E-181 : Practical 1

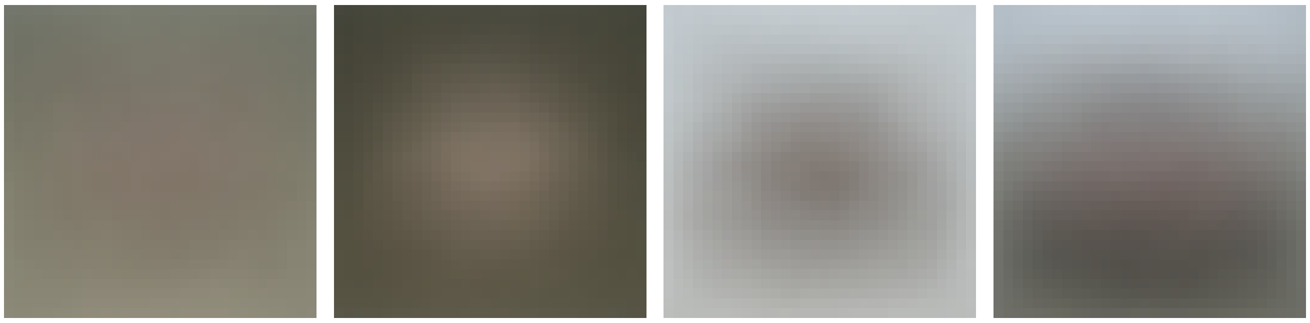
Due on Friday, February 14, 2014

Ben Athiwaratkun

Contents

Section 1: K means on CIFAR-10

I implemented K Means in Python and run the algorithm on CIFAR-10. Initially, for the number of clusters K, distinct K points are picked from the data set to use as initial centers of clusters. For each iteration, I observe that the distortions always decrease. Running the algorithms multiple times seem to give consistent looks for the centers of clusters. Below are the centers of clusters for $K = 4$.



(a) Center of Cluster 1 (b) Center of Cluster 2 (c) Center of Cluster 3 (d) Center of Cluster 4

Figure 1: Pictures of Centers of Clusters for $K = 4$.

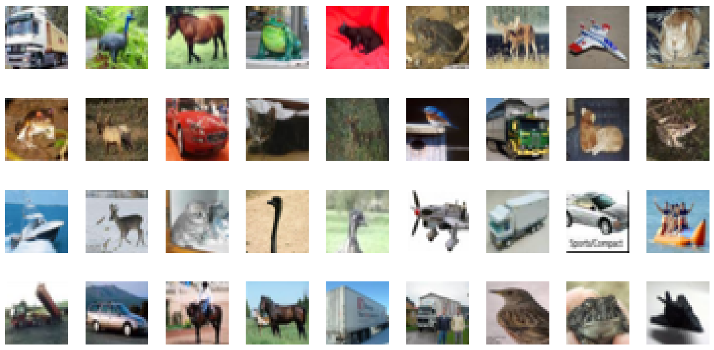


Figure 2: Representative Pictures for Each Cluster by Row

Cluster 2 represents pictures with the center lighter than the surrounding while cluster 3 represents the pictures with centers that are darker than the surrounding. Cluster 4 represents pictures with a upper half horizon that is somewhat light compared to the bottom half.

Section 2: Kaggle Book Ratings

Overview

In the attempt to predict the book ratings, I implemented several models and compare the results using the cross-validation set. (as occasionally on the Kaggle Leaderboard) Below is a quick summary of the methods used.

1. Probabilistic Matrix Factorization (PMF) linear model. The model probabilistically factorizes the rating matrix into latent user matrix and latent item matrix.
2. Probabilistic Matrix Factorization (PMF) bounded model. This model is similar to the first model but uses a logistic function, scaled from 1 to 5, to bound the prediction.
3. PMF model with a similarity constraint. The model is similar to the second model but adds a latent item similarity matrix which contributes to the latent user matrix in the factorization.
4. Multi-class one-vs-all classification with PMF model. This model is similar to the second PMF model but factorizes each class of rating into its own set of latent user and item matrices. I compare the results of the discrete prediction where the model predicts the class with the highest probability and the expectation value of rating based on the normalized probability.
5. Weighted User-Item Means. After observing that the best result is highly correlated with being close to the user mean, I try to use a model that is directed based on the user mean. This model predicts the rating based on the linear combination of the user mean and the item mean. The optimal weight is obtained by consider cross-validation set root mean square error. This method seems to do best on the Kaggle leaderboard compared to models mentioned above.

The implementation is done in `C++`The training set used in all models is the first 80% of the data in `ratings-train.csv`. The cross-validation set is the rest (20%) of the data.
The sections below describe the details and results of each model.

Section 2: Kaggle Book Ratings continued on next page...

Model 1: PMF Model

Setting up

This model is described in *Probabilistic Matrix Factorization* by S. Salakhutdinov and A. Mnih and *Matrix Factorization Techniques for Recommender Systems* by Y. Koren, R. Bell, and C. Volinsky.
Let R_{ij} be the rating from user i of item j and I_{ij} be the indicator (1 or 0) of whether a user i has rated the item j in the training set. Let the vector $\vec{P_i}$ be a latent vector for user i and $\vec{Q_j}$ be a latent vector for item j. The objective function that we desired to minimize is the sum of square error, given by

$$E = \sum_{i,j} I_{i,j} \cdot (R_{ij} - \vec{P_i} \cdot \vec{Q_j})^2 + \frac{\lambda_P}{2} \sum_i \left\| \vec{P_i} \right\|_2 + \frac{\lambda_Q}{2} \sum_i \left\| \vec{Q_i} \right\|_2$$

To minimize the cost function, we decrement each P_i and Q_j proportionally to its gradient. The gradients are given by

$$\vec{\nabla}_{P_i} E = -2 \sum_j I_{ij} \cdot (R_{ij} - \vec{P_i} \cdot \vec{Q_j})\ \vec{Q_j} + \lambda_P \vec{P_i}$$

and

$$\vec{\nabla}_{Q_j} E = -2 \sum_i I_{ij} \cdot (R_{ij} - \vec{P_i} \cdot \vec{Q_j})\ \vec{P_j} + \lambda_Q \vec{Q_j}$$

Implementation

The gradients are validated against finite differences without regularization term. The gradients computed from derivatives are correct close to the ones from finite differences for small step. For larger step, I observe the second order term having an effect, which is expected. For the number of latent features 1 and $Q_j = 1$, the model prediction is expected to be close to the prediction from user mean. This is because

$$0 = -2 \sum_j I_{ij} \cdot (R_{ij} - p_i \cdot q_j)\ q_j$$

yields the optimal p_i given by

$$p_i = \frac{\sum_j I_{ij} \cdot R_{ij}}{\sum_j I_{ij}}.$$

which is the user mean. With $3,000$ iterations, the root mean square error of the difference between the user mean and PMF's prediction is 0.0668. For $10,000$ iteration, the root mean square error is 0.021714. The cross validation RMSE is 0.792, which is close to the cross validation RMSE when predicted by the user mean method.

Results

Table 1. shows the training RMSE as well as the test/cross-validation set RMSE. Gradient descent is repeated until the training RMSE, calculated every 20 iterations, decrease less than 1%, which is equivalent to amortized 0.05% for consecutive iterations. For each set, the step size γ is controlled to be not too large such that it the gradient descent converges.

Discussion

I observe that the gradient descent can diverge or does not always converge for high step size γ. Reducing the step size translates to longer running time, which is not desirable. In the next model, I use a scaled logistic function to bound the prediction which results in lower RMSE as well as allowing higher γ and faster gradient descent. Before doing that, I experimented with a stochastic method of gradient descent with

$$\vec{\nabla}_{P_i} E = -2 \left(\sum_{j'} I_{ij'} \right) \cdot (R_{ij} - \vec{P_i} \cdot \vec{Q_j})\ \vec{Q_j} + \lambda_P \vec{P_i}$$

Table 1: Root Mean Square Errors

No. Features	Train RMSE	C.V. RMSE	Number of Iterations	Step Size (γ)
1	0.606	0.930	1120	0.0001
2	0.571324	0.948278	1080	0.0001
4	0.521921	0.946928	1120	0.0001
8	0.438776	0.947267	1280	1

and similar gradient for $\vec{\nabla}_{Q_j}$. It does not seem to help with running time and fail easily fail to converge for a moderate γ.

Section 2: Kaggle Book Ratings continued on next page. . .

Model 2: Bounded PMF Model

Setting up

The linear PMF model allows the prediction to be out of the desiable range (1 to 5). I use the logistic function

$$g(x) = 1 + 4\frac{e^{-x}}{1+e^{-x}}$$

to constrain the prediction to be from 1 to 5. The objective function becomes

$$E = \sum_{i,j} I_{i,j} \cdot (R_{ij} - g(\vec{P}_i \cdot \vec{Q}_j))^2 + \frac{\lambda_P}{2} \sum_i \left\| \vec{P}_i \right\|_2 + \frac{\lambda_Q}{2} \sum_i \left\| \vec{Q}_i \right\|_2 .$$

The gradients are then given by

$$\vec{\nabla}_{P_i} E = -2 \sum_j I_{ij}(R_{ij} - g(\vec{P}_i \cdot \vec{Q}_j))\ g'(\vec{P}_i \cdot \vec{Q}_j)\ \vec{Q}_j + \lambda_P \vec{P}_i$$

and

$$\vec{\nabla}_{Q_j} E = -2 \sum_i I_{ij}(R_{ij} - g(\vec{P}_i \cdot \vec{Q}_j))\ g'(\vec{P}_i \cdot \vec{Q}_j)\ \vec{P}_j + \lambda_Q \vec{Q}_j$$

Results

I observe that the root mean square errors of the cross validation set decrease as the number of features increase. I submitted multiple results on Kaggle for this model. The scores are 0.79872 for $n = 32$, 0.78595 for $n = 1536$ with no regularization and 0.78479 with regularization.

Table 2: Root Mean Square Errors for Bounded PMF Model

No. Features	Train RMSE	C.V. RMSE	RMSE with User Mean	Leaderboard RMSE	No. Iterations
1	0.528981	0.845488	0.182	-	230
2	0.396184	0.900929	0.275	-	320
4	0.316019	0.914588	0.303587	-	330
8	0.267688	0.897056	0.264698	-	310
16	0.254104	0.86257	0.206512	-	280
32	0.25182	0.836671	0.163543	0.79872	260
64	0.251228	0.821671	0.138342	-	250
128	0.247459	0.813024	0.127288	-	250
256	0.251391	0.810633	0.12253	-	240
512	0.244701	0.809246	0.120423	-	250
1536	-	-	-	0.78595	250

For $\lambda = 0.001$, gradient descent does not converge because the regularization term is too big.

Table 3: Root Mean Square Errors of Bounded PMF with regularization

No. Features	Lambda	Train RMSE	C.V. RMSE	RMSE w. User Mean	No. Iterations
16	0.0	0.528981	0.845488	0.182	230
16	0.0000001	0.251877	0.862303	0.204973	290
16	0.000001	0.220853	0.871911	0.222887	410
16	0.00001	0.131554	0.866618	0.214715	400
16	0.0001	0.210647	0.837719	0.160112	200

Model 3: PMF Model with Similarity Constraint

Setting up

The article *Probabilistic Matrix Factorization* by S. Salakhutdinov and A. Mnih introduces a similarity constraint which influences user's latent features via the movies each user has rated. The user latent feature vector is given by

$$\vec{P}_i = \vec{Y}_i + \frac{\sum_i I_{ik} \vec{W}_k}{\sum_k I_{ik}}$$

The objective function is given by

$$E = \sum_{i,j} I_{i,j} \cdot (R_{ij} - g(\vec{P}_i \cdot \vec{Q}_j))^2 + \frac{\lambda_Y}{2} \sum_i \left\| \vec{Y}_i \right\|_2 + \frac{\lambda_W}{2} \sum_j \left\| \vec{W}_j \right\|_2 + \frac{\lambda_Q}{2} \sum_i \left\| \vec{Q}_i \right\|_2 .$$

The gradients are

$$\vec{\nabla}_{Y_i} E = -2 \sum_j I_{ij} (R_{ij} - g(\vec{P}_i \cdot \vec{Q}_j)) \; g'(\vec{P}_i \cdot \vec{Q}_j) \; \vec{Q}_j + \lambda_P \vec{Y}_i$$

$$\vec{\nabla}_{Q_j} E = -2 \sum_i I_{ij} (R_{ij} - g(\vec{P}_i \cdot \vec{Q}_j)) \; g'(\vec{P}_i \cdot \vec{Q}_j) \; \vec{P}_i + \lambda_P \vec{Q}_j$$

$$\vec{\nabla}_{W_k} E = -2 \sum_{ij} I_{ij} (R_{ij} - g(\vec{P}_i \cdot \vec{Q}_j)) \; \frac{I_{ik}}{\sum_{k'} I_{ik'}} g'(\vec{P}_i \cdot \vec{Q}_j) \; \vec{P}_i + \lambda_P \vec{W}_j$$

Results

Table 4: Root Mean Square Errors for Constrained PMF Model

No. Features	Train RMSE	C.V. RMSE	Leaderboard RMSE
1	0.667	0.996	-
3	0.26	0.917	-
8	0.07	0.889	-
32	0.02	0.7876	0.83528

Discussion

The number of latent features are restricted to low values as the running time is longer than other models mentioned. Low RMSE is achived on the cross validation set but the model seems to do poorly on the leaderboard.

Section 2: Kaggle Book Ratings continued on next page. . .

Model 4: One-vs-All Classification with PMF

Setting up

Since the results for the bounded model does not seem to beat the user mean approach, another model that I attempted is the one-vs-all classfication method with PMF. While the bounded PMF version puts one distribution on all classes of rating, this approach takes each class into account separately.

The data are trained for each of the 5 classes, 1 to 5. For each class, one obtains the probability of the rating belonging to each class and take the class with the highest probability. A slight modification to smooth out the result is also done where I normalize the probabilities and calculate the expectation value of the ratings. For each class, the R_{ij} is converted to a boolean (1 or 0) indicating whether a user-item pair gives the particular rating. Let $g(x) = \frac{1}{1+e^{-x}}$. The objective function is

$$\sum_{ij} I_{ij} \left(-R_{ij} \log g(\vec{P}_i \cdot \vec{Q}_j) - (1 - R_{ij}) \log(1 - g(\vec{P}_i \cdot \vec{Q}_j)) \right) + \lambda_P \sum_j \|P_i\|_2 + \lambda_Q \sum_j \|Q_j\|_2$$

The gradient is given by

$$\vec{\nabla}_{P_i} = -\sum_j I_{ij}(R_{ij} - g(\vec{P}_I \cdot \vec{Q}_j))\vec{Q}_j - \lambda_P \vec{P}_i$$

$$\vec{\nabla}_{Q_j} = -\sum_i I_{ij}(R_{ij} - g(\vec{P}_I \cdot \vec{Q}_j))\vec{Q}_j - \lambda_P \vec{P}_i$$

Results

As the number of latent features increase, the training errors on all 5 classes decrease. (not shown in the table) However, the model does not seem to agree witht the prediction well, measured in root mean square error. The result in Table 5. uses the integer prediction for the class with highest probability. The result in Table 6. uses the expectation value obtained by normailizing the probabilities for all the 5 classes and use them to do weighted average.

Table 5: Root Mean Square Errors One-vs-All Model

No. Features	Train RMSE	C.V. RMSE
1	1.048	1.09
2	0.955	1.053
3	0.901	1.037
4	0.857	1.026
5	0.831	1.010
8	0.789	0.992
16	0.789	0.992
32	0.745	0.973

Table 6: Root Mean Square Errors One-vs-All Model with Expectation Prediction

No. Features	Train RMSE	C.V. RMSE
1	1.1155	1.05322
2	1.09164	1.04119
3	1.0834	1.03713
4	1.07836	1.03412
8	1.06444	1.03174
16	1.0585	1.03006
32	1.05549	1.02872

Model 4: Weighted User-Item Mean

Setting up

I observe that the the PMF models rarely beats the prediction by user mean method and it might not be the best model for this data. I attempt another model which predicts the rating based on the user mean and item mean, if available, with an adjustable weight. That is,

$$r_{ij} = (1-\alpha)\frac{\sum_j I_{ij}R_{ij}}{\sum_j I_{ij}} + \alpha\frac{\sum_i I_{ij}R_{ij}}{\sum_i I_{ij}}$$

Results

The optimal weight is emperically obtained by performing cross validation test. Below summarizes the results for varying weights.

Table 7: Root Mean Square Errors for Weighted User-Item Mean

α	Cross Validation RMSE	Leaderboard RMSE
0	0.793076	-
0.01	0.792135	-
0.02	0.791236	-
0.1	0.785531	-
0.20	0.782184	0.77305
0.21	0.782083	-
0.22	0.782024	-
0.23	0.782009	0.77295
0.24	0.782035	-
0.25	0.782104	-
0.26	-	0.77322
0.3	0.0783089	-
0.4	0.07880	-

The optimal value of α appears to be 0.23 based on the results. Submission of predictions for $\alpha = 0.20, 0.23, 0.26$ results in the lower Kaggle Leaderboard RMSE of 0.77295, which is lower than the predictions from PMF models mentioned.

This method, while not being the most statistically rigorous, reflects the subjective nature of rating based on users. Users who rate highly seem to give high ratings. It can also be because the books rated (and presumably have been read) are self-selective in that each user chooses a particular collection of books that they believe they will like.