9 Topic Models, Nonnegative Matrix Factorization, Hidden Markov Models, and Graphical Models

In the chapter on machine learning, we saw many algorithms for fitting functions to data. For example, suppose we want to learn a rule to distinguish spam from nonspam email and we were able to represent email messages as points in R^d such that the two categories are linearly separable. Then, we could run the Perceptron algorithm to find a linear separator that correctly partitions our training data. Furthermore, we could argue that if our training sample was large enough, then with high probability, this translates to high accuracy on future data coming from the same probability distribution. An interesting point to note here is that these algorithms did not aim to explicitly learn a model of the distribution D^+ of spam emails or the distribution D^- of nonspam emails. Instead, they aimed to learn a separator to distinguish spam from nonspam. In this chapter, we look at algorithms that, in contrast, aim to explicitly learn a probabilistic model of the process used to generate the observed data. This is a more challenging problem, and typically requires making additional assumptions about the generative process. For example, in the chapter on high-dimensional space, we assumed data came from a Gaussian distribution and we learned the parameters of the distribution. In the chapter on SVD, we considered the more challenging case that data comes from a mixture of k Gaussian distributions. For $k = 2$, this is similar to the spam detection problem, but harder in that we are not told which training emails are spam and which are nonspam, but easier in that we assume D^+ and D^- are Gaussian distributions. In this chapter, we examine other important model-fitting problems, where we assume a specific type of process is used to generate data, and then aim to learn the parameters of this process from observations.

9.1 Topic Models

Topic Modeling is the problem of fitting a certain type of stochastic model to a given collection of documents. The model assumes there exist r "topics", that each document is a mixture of these topics, and that the topic mixture of a given document determines the probabilities of different words appearing in the document. For a collection of news articles, the topics may be politics, sports, science, etc. A topic is a set of word frequencies. For the topic of politics, words like "president" and "election" may have high frequencies, whereas for the topic of sports, words like "pitcher" and "goal" may have high frequencies. A document (news item) may be 60% politics and 40% sports. In that case, the word frequencies in the document are assumed to be convex combinations of word frequencies for each of these topics with weights 0.6 and 0.4 respectively.

Each document is viewed as a "bag of words" or *terms*.[37]. Namely, we disregard the order and context in which each word occurs in the document and instead only list the frequency of occurrences of each word. Frequency is the number of occurrences of the

[37]In practice, terms are typically words or phrases, and not all words are chosen as terms. For example, articles and simple verbs, pronouns etc. may not be considered terms.

word divided by the total count of all words in the document. Throwing away context information may seem wasteful, but this approach works fairly well in practice. Each document is a vector with d components where d is the total number of different terms that exist; each component of the vector is the frequency of a particular term in the document.

We can represent a collection of n documents by a $d \times n$ matrix A called the *term-document* matrix, with one column per document and one row per term. The topic model hypothesizes that there exist r topics (r is typically small) such that each document is a mixture of these topics. In particular, each document has an associated vector with r nonnegative components summing to one, telling us the fraction of the document that is on each of the topics. In the example above, this vector would have 0.6 in the component for politics and 0.4 in the component for sports. These can be arranged vectors as the columns of a $r \times n$ matrix C, called the *topic-document* matrix. Finally, there is a third $d \times r$ matrix B for the topics. Each column of B is a vector corresponding to one topic; it is the vector of expected frequencies of terms in that topic. The vector of expected frequencies for a document is a convex combination of the expected frequencies for topics, with the topic weights given by the vector in C for that document. In matrix notation, let P be a $n \times d$ matrix with column $P(:, j)$ denoting the expected frequencies of terms in document j. Then,

$$P = BC. \tag{9.1}$$

Pictorially, we can represent this as:

$$\begin{array}{c} \\ \text{TERM} \end{array} \overset{\text{DOCUMENT}}{\begin{pmatrix} & & \\ & P & \\ & & \end{pmatrix}} = \begin{array}{c} \text{TERM} \\ \\ \end{array} \overset{\text{TOPIC}}{\begin{pmatrix} & & \\ & B & \\ & & \end{pmatrix}} \begin{array}{c} \text{TOPIC} \\ \\ \end{array} \overset{\text{DOCUMENT}}{\begin{pmatrix} & & \\ & C & \\ & & \end{pmatrix}}$$

Topic Models are stochastic models that generate documents according to the frequency matrix P above. p_{ij} is viewed as the probability that a random term of document j is the i^{th} term in the dictionary. We make the assumption that terms in a document are drawn independently. In general, B is assumed to be a fixed matrix, whereas C is random. So, the process to generate n documents, each containing m terms, is the following:

Definition 9.1 (Document Generation Process) *Let $\mathcal{D}$ be a distribution over a mixture of topics. Let B be the term-topic matrix. Create a $d \times n$ term-document matrix A as follows:*

- *Intialize $a_{ij} = 0$ for $i = 1, 2, \ldots, d; j = 1, 2, \ldots, n$.*[38]

[38]We will use i to index into the set of all terms, j to index documents and l to index topics.

- *For $j = 1, 2, \ldots, n$*
  - *Pick column j of C from distribution $\mathcal{D}$. This will be the topic mixture for document j, and induces $P(:, j) = BC(:, j)$.*
  - *For $t = 1, 2, \ldots, m$, do:*
    - *Generate the t^{th} term x_t of document j from the multinomial distribution over $\{1, 2, \ldots, d\}$ with probability vector $P(:, j)$ i.e., $Prob(x_t = i) = p_{ij}$.*
    - *Add $1/m$ to $a_{x_t,j}$.*

The topic modeling problem is to infer B and C from A. The probability distribution $\mathcal{D}$, of the columns of C is not yet specified. The most commonly used distribution is the Dirichlet distribution that we study in detail in Section 9.6.

Often we are given fewer terms of each document than the number of terms or the number of documents. Even though

$$E(a_{ij}|P) = p_{ij}, \tag{9.2}$$

and in expectation A equals P, the variance is high. For example, for the case when $p_{ij} = 1/d$ for all i with m much less than $\sqrt{d}$, $A(:, j)$ is likely to have $1/m$ in a random subset of m coordinates since no term is likely to be picked more than once. Thus

$$||A(:, j) - P(:, j)||_1 = m\left(\frac{1}{m} - \frac{1}{d}\right) + (d - m)\left(\frac{1}{d}\right) \approx 2,$$

the maximum possible. This says that in l_1 norm, which is the right norm when dealing with probability vectors, the "noise" $\mathbf{a}_{\cdot\mathbf{j}} - \mathbf{p}_{\cdot\mathbf{j}}$ is likely to be larger than $\mathbf{p}_{\cdot\mathbf{j}}$. This is one of the reasons why the model inference problem is hard. Write

$$A = BC + N, \tag{9.3}$$

where, A is the $d \times n$ term-document matrix, B is a $d \times r$ term-topic matrix and C is a $r \times n$ topic-document matrix. N stands for noise, which can have high norm. The l_1 norm of each column of N could be as high as that of BC.

There are two main ways of tackling the computational difficulty of finding B and C from A. One is to make assumptions on the matrices B and C that are both realistic and also admit efficient computation of B and C. The trade-off between these two desirable properties is not easy to strike and we will see several approaches beginning with the strongest assumptions on B and C in Section 9.2. The other way is to restrict N. Here again, an idealized way would be to assume $N = 0$ which leads to what is called the Nonnegative Matrix Factorization (NMF) (Section 9.3) problem of factoring the given matrix A into the product of two nonnegative matrices B and C. With a further restriction on B, called Anchor terms, (Section 9.4), there is a polynomial time algorithm to do NMF. The

strong restriction of $N = 0$ can be relaxed (Section **??**), but at the cost of computational efficiency.

The most common approach to topic modeling makes an assumption on the probability distribution of C, namely, that the columns of C are independent Dirichlet distributed random vectors. This is called the Latent Dirichlet Allocation model (Section 9.6), which does not admit an efficient computational procedure. We show that the Dirichlet distribution leads to many documents having a "primary topic," whose weight is much larger than average in the document. This motivates a model called the "Dominant Admixture model" (Section 9.7) which admits an efficient algorithm.

On top of whatever other assumptions are made, we assume that in each document, the m terms in it are drawn independently as in Definition 9.1. This is perhaps the biggest assumption of all.

9.2 An Idealized Model

The Topic Model inference problem is in general computationally hard. But under certain reasonable assumptions, it can be solved in polynomial time as we will see in this chapter. We start here with a highly idealized model that was historically the first for which a polynomial time algorithm was devised. In this model, we make two assumptions:

The Pure Topic Assumption: Each document is purely on a single topic. I.e., each column j of C has a single entry equal to 1, and the rest of the entries are 0.

Separability Assumption: The sets of terms occurring in different topics are disjoint. I.e., for each row i of B, there is a unique column l with $b_{il} \neq 0$.

Under these assumptions, the data matrix A has a block structure. Let T_l denote the set of documents on topic l and S_l the set of terms occurring in topic l. After rearranging columns and rows so that the rows in each S_l occur consecutively and the columns of each T_l occur consecutively, the matrix A looks like:

$$A = \begin{array}{c} \\ \\ S_1 \\ \\ \\ S_2 \\ \\ \\ S_3 \\ \\ \end{array} \begin{array}{c} \begin{array}{ccccccccc} & \multicolumn{3}{c}{\text{TOPIC}} & & & & & \\ & T_1 & & & & T_2 & & & T_3 & \\ \end{array} \\ \begin{pmatrix} * & * & * & 0. & 0 & 0 & 0 & 0 & 0 \\ * & * & * & 0 & 0 & 0 & 0 & 0 & 0 \\ * & * & * & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & * & * & * & 0 & 0 & 0 \\ 0 & 0 & 0 & * & * & * & 0 & 0 & 0 \\ 0 & 0 & 0 & * & * & * & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & * & * & * \\ 0 & 0 & 0 & 0 & 0 & 0 & * & * & * \\ 0. & 0 & 0 & 0 & 0 & 0 & * & * & * \end{pmatrix} \end{array} \begin{array}{c} \\ \\ \text{T} \\ \\ \text{E} \\ \\ \text{R} \\ \\ \text{M} \\ \\ \end{array}$$

If we can partition the documents into r clusters, $T_1, T_2, \ldots, T_r$, one for each topic, we can take the average of each cluster and that should be a good approximation to the corresponding column of B. It would also suffice to find the sets S_l of terms, since from them we could read off the sets T_l of topics. We now formally state the document generation process under the Pure Topic Assumption and the associated clustering problem. Note that under the Pure Topics Assumption, the distribution $\mathcal{D}$ over columns of C is specified by the probability that we pick each topic to be the only topic of a document. Let $\alpha_1, \alpha_2, \ldots, \alpha_r$ be these probabilities.

Document Generation Process under Pure Topics Assumption:

- Intialize all a_{ij} to zero.
- For each document do
  - Select a topic from the distribution given by $\{\alpha_1, \alpha_2, \ldots, \alpha_r\}$.
  - Select m words according to the distribution for the selected topic.
  - For each selected word add $1/m$ to the document-term entry of the matrix A.

Definition 9.2 (Clustering Problem) *Given A generated as above and the number of topics r, partition the documents $\{1, 2, \ldots, n\}$ into r clusters $T_1, T_2, \ldots, T_r$, each specified by a topic.*
Approximate Version: *Partition the documents into r clusters, where at most εn of the $j \in \{1, 2, \ldots, n\}$ are misclustered.*

The approximate version of Definition 9.2 suffices since we are taking the average of the document vectors in each cluster j and returning the result as our approximation to column j of B. Note that even if we clustered perfectly, the average will only approximate the column of B. We now show how we can find the term clusters S_l, which then can be used to solve the Clustering Problem.

Construct a graph G on d vertices, with one vertex per term, and put an edge between two vertices if they co-occur in any document. By the separability assumption, we know that there are no edges between vertices belonging to different S_l. This means that if each S_l is a connected component in this graph, then we will be done. Note that we need to assume $m \geq 2$ (each document has at least two words) since if all documents have just one word, there will be no edges in the graph at all and the task is hopeless.

Let us now focus on a specific topic l and ask how many documents n_l we need so that with high probability, S_l is a connected component. One annoyance here is that some words may have very low probability and not become connected to the rest of S_l. On the other hand, words of low probability can't cause much harm since they are unlikely to be the only words in a document, and so it doesn't matter that much if we fail to cluster them. We make this argument formal here.

Let $\gamma < 1/3$ and define $\varepsilon = \gamma^m$. Consider a partition of S_l into two subsets of terms W and $\overline{W}$ that each have probability mass at least γ in the distribution of terms in topic l. Suppose that for every such partition, there is at least one edge between W and $\overline{W}$. This would imply that the largest connected component $\hat{S}_l$ in S_l must have probability mass at least $1-\gamma$. If $\hat{S}_l$ had probability mass between γ and $1-\gamma$ then using $W = \hat{S}_l$ would violate the assumption about partitions with mass greater than γ having an edge between them. If the largest partition $\hat{S}_l$ had probability mass less than γ, then one could create a union of connected components W that violates the assumption. Since $\text{Prob}(\hat{S}_l) \geq 1-\gamma$, the probability that a new random document of topic l contains only words not in $\hat{S}_l$ is at most $\gamma^m = \varepsilon$. Thus, if we can prove the statement about partitions, we will be able to correctly cluster nearly all new random documents.

To prove the statement about partitions, fix some partition of S_l into W and $\overline{W}$ that each have probability mass at least γ. The probability that m words are all in W or $\overline{W}$ is at most $\text{Prob}(W)^m + \text{Prob}(\overline{W})^m$. Thus the probability that none of n_l documents creates an edge between W and $\overline{W}$ is

$$\begin{aligned} \left(\text{Prob}(W)^m + \text{Prob}(\overline{W})^m\right)^{n_l} &\leq (\gamma^m + (1-\gamma)^m)^{n_l} \\ &\leq ((1-\gamma/2)^m)^{n_l} \\ &\leq e^{-\gamma m n_l/2} \end{aligned}$$

where the first inequality is due to convexity and the second is a calculation. Since there are at most 2^d different possible partitions of S_l into W and $\overline{W}$, the union bound ensures at most a δ probability of failure by having

$$2^d e^{-\gamma m n_l/2} \leq \delta.$$

This in turn is satisfied for

$$m n_l \geq \frac{2}{\gamma}\left(d \ln 2 + \ln \frac{1}{\delta}\right).$$

This proves the following result.

Lemma 9.1 *If $n_l m \geq \frac{2}{\gamma}\left(d \ln 2 + \ln \frac{1}{\delta}\right)$, then with probability at least $1-\delta$, the largest connected component in S_l has probability mass at least $1-\gamma$. This in turn implies that the probability to fail to correctly cluster a new random document of topic l is at most $\varepsilon = \gamma^{1/m}$.*

9.3 Nonnegative Matrix Factorization - NMF

We saw in Section 9.1, while the expected value $E(A|B,C)$ equals BC, the variance can be high. Write

$$A = BC + N,$$

where, N stands for noise. In topic modeling, N can be high. But it will be useful to first look at the problem when there is no noise. This can be thought of as the limiting case

as the number of words per document goes to infinity.

Suppose we have the exact equations $A = BC$ where A is the given matrix with nonnegative entries and all column sums equal to 1. Given A and the number of topics r, can we find B and C such that $A = BC$ where B and C have nonnegative entries? This is called the Nonnegative Matrix Factorization (NMF) problem and has applications besides topic modeling. If B and C are allowed to have negative entries, we can use Singular Value Decomposition on A using the top r singular vectors of A.

Before discussing NMF, we will take care of one technical issue. In topic modeling, besides requiring B and C to be nonnegative, we have additional constraints stemming from the fact that frequencies of terms in one particular topic are nonnegative reals summing to one, and that the fractions of each topic that a particular document is on are also nonnegative reals summing to one. All together, the constraints are:

1. $A = BC$.
2. The entries of B and C are all nonnegative.
3. Columns of both B and C sums to one.

It will suffice to ensure the first two conditions.

Lemma 9.2 *Let A be a matrix with nonnegaitve elements and columns summing to one. The problem of finding a factorization BC of A satisfying the three conditions above is reducible to the NMF problem of finding a factorization BC satisfying conditions (1) and (2).*

Proof: Suppose we have a factorization BC that satisfies (1) and (2) of a matrix A whose columns each sum to one. We can multiply the l^{th} column of B by a positive real number and divide the l^{th} row of C by the same real number without violating $A = BC$. By doing this, we may assume that each column of B sums to one. Now we have $a_{ij} = \sum_l b_{il}c_{lj}$ which implies $\sum_i a_{ij} = \sum_{i,l} b_{il}c_{lj} = \sum_l c_{lj}$, the sum of the j^{th} column of C, $\sum_i a_{ij}$, is 1. Thus the columns of C sum to one giving (3). ∎

Given an $d \times n$ matrix A and an integer r, the exact NMF problem is to determine whether there exists a factorization of A into BC where B is an $d \times r$ matrix with nonnegative entries and C is $r \times n$ matrix with nonnegative entries and if so, find such a factorization.[39]

Nonnegative matrix factorization is a general problem and there are many heuristic algorithms to solve the problem. In general, they suffer from one of two problems. They could get stuck at local optima which are not solutions or take exponential time. In fact,

[39] B's columns form a "basis" in which A's columns can be expressed as nonnegative linear combinations, the "coefficients" being given by matrix C.

the NMF problem is NP-hard. In practice, often r is much smaller than n and d. We show first that while the NMF problem as formulated above is a nonlinear problem in $r(n+d)$ unknowns (the entries of B and C), it can be reformulated as a nonlinear problem with just $2r^2$ unknowns under the simple nondegeneracy assumption that A has rank r. This, in turn, allows for an algorithm that runs in polynomial time when r is a constant.

Lemma 9.3 *If A has rank r, then the NMF problem can be formulated as a problem with $2r^2$ unknowns. Using this, the exact NMF problem can be solved in polynomial time if r is constant.*

Proof: If $A = BC$, then each row of A is a linear combination of the rows of C. So the space spanned by the rows of A is contained in the space spanned by the rows of the $r \times n$ matrix C. The latter space has dimension at most r, while the former has dimension r by assumption. So they must be equal. Thus every row of C must be a linear combination of the rows of A. Choose any set of r independent rows of A to form a $r \times m$ matrix A_1. Then $C = SA_1$ for some $r \times r$ matrix S. By analogous reasoning, if A_2 is a $n \times r$ matrix of r independent columns of A, there is a $r \times r$ matrix T such that $B = A_2T$. Now we can easily cast NMF in terms of unknowns S and T:

$$A = A_2TSA_1 \quad ; \quad (SA_1)_{ij} \geq 0 \quad ; \quad (A_2T)_{kl} \geq 0 \quad \forall i, j, k, l.$$

It remains to solve the nonlinear problem in $2r^2$ variables. There is a classical algorithm which solves such problems in time exponential only in r^2 (polynomial in the other parameters). In fact, there is a logical theory, called the Theory of Reals, of which this is a special case and any problem in this theory can be solved in time exponential in the number of variables. We do not give details here. ∎

9.4 NMF with Anchor Terms

An important case of NMF, which can be solved efficiently, is the case where there are *anchor terms*. An anchor term for a topic is a term that occurs in the topic and does not occur in any other topic. For example, the term "batter" may be an anchor term for the topic baseball and "election" for the topic politics. Consider the case that each topic has an anchor term. This assumption is weaker than the separability assumption of Section 9.2, which says that all terms are anchor terms.

In matrix notation, the assumption that each topic has an anchor term implies that for each column of the term-topic matrix B, there is a row whose sole nonzero entry is in that column.

Definition 9.3 (Anchor Term) *For each $l = 1, 2, \ldots r$, there is an index i_l such that*

$$b_{i_l,l} \neq 0 \quad \textit{and} \quad \forall l' \neq l \;\; b_{i_l,l'} = 0 \, .$$

In this case, it is easy to see that each row of the topic-document matrix C has a scalar multiple of it occurring as a row of the given term-document matrix A.

$$\begin{pmatrix} \\ 0.3 \times \mathbf{c_4} \\ \\ A \\ \\ 0.2 \times \mathbf{c_2} \\ \\ \end{pmatrix} = \begin{matrix} \\ \text{election} \\ \\ \\ \\ \text{batter} \\ \\ \end{matrix} \begin{pmatrix} 0 & 0 & 0 & 0.3 \\ & & B & \\ 0 & 0.2 & 0 & 0 \end{pmatrix} \begin{pmatrix} \leftarrow \mathbf{c_1} \rightarrow \\ \leftarrow \mathbf{c_2} \rightarrow \\ \\ \leftarrow \mathbf{c_4} \rightarrow \\ \\ \end{pmatrix}.$$

If there is a NMF of A, there is one in which no row of C is a nonnegative linear combination of other rows of C. If some row of C is a nonnegative linear combination of the other rows of C, then eliminate that row of C as well as the corresponding column of B and suitably modify the other columns of B maintaining $A = BC$. For example, if

$$c_5 \;=\; 4 \times c_3 + 3 \times c_6,$$

delete row 5 of C, add 4 times column 5 of B to column 3 of B, add 3 times column 5 of B to column 6 of B, and delete column 5 of B. After repeating this, each row of C is positively independent of the other rows of C, i.e., it cannot be expressed as a nonnegative linear combination of the other rows.

If $A = BC$ is a NMF of A and there are rows in A that are positive linear combinations of other rows, the rows can be remove and the corresponding rows of B remove to give a NMF $\hat{A} = \hat{B}C$ where $\hat{A}$ and $\hat{B}$ are the matrices A and B with the removed rows. Since there are no rows in $\hat{A}$ that are linear combinations of other rows of $\hat{A}$, $\hat{B}$ is a diagonal matrix and the rows of $\hat{A}$ are scalar multiples of rows of C. Now set $C = \hat{A}$ and $\hat{B} = I$ and restore the rows to $\hat{B}$ to get B such that $A = BC$.

To remove rows of A that are scalar multiples of previous rows in polynomial time check if there are real numbers $x_1, x_2, \ldots x_{i-1}, x_{i+1}, \ldots x_n$ such that

$$\sum_{j \neq i} x_j \mathbf{a_j} = \mathbf{a_i} \quad x_j \geq 0.$$

This is a linear program and can be solved in polynomial time. While the algorithm runs in polynomial time, it requires solving one linear program per term. An improved method, not presented here, solves just one linear program.

9.5 Hard and Soft Clustering

In Section 9.2, we saw that under the assumptions that each document is purely on one topic and each term occurs in only one topic, approximately finding B was reducible

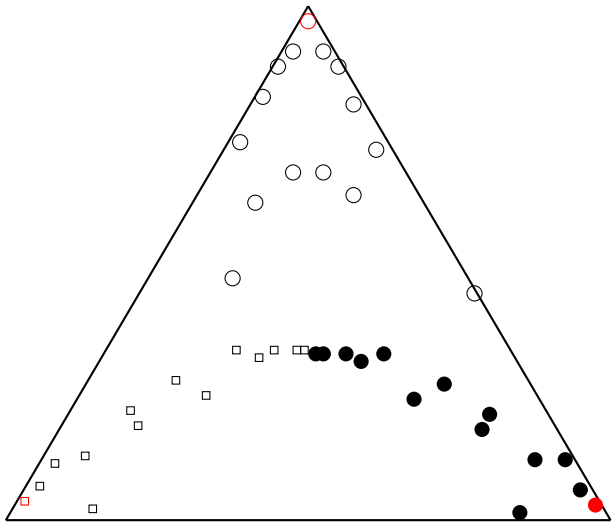


Figure 9.1: Geometry of Topic Modeling. The corners of the triangle are the columns of B. The columns of A for topic 1 are represented by circles, for topic 2 by squares, and for topic 3 by dark circles. Columns of BC (not shown) are always inside the big triangle, but not necessarily the columns of A.

to clustering documents according to their topic. Clustering here has the usual meaning of partitioning the set of documents into clusters. We call this *hard clustering*, meaning each data point is to be assigned to a single cluster.

The more general situation is that each document has a mixture of several topics. We may still view each topic as a cluster and each topic vector, i.e., each column of B, as a "cluster center" (Figure 9.1). But now, each document belongs fractionally to several clusters, the fractions being given by the column of C corresponding to the document. We may then view $P(:, j) = BC(:, j)$ as the "cluster center" for document j. The document vector $A(:, j)$ is its cluster center plus an offset or noise $N(:, j)$.

Barring ties, each column of C has a largest entry. This entry is the primary topic of document j in topic modeling. Identifying the primary topic of each document is a "hard clustering" problem, which intuitively is a useful step in solving the "soft clustering" problem of finding the fraction of each cluster each data point belongs to. "Soft Clustering" just refers to finding B and C so that $N = A - BC$ is small. In this sense, soft clustering is equivalent to NMF.

We will see in Sections 9.8 and 9.9 that doing hard clustering to identify the primary topic and using that to solve the soft clustering problem can be carried out under some assumptions. The primary topic of each document is used to find the "catchwords" of each topic, the important words in a weaker sense than anchor words, and then using the catchwords to find the term-topic matrix B and then C. But as stated earlier, the

general NMF problem is NP-hard. So, we make some assumptions before solving the problem. For this, we first look at Latent Dirichlet Allocation (LDA), which guides us towards reasonable assumptions.

9.6 The Latent Dirichlet Allocation Model for Topic Modeling

The most widely used model for topic modeling is the Latent Dirichlet Allocation (LDA) model. In this model, the topic weight vectors of the documents, the columns of C, are picked independently from what is known as a Dirichlet distribution. The term-topic matrix B is fixed. It is not random. The Dirichlet distribution has a parameter μ called the "concentration parameter", which is a real number in $(0, 1)$, typically set to $1/r$. For each vector $\mathbf{v}$ with r nonnegative components summing to one,

$$\text{Prob density } (\text{ column } j \text{ of } C = \mathbf{v}) = \frac{1}{g(\mu)} \prod_{l=1}^{r} v_l^{\mu-1},$$

where, $g(\mu)$ is the normalizing constant so that the total probability mass is one. Since $\mu < 1$, if any $v_l = 0$, then the probability density is infinite.

Once C is generated, the Latent Dirichlet Allocation model hypothesizes that the matrix

$$P = BC$$

acts as the probability matrix for the data matrix A, namely,

$$E(A|P) = P.$$

Assume the model picks m terms from each document. Each trial is according to the multinomial distribution with probability vector $P(:, j)$; so the probability that the first term we pick to include in the document j is the i^{th} term in the dictionary is p_{ij}. Then, a_{ij} is set equal to the fraction out of m of the number of times term i occurs in document j.

The Dirichlet density favors low v_l, but since the v_l have to sum to one, there is at least one component that is high. We show that if μ is small, then with high probability, the highest entry of the column is typically much larger than the average. So, in each document, one topic, which may be thought as the "primary topic" of the document, gets disproportionately high weight. To prove this, we have to work out some properties of the Dirichlet distribution. The first Lemma describes the marginal probability density of each coordinate of a Dirichlet distributed random variable:

Lemma 9.4 *Suppose the joint distribution of* $\mathbf{y} = (y_1, y_2, \ldots, y_r)$ *is the Dirichlet distribution with concentration parameter* μ*. Then, the marginal probability density* $q(y)$ *of* y_1 *is given by*

$$q(y) = \frac{\Gamma(r\mu + 1)}{\Gamma(\mu)\Gamma((r-1)\mu + 1)} y^{\mu-1}(1-y)^{(r-1)\mu}\ ,\ \mu \in (0, 1],$$

where, Γ *is the Gamma function (see Appendix for the definition).*

Proof: By definition of the marginal,

$$q(y) = \frac{1}{g(\mu)} y^{\mu-1} \int_{\substack{y_2,y_3,\ldots,y_r \\ y_2+y_3+\cdots+y_r=1-y}} (y_2\ y_3 \cdots y_r)^{\mu-1}\ dy_2\ dy_3\ \ldots dy_r.$$

Put $z_l = y_l/(1-y)$. With this change of variables,

$$q(y) = \frac{1}{g(\mu)} y^{\mu-1}(1-y)^{(r-1)\mu} \left(\int_{\substack{z_2,z_3,\ldots,z_r \\ z_2+z_3+\cdots+z_r=1-y}} (z_2 z_3 \cdots z_r)^{\mu-1}\ dz_2\ dz_3\ \ldots dz_r \right).$$

The quantity inside the parentheses is independent of y, so for some c we have

$$q(y) = c y^{\mu-1}(1-y)^{(r-1)\mu}.$$

Since $\int_0^1 q(y)\ dy = 1$, we must have

$$c = \frac{1}{\int_0^1 y^{\mu-1}(1-y)^{(r-1)\mu}} = \frac{\Gamma(r\mu+1)}{\Gamma(\mu)\Gamma((r-1)\mu+1)}.$$

■

Lemma 9.5 *Suppose the joint distribution of* $\mathbf{y} = (y_1, y_2, \ldots, y_r)$ *is the Dirichlet distribution with parameter* $\mu \in (0,1)$. *For* $\zeta \in (0,1)$,

$$Prob\,(y_1 \geq 1-\zeta) \geq \frac{0.85\mu\zeta^{(r-1)\mu+1}}{(r-1)\mu+1}.$$

Hence for $\mu = 1/r$, *we have* $Prob(y_1 \geq 1-\zeta) \geq 0.4\zeta^2/r$. *If also,* $\zeta < 0.5$, *then,*

$$Prob\,(Max_{l=1}^r y_l \geq 1-\zeta) \geq 0.4\zeta^2.$$

Proof: Since $\mu < 1$, we have $y^{\mu-1} > 1$ for $y < 1$ and so $q(y) \geq c(1-y)^{(r-1)\mu}$, so

$$\int_{1-\zeta}^1 q(y)\ dy \geq \frac{c}{(r-1)\mu+1}\zeta^{(r-1)\mu+1}.$$

To lower bound c, note that $\Gamma(\mu) \leq 1/\mu$ for $\mu \in (0,1)$. Also, $\Gamma(x)$ is an increasing function for $x \geq 1.5$, so if $(r-1)\mu+1 \geq 1.5$, then, $\Gamma(r\mu+1) \geq \Gamma((r-1)\mu+1)$ and in this case, the first assertion of the lemma follows. If $(r-1)\mu+1 \in [1, 1.5]$, then, $\Gamma((r-1)\mu+1) \leq 1$ and $\Gamma(r\mu+1) \geq \min_{z\in[1,2]} \Gamma(z) \geq 0.85$, so again, the first assertion follows.

If now, $\mu = 1/r$, then $(r-1)\mu+1 < 2$ and so $\zeta^{(r-1)\mu+1}/((r-1)\mu+1) \geq \zeta^2/2$. So the second assertion of the lemma follows easily. For the third assertion, note that $y_l > 1-\zeta$, $l = 1, 2, \ldots, r$ are mutually exclusive events for $\zeta < 0.5$ (since at most one y_l can be greater than $1/2$), so $\text{Prob}\left(\max_{l=1}^r y_l \geq 1-\zeta\right) = \sum_{l=1}^r \text{Prob}(y_l > 1-\zeta) = r\text{Prob}(y_1 \geq 1-\zeta) \geq 0.4\zeta^2$.

■

For example, from the last lemma, it follows that

1. With high probabilty, a constant fraction of the documents have a primary topic of weight at least 0.6. In expectation, the fraction of documents for which this holds is at least $0.4(0.6)^2$.

2. Also with high probability, a smaller constant fraction of the documents are nearly pure (weight at least 0.95 on a single topic). Take $\zeta = 0.05$.

If the total number of documents, n, is large, there will be many nearly pure documents. Since for nearly pure documents, $c_{l,j} \geq 0.95$, $BC_{:,j} = B(:,j) + \Delta$, where, $||\Delta||_1 \leq 0.05$. If we could find the nearly pure documents for a given topic l, then the average of the A columns corresponding to these documents will be close to the average of those columns in the matrix BC (though this is not true for individual columns) and it is intuitively clear that we would be done.

We pursue (1) and (2) in the next section, where we see that under these assumptions, plus one more assumption, we can indeed find B.
More generally, the concentration parameter may be different for different topics. We then have $\mu_1, \mu_2, \ldots, \mu_r$ so that

$$\text{Prob density } (\text{ column } j \text{ of } C = \mathbf{v}) \propto \prod_{l=1}^{r} v_l^{\mu_l - 1},$$

The model fitting problem for Latent Dirichlet Allocation given A, find the B, the term-topic matrix, is in general NP-hard. There are heuristics, however, which are widely used. Latent Dirichlet Allocation is known to work well in several application areas.

9.7 The Dominant Admixture Model

In this section, we formulate a model with three key assumptions. The first two are motivated by Latent Dirichlet Allocation, respectively by (1) and (2) of the last section. The third assumption is also natural; it is more realistic than the anchor words assumptions discussed earlier. This section is self-contained and no familiarity with Latent Dirichlet Allocation is needed.

We first recall the notation. A is a $d \times n$ data matrix with one document per column, which is the frequency vector of the d terms in that document. m is the number of words in each document. r is the "inner dimension", i.e., B is $d \times r$ and C is $r \times n$. We always index topics by l and l', terms by i, and documents by j.

We give an intuitive description of the model assumptions first and then make formal statements.

1. **Primary Topic** Each document has a primary topic. The weight of the primary topic in the document is high and the weight of each non-primary topic is low.

2. **Pure Document** Each topic has at least one *pure document* that is mostly on that topic.

3. **Catchword** Each topic has at least one *catchword*, which has high frequency in that topic and low frequency in other topics.

In the next section, we state quantitative versions of the assumptions and show that these assumptions suffice to yield a simple polynomial time algorithm to find the primary topic of each document. The primary topic classification can then be used to find B approximately, but this requires a further assumption (4) in Section (9.9) below, which is a robust version of the Pure Document assumption.

Let's provide some intuition for how we are able to do the primary topic classification. By using the primary topic and catchword assumptions, we can show (quantitative version in Claim 9.1 below) that if i is a catchword for topic l, then there is a threshold μ_i, which we can compute for each catchword, so that for each document j with primary topic l, p_{ij} is above μ_i and for each document j whose primary topic is not l, p_{ij} is substantially below μ_i. So, if

1. we were given P, and

2. knew a catchword for each topic and the threshold, we can find the primary topic of each document.

We illustrate the situation in Equation 9.4, where rows $1, 2, \ldots, r$ of P correspond to catchwords for topics $1, 2, \ldots, r$ and we have rearranged columns in order of primary topic. H stands for a high entry and L for a low entry.

$$P = \begin{pmatrix} H & H & H & L & L & L & L & L & L & L & L & L \\ L & L & L & H & H & H & L & L & L & L & L & L \\ L & L & L & L & L & L & H & H & H & L & L & L \\ L & L & L & L & L & L & L & L & L & H & H & H \\ . & . & . & . & . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . & . & . & . & . \end{pmatrix} \tag{9.4}$$

We are given A, but not P. While $E(A|P) = P$, A could be far off from P. In fact, if in column j of P, there are many entries smaller than c/m in A (since we are doing only m multinomial trials), they could all be zeros and so are not a good approximation to the entries of $P(:, j)$. However, if $p_{ij} > c/m$, for a large value c, then $a_{ij} \approx p_{ij}$. Think of tossing a coin m times whose probability of heads is p_{ij}. If $p_{ij} \geq c/m$, then the number of heads one gets is close to $p_{ij}m$. We will assume c larger than $\Omega(\log(nd))$ for catchwords so that for every such i and j we have $p_{ij} \approx a_{ij}$. See the formal catchwords assumption in the next section. This addresses (1), namely, $A \approx B$, at least in these rows.

One can ask if this is a reasonable assumption. If m is in the hundreds, the assumption is arguably reasonable. But a weaker and more reasonable assumption would be that there is a set of catchwords, not just one, with total frequency higher than c/m. However, here we use the stronger assumption of a single high frequency catchword.

(2) is more difficult to address. Let $l(i) = \arg\max_{l'=1}^{r} b_{il'}$. Let T_l be the set of j with primary topic l. Whether or not i is a catchword, the primary topic assumption will imply that p_{ij} does not drop by more than a certain factor α among $j \in T_{l(i)}$. We prove this formally in Claim 9.1 of Section 9.8. That claim also proves that if i is a catchword for topic l, that there is a sharp drop in p_{ij} between $j \in T_l$ and $j \notin T_l$.

But for noncatchwords, there is no guarantee of a sharp fall in p_{ij} between $j \in T_{l(i)}$ and $j \notin T_{l(i)}$. However, we can identify for each i, where the first fall of roughly α factor from the maximum occurs in row i of A. For catchwords, we show below (9.8) that this happens precisely between $T_{l(i)}$ and $[n] \setminus T_{l(i)}$. For noncatchwords, we show that the fall does not occur among $j \in T_{l(i)}$. So, the minimal sets where the fall occurs are the T_l and we use this to identify them. We call this process Pruning.

9.8 Formal Assumptions

Parameters α, β, ρ and δ are real numbers in $(0, 0.4]$ satisfying

$$\beta + \rho \leq (1 - 3\delta)\alpha. \tag{9.5}$$

(1) **Primary Topic** There is a partition of $[n]$ into $T_1, T_2, \ldots, T_k$ with:

$$c_{lj} \begin{cases} \geq \alpha & \text{for } j \in T_l \\ \leq \beta. & \text{for } j \notin T_l. \end{cases} \tag{9.6}$$

(2) **Pure Document** For each l, there is some j with

$$c_{lj} \geq 1 - \delta.$$

(3) **Catchwords** For each l, there is at least one catchword i satisfying:

$$b_{il'} \leq \rho b_{il} \qquad \text{for } l' \neq l \tag{9.7}$$

$$b_{il} \geq \mu\ , \quad \text{where, } \mu = \frac{c \log(10nd/\delta)}{m\alpha^2\delta^2}\ ,\ c \text{ constant.} \tag{9.8}$$

Let

$$l(i) = \arg\max_{l'=1}^{r} b_{il'}. \tag{9.9}$$

Another way of stating the assumption $b_{il} \geq \mu$ is that the expected number of times term i occurs in topic l among m independent trials is at least $c \log(10nd/\delta)/\alpha^2\delta^2$ which grows only logarithmically in n and d. As stated at the end of the last section, the point of requiring $b_{il} \geq \mu$ for catchwords is so that using the Hoeffding-Chernoff inequality, we can assert that $a_{ij} \approx p_{ij}$. We state the Hoeffding-Chernoff inequality in the form we use it:

Lemma 9.6

$$Prob(|a_{ij} - p_{ij}| \geq \delta\alpha Max(p_{ij}, \mu)/4) \leq \frac{\delta}{10nd}.$$

So, with probability at least $1 - (\delta/10)$*,*

$$|a_{ij} - p_{ij}| \leq \delta\alpha Max(\mu, p_{ij})/4 \quad \forall i, j$$

simultaneously. After paying the failure probability of $\delta/10$*, we henceforth assume that the above holds.*

Proof: Since a_{ij} is the average of m independent Bernoulli trials, each with expectation p_{ij}, the Hoeffding-Chernoff inequality asserts that

$$\text{Prob}\left(|a_{ij} - p_{ij}| \geq \Delta\right) \leq 2\exp\left(-cm\text{Min}\left(\frac{\Delta^2}{p_{ij}}, \Delta\right)\right).$$

Plugging in $\Delta = \alpha\delta\text{Max}(p_{ij}, \mu)/4$, the first statement of the lemma follows with some calculation. The second statement is proved by a union bound over the nd possible (i, j) values.

Algorithm

1. **Compute Thresholds:** $\mu_i = \alpha(1 - \delta) \max_j a_{ij}$.
2. **Do thresholding:** Define a matrix $\hat{A}$ by

$$\hat{a}_{ij} = \begin{cases} 1 & \text{if } a_{ij} \geq \mu_i \text{ and } \mu_i \geq \mu\alpha\left(1 - \frac{5\delta}{2}\right). \\ 0 & \text{otherwise .} \end{cases}$$

3. **Pruning:** Let $R_i = \{j|\hat{a}_{ij} = 1\}$. If any R_i strictly contains another, set all entries of row i of $\hat{A}$ to zero.

Theorem 9.7 *For* $i = 1, 2, \ldots, d$*, let* $R_i = \{j|\hat{a}_{ij} = 1\}$ *at the end of the algorithm. Then, each nonempty* $R_i = T_{l(i)}$*, with* $l(i)$ *as in (9.9).*

Proof: We start with a lemma which proves the theorem for catchwords. This is the bulk of the work in the proof of the theorem.

Lemma 9.8 *If* i *is a catchword for topic* l*, then* $R_i = T_l$*.*

Proof: Assume throughout this proof that i is a catchword for topic l. The proof consists of three claims. The first argues that for $j \in T_l$, p_{ij} is high and for $j \notin T_l$, p_{ij} is low. The second claim argues the same for a_{ij} instead of p_{ij}. It follows from the Hoeffding-Chernoff inequality since a_{ij} is just the average of m Bernoulli trials, each with probability p_{ij}. The third claim shows that the threshold computed in the first step of the algorithm falls between the high and the low.

Claim 9.1 *For i, a catchword for topic l,*

$$\begin{aligned} b_{il} \geq p_{ij} \geq b_{il}\alpha \qquad & \text{for } j \in T_l \\ p_{ij} \leq b_{il}\alpha(1-3\delta) \qquad & \text{for } j \notin T_l. \end{aligned}$$

Proof: For $j \in T_l$, using (9.6)

$$p_{ij} = \sum_{l'=1}^{r} b_{il'} c_{l',j} \in [b_{il}\alpha, b_{il}]$$

since $b_{il} = \max_{l'} b_{il'}$. For $j \notin T_l$,

$$p_{ij} = b_{il}c_{lj} + \sum_{l' \neq l} b_{il'}c_{l'j} \leq b_{il}c_{lj} + \rho b_{il}(1-c_{lj}) \leq b_{il}(\beta+\rho) \leq b_{il}\alpha(1-3\delta), \tag{9.10}$$

where, the first inequality is from (9.7) and the second inequality is because subject to the constraint $c_{lj} \leq \beta$ imposed by the Primary Topic Assumption (9.6), $b_{il}c_{lj} + \rho b_{il}(1-c_{lj})$ is maximized when $c_{lj} = \beta$. We have also used (9.5).

Claim 9.2 *With probability at least $1-\delta/10$, for every l and every catchword i of l:*

$$a_{ij} \begin{cases} \geq b_{il}\alpha(1-\delta/4) & \text{for } j \in T_l \\ \leq b_{il}\alpha(1-(11/4)\delta), & \text{for } j \notin T_l \end{cases}$$

Proof: Suppose for some $j \in T_l$, $a_{ij} < b_{il}\alpha(1-\delta/4)$. Then, since $p_{ij} \geq b_{il}\alpha$ by Claim (9.1), $|a_{ij} - p_{ij}| \geq \delta\alpha b_{il}/4$ by Claim (9.1). Since i is a catchword, $b_{il} \geq \mu$ and so $|a_{ij} - p_{ij}| \geq \delta\alpha b_{il}/4 \geq (\delta\alpha \text{Max}(p_{ij}, \mu)/4)$ and we get the first inequality of the current claim using Lemma 9.6.

For the second inequality: for $j \notin T_l$, $p_{ij} \leq b_{il}\alpha(1-3\delta)$ by Claim 9.1 and so if this inequality is violated, $|a_{ij} - p_{ij}| \geq b_{il}\alpha\delta/4$ and we get a contradiction to Lemma (9.6). ∎

Claim 9.3 *With probability at least $1-\delta$, for every topic l and every catchword i of topic l, the μ_i computed in step 1 of the algorithm satisfies: $\mu_i \in \big((1-(5/2)\delta)b_{il}\alpha \ , \ b_{il}\alpha(1-\delta/2)\big)$.*

Proof: If i is a catchword for topic l and j_0 a pure document for l, then

$$p_{ij_0} = \sum_{l'=1}^{k} b_{il'}c_{l'j_0} \geq b_{il}c_{lj_0} \geq (1-\delta)b_{il}.$$

Applying Lemma 9.6, $a_{ij_0} > (1-(3/2)\delta)b_{il}$. Thus, μ_i computed in step 1 of the algorithm satisfies $\mu_i > (1-(3\delta/2))(1-\delta)b_{il}\alpha \geq (1-(5/2)\delta)\alpha b_{il}$. Hence, $\hat{a}_{ij}$ is not set to zero for all j. Now, since $p_{ij} \leq b_{il}$ for all j, $a_{ij} \leq (1+\delta/4)b_{il}$ by Lemma 9.6 implying

$$\mu_i = \text{Max}_j a_{ij}(1-\delta)\alpha \leq b_{il}(1+(\delta/4))(1-\delta)\alpha \leq b_{il}\alpha(1-\delta/2).$$

∎

Claims 9.2 and 9.3, complete the proof of Lemma 9.8. ∎

The lemma proves Theorem 9.7 for catchwords. Note that since each topic has at least one catchword, for each l, there is some i with $R_i = T_l$.

Suppose i is a noncatchword. Let $a = \max_j a_{ij}$. If $a < \mu(1 - (5\delta/2))$, then $\mu_i < \mu\alpha(1-(5\delta/2))$ and the entire row of $\hat{A}$ will be set to all zeros by the algorithm, so $R_i = \emptyset$ and there is nothing to prove. Assume that $a \geq \mu(1 - (5\delta/2))$. Let $j_0 = \arg\max_j a_{ij}$. Then $a = a_{ij_0} \geq \mu(1 - (5\delta/2))$. We claim $p_{i,j_0} \geq a(1 - \delta/2)$. If not, $p_{ij_0} < a(1 - \delta/2)$ and

$$|a_{ij_0} - p_{ij_0}| > \max\left(\frac{p_{ij_0}\delta}{4}, \frac{\mu\alpha\delta}{4}\right),$$

which contradicts Lemma 9.6. So,

$$b_{il} \geq p_{ij_0} \geq \mu(1 - 3\delta). \tag{9.11}$$

Let $l = l(i)$. Then

$$a(1 - \delta/2) \leq p_{ij_0} = \sum_{l'=1}^{r} b_{il'} c_{l'j_0} \leq b_{il}.$$

Also, if j_1 is a pure document for topic l, $c_{l,j_1} \geq (1 - \delta)$ so, $p_{i,j_1} \geq b_{il}c_{l,j_1} \geq b_{il}(1 - \delta)$. Now, we claim that

$$a_{i,j_1} \geq b_{il}(1 - (3\delta/2)). \tag{9.12}$$

If not,

$$p_{ij_1} - a_{ij_1} > b_{il}(1 - \delta) - b_{il}(1 - (3\delta/2)) = b_{il}(\delta/2) \geq \max\left(\frac{\mu\delta}{4}, \frac{p_{ij_1}\delta}{4}\right),$$

contradicting Lemma 9.6. So (9.12) holds and thus,

$$a \geq b_{il}(1 - (3\delta/2)) \tag{9.13}$$

Now, for all $j \in T_l$, $p_{ij} \geq b_{il}c_{lj} \geq a(1 - \delta/2)\alpha$. So, by applying Lemma 9.6 again, for all $j \in T_l$,

$$a_{ij} \geq a(1 - \delta)\alpha.$$

By step 1 of the algorithm, $\mu_i = a(1 - \delta)\alpha$, so $a_{ij} \geq \mu_i$ for all $j \in T_l$. So, either $R_i = T_l$ or $T_l \subsetneq R_i$. In the latter case, the pruning step will set $\hat{a}_{ij} = 0$ for all j, since topic l has some catchword i_0 for which $R_{i_0} = T_l$ by Lemma 9.8.

9.9 Finding the Term-Topic Matrix

For this, we need an extra assumption, which we first motivate. Suppose as in Section 9.8, we assume that there is a single pure document for each topic. In terms of the Figure

9.1 of three topics, this says that there is a column of P close to each vertex of the triangle. But the corresponding column of A can be very far from this. So, even if we were told which document is pure for each topic, we cannot find the column of B. However, if we had a large number of nearly pure documents for each topic, since the corresponding columns of A are independent even conditioned on P, the average of these columns gives us a good estimate of the column of B. We also note that there is a justification for assuming the existence of a number of documents which are nearly pure for each topic based on the Latent Dirichlet Allocation model, (See (2) of Section 9.6). The assumption is:

Assumption (4): Set of Pure Documents For each l, there is a set W_l of at least δn documents with

$$c_{lj} \geq 1 - \frac{\delta}{4} \; \forall j \in W_l.$$

If we could find the set of pure documents for each topic with possibly a small fraction of errors, we could average them. The major task of this section is to state and prove an algorithm that does this. For this, we use the primary topic classification, $T_1, T_2, \ldots, T_r$ from the last section. We know that a for catchword i of topic l, the maximum value of $p_{ij}, j = 1, 2, \ldots, n$ occurs for a pure document and indeed if the assumption above holds, the set of $\delta n/4$ documents with the top $\delta n/4$ values of p_{ij} should be all pure documents. But to make use of this, we need to know the catchword, which we are not given. To discover them, we use another property of catchwords. If i is a catchword for topic l, then on $T_{l'}, l' \neq l$, the values of p_{ij} are (substantially) lower. So we know that if i is a catchword of topic l, then it has the property:

Property: $\delta n/4$ th maximum value among $p_{ij}, j \in T_l$ is substantially higher than than the $\delta n/4$ th maximum value among $p_{ij}, j \in T_{l'}$ for any $l' \neq l$.

We can computationally recognize the property for A (not P) and on the lines of Lemma 9.6, we can show that it holds essentially for A if and only if it holds for P.

But then, we need to prove a converse of the statement above, namely we need to show that if the property holds for i and l, then i is a catchword for topic l. Since catchwords are not necessarily unique, this is not quite true. But we will prove that any i satisfying the property for topic l does have $b_{il'} < \alpha b_{il} \; \forall l' \neq l$ (Lemma 9.11) and so acts essentially like a catchword. Using this, we will show that the $\delta n/4$ documents among all documents with the highest values of a_{ij} for an i satisfying the property, will be nearly pure documents on topic l in Lemma 9.12 and use this to argue that their average gives a good approximation to column l of B (Theorem 9.13).

The extra steps in the Algorithm: (By the theorem, the $T_l, l = 1, 2, \ldots, r$ are now known.)

1. For $l = 1, 2, \ldots, r$, and for $i = 1, 2, \ldots, d$, let $g(i, l)$ be the $(1 - (\delta/4))^{/;/;th}$ fractile of

$\{A_{ij} : j \in T_l\}$. [40]

2. For each l, choose an $i(l)$, (we will prove there is at least 1) such that

$$g(i(l), l) \geq (1 - (\delta/2))\mu \ ; \ g(i(l), l') \leq (1 - 2\delta) \ \alpha \ g(i(l), l) \ \forall l' \neq l. \tag{9.14}$$

3. Let R_l be the set of $\delta n/4$ j's among $j = 1, 2, \ldots, n$ with the highest $A_{i(l),j}$.

4. Return $\tilde{B}_{\cdot,l} = \frac{1}{|R_l|} \sum_{j \in R_l} A_{\cdot,j}$ as our approximation to $B_{\cdot,l}$.

Lemma 9.9 *$i(l)$ satisfying (9.14) exists for each l.*

Proof: Let i be a catchword for l. Then, since, $\forall j \in W_l$, $p_{ij} \geq b_{il}c_{lj} \geq b_{il}(1 - (\delta/4))$ and $b_{il} \geq \mu$, we have $a_{ij} \geq b_{il}(1 - (\delta/2))$ and so $g(i, l) \geq (1 - (\delta/2))b_{il} \geq (1 - (\delta/2))\mu$, by Lemma 9.6. For $j \notin T_l$,

$$a_{ij} \leq b_{il}\alpha(1 - (5\delta/2))$$

by Claim 1.2 and so $g(i, l') \leq b_{il}\alpha(1 - (5\delta/2))$. So $g(i, l)$ satisfies both the requirements of step 2 of the algorithm. ∎

Fix attention on one l. Let $i = i(l)$. Let

$$\rho_i = \max_{k=1}^{r} b_{ik}.$$

Lemma 9.10 $\rho_i \geq \left(1 - \frac{3}{4}\delta\right)\mu$.

Proof: We have $p_{ij} = \sum_{k=1}^{r} b_{ik}c_{kj} \leq \rho_i$ for all i. So, $a_{ij} \leq \rho_i + \frac{\alpha\delta}{4} \max(\mu, \rho_i)$, from Lemma 9.6. So, either, $\rho_i \geq \mu$ whence the lemma clearly holds or $\rho_i < \mu$ and

$$\forall j, \ a_{ij} \leq \rho_i + \frac{\alpha\delta}{4}\mu \ \forall j \implies g(i, l) \leq \rho_i + (\alpha\delta/4)\mu.$$

By definition of $i(l)$, $g(i, l) \geq (1 - (\delta/2))\mu$, so

$$\rho_i + \frac{\alpha\delta}{4}\mu \geq (1 - (\delta/2))\mu,$$

from which the lemma follows.

Lemma 9.11 *$b_{ik} \leq \alpha b_{il}$ for all $k \neq l$.*

[40] The γ^{th} fractile of a set S of real numbers is the largest real number a so that at least $\gamma|S|$ elements of S are each greter than or equal to a.

Proof: Suppose not. Let

$$l' = \arg\max_{k:k\neq l} b_{ik}.$$

We have $b_{il'} > \alpha b_{il}$ and

$$\rho_i = \text{Max}(b_{il}, b_{il'}) < \frac{b_{il'}}{\alpha}. \tag{9.15}$$

Since for $j \in W_{l'}$, at most $\delta/4$ weight is put on topics other than l',

$$\forall j \in W_{l'}\ ,\ p_{ij} \leq b_{il'}(1-\frac{\delta}{4})) + \frac{\delta}{4}\rho_i < b_{il'}\left(1 - \frac{\delta}{4} + \frac{\delta}{4\alpha}\right). \tag{9.16}$$

Also, for $j \in W_{l'}$,

$$p_{ij} \geq b_{il'}c_{l'j} \geq b_{il'}(1-\frac{\delta}{4}). \tag{9.17}$$

By Lemma 9.6,

$$\begin{aligned}\forall j \in W_{l'}, a_{ij} &\geq p_{ij} - \frac{\alpha\delta}{4}\max(\mu, p_{ij}) \geq b_{il'}(1-\frac{\delta}{4}) - \frac{\alpha\delta}{4}\frac{\rho_i}{1-(3\delta/4)} \text{ by Lemma 9.10} \\ &\geq b_{il'}\left(1 - \frac{3}{4}\delta\right), \end{aligned} \tag{9.18}$$

using (9.15) and $\delta \leq 0.4$. From (9.18), it follows that

$$g(i,l') \geq b_{il'}\left(1 - \frac{3}{4}\delta\right). \tag{9.19}$$

Since $p_{ij} \leq \rho_i$ for all j, using Lemma 9.10,

$$\forall j, a_{ij} \leq \rho_i + (\alpha\delta/4)\text{Max}(\mu, p_{ij}) \leq \rho_i\left(1 + \frac{\alpha\delta}{4}\frac{1}{1-(3\delta/4)}\right) < b_{il'}\frac{1+(5\delta/6)}{\alpha},$$

by (9.15); this implies

$$g(i,l) \leq \frac{b_{il'}(1+(5\delta/6))}{\alpha}. \tag{9.20}$$

Now, the defintion of $i(l)$ implies

$$g(i,l') \leq g(i,l)\alpha(1-2\delta) \leq b_{il'}(1+(5\delta/6))\alpha(1-2\delta)/\alpha \leq b_{il'}(1-\delta)$$

contradicting (9.19) and proving Lemma 9.11.

Lemma 9.12 *For each $j \in R_l$ of step 3 of the algorithm, we have*

$$c_{lj} \geq 1 - 2\delta.$$

Proof: Let $J = \{j : c_{lj} < 1 - 2\delta\}$. Take a $j \in J$. We argue that $j \notin R_l$.

$$p_{ij} \leq b_{il}(1 - 2\delta) + 2\delta\alpha b_{il} \leq b_{il}(1 - 1.2\delta),$$

by Lemma 9.11 using $\alpha \leq 0.4$. So for $j \in J$, we have

$$a_{ij} \leq b_{il}(1 - 1.2\delta) + \frac{\alpha\delta}{4}\max(\mu, b_{il}) < b_{il}(1 - \delta).$$

But

$$\forall j \in W_l, p_{ij} \geq b_{il}(1 - \frac{\delta}{4}) \implies a_{ij} \geq b_{il}(1 - \delta) \implies g(i, l) \geq b_{il}(1 - \delta).$$

So for no $j \in J$ is $a_{ij} \geq g(i, l)$ and hence no $j \in J$ belong sto R_l.

Theorem 9.13 *Assume*

$$n \geq \frac{cd}{m\delta^3} \; ; \; m \geq \frac{c}{\delta^2}.$$

For all $l, 1 \leq l \leq r$, the $\hat{\mathbf{b}}_{\cdot,\mathbf{l}}$ returned by step 4 of the Algorithm satisfies

$$||\mathbf{b}_{\cdot,\mathbf{l}} - \hat{\mathbf{b}}_{\cdot,\mathbf{l}}||_1 \leq 6\delta.$$

Proof: Recall that $BC = P$. Let $V = A - P$. From Lemma 9.12, we know that for each $j \in R_l$, $c_{lj} \geq 1 - 2\delta$. So

$$P_{\cdot,j} = (1 - \gamma)B_{\cdot,l} + \mathbf{v},$$

where, $\gamma \leq 2\delta$ and $\mathbf{v}$ is a combination of other columns of B with $||\mathbf{v}||_1 \leq \gamma \leq 2\delta$. Thus, we have that

$$\left|\left|\frac{1}{|R_l|}\sum_{j \in R_l} \mathbf{p}_{\cdot,\mathbf{j}} \; - \; \mathbf{b}_{\cdot,\mathbf{l}}\right|\right|_1 \leq 2\delta. \tag{9.21}$$

So it suffices now to show that

$$\left|\left|\frac{1}{|R_l|}\sum_{j \in R_l} \mathbf{p}_{\cdot,\mathbf{j}} \; - \; \mathbf{a}_{\cdot,\mathbf{l}}\right|\right|_1 \leq 2\delta. \tag{9.22}$$

Note that for an individual $j \in R_l$, $||\mathbf{a}_{\cdot,\mathbf{j}} - \mathbf{p}_{\cdot,\mathbf{j}}||_1$ can be almost two. For example, if each $\mathbf{p_{i,j}} = 1/d$, then, A_{ij} would be $1/m$ for a random subset of m j 's and zero for the rest. What we exploit is that when we average over $\Omega(n)$ j 's in R_l, the error is small. For this, the independence of $\mathbf{a}_{\cdot,\mathbf{j}}, j \in R_l$ would be useful. But they are not necessarily independent, there being conditioning on the fact that they all belong to R_l. But there is a simple way around this conditioning. Namely, we prove (9.22) with very high probability for each $R \subseteq [n], |R| = \delta n/4$ and then just take the union bound over all $\binom{n}{(\delta n/4)}$ such subsets.

We know that $E(\mathbf{a}_{\cdot,\mathbf{j}}) = \mathbf{p}_{\cdot,\mathbf{j}}$. Now consider the random variable x defined by

$$x = \frac{1}{|R|}\left|\left|\sum_{j \in R} \mathbf{v}_{\cdot,\mathbf{j}}\right|\right|_1.$$

x is a function of $m|R|$ independent random variables, namely, the choice of $m|R|$ terms in the $|R|$ documents. Changing any one, changes x by at most $1/m|R|$. So the Bounded Difference Inequality from probability (??? REF ???) implies that

$$\text{Prob}\left(|x - Ex| > \delta\right) \leq 2\exp\left(-\delta^2\delta mn/8\right). \tag{9.23}$$

We also have to bound $E(x)$.

$$\begin{aligned}
E(x) &= \frac{1}{|R|}E\left(||\sum_{j\in R} V_{\cdot,j}||_1\right) = \frac{1}{|R|}\sum_{i=1}^{d} E\left|\sum_{j\in R} v_{ij}\right| \\
&\leq \frac{1}{|R|}\sum_{i=1}^{d}\sqrt{E\left[\left(\sum_{j\in R} v_{ij}\right)^2\right]} \text{ Jensen's inequality:} E(y) \leq \sqrt{E(y^2)} \\
&= \frac{1}{|R|}\sum_{i=1}^{d}\sqrt{\sum_{j\in R} E(v_{ij}^2)} \text{ since } \{v_{ij}, j \in R\} \text{are indep. and var adds up} \\
&\leq \frac{\sqrt{d}}{|R|}\left(\sum_{i=1}^{d}\sum_{j\in R} E(v_{ij}^2)\right)^{1/2} \text{ Chauchy-Schwartz} \\
&= \frac{\sqrt{d}}{|R|}\sqrt{\sum_{j}\sum_{i=1}^{d} E(v_{ij}^2)} \leq \frac{\sqrt{d}}{\sqrt{m\delta n}} \leq \delta,
\end{aligned}$$

since $E(v_{ij}^2) = p_{ij}/m$ and $\sum_i p_{ij} = 1$ and by hypothesis, $n \geq cd/m\delta^3$. Using this along with (9.23), we see that for a single $R \subseteq \{1, 2, \ldots, n\}$ with $|R| = \delta n/4$,

$$\text{Prob}\left(\left|\left|\frac{1}{|R|}\sum_{j\in R}\mathbf{v}_{\cdot,\mathbf{j}}\right|\right|_1 \geq 2\delta\right) \leq 2\exp\left(-c\delta^3 mn\right),$$

which implies using the union bound that

$$\text{Prob}\left(\exists R, |R| = \frac{\delta n}{4} \ : \ \left|\left|\frac{1}{|R|}\sum_{j\in R}\mathbf{v}_{\cdot,\mathbf{j}}\right|\right|_1 \geq 2\delta\right) \leq 2\exp\left(-c\delta^3 mn + c\delta n\right) \leq \delta,$$

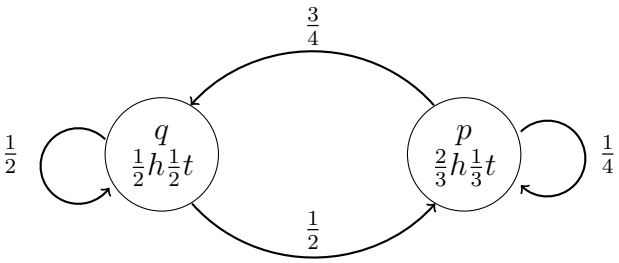
because the number of R is $\binom{n}{(\delta n/4)} \leq (cn/\delta n)^{\delta n/4} \leq \exp(c\delta n)$ and $m \geq c/\delta^2$ by hypothesis. This completes the proof of the theorem. ∎

9.10 Hidden Markov Models

A *hidden Markov model* (HMM) consists of a finite set of states with a transition between each pair of states. There is an initial probability distribution α on the states

and a transition probability a_{ij} associated with the transition from state i to state j. Each state also has a probability distribution $p(O, i)$ giving the probability of outputting the symbol O in state i. A transition consists of two components. A state transition to a new state followed by the output of a symbol. The HMM starts by selecting a start state according to the distribution α and outputting a symbol.

Example: An example of an HMM with two states q and p and two output symbols h and t is illustrated below.



The initial distribution is $\alpha(q) = 1$ and $\alpha(p) = 0$. At each step a change of state occurs followed by the output of heads or tails with probability determined by the new state. ∎

We consider three problems in increasing order of difficulty. First, given an HMM what is the probability of a given output sequence? Second, given an HMM and an output sequence, what is the most likely sequence of states? And third, knowing that the HMM has at most n states and given an output sequence, what is the most likely HMM? Only the third problem concerns a "hidden" Markov model. In the other two problems, the model is known and the questions can be answered in polynomial time using dynamic programming. There is no known polynomial time algorithm for the third question.

How probable is an output sequence

Given an HMM, how probable is the output sequence $O = O_0O_1O_2 \cdots O_T$ of length $T+1$? To determine this, calculate for each state i and each initial segment of the sequence of observations, $O_0O_1O_2 \cdots O_t$ of length $t+1$, the probability of observing $O_0O_1O_2 \cdots O_t$ ending in state i. This is done by a dynamic programming algorithm starting with $t = 0$ and increasing t. For $t = 0$ there have been no transitions. Thus, the probability of observing O_0 ending in state i is the initial probability of starting in state i times the probability of observing O_0 in state i. The probability of observing $O_0O_1O_2 \cdots O_t$ ending in state i is the sum of the probabilities over all states j of observing $O_0O_1O_2 \cdots O_{t-1}$ ending in state j times the probability of going from state j to state i and observing O_t. The time to compute the probability of a sequence of length T when there are n states is $O(n^2T)$. The factor n^2 comes from the calculation for each time unit of the contribution from each possible previous state to the probability of each possible current state. The space complexity is $O(n)$ since one only needs to remember the probability of reaching

each state for the most recent value of t.

Algorithm to calculate the probability of the output sequence

The probability, $\text{Prob}(O_0O_1\cdots O_T, i)$ of the output sequence $O_0O_1\cdots O_T$ ending in state i is given by

$$\text{Prob}(O_0, i) = \alpha(i)p(O_0, i)$$

and for $t = 1$ to T

$$\text{Prob}(O_0O_1\cdots O_t, i) = \sum_j \text{Prob}(O_0O_1\cdots O_{t-1}, j)a_{ij}p(O_{t+1}, i).$$

Example: What is the probability of the sequence hhht by the HMM in the above two state example?

$t = 3$	$\frac{3}{32}\frac{1}{2}\frac{1}{2} + \frac{5}{72}\frac{3}{4}\frac{1}{2} = \frac{19}{384}$	$\frac{3}{32}\frac{1}{2}\frac{1}{3} + \frac{5}{72}\frac{1}{4}\frac{1}{3} = \frac{37}{64\times27}$
$t = 2$	$\frac{1}{8}\frac{1}{2}\frac{1}{2} + \frac{1}{6}\frac{3}{4}\frac{1}{2} = \frac{3}{32}$	$\frac{1}{8}\frac{1}{2}\frac{2}{3} + \frac{1}{6}\frac{1}{4}\frac{2}{3} = \frac{5}{72}$
$t = 1$	$\frac{1}{2}\frac{1}{2}\frac{1}{2} = \frac{1}{8}$	$\frac{1}{2}\frac{1}{2}\frac{2}{3} = \frac{1}{6}$
$t = 0$	$\frac{1}{2}$	0
	q	p

For $t = 0$, the q entry is 1/2 since the probability of being in state q is one and the probability of outputting heads is $\frac{1}{2}$. The entry for p is zero since the probability of starting in state p is zero. For $t = 1$, the q entry is $\frac{1}{8}$ since for $t = 0$ the q entry is $\frac{1}{2}$ and in state q the HMM goes to state q with probability $\frac{1}{2}$ and outputs heads with probability $\frac{1}{2}$. The p entry is $\frac{1}{6}$ since for $t = 0$ the q entry is $\frac{1}{2}$ and in state q the HMM goes to state p with probability $\frac{1}{2}$ and outputs heads with probability $\frac{2}{3}$. For $t = 2$, the q entry is $\frac{3}{32}$ which consists of two terms. The first term is the probability of ending in state q at $t = 1$ times the probability of staying in q and outputting h. The second is the probability of ending in state p at $t = 1$ times the probability of going from state p to state q and outputting h.

From the table, the probability of producing the sequence hhht is $\frac{19}{384} + \frac{37}{1728} = 0.0709$.

∎

The most likely sequence of states - the Viterbi algorithm

Given an HMM and an observation $O = O_0O_1\cdots O_T$, what is the most likely sequence of states? The solution is given by the Viterbi algorithm, which is a slight modification to the dynamic programming algorithm just given for determining the probability of an output sequence. For $t = 0, 1, 2, \ldots, T$ and for each state i, we calculate the probability

of the most likely sequence of states to produce the output $O_0O_1O_2\cdots O_t$ ending in state i as follows. For each state j, we have already computed the probability of the most likely sequence producing $O_0O_1O_2\cdots O_{t-1}$ ending in state j, and we multiply this by the probability of the transition from j to i producing O_t. We then select the j for which this product is largest. Note that in the previous example, we added the probabilities of each possibility together. Now we take the maximum and also record where the maximum came from. The time complexity is $O(n^2T)$ and the space complexity is $O(nT)$. The space complexity bound is argued as follows. In calculating the probability of the most likely sequence of states that produces $O_0O_1\ldots O_t$ ending in state i, we remember the previous state j by putting an arrow with edge label t from i to j. At the end, can find the most likely sequence by tracing backwards as is standard for dynamic programming algorithms.

Example: For the earlier example what is the most likely sequence of states to produce the output hhht?

$t=3$	$\max\{\frac{1}{48}\frac{1}{2}\frac{1}{2}, \frac{1}{24}\frac{3}{4}\frac{1}{2}\} = \frac{1}{64}$ q or p	$\max\{\frac{3}{48}\frac{1}{2}\frac{1}{3}, \frac{1}{24}\frac{1}{4}\frac{1}{3}\} = \frac{1}{96}$ q
$t=2$	$\max\{\frac{1}{8}\frac{1}{2}\frac{1}{2}, \frac{1}{6}\frac{3}{4}\frac{1}{2}\} = \frac{3}{48}$ p	$\max\{\frac{1}{8}\frac{1}{2}\frac{2}{3}, \frac{1}{6}\frac{1}{4}\frac{2}{3}\} = \frac{1}{24}$ q
$t=1$	$\frac{1}{2}\frac{1}{2}\frac{1}{2} = \frac{1}{8}$ q	$\frac{1}{2}\frac{1}{2}\frac{2}{3} = \frac{1}{6}$ q
$t=0$	$\frac{1}{2}$ q	0 p
	q	p

Note that the two sequences of states, $qqpq$ and $qpqq$, are tied for the most likely sequences of states. ∎

Determining the underlying hidden Markov model

Given an n-state HMM, how do we adjust the transition probabilities and output probabilities to maximize the probability of an output sequence $O_1O_2\cdots O_T$? The assumption is that T is much larger than n.[41] There is no known computationally efficient method for solving this problem. However, there are iterative techniques that converge to a local optimum.

Let a_{ij} be the transition probability from state i to state j and let $b_j(O_k)$ be the probability of output O_k given that the HMM is in state j. Given estimates for the HMM parameters, a_{ij} and b_j, and the output sequence O, we can improve the estimates by calculating for each time step the probability that the HMM goes from state i to state j and outputs the symbol O_k, conditioned on O being the output sequence.

[41] If $T \leq n$ then one can just have the HMM be a linear sequence that outputs $O_1O_2\ldots O_T$ with probability 1.

a_{ij}	transition probability from state i to state j
$b_j(O_{t+1})$	probability of O_{t+1} given that the HMM is in state j at time $t+1$
$\alpha_t(i)$	probability of seeing $O_0O_1\cdots O_t$ and ending in state i at time t
$\beta_{t+1}(j)$	probability of seeing the tail of the sequence $O_{t+2}O_{t+3}\cdots O_T$ given state j at time $t+1$
$\delta(i,j)$	probability of going from state i to state j at time t given the sequence of outputs O
$s_t(i)$	probability of being in state i at time t given the sequence of outputs O
$p(O)$	probability of output sequence O

Given estimates for the HMM parameters, a_{ij} and b_j, and the output sequence O, the probability $\delta_t(i,j)$ of going from state i to state j at time t is given by the probability of producing the output sequence O and going from state i to state j at time t divided by the probability of producing the output sequence O.

$$\delta_t(i,j) = \frac{\alpha_t(i)a_{ij}b_j(O_{t+1})\beta_{t+1}(j)}{p(O)}$$

The probability $p(O)$ is the sum over all pairs of states i and j of the numerator in the above formula for $\delta_t(i,j)$. That is,

$$p(O) = \sum_i \sum_j \alpha_t(j)a_{ij}b_j(O_{t+1})\beta_{t+1}(j).$$

The probability of being in state i at time t is given by

$$s_t(i) = \sum_{j=1}^{n} \delta_t(i,j).$$

Summing $s_t(i)$ over all time periods gives the expected number of times state i is visited and the sum of $\delta_t(i,j)$ over all time periods gives the expected number of times edge i to j is traversed.

Given estimates of the HMM parameters $a_{i,j}$ and $b_j(O_k)$, we can calculate by the above formulas estimates for

1. $\sum_{i=1}^{T-1} s_t(i)$, the expected number of times state i is visited and departed from

2. $\sum_{i=1}^{T-1} \delta_t(i,j)$, the expected number of transitions from state i to state j

Using these estimates we can obtain new estimates of the HMM parameters

$$\overline{a_{ij}} = \frac{\text{expected number of transitions from state } i \text{ to state } j}{\text{expected number of transitions out of state } i} = \frac{\sum_{t=1}^{T-1} \delta_t(i,j)}{\sum_{t=1}^{T-1} s_t(i)}$$

$$\overline{b_j}(O_k) = \frac{\text{expected number of times in state } j \text{ observing symbol } O_k}{\text{expected number of times in state } j} = \frac{\sum_{\substack{t=1 \\ \text{subject to} \\ O_t=O_k}}^{T-1} s_t(j)}{\sum_{t=1}^{T-1} s_t(j)}$$

By iterating the above formulas we can arrive at a local optimum for the HMM parameters $a_{i,j}$ and $b_j(O_k)$.

9.11 Graphical Models and Belief Propagation

A graphical model is a compact representation of a probability distribution over n variables $x_1, x_2, \ldots, x_n$. It consists of a graph, directed or undirected, whose vertices correspond to variables that take on values from some set. In this chapter, we consider the case where the set of values the variables take on is finite, although graphical models are often used to represent probability distributions with continuous variables. The edges of the graph represent relationships or constraints between the variables.

In the directed model, it is assumed that the directed graph is acyclic. This model represents a joint probability distribution that factors into a product of conditional probabilities.

$$p(x_1, x_2, \ldots, x_n) = \prod_{i=1}^{n} p(x_i | \text{parents of } x_i)$$

The directed graphical model is called a *Bayesian* or *belief network* and appears frequently in the artificial intelligence and the statistics literature.

The undirected graphical model, called a *Markov random field*, can also represent a joint probability distribution of the random variables at its vertices. In many applications the Markov random field represents a function of the variables at the vertices which is to be optimized by choosing values for the variables.

A third model called the *factor model* is akin to the Markov random field, but here the dependency sets have a different structure. In the following sections we describe all these models in more detail.

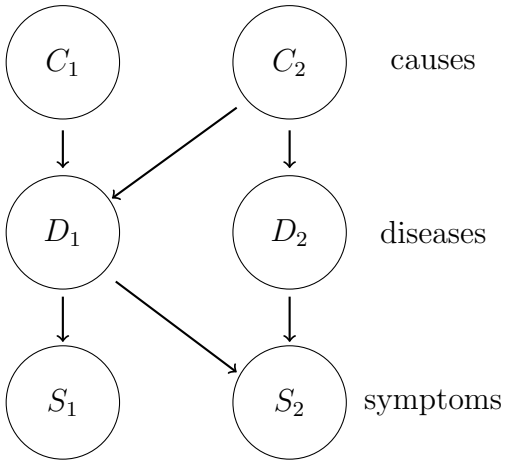


Figure 9.2: A Bayesian network

9.12 Bayesian or Belief Networks

A *Bayesian network* is a directed acyclic graph where vertices correspond to variables and a directed edge from y to x represents a conditional probability $p(x|y)$. If a vertex x has edges into it from $y_1, y_2, \ldots, y_k$, then the conditional probability is $p\left(x \mid y_1, y_2, \ldots, y_k\right)$. The variable at a vertex with no in edges has an unconditional probability distribution. If the value of a variable at some vertex is known, then the variable is called *evidence*. An important property of a Bayesian network is that the joint probability is given by the product over all nodes of the conditional probability of the node conditioned on all its immediate predecessors.

In the example of Fig. 9.1, a patient is ill and sees a doctor. The doctor ascertains the symptoms of the patient and the possible causes such as whether the patient was in contact with farm animals, whether he had eaten certain foods, or whether the patient has an hereditary predisposition to any diseases. Using the above Bayesian network where the variables are true or false, the doctor may wish to determine one of two things. What is the marginal probability of a given disease or what is the most likely set of diseases. In determining the most likely set of diseases, we are given a T or F assignment to the causes and symptoms and ask what assignment of T or F to the diseases maximizes the joint probability. This latter problem is called the *maximum a posteriori probability* (MAP).

Given the conditional probabilities and the probabilities $p\left(C_1\right)$ and $p\left(C_2\right)$ in Figure 9.1, the joint probability $p\left(C_1, C_2, D_1, \ldots\right)$ can be computed easily for any combination of values of $C_1, C_2, D_1, \ldots$. However, we might wish to find the value of the variables of highest probability (MAP) or we might want one of the marginal probabilities $p\left(D_1\right)$ or $p\left(D_2\right)$. The obvious algorithms for these two problems require evaluating the probability $p\left(C_1, C_2, D_1, \ldots\right)$ over exponentially many input values or summing the probability $p\left(C_1, C_2, D_1, \ldots\right)$ over exponentially many values of the variables other than those for

which we want the marginal probability. In certain situations, when the joint probability distribution can be expressed as a product of factors, a belief propagation algorithm can solve the maximum a posteriori problem or compute all marginal probabilities quickly.

9.13 Markov Random Fields

The Markov random field model arose first in statistical mechanics where it was called the Ising model. It is instructive to start with a description of it. The simplest version of the Ising model consists of n particles arranged in a rectangular $\sqrt{n} \times \sqrt{n}$ grid. Each particle can have a spin that is denoted ± 1. The energy of the whole system depends on interactions between pairs of neighboring particles. Let x_i be the spin, ± 1, of the i^{th} particle. Denote by $i \sim j$ the relation that i and j are adjacent in the grid. In the Ising model, the energy of the system is given by

$$f(x_1, x_2, \ldots, x_n) = \exp\left(c \sum_{i \sim j} |x_i - x_j|\right).$$

The constant c can be positive or negative. If $c < 0$, then energy is lower if many adjacent pairs have opposite spins and if $c > 0$ the reverse holds. The model was first used to model probabilities of spin configurations in physical materials.

In most computer science settings, such functions are mainly used as objective functions that are to be optimized subject to some constraints. The problem is to find the minimum energy set of spins under some constraints on the spins. Usually the constraints just specify the spins of some particles. Note that when $c > 0$, this is the problem of minimizing $\sum_{i \sim j} |x_i - x_j|$ subject to the constraints. The objective function is convex and so this can be done efficiently. If $c < 0$, however, we need to minimize a concave function for which there is no known efficient algorithm. The minimization of a concave function in general is NP-hard. Intuitively, this is because the set of inputs for which $f(x)$ is less than some given value can be nonconvex or even consist of many disconnected regions.

A second important motivation comes from the area of vision. It has to to do with reconstructing images. Suppose we are given noisy observations of the intensity of light at individual pixels, $x_1, x_2, \ldots, x_n$, and wish to compute the true values, the true intensities, of these variables $y_1, y_2, \ldots, y_n$. There may be two sets of constraints, the first stipulating that the y_i should generally be close to the corresponding x_i and the second, a term correcting possible observation errors, stipulating that y_i should generally be close to the values of y_j for $j \sim i$. This can be formulated as

$$\min_{\mathbf{y}} \left(\sum_i |x_i - y_i| + \sum_{i \sim j} |y_i - y_j| \right),$$

where the values of x_i are constrained to be the observed values. The objective function is convex and polynomial time minimization algorithms exist. Other objective functions

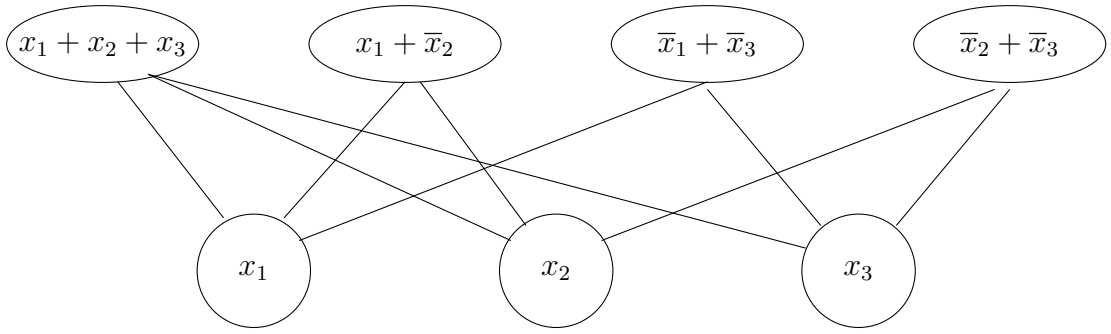


Figure 9.3: The factor graph for the function
$f(x_1, x_2, x_3) = (x_1 + x_2 + x_3)(x_1 + \bar{x}_2)(x_1 + \bar{x}_3)(\bar{x}_2 + \bar{x}_3)$.

using say sum of squares instead of sum of absolute values can be used and there are polynomial time algorithms as long as the function to be minimized is convex.

More generally, the correction term may depend on all grid points within distance two of each point rather than just immediate neighbors. Even more generally, we may have n variables $y_1, y_2, \ldots y_n$ with the value of some of them already specified and subsets $S_1, S_2, \ldots S_m$ of these variables constrained in some way. The constraints are accumulated into one objective function which is a product of functions $f_1, f_2, \ldots, f_m$, where function f_i is evaluated on the variables in subset S_i. The problem is to minimize $\prod_{i=1}^{m} f_i(y_j, j \in S_i)$ subject to constrained values. Note that the vision example had a sum instead of a product, but by taking exponentials we can turn the sum into a product as in the Ising model.

In general, the f_i are not convex; indeed they may be discrete. So the minimization cannot be carried out by a known polynomial time algorithm. The most used forms of the Markov random field involve S_i which are cliques of a graph. So we make the following definition.

A *Markov Random Field* consists of an undirected graph and an associated function that factorizes into functions associated with the cliques of the graph. The special case when all the factors correspond to cliques of size one or two is of interest.

9.14 Factor Graphs

Factor graphs arise when we have a function f of a variables $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ that can be expressed as $f(\mathbf{x}) = \prod_{\alpha} f_\alpha(\mathbf{x}_\alpha)$ where each factor depends only on some small number of variables $\mathbf{x}_\alpha$. The difference from Markov random fields is that the variables corresponding to factors do not necessarily form a clique. Associate a bipartite graph where one set of vertices correspond to the factors and the other set to the variables.

Place an edge between a variable and a factor if the factor contains that variable. See Figure 9.3.

9.15 Tree Algorithms

Let $f(\mathbf{x})$ be a function that is a product of factors. When the factor graph is a tree there are efficient algorithms for solving certain problems. With slight modifications, the algorithms presented can also solve problems where the function is the sum of terms rather than a product of factors.

The first problem is called *marginalization* and involves evaluating the sum of f over all variables except one. In the case where f is a probability distribution the algorithm computes the marginal probabilities and thus the word marginalization. The second problem involves computing the assignment to the variables that maximizes the function f. When f is a probability distribution, this problem is the maximum a posteriori probability or MAP problem.

If the factor graph is a tree (such as in Figure 9.4), then there exists an efficient algorithm for solving these problems. Note that there are four problems: the function f is either a product or a sum and we are either marginalizing or finding the maximizing assignment to the variables. All four problems are solved by essentially the same algorithm and we present the algorithm for the marginalization problem when f is a product. Assume we want to "sum out" all the variables except x_1, leaving a function of x_1.

Call the variable node associated with some variable x_i node x_i. First, make the node x_1 the root of the tree. It will be useful to think of the algorithm first as a recursive algorithm and then unravel the recursion. We want to compute the product of all factors occurring in the sub-tree rooted at the root with all variables except the root-variable summed out. Let g_i be the product of all factors occurring in the sub-tree rooted at node x_i with all variables occurring in the subtree except x_i summed out. Since this is a tree, x_1 will not reoccur anywhere except the root. Now, the grandchildren of the root are variable nodes and suppose inductively, each grandchild x_i of the root, has already computed its g_i. It is easy to see that we can compute g_1 as follows.

Each grandchild x_i of the root passes its g_i to its parent, which is a factor node. Each child of x_1 collects all its children's g_i, multiplies them together with its own factor and sends the product to the root. The root multiplies all the products it gets from its children and sums out all variables except its own variable, namely here x_1.

Unraveling the recursion is also simple, with the convention that a leaf node just receives 1, product of an empty set of factors, from its children. Each node waits until it receives a message from each of its children. After that, if the node is a variable node, it computes the product of all incoming messages, and sums this product function over

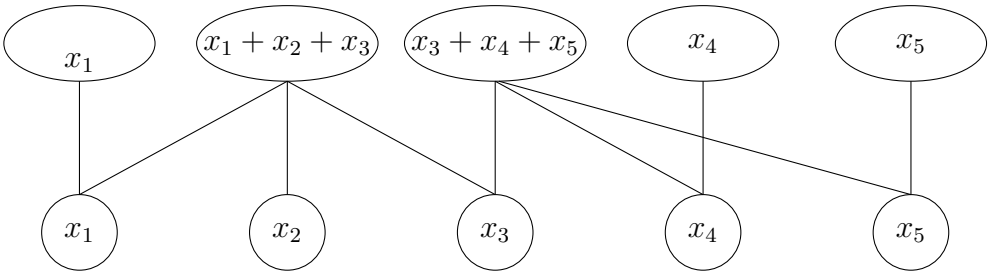


Figure 9.4: The factor graph for the function $f = x_1 (x_1 + x_2 + x_3) (x_3 + x_4 + x_5) x_4 x_5$.

all assignments to the variables except for the variable of the node. Then, it sends the resulting function of one variable out along the edge to its parent. If the node is a factor node, it computes the product of its factor function along with incoming messages from all the children and sends the resulting function out along the edge to its parent.

The reader should prove that the following invariant holds assuming the graph is a tree:

Invariant The message passed by each variable node to its parent is the product of all factors in the subtree under the node with all variables in the subtree except its own summed out.

Consider the following example where

$$f = x_1 (x_1 + x_2 + x_3) (x_3 + x_4 + x_5) x_4 x_5$$

and the variables take on values 0 or 1. Consider marginalizing f by computing

$$f (x_1) = \sum_{x_2 x_3 x_4 x_5} x_1 (x_1 + x_2 + x_3) (x_3 + x_4 + x_5) x_4 x_5,$$

In this case the factor graph is a tree as shown in Figure 9.4. The factor graph as a rooted tree and the messages passed by each node to its parent are shown in Figure 9.5. If instead of computing marginals, one wanted the variable assignment that maximizes the function f, one would modify the above procedure by replacing the summation by a maximization operation. Obvious modifications handle the situation where $f(\mathbf{x})$ is a sum of products.

$$f (\mathbf{x}) = \sum_{x_1, \ldots, x_n} g (\mathbf{x})$$

9.16 Message Passing in General Graphs

The simple message passing algorithm in the last section gives us the one variable function of x_1 when we sum out all the other variables. For a general graph that is not

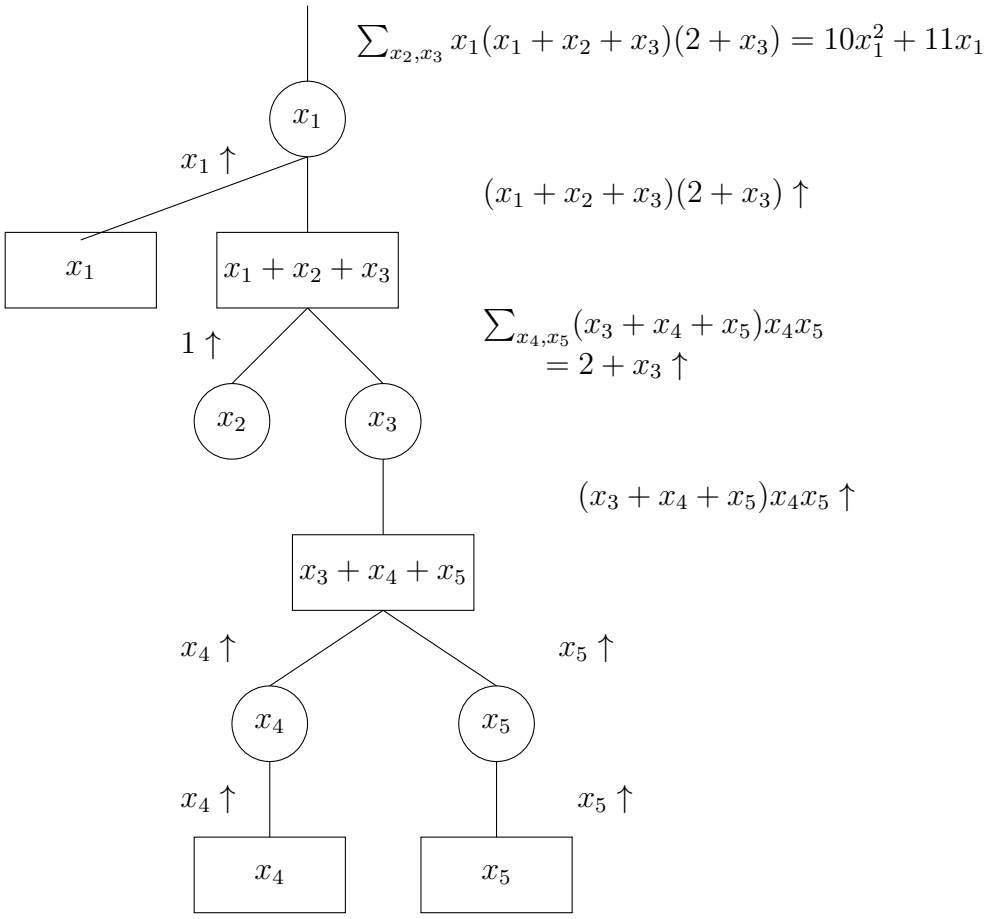


Figure 9.5: Messages.

a tree, we formulate an extension of that algorithm. But unlike the case of trees, there is no proof that the algorithm will converge and even if it does, there is no guarantee that the limit is the marginal probability. This has not prevented its usefulness in some applications.

First, lets ask a more general question, just for trees. Suppose we want to compute for each i the one-variable function of x_i when we sum out all variables $x_j, j \neq i$. Do we have to repeat what we did for x_1 once for each x_i? Luckily, the answer is no. It will suffice to do a second pass from the root to the leaves of essentially the same message passing algorithm to get *all* the answers. Recall that in the first pass, each edge of the tree has sent a message "up", from the child to the parent. In the second pass, each edge will send a message down from the parent to the child. We start with the root and work downwards for this pass. Each node waits until its parent has sent it a message before sending messages to each of its children. The rules for messages are:

Rule 1 The message from a factor node v to a child x_i, which is the variable node x_i, is the product of all messages received by v in both passes from all nodes other than x_i times the factor at v itself.

Rule 2 The message from a variable node x_i to a factor node child, v, is the product of all messages received by x_i in both passes from all nodes except v, with all variables except x_i summed out. The message is a function of x_i alone.

At termination, when the graph is a tree, if we take the product of all messages received in both passes by a variable node x_i and sum out all variables except x_i in this product, what we get is precisely the entire function marginalized to x_i. We do not give the proof here. But the idea is simple. We know from the first pass that the product of the messages coming to a variable node x_i from its children is the product of all factors in the sub-tree rooted at x_i. In the second pass, we claim that the message from the parent v to x_i is the product of all factors which are not in the sub-tree rooted at x_i which one can show either directly or by induction working from the root downwards.

We can apply the same rules 1 and 2 to any general graph. We do not have child and parent relationships and it is not possible to have the two synchronous passes as before. The messages keep flowing and one hopes that after some time, the messages will stabilize, but nothing like that is proven. We state the algorithm for general graphs now:

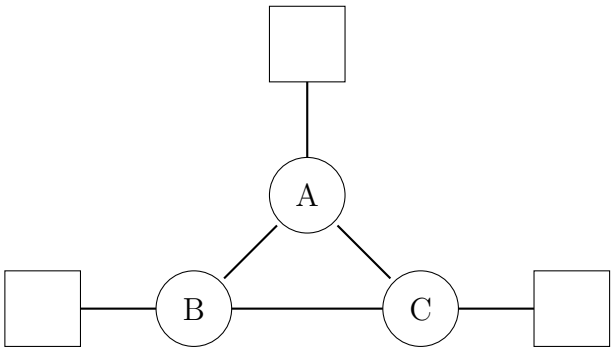
Rule 1 At each unit of time, each factor node v sends a message to each adjacent node x_i. The message is the product of all messages received by v at the previous step except for the one from x_i multiplied by the factor at v itself.

Rule 2 At each time, each variable node x_i sends a message to each adjacent node v. The message is the product of all messages received by x_i at the previous step except the one from v, with all variables except x_i summed out.

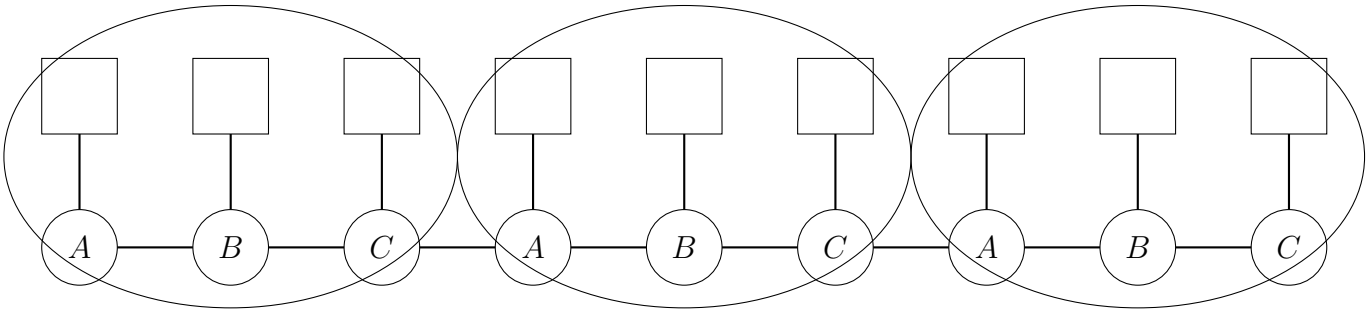
9.17 Graphs with a Single Cycle

The message passing algorithm gives the correct answers on trees and on certain other graphs. One such situation is graphs with a single cycle which we treat here. We switch from the marginalization problem to the MAP problem as the proof of correctness is simpler for the MAP problem. Consider the network in Figure 9.6a with a single cycle. The message passing scheme will count some evidence multiply. The local evidence at A will get passed around the loop and will come back to A. Thus, A will count the local evidence multiple times. If all evidence is multiply counted in equal amounts, then there is a possibility that though the numerical values of the marginal probabilities (beliefs) are wrong, the algorithm still converges to the correct maximum a posteriori assignment.

Consider the unwrapped version of the graph in Figure 9.6b. The messages that the



(a) A graph with a single cycle



(b) Segment of unrolled graph

Figure 9.6: Unwrapping a graph with a single cycle

loopy version will eventually converge to, assuming convergence, are the same messages that occur in the unwrapped version provided that the nodes are sufficiently far in from the ends. The beliefs in the unwrapped version are correct for the unwrapped graph since it is a tree. The only question is, how similar are they to the true beliefs in the original network.

Write $p(A, B, C) = e^{\log p(A,B,C)} = e^{J(A,B,C)}$ where $J(A, B, C) = \log p(A, B, C)$. Then the probability for the unwrapped network is of the form $e^{kJ(A,B,C)+J'}$ where the J' is associated with vertices at the ends of the network where the beliefs have not yet stabilized and the $kJ(A, B, C)$ comes from k inner copies of the cycle where the beliefs have stabilized. Note that the last copy of J in the unwrapped network shares an edge with J' and that edge has an associated Ψ. Thus, changing a variable in J has an impact on the value of J' through that Ψ. Since the algorithm maximizes $J_k = kJ(A, B, C) + J'$ in the unwrapped network for all k, it must maximize $J(A, B, C)$. To see this, set the variables A, B, C, so that J_k is maximized. If $J(A, B, C)$ is not maximized, then change A, B, and C to maximize $J(A, B, C)$. This increases J_k by some quantity that is proportional to k. However, two of the variables that appear in copies of $J(A, B, C)$ also appear in J'

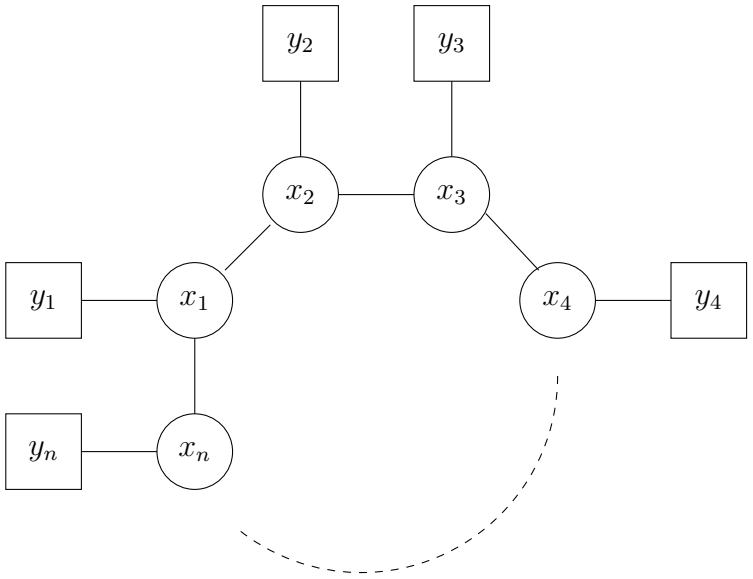


Figure 9.7: A Markov random field with a single loop.

and thus J' might decrease in value. As long as J' decreases by some finite amount, we can increase J_k by increasing k sufficiently. As long as all Ψ's are nonzero, J' which is proportional to $\log \Psi$, can change by at most some finite amount. Hence, for a network with a single loop, assuming that the message passing algorithm converges, it converges to the maximum a posteriori assignment.

9.18 Belief Update in Networks with a Single Loop

In the previous section, we showed that when the message passing algorithm converges, it correctly solves the MAP problem for graphs with a single loop. The message passing algorithm can also be used to obtain the correct answer for the marginalization problem. Consider a network consisting of a single loop with variables $x_1, x_2, \ldots, x_n$ and evidence $y_1, y_2, \ldots, y_n$ as shown in Figure 9.7. The x_i and y_i can be represented by vectors having a component for each value x_i can take on. To simplify the discussion assume the x_i take on values $1, 2, \ldots, m$.

Let m_i be the message sent from vertex i to vertex $i + 1$ mod n. At vertex $i + 1$ each component of the message m_i is multiplied by the evidence y_{i+1} and the constraint function Ψ. This is done by forming a diagonal matrix D_{i+1} where the diagonal elements are the evidence and then forming a matrix M_i whose jk^{th} element is $\Psi\left(x_{i+1} = j,\ x_i = k\right)$. The message m_{i+1} is $M_i D_{i+1} m_i$. Multiplication by the diagonal matrix D_{i+1} multiplies the components of the message m_i by the associated evidence. Multiplication by the matrix M_i multiplies each component of the vector by the appropriate value of Ψ and sums over the values producing the vector which is the message m_{i+1}. Once the message

has travelled around the loop, the new message m_1' is given by

$$m_1' = M_n D_1 M_{n-1} D_n \cdots M_2 D_3 M_1 D_2 m_1$$

Let $M = M_n D_1 M_{n-1} D_n \cdots M_2 D_3 M_1 D_2 m_1$. Assuming that M's principal eigenvalue is unique, the message passing will converge to the principal vector of M. The rate of convergences depends on the ratio of the first and second eigenvalues.

An argument analogous to the above concerning the messages going clockwise around the loop applies to messages moving counter-clockwise around the loop. To obtain the estimate of the marginal probability $p(x_1)$, one multiples component-wise the two messages arriving at x_1 along with the evidence y_1. This estimate does not give the true marginal probability but the true marginal probability can be computed from the estimate and the rate of convergences by linear algebra.

9.19 Maximum Weight Matching

We have seen that the belief propagation algorithm converges to the correct solution in trees and graphs with a single cycle. It also correctly converges for a number of problems. Here we give one example, the maximum weight matching problem where there is a unique solution.

We apply the belief propagation algorithm to find the maximal weight matching (MWM) in a complete bipartite graph. If the MWM in the bipartite graph is unique, then the belief propagation algorithm will converge to it.

Let $G = (V_1, V_2, E)$ be a complete bipartite graph where $V_1 = \{a_1, \ldots, a_n\}$, $V_2 = \{b_1, \ldots, b_n\}$, and $(a_i, b_j) \in E, \quad 1 \leq i, j \leq n$. Let $\pi = \{\pi(1), \ldots, \pi(n)\}$ be a permutation of $\{1, \ldots, n\}$. The collection of edges $\left\{\left(a_1, b_{\pi(1)}\right), \ldots, \left(a_n, b_{\pi(n)}\right)\right\}$is called a *matching* which is denoted by π. Let w_{ij} be the weight associated with the edge (a_i, b_j). The weight of the matching π is $w_\pi = \sum_{i=1}^{n} w_{i\pi(i)}$. The maximum weight matching π^* is $\pi^* = \arg\max_{\pi} w_\pi$

The first step is to create a factor graph corresponding to the MWM problem. Each edge of the bipartite graph is represented by a variable c_{ij} which takes on the value zero or one. The value one means that the edge is present in the matching, the value zero means that the edge is not present in the matching. A set of constraints is used to force the set of edges to be a matching. The constraints are of the form $\sum_j c_{ij} = 1$ and $\sum_i c_{ij} = 1$. Any 0,1 assignment to the variables c_{ij} that satisfies all of the constraints defines a matching. In addition, we have constraints for the weights of the edges.

We now construct a factor graph, a portion of which is shown in Figure 9.8. Associated with the factor graph is a function $f(c_{11}, c_{12}, \ldots)$ consisting of a set of terms for each c_{ij}

enforcing the constraints and summing the weights of the edges of the matching. The terms for c_{12} are

$$-\lambda \left| \left(\sum_i c_{i2} \right) - 1 \right| - \lambda \left| \left(\sum_j c_{1j} \right) - 1 \right| + w_{12} c_{12}$$

where λ is a large positive number used to enforce the constraints when we maximize the function. Finding the values of $c_{11}, c_{12}, \ldots$ that maximize f finds the maximum weighted matching for the bipartite graph.

If the factor graph was a tree, then the message from a variable node x to its parent is a message $g(x)$ that gives the maximum value for the subtree for each value of x. To compute $g(x)$, one sums all messages into the node x. For a constraint node, one sums all messages from subtrees and maximizes the sum over all variables except the variable of the parent node subject to the constraint. The message from a variable x consists of two pieces of information, the value $p(x = 0)$ and the value $p(x = 1)$. This information can be encoded into a linear function of x:

$$[p(x = 1) - p(x = 0)] x + p(x = 0).$$

Thus, the messages are of the form $ax + b$. To determine the MAP value of x once the algorithm converges, sum all messages into x and take the maximum over x=1 and x=0 to determine the value for x. Since the arg maximum of a linear form ax+b depends only on whether a is positive or negative and since maximizing the output of a constraint depends only on the coefficient of the variable, we can send messages consisting of just the variable coefficient.

To calculate the message to c_{12} from the constraint that node b_2 has exactly one neighbor, add all the messages that flow into the constraint node from the c_{i2}, $i \neq 1$ nodes and maximize subject to the constraint that exactly one variable has value one. If $c_{12} = 0$, then one of c_{i2}, $i \neq 1$, will have value one and the message is $\max_{i \neq 1} \alpha(i, 2)$. If $c_{12} = 1$, then the message is zero. Thus, we get

$$-\max_{i \neq 1} \alpha(i, 2) x + \max_{i \neq 1} \alpha(i, 2)$$

and send the coefficient $-\max_{i \neq 1} \alpha(i, 2)$. This means that the message from c_{12} to the other constraint node is $\beta(1, 2) = w_{12} - \max_{i \neq 1} \alpha(i, 2)$.

The alpha message is calculated in a similar fashion. If $c_{12} = 0$, then one of c_{1j} will have value one and the message is $\max_{j \neq 1} \beta(1, j)$. If $c_{12} = 1$, then the message is zero. Thus, the coefficient $-\max_{j \neq 1} \alpha(1, j)$ is sent. This means that $\alpha(1, 2) = w_{12} - \max_{j \neq 1} \alpha(1, j)$.

To prove convergence, we unroll the constraint graph to form a tree with a constraint node as the root. In the unrolled graph a variable node such as c_{12} will appear a number

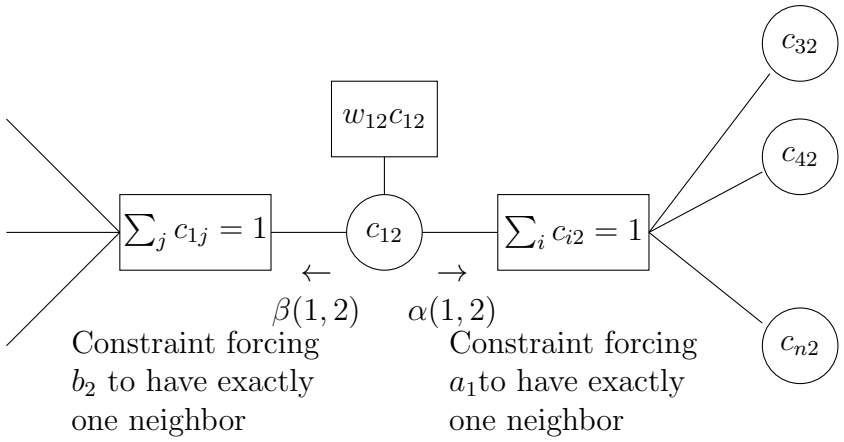


Figure 9.8: Portion of factor graph for the maximum weight matching problem.

of times which depends on how deep a tree is built. Each occurrence of a variable such as c_{12} is deemed to be a distinct variable.

Lemma 9.14 *If the tree obtained by unrolling the graph is of depth k, then the messages to the root are the same as the messages in the constraint graph after k-iterations.*

Proof: Straightforward. ∎

Define a matching in the tree to be a set of vertices so that there is exactly one variable node of the match adjacent to each constraint. Let Λ denote the vertices of the matching. Heavy circles in Figure 9.9 represent the nodes of the above tree that are in the matching Λ.

Let Π be the vertices corresponding to maximum weight matching edges in the bipartite graph. Recall that vertices in the above tree correspond to edges in the bipartite graph. The vertices of Π are denoted by dotted circles in the above tree.

Consider a set of trees where each tree has a root that corresponds to one of the constraints. If the constraint at each root is satisfied by the edge of the MWM, then we have found the MWM. Suppose that the matching at the root in one of the trees disagrees with the MWM. Then there is an alternating path of vertices of length $2k$ consisting of vertices corresponding to edges in Π and edges in Λ. Map this path onto the bipartite graph. In the bipartite graph the path will consist of a number of cycles plus a simple path. If k is large enough there will be a large number of cycles since no cycle can be of length more than $2n$. Let m be the number of cycles. Then $m \geq \frac{2k}{2n} = \frac{k}{n}$.

Let π^* be the MWM in the bipartite graph. Take one of the cycles and use it as an alternating path to convert the MWM to another matching. Assuming that the MWM is unique and that the next closest matching is ε less, $W_{\pi *} - W_{\pi} > \varepsilon$ where π is the new

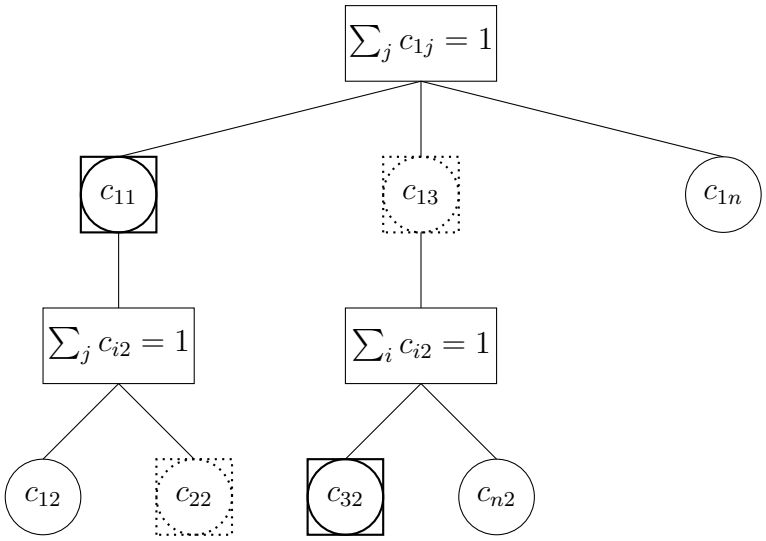


Figure 9.9: Tree for MWM problem.

matching.

Consider the tree matching. Modify the tree matching by using the alternating path of all cycles and the left over simple path. The simple path is converted to a cycle by adding two edges. The cost of the two edges is at most 2w* where w* is the weight of the maximum weight edge. Each time we modify Λ by an alternating cycle, we increase the cost of the matching by at least ε. When we modify Λ by the left over simple path, we increase the cost of the tree matching by $\varepsilon - 2w*$ since the two edges that were used to create a cycle in the bipartite graph are not used. Thus

$$\text{weight of } \Lambda \text{ - weight of } \Lambda' \geq \tfrac{k}{n}\varepsilon - 2w*$$

which must be negative since Λ' is optimal for the tree. However, if k is large enough this becomes positive, an impossibility since Λ' is the best possible. Since we have a tree, there can be no cycles, as messages are passed up the tree, each subtree is optimal and hence the total tree is optimal. Thus the message passing algorithm must find the maximum weight matching in the weighted complete bipartite graph assuming that the maximum weight matching is unique. Note that applying one of the cycles that makes up the alternating path decreased the bipartite graph match but increases the value of the tree. However, it does not give a higher tree matching, which is not possible since we already have the maximum tree matching. The reason for this is that the application of a single cycle does not result in a valid tree matching. One must apply the entire alternating path to go from one matching to another.

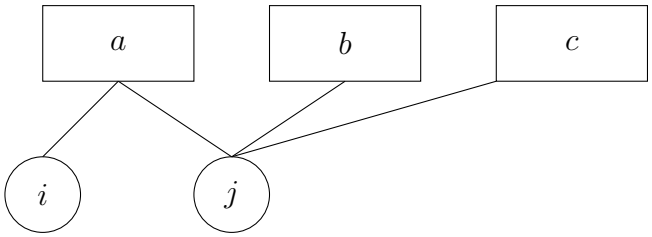


Figure 9.10: warning propagation

9.20 Warning Propagation

Significant progress has been made using methods similar to belief propagation in finding satisfying assignments for 3-CNF formulas. Thus, we include a section on a version of belief propagation, called warning propagation, that is quite effective in finding assignments. Consider a factor graph for a SAT problem (Figure 9.10). Index the variables by i, j, and k and the factors by a, b, and c. Factor a sends a message m_{ai} to each variable i that appears in the factor a called a warning. The warning is 0 or 1 depending on whether or not factor a believes that the value assigned to i is required for a to be satisfied. A factor a determines the warning to send to variable i by examining all warnings received by other variables in factor a from factors containing them.

For each variable j, sum the warnings from factors containing j that warn j to take value T and subtract the warnings that warn j to take value F. If the difference says that j should take value T or F and this value for variable j does not satisfy a, and this is true for all j, then a sends a warning to i that the value of variable i is critical for factor a.

Start the warning propagation algorithm by assigning 1 to a warning with probability 1/2. Iteratively update the warnings. If the warning propagation algorithm converges, then compute for each variable i the local field h_i and the contradiction number c_i. The local field h_i is the number of clauses containing the variable i that sent messages that i should take value T minus the number that sent messages that i should take value F. The contradiction number c_i is 1 if variable i gets conflicting warnings and 0 otherwise. If the factor graph is a tree, the warning propagation algorithm converges. If one of the warning messages is one, the problem is unsatisfiable; otherwise it is satisfiable.

9.21 Correlation Between Variables

In many situations one is interested in how the correlation between variables drops off with some measure of distance. Consider a factor graph for a 3-CNF formula. Measure the distance between two variables by the shortest path in the factor graph. One might ask if one variable is assigned the value true, what is the percentage of satisfying assignments of the 3-CNF formula in which the second variable also is true. If the percentage is the same as when the first variable is assigned false, then we say that the two variables

are uncorrelated. How difficult it is to solve a problem is likely to be related to how fast the correlation decreases with distance.

Another illustration of this concept is in counting the number of perfect matchings in a graph. One might ask what is the percentage of matching in which some edge is present and ask how correlated this percentage is with the presences or absence of edges at some distance d. One is interested in whether the correlation drops off with distance. To explore this concept we consider the Ising model studied in physics.

As mentioned earlier, the Ising or ferromagnetic model is a pairwise random Markov field. The underlying graph, usually a lattice, assigns a value of ± 1, called spin, to the variable at each vertex. The probability (Gibbs measure) of a given configuration of spins is proportional to $exp(\beta \sum_{(i,j)\in E} x_i x_j) = \prod_{(i,j)\in E} e^{\beta x_i x_j}$ where $x_i = \pm 1$ is the value associated with vertex i. Thus

$$p\left(x_1, x_2, \ldots, x_n\right) = \frac{1}{Z} \prod_{(i,j)\in E} exp(\beta x_i x_j) = \frac{1}{Z} e^{\beta \sum_{(i,j)\in E} x_i x_j}$$

where Z is a normalization constant.

The value of the summation is simply the difference in the number of edges whose vertices have the same spin minus the number of edges whose vertices have opposite spin. The constant β is viewed as inverse temperature. High temperature corresponds to a low value of β and low temperature corresponds to a high value of β. At high temperature, low β, the spins of adjacent vertices are uncorrelated whereas at low temperature adjacent vertices have identical spins. The reason for this is that the probability of a configuration is proportional to $e^{\beta \sum_{i\sim j} x_i x_j}$. As β is increased, for configurations with a large number of edges whose vertices have identical spins, $e^{\beta \sum_{i\sim j} x_i x_j}$ increases more than for configurations whose edges have vertices with non identical spins. When the normalization constant $\frac{1}{Z}$ is adjusted for the new value of β, the highest probability configurations are those where adjacent vertices have identical spins.

Given the above probability distribution, what is the correlation between two variables x_i and x_j? To answer this question, consider the probability that $x_i = +1$ as a function of the probability that $x_j = +1$. If the probability that $x_i = +1$ is $\frac{1}{2}$ independent of the value of the probability that $x_j = +1$, we say the values are uncorrelated.

Consider the special case where the graph G is a tree. In this case a phase transition occurs at $\beta_0 = \frac{1}{2} \ln \frac{d+1}{d-1}$ where d is the degree of the tree. For a sufficiently tall tree and for $\beta > \beta_0$, the probability that the root has value +1 is bounded away from $1/2$ and depends on whether the majority of leaves have value +1 or -1. For $\beta < \beta_0$ the probability that the root has value +1 is $1/2$ independent of the values at the leaves of the tree.

Consider a height one tree of degree d. If i of the leaves have spin +1 and $d-i$ have spin -1, then the probability of the root having spin +1 is proportional to

$$e^{i\beta-(d-i)\beta} = e^{(2i-d)\beta}.$$

If the probability of a leaf being +1 is p, then the probability of i leaves being +1 and $d-i$ being -1 is

$$\binom{d}{i} p^i \left(1-p\right)^{d-i}$$

Thus, the probability of the root being +1 is proportional to

$$A = \sum_{i=1}^{d} \binom{d}{i} p^i (1-p)^{d-i} e^{(2i-d)\beta} = e^{-d\beta} \sum_{i=1}^{d} \binom{d}{i} \left(pe^{2\beta}\right)^i \left(1-p\right)^{d-i} = e^{-d\beta} \left[pe^{2\beta}+1-p\right]^d$$

and the probability of the root being –1 is proportional to

$$\begin{aligned} B &= \sum_{i=1}^{d} \binom{d}{i} p^i (1-p)^{d-i} e^{-(2i-d)\beta} \\ &= e^{-d\beta} \sum_{i=1}^{d} \binom{d}{i} p^i \left[(1-p)e^{-2(i-d)\beta}\right] \\ &= e^{-d\beta} \sum_{i=1}^{d} \binom{d}{i} p^i \left[(1-p)e^{2\beta}\right]^{d-i} \\ &= e^{-d\beta} \left[p+(1-p)e^{2\beta}\right]^d. \end{aligned}$$

The probability of the root being +1 is

$$q = \frac{A}{A+B} = \frac{\left[pe^{2\beta}+1-p\right]^d}{\left[pe^{2\beta}+1-p\right]^d + \left[p+(1-p)e^{2\beta}\right]^d} = \frac{C}{D}$$

where

$$C = \left[pe^{2\beta}+1-p\right]^d$$

and

$$D = \left[pe^{2\beta}+1-p\right]^d + \left[p+(1-p)\,e^{2\beta}\right]^d.$$

At high temperature, low β, the probability q of the root of the height one tree being +1 in the limit as β goes to zero is

$$q = \frac{p+1-p}{[p+1-p]+[p+1-p]} = \frac{1}{2}$$

independent of p. At low temperature, high β,

$$q \approx \frac{p^d e^{2\beta d}}{p^d e^{2\beta d} + (1-p)^d e^{2\beta d}} = \frac{p^d}{p^d + (1-p)^d} = \begin{cases} 0 & p=0 \\ 1 & p=1 \end{cases}.$$

q goes from a low probability of +1 for p below 1/2 to high probability of +1 for p above 1/2.

Now consider a very tall tree. If the p is the probability that a root has value +1, we can iterate the formula for the height one tree and observe that at low temperature the probability of the root being one converges to some value. At high temperature, the probability of the root being one is $1/2$ independent of p. At the phase transition, the slope of q at p=1/2 is one. See Figure 9.11.

Now the slope of the probability of the root being 1 with respect to the probability of a leaf being 1 in this height one tree is

$$\frac{\partial q}{\partial p} = \frac{D\frac{\partial C}{\partial p} - C\frac{\partial D}{\partial p}}{D^2}$$

Since the slope of the function $q(p)$ at p=1/2 when the phase transition occurs is one, we can solve $\frac{\partial q}{\partial p} = 1$ for the value of β where the phase transition occurs. First, we show that $\frac{\partial D}{\partial p}\Big|_{p=\frac{1}{2}} = 0$.

$$\begin{aligned} D &= \left[pe^{2\beta} + 1 - p\right]^d + \left[p + (1-p)\,e^{2\beta}\right]^d \\ \frac{\partial D}{\partial p} &= d\left[pe^{2\beta} + 1 - p\right]^{d-1}\left(e^{2\beta} - 1\right) + d\left[p + (1-p)\,e^{2\beta}\right]^{d-1}\left(1 - e^{2\beta}\right) \\ \frac{\partial D}{\partial p}\Big|_{p=\frac{1}{2}} &= \tfrac{d}{2^{d-1}}\left[e^{2\beta} + 1\right]^{d-1}\left(e^{2\beta} - 1\right) + \tfrac{d}{2^{d-1}}\left[1 + e^{2\beta}\right]^{d-1}\left(1 - e^{2\beta}\right) = 0 \end{aligned}$$

Then

$$\begin{aligned} \frac{\partial q}{\partial p}\Big|_{p=\frac{1}{2}} &= \frac{D\frac{\partial C}{\partial p} - C\frac{\partial D}{\partial p}}{D^2}\Big|_{p=\frac{1}{2}} = \frac{\frac{\partial C}{\partial p}}{D}\Big|_{p=\frac{1}{2}} = \frac{d\left[pe^{2\beta} + 1 - p\right]^{d-1}\left(e^{2\beta} - 1\right)}{\left[pe^{2\beta} + 1 - p\right]^d + \left[p + (1-p)\,e^{2\beta}\right]^d}\Big|_{p=\frac{1}{2}} \\ &= \frac{d\left[\frac{1}{2}e^{2\beta} + \frac{1}{2}\right]^{d-1}\left(e^{2\beta} - 1\right)}{\left[\frac{1}{2}e^{2\beta} + \frac{1}{2}\right]^d + \left[\frac{1}{2} + \frac{1}{2}e^{2\beta}\right]^d} = \frac{d\left(e^{2\beta} - 1\right)}{1 + e^{2\beta}} \end{aligned}$$

Setting

$$\frac{d\left(e^{2\beta} - 1\right)}{1 + e^{2\beta}} = 1$$

And solving for β yields

$$\begin{aligned} d\left(e^{2\beta} - 1\right) &= 1 + e^{2\beta} \\ e^{2\beta} &= \tfrac{d+1}{d-1} \\ \beta &= \tfrac{1}{2}\ln\tfrac{d+1}{d-1} \end{aligned}$$

To complete the argument, we need to show that q is a monotonic function of p. To see

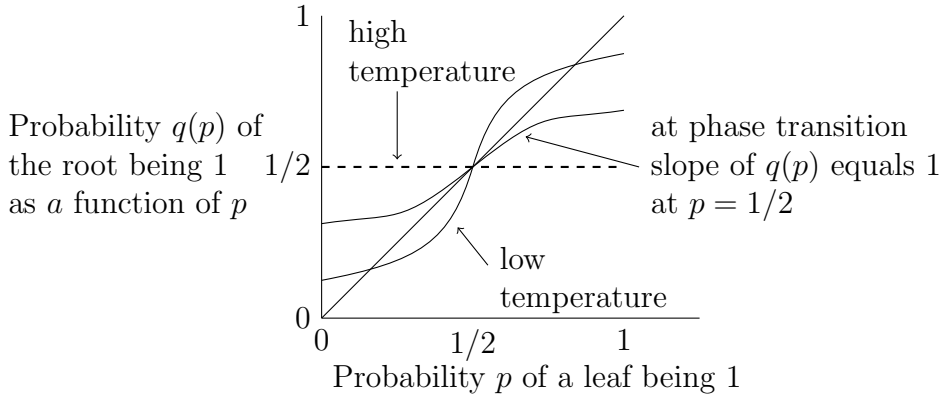


Figure 9.11: Shape of q as a function of p for the height one tree and three values of β corresponding to low temperature, the phase transition temperature, and high temperature.

this, write $q = \frac{1}{1+\frac{B}{A}}$. A is a monotonically increasing function of p and B is monotonically decreasing. From this it follows that q is monotonically increasing.

In the iteration going from p to q, we do not get the true marginal probabilities at each level since we ignored the effect of the portion of the tree above. However, when we get to the root, we do get the true marginal for the root. To get the true marginal's for the interior nodes we need to send messages down from the root.

Note: The joint probability distribution for the tree is of the form $e^{\beta \sum\limits_{(ij)\in E)} x_i x_j} = \prod\limits_{(i,j)\in E} e^{\beta x_i x_j}$.

Suppose x_1 has value 1 with probability p. Then define a function φ, called evidence, such that

$$\begin{aligned}\varphi(x_1) &= \begin{cases} p & \text{for } x_1 = 1 \\ 1-p & \text{for } x_1 = -1 \end{cases} \\ &= \left(p - \tfrac{1}{2}\right) x_1 + \tfrac{1}{2}\end{aligned}$$

and multiply the joint probability function by φ. Note, however, that the marginal probability of x_1 is not p. In fact, it may be further from p after multiplying the conditional probability function by the function φ.

9.22 Bibliographic Notes

A formal definition of Topic Models described in this chapter as well as the LDA model are from Blei, Ng and Jordan [BNJ03]; see also [Ble12]. Non-negative Matrix Factorization has been used in several contexts, for example [DS03]. Anchor terms were defined