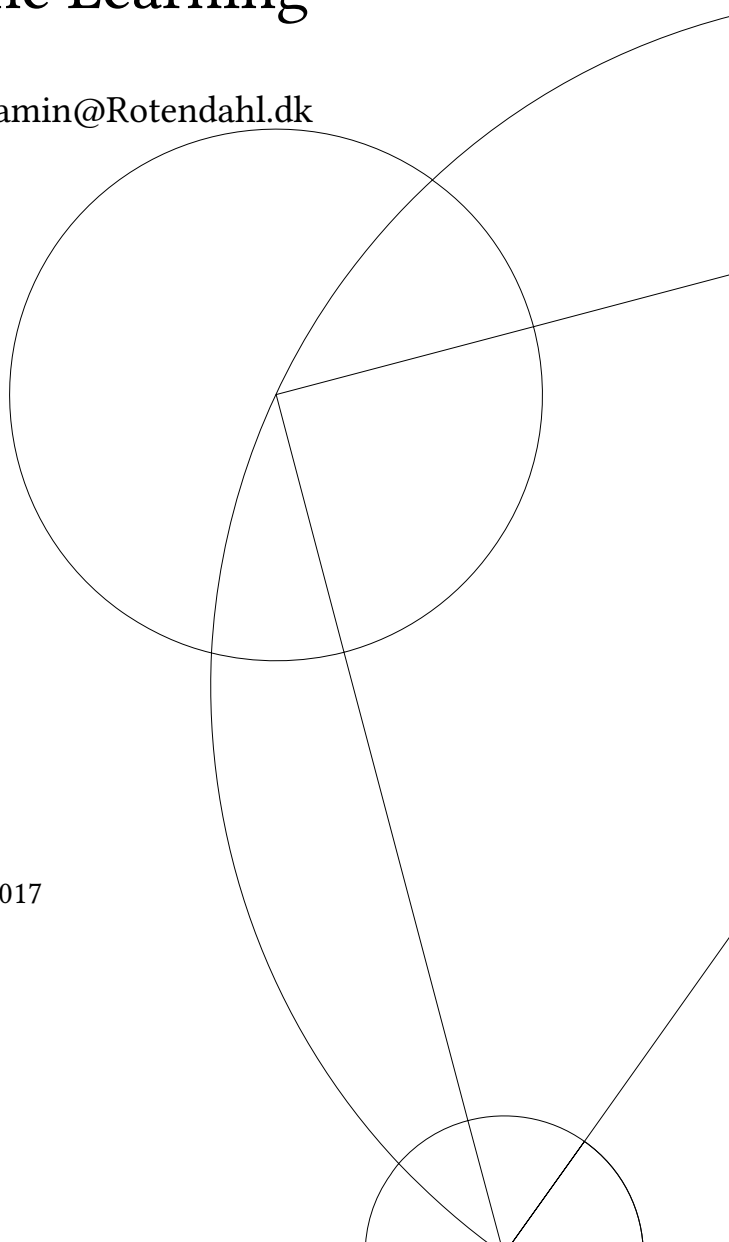
FACULTY OF SCIENCE
UNIVERSITY OF COPENHAGEN

Espergærde Gymnasium

Opgave i Machine Learning

Benjamin Rotendahl — Benjamin@Rotendahl.dk



February 5, 2017

Introduktion til opgaven

Denne opgave er en videreudvidelse til det machine learning i lærte den 1/2. Målet med opgaven er at implementere en "Supervised Learning" algoritme der kan kende forskel på kræftpatienter.

Data'en til opgaven er en større del af data'en fra perceptron algoritmen. Hvert data punkt består af et sæt (x, y) hvor x er en vektor i 10 dimensioner og y er et "label", der enten er 1 for at indikere en godartet svulst eller -1 for en ondartet. Formatet for x er:

Threshold	1
Clump Thickness	7
Uniformity of Cell Size	1
Uniformity of Cell Shape	4
Epithelial Cell Size	2
Bare Nuclei	3
Bland Chromatin	8
Normal Nucleoli	10
Mitoses	3

Målet med opgaven er at lave en funktion der kan tage en patient som input og giver et label som output.

Introduktion til algoritmen

Algoritmen vi skal bruge er den såkaldte *K Nearest Neighbor*(NN), vi benytter den i varianten hvor $k = 3$. Navnet til algoritmen skyldes at den benytter ideen om "naboskab/enshed" til at forudse hvilke labels den skal tilskrive data punkter. Uformelt kan ideen opsummeres med følgende scenarie:

- Du har en veninde der hedder Anna og en ven der hedder Anders.
- Du kan godt lide samme musik, film, mad osv. som Anna kan lide.
- Anders kan normalt bedst lide ting du ikke kan.
- Anders og Anna har været på et nyt sted og spise.
- Anna synes stedet var godt, Anders kunne ikke lide det.

Givet de fem stykker information præsenteret ovenfor, gætter du på du ville synes det nye sted eller ej? Siden at du generelt har meget tilfælles med Anna, ville det være mest rimeligt at gætte på ja.

NN benytter samme tanke gang, da vi arbejde med vektorer der ikke har nogen mening skal vi bruge en måde at måle naboskab. Ser vi på vektorerne som værende repræsenteret i det Euklidiske rum kan afstandsformlen bruges til at måle hvor "ens" de er. Afstandsformlen er defineret som:

$$d = \sqrt{\sum_{i=0}^{i=d} (u_i - v_i)^2}$$

For at gøre formlen simplere dropper vi kvadratroden, dette kan vi gøre da vi er ligeglade med den faktiske afstand mellem punkter og kun interessere os for forholdet mellem. Det er lige meget om vi har $|AB| = 5, |AC| = 4$ eller $|AB| = 25, |AC| = 16$, da vi stadig kan se at afstanden mellem $|AC|$ er kortets. Derfor bruger vi følgende formel der er mere overskuelig og effektiv da kvadratroden er en "dyr" operation for computere.

$$d = \sum_{i=0}^{i=d} (u_i - v_i)^2$$

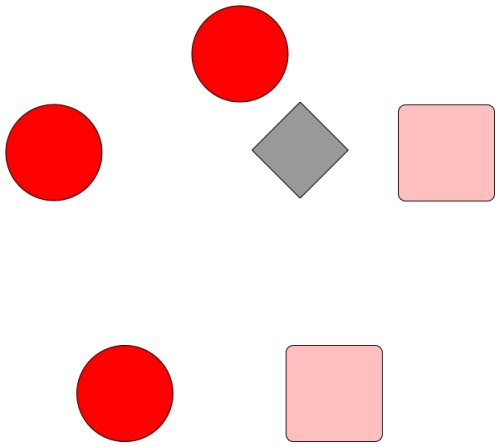


Figure 1: Eksempel på punkter i rummet

Givet overstående figur skal vi bestemme om den grå diamant en cirkel eller en firkant. Det første vi gør er at bestemme afstanden til alle punkterne for at se hvilke tre den er tættest på. Vi ser at den er tættest på 2 cirkler og en firkant. Eftersom at det er 2 mod 1, konkluderer vi at det er mest sandsynligt at det er en cirkel.

k'et i *K Nearest Neighbor* afgør hvor mange nabo'er vi sammenligner med. Alt efter hvilket k vi vælger havde vi fået et forskelligt svar. Valget afhænger af datasættet og er noget man kan prøve sig frem med.

Algoritmen kan opsummeres i følgende trin, givet punktet p og data sættet $D = [(x_1, y_1), (x_2, y_2), \ldots (x_n, y_n)$ skal følgende trin udføres:

1. Beregn afstanden fra p til alle punkter x.
2. Find de 3 punkter hvor afstanden var mindst.
3. Se om der er flest naboer med -1 eller 1.
4. Meld flertallets markering tilbage.

Introduktion til koden

Sammen med denne pdf er der udleveret en fil ved navn *opgave.php*, i denne fil er der 5 funktioner der skal implementeres før algoritmen virker. Kan du ikke køre *php* på din maskine kan du åbne koden i f.eks notepad og kopiere den ind på f.eks http://www.writephponline.com

Nedenfor præsenteres de fem funktioner og deres indgangs og udgangsbetingelser sammen med nogle tips.

I koden kan variablen *$data* bruges til at tilgå data'en.

- *$data[i]* Giver det datapar på formen (y, x) der står på plads i
- *$data[i][0]* Giver markeringen for det $i$te data par altså y_i
- *$data[i][1]* Giver vektoren for det $i$te data par altså x_i

dist — Afstandsfunktionen Denne funktion modtager to vektorer og skal bruge afstandsformlen til at returnere hvor langt der er mellem dem. Altså givet de to vektorer $u = \begin{pmatrix} 9 \\ 2 \\ -1 \end{pmatrix}$ og $v = \begin{pmatrix} 3 \\ 5 \\ 7 \end{pmatrix}$ Skal den returnere:

$$(9-3)^2 + (2-5)^2 + (-1-7)^2 = 109$$

Hint : Lav en værdi der først regner $(9-3)^2$ ud og så lægger de andre oven i

guessLabel — Gæt label Denne funktion modtager en liste på formen $[-1, 1, -1]$. Den skal så tælle om der er flest -1 eller 1 og returnere den der er flest af. **Hint : Det kan gøres med et regnestykke og en if statement**

sortClose — Sorter de nærmeste Vi ønsker at holde styr på en "afstandsvektor" der fortæller os hvilke 3 punkter der lige nu er tættest på vores nye punkt. Denne funktion modtager data på formen $[[2, 109], [1, 152], [2, 293]]$, hvor det første tal i parret $[2, 109]$ står for hvilken plads i *data* den har, og det andet tal er afstanden til p Målet er at returnere en liste hvor det altid er den med højest afstand der står først. Så svaret er:

$$[[2, 293], [2, 109], [1, 152]]$$

Det smarte ved denne funktion er at vi nemmere kan holde styr på de tre der er tættest da det altid vil være den første vi skal udskifte hvis en var lavere. **Hint : Lav en ny tom liste, find det største element og placer det først**

getClosest — Find de nærmeste Funktionen skal lede hele *$data* igennem, finde de punkter der er nærmest og returnere en liste på formen $[[i_1, d_1], [i_2, d_2], [i_3, d_3]]$ Hvor i punktets plads, og d er dens afstand. **Hint : Start med at gæt på at de 3 første punkter er nærmest og brug *sortClose* til at få nye element ind i listen.**

NearestNeighbours — 3NN Kalder alle de overstående funktioner og er bindeled mellem dem.