Sublinear-Time Learning for High-Dimensional Models with Large Output Domains

Ian En-Hsu Yen

Machine Learning Department
Carnegie Mellon University

February 20, 2018

Outline

Outline

Extreme Classification

- Many problems involve large output domain where prediction and loss evaluation are very expensive.

	K=1 class	K$\approx 3 \times 10^5$ Classes
Train Time	25 sec.	$>$ 96 days
Model Size	2.7MB	$\approx$ 870GB

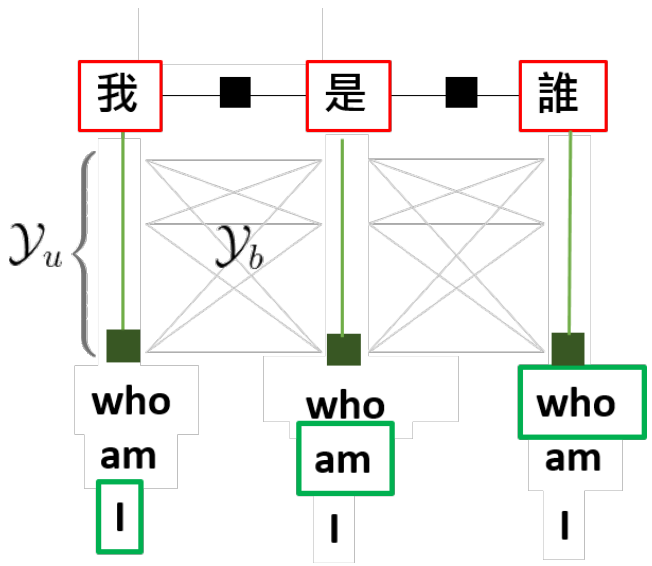
Table: Linear classifer on Wiki-LSHTC.

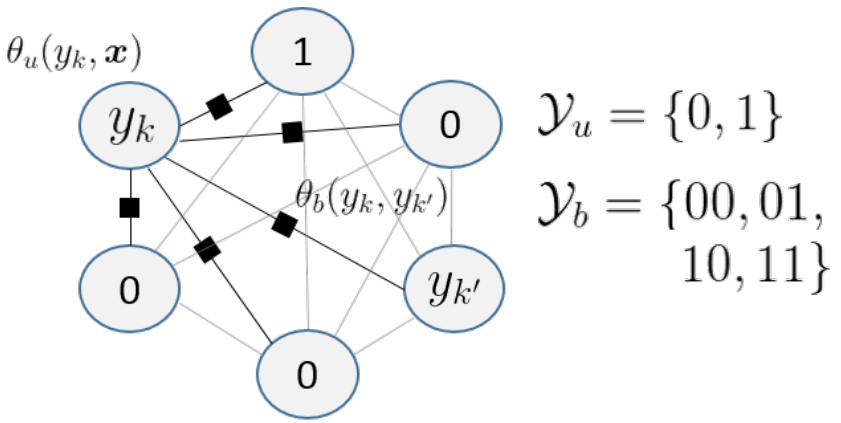
Wikipedia Dataset: $N \approx 10^6$, $K \approx 10^6$, $D \approx 10^6$

Megaface: $N \approx 10^6$, $K \approx 10^6$, $D \approx 10^3$

Extreme Structured Prediction

- Structured prediction and loss evaluation could have costs quadratic or cubic to the size of domain.





- Can we have training/prediction time, model size sublinear to the size of output domain K?

Approaches to Large Output Domains

- **Structural Assumption**: low-rank, cluster hierarchy, binary encoding.
  Issue: the structure might not holds $\Rightarrow$ lower accuracy.

Approaches to Large Output Domains

- **Structural Assumption**: low-rank, cluster hierarchy, binary encoding.

  Issue: the structure might not holds $\Rightarrow$ lower accuracy.

- **Sampling Approximation**: sample a small portion of classes for the evaluation of loss and its gradient.

  Issue: *classifier* confuses only few classes $\Rightarrow$ slow convergence.

Approaches to Large Output Domains

- **Structural Assumption**: low-rank, cluster hierarchy, binary encoding.

  Issue: the structure might not holds $\Rightarrow$ lower accuracy.

- **Sampling Approximation**: sample a small portion of classes for the evaluation of loss and its gradient.

  Issue: *classifier* confuses only few classes $\Rightarrow$ slow convergence.

- **Question:** Can we speed it up *without approximation*?

Loss of Concentration (Dual Sparsity)

- $X : N \times D$ are features or embeddings of last layer in a NeuralNet.
- $W : D \times K$ are weights of K classes
- The empirical loss is $L_Y(\Theta) = \sum_{i=1}^{N} \ell_{\boldsymbol{y}_i}(\Theta_{i,:})$, where $\Theta = XW$.
- Softmax Loss: $\ell_y(\boldsymbol{\theta}) := \log(\sum_{k=1}^{K} \exp(\theta_k)) - \theta_y$
- Max-Margin Loss: $\ell_{\boldsymbol{y}}(\boldsymbol{\theta}) := \left[\max_{k_n \in \mathcal{N}_{\boldsymbol{y}}} \theta_{k_n} - \min_{k_p \in \mathcal{P}_{\boldsymbol{y}}} \theta_{k_p} + 1\right]_+$

$$-\nabla L_Y(\Theta^*) = \overbrace{\left(\begin{array}{c} \\ A \\ \\ \end{array}\right)}^{k_A \text{ Support Labels}}_{N \times K}$$

Question: How to find those non-zero entries?

Approaches to Large Output Domains

- **Structural Assumption**: low-rank, clustered, hierarchy, binary encoding etc..

  Issue: the structure might not holds $\Rightarrow$ lower accuracy.

- **Sampling Approximation**: sample a small portion of classes for the evaluation of loss and its gradient.

  Issue: classifier confuses only few classes $\Rightarrow$ slow convergence.

- **Question:** Can we speed it up *without approximation*?

- **Learning by Search**: Find $k : \nabla_k L(\boldsymbol{\theta}) \neq 0$.

  1. Search via Sparse Matrix: $\boldsymbol{\theta} = W^T \boldsymbol{x}$.
  2. Search via Data Structures: $\underset{k}{argmax} \langle \boldsymbol{w}_k, \boldsymbol{x} \rangle$.

Joint Primal & Dual Sparsity

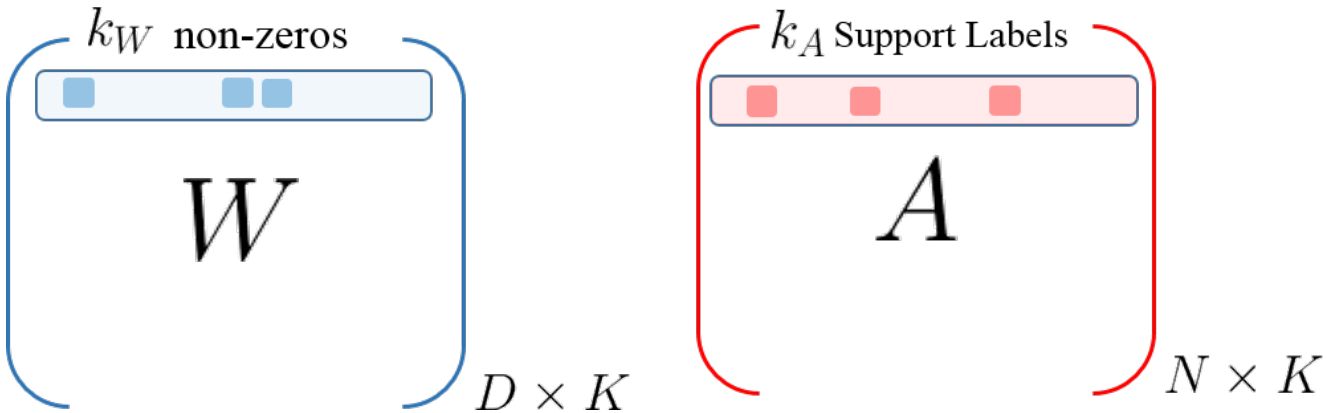
Theorem (Joint Primal & Dual Sparsity)

For any $\lambda > 0$ *and* X *in general position,*

$$W^* \in \underset{W}{\operatorname{argmin}}\ \lambda \|W\|_1 + L_Y(XW) \quad \text{and} \quad A^* \in -\nabla L_Y(\Theta^*)$$

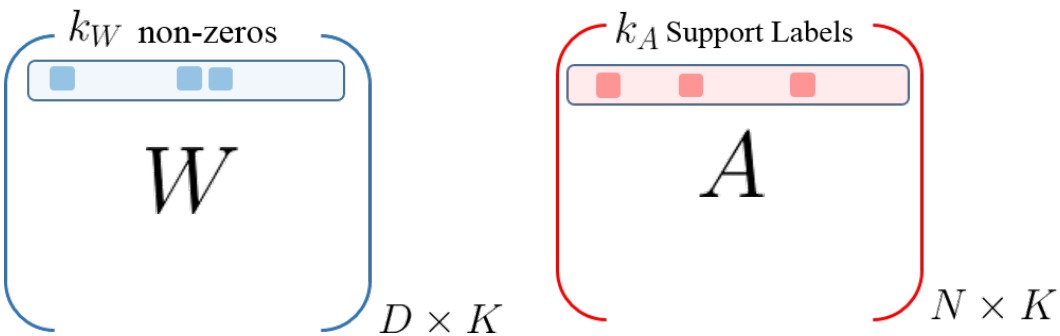
satisfy $\mathbf{Dk_W} = \mathbf{nnz(W^*)} \leq \mathbf{nnz(A^*)} = \mathbf{Nk_A}$.

- **Intuition:** $\mathbf{Nk_A}$ constraints vs **DK** variables at optimal $\Rightarrow$ under-determined.



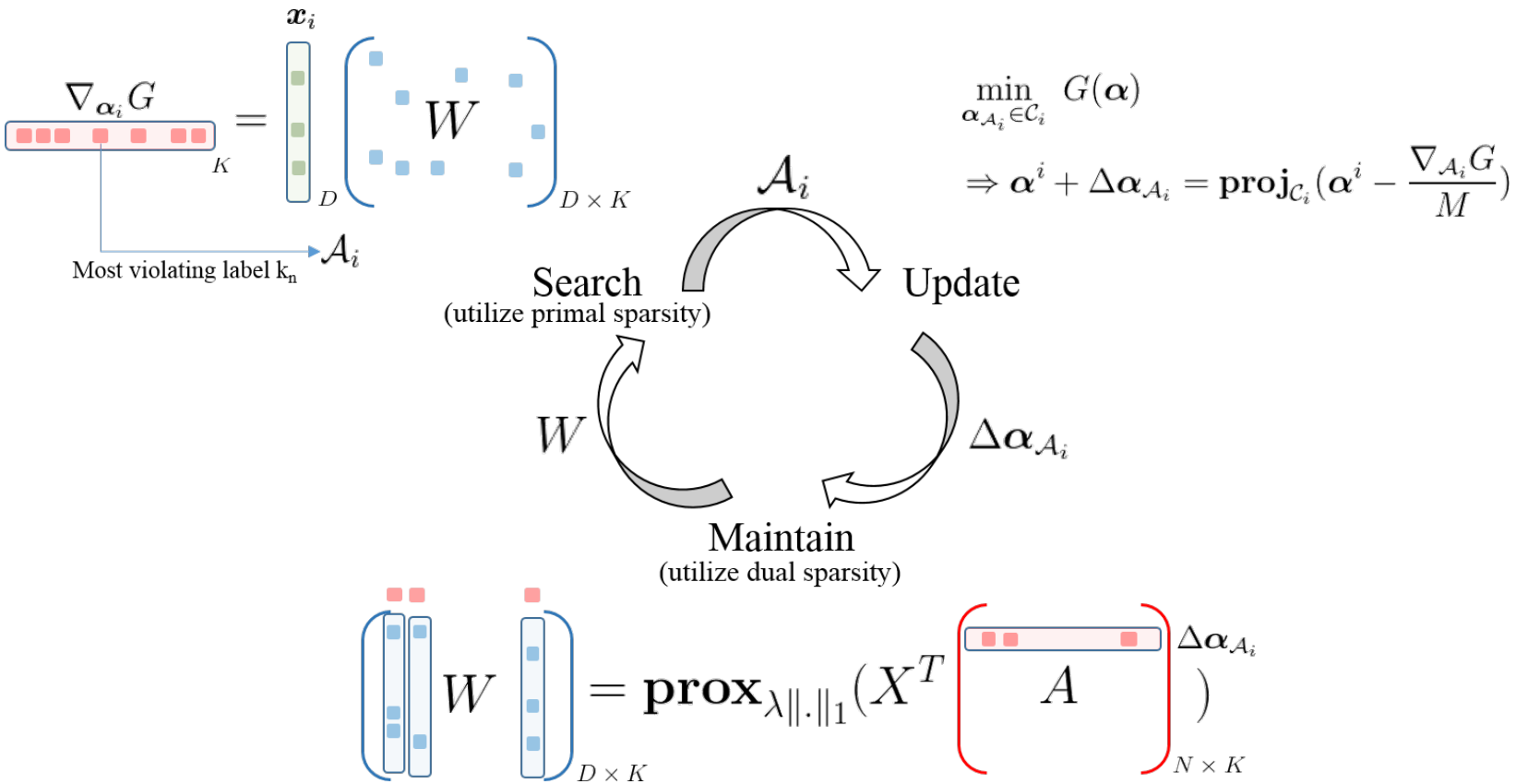
Search via Sparse Matrix

Data Set [1]	k_A: #support labels/sample	k_W: #nonzero/feature
bibtex (K=159)	18.17	1.94
aloi.bin (K=1,000)	3.24	0.31
EUR-Lex (K=3,956)	20.73	45.24
Dmoz (K=11,947)	5.87	0.116
LSHTC (K=12,294)	7.15	4.88
LSHTC-wiki (K=320,338)	18.24	20.95



[1] Yen et al. "PD-Sparse: A primal and dual sparse approach to extreme multiclass and multilabel classification. ICML 16.

Dual Block-Coordinate Frank-Wolfe (BCFW)



- Sparse $W \Rightarrow$ Forward pass $\nabla L(W^T \boldsymbol{x})$ in $O(nnz(\boldsymbol{x})k_W)$.
- Sparse $\nabla L \Rightarrow$ Backward pass $\Delta W \sim \mathbf{prox}_{\|\|_1}(-\boldsymbol{x} \times \nabla L^T)$ in $O(nnz(\boldsymbol{x})k_A)$.
- $\mathbf{O(nnz(X)k_W + nnz(X)k_A)}$ per epoch; $\mathbf{O(1/\epsilon)}$ passes for ϵ-suboptimality.

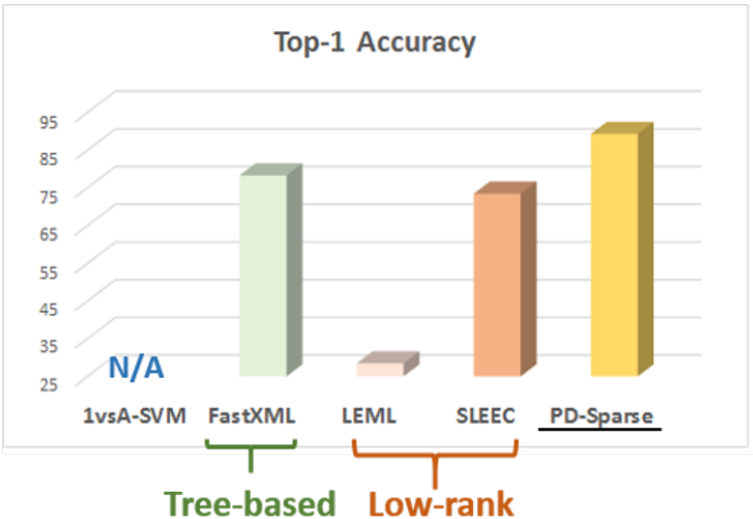
Results

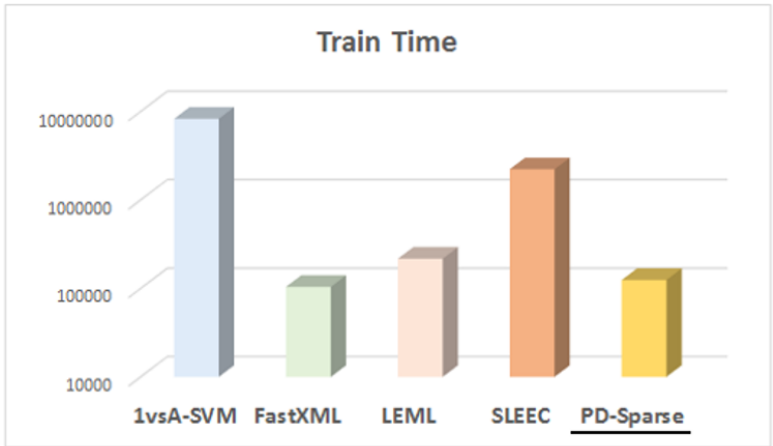
LSHTC-wiki

N= 2×10^6

D= 2×10^6

K= 3×10^5

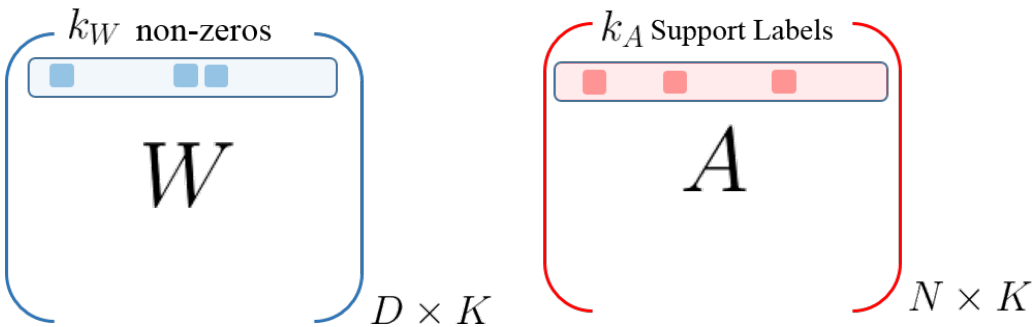




- **1-vs-All** takes $>$ 3 months to train (single CPU).
- **PD-Sparse** takes $\approx$ 1 day to train. The accuracy is higher by $\approx$ 10% than tree-based, low-rank, and local-embedding approaches.
- **PD-Sparse** predicts $\approx$ 100x faster than models from **1-vs-All**.
- (See more multiclass/multilabel results in (Yen et al. 2016).)

Algorithmic Choices

- **Dual BCFW**: better for maintaining dual sparsity.
- **Stochastic Gradient Descent**: better for non-convex objective.
- **Dual Greedy Coordinate Descent**: better for parallelization [2]
  (solve each column independently; only applicable to separable loss.)



(Bounds on $\mathbf{nnz}(\mathbf{W^t})$, $\mathbf{nnz}(\mathbf{A^t})$ are algorithm-dependent.)

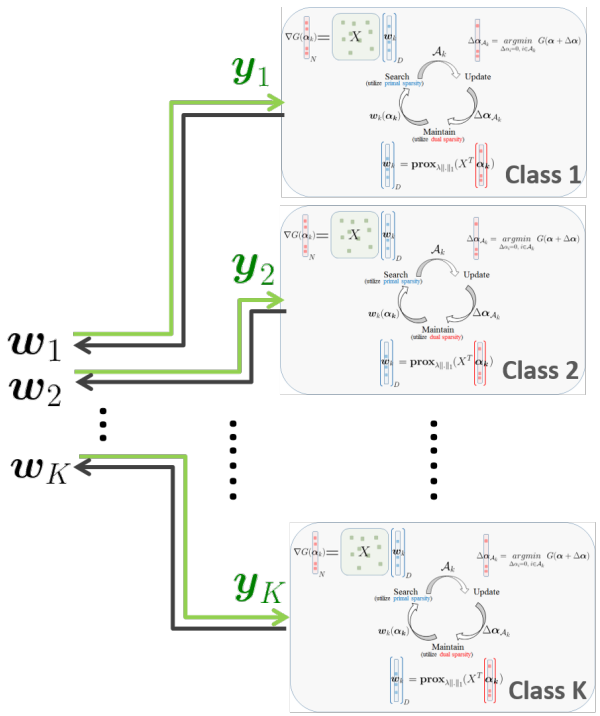
[2] Yen et al.. PPDSparse: A Parallel Primal-Dual Sparse Method for Extreme Classification. KDD 2017.

Algorithmic Choices

- **Dual BCFW**: better for maintaining sparsity.
- **Dual Greedy Coordinate Descent**: better for parallelization (solve each column independently; only applicable to separable loss.).

Acc/Time	Wiki (K=325k)	Amazon (K=670k)
PPD-Sparse	64.1 / 6min	43.0 / 15min
PD-Sparse	60.7 / 1day	N/A
Parallel-1vsA	64.0 / 3day	43.0 / 2day
FastXML	57.2 / 6hr	33.1 / 1.5hr
SLEEC	58.3 / 11hr	35.6 / 6hr

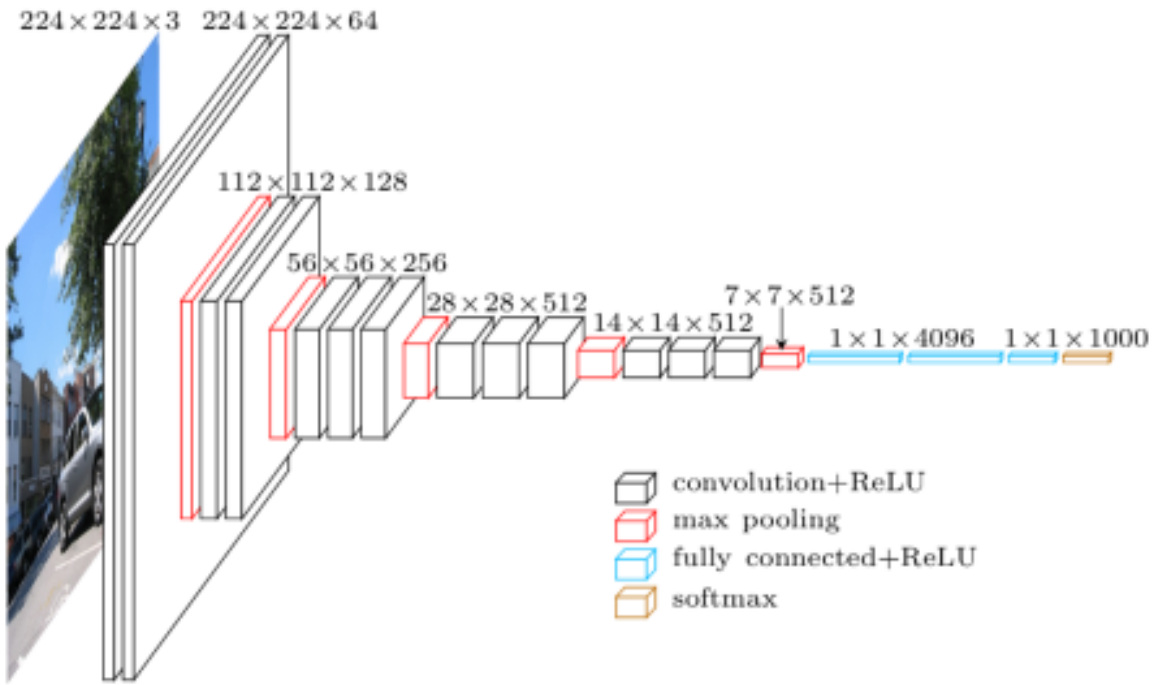
Table: **Parallel PD-Sparse** (100 cores). [3]



[3] Yen et al.. PPDSparse: A Parallel Primal-Dual Sparse Method for Extreme Classification. KDD 2017.

Outline

Sparsifying Neural Network



- A state-of-the-art DNN typically has $10^7 \sim 10^8$ parameters.
- **VGG16:** 138M parameters, with 102M parameters at the bottleneck layer.
- Many works try to sparsify DNNs, but how sparse? Is it data dependent?

Sparsifying Neural Network

Theorem (Sparse Neural Network)

Given a Neural Network $f_{W^*}(X)$ of loss L^*, for any $\epsilon > 0$ there is a sparse Neural Network $f_{\hat{W}}(X)$ of loss $L^* + \epsilon$ with

$$nnz(\hat{W}_{(l)}) \leq Nk_A, \ l \in layers.$$

where k_A is the number of penalized classes per sample.

$\hat{W}_{(l)}$ can be any stationary point got by minimizing

$$\lambda \|W_{(l)}\|_1 + L(f_W(X)), \ l \in layers$$

with initialization $W^*_{(l)}$ and some $\lambda(\epsilon) > 0$.

Sparsifying Neural Network

Theorem (Sparse Neural Network)

Given a Neural Network $f_{W^*}(X)$ *of loss* L^*, *for any* $\epsilon > 0$ *there is a sparse Neural Network* $f_{\hat{W}}(X)$ *of loss* $L^* + \epsilon$ *with*

$$nnz(\hat{W}_{(l)}) \leq Nk_A, \ l \in layers.$$

where k_A *is the number of penalized classes per sample.*

Empirical Results: (without sacrificing accuracy)

- **ImageNet** ($N \approx 10^6$, $k_A \approx 5$):
  - $dim(W) \approx 10^8 \Rightarrow nnz(\hat{W}) \approx 5 \times 10^6$ ($\approx 20x$ reduction).
- **CIFAR-10:** ($N = 5 \times 10^4$, $k_A \approx 0.1$):
  - $dim(W) \approx 10^8 \Rightarrow nnz(\hat{W}) \approx 4 \times 10^4$ ($\approx 2000x$ reduction !!)
  - (current best known result: 68x)

Outline

Extreme Classification via Data Structures

- For DNNs of embedding size $\mathbf{10^2 \sim 10^3}$, sparsity bound $\mathbf{nnz(W^*) \leq Nk_A}$ is not very useful ($\mathbf{DK \not\ll Nk_A}$).
- **Alternative:** *Maximum Inner Product Search (MIPS)*:

$$k^*(\boldsymbol{x}) = \underset{k}{\mathit{argmax}}\ [W\boldsymbol{x}]_k = \underset{k}{\mathit{argmax}}\ \boldsymbol{w}_k^T \boldsymbol{x}.$$

- Maintain vectors $\{\boldsymbol{w}_k\}_{k=1}^K$ in a data structure. Query by $\{\boldsymbol{x}_i\}_{i=1}^N$.
- Issue: high-recall MIPS is difficult in *high dimension*.

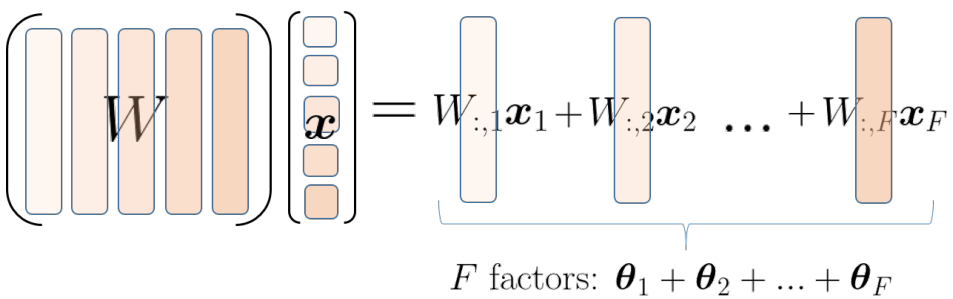
Extreme Classification via Data Structures

- **Alternative:** *Maximum Inner Product Search (MIPS)*:

$$k^*(\boldsymbol{x}) = \underset{k}{\mathit{argmax}}\ [W\boldsymbol{x}]_k = \underset{k}{\mathit{argmax}}\ \boldsymbol{w}_k^T \boldsymbol{x}.$$

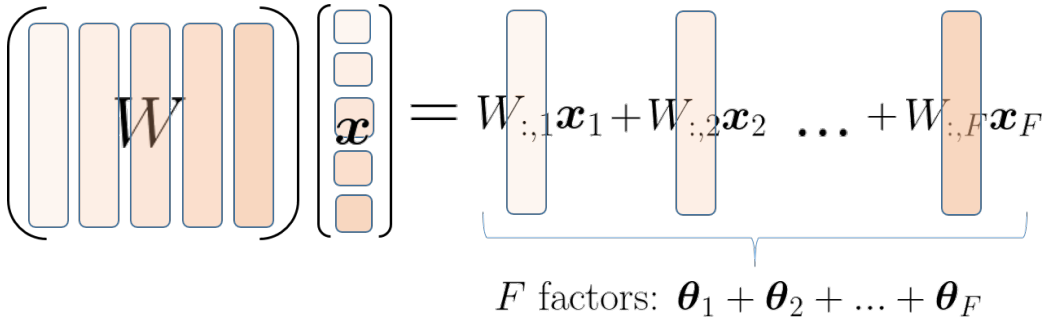
(Issue: high-recall MIPS is difficult in high dimension.)

- **To search accurately:** decompose high-dimensional MIPS into several low-dimensional MIPS with message passing.

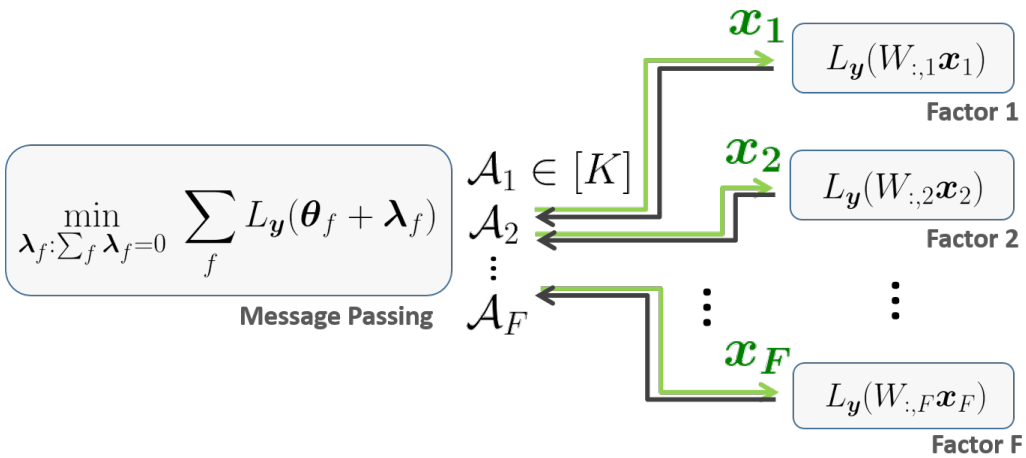


Extreme Classification via Data Structures

- **To search accurately:** decompose high-dimensional MIPS into several low-dimensional MIPS with message passing.



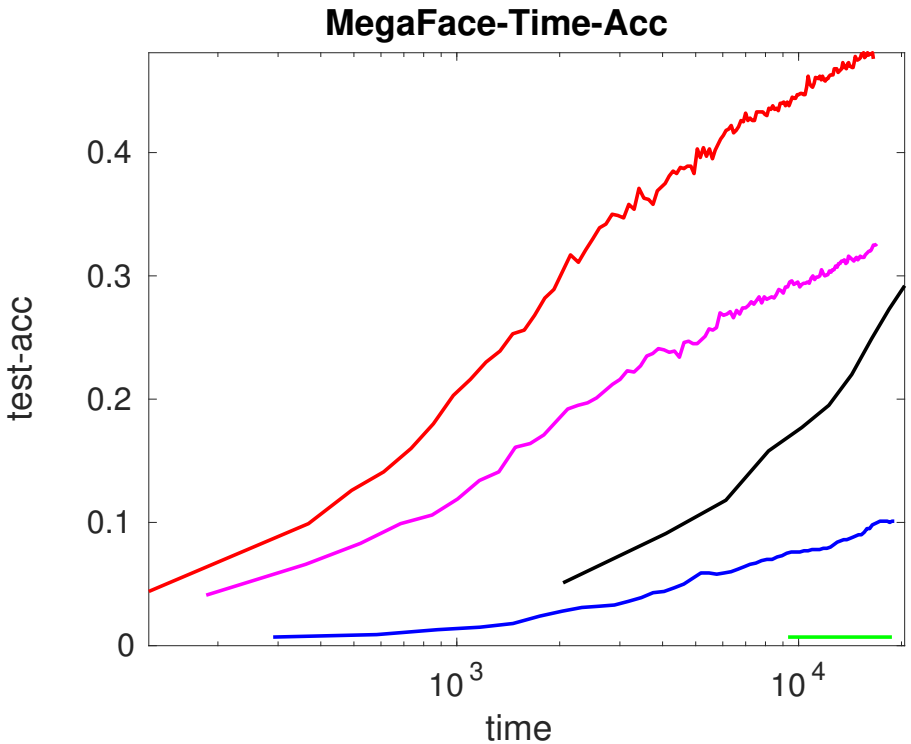
- **Key Insight:** $L\left(\sum_f \theta_f\right) = \min_{\lambda \in \Lambda} \sum_f L_f\left(\theta_f + \lambda_f\right).$



(Messages λ_f are sparse due to sparse $\nabla L_f(\theta_f)$.)

Experiments: Face Recognition

Data set	#Person	#Image	Embedding Size	#Image / Person
MegaFace	672k	4.7M	128 (**Facenet**)	≈ 7

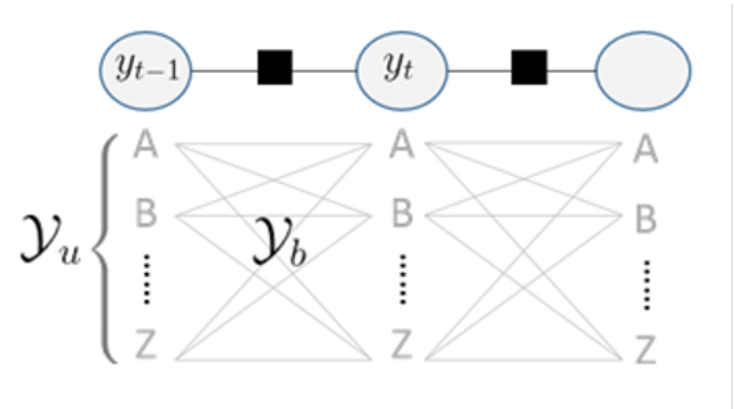


- Compare-1: **Max-Margin** vs. **MIPS** vs. **Dual-Decomp-MIPS**.
- Compare-2: **Max-Margin** vs. **Softmax** (and **Sampled-Softmax**).

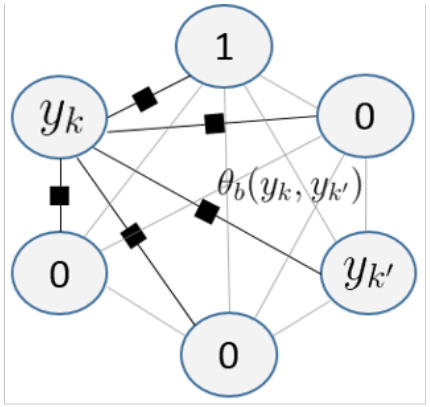
Outline

Structured Classifier with Large Output Domains

- What if there are multiple factors?
- Typically gradient evaluation takes $O(K^2)$ or more.



Sequence Labeling



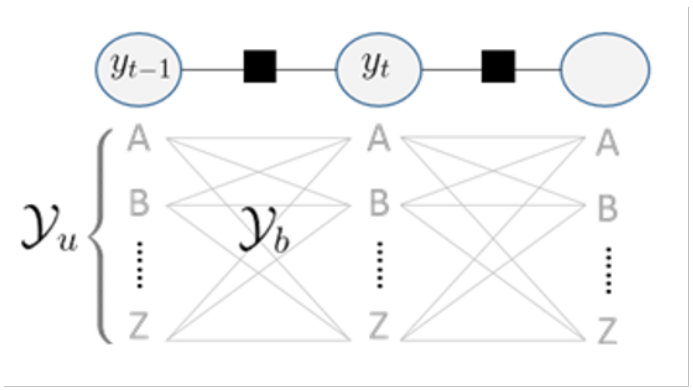
Multilabel w/ Correlation

Structured Classifier with Large Output Domains

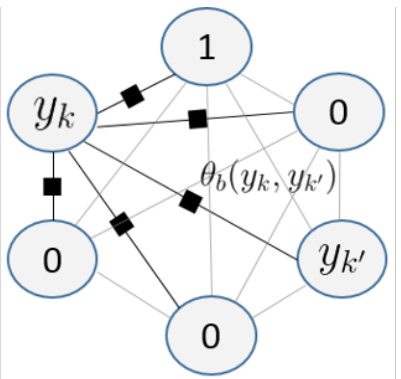
- Let $\boldsymbol{\theta}(\boldsymbol{w}) := \sum_f \boldsymbol{\theta}_f(\boldsymbol{w}) \in \mathbb{R}^{|\mathcal{Y}|}$ be the scores of each labeling $\boldsymbol{y} \in \mathcal{Y}$.
- **Key Insight:**

$$\min_{\boldsymbol{w}} L\left(\sum_{f \in F} \boldsymbol{\theta}_f(\boldsymbol{w})\right) = \min_{\boldsymbol{w}} \min_{\lambda \in \Lambda} \sum_{f \in F} L_f\left(\boldsymbol{\theta}_f(\boldsymbol{w}_f) + \sum_{j \in N(f)} \boldsymbol{\lambda}_{jf}\right)$$

where $\boldsymbol{\lambda}_{jf}$ are messages from variable j to factor f.



Sequence Labeling



Multilabel w/ Correlation

Structured Classifier with Large Output Domains

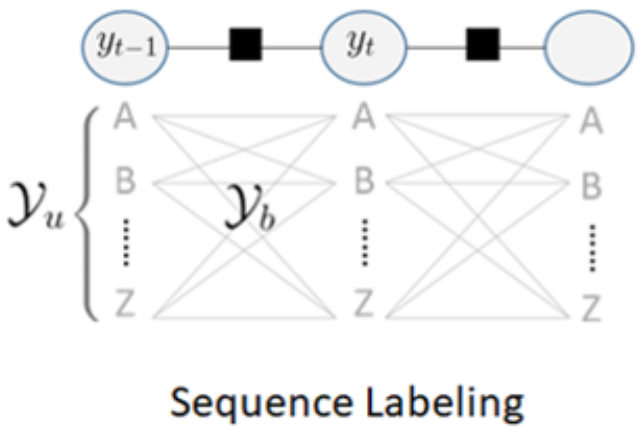
Key Insight: (λ_{jf} are messages from variable j to factor f.)

$$\underbrace{\min_{\boldsymbol{w}} \min_{\lambda \in \Lambda} \sum_{f \in F} L_f \left(\theta_f(\boldsymbol{w}_f) + \sum_{j \in N(f)} \lambda_{jf} \right)}_{\textbf{Inference}} = \underbrace{\min_{\lambda \in \Lambda} \min_{\boldsymbol{w}} \sum_{f \in F} L_f \left(\theta_f(\boldsymbol{w}_f) + \sum_{j \in N(f)} \lambda_{jf} \right)}_{\textbf{Multiclass Learning}}$$

- After solving multiclass, each factor's #active classes reduces from K to k_A, giving a sparse message λ_{jf} of size $O(k_A)$.

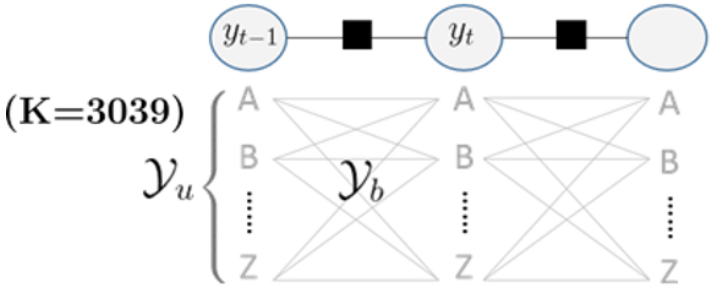
$$\textbf{Inference: } O(NK^2 + NdK)$$

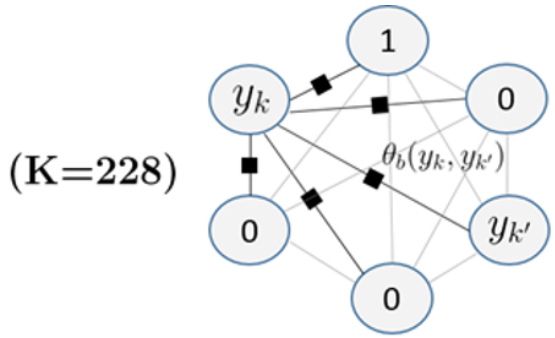
$$\textbf{multiclass: } O(Nk_A^2 + Nd\max(k_A, k_W))$$

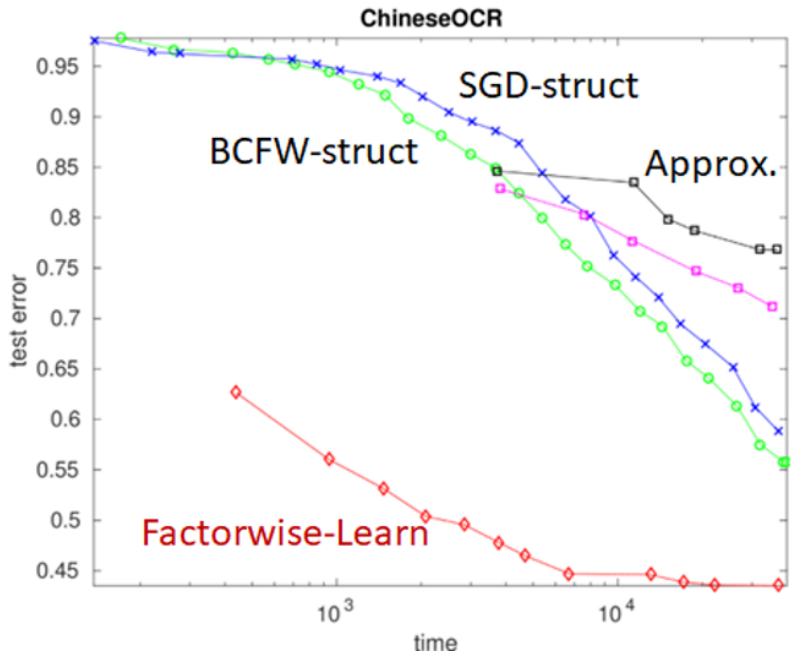


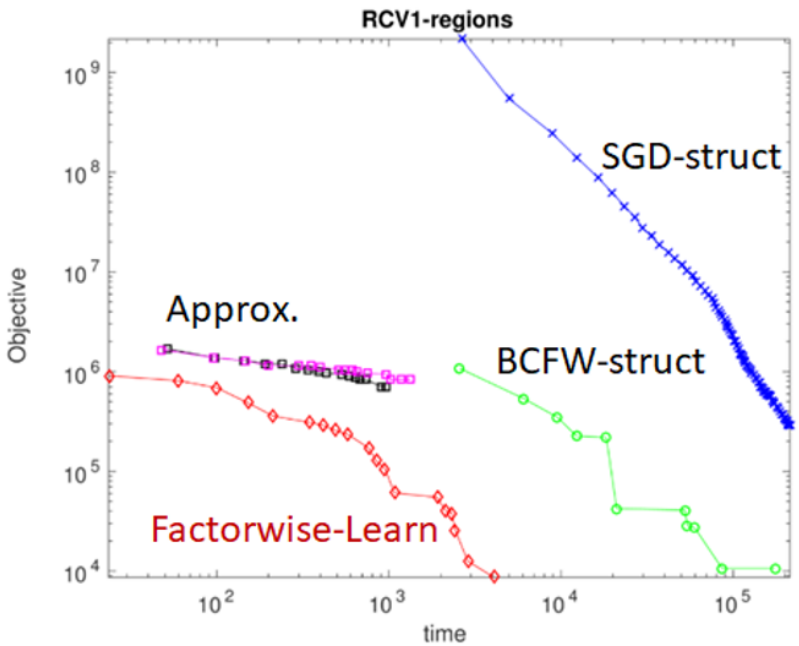
Sequence Labeling

Experiments: Sequence Labeling & Correlated Multilabel









[3]Yen. et al. Dual-Decomposed Learning with Factorwise Oracles for Structural SVMs of Large Output Domain. NIPS 2016.

Outline

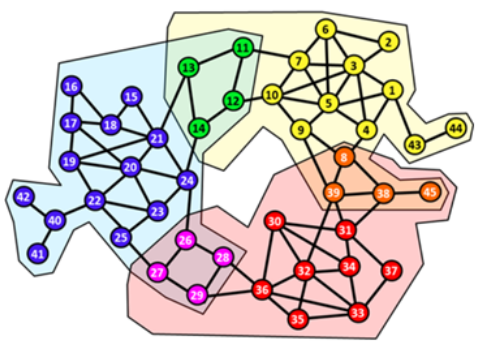
Binary Matrix Factorization (BMF)

- We are interested in finding factorization of the form:

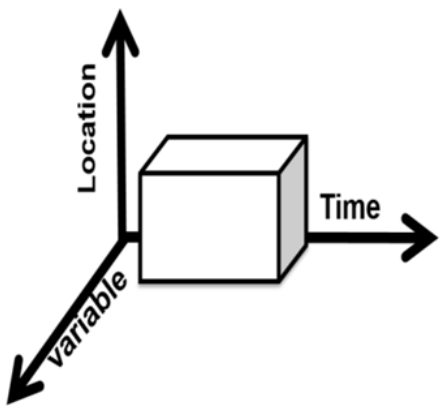
$$\min_{U,V} \; L(UV^T, R).$$

  where $U \in \{0,1\}^{m \times k}$, and $V \in \{0,1\}^{n \times k}$ or $V \in \mathbb{R}^{n \times k}$.

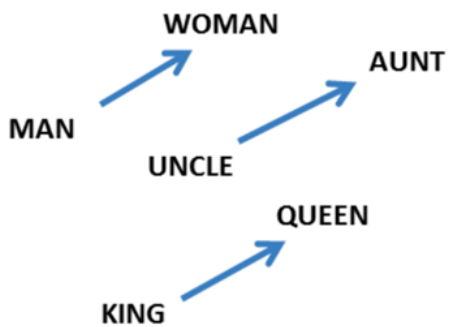
- Why binary? (interpretability, semi-supervision, and computational efficiency.)



Community Detection



Spatial-Temporal Factorization



Word Embedding

Binary Matrix Factorization (BMF)

- Let $U \in \{0,1\}^{m \times k}$, and $V \in \{0,1\}^{n \times k}$ or $V \in \mathbb{R}^{n \times k}$:

$$\min_{U,V} \; L(UV^T, R).$$

- **Existing Approaches:**
  - MCMC, Variational, Local Search (Indian Buffet Process):
    No finite-time guarantee.
  - Spectral Method:
    Requires knowledge of **data distribution**.
  - Algebraic Methods:
    Can handle only exact case $R = UV^T$, in $O(mk\mathbf{2^k})$.

- **Our Approach:**
  - Relaxation in very high-dimensional space.
  - First polynomial-time method of strong approximation guarantees.

BMF via Atomic Norm Minimization

- Let $\mathcal{S} := \{\boldsymbol{u}\boldsymbol{v}^T \mid \boldsymbol{u} \in \{0,1\}^m, \boldsymbol{v} \in \{0,1\}^n\}$.
- Define the Atomic Norm:

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

- Our model estimator is:

$$\min_{M} \; L_R(M) + \lambda \|M\|_{\mathcal{S}},$$

which is equivalent to:

$$\min_{\boldsymbol{c} \in \mathbb{R}_+^{|\mathcal{S}|}} \; L_R\left(\sum_{k=1}^{2^{m \times n}} c_k \boldsymbol{u}_k \boldsymbol{v}_k^T\right) + \lambda \|\boldsymbol{c}\|_1$$

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

Greedy Coordinate Descent via MAX-CUT

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

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

1. Find an approximate greedy atom $\boldsymbol{u}\boldsymbol{v}^T$ via solving MAX-CUT:

$$\max_{\boldsymbol{u} \in \{0,1\}^m, \boldsymbol{v} \in \{0,1\}^n} \langle -\nabla L_R(M), \boldsymbol{u}\boldsymbol{v}^T \rangle.$$

.

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

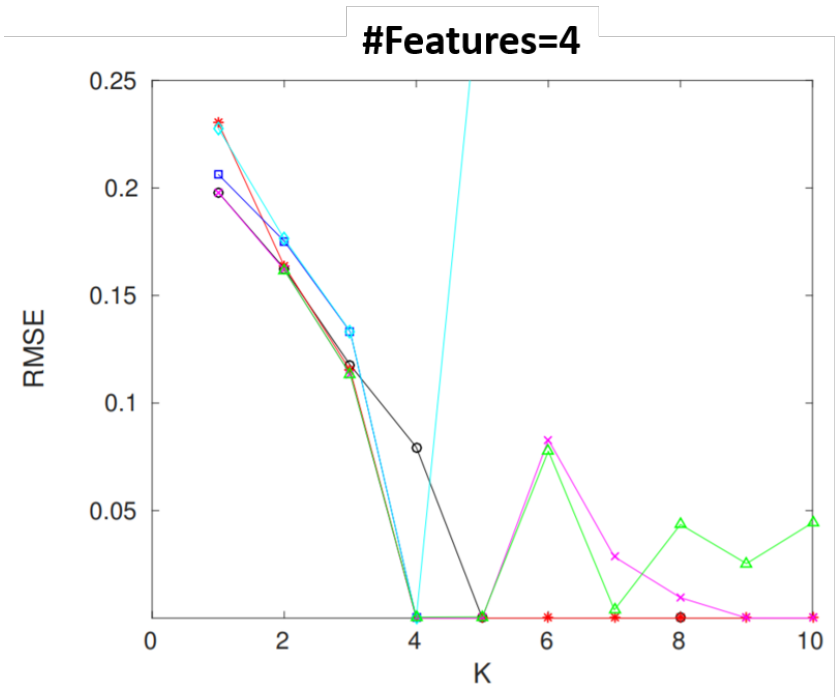
$$\min_{\boldsymbol{c} \geq 0} \; L_R(\sum_{k \in \mathcal{A}} c_k \boldsymbol{u}_k \boldsymbol{v}_k^T) + \lambda \|\boldsymbol{c}\|_1$$

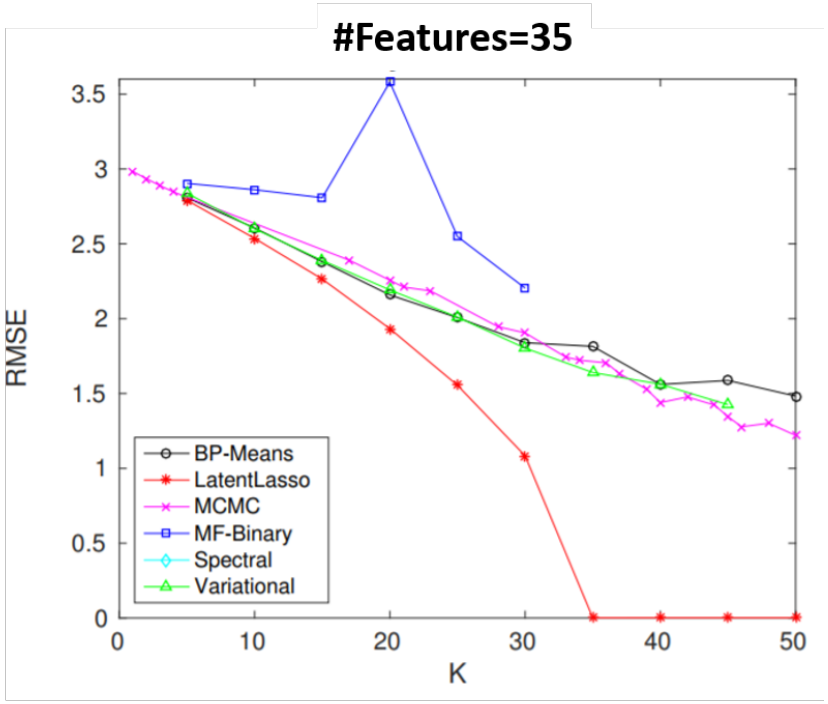
4. Eliminate $\{\boldsymbol{u}_k \boldsymbol{v}_k^T | c_k = 0\}$ from $\mathcal{A}$.

end for.

- MAX-CUT: 3/5-approximate solution in $O(nnz(\nabla L_R(M)))$.
- Running $\hat{k} = O(k^*/\epsilon)$ iterations, we have $L_R(USV^T) \leq L_R(U^*S^*V^{*T}) + \epsilon$ for any binary factorization $U^*S^*V^{*T}$ of rank k^*.

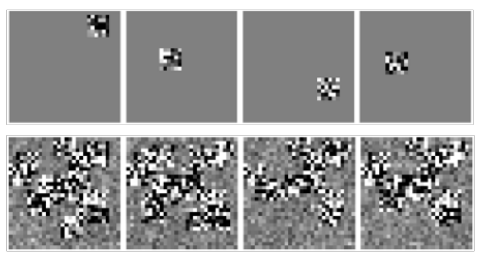
Results: Signal Decomposition Task





$K_{True}=4$

$K_{True}=35$



[3] See more results in: Yen et al. Latent Feature Lasso. ICML 2017.

Conclusion

- Many high-dimensional models have sparse structures, allowing us to:
  - Learn *Extreme Classifier* in sublinear time.
  - *Sparsify Neural Network*.
- Decomposition with sparse messages allows us to:
  - Handle large output domains for *Structured Predictor*.
  - Apply *Data Structures* effectively in learning.
- Ability to handle exponential number of variables allows us to design better estimators for some latent-variable models:
  - *Binary Matrix Factorization*, *Multiple Alignment*, *Motif Finding*, and *MAD-Bayes clustering*.

Appendix: Greedy Coordinate Descent via MAX-CUT

- The objective is:

$$\min_{\boldsymbol{c} \in \mathbb{R}_+^{|\mathcal{S}|}} \; f(\boldsymbol{c}) + \lambda \|\boldsymbol{c}\|_1$$

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

$$j^* = \underset{j}{argmax} \; -\nabla_j f(c) = \underset{\boldsymbol{u} \in \{0,1\}^m, \boldsymbol{v} \in \{0,1\}^n}{argmax} \langle -\nabla L_R(M), \boldsymbol{u}\boldsymbol{v}^T \rangle \quad (1)$$

  that reduces to a problem similar to MAX-CUT:

$$\max_{\boldsymbol{u} \in \{0,1\}^m, \boldsymbol{v} \in \{0,1\}^n} \boldsymbol{u}^T C \boldsymbol{v} = \frac{1}{2} \begin{bmatrix} \boldsymbol{u} \\ \boldsymbol{v} \end{bmatrix}^T \begin{bmatrix} O & C \\ C & O \end{bmatrix} \begin{bmatrix} \boldsymbol{u} \\ \boldsymbol{v} \end{bmatrix}$$

- It has 3/5-approximate algorithm by rounding from a special type of SDP with $O(nnz(C))$ iterative solver.

Appendix: Binary Factor + Real-Valued Factor

- What if $V \in \mathbb{R}^{n\times k}$?

$$\min_{U\in\{0,1\}^{m\times k}} \left\{ \min_{V\in\mathbb{R}^{n\times k}} L_R(UV^T) + \frac{\tau}{2}\|V\|_F^2 \right\},$$

- Given U, the dual problem w.r.t. V is:

$$\min_{M=UU^T\in\{0,1\}^{m\times m}} \underbrace{\left\{ \max_{A\in\mathbb{R}^{m\times n}} -\frac{1}{2\tau} tr(AA^T M) - L_R^*(-A) \right\}}_{\tilde{L}_R(M)}.$$

- Define the Atomic Set: $\mathcal{S} := \{\boldsymbol{u}\boldsymbol{u}^T \mid \boldsymbol{u} \in \{0,1\}^m\}$. We have estimator:

$$\min_M \tilde{L}_R(M) + \lambda\|M\|_{\mathcal{S}}.$$