Универзитет у Београду
Електротехнички факултет

Компаративна анализа линеарних и нелинеарних метода за редукцију димензионалности

Ментор
доцент др Предраг Тадић

Студент
Драгана Нинковић 2023/3010

9. јул 2024.

Садржај

1 Абстракт

У данашње време наука о подацима постаје све више примењива и све више коришћена у многим областима попут финансија, медицине, инжењеринга... Иако се дубоко учење показало јако ефикасним за решавање доста проблема, оно има једну значајнуману у поређењу са традиционалним алгоритмима машинског учења, а то је интерпретабилност резултата. На пример, иако је докторима за неку болест довољно само пар обележја да је дијагностикују, наш модел као улаз добија пар стотина хиљада обележја у случају слика. За доста проблема није довољно само решење већ и објашњење како је модел дошао до закључка којим се решава проблем. Такође, када број података потрбних за решавање модела постане превелик долази до такозваног "проклетсва димензиналности" услед кога подаци у простору постају прорећени, дистанце између података носе недовољно информација, и модел захтева превише параметара, што његово тренирање чини превише компјутерски комплексним. Из свих наведених разлога, методи претпроцесирања података, а посебно редукција димензија су јако битни за ефикасност и рад самог модела.

2 Увод

Задатак сваког метода за редукцију димензија је да од улазних података великих димензија добије податке мањих димензија који имају очуван контекст података на улазу и када је то могуће, обезбеђује њихово графичко представљање како би повезаности између података биле интуитивно видљиве пре уласка у модел.
Најпознатији и најкоришћенији методи за најосновнију редукцију димензија су PCA (Анализа главних компоненти) и LDA (Линеарна дискриминантна анализа).
Линеарна дискриминантна анализа редукује димензионалност података тако што максимизује растојање између података који припадају различитим класама и минимизује растојање између података који припадају истој класи.Да би се LDA интерпретирао као логаритам односа апостериорних вероватноћа, претпоставља се да су подаци нормално распоређени унутар сваке класе. У случајевима када не знамо расподелу података и када их не можемо довољно добро раздвојити линеарном границом, израз постаје доста сложенији и овај метод постаје компјутерски неефикасан.
Анализа главних компоненти издваја независне компоненте трансформацијом података у нормалне компоненте при чему максимизује варијансу, ипак, њен велики недостатак је то што ничим не гарантује очување сепарабилности података у новом простору. Овај метод је такође линеаран, али постоји надоградња на њега коришћењем кернела која решава проблем нелинеарности података.
Наравно, постоји још много метода које се користе за редукцију димензија које су засноване на разним другим идејама попут очувања самих растојања, локалне геометрије, коришћење теорије графова и слично. Ипак, већина метода пре стохастичког угњеждавања суседа коришћењем т-расподеле користила је мапирање једног оригиналног податка у искључиво један трансформисан податак, док овај метод први пут уводи мапирање података један према више. Такође, већина до тада нађених техника није била употребива над правим великим скуповима података или не би очувала и глобалну и локалну структуру података. Рад који уводи главну идеју овог метода је **2002. G. Hinton и S. Roweis**, *Stochastic Neighbor Embedding* и он решава дати проблем посматрајући расподелу сличности између тачака у простору. За сваку тачку посматра расподелу сличности између те тачке и осталих тачака коју моделира Гаусовом расподелом. Слична идеја се користи и у трансформисаном простору, након чега се пореде ове две расподеле и тежи се томе да расподела која одговара тачки у оригиналном простору буде што сличнија расподели која одговара тачки у трансформисаном простору. За меру сличности између расподела користи се КЛ-дивергенција а нов простор се добија минимизацијом КЛ-дивергенције коришћењем метода градијентног спуста. Као што је већ речено, сваком објекту у оригиналном простору се може придружити не само једна већ и више слика новог нискодимензионог простора које су међусобно удаљене. Ова особина је посебно корисна за примену код обраде језика, јер онда у случају више речи које се исто пишу, не морамо да форсирамо сличност вокабулара два различита контекста, већ их можемо оставити удаљенима а оригиналној речи придружити пројекције из оба контекста.
Рад који допуњује ову идеју је **2002.G. Hinton и L. van der Maaten** , *Visualizing Data using t-SNE*. У овом раду, уместо Гаусове расподеле у трансформисаном простору се посматра студентова расподела. Ова промена доводи до знатно брже оптимизације и веће репрезентативности јер се кластери не гомилају у средини простора већ су више међусобно удаљени. Овај метод спада у групу метода који индиректно користе метод најближих суседа. Стохастичко угњеждавање суседа коришћењем т-расподеле се најчешће користи за визелизацију података.
UMAP (*Uniform Manifold Approximation and Projection*) је савремени метод за редукцију димензија који се истиче по својој способности да ефикасно сачува и локалну и глобалну структуру података. Рад који уводи и развија овај метод је **2018. L. McInnes, J. Healy и N. Melville**, *UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction*. Настао је са циљем да омогући бољу визуелизацију и анализу великих скупова података. Главна идеја *UMAP*-а је да конструише граф најближих суседа у високо-димензионалном простору и затим пројектује ове податке у ниско-димензионални простор на начин који минимизира разлике између ова два простора. За то користи параметризовани облик расподеле чији се параметри бирају тако да минимизују ове разлике. *UMAP* је познат по својој брзини и ефикасности у поређењу са другим методама редукције димензија, попут *t-SNE*. Он пружа могућност брзе пројекције и визуелизације, што га чини изузетно корисним алатом у анализи података.

Сви алгоритми који користе метод најближих суседа подразумевају следеће кораке приликом редукције димензија:

1. Конструише се граф к најближих суседа
2. Примњеује се нека трансформација на тежине грана која осликава локалну структуру података
3. Решава се проблем асиметричности графа
4. Дефинише се критеријумска функција која ће да осликава одступање од особине коју желимо да очувамо у подацима
5. Оптимизовањем критеријумске функције налази се представа у трансформисаном простору са мањим бројем димензија него подаци у почетном простору

Циљ овог рада је да објасни рад Анализе главних компоненти са коришћењем кернела, стохастичког угњеждавања суседа коришћењем т-расподеле и *UMAP* - а и да их међусобно упореди над скуповима података: MNIST, swiss-roll, S-curve.
Све имплементиране функционалности доступне су на линку.

3 Нелинеарна анализа главних компоненти коришћењем кернела

3.1 Увод

Анализа главних компоненти је најпознатији алгоритам редукције димензија који се користи у разним наукама које подразумевају примењену статистику укључујући социологију, психологију, медицину и инжењеринг. Нелинеарна анализа главних компоненти коришћењем кернела допуњује ову методу тако што користи трик коришћења кернела - идеју која се користи у класификацији помоћу потпорних вектора. Овај трик решава нелинеарност тако што податке пресликава у више димензије простора од оне у којој се он већ налази, тако да у новом простору они постану линеарно сепарабилни тј. граница која одваја податке буде нека хипер раван. Практичност овог приступа је то што често није потребно да знамо конкретну функцију којом се оригинални простор трансформише у вишедимензиони $\phi(x)$, већ је довољно да знамо функцију унутрашњег производа између две трансформације тачака који се назива кернел.

Ако посматрамо низ $(x_1, x_2, \ldots, x_n)$ N d-димензионалних тачака које чине улазни скуп података и претпоставимо да су средње вредности координата након трансформације једнаке нули, коваријациона матрица Σ у ϕ-простору је (1).

$$\Sigma = \frac{1}{N}\sum_{1}^{N}\phi(x_i)\phi(x_i)^T \tag{1}$$

Анализа главних компоненти сада се може применити директно у ϕ-простору, сопственом декомпозицијом матрице Σ. Сопствена декомпозиција своди се на налажење вектора v који испуњава $\Sigma v = \lambda v$.

3.2 Извођење

Иако се на први поглед чини да ово није могуће урадити без експлицитног рачунања Σ, испоставља се да то није неопходно ако се искористи следећа теорема:
Теорема: Сваки спствени вектор v може се представити као отежињена сума $\phi(x^{(i)})$ (2).

$$v = \sum_{i=1}^{N}\alpha_i\phi(x_i) \tag{2}$$

Ако се крене од тога да је $\Sigma v = \lambda v$ добија се (3)

$$\lambda v = \Sigma v = \frac{1}{N}\sum_{i=1}^{N}\sum_{1}^{N}\phi(x_i)\phi(x_i)^T v \tag{3}$$

Ако се у једначини (3) v замени са формулом из теореме (2) добија се формула (4).

$$\lambda\sum_{j=1}^{N}\alpha_j\phi(x_j) = \frac{1}{N}\sum_{i=1}^{N}\phi(x_i)\phi(x_i)^T\sum_{j=1}^{N}\alpha_j\phi(x_j) \tag{4}$$

Ако се мало боље групишу изрази у формули (4), добија се израз (5)

$$\lambda\sum_{j=1}^{N}\alpha_j\phi(x_j) = \frac{1}{N}\sum_{i=1}^{N}\phi(x_i)\sum_{j=1}^{N}(\phi(x_i)^T\phi(x_j)\alpha_j) \tag{5}$$

Након тога, потребно је помножити обе стране једначине са $\phi(x_k)^T$ да бисмо добили облик који се може представити само преко кернела. Тиме се добија једначина (6).

$$\lambda\sum_{j=1}^{N}\alpha_j\phi(x_k)^T\phi(x_j) = \frac{1}{N}\sum_{i=1}^{N}\phi(x_k)^T\phi(x_i)\sum_{j=1}^{N}(\phi(x_i)^T\phi(x_j)\alpha_j) \tag{6}$$

Ако се елемент грам матрице G са индексима i, j дефинише се као (7) формула (6) може се представити искључиво преко кернела као (8).

$$G_{i,j} = K(x_i, x_j) = \phi(x_i)^T \phi(x_j) \tag{7}$$

$$\lambda \sum_{j=1}^{N} G_{k,j}\alpha_j = \frac{1}{N} \sum_{i=1}^{N} G_{k,i} \sum_{j=1}^{N} \alpha_j G_{i,j} \tag{8}$$

Ако се мало детаљније посматра добијени израз, може се уочити да сума са леве стране представља производ грам матрице G и вектора α, док суме са десне стране представљају резултат производа између грам матрице G и производа вектора α и помово грам матрице. Овиме, формула (8) се може приказати у векторском облику (9)

$$\lambda G\alpha = \frac{1}{N} G^2 \alpha \tag{9}$$

Ако се обе стране једначине помноже са NG^T добиће се израз (10) из кога закључујемо да је α сопствени вектор грам матрице са сопственом вредношћу $N\lambda$.

$$N\lambda\alpha = G\alpha \tag{10}$$

Даљи поступак је сличан као код стандардне анализе главних компоненти. Сопствене вредности се сортирају у опадајућем поретку и бира се првих к вредности. С обзиром да се након сопствене декомпозиције грам матрице зна вредност вектора α, вредности координата података у трансформисаном простору се лако могу израчунати помоћу формуле (2). Када приликом предикције наиђемо на нови улаз x, он се може пројектовати у простор сопствених вектора као (11) за шта је поново потребна само функција кернела а не и конкретна трансформација.

$$\phi(x) = \phi(x)^T v = \sum_{i=1}^{N} \phi(x)^T \alpha_i \phi(x_i) = \sum_{i=1}^{N} K(x, x_i)\alpha_i \tag{11}$$

Неке од најчешће коришћених кернела су

- **Линеарни Кернел**: $k(x_i, x_j) = x_i^T x_j$
- **Полиномијални Кернел**: $k(x_i, x_j) = (\gamma x_i^T x_j + r)^d$
- **Кернел који користи Функцију са радијалном основом (RBF)**: $k(x_i, x_j) = \exp\left(-\frac{\|x_i - x_j\|^2}{2\sigma^2}\right)$
- **Сигмоидни Кернел**: $k(x_i, x_j) = \tanh(\gamma x_i^T x_j + r)$

3.3 Опис имплементације алгоритма

Алгоритам за Кернел PCA може се сажети у следећим корацима:

1. **Рачунање Кернел Матрице**: Конструише се $n \times n$ кернел матрица K, где је сваки елемент $K_{ij} = k(x_i, x_j)$.
2. **Центрирање Кернел Матрице**: Центрира се кернел матрица K да би се обезбедило да су подаци центрирани у простору функција. Центрирана кернел матрица K_c се даје са:

   $$K_c = K - \mathbf{1}_n K - K\mathbf{1}_n + \mathbf{1}_n K \mathbf{1}_n$$

   где је $\mathbf{1}_n$ $n \times n$ матрица са свим елементима једнаким $\frac{1}{n}$.
3. **Декомпозиција на Сопствене Вредности**: Изврши се декомпозиција на сопствене вредности центриране кернел матрице K_c. Нека су $\lambda_1, \lambda_2, \ldots, \lambda_n$ сопствене вредности и $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_n$ одговарајући сопствени вектори.

4. **Пројекција података**: Пројекција података на главне компоненте у простору функција је дата са:

$$\mathbf{z}_i = \left(\frac{1}{\sqrt{\lambda_1}} \mathbf{v}_1^T, \frac{1}{\sqrt{\lambda_2}} \mathbf{v}_2^T, \ldots, \frac{1}{\sqrt{\lambda_m}} \mathbf{v}_m^T \right) K_{i,:}^T$$

где $K_{i,:}$ означава i-ти ред кернел матрице K. Такође, извршена је нормализација са сопственим вредностима како би варијансе у новодобијеном простору биле 1, зато што за сопствене векторе знамо да су међусобно нормални, али не знамо да ли им је интензитет 1 да бисмо могли да их посматрамо као орт-ове.

4 Стохастичко угњеждавање суседа

У овом поглављу биће теоријски описан метод стохастичког угњеждавања суседа.

4.1 Алгоритам када се једној тачки из оригиналног простора додељује једна тачка у трансформационом простору

Први корак стохастичког угњеждавања суседа је налажење условне вероватноће $p_{j|i}$ (12) да тачка i види тачку j као свог суседа. Условна вероватноћа $p_{j|i}$ је пропорционална функцији густине вероватноће Гаусове расподеле центриране у тачки i са стандардном девијацијом σ_i. На овај начин се уједно представља и сличност посматраних тачака.

$$p_{j|i} = \frac{\exp(\frac{-||x_i - x_j||^2}{2\sigma_i^2})}{\sum_{k \neq i} \exp(\frac{-||x_i - x_k||^2}{2\sigma_i^2})} \ \forall i \ \forall j \tag{12}$$

У једначини (12) са $p_{j|i}$ означена је сличност, са x_i и x_j означене су координате одговарајућих тачака између којих се рачуна сличност у оригиналном простору, а са σ_i означена је стандардна девијација Гаусове расподеле центриране око тачке x_i.
Како је густина тачака различита у различитим деловима простора, вредност стандардне девијације није иста за све тачке. Да би Гаусова расподела P_i добро представила различите густине, неопходно је да стандардне девијације буду различите за сваку тачку. Очекује се мања стандардна девијација у регионима где је густина тачака велика и већа стандардна девијација у деловима простора где је густина тачака мала. Такође, са повећањем стандардне девијације повећава се ентропија расподеле (13) у одговарајућој тачки.

$$H(P_i) = -\sum_j p_{j|i} \log_2 p_{j|i} \ \forall i \tag{13}$$

У једначини (13) са $H(P_i)$ означена је ентропија Гаусове расподеле центриране у тачки x_i, а са $p_{j|i}$ означена је сличност из формуле (12).
Одговарајућа вредност стандардне девијације може се наћи алгоритмом бинарне претраге тако што се бира она вредност стандардне девијације за коју је ентропија расподеле једнака некој фиксној вредности. Ова фиксна вредност представља хиперпараметар модела и интуитивно представља логаритам глатке процене ефективног броја суседа (*perplexity*) (14) који се дефинише се као:

$$Perplexity(P_i) = 2^{H(P_i)} \ \forall i \tag{14}$$

Алгоритам стохастичког угњеждавања суседа је поприлично робустан на вредност ефективног броја суседа али се типично узимају вредности између 5 и 50.
С обзиром да се подразумева да је удаљеност тачке од саме себе 0 и да приликом оптимизације не желимо да она има утицаја на даљи рачун, вредност условне вероватноће да тачка саму себе изабере за суседа поставља се на вредност 0 (15).

$$p_{i|i} = 0 \ \forall i \tag{15}$$

Сличности у оригиналном простору се могу задати и константама које су добијене експерименталним путем, ако је то могуће за дату поставку проблема, и не морају бити симетричне.
У трансформационом простору, сличности између тачака (16) се рачунају на исти начин при чему је једина разлика то да је стандардна девијација постављена на константну вредност $\frac{1}{\sqrt{2}}$

$$q_{j|i} = \frac{\exp(-||y_i - y_j||^2)}{\sum_{k \neq i} \exp(-||y_i - y_k||^2)} \ \forall i \ \forall j \tag{16}$$

У формули (16) са $q_{j|i}$ представљена је сличност у трансформисаном простору а са y_i и y_j координате тачака чија сличност се рачуна у трансформисаном простору. Константна вредност стандардне девијације не умањује општост јер једина разлика до које то доводи је хомогенији простор излазног система

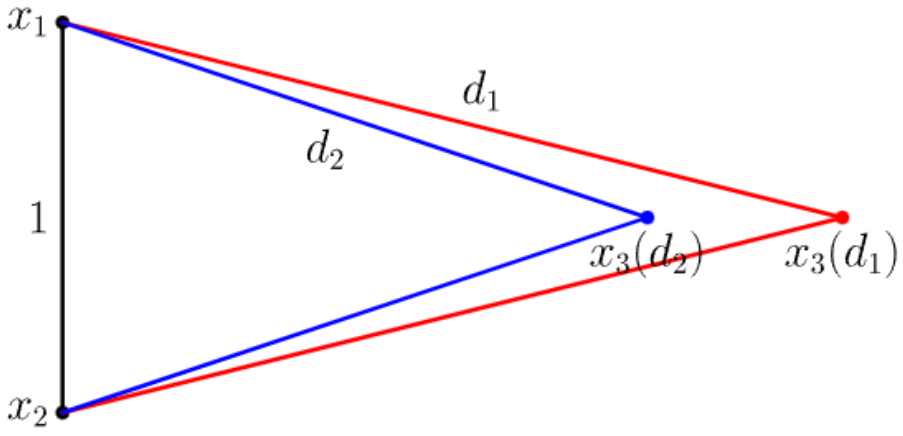
у односу на улазни.
Слично као за високодимензиони простор, сличност тачке са самом собом је постављена на 0 (17).

$$q_{i|i} = 0 \ \forall i \tag{17}$$

Димензионалност новодобијеног простора је такође хиперпараметар, али је, наравно, пожељно да она буде што мања могућа.
Циљ стохастичког угњеждавања суседа је да што више повећа подударање ове две расподеле. У идеалном случају, требало би да оне буду исте. Мера сличности која се користи у ову сврху је КЛ - дистанца (18) као један од најпознатијих статистичких метода за меру сличности две расподеле.

$$C = \sum_i (P_i||Q_i) = \sum_i \sum_j p_{j|i} \log \frac{p_{j|i}}{q_{j|i}} \tag{18}$$

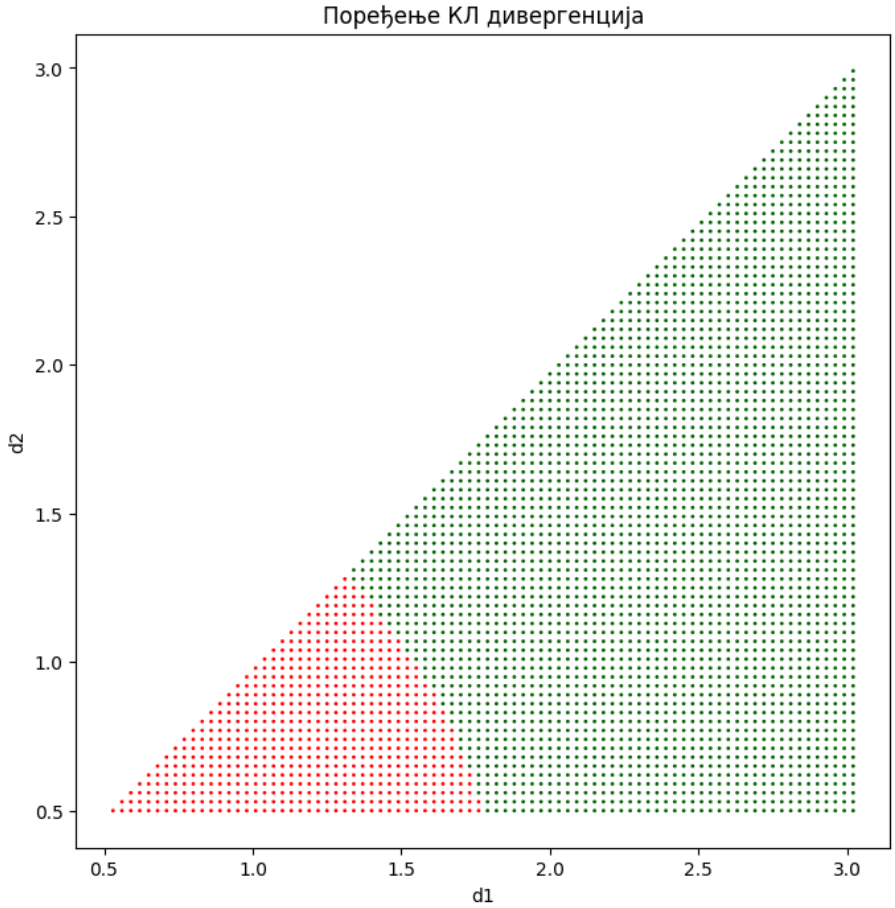
У формули (18) са P_i и Q_i означене су Гаусове расподеле центриране у тачки i у оригиналном и трансформисаном простору респективно, а са $p_{j|i}$ и $q_{j|i}$ означене су величине дефинисане у формулама (12) и (16). Дати израз суштински представља очекивање различитости расподела у оригиналном и трансформисаном простору при чему је вероватноћа сваке од разлика сразмерна одговарајућем $p_{j|i}$.
Треба приметити да када је $p_{j|i}$ мало и $q_{j|i}$ велико њихов количник тежи нули јер је $lim_{x \hookrightarrow 0+} x \log x = 0$. Наравно, за неке конкретне константе између нула и један дата вредност ће бити неки мали број различит од нуле и заправо ће представљати грешку моделовања. С обзиром да је вероватноћа да су суседи за мале дистанце велика, значење ових формула је да трансформисање великих дистанци у мале неће резултовати превише великом грешком. Ипак, ако је $p_{j|i}$ велико и $q_{j|i}$ мало модел се кажњава у великој мери, тиме што ће израз у логаритму бити неки јако велик број. Односно, много је горе када мале дистанце у оригиналном простору трансформишемо у велике дистанце у трансформисаном простору. Интуитивно, стохастичко угњеждавање суседа фокусира се да одржи локалну структуру оригиналних података тако што ће да одржи блискост тачака које су биле блиске пре трансформације. С обзиром да је ово тврђење на основу лимеса, оно неће важити за сваки пар ових вероватноћа већ само за екстреме, али може се у програмском језику *Python* интуитивно приказати на примеру три тачке да постоји одређена граница након које овај услов почиње да важи.
На слици 1 су приказане 4 тачке. Прве две су на удаљености 1 једна од друге а трећа и четврта се налазе на симетрали дужи која их спаја, при чему је трећа ближа а четврта даља. Идеја је да посматрамо КЛ - дистанцу у случају када ближу тачку преместимо у даљу и када даљу тачку преместимо у ближу.



Слика 1 Изглед три изабране тачке у простору [9]

У *Python* програмском језику је генерисана решетка дистанци између 0.5 и 3. Узете су само оне дистанце за које је d1 > d2, а затим је за све комбинације оваквих дистанци израчунато да ли је КЛ - дивергенција преласка d1 у d2 мања од преласка d2 у d1. Тачке са координатом (d1, d2) за које је ово испуњено су обојене у зелено, а оне за које није у црвено и добијена је слика 2. Са слике се може

приметити да тврдња важи када већа дистанца постане довољно велика што се уклапа са асимптотским претпоставкама али не важи за мале дистанце. То такође значи да чак и за дистанце за које ова тврдња не важи тачке које су близу неће се удаљити више од тог ограничења за d1 након кога тврдња почиње да важи.



Слика 2 Поређење КЛ дивергенција у простору

Једно од решења које се прво намеће за минимизацију КЛ-дистанце је коришћење градијентног спуста или неке од његових модификација за шта нам је потребно рачунање градијента. Рачун за то је доста комплексан али крајњи резултат (19) који се овим добија је поприлично једноставан и интуитиван. У датој формули са C је означена функција губитка, док су остале ознаке дефинисане као у претходним формулама. Први члан се може протумачити као збир крутости привлачења (или, ако је разлика негативна, одбијања) која је сразмерна томе колико j види i као комшију и колико i види j као комшију и овај збир је отежињен дистанцом између ове две тачке. Самим тим, градијент се може протумачити као резултујућа сила (силу можемо као у случају силе еластичности посматрати као производ крутости и растојања између тачака) које привлаче или одбијају ову тачку у различитим правцима.

$$\frac{\delta C}{\delta y_i} = 2\sum_j (p_{j|i} - q_{j|i} + p_{i|j} - q_{i|j})(y_i - y_j)\ \forall i \tag{19}$$

Као оптимизациони метод се користи градијентни спуст али са додатим моментом (20) (како би се алгоритам убрзао и да се не би заглавио у локалном оптимуму). Форумла (20) може се упростити увођењем ознаке (21) чиме се долази до просте рекурзивне једначине (22) из које видимо да се суштински уместо додавања само тренутног градијента додаје и експоненцијално опадајућа сума претходних

градијената. У формулама (20) (21) и (22) је са η означена брзина учења а са $\alpha(t)$ означен моментум у итерацији t којим утичемо на то колику значај ће имати претходни градијенти, а са y_t означена је вредност трансформације у итерацији t.

$$y_t = y_{t-1} + \eta \frac{\partial C}{\partial y} + \alpha(t)\left(y_{t-1} - y_{t-2}\right) \tag{20}$$

$$dy_t = y_t - y_{t-1} \tag{21}$$

$$dy_t = \eta \frac{\partial C}{\partial y} + \alpha(t)\ dy_{t-1} \tag{22}$$

Поред додатог моментума, додаје се и Гаусов шум чија се варијанса временом смањује до тренутка када у потпуности не нестане. Ово је нека форма симулираног каљења која помаже при избегавању локалних оптимума и долажењу до глобалног оптимума. Ако би се варијанса овог шума мењала веома споро у моменту када глобална структура почиње да се формира, трансформисан простор ће имати бољу глобалну организацију али за тако нешто би морало да се посвети доста времена тражењу праве мере како почетне варијансе тако и стопе са којом ће варијанса да опада за конкретан скуп података. Поред тога, битна је и количина моментума која на ово може утицати тако да је тражење погодних хиперпараметара доста захтевно.
Погодна иницијализација је случајан распоред тачака али близу координатног почетка (Случајно узимамо одбирке Гаусове расподеле са малом варијансом и нултом средњом вредношћу).

4.2 Алгоритам када се једној тачки из оригиналног простора додељује више тачака у трансформационом простору

Стохастичко угњеждавање суседа се може лако прилагодити за проблеме који захтевају да се омогући трансформисање једног оригиналног податка у више трансформисаних. Један од најбољих примера оваквог проблема је трансформисање вокабулара речи у неке векторе бројева који су адекватни улази модела. У том случају, неке речи имају више значења и желимо да посебним вектором обележимо свако од значења речи за шта нам треба више тачака у трансформисаном простору. Свака од тих тачака у трансформисаном простору учествоваће са одређеним уделом π који је број између 0 и 1. Сума удела свих трансформисаних тачака оригиналне тачке је 1. На овај начин, при формирању трансформације тачке за сваку тачку ћемо као њену расподелу добити мешавину Гаусових расподела (23) где је y_{i_b} положај трансформације b оригиналне тачке i, а π_{i_b} је њен удео.

$$q_{ij} = \frac{\sum_b \pi_{i_b} \sum_c \pi_{j_c} \exp(-||y_{i_b} - y_{j_c}||^2)}{\sum_k \sum_d \pi_{k_d} \exp(-||y_{i_b} - y_{k_d}||^2)}\ \forall i\ \forall j \tag{23}$$

Ако уведемо ознаку $r_{i_b j_c}$ (24) за вероватноћу да трансформисана верзија b тачке i као суседа изабере трансформисану верзију c тачке j.

$$r_{i_b j_c} = \frac{\pi_{j_c} \exp(-||y_{i_b} - y_{j_c}||^2)}{\sum_k \sum_d \pi_{k_d} \exp(-||y_{i_b} - y_{k_d}||^2)}\ \forall i\ \forall j \tag{24}$$

Градијент q_{ij} по уделу верзије g тачке m дат је формулом (25) где је δ_{m_j} Кронекерова делта.

$$\frac{\partial q_{ij}}{\partial \pi_{m_g}} = \delta_{m_i} \sum_c r_{m_g j_c} + \sum_b \pi_{i_b} \pi_{m_g} r_{i_b m_g} \left(\delta_{m_j} - \sum_c r_{i_b j_c}\right)\ \forall i\ \forall j\ \forall m\ \forall g \tag{25}$$

Као крајњи градијент функције губитка по уделу верзије g тачке m дат је формулом (26)

$$\frac{\partial C}{\partial \pi_{mg}} = -\sum_i \sum_j p_{ij} q_{ij} \frac{\partial q_{ij}}{\partial \pi_{mg}}\ \forall m\ \forall g \tag{26}$$

Ипак, доста је једноставније посматрати уделе као *softmax* функцију одговарајућих тежина које се оптимизују.

4.3 Предности и мане

Из свега наведеног, можемо закључити да су два главна недостатка стохастичког угњеждавања суседа његова велика компјутерска комплексност и доста велики број хиперпараметара у односу на остале алгоритме које морамо сами подешавати (увећање за метод раног преувеличавања са увећањем и број итерација након којих увећање престаје, вредности моментума у току тренирања, брзина учења, глатка процена ефективног броја суседа).За само 3000 примера овом алгоритму је потребно пар сати да нађе одговарајуће трансформације.Такође, размотримо шта се дешава када су нађене варијансе велике. У овом случају ће именилац унутар експонента бити велики што доводи до тога да вредност унутар експонента буде близу нуле и сви експоненти ће тежити јединици. Самим тим, Гаусова расподела се приближила униформној. До овог закључка се , такође, може доћи интуитивно јер ако замислимо изглед гаусове расподеле када се стандардна девијација повећава и врх функције густине вероватноће гаусове расподеле се приближава вредности репова. Сада, алгоритам нема на основу чега да донесе одлуку о томе ком кластеру која тачка припада јер су све тачке превише сличне и због тога ће све тачке завршити у истом кластеру.

5 Т-расподељено стохастичко угњеждавање суседа

Т-расподељено стохастичко угњеждавање суседа решава наведене проблеме тако што мења функцију губитка на два начина:

1. Користи симетричну верзију функције губитка тј. користи унакрсну ентропију уместо КЛ дивергенције

2. Користи студентову т-расподелу за налажење сличности у трансформационом простору

5.1 Симетризација функције губитка

Један начин да симетризујемо цео модел је да уместо условних расподела посматрамо заједничке расподеле, што би довело до практично истих формула у случају Гаусове расподеле али би подразумевало да су сада вероватноћа да i изабере j за суседа иста као да j изабере i за суседа и у случају другачијих расподела или унапред одређених сличности. Ипак, с обзиром да све формуле у случају Гаусове расподеле остају исте, можемо приметити да када имамо неки аутлајер у скупу улазних података, његова удаљеност од свих осталих података је јако велика, самим тим сличност ће бити јако мала што ће учинити да он скоро уопште неће променити функцију губитка. Самим тим, модел неће детектовати никакав проблем приликом наиласка на аутлајере и позиција аутлајера неће бити дефинисана, већ ће он моћи да се нађе на било којој позицији близу координатног почетка, што је јако лоше. Из овог разлога, заједничка вероватноћа у оригиналном простору се рачуна симетризовањем условних вероватноћа по формули $p_{ij} = \frac{p_{j|i}+p_{i|j}}{2n} \ \forall i \ \forall j$. Ова формула омогућава да увек буде испуњен услов $\sum_j p_{ij} > \frac{1}{2n} \quad \forall i \ \forall j$, где је n величина скупа, што спречава да вредности сличности постану превише мале и решава претходно наведени проблем у великој мери. У трансформационом простору се за симетричну вероватноћу користи израз (16) јер његовим коришћењем не долази ни до каквих проблема. Главна предност симетризовања модела је то што добијамо значајно једноставнији градијент чиме се убрзава алгоритам (27).

$$\frac{\partial C}{\partial y_i} = 4 \sum_j (p_{ij} - q_{ij})(y_i - y_j) \ \forall i \tag{27}$$

Резултати добијени коришћењем симетричног модела су исти а некад чак и бољи него коришћењем стандардног асиметричног модела.

5.2 Проблем гомилања тачака

У овом подпоглављу се детаљније анализира проблем гомилања тачака као један од највећих проблема стохастичког угњеждавања суседа. Посматрајмо скоро униформну дводимензионалну површ у тродимензионалном простору. Могуће је поприлично добро моделовати овакву површ у дводимензионалном простору и пример оваквог скупа података су С-крива и швајцарска ролница. (О овим скуповима података биће више речи у поглављу *Коришћени скупови података*). Затим, посматрајмо неки скуп података који је десетодимензионалан а налази се унутар неког простора са много више димензија (пример оваквог скупа података је MNIST), овакви подаци не могу довољно прецизно да очувају сву своју структуру у дводимензионалном простору из више очигледних разлога. На пример, ако имамо тачке које се налазе све на једнакој удаљености (сфера са центром у тренутно посматраној тачки) у дводимензионалном простору оне више неће бити на истој удаљености од посматране тачке и као последица тога расподела ће се значајно променити. Све тачке које су лежале на сфери у n димензија биће пресликане у кружницу чиме ће растојања између неких тачака значајно да се мења и густина тачака постаће значајно већа. Из ових разлога дешава се да тачке које имају различит контекст неће бити много више удаљене него тачке које имају исти контекст. Да би метод дао добре резултате, тачке у оригиналном скупу које припадају истим кластерима морају бити значајно удаљене једна од друге да би у дводимензионалном простору остале удаљене.

5.3 Зашто се користи баш Студентова расподела?

У т-расподељеном стохастичком угњеждавању суседа овај проблем се решава коришћењем т-расподеле уместо Гаусове расподеле приликом налажења сличности у трансформисаном простору. Т-расподела са једним степеном слободе (Кошијева расподела) (28) је позната по томе што има дебље репове од Гаусове расподеле што омогућава да тачке које не припадају истом кластеру у оригиналном простору а нису на превише великој удаљености буду представљене довољно великим удаљеностима у трансформисаном простору.

$$q_{ij} = \frac{(1 + ||y_i - y_j||^2)^{-1}}{\sum_{k \neq l}(1 + ||y_k - y_l||^2)^{-1}} \tag{28}$$

Користи се један степен слободе зато што има особину да када је разлика у дистанцама велика почиње да се понаша по закону инверзних квадрата који је познат у физици. По овом закону сила (на пример електрична и гравитациона) опада са квадратом удаљености између посматраних тачака. Самим тим, више се не форсира да све тачке буду близу, већ је веза између удаљенијих тачака лабавија, а између ближих тачака чвршћа. Између удаљених кластера се успоставља иста интеракција као између појединачних тачака без обзира на величину кластера чиме се чува и глобална и локална структура података. Из свих наведених разлога, заједничка расподела остаје скоро непромењена приликом мењања скале мапе, и тиме се спречава проблем нагомилавања. Ова расподела је такође добар избор из практичних разлога. Она представља мешавину бесконачног броја Гаусових расподела али је компјутерски знатно једноставнија за израчунавање јер не садржи никакав експонент.
КЛ дивергенција између Гаусове расподеле у оригиналном простору и т-расподеле у трансформисаном простору дата је формулом (29)

$$\frac{\partial C}{\partial y_i} = 4 \sum_j \frac{(p_{ij} - q_{ij})(y_i - y_j)}{1 + |y_i - y_j|^2} \ \forall i \tag{29}$$

Може се приметити да *t-SNE* у значајно већем делу простора користи мање градијенте за исте улазе. *t-SNE* одбија тачке чија је сличност мала а моделоване су великом сличношћу. *SNE* ради то такође али одбијање које он креира је значајно мање. Из тог разлога за *t-SNE* није неопходно симулирано каљење јер он већ има поприлично велике силе којима може да привуче или одбије два кластера на основу њихове сличности односно разлике.

5.4 Имплементација т-расподељеног стохастичког угњеждавања суседа

У имплементацији овог алгоритма коришћен је моментум како би се смањио број итерација потребних за конвергенцију као у формули (22). Овај моментум се одржава на малој вредности до момента док кластери не постану довољно организовани, након чега се повећава (30).

$$\alpha(t) = \begin{cases} \alpha_1 & \text{до итерације} t_1 \text{ када су подаци релативно раздвојени} \\ \alpha_2 & \text{иначе} \end{cases} \tag{30}$$

Поред тога, за брзину учења коришћена је адаптивна шема (31)

$$\eta = \begin{cases} (\eta \cdot 0.8) & \text{за } (\frac{\partial C}{\partial y} > 0)(dy_{t-1} > 0) \\ (\eta + 0.2) & \text{иначе} \end{cases} \tag{31}$$

где су све ознаке исте као у формули (22). Идеја иза овакве промене брзине учења је да у случају када су претходна промена и тренутни градијент истог знака спречимо да моментум превише надјача утицај тренутног градијента, а у случају кад су истог знака ове две промене раде исту ствар па се утицај градијента може смањити ради боље стабилности алгоритма.
Ипак, како би се спречио нестајући градијент, неопходно је након адаптације брзине учења проверити да ли је њена вредност премала (мања од неке минималне вредности) и ако јесте поставља се на ту минималну вредност. Овај корак је неопходан да не би дошло до тога да се у некој итерацији уопште не деси никакво ажурирање вредности иако имамо ненулти градијент.

Постоје два начина иницијализације којима алгоритам може додатно да се побољша а то су рана компресија и рано преувеличавање.
Рана компресија подразумева да на самом почетку тренирања додајемо L2 кажњавајући члан којим кажњавамо алгоритам када превише одступа од координатног почетка тако што овај члан представља средње квадратно одстојање тачака од координатног почетка. На овај начин, почетних пар итерација алгоритам држи све тачке близу једне другима како би лакше и брже обишао све тачке и научио што више о њима, након чега их пушта да се удаље.
Рано преувеличавање подразумева множење сличности у оригиналном простору неким бројем већим од 1 (у имплементацији је коришћен број 4) што ће довести до премалих сличности у трансформисаном простору. Овиме се алгоритам подстиче да се сконцентрише на највеће сличности у оригиналном простору и прво њих одвоји, чиме се постиже што веће могуће раздвајање кластера међусобно. Како би се алгоритам убрзао, не иницијализује се директно скупом података већ се прво скуп података редукује на 30 компоненти помоћу анализе главних компоненти, а онда се те компоненте користе као улаз у алгоритам.

6 *UMAP(Uniform Manifold Approximation and Projection)*

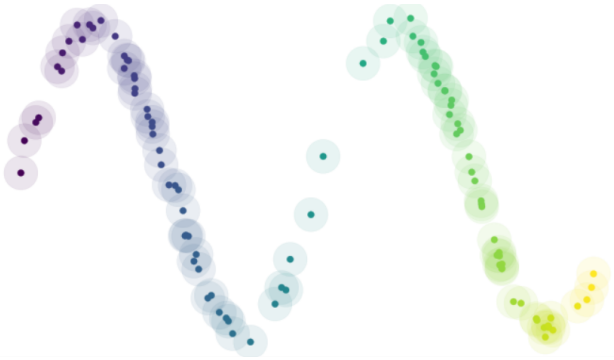
6.1 Увод

Метод униформне апроксимације и пројекције на равни је метод који се заснива на методама учења површи, користи fuzzy скупове и топологију. Из тог разлога, прво ће бити дефинисано пар основних теоријских појмова потребних за даљу анализу алгоритма. За разлику од других алгоритама покушава да реши проблем у што општијем облику а тек накнадно га прилагођава конкретним скуповима података. У раду се креће од стриктних теоријских формула и дефиниција које овде неће бити разматране. Овако формулисан алгоритам није примењив у пракси, па након теоријске анализе аутори рада прилагођавају алгоритам решавању реалних проблема. Метод униформне апроксиманције и пројекције на равни парира алгоритму т-расподељеног стохастичког угњеждавања суседа што се тиче визуелизације података, али за разлику од њега може да се користи за редукцију података на веће димензионалности и има значајно боље компјутерске перформансе. Да би овај алгоритам могао да се користи следећи услови морају бити испуњени:

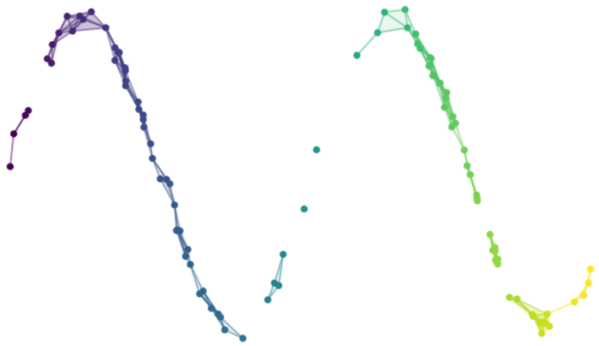
1. Постоји површ на којој су подаци униформно расподељени
2. Та површ је локално повезана
3. Главни циљ редукције димензија је да се очува тополошка структура те површи

6.2 Основни појмови и идеја алгоритма

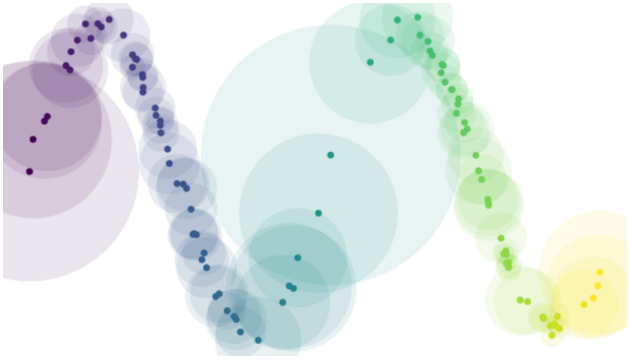
k-simplex представља градивни блок у простору који се формира од к+1 независних тачака у топологији и његов омотач се састоји од *(к-1) -simplex* - а. 0-симплекс је тачка, 1 - симплекс је дуж која спаја две тачке, 2 симплекс је троугао (састоји се од 3 независне дужи а омотач му се састоји од три 2-симплекса), 3 симплекс је тетраедар а омотач су му 4 троугла. Симплексе можемо посматрати и као све могуће подскупове које можемо формирати унутар скупа. На пример, у случају 3 симплекса који чине 4 независна елемента , сви 2-симплекси који чине његов омотач чине све могуће подскупове са 3 елемента, сви 1-симплекси који су део њега чине све могуће подскупове са два елемента, а 0-симплекси су сви појединачни елементи тог скупа. Могу се формирати уније и пресеци симплекса тако што их спајамо на месту заједничког дела омотача (слично формирању уније и пресека скупа). Ово се може применити на реалне скупове података тако што ће свака тачка бити 0-симплекс, а везе између тачака се формирају тако што око сваке тачке посматрамо круг фиксног радијуса и спајамо тачке чији се одговарајући кругови преклапају као што је приказано на слици 3. На овај начин омотач података се апроксимира површима лопти центрираним у подацима. На слици 4 приказан је пример овако формираног графа. Овиме се претпоставља да су подаци део неког тополошког просотра, али ми немамо довољно података о том простору, с обзиром да нам је дат само одређен број података а не сви могући. Ова апроксимација доста добро приказује топологију података и нема велику компјутерску комплексност из разлога што је доста симплекса који је чине величине 0 или 1.



Слика 3 Приказ омотача реалних података [8]



Слика 4 Приказ топологије за реалне податке [8]



Слика 5 Приказ омотача у случају униформне расподеле података по контури [8]

Ипак, може се одмах уочити да ће у општем случају величина датог полупречника представљати хиперпараметар модела који није толико лако подесити. Наиме, ако за полупречник узмемо премалу вредност добићемо премало пресека и самим тим превише одвојених компоненти а ако узмемо превелику вредност добићемо премали број компоненти. Овај проблем не би постојао када би расподела била униформна, јер би онда све суседне тачке биле приближно једнако удаљене једне од других. У том случају потребан полупречник хиперсфере је једнак половини удаљености две суседне тачке. Овиме не би постојао проблем ни превелике димензионалности у густим деловима ни премале димензионалности која доводи до раскидања графа на компоненте у ретким деловима.

На жалост, реални подаци скоро никада нису униформно расподељени и зато је потребно локализовати ову идеју у деловима простора који се могу сматрати униформно расподељеним. Из тог разлога, уместо да се посматра фиксан полупречник, може се посматрати број суседа. Самим тим, ако фиксирамо број суседа који посматрамо, мањи полупречник ће бити додељен гушћим деловима простора у односу на ређе као што је приказано на слици 5. Овиме се добија граф заснован на к најближих суседа који се често користи и у доста познатих алгоритама. Приметити да је идаље потребно изабрати вредност овог хиперпараметра али је доста лакша и интуитивнија за бирање него конкретне удаљености за које је потребно посматрати хистограм удаљености који је директно зависан од конкретних улаза алгоритма. Као и у већини сличних алгоритама заснованих на к најближих суседа, за мале вредности к добијамо процену која добро учи детаље и локална понашања док а за велико к добијамо глатку процену која учи само већинска и глобална понашања података. Разлика код *UMAP-a* у односу на друге алгоритме је чињеница да се не посматра детерминистички пресек хиперкружница већ се уводи стохастички део модела који посматра и колико се пресек налази далеко од дате тачке. Што је пресек даље од дате тачке, то је мања вероватноћа да припада истом скупу података. Из тог разлога се класични детерминистички скупови замењују *fuzzy* скуповима података.

Метрика заснована на броју суседа неће увек бити симетрична већ може дати различите дистанце ако се удаљеност посматра са леве односно десне стране дужи. С обзиром да је за излаз алгоритма потребно да сваки 1-симплекс буде окарактерисан једном тежином потребно је да се ове две вредности

укомбинују у једну. За то постоји доста могућности, али она која се интуитивно намеће с обзиром да се ради о *fuzzy* скуповима је њихова *fuzzy* унија која ће обезбедити да на излазу добијемо потпуно повезану целину. Разлика између метрике трансформисаног и оригиналног простора је то што у трансформисаном простору не желимо униформност по некој површи већ по еуклидском простору са онолико димензија на колико желимо да редукујемо податке, што не захтева више променљиву метрику већ је довољно да користимо исту метрику за све тачке тј. довољно нам је посматрање дистанце а не варирајуће метрике и суседа.

6.3 Имплементација алгоритма

Као за све методе који се заснивају на методу најближих суседа, прво се налази к најближих суседа коришћењем било ког од метода који то раде. Након тога налазе се параметри ρ_i и σ_i. Параметар ρ_i се рачуна као (32) која обезбеђује услов локалне повезаности. Захваљујући томе ће сваки кластер имати више од једне тачке у себи што побољшава алгоритам у односу на друге и брани га од проклетства димензионалности. Параметар σ_i се добија из једначине (33) и може се наћи бинарном претрагом као код *t-SNE*. У овим једначинама је к коришћени број суседа, x_i посматрана тачка, x_{i_j} њен j-ти сусед а $d(x_i, x_{i_j})$ оператор међусобне удаљености.

$$\rho_i = \min\{d(x_i, x_{i_j}) \,|\, 1 \leq j \leq k, d(x_i, x_{i_j}) > 0\} \tag{32}$$

$$\sum_{j=1}^{k} \exp\left(-\frac{\max(0, d(x_i, x_{i_j}) - \rho_i)}{\sigma_i}\right) = \log_2(k) \tag{33}$$

Након дефинисања ових параметара могуће је дефинисати граф проблема помоћу темена графа која представљају тачке улазног скупа података и ивица које спајају тачке са њихових к најближих суседа и са тежинама приказаним у формули (34).

$$w((x_i, x_{i_j})) = \exp\left(-\frac{\max(0, d(x_i, x_{i_j}) - \rho_i)}{\sigma_i}\right) \tag{34}$$

Ове тежине се могу интерпретирати као вероватноће постојања гране између две тачке. Ово се такође може посматрати као 1 симплекс *fuzzy*-скуп чија је функција припадања дефинисана овим тежинама. Приметити да је овај граф усмерен и да се може посматрати и као скуп више локалних графова од којих сваки одговара тачно једној од тачака. оришћењем т-конорме може се добити унија ових скупова као (35)

$$\mathbf{B} = \mathbf{A} + \mathbf{A}^T - \mathbf{A} \circ \mathbf{A}^T \tag{35}$$

где је A матрица суседства формираног графа а $\circ$ Хадамаров производ, а добијена матрица B се може интерпретирати као матрица суседства неусмереног графа са истим теменима али са измењеним тежинама где свака тежина представља вероватноћу да барем једно од два посматрана темена сматра оно друго за суседа. Фази скуп са својом функцијом припадности у трансформисаном простору који је подскуп простора R^d, који ће представљати вероватноће суседства тачака дефинише се као (36) $\Phi : R^d \times R^d \to [0, 1]$.

$$\Phi(x, y) = \frac{1}{(1 + a\|x - y\|_2^2)} \tag{36}$$

где се параметри a и b налазе оптимизацијом нелинеарним медотом најмањих квадрата, где је функција која се оптимизује $\Psi : R^d \times R^d \to [0, 1]$ дата формулом (37)

$$\Psi(x, y) = \begin{cases} 1 & \text{ако } \|x - y\|_2 \leq \text{min-dist} \\ \exp\left(-(\|x - y\|_2 - \text{min-dist})\right) & \text{иначе} \end{cases}. \tag{37}$$

Након налажења ових параметара оптимизација се ради тражењем градијента губитка унакрсне ентропије између улазног и излазног скупа(38).

$$C((A, \mu), (A, \nu)) = \sum_{a \in A} \mu(a) \log\left(\frac{\mu(a)}{\nu(a)}\right) + (1 - \mu(a)) \log\left(\frac{1 - \mu(a)}{1 - \nu(a)}\right) \tag{38}$$

Ова грешка осликава два дела која ће се рачунањем градијента трансформисати у привлачну и одбојну силу. Први члан представља привлачни део грешке јер се његовом минимизацијом минимизује $\nu(a)$ тј. покушава да што више приближи тачке у простору док други члан минимизује $1-\nu(a)$ тј. максимизује $\nu(a)$ чиме удаљава тачке у простору Наведени израз се може приказати као формула (39)

$$C((A,\mu),(A,\nu)) = \sum_{a\in A}(\mu(a)\log(\mu(a)) + (1-\mu(a))\log(1-\mu(a))) - \sum_{a\in A}(\mu(a)\log(\nu(a)) + (1-\mu(a))\log(1-\nu(a))) \tag{39}$$

С обзиром да видимо да су прва два члана константна и не могу да се оптимизују ова функција губитка своди се на формулу (40)

$$C((A,\mu),(A,\nu)) = \sum_{a\in A}(\mu(a)\log(\nu(a)) + (1-\mu(a))\log(1-\nu(a))) \tag{40}$$

У формулама (38), (39) и (40) коришћене су ознаке (A,μ) и (A,ν) за фази скупове који одговарају улазном скупу и трансформисаном скупу података. Градијент овог губитка састоји се од две силе привлачне (41) и одбојне (42). У овим формулама са y_i означене су координате у трансформисаном простору док је са ϵ означена мала позитивна константа како би се избегло дељење са нулом. Остале ознаке имају значење као у досадашњим формулама. С обзиром да радимо са фази скуповима, градијент се примењује стохастички негативним одабирањем. За сваки уређен пар тачака из скупа генерише се случајан број од 0 до 1 и ако је он мањи од вероватноће да прва тачка уређеног пара види другу тачку уређеног пара као суседа, примењује се привлачна сила од прве ка другој тачки (мења се позиција прве тачке тако да се приближи другој тачки). Затим се одређен број пута (који је хипер параметар) случајно бира тачка из трансформисаног простора и примењује се негативна сила од прве ка случајно изабраној тачки (мења се позиција прве тачке тако да се удаљи од случајно изабране тачке). Тј. случајним избором те тачке сматрамо далеким чак и ако нису. Овај метод се користи да би се убрзао процес оптимизације и да би метод радио и за велике скупове података.

$$.F_{attractive} = \frac{\partial \sum_{a\in A}\mu(a)\log(\nu(a))}{\partial y_i} = \frac{-2ab\|y_i-y_j\|_2^{2(b-1)}w((x_i,x_j))(y_i-y_j)}{(1+a\|y_i-y_j\|_2^2)} \tag{41}$$

$$F_{repelling} = \frac{\partial \sum_{a\in A}(1-\mu(a))\log(1-\nu(a))}{\partial y_i} = \frac{2b(1-w((x_i,x_j)))(y_i-y_j)}{(\epsilon+\|y_i-y_j\|_2^2)(1+a\|y_i-y_j\|_2^2)} \tag{42}$$

7 Експериментална поставка

За анализу главних компоненти са и без кернела коришћено је издвајање 30 димензија а приказане су само прве две. За анализу главних компоненти са коришћењем кернела коришћен је кернел који користи Функцију са радијалном основом и варијансом 10 у случају швајцарске ролнице. У случају с-криве коришћене су варијансе 0.05, 0.5, 5 како би се приказао утицај овог хипер-параметра, док су најбољи резултати добијени за 0.5. За *MNIST* скуп података било је непоходно да варијанса буде 5. За т-расподељено стохастичко угњеждавање суседа коришћени су подразумевани хипер параметри:

1. Укупан број итерација 500
2. Метод раног преувеличавања са увећањем 4 првих 50 итерација
3. Моментум 0.5 до 250-те итерације и 0.8 након 150-те итерације
4. Брзина учења је иницијално 100 и временом се мења адаптивно на основу формуле (31)
5. За скуп података који формира С-криву као улаз у модел се даје оригиналан скуп података, док се за остале податке на улазу шаље 30 PCA компоненти
6. Глатка процена ефективног броја суседа 40

У случају *UMAP* алгоритма коришћени су следећи хиперпараметри:

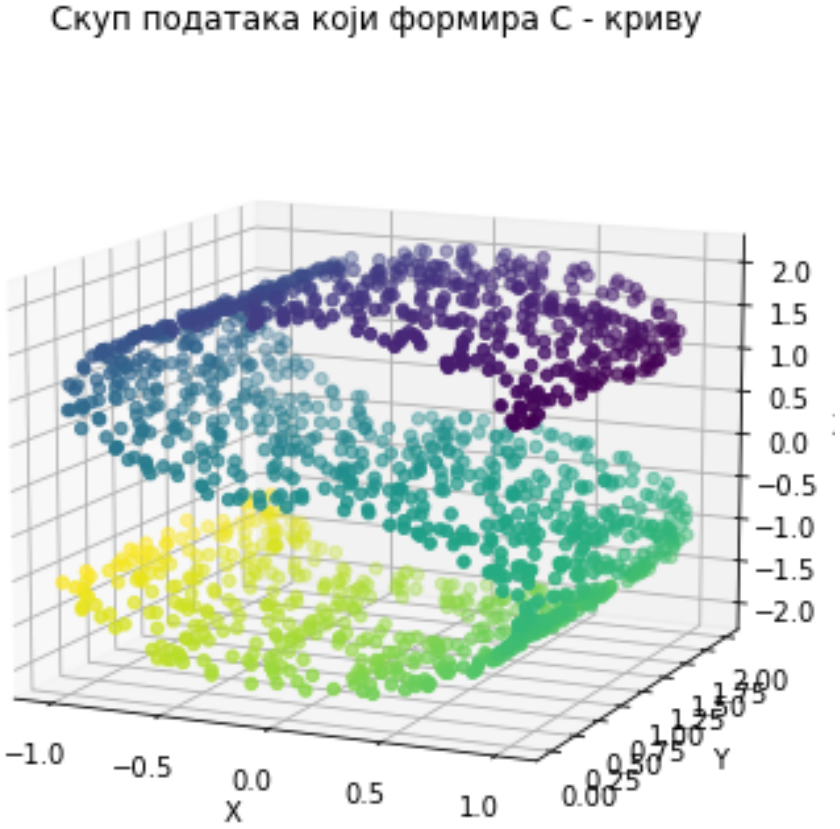
1. Минимална дистанца 0.1, 0.5, 0.9
2. Број к најближих суседа 5, 15, 50

8 Коришћени скупови података

У овом поглављу представљене су основне информације о коришћеним скуповима података за демонстрацију рада анализираних алгоритама.

8.1 С-крива

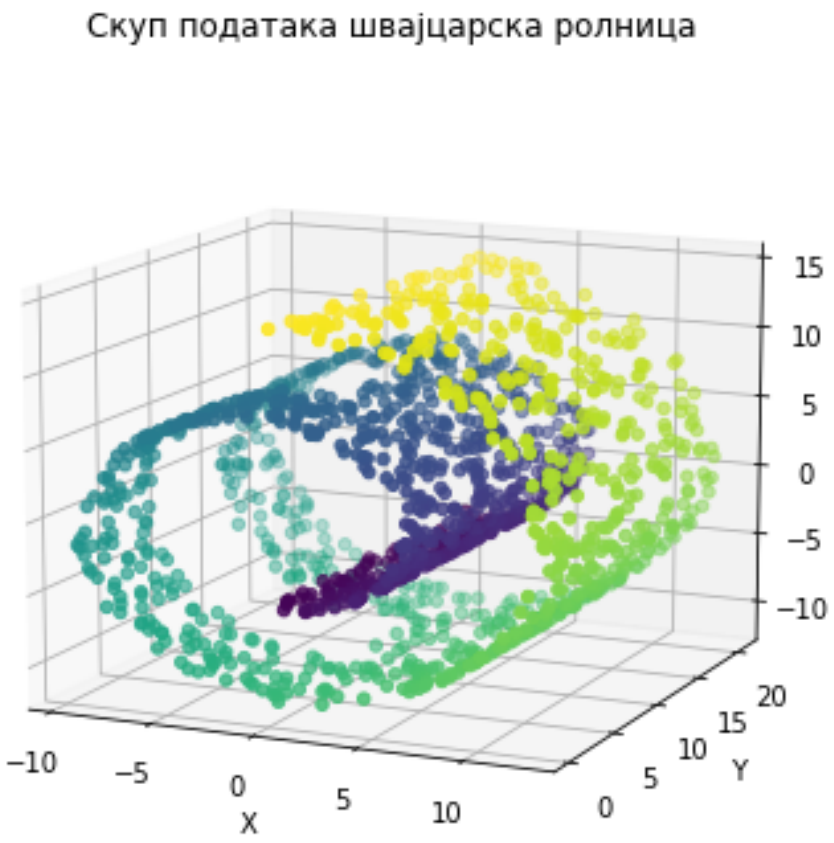
Овај скуп података чини 1500 тродимензионалних података који формирају површ у облику слова С при чему је идеја да тачке које су близу једне другима у овој површи (еуклидско растојање) остану близу али и да буде очувана локална структура тј. да тачке сличних нијанси буду близу једне других. Приказ С- криве приказан је на слици 6.



Слика 6 Изглед С-криве у оригиналном простору

8.2 Швајцарска ролница

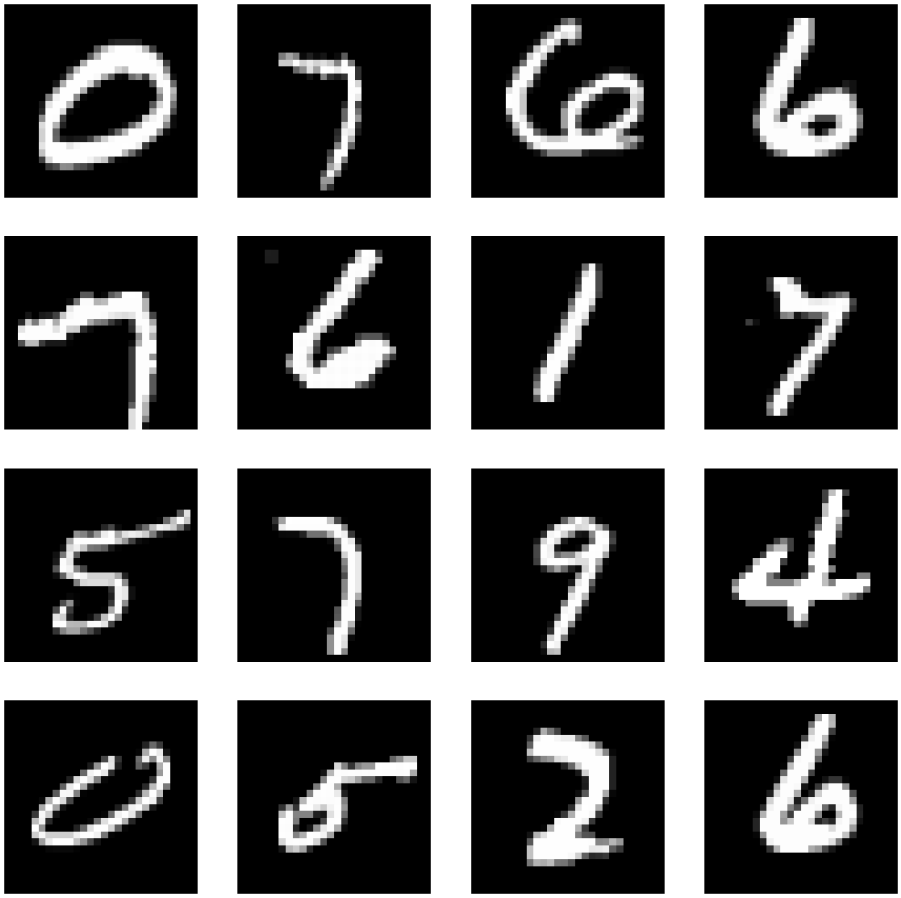
Овај скуп података чини 1500 тродимензионалних података који формирају површ швајцарске ролнице. Слично као мало пре, идеја је да се очувају локална и глобална структура података. Приказ С- криве приказан је на слици 7.



Слика 7 Изглед Швајцарске ролнице у оригиналном простору

8.3 *MNIST* скуп података

MNIST скуп података један је од најосновнијих и најлакших скупова података за тренирање модела за случај компјутерске визије. Ипак, за случај редукције димензија, овај скуп података није толико једноставан с обзиром да слике могу имати и по пар стотина хиљада обележја. Величина обучавајућег скупа је 60 000 података а величина тест скупа је 10 000 података. Састоји се од сивих слика димензија 28x28 које представљају цифре написане руком од 0 до 9. Самим тим, очекујемо раздвајање у 10 групација тачака у трансформисаном простору. Због великог времена тренирања, нису коришћени сви подаци из овог скупа већ само 600 случајно изабраних слика, при чему се водило рачуна да скуп остане избалансиран. Примери неких слика из овог скупа приказани су на слици 8.



Слика 8 Изглед *MNIST* скупа података

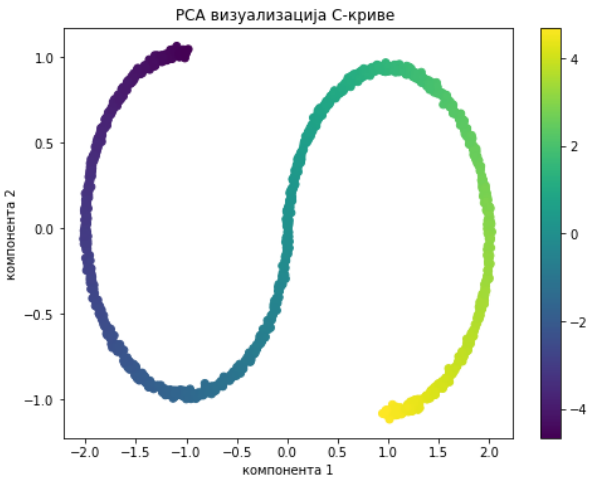
9 Резултати

У овом поглављу приказани су сви добијени резултати при чему су у сваком подпоглављу приказани резултати за одређен алгоритам.

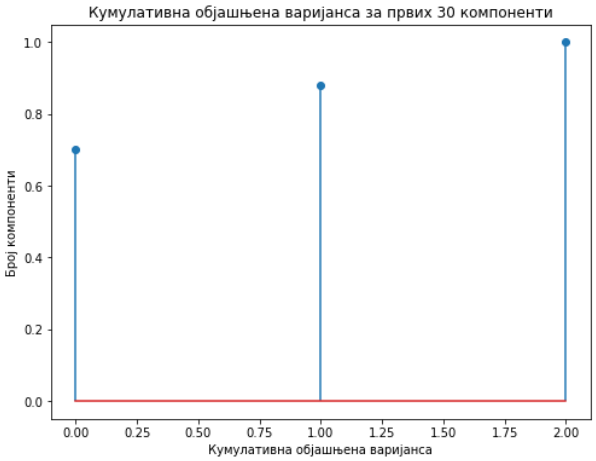
9.1 Резултати над различитим скуповима података

У овом подпоглављу приказани су резултати за сваки од метода појединачно по различитим скуповима података.

9.1.1 PCA

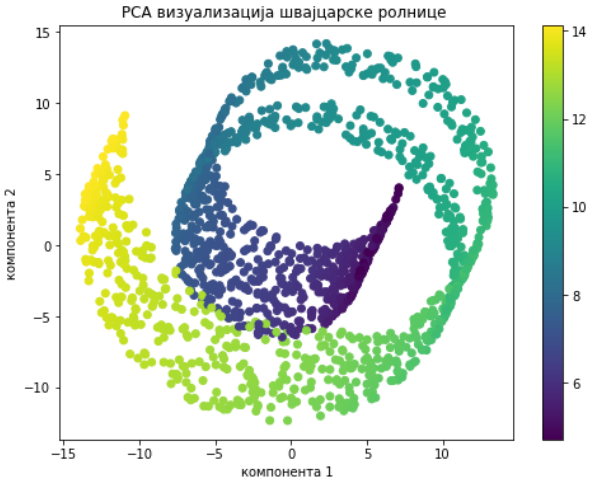


(а) Изглед С-криве након анализе главних компоненти

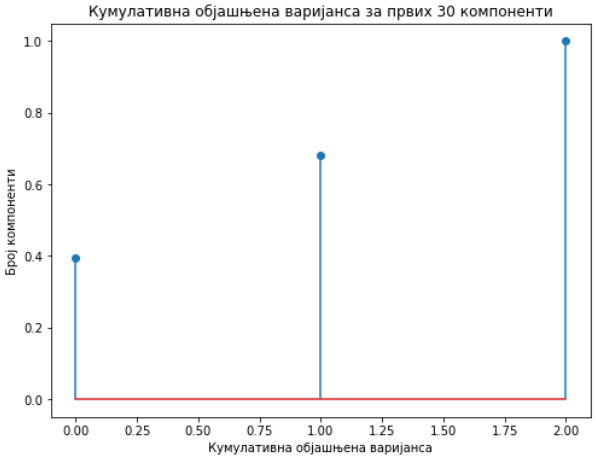


(б) Кумулативна објашњена варијанса анализе главних компоненти С-криве

Слика 9 Скуп података С-крива

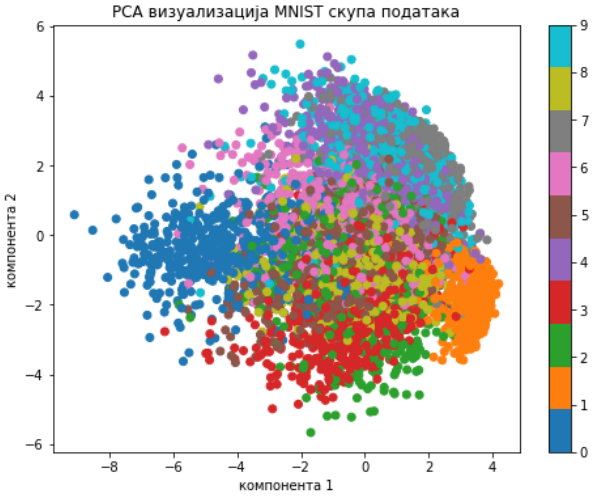


(а) Резултати анализе главних компоненти швајцарске ролнице

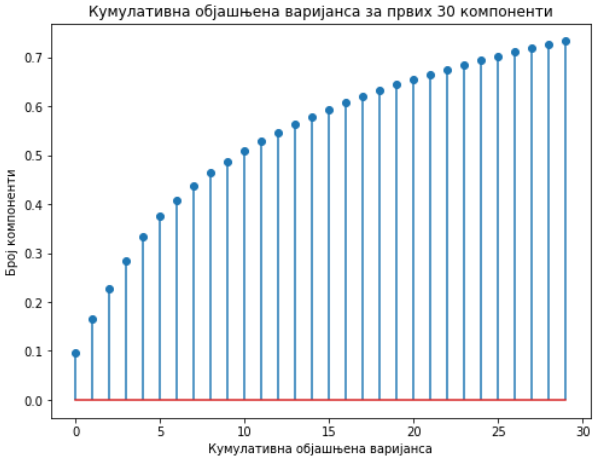


(б) Кумулативна објашњена варијанса анализе главних компоненти швајцарске ролнице

Слика 10 Скуп података Швајцарска ролница



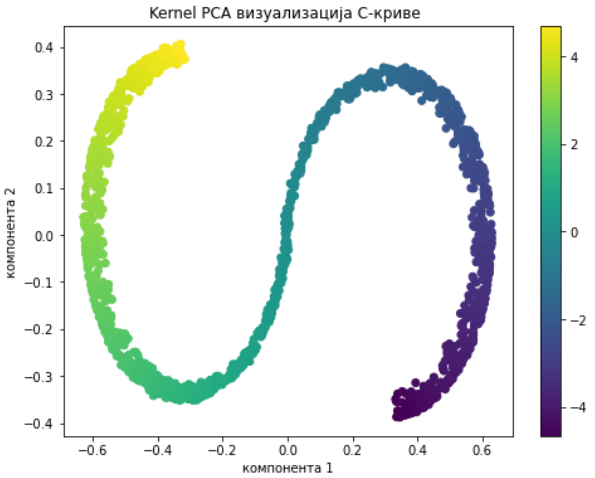
(а) Резултати анализе главних компоненти *MNIST* скупа података



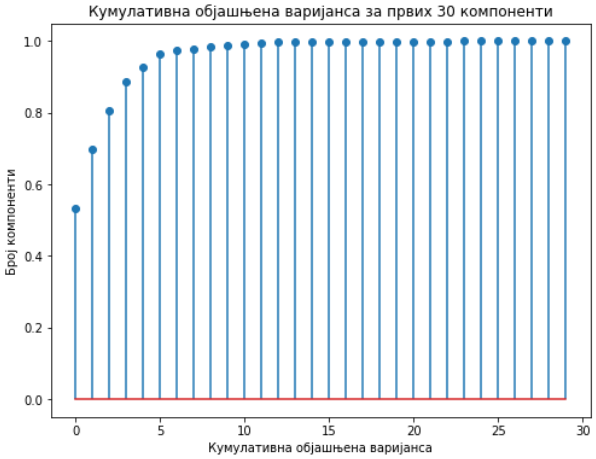
(б) Кумулативна објашњена варијанса анализе главних компоненти *MNIST* скупа података

Слика 11 *MNIST* скуп података

9.1.2 Kernel PCA

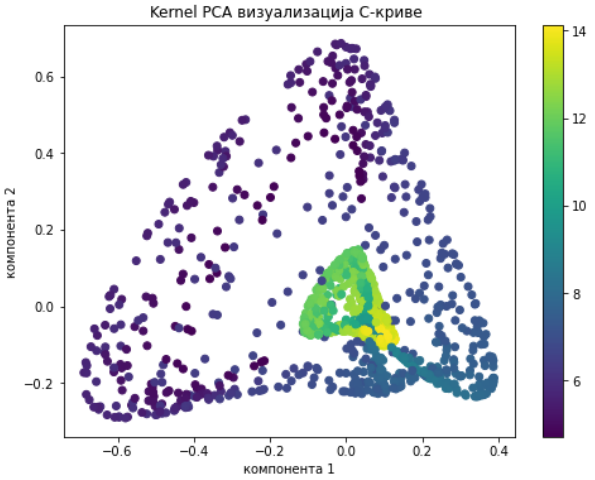


(а) Изглед C-криве након анализе главних компоненти са кернелом

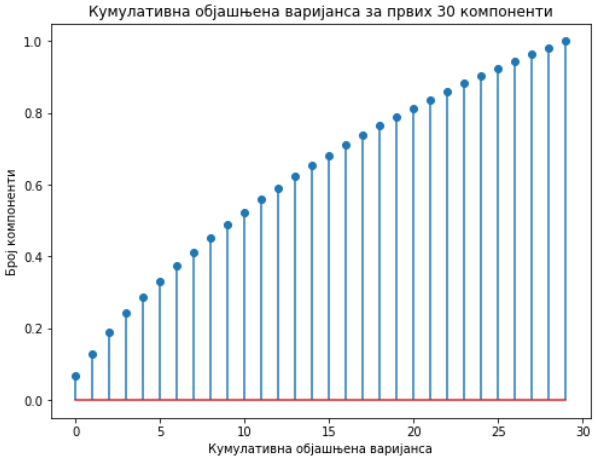


(б) Кумулативна објашњена варијанса анализе главних компоненти са кернелом C-криве

Слика 12 Скуп података C-крива

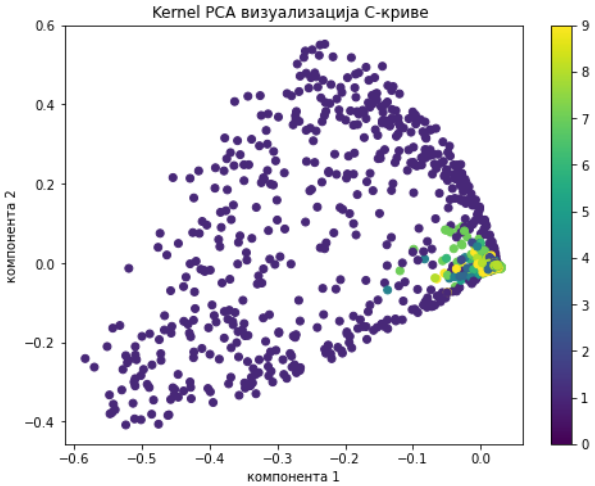


(а) Резултати анализе главних компоненти са кернелом швајцарске ролнице

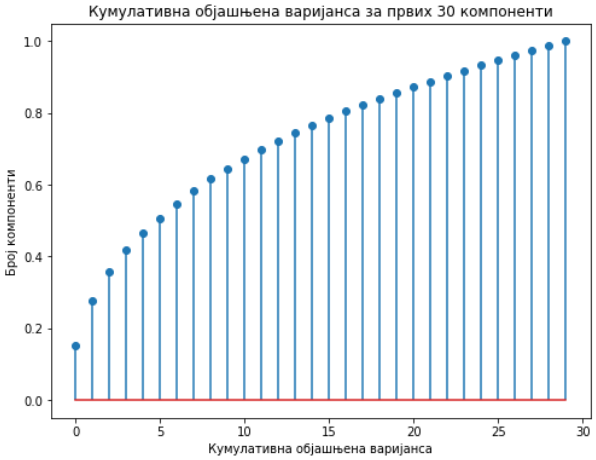


(б) Кумулативна објашњена варијанса анализе главних компоненти са кернелом швајцарске ролнице

Слика 13 Скуп података швајцарска ролница



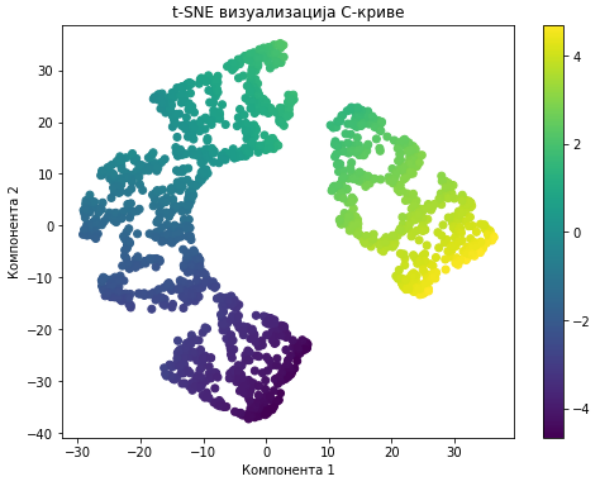
(а) Резултати анализе главних компоненти са кернелом за *MNIST* скуп података



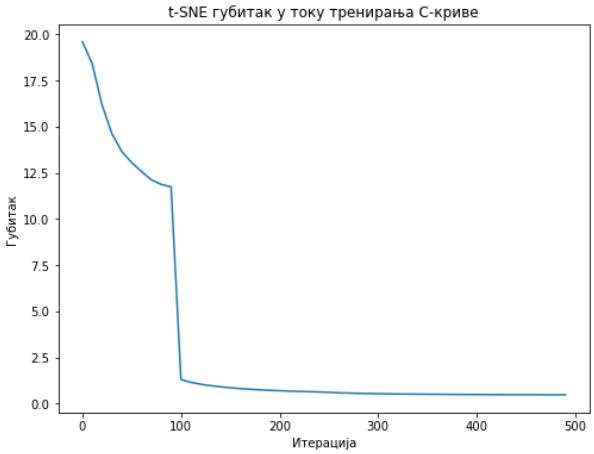
(б) Кумулативна објашњена варијанса анализе главних компоненти са кернелом *MNIST* скупа података

Слика 14 *MNIST* скуп података

9.1.3 TSNE

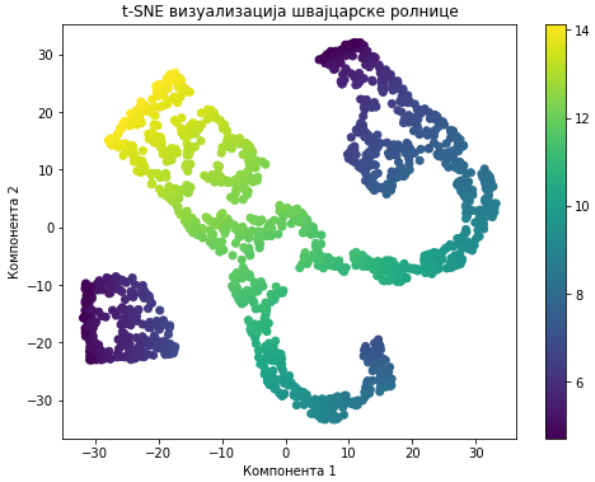


(а) Резултати *TSNE* алгоритма примењеног над С кривом

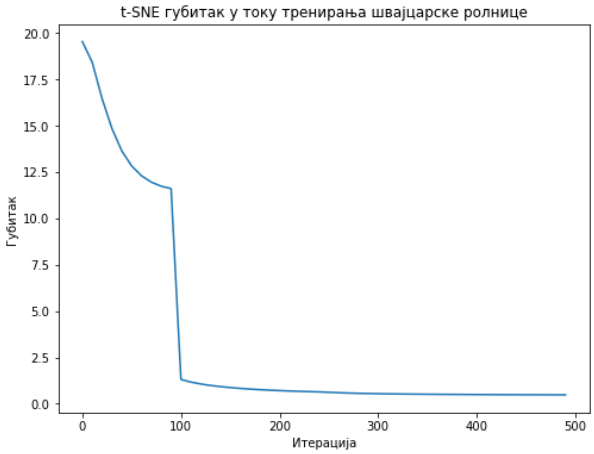


(б) Губитак *TSNE* алгоритма примењеног над С кривом

Слика 15 Скуп података С-крива

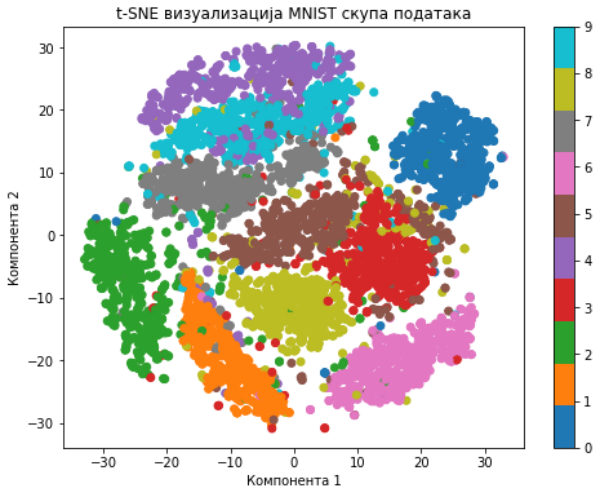


(а) Резултати *TSNE* алгоритма примењеног над швајцарском ролницом

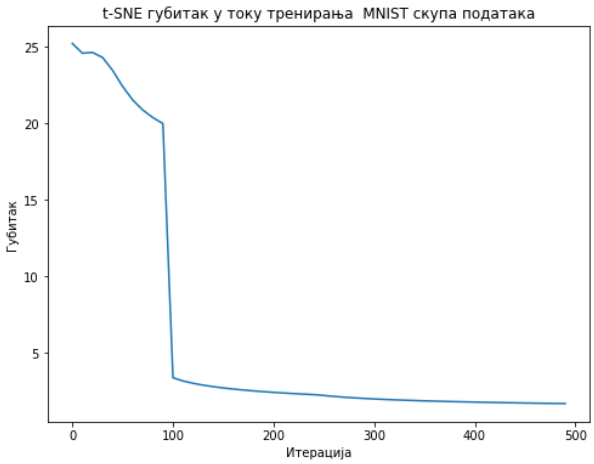


(б) Губитак *TSNE* алгоритма примењеног над швајцарском ролницом

Слика 16 Скуп података Швајцарска ролница



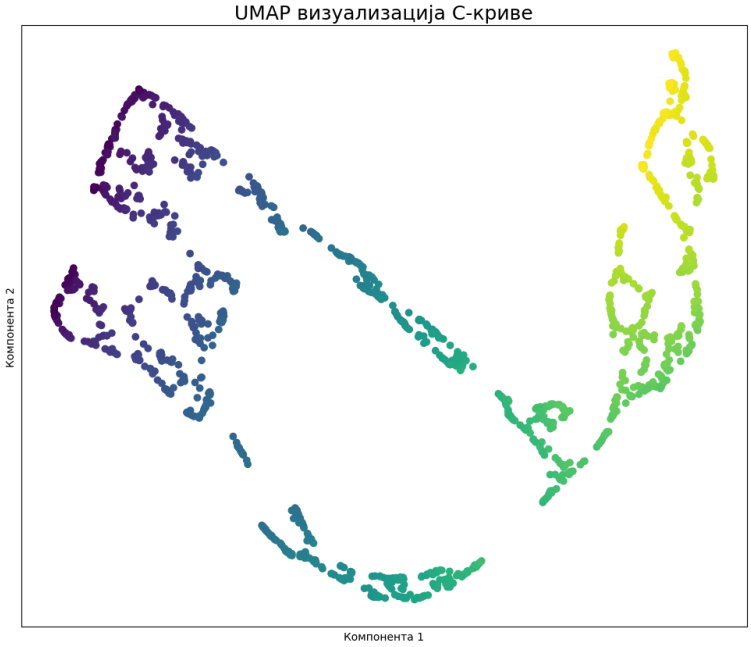
(а) Резултати *TSNE* алгоритма примењеног над *MNIST* скупом података



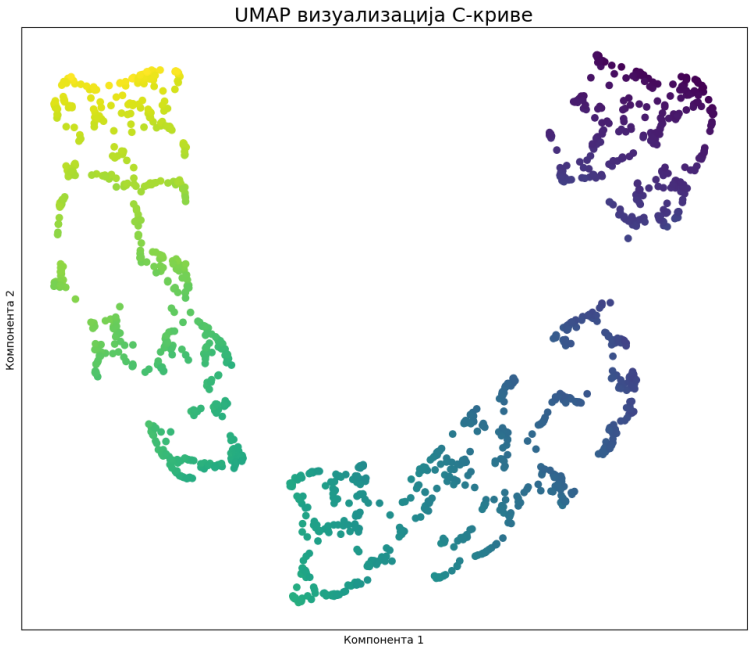
(б) Губитак *TSNE* алгоритма примењеног над *MNIST* скупом података

Слика 17 *MNIST* скуп података

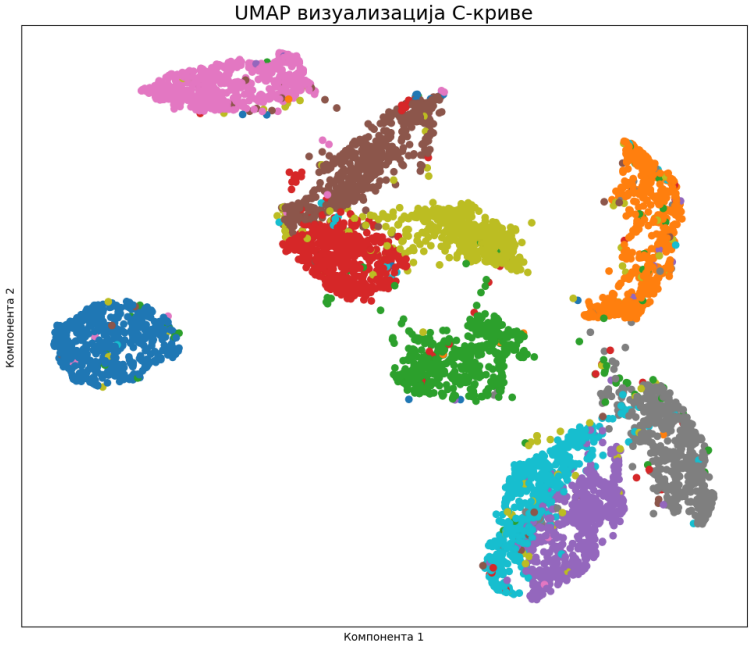
9.1.4 UMAP



Слика 18 Резултати *UMAP* алгоритма примењеног над швајцарском ролницом



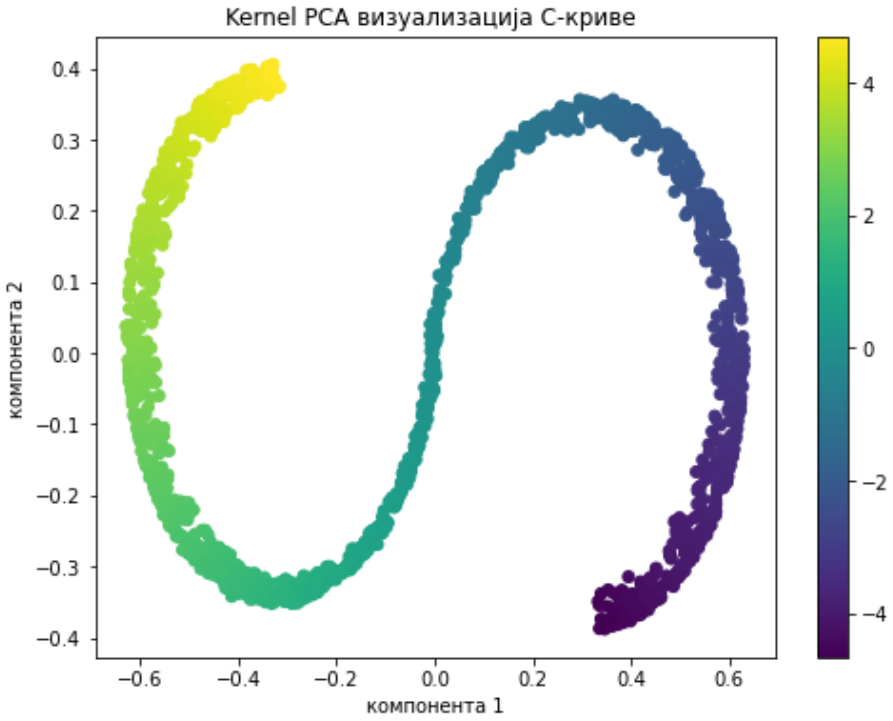
Слика 19 Резултати *UMAP* алгоритма примењеног над C кривом



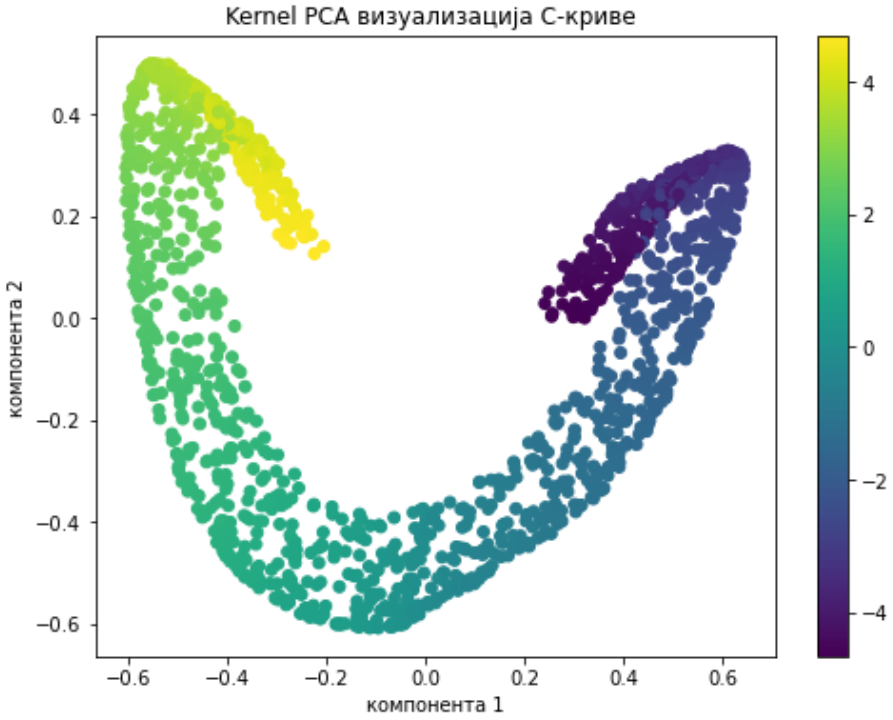
Слика 20 Резултати *UMAP* алгоритма примењеног над *MNIST* скупом података

9.2 Кернел PCA за различите вредности стандардне девијације

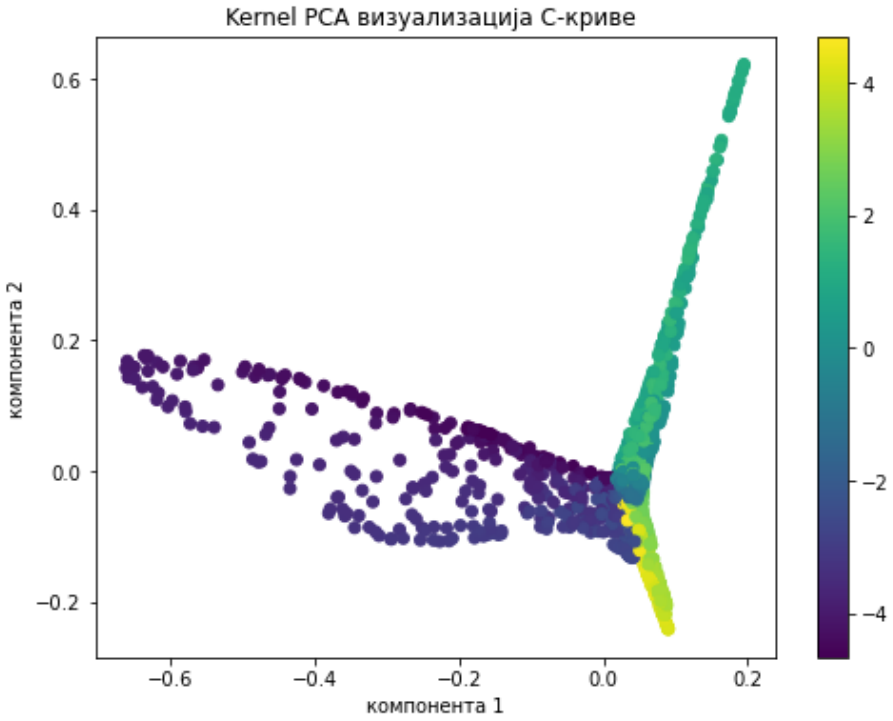
У овом поглављу разматране су различите вредности стандардних девијација за *Kernel PCA* метод.



Слика 21 Изглед C-криве након анализе главних компоненти са кернелом варијансе 5



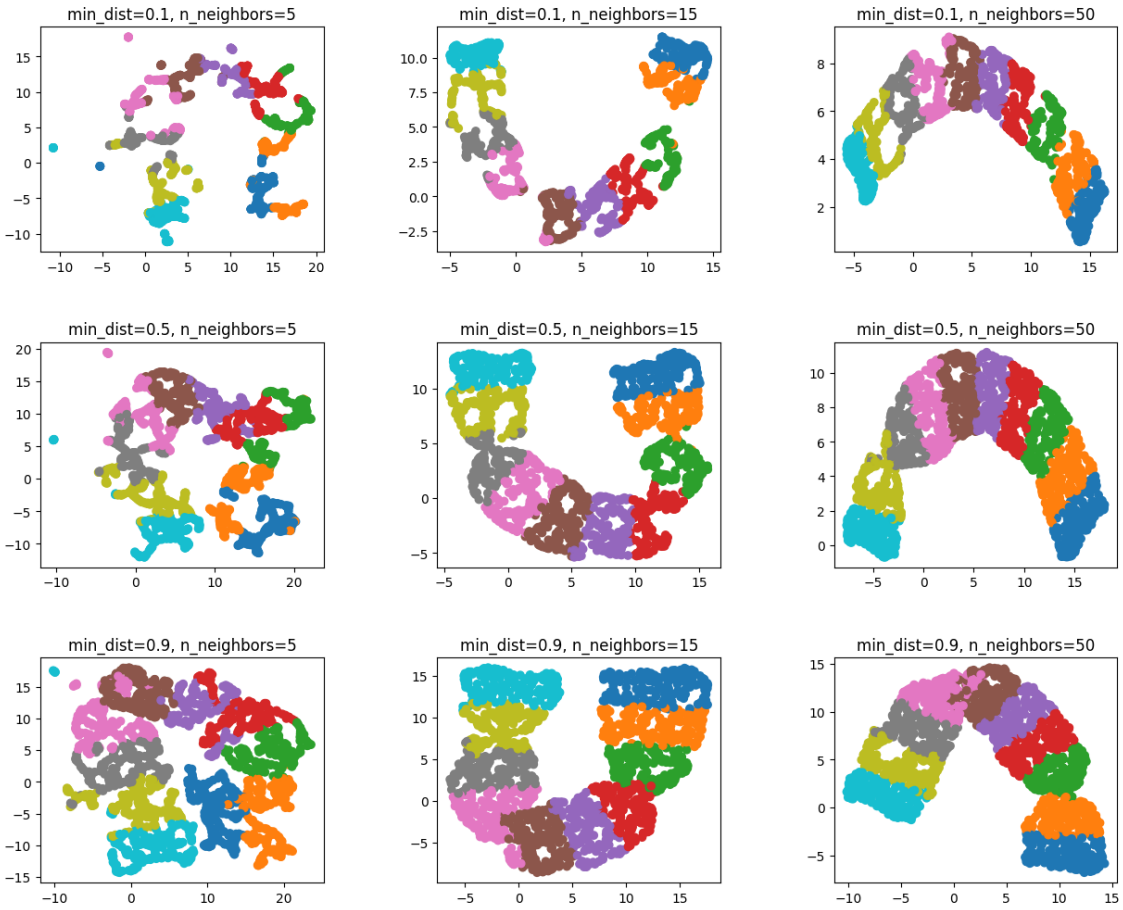
Слика 22 Изглед C-криве након анализе главних компоненти са кернелом варијансе 0.5



Слика 23 Изглед C-криве након анализе главних компоненти са кернелом варијансе 0.05

9.3 UMAP за различите хиперпараметре *min_ dist* и *n_ neighbors*

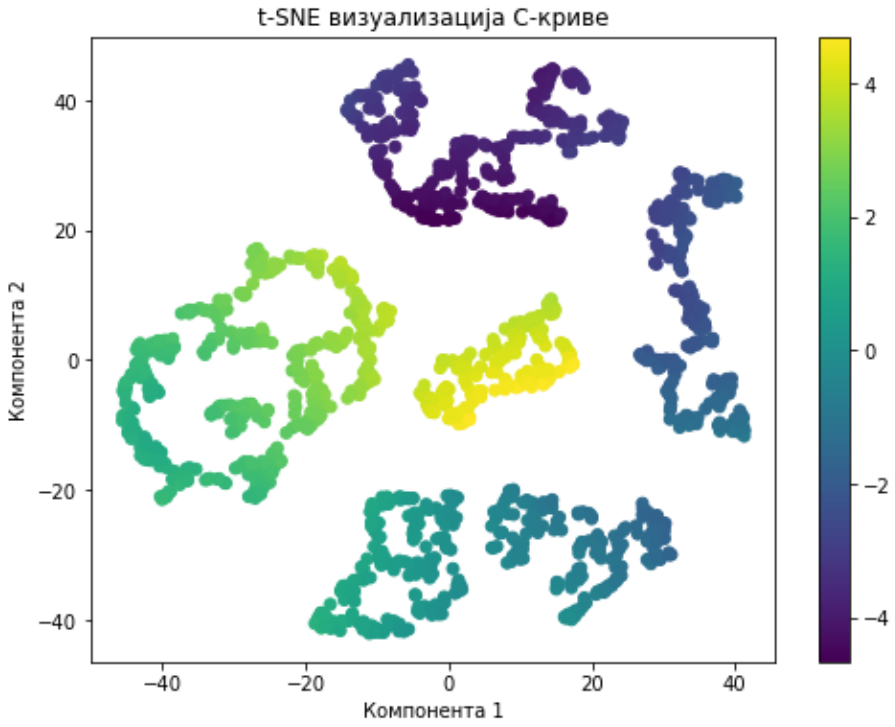
У овом поглављу приказани су резултати *UMAP* метода за различите вредности минималне дистанце и најближег броја суседа.



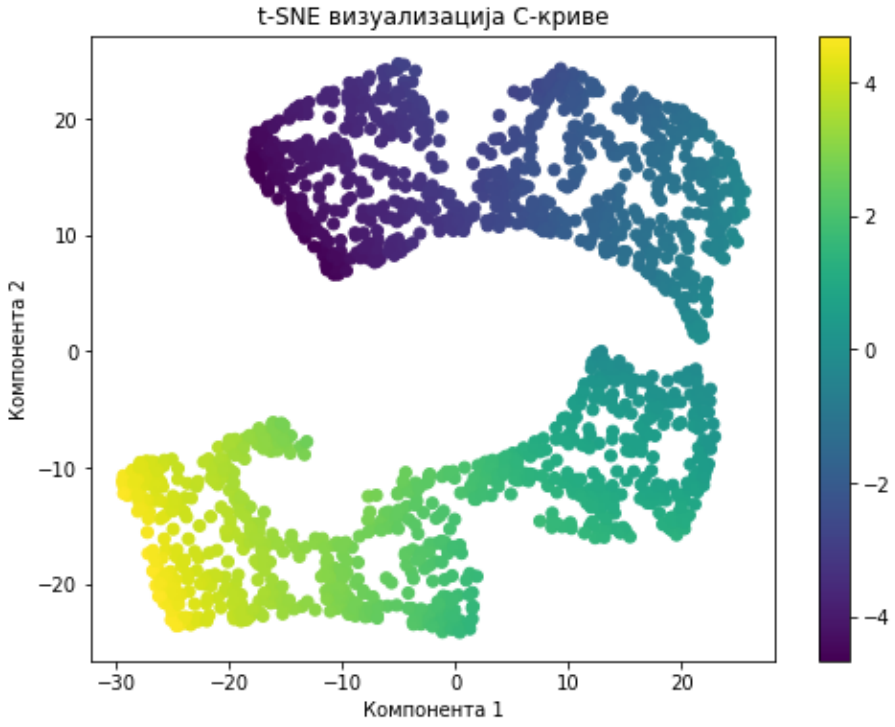
Слика 24 Резултати *UMAP* алгоритма примењеног над *MNIST* скупом података за различите хиперпараметре

9.4 Зависност од различитог ефективног броја суседа

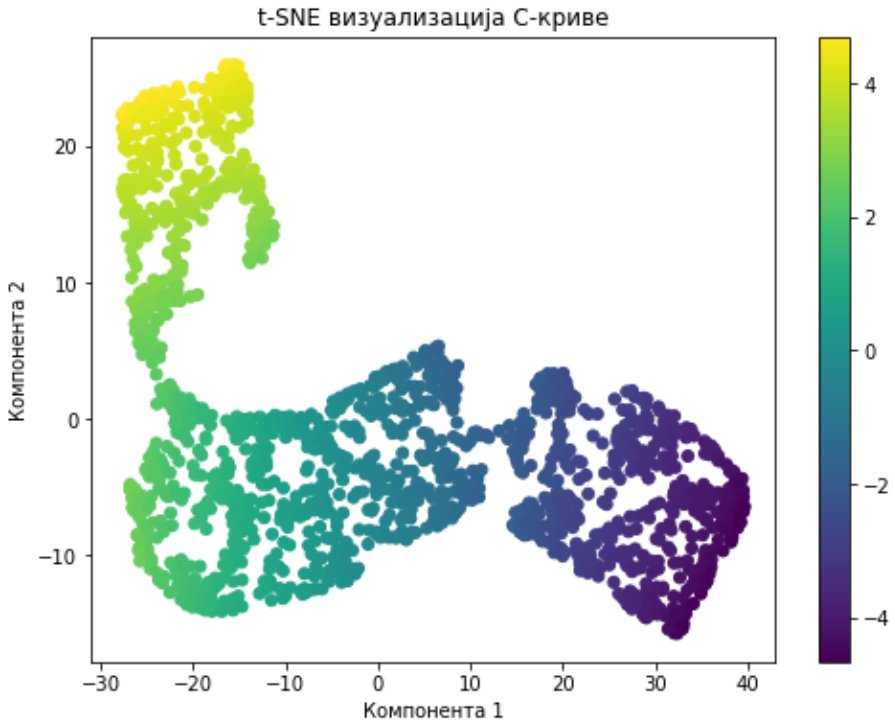
У овом поглављу приказани су резултати *TSNE* алгоритма за различите вредности броја најближих суседа.



Слика 25 Резултати *TSNE* алгоритма примењеног над C кривом за *Perplexity* = 20



Слика 26 Резултати *TSNE* алгоритма примењеног над C кривом за *Perplexity* = 40



Слика 27 Резултати *TSNE* алгоритма примењеног над C кривом за *Perplexity* = 60

10 Дискусија

Један од главних проблема је квадратна компјутерска сложеност *t-SNE* алгоритма због које није могуће редуковати цео скуп података већ само одређен део скупа који може репрезентативно да представи цео скуп података али овиме губимо доста битних информација које садрже остале слике из скупа. Из тог разлога резултати који су добијени нису идентични неким од резултата из других радова јер је на пример за *UMAP* алгоритам у оригиналном раду коришћен цео скуп података. Ипак, добијени резултати су идаље довољно репрезентативни и показују битне разлике између анализираних метода.
На сликама 9а, 10а, 11а приказани су резултати алгоритма анализе главних компоненти док су на сликама 9б, 10б, 11б приказане кумулативне суме варијансе за сваки од скупова података. Може се приметити да је за два простија скупа података потребно само 3 компоненте да би се очувале све информације што је и очекивано с обзиром да су подаци тридимензионални. За *MNIST* скуп података, потребно је свих 30 компоненти, а чак и тада кумулативна сума је само 0.7.
На сликама 12а, 13а,14а приказани су резултати метода налажења главних компоненти који користи кернел, а на сликама 12б, 13б, 14б приказане су кумулативне суме. Приметимо да овај алгоритам даје доста сложеније резултате у случају прва два скупа података али за *MNIST* скуп података успева да обухвати све информације са 30 компоненти, а кумулативну суму већу од 0.7 добија већ са између 15 и 20 компоненти. Ипак, када прикажемо информације у простору преко прве две компоненте, видимо да већи део простора обухвата једна класа података.
На сликама 15а, 16а, 17а приказани су резултати *TSNE* алгоритма, а на сликама 15б, 16б, 17б приказане су функције губитка у току тренирања. Приметити да за сва три скупа постоји исто понашање губитка, и да су све три успеле да конвергирају. Такође, постоји моменат наглог пада након 100 итерација. Подаци су за сва три скупа успешно кластеризовани тако да подаци који су били близу једни другима у оригиналном простору, остали су близу једни другима и у трансформисаном простору. Приметити да ово важи не само појединачне податке међусобно, већ и за кластере. Такође, приметити да у случају *MNIST* скупа података, алгоритам групише бројеве који се слично пишу.
На сликама 19, 18, 20 приказани су резултати *UMAP* алгоритма. Може се приметити да су исти бројеви кластеризовани на сличан начин као *TSNE* алгоритмом, али да су овог пута кластери на значајно већем међусобном одстојању. Такође, групе кластера које су сличне су остале близу међусобно, док су групе кластера које су доста различите додатно удаљене.
На сликама 21, 22, 23 упоређен је рад *Kernel PCA* алгоритма за различите вредности стандардне девијације кернела. Приметити да се за малу вредност варијансе, када је кернел униформнији, добија скоро исти резултат као са линеарним кернелом. Када је вредност варијансе доста велика, кернел тежи нули у доста тачака и онда остаје само пар компоненти, док када је за кернел изабрана добра вредност варијансе, добијамо прелаз итмеђу боја у мало растегнутом облику (неке класе заузимају већи део простора од других).
На слици 24 приказано је поређење *UMAP* алгоритма за различите вредности хипер-параметара. Видимо да када је ограничење дистанце мање, тачке су у трансформисаном простору ограничене да буду што ближе једна другој ако је њихово растојање мање од ове вредности, овиме се тачке које су довољно сличне привлаче заједно у густ кластер. Када је вредност овог хиперпараметра велика, превише дистанци ће бити мање од ове вредности и добићемо густо збијене тачке. Када је вредност овог параметра премала може да се деси да кластери не буду добро формирани јер ће се један кластер приказати у форми више делова. Када је вредност овог параметра добро изабрана, добиће се јасно груписани кластери који су довољно удаљени једни од других. Други хиперпараметар је број суседа који учествује у формирању графа. Слично као за минималну дистанцу, ако је овај хиперпараметар превелики превише тачака ће бити детектовано за суседа и добићемо превелике кластере, а за премале вредности може да се деси да не оформимо кластер уопште.
На сликама 25, 26 и 27 приказани су резултати *TSNE* алгоритма за различите вредности глатке процене броја најближих суседа P. Видимо да слично као за претходни алгоритам, за P = 20 добијамо неформиране кластере, за P= 40 добијамо очекивано груписање док за превелику вредност добијемо испресавијан кластер јер се вредности из различитих кластера више привлаче.

11 Закључак

Можемо закључити да избор модела за редукцију димензија значајно зависи од контекста података и шта нам је циљ у њима да очувамо. Ипак, *UMAP* метод значајно превазилази остале методе по питању како компјутерске комплексности тако и количине битних података која остаје очувана након редукције димензија. Такође, ово је метод који има довољно мали број хиперпараметара који су интуитивни и могу се лако подасити да метод да довољно добре резултате. Може се користити за визуелизацију података као и редукцију на више од две димензије. Због свих наведених предности, овај метод се показао као најбољи у општем случају од свих разматраних метода. На основу добијених резултата, такође можемо закључити да поред тога што се овај метод користи као први корак у претпроцесирању и визуализацији података, он често може служити и као метод за кластеризацију податка, а самим тим има и јако широк спектар примена. Једна од практичних примена где овај алгоритам даје добре резултате је кластеризовање података везаних за рак дојке у подкатегорије ове болести као што се може прочитати у чланку [10].

12 Извори података

1. Sch Olkopf, B.; Smola, A.; Robert, K. Kernel Principal Component Analysis. Доступно на линку
2. Cheng S. Kernel PCA. Medium. 2020. Доступно на линку
3. Kernel PCA — scikit-learn 0.20.3 documentation. Scikit-learn.org. 2018. Доступно на линку
4. Hinton G, Roweis S. Stochastic Neighbor Embedding. [cited 2024 Jul 8]. Доступно на линку
5. Com L, Hinton G. Visualizing Data using t-SNE Laurens van der Maaten. Journal of Machine Learning Research. 2008. Доступно на линку
6. McInnes, L.; Healy, J.; Melville, J. UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction. Доступно на линку
7. Coenen, A.; Pearce, A. Understanding UMAP. Доступно на линку
8. How UMAP Works — umap 0.5 documentation. umap-learn.readthedocs.io. Доступно на линку
9. Asymmetry of costs in (t-)SNE. ro-che.info. Доступно на линку
10. Bartoschek M, Oskolkov N, Bocci M, Lövrot J, Larsson C, Sommarin M, et al. Spatially and functionally distinct subclasses of breast cancer-associated fibroblasts revealed by single cell RNA sequencing. Nature Communications. 2018 Dec 4 [cited 2021 Oct 23];9:5150. Доступно на линку