Višerezolucijska detekcija anomalija korištenjem difuzijskih preslikavanja

Doris Đivanović i Karlo Gjogolović

7. svibnja 2025.

Sadržaj

1 Uvod

U raznim problemima koji se odnose na analizu podataka, podaci iz stvarnog svijeta često su velikih dimenzija, pa je s njima teško raditi. Zbog toga je korisno pronaći eventualni način za preslikati takve podatke u neki prostor puno manje dimenzije, odnosno reprezentirati ih podacima iz takvog prostora. Tada, umjesto s originalnim, možemo raditi s tim novodobivenim podacima, a, zbog male dimenzije, to je puno lakše i praktičnije, što u smislu vizualizacije i intuicije, što u smislu veće efikasnosti računskih operacija nad njima.

Standardna klasa problema u analizi podataka su problemi klasteriranja. Jedan takav problem je i problem detekcije anomalija u podacima. U takvim problemima, pretpostavlja se da su podaci koji su anomalija daleko od ostalih normalnih podataka, i njih želimo odvojiti u zaseban klaster. Za probleme klasteriranja veliku važnost igraju koncepti povezanosti, sličnosti ili bliskosti podataka. Stoga bi nam bilo korisno kada bismo podatke velike dimenzije mogli reprezentirati niždimenzionalnim podacima preslikavanjem koje čuva bliskost, tj. udaljenost među podacima.

2 Definicija difuzijskog preslikavanja

2.1 Konstrukcija težinskog grafa skupa podataka

Neka je

$$\Gamma = \{x_1, ..., x_n\}$$

skup od n podataka iz prostora velike dimenzije. Konstruira se težinski graf gdje su čvorovi podaci x_i, a težina brida $w(x_i, x_j)$ je mjera sličnosti (afiniteta) točaka x_i i x_j.

Budući da je težinska funkcija mjera sličnosti dvaju podataka, logično je i očekivano da je simetrična, dakle

$$w(x_i, x_j) = w(x_j, x_i) \qquad \forall x_i, x_j \in \Gamma,$$

i nenegativna, dakle

$$w(x_i, x_j) \geq 0 \qquad \forall x_i, x_j \in \Gamma.$$

Nadalje, često se očekuje i da poprima vrijednosti unutar segmenta $[0, 1]$, s tim da za svaki podatak x_i vrijedi $w(x_i, x_i) = 1$, jer je on potpuno sličan sam sebi. Dakle, očekuje se i da je normalizirana:

$$w(x_i, x_j) \in [0, 1], \quad w(x_i, x_i) = 1, \qquad \forall x_i, x_j \in \Gamma.$$

Prirodno je mjeru sličnosti dvaju podataka povezati s njihovom međusobnom (npr. Euklidskom) udaljenosti, i to na način da je sličnost veća što je udaljenost manja, odnosno da su obrnuto proporcionalne. Dakle, postoji funkcija $f : \mathbb{R} \to \mathbb{R}^+$, f proporcionalna s $|| \cdot ||_2$, takva da vrijedi:

$$w(x_i, x_j) = \frac{1}{f(||x_i - x_j||_2)}, \qquad \forall x_i, x_j \in \Gamma.$$

Udaljenost je također kao takva mjera simetrična i nenegativna, pa je i težinska funkcija gornjeg oblika takva.

Moglo bi se uzeti da težinska funkcija o euklidskoj udaljenosti ovisi eksponencijalno, dakle

$$w(x_i, x_j) = \frac{1}{e^{||x_i - x_j||_2^2}} = e^{-||x_i - x_j||_2^2}.$$

Budući da je udaljenost podatka do samog sebe jednaka nuli, te da je

$$e^{||x_i - x_j||_2^2} \geq 1,$$

težinska funkcija gornjeg oblika bit će i normalizirana.

Bilo bi još dobro težinsku funkciju skalirati nekim faktorom, $\sigma > 0$. Za taj bi se faktor mogao uzeti neki varijabilni parametar pa promatrati njegov utjecaj s obzirom na njegove različite vrijednosti.

Ako želimo da težinska funkcija bude proporcionalna tom parametru, bilo bi

$$w(x_i, x_j) = e^{-\frac{||x_i - x_j||_2^2}{\sigma}}, \qquad \sigma > 0,$$

odnosno, ukoliko želimo naznačiti da oba podatka doprinose težini s tim faktorom,

$$w(x_i, x_j) = e^{-\frac{||x_i - x_j||_2^2}{\sigma^2}}, \qquad \sigma > 0.$$

Ova funkcija vrlo je poznata i u literaturi se naziva **gaussovska** ili **Gaussova jezgra**.

Postavlja se pitanje kako odabrati parametar σ. Vrijednost ovakve težinske funkcije, dakle, proporcionalna je vrijednosti parametra σ. Stoga, ako za σ uzmemo neku jako malu vrijednost, većina će težina u grafu biti jako mala - u smislu jako bliska 0, osim težine svakog podatka za samog sebe, koja je, neovisno o parametru σ, uvijek jednaka 1, pa bismo dobili graf koji je jako nepovezan, tj. u kojem je većina podataka povezana samo sa sobom, tj. jedini njihov susjed su oni sami sebi. Analogno, ako za σ uzmemo neku jako veliku vrijednost, većina će težina u grafu biti jako velika - u smislu jako bliska 1, pa bismo dobili graf u kojem su gotovo svi podaci međusobno povezani. To je loše za primjene, posebno i za problem detekcije anomalija. Naime, za detekciju anomalije, pretpostavlja se da su podaci koji su anomalija jako udaljeni od svih ostalih podataka, izolirani su od tzv. *pozadine*, dakle nalaze se u rijetko *naseljenom* području, dok su *normalni* podaci bliski ostalim podacima, pripadaju *pozadini*, dakle nalaze se u gusto *naseljenom* području. Stoga, kada bi σ bio mali, dakle kada bi graf bio nepovezan, zaključili bismo da su svi podaci anomalija. Analogno, kada bi σ bio velik, dakle kada bi graf bio jako povezan, zaključili bismo da su skoro svi podaci *normalni*. Primijetimo i da bi težinska funkcija gornjeg oblika ovisila isključivo o međusobnoj euklidskoj udaljenosti odogovarajućih dvaju podataka, dakle nikako ne bi ovisila o ostatku skupa podataka.

Faktor σ očito skalira, tj. povećava ili umanjuje, utjecaj euklidske udaljenosti dvaju podataka na mjeru njihove sličnosti. Ključno ga je empirijski odrediti tako da je umanjuje ili uvećava onda kada je prikladno.

Mogli bismo uzeti neku *usrednjenu* vrijednost za taj parametar, na primjer, medijan svih međusobnih udaljenosti podataka u skupu, ili, na primjer, standardnu devijaciju udaljenosti (tada je σ^2 varijanca). Također, budući da je takav σ *globalan* za skup podataka, tj. ovisi o svim međusobnim udaljenostima, time bismo postigli i da težinska funkcija ovisi o međusobnim udaljenostima svih podataka u skupu. Međutim, tada bi za svaka dva podatka njihova težina ovisila o ostalim podacima na isti način. Dakle, možda bi bilo dobro sličnost između dvaju podataka skalirati nekim faktorom koji ovisi baš o tim konkretnim podacima.

Budući da za detekciju anomalije želimo da mjera sličnosti oslikava lokalnu geometriju, sličnost dvaju podataka mogli bismo, štoviše, skalirati takvim faktorom koji odražava lokalna svojstva tih podataka. Možemo definirati preslikavanje

$$x_i \longrightarrow \sigma_i, \qquad \sigma_i > 0,$$

koje svakom podatku pridružuje faktor koji ovisi samo o njegovoj maloj okolini, a ne cijelom skupu podataka, te definirati težinsku funkciju tako da oba podatka doprinose sa svojim faktorom, dakle ovako:

$$w(x_i, x_j) = e^{-\frac{||x_i - x_j||_2^2}{\sigma_i \sigma_j}}, \qquad \sigma_i, \sigma_j > 0.$$

Mala okolina podatka mogla bi se, na primjer, odrediti s k najbližih susjeda podatka, u smislu euklidske udaljenosti.

U literaturi za ovaj seminarski rad koristi se preslikavanje koje svakom podatku x_i pridružuje njegovu euklidsku udaljenost (ili njezin kvadrat) do njegovog K-tog najbližeg susjeda x_K, $K < n$, dakle

$$\sigma_i = ||x_i - x_K||_2^2, \qquad \forall x_i \in \Gamma.$$

Ovime se, dakle, postiže da težina za dva podatka u grafu kojeg kreiramo ovisi i o udaljenostima s ostalim podacima u skupu, ali tako da odražava lokalnu geometriju tih dvaju podataka.

Zašto je baš takvo preslikavanje σ prikladno za problem detekcije anomalija u podacima, ili općenito problem klasteriranja? Pretpostavljamo, dakle, da se podaci koji nisu anomalija, odnosno u tom smislu *normalni* podaci, nalaze u gusto *naseljenom* području, tj. nalaze se blizu većine ostalih podataka iz skupa podataka, većina ostalih podataka su im susjedi, tj. povezani su s njima. Podaci koji jesu anomalija nalaze se u rijetko *naseljenom* području, odnosno daleko od skoro svih ostalih podataka iz skupa podataka. Lokalnu geometriju možemo mjeriti ili odrediti na više načina. Na primjer, mogli bismo unaprijed definirati neki radijus za okolinu podatka koju smatramo malom te za svaki podatak odrediti broj susjeda u njegovoj bliskoj okolini. Drugi pristup je određivanjem k najbližih susjeda, čime za svaki podatak dobijemo radijus u kojem se nalazi njemu najbližih k susjeda. Iz toga da je za neki podatak taj radijus velik, tj. da je k-ti najbliži susjed jako udaljen, možemo zaključiti da su i svi preostali podaci iz skupa (dakle, preostalih $n - k - 1$ podataka) jako udaljeni od njega, tj. barem više od tog radijusa, pa da se on nalazi u rijetko *naseljenoj* okolini, dakle da je anomalija. Za podatak za koji je k-ti najbliži susjed jako blizu, možemo pretpostaviti da se nalazi u gusto *naseljenoj* okolini, odnosno da je *normalan* podatak koji pripada *pozadini*. Dakle, kada računamo težinu, tj. mjeru sličnosti dvaju podataka, formulom

$$w(x_i, x_j) = e^{-\frac{||x_i - x_j||_2^2}{\sigma_i \sigma_j}}, \qquad \sigma_i, \sigma_j > 0,$$

uzimamo u obzir koliko je ostatak skupa podataka udaljen od svakog od tih dvaju podataka, tj. njihov odnos s ostatkom skupa podataka, dakle težinska funkcija stvarno odražava lokalnu geometriju.

Očito, ako su i σ_i i σ_j veliki, dakle ako su i x_i i x_j jako udaljeni od ostatka skupa podataka, onda će i produkt $\sigma_i \sigma_j$ biti velik, pa će težina (mjera sličnosti) za ta dva podatka biti jako velika, što ima smisla, jer ako su oba podatka daleko od ostatka skupa, oba vjerojatno pripadaju anomaliji. Nadalje, ako su i σ_i i σ_j mali, tj. i x_i i x_j su gusto okruženi podacima, onda će i produkt $\sigma_i \sigma_j$ biti mali, pa će težina (mjera sličnosti) za ta dva podatka biti mala, što ima smisla, jer ako su oba podatka okružena drugim podacima, oba vjerojatno pripadaju *pozadini*, i ne moraju biti bliski, jer gotovo svi podaci pripadaju *pozadini*, tj. *pozadina* zauzima većinu slike.

Težinska funkcija $w : \Gamma \times \Gamma \to \mathbb{R}^+$, za $\Gamma = \{x_1, \ldots, x_n\}$, definira kvadratnu matricu W dimenzije n, takvu da

$$w_{ij} = w(x_i, x_j), \quad i, j \in \{1, \ldots, n\},$$

koja se standardno naziva matrica sličnosti. Matrica W očito je simetrična i s jedinicama na dijagonali.

2.2 Slučajna šetnja i matrica prijelaza

Težinska funkcija $w : \Gamma \times \Gamma \to \mathbb{R}^+$ standardno se normalizira, dijeljenjem sličnosti određenog podatka i nekog drugog podatka iz skupa sumom sličnosti tog podatka sa svim ostalim podacima, koja predstavlja stupanj povezanosti tog podatka s ostatkom skupa,

$$d(x_i) = \sum_{x_j \in \Gamma} w(x_i, x_j), \qquad \forall\, x_i \in \Gamma,$$

čime se dobiva nova funkcija

$$p(x_i, x_j) = \frac{w(x_i, x_j)}{d(x_i)}, \qquad \forall\, x_i, x_j \in \Gamma.$$

Očito, vrijedi

$$\sum_{x_j \in \Gamma} p(x_i, x_j) = 1, \qquad p(x_i, x_j) \geq 0, \qquad \forall\, x_i, x_j \in \Gamma,$$

stoga se vrijednost $p(x_i, x_j)$, za svaki x_i i x_j iz Γ, može shvatiti kao vjerojatnost prijelaza iz podatka x_i u podatak x_j u jednom koraku, tj. funkcija p definira slučajnu šetnju na skupu podataka Γ.

Tada je matrica P s elementima

$$p_{ij} = p(x_i, x_j), \quad i, j \in \{1, \ldots, n\},$$

retčano stohastička matrica prijelaza Markovljeva lanca na skupu Γ.

Nadalje, matrica P^t s elementima

$$p^t_{ij} = p_t(x_i, x_j), \quad i, j \in \{1, \ldots, n\},$$

čije vrijednosti predstavljaju vjerojatnosti prijelaza iz podatka x_i u podatak x_j u t koraka, je matrica prijelaza u t koraka, tj. potenciranje matrice P t puta odgovara izvođenju lanca t koraka unaprijed.

Poznato je da retčano stohastička matrica P ima potpun skup svojstvenih vrijednosti, od kojih je jedna i najveća $\lambda = 1$, čiji je svojstveni potprostor skup svih višekratnika jediničnog vektora.

Nadalje, spektar retčano stohastičke matrice P brzo opada, dakle vrijedi

$$1 = |\lambda_0| \geq |\lambda_1| \geq \ldots.$$

Ukoliko s D označimo dijagonalnu matricu čiji su dijagonalni elementi redom

$$d_{ii} = d(x_i) = \sum_{x_j \in \Gamma} w(x_i, x_j), \quad \forall\, i \in \{1, \ldots, n\},$$

matricu P očito možemo prikazati kao

$$P = D^{-1} W.$$

Matrica P stoga je slična simetričnoj matrici

$$P_s = D^{-1/2} W D^{-1/2},$$

gdje je

$$D^{-1/2} D^{-1/2} = D^{-1},$$

pa te matrice imaju iste svojstvene vrijednosti, a svojstveni vektori im se razlikuju za faktor $D^{-1/2}$.

Kako je matrica P_s simetrična, ima ortonormirani skup lijevih svojstvenih vektora i možemo ga dobiti spektralnom dekompozicijom

$$P_s = U \Lambda U^T,$$

pa skup lijevih svojstvenih vektora matrice P možemo dobiti kao

$$\Phi = D^{-1/2} U.$$

Markovljev lanac matrice P može se prikazati s pomoću lijevih svojstvenih vektora, ϕ_i, i desnih svojstvenih vektora, ψ_i, matrice P, ovako:

$$p_t(x_i, x_j) = \sum_{l \geq 0} \lambda_l^t \psi_l(x_i) \phi_l(x_j).$$

Za ortonormirani skup lijevih svojstvenih vektora matrice P, očito bismo svakom podatku x_i iz Γ mogli pridružiti težinsku sumu njemu pripadnih koordinata lijevih svojstvenih vektora s odgovarajućim svojstvenim vrijednostima kao težinama, dakle

$$x_i \longrightarrow (\lambda_0 \phi_0(x_i), \lambda_1 \phi_1(x_i), \lambda_2 \phi_2(x_i), ..., \lambda_n \phi_n(x_i))^T.$$

Budući da je ϕ_0 konstantan, možemo ga izostaviti. Nadalje, zbog brzog opadanja spektra svojstvenih vrijednosti (λ_l brzo teže k nuli), za određenu točnost gornje sume dovoljno je uzeti određeni broj vodećih svojstvenih parova, neki $l < n$.

Stoga, svakom podatku x_i iz Γ mogli bismo pridružiti

$$x_i \in \Gamma \longrightarrow (\lambda_1 \phi_1(x_i), \lambda_2 \phi_2(x_i), ..., \lambda_\ell \phi_\ell(x_i))^T \in \mathbb{R}^\ell,$$

odnosno

$$x_i \in \Gamma \longrightarrow \left(\lambda_1^t \phi_1(x_i), \lambda_2^t \phi_2(x_i), ..., \lambda_\ell^t \phi_\ell(x_i)\right)^T \in \mathbb{R}^\ell,$$

za odabrani broj koraka t.

Očito je gornje preslikavanje dobar kandidat za redukciju dimenzije, dakle za ulaganje originalnih podataka velike dimenzije u prostor puno manje dimenzije, tj. za difuzijsko preslikavanje.

Difuzijska udaljenost, tj. mjera bliskosti podataka, u prostoru Γ može se definirati s

$$D_t^2(x_i, x_j) = \sum_{x_j \in \Gamma} \frac{(p_t(x_i, x_j) - p_t(x_i, x_j))^2}{\psi_0(x_j)}, \qquad \forall\ x_i, x_j \in \Gamma.$$

Zašto je ovu funkciju dobro uzeti kao difuzijsku udaljenost, dakle za mjeru bliskosti u prostoru Γ? Ova funkcija mjeri sličnost distribucija vjerojatnosti podataka x_i i x_j nakon t koraka. Intuitivno, udaljenost između podataka x_i i x_j je mala, tj. bliskost je velika, ako postoji puno

kratkih puteva koji povezuju x_i i x_j . Nadalje, ovakva mjera bliskosti robusna je na šum jer ovisi o svim putevima duljine t u skupu podataka, dakle, za razliku od euklidske udaljenosti, ovisi o cijelom skupu podataka.

Ovako definirana difuzijska udaljenost može se izraziti i s pomoću svojstvenih vektora:

$$D_t^2(x_i, x_j) = \sum_{x_j \in \Gamma} \frac{(p_t(x_i, x_j) - p_t(x_i, x_j))^2}{\psi_0(x_j)} = \sum_{j \geq 1} \lambda_j^{2t} (\phi_j(x_i) - \phi_j(x_j))^2.$$

Ponovno, zbog brzog opadanja spektra, difuzijska daljenost može se dovoljno dobro aproksimirati koristeći samo prvih l svojstvenih parova, što je računski efikasno.

2.3 Difuzijsko preslikavanje i difuzijska udaljenost

Može se pokazati da, ukoliko difuzijsko preslikavanje na prostoru Γ definiramo kao

$$\Psi_t : x_i \in \Gamma \longrightarrow \left(\lambda_1^t \phi_1(x_i), \lambda_2^t \phi_2(x_i), ..., \lambda_\ell^t \phi_\ell(x_i)\right)^T \in \mathbb{R}^\ell,$$

difuzijska udaljenost u prostoru Γ definirana s

$$D_t^2(x_i, x_j) = \sum_{x_j \in \Gamma} \frac{(p_t(x_i, x_j) - p_t(x_i, x_j))^2}{\psi_0(x_j)}, \qquad \forall\ x_i, x_j \in \Gamma,$$

jednaka je euklidskoj udaljenosti u prostoru difuzijskog preslikavanja, dakle vrijedi

$$D_t^2(x_i, x_j) = \|\Psi_t(x_i) - \Psi_t(x_j)\|^2, \qquad \forall\ x_i, x_j \in \Gamma.$$

Dakle, ovako definirano difuzijsko preslikavanje podatke iz prostora velike dimenzije preslikava u prostor puno manje dimenzije, ali i čuva bliskost podataka.

3 Proširenje preslikavanja s podskupa na cijeli skup

Za velike skupove podataka, poput, na primjer, digitalnih slika, računanje difuzijskog preslikavanja za sve podatke je nepraktično. Umjesto toga, preslikavanje se računa za odabrani manji skup podataka $\Gamma \subseteq \overline{\Gamma}$, a zatim se proširuje na sve podatke u $\overline{\Gamma}$ s pomoću metoda za proširenje preslikavanja. Poznate metode su Nyströmova ekstenzija i geometrijske harmonike.

U literaturi za ovaj seminarski rad koristi se metoda proširenja preslikavanja temeljena na *višerezolucijskoj Laplaceovoj piramidi*. Metoda je iterativna, a u svakoj iteraciji, odnosno na svakoj *razini piramide*, konstruira se aproksimacija preslikavanja, tj. funkcije, f (u našem slučaju, jedne difuzijske koordinate Ψ_j), a zatim se razlika, to jest rezidual, funkcije f i izračunate aproksimacije koristi kao funkcija koja će se aproksimirati u sljedećoj iteraciji. U svakoj se sljedećoj iteraciji rezidual aproksimira s pomoću normalizirane Gaussove jezgre sa sve finijim faktorom σ. Na najnižoj se *razini*, $l = 0$, Gaussova jezgra definira s unaprijed zadanim faktorom σ_0. Na sljedećim se razinama, $l \geq 1$, Gaussova jezgra definira sa $\sigma = \frac{\sigma_0}{2^l}$.

Reprezentacija funkcije f *Laplaceovom piramidom* na skupu Γ definirana je iterativno s

$$s_0(x_k) = \sum_{i=1}^{n} k_0(x_i, x_k) f(x_i), \quad \forall x_k \in \Gamma$$

$$s_l(x_k) = \sum_{i=1}^{n} k_l(x_i, x_k) d_l(x_i), \quad \forall x_k \in \Gamma, \quad l \geq 1,$$

gdje je K_l,

$$K_{l(ij)} = k_l(x_i, x_j),$$

normalizirana Gaussova jezgra s faktorom

$$\sigma_l = \frac{\sigma_0}{2^l}$$

na skupu Γ, a rezidual na skupu Γ definiran je s

$$d_l(x_k) = f(x_k) - \sum_{m=0}^{l-1} s_m(x_k), \quad \forall x_k \in \Gamma, \quad l \geq 1.$$

Iteracije se provode dok pogreška aproksimacije, dakle

$$||f - \sum_{m=0}^{l-1} s_m||,$$

ne padne ispod unaprijed zadanog praga.

Proširenje funkcije f na novu točku $\overline{x}_k \in \overline{\Gamma}$ definira se kao

$$f(\overline{x_k}) = \sum_i s_i(\overline{x_k}),$$

gdje je

$$s_0(\overline{x_k}) = \sum_{i=1}^{n} k_0(x_i, \overline{x_k}) f(x_i), \quad \forall \overline{x_k} \in \overline{\Gamma}$$

$$s_l(\overline{x_k}) = \sum_{i=1}^{n} k_l(x_i, \overline{x_k}) d_l(x_i), \quad \forall \overline{x_k} \in \overline{\Gamma}, \quad l \geq 1.$$

U slučaju proširenja difuzijskog preslikavanja, ova se metoda proširenja provodi za svaku difuzijsku koordinatu zasebno. Broj potrebnih iteracija ovisi o glatkoći određene koordinate.

4 Primjena

Cilj nam je na digitalnoj slici detektirati dijelove koji odskaču od ostatka slike, tj. od *pozadine* slike, dakle koji su u određenom smislu anomalija u odnosu na ostatak. Pokazat ćemo na koji se način takav problem svodi na problem klasteriranja podataka velike dimenzije koristeći ulaganje tih podataka u prostor puno manje dimenzije.

4.1 Podaci

Ulaz našeg algoritma je digitalna slika nekih dimenzija u pikselima, slikovni objekt, objekt tipa *Image*. Mi ćemo promatrati 'grayscale' digitalne slike. Takva je slika učitana u MATLAB funkcijom **imread(...)** pravokutna matrica cjelobrojnih elemenata iz intervala $[0, 255]$, čije vrijednosti reprezentiraju intenzitet sive boje odgovarajućeg piksela slike. Takvu ćemo matricu najprije normalizirati, da bismo kao sliku promatrali matricu realnih brojeva iz intervala $[0, 1]$. Podaci među kojima ćemo pokušati detektirati one koji su anomalija, odnosno klasterirati ih u klaster zaseban od ostalih podataka, su pojedinačni pikseli slike, tj. elementi matrice.

Difuzijsko preslikavanje nećemo računati na originalnim podacima, tj. na pojedinačnim pikselima, odnosno elementima matrice, već na skupu kvadratnih blok-matrica (podmatrica matrice

koja predstavlja sliku) odabrane dimenzije. Naime, u mnogim se problemima analize digitalnih slika ne promatraju izravno pikseli, već kvadratne podmatrice, koje se u engleskoj literaturi nazivaju *patchevi*. Svakom se pikselu pridruži jedinstveni kvadratni *patch*, i to takav da se piksel nalazi unutar pridruženog mu *patcha*. Primijetimo da ćemo ovime vrijednost difuzijskog preslikavanja *patcha* moći pridružiti odgovarajućem pikselu (dakle promatrati kompoziciju preslikavanja), pa ukoliko ju shvatimo kao vrijednost difuzijskog preslikavanja piksela, ona neće ovisiti samo o tom pikselu, već će biti izračunata iz cijele njegove kvadratne okoline, dakle sadržavat će u sebi sve te informacije. Ovo je očito vrlo pogodno za difuzijska preslikavanja i detekciju anomalije.

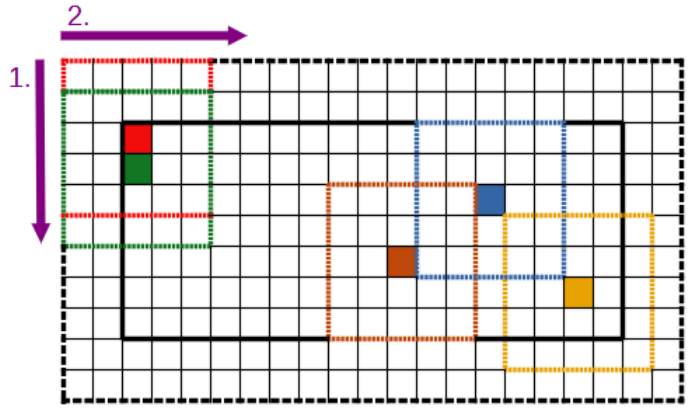
Dimenziju pojedinog bloka postavit ćemo kao parametar **patchDim**. Takve ćemo podmatrice vektorizirati, dakle primijeniti na njima tzv. *flattening*. Time će podaci na kojima računamo difuzijsko preslikavanje biti vektori realnih vrijednosti duljine koja je kvadrat dimenzije bloka, dakle $\text{patchDim}^2 = m$, dakle vektori iz prostora $\mathbb{R}^m$, dakle stvarno podaci "velike dimenzije".

Bilo bi dobro da su vrijednosti difuzijskog preslikavanja pridružene različitim pikselima različite, dakle da je preslikavanje injekcija (bijekcija). Stoga bi trebalo, na neki način, svakom pikselu, tj. elementu matrice, pridružiti drugi *patch*. To se može postići tako da se svakom pikselu pridruži kvadratni blok dimenzije patchDim čije je on središte (centar). Jedan drugi mogući način je da se svakom pikselu pridruži kvadratni blok dimenzije patchDim čiji je on gornji lijevi vrh.

Mi ćemo se odlučiti za *patcheve* centrirane u pikselima. Budući da takvi blokovi nisu definirani, tj. nije ih moguće izdvojiti iz matrice, za sve elemente koji su od nekog ruba matrice udaljeni za manje od patchDim, umjesto matrice koja predstavlja sliku, promatrat ćemo matricu nastalu njezinim proširivanjem za $\lfloor \text{patchDim}/2 \rfloor$ elemenata od svakog ruba prema van. Proširivanje znači da će elementi unutar tog okvira imati jednake vrijednosti kao elementi početne matrice, a za elemente okvira definirat ćemo neke vrijednosti. Za ovakve pothvate postoji MATLAB funkcija **padarray(...)**, koja kao argumente uzima originalnu matricu, duljine proširenja matrice izvan donjeg i gornjeg ruba te izvan lijevog i desnog ruba, te način proširivanja, tj. definiranja vrijednosti elemenata "okvira". Za posljednje postoje tri opcije: 'constant', koja sve elemente okvira postavlja na nulu, 'replicate', koja elementima "okvira" pridružuje vrijednosti koje se nalaze na rubovima početne matrice, te 'symmetric', koja na okvir zrcali dovoljno rubu najbližih elemenata originalne matrice. Za našu se primjenu najprirodnija čini treća opcija. Ovime ćemo svakom pikselu pridružiti podatak iz istog prostora $\mathbb{R}^m$, $m = \text{patchDim}^2$. Slično bi se moralo proširivati u slučaju da piksele promatramo kao gornje lijeve vrhove pripadnog *patcha*, ili bi se, eventualno, pikseli za koje pripadni *patchevi* ne postoje unutar slike morali izostaviti iz analize. Broj promatranih podataka tada bi bio manji od visina slike $\times$ širina slike.

Skup ovakvih podataka iz naše ćemo (proširene) matrice dobiti izdvajanjem svih mogućih (do na piksel razlike) preklapajućih kvadratnih blokova dimenzije patchDim, na način da prolazimo svim elementima originalne matrice najprije po stupcima pa po retcima te uzimamo blok dimenzije patchDim za kojeg je promatrani element središte. Primijetimo sada da prolazeći po svim indeksima redaka i stupaca originalne matrice, dakle prolazeći središtima podblokova, prolazimo po indeksima redaka i stupaca proširene matrice kao gornjim lijevim vrhovima odgovarajućih podblokova, pa možemo istovremeno izdvajati odgovarajuće blokove podataka iz proširene matrice i koordinate njihovih središta unutar originalne matrice. Svaki od ovih blokova vektoriziramo (standardna MATLAB vektorizacija također je u smjeru stupaca pa redaka), i opisanim ih redoslijedom spremamo kao retke ili stupce jedne matrice. Radi kasnije upotrebe MATLAB funkcija koje uključuju računanje udaljenosti među vektorima, a djeluju na vektorima-retcima, poput **pdist2(...)** ili **knnsearch(...)**, mi ćemo *patcheve* spremati kao retke matrice. Širina te matrice je, dakle, duljina vektora-podataka, dakle patchDim^2, a visina je ukupan broj mogućih

ovakvih podblokova matrice, dakle broj elemenata originalne matrice. Prilikom ovog postupka, kreiramo također i matricu koordinata središta svakog podbloka, na način da odgovarajućim redoslijedom koordinate spremamo u retke te matrice. Ovime pamtimo koji podaci odgovaraju kojem središtu, tj. pamtimo gdje se određeni podblok nalazi u početnoj (odnosno proširenoj) matrici.



4.2 Računanje difuzijskog preslikavanja

4.2.1 Restrikcija skupa podataka

Kako bismo smanjili složenost računanja, difuzijsko preslikavanje nećemo računati na cijelom skupu podataka, dakle na svim mogućim podblokovima matrice, odnosno dobivenim vektorima. Umjesto toga, definirat ćemo neki postotak, tj. udio, ukupnog broja podataka za koje ćemo vrijednosti difuzijskog preslikavanja egzaktno izračunati, a zatim na slučajan način izdvojiti toliko podataka iz skupa. To ćemo implementirati tako da odabrani postotak ukupnog broja podataka zaokružimo na cijeli broj, a zatim na slučajan način permutiramo indekse redaka matrice s podacima, te odaberemo odgovarajući broj početnih elemenata permutacije. U MATLAB-u to radimo s pomoću metoda **round(...)** i **randperm(...)**. Iz matrice svih podataka izdvojit ćemo odgovarajuće retke u novu matricu koja će predstavljati skup podataka na kojem računamo difuzijsko preslikavanje.

4.2.2 Računanje matrice sličnosti za restrikciju

Nadalje, (samo) za odabrani skup podataka računamo matricu sličnosti, W. Matricu ćemo računati kako je opisano u teorijskom uvodu. Međutim, iako je skup podataka smanjen u odnosu na početni, matrica W će biti jako velike dimenzije, pa će složenost računanja biti velika. Stoga, težine, tj. međusobne mjere sličnosti, nećemo računati za sve promatrane podatke. Za svaki podatak iz skupa najprije ćemo pronaći neki unaprijed odabrani broj **kNN** njegovih najbližih susjeda, te njegove sličnosti s njima izračunati po ranije opisanoj formuli. Naime, opisano preslikavanje

$$\sigma_i = ||x_i - x_k||_2^2,$$

gdje je x_i podatak, a x_k njegov k-ti najbliži susjed, prikladno je upravo za naš problem klasteriranja, detekciju anomalije na slikama, u kojem podatke koji su anomalija pretpostavljamo (jako) udaljenim od svih ostalih podataka u skupu, koji pripadaju normalnoj *pozadini* na slici. Budući da je težinska funkcija simetrična, vrijednost težine za neki podatak i neki njegov najbliži susjed automatski će definirati i vrijednost težine za taj susjedni podatak i podatak kojeg promatramo, tj. i taj element matrice sličnosti. Dijagonala matrice sličnosti mora biti jedinična. Sve preostale

ovim postupkom neobuhvaćene elemente matrice sličnosti, tj. mjere sličnosti dvaju podataka, postavit ćemo na 0.

Baš ovakva prilagodba računanja matrice sličnosti u svrhu smanjenja složenosti računanja, dakle takva da se za podskup podataka čije će sličnosti ili težine biti netrivijalne uzmu baš najbliži susjedi, česta je i standardna pri praktičnim primjenama. To očito ima smisla. Naime, najbliži susjedi najmanje su udaljeni od podatka, pa je njihova sličnost s podatkom najveća, a takve težine želimo uzeti u obzir. Nadalje, udaljenosti podatka od svih ostalih podataka sve su veće, odnosno sličnosti su sve manje, pa ih možemo zanemariti, tj. smatrati bliskima, odnosno jednakima, nuli. Za bliskog susjeda podatka u definiciji preslikavanja σ očito ima smisla da bude među **kNN** najbližih susjeda za koje težine smatramo netrivijalnima, jer, ukoliko bi susjed za σ bio onaj za kojeg pri računanju matrice sličnosti smatramo da je njegova sličnost s podatkom bliska ili jednaka nuli, značilo bi da mu je taj susjed jako daleko, pa njihova udaljenost, tj. odgovarajuća vrijednost σ, ne bi dobro informirala o lokalnoj geometriji. Stoga, za indeks **k** najbližeg susjeda za preslikavanje σ očito ima smisla da bude (dosta) manji od **kNN**. Konačno, za dovoljno velik parametar **kNN**, ovakva definicija težinske funkcije i dalje će dovoljno dobro oslikavati i globalnu i lokalnu geometriju podataka.

Računanje vrijednosti samo manjeg skupa elemenata matrice ne smanjuje samo potrebnu količinu računanja pri kreiranju matrice. Matrice kojima je većina elemenata jednaka nuli takozvane su *rijetke* matrice (eng. *sparse*). U MATLAB-u je rijetku matricu poželjno alocirati funkcijom **sparse(...)**, jer ona na vrlo prostorno štedljiv način reprezentira rijetku matricu u memoriji računala, a također takav da je daljnja upotreba MATLAB-funkcija na takvoj strukturi također računalno efikasnija.

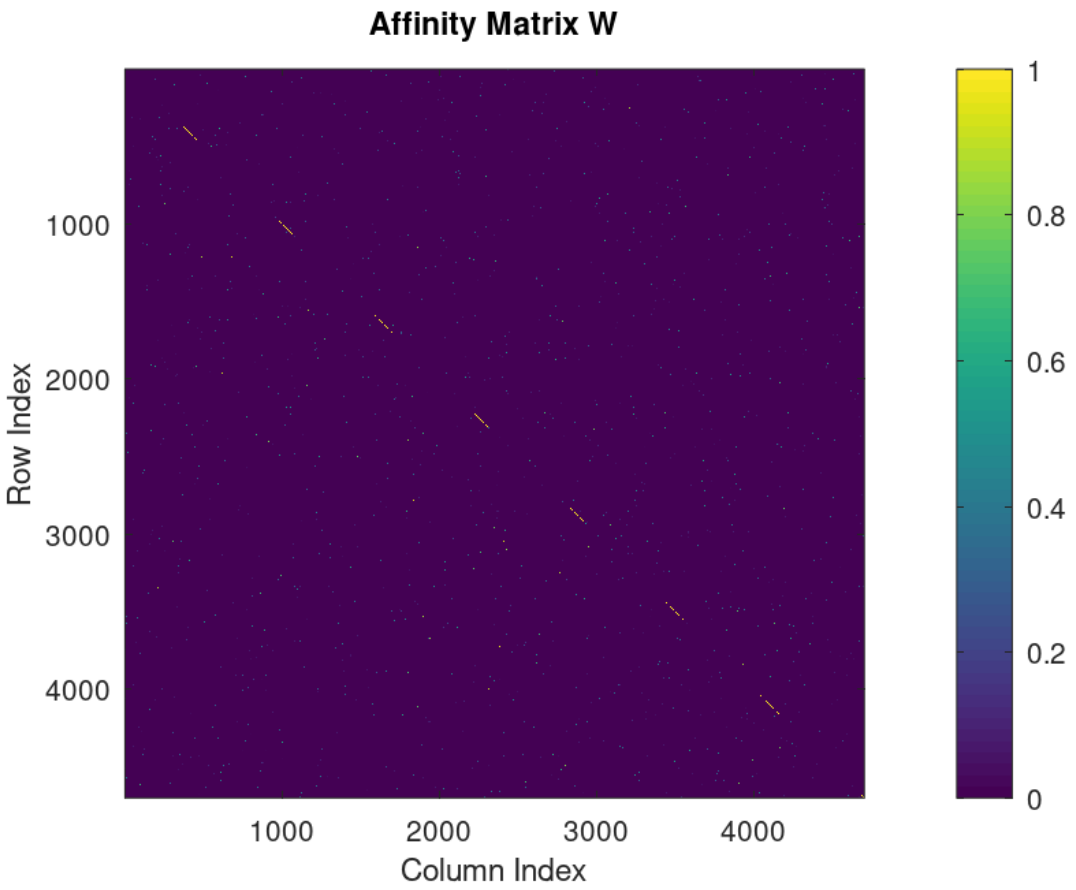
Najprije, dakle, određujemo i postavljamo parametar kNN te tražimo kNN najbližih susjeda za svaki podatak iz skupa. Za ovo u MATLAB-u postoji efikasna funkcija **knnsearch(...)**, koja vraća matricu uzlazno sortiranih udaljenosti svakog podatka do njegovih kNN najbližih susjeda te matricu indeksa pripadnih susjeda u skupu podataka. U istu svrhu i za ekvivalentan rezultat možemo upotrijebiti i funkciju **pdist2(...)**, koja inače služi za računanje (svih) međusobnih udaljenosti podataka iz dvaju skupova, tako da je pozovemo sa sljedećim argumentima: [Dist, Idx] = pdist2(X, X, 'euclidean', 'Smallest', kNN). Za razliku od **knnsearch(...)**, ona je dostupna i u programu Octave. Ove će nam funkcije, dakle, osim pronalaženja najbližih susjeda, automatski i izračunati pripadne udaljenosti, koje su nam potrebne za računanje vrijednosti elemenata matrice sličnosti. Također, budući da ćemo, kao što je rečeno gore, osigurati da je bliski susjed za vrijednost preslikavanja σ među kNN najbližih susjeda podatka, bit će automatski izračunate i pohranjene i vrijednosti preslikavanja σ za svaki podatak iz skupa. Funkcija **knnsearch(...)** svaki podatak smatra svojim najbližim susjedom, što i jest, a funkcija **pdist2(...)** s gornjim argumentima također stavlja udaljenost jednaku nuli i vlastiti indeks na početno mjesto lista najmanjih udaljenosti i indeksa pripadnih podataka. Stoga, kada se u primjenama traži k najbližih susjeda, uglavnom se podatak ne smatra svojim susjedom, pa se ove funkcije, umjesto s parametrom k, pozivaju s parametrom $k+1$, a iz povratnih se matrica izostavlja prvi stupac. U našem ćemo kodu, umjesto s kNN, funkciju pozvati s kNN+1, a, budući da želimo da je sličnost svakog podatka sa samim sobom jednaka 1, dakle da svakako nije zanemarena u matrici W, zadržat ćemo podatke o nul-udaljenostima, pa će se u nastavku algoritma pripadne sličnosti ispravno izračunati kao jedinice.

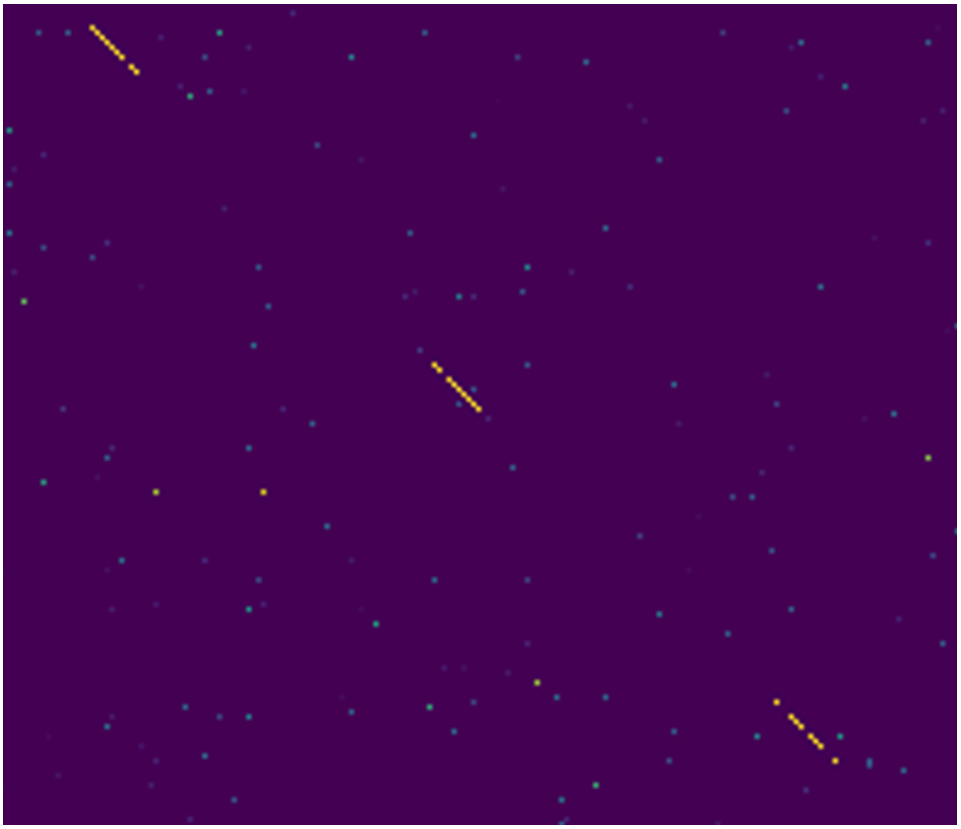
Za konstrukciju rijetke matrice funkcijom **sparse(...)** dovoljno je zadati željene dimenzije matrice te konstruirati tri vektora koji će sadržavati potrebne informacije o elementima matrice koji nisu 0: vektor indeksa redaka, vektor indeksa stupaca te vektor pripadnih vrijednosti. Očito će broj netrivijalnih elemenata u našoj matrici, dakle duljina tih triju vektora, biti jednak broju

podataka koje promatramo pomnoženom s kNN+1. To jest, to je broj vrijednosti koje ćemo egzaktno računati, a budući da matrica mora biti simetrična, na kraju ćemo simetrizacijom dobivene rijetke matrice dodati još netrivijalnih vrijednosti, jer ukoliko je, na primjer, neki podatak među kNN najbližih susjeda nekog drugog podatka, ne znači da je taj drugi podatak među njegovih kNN najbližih susjeda. Vektore ćemo popuniti na efikasan način, koristeći strukturu matrica $Dist$ i Idx. Budući da su informacije za odgovarajuće podatke u matricama $Dist$ i Idx spremljene u istom redoslijedu u kojem su podaci spremljeni u ulaznom skupu podataka, te da će u matricama $Dist$ i Idx informacije za odgovarajuće podatke biti spremljene po retcima, tri spomenuta vektora popunjavat ćemo u segmentima, pridruživanjem redaka matrica $Dist$ i Idx, koji su duljine kNN +1, redom po podacima, postavljajući u vektor indeksa redaka indeks odgovarajućeg podatka, dakle indeks retka matrice $Dist$ ili Idx. Elementi matrice Idx bit će indeksi stupaca, a elementi matrice $Dist$, dakle udaljenosti između podataka, poslužit će nam za računanje vrijednosti pripadnih težina, tj. mjera sličnosti, po ranije opisanoj formuli. Sve vrijednosti preslikavanja σ očito su pohranjene u stupcu matrice $Dist$ s indeksom jednakom indeksu odabranog bliskog susjeda, kojeg smo u našem programu nazvali **selfTuneNNIndex**. Taj je stupac praktično izdvojiti u zaseban vektor, na primjer **sigma**. Kako bi se izbjeglo da nazivnik u formuli za težinu, dakle produkt $\sigma_i\sigma_j$, bude jako blizu nuli, u MATLAB-u ćemo ga korigirati dodavanjem faktora **eps**.

Rijetku kvadratnu matricu dobivenu funkcijom **sparse(...)**, pozvanom s tri konstruirana vektora i dimenzijom jednakom broju podataka u skupu, na kraju ćemo simetrizirati. Ukoliko se pri korištenju funkcija **knnsearch(...)** ili **pdist2(...)** podatak ne smatra svojim vlastitim susjedom, tj. ukoliko se iz povratnih matrica $Dist$ i Idx izuzme prvi stupac, dobivena matrica W neće imati jediničnu dijagonalu, pa bi to trebalo postići korištenjem MATLAB-funkcije **spdiags(ones(...), 0, W)**, čiji je rezultat ponovno rijetka (***sp****arse*) matrica.

Strukturu dobivene matrice možemo na razne načine testirati, pišući prikladne testne programe, ispisujući manje blokove matrice u konzolu ili vizualizirajući matricu. Funkcija **imagesc(...)** vizualizira matricu prikazujući elemente matrice obojene s obzirom na njihove vrijednosti, pri čemu je boja elementa toplija što je vrijednost elementa veća.





Gornje slike prikazuju dobivenu matricu sličnosti za odabrani podskup podataka gdje je druga

slika uvećana dijagonala matrice. Vidimo da je većina matrice obojena istom najhladnijom bojom, iz čega zaključujemo da je većina elemenata u matrici stvarno jednaka nuli, odnosno matrica je rijetka. Nadalje, uočavamo da su elementi dijagonale matrice obojeni istom najtoplijom bojom, pa matrica sličnosti stvarno ima jediničnu dijagonalu. Također, vidimo da je matrica obojena simetrično, dakle stvarno je simetrična.

Podaci u originalnom skupu podataka poredani su možda redoslijedom koji nije onaj koji grupira slične podatke, a budući da smo podatke za podskup čija je ovo matrica sličnosti odabrali nasumično iz originalnog skupa, slični podaci su raspršeni po matrici, a nisu klasterirani zajedno.

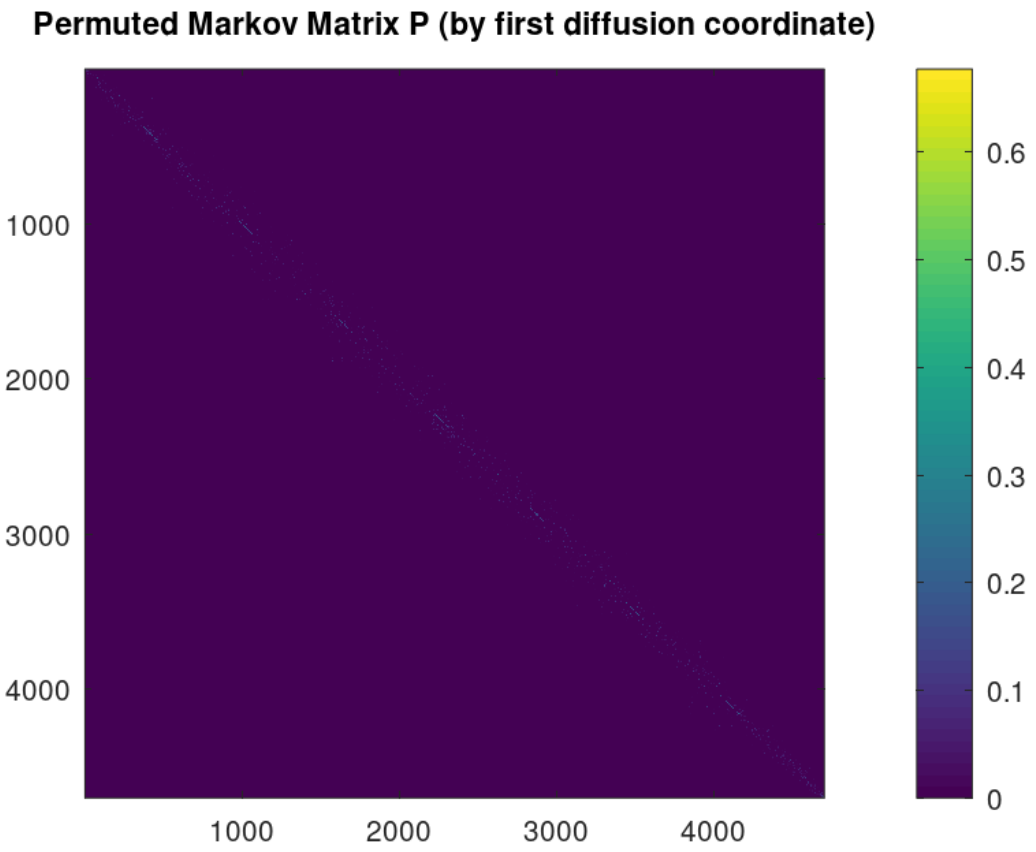
4.2.3 Računanje restrikcije difuzijskog preslikavanja

Nakon što smo izračunali matricu sličnosti, možemo izračunati matricu

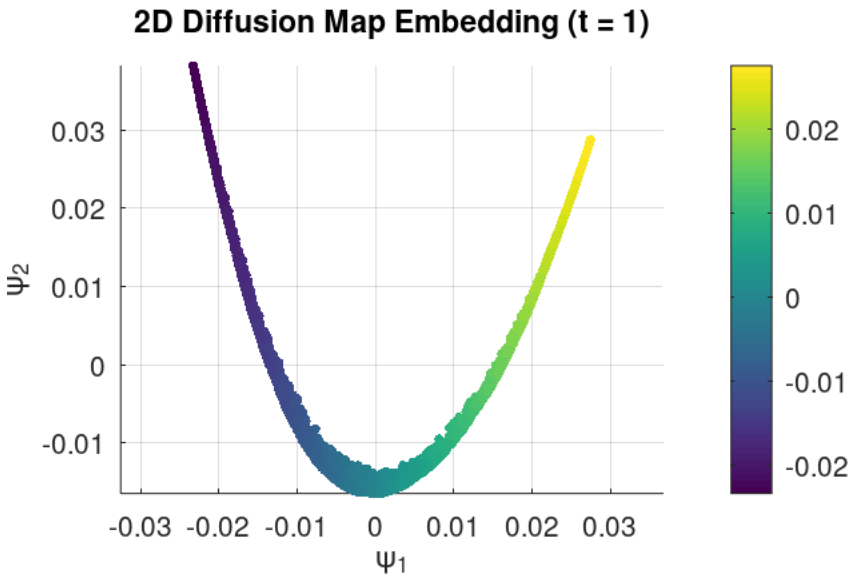
$$P_s = D^{-1/2} W D^{-1/2},$$

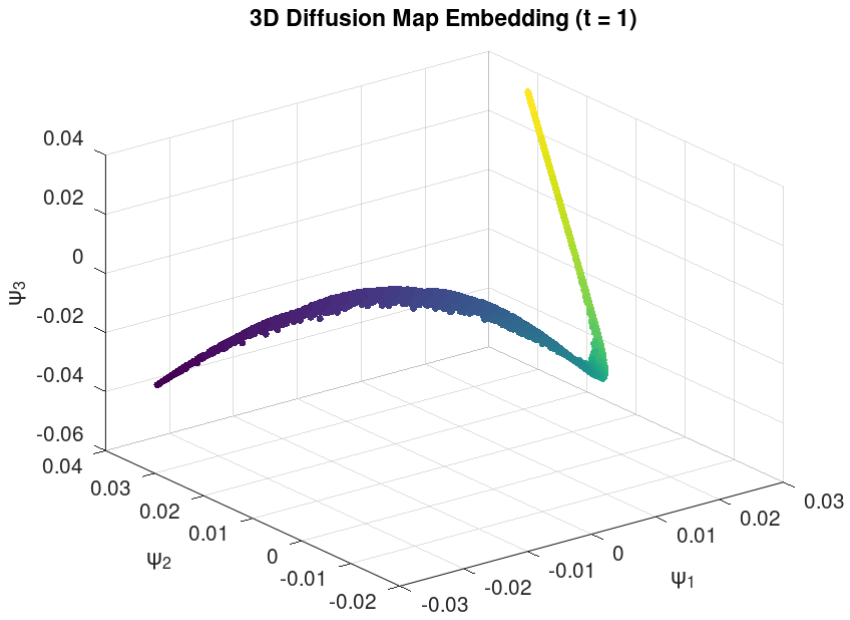
čiju spektralnu dekompoziciju koristimo za definiciju difuzijskog preslikavanja, gdje je W dobivena matrica sličnosti. Matrica $D^{-1/2}$ očito se može dobiti računanjem suma svih redaka matrice W te kreiranjem dijagonalne matrice iz vektora čiji su elementi redom dobivene sume, ali na koje je primijenjen drugi korijen, te su invertirane. To možemo napraviti MATLAB-funkcijom **spdiags(...)**. Spektralnu dekompoziciju matrice možemo izračunati efikasnom MATLAB-funkcijom **eigs(..)**, takvom (s argumentom 'LM' - *largest magnitude*) koja vraća podskup vodećih svojstvenih vektora i svojstvenih vrijednosti. Ona kao argument prima željeni broj svojstvenih parova. Željenu dimenziju potprostora manje dimenzije u koji želimo uložiti podatke, dakle broj vodećih svojstvenih vektora podataka koje uzimamo u obzir, također želimo na neki način kontrolirati. Zbog toga ćemo tu dimenziju definirati kao parametar **numEigs**. Kako podatke želimo uložiti u potprostor neke puno manje dimenzije, poželjno ga je definirati kao minimum dimenzije originalnih podataka i nekog broja kojeg smatramo zadovoljavajuće malim, npr. 7, ili npr. 10. Budući da metoda **eigs(..)** za matricu P_s vraća matricu U vodećih lijevih svojstvenih vektora matrice P_s posloženih redom po stupcima, te dijagonalnu matricu Λ s vodećim svojstvenim vrijednostima redom na dijagonali, *numEigs* vodećih lijevih svojstvenih vektora matrice P možemo dobiti redom kao stupce matrice $\Phi = D^{-1/2}U$, a vrijednosti svih koordinata difuzijskog preslikavanja za cijeli promatrani skup podataka možemo dobiti skaliranjem svakog stupca matrice svojstvenih vektora Φ odgovarajućom svojstvenom vrijednosti.

Na sljedećim je slikama vizualizarana matrica P te njena uvećana dijagonala za promatrani poskup podataka permutirana na temelju vrijednosti prve koordinate izračunatog difuzijskog preslikavanja.



Na sljedećim su slikama vizualizirane prve dvije te prve tri koordinate izračunatog difuzijskog preslikavanja.





4.2.4 Proširenje difuzijskog preslikavanja na cijeli skup

Premda smo difuzijsko preslikavanje izračunali samo na podskupu svih podataka, sada ćemo to preslikavanje proširiti i na sve preostale originalne podatke, i to koristeći prethodno opisanu *Laplaceovu piramidu*, koju ćemo implementirati u MATLAB-u izravno kako je opisano, postavljajući parametre. Primijetimo da u svakoj iteraciji moramo računati matricu sličnosti između podataka u podskupu, jer se u svakoj iteraciji koristi drugi parametar σ, pa je korisno unaprijed izračunati euklidske udaljenosti među podacima u podskupu, pohraniti ih u matricu, a zatim s pomoću nje računati matricu sličnosti u svakoj iteraciji. Budući da smo matricu sličnosti upravo za ovaj podskup već računali, mogli smo već tada pohraniti međusobne euklidske udaljenosti podataka. Također, u svakoj iteraciji moramo računati matricu sličnosti između podataka u podskupu i podataka u ostatku skupa, na koje proširujemo difuzijske koordinate, pa je, analogno, korisno unaprijed, dakle prije početka provođenja iteracija, izračunati i pohraniti

euklidske udaljenosti među podacima u podskupu i ostatku skupa podataka. Konačno, i ovdje ćemo sve matrice sličnosti kreirati kao rijetke matrice, računajući egzaktne sličnosti samo za kNN najbližih susjeda svakog podatka, a sve ostale sličnosti smatrat ćemo zanemarivima, tj. postaviti ih na nulu. Naravno, matrice euklidskih udaljenosti također se kreiraju kao rijetke, računajući egzaktne vrijednosti udaljenosti do samo kNN najbližih susjeda, jer se samo za te i točno te parove egzaktno računaju sličnosti.

4.3 Primjena difuzijskog preslikavanja - *anomaly score*

Svakom smo pikselu ulazne digitalne slike, dakle, prethodnim postupkom, pridružili vrijednost difuzijskog preslikavanja. Pritom, vrijednost difuzijskog preslikavanja za pojedini piksel dobivena je na temelju kvadratnog *patcha* digitalne slike (ili njezine *padded* verzije), čiji je taj piksel središte, dakle nastala je na temelju informacije o cijeloj kvadaratnoj okolini piksela.

Sada izračunato difuzijsko preslikavanje, dakle njegove vrijednosti za svaki piksel, želimo iskoristiti za detekciju piksela ulazne digitalne slike koji su *anomalija*.

To ćemo napraviti tako da ćemo za svaki piksel ulazne slike računati *anomaly score*, te ćemo piksele za koje je *anomaly score* viši od određenog praga smatrati *anomalijom*, a ostale ćemo smatrati *normalnima*.

Za svaki piksel slike računat ćemo njegove sličnosti s pikselima unutar njegove kvadratne okoline na slici, veličine jednake za svaki piksel, definirane Gaussovom jezgrom s euklidskom udaljenošću pripadnih vrijednosti difuzijskog preslikavanja, jer je ona, dakle, jednaka difuzijskoj udaljenosti tih piskela, odnosno pripadnih *patcheva*. Faktor σ Gaussove jezgre postavit ćemo kao varijancu euklidskih udaljenosti vrijednosti difuzijskog preslikavanja, dakle difuzijskih udaljenosti, nasumično odabranog skupa parova piksela.

Ukupnu sličnost piksela s pikselima unutar njegove kvadratne okoline na slici odredit ćemo kao srednju vrijednost (aritmetičku sredinu) sličnosti s pikselima u toj okolini.

Budući da za *anomaly score* piksela želimo da je veći što je sličnost tog piskela s okolinom manja, a manji što je sličnost s okolinom veća, za pojedini piksel, tj. podatak x_i, definirat ćemo ga sljedećom formulom:

$$C(i) = 1 - \frac{1}{m} \sum_{j \in N_i} \bar{w}(i,j),$$

gdje je

$$\overline{w}(i,j) = e^{\frac{-||\Psi(x_i) - \Psi(x_j)||^2}{\overline{\sigma}}}$$

i

$$\overline{\sigma} = r\sigma^2_{pair},$$

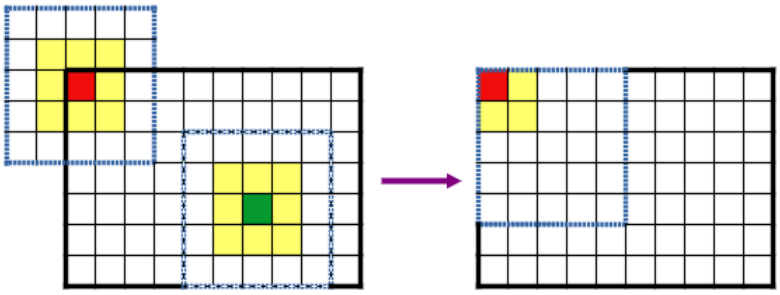
npr. $r = 20$, te je m unaprijed određeni željeni broj promatranih piksela u okolini piksela N_i koji odgovara podatku x_i.

Dakle, visoka vrijednost $C(i)$ (tj. vrijednost blizu 1) ukazuje na malu sličnost piksela koji odgovara x_i s okolinom u difuzijskom prostoru, tj. na potencijalnu *anomaliju*.

Također, unutarnji dio prozora oko piksela x_i se *maskira* kako bi se izbjegla usporedba piksela s neposrednim susjedima, pod pretpostavkom da je anomalija veća od jednog piksela, pa su onda i pikseli koji su *anomalija* jako slični svojim najbližim okolnim susjedima. Veličina prozora

N_i i maske M ovise o očekivanoj veličini anomalije.

S obzirom na položaj piksela digitalne slike u odnosu na rubove slike, određivanje njegove kvadratne okoline određene veličine, te maske određene veličine, provodi se kako je vizualizirano na sljedećoj slici.



5 *Višerezolucijski* pristup

Opisani algoritam primjenjuje se na jednu matricu koja predstavlja originalnu sliku. Međutim, u obradi slika poznat je takozvani višerezolucijski pristup, koji koristi takozvanu **Gaussovu piramidu** slike - familiju slika $\{G_l\}_{l=0}^{L}$, gdje je G_0 originalna slika, a svaka sljedeća slika dobivena je pogoršavanjem rezolucije prethodno dobivene slike. Slika G_L je slika najlošije rezolucije. Opisani algoritam odvija se iterativno po razinama. Kreće od razine najgrublje rezolucije - L:

1. Na razini l (počevši od L), odabire se podskup podataka Γ_l. Za razinu najgrublje rezoplucije L, Γ_L se bira nasumično (možda i svi podaci ako je G_L dovoljno mala). Za razine finije rezolucije ($l < L$), Γ_l obavezno uključuje podatke koji odgovaraju *sumnjivim* podacima s prethodne (grublje) razine $l + 1$, dok se ostatak Γ_l popunjava nasumičnim podacima.

2. Izračuna se difuzijsko preslikavanje za podskup Γ_l i proširi se na sve podatke u G_l.

3. Izračuna se stupanj anomalije $C_l(i)$ za svaki podatak x_i na razini l.

4. Podaci čija ocjena $C_l(i)$ prelazi određeni prag (eng. *treshold*)τ_l (npr. 95-ti percentil svih procjena na toj razini) označavaju se kao *sumnjivi*.

5. Ako je $l > 0$, prijeđi na sljedeću (finiju) razinu $l - 1$, koristeći informacije o sumnjivim podacima za bazu uzorkovanja u koraku 1.

6. Ako je $l = 0$ (originalna rezolucija), konačno pridruživanje stupnjeva anomalije C_0 koristi se za detekciju. Može se primijeniti konačni prag τ i prostorno filtriranje/zaglađivanje.

Ovakav proces osigurava da se informacije o potencijalnim anomalijama prenose s grubljih na finije razine, čime se povećava vjerojatnost da će anomalija biti prikladno zastupljena u uzorku na finijim razinama, čak i ako je bila slabo vidljiva ili slabo zastupljena u uzorcima na grubljim. Postavljanje nižeg praga τ_l na grubljim razinama je prihvatljivo jer konačna odluka pada tek na razini $l = 0$, čime se smanjuje rizik od pogreške na grubim razinama bez nužnog povećanja konačne stope *lažnih uzbuna*. Visok stupanj na razini l vodi do gušćeg uzorkovanja tih područja na razini $l - 1$, što rezultira boljim odvajanjem anomalije na finijoj razini. Pristup je također računski efikasniji od nekih jednorezolucijskih metoda jer se na grubljim razinama koriste manje značajke (manji isječci).

Literatura

[1] G. Mishne and I. Cohen, "Multiscale Anomaly Detection Using Diffusion Maps," *IEEE Journal of Selected Topics in Signal Processing*, vol. 7, no. 1, pp. 111–121, Feb. 2013.

[2] [Anonymous Authors or Authors as cited in source [23]], "Detection of anomaly trends in dynamically evolving systems," in *Proc. AAAI Fall Symposium Series*, 2010.

[3] The MathWorks, Inc., *MATLAB Documentation*. Natick, Massachusetts, 2025. Available at: https://www.mathworks.com/help/matlab/

[4] G. Mishne and I. Cohen, "Code for Multiscale Anomaly Detection Using Diffusion Maps," 2018. Available at: https://israelcohen.com/wp-content/uploads/2018/05/multiscale_dm_ano_detect.zip

[5] R. R. Coifman and S. Lafon, "Diffusion maps," *Applied and Computational Harmonic Analysis*, vol. 21, no. 1, pp. 5–30, 2006.

[6] B. Nadler, S. Lafon, R. R. Coifman, and I. G. Kevrekidis, "Diffusion maps, spectral clustering and reaction coordinates of dynamical systems," *Applied and Computational Harmonic Analysis*, vol. 21, no. 1, pp. 113–127, 2006.

[7] Z. Drmač, *Difuzijska preslikavanja* (Prezentacija s predavanja). 2025.