Metodika analýzy dat: Od základů po aplikace metod strojového učení

Předzpracování dat

Jakub Steinbach, Jan Vrba

Ústav počítačové a řídicí techniky
VŠCHT

2.10.2024

Obsah slajdů I

Normalizace dat, transformace, kódování

Proč normalizovat a transformovat data?

- některé ML algoritmy mají lepší výsledky s daty, která jsou na stejných škálách, případně mají standardizované normální rozdělení
- vyšší stabilita učení
- rychlejší konvergence
- snadnější porovnávání různých datasetů
- povinnost při použití algoritmů, které závisí na měření vzdáleností (euklidovské, manhattanské,..) jako KNN, NN, GD, PCA, K-means

Základní očištění dat a jejich převedení do číselné podoby

- odstranění nepoužitelných hodnot z datasetu (NaN, Inf, případně poškozené a chybějící záznamy)
- odstranění formátovacích znaků a symbolů (konce řádek, symboly měn,...)
- převod nečíselných hodnot na číselné (např. rok narození na věk, převod textových řetězců na vektory,...)
- přizpůsobení velikosti vstupu (např. změna rozlišení obrázku, převzorkování signálu,...)

$$\Longrightarrow \textbf{škálovací transformace}$$

Min-max normalizace

- transformace která nemění typ distribuce dat
- škálování na interval $[0, 1]$

$$x' = \frac{x - \min(x)}{\max(x) - \min(x)} \tag{1}$$

- škálování na libovolný interval $[a, b]$

$$x' = a + \frac{(b - a)(x - \min(x))}{\max(x) - \min(x)} \tag{2}$$

- **použití**: když známe hodnoty mezí (např. intenzita pixelů),řádově rozdílné features nebo máme hodnoty features s různými negaussovskými distribucemi
- v praxi pomocí:

```
sklearn.preprocessing.MinMaxScaler
```

Standardizace dat

- transformace, která zajistí, že featura bude mít $\mu = 0$ a $\sigma^2 = 1$
- nemění distribuci dat!!

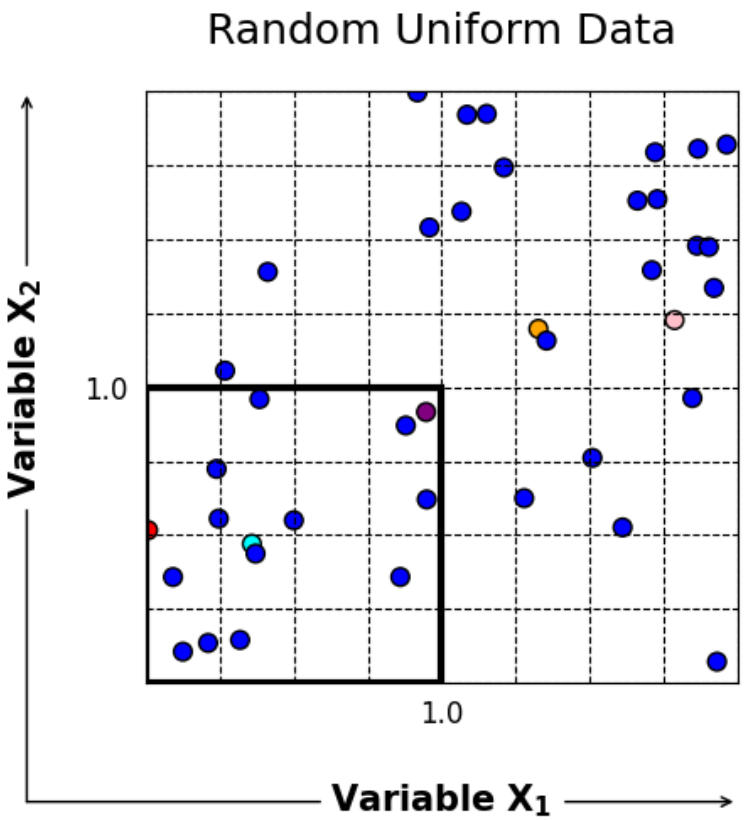
$$x' = \frac{x - \mu}{\sigma} \tag{3}$$

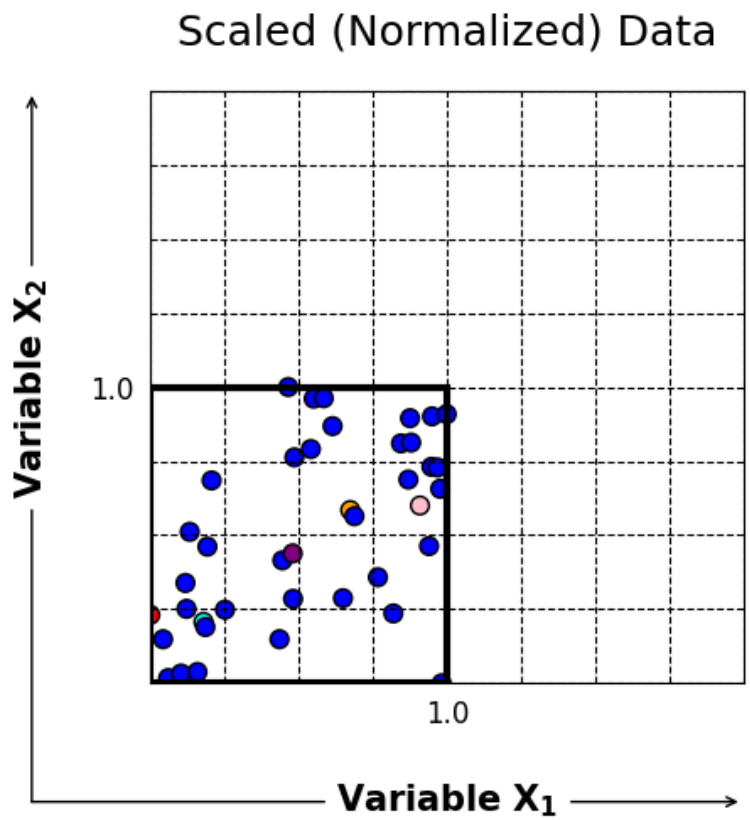
- narozdíl od normalizace, není tolik ovlivněna outlierama
- interpretace: translace dat a změna jejich měřítka
- **použití**: potlačení outlierů, různé fyz. jednotky, různé rozsahy, data s normálním rozdělením, algoritmy co jsou postavené na normální distribuci
- v praxi pomocí:

```
sklearn.preprocessing.StandardScaler
```

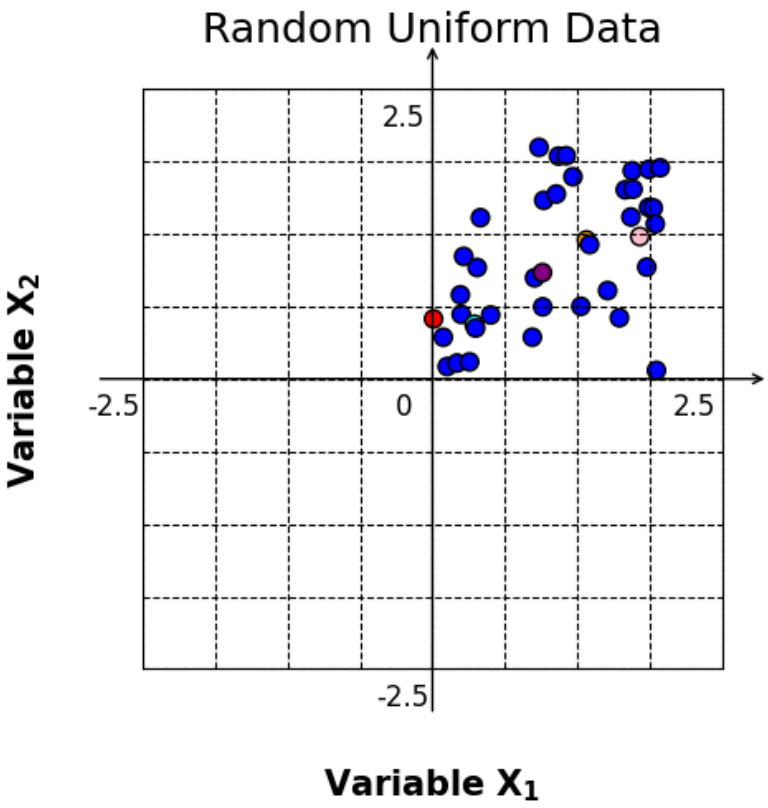
Př: škálování výšky na metry a hmotnost na kilogramy... hmotnost bude výrazně ovlivňovat výsledek

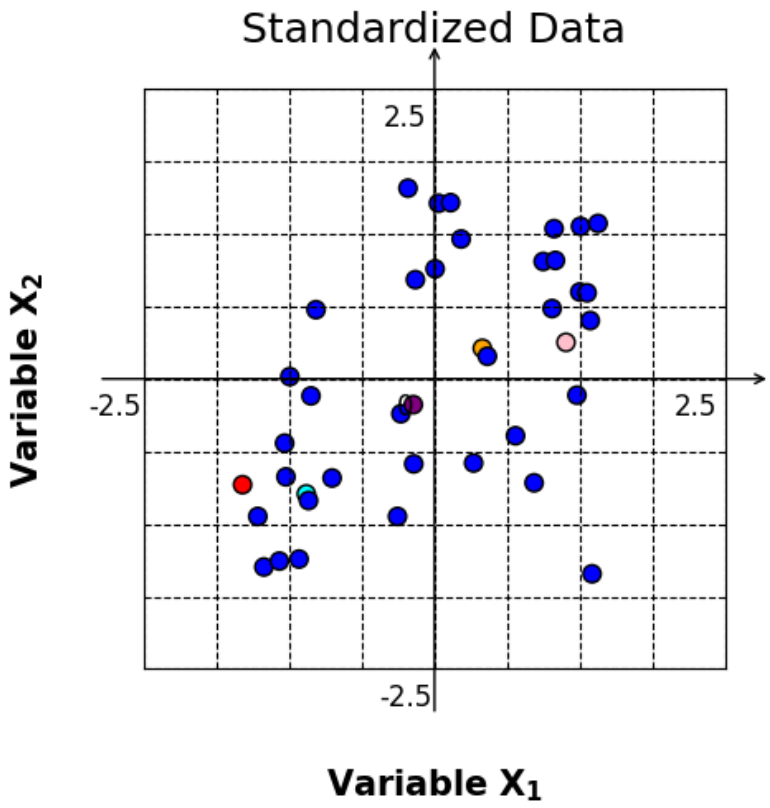
Normalizace vs Standardizace





Normalizace vs Standardizace





Log trasformace

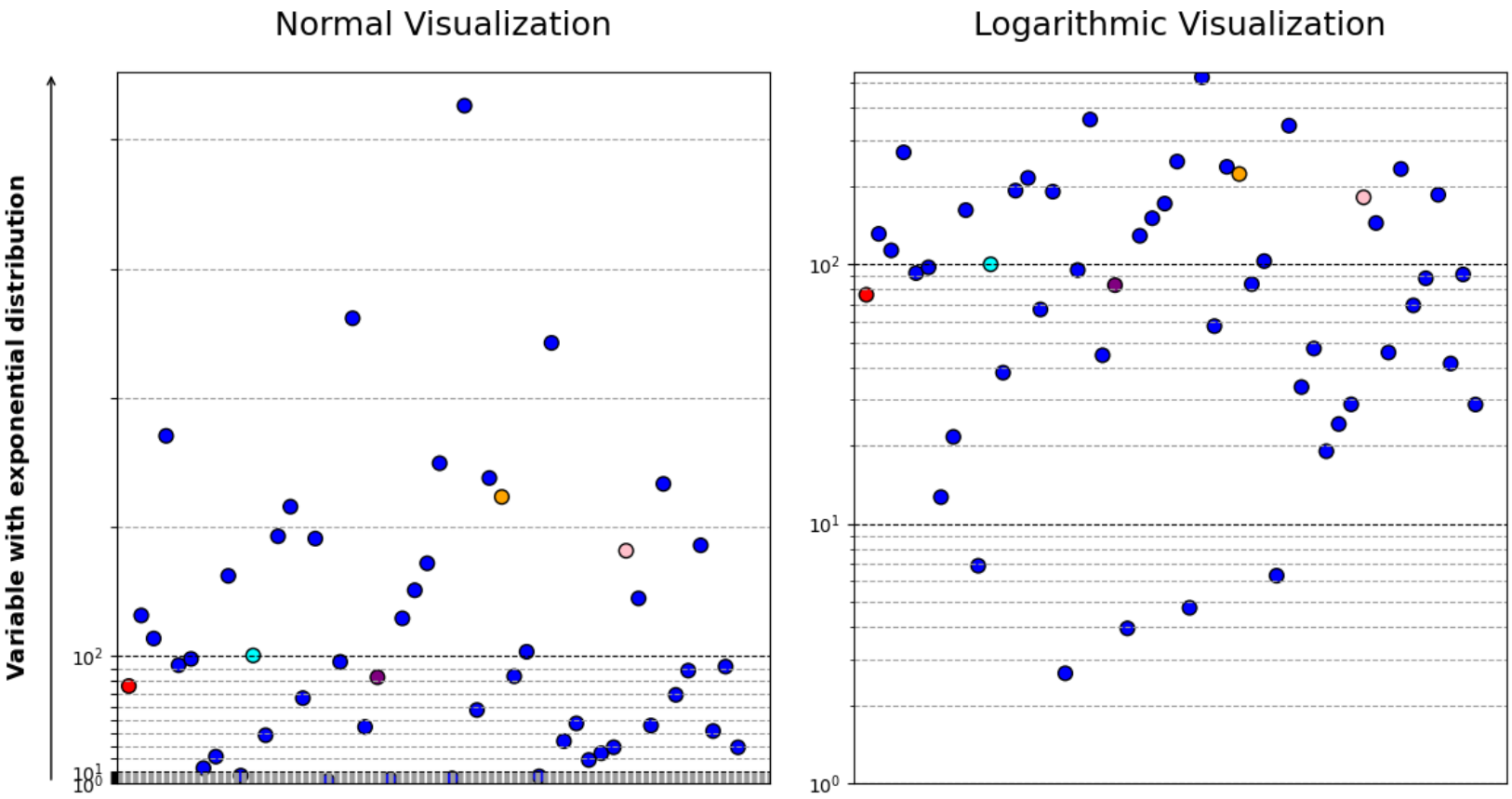
- ze zkosené distribuce (vpravo) udělá méně zkosenou
- ideální pro data s lognormální distribucí

$$x' = \ln x \tag{4}$$

- **použití:** v případě, že featura má rozsah několika řádů (např. příjmy, GDP,..), data nejsou záporná nebo nejsou z intervalu $(0, 1)$
- v praxi:

```
np.log(data)
```

Log trasformace



MaxAbs transformace

- škálování dat na interval $[-1, 1]$
- citlivá na přítomnost outlierů
- zachovává řídkost dat (velkým počtem nulových hodnot)

$$x' = \frac{x}{\max |x|} \tag{5}$$

- **použití**: pro centrovaná nebo řídká data
- v praxi:

```
sklearn.preprocessing.MaxAbsScaler
```

Robust Scaler

- odstraní medián a škáluje podle 1. a 3. kvartilu

$$x' = \frac{x - x_{median}}{x_{75} - x_{25}}$$

- někdy se pro výpočet používá i horní a dolní decil

$$x' = \frac{x - x_{median}}{x_{90} - x_{10}}$$

- **použití**: vhodná transformace pro data zatížená outliery, nevhodná pro velice řídká data!!!
- v praxi:

```
sklearn.preprocessing.RobustScaler
```

Unit scaler

- metoda, která hodnoty featury naškáluje tak, aby euklidovská velikost vektoru byla $||x|| = 1$

$$x' = \frac{x}{||x||}$$

- pokud ML algoritmus pracuje s jinou metrikou (např. Manhattan distance), lze použít odpovídající škálování
- zásadní pro shlukovací algoritmy nebo NN v kombinaci s gradient descentem
- standardní metoda pro klasifikaci textů a NLP
- v praxi:

```
sklearn.preprocessing.Normalizer
```

Box-Coxova transformace

- transformace sloužící ke "znormálnění"distribuce dat (kladných hodnot - pokud nejsou, přičteme konstantu)

$$x_i' = \begin{cases} \log(x_i) & \lambda = 0 \\ \frac{x_i^\lambda - 1}{\lambda} & \lambda \in \langle -5; 5 \rangle \setminus \{0\} \end{cases}$$

- **problém:** jak najít optimální hodnotu lambda, s omezením $\lambda \in \langle -5; 5 \rangle$
- **řešení:** metodou maximální věrohodnosti (μ, σ jsou stat. transformovaných dat)

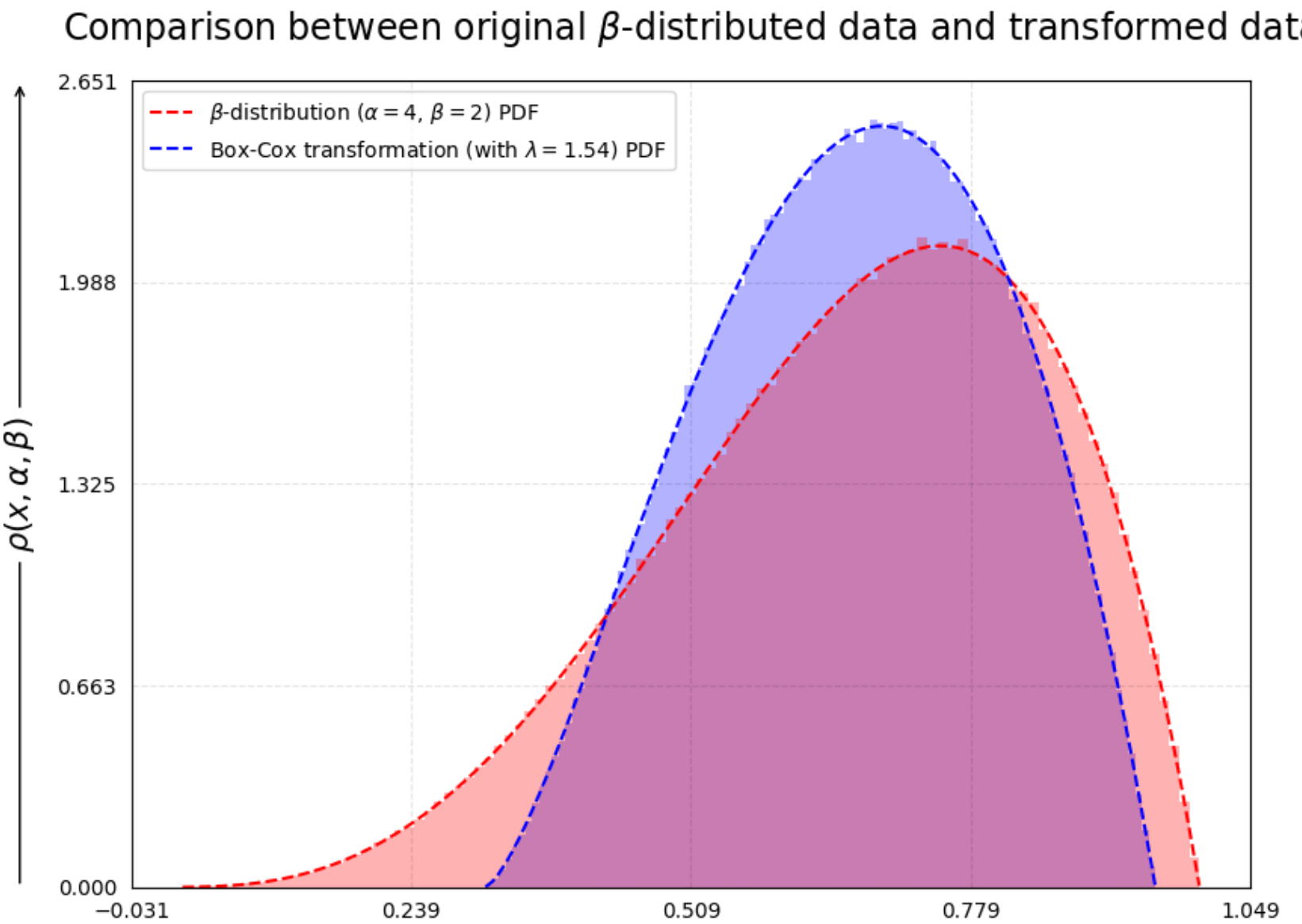
$$\max_{\lambda \in \langle -5;5 \rangle} \left(\frac{n}{2}(-\ln(2\pi) + \ln \sigma^2) - \frac{1}{2\sigma^2} \sum_{i=1}^{m} (x_i' - \mu)^2 + (\lambda - 1) \sum_{i=1}^{m} \ln x_i \right)$$

- v praxi pomocí:

```
scipy.stats.boxcox(x)
```

- **pozor:** Box-Coxova transformace nezaručuje normalitu, pouze minimalizuje směrodatnou odchylku σ souboru dat!!

Box-Coxova transformace - ukázka



Yeo-Johnson transformace

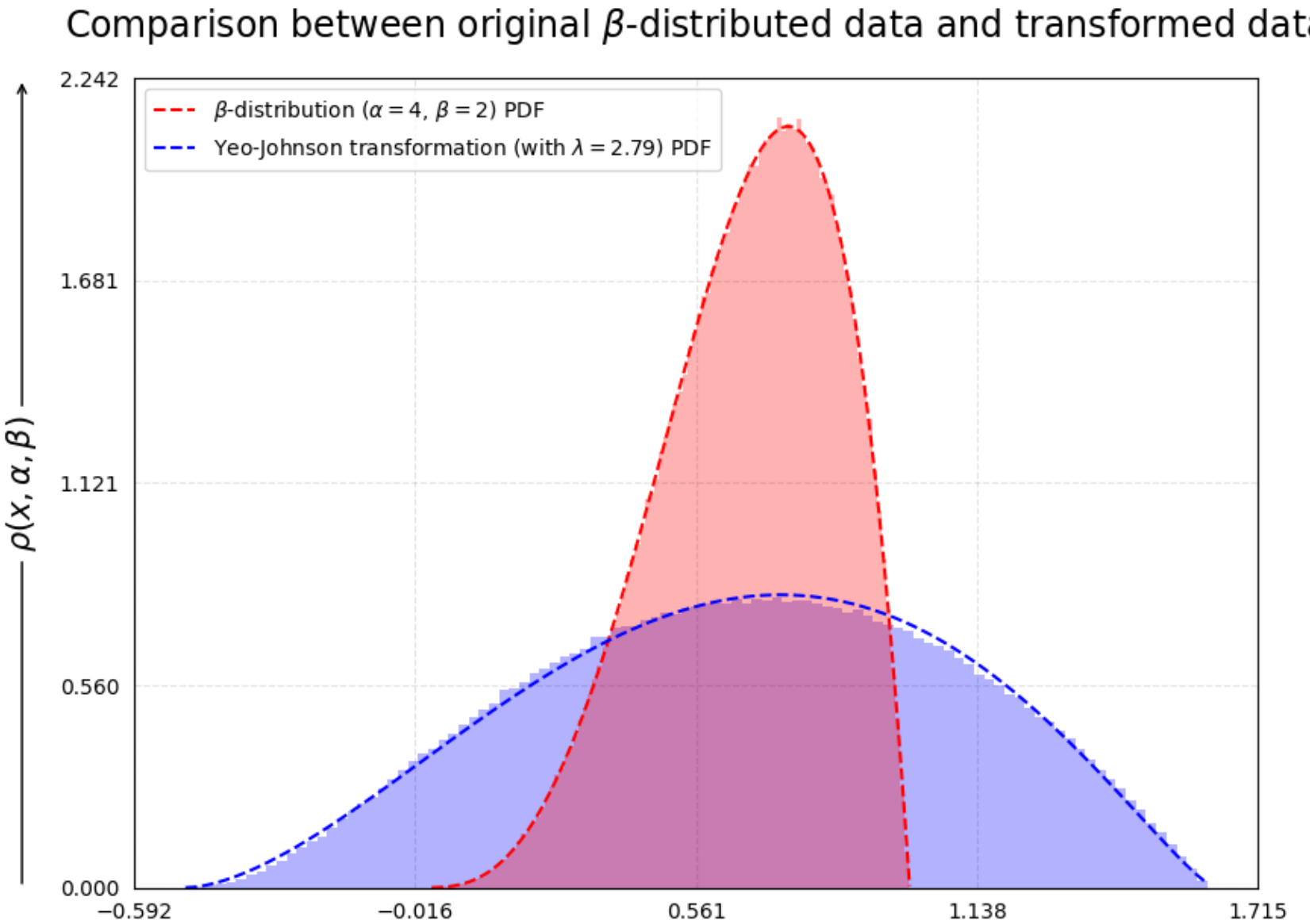
- alternativa k Box-Coxově transformaci
- není omezena nezáporností hodnot x

$$x' = \begin{cases} \frac{(x+1)^{\lambda}-1}{\lambda} & \lambda \neq 0, x \geq 0 \\ \ln(x+1) & \lambda = 0, x \geq 0 \\ -\frac{(-x+1)^{2-\lambda}-1}{2-\lambda} & \lambda \neq 2, x < 0 \\ -\ln(-x+1) & \lambda = 2, x < 0 \end{cases}$$

- pokud jsou hodnoty x pouze záporné, je to Box-Cox transformace $(-x+1)$ s hodnotou $\lambda - 2$
- pro kladná x je to Box-Cox transformace $(x+1)$
- pro $\lambda = 1$ je to identická transformace
- v praxi pomocí:

```
scipy.stats.yeojohnson(x)
```

Yeo-Johnson transformace



Ořezávání (clipping)

- metoda pro potlačení odlehlých hodnot

$$x' = \begin{cases} x_{max} & x \geq x_{max} \\ x & x \in (x_{min}, x_{max}) \\ x_{min} & x \leq x_{min} \end{cases} \tag{6}$$

- někdy se využívá i pro potlačeních velkých hodnot gradientu a zlepšení konvergence
- v praxi:

```
np.clip(data, data_min, data_max)
```

Nominální a ordinální featury

- **nominální (jmenná) featura**
  - rovnocené varianty, nelze srovnávat ani seřadit (např.rodinný stav, místo narození, značka auta, pohlaví, živočišný druh,..)
  - nedají se s nimi provádět početní operace (např. sčítání názvu města)
- **ordinální featura**
  - jednotlivým variantám proměnné lze přiřadit pořadí (např. známka u zkoušky, hodnotící škály typu: velmi často, často, občas, moc ne, vůbec ne)
  - varianty lze mezi sebou porovnávat

Label encoding

- místo kategorické hodnoty je přiřazena celé číslo mezi 0 a počtem kategorických hodnot -1
- kategorické hodnoty se obvykle seřadí podle abecedy a jejich index odpovídá kódové hodnotě
- problém: vysoké hodnoty kategorie můžou vést ke zvýšení její důležitosti, vzniká možnost porovnávat hodnoty mezi sebou
- v praxi:

```
sklearn.preprocessing.LabelEncoder
```

Stav	Label encoding	Label encoding (alternativní)
Svobodná/ý	1	0
Vdaná/Ženatý	2	1
Rozvedená/ý	0	2

One hot encoding

- pro každou možnou hodnotu kategorické proměnné se zavede nová dummy featura
- nová featura nabývá pouze hodnot 0 a 1
- **problém:** výrazný nárůst dimenzionality
- v praxi:

```
sklearn.preprocessing.OneHotEncoder
```

zvíře	hmotnost
kočka	4
pes	11
opice	8
kočka	5

hmotnost	kočka	pes	opice
4	1	0	0
11	0	1	0
8	0	0	1
5	1	0	0

Frequency encoding

- frekvence výskytu (resp. jejich relativní četnost) kategorie v datovém souboru je použita pro kódování

Triplet	Target	Freq. encoding
TAA	1	0,4
TAA	0	0,4
TAA	1	0,4
TAA	0	0,4
GTT	1	0,3
GTT	1	0,3
GTT	1	0,3
GTA	1	0,2
GTA	0	0,2
AGG	0	0,1

- TTA: $4/10 = 0.4$, GTT $3/10 = 0.3$,...

Mean encoding

- kódování založené na střední hodnotě výstupní proměnné (target) asociované s kódovanou hodnotou featury (triplet)

Triplet	Target	Mean encoding
TAA	1	0,5
TAA	0	0,5
TAA	1	0,5
TAA	0	0,5
GTT	1	1
GTT	1	1
GTT	1	1
GTA	1	0,5
GTA	0	0,5
AGG	0	0

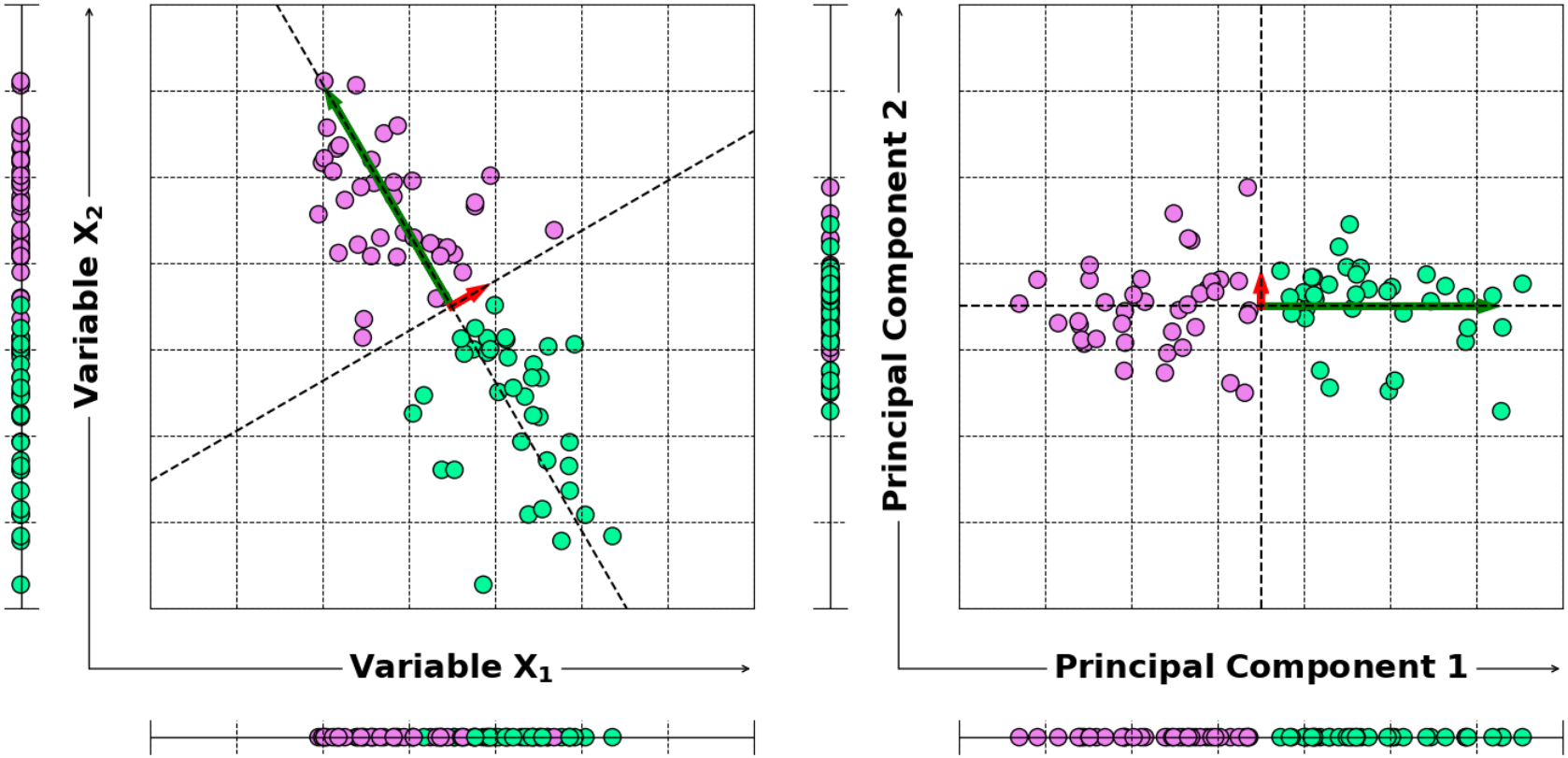
- TAA: $(1+0+1+0)/4-0,5$, GTT: $(1+1+1)/3=1$, ...

PCA - redukce dimenze dat (extrakce příznaků)

- typické problémy v ML: málé nebo velké množství dat, šum
- vysoká dimenzionalita dat $\Longrightarrow$ velká výpočetní náročnost, náchylnost k přeučení, často řídké featury, nedostatečná generalizace modelů
- otázky:
  - lze snížit dimenzi problému?
  - můžeme se na data podívat jinak?

PCA

PCA - redukce dimenze dat (extrakce příznaků)



PCA - redukce dimenze dat

- z množiny vycentrovaných dat $\mathbf{X} \in \mathcal{R}^{m \times n}$ chceme udělat množinu dat $\mathbf{Z} \in \mathcal{R}^{m \times d}$
- hledáme ortogonální transformaci $\mathbf{V} \in \mathcal{R}^{d \times n}$ ($\mathbf{V}^T\mathbf{V} = \mathbf{I}$), aby $\mathbf{Z} = \mathbf{XV}$
- požadované vlastnosti $\mathbf{Z}$
  1. d-dimenzionální podprostor je nejlepší možný pro reprezentaci $\mathbf{X}$ ve smyslu

$$\min_{\mathbf{V},\mathbf{Z}} \sum_{i=1}^{m} ||x_i - \mathbf{V}z_i||_2^2,$$

     s omezením

$$\mathbf{V}^T\mathbf{V} = \mathbf{I}$$

  2. $\mathbf{Z}$ zachová maximální množství variability (rozptylu)

PCA - redukce dimenze dat

- sloupcové vektory transformační matice $\mathbf{V} = (v_1, \ldots, v_d)$ jsou tzv. hlavní komponenty, i-tý komponent představuje lineární funkci $f(x) = v_i^T x$
- najdeme postupně vektory $||v_i|| = 1$ (vzájemně kolmé) tak aby byl maximalizován rozptyl projekce bodů z $\mathbf{X}$ na příslušný jednodimenzionální podprostor
  1. pro první hlavní komponentu

$$\max_{v_1} var(Xv_1) = \max_{v_1} v_1^T C v_1 = \max_{v_1} v_1^T X^T X v_1$$

     kde C je kovarianční matice ($C = \frac{1}{m-1} X^T X$)
  2. pro k-tou komponentu (pozor, musíme mít vypočítán předchozí komponenty)

$$\max_{v_k} v_k^T X^T X v_k$$

     s.t.

$$||v_k|| = 1, \quad v_j^T v_k = 0, \quad j = 1, \ldots, k-1$$

- řešení: vektor v_i je vlastní vektor příslušný i-tému největšímu vlastnímu číslu λ_i kovarianční matice C

PCA - vybrané vlastnosti

- celkový rozptyl všech hlavních komponent

$$\mathrm{Tr}\,\mathbf{C} = \sum_{i=1}^{m} \lambda_i$$

- podíl variability objasněný i-tou hlavní komponenou

$$P_i = \frac{\lambda_i}{\mathrm{Tr}\,\mathbf{C}}$$

Maximalizace variance

- délka projekce $\mathbf{x}_j$ ve směru jednotkového vektoru $\mathbf{v}$

$$y_j = \mathbf{x}_j^T \mathbf{v}$$

- střední hodnota délek druhých mocnin projekcí všech bodů $\mathbf{x}_i$ je variance $\mathbf{V}$

$$\mathbf{V} = \frac{1}{m}\sum_{i=1}^{m} y_i^2 = \frac{1}{m}\sum_{i=1}^{m}(\mathbf{x}_i^T\mathbf{v})^2 = \frac{1}{m}\sum_{i=1}^{m}(\mathbf{x}_i^T\mathbf{v})\cdot(\mathbf{v}^T\mathbf{x}_i) =$$

$$= \frac{1}{m}\sum_{i=1}^{m}\mathbf{v}^T\mathbf{x}_i\cdot\mathbf{x}_i^T\mathbf{v} = \mathbf{v}^T\left(\frac{1}{m}\sum_{i=1}^{m}\mathbf{x}_i\mathbf{x}_i^T\right)\mathbf{v} = \mathbf{v}^T\mathbf{C}\mathbf{v} \quad (7)$$

pozn. pro nevychýlený odhad kovariance (v praxi) použijeme $\frac{1}{m-1}$

Maximalizace variance - nalezení první komponenty

- maximalizujeme $\mathbf{v}_1^T \mathbf{C} \mathbf{v}_1$ s omezením $\mathbf{v}_1^T \mathbf{v}_1 = 1$
- příslušný Lagrangian

$$\mathcal{L}(\mathbf{x}, \lambda) = \mathbf{v}_1^T \mathbf{C} \mathbf{v}_1 - \lambda(\mathbf{v}_1^T \mathbf{v}_1 - 1)$$

- řešíme problém

$$\left\{ \frac{\partial \mathcal{L}(\mathbf{v}_1, \lambda)}{\partial \mathbf{v}_1} = 0, \frac{\partial \mathcal{L}(\mathbf{w}_1, \lambda)}{\partial \lambda} = 0 \right\}$$

$$\frac{\partial \mathcal{L}(\mathbf{x}, \lambda)}{\partial \mathbf{v}_1} = \mathbf{v}_1^T (\mathbf{C} + \mathbf{C}^T) - 2\lambda \mathbf{v}_1^T = 2\mathbf{v}_1^T \mathbf{C} - 2\lambda \mathbf{v}_1^T = 0$$

$$\frac{\partial \mathcal{L}(\mathbf{v}_1, \lambda)}{\partial \lambda} = \mathbf{v}_1^T \mathbf{v}_1 - 1 = 0$$

- z první rovnice dostaneme

$$\mathbf{C} \mathbf{v}_1 = \lambda \mathbf{v}_1$$

- protože $\mathbf{v}_1^T \mathbf{C} \mathbf{v}_1 = \lambda$, bude první hlavní komponenta vlastní vektor

PCA - nalezení druhé komponenty

- maximalizujeme $\mathbf{v}_2^T \mathbf{C} \mathbf{v}_2$ s omezeními $\mathbf{v}_2^T \mathbf{v}_2 = 1$ a $\mathbf{v}_2^T \mathbf{v}_1 = 0$
- příslušný Lagrangian

$$\mathcal{L}(\mathbf{v}_2, \lambda, \Psi) = \mathbf{v}_2^T \mathbf{C} \mathbf{v}_2 - \lambda(\mathbf{v}_2^T \mathbf{v}_2 - 1) - \Psi \mathbf{v}_2^T \mathbf{v}_1$$

- řešíme rovnici

$$\frac{\partial \mathcal{L}(\mathbf{v}_2, \lambda, \Psi)}{\partial \mathbf{v}_2} = 2\mathbf{v}_2^T \mathbf{C} - 2\lambda \mathbf{v}_2 - \Psi \mathbf{v}_1 = 0$$

$$2\mathbf{v}_1^T \mathbf{C} \mathbf{v}_2 - 2\lambda \mathbf{v}_1^T \mathbf{v}_2 - \Psi \mathbf{v}_1^T \mathbf{v}_1 = 0 \implies \Psi = 0$$

- pro $\Psi = 0$ řešíme problém

$$\mathbf{C} \mathbf{v}_2 = \lambda \mathbf{v}_2$$

- protože $\lambda \neq \lambda_1$, bude druhá hlavní komponenta vlastní vektor (jednotkový) asociovaný s druhým největším vlastním číslem příslušné kovarianční matice

PCA - algoritmus

1. standardizace dat

$$X_j = \frac{x_j - \mu_j}{\sigma_j}$$

2. výpočet kovarianční matice

$$C_{X_i X_j} = cov[X_i, X_j] = E[(X_i - E[X_i])(X_j - E[X_j])]$$

$$C_{xx} = \begin{bmatrix} var(X_1) & cov(X_1, X_2) & \dots & cov(X_1, X_n) \\ cov(X_1, X_2) & var(X_2) & \dots & cov(X_2, X_n) \\ \vdots & \vdots & \ddots & \vdots \\ cov(X_1, X_n) & \dots & \dots & var(X_n) \end{bmatrix}$$

PCA - redukce dimenze dat

3. výpočet vlastních čísel a vlastních vektorů

$$C_{xx} v_i = v_i \lambda_i$$

   a jejich setřídění $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_d \geq \cdots \geq \lambda_n$

4. sestavení transformační matice

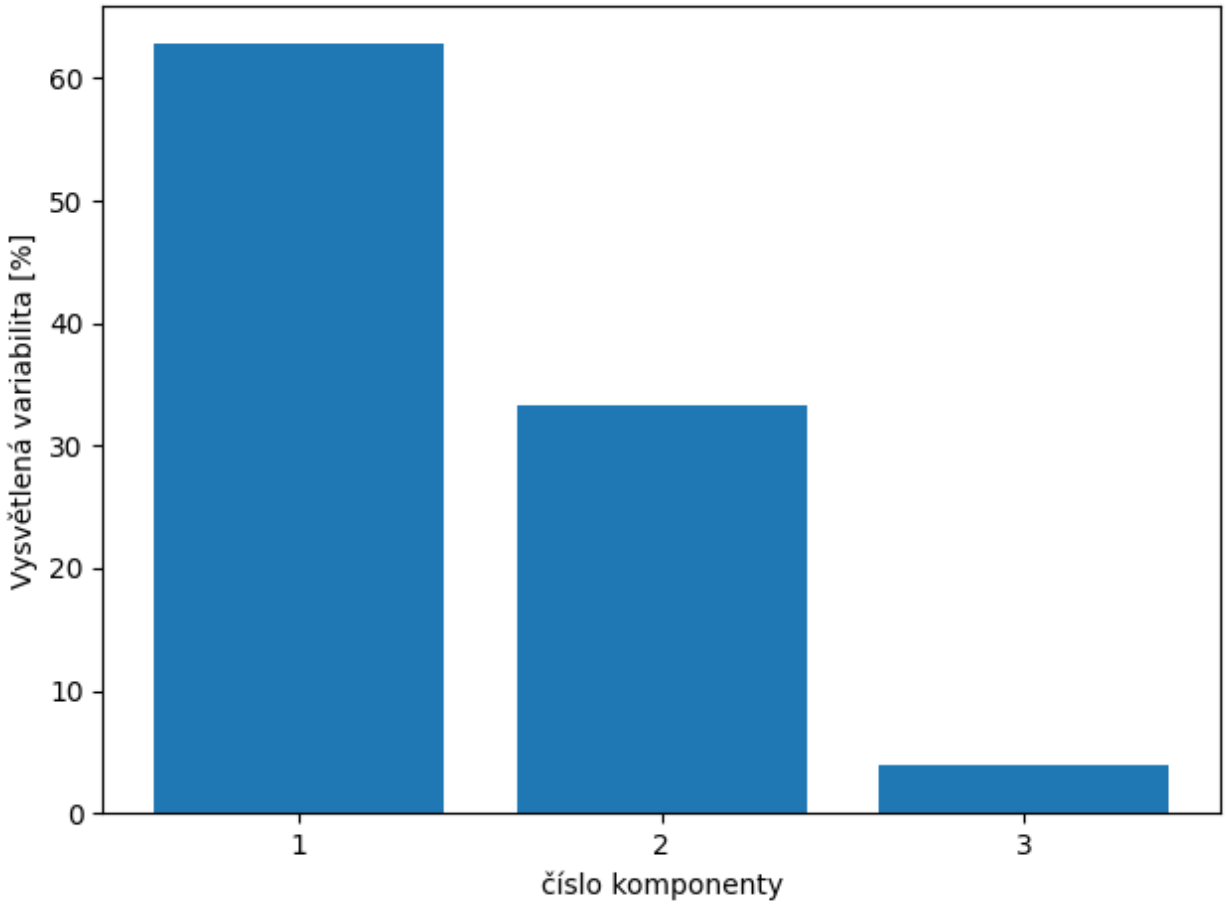
$$V = \begin{bmatrix} v_1 & v_2 & \dots & v_d \end{bmatrix}$$

5. výpočet nových souřadnic bodů v prostoru redukované dimenze

$$Z = X \cdot V$$

Graf úpatí

- sloupcový graf uspořádanách vlastních čísel, resp. podílu hlavních komponent na vysvětlení celkové variability $\Longrightarrow$ dobrá vizualizace důležítých a nepodstatných hlavních komponent



PCA - interpretace a aplikace

PCA interpretace:

- centrování dat a změna jejich směrodatné odchylky
- otoč data
- zapomeň dimenze

PCA aplikace:

- extrakce příznaků
- komprese
- odstranění šumu

Volba d

Kaiserovo pravidlo

Použijeme k komponent, které splňují podmínku

$$\lambda_k > \frac{1}{n} \sum_{i=1}^{n} \lambda_i$$

Kaiser-Guttmanovo kritérium

Použijeme komponenty pro jejichž odpovídající vlastní číslo platí, že $\lambda \geq 1$.

Broken stick model

Použijeme všechny komponenty, pro které platí

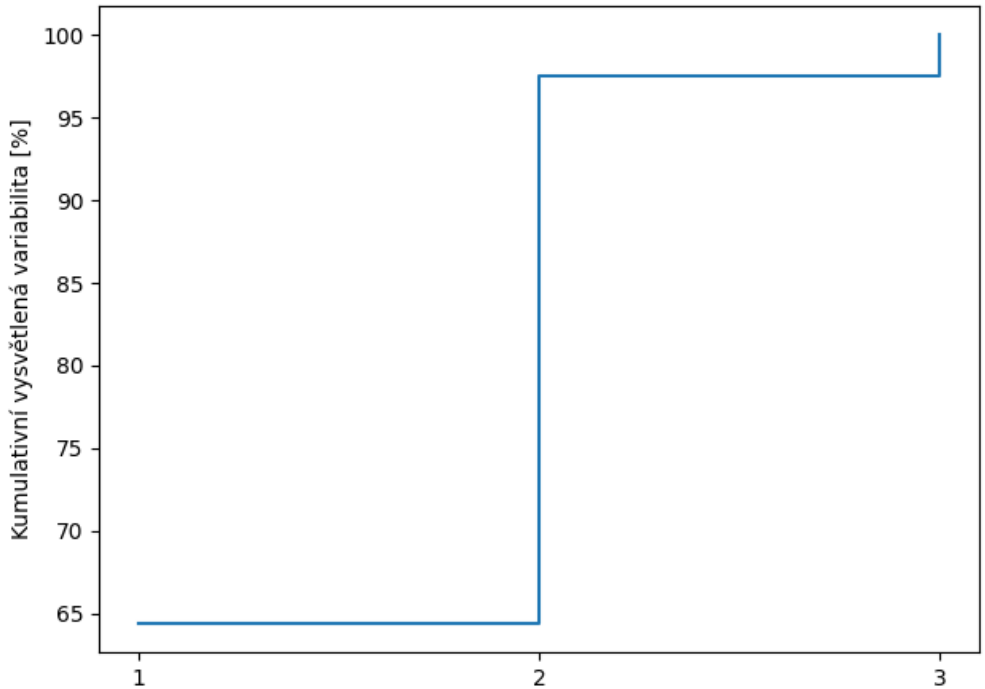
$$\lambda_i > \frac{1}{n} \sum_{j=i}^{n} \frac{1}{j}, \quad j = i, \ldots, n, \quad i = 1, \ldots, n$$

Volba d

70% - 90%

Použijeme k komponent tak, aby

$$\sum_{i=1}^{k} P_i > 0.7 \cdot \mathrm{Tr}\,\mathbf{C}$$



PCA - implementace

```
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler

scaled_data = StandardScaler().fit_transform(raw_data)
pca = PCA(n_components = PC_NUMBER)
pca.fit(scaled_data)
print(pca.components_)
print(pca.explained_variance_)
print(pca.singular_values_)
print(pca.mean_)
print(pca.noise_variance_)
```

PCA - omezení

- PCA funguje dobře pro features mezi kterými existuje lin. závislost (je to lin. transformace)
- outliers výrazně ovlivní výsledné hlavní komponenty
- PCA je citlivá na škálování příznaků
- pozor na nepoužitelnost pro kategorické proměnné resp. diskrétní proměnné