Univerza v Ljubljani
Fakulteta za računalništvo in informatiko
Fakulteta za matematiko in fiziko

Maj Alter

Topologija učenja nevronskih mrež

diplomsko delo

interdisciplinarni univerzitetni
študijski program prve stopnje
računalništvo in matematika

Mentor: doc. dr. Dejan Govc

Ljubljana, 2025

Besedilo je oblikovano z urejevalnikom besedil LaTeX.

Kandidat: Maj Alter

Naslov: Topologija učenja nevronskih mrež

Vrsta naloge: Diplomska naloga na interdisciplinarnem univerzitetnem študijskem programu prve stopnje Računalništvo in matematika

Mentor: doc. dr. Dejan Govc

Opis:

TODO: Besedilo teme diplomskega dela študent prepiše iz študijskega informacijskega sistema, kamor ga je vnesel mentor. V nekaj stavkih bo opisal, kaj pričakuje od kandidatovega diplomskega dela. Kaj so cilji, kakšne metode naj uporabi, morda bo zapisal tudi ključno literaturo.

Title: Topologija učenja nevronskih mrež

Description:

TODO: opis diplome v angleščini

Na tem mestu bi se rad zahvalil družini za podporo v času mojega študija in mentorju doc. dr. Dejanu Govcu za vso pomoč pri pisanju diplomskega dela.

Kazalo

Seznam uporabljenih kratic

kratica	**angleško**	**slovensko**
TDA	topological data analysis	topološka analiza podatkov
PCA	principal component analysis	metoda glavnih komponent

Povzetek

Naslov: Topologija učenja nevronskih mrež

Avtor: Maj Alter

Z nevronskimi mrežami lahko rešimo širok nabor problemov, kljub temu pa mnogokrat težko interpretiramo, kaj se ti modeli zares naučijo. Uteži v nevronski mreži se v postopku učenja razvijejo tako, da so zadani problem sposobne rešiti. V diplomskem delu bomo spoznali metodo, s katero lahko s pomočjo topološke analize podatkov zajamemo strukturo sestavljeno iz uteži, ki nastane med učenjem. Delo temelji na članku [3], kjer je metoda opisana in predstavljena na preprostem primeru. Opisali bomo matematično ozadje in koncepte s pomočjo katerih lahko na podlagi algoritma Mapper, grafično interpretiramo nevronsko mrežo.

Ključne besede: Nevronske mreže, Metoda glavnih komponent, Mapper algoritem.

Abstract

Title: Topology of learning in neural networks

Author: Maj Alter

With neural networks, we can solve a wide range of problems, yet it is often difficult to interpret what these models actually learn. The weights in a neural network develop during the learning process to solve the given problem. In this thesis, we will explore a method by which we can use topological data analysis to capture the structure composed of weights that arises during learning. The work is based on the article [1], where the method is described and demonstrated on a simple example. We will describe the mathematical background and concepts that allow us to graphically interpret the neural network based on the Mapper algorithm.

Keywords: Neural networks, Principal component analysis, Mapper algorithm.

Poglavje 1

Uvod

Nevronske mreže so modeli strojnega učenja, ki zelo dobro prepoznavajo vzorce in rešujejo kompleksne probleme. Zaradi njihove uporabe v medicini, ekonomiji, avtonomnih sistemih in še mnogo več, kjer bi pri napaki modela lahko prišlo do hujših posledic, je pomembno da preverimo njegovo verodostojnost. Pri tem je pomembno, da se prepričamo, da pri učenju modela, kljub visoki natančnosti, ni prišlo do zlorabe, kjer bi se recimo odločitve sprejemale na podlagi nekih nepredvidenih spremenljivk v podatkih, ki zares nimajo nobene povezave s problemom. Tukaj govorimo o funkcionalnem razumevanju modela in ne o razumevanju ostalih nizkonovijskih algoritmov in konceptov, saj te že razumemo zelo dobro. Interpretacijo bomo razumeli kot preslikavo abstraktnega koncepta v domeno, ki jo lahko ljudlje razumemo.[11]

V diplomskem delu si bomo ogledali metodo iz članka [3], katere ideja temelji na tem, da bi s topologijo lahko zajeli strukturo, ki nastane v procesu učenja. Najprej bomo predstavili matematično ozadje in ugotovitve predstavili v zaključku.

Cilj interpretacije nevronskih mrež je da bi lahko ljudje bolj zaupali umetni inteligenci ter da bi zagotovili pošteno, etično in odgovorno uporabo te tehnologije.

Poglavje 2

Nevronske mreže

V tem poglavju bomo ponovili zgradbo in delovanje nevronskih mrež. Osredotočili se bomo le na usmerjene nevronske mreže (angl. feed forward) z vzratnim razširjanjem napake (angl. backpropagation); ko bomo omenili nevronsko mrežo bomo imeli v mislih to vrsto. Nevronske mreže so nelinearni modeli, katerih osnovni koncept je, da lahko iz linearne kombinacije vhodnih vrednosti, ki jih lahko preslikamo z neko nelinearno funkcijo, dobimo nove vrednosti.[6]

2.1 Zgradba

Gradniki nevronskih mrež so nevroni, ki so urejeni v sloje (angl. layer). Mrežo si lahko predstavljamo kot usmerjen acikličen graf. Povezave med nevroni so utežene in lahko potekajo le v naslednji sloj. Vrednost nevrona izračunamo, tako da linearno kombinacijo vrednosti nevronov prejšnjega sloja preslikamo s t.i. aktivacijsko funkcijo. Ker želimo nelinearen model, je pomembno, da izberemo nelinearno funkcijo. Vrednostim pravimo tudi aktivacije in ta izraz bomo uporabljali v nadaljevanju.

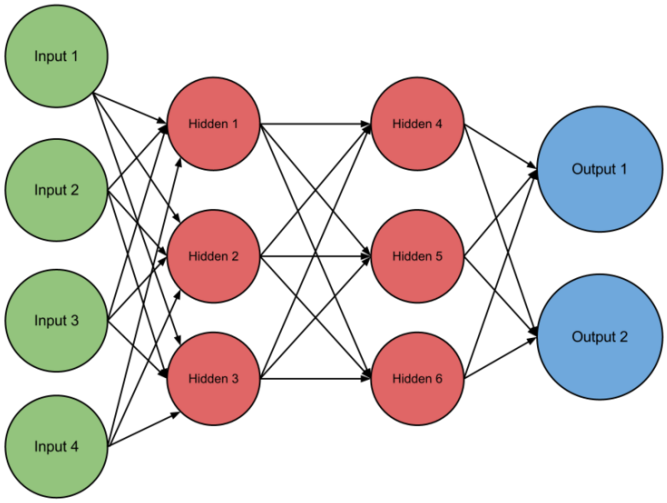
Sloje razdelimo v vhodni, enega ali več skritih in izhodni sloj:

- Vhodni sloj, sprejema vhodne podatke in jih posreduje naslednjemu sloju brez sprememb. Število nevronov v tem sloju določa naša vhodna

množica.

- Skriti sloji so vsi sloji med vhodnim in izhodnim slojem ter lahko vsebujejo različno število nevronov. Ti sloji opravljajo večino dela in transformacij podatkov.

- Izhodni sloj nam vrne končne vrednosti, ki predstavljajo napoved mreže. Število nevronov v tem sloju je odvisno od vrste problema. Pri regresijskih problemih je to ponavadi samo en nevron, pri klasifikacijskih problemih pa je to običajno en nevron za vsak razred. Aktivacije predstavljajo verjetnosti, da vhod klasificiramo v ta razred.



Slika 2.1: Sloji naprej usmerjene nevronske mreže. Vir: [18]

Inspiracija za model nevrona prihaja iz biologije, saj so človeški možgani povezani v mrežo nevronov. V matematičnem modelu povezave predstavljajo sinapse, uteži pa moč povezave. Negativne uteži so zaviralne, pozitivne pa vzpodbujevalne. Aktivnost nevronske celice predstavimo z aktivacijo nevrona.[10]

2.2 Matematična formulacija

Da bomo lahko z nevronsko mrežo računali in opazovali njeno delovanje bomo določene stvari še matematično formulirali. Uporabili bomo naslednjo notacijo:

- Naj bodo $\{L^{(i)} \mid i = 0, \ldots, k\}$ urejeni sloji,
- $N^{(i)}$ število nevronov na sloju $L^{(i)}$,
- $a_j^{(i)}$ aktivacijska funkcija j-tega nevrona v i-tem sloju,
- $z^{(i)}$ linearna kombinacija nevronov $i-1$-tega sloja za $i = 0, 1, \ldots, k$, kjer je $z^{(0)}$ vhodni podatek,
- $h^{(i)}$ aktivacija nevronov na i-tem sloju za $i = 1, \ldots, k$, kjer je $h^{(0)}$ vhodni podatek, $h^{(k)}$ rezultat,
- $W^{(i)}$ matrika $m \times n$ uteži na i-tem sloju, kjer je m število nevronov v i-tem sloju in n število nevronov v predhodnem sloju.

$$W^{(i)} = \begin{bmatrix} w_{11}^{(i)} & w_{12}^{(i)} & \cdots & w_{1n}^{(i)} \\ w_{21}^{(i)} & w_{22}^{(i)} & \cdots & w_{2n}^{(i)} \\ \vdots & \vdots & \ddots & \vdots \\ w_{m1}^{(i)} & w_{m2}^{(i)} & \cdots & w_{mn}^{(i)} \end{bmatrix}$$

Aktivacije nevronov za sloje $i = 1, 2, \ldots, k$ izračunamo po formuli:

$$h^{(i)} = a^{(i)}(W^{(i)} \cdot h^{(i-1)}) \tag{2.1}$$

Aktivacijsko funkcijo $a^{(i)}$ definiramo po komponentah. Rezultat nevronske mreže, torej predstavlja zadnji sloj, ki ga lahko s pomočjo zgornje zveze rekurzivno izračunamo:

$$h^{(k)} = a(W^{(k-1)} \cdot h^{(k-1)}) \tag{2.2}$$

Ustavitveni pogoj je $h^{(0)}$, ki je vhodni podatek.

2.3 Učenje nevronske mreže

V razdelku bomo opisali algoritem vzratnega razširjanja napake, ki je verjetno eden izmed najbolj študiranih algoritmov učenja nevronskih mrež. Sledili bomo razlagi [12]. Učenje v nevronski mreži poteka s prilagajanjem uteži tako, da minimiziramo razlike med dejanskimi izhodnimi vrednostmi in tistimi, ki jih mreža napove. Temu pravimo nadzorovano učenje, kjer imamo podano učno množico podatkov v obliki parov vektorjev vhodnih in izhodnih vrednosti: $\{(x_i, t_i) \mid i = 1, \ldots, p\}$ in iz njih lahko izračunamo napako izhodnega sloja:

$$E = \frac{1}{2} \sum_{i=1}^{p} \|\mathbf{o}_i - \mathbf{t}_i\|^2 \tag{2.3}$$

Z o_i smo označili izhodne vrednosti ki jih vrne mreža. Ker pa je izhod nevronske mreže pravzaprav funkcija uteži 2.2, je tudi E funkcija uteži. Ker Želimo funkcijo napake minimizirati, potrebujemo gradient. Začnemo z zadnjim slojem:

$$\nabla E = \left(\frac{\partial E}{\partial w_1^{(k)}}, \frac{\partial E}{\partial w_2^{(k)}}, \ldots, \frac{\partial E}{\partial w_m^{(k)}} \right) \tag{2.4}$$

Poskrbeti moramo, še da bo aktivacijska funkcija a odvedljiva, pogosta izbira je sigmoidna funkcija:

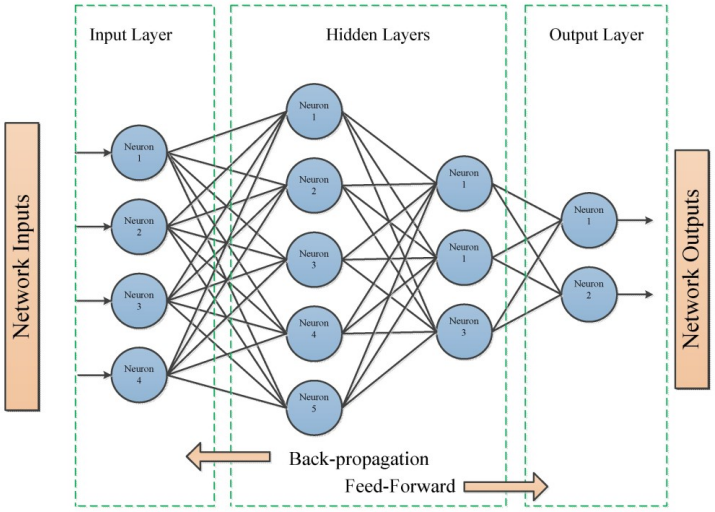
$$\sigma(x) = \frac{1}{1 + e^{-x}}. \tag{2.5}$$

Katere odvod je

$$\frac{d}{dx}\sigma(x) = \frac{e^{-x}}{(1 + e^{-x})^2} = \sigma(x)\left(1 - s(x)\right). \tag{2.6}$$

Sedaj si bo lažje predstavaljati rezultat nevronske mreže in funkcijo napake kot kompozitum velikega števila funkcij in ne več kot rekurzivno funkcijo. Razlog za to je, da bomo lahko pri računanju gradienta uporabili verižno pravilo odvajanja. Uteži nato spreminjamo z metodo gradientnega spusta, ki iterativno zmanjšuje vrednosti uteži v nasprotni smeri vektorja gradienta, dokler ne najde minimuma. Tako se uteži popravijo na zunanjem sloju in nato nadaljujemo s predhodnjim slojem, dokler ne pridemo do začetka. V

zadnjem sloju lahko neposredno primerjamo napovedan izhod z dejanskim izhodom, da izračunamo gradient. V skritih slojih pa se gradient izračuna na podlagi gradienta sloja pred njim, od tod izvira ime vzratno razširjanje napake.



Slika 2.2: Vzratno razširjanje napake. Vir: [1]

Od sedaj naprej nas bo samo zanimala evolucija uteži v procesu učenja. Na posameznih slojih bomo opazovali vektorje uteži za vsak nevron, ki jih bomo interpretirali kot točke v visoko dimenzionalnem prostoru. Za vsak sloj i so to ravno vrstice v matriki uteži $W^{(i)}$. Ker pa si visoko dimenzionalnih prostorov ne moremo predstavljati, se bomo v naslednjem razdelku ukvarjali s tem kako točke preslikati v nižje dimenzionalni prostor s čim manjšo izgubo informacij.

Poglavje 3

Metoda glavnih komponent

Vektorji vhodnih uteži na sloju $N^{(i)}$ so velikosti $N^{(i-1)}$. Te številke so običajno veliko večje od 2 ali 3, zato si teh vektorjev ne moremo predstavljati, kot točk v 2 oz. 3 dimenzionalnem prostoru. V tem poglavju si bomo pogledali linearno metodo za zmanjševanje dimenzij, ki se imenuje metoda glavnih komponent. Pri tem bomo sledili [14].

3.1 Sprememba baze

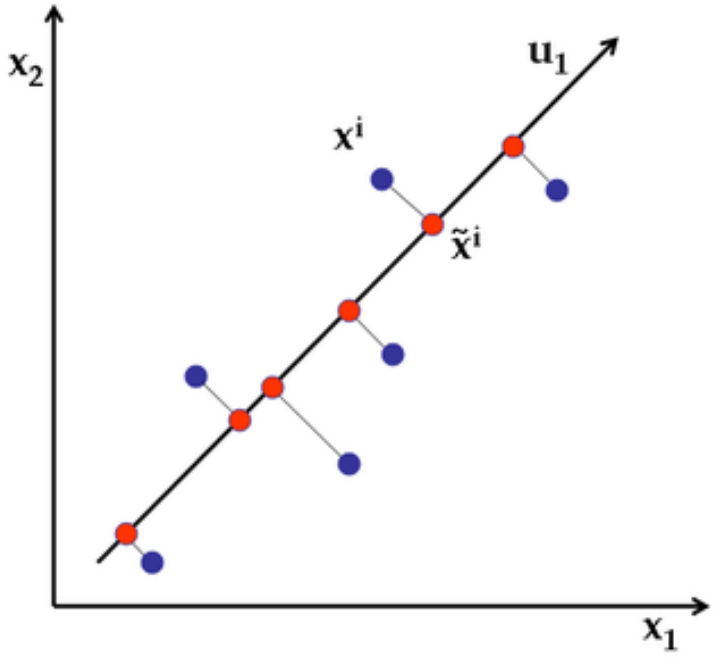
Cilj metode glavnih komponent je zbirko podatkov predstaviti z bazo, katere bazni vektorji bodo po vrsti razkrili največ informacije o originalnih podatkih. Glavna ideja je, da bodo najpomembnejši bazni vektorji zadostovali za predstavo strukture podatkov, medtem ko bodo manj pomembnejši le filtrirali morebitne šume. Originalne podatke, ki smo jih predstavili z matriko X, bomo torej transformirali v matriko Y, ki bo predstavljala podatke zapisane v novi bazi. Vsak nov bazni vektor lahko zapišemo kot linearno kombinacijo originalnih baznih vektorjev, zato lahko poiščemo matriko linearne preslikave P, tako da bo veljalo:

$$Y^T = PX^T \tag{3.1}$$

3.2 Napaka rekonstrukcije in varianca

Vrednost podatkov v novi bazi dobimo s pravokotno projekcijo na nove normirane bazne vektorje. Iskanja nove komponente se lahko lotimo, tako da poiščemo enotski vektor u_1, ki bo po projekciji podatkov minimiziral napako rekonstrukcije originalnega prostora.

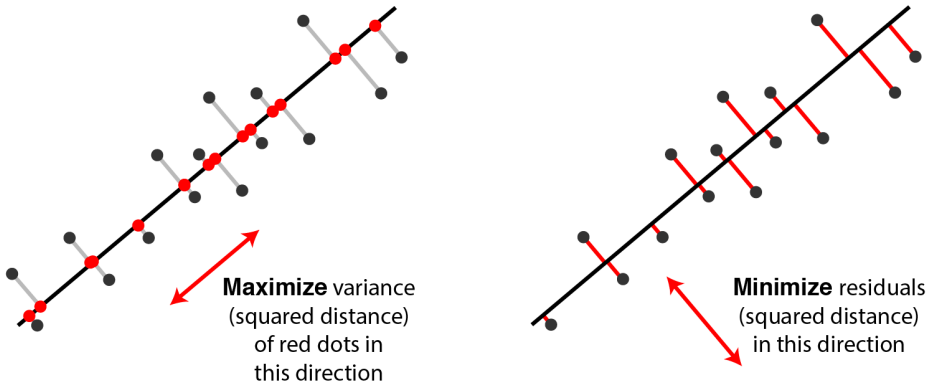
$$E(u_1) = \sum_i \left\| x^{(i)} - (u_1^T x^{(i)}) u_1 \right\|^2 \tag{3.2}$$



Slika 3.1: Napaka rekonstrukcije. Vir: [2]

To pa je ravno ekvivalenten problem iskanja vektorja, ki bo po projekciji originalnih podatkov jih ohranjal najbolj razpršene. Razpršenost merimo z varianco, ki je povprečna oddaljenost projekcije podatkov od projekcije središčne točke podatkov $\bar{x} = \frac{1}{m} \sum_i x^{(i)}$:

$$\mathrm{Var}(u_1^\top X^\top) = \frac{1}{m} \sum_{i=1}^{m} \left(u_1^\top x^{(i)} - u_1^\top \bar{x} \right)^2 \tag{3.3}$$



Slika 3.2: Varianca in napaka rekonstrukcije. Vir: [7]

Dokažimo zgornjo trditev.

Trditev 3.0.1 $u_1 = \underset{u:\|\mathbf{u}\|_2=1}{\operatorname{argmin}} E(u) = \underset{u:\|\mathbf{u}\|_2=1}{\operatorname{argmax}} \operatorname{Var}(u^\top X^\top)$

Dokaz.

$$\mathbf{u_1} = \underset{\mathbf{u}:\|\mathbf{u}\|_2=1}{\operatorname{argmin}} \frac{1}{m} \sum_{i=1}^{m} \left\| \mathbf{u}^{(i)} - (\mathbf{u}^T \mathbf{x}^{(i)})\mathbf{u} \right\|^2 \tag{3.4}$$

$$= \underset{\mathbf{u}:\|\mathbf{u}\|_2=1}{\operatorname{argmin}} \frac{1}{m} \sum_{i=1}^{m} \left\| \mathbf{x}^{(i)} \right\|^2 - (\mathbf{u}^T \mathbf{x}^{(i)})^2 \tag{3.5}$$

$$= \underset{\mathbf{u}:\|\mathbf{u}\|_2=1}{\operatorname{argmax}} \frac{1}{m} \sum_{i=1}^{m} (\mathbf{u}^T \mathbf{x}^{(i)})^2 \tag{3.6}$$

□ [4]

Če enačbo za varianco malenkost preoblikujemo, dobimo naslednje. [19]

$$\begin{aligned}
\mathrm{Var}(u_1^T X) &= \frac{1}{m}\sum_{i=1}^{m}(u_1^T x^{(i)} - u_1^T \bar{x})^2 \\
&= \frac{1}{m}\sum_{i=1}^{m}\left((u_1^T x^{(i)})^2 - 2u_1^T x^{(i)} u_1^T \bar{x} + (u_1^T \bar{x})^2\right) \\
&= \frac{1}{m}\sum_{i=1}^{m} u_1^T\left(x^{(i)}(x^{(i)})^T - 2x^{(i)}\bar{x}^T + \bar{x}\bar{x}^T\right) u_1 \\
&= u^T \underbrace{\left(\frac{1}{m}\sum_{i=1}^{m}(x_i - \mu)(x_i - \mu)^T\right)}_{\text{Kovariančna matrika } C} u
\end{aligned} \tag{3.7}$$

Zgornjo maksimizacijo variance lahko pod pogojem, da je u_1 enotski vektor, rešimo s pomočjo Langrangeovih multiplikatorjev.

$$f(u_1) = u_1^T C u_1 + \lambda_1(1 - u_1^T u_1) \tag{3.8}$$

$$\frac{\partial f(u_1)}{\partial u_1} = Cu_1 - \lambda_1 u_1 = 0 \tag{3.9}$$

Po preoblikovanju enačbe, sledi da so to rešitve ravno lastni vektorji kovariančne matrike.

$$Cu_1 = \lambda_1 u_1 \tag{3.10}$$

Če zgornjo še malo preoblikujemo, ugotovimo, da lastni vektor z največjo lastno vrednostjo ohranja največ informacije o originalnem prostoru.

$$u_1^T C u_1 = u_1^T \lambda_1 u_1 = \lambda_1 u_1^T u_1 = \lambda_1 \tag{3.11}$$

Podobno lahko za ostale nove bazne vektorje izberemo ostale lastne vektorje kovariančne matrike z najvišjimi lastnimi vrednosti.

Opazimo lahko tudi, da je kovariančna matrika simetrična $\Sigma = \Sigma^T$, zato bodo lastni vektorji ortogonalni, torej preslikava, ki smo jo dobili je samo rotacija originalnega prostora.

3.3 Rešitev z uporabo SVD

Lastne vektorje kovariančne matrike C lahko izračunamo s pomočjo singularnega razcepa. [19]

$$C = U\Sigma V^T \tag{3.12}$$

Nato prvih k lastnih vektorjev vstavimo v matriko

$$P = [u_1, u_2, \ldots, u_k] \tag{3.13}$$

Z matriko P, lahko sedaj preslikamo originalne podatke v nižje k-dimenzionalni prostor.

$$y_i = P^T(x_i - \mu) \tag{3.14}$$

Poglavje 4

Topološka analiza podatkov

Topološka analiza podatkov (TDA, ang. topological data analysis) je novejša veda, ki se je začela razvijati šele v zgodnjih letih tega tisočletja. Navdih za njen razvoj izhaja iz tega, da lahko s koncepti iz topologije (vede oblik) in geometrije (vede prostora in dimenzij) razumemo strukturo kompleksnih podatkov kot tudi nekatere podrobne in specifične informacije o njih. Ta pristop nam pomaga vizualizirati in analizirati celotno strukturo in vzorce, ki se skrivajo v podatkih, ter je manj občutljiv na majhne spremembe ali šume. TDA nam ponuja ogromno metod za analizo podatkov, ki so običajno predstavljeni v obliki oblaka oz. množice točk (ang. point cloud).

V tem poglavju bomo obravnavali nekaj osnovnih pojmov TDA, ki nam bodo kasneje v pomoč za vizualizacijo nevronske mreže s pomočjo algoritma Mapper. Defnirali bomo simplicialne komplekse in predstavili različne načine konstrukcije na podatkovnih množicah. Podali bomo Izrek o živcu, ki je glava motivacija za uporabo metod topološke analize podatkov.

4.1 Glavni koraki TDA

Kljub temu, da dandanes obstaja že veliko metod z različnimi aplikacijami, jih večina sledi naslednjim korakom:

1. Vhod predstavlja končna množica točk z določeno razdaljo ali merilom

podobnosti, ki je lahko osnovana na metriki okoljskega prostora (kot je evklidska v R^d) ali na lastni metriki, izračunani iz matrike razdalj med pari. Ta metrika je pogosto vnaprej določena ali diktirana s strani specifične uporabe. Vendar pa je izbira prave metrike ključnega pomena za odkrivanje pomembnih topoloških in geometrijskih lastnosti podatkov.

2. Nad podatki se ustvari kontinuirana struktura, običajno preprost kompleks ali serija takih kompleksov, znana kot filtracija, ki razkriva topologijo ali geometrijo podatkov. Ti kompleksi, ki so v bistvu razširitve grafov v višje dimenzije, pomagajo prikazati strukturo podatkov na različnih ravneh. Izziv je ustvariti strukture, ki ne le natančno odražajo strukturo podatkov, ampak jih je tudi mogoče učinkovito graditi in spreminjati.

3. Iz teh struktur se izluščijo topološke ali geometrijske podrobnosti. To lahko vključuje rekonstrukcijo osnovne oblike podatkov, na primer s triangulacijo, kar olajša dostop do njenih značilnosti, ali ustvarjanje preprostih povzetkov, ki zahtevajo specializirane metode, kot je trajna homologija, za pridobivanje nekaterih informacij. Ključna naloga tukaj je dokazati uporabnost in zanesljivost teh informacij, še posebej, kako se obnesejo v primeru napak ali šuma v podatkih.

4. Razkrite informacije o topologiji in geometriji podatkov nudijo nove vrste značilnosti in opisov. Te lahko izboljšajo naše razumevanje podatkov, zlasti preko vizualnih metod, ali pa se integrirajo z drugimi značilnostmi podatkov za širšo analizo in aplikacije strojnega učenja. Uporaba teh informacij za ustvarjanje prilagojenih modelov analize podatkov in strojnega učenja je ključni del izkoriščanja tega, kar ponujajo orodja TDA.

4.2 Simpleks

V nadaljevanju bomo potrebovali pojem simplicialnih kompleksov, katerih osnovni gradniki so simpleksi, ki jih bomo opisali v tem razdelku. Pri tem bomo sledili [17] in Intuitivno so n-simpleksi geometrijski objekti z $(n+1)$ oglišči, ki ležijo v n-dimenzionalnem prostoru in jih ne moremo vstaviti v nižje dimenzionalni prostor. Za gradnjo n-simpleksa v n-dimenzionalnem prostoru potrebujemo $(n+1)$ afino neodvisnih točk.

Definicija 4.0.1 *Množica točk $\{x_0, \ldots, x_n\}$ v vektorskem prostoru V je **afino neodvisna**, če je $\{x_1 - x_0, \ldots, x_n - x_0\}$ linearno neodvisna.*

Matematično lahko simplekse definiramo na naslednji način.

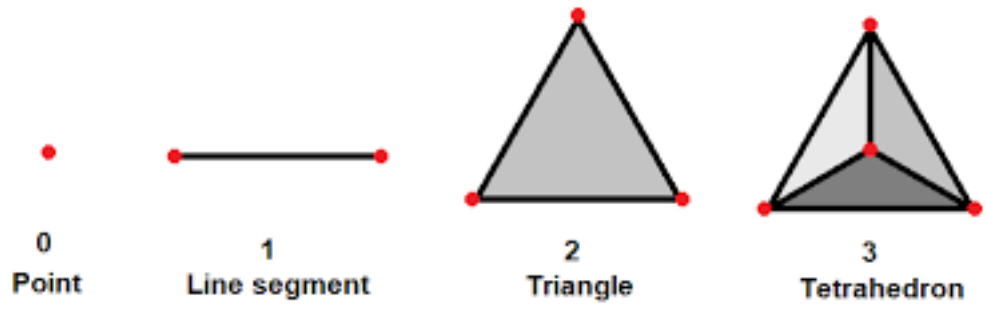
Definicija 4.0.2 *Naj bo $X = \{x_0, x_1, \ldots, x_n\}$ poljubna družina afino neodvisnih točk v prostoru $\mathbb{R}^d$ za nek $n \geq d$. Konveksno ovojnico množice X,*

$$C_n(X) = C_k(x_0, x_1, \ldots, x_n) = \{\sum_{i=0}^{n} \alpha_i x_i : \forall \alpha_i \in [0,1], \sum_{i=0}^{n} \alpha_i = 1\},$$

imenujemo n-simpleks na množici X.

Točke X imenujemo oglišča σ, simplekse ki jih razpenjajo podmnožice X pa imenujemo lica σ.

Simplekse za $n = 0, 1, 2, 3$ si lahko predstavljamo vizualno.



Slika 4.1: Simpleksi v dimenzijah 0, 1, 2 in 3. Vir: [13]

4.3 Simplicialni kompleks

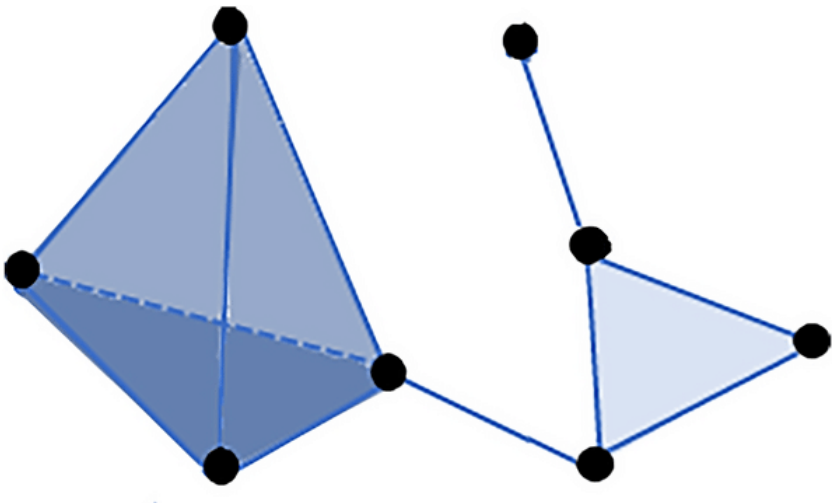
Za analizo topoloških prostorov je zelo pomembno, da jih znamo predstaviti na enostavnejši način. Za to lahko uporabljamo simplicialne komplekse, na katere lahko gledamo kot na kombinatorične opise topološkega prostora.

Definicija 4.0.3 *Simplicialni kompleks K v prostoru R^d je zbirka simpleksov, tako da velja:*

1. *Vsako lice poljubnega simpleksa v simplicialnemu kompleksu K je tudi samo simpleks v K.*
2. *Naj bosta σ_1 in σ_2 poljubna simpleksa v K. Če se sekata, potem je njun presek $\sigma_1 \cap \sigma_2$ lice simpleksov obeh σ_1 in σ_2 ter je simpleks v K.*

Definicija 4.0.4 *Geometrijski simplicialni kompleks K na množici V je končna množica simpleksov, ki zadoščajo naslednjima dvema pogojema:*

1. *Poljubno lice kateregakoli simpleksa K je simpleks v K.*
2. *Presek poljubnih dveh simpleksov v K je prazna množica ali skupno lice obeh simpleksov.*



Slika 4.2: Simplicialni kompleks z osmimi vozlišči Vir: [16]

Definicija 4.0.5 *Abstraktni simpličialni kompleks K na množici V je družina podmnožic množice V, za katero velja*

$$\forall F \in K.\ G \subseteq F \implies G \in K.$$

Množice $F \in K$ imenujemo abstraktni simpleksi, njihovim podmnožicam $G \subseteq F$ pa lica abstraktnega simpleksa F. Dimenzijo abstraktnega simpleksa F definiramo kot $\dim(F) = |F| - 1$. *Dimenzijo abstraktnega simpličialnega kompleksa definiramo kot maksimalno dimenzijo njegovih simpleksov:*

$$\dim(K) = \max(\{\dim(F) \mid F \in K\}).$$

4.4 Gradnja simplicialnih kompleksov iz podatkov

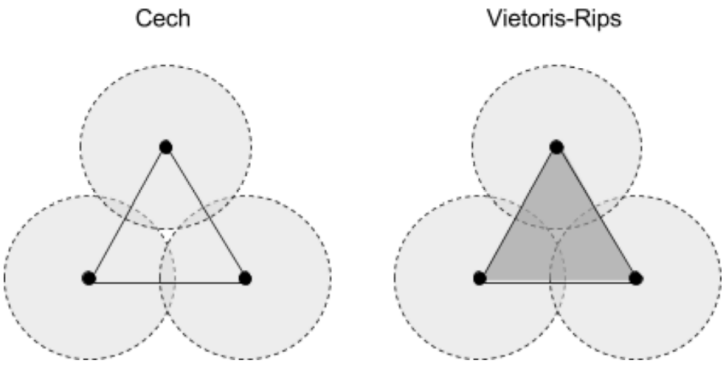
Simplicialne komplekse je možno zgraditi iz množice podatkov. V tem poglavju bomo opisali dva pogosto uporabljena načina. Pri tem bomo sledili [10]. Privzamemo da imamo podano množico točk X v metričnem prostoru (M, d) in realno število $\alpha \geq 0$. Predstavili bomo dva načina, ki sta pogosto uporabljena v praksi. [8]

4.4.1 Vietoris-Ripsov kompleks

Prvi primer temelji na konceptu α-soseščine. Vietoris-Ripsov kompleks $Rips_\alpha(\mathcal{X})$ je množica s
α.za vse (i,j). Po definiciji sledi, da je abstraktni simplicialni kompleks, v
splošnem pa ne velja, da je tudi geometrični.

4.4.2 Čehov kompleks

Zelo podoben Vietoris-Ripsovem kompleksu je Čehov kompleks C_ε, katerega
simpleksi so definirani kot množica simpleksov $[x_0, \ldots, x_k]$, tako da ima k + 1 zaprtih krogel$B$
neprazen presek.



Slika 4.3: Čehov in Vietoris-Ripsov kompleks. Vir: [15]

4.5 Izrek o živcu

V tem razdelku bomo predstavili izrek o živcu, ki je osnova za Mapper algoritem, saj nam zagotavlja, da je živec, ki ga dobimo iz pokritja topološkega prostora homotopsko ekvivalenten prostoru. Izrek bomo podali brez dokaza. [5]

Izrek 4.1 (Izrek o živcu) *Naj bo X topoološki prostor in $\mathcal{U} = \{U_i\}_{i \in I}$ odprto pokritje X. Živec $N(\mathcal{U})$ pokritja $\mathcal{U}$ je simplicialni kompleks:*

- *Vozlišča $N(\mathcal{U})$ ustrezajo mnnožicam U_i v pokritju.*
- *Končna podmnožica $\{U_{i_0}, U_{i_1}, \ldots, U_{i_k}\}$ pokritja $\mathcal{U}$ tvori k-simpleks v $N(\mathcal{U})$, če in samo če je presek $U_{i_0} \cap U_{i_1} \cap \cdots \cap U_{i_k}$ neprazen.*

Če je pokritje $\mathcal{U}$ is dobro (t.j., vsako končno presečišče množic v pokritju je bodisi prazno bodisi kontraktibilno), potem je topološki prostor X homotopsko ekvivalenten živcu $N(\mathcal{U})$. Z drugimi besedami,

$$X \simeq N(\mathcal{U}).$$

Poglavje 5

Mapper algoritem

Algoritem Mapper je orodje za ekstrakcijo globalnih značilnosti iz visokodimenzionalnih podatkov, ki preoblikuje zapletene podatkovne množice v preproste simplicialne komplekse. Ti simplicialni kompleksi omogočajo kompakten globalni prikaz podatkov, neodvisno od individualnih razdalj, kotov ali točk. Algoritem ustvarja graf, ki odraža topološko strukturo izvornega oblaka točk. Začne z izbiro zvezne funkcije, ki se imenuje filter, $f : X \to Z$, ki na primer projicira prostor X na podprostor Z. Nato določi končno odprto pokritje $\mathcal{U} = \{U_i\}_{i \in I}$ prostora Z. S pomočjo algoritma za združevanje v gruče se neodvisno obdela vsaka praslika $f^{-1}(U_i)$, od koder dobimo odprto pokritje $\mathcal{V} = \{V_j\}_{j \in J}$ oblaka točk. Končni Mapperjev graf ali simplicialni kompleks je opredeljen z živcem $\mathcal{V}$, nizom podmnožic $\mathcal{K}$ iz množice J z nepraznimi presečišči. Vsaka podmnožica z $n + 1$ indeksi predstavlja n-simpleks, kjer enočlenske podmnožice tvorijo vozlišča grafa, dvodelne pa njegove robove.

[9]

5.1 Algoritem

V nadaljevanju so opisani koraki algoritma Mapper:

Algorithm 1 Mapper algoritem

Vhod: Zbirka (množica) podatkov X z metriko, funkcija $f : X \to \mathbb{R}$ (ali $\mathbb{R}^d$), in pokritje $\mathcal{U}$ prostora $f(X)$.

Algoritem:

for each $U \in \mathcal{U}$ **do**

Razdeli $f^{-1}(U)$ v skupine $C_{U,1}, \ldots, C_{U,k_U}$.

Izračunaj živec pokritja X, ki ga definira $\{C_{U,1}, \ldots, C_{U,k_U}\}$ for each $U \in \mathcal{U}$.

end for

Izhod: Simplicialni kompleks, živec (pogosto graf za dobro izbrana pokritja):

- Vozlišče $v_{U,i}$ za vsako skupino $C_{U,i}$.
- Povezava $v_{U,i}$ in $v_{U',j}$ če $C_{U,i} \cap C_{U',j} \neq \emptyset$.

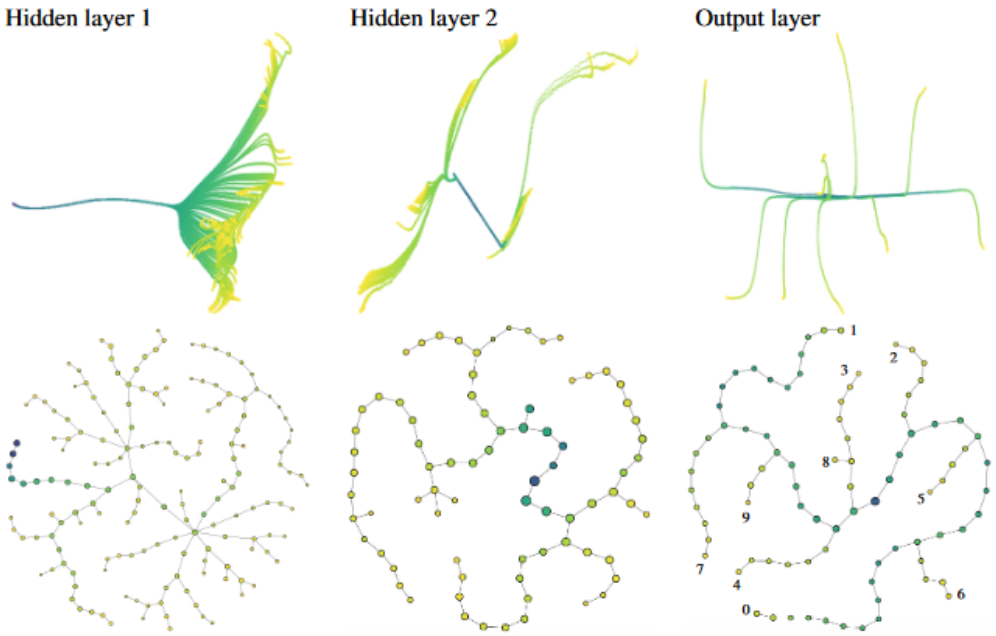
Poglavje 6

Zaključek in rezultati

V tem poglavju si bomo pogledali metodo vizualizacije nevronske mreže in kakšne rezultate dobimo. Pri tem bomo sledili [3].

Pri gradnji Mapper grafa iz nevronske mreže lahko za vsak sloj i z N_i nevroni, spremljamo evolucijo uteži nevronov, ki vstopajo vanj. Z drugimi besedami spremljamo stolpce v matriki uteži med nivojema $i-1$ in i. Nato pa za vsak korak v procesu učenja, ki ga izvedemo z metodo gradientnega spusta dobimo N_i vektorjev uteži velikosti N_{i-1}, kar interpretiramo kot N_i točk v prostoru dimenzije N_{i-1}. Ob koncu učenja dobimo oblak točk z št. korakov$\times$ N_i točk, ki so dimenzije N_{i-1}. Točke so nato s PCA projekcijo projicirane v 2-dimenzionalni prostor. Od tod potem z Mapper algoritmom zgradimo graf.

Rezultati, ki jih bomo predstavili so povzeti iz članka [3]. Pogledali si bomo Mapper graf nevronske mreže, ki je bila zgrajena na podatkovni zbirki MNIST. Gre za veliko podatkovno zbirko ročno napisanih števk (slik velikosti 28×28), ki se pogosto uporablja za učenje in testiranje metod strojnega učenja. Nevronska mreža ima dva skrita nivoja s 100 nevroni in sigmoidno aktivacijsko funkcijo. Začetne vrednosti uteži so bile nastavljene na naključne majhne vrednosti. Za filter funkcijo je bila uporabljena L^2 norma in DBSCAN kot algoritem gručenja.



Slika 6.1: Evolucija uteži učenja nevronske mreže. Vir: [3]

Z modro barvo so predstavljene uteži na začetku učenja in z rumeno na koncu. Opazimo, da se graf v zadnjem sloju razveji na 10 vej, kjer vsaka predstavlja eno izmed desetih števk. V skritem nivoju pa lahko opazimo, da se uteži nekaj časa razvijajo v enaki smeri in se šele nato razvejijo, kar bi lahko bila posledica PCA projekcije. Omenimo še, da so smeri razvejanja v različnih poskusih učenja ostale skoraj konstantne. V zadnjem sloju opazimo, da se graf že na začetku usmeri na dve nasprotni veji, ki se nato še dalje razvejita. Vidno je tudi, da so si nekatere števke zelo podobne, recimo osmica in trojka sta si zelo sorodni.

Literatura

[1] Maher G. M. Abdolrasol in sod. "Artificial Neural Networks Based Optimization Techniques: A Review". V: *Electronics* 10.21 (2021). ISSN: 2079-9292. DOI: 10.3390/electronics10212689. URL: https://www.mdpi.com/2079-9292/10/21/2689.

[2] Humayra Ali in David Powers. *Published DICTA Humayra*. Sep. 2014.

[3] Maxime Gabella. "Topology of Learning in Feedforward Neural Networks". V: *IEEE Transactions on Neural Networks and Learning Systems* 32.8 (avg. 2021), str. 3588–3592. ISSN: 2162-2388. DOI: 10.1109/tnnls.2020.3015790. URL: http://dx.doi.org/10.1109/TNNLS.2020.3015790.

[4] Matthew Gormley. *Lecture 11: Principal Component Analysis (PCA)*. http://www.cs.cmu.edu/~mgormley/courses/606-607-f18/slides606/lecture11-pca.pdf. Accessed: 2024-07-31. 2018.

[5] Laura Guzelj Blatnik. "Nevronske mreže z vzvratnim razširjanjem napak v funkcijskem programskem jeziku: delo diplomskega seminarja". Dostopano 31 marec 2024. Doktorska disertacija. Diplomsko delo, 2020. URL: https://repozitorij.uni-lj.si/IzpisGradiva.php?lang=slv&id=117661.

[6] Trevor Hastie, Robert Tibshirani in Jerome Friedman. *The Elements of Statistical Learning*. Springer New York, 2009. DOI: 10.1007/978-0-387-84858-7.

[7] Amit Khumar. *What is an intuitive explanation for PCA?* https://www.quora.com/What-is-an-intuitive-explanation-for-PCA. Accessed: 2024-07-31. Year.

[8] Jeremy Kun. *Čech, Vietoris-Rips, and Dowker Nerves*. https://www.jeremykun.com/2015/08/06/cech-vietoris-rips-complex/. Accessed: 2024-07-31. 2015.

[9] Gretchen Langenbahn. *Topological Data Analysis with Mapper*. https://scholarworks.bgsu.edu/honorsprojects/732. Honors Projects. 732. 2022.

[10] Bohdan Macukow. "Neural Networks – State of Art, Brief History, Basic Models and Architecture". V: *Computer Information Systems and Industrial Management*. Ur. Khalid Saeed in Władysław Homenda. Cham: Springer International Publishing, 2016, str. 3–14. ISBN: 978-3-319-45378-1.

[11] Grégoire Montavon, Wojciech Samek in Klaus-Robert Müller. "Methods for interpreting and understanding deep neural networks". V: *Digital Signal Processing* 73 (2018), str. 1–15. ISSN: 1051-2004. DOI: https://doi.org/10.1016/j.dsp.2017.10.011. URL: https://www.sciencedirect.com/science/article/pii/S1051200417302385.

[12] Raul Rojas in Raúl Rojas. "The backpropagation algorithm". V: *Neural networks: a systematic introduction* (1996), str. 149–182.

[13] Jesse Schneider. *Simplexes*. http://homepages.math.uic.edu/~jschnei3/Writing/Simplexes. Accessed: 2024-07-31. 2024.

[14] Jonathon Shlens. *A Tutorial on Principal Component Analysis*. 2014. arXiv: 1404.1100 [cs.LG].

[15] Justin Skycak. *Intuiting Persistent Homology*. https://www.justinmath.com/intuiting-persistent-homology/. Accessed: 2024-07-31. 2023.

[16] Unknown. *Simplicial Complex: An example of a simplicial complex composed of eight vertices.* https://www.researchgate.net/publication/358508552/figure/fig4/AS:1134810642817026@1647571349635/Simplicial-complex-An-example-of-a-simplicial-complex-composed-of-eight-vertices.png. Accessed: 2024-07-31. 2022.

[17] Živa Urbančič. "Aproksimacija mnogoterosti z mehkimi simplicialnimi množicami in njena implementacija v algoritmu UMAP". Doktorska disertacija. 2020. URL: https://repozitorij.uni-lj.si/IzpisGradiva.php?lang=slv&id=120124.

[18] Sai Venkatesh. *The Role of CSPs in Grabbing Opportunities for Survival.* https://www.linkedin.com/pulse/role-csps-grabbing-opportunities-survival-sai-venkatesh/. Accessed: 2024-07-31. 2021.

[19] Blaz Zupan. *Uvod v Odkrivanje Znanj iz Podatkov.* https://github.com/BlazZupan/uozp-zapiski/blob/master/zapiski.pdf. Accessed: 2024-07-31. 2024.