Interview with Simon Funk: Why SVD approach?

Gregory Piatetsky-Shapiro

KDnuggets : News : 2007 : n08 : item6

Abstract

This is part of the KDnuggets interview with Simon Funk who became well-known in the Netflix prize competition for his application of Singular Value Decomposition (SVD) to that recommendation problem. The original article has been adapted by Dell Zhang for his NLP module at Birkbeck to make the notations consistent with the Stanford IR textbook (Chapter 18: Matrix decompositions and latent semantic indexing).

GPS: Q3) What led you to choose SVD approach over other methods. You have explained it well in your blog post , but can you briefly summarize the SVD approach?

Simon Funk: The best way to understand SVD is probably in reverse: to look at how one re-constructs a data matrix from the singular vectors. Consider just a single column-vector $\mathbf{u}_1$ and corresponding row-vector $\mathbf{v}_1^T$. If you multiply $\mathbf{u}_1$ by $\mathbf{v}_1^T$ you get a matrix $\mathbf{u}_1\mathbf{v}_1^T$ as tall as $\mathbf{u}_1$ and as wide as $\mathbf{v}_1^T$. Now if you have some target data matrix $\mathbf{C}$ of the same size (say the Netflix movie-by-user ratings matrix) you can ask: What choice of $\mathbf{u}_1$ and $\mathbf{v}_1^T$ would make that reconstructed matrix $\mathbf{C}_1 = \sigma_1\mathbf{u}_1\mathbf{v}_1^T$ as close to my target matrix $\mathbf{C}$ as possible? Here σ_1 is a weight to make the two vectors $\mathbf{u}_1$ and $\mathbf{v}_1^T$ both normlized (i.e., of length 1).

SVD is a mathematical trick for finding that optimal $\mathbf{u}_1$ and $\mathbf{v}_1^T$ pair. It's really just that simple, and the only additional tidbit is that once you've found that first $\mathbf{u}_1$, $\mathbf{v}_1^T$ pair, you can repeat the process with the leftovers (the difference between the original target matrix $\mathbf{C}$ and $\sigma_1\mathbf{u}_1\mathbf{v}_1^T$) to get a second pair of vectors, $\mathbf{u}_2$ and $\mathbf{v}_2^T$, and so on, such that the target matrix $\mathbf{C}$ is approximated by:

$$\mathbf{C} = \sigma_1\mathbf{u}_1\mathbf{v}_1^T + \sigma_2\mathbf{u}_2\mathbf{v}_2^T + \ldots,$$

with each successive term being progressively less significant due to the "biggest bite first" nature of the basic algorithm.

Looked at that way, it's clear that SVD is a particular way of modeling the data matrix $\mathbf{C}$. Assuming we trust the math to find the optimal parameters ($\mathbf{u}_i$, $\mathbf{v}_i^T$, ...), the question is how well does the model reflect the true process behind the data matrix? Here $\mathbf{u}_i$ is a vector over movies, and $\mathbf{v}_i^T$ over users, and the matrix $\mathbf{u}_i\mathbf{v}_i^T$ has in each cell the movie value from $\mathbf{u}_i$ times the user

value from $\mathbf{v}_i^T$. So in effect we are saying there is some aspect or attribute which is measured for each movie by $\mathbf{u}_i$, such as "how much Action does this movie have?", and for each user by $\mathbf{v}_j^T$, such as "how much does this user like Action movies?", and we just multiply those together to get our $\mathbf{u}_i\mathbf{v}_i^T$ estimate for how much each user would like each movie based only on the amount of Action. The model further says that the contributions from the different attributes are just added up linearly to make a final prediction. And that's it.

It's a very simple model, but it feels qualitatively correct to a first degree of approximation. And I had a very simple way to implement it using the incremental SVD method that I had come up with previously in my attempts to filter spam among other things, so I gave it a go. Honestly I only expected it to do about as well as Netflix's posted baseline — I was quite surprised when it did as well as it did!