Optical Character Recognition: Classification of Handwritten Digits and Computer Fonts

George Margulis, CS229 Final Report

Abstract

Optical character Recognition (OCR) is an important application of machine learning where an algorithm is trained on a data set of known letters/digits and can learn to accurately classify letters/digits. A variety of algorithms have shown excellent accuracy for the problem of handwritten digits, 4 of which are looked at here. Additionally, we attempt to extend these techniques to the harder problem of classifying various characters written in different fonts, and achieve accuracies of ~4% and ~7% on these two data sets, respectively, using support vector machines. Finally, we implement PCA to see how the algorithms will fare using a smaller dimensional space and find the interesting result that the more difficult to classify data set (computer fonts) can be accurately classified using a smaller dimensional projection space.

Data Sets

For this project, I used two data sets of digits: a subset of the MNIST database of handwritten digits[1] and the "not MNIST"[2] database of various fonts/images of computer writing (perhaps something that can be found from trying to read an image). Each set contained 10 various characters/digits, 1500 instances of each classification and 300 testing examples for each letter/digit, with representative images shown in Fig. 1. Each of the digits is a 28x28 pixel image, resulting in a 784 dimensional space. The MNIST database is well-studied and has had many algorithms applied to it for letter classification[3], and serves as a baseline for implementing our algorithms. The notMNIST database appears to be a harder task, and so we want to understand why that is so and where our algorithms fail.

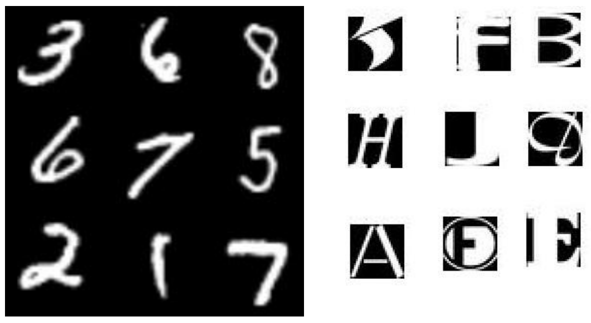


Fig. 1 Images of representative digits and characters from the MNIST (left) and notMNIST datasets.

Algorithms Implemented

kNN

We implemented k-nearest neighbors, where the classification of a point only depends on the closest k points in Euclidean distance. Varying the values of k, the best results were obtained by using small values of k and results are shown for k=3.

C-SVM with 4th degree polynomial kernel

With the help of LIBSVM[4], we implemented C-SVM which attempts to solve the following optimization problem:

$$min_{\varepsilon_i,w,b}\ \frac{1}{2}||w||^2 + C\sum_{i=1}^{m}\varepsilon_i \text{ s.t.}$$

$$y^{(i)}\left(w^T x^{(i)} + b\right) \geq 1 - \varepsilon_i, \qquad i = 1,2 \dots . m$$
$$\varepsilon_i \geq 0, \qquad i = 1,2 \dots . m$$

Utilizing a 4th degree polynomial kernel $\varphi(x)^T \varphi(y) = (\gamma x^T y)^4$. Once the optimum margin classifier is known, binary classification can be performed. Since we are classifying a set of 10 digits/characters, we separately implemented one vs. one classification, where each training setoff each character is trained against each other training set. Finally, during testing, the character is classified according to all the optimum margin classifiers and then the one which occurs the most often is taken as the classification. Values for C and gamma were determined by getting low training error.

C-SVM with linear kernel
Using LIBLINEAR[5], we also implemented an SVM using a linear kernel, very similar to above. However in this case, the kernel was a linear function, and the C term in the above equation was $C \sum_{i=1}^{m} \varepsilon_i{}^2$, ie. L2 regularization rather than L1. Again, we used one vs. one method of making a multi-class classifier.

L1 Regularized logistic regression
We used LIBLINEAR to implement logistic regression with L2 regularization, where there is a term in the maximized likelihood. We would like to maximize $\sum_{i=1}^{m} \log p\left(y^{(i)} \middle| x^i; \theta\right) - \theta^T \theta$ w.r.t. θ where $p\left(y^{(i)} \middle| x; \theta\right)$ is a sigmoid function: $\left(\frac{1}{1+e^{-\theta^T x}}\right)^{y^{(i)}} \left(1 - \frac{1}{1+e^{-\theta^T x}}\right)^{1-y^{(i)}}$This is once again a binary classifier, and we implemented a one vs. one extension to classify our 10 digits/characters.

It should be pointed out that while the first 2 techniques lead to non-linear classifiers (the 4-th degree polynomial kernel maps to a high dimensional space, where it fits a linear classifier, but that boundary will not be linear in the original space), while the second two decision boundaries are hyperplanes in the original space of the characters. The set of handwritten characters is in an extremely high dimensional space (784 pixels), but is without a doubt, much lower dimensional. In addition, this space is highly non-linear: one might imagine that space of characters are a k-dimensional manifold mapped into R^n, where n in this case is 784, and k is much much less than n.

Results of Classifiers
The results of our 4 techniques are shown below in table 2, with our polynomial SVM outperforming the other techniques by a reasonable margin. Additionally, kNN performed the second best, which may be rather surprising due to the extremely unsophisticated nature of the technique, however, it is interesting that the best techniques are ones where the classifier boundary is not linear in the original space of the characters. This might be expected, and

Algorithm	Classification Error MNIST	Classification Error MNIST
Logistic Regression	9.6%	10.6%
Linear SVM	11.1%	14.6%
SVM 4 poly	4.1%	7.7%
kNN (3)	6.6%	10.5%

Table 1. Classification error

points to the fact that the data set is highly non-linear. However, it is interesting that the gap between the non-linear techniques and linear techniques seems to go down a bit for the notMNIST dataset, despite the fact that is more challenging to classify.

It should be pointed out that current state-of-the-art OCR techniques are able to get sub 1% classification error on the full 60000 training example MNIST database. Our best technique had 4% error on a smaller subset of the data set. On the full dataset, we were able to get errors below 3%, although classification took so long that we were unable to repeat it for our final best parameters C and γ. However, our goal was to use these techniques to classify and understand the datasets rather than purely get the best performance. Additionally, this performance is fairly close to the best reported performances on the MNIST dataset using these techniques, which suggests that our performance on the notMNIST dataset is pretty good (for these 4 techniques).

Dimensionality Reduction (PCA)

While classification using the values of all the pixels gives good results, the extra dimensionality can lead to "random noise" which will make our measurement less accurate. Additionally, by lowering the number of dimensions of the data, we are able to run our classification algorithms much faster and on larger data sets. A technique that can reduce dimensionality is principle component analysis, or PCA, where the data is projected on its k principle components, or k directions of largest variance. In addition, PCA is a good tool for visualizing high dimensional data: we show the projection of a subset of the MNIST data set on the first two principle components of our full training set in Fig. 3, and we see the separation between 1s and 2s in the dataset.

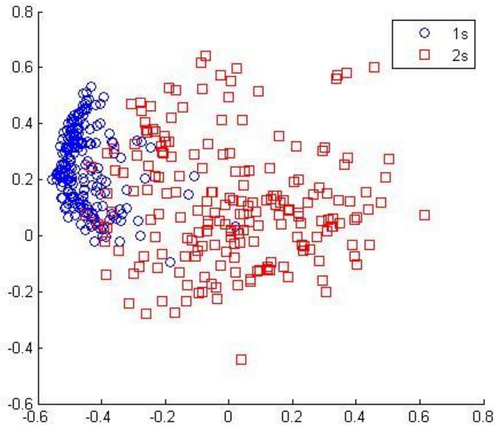


Fig 2. Projection of 1's and 2's from the MNIST data set onto first 2 principle components (x and y axes)

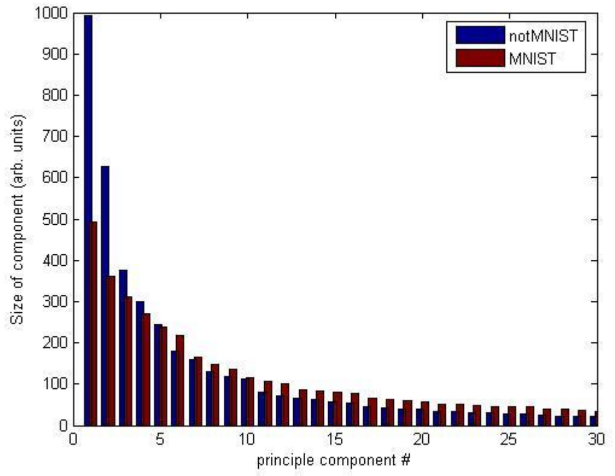
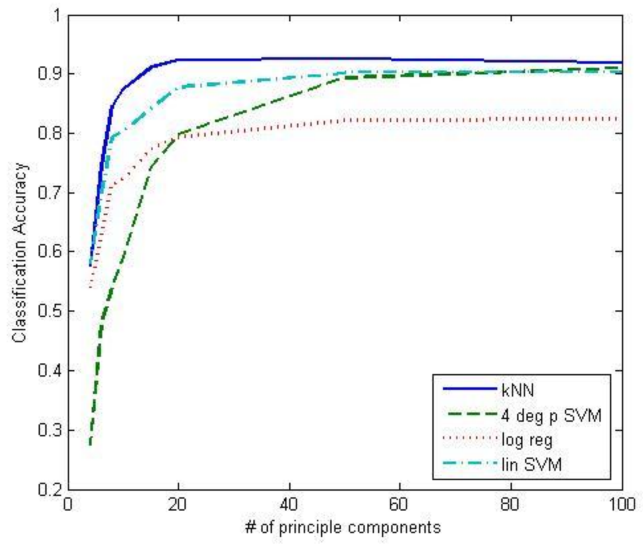


Fig 3. Relative size of the projection onto the first 3 principle components for the MNIST and notMNIST data sets.

We performed PCA on 5000 training examples from each of the datasets and the magnitude of the first 30 principle components is shown below in figure 3. Interestingly enough, we see that the relative size of the first few principle components for the MNIST data set is smaller than the size of the notMNIST projection onto the first few principle components. This is particularly interesting because while our results show that the notMNIST data set is more complex, and harder to classify, it actually can be described better by a projection onto fewer components. This is intriguing, as it suggests that the notMNIST data is less “non-linear” and that projecting onto a small dimensional space, once can still get accurate classification. Upon thinking about this further, this seems to make intuitive sense: handwritten characters have interesting nonlinear embellishment such as additional loops, or curvature. While there is more variation in fonts, there are more linear transformations, such as widening a character and less “variation” around the border.



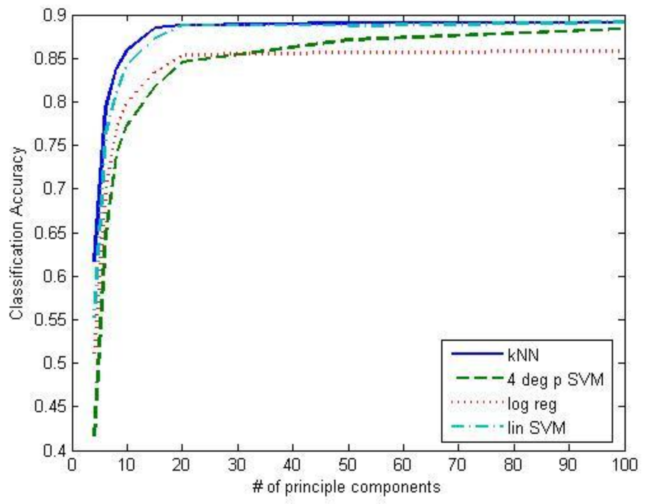


Fig 4. Classification accuracy of the 4 algorithms as a function of # of principle components projected upon. Top graph is for the MNIST data set. Bottom is for the notMNIST dataset.

Using this intuition, we decided to see if that meant we can get high classification accuracy was still high using a smaller number of principle components. Using the same 4 algorithms, but with a smaller number of training examples and testing examples: 5000 and 1000 respectively (for speed), we obtained the following classification accuracy vs. number of principle components used (Fig. 4). As would be suggested by our PCA, we find that the classification accuracy drops off quickly below about 20 principle components for the MNIST data set and about 15 components for the notMNIST data set. This seems to be consistent with our interpretation that the notMNIST data set is “more linear’, which allows PCA to be more effective. Additionally, this might be the reason that the linear classifier techniques used are closer to the non-linear techniques on this data set. It is pretty interesting that despite the notMNIST data being harder to classify, it appears to exist in a smaller dimension Euclidean space. I would guess this to be interpreted that there must be more variance in the various fonts of the notMNIST data set as compared to the standard MNIST dataset.

Improvements/Additional Work

There can be a variety of improvements on the work to investigate the notMNIST dataset further. There has been a wealth of algorithms implemented for OCR[3], with varying degrees of success, up to classification errors of less than 1%: neural networks, various other SVMs with different kernels. It would be interesting to see how these various algorithms fare on the notMNIST dataset. Also, non-linear dimensional reduction using algorithms like ISOMAP[7], LLE[8] or nonlinear extensions of PCA can be used to project the data onto a nonlinear subspace, in order to understand the intrinsic dimension of the data. From here we might be able to further understand why the classification of the notMNIST data set is a hard problem and figure out algorithms that allow for better classification accuracy.

Conclusion

We have applied a variety of algorithms to digit/character classification using 2 dataset: one of handwritten characters (MNIST) and one of a variety of fonts (notMNIST). Both of these are interesting for the problem of OCR, where one might try to read a handwritten letter or address, or read the font on a banner image in a website. SVMs were able to achieve the best accuracy on both of the data sets, and this was attributed to the ability to handle nonlinearity. However, PCA showed that the linear component of the notMNIST dataset was actually less than that of the MNIST and we were able to achieve high classification accuracy with slightly less principle components, which was a bit interesting considering the difficulty of classifying this data, which I thought it was interesting tidbit of insight.

References

[1] MNIST digit database of handwritten digits. Yann LeCun, Corrina Cortes. http://yann.lecun.com/exdb/mnist/

[2] the notMNIST data set. Yaroslav Bulatov. http://yaroslavvb.blogspot.com/2011/09/notmnist-dataset.html

[3] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. "Gradient-based learning applied to document recognition." Proceedings of the IEEE, 86(11):2278-2324, November 1998

[4] Chih-Chung Chang and Chih-Jen Lin, LIBSVM : a library for support vector machines. ACM Transactions on Intelligent Systems and Technology, 2:27:1--27:27, 2011. Software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm

[5] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin. LIBLINEAR: A Library for Large Linear Classification, Journal of Machine Learning Research 9(2008), 1871-1874. Software available at http://www.csie.ntu.edu.tw/~cjlin/liblinear

[7] J. B. Tenenbaum, V. de Silva and J. C. Langford. A Global Geometric Framework for Nonlinear Dimensionality Reduction. Science 290 5500,2319, (2000)

[8] Sam Roweis, and Lawrence Saul. Nonlinear dimensionality reduction by locally linear embedding. Science, 290 5500, 2323 (2000).