Optimization and Engineering (2022) 23:397–419
https://doi.org/10.1007/s11081-020-09592-x

RESEARCH ARTICLE


# Eigen-stratified models

**Jonathan Tuck**[1] · **Stephen Boyd**[1]

Received: 2 February 2020 / Revised: 23 December 2020 / Accepted: 24 December 2020 /
Published online: 6 January 2021


**Abstract**
Stratified models depend in an arbitrary way on a selected categorical feature that takes $K$ values, and depend linearly on the other $n$ features. Laplacian regularization with respect to a graph on the feature values can greatly improve the performance of a stratified model, especially in the low-data regime. A significant issue with Laplacian-regularized stratified models is that the model is $K$ times the size of the base model, which can be quite large. We address this issue by formulating *eigen-stratified models*, which are stratified models with an additional constraint that the model parameters are linear combinations of some modest number $m$ of bottom eigenvectors of the graph Laplacian, i.e., those associated with the $m$ smallest eigenvalues. With eigen-stratified models, we only need to store the $m$ bottom eigenvectors and the corresponding coefficients as the stratified model parameters. This leads to a reduction, sometimes large, of model size when $m \leq n$ and $m \ll K$. In some cases, the additional regularization implicit in eigen-stratified models can improve out-of-sample performance over standard Laplacian regularized stratified models.

**Keywords** Optimization · Convex optimization · Data science · Machine learning

## 1 Introduction

Stratified models are models that depend in an arbitrary way on a selected categorical feature (or set of features) that takes $K$ values, and depend linearly on the other features (Tuck et al. 2019). For example in a date-stratified model we might have a different linear model for each day of the year, with $K = 365$. Laplacian regularization can be added to exploit some known relations among the categorical features, expressed as a graph. In our date-stratified example, Laplacian regularization encourages the models for adjacent dates to be close, including the January 1 and December 31 models. In this example, the underlying graph is a cycle with 365 vertices.

✉ Jonathan Tuck
jonathantuck@stanford.edu

[1] Stanford University, 350 Jane Stanford Way, Packard Rm. 243, Stanford, CA 94305, USA

Laplacian regularization can greatly improve the performance of a stratified model, especially in the low-data regime. In particular, it allows us to form a reasonable model even when we have no training data for some values of the categorical variable. The number of parameters in a Laplacian-regularized stratified model is $K$, the number of values of the categorical feature, times the size of the base model, which can be quite large when the catgorical values take many values. For example, a date-stratified model contains 365 times more coefficients than the base model. This is one of the challenges that we address in this paper.

Laplacian regularization encourages the model parameters to vary smoothly across the graph that encodes our prior information about the categorical values. If the model parameters vary smoothly across the graph, it is reasonable to assume they can be well approximated as linear combinations of a modest number of the eigenvectors of the associated graph Laplacian associated with the smallest eigenvalues. Our idea is simple: We impose the constraint that the model parameters are linear combinations of some number $m$ of the bottom eigenvectors of the graph Laplacian. We refer to such a model as an *eigen-stratified model*. The resulting eigen-stratified model uses only a factor $m$ parameters more than the base model, compared to a factor $K$ for a general stratified model. In addition to this savings in model size, insisting that the model parameters are linear combinations of the bottom $m$ eigenvectors acts as an additional useful regularization, that enforces smooth variation of the model parameters across the graph.

In our date-stratified example, the bottom eigenvector is constant, and the next ones occur in sine and cosine pairs, with periods one year, a half year, one-third of year, and so on. Using $m = 7$, say, requires that the model parameters are Fourier series with 7 terms (i.e., constant plus three harmonics). So here the eigen-stratified model is very natural.

In more complex cases, the eigen-stratified model is far less obvious. For example, the underlying graph can contain multiple edge weights, which are hyperparameters. In any but the simplest cases, we do not have analytical expressions for the eigenvalues and eigenvectors, but they are readily computed, even for very large graphs.

### 1.1 Related work

*Model approximations* It is quite common to approximate a larger model with a smaller, but only slightly less accurate model. In signal processing, discrete signals are transformed into a basis where they may be approximated by a linear combination of a small number of basis vectors, such as complex exponentials or cosines, in order to achieve significant size compression at the cost of signal degradation, which in many cases is minimal (Ahmed et al. 1974; Oppenheim and Schafer 2009). Likewise, eigen-stratified models can be interpreted as an approximate stratified model, whose model parameters are approximated as linear combinations of bottom eigenvectors of the stratified model's graph Laplacian.

*Categorical embeddings* Learning low-dimensional vector representations of discrete variables is consistently used as a method to handle categorical features.

Embeddings are a popular tool in fields such as natural language processing, to embed text as continuous vectors (Elman 1990; Mikolov et al. 2013; Guo and Berkhahn 2016). We can associate with each vertex the $m$ coefficients of the bottom Laplacian eigenvectors. This gives a Laplacian or spectral embedding of the features into $\mathbf{R}^m$.

*Spectral graph theory* The study of properties of graphs through their Laplacian eigen-decomposition is a long studied field in graph theory (Chung 1997; Cohen-Steiner et al. 2018). Three example applications include spectral clustering (Ng et al. 2002), which is a form of dimensionality reduction that uses the the eigen-decomposition of the graph Laplacian to cluster nodes in a graph; finding the fastest Mixing Markov process on a graph, whose convergence guarantees rely on the spectrum of the graph's Laplacian matrix (namely, the Fiedler eigenvalue of the Laplacian) (Boyd et al. 2004; Sun et al. 2006; Boyd et al. 2009); and graph coloring (Brooks 1941; Brélaz 1979), where the goal is to assign one of a set of colors to a graph node such that no two adjacent nodes share a color. Graph coloring is an NP-hard task in general, but ideas from spectral graph theory are naturally used as heuristics to sub-optimally solve this problem (Aspvall and Gilbert 1984).

*Laplacian regularization in large-scale optimization* There are many general methods to solve convex optimization problems with Laplacian regularization. Examples include the alternating direction method of multipliers (ADMM) (Tuck et al. 2019), majorization-minimization (MM) (Tuck et al. 2019), and Anderson accelerated Douglas-Rachford splitting (Fu et al. 2019b). ADMM and proximal algorithms in general have also been used in the past to fit large-scale network models, such as the network lasso and the logistic network lasso, which are stratified models that encourage closeness of parameters by their difference as measured by the $\ell_2$-norm, rather than by the $\ell_2$-norm squared (Hallac et al. 2015, 2017; Jung and Tran 2019; Tran et al. 2020). In addition, the idea of applying Laplacian approximations to large-scale optimization problems has been studied in the past, where one approximates the graph Laplacian by a linear combination of the eigenvectors to solve extremely large semidefinite programs in, *e.g.*, maximum variance unfolding (Weinberger et al. 2007).

### 1.2 Outline

In Sect. 2.1 we review stratified models, fixing our notation; in Sect. 2.2 we formally describe the eigen-stratified model fitting problem, and in Sect. 3, we give a distributed solution method. In Sect. 4 we give some simple numerical examples, carried out using an accompanying open-source implementation of our method.

## 2 Eigen-stratified models

In this section, we give a brief overview of stratified models; see (Tuck et al. 2019) for much more detail.

### 2.1 Stratified models

We fit a model to data records of the form $(z, x, y) \in \mathcal{Z} \times \mathcal{X} \times \mathcal{Y}$. Here $z \in \mathcal{Z}$ is the feature over which we stratify, $x \in \mathcal{X}$ is the other features, and $y \in \mathcal{Y}$ is the outcome, label, or dependent variable. The feature and label spaces $\mathcal{X}$ and $\mathcal{Y}$ are arbitrary data types; the stratified feature values $\mathcal{Z}$, however, must consist of only $K$ possible values, which we denote as $\mathcal{Z} = \{1, \ldots, K\}$.

A stratified model is built on top of a base model, which models pairs $(x, y)$ (or, when $x$ is absent, just $y$). The base model is parametrized by a parameter vector $\theta \in \Theta \subseteq \mathbf{R}^n$. In a stratified model, we use a different value of the parameter $\theta$ for each value of $z$. We denote these parameters as $\theta_1, \ldots, \theta_K$, where $\theta_k$ is the parameter value used when $z = k$. We let $\theta \in \mathbf{R}^{n \times K}$ denote the parameter values for the stratified model, where

$$\theta = [\theta_1 \cdots \theta_K] \in \mathbf{R}^{n \times K}.$$

(In (Tuck et al. 2019), the individual parameter vectors $\theta_k$ were stacked into one vector of dimension $nK$; here it will be more convenient to assemble them into a matrix.)

To choose the parameters $\theta_1, \ldots, \theta_K$, we minimize

$$\sum_{k=1}^{K} \left( \ell_k(\theta_k) + r(\theta_k) \right) + \mathcal{L}(\theta). \tag{1}$$

The first term is the sum of $K$ local objective functions, with the $k$th local objective function consisting of a local loss of the form

$$\ell_k(\theta) = \sum_{i : z_i = k} l(\theta, x_i, y_i), \tag{2}$$

with loss function $l : \Theta \times \mathcal{X} \times \mathcal{Y} \to \mathbf{R}$, and local regularizer $r : \Theta \to \mathbf{R} \cup \{\infty\}$. (Infinite values of the regularizer encode constraints on allowable into a matrix.) Examples of local regularization $r$ are $\| \cdot \|_2^2$ and $\| \cdot \|_1$. Choosing $\theta_k$ to minimize $\ell_k(\theta_k) + r(\theta_k)$ gives the regularized empirical risk minimization model parameters, based only on the data records that take the particular value of the stratification feature $z = k$.

The second term $\mathcal{L}(\theta)$ in (1) measures the non-smoothness of the model parameters over $z \in \mathcal{Z}$. Let $W \in \mathbf{R}^{K \times K}$ be a symmetric matrix with nonnegative entries. The associated *Laplacian regularization* or *Dirichlet energy* is the function $\mathcal{L} : \mathbf{R}^{n \times K} \to \mathbf{R}$ given by

$$\mathcal{L}(\theta) = \frac{1}{2} \sum_{i,j=1}^{K} W_{ij} \|\theta_i - \theta_j\|_2^2. \tag{3}$$

We can associate the Laplacian regularization with a graph with $K$ vertices, with an edge $(i, j)$ for each positive $W_{ij}$, with weight $W_{ij}$. We can express the Laplacian regularization as the positive semidefinite quadratic form

$$\mathcal{L}(\theta) = (1/2)\,\mathbf{Tr}(\theta L \theta^T),$$

where $L \in \mathbf{R}^{K\times K}$ is the (weighted) Laplacian matrix associated with the weighted graph, given by

$$L_{ij} = \begin{cases} -W_{ij} & i \neq j \\ \sum_{k=1}^{K} W_{ik} & i = j \end{cases}, \qquad i,j = 1,\ldots,K.$$

We note that the Laplacian regularization $\mathcal{L}(\theta)$ is separable in the rows of $\theta$.

We refer to the model obtained by solving (1) as a *standard stratified model*. When the loss function $\ell$ and local regularization function $r$ are convex, the objective in (1) is convex, which implies that a global solution can be found efficiently (Boyd and Vandenberghe 2004). When this assumption does not hold, heuristic methods can be used to approximately solve (1).

### 2.2 Eigen-stratified models

The eigen-decomposition of the Laplacian matrix $L$ is

$$L = Q\Lambda Q^T,$$

where $\Lambda \in \mathbf{R}^{K\times K}$, a diagonal matrix consisting of the eigenvalues of $L$, is of the form $\Lambda = \mathbf{diag}(\lambda_1,\ldots,\lambda_K)$ with $0 = \lambda_1 \leq \cdots \leq \lambda_K$, and $Q = (q_1,\ldots,q_K) \in \mathbf{R}^{K\times K}$ is a matrix of orthonormal eigenvectors of $L$. Since $L\mathbf{1} = 0$, where $\mathbf{1}$ is the vector with all entries one, we have $\lambda_1 = 0$, and $q_1 = \mathbf{1}/\sqrt{K}$ (Spielman 2010). (When the graph is connected, $q_1$ is unique, and $\lambda_2 > 0$.) In many cases, the eigenvectors and eigenvalues of a graph Laplacian matrix can be computed analytically; in "Appendix 1", we mention a few of these common graphs and give their eigenvectors and eigenvalues.

For $m \leq K$, we refer to $\lambda_1,\ldots,\lambda_m$ as the bottom $m$ eigenvalues, and $q_1,\ldots,q_m$ as the bottom $m$ eigenvectors. They are an orthonormal basis of the subspace of $\mathbf{R}^K$ that is smoothest, i.e., minimizes $\mathbf{Tr}\,\tilde{Q}^T L\tilde{Q}$, where $\tilde{Q} = [q_1\ \cdots\ q_m] \in \mathbf{R}^{K\times m}$, subject to $\tilde{Q}^T\tilde{Q} = I_m$. Roughly speaking, functions on $\mathcal{Z}$ that are smooth should be well approximated by a linear combination of the bottom $m$ eigenvectors (for suitable $m$).

Assuming that $\theta$ has low Dirichlet energy, i.e., a small Laplacian regularization term, we conclude that its rows are well approximated by a linear combination of the bottom $m$ eigenvectors. This motivates us to impose a further constraint on the rows of $\theta$: They must be linear combinations of the bottom $m$ eigenvectors of $L$. This can be expressed as

$$\theta = Z\tilde{Q}^T, \tag{4}$$

where $Z \in \mathbf{R}^{n\times m}$ are the (factorized) model parameters and $\tilde{Q} \in \mathbf{R}^{K\times m}$ are the bottom $m$ eigenvectors of $L$.

Adding the constraint (4) to the Laplacian regularized stratified model fitting problem (1), we obtain the problem

$$\begin{array}{ll} \text{minimize} & \sum_{k=1}^{K}(\ell_k(\theta_k) + r(\theta_k)) + \mathcal{L}(\theta) \\ \text{subject to} & \theta = Z\tilde{Q}^T, \end{array} \tag{5}$$

where now both $\theta$ and $Z$ are optimization variables, coupled by the equality constraint. We can express the Laplacian regularization term in (5) directly in terms of $Z$ as

$$\mathcal{L}(\theta) = (1/2)\,\mathbf{Tr}(\theta L\theta^T) = (1/2)\,\mathbf{Tr}(Z\tilde{Q}^T L\tilde{Q}Z^T) = (1/2)\|Z\Lambda_m^{1/2}\|^2,$$

where $\Lambda_m = \mathbf{diag}(\lambda_1, \ldots, \lambda_m)$ is the diagonal matrix of the eigenvalues corresponding to the bottom $m$ eigenvectors of $L$. We refer to the model obtained by solving (5) as an *eigen-stratified model*.

We note that the sum of empirical losses and local regularization are clearly separable in the columns of $\theta$, and the Laplacian regularization is a separable function in the rows of $Z$; we utilize this fact in Sect. 3.

*Comparison to standard stratified models* With standard stratified models, we allow arbitrary variations of the model parameter $\theta$ across the graph. With eigen-stratified models, we sharply limit how $\theta$ varies across the graph by constraining $\theta$ to be a linear combination of the $m$ bottom eigenvectors of the graph.

*Storage* The standard stratified model requires us to store $Kn$ model parameters. An eigen-stratified model, on the other hand, stores $m(K + n)$ variables in the eigenvectors $\tilde{Q}$ and the factorized model parameters $Z$. This implies that when $m \leq n$ and $m \ll K$, the storage savings is significant.

*Convexity* If the local losses $\ell_k$ and local regularizers $r$ are convex, then (5) is a convex problem, which is readily solved globally in an efficient manner. It is easily formulated using domain specific languages for convex optimization (Boyd and Vandenberghe 2004; Grant et al. 2006; Grant and Boyd 2014; Diamond and Boyd 2016; Fu et al. 2019a). If any of the $\ell_k$ or $r$ are nonconvex, it is a hard problem to solve (5) globally. In this case, our method (described in Sect. 3) will provide a good heuristic approximate solution.

*The two extremes* For a given set of edge weights, we analyze the behavior of the eigen-stratified model as we vary $m$. When we take $m = 1$ and the graph is connected, we recover the common model (i.e., a stratified model with all $\theta_i$ equal). We can see this by noting that when $m = 1$ and the graph is connected, the constraint in (5) becomes a consensus constraint (recall that the bottom eigenvector of a Laplacian matrix is a scalar multiple of $\mathbf{1}$). If we take $m = K$, the eigen-stratified model is the same as the standard stratified model.

## 3 Distributed solution method

In this section we describe a distributed algorithm for solving the fitting problem (5). To derive the algorithm, we first express (5) as

$$\begin{array}{ll} \text{minimize} & \sum_{k=1}^{K}(\ell_k(\theta_k) + r(\tilde{\theta}_k)) + (1/2)\|Z\Lambda_m^{1/2}\|^2 \\ \text{subject to} & \theta = Z\tilde{Q}^T \quad \theta = \tilde{\theta}, \end{array} \tag{6}$$

where we have introduced an additional optimization variable $\tilde{\theta} \in \mathbf{R}^{n\times K}$.

The augmented Lagrangian $L_\rho$ of (6) has the form

$$L_\rho(\theta, \tilde{\theta}, Z, u, \tilde{u}) = \sum_{k=1}^{K}(\ell_k(\theta_k) + r(\tilde{\theta}_k)) + (1/2)\|Z\Lambda_m^{1/2}\|^2 + (1/2\rho)\|\theta - \tilde{\theta} + u\|_2^2 + (1/2\rho)\|\tilde{\theta} - Z\tilde{Q}^T + \tilde{u}\|_2^2,$$

where $u \in \mathbf{R}^{n\times K}$ and $\tilde{u} \in \mathbf{R}^{n\times K}$ are the (scaled) dual variables associated with the two constraints in (6), respectively, and $\rho > 0$ is the penalty parameter. The ADMM algorithm (in scaled dual form) for the splitting $(\theta, Z)$ and $\tilde{\theta}$ consists of the iterations

$$\theta^{i+1}, Z^{i+1} := \operatorname*{argmin}_{\theta,Z} L_\rho(\theta, \tilde{\theta}^{i+1}, Z, u^i, \tilde{u}^i) \tag{7}$$

$$\tilde{\theta}^{i+1} := \operatorname*{argmin}_{\tilde{\theta}} L_\rho(\theta^{i+1}, \tilde{\theta}, Z^{i+1}, u^i, \tilde{u}^i) \tag{8}$$

$$u^{i+1} := u^i + \theta^{i+1} - \tilde{\theta}^{i+1} \tag{9}$$

$$\tilde{u}^{i+1} := \tilde{u}^i + \tilde{\theta}^{i+1} - Z^{i+1}\tilde{Q}^T. \tag{10}$$

If the $\ell_k$ and $r$ are convex, the iterates $\theta^i$, $\tilde{\theta}^i$ are guaranteed to converge to each other and $\theta^i$, $\tilde{\theta}^i$, and $Z^i$ are guaranteed to converge to a primal optimal point of (6) (Boyd et al. 2011).

This algorithm can be greatly simplified (and parallelized) by making use of a few observations. Our first observation is that the first step in ADMM (7) can be expressed as

$$\theta_k^{i+1} = \mathbf{prox}_{\rho l_k}(\tilde{\theta}_k^i - u_k^i), \quad k = 1, \ldots, K,$$

where $\mathbf{prox}_g : \mathbf{R}^n \to \mathbf{R}^n$ is the proximal operator of the function $g$ (Parikh and Boyd 2014), and

$$Z^{i+1} = (1/\rho)(\tilde{u}^i + \tilde{\theta}^i)\tilde{Q}(\Lambda_m + (1/\rho)I)^{-1}.$$

Note that the updates for $\theta^{i+1}$ and $Z^{i+1}$ can be carried out in parallel, since they do not depend on each other. Also, we can compute $\theta_1^{i+1}, \ldots, \theta_K^{i+1}$ in parallel.

Our second observation is that the second step in ADMM (8) can be expressed as

$$\tilde{\theta}_k^{i+1} = \mathbf{prox}_{\rho r}(Z^{i+1}\tilde{Q}^T - \tilde{u}^i), \quad k = 1, \ldots, K,$$

Similarly, we can compute $\tilde{\theta}_1^{i+1}, \ldots, \tilde{\theta}_K^{i+1}$ in parallel.

Combining these observations leads to Algorithm 3.1.

**Algorithm 3.1** *Distributed method for fitting eigen-stratified models.*

**given** Loss functions $\ell_1, \ldots, \ell_K$, local regularization function $r$, penalty parameter $\rho > 0$, $m$ bottom eigenvectors of the graph Laplacian matrix $\tilde{Q} \in \mathbf{R}^{K \times m}$, and diagonal matrix with corresponding $m$ bottom eigenvalues $\Lambda_m \in \mathbf{R}^{K \times K}$.
*Initialize.* $\tilde{\theta}^0 = u^0 = \tilde{u}^0 = 0$.
**repeat**
  **in parallel**
    *Evaluate proximal operator of* $\ell_k$. $\theta_k^{i+1} = \mathbf{prox}_{\rho \ell_k}(\tilde{\theta}_k^i - u_k^i), \quad k = 1, \ldots, K$
  *Update* $Z$. $Z^{i+1} = (1/\rho)(\tilde{u}^i + \tilde{\theta}^i)\tilde{Q}(\Lambda_m + (1/\rho)I)^{-1}$
  **in parallel**
    *Evaluate proximal operator of* $r$. $\tilde{\theta}_k^{i+1} = \mathbf{prox}_{\rho r}(Z^{i+1}\tilde{Q}^T - \tilde{u}^i), \quad k = 1, \ldots, K$
  *Update the dual variables.* $u^{i+1} := u^i + \theta^{i+1} - \tilde{\theta}^{i+1}; \quad \tilde{u}^{i+1} := \tilde{u}^i + \tilde{\theta}^{i+1} - Z^{i+1}\tilde{Q}^T$
**until convergence**

*Complexity* Generally, the dominant cost of this algorithm depends on the complexity of computing a single proximal operator of $l_k$ or $r$. Otherwise, the dominant costs are in multiplying a dense $n \times K$ matrix, a dense $K \times m$ matrix, and a diagonal $K \times K$ matrix together.

## 4 Examples

In this section, we illustrate the efficacy of the proposed method on two simple and relatively small examples.

*Software implementation* An implementation of our method for fitting an eigen-stratified model is given as an extension of the stratified model fitting implementation in (Tuck et al. 2019), available at www.github.com/cvxgrp/strat_models (along with the accompanying examples). To fit an eigen-stratified model, one may invoke

$$\texttt{model.fit(data, num\_eigen} = \texttt{m)}.$$

Here, `data` are the problem data (i.e., $(z, x, y)$ or $(z, y)$) and `num_eigen` is the number of bottom eigenvectors to use in the eigen-stratified model (i.e., $m$); if `num_eigen` is `None`, a standard Laplacian-regularized stratified model is fit.

### 4.1 Cardiovascular disease prediction

We consider the problem of predicting whether a patient has cardiovascular disease, given their sex, age, and other medical features.

*Dataset* We use data describing approximately 70000 patients across the world (Kaggle 2019). The dataset is comprised of males and females between the ages of 39 and 65 (inclusive), with approximately 50% of the patients diagnosed with cardiovascular disease.

There are 9 raw medical features in this dataset, which include: height, weight, systolic blood pressure (a categorical feature with values “below average", “average", and “above average"), diastolic blood pressure (a categorical feature with values “below

average", "average", and "above average"), cholesterol, glucose levels, whether or not the patient smokes, whether or not the patient drinks alcohol, and whether or not the patient undergoes regular physical activity. We randomly partition the data into a training set consisting of 5% of the records, a validation set containing 5% of the records, and a test set containing the remaining 90% of the records. We choose extremely small training and validation sets to illustrate the efficacy of stratified models in low-data regimes.

*Data records* We performed basic feature engineering on the raw medical features to derive a feature vector $x \in \mathbf{R}^{14}$ (i.e., $n = 14$), namely scalarization and converting the systolic blood pressure and diastolic blood pressure basic categorical features into multiple features via one-hot encoding, and adding a constant feature. The outcomes $y \in \{0, 1\}$ denote whether or not the patient has contracted cardiovascular disease, with $y = 1$ meaning the patient has cardiovascular disease. The stratification feature $z$ is a tuple of the patient's sex and age; *e.g.*, $z = (\text{Male}, 47)$ corresponds to a 47 year old male. The number of stratification feature values is thus $K = 2 \cdot 27 = 54$.

*Data model* We model the conditional probability of contracting cardiovascular disease given the features using a logistic regression base model (with intercept). We use logistic loss and sum of squares regularization, i.e., $r = (1/2)\|\cdot\|_2^2$, with associated hyper-parameter $\gamma_{\text{local}}$.

*Regularization graph* We take the Cartesian product of two regularization graphs:

- *Sex.* The regularization graph is a path graph that has one edge between male and female, with edge weight $\gamma_{\text{sex}}$.
- *Age.* The regularization graph is a path graph between ages, with edge weight $\gamma_{\text{age}}$.

Figure 1 illustrates the structure of this regularization graph.

From "Appendix 1", the eigenvectors of the regularization graph's Laplacian, $q_{i,j}$ for $i = 1, 2$ and $j = 1, \ldots, 27$, are given in closed-form as

$$q_{i,j} = \tilde{q}_{i,j}/\|\tilde{q}_{i,j}\|_2,$$
$$\tilde{q}_{i,j} = \cos((\pi/2)(i-1)(v-1/2)) \otimes \cos((\pi/27)(j-1)(v-1/2)) \quad i = 0, 1, \quad j = 0, \ldots, 26,$$

where $\otimes$ denotes a Kronecker product, $v = (0, \ldots, K-1)$, and $\cos(\cdot)$ is applied elementwise. (It is convenient for the eigenvectors to be indexed by $i$ and $j$, with $i$ corresponding to sex and $j$ corresponding to age.)

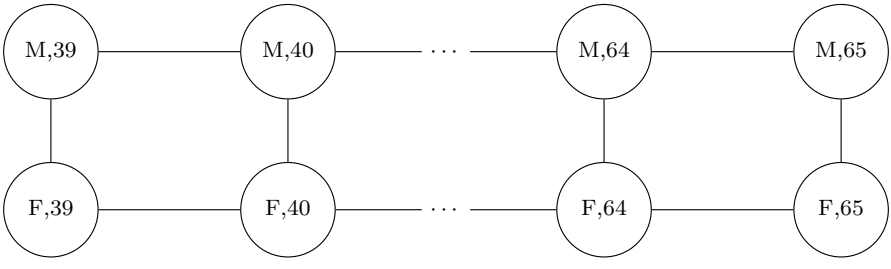


**Fig. 1** Regularization graph for Sect. 4.1

Figure 2 plots 8 of the 54 eigenvectors of the sex/age regularization graph Laplacian, with the particular sex/age edge weights $(\gamma_{\text{sex}}, \gamma_{\text{age}}) = (15, 175)$, sorted in increasing order corresponding to the bottom 8 eigenvalues of the Laplacian.

*Results* For each of the fitting methods (the separate, common, standard stratified and eigen-stratified models), we ran a hyper-parameter search over a grid of hyper-parameters and selected hyper-parameters that performed well over the validation set. For $\gamma_{\text{local}}$, we looked at a grid of values between 0.001 and 100. For $\gamma_{\text{sex}}$ and $\gamma_{\text{age}}$, we looked at a grid of values between 0.1 and 1000. For the separate model, we used $\gamma_{\text{local}} = 35$, and for the common model, we used $\gamma_{\text{local}} = 5$. (Recall that the separate model is a stratified model with all edge weights zero, and a common model is a stratified model with all edge weights $+\infty$ (Tuck et al. 2019).) For the standard

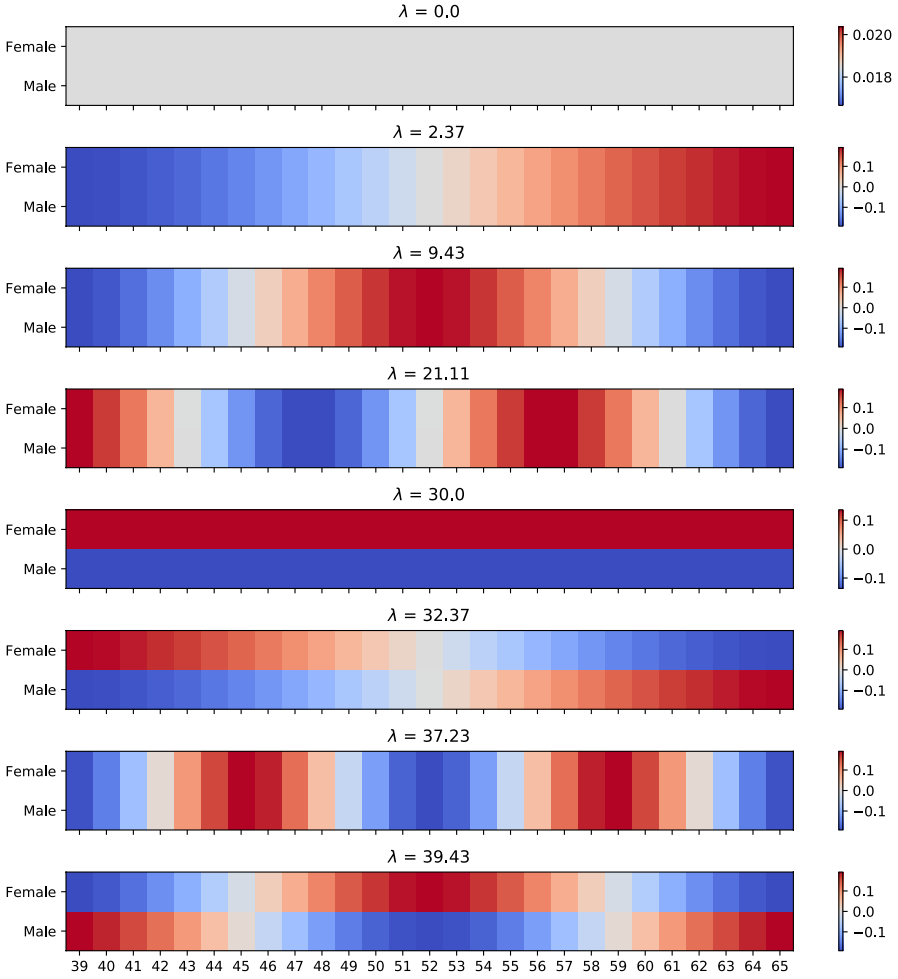


**Fig. 2** Heatmaps of the eigenvectors of the sex/age regularization graph Laplacian corresponding to the bottom 8 eigenvalues of the Laplacian

stratified model, we used $\gamma_{\text{local}} = 0.01$, $\gamma_{\text{sex}} = 125$ and $\gamma_{\text{age}} = 150$. For the eigen-stratified model, we used $\gamma_{\text{local}} = 2.5$, $\gamma_{\text{sex}} = 15$ and $\gamma_{\text{age}} = 175$, and $m = 5$. Table 1 shows the average negative log likelihood (ANLL) over the training, validation, and test datasets for the separate, common, standard stratified and eigen-stratified models.

We see that this test ANLL attains a minimum when only 5 bottom eigenvectors are used for the eigen-stratified model. This minimum test ANLL of the eigen-stratified model is competitive with (in fact, slightly smaller than) the test ANLL of the standard stratified model.

In the eigen-stratified model with $m = 5$, the model parameters are linear combinations of 5 bottom eigenvectors. There are $nK = 14 \cdot 54 = 756$ parameters in the standard stratified model to store, whereas the eigen-stratified model with minimum test ANLL stores $m(n + K) = 5 \cdot (14 + 54) = 340$ values, or approximately 45% as many parameters. So there is some storage efficiency gain even in this very simple example.

### 4.2 Weather distribution modeling

We consider the problem of modeling the distribution of weather temperature as a function of week of year and hour of day.

*Data records and dataset* We use temperature measurements from the city of Atlanta, Georgia for all of 2013 and 2014, sampled every hour (for a total of approximately 17500 measurements). The temperature is in Celsius; we round the temperatures to the nearest integer. There are $n = 43$ unique temperatures, ranging from -9 to 33 Celsius. Each data record includes the temperature, as well as the week of the year and the hour of the day (which will be the stratification features). The number of stratification features is $K = 52 \cdot 24 = 1248$.

We partition the dataset into three separate sets; a training set consisting of 30% of the data, a validation set consisting of 35% of the data, and a held-out test set consisting of the remaining 35% of the data. The model is trained on approximately 4.2 samples per stratification feature.

*Data model* We model the distribution of temperature in Atlanta at each week and hour using a non-parametric discrete distribution (Tuck et al. 2019). Our local regularizer is a sum of two regularizers: a sum of squares regularizer and a scaled sum of squares regularizer on the difference between adjacent parameters, i.e., $r(\theta) = \gamma_1 r_1(\theta) + \gamma_2 r_2(\theta)$, with $r_1(\theta) = (1/2)\|\theta\|_2^2$ and $r_2(\theta) = (1/2)\sum_{i=1}^{n}(\theta_{i+1} - \theta_i)^2$; the associated hyper-parameters with each are $\gamma_1$ and $\gamma_2$. The distribution $p_z$ at each node $z$ is calculated as

**Table 1** Results for Sect. 4.1

| Model | Train ANLL | Validation ANLL | Test ANLL |
|---|---|---|---|
| Separate | 0.607 | 0.656 | 0.658 |
| Common | 0.610 | 0.597 | 0.615 |
| Standard stratified | 0.572 | 0.567 | 0.597 |
| Eigen-stratified | 0.581 | 0.563 | 0.596 |

$$p_z = \frac{\exp(\theta_z)}{\sum_{i=1}^{n} \exp(\theta_z)_i},$$

where $\exp(\cdot)$ is evaluated elementwise.

*Regularization graph* We take the Cartesian product of two regularization graphs:

- *Week of year*. The regularization graph is a cycle graph with 52 nodes (one for each week of the year) with edge weights $\gamma_{\text{week}}$.
- *Hour of day*. The regularization graph is a cycle graph with 24 nodes (one for each hour of the day) with edge weights $\gamma_{\text{hr}}$.

The Cartesian product of these two graphs is a torus, illustrated in Fig. 3.

This graph has $K = 1248$ eigenvectors. The eigenvectors of this graph are given by

$$\begin{aligned}
&\tilde{s}_{i,j}/\|\tilde{s}_{i,j}\|_2 \\
&\tilde{u}_{i,j}/\|\tilde{u}_{i,j}\|_2 \\
&\tilde{v}_{i,j}/\|\tilde{v}_{i,j}\|_2 \\
&\tilde{w}_{i,j}/\|\tilde{w}_{i,j}\|_2,
\end{aligned}$$

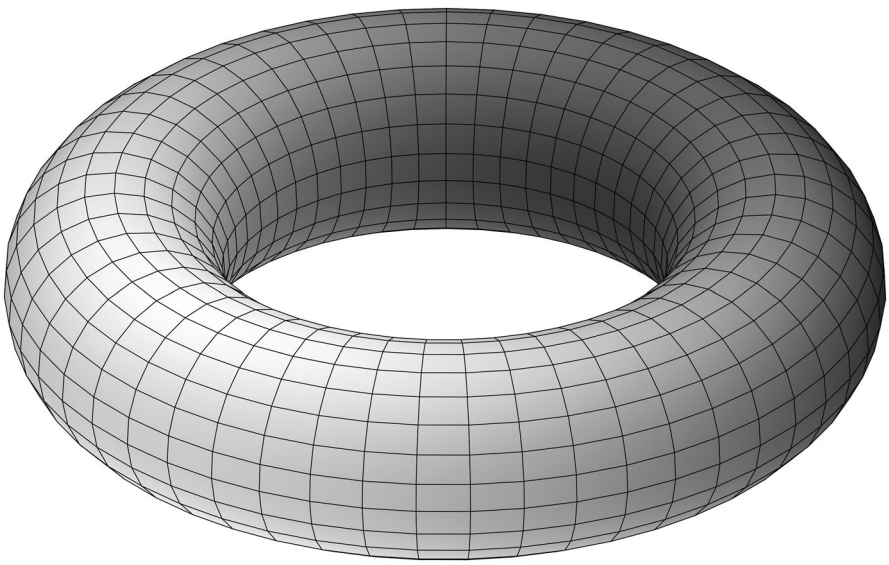

**Fig. 3** Regularization graph for Sect. 4.2. Each node corresponds to a week of year and hour of day, where the toroidal direction corresponds to increasing week of year, and the poloidal direction corresponds to increasing hour of day

where

$$
\begin{aligned}
s_{i,j} &= \cos(\pi(i-1)u/26) \otimes \sin(\pi(j-1)v/12), \\
u_{i,j} &= \cos(\pi(i-1)u/26) \otimes \cos(\pi(j-1)v/12), \\
v_{i,j} &= \sin(\pi(i-1)u/26) \otimes \sin(\pi(j-1)v/12), \\
w_{i,j} &= \sin(\pi(i-1)u/26) \otimes \cos(\pi(j-1)v/12),
\end{aligned}
$$

for $i = 1, \ldots, 26$ and $j = 1, \ldots, 12$, $u = (0, \ldots, 51)$, $v = (0, \ldots, 23)$, and $\cos(\cdot)$ and $\sin(\cdot)$ are applied elementwise. Figures 4 and 5 plots the bottom 10 eigenvectors of the week/hour regularization graph Laplacian, with the particular week/hour edge weights $(\gamma_{\text{week}}, \gamma_{\text{hr}}) = (.45, .55)$, sorted in increasing order corresponding to the bottom 10 eigenvalues of the Laplacian.

*Results* For each of the fitting methods, we ran a hyper-parameter search over a grid of hyper-parameters and selected hyper-parameters that yielded the smallest ANLL over the validation set. For $\gamma_1$ and $\gamma_2$, we looked at a grid of values between 0.001 and 1. For $\gamma_{\text{week}}$ and $\gamma_{\text{hr}}$, we looked at a grid of values between 0.1 and 1. For the separate model, we used $\gamma_1 = 0.75$ and $\gamma_2 = 0.3$, and for the common model, we used $\gamma_1 = 0.65$ and $\gamma_2 = 0.55$. For the standard stratified model, we used $\gamma_1 = 0.05$, $\gamma_2 = 0.05$, $\gamma_{\text{week}} = 0.6$ and $\gamma_{\text{hr}} = 0.5$. For the eigen-stratified model, we used $\gamma_1 = 0.01$, $\gamma_2 = 0.001$, $\gamma_{\text{week}} = 0.45$ and $\gamma_{\text{hr}} = 0.55$, and $m = 90$ (roughly 7% of the $52 \cdot 24 = 1248$ eigenvectors). We compare the ANLLs over the training, validation, and test datasets for the separate, common, standard stratified and eigen-stratified models in Table 2.

The validation and held-out test ANLLs of the eigen-stratified model were smaller than the respective ANLLs of every other model in Table 2, including those of the standard stratified model.

In Fig. 6 we plot the cumulative distribution functions (CDFs) of temperature for week 1, hour 1; week 28, hour 12; and week 51, hour 21 (which have 2, 3, and 2 empirical measurements in the training dataset, respectively), for the eigen-stratified model, and for the test empirical data.

In Fig. 7, we plot heatmaps of the expected value and standard deviation of the distributions given by the eigen-stratified model. The statistics vary smoothly as hours of day and weeks of year vary.

**Acknowledgements** We gratefully acknowledge discussions with Shane Barratt and Peter Stoica, who provided us with useful suggestions that improved the paper. J. Tuck is supported by the Stanford Graduate Fellowship.

**Funding** J. Tuck is supported by the Stanford Graduate Fellowship.

**Availability of data and material** All data is made available at www.github.com/cvxgrp/strat_models.

**Conflicts of interest** The authors declare that a possible conflict of interest is that S. Boyd is an author of this paper and an editor of this journal.

**Code availability** All code is made available at www.github.com/cvxgrp/strat_models.

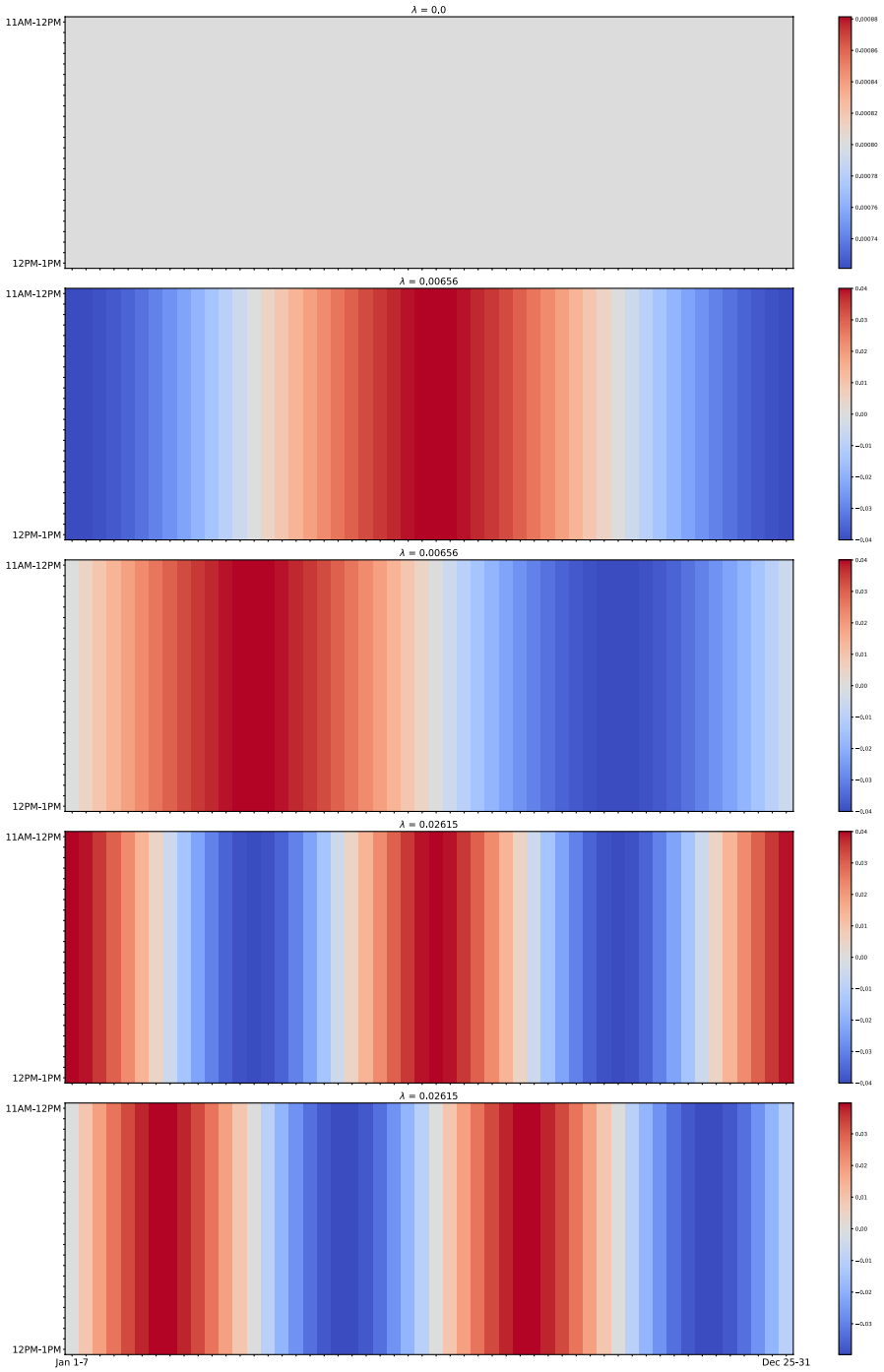


**Fig. 4** Heatmaps of the eigenvectors of the week/hour regularization graph Laplacian corresponding to $\lambda_1, \ldots, \lambda_5$

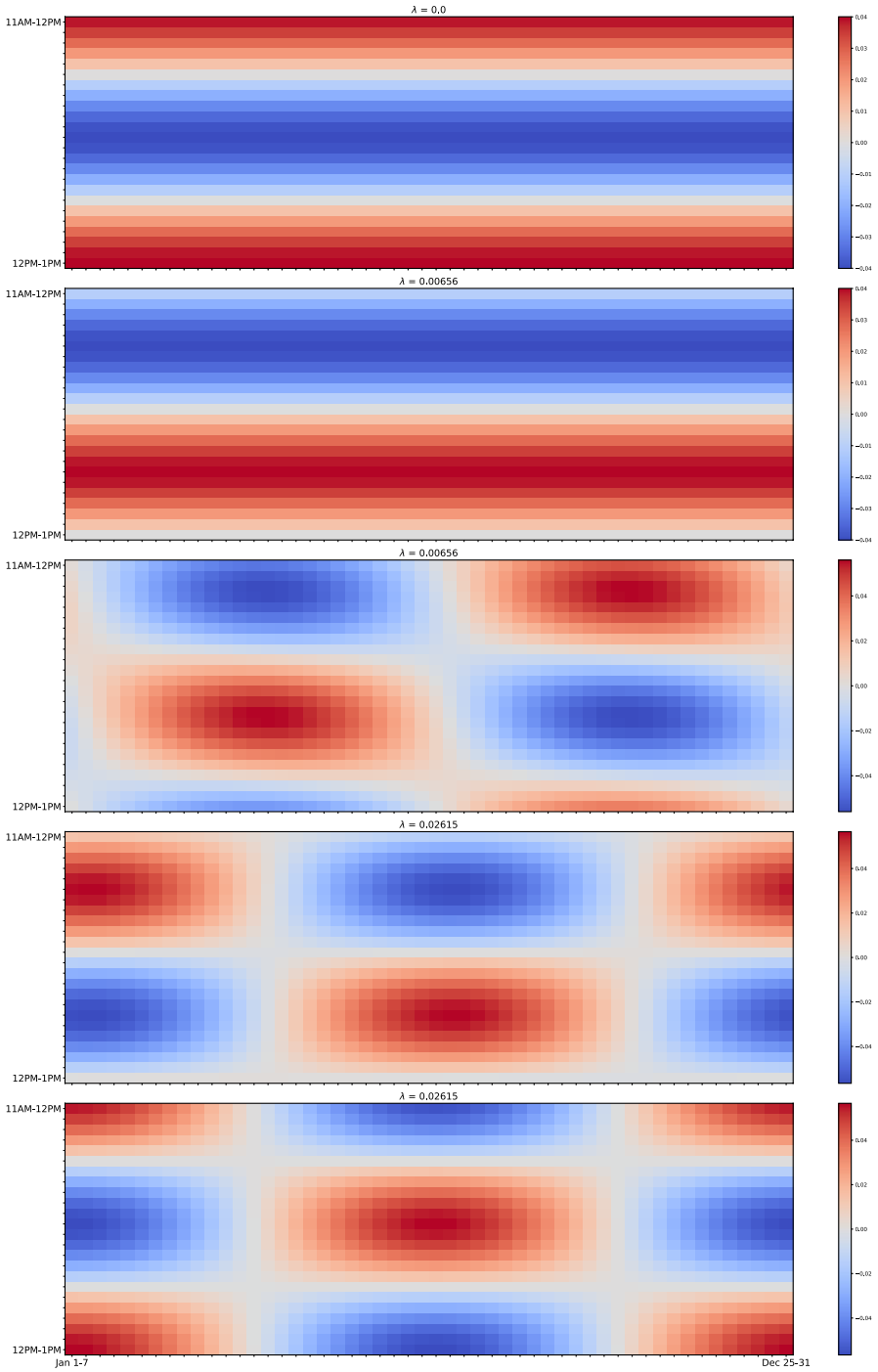


**Fig. 5** Heatmaps of the eigenvectors of the week/hour regularization graph Laplacian corresponding to $\lambda_5, \ldots, \lambda_9$

**Table 2** Results for Sect. 4.2

| Model | Train ANLL | Validation ANLL | Test ANLL |
|---|---|---|---|
| Separate | 0.186 | 0.447 | 0.448 |
| Common | 0.255 | 0.488 | 0.488 |
| Standard stratified | 0.172 | 0.393 | 0.394 |
| Eigen-stratified | 0.183 | 0.377 | 0.378 |

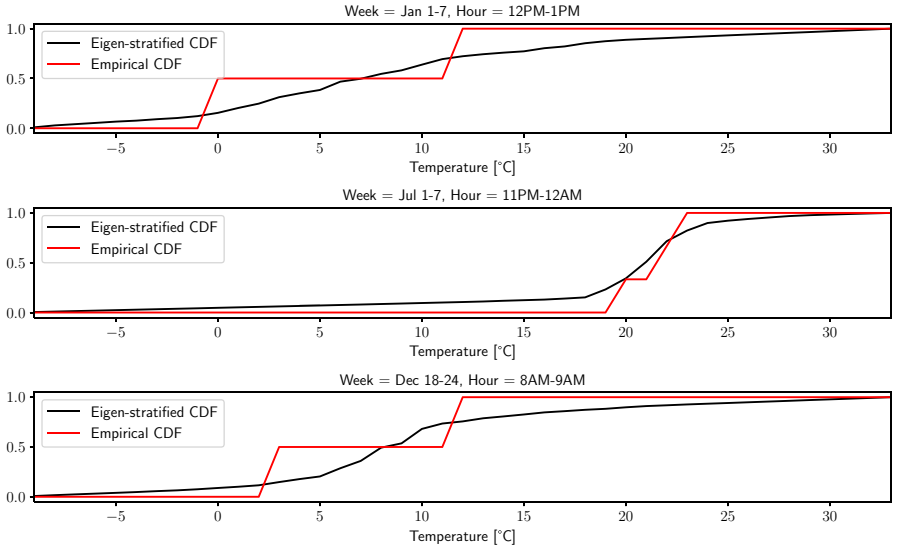


**Fig. 6** CDFs of various weeks of the year and hours of the day, as given by the eigen-stratified model, along with their corresponding empirical CDFs

## Appendix: Eigenvectors and eigenvalues of common graphs

The direct relation of a graph's structure to the eigenvalues and eigenvectors of its corresponding graph Laplacian is well-known (Jr and Morley 1985). In some cases, mentioned below, we can find them analytically, especially when the graph has many symmetries. The eigenvectors are given in normalized form (i.e., $\|q_k\|_2 = 1$.) Outside of these common graphs, many other simple graphs can be analyzed analytically; see, *e.g.*, (Brouwer and Haemers 2012).

*A note on complex graphs.* If a graph is complex, i.e., there is no analytical form for its graph Laplacian's eigenvalues and eigenvectors, the bottom eigenvalues and eigenvectors of the Laplacian of a graph can be computed extremely efficiently by, *e.g.*, the Lanczos algorithm, variants of power iteration, the Jacobi eigenvalue algorithm, the folded spectrum method, and other exotic methods. We refer the reader to (Jacobi 1846; Mises and Pollaczek-Geiringer 1929; MacDonald 1934; Lanczos 1950; Ojalvo and Newman 1970; Paige 1971) for these methods.

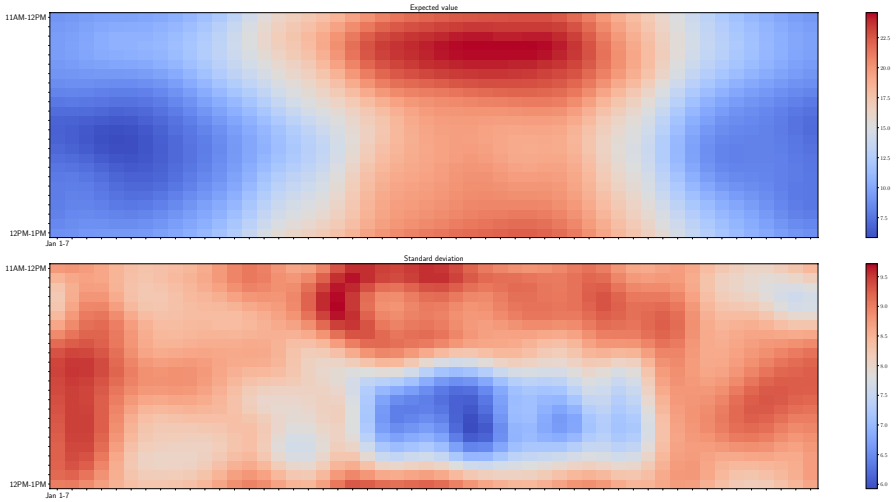


**Fig. 7** Heatmaps of the expected value (top) and standard deviation (bottom) of the distributions given by the eigen-stratified model

*Path graph.* A path or linear/chain graph is a graph whose vertices can be listed in order, with edges between adjacent vertices in that order. The first and last vertices only have one edge, whereas the other vertices have two edges. Figure 8 shows a path graph with 8 vertices and unit weights.

Eigenvectors $q_1, \ldots, q_K$ of a path graph Laplacian with $K$ nodes and unit edge weights are given by

$$q_k = \cos(\pi kv/K - \pi k/2K)/\|\cos(\pi kv/K - \pi k/2K)\|_2 \quad k = 0, \ldots, K-1,$$

where $v = (0, \ldots, K-1)$ and $\cos(\cdot)$ is applied elementwise. The eigenvalues are $2 - 2\cos(\pi k/K), k = 0, \ldots, K-1$.

*Cycle graph.* A cycle graph or circular graph is a graph where the vertices are connected in a closed chain. Every node in a cycle graph has two edges. Figure 9 shows a cycle graph with 10 vertices and unit weights.

Eigenvectors of a cycle graph Laplacian with $K$ nodes and unit weights are given by

$$\frac{1}{\sqrt{K}}\mathbf{1}, k = 0$$
$$\cos(2\pi kv/K)/\|\cos(2\pi kv/K)\|_2 \text{ and } \sin(2\pi kv/K)/\|\sin(2\pi kv/K)\|_2, \quad k = 1, \ldots, K/2,$$

where $v = (0, \ldots, K-1)$ and $\cos(\cdot)$ and $\sin(\cdot)$ are applied elementwise. The eigenvalues are $2 - 2\cos(2\pi k/K), k = 0, \ldots, K-1$.

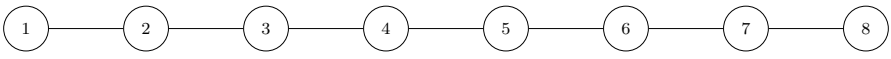


**Fig. 8** A path graph with 8 vertices and unit weights

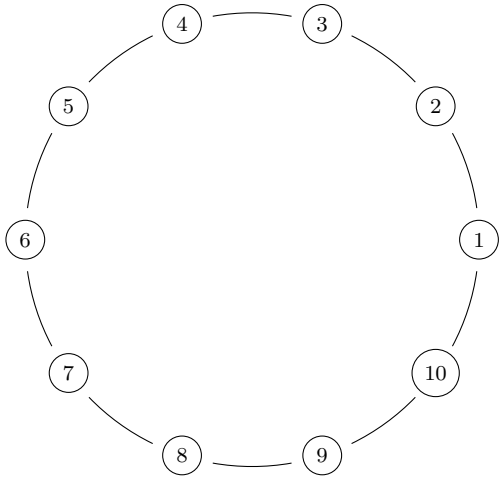


**Fig. 9** A cycle graph with 10 vertices and unit weights

*Star graph.* A star graph is a graph where all of the vertices are only connected to one central vertex. Figure 10 shows an example of a star graph with 10 vertices (9 outer vertices) and unit weights.

Eigenvectors of a star graph with $K$ vertices (*i.e.*, $K-1$ outer vertices) and unit edge weights are given by

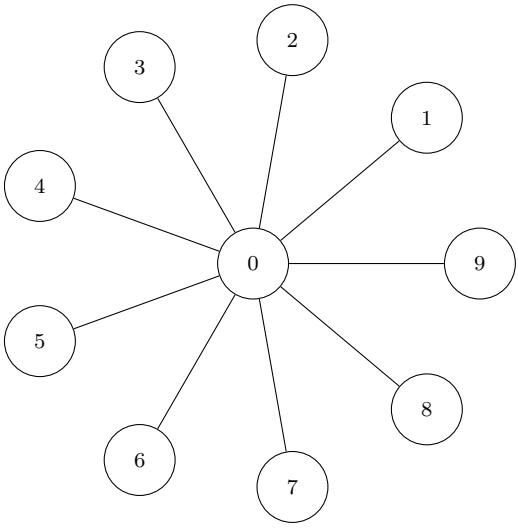


**Fig. 10** A star graph with 10 vertices (9 outer vertices) and unit weights

$$
\begin{aligned}
q_0 &= \frac{1}{\sqrt{K}}\mathbf{1} \\
q_k &= \frac{1}{\sqrt{2}}(e_i - e_{i+1}), \quad 1 \le i \le K-2 \\
q_{K-1} &= \frac{1}{\sqrt{K(K-1)}}(K-1, -1, -1, \ldots, -1, -1),
\end{aligned}
$$

where $e_i$ is the $i$th basis vector in $\mathbf{R}^K$. The smallest eigenvalue of this graph is zero, the largest eigenvalue is $K$, and all other eigenvalues are 1.

*Wheel graph.* A wheel graph with $K$ nodes consists of a center (*hub*) vertex and a ring of $K-1$ peripheral vertices, each connected to the hub (Boyd et al. 2009). Figure 11 shows a wheel graph with 11 vertices (10 peripheral vertices) and unit weights.

Eigenvectors of a wheel graph with $K$ vertices (i.e., $K-1$ peripheral vertices) are given by (Zhang et al. 2009)

$$
\begin{aligned}
q_0 &= \frac{1}{\sqrt{K}}\mathbf{1} \\
q_k &= \sin(2\pi kv/K)/\|\sin(2\pi kv/K)\|_2, \quad 1 \le i \le K-2, i \text{ odd} \\
q_k &= \cos(2\pi kv/K)/\|\cos(2\pi kv/K)\|_2, \quad 1 \le i \le K-2, i \text{ even} \\
q_{K-1} &= \frac{1}{\sqrt{K(K-1)}}(K-1, -1, -1, \ldots, -1, -1),
\end{aligned}
$$

where $v = (0, \ldots, K-1)$ and $\cos(\cdot)$ and $\sin(\cdot)$ are applied elementwise. The smallest eigenvalue of the graph is zero, the largest eigenvalue is $K$, and the middle eigenvalues are given by $3 - 2\cos(2\pi i/(K-1)), i = 1, \ldots, (K-2)/2$, with multiplicity 2 (Butler 2008).

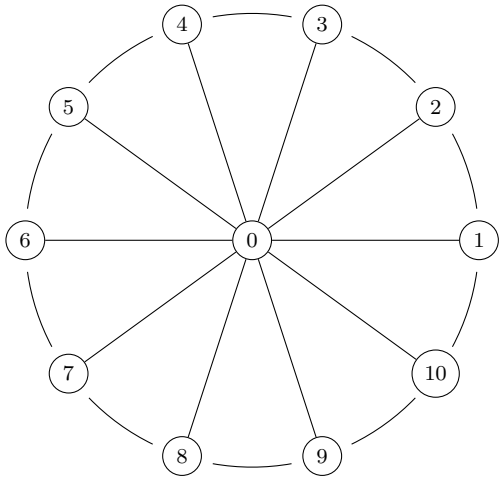


**Fig. 11** A wheel graph with 11 vertices (10 peripheral vertices) and unit weights

*Complete graph* A complete graph contains every possible edge; we assume here the edge weights are all one. The first eigenvector of a complete graph Laplacian with $K$ nodes is $\frac{1}{\sqrt{K}}\mathbf{1}$, and the other $K-1$ eigenvectors are any orthonormal vectors that complete the basis. The eigenvalues are 0 with multiplicity 1, and $K$ with multiplicity $K-1$.

Figure 12 shows an example of a complete graph with 8 vertices and unit weights.

*Complete bipartite graph* A bipartite graph is a graph whose vertices can be decomposed into two disjoint sets such that no two vertices share an edge within a set. A *complete* bipartite graph is a bipartite graph such that every pair of vertices in the two sets share an edge. We denote a complete bipartite graph with $\alpha$ vertices on the first set and $\beta$ vertices on the second set as an $(\alpha, \beta)$-complete bipartite graph. We have that $\alpha + \beta = K$, and use the convention that $\alpha \leq \beta$. Figure 13 illustrates an example of a complete bipartite graph with $(\alpha, \beta) = (3, 6)$ and unit weights.

Eigenvectors of an $(\alpha, \beta)$-complete bipartite graph with unit edge weights are given by (Merris 1998):

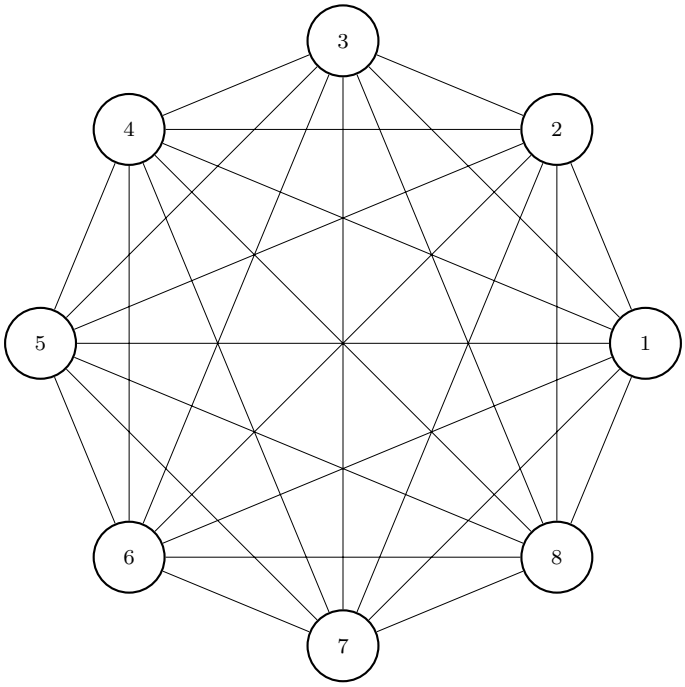


**Fig. 12** A complete graph with 8 vertices and unit weights

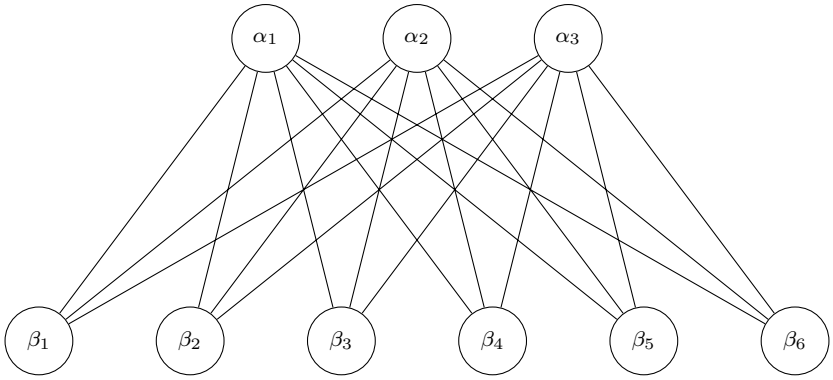


**Fig. 13** A (3,6)-complete bipartite graph with unit weights

$$
\begin{aligned}
q_0 &= \frac{1}{\sqrt{K}}\mathbf{1} \\
q_k &= \frac{1}{\sqrt{2}}(e_k - e_{k+1}), \quad 1 \le k \le \alpha - 1 \\
q_k &= \frac{1}{\sqrt{2}}(e_k - e_{k+1}), \quad \alpha \le k \le K - 1 \\
(q_{K-1})_i &= \begin{cases} \frac{-\beta}{\sqrt{\alpha^2\beta+\beta^2\alpha}} & 1 \le i \le \alpha \\ \frac{\alpha}{\sqrt{\alpha^2\beta+\beta^2\alpha}} & \alpha < i \le K \end{cases}.
\end{aligned}
$$

The eigenvalues are zero (multiplicity 1), $\alpha$ (multiplicity $\beta - 1$), $\beta$ (multiplicity $\alpha - 1$), and $K = \alpha + \beta$ (multiplicity 1).

*Scaling and products of graphs* We can find the eigenvectors and eigenvalues of the graph Laplacian of more complex graphs using some simple relationships. First, if we scale the edge weights of a graph by $\alpha \ge 0$, the eigenvectors remain the same, and the eigenvalues are scaled by $\alpha$. Second, the eigenvectors of a Cartesian product of graph Laplacians are given by the Kronecker products between the eigenvectors of each of the individual graph Laplacians; the eigenvalues consist of the sums of one eigenvalue from one graph and one from the other. This can be seen by noting that the Laplacian matrix of the Cartesian product of two graphs with graph Laplacians $L_1 \in \mathbf{R}^{P\times P}$ and $L_2 \in \mathbf{R}^{Q\times Q}$ is given by

$$L = (L_1 \otimes I) + (I \otimes L_2),$$

where $L$ is the Laplacian matrix of the Cartesian product of the two graphs. With Cartesian products of graphs, we find it convienent to index the eigenvalues and eigenvectors of the Laplacian by two indices, i.e., the eigenvalues may be denoted as $\lambda_{i,j}$ with corresponding eigenvector $q_{i,j}$ for $i = 0, \ldots, P-1$ and $j = 0, \ldots, Q-1$. (The eigenvalues will need to be sorted, as explained below.)

As an example, consider a graph which is the product of a chain graph with $P$ vertices, edge weights $\alpha^{\text{ch}}$ and eigenvalues $\lambda^{\text{ch}} \in \mathbf{R}^P$; and a cycle graph with $Q$ vertices, edge weights $\alpha^{\text{cy}}$, and eigenvalues $\lambda^{\text{ch}} \in \mathbf{R}^P$. The eigenvalues have the form

$$\lambda_i^{\text{ch}} + \lambda_j^{\text{cy}}, \quad i = 0 \ldots, P-1, \quad j = 0, \ldots, Q-1,$$

To find the $m$ smallest of these, we sort them. The order depends on the ratio of the edge weights, $\alpha^{\text{ch}}/\alpha^{\text{cy}}$.

As a very specific example, take $P = 4$ and $Q = 5$, $\alpha^{\text{ch}} = 1$, and $\alpha^{\text{cy}} = 2$. The eigenvalues of the chain and cycle graphs are

$$\lambda^{\text{ch}} = (0, 0.586, 2, 3.414), \quad \lambda^{\text{cy}} = (0, 2.764, 2.764, 7.236, 7.236).$$

The bottom six eigenvalues of the Cartesian product of these two graphs are then

$$0, 0.586, 2, 2.764, 2.764, 3.350.$$

## References


Ahmed N, Natarajan T, Rao K (1974) Discrete cosine transform. IEEE Trans Comput C-23(1):90–93

Aspvall B, Gilbert J (1984) Graph coloring using eigenvalue decomposition. SIAM J Matrix Anal Appl 5(4):526–13

Boyd S, Vandenberghe L (2004) Convex Optimization. Cambridge University Press, Cambridge

Boyd S, Diaconis P, Xiao L (2004) Fastest mixing markov chain on a graph. SIAM Rev 46(4):667–689

Boyd S, Diaconis P, Parrilo P, Xiao L (2009) Fastest mixing markov chain on graphs with symmetries. SIAM J Optim 20(2):792–819. https://doi.org/10.1137/070689413

Boyd S, Parikh N, Chu E, Peleato B, Eckstein J (2011) Distributed optimization and statistical learning via the alternating direction method of multipliers. Found Trends Mach Learn 3(1):1–122

Brélaz D (1979) New methods to color the vertices of a graph. Commun ACM 22(4):251–256

Brooks R (1941) On colouring the nodes of a network. Math Proc Cambridge Philos Soc 37:194–197

Brouwer A, Haemers W (2012) Spectra of Graphs. Springer, New York, New York

Butler S (2008) Eigenvalues and structures of graphs. PhD thesis, University of California, San Diego

Chung F (1997) Spectral Graph Theory. American Mathematical Society, Providence

Cohen-Steiner D, Kong W, Sohler C, Valiant G (2018) Approximating the spectrum of a graph. In: Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery and data mining, KDD '18. pp 1263–1271

Diamond S, Boyd S (2016) CVXPY: A Python-embedded modeling language for convex optimization. J Mach Learn Res 17(1):2909–2913

Elman J (1990) Finding structure in time. Cognit Sci 14(2):179–211

Fu A, Narasimhan B, Boyd S (2019a) CVXR: an R package for disciplined convex optimization. J Stat Softw

Fu A, Zhang J, Boyd S (2019b) Anderson accelerated Douglas-Rachford splitting. http://stanford.edu/~boyd/papers/a2dr.html

Grant M, Boyd S (2014) CVX: Matlab software for disciplined convex programming, version 2.1. http://cvxr.com/cvx

Grant M, Boyd S, Ye Y (2006) Disciplined convex programming. Global optimization: from theory to implementation. Optimization and its application series. Springer, Nonconvex, pp 155–210

Guo C, Berkhahn F (2016) Entity embeddings of categorical variables. CoRR arxiv: abs/1604.06737

Hallac D, Leskovec J, Boyd S (2015) Network lasso: Clustering and optimization in large graphs. In: Proceedings of the ACM international conference on knowledge discovery and data mining. ACM, pp 387–396

Hallac D, Park Y, Boyd S, Leskovec J (2017) Network inference via the time-varying graphical lasso. In: Proceedings of the ACM international conference on knowledge discovery and data mining. ACM, pp 205–213

Jacobi C (1846) Über ein leichtes verfahren die in der theorie der säcularstörungen vorkommenden gleichungen numerisch aufzulösen. Journal für die reine und angewandte Mathematik 1846(30):51–94

Jr WA, Morley T (1985) Eigenvalues of the laplacian of a graph. Linear and Multilinear Algebra 18(2):141–145

Jung A, Tran N (2019) Localized linear regression in networked data. IEEE Signal Process Lett 26(7)1090–1094. Doi: https://doi.org/10.1109/LSP.2019.2918933

Kaggle (2019) Cardiovascular disease dataset. https://www.kaggle.com/sulianova/cardiovascular-disease-dataset

Lanczos C (1950) An iteration method for the solution of the eigenvalue problem of linear differential and integral operators. J Res Natl Bureau Standards 45

MacDonald J (1934) On the modified ritz variation method. Phys Rev 46:828–828

Merris R (1998) Laplacian graph eigenvectors. Linear Algebra Appl 278(1):221–236

Mikolov T, Chen K, Corrado G, Dean J (2013) Efficient estimation of word representations in vector space. arXiv e-prints

Mises R, Pollaczek-Geiringer H (1929) Praktische verfahren der gleichungsauflösung. ZAMM J Appl Math Mech 9(2):152–164

Ng A, Jordan M, Weiss Y (2002) On spectral clustering: Analysis and an algorithm. In: Dietterich TG, Becker S, Ghahramani Z (eds) Advances in Neural Information Processing Systems 14. MIT Press, Cambridge, pp 849–856

Ojalvo I, Newman M (1970) Vibration modes of large structures by an automatic matrix-reduction method. AIAA J 8(7):1234–1239

Oppenheim A, Schafer R (2009) Discrete-Time Signal Processing, 3rd edn. Prentice Hall Press, Upper Saddle River, NJ, USA

Paige C (1971) The computation of eigenvalues and eigenvectors of very large sparse matrices. PhD thesis, University of London

Parikh N, Boyd S (2014) Proximal algorithms. Found Trends Optim 1(3):127–239

Spielman D (2010) Algorithms, graph theory, and linear equations in Laplacian matrices. In: Proceedings of the international congress of mathematicians. World Scientific, pp 2698–2722

Sun J, Boyd S, Xiao L, Diaconis P (2006) The fastest mixing Markov process on a graph and a connection to a maximum variance unfolding problem. SIAM Rev 48(4):681–699

Tran N, Ambos H, Jung A (2020) Classifying partially labeled networked data via logistic network lasso. In: ICASSP 2020—2020 IEEE international conference on acoustics, speech and signal processing (ICASSP), pp 3832–3836. https://doi.org/10.1109/ICASSP40776.2020.9054408

Tuck J, Barratt S, Boyd S (2019) A distributed method for fitting Laplacian regularized stratified models. arXiv preprint arXiv:190412017

Tuck J, Hallac D, Boyd S (2019) Distributed majorization-minimization for Laplacian regularized problems. IEEE/CAA J Autom Sinica 6(1):45–52

Weinberger K, Sha F, Zhu Q, Saul L (2007) Graph Laplacian regularization for large-scale semidefinite programming. In: Schölkopf B, Platt JC, Hoffman T (eds) Advances in Neural Information Processing Systems 19. MIT Press, Cambridge, pp 1489–1496

Zhang Y, Liu X, Yong X (2009) Which wheel graphs are determined by their Laplacian spectra? Comput Math Appl 58(10):1887–1890