UNSUPERVISED LEARNING: EVALUATION, DISTRIBUTED SETTING, AND PRIVACY

Maksim Tsikhanovich

Submitted in Partial Fulfillment of the Requirements
for the Degree of

DOCTOR OF PHILOSOPHY

Approved by:
Malik Magdon-Ismail, Chair
Heng Ji,
John Mitchell,
Lirong Xia

Department of Computer Science
Rensselaer Polytechnic Institute
Troy, New York

March 2018
(For Graduation May 2018)

© Copyright 2018
by
Maksim Tsikhanovich
All Rights Reserved

CONTENTS

LIST OF TABLES

LIST OF FIGURES

ACKNOWLEDGMENT

A PhD thesis is
        two decades or so:
interactions and input
from people
        and thoughts.

Some happen only once,
        some are repeated.
Some show what can be.
Some let go what wasn't or was.
Some patch and build up
        (continued support).

And every once in a while
an author feels fortunate.

Thank You.

Especially: Kerah, Lally Basement 2011-2018, Mama, Jarda, Teachers at ISP, Bard, RPI, and Malik. :)

ABSTRACT

Chapter 1 is an overview of topic modeling as a set of unsupervised learning tasks. We present the Latent Dirichlet Allocation (LDA) model, and show how k-means as well as non-negative matrix factorization (NMF) can also be interpreted as topic models. We present a variety of quantitative and qualitative evaluation techniques that aim to capture different properties of the model. Finally we show how we can leverage evaluation techniques and hyperparameter optimization tools to answer typical parameter selection questions. We hope to facilitate future research on topic modeling by encapsulating each of the above parts as a robust and re-usable set of tools, so that a future researcher can focus on one part at a time.

In Chapter 2 we present two algorithms for the data-distributed non-negative matrix factorization (NMF) task, and one for the singular value decomposition (SVD). In the *offline* setting, M parties have already computed NMF models of their local data. Our algorithm ensembles these into a global model by minimizing an upper bound on the reconstruction error for the original data in terms of reconstruction error on the local models. In the *online* setting, the M parties are all participating in a synchronous distributed computation. We present an algorithm that reconstructs the centralized NMF solution exactly if given the same initialization. Finally we present an online SVD algorithm. We compare these algorithms in terms of how well they initialize NMF.

In Chapter 3 we study empirical measures of Distributional Differential Privacy. We want to measure to what extent one participant in a distributed computation can correctly identify the presence of a single document in another participant's database. We propose a measure based on the p-value of the Kolmogorov-Smirnov two-sample hypothesis test. We compare our measures to existing measures such as Differential Privacy, and use it to evaluate the privacy of our online algorithms.

Introduction

The shared narrative in the three chapters of my PhD thesis is best understood in terms of how my research interests evolved.

We started by studying topic modeling, the task of summarizing a collection of documents in terms of the topics they contain. Topic modeling is a unsupervised learning task. In the literature Latent Dirichlet Allocation (LDA) is often considered to be synonymous with 'topic modeling.' However we wanted to investigate whether K-Means and Non-negative Matrix Factorization (NMF) can also work as topic models. To me, the most challenging part of this question was what does it mean to "also work." This lead me down the path of investigating how topic models can be evaluated, which turns out to be a challenging task because they are a form of unsupervised learning. After getting a sense of evaluation methods, the next question was how does one pick model hyper-parameters, such as the number of topics, k. I made a brief divergence into hyper-parameter optimization and found an algorithm that is model and evaluation agnostic. At the end of this we found that indeed the NMF and K-means are comparable to LDA. That is not to say that there aren't measures for which LDA performs best, such as out-of-sample log-likelihood, but for many evaluation measures we found that either K-means or NMF was the best-performing topic model.

Next we wanted to study topic models in a distributed setting, so we looked at the distributed NMF. At first we started in the context of assuming that there are M parties and each party has already computed a topic model of their local data. Then we sought to ensemble these models into a good global model of everyone's combined data. We didn't want each party to simply publish their local data, or their entire model which has $O(n + d)$ variables. This would involve $O(n)$ communication, and wouldn't really take advantage of the local models already being computed. So we focused on ways in which parties could publish summaries of their local models that would required $O(d)$ communication. Since we weren't able to show that we could reconstruct the centralized NMF solution using only the summaries, we developed a distributed NMF algorithm that could be run by all the parties together. This algorithm performs the same iterative updates as the centralized algorithm, however we didn't know how to perform the initialization, since that depended on the SVD. As a

result, we developed a distributed SVD algorithm that can be used to compute the right singular vectors of a matrix, and to initialize the NMF. In the end we ended up with an NMF algorithm that can be used for general NMF, not just for topic modeling.

Finally, although privacy wasn't initially our goal, we decided that it would be interesting to study what kind of privacy guarantees we can make for our distributed NMF and SVD algorithms. Initially our goal was to simply create efficient algorithms, whose communication didn't depend on the number of documents, n. As a result as $n \to \infty$ each document's contribution to the NMF and SVD that we compute should go down. The context is that multiple parties are collaborating in order to build a better model of the underlying distribution from which each of their databases is sampled. The parties also want to get a sense of how much privacy they're losing by executing our algorithms for the NMF and SVD. In the context of a distributed multiparty computation, we defined privacy as a measure of to what extent a coalition of parties can infer whether a document is present or absent in a particular party's database. We wanted a way to measure whether a particular document's contribution is discernible from the cumulative contribution made by all documents in a particular database. To do so we built a method that estimates the distribution of NMF and SVD models for databases that have a particular document compared to databases that don't have that document. Then we measure distance between those distributions as a way to measure 'privacy.' Our method differs from popular methods like Differential Privacy, which guarantee privacy (as opposed to measure it) by adding noise. We avoid adding noise, since in a distributed multi-party computation it often makes the resulting model useless, which voids the original purpose of the multi-party computation: to get a better model.

CHAPTER 1

The Inference and Evaluation of Topic Models

Topic modeling is the task of summarizing a collection of documents in terms of the topics that are most prevalent in the collection. Essentially topic modeling consists of four subtasks:

1. Specifying the model. This means formally answering 'what is a topic?' and 'what does it mean for a document to be about a topic?'.
2. Evaluating the model. What does it mean for a topic model to be 'good' or 'useful'.
3. Fitting a model in order to attempt to maximize a set of evaluation criteria.
4. Presenting or using the model. What information do we want to get out of the model?

The fundamental challenge with topic modeling is that it is a unsupervised learning task, and the task itself isn't fully formally specified. As a result, none of the sub-tasks have one specification that is appropriate for all settings. So that if a researcher wants to propose a new topic model, they must typically study each of the four above tasks. The purpose of this chapter is to provide a survey of tools and methods for each of the sub-tasks, so that future researchers and practitioners can focus on one sub-task of topic modeling at a time.

A current issue in the literature is that when researchers propose a new model, they typically only consider a small subset of evaluation measures. For example, it is common to only consider (one of) a form of the out-of-sample reconstruction error, or a form of correlation to human labels. Both of these are indeed reasonable evaluation measures, but even if they are used together (which is rare) they don't fully capture the purpose of a topic model. As an example, rather than caring about the above two measures we might care about how understandable the topics are. Furthermore, when comparing models in their work, it's not clear how much effort was put into finding reasonable parameters for each of the models. As a result, the pace of research has been slower compared to fields with clear evaluation measures.

One of our contributions is to show that using k-means as a topic model is (a) easy to

implement (b) reasonably easy to find good parameters for (c) is competitive with more complicated topic models. As a result it is a good baseline that can easily be included in future work. We also survey 17 evaluation measures that capture different aspects of what it means to be a good topic model. These can be selected based on the use case of a model, or reported in aggregate. We show that evaluation measures can also be used as guidelines for hyper-parameter optimization, e.g. the problem of picking the number of topics k. Finally we discuss issues and nuances that come with the presentation of topic models. Since there aren't universally useful objective evaluation measures for topic modeling, subjective evaluation is typically more useful than quantitative measures. We have identified several reasons why current approaches for presenting and visualizing models are limited, with the hope that future efforts will make visualization easier and make subjective evaluation more prevalent.

1.1 Probabilistic Formulation of Topic Models

The most general interpretation of topic modeling is that it is equivalent to document clustering, that is the task of grouping similar documents together and dissimilar documents apart. However to narrow down our scope we focus on topic models that have a probabilistic interpretation. We express each topic T_t as a probability distribution over features $P(\cdot|T = t)$, and express each document X_i as a mixture of k topics

$$P(\cdot|X = i) \approx \sum_{t=1}^{k} P(\cdot|T = t)P(T = t|X = i). \tag{1.1}$$

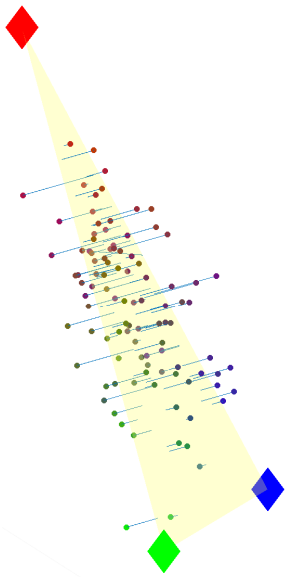
This expression allows us to interpret topic models as generative models for documents. The generative process is to repeat for number of features (e.g. words) in document i:

1. Sample a topic $t \sim P(T = t'|X = i)$.
2. Sample a feature $w \sim P(\cdot|T = t)$.
3. Add w to X_i.

Note that because the priors $P(X = i)$ and $P(T = t)$ are undefined, this isn't a complete generative model. This will be addressed in Section 1.2.4.

Notice that the topics T_t and the documents X_i are distributions over the same space, called

the feature space. As a result topics can be thought of as representative documents which summarize the other documents. The geometric interpretation is that the $(k-1)$-dimensional polytope spanned by the topic vectors should encompass as many documents as possible. This is trivial if k is equal to the number of features, but challenging for any $k \geq 2$ since the documents $\{X_i\}_{i=1}^n$ usually form a full rank matrix. Figure 1.1 depicts these ideas.



The triangle is the 2-dimensional polytope spanned by the three topics represented by diamonds, the documents are represented by small dots. Under a particular topic model the goal is to minimize the ℓ_2 distance from each document to the polytope spanned by the topics.

Fig. 1.1: Visual Representation of a Topic Model

While Equation 1.1 does provide some interpretation for what a topic model is, it still leaves many variables open.

1. What is meant by $\approx$ when comparing two probability distributions? We will examine models where $\approx$ is measured using the ℓ_2 norm, as well as models where it is measured using the loglikelihood or KL-divergence.

2. How should a raw document be converted to a feature vector? We consider stemming, stopword removal, TF-IDF, normalization.

3. How should the number of topics, k, be chosen? We consider model free approach such as analyzing the singular value spectrum, as well as approaches based on fitting models.

4. What kind of prior assumptions or regularizations should be applied to the document-topic distribution $P(T|X)$ as well as the topic-feature distribution $P(\cdot|T)$? We consider Dirichlet priors as well as ℓ_1 and ℓ_2 regularization penalties.

Some of these questions (e.g. the definition of $\approx$) are answered by specifying a particular model. However, others (e.g. feature definition, choice of k) are orthogonal to the model choice and therefore need to be analyzed separately from the choice of which topic model is used. Therefore in order to answer them, we need to build a comprehensive set of evaluation tools.

The rest of this chapter is organized as follows: in Section 1.2 we define and examine several popular topic models. In Section 1.3 we present and motivate a battery of evaluation methods for topic models. Finally in Section 1.4 we show how the evaluation methods we present can be used to answer the questions we posed above as well as evaluate topic models for specific applications.

1.2 A Survey of Topic Models and Inference Algorithms

We study two classes of topic models: those that minimize ℓ_2 reconstruction error, and those that minimize the Kullback-Leibler divergence (D_{KL}). We begin by presenting two ℓ_2-based topic models: k-means (Section 1.2.1) and non-negative matrix factorization (Section 1.2.2). Then we present two D_{KL}-based topic models: probabilistic semantic analysis (Section 1.2.3), and latent Dirichlet allocation (Section 1.2.4). Notation is summarized in Table 1.1.

1.2.1 K-Means

K-means is a well known and conceptually simple point clustering algorithm. Its output can be interpreted as a topic model since it satisfies Equation 1.1. Despite conceptual simplicity, the k-means topic model has nice properties related to interpretability that we will explore in Section 1.4.

In the k-means problem, defined in Figure 1.2, we are given n points in d dimensions, represented as rows in a matrix X. If the rows are non-negative and sum to 1, then the solution to the k-means problem can be interpreted as a topic model.

Table 1.1: Notation for Topic Models

Notation	Meaning	Indexing Variables
n	Number of documents in the collection	i
d	Number of terms in the dictionary	j
k, K	Number of topics in the topic model	t, k
N_i	Number of words of document i	
E_i	The token stream of document i, $\|E_i\| = N_i$	w
X	Document-word distributions in $[0,1]^{n \times d}$	
W	Document-topic distributions in $[0,1]^{n \times K}$	
T	Topic-word distributions in $[0,1]^{K \times d}$	
$W_{:t}$	tth column of W matrix	
$T_{t:}$	tth row of T matrix	
$\|\|X\|\|_F^2$	$\sum_{i=1}^{n} \sum_{j=1}^{d} X_{ij}^2$ Squared Matrix Frobenius Norm	
$\|\|X\|\|_1$	$\sum_{i=1}^{n} \sum_{j=1}^{d} \|X_{ij}\|$ Elementwise Matrix ℓ_1 Norm	
Δ^d	$d-1$-dimensional simplex, $\Delta^d = \{x \in [0,1]^d \colon x_1 + x_2 + \ldots + x_d = 1\}$	

Input:
(1) Document matrix $X \subseteq \Delta^d$.
(2) Number of topics $k \in \mathbf{Z}_{\geq 0}$.

Objective: $\min_{W,T} \sum_{i=1}^{n} ||X_{i:} - W_{i:}T||_2^2$.

Constraints:
(1) Topics are probability distributions over d features. For all $t \in \{1, 2, \ldots, k\}$: $T_{t:} \in \Delta^d$.
(2) One topic per document. For all $i \in \{1, 2, \ldots, n\}$, $t \in \{1, 2, \ldots, k\}$: $W_{it} \in \{0, 1\}$, and $W_{i:} \in \Delta^k$.

Fig. 1.2: K-means for Topic Modeling Problem Statement

If a set of points is assigned to the same topic t, then the topic mean $T_{t:} = \frac{1}{n_t} \sum_{W_i = t} X_{i:}$ is the best representative point. Thus for a given W, T is easy to find, however finding the optimal W is NP-hard [1]. Lloyd's algorithm is a EM-like approximation for k-means.

Each change to the model W, T the k-means algorithm makes improves the objective. Since there is a minimum improvement related to the minimum distance between two points in X and a finite minimum, we conclude that k-means converges to a local minimum of its objective function.

There are different ways to initialize W, T for Lloyd's algorithm:

Algorithm 1 Lloyd's algorithm for K-means

Input: $X \in \mathbf{R}^{n \times d}$, $k \in \mathbf{Z}_{\geq 0}, k > 0$
Output: (W,T) are local minimizers of the k-means objective.
repeat
Set $T \leftarrow SW^T X$, where $S \in [0,1]^{k \times k}$ is a diagonal matrix, $S_{tt} = \frac{1}{|W_{:t}|_1}$. ▷ E-step, set each row of T to be the mean of points with $W_{:t} = 1$
for i={1,...,n} **do** ▷ M-step
Set tth entry of $W_{i:}$ to 1, and all other $W_{i:}$ to 0, where $t = \text{argmin}_{z=1,\ldots,k} ||X_{i:} - T_{z:}||^2$.
end for
until No documents are assigned to different topics.

1. For each row of W, pick the non-zero entry uniformly at random. This is the initialization that Lloyd suggests. It is a quick way to initialize, however nothing can be guaranteed about the quality of a solution of a uniformly random initialization.

2. Greedy Initialization: pick $T_{1:}$ uniformly from one of the points X. Then until k points have been picked, pick the point in X whose distance to the closest point in T is furthest. Proceed with the M-step of Lloyd's algorithm.

3. K-means++ (relaxed greedy initialization): pick $T_{1:}$ uniformly from one of the points X. Let $D(i) = \min_{t \in \{1,\ldots,k\}} ||X_{i:} - T_{t:}||^2$, then until k points have been picked, pick point i with probability $\frac{D(i)^2}{\sum_{i'=1}^{n} D(i')^2}$. Proceed with the M-step of Lloyd's algorithm. In [2] Arthur and Vassilvitskii were able to show that the objective value achieved after using the k-means++ initialization is at most a factor $8 \log k + 2$ of the optimal solution in expectation.

In our experiments we focus on k-means++ as initialization technique. It is not much slower than the greedy initialization method since most of the time cost of fully running k-means is spent on Lloyd's algorithm, and it is less vulnerable to outliers.

1.2.2 Non-negative Matrix Factorization

Since finding the optimal k-means solution is NP-Hard, an initial approach to find an optimal rank k approximation of a input matrix X is based on the singular value decomposition [3]. Let $X = U\Sigma V^T$ be the SVD of a matrix X. Let Σ_k be the matrix Σ where diagonal entries past k are set to 0. Then $\hat{X} = U\Sigma_k V^T$ minimizes the Frobenius reconstruction error

Input:
(1) Document matrix $X \subseteq \Delta^d$.
(2) Number of topics $k \in \mathbf{Z}_{\geq 0}$.
(3) Regularization parameters ℓ_1^T, ℓ_1^W, ℓ_2^T, and ℓ_2^W.

Objective: $\min_{W,T} \frac{1}{2} \left(||X - WT||_F^2 + 2\ell_1^T ||T||_1 + 2\ell_1^W ||W||_1 + \ell_2^W ||W||_F^2 + \ell_2^T ||T||_F^2 \right).$

Constraints:
(1) Topics are probability distributions over d features. For all $t \in \{1, 2, \ldots, k\}$: $T_{t:} \in \Delta^d$.
(2) Documents are convex combinations of topics. For all $i \in \{1, 2, \ldots, n\}$: $W_{i:} \in \Delta^k$.

Fig. 1.3: NMF for Topic Modeling Problem Statement

$||\hat{X} - X||_F$ over all rank k matrices $\hat{X}$. The decomposition can be interpreted as consisting of a document-topic mixing factor $W = U\Sigma_k^{\frac{1}{2}}$ and topic-word mixing factor $T = \Sigma_k^{\frac{1}{2}} V^T$. The issue is that even for a non-negative input matrix X, U and V^T are not guaranteed to be non-negative. As a result this model doesn't satisfy our notion of a topic model given by Equation 1.1, and is tricky to interpret (e.g. what does it mean to have a document that is '+5 cactus' and '-33 submarine').

Non-negative matrix factorization (NMF), defined in Figure 1.3, aims to solve this interpretability problem by constraining the factors W and T to be non-negative. It was popularized by Lee & Seung [4], as a way to learn parts-based representations of data. For a collection of 64x64 images of faces, they showed that PCA features (based on the SVD) would have negative and positive components along most of the 64x64 pixels, whereas the NMF features would be mostly zeros, except for some components; as a result the NMF features looked like parts of the face: eyes, ears, mouth, noses, eyebrows, etc. NMF can be used for topic modeling, since the factors W, T are non-negative and can be rescaled in order to correspond to probability distributions.

While both NMF and k-means are based on the ℓ_2 reconstruction error, they have two important differences:

1. In k-means each row of the W matrix may have exactly one 1 and the rest 0s, which corresponds to assigning a document to exactly one topic. Conversely, in NMF each row of the W matrix simply has to be non-negative and sum to 1. This corresponds to

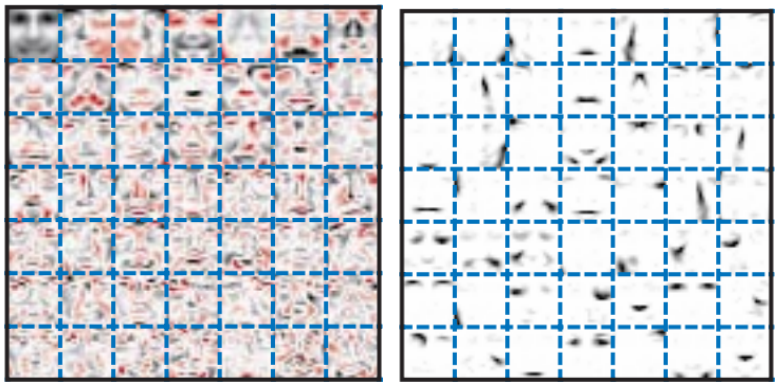


Fig. 1.4: PCA Features Compared to NMF Features, from [5]

expressing each document as a convex combination (or probabilistic mixture) of topics.

2. The topics found by NMF are vertices of the outer $k-$hull (see, e.g., [6]), i.e. the convex hull of X after all but k of the facet-defining hyperplanes have been removed. The plane the topics lie on 'bisects' X in order to minimize the ℓ_2 reconstruction error for each document. Conversely, in k-means topics are cluster centers, and documents are represented as those cluster centers. Therefore we expect the volume of the polytope spanned by NMF's topics to be larger than the volume of the polytope spanned by k-means topics.

The algorithm we use to solve the NMF problem is called rank-one residue iterations (RRI) from [7]. The principle behind this algorithm is to consider the responsibility of a column $W_{:t}$ (resp. row of $T_{t:}$) for explaining X given what the other columns have already explained. To make this sub-problem convex we assume that when solving for $W_{:t}$ (resp. $T_{t:}$) all of T (resp. W) is fixed. Then Column $W_{:t}$ must explain the residual $R_t = X - \sum_{t'=1,t'\neq t}^{d} W_{:t'}T_{t':}$. Thus for column $W_{:t}$ we solve the non-negative least squares problem

$$\min_{W_{:t}>0} \frac{1}{2}\left(||R_t - W_{:t}T_{t:}||_F^2 + 2\ell_1^W||W||_1 + \ell_2^W||W||_F^2\right). \tag{1.2}$$

In [7], Ho shows that the global minimizer is

$$W_{:t} = \begin{cases} \frac{[R_tT_{t:} - \ell_1^W\mathbf{1}_n]_+}{||T_{t:}||_2^2 + \ell_2^W} & \text{if } [R_tT_{t:} - \ell_1^W\mathbf{1}_n]_+ \neq \mathbf{0}_n \\ \mathbf{0}_n & \text{otherwise.} \end{cases} \tag{1.3}$$

Similarly for the minimizer for $T_{t:}$ is

$$T_{t:} = \begin{cases} \frac{[W_{:t}^T R_t - \ell_1^T \mathbf{1}_d^T]_+}{||W_{:t}||_2^2 + \ell_2^T} & \text{if } [W_{:t}^T R_t - \ell_1^T \mathbf{1}_d^T]_+ \neq \mathbf{0}_d^T \\ \mathbf{0}_d^T & \text{otherwise.} \end{cases} \tag{1.4}$$

By $[\mathbf{u}]_+$ we mean the projection of $\mathbf{u}$ onto the positive orthant, i.e. $u_j = \max(u_j, 0)$.

In practice, RRI-NMF and other similar alternating non-negative least squares methods [8], [9] converge faster and to better solutions than the original multiplicative algorithms for NMF [4].

Adapting RRI-NMF to Topic Modeling Since the above update rules aren't designed for topic modeling we must modify them to meet additional constraints. We make the following changes:

Sum to 1 constraints, $T \subseteq \Delta^d$, and $W \subseteq \Delta^k$: The above update rules don't enforce the sum-to-1 constraints. Notice that if all the regularization terms are 0, the objective is invariant under the scaling $WT = (WS)(S^{-1}T)$. We take advantage of this invariance to satisfy the $T \subseteq \Delta^d$ constraint by setting $W_{:t} \leftarrow W_{:t}||T_{t:}||_1$, and $T_{t:} \leftarrow T_{t:}/||T_{t:}||_1$. To satisfy the $W \subseteq \Delta^k$ constraint we notice that Equation 1.2 is a convex objective, and the constraint set Δ^k itself is also convex. This means we can find the optimal $W_{:t}$ by first finding the optimal unconstrained $W_{:t}$ for all t using Equation 1.3 and then for all i projecting

$$W_{i:} \leftarrow \min_{x \in \Delta^k} ||x - W_{i:}||_2.$$

This projection can be done efficiently, e.g. [10].

ℓ_1 regularization In the presence of sum-to-1 constraints, ℓ_1 regularization on W and T have no meaning, so we drop these regularization parameters for the topic modeling setting.

Negative 2nd Derivative When the denominators of Equations 1.3, 1.4 are negative, We cannot simply set the gradient equal to zero and solve for $W_{:t}$ (resp. $T_{t:}$). As a result Equations 1.3, 1.4 are no longer expressions for the minima. This scenario wasn't considered in [7] since they were interested in general purpose NMF algorithms. However

in the presence of the sum-to-1 constraints, negative ℓ_2 regularization is interesting because it pushes vectors towards the standard basis vectors, and hence encourages sparsity. However negative ℓ_2 regularization can make the 2nd derivatives negative, so we must solve a slightly more general optimization problem. The general problem optimization problem we must now solve is of the form

$$\min_{x \geq 0, ||x||_1 = 1} w^T x + c x^T x.$$

When $c < 0$ this is not efficiently solvable by quadratic programming, so we derive the minimum by considering the KKT conditions. This result is presented as Lemma 1.

Lemma 1. *Let* $\mathbf{w} \in \mathbf{R}^d$, $c \in \mathbf{R}$. *Consider the optimization problem*

$$\begin{aligned} \min_{\mathbf{x} \in [0,1]^d} \; & f(x) \equiv \mathbf{w}^T \mathbf{x} + c \mathbf{x}^T \mathbf{x} \\ s.t. \; & h_1(\mathbf{x}) \equiv \mathbf{x}^T \mathbf{1}_d - 1 = 0 \\ & g_i(\mathbf{x}) \equiv -x_i \leq 0 \qquad\qquad \forall i \in 1, \ldots, d. \end{aligned}$$

When $c \leq 0$, f *is minimized by setting* $x_j = 1$ *where* $j = \arg\min_i \{w_i\}$ *and setting* $x_{i \neq j} = 0$.

Proof. The KKT conditions are

Stationarity $-\nabla f = \sum_i u_i \nabla g_i + \lambda \nabla h \iff -w_i - 2cx_i = -u_i + \lambda \, \forall i.$

Primal Feasibility $||\mathbf{x}||_1 = 1$, $x_i \geq 0 \, \forall i$.

Dual Feasibility $u_i \geq 0 \, \forall i$.

Complementary Slackness $u_i x_i = 0$.

We will look for a feasible solution $(\mathbf{x}, \mathbf{u}, \lambda)$ of the KKT conditions by eliminating u.

1. From stationarity, $u_i = w_i + 2cx_i + \lambda$.
2. From dual feasibility, $w_i + \lambda \geq -2cx_i$.
3. Suppose that for all i we have $w_i + \lambda > -2cx_i$, then for all i $u_i > 0$ and by complementary slackness $x_i = 0$; this violates primal feasibility.

4. Therefore, there exists some j for which $w_j + \lambda = -2cx_j$, and $x_j > 0$. In particular let $j = \arg\min_i w_i$, and set $x_j = 1$, and set $x_{i \neq j} = 0$.

5. Now we all conditions are ensured, except we must verify dual feasibility. Consider $u_{i \neq j} = w_i + 2c0 + \lambda$. Since $w_j + \lambda = -2c$, $w_j + \lambda > 0$ since $c < 0$. And since $w_i \geq w_j$, $w_i + \lambda \geq 0$, and therefore $u_i \geq 0$.

□

Corollary 1. *Suppose we are minimizing the same objective as in Lemma 1, with $c \leq 0$. However now the constraints are $0 \leq x_i \leq 1$, i.e. there is no $||\mathbf{x}||_1 = 1$ constraint. Then the optimal setting is $x_i = 1$ if $w_i + c \leq 0$, and $x_i = 0$ otherwise.*

Proof. The problem decouples into separate problems for each component of $\mathbf{x}$. Suppose the problem for the ith component is

$$\min_{x_i} f_i(\mathbf{x}) \equiv w_i x_i + c x_i^2.$$

Since $c \leq 0$, the solution will be at the boundary $x_i = 0$ or $x_i = 1$. The value at $x_i = 1$ is $w_i + c$ and the value at $x_i = 0$ is 0. Therefore we set $x_1 = 1$ whenever $w_i + c \leq 0$. □

The RRI NMF algorithm modified for topic modeling is summarized as Algorithm 2. Once again we need to provide an initial W_0 and T_0. In [7] the author suggests starting with uniformly random W_0, T_0 and scaling each by $\sqrt{\alpha}$ where $\alpha = \text{argmin}_\alpha ||X - \alpha W_0 T_0||_F^2$. If we let $\mathbf{x}$ be the column representation of X and $\mathbf{y}$ be the column representation of $W_0 T_0$ then $\alpha = \frac{\mathbf{x}^T \mathbf{y}}{\mathbf{y}^T \mathbf{y}}$. In our experiments we found that a initialization based on the non-negative SVD of the matrix X [11] leads to faster convergence to better local minima, as a result we use the SVD-based initialization for Chapter 1. In Section 2.1 we will explore additional initializations for RRI NMF.

1.2.3 Probabilistic Latent Semantic Analysis

The idea of modeling of topics as probability distributions over words, and documents as probability distributions over topics was popularized by Hofmann in [12], it is defined in Figure 1.5. Similar to NMF, the motivation is to create a more interpretable model than latent semantic analysis [3] which was based on the SVD. In fact NMF can be seen as

Algorithm 2 Rank-one Residue Iterations for NMF

Input: $X \in [0,1]^{n \times d}, k \in \mathbf{Z}_{\geq 0}$, ℓ_2^T, and ℓ_2^W.
Output: $W \in [0,1]^{n \times k}, T \in [0,1]^{k \times d}$ are local minimizers of

$$\frac{1}{2}\left(||X - WT||_F^2 + \ell_2^W ||W||_F^2 + \ell_2^T ||T||_F^2\right).$$

repeat
  for $t \in \{1, 2, \ldots, k\}$ **do**
    $R_t = X - \sum_{t'=1, t' \neq t}^{d} W_{:t'} T_{t':}$.
    if $[R_t T_{t:}]_+ \neq \mathbf{0}_n$ **then**
      if $||T_{t:}||_2^2 + \ell_2^W > 0$ **then**
        $W_{:t} \leftarrow \frac{[R_t T_{t:}]_+}{||T_{t:}||_2^2 + \ell_2^W}$
        if $\ell_2^T = 0 \wedge \ell_2^W = 0$ **then**
          $W_{:t} \leftarrow s_t W_{:t}$ ▷ Scale invariance only in the absence of regularization.
        end if
      else
        $W_{:t} \leftarrow$ Corollary 1 with $\mathbf{w} = -R_t T_{t:}$, $c = ||T_{t:}||_2^2 + \ell_2^W$.
      end if
    else
      $W_{:t} \leftarrow \mathbf{0}_n$
    end if
    if $[R_t^T W_{:t}]_+ \neq \mathbf{0}_d$ **then**
      if $||W_{:t}||_2^2 + \ell_2^T > 0$ **then**
        $T_{t:} \leftarrow \frac{[R_t^T W_{:t}]_+}{||W_{:t}||_2^2 + \ell_2^T}$
      else
        $T_{t:} \leftarrow$ Lemma 1 with $\mathbf{w} = -R_t^T W_{:t}$, $c = ||W_{:t}||_2^2 + \ell_2^T$.
      end if
      $s_t \leftarrow |T_{t:}|_1$
      $T_{t:} \leftarrow \min_{x \in \Delta^d} ||x - T_{t:}||_2$.
    else
      $s_t \leftarrow 1$
      $T_{t:} \leftarrow \mathbf{0}_d$
    end if
  end for
  for $i \in \{1, 2, \ldots, n\}$ **do**
    $W_{i:} \leftarrow \min_{x \in \Delta^k} ||x - W_{i:}||_2$.
  end for
until Convergence criteria are met.

Input: (1) Document matrix $X \subseteq \Delta^d$.
(2) Number of topics $k \in \mathbf{Z}_{\geq 0}$.

Objective: $\max_{W,T} \mathcal{L}(X|W,T) = \max_{W,T} \sum_{i=1}^{n} \sum_{j=1}^{d} X_{ij} \log \left[\sum_{t=1}^{K} W_{it} T_{tj} \right]$.

Constraints: (1) Topics are probability distributions over d features. For all $t \in \{1, 2, \ldots, k\}$: $T_{t:} \in \Delta^d$.
(2) Documents are probability distributions over topics. For all $i \in \{1, 2, \ldots, n\}$: $W_{i:} \in \Delta^k$.

Fig. 1.5: PLSA Problem Statement

equivalent to PLSA [13] if the ℓ_2 reconstruction error is replaced by D_{KL} and regularization terms are omitted.

The PLSA model is parameterized by the document-topic multinomials in each row of $W \in [0,1]^{n \times K}$ and the topic-word multinomials in each row of $T \in [0,1]^{K \times d}$. Given this model the ith document of length N_i is generated as follows:

1. Repeat N_i times:
   (a) Sample topic z_j from $W_{i:}$, the multinomial distribution described by the ith row of W.
   (b) Sample the jth word of document i from $T_{z:}$.

For a given model W, T we can write the likelihood of our data X:

$$P(X|W,T) = \prod_{i=1}^{n} P(\text{doc} = i) \prod_{j=1}^{d} \left[\sum_{t=1}^{K} P(\text{topic} = t|\text{doc} = i) P(\text{word} = j|\text{topic} = t) \right]^{X_{ij}} \tag{1.5}$$

$$= \prod_{i=1}^{n} \frac{1}{n} \prod_{j=1}^{d} \left[\sum_{t=1}^{K} W_{it} T_{tj} \right]^{X_{ij}}. \tag{1.6}$$

We ignore the $1/n$ factor since its omission doesn't affect the inference algorithm. Then we

take the logarithm of the likelihood to get the log-likelihood:

$$\mathcal{L}(X|W,T) = \sum_{i=1}^{n}\sum_{j=1}^{d} X_{ij} \log \left[\sum_{t=1}^{K} W_{it}T_{tj}\right]. \tag{1.7}$$

To optimize $\mathcal{L}$ we need to be able to take derivatives of it in terms of W_{ij} and T_{tj}. Since these terms occur inside a sum inside a logarithm the resulting expressions are complex. Instead we use the Expectation Maximization (EM) algorithm [14]. The key is to assume that we know the topic that generated each word, and instead of maximizing $\mathcal{L}$ we maximize the expectation of $\mathcal{L}$ over the probability that topic t generated word j in document i. We write this expectation as:

$$E[\mathcal{L}] = \sum_{i=1}^{n}\sum_{j=1}^{d}\sum_{t=1}^{K} X_{ij}P(z=t|i,j)\log W_{it}T_{tj} \tag{1.8}$$

The EM algorithm iterates between updating $P(z=t|i,j)$ based on our model W,T, and then updating W,T based on the data and $P(z=t|i,j)$. The resulting algorithm is summarized as Algorithm 3, see [15] for the derivation.

Algorithm 3 EM for PLSA

Input: $X \in [0,1]^{n\times d}$, $\forall i \sum_{j=1}^{d} X_{ij} = 1$, $k \in \mathbf{Z}_{\geq 0}$.
Output: $W \in [0,1]^{n\times k}$, $\forall i \sum_{i=1}^{k} W_{ij} = 1$, $T \in [0,1]$, $\forall z \sum_{j=1}^{d} T_{zj} = 1$

repeat
  for $t = \{1,\ldots,K\}$ **do** ▷ E-Step
    for $i = \{1,\ldots,n\}$, $j = \{1,\ldots,d\}$ **do**
      $P(z=t|i,j) \leftarrow \frac{W_{it}T_{tj}}{\sum_{k=1}^{K} T_{kj}W_{ik}}$
    end for
  end for
  for $t = \{1,\ldots,k\}, j = \{1,\ldots,d\}$ **do** ▷ M-Steps
    $T_{tj} \leftarrow \frac{\sum_{i=1}^{n} X_{ij}P(z=t|i,j)}{\sum_{j'=1}^{d}\sum_{i=1}^{n} X_{ij'}P(z=t|i,j')}$
  end for
  for $t = \{1,\ldots,k\}, i = \{1,\ldots,n\}$ **do**
    $W_{it} \leftarrow \frac{\sum_{j=1}^{d} X_{ij}P(z=t|i,j)}{\sum_{j=1}^{d} X_{ij}}$
  end for
until Convergence criteria are met.

It can be shown [14] that the EM algorithm maximizes $\mathcal{L}$ in terms of W, T. The Kullback-Leibler divergence between X and WT is

$$D_{KL}(X||WT) = \sum_{i=1}^{n}\sum_{j=1}^{d} X_{ij} \log \frac{X_{ij}}{(WT)_{ij}} \tag{1.9}$$

$$= \sum_{i=1}^{n}\sum_{j=1}^{d} X_{ij} \log X_{ij} - \sum_{i=1}^{n}\sum_{j=1}^{d} X_{ij} \log (WT)_{ij} \tag{1.10}$$

$$= -H(X) - \mathcal{L}(X|W, T). \tag{1.11}$$

Thus an algorithm that maximizes the likelihood of the data can be thought of as minimizing the KL-divergence between the data and the model. LDA (Section 1.2.4) is another topic model that minimizes the KL-divergence between data and model, though its model is silghtly more complex than in PLSA, and we don't use EM to fit the model. This is in contrast to both K-means (Section 1.2.1) and non-negative matrix factorization (Section 1.2.2), which minimize Frobenius norms between the data X and its reconstruction WT.

Notice that the above algorithm is not fully defined. In order to perform the initial E-step, the algorithm requires W, T to be initialized. In our experiments we tet T to have random [0,1] entries, and normalize rows. We set each entry of W to be $\frac{1}{k}$. An alternative is to initialize with the W, T returned by an other algorithm; although not illustrated in the results section, initializing from the Kmeans++ solution worked well.

In his paper Hofmann refines the above algorithm by adding a objective to the log-likelihood of the fit that penalizes low entropy solutions. The idea is that a low entropy solution will be peaky and fit the training data better, but may have worse out of sample error. In this overview we don't explore the merits of this smoothing technique, however Hofmann found it to do well in terms of reducing perplexity (introduced later). Instead we consider LDA as an example of a D_{KL}-minimizing topic model with smoothing.

1.2.4 Latent Dirichlet Allocation

Latent Dirichlet Allocation (LDA) [16] extends the generative model of PLSA by assigning a Dirichlet prior to both the document-topic distributions represented by the matrix W and the topic-word distributions represented by matrix T, it is defined in Figure 1.6. Doing this

Input:
(1) Document matrix $X \subseteq \Delta^d$.
(2) Number of topics $k \in \mathbf{Z}_{\geq 0}$.
(3) Symmetric Dirichlet prior parameter for W, α and for T, β.

Objective:
$$\max_{W,T} \mathcal{L}(X|W,T) = \max_{W,T} \sum_{i=1}^{n} \sum_{j=1}^{d} X_{ij} \times \log \left[\sum_{t=1}^{K} W_{it} T_{tj} P(w = W_{it}|\alpha) P(t = T_{tj}|\beta) \right].$$

Constraints:
(1) Topics are probability distributions over d features. For all $t \in \{1, 2, \ldots, k\}$: $T_{t:} \in \Delta^d$.
(2) Documents are probability distributions over topics. For all $i \in \{1, 2, \ldots, n\}$: $W_{i:} \in \Delta^k$.

Fig. 1.6: LDA Problem Statement

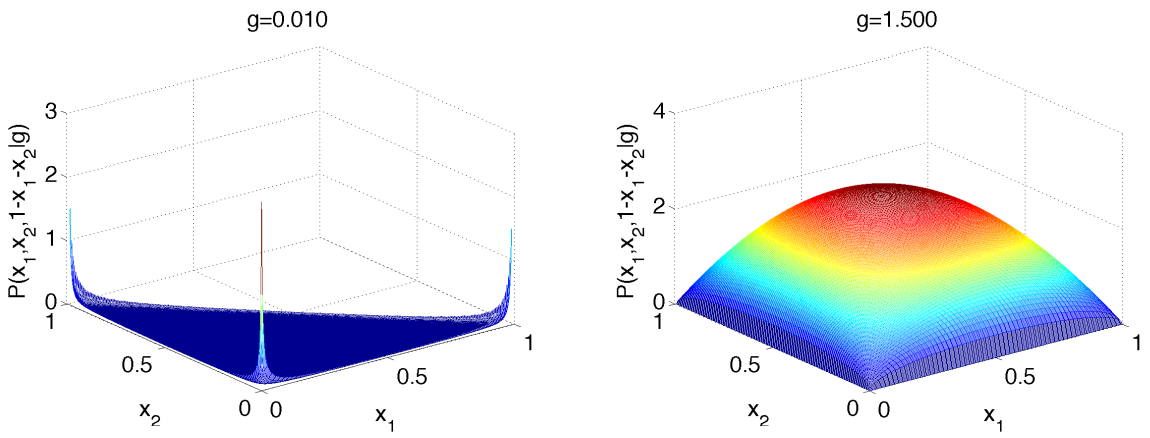
defines a way to generate new documents, as well as determine the probability of documents that weren't used to train the model.

First we describe LDA's generative model. The ith document of length N_i is generated as follows:

1. Sample distribution $W_{i:}$ from $\text{Dir}(\alpha)$.
2. Repeat N_i times:
   (a) Sample topic z_j from $W_{i:}$.
   (b) Sample the jth word of document i from $T_{z:}$.

Each row of T is sampled from $\text{Dir}(\beta)$ independent of the above process. Here we are using symmetric Dirichlet distributions, so $\text{Dir}(\alpha)$ is shorthand for $\text{Dir}(\alpha \mathbf{1}_k^T)$.

The Dirichlet priors on W and T and their hyper-parameters, α and β, respectively, are what makes LDA different from the other topic models we have discussed so far. Since the Dirichlet distribution is a conjugate prior to the multinomial distribution, we can interpret α to represent the expected number of times a topic occurs in a document, and β to represent the number of times a word occurs in a topic. Since we are using a symmetric Dirichlet, as we scale α, β the expectation of the draw doesn't change. However, the variance of each x_i



The x_1 and x_2 axes show the values of the first two components drawn from $Dir(\gamma)$, $x_3 = 1 - x_1 - x_2$. The z-axis shows the Dirichlet pdf evaluated at each point $\mathbf{x}, \gamma$. The first plot $\gamma = 0.01$ is close to the value we use for β, the prior for T. It is very peaked at points with only one non-zero component. The $\gamma = 1.5$ case is close to the α hyper-paramater when it is around $50/k$. The Dirichlet distribution inflects at $\gamma = 1$.

Fig. 1.7: PDF of the Dirichlet Distribution

of $\mathbf{x} \sim Dir(\gamma)$ is proportional to $1/\gamma$, so for smaller γ we expect to see sparser draws, see Figure 1.7. It isn't obvious what these hyper-parameters should be, and we are interested in finding good values for them. The authors of the Gibbs sampling algorithm we use for LDA [17] suggest $\alpha = 50/k$ and $\beta = 0.01$. There are methods for learning α, β specifically for each dataset that rely on the output of the Gibbs Sampling algorithm we describe below, as described in [18] and [19]. In Section 1.4 we will examine a fixed point iteration method, as well as the evaluation-based methods from Section 1.3.

There are different algorithms for fitting the LDA model for a given α, β. For this overview we consider Collapsed Gibbs Sampling as presented in [17], and [20]. In [21] Asuncion, et al., show that Collapsed Gibbs Sampling is equivalent to other LDA inference algorithms in terms of perplexity, and it is one of the faster inference algorithms for LDA.

Under the LDA model, the probability of the wth word in the ith document being τ is

$$P(E_{iw} = \tau|\alpha, \beta) = P(W_{i:}|\alpha)P(T|\beta)\sum_{k=1}^{K} P(z_{iw} = k)P(\tau|z_{iw} = k) \tag{1.12}$$

where z_{iw} refers to the topic assigned to the wth word in the ith document. Thus by approximating $P(z_{iw} = k)$ for $i \in \{1, \ldots, n\}$, $w \in \{1, \ldots, N_i\}$, and $k \in \{1, \ldots, K\}$, we are

finding the LDA model that fit our data.

The method we use to approximate $P(z_{iw})$ is Collapsed Gibbs Sampling. The idea is that we have a vector $\mathbf{z} \in \{1, \ldots, K\}^{\sum_{i=1}^{n} N_i}$ where z_{iw} is the topic assigned to the wth token of the ith document. We then form a Markov chain with transition probabilities proportional to $P(z_{iw} = k|\mathbf{z}_{\neg iw}, X)$ where X is our document collection, and $\mathbf{z}_{\neg iw}$ are the components of $\mathbf{z}$ excluding iw. In other words in the Markov chain we change one coordinate of $\mathbf{z}$ at a time, conditioned on the current state of all the other components. Starting with a initial $\mathbf{z}_0$, after $|\mathbf{z}|$ coordinate updates, we get a new state $\mathbf{z}_1$, and after $L|\mathbf{z}|$ coordinate updates for some L, we get z_{L+1}, where $\mathbf{z}_1$ and $\mathbf{z}_{L+1}$ are sufficiently independent. The Gibbs Sampling algorithm ensures that we are drawing from a distribution proportional to $P(\mathbf{z})$, so that after I transitions of the Markov chain we have $\frac{I}{|\mathbf{z}|L}$ independent samples from $P(\mathbf{z})$. Finally, we can form an unbiased estimator for $P(\mathbf{z})$ by estimating it as a uniform distribution over our independent samples. The difficulty with this method is finding L, the inter-sample period, and determining a good number of samples.

To use this method, it is necessary to find a expression proportional to $P(z_{iw} = k|\mathbf{z}_{\neg iw}, \mathcal{X})$. We do not repeat the derivation, which involves integrating over W, and T, and rather present the result. A clear derivation can be found in [18]. Let $n_{i,j,k} \in \mathbf{Z}_{\geq 0}$ represent the number of times in $\mathbf{z}$ the jth dictionary token was generated by topic k in document i. Also let $n_{:,j,k} = \sum_{i=1}^{n} n_{i,j,k}$ be the number of times token j was generated by topic k across all documents, and $n_{i,:,k} = \sum_{j=1}^{d} n_{i,j,k}$ be the number of times topic k generated a word in document i. Finally define the count that excludes the current token iw, where j is the dictionary index of the current token, as

$$n_{i,j,k}^{\neg iw} = \begin{cases} n_{i,j,k} & \text{if } z_{iw} \neq k \\ n_{i,j,k} - 1 & \text{otherwise} \end{cases}$$

Then we have

$$P(z_{iw} = k|\mathbf{z}_{\neg iw}, X, \alpha, \beta) = \frac{n_{:,k,j}^{\neg iw} + \beta_j}{\sum_{j'=1}^{d} n_{:,k,j'}^{\neg iw} + \beta_{j'}} (n_{i,:,k}^{\neg iw} + \alpha_k). \tag{1.13}$$

Since we are using symmetric priors in our case this reduces to

$$= \frac{n_{:,k,j}^{\neg iw} + \beta}{d\beta + \sum_{j'=1}^{d} n_{:,k,j'}^{\neg iw}} (n_{i,:,k}^{\neg iw} + \alpha). \tag{1.14}$$

Ultimately we are interested in finding the multinomial distributions represented by the W, T matrices, not $P(\mathbf{z})$. To find these distributions we use Bayes' Theorem

$$P(W_{i:}|\mathbf{z}, \alpha) = \frac{1}{Z} P(\mathbf{z}|W_{i:}, \alpha) P(W_{i:}|\alpha) \tag{1.15}$$

$$= \frac{1}{Z} \prod_{k=1}^{K} W_{ik}^{n_{i,:,k}} \mathrm{Dir}(W_{i:}|\alpha) \tag{1.16}$$

$$= \mathrm{Dir}(W_{i:}|\alpha + \mathbf{n}_i). \tag{1.17}$$

Similarly for T:

$$P(T_{k:}|\mathbf{z}, \beta) = \frac{1}{Z'} \prod_{j=1}^{d} T_{kj}^{n_{:,k,j}} \mathrm{Dir}(T_{k:}|\beta) \tag{1.18}$$

$$= \mathrm{Dir}(T_{k:}|\beta + \mathbf{n}^{(k)}). \tag{1.19}$$

To get realizations of W, T, we take the expected value of their distributions, and for the Dirichlet distribution we have

$$E[\mathrm{Dir}(x_i|\mathbf{v})] = \frac{v_i}{\sum_{i'} v_{i'}}.$$

Equations (1.15-1.19) are written as if we only had one sample $\mathbf{z}$ from $P(\mathbf{z})$, which is what we use in our experiments, however it is simple to average across m samples of $\mathbf{z}$; the difficulty lies in ensuring the samples are sufficiently independent. The Collapsed Gibbs Sampling algorithm is summarized as Algorithm 4 (p.g. 20).

1.2.5 Visual Comparison of Topic Models

To conclude this survey of topic models and inference algorithms, we depict them on a simple input. We consider two datasets, the first is is a mixture of three 2-dimensional Gaussians on Δ^3. The second is generated according to the LDA generative model.

In Figure 1.8 (p.g. 21) we depict the result of running each model's inference algorithm until

Algorithm 4 Collapsed Gibbs Sampling for LDA

Input: I, J, k, α, β. $I_i \in \{1, \ldots, n\}$ and $J_i \in \{1, \ldots, d\}$ contain the document and dictionary indices of the ith token.

Output: $W \in [0,1]^{n \times k}, T \in [0,1]^{k \times d}$ are based on topics drawn from the LDA model that generated the data.

```
Let N = |I| = |J|
Initialize z ∈ {1, ..., k}^N randomly.
repeat
    for i ∈ {1, ..., N} do
        Update z_i using Equation 1.14.
    end for
until stationary distribution reached. (500-1000 iterations in practice, [17])
```

convergence. Diamonds represent topic locations, numbers by topics indicate how many documents are assigned to them in expectation. Points are colored according to mixing weights. Lines connect documents to their reconstructions under the topic model. Note that with NMF reconstruction lines are perpendicular to the topic line unless they map to a vertex, whereas with PLSA and LDA reconstructions are not necessarily the closest points in terms of Euclidean distance.

Details to notice are:

- PLSA and LDA have visually similar outputs.
- K-means reconstructs each document as its assigned topic.
- The NMF topic simplex bisects the points so that all reconstruction vectors are perpendicular to the topic simplex.
- The PLSA and LDA reconstructions of points usually aren't the closest points on the simplex in terms of ℓ_2 norm.
- Despite optimizing different objectives, the LDA and NMF solutions look close to each other on the LDA-generated dataset.

This visualization is meant to serve as a intuitive summary for the reader. However some of the visual intuitions will be used when evaluating algorithms as well as improving them.

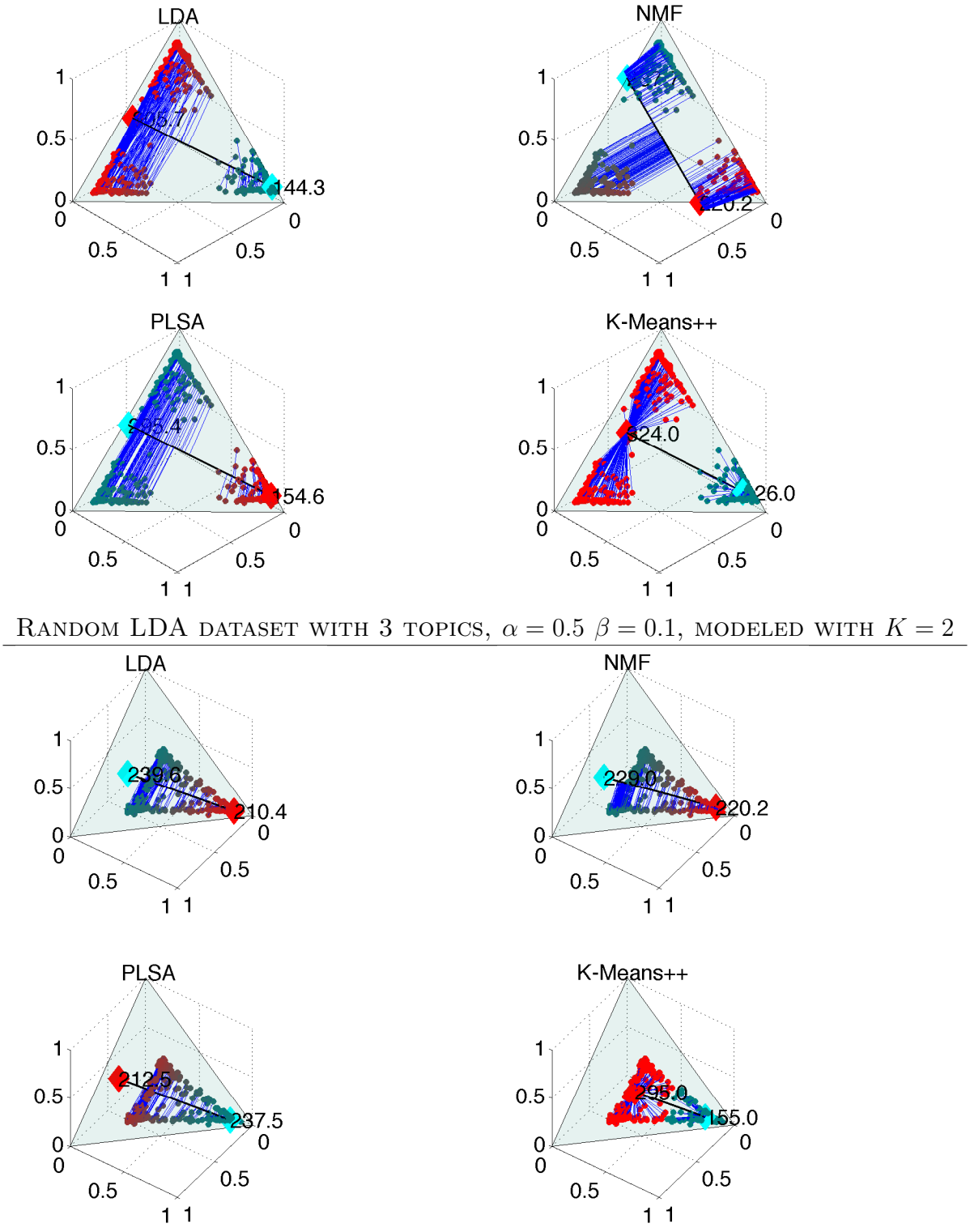


Fig. 1.8: Visual Comparison of Topic Models in 3-dimensions

1.2.6 Further Topic Models in the Literature

The main properties of a topic model are grouping similar documents together, and providing summaries of the topics within a corpus. Many models have been studied under diverse scenarios that fit this definition.

Topic Models with Additional Constraints One vein of research is to use topic models in relation with 'concepts' that are input by humans. In [22] there are 12000 human defined concepts that are tree-based ontologies from the Open Directory Project and the Cambridge Advanced Learners Dictionary. In [23] concepts come from the Leuven Natural Concept Database. In [24] 202 concepts for the InfoVis conference are defined by experts. One task considered is finding documents explained by concepts, and finding topics that the concepts don't explain well. Another task is to measure the correspondence between human input concepts and automatically generated topics.

In clustered and correlated topic models [25], [26] the idea is that it is more sensible for some topics to co-appear together in documents than it is for others. As a result the aim is to identify topics that are related to each other, while allowing them to be semantically distinct and individually more coherent.

Some effort has been put into building topic models that infer the number of topics based on a prior assumptions. In [19], [27] a Heirarchical Dirichlet Process is used to represent a topic model with infinitely many topics (of which only a finite number have non-negligible weight). Heirarchical clustering based on Ward's criterion [28] creates clusters similar in shape to k-means. However there is no free lunch, since both of the above families of methods require a way to decide which topics should be output. The advantage is that re-parameterizing how we express the number of topics to be output may be more meaningful and adaptable to a multiple datasets than simply requiring a number.

Topic Models with Additional Metadata Author-topic models [29] take as input the author of each document in addition to the words of the documents. The goal is to identify what topics individual authors typically write about. One way to do this is by treating each author as a 'meta-document' where all the documents written by them are modeled together.

Dynamic topic models [30] take as input the timestamp when each document was written. The goal is to identify how topics evolve over time. An interesting challenge is to identify the same topic (whose word distribution may have changed) at different times.

Topic Models Based on Automated Differentiation The explosion in the technology of automated differentiation (i.e. neural networks, deep learning) allows us to study models whose complexity would have previously limited our ability to infer their parameters.

One vein of work [31], [32] aims to build more expressive representations for how words are drawn from topics, and what it means for documents to be composed of topics. As a result many of the above settings, such as additional metadata, or correlation constraints can be added as additional inputs into the generative expressions. The careful work that was previously done to model their influence can now be done automatically. But care must be taken to ensure that the results are meaningful.

Another vein of work aims to drop the bag-of-words language model, and instead make use of sequential and structural properties of text. In [33] Paragraph Vectors, i.e. embeddings of documents into a latent space, were shown to perform better (on a variety of tasks) than the representation of documents in terms of LDA topics. In order to use Paragraph Vectors for topic modeling, an additional challenge is to summarize the latent space (i.e. the identity of the topics) rather than just produce embeddings.

1.3 A Survey of Topic Model Evaluation Techniques

Since a topic model's definition (Equation 1.1 (p.g. 2)) leaves details, open it isn't clear what criteria should be used to evaluate a topic model. We present five types of evaluation techniques that can be used to select the most useful topic model for a particular case. The first four types of techniques are measures with an associated algorithm that outputs a number. The last type of techniques is subjective, it is based on an expert inspecting and assessing a model.

This Section is organized as follows: first we examine measures based on in-sample objective, (Section 1.3.1). Second, measures based on out-of-sample error, or generalization error (Section 1.3.2). Third, measures based on the application of using topic models for docu-

ment classification tasks (Section 1.3.3). Fourth, measures based on the semantic coherence of the found topics (Section 1.3.4). Finally, we study visual evaluation of topic models (Section 1.3.6).

1.3.1 In-sample Objective Evaluation Measures

We consider the in-sample objective measures used by the two classes of topic models, those based on the log-likelihood and those based on the Frobenius reconstruction error. Although the in-sample objective cannot typically be used to select between different topic models, it can be used to compare different instantiations of a particular model.

1. By looking at the standard deviations of in-sample errors we can characterize the stability of our algorithms with respect to random initialization, or alternative initialization algorithms.

2. We can also try to predict whether algorithms with the best in-sample error will perform best on out-of-sample evaluation measures.

The first in-sample objective we consider is the Frobenius reconstruction error (referred to as Fro^2), defined in Figure 1.9. A well-known lower-bound on the Fro^2 error is based on the k-truncated SVD. The lower bound states that the rank k reconstruction of a matrix X that minimizes the Fro^2 objective is the SVD with singular values $\{\sigma_t\}_{t=k+1}^{\text{rank}(X)}$ set to 0. Since the SVD uses negative basis vectors, in general we expect topic models that use only positive basis vectors to have a higher Fro^2 reconstruction error. For example, in our experiments NMF attains a Fro^2 objective with k topics that the $0.3k$-$0.6k$ truncated SVD achieves. Without simplifying assumptions, it's hard to relate the number of topics NMF would require to match the k-truncated SVD, because this problem is equivalent to finding the non-negative rank of a matrix, which is NP-Hard [34]. We expect the loglikelihood-maximizing models to have good Fro^2 objectives as well [35].

Next we consider in-sample log-likelihood, defined in Figure 1.10. If X_{ij} represents the number of times the term j appears in document i, then the in-sample likelihood is defined as $\prod_{i=1}^{n}\prod_{j=1}^{d}(WT)_{ij}^{X_{ij}}$. Again, we expect that models that minimize Fro^2 will still achieve good loglikelihoods.

Input:	(1) Document matrix $X \subseteq \Delta^d$. (2) Topic model $W \subseteq \Delta^k$, $T \subseteq \Delta^d$.
Measure:	$\text{Fro}^2(X\|W,T) = \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{d}(X_{ij} - W_{i:}T_{:j})^2$, lower is better.
Notes:	Given incomplete inputs (X, W) or (X, T) this objective is still well defined because the missing input can be found using non-negative least squares (or by the RRI NMF algorithm).

Fig. 1.9: Frobenius Reconstruction Error Objective

Input:	(1) Document matrix $X \subseteq \Delta^d$. (2) Topic model $W \subseteq \Delta^k$, $T \subseteq \Delta^d$.
Measure:	$\mathcal{L}(X\|W,T) = \sum_{i=1}^{n}\sum_{j=1}^{d} X_{ij} \log W_{i:}T_{:j}$, higher is better.
Notes:	We assume there are no priors on W and T, rather that each row simply defines categorical distribution.

Fig. 1.10: Loglikelihood Objective

1.3.2 Out-of-sample Evaluation Measures

Out-of-sample evaluation is based on holding out a test portion of a dataset, and computing an objective on that portion. The goal is to evaluate how well the topic model applies to unseen data.

The first class of objectives are based on using the in-sample objectives. Given a test set $E \subseteq \Delta^d$ and a set of topic vectors $T \subseteq \Delta^d$ we can find weights / mixing probabilities W_E that minimize either Fro^2 or $\mathcal{L}$. This can be done by adapting existing topic model inference algorithms and simply not allowing the algorithm to update T. We can then compute the objective using Fro^2 or $\mathcal{L}$ to see how well the found topics fit unseen data. This is summarized in Figure 1.11.

A historically and presently popular objective is called perplexity [16]. Intuitively the goal is to measure how much probability a language model assigns to each document per word. It is defined in Figure 1.12. It is similar to the per word log likelihood of the test set, however unlike in $\mathcal{L}(E|\cdot, T)$ we are not allowed to fit new weights W_E to the test set E. Rather, we must compute the expectation over the learned document-topic weights. This is

Input:	(1) Unseen document matrix $E \subseteq \Delta^d$. (2) Topics $T \subseteq \Delta^d$.
OoS $\mathcal{L}$:	$\mathcal{L}(E\|\cdot, T) = \mathcal{L}(E\|W_E, T)$; $W_E \leftarrow \arg\min_W \mathcal{L}(E\|W, T)$.
OoS Fro^2:	$\text{Fro}^2(E\|\cdot, T) = \text{Fro}^2(E\|W_E, T)$; $W_E \leftarrow \arg\min_W \text{Fro}^2(E\|W, T)$.

Fig. 1.11: Out-of-sample Evaluation of Model Objective

a sensible restriction, since if our collection hasn't seen any documents that have the 'polo' and 'chapstick' topics together, it should be surprised to observe such a document in the test set.

Input:	(1) Unseen document matrix $E \subseteq \Delta^d$. (2) Topic model $W \subseteq \Delta^k$, $T \subseteq \Delta^d$..
Measure:	$\text{Perplexity}(E) = \exp\left\{-\frac{\hat{\mathcal{L}}(E\|W,T)}{\|\|E\|\|_1}\right\}$, lower is better.
Notes:	$\hat{L}(E\|W, T) = \sum_{m=1}^{\|E\|}\sum_{j=1}^{d} E_{ij} \log\left(\frac{1}{n}\sum_{i=1}^{n} W_{i:}T_{:j}\right)$

Fig. 1.12: Perplexity

We compute the expectation over seen document-topic weights even for models with a prior distribution over these weights. For example, for LDA we can compute the probability of an unseen document E_m as

$$P(E|\alpha, T) = \prod_{j=1}^{d} \int_{\mathbf{w}} P(\mathbf{w}|\alpha)(\mathbf{w}T_{:j})^{N_j(E_m)} d\mathbf{w},$$

where $N_j(E_m)$ is the number of times token j occurs in document E_m. The above integral is not easy to compute, and is generally estimated using techniques as in [36]. For this overview, we want to treat the output of topic models (even LDA) just as the W, T distributions, and therefore we are interested in calculating the probability of a unseen document E given just W, and T, without prior distributions over them. In [16] the idea is that the document-topic weights observed on the training data are samples from the true document-topic distribution

for that dataset. Therefore we can approximate

$$P(E|W,T) = \prod_{m=1}^{|E|}\prod_{j=1}^{d}\int_{\mathbf{w}} P(\mathbf{w})(\mathbf{w}T_{:j})^{N_j(E_m)}d\mathbf{w} \tag{1.20}$$

$$\approx \prod_{m=1}^{|E|}\prod_{j=1}^{d}\left(\frac{1}{n}\sum_{i=1}^{n} W_{i:}T_{:j}\right)^{N_j(E_m)}. \tag{1.21}$$

We are assuming $P(\text{Doc} = i) = 1/n$, which is the highest entropy (least informative) assumption we can make in the absence of other information. Equation 1.21 states that the probability of a test document is the average of the probability of that test document if its topic distribution was one of the training documents. The probability of a test document is maximized if it is drawn from a distribution with probabilities proportional to its word counts. Thus we can minimize the perplexity of a test collection if our model is exactly that test collection. In practice we minimize perplexity if our model contains documents with low cross-entropies to the test documents; we do poorly if there are topic combinations or word combinations that we never consider in our training set, that occur in the test set. The priors in LDA attempt to smooth its model to account for both of these issues.

1.3.3 Classifier-based Evaluation Measures

Suppose each document is given a set of target class labels Y, such that $y_i \in \{1, \ldots, C\}$ is the class index of the ith document, and C is the total number of classes. We can ask how much does the topic distribution of a document reveal about its true class. In practice we are not given class labels, and we are not trying to solve the classification problem; rather we are interested in observing how well topic models capture the semantic relationships that humans detect between documents when they group documents together into classes. An issue we want to avoid is solving the topic to class correspondence problem, so we only consider measures μ that are invariant under permutation, i.e. for a $k \times k$ permutation Π we should have $\mu(W,T) = \mu(W\Pi, \Pi^T T)$.

The first measure we consider is called the Adjusted Rand Index, defined in Figure 1.13. Suppose we have our $n \times k$ matrix of topic distributions W, and we have a $n \times k$ matrix of class distributions Y. Let $A = W^T Y$ be the topic-class correlation matrix. Since rows of W, and Y sum to 1, entry A_{ij} can be interpreted as the number of documents assigned to topic

Input:
(1) Document-Class probabilities matrix $Y \subseteq \Delta^k$.
(2) Document-Topic probabilities $W \subseteq \Delta^k$.
(3) Alternatively, the topic-class correlation matrix $A = W^T Y$.

Measure: $\mathrm{ARI}(A) = \dfrac{\sum_i \sum_j \binom{A_{ij}}{2} - E[\mathrm{RI}(A)]}{\binom{n}{2} - E[\mathrm{RI}(A)]}$, higher is better.

Fig. 1.13: Adjusted Rand Index

i and class j simultaneously. The rand index is defined as

$$\mathrm{RI}(A) = \frac{\sum_i \sum_j \binom{A_{ij}}{2}}{\binom{n}{2}}. \tag{1.22}$$

The Rand index is normalized between 0 and 1, where 1 is a perfect match. The Rand index assumes W and Y are hard labels, meaning each row contains a single 1, so that the entries A_{ij} will be integers. In the case where at least one of W, Y contains probability distributions along the rows, the A_{ij} are no longer guaranteed to be integers, so we approximate $\binom{A_{ij}}{2}$ as $A_{ij}(A_{ij} - 1)/2$.

A minor problem with the Rand Index is that for a given Y, if we generate W by picking one of the elements in each row to be 1 uniformly at random, we expect the Rand Index to be about 0.86. [37] The Adjusted Rand Index arises from subtracting the expected Rand Index from both numerator and denominator of 1.22. In [38] the authors derived the expectation in terms of A. Let $a_i = \sum_j A_{ij}$, similarly $b_j = \sum_i A_{ij}$, then

$$E[\mathrm{RI}(A)] = \frac{\sum_i \binom{a_i}{2} \sum_j \binom{b_j}{2}}{\binom{n}{2}}. \tag{1.23}$$

Finally combining Equation 1.23 with Equation 1.22 we get

$$\mathrm{ARI}(A) = \frac{\sum_i \sum_j \binom{A_{ij}}{2} - E[\mathrm{RI}(A)]}{\binom{n}{2} - E[\mathrm{RI}(A)]}. \tag{1.24}$$

Besides the rand index a measures that are invariant under permutation are based on the mutual information of two random variables. A particularly popular measure is called the variation of information (VI) [39], defined in Figure 1.14. VI is based on the conditional

Input:	(1) Document-Class probabilities matrix $Y \subseteq \Delta^k$. (2) Document-Topic probabilities $W \subseteq \Delta^k$. (3) Alternatively, the topic-class correlation matrix $A = W^T Y$.
Measure:	$\text{VI}(A) = H(W\|Y) + H(Y\|W)$, lower is better. Where $H(W\|Y) = \sum_{c=1}^{C} H(W\|Y = c)P(Y = c)$, and $H(Y\|W) = \sum_{t=1}^{k} H(Y\|W = t)P(W = t)$.

Fig. 1.14: Variation of Information

entropy between the two random variables defined as

$$H(W|Y) = \sum_{c=1}^{C} H(W|Y = c)P(Y = c) \tag{1.25}$$

where $H(X) = -\sum_{x \in \Omega_X} P(x) \ln P(x)$. Then VI is defined as

$$\text{VI}(W, Y) = H(W|Y) + H(Y|W) \tag{1.26}$$

This can be computed from the contingency matrix A, using the $a/\sum_i a_i$ row and $b/\sum_j b_j$ column sum proportions for the $P(Y = c)$ factor of Equation 1.25 and then computing the entropies of normalized rows and columns.

VI is bounded between 0 and $\ln C + \ln K$ and is 0 iff $W = Y$, and both have exactly one non-zero per row.

1.3.4 Language Modeling-based Evaluation Measures

One purpose of a language model is to create a summary of the dataset being modeled. In order to be useful this model should be interpretable to a human. A way to measure human interpretability is to relate the topic model to a reference that we known is interpretable to humans because it was created by humans. To do this for topic models we first notice that both topics and documents are elements of Δ^d. Then we can assess how similar the groups (i.e. pairs, triples, etc) of words used to define topics are to the groups that appear in documents, this is called coherence. For purposes of exposition and motivation we explore two early attempts to formalize this notion, and then present the state-of-art formulation.

Perhaps the simplest attempt to define coherence comes from [40]. For a specific topic T_t: and its M most probable words $w_1, w_2, \ldots, w_M$, let $D(w)$ be the number of documents that word w occurs in, and let $D(w, w')$ be the number of documents where w and w' occur together. Coherence (Mimno) is defined as

$$C_{\text{Mimno}}(\text{Topic } t) = \sum_{m=2}^{M} \sum_{l=1}^{m-1} \log \frac{D(w_m, w_l) + 1}{D(w_l)}. \tag{1.27}$$

We define the coherence for a solution W, T as

$$E[C(\text{Mimno})] = \sum_{t=1}^{K} C_{\text{Mimno}}(t) \sum_{i=1}^{n} W_{it}/n. \tag{1.28}$$

Here we are simply weighing the coherence of each topic by the proportion of the collection it is responsible for. When computing $D()$ in Equation 1.28, we base it on the same dataset that was used for training.

A similar but slightly different definition of coherence (Newman) comes from [41]. Let $P(w_i, w_j)$ be the probability of seeing words i and j together in a given context, and $P(w_k)$ be the probability of seeing word k in any context. Then the point-wise mutual information between w_i, w_j, can be defined as

$$\text{PMI}(w_i, w_j) = \log \frac{P(w_i, w_j) + \epsilon}{P(w_i)P(w_j)}$$

and by focusing on only the top-10 most probable words in a topic ($M = 10$) we get

$$C_{\text{Newman}} = \frac{1}{\binom{10}{2}} \sum_{i=1}^{10} \sum_{j=i+1}^{10} \text{PMI}(w_i, w_j). \tag{1.29}$$

For a topic model we can get the expected coherence using Equation 1.28. What makes Newman's notion of coherence unique is that it is defined over contexts. In their paper they tried different definitions of contexts: WordNet subtrees, Wikipedia articles, Google Search Results, and sliding windows. This last context, when applied to Wikipedia (treated as one long document) with a window size of 10 gave best results.

Currently the state-of-the-art coherence measure has been developed by Röder [42] by ex-

haustively trying alternative formulations for different parts of the calculation. It is summarized in Figure 1.15. The particular combination below, denoted by the authors as C_V, was chosen because it had the highest agreement with human annotators. The following components make up C_V:

Support Rather than requiring average pairwise probabilities to be large, the authors found that a larger support is better. Let W^{10} be the set of the 10 most probable words in a topic. Then the support to be maximized is

$$\sum_{i=1}^{10} \text{Supp}(w_i|W^{10}).$$

Context The authors found that a sliding window with 110 words trained on a Wikipedia corpus works best.

Probability Calculation How should $\text{Supp}(w_i|W^{10})$ be computed? The authors begin by representing each word w_i as a vector in $\mathbf{v}^{(i)} \in \mathbf{R}^d$, such that $v_j^{(i)} = \text{PMI}(w_i, w_j)$, where the PMI is computed over the sliding window contexts. Let $\mathbf{v}^* = \sum_{i=1}^{10} \mathbf{v}^{(i)}$, then the authors define

$$\text{Supp}(w_i|W^{10}) = \frac{\mathbf{v}^{(i)} \cdot \mathbf{v}^*}{||\mathbf{v}^{(i)}||_2||\mathbf{v}^*||_2}. \tag{1.30}$$

One crucial advantage this method has is that word combinations that are semantically related, but unlikely to appear together, e.g. 'car' and 'automobile', are no longer penalized. A measure of coherence based on the cosine similarity of the word2vec embedding has also been proposed [43].

1.3.5 Discriminativeness Evaluation Measures

The intuitive goal in this section is to measure how discriminative topics and words are. The reason is that we are interested in building a model that provides a good and succinct summary of the underlying dataset. By saying that we want a topic model to be succinct we have two desires:

1. The presence (or absence) of a topic in a document is informative of the document's

Input: (1) Topic Model W, T.
(2) For each word w_j, $j \in \{1, 2, \ldots, d\}$, a embedding of w_j as a vector $\mathbf{v}_i$ s.t., semantic similarity between words is proportional to inner product between the embedding vectors.

Measure: $E[C_V] = \sum_{t=1}^{K} C_V(t) \sum_{i=1}^{n} \frac{W_{it}}{n}$.
Where $C_V(t) = \sum_{i=1}^{10} \frac{\mathbf{v}^{(i)} \cdot \mathbf{v}^*}{||\mathbf{v}^{(i)}||_2 ||\mathbf{v}^*||_2}$ is calculated using the top-10 (by probability) words of topic t.
A higher coherence is better.

Fig. 1.15: Coherence

contents, and is a good way to distinguish documents from one-another.

2. The presence (or absence) of a word in a topic is informative about the topics contents, and is a good way to distinguish topics from each other.

To see why the above desires are sensible, it is worth considering their contradictions. The contradiction of (1.) is "the presence or absence of a topic in a document is not informative of the document's contents." This would happen if a particular topic is present in all documents, so that its presence would contain zero information. Similarly, if a word is absent from all topics, its absence is not-informative. On the other hand suppose a particular set of words only occurred in one topic, with high probability for that topic. Then (a) those words would be meaningful identifiers for that topic and (b) if observed in a new document, we would know that the topic is present with high probability.

In the literature, multiple measures have been proposed to measure discriminativeness of topics and words. Typically these measures have been proposed in a context of a certain type of model selection, evaluation, or ranking task. We have adapted these measures for the purposes of model evaluation.

The first set of measures we consider are based on the distance of our topics to known bad outcomes [44]. Let X be the dataset we are modeling, and WT the model we are evaluating. We define the following bad outcomes,

Topic Uniform $\mathbf{t}^u = \left(\frac{1}{d} \quad \frac{1}{d} \quad \ldots \quad \frac{1}{d}\right)$, a topic whose distribution over words is uniform.

Topic Average $\mathbf{t}^a = \frac{1}{k}\sum_{t=1}^{k}(WT)_{t:}$, the mean topic in our model.

Doc Average $\mathbf{t}^d = \frac{1}{n}\sum_{i=1}^{n} X_{i:}$, the mean word distribution in X.

Weights Background $\mathbf{w}^b = \begin{pmatrix}\frac{1}{n} & \frac{1}{n} & \dots & \frac{1}{n}\end{pmatrix}^T$, a weight vector for a topic that has been assigned with uniform probability $1/n$ to every document.

The authors then suggest that the discriminativeness is proportional to its distance from these known bad outcomes. To measure distance they propose using both cosine similarity, as well as the KL-divergence. We focus on the KL divergence, and omit cosine similarity. The KL divergence has a nice interpretation in this setting. For example, let $\mathbf{t}$ represent a topic distribution, and $H(\mathbf{t}) = -\sum_j t_j \log t_j$ its entropy, then

$$\begin{aligned} D_{\mathrm{KL}}(\mathbf{t}||\mathbf{t}^u) &= H(\mathbf{t},\mathbf{t}^u) - H(\mathbf{t}) \\ &= -\sum_{j=1}^{d} P(j|\mathbf{t}) \log P(j|\mathbf{t}^u) - H(\mathbf{t}) \\ &= -\sum_{j=1}^{d} P(j|\mathbf{t}) \log \frac{1}{d} - H(\mathbf{t}) \\ &= \log d - H(\mathbf{t}). \end{aligned}$$

Since $H(\mathbf{t}) \leq \log d$, the KL divergence to the uniform topic can be interpreted as how many nats (bits) of information we save by using topic $\mathbf{t}$ as opposed to the uniform topic. Furthermore, the larger the savings, the more easy it is to communicate topic $\mathbf{t}$, and consequently understand what is meant by it.

Next we transform these distances into an evaluation measure for an entire topic model. To do so we simply weigh each topic's distance to the bad outcomes by its overall weight in the model W, T. Details are in Figure 1.16

In [45] the authors showed that giving discriminative words more weight, they were able to infer topic models. A word is discriminative if its presence provides a lot of information about what topic generated it. Let $P(j|t)$ represent the probability of a word j given a topic t. Then the *discriminability* of word j with respect to topics is

$$\mathrm{disc}_{\mathrm{word}}(j) = 1 - \frac{H(T_{:j})}{\log k}. \tag{1.31}$$

Input: (1) Topic Model W, T.
(2) Corpus X

Measures: Let $w_t = \frac{1}{||W||_1}\sum_{i=1}^{n} W_{it}$ be the weight of topic t in the model. Then for each bad outcome the corresponding measure is defined as,
Topic Uniform $\sum_{t=1}^{k} w_t D_{\mathrm{KL}}(T_{t:}, \mathbf{t}^u)$ where $\mathbf{t}^u = \left(\frac{1}{d} \quad \frac{1}{d} \quad \dots \quad \frac{1}{d}\right)$.
Topic Average $\sum_{t=1}^{k} w_t D_{\mathrm{KL}}(T_{t:}, \mathbf{t}^a)$ where $\mathbf{t}^a = \frac{1}{k}\sum_{t=1}^{k}(WT)_{t:}$.
Doc Average $\sum_{t=1}^{k} w_t D_{\mathrm{KL}}(T_{t:}, \mathbf{t}^d)$ where $\mathbf{t}^d = \frac{1}{n}\sum_{i=1}^{n} X_{i:}$.
Weights Background $\sum_{t=1}^{k} w_t D_{\mathrm{KL}}(W_{:t}, \mathbf{w}^b)$ where $\mathbf{w}^b = \left(\frac{1}{n} \quad \frac{1}{n} \quad \dots \quad \frac{1}{n}\right)^T$.

A higher distance is better for each measure.

Fig. 1.16: Distance to Topic Uniform, Topic Average, Doc Average, and Weights Background

Although the authors don't propose it, the discriminability of a topic t (with respect to documents) would be

$$\mathrm{disc}_{\mathrm{topic}}(t) = 1 - \frac{H(W_{:t})}{\log n}.$$

These quantities are identical in form to 'redundancy' in information theory [46].

For topics the authors propose to define the 'sparsity' of topic t as

$$\mathrm{sparsity}(t) = 1 - \frac{H(T_{t:})}{\log d}.$$

We won't consider this definition of sparsity, since it's closely related to the 'distance to topic uniform' measure.

The intuition behind $\mathrm{disc}_{\mathrm{word}}$ (wlog $\mathrm{disc}_{\mathrm{topic}}$) is as follows. $H(T_{:j}) \leq \log k$ represents how many nats are needed to specify what topic word j came from given that j occurred. As a result dividing by $\log k$ represents what percentage of the maximum number of nats are needed to identify the topic. One minus this percentage, is the percentage of nats saved, compared to the worst case, for identifying the topic t that generated word j. If a word j saves 100% of its nats, then knowing it is present is in a document as good as knowing the topic that generated it is present in a document, therefore j is discriminative with respect to topics.

As with distances to bad outcomes, to turn these into quality measures for a entire model we simply weigh each word and topics discriminability by its probability. This is summarized in Figure 1.17.

Input: (1) Topic Model W, T.

Measures: Let $w_t = \frac{1}{||W||_1} \sum_{i=1}^{n} W_{it}$ be the weight of topic t in the model. Let $v_j = \sum_{t=1}^{k} w_t T_{:j}$ be the weight of a word j.

Discr. Words $\sum_{j=1}^{d} v_j \text{disc}_{\text{word}}(j)$ where $\text{disc}_{\text{word}}(j) = 1 - \frac{H(T_{:j})}{\log k}$.
Discr. Topics $\sum_{t=1}^{k} w_j \text{disc}_{\text{topic}}(j)$ where $\text{disc}_{\text{topic}}(t) = 1 - \frac{H(W_{:t})}{\log n}$.

A higher value is better for each measure.

Fig. 1.17: Discriminability of Words and Topics

The final measure for this category is called closeness [45]. *Closeness* of a topic t is the maximum number of its top words shared with another topic. Concretely, suppose we have a method to select the top m words of a topic t, call this set $V_m(t)$. Then the closeness of t is

$$\text{closeness}(t) = \max_{t' \neq t} |V_m(t) \cap V_m(t')|.$$

The top m words of a topic are typically used to give a brief summary of the topic. As a result if there is no overlap among these top m words, then a topic is distinctive from other topics.

Closeness depends on the method used to select the top m words, V_m. In order for it to be a meaningful topic quality measure, the method used to select the top m words should be good. The straightforward method is to simply pick the top m words for topic t by 'prevalance', i.e. order words j by descending T_{tj}. Another method would be to order words by decreasing discriminability $\text{disc}_{\text{word}}$. It turns out selecting the top m words is a important task in topic model visualization/presentation. So as a preview for the next section, we use the 'weighted uplift' method of [47]. The authors suggest picking the top m words j in a topic t by their uplift,

$$\text{uplift}(j, t) = (1 - \lambda) T_{tj} + \lambda \frac{T_{tj}}{\sum_{t'=1}^{k} T_{t'j}}.$$

Finally, we denote picking the top m words by uplift as

$$V_m(t) = \arg_j \text{sort} \left\{ \text{uplift(t,j)} \right\}_{1...m} . \tag{1.32}$$

An optimal value of $\lambda = 0.6$ was found by evaluating inter-annotator agreement when presenting human users with topics whose top m words were selected with different values of λ. The intuition was that the right λ will make it easier for humans to understand topics, so that the same conclusions will be more consistent (and hence yield higher inter-annotator agreement).

As with other measures in this section, we weigh the per-topic closeness to form a model evaluation closeness; presented in Figure 1.18.

Input:
(1) Topic Model W, T.
(2) $V_m(t)$, a method to pick the top m words in a topic t. We use Equation 1.32.

Measures: Let $w_t = \frac{1}{||W||_1} \sum_{i=1}^{n} W_{it}$ be the weight of topic t in the model.
Closeness is

$$\sum_{t=1} w_t \max_{t' \neq t} |V_m(t) \cap V_m(t')|.$$

A lower value is better.

Fig. 1.18: Closeness

1.3.6 Visual & Exploratory Evaluation

So far Section 1.3 has introduced (for those counting) 17 application-independent quantitative evaluation measures for topic models. In selecting those measures we are trying to predict what dimensions of model evaluation users will care about, and present ways to measure those dimensions. However it's simply not reasonable to expect that we can predict what's relevant for all applications and data sets where topic models are used. As a result subjective, i.e. qualitative, measures of model quality will typically be more powerful and meaningful. Their main downside is that, by definition, they are difficult to quantify, and as a result require human 'expert' time in order to perform evaluation on a case-by-case basis. However, before subjective evaluation can be performed, we must present the topic model.

Topic model presentation has multiple challenges, three that seem important are:

1. Identifying the most important features of T.
2. Presenting W, and T jointly.
3. Enabling the user to interactively adjust how (1) and (2) are performed.

A final level of complexity could be allowing the user to identify features in the presentation that can be used to build a better model. We will briefly discuss this at the end of this section. For now we examine how the above challenges have been address in the literature.

1.3.6.1 Identifying the Most Important Features of T

In the work that popularized topic modeling [16] the only presentation of a topic model is a list of top words for each row of T. For a topic t they selected the top m words j by by magnitude of $T_{tj} = P(j|t)$. Based on the tools we already introduced in Section 1.3.5, we have two other compelling methods for selecting the top m words. Discriminability of a word, Equation 1.31 (p.g. 33). Uplift of a word within a topic, Equation 1.32 (p.g. 36). We examine the effects of these three word orderings by using each of them to present the same topic model. The model is a $k = 10$ model fitted using LDA to the Reuters news dataset. We consider the first three topics (the ordering LDA generates is arbitrary), and display their top $m = 5$ words by each ordering method in Table 1.2. There are some interesting take aways:

- For topics with clear top words, like topic 0, ordering method doesn't make a significant impact.
- The difference between PREVALENCE and DISCRIMINABILITY is typically a few changes of word order.
- However DISCRIMINABILITY is useful for eliminating common words, such as 'meet' in topic 1.
- Using UPLIFT can sometimes make topics unintelligible, e.g. topic 2.

Overall using DISCRIMINABILITY seems like a safe option, however tuning UPLIFT or combining the two together tunable parameter is possible. Although this is most likely a redundant

Table 1.2: Top Words by Three Ordering Methods for a LDA Model of the Reuters Dataset

Topic 0					
PREVALENCE	wheat	grain	corn	crop	depart
DISCRIMINABILITY	wheat	grain	corn	crop	usda
UPLIFT $\lambda = 0.05$	wheat	grain	corn	crop	usda
UPLIFT $\lambda = 0.600$	wheat	grain	corn	crop	usda
Topic 1					
PREVALENCE	franc	meet	brazil	quota	opec
DISCRIMINABILITY	franc	brazil	quota	cocoa	buffer
UPLIFT $\lambda = 0.05$	franc	brazil	cocoa	buffer	saudi
UPLIFT $\lambda = 0.600$	brazil	cocoa	buffer	saudi	bag
Topic 2					
PREVALENCE	loan	asset	cash	term	debt
DISCRIMINABILITY	loan	asset	save	cash	term
UPLIFT $\lambda = 0.05$	loan	asset	save	chemic	caesar
UPLIFT $\lambda = 0.600$	caesar	dome	supermarket	tire	dart
Topic 6					
PREVALENCE	quarter	dividend	earn	4th	gain
DISCRIMINABILITY	quarter	dividend	4th	prior	earn
UPLIFT $\lambda = 0.05$	quarter	dividend	earn	4th	gain
UPLIFT $\lambda = 0.600$	dividend	4th	prior	jan	1st

optimization to spend time on, since it is also the topic model's responsibility to push meaningful words to the top for each topic. I.e. topic modeling could be interpreted as the task of select the top words for each topic from an underlying distribution.

1.3.6.2 Presenting W, T Jointly

Since topic identifier numbers are typically random, in order to meaningful present the W matrix, it needs to be presented alongside T. We will present the W from the same LDA model of Reuters as the T from Table 1.2.

The first two things of interest are topic weights and contingencies. Weights are simply sums of rows of W, and can be interpreted as the expected number of documents a topic appears in out of all the documents in the corpus. The document-based *contingency matrix* that corresponds to a clustering W is

$$C = W^T W.$$

The ijth entry represents in how many documents we expect to see a word from both topic i and topic j. The diagonal entries represent the number of documents we expect to see two words from the ith topic. We present these in Figure 1.19

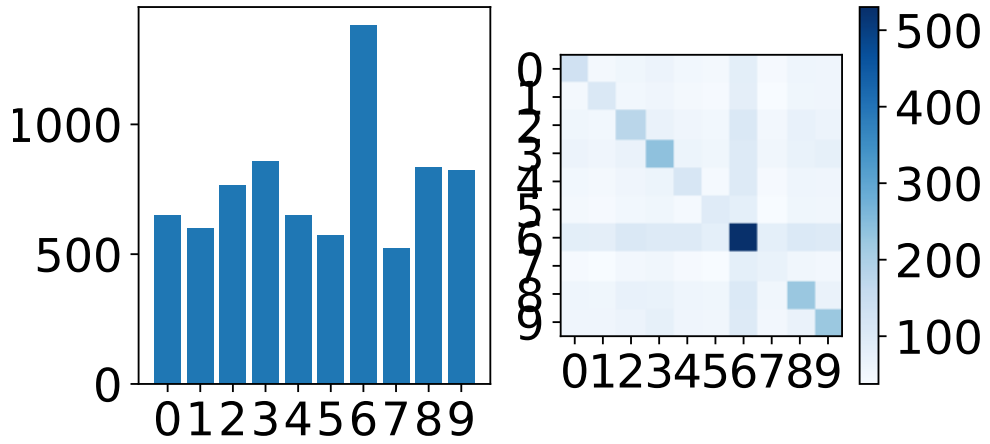


Fig. 1.19: Topic Weights and Document Co-occurrence Contingency

Things of note are that most topics have approximately equal weight, while one topic, topic 6, has about double the average weight. Topic 6 is also correlated with all other topics, whereas the other 9 topics are much more (though not fully) independent. Consulting Table 1.2 we see that Topic 6 seems to be about finance, which apparently is relevant to all the topics within this Reuters dataset.

For small contingency matrices, such as the 10×10 matrix in Figure 1.19, presenting the full matrix is appropriate. However for larger models it's more important to convey local contingency structure in order to not overwhelm the user and expect them to get something out of a 200×200 or 1000×1000 matrix.

To do so, we reduce this problem to embedding (vertex, edge) graphs into two dimensions. The main challenge is that the graph embedding problem takes as input distances between nodes. Meanwhile by computing $C = W^T W$ we get similarities.

A standard way to convert between distances and similarities is to assume there is an underlying probabilistic generative model. This is one of the key ideas for the popular embedding

algorithm t-SNE [48]. To use this idea, first we normalize the rows of C to sum to 1, i.e. set

$$C_{ij} \leftarrow \frac{C_{ij}}{\sum_{ij'} C_{ij'}}.$$

Then $C_{ij} = P(j|i)$ can be interpreted as the conditional probability of observing topic j given that we've observed topic i. Suppose the underlying generative model is spherical Gaussian distribution centered at $W_{:i}$ with standard deviation σ_i, then

$$C_{ij} = \frac{1}{\sqrt{2\pi\sigma_i^2}} \exp\left(\frac{||W_{:i} - W_{:j}||^2}{2\sigma_i^2}\right) \tag{1.33}$$

so that the output $k \times k$ matrix of distances D is

$$D_{ij} = ||W_{:i} - W_{:j}||^2 = 2\sigma_i^2 \log\left(C_{ij}\sqrt{2\pi\sigma_i^2}\right). \tag{1.34}$$

The advantage of this method compared to computing the distances directly is that the σ_is (or other parameters of the kernel, if not using a Gaussian kernel) can be chosen to limit model complexity. For example we may want to limit the size of local neighborhoods so that they are interpretable at a glance. In t-SNE this is based on the entropy of the Gaussian distribution, but other measures of model complexity with respect to visualization are worth exploring.

An alternative to starting from the contingency matrix C, is to compute pairwise distances directly. Since we are working with probability distributions, the Euclidean distance, KL divergence, and earth-mover's distance [49] are reasonable candidates. The cost of computing each of these distances between $W_{:i}$ and $W_{:j}$ is $O(n)$, or $O(d)$ between $T_{i:}$ and $T_{j:}$. We can avoid computing all pairwise distances if necessary. We may want to do so if there are too many pairs, or if we believe that the distance measures are only reliable for local neighborhoods of radius at most r or cardinality at most n. If we have a radius r in mind, we can make use of consequences of the triangle inequality [50] to lower bound which distances will be at least r, and avoid computing them. Then we will have a sparse graph of distances, and can using a method like SSDE [51] to create the 2-d embedding. This method uses breadth-first search to estimate distances between nodes. Then the distance matrix is embedded using PCA. This type of method is also often referred to as locally linear embedding in the

literature [52].

We present an example of Locally Linear Embedding as well as t-SNE in Figure 1.20. The dataset is 20 news groups, and we are visualizing a $k = 20$ LDA model. For the locally linear embedding we used a neighborhood size of 16. Since $k = 20$ we are essentially performing global embedding. For the t-SNE model we use a entropy of 3 bits. Topics are represented using the top 5 words ordered by discriminability. These are the best looking parameters we found using a binary search. However, the visualizations leave a lot to be desired. First there are some topics that look meaningless. These probably should not be displayed. Second, since W isn't meaningful without knowing the corresponding T, we need to represent the topics somehow. But placing labels in the same space as the embedding is challenging with large amounts of topics. Third, t-SNE seems to separate related topics. In general there are a lot of parameters and expertise that go into tuning the embedding algorithms, and evaluating the usefulness of an embedding seems like a task best accomplished by a human reviewer.

1.3.6.3 Enabling the User to Interactively Adjust How (1) and (2) are Performed

One way to facilitate building higher quality topic models is to increase the speed of iteration. Recall that there are different ways to select the top m words for a topic. There are also different ways to arrange topics in space. In order to build the best presentation possible, tools should allow users to interactively adjust presentation parameters. While we are not aware of a perfect tool, in this section we will present some tools that address some of the presentation problems by relying on user interaction.

The first tool we examine is called Termite, and it was specifically designed to help assess topic models [53], Figure 1.21 The authors allow the user to have top words presented by a quantity related to uplift, or by prevalence. In order to present correlations between topics, the authors do not use the contingency matrix that is based on W. As a result they can not present a global sense of which topics co-occur in documents. Rather they present the correlations between topics in terms of the words they contain. To do so all topics are plotted as columns, and words as rows; essentially the authors present T^T. For each topic, the probability of generating a particular word is represented with a proportionally large

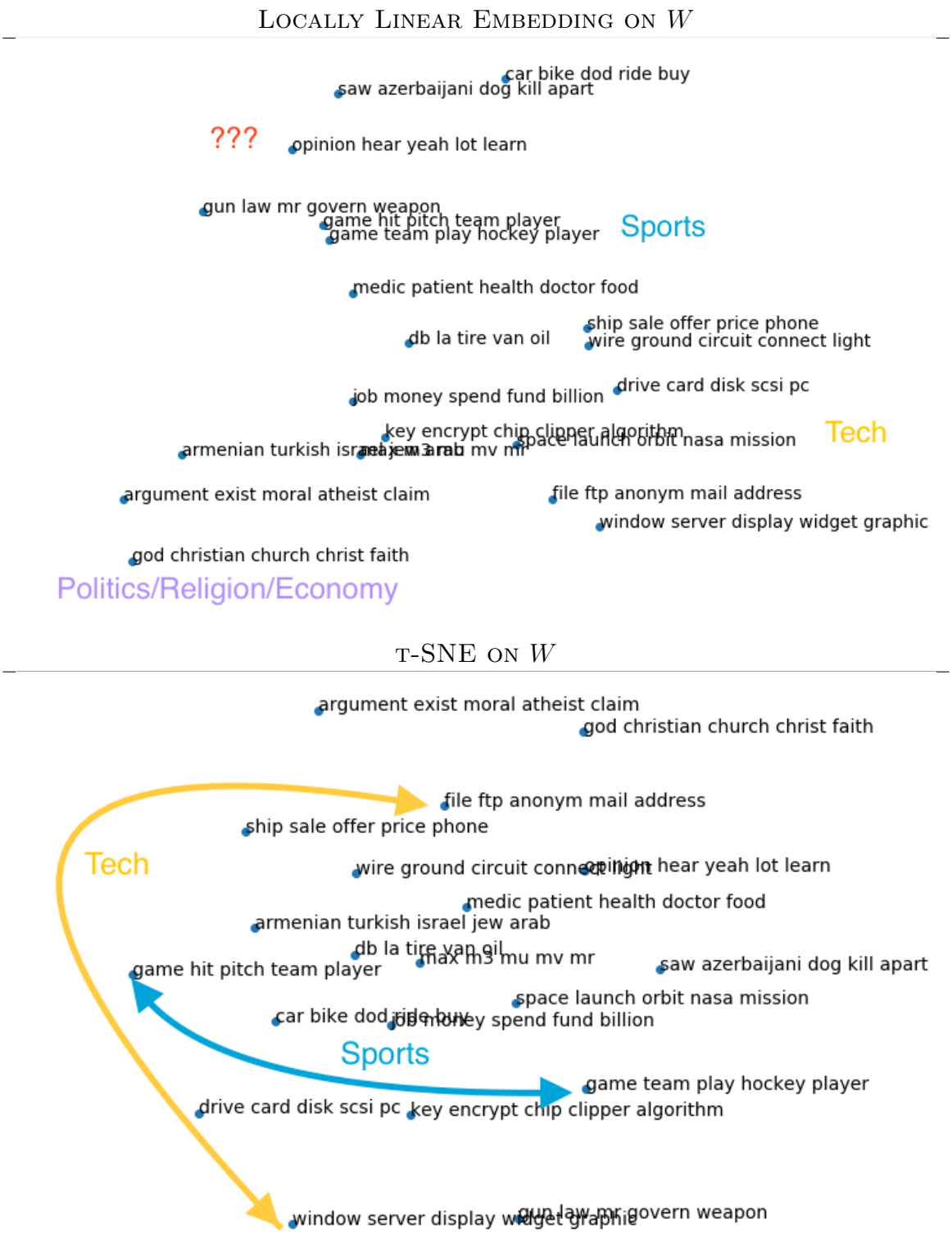


Fig. 1.20: Embeddings of W for a $k = 20$ Model of 20 News Groups

disc. An interesting observation is that area is much more space-efficient for representing proportions than length. In order facilitate comparing topics, the authors allow the user to order words by either overall prevalence or by a cluster-revealing seriation. The goal of this seriation is to re-order the rows (i.e. the words) such that the matrix will appear block diagonal, and clustering among topics will become more apparent. The authors use the Bond Energy Algorithm [54] for this task. We highlight BEA since it is a useful algorithm for manipulating contingency matrices. Finally in order to help the user validate that topics make sense, upon clicking on a topic the user is presented with documents wherein that topic is prevalent. When a lot of parameters are being tuned, sanity checks are useful.

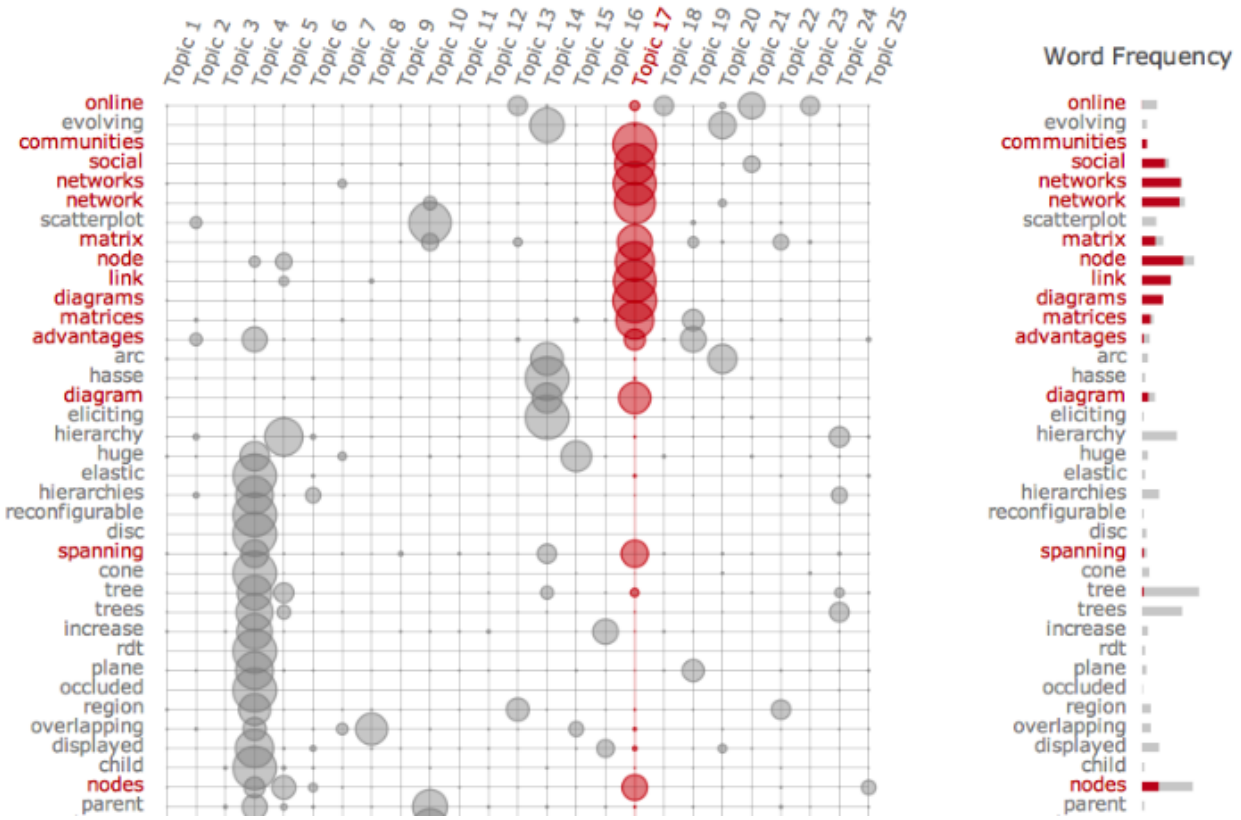


Fig. 1.21: Termite for Topic Model Visualization [53]

The second tool we examine is called LDAvis [47], Figure 1.22. One of the key contributions of LDAvis is highlighting the value of interactive topic model visualization. The top navigation bar provides the first set of interactive features:

1. The method used to calculate the distance between $T_{i:}$ and $T_{j:}$ is toggle-able with a dropbox.

2. The embedding algorithm is also selectable.
3. Number of clusters (addressed later) is also easily adjustable.
4. The number of terms displayed per topic is easily adjustable.
5. The uplift word ordering method we introduced previously comes from LDAvis. Conveniently we see a slider for λ, in case 0.6 doesn't seem to work.

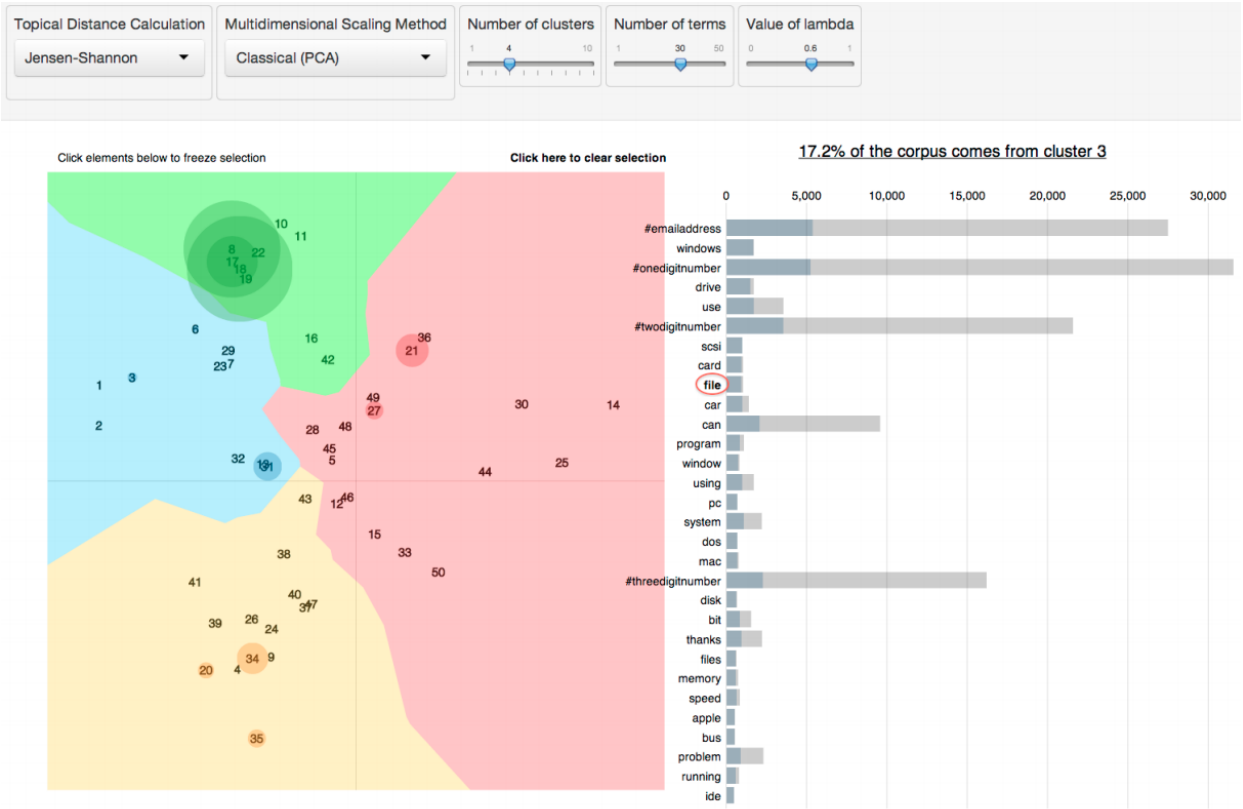


Fig. 1.22: LDAvis for Topic Model Visualization [47]

To embed topics the authors once again look at the T matrix, distances between rows specifically, rather than the contingency matrix W. As a result co-occurrences between topics in documents are not presented. However, the overall prevalence of a topic within the corpus is presented as the area of the disc used to represent it. To avoid the issue of having text and topics in the same space, a topic's word distribution is displayed only when it is selected. In order to allow the user to validate the embedding, by clicking on a word, the topics in which that word has high probability are highlighted. The intuition is that for most words topics that contain those words should be embedded closer together. An interesting

final touch is the idea to perform clustering in the embedded space to automatically identify groups of topics. Since k-means is used, it seems like a good extension would be to allow the user to place the initial cluster centroids, or even fully specify the clustering by drawing partitions.

An unfortunate omission of LDAvis is an ability to perform sanity checks by finding documents related to the presented topics. Another tool called Serendip [55] was designed for the exploration of text corpora based on topic models. They have rich tools for selecting documents related to particular topics, as well as finding passages within those documents that have high prevalence for a particular topic. Unfortunately their work has focused on models of small corpora (mainly classic literature), so as a result some of the useful design decisions made in LDAvis and Termite for presenting models of large corpora have been omitted. Interactivity for the purposes of adjusting the evaluation has also been omitted.

Conclusion & Take-aways Based on our study of topic model visualization and presentation we believe there are three important tennets of useful visualization tools.

1. Concise representation of W and T. A user wants to see what the topics are, and how they tend to co-occur.

2. Interactive visualization settings and rapid iteration. We discussed a multitude of algorithms and parameters that go into creating a visualization. Best outcomes happen when users can easily experiment with parameters and observe their interactions. One caveat is that when designing a GUI that allows users to adjust a multitude of parameters, it is helpful to have a bijective map between the GUI and textual commands. That way it's easy to maintain a log of a user's interaction history, so that they can undo, recall, save, or programmatically generate settings.

3. A connection to the ground truth. Since in this section we are asking humans to evaluate an unsupervised learning algorithm, why not provide them with the underlying data, and request some supervision? I.e. it certainly is useful for a human to check whether it makes sense to embed topics 92 and 113 nearby by viewing two documents where those topics are prevalent. But while that human is viewing the documents, maybe we can provide a convenient UI to label the document or make annotations.

1.3.7 Guided Subjective Evaluation

A final class of evaluation measures might be called 'guided subjective.' The idea is to compute the evaluation of a topic model by having humans perform certain tasks related to the model, and then evaluate human performance. An example of such a task is the 'word intrusion' task [56]. For a particular topic, its top m words are selected, and then one of the words is replaced with a random word from the dictionary. The idea is that if a topic has a clear meaning to humans, it will be easy to *consistently* (among human evaluators) identify the intruder. On the other hand if a topic has non-related words together, its meaning is hidden from humans, so we don't expect humans to be able to consistently pick the correct intruder. The authors give the following topics as examples:

I	dog, cat, horse, apple, pig, cow
II	car, teacher, platypus, agile, blue, Zaire

Are you able to identify the intruder word in I and II?

In general the 'word intruder' task may not be the most meaningful for a particular application, but the meta-algorithm is useful as a powerful way to involve end-users in guided form of subjective evaluation.

1.4 Leveraging Evaluation Techniques for Hyper-parameter Optimization

Hyper parameter optimization algorithms such as HYPERBAND [57], and the references there-in, aim to optimize hyper-parameters of an inference algorithm. Examples of hyper-parameters that we consider are

1. term-weighting, i.e. TF or TF-IDF,
2. the number of topics k,
3. the document-topic regularization prior α and topic-word regularization prior β for LDA,
4. the ℓ_2 regularization on W and T in NMF.

The reason we call these hyper-parameters is because other model parameters depend on them, so that they are parameters for the parameters. Since model parameters, i.e. W, T, are typically recursive functions of the data X as well as the hyper-parameters, it is typically difficult to differentiate model objectives with respect to hyper-parameters. This isn't always impossible, but is typically involved and requires a specialized analysis for each hyperparemeter. See e.g. [18] for an attempt to estimate α, β using fixed-point iterations. Also sometimes hyper-parameters have semantic meaning, so that it's not always appropriate to optimize model objective with respect to them. E.g. if one of the parameters is related to regularization, in order to minimize in-sample reconstruction error we would always set it to 0. As a result we have to be careful with respect to what objective we optimize hyper-parameters.

Fortunately for us, Section 1.3 provides a variety of evaluation criteria. Since the hyper-parameters, and evaluation functions we use are heterogeneous, we need a flexible framework to perform optimization. The only commonality we take advantage of is that the topic model inference algorithms we study are iterative.

The Hyperband framework is based around bandit algorithms. In a prototypical bandit algorithm, one is given 100 coins, presented with 3 variable reward slot machines, and asked to maximize profit after 100 coins. The fundamental trade-off is between exploration: identifying the payouts of each of the slot machines, versus exploitation: playing only the slot machine with highest expected payout. Doing so with a fixed budget of only 100 coins is the challenge.

In the hyperparameter optimization setting, a setting for each of the hyper-parameters, e.g. (TFIDF, $k = 20$, $\alpha = 0.05$, $\beta = 0.01$), corresponds to a slot machine. The coins we have correspond to a limit on the number of iterations we are allowed to perform (or equivalent limitations on time/cost). Payouts are defined in terms of quantitative or qualitative evaluations of the models produced.

Finally, in Hyperband a clever strategy is employed to control the trade-off between exploitation and exploration. Suppose we want the best 3 LDA models after 27 iterations. First we will sample 27 hyper-parameter settings randomly. Then we run each of them for 3 iterations. We will select the top 1/3rd of those (according to our evaluation measure), and run them until 9 iterations. Then we again pick the top 1/3rd and run them until 27 iterations.

As a result we have $27 \times 1/3 \times 1/3 = 3$ best models after 27 iterations. During meta-iteration one we performed $27 \times 3 = 81$ iterations. During meta-iteration two we performed $9 \times 6 = 54$ iterations, and during meta-iteration three we performed $18 \times 3 = 54$ iterations; future meta-iterations would also require 54 iterations. A **important caveat** for HYPERBAND is that if hyper-parameters affect a model's convergence rate, then this method to control exploitation/exploration may not satisfy the nice theoretical properties the authors proved. As a result we do not treat the topic model as a hyper-parameter, and instead run HYPERBAND separately for each topic model. We present HYPERBAND as Algorithm 5.

Algorithm 5 Hyper-parameter Optimization for Topic Modeling using HYPERBAND

Input: Culling ratio η (we use 3), distribution for hyper-parameters $\mathcal{D}$, topic model M, evaluation function $\mathcal{E}$, number of models desired b (a multiple of η for simplicity), after how many iterations τ (also a multiple of η, we use 81).

Output: A set of b models and hyper-parameter settings that are good after τ iterations.

$B_0 \leftarrow \{M(\theta_i)\}_{i=1}^{\frac{\tau b}{\eta}}$ where $\theta_i \sim \mathcal{D}(i)$ is a random setting of hyper-parameters seeded by i.
Run each $M \in B_0$ for η iterations.
for i in $\{1, 2, \ldots, \log_\eta \tau - 1\}$: **do**
    Evaluate each model in B_{i-1} using $\mathcal{E}$.
    $B_i \leftarrow$ top $\frac{1}{\eta}$ models of B_{i-1} by evaluation score.
    Run each $M \in B_i$ until η^{i+1} iterations.
end for
    return last B.

HYPERBAND has several convenient properties. By saving models and using a consistent seeding system for sampling from $\mathcal{D}$ we can re-use past results if we decide to increase τ or b. If we change the evaluation measure $\mathcal{E}$ we will have to re-evaluate previously computed models, but we may get lucky and some of them may have been already computed. For selecting the distributions to form $\mathcal{D}$ we found that it is convenient to be able to parametrize modes and standard deviations separately. For small discrete sets we simply use a uniform categorical distribution. For positive real numbers such as α, β we used the log-normal with medians set to the rule-of-thumb values. For real numbers such as the regularization on W and T we used a 0-mean normal. Standard deviations should be picked in proportion to the number of initial settings considered, τb.

When using power subjective evaluation criteria, HYPERBAND must query humans to evaluate models. During the early stages this can be very expensive since there are a lot of models

to evaluate. A way to control this expense is to use the techniques from active learning [58]. For example we may fit a logistic regression to predict subjective quality using the quantitative evaluations as features, and only ask the human annotator if we have low confidence in our prediction.

In general the evaluation function can be an arbitrary function of quantitative measures and qualitative ratings. To compare heterogeneous features it is useful to first convert them into unitless measures, e.g. using z-scores. Then a multi-criterion hyper-parameter optimization can be achieved by taking a weighted combination of the measures we want to investigate.

We summarize with a recommendation for a iterative workflow for the development of topic models based on the Agile methodology for software development, Figure 1.23.

Requirements:

(1) U: a set of users (ideally domain experts) seriously interested in using topic modeling for domain X.
(2) D: a set of topic modeling developers. (Note that U, D do not have to be disjoint)

Method:

(1) D explains to U what a topic model is, and what the best-case outcome is.
(2) D surveys U what they would like to get out of topic modeling for X.
(3) D identifies existing measures that seem relevant based on steps (2,5), and builds b topic models to maximize those measures.
(4) D presents U with b topic models using visualization or other presentation techniques.
(5) D and U identify which of the b models is best, and identify what the problems areas are.
    - Problems that are easy to quantify are added as measures.
    - Problems that are difficult to quantify are formulated as human evaluation tasks. These tasks become part of model selection in step (3), i.e. part of $\mathcal{E}$ in HYPERBAND.
(6) Repeat from step (2) until the model is good enough.

Fig. 1.23: Agile Topic Modeling

1.5 Examples of Empirical Evaluation

We conclude this Chapter by presenting insights based on comprehensive experimental evaluation of NMF, LDA, and K-means on three datasets. The datasets are all well known, and freely available; we describe them in Table 1.3.

For pre-processing we first dropped non-alphanumerics, and mapped each term to its closest WordNet morph using the NLTK package [59]. Then we dropped terms part of a standard 'stop list' as well as terms that occurred in fewer than 5 documents, or occurred in more than 10% of all documents.

Table 1.3: Dataset Summary

Name	Description	n	d	True k	Density
NIPS [60]	Contents of NIPS papers.	1500	12419	?	10.69%
20 News Groups [61]	Documents from a bulletin board about 20 different topics.	18846	7097	20	1.01%
Reuters-21578 [60]	Documents from the Reuters newswire.	10788	3169	90	1.71%

For each dataset, and for each topic modeling algorithm we used Hyperband to find the best hyper-parameters for that dataset, algorithm pair. We specified the hyperparameter sampling distribution as follows:

1. uniform over TF and TFIDF for pre-processing,
2. uniform over k in $\{5, 10, \ldots, 200\}$,
3. log-normal with median $1/k$ for α and β for LDA,
4. normal with mean 0 and standard deviation 0.5 for ℓ_2 regularization for W and T for NMF.

We asked Hyperband for the 9 best models after 81 iterations per model, and picked the best model out of those for each evaluation measure under consideration.

In Table 1.4 we present and discuss the rankings of topic models for each measure. For each measure we report the best performing topic model, and the 2nd best performing topic model. We also report the mean (over the three datasets) percentage improvement in using

Table 1.4: Per Measure, Best Topic Model and Percent Improvement Over 2nd Best

Measure (Figure)		Best	2nd	Mean Δ
VI IS	(A.1)	LDA	Kmeans	3.97
VI oos	(A.2)	LDA	Kmeans	0.23
ARI IS	(A.3)	LDA	NMF	204.71
ARI oos	(A.4)	LDA	NMF	166.29
coherence	(A.5)	Kmeans	NMF	26.87
frob fit oos	(A.6)	NMF	LDA	26.90
loglike oos	(A.7)	LDA	Kmeans	9.61
perplexity oos	(A.8)	Kmeans	LDA	0.01
frob fit is	(A.9)	NMF	Kmeans	35.21
loglike is	(A.10)	LDA	Kmeans	7.52
topic uniform	(A.11)	NMF	LDA	100.60
topic average	(A.12)	NMF	LDA	63.61
weights background	(A.13)	Kmeans	NMF	159.37
doc average	(A.14)	NMF	LDA	117.20
discriminability topics	(A.15)	Kmeans	NMF	159.37
discriminability words	(A.16)	NMF	Kmeans	27.82
closeness	(A.17)	LDA	Kmeans	100

Table 1.5: Relative Topic Model Performance

Algorithm	# Times Best	Mean Δ When Best
NMF	6	61.89
LDA	7	70.32
Kmeans	4	86.40

the best model compared to the 2nd best. We refer to this improvement as Δ, defined for evaluation measures where high scores are better as

$$\Delta = \frac{\text{score(best)} - \text{score(2nd)}}{|\text{score(2nd)}|}. \tag{1.35}$$

For evaluation measures where low scores are better, we report $-\Delta$. Detailed plots are available in Appendix A.1 (p.g. 113).

These results can be further summarized as the number of times each algorithm is best, and its lead when best, in Table 1.5.

The results worth highlighting are the following,

ARI, VI It is surprising that LDA performs so much better on ARI than it does VI, since

both of these measure correspondance to human labels. One possible explanation is that VI handles 'soft-clusterings' much better than ARI. As a result VI is a more stable measure, whereas ARI can fluctuate since noise can have significant outcome on the mapping [0.34,0.33,0.33]→[1,0,0].

Frobenius Reconstruction Error, Loglikelihood Unsurprisingly NMF and LDA perform best, respectively. It is interesting that K-means often comes second; perhaps the fact that its model is simpler allows it to generalize better for diverse measures.

Coherence We use the UMass version of Coherence [36], where-in we measure to what extent words that co-occur in topics co-occur in the input documents. It's interesting but not surprising that k-means does well, since the topics it generates are means of input documents.

Perplexity Despite this measure being the original motivation for LDA, it is interesting that K-means and LDA are essentially tied in our experiments. We believe that this may be due to the simple way we estimate the topic-distribution of out-of-sample documents. Other methods for estimating the these distributions [62] could change this result.

The Remaining Measures aren't interesting on their own. Rather they should be used as terms in a linear combination whose sum is the evaluation measure that HYPERBAND optimizes. For example, an issue with NMF and Kmeans models are large global background topics of common words shared by all documents. Adding the `topic_uniform` evaluation measure would penalize the hyper-parameter combinations that lead to such global background topics.

In Table 1.6 we present the best hyper-parameters (of those we examined) for each topic model for each measure for the 20NG dataset. It is interesting that for the out-of-sample based measures the best values of K tend to be close to the human-labeled K of 20. Such a result isn't obvious because the topics inferred by an algorithm may not necessarily coincide with what humans identify as classes. Indeed K-means and NMF both have an instance where they use K much higher than necessary. For the optimization-based objectives, it's unsurprising that K as large as possible is best. For coherence, both Kmeans and LDA favor a small number of topics. Whereas NMF actually uses a high number of topics,

Table 1.6: Best Params for 20NG Dataset

Measure	LDA			NMF			Kmeans
	K	α	β	K	ℓ_2^T	ℓ_2^W	K
VI IS	5	0.20	0.20	100	0.02	-0.02	15
VI oos	5	0.20	0.20	35	0.01	-0.02	15
ARI IS	30	0.03	0.03	45	-0.12	-0.03	120
ARI oos	30	0.03	0.03	45	-0.12	-0.03	120
coherence	15	0.07	0.07	145	-0.05	-0.06	20
frob fit oos	195	0.01	0.01	190	-0.07	0.02	190
loglike oos	195	0.01	0.01	160	-0.01	0.09	190
perplexity oos	195	0.01	0.01	5	0.06	0.03	5
frob fit is	195	0.01	0.01	190	-0.07	0.02	190
loglike is	195	0.01	0.01	160	-0.01	0.09	190
topic uniform	195	0.01	0.01	155	-0.04	0.15	155
topic average	190	0.01	0.01	195	-0.02	0.01	155
weights background	185	0.01	0.01	100	0.05	-0	155
doc average	190	0.01	0.01	195	-0.02	0.01	155
discriminability topics	185	0.01	0.01	100	0.05	-0	155
discriminability words	190	0.01	0.01	195	-0.02	0.01	15
closeness	5	0.20	0.20	5	0.01	-0.03	15

along with negative regularization (which encourages sparsity). This is interesting because although K-means achieves the highest coherence on this dataset, NMF does better than LDA. This highlights the important point that selecting a set of hyper-parameters that work well for one topic model isn't necessarily even close to the hyper-parameters that will work well for another model. Doubly so if there are interactions among the parameters, such as regularization and K. The most striking example of this may be perplexity. LDA with K=195 essentially has the same perplexity, as the best algorithm Kmeans with K=5 (see Figure A.8 (p.g. 117)). This is particularly impressive on the part of LDA since this hyper-parameter setting is strong for 11 of the 17 measures we consider.

To conclude, we look at the median improvement by going from the 2nd best to best algorithm for each measure in Figure 1.24. The median improvement Δ is nearly 28%. As a result if one is interested in a combination that includes many of the above measures, it is necessary to try all topic models. Even if only one of the quantitative measures is of interest it's worth trying multiple models. Our sample of three data sets isn't large enough to make conclusive judgements off of. What we can show is that with respect to certain measures, some conceptually simple algorithms, such as K-means can be competitive. This means that

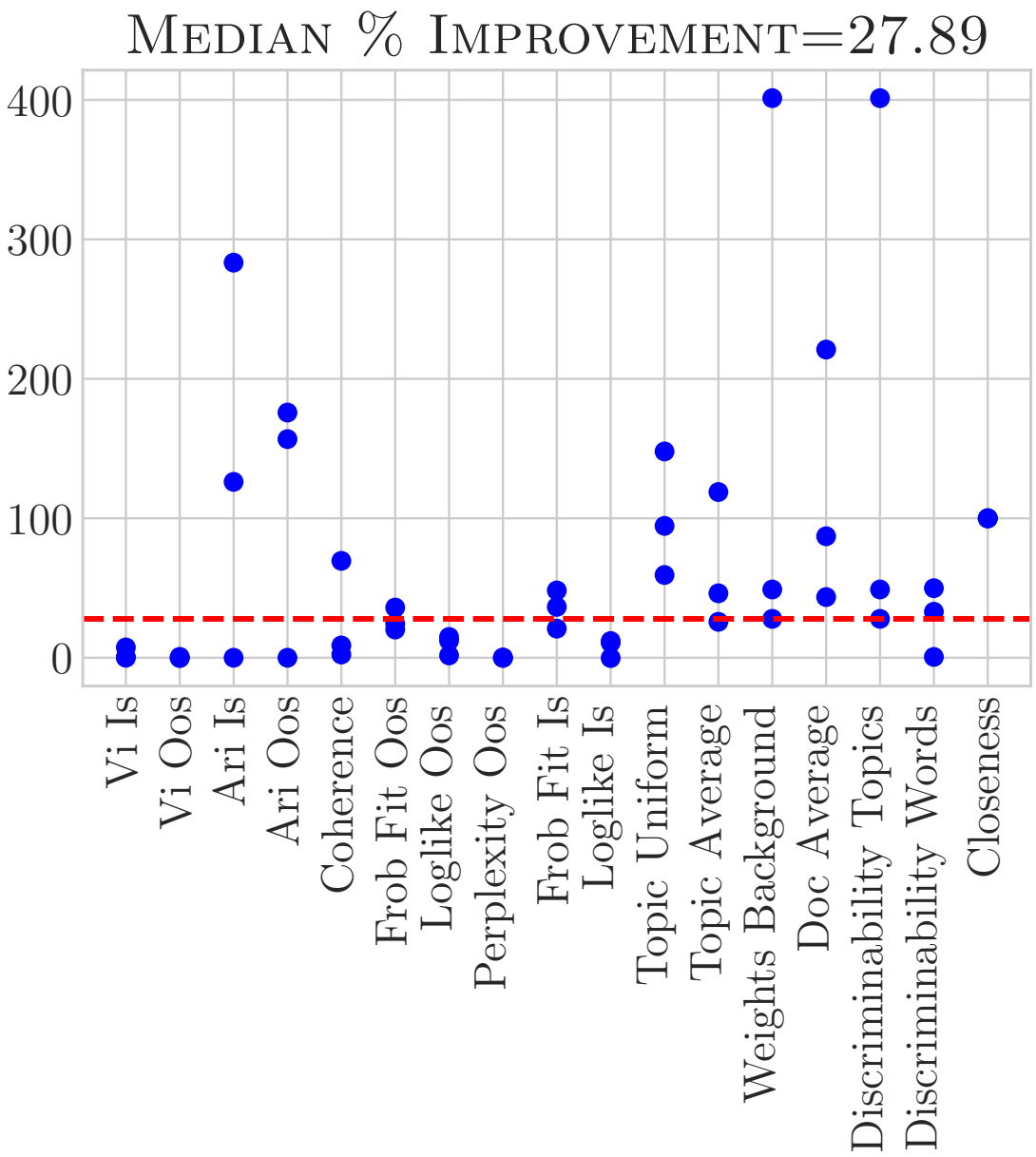


Fig. 1.24: Median Improvement from 2nd to Best

it isn't always necessary to start with a complex model to achieve good results, but complex models shouldn't be dismissed if the overall payoff of the task may be worth the upfront cost. Finally, the three step methodology of (1) identifying useful evaluation measures (2) optimizing each model's hyper-parameters for each of the measures or a combination of them (3) comparing performance across measures, including subjective evaluation, should be the main take-away of this chapter.

CHAPTER 2

Algorithms for Distributed Non-negative Matrix Factorization

In Chapter 1 we argued and showed why the non-negative matrix factorization (NMF) is a good way to perform topic modeling. In addition to topic modeling, NMF is a useful primitive for many other tasks in machine learning. Recall that the objective of NMF is to factorize an input matrix X as a product of non-negative matrices W, T such that $X \approx WT$. By adapting the meaning of what rows, and columns of X correspond to, and by adding problem-specific constraints or regularizations we can adapt NMF to various applications such as:

- Recommender systems, where rows represent users and columns represent items. NMF is used to predict what scores a particular user will give to an item they haven't yet tried. [65]
- Hyper-spectral imaging, where each row represents an individual pixel and each column represents the abundance of a particular signal in that pixel. NMF is used to identify the signals ('endmembers') and their weights ('abundances') in each pixel of an image. This is useful e.g. for predicting crop yields from aerial imagery. [66]
- Sparse coding and dictionary learning for fMRI Images and other domains. [67] By holding T fixed while fitting W we are performing sparse coding, and by holding W fixed while fitting T we are performing dictionary learning. Regularization in the form of ℓ_1 or ℓ_2 penalties on the norms of W, and T can be used to control sparsity.
- Outlier analysis for most of the above applications. [68]

As a result algorithms that can compute the NMF in a variety of settings are of interest.

Portions of this chapter have been submitted to M. Tsikhanovich, M. Magdon-Ismail, and V. Zikas, "PD-NMF: Private distributed non-negative matrix factorization," Submitted to ICML, 2018 and M. Tsikhanovich, M. Ishaq, M. Magdon-Ismail, *et al.*, "PD-SVD: Private distribute singular value decomposition," Submitted to UAI, 2018.

In particular we focus on algorithms for when the rows of an input X matrix have been distributed among M parties. We call this problem the 'row-distributed NMF', and the formal problem statement is in Figure 2.1. Note that the M parties have to agree on which set of d features to use. This is a non-trivial task and needs to be addressed for each application, however it is outside the scope of this work.

Working in the row-distributed setting is interesting for several applications:

- Sensor networks where-in the data is collected at separate locations. [69]
- Multi-datacenter applications of NMF. E.g. for calculating trending news topics for a social media platform where different customers are served from separate datacenters, or for calculating product recommendations. [70]
- Parallelizing and Batching NMF. E.g. when the data is on one server and we wish to parallelize NMF it can be split row-wise to M servers. Or if the data is too large to fit into memory it can be operated on at a per-batch basis. However this isn't the best use case for row-distributed NMF since more flexible approaches where the input X is distributed by blocks of arbitrary size rather than rows tend to work better. [71], [72]
- When data is owned by separate organization or cross-border expatriation laws such as the GPDR [73] govern its transfer, a privacy-preserving computation of the NMF (and hence privacy-preserving variant of the application for which NMF is being used) is needed. We explore this in detail in Chapter 3.

Within the row-distributed setting we study an 'offline' setting and an 'online' setting. In the **offline** setting, each of m parties has already computed a local NMF $X^{(m)} \approx W^{(m)}T^{(m)}$. The goal is to compute the global T (Equation 2.1) without requiring each party to access $X^{(m)}$. This setting is interesting because each party m can simply publish functions of its $W^{(m)}T^{(m)}$ and then go offline; alternatively each party could forget $X^{(m)}$ to save storage or destroy it to minimize exposure to a data breach risk. In the **online** setting the focus is on re-creating the centralized solution[1] exactly. Since we are interested in algorithms whose communication cost is sub-linear in the number of documents, n, we can not simply have each party publish their $X^{(m)}$. We also can not have each party publish their $W^{(m)}$ since

[1]By *centralized solution* we refer to the set of topics T that would be obtained by first aggregating all $X^{(m)}$ on a central server, vertically stacking them to form X and then running RRI-NMF, Algorithm 2 (p.g. 12), on X to obtain T.

M parties each have a non-negative matrix $X^{(m)}$ of n_m rows over d features, for $m = 1, \ldots, M$. Given a rank k, each party must compute the same non-negative row-basis matrix $T \in \mathbf{R}^{k \times d}$, $T \geq 0$, such that:

- The sum of Frobenius reconstruction errors for each matrix $X^{(m)}$ onto the basis T plus an optional regularization term is minimized:

$$\min_{T} \ \frac{1}{2} \sum_{m=1}^{M} ||X^{(m)} - W^{(m)}T||_F^2 + \text{reg}(W, T), \tag{2.1}$$

  where $W^{(m)} \in \mathbf{R}^{n_m \times k}$ is an optimal non-negative least-squares fit of $X^{(m)}$ to T.

- Communication is small (sub-linear in $\sum_m n_m$).

This can be thought of as the NMF problem where the $X^{(m)}$s have been vertically stacked to form X, and each finds the $W^{(m)}$ that corresponds to their rows, meanwhile all parties find the same T:

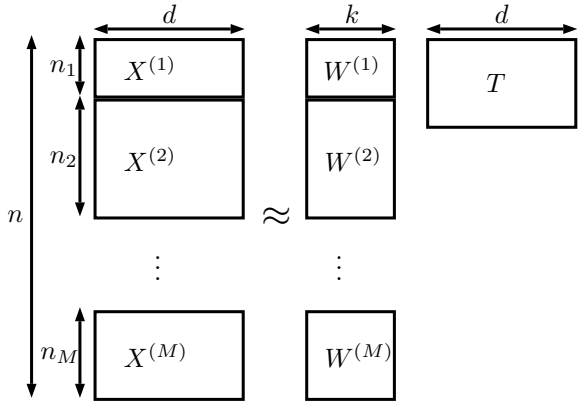


Fig. 2.1: Row-Distributed NMF

it has n rows. As a result, in the online setting each party will have to participate in a distributed iterative computation, where it must compute a function of its $X^{(m)}$ during each iteration.

Terminology & Notation In this chapter we will call rows of X *documents*, rows of T *topics* and rows of W *document-topic weights*, or sometimes omitting the qualifier, *weights*. This terminology is borrowed from the topic modeling application of NMF, but it is a useful mental model for all applications of NMF. For example in recommender systems, 'topics' refer to items frequently purchased/consumed together, whereas a 'document' refers to a single person's purchase/consumption history. Similar intuitive analogues are not hard to construct for the other applications of NMF.

When a norm isn't specified we refer to the matrix 2-norm $||A|| = ||A||_2$, and vector 2-norm $||\mathbf{v}|| = ||\mathbf{v}||_2$ respectively. Let $\sigma_1(A)$ be the top singular value of A, and $\lambda_1(A)$ be the top eigenvalue (by magnitude) of A, then $||A||_2 = \sigma_1(A) = \sqrt{\lambda_1(A^T A)}$. We will also sometimes use the matrix and vector 1-norm, which we denote as $||A||_1 = ||\text{vec}(A)||_1 = \sum_{ij} |A_{ij}|$. We will frequently use the squared Frobineus norm of a matrix, which is defined as

$$||A||_F^2 = ||\text{vec}(A)||_2^2 = \sum_{ij} A_{ij}^2.$$

We refer to the $k \times k$ identity matrix as I_k, and the d-dimensional column vectors of a constant c as $\mathbf{c}_d$. We refer to the jth column of the identity matrix as $\mathbf{e}_j$; note that this is a vector of 0s except for a 1 in the jth position.

We use the superscript $A^{(m)}$ to mean that an object A is associated with party m. Meanwhile $T_{t:}$ refers to the tth row of T and $W_{:t}$ refers to the tth column of W.

2.1 WR-NMF: Algorithm for Offline Row-Distributed NMF

The key idea for our offline algorithm is that topics can be used as representatives for documents. NMF allows us to reconstruct an appropriately weighted topic matrix to obtain a good reconstruction for the original documents. This is why it's called Weighted Representative NMF, or WR-NMF.

In the offline setting each party m is willing to publish $T^{(m)}$, as well as a summary of $W^{(m)}$.[2] We know that for each $X^{(m)}$, $T^{(m)}$ is an approximately optimal set of topics. As a result we can not expect the global T that must minimize the reconstruction error of Equation 2.1 (p.g. 57) for all $X^{(m)}$, to be as good as each $T^{(m)}$ for each $X^{(m)}$. As a result we wish to minimize the additional error that T will introduce in reconstructing $X^{(m)}$.

To avoid needing access to the $X^{(m)}$s, we upper bound the additional error introduced by T in terms of the $T^{(m)}$s and a summary of the $W^{(m)}$s. Concretely, for the offline setting we suppose that party m publishes k_m topics $T^{(m)}$. Then $\mathbf{v}^{(m)} \in \mathbf{R}^{k_m}$ is a k_m-dimensional vector where $v_t^{(m)}$ is the weight for party m's tth topic. We want to bound Equation 2.1 (p.g. 57)

[2]Since $W^{(m)}$ is a $n_m \times d$ matrix, in order to achieve sub-linear communication cost we cannot afford to publish the entire $W^{(m)}$.

in terms of a NMF-objective dependent on the topics $\{T^{(m)}\}_{m=1}^{M}$

$$\sum_{m=1}^{M} \mathcal{E}(X^{(m)}, T) \leq \sum_{m=1}^{M} \mathcal{E}(X^{(m)}, T^{(m)}) + \frac{1}{2}||(I_{k_m}\mathbf{v}^{(m)})(\overbrace{T^{(m)} - \hat{W}^{(m)}T}^{\Delta^{(m)}})||_F^2 \tag{2.2}$$

where

$$\mathcal{E}(X^{(m)}, T) = \frac{1}{2}||X^{(m)} - W^{(m)}T||_F^2 + \text{reg}(W, T). \tag{2.3}$$

The reason $W^{(m)}$ is omitted from $\mathcal{E}(X^{(m)}, T)$ is because each party can locally minimize $\mathcal{E}(X^{(m)}, T)$ with respect to $W^{(m)}$. This is a non-negative least-squares task, which can be solved efficiently (e.g. with quadratic programming, or NMF while holding T fixed) as long as $\text{reg}(W, T)$ is reasonable, e.g. convex.

The bound as formulated in Equation 2.2 is useful for three reasons:

1. We have formulated the additional error introduced by using T to model each $X^{(m)}$ in terms of the error used to model each $T^{(m)}$.
2. This error depends only the $T^{(m)}$s and a single weight for each $T^{(m)}$, which we will show can be computed as a summary of $W^{(m)}$.
3. The reconstruction error $\Delta^{(m)}$ for documents $T^{(m)}$ using topics T can be minimized using standard NMF.

To achieve the third goal, we set

$$\hat{X}^{(m)} \leftarrow (I_{k_m}\mathbf{v}^{(m)})T^{(m)}$$
$$Y^{(m)} = \hat{W}^{(m)}(I_{k_m}\mathbf{v}^{(m)})$$

and use standard NMF to find a minimizer T for

$$\min_{Y,T\geq 0} \frac{1}{2}||\hat{X}^{(m)} - Y^{(m)}T||_F^2. \tag{2.4}$$

We can discard $Y^{(m)}$ or solve for $\hat{W}^{(m)} = Y^{(m)}(I_{k_m}\frac{1}{\mathbf{v}^{(m)}})$ if it's of interest outside of the row-distributed NMF task.

Now that we have motivated why the bound in Equation 2.2 is useful, we find an appropriate $\mathbf{v}$ to satisfy it. We can do so by assuming that each party has found $W^{(m)}, T^{(m)}$ that locally minimize $\mathcal{E}(X^{(m)}, T^{(m)})$, and that $\text{reg}(W, T)$ is well behaved.

Theorem 1. *Suppose that*

1. *Each of m local optimizations has converged in terms of $W^{(m)}$ and $T^{(m)}$.*
2. *reg(W, T) is at most a quadratic function of T, and the Hessian of reg(W, T), $H_{reg}^{(m)} = \frac{\partial^2 reg(W,T)}{\partial T^2}\Big|_{W^{(m)},T^{(m)}}$ is positive semi-definite.*

Then, let $S^{(m)} = \frac{H_{reg}^{(m)} + (H_{reg}^{(m)})^T}{2}$ and set

$$v_j^{(m)} = \sqrt{||W_{:j}^{(m)}||_2^2 + ||S^{(m)}||}$$

to ensure that the upper bound in Equation 2.2 holds.

Proof. The LHS and RHS of the upper bound each have M terms, one for each party m. We will show that the mth term on the LHS is less than the mth term on the RHS. As a result we drop the superscript, i.e. $T^{(m)} = T$ within this proof.

In order to reason about the effect of reconstruction error of individual topics, it is helpful to once again cast the NMF objective $\mathcal{E}$ in terms of the sum of rank-one residuals. Recall from Equation 1.2 (p.g. 8) that the residual that the tth topic is responsible for is defined as

$$R_t = X - \sum_{t'=1, t' \neq t}^{d} W_{:t'} T_{t':} = X - WT + W_{:t} T_{t:}.$$

Now we can rewrite our objective $\mathcal{E}$ in terms of R_t since

$$\begin{aligned}
\mathcal{E}(X, T) &= \frac{1}{2}||X - WT||_F^2 + \text{reg}(W, T) \\
&= \frac{1}{k} \sum_{t=1}^{k} \frac{1}{2}||X - WT||_F^2 + \text{reg}(W, T) \\
&= \frac{1}{k} \sum_{t=1}^{k} \overbrace{\frac{1}{2}||R_t - W_{:t} T_{t:}||_F^2 + \text{reg}(W, T)}^{d_t(T_{t:})}
\end{aligned}$$

We drop the $\frac{1}{k}$ since it doesn't affect the solution of the minimization problem.

Then we can isolate the portion of the NMF objective that topic t is responsible for as

$$d_t(T_{t:}) = \frac{1}{2}||R_t - W_{:t}T_{t:}||_F^2 + \text{reg}(W, T). \tag{2.5}$$

The goal is to bound the objective resulting from reconstructing the t-th topic T_t as $\hat{T}$ minus an error vector Δ_t. As a result, the quantity we are interested in is $d_t(\hat{T}_t) = d_t(T_t + \Delta_t)$. We can analyze $d_t(T_t + \Delta_t)$ by expanding it using Taylor's Theorem,

$$d_t(T_t + \Delta_t) = d_t(T_t) + \left(\frac{\partial d_t}{\partial T_t}\right)^T \Delta_t + 0.5\Delta_t^T \left(\frac{\partial^2 d_t}{\partial T_t^2}\right) \Delta_t. \tag{2.6}$$

The first two partial derivatives (gradient, and Hessian) of d_t with respect to T are

$$\frac{\partial d_t}{\partial T_t} = ||W_t||^2 T_t - W_t^T R_t + \frac{\partial \text{reg}(W, T)}{\partial T} \tag{2.7}$$

and

$$\frac{\partial^2 d_t}{\partial T_t^2} = I_d||W_t||^2 + \frac{\partial^2 \text{reg}(W, T)}{\partial T^2} = I_d||W_t||^2 + H_{\text{reg}}. \tag{2.8}$$

We assumed that the W, T are the result of a local NMF optimization which has converged, therefore we have $\frac{\partial d_t}{\partial T_t} = 0$. Then we can rewrite Equation 2.6 as

$$d_t(T_t + \Delta_t) = d_t(T_t) + 0.5\Delta_t^T \left(\frac{\partial^2 d_t}{\partial T_t^2}\right) \Delta_t \tag{2.9}$$

Substituting 2.8 into 2.9 we get

$$d_t(T_t + \Delta_t) = d_t(T_t) + 0.5||\Delta_t||^2||W_t||^2 + \Delta_t^T H_{\text{reg}} \Delta_t \tag{2.10}$$

Note that for a positive semi-definite matrix H_{reg} we have that $x^T H_{\text{reg}} x \leq \lambda_1(H_{\text{reg}})||x||^2$. In order to avoid the dependence on λ_1, we let $S = \frac{H_{\text{reg}} + H_{\text{reg}}^T}{2}$, then $x^T H_{\text{reg}} x = x^T S x$. Since for symmetric matrices eigenvalues and singular values are equal $\lambda_1(S) = \sigma_1(S) = ||S||_2$. As a result $x^T H_{\text{reg}} x = x^T S x \leq \lambda_1(S)||x||^2 = ||S||_2||x||^2$.

Therefore, from the assumption that H_{reg} is positive semi-definite, we have that

$$d_t(T_t + \Delta_t) \leq d_t(T_t) + 0.5||\Delta_t||^2 \left(||W_t||^2 + ||S||_2\right) \tag{2.11}$$

Now we can put this back into the definition of a weight

$$v_t = \sqrt{||W_t||^2 + ||S||_2},$$

and substitute into Equation 2.2 (and re-introduce superscripts for clarity) to get

$$\begin{aligned}
LHS^{(m)} &= \mathcal{E}(X^{(m)}, T) && (2.12)\\
&= \sum_{t=1}^{k} d_t(T_{t:}) && (2.13)\\
&\leq \sum_{t=1}^{k} d_t(T_{t:}^{(m)}) + 0.5||\Delta_t||^2 \left(||W_t||^2 + ||S||_2\right) && (2.14)\\
&= \mathcal{E}(X^{(m)}, T^{(m)}) + \sum_{t=1}^{k} 0.5||\Delta_t||^2 v_t^2 && (2.15)\\
&= \mathcal{E}(X^{(m)}, T^{(m)}) + \sum_{t=1}^{k} 0.5||v_t\Delta_t||^2 && (2.16)\\
&= \mathcal{E}(X^{(m)}, T^{(m)}) + 0.5||(I_k\mathbf{v})\Delta^{(m)}||_F^2 && (2.17)\\
&= \mathcal{E}(X^{(m)}, T^{(m)}) + 0.5||(I_k\mathbf{v})(T^{(m)} - Y^{(m)}T)||_F^2 && (2.18)\\
&= RHS^{(m)}. && (2.19)
\end{aligned}$$

□

An interesting question is whether we can do better than this upper bound by potentially considering element-wise weighting for the $T^{(m)}$s rather than row-wise weighting. A simple example shows that we can not without making additional assumptions.

Claim 1. *The upper bound that results from Theorem 1 is tight.*

Proof. Let $X = I_3$, $\text{reg}(W, T) = 0$ and consider a $k = 1$ optimal NMF solution:

$$X = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \approx \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} \begin{pmatrix} \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \end{pmatrix} = WT \tag{2.20}$$

Then the additional error for using $\hat{T}$ is

$$0.5||\sqrt{3} \begin{pmatrix} \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \end{pmatrix} - y\hat{T}||_F^2, \tag{2.21}$$

Which is 0 when $y = \sqrt{3}$ and $\hat{T} = T$. □

In the absence of regularization, the expression in Equation 2.2 is in fact an equality for the additional error.

Corollary 2. *Suppose local models have converged, and reg*$(W, T) = 0$*, then*

$$\mathcal{E}(X^{(m)}, T) = \mathcal{E}(X^{(m)}, T^{(m)}) + \sum_{t=1}^{k} 0.5||\Delta_t||^2 v_t^2.$$

Proof. Similar to Theorem 1, except that we use Equation 2.10 directly rather than having to upper-bound the term that results from the regularization. □

We also provide a simple corollary based on a standard form of regularization.

Corollary 3. *Let* ℓ_1^T, ℓ_1^W, ℓ_2^T, *and* ℓ_2^W *be regularization parameters, each non-negative. Define*

$$reg(W, T) = \ell_1^T||T||_1 + \ell_1^W||W||_1 + 0.5\ell_2^W||W||_F^2 + 0.5\ell_2^T||T||_F^2$$

then $||S|| = \ell_2^T$.

We use the result of Theorem 1 to design Algorithm 6, WR-NMF. To use WR-NMF, each party must first fit a local model $X^{(m)} \approx W^{(m)}T^{(m)}$ until convergence, publish $T^{(m)}$, and publish the sum of the rows of $W^{(m)}$ squared element-wise. Then each party can locally compute the same global T independently, by agreeing on a random seed. As a final step, each party can fit a local $\hat{W}^{(m)}$ to minimize $\mathcal{E}(X^{(m)}, T)$, and then proceed with using $\hat{W}^{(m)}$ and T for their application of row-distributed NMF.

Algorithm 6 WR-NMF for Offline Row-Distributed NMF

Input: Number of global topics k, regularization parameter ℓ_2^T, each party m's $(\mathbf{v}^{(m)})^2 = \sum_{i=1}^{n_m}(W_{i:}^{(m)})^2$, and $T^{(m)}$.

Output: Global topics T that minimize the upper bound of Equation 2.2 (p.g. 59).

1: Scale $\hat{T}^{(m)} \leftarrow (I_{k_m}(\sqrt{\mathbf{v}^{(m)} + \ell_2^T}))T^{(m)}$

2: Combine $T_{\text{in}} \leftarrow \text{vstack}_{m=1}^{M} \hat{T}^{(m)}$

3: Solve $T \leftarrow \text{RRI-NMF}(T_{\text{in}}, k)$ using Algorithm 2 (p.g. 12).

Elementwise Weighted NMF We also provide a generalization of WR-NMF to the scenario when we wish to solve NMF with elementwise reconstruction penalties. This is useful e.g. in the recommender-system application, if we don't want to reconstruct unobserved entries, or if we don't want to give users who rate a lot of items too much weight. Let $\Psi^{(m)}$ be a $n_m \times d$ non-negative weight matrix, and suppose the mth party minimizes

$$\mathcal{E}_{\Psi^{(m)}}(X^{(m)}, T^{(m)}) = \frac{1}{2}||X^{(m)} - W^{(m)}T^{(m)}||_{\Psi}^2 + \text{reg}(W, T) \tag{2.22}$$

$$= \frac{1}{2}\sum_{i=1}^{n_m}\sum_{j=1}^{d}\Psi_{ij}(X_{ij}^{(m)} - W_{i:}^{(m)}T_{:j}^{(m)})^2 + \text{reg}(W, T). \tag{2.23}$$

We wish to find T that minimizes

$$\sum_{m=1}^{M}\mathcal{E}_{\Psi^{(m)}}(X^{(m)}, T) \tag{2.24}$$

Corollary 4. *Let $(\mathbf{x}^2)_j = x_j^2$ be the operator that squares a vector elementwise, and let S be defined as in Theorem 1. Then*

$$v_t^{(m)} = \sqrt{\left|\left|\left((W_{:t}^{(m)})^2\right)^T \Psi^{(m)}\right|\right| + ||S||}$$

will ensure that Algorithm 6 minimizes a weighted upper bound similar to Equation 2.2 (p.g. 59).

Proof. The framework is similar to the proof of Theorem 1, as result we will omit most of it.

Let

$$d_t(T_{t:}) = \frac{1}{2}||R_t - W_{:t}T_{t:}||_\Psi^2 + \text{reg}(W, T). \tag{2.25}$$

Then from [7] pp 117-118

$$\frac{\partial^2 d_t}{\partial T_t^2} = \left(W_{:t}^{(m)})^2\right)^T \Psi^{(m)} \tag{2.26}$$

and

$$\Delta_t^T \frac{\partial^2 d_t}{\partial T_t^2} \Delta_t \leq \left|\left|\left((W_{:t}^{(m)})^2\right)^T \Psi^{(m)}\right|\right|. \tag{2.27}$$

□

The algorithm is then to simply use these updated vs as inputs to Algorithm 6.

Related Work Our algorithm was inspired by literature on ensemble learning.

There are multiple methods that consider different clusterings of the same n documents, and require $O(n)$ communication/computation. In [74] the authors propose solving general ensemble clustering by requiring the ensemble clustering connectivity matrix[3] resemble a weighted average of the input clusterings connectivity matrices. In [75] the authors address the ensemble NMF problem. They propose weighing documents that are consistently (across different clusterings) clustered together more highly than pairs of documents that are inconsistently clustered. In [76] the authors suppose that for each clustering a different set of features was used to identify the document clusters. As a result they seek a consensus clustering in terms of connectivity matrices. Since NMF can be interpreted to be a clustering algorithm, it is reasonable to use the above methods as a solution (or at least an initialization) for the offline row-distributed NMF task. The main disadvantages as compared to Algorithm 6 are that (a) all parties are required to have the same documents (b) communication and computation is a function of n.

The closest work to ours comes from the ensemble topic modeling setting [77]. The authors study the task of ensembling multiple Latent Dirichlet Allocation models (Section 1.2.4 (p.g. 15)). Like our algorithm, in theirs communication does not depend on n. The main issues with their result stems from technical challenges posed by the LDA inference problem,

[3] A *connectivity matrix* is a $n \times n$ matrix M such that $M_{ij}(C) = 1$ iff documents i and j were assigned to the same cluster in clustering C.

and a short-coming of the inference algorithm they use. Like us, they also use the idea of treating topics as representatives for documents. However, rather than simply weighting topics appropriately, they must create document samples by repeatedly sampling from the topic distributions. As a result the computation cost of their inference step is still a function of n. A interesting research question is whether using a different inference algorithm for LDA would circumvent this 'topic weighting' issue.

Conclusion Algorithm 6, WR-NMF, has several interesting applications that can be explored:

1. As an algorithm to speed up NMF. First we partition X into $X^{(m)}$s. Then we interleave coarse-grained updates based on $T^{(m)}$s and fine-grained updates based on $X^{(m)}$s. This may help step out of local minima and provide regularization.

2. As a synchronization step between distributed NMF workers. Rather than requiring synchronization at every step as in our online algorithm (Section 2.2), we use Algorithm 6 every τ iterations for synchronization, greatly reducing communication cost and allowing asynchronous communication.

3. As an initialization algorithm for NMF, and distributed NMF. This is the application that we will examine after presenting our online algorithm.

Further investigation into when the bound of Equation 2.2 (p.g. 59) is tight is a interesting direction for future work. One issue is that requiring first derivatives to be equal to zero is very strict. If we relax this to require that they have norm at most ϵ, the resulting objective will have a $||\Delta_t||_1\epsilon$ term. This doesn't fit neatly with the $||\Delta_t||^2$ term, and so the resulting sub-problem is not solvable by standard NMF. For now we conjecture that in the absence of regularization, $(\text{reg}(W, T) = 0)$, and with $\epsilon = 0$, the bound will be tight.

2.2 PD-NMF: Algorithm for Online Row-Distributed NMF

Our algorithm for online row-distributed NMF is called Private Distributed NMF, PD-NMF. Even though we won't discuss privacy until Chapter 3, we will present the private version of our algorithm. We do this because privacy comes at essentially no cost. We rely

on light-weight cryptographic primitives that don't increase communication cost (which is the bottleneck) and only increase computation cost by a small constant.

Since PD-NMF is based on the centralized NMF, Algorithm 2 (p.g. 12), we review the NMF problem setting and the centralized algorithm. Given a rank $k \in \mathbf{N}$, the objective of NMF is to find non-negative matrices W $(n \times k\ )$, and T $(k \times d)$ that (locally) minimize

$$\mathcal{E}(W,T) = 0.5||X - WT||_F^2 + \text{reg}(W,T), \tag{2.28}$$

where we define the regularization term as

$$\text{reg}(W,T) = \ell_1^T||T||_1 + 0.5\ell_2^T||T||_F^2 + \ell_1^W||W||_1 + 0.5\ell_2^W||W||_F^2, \tag{2.29}$$

and $\ell_1^T, \ell_2^T, \ell_1^W, \ell_2^W$ are regularization parameters. Note that when either T or W is fixed, NMF reduces to non-negative least squares which can be solved efficiently (e.g. using quadratic programming). A consequence is that given X and T it's easy to find the optimal W.

We can separate 2.28 into a sum of k rank-one terms by defining the residuals R_t, where

$$R_t = X - \sum_{l \neq t}^{k} W_{:l}T_{l:} \qquad \text{for } t = 1, \ldots, k. \tag{2.30}$$

The residual R_t is the amount of X that is to be explained by basis t (the tth row of T). The objective now becomes

$$\mathcal{E}(W,T) = \sum_{t=1}^{k} \frac{1}{k} 0.5\|R_t - W_{:t}T_{t:}\|_F^2 + \text{reg}(W,T).$$

Recall that the centralized RRI-NMF method [7] alternates between fixing $W_{:t}$ (resp. $T_{t:}$) and optimizing for $T_{t:}$ (resp. $W_{:t}$) sequentialy for each t until a convergence condition is met. The essential step is the iterative update of each topic t, given by the equations of [7] as well as our adaptations for negative 2nd derivative,

$$\begin{aligned}
&\text{for } t = 1, \ldots, k: \\
W_{:t} \leftarrow &\begin{cases} \frac{[R_t T_{t:}^T - \ell_1^W \mathbf{1}_n]_+}{||T_{t:}||_2^2 + \ell_2^W} & \text{if } ||T_{t:}||_2^2 + \ell_2^W > 0 \\ \text{Corollary 1 with } \mathbf{w} = -R_t T_{t:} + \ell_1^W \mathbf{1}_n,\ c = ||T_{t:}||_2^2 + \ell_2^W & \text{otherwise.} \end{cases} \\
T_{t:} \leftarrow &\begin{cases} \frac{[W_{:t}^T R_t - \ell_1^T \mathbf{1}_d]_+}{||W_{:t}||_2^2 + \ell_2^T} & \text{if } ||W_{:t}||_2^2 + \ell_2^T > 0 \\ \text{Lemma 1 with } \mathbf{w} = -R_t^T W_{:t} + \ell_1^T \mathbf{1}_d,\ c = ||W_{:t}||_2^2 + \ell_2^T & \text{otherwise.} \end{cases}
\end{aligned} \tag{2.31}$$

The important take-away is that we can design an algorithm to solve Problem 2.28 that

de-couples the update of W and T. The update of W is a sum of columns of the data X (since R is a function of X by Equation 2.30) and can be done locally by each party. The update of T depends on a sum of rows of the data X. Each party can evaluate their portion of the sum (given by the expression $(W_{:t}^{(m)})^T R_t^{(m)}$) on their $X^{(m)}$, and then communicate.

Inter-party communication Suppose that the mth party computes their share of the update to $T_{t:}$, $(W_{:t}^{(m)})^T R_t^{(m)}$ using Equation 2.31. We assume that we are in the peer-to-peer communication model, and due to processing power we are limited by the latency of our communication rather than bandwidth. To minimize latency each party will send a copy of their share to each of the remaining $M-1$ parties, and will receive a share from each of the remaining $M-1$. This means that each party's total communication is $2(M-1)d$ doubles per round, in order for each party to learn $\sum_{m=1}^{M}(W_{:t}^{(m)})^T R_t^{(m)}$. Each party also learns every other party's contribution to the sum, but this isn't required for NMF. As a result, we use a lightweight cryptographic protocol called SecureSum that only reveals the sum of inputs to each party. To use SecureSum each party has to communicate $2(M-1)d$ longs per round, along with a small amount of computation.

SecureSum is a distributed peer-to-peer protocol among M parties $p_1, \ldots, p_M$. It takes an input from each p_m a vector $\mathbf{x}^{(m)}$ and outputs (to everyone) the sum $\sum_{m \in M} \mathbf{x}^{(m)}$. Any subset of parties only learns this sum and no additional information about the inputs of other parties. More formally, SecureSum satisfies two security properties for any given (distribution of) inputs $\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(m)}$. *Correctness:* Every party receives the sum $\sum_{m \in M} \mathbf{x}^{(m)}$ as their output. *Security:* Any coalition of parties can simulate their entire joint protocol view given only their inputs and the output. The joint protocol view is what these parties see, so if they can generate it with only their inputs and the output, then they learn nothing more beyond what is contained in their inputs and the output [78].

We give an example of how the SecureSum protocol can be implemented in with M=3 parties. Party 1's input is a vector $x^{(1)} \in R^d$. First they convert it to a vector $\hat{x}^{(1)}$ in a finite field F^d, e.g. using a fixed decimal place. Then party 1 generates random vectors $r_1^{(2)}, r_1^{(3)} \in F^d$, and picks $r_1^{(1)} = \hat{x}_1 - r_1^{(2)} - r_1^{(3)}$; i.e. $r_1^{(1)} + r_1^{(2)} + r_1^{(3)} = \hat{x}_1$. Party 1 sends $r_1^{(2)}$ to party 2 and $r_1^{(3)}$ to party 3. The other parties generate random vectors that satisfy the same constraints for their x_ms. The resulting information known to each party is displayed

in Table 2.1.

Table 2.1: Example of SecureSum with $M = 3$

	Known by party			
Known by party	1	2	3	Row $\sum$s to:
1	$r_1^{(1)}$	$r_1^{(2)}$	$r_1^{(3)}$	$\hat{x}^{(1)}$
2	$r_2^{(1)}$	$r_2^{(2)}$	$r_2^{(3)}$	$\hat{x}^{(2)}$
3	$r_3^{(1)}$	$r_3^{(2)}$	$r_3^{(3)}$	$\hat{x}^{(3)}$
All	$r^{(1)} = \sum r_m^{(1)}$	$r^{(2)} = \sum r_m^{(2)}$	$r^{(3)} = \sum r_m^{(3)}$	$\sum \hat{x}^{(m)}$

Then each party sends $r^{(m)}$, to the remaining $M-1$ parties, and each party locally computes $\sum r^{(m)} = \sum x^{(m)}$. Correctness of the protocol follows from the fact that the sum of the sum of rows (which is the sum we want to compute) equals the sum of the sum of columns (which is how we compute it). We can achieve communication parity by using a pseudo-random number generator (PRNG) to generate the $r_i^{(j)}$s. Party i, j simply have to coordinate a random seed for their corresponding PRNGs, and they can each compute the same rs without having to communicate. Then each party only needs to distribute $r^{(m)}$ which is a d-dimensional vector to $M-1$ parties, and receive such a vector from $M-1$ parties, meaning the communication cost is $2(M-1)d$ longs, which is the same as the communication when not using SecureSum.

Our implementation assumes parties are *semi-honest*. This is a standard assumption in cryptography/security, and it means that while parties do not deviate from the protocol, they may collaborate to infer additional information. There are cryptographic methods to detect that each party executes SecureSum faithfully. However, we can not easily detect if a malicious (not semi-honest) party is sending inputs to SecureSum that would compromise privacy of PD-NMF.

As a preview of Chapter 3, the fact that the observed output $W_{:t}^T R_t$ is the sum of functions of iid data points (rows of X) and hence applicability of the Central Limit Theorem is the intuition behind what makes PD-NMF private with respect to the absence or presence of a individual data point (a single row of X) as $n \to \infty$.

Private Distributed RRI-NMF Iterations (PD-NMF-Iter) We assume we have an implementation SecureSum, and use it to implement a private distributed RRI-NMF

Algorithm 7 Private Distributed RRI-NMF (PD-NMF-ITER), party m's view of the protocol.

Input: Local data $X^{(m)}$, initial topics T, regularization parameters $\ell_1^T, \ell_2^T, \ell_1^W, \ell_2^W$, access to SECURESUM

Output: Global topics T^G

1: **repeat**
2: **for** $t \in \{0, 1, \ldots, k\}$ **do**
3: $R_t^{(m)} \leftarrow X^{(m)} - W^{(m)}T + W_{:t}^{(m)}T_{t:}$
4: $W_{:t}^{(m)} \leftarrow \frac{[R_t^{(m)}T_{t:}^T - \ell_1^W \mathbf{1}_{n_m}]_+}{||T_{t:}||_2^2 + \ell_2^W}$
5: SUMWR $\leftarrow$ SECURESUM$(W_{:t}^{(m)T} R_t^{(m)})$
6: SUMWW $\leftarrow$ SECURESUM$(||W_{:t}^{(m)}||_2^2,)$
7: $T_{t:} \leftarrow \begin{cases} \leftarrow \frac{[\text{SUMWR} - \ell_1^T \mathbf{1}_d]_+}{\text{SUMWW} + \ell_2^T} & \text{if } ||W_{:t}||_2^2 \quad + \ell_2^T > 0 \\ \text{Lemma 1 with } \mathbf{w} = -\text{SUMWR} + \ell_1^T \mathbf{1}_d,\ c = \text{SUMWW} + \ell_2^T & \text{otherwise.} \end{cases}$
8: $T_{t:} \leftarrow \arg\min_{x \geq 0, ||x||_1 = 1} ||x - T_{t:}||_2$
9: **end for**
10: **until** Convergence criteria are met

iterations. The term 'iterations' means that we are making iterative improvements to an initial set of topics T_0. We call this algorithm PD-NMF-ITER, and its details are presented in Algorithm 7. Our full algorithm PD-NMF consists of first running PD-NMF-INIT (presented later) to get an initial set of T_0, and then use T_0 as the initial topics for PD-NMF-ITER.

We can prove a simple but essential (in terms of the learning outcome) theorem about PD-NMF-ITER:

Theorem 2. *Starting from the same initial topics T_0,* PD-NMF-ITER *(Algorithm 7) converges to the same solution T as the centralized* RRI-NMF *(Algorithm 2).*

Proof. Algorithm 7 was derived by breaking up a matrix-vector product into a sum of sub-matrix vector products, and this forms the basis for this result. Note that T and the regularization constants are known to all parties for all iterations.

First, for the update of W in 2.31, let I_m be the rows of X that belong to party m. Then in centralized RRI-NMF the update for $W_{I_m t}^{(m)}$ depends on $(R_t[I_m, :]T_{t:}^T)^{(m)}$ and globally known information. Since $R_t[I_m, :] = R_t^{(m)}$, each party can compute this locally, which gives step 4 in the algorithm.

The update to the topics T requires $\|W_{:t}\|_2^2$ and $W_{:t}^T R_t$ which are given by sums,

$$\|W_{:t}\|_2^2 = \sum_{m=1}^{M} \|W_{:t}^{(m)}\|_2^2 ; W_{:t}^T R_t = \sum_{m=1}^{M} W_{:t}^{(m)T} R_t^{(m)}.$$

($R_t^{(m)}$ is party m's residual matrix.) Since these sums are computed using SecureSum in steps 5, 6 we know that no additional information is leaked to the network, and that the output is correct. Step 7 combines these sums in the same way as the update in Equation 2.31. □

Initializing T with PD-NMF-Init All heuristics for NMF (including RRI-NMF) are iterative and must be initialized, however there are several classes of initialization algorithms each with their pros and cons.

For non-private NMF there are several state-of-the-art initialization algorithms [11], [79], [80] that are comparable in terms (a) computational complexity (b) solution quality after τ iterations. We will use the SVD-based initialization (referred to as `nndsvd`) of [11] as a representative of this class. It produces T_0 from a row-basis for the top singular non-negative subspaces of X. While convergence from T_0 is quick, `nndsvd` needs access to X or a private distributed SVD algorithm.

A much simpler initialization algorithm (`random`) is to set each entry of W, T uniformly at random from $U[0, 1]$. This doesn't require access to X, and is private by definition. However an issue is that after τ iterations, NMF initialized from `random` will typically be at a worse solution than when initialized from `nndsvd`. A simple work-around (`random 2x`) is to simply run NMF for 2τ iterations after initializing with `random`.

We propose a distributed initialization algorithm PD-NMF-Init based on WR-NMF that is private if PD-NMF-Iter is private, and comparable to `nndsvd` after τ iterations. The essential idea is to have each party compute a local model first. Then we use PD-NMF-Iter to solve the WR-NMF problem where documents are represented by weighted topics; This *first* run of PD-NMF-Iter treats weighted topics as documents, and is initialized from a random set of topics. This gives us a set of topics T_{init}, which we feed into a *second* instantiation of PD-NMF-Iter, this time run on the actual documents $X^{(m)}$. The combination of PD-NMF-Init and PD-NMF-Iter corresponds to PD-NMF.The details for PD-NMF-Init are presented in Algorithm 8.

Algorithm 8 PD-NMF-INIT: initialization by merging local topic models, party m's view of the protocol.

Input: Local data $X^{(m)}$, number of topics k, access to PD-NMF-ITER, access to SECURESUM

Output: Initial global topics T_{init}

1: Initialize $T_{\text{local}} \leftarrow \texttt{nndsvd}(X^{(m)}, k)$.
2: $W^{(m)}, T^{(m)} \leftarrow \text{RRI-NMF}(X^{(m)}, k, T_{\text{local}})$.
3: Initialize $V^{(m)} \leftarrow 0^{k \times k}$
4: Set $V^{(m)}_{jj} \leftarrow \sqrt{\sum_{i=1}^{n} (W^{(m)}_{ij})^2}$.
5: Scale $\hat{T}^{(m)} \leftarrow V^{(m)} T^{(m)}$.
6: Sample $T_0^{(m)} \sim U[0,1]^{k \times d}$.
7: Compute $T_0 \leftarrow \text{SECURESUM}(\frac{1}{M} T_0^{(m)})$.
8: Normalize T_0's rows to sum to 1.
9: $T_{\text{init}} \leftarrow \text{PD-NMF-ITER}\ (\hat{T}^{(m)}; T_0)$.

2.3 PD-SVD: Algorithm for Online Row-Distributed SVD

Suppose all M parties have agreed upon which set of d features to use to represent their inputs, and what order these features should be in. Then each party's local input is a $n_m \times d$ matrix $X^{(m)}$, where each row corresponds to a document, and each column corresponds to a feature. Let $n = \sum_m n_m$ be the combined number of documents among all M parties, and let $X = \text{vstack}_{m=1}^{M} X^{(m)}$ be the combined data matrix, formed by vertically stacking the m parties' matrices. X is a $n \times d$ matrix.

The *k-truncated Singular Value Decomposition (SVD)* of X is defined as $X = U_k \Sigma_k V_k^T$. U_k is a $n \times k$ matrix containing the left singular vectors, V is a $d \times k$ matrix containing the right singular vectors, and Σ is a $k \times k$ diagonal matrix containing the top k singular values by magnitude corresponding to each of the singular vector pairs. As a result the *rank-k approximation* of X is $X_k = \sum_{t=1}^{k} \Sigma_{tt} U_{:t} V_{:t}^T$.

The goal of private distributed SVD, PD-SVD, is for all parties to learn the exact V_k while limiting, to the extent possible under the exactness constraint, what a party m learns about $U_k^{(m')}$, and $X^{(m')}$. If we don't care about hiding Σ_k, we can achieve the above by relying on SECURESUM. However, if we want to hide Σ_k, we will need a more powerful, and expensive (but still $O(1)$) cryptographic primitive called the normalized secure sum, NORMEDSECURESUM.

This problem setting is visualized in Figure 2.2.

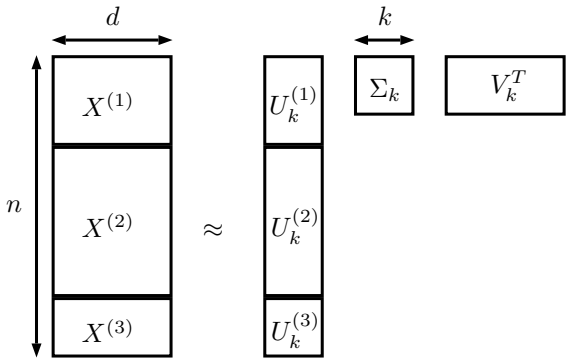


Fig. 2.2: Row-Distributed k-truncated SVD with $M = 3$

Most methods for finding the singular values/vectors of a matrix reduce the problem to finding eigen values and vectors of a symmetric matrix [81]. We use the $d \times d$ symmetric matrix $S = X^T X$, so that the eigenvalue decomposition reveals the right-singular vectors V_k:

$$\begin{aligned} S &\approx V_k \Sigma_k U_k^T U_k \Sigma_k V_k^T \\ &= V_k \Sigma_k^2 V_k^T. \end{aligned}$$

This formulation of S isn't optimal for numerical stability in the centralized setting when privacy and communication costs aren't a concern. However in the distributed setting it's useful because the left-singular vectors U_k are hidden, and communication doesn't depend on n.

In order to find the top k eigenvectors we will use block power iterations. This is a standard method, whose convergence rate depends on the ratio between successive singular values.

Normalized Secure Sum In order to avoid revealing Σ_k in the power iterations, we will use a cryptographic primitive called *the normalized secure sum*, which we denote as NormedSecureSum, presented in Figure 2.3.

Since NormedSecureSum is difficult to implement in a easy to analyze manner, we analyze its performance empirical. In Table 2.2 we report the total wall-time that it takes to compute the normalized secure sum for vectors of various d among various amounts of parties M. This includes conversion to and from fixed-point representation as well as the online phase in the SPDZ framework. The cost of the offline phase is not included since it can be run without

Each party inputs a vector x_m. Let $s = \sum_{m=1}^{M} x_m$ be the sum of the inputs. The output to each party is the normalized sum

$$\textsc{NormedSecureSum}(\{x_m\}_{m=1}^{M}) = \frac{s}{||s||_2}.$$

There exist implementations that are

Correct The output is the same up to a specified precision as if it was computed in the clear.

Secure Each party only learns the output and information deducible from the output.

Efficient Communication and computation overhead is $O(1)$, with small constant, above what it would take to perform the normalized sum in the clear.

Fig. 2.3: Properties of NormedSecureSum

access to $X^{(m)}$. This means the objects pre-computed in the offline phase can be used for diverse MPC applications (although they cannot be re-used).

Table 2.2: Execution Time in Seconds per One Invocation of NormedSecureSum for Various Vector Dimensionality d and Number of Parties M

M	d		
	100	1000	10000
2	1.4	11.5	113.9
3	2.1	17.4	168.1
4	3.5	29.3	293.0
5	4.6	40.3	403.9

For $d = 100$, typical for PCR, the cost per iteration is about 2 sec with $M = 3$. This means $\tau = 300$ iterations of PCR with $k = 7$ can be computed in roughly 1.2 hours using PD-SVD. For LRA $d = 10000$ may be more typical, in which case the same would take roughly 4 days. For the latter case if such a run-time is too long, PD-SVD can be run with SecureSum replacing NormedSecureSum (in step 6 of Algorithm 10) to make communication overhead smaller; then PD-SVD could compute the same LRA in less than 30 minutes. To inform this trade-off, we examine the effect of revealing Σ, and hence the privacy benefit of NormedSecureSum over SecureSum, in Section 3.4.2.

Algorithm 9 Non-private block power iteration.

Input: S: $d \times d$ symmetric matrix, $k \in \mathbf{Z}_{\geq 0}$ truncation rank, $\tau \in \mathbf{Z}_{\geq 0}$ number of power iterations.

Output: $\hat{V}_k$: estimates of the top k eigenvectors of S.

1: $\hat{V} \leftarrow \mathcal{N}(0, \frac{1}{d})^{n \times d}$
2: **for** $t \in \{1, 2, \ldots, \tau\}$: **do**
3: $\quad \hat{V}_{\leftarrow} S\hat{V}$
4: $\quad \hat{V} \leftarrow$ Orthonormalize $(\hat{V})$
5: **end for**
6: Output $\hat{V}$ as an estimate of V_k.

Block Power Iterations Before describing our implementation of block power iterations using NormedSecureSum, we describe the centralized (one-party) version as Algorithm 9. In order to derive a private version we need to perform the *initialization* (step 2), *iteration* (step 3) in a private and distributed way.

For initialization, each party m simply samples

$$\hat{V}^{(m)} \sim \mathcal{N}\left(0, \frac{1}{\sqrt{M}d}\right)^{n \times d}.$$

Then

$$\sum_{m=1}^{M} \hat{V}^{(m)} \sim M\mathcal{N}\left(0, \frac{1}{\sqrt{M}d}\right)^{n \times d} = \mathcal{N}\left(0, \frac{1}{d}\right)^{n \times d}.$$

For iteration first note that step 3 actually reveals Σ_k, i.e. $\hat{\sigma}_i = ||\hat{V}_{:i}||_2$. However these sums are not necessary for finding V_k, since they are destroyed by orthonormalization (step 4). As a result we can hide them using NormedSecureSum. To do so, note that S can be broken down into M sums,

$$\begin{aligned} S &= X^T X \\ &= \sum_{i=1}^{n} X_{i:}^T X_{i:} \\ &= \sum_{m=1}^{M} \sum_{i_m=1}^{n_m} \left(X_{i_m:}^{(m)}\right)^T X_{i_m:}^{(m)} \\ &= \sum_{m=1}^{M} \left(X^{(m)}\right)^T X^{(m)}. \end{aligned}$$

Algorithm 10 Private-Distributed SVD, party m's view.

Input: $X^{(m)}$: $n_m \times d$ data matrix, $k \in \mathbf{Z}_{\geq 0}$ truncation rank, $\tau \in \mathbf{Z}_{\geq 0}$ number of power iterations, number of parties M, access to SecureSum and NormedSecureSum.

Output: $\hat{V}_k$: estimates of the top k eigenvectors of S

1: $S^{(m)} \leftarrow \left(X^{(m)}\right)^T X^{(m)}$
2: $\hat{V}^{(m)} \leftarrow \mathcal{N}(0, \frac{1}{\sqrt{Md}})^{n \times d}$
3: $\hat{V} \leftarrow \text{SecureSum}(\hat{V}^{(m)})$.
4: **for** $t \in \{1, 2, \ldots, \tau\}$: **do**
5: **for** $i \in \{1, 2, \ldots, k\}$: **do**
6: $\hat{V}_{:i} \leftarrow \text{NormedSecureSum}(S^{(m)}\hat{V}_{:i})$
7: **end for**
8: $\hat{V} \leftarrow$ Orthonormalize $(\hat{V})$
9: **end for**

Then, the ith column of step 3 can be calculated as

$$\text{NormedSecureSum}\left(\left\{\left(X^{(m)}\right)^T X^{(m)}\hat{V}_{:i}\right\}_{m=1}^{M}\right).$$

We simply repeat this (or execute it in parallel) for all k columns. The last remaining step is orthonormalization which should be performed using the same algorithm by every party, e.g. the modified Gramm-Schmidt [81]. We call this procedure the Private-Distributed SVD. Party m's view of PD-SVD is presented as Algorithm 10.

Note that if we don't want to hide Σ_k we can simply have each party us SecureSum in place of NormedSecureSum, so that they observe $\hat{\sigma}_i = ||V_{:i}||_2$ for $i \in \{1, 2, \ldots, k\}$. This is also more computationally efficient.

2.4 WR-NMF and PD-SVD as Distributed Initialization Algorithms

We conclude Chapter 2 by comparing how well WR-NMF and PD-SVD perform as distributed initialization algorithms for PD-NMF. We examine the use of distributed NMF for topic modeling on the 20NG [61] and Enron [60] datasets, as well as for recommender systems on the ML-1M [82] and Yelp [83] datasets.

Recall that PD-NMF-Iter converges to the same solution as the centralized RRI-NMF

given the same initialization. As a result the question we investigate here is how are convergence rate and solution quality affected by choice of initialization.

Two commonly used initializations for the centralized NMF are based on the SVD, called `nndsvd` [11], as well as a `random` initialization. A simple benchmark is to compare initializations to how well they perform to running the twice as many iterations starting from the `random` initialization; we call this `random_2x`.

While the `random` initialization is easy to implement in a distributed context, `nndsvd` requires the SVD. We use PD-SVD for each party to compute Σ_k, and V_k and then they each compute their local

$$U_k^{(m)} = X^{(m)} V_k \frac{1}{\Sigma_k},$$

which can then be used as inputs to the `nndsvd` initialization.

We also implement PD-NMF-Init, which depends on WR-NMF. This means we need the local topics $T^{(m)}$ each of M parties computes, as well a weight for each topic in each party party's $T^{(m)}$. For the topic modeling task we use weights as dictated by Theorem 1 (p.g. 60). Whereas for the recommender system task we use weights as dictated by Corollary 4 (p.g. 64).

In Figure 2.4 we compare initializations for the topic modeling task, and in Figure 2.5 we look at recommender systems. For each dataset we tried $k = 5$ and $k = 50$; the idea is that it is harder to find 50 good initial topics than it is to find 5 good initial topics. For each initialization we perform 10 trials; the plots show the mean objective $\pm$ one standard deviation of the mean. If a line is missing from a plot, that means the corresponding initialization's reconstruction error was worse than the highest value on the y-axis. For WR-NMF and PD-SVD we tested the initialization with $M \in \{3, 9, 27\}$ parties, but only present the $M = 9$ setting; the other settings were within one standard deviation of the mean of the $M = 9$ setting.

The conclusions from our experiments can be summarized as follows:

Topic Modeling, $k = 5$ WR-NMF is a clear favorite, only losing to `random_2x` on Enron.

Topic Modeling, $k = 50$ WR-NMF is still a clear favorite, however `nndsvd` and its distributed variant that comes from PD-SVD are catching up.

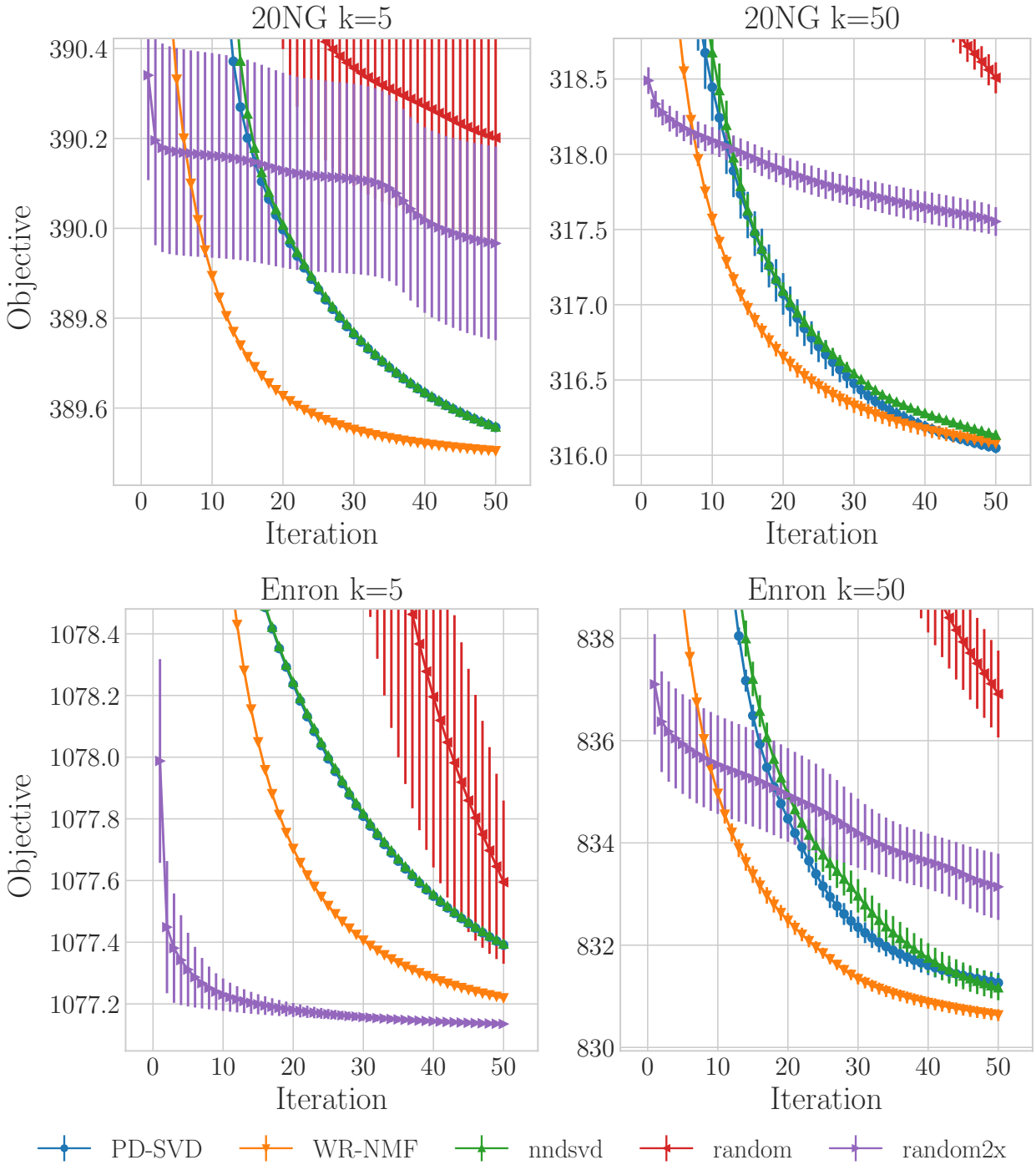


Fig. 2.4: Comparison of NMF Initializations for Topic Modeling

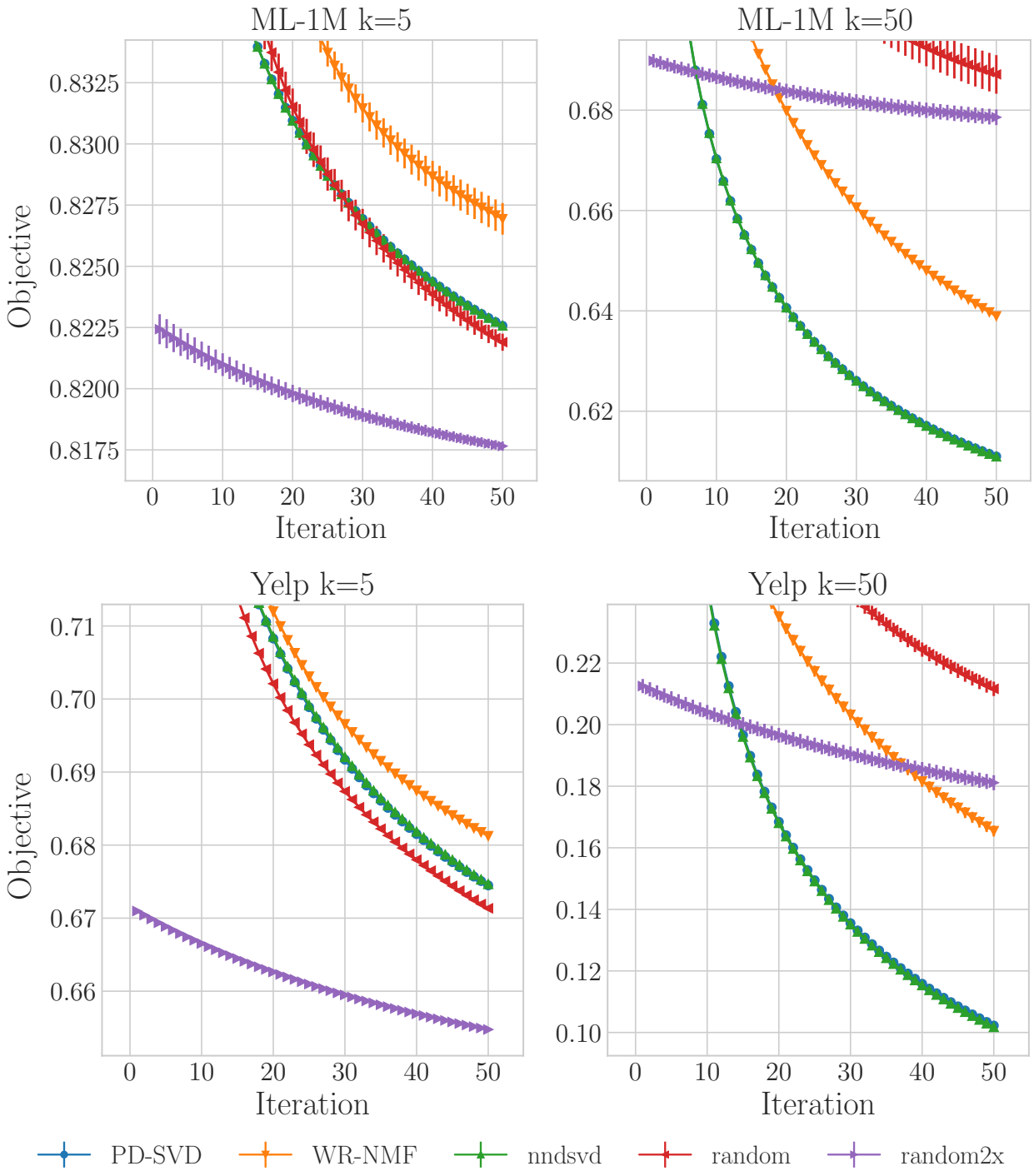


Fig. 2.5: Comparison of NMF Initializations for Recommender Systems

Recommender Systems, $k = 5$ `random_2x` and `random` perform very well.

Recommender Systems, $k = 50$ `nndsvd` and PD-SVD are clear favorites. However both WR-NMF performs better than `random_2x`, so it can be used in the distributed setting.

Overall our conclusion is that for topic modeling WR-NMF is always a good initialization choice. For recommender systems with small k, `random` is a reasonable initialization method. For medium k the initialization based on PD-SVD seems like the best choice.

CHAPTER 3

Empirical Measures of Privacy for Distributed Machine Learning

More data is better for all forms of learning. However, in many interesting domains one organization cannot collect all available data. Therefore the best learning outcome requires sharing data between organizations. An issue is that sharing raw data between organizations can be economically undesirable and/or subject to legislation, such as the nascent GPDR [73]. For example, consider the following applications where-in private distributed learning may be useful:

- *Private Distributed Topic Modeling.* Government agencies with protected data (FBI, CIA, NSA, ...) wish to build a distributed peer-to-peer information retrieval system, and doing so requires a topic model over all their data [84], [85].
- *Private Distributed Recommender Systems.* Businesses with proprietary consumer data would like to build recommender systems which can leverage data across all the businesses without compromising the privacy of any party's data [86].

In such applications, a key constraint is that each organization already has a system that serves their needs to some extent. Thus an organization's motivation for participating in distributed learning is in *improving* the quality of their local system while ensuring their data remains *private*.

A naive approach to privacy is for each of M parties to publish an anonymized version of their database by removing identifiers. A newer and more sophisticated approach is called k-anonymity. It requires that when the rows of a database are grouped by any subset of columns, then each subset has at least k rows. o achieve this one typically has to bin columns, e.g. instead of reporting that someone's age is 16 we would report that their age is between

Portions of this chapter have been submitted to M. Tsikhanovich, M. Magdon-Ismail, and V. Zikas, "PD-NMF: Private distributed non-negative matrix factorization," Submitted to ICML, 2018 and M. Tsikhanovich, M. Ishaq, M. Magdon-Ismail, *et al.*, "PD-SVD: Private distribute singular value decomposition," Submitted to UAI, 2018.

15 and 24. However both of these approaches are susceptible to many de-anonymization attacks [87], [88].

Let X be a database, and let X' be a neighboring database. We say two databases are *neighboring* if they differ in at most one row.
Suppose we want to output a function f of the database, and we use a mechanism M_f to generate the output. The output of M_f will be a random variable in the space Ω; let $A \subseteq \Omega$. The mechanism M_f is said to be (ϵ, δ)-*Differentially Private* if with probability at least $1-\delta$

$$\exp(-\epsilon)P(M_f(X') \in A) \leq P(M_f(X) \in A) \leq \exp(\epsilon)P(M_f(X') \in A). \tag{3.1}$$

Lower ϵ and lower δ correspond to more privacy.

Fig. 3.1: Differential Privacy

In reaction to cycles of new privacy techniques being proposed, and then broken, the cryptography community has developed a definition of privacy with provable post-processing guarantees. This methodology is called *differential privacy (DP)* [89], defined in Figure 3.1. Intuitively, differential privacy places a requirement on the mechanism that evaluates functions of our database. Specifically, we consider what happens when the database changes by including or removing one individual's records. DP requires that this change has a bounded effect on the change in probability mass function for the output of our mechanism.

To provide a concrete example, suppose the Medifax medical informatics company is paying people to participate in a survey on how smoking affects the pitch of a persons voice. They are open about the fact that they sell access to this data through a $\epsilon = 0.001, \delta$ =1e-6 Differentially Private mechanism, which takes as input someone's voice features and outputs a probability of them being a smoker. Johan, who smokes, is considering participating in the survey. Medifax tells Johan, that currently their model outputs P(smoker)=0.93 based on his voice. They ask him if he would like to answer whether or not he actually smokes. Johan is worried that if he participates in the survey, his insurance company will be able to learn that he's a smoker when they talk to him on the phone. Johan's insurance company uses Medifax's data to calculate quotes, and charges a lifetime amount of $1000 per percentage probability that someone is a smoker. How much would Medifax have to pay Johan to participate? Since they output a $\epsilon = 0.001$ differentially private prediction, Johan calculates that with very high probability P(smoker) could change by at most exp(0.001)0.93=9.3e-4. This costs Johan $93 in expectation, so he should be happy to participate if he is offered

more than \$93.[4]

An important property of differential privacy is *immunity to post-processing*. This means that if the output of a mechanism M is (ϵ, δ)-DP, then $f(M)$ is also (ϵ, δ)-DP for **any** f. This is a very strong requirement, and partly a reaction to previous de-anonymization attacks. For example in one instance an anonymized health-records database was de-anonymized by linking ZIP codes and ages to a voter-records database. Immunity to post-processing against arbitrary f means that with high probability rows in a differentially private health-records database couldn't be de-anonymized even when faced with arbitrary computational power and auxiliary information.

So how are differentially private mechanisms built? Suppose we wish to output a private version of a function f on our database X, which must be optimized with respect to some parameters θ. Suppose that we use gradient descent, to find a point where $\nabla f(X;\theta) = 0$. Essentially DP is achieved by adding noise ξ to one or more of [90]

1. The objective function, so that we minimize $f(X;\theta) + \xi(\theta)$.
2. The gradients within the optimization algorithm so that we seek $\nabla f(X;\theta) + \xi(\theta) = 0$
3. The final output, so that we output $\theta + \xi$.

While DP guarantees privacy (with high probability) against an adversary who performs arbitrary post-processing, the cost is that the quality of the machine learning outcome *must* be affected. For example, the gradient descent optimizer that satisfies DP is *not allowed to* converge to the exact minimum found by the non-DP optimizer regardless of the number of iterations performed. Otherwise the arbitrarily powerful adversary could try all (combinatorially many) possible input databases, and determine which one yields the same exact minimum as the DP minimizer, compromising privacy of the entire database. In some cases, not converging to the same minimum as the non-private minimizer could be beneficial in terms of generalization error [91], [92]. However, the necessity of this 'learning gap' means that in building a DP-based solutions one faces the problem of having to analyze the effect

[4]Currently many people are happy to give away their data in exchange for not having to pay for a service, or for convenience. We believe a lot of the apathy comes from lack of awareness of what the costs of privacy are. In addition to building private tools, it's very important for researchers to work on building awareness of the costs of privacy. Some domains that should be easy wins are contacts in a social network, search, and browsing histories, and purchase histories.

of this learning gap on model quality measures on a per-application basis. In our setting, where M parties that already have a working model are collaborating in order to improve their model, this problem is particularly important. For example, if a recommender system produced by DP a mechanism is worse than a system one party could construct from its own data, then this party gained nothing and lost some privacy by participating in distributed learning. Therefore, for the M-party distributed learning setting, we take it as a fundamental constraint that the *learning outcome of the private distributed algorithm must match the centralized*[5] *algorithm's outcome.*

Recently, cryptographically secure multi-party computation (MPC) protocols have been proposed for learning from private distributed data. MPC allows M mutually distrustful parties to perform a private computation on their combined data. The goal of MPC is *security*, i.e. the ability to identify and prove exactly what is revealed to each party during the course of the communication. In theory, one can run an arbitrary optimization/inference algorithm by 'compiling' it to some type of 'MPC byte code' and then executing this on a 'distributed virtual machine.' We saw examples of simple MPC protocols such as the SecureSum and NormedSecureSum in Chapter 2. Typically algorithm-agnostic MPC incurs intolerable communication and computation overhead [93]. Recently, the cryptography/security community has produced optimized MPC protocols for machine learning by taking advantage of trusted third parties [94], [95]. While we believe this vein of research is very interesting, and important, for this work we focus on the scenario where there are no trusted third parties, and the distributed algorithm must be run entirely peer-to-peer.[6] Furthermore, even though MPC guarantees security, it doesn't say anything about privacy, or what can be inferred from the outputs of a computation. In the NMF application the output is the T topic matrix. An arbitrarily powerful adversary can break privacy by trying all input combinations here as well. his means that despite its heavy costs, MPC alone doesn't guarantee privacy.

As a result, we are interested in learning algorithms that are computationally efficient, and are guaranteed to reconstruct the centralized solution. In this scenario, the interesting question is: what kind of privacy guarantees can we provide, and what kind of privacy

[5]By centralized solution we refer to the solution where all data is aggregated non-privately to a central server, where non-private NMF is used to compute the solution.

[6]In the trusted third party setting, many privacy concerns are trivially solvable by having the third party perform all computations.

Let m be a organization who is considering participating in a distributed learning computation with $M-1$ other organizations. To define economically incentivized privacy, m needs three models [96]:

Privacy Cost Model the cost of a coalition of A adversaries learning whether or not document x is in m's database $X^{(m)}$.

Learning Uplift Profit Model the expected improvement in model quality and resulting improvement in m's profits (taking into account that others' models may improve as well).

Adversary Model the probability of the adversary identifying whether or not row x is in m's database.

If m has the above three models, they can estimate their expected privacy-risk adjusted profit from participating in distributed learning.

Fig. 3.2: Economically-Incentivized Privacy

guarantees are useful? As a result, we focus on privacy derived from economic incentives, defined in Figure 3.2.

In this chapter we focus on the adversary model, as well as the learning uplift. However we will not estimate the monetary profits and losses resulting from our models. Estimating profits is typically business specific, and a cross-disciplinary endeavor that involves people who understand how a particular business operates. Estimating the cost of privacy is also typically business specific, although it's an interesting problem and is starting to receive some attention [97]. Differential Privacy gives guarantees against adversaries with arbitrary computational power, and arbitrary auxiliary information. However providing learning guarantees is still very much a developmental area for differential privacy [98]. We will focus on semi-honest adversaries with polynomial computational power and no auxiliary information. Additionally we will only be able to guarantee privacy against a pre-defined set of post-processing functions. We provide *empirically measurable adversary models*, wherein each party is able estimate the adversary's probability of identifying a document by running an algorithm on their dataset. This is in contrast to differential privacy, where the adversary model results from the theoretical properties of how the mechanism is designed. Our learning uplift is theoretically guaranteed as more data is used. We focus on experimentally showing how much row-distributed NMF and SVD (which were introduced in Chapter 2) improve.

In Section 3.1 we introduce a simple database, where each user simply has a 0 or a 1. We show how differential privacy is related to our measures of privacy. In Section 3.2 we present

our first empirical measure of privacy, based on the Kolmogorov-Smirnov hypothesis test. In Section 3.3 we compare the learning outcome of PD-NMF and PD-SVD to differentially private variants thereof. In Section 3.4 we use our method to measure the privacy of PD-NMF and PD-SVD. Our conclusion is that although differential privacy works with a impressive adversary model, the fact that its learning model outcome model is so poorly understood, makes it a economically poor choice for privacy-preserving M-party distributed learning. As a result more research is needed towards building ways to measure privacy and build private algorithms.

3.1 One Bit Sum

In order to motivate our measures of privacy we study a task that is analytically tractable, presented in Figure 3.3.

Each individual i in our database samples their bit X_i iid with $P(X_i = 1) = p$.
Wlog let $0.5 \leq p < 1$ and let $0.5 \geq q = 1 - p > 0$.
The database with n individuals is denoted as X^n.
The function we are interested in is the sum $f(X^n) = \sum_{i=1}^{n} X_i$.

Fig. 3.3: One Bit Sum

Differential privacy typically supposes we have a concrete realization of X^n and we wish to output an $(\epsilon, \delta) - DP$ output $\hat{f}(X^n)$. Randomness comes from $\hat{f}$. However we want to output the exact f. As a result we take advantage of the fact that X^n is itself a random variable. If the adversary can not distinguish between neighboring probability distributions over X^n, then we can achieve a sense of privacy close to differential privacy. In the literature this has been called *distributional differential privacy* as well as *coupled worlds privacy* [99]. In this section we compare alternate ways to define what it means for two distributions to be indistinguishable. First we examine the ratio of densities method that comes from differential privacy, and then we compare it to the Kolmogorov-Smirnov statistic (Sections 3.2), as well as measures that we would like to address in future work, but haven't gotten working yet, such as the statistical distance.

Ratio of densities We study two random variables that result from having two neighboring databases. Let X_0^n be the random database where $X_1 = 0$ and X_1^n where $X_1 = 1$. The sum $f(X^n)$ is a binomial random variable $\mathcal{B}(n, p)$. Its probability mass function (pmf) is

$$P(f(X^n) = k) = \mathcal{B}(k; n, p) = \binom{n}{k} p^k q^{n-k}.$$

For $f(X_1^n)$, the pmf is

$$P(f(X_1^n) = k) = \mathcal{B}(k-1; n-1, p).$$

For $f(X_0^n)$ the pmf is

$$P(f(X_0^n) = k) = \mathcal{B}(k; n-1, p).$$

For differential privacy (defined in Figure 3.1 (p.g. 82)), we can set $\delta = 0$ and calculate ϵ using

$$\exp(\epsilon) = \max_k \max \left\{ r(n,k), \frac{1}{r(n,k)} \right\}, \tag{3.2}$$

where $r(n, k)$ is the *mass ratio*

$$r(n,k) \equiv \frac{\mathcal{B}(k; n-1, p)}{\mathcal{B}(k-1; n-1, p)} = \frac{p\,(k-n)}{k\,(p-1)} = \frac{p(n-k)}{q\,k}.$$

Since $f(X_1^n)$ can never be 0, and $f(X_0^n)$ can never be n, no finite ϵ satisfies Equation 3.2. This is why the $\delta = 0$ setting is not interesting in practice. The smallest δ we can achieve is

$$\begin{aligned}
\delta_{\min}(n) &= 1 - P(f(X_0^n) \geq 1) P(f(X_1) \leq n-1) \\
&= 1 - (1 - P(f(X_0^n) = 0))(1 - P(f(X_1) = n)) \\
&= 1 - \left(1 - q^{n-1}\right)\left(1 - p^{n-1}\right).
\end{aligned}$$

Since

$$\frac{d^2 r}{dk^2} = \frac{2np}{k^3\,q} > 0 \text{ and } \frac{d^2 \frac{1}{r}}{dk^2} = \frac{2n\,(q)}{p\,(n-k)^3} > 0$$

the maximum ratio r and $1/r$ will be at the boundary, i.e. either $k = 1$ or $k = n - 1$. Since

$$r(n,1) = \frac{p\,(n-1)}{q} \text{ and } r(n,n-1) = \frac{p}{(n-1)\,q}$$

using the assumption that $p \geq q$ we have that

$$r(n,1) \geq r(n,n-1), \frac{1}{r(n,1)}, \frac{1}{r(n,n-1)}.$$

As a result with probability at least $1 - \delta_{\min}(n)$

$$\exp(\epsilon) = r(n,1) = \frac{p(n-1)}{q}. \tag{3.3}$$

Therefore, if we are interested in the smallest δ, we must live with $\epsilon \in O(\log n)$. This is also intuitive, as n increases if we observe a sum of 1 there are n ways it could have resulted from X_0^n, but only one way it could have resulted from X_1^n: if all outcomes were 0. However, this is a undesirable result since it says that the worst case information leak increases as our database size increases. This also shows us that there is a lower limit on how small $\delta > 0$ we can make. Suppose we want δ to become small as n increases. This is desirable, because even if $\epsilon > 0$, we can still bound by how much our output can be perturbed. To do so, let's pick a reasonable δ that goes to zero, and see what happens to ϵ. We will also include constant delta for comparison, we consider

$$\begin{aligned}
\delta_{\text{const}} &= \text{1e-3} \\
\delta_{\text{lin}} &= \frac{1}{n} \\
\delta_{\text{quad}} &= \frac{1}{n^2} \\
\delta_{\text{cube}} &= \frac{1}{n^3}.
\end{aligned}$$

We must find $1 \leq k_l \leq k_u \leq n-1$ such that

$$\sum_{k=k_l}^{k_u} \mathcal{B}(k; n, p) \geq \delta \tag{3.4}$$

Since the actual binomial CDF is inconvenient to work with in this case, we make use of the central limit theorem. Let $\mu = np$ and $\sigma = \sqrt{np(1-p)}$, then as $n \to \infty$ the binomial CDF

F is well approximated by the normal CDF Φ as,

$$F(k; n, p) \approx \Phi\left(\frac{k-\mu}{\sigma}\right).$$

Then we set

$$k_l = \mu + \sigma\Phi^{-1}(\delta/2) \qquad k_u = \mu + \sigma\Phi^{-1}(1-\delta/2).$$

Note that since the Gaussian distribution is symmetric around 0, $\Phi^{-1}(x)$ is negative if $x < 0.5$.

We find

$$\begin{aligned}
\exp(\epsilon_{k_l,k_u}) &= \max\left\{r(n,k_l), r(n,k_u), \frac{1}{r(n,k_l)}, \frac{1}{r(n,k_u)}\right\} \\
\epsilon_{k_l,k_u} &= \max\left\{|\log r(n,k_l)|, |\log r(n,k_u)|\right\} \\
&= |\log r(n,k_l)|
\end{aligned}$$

We set $p = 0.5$ and plot the resulting ϵ as functions of each delta and n in Figure 3.4.

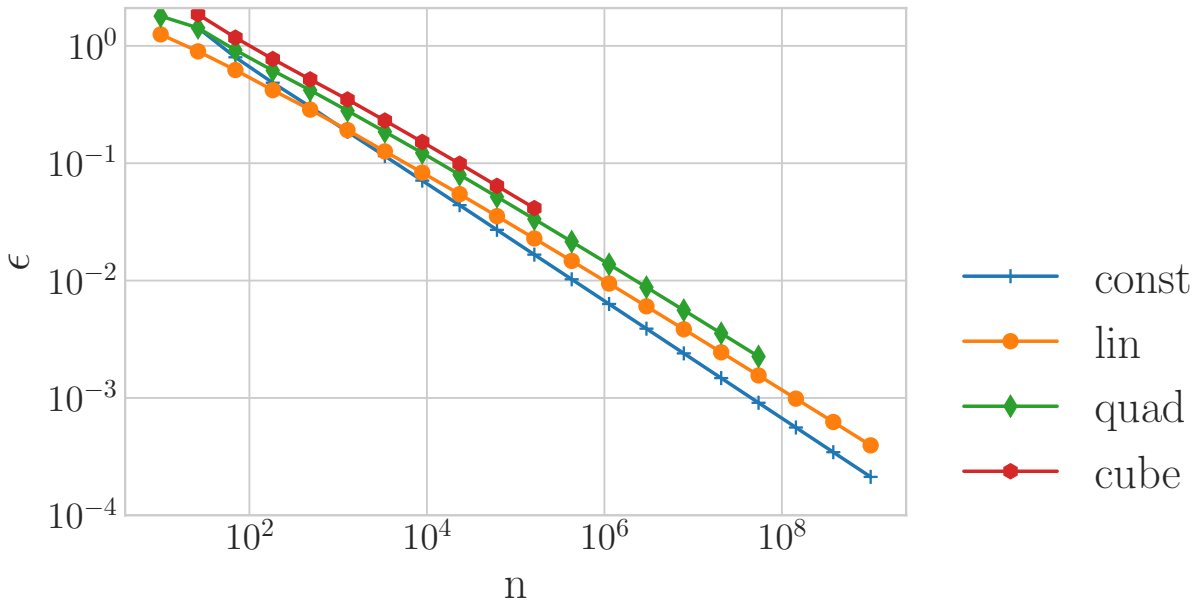


Fig. 3.4: ϵ as a Function of n for Several δ with $p = 0.5$

From the plot, we observe that even for very small delta, like $1/n^3$, we have

$$\epsilon \in O\left(\frac{1}{\sqrt{n}}\right).$$

As $|p - 0.5|$ increases, ϵ will be larger, but go to 0 at the same rate.

Kolmogorov-Smirnov Statistic The KS statistic is the maximum vertical distance between the cumulative density functions (CDFs) of two probability distributions. The CDF of the binomial mass function $\mathcal{B}$ is

$$F(k; n, p) \equiv P(\mathcal{B}(\cdot; n, p) \leq k) = \sum_{j=0}^{k} \mathcal{B}(j; n, p).$$

Let F_1 be the CDF of $f(X_1^n)$ and F_0 be the CDF of $f(X_0^n)$. The KS statistic is defined as

$$KS(F_0, F_1) \equiv \max_k |F_0(k; n-1, p) - F_1(k-1; n-1, p)|.$$

We plot an example with $n = 10$ for several values of p in Figure 3.5.

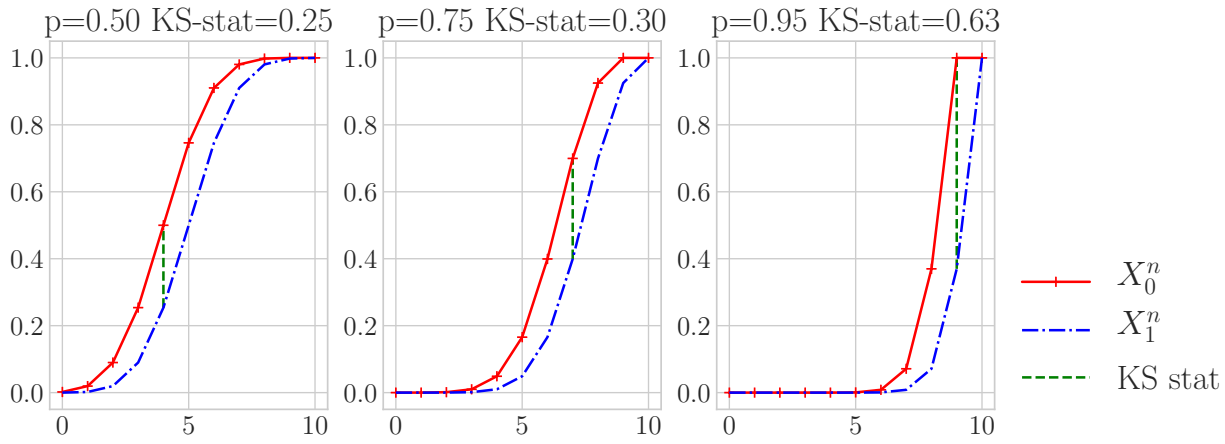


Fig. 3.5: KS-stat for the One Bit Sum with $n = 1$

Since $F(k) \geq F(k-1)$ we can drop the absolute value sign, as a result

$$\begin{aligned} KS(F_0, F_1) &= \max_k F_0(k; n-1, p) - F_1(k-1; n-1, p) \\ &= \max_k \sum_{j=0}^{k} \mathcal{B}(j; n-1, p) - \sum_{j=0}^{k-1} \mathcal{B}(j; n, p) \\ &= \max_k \mathcal{B}(k; n-1, p). \end{aligned}$$

For the binomial distribution the mode is

$$\arg\max_k \mathcal{B}(k; n-1, p) \in \{\lfloor np \rfloor, \lceil np \rceil - 1\}.$$

As $n \to \infty$, with $p = 0.5$ we have

$$\mathcal{B}(k; n-1, p) \to \sqrt{\frac{2}{\pi(n-1)}} 2^{n-1} \frac{1}{2^{n-1}} = \sqrt{\frac{2}{\pi(n-1)}} \in O\left(\frac{1}{\sqrt{n}}\right).$$

For $p > 0.5$ the order still holds, but larger p corresponds to a larger KS-stat. This means that for a fixed n, as p gets further from 0.5, it becomes easier to distinguish F_0 and F_1.

This is the same rate of convergence to zero as ϵ in DP, meaning if the underlying distributions are close enough to begin with, they will become indistinguishable at the same rate.

Integrated Squared Divergence & Statistical Distance As a preview of some directions that are interesting for future research (Section 3.5), we examine two additional measures of privacy. The integrated squared deviation (ISD) between two random variables X, Y (with densities f_X, f_Y) with respect to X is the expectation

$$\text{ISD}(Y; X) \equiv E_X[(X - Y)^2] = \int_{-\infty}^{\infty} (f_X(x) - f_Y(x))^2 f_X(x) dx. \tag{3.5}$$

For the random variables $S_1 = f(X_1^n)$ and $S_0 = f(X_0^n)$ defined previously we have

$$\text{ISD}(S_1; S_0) = \sum_{k=0}^{n} \left(\mathcal{B}(k; n-1, p) - \mathcal{B}(k-1; n-1, p)\right)^2 \mathcal{B}(k; n-1, p).$$

Meanwhile the statistical distance is defined as

$$\delta(S_1, S_0) = \sum_{k=0}^{n} \left| \mathcal{B}(k; n-1, p) - \mathcal{B}(k-1; n-1, p) \right|.$$

The ISD is something we know how to estimate [100] whereas the statistical distance is (something we would like to figure out how to estimate, since it is) related to the maximum probability that an adversary could distinguish between X_0^n and X_1^n by observing the sum of the rows [101].

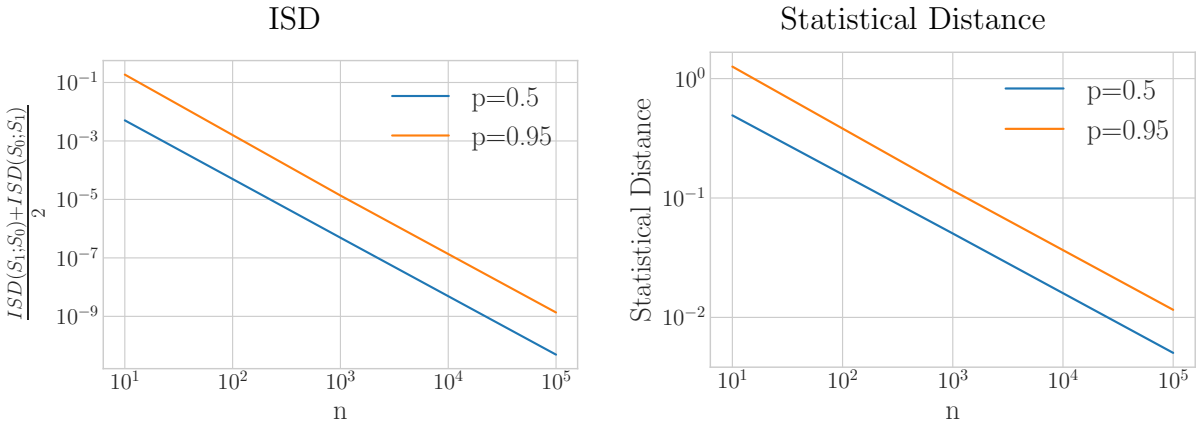


Fig. 3.6: ISD & Statistical Distance

3.2 Kolmogorov-Smirnov Distributional Privacy

In the context of protecting privacy in a multiparty computation, it is appropriate to study the privacy of individual rows of each party's database [102]. Informally, Distributional Differential Privacy (DDP) aims to achieve similar guarantees as DP but without necessarily using external noise in the mechanism; instead uncertainty about the data is used to conceal the relevant information. More concretely, in Distributional Differential Privacy, e.g. [99], [103], a mechanism $\mathcal{A}$ is said to be "distributionally differentially private" for dataset X drawn from sampling distribution $\mathcal{U}$, and all individual rows $x \in X$ if there exists some mechanism $\mathcal{B}$ such that

$$P[\mathcal{A}(X)] \approx_{\epsilon,\delta} P[\mathcal{B}(X \setminus \{x\})].$$

For probability density functions f, g over sample space Ω with sigma-algebra $\mathcal{S} \subseteq 2^{\Omega}$, $f \approx_{\epsilon,\delta} g$ if

$$\begin{aligned} \forall \omega \in \mathcal{S}\, f(\omega) &\leq e^{\epsilon} g(\omega) + \delta \\ \text{and } g(\omega) &\leq e^{\epsilon} f(\omega) + \delta. \end{aligned} \tag{3.6}$$

In order to better understand this definition in our context it is useful to define the concept of an *observable*.

Observable. Suppose a distributed mechanism $\mathcal{A}$ is executed among several parties. The party of interest uses database a X, while the concatenation of the remaining parties' databases is Y. Then $\mathcal{O}_{\mathcal{A},X,Y}$ is the set of all objects that a coalition of adversaries observes from the communication transcript of $\mathcal{A}$.

Therefore a mechanism $\mathcal{A}$ can be interpreted to satisfy distributional privacy for document x if it can be simulated by a mechanism $\mathcal{B}$. Where $\mathcal{B}$ must operate without access to x but still emit *all* observables $\mathcal{O}_{\mathcal{A},X,Y}$ with a comparable probability distribution to $\mathcal{A}$.

The easiest way to satisfy such a strict privacy requirement for all observables is to build mechanism $\mathcal{A}$ using Differential Privacy [89], [104], [105]. However the loss in learning accuracy is difficult to quantify. Theoretical results in [106] give estimates for the number of samples needed to match centralized learning error plus/minus some tolerance for given ϵ, δ for simple mechanisms. But what happens to the learning outcome when an algorithm is constructed out of the composition of these simple mechanisms needs to be examined on case-by-case basis, and things get ugly as the tolerance goes to 0. More concretely, in one-party Differentially Private NMF [107] the authors observe a 19% loss in learning quality (RMSE in their case) when strict DP is satisfied even for $\epsilon = 0.25$. In order to improve the learning outcome, they propose violating privacy guarantees for atypical users. This lets them use a smaller ϵ and lets them add less noise to the mechanism, similar to how we increased δ for the one bit sum example.

Most differential privacy efforts have been focused on the one-party setting. Here a curator has access to the data and is trying to answer queries in a differentially private manner. However in our setting there are M parties co-operating to answer 'a query.' This task is much more challenging is now the parties are also interested in protecting intermediate results from one another. As a result, due to a possible difference attack on observables at

iteration i and $i+1$ it is necessary for each party to add noise to all observables they emit for every iteration [90], [108]. In Section 3.3 we will examine the potential effect of adding noise at every iteration for NMF and SVD.

Furthermore, because of the "curse of dimensionality," privacy as defined by Equation 3.6 is often not computationally tractable. The fundamental issue is that the volume of the sample space from which observables $\mathcal{O}$ are drawn grows exponentially in d. A standard result in probability density estimation states that expected square error for the probability density decreases at a rate of $O(n^{\frac{1}{d}})$ [100]. This means that an adversary with polynomial computation power cannot reliably identify an outcome $\omega \in \mathcal{S}$ as indicative of the absence or presence of a particular document x, because this adversary doesn't have an accurate estimate for $P[\omega]$. Even if the adversary could estimate $P[\omega]$, they would still face the issue that $\mathcal{S}$ is exponentially large, which makes it difficult to find non-private ω in this space.

Therefore we make the assumption that economically-constrained adversaries will reason about distributions of statistics of observables, rather than distributions of the observables themselves. Let $\sigma\colon \mathcal{O} \times x \to \mathbf{R}$ be a document-dependent statistic. Within the context of a statistic σ and a document x, let f be the distribution of σ with observables $\mathcal{O}_{\mathcal{A},X,Y}$ and g by the distribution of σ with observables $\mathcal{O}_{\mathcal{B},X\setminus\{x\},Y}$ (i.e. those emitted by a simulator $\mathcal{B}$ which doesn't have access to x). Equation 3.6 proposes one way of defining the similarity between PDFs f, and g. However Equation 3.6 is both difficult to satisfy, and unnecessarily strict for protecting against polynomial-compute adversaries. Since the spirit of distributional privacy is maintained under other notions of similarity between distributions (however useful properties like composition are lost) we look elsewhere. In particular, instead of attempting to measure the similarity between PDFs f and g we propose measuring similarity between the corresponding CDFs F and G using the well-known Kolmogorov-Smirnov statistic. It is defined for empirical CDFs F, G over an interval $[a, b]$ as

$$\mathrm{KS}(F, G) \equiv \sup_{x \in [a,b]} |F(x) - G(x)|.$$

This statistic can then be used in the Kolmogorov-Smirnov 2-sample hypothesis test, which outputs a p-value that represents the probability that a statistic at least as extreme as the one observed would be drawn if F and G were sampled from the same underlying distribution. Now we are ready to define our version of distributional privacy.

Algorithm 11 Measuring π-KSDP.

Require Mechanism f, set of statistics S, database X, document of interest x, number of samples per document t.
Output: Minimum p-value for all statistics.
Determine what the *observables* of $f(A)$ are.
Generate database sub-samples $\{X'_i\}_{i=1}^t$, $X'_i \subseteq X$
Evaluate the mechanism $f(X'_i \cup \{x\})$ for all i.
Generate database sub-samples $\{Y_i\}_{i=1}^t$, $Y_i \subseteq X \setminus \{x\}$
Evaluate the mechanism $f(Y_i)$ for all i.
Set $\pi \leftarrow 1$
for each $s \in S$ **do**:
    $p \leftarrow$ KS 2 sample test on ECDF($\{s(f(X_i), x)\}_{i=1}^t$) and ECDF($\{s(f(Y_i), x)\}_{i=1}^t$).
    $\pi \leftarrow \min(p, \pi)$
end for
return π

KS Distributional Privacy (π-KSDP). A mechanism $\mathcal{A}$ satisfies $\pi-$KS Distributional Privacy if there exists a simulator mechanism $\mathcal{B}$ such that for all statistics $\sigma \in \{\sigma_1, \sigma_2, \ldots\}$, and for all documents $x \in X$, the KS 2-sample test run on ECDF($\sigma(\mathcal{O}_{\mathcal{A},X,Y}, x)$) and ECDF($\sigma(\mathcal{O}_{\mathcal{B},X\setminus\{x\},Y}, x)$) returns $p \geq \pi$.

In words, KSDP means that the observables generated by a simulator $\mathcal{B}$ that doesn't have access to a document x cannot be statistically distinguished from the actual algorithm $\mathcal{A}$ running on the database with x included.

The KS-test's p-value is simply used as a measure of distance between two distributions which takes number of samples into account. A high p-value means that we cannot reject that the two ECDFs are the result of sampling from the same underlying distribution. It doesn't mean that we can conclude they come from the same distribution. Although hypothesis tests with the reverse null/alternate hypothesis have been studied [109], computational efficiency and interpretability are significant challenges.

The definition of π-KSDP leads to a method for measuring the privacy of a distributed mechanism, see Algorithm 11. A weakness of π-KSDP is that it doesn't guarantee future-proof privacy against new statistics and auxiliary information. Rather it tests whether a given set of statistics is discriminative enough to break distributional differential privacy. An advantage of Algorithm 11 is that samples can be generated, stored, and re-used to test many different statistics quickly. Furthermore, Algorithm 11 is suitable as a defensive algorithm

for identifying sensitive documents x, which can be excluded from distributed learning if need be. Effective simulators $\mathcal{B}$ are to run $\mathcal{A}$ but either replace x by a random document, or sample n random documents to create an entirely new database. In Section 3.4 we use the latter simulator and Algorithm 11 to give evidence that PD-NMF satisfies π-KSDP.

3.3 Learning Outcomes

We study the scenario where M parties want to learn on their shared data in order to improve the outcome of their local models. We know that for all forms of statistical learning, with more data we will be better able to capture the underlying distribution, and therefore learn better models. In this chapter, what we are interested in exploring the following questions

1. What improvements do we see for applications of NMF, and SVD?
2. What is the difference in terms of learning uplift between small parties and large parties (as a function of how big their local databases $X^{(m)}$ are).
3. What effect do parameters within our applications have on the learning outcome?
4. How well do differentially private versions of NMF and SVD adapted to the M party setting fair?

3.3.1 NMF

We can measure % improvement compared to using local models for both topic modeling and recommender systems. For topic modeling we use the Enron dataset, and measure the percentage decrease in Frobenius reconstruction error *on a held-out local test set* using the global topics compared to the best local topics. For recommender systems we use the MovieLens-1M dataset and measure the percent decrease in RMSE each party gets on a held-out portion of their local users' ratings by using the global model instead of their local model. Figure 3.7 (a)-(b) shows that parties of all sizes have something to gain from cooperative distributed learning, but small parties (with the least local data) stand to gain the most. We also observe that simpler models, with smaller k, see lesser uplift from using more data compared to more complex model, with larger k.

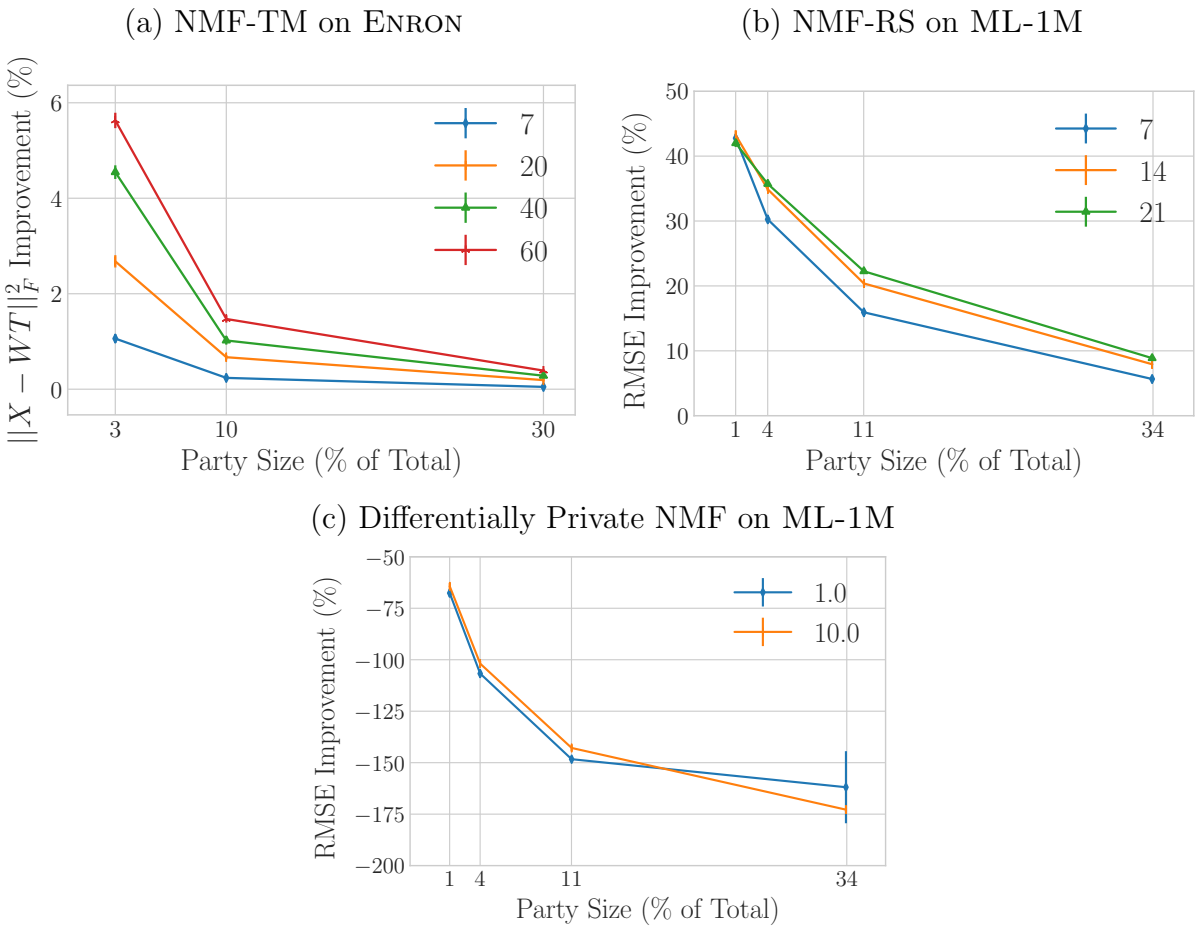


Fig. 3.7: Mean Percent Improvement: for Various k in (a), (b); for Various ϵ with $\delta = 0.01$, k=7 in (c)

Next, we examine what happens to the learning outcome if we add noise at every iteration to make PD-NMF (ϵ, δ)-differentially private. To do so we use the Gaussian Mechanism, which requires iid noise distributed as $\mathcal{N}(0, v_{\epsilon,\delta,\Delta_2 f})$ is added to every observable. The standard deviation of the noise is roughly proportional to $1/\epsilon$. Even though adding noise at every iteration isn't the only way to achieve differential privacy, without a trusted third party present, adding noise at every iteration seems inevitable [90], [108]. Even though we can make use of SecureSum to allow each party to add $1/M$th of the noise that would be required without it, the learning outcome is still disastrous, depicted in Figure 3.7 (c). The RMSE is only slightly better than random guessing.

This adverse effect on learning quality is actually guaranteed by differential privacy. To see why, suppose we had a algorithm that claims it is differentially private and converges to the

exact same solution as the centralized algorithm. Then an arbitrarily powerful adversary could try all (combinatorially many) possible input databases, and determine which one yields the same exact minimum as the DP minimizer, compromising privacy of the entire database.

The necessity of this 'learning gap' means that one has to analyze the decreased quality on a per-application basis. In our setting this problem is particularly important: if a recommender system produced by a DP mechanism is worse than a system one party could construct from its own data, then this party gained nothing and lost some privacy by participating in distributed learning.

3.3.2 SVD

For SVD we examine performance on the low-rank approximation (LRA) task.

For a matrix X, $XV_kV_k^T$ is its projection onto the rank k subspace spanned by its top k right singular vectors. We measure the reconstruction error as

$$||X - XV_kV_k^T||_F,$$

and compare the reconstruction error each party gets on a held-out X_{test} using V_k computed from its local $X^{(m)}$ compared to the V_k obtained from the global X using PD-SVD. We evaluate this on the, by now familiar, 20NG and Enron datasets.

In Figure 3.8 we compare PD-SVD to a particular Differentially Private SVD mechanism [110]. Unlike the NMF section, we report absolute changes in score rather than 'percentage improvements.' Since LRA is scored in terms of an 'error', negative changes are good. For the LRA task, results are reminiscent of NMF. Differential Privacy fails unless we set epsilon so high that the DP bound becomes meaningless; recall that DP bounds outcomes by $\exp(\epsilon)$.

3.4 Privacy Measurements for PD-NMF and PD-SVD

In this Section we evaluate PD-NMF and PD-SVD using the Kolmogorov-Smirnov Distributional-Privacy (KSDP) measure of Section 3.2.

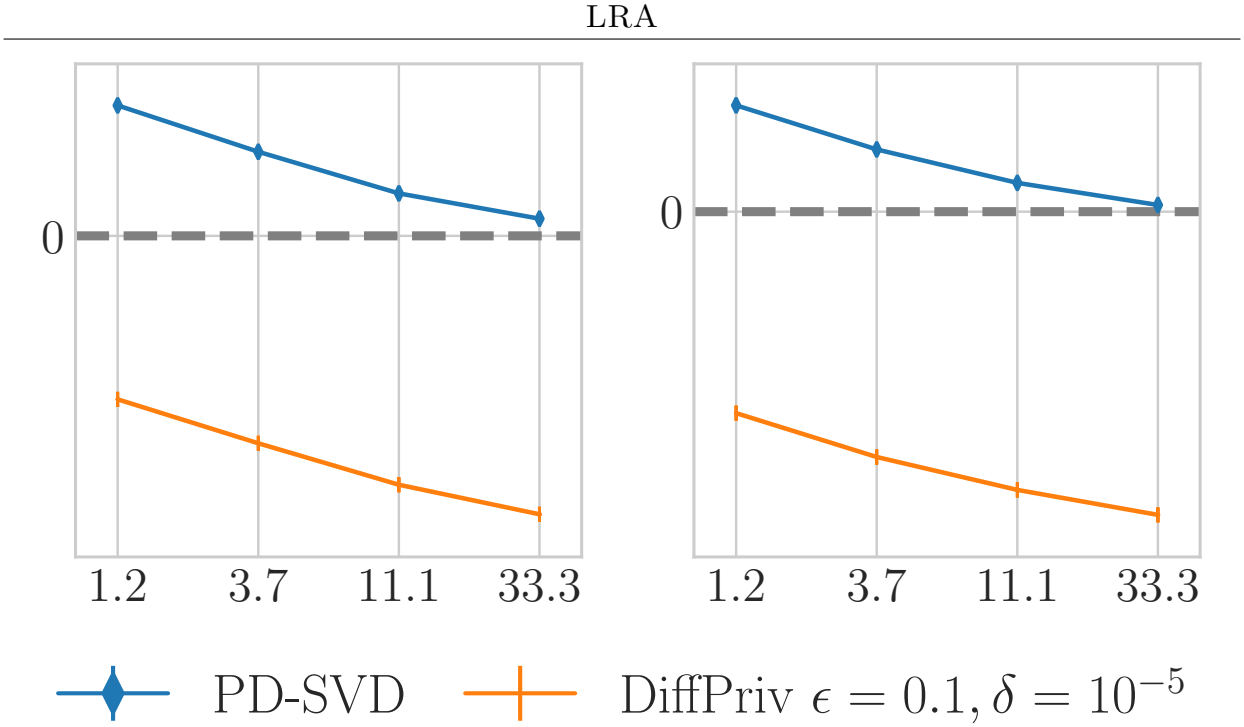


Fig. 3.8: Mean Learning Uplift for PD-SVD and Differentially Private SVD

Recall that to use KSDP we must first identify the observables seen by an adversary, and then form functions that map a (observable, document x) vector to a scalar. We call such functions *statistics*.

Since PD-NMF and PD-SVD are iterative algorithms, the adversary actually observes a sequence of observables. However we will only study the final output. We believe this is a reasonable simplification to make for several resaons,

1. A economically rational adversary will not be able to devote resources to study extremely high-dimensional spaces. By focusing on one iteration, the dimensionality of our space is reduced (but still too high).

2. If using the random initialization for NMF (by default we use random initialization for SVD), the first few observables will be functions of the random initialization.

3. As NMF and SVD converge, the observables emitted have to become more autocorrelated. This means that the adversary effectively sees fewer independent samples in the later iterations [111], and doesn't get much more information beyond the final

output.

If this assumption is unsatisfactory, it is possible to upper bound the amount of information contained over multiple iterations. For SVD, an upper bound on the information contained in the iterations is the information in the matrix $X^T X$. This is because having $X^T X$ we can perfectly simulate PD-SVD. In order to do something similar for NMF we would have to reformulate the problem as

$$\min_{w,T} ||X^T X - T^T \text{diag}(w) T||_F^2,$$

and use a modified variant of RRI-NMF, e.g. Chapter 7 of [7].

Having identified observables we must, build document-specific statistics for both NMF and SVD. The reason we construct document specific statistics is because we presume the adversary knows what the document x they are trying to identify is. As a result they are allowed to use this information in trying to distinguish between observables emitted by the mechanism running with document x included in its database, and the simulator that doesn't have x. An additional assumption we make is that the adversary knows how many documents we have in total.

3.4.1 PD-NMF

For PD-NMF we examine the observables emitted for topic modeling and recommender systems.

We assume there are $M = 2$ parties, and the adversary knows exactly what the victims observables are. In practice, unless colluding with everyone except for the victim, the adversary wouldn't be able to identify a particular party's observables; they would be hidden in the SecureSum. The parameters we vary are f, the size of each party, between 0.5% to 10% of all documents in the given dataset, and the model dimension $k \in \{7, 14\}$. We examine this on the Enron dataset for topic modeling and the MovieLens-1M dataset for recommender systems.

To measure privacy of PD-NMF, we use Algorithm 11. The simulator we use runs PD-NMF with the database X that contains a given document x replaced by a completely random

database X' for which the only requirement is that $x \notin X'$. The most-discriminative (out of 7 that we tested) document-specific statistic we found is computed as follows: first, find a particular document's coefficients with respect to topics T $\mathbf{a} \leftarrow \arg\min_{\mathbf{u} \in [0,1]^{1 \times k}} ||\mathbf{x} - \mathbf{u}T||_F^2$, and then measure the weighted inner product of these coefficients and the observed topic weights (from the denominator in PD-NMF-ITER) $\sigma \equiv \mathbf{a} \cdot \mathbf{w}^2$. We call this statistic the *document-weighted topic weights*. We randomly picked 50 documents of interest x (in practice this would have to be done for all documents), and for each we generated 200 samples of PD-NMF running on a random database with x, and 200 of it running on a (otherwise iid) random database without x. The former are samples of the original mechanism $\mathcal{A}$, while the latter correspond to the simulator $\mathcal{B}(X \setminus \{x\})$.

The results are summarized in Figure 3.9. A larger p-value corresponds to more privacy. Similar to the one-bit sum, we observe the p-value generally increase as each party's database becomes larger. Surprisingly, larger k seems to also lead to a larger p-value. We expected that a more complex model would reveal more information about the underlying data. When we look at other statistics (that are less informative than the document-weighted topic weights), we observe that indeed higher k leads to less privacy. So we don't have a conclusive result on the effect of k. In the recommender system setting we observe larger p values for this statistic. It's probably not a good statistic for that setting, it just happens to be the best we've tested. However in the topic modeling setting we observe some low p-values, especially for the small party sizes.

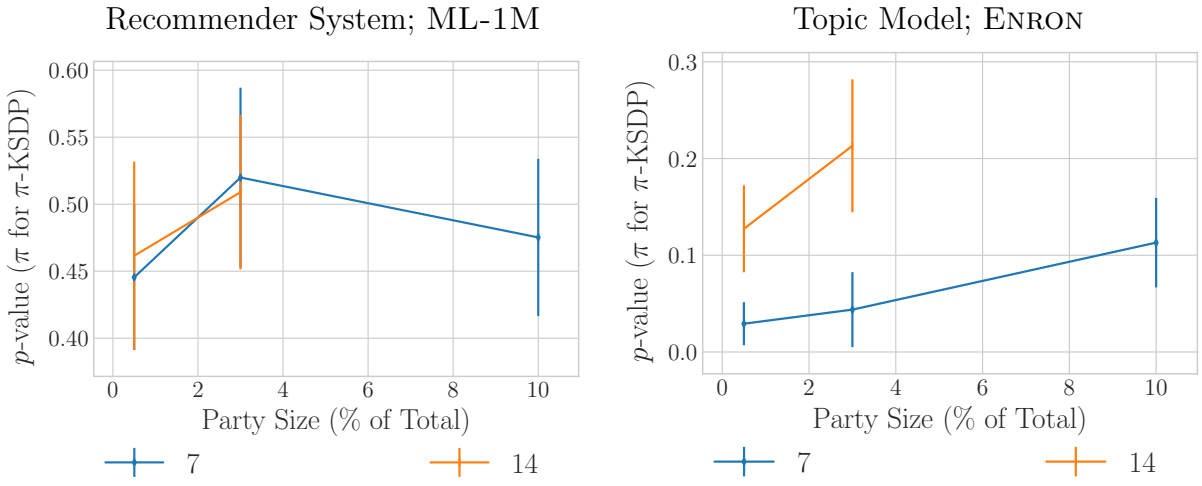


Fig. 3.9: KSDP for Document-Weighted Topic Weights Statistic

Overall there is a interesting trade-off between privacy and learning outcome. While smaller parties have more to gain in terms of model quality (Section 3.3) they also risk losing more privacy. What is nice about embedding all communication of PD-NMF in the secure SecureSum protocol is that the effective party size that a coalition of adversaries sees is the sum of the remaining parties. Our experiments show the p-values an attacker would get if they could directly identify the observables that belong to their victim. In a practical multi-party application of PD-NMF a coalition of A adversaries wouldn't be able to attribute the remaining $M - A$ parties' observables to a particular party.

3.4.2 PD-SVD

For the SVD setting, the setup is identical to the NMF setting except for datasets and corresponding machine learning problem. We study LRA on Enron and PCR on MillionSongs. To measure the privacy of PD-SVD we use KSDP.

We compare PD-SVD to prior secure multiparty SVD algorithms that reveal Σ. In [112], [113] the algorithms essentially reduce to sharing $X^TX = V\Sigma^2V^T$ securely, and as a result each party learns Σ. In [114] a method is proposed based on the QR decomposition that allows parties to only share Σ if they choose to, however it is only developed for $M = 2$ parties. In [115], the SVD is solved using a method similar to ours, but once again Σ is revealed. We show that we have a higher p-value, corresponding to more privacy thanks to hiding Σ.

We consider 200 documents x (in practice a party would run this for all their documents, or the ones they are worried about), and generate 50 sample databases that have x and 50 (otherwise independent) databases that don't have x. We then run PD-SVD as well as Σ-revealing SVD on each of the sample databases, and collect all the observables that could be inferred from the communication transcript. Although PD-SVD is an iterative algorithm, we only study the observables at the end of all iterations, the final V_k. We make this simplification for several reasons: (1) a polynomially-bound adversary can not easily work with time-series of observables due to the curse of dimensionality (2) these observables are highly auto-correlated, meaning that observing V_k over τ iterations doesn't give one much more information than observing V_k once (3) the observables over multiple iterations can be simulated by working from X^TX, so that the privacy of statistics computed from

X^TX is a lower bound for statistics computed from multiple iterations' worth of V_ks. Since $X^TX = V\Sigma^2V^T$, this lower bound comes from evaluating the Σ-revealing SVD.

Next, since the adversary is limited by the curse of dimensionality, they must build statistics of the observables. We consider an adversary who tries to be sophisticated. First, they observe V_k as an output from the protocol. Then for PD-SVD they attempt to estimate the global Σ_k^2 from the knowledge of how many documents the victim has n_v. They know their own $\Sigma_k^{(A)}$, and estimate $\hat{\Sigma}_k^2 = \frac{n_v+n_a}{n_a}(\Sigma_k^{(A)})^2$. From this the adversary estimates $S = X^TX$ as $\hat{S} = V_k\hat{\Sigma}_k^2V_k^T$. For Σ-revealing SVD the adversary simply uses the revealed Σ rather than this estimate. Since the adversary is building a statistic for a particular document x, they can consider the relationship between $\hat{S}$ and the outer-product xx^T. We derived one family of statistics from the weighted correlation matrix $W = |\hat{S} \circ xx^T|$, where $\circ$ and $|\cdot|$ represent elementwise product and absolute value, respectively. To reduce W to a scalar we consider taking the max entry, as well as sum, and sum of squares. We also consider a similar family of statistics originating from $D = \left|\frac{\hat{S}}{||\hat{S}||_F} - \frac{xx^T}{||x||}\right|$.

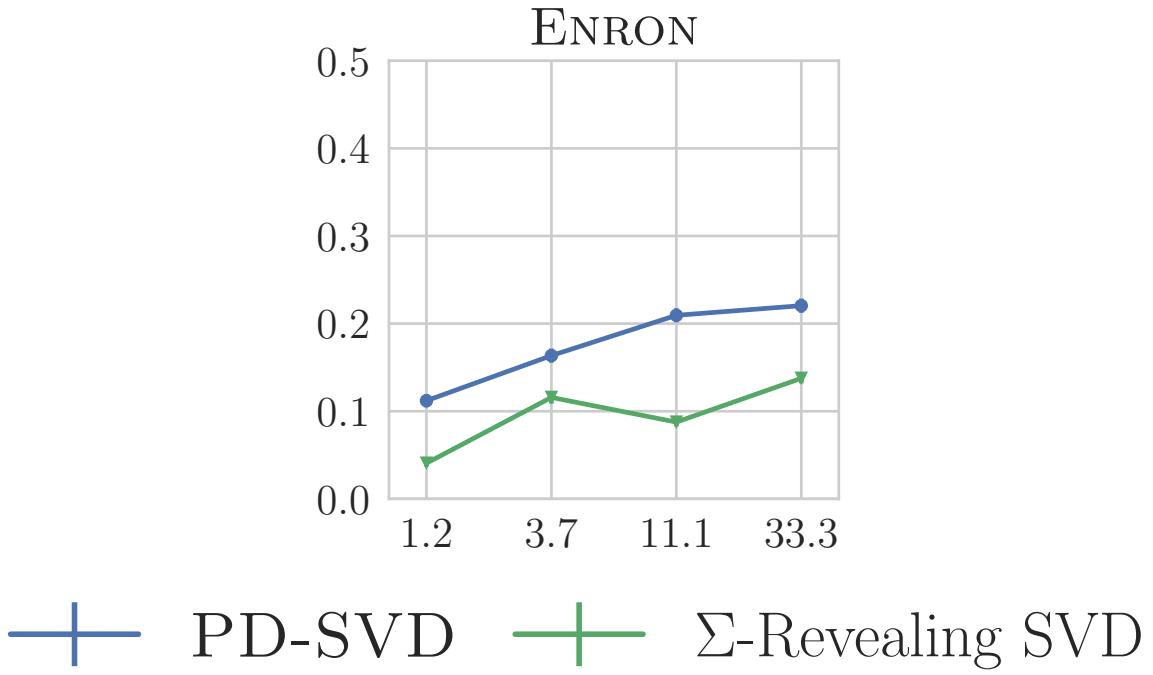


Fig. 3.10: Mean $\pm\sigma_u$ Minimum p-value as Measured by KS$\mathcal{D}$P, y-axis, as a Function of Party Size, x-axis

In Figure 3.10 we present the mean, over documents x, minimum p-value resulting from both

families of statistics. For the LRA task on Enron, we see that small parties are at highest risk for privacy breach. However, since in PD-SVD observables are communicated through SecureSum and NormedSecureSum mechanisms a coalition of A colluding parties would only be able to observe the sum of the remaining $M - A$ non-colluding parties' observables. As a result, a party's 'effective party size' would be the sum of the non-colluding parties' sizes. We observe that PD-SVD has a significantly higher p-value than the Σ-revealing SVDs, except for the 33.3% party size on Enron. This confirms the intuitive idea that Σ contains informative information. However, since even the Σ-revealing SVD has p-values that are much larger than 0.05, so running PD-SVD with SecureSum may be acceptable for many applications.

We also experimented with different values of k $\{7, 14, 21, 42\}$ and the number of bits $\{10, 20, 31, 60\}$ we use in the fixed point representation. The learning uplift trends are the same, and we still have privacy. However large k and low bits are dangerous, since vectors V_k that correspond to small singular values may need more bits since their entries will be small before orthonormalization.

Overall, PD-SVD enables small parties to enjoy large learning uplifts, while hiding their individual contributions in a sum, which increases their privacy.

3.5 Directions for Future Research

We believe there are essentially two directions for improving the results without leaving the paradigm of noiseless privacy.

The first direction is in providing empirical measures more meaningful than the KS-statistic. Note that when measuring KSDP we are creating statistics for the adversary, and then measuring how well those statistics can distinguish between outputs that result from the document being part of the database x (call this event '+'), and when documents are not part of the database (corresponding to the event '-'). This is essentially a classification problem. And being worried about clever adversaries, we are concerned with how well the best possible classifier could do. We know that the optimal classifier is the Bayes-optimal classifier. Upon observing a statistic s it answers '+' iff $P(s|+)P(+) > P(s|-)P(-)$. We can use k-nearest neighbors to get an asymptotically unbiased estimate of the adversary's lowest error rate.

Although in trying to build practical measures, we want to be able to say something about the convergence rate of the estimate. After all the purpose of algorithms like Algorithm 11 (p.g. 95) is for a party to run locally in order to see which documents in its database are too risky to include in a distributed computation. So knowing a concrete sample number is needed. Also working from the Bayes-optimal classifier, the cryptography community has defined the maximum possible advantage an adversary can have as the *statistical distance* [101], defined as

$$\delta(+,-) = \sum_{s \in \Omega(+) \cup \Omega(-)} |P(s|+) - P(s|-)|.$$

Meanwhile we know how to estimate

$$\int_s ds P(s) \left(P(s|+) - P(s|-)\right)^2$$

using standard density estimation techniques [100]. Can we do better than using the inequality

$$||x|| \leq \sqrt{d}||x||_2$$

to relate these? Can existing results for estimating the total-variation distance [116] be used? Instead of trying to measure distances should we try to estimate the ratio between the probability densities instead [117]?

A second direction is in providing more powerful statistics or families of statistics. Since a lot of work is being published on automatically differentiable programming, it is seems plausible to reuse architectures that have been developed for classification in order to generate statistics. If a particular architecture is suitable for classifying word distributions, maybe that architecture can output the probability that a document was in the database that generated a particular set of topics. Recently adversarial auto-encoders have started gaining traction as methods for measuring privacy [118]. The issue seems to be that it's difficult to work with a meaningful definition of what an optimal adversary's error rate is.

Overall it seems that differential privacy doesn't work for many situations in practice. As a result it's important to work on the tooling for those situations, so that if privacy ever becomes popular, the tools will be ready.

REFERENCES

[1] S. Dasgupta, "The hardness of k-means clustering," UCSD, San Diego, CA, USA, Tech. Rep. 0916, 2008.

[2] D. Arthur and S. Vassilvitskii, "K-means++: The advantages of careful seeding," in *Proc. SODA*, 2007, pp. 1027–1035.

[3] S. Deerwester, S. T. Dumais, G. W. Furnas, T. K. Landauer, and R. Harshman, "Indexing by latent semantic analysis," *J. Of The Amer. Soc. For Inform. Sci.*, vol. 41, no. 6, pp. 391–407, 1990.

[4] D. D. Lee and H. S. Seung, "Algorithms for non-negative matrix factorization," in *Proc. NIPS*, 2001, pp. 42–50.

[5] ——, "Learning the parts of objects by non-negative matrix factorization," *Nature*, vol. 401, no. 6755, pp. 788–791, 1999.

[6] R. Graham and A. M. Oberman, "Approximate convex hulls," *arXiv:1703.01350*, 2017.

[7] N.-D. Ho, "Nonnegative matrix factorization algorithms and applications," PhD thesis, Comp. Sci. Dept, Université catholique de Louvain, Louvain, Belgium, 2008.

[8] C.-J. Lin, "Projected gradient methods for nonnegative matrix factorization," *Neural Computation*, vol. 19, no. 10, pp. 2756–2779, 2007.

[9] A. Cichocki, R. Zdunek, and S.-i. Amari, "Hierarchical als algorithms for nonnegative matrix and 3d tensor factorization," in *Proc. LVA-ICA*, 2007, pp. 136–144.

[10] L. Condat, "Fast projection onto the simplex and the ℓ_1 ball," *Math. Programming*, vol. 158, no. 1-2, pp. 575–585, 2016.

[11] C. Boutsidis and E. Gallopoulos, "Svd based initialization: A head start for nonnegative matrix factorization," *Pattern Recognition*, vol. 41, no. 4, pp. 1350–1362, 2008.

[12] T. Hofmann, "Probabilistic latent semantic analysis," in *Proc. UAI*, 1999, pp. 794–804.

[13] C. Ding, T. Li, and W. Peng, "On the equivalence between non-negative matrix factorization and probabilistic latent semantic indexing," *Computational Statist. & Data Anal.*, vol. 52, no. 8, pp. 3913–3927, 2008.

[14] C. M. Bishop, *Pattern Recognition and Machine Learning*. Secaucus, NJ, USA: Springer, 2006.

[15] L. Hong, "A Tutorial on Probabilistic Latent Semantic Analysis," *arXiv:1212.3900*, 2012.

[16] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent dirichlet allocation," *J. Of Mach. Learning Res.*, vol. 3, pp. 993–1022, 2003.

[17] T. L. Griffiths and M. Steyvers, "Finding scientific topics," in *Proc. PNAS*, 2004, pp. 602–612.

[18] G. Heinrich. (2005). Parameter estimation for text analysis. Accessed on: Mar 1 2014, [Online]. Available: http://www.arbylon.net/publications/text-est.pdf.

[19] H. M. Wallach, "Structured topic models for language," PhD thesis, Comp. Sci. Dept, University of Cambridge, Combridge, UK, 2008.

[20] T. Griffiths, "Gibbs sampling in the generative model of latent dirichlet allocation," *unpublished note*, 2002.

[21] A. Asuncion, M. Welling, P. Smyth, and Y. W. Teh, "On smoothing and inference for topic models," in *Proc. UAI*, 2009, pp. 1108–1115.

[22] C. Chemudugunta, P. Smyth, and M. Steyvers, "Combining concept hierarchies and statistical topic models," in *Proc. CIKM*, 2008, pp. 1469–1470.

[23] M. Steyvers, "Combining feature norms and text data with topic models," *Acta Psychologica*, vol. 133, no. 3, pp. 234–243, 2010.

[24] J. Chuang, S. Gupta, C. Manning, and J. Heer, "Topic model diagnostics: Assessing domain relevance via topical alignment," in *Proc. ICML*, 2013, pp. 408–415.

[25] D. M. Blei and J. D. Lafferty, "Correlated topic models," in *Proc. NIPS*, 2005, pp. 268–278.

[26] I. Titov and R. McDonald, "Modeling online reviews with multi-grain topic models," in *Proc. WWW*, 2008, pp. 111–120.

[27] Y. W. Teh, M. I. Jordan, M. J. Beal, and D. M. Blei, "Sharing clusters among related groups: Hierarchical dirichlet processes," in *Proc. NIPS*, 2005, pp. 1448–1456.

[28] F. Murtagh and P. Legendre, "Ward's hierarchical clustering method: Clustering criterion and agglomerative algorithm," *arXiv:1111.6285*, 2011.

[29] M. Rosen-Zvi, T. Griffiths, M. Steyvers, and P. Smyth, "The author-topic model for authors and documents," in *Proc. UAI*, 2004, pp. 1412–1420.

[30] D. M. Blei and J. D. Lafferty, "Dynamic topic models," in *Proc. ICML*, 2006, pp. 113–120.

[31] H. Larochelle and S. Lauly, "A neural autoregressive topic model," in *Proc. NIPS*, 2012, pp. 1067–1077.

[32] Z. Cao, S. Li, Y. Liu, W. Li, and H. Ji, "A novel neural topic model and its supervised extension," in *Proc. AAAI*, 2015, pp. 1779–1788.

[33] A. M. Dai, C. Olah, and Q. V. Le, "Document embedding with paragraph vectors," *arXiv:1507.07998*, 2015.

[34] S. A. Vavasis, "On the complexity of nonnegative matrix factorization," *Siam J. On Optimization*, vol. 20, no. 3, pp. 1364–1377, 2009.

[35] S.-H. Cha, "Comprehensive survey on distance/similarity measures between probability density functions," *City*, vol. 1, no. 2, p. 1, 2007.

[36] H. M. Wallach, I. Murray, R. Salakhutdinov, and D. Mimno, "Evaluation methods for topic models," in *Proc. ICML*, 2009, pp. 1105–1112.

[37] A. K. Jain and R. C. Dubes, *Algorithms for Clustering Data*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1988.

[38] N. X. Vinh, J. Epps, and J. Bailey, "Information theoretic measures for clusterings comparison: Is a correction for chance necessary?" In *Proc. ICML*, 2009, pp. 1073–1080.

[39] M. Meilă, "Comparing clusterings by the variation of information," in *Proc. COLT/Kernel*, 2003, pp. 535–543.

[40] D. Mimno, H. M. Wallach, E. Talley, M. Leenders, and A. McCallum, "Optimizing semantic coherence in topic models," in *Proc. EMNLP*, 2011, pp. 736–744.

[41] D. Newman, J. H. Lau, K. Grieser, and T. Baldwin, "Automatic evaluation of topic coherence," in *Proc. NAACL*, 2010, pp. 100–108.

[42] M. Röder, A. Both, and A. Hinneburg, "Exploring the space of topic coherence measures," in *Proc. WSDM*, 2015, pp. 1971–1980.

[43] D. O'Callaghan, D. Greene, J. Carthy, and P. Cunningham, "An analysis of the coherence of descriptors in topic modeling," *Expert Syst. With Appl.*, vol. 42, no. 13, pp. 5645–5657, 2015.

[44] L. AlSumait, D. Barbará, J. Gentle, and C. Domeniconi, "Topic significance ranking of lda generative models," in *Proc. ECML-PKDD*, 2009.

[45] Y. Jiang, X. Li, and W. Meng, "Discword: Learning discriminative topics," in *Proc. WI/IAT*, 2014, pp. 75–82.

[46] D. J. C. MacKay, *Information Theory, Inference & Learning Algorithms*. New York, NY, USA: Cambridge University Press, 2002.

[47] C. Sievert and K. E. Shirley, "Ldavis: A method for visualizing and interpreting topics," in *Proc. ACL*, 2014, pp. 319–325.

[48] L. van der Maaten and G. Hinton, "Visualizing data using t-SNE," *J. Of Mach. Learning Res.*, vol. 9, pp. 2579–2605, 2008.

[49] M. Kusner, Y. Sun, N. Kolkin, and K. Weinberger, "From word embeddings to document distances," in *Proc. ICML*, 2015, pp. 1023–1029.

[50] C. Elkan, "Using the triangle inequality to accelerate k-means," in *Proc. ICML*, 2003, pp. 377–385.

[51] A. Civril, M. Magdon-Ismail, and E. Bocek-Rivele, "SSDE: fast graph drawing using sampled spectral distance embedding," in *Proc. GD*, 2006, pp. 1449–1456.

[52] S. T. Roweis and L. K. Saul, "Nonlinear dimensionality reduction by locally linear embedding," *Sci.*, vol. 290, no. 5500, pp. 2323–2326, 2000.

[53] J. Chuang, C. D. Manning, and J. Heer, "Termite: Visualization techniques for assessing textual topic models," in *Proc. AVI*, 2012, pp. 658–665.

[54] W. T. McCormick Jr, P. J. Schweitzer, and T. W. White, "Problem decomposition and data reorganization by a clustering technique," *Operations Res.*, vol. 20, no. 5, pp. 993–1009, 1972.

[55] E. Alexander, J. Kohlmann, R. Valenza, M. Witmore, and M. Gleicher, "Serendip: Topic model-driven visual exploration of text corpora," in *Proc. VAST*, 2014, pp. 1081–1087.

[56] J. Chang, J. Boyd-Graber, C. Wang, S. Gerrish, and D. M. Blei, "Reading tea leaves: How humans interpret topic models," in *Proc. NIPS*, 2009, pp. 1004–1010.

[57] L. Li, K. G. Jamieson, G. DeSalvo, A. Rostamizadeh, and A. Talwalkar, "Efficient hyperparameter optimization and infinitely many armed bandits," *arXiv:1603.06560*, 2016.

[58] S. Tong and D. Koller, "Support vector machine active learning with applications to text classification," *J. Of Mach. Learning Res.*, vol. 2, no. Nov, pp. 45–66, 2001.

[59] S. Bird, "NLTK: the natural language toolkit," in *Proc. COLING/ACL*, Sydney, Australia, 2006, pp. 1772–1782.

[60] D. Dheeru and E. Karra Taniskidou, *UCI Machine Learning Repository*, Accessed on: Jan 2 2018, University of California, Irvine, School of Information and Computer Sciences, 2017. [Online]. Available: http://archive.ics.uci.edu/ml.

[61] T. Joachims, "A probabilistic analysis of the rocchio algorithm with tfidf for text categorization," CMU, Tech. Rep., 1996.

[62] W. Buntine, "Estimating likelihoods for topic models," in *Proc. ACML*, 2009, pp. 1734–1740.

[63] M. Tsikhanovich, M. Magdon-Ismail, and V. Zikas, "PD-NMF: Private distributed non-negative matrix factorization," Submitted to ICML, 2018.

[64] M. Tsikhanovich, M. Ishaq, M. Magdon-Ismail, and V. Zikas, "PD-SVD: Private distribute singular value decomposition," Submitted to UAI, 2018.

[65] R. Salakhutdinov and A. Mnih, "Probabilistic matrix factorization," in *Proc. NIPS*, 2007, pp. 820–830.

[66] B. Du, S. Wang, N. Wang, L. Zhang, D. Tao, and L. Zhang, "Hyperspectral signal unmixing based on constrained non-negative matrix factorization approach," *Neurocomputing*, vol. 204, pp. 153 –161, 2016.

[67] I. A. Qureshi, S. Zubair, and K. Khurshid, "Non negative matrix factorization, sparse coding dictionary learning techniques on fmri images," in *Proc. RAEE*, 2017, pp. 377–384.

[68] C. C. Aggarwal, *Outlier Analysis*, 2nd. Secaucus, NJ, USA: Springer, 2016.

[69] K. M. Jeon, H. K. Kim, S. J. Lee, and Y. K. Lee, "Nonnegative matrix factorization based adaptive noise sensing over wireless sensor networks," *Int. J. Of Distributed Sensor Networks*, vol. 10, no. 4, p. 640 915, 2014.

[70] A. Pratama, H. Murfi, S. Nurrohmah, and A. Iskandar, "Sensing trending topics in twitter for greater jakarta area," *Int. J. Of Elect.al And Comput. Eng.*, vol. 7, pp. 91–105, Feb. 2017.

[71] R. Gemulla, E. Nijkamp, P. J. Haas, and Y. Sismanis, "Large-scale matrix factorization with distributed stochastic gradient descent," in *Proc. KDD*, 2011, pp. 69–77.

[72] S. S. Du, Y. Liu, B. Chen, and L. Li, "Maxios: Large scale nonnegative matrix factorization for collaborative filtering," in *Proc. NIPS Workshop on Distributed Matrix Computations*, 2014, pp. 1052–1059.

[73] G. Danezis, J. Domingo-Ferrer, M. Hansen, J.-H. Hoepman, D. Le Métayer, R. Tirtea, and S. Schiffner, "Privacy and data protection by design," *Ensia*, 2015.

[74] T. Li and C. Ding, "Weighted consensus clustering," in *Proc. ICDM*, 2008, pp. 938–944.

[75] X. Zheng, S. Zhu, J. Gao, and H. Mamitsuka, "Instance-wise weighted nonnegative matrix factorization for aggregating partitions with locally reliable clusters," in *Proc. IJCAI*, 2015, pp. 817–823.

[76] J. Liu, C. Wang, J. Gao, and J. Han, "Multi-view clustering via joint nonnegative matrix factorization," in *Proc. SDM*, 2013, pp. 1413–1420.

[77] Z. Shen, P. Luo, S. Yang, and X. Shen, "Topic modeling ensembles," in *Proc. ICDM*, 2010, pp. 1839–1845.

[78] O. Goldreich, *Foundations of Cryptography: Volume 2, Basic Applications*. New York, NY, USA: Cambridge University Press, 2004.

[79] A. Kumar, V. Sindhwani, and P. Kambadur, "Fast conical hull algorithms for near-separable non-negative matrix factorization," *arXiv:1210.1190*, 2012.

[80] N. Gillis, "Successive nonnegative projection algorithm for robust nonnegative blind source separation," *Proc. Siims*, 2014.

[81] L. N. Trefethen and D. Bau III, *Numerical Linear Algebra*. Philadelphia, PA, USA: Siam, 1997, vol. 50.

[82] GroupLens. (2003). Movielens 1m dataset. Accessed on Sep 17 2017, [Online]. Available: https://grouplens.org/datasets/movielens/.

[83] Yelp. (2013). Yelp recsys-2013 kaggle dataset. Accessed on Jan 1 2018, [Online]. Available: https://www.kaggle.com/c/yelp-recsys-2013.

[84] C. Tang, Z. Xu, and S. Dwarkadas, "Peer-to-peer information retrieval using self-organizing semantic overlay networks," in *Proc. SIGCOMM*, 2003, pp. 175–186.

[85] X. Yi and J. Allan, "A comparative study of utilizing topic models for information retrieval," in *Proc. ECIR*, 2009, pp. 501–509.

[86] P. Limbeck, M. Suntinger, and J. Schiefer, "Sari openrec – empowering recommendation systems with business events," in *Proc. DBKDA*, 2010, pp. 1780–1788.

[87] J. Brickell and V. Shmatikov, "The cost of privacy: Destruction of data-mining utility in anonymized data publishing," in *Proc. SIGKDD*, 2008, pp. 70–78.

[88] B. A. Malin, K. El Emam, and C. M. O'keefe, "Biomedical data privacy: Problems, perspectives, and recent advances," *Proc. Jamia*, 2013.

[89] C. Dwork and A. Roth, "The algorithmic foundations of differential privacy," *Found. Trends Theor. Comput. Sci.*, vol. 9, no. 3, pp. 211–407, Aug. 2014.

[90] A. Rajkumar and S. Agarwal, "A differentially private stochastic gradient descent algorithm for multiparty classification," in *Proc. AIStats*, 2012, pp. 1609–1618.

[91] R. Bassily, K. Nissim, A. Smith, T. Steinke, U. Stemmer, and J. Ullman, "Algorithmic stability for adaptive data analysis," in *Proc. STOC*, 2016, pp. 1046–1059.

[92] C. Dwork, V. Feldman, M. Hardt, T. Pitassi, O. Reingold, and A. L. Roth, "Preserving statistical validity in adaptive data analysis," in *Proc. STOC*, 2015, pp. 1680–1686.

[93] V. Nikolaenko, S. Ioannidis, U. Weinsberg, M. Joye, N. Taft, and D. Boneh, "Privacy-preserving matrix factorization," in *Proc. SIGSAC*, 2013, pp. 801–812.

[94] S. Kim, J. Kim, D. Koo, Y. Kim, H. Yoon, and J. Shin, "Efficient privacy-preserving matrix factorization via fully homomorphic encryption: Extended abstract," in *Proc. AsiaCCS*, 2016, pp. 617–628.

[95] P. Mohassel and Y. Zhang, "Secureml: A system for scalable privacy-preserving machine learning," *Proc. Iacr*, 2017.

[96] S. Dynes, E. Goetz, and M. Freeman, "Cyber security: Are economic incentives adequate?" In *Proc. ICCIP*, 2007, pp. 1323–1332.

[97] R. Hirschprung, E. Toch, F. Bolton, and O. Maimon, "A methodology for estimating the value of privacy in information disclosure systems," *Comput.s In Human Behavior*, vol. 61, pp. 443–453, 2016.

[98] K. Ligett, S. Neel, A. Roth, B. Waggoner, and Z. S. Wu, "Accuracy first: Selecting a differential privacy level for accuracy-constrained ERM," *arXiv:1705.10829*, 2017.

[99] R. Bassily, A. Groce, J. Katz, and A. Smith, "Coupled-worlds privacy: Exploiting adversarial uncertainty in statistical data privacy," in *Proc. FOCS*, 2013, pp. 1279–1286.

[100] B. E. Hansen, "Lecture notes on nonparametrics," *Lecture Notes*, 2009.

[101] R. Cramer, I. B. Damgard, *et al.*, *Secure Multiparty Computation*. New York, NY, USA: Cambridge University Press, 2015.

[102] P. Kairouz, "The fundamental limits of statistical data privacy," PhD thesis, Comp. Sci. Dept, UIUC, Urbana-Champagne, IL, USA, 2016.

[103] A. D. Groce, "New notions and mechanisms for statistical privacy," PhD thesis, Comp. Sci. Dept, University of Maryland, College Park, MD, USA, 2014.

[104] A. Blum, C. Dwork, F. McSherry, and K. Nissim, "Practical privacy: The sulq framework," in *Proc. PODS*, 2005, pp. 128–138.

[105] Y.-X. Wang, S. Fienberg, and A. Smola, "Privacy for free: Posterior sampling and stochastic gradient monte carlo," in *Proc. ICML*, 2015, pp. 1088–1094.

[106] M. F. Balcan, A. Blum, S. Fine, and Y. Mansour, "Distributed learning, communication complexity and privacy," in *Proc. COLT*, 2012, pp. 603–610.

[107] Z. Liu, Y. Wang, and A. J. Smola, "Fast differentially private matrix factorization," *arXiv:1505.01419*, 2015.

[108] J. Hua, C. Xia, and S. Zhong, "Differentially private matrix factorization," in *Proc. IJCAI*, 2015, pp. 1668–1678.

[109] S. Wellek, *Testing Statistical Hypotheses of Equivalence and Noninferiority, Second Edition*. Boca Raton, FL, USA: CRC Press, 2010.

[110] M. Hardt and A. Roth, "Beyond worst-case analysis in private singular vector computation," in *Proc. STOC*, 2013, pp. 331–340.

[111] C. Shalizi, "Advanced data analysis from an elementary point of view," *lecture notes*, 2013.

[112] H.-S. Won, S.-P. Kim, S. Lee, M.-J. Choi, and Y.-S. Moon, "Secure principal component analysis in multiple distributed nodes," *Security And Communication Networks*, vol. 9, no. 14, pp. 2348–2358, 2016, sec.1501.

[113] S. Chen, R. Lu, and J. Zhang, "A flexible privacy-preserving framework for singular value decomposition under internet of things environment," *arXiv:1703.06659*, 2017.

[114] S. Han, W. K. Ng, and P. S. Yu, "Privacy-preserving singular value decomposition," in *Proc. ICDE*, 2009, pp. 381–388.

[115] N. Youdao, "P4p: Practical large-scale privacy-preserving distributed computation robust against malicious users," in *Proc. USENEX*, 2010, pp. 1632–1639.

[116] L. Fortnow, R. Rubinfeld, W. D. Smith, and P. White, "Testing that distributions are close," in *Proc. FOCS*, 2000, pp. 1715–1725.

[117] M. Sugiyama, T. Suzuki, and T. Kanamori, *Density Ratio Estimation in Machine Learning*. New York, NY, USA: Cambridge University Press, 2012.

[118] C. Huang, P. Kairouz, X. Chen, L. Sankar, and R. Rajagopal, "Context-aware generative adversarial privacy," *arXiv:1710.09549*, 2017.

APPENDIX A

Topic Modeling Evaluation Experiments

A.1 For Each Measure: Comparison of LDA, NMF, Kmeans

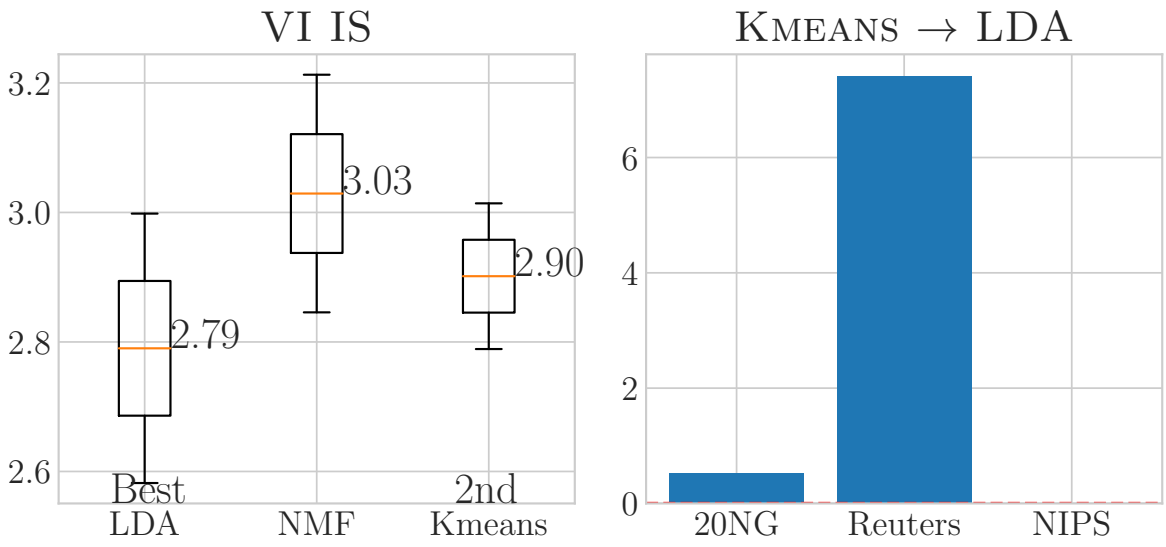


Fig. A.1: Vi Is Pct. Improvement from 2nd to Best

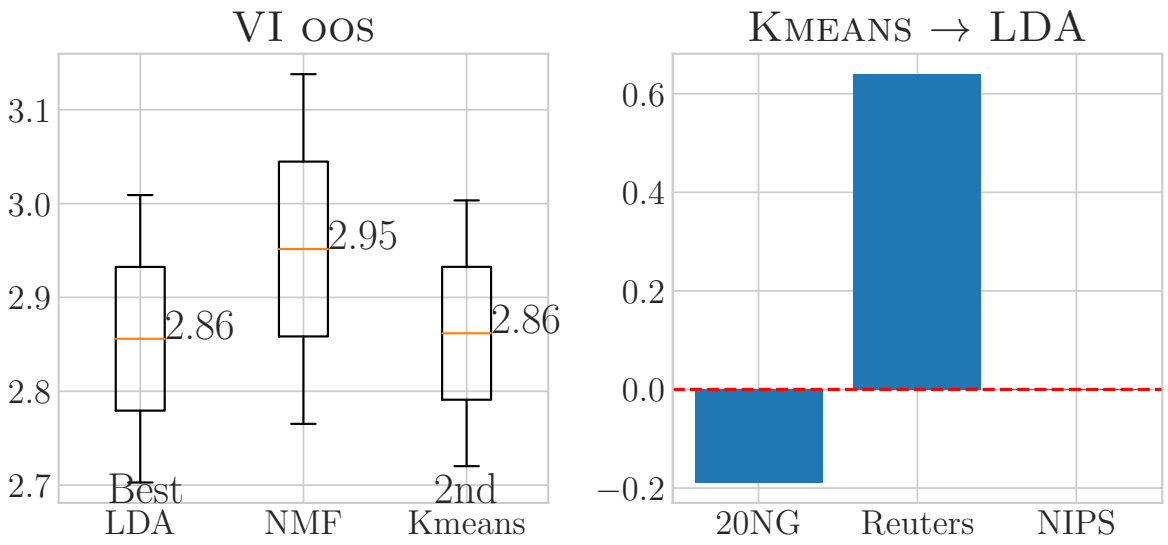


Fig. A.2: Vi Oos Pct. Improvement from 2nd to Best

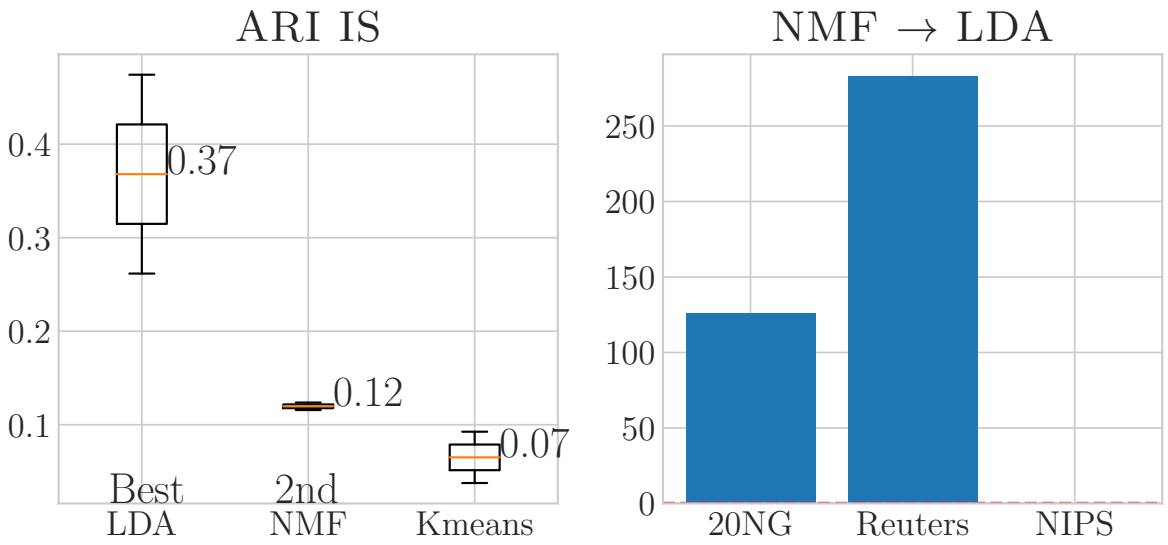


Fig. A.3: Ari Is Pct. Improvement from 2nd to Best

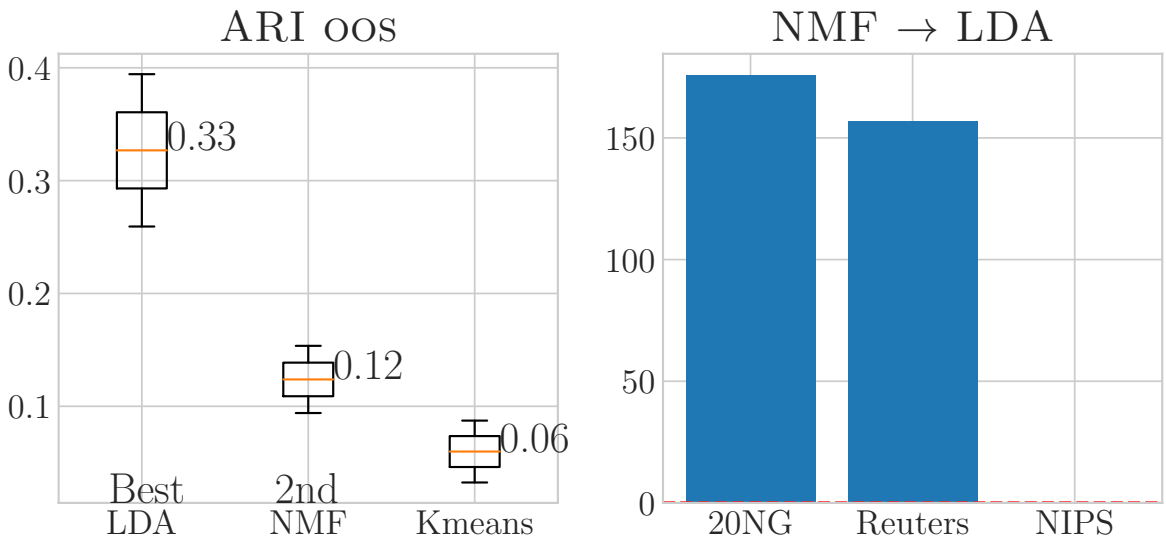


Fig. A.4: Ari Oos Pct. Improvement from 2nd to Best

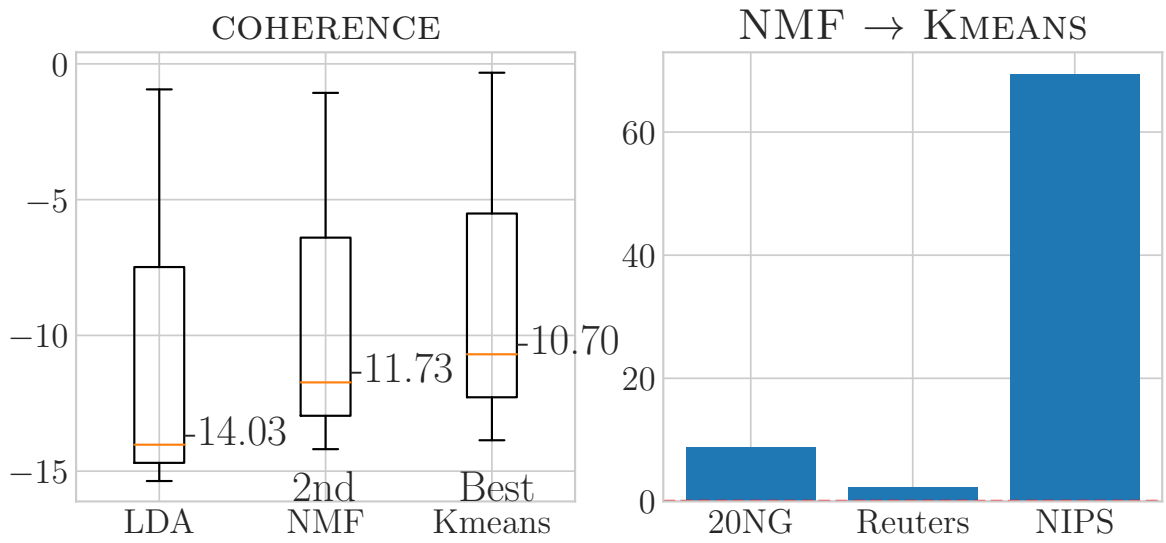


Fig. A.5: Coherence Pct. Improvement from 2nd to Best

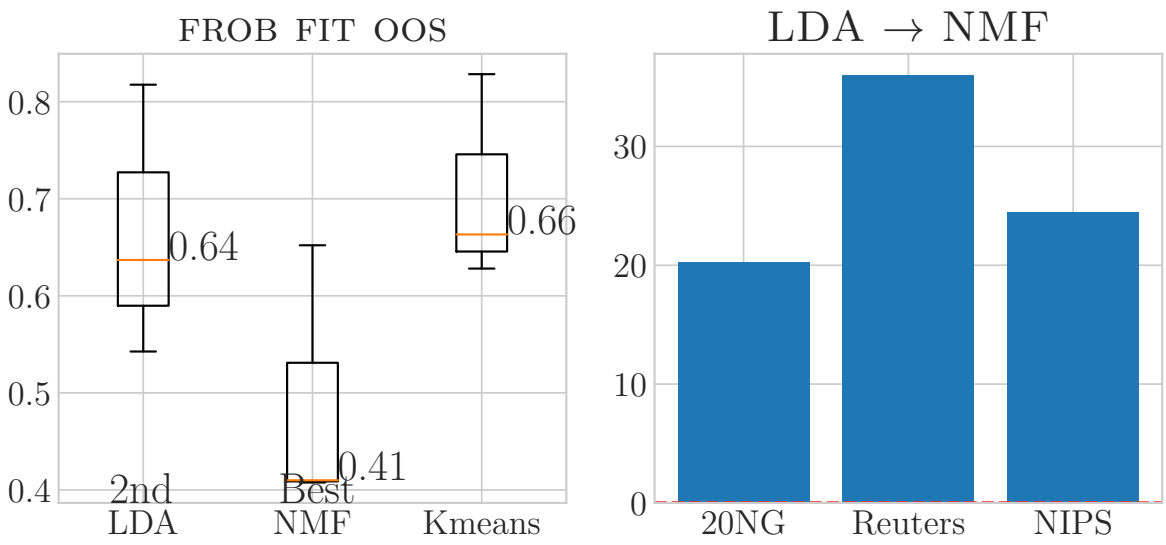


Fig. A.6: Frob Fit Oos Pct. Improvement from 2nd to Best

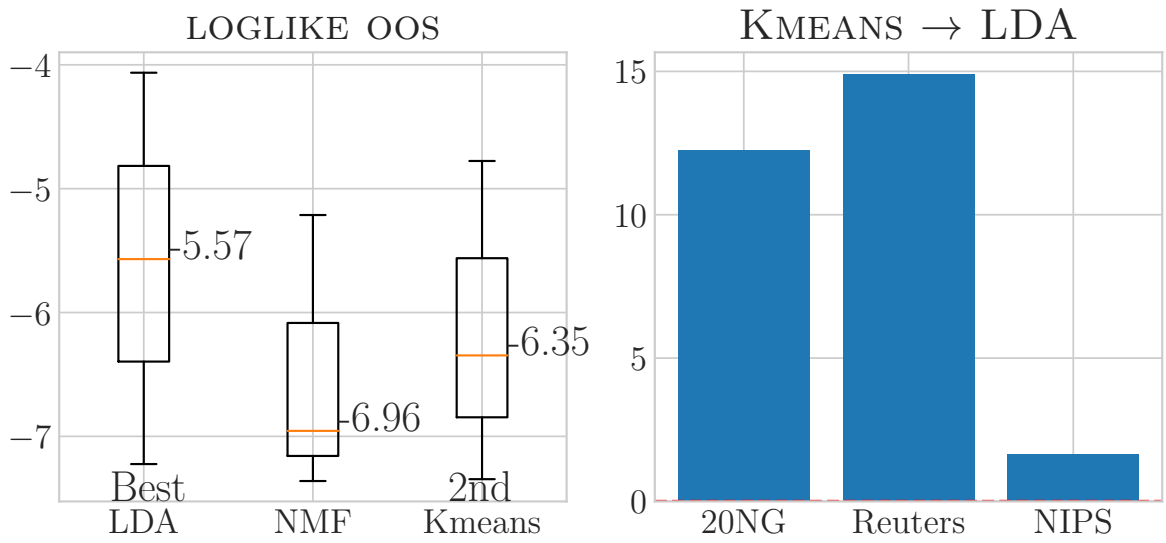


Fig. A.7: Loglike Oos Pct. Improvement from 2nd to Best

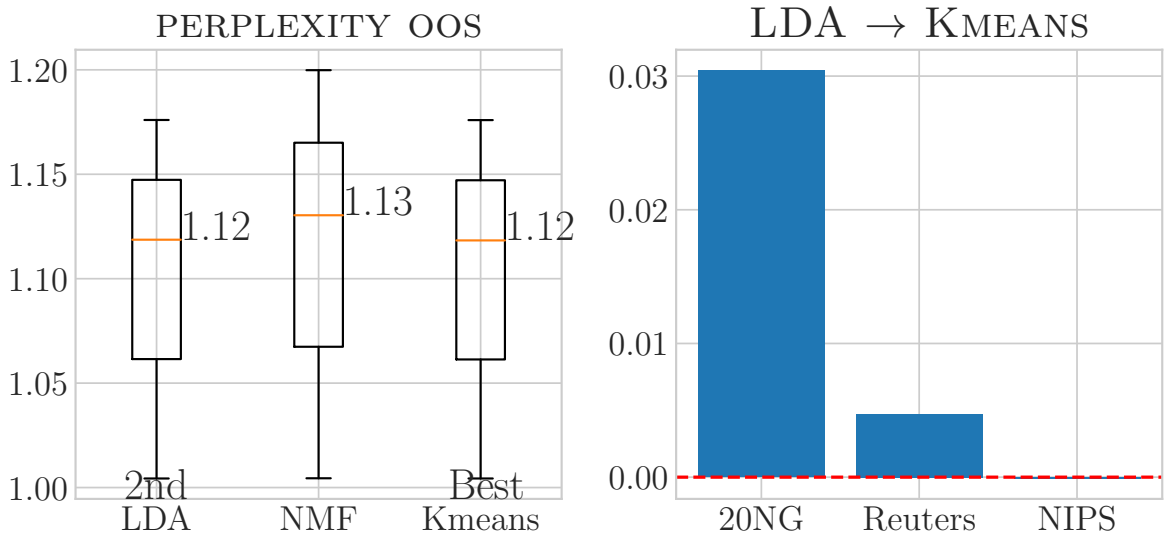


Fig. A.8: Perplexity Oos Pct. Improvement from 2nd to Best

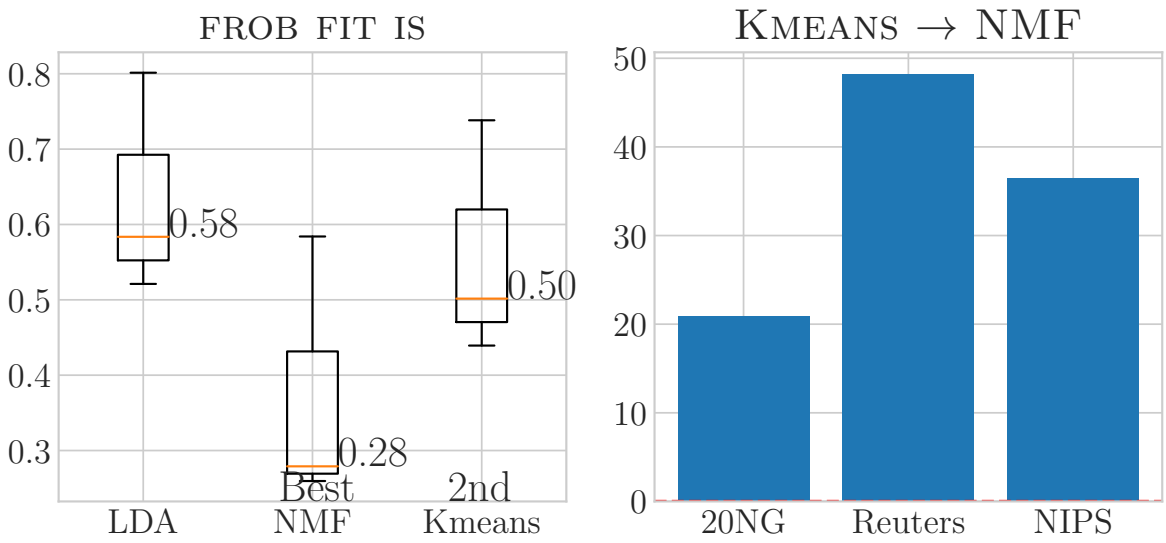


Fig. A.9: Frob Fit Is Pct. Improvement from 2nd to Best

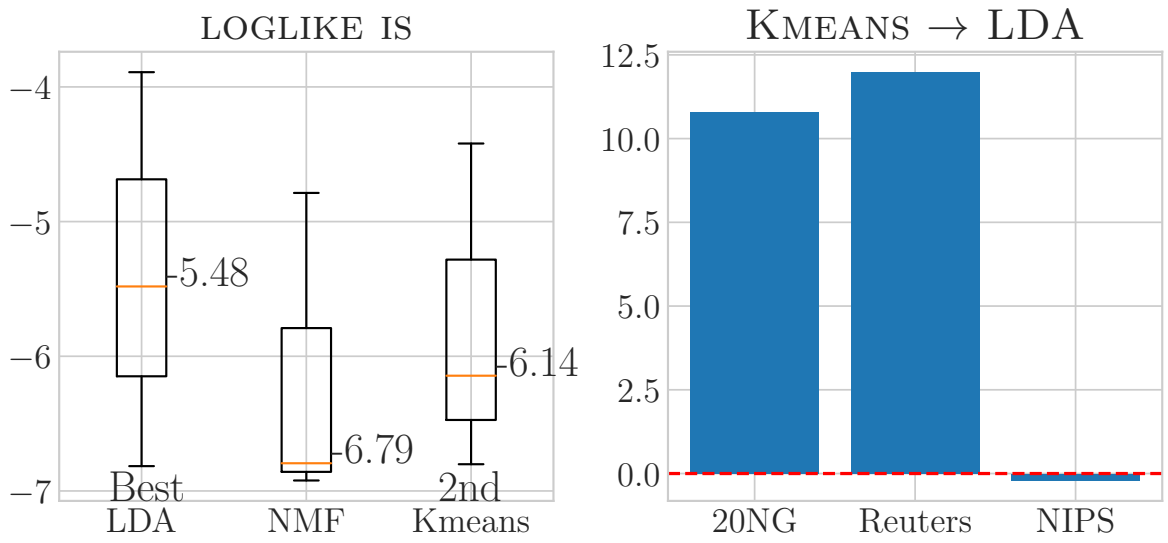


Fig. A.10: Loglike Is Pct. Improvement from 2nd to Best

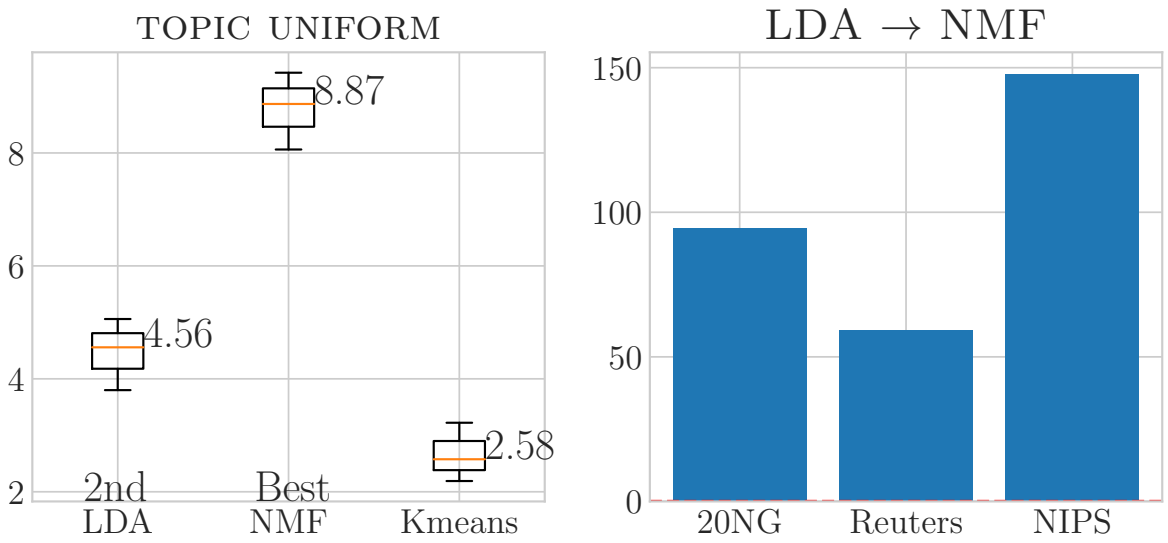


Fig. A.11: Topic Uniform Pct. Improvement from 2nd to Best

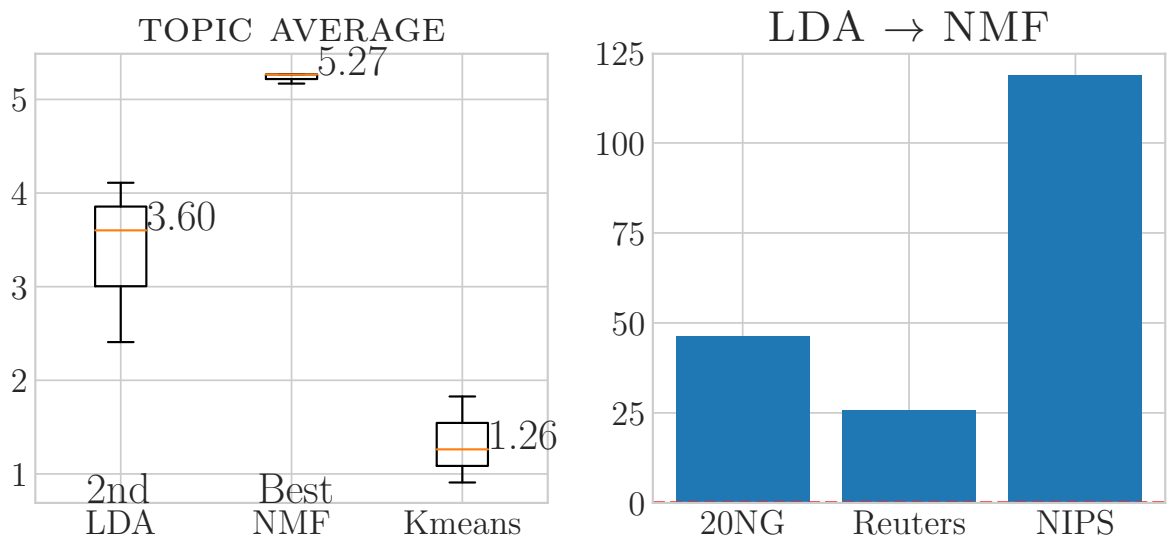


Fig. A.12: Topic Average Pct. Improvement from 2nd to Best

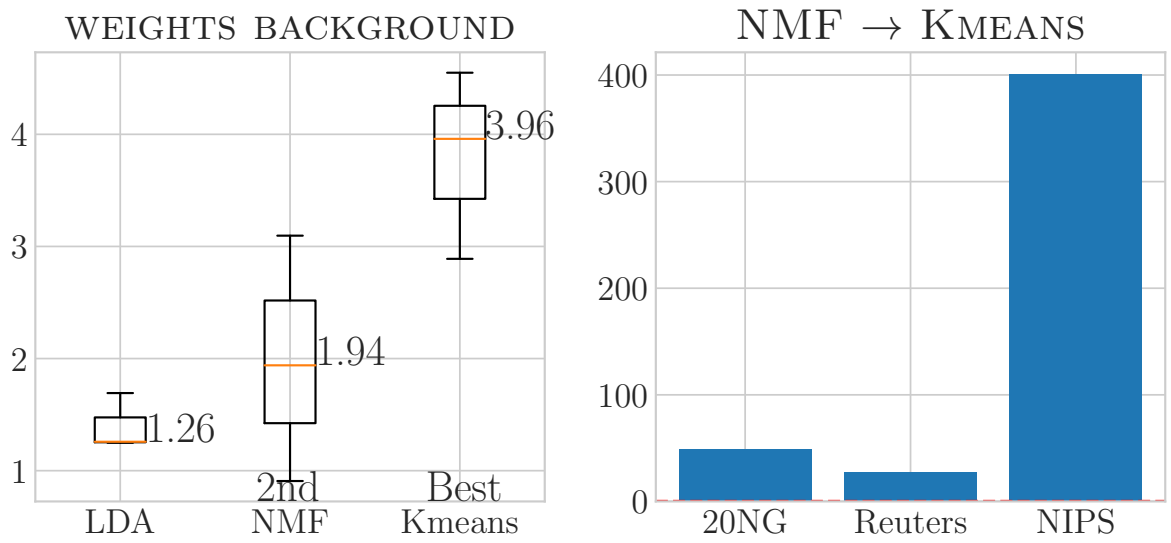


Fig. A.13: Weights Background Pct. Improvement from 2nd to Best

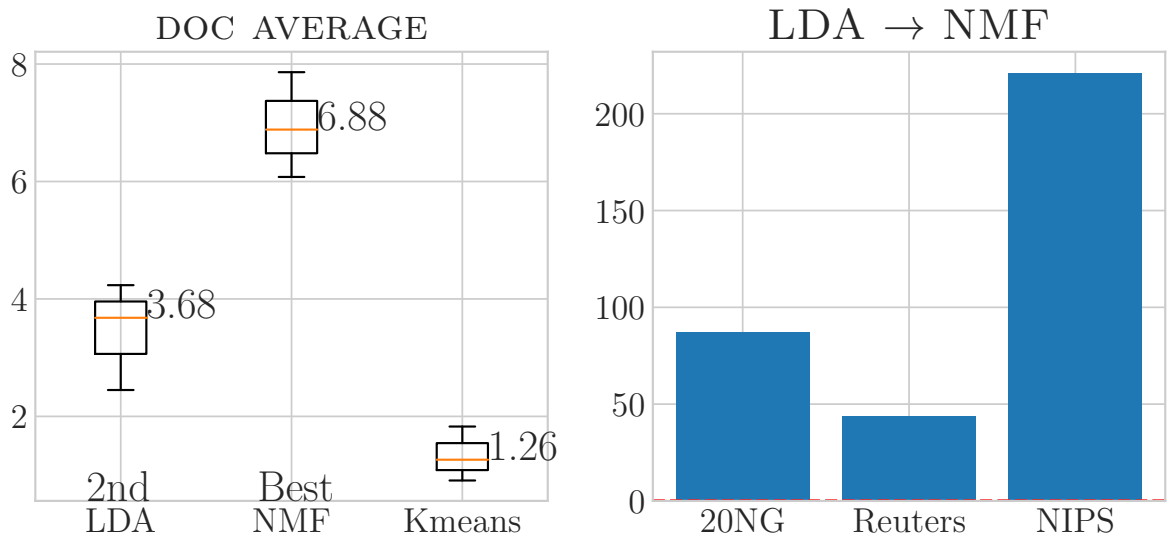


Fig. A.14: Doc Average Pct. Improvement from 2nd to Best

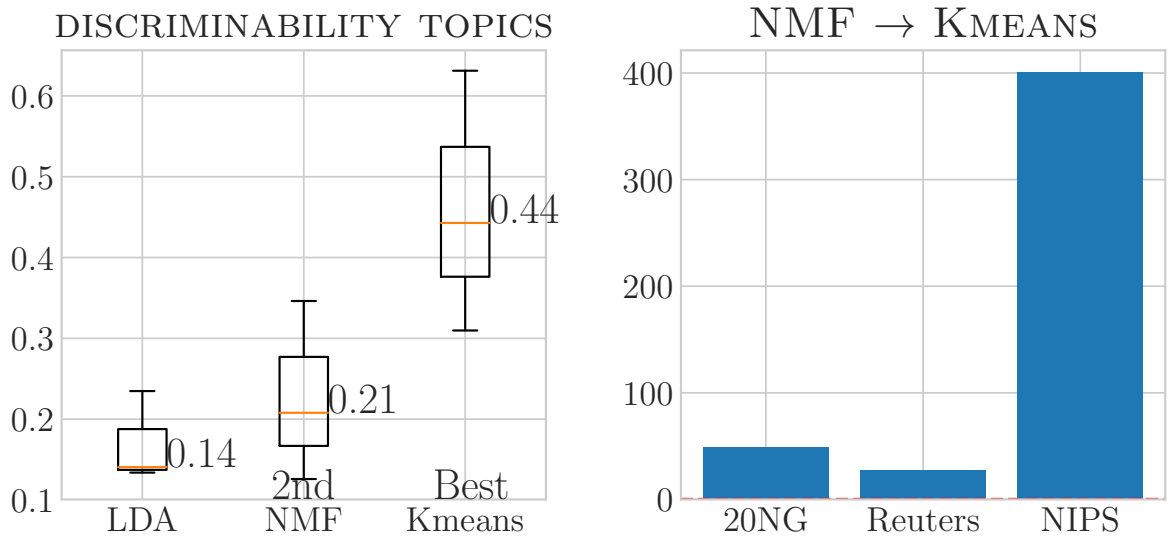


Fig. A.15: Discriminability Topics Pct. Improvement from 2nd to Best

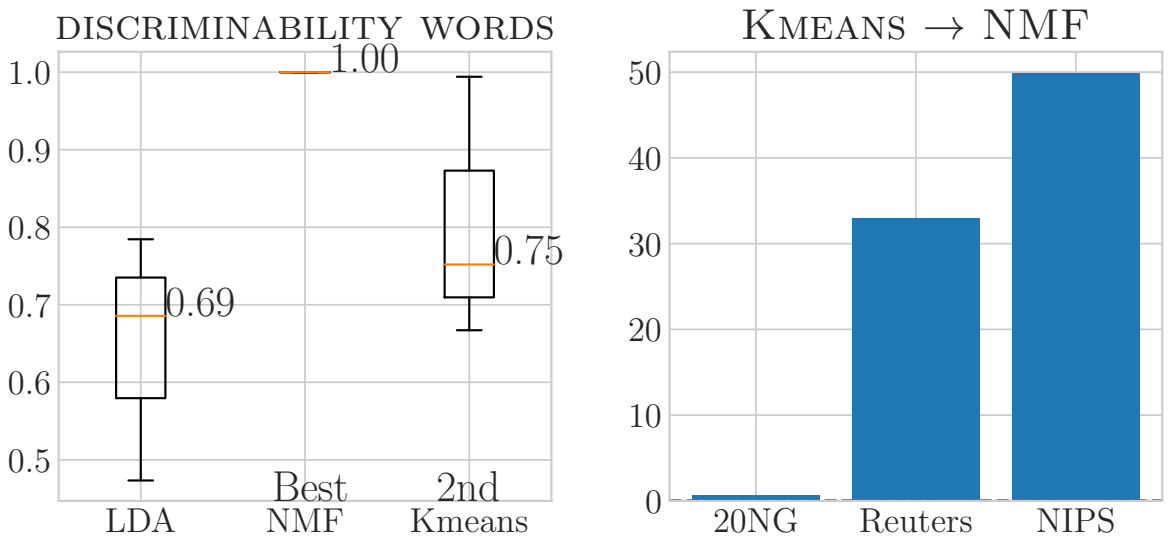


Fig. A.16: Discriminability Words Pct. Improvement from 2nd to Best

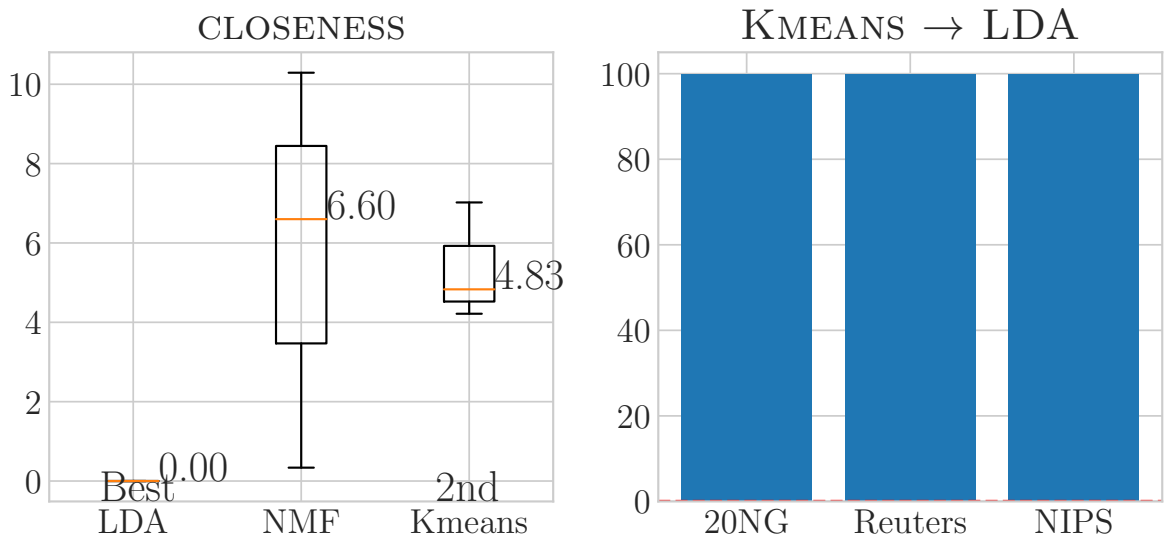


Fig. A.17: Closeness Pct. Improvement from 2nd to Best