CLASSY: A SINGULAR VALUE DECOMPOSITION BASED RECOMMENDATION ENGINE

LEV DUBINETS & TYLER YEATS

Academic advisors can recommend classes to students based on the student's past classes and how much they enjoyed said classes. In this paper, however, we propose to supplement such advisors with a system that automates the process using a common linear algebra technique called Singular Value Decomposition (SVD). The system we designed, named Classy, is able to, given a student, find other students who rated their classes similarly and recommend these other students' classes to the first student.

The advantage of this system over a more naïve direct matching approach is that it is much more comprehensive. Instead of just looking for students who have taken the same class, Classy actually knows what it means for classes to be similar to one another. Similar classes can be found by determining which groups of classes are commonly taken together, and this structure appears later in the decomposed matrices.

To compute recommendations, we first collect our information in a vector space where each dimension represents a class and each student is a vector in this space. A student's coordinate with respect to a given axis is his rating of the class that the axis represents. For example, if our space is based on classes $c_1, c_2, \dots, c_n$ and a student gave them ratings $r_1, r_2, \dots, r_n$, respectively, then this student's coordinates in this n-dimensional vector space are $[r_1, r_2, \dots, r_n]$. In our system we let students rate classes on a scale of 1-10 and a student's coordinate with respect to a class he has not taken is 0. To accentuate differences in ratings, the entry in the student-class matrix is the student's rating of a particular class minus 5.5. So, a rating of 10 corresponds to 4.5 in the matrix, and a rating of 1 to -4.5. Roughly, this means that favorable ratings become positive matrix entries, and unfavorable ratings become negative ones.

In such a vector space, two students who have taken similar classes and rated them similarly will be represented by vectors that have a relatively small angle between them. A measure of the similarity can be calculated by taking the cosine of the angle between the vectors in the desired vector space. This makes it easy to quantify the goal of our system: given a student, we find other students whose vectors have a small angle with respect to the original student's, and recommend classes based on these newfound students. However, as we add more students and classes to our space, the dimensionality of our vector space grows and computing recommendations becomes infeasible. We use the SVD to reduce the dimensionality of our system.

To give a more intuitive understanding of how the SVD functions, we will start with some background. Consider the matrix

$$A = \begin{pmatrix} 80 & 0 & 0 \\ 0 & 75 & 0 \\ 0 & 0 & 5 \end{pmatrix}$$

As this matrix is diagonal, its eigenvalues are $\lambda_1 = 80, \lambda_2 = 75$, and $\lambda_3 = 5$, and its eigenvectors are just the standard basis in R^3. Any arbitrary vector in R^3 can be expressed as a linear combination of the eigenvectors, then, such as

$$v = \begin{pmatrix} 5 \\ 7 \\ 3 \end{pmatrix} = 5i + 7j + 3k$$

When we multiply by A, we get

$$\begin{aligned} Av &= A(5i + 7j + 3k) \\ &= 5A \times i + 7A \times j + 3A \times k \\ &= 5(\lambda_1 \times i\,) + 7(\lambda_2 \times j) + 3(\lambda_3 \times k) \\ &= 400i + 525j + 15k \end{aligned}$$

It is clear that the smallest eigenvalue of A does not have much of an effect on the final vector compared to the two larger eigenvalues (Manning). In the case of larger matrices, the importance of the eigenvalues decreases with their relative size. This result is very important for understanding the truncated SVD.

To apply the SVD, we represent our data as a matrix in which students are rows, classes are columns, and values represent a student's rating of a class. For example, if our system contains n students and m classes, then the matrix we use to represent it is an $n \times m$ matrix A where each student is a row and each class is a column, and therefore student s_i's rating of class c_j is the value of the entry $A_{j,i}$. Since not all students take and rate all classes, the matrix A will usually be very sparse.

If A is the student-class matrix, then AA^T is a measure of similar students. Specifically, the $(i,j)^{th}$ entry of AA^T is the number of classes that students i and j have both taken, weighted by their ratings. If both users rated a class positively, the contribution to the sum of that class will be high, as it will if both users rated it negatively. If users rate classes differently, however, the entry will be more negative. The fact that similar ratings, whether good or bad, correspond to high similarities between users was one reason for the scaling of ratings from -4.5 to 4.5 instead of 1 to 10. Otherwise, if two users rated a class poorly, they would clearly be similar, but this would not be reflected in the matrix AA^T.

Similarly, the matrix A^TA maps the similarities between classes. The $(i,j)^{th}$ entry in this matrix corresponds to the number of students that have taken classes i and j, weighted by how they rated the class.

The SVD of matrix can then be computed as follows: set the columns of U equal to the eigenvectors of AA^T and the columns of V to the eigenvectors of A^TA. Note that since both AA^T and A^TA are symmetric, the eigenvectors are orthogonal. The SVD is then

$$A = U\Sigma V^T$$

where the values $\lambda_1 \dots \lambda_n$ are the eigenvalues of both AA^T and A^TA and $\Sigma_{ii} = \sqrt{\lambda_i}$

Given a matrix $A = U\Sigma V^T$, let $A_k = \sum_{i=1}^{k} u_i \sigma_i v_i^T$, also known as the truncated SVD. As shown in *Data-Driven Modeling*, this is the best rank-k approximation of A, meaning that it minimizes the Frobenius norm $\|A - A_k\|_F$, defined as the square root of the sum of the squares of the elements of A. The intuition behind this ties back to the fact that the truncated SVD can be represented as a sum of matrices scaled by the square roots of their eigenvalues, and the smaller eigenvalues will be smaller scaling factors and are therefore less important.

Essentially, computing the SVD and recombining it using k singular values allows us to reduce the dimension of our space to k while preserving as best as possible the original values in A. This result is also known as the Eckart-Young Theorem (Manning). Such a decomposition isolates the important parts of A and allows us to approximate our original space with a lower dimensional vector space.

Representing classes and users in a vector space allows for comparison between entities by cosine similarity. However, the main problem is that similar students that haven't taken overlapping classes would not be recognized by the system. Reducing the dimensionality by using the truncated SVD emphasizes the structure of the clusters in the larger vector space, bringing similar students closer together, even if they have not taken the same classes (Manning).

Note that choosing the value of k is non-trivial. Usually there is a sharp drop in the magnitude of the singular values, and that indicates what k should be. After we choose our k, we refer to A_k as A'.

$$
\begin{aligned}
[\,U \quad D \quad V\,] &= \mathrm{SVD}(A), \quad A' = UDV^{\mathrm{T}} \\
&= \begin{bmatrix} \boldsymbol{u}_{1,1}, \ldots, \boldsymbol{u}_{1,n} \\ \boldsymbol{u}_{2,1}, \ldots, \boldsymbol{u}_{2,n} \\ \vdots \\ \boldsymbol{u}_{n,1}, \ldots, \boldsymbol{u}_{n,n} \end{bmatrix}
\begin{bmatrix} \boldsymbol{\sigma}_{1,1}, 0, \ldots, 0 \\ 0, \boldsymbol{\sigma}_{2,2}, \ldots, 0 \\ \vdots \\ 0, 0, \ldots, \boldsymbol{\sigma}_{n,n} \end{bmatrix}
\begin{bmatrix} \boldsymbol{v}_{1,1}, \ldots, \boldsymbol{v}_{1,n} \\ \boldsymbol{v}_{2,1}, \ldots, \boldsymbol{v}_{2,n} \\ \vdots \\ \boldsymbol{v}_{n,1}, \ldots, \boldsymbol{v}_{n,n} \end{bmatrix}^{\mathrm{T}} \\
&= \sum_{i=1}^{n} \boldsymbol{\sigma}_i \boldsymbol{u}_i \boldsymbol{v}_i^{\mathrm{T}},
\end{aligned}
$$

FIGURE 1. *SVD Decomposition (Martínez)*

After the initial decomposition on our sparse matrix A is performed, we can use the resulting matrices U and V to determine coordinates of both students and classes. Since the reduced SVD gives us a k-dimensional space, we can represent each class a vector with k components. This vector, representing a class c, can be determined by taking the sum of the products of the ith column of U with the ith singular value, for i from 0 to k. Similarly, the vector representing a student s in this reduced space is the sum of the products of the jth column of V with the jth singular value, for j from 0 to k. So, to compute recommendations for a given student we simply find classes whose angles are small with respect to the student's. Note that it is the angle which is important, not necessarily the proximity of the points.

As an example of this technique, consider the following dataset, where the classes $C1 \dots C9$ correspond to MATH134, MATH135, MATH136, PHYS121, PHYS122, PHYS123, PHYS124, AMATH301, ENGL111, respectively:

$$\begin{array}{c|ccccccccc} & C1 & C2 & C3 & C4 & C5 & C6 & C7 & C8 & C9 \\ \hline Tyler & 9 & 8 & 9 & 1 & 2 & 3 & 2 & 0 & 0 \\ Lev & 9 & 8 & 9 & 2 & 2 & 2 & 1 & 7 & 1 \\ Nick & 9 & 8 & 8 & 2 & 1 & 3 & 1 & 8 & 1 \\ Jane & 8 & 9 & 8 & 1 & 3 & 1 & 2 & 9 & 1 \\ Joe & 6 & 7 & 6 & 6 & 7 & 6 & 4 & 8 & 1 \\ Lucy & 1 & 2 & 2 & 8 & 8 & 9 & 8 & 3 & 8 \end{array}$$

FIGURE 2. *Example dataset of students and classes.*

We compute the SVD of this matrix and then reduce it to rank 2. We then plot this data in 2-dimensions, using the first two columns of U, scaled by the first two singular values, as coordinates for classes, and the first two columns of V, scaled by the first two singular values, as coordinates for users:

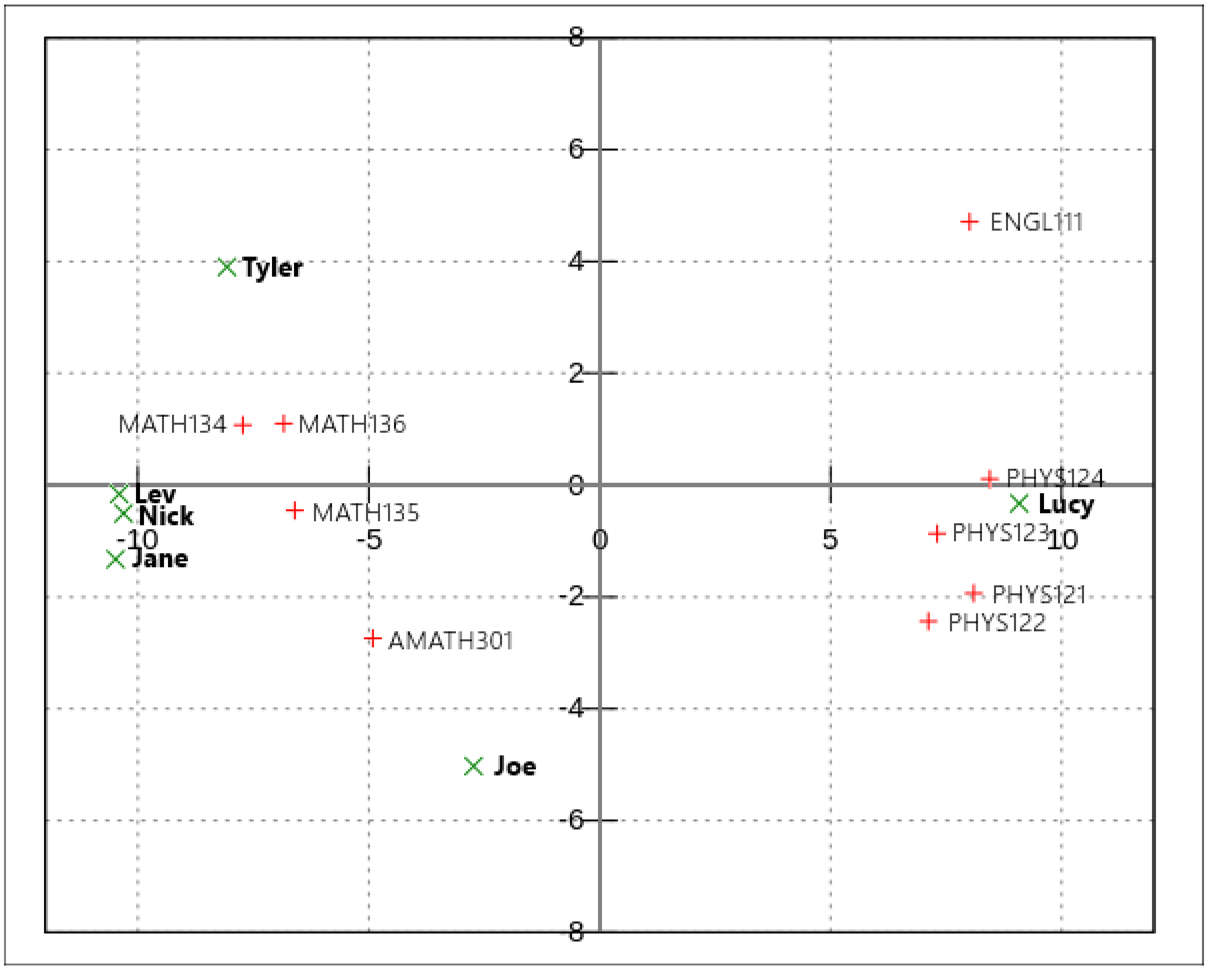


FIGURE 3. *Plot of the 2-dimensional approximation of the example system in* FIG. 2

This plot shows the kind of intelligence that the SVD allows us to extract. For example, we can see that the students Lev, Nick, and Jane have similar tastes, because they are clustered together. This is expected because they have very similar ratings for all the classes. Tyler is also fairly close to this cluster, however, as expected, because he has not taken AMATH301 and ENGL211, he is slightly distanced from Lev, Nick, and Jane. This is the information that ends up being used for recommendations.

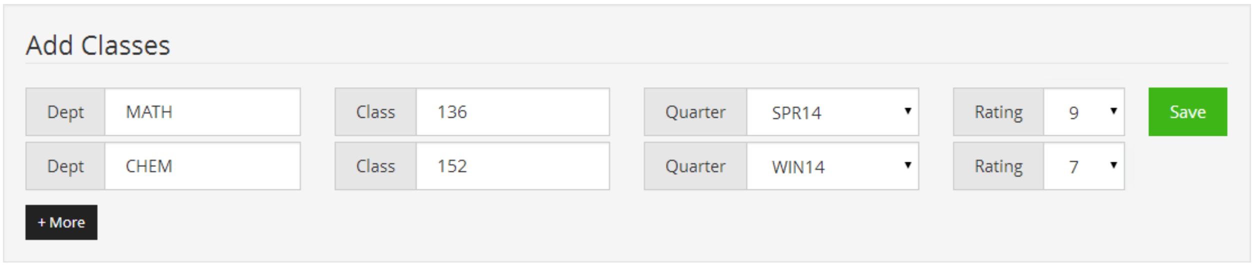
All that is left is to devise formulas to update our dataset with new ratings that we obtain from students and new classes that are added to the course catalog. This process is called folding in, and Berry *et al.* shows how to compute such projections:

For a new student vector: $s' = s^T U_k \Sigma_k^{-1}$ and a new class vector: $c' = cV_k \Sigma_k^{-1}$

When this occurs, the students and classes already in the data retain their original locations, so the new data has no effect on the old. However, continually folding in new students and classes may eventually degrade the accuracy and representation of students and classes in our space, as the data will not be properly updated to reflect some of the new connections that are made (Berry et al.). Therefore we must periodically rerun the SVD based on all of the data we have collected. For a large data set, this would be an expensive operation and an optimal schedule of re-computing and folding in would have to be decided upon. For our example, however, the data set is sufficiently small so that we can recalculate the SVD each time we receive new data.

To demonstrate the functionality of all these techniques, we have built a web application that provides an interface for students to add classes to their profile, stores all of these entries in a database, and gives recommendations back to the students by using the method described in this paper. The app is available at https://classy.dubinets.io .

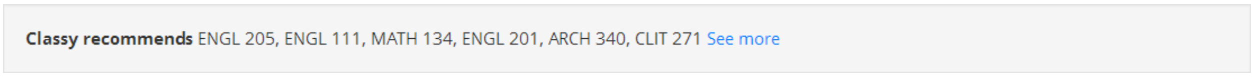
Upon signing up for the service, a user is greeted with a page explaining that he must add classes to his profile and rate them in order to receive recommendations. Ratings are assigned from 1-10, with 10 being the highest possible rating.


Add Classes

Dept	MATH	Class	136	Quarter	SPR14	Rating	9	Save
Dept	CHEM	Class	152	Quarter	WIN14	Rating	7	

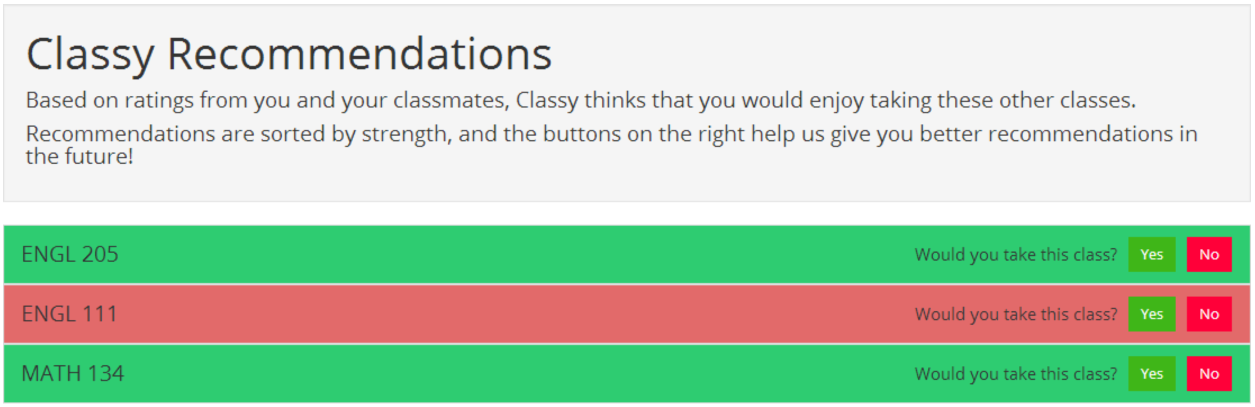
FIGURE 4. *A user adding and rating a few classes he has already taken.*

After adding and rating all or some of the classes he has taken, Classy will immediately recommend the top six classes. There is another page that lists all of the recommendations and contains buttons for a user to respond to the recommendations by indicating whether he would actually enroll in the recommended courses.



Classy recommends ENGL 205, ENGL 111, MATH 134, ENGL 201, ARCH 340, CLIT 271 See more

FIGURE 5. *Classy's top six recommendations for this user*



Classy Recommendations

Based on ratings from you and your classmates, Classy thinks that you would enjoy taking these other classes.

Recommendations are sorted by strength, and the buttons on the right help us give you better recommendations in the future!

ENGL 205
Would you take this class?
Yes
No
ENGL 111
Would you take this class?
Yes
No
MATH 134
Would you take this class?
Yes
No

FIGURE 6. *All of the recommendations are laid out like so, allowing the user to give feedback.*

Works Cited

Berry, Michael W., Susan T. Dumais, and Gavin W. O'Brien. "Using Linear Algebra for Intelligent Information Retrieval." SIAM Review 37 (1995): 573-95. CiteSeerX. Web. 8 May 2014. <http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.34.9579>.

Kutz, J. Nathan. Data-Driven Modeling & Scientific Computation. Oxford: Oxford University, 2013. Print.

Manning, Christopher D., Prabhakar Raghavan, and Hinrich Schütze. Introduction to Information Retrieval. Cambridge: Cambridge UP, 2008. Print.

Martínez, Jaime. Matrix decomposition to obtain singular values. Engineer Jau. WordPress, 4 May 2013. Web. 8 May 2014. <http://jauelingeniero.files.wordpress.com/2013/05/1-s2-0-s0167865505000140-si1.gif>.