CS264: Beyond Worst-Case Analysis
Lecture #15: Topic Modeling and Nonnegative Matrix Factorization*

Tim Roughgarden†

February 28, 2017

1 Preamble

This lecture fulfills a promise made back in Lecture #1, to investigate theoretically the unreasonable effectiveness of machine learning algorithms in practice. As always, pursuing provable bounds for algorithms for hard problems can pay off in two different ways. The first is to achieve a theoretical explanation of the empirical performance of existing algorithms.[1] The second is to use a better understanding of the properties of "real-world" instances to design new and better algorithms. This lecture falls into the second category.

We'll consider the problem of *discovering topics* from an unlabeled collection of documents (so, another unsupervised learning problem). The goal is to discover which documents are about which topics. But the topics are not given in advance! The algorithm is also responsible for automatically figuring out what the topics are. It is easy to imagine why solving this task would be useful, for example in organizing a large collection of documents for easy navigation by a human. (E.g., recommending documents that are similar to the one currently being read.)

We'll see that a natural formulation of this topic discovery problem is NP-hard in the worst case, but becomes polynomial-time solvable under an assumption that makes sense in the motivating applications. (So like our work in clustering, articulating plausible structure in "real" instances leads to provably tractability.) This contributes to an ongoing theme of the course, of being willing to make compromises in algorithm analysis that are specific to the application domain, with the goal of getting good advice about how to solve the problem,

†Department of Computer Science, Stanford University, 474 Gates Building, 353 Serra Mall, Stanford, CA 94305. Email: tim@cs.stanford.edu.

[1]Much of our study of "stable" and "planted" solutions has this flavor, with the theory explaining when tried-and-true algorithms like spectral algorithms or linear programming work well.

even if the specific assumptions used do not make sense for other problems. (This contrasts with worst-case analysis, which is defined generically across all problems.)

2 The Singular Value Decomposition Approach

2.1 The Setup

We summarize a corpus of documents with respect to a fixed vocabulary as an $m \times n$ matrix $\mathbf{M}$, where rows correspond to words and columns to documents. (So m is the number of words that we're keeping track of, and n is the number of documents we've got.) The entry M_{ij} denotes the number of times that the ith word occurs in the jth document. This matrix $\mathbf{M}$ is sometimes called the *term-by-document* matrix. Note that summarizing the documents by $\mathbf{M}$ throws away some information, namely the ordering of the words in each document. This is sometimes called the "bag of words" assumption, and it is standard in data mining, natural language processing, etc.

Our goal is to find k *topic vectors*, which are vectors indexed by words (so in $\mathbb{R}^m$), such that every document can be described (at least approximately) as a linear combination of the topic vectors. Intuitively, a topic vector indicates the relative propensity of the different words to appear in a document on that topic.[2]

2.2 Low-Rank Factorizations

In matrix terminology, this goal translates to the problem of finding a low-rank approximation of $\mathbf{M}$. That is, we want to express $\mathbf{M}$ (at least approximately) as the product of a long and skinny $(m \times k)$ matrix $\mathbf{A}$ and a short and wide $(k \times n)$ matrix $\mathbf{B}$ (Figure 1). Note that when such a factorization exists, it means that every column of $\mathbf{M}$ can be expressed as a linear combination of the columns in $\mathbf{A}$ (with the corresponding column of $\mathbf{B}$ giving the k coefficients in the linear combination); similarly, every row of $\mathbf{M}$ can be expressed as a linear combination of the rows of $\mathbf{B}$ (with the coefficients described by the corresponding row of $\mathbf{A}$). Thus, we can take the columns of $\mathbf{A}$ as our topic vectors—every document arises a suitable linear combination of them. Each column of $\mathbf{B}$ can then be interpreted as the mixture of topics that the corresponding document is about.

For example, suppose there are three words in the vocabulary and four documents, with term-by-document matrix

$$\mathbf{M} = \begin{bmatrix} 1 & 3 & 2 & 0 \\ 2 & 0 & 1 & 3 \\ 3 & 1 & 2 & 4 \end{bmatrix}. \tag{1}$$

[2]How do we know what k is? The same comments apply as in the k-median and k-means problems: perhaps you have some domain knowledge telling you roughly what k should be, or perhaps you just rerun an algorithm for several choices of k, settling on one that seems to give you the best results.

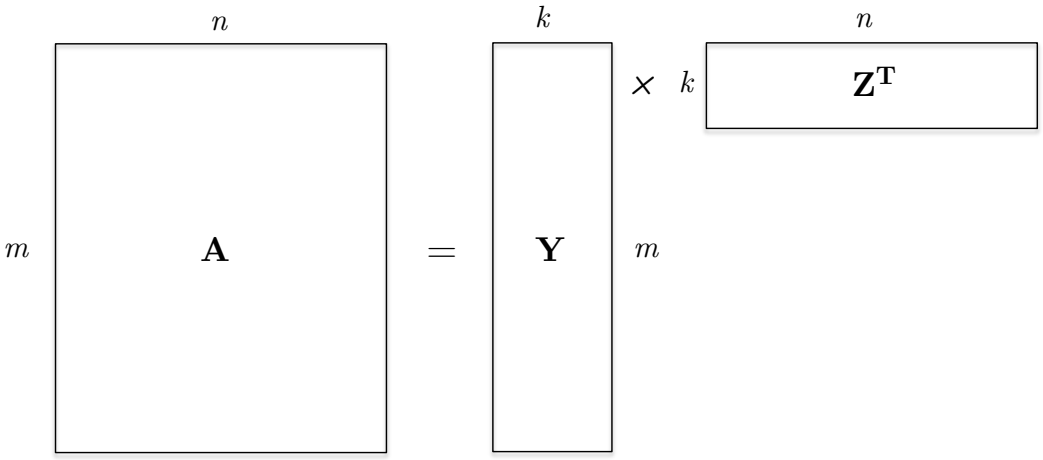

Figure 1: A low-rank factorization. (With $\mathbf{A}, \mathbf{Y}, \mathbf{Z}^\top$ playing the role of $\mathbf{M}, \mathbf{A}, \mathbf{B}$ in the text.)

This matrix has a low-rank factorization with $k = 2$, such as

$$\begin{bmatrix} 1 & 3 & 2 & 0 \\ 2 & 0 & 1 & 3 \\ 3 & 1 & 2 & 4 \end{bmatrix} = \begin{bmatrix} 1 & 3 \\ 2 & 0 \\ 3 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & \frac{1}{2} & \frac{3}{2} \\ 0 & 1 & \frac{1}{2} & -\frac{1}{2} \end{bmatrix}.$$

For example, perhaps the two topics represent "politics" and "baseball." Then, looking at the first matrix on the right-hand side (whose columns indicate the topics), we see that the middle word is one that never appears in document on baseball, but sometimes does in documents on politics (e.g., "filibuster"). The first word is more frequent in documents on baseball than politics (e.g., "home run") and vice versa for the third word (e.g., "president"). The second matrix on the right-hand side indicates what each document is about. So the first document is solely about politics, the second solely about baseball, and the third about both, and the fourth document would seem to be strongly about politics and the "opposite" of baseball (whatever that means).[3]

Assuming that $\mathbf{M}$ admits a low-rank factorization, how can we find one? The standard approach (as covered in e.g. CS168) is the *singular value decomposition (SVD)*. To remind you how this works, recall that every $m \times n$ matrix $\mathbf{M}$ can be written as the product $\mathbf{USV}^\top$ (Figure 2), where $\mathbf{U}$ is an $m \times m$ orthogonal matrix (i.e., whose columns form an orthonormal basis of $\mathbb{R}^m$), $\mathbf{V}$ is an $n \times n$ orthogonal matrix, and $\mathbf{S}$ is a $m \times n$ diagonal matrix (meaning with zeroes in every entry not of the form (i, i) for some i). Moreover, the diagonal entries of $\mathbf{S}$ are nonnegative, and we can permute the rows and columns so that they occur is descending order. For a proof, see any course on linear algebra or numerical analysis. The columns of

[3]This model of topics and documents is obviously simplistic. For example, two documents about the same topic have exactly the same set of words! A more realistic model is that two different documents about the same topic are i.i.d. samples from some (topic-dependent) distribution over words. The ideas discussed in this lecture can be extended to this more realistic case (with some additional work); see e.g. [**?**].

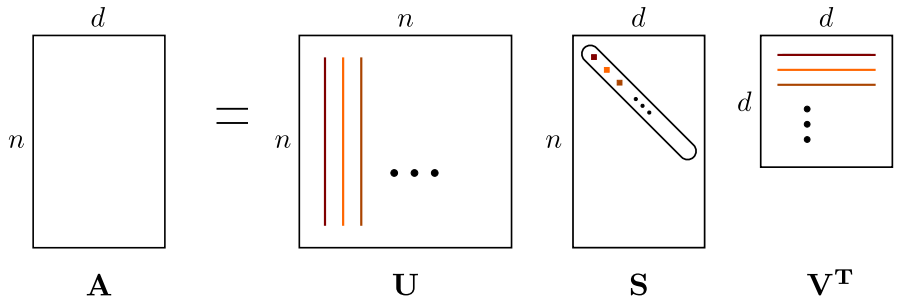

Figure 2: The singular value decomposition (SVD). Each singular value in $\mathbf{S}$ has an associated left singular vector in $\mathbf{U}$, and right singular vector in $\mathbf{V}$.

$\mathbf{U}$ are called the *left singular vectors* of $\mathbf{M}$, the columns of $\mathbf{V}$ (rows of $\mathbf{V}^\top$) the *right singular vectors*, and the diagonal entries of $\mathbf{S}$ the *singular values*. Note that each singular value has associated left and right singular vectors. By the "top" left or right singular vector, we mean the one associated with the largest singular value. (So after the permutation above, the first column of $\mathbf{U}$ or $\mathbf{V}$.) Note that the SVD expresses the columns of the matrix $\mathbf{M}$ as linear combinations of the first $\min\{m, n\}$ left singular vectors (with coefficients given in the columns of $\mathbf{V}^\top$, after scaling by the singular values)), and similarly the rows of $\mathbf{M}$ as linear combinations of the first $\min\{m, n\}$ right singular vectors. Thus left singular vectors are useful for exploring and explaining the column structure of a matrix, while the right singular vectors are useful for understanding the row structure.

Conceptually, re-representing the matrix $\mathbf{M}$ using the SVD provides a list of $\mathbf{M}$'s ingredients (the outer products of the left and right singular vectors), ordered by "importance" (as measured by the singular values).

We saw a special case of the SVD back in Lecture #9, when we discussed the spectral theorem for symmetric matrices. There, we could take $\mathbf{U} = \mathbf{V}$ (provided we allow negative entries in the diagonal matrix). Of course, a term-by-document matrix is not going to be symmetric (or even square), so we really do need to talk about the more generally defined SVD.

2.3 Using the SVD to Define Topic Vectors: Pros and Cons

There is no k in the SVD. So how can we extract k topic vectors from it? Recall that the goal is to (at least approximately) express documents (i.e., columns of our matrix $\mathbf{M}$) as linear combinations of the topic vectors. The left singular vectors of the SVD can be used express the columns of $\mathbf{M}$. We only get to choose k vectors, though, so which ones? The most important ones, naturally, corresponding to the largest singular values. To summarize, the SVD approach to topic modeling is to take the topic vectors as the top k left singular vectors of the term-by-document matrix.

The SVD approach has several things going for it, and in many case produces reasonable results. First, it is an optimal low-rank approximation of a matrix in various senses.

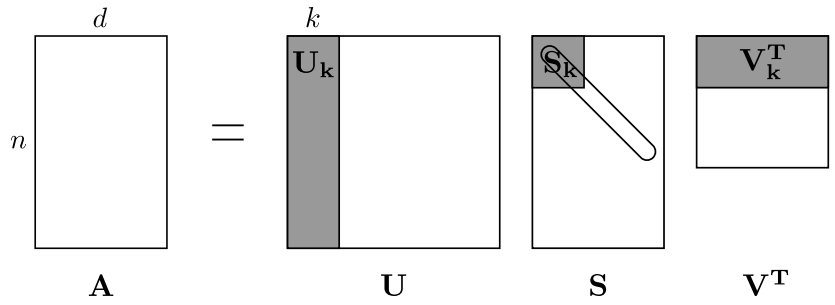

Figure 3: Low rank approximation via SVD. Recall that $\mathbf{S}$ is non-zero only on its diagonal, and the diagonal entries of S are sorted from high to low. Our low rank approximation is $\mathbf{A}_k = \mathbf{U}_k\mathbf{S}_k\mathbf{V}_k^\top$.

Specifically, suppose we derive a rank-k matrix $\mathbf{M}^{(k)}$ from $\mathbf{M}$ by throwing out all but the top k left and right singular vectors and singular values. This leaves us with the product of an $m \times k$ matrix (the top k left singular vectors), a $k \times k$ diagonal matrix (of the top k singular values), and a $k \times n$ matrix (the top k right singular vectors). See Figure 3. Then $\mathbf{M}^{(k)}$ minimizes the norm $\|\mathbf{M} - \mathbf{A}\|$ of the residual matrix over all rank-k matrices $\mathbf{A}$ (for the Frobenius norm, or the operator norm). Second, there are reasonably fast algorithms for computing the SVD, running time $O(n^2m)$ or $O(m^2n)$ (whichever is smaller). There are highly optimized implementations in, e.g., Matlab (where a one-line command returns you $\mathbf{U}$, $\mathbf{S}$, and $\mathbf{V}$ on a silver platter).

However, there are also at least two significant drawbacks to the SVD solution. First, the SVD approach forces the topic vectors to be orthogonal. (Recall $\mathbf{U}$ is an orthonormal matrix, and so its columns are all orthogonal to each other.) Whereas we expect many real-life topics to be correlated. For example, articles about baseball and articles about basketball might be treated as different topics, even though there is a significant shared vocabulary. The second problem is already evident in (1)—negative entries. In the example the negative entry is not in a topic vector but in the mixing coefficients, but there are also examples where the topic vectors have negative entries (exercise). Negative entries are bad for interpretability. One can interpret a negative value for a word in a topic vector as a signal that documents containing that word are almost never about that topic. But these semantics have problems: for example, they suggest that two documents that are each very much not about some topic should be viewed as similar documents.

3 Nonnegative Matrix Factorization

3.1 The Problem

To address both of the issues above, we consider a different formulation of the low-rank matrix approximation problem. Given is an $m \times n$ nonnegative matrix $\mathbf{M}$ (meaning all its

entries are nonnegative), like a term-by-document matrix. The goal is to express $\mathbf{M}$, at least approximately, as the product of *nonnegative* $m \times k$ and $k \times n$ matrices $\mathbf{A}$ and $\mathbf{B}$, where k is given (and hopefully small).[4,5] Figure 1 applies also here, with the additional constraint that the factor matrices be nonnegative. Note that there is no orthogonality requirement whatsoever. This is the *nonnegative matrix factorization (NMF)* problem. For a term-by-document matrix $\mathbf{M}$, a solution to the NMF problem is naturally interpreted as a solution to the topic discovery problem, with the k columns of $\mathbf{A}$ providing the (nonnegative) topic vectors, and the n columns of $\mathbf{B}$ providing the (nonnegative) topic mixture of each of the documents.

Unlike the SVD, the NMF problem is NP-hard in the worst case.[6] Are there any well-motivated assumptions that we can impose on the problem to recover computational tractability?

3.2 The Anchor Words (Separability) Assumption

For the rest of this lecture, we'll assume that the given term-by-document matrix $\mathbf{M}$ admits a nonnegative matrix factorization $\mathbf{M} = \mathbf{AB}$ with the following property:

(*) for every column j of $\mathbf{A}$, there exists a row i such that $A_{ij} > 0$ and $A_{ij'} = 0$ for all $j' \neq j$.

Recall that we interpret the columns of $\mathbf{A}$ as the topic vectors, indexed by words (the rows). So what (*) asserts is that each topic (corresponding to column j) has an *anchor word* (corresponding to row i) that *only ever appears in documents that are about topic j*. While a fairly extreme assumption, it does seem to often hold approximately in practice. For example, "401K" might only appear in documents that are at least in part about the topic of personal finance, and similarly for "Buster Posey" and the topic of baseball. We will call this the *anchor words* assumption. It is also called the *separability* assumption. Importantly, while we assume the existence of an anchor word for each topic, we have no advance knowledge of which words are the anchor words.

4 The Main Result

We'll prove the following theorem.

Theorem 4.1 ([?]) *Under the anchor words assumption, the NMF problem is polynomial-time solvable (for arbitrary k).*

[4]It is obviously necessary for $\mathbf{M}$ to be nonnegative for this to be possible. Conversely, if $\mathbf{M}$ is nonnegative, the problem is trivially solvable when $k = \min\{m, n\}$ (why?).

[5]The smallest k for which this can be done is called the *nonnegative rank* of $\mathbf{M}$. This definition is the same as (one of the standard definitions of) matrix rank, except for the extra nonnegativity requirement.

[6]It can be solved in polynomial time when k is a constant (independent of m and n), but the dependence on k is badly exponential, precluding any practical use [**?**].

A nice feature of this result is that the quest for provable bounds naturally leads to the development of new algorithms, some of which have proved superior in practice to the previous state-of-the-art.

4.1 Normalization

We begin with some preliminary observations regarding normalization. This will make the solution "more unique," and also clarify the role of the anchor words assumption.

1. Without loss of generality, for every row of $\mathbf{M}$, the sum of its entries equals 1. We can enforce this assumption by explicitly scaling each row of the given matrix $\mathbf{M}$ appropriately.[7] Any solution $\mathbf{M} = \mathbf{AB}$ of the original problem maps to a solution of the new problem, with the rows of $\mathbf{A}$ scaled in exactly the same way, and conversely. Thus any solution to the scaled problem is easily mapped to one of the original problem.

2. Without loss of generality, for every row of $\mathbf{B}$, the sum of its entries equals 1. This is because for any solution $\mathbf{M} = \mathbf{AB}$ to the problem, there is another solution $\mathbf{M} = \mathbf{A}'\mathbf{B}'$ with the desired property. If the ith row of $\mathbf{B}$ has sum λ_i, then scale it through by λ_i, and scale the ith column of $\mathbf{A}$ by $1/\lambda_i$. The product of the two matrices is unchanged.

3. Given the previous two assumptions, it follows logically that every row of $\mathbf{A}$ also has unit sum. This is because the NMF expresses every row of $\mathbf{M}$ as a nonnegative linear combination of the rows of $\mathbf{B}$. Since every row of $\mathbf{M}$ and of $\mathbf{B}$ has row sum 1, this nonnegative linear combination must actually be a convex combination (i.e., a weighted average of the rows of $\mathbf{B}$). Thus the entries of the corresponding row of $\mathbf{A}$ sum to 1.

With these normalizations, we can now understand the NMF problem as: given a (normalized) $m \times n$ matrix $\mathbf{M}$, find a $k \times n$ matrix $\mathbf{B}$ such that every row of $\mathbf{M}$ is in the convex hull of the rows of $\mathbf{B}$.[8] Given such a matrix $\mathbf{B}$, the rows of $\mathbf{A}$ then simply describe the coefficients of the relevant convex combinations.

4.2 The Anchor Words Assumption

We have not yet used the anchor words assumption in any way—how is it helpful? Recall that the original version (*) of the assumption asserts that for each column j of the (nonnegative) matrix $\mathbf{A}$, there is a row i (corresponding to the anchor word for topic j) such that A_{ij} is the only positive entry in the row. Now that we know we can assume that all of the rows of $\mathbf{A}$ have unit sum, we see that this ith row of $\mathbf{A}$ must actually have a 1 in the jth column and zeros elsewhere. Since there exists such a row (i.e., an anchor word) for each column j (by assumption), we see that $\mathbf{A}$ has embedded in it a copy of the $k \times k$ identify matrix $\mathbf{I}_k$ (after a permutation of the rows and columns). Since $\mathbf{M} = \mathbf{AB}$, with rows of $\mathbf{A}$ specifying

[7] We can also delete any all-zero rows without affecting the problem.

[8] Recall that the *convex hull* of a set $S = \{\mathbf{x}_1, \ldots, \mathbf{x}_k\} \subseteq \mathbb{R}^d$ of points is the set of their convex combinations: $CH(S) = \{\sum_{i=1}^k \lambda_i \mathbf{x}_i : \lambda_1, \ldots, \lambda_k \geq 0 \text{ and } \sum_{i=1}^k \lambda_i = 1\}$.

the coefficients of the appropriate convex combinations of the rows of $\mathbf{B}$, we see that there is a solution $\mathbf{M} = \mathbf{AB}$ such that *every row of* $\mathbf{B}$ *appears as a row of* $\mathbf{M}$, unsullied by any combination with other rows of $\mathbf{B}$. (If the ith row of $\mathbf{A}$ has a 1 in the jth column and zeroes elsewhere, then the ith row of $\mathbf{M}$ is the same as the jth row of $\mathbf{B}$.) This is kind of amazing: $\mathbf{B}$ is one of the unknown matrices that we're trying to reconstruct, and it turns out to be hiding in plain sight as a subset of the rows of the known matrix $\mathbf{M}$! (And all other rows of $\mathbf{M}$ are in the convex hull of this subset of rows.) This is the crucial implication of the anchor words assumption. We're not done though, since we don't know a priori which rows of $\mathbf{M}$ are also rows of $\mathbf{B}$—equivalently, we don't know in advance what the anchor words are.

4.3 The Algorithm

Summarizing our previous observations: we are looking for a $k \times n$ matrix $\mathbf{B}$ such that every row of $\mathbf{M}$ is in the convex hull of the rows of $\mathbf{B}$, and (by the anchor words assumption) we can restrict attention to matrices $\mathbf{B}$ that consist of k of the rows of $\mathbf{M}$. Identifying the appropriate rows of $\mathbf{M}$ (equivalently, the anchor words for the topic vectors of $\mathbf{A}$) can be done greedily.[9]

Finding Anchor Words

1. Initialize S to contain all of the rows of $\mathbf{M}$.
2. While there is a row in S that lies in the convex hull of the other rows in S:
   (a) Delete an arbitrary such row from S.
3. Return the matrix $\mathbf{B}$ whose rows are the remaining rows in S.

Checking if one point is in the convex hull of other points can be done using linear programming (Homework #8). Note that duplicate rows will automatically be deleted (one copy is trivially in the convex hull of another), so we assume from now on that all of the rows under consideration in S are distinct.

For the analysis, let's first review some basics about convex hulls. First, recall the definition (for distinct points $S = \{\mathbf{x}_1, \ldots, \mathbf{x}_\ell\}$ in $\mathbb{R}^d$ for some d):

$$CH(S) = \left\{ \sum_{i=1}^{\ell} \lambda_i \mathbf{x}_i \ : \ \lambda_1, \ldots, \lambda_\ell \geq 0 \text{ and } \sum_{i=1}^{\ell} \lambda_i = 1 \right\}.$$

A point $\mathbf{y}$ in this convex hull is a *vertex* if the only way to express $\mathbf{y}$ as a convex combination of the $\mathbf{x}_i$'s is by taking $\lambda_i = 1$ for some i (and hence $\lambda_j = 0$ for $j \neq i$). The reader should verify the following:

[9] There is also a faster "bottom-up" version; see [**?**].

(i) Every vertex of the convex hull is one of the points in S, but the converse does not always hold.

(ii) The vertices of $CH(S)$ are precisely the points of S that are not in the convex hull of the other points of S.

(iii) Deleting a non-vertex from S does not change the convex hull $CH(S)$.

For the analysis of the algorithm, consider k rows of $\mathbf{M}$ such that every row of $\mathbf{M}$ lies in their convex hull. (Which exist by the anchor words assumption and our normalizations.) By (i), the convex hull of all of the rows of $\mathbf{M}$ has at most k vertices (corresponding to the k chosen rows). By (ii) and (iii), the top-down algorithm will delete all of the rows of $\mathbf{M}$ except for the k (at most) vertices of the convex hull, and so will halt with a set S of at most k rows such that every row of $\mathbf{M}$ is in the convex hull of the rows of S. These rows form the rows of the matrix $\mathbf{B}$. With $\mathbf{B}$ now fixed, the equation $\mathbf{M} = \mathbf{AB}$ becomes a linear system (in the A_{ij}'s) and can be solved by (say) Gaussian elimination. This final step is just solving, for each row i of $\mathbf{M}$, for the coefficients of a convex combination of the rows of $\mathbf{B}$ that is equal to the row.

References