Practicum Topografische Kaarten

Peter Kleiweg

December 1995

1 Bestanden

Tabel 1 geeft een overzicht van de bestanden die bij dit practicum worden gebruikt. In de rest van dit document zullen steeds de Unix-namen gebruikt worden als het om de bestanden gaat die voor zowel Unix als DOS aanwezig zijn. De DOS-namen zullen gebruikt worden in die gevallen waarin geen bestand voor Unix aanwezig is. De `.c` bestanden en `koh_test.vec` zijn gelijk voor Unix en DOS (met uitzondering van Unix LineFeed en DOS CarriageReturn/LineFeed). Onder Unix geldt: `cluster`, `koh` en `vec2ps` moeten executie-permissie hebben. Als deze drie bestanden nog niet aanwezig zijn kunnen ze worden aangemaakt door middel van `make`.

2 Bestandsformaat

Alle programma's met uitzondering van `KOHVIEW.EXE` maken gebruik van een bestand met definities voor vectoren. De eerste regel van dit bestand bevat een getal dat aangeeft uit hoeveel getallen één vector bestaat (de dimensie van de vector). Daarna volgen de vectoren zelf, eerst de naam en daarna de waarden. De waarden mogen zowel uit integers als uit reële waarden bestaan. Elke naam en getal moet steeds op een aparte regel staan. Lege regels worden genegeerd, evenals alles wat volgt op `#`. Namen van vectoren kunnen dus niet met `#` beginnen. Zie `koh_test.vec` voor een voorbeeld.

3 Gebruik van de programma's

Voor alle programma's geldt: Toets het commando zonder argumenten in om een overzicht te krijgen van de geldige argumenten. Argumenten tussen `[` en `]` zijn optioneel, de overige argumenten zijn verplicht.

De programma's `cluster`, `koh` en `vec2ps` maken PostScript bestanden aan. Een algemene opmerking vooraf over deze bestanden:

Als je een PostScript afbeelding wilt opnemen in een LaTeX-document met behulp van `psfig` dan is het noodzakelijk dat de BoundingBox in het PostScript document correct is gedefinieerd. Die vier getallen die je hier ziet zijn resp. de

Unix	DOS
`Makefile`	`MAKEFILE`
	`EGAVGA.OBJ`
`cluster`	`CLUSTER.EXE`
`cluster.c`	`CLUSTER.C`
	`CLUSVIEW.EXE`
	`CLUSVIEW.CPP`
`koh`	`KOH.EXE`
`koh.c`	`KOH.C`
	`KOHVIEW.EXE`
	`KOHVIEW.CPP`
`koh_test.vec`	`KOH_TEST.VEC`
`vec2ps`	`VEC2PS.EXE`
`vec2ps.c`	`VEC2PS.C`
	`VECVIEW.EXE`
	`VECVIEW.CPP`

Tabel 1: Bestanden voor practicum topografische kaarten

linker-, onder-, rechter- en boven-begrenzing. Wanneer je met een editor het bestand verandert zodat de afbeelding van grootte of positie verandert moet je niet vergeten deze waarden aan te passen. Voor het verkrijgen van de correcte waarden kun je bijvoorbeeld `ghostview` gebruiken.

3.1 cluster

Dit programma past het *merge clustering algorithm* toe op een set vectoren. Er wordt een PostScript afbeelding gegenereerd. Syntax voorbeelden:

```
cluster -i oote.vec -o boe
cluster -i oote.vec
```

Beide voorbeelden gebruiken `oote.vec` als invoer. Het eerste voorbeeld genereert `boe.ps`, het tweede genereert `clus_out.ps`.

3.1.1 PostScript aanpassingen

Hier volgt een overzicht van de wijzigingen die aangebracht kunnen worden in het PostScript bestand dat door `cluster` wordt gemaakt.

Hoogte De variabele `DY` wordt gebruikt voor de regelafstand.

Breedte De variabele `Width` definieert de breedte van de figuur zonder de tekst voor de labels.

3.2 CLUSVIEW.EXE

Alleen DOS, VGA scherm vereist. Dit programma past het *merge clustering algorithm* toe op een set vectoren. Het resultaat wordt grafisch weergegeven. Syntax voorbeeld:

```
CLUSVIEW OOTE.VEC
```

3.3 koh

Dit programma genereert een Kohonen-kaart van een set vectoren, inclusief vectorgrenzen en minimal spanning tree. Syntax voorbeelden:

```
koh -i oote.vec -o boe
koh -i oote.vec
```

De eerste variant genereert bestanden waarvan de naam begint met `boe`, de tweede variant genereert bestanden waarvan de naam begint met `koh_out`.

Overzicht van de overige parameters:

-m int Grootte van de kaart. Een waarde 10 resulteert in een 10×10 kaart.

-s int Begin-grootte van de kaart. (Default gelijk aan eindgrootte voor vectoren van dimensie 1, default 2 in andere gevallen.)

-e int Aantal epochs. In elk epoch worden alle vectoren één keer aangeboden, in willekeurige volgorde.

Het begrip 'begin-grootte' vraagt enige uitleg. Gewoonlijk wordt een Kohonen-kaart gevormd door in het begin de regio rond de winnaar waarin geleerd wordt groot te nemen, en tijdens het trainen langzaam te verkleinen. Daardoor kosten de eerste epochs in het begin meer computertijd dan de epochs aan het eind. Om tijd te besparen is de volgende methode gebruikt: Begin met een zo klein mogelijke kaart (2 bij 2) en vergroot de kaart tijdens het verloop van de training. Wanneer de kaart een stap vergroot wordt worden de units herverdeelt over de nieuwe kaart, proportioneel naar het oppervlak van de oude units dat door de units van de nieuwe kaart 'bedekt' worden. Leren blijft tijdens heel het verloop van de training beperkt tot de winnaar en de acht units direct er om heen. (Omdat de winnaar iets sneller leert dan de acht buur-units heeft trainen van een 2 bij 2 kaart zin.) Het gevolg is dat de eerste epochs minder tijd vragen dan de laatste epochs in plaats van meer.

Het programma maakt een reeks bestanden aan die de volgende extensies hebben:

.dif Bevat de verschillen (Euclidean Distance) tussen units die naast of boven elkaar liggen.

.log Bevat gegevens over gebruikte paramaters voor — en resultaten van — het leerproces.

.map Bevat alle waarden voor alle units in de kaart. Deze gegevens zouden gebruikt kunnen worden door een ander programma dat een topografisch kaart benut.

.ps PostScript bestand. De andere bestanden worden niet gebruikt voor het maken van de afbeelding.

.spn Bevat gegevens voor de minimal spanning tree: Verbindingen tussen kaart-posities.

.top Bevat de posities van de invoervectoren op de kaart.

3.3.1 Resultaten van het leerproces

Wanneer aan het eind van het leerproces meerdere vectoren op één unit worden geplaatst wordt dit gemeld door regels zoals:

```
aap == noot
```

De kwaliteit van de gevormde kaart kan worden afgeleid uit de waarde `Badness`, die aan het eind wordt afgedrukt. Hoe lager deze waarde hoe beter. Vergelijking van de waarde tussen verschillende trainingen met **`koh`** is alleen zinvol wanneer het programma wordt aangeroepen met identieke waarden voor alle parameters.

Een tweede indicatie over het succes vormt de minimal spanning tree. Korte verbindingen zijn een aanwijzing voor een goed resultaat. Kruisende verbindingen duiden op een slechter resultaat.

3.3.2 PostScript aanpassingen

Hier volgt een overzicht van de wijzigingen die aangebracht kunnen worden in het PostScript bestand dat door **`koh`** wordt gemaakt.

Meerdere vectoren op één unit Het kan voorkomen dat meerdere invoervectoren op de zelfde unit terecht komen. Het PostScript programma drukt de labels van deze vectoren over elkaar af waardoor ze onleesbaar worden. Aan het eind van het bestand staan de labels met hun posities. Je kunt deze op twee manieren aanpassen. Lange labels kunnen boven elkaar geplaatst worden. Verander de volgende regels:

```
3 4 (aap) n
3 4 (noot) n
```

in de volgende regels:

```
3 4.15 (aap) n
3 3.85 (noot) n
```

Labels van één letter kunnen naast elkaar geplaatst worden. Vervang de volgende regels:

```
3 4 (a) n
3 4 (n) n
```

door de volgende regel:

```
3 4 (a n) n
```

Zichtbare clustergrenzen Als je geen clustergrenzen wilt zien verander dan de waarde van de variabele `SHOW_GRAY` in `false`.

Minimal spanning tree Als je geen minimal spanning tree wilt zien verander dan de waarde van de variabele `SHOW_LINKS` in `false`.

De variabele `ANGLE` bepaalt de hoek van de gebogen lijnen. Wanneer de gebogen lijnen ongelukkig geplaatst zijn doordat ze precies door het midden van andere labels gaan kun je de waarde van deze variabele veranderen. De waarde moet tussen -120 en 120 blijven.

Grootte De variabele `DX` geeft de breedte van één unit weer, `DY` geeft de hoogte weer.

3.4 KOHVIEW.EXE

Alleen DOS, VGA scherm vereist. Dit programma toont de Kohonen-kaart die gegenereerd is door het programma `koh` op een VGA-scherm. Het maakt gebruik van de bestanden met de extensies `.dif`, `.spn` en `.top`. Syntax voorbeeld, eerst:

```
koh -i oote -o boe
```

en daarna:

```
kohview boe
```

In tegenstelling tot het PostScript bestand is dit programma niet perfect in de plaatsing van begin- en eindpunt van de gebogen lijnen.

Wanneer meerdere vectoren op één unit terecht komen kun je vergelijkbare wijzigingen aanbrengen zoals die beschreven zijn voor het PostScript bestand. In dit geval moet je de wijzigingen aanbrengen in het bestand met de extensie `.top`.

3.5 vec2ps

Dit programma maakt een PostScript afbeelding van een set vectoren. Syntax voorbeeld:

```
vec2ps -i oote.vec > boe.ps
```

Overzicht van de overige parameters:

-c In plaats van grijswaarden worden zwart gevulde cirkels gebruikt. De **oppervlakte** geeft de waarde aan.

-l float De *lower bound* voor de waarden in de vectoren. Default is het minimum van 0 en de kleinste waarde in de vectoren.

-u float De *upper bound* voor de waarden in de vectoren. Default is het maximum van 1 en de grootste waarde in de vectoren.

3.5.1 PostScript aanpassingen

Hier volgt een overzicht van de wijzigingen die aangebracht kunnen worden in het PostScript bestand dat door `vec2ps` wordt gemaakt.

Titel De naam van het invoerbestand wordt gebruikt als titel. Deze staat boven in het PostScript bestand. Wanneer je geen titel wilt verwijder dan de volgende twee regels die aan het eind van het bestand zijn te vinden:

```
0 SHIFT rmoveto
Title show
```

Hoogte De variabele `Height` bepaalt de hoogte van één regel.

Breedte De variabele `Width` bepaalt de breedte van één vakje of cirkel.

3.6 VECVIEW.EXE

Alleen DOS, VGA scherm vereist. Dit programma toont een set vectoren. Voor dit programma geldt: zwart is lage waarde, wit is hoge waarde. (Dus omgekeerd aan `vec2ps`.) Syntax voorbeeld:

```
VECVIEW -I OOTE.VEC
```

Overzicht van de overige parameters:

-T In plaats van grafische uitvoer worden rasters gebruikt. De uitvoer gaat naar `stdout`, zodat het resultaat opgenomen kan worden in een DOS ascii-bestand.

-L float De *lower bound* voor de waarden in de vectoren. Default is het minimum van 0 en de kleinste waarde in de vectoren.

-U float De *upper bound* voor de waarden in de vectoren. Default is het maximum van 1 en de grootste waarde in de vectoren.

4 Practicum opgaven

4.1 Kohonen testset

Het bestand `koh_test.vec` bevat de set vectoren die Kohonen heeft gebruikt voor zijn demonstratie. Creëer een Kohonen kaart voor deze set vectoren. Experimenteer met verschillende waarden voor mapgrootte, begin-grootte en aantal epochs.

Bekijk de set vectoren met het merge clustering algoritme. Verklaar het resultaat.

4.2 Verschillende dimensies

Maak een bestand met een kleine set vectoren met 2 dimensies, bijvoorbeeld:

```
2
  11
1
1
  12
1
2
  13
1
3
  21
2
1
```

Enzovoort, minimaal 5 bij 5.

Genereer een Kohonen-kaart met deze set en bekijk het resultaat. Je hebt in dit geval maar heel weinig epochs nodig. Waarom?

Experimenteer met een 1-dimensionale en een 3-dimensionale dataset.

4.3 Complexe dataset

Genereer een set vectoren met een hoge dimensie, waarbij de waarden niet kunstmatig zijn, maar bijvoorbeeld statistische eigenschappen representeren.

Bekijk deze set vectoren met `vec2ps` of `vecview`. Kun je enige conclusies trekken uit wat je ziet?

Pas merge clustering toe en bekijk het resultaat. Welke conclusies over de dataset zijn er te trekken?

Genereer een Kohonen-kaart. Tot welke inzichten leidt dit?

Suggestie voor verkrijgen van een dataset: Neem een woordenlijst met minimaal enkele duizenden (verschillende) woorden. Genereer een vector voor elke letter uit het alfabet. (Voor het Nederlands: *ij* is één letter, *y* is een andere letter.) Tel voor elke letter het aantal keer dat deze aan het begin van het woord voor komt, hoe vaak hij aan het eind staat, hoe vaak hij vooraf gegaan wordt door een *a*, hoe vaak hij gevolgd wordt door een *a*, hoe vaak hij vooraf gegaan wordt door een *b*, enzovoort. Dit levert vectoren van 54 waarden. Deel per vector elke waarde door het aantal keer dat de letter van die vector voorkomt in de woordenlijst, om vectoren te krijgen met waarden van vergelijkbare grootte. Een vlotte programmeur kan in een kwartier een programma schrijven dat zo'n reeks vectoren automatisch genereert aan de hand van een woordenlijst.

Wanneer je zo'n set vectoren gebruikt beantwoord dan de volgende vragen: Worden klinkers en medeklinkers gescheiden in aparte clusters? Worden minimale paren zoals `t` en `d` of `p` en `b` dicht bij elkaar geplaatst? Waarom wel/niet? Welke medeklinkers vertonen veel overeenkomst in het gebruik en waarom?