Chapter 2

Nonnegative Matrix Factorization

In this chapter we will explore the nonnegative matrix factorization problem. We will first recap the motivations from this problem. Next, we give new algorithms that we apply to the classic problem of learning the parameters of a topic model.

2.1 Introduction

In order to understand why nonnegative matrix factorization is useful in applications, it will be helpful to compare it to the singular value decomposition. We will focus on applications of both of these to text analysis in this chapter.

Singular Value Decomposition

Given an $m \times n$ matrix M, its singular value decomposition is

$$M = U\Sigma V^T$$

where U and V are orthonormal and Σ is diagonal and its entries are nonnegative. Alternatively we can write

$$M = \sum_{i=1}^{r} u_i \sigma_i v_i^T$$

where u_i is the i^{th} column of U, v_i is the i^{th} column of V and σ_i is the i^{th} diagonal entry of Σ.

Every matrix has a singular value decomposition! In fact, this representation can be quite useful in understanding the behavior of a linear operator or in general

for extracting significant "features" from a large data matrix. We will focus our discussion of the singular value decomposition on the latter. One of the many useful properties of this decomposition is that we can immediately read-off the best low rank approximation to M from it.

Definition 2.1.1 *The Frobenius norm of a matrix M is $\|M\|_F = \sqrt{\sum_{i,j} M_{i,j}^2}$. Alternately, if $M = \sum_{i=1}^{r} u_i \sigma_i v_i^T$, $\|M\|_F = \sqrt{\sum \sigma_i^2}$.*

Consider the following optimization problem: Let B be the best rank k approximation to M in the Frobenius norm - i.e. B is the minimizer of $\|M - B\|_F$ over all rank at most k matrices. Then we can without loss of generality choose B to be the first k terms of the singular value decomposition.

Theorem 2.1.2 (Eckart-Young) *The best rank k approximation to M in Frobenius norm is attained by $B = \sum_{i=1}^{k} u_i \sigma_i v_i^T$, and its error is $\|M-B\|_F = \sqrt{\sum_{i=k+1}^{r} \sigma_i^2}$.*

This is one of the reasons why the singular value decomposition is so widely useful: if we are given data in the form of a matrix M but we believe that the data is approximately low-rank, a natural approach to making use of this structure is to instead work with the best rank k approximation to M. This theorem is quite robust and holds even when we change how we measure how good B is as an approximation to M:

Definition 2.1.3 *The operator norm of a matrix M is $\|M\|_2 = \max_{|v|=1} \|Mv\|_2$. Then if $M = \sum_{i=1}^{r} u_i \sigma_i v_i^T$, $||M||_2 = \sigma_1$ (the largest singular value).*

The best approximation to M in the operator norm is also attained by $B = \sum_{i=1}^{k} u_i \sigma_i v_i^T$, in which case the error is $\|M - B\|_2 = \sigma_{k+1}$.

Let us give one more interpretation of the singular value decomposition. We can regard an $m \times n$ matrix M as a collection of n data points in $\Re^m$. We associate a distribution Δ with this set of points which chooses a point uniformly at random. Further suppose that the expectation of this distribution is zero. Our data is in high dimension, and a natural question to ask is: how should we project our data onto a one dimensional subspace in a manner that preserves as much information as possible? One concrete goal is to find a direction u so that projecting Δ on u maximizes the variance (among all one-dimensional projections). The question leads to another characterization of the singular vectors:

$$u_1 = \text{argmax} \frac{\|u^T M\|_2}{\|u\|_2}$$

and the maximum is σ_1. Similarly if we want to project onto a two-dimensional subspace so as to maximize the projected variance we should project on $\text{span}(u_1, u_2)$. Relatedly

$$u_2 = \min_{u_1} \text{argmax}_{u \perp u_1} \frac{\|u^T M\|_2}{\|u\|_2}$$

and the maximum is σ_2. This is called the variational characterization of singular vectors. (Here we have assumed the singular values are distinct).

There are efficient algorithms to compute the singular value decomposition. If $n \neq m$ then these algorithms run in time $O(mn^2)$. The basic idea is to reduce M to bidiagonal form using Householder reflections, and then to compute the singular value decomposition from this representation using the QR algorithm. Next we will describe an application to text analysis.

Applications to Text Analysis

Latent Semantic Indexing: [49]

Suppose we are give a large collection of documents, and we would like to extract some hidden structure in this collection (either with the goal of performing information retrieval, or clustering). The usual first step is to collect the data in a very large, very sparse matrix:

Definition 2.1.4 *The term-by-document matrix M is an $m \times n$ matrix where each row represents a word, each column represents a document and the entry in row i, column j is the number of times that word i occurs in document j.*

We have clearly lost some information, since this representation does not take into account the order of the words. However matrices are much easier to work with, and the underlying assumption is that it should still be possible to cluster the documents just knowing what words each one contains but not their order. This is often called the *bag-of-words* assumption.

The idea behind latent semantic indexing is to compute the singular value decomposition of M and use this for information retrieval and clustering. More precisely, if we write

$$M \approx U^{(k)} \Sigma^{(k)} V^{(k)T}$$

where $U^{(k)}$ is the first k columns of U, etc. then the columns of $U^{(k)}$ are the k directions that maximize the projected variance of a random document. These

vectors are interpreted as "topics". More precisely, suppose we want to compute a "similarity" score for document i and document j. We could do this by computing

$$\langle M_i, M_j \rangle$$

where M_i is the i^{th} column of M, etc. This function "counts" the number of words in common. In particular, given a query we would judge how similar a document is to it just be counting how many of its words occur in each document. This is quite naive. Instead, we could compute

$$\langle M_i^T U^{(k)}, M_j^T U^{(k)} \rangle$$

Intuitively this maps each document to a vector of length k that measures how much of each topic is present in the document, and computes the similarly of the documents by taking an inner-product in this low-dimensional space. In practice this is a much better way to measure similarity and was introduced by the seminal paper of Deerwester et al [49].

However it has its own weaknesses. This approach has some rather undesirable properties:

(a) "topics" are orthonormal

Consider topics like "politics" and "finance". Are the sets of words that describe these topics uncorrelated? No!

(b) "topics" contain negative values

This is more subtle, but negative words can be useful to signal that document is not about a given topic. But then when we compute similarly, two documents are judged to be more similar based on a topic that they are both decidedly not about. This is another counter intuitive and undesirable property.

Nonnegative Matrix Factorization

The idea due to [73] and [98] is to write

$$M \approx AW$$

where A and W are $m \times k$ and $k \times n$ respectively and are required to be entry-wise nonnegative. In fact, let us suppose that the columns of M each sum to one. It is not hard to see that if D is a diagonal matrix where the i^{th} entry is the reciprocal

of the sum of the entries in the i^{th} column of A then $M = \widetilde{A}\,\widetilde{W}$ where $\widetilde{A} = AD$ and $\widetilde{W} = D^{-1}W$ normalizes the data so that the columns of $\widetilde{A}$ and of $\widetilde{W}$ each sum to one. Hence we are finding a set of topics (the columns of $\widetilde{A}$ which are each distributions on words) so that every document can be obtained as a convex combination of the topics that we have found.

This optimization problem plays a crucial role in many machine learning systems, such as image segmentation, text analysis, recommendation systems, etc. But this optimization problem is NP-hard [115]. So what should we do now? Give up?

In contrast, singular value decomposition is a problem where theory and practice agree! It can be computed efficiently, and it has many uses. But in spite of this intractability result, nonnegative matrix factorization really is used in practice. The standard approach is to use *alternating minimization*:

Alternating Minimization: This problem is non-convex, but suppose we guess A. Then computing the nonnegative W that minimizes $\|M - AW\|_F$ is convex and can be solved efficiently. The approach is to guess A, compute the best W then set W as fixed and compute the best A, and so on. This process converges, but not necessarily to the optimal solution.

It can and does get stuck in local minima in practice!

We note that this approach is also called *expectation-maximization* [50], and is the standard approach not just for nonnegative matrix factorization, but for many other problems we will study in this course such as *dictionary learning* and *learning mixtures models*.

Food for Thought

But maybe heuristics like this are identifying interesting instances of the problem. The goal of this course is to not give up when faced with intractability, and to look for new explanations. These explanations could be new models (that avoid the aspects of the problem that allow us to embed hard problems) or could be identifying conditions under which heuristics that are already used, do work. This is a largely unexplored area.

In the next section, we will ask what happens if we restrict the number of topics. The instances generated by [115] have k linear in m and n, but when we look for a set of topics that explain $300,000$ New York Times articles, we are looking for only a few hundred topics. So one way to reformulate the question is to ask what its complexity is as a function of k. We will essentially resolve this using algebraic techniques. Nevertheless if we want even better algorithms, we need more

assumptions. We will see how a geometric interpretation of this problem implies that these hard instances are unstable, and we will examine a condition (separability) that enforces stability, and allows us to give much better algorithms - ones that run in time polynomial in all of the parameters.

2.2 Algebraic Algorithms

In the previous section we introduced the nonnegative matrix factorization problem and described its applications to text analysis (it has many other applications). Vavasis proved that this problem is NP-hard in the worst-case, but the instances he contracted have k – the number of topics – linear in the size of the matrix [115]. In most practical applications, k is much smaller than m or n and with this in mind we will instead ask: What is the complexity of this problem as a function of k? We will make use of tools from algebra to give a polynomial time algorithm for any $k = O(1)$. In fact, the algorithm we present here will be nearly optimal in terms of its dependence on k.

Definitions

Let us define the nonnegative matrix factorization problem formally, since we did so only informally in the previous section: Suppose we are given an entry-wise nonnegative matrix M of size $m \times n$.

Definition 2.2.1 *The nonnegative rank of M – denoted by rank$^+(M)$– is the smallest k such that there are nonnegative matrices A and W of size $m \times k$ and $k \times n$ respectively that satisfy $M = AW$.*

Equivalently, rank$^+(M)$ is the smallest k such that there are k nonnegative rank one matrices $\{M_i\}$ that satisfy $M = \sum_i M_i$.

Both of these equivalent formulations of the problem will be useful throughout our discussion. To gain some familiarity with this parameter, it is helpful to compare it to a more familiar one: If we omit the requirement that A and W be entry-wise nonnegative, then the smallest k is precisely the rank of M. Hence the following relation is immediate:

Fact 2.2.2 *rank$^+(M) \geq$ rank(M)*

In fact the rank and the nonnegative rank of a matrix can be quite different:

Example. Let $M \in \mathbb{M}_{n\times n}$, where $M_{ij} = (i-j)^2$. It is easy to see that the columns of M are spanned by

$$\left\{ \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix}, \begin{bmatrix} 1 \\ 2 \\ \vdots \\ n \end{bmatrix}, \begin{bmatrix} 1^2 \\ 2^2 \\ \vdots \\ n^2 \end{bmatrix} \right\}$$

It is easy to see that $\text{rank}(M) = 3$ However, M has zeros along the diagonal and non-zeros off it. Furthermore for any rank one nonnegative matrix M_i, its pattern of zeros and non-zeros is a *combinatorial rectangle* - i.e. the intersection of some set of rows and columns - and a standard argument implies that $\text{rank}^+(M) = \Omega(\log n)$. There are examples with even larger separations too.

Next we will connect nonnegative matrix factorization to computational problems involving systems of polynomial inequalities.

Systems of Polynomial Inequalities

We can reformulate the problem of finding an A and W that prove $\text{rank}^+(M) \leq k$ as a problem of finding a feasible solution to a particular system of polynomial inequalities. More specifically, the problem we want to solve is:

$$\begin{cases} M & = AW \\ A & \geq 0 \\ W & \geq 0 \end{cases} \tag{2.1}$$

This system consists of quadratic equality constraints (one for each entry of M), and linear constraints that A and W be entry-wise nonnegative. Before trying to design better algorithms for $k = O(1)$, we should ask a more basic question (whose answer is not at all obvious):

Question 3 *Is there any finite time algorithm?*

The difficulty is that even if there is a solution, the entries of A and W could be irrational. This is quite different than, say, 3-SAT where there is a simple brute-force algorithm. In contrast for nonnegative matrix factorization it is quite challenging to design algorithms that run in any finite amount of time. But indeed there are algorithms (that run in some fixed amount of time) to decide whether a system of polynomial inequalities has a solution or not in the real RAM model. These algorithms can also compute an implicit representation of the solution, if there is

one. The output is a polynomial and an interval (for each variable) in which there is only one root, which is the value of the variable in the true solution. And you can find as many bits of the solution as you would like by performing binary search for the root.

The first algorithm follows from the seminal work of Tarski, and there has been a long line of improvements based on successively more powerful algebraic decompositions. This line of work culminated in algorithms whose running time is exponential in the number of variables but is polynomial in all the other parameters of the problem (the number of polynomial inequalities, the maximum degree and the bit complexity of the coefficients). The running time is $(nD)^{O(r)}$ where n is the number of polynomial inequalities, D is the maximum degree and r is the number of variables [106]. This running time is essentially optimal under the exponential time hypothesis [78]. In particular, if there is an algorithm for this problem that runs in time $(pD)^{o(r)}$ then it would yield sub-exponential time algorithms for 3-SAT.

We can use these algorithms to solve nonnegative matrix factorization. However the number of variables we would need in the naive representation is $nk + mk$, one for each entry in A or W. So even if $k = O(1)$, we would need a linear number of variables and the running time would be exponential. However we could hope that even though the naive representation uses many variables, perhaps there is a more clever representation that uses many fewer variables. Can we reduce the number of variables in the system of polynomial inequalities from $O(nk + mk)$ to $f(k)$?

If we could do this, then we could solve nonnegative matrix factorization in polynomial time for any $k = O(1)$. Next, we will describe some basic tools in the first-order theory of the reals. These results will help formalize our intuition from above that the number of variables is the right complexity measure when reasoning about how difficult it is to solve a system of polynomial inequalities, but their proof is out of scope of this course.

First-Order Theory of the Reals

Definition 2.2.3 *A set S is semialgebraic if there exist multivariate polynomials $p_1, ..., p_n$ such that*

$$S = \{x_1, ..., x_r | p_i(x_1, ..., x_r) \geq 0\}$$

or if S is a finite union or intersection of such sets.

Definition 2.2.4 *The projection of a semialgebraic set S is defined as*

$$proj_S(X_1, ..., X_\ell) = \{x_1, ..., x_\ell | \exists\, x_{\ell+1}, ..., x_r \text{ such that } p(x_1, ..., x_r) \in S\}$$

Theorem 2.2.5 (Tarski) *The projection of a semialgebraic set is semialgebraic.*

This is one of the foundational results in the field, and is often called *quantifier elimination* [110], [107]. To gain some familiarity with this notion, consider the case of algebraic sets (defined analogously as above, but with polynomial equality constraints instead of inequalities). Indeed, the above theorem implies that the projection of an algebraic set is itself semi-algebraic. Is its projection also algebraic? No (e.g. think about the projection of a circle)!

Earlier, we stated that there are algorithms to solve systems of polynomial inequalities (and find an implicit representation for the solution, if there is one) in time $(nD)^{O(r)}$ where n is the number of polynomial inequalities, D is the maximum degree and r is the number of variables [106]. In fact, these algorithms work in a more general setting where there is additionally a boolean function $\mathbb{B}$ that constraints the sign pattern of the polynomials. We are interested in deciding whether the set

$$S = \{x_1, ..., x_r | \mathbb{B}(p_1(x_1, ..., x_r), ..., p_n(x_1, ..., x_r)) = \text{true}\}$$

is non-empty, and we assume that we can evaluate $\mathbb{B}$ (but not, say, that it has a succinct circuit). A related result is the famous Milnor-Warren bound (see e.g. [7]):

Theorem 2.2.6 (Milnor-Warren) *Given n polynomials $p_1, ..., p_m$ of degree $\leq D$ on r variables $\mathbf{x} = x_1, ...x_r$, consider the sign pattern at $\mathbf{x}$:*

$$\mathbf{x} \to \big(sgn(p_1(\mathbf{x})), sgn(p_2(\mathbf{x})), ..., sgn(p_m(\mathbf{x}))\big)$$

Then as $\mathbf{x}$ ranges over $\mathbb{R}^r$ the number of distinct sign patterns is at most $(nD)^r$.

A priori we could have expected as many as 3^n sign patterns. In fact, algorithms for solving systems of polynomial inequalities are based on cleverly enumerating the set of sign patterns so that the total running time is dominated by the maximum number of distinct sign patterns that there could be! In fact, the Milnor-Warren bound can be thought of as an analogue of the Sauer-Shelah lemma that is used throughout supervised learning where the number of variables plays the role of the VC-dimension.

Next we will give a technique to reduce the number of variables.

Variable Reduction

It is clear that the set of points satisfying (2.1) is a semialgebraic set. However even for $k = 3$ this system has a linear (in n and m) number of variables, so directly solving (2.1) would require exponential time.

Question 4 *Can we find an alternate system of polynomial inequalities that expresses the same decision problem but uses many fewer variables?*

We will focus on a special case called *simplicial factorization* where $\text{rank}(M) = k$. In this case, we are asking whether or not $\text{rank}^+(M) = \text{rank}(M) = k$ and this simplifies matters because of the following observation:

Claim 2.2.7 *In any solution, A and W must have full column and row rank respectively.*

Proof: The span of the columns of A must contain the columns of M and similarly the span of the rows of W must contain the rows of M. Since $\text{rank}(M) = k$ and A and W have k columns and rows respectively we conclude that the A and W must have full column and row rank respectively. Moreover their span must be the column space and row space of M respectively. ■

Hence we know that A and W have left and right pseudo-inverses A^+ and W^+ respectively. We will make use of these pseudo-inverses to reduce the number of variables in our system of polynomial inequalities: We have that $A^+A = I_r$ where I_k is the $k \times k$ identity. Hence

$$A^+AW = W$$

and so we can recover the columns of W from a linear transformation of the columns of M. This leads to the following alternative system of polynomial inequalities:

$$\begin{cases} MW^+A^+M & = M \\ MW^+ & \geq 0 \\ A^+M & \geq 0 \end{cases} \tag{2.2}$$

A priori, it is not clear that we have made progress since this system also has $nk + mk$ variables corresponding to the entries of A^+ and W^+. However consider the matrix A^+M. If we represent A^+ as an $k \times n$ matrix then we are describing its action on all vectors, but the crucial observation is that we only need to know how A^+ acts on the columns of M which span a k dimensional space. Hence we can apply a change of basis to rewrite M as M_R which is an $k \times m$ matrix, and there is an $k \times k$ linear transformation T (obtained from A^+ and the change of basis) so that $TM_R = W$. A similar approach works for W, and hence we get a new system:

$$\begin{cases} M_CSTM_R & = M \\ M_CS & \geq 0 \\ TM_R & \geq 0 \end{cases} \tag{2.3}$$

The variables of this system are the entries in S and T. So there are $2k^2$ variables. And the properties we need of this system are that

(a) If the simplicial factorization problem has a solution, then there is a solution to this system (completeness)

(b) If there is any solution to the system, then the simplicial factorization has a solution (soundness)

We have already proven the first property, and the second property follows because we can set $A = M_C S$ and $W = T M_R$ and this is a valid factorization with inner-dimension k. Hence if we apply Renegar's algorithm to this new system, the algorithm runs in time $(nm)^{O(k^2)}$ and solves the simplicial factorization problem.

The above approach is based on the paper of Arora et al [13] where the authors also give a variable reduction procedure for nonnegative matrix factorization (in the general case where A and W need not have full column or row rank respectively). The authors reduce the number of variables from $(nk + mk)$ to $f(k) = 2k^2 2^k$ and this yields a doubly-exponential time algorithm as a function of k. The crucial observation is that even if A does not have full column rank, we could write a system of polynomial inequalities that has a pseudo-inverse for each set of its columns that is full rank (and similarly for W). However A could have as many as $\binom{k}{k/2}$ maximal sets of linearly independent columns, and hence the resulting system of polynomial inequalities has $f(k)$ variables but $f(k)$ is itself exponential in k.

In [94] the author further reduces the number of variables to $2k^2$ for nonnegative matrix factorization, and the main idea is that even though A could have exponentially many maximal sets of linearly independent columns, their psueudo-inverses are algebraically dependent and can be expressed over a common set of k^2 variables using Cramer's rule. This yields a singly exponential time algorithm for nonnegative matrix factorization that runs in $(nm)^{O(k^2)}$ time which is essentially optimal since any algorithm that runs in time $(nm)^{o(k)}$ would yield a sub-exponential time algorithm for 3-SAT [13].

2.3 Stability and Separability

In the previous section we took an algebraic approach and here instead we will work with an assumption called *separability* [54] which will allow us to give an algorithm that runs in polynomial time (even for large values of r). Our discussion will revolve around the intermediate simplex problem.

Intermediate Simplex Problem

Let us define the intermediate simplex problem:

> We are given two polytopes Q and P with $P \subseteq Q$ and furthermore P is encoded by its vertices and Q is encoded by its facets. Is there a simplex K with $P \subseteq K \subseteq Q$?

We would like to connect this problem to nonnegative matrix factorization, since it will help us build up a geometric view of the problem. Consider the following problem:

> Given nonnegative matrices M and A, does there exists $W \geq 0$ such that $M = AW$?

The answer is "Yes", if and only if each column of M is in the cone spanned by nonnegative combinations of the columns of A. Moreover if we normalize the columns of M and A so that they sum to one, then the answer is "Yes" if and only if the convex hull of the columns of A contains the columns of M. Recall in simplicial factorization we are given a nonnegative matrix M with $\text{rank}(M) = k$, and our goal is to decide whether or not $\text{rank}^+(M) = k$. We will prove that the simplicial factorization problem and the intermediate simplex problem are equivalent [115]. Consider the following helper problem, which we call **(P0):**

> Given $M = UV$, is there an invertible $k \times k$ matrix T such that UT^{-1}, and TV are nonnegative?

In fact, Vavasis [115] proved that **(P0)**, intermediate simplex and the simplicial factorization problem are each polynomial time interreducible. It is easy to see that **(P0)** and the simplicial factorization problem are equivalent since in any two factorizations $M = UV$ or $M = AW$ (where the inner-dimension equals the rank of M), the column spaces of M, U and A are identical. Similarly the rows spaces of M, V and W are also identical.

The more interesting aspect of the proof is the equivalence between **(P0)** and the intermediate simplex problem. The translation is:

(a) rows of $U \iff$ vertices of P

(b) rows of $T \iff$ vertices of K

(c) columns of $V \iff$ facets of Q

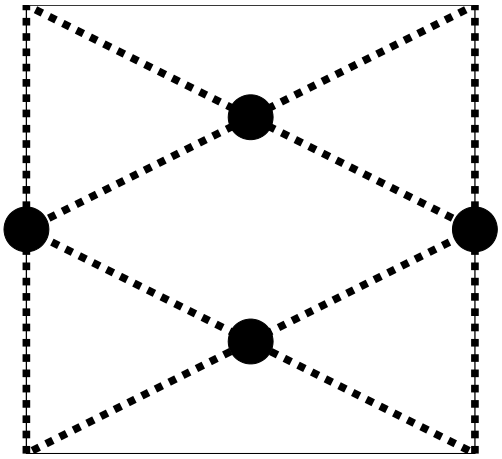

Figure 2.1: This figure is taken from [115]. The intermediate simplex problem has two solutions which will be used to encode the truth assignment of a variable.

Then the constraint that UT^{-1} is nonnegative is (roughly) the constraint that $P \subseteq K$ and the constraint TV is (roughly) the constraint $K \subseteq Q$. There are some tedious normalization issues that arise since we need to be careful about the distinction between the convex hull of a set of vectors and the cone generated by all nonnegative combinations. However this equivalence gives us a geometric view that will be helpful.

Vavasis made use of the equivalences in the previous subsection to prove that nonnegative matrix factorization is NP-hard. Consider the gadget in Figure 2.1; the crucial property is that there are only two possible intermediate triangles, which can then be used to represent the truth assignment for a variable x_i. The description of the complete reduction, and the proof of its soundness are involved (see [115]).

The trouble is that gadgets like those in Figure **??** are *unstable. We can change the number of solutions by small perturbations to the problem.* Motivated by issues of uniqueness and robustness, Donoho and Stodden [54] introduced a condition called *separability* that alleviates many of these problems, which we will discuss in the next subsection.

Separability

Definition 2.3.1 *We call A* separable *if, for every column of A, there exists a row of A whose only non-zero entry is in that column.*

Furthermore in the separable nonnegative matrix factorization problem we are given M and the promise that if there is a nonnegative matrix factorization, there is one in which A is separable. Donoho and Stodden used this condition (and others) to show that there are somewhat natural conditions under which the nonnegative matrix factorization is unique. Arora, Ge, Kannan and Moitra gave an algorithm for finding it:

Theorem 2.3.2 *[13] Given a nonnegative matrix M with the promise that there is a nonnegative matrix factorization $M = AW$ where A is separable, there is a polynomial time algorithm to compute such a factorization of minimum inner-dimension.*

In fact, separability is quite natural in the context of text analysis. Recall that we interpret the columns of A as topics. We can think of separability as the promise that these topics come with *anchor words*; informally, for each topic there is an unknown anchor word that if it occurs in a document, the document is (partially) about the given topic. For example, *401k* could be an anchor word for the topic *personal finance*.

Why do anchor words help? It is easy to see that if A is separable, then the rows of W appear as rows of M (after scaling). Hence we just need to determine which rows of M correspond to anchor words. We know from our discussion in Section 2.3 that (if we scale M, A and W so that their rows sum to one) the convex hull of the rows of W contain the rows of M. But since these rows appear in M as well, we can try to find W by iteratively deleting rows of M that do not change its convex hull.

Let M^i denote the ith row of M and let M^I denote the restriction of M to the rows in I for $I \subseteq [n]$. So now we can find the anchor words using the following simple procedure:

Find Anchors [13]
Input: matrix $M \in \mathbb{R}^{n \times m}$ satisfying the conditions in Theorem 2.3.2
Output: $W = M^I$

Set $I = [n]$
For $i = 1, 2, ..., n$
    If $M^i \in \text{conv}(\{M^j | j \in I, j \neq i\})$, set $I \leftarrow I - \{i\}$
End

It is easy to see that deleting a row of M that is not an anchor word will not change the convex hull of the remaining rows, and so the above algorithm terminates

with a set I that only contains anchor words. Moreover at termination

$$\text{conv}(\{M^i | i \in I\}) = \text{conv}(\{M^j\}_j)$$

Alternatively the convex hull is the same as at the start. Hence the anchor words that are deleted are redundant and we could just as well do without them.

Separable NMF [13]
Input: matrix $M \in \mathbb{R}^{n \times m}$ satisfying the conditions in Theorem 2.3.2
Output: A, W

Run **Find Anchors** on M, let W be the output
Solve for nonnegative A that minimizes $\|M - AW\|_F$ (convex programming)
End

The proof of theorem follows immediately from the proof of correctness of **Find Anchors** and the fact that $\text{conv}(\{M^i\}_i) \subseteq \text{conv}(\{W^i\}_i)$ if and only if there is a nonnegative A (whose rows sum to one) with $M = AW$.

The above algorithm when naively implemented would be prohibitively slow. Instead, there have been many improvements to the above algorithm [27], [84] [65], and we will describe one in particular that appears in [12]. Suppose we choose a row M^i at random. Then it is easy to see that the furthest row from M^i will be an anchor word.

Similarly, if we have found one anchor word the furthest row from it will be another anchor word, and so on. In this way we can greedily find all of the anchor rows, and moreover this method only relies on pair-wise distances and projection so we can apply dimension reduction before running this greedy algorithm. This avoids linear programming altogether in the first step in the above algorithm, and the second step can also be implemented quickly because it involves projecting a point into an $k-1$-dimensional simplex.

2.4 Topic Models

Here we will consider a related problem called *topic modeling*; see [28] for a comprehensive introduction. This problem is intimately related to nonnegative matrix factorization, with two crucial differences. Again there is some factorization $M = AW$ but now we do not get access to M but rather $\widetilde{M}$ which is a very crude approximation. Intuitively, each column in M is a document that is itself a distribution on

words. But now the words that we observe are samples from this distribution (so we do not actually know the columns of M).

The second difference is that we think of W as stochastically generated. There are in fact many popular choices for this distribution:

(a) **Pure Documents:** Each document is about only one topic, hence each column of W has exactly one non-zero.

(b) **Latent Dirichlet Allocation [30] :** The columns of W are generated from a Dirichlet distribution.

(c) **Correlated Topic Model [29] :** Certain pairs of topics are allowed to be positively or negatively correlated, and the precise distribution that generates the columns of W is log-normal.

(d) **Pachinko Allocation Model [89] :** This is a multi-level generalization of LDA that also allows for certain types of structured correlations.

There are many more choices. Regardless, our goal is to learn A from $\widetilde{M}$. To emphasize the differences, note that *even if we knew* A we cannot compute W exactly. Alternatively, $\widetilde{M}$ and M can be quite different since the former may be sparse while the latter is dense. Are there provable algorithms for topic modeling?

The Gram Matrix

We will follow an approach of Arora, Ge and Moitra [14]. At first this seems like a fundamentally different problem than the ones we have considered because in this model we cannot ask for longer documents, we can only ask for more of them. Hence we are increasing the number of columns of $\widetilde{M}$ but each column is not that close to the corresponding column in M. The basic idea is to work instead with the *Gram matrix* G:

Definition 2.4.1 *Let G denote the word $\times$ word matrix whose entry in (a, b) is the probability that the first two words in a randomly chosen document are a and b respectively.*

Definition 2.4.2 *Let R denote the topic $\times$ topic matrix whose entry in (i, j) is the probability that the first two words (again in a randomly chosen document) are generated from the topics i and j respectively.*

Note that we can approximate G from our samples, however we cannot (directly) approximate R and it is controlled by the choice of which distribution we use to generate the columns of W. More precisely:

Lemma 2.4.3 $G = ARA^T$

Proof: Let w_1 denote the first word and let t_1 denote the topic of w_1 (and similarly for w_2). We can expand $\mathbb{P}[w_1 = a, w_2 = b]$ as:

$$\sum_{i,j} \mathbb{P}[w_1 = a, w_2 = b | t_1 = i, t_2 = j] \mathbb{P}[t_1 = i, t_2 = j]$$

and the lemma is now immediate. ■

The key observation is that G has a separable nonnegative matrix factorization given by A and RA^T since A is separable and the latter matrix is nonnegative. Indeed if RA^T has full row rank then the algorithm in Theorem 2.3.2 will find the true set of anchor words. However since the rows of RA^T are no longer normalized to sum to one, the above factorization is not necessarily unique. Nevertheless we have made some progress, and we can adopt a Bayesian interpretation (see [12]).

Recovery via Bayes Rule

In fact, the entries of A are conditional probabilities $\mathbb{P}(w_1|t_1)$ and so we can reason about the posterior distribution $\mathbb{P}(t_1|w_1)$. In fact this gives us an alternate characterization of an anchor word: A word is an anchor word if and only if its posterior distribution is supported on just one topic. In particular

$$\mathbb{P}(t_1 = t | w_1 = w) = \begin{cases} 1, & w \text{ is an anchor word for } t, \\ 0, & \text{otherwise,} \end{cases}$$

Now we can expand:

$$P(w_1 = w' | w_2 = w) = \sum_t \mathbb{P}(w_1 = w' | w_2 = w,\, t_2 = t) \cdot \mathbb{P}(t_2 = t | w_2 = w),$$

In fact w_1 is independent of w_2 if we condition on t_2 and so:

$$\begin{aligned} \mathbb{P}(w_1 = w' | w_2 = w,\, t_2 = t) &= \mathbb{P}(\text{word1} = w' | \text{topic2} = t) \\ &= \mathbb{P}(\text{word1} = w' | \text{word2} = \text{anchor}(t)), \end{aligned}$$

which we can compute from G after having determined the anchor words. Hence:

$$\mathbb{P}(w_1 = w'|w_2 = w) = \sum_t \mathbb{P}(\text{word1} = w'|\text{word2} = \text{anchor}(t))\mathbb{P}(t_2 = t|w_2 = w)$$

which we can think of a linear systems in the variables $\{\mathbb{P}(t_2 = t|w_2 = w)\}$. It is not hard to see that if R has full rank then it has a unique solution. Finally, we compute the probabilities we were originally interested in by Bayes' rule:

$$\begin{aligned}\mathbb{P}(\text{word } w|\text{topic } t) &= \frac{\mathbb{P}(\text{topic } t|\text{word } w) \cdot \mathbb{P}(\text{word } w)}{\mathbb{P}(\text{topic } t)} \\ &= \frac{\mathbb{P}(\text{topic } t|\text{word } w) \cdot \mathbb{P}(\text{word } w)}{\sum_{w'} \mathbb{P}(\text{topic } t|\text{word } w') \cdot \mathbb{P}(\text{word } w')}.\end{aligned}$$

We can now state the algorithm **Recover**. Let $\widetilde{G}$ be the empirical Gram matrix, where $\widetilde{G}_{a,b}$ is the fraction of documents in our sample whose first word is a and whose second word is b.

Suppose each anchor word has probability at least p. Then the main result in this subsection is:

Theorem 2.4.4 *[14] For any separable topic model where R is full rank there is a polynomial time algorithm to compute $\widetilde{A}$ that is ε-close to A and the running time and sample complexity (number of documents) is poly$(n, 1/p, 1/\varepsilon, 1/\sigma_{min}(R))$, provided documents have length at least two.*

In the next subsection we describe some experimental results.

Recover [14], [12]
Input: term-by-document matrix $M \in \mathbb{R}^{n \times m}$
Output: A, R

Compute $\widetilde{G}$, compute $\mathbb{P}(w_1 = w|w_2 = w')$
Run **Find Anchors**
Solve for $\mathbb{P}(\text{topic } t|\text{word } w)$ and use Bayes' rule to compute A
End

Experiments

We are faced with a basic *scientific* question now: Are there really anchor words? The following experiment was conducted in [12]:

(a) Run MALLET (a popular topic modeling toolkit) on a collection of New York Times articles, its output is a topic matrix A.

(b) Use A to generate data from a topic model, run MALLET on this data.

The important point is that here the data that we are running on is actually from a topic model and we can compare how well one algorithm can recover the true matrix compared to how well another algorithm does. Then:

(c) Run the new algorithm on this data.

This is a seemingly unfair comparison, since we have restricted ourselves to a topic matrix A that MALLET has already found once (so this is our notion of what constitutes a realistic topic model). Yet surprisingly the algorithm in the previous subsection was able to find the topic matrix A more accurately and orders of magnitude faster! *This is an important example where finding conditions under which we can give provable algorithms indeed led to much better algorithms in practice.*

18.409 Algorithmic Aspects of Machine Learning

Spring 2015