Notițe Seminar 8

November 25, 2019

Intro: Seminarul acesta vom continua cu un alt algoritm de clasificare. Vom vedea că el poate fi folosit și pentru regresie. Va fi ultimul algoritm de clasificare pe care îl vom studia. De săptămâna viitoare vom trece la învățare nesupervizată de tip clusterizare.

1 Remember

1.1 Vectori

Vom lucra cu **vectori** din $\mathbb{R}^n$, care sunt niște tuple de numere reale: $(x_1, \ldots, x_n) \in \mathbb{R}^n$. Noi vom folosi notația prin care un vector va fi o matrice cu o singură coloană: $\begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} \in \mathbb{R}^n$.

1.2 Distanțe

Dacă vă interesează definiția, vedeți ex. 2/pag. 427.

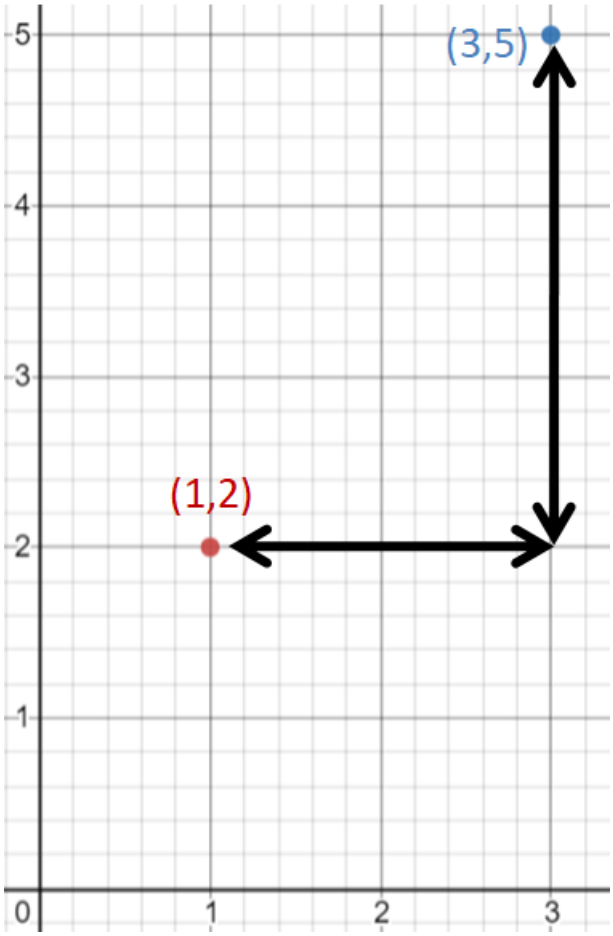
O familie de distanțe este dată de distanța Minkowski (sau distanța indusă de norma p):

$$d_p\left(\begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix}, \begin{bmatrix} y_1 \\ \vdots \\ v_n \end{bmatrix}\right) = \left(|x_1 - y_1|^p + \cdots + |x_n - y_n|^p\right)^{\frac{1}{p}},$$

unde $p \geq 1$.

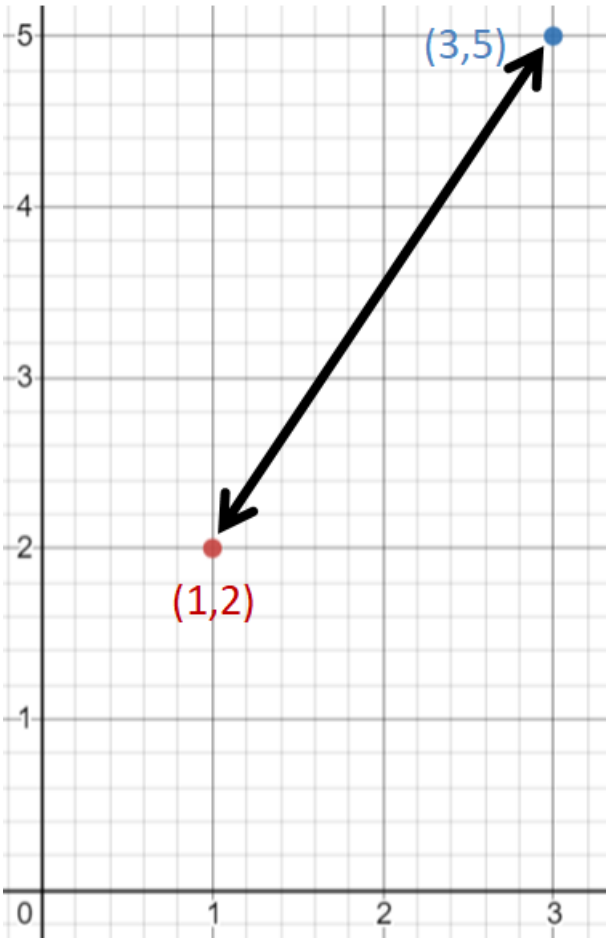
Pentru $p = 1$ obținem distanța Manhattan (*taxicab/city block*). Exemplu:

$$d_1\left(\begin{bmatrix}1\\2\end{bmatrix}, \begin{bmatrix}3\\5\end{bmatrix}\right) = |1-3| + |2-5| = 5$$



Pentru $p = 2$ obținem **distanța euclidiană**. Exemplu:

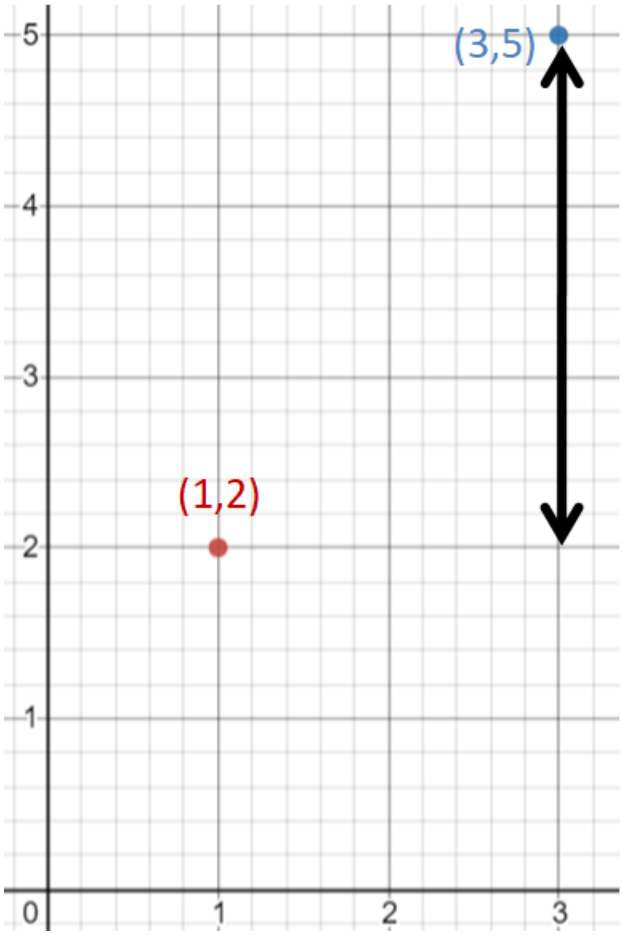
$$d_2\left(\begin{bmatrix}1\\2\end{bmatrix}, \begin{bmatrix}3\\5\end{bmatrix}\right) = d\left(\begin{bmatrix}1\\2\end{bmatrix}, \begin{bmatrix}3\\5\end{bmatrix}\right) = \sqrt{(1-3)^2 + (2-5)^2} = \sqrt{2^2 + 3^2} = \sqrt{13}$$



Implicit, dacă într-un exercițiu nu se specifică cu ce distanță se lucrează, atunci se va lucra cu distanța euclidiană.

Pentru $p = \infty$ obținem distanța Cebîșev. Exemplu:

$$d_\infty \left(\begin{bmatrix} 1 \\ 2 \end{bmatrix}, \begin{bmatrix} 3 \\ 5 \end{bmatrix} \right) = \max\{|1-3|, |2-5|\} = \max\{2, 3\} = 3$$



Nu trebuie să rețineți numele distanțelor, ci doar să aplicați o distanță în funcție de p.

2 kNN = k Nearest Neighbours

- k este un număr natural nenul pe care îl setăm înainte de a aplica algoritmul.

- suntem tot în contextul învățării supervizate de tip clasificare, dar vom putea face și regresie

- totuși, până acum am fost în cazul în care algoritmii făceau ceva la antrenare (ID3 făcea un arbore; AB făcea funcția/ansamblul $H_T(x)$; NB și JB făceau niște estimări de probabilități/parametri); kNN nu face mai nimic la antrenare: doar memorează datele (de aici vine și titlul capitolului din

carte: *Învățare bazată pe memorare*); abia la testare, când vine o instanță își creează modelul și furnizează eticheta prezisă; astfel, despre kNN se va spune că este ***lazy*** și despre ID3, AB, NB, JB, că sunt ***eager***.

3.

1. k-Nearest Neighbor Learning

[E. Fix,J. Hodges, 1951]

Training:
Store all training examples

Classification:
Given a query/test instance x_q,
first locate the k nearest training examples $x_1, \ldots, x_k$,
then estimate $\hat{f}(x_q)$:

- in case of discrete-valued $f : \Re^n \to V$,
  take a vote among its k nearest neighbors

$$\hat{f}(x_q) \leftarrow \underset{v \in V}{\operatorname{argmax}} \sum_{i=1}^{k} \delta(v, f(x_i))$$

  where $\delta(a, b) = 1$ if $a = b$, and $\delta(a, b) = 0$ if $a \neq b$
- in case of continuous-valued f,
  take the mean of the f values of its k nearest neighbors

$$\hat{f}(x_q) \leftarrow \frac{\sum_{i=1}^{k} f(x_i)}{k}$$

8.

A k-NN Variant: Distance-Weighted k-NN

We might want to weight nearer neighbors more heavily:

- for discrete-valued f: $\hat{f}(x_q) \leftarrow \operatorname{argmax}_{v \in V} \sum_{i=1}^{k} w_i \delta(v, f(x_i))$

  where

  $w_i \equiv \frac{1}{d(x_q, x_i)^2}$

  $d(x_q, x_i)$ is the distance between x_q and x_i
  but if $x_q = x_i$ we take $\hat{f}(x_q) \leftarrow f(x_i)$

- for continuous-valued f: $\hat{f}(x_q) \leftarrow \frac{\sum_{i=1}^{k} w_i f(x_i)}{\sum_{i=1}^{k} w_i}$

Remark: Now it makes sense to use *all* training examples instead of just k. In this case k-NN is known as **Shepard's method** (1968).

(slide-uri preluate din `https://profs.info.uaic.ro/~ciortuz/SLIDES/ml8.pdf`)

Aplicare 3NN (kNN când $k = 3$):

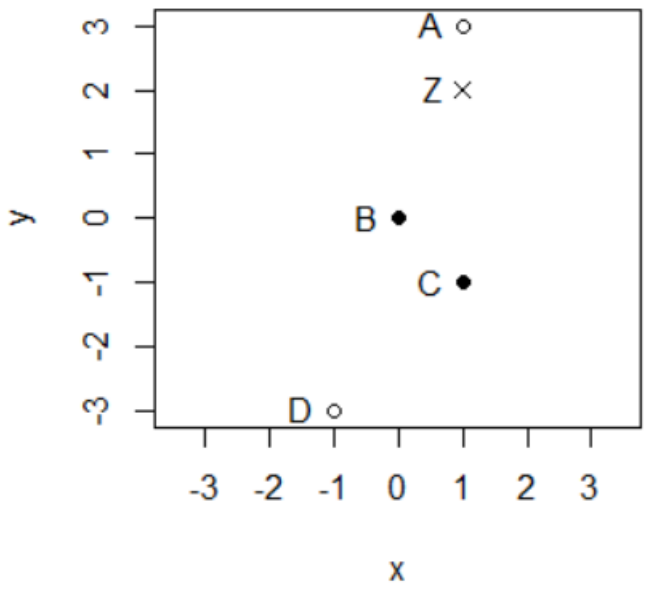
Date de antrenare: A, B, C, D (etichetele sunt date în imagini)

Date de test: Z

3NN vecinătatea lui Z (adică cei mai apropiați 3 vecini ai lui Z, adică cele mai apropiate 3 puncte de Z) este $\{A, B, C\}$.

Observație: pentru a afla kNN vecinătatea nu trebuie neapărat să calculați distanțe, ci puteți să desenați niște cercuri pe desen. De exemplu, vedeți ex. 1/pag. 426.

1. **Clasificare** (prezice o clasă/*ceva discret*)



(a) **fără ponderi**

Pentru ∘ vom avea un singur vot: de la A.

Pentru • vom avea 2 voturi: de la B și C.

$$\left.\begin{array}{l} \circ : 1 \\ \bullet : 2 \end{array}\right\} \overset{\text{argmax}}{\Rightarrow} \bullet$$

Deci, returnăm • pentru Z.

Observație: Când lumea vorbește despre kNN, de obicei, se referă la aest caz: clasificare, fără ponderi.

(b) **cu ponderi**

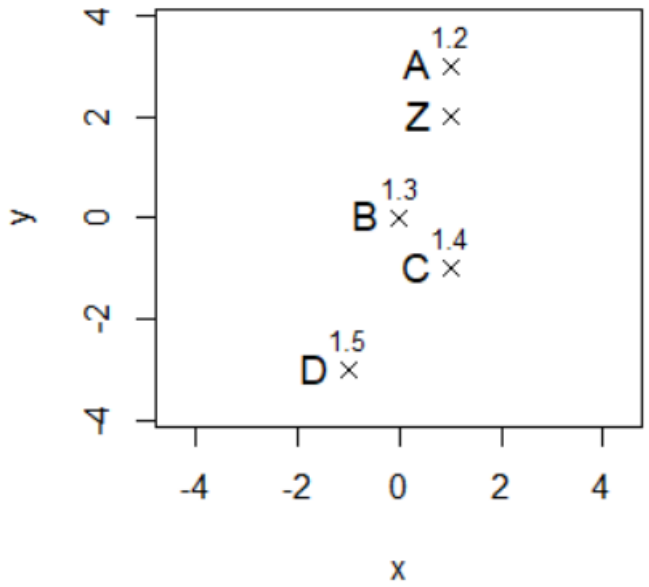
Voturile de mai înainte vor fi înlocuite cu ponderi. Ponderile sunt construite în așa fel încât o pondere mare corespunde unui punct apropiat, iar o pondere mică, unui punct îndepărtat. Dacă nu se specifică atfel în enunț, veți folosi ponderile din slide-uri de mai sus (și distanța euclidiană).

$$\left.\begin{array}{l} \circ : w_A = \frac{1}{d^2(A,Z)} = \frac{1}{1} = 1 \\ \bullet : w_B + w_C = \frac{1}{d^2(B,Z)} + \frac{1}{d^2(C,Z)} = \frac{1}{\sqrt{1^2+2^2}^2} + \frac{1}{\sqrt{3^2}^2} = \frac{1}{5} + \frac{1}{9} = 0.3111 \end{array}\right\} \overset{\text{argmax}}{\Rightarrow} \circ$$

Deci, returnăm ∘ pentru Z.

Observație: eticheta furnizată pentru Z de către kNN cu ponderi a ieșit diferită de eticheta dată de kNN fără ponderi.

2. **Regresie** (prezice un număr real/*ceva continuu*)



(a) **fără ponderi**

Pentru Z vom returna media aritmetică a etichetelor pentru A, B, C:

$$\frac{1.2 + 1.3 + 1.4}{3} = 1.3$$

(b) **cu ponderi**

Pentru Z vom returna media aritmetică ponderată a etichetelor pentru A, B, C:

$$\frac{w_A \cdot 1.2 + w_B \cdot 1.3 + w_C \cdot 1.4}{w_A + w_B + w_C} = \frac{\frac{1}{d^2(A,Z)} \cdot 1.2 + \frac{1}{d^2(B,Z)} \cdot 1.3 + \frac{1}{d^2(C,Z)} \cdot 1.4}{\frac{1}{d^2(A,Z)} + \frac{1}{d^2(B,Z)} + \frac{1}{d^2(C,Z)}} =$$

$$= \frac{1 \cdot 1.2 + 0.2 \cdot 1.3 + 0.1111 \cdot 1.4}{1 + 0.2 + 0.1111} = 1.2322$$

Algoritmul lui **Shepard**: Cazul 2b de mai sus, DAR impunem k să fie egal cu numărul de instanțe din setul de antrenament, adică, pe exemplul anterior, avem că eticheta furnizată pentru Z va fi:

$$\frac{w_A \cdot 1.2 + w_B \cdot 1.3 + w_C \cdot 1.4 + w_D \cdot 1.5}{w_A + w_B + w_C + w_D} = \dots$$

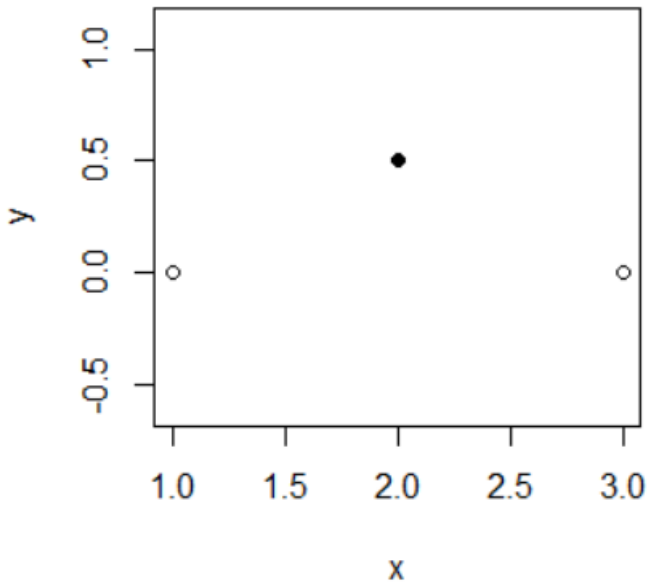
De obicei, ar trebui **să se specifice 3 chestiuni înainte de aplicarea lui kNN**:

- k
- distanța folosită; dacă nu se specifică, atunci folosiți distanța euclidiană, după cum am mai zis
- 1 sau 2 euristici/convenții:
  - ce se întâmplă dacă pentru al k-lea vecin avem mai mulți candidați aflați la aceeași distanță?
  - *doar dacă e cazul*: ce se întâmplă în caz de paritate de voturi

- k poate fi par, deși nu se recomandă să luam k par pentru că trebuie să spunem ce se întâmplă în caz de paritate de voturi;

- chiar dacă k este un număr par, putem avea un caz în care avem paritate de voturi; prima euristică poate crea acest caz

Atenție! Calculați eroarea la antrenare și eroarea la CVLOO pentru datele din desen **pentru 1NN**:



Eroarea la antrenare = 0 (cel mai apropiat vecin al unui punct este chiar acel punct)

Eroarea la CVLOO = $\frac{3}{3} = 1$

În general: **Eroarea la antrenare pentru 1NN (folosind orice distanță) pentru date consistente este 0.**

2.1 Granițe de decizie pentru 1NN cu distanța euclidiană

Varianta 1: cea din carte, urmărind exercițiile rezolvate din carte unde se desenează granițele de decizie (vezi, spre exemplu, ex. 11a/pag. 447)

Varianta 2: urmărind algoritmul descris mai jos. De menționat este că algoritmul nu este descris foarte în detaliu. Sper totuși că vă prindeți de ceea ce am vrut să vă zic prin el.

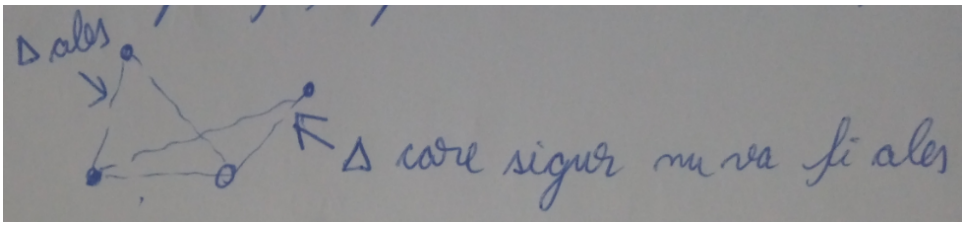
Algoritm pentru desenarea granițelor de decizie 1NN cu distanța euclidiană în 2 dimensiuni

1. start
   (a) ne uităm la înfășurătoarea convexă a punctelor
   (b) există segment pe ea cu vârfurile etichetate diferit?
      - DA: de acolo începem
      - NU: luăm un triunghi cu o latură cu vârfurile etichetate diferit astfel încât interiorul cercului circumscris să nu aibă vreun alt punct
2. Având o latură cu vârfurile etichetate diferit căutăm *"cel mai apropiat"* punct, adică un punct astfel încât triunghiul format să nu includă alte puncte
3. Ducem semi-mediatoarele în triunghiul respectiv
4. Verificăm, având deja centrul cercului circumscris (de la 3)), dacă interiorul cercului circumscris triunghiului conține vreun punct (cu compasul)
   - DA: revenim la 2), alegând alt punct și ștergând desenele de la 2) și 3)
   - NU: atunci triunghiul este OK; mergem la pasul 5)
5. Pe o altă latură a triunghiului cu vârfurile etichetate diferit:
   - dacă latura aparține înfășurătorii convexe (*"dă în afară"*): desenăm mediatoarea și *în afară* și apoi revenim la pasul 1)
   - altfel: revenim la pasul 2)

Ne oprim: când toate punctele etichetate diferit sunt separate.

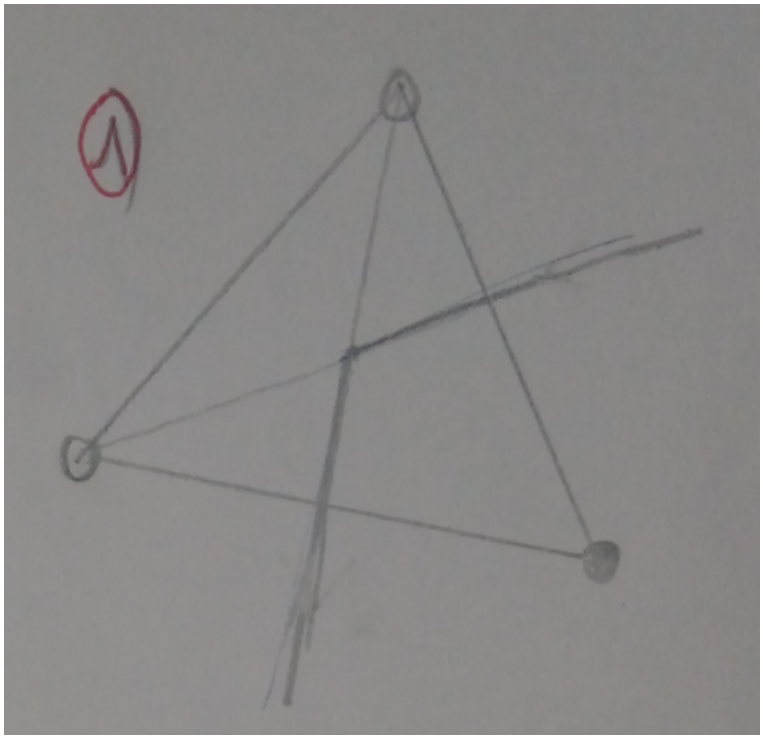
Observații:

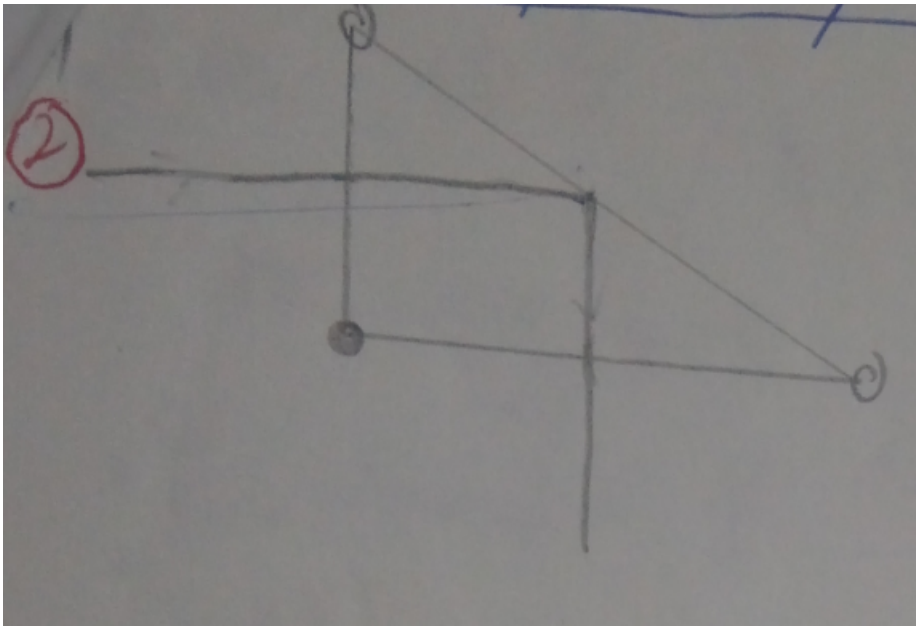
- Dacă un punct se află pe un cerc, atunci punctul nu este considerat a fi în interiorul cercului.
- Triunghiurile OK care au fost alese pe parcurs nu trebuie să aibă vreo suprafață pe care să se suprapună. De exemplu:

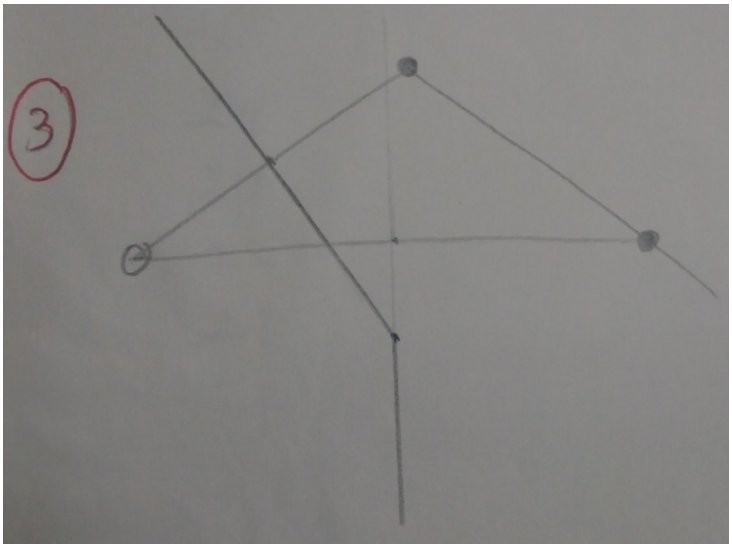


- Pasul 1) ne dă o latură cu vârfurile etichetate diferit.
- Pașii 2)-4) ne dau un triunghi (cu semi-mediatoarele desenate).
- Pasul 5) ne spune cum continuăm.
- mediatoarea unui segment = perpendiculara pe segment ce conține mijlocul segmentului
- centrul cercului circumscris unui triunghi se află la intersecția mediatoarelor laturilor triunghiului.

Exemple pentru pasul 3):
(**Regula**: dinspre centru (centrul cercului circumscris/intersecția mediatoarelor) înspre *"afara laturii"*))


1


2


3

Schemă de final

1. Remember
   (a) Vectori
   (b) Distanțe
2. kNN
   (a) lazy vs eager
   (b) i. kNN pentru clasificare; fără ponderi
       ii. kNN pentru clasificare; cu ponderi
       iii. kNN pentru regresie; fără ponderi
       iv. kNN pentru regresie; cu ponderi
       v. algoritmul lui Shepard
   (c) 3 chestiuni de specificat înainte de aplicarea lui kNN
   (d) eroarea la antrenare (pentru 1NN este zero dacă setul de date este consistent)
   (e) eroarea la CVLOO
   (f) granițe de decizie pentru 1NN cu distanța euclidiană