# Chapter 10

# Classification of Handwritten Digits

Classification by computer of handwritten digits is a standard problem in pattern recognition. The typical application is automatic reading of zip codes on envelopes. A comprehensive review of different algorithms is given in [62].

## 10.1 Handwritten Digits and a Simple Algorithm

In Figure 10.1 we illustrate handwritten digits that we will use in the examples in this chapter.

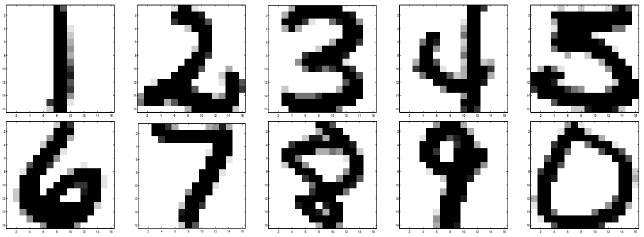


**Figure 10.1.** *Handwritten digits from the U.S. Postal Service database; see, e.g.,* [47].

We will treat the digits in three different but equivalent formats:

1. As $16 \times 16$ gray scale images, as in Figure 10.1;

2. As functions of two variables, $s = s(x, y)$, as in Figure 10.2; and

3. As vectors in $\mathbb{R}^{256}$.

In the classification of an unknown digit we need to compute the distance to known digits. Different distance measures can be used, and perhaps the most natural one to use is the Euclidean distance: stack the columns of the image in a

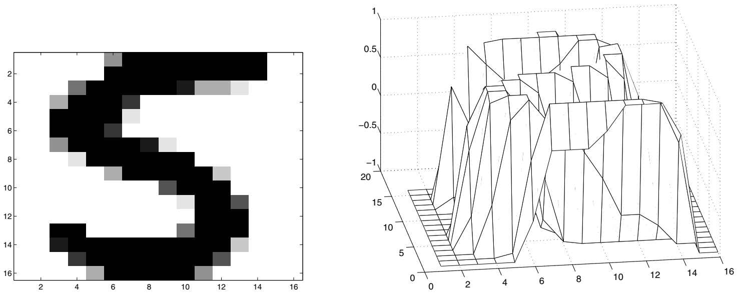


**Figure 10.2.** *A digit considered as a function of two variables.*

vector and identify each digit as a vector in $\mathbb{R}^{256}$. Then define the distance function

$$(x, y) = \| x - y \|_2.$$

An alternative distance function is the cosine between two vectors.

In a real application of handwritten digit classification, e.g., zip code reading, there are hardware and real-time factors that must be taken into account. In this chapter we describe an idealized setting. The problem is as follows:

> *Given a set of manually classified digits* (*the training set*), *classify a set of unknown digits* (*the test set*).

In the U.S. Postal Service database, the training set contains 7291 handwritten digits. Here we will use a subset of 1707 digits, relatively equally distributed between 0 and 9. The test set has 2007 digits.

If we consider the training set digits as vectors or points, then it is reasonable to assume that all digits of one kind form a cluster of points in a Euclidean 256-dimensional vector space. Ideally the clusters are well separated (otherwise the task of classifying unknown digits will be very difficult), and the separation between the clusters depends on how well written the training digits are.

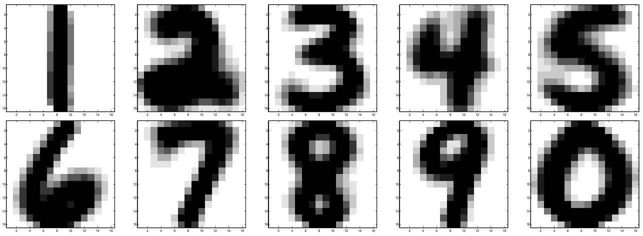


**Figure 10.3.** *The means* (*centroids*) *of all digits in the training set.*

In Figure 10.3 we illustrate the means (centroids) of the digits in the training set. From the figure we get the impression that a majority of the digits are well written. (If there were many badly written digits, the means would be very diffuse.)

This indicates that the clusters are rather well separated. Therefore, it seems likely that a simple classification algorithm that computes the distance from each unknown digit to the means may be reasonably accurate.

**A simple classification algorithm**

**Training:** Given the manually classified training set, compute the means (centroids) $m_i$, $i = 0, \ldots, 9$, of all the 10 classes.

**Classification:** For each digit in the test set, classify it as $k$ if $m_k$ is the closest mean.

It turns out that for our test set, the success rate of this algorithm is around 75%, which is not good enough. The reason for this relatively bad performance is that the algorithm does not use any information about the variation within each class of digits.

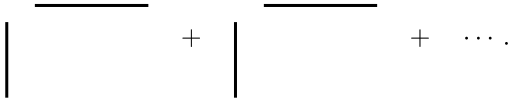
## 10.2 Classification Using SVD Bases

We will now describe a classification algorithm that is based on the modeling of the variation within each digit class using orthogonal basis vectors computed using the SVD. This can be seen as a least squares algorithm based on a *reduced rank model*; cf. Chapter 7.

If we consider the images as $16 \times 16$ matrices, then the data are multidimensional; see Figure 10.4. Stacking all the columns of each image above each other gives a matrix. Let $A \in \mathbb{R}^{m \times n}$, with $m = 256$, be the matrix consisting of all the training digits of one kind, the 3's, say. The columns of $A$ span a linear subspace of $\mathbb{R}^m$. However, this subspace cannot be expected to have a large dimension, because if it did, then the subspaces of the different kinds of digits would intersect (remember that we are considering subspaces of $\mathbb{R}^{256}$).

Now the idea is to "model" the variation within the set of training (and test) digits of one kind using an orthogonal basis of the subspace. An orthogonal basis can be computed using the SVD, and any matrix $A$ is a sum of rank 1 matrices:

$$A = \sum_{i=1}^{m} \sigma_i u_i v_i^T =$$



(10.1)

Each column in $A$ represents an image of a digit 3, and therefore the left singular vectors $u_i$ are an orthogonal basis in the "image space of 3's." We will refer to the left singular vectors as "singular images." From (10.1) the $j$th column of $A$ is equal to

$$a_j = \sum_{i=1}^{m} (\sigma_i v_{ij}) u_i,$$

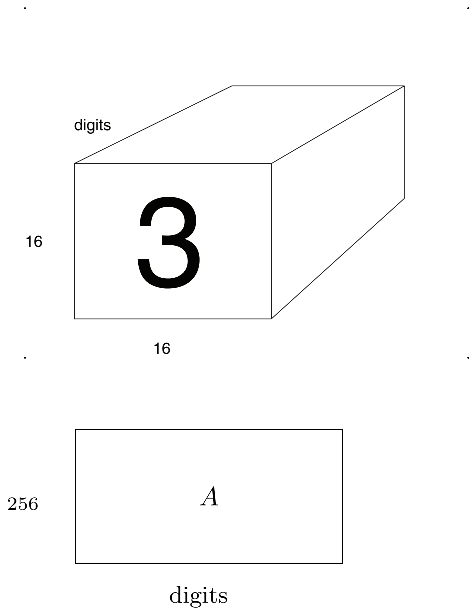


**Figure 10.4.** *The image of one digit is a matrix, and the set of images of one kind form a tensor. In the lower part of the figure, each digit (of one kind) is represented by a column in the matrix.*

and we see that the coordinates of image $j$ in $A$ in terms of this basis are $\sigma_i v_{ij}$. From the matrix approximation properties of the SVD (Theorems 6.6 and 6.7), we know that the first singular vector represents the "dominating" direction of the data matrix. Therefore, if we fold the vectors $u_i$ back into images, we expect the first singular vector to look like a 3, and the following singular images should represent the dominating variations of the training set around the first singular image. In Figure 10.5 we illustrate the singular values and the first three singular images for the training set 3's. In the middle graph we plot the coordinates of each of the 131 digits in terms of the first three singular vectors. We see that all the digits have a large portion (between 0.05 and 0.1) of the first singular image, which, in fact, looks very much like the mean of 3's in Figure 10.3. We then see that there is a rather large variation in the coordinates in terms of the second and third singular images.

The SVD basis classification algorithm will be based on the following assumptions:

1. Each digit (in the training set and the test set) is well characterized by a few of the first singular images of its own kind. The more precise meaning of "few" should be investigated in experiments.

2. An expansion in terms of the first few singular images discriminates well between the different classes of digits.

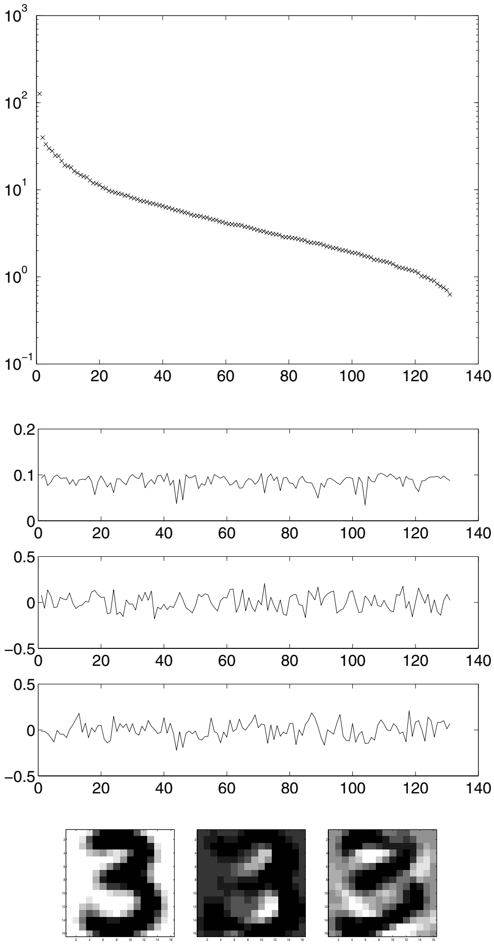


**Figure 10.5.** *Singular values* (*top*), *coordinates of the* 131 *test digits in terms of the first three right singular vectors* $v_i$ (*middle*), *and the first three singular images* (*bottom*).

3. If an unknown digit can be better approximated in one particular basis of singular images, the basis of 3's say, than in the bases of the other classes, then it is likely that the unknown digit is a 3.

Thus we should compute how well an unknown digit can be represented in the 10 different bases. This can be done by computing the residual vector in *least squares problems* of the type

$$\min_{\alpha_i} \left\| z - \sum_{i=1}^{k} \alpha_i u_i \right\|,$$

where $z$ represents an unknown digit and $u_i$ represents the singular images. We can

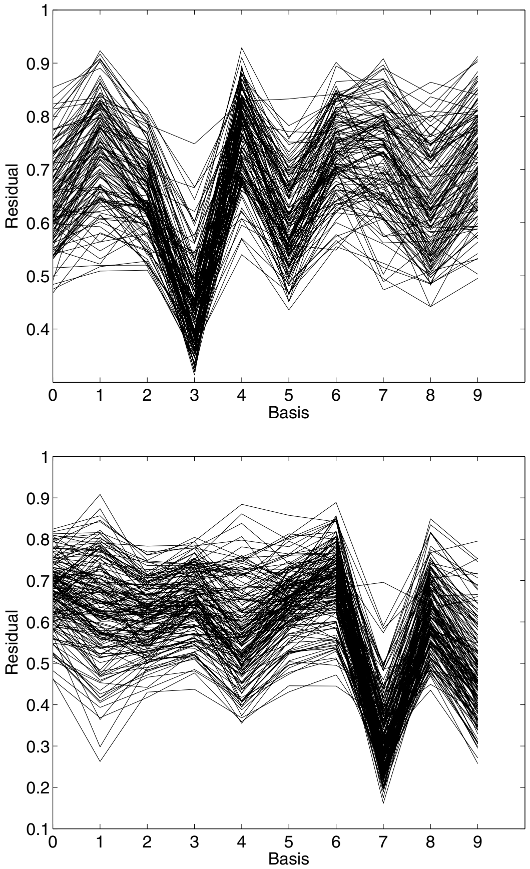


**Figure 10.6.** *Relative residuals of all test* 3*'s* (*top*) *and* 7*'s* (*bottom*) *in terms of all bases. Ten basis vectors were used for each class.*

write this problem in the form

$$\min_{\alpha} \| z - U_k \alpha \|_2,$$

where $U_k = \begin{pmatrix} u_1 & u_2 & \cdots & u_k \end{pmatrix}$. Since the columns of $U_k$ are orthogonal, the solution of this problem is given by $\alpha = U_k^T z$, and the norm of the residual vector of the least squares problems is

$$\| (I - U_k U_k^T) z \|_2, \tag{10.2}$$

i.e., the norm of the projection of the unknown digit onto the subspace orthogonal to span$(U_k)$.

To demonstrate that the assumptions above are reasonable, we illustrate in Figure 10.6 the relative residual norm for all test 3's and 7's in terms of all 10 bases.

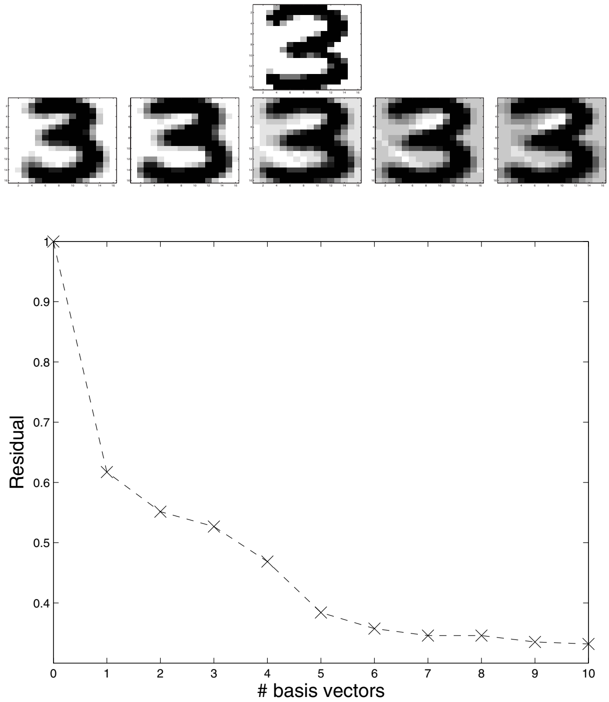


**Figure 10.7.** *Unknown digit* (*nice* 3) *and approximations using* 1*,* 3*,* 5*,* 7*, and* 9 *terms in the* 3*-basis* (*top*)*. Relative residual* $\| (I - U_k U_k^T) z \|_2 / \| z \|_2$ *in least squares problem* (*bottom*)*.*

In the two figures, there is one curve for each unknown digit, and naturally it is not possible to see the individual curves. However, one can see that most of the test 3's and 7's are best approximated in terms of their own basis. The graphs also give information about which classification errors are more likely than others. (For example, 3's and 5's are similar, whereas 3's and 4's are quite different; of course this only confirms what we already know.)

It is also interesting to see how the residual depends on the number of terms in the basis. In Figure 10.7 we illustrate the approximation of a nicely written 3 in terms of the 3-basis with different numbers of basis images. In Figures 10.8 and 10.9 we show the approximation of an ugly 3 in the 3-basis and a nice 3 in the 5-basis.

From Figures 10.7 and 10.9 we see that the relative residual is considerably smaller for the nice 3 in the 3-basis than in the 5-basis. We also see from Figure 10.8 that the ugly 3 is not well represented in terms of the 3-basis. Therefore, naturally, if the digits are very badly drawn, then we cannot expect to get a clear classification based on the SVD bases.

It is possible to devise several classification algorithms based on the model of expanding in terms of SVD bases. Below we give a simple variant.

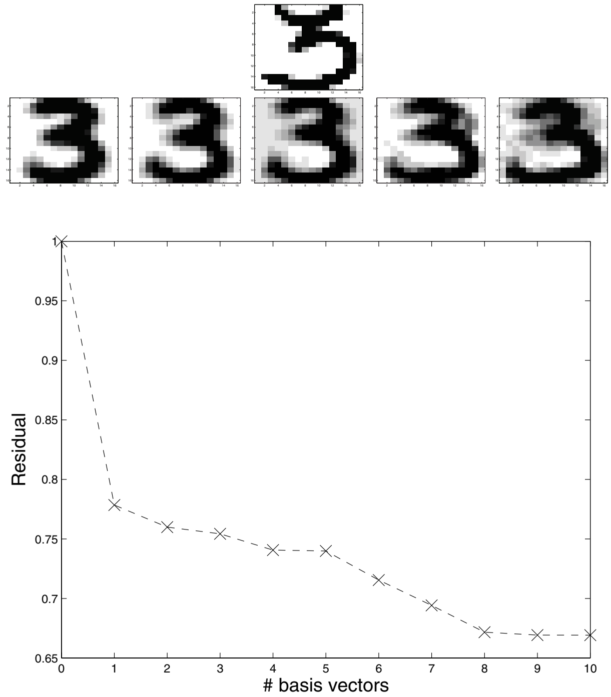


**Figure 10.8.** *Unknown digit* (*ugly* 3) *and approximations using* 1, 3, 5, 7, *and* 9 *terms in the* 3*-basis* (*top*). *Relative residual in least squares problem* (*bottom*).

### An SVD basis classification algorithm

**Training:** For the training set of known digits, compute the SVD of each set of digits of one kind.

**Classification:** For a given test digit, compute its relative residual in all 10 bases. If one residual is significantly smaller than all the others, classify as that. Otherwise give up.

The work in this algorithm can be summarized as follows:

**Training:** Compute SVDs of 10 matrices of dimension $m^2 \times n_i$.
Each digit is an $m \times m$ digitized image.
$n_i$: the number of training digits $i$.

**Test:** Compute 10 least squares residuals (10.2).

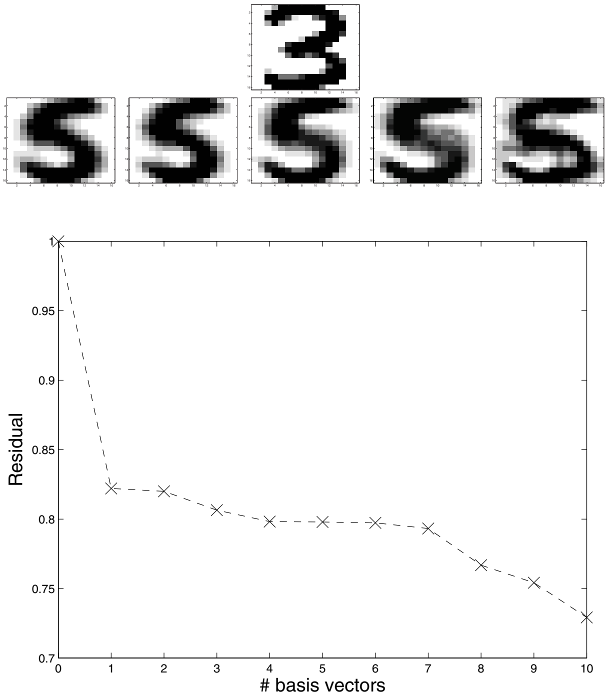


**Figure 10.9.** *Unknown digit (nice* 3*) and approximations using* 1*,* 3*,* 5*,* 7*, and* 9 *terms in the* 5*-basis (top). Relative residual in least squares problem (bottom).*

**Table 10.1.** *Correct classifications as a function of the number of basis images (for each class).*

| # basis images | 1 | 2 | 4 | 6 | 8 | 10 |
|---|---|---|---|---|---|---|
| correct (%) | 80 | 86 | 90 | 90.5 | 92 | 93 |

Thus the test phase is quite fast, and this algorithm should be suitable for real-time computations. The algorithm is related to the SIMCA method [89].

We next give some test results (from [82]) for the U.S. Postal Service database, here with 7291 training digits and 2007 test digits [47]. In Table 10.1 we give classification results as a function of the number of basis images for each class.

Even if there is a very significant improvement in performance compared to the method in which one used only the centroid images, the results are not good enough, as the best algorithms reach about 97% correct classifications. The training and test contain some digits that are very difficult to classify; we give a few examples in Figure 10.10. Such badly written digits are very difficult to handle automatically.

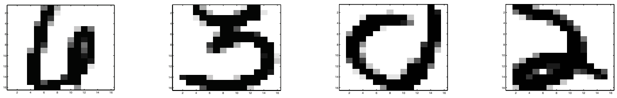


**Figure 10.10.** *Ugly digits in the U.S. Postal Service database.*

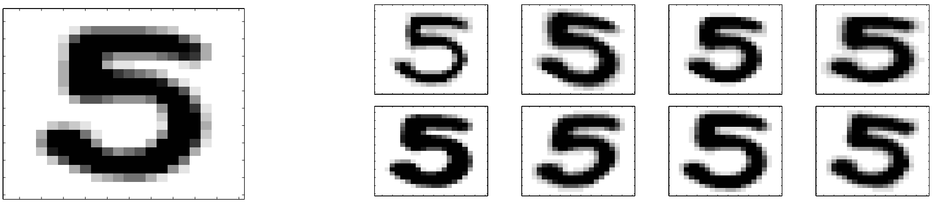


**Figure 10.11.** *A digit* (*left*) *and acceptable transformations* (*right*). *Columnwise from left to right the digit has been* (1) *written with a thinner and a thicker pen,* (2) *stretched diagonally,* (3) *compressed and elongated vertically, and* (4) *rotated.*

## 10.3 Tangent Distance

A good classification algorithm should be able to classify unknown digits that are rather well written but still deviate considerably in Euclidean distance from the ideal digit. There are some deviations that humans can easily handle and which are quite common and acceptable. We illustrate a few such variations[20] in Figure 10.11. Such transformations constitute no difficulties for a human reader, and ideally they should be very easy to deal with in automatic digit recognition. A distance measure, *tangent distance*, that is invariant under small such transformations is described in [86, 87].

$16 \times 16$ images can be interpreted as points in $\mathbb{R}^{256}$. Let $p$ be a fixed pattern in an image. We shall first consider the case of only one allowed transformation, translation of the pattern (digit) in the $x$-direction, say. This translation can be thought of as moving the pattern along a curve in $\mathbb{R}^{256}$. Let the curve be parameterized by a real parameter $\alpha$ so that the curve is given by $s(p,\alpha)$ and in such a way that $s(p,0)=p$. In general, the curve is nonlinear and can be approximated by the first two terms in the Taylor expansion,

$$s(p,\alpha) = s(p,0) + \frac{ds}{d\alpha}(p,0)\,\alpha + O(\alpha^2) \approx p + t_p\alpha,$$

where $t_p = \frac{ds}{d\alpha}(p,0)$ is a vector in $\mathbb{R}^{256}$. By varying $\alpha$ slightly around 0, we make a small movement of the pattern along the tangent at the point $p$ on the curve. Assume that we have another pattern $e$ that is approximated similarly:

$$s(e,\alpha) \approx e + t_e\alpha.$$

[20] Note that the transformed digits have been written not manually but by using the techniques described later in this section. The presentation in this section is based on the papers [86, 87, 82].

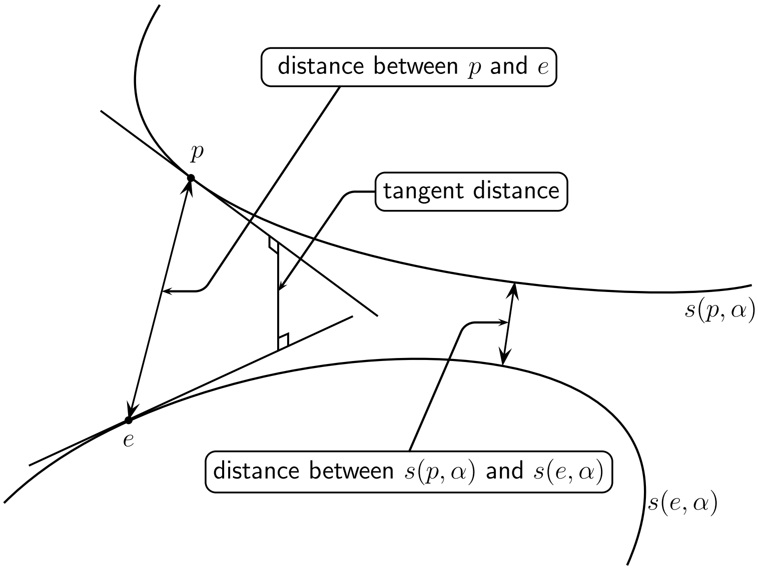

distance between $p$ and $e$
$p$
tangent distance
$s(p,\alpha)$
$e$
distance between $s(p,\alpha)$ and $s(e,\alpha)$
$s(e,\alpha)$

**Figure 10.12.** *The distance between the points p and e, and the tangent distance.*

Since we consider small movements along the curves as allowed, such small movements should not influence the distance function. Therefore, ideally we would like to define our measure of closeness between $p$ and $e$ as the closest distance between the two curves; see Figure 10.12 (cf. [87]).

However, since in general we cannot compute the distance between the curves, we can use the first order approximations and compute the closest distance between the two tangents in the points $p$ and $e$. Thus we shall move the patterns independently along their respective tangents, until we find the smallest distance. If we measure this distance in the usual Euclidean norm, we solve the least squares problem,

$$\min_{\alpha_p,\alpha_e} \left\| p + t_p\alpha_p - e - t_e\alpha_e \right\|_2 = \min_{\alpha_p,\alpha_e} \left\| (p-e) - \begin{pmatrix} -t_p & t_e \end{pmatrix} \begin{pmatrix} \alpha_p \\ \alpha_e \end{pmatrix} \right\|_2 .$$

Now consider the case when we are allowed to move the pattern $p$ along $l$ different curves in $\mathbb{R}^{256}$, parameterized by $\alpha = (\alpha_1 \cdots \alpha_l)^T$. This is equivalent to moving the pattern on an $l$-dimensional surface (manifold) in $\mathbb{R}^{256}$. Assume that we have two patterns, $p$ and $e$, each of which is allowed to move on its surface of allowed transformations. Ideally we would like to find the closest distance between the surfaces, but instead, since that is impossible to compute, we now define a distance measure, where we compute the distance between the two *tangent planes* of the surface in the points $p$ and $e$.

As before, the tangent plane is given by the first two terms in the Taylor expansion of the function $s(p, \alpha)$:

$$s(p, \alpha) = s(p, 0) + \sum_{i}^{l} \frac{ds}{d\alpha_i}(p, 0)\, \alpha_i + O(\| \alpha \|_2^2) \approx p + T_p \alpha,$$

where $T_p$ is the matrix

$$T_p = \begin{pmatrix} \dfrac{ds}{d\alpha_1} & \dfrac{ds}{d\alpha_2} & \cdots & \dfrac{ds}{d\alpha_l} \end{pmatrix},$$

and the derivatives are all evaluated in the point $(p, 0)$.

Thus the *tangent distance* between the points $p$ and $e$ is defined as the smallest possible residual in the least squares problem,

$$\min_{\alpha_p, \alpha_e} \| p + T_p \alpha_p - e - T_e \alpha_e \|_2 = \min_{\alpha_p, \alpha_e} \left\| (p - e) - \begin{pmatrix} -T_p & T_e \end{pmatrix} \begin{pmatrix} \alpha_p \\ \alpha_e \end{pmatrix} \right\|_2 .$$

The least squares problem can be solved, e.g., using the SVD of $A = \begin{pmatrix} -T_p & T_e \end{pmatrix}$. Note that we are interested not in the solution itself but only in the norm of the residual. Write the least squares problem in the form

$$\min_{\alpha} \| b - A\alpha \|_2, \qquad b = p - e, \quad \alpha = \begin{pmatrix} \alpha_p \\ \alpha_e \end{pmatrix}.$$

If we use the QR decomposition[21]

$$A = Q \begin{pmatrix} R \\ 0 \end{pmatrix} = (Q_1\, Q_2) \begin{pmatrix} R \\ 0 \end{pmatrix} = Q_1 R,$$

the norm of the residual is given by

$$\begin{aligned} \min_{\alpha} \| b - A\alpha \|_2^2 &= \min_{\alpha} \left\| \begin{pmatrix} Q_1^T b - R\alpha \\ Q_2^T b \end{pmatrix} \right\|^2 \\ &= \min_{\alpha} \left[ \| (Q_1^T b - R\alpha) \|_2^2 + \| Q_2^T b \|_2^2 \right] = \| Q_2^T b \|_2^2. \end{aligned}$$

The case when the matrix $A$ should happen to not have full column rank is easily dealt with using the SVD; see Section 6.7. The probability is high that the columns of the tangent matrix are almost linearly dependent when the two patterns are close.

The most important property of this distance function is that it is *invariant under movements of the patterns on the tangent planes.* For instance, if we make a small translation in the $x$-direction of a pattern, then with this measure, the distance it has been moved is equal to zero.

### 10.3.1 Transformations

Here we consider the image pattern as a function of two variables, $p = p(x, y)$, and we demonstrate that the derivative of each transformation can be expressed as a differentiation operator that is a linear combination of the derivatives $p_x = \frac{dp}{dx}$ and $p_y = \frac{dp}{dy}$.

[21] $A$ has dimension $256 \times 2l$; since the number of transformations is usually less than 10, the linear system is overdetermined.

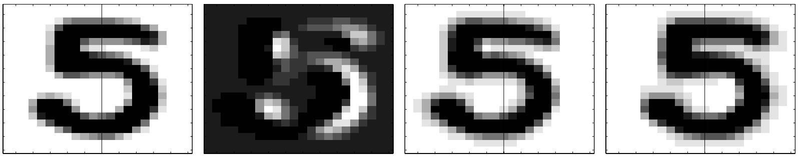


**Figure 10.13.** *A pattern, its $x$-derivative, and $x$-translations of the pattern.*

**Translation.** The simplest transformation is the one where the pattern is translated by $\alpha_x$ in the $x$-direction, i.e.,

$$s(p, \alpha_x)(x, y) = p(x + \alpha_x, y).$$

Obviously, using the chain rule,

$$\frac{d}{d\alpha_x} \left( s(p, \alpha_x)(x, y) \right) |_{\alpha_x = 0} = \frac{d}{d\alpha_x} p(x + \alpha_x, y)|_{\alpha_x = 0} = p_x(x, y).$$

In Figure 10.13 we give a pattern and its $x$-derivative. Then we demonstrate that by adding a small multiple of the derivative, the pattern can be translated to the left and to the right.

Analogously, for $y$-translation we get

$$\frac{d}{d\alpha_y} \left( s(p, \alpha_y)(x, y) \right) |_{\alpha_y = 0} = p_y(x, y).$$

**Rotation.** A rotation of the pattern by an angle $\alpha_r$ is made by replacing the value of $p$ in the point $(x, y)$ with the value in the point

$$\begin{pmatrix} \cos \alpha_r & \sin \alpha_r \\ -\sin \alpha_r & \cos \alpha_r \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}.$$

Thus we define the function

$$s(p, \alpha_r)(x, y) = p(x \cos \alpha_r + y \sin \alpha_r, -x \sin \alpha_r + y \cos \alpha_r),$$

and we get the derivative

$$\frac{d}{d\alpha_r} \left( s(p, \alpha_r)(x, y) \right) = (-x \sin \alpha_r + y \cos \alpha_r) p_x + (-x \cos \alpha_r - y \sin \alpha_r) p_y.$$

Setting $\alpha_r = 0$, we have

$$\frac{d}{d\alpha_r} \left( s(p, \alpha_r)(x, y) \right) |_{\alpha_r = 0} = y p_x - x p_y,$$

where the derivatives are evaluated at $(x, y)$.

An example of a rotation transformation is given in Figure 10.14.

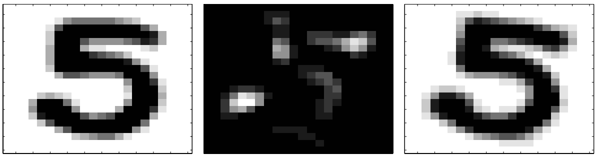


**Figure 10.14.** *A pattern, its rotational derivative, and a rotation of the pattern.*

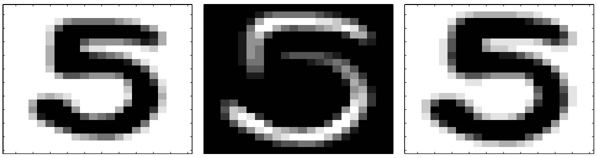


**Figure 10.15.** *A pattern, its scaling derivative, and an "up-scaling" of the pattern.*

**Scaling.** A scaling of the pattern is achieved by defining

$$s(p,\alpha_s)(x,y) = p((1+\alpha_s)x,(1+\alpha_s)y),$$

and we get the derivative

$$\frac{d}{d\alpha_s}\left(s(p,\alpha_s)(x,y)\right)|_{\alpha_s=0} = xp_x + yp_y.$$

The scaling transformation is illustrated in Figure 10.15.

**Parallel Hyperbolic Transformation.** By defining

$$s(p,\alpha_p)(x,y) = p((1+\alpha_p)x,(1-\alpha_p)y),$$

we can stretch the pattern parallel to the axis. The derivative is

$$\frac{d}{d\alpha_p}\left(s(p,\alpha_p)(x,y)\right)|_{\alpha_p=0} = xp_x - yp_y.$$

In Figure 10.16 we illustrate the parallel hyperbolic transformation.

**Diagonal Hyperbolic Transformation.** By defining

$$s(p,\alpha_h)(x,y) = p(x+\alpha_h y, y+\alpha_h x),$$

we can stretch the pattern along diagonals. The derivative is

$$\frac{d}{d\alpha_h}\left(s(p,\alpha_h)(x,y)\right)|_{\alpha_h=0} = yp_x + xp_y.$$

In Figure 10.17 we illustrate the diagonal hyperbolic transformation.

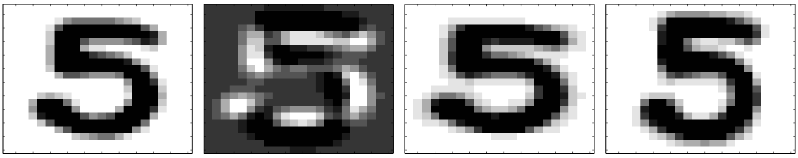


**Figure 10.16.** *A pattern, its parallel hyperbolic derivative, and two stretched patterns.*

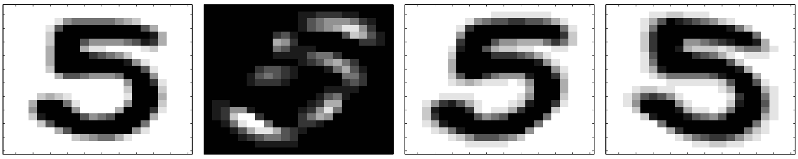


**Figure 10.17.** *A pattern, its diagonal hyperbolic derivative, and two stretched patterns.*

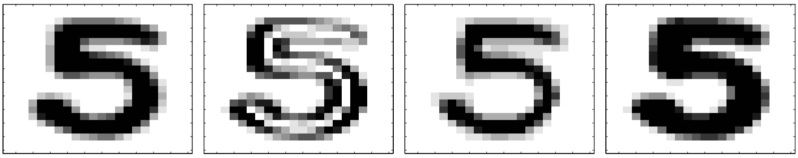


**Figure 10.18.** *A pattern, its thickening derivative, a thinner pattern, and a thicker pattern.*

**Thickening.** The pattern can be made thinner or thicker using similar techniques; for details, see [87]. The "thickening" derivative is

$$(p_x)^2 + (p_y)^2.$$

Thickening and thinning are illustrated in Figure 10.18.

### A tangent distance classification algorithm

**Training:** For each digit in the training set, compute its tangent matrix $T_p$.

**Classification:** For each test digit,

- compute its tangent matrix;
- compute the tangent distance to all training digits and classify the test digit as the closest training digit.

Although this algorithm is quite good in terms of classification performance (96.9% correct classification for the U.S. Postal Service database [82]), it is very expensive, since each test digit is compared to all the training digits. To be competitive, it must be combined with some other algorithm that reduces the number of tangent distance comparisons.

We end this chapter by remarking that it is necessary to preprocess the digits in different ways in order to enhance the classification; see [62]. For instance, performance is improved if the images are smoothed (convolved with a Gaussian kernel) [87]. In [82] the derivatives $p_x$ and $p_y$ are computed numerically by finite differences.