

# Classification of Handwritten Digits using Two Feature Spaces

Charles Otto  
Matthew St. Peter

May 2, 2008

**Abstract** Handwritten digit recognition is an important and challenging problem in pattern recognition. This paper reports on experiments done on the MNIST set of handwritten digits, using two different feature spaces and a variety of classifiers on each space. Performance is compared to benchmarks in the field.

## Table of Contents

Introduction.....	1
Feature Extraction using HOSVD.....	2
Feature Extraction using Chaincode Directions.....	3
Classifier Description.....	4
Classifier Performance.....	6
Conclusions.....	7
References.....	8

## Introduction

Handwritten digit OCR recognition has important applications in automatic mail sorting, as well as in recognition of account numbers on checks. The first fully-computerized handwritten OCR system was installed in Los Angeles over 25 years ago, and the Postal Service now has the capability to read multiple-lined addresses and automatically print bar codes at a rate of 450,000 pieces of mail per hour. In fact, automatic recognition of handwritten forms is now prevalent in many industries where forms- and paper-processing is common; e.g. the banking, utility, and industries all use handwritten OCR methods to automatically determine account numbers and payment amounts.

Though it is machine-stamped and not handwritten, an emerging use of OCR is for Automatic Number Plate Recognition, already used in the UK. Cars in London may be monitored real-time as part of the City of London's congestion charge program. In fact, as of 2006, the UK has the capacity to track any vehicles' movements throughout the country for a period of at least two years.

Though the state of the art is quite impressive, OCR (and particularly handwritten OCR) is not problem-free. Individual differences in handwriting are a source of significant within class variation; indeed, an individual will not write digits in exactly the same way all the time. Different conventions for drawing digits introduce further variation; some include a diagonal slash on zeros or a small horizontal stroke across the middle of the sevens further increasing within class variability.

Practical applications may also face difficulties in selecting the digits to be recognized. Potential problems in determining digit selection can include:

**Normalization:** determining characters or character blocks from the background may prove difficult in some cases, such as colored pencil on colored envelopes.

**Segmentation problems:** Uneven writing may lead to poor segmentation results.

**Difficult-to-classify handwriting:** Handwriting may be highly stylized or purely illegible, making computer classification, not to mention human classification, remarkably difficult.

The first two problems are remedied through the preprocessing already performed on the MNIST database of digits. The MNIST set focuses solely on the recognition task because it guarantees a single character per example image and also ensures that the given character is centered in the digital image [Lec08]. Thus the problem to be attacked is that of classification, with a minimal amount of preprocessing work necessary. With well-separated classes, attention may be turned onto feature extraction

Treating the scanned digits as images allows for simple, fast, dimensionality reduction using a generalization of the SVD/PCA algorithm detailed by Savas and Elden in [Sav07]. Savas and Elden also show that the resulting dimensionality reduction requires merely having to compute linear least-squares residuals for all of the "eigendigit" bases to achieve a reasonable (~95%) classification accuracy with maximal

online speed. Traditional algorithms such as k-NN and SVM can also be used [Sav07].

In recent years error rates of less than 1% have been reported on the MNIST data set from several different classifiers. Belongie et al. report an error rate of .63% from an edited k-NN classifier with features based on the distribution of a sampling of edge points (as flagged by a Canny edge detector) relative to each other in space [Bel02]. Liu et al evaluated four different feature vectors on eight classifiers each on the MNIST data set. The feature vector with highest accuracy on all tested classifiers was computed by calculating the local gradient at each point of the input image using a Sobel operator and decomposing the calculated gradient into the 8 chaincode direction components [Liu03]. Using an SVM based classifier with a radial basis function resulted in an error rate of .42% for the gradient based features, while a polynomial classifier network had an error rate of .58% for the same features. The results from gradient based features were more accurate than a PCA reduction of the image pixels for all classifiers tested in [Liu03].

Results show significant accuracy gains over PCA approaches by using more sophisticated feature vectors. This project will develop the PCA–HOSVD approach as a baseline, then use the chaincode direction approach in order to improve accuracy. A k-NN classifier based on the chaincode features will be the next benchmark; although k-NN classifiers may fall behind more sophisticated methods in both accuracy and computational cost they do provide a baseline by which to judge other methods. The most accuracy classifier on MNIST reported in [Liu03] was an SVM classifier with an RBF kernel, hence it will be repeated here. Finally, the two approaches will be combined using various fusion classifiers.

## Feature Extraction Using HOSVD

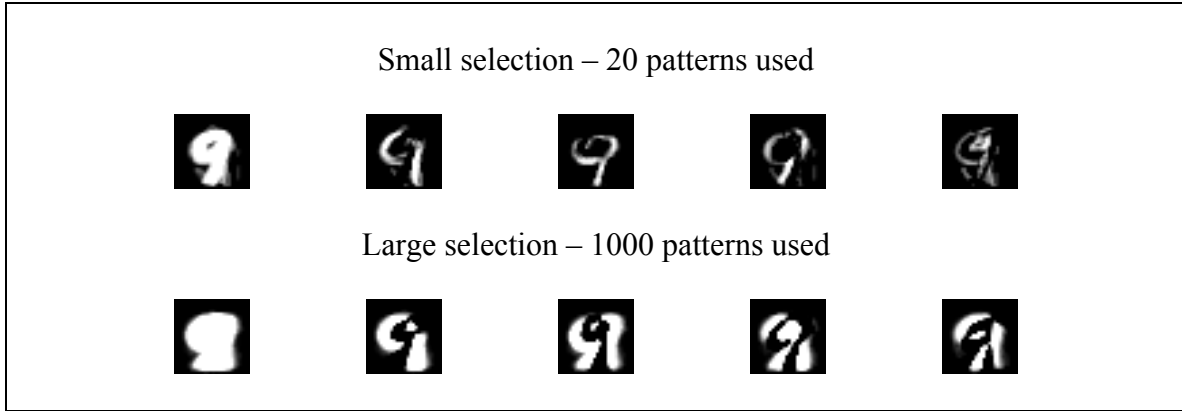
The singular value decomposition (SVD) of a matrix  $A$  is given by  $A = U\Sigma V^t$ , where  $U$  and  $V$  are orthogonal matrices. It is equivalent to the PCA algorithm – it orders the eigenvalues of  $A$  in descending order and reveals the dominant subspaces to be the corresponding eigenvectors. The decomposition can also be shown to be equivalent to writing matrix  $A$  as the sum of  $k$  rank one matrices  $A_k$ .

The SVD of matrix  $A$  can be generalized to a tensor (3–mode array)  $T$  by the higher-order SVD (HOSVD) method. HOSVD has also been used in data compression and face recognition. Each set of training digits  $j = 0, 1, \dots, 9$  can be ordered as a tensor (3-array)  $A^{(j)} \in R^{28 \times 28 \times k}$ , where  $k$  is the number of training samples in each class. A sufficiently large subset of the training digits are selected from which to take the HOSVD.

The HOSVD of the tensor representing samples of digit  $j$  can be written as the outer product of basis images  $A_v^{(j)}$  with weights  $w_v^{(j)}$

$$T^{(j)} = \sum_{v=1}^k A_v^{(j)} \times_3 w_v^{(j)}$$

As with the SVD for matrices, the HOSVD may be truncated to  $k' < k$  in order to provide a lower-dimension approximation. Specifically,  $k'$  is the number of basis images to store for each class. We choose  $k' = 20$ , after which very little additional data is stored.






**Figure 1.** Basis images for digit 9 using different random selections. Too few images results in poorly-defined basis images, while a large number provides good performance.

Once all the basis images are computed for all classes, each training pattern is presented to the basis of class  $j, j = 0, 1, \dots, 9$ , in an attempt to determine how closely the training pattern is spanned by each basis. The distribution of residuals becomes the feature vector: each feature of the training pattern can be thought of as the degree of similarity to a basis of eigenimages.

### Feature Extraction Using Chaincode Directions

The training images are convolved with discrete Sobel operators to determine the  $x$  and  $y$  components of the gradient. The (normalized) components of the gradient are shown below

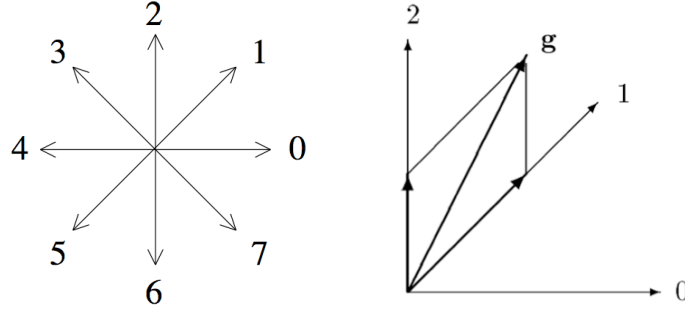
Original Image Matrix	$x$ -Derivative	$y$ -Derivative
$A$	$g_x = \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & 1 \end{bmatrix} * A$	$g_y = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} * A$
		

**Figure 2.** Gradients computed by Sobel operator.

Once the  $x$  and  $y$  directions of the gradient have been computed, they are projected onto the maps onto the eight chaincode directions shown below. Each value  $n = 0, 1, \dots, 7$  refers to angle  $\frac{n\pi}{4}$ . First, the gradient map  $(x, y)$  for each pixel must be transformed into polar coordinates  $(r, \theta)$  via the usual transformation

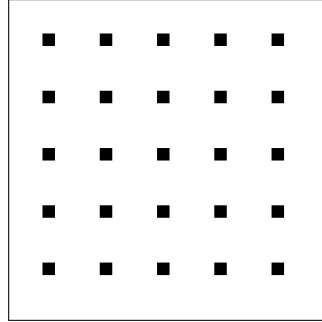
$$r = \sqrt{g_x^2 + g_y^2} \quad , \quad \theta = \tan^{-1}\left(\frac{y}{x}\right).$$

For instances in which the gradient angle falls between the eight chaincode directions, it is decomposed between the two nearest angles, as shown below.



**Figure 3.** Eight chaincode directions and projection of gradient.

The result is then the ‘directional derivative’ of the image in eight directions. Since the full feature vector now has  $28 \times 28 \times 8 = 6272$  elements, each directional image is sampled by convolving uniformly spaced points throughout the image with a  $5 \times 5$  Gaussian kernel in order to measure pixel density in regions. Twenty-five measurements are made throughout each directional image, yielding a final feature vector of length  $25 \times 8 = 200$ .



**Figure 4.** Pixel density measurement is done at the twenty-five points marked in black.



### Classifier Description

A brief description of the more important classifiers used to perform digit recognition within this project are given below. These classifiers are all implemented using either dedicated pattern recognition software Weka, from the University of Waikato [Wit05], or with the STPRTool pattern recognition toolbox for MATLAB, from the Czech Technical University in Prague [Sch02]. The decision tree method J48 and its boosted cousin are not discussed here, due to their poor performance on the chaincode feature space.

### Least Squares Classifier

The least squares classifier is used on the HOSVD feature vectors only. It merely looks at the ten-dimensional vector for each test pattern and chooses the minimum value, the

minimum spanning residual. The interpretation is thus: since the image has a lower residual for class  $j$  than any other class, it bears the most similarity to the basis images of that class, and thus must belong to that class. Unfortunately, the following figure shows that poorly written digits can be ‘extremely’ misclassified: the residual for the correct class is far higher than the residual for an incorrect class.

Original Image	Nine-basis	Zero-basis
		
Poorly written nine	Residual 0.3403	Residual 0.1565

**Figure 5.** Spanning a test image across different bases. The digit in question is misclassified as a zero with a small tail.

The least squares classifier is a good choice for an initial classifier. The reasonable high accuracy rate (shown in the Performance section) coupled with the speed at which classification can be made is useful as a ‘first line’ in a nested classification system. However, the implementation of such a classification system is beyond the scope of this project.

### k-NN

The k-NN rule is computationally the easiest to understand. A ball centered around each test pattern is inflated in the feature space until exactly  $k$  training patterns are located inside it. Then the test pattern is classified as  $j$  if the majority of training patterns  $\{x_1, x_2, \dots, x_k\}$  are also labeled  $j$ . If there is a tie in label count, then the test point is classified to be the same class as  $x_c$ , where  $\|x_{test} - x_{closest}\| < \|x_{test} - x_l\|$ ,  $l \neq closest$ .

With a large number of training patterns, the k-NN rule becomes computationally intensive, as a large number of distance comparisons must be made. Further, there are classifiers that provide better classification performance.

### SVM – RBF kernel

The SVM approach with a linear discriminant function constructs a separating hyperplane in  $d$ -dimensional space that maximizes the margin between two data sets. An SVM typically handles binary problems; to generalize to multiclass classification, a combination of binary classifiers is used. In this case, the ten-class problem is decomposed into 10 binary problems, separating class  $j$  from class *not-j*.

Further, the discriminant function may be generalized to be nonlinear for improved performance through the means of the ‘kernel trick,’ where a dot product may be replaced with a nonlinear kernel function. Here, the kernel function is taken to be a Gaussian. Thus, the discriminant function for test pattern  $x$  takes the form:

$$f(x) = \sum_{i=1}^n y_i \alpha_i \exp\left(-\frac{\|x - x_i\|^2}{2\sigma^2}\right) + b$$

where  $n$  is the number of training patterns,  $y_i$  is the class label, and  $b$  is bias. It then remains to solve for the appropriate weights  $\alpha_i$  through the maximization of the quadratic optimization problem

$$W(\alpha) = \sum_{i=1}^n \alpha_i - \frac{1}{2} \sum_{i,j=1}^n \alpha_i \alpha_j y_i y_j \exp\left(-\frac{\|x_i - x_j\|^2}{2\sigma^2}\right)$$

subject to  $\sum_{i=1}^n \alpha_i y_i = 0$  and  $0 \leq \alpha_i \leq C$ ,

where control parameter  $C$  deals with error tolerance.

Once the optimization is complete, a subset of the training patterns with nonzero weights  $\alpha_i$  remain as the support vectors that determine the discriminant function values. For classification, all discriminant functions  $f_j(x)$  are computed for  $j = 0, 1, \dots, 9$ : the test pattern is then assigned to the class with the highest  $f_j(x)$ .

### Classifier Performance

The performance of various classifiers on the two feature spaces is given below. Also, the performance of basic fusion classifiers is also evaluated.

#### Twenty-image basis from HOSVD

Classifier	Error Rate (%)
Least-squares	4.64
3-NN	<b>3.93</b>
SVM with RBF kernel	12.90

#### Chaincode maps from Sobel gradient

Classifier	Error Rate (%)
SVM with RBF kernel	<b>0.70</b>
AdaBoost with SVM-RBF	0.85
PCA then SVM-RBF	1.58
3-NN	1.22
J48	8.10
Boosted J48	4.30



### Fusion classifiers

Fusion Classifier	Error Rate (%)
Product	0.77
Sum	0.70
Weighted Sum (0.3 LS + 0.7 SobelSVM)	<b>0.68</b>

### Conclusions

A variety of classifiers were tried on two separate feature spaces derived from handwritten digits. The HOSVD approach is shown to provide reasonable performance at a low on-line training cost, while an approach using a support vector machine with Gaussian kernel is shown to provide superior classification accuracy, albeit at a much increased computational cost.

Fusion classifiers that combine the two feature spaces fail to provide increased performance. This is likely due to the fact that the HOSVD classification is much less accurate than the SVM–RBF approach. Thus, the SVM–RBF approach dominates the HOSVD approach. This can be attributed to the local nature of the chaincode maps via Sobel gradient. While the HOSVD algorithm extracts an image–wide, global feature, the chaincode maps are localized to regions of the image, providing improved accuracy.

It is likely that the two classifiers can be placed in succession, rather than in fusion, to create a nested classifier that provides both high speed and high accuracy.

## References

- [Bel02] S. Belongie, J. Malik and J. Puzicha, Shape matching and object recognition using shape contexts. *IEEE Trans. Pattern Anal. Mach. Intell.* **24** 4 (2002), pp. 509–522.
- [Lec98] LeCun, Y. et al. “Gradient-Based Learning for Object Detection, Segmentation, and Recognition.”
- [Lec08] LeCun, Y. and Cortes, C. “The MNIST Database of Handwritten Digits.” <http://yann.lecun.com/exdb/mnist/>. Accessed March 2008.
- [Liu03] C.-L. Liu, K. Nakashima, H. Sako, H. Fujisawa, Handwritten digit recognition: benchmarking of state-of-the-art techniques, *Pattern Recognition*, 36(10): 2271-2285, 2003.
- [Mil05] J. Milgram, R. Sabourin, M. Cheriet, “Combining Model-based and Discriminative Approaches in a Modular Two-stage Classification System: Application to Isolated Handwritten Digit Recognition,” *Electronic Letters on Computer Vision and Image Analysis*, vol. 5, no. 2, pp. 1–15, 2005.
- [Sav07] Savas, B., and Elden, L. “Handwritten digit classification using higher order singular value decomposition.” *Pattern Recognition*, Vol 40, 2007. 993 - 1003.
- [Sch02] Schlesinger, M and Hlavac, V. *Ten lectures on statistical and structural pattern recognition*. Kluwer Academic Publishers, 2002.
- [Sim03] Simard, P.Y.; Steinkraus, D.; Platt, J.C., "Best practices for convolutional neural networks applied to visual document analysis," *Document Analysis and Recognition, 2003. Proceedings. Seventh International Conference on* , vol., no., pp. 958-963, 3-6 Aug. 2003
- [Wit05] Witten, I.H. and Frank, E. "Data Mining: Practical machine learning tools and techniques", 2nd Edition, Morgan Kaufmann, San Francisco, 2005.