گزارش تکلیف ۵ درس یادگیری ماشین

کسرا سینایی
شماره دانشجویی ۸۱۰۶۹۶۲۵۴

۲۴ دی ۱۴۰۰

سؤال یک

الف

روش‌های جست و جو:

- Exhustive: در این روش‌ها تمام حالات ممکن برای به دست آوردن زیرمموعه‌ای از فیچرها امکان پذیر است در نظر گرفته می‌شوند. اگر n فیچر داشته باشیم، پیچیدگی محاسباتی این روش $O(n^2)$ است. به دلیل پیچیدگی محاسباتی بالا، این روش معمولا کاربردهای کمی دارند (مثال: Breadth First Search)

- Heuristic: روش‌هایی مانند SFS، SBS، BDS و ... هستند. در این روش‌ها یا با مجموعه‌ی کامل فیچرها شروع کرده و به ترتیب فیچرهایی که حذف آن‌ها بهینه ترین زیرمجموعه جدید را نتیجه دهد حذف می‌شوند، یا با زیرمجوعه تهی از فیچرها شروع کرده و به مرور فیچرهایی را که بهینه ترین زیرمجموعه جدید را حاصل می‌کنند اضافه می‌شوند به زیرمجموعه فیچرها.

- Randomize: در این روش ابتدا به صورت اتفاقی زیرمجموعه‌ای از فیچرها انتخاب می‌شوند، سپس با استفاده از الگوریتم‌هایی مانند ژنتیک، RGSS و ... به بهینه سازی تابع هزینه می‌پردازند تا زیرمجموعه بهینه از فیچرها به دست آید.

روش‌های ارزیابی:

- Filter Methods: این روش‌ها بدون توجه به اللگوریتم طبقه‌بندی زیرمجموعه انتخابی از فیچرها را ارزیابی می‌کنند. معیار اصلی ارزیابی اطلاعات موجود در هر زیرمجموعه از فیچر است. این روش‌ها سریع هستند و تمایل به انتخاب زیر مجموعه‌های بزرگی از فیچرها را داغرند.

- Wrpper Methods: برای ارزیابی زیرمجموعه انتخاب شده از فیچرها، معیارهایی در نظر گرفته می‌شود که به الگوریتم طبقه بندی مربوط است. برای مثال پروسه ارزیابی کیفیت زیرمجموعه فیچر انتخاب شده از دقت آن در پیش بینی تعدادی داده تست استفاده می‌شود. این روش‌ها آهسته هستند اما دقیق تر از Filter Methods کار می‌کنند.

ب

در محاسبات LDA لازم است معکوس ماتریس S_w را حساب کرد و سپس مقادیر ویژه و بردارهای ویژه $S_w^{-1}S_b$ را به دست آورد. با افزایش ابعاد مسئله حجم محاسبات جبری افزایش می‌یابد. همچنین اگر تعداد نمونه‌ها کم باشد ممکن است ماتریس S_w سینگولار شود.
در PCA نیز باید مقادیر ویژه ماتریس کواریانس نمونه‌ها و بردارهای متناظر با آن‌ها محاسبه شوند تا PCها به دست آیند. اگر ابعاد مسئله زیاد شود علاوه بر حجم محاسباتی مقادیر ویژه امکان کاهش دقت تخمین ماتریس کواریانس هم به وجود می‌آید. یکی از نقاط ضعف PCA حساسیت ان به اسکیل فیچرها می‌باشد به همین دلیل نرمال کردن دیتا قبل از اجرای الگوریتم اهمیت ویژه‌ای دارد.

سوال دو

الف)

$$S_T = \sum_{x} (x-\mu)(x-\mu)^T = \sum_{k=1}^{c} \sum_{x \in D_k} (x-\mu_k+\mu_k-\mu)(x-\mu_k+\mu_k-\mu)^T$$

$$= \underbrace{\sum_{k=1}^{c} \sum_{x \in D_k} (x-\mu_k)(x-\mu_k)^T}_{S_W} + \underbrace{\sum_{k=1}^{c} \sum_{x \in D_k} (\mu-\mu_k)(\mu-\mu_k)^T}_{S_B}$$

$$\longrightarrow S_T = S_W + S_B$$

ب) هر ترم از حاصل جمع های عبارت S_B به صورت quadratic است و از ضرب یک عبارت در transpose خود تشکیل می‌شود. بنابراین حداکثر رنگ این ماتریس‌ها ۱ است. در مسائل C کلاسه، S_B حاصل جمع C تا از این ماتریس‌ها است. پس رنگ آن حداکثر می‌تواند C باشد. یک نکته باقی می‌ماند و آن، این است که μ خود میانگین μ_k ها است و باعث می‌شود یک درجه آزادی از S_B سلب شود؛ در نتیجه $rank(S_B) \leq C-1$ است.

سؤال چهار

$P(x|\omega_i) \sim m_i, \sum_i$ cost: $J(w) = \frac{(\mu_1 - \mu_2)^2}{\delta_1^2 + \delta_2^2}$

projection: $y = w^T x \rightarrow P(y|\omega_i) \sim \mu_i, \delta_i^2$

$$S_i = \sum_{x \in D_i} (x - m_i)(x - m_i)^T \quad ; \quad S_w = S_1 + S_2 \overset{*}{}$$

$$\mu_i = w^T m_i \quad ; \quad \delta_i^2 = \sum_{y \in D_i} (y - \mu_i)^2 = \sum_{y \in D_i} (w^T(x - m_i))(w^T(x - m_i))^T$$

$$= \sum_{y \in D_i} w^T (x - m_i)(x - m_i)^T = w^T S_i w$$

$$\Rightarrow \delta_1^2 + \delta_2^2 = w^T S_1 w + w^T S_2 w \overset{*}{=} w^T (S_1 + S_2) w = w^T S_w w \quad (I)$$

$$S_B = (m_1 - m_2)(m_1 - m_2)^T \qquad (\mu_1 - \mu_2)^2 = (w^T m_1 - w^T m_2)^2 = w^T (m_1 - m_2)(m_1 - m_2)^T$$

$$= w^T S_B w \quad (II)$$

$$\overset{I, II}{\Longrightarrow} J(w) = \frac{(\mu_1 - \mu_2)^2}{\delta_1^2 + \delta_2^2} = \frac{w^T S_B w}{w^T S_w w}$$

$$\frac{\delta J}{\delta w} = \frac{\left(\frac{\partial}{\partial w} w^T S_B w\right) w^T S_w w - \left(\frac{\partial}{\partial w} w^T S_w w\right) w^T S_B w}{(w^T S_w w)(w^T S_w w)^T} = 0$$

$$\rightarrow \quad 2 (S_B w) w^T S_w w - 2 (S_w w) w^T S_B w = 0$$

$$\frac{(S_B w) w^T S_w w}{w^T S_w w} - \frac{(S_w w) w^T S_B w}{w^T S_w w} = 0 \rightarrow S_B w - \lambda S_w w = 0$$

$$S_B w = \lambda S_w w$$

if S_w is invertable $\longrightarrow S_w^{-1} S_B w = \lambda w$ eigen value problem

$S_B w = (m_1 - m_2)(m_1 - m_2)^T w = \alpha (m_1 - m_2) \longrightarrow$ $S_B w$ is aligned with $m_1 - m_2$

$$\Rightarrow w = S_w^{-1} (m_1 - m_2) = (\Sigma_1 + \Sigma_2)^{-1} (m_1 - m_2)$$

2000

سؤال پنج

الف) $\mu_1 = (1.33, 1.33) \quad \mu_2 = (2.5, 2) \quad m = (1.8, 1.6)$

$$S_B = \sum_{i=1}^{c} N_i (\mu_i - m)(\mu_i - m)^T = 3 \begin{bmatrix} -0.47 \\ -0.27 \end{bmatrix} \begin{bmatrix} -0.47 & -0.27 \end{bmatrix} + 2 \begin{bmatrix} 0.7 \\ 0.4 \end{bmatrix} \begin{bmatrix} 0.7 & 0.4 \end{bmatrix}$$

$$\longrightarrow S_B = \begin{bmatrix} 0.635 & 0.359 \\ 0.359 & 0.203 \end{bmatrix} + \begin{bmatrix} 0.98 & 0.56 \\ 0.56 & 0.32 \end{bmatrix} = \begin{bmatrix} 1.615 & 0.919 \\ 0.919 & 0.523 \end{bmatrix}$$

ب)

$$S_1 = \sum_{x} (x - \mu_1)(x - \mu_1)^T = \begin{bmatrix} -0.33 \\ -0.33 \end{bmatrix} \begin{bmatrix} -0.33 & -0.33 \end{bmatrix} + \begin{bmatrix} 0.66 \\ -0.33 \end{bmatrix} \begin{bmatrix} 0.66 & -0.33 \end{bmatrix} + \begin{bmatrix} -0.33 \\ 0.66 \end{bmatrix} \begin{bmatrix} -0.33 & 0.66 \end{bmatrix}$$

$$\longrightarrow S_1 = \begin{bmatrix} 0.1089 & 0.1089 \\ 0.1089 & 0.1089 \end{bmatrix} + \begin{bmatrix} 0.4356 & -0.2178 \\ -0.2178 & 0.1089 \end{bmatrix} + \begin{bmatrix} 0.1089 & -0.2178 \\ -0.2178 & 0.4356 \end{bmatrix} = \begin{bmatrix} 0.6534 & -0.3285 \\ -0.3285 & 0.6534 \end{bmatrix}$$

$$S_{-1} = \begin{bmatrix} -0.5 \\ 0 \end{bmatrix} \begin{bmatrix} -0.5 & 0 \end{bmatrix} + \begin{bmatrix} 0.5 \\ 0 \end{bmatrix} \begin{bmatrix} 0.5 & 0 \end{bmatrix} = \begin{bmatrix} 0.25 & 0 \\ 0 & 0 \end{bmatrix} + \begin{bmatrix} 0.25 & 0 \\ 0 & 0 \end{bmatrix} = \begin{bmatrix} 0.5 & 0 \\ 0 & 0 \end{bmatrix}$$

$$S_W = S_1 + S_{-1} = \begin{bmatrix} 1.1534 & -0.3285 \\ -0.3285 & 0.6534 \end{bmatrix}$$

ج)

$$S_W^{-1} S_B = \begin{bmatrix} 1.011 & 0.508 \\ 0.508 & 1.786 \end{bmatrix} \begin{bmatrix} 1.615 & 0.919 \\ 0.919 & 0.523 \end{bmatrix} = \begin{bmatrix} 2.101 & 1.196 \\ 2.462 & 1.401 \end{bmatrix}$$

eigen values: $\det \begin{bmatrix} 2.101 - \lambda & 1.196 \\ 2.462 & 1.401 - \lambda \end{bmatrix} = 0 \longrightarrow (2.101 - \lambda)(1.401 - \lambda) = (1.196)(2.462)$

$\lambda_1 = 3.502$

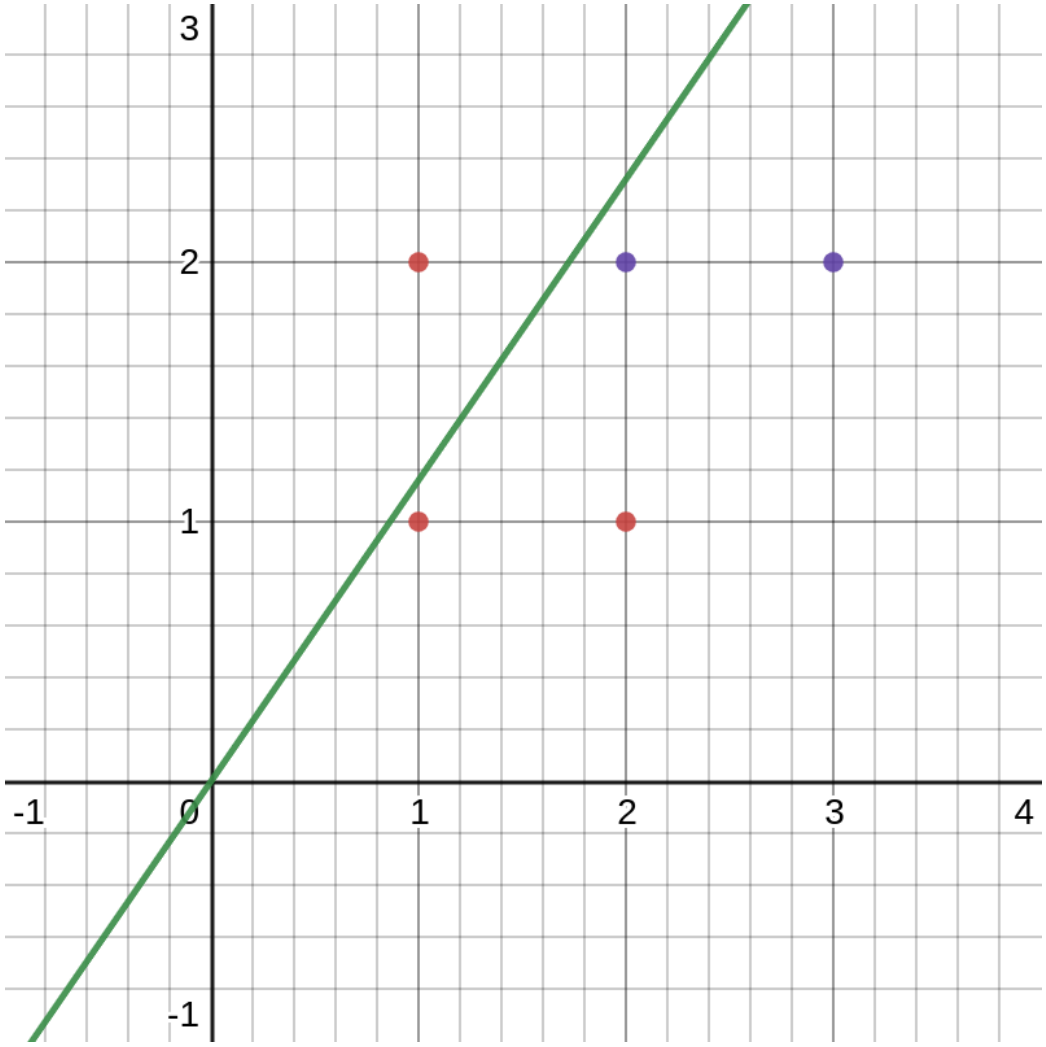
$\lambda_2 = 1.06 \times 10^{-5}$

eigen vectors $= \begin{bmatrix} +1.0 \\ +1.16 \end{bmatrix} > \begin{bmatrix} 0 \\ 0 \end{bmatrix}$

Best LDA directions: $tg^{-1}\left(\frac{1.16}{1.0}\right)$

در شکل خط $y = 1.16x$ رسم شده است

با توجه به شیب خطهای به دست آمده از قسمت قبل جهت‌های به دست آمده را همراه با داده‌ها رسم می‌کنیم. نقاط قرمز مربوط به $y_i = 1$ و نقاط بنفش مربوط به کلاس $y_i = -1$ هستند.

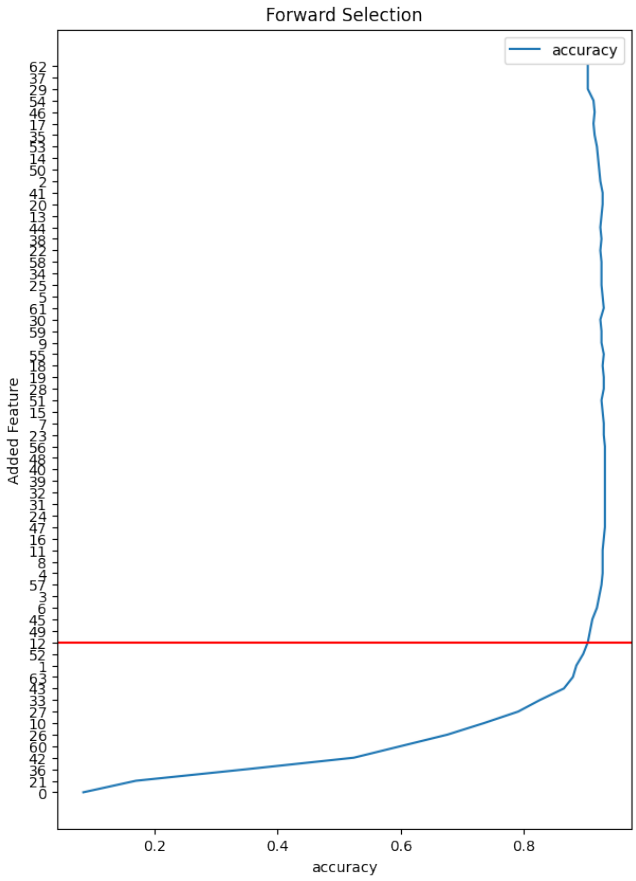

3
2
1
-1
0
1
2
3
4
-1

سؤال شش

الف

برای پیاده سازی SFS در حلقه for بیرونی دو متغیر تعریف می‌کنیم برای یافتن بهترین فیچر جهت اضافه کردن به لیست selected feature index. در حلقه for درونی که بر روی i لوپ می‌زند ابتدا چک می‌کنیم که i قبلا انتخاب نشده باشد، سپس بردارهای x و x test را از روی متغییرهای کلاس و با توجه به ایندکس فیچرهای انتخاب شده تشکیل می‌دهیم و با استفاده از دستورات کتابخانه scikit learn دقت آن را بر روی مجموعه تست اندازه گیری می‌کنیم. پس از هر بار ایتریش بر روی متغییر j یک ایندکس به زیر مجموعه فیچرها اضافه می‌شود و دقت طبقه بندی آن نیز به لیست acc list اضافه می‌شود.

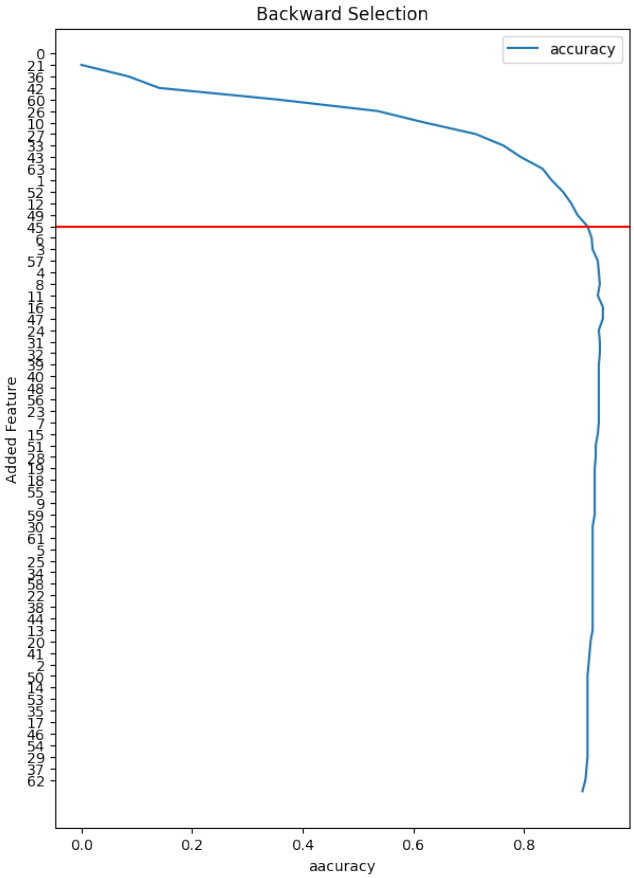
برای رسم نمدار مشابه نمودار مثال از کتابخانه pandas و matplotlib استفاده شده است. متغییر th (threshold) مقداری است که در آن دقت طبقه بندی با زیر مجموعه به دست آمنده بیشتر از ۹۰٪ شود در نمودار با رنگ قرمز رسم شده است. تیک‌های محور y نیز لیست بازگردانده شده از متد forward می‌باشند. این نمودار در شکل ۱ آورده شده است.



شکل ۱: نتایج اجرای الگوریتم SFS

ب

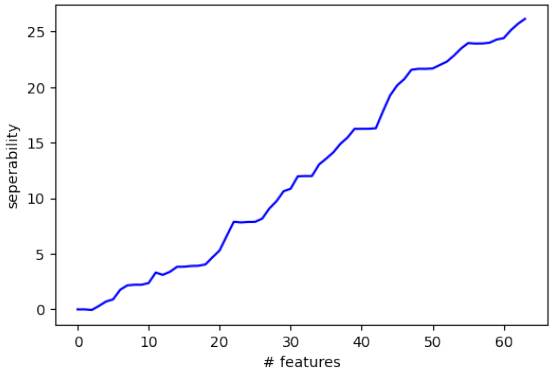
این قسمت شباهت زیادی به بخش الف دارد. فقط برای به دست آوردن زیرمجموعه فیچرها از تابع delete از کتبخانه numpy استفاده می‌کنیم تا بردارهای x و x test را بسازیم.
نمودار رسم شده نیز مانند بخش قبل می‌باشد و تنها تفاوت در به دست آوردن مقدار th می‌باشد که باید از انتها به ابتدا لوپ زد و مقدار بهینه ترشهولد را به دست آورد. نمودار خواسته شده در شکل ۲ نشان داده شده است.



شکل ۲: نتایج اجرای الگوریتم SBS

سؤال هفت

برای تکمیل تابع Si که مقدار scatter matrix کلاس iام می‌باشد از رابطه $S_i = \sum_{\boldsymbol{x} \in D_i}^{n} (\boldsymbol{x} - \boldsymbol{m}_i)(\boldsymbol{x} - \boldsymbol{m}_i)^T$ استفاده می‌کنیم. در این رابطه $\boldsymbol{m}_i$ میانگین داده‌های کلاس iام است.
داده‌های ارسال شده در آرگومان ورودی این تابع باید مختص به یک کلاس باشند. جدا سازی داده‌های کلاس‌های مختلف در تابع Sb انجام می‌گیرد. در واقع از آرگومان دوم این تابع استفاده نشده است.
برای پیاده سازی تابع Sw که within class scatter matrix می‌باشد از رابطه $S_w = \sum_{i=1}^{c} S_i$ استفاده می‌کنیم.
آخرین تابع که برای محاسبه LDA به آن نیاز داریم Between class scatter matrix می‌باشد که در تابع Sb و به کمک رابطه $S_B = \sum_{i=1}^{c} N_i (\boldsymbol{m}_i - \boldsymbol{m})(\boldsymbol{m}_i - \boldsymbol{m})^T$ می‌توان آن را محاسبه کرد. در این رابطه $\boldsymbol{m}_i$ میانگین نمونه‌های کلاس iام و $\boldsymbol{m}$ میانگین داده‌های تمام کلاس‌ها است. N_i نیز تعداد نمونه‌های کلاس iام است.
برای محاسبه ترنسفورمیشن‌های LDA مسئله مقادیر ویژه ارائه شده در کلاس درس را باید حل کنیم. بردار w بردار ویژه‌های ماتریس $S_W^{-1} S_B$ می‌باشد. برای اینکه تابع نوشته شده برای مواردی که ماتریس S_W سینگولار می‌شود نیز کار کند از pseudo inverse استفاده شده است.
ورودی تمام کلاس‌ها باید آرایه numpy باشند. از روی دیتا ست استفاده شده می‌توان مقادیر data و labels را برای فراخوانی تمام توابع استفاده کرد.
نمودار seperability بر حسب تعداد ویژگی‌ها نیز در شکل ۳ نشان داده شده است.



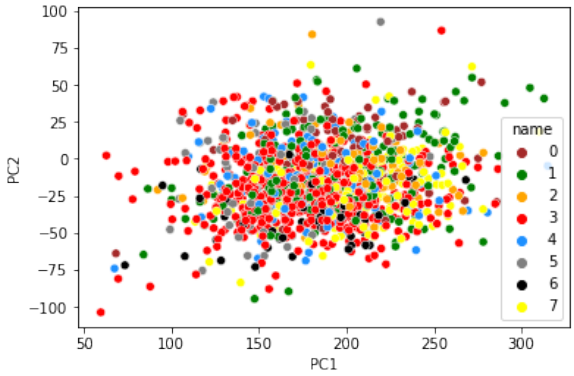
شکل ۳: معیار تفکیک پذیری بر حسب تعداد ویژگی‌های LDA

سؤال هشت

برای تکمیل تابع PCATransform کافی است با دستورات کتابخانه numpy ابتدا ماتریس کواریانس داده‌های ورودی را محاسبه کنیم و سپس بردارهای ویژه آن را بر حسب مقادیر ویژه متناظرشان سورت کنیم و در نهایت به تعداد PCهای خواسته شده بازگردانیم. در این تابع علاوه بر PCهای خواسته شده واریانس داده‌های تصویر شده بر روی بردارهای به دست آمده و مقادیر بردارهای ویژه که همان PCها هستند نیز بازگردانده می‌شوند.

الف

در این قسمت با استفاده از خروجی اول تابع پیاده سازی شده که داده‌های تصویر شده بر روی PCها هستند تصاویر چهره‌ها را بر روی دو کامپننت اول تصویر کرده و در نموداری مانند مثال رسم می‌کنیم. این نمودار در شکل ۴ نشان داده شده است.



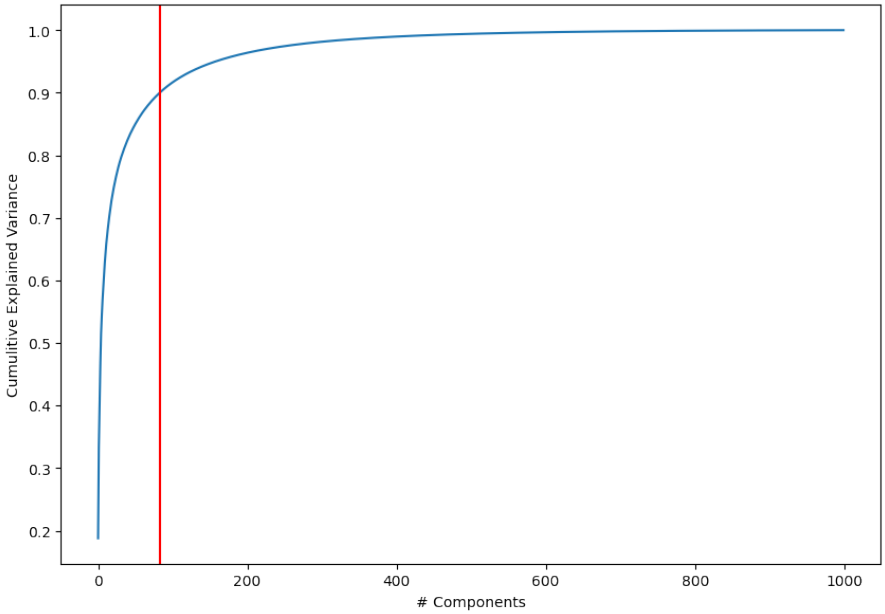
شکل ۴: تصاویر پس از تصویر شدن بر روی ۲ کامپننت اول PCA

ب

PCA یک الگوریتم unsupervised است که می‌توان به کمک آن جهت بردارهایی را پیدا کرد که اگر داده را بر روی آن‌ها تصویر کرد به داده‌های در ابعاد پایین تر رسید که بیشترین مقدار واریانس داده‌های اصلی را حفظ می‌کنند. دو این جهت‌ها بردارهای ویژه ماتریس کواریانس هستند و دو کامپوننت اول بردارهای ویژه‌ای هتند که اولین و دومین مقدار ویژه بزرگتر را نسبت به سایر بردارها دارند.

ج

خروجی دوم تابع پیاده سازی شده در این سؤال نسبت واریانس بیان شده توسط داده‌های تصویر شده بر روی n کامپننت اول به صورت تجمعی را نشان می‌دهند. این آرایه numpy دقیقاً حاوی داده خواسته شده برای قسمت ب می‌باشد تنها لازم است مقدار ترشهولد را را نیز پیدا کنیم که به سادگی قابل یافتن می‌باشد. این مقدار بر روی نمودار با خط قرمز مشخص شده است. مقدار ترشهولد تعداد کامپوننت‌هایی را نشان می‌دهد که با تصویرسازی داده‌ها بر روی آن‌ها می‌توان ۹۰٪ اطلاعات نمونه‌ها را پس از dimentionality reduction نشان داد.



شکل ۵: واریانس داده‌ها پس از انتقال بر روی PCها بر حسب تعداد کامپوننت

د

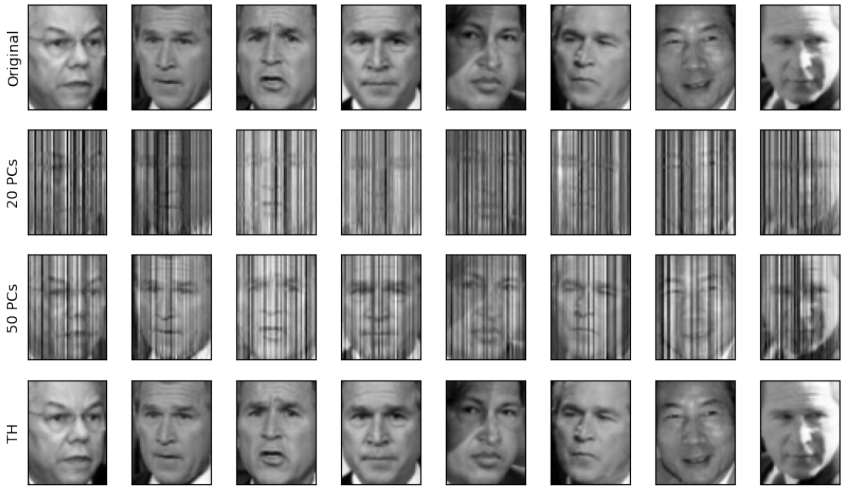
اگر ۳۰ بردار ویژه بازگردانده شده از تابع نوشته شده در این قسمت را با دستور reshape به شکل تصاویر در بیاریم نتیجه مانند شکل ۶ می‌شود. هر کامپونت PCA نشان دهنده جهتی است که فیچرها بر روی آن از داده اصلی تصویر می‌شوند.

شکل ۶: ۳۰ کامپوننت اول

در واقع قسمت‌های پر رنگ تر در این شکل قسمت‌هایی از عکس هستند که با تصویر بر روی PC آن به وجود می‌آیند.

۵

در این قسمت به ازای مقدار ترشهولد به دست آمده در قسمت ج و همچنین دو مقدار کمتر از آن تصاویر هر هشت نفر انتقال داده شده و مجددا با معکوس انتقال باز گردانده شده از تابع PCATransform عکس‌ها بازیابی شده‌اند. نتایج در شکل ۷ نشان داده شده‌اند.



شکل ۷: نتایج reconstruction تصاویر پس از انتقال به ازای تعداد مختلفی PC