence constraint (2), the query x[r,Q] then factors into a product of density terms for each element x[r,q] . 3

To aggregate over {? k }K k=1 , for simulate the runtime obtains the queried sample by first drawing k ? C ATEGORICAL({w1 , . . . , wK }), then returns the sample x[r,Q] drawn from trace ? k . Similarly, logpdf is computed using the weighted Monte Carlo estimator (6). Algorithms 2a and 2b summarize implementations of simulate and logpdf in a general probabilistic programming environment. Algorithm 2a simulate for CGPMs in a probabilistic programming environment. 1: function S IMULATE(G, r, Q, x[r,E] , yr) 2: for k = 1, . . . , K do 3: if zrk ? 6 Z k then k 4: zr ?G —{? k , Z k , D} Q k 5: w ? e?E pG (x[r,e] —? k , zrk) 6: k ? C ATEGORICAL ({w1 , . . . , wk }) 7: {x[r,Q] , zrk } ?G —{? k , Z k , D ? {yr , x[r,E] }} 8: return x[r,Q]

. for each trace k . if member r has unknown local latents . sample them from the prior . weight the trace by likelihood of evidence . importance resample the traces . run a transition operator leaving target invariant . select query variables from the resampled trace

Algorithm 2b logpdf for CGPMs in a probabilistic programming environment. 1: function L OG P DF(G, r, x[r,Q] , x[r,E] , yr) 2: for k = 1, . . . , K do 3: Run steps 2 through 5 from Algorithm 2a 4: for j = 1, . . . , J do 5: zrk,j ?G —{? k , Z k , D ? {yr , x[r,E] }} Q 6: hk,j ? q?Q pG (x[r,q] —? k , zrk,j) P 7: rk ? J1 Jj=1 hk,j k k 8: q k ? r w

P

PK K k k 9: return log ? log k=1 q k=1 w

2.3

. for each trace k . retrieve the trace weight . obtain J samples of latents in scope of member r . run a transition operator leaving target invariant . compute the density estimate . aggregate density estimates by simple Monte Carlo . importance weight the estimate . weighted importance sampling over all traces

Inference in a composite network of CGPMs

This section shows how CGPMs are composed by applying the output of one to the input of another. This allows us to build complex probabilistic models out of simpler primitives directly as software. Section 3 demonstrates surface-level syntaxes in the Metamodeling Language for constructing these composite structures. We report experiments including up to three layers of composed

CGPMs. Let G a be a CGPM with output xa? and input y?a , and G b have output xb? and input y?b (the symbol ? a indexes all members r ? N). The composition GBb ? GA applies the subset of outputs xa[?,A] of G a to b the inputs y[?,B] of G b , where —A— = —B— and the variables are type-matched (Figure 1). This operation b results in a new CGPM G c with output xa? ? xb? and input y?a ? y[?,B] . In general, a collection k {G : k ? [K]} of CGPMs can be organized into a generalized directed graph G [K] , which itself is a CGPM. Node k is an ?internal? CGPM G k , and the labeled edge aA ? bB denotes the composition a GA ? GBb . The directed acyclic edge structure applies only to edges between elements of different CGPMs in the network; elements xk[?,i] , xk[?,j] within G k may satisfy the more general constraint (1). Algorithms 3a and 3b show sampling-importance-resampling and ratio-likelihood weighting algorithms that combine simulate and logpdf from each individual G k to compute queries against network G [K] . The symbol ? k = {(p, t) : xp[?,t] ? y?k } refers to the set of all output elements from upstream CGPMs connected to the inputs of G k , so that {? k : k ? [K]} encodes the graph adjacency matrix. Subroutine 3c generates a full realization of all unconstrained variables, and weights forward samples from the network by the likelihood of constraints. Algorithm 3b is based on ratio-likelihood weighting (both terms in line 6 are computed by unnormalized importance sampling) and admits an analysis with known error bounds when logpdf and simulate of each G k are exact [7]. Algorithm 3a simulate in a directed acyclic network of CGPMs. 1: function S IMULATE(G k , r, Qk , xk[r,E k] , yrk , for k ? [K]) 2: for j = 1, . . . , J do 3: (sj , wj) ? W EIGHTED -S AMPLE ({xk[r,E k] : k ? [K]}) 4: 5:

m ? C ATEGORICAL ({w1 , . . . , wJ }) return {xk[r,Qk] ? sm : k ? [K]}

. generate J importance samples . retrieve jth weighted sample

. resample by importance weights . return query variables from the selected sample

4

Algorithm 3b logpdf in a directed acyclic network of CGPMs. 1: function S IMULATE(G k , r, xkQ , xk[r,E k] , yrk , for k ? [K]) 2: for j = 1, . . . , J do 3: (sj , wj) ? W EIGHTED -S AMPLE ({xk[r,Qk ?E k] : k ? [K]}) 4: 5: 6:

. generate J importance samples . joint density of query/evidence

for j = 1, . . . , J 0 do . generate J 0 importance samples (s0j , w0j) ? W EIGHTED -S AMPLE ({xk[r,E k] : k ? [K]}) . marginal density of evidence P

j P 0j 0 return log ? log(J/J) . return likelihood ratio importance estimate [J] w / [J 0] w

Algorithm 3c Weighted forward sampling in a directed acyclic network of CGPMs. 1: function W EIGHTED -S AMPLE (constraints: xk[r,C k] , for k ? [K]) 2: (s, log w) ? (?, 0) . initialize empty sample with zero weight 3: for k ? T OPO S ORT ({? 1 , . . . , ? K }) do . topologically sort CGPMs using adjacency matrix 4: y?rk ? yrk ? {xp[r,t] ? s : (p, t) ? ? k } . retrieve required inputs at node k 5: log w ? log w + logpdf (G k , r, xk[r,C k] , ?, y?rk) . update weight by likelihood of constraint 6: xk[r,C k] ? simulate (G k , r, C k , xk[r,C k] , y?rk) . simulate unconstrained nodes 7: s ? s ? xk[r,C k ?C k] .

append all node values to sample 8:

3

return (s, w)

. return the overall sample and its weight

Analyzing satellites using CGPMs built from causal probabilistic programs, discriminative machine learning, and Bayesian non-parametrics

This section outlines a case study applying CGPMs to a database of 1163 satellites maintained by the Union of Concerned Scientists [12]. The dataset contains 23 numerical and categorical features of each satellite such as its material, functional, physical, orbital and economic characteristics. The list of variables and examples of three representative satellites are shown in Table 1. A detailed study of this database using BayesDB provided in [10]. Here, we compose the baseline CGPM in BayesDB, CrossCat [9], a non-parametric Bayesian structure learner for high dimensional data tables, with several CGPMs: a classical physics model written in VentureScript, a random forest classifier, factor analysis, and an ordinary least squares regressor. These composite models allow us to identify satellites that probably violate their orbital mechanics (Figure 2), as well as accurately infer the anticipated lifetimes of new satellites (Figure 3). We refer to [14, Section 6] for several more experiments on a broader set of data analysis tasks, as well as comparisons to baseline machine learning solutions. Name Country of Operator Operator Owner Users Purpose Class of Orbit Type of Orbit Perigee km Apogee km Eccentricity Period minutes Launch Mass kg Dry Mass kg Power watts Date of Launch Anticipated Lifetime Contractor Country of Contractor Launch Site Launch Vehicle Source Used for Orbital Data longitude radians of geo Inclination radians

International Space Station Multinational NASA/Multinational Government Scientific Research LEO Intermediate 401 422 0.00155 92.8 NaN NaN NaN 36119 30 Boeing Satellite Systems/Multinational Multinational Baikonur Cosmodrome Proton www.satellitedebris.net 12/12 NaN 0.9005899

AAUSat-3 Denmark Aalborg University Civil Technology Development LEO NaN 770 787 0.00119 100.42 0.8 NaN NaN 41330 1 Aalborg University Denmark Satish Dhawan Space Center PSLV SC - ASCR NaN 1.721418241

Advanced Orion 5 (NRO L-32, USA 223) USA National Reconnaissance Office (NRO) Military Electronic Surveillance GEO NaN 35500 35500 0 NaN 5000 NaN NaN 40503 NaN National Reconnaissance Laboratory USA Cape Canaveral Delta 4 Heavy SC - ASCR 1.761037215 0

Table 1: Variables in the satellite population, and three representative satellites. The records are multivariate, heterogeneously typed, and contain arbitrary patterns of missing data.

5

CREATE TABLE satellites_ucs FROM 'satellites.csv'; CREATE POPULATION satellites FOR satellites_ucs WITH SCHEMA (GUESS STATTYPES FOR (*));

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50

CREATE METAMODEL satellites_hybrid FOR satellites WITH BASELINE CROSSCAT (OVERRIDE GENERATIVE MODEL FOR type_of_orbit GIVEN apogee_km, perigee_km, period_minutes, users, class_of_orbit USING RANDOM_FOREST (num_categories = 7); OVERRIDE GENERATIVE MODEL FOR launch_mass_kg, dry_mass_kg, power_watts, perigee_km, apogee_km USING FACTOR_ANALYSIS (dimensionality = 2); OVERRIDE GENERATIVE MODEL FOR period_minutes AND EXPOSE kepler_cluster_id CATEGORICAL, kepler_noise NUMERICAL GIVEN apogee_km, perigee_km USING VENTURESCRIPT (program = ' define dpmm_kepler = () -¿ { // Definition of DPMM Kepler model program. assume keplers_law = (apogee, perigee) -¿ { (GM, earth_radius) = (398600, 6378); a = .5*(abs(apogee) + abs(perigee)) + earth_radius; 2 * pi * sqrt(a**3 / GM) / 60 }; // Latent variable priors. assume crp_alpha = gamma(1,1); assume cluster_id_sampler = make_crp(crp_alpha); assume noise_sampler = mem((cluster) -¿ make_nig_normal(1, 1, 1, 1)); // Simulator for latent variables (kepler_cluster_id and kepler_noise). assume sim_cluster_id = mem((rowid, apogee, perigee) -¿ { cluster_id_sampler() #rowid:1 }); assume sim_noise = mem((rowid, apogee, perigee) -¿ { cluster_id = sim_cluster_id(rowid, apogee, perigee); noise_sampler(cluster_id)() #rowid:2 }); // Simulator for observable variable (period_minutes). assume sim_period = mem((rowid, apogee, perigee) -¿ { keplers_law(apogee, perigee) + sim_noise(rowid, apogee, perigee) }); assume outputs = [sim_period, sim_cluster_id, sim_noise]; // List of output variables. }; // Procedures for observing the output variables. define obs_cluster_id = (rowid, apogee, perigee, value, label) -¿ { $label: observe sim_cluster_id($rowid, $apogee, $perigee) = atom(value); }; define obs_noise = (rowid, apogee, perigee, value, label) -¿ { $label: observe sim_noise($rowid, $apogee, $perigee) = value; }; define obs_period = (rowid, apogee, perigee, value, label) -¿ { theoretical_period = run(sample keplers_law($apogee, $perigee)); obs_noise(rowid, apogee, perigee, value - theoretical_period, label); }; define observers = [obs_period, obs_cluster_id, obs_noise]; // List of observer procedures. define inputs = ["apogee", "perigee"]; // List of input variables. define transition = (N) -¿ { default_markov_chain(N) }; // Transition operator. ')); INITIALIZE 10 MODELS FOR satellites_hybrid; ANALYZE satellites_hybrid FOR 100 ITERATIONS; INFER name, apogee_km, perigee_km, period_minutes, kepler_cluster_id, kepler_noise FROM satellites; Clusters Identified by Kepler CGPM

Period [mins]

4000

Geotail

3000

28

Cluster 1 Cluster 2 Cluster 3 Cluster 4 Theoretically Feasible Orbits

26

2000 Amos5 NavStar

1000 Meridian4

20000 30000 Perigee [km]

Geotail

Negligible Noticeable Large Extreme
24 23
Amos5
Meridian4 Orion6
21
0 10000
25
22 Orion6
0
Empirical Distribution of Orbital Deviations
27
Number of Satellites
5000
20 1e-10
40000
1e-5 1e0 1e5 Magntiude of Deviation from Keplers? Law [mins2]
1e10

Figure 2: A session in BayesDB to detect satellites whose orbits are likely violations of Kepler?s Third Law using a causal composable generative population model written in VentureScript. The dpmm_kepler CGPM (line 17) learns a DPMM on the residuals of each satellite?s deviation from its theoretical orbit. Both the cluster identity and inferred noise are exposed latent variables (line 14). Each dot in the scatter plot (left) is a satellite in the population, and its color represents the latent cluster assignment learned by dpmm_kepler. The histogram (right) shows that each of the four detected clusters roughly translates to a qualitative description of the deviation: yellow (negligible), magenta (noticeable), green (large), and blue (extreme). 6

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32

CREATE TABLE data_train FROM 'sat_train.csv'; .nullify data_train 'NaN';

def dummy_code_categoricals(frame, maximum=10): def dummy_code_categoricals(series): categories = pd.get_dummies(series, dummy_na=1) if len(categories.columns) ¿ maximum-1: return None if sum(categories[np.nan]) == 0: del categories[np.nan] categories.drop(categories.columns[-1], axis=1, inplace=1) return categories

CREATE POPULATION satellites FOR data_train WITH SCHEMA(GUESS STATTYPES FOR (*)); CREATE METAMODEL crosscat_ols FOR satellites WITH BASELINE CROSSCAT(OVERRIDE GENERATIVE MODEL FOR anticipated_lifetime GIVEN type_of_orbit, perigee_km, apogee_km, period_minutes, date_of_launch, launch_mass_kg USING LINEAR_REGRESSION);

def append_frames(base, right): for col in right.columns: base[col] = pd.DataFrame(right[col]) numerical = frame.select_dtypes([float]) categorical = frame.select_dtypes([object])

INITIALIZE 4 MODELS FOR crosscat_ols; ANALYZE crosscat_ols FOR 100 ITERATION WAIT;

categorical_coded = filter(lambda s: s is not None, [dummy_code_categoricals(categorical[c]) for c in categorical.columns])

CREATE TABLE data_test FROM 'sat_test.csv'; .nullify data_test 'NaN'; .sql INSERT INTO data_train SELECT * FROM data_test;

joined = numerical

CREATE TABLE predicted_lifetime AS INFER EXPLICIT PREDICT anticipated_lifetime CONFIDENCE prediction_confidence FROM satellites WHERE _rowid_ ¿ 1000;

for sub_frame in categorical_coded: append_frames(joined, sub_frame) return joined

(a) Full session in BayesDB which loads the training and test sets, creates a hybrid CGPM, and runs the regression using CrossCat+OLS.

(b) Ad-hoc Python routine (used by baselines) for coding nominal predictors in a dataframe with missing values and mixed data types.

Mean Squared Error

102

101

ridge ols lasso kernel forest bayesdb(crosscat+ols) bayesdb(crosscat)

100 1 10

102 Lines of Code

Figure 3: In a high-dimensional regression problem with mixed data types and missing data, the composite CGPM improves prediction accuracy over purely generative and purely discriminative baselines. The task is to infer the anticipated lifetime of a held-out satellite given categorical and numerical features such as type of orbit, launch mass, and orbital period. As feature vectors in the test set have missing entries, purely discriminative models (ridge, lasso, OLS) either heuristically impute missing features, or ignore the features and predict the anticipated lifetime using the mean in the training set. The purely generative model (CrossCat) can impute missing features from their joint distribution, but only indirectly mediates dependencies between the predictors and response through latent variables. The composite CGPM (CrossCat+OLS) in panel (a) combines advantages of both approaches; statistical imputation followed by regression on the features leads to improved predictive accuracy. The reduced code size is a result of using SQL, BQL, & MML, for preprocessing, model-building and predictive querying, as opposed to collections of ad-hoc scripts such as panel (b). Figure 2 shows the MML program for constructing the hybrid CGPM on the satellites population. In terms of the compositional formalism from Section 2.3, the CrossCat CGPM (specified by the MML BASELINE keyword) learns the joint distribution of variables at the ?root? of the network (i.e., all variables from Table 1 which do not appear as arguments to an MML OVERRIDE command). The dpmm_kepler CGPM in line 16 of the top panel in Figure 2 accepts apogee_km and perigee_km as input variables y = (A, P), and produces as output the period_minutes x = (T). These variables characterize the ellipticalporbit of a satellite and are constrained by the relationships e = (A ? P)/(A + P) and T = 2? ((A + P)/2))3 /GM where e is the eccentricity and GM 7

is a physical constant. The program specifies a stochastic version of Kepler?s Law using a Dirichlet process mixture model for the distribution over errors (between the theoretical and observed period), P ? DP(?, N ORMAL -I NVERSE -G AMMA(m, V, a, b)),

(?r , ?r2)—P ? P

r —{?r , ?r2 , yr } ? N ORMAL(?—?r , ?r2), where r := Tr ? K EPLER(Ar , Pr). The lower panels of Figure 2 illustrate how the dpmm_kepler CGPM clusters satellites based on the magnitude of the deviation from their theoretical orbits; the variables (deviation, cluster identity, etc) in these figures are obtained from the BQL query on line 50. For instance, the satellite Orion6 shown in the right panel of Figure 2, belongs to a component with ?extreme? deviation. Further investigation reveals that Orion6 has a recorded period 23.94 minutes, most likely a data entry error for the true period of 24 hours (1440 minutes); we have reported such errors to the maintainers of the database. The data analysis task in Figure 3 is to infer the anticipated_lifetime xr of a new satellite, given a set of features yr such as its type_of_orbit and perigee_km. A simple OLS regressor with normal errors is used for the response pG ols (xr —yr). The CrossCat baseline learns a joint generative model for the covariates pG crosscat (yr). The composite CGPM crosscat_ols built Figure 3 (left panel) thus carries the full joint distribution over the predictors and response pG (xr , yr), leading to more accurate predictions. Advantages of this hybrid approach are further discussed in the figure.

4 Related Work and Discussion

This paper has shown that it is possible to use a computational formalism in probabilistic programming to uniformly apply, combine, and compare a broad class of probabilistic data analysis techniques. By integrating CGPMs into BayesDB [10] and expressing their compositions in the Metamodeling Language, we have shown it is possible to combine CGPMs synthesized by automatic model discovery [9] with custom probabilistic programs, which accept and produce multivariate inputs and outputs, into coherent joint probabilistic models. Advantages of this hybrid approach to modeling and inference include combining the strengths of both generative and discriminative techniques, as well as savings in code complexity from the uniformity of the CGPM interface. While our experiments have constructed CGPMs using VentureScript and Python implementations, the general probabilistic programming interface of CGPMs makes it possible for BayesDB to interact with a variety systems such as BUGS [15], Stan [1], BLOG [11], Figaro [13], and others. Each of these systems provides varying levels of model expressiveness and inference capabilities, and can be used to be construct domain-specific CGPMs with different performance properties based on the data analysis task on hand. Moreover, by expressing the data analysis tasks in BayesDB using the model-independent Bayesian Query Language [10, Section 3], CGPMs can be queried without necessarily exposing their internal structures to end users. Taken together, these characteristics help illustrate the broad utility of the BayesDB probabilistic programming platform and architecture [14, Section 5], which in principle can be used to create and query novel combinations of black-box machine learning, statistical modeling, computer simulation, and probabilistic generative models.

Our applications have so far focused on CGPMs for analyzing populations from standard multivariate statistics. A promising area for future work is extending the computational abstraction of CGPMs, as well as the Metamodeling and Bayesian Query Languages, to cover analysis tasks in other domains such longitudinal populations [3], statistical relational settings [6], or natural language processing and computer vision. Another extension, important in practice, is developing alternative compositional algorithms for querying CGPMs (Section 2.3). The importance sampling strategy used for compositional simulate and logpdf may only be feasible when the networks are shallow and the constituent CGPMs are fairly noisy; better Monte Carlo strategies or perhaps even variational strategies may be needed for deeper networks. Additional future work for composite CGPMs include (i) algorithms for jointly learning the internal parameters of each individual CGPM, using, e.g., imputations from its parents, and (ii) new meta-algorithms for structure learning among a collection of compatible CGPMs, in a similar spirit to the non-parametric divide-and-conquer method from [9]. We hope the formalisms in this paper lead to practical, unifying tools for data analysis that integrate these ideas, and provide abstractions that enable the probabilistic programming community to collaboratively explore these research directions. 8

2 References

[1] B. Carpenter, A. Gelman, M. Hoffman, D. Lee, B. Goodrich, M. Betancourt, M. A. Brubaker, J. Guo, P. Li, and A. Riddell. Stan: A probabilistic programming language. J Stat Softw, 2016. [2] G. Casella and R. Berger. Statistical Inference. Duxbury advanced series in statistics and decision sciences. Thomson Learning, 2002. [3] M. Davidian and D. M. Giltinan. Nonlinear models for repeated measurement data, volume 62. CRC press, 1995. [4] L. Devroye. Sample-based non-uniform random variate generation. In Proceedings of the 18th conference on Winter simulation, pages 260?265. ACM, 1986. [5] D. Fink. A compendium of conjugate priors. 1997. [6] N. Friedman, L. Getoor, D. Koller, and A. Pfeffer. Learning probabilistic relational models. In Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence, IJCAI 99, Stockholm, Sweden, July 31 - August 6, 1999. 2 Volumes, 1450 pages, pages 1300?1309, 1999. [7] D. Koller and N. Friedman. Probabilistic graphical models: principles and techniques. MIT press, 2009. [8] V. Mansinghka, D. Selsam, and Y. Perov. Venture: a higher-order probabilistic programming platform with programmable inference. CoRR, abs/1404.0099, 2014. [9] V. Mansinghka, P. Shafto, E. Jonas, C. Petschulat, M. Gasner, and J. B. Tenenbaum. Crosscat: A fully bayesian nonparametric method for analyzing heterogeneous, high dimensional data. arXiv preprint arXiv:1512.01272, 2015. [10] V. Mansinghka, R. Tibbetts, J. Baxter, P. Shafto, and B. Eaves. Bayesdb: A probabilistic programming system for querying the probable implications of data. arXiv preprint arXiv:1512.05006, 2015. [11] B. Milch, B. Marthi, S. Russell, D. Sontag, D. L. Ong, and A. Kolobov. 1 blog: Probabilistic models with unknown objects.

Statistical relational learning, page 373, 2007. [12] U. of Concerned Scientists. UCS Satellite Database, 2015. [13] A. Pfeffer. Figaro: An object-oriented probabilistic programming language. Charles River Analytics Technical Report, 137, 2009. [14] F. Saad and V. Mansinghka. Probabilistic data analysis with probabilistic programming. arXiv preprint arXiv:1608.05347, 2016. [15] D. J. Spiegelhalter, A. Thomas, N. G. Best, W. Gilks, and D. Lunn. Bugs: Bayesian inference using gibbs sampling. Version 0.5,(version ii) http://www. mrc-bsu. cam. ac. uk/bugs, 19, 1996.

9