

Pattern Recognition: Week 4

Principal components

John Quinn

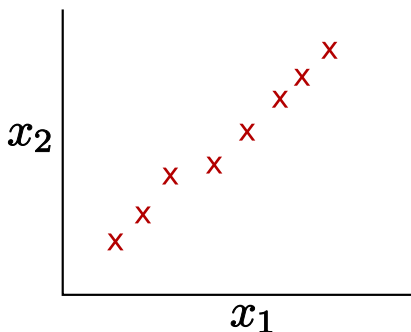
March 14, 2008

Dimensionality reduction

- A lot of interesting data has very high dimensionality. For example, even the digits that we were processing have several hundred dimensions (pixels).
- Pattern recognition is easier if we can find a way to reduce the dimensionality of the data.
- We will begin by looking at one common technique: principal components analysis.

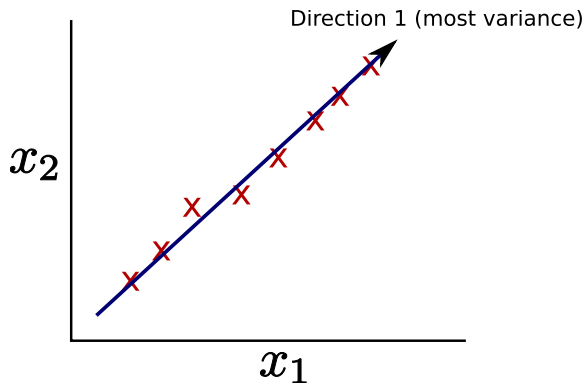
Two dimensions to one dimension

- Each of these data points needs two coordinates to specify its position: (x_1, x_2) .
- What system could you use to specify the positions approximately using only one number?



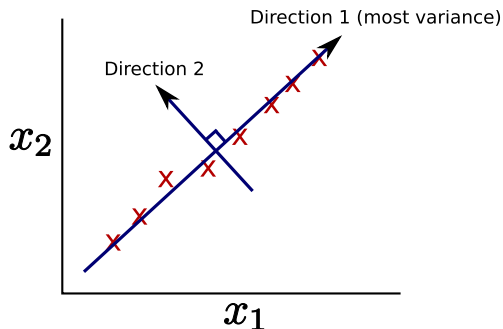
The first principal component

- One way of doing this is to find the mean of the data, and the first principal component (the direction of maximum variance).
- We can then specify how far along the line each point is, giving its approximate position.



The second principal component

- If we use another direction at right angles, then we can specify exactly where each point is.
- Starting at the mean, we say how far to go in direction 1, then how far to go in direction 2.



- Notice that we have effectively replaced the x_1 and x_2 axes with something more “natural” to the data.

Calculating the principal components

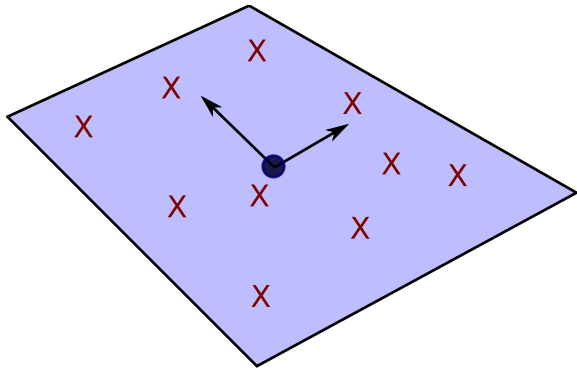
How to calculate the principal components:

- 1 Calculate the mean of the data (`mean()` in numpy).
- 2 Calculate the covariance of the data (`C=cov()` in numpy).
- 3 Calculate the eigenvectors of the data (`eig(C)` in numpy).
- 4 Keep the first few, and throw the rest away.

How many principal components should you keep, and how many should you throw away?

3D to 2D

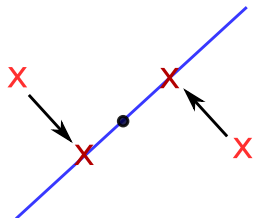
- From three dimensional data, we can use two dimensions to get a more compact representation:



- But remember that this only works if the data is approximately linear in a low dimension.

Calculating the low-D representation

- We need to *project* each point onto the different “axes” (principal components).



- If \mathbf{e}_1 is the first eigenvector, \mathbf{x} is the datapoint and \mathbf{m} is the mean, then the projection is given by:

$$\mathbf{a} = \mathbf{e}_1^\top (\mathbf{x} - \mathbf{m}) \quad (1)$$

which is just the dot product of two vectors (the `dot()` function from the last practical).

Low-D representation of faces

- We can use this technique to reduce the dimensionality of face images. The principal components/eigenvectors are sometimes called “eigenfaces”.

