Вежбе 5

Увод у невођено учење: Кластеризација

Напомена. Неке од слика сам преузео из скрипте др Младена Николића и Анђелке Ковачевић, али сам те исте слике налазио по интернету, те им је тешко ући у траг, јер су их и они однекле преузели.

До сада смо се сретали само са примерима вођеног учења. То је значило да су наши подаци били означени (енг. *labeled*), тј. за дате податке знали смо вредност зависне променљиве Y. На основу познатог узорка правили смо модел, који смо после користили на новим подацима.

Код невођеног учења не постоји зависна променљива Y. Просто су нам дати подаци и ми сами треба да одлучимо шта са њима да радимо. Овде се, поред непостојања зависне променљиве, отварају и многи други проблеми:

- Како проверити тачност добијених резултата, када немамо са чиме да их упоредимо?
- Како извршити поделу скупа на валидацију и тест?
- У зависности од циља, постоји ли више (и колико тачних решења)?

Једна од основних грана невођеног учења јесте кластеризација (кластеровање).

5.1 Кластеризација: теорија

Кластеризација представља процес груписања података у групе (тзв. кластере) на тај начин да су подаци унутар кластера међусобно слични, а кластери међусобно што различитији.

ОПРЕЗ! Наш појам кластера није исти као из теорије узорака! Оно што се у машинском учењу зове кластер, у теорији узорака звало би се стратум.

Потребе за кластеризацијом су многе, а неке од њих укључују издвајање група на друштвеним мрежама зарад циљаног оглашавања, детекција малигног и бенигног ткива на рендгенским снимцима, раздвајање људи у групе на основу разних психолошких тестова итд. Такође, кластеризација се може користити у сврхе **претпроцесирања** података, тако што у првом кораку обраде података кластери бивају замењени својим представницима. Ово може бити корисно у смислу рачунске једноставности, али у зависности од ситуације може и смањити квалитет модела.

Такође, кластеризација је корисна и за **кросвалидацију**. Наиме, веома је корисно поделити податке на кластере, те затим сваки од кластера на k једнаких делова, те фолдове за k-fold правити тако

што се бира по један из сваког кластера. Тиме су фолдови учињени репрезентативнијим него што би потенцијално били случајним одабиром.

Треба напоменути и да кластеризација није јединствена, јер увек можемо разматрати грубље, односно финије поделе. Већина алгоритама за кластеризацију омогућава подешавање („штеловање'') финоће самог процеса.

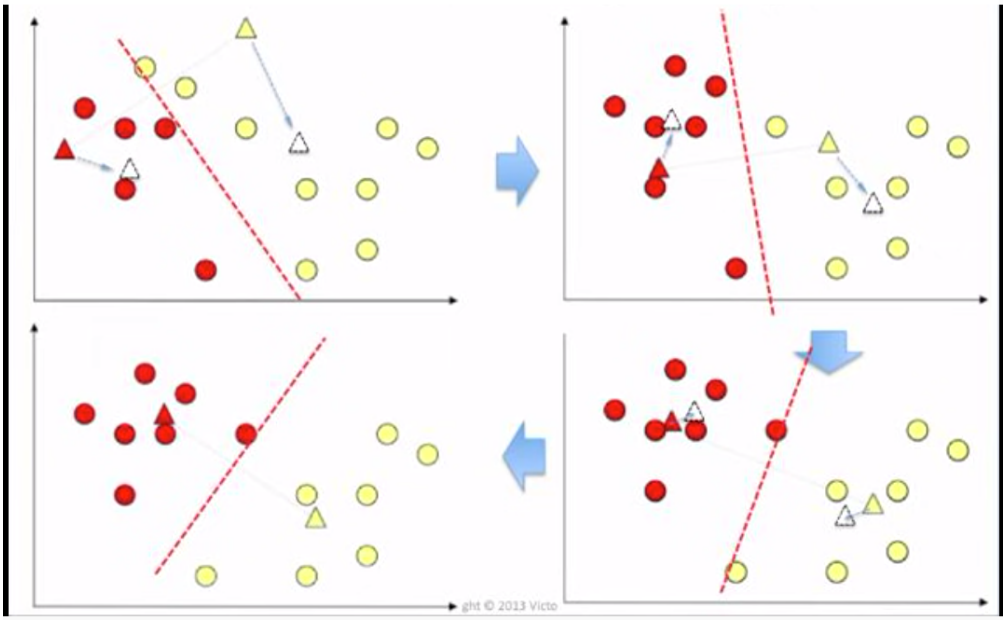
У вези са претходним разликујемо и различите врсте кластера: *Глобуларни* или *центрички* кластери јесу они који попуњавају унутрашњост лопте или евентуално елипсоида (у одговарајућем броју димензија, не мора три). Кластери су *добро раздвојени* уколико је свака тачка неког кластера ближа свим тачкама свог кластера неголи било којој тачки неког другог кластера. Кластери су *хијерархијски*, уколико у оквиру кластера можемо уочавати кластере.

5.1.1 *K*-means кластеризација

Један од најједноставнијих типова кластеризације јесте K-means кластеризација, или, на српском, кластеризација K-средина. Сада ћемо изложити како алгоритам ради.

За почетак, бира се број кластера, K. Очекује се да овај број буде процењен на основу претходних сазнања о подацима. Бира се K такозваних *центроида*, односно центара кластера, на случајан начин. Ако се зна нешто више о подацима, овај одабир свакако не мора бити случајан, и пожељније је да не буде. Затим се понављају следећи кораци:

1. За сваку опсервацију рачунамо којој је центроиди најближа. Када срачунамо, доделимо јој кластер који одговара тој центроиди.
2. Рачунамо центар новодобијеног кластера и у њега премештамо центроиду.



Претходна два корака понављамо све док процес не исконвергира. У вези са K-means кластеризацијом треба напоменути пар ствари:

- **Одабир метрике**. Особина „бити најближа центроиди'' коју има нека опсервација зависи од одабира метрике. За нумеричка обележја, јер таква морају да буду пошто рачунамо просеке, је то углавном еуклидска метрика, или, евентуално, нека од l^p метрика. За одабир еуклидске

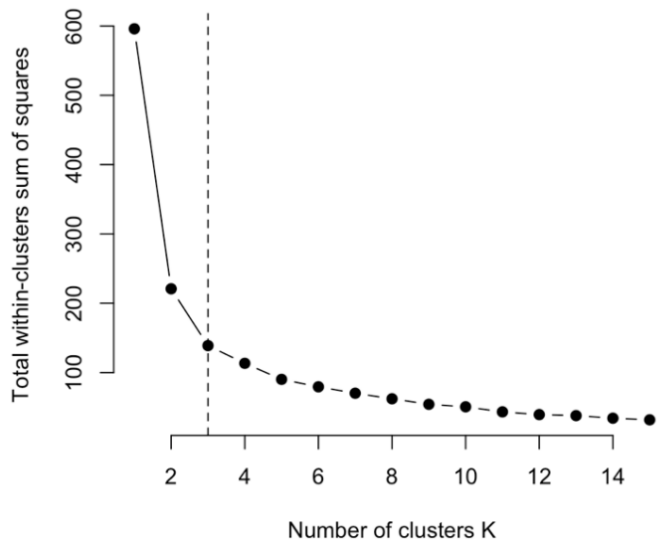
метрике d, може се показати да k-means кластеризација заправо минимизује функцију

$$\sum_{i=1}^{K} \sum_{x \in C_i} d(x, c_i)^2,$$

где је C_i кластер а c_i центроида. Минимизација наравно иде по центроидама.

Како је у овом случају алгоритам заснован на минимизовању еуклдских растојања, он тежи проналаску кластера у облику лопте.

- **Кластеровање, макар и за фиксно K, не мора да буде јединствено.** На пример, довољно је да имамо податке униформно распоређене унутар круга и да их треба поделити на два кластера. Такође, горња сума може имати локалне минимуме, који су већи од глобалног, и алгоритам у њима може да „заглави". Због тога се у пракси кластеризација понавља више пута, са различитим почетним одабиром центроида, те се гледа шта буде у највећем броју случајева.
- **Одабир броја K.** Веома често није јасно колико кластера се очекује, те се не зна које K проследити алгоритму. Једно од правила јесте и *„правило лакта"*. Оно подразумева да се за различите вредности броја K изврши кластеризација, те се у сваком од тих случајева срачуна унутаркластерна сума квадрата растојања, а затим се нацрта зависност збира тих сума у односу на K, као на слици испод.



Бирамо оно K које је „на лакту", јер након њега видимо заравњење у суми квадрата, што значи да је додавање нових кластера скоро па никако не смањује.

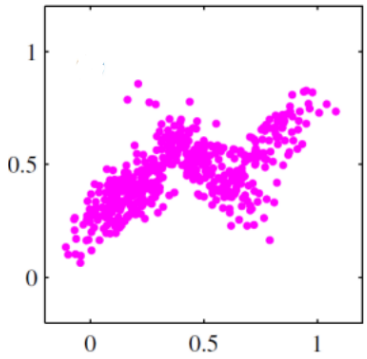
5.1.2 Мешавина нормалних расподела и EM алгоритам

Предност алгоритма K-means свакако јесте његова једноставност. Међутим, појављују се и многи проблеми:

1. Свака опсервација била је додељена тачно једном кластеру. Међутим, постоје опсервације које су „на граници", и чије додељивање кластеру јесте толико несигурно, да се може сматрати и случајним погађањем. Желели бисмо да некако за такве опсервације **сачувамо неодређеност.**
2. Користи еуклидско растојање. Чак иако одаберемо неку од других l^p норми, облик кластера никад неће бити, рецимо, елипса. Шта ако кластер није кружан, већ елиптичан?

3. Након извршене кластеризације, исти је однос према подацима близу и далеко од центроиде кластера. А за ове ближе смо сигурнији.

4. Шта ако се кластери преклапају, као на слици испод?



Због тога ћемо изложити мало софистицираниј верзију K-means алгоритма, која је позната под називом Мешавина нормалних расподела, у комбинацији са EM алгоритмом. Користићемо скраћеницу GMM, од енглеског *Gaussian Mixture Model.* EM алгоритам би требало да је, као концепт, познат са Математичке статстике, те му овде нећемо доказивати теоријска својства, већ ћемо их само наводити по потреби.

Идеја GMM алгоритма јесте да се опсервацијама, уместо кластера коме припадају, додели **вероватноћа** припадања сваком од кластера. За почетак, треба дефинисати број кластера, C, што и овде радимо „од ока'' или на основу неких сазнања о подацима „са стране''. Рецимо да имамо n опсервација и d предиктора. Ако погледамо на слику изнад, делује нам да су подаци дати у виду три дводимензионе нормалне расподеле. Ако се подсетимо, густина вишедимензионе нормалне расподеле зависи од вектора просека μ и од коваријационе матрице Σ, а дата је формулом:

$$f_{\mu,\Sigma}(\mathbf{x}) = \frac{|\Sigma|^{-1/2}}{(2\pi)^{d/2}} \exp\left(-(\mathbf{x}-\mu)^T\Sigma^{-1}(\mathbf{x}-\mu)\right).$$

Сада је природно задати густину свих наших података као неку **конвексну** комбинацију густина појединачних кластера:

$$f(\mathbf{x}) = \sum_{c=1}^{C} \pi_c f_{\mu_c,\Sigma_c}(\mathbf{x}).$$

Идеја је да се крене од произвољних параметара π_c, μ_c и Σ_c, те да се они итеративно „побољшавају''. Једну илустрацију густине f можемо видети на слици испод.

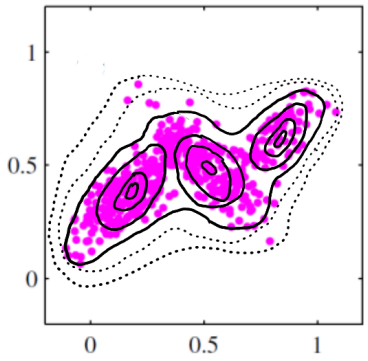
Густина f може се схватити и на следећи начин. Можемо уочити да заправо постоји једна скривена, **латентна** случајна величина Z, која заправо представља расподелу вероватноћа π_j по кластерима:

$$Z : \begin{pmatrix} c_1 & c_2 & \cdots & c_C \\ \pi_1 & \pi_2 & \cdots & \pi_C \end{pmatrix}.$$

Сада се може записати да је

$$f(\mathbf{x} \mid Z = c) = f_{\mu_c,\Sigma_c}(\mathbf{x}).$$

Ово је згодан начин за представљање, уколико бисмо желели да генеришемо опсервацију са густином f, јер је $f(\cdot) = f(\cdot \mid Z = c)\pi_c$, а све на десној страни знамо да генеришемо.



Океј, вратимо се GMM алгоритму. Крећемо од произвољног (или не, ако знамо нешто) одабира почетних елемената μ_c, Σ_c, π_c. Затим спроводимо следеће кораке:

- **E корак.** За сваку опсервацију $\mathbf{x}^i$ рачунамо вероватноћу да припада c-том кластеру као:

$$r_{ic} = \frac{\pi_c f_{\mu_c, \Sigma_c}(\mathbf{x}^i)}{\sum_{c'=1}^{C} \pi_{c'} f_{\mu_{c'}, \Sigma_{c'}}(\mathbf{x}^i)}.$$

- **M корак.** На основу срачунате вероватноће припадања, ажурирамо информације о кластерима. Прво ћемо да рачунамо n_c које представља суму свих вероватноћа припадања за c-ти кластер:

$$n_c = \sum_{i=1}^{n} r_{ic}.$$

  Ажурирану вредност за π_c сада добијамо као количник n_c и n:

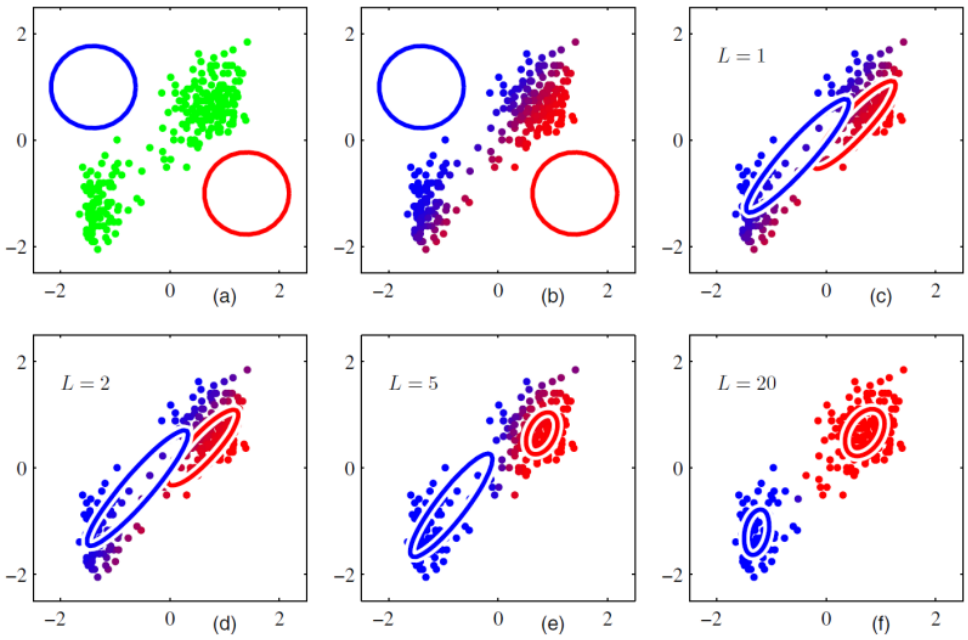
$$\pi_c = \frac{n_c}{n}.$$

  Она се ажурира на овај начин јер у неком смислу представља „тежину" c-тог кластера у заједничкој суми. Средњу вредност и коваријациону матрицу сада, природно, рачунамо као тежинске средине:

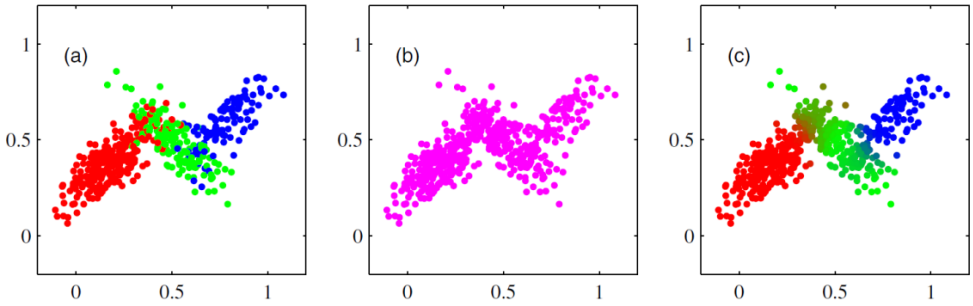
$$\mu_c = \sum_{i=1}^{n} \frac{r_{ic}}{n_c} \mathbf{x}^i, \quad \Sigma_c = \sum_{i=1}^{n} \frac{r_{ic}}{n_c} (\mathbf{x}^i - \mu_c)(\mathbf{x}^i - \mu_c)^T.$$

Напомена. Може се доказати (доказ у [23]) да свака итерација повећава функцију веродостојности, те овај алгоритам сигурно конвергира. Међутим, он може исконвергирати локалном, а не глобалном максимуму. Стога се и овде препоручује „пуштање'' алгоритма више пута, са различитим почетним параметрима.

Илустрација GMM+EM за два кластера и 20 итерација дата је на слици испод:



На наредној слици, скроз лево приказани су подаци, вештачки генерисани, онаквима какви јесу. На средњој слици видимо их обједињене. На десној слици, интензитет боје сваке тачке сразмеран је броју r_{ic}. Видимо да је на преклапању кластера боја најблеђа, и долази до преклапања боја, што осликава нашу несигурност око одабира кластера за те тачке.



Уколико ипак желимо да опсервацију доделимо неком кластеру, а не да је оставимо са вероватноћом, просто ћемо одабрати онај кластер чија је вероватноћа припадности највећа. Уколико добијемо **нову опсервацију** поступамо идентично.

За одабир броја кластера и овде важи „лакат правило''.

Уочимо још и да је k-means алгоритам заправо специјалан случај алгоритма GMM и то за

- $\Sigma_c = \sigma^2 I$ за све c;
- $\pi_c = 1/C$, за све c;
- $P(Z = c \mid \mathbf{x}^i)$ је 1 ако је $\mathbf{x}^i$ најближи просеку μ_c, а нула иначе.

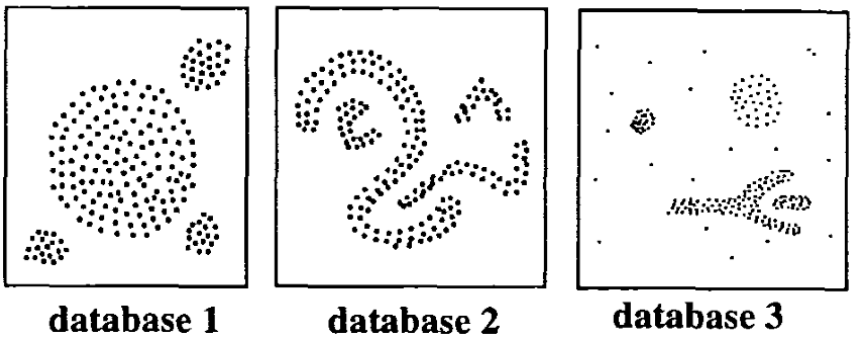
5.1.3 DBSCAN кластеризација

Напомена. Део слика преузет је из [3].

Следећи тип кластеризације са којим ћемо се упознати јесте алгоритам DBSCAN. Име му је акроним и скраћеница је од енглеског *Density-Based Spatial Clustering and Application with Noise*. Пре него што дамо детаљан опис алгоритма, мотивисаћемо.

Код алгоритма K-means успевали смо да издвајамо кластере правилног(симетричног) облика. За еуклидску метрику то су били кружни кластери, док би за неки други одабир метрике то био неки други правилан облик. Коришћењем GMM+EM успели смо да кластере које смо у стању да уочимо проширимо на елиптичне кластере.

Међутим, размотримо наредну слику.



У сва три случаја јасно су, људском оку, уочљиви кластери. Међутим, јасно је да свака од поменуте две методе неће бити у стању да их препозна на адекватан начин, јер сами клсатери нису одговарајућег облика (осим на првој слици).

Идеја DBSCAN алгоритма јесте да кластере замишљамо као области велике густине, а да их раздвајамо областима мање густине. Два основна параметра у вези са овим јесу

- ε (епсилон) - параметар који дефинише полупречник, такав да за неку тачку x кажемо да је лопта $B(x, \varepsilon)$ њен „комшилук", а да свака већа није њен комшилук.

  Напомена! На енглеском језику, појам се зове *neighbourhood*, што би се на српски превело као *околина*. Међутим, како овај појам не одговара тополошком појму околине, одлучио сам се да уведем термин комшилук.

- z - минималан број тачака у околини неке тачке - њених „комшија".

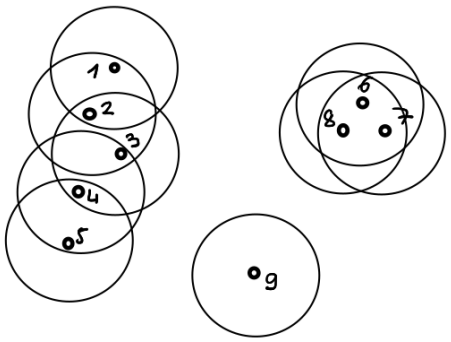
Сада ћемо дефинисати неколико појмова.

- Опсервација **x** са бројем комшија барем z (рачунајући и њу) је **главна тачка** (енг. *core point*).
- Ако је **x** таква да има мање од z комшшија, али се налази у комшилуку неке главне тачке, кажемо да је она **гранична тачка** (енг. *border point*).
- Ако **x** није ни главна ни гранична тачка, кажемо да је **тачка шума** (*noise point*).

На наредној слици дата је једна илустрација поменутих типова тачака.

Сада смо корак ближе томе да извршимо кластеризацију. Треба нам још појмова.

- Тачка (опсервација) A је **директно дохватљива** (*direct density reachable*) из тачке B ако је A у комшилуку од B и ако је B главна тачка.



$z = 3$

2, 3, 4, 6, 7, 8 - главне

1, 5 - граничне

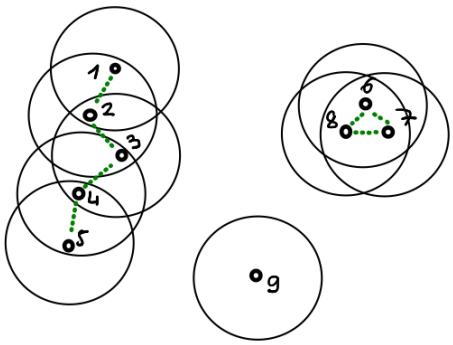
9 - шум

- Тачка A је **дохватљива** (*density reachable*) из тачке B ако постоји низ главних тачака који води од B до A.
- Тачке A и B су **повезане** (*density connected*) ако постоји главна тачка C таква да су обе из ње дохватљиве.

Сада смо коначно спремни да опишемо DBSCAN алгоритам.

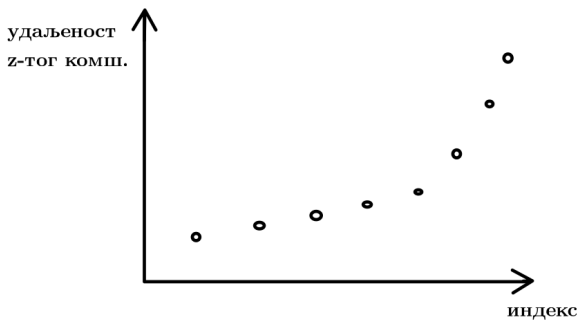
1. За сваку тачку $\mathbf{x}^i$ рачунамо њено растојање од осталих тачака. Уочавамо све тачке које су јој комшије. Сваку која има више од z комшија прогласимо за главну тачку.
2. За сваку главну тачку, ако већ није у кластеру, правимо нови кластер. У њен кластер смештамо све са њом повезане тачке.
3. Понављамо процес кроз све тачке.

Она тачка која није главна и није припала ниједном кластеру, јесте тачка шума и њу не класификујемо. На наредној слици видимо наставак примера са претходне слике. За дато $z = 3$ и одговарајуће ε алгоритам је уочио два кластера и једну тачку шума.



Неке од главних предности овог алгоритма јесу то што не захтева да му се предефинише број кластера, већ га он сам проналази. Овај алгоритам проналази кластере произвољног облика, а у стању је и да детектује аутлајере (тачке шума).

Основни недостатак јесте тај што **није јасно како одабрати** ε **и** z. Наравно да ни овде не постоји генерално правило. Пракса је показала да, уколико имамо велики број опсервација, можемо бирати z које је ред величине броја предиктора. За ε прича иде мало теже. Овде је пракса показала да можемо за сваку опсервацију срачунати њену удаљеност од z-тог комшије, затим сортирати тај низ удаљености. Идеално, добићемо график сличан оном на наредној слици.



Бирамо удаљеност оног z-тог комшије који је „на лакту''. Ако немамо „лакат", наравно, онда ништа, морамо на осећај.

5.2 Кластеризација - практично у R-у

5.2.1 K-means

Користићемо функцију `kmeans` из основног пакета `stats`. Користићемо базу података `USArrests`. Свака опсервација је једна Америчка савезна држава, а предиктора има 4: стопа убистава, стопа напада, стопа урбане популације и стопа силовања.

```
data("USArrests")
```

Одмах треба скалирати податке, јер желимо да сваки предиктор има исти ред величине, због рачунања растојања.

```
df <- data.frame(scale(USArrests))
```

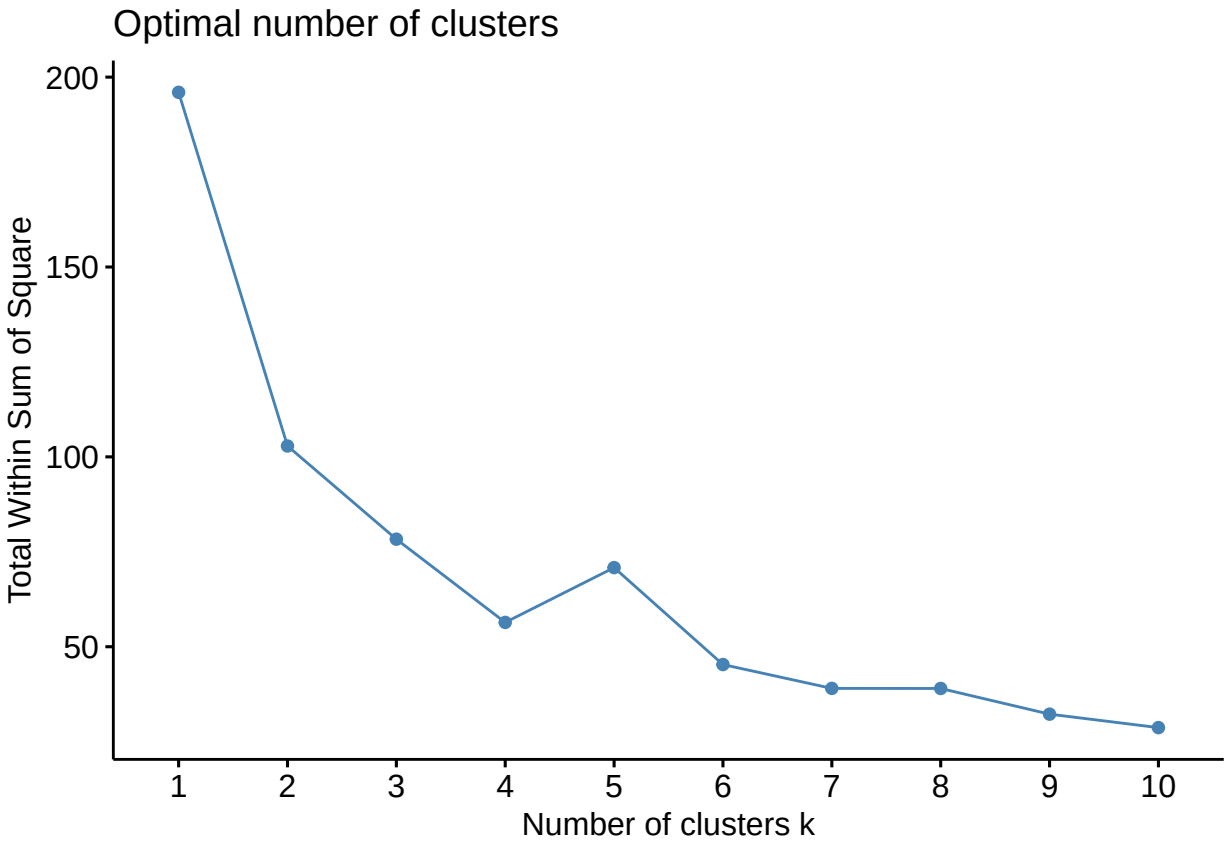
Потпис функције `kmeans` је следећи

$$\texttt{kmeans}(\texttt{x}, \texttt{centers}, \texttt{iter.max} = 10, \texttt{nstart} = 1).$$

Аргумент `x` представља базу података. Аргумент `centers` представља почетни одабир центроида, а ако му се проследи само један број онда то схвата као број кластера, а почетне центроиде бира насумично из врста од `x`. Аргумент `iter.max`, јасно, представља максималан дозвољен број итерација алгоритма, док последњи аргумент представља број почетних одабира центроида. Ако је већи од један, онда се бира она кластеризација која минимизује суму квадрата растојања од центроида, у свим кластерима ([4]). Пожељно је да овај аргумент буде већи од 1.

Користићемо и библиотеку `factoextra` због лепих графичких приказа. За почетак, треба одредити оптималан број кластера „правилом лакта''.

```
library(factoextra)
fviz_nbclust(x = df,
             method = "wss", # "within sum of squares", pogledati help za ostale)
             FUNcluster = kmeans
            )
```



Видимо да сума квадрата достиже минимум за 4 кластера, па расте за 5, те наставља да опада након 6 кластера. Овде није јасно да ли узети 4 или 6 кластера. Без неког нарочитог разлога одлучићемо се за 4 кластера, јер нам је циљ да демонстрирамо метод.

Сада спроводимо кластеризацију.

```
set.seed(123)
km_result <- kmeans(df, 4, nstart = 15)
```

То је у суштини то, још да видимо шта враћа функција `kmeans`. Она враћа листу компоненти, а у њој је

- `cluster` - вектор природних бројева дужине оригиналних података, који говори о томе ком је кластеру опсервација додељена;
- `centers` - финални вектор центроида;
- `totss` - SSTO са ЛСМ; мери укупну суму квадрата растојања свих података;
- `withinss` - вектор унутаркластерних сума квадрата растојања;
- `tot.withinss` - сума претходног вектора;
- `betweenss` - једнака `totss` - `tot.withinss`;
- `size` - вектор величина кластера.

Визуализацију ћемо оставити за након што одрадимо метод PCA.

`GMM+EM`

Пошто смо у теоријском делу видели како алгоритам ради, трудићемо се да што мање ствари програмирамо ручно, већ да користимо уграђене функције. Користимо пакет `mclust`.

```
library(mclust)
```

База коју ћемо користити јесте база `iris`, која садржи податке о цвету рода Ирис. Она има следеће колоне

```
names(iris)
```

```
## [1] "Sepal.Length" "Sepal.Width"  "Petal.Length" "Petal.Width"  "Species"
```

Прве 4 говоре о вредностима одређених димензија цвета, док последња представља једну од три врсте у оквиру рода Ирис, а то су:

```
levels(iris$Species)
```

```
## [1] "setosa"     "versicolor" "virginica"
```

Последњу нећемо користити јер представља баш кластере, које ми желимо да одредимо невођено, те је се решавамо.

```
df <- iris[, 1:4]
```

Сада све иде аутоматски, позивом функције `Mclust`. Сви параметри могу се и ручно иницијализовати, али ми ћемо оставити функцији да ради сама, битно је само да знамо да може.

```
mcl_model_iris <- Mclust(df, 3)
```

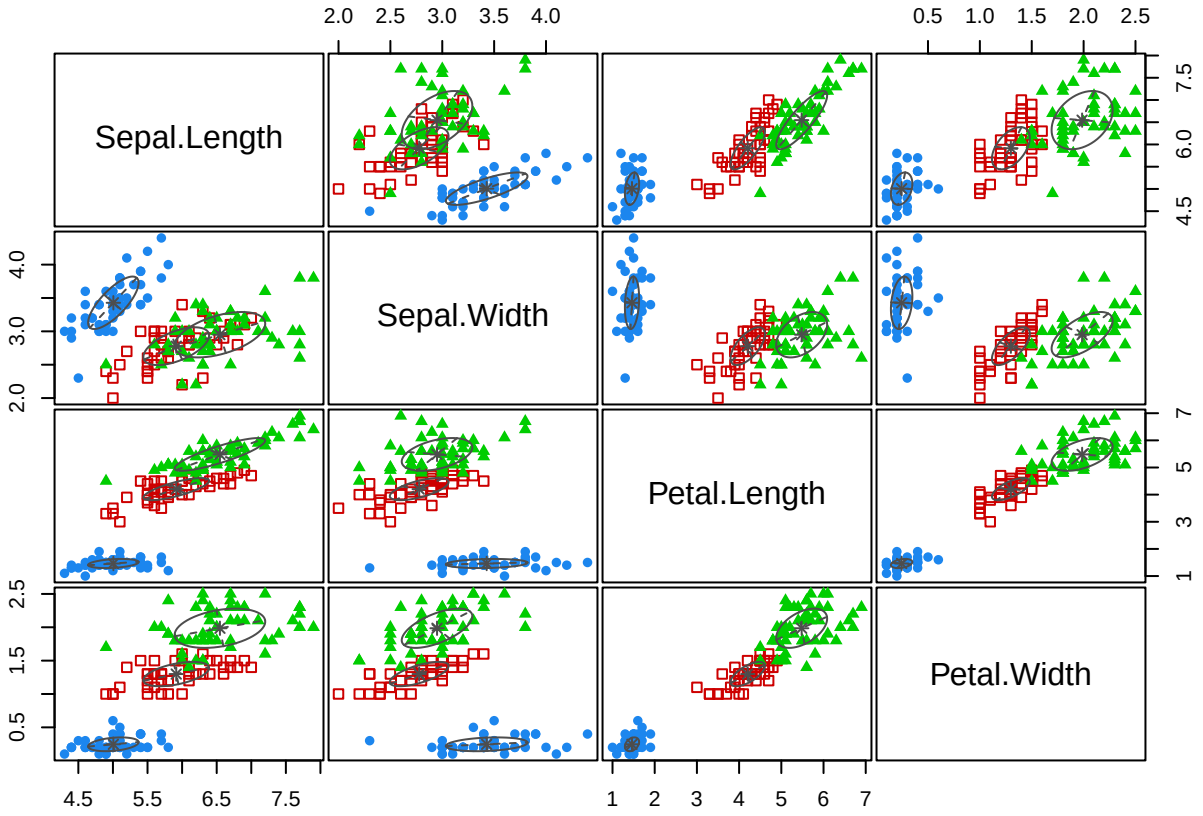
Ако позовемо `class`, добијамо

```
class(mcl_model_iris)
```

```
## [1] "Mclust"
```

Ово је заправо листа неких стандардних повратних вредности, свих оних о којима смо говорили у теоријском уводу, те се нећемо замарати свима, за то постоји `help`. Оно што је згодно јесте то што се објекат ове класе може директно проследити функцији `plot`.

```
plot(mcl_model_iris, what = "classification")
```



Јасно, на овој слици су кластери додељени по принципу највеће вероватноће, па нема флуидног прелаза боја. То се може добити играњем са `ggplot`-ом и повратним вредностима за `Mclust`.

5.2.2 DBSCAN

Овај алгоритам у R-у може бити имплементиран коришћењем два пакета: `dbscan` и `fpc`. Ми ћемо се одлучити за овај други за саму кластеризацију, док ћемо за одабир епсилона користити први. Разлика је мање-више у синтакси. Користићемо и пакет `factoextra` због визуализације кластера, уграђене базе `multishapes`, а `fpc` смо бирали између осталог јер се слаже са овим пакетом у смилсу визуализације. Прецизније, оно што враћа функција из пакета `fpc` можемо прослеђивати функцијама из `factoextra` као аргумент, док за `dbscan` то не можемо.

```
library(fpc)
library(factoextra)
library(dbscan)
```

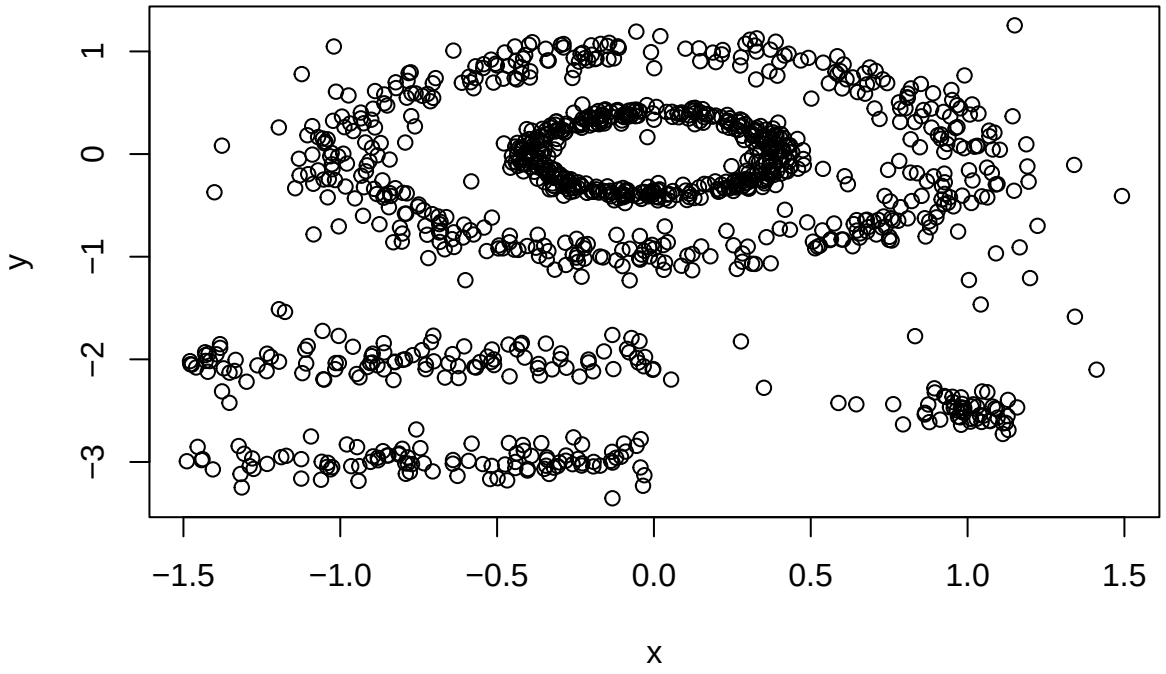
Напомена за кориснике ГНУ/Линукс дистрибуција! За инсталацију ових пакета морате имати инсталиран `gcc-fortran`.

Прво ћемо учитати податке.

```
data("multishapes", package = "factoextra")
df <- multishapes[, 1:2]
```

Узели смо прве две промењиве јер је трећа категоричка и говори о томе која је права вредност кластера у који опсервација спада (подаци су вештачки генерисани па се зна).

```
plot(df)
```



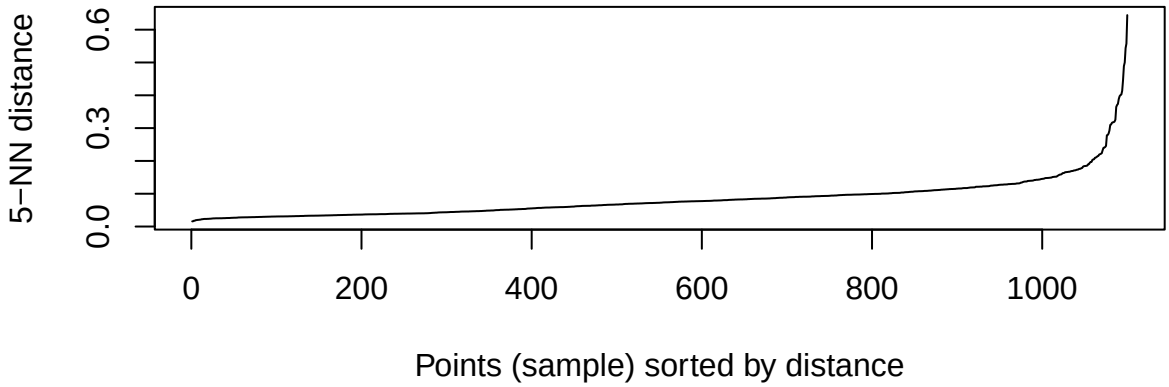
Јасно је да уочавамо 5 кластера. Након кластеровања функцијом из пакета `factoextra` направићемо лепшу слику. Сада вршимо кластеризацију помоћу пакета `fpc`.

```
set.seed(123)
db <- fpc::dbscan(df, eps = 0.15, MinPts = 5)
# class(db) враца "dbscan"
```

Број z дат је аргументом `MinPts`. Бирали смо 5, иако имамо 2 предиктора, јер бисмо спољни „прстен'' желели да класификујемо као један кластер, па за сваки случај, да не направи прекид.

Епсилон смо одабрали уз помоћ „лакат'' технике, а то се спроводи на следећи начин.

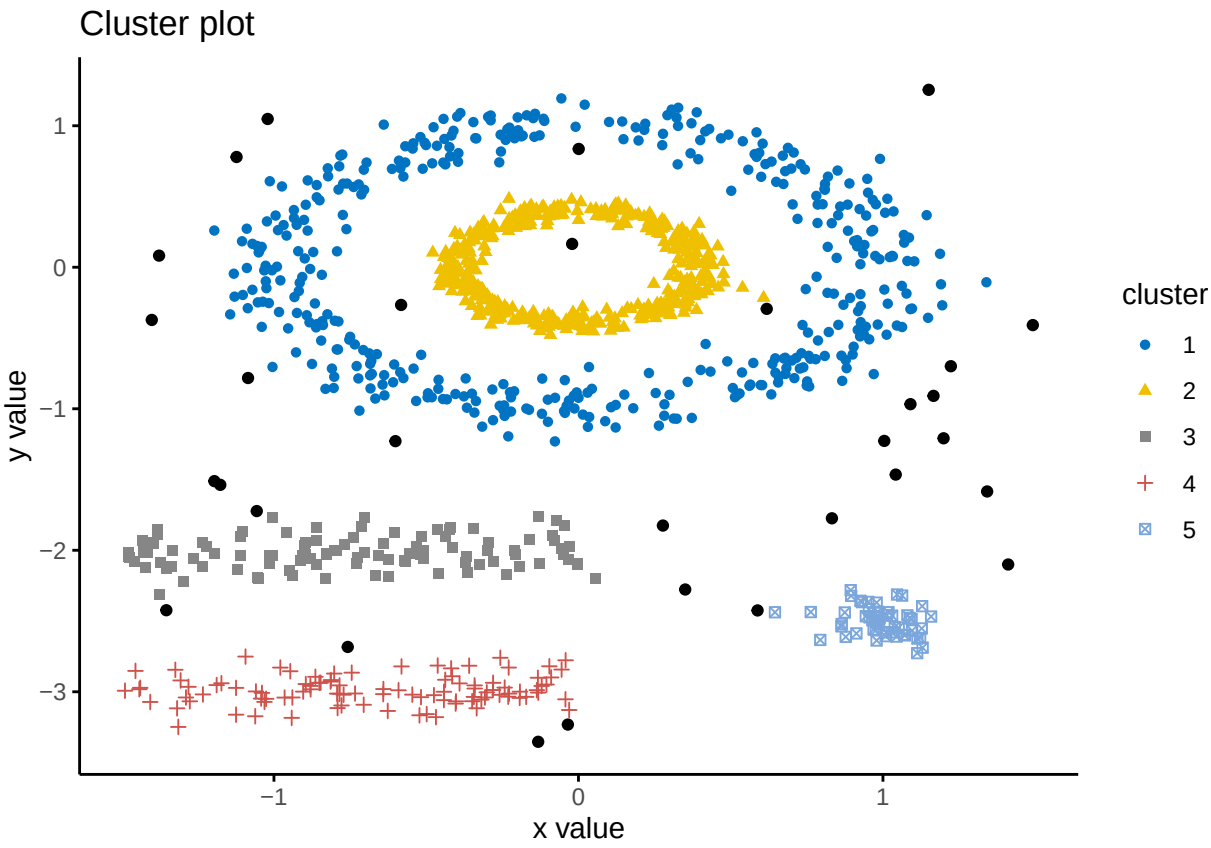
```
dbscan::kNNdistplot(df, k = 5)
```



Видимо да је 0.15 отприлике „на лакту''.

Сада цртамо лепшу слику.

```
fviz_cluster(db,
             data = df,
             stand = FALSE, # ne radimo PCA, ne treba nam (zvati help)
             ellipse = FALSE, # da ne zaokruzuje klastere
             show.clust.cent = FALSE,
             geom = "point",
             palette = "jco",
             ggtheme = theme_classic()
             )
```

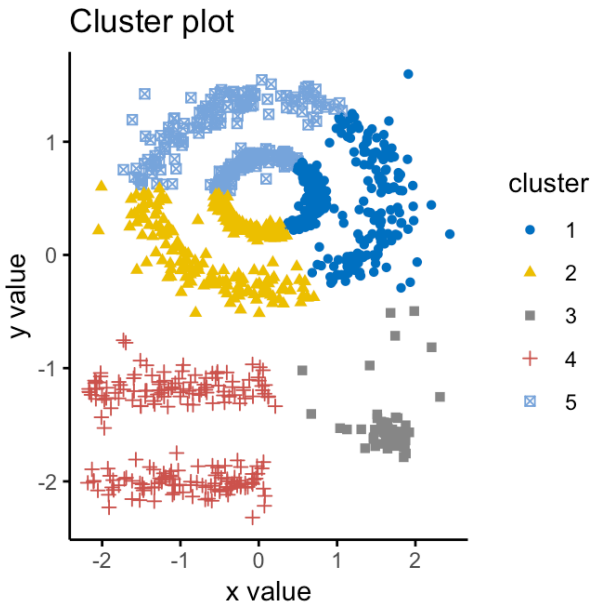


И голим оком да се наслутити да је већину аутлајера погодио како треба. Тек за неке нам је смислено да припадају неком кластеру, а да их је алгоритам прогласио аутлајерима (шумом).

Предвиђање за нове тачке врши се функцијом `predict.dbscan` из пакета `fpc`.

На наредној слици дат је пример шта се деси када исте податке покушамо да кластерујемо методом K-means, за одређене почетне центроиде (нећемо спроводити поступак, само као пример).

Није баш како треба.


Cluster plot
y value
x value
1
0
-1
-2
-2
-1
0
1
2
cluster
1
2
3
4
5