PowerCast: Mining and Forecasting Power Grid Sequences

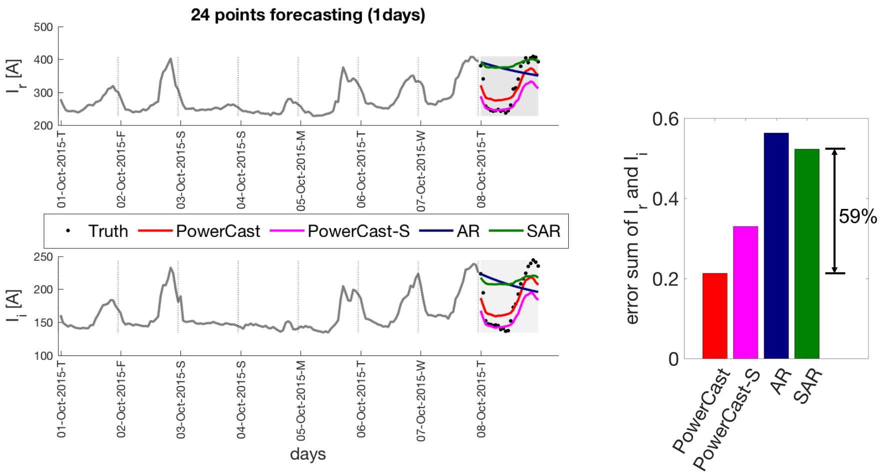
Hyun Ah Song[1], Bryan Hooi[2], Marko Jereminov[3], Amritanshu Pandey[3], Larry Pileggi[3], and Christos Faloutsos[1]

[1] School of Computer Science
[2] Department of Statistics
[3] Deptartment of Electrical and Computer Engineering
Carnegie Mellon University
{hyunahs}@cs.cmu.edu, {bhooi, mjeremin, amritanp, pileggi}@andrew.cmu.edu, {christos}@cs.cmu.edu

Abstract. What will be the power consumption of our institution at 8am for the upcoming days? What will happen to the power consumption of a small factory, if it wants to double (or half) its production? Technologies associated with the smart electrical grid are needed. Central to this process are algorithms that accurately model electrical load behavior, and forecast future electric power demand. However, existing power load models fail to accurately represent electrical load behavior in the grid. In this paper, we propose PowerCast, a novel domain-aware approach for forecasting the electrical power demand, by carefully incorporating domain knowledge. Our contributions are as follows: 1. Infusion of **domain expert knowledge**: We represent the time sequences using an equivalent circuit model, the "BIG" model, which allows for an *intuitive interpretation* of the power load, as the BIG model is derived from physics-based first principles. 2. **Forecasting of the power load**: Our PowerCast uses the BIG model, and provides (a) *accurate* prediction in multi-step-ahead forecasting, and (b) *extrapolations*, under *what-if* scenarios, such as variation in the demand (say, due to increase in the count of people on campus, or a decision to half the production in our factory etc.) 3. **Anomaly detection**: PowerCast can spot and, even explain, anomalies in the given time sequences. The experimental results based on two real world datasets of up to three weeks duration, demonstrate that PowerCast is able to forecast several steps ahead, with **59%** error reduction, compared to the competitors. Moreover, it is fast, and scales linearly with the duration of the sequences.

1 Introduction

The goal of the smart electrical grid is to manage the demand and supply of electricity while maintaining both efficiency and reliability. Indeed, [1] finds that improving the reliability of the U.S. grid could provide savings of around $49 *billion* per year and provide a 12% to 18% reduction in emissions, while improving efficiency could save an additional $20 *billion*. Toward this goal, monitoring



(a) (b)

Fig. 1: **PowerCast forecasts accurately.** (a) PowerCast (*red*) and PowerCast-S (*pink*) forecasts 24 steps (1-day) on I_r (top row), and I_i (bottom row) more accurately compared to competitors (AR: *blue*; SAR: *green*; ground truth *black circles*). (b) RMSE comparison of the methods

systems have been put in place, including Phasor Measurement Units (PMUs) and newer high-precision micro-PMUs [34]. Using these data sources to accurately model load behavior in the grid, as well as to forecast future power load, is important in protecting the grid from failure and for maintaining reliability.

Our main goal is to understand how a specific service area consumes power (a university campus, a small factory, a village or neighborhood), by studying its past behavior. Once we have a good model for the power-consumption behavior, then we can do forecasting (*how much power will our campus/factory need tomorrow*), spot anomalies (when our forecast is too far from what actually happened), and answer "what-if" scenarios, like *how much power will we need, during spring-break on campus*; or *during a heat-wave, in our neighborhood.*

We focus on these two problems: forecasting, and 'what-if' scenarios. The informal definitions, are as follows. Note that alternating current (AC) (I) and voltage (V) are modeled as complex numbers.

Informal Problem 1 (Multi-step forecasting on power grid)

- ***Given:*** *real and imaginary current* $(I_r(t), I_i(t))$ *and voltage* $(V_r(t), V_i(t))$ *of previous* N *time points* $(t = 1, \cdots, N)$,
- ***Forecast:*** *the electric current demand, for* N_f *steps in the future (i.e., guess* $I_r(t), I_i(t)$, *for* $t = N+1, \cdots, N+N_f$)

Throughout this work, we assume that the voltage in the future, is given ($V_r(t), V_i(t)$ for $t = N+1, \cdots, N+N_f$). This is realistic: except for rare brown-outs, power-plants try hard to provide near-constant voltage to consumers.

In Figure 1, 24 step (1-day) forecasting result on Lawrence Berkeley National Laboratory (LBNL) Open μPMU project data [34] is shown. In (a), PowerCast (*red*) and PowerCast-S (*pink*) provides more accurate forecasting, following closely to the truth (*black* dots) while the competitors (AR: *blue*, SAR:*green*) fail. We observe that PowerCast is able to forecast the the current demand with an accurate daily pattern while AR and SAR fail to consider the daily pattern. In (b), a quantitative comparison on the forecasting accuracy is shown. PowerCast forecasts with **59%** reduction in error, compared to the competitors.

An additional benefit of our domain-aware approach is that it can handle the *what-if* extrapolation problem:

Informal Problem 2 ('what-if' extrapolation)

- **Given** *the historical data, as above (real and imaginary current $I_r(t)$, $I_i(t)$, and voltage $V_r(t)$, $V_i(t)$, $t = 1, \ldots, N$*
- **Guess** *what will be the power demand in the future (currents $I_r(t)$, $I_i(t)$, $t > N$), if, say, the student population doubles on our campus (or our factory cuts production in half, etc).*

Handling what-if scenarios is beyond the reach of black-box methods (ARIMA etc), exactly because it demands domain knowledge, which we carefully infuse, using the established "BIG" model.

Our contributions are as follows:

1. **Infusion of domain knowledge**: Our method is domain-aware: among the many aggregated load models (BIG, "PQ", "ZIP"), we chose the first, because it allows for an accurate and *intuitive interpretation* of the power load, being derived from physics-based first principles.
2. **Forecasting/What-if**: The proposed PowerCast (a) leads to *more accurate* forecasts (up to **59%** lower error) compared to textbook, black-box methods, and (b) it can answer *what-if* scenarios that black-box methods can not.
3. **Anomaly detection**: PowerCast can spot, and even explain, anomalies in the given time sequences. (see section 4.3, Figure 6)

Reproducibility: Our code is open-sourced at: `www.cs.cmu.edu/~hyunahs/code/PowerCast.zip`; the LBNL dataset is at: `powerdata.lbl.gov/`.

The structure of the paper is typical: We give the background and related works (section 2), our proposed method (section 3), experiments (section 4), and conclusions (section 5).

2 Background and Related Work

2.1 Related works

Load model for aggregated electrical demand: the BIG model Electrical load modeling for power system analysis has been traditionally done using the constant power *constant PQ* and *ZIP* models [28]. However, industry experience has shown that these models incorrectly characterize load behavior [22]. Recent advances in steady-state power system simulations [6, 27] have introduced the use of physics based state variables, i.e. currents and voltages, and shown that load behavior at a given time instance can be accurately described by a linear relationship between current and voltage. From circuit theory, this can be represented by a parallel or series combination of reactance (B) and conductance (G). This model captures both voltage magnitude and angle information, in contrast to existing traditional load models [28]. Further adding a current source (I) results in the BIG load model, which accurately captures load sensitivities to voltage variations over a period of time [13]. The complete description of the BIG load model is given in section 3.

Table 1: **PowerCast captures all of the listed properties**. *AR++* = ARIMA, seasonal ARIMA etc. *LDS* = Linear Dynamical Systems. '*Pattern discovery*' = concept / latent-variable discovery.

Properties	AR++[5, 39, 4]	Kalman/LDS[12, 15]	PARAFAC++[18, 31]	HMM++[19, 37]	AutoCyclone [35]	PowerCast
Pattern (=concept) discovery			✓	✓	✓	✔
Forecasting	✓	✓		✓		✔
Seasonal patterns	✓				✓	✔
Domain knowledge inclusion						✔
What-if scenarios						✔

Time series forecasting Classical methods for time-series forecasting include the family of autoregression (AR)-based methods, including ARMA, exponential smoothing [7], ARIMA [5], ARMAX [39] and ARFIMA [4] models. Seasonal ARIMA [5] incorporates pre-determined, constant periods allowing the model to capture seasonal patterns. More recent generalizations include TBATS [9] which allows for more complex seasonality patterns. Other methods for time series forecasting include Kalman filtering [15], Linear Dynamical Systems (LDS) [12], Hidden Markov Models (HMMs) [19], wavelet-based methods such as AWSOM [29] and non-linear dynamical systems such as RegimeCast [23].

In the area of modeling of aggregated load in the power grid, autoregression is an extremely common approach due to its simplicity and interpretability [32, 33, 8, 11]. [30] uses ARMA models and exponential smoothing for short-term load forecasting. [26] uses a two-step procedure of seasonality removal, followed by ARMA with hyperbolic noise.

Tensor-based time series analysis Tensor decomposition methods, including PARAFAC decomposition, Tucker decomposition [17, 18], multilinear principal components analysis [21], and Bayesian tensor analysis [36] are powerful tools to understand latent factors of a target dataset. Recent work such as Marble [10] and Rubik [38] applies tensor factorization for concept analysis with domain knowledge. In terms of time series applications, tensors have been used for modeling multiple coevolving time series for epidemiology [25], community discovery [20], and concept discovery [16]. [24] uses a tensor-based approach for forecasting based on complex sequences of timestamped events. AutoCyclone [35] models seasonality in tensor datasets by 'folding' the data into a higher-order tensor.

However, all of the aforementioned methods are typically not based on electrical load models derived from first principles, as is the case for the BIG load model, which the herein proposed method is based on, thus allowing for more accurate forecasts and interpretable models. In addition, our approach allows for *what-if* scenarios, e.g. variation of electrical load due to change in number of people in the building.

2.2 Background

BIG model The BIG equivalent circuit model [14] is a linear representation of the current using the voltage data follows:

$$I_r(t) = G(t)V_r(t) - B(t)V_i(t) + \alpha_r(t) \tag{1}$$

$$I_i(t) = B(t)V_r(t) + G(t)V_i(t) + \alpha_i(t) \tag{2}$$

By modeling the given time sequences I_r, I_i, V_r, V_i using the BIG model, we learn the BIG parameters G, B, α_r, α_i that provides us with an intuitive interpretation of the aggregated conductance (G) and susceptance (B) of the power grid as it varies over time. In PowerCast, we convert the given time sequences to the BIG domain $(G, B, \alpha_r, \alpha_i)$ and further proceed with pattern analysis and forecasting. As we will demonstrate in our analysis with experimental results in section 4, working in the BIG domain provides more stable and interpretable analysis of the power systems: understanding the power system (section 4.1), accurate forecasting and exploration of various what-if scenarios (section 4.2), and anomaly detection (section 4.3).

3 Proposed Method

In this work, our interest is in: 1) including the domain knowledge of the intrinsic behavior of the campus (G, B), 2) capture the latent periodic patterns in the sequences, and 3) do forecasting.

The core parts of PowerCast algorithm is: 1) to convert the given time sequences into the BIG domain, and 2) forecasting in tensor structure. By converting the time sequences into the BIG domain, we try to understand the power system (G, B value range, dynamics, ratio, etc) that generated the sequences I_r, I_i, V_r, V_i that we observe, which enables us to do forecasting under *what-if* scenarios of change in the power systems on campus. Working with tensors tells us the long-term, and daily patterns. By performing forecasting based on the learned patterns from the past, PowerCast provides more stable forecasting result (section 4.2). In this section, we will explain how we convert the time sequences into the BIG domain and how we analyze tensors to do forecasting in more details.

A table of symbols that is used throughout this paper is shown in Table 2.

Table 2: Symbols and definitions

Symbols	Definitions
I_r, I_i, V_r, V_i	given time sequences. Real and imaginary part of complex current and voltage.
N	total number of timeticks in the given time sequences. $N = N_d \times N_{dt}$
N_d	number of days in the given time sequences
N_{dt}	number of time points for each day
G, B, α_r, α_i	BIG parameters. (conductance, susceptance, offset to I_r and I_i)
$\mathcal{X}$	a tensor of given time sequences I_r,I_i,V_r,V_i. $\mathcal{X} \in \mathbb{R}^{N_d \times N_{dt} \times 4}$
$\mathcal{B}_a$	a tensor of BIG parameters G, B, α_r, α_i. $\mathcal{B}_a \in \mathbb{R}^{N_d \times N_{dt} \times 4}$
$\mathcal{B}_b$	a tensor of BIG parameters G, B, α_r, α_i, after extension. $\mathcal{B}_b \in \mathbb{R}^{(N_d+N_{fd}) \times N_{dt} \times 4}$
N_{fd}	number of days for forecasting
N_f	number of time steps for forecasting. $N_f = N_{fd} \times N_{dt}$
P_{ar}	auto-regression parameter
R	rank for tensor decomposition
N_w, σ	Parameters for Gaussian filter. (window size, standard deviation of Gaussian)

3.1 PowerCast algorithm

Pseudocode of PowerCast is described in Algorithm 1, where each function will be explained in more detail. In Figure 2, a flowchart of the Algorithm 1 is illustrated.

Step 1: Seq2Tensor (Tensor construction): Given four time sequences $I_r(1:N), I_i(1:N), V_r(1:N), V_i(1:N)$, we construct a tensor $\mathcal{X} \in \mathbb{R}^{N_d \times N_{dt} \times 4}$

Algorithm 1: PowerCast.

Data: $I_r(1:N), I_i(1:N), V_r(1:N+N_f), V_i(1:N+N_f)$
Result: $\hat{I}_r(N+1:N+N_f), \hat{I}_i(N+1:N+N_f)$
Construct a tensor from given sequences:;
$[\mathcal{X}] = \text{Seq2Tensor}(I_r(1:N), I_i(1:N), V_r(1:N), V_i(1:N))$;
Covert the data to BIG domain (fit the data to BIG model):;
$[\mathcal{B}_a] = \text{X2B}(\mathcal{X})$;
Extend the tensor for forecasting:;
$[\mathcal{B}_b] = \text{Bextension}(\mathcal{B}_a)$;
Recover the data from BIG domain:;
$[\hat{I}_r(N+1:N+N_f), \hat{I}_i(N+1:N+N_f)] =$
$\text{B2Seq}(\mathcal{B}_b, V_r(N+1:N+N_f), V_i(N+1:N+N_f))$;

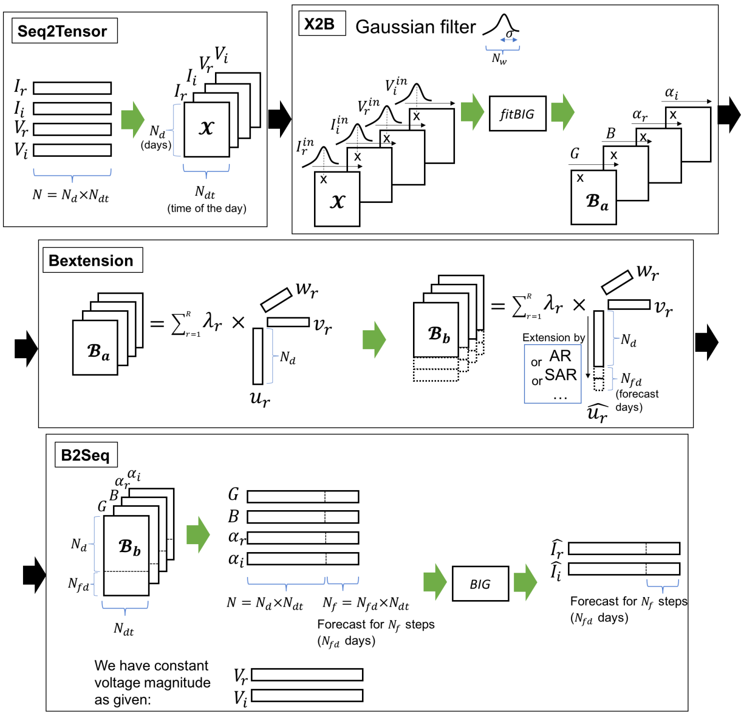


Fig. 2: **Flowchart of PowerCast**.

by cutting the sequences into daily unit as described in Figure 2 "*Seq2Tensor*." The first, second, and third mode of the tensor corresponds to *days*, *hour of the day*, and *given time sequences*, respectively. For example, if the time sequences are hourly samples (24 points per day) for 7 days, then the dimensions of the first and second modes become 7 (days) and 24 (samples per day) respectively.

Step 2: X2B (Convert the data into BIG domain): Then we convert the data tensor ($\mathcal{X}$) to BIG domain ($\mathcal{B}_a$) by fitting the data to BIG model as described in "*X2B*" in Figure 2. For fitting the BIG parameters, as we cannot fit the BIG parameters to a single point, we consider multiple points before and after the current timetick by apply a moving Gaussian filter to a window size of N_w, standard deviation σ to the data.

Function *fitBIG* takes in $[I_r^{in}, I_i^{in}, V_r^{in}, V_i^{in}]$ and fits the BIG models in equations (1) and (2) to the given window of data, using weighted least squares fitting, using this Gaussian filter as weights. Then we construct a new tensor $\mathcal{B}_a$ consisting of BIG parameters.

We next combine these fitted values into a tensor in the BIG domain whose first, second, and third mode of the tensor corresponds to *days*, *hour of the day*, and *BIG parameters*, respectively.

Step 3: Bextension (Extend the tensor for forecasting): After constructing a tensor from the given data in the BIG domain, we decompose the tensor using canonical polyadic alternating least squares (CP_ALS) algorithm [17, 18]: $\mathcal{B}_a = \sum_{r=1}^{R} \lambda_r \times \mathbf{u}_r \times \mathbf{v}_r \times \mathbf{w}_r$. Here, $\mathbf{u}, \mathbf{v}, \mathbf{w}$ corresponds to the hidden variables across multiple days, within a day, and among the BIG parameters. We will refer to these hidden variables as *long-term-concepts, daily-concepts, users-profile-concepts*, respectively.

After learning the tensor components, we extend the first mode of the tensor ($\mathbf{u}$, corresponding to *days*) for N_{fd} more points for forecasting - we conduct auto-regression on the $\mathbf{u}$ vector and forecast/extend it. We learn the AR parameters for P_{ar}^{th} order auto-regression $AR(P_{ar})$ using least squares on $\mathbf{u}$. Then we use AR parameters for the extension of the $\mathbf{u}$ vector to get an extended vector $\hat{\mathbf{u}}$: $\hat{\mathbf{u}}_r(N_d + f) = c_0 + \sum_{p=1}^{P_{ar}} c_p \hat{\mathbf{u}}_r(N_d + f - p)$

From the extended components $\hat{\mathbf{u}}$, we can reconstruct an *extended* tensor, $\mathcal{B}_b = \sum_{r=1}^{R} \lambda_r \times \hat{\mathbf{u}}_r \times \mathbf{v}_r \times \mathbf{w}_r$ with next N_{fd} days worth of data. A detailed steps are illustrated in Figure 2 "*Bextension*" part.

Step 4: B2Seq (Recover the data from BIG domain): Now that we have a extended tensor $\mathcal{B}_b$ on BIG domain, we need to recover the N_{fd} days of forecast data back into the original data domain. A detailed description of steps is illustrated in Figure 2 "*B2Seq*" part. We matricize the tensor into four matrices and reshape each matrix into a vector in row-wise manner. We apply the BIG equations (1), and (2) to the assumed voltage data and forecast corresponding $I_r(t), I_i(t)$ for the next $t = N+1, \cdots, N+N_f$ steps.

3.2 PowerCast-S

PowerCast-S is a variation of PowerCast. In function **Bextension**, we can use any type of forecasting function in exchange of *AR*.

We tried applying seasonal periodicity on the *long-term-concepts* ($\mathbf{u}$ vector), but on our dataset (which does not span over long enough period of time to capture the seasonal pattern) PowerCast-S did not work well. Thus we recommend users to use plain PowerCast unless: 1) you have a long enough history of the time sequences to capture the weekly pattern, and 2) you strongly believe that there is a seasonality in the data (weekly, monthly, seasonal, etc).

4 Experiments

In this section we conduct experiments on the real world data to answer the following research questions: **Q1: Interpretability**, **Q2: Forecasting**, and **Q3: Anomaly detection**.

Dataset description: We next give a brief explanation of the datasets we used for experiments. All datasets are 5-tuples of the form (V_r, V_i, I_r, I_i, t).

- **CMU data:** The voltage and current measurements were recorded for the Carnegie Mellon University (CMU) campus for 23 days, from July 29, 2016 to August 20, 2016. The time sequences were sampled every hour ($N_{dt} = 24$).
- **LBNL data:** This is from the Lawrence Berkeley National Laboratory (LBNL) Open μPMU project[4] [34]. It spans 3 months, with sampling rate of 120Hz; but we used only the interval (October 1, 2015 to October 8, 2015) and we down-sampled it to hourly samples ($N_{dt} = 24$). Moreover, we de-noised it using moving averages.

Experimental setup: For the parameters, we used $P_{ar} = 1$ for $AR(P_{ar})$ for extension of the $\mathbf{u}$ vector, $R = 2$ for the tensor decomposition, and $N_w = 5$, $\sigma = 0.5$ for the moving Gaussian filter. We used tensor tool box from [3] [2].

Baseline methods: As baseline methods, we conducted experiments using autoregression (AR) and seasonal auto-regression (SAR) for comparison with our POWERCAST. For the baseline experiments, we assume that we do not have the domain knowledge on the power systems such as BIG model, etc. Thus we run AR and SAR methods on the given time sequences of I_r, I_i directly. The autoregression order for AR and SAR are determined via AIC criterion.

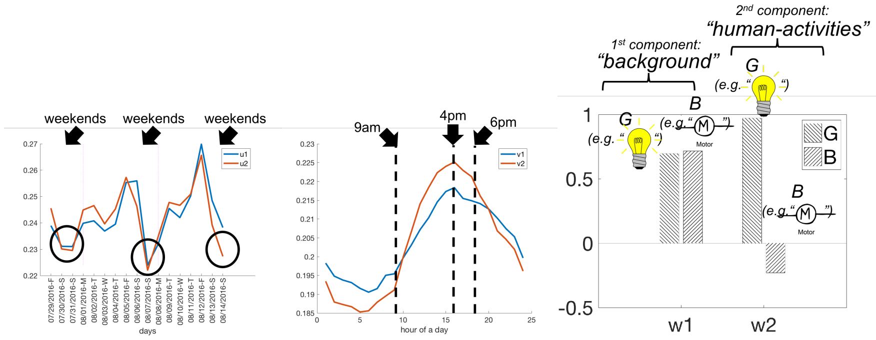
4.1 Q1: Interpretability

In this section we analyze the hidden variables in CMU data, and interpret the result using our domain knowledge.

As a reminder, G can be interpreted as a component of electrical load that contributes to real power consumption (for e.g. *light bulbs*), while B can be interpreted as a component of electric load that contributes to reactive power (for e.g. lagging reactive power (+Q) is absorbed by the *motors* where leading reactive power (-Q) is supplied by the capacitor) In our analysis below, we include images of "*light bulbs*" and "*motors*" to represent one example of the sources of G and B for more intuitive understanding.

Observations 1 (Weekly pattern in Figure 3 (a))

[4] http://powerdata.lbl.gov/



(a) *long-term-concepts* (b) *daily-concepts* (c) *users-profile-concepts*

Fig. 3: (a) **Weekly periodicity** in *long-term-concepts* (**u**), (b) **Daily pattern** in *daily-concepts* (**v**), (c) ***users-profile-concepts*** (**w**), on CMU data

- The *long-term-concepts* (**u** vector) shows weekly periodicity accounting for the activities across the days (drops during weekends)
- Both of the two components capture similar weekly pattern

Observations 2 (Daily pattern in Figure 3 (b))

- The *daily-concepts* (**v** vector) captures daily activity patterns throughout the day, lowest during the night (midnight to 9am), and grows from 9am, when peaking start coming to campus, peaking at 4pm.
- The first component, $\mathbf{v}_1$, shows smoother varying pattern throughout the day. We can think of this component as the representation of the "*background*" activities, as it is insensitive to the dynamics of the human activities throughout the day.
- The second component, $\mathbf{v}_2$, shows more dynamic changes with 1) deeper drop during the night, 2) faster growth in early stage (logarithmic), and 3) rapid decay after the peak. This pattern closely follows the dynamics of the expected human activities throughout the day. We can think of this component as the representation of the "*human-activity*" factor.

Observations 3 (G and B factor analysis in Figure 3 (c))

- The *users-profile-concepts* (**w** vector) explains how much G and B account for each of the component.
- The first component - hidden variable (left of Figure 3(a)) accounts for both G (e.g"*light bulbs*") and B (e.g."*motors*") in a close to 50-50 ratio. Combined with our interpretation in the *daily-concepts* for plot (b), we can interpret the first component as the *background* component that explains the background activities of the G and B (e.g. *light bulbs* and *motors*) factors that are operated independently of the dynamics of the major human factors. (basic

facilities such as air conditioning units, *light bulbs*, etc, that runs throughout the day to maintain the minimum required conditions in the buildings)

- The second component (right of Figure 3(a)) accounts dominantly for G (e.g.“*light bulbs*”). Aligned with our interpretation of the second component as accounting for the “*human-activity*” factor, this tells us that G (e.g. *light bulbs*) is more dependent on the dynamics of the daily activities compared to B. This is reasonable since addition of one more person on campus may result in turning on 10 G-related facilities (e.g. *light bulbs*) while it merely affects B-related facilities (e.g. *motors*- air-conditioner, heaters, etc) that are operated to maintain the background conditions regardless of the human factors.

4.2 Q2: Forecasting

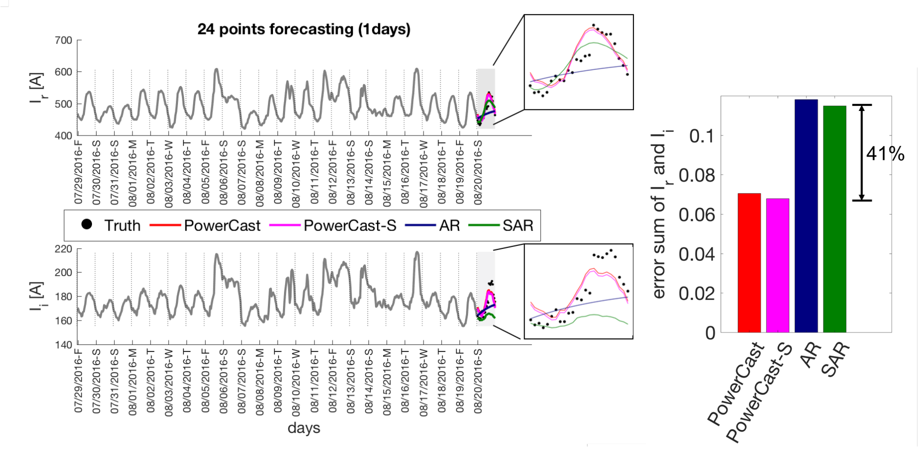
In this section we demonstrate forecasting results by PowerCast including various *what-if* scenarios.

Multi-step forecasting We start by showing multi-step ahead forecasting by PowerCast in comparison with other competitors on two different real world datasets.

- **CMU data**: Comparisons on the forecasting by PowerCast and competitors are shown for $N_f = 24$ steps (N_{fd} = 1-day) (Figure 4).
  In (a), we observe that PowerCast and PowerCast-S are able to demonstrate the daily periodic pattern while competitors fails to correctly demonstrate daily periodicity. In (b), a quantitative comparison is given by root mean square error (RMSE) sum of I_r, I_i forecast results. The RMSE is computed as follows: $RMSE(x, \hat{x}) = \sqrt{\frac{\sum_{t=1}^{N_f}(x(t)-\hat{x}(t))^2)}{\sum_{t=1}^{N_f}(x(t))^2}}$
- **LBNL data**:
  Figure 1 shows a similar quantitative comparison on the LBNL dataset, where PowerCast also outperforms its competitors.

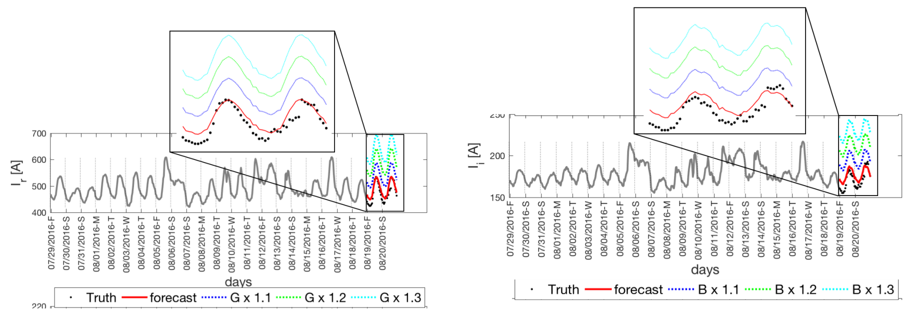
What-if scenario If we know that the number of people on campus will increase by 10% tomorrow due to a big event, how much more power do we expect? Naive forecasting methods cannot tell us the future demand under various scenarios.

In this section, we illustrate how PowerCast can handle forecasting under various *what-if* scenarios by adjusting G, B accordingly. We created three scenarios, assuming that we will have $\{10, 20, 30\}$% more activities on campus for next 2 days (thus increased values for G and B); and performed forecasting of the power demand under our scenarios. In Figure 5, 48 step (2-day) ahead forecasting on CMU data under our scenarios are shown for (a) real current demand and (b) imaginary current demand. We see that I_r and I_i increase, according to the “BIG” model (see (a) and (b) respectively), allowing us to plan ahead. (Red, blue, green, and cyan, correspond to $\{0, 10, 20, 30\}$% increases).



(a) (b)

Fig. 4: **PowerCast forecasts 24 steps (1-day) accurately on CMU data** (a) PowerCast (*red*) and PowerCast-S (*pink*) forecasts 24 steps (I_r on the top row, and I_i on the bottom row) more accurately compared to the competitors (AR: *blue* and SAR: *green*). (b) RMSE comparison of the methods. PowerCast achieves **41%** error reduction.

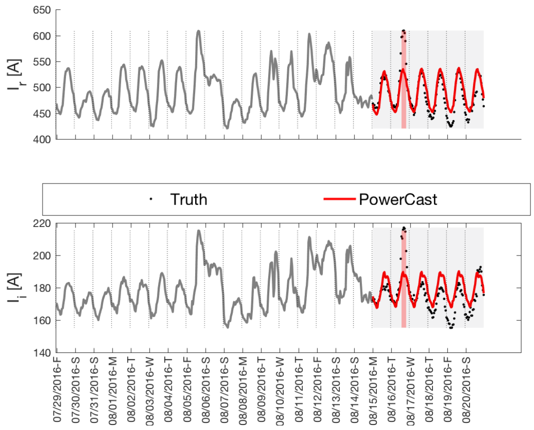


(a) (b)

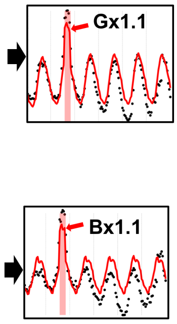
Fig. 5: **PowerCast can handle forecasting under various what-if scenarios** on demands for (a) I_r and (b) I_i on CMU data

4.3 Q3: Anomaly detection

In Figure 6 (a) $N_f = 144$ step forecasting ($N_{fd} = 6$ days) on CMU data is shown in solid red along with the actual reading in black circles. We report the highest deviation from our PowerCast forecast, with a red vertical line. This



(a) Anomaly detection



(b) Possible Explanation

Fig. 6: **PowerCast spots anomalies**: (a) August 16, 2017 (spotted, and marked with red stripe) was during the new graduate student orientation. (b) PowerCast shows that a 10% increase in activities ('*B*' and '*G*'), could explain this spike.

happened on Tuesday, August 16, which was in the middle of the 3-day new graduate student orientation, probably ≈1,000 people.

The natural follow up question is: Was the extra population the reason for this anomaly? The answer seems 'no', for two reasons: extra people would mainly boost I_r, that is 'G' (e.g. "*light bulbs*"), as our "*human-activity*" component showed in Figure 3, but we have spikes in both I_r and I_i; the second reason is that there was no spike during the other two days of orientation.

The only explanation is that something boosted the "*background*" component of Figure 3 (e.g., "*motors*", like elevators, heaters, air-conditioners), resulting in roughly equal boost to both I_r and I_i. Does reality corroborate the conjecture? The answer is '*yes*': the temperature on Aug. 16 spiked, to a scorching 88°F ($\approx 30.5°C$), while the surrounding days was closer to $80°F$. This is exactly what we show in (b): PowerCast shows that for a 10% increase in the component "*background*" (air-conditioners, etc), we get a 10% increase in both G (e.g. "*light bulbs*") and B (e.g. "*motors*"); the resulting answers correspond to the red line in (b), which is very close to the ground-truth black circles.

In short, PowerCast can not only spot anomalies, but also give hints about their cause.

4.4 Scalability

In Figure 7, the wall clock time for PowerCast to run on a dataset of different number of timeticks (N) is plotted in black solid dots, along with a linear line in blue. The time was measured for 24 step (1-day) forecast given past time sequences of length $N = 240 - 528$ timeticks ($N_d = 10 - 22$ days) on CMU data.

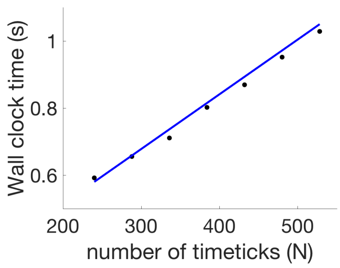


Fig. 7: **PowerCast scales linearly** with respect to the number of timeticks (N)

5 Conclusions

We proposed PowerCast, a novel domain-aware method to mine, understand, and forecast the power demand of an enterprise (university, factory, neighborhood). Our contributions are as follows:

1. **Infusion of domain knowledge**: We carefully picked a successful electric-power model, "BIG", which is derived from first-principles, and allows for *intuitive interpretation* of the power system of interest.
2. **Forecasting/What-if**: Our domain-aware approach, coupled with the BIG model provides (a) *accurate* prediction in multi-step ahead forecasting with up to 59% lower error, and (b) answers *what-if* scenarios, like "what will be our power demands when 80% of our students leave campus for March-break"
3. **Anomaly detection**: PowerCast can spot anomalies and explain as we show in section 4.3, Figure 6.

Reproducibility: Our code is open-sourced at: `www.cs.cmu.edu/~hyunahs/code/PowerCast.zip`, and the LBNL dataset is at: `powerdata.lbl.gov`

Acknowledgments. We would like to thank Dr. Tsubasa Takahashi, for discussions and feedback.

This material is based upon work supported by the National Science Foundation under Grant No. IIS-1247489, and by the Army Research Laboratory under Cooperative Agreement Number W911NF-09-2-0053. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation, or other funding parties. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation here on.

References

1. Amin, S.M.: U.S. grid gets less reliable [The Data]. IEEE Spectrum 48(1), 80–80 (2011)
2. Bader, B.W., Kolda, T.G.: Algorithm 862: MATLAB tensor classes for fast algorithm prototyping. ACM Transactions on Mathematical Software 32(4), 635–653 (December 2006)
3. Bader, B.W., Kolda, T.G., et al.: Matlab tensor toolbox version 2.6. Available online (February 2015), `http://www.sandia.gov/~tgkolda/TensorToolbox/`
4. Beran, J.: Statistics for long-memory processes, vol. 61. CRC press (1994)
5. Box, G.E., Jenkins, G.M., Reinsel, G.C., Ljung, G.M.: Time series analysis: forecasting and control. John Wiley & Sons (2015)
6. Bromberg, D.M., Jereminov, M., Li, X., Hug, G., Pileggi, L.: An equivalent circuit formulation of the power flow problem with current and voltage state variables. In: PowerTech, 2015 IEEE Eindhoven. pp. 1–6 (2015)
7. Brown, R.G.: Statistical forecasting for inventory control. McGraw/Hill (1959)
8. Bunn, D., Farmer, E.D.: Comparative models for electrical load forecasting (1985)
9. De Livera, A.M., Hyndman, R.J., Snyder, R.D.: Forecasting time series with complex seasonal patterns using exponential smoothing. Journal of the American Statistical Association 106(496), 1513–1527 (2011)
10. Ho, J.C., Ghosh, J., Sun, J.: Marble: high-throughput phenotyping from electronic health records via sparse nonnegative tensor factorization. In: ACM SIGKDD. pp. 115–124 (2014)
11. Hyndman, R.J., Fan, S.: Density forecasting for long-term peak electricity demand. IEEE Transactions on Power Systems 25(2), 1142–1153 (2010)
12. Jain, A., Chang, E.Y., Wang, Y.F.: Adaptive stream resource management using kalman filters. In: ACM SIGMOD. pp. 11–22 (2004)
13. Jereminov, M., Bromberg, D.M., Li, X., Hug, G., Pileggi, L.: Improving robustness and modeling generality for power flow analysis. In: Transmission and Distribution Conference and Exposition (T&D), 2016 IEEE/PES. pp. 1–5 (2016)
14. Jereminov, M., Pandey, A., Song, H.A., Hooi, B., Faloutsos, C., Pileggi, L.: Linear load model for robust power system analysis. In: IEEE PES Innovative Smart Grid Technologies. p. (submitted) (2017)
15. Kalman, R.E.: A new approach to linear filtering and prediction problems. Journal of Basic Engineering 82(1), 35–45 (1960)
16. Kang, U., Papalexakis, E., Harpale, A., Faloutsos, C.: Gigatensor: scaling tensor analysis up by 100 times-algorithms and discoveries. In: ACM SIGKDD. pp. 316–324 (2012)
17. Kolda, T.G., Bader, B.W.: Tensor decompositions and applications. SIAM review 51(3), 455–500 (2009)
18. Kolda, T.G., Sun, J.: Scalable tensor decompositions for multi-aspect data mining. In: ICDM. pp. 363–372 (2008)
19. Letchner, J., Re, C., Balazinska, M., Philipose, M.: Access methods for markovian streams. In: ICDE. pp. 246–257 (2009)

20. Lin, Y.R., Sun, J., Sundaram, H., Kelliher, A., Castro, P., Konuru, R.: Community discovery via metagraph factorization. ACM Transactions on Knowledge Discovery from Data (TKDD) 5(3), 17 (2011)
21. Lu, H., Plataniotis, K.N., Venetsanopoulos, A.N.: Multilinear principal component analysis of tensor objects for recognition. In: ICPR. vol. 2, pp. 776–779 (2006)
22. Martí, J.R., Ahmadi, H., Bashualdo, L.: Linear power-flow formulation based on a voltage-dependent load model. IEEE Transactions on Power Delivery 28(3), 1682–1690 (2013)
23. Matsubara, Y., Sakurai, Y.: Regime shifts in streams: Real-time forecasting of co-evolving time sequences. In: ACM SIGKDD. pp. 1045–1054 (2016)
24. Matsubara, Y., Sakurai, Y., Faloutsos, C., Iwata, T., Yoshikawa, M.: Fast mining and forecasting of complex time-stamped events. In: ACM SIGKDD. pp. 271–279 (2012)
25. Matsubara, Y., Sakurai, Y., Van Panhuis, W.G., Faloutsos, C.: Funnel: automatic mining of spatially coevolving epidemics. In: ACM SIGKDD. pp. 105–114 (2014)
26. Nowicka-Zagrajek, J., Weron, R.: Modeling electricity loads in california: Arma models with hyperbolic noise. Signal Processing 82(12), 1903–1915 (2002)
27. Pandey, A., Jereminov, M., Hug, G., Pileggi, L.: Improving power flow robustness via circuit simulation methods. In: PES General Meeting (2017)
28. Pandey, A., Jereminov, M., Li, X., Hug, G., Pileggi, L.: Aggregated load and generation equivalent circuit models with semi-empirical data fitting. In: Green Energy and Systems Conference (IGSEC), 2016 IEEE. pp. 1–6 (2016)
29. Papadimitriou, S., Brockwell, A., Faloutsos, C.: Adaptive, hands-off stream mining. In: VLDB. pp. 560–571 (2003)
30. Park, J., Park, Y., Lee, K.: Composite modeling for adaptive short-term load forecasting. IEEE Transactions on Power Systems 6(2), 450–457 (1991)
31. Rogers, M., Li, L., Russell, S.J.: Multilinear dynamical systems for tensor time series. In: NIPS. pp. 2634–2642 (2013)
32. Smith, M.: Modeling and short-term forecasting of new south wales electricity system load. Journal of Business & Economic Statistics 18(4), 465–478 (2000)
33. Soliman, S., Persaud, S., El-Nagar, K., El-Hawary, M.: Application of least absolute value parameter estimation based on linear programming to short-term load forecasting. International Journal of Electrical Power & Energy Systems 19(3), 209–216 (1997)
34. Stewart, E.M., Liao, A., Roberts, C.: Open pmu: A real world reference distribution micro-phasor measurement unit data set for research and application development (10/2016 2016)
35. Takahashi, T., Hooi, B., Faloutsos, C.: Autocyclone: Automatic mining of cyclic online activities with robust tensor factorization. In: WWW. pp. 213–221 (2017)
36. Tao, D., Song, M., Li, X., Shen, J., Sun, J., Wu, X., Faloutsos, C., Maybank, S.J.: Bayesian tensor approach for 3-d face modeling. IEEE Transactions on Circuits and Systems for Video Technology 18(10), 1397–1410 (2008)
37. Wang, P., Wang, H., Wang, W.: Finding semantics in time series. In: ACM SIGMOD. pp. 385–396 (2011)
38. Wang, Y., Chen, R., Ghosh, J., Denny, J.C., Kho, A., Chen, Y., Malin, B.A., Sun, J.: Rubik: Knowledge guided tensor factorization and completion for health data analytics. In: ACM SIGKDD. pp. 1265–1274 (2015)
39. Yang, H.T., Huang, C.M.: A new short-term load forecasting approach using self-organizing fuzzy armax models. IEEE Transactions on Power Systems 13(1), 217–225 (1998)