# 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 each had dimension $M$ x $K$ and $K$ x $N$ respectively. We accomplished this by ysing 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 have the following process for stochastic gradient descent:

1. Randomly pick a $y_{ij}$
2. Compute 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 $b$ (for the advanced formulation, i.e. see above).
3. Subtract $\eta\nabla$ from $U$ $V$, $a$, and $b$
4. Repeat steps 1-3 until $\eta\nabla < \epsilon$.

[1] See SGD.py

## 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, $\eta$ 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

[2] See Visualizer.py