Assignment 5:  
Collaborative Filtering

Netflix Challenge Problem

## Objective

In this project you will implement a gradient descent method to compute the top singular vectors of a sparsely populated matrix. You will test the method against a subsampled known matrix whose true SVD you know. Using various Cross Validation techniques, you will calculate the error in your prediction.

## Data Description

In this project you will subsample the matrix corresponding to the greyscale pixel values of [this image](https://userscontent2.emaze.com/images/8affdf21-ab35-4db3-bcb7-79519c3a2207/e46ad53a68c5c8289b076159e534c635.jpg) for s=0.1%, 1%, 5%, 10% random subsampling. This will be your dataset. For a given you will set .

## Methodology and Deliverables

Your goal is to predict via sparse-SVD using information obtained from . You will calculate the average bias and error of your predictions using 10-fold cross validation at a given level of subsampling s.

### True SVD

Calculate the true top-10 SVD components of the true image and store them somewhere.

### Random Projection SVD algorithm

First, let’s use sklearn.decomposition.TruncatedSVD(algorithm=’randomized’) as in Assignment 3. Use rank 10 approximation for the matrix.

* Calculate the mean bias and standard deviation of your predictions as a function of the sparsity s. Plot examples of the predicted images at various level of s. Plot the cross validation prediction error as a function of sparsity s.
* Compare the top K SVD eigenvalues/vectors of with the top K eigenvalues/vectors from of from the previous section. A useful comparison is to normalize each set of eigenvectors to unit norm (which already is the case for the TruncatedSVD algorithm) and then compute the [cosine distance](https://en.wikipedia.org/wiki/Cosine_similarity) KxK matrix between the two sets, such that its k1, k2 matrix entry is the dot product between the normalized k1-th svd\_solve eigenvector and the normalized k2-th eigenvector of the true SVD. Compute the cosine distance matrices for both the left (U) and the right(V) eigenvectors. What is the average cosine distance as a function of the sparsity s?

### Sparse-SVD Gradient Descent algorithm

Let’s use another gradient descent-based algorithm which was described in [Simon Funk’s 2006 blogpost on the Netflix Prize](http://sifter.org/~simon/journal/20061211.html). In particular, you will implement the method described in the blog article and produce a function svd\_solve(X, K, lrate, lambda) which takes a [sparse matrix](https://docs.scipy.org/doc/scipy/reference/sparse.html) and an input rank K, and outputs matrices U (NxK) and V (MxK) such that UV^T is the best rank-K approximation of X. The learning rate parameter (lrate) and regularization parameter (lambda) are also inputs. Simon Funk has a discussion about which parameters worked for him. Make sure you understand how to set the initial conditions for the algorithm as discussed in the blog post, as well as how to incorporate mini-batching in order to speed up the convergence. Once you implement your algorithm:

* For each of the pairs in the previous section, apply the svd\_solve(X\_train, 10, lambda) and obtain the top K SVD components. Before you feed the image into the svd\_solve method, make sure that you normalize X\_ij so that mean(X\_ij) = 0 and std(X\_ij) = 1 along the longer of the two dimensions. That way the total variance of the matrix is min(N, M) and finding the optimal learning rate will be be as in Simon Funk’s blog post.
* Compare the top K SVD eigenvalues/vectors of with the top K eigenvalues/vectors from of from the first section. A useful comparison is to normalize each set of eigenvectors to unit norm and then compute the [cosine distance](https://en.wikipedia.org/wiki/Cosine_similarity) KxK matrix between the two sets, such that its k1, k2 matrix entry is the dot product between the normalized k1-th svd\_solve eigenvector and the normalized k2-th eigenvector of the true SVD. Compute the cosine distance matrices for both the left (U) and the right(V) eigenvectors. What is the average cosine distance as a function of the sparsity s?
* Choose K\_max to be the threshold at which the cosine distance between the svd\_solve and the true eigenvectors is less than 10%. How does K\_max vary with the subsampling rate s?
* Show the reconstructed image for each K\_max for the various subsampling rates s.

Which of the two models is better and why? When would you use the random projection TruncatedSVD method and when would you use the Gradient descent method? Explain