دانشگاه صنعتی خواجه نصیرالدین طوسی

دانشکده ریاضی

پروژه کارشناسی پردازش تصویر در متلب

استاد راهنما: دکتر جواد کوشکی

دانشجو: شاهین منصوری

مقدمه

- پردازش تصویر چیست ؟

مقدمه

- پردازش تصویر چیست ؟
- متلب

مقدمه

- پردازش تصویر چیست ؟
- متلب
- کاربردهای پردازش تصویر در متلب

مقدمه

- پردازش تصویر چیست ؟
- متلب
- کاربردهای پردازش تصویر در متلب
- مزایای استفاده از متلب برای پردازش تصویر

تعاریف اولیه

- معرفی ساختار داده در متلب

تعاریف اولیه

- معرفی ساختار داده در متلب
- ساخت پروژه

تعاریف اولیه

- معرفی ساختار داده در متلب
- ساخت پروژه
- باز کردن تصویر در متلب و برخی قواعد دسترسی به درایه های ماتریس

تعاریف اولیه

- معرفی ساختار داده در متلب
- ساخت پروژه
- باز کردن تصویر در متلب و برخی قواعد دسترسی به درایه های ماتریس

تبدیل تصویر رنگی به تصویر خاکستری

- ضرورت تبدیل تصویر به خاکستری

تبدیل تصویر رنگی به تصویر خاکستری

- ضرورت تبدیل تصویر به خاکستری
- الگوریتم تبدیل به تصویر خاکستری

تبدیل تصویر رنگی به تصویر خاکستری

- ضرورت تبدیل تصویر به خاکستری
- الگوریتم تبدیل به تصویر خاکستری

1. معیار قرار دادن یکی از رنگ های قرمز، سبز یا آبی

تبدیل تصویر رنگی به تصویر خاکستری

- ضرورت تبدیل تصویر به خاکستری
- الگوریتم تبدیل به تصویر خاکستری
  1. معیار قرار دادن یکی از رنگ های قرمز، سبز یا آبی
  2. معیار قرار دادن ماکسیمم مقدار رنگ یک پیکسل

تبدیل تصویر رنگی به تصویر خاکستری

- ضرورت تبدیل تصویر به خاکستری
- الگوریتم تبدیل به تصویر خاکستری
  1. معیار قرار دادن یکی از رنگ های قرمز، سبز یا آبی
  2. معیار قرار دادن ماکسیمم مقدار رنگ یک پیکسل
  3. میانگین گیری حسابی

تبدیل تصویر رنگی به تصویر خاکستری

- ضرورت تبدیل تصویر به خاکستری
- الگوریتم تبدیل به تصویر خاکستری
  1. معیار قرار دادن یکی از رنگ های قرمز، سبز یا آبی
  2. معیار قرار دادن ماکسیمم مقدار رنگ یک پیکسل
  3. میانگین گیری حسابی
  4. میانگین گیری وزنی

تجزیه SVD

- تجزیه SVD یک روش در درس جبرخطی عددی است که در آن یک ماتریس بزرگ را به مجموعه ای از ماتریس های کوچکتر تجزیه میکند

$$A = U \times S \times V^t$$

که در آن

- A یک ماتریس $m \times n$
- U یک ماتریس متعامد $m \times m$
- S یک ماتریس قطری $m \times n$
- V یک ماتریس متعامد $n \times n$

مقدمه تجزیه SVD

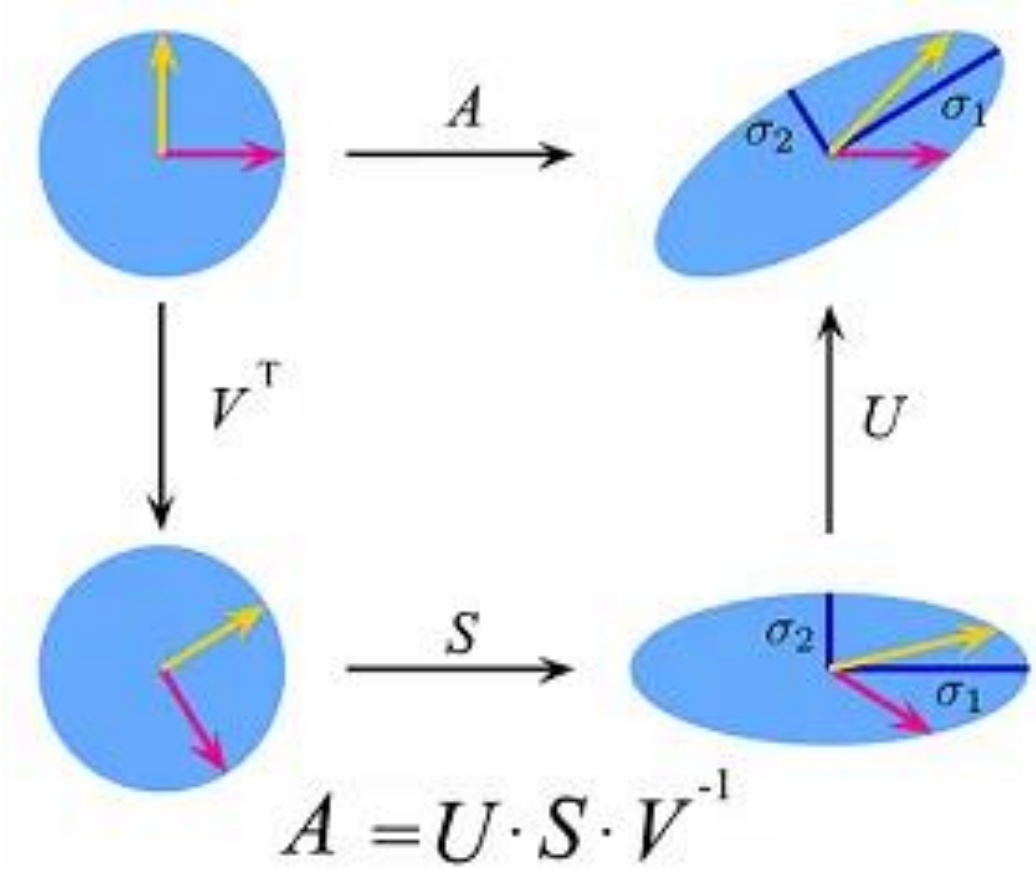
- درک شهودی:

برای درک شهودی تجزیه مقادیر منفرد، میتوان ماتریس A را به عنوان یک تبدیل خطی در نظر گرفت. این تبدیل خطی را نیز میتوان به سه زیر تبدیل تجزیه کرد:

1) چرخش یا دوران
2) تغییر مقیاس
3) چرخش

این سه گام منتاظر با سه ماتریس U، S و V هستند.

تجزیه SVD



دلیل استفاده از تجزیه SVD در پردازش تصویر

- تجزیه SVD برای کاهش ابعاد، فشرده سازی و استخراج ویژگی های مهم از تصویر استفاده میشود. به عنوان مثال، با استفاده از تجزیه SVD میتوان یک تصویر را به یک مجموعه از اجزای اصلی تجزیه کرد و از این اجزا برای تشخیص الگوها و میزگی های مهم در تصویر استفاده کرد

- Image compression
- Classification
- Feature extraction
- Pattern recognition
- Projection
- Multiscale signal analysis

روش به دست آوردن تجزیه SVD در متلب

- در نرم افزار متلب تابعی وجود دارد با نام svd که این تجزیه را برای ما انجام میدهد به شکل زیر:

```
[u, s, v] = svd(A)
```

تابع svd یک آرگومان اصلی دریافت میکند که ماتریس $n \times m$ است و به عنوان خروجی سه مقدار u,s,v را برمیگرداند.

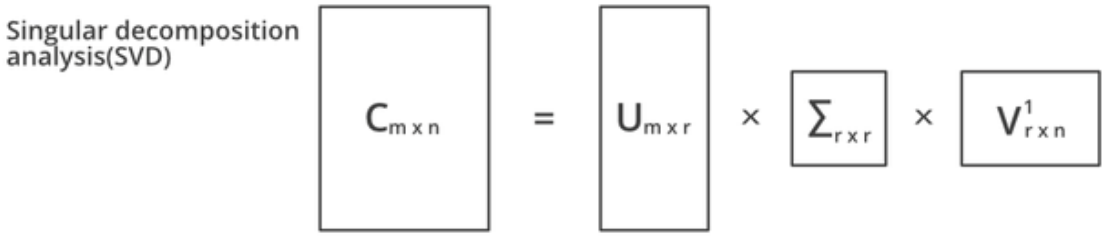
تئوری تجزیه SVD

- ماتریس A با m ردیف و n ستون، با رتبه rبطوری که

$$r \leq n \leq m$$

سپس ماتریس A را میتوان به سه ماتریس تقسیم کرد:

$$A = USV^t$$



تئوری تجزیه SVD

- که در آن ماتریس U یک ماتریس متعامد $m \times m$ است.
- $U = [u_1, u_2, \dots, u_r, u_{r+1}, \dots, u_m]$
- برای بردار های u_i و $for\ i = 1,2, \dots, m$ یک مجموعه متعارف را تشکیل میدهند:

$$u_i^t\ u_j = \delta_y = \begin{cases} 1,,, i = j \\ 0,,, i \neq j \end{cases}$$

- و ماتریس V یک ماتریس $n \times n$ است

$$v = [v_1, v_2, \dots, v_r, v_{r+1}, \dots, v_n]$$

$$v_i^t\ v_j = \delta_{ij} = \begin{cases} 1,,, i = j \\ 0,,, i \neq j \end{cases}$$

تئوری تجزیه SVD

- S یک ماتریس مورب $n \times m$ با مقادیر منفرد (SV)است . ماتریس S به شکل زیر است

$$S = \begin{bmatrix} \sigma_1 & 0 & \cdots & 0 & 0 \\ 0 & \sigma_2 & & 0 & 0 \\ \vdots & & \ddots & \vdots & \\ 0 & 0 & & \sigma_{n-1} & 0 \\ 0 & 0 & \cdots & 0 & \sigma_n \\ 0 & 0 & & 0 & 0 \end{bmatrix}$$

برای $i = 1,2, \dots ,n$ و σ_i را مقادیر منفرد ماتریس A مینامیم که به صورت زیر است

$$\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_r > 0$$

$$And$$

- $\sigma_{r+1} = \sigma_{r+2} = \cdots = \sigma_N = 0$

مقادیر v_i و u_i ها را مینامیم بردار های راست و چپ ماتریس A.

روش تجزیه SVD کاربردی به پردازش تصویر

- تجزیه SVD برای فشرده سازی تصویر

فشرده سازی تصویر با مشکل کاهش داده های مورد نیاز برای نمایش یک تصویر دیجیتال سر و کار دارد. در بسیاری از کاربردها، مقدار منفرد یک ماتریس با افزایش رتبه، به سرعت کاهش پیدا میکند. این ویژگی به ما این اجازه را میدهد تا با حذف مقادیر کوچک، نویز را کاهش دهیم یا داده های ماتریسی را فشرده کنیم.

روش تجزیه SVD کاربردی به پردازش تصویر

- تجزیه SVD برای تشخیص چهره

رویکرد SVD مجموعه ای از چهره های شناخته شده را به عنوان بردار هایی در یک زیرفضا به نام "*Face Space*" که توسط گروه کوچکی از "$base\ faces$" پوشانده شده است در نظر میگیرد.

تشخیص با نمایش یک تصویر جدید بر روی فضای صورت انجام میشود و سپس با مقایسه مختصات آن در فضای چهره با مختصات چهره شناخته شده، صورت را طبقه بندی میکند.

تشخیص چهره با تجزیه SVD

- فرض کنید هر تصویر چهره دارای $M = m \times n$ پیکسل است و به عنوان یک بردار ستونی f_i $M \times 1$ نشان داده میشود. یک مجموعه آموزشی S با N تعداد تصویر چهره افراد شناخته شده یک ماتریس $M \times N$ را تشکیل میدهد:

$$S = [f_1, f_2, \dots, f_N]$$

- تصویر میانگین $\overline{f}$ از مجموعه S توسط

- $\overline{f} = \frac{1}{N}\sum_{i=1}^{N} f_i$

- با کم کردن $\overline{f}$ از وجه های اصلی به دست می آید.

- $a_i = f_i - \overline{f}$,,,$i = 1,2, \dots, N$

تشخیص چهره با تجزیه SVD

- این یک ماتریس $m \times n$ دیگر A را میدهد:

$$A = [a_1, a_2, \dots, a_N]$$

- این ماتریس A را به عنوان مجموعه ای از چهره train تعریف میکنیم.
- فرض کنید $x = [x_1, \dots, x_r]^t$ مختصات (موقعیت) هر *mxn* تصویر صورت f در زیرفضای صورت باشد. سپس این نمایش اسکالر $f - \overline{f}$ بر روی سطوح پایه است:

- $$x = [u_1, \dots, u_r]^T (f - \overline{f})$$

- از این بردار مختصات *x* برای یافتن اینکه کدام یک از چهره های آموزشی چهره f را به بهترین شکل توصیف میکند استفاده میشود. یعنی پیدا کردن مقداری وجه آموزشی f_i که *i=1, 2, ..., N* که فاصله را به حداقل میرساند:

تشخیص چهره با تجزیه SVD

$$\varepsilon_i = \left|\left|x - x_i\right|\right|_2 = [(x - x_i)^t (x - x_i)]^{.5}$$

- که در آن x_i بردار مختصات f_i است که نمایش اسکالر $f - \bar{f}$ بر روی وجوه پایه است:

$$x_i = [u_1, \ldots., u_r]^T \left(f_i - \overline{f}\right)$$

- وقتی که حداقل ε_i کمتر از آستانه از پیش تعریف شده ε_0 باشد، صورت f به عنوان وجه f_i طبقه بندی میشود. در غیر این صورت، چهره f به عنوان چهره ناشناخته طبقه بندی میشود. اگر f یک صورت نباشد، فاصله آن تا زیر فضای صورت بزرگتر از صفر خواهد شد. از آنجایی که طرح برداری $f - \overline{f}$ بر روی فضای صورت با

- $$f_p = \ [u_1, \ldots, u_r]x$$

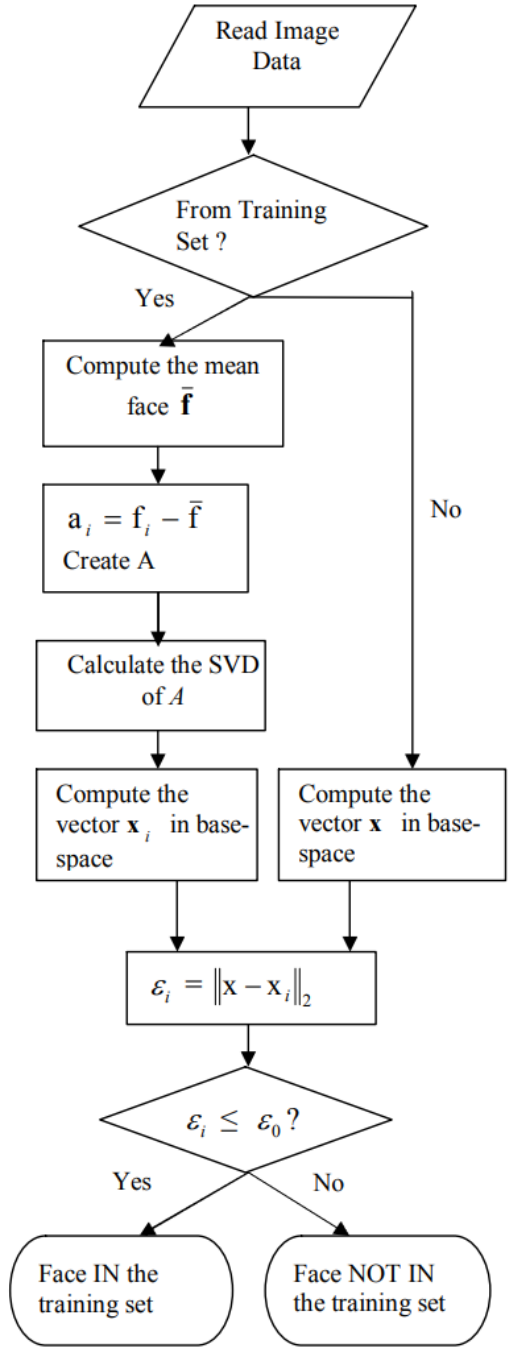
- داده میشود که در آن x داده شده است.

تشخیص چهره با تجزیه SVD

- فاصله f تا فضای صورت، فاصله بین $f - \overline{f}$ و طرح f_p روی فضای صورت است:

$$\varepsilon_f = \left|\left|(f - \overline{f}) - f_p\right|\right|_2 = \left[(f - \overline{f} - f_p]^T (f - \overline{f} - f_p)\right]^{1/2}$$

- اگر ε_f بزرگتر از آستانه از پیش تعریف شده ε_1 باشد، آنگاه f تصویر چهره نیست.


Read Image Data
From Training Set ?
Yes
No
Compute the mean face $\bar{\mathbf{f}}$
$\mathbf{a}_i = \mathbf{f}_i - \bar{\mathbf{f}}$
Create A
Calculate the SVD of A
Compute the vector $\mathbf{x}_i$ in base-space
Compute the vector $\mathbf{x}$ in base-space
$\varepsilon_i = \|\mathbf{x} - \mathbf{x}_i\|_2$
$\varepsilon_i \leq \varepsilon_0$?
Yes
No
Face IN the training set
Face NOT IN the training set

پیاده سازی در متلب

- خواندن تصویر

```
function img = readImage( img_path )
% open input image
img = imread(img_path);
img = imresize(img, 0.5);
%converting to grayscale
if size(img,3)==3
img=rgb2gray(img);
end
% convert from uint8 to double
img=double(img);
```

پیاده سازی در متلب

- تشخیص چهره

```
% Load one of images to work with sizes
input_image = readImage(strcat('train/', '1.jpg'));
% Number of individuals
N = 36;
% Size of each image
% assume each face image has mxn=M
M = size(input_image, 1) * size(input_image, 2);
% Creating the train data matrix
% S = [f_1, f_2, ..., f_N]
S = zeros(M, N);
for i = 1:N
    temp = readImage(sprintf('train/%d.jpg', i));
    S(:, i) = temp(:);
end
```

```
% Computing the mean image of train data
% mean(S, 2) = row average of S
% fbar = (1/N)*sum(f_i, i=1:N)
train_mean = mean(S, 2);
% imwrite(uint8(train_mean(:, ones(1, N))), 'mean.jpg');
```

```
% Normalizing images by subtracting mean
A = S - train_mean(:, ones(1, N));
```

```
% Performing the Singular Value Decomposition over A
% econ = SVD decomposition is done economically with high speed
% The outputs of svd decomposition are [u, s, v], we only need u
[u, ~, ~] = svd(A, 'econ');
```

پیاده سازی در متلب

```
% Classification
% images = ['1' '3' '5' '11' '25' '38' 'nothing' 'U1' 'U2' 'U3'];
% Test Images Available
epsilons = zeros(N, 1);
test_image = readImage('test/3.jpg');
test_image = test_image(:) - train_mean; % Normalizing test image
x = u(:, 1:rank)' * test_image; % Calculating coordinate vector x of test image
```

```
% epsilon_f = ||(f - fbar) - f_p||_2 = [(f - fbar - f_p)^T*(f - fbar - f_p)]^.5
epsilon_f = ((test_image - u(:, 1:rank) * x)' * (test_image - u(:, 1:rank) * x)) ^0.5;
```

•

```
% Computing coordinate vector xi for each known individual
% size(A, 2) = The size of the columns
rank = size(A, 2);
% u(:, 1:rank) = for compression
xi = u(:, 1:rank)' * A;
% xi = u' * A;
```

```
% Defining thresholds, these values were defined by trial and error
epsilon_0 = 50; % Maximum allowable distance from any known face in the training set S
epsilon_1 = 15; % Maximum allowable distance from face space
```

پیاده سازی در متلب

```
% Checks if it is in face space
if epsilon_f < epsilon_1
    % Computing distance epsilon_i to the face space
    for i = 1:N
       epsilons(i, 1) = (xi(:, i) - x)' * (xi(:, i) - x);
    end
    [val, idx] = min(epsilons(:, 1));
    if val < epsilon_0
        fprintf('The face belongs to %d\n', idx);
    else
        disp('Unknown face’);
    end
else
    disp('Input image is not a face');
end


fprintf('Finished :))\n')
```

فشرده سازی تصویر با استفاده از تجزیه SVD

```
function [AK, Cr] = svdComp(A,K)
    A=double(A);
    m = size(A, 1);
    n = size(A, 2);
    Cr = (m*n)/(K*(m+n+1));
    [u,s,v]=svd(A);
    AK=u(:,1:K)*s(1:K,1:K)*v(:,1:K)';
    AK=uint8(AK);
    %imshow(AK);
    end
```

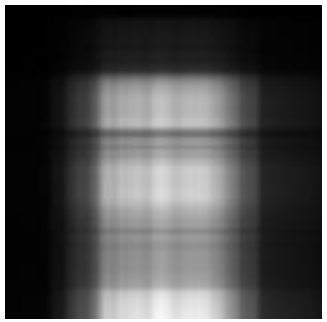
- که در اینجا Cr جهت اندازه گیری فشرده سازی تصویر است و در فشرده سازی نقشی ندارد.

فشرده سازی تصویر با استفاده از تجزیه SVD

- به طور خلاصه

1) تعداد ستون های ماتریس u به k محدود شده است و تعداد ردیف های آن دستخوش تغییر نشده است.
2) تعداد ستون ها و ردیف های ماتریس s نیز تا مقدار k محدود شده است.
3) تعداد ستون های ماتریس v نیز به k محدود شده است و چون ترانهاده شده است، مشکلی در ضرب این سه ماتریس ایجاد نمیکند. انگار گوییم تعداد ردیف های ماتریس v^T به k محدود شده است.

خروجی قطعه کد بالا به ازای k های مختلف



$k = 1$

$k = 5$

$k = 10$

$k = 15$

$k = 20$

$k = 30$

$k = 40$

اندازه گیری کیفیت و فشرده سازی تصویر

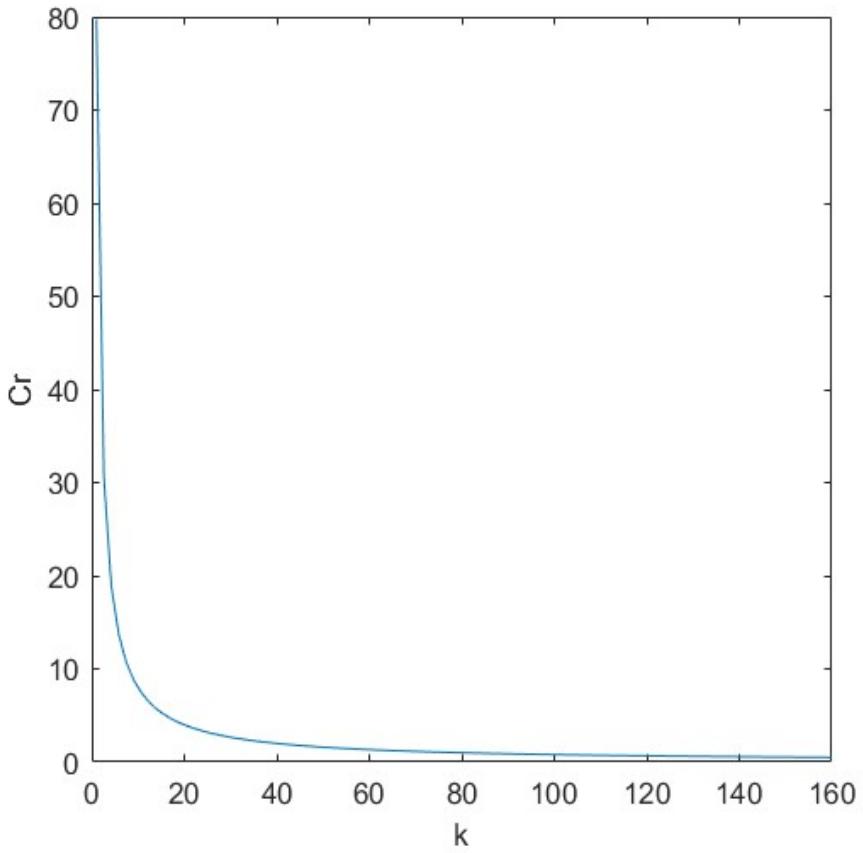
- برای اندازه گیری عملکرد فشرده سازی تصویر با کمک تجزیه SVD میتوانیم ضریب فشرده سازی و کیفیت تصویر فشرده شده را محاسبه کنیم.
  ضریب فشرده سازی تصویر را میتوان با استفاده از نسبت فشرده سازی محاسبه کرد:

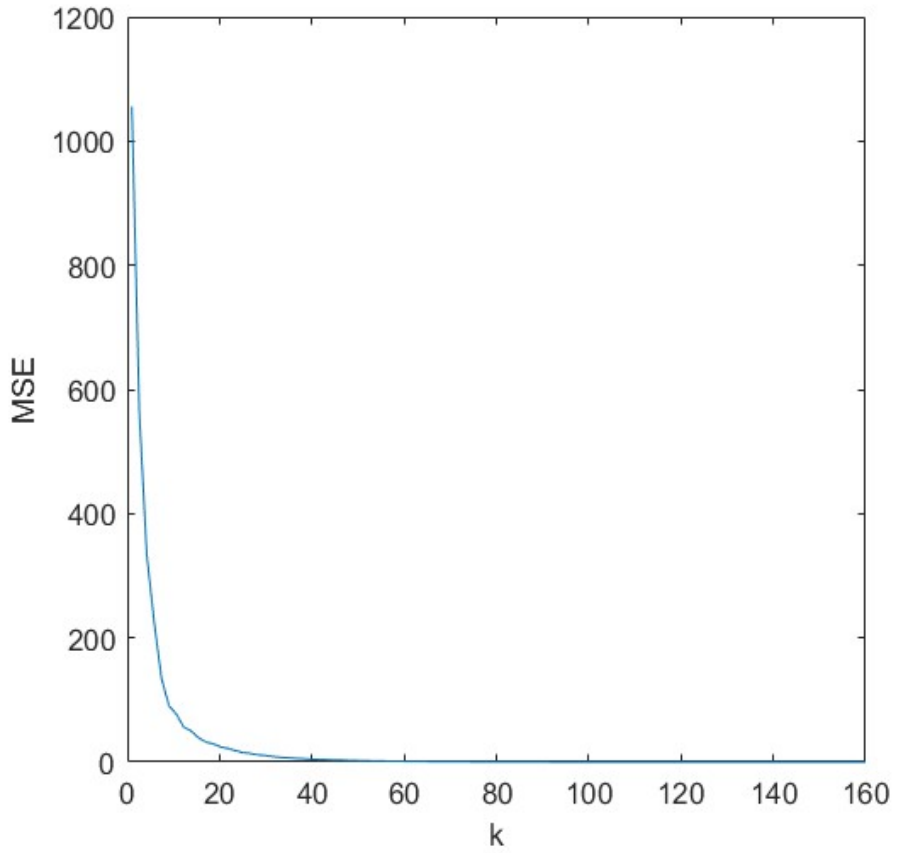
$$C_r = m * n \,/\, (k(m + n + 1))$$

- برای اندازه گیری کیفیت بین تصویر اصلی A و تصویر فشرده شده Ak میتوان اندازه گیری میانگین مربع خطا (MSE) را محاسبه کرد:

$$MSE = \frac{1}{nm} \sum_{y=1}^{m} \sum_{x=1}^{n} (f_A(x, y) - f_{A_k}(x, y))$$

بررسی ضریب فشردگی و کیفیت تصاویر فوق





نتیجه گیری

1) با استفاده از k کوچکتر (مقدار منفرد کمتر)، نسبت تراکم بهتری به دست می آید
2) مقدار منفرد بیشتر استفاده میشود (k بزرگتر) اندازه گیری کیفیت MSE کوچکتر است (کیفیت تصویر بهتر) و تصاویر باز سازی شده برابر با اصلی هستند.
3) برای تصویر تست، کیفیت تصویر قابل قبول $k = 20$است که نسبت فشرده سازی $Cr = 3.9875$ است.
1) تصویر نزدیک به تصویر اصلی زمانی که $k = 40$ است حاصل میشود. در این مرحله $Cr = 1.9938$ و $MSE = 4.3370$ است.

تجزیه QR

- تجزیه QRروشی در جبرخطی است. تجزیه QR یک روش در جبرخطی عددی است که می‌تواند به صورت مفید در چندین زمینه از پردازش تصویر به کار رود. به‌طور کلی، تجزیه QRیک ماتریس Aرا به صورت حاصل‌ضرب دو ماتریس Q و R به‌دست می‌آورد، که Q یک ماتریس متعامد و R یک ماتریس مثلثی بالاست.

تجزیه QR

- **برخی از کاربردهای تجزیه QR در پردازش تصویر:**

1. **کاهش ابعاد و فشرده‌سازی تصویر** :تجزیه QRمی‌تواند برای کاهش ابعاد داده‌های تصویری استفاده شود. این روش به خصوص در کاربردهای مانند PCA(تحلیل مؤلفه‌های اصلی) به کار می‌رود.
2. **حل سیستم‌های خطی** :در برخی از مسائل پردازش تصویر، مانند حل سیستم‌های معادلات خطی که ممکن است در بازسازی تصویر یا تحلیل تصویر بکار رود، تجزیه QRمی‌تواند به عنوان یک روش پایدار و مؤثر برای حل این معادلات استفاده شود.
3. **الگوریتم‌های تشخیص الگو و یادگیری ماشین** :در الگوریتم‌های تشخیص الگو و یادگیری ماشین، تجزیه QRمی‌تواند برای بهینه‌سازی و تثبیت محاسبات استفاده شود.
4. **فیلترهای تطبیقی** :در برخی از روش‌های فیلترینگ، مانند فیلترهای تطبیقی، تجزیه QRمی‌تواند برای به روز رسانی ضرایب فیلتر به کار رود.

تئوری تجزیه QR

- فرض میکنیم ماتریس A یک ماتریس $m \times n$ است. یک ماتریس متعامد Q و یک ماتریس بالا مثلثی R وجود دارد به قسمی که

$$A = QR$$

ماتریس Q میتواند به صورت $Q = H_1 H_2 \dots H_{n-1}$ نوشته شود که در آن H_i یک ماتریس هاوس هولدر است. یک تجزیه از A به صورت QR به صورت QR یک تجزیه QR از A نامیده میشود. در تجزیه QR ماتریس Q یک ماتریس متعامد $m \times n$ است که نرم واحد دارند و R یک ماتریس بالا مثلثی با ابعاد $n \times n$ است.

کاهش ابعاد

- برای فشرده سازی، ما یک مقدار k را انتخاب میکنیم که کوچکتر از حداقل ابعاد ماتریس A است. یعنی

$$k < \min(n, m)$$

سپس ماتریس های Q و R را به زیرماتریس هایی با ابعاد k کاهش میدهیم

$$R_k = R(:, 1:k)$$

$$Q_k = Q(1:k, :)$$

که در اینجا Q_k به k سطر محدود و R_k نسبت به k ستون محدود شده است.

کاهش ابعاد

- عمل محدود سازی سطر و ستون ماتریس Q و R را به شکل زیر انجام میدهیم

- $$Q = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{bmatrix}, R = \begin{bmatrix} r_{11} & 0 & \cdots & 0 \\ r_{21} & r_{22} & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots \\ r_{n1} & r_{n2} & \cdots & r_{nn} \end{bmatrix}$$

- که Q_k و R_k به شکل زیر میشود

- $$Q_k = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \dots & a_{mk} \end{bmatrix}, R = \begin{bmatrix} r_{11} & 0 & \cdots & 0 \\ r_{21} & r_{22} & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots \\ r_{k1} & r_{n2} & \cdots & r_{nn} \end{bmatrix}$$

بازسازی تصویر

- با استفاده از این زیر ماتریس ها، تصویر بازسازی شده A_k را محاسبه میکنیم به صورت

$$A_k = Q_k R_k$$

این ماتریس بازسازی شده یک تقریب از ماتریس اصلی A است که تنها از k مولفه اصلی استفاده کرده است. این فرآیند باعث کاهش ابعاد داده ها میشود و برخی از اطلاعات موجود در تصویر اصلی از دست میرود، اما هدف این است که با حفظ بخش عمده ای از اطلاعات اصلی، حجم داده ها را کاهش دهیم.

مزایا و محدودیت ها

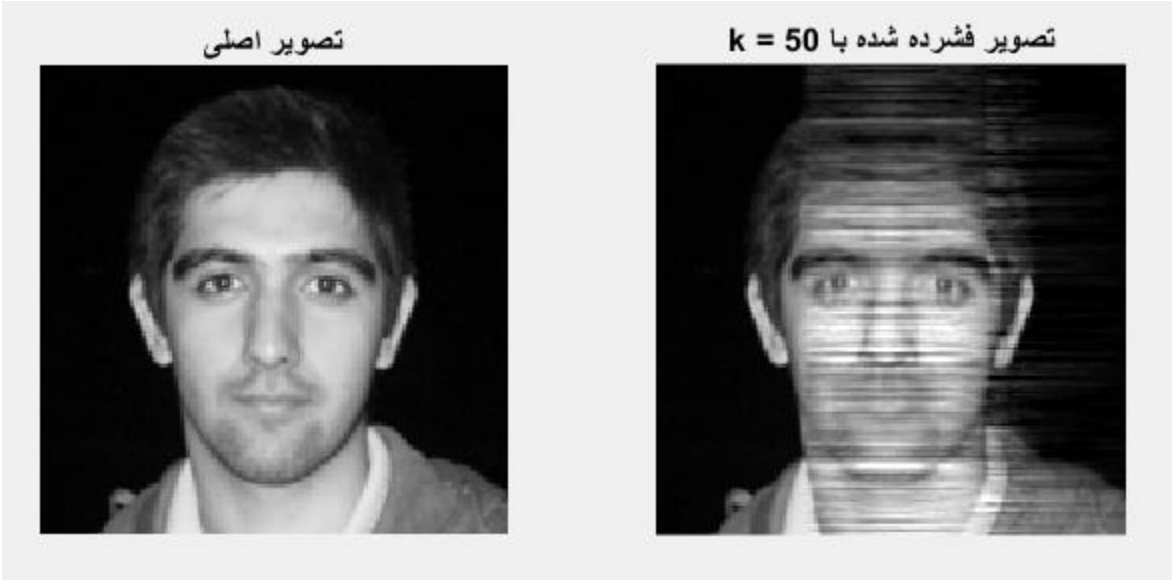
- **مزایا:**
- کاهش حجم داده ها و ذخیره سازی بهتر
- تسریع در پردازش به دلیل کاهش ابعاد
- حذف نویز و اطلاعات غیرضروری
- **محدودیت ها:**
- ممکن است برخی از جزئیات مهم تصویر از دست برود
- مقدار k باید به دقت انتخاب شود تا تعادل مناسبی بین کاهش حجم و حفظ کیفیت تصویر برقرار شود
- به طور کلی، تجزیه QR یک ابزار مفید برای فشرده سازی داده ها است که با کاهش ابعاد ماتریس ها میتواند به صورت کارآمدی اطلاعات اصلی را حفظ کند.

فشرده سازی تصویر با تجزیه QR در متلب

```
function Ak = qrComp(A, k)
    [Q, R] = qr(A);
    Qk = Q(:, 1:k);
    Rk = R(1:k, :);
    Ak = Qk*Rk;
    % Ak = uint8(Ak);
end
```

مقایسه تصویر فشرده شده با تصویر اصلی

- به ازای $k = 50$ داریم

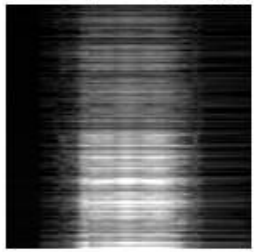


مقایسه تصویر فشرده شده با تصویر اصلی

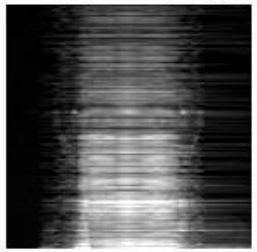
- به ازای k های مختلف

تصویر فشرده شده با k = 1

تصویر فشرده شده با k = 10



تصویر فشرده شده با k = 20



تصویر فشرده شده با k = 40

تصویر فشرده شده با k = 80

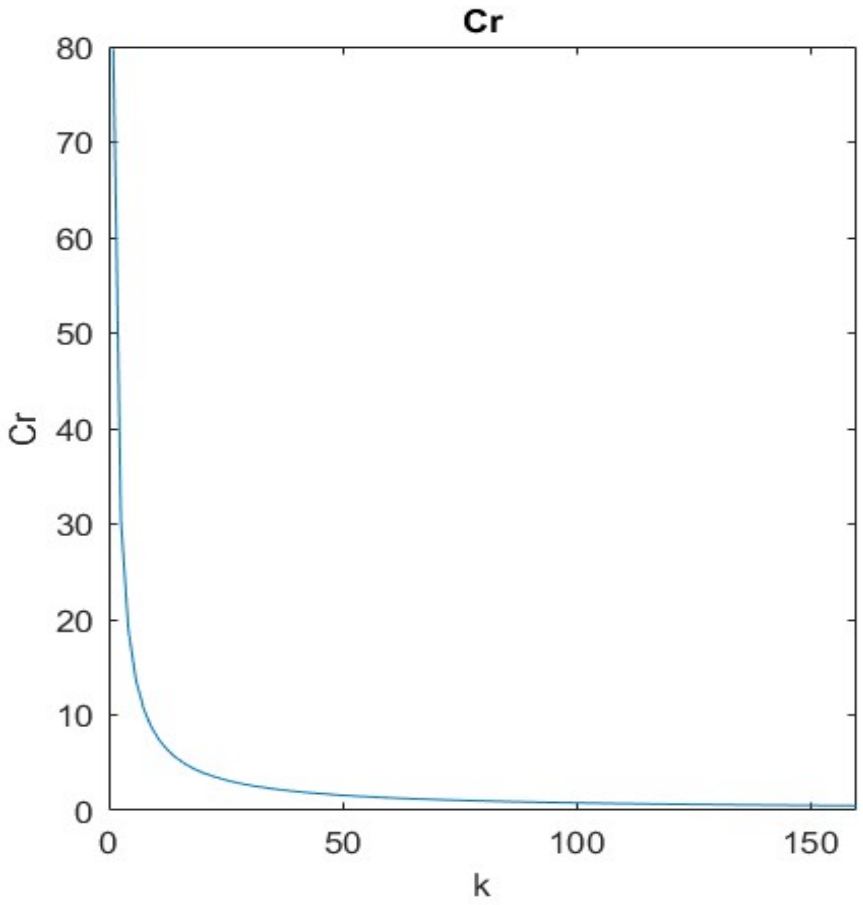
تصویر فشرده شده با k = 120

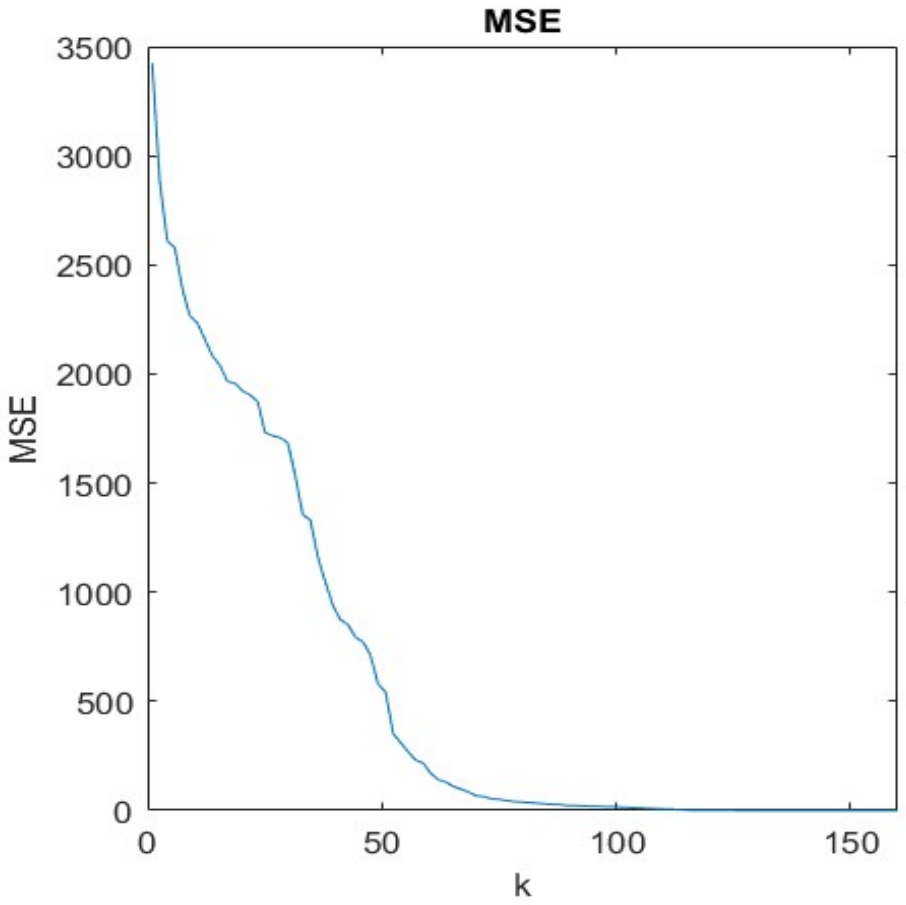
تصویر اصلی

اندازه گیری فشرده سازی و کیفیت تصویر

k	Cr	MSE
1	78.7508	3.4236e+03
10	7.9751	2.2339e+03
20	3.9875	1.9226e+03
40	1.9938	889.1603
80	0.9969	37.4356
120	0.6646	0.7340

اندازه گیری فشرده سازی و کیفیت تصویر





- چون تصویر انتخابی ما 160 در 160 بود، بیشینه مقدار k انتخابی ما نیز بین 1 تا 160 است

نتیجه گیری

- از نمودار فوق میتوان نتیجه گرفت مقدار k از یک مقدار مشخصی کمتر میشود، میزان رشد نمودار بسیار زیاد میشود. پس میتوان مقدار k را طوری انتخاب کرد که فشرده سازی انجام گیرد درحالی که کیفیت تصویر شامل تغییر چندانی نشود. در این تصویر خاص، حدودا مقدار $k = 80$ میتواند معیار مناسبی برای فشرده سازی تصویر باشد.

الگوریتم KNN

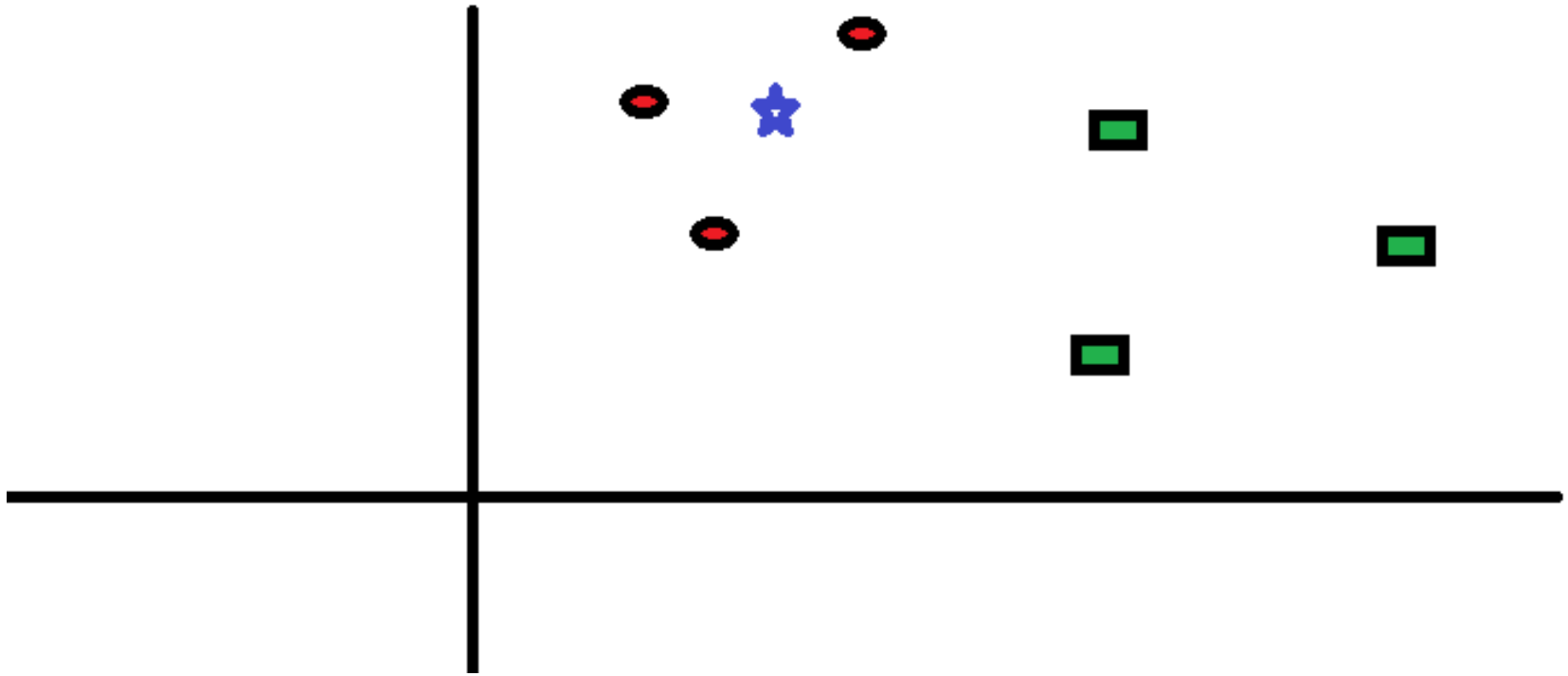
- به معنی K نزدیک ترین همسایگی است که یک روش ناپارامتری است که در داده کاوری، یادگیری ماشین و تشخیص الگور مورد استفاده قرار میگیرد. این الگوریتم یکی از ۱۰ الگوریتمی است که بیشترین استفاده در پروژه های گوناگون یادگیری ماشین و داده کاوی، هم در صنعت و هم در دانشگاه داشته است.

- **چه زمانی از این الگوریتم استفاده میشود ؟**

الگوریتم KNN برای مسائل طبقه بندی و رگرسیون قابل استفاده است که ما در ادامه برای طبقه بندی (خوشه بندی داده ها) استفاده میکنیم.

الگوریتم KNN

- چگونه کار میکند ؟



الگوریتم KNN در متلب

- در متلب تابعی وجود دارد تحت نام *kmeans* که در ورودی یک بردار و عدد k را دریافت میکند که k تعداد خوشه های ما در این بردار است. برای مثال ما میخواهیم داده هایی که به صورت برداری داریم را به دو دسته تقسیم کنیم. در این صورت به عنوان ورودی مقدار k را برابر با ۲ به *kmeans* میدهیم. ورودی های دیگری برای تابع *kmeans* وجود دارد که در صورت نیاز میتوانیم وارد کنیم.

- `kmeans(X,20,'Distance','sqeuclidean');`

$$d(x,c) = (x-c)(x-c)$$

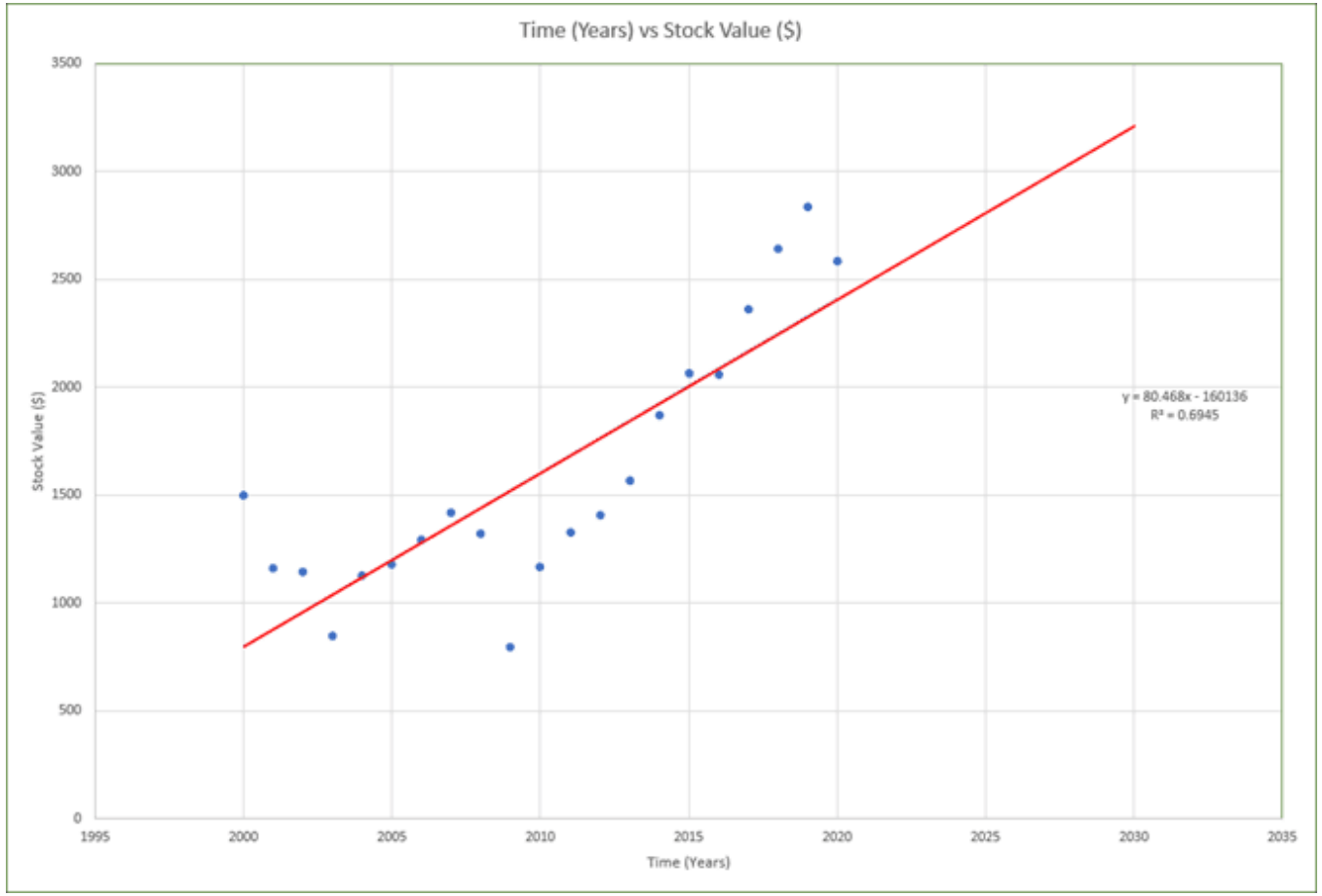
فاصله اقلیدسی مربع که هر مرکز، میانگین نقاط آن خوشه است.

در خروجی

C: مکان های مرکز خوشه k را در یک ماتریس به ما برمیگرداند

Idx: شاخص های خوشه را با گزینه های اضافی مشخص شده توسط یک یا چند آرگومان برمیگرداند.

رگرسیون خطی


Time (Years) vs Stock Value ($)
Stock Value ($)
Time (Years)
y = 80.468x - 160136
R² = 0.6945
3500
3000
2500
2000
1500
1000
500
0
1995
2000
2005
2010
2015
2020
2025
2030
2035

رگرسیون خطی در متلب

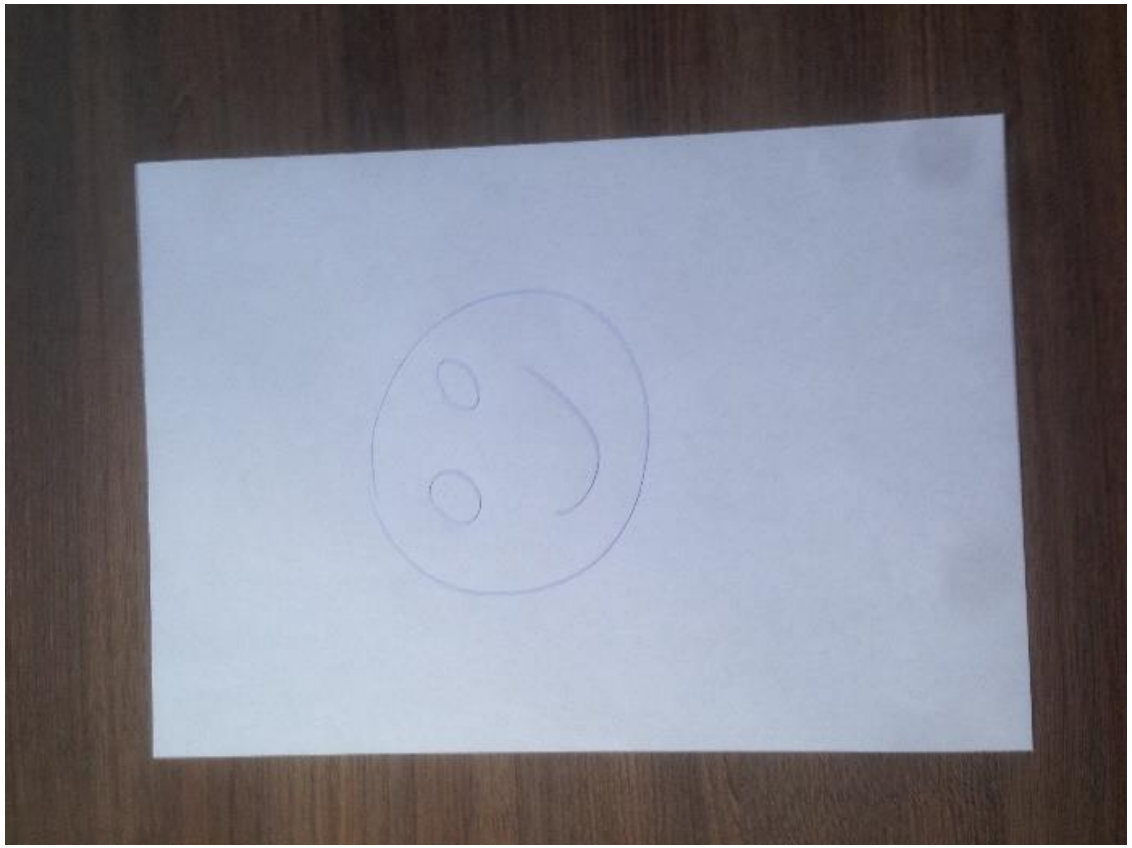
```
function [a, b] = linregfunc(X, Y)
    syms a; syms b
    Sum = 0;
    for i = 1:length(X)
        x = X(i);
        y = Y(i);
        Sum = Sum + (a*x + b - y)^2;
    end
    s1 = diff(Sum, a);
    s2 = diff(Sum, b);
    [a, b] = solve([s1, s2], [a, b]);
```

تشخیص سند با استفاده از رگرسیون خطی و الگوریتم KNN

- فرض کنیم یک تصویر وجود دارد که در آن یک برگه وجود دارد. میخواهیم این برگه را از میان اشیاء دیگر پیدا کنیم. برای این کار لازم است به طور کلی، با استفاده از الگوریتم KNNتصویر را به دو طبقه کاغذ و غیر کاغذ تقسیم کنیم و در نهایت با استفاده از رگرسیون خطی، شیب برگه را پیدا کنیم و در نهایت این شیب را خنثی کنیم، یعنی برگه را صاف کنیم.

تشخیص سند با استفاده از رگرسیون خطی و الگوریتم KNN

- تصویر مدنظر



تشخیص سند در متلب

```
% بارگذاری تصویر
I = imread('image2.JPG');

% تبدیل تصویر به خاکستری
grayImage = rgb2gray(I);

% تبدیل تصویر به بردار
pixelValues = double(grayImage(:));

% تعیین تعداد خوشه‌ها
numClusters = 2;

% اجرای K-means با فاصله اقلیدسی
]idx, ~] = kmeans(pixelValues, numClusters, 'Distance', 'sqeuclidean');
```

```
% تبدیل درایه ها به سیاه و سفید
idx = (idx - 1) * 255;

% بازسازی تصویر خوشه‌بندی شده
img = reshape(idx, size(grayImage));

imshow(uint8(img));
pause

img = cropBlacks(img);
imshow(uint8(img))
pause;
```

تشخیص سند در متلب

```
[a, b] = linregfunc(x, y);


img = uint8(img);
a = double(a);
img = imrotate(img, rad2deg(asin(a)), 'bilinear', 'loose');


img = cropBlacks(img);




imshow(uint8(img))
pause
close all;
```

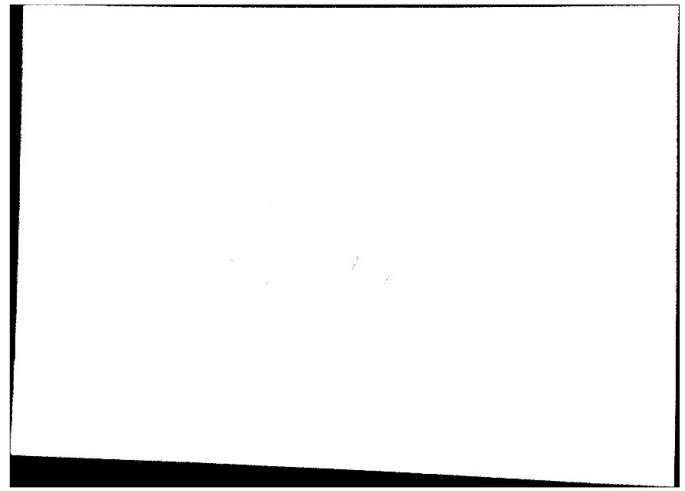
```
img = cropBlacks(img);
imshow(uint8(img))
pause;


x = floor(linspace(200, size(img, 2)-200, 30));
y = [];
for i=x
    r=1;
    while r < size(img, 1) && img(r, i) ~= 255
        r = r+1;
    end
    y = [y, r];
end
```

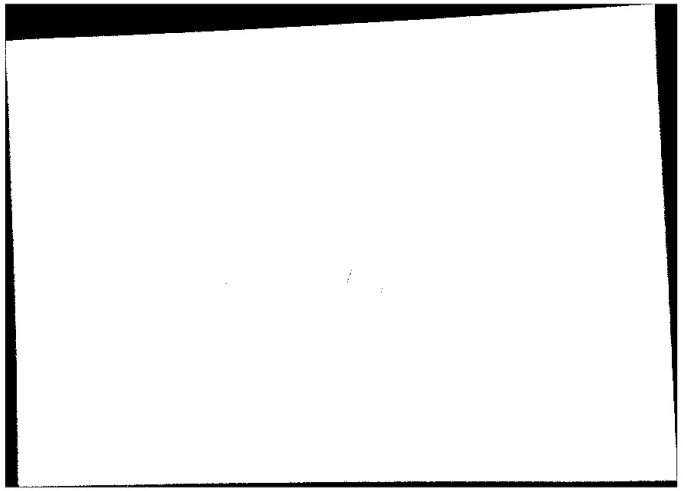
تشخیص سند در متلب

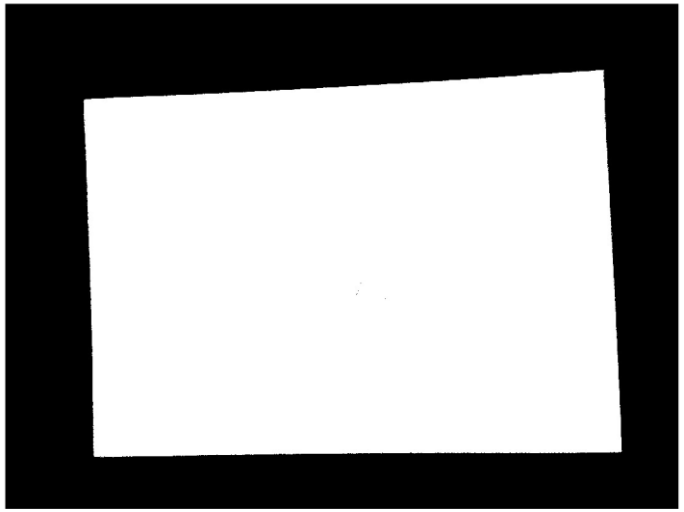
```
function answer = cropBlacks(img)
    img = double(img);
    zeroRows = all(img == 0, 2);
    img(zeroRows, :) = [];
    img = img’;
    zeroRows = all(img == 0, 2);
    img(zeroRows, :) = [];
    img = img’;
    answer = img;
end
```

تشخیص سند در متلب

- تصاویر حاصل در هر مرحله







ممنون از توجه شما.