Spectral bi-clustering

Here's a summary of spectral bi-clustering.

The focal point was a paper by Inderjit Dhillon. There are two versions of that paper on the web. A short one was published in KDD. A long one is a UT technical report. I worked from the long one.

Dhillon illustrated the problem with term document matrices. You could also use market baskets (customers, items purchased) or genes and experiments.

Reducing words to word clusters could help you cluster documents. For example if the cluster corresponds to synonyms then you can judge similarity between two documents through synonyms in common which might be more powerful than mere words in common. Conversely, clusters of documents help you find the word clusters in the first place.

It sounds circular and we might anticipate an alternating algorithm. If so, we anticipated wrong. It gets solved via an SVD. Then again maybe the iterations inside the SVD computation look like alternations.

The bipartite graph is $G = (\mathcal{D}, \mathcal{W}, \mathcal{E})$, respectively documents, words and edges. Suppose that word i appears $t_{ij} \geq 0$ times in document j. Then the edge weight is $e_{ij} = t_{ij} \times \log(|\mathcal{D}|/|\mathcal{D}_i|)$ where $|\mathcal{D}|$ is the number of documents in the set and $|\mathcal{D}_i| = \sum_j 1_{t_{ij}>0}$ is the number of those documents containing word i. This is one of many variants on 'term frequency inverse document frequency' scoring.

Letting A be the matrix with $A_{ij} = e_{ij}$ the weight matrix for the bipartite graph is

$$M = \begin{pmatrix} 0 & A \\ A^{\mathsf{T}} & 0 \end{pmatrix}.$$

This is the matrix we usually call W but here W stands for words. The diagonal zero blocks arise because the graph is bipartite: no word-word or doc-doc edges appear.

If we cut the graph into sets $\mathcal{V}_1, \dots, \mathcal{V}_k$ then of course we can split each $\mathcal{V}_r$ into $\mathcal{D}_r \cup \mathcal{W}_r$.

The graph incidence matrix is I_G with a row per vertex and a column per edge. The column for the edge from word i to document j has the value $\sqrt{e_{ij}}$ in row i and $-\sqrt{e_{ij}}$ in row j. (These could be reversed without changing anything important.)

The graph Laplacian is $L = D - M$ as usual where D is diagonal with word elements $d_i = \sum_j e_{ij}$ and document elements $d_j = \sum_i e_{ij}$. We bundle up the word part in into D_1 and the document part into D_2. So $D = \mathrm{diag}(\mathrm{diag}(D_1), \mathrm{diag}(D_2))$.

Now

$$L = \begin{pmatrix} D_1 & -A \\ -A^{\mathsf{T}} & D_2 \end{pmatrix}.$$

$L = I_G I_G^{\mathsf{T}}$.

We looked at the cutting of the graph and ended up with the idea that we want the second smallest generalized eigenvector $z = z_2$ satisfying $Lz = \lambda Mz$.

The algorithm can be done in terms of the matrix A which is smaller than L. Long story short, the recipe is:

1. Put $A_n = D_1^{-1/2} A D_2^{-1/2}$.
2. Get the second singular vectors u_2 from the left and v_2 from the right. Use the second **largest** singular value of A_n which corresponds to the second **smallest** singular value of L. That is handy because a truncated SVD is easier to compute.
3. Put
$$z_2 = \begin{pmatrix} D_1^{-1/2} u_2 \\ D_2^{-1/2} v_2 \end{pmatrix}.$$
4. Run k-means with $k = 2$ on z_2.

When the k-means finishes up, there are two clusters of z_2 points and they split both the terms and documents into two clusters.

You can recursively bisect or do a k way split. For the k-way split, take $\ell = \lceil \log_2(k) \rceil$ of the left singular values into U and ℓ of the right hand ones into V. Then run k-means on
$$\begin{pmatrix} D_1^{-1/2} U \\ D_2^{-1/2} V \end{pmatrix}.$$

It seems that one could reasonably use a larger ℓ instead.

The article includes some worked examples. It is very successful at telling apart abstracts from different corpora. Those examples have nearly equally sized contributions from the corpora. There are also some examples splitting Reuters news stories into sports, health, technology and so on. Those results are less satisfactory. Part of the problem might be that the groups vary a lot in size.

k-means has trouble when some of the clusters are much smaller than others. It might prefer to split one of the big clusters. That's built into the criterion, not an artifact of the algorithm.

What is the importance?

The 2012 class raised an interesting question as to whether there is more going on here than just ordinary spectral clustering with a bipartite graph. For example the computational savings of reducing an $(m + n) \times (m + n)$ SVD to an $M \times n$ SVD might be minor given software that handles sparse SVDs.

It is intriguing that a clustering algorithm can be used to do biclustering. The heavy lifting seems to be in forming the adjacency particular matrix used. That already puts words and documents on the same footing allowing them to be clustered.

Another issue that was raised was how many singular vectors to use when seeking k groups. k perfectly separated groups will lead to k singular values equal to zero and a $k - 1$ dimensional space in which they're separated. Real data could be messier. Left unsaid is how to choose k or even a criterion for k.

Spectral learning

A related area is spectral learning. There the goal is to figure out the value of θ in a parameterized similarity measure $W(x_i, x_j; \theta)$ using a known clustering of some of the observations.

Some refs:

Bach and Jordan (2004) NIPS "learning spectral clustering".

Meila and Shi (2000)