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

Stratified models

Data models

Regularization graphs

Solution method

Examples

Conclusions
Data records

- Data records have form $(z, x, y) \in \mathcal{Z} \times \mathcal{X} \times \mathcal{Y}$
- $z \in \mathcal{Z} = \{1, \ldots, K\}$ 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 \(z\)
- \(\theta_k\) is model parameter for \(z = k\)
- \(\theta = (\theta_1, \ldots, \theta_K) \in \Theta \subseteq \mathbb{R}^{Kn}\) 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 models
Stratified model

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

- If $K$ is large, might not have enough training data to fit $\theta_k$
- Extreme case: no training data for some values of $z$
- We’ll add regularization so nearby $\theta_k$’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 $\theta$
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:
  - $\theta_{\text{male}}$ and $\theta_{\text{female}}$ should be close
  - $\theta_{\text{jan}}$ should be close to $\theta_{\text{feb}}$ and $\theta_{\text{dec}}$
- Model for each value of $z$ ‘borrows strength’ from its neighbors
- Works even when there’s no data for some values of $z$
- As $W_{uv} \rightarrow 0$, get traditional (unregularized) stratified model
- As $W_{uv} \rightarrow \infty$, 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

Stratified models

Data models

Regularization graphs

Solution method

Examples

Conclusions
Point estimate: Predict $y$

- **Regression:** $X = \mathbb{R}^n$, $Y = \mathbb{R}$
  - $l(\theta, x, y) = p(x^T \theta - y)$, $p$ is penalty function
  - $\hat{y} = x^T \theta$

- **Classification:** $X = \mathbb{R}^n$, $Y = \{-1, 1\}$
  - $l(\theta, x, y) = p(yx^T \theta)$
  - $\hat{y} = \text{sign}(x^T \theta)$

- **$M$-class classification:** $X \in \mathbb{R}^n$, $Y = \{1, \ldots, M\}$
  - $l(\theta, x, y) = p_y(x^T \theta)$, $\theta \in \mathbb{R}^{n \times M}$, $p_y$ is penalty function for class $y$
  - $\hat{y} = \text{argmax}(x^T \theta)$
Conditional distribution estimate: Predict $\text{prob}(y \mid x, z)$

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

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

- Loss function (convex in $\theta$):

$$l(\theta, x, y) = \log \left( \sum_{j=1}^M \exp(x^T \theta)_j \right) - (x^T \theta)_y$$
Distribution estimate: Predict $p(y \mid z)$

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

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

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

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

Stratified models

Data models

Regularization graphs

Solution method

Examples

Conclusions
Path graph

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

Regularization graphs
Cycle graph

- Yields diurnal, weekly, seasonally-varying models
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 $\times$ age)

Yields sex, age-varying models
Outline

Stratified models

Data models

Regularization graphs

Solution method

Examples

Conclusions
To fit stratified model, minimize $\ell(\theta) + r(\theta) + L(\theta)$

$\ell(\theta) = \sum_{k=1}^{K} \ell_k(\theta_k)$ 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

$L(\theta) = \sum_{u,v=1}^{K} W_{uv} \| \theta_u - \theta_v \|^2$ is Laplacian regularization

$\ell, r$ are separable in $\theta_k$

$L$ is quadratic, separable in components of $\theta_k$

We’ll use operator splitting method (ADMM)
Reformulation

- Replicate variables:

\[
\begin{align*}
\text{minimize} & \quad \ell(\theta) + r(\tilde{\theta}) + L(\hat{\theta}) \\
\text{subject to} & \quad \theta = \tilde{\theta} = \hat{\theta}
\end{align*}
\]

- Augmented Lagrangian

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

- \(u, \tilde{u}\) dual variables for the two constraints, \(t > 0\) is algorithm parameter
Augmented Lagrangian

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

ADMM: for \( i = 1, 2, \ldots \)

\[
\begin{align*}
\theta^{i+1}, \tilde{\theta}^{i+1} &:= \arg\min_{\theta, \tilde{\theta}} L(\theta, \tilde{\theta}, \hat{\theta}^i, u^i, \tilde{u}^i) \\
\hat{\theta}^{i+1} &:= \arg\min_{\hat{\theta}} L(\theta^i, \tilde{\theta}^i, \hat{\theta}, u^i, \tilde{u}^i) \\
u^{i+1} &:= u^i + \theta^{i+1} - \hat{\theta}^{i+1} \\
\tilde{u}^{i+1} &:= \tilde{u}^i + \tilde{\theta}^{i+1} - \hat{\theta}^{i+1}
\end{align*}
\]
First step can be expressed as

\[ \theta_{i+1}^k = \text{prox}_{\ell_k}(\hat{\theta}_k^i - u_k^i), \quad \tilde{\theta}_{i+1}^k = \text{prox}_{r}(\hat{\theta}_k^i - u_k^i), \quad k = 1, \ldots, K \]

proximal operator of \( tg \) is

\[ \text{prox}_{tg}(\nu) = \arg\min_{\theta} \left( tg(\theta) + \frac{1}{2} \| \theta - \nu \|_2^2 \right) \]

Can evaluate these \( 2K \) proximal operators in parallel
Loss proximal operators

- Often has closed form expression or efficient implementation
- **Square loss**: \( \ell_k(\theta_k) = \|X_k \theta_k - y_k\|_2^2 \)
  - \( \text{prox}_{\ell_k}(\nu) = (I + tX_k^T X_k)^{-1}(\nu - tX_k^T y_k) \)
  - 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 \( \text{prox}_{\ell_k}(\nu) \)
  - Warm-start
- Many others . . .
Regularizer proximal operators

- Often has closed form expression or efficient implementation

<table>
<thead>
<tr>
<th>Regularizer</th>
<th>$r(\theta_k)$</th>
<th>$\text{prox}_{tr}(\nu)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sum of squares ($\ell_2$)</td>
<td>$(\lambda/2)|\theta_k|^2_2$</td>
<td>$\nu/(t\lambda + 1)$</td>
</tr>
<tr>
<td>$\ell_1$ norm</td>
<td>$\lambda|\theta_k|_1$</td>
<td>$(\nu - t\lambda)<em>+ - (-\nu - t\lambda)</em>+$</td>
</tr>
<tr>
<td>Nonnegative</td>
<td>$l_+(\theta_k)$</td>
<td>$(\theta_k)_+$</td>
</tr>
</tbody>
</table>

- $\lambda > 0$ local regularization parameter
Laplacian proximal operator

- Separable across each component of $\theta_k$
- To find each component $\left(\theta_k\right)_i$, 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 $\theta_k$ on graph nodes)
House price prediction

- $N \approx 22000 \ (z, x, y)$ from King County, WA
- Split into train/test $\approx 16200/5400$
- $z = (\text{latitude bin, longitude bin})$
- $x \in \mathbb{R}^{10} = \text{features of house, } y = \log \text{ of house price}$
- Graph is $50 \times 50$ 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 stratified, common, and random forest with 50 trees

<table>
<thead>
<tr>
<th>Model</th>
<th>Parameters</th>
<th>RMS test error</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stratified</td>
<td>25000</td>
<td>0.181</td>
</tr>
<tr>
<td>Common</td>
<td>10</td>
<td>0.316</td>
</tr>
<tr>
<td>Random forest</td>
<td>985888</td>
<td>0.184</td>
</tr>
</tbody>
</table>
House price prediction: Parameters

- bedrooms
- bathrooms
- sqft living
- sqft lot
- floors
- waterfront
- condition
- grade
- yr built
- intercept

Examples
Chicago crime prediction

- $N \approx 7\,000\,000$ $(z, y)$ pairs for 2017, 2018
- Train on 2017, test on 2018
- $y =$ number of crimes
- $z = ($location bin, week of year, day of week, 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

<table>
<thead>
<tr>
<th>Model</th>
<th>Train</th>
<th>Test</th>
</tr>
</thead>
<tbody>
<tr>
<td>Separate</td>
<td>0.068</td>
<td>0.740</td>
</tr>
<tr>
<td>Stratified</td>
<td><strong>0.221</strong></td>
<td><strong>0.234</strong></td>
</tr>
<tr>
<td>Common</td>
<td>0.279</td>
<td>0.278</td>
</tr>
</tbody>
</table>
Chicago crime prediction

Crime rate as a function of latitude/longitude
Chicago crime prediction

Crime rate as a function of week of year

![Graph showing crime rate as a function of week of year. There is a peak in August and a significant decline in December.]
Chicago crime prediction

Crime rate as a function of hour of week
Outline

Stratified models

Data models

Regularization graphs

Solution method

Examples

Conclusions
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