CS 181 – PRACTICAL 1

CASEY GRUN, SAM KIM, RHED SHI

1. Warmup

We used the K-Means algorithm to cluster the images of the CIFAR-10 dataset, which consists of 60000 total 32x32 color images. We only clustered a single batch, consisting of 10000 images. The algorithm was run for $k = 5, 10, 15$.
The following sets of images show the mean images of the k clusters for $k = 5, 10, 15$ as well as the 25 images that are closest to each of the respective means for $k = 5, 10$.

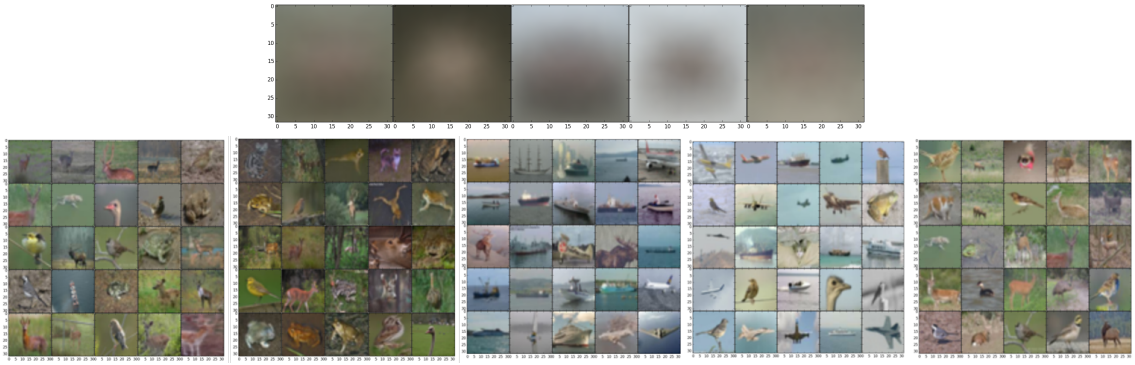


Figure 1. (Top) Cluster means and (bottom) 25 representative images for each of the cluster means for $k = 5$.

Although features are not discernible from the mean images, there are still trends of lighter colors versus darker colors, which is consistent with the representative images. For example, in $k = 5$, the 2nd cluster mean is mostly dark around the edges whereas the 4th cluster mean is mostly light around the edges, and the representative images share these features. Additionally, we see that as we increase k, the distinction between mean images increase as well. For $k = 5$, the means are mostly greyish brown images, whereas for $k = 15$, several of the mean images have tints of blue.

We use the objective function:

$$J(\{r_n\}_{n=1}^N, \{\mu_k\}_{k=1}^K) = \sum_{n=1}^{N} \sum_{k=1}^{K} r_{nk} ||x_n - \mu_k||_2^2$$

FIGURE 2. (Top) Cluster means and (bottom) 25 representative images for each of the cluster means for $k = 10$.

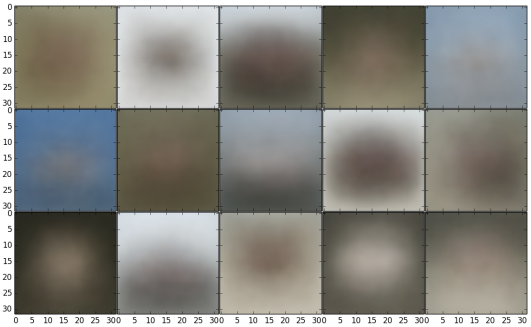
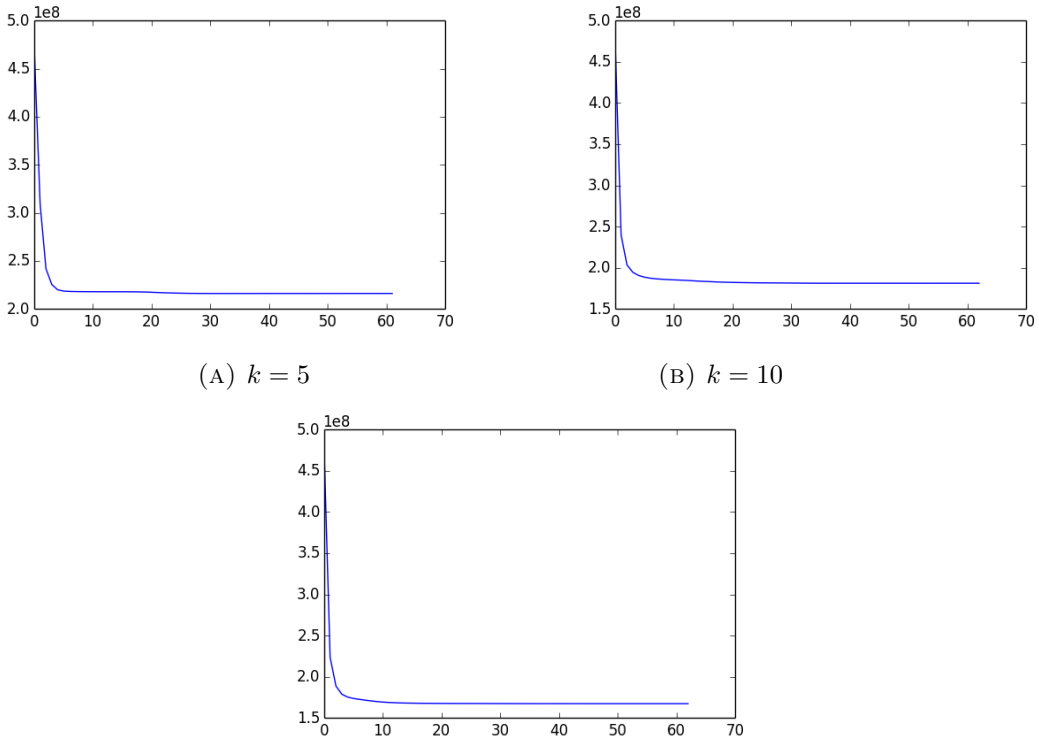


FIGURE 3. Cluster means for $k = 15$

where r_n is the responsibilities vector of the nth data point, μ_k is the mean of the kth cluster, and x_n is the nth data point. Plots of the objective function as a function of iteration for $k = 5, 10, 15$ are shown below.

We can see that for all 3 ks, the objective function never increases over the iteration, so the algorithm is constantly finding a better solution and converging to a (local) minimum. In addition, comparing the objective functions, $k = 5$ reaches approximately $J = 2.2 \times 10^8$,



(A) $k = 5$ (B) $k = 10$

(C) $k = 15$

FIGURE 4. Objective function as a function of iteration for K-Means

$k = 10$ converges to $J = 1.8 \times 10^8$, and $k = 15$ converges to $J = 1.7 \times 10^8$. As suspected, allowing more clusters allows the images to be closer to their respective cluster means.

We also implemented the K-Means++ algorithm. Shown below are mean images of the k clusters fo $k = 5, 10$. These are not qualitatively different from those of the K-Means algorithm. The objective function for $k = 5$ reaches $J = 2.15 \times 10^8$, and for $k = 10$ reaches $J = 1.8 \times 10^8$.

2. Approaches considered

K-Means, K-nearest neighbor, PCA, SVD, PMF

3. K-means and K-nearest neighbor

We first tried a preliminary approach of K-means and K-nearest neighbor. We used $K = 20$ to cluster our set of users and then predicted the book ratings based on the cluster mean. For both approaches, we standardized the user book ratings matrix by subtracting

FIGURE 5. Cluster means for (top) $k = 5$ and (bottom) $k = 10$

the global mean and dividing by the global variance. The predictions made with the K-means approach were almost entirely filled with 4 ratings with 3.999 and 4.0001 appearing often as well. Diagnosing our approach, we realized that the sparsity of the user book ratings matrix had essentially substituted the global mean for all unfilled spots as we clustered the users together. This meant that this approach was heavily biased towards the global mean and the results show that. The K-means approach yielded a RMSE of 0.86903, which was just slightly above the global mean.

We also attempted to use K-nearest neighbor approach where we calculated how close certain users are to each other by the number of books they have read in common and the average difference in ratings for those books. Calculating the hamming distance between users took nearly 4 hours with multiple runs due to small bugs. By that time, we decided it was more effective to pursue the probabilitistic matrix factorization approach, which, according to the Netflix competition paper, would capture unkown latent factors and be more predictive if implemented well.

4. Principal Component Analysis

Before moving onto matrix factorization techniques, we also tried the PCA approach. We found the covariance matrix for the user book ratings but realized that imputation was a huge handicap because the user book ratings matrix was so sparse. Thus, PCA was not the best approach for this data set.

5. Matrix Factorization

5.1. **Algorithm.** Given a set of ratings described by the matrix $\mathbf{R}$, matrix factorization attempts to decompose this into two matrices: $\mathbf{P}$ representing the users and $\mathbf{Q}$ representing the books. Each entry $\vec{p_i}$ in $\mathbf{P}$ represents a vector of K latent features of user i, while $\vec{q_j}$ represents the k latent factors of the book. If we can produce the matrices $\mathbf{P}$ and $\mathbf{Q}$, we can predict any entry $r_{i,j}$ in $\mathbf{R}$ by taking inner products of the vectors $\vec{p_i}$ and $\vec{q_j}$:

$$\hat{r}_{i,j} = \vec{p_i} \cdot \vec{q_j}^T \approx r_{i,j}$$

In the algorithm outlined by Koren et al., we iteratively train $\mathbf{P}$ and $\mathbf{Q}$ based only on the available data [**?**].

In addition, biases are introduced to account for ratings that are dependent on either the user or the book alone [**?**, **?**]. Let μ be the global mean, $\vec{b}$ represent the biases for each users and $\vec{c}$ represent the biases for each book. The predicted rating is thus

$$\hat{r}_{i,j} = \mu + b_i + c_j + \vec{p_i} \cdot \vec{q_j}^T \approx r_{i,j}$$

To learn the values of $\mathbf{P}$ and $\mathbf{Q}$, wel minimize an error function:

$$E = \sum_{i,j,r_{i,j} \in T} (r_{i,j} - \hat{r}_{i,j})^2 + \frac{\beta}{2} \left(||p_i||^2 + ||q_i||^2 + b_i^2 + b_j^2 \right)$$

The first term of this error function simply describes the squared reconstruction error. The second term enforces regularization—penalizing rows of $\mathbf{P}$ and $\mathbf{Q}$ that have high magnitude.

We use $\nabla e_{i,j}$, the gradient , to minimize $e_{i,j}$ (and hence E) by stochastic gradient descent.

$$\begin{aligned}
p_{i,k} &\leftarrow p_{i,k} + \alpha \left(2 \left(r_{i,j} - \hat{r}_{i,j} \right) q_{j,k} - \beta p_{i,k} \right) \\
q_{j,k} &\leftarrow p_{i,k} + \alpha \left(2 \left(r_{i,j} - \hat{r}_{i,j} \right) p_{i,k} - \beta q_{j,k} \right) \\
b_i &\leftarrow p_{i,k} + \alpha \left(2 \left(r_{i,j} - \hat{r}_{i,j} \right) + \beta b_i \right) \\
c_j &\leftarrow p_{i,k} + \alpha \left(2 \left(r_{i,j} - \hat{r}_{i,j} \right) + \beta c_j \right)
\end{aligned}$$

We update $\vec{p_i}$, $\vec{q_j}$, bi, and c_j for each pair $(i, j, r_{i,j}) \in T$. This process is repeated for some number of steps, or *epochs*, until either some criterion for epochs is met (e.g. the change in the total error E is less than some ϵ, or the maximum number of epochs is exceeded).

5.2. **Optimization.** The main issue we had with the PMF approach was that there were many of meta-parameters to tune: the learning rate α, the regularization parameter β, the number of latent factors K, and the criteria for determining convergence. We estimated appropriate starting values for $K = 5$ and $\beta = 0.02$ based on literature on the Netflix prize [**?**, **?**], and we experimented to determine that $\alpha = 0.001$ generally produced convergent solutions. We tried the following reductionist methods for validating our approach, and for further tuning these parameters:

(1) Checking the gradient. We verified that our gradient function was correct by finite differences.
(2) Recapitulating the user mean results. We tried setting $K = 1$, $\beta = 0$ and fixing $q_{j,k} = 1$. This gave us an RMSE of 0.77857—slightly better than the user mean results.
(3) Reconstruction of fake data. We generated fake values for $\mathbf{P}$ and $\mathbf{Q}$, and multiplied them together to get a fake $\mathbf{R}$. We then trained the model on T containing some fraction of the $r_{i,j}$'s.
(4) Cross-validation. We withheld a subset (between 1–10%) of our training data, trained the model on the remainder, computed the difference between the predicted

ratings and the withheld ratings. We used this method to attempt to set our parameters

5.3. **Results and Conclusion.** We achieved our best RMSE value of 0.77857 with the probabilistic matrix factorization approach that the winners of the Netflix competition used. This was with latent factor $K = 1$, $\alpha = 0.001$, and $\beta = 0.0$, with no bias vectors. As noted, this is slightly better than the user mean RMSE value.

Increasing the number of latent factors seemed to make the results worse. Likewise, adding in the bias vectors or increasing β also seemed to be detrimental to the results. This observation might be explained by the overfitting of the sparse data set. Having too many factors to learn from the training set causes the machine learning algorithm to overfit the training data and perform less optimally on the test set.