Stochastic Optimization for Coupled Tensor Decomposition with Applications in Statistical Learning

Shahana Ibrahim, Xiao Fu

IEEE Data Science Workshop 2019

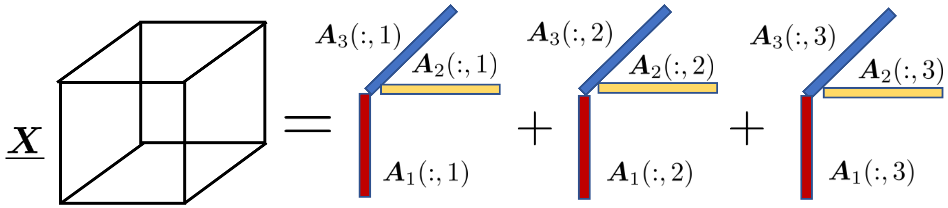
School of Electrical Engineering and Computer Science,
Oregon State University, Corvallis, USA

Outline

- **Motivation.**
- **Problem Modeling.**
- **Existing Approaches.**
- **Proposed Algorithm.**
- **Experimental Results.**
- **Conclusion.**

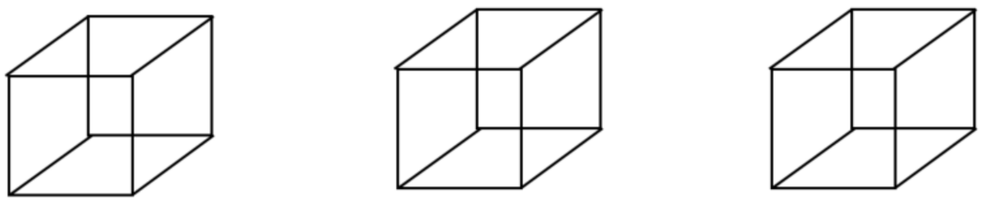
Coupled Tensor Decomposition

- *Canonical Polyadic Decomposition (CPD)* aims at factoring an K-way tensor $\underline{\boldsymbol{X}}$ with rank F to its latent factors, represented as $\underline{\boldsymbol{X}} = [\![\boldsymbol{A}_1, \ldots, \boldsymbol{A}_K]\!]$.



3-way tensor $\underline{\boldsymbol{X}}$ with $F = 3$, $\underline{\boldsymbol{X}} = [\![\boldsymbol{A}_1, \boldsymbol{A}_2, \boldsymbol{A}_3]\!]$.

- *Coupled Tensor Decomposition* simultaneously performs CPD to a number of tensors $\underline{\boldsymbol{X}}_1, \ldots, \underline{\boldsymbol{X}}_N$ that share some of the latent factors.



$\underline{\boldsymbol{X}}_1 = [\![\boldsymbol{A}_1, \boldsymbol{A}_2, \boxed{\boldsymbol{A}_3}]\!]$ $\quad$ $\underline{\boldsymbol{X}}_2 = [\![\boxed{\boldsymbol{A}_3}, \boldsymbol{A}_4, \boxed{\boldsymbol{A}_5}]\!]$ $\quad$ $\underline{\boldsymbol{X}}_3 = [\![\boxed{\boldsymbol{A}_5}, \boldsymbol{A}_6, \boldsymbol{A}_7]\!]$

Applications of Coupled Tensor Decomposition

- **Joint PMF Learning.**
  - Joint PMF estimation is of great interest in statistical learning applications such as classification, recommender systems etc.
  - In general, joint PMF learning is a hard problem due to the very high dimensionality.
    * For e.g, if we have K random variables $Z_1, \ldots, Z_K$ each taking I_k values, then joint PMF $\Pr(Z_1 = i_1, \ldots, Z_K = i_K)$ is an estimation of $\prod_{k=1}^{K} I_k$ parameters.
  - In practice, it is impossible to directly estimate $\Pr(Z_1 = i_1, \ldots, Z_K = i_K)$ from sample averaging when K is large.
    * For large K, the probability of encountering any particular joint appearance of all the random variables is very negligible.

Joint PMF Learning

- Joint PMF $\Pr(Z_1 = i_1, \ldots, Z_K = i_K)$, where Z_k can take I_k different values can be represented as a Kth-order tensor $\underline{\boldsymbol{X}} \in \mathbb{R}^{I_1 \times \cdots \times I_K}$ with

$$\underline{\boldsymbol{X}}(i_1, \ldots, i_K) = \Pr(Z_1 = i_1, \ldots, Z_K = i_K).$$

- Since every tensor has a CPD representation, the joint PMF tensor $\underline{\boldsymbol{X}}$ can be written as [Kargas, et al. 2018],

$$\underline{\boldsymbol{X}}(i_1, \ldots, i_K) = \sum_{f=1}^{F} \prod_{k=1}^{K} \boldsymbol{\lambda}(f) \boldsymbol{A}_k(i_k, f), \quad \underline{\boldsymbol{X}} = [\![\boldsymbol{\lambda}, \boldsymbol{A}_1, \ldots, \boldsymbol{A}_K]\!].$$

where $\boldsymbol{\lambda} \in \mathbb{R}^F$, $\mathbf{1}^\top \boldsymbol{\lambda} = 1$, $\boldsymbol{\lambda} \geq \mathbf{0}$, $\boldsymbol{A}_k \in \mathbb{R}^{I_k \times F}$, $\mathbf{1}^\top \boldsymbol{A}_k = \mathbf{1}$ and $\boldsymbol{A}_k \geq \mathbf{0}$.

- Interestingly, CPD representation can be considered as naive Bayesian model w.r.t a latent random variable H,
  - $\boldsymbol{\lambda}$ can be the prior probability vector with $\boldsymbol{\lambda}(f) = \Pr(H = f)$,
  - $\boldsymbol{A}_k$ can be the conditional PMF matrix with $\boldsymbol{A}_k(i_k, f) = \Pr(Z_k = i_k | H = f)$.

Joint PMF Learning from Lower-order Marginals

- If we can estimate the latent factors $\lambda, \boldsymbol{A}_1, \ldots, \boldsymbol{A}_K$, the complete joint PMF matrix $\underline{\boldsymbol{X}}$ can be reconstructed.
  - If the tensor is of low rank, then the number parameters to be estimated is only around $\sum_{k=1}^{K} I_k F$.

- **How do we estimate the latent factors?**

Joint PMF Learning from Lower-order Marginals

- If we can estimate the latent factors $\lambda, \boldsymbol{A}_1, \ldots, \boldsymbol{A}_K$, the complete joint PMF matrix $\underline{\boldsymbol{X}}$ can be reconstructed.
  - If the tensor is of low rank, then the number parameters to be estimated is only around $\sum_{k=1}^{K} I_k F$.
- **How do we estimate the latent factors?**
  - Lower-order marginals of two or three random variables are much easier to estimate and is related to the joint PMF as,

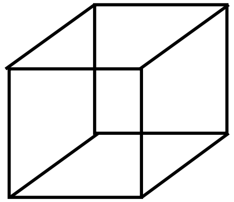
$$\mathsf{Pr}(Z_\ell = i_\ell, Z_m = i_m, Z_n = i_n) = \sum_{i_k \neq i_\ell, i_m, i_n} \sum_{i_k=1}^{I_k} \mathsf{Pr}(Z_1 = i_1, \ldots, Z_K = i_K)$$

    where $\mathsf{Pr}(Z_\ell = i_\ell, Z_m = i_m, Z_n = i_n)$ is the third-order marginal PMF.
  - [Kargas, et al. 2018] showed that given the joint PMFs of three random variables, the joint PMF of all the random variables can be provably recovered under mild conditions.

Joint PMF Learning from Lower-order Marginals

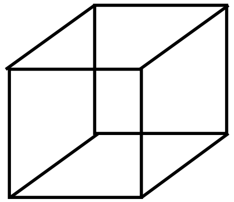
Marginalization | **Coupled Tensor Decomposition** | Reconstruction

$\Pr(Z_1 = i_1, \ldots, Z_7 = i_7)$



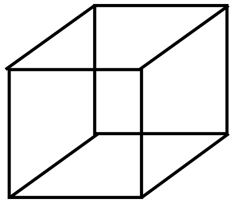
$\underline{\boldsymbol{X}}_{1,2,3} = [\![\boldsymbol{\lambda}, \boldsymbol{A}_1, \boldsymbol{A}_2, \boldsymbol{A}_3]\!]$

$\underline{\boldsymbol{X}}_{1,2,3}(i_1, i_2, i_3) = \Pr(Z_1 = i_1, Z_2 = i_2, Z_3 = i_3)$



$\underline{\boldsymbol{X}}_{3,4,5} = [\![\boldsymbol{\lambda}, \boldsymbol{A}_3, \boldsymbol{A}_4, \boldsymbol{A}_5]\!]$

$\underline{\boldsymbol{X}}_{3,4,5}(i_3, i_4, i_5) = \Pr(Z_3 = i_3, Z_4 = i_4, Z_5 = i_5)$



$\underline{\boldsymbol{X}}_{5,6,7} = [\![\boldsymbol{\lambda}, \boldsymbol{A}_5, \boldsymbol{A}_6, \boldsymbol{A}_7]\!]$

$\underline{\boldsymbol{X}}_{5,6,7}(i_5, i_6, i_7) = \Pr(Z_5 = i_5, Z_6 = i_6, Z_7 = i_7)$

$\hat{\Pr}(Z_1 = i_1, \ldots, Z_7 = i_7)$

$\underline{\boldsymbol{X}} = [\![\boldsymbol{\lambda}, \boldsymbol{A}_1, \ldots, \boldsymbol{A}_7]\!]$

Coupled Tensor Decomposition

- Joint PMF can be learned via coupled tensor decomposition of the third order marginals using the below optimization problem [Kargas, et al. 2018],

$$\begin{aligned} \underset{\{\boldsymbol{A}_k\}_{k=1}^{K}, \boldsymbol{\lambda}}{\text{minimize}} \; & \sum_{\ell=1}^{K} \sum_{m=\ell+1}^{K} \sum_{n=m+1}^{K} \left\| \underline{\boldsymbol{X}}_{\ell,m,n} - [\![\boldsymbol{\lambda}, \boldsymbol{A}_\ell, \boldsymbol{A}_m, \boldsymbol{A}_n]\!] \right\|_F^2 \\ \text{subject to } & \mathbf{1}^\top \boldsymbol{A}_k = \mathbf{1}^\top, \; \boldsymbol{A}_k \geq \mathbf{0}, \; \forall k \\ & \mathbf{1}^\top \boldsymbol{\lambda} = 1, \; \boldsymbol{\lambda} \geq \mathbf{0}. \end{aligned}$$

- [Traganitis, et al. 2018] proposed a similar model and coupled tensor decomposition formulation for another popular statistical learning, ie. **crowsourcing**.

Coupled Tensor Decomposition

- Joint PMF can be learned via coupled tensor decomposition of the third order marginals using the below optimization problem [Kargas, et al. 2018],

$$
\begin{aligned}
\underset{\{\boldsymbol{A}_k\}_{k=1}^{K},\boldsymbol{\lambda}}{\text{minimize}} \; & \sum_{\ell=1}^{K} \sum_{m=\ell+1}^{K} \sum_{n=m+1}^{K} \left\| \underline{\boldsymbol{X}}_{\ell,m,n} - [\![\boldsymbol{\lambda}, \boldsymbol{A}_\ell, \boldsymbol{A}_m, \boldsymbol{A}_n]\!] \right\|_F^2 \\
\text{subject to } & \mathbf{1}^\top \boldsymbol{A}_k = \mathbf{1}^\top, \; \boldsymbol{A}_k \geq \mathbf{0}, \; \forall k \\
& \mathbf{1}^\top \boldsymbol{\lambda} = 1, \; \boldsymbol{\lambda} \geq \mathbf{0}.
\end{aligned}
$$

- [Traganitis, et al. 2018] proposed a similar model and coupled tensor decomposition formulation for another popular statistical learning, ie. **crowsourcing**.

How do we design an algorithm that can efficiently handle very large number of coupled tensors?

Existing Approaches

- The algorithms proposed in [Kargas, et al. 2018,Traganitis, et al. 2018] handling coupled tensor decomposition has the following features
  - Cyclic updates for each of $\boldsymbol{A}_1, \ldots, \boldsymbol{A}_K, \boldsymbol{\lambda}$
  - In each update of $\boldsymbol{A}_k$, solving a subproblem using unfolded version of $\underline{\boldsymbol{X}}_{\ell,m,n}$,

$$\underset{\substack{\mathbf{1}^\top \boldsymbol{A}_k = \mathbf{1}^\top \\ \boldsymbol{A}_k \geq \mathbf{0}}}{\text{minimize}} \sum_{m \neq k} \sum_{\substack{n \neq k \\ n > m}} \|\boldsymbol{X}_{k,m,n}^{(1)} - (\boldsymbol{A}_n \odot \boldsymbol{A}_m)\boldsymbol{D}\boldsymbol{A}_k^\top\|_F^2, \tag{1}$$

  - For e.g, for updating $\boldsymbol{A}_1$, (1) may be written as,

$$\underset{\substack{\mathbf{1}^\top \boldsymbol{A}_1 = \mathbf{1}^\top \\ \boldsymbol{A}_1 \geq \mathbf{0}}}{\text{minimize}} \left\| \begin{bmatrix} \boldsymbol{X}_{1,2,3}^{(1)} \\ \boldsymbol{X}_{1,2,4}^{(1)} \\ \vdots \\ \boldsymbol{X}_{1,2,K}^{(1)} \end{bmatrix} - \begin{bmatrix} (\boldsymbol{A}_3 \odot \boldsymbol{A}_2)\boldsymbol{D} \\ (\boldsymbol{A}_4 \odot \boldsymbol{A}_2)\boldsymbol{D} \\ \vdots \\ (\boldsymbol{A}_K \odot \boldsymbol{A}_2)\boldsymbol{D} \end{bmatrix} \boldsymbol{A}_1^\top \right\|_F^2$$

  - Using an ADMM algorithm to handle the subproblems.

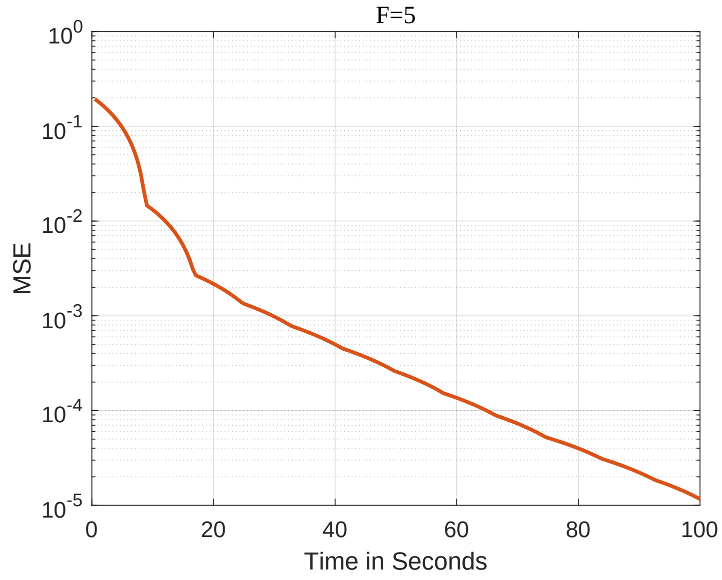
Existing Approaches

- **Challenges:**
  - Very large matrix-matrix product operation is to be performed in each step which substantially worsens the complexity.
  - This way, each step needs $O\left(\binom{K}{2} I_k I_m I_n F\right)$ flops $\implies O(I^5)$ if $K \approx I_\ell = I$.

Existing Approaches

- **Challenges:**
  - Very large matrix-matrix product operation is to be performed in each step which substatntially worsens the complexity.
  - This way, each step needs $O\left(\binom{K}{2} I_k I_m I_n F\right)$ flops $\implies O(I^5)$ if $K \approx I_\ell = I$.

Huge per-iteration complexity.



MSE of the latent factors when $K = 20, I_k = I = 15, F = 5$

Stochastic Optmization

- **Goal:** Accelerate coupled tensor decomposition using stochastic optimization.
- **How do we achieve this?**

Stochastic Optmization

- **Goal:** Accelerate coupled tensor decomposition using stochastic optimization.

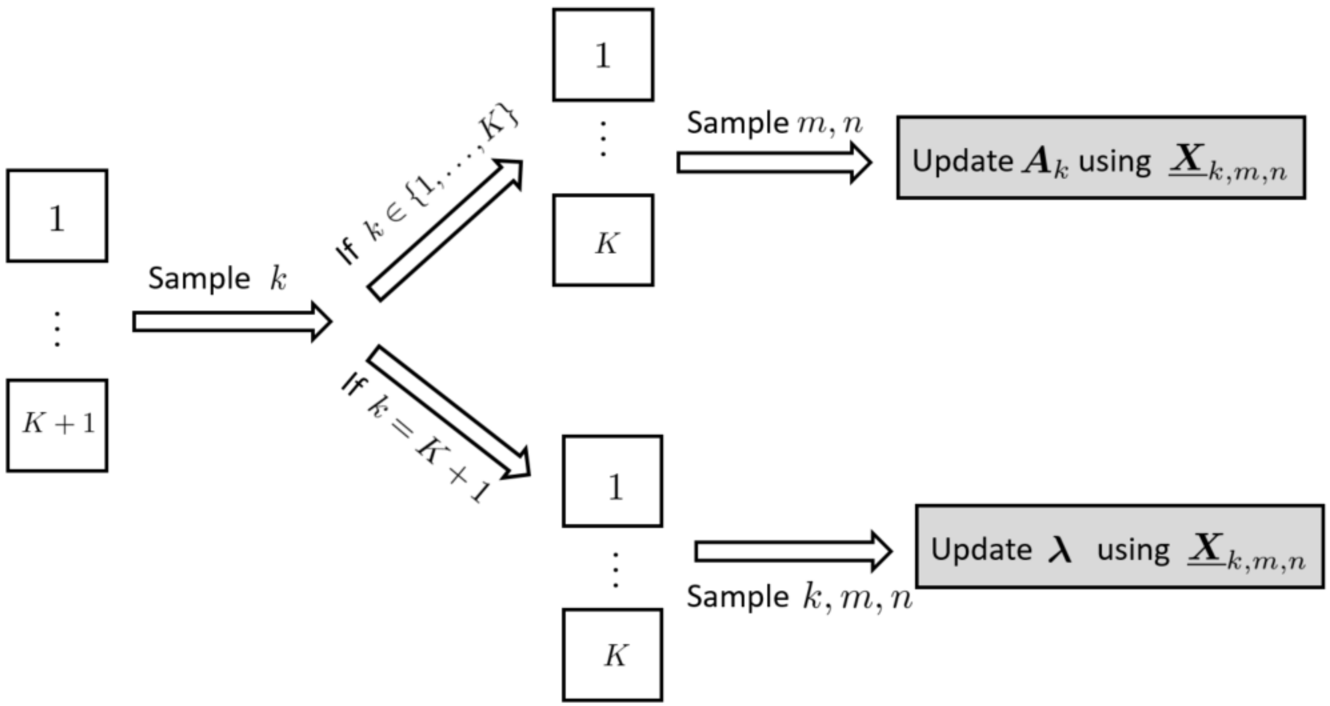
- **How do we achieve this?**

- An example of an existing idea used in [Vervliet, et al. 2016] is as below
  - Randomly sample part of the tensor, ie., a subtensor from $\underline{\boldsymbol{X}}$.
  - Then apply CPD to the sampled subtensor and update the latent factors.

- The idea of random sampling can be utilized in our case also, but
  - Do we need to run a complete CPD on the sampled tensor?
  - How do we ensure convergence of the algorithm involving random sampling?

Proposed Algorithm

- **Sampling strategy at each iteration t:**
  - Sample a block variable to update from $k \in \{1, \ldots, K, K+1\}$
  - Then, sample m, n from $\{1, \ldots, k-1, k+1, \ldots, K\}$ to update $\boldsymbol{A}_k$ or λ.



Proposed Algorithm

- **Update strategy at each iteration t:**
  - Assign

$$\boldsymbol{A}_k^{(t+1)} \leftarrow \mathsf{Proj}\left(\boldsymbol{A}_k^{(t)} - \alpha^{(t)}\boldsymbol{G}_k^{(t)}\right).$$

    where,

    * Stochastic Gradient, $\boldsymbol{G}_k^{(t)} = \boldsymbol{A}_k^{(t)}\boldsymbol{V}_k - (\boldsymbol{X}_{k,m,n}^{(1)})^\top \boldsymbol{H}_k$,
    * $\boldsymbol{H}_k = (\boldsymbol{A}_n \odot \boldsymbol{A}_m)\boldsymbol{D}$,
    * $\boldsymbol{V}_k = (\boldsymbol{\lambda}\boldsymbol{\lambda}^\top) \circledast (\boldsymbol{A}_n^\top \boldsymbol{A}_n) \circledast (\boldsymbol{A}_m^\top \boldsymbol{A}_m)$
    * Proj($\mathbb{Z}$) projects the columns of $\mathbb{Z}$ onto the probability simplex.
  - We also let $\boldsymbol{A}_j^{(t+1)} \leftarrow \boldsymbol{A}_j^{(t)}$, $\forall j \neq k$ and $\boldsymbol{\lambda}^{(t+1)} \leftarrow \boldsymbol{\lambda}^{(t)}$.
  - $\boldsymbol{\lambda}$ can also be updated in similar fashion.

Proposed Algorithm

- **Update strategy at each iteration t:**
  - Assign

$$\boldsymbol{A}_k^{(t+1)} \leftarrow \text{Proj}\left(\boldsymbol{A}_k^{(t)} - \alpha^{(t)}\boldsymbol{G}_k^{(t)}\right). \Longleftarrow \textbf{Stochastic Proximal Gradient (SPG)}$$

where,

    * Stochastic Gradient, $\boldsymbol{G}_k^{(t)} = \boldsymbol{A}_k^{(t)}\boldsymbol{V}_k - (\boldsymbol{X}_{k,m,n}^{(1)})^\top \boldsymbol{H}_k$,
    * $\boldsymbol{H}_k = (\boldsymbol{A}_n \odot \boldsymbol{A}_m)\boldsymbol{D}$,
    * $\boldsymbol{V}_k = (\boldsymbol{\lambda}\boldsymbol{\lambda}^\top) \circledast (\boldsymbol{A}_n^\top \boldsymbol{A}_n) \circledast (\boldsymbol{A}_m^\top \boldsymbol{A}_m)$
    * Proj$(\mathbb{Z})$ projects the columns of $\mathbb{Z}$ onto the probability simplex.
  - We also let $\boldsymbol{A}_j^{(t+1)} \leftarrow \boldsymbol{A}_j^{(t)}$, $\forall j \neq k$ and $\boldsymbol{\lambda}^{(t+1)} \leftarrow \boldsymbol{\lambda}^{(t)}$
  - $\boldsymbol{\lambda}$ can also be updated in similar fashion

Features of the Proposed Algorithm

- **Favourable run-time for large scale problems.**
  - Lightweight algorithm compared to [Kargas, et al. 2018, Traganitis, et al. 2018] since update step uses only a single tensor $\underline{\boldsymbol{X}}_{k,m,n}$.
  - We do not have very large matrix-matrix product operation in each iteration.

- **Constraints to $\boldsymbol{A}_k$ or $\boldsymbol{\lambda}$ can be applied**
  - Each sampled $\underline{\boldsymbol{X}}_{k,m,n}$ contains information about entire $\boldsymbol{A}_k$ which makes applying constraints possible.

Features of the Proposed Algorithm

- **Favourable run-time for large scale problems.**
  - Lightweight algorithm compared to [Kargas, et al. 2018,Traganitis, et al. 2018] since update step uses only a single tensor $\underline{\boldsymbol{X}}_{k,m,n}$.
  - We do not have very large matrix-matrix product operation in each iteration.

- **Constraints to $\boldsymbol{A}_k$ or $\boldsymbol{\lambda}$ can be applied**
  - Each sampled $\underline{\boldsymbol{X}}_{k,m,n}$ contains information about entire $\boldsymbol{A}_k$ which makes applying constraints possible.

- **What are the convergence gurantees?**

Convergence of the Proposed Algorithm

Proposition 1: Let,

$$f(\boldsymbol{\theta}) = \underset{\substack{\mathbf{1}^\top \boldsymbol{A}_k = \mathbf{1}^\top, \boldsymbol{A}_k \geq \mathbf{0} \\ \mathbf{1}^\top \boldsymbol{\lambda} = 1, \boldsymbol{\lambda} \geq \mathbf{0}}}{\text{minimize}} \sum_{\ell=1}^{K} \sum_{m=\ell+1}^{K} \sum_{n=m+1}^{K} \left\| \underline{\boldsymbol{X}}_{\ell,m,n} - [\![\boldsymbol{\lambda}, \boldsymbol{A}_\ell, \boldsymbol{A}_m, \boldsymbol{A}_n]\!] \right\|_F^2$$

where $\boldsymbol{\theta} = [\boldsymbol{\theta}_1^\top, \ldots, \boldsymbol{\theta}_K^\top]^\top$, $\boldsymbol{\theta}_k = \text{vec}(\boldsymbol{A}_k)$ for $k = 1, \ldots, K$ and $\boldsymbol{\theta}_{K+1} = \boldsymbol{\lambda}$. Let J_k denote the number of available tensors whose mode-1 factor is $\boldsymbol{A}_k$. Also let $\mathcal{B}^{(t)}$ be the filtration up to iteration $t-1$. Then, by uniform sampling of the tensors, the gradient computed at iteration t, $\boldsymbol{G}_k^{(t)}$ satisfies

$$\overline{\boldsymbol{G}}_k^{(t)} = \mathbb{E}\left[\boldsymbol{G}_k^{(t)} \mid \mathcal{B}^{(t)}\right] = C_k \nabla_{\boldsymbol{\theta}_k} f(\boldsymbol{\theta}), \ \forall k$$

where $C_k > 0$ is a certain constant.

Convergence Properties

- Proposition states that $\overline{\boldsymbol{G}}_k^{(t)}$ is a scaled version of the gradient of the objective function $f(\boldsymbol{\theta})$ taken w.r.t. ${\boldsymbol{A}_k}^{(t)}$.

- Two-stage sampling strategy results in SPG of $f(\boldsymbol{\theta})$ using an unbiased stochastic oracle $g^{(t)} = [\mathrm{vec}(\boldsymbol{G}_1^{(t)})^\top, \ldots, \mathrm{vec}(\boldsymbol{G}_{K+1}^{(t))})^\top]^\top$ in each iteration t.
  - all the convergence properties of the single-block SPG algorithm hold for the proposed algorithm.

- Another key consideration is stepsize scheduling for $\alpha^{(t)}$
  - SPG normally works under the Robbins-Monroe rule, i.e., $\sum_{t=0}^{\infty} \alpha^{(t)} = \infty$ and $\sum_{t=0}^{\infty} (\alpha^{(t)})^2 < \infty$.
  - In this work, we use the $Adagrad$ rule as proposed in [Fu, et al. 2018, Duchi, et al. 2011].

Experiments - Synthetic Data

- We consider a joint PMF recovery problem to the proposed method and the baseline `LS-BCD` [Kargas, et al. 2018].

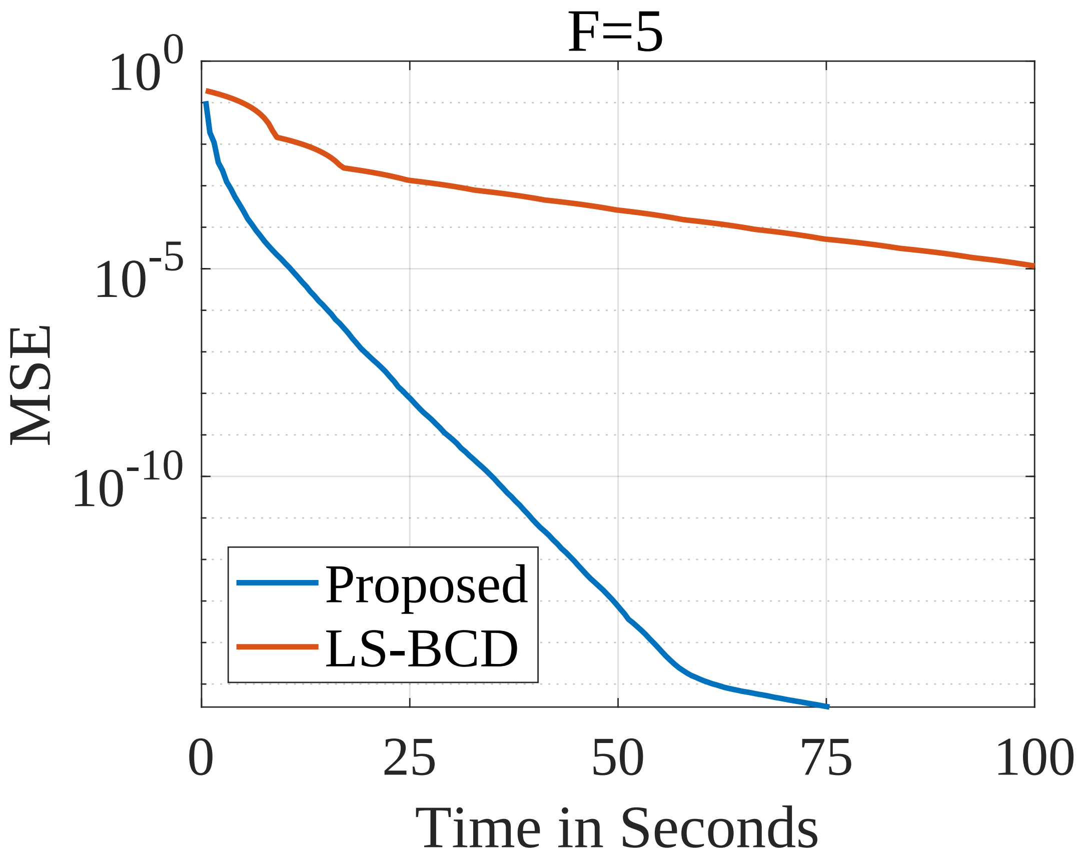
- We take $K = 20$ random variables with each variable taking $I_k = 15$ discrete values.

- The rank of the joint PMF tensor is set to different values $F \in \{5, 10\}$

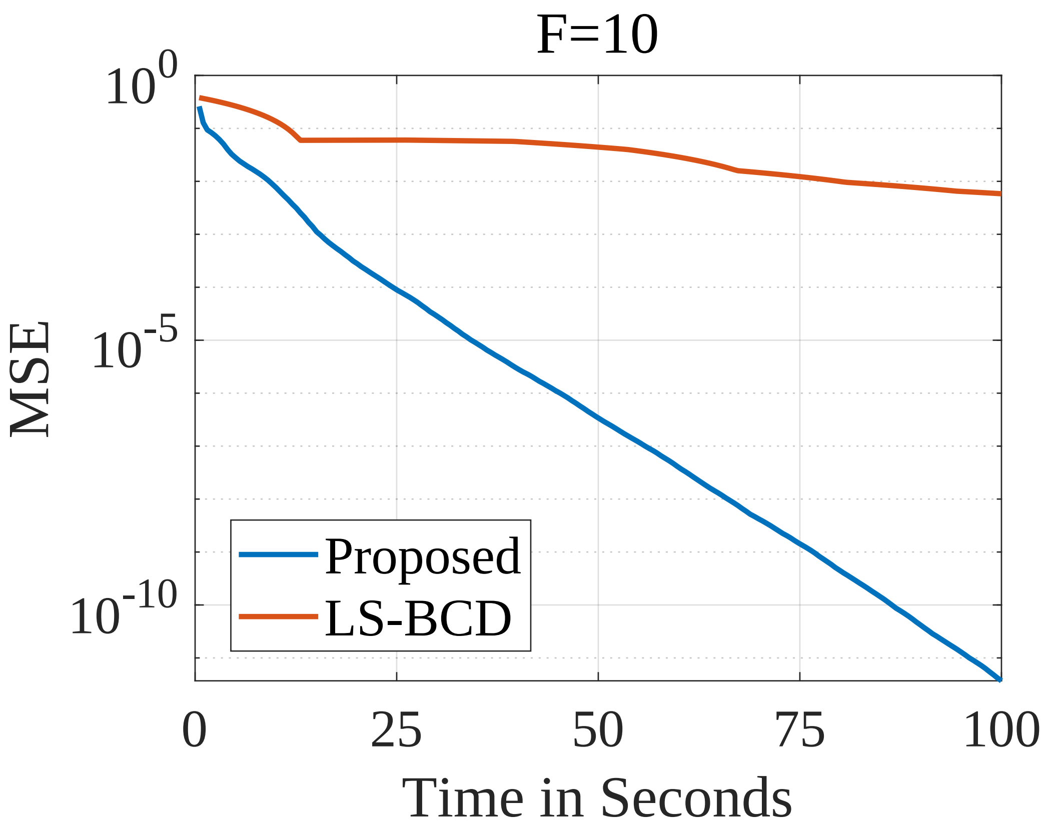
- The columns of true latent factors $\boldsymbol{A}_k \in \mathbb{R}^{I_k \times F}$ and $\boldsymbol{\lambda} \in \mathbb{R}^F$ are drawn from the probability simplex uniformly at random.

- The third order statistics of the random variables $\underline{\boldsymbol{X}}_{\ell,m,n} = [\![\boldsymbol{\lambda}, \boldsymbol{A}_\ell, \boldsymbol{A}_m, \boldsymbol{A}_n]\!]$, $\forall \ell, m, n \in [K]$ are generated using the true latent factors and used for estimation.

Synthetic Data Experiments



Synthetic Data Experiments



- The proposed algorithm outperforms the deterministic BCD algorithm `LS-BCD` in both accuracy and runtime by very large margin.

Real Data Experiments - Classification

- In this case, different UCI datasets[1] are used to evaluate the classification performance using the proposed method and the baselines `LS-BCD` [Kargas, et al. 2018] and `KL-BCD` [Kargas, et al. 2019]

- For each dataset, we run 10 Monte Carlo simulations by randomly partitioning the dataset into training, validation and testing sets.

- Using the training dataset, $\underline{\boldsymbol{X}}_{\ell,m,m}$ are estimated via counting the co-occurrences of the values taken by features ℓ, m and n.

- For each dataset, F is chosen by observing classification accuracy on the validation set.

- After identifying the parameters $\boldsymbol{A}_k$ and $\boldsymbol{\lambda}$, we use the *maximum a posteriori* (MAP) predictor to estimate the labels of the testing set.

[1] https://archive.ics.uci.edu/ml/datasets.html

Experiments-Real Data -Classification

Table 1: Real data-classification results using UCI dataset

	Misclassification(%)			Runtime(seconds)		
UCI Dataset (K, I_{avg}, F)	Proposed	LS-BCD	KL-BCD	Proposed	LS-BCD	KL-BCD
Nursery (9,4,15)	**0.086**	0.087	0.094	**2.98**	9.85	8.34
Car (7,4,15)	**0.097**	0.107	0.1068	**4.29**	14.79	7.725
Adult (15,14, 15)	**0.187**	0.247	†	**6.57**	48.49	†
Connect4 (22,7,15)	**0.338**	0.363	0.356	**6.54**	52.47	389.22
Credit (15,10,10)	**0.189**	0.347	0.254	**5.22**	40.02	30.51
Heart (9,3,10)	**0.198**	0.213	0.2113	**2.02**	8.87	8.11
Mushroom (21,6,15)	**0.042**	0.043	0.043	**8.19**	69.95	378.67
Voters (17,2,15)	**0.045**	0.076	0.053	**3.87**	27.44	27.64

† means the algorithm does not converge in 500 sec. and the result is not meaningful.

- The proposed algorithm outperforms the baselines in terms of accuracy and enjoys favourable run-time.

Experiments-Real Data -Crowdsourcing

- We use $K = 10$ different classification algorithms from the MATLAB machine learning toolbox to serve as annotators.

- Using 20% of the available data samples, each annotator is trained. Then, we allow the annotators to label the unseen data samples with probability p.

- Setting $p < 1$ is equivalent to the practical scenario where not all data samples are annotated by an annotator.

- Once the annotator responses are available, we estimate the co-occurrences of the annotator responses ℓ, m and n to obatin $\underline{\boldsymbol{X}}_{\ell,m,n}$.

- We perform 10 trials to take the average of the results and in each trial, a randomly selected testing set is labeled by the annotators with probability p.

- In our experiments, we set $p = 0.2$ for all annotators.

Experiments-Real Data -Crowdsourcing

- The classification accuracy of the proposed method is compared with different crowdsourcing algorithms such as `LS-ADMM` [Traganitis, et al. 2018] and `S-D&S` [Zhang, et al. 2014].

Table 2: Real data-crowdsouring results using UCI dataset

	Misclassification(%)			Runtime(seconds)		
UCI Dataset (K,F)	`Proposed`	`LS-ADMM`	`S-D&S`	`Proposed`	`LS-ADMM`	`S-D&S`
Adult (10,2)	**0.182**	0.258	0.238	**0.19**	4.17	2.10
Connect4(10,3)	**0.273**	0.344	0.333	**0.72**	50.96	14.38
Credit (10,2)	**0.166**	0.175	**0.166**	**0.18**	0.45	1.49
Mushroom (10,2)	**0.061**	0.064	**0.061**	**0.18**	0.44	2.40

- The proposed algorithm is very competitive in both accuracy and runtime.

Conclusion

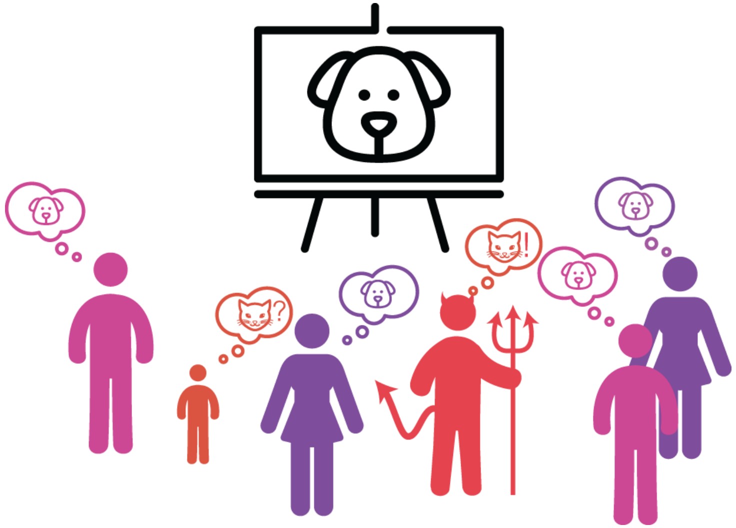
- In this work, we proposed a stochastic sampling and optimization strategy for coupled tensor decomposition tailored for statistical learning problems.
- The algorithm can handle a large number of latent factor-coupled tensors and can easily deal with a variety of constraints on the latent factors.
- The algorithm admits an interesting connection to the classic single-block stochastic proximal gradient scheme—thereby enjoying the same convergence properties.
- Simulations and real experiments showed that the proposed algorithm outperforms various existing algorithms devised for similar problems in both runtime and accuracy.

Thank You

Back up Slides

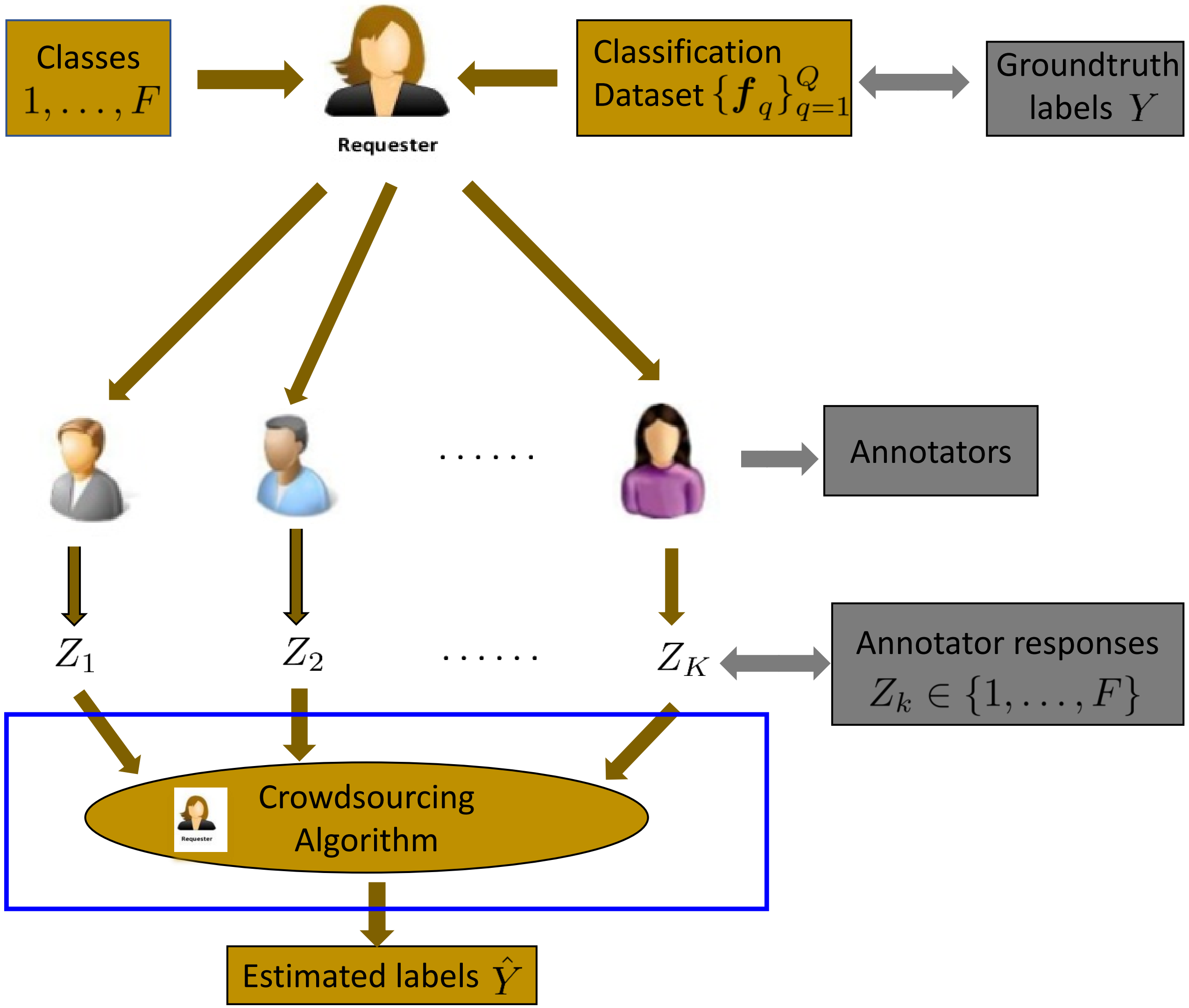
Applications of Coupled Tensor Decomposition

- **Crowdsourcing**
  - In machine learning, data labeling is oftentimes crowdsourced to multiple annotators for efficiency and robustness.



  - Since different annotators may create different labels for the same sample, an effective algorithm for result fusion is desired.

Crowdsourcing Dataflow



Crowdsourcing Model

- The classic model in crowdsourcing proposed by Dawid-Skene [1] is also a naive Bayesian model.

- ie., the responses of the annotators, $Z_1, \ldots, Z_K$ are conditionally independent given the true label Y,

$$\mathsf{Pr}(Z_1 = i_1, \ldots, Z_K = i_K) = \sum_{f=1}^{F} \mathsf{Pr}(Y = f) \prod_{k=1}^{K} \mathsf{Pr}(Z_k = i_k | Y = f),$$

  where $f \in \{1, \ldots, F\}$ represents the class label, and i_k denotes the response of the kth annotator.

Crowdsourcing Model

- The classic model in crowdsourcing proposed by Dawid-Skene [1] is also a naive Bayesian model.

- ie., the responses of the annotators, $Z_1, \ldots, Z_K$ are conditionally independent given the true label Y,

$$\underbrace{\mathsf{Pr}(Z_1 = i_1, \ldots, Z_K = i_K)}_{\underline{\boldsymbol{X}}(i_1,\ldots,i_K)} = \sum_{f=1}^{F} \underbrace{\mathsf{Pr}(Y = f)}_{\lambda(f)} \prod_{k=1}^{K} \underbrace{\mathsf{Pr}(Z_k = i_k | Y = f)}_{\boldsymbol{A}_k(i_k,f)},$$

where $f \in \{1, \ldots, F\}$ represents the class label, and i_k denotes the response of the kth annotator

Crowdsourcing Model

- Crowdsourcing problem admits the CPD model $\underline{\boldsymbol{X}} = [\![\boldsymbol{\lambda}, \boldsymbol{A}_1, \ldots, \boldsymbol{A}_K]\!]$. where the latent factors,
  - $\boldsymbol{\lambda}$ is called as the prior probability of true labels,
  - $\boldsymbol{A}_1, \ldots, \boldsymbol{A}_K$ are called as the confusion matrices.

- The estimation of true labels from the latent factors can be done by MAP predictor [2].

- Here also, jointly observing the responses of all annotators is hard, thereby estimating the joint co-occurences ($\underline{\boldsymbol{X}}_{\ell,m,n}$) of three annotator responses is a viable solution.

Crowdsourcing Model

- Crowdsourcing problem admits the CPD model $\underline{\boldsymbol{X}} = [\![\boldsymbol{\lambda}, \boldsymbol{A}_1, \ldots, \boldsymbol{A}_K]\!]$. where the latent factors,
  - $\boldsymbol{\lambda}$ is called as the prior probability of true labels,
  - $\boldsymbol{A}_1, \ldots, \boldsymbol{A}_K$ are called as the confusion matrices.
- The estimation of true labels from the latent factors can be done by MAP predictor [2].
- Here also, jointly observing the responses of all annotators is hard, thereby estimating the joint co-occurences ($\underline{\boldsymbol{X}}_{\ell,m,n}$) of three annotator responses is a viable solution.

Coupled decomposition of all available tensors $\underline{\boldsymbol{X}}_{\ell,m,n} = [\![\boldsymbol{\lambda}, \boldsymbol{A}_\ell, \boldsymbol{A}_m, \boldsymbol{A}_n]\!]$ can solve the crowdsourcing problem.

References

[1] A. P. Dawid and A. M. Skene, "Maximum likelihood estimation of observer error-rates using the em algorithm," *Applied Statistics*, pp. 20–28, 1979.

[2] P. Traganitis, A. Pages-Zamora, and G. B. Giannakis, "Blind multiclass ensemble classification," *IEEE Trans. Signal Process.*, vol. 66, pp. 4737–4752, Jul 2018.