CSE 252C: Computer Vision III

Lecturer: Serge Belongie
Scribes: Andrew Rabinovich and Vincent Rabaud
Edited by: Catherine Wah

LECTURE 2
Image Representation and Distance Measurement

2.1. Images as Vectors

Representing images as vectors allows one to perform various mathematical operations on the image data. In its simplest form, transforming an image into a vector may be performed by simply stacking all the columns on the matrix I (original image). This is illustrated in Figure 1: a 28×28 grayscale image, in this case an MNIST digit [1] represented as a 2-D pixel array $I(x, y)$, is concatenated into a column vector $\boldsymbol{x} \in \mathbb{R}^d$, where $d = 784$. In this vector form, we can do all the things one usually does with vectors, such as averaging (mean), calculating the covariance matrix, clustering, manifold analysis, and so on. This technique of transforming a full image can be seen in applications such as the eigenfaces used in face recognition.

[1]Department of Computer Science and Engineering, University of California, San Diego.

July 16, 2009

[1]http://yann.lecun.com/exdb/mnist

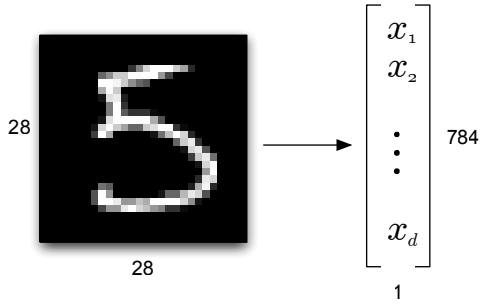

Figure 1. Transformation from an image (left) to a column vector (right).

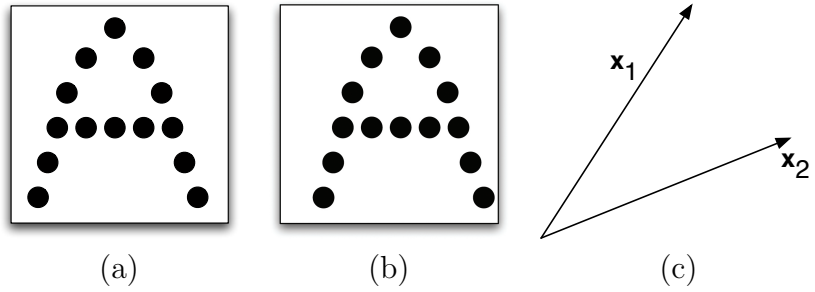

Figure 2. (a) Original image. (b) Slightly translated image with respect to (a). (c) Difference in vectors $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ corresponding to images in (a) and (b), depicted in a d-dimensional space (where d is the length of $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$) with both vectors starting at the origin.

Much like how an entire image can be transformed into a vector, image regions or patches can similarly be transformed into vectors. A third representation involves vectors of filter responses. [2]

Although vectorial representations of images are crucial to image processing and computer vision, there are some drawbacks, namely that the spatial organization of the image disappears in the vector format. As an example, consider an image in Figure 2(a). Translating the letter "A" by one pixel in (x, y) space to Figure 2(b) results in unintuitive leaps in $\mathbb{R}^d$, shown in Figure 2(c). This issue provides motivation for using filter responses or extracting patches around interest points; we will be covering these topics in later lectures.

[2]It is worth noting that the transformation of images into vectors in image processing and computer vision is different than "vectorization" in computer graphics. "Vectorization" refers to parametric modeling of curves in the image for loss-less zooming and object manipulations.

Effects of 2-D transformations in $\mathbb{R}^d$ are humbling enough; it is even heavier to consider that the 2-D images are projections of 3-D objects – consider out-of-plane rotations, for example. Nevertheless, the image-as-vector representation is widely used and very powerful in many applications. We follow by reviewing some relevant properties from vector spaces.

2.2. Vector Norms and Measurements of Distance

Norms do for a vector space what the absolute value does for the one-dimensional number line: they provide a measure of distance: $\mathbb{R}^n$+ 'a norm' = metric space (*e.g.* 3-D Euclidean space).

Definition 2.1. The vector norm is a function $f : \mathbb{R}^n \to \mathbb{R}$ satisfying:

- $f(\boldsymbol{x}) \geq 0, \forall \boldsymbol{x} \in \mathbb{R}^n$
- $f(\boldsymbol{x}) = 0 \Leftrightarrow \boldsymbol{x} = 0$
- $f(\boldsymbol{x} + \boldsymbol{y}) \leq f(\boldsymbol{x}) + f(\boldsymbol{y}), \forall (\boldsymbol{x},\boldsymbol{y}) \in \mathbb{R}^n$
- $f(\alpha \boldsymbol{x}) = |\alpha| f(\boldsymbol{x}), \forall (\alpha, \boldsymbol{x}) \in \mathbb{R} \times \mathbb{R}^n$

and we notate the norm as: $f(\boldsymbol{x}) = \|\boldsymbol{x}\|$.

2.2.1. The L_p-norm, for $p = \{1, 2, \infty\}$

A widely used class of norms is the p-norm or L_p-norm:

$$\|\boldsymbol{x}\|_p = (|x_1| + \cdots + |x_n|)^{1/p},\ p \geq 1 \tag{2.2}$$

Most commonly, we see $p = \{1, 2, \infty\}$:

- $\|\boldsymbol{x}\|_1 = |x_1| + \cdots + |x_n|$, the L_1 or the Manhattan norm.
- $\|\boldsymbol{x}\|_2 = \sqrt{|x_1|^2 + \cdots + |x_n|^2}$, the L_2 or Euclidean norm.
- $\|\boldsymbol{x}\|_\infty = \max_{1 \leq i \leq n} |x_i|$, the L_∞-norm.

Note that the unit vector for any norm $\|\cdot\|$ satisfies $\|\boldsymbol{x}\| = 1$. For intuition, we can visualize what the *unit circle* looks like for the above norms, as shown in Figure 3.

L_p-norms are defined for $p \geq 1$. However, there is a notion of 0-norm, yet it is not a norm in the strict sense, but simply a count of non-zero entries in the vector.

2.2.2. Norm Properties and Relationships

The Cauchy-Schwartz inequality is probably familiar:

$$|\boldsymbol{x}^\top \boldsymbol{y}| \leq \|\boldsymbol{x}\|_2 \|\boldsymbol{y}\|_2 \tag{2.3}$$

(This is a special case of the Hölder inequality: $|\boldsymbol{x}^\top \boldsymbol{y}| \leq \|\boldsymbol{x}\|_p \|\boldsymbol{y}\|_p$ for $\frac{1}{p} + \frac{1}{q} = 1$.) Recall for intuition that in $\mathbb{R}^2$ we have

$$|\boldsymbol{x}^\top \boldsymbol{y}| = \|\boldsymbol{x}\| \|\boldsymbol{y}\| \cdot \cos(\theta) \leq \|\boldsymbol{x}\| \|\boldsymbol{y}\|. \tag{2.4}$$

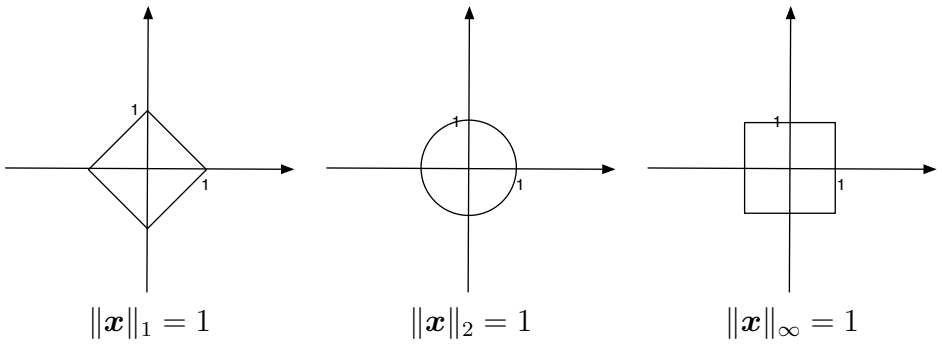

Figure 3. Unit "circle" representation for L_1, L_2, and L_∞ norms.

We also formalize the concept of norm *equivalence*.

Definition 2.5. If we can find positive constants c_1 and c_2 such that

$$c_1\|\boldsymbol{x}\|_\alpha \leq \|\boldsymbol{x}\|_\beta \leq c_2\|\boldsymbol{x}\|_\alpha, \, \forall \|\boldsymbol{x}\|_2 \in \mathbb{R}^n,$$

we say the norms $\|\cdot\|_\alpha$ and $\|\cdot\|_\beta$ are *equivalent*.

Example 1. Here are a few equivalences between norms:

- $\|\boldsymbol{x}\|_2 \leq \|\boldsymbol{x}\|_1 \leq \sqrt{n}\|\boldsymbol{x}\|_2$
- $\|\boldsymbol{x}\|_\infty \leq \|\boldsymbol{x}\|_2 \leq \sqrt{n}\|\boldsymbol{x}\|_\infty$
- $\|\boldsymbol{x}\|_\infty \leq \|\boldsymbol{x}\|_1 \leq n\|\boldsymbol{x}\|_\infty$

Hence, L_1, L_2, and L_∞ norms are equivalent. Formally, equivalence does not mean that these norms are the same; this is evident from Figure 3.

2.2.3. Definitions of Errors

Here we'll define the error measurement using the L_2-norm and the distance between vectors. We define

$$\epsilon_{abs} = \|\boldsymbol{x} - \boldsymbol{y}\|_2^2 = (\boldsymbol{x} - \boldsymbol{y})^\top(\boldsymbol{x} - \boldsymbol{y}) = \|\boldsymbol{x}\|_2^2 + \|\boldsymbol{y}\|_2^2 - 2\boldsymbol{x}^\top\boldsymbol{y} \tag{2.6}$$

where ϵ_{abs} is the absolute error, also called the sum of squared errors (SSE) in image processing, or the sum of squared distances (SSD):

$$\boldsymbol{e} = \boldsymbol{x} - \boldsymbol{y}, \, \|\boldsymbol{e}\|_2^2 = \boldsymbol{e}^\top\boldsymbol{e} = \sum_{i=1}^{n} e_i^2 = \sum_i (x_i - y_i)^2. \tag{2.7}$$

Recall that $\boldsymbol{x}$ and $\boldsymbol{y}$ could be entire images or patches. A historically relevant approach to recognition is to compare images, or patches, to a predefined template, or in particular measuring the distance between the two. An important case of error evaluation is *cross correlation*, where $\boldsymbol{x}$ is a "template" and $\boldsymbol{y}$ is an image patch (extracted by sliding a window over an

image). Consider the template and an image in Figure 4. By sliding a window across this image, we want to measure the error between the template and the given patch. Cross correlation between these vectorized patches is the cross term of $\|\boldsymbol{x}\|_2^2 + \|\boldsymbol{y}\|_2^2 - 2\boldsymbol{x}^\top\boldsymbol{y}$, which is just as informative as the SSD if $\|\boldsymbol{y}\|_2$ is approximately constant (clearly $\|\boldsymbol{x}\|_2$ is always constant).

In practice, cross correlation has some shortcomings. For example, consider the effects of brightness and contrast variations of the image. Let images I_1 and I_2 be related as follows in a simple linear case of a brightness transform: $I_2(x, y) = \alpha I_1(x, y) + \beta$, where α and β are contrast scale factor and brightness offset respectively. Perceptually, α and β do not have a dramatic effect on human recognition ability, but computationally, cross correlation completely misinterprets these variations.

To avoid such behavior of cross correlation, normalized cross correlation (NCC) has been adopted. Let

$$\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i, \tag{2.8}$$

and define $\tilde{\boldsymbol{x}} = \boldsymbol{x} - \bar{x}\mathbf{1}$ and $\tilde{\boldsymbol{y}} = \boldsymbol{y} - \bar{y}\mathbf{1}$, where $\mathbf{1}$ is a column vector of ones. Then,

$$NCC(\boldsymbol{x}, \boldsymbol{y}) = \frac{\tilde{\boldsymbol{x}}^\top\tilde{\boldsymbol{y}}}{\|\tilde{\boldsymbol{x}}\|_2\|\tilde{\boldsymbol{y}}\|_2}; \tag{2.9}$$

by subtracting out the mean we address the brightness offset and by dividing by the product of the norms we address contrast differences.[3] Note that NCC is not a norm, it is a function of two vectors: it uses a norm in the denominator. NCC, as humble and simple as it is, is widely used (the Matlab implementation *normxcorr2.m* is very fast; it uses FFTs and the integral image trick).

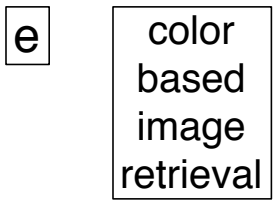

Figure 4. Example template (left) and test image (right).

[3]It is important to note that NCC is undefined in constant regions of the image, as the denominator goes to zero. In later lectures we will address this problem by only considering non-constant regions using interest point detectors.

2.3. Applications

Measuring distances between vectors is commonly used in many areas of computer vision. In this class we will pay close attention to two of them: *distance based classification* and *clustering*. Classification and clustering, instances of supervised and unsupervised learning, respectively, rely on computing distances between vectors, whether it be all pairs of vectors (clustering) or finding the smallest distance between a query vector and a vector in the training set (classification).

2.3.1. Classification

Minimum-distance based classification is both simple and effective; however, this brute force approach can be expensive and memory intensive. One example of this type of classification is *k-nearest neighbor* (k-NN), a form of instance-based learning, also called "lazy learning" (versus "eager learning," which attempts to do some kind of generalization in the training stage, or before testing data is even considered).

Choosing an appropriate k is a case of model-order selection, and one possible heuristic for this task is cross validation, which involves partitioning the data into subsets for training and testing, and repeating the validation multiple times for different partitions.

Classification is an instance – probably one of the simplest – of supervised learning; we can also use distances between vectors in an unsupervised fashion, for example, using a clustering algorithm.

2.3.2. Clustering

Clustering involves partitioning a dataset (a collection of vectors, in this case) into subsets or *clusters* so that the data in each subset share a common trait, e.g. proximity. A sizeable area of research in machine learning, clustering was recently re-energized by the emergence of kernel-based methods in the mid-1990s.

Qualitatively, in this course, we will encounter two motivations for clustering:

(1) data-compression-style (for instance, vector quantization), where we do not impute deep meaning to the resulting clusters; and
(2) prototype-seeking, where we aspire to discover intuitively meaningful structures.

In (1), the data could be spread uniformly and we would still get a benefit, as when computing a compact palette for a color image. For (2), we presuppose that the data exhibits a "clumpy" structure (kernels generalize the notion of clumpiness, but that's for another time).

Examples of clustering algorithms include *k-means* (Lloyd-Max, 1982), which is implemented as *kmeans.m* in Matlab; and *mean-shift* (Fukunaga and Hostetler, 1975). Both of these are iterative and need a magic parameter (either k or kernel width). For k-means, we start by picking k centroids and they migrate; in the other case (mean-shift), every point starts as a cluster center, and they all migrate to the modes, hopefully coalescing.