Algoritmus na rozpoznávání táří pomocí metody PCA, neboli analýzy havních komponentů

Michal Sekerka

19. září 2017

1 Úvod

Lidská schopnost rozpoznávat tváře je fascinující. Dospělý člověk je schopný si zapamatovat a napříč svým životem od sebe správně rozeznávat tisíce obličejů a to i přes celou řadu změn obličejových charakteristik, které si na nás příroda přichystala. Opravdu ani změna v délce vlasů, hloubce vrásek, intenzitě opálení, či výrazu tváře nám nikterak nepřekáží při správné identifikaci jednotlivce. Zmíněné lidské nadání po celá staletí zaráželo a inspirovalo řadu intelektuálních autorit své doby (Aristoteles, Darwin,. . .)[4] napříč všemi možnými obory jako je psychologie, biologie či informatika.

První algoritmy na zpracování a rozpoznání objektů v digitální fotografii se objevovaly již na konci šedesátých let a nemalým způsobem formovaly dnešní pojetí strojového učení. Ovšem i přes od té doby neustávající výzkum v oblasti problematiky obecných klasifikačních algoritmů digitálních fotografií, pracuje dnes počítač stále na úrovni malého dítěte. Podle profesora Flussera (pořad Na plovárně, 2015), dosáhlo zmíněné dítě pěti a dle Richarda Szeliskiho (2010, [10]), dvou let. Optimisticky vypadá alespoň fakt, že věk imaginární ratolesti a délka vývoje klasifikačních algoritmů se zatím zdají přímo úměrné.

1.1 Klasifikační problémy

V předchozí části textu jsem často používal výrazy jako přiřazení, klasifikace a rozpoznávání. Pokusme se daný problém tedy formalizovat [9][8].

Uvažujme vektorový prostor V nad tělesem T, množinu W a zobrazení $f : V \longmapsto$ W. Dále předpokládejme, že známe hodnoty $u_1, \ldots, u_n \in V$ a $f(u_1), \ldots, f(u_n) \in W$. Procesem učení (s učitelem) pak nazýváme akt aproximace funkce f při znalosti dvojic bodů $(u_i, f(u_i))$, i=1,...,n. Klasifikčním problémem rozumíme problematiku přiřazení bodu $f(u) \in W$ hodnotě $u \in V$. Množina bodů P $= \{(u_i); i = 1, .., n\}$ se označuje trénovací množinou, V prostorem dat a W množinou přihrádek. Za zmínku pak dále stojí, že o rozumném odpovídání na vstupy nezahrnuté v procesu učení se obvykle mluví jako o generalizaci.
Možnosti toho, jak reprezentovat přihrádky se pohybují všude od kanonických vektorů, přes integery až po booleovské proměnné.

Př.
Než se pustíme do rozpoznávání tváří, zamysleme se nad jednodušším problémem a to nad přiřazením obrázků ručně psaných a tištěných cifer velikosti n x n pixelů k číslicím 0-9. V tomto případě je prostorem dat vektorový prostor $\mathbb{R}^{n^2}$ (hodnoty pixelů bývají celá čísla od -127 do 128), množinou přihrádek je P $= \{0, 1, .., 9\}$ a naším cílem je aproximovat funkci f, která přiřazuje obrázku cifry její identitu.[8]

Jak k problému přistoupit?
Nejranější přístupy se snažily o to, co by většinu z nás, kdybychom neznali moderní metody strojového učení, při řešení daného problému asi také napadlo jako první: Sepsat příznaky, číslice jimi nějak charakterizovat a jednotlivé obrázky podle nich klasifikovat: Nula je kulatá, trojka je horizontálně, ale nikoliv vertikálně symetrická,a tak dále.... Daný přístup má velké ale. V podstatě ke každému pravidlu existuje výjimka. Ručně psaná nula bývá kulatá, ovšem nula na digitálních hodinách je hranatá, mnoho lidí nepíše trojku symetricky... V praxi se s jednotlivými rukopisy velmi

rychle množí pravidla a výjimky, které jsme nuceni formulovat. I s nimi ovšem takové přístupy nedosahují optimálních výsledků. Ukázalo se, že mnohem lepším řešením je vypracovat model, nebo černou skříňku, chcete-li, která bude příznaky extrahovat za nás a zprostí nás tak břemene jejich formulace. Jak v této práci uvidíme, nemusí se jednat o konkrétní charakteristiky jako je symetrie či oblost tvarů, ale počítač může pracovat s něčím abstraktnějším, slovy těžko či nemožně formulovatelným.
Jak už název napovídá, mimo jiné se problematikou nalezení těchto fundamentálních vlastností zabývá právě metoda analýzy hlavních komponentů, nebo-li PCA.

2 Matematický aparát

V této sekci se budu zabývat matematickým aparátem nezbytným k formulaci algoritmu.

2.1 Lineární algebra

Důkazy všech vět v této sekci jsou dohledatelné v [11]

2.1.1 Skalární součin

Definice (Skalární součin, kolmost)
Mějme vektorový prostor $\mathbb{R}^n$ a prvky $u = (u_1, .., u_n), v = (v_1, \dots v_n) \in \mathbb{R}^n$. Skalární součin u s v nad prostorem $\mathbb{R}^n$ definujeme jako

$$u \cdot v = u_1 v_1 + \dots + u_n v_n$$

Navíc řekneme, že u, v jsou na sebe kolmé (nad $\mathbb{R}^n$), pokud $u \cdot v = 0$. Tento fakt značíme $u \perp v$.
Definice (Ortonormální posloupnost a báze, ortogonální matice)
Mějme posloupnost vektorů $B = b_1, \dots, b_n \in \mathbb{R}^n$, $n \in N$. B pak nazveme ortonormální, pokud jsou splněny následující podmínky:

1. $b_i \perp b_j$ pro každé $i, j \in \{1, \dots, n\}, i \neq j$
2. $b_i \cdot b_i = 1$ pro každé $i \in \{1, \dots, n\}$

B navíc nazveme ortonormální bází prostoru $< b_1, ..., b_n > \leq \mathbb{R}^n$.
Matici $A \in [\mathbb{R}]_{nxn}$ jejíž sloupce tvoří ortonormální bázi prostoru $\mathbb{R}^n$ nazveme ortogonální.
Definice (Ortogonální projekce)
Buď $U \leq \mathbb{R}^n$ a $v \in \mathbb{R}^n$, potom $w \in U$ nazveme ortogonální projekcí v na prostor U, pokud $(v - w) \perp U$, tedy pokud $(v - w) \cdot q = 0$ pro každé $q \in U$.
Věta 2.1.1.1. (Souřadnice vzhledem k ON bázi)
Mějme vektor $v \in \mathbb{R}^n$, prostor $U \leq \mathbb{R}^n$, ortonormální bázi $B = u_1, \dots, u_k$ prostoru U. Dále nechť $w \in U$ je ortogonální projekcí v na prostor U, pak

$$[w]_B = (v \cdot u_1, ..., v \cdot u_k)$$

2.1.2 Vlastní vektory + SVD

Definice (Vlastní čísla a vektory)
Buď $A \in [\mathbb{R}]_{nxn}$ čtercová matice a $\lambda \in \mathbb{R}$. Existuje-li $x \in \mathbb{R}^n, x \neq 0$ takové, že

$$Ax = \lambda x,$$

pak λ nazveme vlastním číslem matice A a vektor x vlastním vektorem matice A příslušým k λ.
Pojem vlastního vektoru úzce souvisí s některými maticovými rozklady. Uveďme si zde dva takové.
Věta 2.1.2.1 (O ortogonální diagonalizaci symetrických matic)
Nechť A je reálná symetrická matice řádu n. Pak existuje ortogonální matice $U = (u_1, ..., u_n)$ taková, že

$$U^T A U = D,$$

pro nějakou diagonální matici $D = diag(d_1, ..., d_n)$, kde $d_i \in \mathbb{R}$, $i = 1, ..., n$. Hodnoty d_i jsou vlastními čísly matice A a u_i jsou k nim příslušné vlastní vektory matice A pro každé $i \in \{1, ..., n\}$.

Věta 2.1.2.2 (Singulární rozklad)
Nechť A je reálná obdélníková matice typu m x n hodnosti r. Pak existuje ortonormální báze $B = v_1, ..., v_n$, prostoru $\mathbb{R}^n$ a $C = u_1, ..., u_m$ prostoru $\mathbb{R}^m$ taková, že

$$A = UDV^{-1} = UDV^T,$$

kde $U = (u_1, .., u_m), V = (v_1, \ldots, v_n)$ jsou ortogonální matice a $D = (d_{i,j})_{mxn}$ je reálná obdélníková matice, přičemž $d_{i,j} = 0$ pro $i \neq j$ a $d_{1,1} \geq d_{2,2} \geq \ldots \geq d_{r,r} \geq 0$. Dále platí, že $d_{1,1}, \ldots, d_{r,r}$ jsou odmocniny vlastních čísel matice A^TA a každé $v_i, i = 1, ..., r$, je vlastním vektorem matice A^TA příslušný k vlastnímu číslu d_i.

2.2 Statistika

Ke zjišťování vztahů mezi jednotlivými vzorky se ve statistice používá řada veličin. Zadefinujme si ty, které budeme v této práci používat:

Definice (Rozptyl a střední hodnota)
Buď $m = (m_1, \ldots, m_n) \in \mathbb{R}^n$ měření, pak

- $Em = \frac{1}{n} \sum_{i=1}^{n} m_i$ nazveme střední hodnotou měření m.
- $var(m) = \frac{1}{n-1} \sum_{i=1}^{n} (m_i - Em)^2$ nazveme rozptylem měření m.

Střední hodnota je důležitým ukazatelem, ovšem sama o sobě toho o našich datech mnoho neříká. Nesděluje nám totiž nic o tom, jak jsou okolo ní data rozmístěna. Rozptyl označuje průměrnou odchylku od střední hodnoty a částečně tento fenomén postihuje.

Střední hodnota a rozptyl jsou jednodimenzionální ukazatele. Mnoho datových množin obsahuje vzorky složené ze dvou a více měření. Abychom to ilustrovali na příkladě, tak si přestavme, že měříme počet odběhaných hodin a spálených kalorií na nějaké populaci sportovců. Tedy máme vzorky skládající se ze dvou měření, jinými slovy pracujeme s dvou-dimenzionálními daty. Intuice nám navíc říká, že z času, který běžec odcvičil už budeme poměrně dobře schopni odhadnout spálené kalorie. To představuje jistou redundaci v našich datech[2]. Původní data bychom totiž mohli poměrně dobře zpětně zrekonstruovat z jednodimenzionálních dat (se zanedbatelnou chybou). Rádi bychom tedy postihli, jak moc jsou daná dvě měření závislá. Data týkající se hodin strávených během a počet spálených kalorií si můžeme zapsat do sloupcových vektorů $u := (u_1, ..., u_n)$, $v := (v_1, ..., v_n)$. Nyní čím více jsou vektory $\overline{u} := (u_1 - Eu, ..., u_n - Eu)$, $\overline{v} := (v_1 - Ev, ..., v_n - Ev)$ ve stejném směru, tím větší je hodnota skalárního součinu $\overline{u} \cdot \overline{v}$. Stejnosměrnost vektorů implikuje přímou úměru a kladnou hodnotu skalárního součinu. Nebo-li kladná hodnota skalárního součinu naznačuje, že s růstem jednoho měření oproti jeho střední hodnotě roste i to druhé vyhledem k jeho střední hodnotě. Tedy daná měření se oproti svému průměru chovají podobně. Pokud jsou vektory protichůdné vychází záporná hodnota součinu a mluvíme o nepřímé úměře. Její nulovost implikuje (lineární) nezávislost těchto dvou veličin a ortogonalitu vektorů. Tohoto faktu využívá následující definice tzv. kovariance, která zde zmíněnou redundanci dat částečně popisuje.

Definice (Kovariance)
Buďte $m = (m_1, \ldots, m_n), o = (o_1, \ldots, o_n)$, m_i, $o_i \in \mathbb{R}, i = 1, ..., n$ dvě měření, pak hodnotu:
Cov(x,y) $= \frac{1}{n-1} \sum_{i=1}^{n} (m_i - Em)(o_i - Eo)$ nazveme kovariancí m a o.

Způsob, jakým postihujeme tento fenomén pro tři a více měření je tzv kovariační matice:

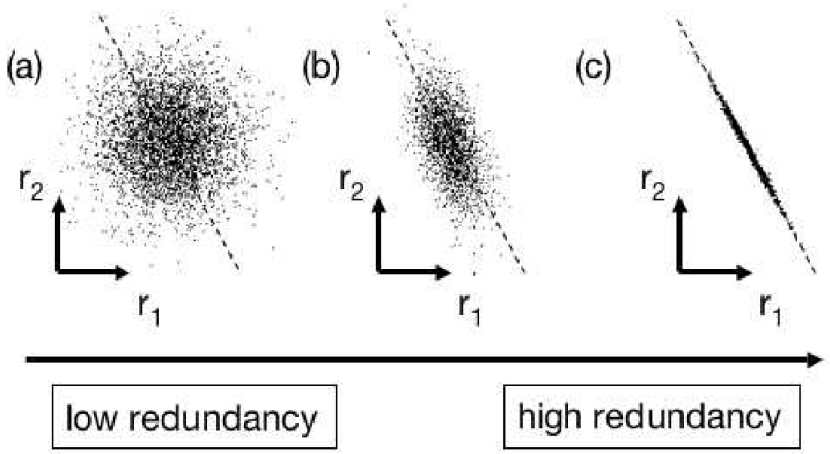
Definice(Kovarianční matice)
Buď $X = (m_1, ..., m_n)^T$ pro měření $m_i \in \mathbb{R}^n$ matice dat. Pak kovarianční matici C_X příslušnou k X definujeme jako:

- $C_X = \begin{pmatrix} Cov(m_1, m_1) & Cov(m_1, m_2) & ... & Cov(m_1, m_n) \\ Cov(m_2, m_1) & Cov(m_2, m_2) & ... & Cov(m_2, m_n) \\ . & . & . & . \\ . & . & . & . \\ . & . & . & . \\ Cov(m_n, m_1) & Cov(m_n, m_2) & ... & Cov(m_n, m_n) \end{pmatrix}$

Pozorování

- Cov(m_i,m_i) = var(m_i), $i = 1, ..., n$

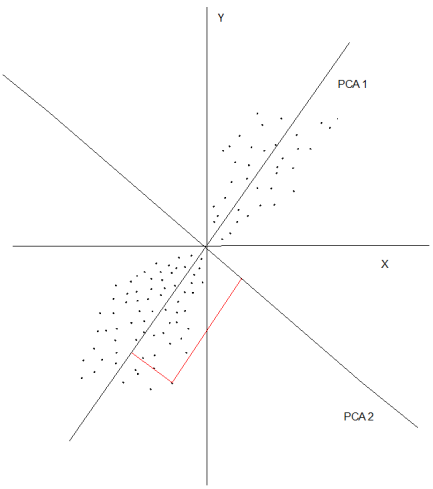
- var(m_i) ≥ 0 , přičemž rovnost nastává pokud m_i je nulovým vektorem pro $i = 1, .., n$
- C_X je symetrická matice s rozptyly jednotlivých vektorů x_i, $i = 1, ..., n$ na diagonále.



Obrázek 1: Zroj[1]:

2.3 PCA

Představme si, že pracujeme s mnohadimenzionálními daty a chceme mezi nimi najít nějaký fenomén. Například můžeme chtít zjistit, které vzorky se chovají podobně. V praxi je skoro nemožné kvalitativně zpracovat vzorky skládající se z velkého množství měření. Navíc víme, že jednotlivá měření mezi sebou obsahují jistou míru kovariance představující redundanci informace.
Uvažujme nyní syrová data v takové podobě, v jaké jsme je nasbírali. Je přirozené se ptát, neexistuje-li lepší způsob, jak naše data reprezentovat. Mimo jiné metody se tímto problémem zabývá právě PCA. V jejím případě chceme nalézt nový koordinační systém a vzorky v něm vyjádřit tak, abychom co nejvíce eliminovali redundanci mezi jednotlivými měřeními. Přeformulováno v matematickém jazyce, chceme najít ortonormální bázi a vzorky vyjádřit jako souřadnice vzhledem k této bázi tak, aby kovariance mezi transformovanými měřeními byla nulová. Tedy aby kovarianční matice příslušná k transformované datové matici byla diagonální. Vektory zmíněné ortonormální báze nazýváme hlavní komponenty a jejich nalezení je zásadní pro náš algoritmus.
Pokud bychom navíc mohli rozhodnout, které z dimenzí jsou nejdůležitější, mohli bychom ty nejméně důležité proměnné vynechat a ponechat si jen ty údaje, která vysvětlují drtivou většinu variance v našich datech. Důležité je podotknout, že PCA až doposud nijak neměnila vztahy mezi našimi vzorky, pouze měnila způsob, jakým se na daná data díváme.



Obrázek 2: PCA ve dvou dimenzích: Hledá směr největší variance v datech. Tento směr představuje osa PCA1.

PCA dělá dva poměrně silné předpoklady[1]

- Pracujeme se silně lineárně korelovanými daty.
- Nejdůležitější osy jsou takové, na nichž mají data maximální rozptyl.

Uvažujme vzorky $x_1, \ldots, x_k \in \mathbb{R}^n$
Bez újmy na obecnosti předpokládejme, že naše měření mají nulovou střední hodnotu. Jinak ji od nich odečtěme.
Na chvíli si navíc představme, že disponujeme námi hledanou ON bází $B = u_1, \ldots, u_n$ a maticí U $= (u_1, .., u_n)$. Chceme naše vzorky vyjádřit jako souřadnice vzhledem k B. Dle 2.1 platí $[x_i]_B = (x_i \cdot u_1, \ldots, x_i \cdot u_n) = U^T x_i$, pro $i = 1, .., n$. Odsud k vyjádření vozrků jako souřadnic vzhledem k bázi B stačí přenásobit jejich vektor maticí U^T zleva.

Víme, že naším cílem je nalezení báze diagonalizující kovarianční matici. Formulováno podrobněji uvažujeme-li datovou matici $X = (x_1, \ldots, x_n)$, pak dle předchozího textu chceme najít ortogonální matici U, že pro transformovaná data $Y = U^T X$ platí:

$$\frac{1}{n-1} YY^T = \frac{1}{n-1}(U^T X)(U^T X)^T = \frac{1}{n-1}(U^T X X^T U) = C_Y$$

pro nějakou diagonální matici C_Y, která je kovarianční maticí transformovaných dat Y.
Naše úloha lze dle doposavadních úvah přeformulovat jako ortogonální diagonalizace matice $A = XX^T$.
Ta dozajista existuje z věty 2.1.2.1. Jelikož $A^T = (XX^T)^T = XX^T = A$, tedy A je symetrická.
Tedy existují matice E, D, že D je diagonální, E ortogonální a navíc platí

$$XX^T = A = EDE^T$$

Kdybychom nyní položili námi hledanou matici $U = E$ a podívali se na kovarianční matici příslušnou k matici transformovaných dat Y, pak:

$$C_Y = \frac{1}{n-1} YY^T = \frac{1}{n-1} U^T X X^T U = \frac{1}{n-1} U^T (UDU^T) U = \frac{1}{n-1}(U^T U) D (U^T U) = \frac{1}{n-1} D.$$

Tedy námi hledaná báze diagonalizující kovarianční matici jsou sloupce matice $U = E$ a data v dekorelované podobě nám vychází jako $Y = U^T X$.[8]

Shrnutí postupu

1. Sesbírání vzorků.
2. Odečtení střední hodnoty od jednotlivých měření.
3. Vzorky do sloupců matice X.
4. Spočtení covariační matice $A = XX^T$.
5. Ortogonální diagonalizace matice A, $D = U^T AU$
6. Nový data-set $U^T X$.

Pokud nyní porovnáme provedení a požadavky na PCA a SVD, najdeme mnoho podobností. Máme-li totiž singulární rozklad $X = VQW^T$, tak

$$XX^T = VQWW^T Q^T V^T = VQ^2 V^T,$$

což je ortogonálním rozkladem naší kovarianční matice. Podíváme-li se pozorněji, pak $Q^2 = D$ a $U = V$. Tedy SVD spočítá námi hledanou bázi a odmocniny variancí mezi jednotlivými měřeními. K výpočtu SVD existuje řada efektivních metod, v praxi se tedy k výpočtu PCA velice často používá singulární rozklad a dané termíny se běžně zaměňují.

Pozorování 2.3.1
Vlastní čísla matice XX^T označují varianci hodnot na příslušné ose. Což dle definice ukazuje, jak moc se v dané dimenzi transformované hodnoty odlišují od střední hodnoty. Na danou osu se dá

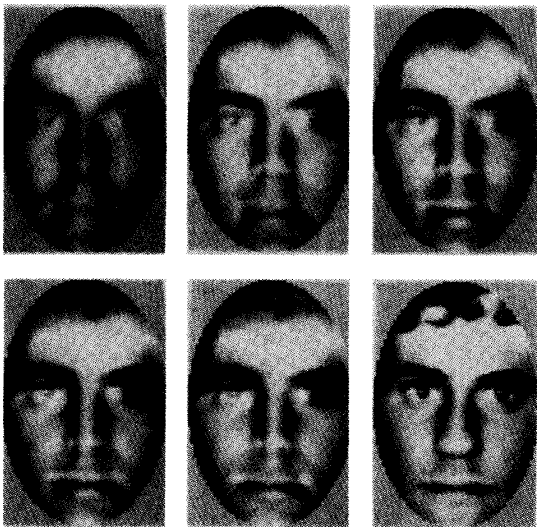
také dívat jako na částečné vysvětlení variance v celkových datech. K vektoru příslušné vlastní číslo pak ukazuje jak moc informace o celkové varianci nám souřadnice k příslušnému hlavnímu komponentu podává. Vlastní čísla lze odsud také chápat jako ohodnocení významnosti jednotlivých hlavních komponentů.

Důsledek[3][**?**]
Předchozí pozorování nám dává návod ke zrychlení vzjadřování věktorů vzhledem k bázi B. V tuto chvíli totiž máme nástroj, určující jak moc variance v datech která z os vysvětluje. Kdybychom z U vynechali ty hlavní komponenty, které přísluší k malým vlastním číslům, pak můžeme z důvodu zmenšení nutného počtu operací při násobení maticí U^T zleva vyjařovat nové vektory vzhledem k bázi B podstatně rychleji a neztratit s tím mnoho informace. Ke všemu budeme vědět jak moc informace bychom tímto aktem ztratili.

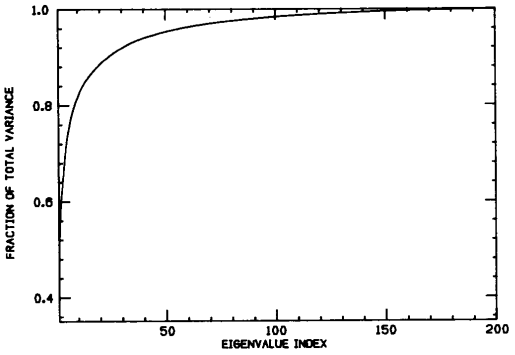
3 Algoritmus

3.1 Popis algoritmu

V předchozích sekcích jsem popsal vše potřebné k pochopení algoritmu, se kterým přišli Turk a Pentland[4]. V tomto algoritmu pracujeme s černobílými obrázky n x n pixelů. Tedy libovolný obrázek si můžeme představit jako bod v prostoru $\mathbb{R}^{n^2}$, kde každý pixel představuje jednu dimenzi. Také víme, že tato reprezentace, která ač vhodná pro vizualizaci v počítači, není vhodná pro klasifikaci obrazu. Je tomu tak, protože jednotlivé hodnoty pixelů různých obrazů tváří mezi sebou obsahují redundanci. My bychom je rádi dekorelovali. Jinými slovy bychom z klasifikovaného obrazu chtěli extrahovat příznaky, které co nejlépe vystihují čím se odlišuje a čím je naopak podobný s ostatními obrázky tváří, které máme v databázi. Tyto požadavky vyloženě volají po PCA.
Tento algoritmus stojí na objevu, který byl opublikován roku 1990 ve článku Siroviche a Kirbyho[7]. Ti přišli na efektivní způsob komprese obrazu obličeje. Na trénovací množině pomocí PCA určili hlavní komponenty, které generují to, co nazvali prostorem tváří. Tvář reprezentovali jakožto souřadnice vzhledem k bázi složené z těchto komponentů. V předchozí kapitole jsme vysvětlili, že velikost vlastních čísel reprezentuje jakousi důležitost hlavního komponentu [Pozorování 2.3.1]. Ovšem ukázalo se, že vlastní čísla naší diagonální matice vzniklé procesem PCA poměrně rychle klesají [Obrázek 3]. Absolutní většina variance ve vektorech obrazů tváří lze tedy vysvětlit prvními pár hlavními komponenty. Původní tvář je pak velice dobře aproximovatelná pomocí lineární kombinace těchto pár komponentů (a průměrné tváře). Tento fakt stál za efektivností jejich kompresního algoritmu a inspiroval Turka s Pentlandem k centrální myšlence našeho postupu[4].



Obrázek 2: Aproximace tváře zleva postupně složené jako kombinace z 10, 20, 30, 40, 50 hlavních komponentů a průměrné tváře.
Zdroj:[7]



Obrázek 3: Graf znázorňující, jak s přibývajícími hlavními komponenty přibývá zlomek vysvětlené variance. Povšimněme si, že k vysvětlení většiny variance stačí užít minimum z celkového počtu os.
Zdroj:[7]:

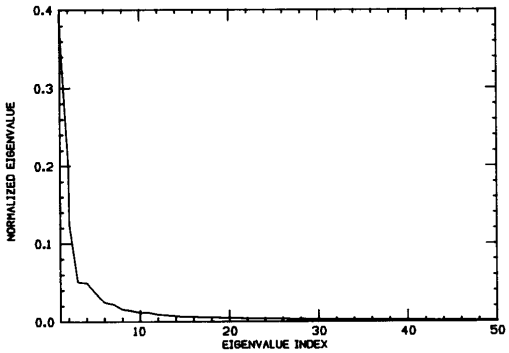


Fig. 4. Eigenvalues normalized by the maximum eigenvalue versus index.

Obrázek 4: Graf poklesu vlastních čísel kovarianční matice v práci Siroviche a Kirbyho.
Zdroj:[7]

Obrázek 5: Zajímavostí je, že prvních pár hlavních komponentů, nebo eigenfaců, jak je Turk s Pentlndem nazvali, když je zvizualizujeme opravdu vypadají jako tváře a postupně se snižujícím se vlastním číslem se stávají čím dál více náhodným šumem.
Zdroj:[4]

Mějme databázi tváří $x_1, \ldots, x_n$ a matici dat $X = (x_1, .., x_n)$. Pomocí PCA jsme schopni najít fundamentální vlastnosti, nebo-li hlavní komponenty, z trénovací množiny n obrázků v naší databázi. Ortogonální matici, která obsahuje hlavní komponenty ve sloupcích označme U. Nyní chceme klasifikovat nový obrázek tváře $x \in \mathbb{R}^{n^2}$. Víme, že výpočet $U^T x$ extrahuje z x informace podstatné k zařazení. Známe transformovanou databázi tváří $Y = U^T X$. Ta obsahuje v každém sloupci transformovaný obrázek z databáze tváří. Pokud porovnáme eukleidovskou vzdálenost $U^T x$ od některého ze sloupců matice Y, pak zjišťujeme, jak moc jsou tyto obrázky co do relevantních informací ke klasifikaci rozdílné. Z této představy je patrné, že takový sloupec matice Y, který svírá s $U^T x$ nejmenší eukleidovskou vzdálenost je tomuto klasifikovanému obrázku „nejpodobnější". Tedy tvář x je klasifikována jako tvář na obrázku, který patří k takovému vektoru.
Často se ještě používá nějaký prah $a \in \mathbb{R}$ a pokud je $|U^T x_i, U^T x|_2 > a$ pro všechna $i \in \{1, ..., n\}$, pak je obrázek interpretován jako tvář nenacházející se v databázi.

Nyní, když máme představu o tom jak to funguje algoritmus, popišme náš postup algoritmicky.

Trénování systému

1. Obstarání datové matice $X = (x_1, \ldots, x_n)$.
2. Odčtení průměrné tváře $\epsilon = \frac{1}{n} \sum_{i=1}^{n} x_i$ od sloupců matice X.
3. Extrakce relevantních informací z databáze, $Y = U^T X$

Trénování systému

1. Projekce obrázku tváře x na prostor tváří, $U^T x$.
2. Nalezení nejpodobnější tváře v databázi porovnáním $|U^T x_i, U^T x|_2$ pro všechny i = 1,...,n .
3. Klasifikace jako tvář od které má projekce x nejmenší eukleidovskou vzdálenost.

3.2 Vylepšení algoritmu

lemma 3.2.1 (O prostorové úspoře algoritmu)
Buď A reálná matice typu m x n. Pak kladná vlastní čísla matic AA^T a $A^T A$ jsou shodná. Navíc je-li $u_1, ..., u_k$ ortonormální posloupnost vlastních vektorů příslušných k nenulovým vlastním číslům $\lambda_1, ..., \lambda_k$ matice $A^T A$, pak ortonormální posloupnost vlastních vektorů v_i matice AA^T, příslušných k vlastním číslům $\lambda_i, i = 1, ..., k$ lze volit jako:

$$v_i = \frac{1}{\sqrt{\lambda_i}} A u_i$$

Dk :
Platí $A^T A u_i = \lambda_i u_i$ pro nějaké $i \in \{1, .., n\}$
Odsud také $AA^T A u_i = AA^T (A u_i) = \lambda_i A u_i$
Tedy Au_i je vlastním vektorem matice AA^T a k němu příslušné vlastní číslo je λ_i
Navíc:

$$v_i^T v_j = (\frac{1}{\sqrt{\lambda_i}} A u_i)^T \frac{1}{\sqrt{\lambda_j}} A u_j = \frac{1}{\sqrt{\lambda_i}} \frac{1}{\sqrt{\lambda_j}} (u_i^T A^T A u_j) = \frac{1}{\sqrt{\lambda_i}} \frac{1}{\sqrt{\lambda_j}} (u_i^T (A^T A u_j)) = \frac{1}{\sqrt{\lambda_i}} \frac{1}{\sqrt{\lambda_j}} (\lambda_j u_i^T u_j) = k$$

kde k je rovno 1 právě tehdy, když i = j a 0 kdykoliv jindy. Odsud je tedy $v_1, ..., v_n$ ortonormální posloupností vlastních vektorů matice $A^T A$.

V praxi [5] zpravidla platí, že $m << n^2$, tedy že pixelů je mnohonásobně více nežli databázových obrázků tváří. Je navíc velice realistické, že kovariační matice XX^T rozměrů n^2 x n^2 se nám nevejde na uložiště. Dle *lemmatu* 3.2.1 vidíme, že existuje mnohem prostorově úspornější způsob spočtení vlastních vektorů matice XX^T. Spočítáme- li totiž vlastní vektory matice $X^T X$, pak manipulujeme s maticí rozměrů m x m a vlastní vektory XX^T dostaneme pouhým přenásobením maticí X zleva, což nám velice usnadňuje práci.

3.3 Algoritmus z pohledu klasifikačního problému

Nyní, když jsem skončili s popisem algoritmu, tak se na něj podívejme z pohledu klasifikačního problému. Přihrádky jsou reprezentovány jako vektory $U^T x_i$, f je funkce přiřazující obrázku tváře jeho identitu. Trénovací množina je databáze tváří $x_1, \ldots, x_n$. Jako proces učení bychom označili hledání hlavních komponentů. Poznamenejme ještě, že náš algoritmus je jenom tak dobrý jako trénovací množina, tedy obecnou zásadou je sehnat co nejvíce různorodou sestavu tváří pro zajištění ideální funkčnosti extrakce příznaků.

3.4 Výhody x Nevýhody

Zakončeme naši práci diskuzí nad použitelností našeho algoritmu. PCA jakožto metoda, jak už jsme naznačili, má svá omezení. Hlavní omezení se nachází v linearitě hledané korelace. Existují další metody na extrakci příznaků (NLPCA,...), které si s nelinearitou jsou schopny poradit, často je však tato odměna vykoupena v úbytku na rychlosti. Ovšem právě rychlost, průhlednost a poměrně snadná implementovatelnost jsou hlavními výhodami Turkova a Pentlandova algoritmu. Většina dalších postupů je založená na neuronových sítích, jejichž trénování bývá obvykle velice časově náročné a při chybě bývá značně obtížné odhlavat původce. Největší slabiny Turkova a Pentlandova algoritmu jsou snadné zmatení pozadím obrázku, jelikož pozadí také ovlivňuje extrakci příznaků a poměrně silná paměťová náročnost spojená s uskladňováním kovarianční matice.

Výhody

1. Rychlost
2. Průhlednost
3. Snadná implementovatelnost

Nevýhody

1. Linearita
2. Zmatení pozadím obrázku
3. Paměťová náročnost

Reference

[1] Johan Shlens, Frank Mittelbach, and Alexander Samarin. *A tutorial on Principal Components Analysis*, 2005.

[2] Lindsay I Smith. *A tutorial on Principal Components Analysis*, 2002.

[3] Jeff Jauregui. *Principal component analysis with linear algebra*, University of Maryland, 2012.

[4] Matthew Turk, Alex Pentland. *Eiganfaces for recognition*, Massachusetts institute of technology, Journal of Cognitive Neuroscience, vol.3, no. 1, 2005.

[5] Kyungnam Kim. *Face Recognition using Principle Component Analysis*, University of Maryland.

[6] M. Kirby, L. Sirovich. *Face Recognition using Principle Component Analysis*, IEEE Transactions on pattern analysis and machine intelligence. vol.12, no. 1 , 1990.

[7] Dimitri Pissarenko. *Eigenface-based facial recognition* , 2002.

[8] Christopher M. Bishop. *Pattern recognition and machine learning*, Springer-Verlag New York Inc., 2006

[9] Miloš Oravec. *Metódy strojového učenia na extrakciu príznakov a rozpoznávanie vzorov*, 2012

[10] Richard Szeliski. *Computer Vision:Algorithms and Applications* , Springer-Verlag New York Inc., 2010

[11] Libor Barto, Jiří Tůma *Lineární algebra* , Univerzita Karlova, 2017.