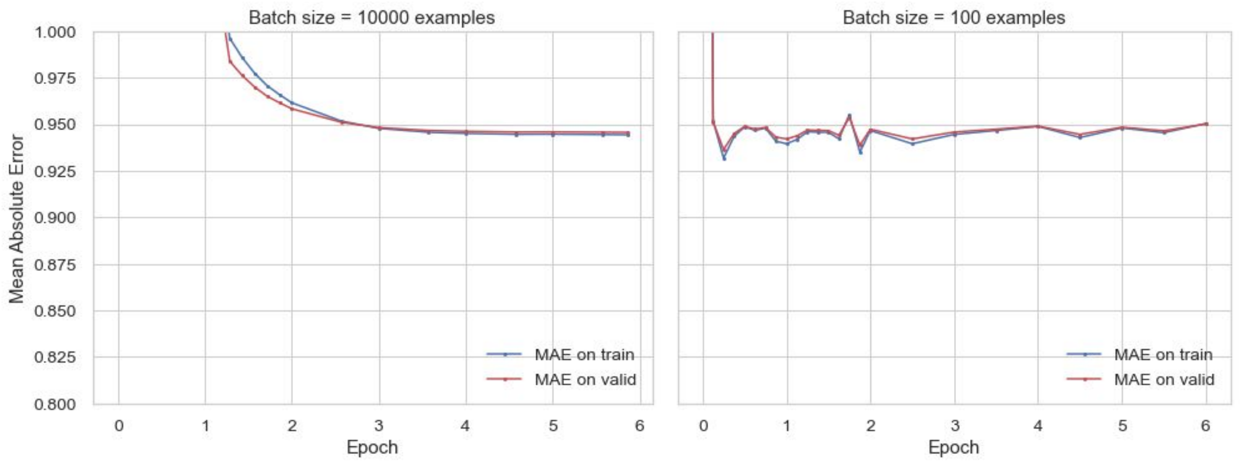
Conrad Walsh and Ethan Schreiber
Introduction to Machine Learning
Project C Report

Problem 1

Figure 1a: Epochs Completed vs Mean Absolute Error for baseline scalar model

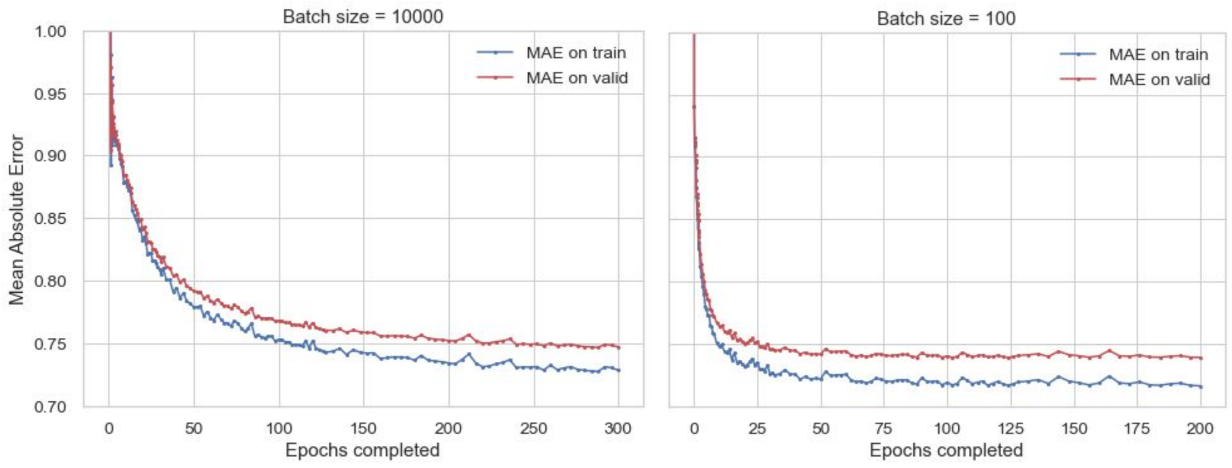


We can see the effects of different batch sizes in the above graphs - the larger batch size gives us a smooth, more stable curve toward the ideal value, while the smaller batch size drops much faster but has a lot more noise. Because of this noise, we actually see the smaller batch size increase in error as it goes through more epochs.

Short Answer 1b:
Because these models are trying to optimize mean squared error, the way to compute the optimal μ value would be to just take the mean of all the training examples. The actual ideal value is about 3.531, and our models got 3.532 and 3.449 for each respective batch size. So our models didn't reach the exact optimal value, but got quite close, so we would say the closed-form solution does agree with our models.

Problem 2

Figure 2a: Epochs Completed vs Mean Absolute Error for Single Scalar Model



This baseline gives us significantly better performance than the first baseline, despite the fact that it is only slightly more complicated. However, it does take many more iterations to train - we didn't see convergence with batch size 10000 until after about 300 epochs. The 100 batch size was significantly faster to converge, however, and we saw good performance after only 75 epochs. We were using a step size of 0.1 still for this problem, which is why both took a while to converge - but the point stands that the larger batch size took longer to converge.

Figure 2b: Learned Bias Parameter for Select Movies

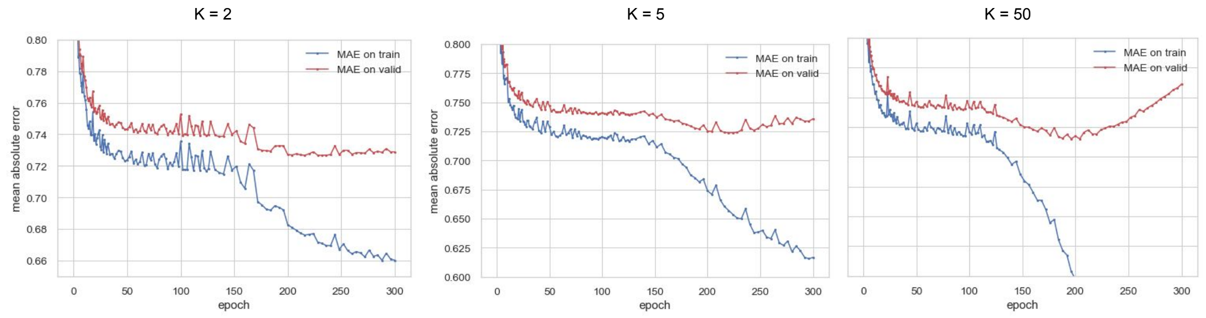
Movie Title	**Learned Bias Parameter**
Toy Story (1995)	0.594
Lion King, The (1994)	0.472
Snow White and the Seven Dwarfs (1937)	0.436
Wizard of Oz, The (1939)	0.791
Sound of Music, The (1965)	0.451
Star Wars (1977)	1.094
Empire Strikes Back, The (1980)	0.921
Return of the Jedi (1983)	0.742
Jurassic Park (1993)	0.400
Lost World: Jurassic Park, The (1997)	-0.388
Raiders of the Lost Ark (1981)	1.054
Indiana Jones and the Last Crusade (1989)	0.666
While You Were Sleeping (1995)	0.251
Sleepless in Seattle (1993)	0.226

My Best Friend's Wedding (1997)	0.017
Nightmare Before Christmas, The (1993)	0.338
Shining, The (1980)	0.476
Nightmare on Elm Street, A (1984)	-0.026
Scream (1996)	0.091
Scream 2 (1997)	-0.261

Large positive bias values mean that the movie is generally liked more than average (since bias is added to the average in prediction). The opposite is true for negative bias - this means the movie is generally disliked. Movies that are generally seen as ‘classics’ tend to have higher learned bias parameters (Star Wars, Raiders of the Lost Ark), and movies that are more niche, like horror movies (Scream), sequels (The Lost World: Jurassic Park), or both (Scream 2) tend to have lower bias values. Interestingly, none of these movies have extremely negative bias values - we think this is due to the list of movies. We’ve heard of almost every movie on this list, which means they’re all relatively well-known. Less well-known movies might have more negative biases if they’re not well liked.

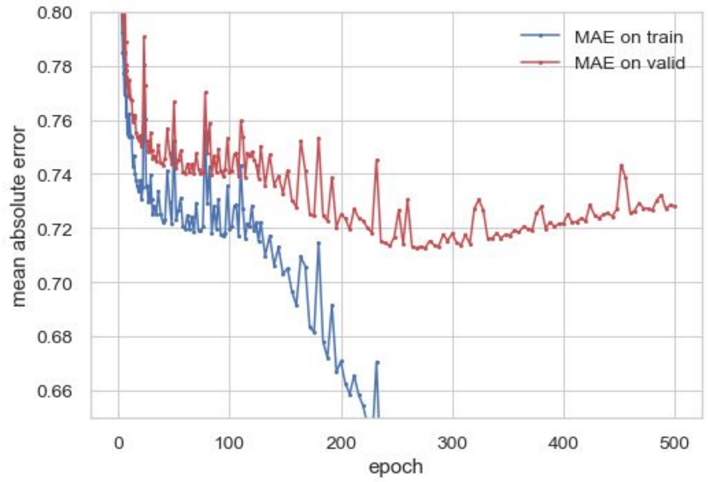
Problem 3

Figure 3a: Mean Absolute Error vs Epoch for One-Vector-Per-Item Model, α = 0



All three of these graphs show clear signs of overfitting after around 200 epochs (validation error increasing, training error still decreasing). However, as K increases the minimum of the validation curve goes lower, and we see the dip in the validation set become more pronounced. To get the best model for each, we needed to use early stopping, since the increase in validation error is so large.

Figure 3b: Mean Absolute Error vs Epoch for One-Vector-Per-Item Model, α > 0, K = 50



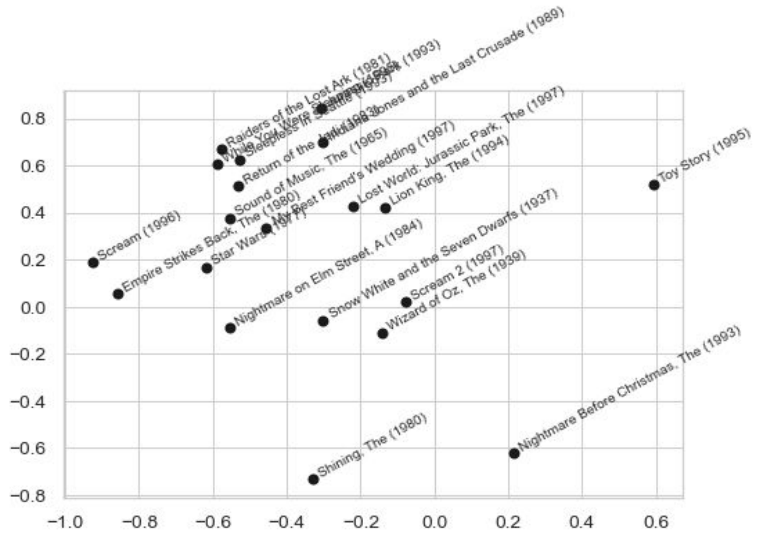
We selected an alpha value of 0.1, batchsize of 1000, and step size of 0.95. The model begins to overfit around 300 epochs, as seen by the increasing validation error and still-decreasing train error. Utilizing early stopping, we do get a better error with this alpha than with no alpha. However, with a larger alpha we saw results quickly regress towards the M2 model - since the vector array starts to approach zero.

Figure 3c: MAE on Validation and Test Sets for Best Models

Model	MAE Validation Set	MAE Test Set
M1 - batch size 10000	0.945	0.946
M2 - batch size 100	0.739	0.747
M3 - K = 2, alpha = 0	0.727	0.733
M3 - K = 5, alpha = 0	0.724	0.730
M3 - K = 50, alpha = 0.1	0.713	0.721

We determined the 'best' version based on validation set performance. For M1, this meant the batch size 10000 model, since the batch size 100 model was unpredictable and did worse after more epochs (even though it had a couple points of better heldout performance). For M2, this meant the batch size 100 model, as it converged much faster and ended with a better heldout performance. The same process was used for the M3 models, except we utilized early stopping to get the lowest possible validation error for each parameter set. We recommend at least 50 factors for M3: more factors would likely be slightly more effective, but would take much longer to train. If time allows, we'd recommend adding more factors. The M3 model with 50 factors and alpha = 0.1 is the best, largely because of the reasons discussed above. A small nonzero alpha value helps avoid some overfitting as well.

Figure 3d: Locations of Embedding Vectors for Select Movies in K = 2 model



While there aren't any hard-and-fast trends, it's telling that the two movies with very low y-values are both horror movies - and the other horror movies have middling y-values. The three Star Wars movies are clustered in the top left, and both Indiana Jones' are near the top. Finally, The Lion King, Snow White, and Wizard of Oz are in the middle. This implies a slight trend of action toward the top and horror toward the bottom, with family somewhere between - but there do seem to be some exceptions to this rule.

Problem 4

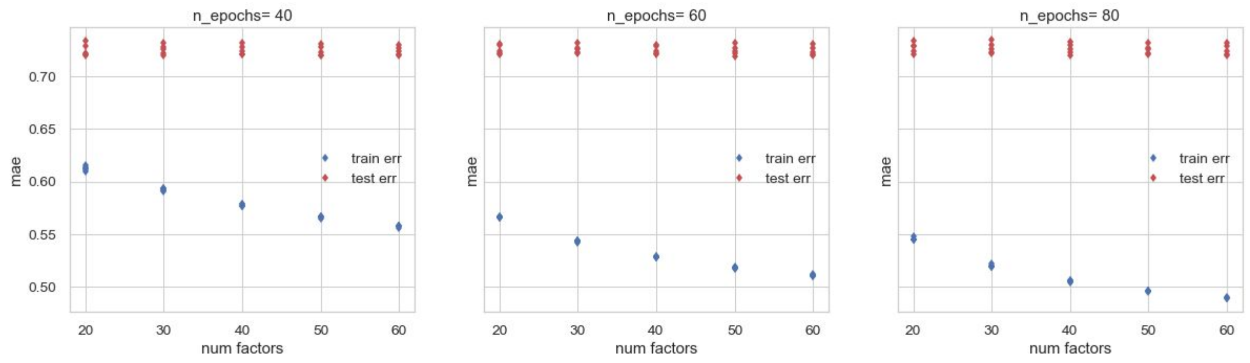
Short Answer 4a:

Our chosen model is the Surprise SVD model, an implementation of the algorithm that won the Netflix prize we talked about in class. The prediction algorithm is the same as the M3 models from earlier problems, and the loss function minimizes MSE with a regularization term on the biases and the per-item vectors. We chose this model as it's a very clean, understandable, and efficient implementation, and it does a very good job in practice. To compute heldout performance for cross-validation, we tried using MSE and MAE as metrics, and got the same results with both.

To select hyperparameters, we used the Surprise implementation of GridSearchCV, which is modeled after sklearn's searcher of the same name. We used grid searching to select the number of factors (K), the regularization on all terms, the learning rate on all terms, and the number of epochs (for early stopping). The 'best' parameters were determined by average performance on heldout data in cross-validation. We started with a very wide range of parameters, and based on the results from that grid search, narrowed in towards the optimal hyperparameters. We repeated this process until we were confident in our hyperparameter choices. We also used 5-fold cross validation, to allow us to use all our data, and ensure we get the most consistent results.

After selecting our hyperparameters, we trained a new model using those best parameters, and used it to predict on the test set. Our final hyperparameters were as follows: 'n_factors': 60, 'lr_all': 0.01, 'n_epochs': 60, 'reg_all': 0.1.

Figure 4b: Plots comparing mean absolute error to the number of factors for various epochs



The above plot shows some of our grid search results for number of factors and number of epochs. The reported best model was with 60 epochs and 60 factors. There are signs of overfitting with 80 epochs, which gave us higher heldout error overall (though it can be hard to see in the plots). Looking at the number of factors, we see that we have reached the point of diminishing returns. As we go from 20 to 40 factors, we see that validation error is still declining - but when we get above 40 factors, the rate of decline slows to be almost imperceptible. With more factors, it would keep going down slowly, but training time would rise significantly. Therefore, we believe these plots support our hyperparameter selections.

Table 4c: Final Surprise Model MAE

Leaderboard Test Set Performance	Internal Heldout Performance
0.7127	0.7237

This table shows our internal heldout performance compared to the leaderboard performance. The leaderboard performance is slightly better, which was unexpected.

Short Answer 4d:
We were pleasantly surprised after submitting our predictions to the leaderboard to find that it performed better there than it did internally. We're not entirely sure why, but we did heavily emphasize heldout performance during hyperparameter selection, which likely helped. Our best M3 model was way better than our best M2 model, which far outstripped our M1 model. This model was also far better than both the M1 and M2 models, but we found that our best model from problem 3 was slightly better on the test set data, despite our best efforts on parameter selection for this model.

Short Answer 4e:
We also tried other models from Surprise, like SVMpp and KNN, but were similarly unable to improve our performance. We think that using demographic data would probably be a good next step - more data on items tends to give a more solid prediction. With more time, we would also like to further investigate the neighbor models discussed in lecture. When we tried this model with the given data, we got very bad results, but we feel that if we had a much larger dataset/we inspected it more, it could be more effective.