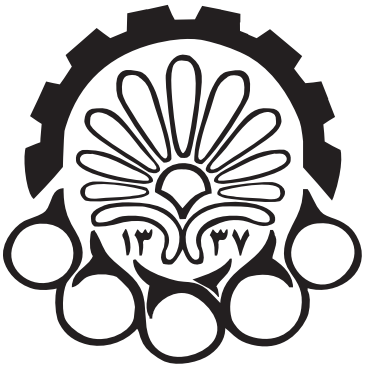


دانشگاه صنعتی امیرکبیر

(پلی‌تکنیک تهران)

دانشکده ریاضی و علوم کامیوتر

پروژه اختیاری درس جبر خطی عددی

عنوان

تشخیص ارقام دست نویس با SVD

نگارش

شیرین محمدی-۹۷۱۲۰۴۶

سروش آریانا-۹۷۱۳۰۳۴

استاد

دکتر دهقان

دی ۱۴۰۱

چکیده

در عصر دیجیتال امروز، تشخیص ارقام دست‌نویس نقش مهمی در پردازش اطلاعات بازی میکند. از کاربرد های آن می‌توان به شناسایی پلاک، خدمات مرتب سازی کاغذ پستی، اتوماسیون اسناد قدیمی در کتابخانه ها و بانک ها اشاره کرد. تمامی این حوزه های با پایگاه داده های بزرگ سروکار دارند و در نتیجه به دقت شناسایی بالا، پیچیدگی محاسباتی کم و عملکرد ثابت شناسایی ناز دارند. این یک کار دشوار برای ماشین است زیرا ارقام دست‌نویس کامل نیستند و می‌توانند به اشکال و اندازه های مختلف نوشته شوند. سیستم تشخیص ارقام دست نویس راهی برای مقابله با این موضوع است،مسئله ای که از تصویر یک رقم استفاده می‌کند و رقم موجود در تصویر را تشخیص می‌دهد. در واقع هدف از سیستم تشخیص ارقام دست نویس تبدیل حروف دست‌نویس به فرمت های قابل خواندن توسط ماشین است. مسئله ما در واقع یک مسئله دسته بندی است .

کلمات کلیدی:

دسته بندی، تجزیه مقدار منفرد، بردار منفرد،دیتاست، دیتا فریم، دقت سنجی، متعامد یکه

فهرست مطالب

فهرست تصاویر

فصل ۱

مقدمه

یکی از مهمترین موضوعات در جبر خطی، روش تجزیه مقادیر تکین SVD گریزان بودن SVD از این واقعیت ناشی می شود که در حالی که این روش به ریاضیات و جبر خطی زیادی برای درک آن نیاز دارد، کاربردهای عملی اغلب نادیده گرفته می شود. افراد زیادی وجود دارند که فکر می کنند مفهوم SVD را درک می کنند، اما در واقع نمی دانند. این فقط یک تکنیک کاهش ابعاد نیست: اساسا، جادوی SVD این است که هر ماتریس A را می توان به عنوان مجموع ماتریس های رتبه ۱ نوشت! که این موضوع را بعدا بیشتر بررسی می‌کنیم.

یادآوری SVD:

رابطه اصلی SVD به شکل زیر است:

$$A_{m\times n} = U_{m\times m} S_{m\times n} V^T_{n\times n} \tag{۱.۱}$$

بطوریکه U و V ماتریس های متعامد هستند و S نیز ماتریس قطری است.

بصورت جزئی تر:

$$A = USV^T = [u_۱ ... u_n] \begin{bmatrix} \sigma_۱ & & \\ & \dots & \\ & & \sigma_n \end{bmatrix} \begin{bmatrix} v_۱^T \\ \dots \\ v_n^T \end{bmatrix} = \Sigma_{i=۱}^{n} \sigma_i u_i v_i^T \tag{۲.۱}$$

که ادعای قبلی ما را که میگفتیم هر ماتریس A را میتوان بصورت مجموع ماتریس های $rank$۱ بنویسیم را نشان میدهد. از ویژگی های مفید SVD

میتوان به موارد زیر اشاره کرد:

۱. ماتریس های U و V به ترتیب از بردارهای ویژه AA^T و A^TA ساخته می شوند.

۲. هر ماتریسی دارای تجزیه SVD است. این به این دلیل است که ماتریس‌های AA^T و A^TA دارای ویژگی خاصی هستند (در میان سایر موارد): آنها حداقل نیمه معین مثبت هستند (به این معنی که مقادیر ویژه آنها مثبت یا صفر هستند).

۳. ماتریس S شامل ریشه های مربع مقادیر ویژه مثبت است. به این مقادیر ،مقادیر منفرد نیز می گویند.
۴. بردار های $u^1, u^2, ...$ را بردار های منفرد چپ نیز مینامند. و این بردار ها تشکیل یک پایه متعامد یکه میدهند. متقابلا $v^1, v^2, ...$ را بردار های منفرد راست میگویند.
۵. $rank$ ماتریس A برابر با تعداد مقادیر نامنفرد غیر صفر ماتریس S می‌باشد.
۶. راه بهینه برای تقریب ماتریس A به رنک پایین تر k این است که SVD را روی A اعمال کنید و اولین بردار های پایه ای k تایی با بالاترین k مقادیر منفرد را انتخاب کنید.

فصل ۲

کتابخانه های مورد نیاز

ابتدا به بررسی کتابخانه های مورد نیاز برای پیاده سازی برنامه تشخیص ارقام دست‌نویس میپردازیم.

۱. کتابخانه $NumPy$ جهت کار با آرایه ها و ماتریس ها

۲.کتابخانه $Pandas$ برای خواندن داده ها و ایجاد دیتا فریم ها.

۳. کتابخانه $matplotlib$ برای رسم نمودار

۴. کتابخانه $scipy$ برای محاسبه تجزیه مقادیر منفرد و نرم

۵. کتابخانه $sklearn$ برای دقت سنجی

۶. کتابخانه های $h5py$ و os برای خواندن داده ها به فرمت مناسب

فصل ۳

مراحل پیاده سازی و آموزش

در اینجا ما از دیتا ست $kaggle$ در پیاده سازی استفاده می‌کنیم. این دیتاست شامل ۷۲۹۱ داده آموزش و ۲۰۰۷ داده تست است که اعداد ۰ تا ۹ را در ۱۶ × ۱۶ پیکسل خاکستری نشان داده است.
در بخش کد زیر داده ها بصورتی درون برنامه لود شده اند که هماهنگ با ابعاد در بخش مقدمه باشند. ستون های x_{train} و x_{test} ارقام را بصورت بردار در خود دارند که به آرایه هایی به اندازه ۲۵۶ مسطح شده است. از طرف دیگر y_{train} و y_{test} بردار های ردیفی هستند که شامل کلاس های واقعی برای هر رقم هستند.

In [128]:
```
# function to read data
def load_data_set():
    with h5py.File(os.path.join(os.getcwd(), 'usps.h5'), 'r') as hf:
        train = hf.get('train')
        test = hf.get('test')

        return pd.DataFrame(train.get('data')[:]).T,\
               pd.DataFrame(train.get('target')[:]).T,\
               pd.DataFrame(test.get('data')[:]).T,\
               pd.DataFrame(test.get('target')[:]).T

    return x_train, y_train, x_test, y_test
```

In [136]:
```
# Get shape of our data
x_train, y_train, x_test, y_test = load_data_set()

print(x_train.shape) #(256, 7291)
print(y_train.shape) #(1, 7291)
print(x_test.shape) # (256, 2007)
print(y_test.shape) #(1, 2007)
```

```
(256, 7291)
(1, 7291)
(256, 2007)
(1, 2007)
```

شکل ۱.۳: خواندن داده ها

می‌توان مراحل دسته بندی ارقام را بصورت زیر بیان کرد:

۱. دیتا فریم x_{train} را به ۱۰ ماتریس بصورت ستونی تقسیم میکنیم یک ماتریس به ازای هر رقم. این ماتریس ها همان ماتریس های A هستند که قبل تر به آن اشاره کردیم. میخواهیم SVD را جداگانه برای هر ماتریس اعمال می‌کنیم. به عنوان مثال A_0 فقط شامل ارقام ۰ است و شکل آن بصورت (۲۵۶،۱۱۹۴) می‌باشد به این معنی که دیتا ست ما شامل ۱۱۹۴ رقم صفر است.

```
In [138]: # Separate matrices correspond to each digit
digits_matrix ={}
for i in range(10):
    digits_matrix.update({f"A{i}": x_train.loc[:,list(y_train.loc[0,:]==i)]})

for digit in range(10):
    print(f"number of {digit}'s are: {digits_matrix['A'+str(digit)].shape[1]}")

number of 0's are: 1194
number of 1's are: 1005
number of 2's are: 731
number of 3's are: 658
number of 4's are: 652
number of 5's are: 556
number of 6's are: 664
number of 7's are: 645
number of 8's are: 542
number of 9's are: 644
```

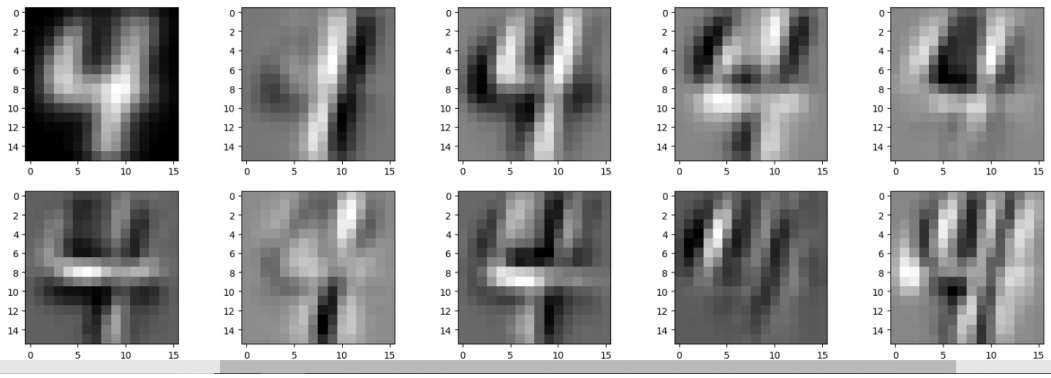
شکل ۲.۳: تعداد ارقام در دیتاست

۲. حال SVD را روی تمام ماتریس های $[A_0, A_1, ..., A_9]$ اعمال می‌کنیم. برای هر ماتریس A ماتریس های S, V و U متناظر با آن را ذخیره می‌کنیم.بیشتر کار ما با ماتریس U می‌باشد. این کار نیز با استفاده از کتابخانه ذکر شده کار پیچیده ای نیست. همچنین در قسمت بعد میتوانیم مشاهده کنیم که این ماتریس ها چه اطلاعاتی را در خود ذخیره کرده اند. شکل ۳.۳ ۱۰ بردار اول منفرد سمت چپ $[u_1, u_2, ..., u_{10}]$ را به عنوان تصویر نمایش می دهد (از ۲۵۶). همه آنها رقم ۴ را نشان می دهند که اولین (بردار u_1) واضح ترین رقم است.

اولین بردار منفرد سمت چپ مقدار ویژگی ذاتی ماتریس $A4$ را نشان می دهد. در واقع، در شکل ۳.۳ ، اولین بردار مفرد u_1 شبیه رقم ۴ است، و بردارهای منفرد چپ مهمترین تغییرات مجموعه آموزشی در اطراف u_1 را نشان می دهند.

In [139]:
```
# calculate SVD of each matrix
u_s, s_s, v_t_s = {}, {}, {}
for i in range(10):
    u, s, v_t = svd(digits_matrix['A'+str(i)], full_matrices=False)
    u_s.update({f"u{i}":u})
    s_s.update({f"s{i}":s})
    v_t_s.update({f"v_t{i}":v_t})
```

In [140]:
```
# visualize columns of u{target_digit}
# target_digit in [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]

plt.figure(figsize=(20,10))
columns = 5
target_digit = '4'
for i in range(10):
    plt.subplot(10 // columns + 1, columns, i + 1)
    plt.imshow(u_s[f"u{target_digit}"][:,i].reshape(16,16), cmap='binary')
```



شکل ۳.۳: ۱۰ بردار اول منفرد ماتریس $U4$

فصل ۴

تست و نتیجه گیری

هر ماتریس داده A ، که با یک رقم نشان داده می شود، دارای یک "ویژگی متمایز" است. این تمایز در چند بردار اول منفرد سمت چپ $(u_۱, u_۲, ...)$ منعکس شده است. از آنجایی که این بردارهای ویژه اساساً بردارهای پایه هستند، اگر یک رقم مجهول را بتوان با مبنای یک رقم دیگر (مثلا رقم ۴) بهتر تقریب زد، می‌توان فرض کرد که رقم مجهول به عنوان آن رقم (به عنوان ۳) طبقه‌بندی می‌شود.
یک رقم مجهول را در نظر بگیرید که با (۱۲۵۶) بردار به نام z نمایش داده می‌شود. همچنین مجموعه بردار های مقادیر منفرد چپ $[u۱, u۲, ..., uk]$ یطوریکه هر مجموعه ماتریس رقم متناظر با آن را نشان می‌دهد. در اینجا اندیس k میباشد و نه n بدین معنی که ما با تمام بردار ویژه های منفرد کار نداریم. باید بصورت زیر عمل کنیم:

$$min_{x_i} \parallel z - \Sigma_{i=۱}^{k} x_i U_i \parallel = min_x \parallel z - U_k x \parallel \quad (۱.۴)$$

جواب مسئله کمترین مربعات برابر است با:

$$x = (U_k^T U_k)^{-۱} U_k^T = U) k^T z \quad (۲.۴)$$

توجه کنید که U متعامد است. پس نرم بردار باقیمانده بصورت زیر میشود:

$$\parallel (I - U_k U_k^T) z \parallel \quad (۳.۴)$$

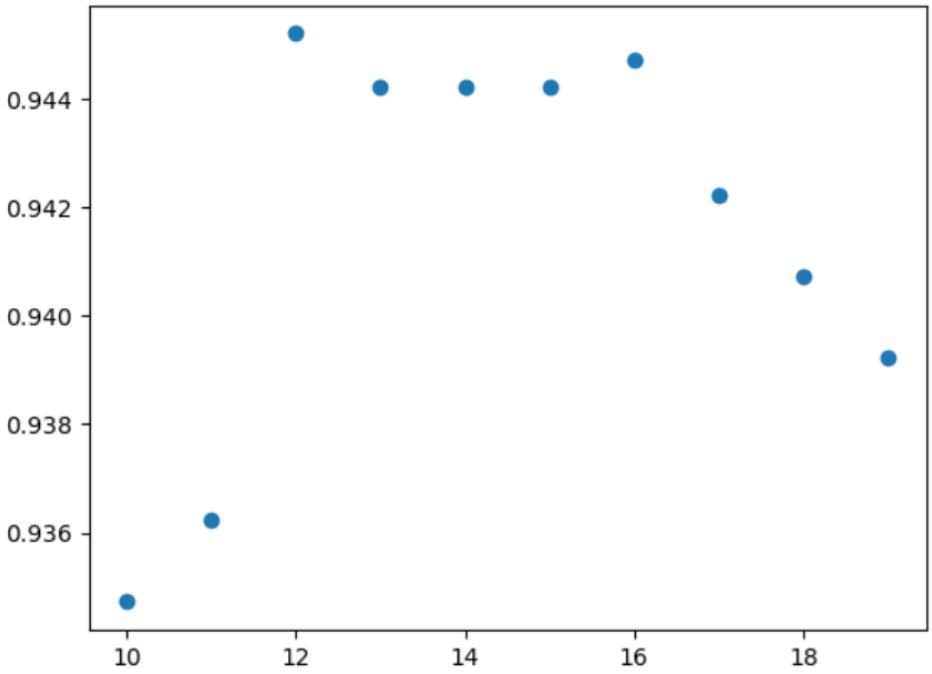
اکنون با استفاده از معادله آخر بخش کد زیر را مینویسیم و به ازای k های مختلف دقت سنجی را انجام میدهیم. حال با استفاده از نمودار زیر متوجه میشویم که زمانیکه $k = ۱۲$ است الگوریتم ما بهترین عملکرد را دارد.

In [144]:
```
# Calculate accuracy score correspond to each k

I = np.eye(x_test.shape[0])
k_s =np.arange(10, 20)
len_test=x_test.shape[1]
predictions=np.empty((y_test.shape[1],0), dtype = int)

for t in list(k_s):
    prediction = []
    for i in range(len_test):
        print(f"t: {t}/{k_s[len(k_s) - 1]}, i: {i}/{len_test}")
        residuals = []
        for j in range(10):
            u=u_s[f"u{j}"][:,0:t]
            res=norm( np.dot(I-np.dot(u,u.T), x_test[i]  ))
            residuals.append(res)
        index_min = np.argmin(residuals)
        prediction.append(index_min)

    prediction=np.array(prediction)
    predictions=np.hstack((predictions,prediction.reshape(-1,1)))
scores=[]

for i in range(len(k_s)):
    score=accuracy_score(y_test.loc[0,:],predictions[:,i])
    scores.append(score)
data={"k":list(k_s), "accuracy_score":scores}
df=pd.DataFrame(data)
```

```
t: 9/19, i: 716/2007
t: 9/19, i: 717/2007
t: 9/19, i: 718/2007
t: 9/19, i: 719/2007
t: 9/19, i: 720/2007
t: 9/19, i: 721/2007
t: 9/19, i: 722/2007
t: 9/19, i: 723/2007
t: 9/19, i: 724/2007
t: 9/19, i: 725/2007
t: 9/19, i: 726/2007
t: 9/19, i: 727/2007
t: 9/19, i: 728/2007
t: 9/19, i: 729/2007
t: 9/19, i: 730/2007
```

شکل ۱.۴: دقت سنجی

```
In [145]: plt.scatter(x=df['k'], y=df['accuracy_score'])
Out[145]: <matplotlib.collections.PathCollection at 0x7fa998bee490>
```




```
In [ ]: # We have the most accuracy when k = 12
```

شکل ۲.۴: نمودار دقت بر اساس k

فهرست مراجع

[1] How to Use Singular Value Decomposition (SVD) for Image Classification in Python(link)

[2] Classification using SVD(link)

[3] Github (link)