Algoritmer för att klassificera handskrivna siffror

Datum: 2019-04-09, Klass: MT1B, Grupp 11: Linus Lagerholm, Ramez Rizek

1. Inledning

Algoritmer är skapade för att datorer ska kunna lösa uppgifter snabbare än människor. Syftet med detta projekt är att implementera tre algoritmer som transformerar handskrivna siffror till information som en dator kan förstå och sedan bestämma vilken av de algoritmerna som är mest effektiv.

2. Metod

För att kunna jämföra två 16x16 matriser (en matris representerar en siffra) måste man först omvandla informationen som en matris har. En träningsmängd på 7291 matriser och en testmängd på 2007 matriser transformeras till klassificerade respektive oklassificerade vektorer med längden 256 element, på så sätt kan man börja med att klassificera datan i testmängden. Det finns ytterligare två matriser, dessa kallas för träningsfacit respektive testfacit. De två facit-matriserna innehåller vilken siffra varje matris representerar i de förstnämnda datamängderna.

2.1 Närmaste granne

Närmaste granne-algoritmen beräknar normen mellan en vektor från testmängden och varje vektor från träningsdatan stegvis för att klassificera vektorerna från testmängden. Detta görs med hjälp av ekvationen för avståndet mellan två vektorer, se ekvation (1)[1]. Variabeln n är antalet element. Det kortaste avståndet mellan testsiffran och träningssiffrorna avgör vilken klass (siffra) vektorn i testmängden klassificeras som.

$$||u - v|| = \sqrt{(u_1 - v_1)^2 + (u_2 - v_2)^2 + \ldots + (u_n - v_n)^2} \qquad (1)$$

2.2 Närmaste medelsiffra

Närmaste medelsiffra-algoritmen behandlar datan annorlunda i jämförelse med närmaste granne-algoritmen. För att klassificera en vektor hittas istället den kortaste normen mellan den aktuella oklassade vektorn från testmängden och tio olika medelvektorer, se ekvation (1). En medelvektor för en av klasserna fås genom att först skapa en matris som är en kvot av summan för alla matriser. Matriserna som summeras representerar samma klass från träningsdatan. Därefter omvandlas matrisen till en vektor. Detta görs för att algoritmen ska kunna beräkna det euklidiska avståndet mellan en medelvektor och den oklassificerade vektorn. Var och en av medelvektorerna utgör en klass (en siffra mellan noll till nio) som de oklassificerade vektorerna kan jämföras med. Den medelvektor som är närmast vektorn från testmängden avgör vilken klass vektorn klassificeras som.

2.3 Projektion

Projektions-algoritmen använder sig av tio olika underrum för att klassificera vektorerna från testmängden. Matrisen A från ekvation (2) innehåller alla matriser från träningsmängden som representerar samma siffra. För att få fram underrummen för varje klass används ekvationen för singulärvärdesfaktorisering (SVD), ekvation (2), på matrisen A_i, där i ges av klassen (siffran). Basvektorerna B_i som spänner upp underrummet ges av antalet kolonner d från ekvation (3) i $U^{(i)}$ där d karaktäriserar antalet dimensioner underrummet består av [2]. Det kortaste avståndet mellan underrummen och de oklassificerade vektorerna bestämmer vilken klass (siffra) de tillhör.

$$U^{(i)}\Sigma^{(i)}\left(V^{(i)}\right)^t = A_i \in \mathbb{R}^{256\times n_i} \qquad i = 0, 1, 2, \ldots 9 \qquad (2)$$

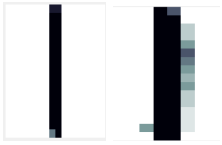
$$B_i = (u_1^{(i)}, \ldots, u_d^{(i)}) \qquad i = 0, 1, 2, \ldots 9 \qquad (3)$$

3. Resultat

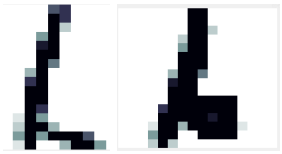
Siffror	Närmaste granne rätt/fel	Närmaste Medelvärde rätt/fel	Projektion rätt/fel
0	350/9	297/62	353/6
1	255/9	259/5	259/5
2	183/15	145/53	178/20
3	154/12	131/35	144/22
4	180/20	150/50	183/17

5	145/15	123/37	150/10
6	164/6	143/27	164/6
7	139/8	117/30	139/8
8	148/18	133/33	150/16
9	169/8	141/36	164/13
Hela testmängden	1887/120	1639/368	1740/267
Snittid	33.9s	0.11s	0.51s

Tabell 1. Tabell för resultaten av algoritmerna för vardera klass/siffra.



Figur 1 & 2. Ettor som klassificerats korrekt.



Figur 3 & 4. Ettor som har klassificerats som sexor.

Figur 5 & 6. Nia och åtta som klassats som ettor.

Då projektions-algoritmen används klassas figur 1 och 2 som ettor, vilket är korrekt. Figur 3 och 4 klassas dock inte som ettor, trots att dem är det enligt **facit.** Figur 5 och figur 6 är siffror som felaktigt klassas som en etta. Av 1743 siffror klassas 12 felaktigt som ettor.

4. Diskussion

Grupp 11 ansvarar för att undersöka klassen som bestod av ettor. Man kan anta att fel klassificeringarna beror på att informationen i matriserna för vanliga ettor och de andra siffrorna som klassificeras som ettor (eller vice versa) är för lika. Detta beror på att man har en parameter som definierar en siffra, detta är de svart- och vit-värdena som framställs i en 16x16 matris. En nia som klassas som en etta, såsom den i figur 5, kan bero på att de vita värdena som finns i nians cirkel är för få, och de svarta värdena tar dess plats, vilket gör att nian liknar en etta istället. Ettorna i figur 1 och 2 klassas rätt eftersom deras svart- och vit-värden passar in med definitionen som algoritmen använder. Ettorna i figur 3 och 4 gör däremot inte det, de har för många svarta värden som får dem att likna en annan siffra, i detta fall sexor. Definitionerna som algoritmerna använder sig av för att klassificera siffrorna är för otydliga.

Av alla algoritmer som vi har använt så är det svårt att avgöra vilken som fungerar bäst eftersom det finns flera saker man måste ta hänsyn till, såsom tid och precision. Om man observerar tabell 1 så kan man se att den snabbaste algoritmen var närmaste medelvärdet, men den hade lägst precision. Man kan också se att närmaste granne-algoritmen var långsammast, och tar dessutom mycket mer minne. Detta kan leda till problem vid större mängd av data, men den visade bättre resultat än medelvärdes-algoritmen. Projektions-algoritmen kräver relativt lite tid för att utföras och den hade högst procent rätt för klassificeringarna, men den kan endast undersöka en klass åt gången.

4.1 Slutsats

Om man vill ha hög precision och inte bryr sig om att man behöver kolla var klass för sig kan man använda sig av projektions-algoritmen. Om man vill ha hög precision, men inte bryr sig om tiden det tar, kan man använda sig av närmaste-granne algoritmen. Närmaste medelvärdes-algoritmen bör användas om man inte behöver bry sig lika mycket om precisionen och vill att det ska ta kort tid. Ingen av dessa tre algoritmer är överlägsen, dem är användbara i olika situationer med olika förutsättningar.

5. Källor

1. G. Baravdish. Linjär algebra TNA002. kompendium utgivet av Linköpings universitet, (2018)
2. S. Berkant. *Föreläsning 6&7*. TNA005. Norrköping: LiU 2019. [citerad 15:e mars 2019]. Hämtad från https://www.itn.liu.se/grundutbildning/kurs/tna005/forelasningar?l=sv