1 Generative Models

Last time we introduced *latent semantic indexing*, a method for classifying documents using the singular value decomposition (SVD) of the term-document incidence matrix. Latent semantic indexing seems to work well, but can we explain why? In this section we provide some justification via generative models for documents. We consider three different models. The first two are due to Papadimitriou, Raghavan, Tamaki and Vempala (1998) and the third is due to Azar, Fiat, Karlin, McSherry and Saia (2001).

1.1 Model 1 [PRTV '98]

We assume that there are k "hidden" fundamental topics in the world that a document might be about. Each topic i has an associated set T_i of terms, and these term sets are disjoint, i.e. $T_i \cap T_j = \emptyset$ for $i \neq j$. A document on topic i is created by sampling w terms (with replacement) from T_i. We generate n documents, each by choosing a topic at random according to some distribution, then generating a document on this topic.

Then, assuming that the maximum probability of any term is a sufficiently small constant, and the number of documents is sufficiently large, one can show that latent semantic indexing with k dimensions will recover the sets T_i — and hence the topics — with high probability.

1.2 Model 2 [PRTV '98]

The assumption that the term sets are disjoint is very strong, but we can relax it. Let us now assume that the T_i are not necessarily disjoint, but that each topic has some core set of terms $T'_i \subseteq T_i$, the probability mass on T'_i is at least $1-\epsilon$, and the sets T'_i are disjoint. Thus,

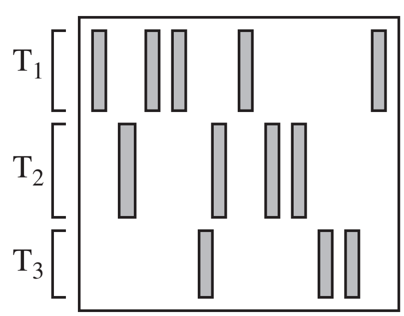

Figure 1: An array for documents and their terms using Model 1.

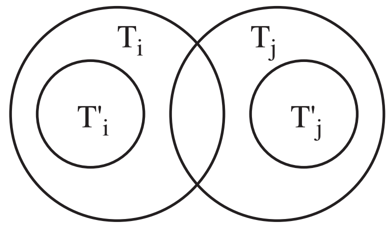

Figure 2: The limited overlap of terms between topics in Model 2.

our assumption is that the topics can have some overlap in the terms they use, but there are still a large number of "distinguishing terms." Again, we will assume that the maximum probability of any term is a sufficiently small constant.

As before, we generate a term-document matrix A and use the SVD to compute $A_k = U_k \Sigma_k V_k$, the best rank k approximation to A. The n documents correspond to $v_1, \ldots, v_n$, the n (unit length) columns of V_k. One can prove a similar theorem in this model.

Theorem 1.1 *With high probability (as the number of documents goes to infinity), each pair of vectors v_i, v_j on the same topic has $v_i \cdot v_j \geq 1 - O(\epsilon)$ and each pair on different topics has $v_i \cdot v_j \leq O(\epsilon)$.*

This theorem can be extended by allowing a "noise matrix" of small norm to be added to the term-document matrix A.

1.3 Model 3 [AFKMS '01]

Azar et al. generalize Model 1. Suppose that each entry of the term-document matrix is a random variable A_{ij}, giving the probability that term i is included in document j. We require that the matrix A have rank at most k (as it does in model 1).

We can generalize this model in several ways. First, we can add to the term-document matrix a "noise matrix" E with $||E||_2$ small. We can corrupt the term-document matrix by considering each entry independently and omitting it with probabiliy at most ϵ, to model missing data. Finally, it is not necessary that A actually have rank at most k; it is enough that it be "close" to rank k, in a certain precise sense.

Under this model, we obtain results similar to the previous ones, showing that we can recover an approximate representation of the underlying generative model with high probability.

2 Searching the Web

The SVD can also help us in designing search tools for the World Wide Web, using the Web's link topology as a guide; the goal is to estimate the "importance" of a page by the

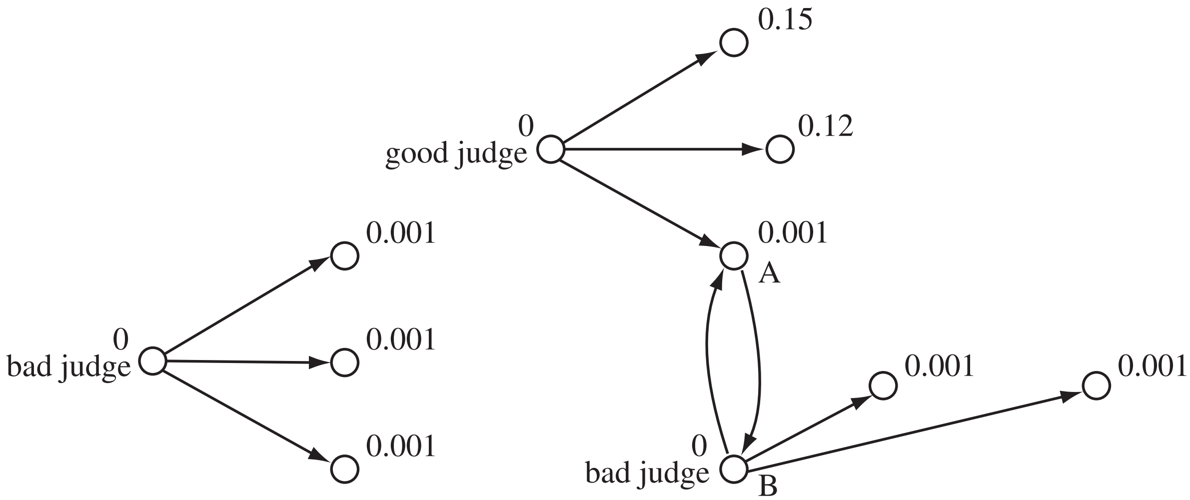

Figure 3: A subgraph of the web with an initial "importance" estimate.

link structure. The idea is to start with a subgraph of the overall web presumed to be rich in relevant pages and rank the pages by importance. For example, one could use the 200 pages most closely matching the text search and all things pointed to by those 200 pages as a subgraph.

As a first attempt one might simply rank the pages in order of the number of in-links. Unfortunately, this may end up being a simple popularity contest, with popular pages (Yahoo, Netscape, etc.) winning every time, regardless of relevance.

As an alternative we want to capture the notion that some pages are good judges of relevance, while others are relevant, but do not necessarily point to other good pages. For example, Ford and Toyota would both be relevant pages for a person wanting to buy a new car, but we would not expect these pages to point to each other. However, a page that pointed to both would be considered a good judge of relevance.

To this end we assign two types of scores to the nodes. For each node i, a_i is the "authority" score, representing the importance of the node, and h_i is the "hub" score, representing its ability to judge other pages. We're going to use authorities to "refine" hubs and vice versa. We'll also maintain the invariant $\sum a_i^2 = 1$ and $\sum h_i^2 = 1$, rescaling as necessary at each iteration. We'll start with an initial estimate h and a and update at each iteration as follows:

$$a_i \leftarrow \sum_{ji \in E} h_j$$

$$h_i \leftarrow \sum_{ij \in E} a_j$$

Figure 3 shows a subgraph with some initial authority scores, perhaps based on textual analysis of the documents. The node labeled "good judge" points to two nodes with a high initial authority scores, so in the first iteration it will receive a high hub score. In the second iteration, node A will get a higher authority score, since the good judge points to it. Even though node B was initially considered a bad judge, its hub score will improve in iteration 3, since it points to node A. Thus we see that good hubs and good authorities reinforce each other, and the results percolate around the graph.

This process makes intuitive sense, but does it even converge to a final estimate of hub and authority significance? If A is our directed adjacency matrix, $A_{ij} = 1 \iff ij \in E$, then we can express the updates in matrix-vector notation as follows:

$$\begin{aligned} h &\leftarrow Aa \\ a &\leftarrow A^T h. \end{aligned}$$

After $2k$ steps we have:

$$\begin{aligned} a_0 &\rightarrow (A^T A)^k a_0 \\ h_0 &\rightarrow (AA^T)^k h_0 \end{aligned}$$

Of course these values are all normalized to have norm 1.

Now express a_0 and h_0 in the space of eigenvectors of $A^T A$, where $\omega_1, \ldots, \omega_n$ are the eigenvectors corresponding to the eigenvalues $\lambda_1, \ldots, \lambda_n$. We index them so that $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_n$, and assume λ_1 is strictly greater than λ_2. Then

$$a_0 = \sum \alpha_i \omega_i,$$

so after $2k$ steps we have

$$a_0 \rightarrow \sum \alpha_i \lambda_i^k \omega_i \stackrel{k\to\infty}{\longrightarrow} \omega_1$$

The first term eventually dominates the sum, while normalization kills the other terms. The same reasoning gives

$$h \rightarrow \omega_1(AA^T).$$

In terms of the SVD, with $A = U\Sigma V$, we have $A^T A = V\Sigma^2 V^T$, and $AA^T = U\Sigma^2 U^T$. Our a is the first column of V, and h is the first column of U. Essentially, this process selects out the densest hub-authority subgraph from our original subgraph, as pictured in Figure 4.

We can view our set of hub and authority values as solving a maximization problem. Recall first the optimization view of the SVD: we seek to maximize $x^T Ay$ subject to the constraint $\|x\| = 1$ and $\|y\| = 1$. In our case $x = U_1$ and $y = V_1$, and so we are maximizing $\sum_{ij\in E} x_i y_j$. This can be viewed as a type of optimization problem in which the goal is a densest bipartite subgraph.

We can also view the hub and authority values in terms of a generative model proposed by Azar, et al. We presume that there exist "hidden" hub and authority values for all documents, and the link (i, j) is generated with probability $h_i a_j$. Notice that the matrix giving link probabilities in this model therefore has rank 1. Thus we can apply the results of Azar, et al. from Section 1.3 to show that computing U_1, V_1 for a randomly generated adjacency matrix recovers vectors "close" to the hidden values h and a.

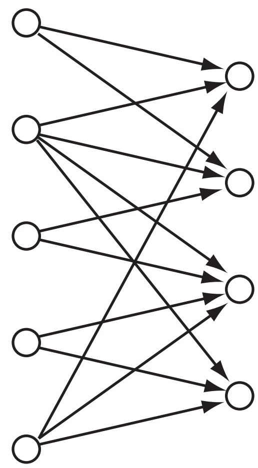

Figure 4: The hub and authority model for webpage classification.

The Google search engine is based on a related algorithm arising from work of Brin and Page, in which authorities directly "endorse" other authorities; the notion of hubs is not used. This too leads a weighting scheme that can be made precise using eigenvectors.

There many extensions to all these techniques, designed to improve overall search performance. But a discussion of this will have to wait for another time.