27 -> 32

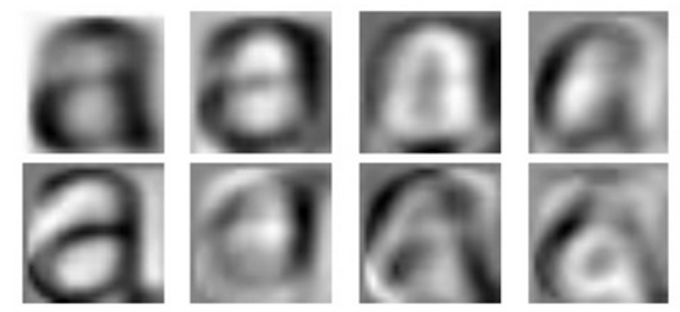
Hàm này trước tính trung tâm của dữ liệu bằng cách trị giá trị trung bình theo tổng chùm cầu. Sau đó, vector riêng tổng hợp với các giá trị riêng lớn nhất của ma trận hiệp phương sai được tính toán bằng cách sử dụng phương pháp SVD. Ở đây, đã sử dụng hàm range() để lấy nguyên n và trả về dãy số nguyên 0... (n - 1).

Ở đây, một mẹo nhỏ được dùng thay vì dùng phương pháp SVD nguyên gốc để tính toán giá trị vector riêng của ma trận hiệp phương sai nếu số lượng điểm dữ liệu nhỏ hơn kích thước của vector. Cũng có những cách nhanh hơn khi chỉ tính toán để vector riêng tổng hợp với giá trị riêng lớn nhất k (k là số lượng kích thước mong muốn). Các hằng của ma trận V và trục giao và chữa các hướng toa độ theo tỷ phương sai giảm dần của dữ liệu huấn luyện.

Chúng ta có thể thể hiện phương pháp này trên một ví dụ về hình ảnh fontchit. Tệp fontimages.zip chứa hình ảnh thu nhận của ký tự "a" trong các fontchữ khác nhau và sau đó được quét. Các font chữ 2359 là một bộ sưu tập các font chữ có sẵn. Giá trị tên tệp của những hình ảnh này được lưu trữ trong một danh sách, imlist, cùng với chương trình trước đó, trong tệp **pca.py**, các thành phần chính có thể được tính toán và hiển thị như sau:

|  |
| --- |
| from PIL import Image from numpy import \* from pylab import \* import pca  im = array(Image.open(imlist[0])) *# open one image to get size* m, n = im.shape[:2] *# get the size of the images* imnbr = len(imlist) *# get the number of images  # create matrix to store all flattened images* immatrix = array([array(Image.open(im)).flatten()  for im in imlist], 'f')  *# perform PCA* V, S, immean = pca.pca(immatrix)  *# show some images (mean and 7 first modes)* figure() gray() subplot(2, 4, 1) imshow(immean.reshape(m, n))  for i in range(7):  subplot(2, 4, i\*2)  imshow(V[i].reshape(m, n))  show() |

Lưu ý rằng hình ảnh cần phải được chuyển đổi trở lại từ hình ảnh một chiều bằng cách sử dụng hàm reshape(). Khi chạy ví dụ sẽ cho ta các hình ảnh trong một cửa sổ hình giống như trong Hình 1.8. Ở đây, đã sử dụng hàm subplot() để đặt nhiều ô trong một cửa sổ.



*Hình 1.8: Hình ảnh trung bình (trên cùng bên trái) và bảy chế độ đầu tiên, tức là các hướng có nhiều biến thể nhất.*

1.3.6. Sử dụng Module Pickle

Nếu muốn lưu một số kết quả hoặc dữ liệu để sử dụng sau này, sử dụng module pickle rất hữu ích. Pickle có thể lấy hầu hết mọi đối tượng Python và chuyển đổi nó thành định dạng chuỗi. Quá trình này được cấu trúc để dữ liệu được gọi là giải nén. Lưu trữ dạng này có thể để dàng lưu trữ hoặc truyền gửi đi.

Có thể minh họa điều này bằng một ví dụ như sau. Giả sử muốn lưu hình ảnh trung bình và các thành phần chính của hình ảnh fontchữ trong phần trước, tiến hành thay hình các dòng lệnh như sau:

|  |
| --- |
| *# save mean and principal components* f = open("font\_pca\_modes.pkl", 'wb') pickle.dump(immean, f) pickle.dump(V, f) f.close() |

Có thể thấy rằng một số đối tượng có thể được chọn vào cùng một tệp. Thông thường có một số giao thức khác nhau có sẵn cho các tệp .pkl nhưng nếu không chắc chắn tốt nhất là đọc và ghi tệp nhị phân. Để tải dữ liệu trong một số phiên bản Python khác, chỉ cần sử dụng lệnh load() như sau:

|  |
| --- |
| *# Tải lại mô hình* f = open("font\_pca\_modes.pkl", 'rb') immean = pickle.load(f) V = pickle.load(f) f.close() *# Áp dụng mô hình lên dữ liệu mới # ... (các phép tính sử dụng immean và V)* |

Lưu ý rằng thứ tự của các đối tượng nên giống nhau. Ngoài ra còn có một phiên bản tối ưu hóa được viết bằng C gọi là cpickle hoàn toàn tương thích với mô hình pickle tiêu chuẩn. Thông tin chi tiết trên trang tài liệu về mô hình pickle tham khảo tại https://docs.python.org/3/ library/pickle.html.

Trong nội dung cuốn sách này, sẽ sử dụng câu lệnh with để xử lý việc đọc và ghi tập. Đây là cấu trúc được giới thiệu trong Python 2.5, tự động xử lý việc mở và đóng tệp (ngay cả khi xảy ra lỗi trong khi tập đang mở). Đây là những gì việc lưu và tải ở trên trông giống như sử dụng lệnh with().

|  |
| --- |
| *# open file and save* with open("font.pca.modes.pkl", 'wb') as f:  pickle.dump(immean, f)  pickle.dump(V,f) |

|  |
| --- |
| *# open file and load* with open("font.pca.modes.pkl", 'rb') as f:  immean = pickle.load(f)  V = pickle.load(f) |

Cấu trúc này rất hữu ích tuy nhiên nếu không muốn sử dụng chỉ cần sử dụng các hàm mở và đóng như trên.

Thay thế cho việc sử dụng pickle, NumPy cũng có các chức năng đơn giản để đọc và ghi tệp văn bản. Ngoài ra còn có những lợi ích khác nếu dữ liệu không chứa các cấu trúc phức tạp, ví dụ: danh sách các điểm được chọn trước trên mộtdữ liệu hình ảnh. Để lưu một mảng x vào tập tin có thể sử dụng dòng lệnh sau:

|  |
| --- |
| savetxt('test.txt',x, 'i') |

Tham số cuối cùng chỉ ra rằng định dạng số nguyên nên được sử dụng. Tương tự, việc đọc được thực hiện như sau:

|  |
| --- |
| x = loadtxt('test.txt') |

Ngoài ra, có thể tìm hiểu thêm từ tài liệu trực tuyến: http://docs.scipy.org/doc/numpy/reference/generated/numpyloadtxt.html.

Cuối cùng, NumPy có các hàm chuyên dụng để lưu và tải mảng. Tìm kiếm save() và load() trong tài liệu trực tuyến để biết chi tiết hơn

**1.4. SCIPY**

SciPy (http://scipy.org/) là gói thư viện nguồn mở dành cho toán học xây dựng trên NumPy và cung cấp các các toán tử cần thiết bao gồm phép tính tích phân, tối ưu hóa, thống kê, xử lý tín hiệu và quan trọng nhất là xử lý hình ảnh. Những phần tiếp theo sau đây sẽ trình bày nhiều module hữu ích trong thư viện SciPy. Thư viện SciPy có sẵn tại tranghttp://scipy.org/ Download. Một số tập lệnh và ứng dụng trong thư viện Scipy được bình bày sau đây.

**1.4.1. Làm mờ ảnh**

Một ví dụ kinh điển và rất hữu ích về tích chập hình ảnh là làm mờ hình ảnh dùng mặt nạ Gaussian. Về bản chất, hình ảnh (thang độ xám) 1 tích chập với mặt nạ Gaussian để tạo ra một hình ảnh được làm mờ.

(1.1)

Trong đó \* biểu thị tích chập và   là mặt nạ 2D Gaussian với độ lệch

chuẩn được định nghĩa là:

(1.2)

Việc làm mờ ảnh dùng mặt nạ Gaussian được sử dụng để định nghĩa không gian tỷ lệ hình ảnh dùng cho các phép nội suy, tính toán các điểm đặc trưng và trong nhiều ứng dụng khác

SciPy đi kèm với một module để lọc ảnh được gọi là scipy. ndimage.filters có thể được sử dụng để tính toán phép tích chập bằng cách sử dụng phân tách nhanh một chiều. Đoạn chương trình mẫu được mô tả bên dưới:

|  |
| --- |
| from PIL import Image from numpy import array from scipy.ndimage import filters  im = array(Image.open('empire.jpg').convert('L')) im2 = filters.gaussian\_filter(im, 5) |

Ở đây, tham số cuối cùng của gaussian filter () là độ lệch chuẩn



*Hình 1.9: Một ví dụ về làm mờ Gaussian bằng cách sử dụng mô hình scipy.ndimage.filters. (a) ảnh gốc trong thang độ xám, (b) bộ lọc*

*Gaussian với 6 = 2, (c) với σ = 5, (d) với 6 = 10*

Hình 1.9 cho thấy các ví dụ về hình ảnh bị mờ khi tăng dần độ giá trị ơ. Giá trị lớn hơn của ở cung cấp ít chi tiết hơn. Để làm mờ hình ảnh màu, chỉ cần áp dụng phương pháp tích chập dùng Gaussian cho mỗi kênh màu riêng biệt.

|  |
| --- |
| im = array(Image.open('empire.jpg')) im2 = zeros(im.shape) for i in range(3):  im2[:,:,i] = filters.gaussian\_filter(im[:,:,i], 5) im2 = uint8(im2) |

Ở đây, việc chuyển đổi trong dòng lệnh cuối cùng dùng để biến đổi ảnh thành kiểu biến “uint8” không phải lúc nào cũng cần thiết ngoài việc đôi khi phải thay đổi các giá trị pixel ở dạng số nguyên không âm 8 bit. Có thể sử dụng lệnh sau để chuyển đổi kiểu biến ảnh.

|  |
| --- |
| in2 = array(im2, 'uint8") |

Để biết thêm về cách sử dụng module này và các lựa chọn tham số khác nhau, có thể xem tài liệu về SciPy của scipy. ndimage tại:

http://docs.scipy.org/doc/scipy/reference/ndimage.html.

**1.4.2. Tính đạo hàm của ảnh (Image derivatives)**

Giá trị cường độ pixel trên ảnh thay đổi như thế nào là thông tin quan trọng, được sử dụng cho nhiều ứng dụng như sẽ thấy trong suốt cuốn sách này. Sự thay đổi cường độ được mô tả với các đạo hàm theo trục x và y là , và , của ảnh màu xám 1 (đối với ảnh màu, các đạo hàm thường được lấy cho mỗi kênh màu).

Vectơ gradient của ảnh Gradient có hai thuộc tính quan trọng đó là độ lớn vectơ gradient

Và góc của vecto gradient

cho biết hướng thay đổi cường độ lớn nhất tại mỗi điểm (pixel) trong ảnh. Hàm NumPy arctan2() trả về tín hiệu góc bằng radiant, trong khoảng π... π

Việc tính toán các đạo hàm hình ảnh có thể được thực hiện bằng cách sử dụng các phép toán xấp xỉ rời rạc. Điều này có thể dễ dàng thực hiện như là phép tích chập như sau:

Hai lựa chọn phổ biến cho Dx và Dy là các bộ lọc Prewitt:

Và các bộ lọc Sobel:

Các bộ lọc đạo hàm này rất dễ thực hiện bằng cách sử dụng tích chập tiêu chuẩn có sẵn trong mô hình scipy.ndimage.filters. Ví dụ:

|  |
| --- |
| from PIL import Image from numpy import \* from scipy.ndimage import filters  im = array(Image.open('empire.jpg').convert('L'))  *# Sobel derivative filters* imx = zeros(im.shape) filters.sobel(im, 1, imx)  imy = zeros(im.shape) filters.sobel(im, 0, imy)  magnitude = sqrt(imx\*\*2 + imy\*\*2) |