Latent Feature Lasso

Ian E.H. Yen*, Wei-Cheng Lee†, Sung-En Chang†, Arun Suggala*,
Shou-De Lin† and Pradeep Ravikumar*.

* Carnegie Mellon University
† National Taiwan University
Presenter: Wei-Cheng Lee

Outline

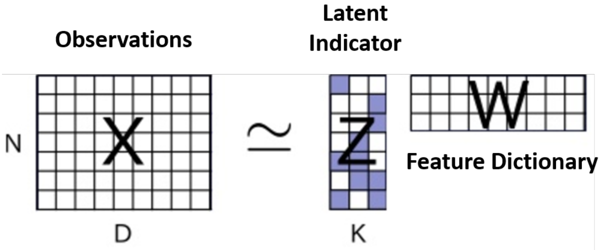
Latent Feature Models

- In **Latent Feature Model**, each observation

$$\boldsymbol{x}_n = W^T \boldsymbol{z}_n + \boldsymbol{\epsilon}_n$$

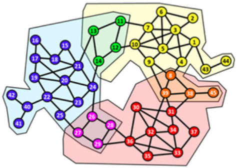
where $\boldsymbol{x}_n \in \mathbb{R}^D$: observation, $W \in \mathbb{R}^{K \times D}$: feature dictionary, $\boldsymbol{z}_n \in \{0, 1\}^K$: binary latent indicators, and $\boldsymbol{\epsilon}_n \in \mathbb{R}^D$: noise.

- **Mixture Model** is a special case with $\|\boldsymbol{z}_n\|_0 = 1$.

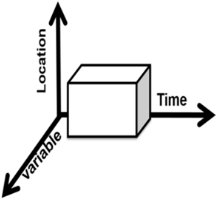


Latent Feature Model:Why Binary and Applications

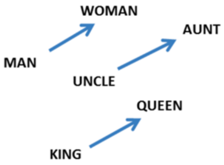
- Why binary? (interpretability, semi-supervision, and computational efficiency.)



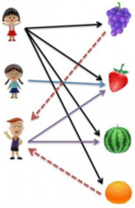
Community Detection



Spatial-Temporal Factorization

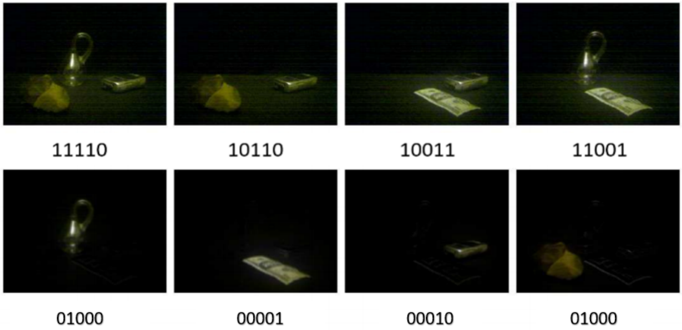


Word Embedding



Recommendation

Latent Feature Model: Tabletop Dataset



- Under Latent Feature Model assumptions, Can you identify the latent features by your own eyes?

Latent Feature Models: Result Summary

- **Goal:** Find dictionary $W_{K\times D}$ and latent indicators $Z : N \times K$ that best approximates observation $X : N \times D$.
- **Existing Approaches:**
  - MCMC, Variational (Indian Buffet Process):
    No finite-time guarantee.
  - Spectral Method (Tung 2014):
    $O(DK^6)$ sample complexity. ($z \sim$Ber(π), $x \sim N(W^T z, \sigma)$).
  - Matrix Factorization (Slawski et al., 2013):
    $O(NK2^K)$ runtime complexity for exact recovery (noiseless).
- **This Paper:**
  - A convex estimator — Latent Feature Lasso.
  - Low-order polynomial runtime and sample complexity.
  - No restrictive assumption on $p(X)$, even allows model mis-specification.

Outline

Outline

Latent Feature Model: Estimation

- Empirical Risk Minimization:

$$\min_{Z\in\{0,1\}^{N\times K}} \left\{ \min_{W\in\mathbb{R}^{K\times D}} \frac{1}{2N}\|X - ZW\|_F^2 + \frac{\tau}{2}\|W\|_F^2 \right\},$$

- Given Z, the dual problem w.r.t. W is:

$$\min_{M=ZZ^T\in\{0,1\}^{N\times N}} \underbrace{\left\{ \max_{A\in\mathbb{R}^{N\times D}} \frac{-1}{2\tau} tr(AA^T M) - L^*(A) \right\}}_{g(M)}.$$

- **Tricks:** Introducing dummy variable $E = ZW$, where A is its dual variable.
- **Key insight:** the function is convex w.r.t. $M = ZZ^T$.

Latent Feature Model: Estimation

- Let $\mathcal{S} := \{\boldsymbol{z}\boldsymbol{z}^T \mid \boldsymbol{z} \in \{0,1\}^N\}$.
- The "Latent-Feature" Atomic Norm:

$$\|M\|_{\mathcal{S}} := \min_{\boldsymbol{c} \geq 0} \sum_{\boldsymbol{z}\boldsymbol{z}^T \in \mathcal{S}} c_{\boldsymbol{z}} \quad s.t. \quad M = \sum_{\boldsymbol{z}\boldsymbol{z}^T \in \mathcal{S}} c_{\boldsymbol{z}} \boldsymbol{z}\boldsymbol{z}^T.$$

- The Latent Feature Lasso estimator:

$$\min_{M} \; g(M) + \lambda \|M\|_{\mathcal{S}}.$$

- Equivalently, one can solve the estimator by

$$\min_{\boldsymbol{c} \in \mathbb{R}_+^{|\mathcal{S}|}} \; g\,\Big(\sum_{k=1}^{2^N} c_k \boldsymbol{z}_k \boldsymbol{z}_k^T\Big) + \lambda \|\boldsymbol{c}\|_1$$

Question: How to optimize with $|\mathcal{S}| = 2^N$ variables?

Outline

Greedy Coordinate Descent via MAX-CUT

- At each iteration, we find the coordinate of steepest descent:

$$j^* = \underset{j}{argmax} \ -\nabla_j f(c) = \underset{z\in\{0,1\}^N}{argmax} \langle -\nabla g(M), zz^T \rangle$$

  which is a Boolean Quadratic problem that can be reformulated to MAX-CUT:

$$\max_{\boldsymbol{z}\in\{0,1\}^N} \boldsymbol{z}^T C \boldsymbol{z}$$

- Can be solved to a $3/5$-approximation (Nesterov) by rounding from a special type of SDP with $O(ND)$ iterative solver (Po-Wei Wangls 2016).

Greedy Coordinate Descent via MAX-CUT

0. $\mathcal{A} = \emptyset$, $c = 0$.

for $t = 1 \ldots T$ **do**

1. Find an approximate greedy atom zz^T by MAX-CUT-like problem:

$$\max_{z \in \{0,1\}^N} \langle -\nabla g(M), zz^T \rangle.$$

.

2. Add $\boldsymbol{z}\boldsymbol{z}^T$ to an active set $\mathcal{A}$.
3. Refine $\boldsymbol{c}_{\mathcal{A}}$ via Proximal Gradient Method on:

$$\min_{\boldsymbol{c} \geq 0} g\Big(\sum_{k \in \mathcal{A}} c_k \boldsymbol{z}_k \boldsymbol{z}_k^T\Big) + \lambda \|\boldsymbol{c}\|_1$$

4. Eliminate $\{\boldsymbol{z}_k \boldsymbol{z}_k^T | c_k = 0\}$ from $\mathcal{A}$.

end for.

- Evaluating $\nabla g(M)$ requires solving a least-square problem of cost $O(DK^2)$.
- Each iteration costs $\underbrace{O(ND)}_{\text{MAX-CUT}} + \underbrace{O(DK^2)}_{\text{Least-Square}}$

Outline

Convergence Analysis

Given any reference solution c^*, the t-th iteration of the Greedy Algorithm satisfies

$$F(c^t) - F(c^*) \leq \frac{2\gamma\|c^*\|_1^2}{\mu^2}\left(\frac{1}{t}\right) + \underbrace{\frac{2(1-\mu)}{\mu}\lambda\|c^*\|_1}_{\Delta(\lambda)},$$

$\mu = 3/5$ is the approximation ratio given by the MAX-CUT-like problem and γ is the Lipschitz-continuous parameter of $\nabla_j f(c)$.

- $\Delta(\lambda)$ decreases with N when λ is chosen to trade off between bias and variance.

Risk Analysis

Let the population risk of a dictionary W be

$$r(W) := E[\min_{z \in \{0,1\}^K} \frac{1}{2} \|\boldsymbol{x} - W^T \boldsymbol{z}\|^2].$$

Let W^* be an optimal dictionary of size K, the algorithm outputs $\hat{W}$ with bounds

$$r(\hat{W}) \leq r(W^*) + \epsilon$$

under probability $1 - \rho$ as long as

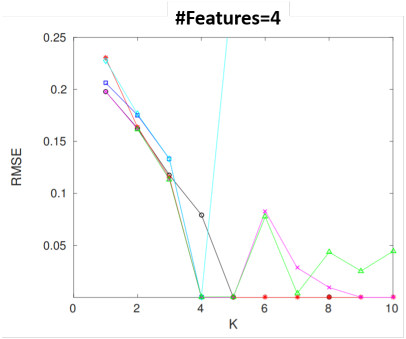
$$t = \Omega(\frac{K}{\epsilon}) \text{ and } N = \Omega(\frac{DK}{\epsilon^3} \log(\frac{RK}{\epsilon\rho})).$$

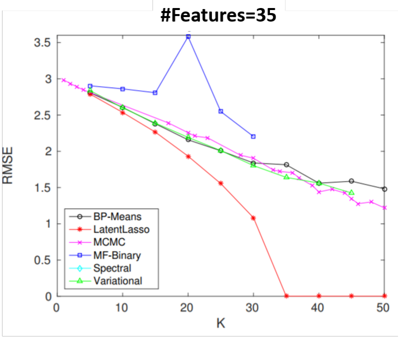
and λ τ chosen appropriately corresponding to N.

- The result trades between risk and sparsity.
- No assumption on $\boldsymbol{x}$ except that of boundedness.
- The sample complexity is (quasi) linear to D and K.

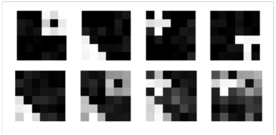
Outline

Results on Synthetic Data

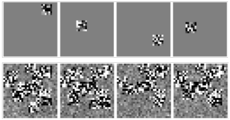




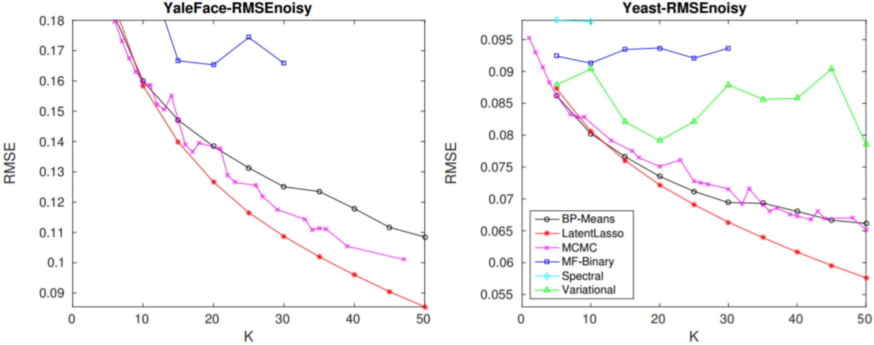
K_{True}=4



K_{True}=35



Results on Real Data



MCMC	Variational	MF-Binary	BP-Means	Spectral	LatentLasso
$(NDK^2)T$	$(NDK^2)T$	$(NK)2^K$	$(NDK^3)T$	$ND + K^5 log(K)$	$(ND + K^2D)T$

- MCMC, Variational, BP-Means take up to 1000s trainimg time, while LatentLasso takes up to 100s.

Conclusion

- In this work, we propose a novel convex estimator (Latent Feature Lasso) for the estimation of Latent Feature Model.
- To best of our knowledge, this is the first method with low-order polynomial runtime and sample complexity without restrictive assumptions on the data distribution.
- In experiments, the Latent Feature Lasso significantly outperforms other methods in terms of accuracy and time, when there is a larger number of latent features.

How to derive W and E and A

$$\min_{W \in \mathcal{R}^{K \times D}} \frac{1}{2N} \|X - Z_{\mathcal{A}} W\|_F^2 + \frac{\tau}{2} \|W\|_F^2$$

When $Z_{\mathcal{A}}$ is fixed, W has closed form solution

$$W^* = (Z_{\mathcal{A}}^T Z_{\mathcal{A}} + N\tau I)^{-1} Z_{\mathcal{A}}^T X$$

Given Lagrangian

$$L(W, E, A) = \frac{1}{2N} \|X - E\|_F^2 + \frac{\tau}{2} \|W\|_F^2 + < A, E - ZW >$$

$$0 \in \partial L \ (stationary)$$

$$E = ZW \ (primal\ \ feasibility)$$

Sub-gradients and Proximal Gradients Method

Danskin's theorem

Suppose $\phi(x, z)$ is a continuous function

$$\phi : \mathcal{R}^n \times Z \to \mathcal{R}$$

where $Z \in \mathcal{R}^m$ is a compact set, and $\phi(x, z)$ is convex in x for every $z \in Z$ define

$$f(x) = \max_{z \in Z} \phi(x, z)$$

$$Z_0(x) = \{\bar{z} : \phi(x, \bar{z}) = \max_{z \in Z} \phi(x, z)\}$$

then f(x) is convex and

$$D_y f(x) = \max_{z \in Z_0(x)} \phi^{'}(x, z; y)$$

Max-Cut Solver-1

We can replace $X \succeq 0$ constraints with $X = V^T V$ for some $V \in \mathcal{R}^{k \times n}$ then $X_{ii} = 1$ translates to $\|v_i\| = 1$ leads to non-convex optimization problem

$$\min_{V \in \mathcal{R}^{k \times n}} < C, V^T V > \ \ subject \ to \ \|v_i\| = 1, \ i = 1, ..., n$$

Solve it by coordinate descent method, minimizing v_i that depends on $v_i^T (\sum_{j=1}^{n} C_{ij} v_j)$ since $\|v_i\| = 1$ we can assume that $C_{ii} = 0$ without affecting the solution, so the problem is equivalent to

$$\min_{v_i \in \mathcal{R}^k} v_i^T g \ subject \ to \ \|v_i\| = 1$$

The solution has closed form

$$v_i = \frac{-g}{\|g\|}$$

Max-Cut Solver-2

This is from The Mixing method: coordinate descent for low-rank semi-definite programming (Po-Wei Wang)

Algorithm 1: The Mixing method

```
Initialize v_i randomly on a unit sphere;
while not yet converged do
    for i = 1, ..., n do
        v_i := normalize(−∑_{j=1}^n C_ij v_j);
    end
end
```

This way, we can initialize v_i on unit sphere and perform cyclic update over all the $i = 1, \dots, n$ in closed-form. We called it the mixing method, because for each v_i it mixes and normalizes the remaining vectors v_j according to weight C_{ij}. Thus, in the case of sparse C (which is the normal case for any large data problem) the time complexity for updating all variable once is $O(k\#\text{nnz})$, which is significantly cheaper than the interior point method. However, the details for efficient computation differ depending on the precise nature of the SDP, so we will describe these in more detail in the subsequent application sections. A complete description of the generic algorithm is shown in Algorithm 1.