Kako videti n dimenzija?

Nikola Jovanović

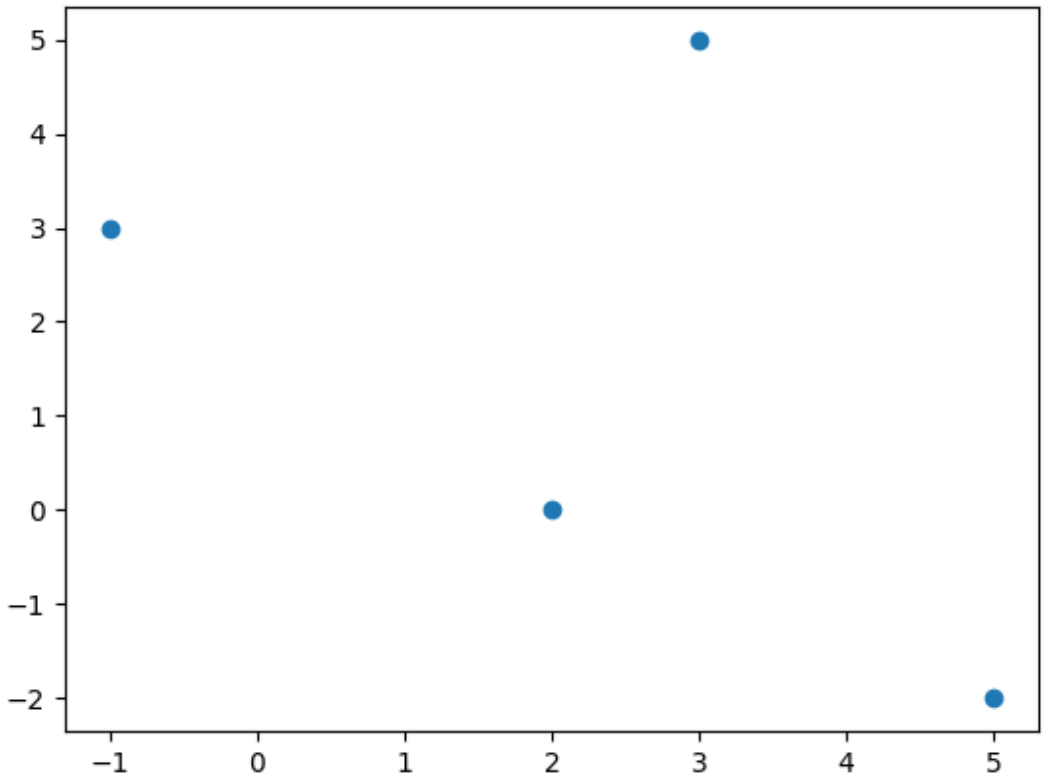
Matematička gimnazija

NEDELJA$^4_{\text{INFORMATIKE}}$

27. mart 2018.

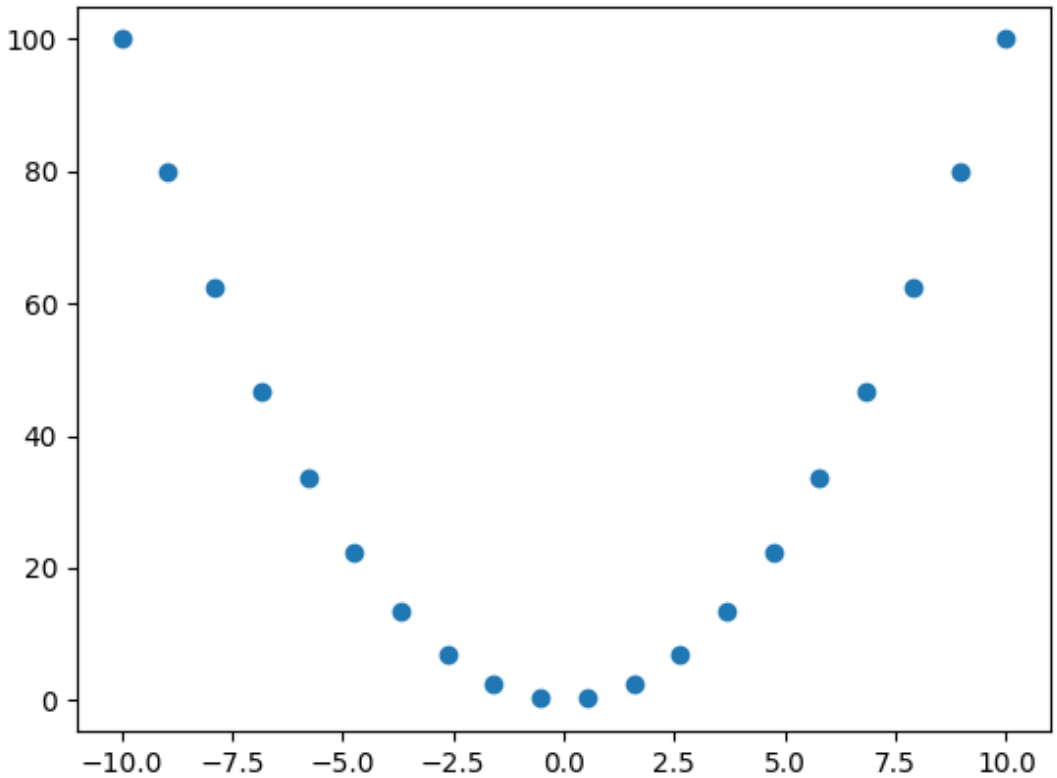
Kako videti dve dimenzije?

- Imamo niz 2D tačaka: $(-1, 3), (2, 0), (3, 5), (5, -2)$
- Lako prikazati (*scatter plot*):



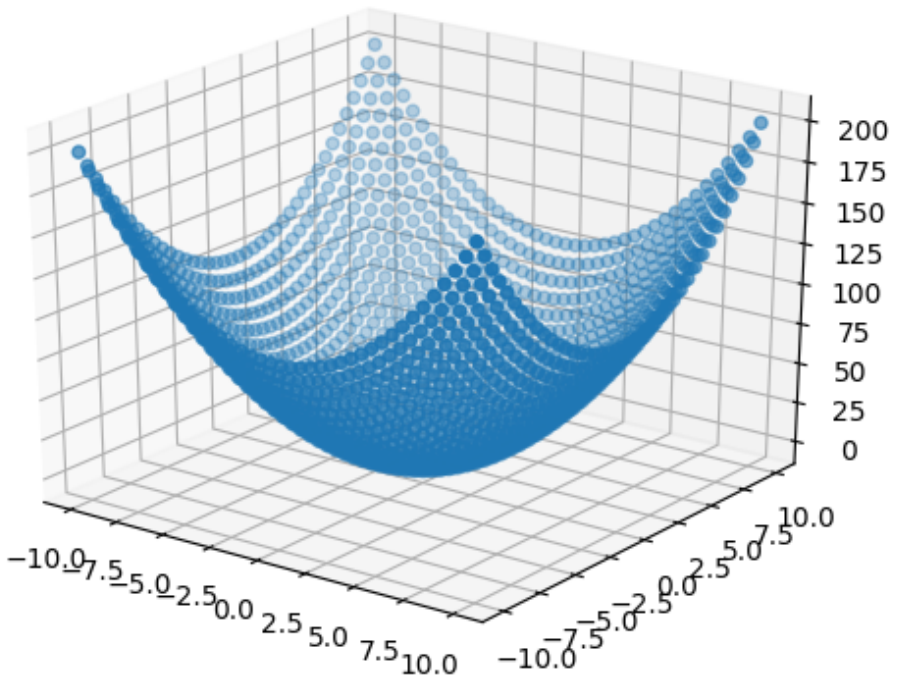
Kako videti dve dimenzije?

- Uzorkujemo funkciju: $y = x^2$
- Lako prikazati (*scatter plot*):



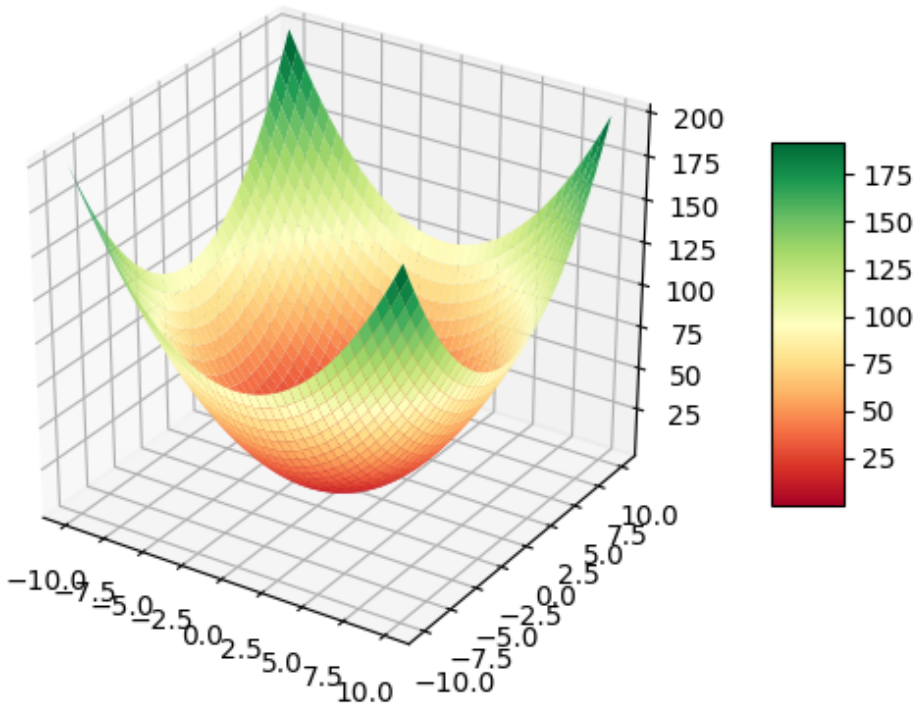
Kako videti tri dimenzije?

- Uzorkujemo funkciju: $z = x^2 + y^2$
- Lako uopštiti (*3D scatter plot*):



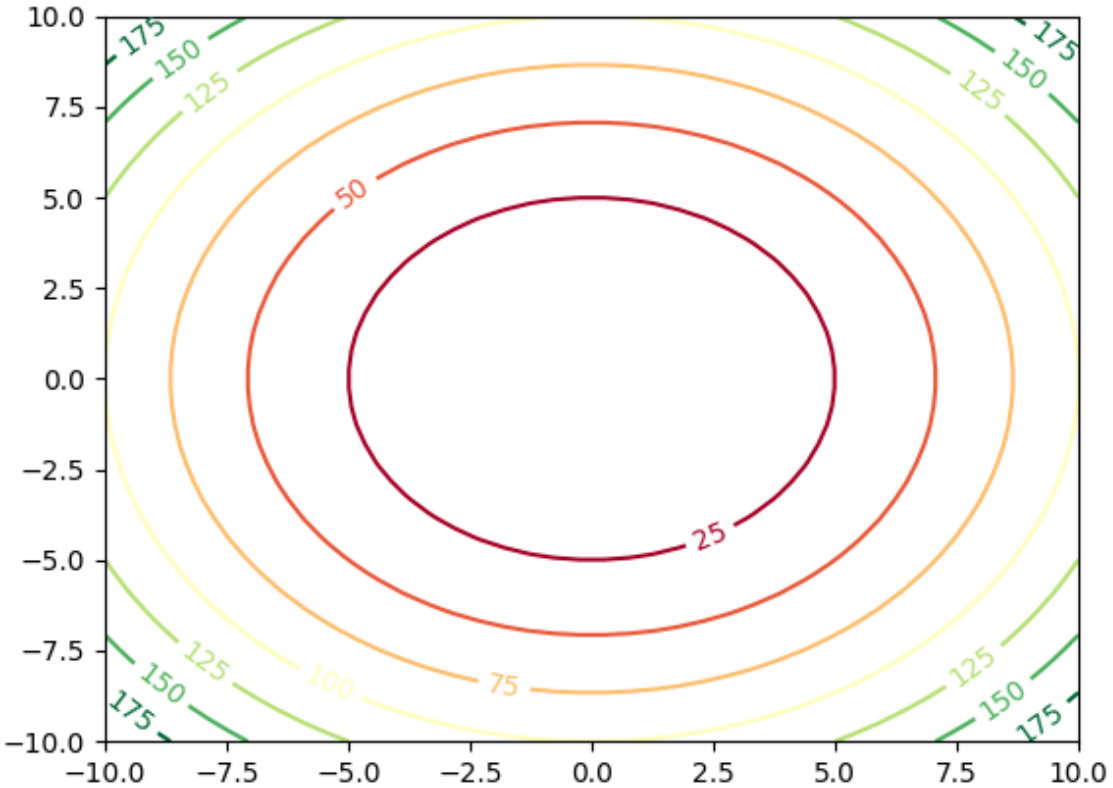
Kako videti tri dimenzije?

- Uzorkujemo funkciju: $z = x^2 + y^2$
- Alternativa, *surface plot* sa kolor mapom:



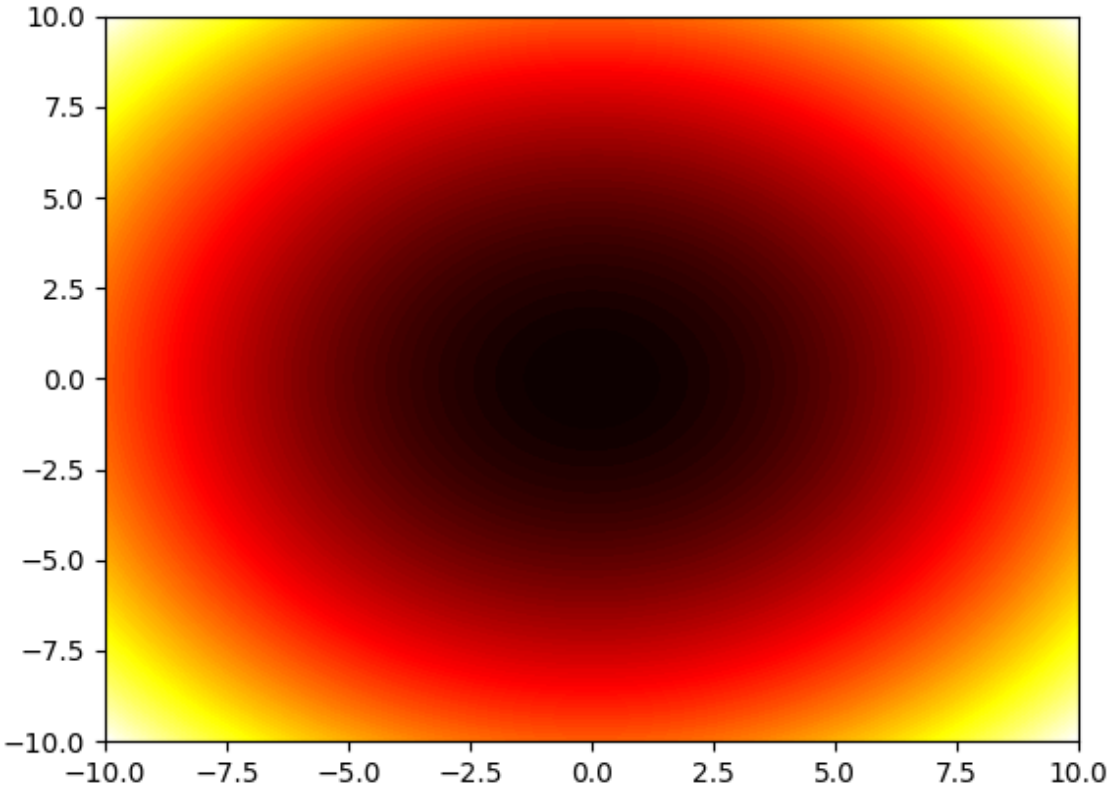
Kako videti tri dimenzije?

- Uzorkujemo funkciju: $z = x^2 + y^2$
- Alternativa, 2D *contour plot*:



Kako videti tri dimenzije?

- Uzorkujemo funkciju: $z = x^2 + y^2$
- Alternativa, 2D *heatmap*:



Kako videti četiri dimenzije?

Kako videti četiri dimenzije?

- Dosta teže
- Moguće je istraživati 4D ali nije jednostavno

Kako videti n dimenzija?

- Praktično nemoguće
- Da li nam je ovo zaista potrebno?
  - Svojstva (feature-i): jednog učenika karakterišu visina, težina, pol, datum rođenja, razred, ocene...
  - Slike (jedan piksel - jedna dimenzija)
- Rešenje: **redukcija dimenzija** (*dimensionality reduction*)

Redukcija dimenzija

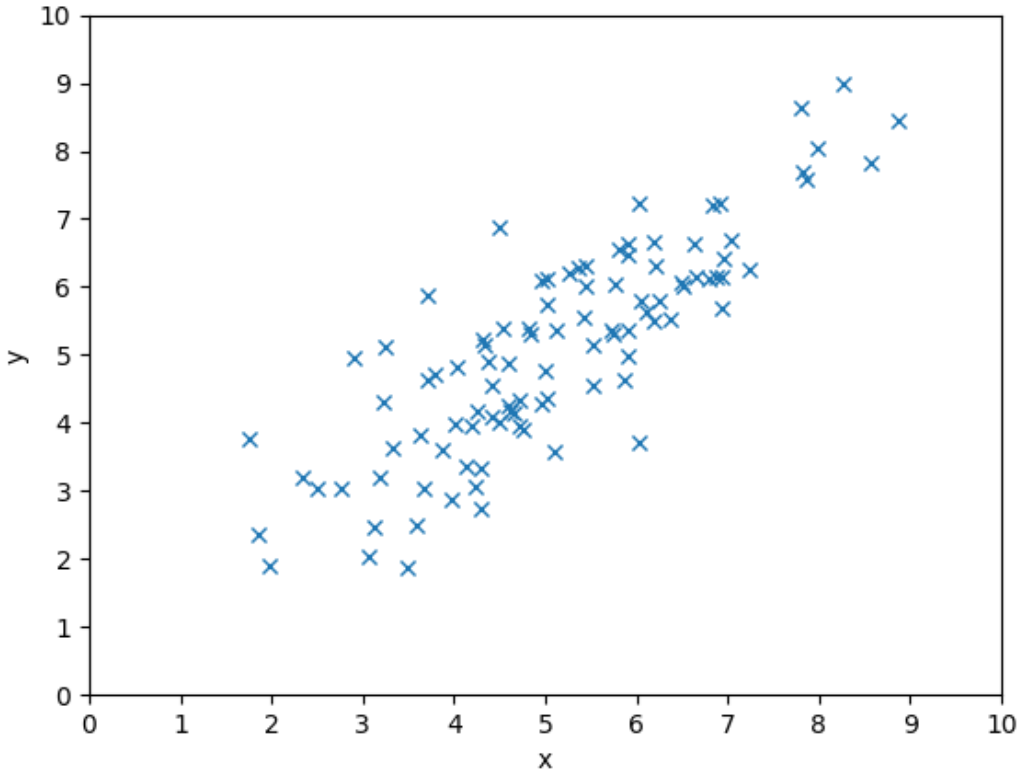
- Želimo skup tačaka iz $\mathbb{R}^n$ da što vernije predstavimo u $\mathbb{R}^d$, gde je $d < n$
- Krenuli smo od vizualizacije: za $d \leq 3$ možemo izvršiti prikaz, ovo je bitno jer nam omogućava da steknemo intuiciju o strukturi podataka u velikom broju dimenzija
- To je često glavna motivacija ali redukcija dimenzija ima primena i kada je $d > 3$:
  - Ušteda memorije
  - Kompresija
  - Ubrzavanje algoritma

Redukcija dimenzija

- Dve klase
  - *Feature selection*: direktno biramo „najbitnijih” d dimenzija (nekoliko poznatih tehnika, manje zanimljivo)
  - *Feature reduction*: proizvoljnom metodom transformišemo tačke u $\mathbb{R}^d$ (otvoren problem, veliki broj različitih pristupa)
- Ovo predavanje se bavi prikazom nekoliko *feature reduction* tehnika za redukciju dimenzija:
  - PCA (Principal Components Analysis)
  - Autoenkoderi
  - t-SNE (t-Distributed Stochastic Neighbor Embedding)
- Svakoj tehnici ćemo pristupiti intuitivno a zatim istražiti matematičku podlogu

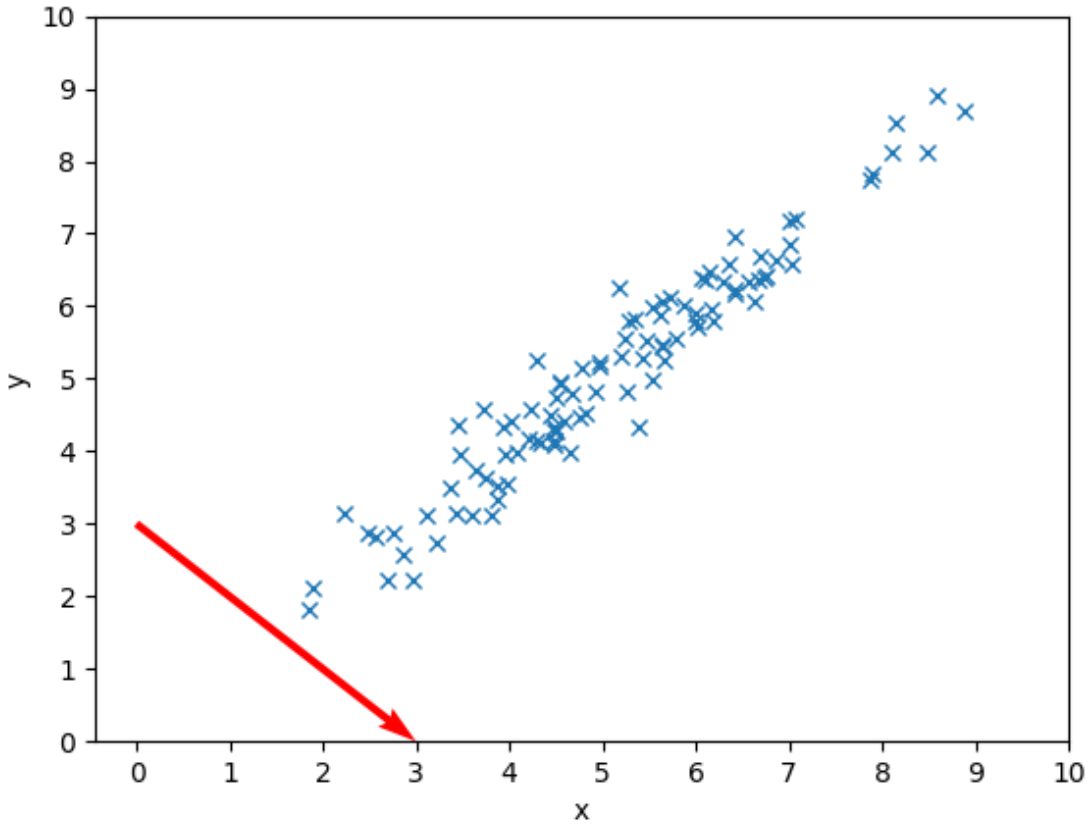
PCA: Primer

- $n = 2, d = 1$
- x - prvi feature, koliko je učenik dobar informatičar
- y - drugi feature, koliko učenik voli informatiku
- Kako bismo ovo predstavili u 1D?



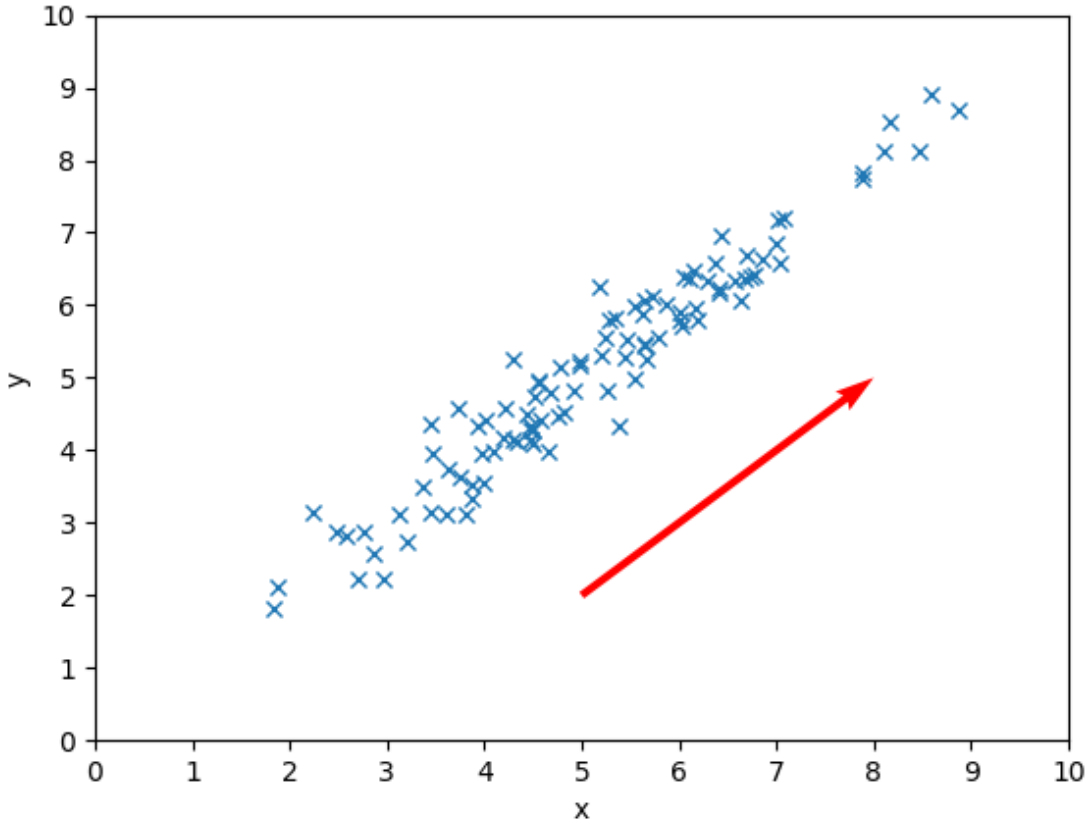
Primer

- Nelogičan izbor (različite tačke $\rightarrow$ ista tačka)



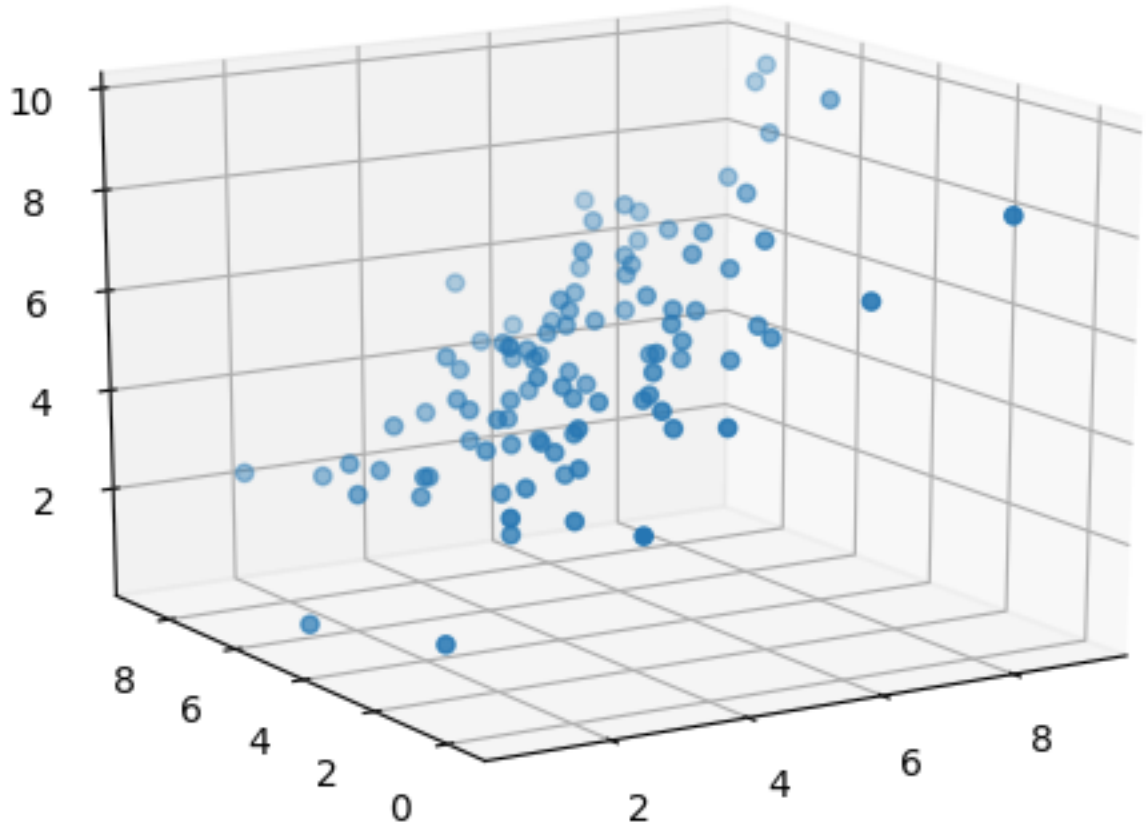
Primer

- Logičan izbor
- Sada umesto 2 feature-a imamo jedan
- Kako ga nazvati?



Primer

- $n = 3$
- $d = 1?$ $d = 2?$
- Ponovo je izbor *logičan*, ali kako ovo objasniti?



Ideja

- Korisni su nam pravci po kojima su podaci što više „razbacani” (kasnije ćemo ovo formalizovati)
- Ako je $d = 1$ biramo pravac koji daje najveću „razbacanost” (pc_1)
- Ako je $d = 2$ biramo pc_1 a zatim pravac ortogonalan njemu koji daje najveću „razbacanost” (pc_2)
- Na isti način uopštavamo, glavna ideja PCA je da sačuva što više „razbacanosti” redom po dimenzijama

Koraci

- Tri glavna koraka:
  1. *Normalizujemo* svaki feature ($x \rightarrow \frac{x-\mu}{\sigma}$) (!)
  2. Opisanom metodom dobijamo novu *bazu* iste dimenzije ($pc_1, pc_2, ..., pc_n$)
  3. Čuvamo prvih d dimenzija ($pc_1, pc_2, ..., pc_d$)
- Sada malo formalnije...

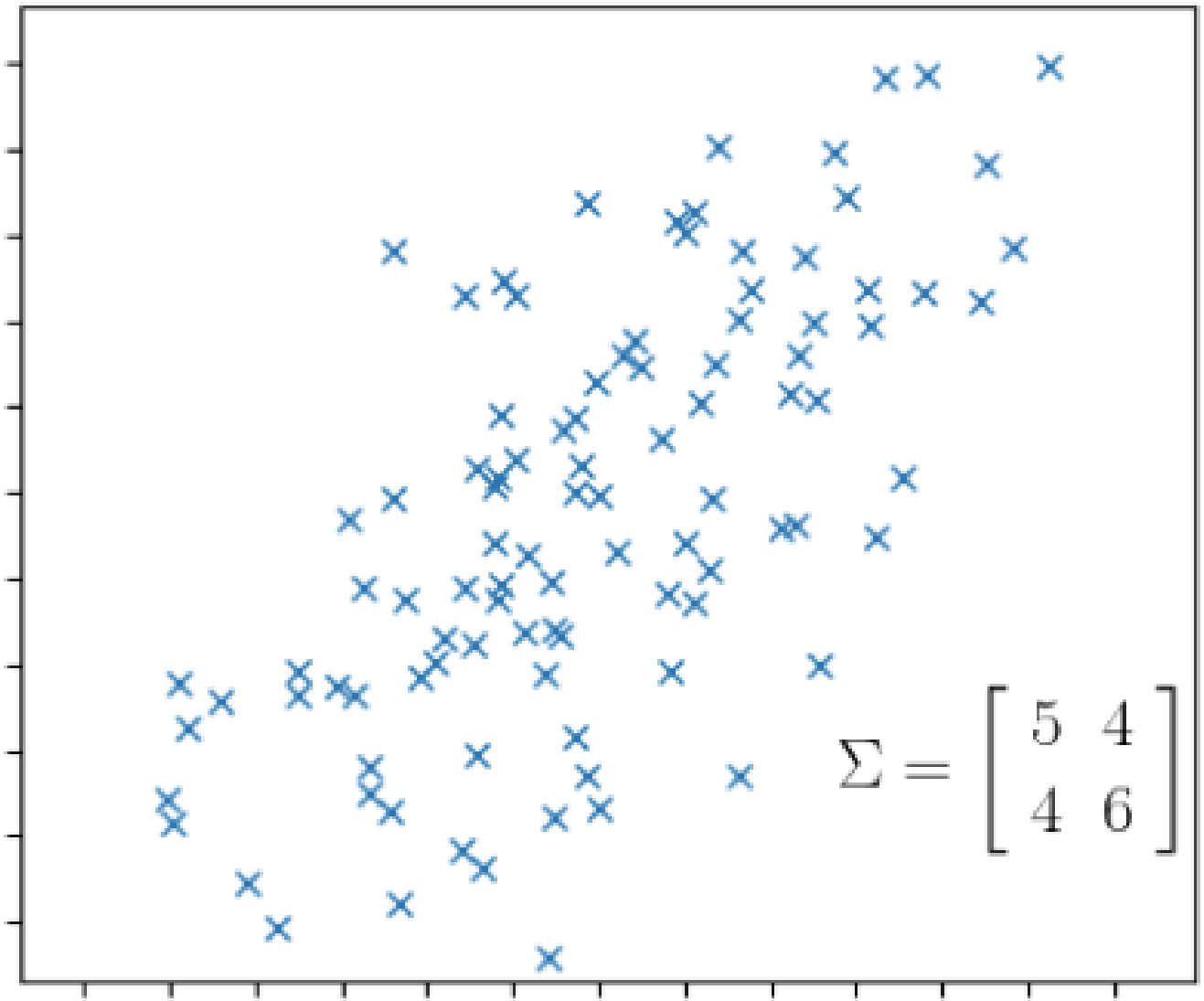
Malo verovatnoće

- Imamo **slučajnu promenljivu** X sa mogućim ishodima $x_1, x_2, x_3..$ i verovatnoćom za svaki ishod $\mathbb{P}(X = x_i) \in [0, 1]$ tako da je $\sum \mathbb{P}(X = x_i) = 1$
- **Matematičko očekivanje** (prosečni ishod): $\mathbb{E}(X) = \sum x_i \mathbb{P}(X = x_i)$
- **Varijansa** („razbacanost” tj. očekivano kvadratno rastojanje od matematičkog očekivanja): $\mathrm{Var}(X) = \mathbb{E}((X - \mathbb{E}(X))^2)$
- **Kovarijansa** (mera linearne povezanosti dve slučajne promenljive): $\mathrm{Cov}(X, Y) = \mathbb{E}((X - \mathbb{E}(X))(Y - \mathbb{E}(Y)))$ (kada veliko X prati veliko Y kovarijansa je pozitivna)

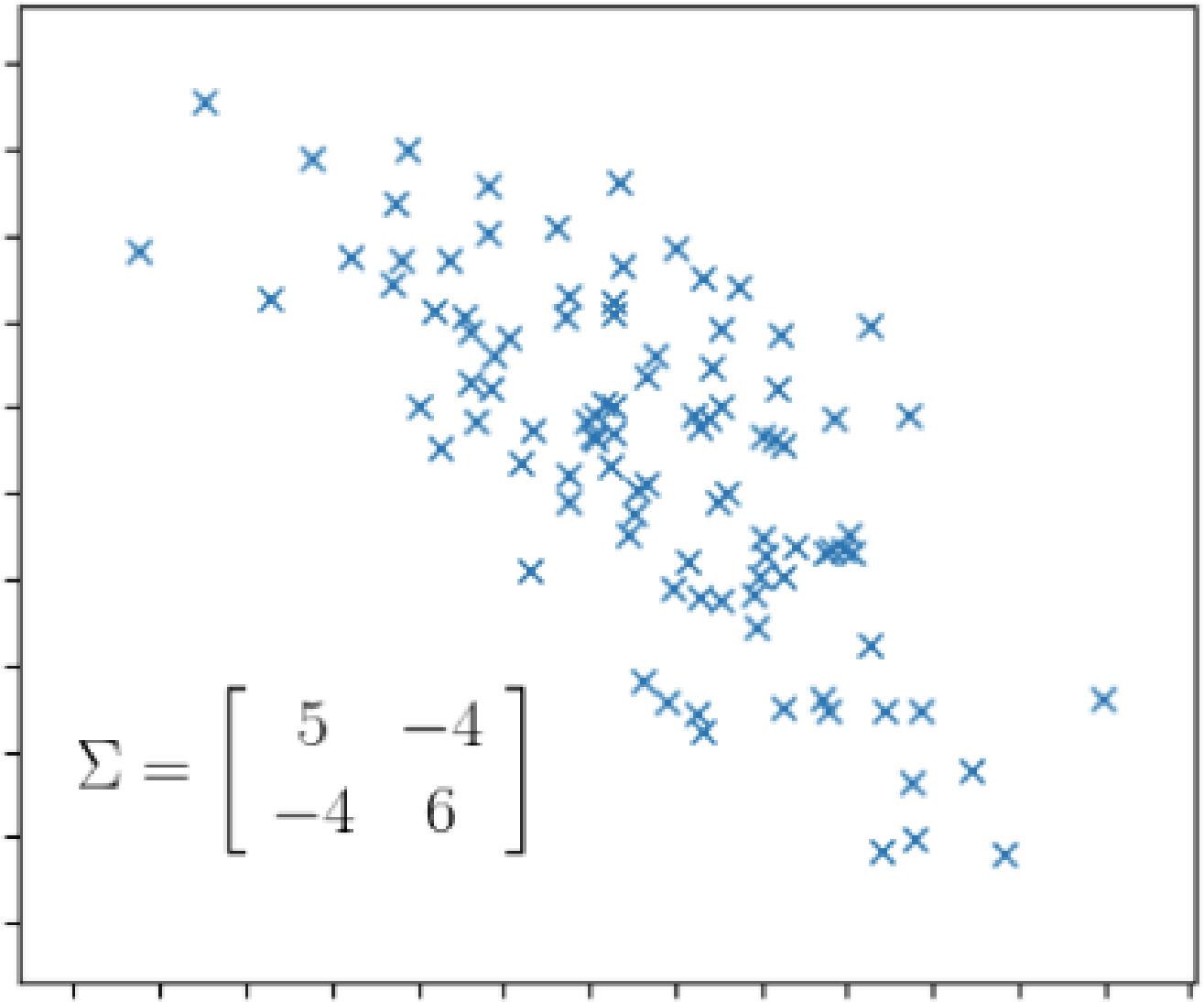
Malo verovatnoće

- **Matrica kovarijanse**: definiše se za slučajan vektor (vektor slučajnih promenljivih) $X = (X_1, ..., X_n)$ i označava se sa Σ
- U i-tom redu i j-toj koloni ove matrice nalazi se kovarijansa promenljivih X_i i X_j.
- Kako je $\mathrm{Cov}(X, Y) = \mathrm{Cov}(Y, X)$ matrica je simetrična u odnosu na glavnu dijagonalu
- Kao što smo varijansu mogli da shvatimo kao meru „razbacanosti” neke slučajne promenljive, tako matrica kovarijanse u potpunosti opisuje „razbacanost” slučajnog vektora

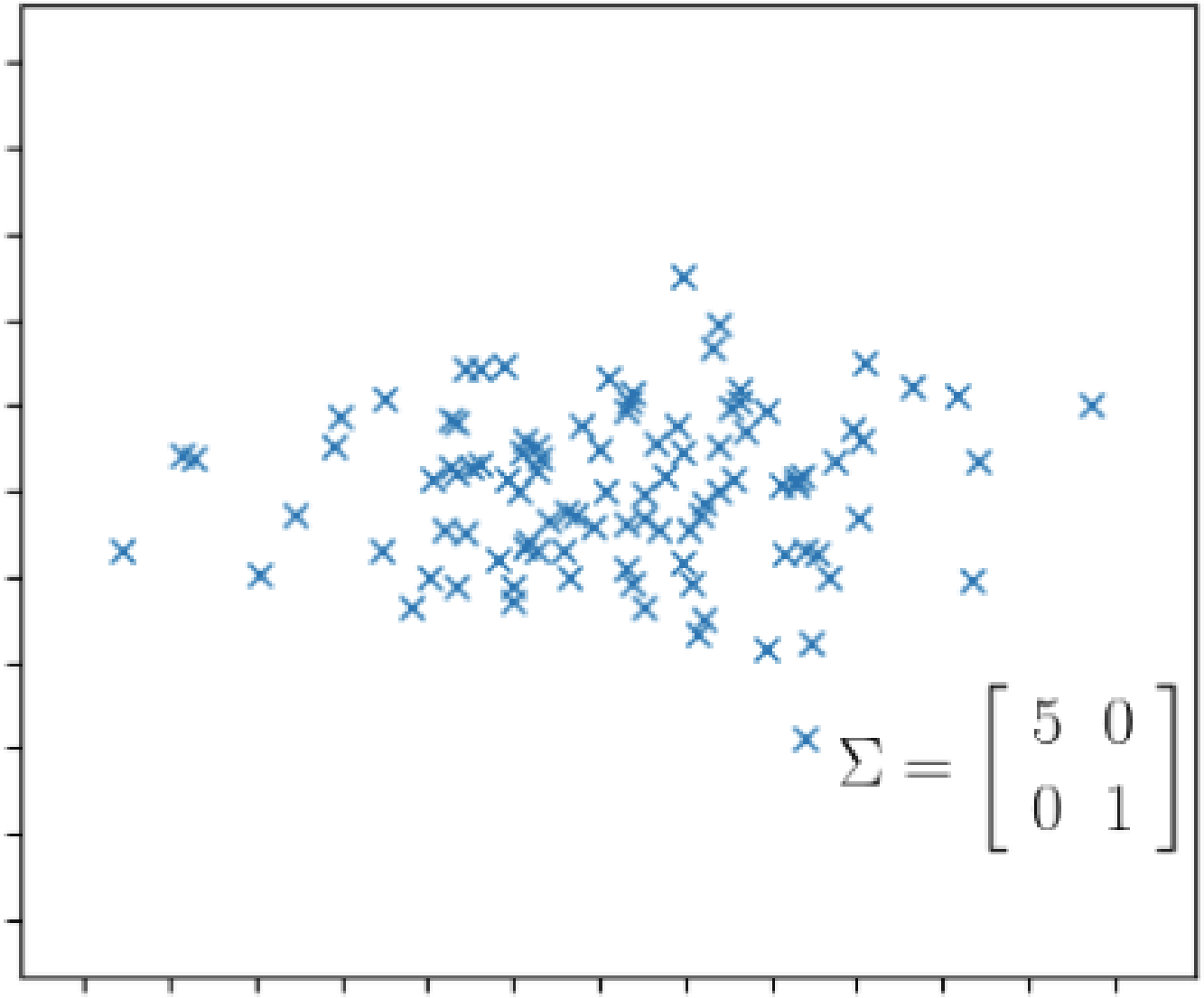
Malo verovatnoće



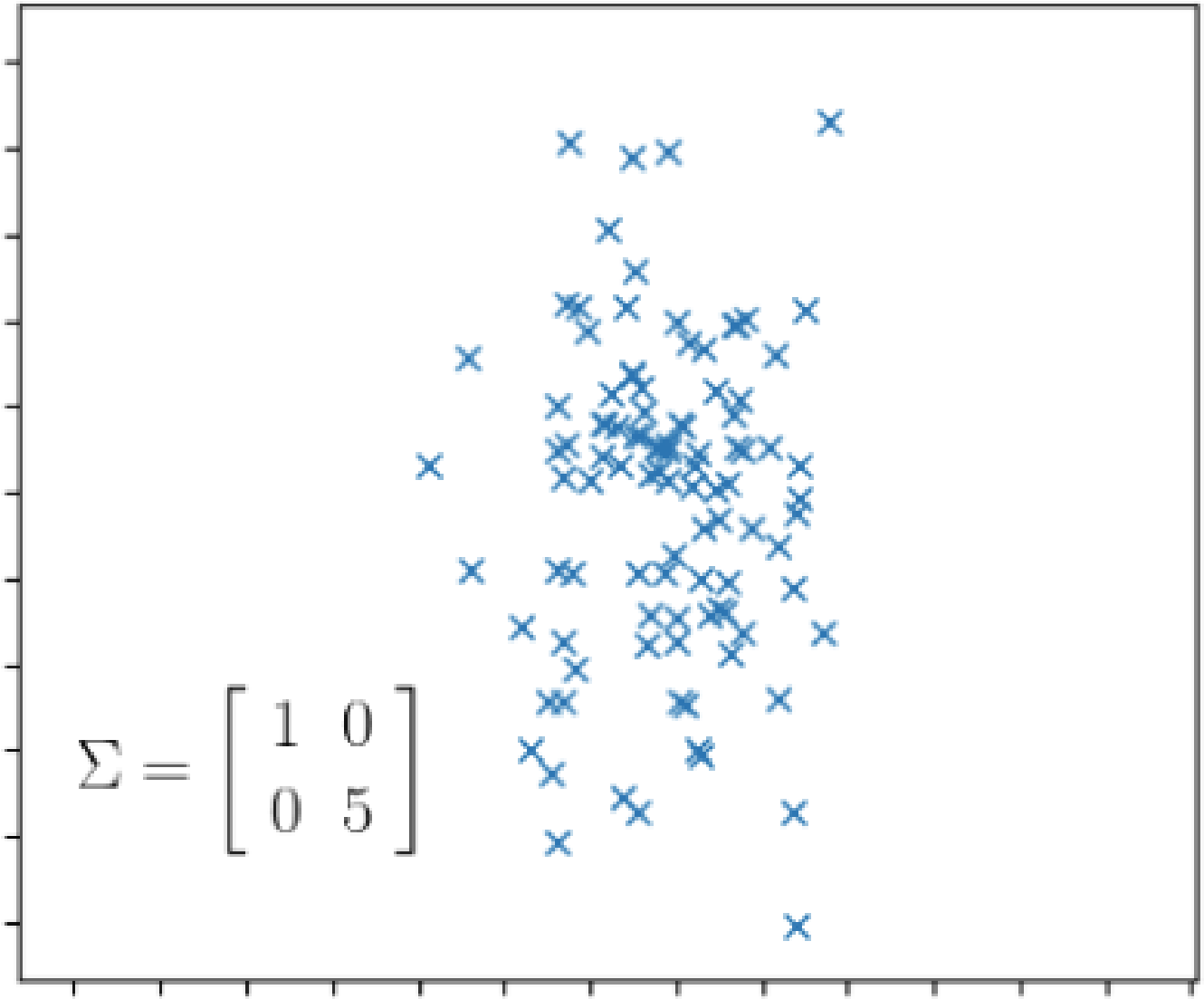
Malo verovatnoće



Malo verovatnoće



Malo verovatnoće



Matematička formulacija

- Tretiramo naših n dimenzija (n feature-a) kao slučajni vektor
- Matrica kovarijanse tog vektora ima ključnu ulogu u PCA!
- **Podsetnik**: glavna ideja PCA je da sačuva što više „razbacanosti” redom po dimenzijama
- `max` „razbacanost” $\leftrightarrow$ `max` varijansa po nekoj dimenziji
- `max` varijansa po nekoj dimenziji $\leftrightarrow$ `min` suma kvadrata dužina projekcija
  - Zašto?

Matematička formulacija

- Dakle, tražimo novu bazu $PC = [pc_1, pc_2, ..., pc_n]$ tako da je za matricu kovarijanse podataka u novoj bazi (Σ_{PC}) element $(1, 1)$ maksimalan, zatim $(2, 2)$, itd.
- Želimo da kada biramo prvih d dimenzija zadržimo što više ukupne varijanse

Matematička formulacija

- Dakle, tražimo novu bazu $PC = [pc_1, pc_2, ..., pc_n]$ tako da je za matricu kovarijanse podataka u novoj bazi (Σ_{PC}) element $(1, 1)$ maksimalan, zatim $(2, 2)$, itd.
- Želimo da kada biramo prvih d dimenzija zadržimo što više ukupne varijanse
- Ispostavlja se:
  - PC se sastoji od *svojstvenih vektora* Σ
  - Σ_{PC} je oblika $\begin{pmatrix} \lambda_1 & & & \\ & \lambda_2 & & \\ & & \ddots & \\ & & & \lambda_n \end{pmatrix}$, gde su λ_i *svojstvene vrednosti* Σ

Matematička formulacija

- Trvđenja sa prethodnog slajda nećemo sada dokazivati jer bi to zahtevalo uvođenje velikog broja koncepata i teorema iz linearne algebre kao i metode *Lagranžovih množilaca*, što nije fokus ovog predavanja
- Bitno je shvatiti da postoje jednostavni matematički postupci kojima možemo naći PC i transformisati Σ u Σ_{PC}
- Takođe, primetimo da Σ_{PC} ima nule van glavne dijagonale $\implies$ dimenzije su *nekorelisane* (kovarijansa im je nula)
  - PCA nalazi **optimalno** rešenje postavljenog problema

Količina objašnjene varijanse

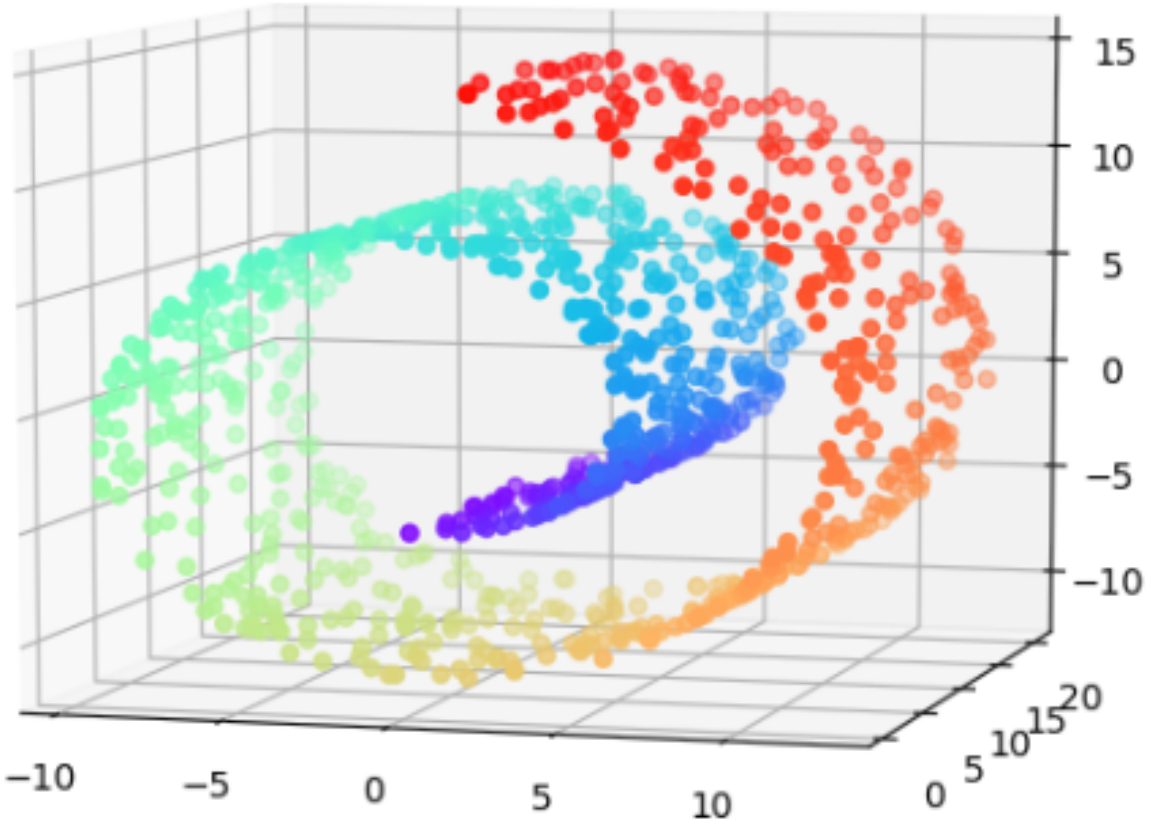
- Mera kvaliteta redukcije (može pomoći pri odabiru d)
- $\begin{pmatrix} 1.344 & 0.160 & 0.186 \\ -0.160 & 0.619 & -0.127 \\ 0.186 & -0.127 & 1.486 \end{pmatrix} \rightarrow \begin{pmatrix} 1.651 & 0.000 & 0.000 \\ 0.000 & 1.220 & 0.000 \\ 0.000 & 0.000 & 0.577 \end{pmatrix}$
- Ukupna varijansa je 3.448
- $d = 1$: Objašnjeno $1.651/3.448 = 47.9\%$ varijanse
- $d = 2$: Objašnjeno $2.871/3.448 = 83.3\%$ varijanse
- $d = 3$: Objašnjeno $3.448/3.448 = 100\%$ varijanse

Prednosti PCA

- Jednostavan i brz
- Interpretabilan (količina objašnjene varijanse)
- Prirodno podržava variranje d
- Popularan
  - Python: `sklearn.decomposition.PCA`
  - Matlab: `pca`
  - R: `prcomp`

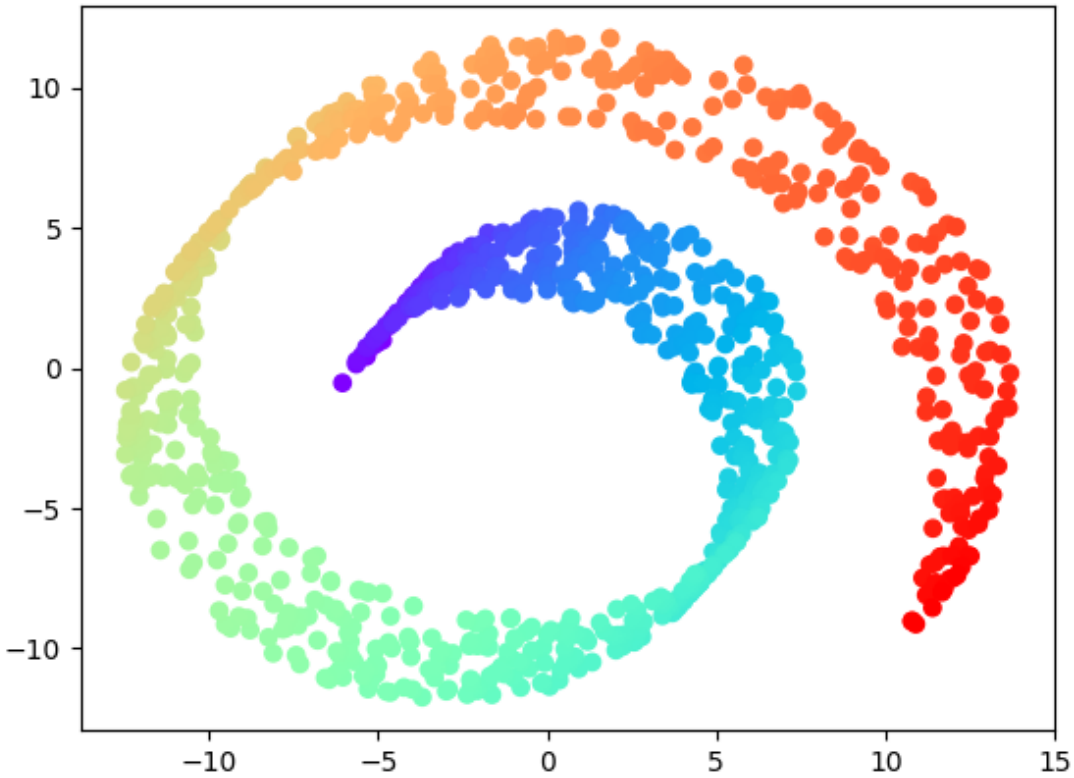
Mane

- *Swiss roll* (3D $\rightarrow$ 2D):



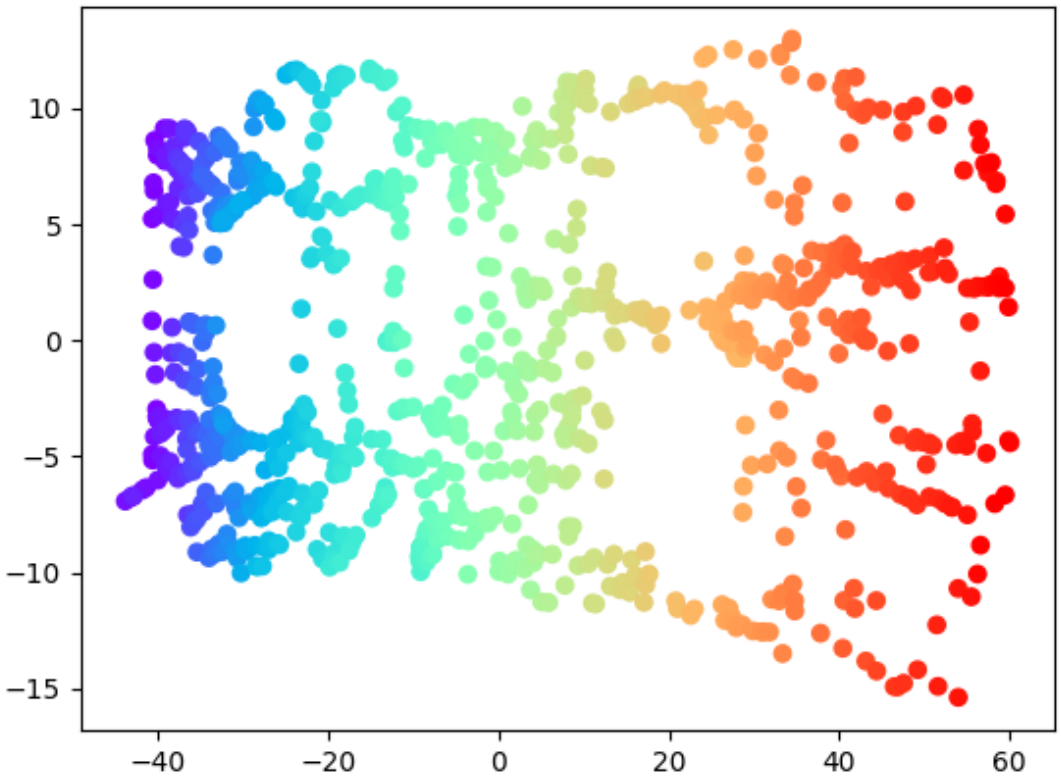
Mane

- PCA je linearna procedura, daje smislenu projekciju ali ne detektuje složenije pravilnosti



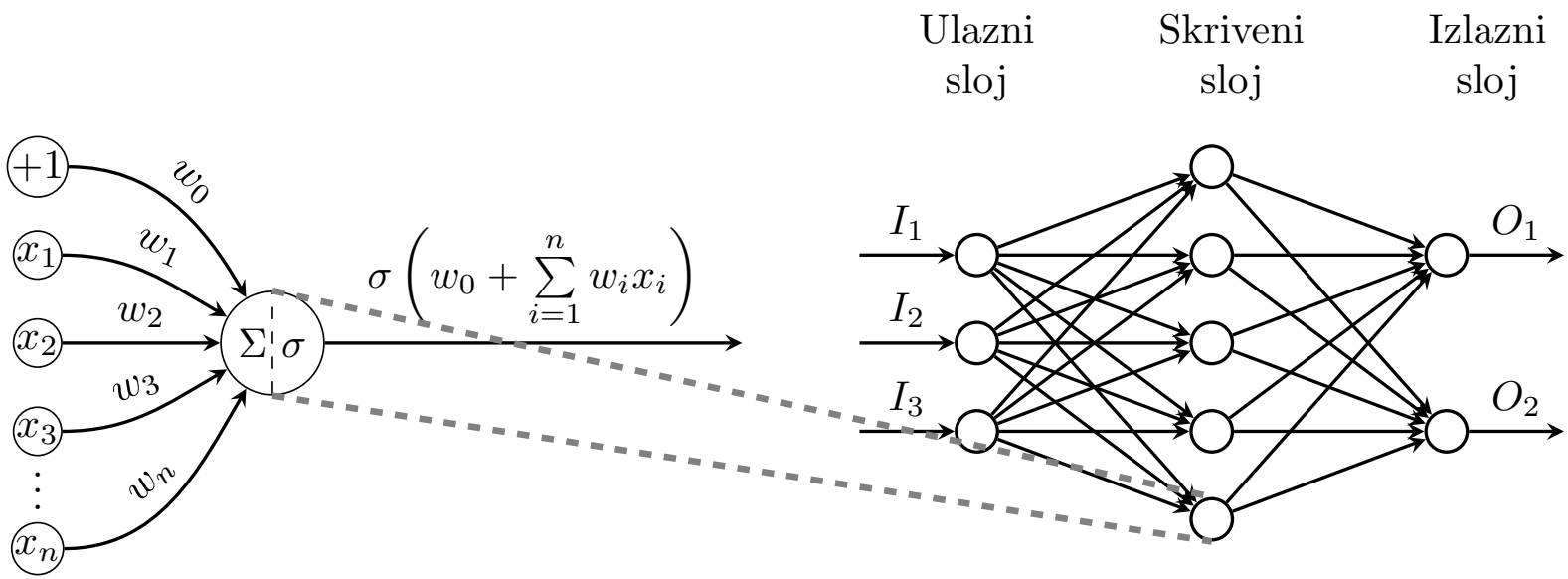
Mane

- Prikaz koristeći Isomap (još jedna složenija tehnika za redukciju dimenzija koju danas nećemo obrađivati)
- Pravilno uočava strukturu podataka, „razmotava” rolnu



Autoenkoderi: NN podsetnik [1]

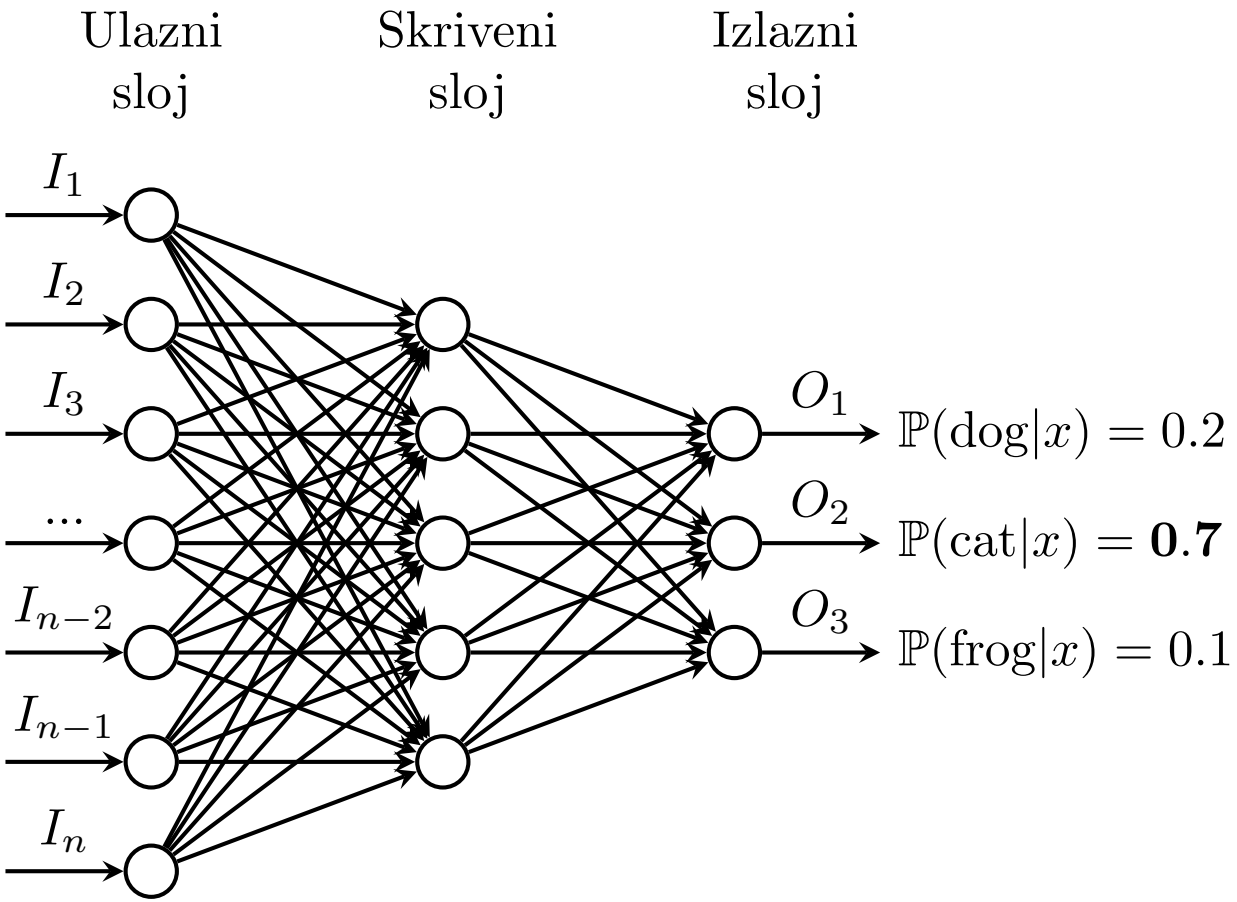
- Perceptron, aktivaciona funkcija, MLP, funkcija troška, gradijentni spust
- Učimo funkciju $x \to y$



[1] Ilustracije: `https://github.com/PetarV-/TikZ`

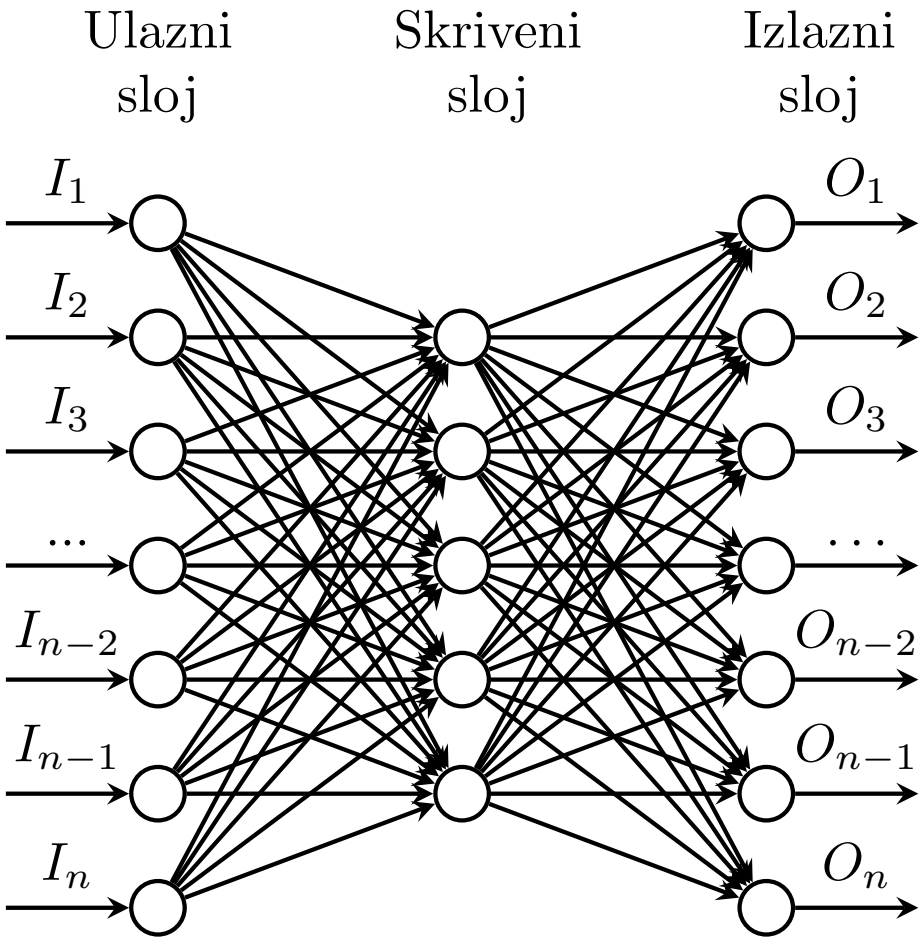
Standardna klasifikacija

- Npr. $x \in \mathbb{R}^n$, $y \in \mathbb{R}^3$



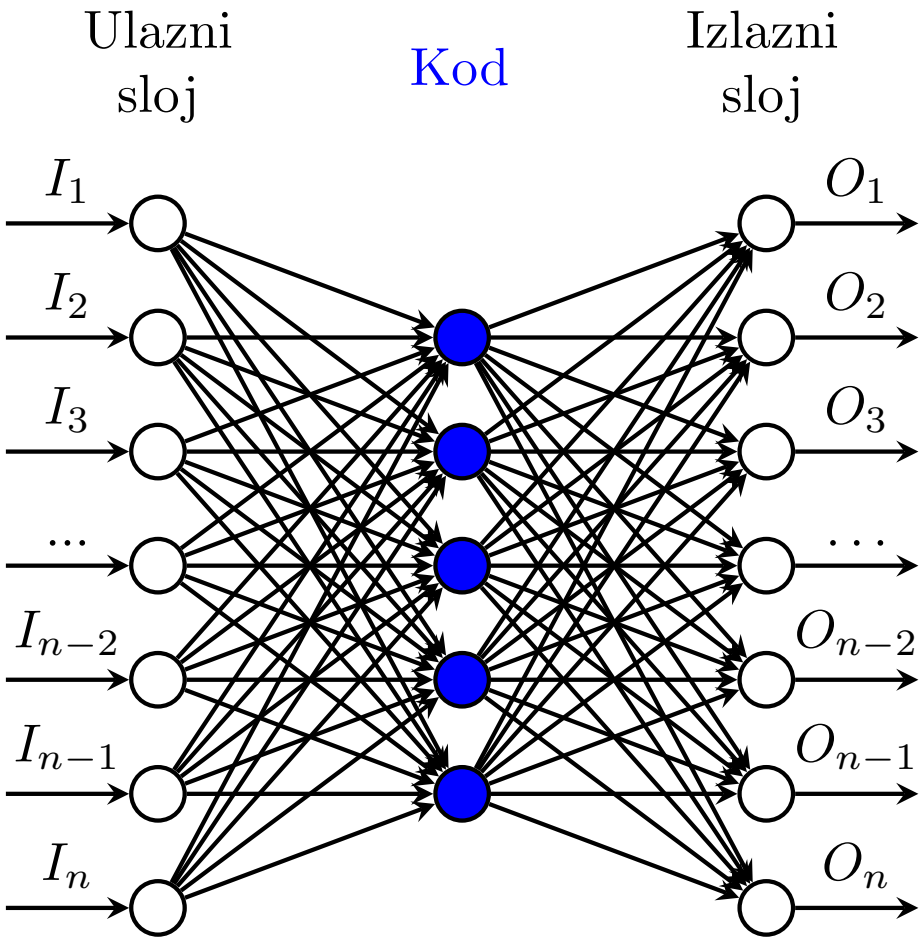
Autoenkoder

- $x \in \mathbb{R}^n$, $y \in \mathbb{R}^n$, $x \equiv y$
- Čemu ovo služi?



Autoenkoder

- $x \in \mathbb{R}^n, y \in \mathbb{R}^n, x \equiv y$



Autoenkoder

- **Autoenkoder** je neuralna mreža koja uči funkciju identiteta ($x \to x$)
- Značaj ovoga je u srednjem sloju (*bottleneck*), vrednosti na izlazu iz njega (z) koristimo kao **kod**
- Redukcija dimenzija: $x \in \mathbb{R}^n \to z \in \mathbb{R}^d$
- Na prethodnom slajdu je ilustrovan najjednostavniji autoenkoder sa jednim skrivenim slojem i pretpostavili smo jednostavnu funkciju troška (npr. euklidska distanca)
- Naravno, postoje mnogo složenije varijante (više slojeva, složeniji trošak, *DAE*, *CAE*, *VAE*...)

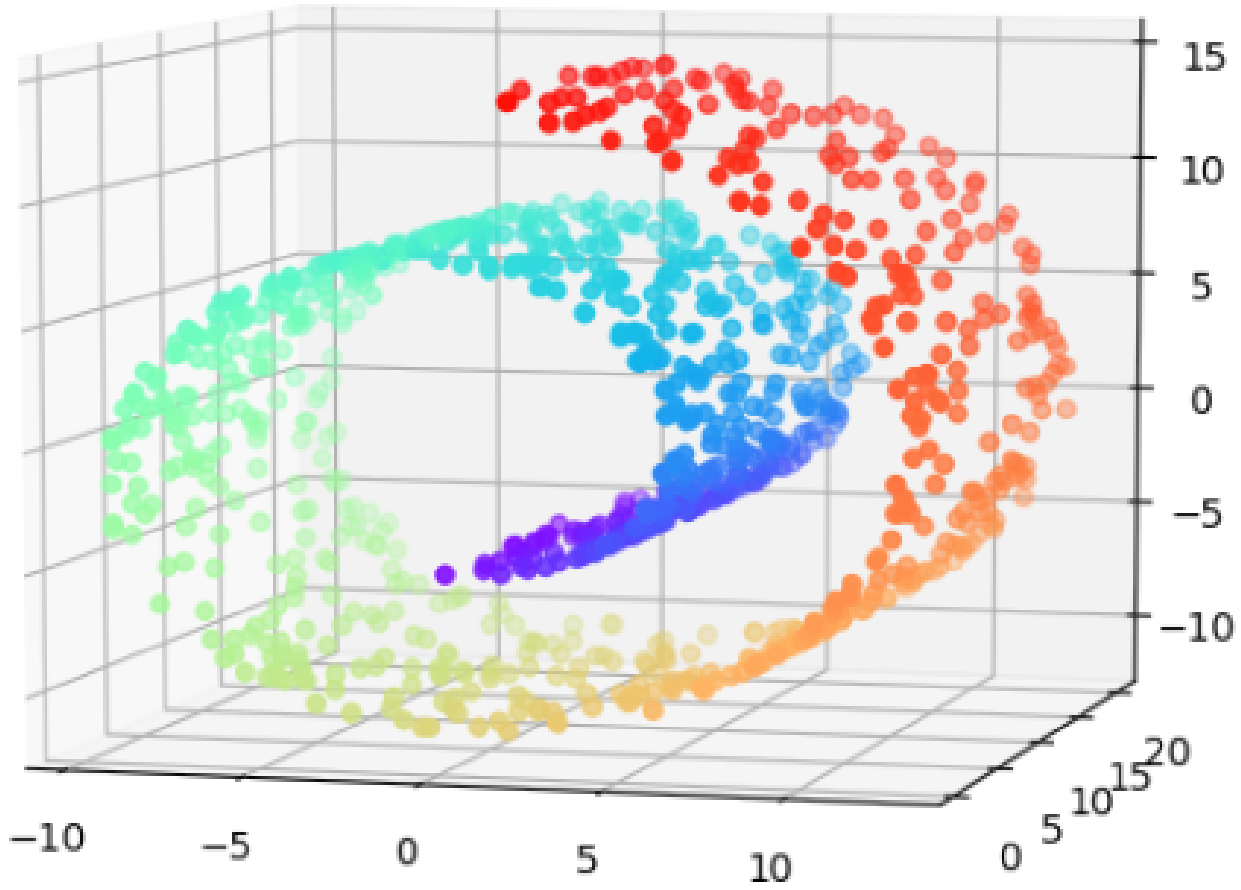
Autoenkoder vs PCA

- Ako su sve aktivacione funkcije linearne autoenkoder konvergira rešenju vrlo sličnom onom koje bi dao PCA (nezavisno od broja slojeva)
  - Zašto?
- Glavna prednost autoenkodera je u nelinearnostima: kompleksnije transformacije $\rightarrow$ fleksibilnija redukcija

Autoenkoder vs PCA

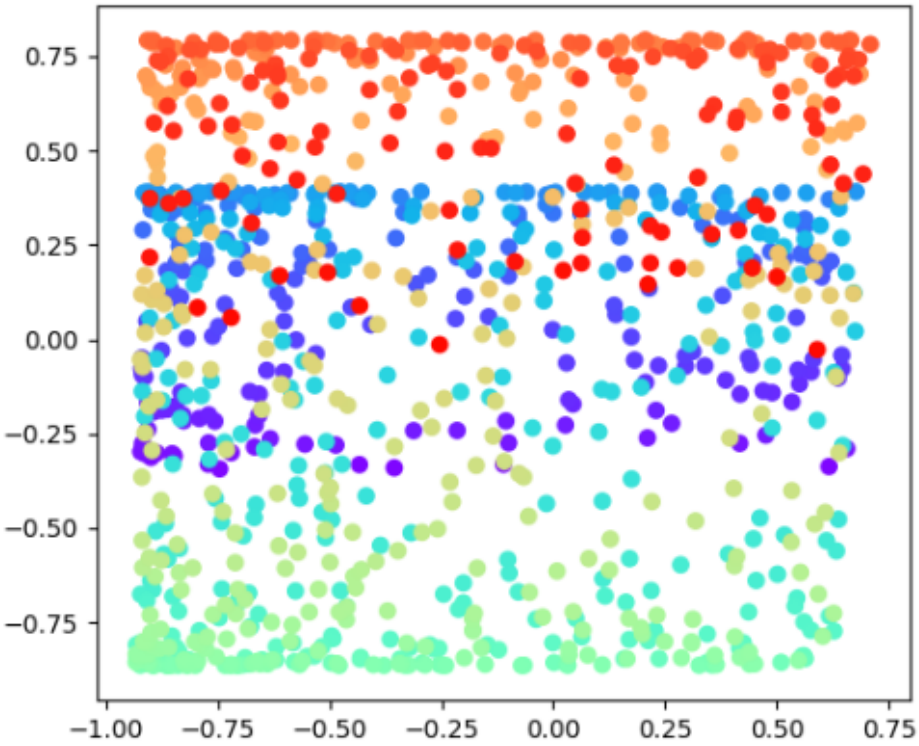
- Stohastički gradijentni spust omogućava da treniramo autoenkoder „primer po primer”, što:
  - Ukida potrebu za učitavanjem celog skupa podataka u memoriju u slučaju ogromnih skupova
  - Omogućava trening u „online” scenarijima kada podaci periodično pristižu
  - Omogućava jednostavnu transformaciju novih tačaka koje nisu bile u originalnom skupu
  - Sve tri stavke su problem za PCA!
- Još jedna prednost: kako je autoenkoder neuralna mreža lako je proširivanje, modifikovanje, nadovezivanje...

Swiss roll



Swiss roll - autoenkoder

- Izlaz najprostijeg nelinearnog autoenkodera
- „Razmotavanje” nije pravilno (!)
- Usložnjavanje modela može pomoći, ali mana je u samoj osnovi pristupa



t-SNE: Ideja

- Oba dosadašnja pristupa su posmatrala globalnu strukturu i pokušavala da je što vernije predstave u manje dimenzija
- Ovo znači da parovi udaljenih tačaka značajno utiču na finalno rešenje (naročito u kontekstu maksimizacije varijanse)
- Kao što *swiss roll* i slični primeri pokazuju, to nije uvek pouzdano
- Ključna promena koju *t-SNE* uvodi: fokus na lokalnu strukturu tj. parove bliskih tačaka

Ideja

- Iterativan algoritam mašinskog učenja koji postepeno adaptira transformaciju $T : \mathbb{R}^n \to \mathbb{R}^d$ tako da što bolje zadrži odnos bliskih tačaka
- Algoritam se odvija bez eksplicitne formule za T:
  - Neka je X naš skup n-dimenzionih podataka
  - Krećemo od nasumične d-dimenzione slike $Y = T(X)$
  - U svakom koraku modifikujemo Y tako da se minimizuje „nesaglasnost” između „sličnosti” tačaka u X i „sličnosti” tačaka u Y (pri čemu se veći akcenat stavlja na bliske tačke)

Ideja

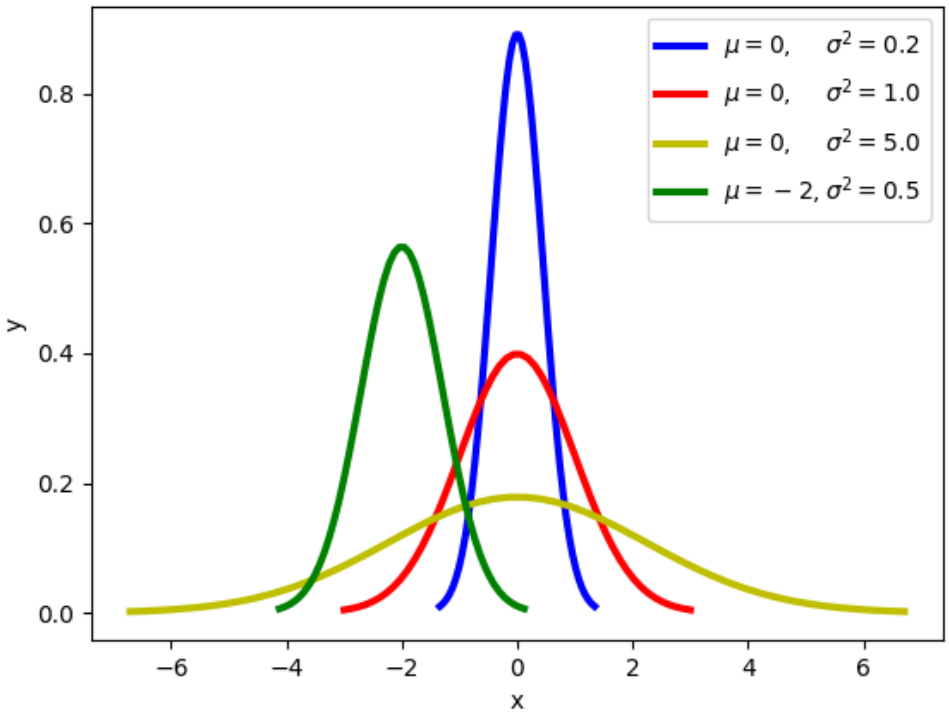
- Ovi koraci su suština algoritma, ali se prirodno postavljaju tri pitanja:
  1. Kako izraziti sličnost tačaka u X?
  2. Kako izraziti sličnost tačaka u Y?
  3. Kako izraziti nesaglasnost tih sličnosti?

Normalna raspodela

- **Gausova (normalna) raspodela**: Jedna od najvažnijih raspodela čija funkcija gustine ima oblik zvona
- ko slučajna promenljiva X ima normalnu raspodelu pišemo $X \sim \mathcal{N}(\mu, \sigma^2)$, pri čemu parametar μ označava matematičko očekivanje X, a σ^2 varijansu

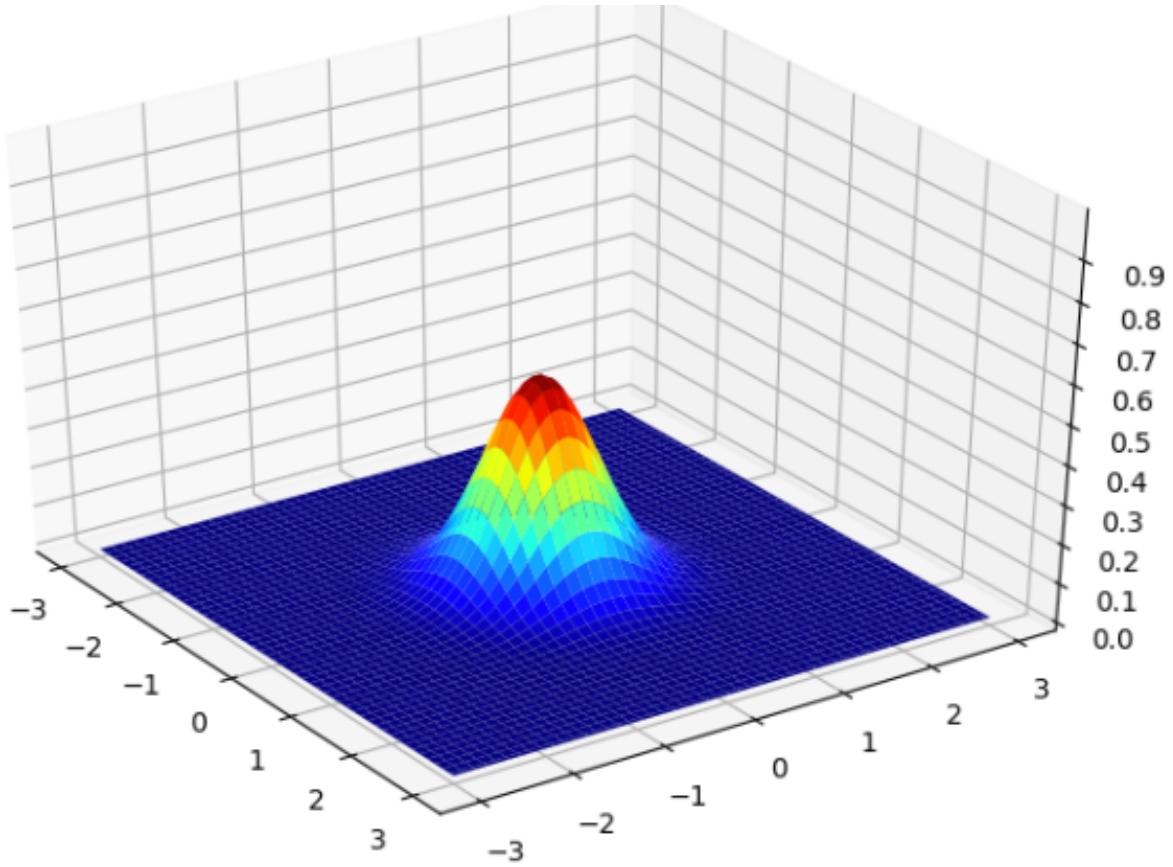
Normalna raspodela

- Najveća je verovatnoća da X uzme vrednost blisku μ tj. u „telu” raspodele, a ta verovatnoća se smanjuje ka „repovima” raspodele
- Prikaz normalne raspodele za različite parametre:



Normalna raspodela

- Normalna raspodela se uopštava na više dimenzija
- Prikaz jedne 2D raspodele za slučajni vektor (X, Y):



1. Kako izraziti sličnost tačaka u X?

- $$p_{j|i} = \frac{\exp(-\|\mathbf{x}_i - \mathbf{x}_j\|^2 / 2\sigma_i^2)}{\sum_{j' \neq i} \exp(-\|\mathbf{x}_i - \mathbf{x}_{j'}\|^2 / 2\sigma_i^2)}$$
- Verovatnoća da x_i odabere x_j za suseda ako se susedi biraju proporcionalno vrednosti funkcije gustine normalne raspodele centrirane u x_i sa varijansom σ_i^2
- σ_i se izvode iz globalnog hiperparametra σ (**perplexity**) tako da neutrališu „gustinu” prostora oko x_i
- „Gusti” delovi prostora $\rightarrow$ manje σ_i (*zoom* na taj deo prostora)
  - Zbog ovoga gornja formula nije simetrična

1. Kako izraziti sličnost tačaka u X?

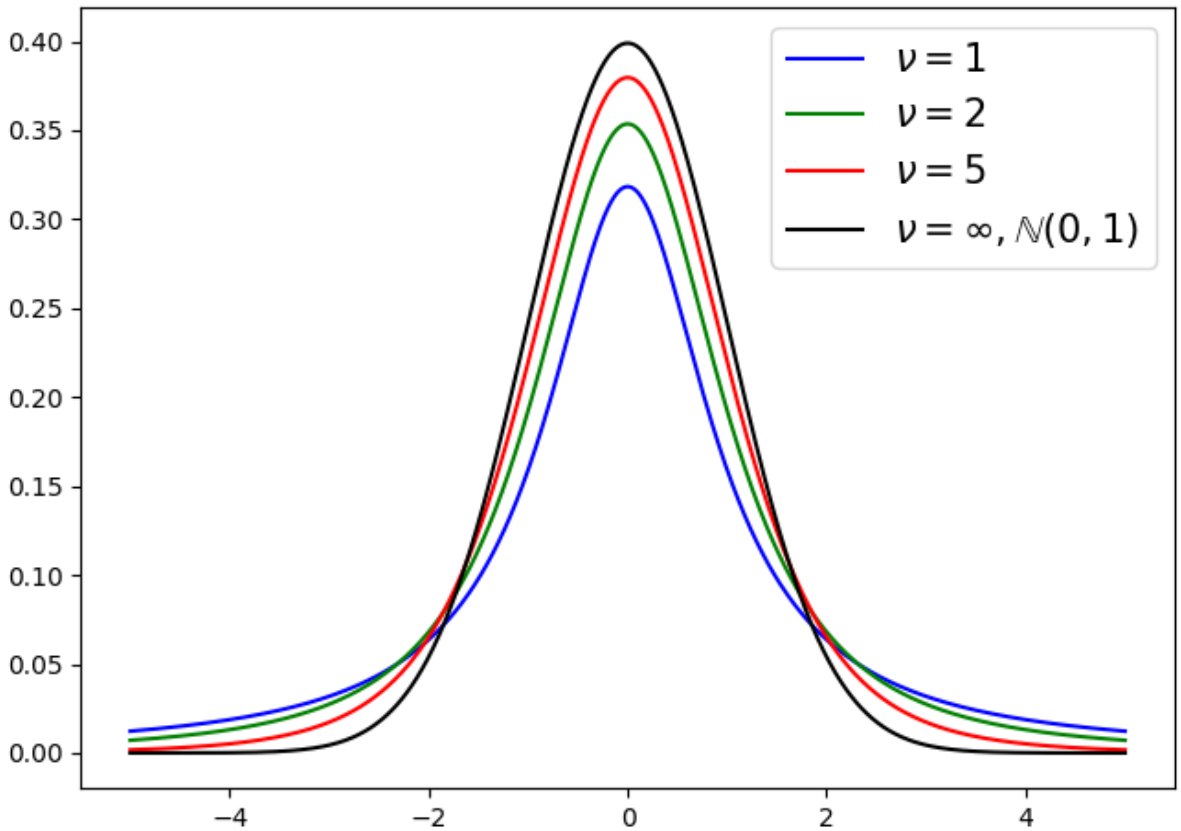
- Ukupna sličnost tačaka x_i i x_j (ako je N ukupan broj tačaka):
- $p_{ij} = \frac{p_{j|i} + p_{i|j}}{2N}$
- x_i i x_j veoma daleko $\rightarrow p_{ij} \sim 0$
- x_i i x_j veoma blizu $\rightarrow p_{ij} \sim 1$
- Važi $\sum p_{ij} = 1 \implies$ Tretiramo P: $(i, j) \rightarrow p_{ij}$ kao raspodelu

Studentova t-raspodela

- Uveo je početkom 20. veka hemičar Vilijam Goset koji je tada radio za pivaru *Guinness*
- Nastala (zajedno sa *t-testom*) kao nova, ekonomična metoda za praćenje kvaliteta piva
- *Guinness* nije dozvoljavao da se otkrića publikuju pa je Goset svoj rad publikovao pod pseudonimom *Student*

Studentova t-raspodela

- Parametar ν - stepen slobode
- Nalik normalnoj raspodeli, sa većim repovima



2. Kako izraziti sličnost tačaka u Y?

- $$q_{ij} = \frac{(1+\|\mathbf{y}_i - \mathbf{y}_j\|^2)^{-1}}{\sum_{j' \neq i} (1+\|\mathbf{y}_i - \mathbf{y}_{j'}\|^2)^{-1}}$$
- Nalik načinu na koji je definisana sličnost u X
- Umesto normalne: Studentova t-raspodela sa $\nu = 1$
  - Da bi bliske tačke ostale bliske neophodno je da neke udaljene tačke u X budu **još više** udaljene u Y, t-raspodela to toleriše
  - y_i i y_j veoma daleko $\rightarrow q_{ij} \sim 0$ ali $q_{ij} > p_{ij}$ (t-raspodela je blaža, veći repovi)
- Važi $\sum q_{ij} = 1 \implies$ Tretiramo Q: $(i, j) \rightarrow q_{ij}$ kao raspodelu

KL divergencija (relativna entropija)

- $KL(P||Q) = \sum_i P(i) \log \frac{P(i)}{Q(i)}$
- Mera nesaglasnosti dve raspodele
- Količina informacija koja je izgubljena kada koristimo Q da aproksimiramo P

3. Kako izraziti nesaglasnost tih sličnosti?

- $$KL(P||Q) = \sum_{i \neq j} p_{ij} \log \frac{p_{ij}}{q_{ij}}$$
- Smislen izbor na osnovu tumačenja KL divergencije sa prethodnog slajda (koristimo Q da aproksimiramo P)
- Kada je KL veće: „daleko $\rightarrow$ blizu” ili „blizu $\rightarrow$ daleko”?

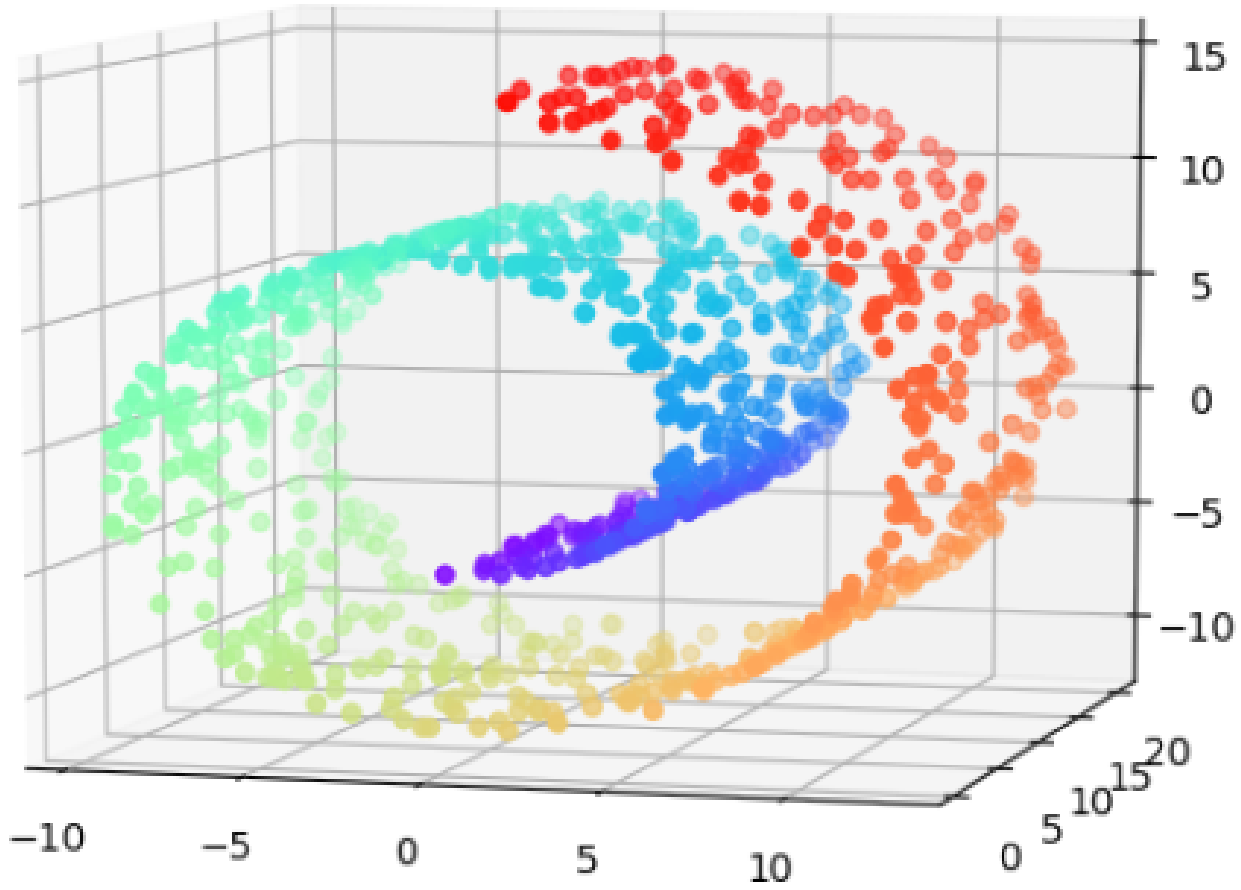
3. Kako izraziti nesaglasnost tih sličnosti?

- $$KL(P||Q) = \sum_{i \neq j} p_{ij} \log \frac{p_{ij}}{q_{ij}}$$
- Smislen izbor na osnovu tumačenja KL divergencije sa prethodnog slajda (koristimo Q da aproksimiramo P)
- Kada je KL veće: „daleko $\rightarrow$ blizu” ili „blizu $\rightarrow$ daleko”?
- Daleko u X (p_{ij} malo), blizu u Y (q_{ij} veliko) $\implies$ malo KL
- Blizu u X (p_{ij} veliko), daleko u Y (q_{ij} malo) $\implies$ veliko KL
- Ovo je funkcija troška koju želimo da minimizujemo, diferencijabilna je pa koristimo **gradijentni spust**
  - Gradijent možemo da tumačimo i kao gravitacionu silu (*problem n tela*)

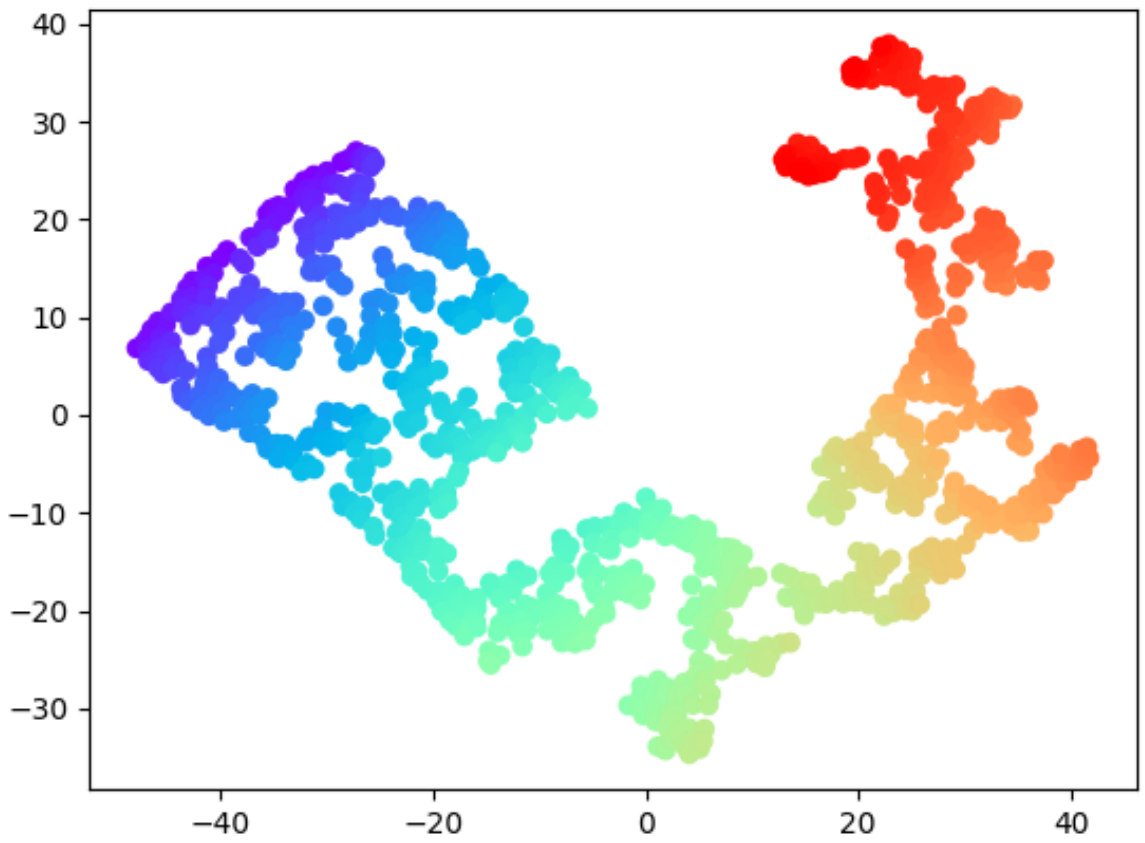
Nazad na ideju

- Iterativan algoritam mašinskog učenja koji postepeno adaptira transformaciju $T : \mathbb{R}^n \to \mathbb{R}^d$ tako da što bolje zadrži odnos bliskih tačaka
- Algoritam se odvija bez eksplicitne formule za T:
  - Neka je X naš skup n-dimenzionih podataka
  - Krećemo od nasumične d-dimenzione slike $Y = T(X)$
  - U svakom koraku modifikujemo Y gradijentnim spustom tako da se minimizuje KL divergencija između „$\mathcal{N}$-sličnosti” tačaka u X i „t-sličnosti” tačaka u Y

Swiss roll



Swiss roll - t-SNE

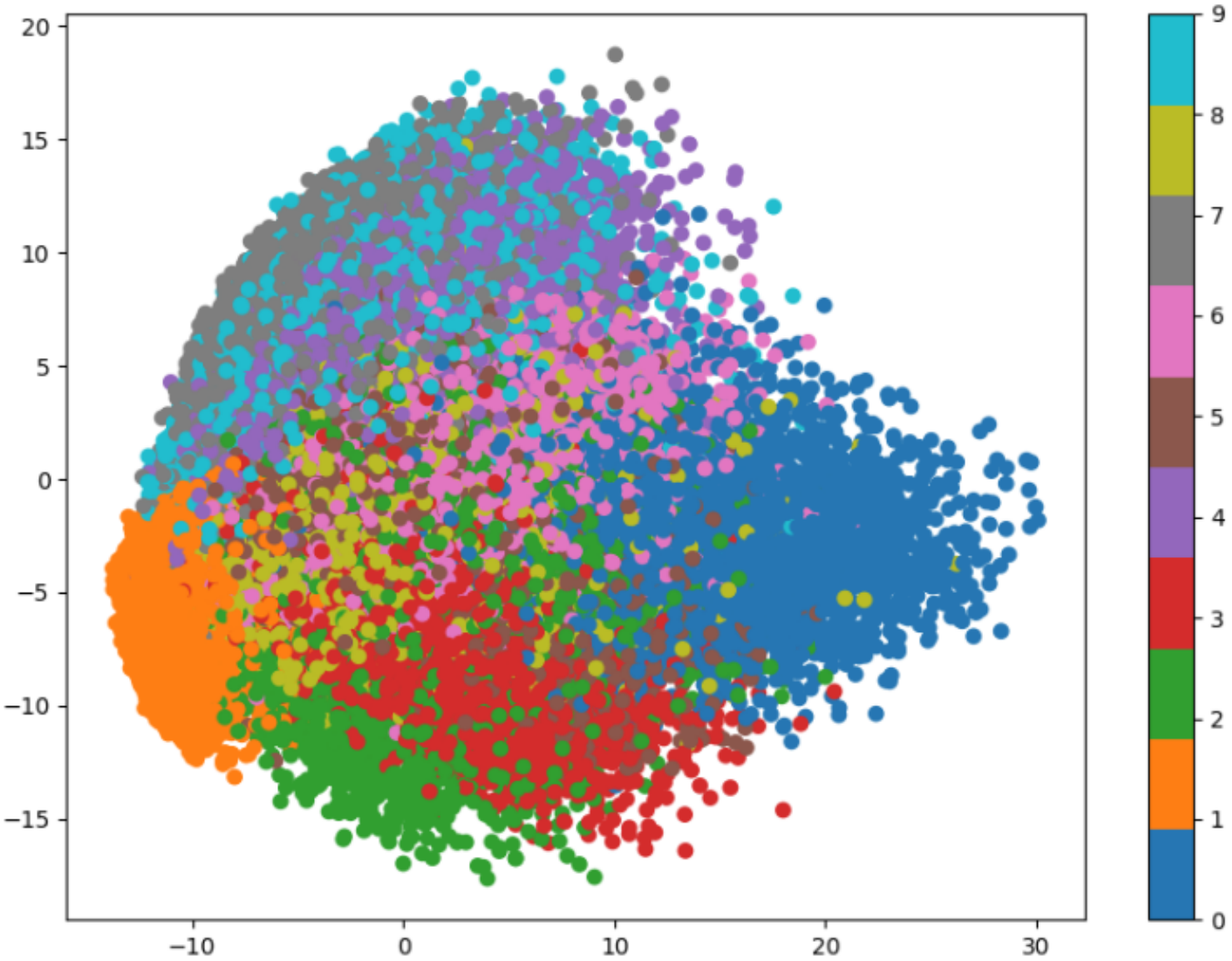


- Autor: *„Frankly. . . who cares about Swiss rolls when you can embed complex real-world data nicely?”*

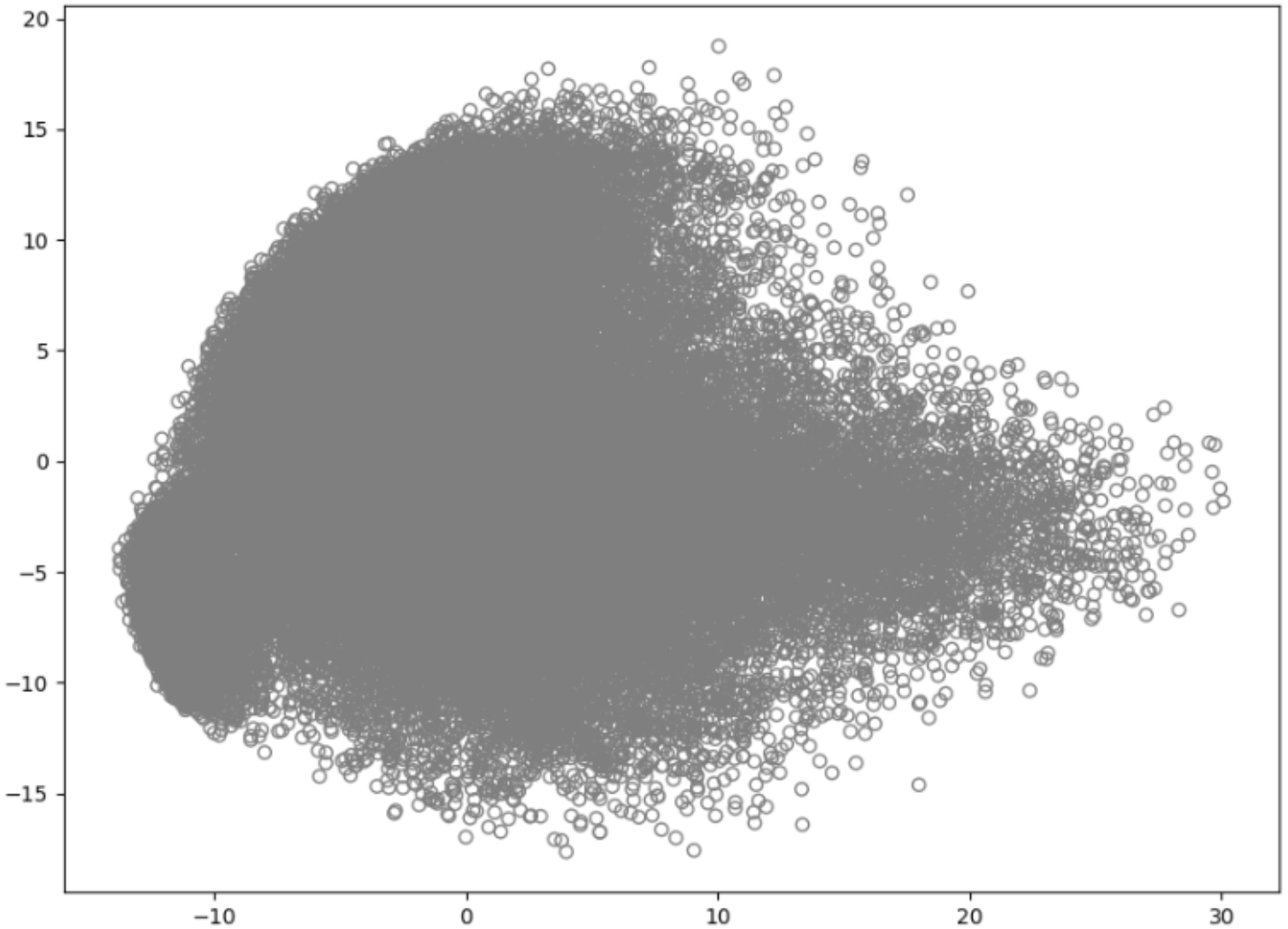
MNIST

- Baza 70000 ručno pisanih cifara u obliku 28×28 *grayscale* slika
- Svaka slika je vektor u $\mathbb{R}^{784}$, želimo da je predstavimo u $\mathbb{R}^2$

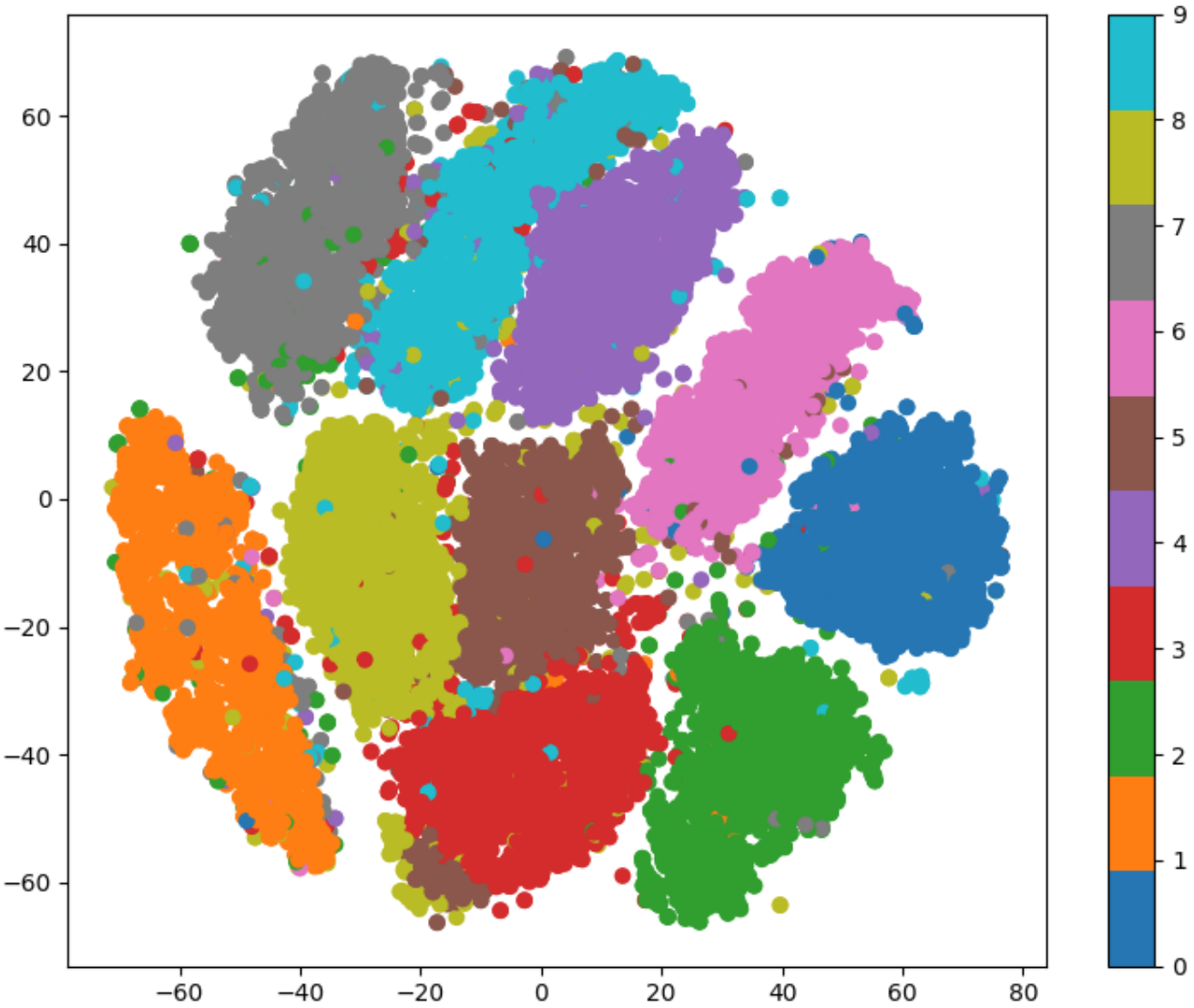
MNIST - PCA



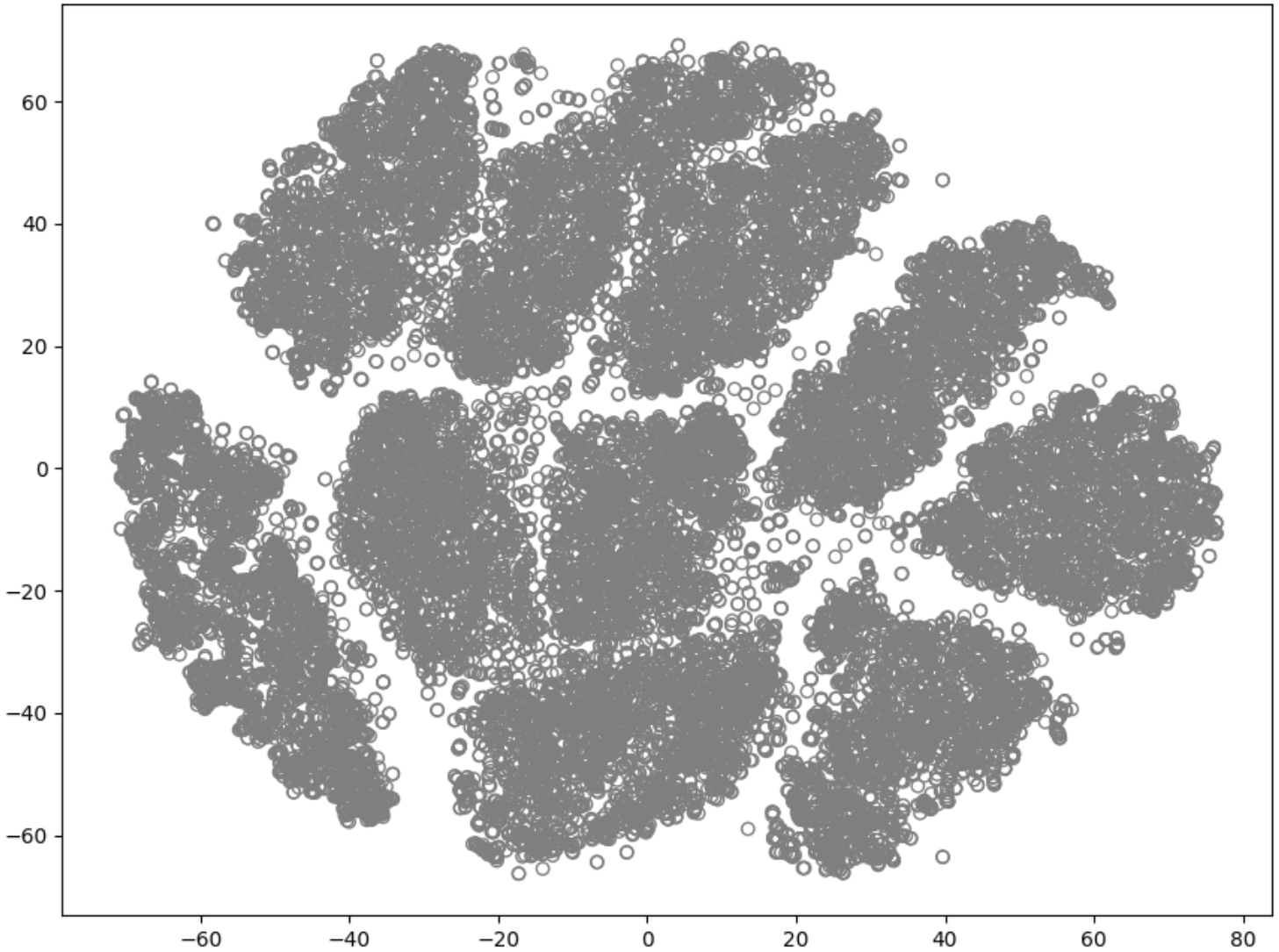
MNIST - PCA



MNIST - t-SNE



MNIST - t-SNE



Korišćenje

- Uglavnom istraživanje i vizualizacija podataka (retko se koristi za $d > 3$)
- Često se koristi za prikaz i tumačenje skrivenih slojeva neuralnih mreža (npr. enkodera)
- Može da bude jedan od koraka pre nekog od algoritama za klasterovanje
- Python: `sklearn.manifold.TSNE`

Mane

- Nedeterminstički (lokalni minimumi)
- Spor (rešenje: PCA pretprocesiranje koje smanji n)
- Ne može da transformiše novu tačku jer nikada eksplicitno nemamo transformaciju T
- Neinterpretabilan, ima nekoliko nezgodnih hiperparametra
  - Naročito σ (perplexity) - efekat na rezultat nije očigledan
  - Za kraj, istražićemo ovu problematiku i prikazati par uočenih pravilnosti koje objašnjavaju „magiju” iza t-SNE i pomažu da razvijemo određenu intuiciju [2]

[2] https://distill.pub/2016/misread-tsne/

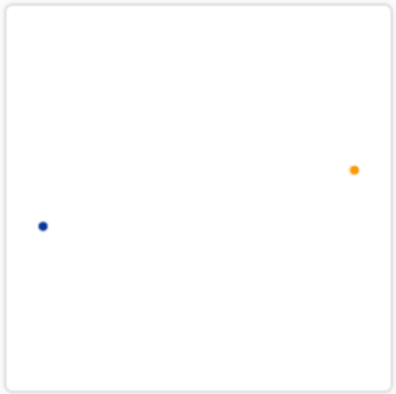
Broj iteracija

- Do stabilnosti, „uštinuti” delovi su loš znak

Original

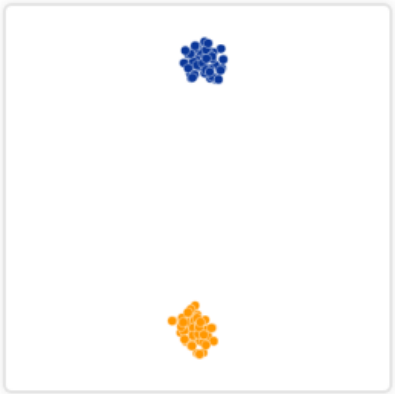
Perplexity: 30
Step: 10

Perplexity: 30
Step: 20



Perplexity: 30
Step: 60

Perplexity: 30
Step: 120



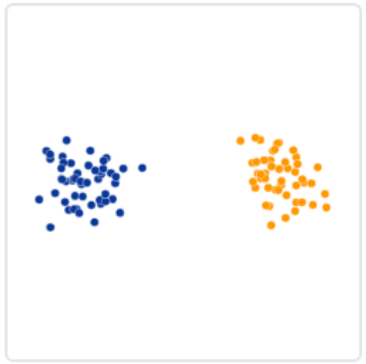
Perplexity: 30
Step: 1,000

Perplexity - podsetnik

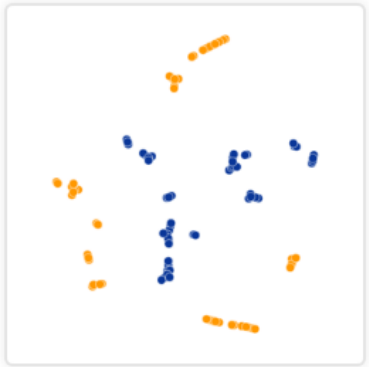
- U izrazu za sličnost tačaka u originalnom prostoru imali smo parametre σ_i
- Oni se izvode iz globalnog hiperparametra σ (**perplexity**) tako da neutrališu „gustinu” prostora oko x_i
- „Gusti” delovi prostora će imati manje σ_i (*zoom* na taj deo prostora)
- Ovo znači da na svaku tačku utiče sličan broj suseda, koji sledi iz σ (veće $\sigma \implies$ više suseda)

Perplexity - uticaj

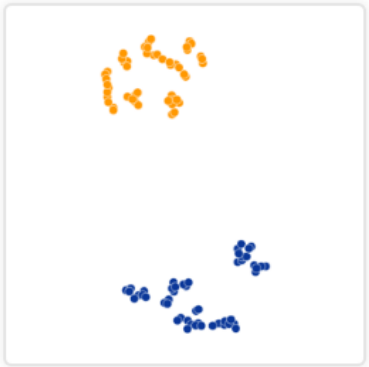
- Savet autora: $\sigma \in [5, 50]$
- σ premalo $\implies$ mali klasteri, σ preveliko $\implies$ haos



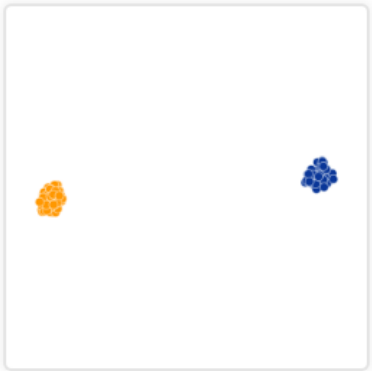
Original



Perplexity: 2
Step: 5,000



Perplexity: 5
Step: 5,000



Perplexity: 30
Step: 5,000

Perplexity: 50
Step: 5,000

Perplexity: 100
Step: 5,000

Perplexity - uticaj

- Malo σ može da dovede do prividnih pravilnosti

Original

Perplexity: 2
Step: 5,000

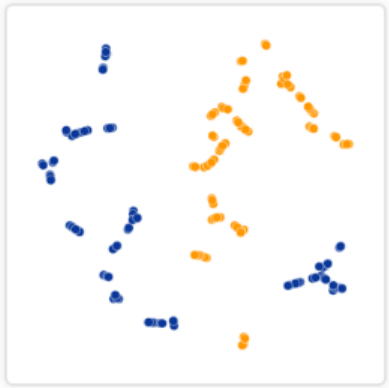
Original

Perplexity: 2
Step: 5,000

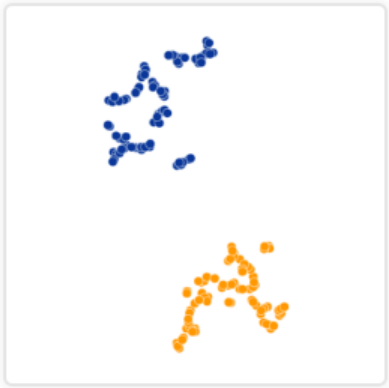
Perplexity - uticaj

- Nije održan odnos veličina klastera (zašto?)

Original

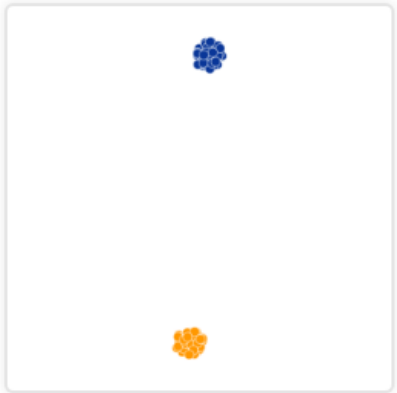


Perplexity: 2
Step: 5,000



Perplexity: 5
Step: 5,000

Perplexity: 30
Step: 5,000



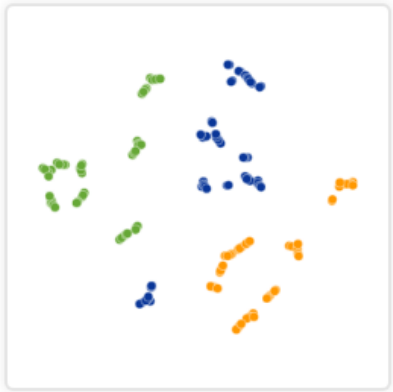
Perplexity: 50
Step: 5,000

Perplexity: 100
Step: 5,000

Perplexity - uticaj

- Nije održan odnos udaljenosti klastera (zašto?)

Original

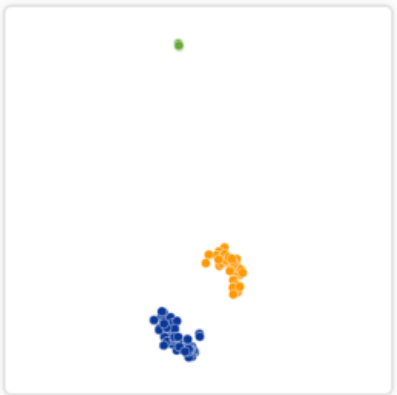


Perplexity: 2
Step: 5,000

Perplexity: 5
Step: 5,000

Perplexity: 30
Step: 5,000

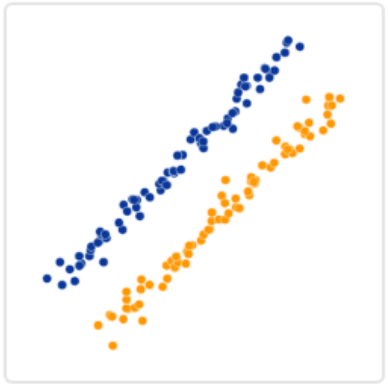
Perplexity: 50
Step: 5,000



Perplexity: 100
Step: 5,000

Perplexity - uticaj

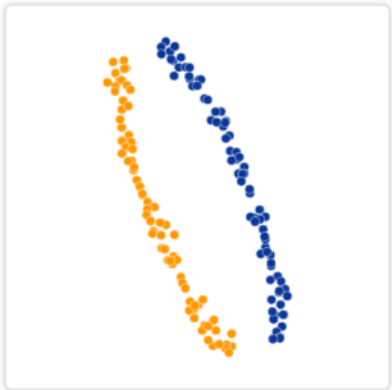
- Pouka: treba uvek posmatrati više grafika odjednom



Original

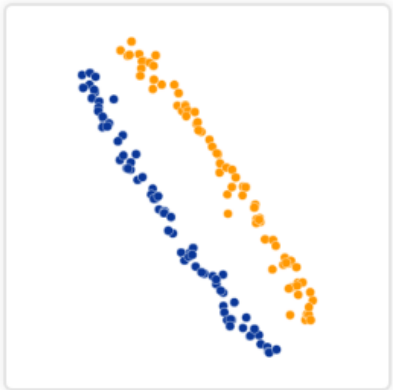
Perplexity: 2
Step: 5,000

Perplexity: 5
Step: 5,000



Perplexity: 30
Step: 5,000

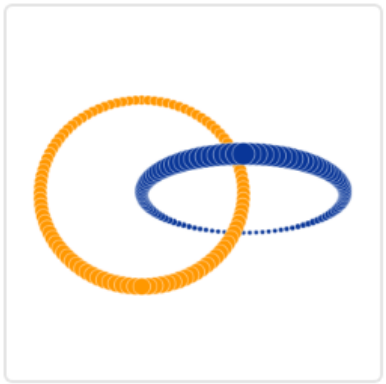
Perplexity: 50
Step: 5,000



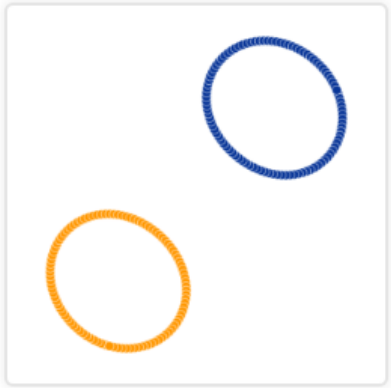
Perplexity: 100
Step: 5,000

Perplexity - uticaj

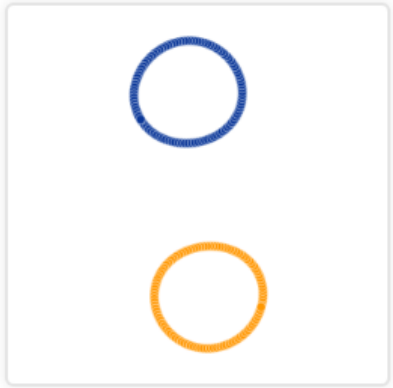
- Pouka: treba uvek posmatrati više grafika odjednom



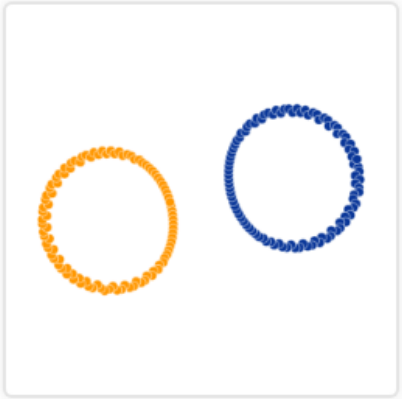
Original



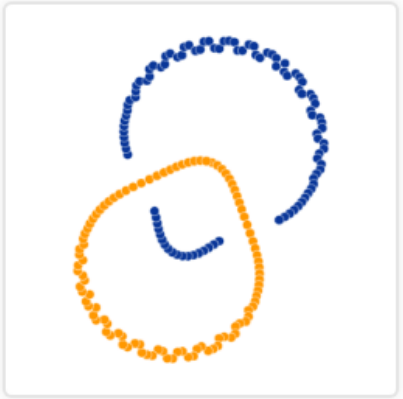
Perplexity: 2
Step: 5,000



Perplexity: 5
Step: 5,000



Perplexity: 30
Step: 5,000



Perplexity: 50
Step: 5,000

Perplexity: 100
Step: 5,000

Rezime

- Problem razumevanja/istraživanja višedimenzionih podataka
- Redukcija dimenzija
- Najpoznatije tehnike redukcije dimenzija, 3 pitanja na koja sada imamo odgovor:
  1. Koje su?
  2. Kako se koriste?
  3. Kako **zapravo** rade?

Hvala na pažnji!

- Pitanja?