Using 1 late day

Introduction

For this miniproject, we wanted to visualize movies and users, in clusters, based on an existing matrix Y, where we had m rows corresponding to m user_id values, and n columns for each of n movies. We first learned a Latent Factor Model in $U^\intercal$ and V, which had dimensions M x K and K x N respectively. We accomplished this by using Stochastic Gradient Descent after initializing U and V to random values between 0 and 1. See below for the gradient functions we optimized. Furthermore, after learning U and V, we visualized these by projecting them to 2 Dimensions and making a scatterplot of both movie and user data.

Stochastic Gradient Descent (Basic Formulation)

In implementing Stochastic Gradient descent, we computed the gradient of the following expression w.r.t $u_n, v_m \forall n \in U, m \in V$ for a given point y_{ij}.

$$l = \min_{U,V} \frac{\lambda}{2N}(||U||_{Fro}^2 + ||V||_{Fro}^2) + \frac{1}{2}(y_{ij} - u_i^\intercal v_j)^2$$

We compute the partial derivatives in this formulation as follows:

$$\frac{\partial l}{\partial_{u_k}} = \frac{\lambda}{N} u_k - \mathbb{1}^{k=i} v_j (y_{ij} - u_i^\intercal v_j)$$

$$\frac{\partial l}{\partial_{v_k}} = \frac{\lambda}{N} v_k - \mathbb{1}^{k=j} u_i (y_{ij} - u_i^\intercal v_j)$$

Stochastic Gradient Descent (Advanced Formulation)

With the advanced formulation, we have a slightly more complex loss function which, now include offset vectors a and b, defined as below:

$$l = \min_{U,V,a,b} \frac{\lambda}{2N}(||U||_{Fro}^2 + ||V||_{Fro}^2 + ||a||_{Fro}^2 + ||b||_{Fro}^2) + \frac{1}{2}(y_{ij} - \mu - (u_i^\intercal v_j + a_i + b_j))^2$$

We can compute the following partial derivatives:

$$\frac{\partial l}{\partial_{a_k}} = \frac{\lambda}{N} a_k - \mathbb{1}^{i=k}((y_{ij} - \mu) - (u_i^\intercal v_j + a_i + b_j))$$

$$\frac{\partial l}{\partial_{b_j}} = \frac{\lambda}{N} b_j - \mathbb{1}^{j=k}((y_{ij} - \mu) - (u_i^\intercal v_j + a_i + b_j))$$

$$\frac{\partial l}{\partial_{u_k}} = \frac{\lambda}{n} u_k - \mathbb{1}^{k=i} v_j((y_{ij} - \mu) - (u_i^\intercal v_j + a_i + b_j))$$

$$\frac{\partial l}{\partial_{v_k}} = \frac{\lambda}{n} v_k + \mathbb{1}^{k=j} u_i((y_{ij} - \mu) - (u_i^\intercal v_j + a_i + b_j))$$

Stochastic Gradient Descent, Process [1]

We ran SGD using Nesterov's accelerated gradient descent. Nesterov gradient descent calls for setting $U = (1-\gamma) * (U - d) + \gamma * U$, where U is a variable and $d = \nu * \nabla(U)$ is the gradient of U multiplied by a learning rate nu.

This can be reduced to:

$$U = U(1 - \gamma + \gamma) + d(\gamma - 1)\ U = U - d(1 - \gamma)$$

[1] See SGD.py

We therefore performed nesterov GD using the same update as rule as normal gradient descent with an added scale factor of $1 - \gamma$ being multiplied to the learning rate ν

A single epoch of SGD consists of updating our Nesterov parameters λ and γ (to clarify, this happens once per epoch) and performing the following process for each of our training points:

1. Randomly pick an unselected y_{ij}
2. Calculate the gradient ∇.

   We implemented two versions of this step. In the version used to generate the plots below (the version used in our submitted code), we only shift U_i, V_j, a_i, and b_j, so we only compute the gradient for those parameters. This version ran much more quickly and converged to an error around 35000.

   In the other version, we computed the gradient of the error function $l(i, j)$ with respect to every column in U and every column in V, and every entry in a and b. This version ran more slowly and converged to an error around 45000
3. Subtract $\eta\nabla$ from U V, a, and b
4. Repeat steps 1-3 for a set number of epochs.

Choice of Parameters

As per the set, we want to over-specify the rank of U and V, so we will pick the rank, $k = 20$. Based on some initial trials, we will pick $\eta = 0.001$. Note that with our tests, η decay did not help, so we decided to leave that out of the code. We picked our stopping criteria s.t. each value in U and V had to change by less than $\epsilon = .0001$.

SVD & Projection [2]

In our implementation, we did not explicity calculate SVD (i.e. using PCA and gradient descent), though note that this was not required for full credit. Instead, we used:

```
A, sigma, B = np.linalg.svd(self.V)
```

However, once we found the A matrix in the SVD decomposition of $V = A\Sigma B^\intercal$,we define $\bar{A} = A^\intercal_{1:2}$. We then have the projections of U and V:

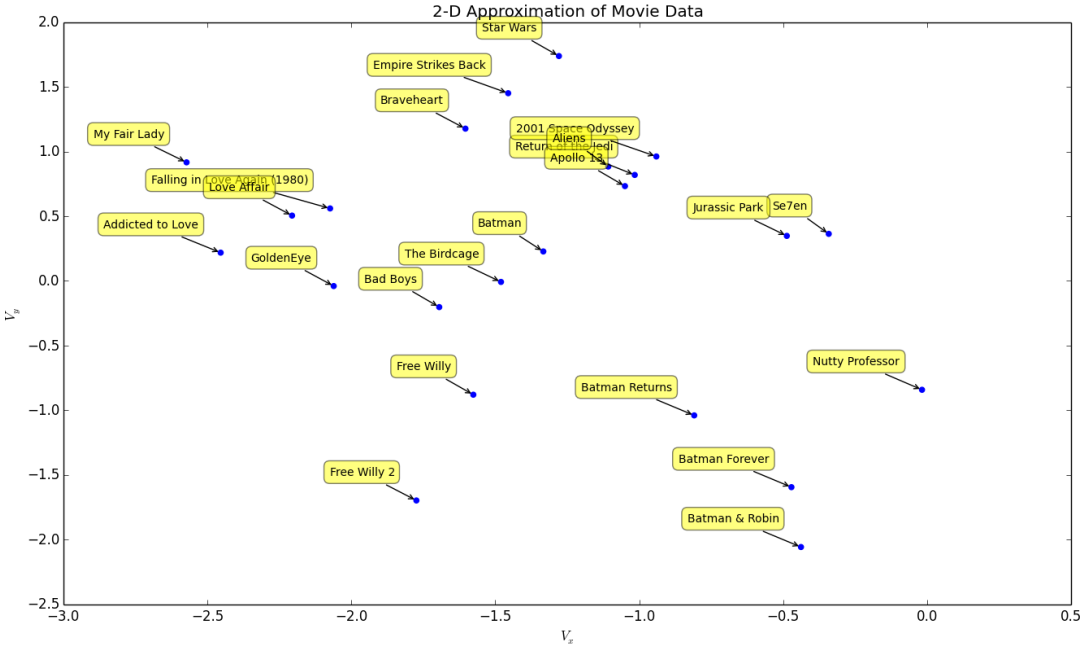
$$\bar{V} = \bar{A}V, \bar{U} = \bar{A}U$$

Visualizations

Note that these visualizations were based on SGD as per the Advanced Formulation, though our results were not completely different with the Basic Formulation.

We first graph specific movies, some of them being similar to the ones that Professor Yue chose to graph in his demonstration.

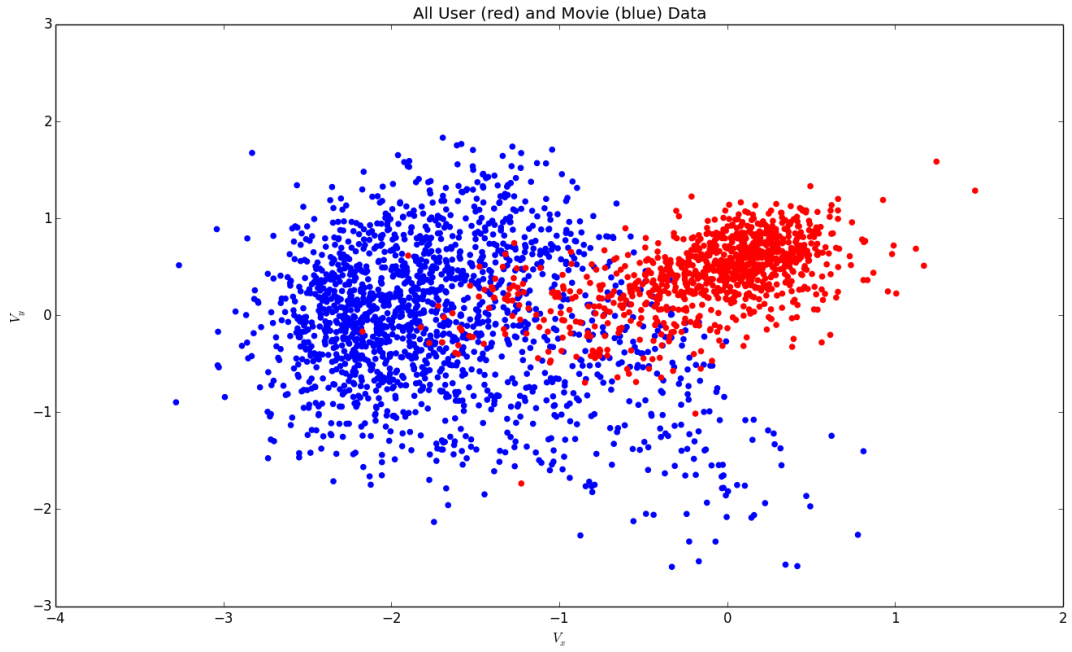
[2] See Visualizer.py



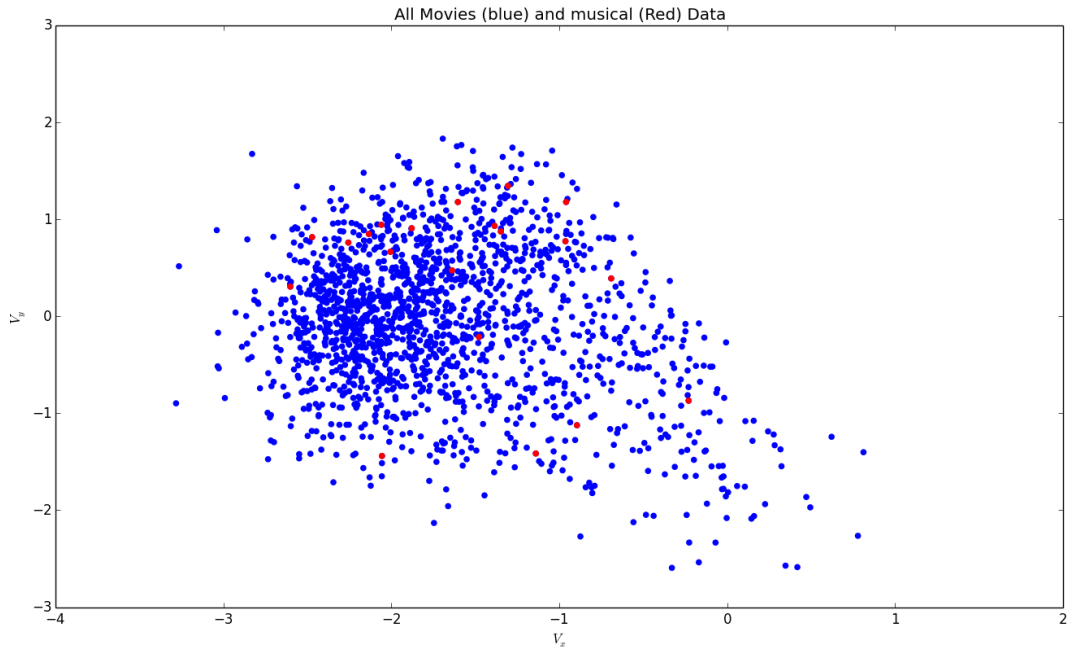
There are some interesting trends in the above figure. Of course, since this only captures 2 dimensions of a dataset which was significantly higher-dimensional, we cannot capture all the variation in the movies. We can observe, however, that there are clusters of similar movies in this figure. Three of the Batman films seem to cluster towards the bottom of the figure, with V_x values varying between -1.0 and -0.25 and V_y values varying between -2.5 and -1.0. Furthermore, there is a group of science-oriented or science fiction movies towards the top of the visual, including movies like Aliens, Apollo 13, Space Odyssey. This cluster makes sense because these movies all seem to revolve around similar thematic elements of science, space, and generally futuristic tendencies. Of course, the Star Wars films are also close together, all having V_y positive and V_x greater than -1.75. The fact that "Free Willy" and "Free Willy 2" are relatively close to each other shows sensible correlation. To conclude, we see that most of the romantic and love-themed movies cluster towards lower V_x values, including Love Affair, Falling in Love Again, and Adicted to Love.

If we had to make some predictions on the axes of these graphs and what they represent, we would guess that moving left on the x axis correlated to perhaps more emotional movies, or movies that appeal to females. Moving higher on the y axis seems to correlate to more science-based films. Although the action films are generally scaterred, action films seem to be heavier towards larger values of V_x.

As required, we also graph all the movies and users in one plot. Although it's hard to interpret the meaning of the user data (the red dots below), we can generally tell from this plot that both datasets are mean-centered close to 0 in the y direction (but maybe the movie dataset is negative-centric in the x direction).



To gain more insight, we picked out specifically the musicals from the set of movies and highlighted them below in red. Since these movies are not all *exclusively* musicals, it's unreasonable to expect strong clustering. However, we see that many of the musicals are indeed clustered between $-3 < V_x - 1$ and $0 < V_y < 1.5$. We chose to graph musicals somewhat arbitrarily, but it seemed like the most unique genre with the least overlap with other genres.



Furthermore, we calculated the average v value for musicals: (-1.5548562, 0.3654483), which makes aligns with our qualitiative observations.