TANA17 Matematiska beräkningar med Matlab

Projekt 1. Identifiering av handskrivna siffror

1 Introduktion

I tillämpningar där man studerar olika slags objekt händer det ofta att objekten förekommer i olika varianter, eller klasser. Man vill då bestämma vilken klass ett visst objekt tillhör. Detta kallas *Klassificeringsproblemet*.

Ett exempel är epostfiltrering där ett inkommande brev antingen är skräp eller inte. Vi delar alltså upp alla inkommande brev i två klasser *skräppost* eller *inte skräp*. Då ett nytt brev dyker upp i inkorgen vill vi automatiskt avgöra vilken av klasserna det tillhör.

Ett annat exempel, som vi skall studera i projektet, handlar om automatisk posthantering. För att avgöra vart ett visst brev skall skickas måste vi läsa det postnummer som skrivits på brevet. Vi fotograferar då varje enskild siffra och försöker utifrån fotot avgöra vilken siffra det är. I detta fallet får vi 10 klasser (bestående av siffertyperna 0–9).

1.1 Närmsta-granne-metoden

En av de enklaste metoderna för att klassificera objekt, som även är en av de noggrannaste, kallas *närmsta-granne-metoden*. För att metoden skall fungera behöver vi ett sätt att mäta skillnaden mellan två objekt, eller *avståndet* mellan objekten. Vi samlar sedan in en stor mängd exempel på objekt där vi noggrant antecknar vilken klass de tillhör. Denna kallas *Referensmängden*.

Då vi träffar på ett nytt okänt objekt så jämför vi det med hela referensmängden tills vi hittar det objekt som skiljer sig minst från det nya. Vi antar sedan att det okända objektet är av samma klass som dess närmsta granne ur referensmängden.

Vi antar alltså att vi har m stycken objekt i referensmängden som kan betecknas

$$\{R_1, R_2, \ldots, R_m\}.$$

Låter vi den okända siffran vara S så hittar vi alltså det objekt R_i ur referensmängden sådant att

$$d(S, R_i) = \min_{1 \le j \le m} d(S, R_j).$$

där $d(\cdot, \cdot)$ är den funktion som mäter skillnaden mellan objekten. Vi antar sedan att S är av samma klass som R_i.

1.2 Datamängden

För att testa metoden finns två uppsättningar handskrivna siffror tillgängliga. Varje siffra består av 16×16 pixlar och lagras som en kolumnvektor av dimension 256.

Siffrorna kommer ursprungligen från brev som scannades av US Postal Service under 1980:talet och är en av de standardmängder som används för att testa klassificieringsalgoritmer.

Du får tillgång till siffrorna genom att ladda ner filen `DataSet.mat`. Referensmängden är sparad i matrisen `RefSet`, där varje kolumn motsvarar en siffra. I vektorn `RefAns` kan vi se vilken siffertyp som finns i motsvarande kolumn i matrisen `RefSet`. Om vi skriver

```
>> load DataSet
>> R100=RefSet(:,100);
>> RefAns(100)
 ans =
    6
```

så får vi en vektor R_{100} med de 256 pixlarna som motsvarar referenssiffra nummer 100. Vi ser även att denna siffra skall föreställa en sexa. För att visualisera siffrorna finns en Matlab funktion `DisplayDigit`. Skriv

```
>> DisplayDigit( R100 )
```

så visas en bild där vi tydligt ser att siffran verkligen är en sexa.

Referensmängden består av de siffror vi antar är kända och som vi skall använda för att genomföra klassificeringen. Matrisen `TestSet` innehåller de test siffror som vi skall försöka klassicifera med vår metod. Vi skall låtsas att vi inte vet vilka siffror som finns i *Testmängden* men vi har ett facit i vektorn `TestAns` som vi kan använda för att kontrollera att metoden gjort rätt.

2 Implementering av Algoritmen

För att utföra klassificeringen behöver vi en funktion som mäter avståndet mellan två siffror. Då varje siffra representeras av en kolumnvektor är det naturligt att använda det Euklidiska avståndet. Skillnaden mellan de två vektorerna x och y blir då

$$\|x - y\|_2 = \left(\sum_{i=1}^{n} |x_i - y_i|^2 \right)^{1/2}.$$

Eftersom siffrorna representeras av kolumnvektorer i $\mathbb{R}^{256}$ måste vi alltså först skriva en Matlab funktion

```
>> E = Distance( S1 , S2 );
```

där inparametrarna `S1` och `S2` är de två vektorer som motsvarar olika siffror ur test- eller referensmängden, och utparametern `E` är den Euklidiska avståndet mellan vektorerna.

För att utföra klassificeringen av en okänd siffra `S` så måste vi söka igenom referensmängden och hitta den siffra som har det minsta avståndet. Vi behöver alltså en funktion

```
>> Type = ClassifyDigit( S , RefSet , RefAns );
```

Enklast är att först beräkna en vektor innehållande avståndet mellan siffran S och alla siffror ur referens mängden. Vi kan sedan hitta den siffra R_i som har det kortaste avståndet till S. Vi tittar sedan i facit `RefAns` för att se vilken typ av siffra S antas vara.

Då ni skrivit både funktionerna `Distance` och `ClassifyDigit` så kan ni börja klassicifiera alla siffrorna i testmängden. Skriv ett huvudprogram där funktionerna ovan används för att typbestämma samtliga siffror som lagrats i matrisen `TestSet`. Huvudprogrammet skall alltså vara strukturerat som följer:

1. Nollställ en räknevariabel som håller reda på antalet lyckade klassificeringar.
2. Försök typbestämma en siffra S_i från testmängden med hjälp av `ClassifyDigit` funktionen.
3. Jämför resultatet från klassicifieringen med facit som finns i vektorn `TestAns`. Om vi lyckades så ränka upp din räknevariabel.
4. Fortsätt med $i = i + 1$ och börja om från steg 2 och klassificiera nästa siffra.

Då programmet är färdigt skall det rapportera hur många fall som lyckades. Exempel på utskrift kan vara

```
A total of 1873 or 2007 digits were classified correctly.
```

Dessutom är vill ni kunna studera de fall där klassiciferingen misslyckades i efterhand. Gör därför en utskrift liknande

```
Test digit i=199 is of type 9 but closest neighbor j=245 is of Type 8. Fail!
```

varje gång klassificeringen misslyckas. Då kan vi enkelt hitta dessa fall senare.

Vi är särskilt intresserade av om någon siffertyp är svårare att klassificera än andra. Ditt huvudprogram skall därför innehålla en vektor `MissClassified` som utökas med en plats varje gång klassificeringen misslyckas. Ni lägger då in korrekt siffertyp i vektorn. Då programmet är färdigt kommer vektorn `MissClassified` ha en längd som motsvarar antalet misslyckade klassificeringar och varje element kommer att innehålla den siffertyp för vilken klassificeringen misslyckades. Ni han sedan illustrera de misslyckade fallen med ett histogram.

3 Redovisning och Rapporten

Rapporten skall ha följande kapitel och innehåll

- *Introduktion* där problemet, dvs klassificiering av handskrivna siffror beskrivs kortfattat. Ni skall även kort beskriva Närmsta-granne-metoden. Data mängden skall också beskrivas.
- *Matlab program* där ni ska beskriva alla Matlab funktioner och program som ni skrivit. Ni skall inte klippa in programkoden utan för varje funktion beskriv kort funktionens syfte, vilka inparametrar den behöver och vad den gör.
- *Resultat* där det beskrivs hur lösningsmetoden fungerade. Alltså vilka resultat ni fick. Viktigast är hur stor andel av klassificeringarna som lyckades.
- *Diskussion* där resultatet sammanfattas. Fungerade metoden? Blev resultatet bra?

De grafer som minst måste ingå i rapporten är

- När ni beskriver datamängden skall ni visa exempel på åtminstone två olika siffror från referensmängden.
- I resultatdelen skall ett exempel på en lyckad klassificiering ingå. Ni visar då både siffran från testmängden och dess märmsta granne från referensmängden. Det skall även ingå ett exempel på en misslyckad klassificiering.
- Det histogram där antalet misslyckade klassificeringar uppdelat på siffertyp syns skall ingå.

Se dessutom generalla krav på rapport och matlab program från hemsidan.