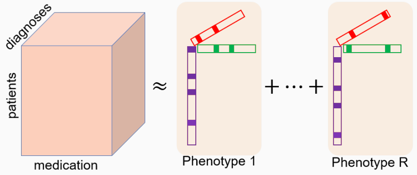
SWIFT: Scalable Wasserstein Factorization for Sparse Nonnegative Tensors

Ardavan Afshar[1] Kejing Yin[2] Sherry Yan[3] Cheng Qian[4] Joyce C. Ho[5]
Haesun Park[1] Jimeng Sun[6]

[1]Georgia Institute of Technology [2]Hong Kong Baptist University [3]Sutter Health [4]IQVIA
[5]Emory University [6]University of Illinois at Urbana-Champaign

Background: CP Tensor Factorization

CP factorization[1] approximates a tensor $\mathcal{X}$ as the sum of R rank-one tensors:



An example of CP factorization with input of a patient-diagnosis-medication tensor.

$$\boxed{\mathcal{X} \approx \hat{\mathcal{X}} = [\![\mathbf{A}^{(1)}, \mathbf{A}^{(2)}, ..., \mathbf{A}^{(N)}]\!] = \sum_{r=1}^{R} \mathbf{a}_r^{(1)} \circ \mathbf{a}_r^{(2)} \circ \ldots \circ \mathbf{a}_r^{(N)},}$$

- $\mathbf{A}^{(n)}$: Factor matrix for the n-th mode.
- $\mathbf{a}_r^{(n)}$: the r-th column of $\mathbf{A}^{(n)}$

- It is widely-used in various applications, *e.g.* healthcare data analytics.
- It is highly interpretable: each rank-one tensor can be treated as a latent factor.

[1] *Tamara G Kolda and Brett W Bader. "Tensor decompositions and applications". In: SIAM Review (2009).*

Motivation

Existing tensor factorization models assume certain distributions of input, for example:

- Gaussian distribution: $\min_{\widehat{\mathcal{X}}} ||\mathcal{X} - \widehat{\mathcal{X}}||_F^2$ $\leftarrow$ *MSE loss*[2]
- Poisson distribution: $\min_{\widehat{\mathcal{X}}} \widehat{\mathcal{X}} - \mathcal{X} * \log(\widehat{\mathcal{X}})$ $\leftarrow$ *KL divergence*[3]
- Bernoulli distribution: $\min_{\widehat{\mathcal{X}}} \log(1 + e^{\widehat{\mathcal{X}}}) - \mathcal{X} * \widehat{\mathcal{X}}$ $\leftarrow$ *logit loss*[4]

Do we always know the distribution of a given input tensor?

- Real-world data often have very complex distributions.
- We usually do not know the underlying distribution of the input tensor.

[2] *J Carroll and J Chang. "Analysis of individual differences in multidimensional scaling via an N-way generalization of "Eckart-Young" decomposition". In: Psychometrika (1970).*
[3] *E Chi and T Kolda. "On tensors, sparsity, and nonnegative factorizations". In: SIAM Journal on Matrix Analysis and Applications (2012).*
[4] *D Hong, T Kolda, and J Duersch. "Generalized canonical polyadic tensor decomposition". In: SIAM Review (2020).*

Motivation

Instead of assuming a specific distribution, the **Wasserstein distance** can be an alternative.

- *a.k.a.* Earth Mover Distance (EMD);
- is a potentially better measure of the difference between two distributions;
- does not assume any particular distributions of input data; and
- can leverage correlation relationship within each mode by defining the cost matrix.

Preliminaries: Wasserstein Distance and Optimal Transport

Definition (Wasserstein distance between vectors)

Wasserstein distance between probability vectors $\mathbf{a}$ and $\mathbf{b}$ is defined as

$$W(\mathbf{a}, \mathbf{b}) = \langle \mathbf{C}, \mathbf{T} \rangle \tag{1}$$

- $\mathbf{C}$ is the cost matrix, where c_{ij} is the cost of moving a_i to b_j.
- $\mathbf{T} \in U(\mathbf{a}, \mathbf{b})$ is an Optimal Transport (OT) solution between $\mathbf{a}$ and $\mathbf{b}$.
- $U(\mathbf{a}, \mathbf{b}) = \{\mathbf{T} \in \mathbb{R}_+^{n \times m} | \mathbf{T}\mathbf{1}_m = \mathbf{a}, \mathbf{T}^T\mathbf{1}_n = \mathbf{b}\}$ is the feasible set of the OT problem.

Solving this OT problem is very expensive[5]: it has a complexity of $O(n^3)$.

[5] *Gabriel Peyré, Marco Cuturi, et al. "Computational optimal transport". In: Foundations and Trends® in Machine Learning (2019).*

Preliminaries: Wasserstein Distance and Optimal Transport

An efficient alternative:

Definition (Entropy regularized OT problem[6])

The entropy regularized OT problem is defined as:

$$W_V(\mathbf{a},\mathbf{b}) = \underset{\mathbf{T}\in U(\mathbf{a},\mathbf{b})}{\text{minimize}} \quad \langle \mathbf{C},\mathbf{T}\rangle - \frac{1}{\rho}E(\mathbf{T}), \tag{2}$$

where $E(\mathbf{T}) = -\sum_{i,j=1}^{M,N} t_{ij}\log(t_{ij})$ is the entropy of $\mathbf{T}$.

- It is strictly convex with a unique solution.
- It can be tackled with $\mathbf{u}$ and $\mathbf{v}$ such that $\operatorname{diag}(\mathbf{u})\exp(-\rho\mathbf{C})\operatorname{diag}(\mathbf{v}) \in U(\mathbf{a},\mathbf{b})$.
- Optimal $\mathbf{u}$ and $\mathbf{v}$ can be computed via the Sinkhorn's algorithm[7].

[6] *Marco Cuturi. "Sinkhorn distances: Lightspeed computation of optimal transport". In: Advances in Neural Information Processing Systems. 2013.*
[7] *Richard Sinkhorn and Paul Knopp. "Concerning nonnegative matrices and doubly stochastic matrices". In: Pacific Journal of Mathematics (1967)*

Challenges

However, applying Wasserstein distance to tensor factorization is challenging:

1. **Wasserstein distance is not well-defined for tensors**: It is well-defined for vectors, yet vectorizing tensor yields extremely large vectors, making it infeasible to solve.

2. **Wasserstein distance is difficult to scale**: It requires to solve the OT problems many times in each iteration, which is extremely time consuming.

3. **Real-world input are often large, sparse and non-negative**: Efficient algorithms are possible only when the sparsity structure are fully utilized.

Our Contributions

Contribution 1: Defining Wasserstein Tensor Distance

- SWIFT is the first work that defines Wasserstein distance for tensors.
- It does not assume any particular distribution.
- Therefore, it can handle non-negative inputs, including binaries, counts, and real-values.

Contribution 2: Formulating Wasserstein Tensor Factorization

- We propose SWIFT model to minimize the Wasserstein distance between the input and its CP reconstructions.

Contribution 3: Efficiently Solving Wasserstein Tensor Factorization

- SWIFT effectively explores the sparsity structure of the input and reduces the number of times required to compute OT.
- It reduces the computational time by efficient rearrangement of its sub-problems.
- As a result, it achieves 921x speed up over a naive implementation.

Defining Wasserstein Tensor Distance

We first define the Wasserstein distance for matrices by **summing that over their columns**:

Definition (Wasserstein Matrix Distance)

Given a cost matrix $\mathbf{C} \in \mathbb{R}_+^{M \times M}$, the Wasserstein distance between two matrices $\mathbf{A} = [\mathbf{a}_1, \dots, \mathbf{a}_P] \in \mathbb{R}_+^{M \times P}$ and $\mathbf{B} = [\mathbf{b}_1, \dots, \mathbf{b}_P] \in \mathbb{R}_+^{M \times P}$ is denoted by $W_M(\mathbf{A}, \mathbf{B})$, and given by:

$$\boxed{W_M(\mathbf{A}, \mathbf{B}) = \sum_{p=1}^{P} W_V(\mathbf{a}_p, \mathbf{b}_p)} = \underset{\overline{\mathbf{T}} \in U(\mathbf{A}, \mathbf{B})}{\text{minimize}} \langle \overline{\mathbf{C}}, \overline{\mathbf{T}} \rangle - \frac{1}{\rho} E(\overline{\mathbf{T}}), \tag{3}$$

where $\overline{\mathbf{C}} = [\underbrace{\mathbf{C}, \dots., \mathbf{C}}_{P \text{ times}}]$, $\overline{\mathbf{T}} = [\mathbf{T}_1, \dots \mathbf{T}_p, \dots, \mathbf{T}_P]$, and the feasible set $U(\mathbf{A}, \mathbf{B})$ is given by:

$$U(\mathbf{A}, \mathbf{B}) = \left\{ \overline{\mathbf{T}} \in \mathbb{R}_+^{M \times MP} \mid \mathbf{T}_p \mathbf{1}_M = \mathbf{a}_p, \mathbf{T}_p^T \mathbf{1}_M = \mathbf{b}_p \quad \forall p \right\} = \left\{ \overline{\mathbf{T}} \in \mathbb{R}_+^{M \times MP} \mid \Delta(\overline{\mathbf{T}}) = \mathbf{A}, \Psi(\overline{\mathbf{T}}) = \mathbf{B} \right\}, \tag{4}$$

where $\Delta(\overline{\mathbf{T}}) = [\mathbf{T}_1 \mathbf{1}_M, \dots, \mathbf{T}_P \mathbf{1}_M] = \overline{\mathbf{T}}(\mathbf{I}_P \otimes \mathbf{1}_M)$, $\Psi(\overline{\mathbf{T}}) = [\mathbf{T}_1^T \mathbf{1}_M, \dots, \mathbf{T}_P^T \mathbf{1}_M]$ and $\mathbf{1}_M$ is a vector of all ones with the size of M.

Defining Wasserstein Tensor Distance

Then we can define the Wasserstein distance for tensors by **summing that over the matricization along each mode of the tensor**:

Definition (Wasserstein Tensor Distance)

The Wasserstein distance between N-th order tensor $\mathcal{X} \in \mathbb{R}_+^{I_1 \times \ldots \times I_N}$ and its reconstruction $\hat{\mathcal{X}} \in \mathbb{R}_+^{I_1 \times \ldots \times I_N}$ is denoted by $W_T(\hat{\mathcal{X}}, \mathcal{X})$:

$$\boxed{W_T(\hat{\mathcal{X}}, \mathcal{X}) = \sum_{n=1}^{N} W_M\left(\widehat{\mathbf{X}}_{(n)}, \mathbf{X}_{(n)}\right)} \equiv \sum_{n=1}^{N} \left\{ \underset{\overline{\mathbf{T}}_n \in U\left(\widehat{\mathbf{X}}_{(n)}, \mathbf{X}_{(n)}\right)}{\text{minimize}} \quad \langle \overline{\mathbf{C}}_n, \overline{\mathbf{T}}_n \rangle - \frac{1}{\rho} E(\overline{\mathbf{T}}_n) \right\}, \qquad (5)$$

where $\mathbf{X}_{(n)} \in \mathbb{R}_+^{I_n \times I_{(-n)}}$ is the n-th mode matricization of $\mathcal{X}$, $\overline{\mathbf{C}}_n = [\mathbf{C}_n, \mathbf{C}_n, \ldots, \mathbf{C}_n] \in \mathbb{R}_+^{I_n \times I_n I_{(-n)}}$, and $\overline{\mathbf{T}}_n = [\mathbf{T}_{n1}, \ldots, \mathbf{T}_{nj}, \ldots, \mathbf{T}_{nI_{(-n)}}] \in \mathbb{R}_+^{I_n \times I_n I_{(-n)}}$. $\mathbf{T}_{nj} \in \mathbb{R}_+^{I_n \times I_n}$ is the transport matrix between the columns $\widehat{\mathbf{X}}_{(n)}(:, j) \in \mathbb{R}_+^{I_n}$ and $\mathbf{X}_{(n)}(:, j) \in \mathbb{R}_+^{I_n}$.

The Wasserstein distance $W_T(\mathcal{X}, \mathcal{Y})$ defined above is a valid distance and satisfies the metric axioms of positivity, symmetry, and triangle inequality.

Defining Wasserstein Tensor Distance

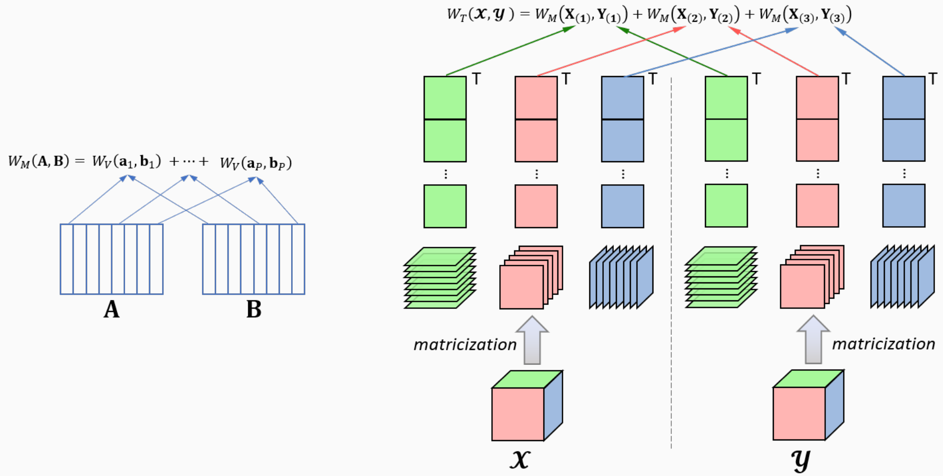

$W_M(\mathbf{A},\mathbf{B}) = W_V(\mathbf{a}_1,\mathbf{b}_1) + \cdots + W_V(\mathbf{a}_P,\mathbf{b}_P)$
A
B
$W_T(\mathcal{X},\mathcal{Y}) = W_M(\mathbf{X}_{(1)},\mathbf{Y}_{(1)}) + W_M(\mathbf{X}_{(2)},\mathbf{Y}_{(2)}) + W_M(\mathbf{X}_{(3)},\mathbf{Y}_{(3)})$
T
matricization
matricization
$\mathcal{X}$
$\mathcal{Y}$

Illustration of the Wasserstein distances. Left: Wasserstein matrix distance; right: Wasserstein tensor distance.

Wasserstein Tensor Factorization

SWIFT minimizes the Wasserstein tensor distance between input and its CP reconstruction:

Optimization problem

$$\begin{aligned}\underset{\{\mathbf{A}_n \geq 0, \overline{\mathbf{T}}_n\}_{n=1}^N}{\text{minimize}} \quad & \sum_{n=1}^{N} \left(\langle \overline{\mathbf{C}}_n, \overline{\mathbf{T}}_n \rangle - \frac{1}{\rho} E(\overline{\mathbf{T}}_n) \right) \\ \text{subject to} \quad & \widehat{\mathcal{X}} = [\![\mathbf{A}_1, \dots, \mathbf{A}_N]\!] \\ & \overline{\mathbf{T}}_n \in U(\widehat{\mathbf{X}}_{(n)}, \mathbf{X}_{(n)}), \ n = 1, \dots, N\end{aligned}$$

Constraint Relaxation using the generalized KL-divergence

$$\underset{\{\mathbf{A}_n \geq 0, \overline{\mathbf{T}}_n\}_{n=1}^N}{\text{minimize}} \quad \sum_{n=1}^{N} \left(\underbrace{\langle \overline{\mathbf{C}}_n, \overline{\mathbf{T}}_n \rangle - \frac{1}{\rho} E(\overline{\mathbf{T}}_n)}_{\text{Part } P_1} + \lambda \left(\underbrace{KL(\Delta(\overline{\mathbf{T}}_n) || \mathbf{A}_n (\mathbf{A}_{\odot}^{(-n)})^T)}_{\text{Part } P_2} + \underbrace{KL(\Psi(\overline{\mathbf{T}}_n) || \mathbf{X}_{(n)})}_{\text{Part } P_3} \right) \right) \qquad (6)$$

We alternate between $\mathbf{A}_n$ and $\overline{\mathbf{T}}_n$ to solve Eq. (6).

Efficient Algorithms: 1. Solving for OT Problems ($\overline{\mathbf{T}}_n$)

Note that: $\overline{\mathbf{T}}_n = [\mathbf{T}_{n1}, ..., \mathbf{T}_{nj}, ..., \mathbf{T}_{nI_{(-n)}}] \in \mathbb{R}_+^{I_n \times I_n I_{(-n)}}$.
The number of optimal transport problems to solve is: $I_{(-n)} = I_1 \times \cdots \times I_{n-1} \times I_{n+1} \times \cdots \times I_N$.

Instead, we use the property of the OT solution $\mathbf{T}_{nj}^* \mathbf{1} = \mathrm{diag}(\mathbf{u}_j)\mathbf{K}_n \mathbf{v}_j = \mathbf{u}_j * (\mathbf{K}_n \mathbf{v}_j)$; therefore:

Proposition 2

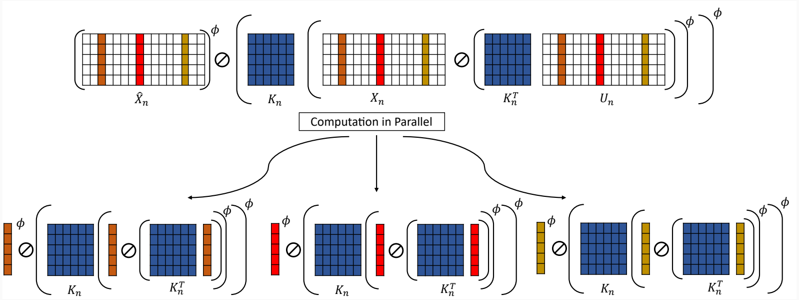
$$\Delta(\overline{\mathbf{T}}_n) = [\mathbf{T}_{n1}\mathbf{1}, ..., \mathbf{T}_{nj}\mathbf{1}, ..., \mathbf{T}_{nI_{(-n)}}\mathbf{1}] = \mathbf{U}_n * (\mathbf{K}_n \mathbf{V}_n) \tag{7}$$

minimizes (6), where $\mathbf{K}_n = e^{(-\rho \mathbf{C}_n - 1)} \in \mathbb{R}_+^{I_n \times I_n}$, $\mathbf{U}_n = (\widehat{\mathbf{X}}_{(n)})^{\Phi} \oslash \left(\mathbf{K}_n \left(\mathbf{X}_{(n)} \oslash (\mathbf{K}_n^T \mathbf{U}_n)\right)^{\Phi}\right)^{\Phi}$, $\mathbf{V}_n = \left(\mathbf{X}_{(n)} \oslash (\mathbf{K}_n^T \mathbf{U}_n)\right)^{\Phi}$, $\Phi = \frac{\lambda\rho}{\lambda\rho + 1}$, and $\oslash$ indicates element-wise division.

Efficient Algorithms: 1. Solving for OT Problems ($\overline{\mathbf{T}}_n$)

Exploring the sparsity structure for efficiently computing $\Delta(\overline{\mathbf{T}}_n)$

- Many columns of the $\mathbf{X}_{(n)}$ are all zeros; thus they can be ignored when computing $\mathbf{V}_n$.
- Besides, each column of $\mathbf{V}_n$ can be computed in parallel.



SWIFT explores sparsity structure in input data $\mathbf{X}_{(n)}$ and drops zero values columns.

Efficient Algorithms: 2. Updating CP factors ($\mathbf{A}_n$)

Sub-problem for the CP factor matrices $\mathbf{A}_n$

$$\underset{\mathbf{A}_n \geq 0}{\text{minimize}} \quad \sum_{i=1}^{N} KL\left(\Delta(\overline{\mathbf{T}}_i) \;||\; \mathbf{A}_i(\mathbf{A}_{\odot}^{(-i)})^T\right) \tag{8}$$

Challenge: $\mathbf{A}_n$ is also involved in the Khatri-Rao product $\mathbf{A}_{\odot}^{(-i)}$.

To tackle this, we define an rearranging operator Π, such that

Efficient rearranging operation

$$\Pi(\mathbf{A}_i(\mathbf{A}_{\odot}^{(-i)})^T, n) = \mathbf{A}_n(\mathbf{A}_{\odot}^{(-n)})^T \in \mathbb{R}_+^{I_n \times I_{(-n)}} \quad \forall\; i \neq n. \tag{9}$$

In this way, the Khatri-Rao product term no longer contain the factor matrix $\mathbf{A}_n$.

Efficient Algorithms: 2. Updating CP factors ($\mathbf{A}_n$)

With this operator, the sub-problem is equivalent to:

Rearranged sub-problem for $\mathbf{A}_n$

$$\underset{\mathbf{A}_n \geq 0}{\text{minimize}} \quad KL\left(\begin{bmatrix} \Pi(\Delta(\overline{\mathbf{T}}_1), n) \\ \dots \\ \Pi(\Delta(\overline{\mathbf{T}}_i), n) \\ \dots \\ \Pi(\Delta(\overline{\mathbf{T}}_N), n) \end{bmatrix} \middle|\middle| \begin{bmatrix} \mathbf{A}_n(\mathbf{A}_{\odot}^{(-n)})^T \\ \dots \\ \mathbf{A}_n(\mathbf{A}_{\odot}^{(-n)})^T \\ \dots \\ \mathbf{A}_n(\mathbf{A}_{\odot}^{(-n)})^T \end{bmatrix} \right) \tag{10}$$

With the rearranged objective function, the factor matrix $\mathbf{A}_n$ can be efficiently updated via multiplicative update rules[8].

[8] *Daniel D Lee and H Sebastian Seung. "Algorithms for non-negative matrix factorization". In: Advances in Neural Information Processing Systems. 2001.*

Experiments: Datasets and Evaluation Metrics

BBC News[9]

- a third-order counting tensor with size of 400 articles by 100 words by 100 words
- downstream task: article category classification; evaluated by *accuracy*.

Sutter

- a dataset collected from a large real-world health provider network
- a third-order binary tensor with size of 1000 patients by 100 diagnoses by 100 medications
- downstream task: heart failure onset; evaluated by *PR-AUC*.

We use the pair-wise cosine distance to compute the cost matrices for each mode of the two datasets.

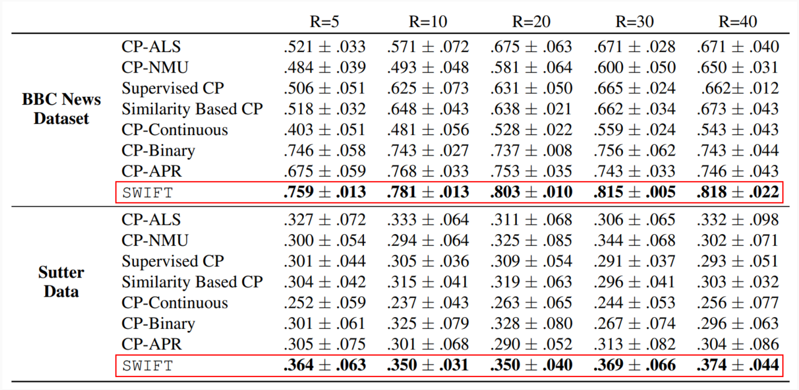
[9] *Derek Greene and Pádraig Cunningham. "Practical Solutions to the Problem of Diagonal Dominance in Kernel Document Clustering". In: International Conference on Machine learning. 2006.*

Experiments: Baselines

We compare against the following tensor factorization models with different loss functions:

Model	Loss Type	Underlying Distribution Assumption	Reference
CP-ALS	MSE Loss	Gaussian	(Bader & Kolda 2007)
CP-NMU	MSE Loss	Gaussian	(Bader & Kolda 2007)
Supervised CP	MSE Loss	Gaussian	(Kim et al. 2017)
Similarity based CP	MSE Loss	Gaussian	(Kim et al. 2017)
CP-Continuous	Gamma Loss	Gamma	(Hong et al. 2020)
CP-Binary	Log Loss	Bernoulli	(Hong et al. 2020)
CP-APR	KL Loss	Poisson	(Chi & Kolda 2012)

Experimental Results: Classification Performance



		R=5	R=10	R=20	R=30	R=40
BBC News Dataset	CP-ALS	.521 ± .033	.571 ± .072	.675 ± .063	.671 ± .028	.671 ± .040
	CP-NMU	.484 ± .039	.493 ± .048	.581 ± .064	.600 ± .050	.650 ± .031
	Supervised CP	.506 ± .051	.625 ± .073	.631 ± .050	.665 ± .024	.662± .012
	Similarity Based CP	.518 ± .032	.648 ± .043	.638 ± .021	.662 ± .034	.673 ± .043
	CP-Continuous	.403 ± .051	.481 ± .056	.528 ± .022	.559 ± .024	.543 ± .043
	CP-Binary	.746 ± .058	.743 ± .027	.737 ± .008	.756 ± .062	.743 ± .044
	CP-APR	.675 ± .059	.768 ± .033	.753 ± .035	.743 ± .033	.746 ± .043
	SWIFT	**.759 ± .013**	**.781 ± .013**	**.803 ± .010**	**.815 ± .005**	**.818 ± .022**
Sutter Data	CP-ALS	.327 ± .072	.333 ± .064	.311 ± .068	.306 ± .065	.332 ± .098
	CP-NMU	.300 ± .054	.294 ± .064	.325 ± .085	.344 ± .068	.302 ± .071
	Supervised CP	.301 ± .044	.305 ± .036	.309 ± .054	.291 ± .037	.293 ± .051
	Similarity Based CP	.304 ± .042	.315 ± .041	.319 ± .063	.296 ± .041	.303 ± .032
	CP-Continuous	.252 ± .059	.237 ± .043	.263 ± .065	.244 ± .053	.256 ± .077
	CP-Binary	.301 ± .061	.325 ± .079	.328 ± .080	.267 ± .074	.296 ± .063
	CP-APR	.305 ± .075	.301 ± .068	.290 ± .052	.313 ± .082	.304 ± .086
	SWIFT	**.364 ± .063**	**.350 ± .031**	**.350 ± .040**	**.369 ± .066**	**.374 ± .044**

SWIFT outperforms all models consistently by a large margin.

Experimental Results: Classification Performance

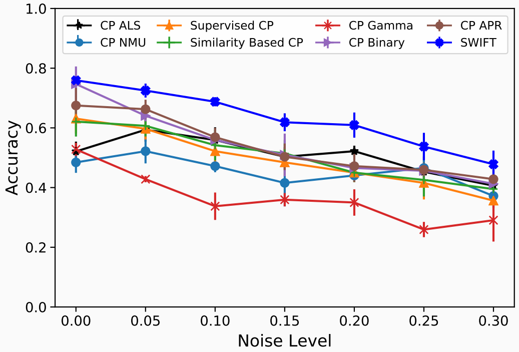
Comparison against widely-adopted classifiers:

	Accuracy on BBC	PR-AUC on Sutter
Lasso Logistic Regression	$.728 \pm .013$	$.308 \pm .033$
Random Forest	$.628 \pm .049$	$.318 \pm .083$
Multi-Layer Perceptron	$.690 \pm .052$	$.305 \pm .054$
K-Nearest Neighbor	$.596 \pm .067$	$.259 \pm .067$
SWIFT (R=5)	$.759 \pm .013$	$.364 \pm .063$
SWIFT (R=40)	$\mathbf{.818} \pm \mathbf{.020}$	$\mathbf{.374} \pm \mathbf{.044}$

SWIFT with rank of 5 already outperforms all other classifiers compared.

Experimental Results: Classification Performance on Noisy Data

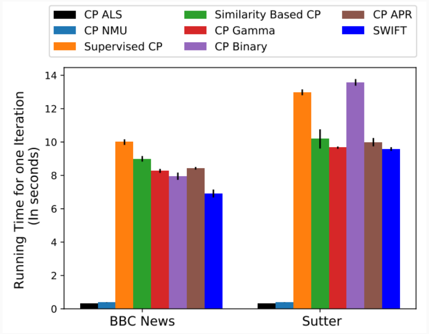
We inject random noise to the BBC News data and run all models using the noisy data:



SWIFT outperforms all baselines, especially for medium and high noise levels.

Experimental Results: Scalability of SWIFT

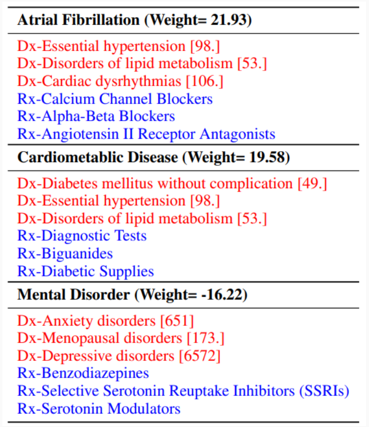
We set $R = 40$, switch off all parallelization of SWIFT for fair comparison and measure the running time of all models.



SWIFT is as scalable as other CP factorization models.

Experimental Results: Interpretability of SWIFT

We interpret the factor matrices learned for Sutter datasets. Following are three examples:



Atrial Fibrillation (Weight= 21.93)
Dx-Essential hypertension [98.]
Dx-Disorders of lipid metabolism [53.]
Dx-Cardiac dysrhythmias [106.]
Rx-Calcium Channel Blockers
Rx-Alpha-Beta Blockers
Rx-Angiotensin II Receptor Antagonists
Cardiometablic Disease (Weight= 19.58)
Dx-Diabetes mellitus without complication [49.]
Dx-Essential hypertension [98.]
Dx-Disorders of lipid metabolism [53.]
Rx-Diagnostic Tests
Rx-Biguanides
Rx-Diabetic Supplies
Mental Disorder (Weight= -16.22)
Dx-Anxiety disorders [651]
Dx-Menopausal disorders [173.]
Dx-Depressive disorders [6572]
Rx-Benzodiazepines
Rx-Selective Serotonin Reuptake Inhibitors (SSRIs)
Rx-Serotonin Modulators

- Each group (phenotype) contains clinically relevant diagnoses and medications.
- The weight indicates the lasso logistic regression coefficient for heart failure (HF) prediction.
- First two groups are clinically relevant to HF, but the third is not.
- The clinical meaningfulness is endorsed by a medical expert.

SWIFT yields interpretable factor matrices.

Conclusion

- We define the Wasserstein distance between two tensors and propose SWIFT, a Wasserstein tensor factorization model.
- We derive an efficient learning algorithm by exploring the sparsity structure and introducing efficient rearrangement operator.
- Empirical evaluations demonstrate that SWIFT consistently outperforms baselines in downstream prediction tasks, even in the presence of heavy noise.
- SWIFT is also shown scalable and interpretable.

Thank you!

All questions and comments are greatly appreciated!