دانشگاه صنعتی امیرکبیر
(پلی تکنیک تهران)

دانشکده ریاضی و علوم کامپیوتر
استاد درس: دکتر مهدی دهقان
پاییز ۱۴۰۱

کاربرد تجزیه SVD در تشخیص اعداد دست نویس

مهرداد اکثری مهابادی
۹۹۱۲۰۱۰
جبر خطی عددی

فهرست مطالب

۱ مقدمه

متود های مختلفی در تشخیص خودکار نوشته های دست نویس[۱] از دهه ۱۹۵۰ تا کنون مورد بررسی قرار گرفته اند. زمینه های تحقیقاتی مختلفی از جمله تایید امضا، هویت، تشخیص اسناد جعلی، OCR [۲] در پردازش دست نوشته وجود دارد.
در این مقاله به بررسی کاربرد تجزیه مقادیر تکین در پردازش اعداد دست نویس می پردازیم. ابتدا ضمن مروری گذرا بر تجزیه SVD [۳]، برخی مفاهیم مورد استفاده در بخش های بعد مانند تقریب ماتریس [۴] را توضیح می دهیم. در بخش دوم مقاله با پیاده سازی آزمایشات عددی الگوریم ارقام ویژه [۵] را بیان می کنیم و عملکرد آن را به ازای پارامتر های متفاوت می سنجیم.

۲ آشنایی با SVD

در این بخش انتظار می رود که خواننده با مفاهیم مقدماتی در جبرخطی آشنا باشد. در صورتی که با تجزیه مقادیر تکین آشنایی دارید، می توانید از این قسمت عبور کنید. برای مطالعه بیشتر می توانید به فصل ۶ [۱] مراجعه کنید.

۱.۲ تعریف

برای هر ماتریس $\mathrm{m} \times \mathrm{n}$ مانند $\mathbf{A}$ که $\mathrm{m} \geq \mathrm{n}$ تجزیه ای به صورت زیر وجود دارد

$$\mathbf{A} = \mathbf{U} \begin{pmatrix} \Sigma \\ 0 \end{pmatrix} \mathbf{V}^T$$

که $\mathbf{U} \in \mathbb{R}^{m \times m}$ و $\mathbf{V} \in \mathbb{R}^{n \times n}$ متعامد و $\mathbf{\Sigma} \in \mathbb{R}^{m \times n}$ قطری است به طوری که

$$\Sigma = diag(\sigma_1, \sigma_2, \ldots, \sigma_n)$$
$$\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_n$$

به تجزیه بالا تجزیه SVD [۶] گفته میشود. به ستون های $\mathbf{U}$ و $\mathbf{V}$ بردار های ویژه [۷] و به σ_i مقادیر ویژه [۸] گفته می شود. دقت کنید که شرط $\mathrm{m} \geq \mathrm{n}$ باعث ایجاد محدودیتی در قضیه نمی شود. اگر این شرط برقرار نباشد همچنان رابطه بالا برای $\mathbf{A}^T$ وجود دارد.
اثبات های متفاوتی برای قضیه بالا وجود دارد. برای دیدن اثباتی استقرایی به بخش ۶.۱ [۱] مراجعه کنید.

[1] automatic handwriting recognition
[2] optical character recognition
[3] singular value decomposition
[4] matrix approximation
[5] eigen digits
[6] singular value decomposition
[7] singualr vectors
[8] singular values

۲.۲ انواع

تجزیه ای که در تعریف بالا ارائه شد فرم کامل [۹] تجزیه SVD بود. تجزیه مقادیر تکین را به فرم های معادل متفاوتی میتوان نوشت. برای راحتی به معرفی چند فرم مختلف که در ادامه از آنها استفاده شده می پردازیم.

۱.۲.۲ فرم سبک

$$\boxed{A} = \boxed{U} \times \boxed{\Sigma_1} \boxed{V^T}$$

از آنجایی که $\Sigma_1 = \begin{pmatrix} \Sigma \\ 0 \end{pmatrix}$ بار قرار دادن $\mathbf{U} = (\mathbf{U_1}, \mathbf{U_2})$ که $\mathbf{U_1}$ ماتریسی $m \times n$ است، میتوان فرم بالا را به صورت زیر نوشت

$$\boxed{A} = \boxed{U_1} \times \boxed{\Sigma} \boxed{V^T}$$

به این فرم تجزیه مقادیر تکین سبک[۱۰] گفته می شود.
توجه کنید که از آنجایی که $\mathbf{V}$ متعامد است با ضرب طرفین در $\mathbf{V}$ داریم

$$\mathbf{A}v_i = u_i\sigma_i \quad i = 0, 1, \ldots, n \tag{۱}$$

به طور مشابه با گرفتن ترانهاده و ضرب در $\mathbf{U}^T$ داریم

$$\mathbf{A^T}u_i = v_i\sigma_i \quad i = 0, 1, \ldots, n \tag{۲}$$

با ضرب (۱) در $\mathbf{A^T}$ داریم

$$\begin{aligned} \mathbf{A^T A}v_i &= \mathbf{A^T}u_i\sigma_i \\ &= \sigma_i^2 v_i \end{aligned} \tag{۳}$$

رابطه بالا نشان می دهد که v_i بردار ویژه و σ_i^2 مقادیر ویژه ماتریس $\mathbf{AA^T}$ هستند.
به طور مشابه با ضرب (۲) در $\mathbf{A}$ داریم :

$$\begin{aligned} \mathbf{AA^T}u_i &= \mathbf{A}v_i\sigma_i \\ &= \sigma_i^2 u_i \end{aligned} \tag{۴}$$

که یعنی u_i بردار ویژه و σ_i^2 همان مقادیر ویژه ماتریس $\mathbf{A^T A}$ هستند.

[9] Full SVD
[10] thin SVD

۲.۲.۲ فرم فشرده

می دانیم با ضرب یک ماتریس در ماتریسی مکوس پذیر رنک ماتریس تغییری نمی کند. در نتیجه با توجه به اینکه $\mathbf{U_1}$و $\mathbf{V}$ متعامد هستند می توان گفت

$$\mathbf{Rank(A)} = \mathbf{Rank(U_1 \Sigma V)} = \mathbf{Rank(\Sigma)}$$

در نتیجه رنک ماتریس $\mathbf{A}$برابر با تعداد عناصر قطری ناصفر $\mathbf{\Sigma}$ است. فرض کنید $\mathbf{Rank(A)} = \mathbf{r}$ باشد. در اینصورت می توان تجزیه مقادیر تکین را به فرم فشرده زیر نوشت

$$\mathbf{A} = \mathbf{U_{m\times r} \Sigma_{r\times r} V^T_{r\times n}}$$

به این فرم تجزیه مقادیر تکین فشرده[۱۱] گفته میشود. دقت کنید که تجزیه فوق در صورتی که $r \ll min\{m, n\}$ باشد باعث کاهش حجم محاسبات و فضای اشغال شده دیسک می شود.

۳.۲.۲ فرم ناقص

در صورتی که فقط از $\mathbf{t}$ مقدار ویژه اول ماتریس $\mathbf{A}$ استفاده کنیم، می توان نوشت

$$\mathbf{A} = \mathbf{U_{m\times t} \Sigma_{t\times t} V^T_{t\times n}}$$

به این فرم تجزیه مقادیر منفرد ناقص [۱۲] گفته می شود. درباره کاربرد این فرم در بخش های بعد توضیح خواهیم داد.

۴.۲.۲ فرم ضرب خارجی

فرم سبک را می توان به صورت دیگری نیز بازنویسی کرد

$$\begin{aligned}
\mathbf{A = U_1 \Sigma V^T} &= (\mathbf{u_1, u_2, \dots, u_n}) \begin{pmatrix} \sigma_1 & & & \\ & \sigma_2 & & \\ & & \ddots & \\ & & & \sigma_n \end{pmatrix} \begin{pmatrix} v_1^T \\ v_2^T \\ \dots \\ v_n^T \end{pmatrix} \\
&= (\mathbf{u_1, u_2, \dots, u_n}) \begin{pmatrix} \sigma_1 v_1^T \\ \sigma_2 v_2^T \\ \dots \\ \sigma_n v_n^T \end{pmatrix} \\
&= \sum_{i=0}^{n} \sigma_i u_i v_i^T
\end{aligned}$$

به این فرم، فرم ضرب خارجی گفته [۱۳] می شود. در این فرم ماتریس A به صورت جمع n ماتریس $m \times n$ نوشته شده است که هر ماتریسی از مرتبه یک می باشد. در بخش تقریب ماتریس از این فرم استفاده خواهیم کرد.

[11] compact SVD
[12] truncated SVD
[13] outer product form

۳.۲ زیرفضا های بنیادی

- پایه های برد

$$\mathbf{R}(\mathbf{A}) = \{y : y = Ax\}$$

رابطه بالا برد ماتریس $\mathbf{A}$ است. اگر $\mathbf{Rank}(\mathbf{A}) = \mathbf{r}$ با استفاده از فرم ضرب خارجی داریم

$$\mathbf{Ax} = \sum_{\mathbf{i=0}}^{\mathbf{r}} \sigma_{\mathbf{i}} \mathbf{u_i} \mathbf{v_i^T x} = \sum_{\mathbf{i=0}}^{\mathbf{r}} \left(\sigma_{\mathbf{i}} \mathbf{v_i^T x}\right) \mathbf{u_i} = \sum_{\mathbf{i=0}}^{\mathbf{r}} \alpha \mathbf{u_i}$$

در نتیجه هر بردار در برد $\mathbf{A}$ را میتوان به صورت جمع u_i ها نوشت. از آنجایی که گفته شد u_i بردار هایی متعامد و در نتیجه مستقل خطی هستند می توان گفت بردار های ویژه $u_1, u_2, \ldots, u_r$ پایه ای متعامد برای برد ماتریس $\mathbf{A}$ است.

- پایه های پوچی

$$\mathbf{N}(\mathbf{A}) = \{x : Ax = 0\}$$

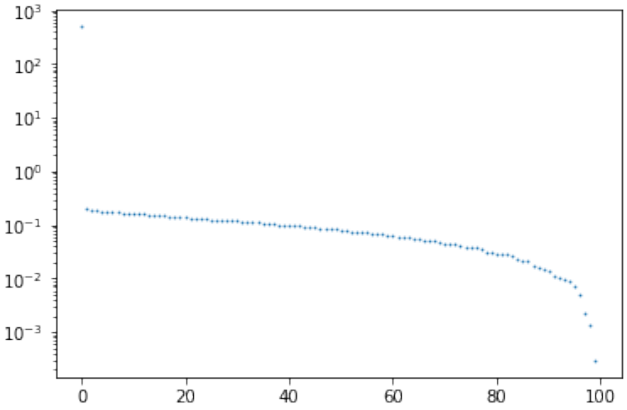
برای هر بردار $z = \sum_{i=r+1}^{n} \beta_i v_i$ با توجه به تعامد v_i ها داریم

$$\mathbf{Ax} = \sum_{\mathbf{i=0}}^{\mathbf{z}} \sigma_{\mathbf{i}} \mathbf{u_i} \mathbf{v_i^T} \left(\sum_{i=r+1}^{n} \beta_i v_i\right) = 0$$

از آن جایی که بعد فضای پوچی $n - r$ است بردار های متعامد $v_r + 1, v_r + 2, \ldots, v_n$پایه ای برای این فضا هستند.

۴.۲ تقریب ماتریس

فرض کنید ماتریس $\mathbf{A}$ ماتریسی با رنک پایین به صورت $\mathbf{A} = \mathbf{A_0} + \mathbf{N}$ باشد که $\mathbf{N}$ نویز می باشد. در این صورت اگر مقدار $\mathbf{N}$در برابر $\mathbf{A_0}$ کوچک باشد، مقادیر ویژه $\mathbf{A}$ رفتاری به شکل زیر دارند



شکل ۱: مقادیر ویژه یک ماتریس بعد از اعمال نویز گاوسی با میانگین صفر و واریانس یک. ماتریس $\mathbf{A_0}$ ماتریسی با ستون های یکسان 10×10 و درایه های صحیح بین ۰ تا ۱۰ بوده.

می توان دید که مقادیر ویژه مقداری نزدیک به صفر دارند. به تعداد مقادیر ویژه بزرگ [۱۴] رنک عددی یک ماتریس گفته می شود. در مسائل عملی معمولا ماتریس نویز را نداریم. اگر بتوانیم با مشاهده مقادیر ویژه رنک عددی یک ماتریس را حدس بزنیم آنگاه می توان $\mathbf{A}$ را به صورت زیر تقریب زد.

$$\mathbf{A} = \sum_{i=0}^{n} \sigma_i u_i v_i^T \approx \sum_{i=0}^{k} \sigma_i u_i v_i^T$$

توانستیم ماتریس $\mathbf{A}$ را با ماتریسی با رنک پایین تر تقریب بزنیم. این کار علاوه بر حذف نویز مزایای دیگری هم دارد. حذف مقادیر ویژه نزدیک به صفر باعث پایدارتر شدن جواب مسائل بد حالت [۱۵] می شود. بعلاوه همانطور که در ادامه خواهیم دید از این روش می توان در فشرده سازی اطلاعات هم استفاده کرد.

[۱۴] در اینجا به تعریفی نسبی از بزرگ بسنده می کنیم

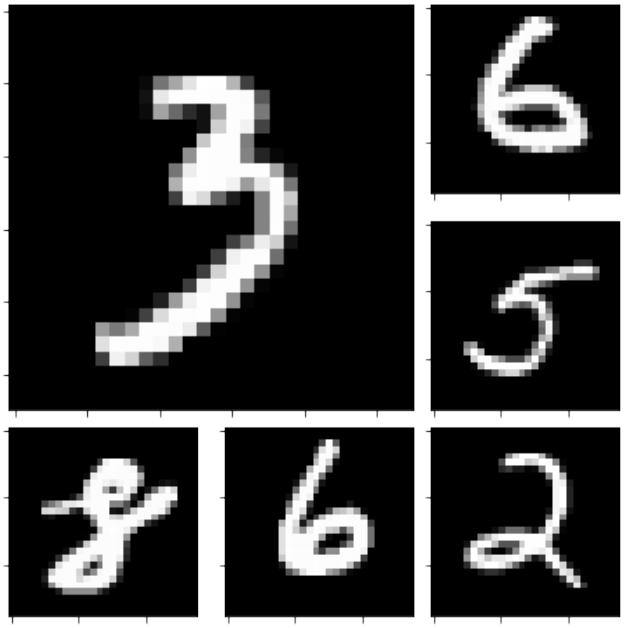
[15] ill-conditioned

۳ تشخیص ارقام دست نویس

در بخش قبل با تجزیه مقادیر تکین آشنا شدیم. در این بخش پس از بررسی داده و ارائه یک راه حل خوشه بندی[۱۶] به بررسی الگوریتم ارقام ویژه[۱۷] می پردازیم. مروری کامل بر متود های مختلف در [۲] داده شده.

۱.۳ پایگاه داده

در این مقاله از دیتاست MNIST [۱۸] برای طبقه بندی ارقام دست نویس استفاده می کنیم. این دیتاست شامل ۶۰۰۰۰ داده آموزش [۱۹] و ۱۰۰۰۰ داده تست [۲۰] است. هر عکس در قالب یک آرایه ۲۸ در ۲۸ به ما داده شده. در مرحله پیش پردازش با صاف [۲۱] کردن عکس ها آنها را به صورت یک بردار با بعد ۷۸۴ نمایش می دهیم.
هدف ما استفاده از داده های آموزشی برا طبقه بندی داده های تست است.



شکل ۲: نمونه ای رندوم از داده های آموزش

[16] clustrering
[17] eigen digits
[18] Modified National Institute of Standards and Technology
[19] train
[20] test
[21] flat

۲.۳ یک راه حل ساده

اگر هر عکس را برداری در فضای ۷۸۴ بعدی در نظر بگیریم انتظار داریم که اعداد هر دسته تشکیل یک خوشه [۲۲] دهند. می توانیم این ادعا را محک بزنیم، به این صورت که با گرفتن میانگین [۲۳] ارقام هر دسته باید اشکال قابل تمایزی به دست آوریم.

شکل ۳: مراکز خوشه بندی

می توانیم الگوریتم خوشه بندی را به صورت زیر فرمول بندی کنیم

۱. مرکز ارقام را با استفاده از داده های آموزشی به دست آورید

۲. فاصله هر داده را[۲۴] با مراکز اندازه گیری کنید. داده نامعلوم را به نزدیک ترین خوشه نسبت دهید.

با اجرای این الگوریتم توانستیم داده های تست را با دقت ۸۲.۰۳ طبقه بندی کنیم.

[22] cluster
[23] mean

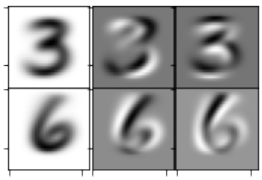
[۲۴] در اینجا منظور فاصله اقلیدسی است.

۳.۳ ارقام ویژه

فرض کنید $\mathbf{A} \in \mathbb{R}^{m \times n}$ ماتریسی باشد که ستون هایش از داده های آموزشی برای یکی از ارقام تشکیل شده. در این صورت ستون های $\mathbf{A}$ زیرفضایی از $\mathbb{R}^m$ را اسپن می کند. انتظار نمی رود که این زیرفضا، زیرفضایی با بعد بالا باشد، چراکه در غیر اینصورت زیرفضا های ارقام متفاوت هم دیگر را می پوشانند و با توجه به خوشه بندی قسمت قبل این امر بعید است. هر ستون در $\mathbf{A}$ یک تصویر را نشان می دهد. از نمایش ضرب خارجی، که در بخش قبل معرفی شد، داریم

$$\mathbf{a_j} = \sum_{i=0}^{m} (\sigma_i v_{ij})\, u_i \approx \sum_{i=0}^{k} (\sigma_i v_{ij})\, u_i \approx \sum_{i=0}^{k} \alpha_i u_i$$

پس می توان هر عکس را به صورت تقریب خطی u_i ها نمایش داد. با توجه به مطالب گفته شده در تقریب ماتریس می دانیم که بردار های متناظر با اولین مقدار ویژه اهمیت ویژه ای در کنترل جهت ماتریس داده دارد. در نتیجه انتظار داریم که با رسم u_1 برداری شبیه به ارقام ماتریس داده داشته باشیم. بعلاوه برای بقیه u_i ها با بزرگ تر شدن i از اهمیت u کاسته می شود. در شکل زیر سه رقم ویژه اول برای سه و شش نمایش داده شده.



شکل ۴: بردار های ویژه برای اعداد ۳ و ۶

به بردار های ویژه، در اینجا ارقام ویژه[۲۵] نیز گفته می شود. برای دیدن بردار ویژه بیشتر به ارقام ویژه به پیوست مراجعه کنید.

۴.۳ الگوریتم ارقام ویژه

برای تقریب هر عکس، z، با استفاده از ارقام ویژه u_k باید $u_1, u_2, \ldots, u_k$ مسئله مینیمم سازی زیر حل شود

$$min_\alpha ||z - U_k \alpha||$$

که $\alpha \in \mathbb{R}^k$ بردار ضرایب می باشد. با توجه به متعامد بودن بردار های ویژه می توان گفت $\alpha = U_k^T z$.
الگوریتم ارقام ویژه را به صورت زیر فرمول بندی می کنیم

۱. تجزیه مقادیر تکین را برای هر رقم با استفاده از داده آموزش به دست می آوریم.

۲. به تعداد مناسب[۲۶] ارقام ویژه از ستون های ماتریس $\mathbf{U}$ انتخاب می کنیم.

۳. هر عکس در داده های تست را با استفاده از پایه های ارقام ویژه برای ارقام مختلف تقریب می زنیم.

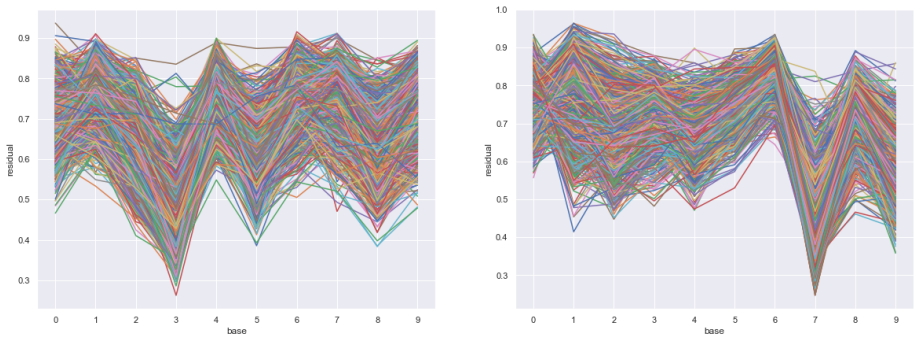
۴. عکس را در دسته ای قرار می دهیم که بهترین تقریب، کمترین خطا، را داشته باشد.

این الگوریتم مربوط به روش SIMCA [۳] است.

[25] eigen digits

[۲۶] راجب تعداد مناسب در ادامه بحث خواهد شد

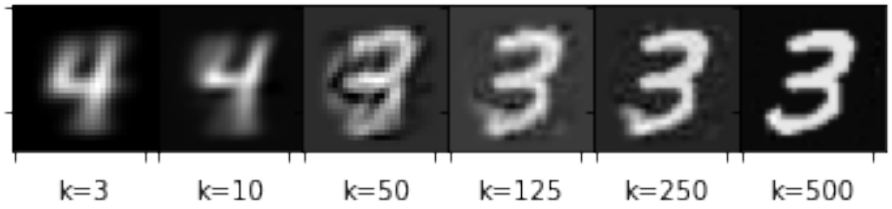
نشان می دهیم الگوریتم بالا می تواند نتایج معقولی را برگرداند. شکل زیر مقدار باقی مانده داده آموزشی برای اعداد ۳ و ۷ را با استفاده از ۱۰ رقم ویژه اول در ایه های مختلف نشان می دهد.



شکل ۵: خطا در پایه های مختلف

در شکل سمت چپ می توان دید که مقدار خطا در پایه ۳ از بقیه پایه ها کمتر است و در نتیجه می توان اعداد را به عدد ۳ نسبت داد. همچنین می توان دید خطا برای اعداد ۳ در پایه ۵ هم مقدار کمتری نسبت به پایه های دگر دارد که بدلیل تشابه عدد ۳ و ۵ است.

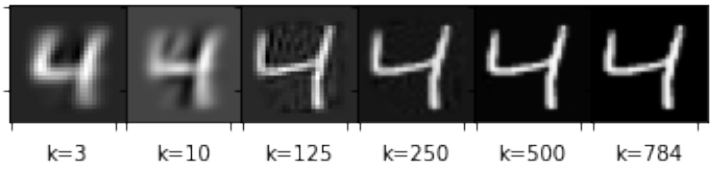
با توجه به شکل قبل می توان هر عدد را در پایه های دگر هم تقریب زد.



شکل ۶: تقریب بردار با پایه های متفاوت

با اینکه خطای باقی مانده برای عدد ۳ در پایه چهار بالاست،اما پایه های چهار توانستند به خوبی ۳ را تقریب بزنند!

بدیهی است که هر چه مقدار k بیشتر باشد تقریب دقیق تری از داده های آموزش خواهیم داشت.

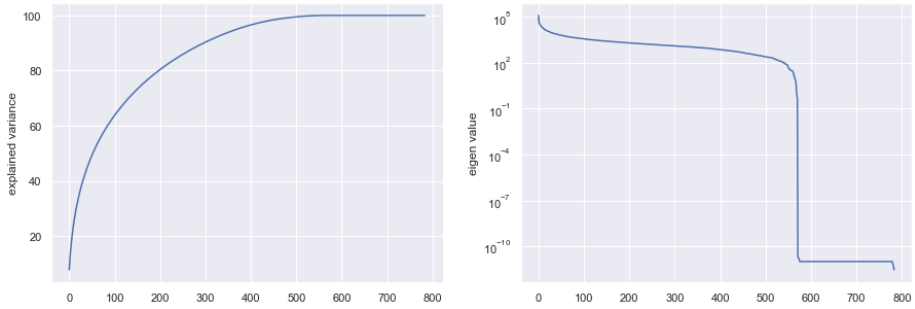


شکل ۷: تقریب بردار برای مقادیر مختلف k

دو رقم آخر تقریبا غیر قابل تمایز هستند. با ۵۰۰ مقدار ویژه به خوبی توانستیم عکس را تقریب بزنیم. در نتیجه میتوان به جای ذخیره فرم اصلی ماتریس، فرم ناقص را ذخیره کرد. هر چه k بزرگتر باشد تقریب دقیق تری از عکس داریم. اما تمایل داریم k را تا جایی که می توانیم کوچک کنیم. می توان با استفاده از واریانس توضیحی [۲۷] مقدار دقت تقریب برای مقادیر مختلف k را به دست آورد. واریانس توضیحی به صورت زیر تعریف می شود.

$$v_k = \frac{\sigma_1+\sigma_2+...+\sigma_k}{\sum_{i=0}^{n}\sigma_i}$$

در شکل زیر واریانس توضیحی و مقادیر ویژه را برای عدد چهار رسم کرده ایم. همانطور که دیده می شود بردار های ویژه ۶۰۰ به بعد عملا نویز هستند و چیز جدیدی به تقریب اضافه نمی کنند.

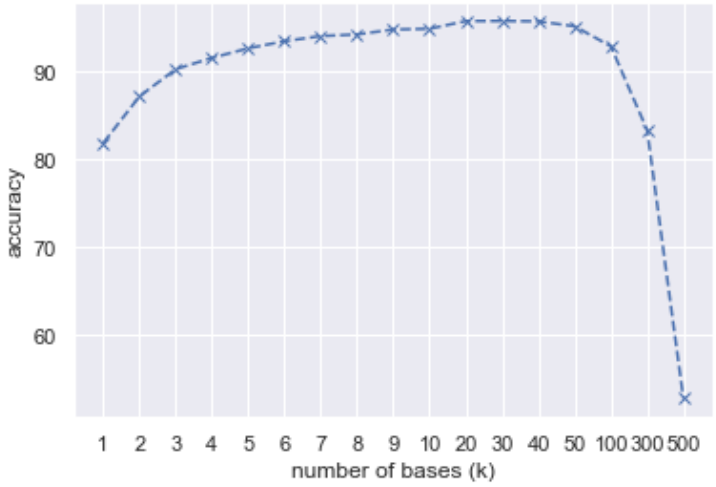


شکل ۸: واریانس توضیحی و مقادیر ویژه

[27] explained variance

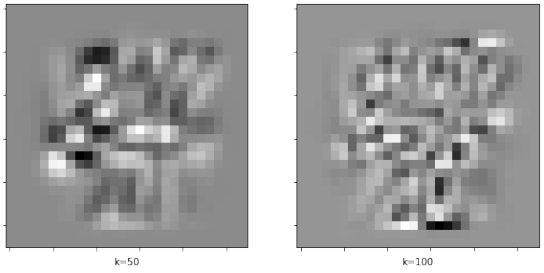
۱.۴.۳ انتخاب پارامتر k

همانطور که دیدم هر چه مقدار k بیشتر باشد،با استفاده می توان تقریب بهتری از داده های آموزشی داشت. اما آیا چنین چیزی برای داده های تست هم برقرار است؟



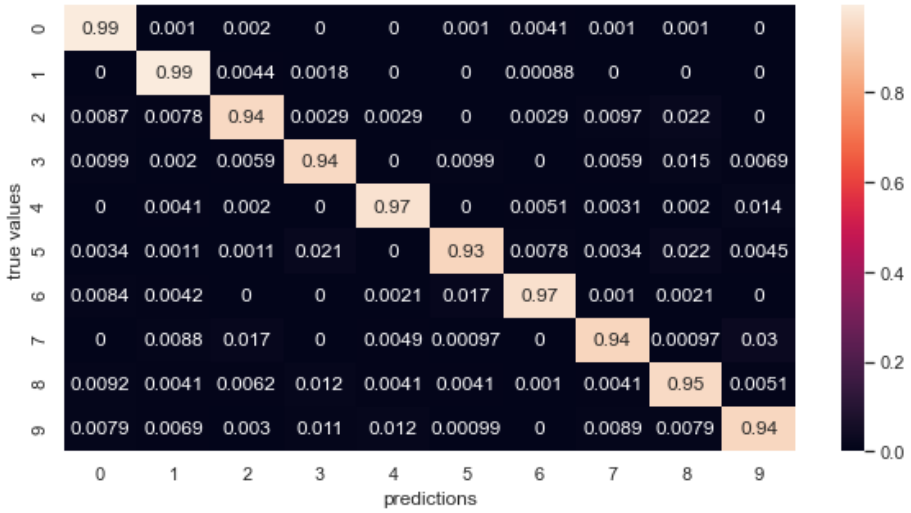
شکل ۹: دقت تقریب داده های تست با تعداد پایه های متغیر

شکل بالا نشان می دهد که چنین چیزی نمی تواند برای داده های تست درست باشد. از آنجایی که پایه های مرتبه بالا از اهمیت کمتری برخوردار هستند، انتظار نمی رود که اطلاعات با عمومیت بالایی را در خود ذخیره کرده باشند و بتوان آنها را به داده های تست تعمیم داد. می توان دید برای $k = 5$ هیچ پیشرفتی نسبت به خوشه بندی قسمت قبل نداشتیم.



شکل ۱۰: پایه های ۴ برای مقادیر بالای k

با $k = 30$ به بهترین دقت رسیدیم. می توان با استفاده از ماتریس درهم ریختگی [۲۸] نقشه گرمایی[۲۹] را برای این مقدار k رسم کرد.



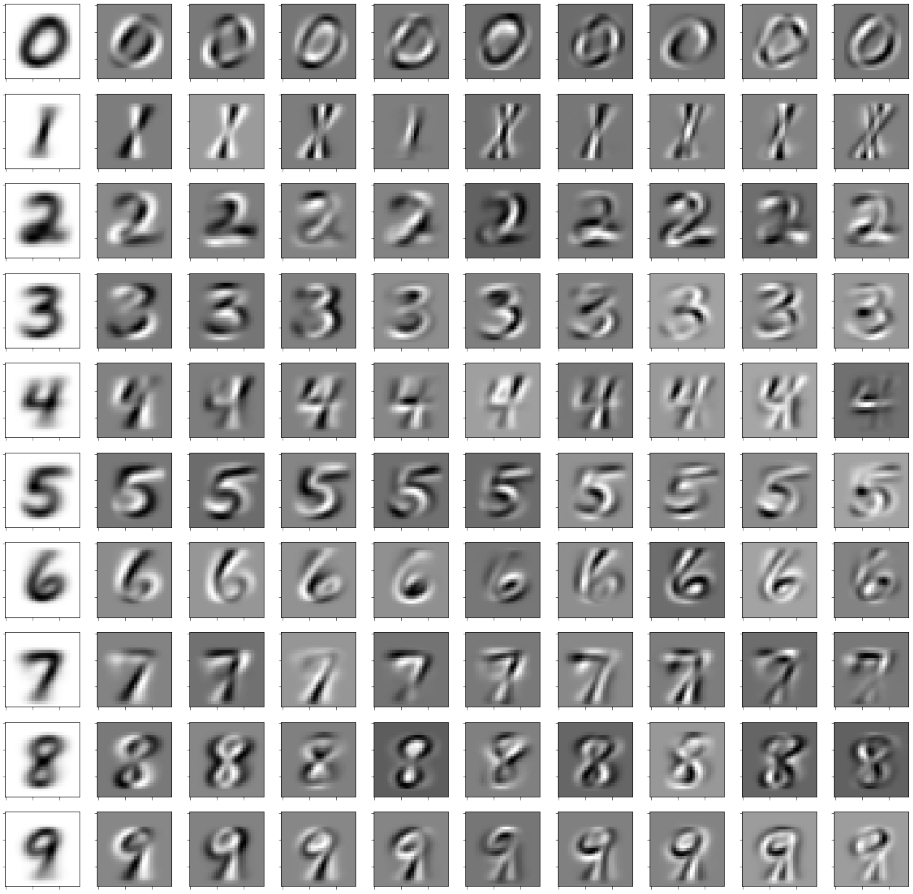
شکل ۱۱: ماتریس درهم ریختگی

با استفاده از شکل بالا می توان دید چه خطا هایی در مدل وجود دارد. به عنوان مثال رقم ۵ بیشتر ۹ و ۳ اشتباه تشخیص داده شده.
برای مطالعه بیشتر روش های تبدیل[۳۰] و استفاده از فاصله تانژانت [۳۱] در دسته بندی ارقام به بخش ۱۰.۳ [۱] مراجعه کنید.

[28] confusion matrix
[29] heatmap
[30] transformation
[31] tangent distance

۴ پیوست

۱.۴ ارقام ویژه



۲.۴ کد جوپیتر

[ ]:
```
from keras.datasets import mnist
import matplotlib.pyplot as plt
import seaborn as sns ; sns.set()
from sklearn import metrics
import numpy as np
from numpy.linalg import svd, norm
from tqdm import tqdm
```

Fig 1

[ ]:
```
A = np.random.randint(0, 10, (100))
print("rank A: ", 1)
A_1 = A + np.random.normal(0, 0.01, (100, 100))
print("rank A after noise: ", np.linalg.matrix_rank(A_1))
ax = plt.gca()
ax.set_yscale('log')
_,e,_ = svd(A_1)
n = e.shape[0]
ax.scatter(np.arange(n), e, 5.0);
```

[ ]:
```
M = 28 * 28
N = 28
K = 10
```

[ ]:
```
(train_x , train_y), (test_x, test_y) = mnist.load_data()
train_x, test_x = train_x.reshape(-1, M), test_x.reshape(-1, M)
```

Fig 2

[ ]:
```
from matplotlib.gridspec import GridSpec
pics = train_x[np.random.choice(train_x.shape[0], 6)]

def format_axes(fig):
    for i, ax in enumerate(fig.axes):
        ax.imshow(pics[i].reshape(N, N), cmap='gray')
        ax.tick_params(labelbottom=False, labelleft=False)

fig = plt.figure(constrained_layout=True)
fig.set_size_inches(5, 5)

gs = GridSpec(3, 3, figure=fig)
ax1 = fig.add_subplot(gs[:2, :2])
ax2 = fig.add_subplot(gs[0, 2])
ax3 = fig.add_subplot(gs[1, 2])
ax4 = fig.add_subplot(gs[2, 2])
ax5 = fig.add_subplot(gs[2, 0])
ax6 = fig.add_subplot(gs[2, 1])
```

```
format_axes(fig)
plt.show()
```

Fig 3

[ ]:
```
fig, ax = plt.subplots(2, 5, sharey=True)
fig.set_size_inches(5, 2)
fig.subplots_adjust(wspace=0, hspace=0)

means = np.zeros(shape=(10, M))
for i in range(10) :
    t = train_x[train_y == i]
    means[i] = np.mean(t, axis=0)
    a = ax[i//5, i%5]
    a.imshow(means[i].reshape(N, N), cmap='gray')
    a.tick_params(labelbottom=False, labelleft=False)
    a.set_xticklabels([])
    a.set_yticklabels([])
```

[ ]:
```
# computing clustering accuracy
pred = np.argmin(norm(means - test_x.reshape(-1, 1 , M), 2, axis=2), axis=1)
print( "accuracy: ", test_accuracy(pred) , "%" )
```

[ ]:
```
bases = np.zeros(shape=(10, M, M))
for i in range(10) :
    t = train_x[train_y == i].T
    u, _, _ = svd(t, full_matrices=False)
    bases[i] = u
```

[ ]:
```
fig, ax = plt.subplots(10, 10, sharex=True, sharey=True)
fig.set_size_inches(20, 20)
fig.subplots_adjust(wspace=0, hspace=0)
for i in range(10) :
    for j in range(K) :
        ax[i, j].imshow(bases[i][:, j].reshape(N, N), cmap='gray')
        ax[i, j].tick_params(labelbottom=False, labelleft=False)
```

Fig 4

[ ]:
```
fig, ax = plt.subplots(2, 3, sharex=True, sharey=True)
fig.set_size_inches(3, 2)
fig.subplots_adjust(wspace=0, hspace=0)
for i, v in enumerate([3,6]):
    for j in range(3) :
        ax[i, j].imshow(bases[v][:, j].reshape(N, N), cmap='gray')
        ax[i, j].tick_params(labelbottom=False, labelleft=False)
        ax[i, j].set_xticklabels([])
```

```
        ax[i, j].set_yticklabels([])
```

Fig 5

[ ]:
```
fig, ax = plt.subplots(1, 2)
fig.set_size_inches(20, 7)
for i, d in enumerate([3, 7]) :
    res = residual(num_bases=10, digit=d)
    ax[i].plot(res.T)
    ax[i].set_xticks(np.arange(10));
    ax[i].set_xlabel("base")
    ax[i].set_ylabel("residual")
```

Fig 6

[ ]:
```
z = train_x[train_y == 3][0]
plt.imshow(z.reshape(N, N), cmap='gray')
```

[ ]:
```
fig, ax = plt.subplots(1, 6, sharex=True, sharey=True)
fig.subplots_adjust(wspace=0, hspace=0)

u = bases[5]
for i, j in enumerate([3, 10, 50, 125, 250, 500]) :
    u_k = u[:, :j]
    a = u_k.T @ z
    ax[i].imshow((u_k @ a).reshape(N, N), cmap='gray')
    ax[i].set_xlabel(f'k={j}')
    ax[i].set_xticklabels([])
    ax[i].set_yticklabels([])
```

Fig 7

[ ]:
```
z = train_x[train_y == 4][0]
plt.imshow(z.reshape(N, N), cmap='gray')
```

[ ]:
```
fig, ax = plt.subplots(1, 6, sharex=True, sharey=True)
fig.subplots_adjust(wspace=0, hspace=0)

u = bases[4]
for i, j in enumerate([3, 10, 125, 250, 500, 784]) :
    u_k = u[:, :j]
    a = u_k.T @ z
    ax[i].imshow((u_k @ a).reshape(N, N), cmap='gray')
    ax[i].set_xlabel(f'k={j}')
    ax[i].set_xticklabels([])
    ax[i].set_yticklabels([])
```

Fig 8

[ ]:
```
fig, ax = plt.subplots(1, 2)
fig.set_size_inches(15, 5)
ax[0].plot(np.cumsum(e) / np.sum(e) * 100)
ax[0].set_ylabel("explained variance")
ax[1].set_yscale('log')
ax[1].set_ylabel("eigen value")
ax[1].plot(np.arange(len(e)), e)
```

[ ]:
```
history = []
b = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10] + [20, 30, 40, 50] + [100, 300, 500]
for K in tqdm(b) :
    res = residual(num_bases=K)
    y = np.argmin(res, axis=1).reshape(-1)
    history += [test_accuracy(y)]
```

[ ]:
```
plt.xticks(np.arange(len(b)), labels=b)
plt.plot(history, 'bx--');
plt.xlabel("number of bases (k)")
plt.ylabel("accuracy")
```

Fig 9

[ ]:
```
history = []
b = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10] + [20, 30, 40, 50] + [100, 300, 500]
for K in tqdm([500, 784]) :
    res = residual(num_bases=K)
    y = np.argmin(res, axis=1).reshape(-1)
    history += [test_accuracy(y)]
```

[ ]:
```
plt.xticks(np.arange(len(b)), b)
plt.plot(history, 'bx--');
```

Fig 10

[ ]:
```
fig, ax = plt.subplots(1, 2)
fig.set_size_inches(10, 7)
for i, b in enumerate([50, 100]) :
    ax[i].imshow(bases[4][:, b].reshape(N, N), cmap="gray") ;
    ax[i].set_xlabel(f'k={b}')
    ax[i].set_xticklabels([])
    ax[i].set_yticklabels([])
```

Fig 11

[ ]:
```
res = residual(num_bases=30)
y = np.argmin(res, axis=1).reshape(-1)
z = metrics.confusion_matrix(test_y, y, normalize='true')
sns.heatmap(z, annot=True)
```

```
plt.rcParams["figure.figsize"] = (10, 5)
plt.xlabel("predictions")
plt.ylabel("true values")
plt.show()
```

[ ]:
```
def test_accuracy(y) :
    return np.sum(y == test_y) / len(test_y) * 100
```

[ ]:
```
def residual(num_bases, digit=None) :
    """
    get_residual returns residual given by :
                    min || z - u @ a || / ||z||
    where `z` is the target vector, `u` is the basis which is
    used to estimate `z`, and `a` are multipliers.
    ---------------
    parameters :
        num_bases : number of bases used to approximate m by n
                    matrix A in SVD decomposition.
        digit : use only digits equal to digit in test set if
                specified, default None
    """
    u = bases[:, :, :num_bases]
    z = test_x if (digit == None) else test_x[test_y == digit]
    z = z.reshape(-1, 1, M, 1)
    res = (np.eye(M) - u @ np.transpose(u, (0, 2, 1))) @ z
    res = norm(res, 2, axis=2) / norm(z, 2, axis=2)
    res = res.reshape(-1, 10)
    return res
```

مراجع

[1] Lars Elden (2007) matrix methods in data mining and pattern recognition.

[2] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. Proc. IEEE, 86:2278–2324, Nov. 1998.

[3] A pattern recognition method based on principal component models. In Pattern Recognition in Practice,