Signal Decomposition via Distributed Optimization

Bennet Meyers and Stephen Boyd

Stanford University

SLAC AI Seminar, Menlo Park, April 23, 2021

Outline

Signal decomposition problem

Parameters and validation

Distributed solution method

PV data example

Signals

- we consider vector valued signals that can include missing values

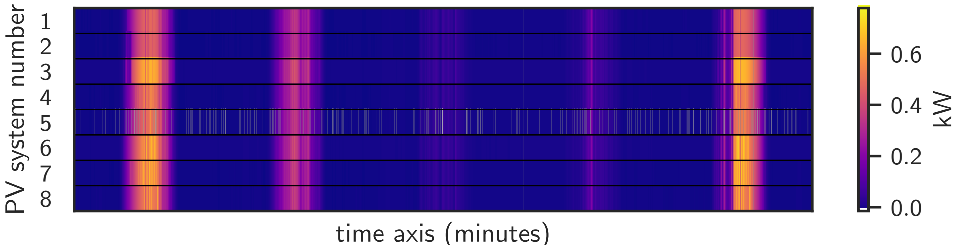
$$y_1, \ldots, y_T \in (\mathbf{R} \cup \{?\})^p, \qquad y = \begin{bmatrix} y_1 & \cdots & y_T \end{bmatrix}$$

- partition indices into known and unknown values

$$\mathcal{K} = \{(t,i) \mid y_{t,i} \in \mathbf{R}\}, \qquad \mathcal{U} = \{(t,i) \mid y_{t,i} = ?\}$$

- examples: financial data, energy production/load data, atmospheric and hydrospheric data

Example: data from 8 PV systems



- 5 days of 1-minute measurements from 8 PV systems in the same geographic area
- white pixels denote missing values

Decomposing a signal into components

- decomposition of signal y into K components

$$y \stackrel{\mathcal{K}}{=} x^1 + \cdots + x^K$$

- $\stackrel{\mathcal{K}}{=}$ means equal at known indexes $t, i \in \mathcal{K}$
- $x^k \in \mathbf{R}^{T \times p}$ for $k = 1, \ldots, K$ are components with no missing data
- component x^k comes from component class k
- example component classes
  - smooth
  - sparse
  - periodic
  - nonnegative
  - piecewise affine
  - Boolean with infrequent switching

Brief comment on notation

- drop the k when refering to a general variable $x \in \mathbf{R}^{T \times p}$
- $x_t \in \mathbf{R}^p$ for $t = 1, \ldots, T$ is a row vector
- $x_i \in \mathbf{R}^T$ for $i = 1, \ldots, p$ is a column vector
- $x_{t,i} \in \mathbf{R}$ is a single entry

Estimating missing values

- missing values in y can be estimated from decomposition as

$$\hat{y} \stackrel{\mathcal{U}}{=} x^1 + \cdots + x^K$$

- basis of validation and parameter tuning method described later on

Component classes

- component classes characterized by loss or implausibility functions

$$\phi_k : \mathbf{R}^{T\times p} \to \mathbf{R} \cup \{\infty\}, \quad k = 1, \ldots, K$$

- smaller $\phi_k(x)$ means more plausible x for class k
- infinite values encode constraints on components
- for statistical model of a component class, $\phi(x)$ is negative log-likelihood
- simple examples (with scale factor $\lambda > 0$):
  - mean-square small class: $\phi(x) = \frac{\lambda}{Tp} \sum_t \|x_t\|_2^2$
  - mean-square smooth class: $\phi(x) = \frac{\lambda}{(T-1)p} \sum_{t=1}^{T-1} \|x_{t+1} - x_t\|_2^2$
  - nonnegative class: $\phi(x) = \begin{cases} 0 & x_{t,i} \geq 0 \text{ for all } t, i \\ \infty & \text{otherwise} \end{cases}$

Signal decomposition problem

- we choose decomposition to minimize total loss or implausibility
- signal decomposition (SD) problem:

$$\begin{array}{ll} \text{minimize} & \phi_1(x^1) + \cdots + \phi_K(x^K) \\ \text{subject to} & y \overset{\mathcal{K}}{=} x^1 + \cdots + x^K \end{array}$$

- variables are components $x^1, \ldots, x^K$
- we refer to a solution as an optimal signal decomposition

Solving the signal decomposition problem

- if all ϕ^k are convex, SD problem is convex, and so can be efficiently solved
- otherwise, we settle for an approximate solution

- our method is based on alternating directions method of multipliers (ADMM)
  - a distributed method that handles the component classes separately
  - easy to define new component classes
  - solves SD problem when it's convex
  - approximately solves SD problem it's not convex

Example

$K = 3$ component classes:

- mean-square small, $\phi_1(x) = \frac{\lambda_1}{Tp} \sum_t \|x_t\|_2^2$
- mean-square smooth, $\phi(x) = \frac{\lambda_2}{(T-1)p} \sum_{t=1}^{T-1} \|x_{t+1} - x_t\|_2^2$
- Boolean with infrequent switching,

$$\phi(x) = \begin{cases} \frac{\lambda_3}{(T-1)p} \sum_i \sum_{t=1}^{T-1} |x_{t+1,i} - x_{t,i}| & x_{t,i} \in \{0,1\} \text{ for all } t, i \\ \infty & \text{otherwise} \end{cases}$$

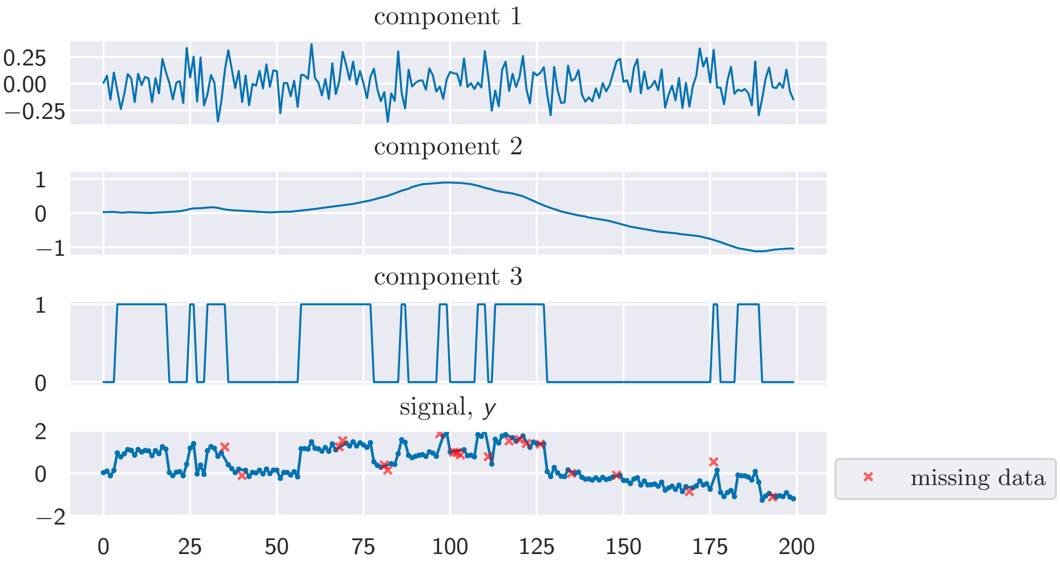
  i.e., rate of switching between values 0 and 1

- λ_i are positive weights; can take $\lambda_1 = 1$

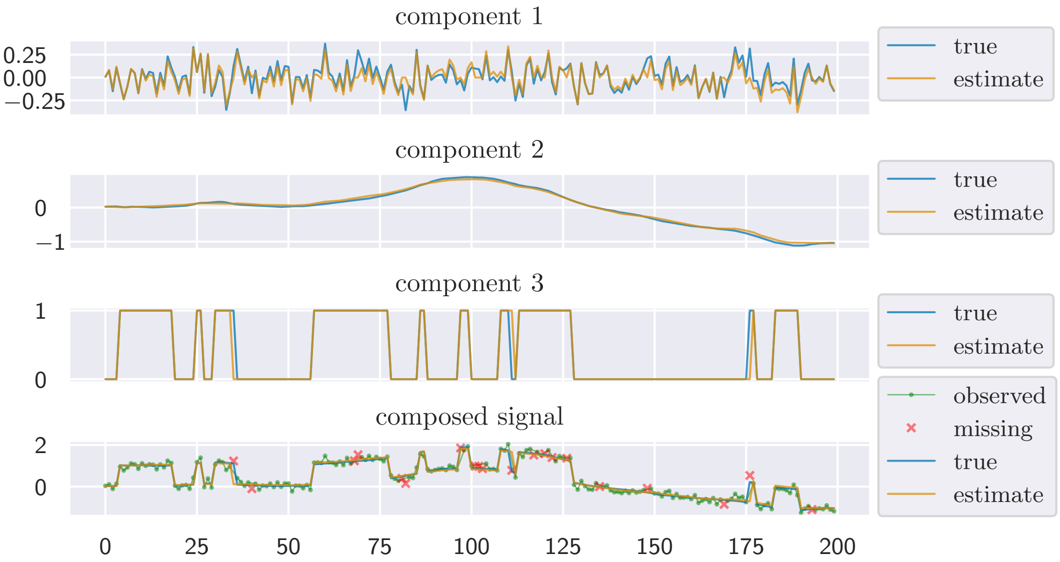
Synthetic data

- scalar signal, *i.e.*, $y \in (\mathbf{R} \cup \{\infty\})^{1 \times T}$
- generate y as $x^1_{\text{true}} + x^2_{\text{true}} + x^3_{\text{true}}$, then randomly make 10% of entries unknown
- entries of x^1_{true} are IID $\mathcal{N}(0, 0.15^2)$
- entries of x^2_{true} are white noise passed through a low-pass filter
- x^3_{true} is realization of Markov chain on $\{0, 1\}$ with probability 0.1 of transitions $0 \to 1$ or $1 \to 0$

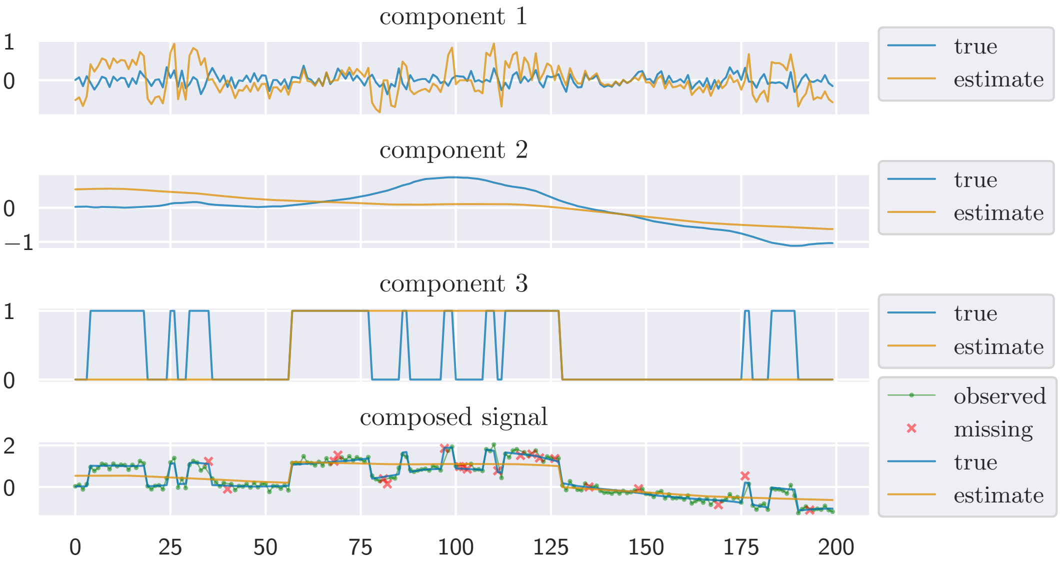
Synthetic data



Decomposition with $\lambda_2 = 25$, $\lambda_3 = 0.5$



Decomposition with $\lambda_2 = 500$, $\lambda_3 = 5$



Outline

Signal decomposition problem

Parameters and validation

Distributed solution method

PV data example

Component class parameters

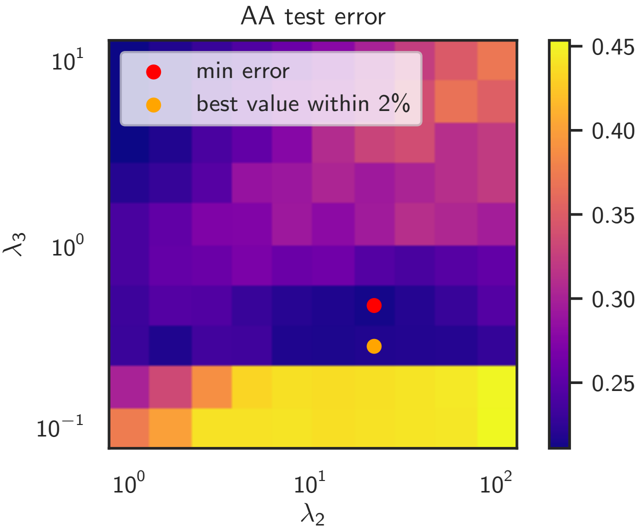
- class losses ϕ_k can have associated parameters, denoted $\phi_k(x^k; \theta_k), \ \theta_k \in \Theta_k$
- some common examples
  - weight or scaling parameters $\phi(x;\theta) = \theta\ell(x), \ \theta \in \Theta = \mathbf{R}_{++}$
    (often denoted with traditional symbol λ)
  - signal scaling parameters $\phi(x;\theta) = \phi^{\text{bool}}(x/\theta) \implies x \in \{0, \theta\}^{T \times p}$
  - constraint parameters $\phi(x;\theta) = \mathcal{I}(\theta_1 \leq x \leq \theta_2)$
  - basis parameters $\phi(x) = \mathcal{I}(x = \theta\alpha \text{ for some } \alpha \in \mathbf{R}^{d \times n})$
- different parameters lead to different decompositions

Validating a decomposition

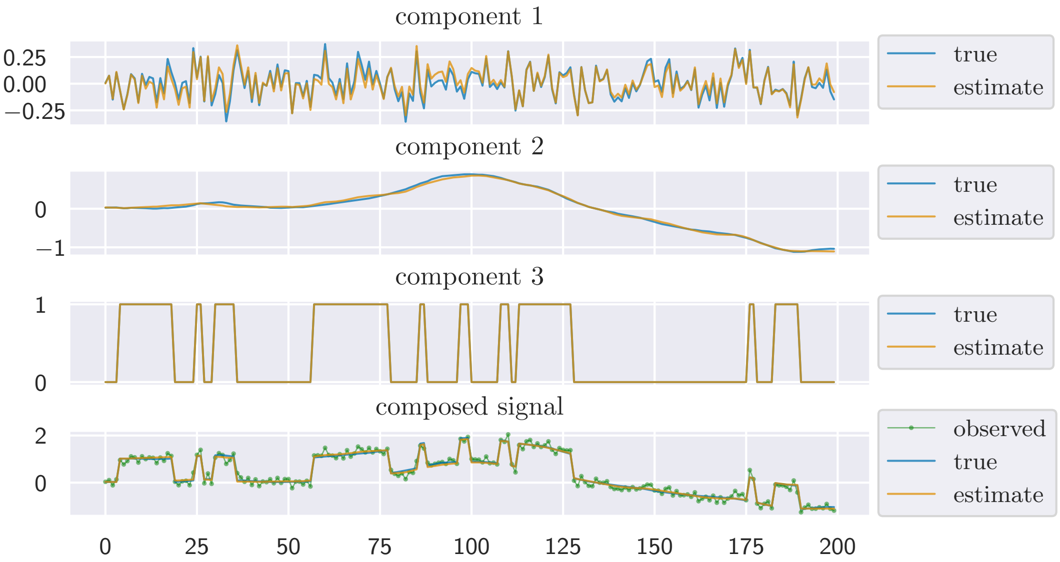
- randomly select test set $\mathcal{T} \subset \mathcal{K}$ and replace associated values in y with ?
- carry out decomposition using entries $\mathcal{K} \setminus \mathcal{T}$
- decomposition yields estimates $\hat{y}_{t,i}$ for $(t, i) \in \mathcal{T}$
- quantify residuals or errors $y_{t,i} - \hat{y}_{t,i}$, $(t, i) \in \mathcal{T}$, with some metric, e.g. RMS or AA (average absolute)
- for more stable validation, process can repeat, e.g. k-fold cross validation or bootstrap sampling
- can be used to choose component classes and class parameters

Example

- set $\lambda_1 = 1$, search parameter space for best λ_2 and λ_3
- randomly select 10% of the data point for test set
- decompose for each parameter value (10×10 grid)
- repeat 12 times and take average error



Final decomposition, $\lambda_2 = 21.5$, $\lambda_3 = 0.278$



Outline

Alternating direction direction of multipliers (ADMM)

- a method for solving convex optimization problems
- developed in 1970s, with roots in 1950s; modern treatment in Boyd et al. [2011]
- can be used as a heuristic for non-convex problems
- a distributed method, with different parts handled separately

SD via ADMM

- for iteration $j = 1, \ldots$

$$
\begin{aligned}
(x^k)^{j+1} &:= \mathbf{prox}_k\left((x^k)^j - u^j\right), \quad k = 1, \ldots, K \\
\hat{y}^{j+1} &:= \sum_{k=1}^{K} (x^k)^{j+1} \\
u_{t,i}^{j+1} &:= u_{t,i}^j + \frac{2}{K}(\hat{y}_{t,i}^{j+1} - y_{t,i}), \quad (t,i) \in \mathcal{K}
\end{aligned}
$$

- $\mathbf{prox}_k(v) = \mathrm{argmin}_x \left(\phi_k(x) + \frac{\rho}{2}\|x - v\|_F^2\right)$, proximal operator of ϕ_k
- $\rho > 0$ is an algorithm parameter
- $u_{t,i}^j$ are dual variables

Convergence and properties

- converges to (global) solution when all ϕ^k are convex, for any $\rho > 0$
- is a good heuristic in other cases, but choice of ρ can matter

- only need proximal operator for each component class
- first step can be carried out in parallel, for the k components
- each component handled separately; coordination is via dual variables $u^j_{t,i}$

Proximal operator

$$\mathbf{prox}_{\phi}(v) = \operatorname*{argmin}_{x} \left(\phi(x) + \frac{\rho}{2} \|x - v\|_F^2 \right)$$

- compromise between making $\phi(x)$ small and x near v
- when ϕ is an indicator function of a set $\mathcal{C}$, proximal operator is projection onto $\mathcal{C}$ (and doesn't depend on ρ)
- for many ϕ, proximal operator can be worked out analytically
- for others, can involve some computation

Examples

name	$\phi(x)$	$\mathbf{prox}_\phi(v)$
mean-square small	$\lambda \sum_t \lVert x_t \rVert_2^2$	$\frac{\rho}{2\lambda+\rho} v_{t,i}$
average-absolute small	$\lambda \sum_t \lVert x_t \rVert_1$	$\begin{cases} v_{t,i} - \lambda/\rho & v_{t,i} > \lambda/\rho \\ 0 & \lvert v_{t,i} \rvert < \lambda/\rho \\ v_{t,i} + \lambda/\rho & v_{t,i} < -\lambda/\rho \end{cases}$
mean-square small rth-order diff.	$\lambda \sum_i \lVert D_r x_i \rVert_2^2$	$\left(I + \frac{2\lambda}{\rho} D_r^T D_r \right)^{-1} v$
non-negative	$\mathcal{I}(x \geq 0)$	$(v)_+$
linear equality constraint	$\mathcal{I}(Ax = b)$	$v - A^T(AA^T)^{-1}(Av - b)$
Boolean set	$\mathcal{I}(x \in \{0,1\}^{T \times p})$	$\begin{cases} 0 & \lvert v_{t,i} \rvert \leq \lvert v_{t,i} - 1 \rvert \\ 1 & \text{otherwise} \end{cases}$

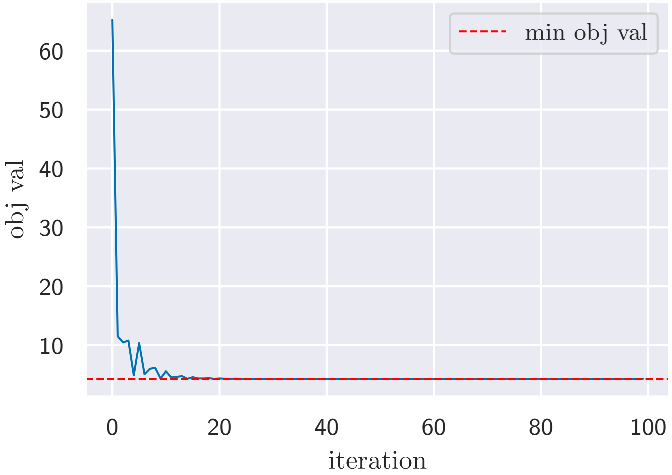
A less obvious example

- Boolean with infrequent switching

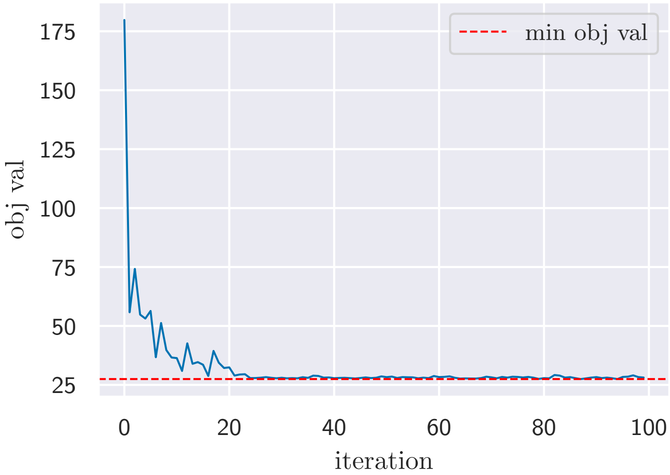
$$\phi(x) = \begin{cases} \lambda \sum_i \sum_{t=1}^{T-1} |x_{t+1,i} - x_{t,i}| & x_{t,i} \in \{0,1\} \text{ for all } t, i \\ \infty & \text{otherwise} \end{cases}$$

- proximal operator can be evaluated by solving a graph shortest path problem using dynamic programming, with cost $O(T)$ flops

Convergence example, convex case



Convergence example, non-convex case



Outline

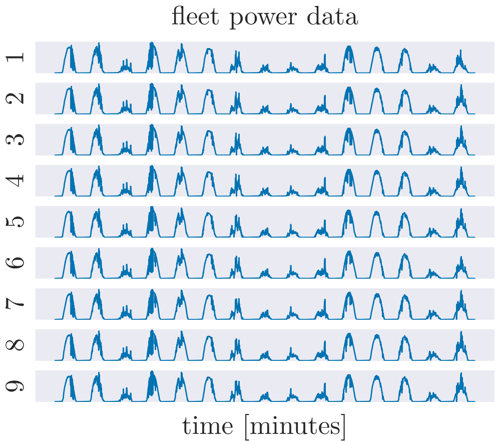
PV fleet outage detection

- recall data example of power signals from 8 PV systems (PV 'fleet') exposed to similar weather patterns
- want to automatically detect drops in system output that might be due to a failure of a PV module or string of modules
- standard industry approach
  - make a physical model of each system
  - obtain local measurements of irradiance and temperature
  - compare actual to predicted for each system

- let's try a purely data driven approach, using SD

The data set

- 15 days of 1-minute measurements from 9 systems
- T=21600, p=9
- artificially induce 'failures' in two systems
  - 25% loss of power output in system 6 during second-to-last day
  - 50% loss of power output in system 1 during final day



Data preprocessing

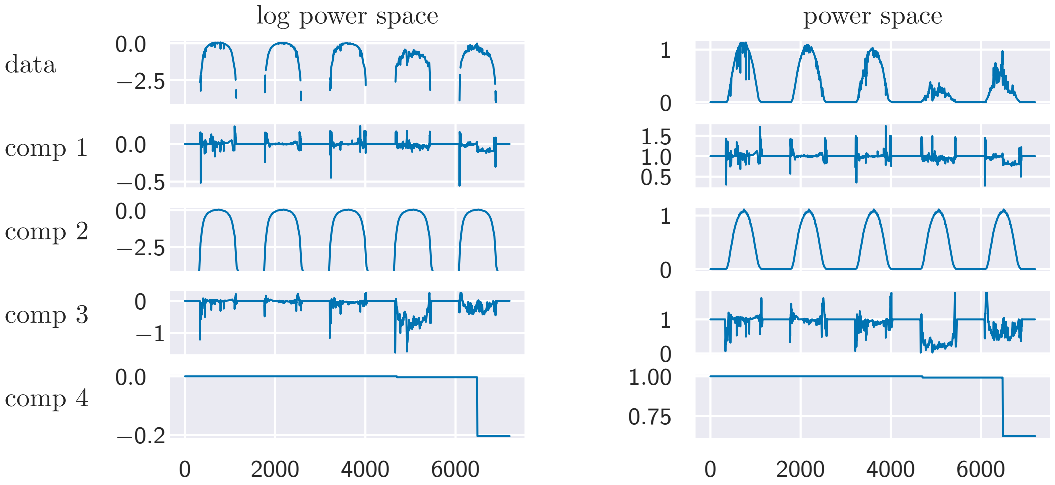
- scale each system data to about $[0, 1]$ (use 95th percentile for UB, not max)
- take $\log_{10}$ and set zero values to ?
- taking log gives a multiplicative component model, instead of additive

SD components

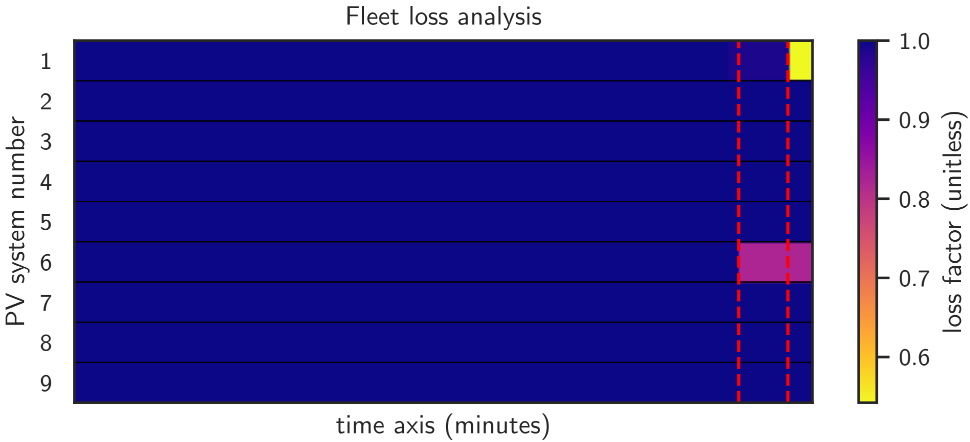
- residual: $\phi_1(x) = \lambda_1 \|x\|_F^2$
- common clear sky component: smooth, equal across systems, daily periodic
  - $\phi_2(x) = \lambda_2 \sum_{t=1}^{T-2} \|x_t - 2x_{t+1} + x_{t+2}\|_2^2$
  - $x_i - x_{i+1} = 0$, for $i = 1, \ldots, p-1$
  - $x_t - x_{t+1440} = 0$, for $t = 1, \ldots, T - 1440$
- common weather component: asymmetric distribution and equal across systems
  - $\phi_3(x) = \lambda_3 \sum_{t,i} 1/2\, |x_{t,i}| + (\tau - 1/2) x_{t,i}$
  - $x_{t,i} - x_{t,i+1} = 0$, for all t and for $i = 1, \ldots, p-1$
- outage detector: non-positive, mostly zero, and mostly constant
  - $\phi_4(x) = \lambda_4 \sum_i \sum_{t=1}^{T-1} |x_{t,i} - x_{t+1,i}| + \lambda_5 \sum_{t,i} (-x_{t,i})$
  - $x \preceq 0$
  - $x_1 = \mathbf{0}$ (first row is the zero vector)

- $T \times p \times K = 777{,}636$ variables to estimate!

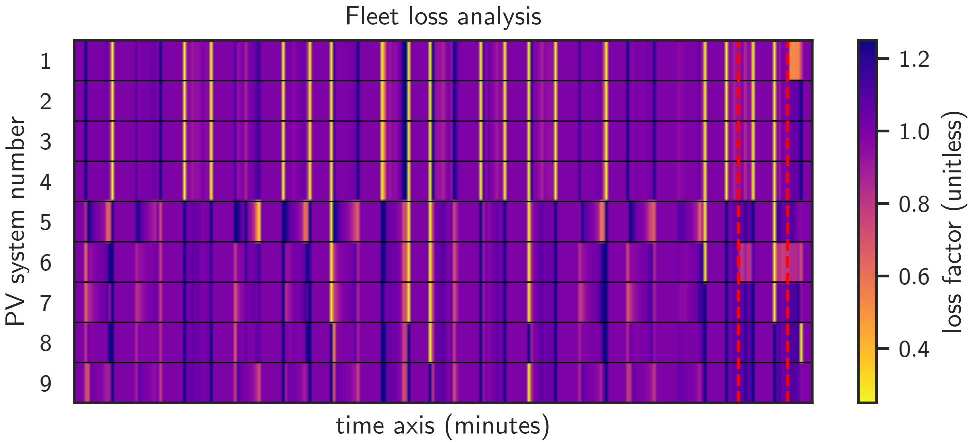
Results: decomposition of system 1 (last 5 days)



Results, outage detector component



Naive approach, compare each to average



Software

- developing Python implementation:
  `https://github.com/bmeyers/optimal-signal-demixing/`
- work in progress
- components defined as objects, proximal operators are attributes
- no requirement to understand ADMM or proximal operators to use!

Conclusions

signal decomposition via distributed optimization

- is interpretable
- provides a good way to describe prior knowledge about the signal
- is extensible
- is scalable to very large data sets
- does not require large training sets (or any labeled training data)

This material is based upon work supported by the U.S. Department of Energy's Office of Energy Efficiency and Renewable Energy (EERE) under the Solar Energy Technologies Office Award Number 34368.