Random Projections for Anchor-based Topic Inference

David Mimno
Department of Information Science
Cornell University
Ithaca, NY 14850
mimno@cornell.edu

Abstract

Recent spectral topic discovery methods are extremely fast at processing large document corpora, but scale poorly with the size of the input vocabulary. Random projections are vital to ensure speed and limit memory usage. We empirically evaluate several methods for generating random projections and measure the effect of parameters such as sparsity and dimensionality. We find that methods with structured sparsity are faster than Gaussian random projections and more accurate than standard sparse random projections.

1 Introduction

Recent advances in spectral inference offer vast improvements in scalability for unsupervised models such as non-negative matrix factorization [1] and latent Dirichlet allocation [2]. Rather than maximizing the likelihood of individual observations, these algorithms use the method of moments to match second-order statistics that summarize a corpus. In the case of a topic model trained from a corpus of text documents, these statistics consist of a square matrix of word-word co-occurrence probabilities, whose size is the square of the number of words in the vocabulary $|\mathcal{V}|$. Spectral algorithms avoid iterating more than once through the training corpus, so they are less sensitive to the number of tokens than likelihood-based algorithms, but the size of the vocabulary quickly becomes a limiting factor. As corpus sizes increase, the size of the vocabulary also increases. Unless we reduce dimensionality, the matrix of sufficient statistics will grow quadratically.

In this paper we evaluate several methods for reducing dimensionality by generating random projections of the matrix of sufficient statistics. There are a number of methods for sampling random projection matrices, including Gaussian matrices [3] and sparse discrete matrices [4]. These methods have parameters, such as the number of random projections and the sparsity of those projections. Although theoretical bounds are known, it is also important to measure how these models and their parameters are affected by the specific characteristics of word co-occurrence data, and how those differences impact the output of a spectral topic model algorithm. We evaluate their computational efficiency, their accuracy in preserving ℓ_2 distances, and their effectiveness at reconstructing topic models. We find that two methods that use structured sparsity offer both efficiency and accuracy, and that standard Achlioptas-style sparse random projections become inaccurate as we increase sparsity.

2 Topic Models

Topic models such as PLSI [5] or its Bayesian equivalent Latent Dirichlet allocation [6] represent a corpus as a mixture of K themes or "topics," each represented by a probability distribution ϕ_k over the vocabulary $\mathcal{V}$. We represent each of the D documents in the corpus with a probability distribution θ_d over these topics. Training a topic model involves inferring values for these distributions from unlabeled document corpora. Given a model, users can quickly assess the contents of vast document collections, browse through thematically related documents, and trace patterns of topic prevalence across time and other variables.

3 Spectral inference

Recent spectral algorithms reduce topic model training to operations on a large square matrix of sufficient statistics. The anchor-based spectral algorithm of Arora et al. [2] rests on the assumption that each topic has an "anchor word" that is unique to that topic. We assume that each topic will contain at least one unambiguous "anchor" word that has non-negligible probability only in that topic. The presence of the kth anchor word in a document is sufficient (though not necessary) to indicate that the document uses topic k.

The algorithm proceeds in two steps. In the first step, anchor selection, we identify one anchor word for each topic. In the second step, topic recovery, we learn the distribution of words given topics $\phi_1, ..., \phi_K$ by first estimating the probability of topics given words, and then using the marginal probability of each word to exchange the conditioning variables using Bayes rule. For the purposes of this work we focus on the first step, anchor word discovery.

The anchor words algorithm factorizes a matrix of sufficient statistics that represent word-word conditional probabilities. We begin by estimating the probability that each pair of words will occur together in the same document. For each document in the training set, we estimate the probability that two randomly chosen tokens from that document will be of each given pair of word types. Averaging over all documents, we define a matrix Q, such that

$$Q_{ij} = \frac{1}{D} \sum_d \frac{n_{di} n_{dj}}{\frac{1}{2} N_d (N_d - 1)}, \tag{1}$$

where n_{di} is the number of times word i occurs in document d, and N_d is the length of the document. For a vocabulary of size V, Q has size $V \times V$. We set $Q_{ii} = 0$ for all i.

We then row-normalize Q to define a matrix $\bar{Q}$ such that $\bar{Q}_{ij} = P(w_j | w_i)$, the probability that if we select a random token of type i from the corpus, a randomly selected word from the same document as that token will be of type j. We can consider each row in this matrix a point in V-dimensional space, which represents the probability that each other word will occur in the same context as the word for that row.

It can be shown that the rows of $\bar{Q}$ corresponding to anchor words are the corners of a simplex in this word co-occurrence space [2]. We can therefore identify K anchor words with a Gram-Schmidt orthogonalization. In the topic recovery step, we represent the points associated with each of the remaining $V - K$ words as convex combinations of the K anchor-word points. As each anchor word corresponds to a single topic, the weights generated for each non-anchor word in the topic recovery step can be interpreted as the probability of the topics given the feature. The efficiency of both processes depend on the size of the $\bar{Q}$ matrix.

4 Random projections

Constructing the full matrix of conditional probabilities for a large vocabulary would be beyond the memory capabilities of most contemporary computers, but we can substitute a smaller matrix generated with random projections. The anchor words algorithm theoretically takes as input a $V \times V$ matrix of conditional probabilities, which grows quadratically with the size of the vocabulary. Although the matrix is likely to be sparse, there are still large numbers of interactions between words in every document.

Random projections allow us to project the rows of $\bar{Q}$ to a lower-dimensional space while maintaining ℓ_2 distances between points [7]. Let P be a $V \times S$ projection matrix, where $S \ll V$. If the elements of P are drawn from a distribution with mean zero and unit variance, then with high probability, the distance between rows of the matrix $\frac{1}{\sqrt{S}} QP$ will be close to the distances between rows of the original matrix.

We consider four possible distributions for P. Examples of each are shown in Figure 1.

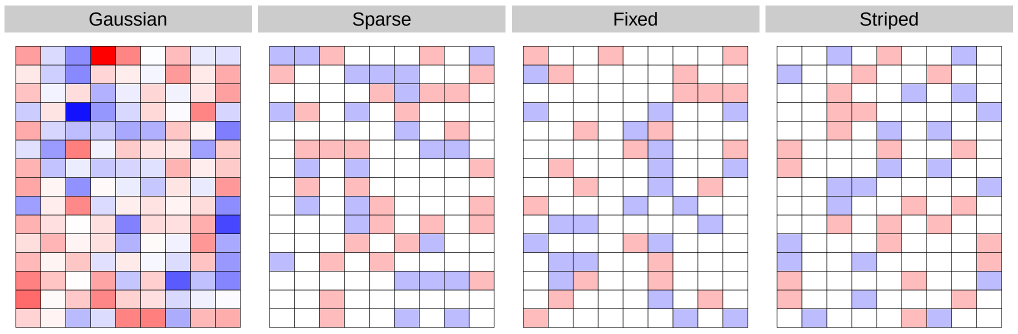

Figure 1: Four sample projection matrices. The *Gaussian* matrix is dense, with different values in each cell. The *Sparse* matrix has between one and five non-zeros per row. The *Fixed* matrix has exactly three non-zeros per row, which can appear in any column. The *Striped* matrix has, for each row, exactly one non-zero in each three-column block.

1. **Gaussian**. A common choice for the elements of P are i.i.d. standard normal variates [3]. This distribution results in a dense projection matrix.
2. **Sparse**. An alternative is to use *sparse* random projections [4, 8]. The simplest sparse distribution is to draw values i.i.d. from a discrete distribution over the values $\{-\frac{1}{\sqrt{s}}, 0, \frac{1}{\sqrt{s}}\}$ with probabilities $\frac{s}{2}, 1-s, \frac{s}{2}$, where s represents the probability of a non-zero value. The expected number of non-zeros per row is sS.
3. **Fixed-row Sparse**. We evaluate two additional random projection schemes that keep the same sparse marginal distribution but add structure to the selection of non-zero elements [9]. The *fixed* scheme enforces the constraint that all rows of P must contain a fixed number of non-zeros, $\lfloor sS \rfloor$. The arrangement of non-zeros is uniform over the $\binom{S}{\lfloor sS \rfloor}$ possible configurations.
4. **Striped Sparse** The *striped* scheme divides the columns of P into $\lfloor sS \rfloor$ groups. Each row contains exactly one non-zero in each group, selected uniformly at random.

An important advantage of a random projection is that it allows us to avoid ever constructing the full Q matrix. We want to calculate QP. The unnormalized term for each document is

$$\frac{1}{2}N_d(N_d-1)Q_d = \boldsymbol{n}_d\boldsymbol{n}_d^T P. \tag{2}$$

We can evaluate $\boldsymbol{n}_d^T P$ in $O(SV)$ and then multiply $\boldsymbol{n}_d$ by the resulting S vector, avoiding any $O(V^2)$ computation. Note that in order to generate an approximation to $\bar{Q}$ we must divide by the sum of the (unprojected) rows of Q. For row i this quantity is equal to $\sum_d \frac{N_d(N_d-1)}{2} n_{di}(N_d - n_{di})$.

5 Evaluation

In this section we measure the empirical performance of different random projections on three real datasets. We evaluate speed, accuracy of ℓ_2 distances, and stability of anchor word selection.

Several factors affect the computational efficiency of a random projection. We would prefer to use as few random projections as possible, but results may become excessively noisy if we compress too much. Dense random matrices, as with the Gaussian distribution, are the most expensive, because they involve sampling $V \times S$ elements, storing those random variates, and multiplying each of the S elements in a row for every distinct word in a document. Sparse matrices are easier to create, store, and multiply, but they add an additional parameter: the degree of sparsity. As with the number of projections, we would prefer a sparser matrix, but it is important to understand the consequences of a given level of sparsity on our performance metric.

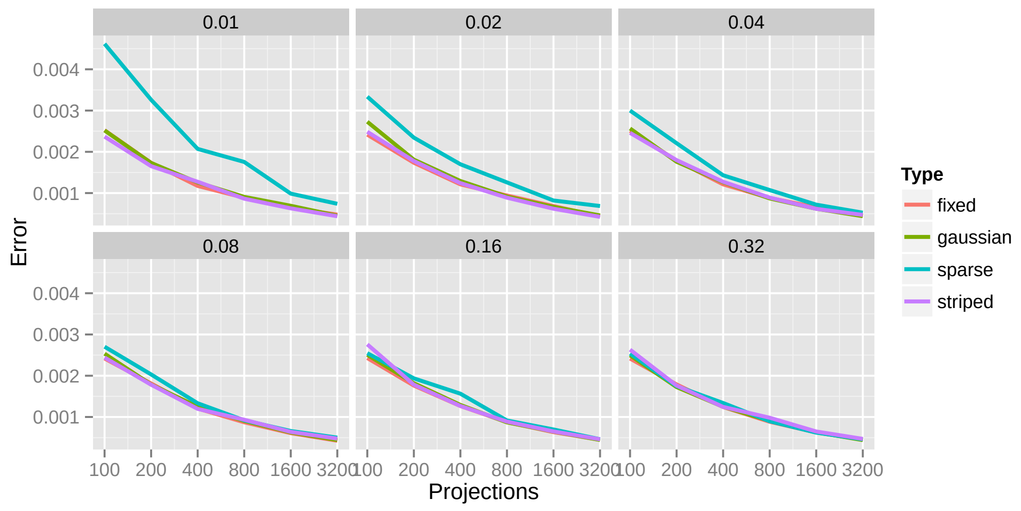

Figure 2: Average ℓ_1 error between unprojected and projected ℓ_2 distances. Panels indicate varying degrees of sparsity, from 1% up to 32% non-zero elements. All methods improve with increasing numbers of random projections. Values for the *fixed*, *striped* and *gaussian* methods are nearly identical. The unstructured *sparse* method is sensitive to the degree of sparsity, while the other methods are relatively insensitive.

We measured performance on three real-world document datasets, a corpus of political blog posts, a corpus of articles from the journal Nature, and a corpus of New York Time articles. In order to compare to unprojected data, we restricted the vocabulary to frequently occurring words, resulting in a vocabulary size of around 4000 for each dataset.

Error. How we sample a projection matrix affects the error in distances between points. If we choose a good projection matrix, ℓ_2 distances between projected points should be within ϵ of the distances between those points in the original $\bar{Q}$ matrix. For each method, we generate random matrices with $S \in \{100, 200, 400, 800, 1600, 3200\}$. For the sparse methods, we additionally use sparsity values varying between 1% and 32% (represented as 0.01 and 0.32, respectively, in the figures). We sampled 10000 random pairs of points and measured the ℓ_2 distance between those points in the unprojected $\bar{Q}$ matrix and the equivalent projected points. We then calculate the average absolute difference between these distances over all sampled pairs. Results for the NYT corpus are shown in Figure 2. Results were similar for the blog and Nature corpora (not shown).

Results for the methods are similar except for the *sparse* method, which has greater error as sparsity increases. The methods with structured sparsity (*fixed* and *striped*) are stable even at 1% non-zero elements: at 100 random projections, average ℓ_1 error for the *sparse* method is 0.0046 at 1% non-zeros and 0.0025 for 32%, while all other methods are close to 0.0025 at all levels of sparsity.

Anchor finding. The quality of a random projection affects the anchor word selection algorithm. In the previous experiment we measured errors in maintaining ℓ_2 distances. The *sparse* method shows larger error when we reduce the number of non-zeros. To determine whether that difference in error is meaningful, we compare the results of the anchor word selection algorithm when we apply different random projection matrices.

For each corpus, we run Gram-Schmidt orthogonalization on the full $\bar{Q}$ matrix to select $K = 100$ anchor words. We use this set of words as a reference. We then sample random projection matrices using each method and parameter setting, with five random initializations each. For each projection of the $\bar{Q}$ matrix we run the anchor word selection algorithm, and report the size of the intersection between the selected set and the reference set. Text data is noisy, and real documents do not actually follow our modeling assumptions, so we do not expect to get 100 out of 100 every time. Neverthe-

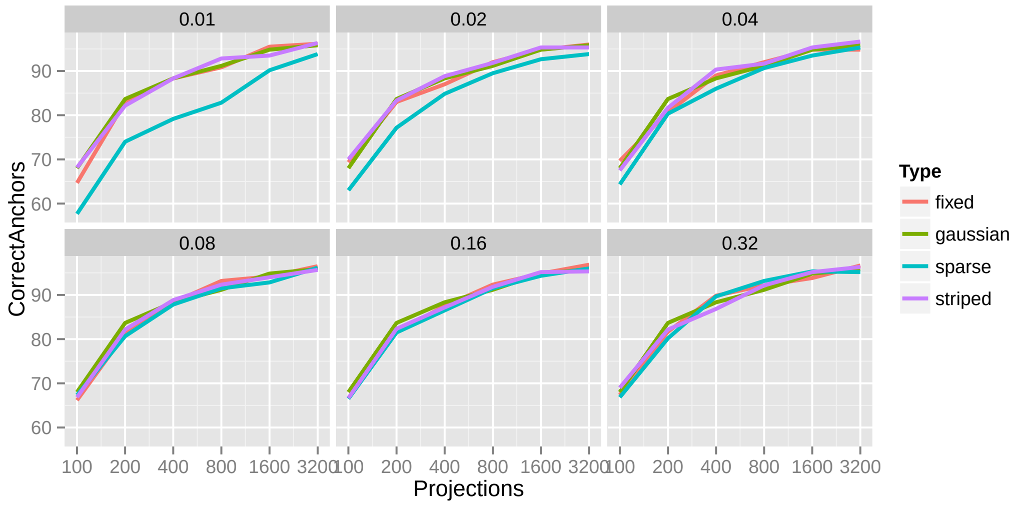

Figure 3: Average overlap between 100 anchor words selected based on the unprojected matrix and each projected matrix. All methods improve with increasing numbers of random projections. The unstructured *sparse* method is sensitive to the degree of sparsity, while the other methods are relatively insensitive.

less, we prefer a random projection method that produces results closer to what we would get from the original, unprojected matrix.

Results for the NYT corpus are shown in Figure 3, and indicate the same trends shown in the earlier experiment. Results for the other two corpora were, again, similar. The number of correct anchor words increases as we add more random projections. The *fixed* and *striped* methods are able to handle high sparsity with little noticeable affect on performance. In contrast, the *sparse* method shows a significant drop in anchor word selection quality as we decrease the number of non-zero elements. There is a slight decrease in the performance of the *striped* method at 32% sparsity. We speculate that as the number of non-zero elements increases, the more rigidly structured matrices are less able to differentiate between rows.

Speed. Methods with structured sparsity offer improved computational efficiency. Although we generally prefer more accurate methods, for practical applications it is necessary to balance accuracy with speed. In this experiment we measure the effects of methods and parameter settings on the time needed to generate a projection of the matrix of sufficient statistics from documents.

Results are shown in Figure 4 for the *gaussian* and *fixed* methods. Timing results for the *sparse* and *striped* methods are equivalent to the previous two methods, respectively, and are not shown for clarity. As we increase the number of random projections, we linearly increase the time to generate the projected matrix. The *fixed* method, because it has a pre-set number of non-zero elements in each row, can be represented and iterated over through the sparse indices of the non-zero elements. When the random matrix is very sparse, the *fixed* method is roughly twice as fast as the dense Gaussian method, but as the number of non-zero elements increases past 8%, the dense method becomes more efficient. Of course, these numbers are subject to the details of the implementation, and may be improved in the future.

In addition to speed, sparse random projections also provide savings in memory. To store a Gaussian matrix we require VS floating point variables. In contrast, to store the matrix for a *fixed* or *striped* random projection with 1% non-zeros, we require only $0.01VS$ integer variables to record the index of the non-zeros and their signs.

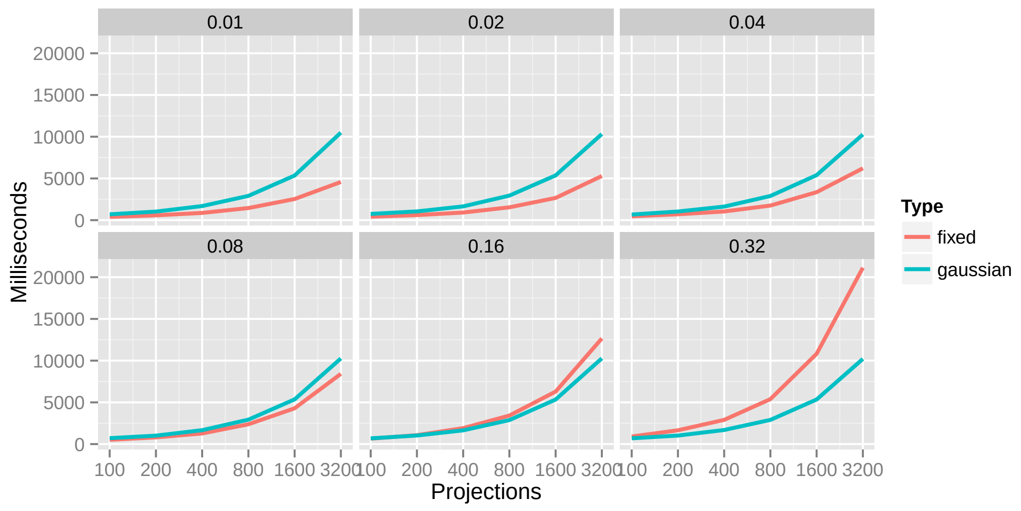

Figure 4: Time to generate a projection of the sufficient statistics for a corpus. The *fixed* method is faster at high sparsity (0.01), but bookkeeping overhead results in slower performance as the number of non-zero elements increases (0.32).

6 Conclusions

Random projections enable us to scale anchor-based topic model algorithms to large vocabularies. The choice of the distribution we use to sample random projection matrices and the parameters for those methods have a significant effect on both the accuracy and efficiency of the resulting projected matrix. We recommend using either of the two methods that use structured sparsity (*fixed* and *striped*). Unlike standard sparse projections, these methods produce good results for the text-based data used in topic inference, although it is possible that additional term reweighting as used by Li et al. [8] might help. We find empirically that accuracy is substantially improved by using at least 800–1000 random projections, but that these projections can be extremely sparse without affecting performance.

References

[1] Sanjeev Arora, Rong Ge, Ravindran Kannan, and Ankur Moitra. Computing a nonnegative matrix factorization – provably. In *STOC*, 2012.

[2] Sanjeev Arora, Rong Ge, Yonatan Halpern, David Mimno, Ankur Moitra, David Sontag, Yichen Wu, and Michael Zhu. A practical algorithm for topic modeling with provable guarantees. In *ICML*, 2013.

[3] Sanjoy Dasgupta. Experiments with random projection. In *UAI*, 2000.

[4] Dimitris Achlioptas. Database-friendly random projections. In *SIGMOD*, pages 274–281, 2001.

[5] Thomas Hofmann. Probabilistic latent semantic analysis. In *UAI*, 1999.

[6] David Blei, Andrew Ng, and Michael Jordan. Latent Dirichlet allocation. *JMLR*, 2003.

[7] William B. Johnson and Joram Lindenstrauss. Extensions of lipschitz mappings into a hilbert space. *Contemporary Mathematics*, 26:189–206, 1984.

[8] Ping Li, Trevor J. Hastie, and Kenneth W. Church. Very sparse random projections. In *KDD*, 2006.

[9] Daniel M. Kane and Jelani Nelson. Sparser johnson-lindenstrauss transforms. In *SODA*, 2012.