Collaborative Filter Pre-processing for Improved Corrupted Image Classification

Lucas Finn

Abstract
This paper investigates the effects of collaborative filtering on the classification of corrupted digit images. Several experiments were carried out using the MNIST digit dataset by applying a corruption model to the original images, two reconstruction algorithms, and an SVM classifier to measure classification accuracy at each stage of processing. The results demonstrate that collaborative filtering, when properly fit to the data, achieves higher accuracy than not filtering or using a Gaussian filter, and retains high accuracy even up to 85% image corruption. The experiments also show that classification performance drops precipitously when the collaborative filter is allowed to over-fit the training images during reconstruction.

Introduction

Collaborative filtering is widely known as the winning algorithm of the Netflix challenge for its ability to predict user preferences given highly sparse data [3]. More generally, it can be applied to matrix completion problems [1]. This paper investigates collaborative filtering as it relates to digit image classification on corrupted images using a support vector machine, with corruption simulating the sparse matrix completion problem.

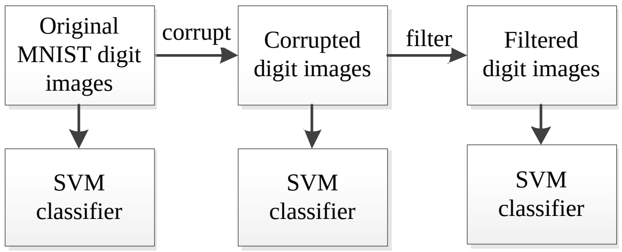

Figure 1 – Experimental setup for assessing SVM classifier accuracy on corrupted and reconstructed images.

Figure 1 shows the experimental setup. The MNIST digit image dataset is taken as input, and contains 70,000 images of digits, each 28^2 pixels [8]. Each digit has been centered and normalized, but the dataset contains multiple handwriting styles; digits can be written multiple ways (e.g. “2” with or without a loop). Example digits are shown in Figure 2 (left). Using the raw images, an SVM was trained and tested, and the confusion matrix was computed as well as the mean digit classification accuracy. This provided a baseline accuracy for comparing other classifiers.

A corruption algorithm was then applied to the MNIST images resulting in degraded image quality (Figure 2 right). The SVM classifier was run to compare the mean classification accuracy of digits. The final processing step applied two filtering algorithms to reconstruct the missing pixels, with the goal of improving SVM classifier accuracy. From these experiments, the effect of corruption and reconstruction was measured having isolated the individual algorithms.

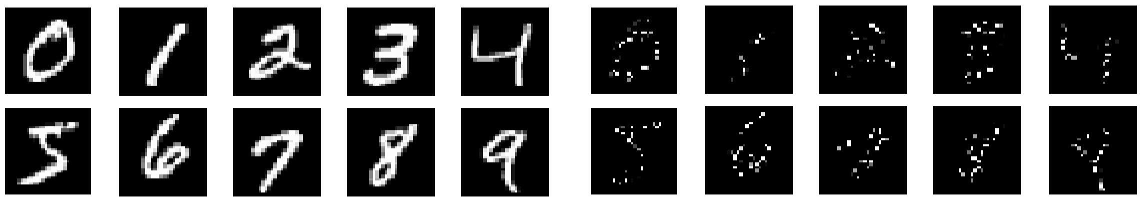

Figure 2 – (left) example digits from the MNIST dataset, (right) the same digits with 85% of the pixels removed, simulating sparse data. Note that some digits remain human-recognizable, but others do not.

Methodology

First, it is interesting to note that while the MNIST dataset appears to have a large number of features (one feature per pixel, 28^2=784 total), these features are largely redundant. For instance, 8% of the pixels are always set to zero, as can be seen in the heat map of unique pixel values (Figure 3 left). Moreover, PCA indicates that 87 out of 784 features in the eigenbasis capture 90% of the variation in the dataset (Figure 3 right). Therefore, it is reasonable to expect that a lower-dimensional image representation would retain a classification accuracy similar to the original high-dimensional representation.

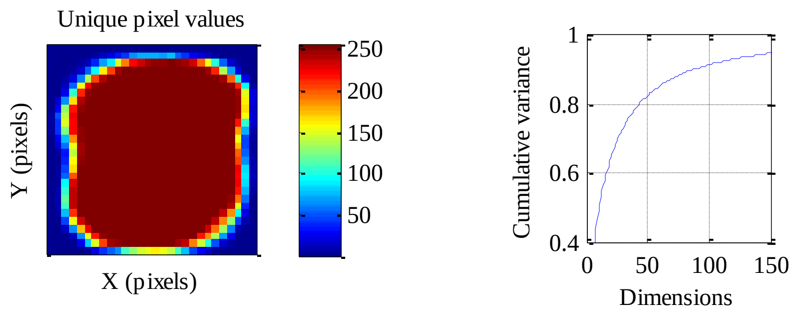

Figure 3 – (left) a heat-map of the number of unique values in the MNIST dataset indicates that 8% of pixels are not used, (right) PCA shows that a low-dimensional feature space captures much of the normalized variation in the dataset.

Several corrupted datasets were created from the original MNIST images by setting a fraction of the pixels to zero, simulating a sparse representation. The corrupted pixels were randomly chosen in each image, not uniformly over the entire dataset. Figure 2 (right) shows the result of corrupting 85% of the pixels in the left image set; note that some digits are not human-recognizable confidently. The corruption fractions tested were 0.0, 0.2, 0.4, 0.6, 0.75, 0.8 and 0.85. The randomization was expected to make SVM classification more challenging because each pixel could indicate the correct value, or a zeroed value containing no discriminating information.

Three filtering algorithms were then applied to the corrupted MNIST dataset to reconstruct the original images: collaborative filtering, Gaussian filtering, and no filtering. As a reconstruction algorithm, Gaussian filtering was chosen to provide a comparison to collaborative filtering [7]. While collaborative filtering exploits the entire available dataset, Gaussian filtering applies to one image at a time. A sigma value of 1.5 was chosen by visual inspection, and simply serves to smooth the information contained in uncorrupted pixels to neighboring pixels.

Given a data matrix D with dimensions MxN, where M = the number of samples (images), and N is the number of features, collaborative filtering factors the data matrix $D = UV$ [1]. The factors U and V have dimensions MxK and KxN, respectively. The resulting data representation is sparse if $MK + KN \ll MN$. With M=70000 and N=784, the reduction in dataset sizes created in this paper are k=60 yields 92.3%, k=30 yields 96.1%, and k=20 yields 97.4%.

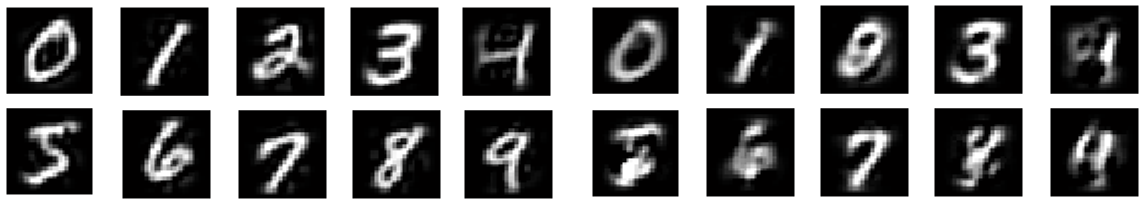

Figure 4 – (left) collaborative filtering with k=60 with no pixel corruption, (right) collaborative filtering with k=20 and 85% pixel corruption. Note that several digits become much more human-recognizable (compare with Figure 2).

The implementation of collaborative filtering used was alternating least squares (ALS), and minimizes the normalized pixel root mean square error (RMSE) [3,5]. This performs linear regression alternating between rows and columns using the uncorrupted information. The results of collaborative filtering on the images in Figure 2 can be seen in Figure 4 with different corruption levels. The parameters for collaborative filtering were chosen by testing several combinations and examining the training RMSE over pixels.

The collaborative filtering algorithm requires two parameters: the number of features in the sparse representation, k, and the regularization parameter, L [1]. Figure 5 measures the pixel RMSE between the training dataset and the low-dimensional reconstruction when no image corruption is applied. Low RMSE can be achieved by choosing a larger k, at the expense of increased model complexity and the danger of over-fitting. The regularization parameter did not significantly affect the RMSE, and so initially k=60 and L=1.05 were used. At high pixel corruption values, k=20 and k=30 were also used. It turns out that tuning collaborative filtering parameters on uncorrupted images, and then applying it to corrupted images resulted in an over-fit. This was remedied by experimenting with lower k.

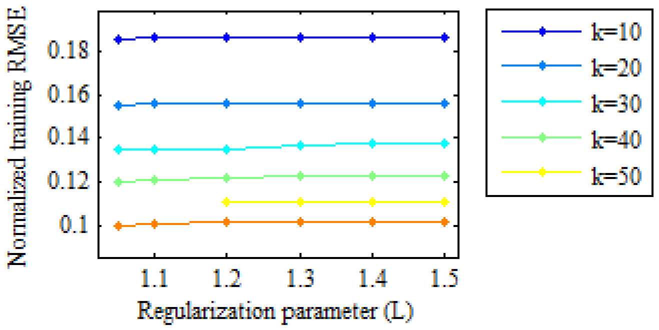

Figure 5 – Collaborative filtering normalized RMSE as a function of number of features (*k*) and regularization parameter (*L*) on the no pixel corruption dataset. Note that the training RMSE decreases with larger *k* and increases slightly with *L*.

Training and testing error were measured on the output of collaborative filtering by computing the RMSE between the original data and the reconstructed pixels [2]. The training error is the normalized RMSE between the original data and the uncorrupted pixels; the testing error is the RMSE between the original data and the corrupted pixels. The ALS algorithm terminates when the training RMSE falls below a small tolerance. Execution took anywhere from five minutes to twelve hours using a 2.66GHz CPU with MATLAB.

The corrupted images were collaboratively filtered in two ways: (1) the entire dataset simultaneously, and (2) the SVM training and SVM testing images separately. In the former case, collaborative filtering built a single representation. In the latter, one representation was created for the SVM training data, another for the SVM testing data. Between these two datasets, the effect of joint versus separate collaborative filtering can be measured. Note that collaborative filtering is unsupervised, so label information is not leaked from testing data into model formation.

Table 1 – Collaborative filtering training and testing tables (left and right tables) of pixel normalized RMSE versus the pixel corruption fraction. Note the decreasing difference between training and testing errors as *k* decreases.

	CF_{60}	CF_{30}	CF_{20}		CF_{60}	CF_{30}	CF_{20}
75%	7.1	11.5	14.1		18.7	17.9	18.4
80%	5.9	10.8	13.6		23.5	19.7	19.4
85%	5.1	9.5	12.6		24.7	24.2	21.4

Table 1 shows the collaborative filtering training and testing errors for high pixel corruption for *k*=60, 30 and 20 on the entire dataset simultaneously. Note that these represent image reconstruction error, not classification error using the SVM. As the fraction of pixel corruption increases, *k*=60 appears to over-fit the data (5.1% training error and 24.7% testing error), while the over-fit is somewhat less for *k*=30 and *k*=20.

The R language kernlab implementation of a multi-class SVM with regularization (ksvm) was used to classify each image dataset [4]. This library builds a multi-class SVM using the one-against-one algorithm. The ksvm radial basis kernel was chosen due to its historically high performance on the MNIST dataset [6]. In addition, ksvm automatically estimates the hyperparameter of the radial basis kernel σ using the sigest feature [4]. This leaves the remaining hyperparameter C, the cost of violating constraints. The default value C=1 was chosen, though future work could estimate this parameter empirically.

Training and testing these SVMs was computationally and memory-intensive, requiring two to twelve hours of CPU time several gigabytes of RAM to complete. These experiments required several hundred total CPU hours and were carried out on a machine with 80 CPUs with a clock speed of 2.66GHz and 256GB of memory over several days.

Results

Table 2 shows the mean classification error for each SVM in a series of experiments with image corruption and reconstruction. Each row denotes the percent of pixels corrupted per image, the left table denotes the SVM training error, and the right table denotes SVM testing error. The four columns in each table are (1) no filtering, (2) Gaussian filtering, (3) collaborative filtering the entire dataset, and (4) collaborative filtering the SVM training images separately from the SVM testing images.

Table 2 – SVM training and testing error tables (left and right tables) for: no filtering, Gaussian filtering, joint and separate collaborative filtering versus pixel corruption fraction.

	NF	GF	CF$_j$	CF$_s$		NF	GF	CF$_j$	CF$_s$
0%	1.4	2.1	1.5	1.5		2.4	2.1	2.2	2.2
20%	2.1	2.7	1.6	1.6		3.6	2.7	2.3	2.4
40%	3.4	3.5	1.8	1.8		5.9	3.5	2.5	2.6
60%	5.7	5.2	2.5	2.4		9.6	5.2	3.5	3.5

With zero percent of the pixels corrupted, the off-the-shelf SVM classifier achieved 2.4% error, which is slightly higher than state-of-the-art [6]. However, because the dataset was divided 70/30 training and testing images instead of the standard 86/14, the comparison is not strictly speaking appropriate. For collaborative filtering, there was only a minor difference in accuracy when the training and testing images were jointly filtered versus separately filtered. Gaussian filtering performed worse than collaborative filtering, especially as corruption levels approached 60%.

Table 3 shows the mean classification error for each SVM at high levels of image corruption. The left and right tables are training and testing errors as in Table 2. The columns are (1) no filtering, (2) Gaussian filtering, (3) collaborative filtering with k=60, (4) k=30 and (5) k=20.

Table 3 – SVM training and testing error tables (left and right tables) at high pixel corruption values demonstrates that the lower-order *k*=20 model outperforms several others, including higher order models.

	NF	GF	CF$_{60}$	CF$_{30}$	CF$_{20}$		NF	GF	CF$_{60}$	CF$_{30}$	CF$_{20}$
75%	9.5	8.0	5.1	6.6	14.1		16.1	10.5	7.0	5.6	7.6
80%	11.8	9.8	8.4	7.9	13.6		19.4	13.0	11.6	9.3	10.2
85%	15.4	13.3	17.2	13.2	12.6		24.7	17.4	25.4	15.3	14.9

Up to 75% corruption, the k=60 experiment outperformed all others, but the error rate rose sharply at 80% and 85% corruption. In fact, k=60 performed worse than no filtering at all. This is likely due to overfitting; the collaborative filter trains on a specific pixel corruption pattern and not the underlying digit. As before, Gaussian filtering continued to perform better than no filtering. At high levels of corruption, it seems that a lower-order model (k=20 and k=30) outperformed the other filtering methods. In fact, at 85% corruption, it is reasonable to conclude that the correct model complexity is approximately near k=20 and k=30.

Figure 6 shows the confusion matrices for two experiments with high image corruption and collaborative filtering applied. The quantity $p(d_1|d_2)$ is shown in decibels to accentuate misclassifications (the off-diagonal yellow-orange values), denoting the probability that an image was declared to be digit d_1 given that the image actually represents digit d_2. As expected, the digits 4, 7 and 9 were confused rather often, as were 3, 5 and 8 [6]. From Table 3, the mean SVM classification error for these experiments is 7.0% and 14.9%.

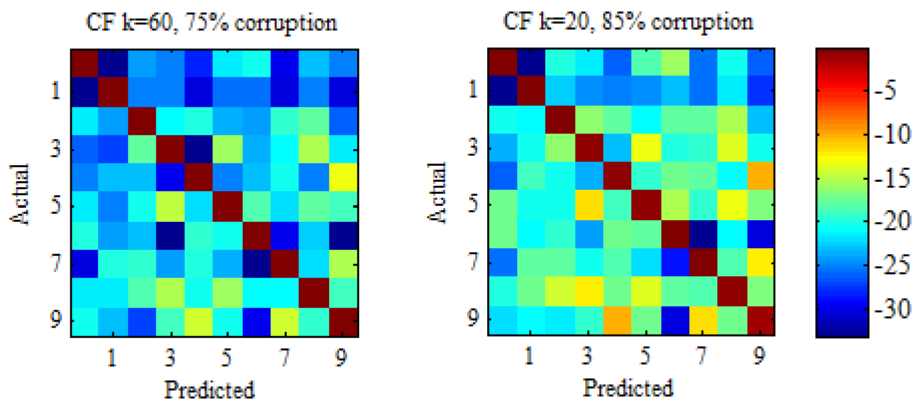

Figure 6 – Example confusion matrices in dB: (left) collaborative filtering with a high order model k=60 at 75% image corruption, and (right) a low order model k=20 at 85% corruption confuses similar-looking digits.

Conclusions

There are several interesting results observable from these experiments:

(1) The baseline SVM classifier achieved high mean digit classification accuracy, but was slightly worse than state-of-the-art. This is likely due to training the SVM with a 70/30 data split, to keep more images for collaborative filtering instead of the standard 86/14 split. Future work might compute the accuracy on the 86/14 split.

(2) When properly tuned, collaborative filtering provides both a low-order reconstruction of the corrupted data set and improves overall SVM classification accuracy across a wide range of pixel corruption fractions.

(3) At even modest amounts of corruption, Gaussian filtering improved mean SVM classification accuracy 1-7% compared to not filtering, likely because it smoothes out the irregular information contained in the randomly corrupted pixels. The smoothed data more naturally admits an SVM separating hyperplane.

(4) At corruption values $\leq$60%, collaborative filtering with the relatively high-order model k=60 achieves higher accuracy than Gaussian filtering or not filtering at all. Moreover, it is highly interesting that collaborative filtering the training/testing images separately or jointly did not affect the SVM classification accuracy, which indicates that the training image dataset size is sufficiently large for the model.

(5) At corruption values $\geq$75%, collaborative filtering indicates that the high-order model k=60 over-trains on the training dataset, and is reflected in low training error and high testing error in both collaborative filtering and the SVM classifier. This can be remedied by choosing a low-order k=20 or k=30 model, and results in high accuracy even at pixel corruption fractions up to 85%. Future work might include 90% or 95% corruption.

(6) As the fraction of corrupted pixels increases, the size of the collaborative filtering training dataset decreases, and therefore it makes sense to simultaneously lower the collaborative filtering model complexity. In these experiments, this was accomplished by lowering k, but future work could investigate retaining a high k while simultaneously increasing the regularization parameter L. Additionally, the SVM hyperparameter C (the cost of violating constraints) may be tunable to increase SVM accuracy.

(7) As the fraction of corrupted pixels increases, whether or not collaborative filtering is trained on the joint SVM training/testing images begins to affect the SVM classification accuracy. The joint collaborative filtering achieves a higher SVM accuracy than separately collaborative filtering the data, likely because the available dataset of uncorrupted pixels is relatively small.

References

[1] Carlos Guestrin. 2013. Collaborative filtering matrix completion alternating least squares. Retrieved from http://courses.cs.washington.edu/courses/cse599c1/13wi/slides/matrix-factorization-sgd-nmf.pdf.

[2] Xiaoyuan Su and Taghi M. Khoshgoftaar. 2009. A survey of collaborative filtering techniques. Adv. in Artif. Intell. 2009, Article 4 (January 2009), 19 pages. DOI=10.1155/2009/421425 http://dx.doi.org/10.1155/2009/421425

[3] Jianchao Yang, Jianping Wang, and Thomas Huang. 2011. Learning the sparse representation for classification. Multimedia and Expo (ICME). DOI= 10.1109/ICME.2011.6012083.

[4] Alexandros Karatzoglou, Alex Smola, Kurt Hornik and Achim Zeileis. 2004. An 'S4' package for kernel methods in R, Journal of statistical software (11):1-20.

[5] Michael Curtis et. al. 2012. A collaborative filtering model: statistical properties of alternating least squares. UMBC Review: Journal of Undergraduate Research, (13): 10-21.

[6] Ming Wu and Zhen Zhang. 2010. Handwritten digit classification using the MNIST data set. Retrieved from http://www.stt.msu.edu/~zhangz19/index.files/files/cse802report.pdf

[7] Thomas Funkhouser. 2000. Image sampling and reconstruction. Retrieved from http://www.cs.princeton.edu/courses/archive/fall2000/cs426/lectures/sampling/sampling.pdf

[8] Yann LeCun, Corinna Cortes, Christopher Burges. 2012. The MNIST database of handwritten digits. Retrieved from http://yann.lecun.com/exdb/mnist