1 Computational results

1.1 Synthetic data

1.1.1 Feasibility problem

In this section, we present experimental results which empirically show when the problem is 'over complete', **AVTA** can be a 'shortcut' solution. In another word, given an $m \times n$ matrix A as data, where the convex hull of the columns of A, denoted by $conv(A)$, has K vertices, $K \ll n$. Let $\overline{A}$ denote the vertices of $conv(A)$. In some applications one simply needs to compute $\overline{A}$. In many other applications the computation of $\overline{A}$ reduces the problem over A to an equivalent problem over $\overline{A}$. Thus the complexity of solving a particular problem $\mathcal{P}$ on input A, denoted by $T(\mathcal{P}(A))$, can alternatively be replaced with $T(\overline{A}) + T(\mathcal{P}(\overline{A}))$, where $T(\overline{A})$ is the complexity of computing $\overline{A}$. We apply the **AVTA** to solve 2 classical problems which appear in many applications.

Convex hull membership problem: The convex hull membership problem is to test if a given $p \in \mathbb{R}^m$ lies in $conv(S)$. The problem can be solved by **Triangle Algorithm** [8] and **Frank Wolfe** [7] by producing an approximate solution p_ϵ where $d(p, p_\epsilon) \leq \epsilon$. This problem can also solved by following linear program:

$$\begin{aligned} & min \quad \vec{1} \cdot y \\ & A\alpha + \beta = p \\ & \sum_{i=1}^{K} \alpha_i = 1 \\ & \alpha \geq 0 \ , \ \beta \geq 0 \end{aligned} \tag{1}$$

where $p \in conv(s)$ if and only if the optimal solution of this linear program is $y^* = 0$. In case when the number of vertices is much less than the number of points i.e. $n \gg K$, the membership problem can be reduced to following query: Is there exists $d+1$ vertices s.t. p is ϵ close to the convex hull of these $d+1$ vertices? Since the **AVTA** can detect vertices progressively, it can be used to solve the membership query:

AVTA for convex hull membership, ($S = \{v_1, \ldots, v_n\}$, p, ϵ)

- **Step 0.** Set $\widehat{S} = \{Farthest(v, S)\}$ for some $v \in S$.

  Step 1. Set $\widehat{S} = \{Farthest(v, S)\}$ for some $v \in S$, $\gamma = 2$.

- **Step 2.** Call **Triangle Algorithm** $(\widehat{S}, p, \gamma/2)$.

- **Step 3.** If the output p' of Step 1 is a p-witness then Goto Step 4. If p' is a $\gamma/2$-approximate solution to p with $\gamma \geq \epsilon$ then $\gamma = \gamma/2$, Goto Step 2. If p' is a $\gamma/2$-approximate solution to p with $\gamma \leq \epsilon$ then Output p' as ϵ approximate.

- **Step 4.** Let $c' = p - p'$.

  Compute S', the set of optimal solutions of $\max\{c'^T x : x \in S \setminus \widehat{S}\}$. Randomly select $v' \in S'$. $v' \leftarrow Farthest(v', S')$, $\widehat{S} \leftarrow \widehat{S} \cup \{v'\}$.

- **Step 5.** If $p = v'$, Output p as a vertex.

In the experiments, vertices of the convex hull are generated by the Gaussian distribution, i.e. $v_i \sim \mathcal{N}(0, \mathcal{I}_m), i \in [K]$. Having generated the vertices, the 'redundant' points d_j where $d_j \in conv(S), j \in [n-K]$ are produced using random convex combination $d_j = \sum_{i=1}^{K} \alpha_i v_i$. Here α_i are scaled standard uniform random variable where α_i are scaled so that $\sum_{i=1}^{K} \alpha_i = 1$. Specifically, comparison is by fixing $K = 100$, $m = 50$ and n varying from $5000 \sim 500000$.

Table 1: Running time(secs)

# of redundant pts	AVTA	TA	FW	Smplx
5000	1.75	0.21	0.52	0.9
20000	1.49	0.66	1.94	2.76
45000	2.94	1.84	5.51	6.16
80000	2.71	3.22	10.87	10.63
125000	3.83	4.28	17.67	15.95
180000	4.15	5.38	23.14	24.13
245000	6.95	9.56	33.42	36.96
320000	8.09	13.24	44.99	44.26
405000	10.01	14.75	56.35	59.5
500000	14.12	15.69	70.7	90.41

Table 1 shows when $n \gg K$, AVTA is more efficient than other algorithms solving the convex hull membership problem. This result supports the output sensitivity conclusion in the analysis of efficiency of AVTA.

Non negative linear system The non negative linear system problem is to find a feasible solution of :

$$\begin{aligned} A\alpha &= p \\ \alpha &\geq 0 \end{aligned} \tag{2}$$

In another word, to test if $p \in cone(A_j)$ where A_j are columns of A. In case when A is over complete, any feasible p can be represented using only the generaters of $cone(A)$ the set $\bar{A} \subset A$. By scaling A so that columns of AD ($D_{ii} = \frac{b}{a \cdot A_i}$) are in a $m-1$ dimensional hyperlane $a \cdot \alpha = b$, one can find the generaters of $cone(A)$ by finding the vertices of the convex hull of the projected points. This could be done efficiently by AVTA. Suppose we have a linear system A and series of query points p , it is sufficient to run AVTA once for dimension reduction and solve the subproblem $\bar{A}\alpha' = p, \alpha' \geq 0$ using simplex method. In the experiment, we compare the running time of **Simplex Method** with **AVTA+Simplex Method**. The generator $\bar{A}$ is entrywise independent $uniform(0, 1)$ random matrix and the 'overcomplete' part of the matrix $\bar{A}^c = A/\bar{A}$ are generated by $\bar{A}^c = \bar{A}B$ where $B \in \mathbb{R}^{K \times (n-K)}$ is entrywise independent $uniform(0, 10)$ random matrix. We set the number of generators $K = 100$, the dimension $m = 50$, and the number of 'redundant' columns $n = 50000$. The series of query points p are generated as $p = Ax$ where $x \in \mathbb{R}^n$ is entrywise independent $uniform(0, 1)$ random vector.

Table 2: Running time(secs)

# of query	AVTA+Smplx	Smplx	# of query	AVTA+Smplx	Smplx
1	284.67	143.13	11	285.43	2348.44
2	284.75	284.63	12	285.50	2604.20
3	284.82	434.52	13	285.58	2856.50
4	284.90	603.32	14	285.66	3106.68
5	284.96	777.74	15	285.70	3360.04
6	285.00	966.73	16	285.77	3618.01
7	285.04	1210.67	17	285.87	3884.61
8	285.19	1498.52	18	285.91	4160.58
9	285.27	1794.97	19	286.01	4443.00
10	285.35	2081.76	20	286.09	4733.16

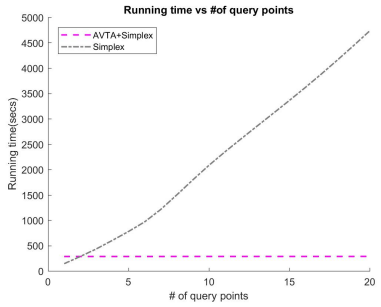


Figure 1: Running time for algorithms to find a feasible solution.

It can be observed from Figure 1 and Table 2 that the running time of AVTA+Simplex doesn't have obvious increase while Simplex increases drastically. This suggests the potential applications of AVTA in dynamic linear feasibility problem.

1.1.2 Computing all vertices

Compute vertices in high dimension The convex hull vertices problem is to find all vertices of a set of points in $\mathbb{R}^m$. In general, computing exact vertices in dimension d has worst complexity exponential in d [4]. However, given $conv(A)$ satisfying γ robust assumption, $AVTA$ can compute the set of vertices efficiently.
In the experiment, we set $K = 100$, $n = 500$ and m varying from $2 \sim 12$. [1]. The computational results is shown in Table 3. In high dimension $m \geq 9$, when $conv(S)$ is γ robust for some $\gamma > 0$, the **AVTA** algorithm successfully find all vertices of the convex hull efficiently while the **Quick hull** algorithm is stuck by its explosion of complexity in dimension m.

Table 3: Running time(secs)

dim	Qhull	AVTA	dim	Qhull	AVTA
2	0.13	14.82	7	2.92	41.51
3	0.02	16.62	8	16.48	39.63
4	0.04	24.49	9	82.09	44.21
5	0.12	32.76	10	391.36	45.79
6	0.59	37.66	11	1479.51	51.19

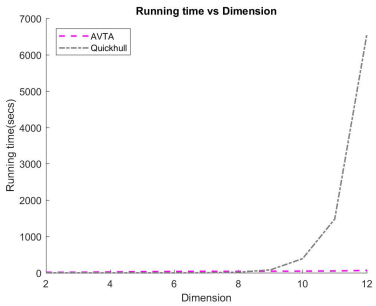


Figure 2: Running time for algorithms to find all vertices.

Compute vertices with perturbation: In this section, we compare **AVTA** with another robust algorithm **Fast Anchor Word**. While the modified AVTA algorithm comes with theoretical guarantees, in certain cases the algorithm might output many more vertices, K_ε, than desired. Here we present a practical implementation that always outputs exactly K vertices, provided K is known. Notice that we want a fast way to detect good approximations to the original vertices of the set S and prune out spurious points, i.e., additional vertices of the set S_ε. The key insight on top of the AVTA algorithm is the following: *If the perturbed set is randomly projected onto a lower dimensional space, it is more likely for an original vertex to still be a vertex than for a spurious vertex.* Using this insight the algorithm outlined below runs the modified AVTA algorithm

[1] The maximum of dimension is 12 in the experiment because of the explosion of running time of the **Quick hull** algorithm

over several random projections and outputs the set of points that appear as vertices in many random projections.

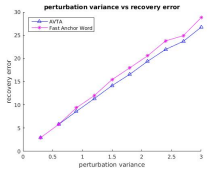
All-Vertex Triangle Algorithm (AVTA) with multiple random projections ($S = \{v_1, \ldots, v_n\}$, K, γ)

- **Step 0.** Set $Freq \leftarrow 0^{|S|}$.
- **Step 1.** For $i = 1$ to M:
  - $S' \leftarrow S$: Project data to randomly chosen $\frac{4log(n)}{\epsilon^2}$ dimensions.
  - $\hat{S} \leftarrow$ All-Vertex TA(S', γ)
  - For each $d_j \in \hat{S}$, $Freq[j] = Freq[j] + 1$.
- **Step 2.** Output top K frequent vertices.

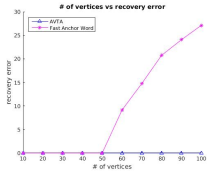
We generate data according to a Gaussian distribution $\mathcal{N}(0, 10)^m$ with $m = 100$. We generate $K = 50$ such points and use them as the vertices of the convex hull. Given these vertices, $n = 500$ interior points are generated as convex combination of the vertices, where the weights are generated scaled i.i.d uniform distribution. We choose a Gaussian perturbation from $\mathcal{N}(0, \tau)^m$ where τ varies from 0.3 to 3. We do an error analysis and evaluate the output of the algorithms by measuring the l_2 distance between true vertices and the convex hull of output vertices of the two algorithms. More precisely, given a true vertex $v_i \in S$ and $\hat{S}$, the output of an algorithm, the error in recovering v_i is defined to be $\min_{u \in conv(\hat{S})} ||u - v_i||_2$. We add up all the errors to get the total accumulated error.

Table 4: Recovery error (Simplex)

variance	AVTA+Multiple Rp	Fast Anhor
0.3	2.96	2.96
0.6	5.79	5.79
0.9	8.61	9.36
1.2	11.34	12.00
1.5	14.16	15.44
1.8	16.60	17.98
2.1	19.40	20.58
2.4	21.93	23.77
2.7	23.69	24.90
3	26.72	28.78



(a) Simplex



(b) General convex hull

Table 4 shows AVTA with multiple random projection has a better recovery error in the simplex case.
It can also be observed from Figure 3b that in general case, as number of vertices exceeds the number of dimensions, FAW fails to recover more vertices and its error explodes.

1.2 Semi synthetic data and real data

1.2.1 Topic modeling

We now show how AVTA can be used to solve the topic modeling in a variety of ways. We first look at how to solve the problem under the separability assumption.
AVTA in the presence of anchor words: Arora et al. [1] provide a practical algorithm for topic modeling with provable guarantees. Their algorithm works under the assumptions that the topic-word matrix is *separable*. In particular, they assume that corresponding to each topic i, there exists an *anchor word* w_i that has a non zero probability of appearing only under topic i. Under this assumption, the algorithm of [1] consists of two stages: a) find the anchor words, and b) use the anchor words to learn the topic word matrix. The problem of finding anchor words corresponds to finding the vertices of the convex hull of the word-word covariance matrix. They propose an algorithm named *fast anchor words* in order to find the vertices. Since AVTA works in general setting, we can instead use AVTA to find the anchor words. Additionally, the fast anchor words algorithm needs to know the value of the number of anchor words, as an input. On the other hand, from the statements of Theorems **??** and**??** it is easy to see that AVTA can work in a variety of settings when other properties of the data are known such as the robustness. We argue that robustness is a parameter that can be tuned in a better manner than trying different values of the number of anchor words. In fact, one can artificially add random noise to the data and make it robust up to certain value. One can then run AVTA with the lower bound on robustness as input and let the algorithm automatically discover the number of anchor words. This is much more desirable in practical settings. Our first implementation of AVTA is named AVTA+RecoverL2 that uses AVTA to detect anchor words and then uses the anchor words to learn the topic word matrix using the approach from [1]. AVTA is also theoretically superior than fast anchor words and achieves slightly better run times in the regime when the number of topics is $o(\log n)$, where n is the number of words in the vocabulary.

This is usually the case in most practical scenarios.

AVTA in the absence of anchor words: The presence of anchor words is a strong assumption that often does not hold in practice. Recently, the work of Bansal et al. [3] designed a new practical algorithm for topic models under the presence of catch words. Catch words for topic i correspond to set S_i such that it's total probability of appearing uunder topic i is significantly higher than in any other topic. Their algorithm called TSVD recovers much better reconstruction of the topic-word matrix in terms of the ℓ_1 error. They also assume that for each topic i, there are a few dominant documents that mostly contain words from topic i. The TSVD algorithm works in two stages. In stage 1, the (thresholded) word-document data matrix is projected onto a K-SVD space to compute a different embedding of the documents. Then, the documents are clustered into K clusters. Under the assumptions mentioned above, one can show that the dominant documents for each topic will be clustered correctly. In stage 2, a simple post processing algorithm can approximate the topic-word matrix from the clustering.

We improve on TSVD by asking the following question: *is K-SVD the right representation of the data?*. Our key insight is that if dominant documents are present in the topic, it is easy to show that most other documents will be approximated by a convex combination of the dominant topics. Furthermore, the coefficients in the convex combinations will provide a much more faithful low dimensional embedding of the data. Using this insight, we propose a new algorithm that runs AVTA on the data matrix to detect vertices and to approximate each point using a convex combination of the vertices. We then use the coefficient matrix as the new representation of the data that needs to be clustered. Once the clustering is obtained, the same post processing step from [3] can be used to recover the topic-word matrix. Our results show that the embedding produced by AVTA leads to much better reconstruction error than of that produced by TSVD. Furthermore, K-SVD is very sensitive to the presence of outliers in the data. In contrast, our new algorithm called AVTA+CatchWord is much more stable to noise in the data.

AVTA+CatchWord
$(S = \{v_1, \ldots, v_n\}, \gamma, K, \epsilon)$

- **Step 0.** Randomly project S onto $2K$ dimensions to get $\hat{S}$.
- **Step 1.** Compute a a super set of vertices $\bar{V}$ by $AVTA(\hat{S}, \gamma)$.
- **Step 2.** Prune $\bar{V}$ into $\hat{V}$ (of size K) by iteratively picking $\bar{v} \in \{\bar{V}\}/\{\hat{V}\}$ which has the maximum distance to $conv(\hat{V})$.
- **Step 2.** For each projected point $\hat{v}_i \in \hat{S} \setminus \hat{V}$, compute a vector α_i such that $\|\hat{V}\alpha_i - \hat{v}_i\| \leq \epsilon$.
- **Step 3.** Initialize cluster assignment for each point by majority weight: $\underset{j \in [K]}{\operatorname{argmax}} \alpha_j$.
- **Step 4.** Clustering using Lloyds algorithm on the embedding provided by the α vectors.
- **Step 5.** Use the post processing as described in [3] to recover the topic-word matrix from the clustering.

We compare our algorithms with the Fast Anchor + Recoverl2 algorithm of [1] and the TSVD algorithm of [3] on two types of data sets: semi-synthetic data and real world data. We next describe our methodology and empirical results in detail.

Semi Synthetic Data: For Semi-Synthetic data set, we use similar methodology as in [1]. We first train the model on real data set using Gibbs sampling with 1000 iterations. We choose 50 as the number of topics which follows [3, **?**]. Given the parameters learned from dataset, we generate documents with α set to be 0.01. The average document length is 1000. Then the reconstruction error is measured by the l_1 distance of bipartite matched pairs between the true word-topic distribution and the word-topic distribution [1]. We then average the errors to compute the final mean error.

Real Data: We use the **NIPS** data set with 1500 documents , and a pruned vocabulary of 2K words, and the NYTimes Corpus with sub sampled 30000 documents, and a pruned vocabulary of 5k words. [2]. For our experiments on NMF we use the Swimmer data set [6] that consists of 256 swimmer figures with each a 32×32 binary pixel image. One can interpret each image as a document and pixels as a word in the document [5]. All swimmers consist of 4 limbs with each limb having 4 different possible poses. One can then consider the different poses of limbs as the true underlying topics [6]. We compare the algorithm proposed in [2] with AVTA+Recoverl2 on the swimmer data set. Since the

[2] `https://archive.ics.uci.edu/ml/datasets/bag+of+words`

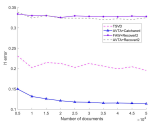
true underlying topics are known, we will plot the output of the algorithms and compare it with the underlying truth.

Implementation Details: We compare 4 algorithms, AVTA+CatchWord, TSVD, the Fast Anchor + Recoverl2 and the AVTA+Recoverl2. We implement our own version of Fast Anchor + Recoverl2 as described in [1]. TSVD is implemented using the code provided by the authors in [3]. AVTA+Recoverl2 corresponds to using AVTA to detect anchor words from the word-word covariance matrix and then using the Recoverl2 procedure from [1] to get the topic-word matrix. AVTA + CatchWord corresponds to finding the low dimensional embedding of each document in terms of the coefficient vector of its representation in the convex hull of the vertices. The next step is to cluster these points. In practice, one could use the Lloyd's algorithm for this step which could be sensitive to initialization. To remedy this, we use similar heuristic as [3] of the initialization step. We repeat **AVTA** for 3 times and pick the set of vertices with highest quality. We set the number of output vertices $K = 50$ which is the same as the number of topics. i.e. each vertex corresponds to a topic. We found that initializing by simply assigning clusters using neighborhoods of highest degree vertices works effectively. As a final step, we use the post processing step from [3] to recover the topic-word matrix from the clustering.

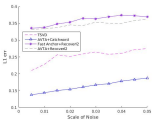
Robustness: We also generate perturbed version of the semi synthetic data. We generate a random matrix with i.i.d. entries uniformly distributed with different scales varying from $0.005 - 0.05$. We test all the algorithms with the document-word matrix added with the noise matrix.

Results on Semi Synthetic Data: Figures 4a and 4b show the ℓ_1 reconstruction of all the four algorithms under both clean and noisy versions of the semi synthetic data set. For topic i, let A_i be the ground truth topic vector and $\hat{A}_i$ be the topic vector recovered by the algorithm. Then the ℓ_1 error is defined as $\frac{1}{K}\sum_{i=1}^{K} \|A_i - \hat{A}_i\|_1$. The plots show that AVTA+CatchWord is consistently better than both TSVD and Fast Anchor + Recoverl2 and produces significantly more accurate topic vectors. In order to further test the robustness of out approach, we plot in Figure 4c the range of the ℓ_1 error obtained across multiple runs of the algorithms on the same data set. The range is defined to be the difference between the maximum and the minimum error recovered by the algorithm across different runs. We see that AVTA+CatchWord produces solutions that are much more stable to the effect of the noise as compared to other algorithms. Table 5 shows the running time of the experiments of 4 algorithms. As can be seen, when using AVTA to learn topic models via the anchor words approach, our algorithm has comparable run time to Fast Anchor + Recoverl2.

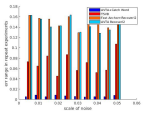
Results on Real Data: For the real world data set, as in prior works [1, 3], we evaluate the coherence to measure topic quality [10]. Given a set of words $\mathcal{W}$ associated with a learned topic, the coherence is computed as: $Coherence(\mathcal{W}) = \sum_{w_1,w_2 \in \mathcal{W}} \log \frac{D(w_1,w_2)+\epsilon}{D(w_2)}$, where $D(w_1)$ and $D(w_1, w_2)$ are the number of documents where w_1 appears and (w_1, w_2) appear together respectively [1], and ε is set to 0.01 to avoid w_1, w_2 that never co-occur [9]. The total coherence is



(a) ℓ_1 error in the semi-synthetic dataset ($K = 50$).



(b) ℓ_1 error in the perturbed semi-synthetic dataset ($K = 50$).



(c) Range of the ℓ_1 error over 10 runs on the same data set.

Table 5: Run time (in secs.) of algorithms on semi synthetic data

Num of documents	5000	15000	30000	50000
Fast anchor+Recoverl2	5.49	6.00	10.30	13.60
AVTA+Recoverl2	7.82	7.68	12.84	16.40
TSVD	17.02	43.27	81.24	112.80
AVTA+Catch Word	29.89	120.04	372.17	864.30

the sum of the coherence of each topic. In the NIPS dataset, 1000 out of the 1500 documents were selected as the training set to learn the word-topic distributions. The rest of the documents were used as the testing set. Table 6 shows the topic coherence obtained by the algorithms. One can see that in both the approaches, either via anchor words or the clustering approach, AVTA based algorithms perform comparably to state of the art methods [3]. The running presented in 7 shows AVTA based algorithm is efficient in real world data.

Table 6: Topic coherence

	Fast Anchor+RecoverL2	AVTA+RecoverL2	TSVD	AVTA+Catch Word
NIPS	-15.8 ±2.24	-16.04 ±2.09	-16.86 ±1.66	-18.65 ±1.78
NYTime	-32.15 ±2.7	-32.13 ±2.43	-29.3933 ±1.43	-30.13 ±1.98

[3]The topic coherence results for TSVD do not match the ones presented in [3] since in their experiments, the authors look at top 10 most frequent words in each topic. In our experiments we compute coherence for the top 5 most frequent words in each topic.

Table 7: Running time on real data experiments

	Fast Anchor+RecoverL2	AVTA+RecoverL2	TSVD	AVTA+Catch Word
NY time	26.05	27.79	237.6	101.07
Nips	3.22	4.41	56.58	22.78

1.3 Non negative matrix factorization

AVTA for NMF: The work of [2] showed that convex hull detection can be used to solve the non negative matrix factorization problem under the separability assumption. We show that by using the more general AVTA algorithm for solving the convex hull problem results in comparable performance guarantee.
Results on NMF: The swimmer data set [6] consists of 256, 32×32 images and the task consists of inferring the underlying "poses" in a given image. In our experiments we use the original swimmer data set and also construct a noisy version by adding spurious poses. Let $\Omega(A)$ be a function that outputs a randomly chosen 32×8 block of an image. We generate a 'spurious pose' of size 32×8 by $\Omega(M_i)$ where M_i is a randomly chosen swimmer image. Then we take another randomly chosen image M_j and compute the corrupted image as $M'_j = M_j + c \cdot \Omega(M_i)$ where we simply set $c = 0.1$. An illustration of the noise data set is shown in Figure 5. We compare the performance of AVTA on these data sets with the performance of the Separable NMF algorithm proposed in [2]. Figures 6 and 7 show the output of the Separable NMF algorithm and that of our algorithm respectively on the noisy data set. Our approach produces competitive results as compared to the Separable NMF algorithm.

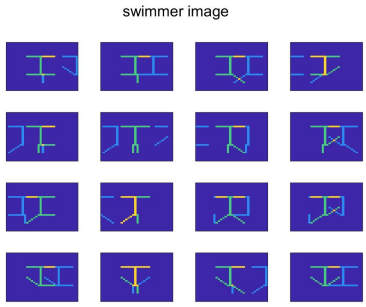


Figure 5: An example of spurious actions in swimmer images.

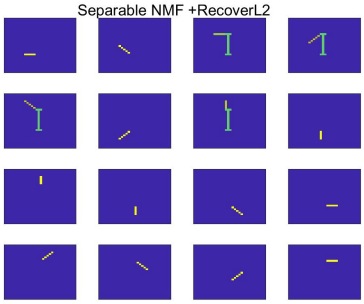


Figure 6: Output of NMF +RecoverL2

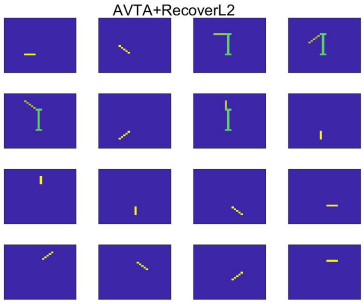


Figure 7: Output of AVTA +RecoverL2

References

[1] Sanjeev Arora, Rong Ge, Yonatan Halpern, David Mimno, Ankur Moitra, David Sontag, Yichen Wu, and Michael Zhu. A practical algorithm for topic modeling with provable guarantees. In *International Conference on Machine Learning*, pages 280–288, 2013.

[2] Sanjeev Arora, Rong Ge, Ravindran Kannan, and Ankur Moitra. Computing a nonnegative matrix factorization–provably. In *Proceedings of the forty-fourth annual ACM symposium on Theory of computing*, pages 145–162. ACM, 2012.

[3] Trapit Bansal, Chiranjib Bhattacharyya, and Ravindran Kannan. A provable svd-based algorithm for learning topics in dominant admixture corpus. In *Advances in Neural Information Processing Systems*, pages 1997–2005, 2014.

[4] Bernard Chazelle. An optimal convex hull algorithm in any fixed dimension. *Discrete & Computational Geometry*, 10(1):377–409, 1993.

[5] Weicong Ding, Mohammad Hossein Rohban, Prakash Ishwar, and Venkatesh Saligrama. Topic discovery through data dependent and random projections. In *ICML (3)*, pages 1202–1210, 2013.

[6] David Donoho and Victoria Stodden. When does non-negative matrix factorization give a correct decomposition into parts? In *Advances in Neural Information Processing Systems*, 2003.

[7] Martin Jaggi. Revisiting frank-wolfe: Projection-free sparse convex optimization. In *ICML (1)*, pages 427–435, 2013.

[8] Bahman Kalantari. A characterization theorem and an algorithm for a convex hull problem. *Annals of Operations Research*, 226(1):301–349, 2015.

[9] Keith Stevens, Philip Kegelmeyer, David Andrzejewski, and David Buttler. Exploring topic coherence over many models and many topics. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 952–961. Association for Computational Linguistics, 2012.

[10] Limin Yao, David Mimno, and Andrew McCallum. Efficient methods for topic model inference on streaming document collections. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 937–946. ACM, 2009.