A Distributed Method for Fitting Laplacian Regularized Stratified Models

Jonathan Tuck Shane Barratt **Stephen Boyd**

International Conference on Statistical Optimization and Learning
Beijing Jiatong University, June 19 2019

Outline

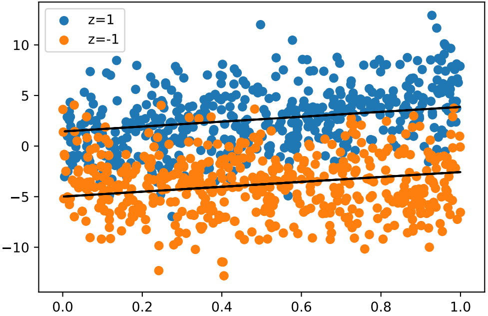
Data records

- Data records have form $(z, x, y) \in \mathcal{Z} \times \mathcal{X} \times \mathcal{Y}$
- $z \in \mathcal{Z} = \{1, \dots, K\}$ are categorical features (we'll stratify over)
- $x \in \mathcal{X}$ are the other features
- $y \in \mathcal{Y}$ is the label/dependent variable

Stratified model

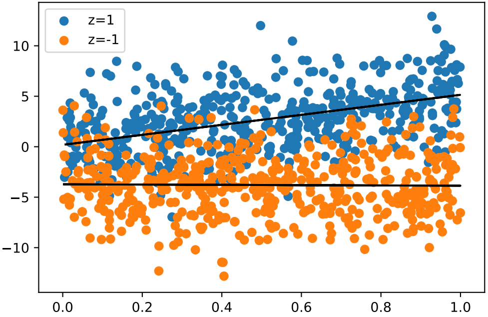
- Fit a model to data (z, x, y)
- Stratified model: fit different model for each value of z
- θ_k is model parameter for $z = k$
- $\theta = (\theta_1, \ldots, \theta_K) \in \Theta \subseteq \mathbf{R}^{Kn}$ parameterizes the stratified model
- Old idea [Kernan 99]
- Example: stratified regression model $\hat{y} = x^T \theta_z$

Example



$$\hat{y} = \theta_1 + \theta_2 x + \theta_3 z, \quad z \in \{-1, 1\}$$

Example



$$\hat{y} = \begin{cases} \theta_{-1,1} + \theta_{-1,2}x & z = -1 \\ \theta_{1,1} + \theta_{1,2}x & z = 1 \end{cases}$$

Stratified model

- Stratified model is simple function of x (*e.g.*, linear), arbitrary function of z
- Examples: separate models for
  - $z \in \{\text{Male, Female}\}$
  - $z \in \{\text{Monday}, \ldots, \text{Sunday}\}$

- If K is large, might not have enough training data to fit θ_k
- Extreme case: no training data for some values of z
- **We'll add regularization so nearby θ_k's are close**

Stratified model with Laplacian regularization

- Choose $\theta = (\theta_1, \ldots, \theta_K)$ to minimize

$$\sum_{i=1}^{N} l(\theta_{z_i}, x_i, y_i) + \sum_{k=1}^{K} r(\theta_k) + \sum_{u,v=1}^{K} W_{uv} \|\theta_u - \theta_v\|^2$$

- l is loss function, r is (local) regularization
- Last term is **Laplacian regularization**
- $W_{uv} \geq 0$ are edge weights of graph with node set $\mathcal{Z}$
- Convex problem when l, r convex in θ

Stratified model with Laplacian regularization

- Graph encodes prior that nearby values of z should have similar models
- Can be used to capture periodicities, other structure
- Examples:
  - θ_{male} and θ_{female} should be close
  - θ_{jan} should be close to θ_{feb} and θ_{dec}
- Model for each value of z 'borrows strength' from its neighbors
- Works even when there's no data for some values of z
- As $W_{uv} \to 0$, get traditional (unregularized) stratified model
- As $W_{uv} \to \infty$, get common model ($\theta_1 = \cdots = \theta_K$)

Related work

- Network lasso [Hallac 15])
- Pliable lasso [Tibshirani 17]
- Varying-coefficient models [Hastie 93, Fan 08]
- Multi-task learning [Caruana 97]

Outline

Point estimate: Predict y

- Regression: $\mathcal{X} = \mathbf{R}^n$, $\mathcal{Y} = \mathbf{R}$
  - $l(\theta, x, y) = p(x^T\theta - y)$, p is penalty function
  - $\hat{y} = x^T\theta_z$
- Classification: $\mathcal{X} = \mathbf{R}^n$, $\mathcal{Y} = \{-1, 1\}$
  - $l(\theta, x, y) = p(yx^T\theta)$
  - $\hat{y} = \mathbf{sign}(x^T\theta_z)$
- M-class classification: $\mathcal{X} \in \mathbf{R}^n$, $\mathcal{Y} = \{1, \ldots, M\}$
  - $l(\theta, x, y) = p_y(x^T\theta)$, $\theta \in \mathbf{R}^{n\times M}$, p_y is penalty function for class y
  - $\hat{y} = \mathrm{argmax}(x^T\theta_z)$

Conditional distribution estimate: Predict $\mathbf{prob}(y \mid x, z)$

- Multinomial logistic regression: $\mathcal{X} = \mathbf{R}^n$, $\mathcal{Y} = \{1, \ldots, M\}$
- Conditional distribution:

$$\mathbf{prob}(y \mid x, z) = \frac{\exp(x^T \theta_z)_y}{\sum_{j=1}^{M} \exp(x^T \theta_z)_j}, \quad y = 1, \ldots, M$$

- Loss function (convex in θ):

$$l(\theta, x, y) = \log \left(\sum_{j=1}^{M} \exp(x^T \theta)_j \right) - (x^T \theta)_y$$

Distribution estimate: Predict $p(y \mid z)$

- Gaussian distribution: $\mathcal{Y} = \mathbf{R}^m$
- Density:

$$p(y \mid z) = (2\pi)^{-m/2} \det(\Sigma)^{-1/2} \exp(-1/2(y - \mu)^T \Sigma^{-1}(y - \mu))$$

- Use natural parameter $\theta = (S, \nu) = (\Sigma^{-1}, \Sigma^{-1}\mu)$ (so $\Sigma = S^{-1}$, $\mu = S^{-1}\nu$)
- Loss function (convex in θ):

$$l(\theta, y) = -\log\det S + y^T S y - 2y^T \nu + \nu^T S^{-1} \nu$$

Outline

Stratified models

Data models

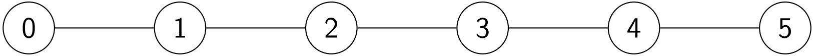
Regularization graphs

Solution method

Examples

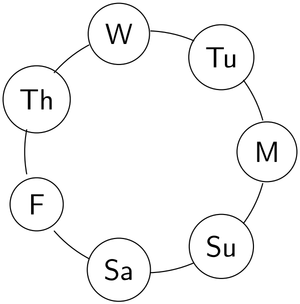
Conclusions

Path graph



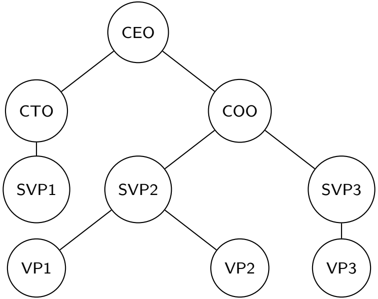
- Models time, distance, ...
- Yields time-varying, distance-varying models

Cycle graph



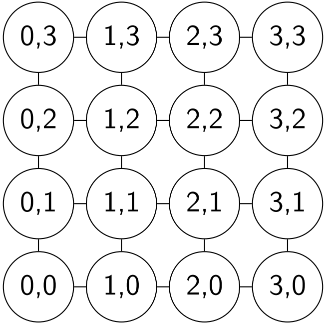
- Yields diurnal, weekly, seasonally-varying models

Tree graph



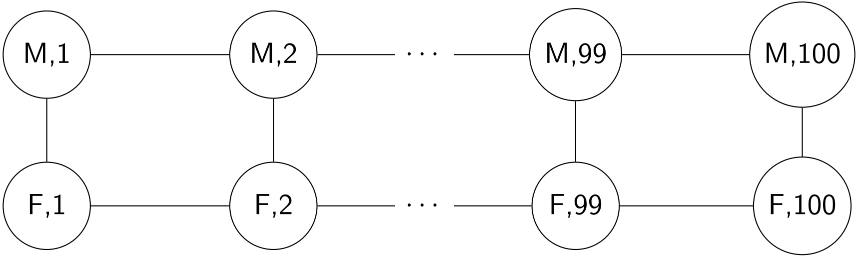
- Yields hierarchical models

Grid graph



- Yields (2D) space-varying models

Products of graphs



- $\mathcal{Z} = \{\text{Male, Female}\} \times \{1, 2, \ldots, 99, 100\}$ (sex × age)
- Yields sex, age-varying models

Outline

Fitting problem

- To fit stratified model, minimize $\ell(\theta) + r(\theta) + \mathcal{L}(\theta)$
- $\ell(\theta) = \sum_{k=1}^{K} \ell_k(\theta_k)$ is loss, $\ell_k(\theta_k) = \sum_{i:z_i=k} l(\theta_k, x_i, y_i)$
- $r(\theta) = \sum_{k=1}^{K} r(\theta_k)$ is (local) regularization
- $\mathcal{L}(\theta) = \sum_{u,v=1}^{K} W_{uv} \|\theta_u - \theta_v\|^2$ is Laplacian regularization

- ℓ, r are separable in θ_k
- $\mathcal{L}$ is quadratic, separable in components of θ_k

- We'll use operator splitting method (ADMM)

Reformulation

- Replicate variables:

$$\begin{array}{ll} \text{minimize} & \ell(\theta) + r(\tilde{\theta}) + \mathcal{L}(\hat{\theta}) \\ \text{subject to} & \theta = \tilde{\theta} = \hat{\theta} \end{array}$$

- Augmented Lagrangian

$$L(\theta, \tilde{\theta}, \hat{\theta}, u, \tilde{u}) = \ell(\theta) + r(\tilde{\theta}) + \mathcal{L}(\hat{\theta}) + \frac{1}{2t}\|\theta - \hat{\theta} + u\|_2^2 + \frac{1}{2t}\|\tilde{\theta} - \hat{\theta} + \tilde{u}\|_2^2$$

- $u, \tilde{u}$ dual variables for the two constraints, $t > 0$ is algorithm parameter

ADMM

- Augmented Lagrangian

$$L(\theta, \tilde{\theta}, \hat{\theta}, u, \tilde{u}) = \ell(\theta) + r(\tilde{\theta}) + \mathcal{L}(\hat{\theta}) + \frac{1}{2t}\|\theta - \hat{\theta} + u\|_2^2 + \frac{1}{2t}\|\tilde{\theta} - \hat{\theta} + \tilde{u}\|_2^2$$

- ADMM: for $i = 1, 2, \ldots$

$$\begin{aligned}
\theta^{i+1}, \tilde{\theta}^{i+1} &:= \operatorname*{argmin}_{\theta, \tilde{\theta}} L(\theta, \tilde{\theta}, \hat{\theta}^i, u^i, \tilde{u}^i) \\
\hat{\theta}^{i+1} &:= \operatorname*{argmin}_{\hat{\theta}} L(\theta^i, \tilde{\theta}^i, \hat{\theta}, u^i, \tilde{u}^i) \\
u^{i+1} &:= u^i + \theta^{i+1} - \hat{\theta}^{i+1} \\
\tilde{u}^{i+1} &:= \tilde{u}^i + \tilde{\theta}^{i+1} - \hat{\theta}^{i+1}
\end{aligned}$$

ADMM

- First step can be expressed as

$$\theta_k^{i+1} = \mathbf{prox}_{t\ell_k}(\hat{\theta}_k^i - u_k^i), \quad \tilde{\theta}_k^{i+1} = \mathbf{prox}_{tr}(\hat{\theta}_k^i - \tilde{u}_k^i), \quad k = 1, \ldots, K$$

- proximal operator of tg is

$$\mathbf{prox}_{tg}(\nu) = \operatorname*{argmin}_{\theta} \left(tg(\theta) + (1/2)\|\theta - \nu\|_2^2 \right)$$

- Can evaluate these $2K$ proximal operators in parallel

Loss proximal operators

- Often has closed form expression or efficient implementation
- Square loss: $\ell_k(\theta_k) = \|X_k\theta_k - y_k\|_2^2$
  - $\mathbf{prox}_{t\ell_k}(\nu) = (I + tX_k^TX_k)^{-1}(\nu - tX_k^Ty_k)$
  - Cache factorization or warm-start CG
- Logistic loss: $\ell_k(\theta_k) = \sum_i \log(1 + \exp(-y_{ki}\theta_k^T x_{ki}))$
  - Use L-BFGS to evaluate $\mathbf{prox}_{t\ell_k}(\nu)$
  - Warm-start
- Many others ...

Regularizer proximal operators

- Often has closed form expression or efficient implementation

Regularizer	$r(\theta_k)$	$\mathbf{prox}_{tr}(\nu)$
Sum of squares (ℓ_2)	$(\lambda/2)\|\theta_k\|_2^2$	$\nu/(t\lambda+1)$
ℓ_1 norm	$\lambda\|\theta_k\|_1$	$(\nu - t\lambda)_+ - (-\nu - t\lambda)_+$
Nonnegative	$I_+(\theta_k)$	$(\theta_k)_+$

- $\lambda > 0$ local regularization parameter

Laplacian proximal operator

- Separable across each component of θ_k
- To find each component $(\theta_k)_i$, need to solve a Laplacian system
- Many efficient ways to solve, *e.g.*, diagonally preconditioned CG
- These n systems can be solved in parallel

Software implementation

- Available at `www.github.com/cvxgrp/strat_models`
- `numpy`, `scipy` for matrix operations
- `networkx` for handling graphs and graph operations
- `torch` for L-BFGS and GPU computation
- `multiprocessing` for parallelism
- `model.fit(X,Y,Z,G)` (writes θ_k on graph nodes)

Outline

House price prediction

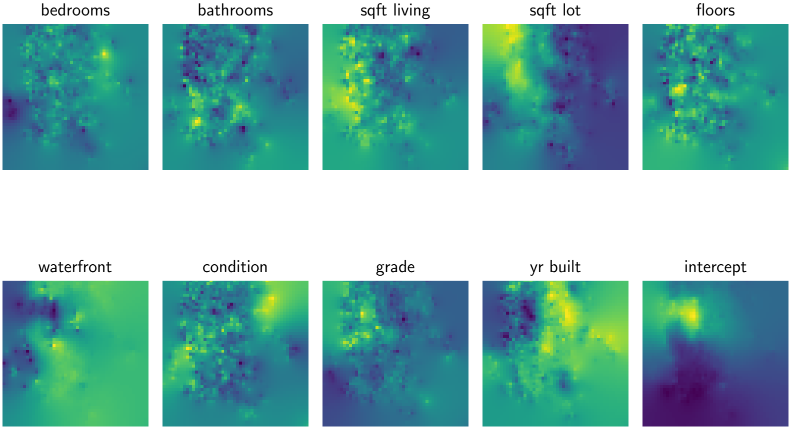
- $N \approx 22000$ (z, x, y) from King County, WA
- Split into train/test $\approx 16200/5400$
- $z = (\text{latitude bin}, \text{longitude bin})$
- $x \in \mathbf{R}^{10}$ = features of house, y = log of house price
- Graph is 50×50 grid with all edge weights same; $K = 2500$
- Stratified ridge regression model with two hyperparameters (one for local regularization, one for Laplacian regularization)

House price prediction: Results

- Compare stratifed, common, and random forest with 50 trees

Model	Parameters	RMS test error
Stratified	**25000**	**0.181**
Common	10	0.316
Random forest	985888	0.184

House price prediction: Parameters



Chicago crime prediction

- $N \approx 7\,000\,000$ (z, y) pairs for 2017, 2018
- Train on 2017, test on 2018
- $y =$ number of crimes
- $z = (\text{location bin}, \text{week of year}, \text{day of week}, \text{hour of day})$; $K \approx 3\,500\,000$
- Graph is Cartesian product of grid, three cycles; four graph edge weights
- Stratified Poisson model with four hyperparameters
  (one for each graph edge weight)

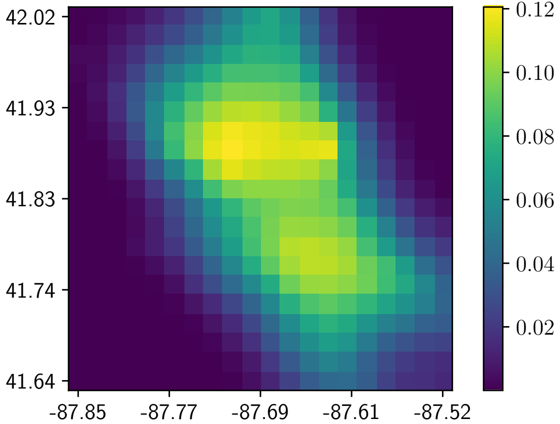
Chicago crime prediction

- Compare three models, average negative log likelihood on test data

Model	Train	Test
Separate	0.068	0.740
Stratified	**0.221**	**0.234**
Common	0.279	0.278

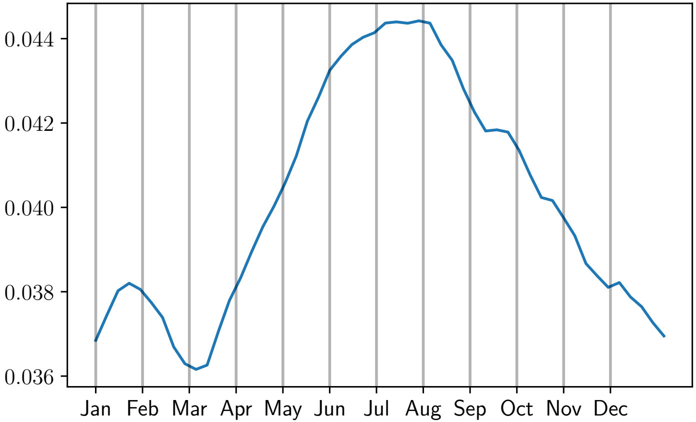
Chicago crime prediction

Crime rate as a function of latitude/longitude



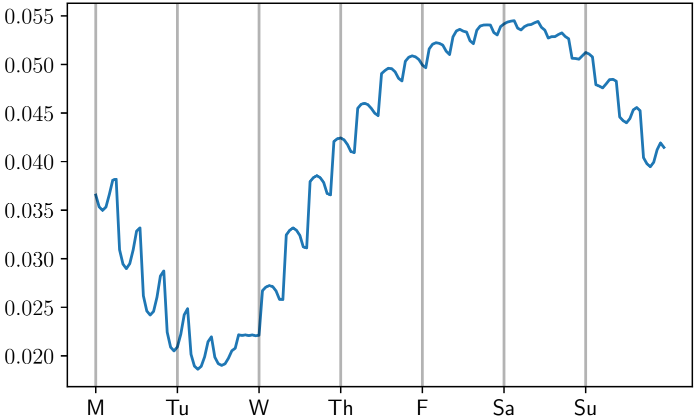
Chicago crime prediction

Crime rate as a function of week of year



Chicago crime prediction

Crime rate as a function of hour of week



Outline

Conclusions

- Stratified models combine
  - simple dependence on some features (x)
  - complex dependence on others (z)
- Often interpretable
- Laplacian regularization encodes prior on values of z, so models can borrow strength from their neighbors
- Effective method to build time-varying, space-varying, seasonally-varying models
- Efficient, distributed ADMM-based implementation for large-scale data