Introduction to Supervised Learning

Erik G. Learned-Miller
Department of Computer Science
University of Massachusetts, Amherst
Amherst, MA 01003

February 17, 2014

Abstract

This document introduces the paradigm of supervised learning. It also discusses nearest neighbor classification and the distance functions necessary for nearest neighbor classification. It discusses Euclidean distance functions in two and three dimensions, and their extensions to higher dimensions, including distance functions between binary images, scalar-valued images, and 3-channel color images.

1 Supervised learning

supervised learning
training set
test set

Supervised learning is simply a formalization of the idea of learning from examples. In supervised learning, the learner (typically, a computer program) is provided with two sets of data, a *training set* and a *test set*. The idea is for the learner to "learn" from a set of labeled examples in the training set so that it can identify unlabeled examples in the test set with the highest possible accuracy. That is, the goal of the learner is to develop a rule, a program, or a procedure that classifies new examples (in the test set) by analyzing examples it has been given that already have a class label. For example, a training set might consist of images of different types of fruit (say, peaches and nectarines), where the identity of the fruit in each image is given to the learner. The test set would then consist of more unidentified pieces of fruit, but from the same classes. The goal is for the learner to develop a rule that can identify the elements in the test set. There are many different approaches that attempt to build the best possible method of classifying examples of the test set by using the data given in the training set. We will discuss a few of these in this document, after defining supervised learning more formally.

In supervised learning, the training set consists of n ordered pairs $(\mathbf{x}_1, y_1)$, $(\mathbf{x}_2, y_2), ..., (\mathbf{x}_n, y_n)$, where each $\mathbf{x}_i$ is some measurement or set of measurements of a single example data point, and y_i is the label for that data point. For example, an $\mathbf{x}_i$ might be a group (sometimes called a vector[1]) of five measurements for a patient in a hospital including height, weight, temperature, blood sugar level, and blood pressure. The corresponding y_i might be a classification of the patient as "healthy" or "not healthy".

The test data in supervised learning is another set of m measurements *without* labels: $(\mathbf{x}_{n+1}, \mathbf{x}_{n+2}, ..., \mathbf{x}_{n+m})$. As described above, the goal is to make educated guesses about the labels for the test set (such as "healthy" or "not healthy") by drawing inferences from the training set.

An example we will consider frequently in this book is when each $\mathbf{x}_i$ is an image and the y_i gives the class of the image. Suppose we are writing software to recognize types of fruit at a supermarket checkout counter so that the price could be entered automatically. To train such a classifier, we would provide sample images (a training set) for each type of fruit, as shown in Figure 1. Then we would use the classifier by having it label new images of fruit (a test set).

1.1 Getting started with supervised learning: Nearest neighbor algorithms

To get a feel for supervised learning, we will start by exploring one of the simplest algorithms that uses training data to help classify test data, the nearest neighbor rule or nearest neighbor algorithm.

[1] A single group of measurements $\mathbf{x}_i$ is often referred to as a **vector** of measurements, although a better term would probably be **tuple**, since the term vector implies certain properties that may not be true for a set of measurements.

Figure 1: Training and test data for a fruit classifier. On the left are training data for the class of apples. In the middle are trainin data for the peaches class. On the right are a mixture of fruit which could be used to test the classifier.

1.1.1 Nearest neighbor classification

The *nearest neighbor* algorithm uses a simple intuitiion to classify test examples. Suppose that we have some way of computing a notion of distance between two observations $\mathbf{x}_1$ and $\mathbf{x}_2$:

$$D(\mathbf{x}_1, \mathbf{x}_2). \tag{1}$$

We will talk about various options for the function D below, but for now you can think of it as a function which returns 0 when the data items $\mathbf{x}_1$ and $\mathbf{x}_2$ are exactly the same, and returns larger numbers as the two items become "more different".

Given a test data item $\mathbf{x}_t$ the nearest neighbor classifier chooses the class corresponding to the data item in the training set with the lowest distance to the test example.

Using the notation we established above, let i^*, the index of the training example closest (i.e., with minimum distance) to the test example $\mathbf{x}_t$, be defined as

$$i^* = \arg\min_{i \in \{1 \ldots n\}} D(\mathbf{x}_t, \mathbf{x}_i).$$

After finding i^*, the nearest neighbor rule would assign to the test example the label y_{i^*}, the label of the training example $\mathbf{x}_{i^*}$ that was closest to the test example $\mathbf{x}_t$. Of course, this assignment of a label is merely a guess, and may be correct or incorrect.

1.1.2 Distance functions

To use the nearest neighbor classifier, we need to define the distance function D in Equation (1). In order to understand what a distance between images might look like, we start with a simpler notion of distance, the *Euclidean distance* in two dimensions. Let's suppose we have two points $\mathbf{p}$ and $\mathbf{q}$ with

Euclidean distance

$$\mathbf{p} = \begin{bmatrix} p_1 \\ p_2 \end{bmatrix}$$

and

$$\mathbf{q} = \begin{bmatrix} q_1 \\ q_2 \end{bmatrix}.$$

p_1, p_2, q_1, and q_2 are just the coordinates of $\mathbf{p}$ and $\mathbf{q}$. These coordinates could represent spatial measurements along axes in a Euclidean plane, but they could also represent other measurements such as the height and weight of a patient. In general, we shall consider each coordinate to represent a particular measurement in a given experiment, and not restrict them to be spatial measurements.

Whatever measurements these coordinates represent, the Euclidean distance between $\mathbf{p}$ and $\mathbf{q}$ is given by

$$D_{\texttt{Euc}}(\mathbf{p}, \mathbf{q}) = \sqrt{(p_1 - q_1)^2 + (p_2 - q_2)^2}.$$

Of course extending this formula to three spatial dimensions or three measurements is easy. If $\mathbf{p}$ and $\mathbf{q}$ consist of three measurements instead of two, we have

$$D_{\texttt{Euc}}(\mathbf{p}, \mathbf{q}) = \sqrt{(p_1 - q_1)^2 + (p_2 - q_2)^2 + (p_3 - q_3)^2}.$$

While it is harder to interpret using familiar 3-dimensional geometry, there is no reason we cannot extend this formula to $\mathbf{p}$ and $\mathbf{q}$ having four or more measurements. In general, when $\mathbf{p}$ and $\mathbf{q}$ consist of n measurements, we can define their $n-$dimensional Euclidean distance to be

$$D_{\texttt{Euc}}(\mathbf{p}, \mathbf{q}) = \sqrt{(p_1 - q_1)^2 + (p_2 - q_2)^2 + ... + (p_n - q_n)^2} \quad (2)$$

$$\sqrt{\sum_{i=1}^{n} (p_i - q_i)^2}. \quad (3)$$

Despite applying this formula to more than the usual three dimensions, such a formula continues to capture many of the underlying intuitions about what a distance function should capture:

- It is zero if and only if all the measurents for each point are the same.
- It obeys the triangle inequality. One consequence of this is that for points $\mathbf{p}$, $\mathbf{q}$ and $\mathbf{r}$, we have that $D(\mathbf{p}, \mathbf{q}) + D(\mathbf{q}, \mathbf{r}) \geq D(\mathbf{p}, \mathbf{r})$. A more intuitive way to say this is that an intermediate stop along a path cannot make your trip shorter than it would be without the stop.

Now, we consider the Euclidean distance as a way to compare images.

1.1.3 Euclidean distance for binary images

To apply the Euclidean distance to a pair of images, we simply treat each image as a collection of measurements (the pixel values), and apply the Euclidean distance formula of Equation (3). That is, we consider the first pixel in each image to be the first measurement, the second pixel to be the second measurement, and so on.

It is worth taking a few moments to think about how to interpret such a function for images. Let's start with the case of binary images, i.e., images in which each pixel just takes on a value of either 0 or 1. In this case, note that each of the terms under the square root in Equation (3) will take on a 0 or 1, since the squared difference between two binary pixels is either 0 or 1. In particular, the terms will be 1 if the corresponding pixels are different, and 0 if they are the same. Thus, the total sum under the square root will be equal to the number of pixels that are different between the two images. This is a satisfying result, since counting the number of pixels that are different between two binary images seems like at least a reasonable starting point for defining a distance between the images.

Note that for the purposes of finding the *nearest neighbor*, the Euclidean distance is equivalent to the squared Euclidean distance since

$$D(\mathbf{p}, \mathbf{q}) > D(\mathbf{p}, \mathbf{r})$$

if and only if

$$D(\mathbf{p}, \mathbf{q})^2 > D(\mathbf{p}, \mathbf{r})^2.$$

Many nearest neighbor implementations take advantage of this fact to avoid computing the square root in Equation (3).

1.1.4 Euclidean distance for scalar-valued images and color images

If images take on continuous values between 0 and 1, or between say, 0 and 255 for 8-bit images, the principles of Euclidean distance are the same, and we can still use the formula in Equation (3) with one term per pixel. With color images that are represented as RGB images, we now need a mechanism for determining the difference between two pixels, since they are no longer scalar values.

Let p^r, p^g, and p^b be the red, green, and blue color channel values of a pixel p, and similarly for a pixel q. Then we can also use the Euclidean distance to define the distance between two color pixel values as

$$D(p, q) = \sqrt{(p^r - q^r)^2 + (p^g - q^g)^2 + (p^b - q^b)^2}.$$

If we take this distance between color pixels as a pixel "difference" and plug it into the general formula for Euclidean distance between images (Equation (3)), we obtain

$$D_{\texttt{Euc}}(\mathbf{p}, \mathbf{q}) = \sqrt{\sum_{i=1}^{n} (p_i - q_i)^2} \tag{4}$$

$$= \sqrt{\sum_{i=1}^{n} (p_i^r - q_i^r)^2 + (p_i^g - q_i^g)^2 + (p_i^b - q_i^b)^2}. \tag{5}$$

Note that the expression on the right size of Equation (5) is just another Euclidean distance in which each component of each pixel has been treated as an individual measurement.