

Pattern Recognition: Week 2

Distance functions and neighbours

John Quinn

February 29, 2008

Distances

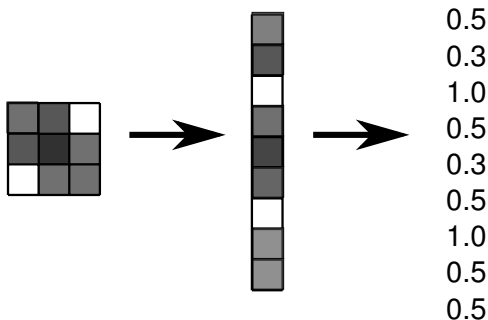
- The idea of *distance* (or equivalently *similarity*) is fundamental to pattern recognition.
- If we are trying to recognise printed characters, for example, then we need some way of specifying that:

A is close to A but far away from B

- We will start by looking at the most common distance measure for data, *Euclidean distance*.
- This will lead to our first pattern recognition tool, *nearest neighbour classification*.

Representing different types of data as points in space

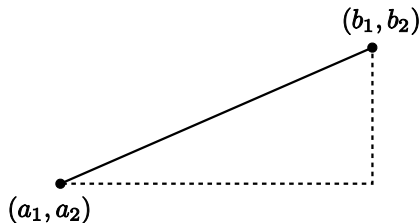
- A pair of measurements (height, weight) can be taken as a single point in a 2D space (as in the snarks and boojums example from the last class).
- Most types of data can be represented as a point. For example, this image with 9 pixels can be manipulated as follows:



to give a 9-dimensional vector.

Euclidean distance

- The most common measure of the distance between two points:



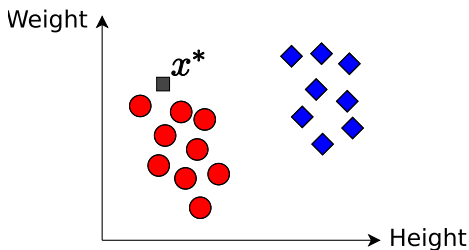
$$d = \sqrt{(b_1 - a_1)^2 + (b_2 - a_2)^2} \quad (1)$$

- We often use the squared Euclidean distance as well.
- It can be extended to any number of dimensions.
- In vector notation, the squared distance is

$$d^2 = (\mathbf{b} - \mathbf{a})^\top (\mathbf{b} - \mathbf{a}). \quad (2)$$

How to do classification

- Return to the problem of classifying different types of animals.
- We have a collection of 2D vectors $\mathbf{x}_{1:N}$. We also have a collection of class labels $c_{1:N}$ telling us which type of animal each vector is.



- What algorithm would you suggest using to classify the new point \mathbf{x}^* ?

Nearest neighbour algorithm

Algorithm 1: NN algorithm

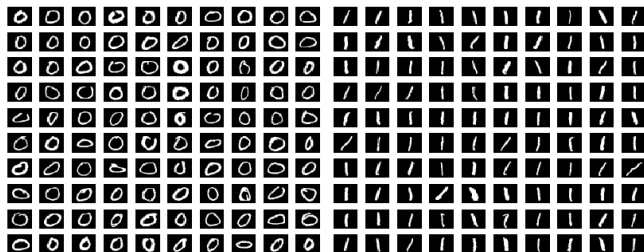
Data: Training examples $\{\mathbf{x}_{1:N}, c_{1:N}\}$, Test point \mathbf{x}^* .

Result: Class of new point c^* .

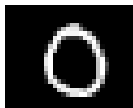
- 1 **for** $i = 1$ **to** N **do**
 - 2 calculate distance $d(\mathbf{x}_i, \mathbf{x}^*)$;
 - 3 **end**
 - 4 find \mathbf{x}_j , for which the distance was smallest;
 - 5 $c^* = c_j$;
-

Application to digit recognition

- We can easily apply this to digit recognition.
- Start by taking training images for each class:

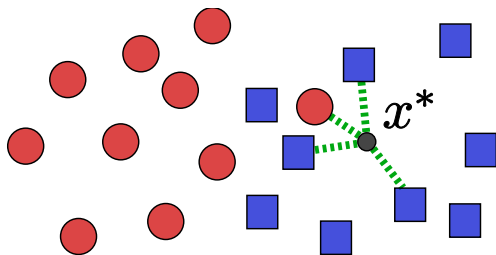


- Then, for a new image \mathbf{x}^* :



we can work out which is the nearest neighbour in the training set.

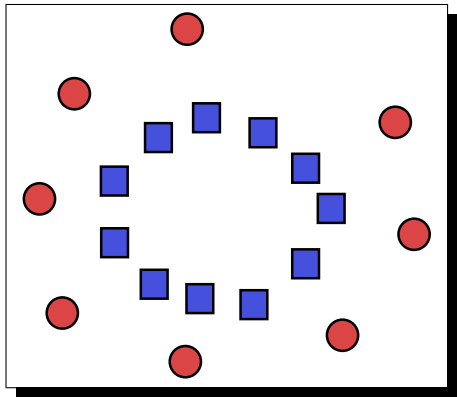
K-nearest neighbours



- The algorithm can be made more robust by taking the k nearest neighbours and selecting the most common class (the training examples vote on the class).

Problems with Euclidean distance

- Here is a case where nearest neighbour and Euclidean distance might cause problems.



- Can you think of other scenarios where this strategy would not work?

Mahalanobis distance

- This is another distance measure which takes into account the variation in the measurements.

$$d_M^2 = (\mathbf{a} - \mathbf{b})^\top \Sigma_i^{-1} (\mathbf{a} - \mathbf{b}) \quad (3)$$

- Σ_i is the covariance of the data from class i .