Multi-convex Programming for Discrete Latent Factor Models Prototyping

Hao Zhu, Shengchao Yan, Jasper Hoffmann, and Joschka Boedecker

IN-CODE Retreat, 2025

NR Lab

IN-CODE

universität freiburg

https://github.com/nrgrp/dlfm

- DLFMs appear in domains such as machine learning, economics, signal processing, control, *etc.*
- in neuroscience and psychology, DLFMs provide interpretable characterizations of neural population activities and subject behavior
- currently, fitting a DLFM to some dataset relies on customized solver for individual models
  - requires lots of background knowledge (both theoretical and technical) to implement
  - limited to the targeted specific instance of DLFMs
  - difficult to add regularization terms and constraints on the DLFM parameters and latent factors
- we propose a framework for specifying and solving DLFM fitting problems
  - supports DLFMs with loss functions and constraints of the fitting problem convex (even the fitting problem itself is not), including a wide range of regression and classification models
  - allows the users to fit a DLFM to some dataset easily (within a couple of lines of code) in high level human readable language, close to the math

Discrete latent factor models (DLFMs)

DLFMs are generally expressed as

$$z \sim \mathbf{prob}(z), \quad y \sim \mathbf{prob}(y \mid x, z, \theta)$$

- $z \in \{e_1, \ldots, e_K\} \subseteq \mathbf{R}^K$ is the *latent factor* (in vector form), θ is the *model parameter*
- x and y are the *feature* and *observation*, respectively

Standard DLFM fitting problems

$$\begin{array}{ll} \text{minimize} & \sum_{i=1}^m z_i^T r_i = \sum_{i=1}^m z_i^T (f(x_i, y_i; \theta_1), \ldots, f(x_i, y_i; \theta_K)) \\ \text{subject to} & z_i \in \{0,1\}^K, \quad \mathbf{card}\, z_i = 1, \quad i = 1, \ldots, m \\ & \theta_i \in \mathcal{C}, \quad i = 1, \ldots, K \end{array} \tag{1}$$

- variables: model parameters $\theta_1, \ldots, \theta_K$ and latent factors $z_1, \ldots, z_m$
- data: feature-observation pairs $\{x_i, y_i\}_{i=1}^m$
- the feasible set $\mathcal{C}$ is closed and convex; the loss function f is convex and resolves to scalar

Regression models

$$f(x, y; \theta) = g(x^T\theta - y)$$

- $x, \theta \in \mathbf{R}^n$, $y \in \mathbf{R}$, $g\colon \mathbf{R} \to \mathbf{R}$ is some loss function, *e.g.*,
  - (squared) ℓ_p-loss: $g(u) = u^2$, $g(u) = \|u\|_p$ for $p \in [1, \infty]$
  - Huber loss: $f(u) = u^2$ for $|u| \le \delta$, and $f(u) = 2\delta|u| - \delta^2$ for $|u| > \delta$
- nonscalar observations: $g(u) = \|u\|_2^2$, $g(u) = \|u\|_1$; $g(U) = \|U\|_F^2 = \mathbf{tr}(U^TU)$

Classification models

$$f(X, y; \theta) = -\log\left(\frac{y^T \exp u}{\sum_{i=1}^p \exp u_i}\right), \quad u = X\theta$$

- $X \in \mathbf{R}^{p \times n}$, $y \in \{e_1, \ldots, e_p\} \subseteq \mathbf{R}^p$, $\theta \in \mathbf{R}^n$
- includes binary logistic regression as a special case
- readily adapted to deal with hinge loss or exponential loss

Constraints on model parameters

- nonnegative orthant $\theta \succeq 0$, unit norm ball $\|\theta\|_2 \le 1$, probability simplex $\mathbf{1}^T\theta = 1$

Heuristic solution via BCD

relaxing the mixed integer constraints in (1), we have

$$\begin{array}{ll} \text{minimize} & \sum_{i=1}^m z_i^T r_i = \sum_{i=1}^m z_i^T (f(x_i, y_i; \theta_1), \ldots, f(x_i, y_i; \theta_K)) \\ \text{subject to} & 0 \preceq z_i \preceq \mathbf{1}, \quad \mathbf{1}^T z_i = 1, \quad i = 1, \ldots, m \\ & \theta_i \in \mathcal{C}, \quad i = 1, \ldots, K \end{array} \tag{2}$$

to solve the multi-convex problem (2), in each block coordinate descent (BCD) iteration, we alternate between solving the problems

$$\text{(P)} \quad \begin{array}{ll} \text{minimize} & \sum_{i=1}^m \tilde{z}_i^T r_i \\ \text{subject to} & r_i = (f(x_i, y_i; \theta_k))_{k=1}^K, \quad \theta_k \in \mathcal{C} \\ & i = 1, \ldots, m, \quad k = 1, \ldots, K \end{array} \qquad \text{(F)} \quad \begin{array}{ll} \text{minimize} & \sum_{i=1}^m z_i^T \tilde{r}_i \\ \text{subject to} & 0 \preceq z_i \preceq \mathbf{1}, \quad \mathbf{1}^T z_i = 1 \\ & i = 1, \ldots, m \end{array}$$

- P-problem has variables: $\theta_1, \ldots, \theta_K$ and data $\{x_i, y_i\}_{i=1}^m$ from the dataset, $\tilde{z}_1, \ldots, \tilde{z}_m \in \mathbf{R}^K$ corresponding to the optimal point of the F-problem in the last iteration
- F-problem has variables: $z_1, \ldots, z_m \in \mathbf{R}^K$ and data $\tilde{r}_i = (f(x_i, y_i; \tilde{\theta}_1), \ldots, f(x_i, y_i; \tilde{\theta}_K))$, $i = 1, \ldots, m$, where $\tilde{\theta}_1, \ldots, \tilde{\theta}_K$ are the optimal point of the P-problem in the last iteration

Regularizations

- for sparse model parameters $\theta_1, \ldots, \theta_K$: $\lambda \sum_{k=1}^K \|\theta_k\|_1$ with $\lambda \ge 0$
- for sparse latent factor change: $\lambda \sum_{i=1}^{m-1} D_{\text{kl}}(z_i, z_{i+1})$ with $\lambda \ge 0$ (D_{kl} is the KL-divergence)

Implementation

Specifying a problem

(only the commented lines need to be specified by the user)

```python
import cvxpy as cp

### problem data
xs = None  # ndarray: dataset features
ys = None  # ndarray: dataset observations
m = None  # int: number of samples in the dataset

### P-problem
K = None  # int: number of latent factors
thetas = []  # list of cp.Variable objects: model parameters
r = []  # list of cp.Expression objects: loss functions
ztil = cp.Parameter((m, K), nonneg=True)
Pobj = cp.sum(cp.multiply(ztil, cp.vstack(r).T))
Preg = 0  # cp.Expression: regularization on model parameters
Pconstr = []  # list of cp.Constraint objects: model parameter constraints
Pprob = cp.Problem(cp.Minimize(Pobj + Preg), Pconstr)

### F-problem
rtil = cp.Parameter((K, m))
z = cp.Variable((m, K))
Fobj = cp.sum(cp.multiply(z, rtil.T))
Freg = 0  # cp.Expression: regularization on latent factors
Fconstr = [z >= 0, z <= 1, cp.sum(z, axis=1) == 1]
Fprob = cp.Problem(cp.Minimize(Fobj + Freg), Fconstr)
```

Running BCD iterations

(quit when the optimal values of the P- and F-problem converge)

```python
while np.abs(Pobj.value - Fobj.value) > 1e-6:
    ztil.value = np.abs(z.value)
    Pprob.solve()
    rtil.value = cp.vstack(r).value
    Fprob.solve()
```

Examples

Hierarchical forgetting Q-learning

consider an agent performing a p-armed bandit:

- reward signal $u(t) \in \{0\} \cup \{e_1, \ldots, e_p\} \subseteq \mathbf{R}^p$ indicates if the action at time $t-1$ is rewarded
- action at time t is selected under parameters $\theta(t) \in \{\theta_1, \ldots, \theta_K\} \subseteq \mathbf{R}^n$, according to

$$\begin{aligned} &v(t) = X(t)\theta(t), \quad X(t) = \begin{bmatrix} u(t) & u(t-1) & \cdots & u(t-n+1) \end{bmatrix} \in \mathbf{R}^{p \times n}, \\ &y(t) \sim \mathrm{Cat}(\{e_1, \ldots, e_p\},\ \exp v(t)/\mathbf{1}^T \exp v(t)) \end{aligned}$$

the optimization problems in each BCD iterations are

$$\text{(P)} \quad \begin{array}{ll} \text{minimize} & \sum_{t=1}^m \tilde{z}(t)^T r(t) \\ \text{subject to} & r(t) = -\log\left(\frac{y(t)^T \exp(X(t)\theta_1)}{\mathbf{1}^T \exp(X(t)\theta_1)}, \frac{y(t)^T \exp(X(t)\theta_2)}{\mathbf{1}^T \exp(X(t)\theta_2)}\right), \quad t = 1, \ldots, m \\ & \theta_1 \ge 0, \quad \theta_{1,1} \ge \cdots \ge \theta_{1,5}, \quad \theta_2 \le 0, \quad \theta_{2,1} \le \cdots \le \theta_{2,5} \end{array}$$

$$\text{(F)} \quad \begin{array}{ll} \text{minimize} & \sum_{t=1}^m z(t)^T \tilde{r}(t) + \lambda \sum_{t=1}^{m-1} D_{\text{kl}}(z(t), z(t+1)) \\ \text{subject to} & 0 \preceq z(t) \preceq \mathbf{1}, \quad \mathbf{1}^T z(t) = 1, \quad t = 1, \ldots, m \end{array}$$

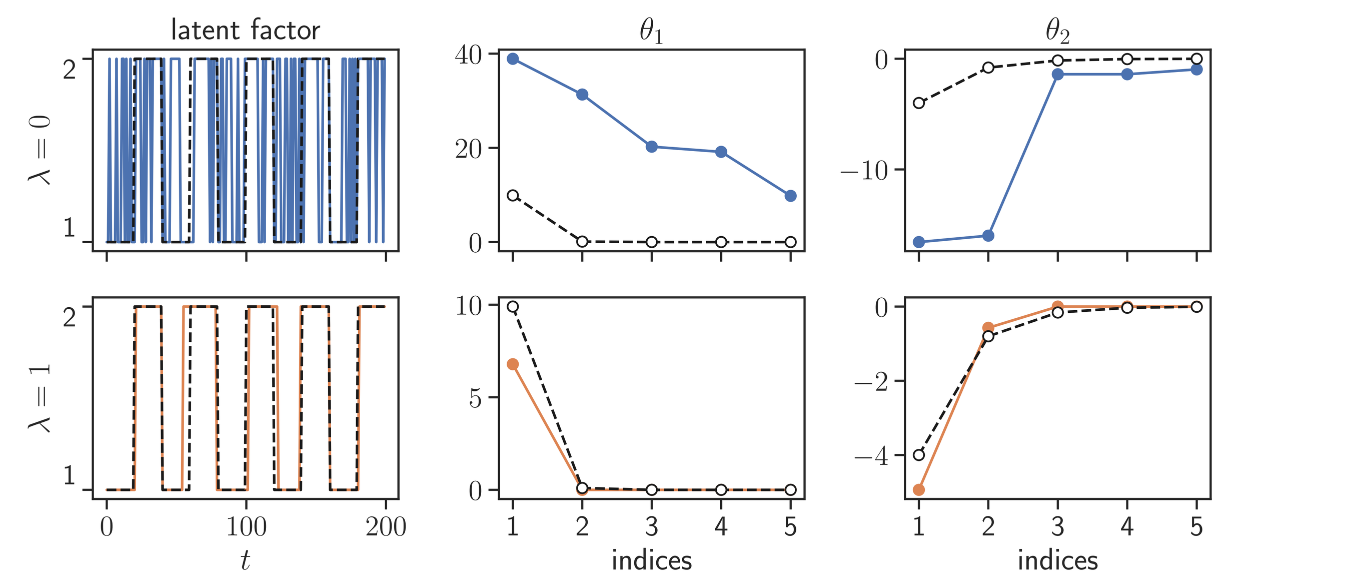

Figure 1: Colored solid lines: recovered latent factors and model parameters. Black dashed lines: ground truth.

Input-output hidden Markov model

consider a dataset generated according to

- $\hat{z}(t) \in \{1, \ldots, K\}$ from a K-state Markov chain, with coefficients $\theta_{\hat{z}(t)} \in \{\theta_1, \ldots, \theta_K\} \subseteq \mathbf{R}^n$
- $y(t) \in \{0, 1\}$ with $\mathbf{prob}(y(t) = 1) = 1/(1 + \exp(-x(t)^T\theta_{\hat{z}(t)}))$, given feature vector $x(t) \in \mathbf{R}^n$

the optimization problems in each BCD iterations are

$$\text{(P)} \quad \begin{array}{ll} \text{minimize} & \sum_{t=1}^m \tilde{z}(t)^T r(t) + \lambda_\theta \sum_{k=1}^3 \|\theta_k\|_2 \\ \text{subject to} & r(t) = -\left(y(t)x(t)^T\theta_k - \log\left(1 + e^{x(t)^T\theta_k}\right)\right)_{k=1}^3 \\ & \theta_{1,1} \le 0, \quad \theta_{2,1} \ge 0, \quad \theta_{3,1} \ge 0 \\ & t = 1, \ldots, m \end{array}$$

$$\text{(F)} \quad \begin{array}{ll} \text{minimize} & \sum_{t=1}^m z(t)^T \tilde{r}(t) + \lambda_z \sum_{t=1}^{m-1} D_{\text{kl}}(z(t), z(t+1)) \\ \text{subject to} & 0 \preceq z(t) \preceq \mathbf{1}, \quad \mathbf{1}^T z(t) = 1, \quad t = 1, \ldots, m \end{array}$$

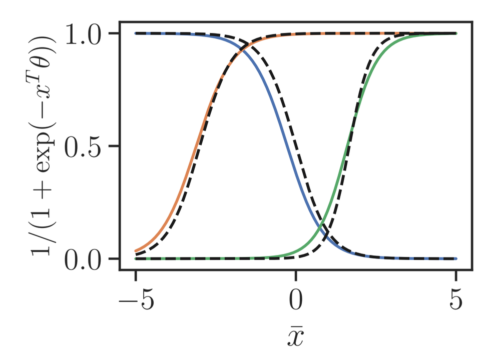

Figure 2: Colored solid lines: recovered decision curve. Black dashed lines: ground truth.

References

[BV04] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

[SDU+17] X. Shen, S. Diamond, M. Udell, Y. Gu, and S. Boyd. Disciplined multi-convex programming. In *29th Chinese Control and Decision Conference (CCDC)*, pages 895–900. IEEE, 2017.