تشخیص چهره با استفاده از svd

استاد: آرتا جمشیدی

نویسنده:علیرضا قنبری

شماره دانشجوی:610396130

مقدمه:

استفاده از الگوریتم ها و محاسبات برای پردازش ، دستکاری و حتی تولید تصاویر به دلیل طیف گسترده ای از کاربردهای عملی که می توان از آنها استفاده کرد ، در دانشگاه ریاضیات و علوم کامپیوتر بسیار محبوب شده است. این را می توان در پیشرفت های تشخیص چهره ، نظارت بر محیط زیست و فناوری تصویربرداری پزشکی مشاهده کرد. پردازش تصویر از طریق تکنیک های عددی اغلب به مقدار زیادی منابع محاسباتی و ذخیره حافظه نیاز دارد. با توجه به این واقعیت ، یکی از موارد اولیه ای که در بسیاری از الگوریتم های پردازش تصویر کم کردن ابعاد مجموعه تصویر مورد نظر است. پس از اتمام این امر ، باید فرایندهایی مانند تجزیه و تحلیل مولفه های اصلی یا تصویر را انجام دهید. در نتیجه محاسبات و تشخیص چهره بسیار راحت تر و در زمان معقول تر انجام می شود.

تصاویر:

تصاویر بانک اطلاعاتی آزمایشی این پروژه شامل 30 تصویر صورت جمع آوری شده از پایگاه داده ای از تصاویر صورت در دانشگاه ماساچوست ، آمهرست در بخش علوم کامپیوتر است. تصاویر در مقیاس خاکستری هستند و نشان می دهند چهره مردان و زنان در موقعیت نسبتاً ثابت است.هر عکس به عنوان یک ماتریس از ابعاد 200*150 نشان داده شده است. هر ورودی در یک ماتریس معین نماینده یک پیکسل در تصویر است و نشان دهنده انحراف آن پیکسل از میانگین خاکستری کل تصویر است. که البته با انجام یک سری از دستورات ما ابتدا تصاویر را سیاه سفیپ می کنیم (برای اطمنیان در صورت وجود عکس رنگی) سپس اندازه همه ی عکس ها را به ۱۰۰*۱۰۰ تبدیل می کنیم:

```
temp = cv2.cvtColor(temp, cv2.COLOR_BGR2GRAY)
temp = cv2.resize(temp, (100, 100), interpolation=cv2.INTER_AREA)
```

تصویر متوسط

اولین قدم برای توصیف این مجموعه تصاویر ، بدست آوردن یک تصویر متوسط است. از این میانگین برای محاسبه بعدی (normalization) در الگوریتم استفاده خواهد شد.

$$\overline{\varphi} = \frac{1}{M}\sum_{i=1}^{M} \varphi_i$$

که در این فرمول M تعداد کل تصاویر(30) است

و هر φ_i ماتریس یک عکس است :

$$\varphi = \begin{pmatrix} x_{11} & x_{12} & \cdots & x_{1m} \\ x_{21} & x_{22} & \cdots & x_{2m} \\ \vdots & \ddots & \vdots & \vdots \\ \cdots & \cdots & \cdots & \cdots \end{pmatrix}$$

که در کد ما به صورت زیر پیاده سازی شده است :

```
mages = np.array(images)

mu = np.mean(images)

images = images - mu
```

ماتریس آشفتگی:

از این تصویر متوسط می توان ماتریس آشفتگی را ایجاد کرد. این ماتریس مقداری را برای انحراف هر تصویر از این میانگین پایگاه داده فراهم می کند و با P نشان داده می شود.

$$P = (\phi_1 \quad \phi_2 \quad \phi_3 \quad \cdots \quad \phi_M) \; where \; \phi_i = \varphi_i - \bar{\varphi}$$

که در کد ما به صورت زیر پیاده سازی شده است :

```
mages = np.array(images)

mu = np.mean(images)

images = images - mu
```

ماتریس کوواریانس

مولفه های بعدی لازم برای تشخیص چهره بردارهای ویژه ماتریس کوواریانس برای گروه تصویر هستند. ماتریس کوواریانس را می توان با گرفتن دوتایی هر یک از عناصر ماتریس آشفتگی نسبت به یکدیگر پیدا کرد. این نیز معادل ضرب ماتریس در ترانهاده خود است. این ماترسی را با C نشان می دهند:

$$C = \left(\frac{1}{1-M}\right) PP^T = \left(\frac{1}{1-M}\right) \sum_{i=1}^{M} \phi_i \phi_i^T$$

چارچوب ریاضی

SVD ، نمایش هر ماتریس n*m توسط سه ماتریس مجزا است که نمایانگر یک تحول در سه مرحله است. این روشی است که به طور مکرر در شرایطی مورد استفاده قرار می گیرد که داده ها پیچیده باشد اما در انجام فرایند مورد نظر فقط اجزای خاص ان لازم باشد.

SVD کامل در مقابل کاهش یافته یا اقتصادی

در چارچوبی که داریم ابعاد ماتریس های U و ∑ و V به شرح زیر است:

$$U \in R^{n*m} \qquad \sum \in R^{m*m} \qquad V \in R^{m*m}$$

این به این دلیل است که در بسیاری از موارد با ماتریس n*m ، U یک ماتریس مربعی نیست ، یک تجزیه کم ارزش ، اقتصاد یا کاهش یافته در نظر گرفته می شود.

در صورت تجزیه کامل مقدار منفرد A ، ستون های متعامد n-m (به گونه ای که خصوصیات لازم را در خود داشته باشند) به ماتریس U اضافه می شود. علاوه بر این ، برای ثابت نگه داشتن ضرب ، n-m ردیف اضافی به ماتریس ∑ اضافه می شود. در بیشتر سناریوها ، این ردیف ها همه از صفر تشکیل شده اند تا هنگام ضرب با ستون های مربوط به U باعث بی اثر شوند.

در این حالت ، یک تجزیه کامل با ارزش منفرد هیچ اطلاعات مفید اضافی به پروژه نمی دهد و باعث محاسبات بسیار کندتر و طولانی تر می شود. به همین دلیل ، از تجزیه ارزش منحصر به فرد کاهش یافته یا نازک استفاده خواهد شد.

قضیه :

فرض کنید A یک ماتریس n * m باشد. سپس بردارهای منفرد سمت چپ (U) A ، در تجزیه ارزش واحد آن(SVD) ، بردارهای ویژه AA^T هستند.

با انجام تجزیه مقدار واحد روی ماتریس P و با استفاده از قضیه بالا ، آخرین مقادیر لازم برای الگوریتم را می توان یافت (بردارهای ویژه ، ماتریس کوواریانس).

حال در پروژه ما بعد از بدست اوردن مقادیر لازم ماتریس حاصل از تصویری که می خواهیم تشخیص چهره روی ان انجام بدهیم در بردارهای منفرد سمت چپ A ضرب کرده همچنین همین کار را با ماتریس عکس های train هم انجام می دهیم سپس این از هر ستون این ماتریس (در واقع یعنی از هر عکس) مقدار بدست امده از حاصل ضرب ماتریس test در ماتریس U را کم می کنیم و هر عکسی که مقدار ان کوچکتر بود به عنوان خروجی چاپ می کنیم .

```
test_x = np.empty(shape=(u.shape[0], u.shape[1]), dtype=np.int8)
print(test_x.shape)

for col in range(u.shape[1]):
    test_x[:, col] = img[:, 0] * u[:, col]

dot_test = np.array(test_x, dtype='int8').flatten()
```

و برای داده های train :

```
for i in range(images.shape[1]):

    for c in range(u.shape[1]):
        temp[:, c] = images[:, i] * u[:, c]

    tempF = np.array(temp, dtype='int8').flatten()
    dot_train[:, i] = tempF[:]
```

و در نهایت:

```
for col in range(u.shape[1]):
    sub[:, col] = dot_train[:, col] - dot_test[:]
```

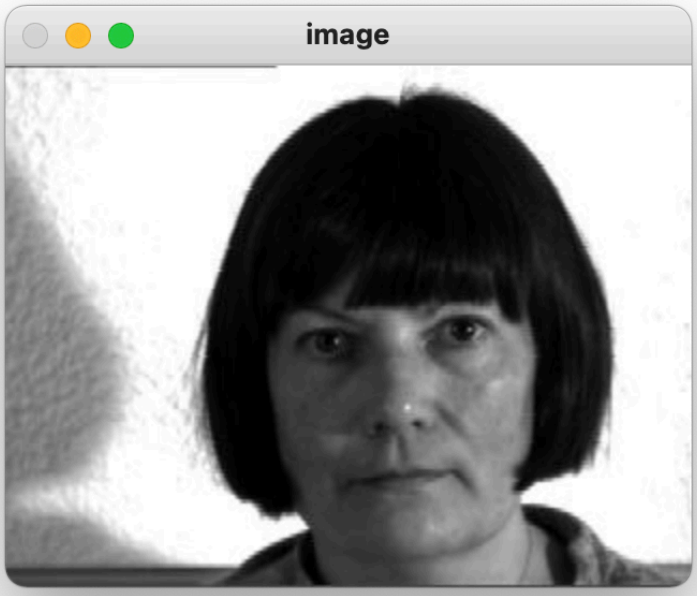
و برای پیدا کردن min برای راحتی کار ابتدا norm هر ستون ماتریس را می گیریم :

```
answer = np.empty(shape=(u.shape[1],))

for c in range(sub.shape[1]):
    answer[c] = np.linalg.norm(sub[:, c])
```

مثال :

ورودی:



خروجی:

The predicted face is: subject11.normal.jpg