Lọc ảnh:

Phép tích chập

Là công thức tính .... .. .. ....

Lọc ảnh:

Là phép biến đổi ảnh bằng cách thay đổi các giá trị điểm ảnh của một ảnh ban đầu dựa trên một hàm biến đổi thao tác lân cận của các điểm ảnh. Mục tiêu của lọc ảnh là khử nhiễu hoặc giảm nhiễu

Các phép lọc ảnh sử dụng một ma trận gọi là nhân lọc (kernel) áp dụng tại tất cả các điểm ảnh 1 cách riêng rẽ.

Khi áp dụng tại 1 điểm ảnh, các điểm lân cận nằm trong nhân với điểm đang xét ở vị trí trong tâm (neo – anchor)

Tiếp theo, thực hiện tính toán dựa trên giá trị của các điểm nằm trong nhân (của ảnh gốc) với các giá trị của nhân lọc để đưa ra kết quả của phép lọc tại vị trí điểm ảnh đang áp dụng.

* Lọc trung vị

Phép lọc thay thế mỗi điểm ảnh bằng một giá trị medin của các điểm trong lận cận(cửa sổ) có vị trí trung tâm là điểm đang xét

* Loại bỏ nhiễu biến đổi đột ngột
* Phép lọc Gaussian:

Là phép lọc làm mượt ảnh sử dụng kernel filter với các trọng số phân bổ của các điểm lân cận dựa trên khoảng cách của nó tới điểm trung tâm

Trọng số Guassian:

Mã giả

function createGaussianKernel(size, sigma):

kernel = new matrix(size, size)

center = size / 2

for i in 0 to size-1:

for j in 0 to size-1:

x = i - center

y = j - center

kernel[i, j] = exp(-(x^2 + y^2) / (2 \* sigma^2)) / (2 \* pi \* sigma^2)

normalize kernel values to sum to 1

return kernel

function applyGaussianFilter(image, kernel):

filteredImage = new image with same dimensions as image

for each pixel in image:

sum = 0

for i in 0 to kernelSize-1:

for j in 0 to kernelSize-1:

neighborX = pixelX + i - kernelSize/2

neighborY = pixelY + j - kernelSize/2

if neighborX and neighborY are within image boundaries:

neighborValue = image[neighborX, neighborY]

weight = kernel[i, j]

sum += neighborValue \* weight

filteredImage[pixelX, pixelY] = sum

return filteredImage

* Loại bỏ tần số cao
* Nhân chập

Là phép toán học sử dụng trong xử lý ảnh, áp dụng để thực hiện các phép lọc và trích xuất đặc trưng

(lưu ý vì nhân chập có giá trị âm hoặc > 256) thì phải làm gì đây taaa => chuẩn hóa trong khoảng giá trị (0 – 255)

* ứng dụng:
* lọc ảnh trong miền không gian
* trích xuất đặc trưng

Phép lọc trung bình:

Phân lớp và Bài toán Phân lớp:

Phân lớp (classification): thao tác gán nhãn cho các dữ liệu input dựa trên đánh giá về độ tương đồng so sánh với dữ liệu chuẩn (có nhãn).

* Bài toán Phân lớp (Classification Problem): Là một dạng bài toán trong máy học, trong đó mô hình được dùng dể đào tạo để dự đoán lớp hoặc nhãn của một đối tượng dữ liệu mới dựa trên dữ liệu huấn luyện có sẵn.

Trong bài toán nhận dạng/phân lớp dữ liệu được chia thành 3 tập:

* 1. tập tham chiếu (reference set) chứa các dữ liệu có nhãn chính xác (with annotation information).
* 2. tập dữ liệu học (training set) chứa các dữ liệu (có thể có nhãn hoặc không) để hệ thống học và đưa ra mô hình dự đoán. Nếu thuật toán học đòi hỏi dữ liệu có nhãn thì gọi là học có giám sát (supervised learning) ngược lại là học không có giám sát (unsupervised learning).
* 3. tập dữ liệu kiểm thử (test/validation set): chứa các dữ liệu dùng để đánh giá độ chính xác của hệ thống. Có nhiều cách đánh giá khác nhau về độ chính xác của hệ thống, phụ thuộc vào bài toán và mục tiêu.

Phương Pháp Phân lớp kNN (k-Nearest Neighbors):

- Trong phương pháp này, có N lớp dữ liệu, mỗi lớp sẽ có ni phần tử, mỗi phần tử được biểu diễn bằng một vector đa chiều đặc trưng (có hang tram đến hang ngàn giá trị) được tính toán bằng cách sử dụng một thuật toán trích chọn đặc trưng và một nhân. Tập dữ liệu được chia thành 2 tập con là tập huấn luyện và tập kiểm thử

- Một số k lẻ và một hàm khoảng cách sẽ được lựa chọn

- Khi cần phân lớp/dự đoán/nhận dạng 1 mẫu (chưa có nhãn), kNN sẽ đi tìm k mẫu trong tập huấn luyện (tham chiếu) có độ tương đồng lớn nhất với mẫu cần nhận dạng bằng cách sử dụng hàm khoảng cách

- Nếu có ít nhất (+1) mẫu trong số k mẫu tương đồng nhất cùng thuộc về 1 lớp thì gán nhãn của các mẫu đó cho mẫu cần nhận dạng

- kNN là một phương pháp phân lớp đơn giản, không đòi hỏi phải có bước học và hiệu quả trong đa số các bài toán, tuy nhiên độ chính xác phụ thuộc nhiều vào hàm khoảng cách

- KNN được ứng dụng trong nhiều lĩnh vực khác nhau: Phân lớp hoa Iris, Bài toán Content Retrieval dựa trên NN, Biểu diễn Gene, ....

Trong phương pháp kNN nếu không nói gì thì k = 1.

Các giá trị k thường sử dụng là 1, 3, 5.

SVM

* Phương Pháp Phân lớp SVM (Support Vector Machine):

SVM là một thuật toán học có giám sát sử dụng cho các việc phân loại hoặc đệ quy

* Các Bước:

Đối với bài toán SVM, thường sẽ có 3 bước:

Trong phương pháp này có N lớp dữ liệu, tuy nhiên bai toán SVM chỉ phân loại nhiều nhất là 2 lớp đờ 2 mặt phẳng đó là H1 và H2 mà thôi(tìm hiểu vè cái chỗ này thêm nhé),

* Trích chọn đặc điểm để biểu diễn mỗi đối tượng thành một điểm trong không gian đặc điểm(feature space) với số chiều lớn. Sau bước này ta có mỗi mẫu là một điểm trong không gian R^m nào đó. Ở bước này đòi hỏi phải có một thuật toán trích chọn đặc điểm
* Học để tìm phân lớp tối ưu: thuật toán SVM sẽ được sử dụng để tìm ra siêu phân lớp tối ưu

+ Tìm siêu phẳng H: wT .x+b=0 và hai siêu phẳng H1:w T .x+b=+1, H2:w T .x+b=-1 song song với H sao cho khoảng cách giữa H1 và H2 đạt cực đại. Kết quả của việc giải bài toán sẽ thu được những mẫu nằm trên H1 và H2

+ Tiếp theo, ta cần tạo ra một hàm mục tiêu để tối ưu hóa. Hàm mục tiêu này cần đảm bảo rằng siêu phẳng phân cách các lớp dữ liệu có khoảng cách lớn nhất với các điểm dữ liệu nằm ở biên

+ Bài toán tối ưu để tìm siêu phẳng lớn nhất là một bài toán lồi, có thể được giải bằng các phương pháp tối ưu lồi tiêu chuẩn(vd: Phương pháp nhân tử Lagrange)

+ Sau khi giải được bài toán tối ưu, ta có thể tính toán siêu phẳng bằng cách sử dụng các vector trọng số và giá trị chặn đã được tìm được

* Phân lớp cho 1 mẫu mới bằng cách sử dụng siêu phẳng đã tìm ra

Neural Network

* Lớp Đầu Vào (Input Layer):

Chứa các nơ-ron tương ứng với số lượng đặc trưng của dữ liệu đầu vào

* Lớp Ẩn (Hidden Layer):

Có thể có nhiều lớp ẩn, số lượng nơ-ron và lớp ẩn được quyết định dựa trên độ phức tạp của mô hình và bài toán cụ thể.

Hàm kích hoạt thường được áp dụng ở đây để giúp mô hình học được các đặc trưng phi tuyến tính.

* Lớp Đầu Ra (Output Layer):

Chứa các nơ-ron tương ứng với số lượng lớp đầu ra mong muốn.

* Hàm Kích Hoạt (Activation Function):

Các nơ-ron trong lớp ẩn và đầu ra thường sử dụng hàm kích hoạt để đưa ra quyết định về việc nơ-ron nên được kích hoạt hay không.

* Trọng Số (Weights):

Tham số điều chỉnh tầm quan trọng của đặc trưng

* Tham Số Điều Chỉnh (Biases):

Tham số giúp điều chỉnh mức độ kích hoạt của nơ-ron dựa trên các tín hiệu đầu vào

* Tổng Số Tham Số:

Tổng số tham số của mạng nơ-ron là tổng số trọng số(weight) và số tham số điều chỉnh(bias) trên tất cả các lớp.

Số tham số mạng nơ ron:

Số lượng weight(trọng số) giữa lớp input với lớp ẩn

Số lượng weigth giữa lớp ẩn thứ 0 đến N

Số lượng weight giữa lớp ẩn tới lớp output

Số lượng bias của lớp ẩn

Số lượng bias của lớp output

Vd:

số lớp weight lớp input và lớp ẩn 2 \* 4 = 8

số lớp weight ẩn và lớp output: 4 \* 2 = 8

số lượng bias của lớp ẩn 1: 4

số lượng bias cuả lớp out: 2

* Số tham số = 8 + 8 + 4 + 2
* 22

Trích chọn đặc trưng và thuật toán trích chọn đặc trưng:

- Trích chọn đặc trưng (feature extraction): là phương pháp trích rút các đặc điểm có tính chất phân biệt nhất của ảnh hoặc đối tượng cần xử lý và biểu diễn dưới dạng số - 1 vector đa chiều. Các đặc trưng phân biệt chỉ là 1 phần thông tin của đối tượng hay dữ liệu phù hợp nhất với bài toán cần giải quyết. Đặc trưng phân biệt theo 2 khía cạnh: giúp phân biệt các ảnh thuộc về các đối tượng khác nhau, phân biệt được các bức ảnh khác nhau của cùng 1 đối tượng

Thuật toán trích chọn đặc trưng (feature selection) là một kỹ thuật tiền xử lý dữ liệu trong học máy, nhằm mục đích giảm số lượng đặc trưng (feature) trong một tập dữ liệu, đồng thời vẫn giữ lại thông tin quan trọng giúp mô hình học máy đạt được hiệu quả tốt.

- Thuật toán trích chọn đặc trưng

LBP

Định nghĩa:

Local Binary Pattern (LBP) là một phương pháp mô tả đặc trưng dựa trên các biến đổi cục bộ của các điểm ảnh trong hình.

Hoạt Động:

Mỗi điểm ảnh trong hình ảnh được so sánh với các điểm xung quanh của nó, và dựa vào sự so sánh, một chuỗi nhị phân được tạo ra.

Chuỗi nhị phân này được chuyển đổi thành một giá trị thập phân để tạo ra một vector đặc trưng.

Thuật toán LBP:

+ Chọn một điểm trung tâm:

Xác định giá trị trung tâm, so sánh Nếu giá trị pixel xung quanh lớn hơn hoặc bằng giá trị của pixel trung tâm, đặt bit = 1; ngược lại, đặt = 0.

+ Xây dựng mã LBP

Sắp xếp các bit thu được từ bước trước theo chiều kim đồng hồ để tạo ra một số nguyên nhị phân

Lặp lại quá trình trên cho tất cả các pixel trong hình ảnh.

+ Tính toán histogram LBP: Đếm tần suất xuất hiện của các giá trị LBP để tạo thành histogram, mô tả đặc trưng của toàn bộ hình ảnh.

Thuật toán:

Input: Ảnh xám, bán kính(r), kích thước kernel(cửa sổ trượt)

Output: Vector histogram LBP có kích thước M\*N\*59

Ứng dụng của LBP: Nhận diện khuôn mặt, Nhận dạng đối tượng và đối tượng

* SIFT:

Là phương pháp trích chọn đặc trưng để giải quyết bài toán đối sánh ảnh, tìm các keypoint ở ảnh mà luôn:

* Bất biến với tỉ lệ
* Bất biến với các hướng khác nhau
* Bất biến với các điều kiện ánh sáng
* Bất biến với khung nhìn

Thuật toán:

Bước 1: xây dựng không gian tỷ lệ (scale space) :

Trong đó là tỷ lệ

Bước 2: Tính DOG, với mỗi scale sẽ được một DOG pyramid

Bước 3: Xác định DOG Extrema bằng cách duyệt ảnh DoG:

+ Xét tất cả các điểm lận cận (cả scale)

+ Xét các giá trị cực đại và cực tiểu

Bước 4: Áp dụng công thức khai triển chuỗi Taylor để xác định sub-pixel

Bước 5: Loại bỏ các điểm tương phản không tốt

Bước 6: Tính gradient cho mỗi ảnh mờ để gán hướng:

+ Với mỗi vùng xung quanh keypoint:

* Tạo histogram gồm 36 hướng khác nhua
* Đánh trọng số mỗi điểm với 1 Guassian kernel có cửa sổ là 1.5
* Tạo keypoint cho các đỉnh có giá trị >= 0.8 giá trị tấn sồ lớn nhất của histogram

Bước 7: Xây dựng đặc trung SIFT:

Tìm ảnh mờ của scale gần nhất

Lấy mẫu các point xung qunh keypoint

Xoay các gradient và tọa độ bởi các hướng đã tính

Chia các vùng thành các vùng con

Tính histogram mỗi vùng con

* Ưu điểm: nhanh, hiểu quả với ảnh dù ảnh có nhiều nhiễu
* Nhược điêm:

Thay đổi cường độ sáng đột ngột

Gabor Wavelet:

Công thức:

Các bước:

Với một tín hiệu 2 chiều hoặc ảnh I, bộ lọc Gabor trong miền không gian (úng với mỗi giá trị scale và orientation cụ thể), biến đổi Gabor của I có thể nhận được bằng cách:

+ Thực hiện nhân chạp trong miền không gian: GT(I) = conv2(I, )

+ Thực hiện phép nhân từng phần tử trong miền tần số: GT(I) = ifft2(fft2(I) \* fft2()) với ifft và fft là các biến đổi Fourier ngược và xuôi còn \* là phép nhân từng phần tử của hai ma trận

* Một hình ảnh biểu diễn bởi biến đổi Gabor Wavelet cho phép mô tả cấu trúc của không gian tần số và mối liên hệ không gian.

PCA

Phân tích thành phần chính (Principal Component Analysis - PCA) là một phương pháp thống kê và toán học được sử dụng để giảm chiều dữ liệu, tìm ra các thành phần chính quan trọng nhất và tạo ra một biểu diễn mới của dữ liệu mà vẫn giữ lại nhiều thông tin nhất có thể

Các bước:

Input: X=[x1,x2,x3, .... , xn] là ma trận gồm các điểm dữ liệu xi là các vector cột D chiều. Ma trận X có kích thước là Dxn chiều

Output: Y=[y1,y2,y3, .... , yn] là ma trận gồm các điểm dữ liệu yi là các vector cột thuộc R^d (D>>d). Ma trận Y có kích thước là dxn chiều

Bước 1: tính trung bình dữ liệu input

Bước 2: lấy hiệu các giá trị trung bình các biến số

Bước 3: tìm các vectơ riêng và các giá trị riêng của ma trận đồng phương sai

Bước 4: tìm các thành phần chính bằng cách chiếu các quan sát lên các vectơ riêng

Bước 5: tính toán các tải (loading) chẳng hạn như sự tương quan giữa các biến số gốc và các thành phần

chính

Ứng dụng:

Ngoài mục đích giảm số chiều vector với các tập dữ liệu lớn, PCA còn được dùng để trích chọn đặc điểm cho hệ thống nhận dạng mặt hoặc PCA có thể dùng để nén dữ liệu ảnh, biểu diễn trực quan các dữ liệu đa chiều trong thống kê

Tính kernel bằng công thức Guassian

Nhớ điểm (0,0) điểm trung tâm

(1,1)

(0,0) (0,1)

G(x,y, ) = e-(x\*x+y\*y)/2\*

Trong đó, x,y là vị trị ( hàng,cột )

/2pi\*\*

0.059 0.091 0.059

0.091 0.16 0.091

0.059 0.091 0.059

Lấy  = 1:

G(0,0) = 1⁄2\*3.14 = 0.16 G(0,1)

= e-1/2/2pi=0.091

G(1,0) = 0.091

G(1,1) = e-2/2/2pi = 0.059