Miniprojekt 2: Klassificering av handskrivna siffror

Emil Gustafsson och Samuel Rising, MT1A/B, Grupp 1

April 24, 2016

1 Inledning

Syftet med denna rapport är att visa hur olika algoritmer kan appliceras för att läsa handskrivna siffror. Detta görs med tre olika algoritmer som sedan analyseras för att visa skillnaden mellan de olika algoritmerna. Olika metoder som används är, *närmaste grannemetoden*[1], *närmaste medelmetoden*[1] samt *projektion på underrum*[1].

2 Metod

För att utföra projektet användes material ifrån kursen. Vilket var *datamängd*, *föreläsningsanteckningar* samt programvaran *Matlab*[2].

Datamängden är matriser som beskriver antalet svart färg på 16x16 pixlar som beskriver handskrivna siffror. Det består av en testmängd som inte är klassificerade samt en träningsmängd som är klassificerad.

Metoden som används i samtliga algoritmer är att svaren på träningsmängden och se om varje element i den kan klassificeras som siffran ett. Sedan körs en funktion beroende på vilken algoritm som testas, *närmaste grannemetoden, närmaste medelmetoden* eller *projektion på underrum.*

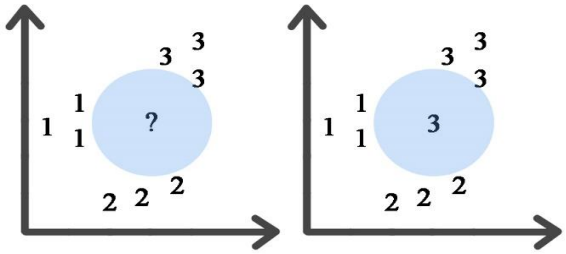
2.1 Närmaste grannemetoden

Varje "skiva" ifrån träningsmängden formas om till en vektor och jämförs med testvektorn. Den träningsvektorn med kortast avstånd till testvektorn anger resultat med tillhörande klass. Algoritmen tar differensen mellan testmängden och tränings-mängden, *se formel.1*. Denna skillnad används för att testa emot träningsmängden som består av 7291 siffror. Det närmaste avståndet klassificerar siffran. *Se fig.1*

Formel 1

$$l = \sqrt{(U1 - V1)^2 + (U2 - V2)^2 + \cdots + (Un - Vn)^2}$$

Där l är längden samt U och V är vektorer av dimensionen n.

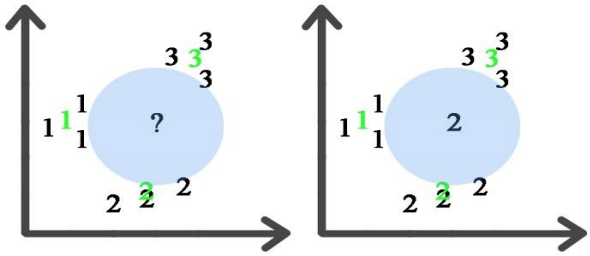


Figur 1 Illustration av närmaste grannemetoden.

2.2 Närmaste medelmetoden

Först skapas tio medelvektor för varje klass, sedan jämförs testvektorn med samtliga träningsvektorer, på samma sätt som i *närmaste grannemetoden.* Skillnaden nu är att testmängden bara jämförs med tio värden istället för 7291 värden vilket den gör i *närmaste grannemetoden*.

I genereringen av en medelvektor sorteras alla omformaterade vektorer efter dess tillhörande siffra och läggs sedan i en ny matris som expanderar vid varje iteration med den aktuella vektorn. När alla vektorer av den aktuella siffran gallrats ut så genereras en medelvektor av dessa vektorer. Detta upprepas för alla klasser och sammanställs i en stor matris. Se *fig.2*.



Figur 2 Illustration av närmaste medelmetoden.

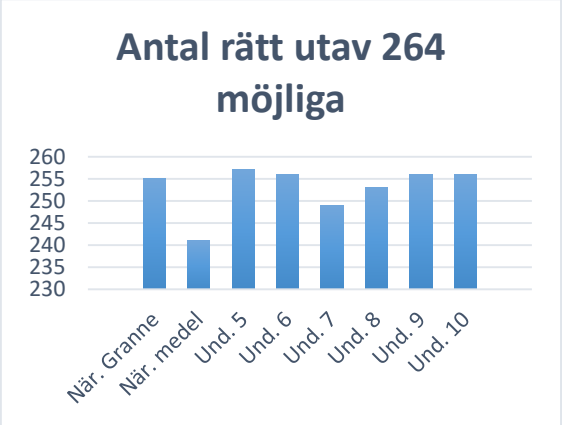
2.3 Projektion på ett underrum

Denna metod genererar ett underrum till R^{256} för varje klass med hjälp av singulärvärdesfaktorisering. Dessa underrum spänns upp av 5 till 10 vektorer. Underrummen sparas i en 4 dimensionell matris sorterad i ordning efter klass. I testfasen skickas denna matris in till en ny algoritm som mäter avstånd mellan en testvektor och varje underrum U_{klass} och svarar med den klass vars underrums avstånd till

vektorn var kortast. Avstånd betyder i detta fall längden av testvektorns ortogonala komponent till U_{klass}.

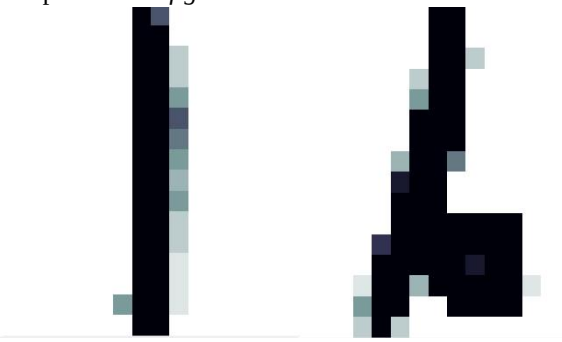
3 Resultat

De tre nämnda algoritmerna tar det olika lång tid för att köra samt att de klassificerar testmängden olika. Totala antalet mätningar för klassen 1 är 264, se *fig.3*.



Figur 3 Stapeldiagram på mängden rätt klassificerade siffror.

Detta resultat ger en fel klassificering mellan tre till nio procent. Se *fig.4*.



Figur 4 Testade med projektion på underrum. Vänster är rätt klassificerad medan den till höger är fel klassificerad som en sjua.

Mätningar utfördes för att ta reda på tiden det tar att utföra generering och mätning av testmängden mot träningsmängden. Se *tabell.1*.

	Generering	*Mätning*
När. Granne	Krävs ej	5.808406
När. medel	0.011432	0.006779
Und. 5	0.646015	0.126891
Und. 6	0.635997	0.153055
Und. 7	0.617574	0.175927
Und. 8	0.625321	0.202342
Und. 9	0.624733	0.222191
Und. 10	0.622874	0.245202

Tabell 1 Tabell över körtider för de olika algoritmerna angivna i sekunder.

4 Diskussion

Den snabbaste algoritmen är *närmaste medelmetoden*, detta på grund av att den bara gör mätning mot tio värden för varje siffra som ska testas.

Snabbast är dock inte alltid den bästa då den har högst antal felklassificerade siffror. Anledningen till detta kan vara att den inte jämför mot genuina siffror utan en medelvektor av siffror.

Närmaste grannemetoden använder all data ifrån träningsmängden utan komprimering, alltså använder den mest minne. Närmaste medelmetoden och projektion på underrum skapar ny data ifrån träningsmängden och kan sedan slänga bort all träningsdata efter initieringsprocessen.

Anledningen till att det blir felklassificeringar som i *fig.4.* är av den anledningen av det mänskliga indata som görs. Om siffran är slarvigt skriven är det större risk att den blir fel klassificerad.

5 Slutsats

I de flesta scenarion fungerar *projektion på ett underrum* bäst då den är relativt snabb efter genereringen av underrum och har en av de mest träffsäkra klassificeringarna och relativt låg minnesanvändning av de som testades i denna undersökning. Det går även att mycket enkelt välja underrummens dimensioner för att optimera resultatet i en specifik applicering.

6 Referenser

[1] Berkant, S. (2016). *Föreläsning 6&7 TNA005.* Norrköping: LIU.

[2] Jönsson, P. (2010). *MATLAB beräkningar inom teknik och naturvetenskap.* Malmö: Studentlitteratur AB.